Universidade Vigo

Escola Superior de Enxeñaría Informática

Memoria del Trabajo de Fin de Grado que presenta

Álvaro Vidal López

para la obtención del Título de Graduado en Ingeniería Informática

Chatbot de Telegram para la creación de imágenes con IA

Junio, 2023

Trabajo de Fin de Grado Nº: Tutor/a: Daniel González Peña

Área de conocimiento: Lenguajes y Sistemas Informáticos

Departamento: Informática

Dedicatoria

A mis padres y mi hermana, por apoyarme con todo,

hasta en los momentos más difíciles.

Agradecimientos

Muchas gracias a mi tutor Daniel

por toda su ayuda y atención.

Muchas gracias también a todos los amigos que he hecho durante el periodo de estudios en la universidad

y en especial a mi pareja, Elena, por hacer

todo este camino más fácil.

Índice de contenidos

Índice de figuras

1. Introducción

Hoy en día estamos viviendo una época de gran crecimiento tecnológico. En particular estamos viendo una gran evolución en el mundo de las IAs (Inteligencias Artificiales), chatGPT, creación de imágenes, reconocimiento de voz, reconocimiento facial, aplicaciones médicas[1], y muchos más modelos son implementados y mejorados día a día. Es por esto que este tipo de herramientas están cada vez más presentes en la vida cotidiana de las personas que tienen acceso a ellas.

Como la mayoría de avances tecnológicos, su uso puede ser tanto para fines positivos como negativos, y el gran potencial que nos proporcionan las IAs no es una excepción. Por ello es responsabilidad tanto de la gente que las usa como de la gente que desarrolla aplicaciones sobre las mismas su uso responsable.

Es complicado integrar una IA en tus tareas diarias para un usuario sin conocimiento sobre las mismas. Por ello hay un gran interés en, no sólo crear aplicaciones que se basen en el uso de las IAs para que el usuario tenga acceso a todos los beneficios que esta pueda aportar, sino también en mejorar aplicaciones ya existentes, de esta forma es posible acelerar procesos, mejorar sistemas de recomendación y o adaptar las aplicaciones del día a día a cada cliente de forma particular.

En este sentido, este Trabajo de Fin de Grado (TFG) propone facilitar la comunicación mediante una aplicación de mensajería instantánea como Telegram incluyendo la posibilidad de generar cualquier imagen a partir de una descripción cualquiera [2] mediante el uso de una IA, y poderla incorporar fácilmente a una conversación en curso[3].

Esta herramienta permite mejorar la comunicación entre usuarios. Es más fácil visualizar una idea con imágenes que con palabras, y aún más fácil si puedes plasmar esa idea con una imagen sin tener que depender de que alguien hubiese tenido la misma idea que tú y decidió transformarla en una imagen. Con esta herramienta cualquier idea que imagines puede ser trasladada a una imagen y enviarla por mensaje sin tener ni siquiera que salirse de la misma aplicación que usas para mandar dicho mensaje, haciendo que esta sea muy útil y cómoda.

Para ello, este trabajo trata de la creación e implementación un Chatbot de Telegram, similar a otros existentes, como por ejemplo PDFbot [4]. Este bot recibirá la frase, también conocido como “prompt” que se quiere plasmar en una imagen y este responderá enviando la imagen generada por el chat

1.1. Objetivos

En este apartado se detalla cuales son los objetivos marcados para este bot de software. Los objetivos han sido separados en dos bloques, principal y específicos, cada uno de ellos se han tenido en cuenta durante todo el desarrollo.

1.1.1. Principal

Como objetivo principal está la integración de una IA de creación de imágenes a partir de texto con Telegram mediante un Chatbot, facilitando así la comunicación entre usuarios y acercando la IA a cualquier persona sin requerir conocimiento de su funcionamiento.

1.1.2. Específicos

- **Creación de imágenes en tiempo real a partir de un texto.** Para ello se hace uso de una una IA, en este caso Stable-Diffusion [5], en una aplicación como Telegram mediante el uso de un bot de chat. De esta manera el usuario tendrá la imagen de respuesta a través de un chat de Telegram y no tendrá que ir a webs externas, descargar la foto etc.

- **Integración de la generación de imágenes en Telegram.** De esta forma, se puedan cursar solicitudes a la IA de forma remota por parte de múltiples usuarios, con la idea de crear una herramienta útil, ya que de nada sirve tener una herramienta que funciona muy bien o es muy sofisticada si no tiene un uso útil o real para el público que está diseñada..

- **Control de peticiones.** Para poder brindar una experiencia de usuario agradable, las peticiones deben estar controladas de dos formas diferentes:
    - Se debe evitar que un usuario pueda monopolizar la aplicación de esta forma un usuario que realice muchas peticiones perderá prioridad en la cola de creación de imágenes frente a un usuario que solicite una nueva petición.
    - Se debe poder enviar peticiones al bot aunque este esté trabajando en la creación de una imagen, de esta forma el bot siempre estará operativo para recoger peticiones.

- **Creación de una tecnología útil para el usuario.** Como último objetivo tenemos el crear una herramienta útil, de nada sirve tener una herramienta que funciona muy bien o es muy sofisticada si no tiene un uso útil o real para el público que está diseñada.

1.2. Resumen

Para hacer posible la creación de una herramienta que integre la tecnología de la que se habla en la introducción, un bot capaz de generar imágenes con una IA y poder usarlo en una aplicación de chat, decidimos que la mejor plataforma para integrar sería Telegram frente a otras aplicaciones de chat de texto por las razones que vamos a exponer a continuación..

En primer lugar, Telegram [6] goza de una gran popularidad a nivel mundial, contando con una extensa base de usuarios activos. Esto significa que un bot desarrollado en esta plataforma tiene un gran potencial para llegar a una amplia y diversa audiencia.

Además, Telegram proporciona una API completa y bien documentada [7] que facilita el proceso de desarrollo de bots de chat. Esta API ofrece una amplia gama de funcionalidades, permitiendo el envío de mensajes multimedia, la gestión de grupos y canales, así como la interacción con teclados personalizados, entre otras opciones avanzadas.

Asimismo, Telegram destaca por sus características avanzadas que permiten la construcción de bots poderosos. Entre ellas se incluyen la capacidad de enviar mensajes enriquecidos con

contenido multimedia, la implementación de comandos personalizados y la interacción con botones para respuestas interactivas.

La seguridad y privacidad son otros aspectos destacados de Telegram. La plataforma brinda opciones de cifrado de extremo a extremo, autodestrucción de mensajes y la posibilidad de crear chats secretos. Esto resulta particularmente relevante cuando se manejan datos sensibles o confidenciales en un bot.

En Telegram existen diferentes tipos de bots, para la implementación de esta herramienta se ha usado un chatbot, que funciona como un usuario más de Telegram que interactúa contigo enviandole mensajes. De esta manera puedes tener una chat donde crear fotos, ya sea para ti solo, o enviarlas de forma sencilla a cualquier otro chat. Además es posible añadir este bot a un grupo, de esta manera todos los integrantes del grupo tienen acceso a él de manera simultánea en esa misma ventana de chat.

El funcionamiento del bot ,teniendo en cuenta las cuestiones anteriores, es el siguiente: Primero tenemos que suscribirnos al mismo mediante Telegram, una tarea sencilla a mano de cualquier usuario. Una vez suscritos al bot este nos indicará cómo usarlo con un mensaje donde se explica qué comando hay como usarlos y para qué funcionan.

A nivel de programación, cada vez que un usuario envía una petición al bot este le la resuelve, la forma de resolverlo depende de qué comando se indique. El comando principal es el de creación de imágenes y este funciona de la siguiente manera:

- El bot recibe la información y la almacena en una cola.
- Cuando hay peticiones en la cola se le envía la información al modelo de IA y se crea la imagen.
- Una vez la imagen es creada se le envía al usuario.

Este sería el flujo de trabajo, a modo de resumen, del cómo se crean las imágenes. El resto de comandos son manejados por el propio bot y solo requieren del funcionamiento del mismo, el bot recibe un comando, lo atiende y le envía una confirmación al usuario.

1.3. Metodología

Para la realización de este trabajo,se llegó a la conclusión de que la mejor metodología para poder ponerlo en práctica sería la Scrum. Scrum es una metodología ágil, una de las más utilizadas en el sector del desarrollo de software [8].

Scrum se basa en un enfoque flexible y adaptable para el desarrollo de productos y servicios. A diferencia de las metodologías tradicionales que intentan predecir el rumbo del proyecto, Scrum enfatiza la necesidad de flexibilidad y la capacidad de aceptar cambios para adaptarse a las circunstancias que podrían cambiar con el paso del tiempo.

En lugar de seguir un proceso estandarizado y rígido, Scrum se define como un marco de trabajo que se adapta a las necesidades del equipo de desarrollo. Scrum utiliza un enfoque iterativo e incremental, donde cada iteración se lleva a cabo en un período de tiempo fijo y con características completas.

Los principales ítems utilizados en Scrum son el Backlog del Producto (Product Backlog) y el Backlog del Sprint (Sprint Backlog). El Backlog del Producto es la guía de aquellas

funcionalidades que deben ser implementadas, y puede modificarse a lo largo del proyecto según las necesidades que surjan. A diferencia del Backlog del Producto, el Backlog del Sprint contiene las funcionalidades específicas a implementar en el sprint correspondiente, como su propio nombre indica, en tareas más pequeñas para facilitar su desarrollo.

Además de los ítems comentados con anterioridad, Scrum utiliza otros elementos como paneles de tareas, que aportan un enfoque más amplio al seguimiento del proyecto. Para hacer más fácil la recopilación de los requisitos del proyecto, Scrum utiliza las historias de usuario, en ellas se encuentra condensada todas las funcionalidades y especificaciones, definidas por el cliente, que tiene que tener el producto final.

Otro de los puntos importantes y característicos de Scrum son los sprints. Estos son iteraciones con una duración fija en las cuales se realiza el trabajo, estableciendo la Definición de Hecho que determina los criterios para considerar un requisito finalizado. Otras actividades incluyen la reunión diaria, la Revisión del Sprint y la Retrospectiva del Sprint, que permiten controlar y mejorar el proceso.

Entre las actividades distintivas de Scrum se encuentra el Grooming del Backlog del Producto, donde se crean, retocan y priorizan los requisitos conseguidos en los pasos anteriores.

En cuanto a los roles en Scrum, se pueden separar en los siguientes:

- El Product Owner es el responsable de decidir qué funcionalidades se desarrollarán, como y el orden de las mismas.
- El Scrum Master es el encargado de orientar y dirigir el proyecto.
- El Equipo de Desarrollo, conformado por las personas responsables del diseño, construcción y prueba del producto.

Pese a que la organización del proyecto se basa en Scrum, se han tenido que realizar pequeños cambios a esta metodología para adaptarla al mismo [9].

