Institut Puig Castellar
Santa Coloma de Gramenet

SYNCRONIBOT

(Projecte de desenvolupament)
CFGS Administració de Sistemes Informàtics i Xarxes

Sistemes Microinformatics y xarxes
Autors: Iker Fernandez Rocamora, Alejandro Medrano Martinez
Grup: 2c
Curs Acadèmic: sistemas Microinformatics y xarxes (smx)

Resumen del proyecto

El proyecto consiste en desarrollar un bot multifunción para Discord llamado SyncronyBot, cuyo objetivo es facilitar la gestión y organización de la información y comunicación dentro de servidores de Discord.

Este bot busca resolver la dificultad que tienen los usuarios para seguir grandes volúmenes de mensajes y actualizaciones, evitando que se pierda información importante o que se tarde mucho en leerla. Para ello, integra inteligencia artificial que resume conversaciones relevantes, permite búsquedas y da respuestas automáticas, aplica moderación inteligente para detectar spam y palabras prohibidas, y automatiza la publicación de contenido externo como posts de Instagram. También incluye funcionalidades como encuestas y un sistema de traducción para mejorar la interacción comunitaria.

La metodología se basa en el uso de lenguajes de programación como Python y JavaScript, junto a frameworks especializados como Discord.js y Botpress para implementar la inteligencia artificial. Se emplean bases de datos ligeras para almacenar configuraciones y se despliega en plataformas cloud como Heroku o Railway, garantizando estabilidad y disponibilidad continua. Las conclusiones apuntan a un bot funcional que centraliza múltiples utilidades en una sola herramienta, mejorando la experiencia de los usuarios de Discord al ahorrar tiempo y facilitar la organización sin necesidad de múltiples bots.

Palabras clave

Código	Palabras Clave
RF2	Discord.js
RF3	Python
RF4	Inteligencia Artificial
RF5	Automatización de tareas
RF6	Procesamiento de lenguaje natural
RF7	Webhooks
RF8	Base de datos SQL

Abstract (English)

The project involves developing a multifunctional Discord bot called SyncronyBot, designed to streamline information management and communication within Discord servers.

This bot aims to address the challenge users face in keeping up with large volumes of messages and updates, preventing them from missing important information or taking too long to read it. To achieve this, it integrates artificial intelligence that summarizes relevant conversations, enables searches and provides automatic responses, applies intelligent moderation to detect spam and prohibited words, and automates the posting of external content such as Instagram posts. It also includes features like polls and a translation system to enhance community interaction.

The methodology is based on the use of programming languages such as Python and JavaScript, along with specialized frameworks like Discord.js and Botpress to implement the artificial intelligence. Lightweight databases are used to store configurations, and the bot is deployed on cloud platforms like Heroku or Railway, ensuring stability and continuous availability. The findings point to a functional bot that centralizes multiple utilities in a single tool, improving the Discord user experience by saving time and facilitating organization without the need for multiple bots.

Keywords

Code	Keywords
RF1	Discord API
RF2	Discord.js
RF3	Python
RF4	Artificial Intelligence
RF5	Task Automation
RF6	Natural Language Processing
RF7	Webhooks
RF8	SQL DataBase

Índex

Llista de figuras

1 Introduccion

1.1 Contexto

El aumento del uso de plataformas de comunicación como Discord ha provocado la creación de muchas comunidades digitales donde cada vez se envían más mensajes y existe una mayor actividad. Esto ha hecho que la gestión de servidores y la moderación de contenidos se conviertan en tareas más difíciles para administradores y moderadores.

Actualmente, muchos servidores utilizan varios bots diferentes para realizar tareas como moderación, tickets, traducciones o automatización de mensajes. Sin embargo, utilizar tantas herramientas distintas puede resultar poco práctico y desorganizado, especialmente en comunidades grandes o empresas que usan Discord como sistema de soporte técnico.

Además, controlar conversaciones, evitar spam y organizar la información requiere tiempo y una buena gestión. Por este motivo, surge la necesidad de crear soluciones más completas que reúnan varias funciones en un solo bot y faciliten la administración del servidor.

SyncronyBot nace con el objetivo de ofrecer una herramienta más cómoda y eficiente, capaz de automatizar tareas, mejorar la organización y facilitar la experiencia de los usuarios dentro de Discord.

1.2 Justificación

El proyecto es importante porque busca solucionar algunos de los problemas más comunes en la gestión de servidores de Discord. Actualmente, muchas comunidades utilizan varios bots diferentes para realizar distintas tareas, lo que puede hacer que la administración sea más complicada y menos organizada. SyncronyBot propone reunir varias funciones útiles en una sola herramienta para facilitar el trabajo de los administradores y mejorar la experiencia de los usuarios.

Además, el uso de inteligencia artificial permite automatizar tareas que normalmente requieren mucho tiempo, como resumir conversaciones largas o detectar mensajes inapropiados y spam. Esto ayuda a reducir la carga de trabajo de los moderadores y permite que los servidores funcionen de manera más rápida, cómoda y organizada. También responde a la necesidad actual de utilizar herramientas más inteligentes y automáticas dentro de comunidades digitales, donde cada vez existe más actividad y una mayor cantidad de información que gestionar.

1.3 Objetivos

Este proyecto tiene como finalidad principal desarrollar un bot multifuncional para discord que mejore la gestión y organización de la información y la comunicación dentro de servidores profesionales o no.

1.3.1 Objetivo general

Desarrollar SynchronyBot, un bot para Discord con funcionalidades avanzadas basadas en inteligencia artificial y automatización, capaz de gestionar conversaciones, moderar contenido, automatizar procesos de investigación, gestión entre otros dentro del propio servidor.

