eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

IKASTENBOT III

Alumno: Abad, Grande, Mikel

Director: Pereira, Varela, Juanan

Curso: 2020-2021

Fecha: Bilbao, 20, Julio, 2021

Resumen

Castellano

En este proyecto se ha reimplementado desde cero, utilizando parte de la lógica de una versión anterior, un bot de Telegram en el lenguaje Node.js mediante el framework Telegraf. El bot ofrece múltiples funcionalidades a alumnos y tutores para facilitar el desarrollo de un Trabajo de Fin de Grado.

Se han realizado todas las fases que requiere un proyecto de software, estudiando las necesidades, diseñando soluciones válidas e implementando, probando y documentando todo el proceso.

Euskara

Proiektu honetan Node.js hizkuntzan Telegram bot bat hutsetik berinplementatu da, Telegraf framework-a erabiliz eta aurreko bertsio baten logikaren zati bat berrerabiliz. Bot-ak hainbat funtzionalitate eskaintzen ditu ikasle eta irakasleei Gradu Amaierako Lanaren garapena errazteko.

Software proiektu batek eskatzen dituen fase guztiak burutu dira, beharrak aztertuz, baliozko irtenbideak diseinatuz. Prozesu osoa inplementatu da, funtzioanalitateak probatuz eta dokumentatuz.

English

This project has reimplemented a Telegram bot from scratch using Node.js language and the Telegraf framework. It reuses parts of the logic of a previous version. It offers multiple features to students and teachers to assist them in the development of a Final Degree Project.

All stages of a software development project have been accomplished, studying the needs, designing valid solutions and implementing, testing and documenting the entire process.

Prefacio

El proyecto IkastenBot surge originalmente con el objetivo de facilitar a los alumnos y tutores la elaboración de los Trabajos de Fin de Grado. Se decide crearlo en forma de *chatbot* para la aplicación de mensajería instantánea Telegram, dada la popularidad y accesibilidad de la misma. De este modo, cualquier estudiante puede hacer uso de IkastenBot con su teléfono móvil u ordenador personal y una aplicación gratuita, como es Telegram.

El presente trabajo es la tercera fase del proyecto, donde se decide junto al tutor rehacer la aplicación en un lenguaje más moderno y sostenible de cara al futuro. Empezando el código desde cero, pero utilizando parte de la lógica ya existente, se abandona el uso de PHP como lenguaje principal para crear el nuevo código en Node.js. Para ello, se utilizará un *framework* especialmente diseñado para explotar la Bot API de Telegram: Telegraf[1].

[1] https://telegraf.js.org/

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Un chatbot, de aquí en adelante bot, es una aplicación que mantiene una conversación con el usuario mediante texto, simulando una conversación con una persona real. Su popularidad se ha disparado en los últimos años por las múltiples ventajas que ofrecen, tanto a las empresas que los utilizan como a los mismos usuarios, estimándose que en 2021 el 85 % de las interacciones con clientes serán tratadas sin la intervención de un humano [2]. Entre estos beneficios destacan el poder reducir drásticamente el coste de atención al cliente, lo que en el ámbito académico implica reducir la dedicación del profesorado, así como la capacidad de atender a varios clientes de manera simultanea, o en este caso alumnos.

IkastenBot está desarrollado específicamente para Telegram, una de las aplicaciones de mensajería instantánea más utilizada del momento. En enero de 2021 fue la aplicación más descargada entre iOS y Android, y contaba con 500 millones de usuarios activos, un incremento del 150 % desde 2018 [3].

El objetivo de este trabajo es reconstruir IkastenBot en Node.js, un lenguaje más actual que el anterior, con las funciones que el tutor del proyecto considera esenciales. En caso de poder ampliarse se implementarían funcionalidades nuevas o se mejorarían, en la medida de lo posible, las ya existentes.

1.1. Origen del proyecto

El proyecto surge como una idea entre Mikel Abad, alumno y autor del presente TFG, y Juanan Pereira, profesor en la UPV[1] (Departamento de Lenguajes y Sistemas Informáticos).

En un inicio el proyecto consistiría en solucionar algunos problemas existentes en el bot y ampliar algunas de sus funcionalidades. Tras un inicio lento debido a una escasa motivación por parte del alumno, debida a la complicación del lenguaje y la dependencia de código anterior, se decide realizar el desarrollo partiendo de cero en un nuevo lenguaje.

De este modo, el nuevo objetivo sería desarrollar la aplicación IkastenBot como si de un nuevo proyecto se tratase, llegando a implementar todas las funcionalidades esenciales y alguna complementaria.

1.2. Descripción y situación del proyecto

El presente proyecto consiste en la elaboración de un bot de Telegram que sirva de ayuda tanto para alumnos, facilitándoles múltiples herramientas en las que apoyarse, como para tutores, permitiendo realizar de manera sencilla el seguimiento de diferentes trabajos.

Actualmente existe una versión de IkastenBot que no se encuentra abierta al público en un entorno de producción. Dado que no se ha dado soporte a la aplicación en un tiempo, algunas de sus funcionalidades se encuentran operativas, pero otras contienen errores y requieren de un mantenimiento.

Es por este motivo que, antes de comenzar una restauración, se valora y finalmente acepta la posibilidad de comenzar de cero con una nueva tecnología. Tras realizar una comparación entre PHP y Node.js, se decide continuar con la segunda, puesto que es más apropiada y eficiente para aplicaciones con un gran volumen de entradas y salidas [4], como es el caso de los chatbots.

[1] Universidad del País Vasco

1.3. Motivaciones para la elección del proyecto

Desde un punto de vista personal, las motivaciones que me llevaron a la elección y desarrollo de este proyecto, como alumno de IIGSI[2] y profesional de la industria, fueron principalmente las siguientes.

Interés en las tecnologías implicadas

En un inicio, el simple hecho de desarrollar un bot para Telegram ya me parecía una opción interesante. En mi vida personal utilizo algunos de estos bot a diario, y también he desarrollado algunos muy simples para otras plataformas, nada comparables al proyecto actual.

Tras comenzar con el análisis inicial y sentir cierto descontento con las tecnologías utilizadas en las versiones anteriores, Juanan y yo acordamos comenzar el desarrollo de cero en un nuevo lenguaje más atractivo: Node.js. Con este nuevo rumbo del proyecto mi motivación aumentó.

Utilidad del proyecto

Bajo mi punto de vista, lo más importante a la hora de comenzar un proyecto es que tenga alguna utilidad. En el caso de IkastenBot la utilidad es clara: alumnos de la carrera de informática, y probablemente otras en el futuro, podrán utilizarlo como herramienta de apoyo en la realización de sus TFG.

A la hora de realizar el trabajo aquí expuesto, IkastenBot es una herramienta que realmente me hubiera gustado tener a mi disposición y que ofrece beneficios reales. Por ello me motivó escoger un proyecto que pudiera ayudar a otros alumnos en la misma situación que yo.

[2] Ingeniería Informática de Gestión y Sistema de Información.

2. Planteamiento Inicial

En esta sección se presentan, en este orden, los objetivos del trabajo así como las herramientas utilizadas para desarrollarlo, el alcance de éste junto con su correspondiente planificación temporal y la evaluación económica y de riesgos.

2.1. Objetivos

El objetivo principal de este TFG podría definirse como: rehacer tanto la aplicación como la base de datos de IkastenBot para obtener un bot funcional y, desde un punto de vista técnico, más comprensible y sostenible que el actual. Para poder llegar a dicho resultado, se definen los siguientes objetivos.

2.1.1. Objetivos principales

- **Rediseño de la Base de Datos**, ya que la actual está fuertemente vinculada al framework utilizado.
- **Rediseño de la aplicación**, dado que son tecnologías diferentes y además debe adaptarse a la nueva Base de Datos.
- **Implementar las funcionalidades** que se consideran de mayor importancia:
  - Consulta de información relacionada con los TFG mediante un listado de preguntas frecuentes.
  - Registro de un TFG en el sistema por parte del alumno con posterior validación por parte del tutor.
  - Análisis y desglose, así como almacenamiento en base de datos, de los diferentes diagramas de Gantt que el alumno aporte en formato gan.
  - Creación y envío de notificaciones periódicas basadas en los hitos marcados por el alumno en su planificación.
- **Satisfacer las necesidades del cliente**, en este caso el tutor del TFG y responsable de IkastenBot.
- Terminar el proyecto en el plazo temporal establecido.

2.1.2. Objetivos secundarios

- Implementar algunas de las funcionalidades secundarias:
  - Soporte multidioma para español, euskera e inglés.
  - Creación de nuevos ítems en el FAQ, funcionalidad limitada a los tutores.
  - Corrección ortográfica y gramatical de la memoria mediante un módulo externo.
  - Análisis de la memoria para detectar posibles plagios, disponible únicamente para los tutores.

2.2. Herramientas seleccionadas

En esta sección se presentan las herramientas que se van a utilizar a lo largo del proyecto, tanto para su desarrollo técnico como la elaboración de la memoria, gestión de recursos, planificación temporal, etc.

De cara a escoger las herramientas se valorarán, sin seguir un orden específico, los siguientes criterios:

- **Estado actual**: se utilizarán únicamente herramientas actuales, con presencia en el mercado y de uso generalizado, así como aplicaciones que reciban actualizaciones y soporte de manera continuada.
- **Experiencia**: partiendo del punto anterior, se valorará de manera positiva el conocer y haber utilizado previamente las herramientas, ahorrando así tiempo de formación y familiarización.
- **Accesibilidad**: todas las herramientas utilizadas serán de uso libre o con licencias obtenidas por la universidad.

Siguiendo dichos criterios, las herramientas a utilizar son las siguientes:

- **Node.js**: Es un entorno JavaScript back-end en tiempo de ejecución, de código abierto y multiplataforma.
- **Visual Studio Code**: Es un editor de código desarrollado por Microsoft, publicado bajo licencia MIT, que dispone de múltiples herramientas para el desarrollo de aplicaciones. Será una de las herramientas principales a la hora de desarrollar el proyecto.
- **Git**: Sistema de gestión de versiones gratuito y de código abierto, desarrollado por Linus Torvalds. Se utilizará tanto de manera local como en conjunto con la siguiente herramienta.
- **GitHub**: Es un proveedor de hosting para el desarrollo de software que, además de ofrecer la utilización de Git como sistema de versiones, tiene múltiples herramientas tanto para desarrolladores individuales como equipos. Se utilizará para alojar el código de la aplicación en un repositorio privado.
- **MySQL**: Sistema de gestión de bases de datos relacionales, de código abierto. Se utilizará para desarrollar la base de datos del proyecto.
- **MySQL Workbench**: Herramienta con interfaz gráfica que permite llevar a cabo diseño, creación, desarrollo, gestión, administración y mantenimiento de bases de datos en MySQL, tanto en local como en remoto.

- **LaTeX**: Es un sistema de composición de textos donde el autor utiliza texto plano para dar el formato al resultado final.
- **Overleaf**: Herramienta en la nube de carácter colaborativo que permite escribir, editar y publicar documentos basados en el lenguaje LaTeX.
- **Cacoo**: Herramienta utilizada para la creación de los diagramas EDT y sus respectivas subsecciones.
- **Visual Paradigm**: Herramienta utilizada para la creación y exportación de los diagramas propios del desarrollo del software utilizados a lo largo del proyecto.
- **PHP**: Lenguaje de programación de propósito general creado originalmente por Rasmus Lerdorf, adaptado especialmente al desarrollo web. Se utilizará únicamente para probar desarrollos anteriores.
- **GanttProject**: Herramienta para la gestión de proyectos mediante la creación de diagramas de Gantt.
- **ngrok**: Herramienta que permite exponer públicamente una URL[1] segura, generada dinámicamente, que apunta a un servicio web alojado en nuestro ordenador. Se utilizará durante el desarrollo[2] ya que Telegram necesita que las conexiones con el webhook indicado sean seguras.

[1] Uniform Resource Locator

[2] No será necesaria una vez el bot sea puesto en funcionamiento en un entorno real, pues se alojará en un servidor con dirección IP pública y certificado SSL.

2.3. Arquitectura

En esta sección se presenta la arquitectura del proyecto, para de esta manera poder entender los sistemas implicados y la conexión entre los mismos.

La estructura del proyecto sigue el modelo Cliente-Servidor. En nuestro caso, el cliente sería la aplicación de Telegram y el chat abierto con nuestro bot, todo desde el dispositivo del usuario. Como se puede observar en la figura 2.1, la conexión se realiza siempre a través de la nube de Telegram, quién procesa y dirige los mensajes. Estos llegan al servidor donde son interpretados y utilizados acorde a nuestras necesidades. La aplicación a su vez se conecta con la base de datos para almacenar toda la información que sea necesaria y las tareas programadas del SO para las notificaciones.

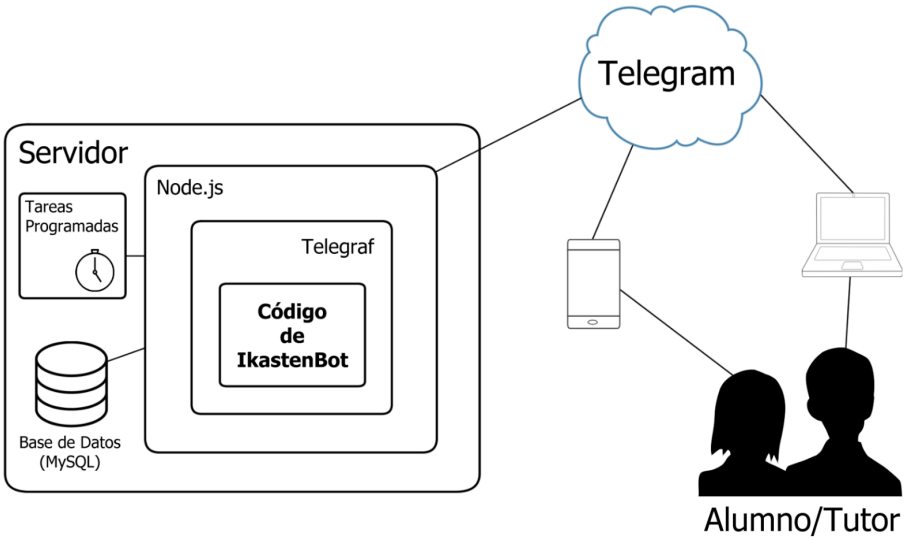

Figura 2.1: Arquitectura general de IkastenBot

2.4. Alcance

El alcance del proyecto viene definido por el diagrama EDT que se puede observar en la figura 2.2. En él se refleja la distribución de las tareas en bloques generales, no necesariamente en orden de realización, que serían los siguientes: Gestión y planificación, estudio previo, formación, captura de requisitos, análisis, diseño, implementación y verificación y pruebas.

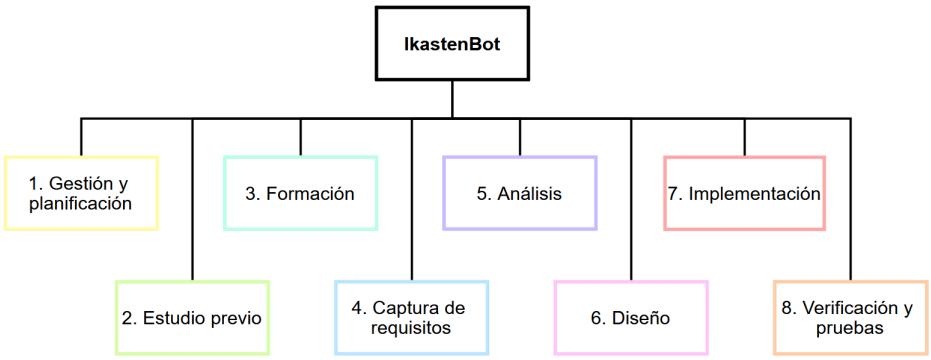

Figura 2.2: Diagrama EDT de IkastenBot

1. **Gestión y planificación**: En este bloque se definen las tareas a realizar para llevar a cabo el proyecto correctamente.

2. **Estudio previo**: Este bloque agrupa las tareas requeridas para comprender y probar el bot anterior ya existente.

3. **Formación**: Bloque que define el aprendizaje de las herramientas escogidas para las que no se disponga de conocimientos previos.

4. **Captura de requisitos**: Bloque que contiene las tareas para recoger los requisitos del cliente, de cara a identificar sus necesidades y poder llegar a la mejor solución.

5. **Análisis**: Este bloque reúne las tareas para analizar los requisitos, así como elegir las herramientas y tecnologías adecuadas para cumplimentarlos.

6. **Diseño**: En este bloque se encuentran las tareas identificadas para definir la solución que se va a llevar a cabo, de cara a cumplir con los requisitos.

7. **Implementación**: Bloque que recoge las tareas sobre el desarrollo de las soluciones definidas.

8. **Verificación y pruebas**: En este bloque se han definido las tareas para comprobar que el resultado es el esperado, que funciona correctamente y que el cliente está satisfecho.

Además, cada uno de estos bloques es desglosado para mostrar de manera atómica las tareas que lo componen.

2.4.1. Gestión y planificación

En este apartado se presentan las tareas identificadas para definir, planificar, documentar y defender el proyecto.

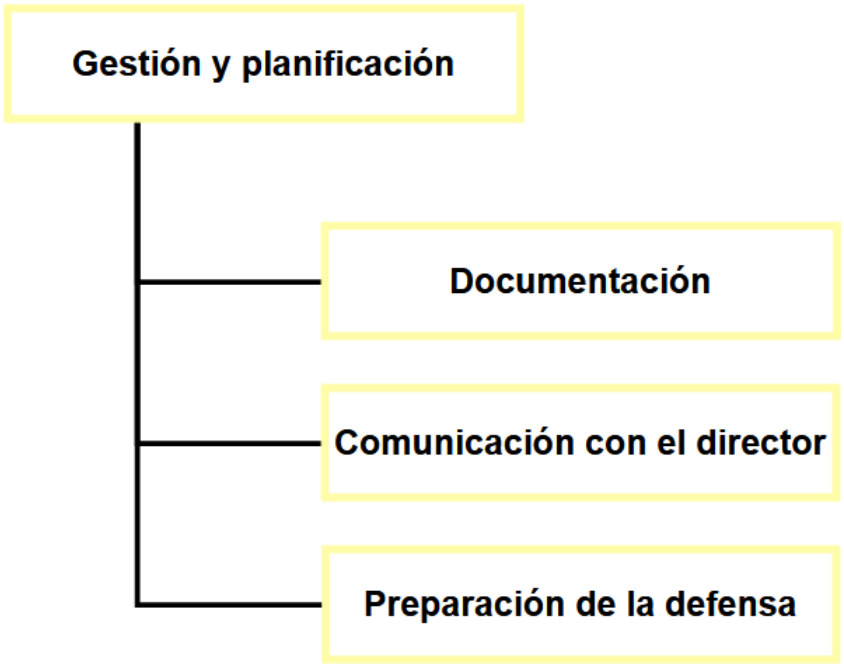

Figura 2.3: Diagrama EDT referente al bloque de Gestión y planificación

En la figura 2.3 podemos ver la sección del EDT correspondiente a la gestión y planificación. A continuación se muestra detalladamente cada tarea.