Se podría decir que en el equipo de desarrollo es representado por el alumno, Álvaro Vidal López, mientras que el Scrum Master y el product owner sería el tutor Daniel Gonzalez Peña. Las historias de usuario no las ha diseñado el cliente si no que han sido tanto el tutor como el alumno los que las han diseñado.

Para poder hacer un seguimiento de los Sprints se ha optado por crear tablas y diagramas de Gantt, tanto para la estimación inicial como para el el tiempo real de trabajo se ha usado ambas, de esta forma se puede comparar sencillamente cuánto difiere el tiempo real del estimado.

Dado que ni el alumno ni el tutor no tenían una disponibilidad total para dedicarle al proyecto , las reuniones diarias se reformularon como reuniones cada 2 semanas. De esta forma el alumno tenía tiempo de realizar avances sustanciales en el proyecto, haciendo así mucho más productivas las sesiones de reunión.

Partiendo con Scrum como base de organización y los cambios realizados, el enfoque ágil resulta realmente beneficioso para las necesidades de un proyecto como este. Hay que tener en cuenta las restricciones por tiempo, por eso reducir el riesgo a largo plazo es algo

verdaderamente importante en proyectos tan ajustados, a esto hay que sumarle la gran facilidad que da este tipo de desarrollos para implementar cambios, que en un primer gran proyecto como es este para el alumno están a la orden del día. Todos estos factores, junto a todos los vistos anteriormente, reafirman que el uso de una metodología ágil como Scrum es una muy buena opción.

2. Planificación

En este apartado se expondrá tanto la planificación inicial como el tiempo real que ha conllevado el desarrollo de este trabajo.

Para poder hacer una estimación del tiempo que nos va a llevar este trabajo, lo primero es separarlo en diferentes tareas y luego asignarle un tiempo aproximado dependiendo de la dificultad de la misma. En este caso la disponibilidad de tiempo no era completa para el trabajo ya que hay que contar con el tiempo a dedicar a otras asignaturas.

En la siguiente tabla se muestra la planificación inicial de tareas y tiempo. Esta planificación es una aproximación que puede funcionar como guía para saber el estado de nuestro proyecto en cada momento y así poder tener una mejor organización respecto a la realización del mismo. Para que la organización del tiempo sea más visual se han realizado diagramas de Gantt [10] (Figura 1).

2.1. Inicial

Sprints	**F.Inicio**	**F.Finalización**	**T.Horas**
1. Spike Tecnológico: Telegram y Stable Diffusion	1 de marzo	28 de marzo	**28 días 60h**
2. Implementación inicial del bot	29 de marzo	25 de abril	**28 días 60h**
3. Gestión de peticiones concurrentes	26 de abril	16 de mayo	**21 días 50 h**
4. Integración de Stable Diffusion	17 de mayo	31 de mayo	**14 días 50h**
5. Adición de funcionalidades y corrección de errores	1 de junio	13 de junio	**14 días 50h**
6. Completar la documentación	14 de junio	22 de junio	**9 días 30h**
Total	1 de marzo	22 de junio	**114 días 300h**

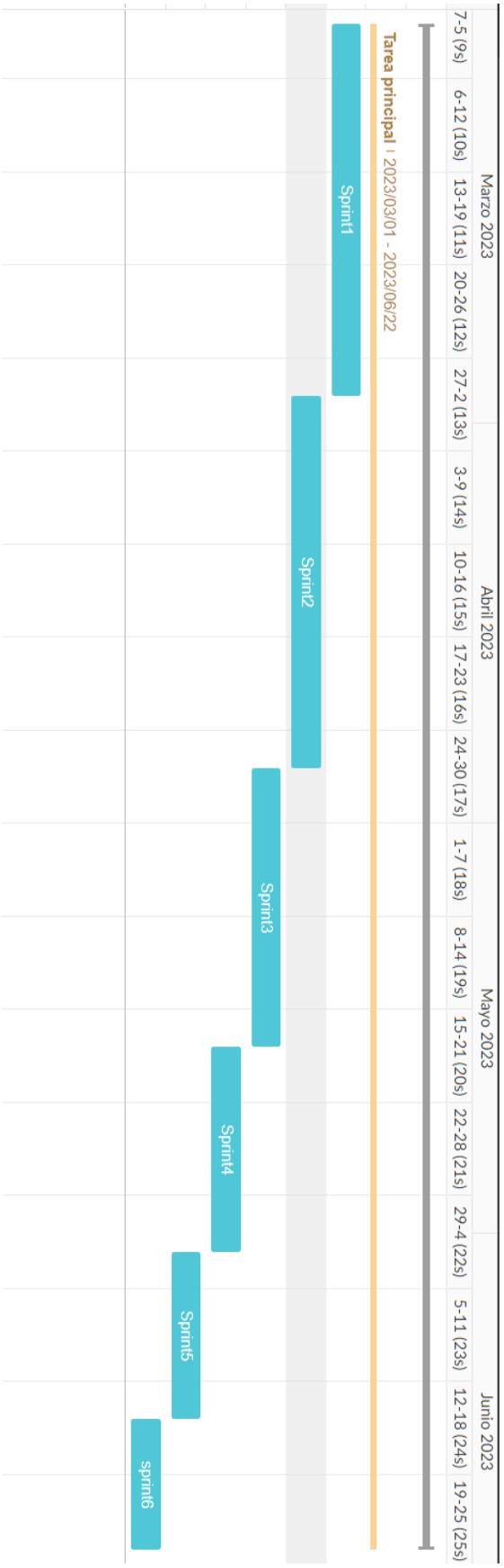


Figura 1. Diagrama de Gantt del tiempo de trabajo estimado.

2.2. Seguimiento

Finalmente el tiempo del proyecto no fue demasiado diferente, dado a la imposibilidad de entregar el trabajo en junio por motivos académicos. Es por ello que las fechas y el tiempo dedicado han cambiado sobre todo en los últimos aspectos del trabajo, mientras qué, hasta junio, la planificación fue según la asignada principalmente. Pese a la dilatación en el tiempo, el número de horas ha sido prácticamente el mismo, simplemente se ha dilatado más entre los nuevos días disponibles.

Por ello se ha vuelto a crear una tabla con los datos reales con el tiempo y las fechas correctas, además se ha creado un nuevo diagrama de Gantt (Figura 2) para representar de forma gráfica la organización del trabajo, esta vez con el tiempo y las fechas precisas.

Dado que hasta junio no sabíamos que tendríamos este percance, el cambio más notorio se realiza a partir de este mes, dilatando el tiempo que lleva completar el apartado 5, pero no aumentando el número de horas dedicadas de forma significativa.

Es verdad que se pueden observar más variaciones respecto al tiempo estimado, pero ninguna es verdaderamente significativa, los plazos se fueron compensando y finalmente tanto la suma de las horas como los plazos hasta junio se mantuvieron relativamente estables respecto a la estimación original.

Sprints	**F.Inicio**	**F.Finalización**	**T.Horas**
1. Spike Tecnológico: Telegram y Stable Diffusion	1 de marzo	28 de marzo	**28 días 60h**
2. Implementación inicial del bot	29 de marzo	25 de abril	**28 días 60h**
3. Gestión de peticiones concurrentes	26 de abril	16 de mayo	**21 días 50 h**
4. Integración de Stable Diffusion	17 de mayo	31 de mayo	**14 días 50h**
5. Adición de funcionalidades y corrección de errores	1 de junio	28 de junio	**28 días 60h**
6. Completar la documentación	28 de junio	7 de junio	**9 días 35h**
Total	1 de marzo	22 de junio	**128 días 315h**

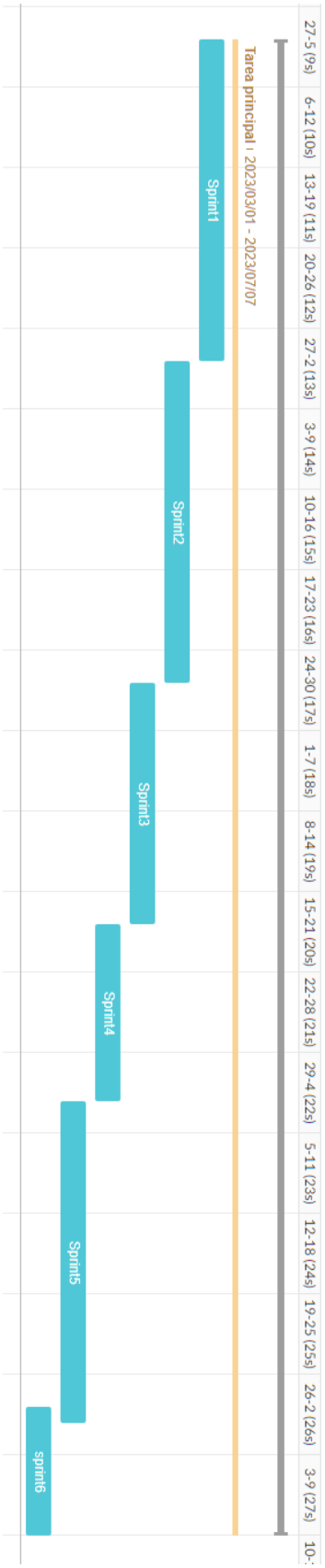


Figura 2. Diagrama de Gantt del tiempo de trabajo real.

3. Arquitectura y Tecnología

El estudio de la arquitectura en un desarrollo de software es uno de los pasos más importantes en el diseño del mismo. Cuando hablamos de arquitectura nos referimos a la vista del sistema de forma funcional, como interactúa a nivel de sistema conjunto. Es una guía de cómo debe funcionar el software muy importante para los desarrolladores que les permite entender cómo debe funcionar el proyecto. Además en el periodo de creación de la arquitectura se detallan también las tecnologías que se van a usar, de forma que se pueda tener una visión del software bien estructurada y sabiendo cómo se va a comportar.