1.3.2 Objetivos específicos

Objetivos	
A1	Implementar un sistema automático de resumen de conversación relevantes mediante procesamiento de lenguaje natural
A2	Desarrollar funcionalidades de moderación inteligente para detectar y gestionar spam, contenidos inapropiados o palabras no deseadas
A3	Automatizar la sincronización y publicación de contenidos externos, como publicaciones de instagram, en un canal específico
A4	Crear herramientas para mejorar la interacción comunitaria incluyendo traducciones automáticas.
A5	Establecer una estructura robusta basada en datos ligeros, plataformas cloud o que local pero 24/7 y sistemas de IA que faciliten las funciones

1.4 Estratègia i planificació del projecte

La estrategia elegida es desarrollar un producto nuevo desde cero, aprovechando frameworks que existen como Discord.js y Botpress. Esta aproximación permite un diseño flexible y adaptado a las necesidades específicas detectadas, garantizando un mayor control sobre la comunicación entre los distintos componentes del sistema y una facilidad para ampliarlo en el futuro según las necesidades del proyecto. Al partir de una base completamente nueva, se puede estructurar la arquitectura de forma más limpia, organizando cada módulo desde el inicio y evitando dependencias innecesarias que podrían limitar el crecimiento del sistema.

Esta decisión también permite optimizar el rendimiento y la integración entre el bot, la inteligencia artificial, la base de datos y el panel web, ya que todos los componentes se diseñan específicamente para trabajar juntos desde el principio. De esta manera, se asegura una mayor coherencia en el flujo de datos y una mejor eficiencia en la ejecución de las distintas funcionalidades.

Se había valorado y descartado la adaptación de bots existentes por limitaciones en personalización, escalabilidad y control del sistema. Muchos bots ya desarrollados ofrecen funcionalidades predefinidas que no siempre se ajustan a los requisitos concretos del proyecto, además de dificultar la integración de sistemas avanzados como inteligencia artificial personalizada o paneles de administración propios. Por este motivo, se consideró más adecuado desarrollar una solución propia que permitiera una evolución constante y totalmente adaptada a los objetivos de SyncronyBot.

1.5 Metodología de trabajo

Se opta por un análisis ágil basado en Scrum, lo que facilita iteraciones rápidas, adaptación continua y una comunicación fluida entre los desarrolladores durante todo el desarrollo del proyecto. Este enfoque permite dividir el trabajo en ciclos cortos denominados sprints, en los que se planifican, desarrollan y revisan las diferentes funcionalidades del sistema de forma progresiva. De este modo, se mejora la organización del trabajo y se reduce el riesgo de errores acumulados al final del desarrollo.

Gracias a esta metodología, es posible ajustar prioridades de manera constante según el avance del proyecto y el feedback obtenido en cada iteración. Esto resulta especialmente útil en proyectos como SyncronyBot, donde los requisitos pueden evolucionar o ampliarse a medida que se prueban nuevas funcionalidades o se detectan mejoras posibles.

Herramientas como Trello y GitHub se utilizan para la gestión de tareas y el control de versiones. Trello permite organizar el trabajo mediante tableros visuales, listas y tarjetas, facilitando la asignación y seguimiento de tareas entre los miembros del equipo. Por otro lado, GitHub se emplea para el control de versiones del código, lo que permite registrar todos los cambios realizados, trabajar de forma colaborativa y mantener un historial completo del desarrollo del sistema.

Este enfoque permite mantener un nivel de entrega constante y organizado, asegurando que cada avance pueda ser probado, validado e integrado correctamente antes de pasar a la siguiente fase. Además, mejora la calidad del producto final al fomentar la revisión continua y la mejora incremental del sistema.

1.6 Estudio económico y presupuestario

Se realizará un inventario de recursos necesarios, incluyendo licencias de software, servidores cloud y horas de desarrollo. Este análisis permite identificar todos los elementos imprescindibles para la correcta implementación del sistema, así como estimar los recursos técnicos y humanos necesarios para su puesta en marcha. De esta forma, se obtiene una visión global de las necesidades del proyecto y de los posibles escenarios de inversión.

A partir de este inventario, se calculará un coste orientativo que incluye las fases de desarrollo, despliegue y mantenimiento a medio plazo. En este cálculo se tienen en cuenta factores como el tiempo de programación, la infraestructura necesaria para mantener el bot activo las 24 horas, el almacenamiento de datos en la base de datos y el posible uso de servicios externos para inteligencia artificial o alojamiento en la nube. Este enfoque permite aproximar el coste real del proyecto en un entorno profesional.

Sin embargo, también es importante destacar que el proyecto puede implementarse de forma prácticamente gratuita si se utilizan recursos de código abierto y servicios con planes gratuitos, como hosting básico o bases de datos en versiones free tier. Esto reduce significativamente la inversión inicial, aunque puede implicar ciertas limitaciones en rendimiento o escalabilidad.

Por otro lado, se considera el posible retorno económico del proyecto, con el objetivo de que los interesados puedan evaluar si su desarrollo resulta rentable o viable a largo plazo. Este retorno puede provenir de suscripciones a funcionalidades premium, donaciones o modelos de monetización dentro de comunidades grandes de Discord. En conjunto, este análisis permite equilibrar la inversión necesaria con los beneficios potenciales, facilitando la toma de decisiones sobre la viabilidad del proyecto.

2 Descripción del proyecto

2.1 Análisis de requisitos SYNCRONYBOT

Los requisitos del proyecto se han de establecer partiendo de las necesidades de comunidades digitales y entornos profesionales que utilitzen Discord como plataforma principal de la comunicación

2.1.1 Requisitos funcionales

El bot ha de poder resumir automàticament converses utilitzant tècniques de processament de llenguatge natural.

RF1	Debe incorporar un sistema de moderación inteligente capaz de identificar y gestionar mensajes con contenido inadecuado, spam o palabras prohibidas.
RF2	Debe permitir sincronizar y publicar contenidos externos, como publicaciones de Instagram o avisos de otras plataformas.
RF3	Incluye una herramienta de traducción automática que permite traducir mensajes en tiempo real entre idiomas.
RF4	El bot gestionará estadísticas de uso y registros de actividad para facilitar el seguimiento de los eventos dentro del servidor.

2.1.2 Requisitos no funcionales

RNF1	Debe funcionar de manera eficiente incluso en servidores con cientos de usuarios simultáneos.
RNF2	El servicio se alojará en un entorno cloud para garantizar un funcionamiento 24/7.
RNF3	La arquitectura modular debe permitir añadir nuevas funcionalidades sin afectar a las ya existentes.
RNF4	Los datos manejados por el bot (mensajes, configuraciones, etc.) deben tratarse de manera confidencial y siguiendo buenas prácticas de seguridad.
RNF5	La interfaz de administración debe ser intuitiva y accesible para usuarios sin formación técnica.
RNF6	El código debe estar documentado y versionado mediante GitHub para facilitar la colaboración y las actualizaciones.