Documentación
Paquete de trabajo: Gestión y planificación. **Duración estimada**: 180 horas.
Descripción: Elaboración de la memoria a entregar una vez finalizado el proyecto. **Salidas/Entregables**: Memoria. **Recursos necesarios**: Ordenador con conexión a internet y Overleaf.

Tabla 2.1: Documentación

Comunicación con el director
Paquete de trabajo: Gestión y planificación. **Duración estimada**: 10 horas.
Descripción: Comunicación con el director/cliente del proyecto para realizar seguimientos, aclarar dudas o presentar entregas. Pueden ser reuniones telemáticas o presenciales. **Salidas/Entregables**: Anotaciones tomadas durante la reunión. **Recursos necesarios**: Ordenador con conexión a internet y Skype o cuaderno de notas.

Tabla 2.2: Comunicación con el director

Preparación de la defensa
Paquete de trabajo: Gestión y planificación. **Duración estimada**: 15 horas.
Descripción: Preparación de las diapositivas para la defensa así como ensayos de la misma. **Salidas/Entregables**: Presentación en formato PPT, PDF y ODP. **Recursos necesarios**: Ordenador, Google Slides, proyecto y memoria.

Tabla 2.3: Preparación de la defensa

2.4.2. Estudio previo

Este bloque consiste de las diferentes tareas a llevar a cabo para comprender la situación actual del proyecto, poner a prueba el código anterior y adquirir todos los conocimientos necesarios sobre éste para poder reconstruirlo.

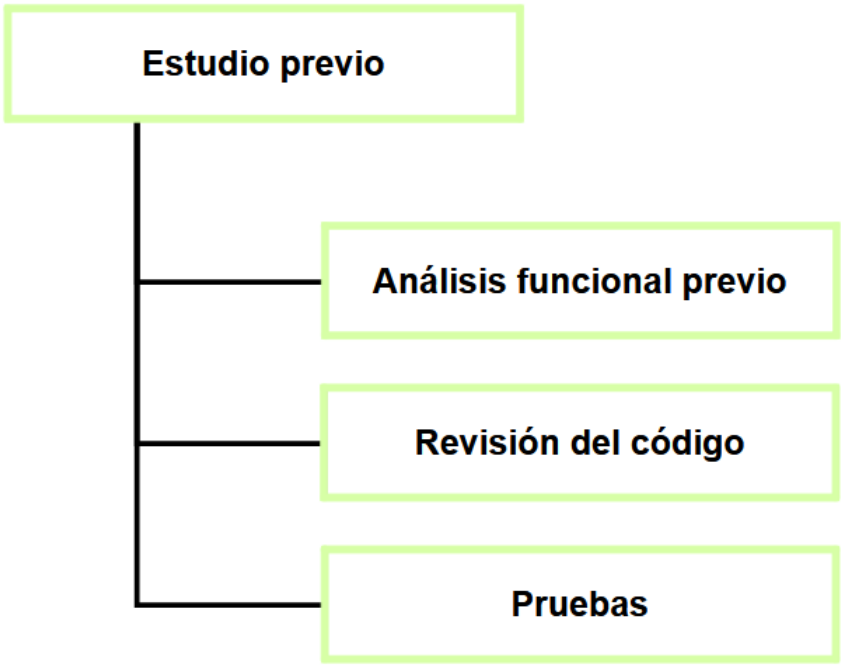

Figura 2.4: Diagrama EDT referente al bloque de Estudio previo

En la figura 2.4 podemos observar la sección del EDT correspondiente al estudio previo. A continuación se muestra detalladamente cada tarea.

Análisis funcional previo
Paquete de trabajo: Estudio previo. **Duración estimada**: 15 horas.
Descripción: Estudio de memorias anteriores implicadas en el proyecto. **Salidas/Entregables**: Anotaciones personales y posibles dudas. **Recursos necesarios**: Ordenador y memorias anteriores.

Tabla 2.4: Análisis funcional previo

Revisión del código
Paquete de trabajo: Estudio previo. **Duración estimada**: 6 horas.
Descripción: Estudio del código existente del proyecto anterior de cara a identificar y comprender los flujos técnicos de la implementación. **Salidas/Entregables**: Anotaciones personales y posibles dudas. **Recursos necesarios**: Ordenador y proyecto anterior.

Tabla 2.5: Revisión del código

Pruebas
Paquete de trabajo: Estudio previo. **Duración estimada**: 6 horas.
Descripción: Ejecución del código anterior y pruebas de sus funcionalidades mediante la plataforma de Telegram. **Salidas/Entregables**: Anotaciones personales y posibles dudas. **Recursos necesarios**: Ordenador y proyecto anterior.

Tabla 2.6: Pruebas

2.4.3. Formación

Este bloque define las tareas a llevar a cabo para realizar un estudio básico de las herramientas y tecnologías escogidas hasta un nivel suficiente como para comenzar a trabajar con ellas.

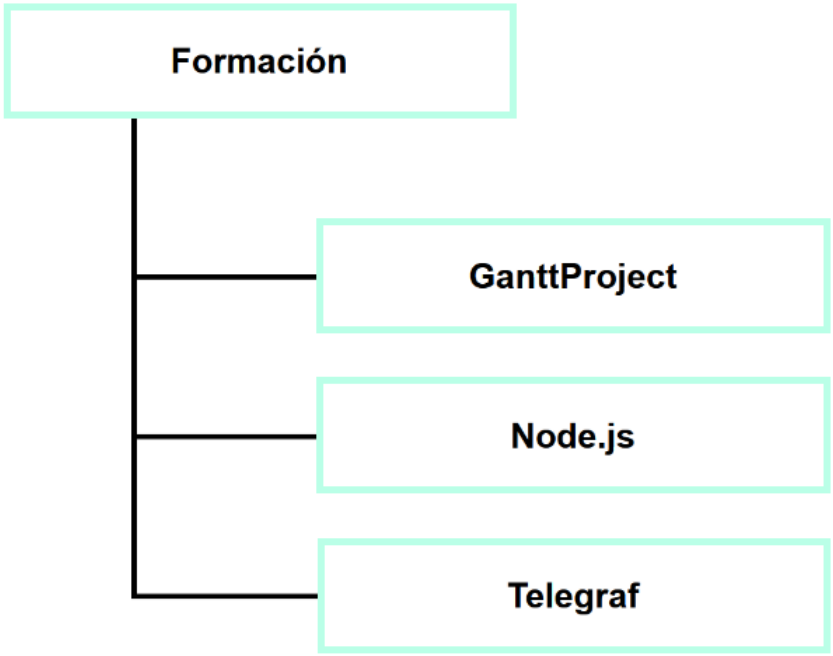

Figura 2.5: Diagrama EDT referente al bloque de Formación

La figura 2.5 nos presenta la sección del EDT correspondiente a la formación. Este bloque se descompone en una tarea por cada tecnología que requiere un estudio inicial, detalladas en las siguientes tablas.

GanttProject
Paquete de trabajo: Formación. **Duración estimada**: 2 horas
Descripción: Familiarización con la interfaz de GanttProject así como un análisis superficial del formato en el que se exportan los diagramas. **Recursos necesarios**: Ordenador y GanttProject.

Tabla 2.7: GanttProject

Node.js
Paquete de trabajo: Formación. **Duración estimada**: 3 horas
Descripción: Recordar y ponerse al día con la tecnología, pues ya se disponen de conocimientos previos. **Recursos necesarios**: Ordenador con acceso a internet, Node.js y Visual Studio Code.

Tabla 2.8: Node.js

Telegraf
Paquete de trabajo: Formación. **Duración estimada**: 3 horas
Descripción: Estudio básico inicial del framework en la página oficial, de cara a identificar las herramientas que pone a disposición de los integradores. **Recursos necesarios**: Ordenador con acceso a internet.

Tabla 2.9: Telegraf

2.4.4. Captura de requisitos

En este bloque se explican las tareas que se van a realizar para llevar a cabo la captura de requisitos del proyecto. Estos deben definir los requisitos que ha de cumplir nuestra aplicación.

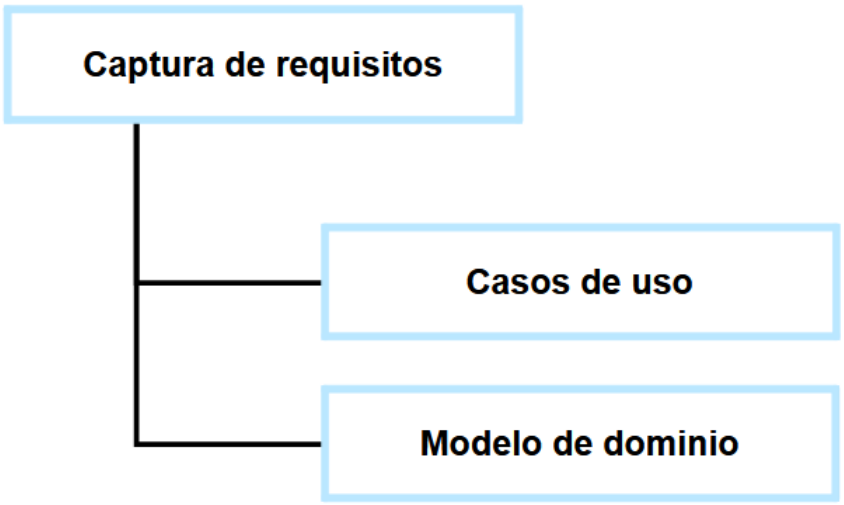

Figura 2.6: Diagrama EDT referente al bloque de Captura de requisitos

La figura 2.6 muestra la sección del EDT correspondiente a la captura de requisitos. Cada una de las tareas definidas es explicada individualmente a continuación.

Casos de uso
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 10 horas.
Descripción: Identificación y definición de los casos de uso referentes a IkastenBot. **Salidas/Entregables**: Diagramas de casos de uso. **Recursos necesarios**: Ordenador con acceso a internet y Visual Paradigm.

Tabla 2.10: Casos de uso

Modelo de dominio
Paquete de trabajo: Captura de requisitos. **Duración estimada**: 15 horas.
Descripción: Elaboración del modelo de dominio que define las diferentes entidades así como las relaciones entre las mismas. **Salidas/Entregables**: Modelo de dominio. **Recursos necesarios**: Ordenador con acceso a internet y Visual Paradigm.

Tabla 2.11: Modelo de dominio

2.4.5. Análisis

Este bloque define todas las tareas necesarias para realizar un análisis del proyecto así como la elección de las herramientas y tecnologías apropiadas para su desarrollo. Se estudiarán a fondo los requisitos del proyecto y se valorarán varias alternativas para cada necesidad.

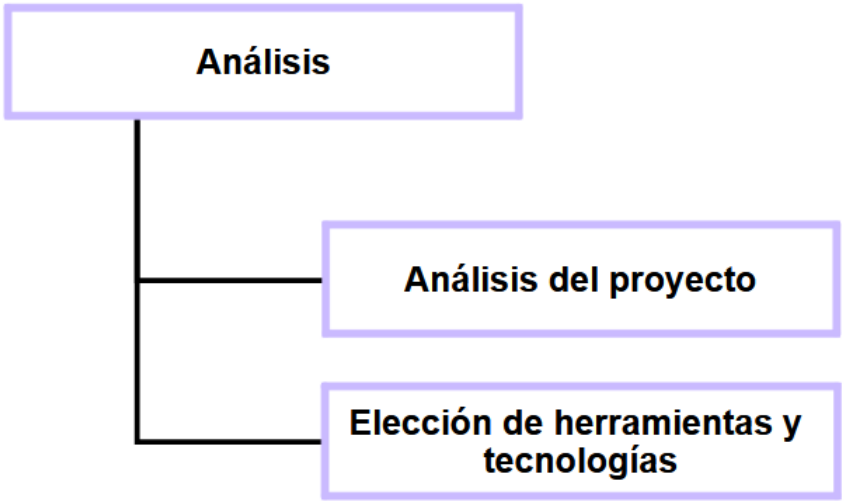

Figura 2.7: Diagrama EDT referente al bloque de Análisis

La figura 2.7 muestra la sección del EDT correspondiente al análisis. Las tareas que conforman este bloque vienen definidas de manera atómica en la siguientes tablas.

Análisis del proyecto
Paquete de trabajo: Análisis. **Duración estimada**: 20 horas.
Descripción: Análisis del proyecto así como de los requisitos identificados previamente. **Salidas/Entregables**: Anotaciones. **Recursos necesarios**: Ordenador, anotaciones previas, casos de uso y modelo de dominio.

Tabla 2.12: Análisis del proyecto

Elección de herramientas y tecnologías
Paquete de trabajo: Análisis. **Duración estimada**: 10 horas.
Descripción: Elección de las herramientas y tecnologías a utilizar en función de las necesidades identificadas. **Recursos necesarios**: Ordenador con acceso a internet.

Tabla 2.13: Elección de herramientas y tecnologías

2.4.6. Diseño

En este bloque vienen descritas todas las tareas necesarias para realizar correctamente el diseño que definirá la solución. Principalmente mediante diagramas, se definirán todos los detalles necesarios para posteriormente guiar la implementación.

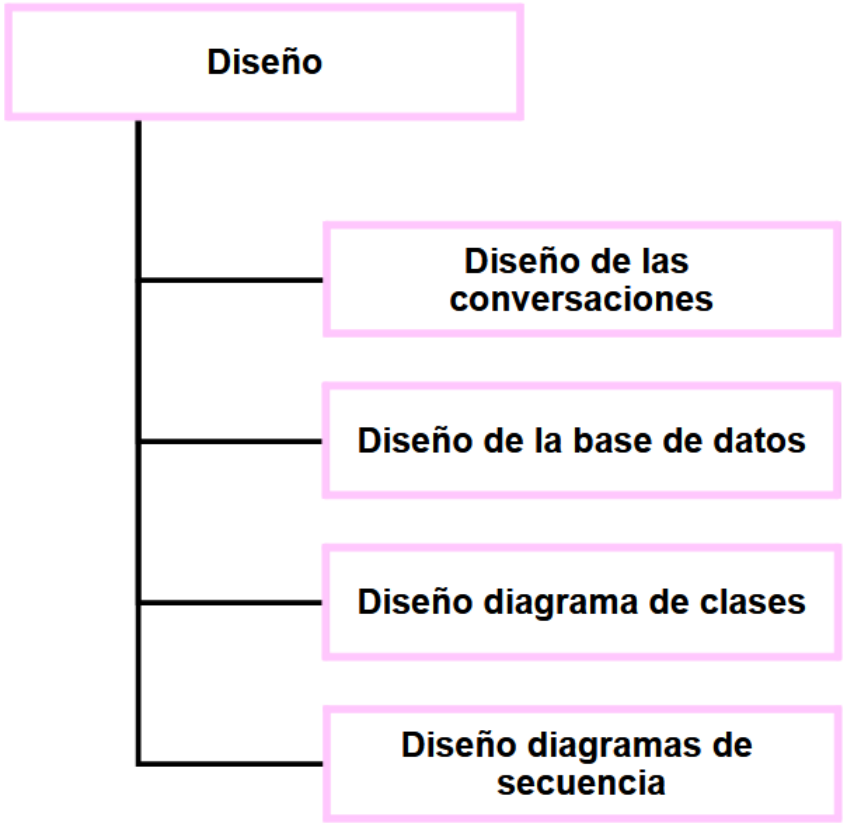

Figura 2.8: Diagrama EDT referente al bloque de Diseño

La figura 2.8 se corresponde con la sección del EDT referente al diseño. Las tareas correspondientes a este bloque son definidas y detalladas en la siguientes tablas.

Diseño de las conversaciones
Paquete de trabajo: Diseño. **Duración estimada**: 10 horas.
Descripción: Diseño de las conversaciones que podrá mantener el bot con los usuarios para los diferentes procesos que se implementarán. **Salidas/Entregables**: Mockups de las conversaciones. **Recursos necesarios**: Ordenador y Telegram.

Tabla 2.14: Diseño de las conversaciones

Diseño de la base de datos
Paquete de trabajo: Diseño. **Duración estimada**: 15 horas.
Descripción: Rediseño de la base de datos en función de los requisitos y las necesidades respecto a la información a almacenar. **Salidas/Entregables**: DER (Diagrama Entidad-Relación). **Recursos necesarios**: Ordenador con acceso a internet y Visual Paradigm.

Tabla 2.15: Diseño de la base de datos

Diseño diagrama de clases
Paquete de trabajo: Diseño. **Duración estimada**: 15 horas.
Descripción: Utilizando como referencia el modelo de dominio definido en la tabla 2.11, definición del diagrama de clases representando éstas y todos sus elementos. **Salidas/Entregables**: Diagrama de clases. **Recursos necesarios**: Ordenador con acceso a internet y Visual Paradigm.

Tabla 2.16: Diseño diagrama de clases

Diseño diagramas de secuencia
Paquete de trabajo: Diseño. **Duración estimada**: 20 horas.
Descripción: Realización de los diagramas de secuencia definiendo los flujos principales que conforman la interacción con IkastenBot. **Salidas/Entregables**: Diagramas de secuencia. **Recursos necesarios**: Ordenador con acceso a internet y Visual Paradigm.

Tabla 2.17: Diseño diagramas de secuencia

2.4.7. Implementación

Tras realizar un diseño apropiado el siguiente paso es uno de los más importantes y costosos del proyecto: realizar la implementación de nuestra aplicación con toda la información trabajada hasta el momento.