3.1. Arquitectura del sistema

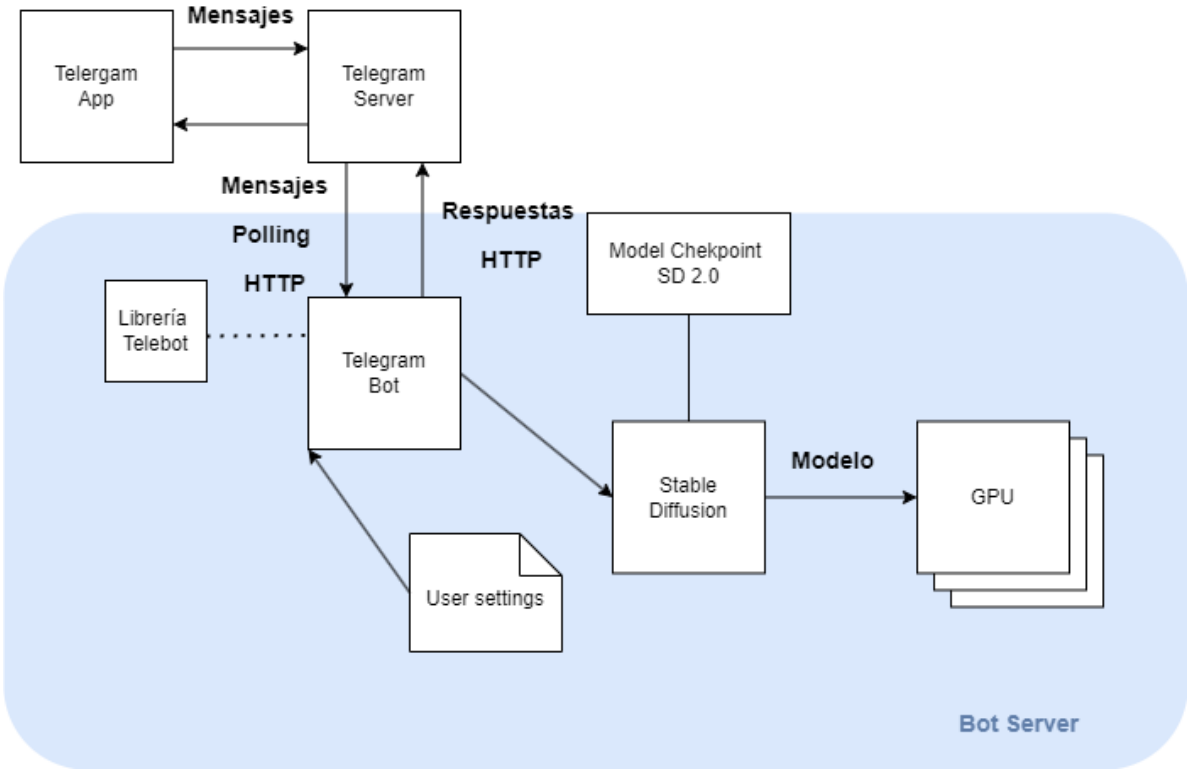


Figura 3. Diagrama de la arquitectura del sistema.

El esquema muestra la arquitectura que sigue el bot para poder dar el servicio para el que está diseñado.

Podemos ver que hay dos bloques diferenciados, el propio Bot Server, lo que sería la parte que se ha creado en este trabajo y el segundo bloque(lo que está fuera del Bot Server) que refleja lo que tiene que pasar para que el bot pueda funcionar.

Empezaremos con el bloque que está fuera del Bot Server, esta parte es muy sencilla y refleja cómo los mensajes mandados desde la aplicación de Telegram, que representa un mensaje de cualquier usuario de la misma, son recibidos en el propio servidor que da alojamiento a Telegram y una vez allí mediante el uso de polling HTTP estos son dirigidos a nuestro bot.

El bloque del Bot Server es donde se encuentra el grueso de nuestro programa. Vamos a explicar cada parte de por separado para que sea más sencillo de comprender.

En primer lugar tenemos Telegram Bot, esta parte, como su propio nombre indica, es la referida al propio bot en sí. Como se puede ver es posible utilizarla gracias a la librería Telebot donde están las herramientas necesarias para la integración con Telegram, por otro lado podemos ver que User Settings también está relacionada con este bloque, ya que las opciones que puede establecer el usuario son manejadas por el bot en cuestión.

Si seguimos hacia la derecha podremos ver que el siguiente bloque, Stable Diffusion, este bloque representa el propio modelo de la IA que utilizamos, enlazado a él está su Model Checkpoint en este caso Stable Diffusion 2.5, este checkpoint podría ser actualizado en el futuro.

Si continuamos aún más a la derecha nos encontraremos con el último bloque del diagrama, este representa las GPU´s donde el modelo es cargado. Como se puede ver esta arquitectura cuenta con más de una GPU esto es por que es posible tener el bot funcionando con más de una GPU activa pudiendo generar así imágenes de manera concurrente, tantas imágenes como GPU´s haya.

Por último tendríamos que volver al bloque Telegram Bot ya que nos había quedado una relación Telegram Server, la que une los dos bloques, esta relación explica cómo las respuestas que da el bot, tanto imágenes como mensajes son enviados al servidor de Telegram y este se encarga de enviarlos a cada chat de cada usuario.

Esta arquitectura permite que toda la parte que implica al servidor del bot esté desacoplada al 100% de los servidores de Telegram, de esta forma cualquier cambio en el bot sería completamente transparente para el mismo, haciendo que los cambios sean mucho más fáciles de realizar.

Además de lo comentado anteriormente para esta arquitectura se ha tenido en cuenta el funcionamiento interno del bot, es decir como el bot se relaciona con el resto de tecnologías, como GPU´s y modelos, de esta forma se ha creado la arquitectura de la forma más óptima posible.

3.2. Tecnologías y Herramientas de terceros

3.2.1. Tecnologías

Python

Python es un lenguaje de programación de alto nivel y de propósito general. Ofrece una amplia gama de bibliotecas y módulos que facilitan el desarrollo de diversas aplicaciones, desde programas simples hasta proyectos complejos.

Una de las características destacadas de Python es su enfoque en la facilidad de aprendizaje y uso. Gracias a su sintaxis intuitiva y su amplia documentación, los programadores pueden familiarizarse rápidamente con el lenguaje y comenzar a desarrollar aplicaciones de manera eficiente.

Python es un lenguaje interpretado, lo que significa que no se requiere una compilación previa del código. Esto permite un desarrollo ágil, donde los cambios pueden ser probados y desplegados rápidamente. Además, Python es multiplataforma, lo que significa que los

programas escritos en Python pueden ejecutarse en diversos sistemas operativos, como Windows, macOS y Linux [11].

La mayor parte del código fue desarrollado con este lenguaje.

PyTorch

PyTorch es un framework de deep learning de código abierto que se utiliza ampliamente en el campo de la IA automático. Proporciona una interfaz flexible y eficiente para construir y entrenar modelos de redes neuronales.

PyTorch se destaca por su enfoque dinámico y expresivo. A diferencia de otros frameworks, PyTorch permite a los desarrolladores definir y modificar modelos de manera interactiva durante el tiempo de ejecución. Esto facilita la experimentación y la depuración, ya que los cambios pueden realizarse de manera incremental y rápida.

Una de las características clave de PyTorch es su capacidad para realizar cómputo en paralelo utilizando unidades de procesamiento gráfico (GPU). Esto acelera significativamente el entrenamiento y la inferencia de modelos de aprendizaje profundo, lo que permite aprovechar al máximo el rendimiento de hardware especializado [12].

Se ha utilizado para la integración de la IA.

Stable Diffusion

Stable Diffusion es una IA que funciona utilizando un modelo de difusión para generar imágenes a partir del texto proporcionado por los usuarios. Cuando un usuario introduce una descripción en lenguaje natural, Stable Diffusion se encarga de realizar una interpretación exhaustiva y precisa de dicha petición. Utilizando técnicas avanzadas de IA, la IA analiza el texto proporcionado y extrae información relevante para comprender completamente las necesidades del usuario.

Stable Diffusion se apoya en un modelo de difusión altamente sofisticado. Este modelo ha sido diseñado específicamente para eliminar el ruido gaussiano presente en imágenes borrosas. Comienza con una imagen inicial que posee tanto ruido como desenfoque, y a través de iteraciones iterativas, el modelo de difusión refinado va transformando la imagen hasta obtener una versión final.

No solo se limita a generar imágenes desde cero, Stable Diffusion también puede editar imágenes existentes de acuerdo con las instrucciones proporcionadas por los usuarios. Esto implica la capacidad de agregar o eliminar objetos, modificar colores, ajustar detalles y realizar una amplia gama de modificaciones para satisfacer las preferencias y requisitos individuales.

Es la herramienta que usará el programa para generar las imágenes[13].

Entornos Conda
Un entorno de Conda es un entorno virtual creado mediante la herramienta de gestión de paquetes Conda. Conda es una herramienta utilizada en el desarrollo de software que permite gestionar de forma eficiente las dependencias y entornos de trabajo en diferentes proyectos.

En un entorno de Conda, se pueden instalar y administrar de manera independiente paquetes y librerías específicas para un proyecto en particular, sin interferir con otras instalaciones de paquetes o librerías en el sistema operativo. Esto garantiza que cada proyecto tenga su propio conjunto de dependencias, evitando conflictos y asegurando la reproducibilidad del entorno de desarrollo.

Los entornos de Conda son altamente flexibles y personalizables. Permiten crear un entorno de trabajo aislado que puede incluir diferentes versiones de Python, paquetes específicos y configuraciones personalizadas. Esto es especialmente útil cuando se trabaja en proyectos con requisitos específicos, donde se pueden tener diferentes versiones de paquetes o librerías sin afectar otros proyectos.

En el trabajo se han usado entornos de conda para todo lo referente a instalaciones de servicios, librerías y paquetes [14].

3.2.2. Software de apoyo

Chat gpt
Chat GPT (Generative Pre-trained Transformer) es un modelo de lenguaje natural desarrollado por OpenAI que utiliza técnicas de IA y aprendizaje automático para generar respuestas en conversaciones en línea. Chat GPT ha sido entrenado en grandes cantidades de datos textuales para mejorar su capacidad de comprensión del lenguaje natural.

Usado para resolución de dudas sobre uso de ciertas herramientas en Python[15].

Visual Studio
Visual Studio es un entorno de desarrollo integrado (IDE) desarrollado por Microsoft para escribir, depurar y compilar programas de software. Ofrece una amplia gama de herramientas y características para la creación de aplicaciones, incluyendo edición de código, depuración, compilación, integración de control de versiones y pruebas de software. Visual Studio admite múltiples lenguajes de programación, como C++, C#, Python, entre otros.

Editor principal para el código.

Git
Git es un sistema de control de versiones de código abierto, diseñado para rastrear y gestionar cambios en archivos de código fuente durante el proceso de desarrollo de software. Permite a los desarrolladores trabajar de manera colaborativa en proyectos de software, rastreando y comparando cambios de código y versiones anteriores, y fusionando múltiples ramas de desarrollo. Git es altamente escalable y se utiliza ampliamente en el desarrollo de software para controlar y gestionar el flujo de cambios de código fuente [16].

Herramienta usada para el control de versiones.

Documentos de drive
Herramienta de procesamiento de texto que forma parte del catálogo de Google drive. Permite el acceso desde cualquier dispositivo con internet.

Procesador de texto usado para la realización de la memoria.

Draw.io
Draw.io es una herramienta de dibujo y diagramación en línea que permite crear gráficos, diagramas y flujos de procesos de manera intuitiva y sencilla. La herramienta se puede utilizar de forma gratuita y se integra con diferentes plataformas como Google Drive.

Con esta herramienta se han creado tanto los diagramas de clases como los de secuencia.