2.1 Previsión de tareas de investigación

El proyecto utilizará herramientas actuales que permitan construir un sistema estable, fácil de ampliar y con buen rendimiento.

Herramienta	Explicación
Node.js Discord.js	Servirán para crear el bot y gestionar todas las acciones que se realicen dentro de Discord, como mensajes, comandos o reacciones de los usuarios.
Python	Se utilizará para desarrollar las partes de inteligencia artificial, como los resúmenes automáticos de conversaciones. Se emplearán librerías como spaCy o transformers, que ayudan a comprender y resumir texto.
Base de datos SQL	Permitirá guardar configuraciones, preferencias de los usuarios y registros de actividad.
Express.js	Estas herramientas ayudarán a crear la parte web del proyecto y a conectar el bot con otros servicios externos.
Docker	Facilitará el despliegue del bot en diferentes entornos, evitando problemas de compatibilidad y haciendo más sencillo el mantenimiento.
Plataforma en la nube	El servicio se alojará en la nube para garantizar que el bot esté activo las 24 horas y pueda responder rápidamente a los usuarios.

2.2 Tecnologías

2.2.1 Comparativa de les tecnologies valorades

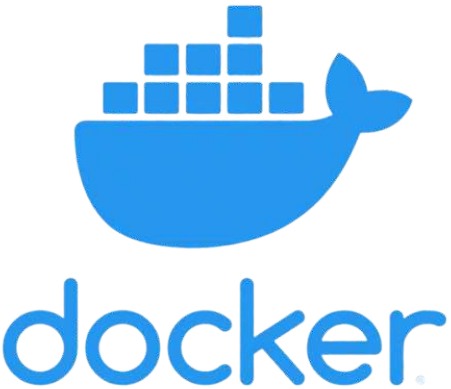
Se han valorado varias herramientas para hacer el bot y la parte web. Estas son las más importantes:

Tecnologias		Descripcion
T1	Discord.js	Es la mejor opción para crear el bot porque es la más usada, tiene mucha ayuda y funciona muy bien con Discord.
T2	Discord.py ()	También sirve para crear el bot, pero es menos recomendable si luego quieres hacer una web conectada, porque funciona mejor con Node
T3	Botpress	Sirve para hacer un asistente conservacional fácil y rápido, pero no es tan buena opción si quieres algo muy personalizado o avanzado.
T4	Express.js Node.js	Es buena para crear el panel web de administración porque es flexible y se integra bien con el bot, aunque hay que programar todo desde cero.

T5	Python fast API	Es ideal para la parte de IA porque es rápido, para usarlo junto con un bot en donde puede complicar el proyecto.
T6	MySQL	Es una base de datos buena para guardar configuraciones y datos del bot porque es flexible y escalable.
T7	SQLite	Es una base de datos simple, útil para pruebas, pero no es buena para un proyecto grande
T8	Docker	Sirve pra que el bot funcione igual en cualquier ordenador o servidor, y facilitar el despliegue
T9	Cloud (descartada)	Se usa para alojar el bot en internet y que este activo 24/7, aunque puede tener un coste

2.2.2 Tecnologías escogidas

Tecnología	Uso en el proyecto
Discord.js	Se utilizará para crear el bot, ya que es la tecnología más popular y estable.
Python	Se utilizará para la inteligencia artificial, como los resúmenes automáticos, porque Python dispone de muy buenas librerías para el tratamiento de texto.
Express.js	Se utilizará para crear el panel web de administración, ya que puede conectarse fácilmente con el bot.
Docker	Se utilizará para empaquetar el proyecto y asegurar que funcione igual en cualquier servidor, evitando futuros problemas de compatibilidad.
Plataforma en la nube	Se utilizará para alojar el bot y mantenerlo activo 24/7, con buena estabilidad y rendimiento.



2.3 Estructura del proyecto

SynchronyBot es un bot para Discord con funciones avanzadas de inteligencia artificial y automatización. El proyecto está dividido en diferentes partes que trabajen juntos pero de forma independiente. Esto hace que sea más fácil crear, arreglar y ampliar.

Componentes Principals	Descripció
Bot de discord (Discord.js)	Es lo que está dentro del servidor que puede hablar con los usuarios, recibir mensajes,comandas y relaciones
Servicio de IA (Phyton)	Es lo que hace las funciones de inteligencia artificial, como por ejemplo hacer resumenes, esta separado del bot y se comunicara con él a través de internet (API)
Base de datos	Es donde se guarda la información del bot como por ejemplo los registros de los usuarios. Tambien sirve para recuperar informacion para cuadno sea necesario
Desplegamiento (Docker + cloud)	El projecto se puede empaquetar en un docker para que funcione en cualquier sitio, se aloja en una plataforma en el nuvol (Render o Railway) para que siempre este activo.

2.4 Descripción de los componentes

En este apartado se describen los diferentes componentes que forman parte del proyecto SyncronyBot, así como su función y responsabilidad dentro del sistema. Cada componente está diseñado para cumplir una tarea específica, pero todos trabajan conjuntamente para ofrecer una solución completa y funcional, garantizando así un rendimiento estable, organizado y fácilmente gestionable. Gracias a esta distribución de responsabilidades, el sistema puede operar de manera eficiente incluso cuando aumenta la cantidad de usuarios o la complejidad de las tareas que debe realizar.

La arquitectura del proyecto se basa en una estructura modular, que permite separar las diferentes partes del sistema (bot, inteligencia artificial, base de datos y panel web). Esta separación facilita el desarrollo, el mantenimiento y la escalabilidad del proyecto, ya que cada componente puede modificarse o ampliarse sin afectar al resto. Además, este enfoque modular reduce significativamente la posibilidad de errores globales, ya que cada módulo puede probarse de forma independiente antes de integrarse con el sistema completo.

Este tipo de arquitectura también mejora la organización del código y permite que distintos desarrolladores puedan trabajar simultáneamente en diferentes partes del sistema sin generar conflictos. Por ejemplo, mientras un componente se encarga de la lógica del bot en Discord, otro puede centrarse en la gestión de la base de datos o en el desarrollo del panel web. De esta manera, el flujo de trabajo es más eficiente y ordenado.