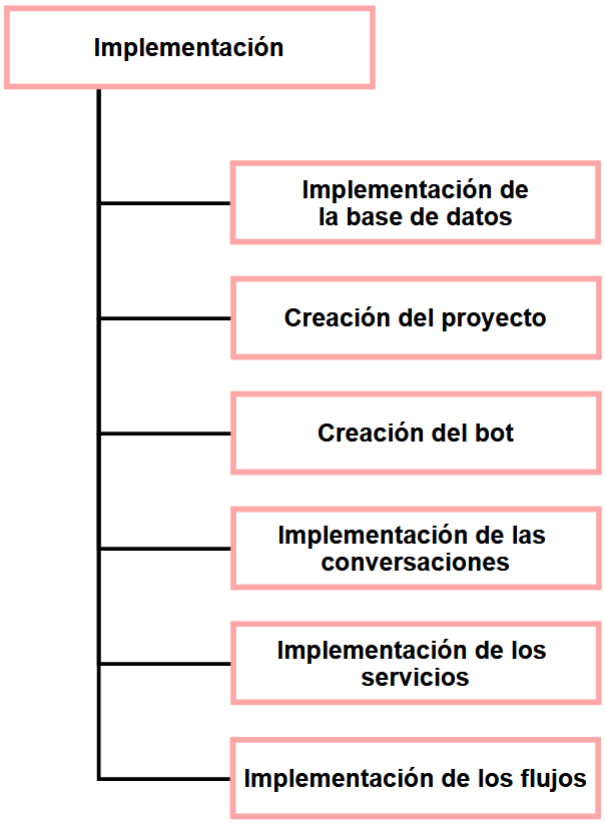

Figura 2.9: Diagrama EDT referente al bloque de Implementación

En la figura 2.9 podemos ver la sección del EDT referente a la implementación. A continuación se presentan las tablas que contienen todos los detalles sobre dichas tareas.

Implementación de la base de datos
Paquete de trabajo: Implementación. **Duración estimada**: 12 horas.
Descripción: Creación de la base de datos con todas las tablas y relaciones necesarias para su funcionamiento, definidas en el EDT diseñado previamente. **Salidas/Entregables**: Base de datos. **Recursos necesarios**: Ordenador, MySQL y MySQL Workbench.

Tabla 2.18: Implementación de la base de datos

Creación del proyecto
Paquete de trabajo: Implementación. **Duración estimada**: 5 horas.
Descripción: Inicialización del proyecto en Node.js con una estructura básica. **Salidas/Entregables**: Código inicial del proyecto. **Recursos necesarios**: Ordenador, Visual Studio Code y Node.js.

Tabla 2.19: Creación del proyecto

Creación del bot
Paquete de trabajo: Implementación. **Duración estimada**: 1 hora.
Descripción: Creación del bot en la plataforma de Telegram para obtener el token de acceso a la API y poder comunicar el código con la plataforma. **Salidas/Entregables**: Token de acceso. **Recursos necesarios**: Ordenador o smartphone y Telegram.

Tabla 2.20: Creación del bot

Implementación de las conversaciones
Paquete de trabajo: Implementación. **Duración estimada**: 25 horas.
Descripción: Implementación de la parte del código encargada de detectar los comandos y eventos que reconoce el bot, así como la interacción con el usuario mediante respuestas. **Salidas/Entregables**: Parte del código referente a la interacción con el usuario. **Recursos necesarios**: Ordenador, Visual Studio Code, Node.js y Telegram.

Tabla 2.21: Implementación de las conversaciones

Implementación de los servicios
Paquete de trabajo: Implementación. **Duración estimada**: 45 horas.
Descripción: Implementación de los servicios que se utilizarán en múltiples puntos de la aplicación, como por ejemplo la conexión con la base de datos o la gestión de la sesión. **Salidas/Entregables**: Parte del código referente a los servicios. **Recursos necesarios**: Ordenador, Visual Studio Code, Node.js y Telegram.

Tabla 2.22: Implementación de los servicios

Implementación de los flujos
Paquete de trabajo: Implementación. **Duración estimada**: 80 horas.
Descripción: Implementación de los comandos así como los flujos que vinculan los eventos detectados, los servicios y la propia ejecución de los comandos. **Salidas/Entregables**: Parte del código referente al funcionamiento del bot. **Recursos necesarios**: Ordenador, Visual Studio Code, Node.js y Telegram.

Tabla 2.23: Implementación de los flujos

2.4.8. Verificación y pruebas

A medida que se finalizan partes de la implementación es de vital importancia realizar las pruebas pertinentes, de cara a validar que el funcionamiento es correcto y se corresponde con el esperado por el cliente.

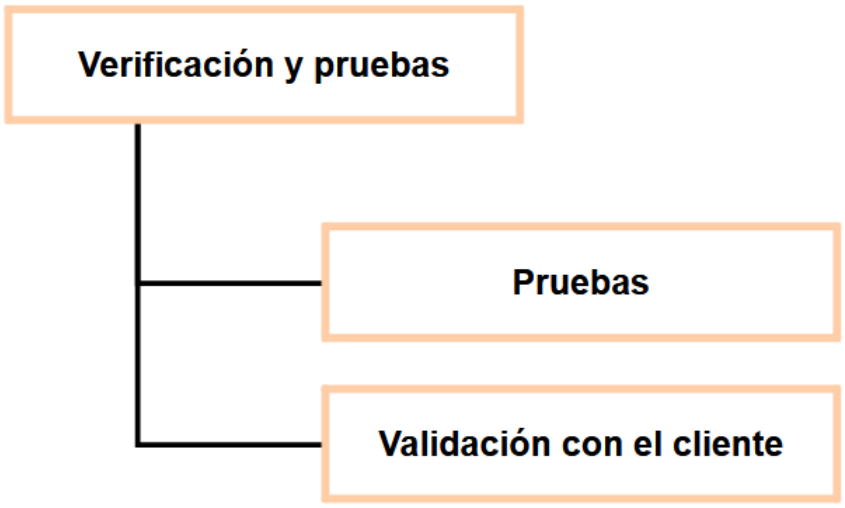

Figura 2.10: Diagrama EDT referente al bloque de Verificación y pruebas

La figura 2.10 muestra la sección del EDT referente a la verificación y pruebas. En las tablas presentadas a continuación se dan los detalles sobre cada una de las tareas implicadas en este bloque.

Pruebas
Paquete de trabajo: Verificación y pruebas. **Duración estimada**: 20 horas.
Descripción: Realización de pruebas a lo largo de todo el desarrollo para validar su correcto funcionamiento. **Recursos necesarios**: Ordenador, Visual Studio Code, MySQL, Telegram y ngrok.

Tabla 2.24: Pruebas

Validación con el cliente
Paquete de trabajo: Verificación y pruebas. **Duración estimada**: 5 horas.
Descripción: Mediante reuniones periódicas, validación con el cliente de que sus requisitos son cumplidos y el proyecto cumple con lo esperado. **Recursos necesarios**: Ordenador, Visual Studio Code, MySQL, Telegram y ngrok.

Tabla 2.25: Validación con el cliente

2.4.9. Resumen planificación

Para finalizar esta sección se resume en la siguiente tabla las tareas definidas en todos los apartados previos, así como el tiempo total estimado. Estos serán los datos utilizados para realizar la planificación temporal.

Tarea	Duración estimada
1. Gestión y planificación	**205 horas**
1.1 Documentación	180 horas
1.2 Comunicación con el director	10 horas
1.3 Preparación de la defensa	15 horas
2. Estudio previo	**27 horas**
2.1 Análisis funcional previo	15 horas
2.2 Revisión del código	6 horas
2.3 Pruebas	6 horas
3. Formación	**8 horas**
3.1 GanttProject	2 horas
3.2 Node.js	3 horas
3.3 Telegraf	3 horas
4. Captura de requisitos	**25 horas**
4.1 Casos de uso	10 horas
4.2 Modelo de dominio	15 horas
5. Análisis	**35 horas**
5.1 Análisis del proyecto	20 horas
5.2 Elección de herramientas y tecnologías	10 horas
6. Diseño	**60 horas**
6.1 Diseño de las conversaciones	10 horas
6.2 Diseño de la base de datos	15 horas
6.3 Diseño diagrama de clases	15 horas
6.4 Diseño diagramas de secuencia	20 horas
7. Implementación	**168 horas**
7.1 Implementación de la base de datos	12 horas
7.2 Creación del proyecto	5 horas
7.3 Creación del bot	1 horas
7.4 Implementación de las conversaciones	25 horas
7.5 Implementación de los servicios	45 horas
7.6 Implementación de los flujos	80 horas
8. Verificación y pruebas	**25 horas**
8.1 Pruebas	20 horas
8.2 Validación con el cliente	5 horas
TOTAL	**547 horas**

Figura 2.11: Tareas identificadas y su duración estimada

Como podemos observar en la tabla 2.11 la duración total estimada para el proyecto completo es de 547 horas.

2.5. Planificación temporal

Teniendo en cuenta el desglose de tareas realizado y presentado, así como las estimaciones temporales relacionadas a éstas, es necesario identificar las dependencias entre ellas y planificar cómo se llevarán a cabo.

Para realizar esta planificación, dado que en el momento de comenzar con el proyecto el alumno se encuentra trabajando por cuenta ajena a jornada completa, se plantea una jornada de trabajo de 20 horas semanales, 4 horas diarias de lunes a viernes. Tomando como referencia esta dedicación, así como la duración estimada de 547 horas (Figura 2.11), se calcula una duración de 27.35 semanas o 6.83 meses. Por lo tanto, comenzando el proyecto a día 14 de Septiembre de 2020, se marca como fecha objetivo para la finalización del proyecto el día 14 de Mayo de 2021, manteniendo así un mes de holgura con respecto a la finalización estimada.

Cabe destacar que con esta planificación se dispone de un mes adicional de holgura para entregar el proyecto en la convocatorio de Junio, o dos meses de holgura en caso de asistir a la convocatorio de Julio. De este modo, hay tiempo de sobra en caso de ser necesario lidiar con algún contratiempo. Además, algunas de las tareas podrán ser llevabas a cabo de manera paralela, ya que las dependencias definidas lo permiten.

A continuación se presenta el diagrama de Gantt que define la planificación temporal a llevar a cabo en el proyecto.

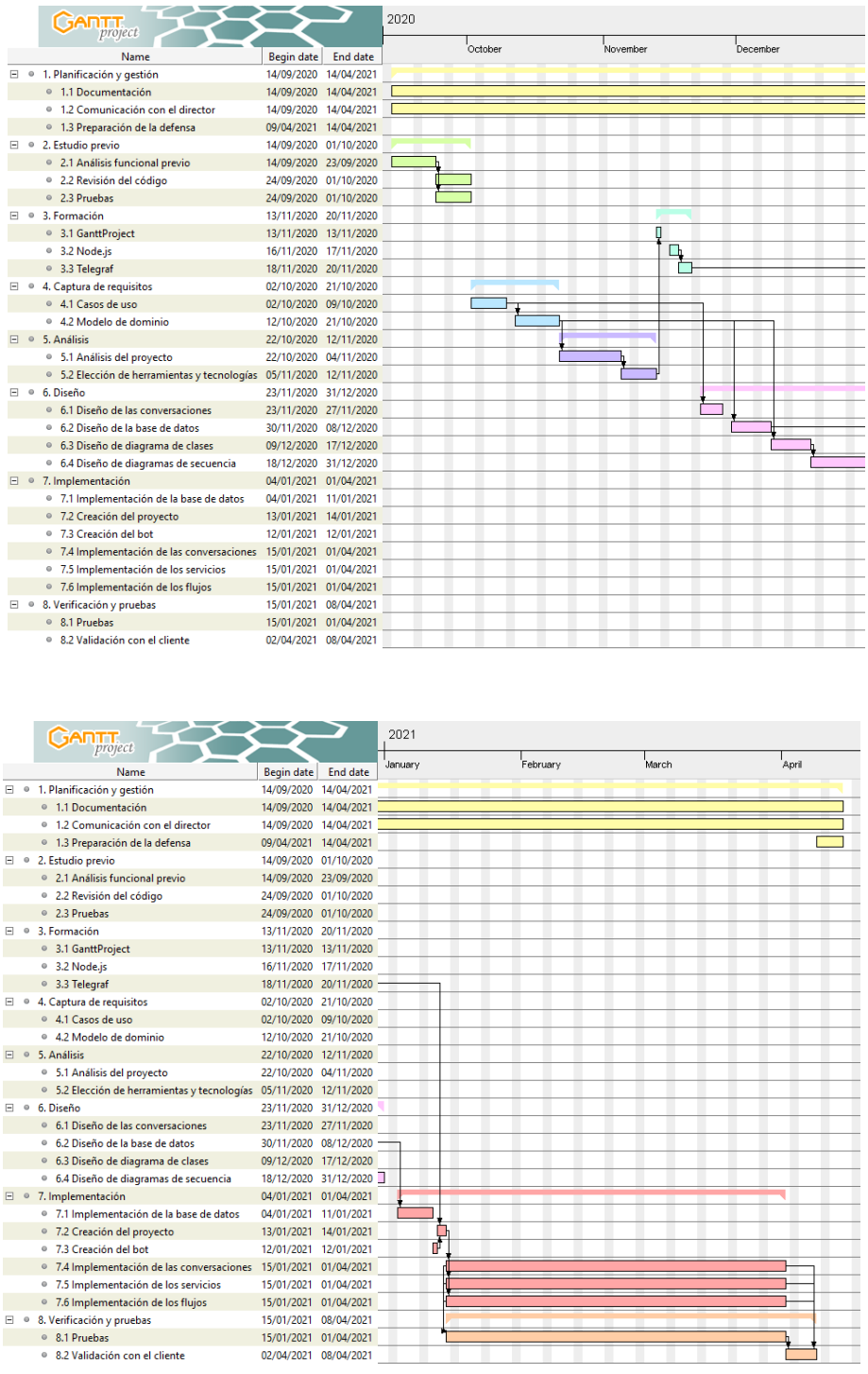

Figura 2.12: Planificación temporal del proyecto

2.6. Evaluación económica

Como en cualquier proyecto, es imprescindible realizar un presupuesto lo más veraz posible para determinar los gastos que se asumen al realizarlo. Si bien al tratarse de un trabajo de fin de grado no se pretende obtener ningún beneficio económico, la siguiente valoración servirá de referencia en caso de llegar a comercializarse.

2.6.1. Mano de obra

Dada la estricta similitud entre el proyecto que se está definiendo y el trabajo que lleva a cabo el alumno en su trabajo a jornada completa, se utilizará el propio sueldo neto como referencia: 20.152,00€/año. Esto significa un sueldo neto de 1.679€/mes (pagas extras incluidas). Partiendo de esta base, realizamos el cálculo del salario por hora con la siguiente fórmula:

$$Sueldo_{/hora} = \frac{Sueldo_{/mes}}{Horas\ trabajo_{/semana} \times Semanas_{/mes}} \tag{2.1}$$

Por tanto, asumiendo que un mes por lo general consta de 4 semanas y la jornada laboral consta de 8 horas diarias 5 días a la semana, resolvemos de la siguiente manera:

$$Sueldo_{/hora} = \frac{1679}{40 \times 4} = \frac{1679}{160} = 10,49 \tag{2.2}$$

Obtenemos como resultado 10,49€/hora.

Con estos datos podemos calcular el coste de mano obra derivado de nuestro proyecto:

$$Mano\ de\ obra = Horas\ estimadas \times Sueldo_{/hora} = 547 \times 10,49 = 5738,03 \tag{2.3}$$

Finalmente obtenemos que el gasto derivado directamente de la mano de obra asciende a la cantidad de 5738,03€.

2.6.2. Gastos de hardware

Al tratarse de un proyecto de desarrollo de software se deben asumir una serie de gastos derivados del equipamiento informático, descritos a continuación.

Ordenador

Para la realización de este proyecto se dispone de un ordenador portátil YOGA 510-14IKB de la marca Lenovo y modelo 80VB. Se estima una vida útil de 4 años y su precio es valorado en 635,59€, por lo que el cálculo de la amortización mensual se realiza mediante la siguiente fórmula:

$$Amortización_{mensual} = \frac{Precio}{Vida\ útil} = \frac{635,59€}{48\ meses} = 13,24€_{/mes} \tag{2.4}$$

De este modo, tomando el valor de la amortización mensual y la duración estimada del proyecto, se calcula el coste total de la siguiente forma:

$$Amortización_{total} = Amortización_{mes} \times Meses = 13,24 \times 6,8 = 90,03€ \tag{2.5}$$

Por lo tanto, el coste de la amortización del equipo es de 90,03€.

Alojamiento

El alojamiento de la aplicación se realizará en un servidor proporcionado por la universidad, por lo que no existen gastos derivados de éste para el desarrollo.

2.6.3. Gastos de software

Para realizar este proyecto se dio prioridad a herramientas de carácter libre y gratuito, por lo que el gasto en software y licencias será nulo. Para aquellas herramientas que requieran una licencia comercial se utilizará, a través de la universidad, una licencia de uso educativo, evitando así añadir costes al proyecto.

2.6.4. Gastos indirectos

Finalmente, es necesario añadir al total aquellos gastos que, si bien no están directamente relacionados con el proyecto, representan un gasto a tener en cuenta.

Luz

El proyecto se desarrollará en su totalidad utilizando el ordenador, por lo que a la hora de calcular el coste de luz se utilizará como referencia el total de horas estimadas hasta su finalización (véase sección 2.5).

Para poder realizar el cálculo es necesario conocer el precio del kilovatio por hora. En este caso particular, el precio contratado es de 0,17203€/kWh, y el coste total se calcula de la siguiente manera:

$$Coste\ luz = Tarifa_{hora} \times Horas = 0,17203 \times 547 = 94,10€ \quad (2.6)$$

Internet

Para el desarrollo de este proyecto se considera imprescindible la conexión ininterrumpida a Internet. Por tanto, el coste se calculará se la siguiente manera:

$$Coste\ internet = Tarifa_{mensual} \times Meses = 38,00 \times 6,8 = 258,40€ \quad (2.7)$$

Dado que la tarifa actual incluye tanto conexión de fibra óptica como la línea de teléfono móvil, no se añaden gastos relacionados con el uso de datos móviles en caso de resultar necesarios.