3.2.3. Librerías de terceros

Telegram API
La API de Telegram es una interfaz de programación de aplicaciones (API) que permite a los desarrolladores interactuar con Telegram y construir aplicaciones y bots personalizados. Permite enviar y recibir mensajes, archivos multimedia, crear grupos y canales, gestionar contactos, y acceder a otros servicios y funcionalidades de Telegram. La API de Telegram es gratuita y se puede utilizar en múltiples lenguajes de programación como Python, Java, PHP, entre otros. Ha sido necesaria para la configuración del bot.

En el proyecto ha sido necesario trabajar con esta api para la creación y codificación del bot.

Dill
Dill es una biblioteca de Python que proporciona una funcionalidad de serialización extendida en comparación con el módulo estándar de pickle. Permite la serialización de una amplia gama de objetos de Python, incluidos aquellos que no son compatibles con pickle, como funciones, clases, objetos con métodos y objetos con bloqueos. Dill ofrece una solución conveniente para guardar y cargar objetos complejos, preservando su estructura y comportamiento. Además, dill es compatible con múltiples versiones de Python y ofrece una forma de extender las capacidades de serialización para casos de uso más avanzados.

En el programa fue usado para poder asegurar la consistencia de las opciones de los usuarios.

Telebot
Telebot es un framework de desarrollo de bots para Telegram basado en Python. Proporciona una interfaz sencilla y poderosa para interactuar con la API de Telegram y crear bots interactivos y personalizados. Con Telebot, los desarrolladores pueden aprovechar la funcionalidad completa de Telegram y crear bots que puedan enviar y recibir mensajes, responder a comandos, enviar archivos y medios, interactuar con usuarios y administrar grupos y canales.

Una de las características destacadas de Telebot es su interfaz de programación fácil de usar. Con una documentación completa y clara, los desarrolladores pueden aprender rápidamente cómo utilizar Telebot y comenzar a crear bots en poco tiempo. La sintaxis simple y concisa de

Telebot permite el desarrollo eficiente y rápido, lo que es especialmente útil para proyectos con plazos ajustados.

Además, Telebot ofrece una amplia gama de funcionalidades y características avanzadas. Los desarrolladores pueden utilizar la programación de respuestas automáticas para crear bots que respondan de forma inteligente a ciertos patrones de mensajes.

El framework fue usado para la construcción del bot [17] .

4. Especificación y análisis de requisitos

En esta sección se presentan los roles identificados, las diferentes historias de usuario identificadas y completadas durante el proyecto, que conforman el backlog de producto, así como los requisitos no funcionales del proyecto. Estos elementos se han generado a partir de los objetivos descritos en la sección de Objetivos del proyecto [18].

Para este proyecto he creado una sola historia de usuario épica la cual será fragmentada en diferentes historias de usuario más pequeñas.

4.1. HU01:Creación de imágenes en un chat de Telegram

Como usuario del bot
Quiero que el bot me envíe una imagen con el prompt que le he pedido generada por IA
Para poder crear la imagen que quiera.

Pruebas de aceptación

- Se debe enviar la imagen al usuario correspondiente.
- El bot debería estar operativo el mayor tiempo posible.
- El bot debe ser justo con el tratamiento de las peticiones.
- El bot debe ser eficiente a la hora de la creación de imágenes.
- El bot debe ser responsivo y tolerar errores.
- El bot debe ser capaz de aceptar peticiones sin esperar a que las peticiones pendientes terminen de procesarse.

Como se puede ver esta historia es demasiado amplia como para tratarla de forma de forma aislada. por ello la dividiremos en historias de usuario más funcionales y concretas con las que podamos ir trabajando.

4.1.1. HU01.1:Creación del chatBot

Como usuario del bot,
Quiero que el bot me acepte mi petición,
Para poder acceder a la herramienta.

Pruebas de aceptación

- Se debe aceptar la petición.
- Se puede encontrar el bot en Telegram.
- El bot responde.

4.1.2. HU01.2:Validación de la entrada del usuario

Como usuario del bot,
Quiero que el bot sea tolerante a errores y responsivo,
Para tener una mejor experiencia de usuario.

Pruebas de aceptación

- El bot deberá indicar si ha indicado de forma incorrecta el valor del prompt.

4.1.3. HU01.3:Concurrencia de peticiones

Como usuario del bot,
Quiero no tener que esperar por la aceptación de mi petición a que se realicen todas las peticiones pendientes,
Para tener feedback sobre si mi petición se está realizando.

Pruebas de aceptación

- Las peticiones se tienen que tratar en cuanto se envían al bot.
- Se debe mantener una lista actualizada de todas las peticiones pendientes.

4.1.4. HU01.4:Ordenación de peticiones por prioridad

Como usuario del bot,
Quiero que otro usuario no pueda monopolizar el bot,
Para que el uso del bot sea justo para los usuarios.

Pruebas de aceptación

- Se debe mantener una lista ordenada de las peticiones en espera .
- Si un usuario tiene muchas peticiones pendientes se debe priorizar a un usuario que ha realizado una nueva petición.

4.1.5. HU01.5:Integración de la IA

Como usuario del bot,
Quiero que funcione lo más rápido posible,
Para que los tiempos de espera sean los más cortos posibles.

Pruebas de aceptación

- El bot debe cargarse en la GPU al comenzar la ejecución.
- Debe mantenerse cargado mientras esté activo.

4.1.6. HU01.6:Comando crear

Como usuario del bot,
Quiero que se me envíe la imagen,
Para poder hacer uso de ella en cualquier chat.

Pruebas de aceptación

- El bot debe enviar la imagen al chat correcto.
- La imagen tiene que guardar relación con el promt ingresado.

4.1.7. HU01.7:Comando help

Como usuario del bot,
Quiero poder tener ayuda,
Para saber como usar el bot.

Pruebas de aceptación

- El comando debe decirle al usuario que comandos tiene disponible.

4.1.8. HU01.8:Comando modopciones

Como usuario del bot,
Quiero poder ajustar las opciones del bot,
Para crear una imagen como yo quiera.

Pruebas de aceptación

- Las opciones deben cambiarse y ser persistentes.
- Cada usuario tiene sus propias opciones.
- El usuario puede cambiar las opciones tantas veces como desee.

4.1.9. HU01.9:Comando start

Como usuario del bot,
Quiero tener una guia al iniciar el bot,
Para saber como puedo usar el mismo.

Pruebas de aceptación

- Este comando debe utilizarse en cuanto te suscribes al bot
- Al igual que el comando help, tiene que darte las instrucciones que se pueden utilizar con el bot.

5. Diseño estático y dinámico del software

Es interesante desde el punto de vista del análisis del software tanto el diseño estático el el dinámico ya que aportan información valiosa en diferentes etapas del proceso de desarrollo siendo, ya que, nos muestran cómo se relacionan las diferentes partes de nuestro proyecto tanto entre ellas mismas como con el usuario que las va a usar.

En esta documentación se mostrarán dos diagramas, uno de clases, para el diseño estático,y uno de secuencia de sistema, representando el diseño dinámico, ambos representantes del software creado.

Combinando los diagramas y las breves explicación de los mismos que se van a dar es posible comprender de manera simple y concisa el funcionamiento de este bot, tanto en tiempo real como a nivel lógico. Además la elaboración de estos diagramas, no solo ayuda a actores externos a comprenderlos con mayor facilidad, si no, como ya se ha explicado con anterioridad, ayuda a los propios desarrolladores a detectar fallos o mejoras en el software creado.

5.1. Diseño estático

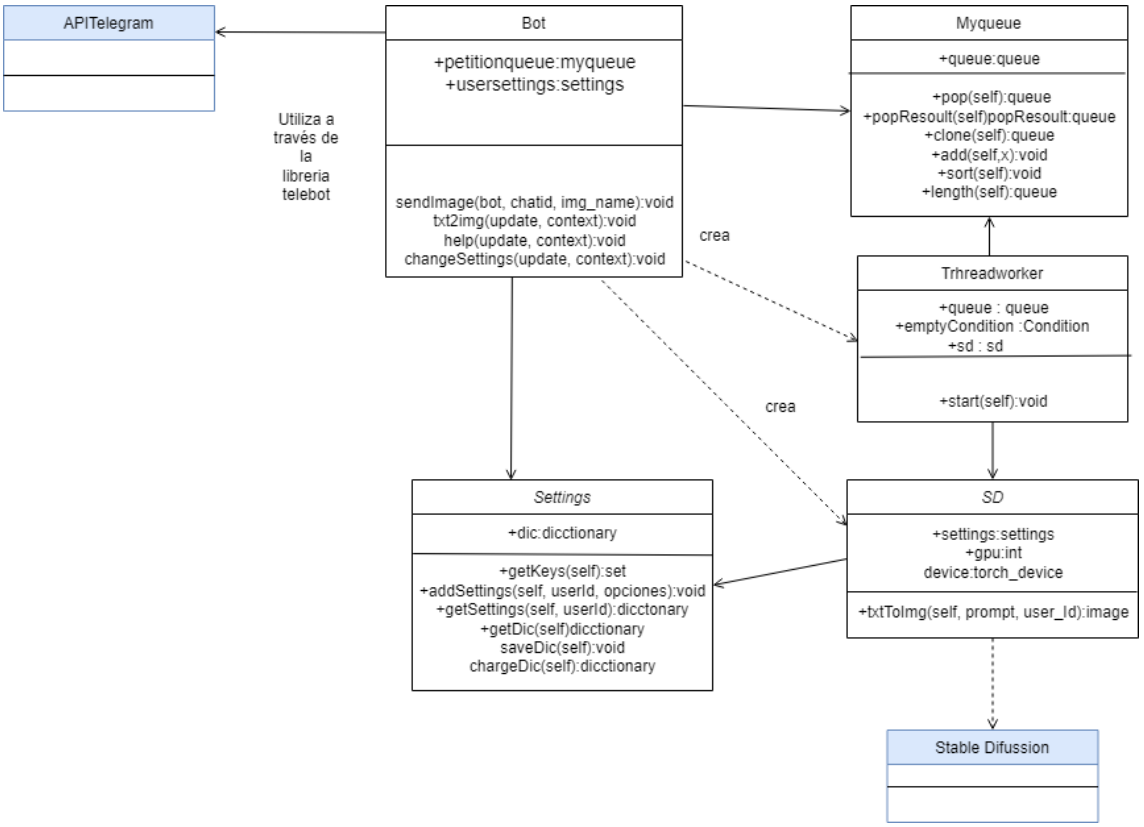


Figura 4. Diagrama de clases.

En el diagrama se puede observar las diferentes clases que conforman el software, estas clases representan los diferentes objetos y las dependencias que tienen entre sí. El programa está separado en 7 clases, cada una con sus métodos y atributos propios.