2.4.1 Componentes- 1

Que es? (Discord.js)

El bot de Discord es el componente principal del sistema y actúa como punto de interacción directa con los usuarios dentro del servidor. Está desarrollado utilizando la librería Discord.js, que permite conectarse a la API de Discord y gestionar todos los eventos que se producen dentro del servidor, como mensajes, reacciones, entradas y salidas de usuarios, o comandos personalizados. Gracias a esta conexión directa con la API, el bot puede responder de forma automática y en tiempo real a las acciones de los usuarios, ofreciendo una experiencia dinámica y fluida dentro de la comunidad.

Este componente funciona de manera continua en un servidor en la nube, manteniéndose activo las 24 horas del día para poder responder en tiempo real a cualquier acción de los usuarios, sin interrupciones ni dependencia del dispositivo local del desarrollador. Para ello, se suele desplegar en servicios de hosting o servidores VPS que garantizan estabilidad, conexión permanente y buen rendimiento incluso con un número elevado de usuarios interactuando simultáneamente.

Además, está diseñado de forma modular para facilitar la incorporación de nuevas funcionalidades en el futuro. Esto significa que sus distintas características (como comandos, sistemas de moderación, respuestas automáticas o integración con la base de datos) se organizan en archivos o módulos separados, lo que permite mantener un código más limpio, ordenado y fácil de mantener. Esta estructura también facilita que diferentes partes del bot puedan actualizarse o mejorarse sin afectar al funcionamiento general del sistema.

Por otro lado, el bot no solo se limita a ejecutar comandos, sino que también puede reaccionar a eventos del servidor, como la creación de canales, cambios de roles o actividad de los usuarios, lo que permite automatizar muchas tareas de administración. Esto reduce la carga de trabajo de los moderadores y mejora la gestión general del servidor, haciendo que la comunidad sea más organizada y eficiente.

Que accion tiene?

Funcion	Descripcion
Recepción de mensajes	Detecta y procesa mensajes de usuarios para identificar acciones.
Ejecución de acciones	Realiza tareas automáticas como crear tickets, responder o gestionar canales.
IA	Envía datos a una IA para analizarlos y devolver resultados como resúmenes o traducciones.
Moderación	Guarda y consulta información como usuarios, configuraciones y estadísticas. Aut

2.4.2 Componentes - 2

Que es? (Servei d'IA (Python))

El servicio de inteligencia artificial es el componente encargado de procesar la información y aplicar técnicas avanzadas de tratamiento de datos y lenguaje natural. Está desarrollado en Python, ya que este lenguaje dispone de numerosas librerías especializadas en inteligencia artificial, aprendizaje automático y procesamiento de texto, lo que facilita la implementación de modelos capaces de interpretar y generar respuestas de forma eficiente y flexible.

Este servicio funciona de manera independiente del bot de Discord, comunicándose con él mediante peticiones a través de una API. Esta comunicación suele realizarse mediante solicitudes HTTP o sistemas similares, donde el bot envía la información que necesita ser procesada y el servicio de IA devuelve una respuesta ya analizada. Este enfoque permite separar completamente la lógica de la inteligencia artificial del funcionamiento principal del bot, lo que mejora significativamente la organización del sistema.

Gracias a esta separación, se consigue una arquitectura más limpia y escalable, ya que cada componente puede desarrollarse, actualizarse o sustituirse de forma independiente sin afectar al resto del sistema. Por ejemplo, es posible mejorar el modelo de inteligencia artificial o cambiarlo por otro más avanzado sin necesidad de modificar el código del bot de Discord, siempre que se mantenga la misma interfaz de comunicación.

Además, este diseño facilita el mantenimiento del proyecto, ya que los errores pueden aislarse más fácilmente al estar cada funcionalidad en un servicio distinto. También permite distribuir la carga de trabajo entre diferentes servidores o procesos, mejorando el rendimiento general del sistema cuando aumenta el número de usuarios o peticiones.

En conjunto, el servicio de inteligencia artificial no solo añade capacidades avanzadas al bot, sino que también contribuye a una arquitectura más profesional, modular y preparada para futuras ampliaciones dentro del proyecto SyncronyBot.

Que hace?

Funcion	Explicación
Resumen automático	Analiza los mensajes y genera resumenes con los puntos relevantes
Traducción de Mensajes	Permite traducir textos entre diferentes idiomas
Moderación automática	Detecta contenido inapropiado,spam o lenguaje ofensivo
Procesamiento	Interpreta el significado de los mensajes para generar la respuesta

2.4.N Componentes 3

Que es (MySQL)

MySQL es un sistema de gestión de bases de datos relacionales que permite almacenar, organizar y administrar información de forma estructurada y segura. Una base de datos se encarga de guardar los datos que utiliza una aplicación o sistema, como usuarios, productos, pedidos, registros o configuraciones, facilitando su acceso y actualización cuando sea necesario.

La información se organiza en tablas compuestas por filas y columnas, lo que permite relacionar datos entre sí y mantener un mejor control de la información. MySQL utiliza el lenguaje SQL (Structured Query Language) para realizar consultas, insertar nuevos datos, modificar registros o eliminar información.

Este sistema es ampliamente utilizado en aplicaciones web, plataformas digitales y entornos empresariales debido a su rendimiento, estabilidad y capacidad para gestionar grandes volúmenes de datos de manera eficiente.

Que guarda?

Nuestra idea principal es lograr que el bot utiliza la misma base de datos que nuestra página web, lo cual eso nos otorgaría la capacidad de tener la información de usuarios de discord directamente clasificada.

Que guarda	Explicación
Configuración del servidor	Paràmetros personalizados como canales configurados, permisos o activación de funcionalidades
Preferencias de usuarios	Datos específicos de los usuarios, como idiomas preferidos u operaciones personalizados
Registros y estadísticas	Historial de actividad del bot incidencias de moderación, uso de comandos y otros datos útiles para el análisis
Datos personales u operativos	Información necesaria para el funcionamiento interno del sistema como sesiones o procesos que estén en curso. Estos se podrán registrar en la cmd o visualizador de procesos del bot los cuales también se guardaran en unos logs.

2.5 Definición de las tareas

En este apartado se describen las diferentes pruebas y funcionalidades que se llevarán a cabo para validar el funcionamiento del proyecto. Cada tarea permite comprobar si los componentes del sistema funcionan correctamente y si cumplen los objetivos planteados inicialmente.