2.6.5. Gastos totales

A continuación se presenta una tabla que contiene el resumen de todos los gastos derivados de la realización del proyecto, cuya suma total asciende a 6.180,56€.

Concepto	***Gasto***
Mano de obra	5738,03€
Gastos de hardware	90,03€
Gastos de software	0€
Gastos indirectos	352,50€
Total	6.180,56€

Tabla 2.26: Resumen de gastos

2.7. Gestión de riesgos

A lo largo del desarrollo de un proyecto, especialmente si la duración es de varios meses como es el caso, es probable que surja algún problema que altere el curso del mismo. Para tratar de prevenir y evitar, o en última instancia reducir el impacto de estos, se debe crear el correspondiente plan de riesgos que identifique los problemas y defina cómo actuar ante los mismos.

En este apartado se detallan los riesgos identificados así como su plan de prevención, para tratar de evitar que dicho riesgo se cumpla, y de contingencia, para tratar de minimizar el impacto de éste en caso de suceder. Para cada uno de ellos se especifica la probabilidad de suceder así como el impacto que tendrían en la planificación temporal. Esta medida se ha definido en base a la pérdida de horas semanales de trabajo de la siguiente manera:

- **Impacto muy bajo**: Pérdida menor a 1 hora de trabajo.
- **Impacto bajo**: Pérdida entre 1 y 2 horas de trabajo.
- **Impacto medio**: Pérdida entre 2 y 3 horas de trabajo.
- **Impacto alto**: Pérdida de 4 horas (1 día) de trabajo.
- **Impacto muy alto**: Pérdida superior a 4 horas (1 día) de trabajo.

2.7.1. Exceso de horas en el trabajo

Partiendo de una situación laboral al inicio del proyecto de trabajo a jornada completa, cabe la posibilidad de que la empresa requiera de la realización de horas extra. Esto implicaría una reducción en las horas disponibles además de estrés y saturación derivado del exceso de trabajo.

Plan de prevención

- Seguimiento estricto de la planificación temporal.

Plan de contingencia

- Modificación de la jornada establecida para la realización del proyecto, desplazando al fin de semana las horas disponibles, ya escasas en un inicio, entre semana.
- Modificación del cómputo total de horas semanales dedicadas al proyecto.

Probabilidad

- Muy baja: 5 %

Impacto

- 0,05 x 7 días = 1,4 horas. Impacto bajo.

2.7.2. Enfermedad o lesión

Se contempla en este riesgo la indisponibilidad para continuar con el desarrollo del proyecto por causa médica, ya sea derivada directamente del trabajo o por causas ajenas.

Plan de prevención

- **Derivadas del trabajo**
  - Forzar y mantener una correcta postura corporal durante el trabajo.
  - Mantener una distancia saludable frente al monitor.
  - Realizar descansos cortos pero abundantes durante la jornada.
- **No derivadas del trabajo**
  - Descansar adecuadamente siguiendo las recomendaciones de los expertos en salud.
  - Realizar actividad deportiva carente de riesgos.

Plan de contingencia

- Acudir al médico para diagnosticar el alcance del problema y seguir las instrucciones correspondientes.
- En caso de diagnóstico favorable, continuar con el trabajo.
- En caso de diagnóstico desfavorable, pausar el desarrollo temporalmente según las indicaciones del médico.

Probabilidad

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

Impacto

- **Diagnóstico favorable**
  - 0,15 x 2 días = 1,2 horas. Impacto bajo.
- **Diagnóstico desfavorable**
  - 0,05 x 5 días = 1 hora. Impacto bajo.

2.7.3. Pérdida del código

Consiste en la pérdida, completa o parcial, del código que se está implementando en el desarrollo del proyecto. Esto puede deberse a fallos humanos, como no guardar los cambios o despistes en el borrado, fallos informáticos, como deterioro del equipo o corrupción de los datos, o fallos externos, como una avería en el suministro eléctrico.

Plan de prevención

- Utilizar un sistema de control de versiones para poder volver a un estado anterior.
- Alojar el código en un repositorio privado en la nube donde se actualice con los cambios de forma regular.
- Mantener una copia local en un dispositivo separado por si fallase el repositorio en línea.

Plan de contingencia

- Recuperación de la última versión disponible, ya sea del repositorio remoto o de la copia local.

Probabilidad

- Baja: 10 %

Impacto

- 0,10 x 1 días = 0,4 horas. Impacto muy bajo.

2.7.4. Pérdida de la documentación

Consiste en la pérdida, completa o parcial, de la documentación que se está redactando como memoria del proyecto. Dado que la herramienta que se está utilizando es en línea, podría darse que los cambios no se guarden por un error en la conexión o que fallase el sistema de la propia aplicación.

Plan de prevención

- Descargar siempre una copia de la documentación antes de cerrar la aplicación.

Plan de contingencia

- Recuperación de la última versión disponible.

Probabilidad

- Baja: 5 %

Impacto

- 0,05 x 1 días = 0,2 horas. Impacto muy bajo.

2.7.5. Pérdida de la base de datos

Consiste en la pérdida de la base de datos local con la que se está trabajando en el desarrollo del proyecto.

Plan de prevención

- Guardar los scripts que definen la creación de la base de datos junto al código del proyecto, sujeto al plan de riesgos 2.7.3.
- Generar y almacenar scripts que inserten en la base de datos toda la información necesaria.

Plan de contingencia

- Generar de nuevo la base de datos utilizando los scripts que se encuentran en el repositorio.
- Utilizar los scripts de población para restaurar toda la información necesaria.

Probabilidad

- Baja: 10 %

Impacto

- 0,10 x 1 días = 0,4 horas. Impacto muy bajo.

2.7.6. Pérdida de conexión a internet

Consiste en la pérdida temporal de conexión a internet imposibilitando, principal pero no únicamente, la búsqueda de información, las pruebas del código que se está desarrollando, la actualización del código fuente en el repositorio y el guardado de la documentación.

Plan de prevención

- Contratación de una conexión a internet con suficiente velocidad y estabilidad.
- Utilización de conexión por cable Ethernet frente a conexión Wi-Fi siempre que sea posible.

Plan de contingencia

- Conexión a internet a través del smartphone personal mediante tethering.
- Trasladarse a un lugar donde poder trabajar y acceder a internet, como una biblioteca o un aula de trabajo en la universidad.

Probabilidad

- Baja: 5 %

Impacto

- 0,05 x 1 días = 0,2 horas. Impacto muy bajo.

2.7.7. Fallo del equipo informático

Relacionado con los posibles daños que el equipo utilizado para el proyecto pudiera sufrir. Esto engloba tanto acciones estrictamente relacionadas con el trabajo como casos externos.

Plan de prevención

- Mantenimiento del equipo bajo unas condiciones ambientales adecuadas.
- Revisiones periódicas sobre el estado del equipo, tanto físico como digital, manteniendo actualizado siempre el sistema operativo evitando así vulnerabilidades.
- Copia de seguridad semanal del estado del equipo.

Plan de contingencia

- Si el fallo es de hardware, reemplazar la pieza dañada en caso de ser posible o el ordenador completo en caso de ser un fallo general. Posteriormente restaurar la última copia de seguridad realizada.
- Si el fallo es de software, recuperar y restaurar la última copia de seguridad realizada.

Probabilidad

- Baja: 15 %

Impacto

- 0,15 x 1 días = 0,6 horas. Impacto muy bajo.

3. Antecedentes

La tarea de valorar el estado de IkastenBot se realiza desde diferentes puntos de vista.

Bots en la docencia

Actualmente existen numerosos chatbots utilizados en la docencia, tanto asistentes genéricos que se integran con otros sistemas (e.g. Mongoose Harmony[1]), como aplicaciones dedicadas a tareas específicas de una institución (e.g. EUITI Bot[2]). La mayoría de estos últimos se limitan a unas pocas funcionalidades, como por ejemplo responder preguntas frecuentes o entregar ejercicios y ofrecer feedback automático.

El presente bot busca la integración de muchas de estas funcionalidades junto a otras novedosas en una sola aplicación, con el objetivo de acompañar al alumno a lo largo de todo el proceso de elaboración del TFG.

Versiones anteriores

Versiones anteriores de la propia aplicación son la mayor referencia a la hora de enfocar este proyecto, mas ninguna de ellas ha llegado a estar operativa para el público.

La primera versión de la aplicación, llevada a cabo por Amaia de Pablo Ruiz [1], consistía en el desarrollo del bot con varias de las funcionalidades descritas previamente, así como un panel de administración. El desarrollo se realizó utilizando el lenguaje PHP con el framework *Symfony*.

La segunda versión, desarrollada por David Pérez Gómez [5], fue una ampliación de la primera, extendiendo varias de sus funcionalidades y aportando mejoras sobre ella. Se mantuvo tanto el lenguaje PHP como la estructura anterior.

[1] https://www.mongooseresearch.com/harmony

[2] Bot de telegram utilizado en la asignatura de DAWE para el envío y corrección de ejercicios

4. Captura de requisitos

En este capítulo se presentan los requisitos que debe cumplir la aplicación IkastenBot. Junto a estos se incluyen los diagramas de casos de uso y el modelo de dominio correspondiente.

4.1. Requisitos funcionales

A continuación se presentan los requisitos funcionales que IkastenBot debe cumplir.

4.1.1. Ayuda sobre el bot

Cualquier usuario, alumno o tutor, debe poder consultar los comandos disponibles así como una descripción de cada uno utilizando el comando /help.

4.1.2. Consultar preguntas frecuentes

Al igual que el comando anterior, cualquier usuario podrá consultar la lista de preguntar frecuentes utilizando el comando /faq. Pulsando en alguna de las preguntas de la lista, el bot responderá con la información necesaria.

4.1.3. Registrar TFG

El bot permitirá a un alumno registrar un único proyecto en el sistema, equivalente a darse de alta como usuario. Para ello, el alumno deberá iniciar una conversación con el bot en la plataforma de Telegram, ejecutar el comando /registerTFG y facilitar los siguientes datos a medida que el bot los solicite:

- **Título del TFG**: Título completo del Trabajo de Fin de Grado.
- **Nombre**: Nombre completo del alumno.
- **Correo**: Dirección de correo de la universidad.
- **Idioma**: Idioma del TFG, a elegir entre ES, EN y EU.
- **Tutor**: Tutor del proyecto, a seleccionar de una lista.

4.1.4. Validar registro TFG

En el momento que un alumno registra un TFG en la plataforma, el bot enviará al tutor seleccionado por el alumno un mensaje informando de dicho registro. Es en este momento que el tutor puede, pulsando una de las dos opciones que el bot muestra, confirmar o rechazar la creación del proyecto.

En ambos casos, sea confirmada o rechazada la creación del TFG, el bot avisa al alumno de la elección que el tutor ha tomado.

4.1.5. Comprobar datos

Permitir a los alumnos comprobar los datos del TFG que está desarrollando. Para ello, el alumno deberá ejecutar el comando /checkData y el bot responderá con la siguiente información:

- **Título del TFG**: Título del TFG sobre el que está trabajando el alumno.
- **Nombre**: Nombre completo del alumno registrado en el sistema.
- **Tutor**: Tutor asignado al TFG sobre el que está trabajando el alumno.

4.1.6. Enviar diagrama Gantt

El alumno podrá enviar al bot un diagrama de Gantt (en formato gan). La aplicación analiza, descompone y guarda la información de dicho diagrama en base de datos para su posterior uso.

4.1.7. Envío de notificaciones

El bot creará notificaciones para los alumnos en base a hitos y fechas importantes, indicando el tiempo que falta para la fecha de finalización. Dichas fechas deben haber sido marcadas en el diagrama de Gantt que se haya enviado previamente al bot.

4.1.8. Soporte multidioma

La información que el bot pone a disposición de los usuarios estará disponible en los tres idiomas que soporta la aplicación: español, euskera e inglés.

4.2. Jerarquía de actores

Para la definición de los casos de uso se han identificado cinco actores diferentes, tal como se puede observar en la figura 4.1.

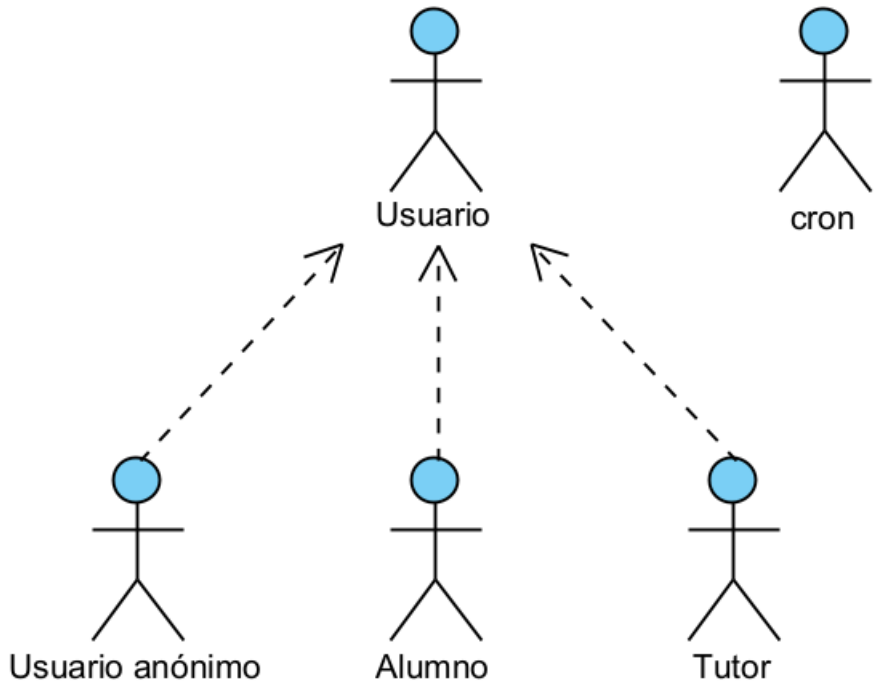

Figura 4.1: Jerarquía de actores

El Usuario representa a todos los usuarios de la aplicación, mientras que el Usuario anónimo representa específicamente a todos los usuarios que acceden a la aplicación por primera vez y no constan en el sistema. Por otra parte, Alumno y Tutor representan usuarios identificados con diferente acceso a los comandos disponibles, mientras que cron representa al actor que enviará las alertas.

4.3. Casos de uso

A continuación se presenta el diagrama de casos de uso, resultante del análisis de los requisitos funcionales así como de los actores previamente definidos.

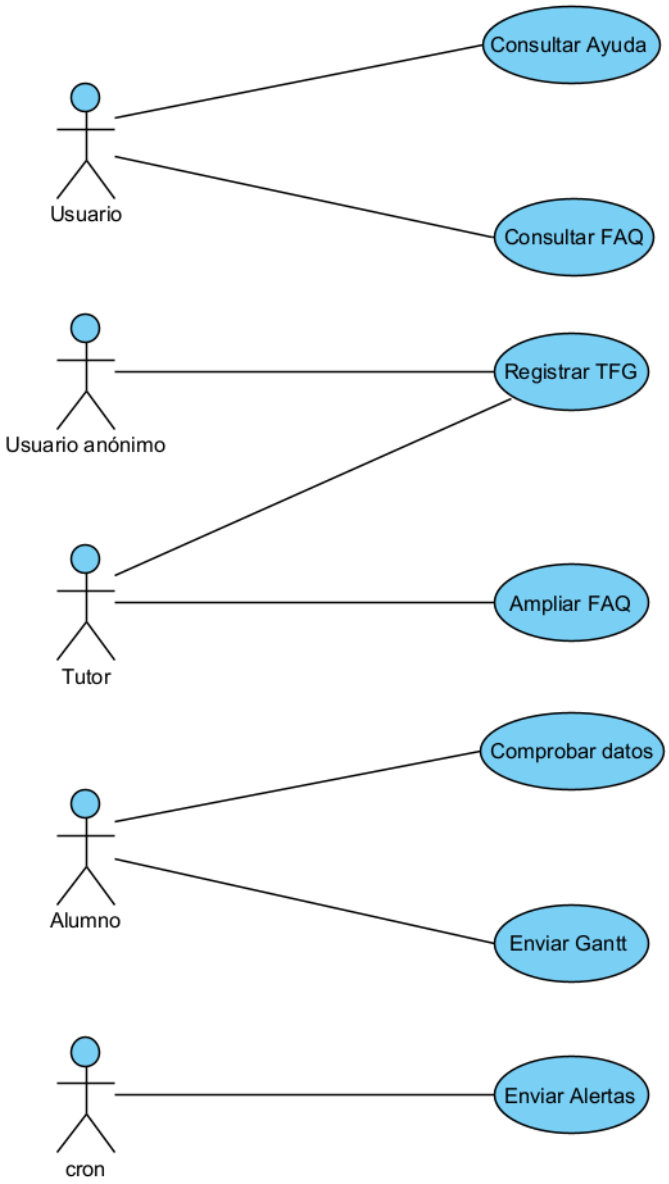

Figura 4.2: Casos de uso

4.4. Modelo de dominio

Como última parte de la captura de requisitos, se presenta el modelo de dominio que define la estructura principal del proyecto en lo que a datos se refiere. En él se indica la información necesaria para el funcionamiento del mismo, así como las entidades principales y las relaciones entre ellas.