Se podría separar este diagrama en 2 tipos de clases, las “propias” y las “externas”. Las “propias” son las clases creadas para este proyecto y que contienen código propio diseñado para suplir las especificaciones del bot y de todo su funcionamiento, estas clases serían *bot, myqueue, settings, Threadworker y sd*. Por otro lado hablamos de “clases externas”, en este caso el API de Telegram y el modelo de stable diffusion, son aquellas que no pertenecen al proyecto en sí pero son necesarias para el funcionamiento del mismo, en este caso estaríamos hablando de las marcadas en el diagrama como Stable Diffusion y APITelegram.

La clase principal sería la clase *bot*, es la clase encargada de orquestar el funcionamiento del bot, se encarga del control de los comandos, del envío de mensajes y, asimismo, funciona como “main” del programa. Además de esto es la clase que se pone en contacto con la API de Telegram para hacer todas las gestiones propias de los mensajes, tanto recibir como enviar.

Por orden de importancia se podría seguir con la clase *hilo*, esta clase es la encargada de gestionar las peticiones de las imágenes, las manda crear, y avisa a la clase *bot* de cuando estas

tienen que ser enviadas. En el proyecto esta clase también es una de las que permite que el programa sea concurrente, ya que se crearía una clase hilo por cada modelo cargado en GPU, de esta manera podríamos estar creando varias imágenes a la vez.

La clase *sd* es la clase donde se alberga el modelo, esta clase, como *hilo*, es una de las clases encargadas de que el programa pueda ser concurrente, al poder cargarse un modelo en cada una de las GPUs del sistema y que funcionen de forma indistinta. Esta clase es la encargada de gestionar el modelo y cargarlo en GPU así como de la propia creación de la imagen.

La clase *settings* y la clase *cola*, podrían tener su propia clasificación como clases que forman una estructura. Empezando con la clase *settings*, es la clase en la que reside el diccionario de opciones. En ella están implementados todos los métodos necesarios para trabajar con este diccionario. La clase *cola* funciona de forma parecida, contiene la cola de peticiones y todas las opciones necesarias para usarla en el programa.

APITelegram es una clase UML que, como se mencionó anteriormente, representa la API de Telegram y como el programa se relaciona con ella. No es una clase a nivel de código fuente, es solo es una abstracción para poder indicar cómo se comporta el programa con esta herramienta. Al igual pasa con Stable Diffusion, esta "clase" representa el modelo obtenido de Stable Diffusion, todas sus dependencias y demás.

5.2. Diseño dinámico

La forma más sencilla de representar el diseño dinámico de un proyecto de software es mediante un diagrama de secuencia de sistema, en él se puede ver con claridad cómo se comporta e interactúa el programa en tiempo de ejecución.

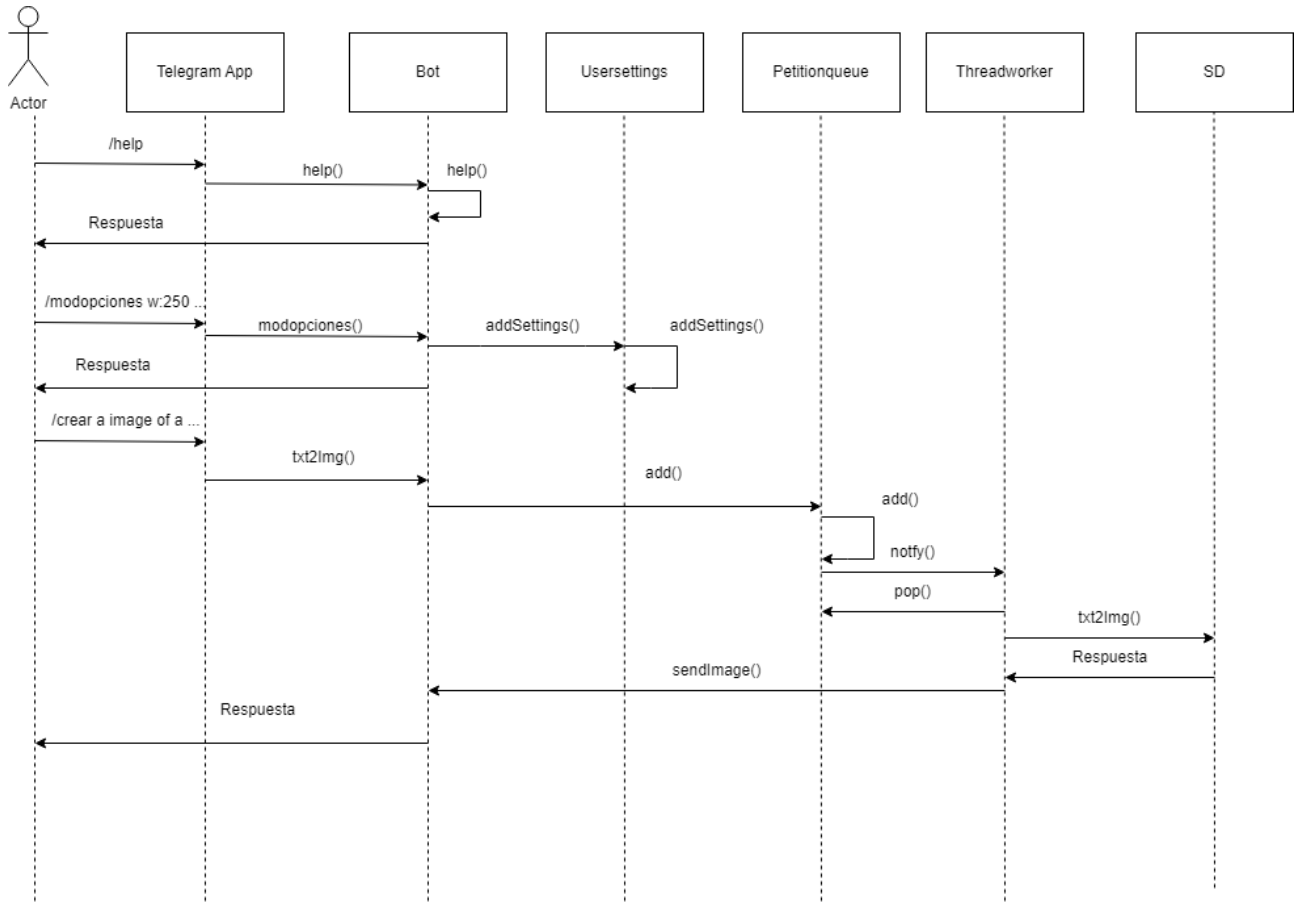


Figura 5. Diagrama de secuencia de sistema.

En el diagrama se muestran ejemplos de las principales funcionalidades del bot y cómo funcionan estas cuando el programa está en ejecución.

Antes de entrar a la explicación en profundidad de cómo se comporta con cada una de las funcionalidades, se podría decir que hay un funcionamiento similar en todas ellas, este es, el que se da al recibir el mensaje con el comando deseado. Este mensaje siempre será tratado por la aplicación de Telegram y enviado a nuestro bot para que lo gestione como se indica en el propio programa.

El comando más simple a tratar es el comando `/help`, en cuanto el bot recibe este mensaje procede a devolver un mensaje al usuario con las opciones de las que dispone el bot para que pueda hacer un funcionamiento correcto del mismo. Es un comando que solo requiere de la clase bot ya que es la encargada de enviar todos los mensajes a los usuarios.

Siguiendo está el comando `/modopciones`, este es un poco más complejo. En este caso tenemos que el bot, cuando recibe el mensaje con el comando /modopciones, accede a la clase settings, la encargada de guardar todas las opciones de cada usuario, en ella se actualizan las opciones de el usuario que lo haya indicado y es de nuevo la clase bot la que indica que el cambio se ha realizado con éxito.

Por último se encuentra la instrucción más complicada `/crear`. Esta instrucción requiere del funcionamiento de 4 clases. Cuando la instrucción `/crear` llega al bot este escribe en la cola la petición que se ordenará y avisará a la clase hilo de que está preparada para que esta la acceda y empiece el proceso de creación de la imagen. La forma en la que se ordena esta cola es mediante

prioridades, las prioridades se asignan de forma qué, si un usuario ha pedido muchas imagenes de golpe y otro usuario solicita una imagen después, se atenderá la imagen del nuevo usuario antes de que se acaben todas las que envió el primero, de esta forma se impide un secuestro del bot por parte de un único usuario. Una vez en la clase hilo esta se encarga de enviarle toda la información que la clase SD, el modelo, necesita para poder crear la imagen. Una vez la imagen es creada es la propia clase hilo la que notifica a la clase bot de nuevo para que envíe la imagen al usuario que la ha solicitado.

6. Gestión de la información

La principal información que guarda este bot se podría separar en dos bloques, por un lado hablamos de la información referente a las imágenes y por otro la información sobre las opciones de los usuarios.

6.1. Imágenes

La información referente a las imágenes que se almacena son:

- Las propias imágenes, estas son almacenadas una vez creadas.
- El nombre de la última imagen creada en un archivo, este archivo almacena el nombre de esa imagen para después leerlo y poder enviar la imagen correcta a cada usuario.
  La forma de poner los nombres a las imágenes es mediante un ID único gracias a la función uuid4() que proporciona la biblioteca estándar de python.

6.2. Opciones del usuario

Las opciones de usuario son guardadas de manera interna mediante un diccionario de diccionarios. El diccionario exterior sería el que relaciona el Id de cada usuario con sus opciones y el interior el que relaciona cada opción con su valor.

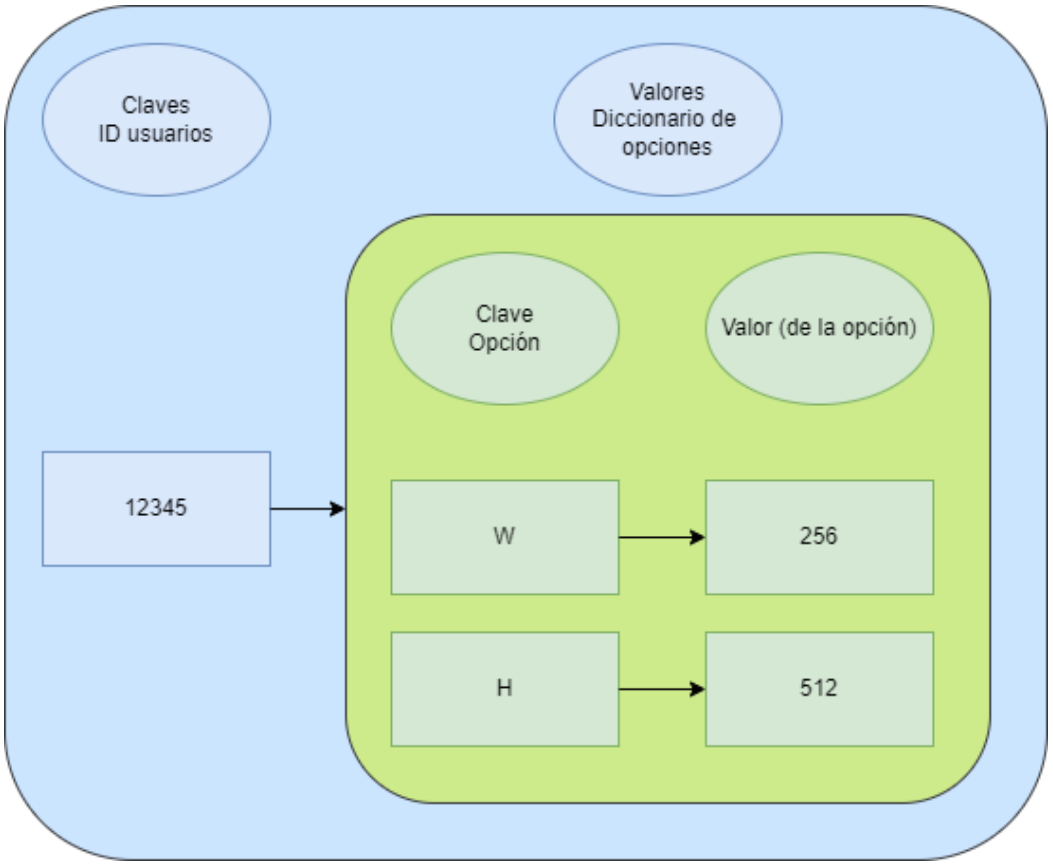


Figura 6. Esquema de la estructura para las opciones de usuario.