2.5.1 Prueba 1

En esta prueba se comprobará si el sistema puede generar resúmenes correctos de conversaciones de Discord. Esta funcionalidad es importante porque permite condensar mensajes largos en un texto más corto y fácil de consultar.

Para realizar la comprobación, se utilizará una conversación real o de prueba y el bot la enviará al servicio de inteligencia artificial. Este servicio analizará el contenido y generará un resumen automático, que después se mostrará en el canal o se guardará en la base de datos.

Para valorar el funcionamiento, se tendrá en cuenta si el resumen es claro, si mantiene la información importante y si se genera en un tiempo razonable. La conclusión esperada es que el sistema pueda ofrecer resúmenes útiles y comprensibles sin perder la idea principal de la conversación.

2.5.2 Prueba 2

En esta segunda prueba se verificará si el sistema es capaz de detectar y gestionar mensajes con spam o contenido inapropiado. El objetivo principal es asegurar que el bot mantenga un entorno limpio y seguro dentro del servidor de Discord, eliminando automáticamente los mensajes problemáticos.

Para realizar la prueba, se generarán mensajes de prueba que incluyan spam o palabras prohibidas. El bot enviará estos mensajes al servicio de inteligencia artificial para que los analice. Una vez detectado el contenido inadecuado, el bot procederá a eliminar los mensajes o avisará al usuario, guardando un registro del incidente en la base de datos para su seguimiento futuro.

Se medirá el porcentaje de mensajes detectados correctamente, la presencia de falsos positivos (mensajes normales considerados incorrectos) y el tiempo de respuesta del sistema. La conclusión esperada es que el sistema detecte la mayoría de los mensajes problemáticos con precisión y responda de manera efectiva y rápida.

2.5.N Prueba 3

En esta prueba se comprobará si el sistema puede traducir mensajes correctamente entre diferentes idiomas. El objetivo es verificar que la funcionalidad permita una comunicación fluida en servidores multilingües, facilitando la comprensión entre usuarios de distintas lenguas y mejorando la interacción entre miembros de comunidades internacionales. De esta forma, se pretende evaluar si el sistema es capaz de eliminar barreras lingüísticas dentro de Discord y favorecer una comunicación más inclusiva y accesible.

Para llevar a cabo la prueba, un usuario solicitará la traducción de un mensaje mediante un comando específico dentro del servidor. Este comando será detectado por el bot, que enviará el texto al servicio de inteligencia artificial desarrollado en Python. Este módulo se encargará de procesar la traducción utilizando modelos o APIs de procesamiento de lenguaje natural, interpretando correctamente el contexto del mensaje para ofrecer una traducción lo más precisa posible. Una vez completado el proceso, la respuesta será devuelta al bot, que la publicará directamente en el canal correspondiente para que todos los usuarios puedan visualizarla de manera inmediata.

Durante este proceso, se tendrá en cuenta la comunicación entre los distintos componentes del sistema, asegurando que el envío y recepción de datos entre el bot y el servicio de inteligencia artificial sea estable y eficiente. Además, se comprobará que no existan retrasos significativos que afecten a la experiencia del usuario, ya que este tipo de funcionalidad está pensada para usarse en tiempo real.

Se medirá la precisión de la traducción, evaluando si el significado del mensaje original se mantiene correctamente tras el procesamiento, así como el tiempo de respuesta del sistema desde que se envía el comando hasta que se muestra la traducción final. También se tendrán en cuenta posibles errores de interpretación o fallos en la detección del idioma.

La conclusión esperada es que las traducciones sean claras, coherentes y suficientemente precisas para mantener el sentido original del mensaje, además de generarse con una velocidad adecuada para su uso en tiempo real dentro del servidor.

2.5 Definición de las funcionalidades

En este apartado se detallan las funcionalidades principales de Syncrony Bot, explicando de forma conceptual cómo funcionan y qué proceso siguen dentro del sistema. El objetivo es ofrecer una visión clara y estructurada de las capacidades del proyecto, permitiendo comprender cómo interactúan sus diferentes módulos para dar respuesta a las necesidades del usuario. Cada funcionalidad específica los componentes implicados, su flujo de funcionamiento y su estado de implementación, lo que permite diferenciar entre aquellas ya desarrolladas y las que se encuentran en fase de mejora o futura incorporación.

Además, este análisis funcional no solo se centra en el comportamiento individual de cada característica, sino también en la forma en la que estas se integran dentro del sistema global. SyncronyBot está diseñado para que todas sus funcionalidades trabajen de manera conjunta y coordinada, compartiendo información a través de la base de datos y el bot principal, lo que permite una experiencia coherente y centralizada dentro del servidor de Discord.

Asimismo, se tiene en cuenta la eficiencia del sistema y la experiencia del usuario, asegurando que cada funcionalidad responda de manera rápida y estable. Esto es especialmente importante en funciones en tiempo real, donde la velocidad de respuesta y la precisión del sistema son factores clave. De esta manera, se garantiza que SyncronyBot no solo sea funcional, sino también escalable, mantenible y preparado para futuras mejoras o ampliaciones.

2.5.1 Funcionalidad 1

Esta funcionalidad permite generar resúmenes de conversaciones cuando un usuario lo solicita mediante un comando o cuando el sistema detecta una conversación larga y relevante. El objetivo es condensar la información esencial para facilitar su consulta rápida.

El proceso funciona de la siguiente manera: primero, el bot recopila los mensajes relevantes del canal. Después, los envía al servicio de inteligencia artificial desarrollado en Python, que analiza el contenido y crea un resumen con los puntos principales. Finalmente, el bot publica el resumen en el canal o lo guarda en la base de datos para su uso futuro.

Componentes implicados:
- Bot de Discord (Discord.js)
- Servicio de IA (Python)
- Base de datos

Estado: Implementada totalmente.