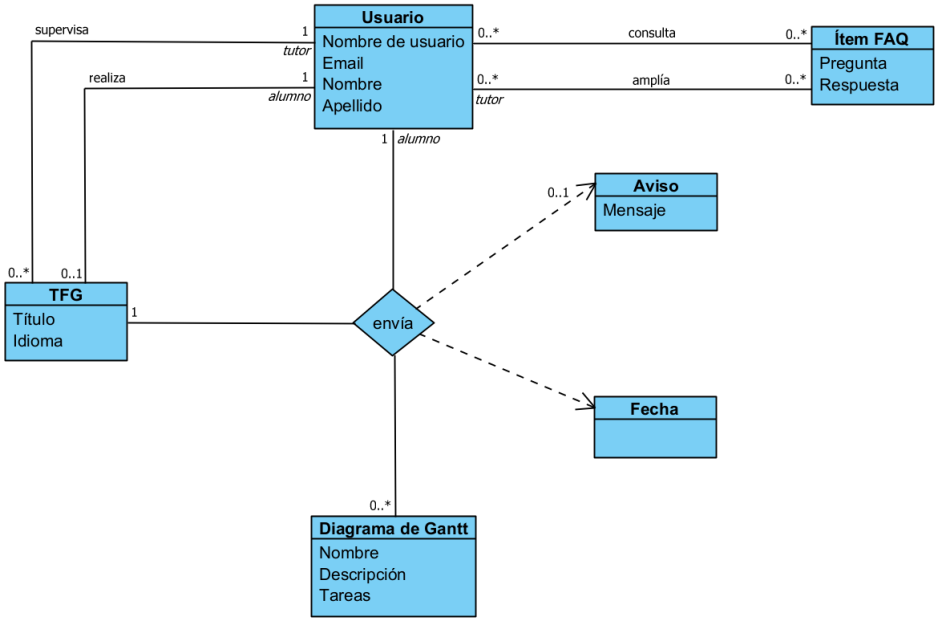

Figura 4.3: Modelo de dominio

4.4.1. Entidades

- **Usuario**: Define a todas las personas que utilicen la aplicación. Como datos guardamos su usuario, correo electrónico, nombre y apellido.
- **TFG**: Representa un proyecto que se está desarrollando. Guardamos su título e idioma.
- **Diagrama de Gantt**: Contiene la información de cada diagrama, nombre y descripción del mismo, así como sus tareas.
- **Aviso**: Registra cada aviso generado a partir de los diagramas con el mensaje a enviar.
- **Ítem FAQ**: Hace referencia a las preguntas frecuentes así como sus respuestas.

4.4.2. Relaciones

- **Realiza**: Describe al usuario (alumno) que realiza un TFG.
- **Envía**: Relación múltiple entre un usuario (alumno), el TFG que realiza, los diagramas de Gantt referentes al proyecto y los avisos que se generan de los mismos.
- **Supervisa**: Describe al usuario (tutor) que supervisa un TFG.
- **Consulta**: Representa la consulta de un usuario a las preguntas frecuentes.
- **Amplía**: Define la creación de nuevos ítems por parte de un usuario (tutor).

5. Análisis y diseño

En este capítulo se explican todas las tareas llevadas a cabo para realizar los diseños necesarios, antes de comenzar con la implementación de la aplicación.

Se presentarán los diseños sobre las conversaciones que tendrán los usuarios con el bot, así como el diseño de la nueva base de datos, detallada en el Diagrama Entidad-Relación. De igual modo, se detalla el diagrama de clases de la aplicación y los diagramas de secuencia correspondientes a los principales flujos de ésta.

5.1. Diseños de las conversaciones

Al tratarse de una interfaz conversacional, esta sección es la equivalente al diseño visual de una página web o a la interfaz gráfica de una aplicación. Antes de comenzar con la implementación es necesario definir los flujos conversacionales, así como el tono y el tipo de frases que utilizará el bot en cada caso.

En esta sección se muestran los diseño realizados sobre las conversaciones así como una explicación de cada una ellas.

5.1.1. Ayuda

Es el comando que probablemente más usuarios utilicen al abrir una conversación con el bot por primera vez, ya que su objetivo es mostrar y explicar los comandos disponibles. Debido a esto, la información será lo más clara y concisa posible, con un tono completamente neutro.

La figura 5.1 muestra a un usuario solicitando información con el comando /help.

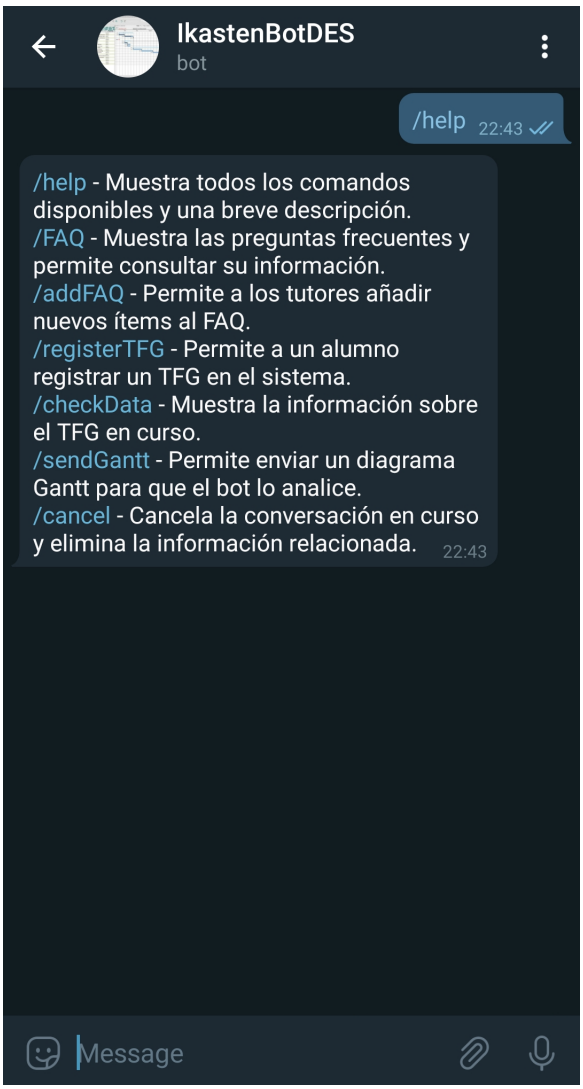

Figura 5.1: Diseño de la conversación para el comando de ayuda.

5.1.2. Preguntas frecuentes

Al tratarse de un comando que cualquier usuario puede utilizar, en este caso el bot tampoco utilizará ningún tono en concreto. Simplemente se limitará a mostrar las preguntas disponibles y, una vez el usuario seleccione una de ellas, responderá con la información que tenga al respecto.

Aunque las preguntas y respuestas estén soportadas en tres idiomas, el

diseño se mantiene igual, por lo que únicamente se muestra un ejemplo.

En la figura 5.2 observamos una conversación donde el usuario solicita información, pulsa en la tercera pregunta y el bot facilita la información asociada. Tras ello, se hace limpieza de la lista para no saturar la conversación, dejando en el chat únicamente la respuesta.

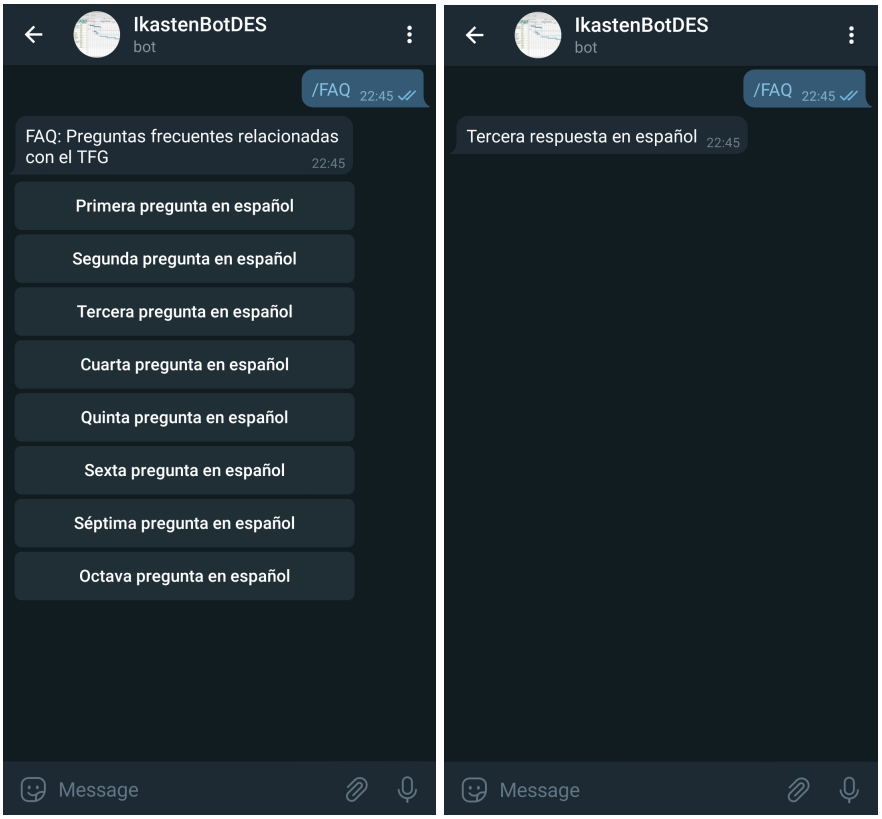

Figura 5.2: Diseño de la conversación para las preguntas frecuentes.

5.1.3. Registro de un TFG

A la hora de realizar el registro de un proyecto, el bot solicitará al alumno que aporte cierta información. Para ello se realizarán varias preguntas entablando una conversación. Algunas de ellas esperan una respuesta escrita mientras que otras esperan que el alumno realice una selección entre las opciones disponibles.

En la figura 5.3 podemos observar el flujo diseñado asumiendo que la conversación se finalice con éxito. Aunque en una conversación real se hubieran eliminado los mensajes que permiten selección, se han mantenido de cara a mostrar el diseño.

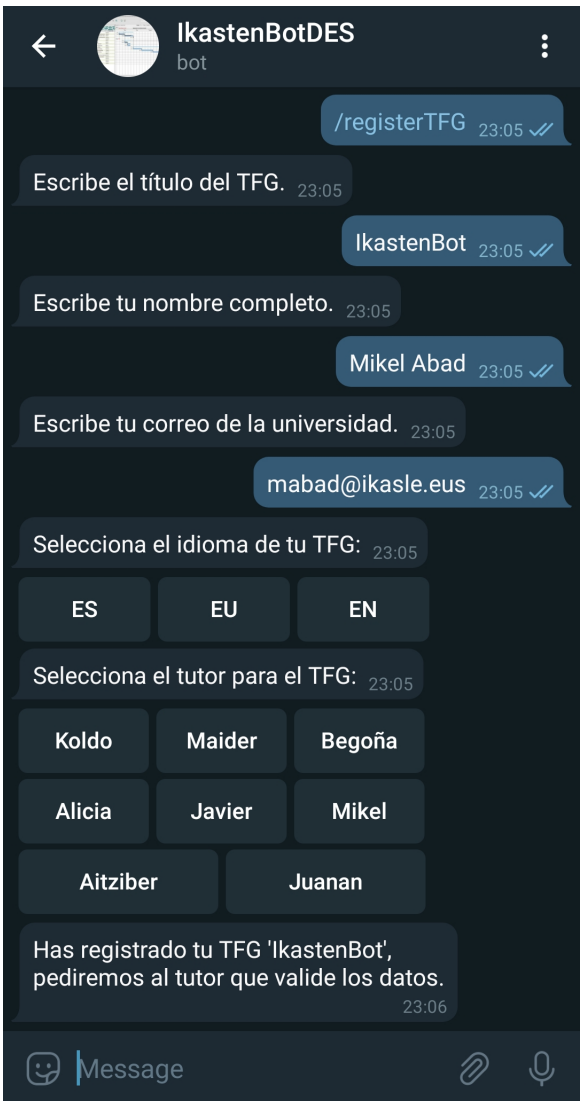

Figura 5.3: Diseño de la conversación para registrar un TFG.

5.1.4. Confirmación de un TFG

Tras el proceso de registro de un TFG por parte del alumno, el bot iniciará una conversación con el tutor seleccionado. En éste se informa sobre la creación de dicho trabajo, mostrando tanto el nombre del proyecto como del alumno, y se solicita confirmación.

A continuación, en la figura 5.4, se muestra el diseño de la conversación que entabla el bot con el tutor para la confirmación del registro de un TFG. Una vez más, de cara a mostrar el diseño completo, no se ha eliminado el mensaje con opciones.

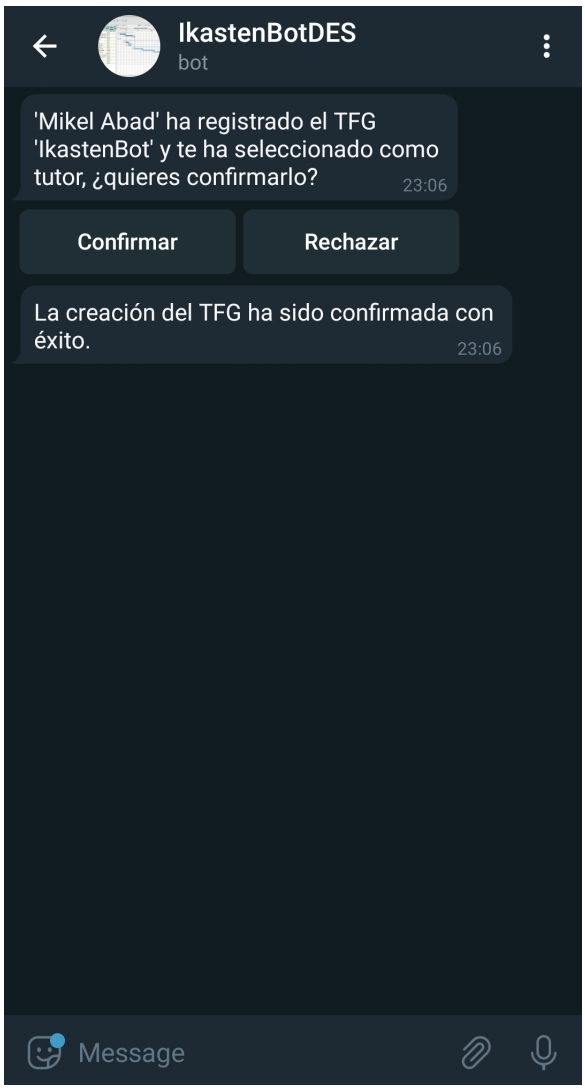

Figura 5.4: Diseño sobre la confirmación de un TFG.

5.1.5. Comprobar datos

Cuando un alumno desea consultar los datos guardados en el bot, puede utilizar este comando para ver su nombre, su proyecto y su tutor. Por ello, la respuesta será simple y concisa, como podemos ver en la figura 5.5.

Figura 5.5: Diseño de la conversación para comprobar los datos del TFG.

5.1.6. Enviar diagrama Gantt

El envío de un diagrama Gantt es un proceso tan sencillo como adjuntar el documento en la conversación con el bot. Como podemos observar en la figura 5.6, la conversación se limita a lo estrictamente necesario.

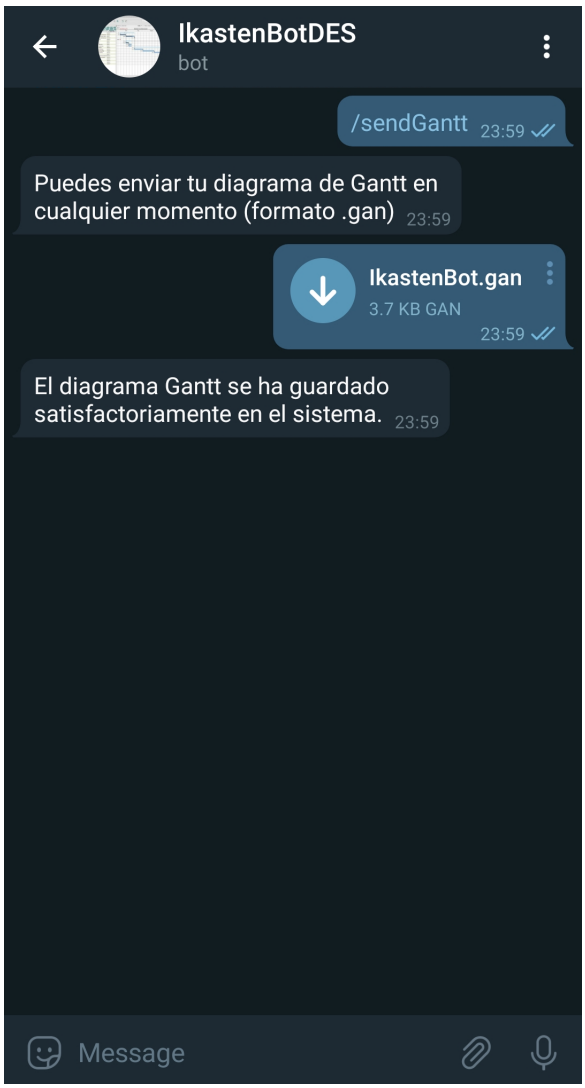

Figura 5.6: Diseño de la conversación para enviar un diagrama Gantt.

5.2. Diagrama de base de datos

El siguiente diagrama mostrado en la figura 5.7 representa el Diagrama Entidad-Relación que define la base de datos de la aplicación.

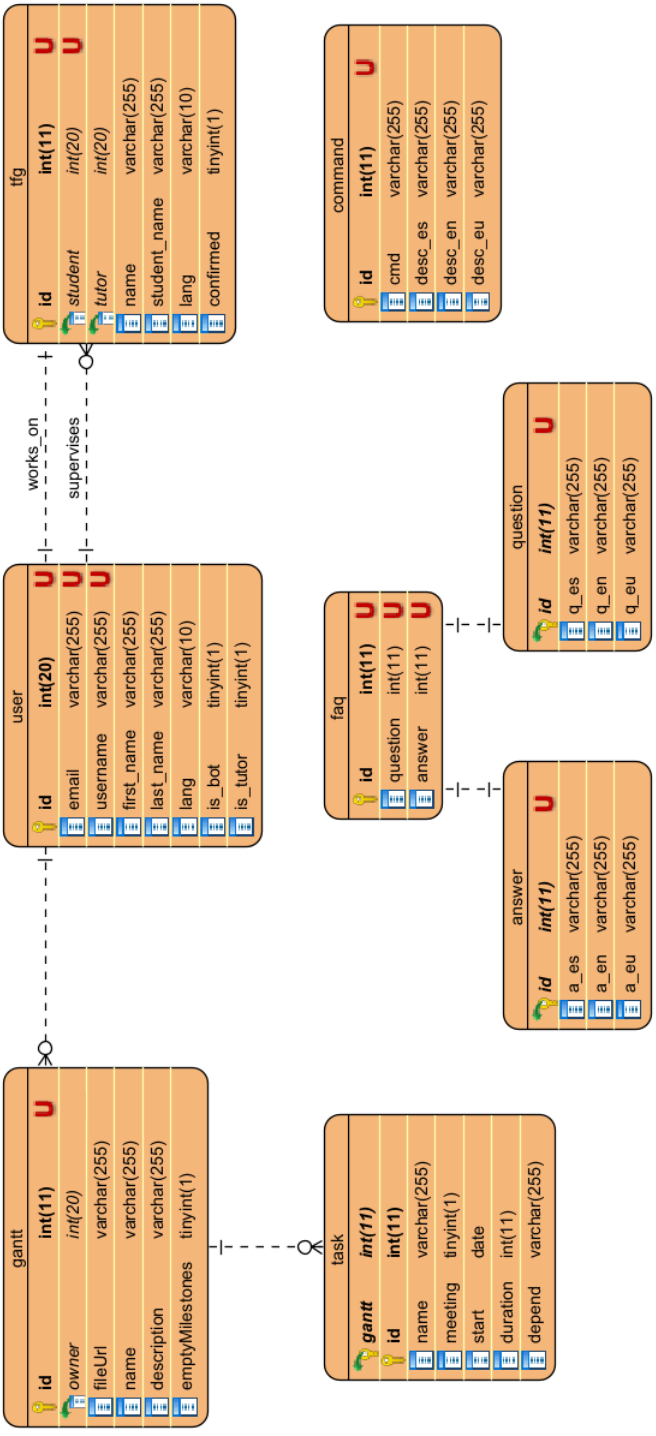

Figura 5.7: Diagrama de la base de datos

A continuación se detallan las tablas definidas así como su funcionalidad, listando la información que se guarda en cada una de ellas.