Para conseguir que estas opciones sean persistentes se utiliza el módulo dill de python que nos permite guardar una estructura como esta en un archivo que se puede guardar y cargar. De esta manera podremos guardar en el archivo el mapa actualizado cada vez que se realice una petición, y además si el bot es apagado que una vez que se inicie recupere las opciones que habían indicado los usuarios antes del cese de servicio.

Estas son las formas en las que el proyecto guarda y maneja la información. En el futuro se podría implementar una especie de memoria caché, la cual envía directamente una imagen ya creada y guardada si tiene los mismos datos y opciones de creación.

7. Pruebas

Durante el desarrollo de este trabajo se han realizado pruebas para verificar el correcto funcionamiento del bot. Las pruebas que se han realizado y que nos resultan más útiles para una aplicación tan cercana a los usuarios, son las llamadas pruebas de caja negra, Estas pruebas se caracterizan por no analizar el código en sí, si no por analizar las respuestas del programa a diferentes entradas, de esta forma sabremos si los requisitos funcionales se cumplen en nuestro programa o no, independientemente de cómo estén implementados.

Para realizar las pruebas se ha usado como base las historias de usuario definidas en puntos anteriores, de esta forma, las pruebas nos indican si los requisitos funcionales definidos con anterioridad se cumplen para nuestro bot. Cada historia tiene una serie de criterios de aceptación que son los que determinarán si la prueba se ha pasado con éxito.

El conjunto de pruebas se fue realizando a medida que se añadían las funcionalidades al código. Si en el código implementado había presencia de errores estos, en la medida de lo posible se corrigen antes de pasar al siguiente punto, de esta forma, sabremos en qué parte del código puede encontrarse el error, si todas las pruebas del código implementado con anterioridad son correctas, hay muchas posibilidades de que el error lo esté generando el nuevo código implementado y no el anterior.

HU01.1 Creación del chatBot

Prueba	**Funcionalidad**	**Evaluación**
Se realiza la búsqueda del bot en Telegram.	Cuando se busca el bot este existe y puede ser usado.	La prueba se ha llevado a cabo con éxito.
Se enciende el bot.	El bot responde.	La prueba se ha llevado a cabo con éxito.

HU01.2 Validación de la entrada del usuario.

Prueba	**Funcionalidad**	**Evaluación**
Se le ingresa un prompt erróneo o inexistente al bot.	El bot informa de que el prompt no es correcto.	La prueba se ha llevado a cabo con éxito.

HU01.3 Concurrencia de peticiones.

Prueba	**Funcionalidad**	**Evaluación**
Se le envía una petición al bot.	El bot responde a esa petición en ese momento.	La prueba se ha llevado a cabo con éxito.
Se le envían muchas peticiones simultáneas al bot.	El bot devuelve todas las peticiones enviadas.	La prueba se ha llevado a cabo con éxito.

HU01.4 Ordenación de peticiones por prioridad.

Prueba	Funcionalidad	Evaluación
Se le envían muchas peticiones simultáneas al bot.	El bot devuelve todas las peticiones enviadas en orden.	La prueba se ha llevado a cabo con éxito.
Se le envían muchas peticiones simultáneas al bot de diferentes usuarios.	El bot devuelve todas las peticiones enviadas, en orden y priorizando a los usuarios que llegan con nuevas peticiones frente a usuarios que envían muchas.	La prueba se ha llevado a cabo con éxito.

HU01.5 Integración de la IA.

Prueba	Funcionalidad	Evaluación
Se deja el bot funcionando durante un largo periodo de tiempo y se le envía una petición.	El bot responde con normalidad a las peticiones.	La prueba se ha llevado a cabo con éxito.

HU01.6 Comando crear.

Prueba	Funcionalidad	Evaluación
Se le envían mensajes con el comando crear desde varios chats.	el bot responde de forma correcta enviando la imagen al chat correspondiente.	La prueba se ha llevado a cabo con éxito.
Se le envían prompts con muy bajo nivel de detalle y complejidad incluso una palabra.	Las imágenes que genera no siempre guardan una relación estrecha con el prompt.	La prueba se ha llevado a cabo con poco éxito.
Se le envían prompts con un nivel de detalle alto o medio.	Las imágenes generadas guardan una estrecha relación en todos los casos.	La prueba se ha llevado a cabo con éxito.

HU01.7 Comando help.

Prueba	Funcionalidad	Evaluación
Se le envía al bot un mensaje con el comando help	El bot responde a esa petición con la información sobre el bot.	La prueba se ha llevado a cabo con éxito.

HU01.8 Comando modopciones.

Prueba	Funcionalidad	Evaluación
Se le envía una petición al bot de cambiar las opciones.	Las opciones son almacenadas.	La prueba se ha llevado a cabo con éxito.
Se cambian opciones de varios usuarios.	Cada usuario tiene sus propias opciones.	La prueba se ha llevado a cabo con éxito.
Se apaga el bot y se vuelve a encender.	Las opciones se han quedado guardadas.	La prueba se ha llevado a cabo con éxito.

HU01.9 Concurrencia de peticiones.

Prueba	Funcionalidad	Evaluación
Al iniciar el bot se escribe el comando /start.	El bot se inicia y envía un mensaje de bienvenida.	La prueba se ha llevado a cabo con éxito.

Como se puede ver Las pruebas realizadas han salido exitosas, es obvio que no han salido bien la primera vez que se han realizado pero, mediante realizarlas y resolver los problemas cuando el resultado no era el deseado, se puede afirmar que el programa cumple con los requisitos funcionales especificados en las historias de usuario del proyecto.

Después de la realización de las mismas se puede concluir que las pruebas son un factor clave en el desarrollo del software. Sin una buena organización de las pruebas podemos encontrarnos errores que son muy complicados de solucionar por el gran número de factores que los puede causar, sin embargo si realizamos pruebas de forma iterativa es mucho más sencillo saber dónde se puede encontrar el error y ser mucho más eficiente en la resolución del mismo.

8. Manual de usuario

El manual de usuario permite tener una guía de cómo hacer funcionar el bot, tanto como usuario de telegram como desde el punto de vista de un desarrollador.

8.1. Telegram

El funcionamiento de la aplicación para un usuario es ciertamente sencillo, podríamos dividirlo en diferentes secciones haciendo alusión a los comandos usados para controlar el bot.

8.1.1. Requisitos

Para poder tener acceso al bot los requisitos son simples, en este caso simplemente necesitaremos tener un cuenta de Telegram y conexión a internet, con estos dos requisitos tendremos todo lo necesario para poder usar todas las funcionalidades del bot.

8.1.2. Suscripción al bot

Para suscribirte al bot solo es necesario buscar el nombre del mismo en Telegram y empezar una conversación con él, una vez que estamos dentro de la conversación para arrancar el bot y ponerlo en funcionamiento debemos usar el comando /start.

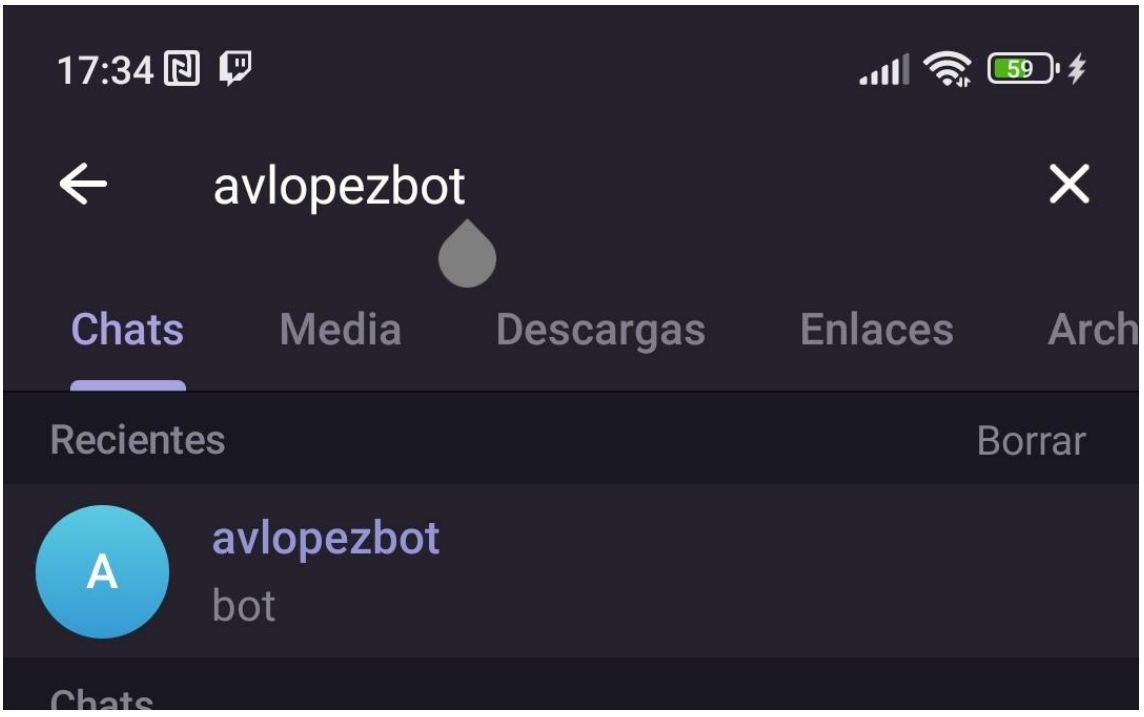


Figura 7. Vista de la suscripción al bot.