2.5.2 Funcionalidad 2

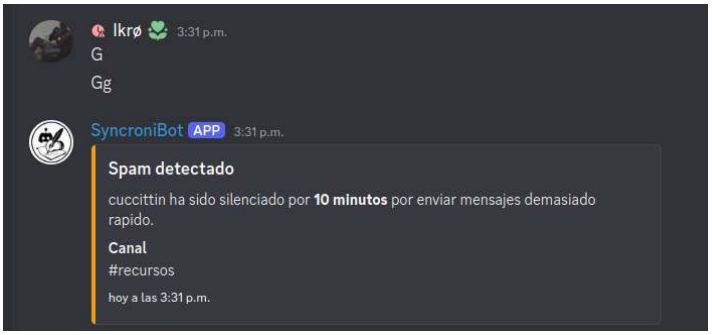
Esta funcionalidad permite al bot detectar mensajes con spam, insultos o palabras prohibidas, actuando automáticamente para mantener el servidor limpio, ordenado y seguro. El objetivo principal es reducir la carga de trabajo de los moderadores y garantizar un entorno saludable dentro de la comunidad, evitando comportamientos tóxicos o que puedan afectar negativamente a la experiencia de los usuarios.

El proceso se ejecuta de forma continua y en tiempo real: cuando un usuario envía un mensaje en el servidor, el bot lo intercepta y lo analiza inmediatamente utilizando los sistemas de detección configurados. En una primera fase, se realiza un filtrado básico de palabras clave o patrones comunes de spam. Posteriormente, si es necesario, la inteligencia artificial entra en acción para clasificar el contenido con mayor precisión, determinando si se trata de spam, lenguaje ofensivo o una infracción más grave de las normas del servidor. Este enfoque híbrido permite combinar rapidez con precisión en la detección.

Una vez identificado el contenido como inapropiado, el bot ejecuta una serie de acciones automáticas según la gravedad del caso. Estas pueden incluir la eliminación del mensaje, la advertencia al usuario, la asignación de un aviso interno o incluso sanciones más estrictas si el comportamiento se repite. Además, todos los incidentes quedan registrados en la base de datos para mantener un historial de moderación que pueda ser consultado posteriormente por los administradores o moderadores del servidor.

También se genera una notificación automática que informa sobre el evento, lo que permite un seguimiento en tiempo real de la actividad de moderación. Esto ayuda a mantener un control transparente y facilita la supervisión del sistema.

Componentes implicados:
- Bot de Discord (Discord.js)
- Servicio de IA (Python)
- Base de datos del bot



Estado: *Implementada parcialmente, dependiendo de la configuración y del modelo de inteligencia artificial utilizado. Actualmente cubre funciones básicas de detección, pero puede ampliarse para incluir más tipos de infracciones, reglas personalizadas por servidor, sistemas de puntuación de reputación o moderación predictiva basada en comportamiento del usuario.*

2.5.N Funcionalidad 3

La integración de inteligencia artificial dentro del bot se ha diseñado para ampliar sus capacidades de comunicación y automatización, permitiendo no solo generar resúmenes de conversaciones, sino también traducir mensajes automáticamente mediante comandos específicos. El objetivo principal de esta funcionalidad es mejorar la interacción entre usuarios de diferentes idiomas y facilitar la comprensión del contenido dentro del servidor, especialmente en comunidades internacionales o con miembros que hablan distintas lenguas. Gracias a la IA, el bot puede interpretar el contexto de los mensajes y devolver traducciones naturales, rápidas y comprensibles directamente desde Discord, sin necesidad de utilizar herramientas externas.

El funcionamiento del sistema se basa en la conexión entre el bot desarrollado con Discord.js y la API de Gemini de Google. Cuando un usuario ejecuta el comando de traducción, el bot obtiene el contenido del mensaje seleccionado o el texto introducido manualmente, y lo envía a la inteligencia artificial junto con instrucciones específicas sobre el idioma de destino y el formato de respuesta. La IA procesa el contenido utilizando modelos avanzados de lenguaje natural y devuelve una traducción contextualizada, evitando traducciones literales incorrectas y mejorando la calidad del resultado final. Todo este proceso ocurre en pocos segundos y de manera completamente transparente para el usuario.

Además de las traducciones, la integración con IA también permite generar resúmenes automáticos de conversaciones recientes dentro de un canal. Para ello, el bot recopila los últimos mensajes enviados, filtra mensajes irrelevantes o provenientes de otros bots y construye un contexto que posteriormente es enviado al modelo de inteligencia artificial. El sistema devuelve una síntesis clara y organizada de los temas tratados, ayudando a los usuarios a ponerse al día rápidamente sin tener que leer conversaciones extensas. Esto resulta especialmente útil en canales muy activos o en comunidades con una gran cantidad de actividad diaria.

Toda la lógica de comunicación con la inteligencia artificial se gestiona desde el propio bot, utilizando solicitudes asíncronas hacia la API externa y manejando automáticamente posibles errores o fallos de conexión. También se implementó un sistema de fallback entre distintos modelos de Gemini para garantizar la continuidad del servicio en caso de que un modelo no esté disponible temporalmente. Gracias a este enfoque, el bot puede adaptarse dinámicamente y seguir funcionando sin interrupciones, ofreciendo una experiencia mucho más estable y profesional para los usuarios del servidor.

Componentes implicados:

- Bot de Discord (Discord.js)
- API de Gemini AI (Google Generative AI)
- Sistema de comandos slash
- Gestión de variables de entorno (.env)

Además, esta integración permite añadir funciones avanzadas de IA como detección automática de idioma, moderación inteligente, respuestas automáticas y análisis de actividad del servidor.