5.2.1. Usuarios

Utilizada para guardar todo lo relacionado con los usuarios de la aplicación. La información a guardar en ella es la siguiente:

- Identificador propio de la aplicación de Telegram (único)
- Email de la universidad (único)
- Nombre de usuario escogido en Telegram (único)
- Nombre real del usuario
- Apellido/s del usuario
- Idioma utilizado por el usuario
- Indicador para saber si el usuario es un bot
- Indicador para saber si el usuario es un tutor

5.2.2. TFGs

En esta tabla se guarda la información referente a los proyectos:

- Identificador del proyecto (único)
- Identificador del alumno que lo realiza (único)
- Identificador del tutor que lo supervisa
- Título del proyecto
- Nombre real del alumno
- Idioma del proyecto
- Indicador sobre si el proyecto ha sido confirmado por el tutor

5.2.3. FAQ

En las tablas relacionadas con las preguntas frecuentes se guardan las preguntas así como las respuestas en diferentes idiomas, de modo que estén disponibles en los tres lenguajes que soporta la aplicación.

faq

- Identificador del ítem en el FAQ (único)
- Identificador de la pregunta (único)
- Identificador de la respuesta (único)

question

- Identificador de la pregunta (único)
- Texto de la pregunta en español
- Texto de la pregunta en inglés
- Texto de la pregunta en euskera

answer

- Identificador de la respuesta (único)
- Texto de la respuesta en español
- Texto de la respuesta en inglés
- Texto de la respuesta en euskera

5.2.4. Diagramas de Gantt

En estas tablas se guarda la información referente a los diagramas de Gantt que envía un usuario.

gantt

- Identificador del diagrama (único)
- Identificador del alumno que lo envió
- Texto para formar la URL con la que recuperar el archivo de la nube de Telegram
- Nombre del diagrama
- Descripción del diagrama
- Indicador sobre si hay hitos definidos

task

- Identificador del diagrama al que pertenece (clave compuesta)
- Identificador de la tarea dentro del propio diagrama (clave compuesta)
- Nombre de la tarea
- Indicador sobre si la tarea es o contiene una reunión
- Fecha de inicio de la tarea
- Duración de la tarea
- Lista de tareas que depende de ésta

5.2.5. Comandos

Finalmente, la tabla en la que se registran los comandos disponibles en la aplicación así como su descripción en los idiomas soportados.

- Identificador del comando (único)
- Texto a utilizar en el chat
- Descripción del comando en español
- Descripción del comando en inglés
- Descripción del comando en euskera

5.3. Diagrama de clases

En esta sección se presenta y explica el diagrama de clases que define la aplicación de IkastenBot. Para tratar de seguir un diseño adecuado, se ha dividido la implementación en cuatro grupos: controladores, comandos, servicios y modelos.

A continuación, en la figura 5.8, se muestra el diagrama de clases de la aplicación y la explicación detallada del objetivo de cada uno de los grupos.

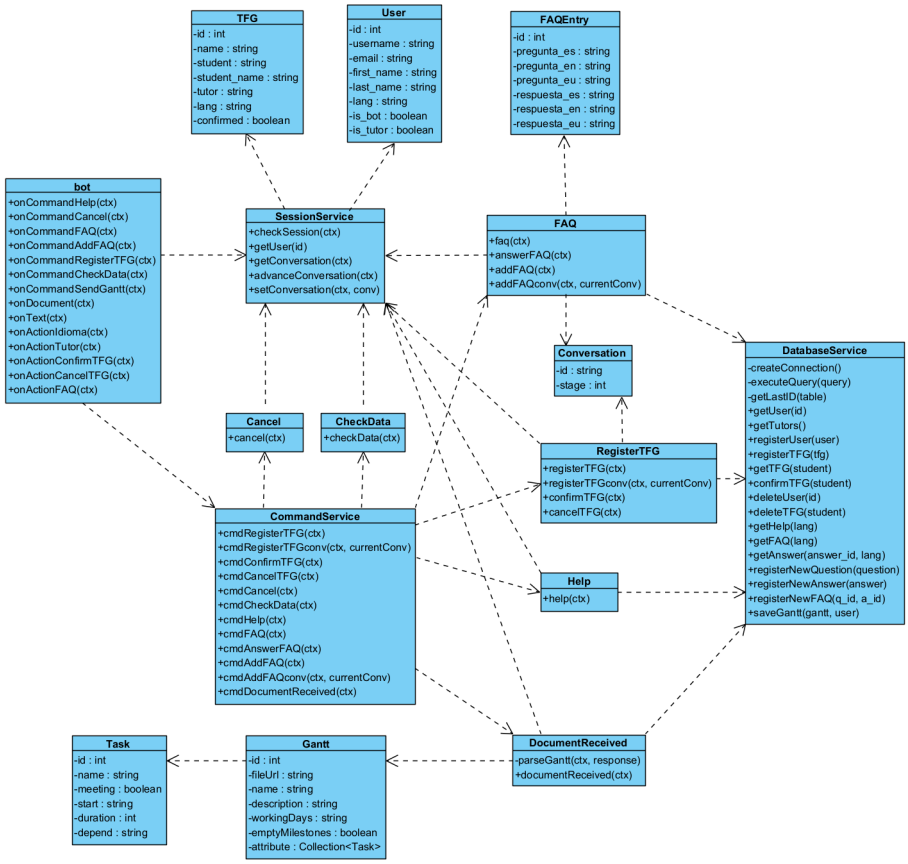

Figura 5.8: Diagrama de clases

Controladores

Las clases definidas en esta parte de la aplicación son las encargadas de recibir y gestionar los mensajes de los usuarios de Telegram. En función del mensaje recibido puede que haya que ignorarlo, tratarlo directamente o remitirlo a otras clases.

En el estado actual de la aplicación, la única clase encargada de gestionar los mensajes de manera directa es *bot*.

Comandos

Las clases que forman esta sección son aquellas que contienen toda la lógica necesaria para la ejecución de los comandos y acciones que solicita el usuario.

Las clases que se consideran comandos son: RegisterTFG, Help, Cancel, FAQ, DocumentReceived y CheckData.

Servicios

El grupo de los servicios contiene aquellas clases que se encargan de ejecutar operaciones que se requieren desde el resto de partes de la aplicación, como por ejemplo la gestión de la sesión o el acceso a la base de datos.

Los servicios definidos son: SessionService, DatabaseService y CommandService.

Modelos

Finalmente se define el grupo de los modelos, formado por las clases utilizadas para representar objetos o entidades. Contienen información individual y se utilizan mediante instancias de los mismos.

Los modelos utilizados son: User, TFG, Conversation, Gantt, Task y FAQEntry.

5.4. Diagramas de secuencia

En esta última sección se presentan y explican los diagramas de secuencia elaborados. Estos tienen como objetivo reflejar el flujo más relevante de la aplicación, entendiendo por qué puntos pasa y en qué orden, así como la información que se transmite.

Registrar TFG

Para representar el flujo a seguir a la hora de registrar un TFG en el sistema, se han realizado dos diagramas diferentes.

En el primero, en la figura 5.9, podemos observar el diagrama de secuencia referente al comienzo del proceso, desde que el usuario solicita registrar un proyecto hasta que el bot comienza a pedirle información.

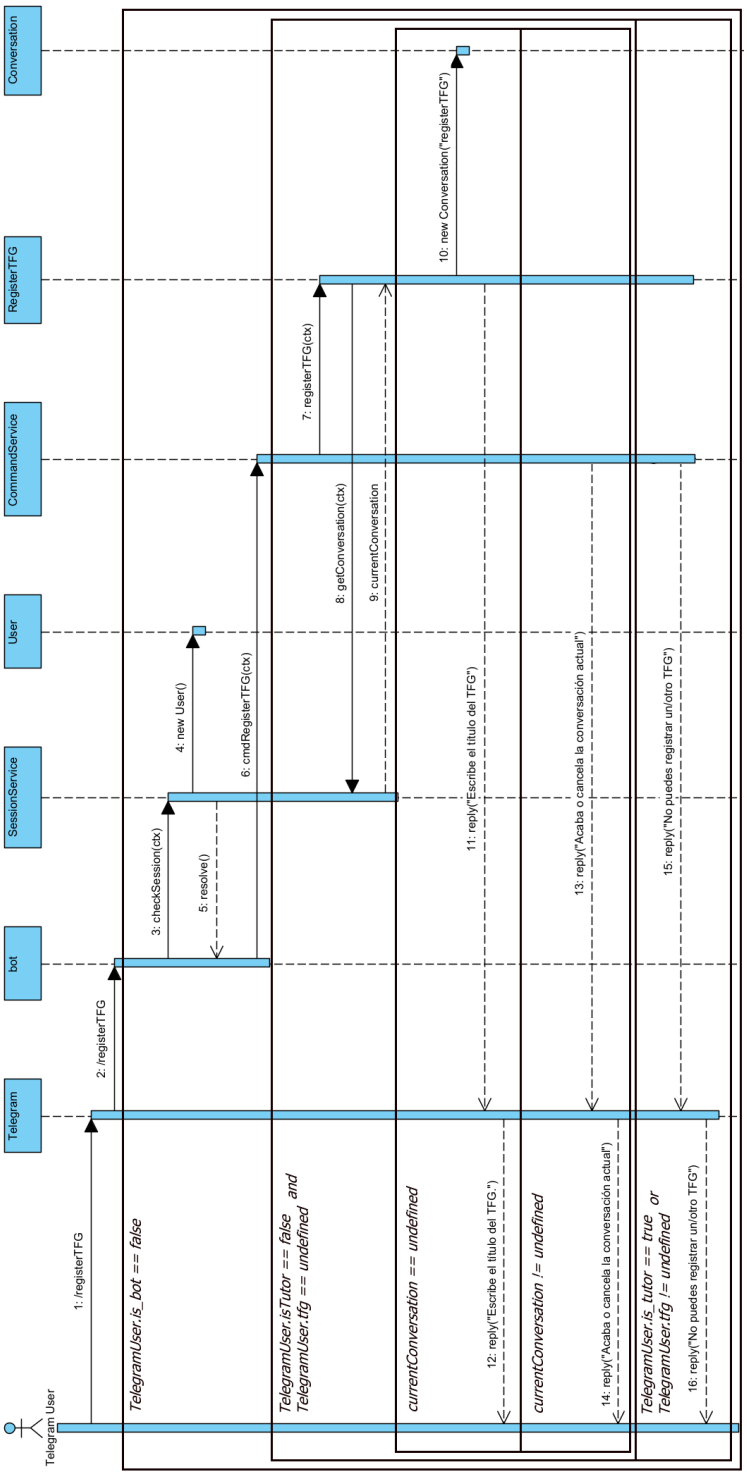

Figura 5.9: Diagrama de secuencia para iniciar el proceso de registro

En el segundo, en la figura 5.10, podemos observar el diagrama de secuencia referente a el intercambio de información que sucede entre el bot y el alumno para llegar a realizar el registro del proyecto.

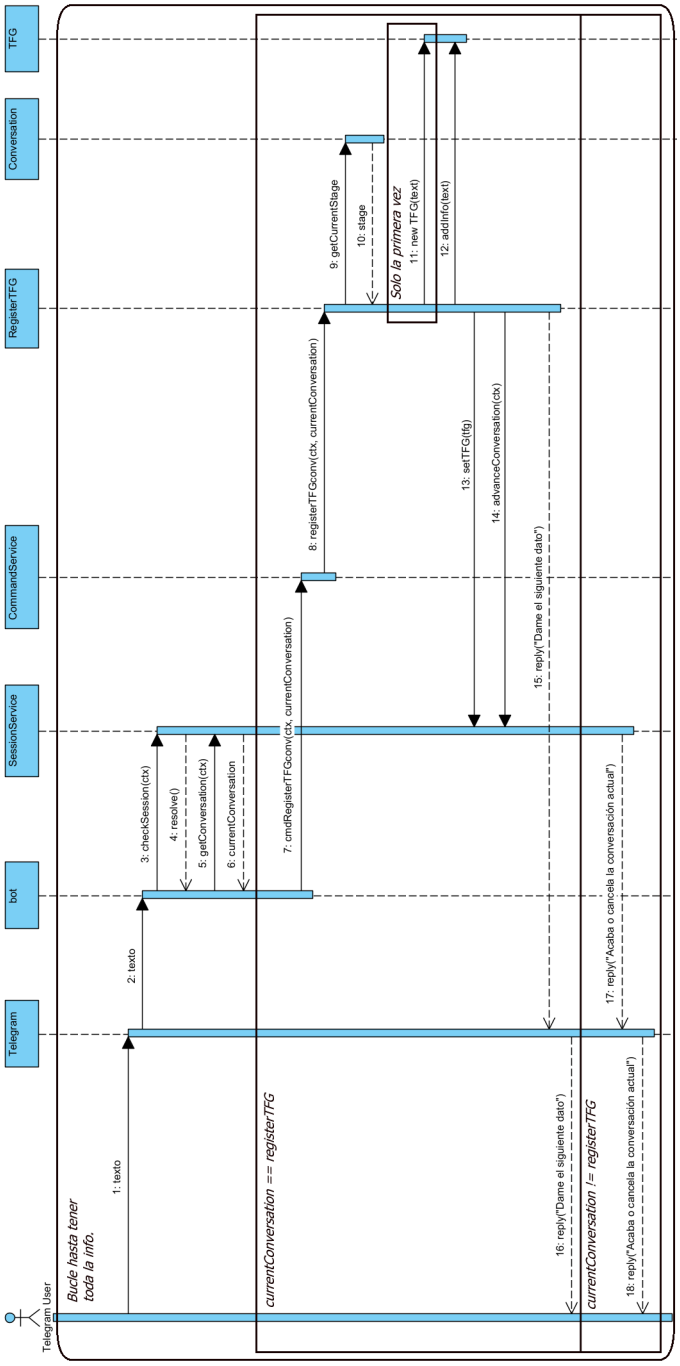

Figura 5.10: Diagrama de secuencia para realizar el registro

6. Implementación

En este capítulo se explicarán diferentes apartados y detalles del propio desarrollo del proyecto, cómo se ha creado y llevado a cabo así como las herramientas que se han utilizado.

Del mismo modo, se presentarán los problemas hallados en el proceso y las soluciones que se han dado a los mismos.

6.1. Creación del bot y configuración del entorno

El primer paso en este proyecto es crear la cuenta para el bot, ya que se necesita en todas las fases del desarrollo. Es un proceso muy sencillo que se realiza desde la propia aplicación de Telegram, siguiendo los siguientes pasos:

- Buscar al usuario *BotFather* e iniciar una conversación con él.
- Ejecutar el comando /newbot.
- Introducir la información requerida, en el momento de crear IkastenBot es la siguiente:
  - Nombre del bot para mostrar.
  - Username del bot, debe ser único y acabar en "bot".
- Guardar el token que nos facilita para usarlo posteriormente.

En la figura 6.1 podemos ver una conversación donde se crea un bot de ejemplo. El token ha sido censurado ya que nunca deberíamos mostrarlo y debemos mantenerlo seguro.