8.1.3. Start

Este comando es el de activación del bot, solo debería ser usado cuando se inicia el bot y lleva implícito el comando help, de forma que cuando el bot se inicia envía un mensaje con las instrucciones para poder usar el bot.

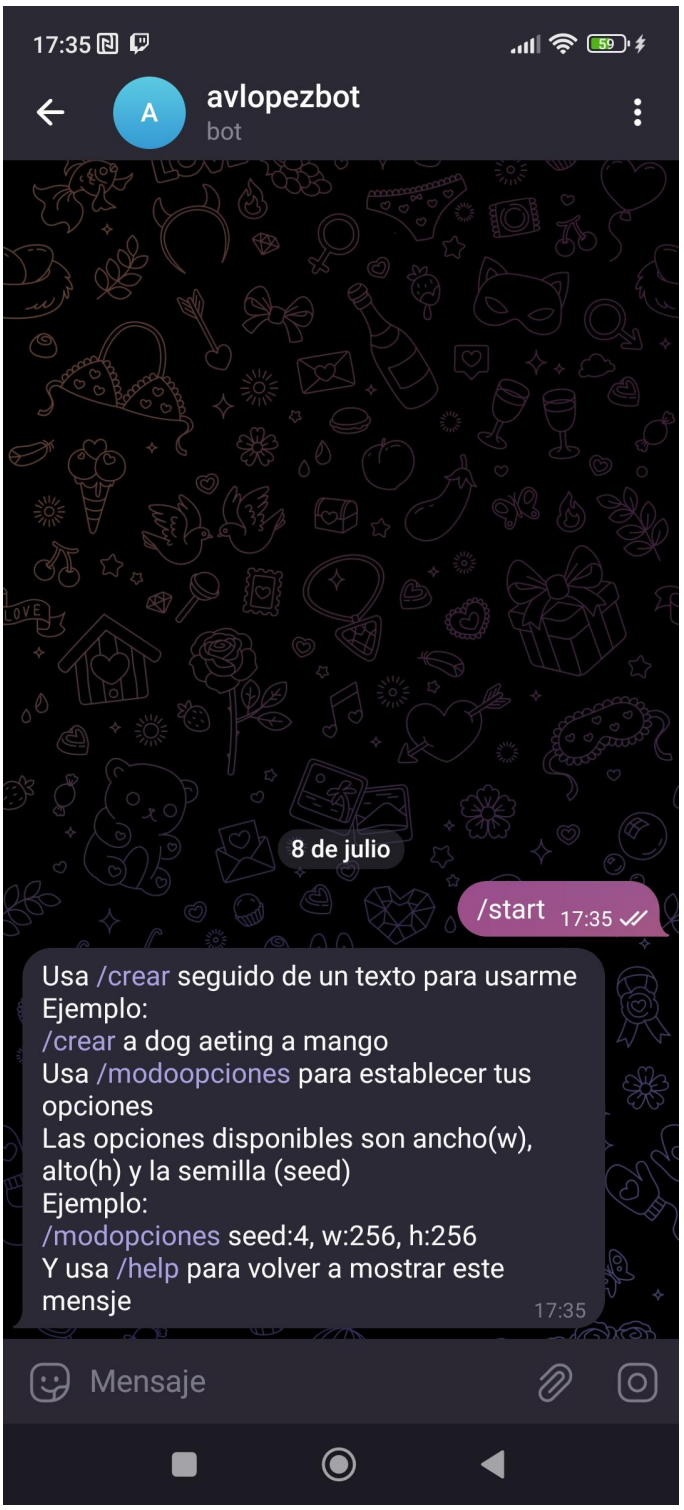


Figura 8. Vista del método start.

Pese a que este comando puede utilizarse en cualquier momento de forma análoga al comando help,aún que no está diseñado para eso y no se indica en el comando help, está solo pensado para el inicio del bot, de forma que el usuario no se sienta perdido al empezar a utilizarlo.

8.1.4. Crear

Este es el uso principal del bot y el que tiene más importancia dentro del proyecto, se trata del comando mediante el cual el usuario debe indicar que imagen quiere crear a partir de un texto que él mismo ingrese.

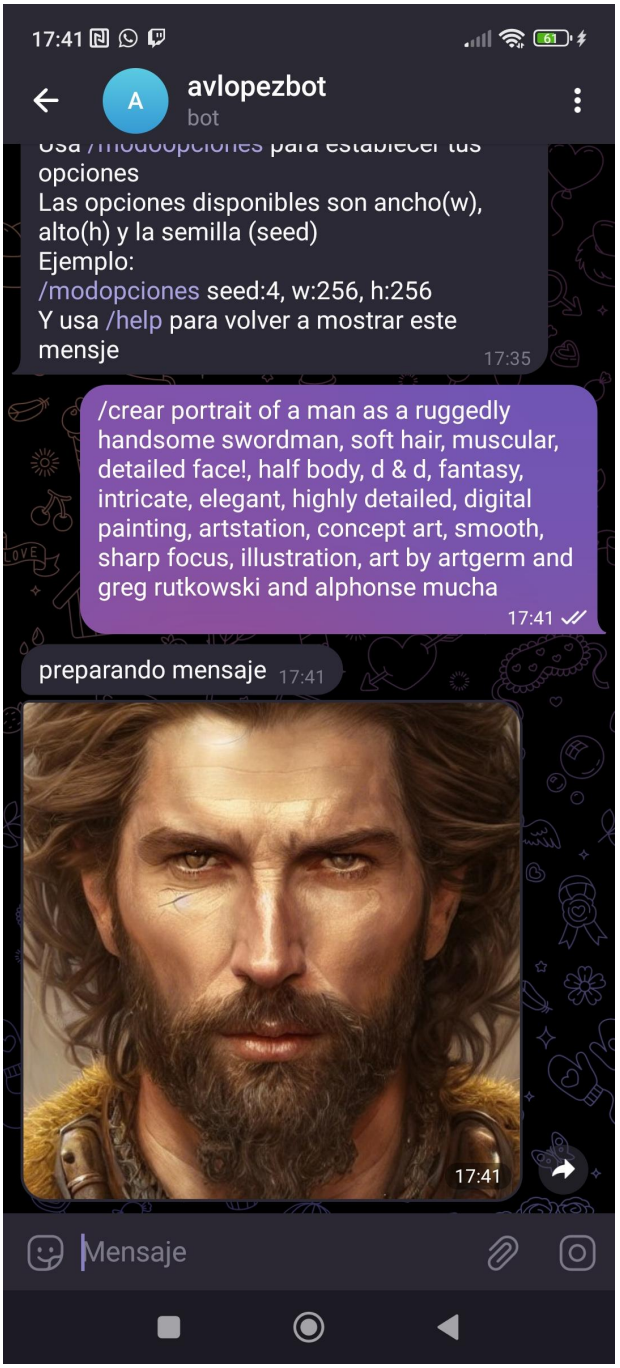


Figura 9. Vista del método crear.

Como se puede ver en la imagen con el comando /crear “prompt” el bot devuelve una imagen basada en el prompt ingresado.

8.1.5. Help

Con este comando el usuario puede ver los comandos disponibles por el bot y cómo usarlos. El comando puede ser invocado en cualquier momento y es una “guía” para el funcionamiento del bot

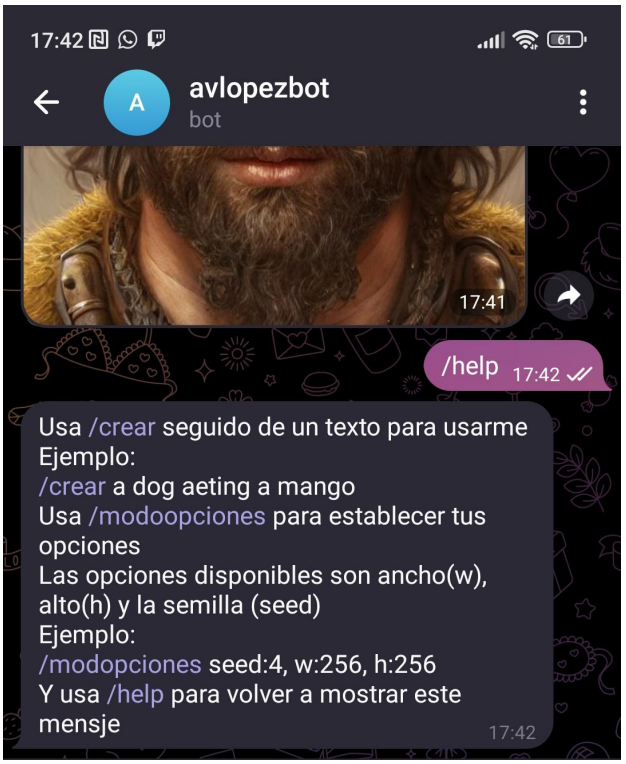
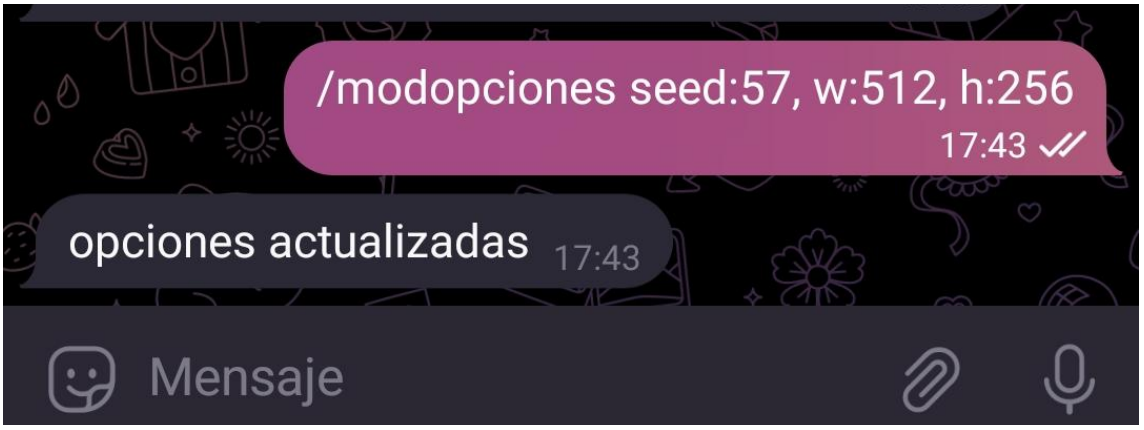


Figura 10. Vista del método help.

En la imagen se muestra el uso de este comando usando /help y como se muestran todos los comandos disponibles.

8.1.6. Modopciones

Este comando junto a /crear son los más importantes a la hora del uso del bot, con este comando podremos modificar el tamaño de la imagen, tanto ancho como bajo, así como la semilla de cada imagen, pudiendo generar la misma o una diferente con el mismo prompt usando este comando.



8.1.6.0.1 Figura 11. Vista del método modopciones 1.

Como se puede ver en la imagen este comando permite cambiar de cualquiera de las 3 opciones, de forma individual o conjunta, es decir puedes cambiar uno, dos o los tres valores disponibles a la vez.