3 Arquitectura i componentes

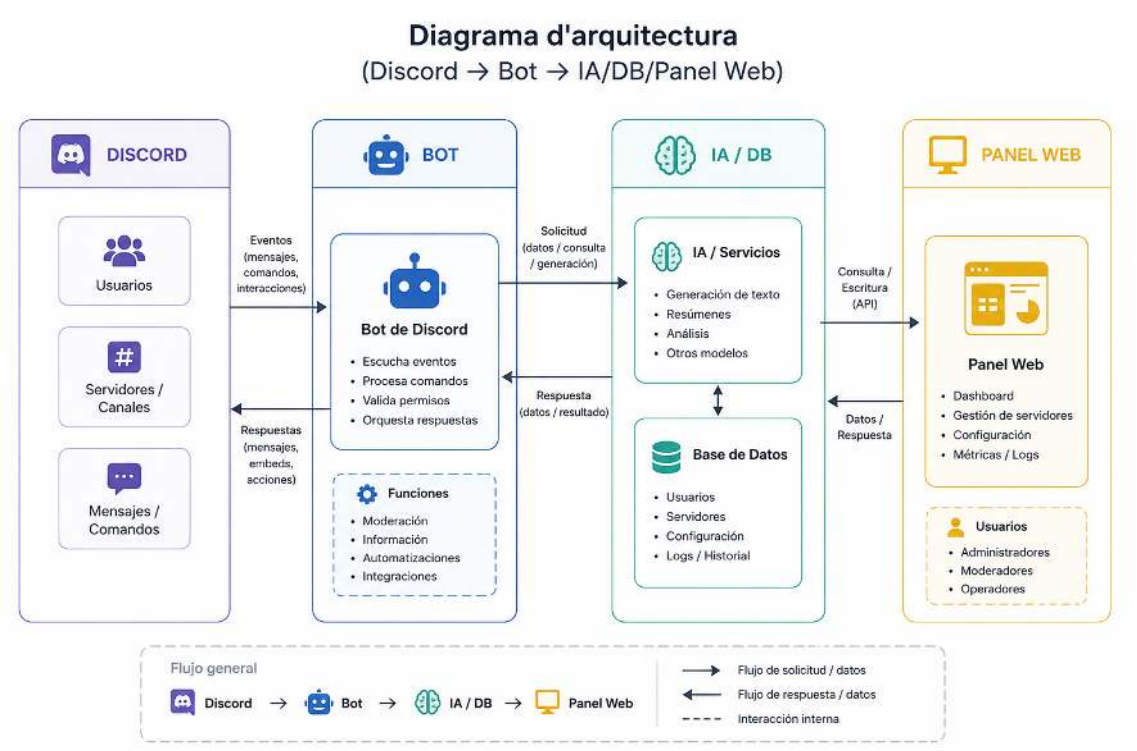
El proyecto Synchrony Bot es un bot de Discord que está dividido en varias partes que trabajan conjuntamente.

La idea general es sencilla los usuarios escriben mensajes en Discord y el bot los analiza para ayudarles, ya sea resumiendo, respondiendo o moderando.

El funcionamiento sería el siguiente:

- El usuario escribe en Discord
- El bot recibe el mensaje
- Lo procesa (con IA o funciones internas)
- Devuelve una respuesta o realiza una acción

Todo esto funciona gracias a un servidor con una función implementada llamada pm2 lo cual hace que cuando ejecutamos el bot “npm run start” se auto inicia, esta se queda atencia y si nuestro bot se cae automáticamente reinicia el bot para que vuelva a estar activo.



Descripció dels components:

Components	Descripció
Bot Principal	Es la parte mas importante del sistema. Se encarga de leer los mensajes y decidir qué hacer en cada caso. Esta desarollado con herramientas como Discord.js o Botpress
Inteligencia artificial	Permite al bot resumir conversaciones largas y responder preguntas automáticamente, ayudando a los usuarios a no tener que leer todo el chat
Moderacion	Sirve para mantener el orden en el servidor. Detecta spam, bloquea palabras prohibidas y puede avisar o sancionar a usuarios.
Búsqueda de mensajes	Permite encontrar información rápidamente dentro del chat sin tener que revisar todo.
Publicaciones automáticas	El bot puede publicar contenido automáticamente desde redes sociales como instagram.
Encuestas y traduccion	Permite crear encuestas dentro del servidor y traducir mensajes para que todos los usuarios lo entiendan
Base de datos	Es donde se guarda la información, como la configuración del servidor y las referencias de los usuarios.

Definición de las funcionalidades

SyncronyBot es tiene muchísimas funciones gracias al implemento de una API-Key de la IA. Este es capaz de:

- Gestionar rangos,permisos y usuarios
- Resolver preguntas y buscar información
- Generar tickets
- Auto Moderar según las reglas del servidor
- Banear usuarios, mutear usuarios
- Poner musica
- Buscar en wikipedia / fandom específicamente a gran velocidad
- Resumir los últimos mensajes registrados en el chat

Todas estas funcionalidades ayudan a los usuarios a ahorrar tiempo y a no perder información importante.

Otros capítulos

(También se tiene en cuenta la seguridad, controlando quién puede usar cada función, que el bot funcione en muchos servidores y que sea rápido y estable.)

Conclusión general

Tras haber trabajado durante todo este tiempo en el desarrollo del bot, podemos concluir que el proyecto ha supuesto un gran aprendizaje tanto a nivel técnico como organizativo. La funcionalidad principal del proyecto ha sido la implementación de una inteligencia artificial integrada en un bot de Discord, algo poco común, ya que normalmente los bots disponibles suelen centrarse en una única función específica.

En nuestro caso, hemos logrado desarrollar un sistema con múltiples funcionalidades útiles, pulidas y funcionales, orientadas a facilitar la gestión de comunidades dentro de Discord. Durante el desarrollo hemos aprendido cómo funciona la creación de un bot, el uso de dependencias, la programación de comandos y la conexión entre procesos y bases de datos. Además, uno de los aspectos más importantes ha sido comprender cómo integrar diferentes herramientas dentro de un mismo proyecto, combinando las funcionalidades tradicionales de un bot con el uso de inteligencia artificial.

El objetivo ha sido demostrar que es posible centralizar distintas herramientas en un único sistema organizado y eficiente, manteniendo funcionalidades similares a las de otros bots pero añadiendo características más avanzadas, como la IA integrada.

Aunque nos hubiera gustado desarrollar también un panel web para la personalización y gestión del bot, los costes de hosting y configuración suponían una inversión elevada. Aun así, conseguimos crear un sistema alternativo para desarrolladores mediante terminal, permitiendo realizar configuraciones similares a las que ofrecería un panel web.

Finalmente, estamos orgullosos del trabajo realizado y consideramos que se trata de un proyecto original con potencial de uso en un entorno profesional en el futuro. Aunque algunas funciones podrían seguir mejorándose, estamos satisfechos con el resultado final obtenido.