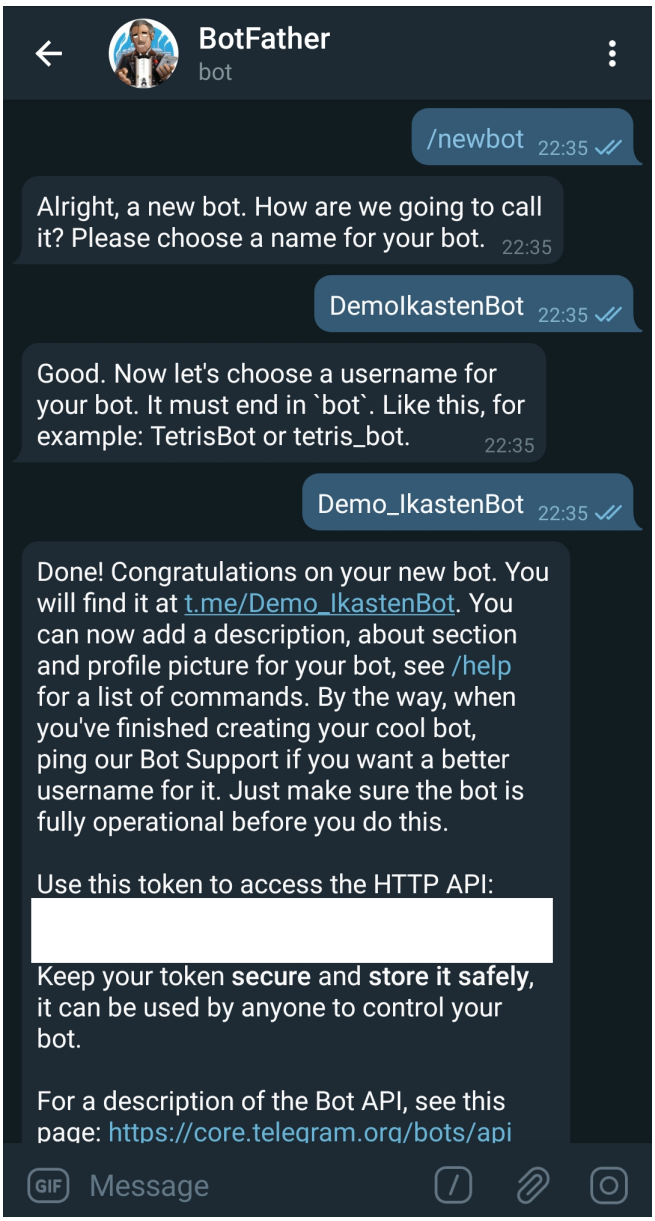

Figura 6.1: Ejemplo de creación de un bot con BotFather

Una vez creado el bot y obtenido el token, para poder comenzar a desarrollar es necesario disponer de las herramientas apropiadas. Para ello, teniendo en cuenta las herramientas que se han escogido para este proyecto, realizamos la siguiente configuración:

- Instalar Git.
- Instalar Visual Studio Code y añadir algunas extensiones que facili-

ten el desarrollo. Ninguna es estrictamente necesaria, pero para este proyecto se han utilizados las siguientes:

  - **npm**[1]: Añade a VS Code soporte para npm. Autor: egamma.
  - **npm Intellisense**[2]: Autocompleta los módulos de npm en los import. Autor: Christian Kohler.
  - **DotENV**[3]: Añade a VS Code soporte para la sintaxis de dotenv. Autor: mikestead.
  - **ESLint**[4]: Integra ESLint JavaScript en el propio VS Code. Autor: Dirk Baeumer.

- Instalar la última versión estable de Node.js (incluye el gestor de paquetes npm).
- Instalar MySQL y MySQL Workbench.
- Descargar ngrok.

En este punto, con todas las herramientas preparadas, el siguiente paso es crear el proyecto. Para ello, inicializamos el proyecto de Node con el comando *npm init* e instalamos los siguientes paquetes mediante npm: *telegraf*, *dotenv*, *body-parser* y *mysql*.

Una vez todo ha sido instalado, preparado e inicializado, es el momento de comenzar con el desarrollo del bot y sus funcionalidades.

6.2. Telegraf y estructura del proyecto

Como ya se ha mencionado previamente, la librería utilizada para el desarrollo el proyecto ha sido Telegraf, un framework moderno para bots de Telegram en Node.

A la hora de escoger dicha librería, la única alternativa con base suficiente como par ser valorada fue *node-telegram-bot-api* por el usuario yagop. Ésta se descartó ya que únicamente se trataba de un módulo que servía para interactuar con la API de Telegram, mientras que Telegraf ofrece distintas herramientas y funciones adicionales que facilitan el desarrollo de un bot.

[1] https://marketplace.visualstudio.com/items?itemName=eg2.vscode-npm-script
[2] https://marketplace.visualstudio.com/items?itemName=christian-kohler.npm-intellisense
[3] https://marketplace.visualstudio.com/items?itemName=mikestead.dotenv
[4] https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint

La estructura del proyecto basado en dicho framework, una vez finalizado el desarrollo y teniendo en cuenta todos los ficheros que han sido creados, se ha organizado de la siguiente manera:

- IkastenBot
  - bot
    - commands
      - Cancel.js
      - CheckData.js
      - DocumentReceived.js
      - FAQ.js
      - Help.js
      - RegisterTFG.js
    - models
      - Conversation.js
      - FAQEntry.js
      - Gantt.js
      - Task.js
      - TFG.js
      - User.js
    - services
      - CommandService.js
      - DatabaseService.js
      - SessionService.js
    - bot.env.json
    - bot.env.json.template
    - bot.js
  - node_modules
  - sql
    - data.sql
    - schema.sql
  - .env
  - .env.template
  - index.js
  - package-lock.json
  - package.json
  - README.md

Como podemos observar, se utilizan dos ficheros para la configuración del proyecto. El fichero .env guarda la información referente a la aplicación (url, puerto, configuración de base de datos) mientras que el fichero bot.env.json guarda la información referente al bot como tal (token y path).

6.3. Desarrollo de la primera versión

El objetivo del primer desarrollo es tener una versión estable que permita a un alumno registrar un TFG en el sistema. Para ello, el alumno debe ejecutar el comando /registerTFG y el bot debe realizar una serie de preguntas, al mismo tiempo que recoge y organiza la información que le aporta el alumno.

Lo primero y más importante es que el bot esté disponible para la escucha y respuesta de mensajes. Para ello, implementado en el fichero index, se utiliza Express tanto para escuchar en el puerto definido como para dirigir las llamadas hacia el código de nuestro bot.

Junto a esta pequeña implementación inicial, comenzamos a usar la herramienta ngrok para poder realizar pruebas en el entorno local.

Una vez el bot está en funcionamiento, la primera parte a implementar es la interacción con el usuario mediante los comandos, concretamente /registerTFG, pues es el desencadenante del proceso de registro. Para detectar el evento Telegraf ofrece la función *command*, a la que el framework invocará directamente cuando un usuario utilice el comando indicado. En nuestro caso "/registerTFG".

Se implementa parte de los servicios SessionService, donde se extiende la funcionalidad de la sesión que ofrece Telegraf para gestionar la información de la sesión actual, y DatabaseService, donde se gestiona el acceso a la base de datos. En esta primera fase SessionService se utilizará para gestionar la información del usuario activo, así como la información del TFG, en conjunto con DatabaseService para guardar dicha información a un entorno persistente.

Como parte de la funcionalidad que tendrá SessionService en esta versión se encuentra el método *checkSession*, cuyo objetivo es obtener de base de datos toda la información del usuario y poner la sesión en orden, teniendo en cuenta los datos que se manejan así como la conversación en curso.

Un concepto importante a explicar en este punto es el de la conversación, ya que a la hora de registrar un proyecto el bot establecerá una con el usuario.

Conversación

Cuando un proceso requiere de más de una interacción con el usuario, se genera una conversación.

En el contexto del bot, una conversación implica generar un entorno cerrado donde solo la información adecuada permite avanzar. Si el bot solicita al usuario un nombre, éste debe contestar con texto plano. En el caso de responder con otro evento, como una imagen, un fichero o un sticker, simplemente será ignorado. En el caso de intentar ejecutar otro comando, el bot le informará que debe finalizar o cancelar la conversación actual.

De este modo, cada vez que se recibe la información esperada correctamente, se procesa y se avanza la conversación al siguiente punto. Esto se repite hasta que se finaliza correctamente el último nivel definido para ella.

Este sistema de conversaciones se gestionará en el propio SessionService ya que se considera que forma parte de la sesión actual.

Una vez implementado el sistema de conversaciones, así como los primeros pasos de solicitud de información, en el siguiente punto se pretende solicitar al usuario que seleccione una opción entre varias haciendo click en ella, en lugar de escribirla él mismo.

Con ese objetivo en mente se implementan las nuevas respuestas con la herramienta de Telegram *inline_keyboard*. Se hace a través de Telegraf, ya que ofrece soporte para este tipo de interacción, que permite mostrar al usuario un conjunto de opciones a seleccionar.

De este modo se implementa la solicitud de idioma, donde se pide al usuario que seleccione entre los idiomas disponibles para la realización de un proyecto, así como que indique su tutor, donde tras una consulta a base de datos para obtener el listado se muestran al alumno los tutores disponibles (véase 5.3).

Para poder recibir la información seleccionada se utilizan los eventos de Telegraf denominados *action*. Estos eventos son lanzados cuando se pulsa un botón, como los del *inline_keyboard*, y trasladan la información que llevase dicha opción. De cara a saber qué tutor ha elegido el alumno, se decide enviar el id de Telegram como información oculta en cada opción, de modo que al seleccionar una de ellas el bot puede saber a qué tutor pertenece.

Finalmente se registra en base de datos el alumno y su TFG, y se envía al tutor un mensaje para confirmar la creación del mismo. El proyecto se

guarda en base de datos con el flag *confirmed* en negativo, de modo que si el tutor confirma su creación éste se cambie a positivo, y si decide rechazarlo se elimine.

6.4. Desarrollo de la segunda versión

Partiendo de una versión estable que permite al usuario registrar en el sistema un proyecto, el objetivo de esta segunda versión es añadir unas cuantas funcionalidades más.

Se incorporan por tanto a la aplicación los siguientes comandos: /help, /checkData, /cancel, /FAQ y /addFAQ. A continuación explicamos brevemente qué implicaciones ha tenido cada uno de ellos.

Help

Para realizar el comando de ayuda se introducen en la base de datos los comandos disponibles así como sus descripciones en todos los idiomas soportados.

Cuando se recibe el evento del comando por parte de un usuario se recupera dicha información en el idioma del usuario actual y se muestra en formato lista (véase 5.1).

CheckData

Este comando, utilizado para mostrar al alumno información sobre su usuario y su proyecto, realiza una consulta a base de datos y forma una frase que envía al usuario (véase 5.5).

Cancel

Funcionalidad imprescindible para poder rectificar, ya sea porque se ha iniciado una conversación por error o porque se ha introducido de forma incorrecta algún dato.

Este comando borra toda la información guardada en la sesión referente a la conversación actual, en caso de haberla, y elimina también ésta.

FAQ

Para implementar el comando de preguntas frecuentes se introducen inicialmente unos datos de prueba de forma manual en la base de datos. Se implementa la consulta para obtener las preguntas en el idioma del usuario y se muestran todas ellas utilizando de nuevo el *inline_keyboard* (véase 5.2). Esto permite al usuario seleccionar cómodamente la pregunta que quiera.

Asimismo también se añaden los actions correspondientes para poder detectar y averiguar qué opción ha seleccionado el usuario, del mismo modo que con el idioma o los tutores previamente. Con dicha información se realiza una consulta a base de datos para poder mostrar al usuario la respuesta correspondiente en el idioma de su elección.

Finalmente se elimina la lista de preguntas para no saturar la conversación con el bot.

addFAQ

Con la intención de que los tutores puedan añadir a las preguntas frecuentes nuevas entradas sin depender del administrador de la base de datos, se incluye esta nueva funcionalidad.

En el momento que se recibe este comando el bot inicia una conversación con el tutor en la que solicita la pregunta y la respuesta en varios idiomas, para finalmente añadirla a la base de datos junto al resto.

6.5. Desarrollo de la versión final

En este punto disponemos de una base muy sólida para la aplicación, con diferentes mecanismos de control y varias funcionalidades disponibles para los usuarios. El último paso es avanzar hasta el máximo posible dentro del tiempo establecido en lo referente a los diagramas de Gantt.

La primera implementación que se realiza es la propia obtención del diagrama que envía el usuario, por lo que se añade un controlador para los eventos de tipo *document*. En este evento, lanzado cuando el bot recibe un documento en cualquier formato, se tiene que obtener el documento de la nube de Telegram, pues únicamente se recibe un identificador.

Tras obtener la información sobre la localización del documento[5] se debe realizar una petición para obtenerlo.

Una vez hemos obtenido la información del fichero es necesario analizarla, para lo que utilizamos una herramienta de validación de datos y la tratamos desde el formato original (gan, que utiliza el formato XML) a un formato manejable por nuestra aplicación: JSON.

[5]Formato: https://api.telegram.org/file/bot{BOT_TOKEN}/documents/file_115.gan

Finalmente, una vez tenemos el contenido a nuestra disposición, extraemos toda la información que nos resulte de utilidad y la guardamos en base de datos para poder darle un uso en el futuro con diferentes funcionalidades.

7. Verificación y pruebas

En este capítulo se explican de manera resumida las pruebas que se han realizado en el proyecto para verificar que las funcionalidades han sido desarrolladas de la manera correcta.

Aunque se han realizado pruebas constantemente a medida que se implementaba cada parte del código, las pruebas aquí definidas son aquellas que se han llevado a cabo una vez finalizado el desarrollo completo. Por tanto, los resultados aquí mostrados son los resultados finales sobre la versión final de la aplicación.

Dada la naturalidad del proyecto se ha decidido basar el plan de pruebas en la interacción con el bot, a excepción de algunos casos aislados. Por tanto, el resultado de la prueba se dará por válido cuando la información que reciba el usuario en la aplicación de Telegram por parte del bot sea la esperada. Asimismo, se dará por erróneo el resultado cuando la respuesta obtenida por el usuario por parte del bot sea mínimamente diferente a la esperada.

De cara a realizar las pruebas de manera ordenada y modular, se ha considerado adecuado realizar un pequeño plan de pruebas por cada comando disponible. A continuación se presentan éstos así como sus resultados.

7.1. Comando /help

Número	Usuario	Descripción	Resultado esperado	Resultado obtenido	Verificación
1	Sin registro	Se utiliza el comando sin haber interactuado antes con el bot	Se recibe el listado de ayuda en español, incluyendo todos los comandos y su descripción.	Se recibe el listado de ayuda en español, incluyendo todos los comandos y su descripción.	OK
2	Alumno	Se utiliza el comando tras haber registrado un TFG en español	Se recibe el listado de ayuda en español, incluyendo todos los comandos y su descripción.	Se recibe el listado de ayuda en español, incluyendo todos los comandos y su descripción.	OK
3	Alumno	Se utiliza el comando tras haber registrado un TFG en otro idioma	Se recibe el listado de ayuda en el idioma del proyecto, incluyendo todos los comandos y su descripción.	Se recibe el listado de ayuda en el idioma del proyecto, incluyendo todos los comandos y su descripción.	OK
4	Tutor	Se utiliza el comando	Se recibe el listado de ayuda en el idioma que consta para el usuario en base de datos, incluyendo todos los comandos y su descripción.	Se recibe el listado de ayuda en el idioma que consta para el usuario en base de datos, incluyendo todos los comandos y su descripción.	OK

Figura 7.1: Plan de pruebas para el comando /help

7.2. Comando /cancel

Número	Usuario	Descripción	Resultado esperado	Resultado obtenido	Verificación
1	Todos	Se utiliza el comando sin ninguna conversación iniciada	No sucede nada	No sucede nada	OK
2	Sin registro	Se utiliza el comando en mitad de la conversación "registerTFG"	Se eliminan los datos, se comunica al usuario que se ha cancelado la conversación y se le permite realizar otro registro.	Se eliminan los datos, se comunica al usuario que se ha cancelado la conversación y se le permite realizar otro registro.	OK
3	Tutor	Se utiliza el comando en mitad de la conversación "addFAQ"	Se eliminan los datos y se comunica al usuario que se ha cancelado la conversación.	Se eliminan los datos y se comunica al usuario que se ha cancelado la conversación.	OK

Figura 7.2: Plan de pruebas para el comando /cancel

7.3. Comando /checkData

Número	Usuario	Descripción	Resultado esperado	Resultado obtenido	Verificación
1	Sin registro	Se utiliza el comando sin haber registrado un TFG en el sistema	Se informa al usuario que no ha registrado aún ningún proyecto	Se informa al usuario que no ha registrado aún ningún proyecto	OK
2	Alumno	Se utiliza el comando tras haber registrado correctamente un TFG en el sistema	Se proporciona al alumno sus datos y los del proyecto que está realizando, junto a los de su tutor	Se proporciona al alumno sus datos y los del proyecto que está realizando, junto a los de su tutor	OK
3	Tutor	Se utiliza el comando	Se informa al tutor que el comando sólo está disponible para los alumnos	Se informa al tutor que el comando sólo está disponible para los alumnos	OK

Figura 7.3: Plan de pruebas para el comando /checkData

7.4. Comando /FAQ

Número	Usuario	Descripción	Resultado esperado	Resultado obtenido	Verificación
1	Sin registro	Se utiliza el comando sin haber interactuado con el bot previamente	Se muestra el listado de preguntas en español y se permite seleccionar cualquiera de ellas	Se muestra el listado de preguntas en español y se permite seleccionar cualquiera de ellas	OK
2	Sin registro	Tras caso de prueba 1, se selecciona una pregunta	Se obtiene la respuesta a la pregunta seleccionada en español	Se obtiene la respuesta a la pregunta seleccionada en español	OK
3	Alumno	Se utiliza el comando tras haber registrado un TFG en español	Se muestra el listado de preguntas en español y se permite seleccionar cualquiera de ellas	Se muestra el listado de preguntas en español y se permite seleccionar cualquiera de ellas	OK
4	Alumno	Tras caso de prueba 3, se selecciona una pregunta	Se obtiene la respuesta a la pregunta seleccionada en español	Se obtiene la respuesta a la pregunta seleccionada en español	OK
5	Alumno	Se utiliza el comando tras haber registrado un TFG en otro idioma	Se muestra el listado de preguntas en el idioma del proyecto y se permite seleccionar cualquiera de ellas	Se muestra el listado de preguntas en el idioma del proyecto y se permite seleccionar cualquiera de ellas	OK
6	Alumno	Tras caso de prueba 5, se selecciona una pregunta	Se obtiene la respuesta a la pregunta seleccionada en el idioma del proyecto	Se obtiene la respuesta a la pregunta seleccionada en el idioma del proyecto	OK
7	Tutor	Se utiliza el comando	Se muestra el listado de preguntas en el idioma que tiene el tutor en base de datos y se permite seleccionar cualquiera de ellas	Se muestra el listado de preguntas en el idioma que tiene el tutor en base de datos y se permite seleccionar cualquiera de ellas	OK
8	Tutor	Tras caso de prueba 7, se selecciona una pregunta	Se obtiene la respuesta a la pregunta seleccionada en el idioma del tutor	Se obtiene la respuesta a la pregunta seleccionada en el idioma del tutor	OK