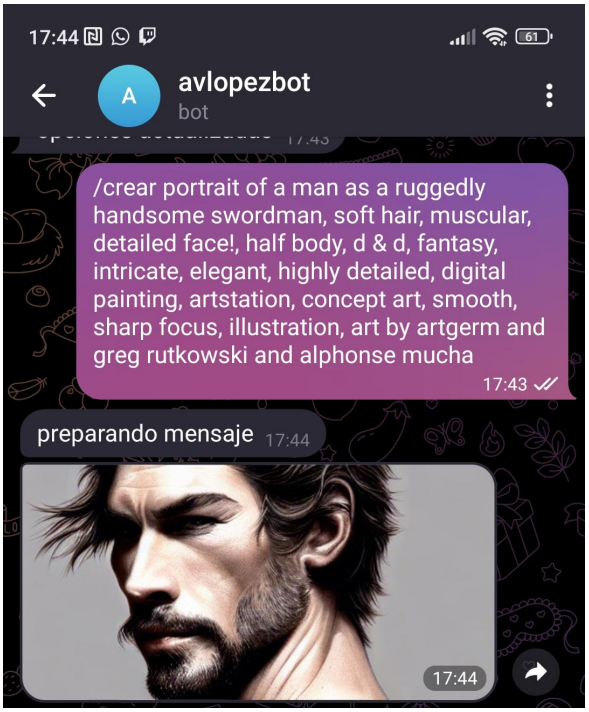


Figura 12. Vista del método modopciones2 .

En esta imagen se puede ver cómo afectan los cambios de las opciones a la creación de una imagen, en este caso se ha hecho la imagen más ancha que alta y este es el resultado.

8.2. Bot

El funcionamiento del bot es ciertamente más complejo y requiere de un conocimiento relativamente avanzado sobre cómo funciona la API de telegram así como python y el propio modelo de Stable Diffusion, para saber como funciona todo, y en caso de querer, poder hacer modificaciones sobre el mismo.

8.2.1. Requisitos

Para que todo funcione de forma correcta es necesario tener una GPU con la tecnología CUDA y con al menos 10GB de VRAM para poder garantizar un funcionamiento correcto de la tecnología empleada. Además será necesario tener python3 instalado, todos los imports de librerías (dill, conda, pyTorch, Omega Conf, numpy) necesarios instalados y la el paquete telebot que es el nos dejara comunicarnos con la propia API de Telegram

8.2.2. Poner el bot en funcionamiento

En caso de querer utilizar el programa con tu propio bot o usarlo de base para crear otro tipo de bot o programa será necesario que crees un bot propio desde telegram, con la ayuda de BotFather este proceso es muy sencillo, y introduzcas el token del bot creado en el código. De esta forma tu bot y tu programa estarán comunicados.

Para poner en marcha el bot solo será necesario ejecutar el archivo bot.py, con el comando python3 bot.py sería suficiente, de esta manera el bot comenzará a funcionar y lo tendremos activo.

Una vez que tengamos el bot en marcha no tendremos que hacer nada más, el bot estará operativo y listo para cumplir con las peticiones que le sean enviadas. Si quisiéramos apagar el bot o reiniciarlo sería tan fácil como interrumpir su ejecución y volver a ponerlo en marcha en cualquier momento.

9. Trabajo a futuro y conclusiones

9.1. Trabajo a futuro

En el futuro se podrían expandir y actualizar las opciones y características de dicho bot mejorando tanto el funcionamiento del mismo como la experiencia de usuario.

En el futuro se podrían aplicar más y diferentes modelos al bot y de esta forma poder ampliar las funcionalidades del mismo haciéndolo más útil y completo.

En este momento el bot usa la versión 2.5 de Stable Diffusion, en el futuro se actualizará el CheckPoint de este modelo, manteniendo así el bot actualizado y siguiendo de cerca las actualizaciones en el campo de IA.

Se podría aplicar el modelo de este bot para otras aplicaciones como Whatsapp en la que se están viendo recientemente este tipo de bots. Aún que funcionen de forma diferente, parte del código y de la arquitectura se podría reutilizar así como el mismo modelo sería reutilizable.

9.2. Conclusiones

En general creo que realizar este trabajo ha aportado conocimientos nuevos y mejorado los ya obtenidos con anterioridad. Para su desarrollo se ha tenido que investigar en campos nuevos, IA, modelos, nuevas tecnologías, como funciona una API y cómo trabajar con ella de manera eficiente, programar en un lenguaje nuevo, python, y se han ampliado mucho los conocimientos adquiridos con anterioridad, trabajar en un software que tenga muy en cuenta el hardware disponible, el desarrollo ágil llevado a la práctica, el desarrollo de una nueva aplicación empezando de 0. Además de todos estos conocimientos personales, también es importante considerar la importancia del trabajo tutelado en un proyecto como este, la figura del tutor es muy importante y puede beneficiar mucho si se realiza de forma correcta, como en este caso.

La creación de este proyecto deja ver los conocimientos adquiridos en esta carrera, no solo en el apartado más técnico como podrían ser conocimientos sobre un lenguaje de programación o el uso de una herramienta en específico, también en cómo llegar a ciertas soluciones, usar los foros que genera la comunidad para la solución de ciertos problemas.

Desde el punto de vista del alumno, es muy gratificante crear un proyecto que tenga una aportación a la vida cotidiana de su entorno y dentro de una aplicación ya existente, que sea palpable y algo interactivo, esto ha conseguido hacer que sienta un gran interés sobre el tema de las IAs, el tratamiento de datos, así como de desarrollar nuevos proyectos y descubrir nuevas formas trabajar.

Como conclusión final, el desarrollo de este proyecto ha generado un impacto positivo en el alumno reafirmando su vocación por la materia, por las nuevas tecnologías y por la constante mejora que debe realizar en su futuro. Es un ejercicio de disciplina y de realidad ver cómo es posible realizar un proyecto así si se le dedica el suficiente esfuerzo y ganas.

9.3. Aportaciones

Este proyecto trata de aportar, a través del uso de las IAs una herramienta más para mejorar el uso de estas en conversaciones por chat, así como acercar esta tecnología a las personas que no tienen conocimientos sobre el tema. Las aportaciones principales se podrían resumir en:

- Acercamiento de Ia IA a cualquier usuario de forma simple.
- Mejora del sistema de chat.
- Adaptable a grupos, el bot puede ser añadido al grupo como participante y cualquier otro participante puede interactuar con él.
- El modelo tiene un filtro que impide la creación de imágenes que puedan contener contenidos inapropiados, de esta forma el bot puede ser usado por cualquier persona de cualquier edad.

Además de estas aportaciones principales se podría decir que este es solo un ejemplo de todas las opciones que nos aporta el uso de estas IAs a nuestra vida cotidiana sin necesidad de usarlas de una forma completamente técnica, sino a nivel de usuario casual.

10. Referencias

[1] Castiglioni, I., Rundo, L., Codari, M., Di Leo, G., Salvatore, C., Interlenghi, M., Gallivanone, F., Cozzi, A., D'Amico, N. C., & Sardanelli, F. (2021). AI applications to medical images: From machine learning to deep learning. Physica Medica, 83, 9-24. https://doi.org/10.1016/j.ejmp.2021.02.006

[2] Enjellina, N., Beyan, E. V. P., & Rossy, N. A. G. C. (2023). Review of AI Image Generator: Influences, Challenges, and Future Prospects for Architectural Field. jarina, 2(1), 53-65. https://doi.org/10.24002/jarina.v2i1.6662

[3] Guzman, A. L., & Lewis, S. C. (2019). Artificial intelligence and communication: A Human–Machine Communication research agenda. New Media & Society, 22(1), 70-86. https://doi.org/10.1177/1461444819858691

[4] Telegram. (2021, 17 marzo). PDF Bot - Bot for telegram. Bot for telegram. https://botfortelegram.com/es/bots-espanol/bots-de-utilidades/pdf-bot/

[5] Lee, S. (2023, 4 mayo). Diffusion Explainer: Visual Explanation for Text-to-image Stable Diffusion. arXiv.org. https://arxiv.org/abs/2305.03509

[6] Telegram – a new era of messaging. Telegram. https://telegram.org/

[7] Telegram APIs. https://core.telegram.org/

[8] Schwaber, K. (1997). SCRUM Development Process. En Springer eBooks (pp. 117-134). https://doi.org/10.1007/978-1-4471-0947-1_11

[9] Hron, M. (2018). Scrum in Practice: an Overview of Scrum Adaptations. https://scholarspace.manoa.hawaii.edu/items/e4482132-3501-456d-814f-88e6e2747305

[10] colaboradores de Wikipedia. (2023). Diagrama de Gantt. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Diagrama_de_Gantt

[11] Python Release Python 3.11.0. Python.org. https://www.python.org/downloads/release/python-3110/

[12] PyTorch.. https://pytorch.org/

[13] CompVis. GitHub - CompVis/stable-diffusion: A latent text-to-image diffusion model. GitHub. https://github.com/CompVis/stable-diffusion

[14] Managing environments — conda 23.5.1.dev61 documentation. . https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

[15] Surameery, N. M. S., & Shakor, M. Y. (2023). Use Chat GPT to Solve Programming Bugs. UJIT, 31, 17-22. https://doi.org/10.55529/ijitc.31.17.22

[16] draw.io - free flowchart maker and diagrams online. https://app.diagrams.net/

[17] Tucnak. . GitHub - tucnak/telebot: Telebot is a Telegram bot framework in Go. GitHub. https://github.com/tucnak/telebot

[18] Suaza, K. V. (2015). Mejora de historias de usuario y casos de prueba de metodologías ágiles con base en TDD. https://ojs.tdea.edu.co/index.php/cuadernoactiva/article/view/246

[19] Manavski, S. A., & Valle, G. (2008). CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment. BMC Bioinformatics, 9(S2). https://doi.org/10.1186/1471-2105-9-s2-s10

[20] Ardimansyah, M. I., & Widianto, M. H. (2021). Development of online learning media based on Telegram Chatbot (Case studies: Programming courses). Journal of physics, 1987(1), 012006. https://doi.org/10.1088/1742-6596/1987/1/012006

[21] python-telegram-bot v20.4. . https://docs.python-telegram-bot.org/en/stable/

[22] BuilesYeison.. GitHub - BuilesYeison/Telegram-bot-for-admin-groups: GitHub. https://github.com/BuilesYeison/Telegram-bot-for-admin-groups

[23] Building Telegram Bots.. Google Books. https://books.google.es/books?hl=es&lr=&id=xrd9DwAAQBAJ&oi=fnd&pg=PR3&dq=telegram+bots&ots=x2rTKXRP2j&sig=p7xYjGn33_anbDW72miKgtqHNr0#v=onepage&q=telegram%20bots&f=false