Figura 7.4: Plan de pruebas para el comando /FAQ

7.5. Comando /addFAQ

Número	Usuario	Descripción	Resultado esperado	Resultado obtenido	Verificación
1	Sin registro	Se utiliza el comando sin haberse registrado en el sistema	Se informa al usuario que únicamente los tutores pueden utilizar el comando	Se informa al usuario que únicamente los tutores pueden utilizar el comando	OK
2	Alumno	Se utiliza el comando	Se informa al usuario que únicamente los tutores pueden utilizar el comando	Se informa al usuario que únicamente los tutores pueden utilizar el comando	OK
3	Tutor	Se utiliza el comando	El bot inicia una conversación y solicita el texto de la pregunta en español	El bot inicia una conversación y solicita el texto de la pregunta en español	OK
4	Tutor	Tras el caso de prueba 3, se envía la respuesta en texto	Se recibe y se solicita el resto de la información hasta finalizar la conversación.	Se recibe y se solicita el resto de la información hasta finalizar la conversación.	OK
5	Tutor	Tras el caso de prueba 3, se envía la respuesta en otro formato (imagen, documento, sticker...)	Dependiendo del formato, o se ignora o se informa que debe responder a la pregunta con lo que se ha solicitado	Dependiendo del formato, o se ignora o se informa que debe responder a la pregunta con lo que se ha solicitado	OK
6	Tutor	Tras el caso de prueba 3, se intenta ejecutar otro comando	Se informa al usuario que debe finalizar o cancelar la conversación actual	Se informa al usuario que debe finalizar o cancelar la conversación actual	OK

Figura 7.5: Plan de pruebas para el comando /addFAQ

7.6. Comando /registerTFG

Número	Usuario	Descripción	Resultado esperado	Resultado obtenido	Verificación
1	Alumno	Se utiliza el comando	Se informa al alumno que no puede registrar más de un TFG	Se informa al alumno que no puede registrar más de un TFG	OK
2	Tutor	Se utiliza el comando	Se informa al tutor que únicamente los alumnos pueden registrar un TFG	Se informa al tutor que únicamente los alumnos pueden registrar un TFG	OK
3	Sin registro	Se utiliza el comando	El bot inicia una conversación y solicita el título del proyecto	El bot inicia una conversación y solicita el título del proyecto	OK
4	Sin registro	Tras el caso de prueba 3, se responde en un formato que no es el esperado (imagen, documento, sticker...)	Dependiendo del formato, se ignora o se solicita que responda en formato texto	Dependiendo del formato, se ignora o se solicita que responda en formato texto	OK
5	Sin registro	Tras el caso de prueba 3, se responde con un texto)	Se asigna el título al TFG y se solicita el resto de la información hasta finalizar la conversación.	Se asigna el título al TFG y se solicita el resto de la información hasta finalizar la conversación.	OK
6	Sin registro	En el paso de seleccionar el idioma, se intenta responder o enviar archivos	Dependiendo del formato, se ignora o se solicita que responda en formato texto	Dependiendo del formato, se ignora o se solicita que responda en formato texto	OK
7	Sin registro	En el paso de seleccionar un tutor, se intenta responder o enviar archivos	Dependiendo del formato, se ignora o se solicita que responda en formato texto	Dependiendo del formato, se ignora o se solicita que responda en formato texto	OK
8	Sin registro	En el paso de seleccionar un tutor, no se consigue recuperar ningún tutor de base de datos para mostrar al usuario	Se informa al usuario que no hay tutores registrados en la aplicación y se cancela la conversación	Se informa al usuario que no hay tutores registrados en la aplicación y se cancela la conversación	OK

Figura 7.6: Plan de pruebas para el comando /registerTFG

7.7. Envío de un diagrama de Gantt

Número	Usuario	Descripción	Resultado esperado	Resultado obtenido	Verificación
1	Sin registro	Se envía un diagrama de Gantt	Se informa al usuario que aún no ha registrado un TFG	Se informa al usuario que aún no ha registrado un TFG	OK
2	Tutor	Se envía un diagrama de Gantt	Se informa al tutor que únicamente los alumnos pueden enviar diagramas	Se informa al tutor que únicamente los alumnos pueden enviar diagramas	OK
3	Alumno	Se envía un diagrama exportado desde GanttProject sin errores	El bot analiza y guarda el diagrama e informa al usuario del éxito en el proceso	El bot analiza y guarda el diagrama e informa al usuario del éxito en el proceso	OK
4	Alumno	Se envía un diagrama con algún error interno en el formato	El bot analiza el diagrama e informa al usuario de que el formato es incorrecto. También solicita que lo reintente tras exportarlo de nuevo	El bot analiza el diagrama e informa al usuario de que el formato es incorrecto. También solicita que lo reintente tras exportarlo de nuevo	OK
5	Alumno	Se envía un documento en otro formato	Se informa al usuario que sólo se aceptan diagramas en formato gan	Se informa al usuario que sólo se aceptan diagramas en formato gan	OK
6	Alumno	Se envía un diagrama pero hay un error al recuperarlo de Telegram	Se informa al usuario sobre el error al recuperar el documento desde la nube de Telegram	Se informa al usuario sobre el error al recuperar el documento desde la nube de Telegram	OK

Figura 7.7: Plan de pruebas para el envío de un diagrama de Gantt

8. Conclusiones

A lo largo de todo el proyecto se ha ralentizado e incluso pausado el desarrollo por causas personales en varias ocasiones, aumentando por tanto de manera equivalente la carga de trabajo en los periodos de mayor disponibilidad. Por este motivo, su desarrollo se ha alargado en el tiempo más de lo previsto.

La planificación inicial, en lo que a estimación de las horas necesarias para cada tarea se refiere (véase figura 2.11), no se ha desviado en exceso de la realidad. Si bien es cierto que se han sufrido ciertos retrasos en algunas de las tareas por la complejidad que éstas suponían, otras han sufrido el trato contrario, llegando al resultado deseado antes de lo previsto.

Pese a que, como será comentado a continuación, no se han podido cumplir todos los objetivos establecidos, se ha cumplido el objetivo principal de rehacer la aplicación en un nuevo entorno, asentando un sólida base para las expansiones que se realicen en el futuro.

Como se establecía en los objetivos principales (véase sección 2.1) se ha rediseñado la base de datos, así como implementado de nuevo el código de la aplicación. Respecto a las funcionalidades que se querían añadir, se han llegado a desarrollar las siguientes: consulta de información relacionada con los TFG mediante un listado de preguntas frecuentes, registro de un TFG en el sistema por parte del alumno con posterior validación por parte del tutor y, finalmente, análisis y desglose, así como almacenamiento en base de datos, de los diferentes diagramas de Gantt que el alumno aporte en formato gan.

Como se puede apreciar, ha quedado sin implementar la creación y envío de notificaciones periódicas basadas en los hitos marcados por el alumno en su planificación. Esta tarea será abordada a continuación en la sección de trabajo futuro.

Pese a que no ha sido posible cumplir con uno de los objetivos principales, se han cumplido algunos de los objetivos secundarios: soporte multidioma para español, euskera e inglés en el FAQ y el comando de Ayuda, así como la posibilidad de añadir nuevos ítems al FAQ, funcionalidad limitada a los

tutores.

En este capítulo, último de la memoria y con el que concluyen los estudios de Grado en Ingeniería Informática de Gestión y Sistemas de Información del alumno Mikel Abad Grande, se realiza una comparación entre la planificación estimada y la que finalmente ha sido en realidad. También se lleva a cabo un análisis de trabajo futuro que puede hacerse sobre el presente proyecto así como una reflexión personal.

8.1. Análisis entre planificación estimada y real

Dado que realizar una estimación exacta es prácticamente imposible, este caso no ha sido una excepción. Se han sufrido tanto retrasos como adelantos en los tiempos estimados para ciertas tareas, y aunque no se ha podido finalizar el proyecto a tiempo para la primera fecha objetivo, se ha llegado a tiempo de la segunda.

A continuación se muestra una comparación entre la planificación inicial y los tiempos reales.

Tarea	Duración estimada	Duración real
1. Gestión y planificación	**205 horas**	**218 horas**
1.1 Documentación	180 horas	200 horas
1.2 Comunicación con el director	10 horas	8 horas
1.3 Preparación de la defensa	15 horas	10 horas
2. Estudio previo	**27 horas**	**44 horas**
2.1 Análisis funcional previo	15 horas	22 horas
2.2 Revisión del código	6 horas	10 horas
2.3 Pruebas	6 horas	12 horas
3. Formación	**8 horas**	**10 horas**
3.1 GanttProject	2 horas	1 horas
3.2 Node.js	3 horas	3 horas
3.3 Telegraf	3 horas	6 horas
4. Captura de requisitos	**25 horas**	**6 horas**
4.1 Casos de uso	10 horas	1 horas
4.2 Modelo de dominio	15 horas	5 horas
5. Análisis	**35 horas**	**22 horas**
5.1 Análisis del proyecto	20 horas	15 horas
5.2 Elección de herramientas y tecnologías	10 horas	7 horas
6. Diseño	**60 horas**	**32 horas**
6.1 Diseño de las conversaciones	10 horas	10 horas
6.2 Diseño de la base de datos	15 horas	7 horas
6.3 Diseño diagrama de clases	15 horas	5 horas
6.4 Diseño diagramas de secuencia	20 horas	10 horas
7. Implementación	**168 horas**	**105 horas**
7.1 Implementación de la base de datos	12 horas	6 horas
7.2 Creación del proyecto	5 horas	3 horas
7.3 Creación del bot	1 horas	1 horas
7.4 Implementación de las conversaciones	25 horas	20 horas
7.5 Implementación de los servicios	45 horas	35 horas
7.6 Implementación de los flujos	80 horas	40 horas
8. Verificación y pruebas	**25 horas**	**32 horas**
8.1 Pruebas	20 horas	30 horas
8.2 Validación con el cliente	5 horas	2 horas
TOTAL	**547 horas**	**469 horas**

Figura 8.1: Comparación de la estimación inicial y los tiempos reales

Como podemos observar, hay algunas tareas que se estimaron bastante por encima del tiempo necesario, mientras que en general ninguna ha llevado mucho más de lo estimado inicialmente. El cómputo total de horas reales ha sido de 469 horas, de las 547 horas estimadas inicialmente. Esto supone una reducción del 15 % de las horas necesarias, más por las razones comentadas anteriormente la finalización del proyecto se ha dilatado en el tiempo hasta la segunda fecha de entrega.

8.2. Trabajo futuro

De cara a las futuras versiones de IkastenBot existen múltiples mejoras y complementos que se pueden llevar a cabo, a continuación se explican algunos de ellos.

Envío de notificaciones

Comenzamos con el objetivo que no se ha podido cumplir en esta versión, el envío de notificaciones periódicas. El objetivo es enviar recordatorios a los alumnos cuando se acercan fechas importantes marcadas por ellos mismos en los diagramas de Gantt.

Partiendo de la versión actual, el momento ideal de generar estas notificaciones y almacenarlas en base de datos sería al recibir un diagrama de Gantt, ya que toda su información es extraída y se tiene en memoria de la sesión. Dicho esto, y dado que actualmente se está guardando dicha información en base de datos, podría recuperarse y generar las notificaciones en cualquier otro momento.

De cara al envío de las mismas, se propone utilizar un administrador regular de procesos (e.g. cron[1]) que ejecute un scrip diseñado expresamente para enviar los mensajes por Telegram.

La información que se considera necesaria por cada aviso es la siguiente:

- Alumno que recibirá la información (id de Telegram)
- Mensaje a incluir en el aviso
- Fecha en la que enviar el aviso
- Indicador sobre si el aviso ya ha sido enviado

[1] En el sistema operativo Unix, cron es un administrador regular de procesos en segundo plano que ejecuta procesos o guiones a intervalos regulares.

Interacción con las notificaciones

Una de las funcionalidades más interesantes de cara al sistema de notificaciones, era la posibilidad de responderle al bot indicando si se iba a llegar a tiempo a la tarea o, por el contrario, iba a sufrir un retraso. En caso de haber un retraso respecto a la fecha inicial podría indicarse de cuantos días, de tal modo que el bot reajustase automáticamente la información del diagrama de Gantt almacenado en función de la nueva fecha.

Adicionalmente, el bot enviaría al alumno el diagrama de Gantt actualizado, así como una captura de la previsualización.

Para llevar a cabo esta tarea sería necesario crear un nuevo diagrama de Gantt, o actualizar el último disponible en base de datos, así como invalidar las notificaciones pendientes y crear unas nuevas.

Corrección de la memoria

Marcado como uno de los objetivos secundarios, una funcionalidad a añadir podría ser la corrección ortográfica y gramatical de la memoria. El alumno enviaría el documento por Telegram, al igual que los diagramas de Gantt, y el bot, en principio mediante un módulo externo especializado en la tarea, realizaría un análisis de la misma.

Tras obtener los resultados, se presentarían al alumno en diferentes mensajes indicando, junto con sus debidas explicaciones, la localización de cada uno de los problemas, de cara a facilitar su solución. En la medida de lo posible se adjuntarían también capturas, consejos y referencias.

Análisis de plagio

Al igual que el comando anterior comenzaría por enviarse el documento de la memoria, mas esta vez sería el tutor quien enviara el documento al bot, de cara a ser analizado para comprobar posibles plagios.

Para ello debería implementarse la herramienta, o buscar alguna ya existente, que analizase la memoria así como otros proyectos de carácter público para buscar similitudes entre ellos. Esto podría incluir, además de otros proyectos, imágenes que se hayan utilizado sin los debidos derechos de autor.

En caso de hallarse alguna coincidencia sospechosa, se le comunicaría al tutor, o directamente al alumno en caso de así desearlo.

Panel de administración

Se plantea el desarrollo de un panel de administración vía aplicación web para visualizar y controlar, por parte del administrador del sistema o los tutores autorizados, la información guardada en base de datos.

En la versión actual tanto las preguntas frecuentes iniciales como la lista de tutores son introducidas manualmente en la base de datos. Esto podría realizarse de una manera mucho más intuitiva a través de un panel de administración, así como los futuros cambios y el mantenimiento de los datos.

8.3. Reflexión personal

En lo personal, es necesario remontarse bastante tiempo atrás para evaluar todo el proceso, ya que antes de comenzar con este proyecto intenté realizar otro que acabó con mi motivación. Cuando Juanan me presentó la idea de cambiar a IkastenBot fue una decisión muy fácil de tomar, con lo que las ganas de empezar y continuar volvieron a estar presentes.

La experiencia del proyecto ha sido muy satisfactoria en muchos sentidos. Aunque ha sido bastante complicado por la dificultad que ha supuesto dedicarle el tiempo necesario, dado que en la totalidad del proyecto me hallaba trabajando a jornada completa, he podido aprender mucho de él, no sólo en el ámbito técnico.

Cabe mencionar que haber podido desarrollar una aplicación en un lenguaje como Node, moderno y con presencia en el mercado, así como un bot de Telegram, una de las aplicaciones de mensajería más utilizada en la actualidad, me ha permitido adquirir una serie de conocimiento que, más allá de ser útiles en lo profesional, también me interesan a nivel personal.

Anexos

Acrónimos

DER Diagrama Entidad-Relación. 21, 47

EDT Estructura de Descomposición del Trabajo. 8, 10, 11, 13, 15, 17, 18, 20, 23, 25

FAQ Frequently Asked Questions. 6, 79

SO Sistema Operativo. 9

TFG Trabajo de Fin de Grado. 2, 3, 5, 39, 41, 42, 50, 59, 67, 68, 79

Glosario

API Siglas de "interfaz de programación de aplicaciones" ("application programming interface" en inglés), es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizada por otro software como una capa de abstracción. 23, 65

back-end También llamado comúnmente "servidor" hace referencia a la capa de acceso a datos en la arquitectura de software. 7

chatbot Conocidos también como "bot conversacionales" son un tipo de software que, para cumplir sus objetivos como aplicación, simulan una conversación con una persona. 1, 2, 39

diagrama de Gantt Herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. 5, 8, 28, 42, 71, 79, 82, 83

Express Express es un framework para la parte backend de aplicaciones web en Node.js, publicado de manera gratuita y de código abierto bajo Licencia MIT. Está diseñado para aplicaciones web y APIs. 67

feedback Referida de forma común como retroalimentación, es un mecanismo por el cual una cierta proporción de la salida de un sistema se redirige a la entrada, con señales de controlar su comportamiento. 39

framework Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. 5, 16, 39, 66, 67

gan Formato utilizado para los proyectos creados por el software GanttProject. 5, 42, 71, 79

hosting En castellano "alojamiento web", es el servicio que provee a los usuarios de Internet un sistema para poder almacenar información accesible vía web. 7

JSON Acrónimo de Notación de Objeto de JavaScript (en inglés JavaScript Object Notation), es un formato de texto sencillo para el intercambio de datos. 71

ngrok Herramienta que permite exponer públicamente una URL segura, generada dinámicamente, que apunta a un servicio web alojado en nuestro ordenador. 25, 26, 67

script Conocidos también como "secuencia de comandos" o "guion" es un término informal que se usa para designar a un programa relativamente simple. 36

Telegraf Framework de la Bot API de Telegram para Node.js. 65, 67, 68

tethering Proceso por el cual un dispositivo móvil con conexión a Internet actúa como pasarela para ofrecer acceso a la red a otros dispositivos. 37

token El token es una referencia (un identificador) que regresa a los datos sensibles a través de un sistema de tokenización. 63, 64

webhook Método de alteración del funcionamiento de una página o aplicación web, con retrollamada personalizadas. 8

XML Siglas en inglés de eXtensible Markup Language, traducido como "Lenguaje de Marcas Extensible", es un metalenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. 71

Bibliografía

[1] Amaia De Pablo Ruiz. "@ikastenbot, GRAL bat garatzeko laguntzailea". En: (30 de oct. de 2018). DOI: 10/29398. URL: https://addi.ehu.es/handle/10810/29398.

[2] Brian Dean. *How Many People Use Telegram in 2021? 55 Telegram Stats*. https://backlinko.com/telegram-users. 2021.

[3] Danica Jovic. *The Future Is Now - 37 Fascinating Chatbot Statistics*. https://www.smallbizgenius.net/by-the-numbers/chatbot-statistics/. 2020.

[4] Kai Lei, Yining Ma y Zhi Tan. "Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js". En: *2014 IEEE 17th International Conference on Computational Science and Engineering*. 2014, págs. 661-668. DOI: 10.1109/CSE.2014.142.

[5] David Pérez Gómez. "Ikastenbot 2.0". En: (27 de nov. de 2019). DOI: 10/36555. URL: https://addi.ehu.es/handle/10810/36555.