Cumplimiento de objetivos

Hemos cumplido el objetivo principal del proyecto quizás no tan pulido, pero mucho más que funcional. Es capaz de hacer todo lo prometido al principio del proyecto y hemos trabajado bien para que todo lo necesario para que el bot fuera capaz de realizar todas las tareas necesarias. La integración de la IA fue un éxito y salió mucho mejor de lo esperado pese a la batalla continua con los errores que presentaba a la hora de integrarse. Estuvo interesante descubrir que a la hora de desarrollarlo podríamos haber implementado que por ejemplo hubieran comandos como /ai generate ya que podíamos especificar en cada comando el API-Key que queríamos usar. Así que podríamos haber implementado diferentes tipos de IA más que una general. Podríamos poner Generativa y de investigación.

A recalcar para futuros proyectos que tras una ardua investigación no existe ninguna forma gratuita de obtener una API-Key lo cual significa que si un proyecto enfocado a utilizar una IA hay que poner presupuesto.

Metodología

Aquí es donde el proyecto a cojear un poco ya que la organización para este proyecto era muy complicada, al tener que alojar el Bot en el mismo servidor que la página Web por temas de rendimiento y agilización de trabajo ya que si lo hacíamos en dos máquinas distintas tendríamos que haber hecho una arquitectura cliente-servidor y no disponíamos del tiempo para programar eso ya que el bot requiere de mínimo 2 meses de programación para poder tener margen de errores. Nos centralizamos tanto en la documentación y en la página web que cuando llegó la hora de hacer el bot la página web no estaba realmente terminada y aún tenía errores con la base de datos. Por ende empezamos a hacer un deslice de tareas. Un día se trabaja el bot, el otro la página web, turnabamos entre horas para ir trabajando los dos desarrolladores. Después uno se dedica a avanzar la documentación mientras que otro hacia la programación. Pero finalmente tras poner orden nos aclaramos y pusimos como prioridad el bot ya que al final era la base de todo. Era la base de nuestro proyecto así que procedimos a trabajar en el y una vez ya casi finalizado el bot empezamos a pulir la página web.

La comunicación en este proyecto fue clave y sobre todo la crítica mutua y la autocrítica, cada cierto tiempo uno se metía a revisar el proceso del otro desarrollador para ver qué tan avanzado estaba, que se podía mejorar y qué cosas podrían trabajar. En nuestro caso cuando uno revisaba el trabajo del otro si teníamos algo que mejorar lo comentamos y lo aplicamos, los dos desarrolladores teníamos el mismo voto de igualdad nadie daba más orden o menos, todo se comentaba y se le daba voto a la opinión de ambos a la hora de trabajar pese que en algún momento uno daba orden a ciertas cosas por planificación.

Aprendizaje

Este proyecto nos ha hecho darnos cuenta de que es desarrollar algo propio, hemos tenido que aprender a agilizarnos lo máximo posible, aprender que debemos tener un margen de error definido y aproximado. La importancia de los meses de investigación y desarrollo para la formación del bot ha sido más importante de lo que parece pese a que se ha usado IA para el desarrollo de la programación, se debía de entender el código fuente con un base básica para entender que estamos desarrollando y saber encontrar y solucionar los errores. En nuestro caso fuimos programando y parcheando nuestro código apoyados en aplicaciones que nos mejoraron/ayudaban a tener una claridad en el código general del bot. Excepto a la hora de aplicar la API-Key y la IA en los comandos tuvimos que documentarnos ya que esto era más difícil porque la información era escasa en cualquier página web o incluso IA ya que nadie explicaba por ejemplo el funcionamiento de un .env y como se debía aplicar sobre los comandos que utilizaban IA.

Gracias a este proyecto hemos mejorado la organización, comunicación, desarrollo personal a la hora de trabajar y la forma en que vemos y desarrollamos ideas que simplemente está en nuestra mente. Porque puedes pensar en crear algo pero al final uno también se da cuenta que debe pensar en el trabajo que hay detrás para crear ese proyecto que uno imagina.

Visión de futuro

Este bot a futuro podría ser perfectamente un bot popular y que la gente utilizaría en sus servidores. Muchas comunidades por ejemplo requieren de asistentes personales en los tickets de discord para poder responder las consultas o las dudas. Pero qué pasaría si aplicamos nuestro bot con el sistema de IA integrado que es capaz de acceder a un manual otorgado por nosotros + acceso web esto quitaría a los asistentes personales que tienen que responder los tickets 1 a 1 y la IA aria todo el trabajo por ellos y solucionaría uno de los grandes problemas.

No solo eso, consideramos que esto mejora totalmente la organización del servidor si siguiéramos desarrollando e implementando más funcionalidades que otros bots cumplen aunque fueran funciones no tan populares nuestro bot llegaría más lejos y la gente preferiría tener SyncroniBot a 30 bots distintos con 30 formas de ejecutar sus comandos distintas. Ya que si tienes solo Syncronbot solo tendrías que ejecutar sus comandos de la misma manera siempre sin necesidad quizás de usar en un bot símbolos distintos para lanzar los comandos. Son detalles que a primera vista no son importantes pero una vez los utilizas te das cuenta de la comodidad que pueden llegar a otorgar.

El desarrollo de este bot está planificado para que continúe de parte de uno de los dos desarrolladores para uso propio y no publicarlo pero si seguir con una documentación para futuros proyectos. Las futuras integraciones al bot serían:

Futuras integraciones (Si continuáramos)	
E1	Un panel web para personalización
E2	Comandos para hacer publicaciones en redes sociales
E3	Implementación de la IA en los tickets
E4	Implementación de distintos tipos de IA
E5	Actividad verdadera 24/7 alojada en un hosting
E6	Capacidad para interactuar con los usuarios sin necesidad de comandos como eventos aleatorios en el que el bot responda como un usuario para hacer algo más divertido
E7	Que el bot pueda hacerte subir de niveles en un panel dependiendo de qué tanto usas el bot
E8	Implementación de Mini juegos con el bot como un botón de pesca

En resumen este bot podría seguir en desarrollo más enfocado a lo profesional si quitamos el enfoque personal y podría incluso venderse a empresas que utilizan discord como método de trabajo por la comodidad o incluso a comunidades grandes que carecen de gestión y un bot como el nuestro sería capaz de solucionar eso.

Errores Documentados.

A lo largo del desarrollo de este bot nos hemos encontrado con errores transitorios y errores persistentes así que los hemos documentado todos los más importantes de cada sección y como los hemos parcheado.

Errores Página web

Errores	Explicacion
ERR_CONNECTION_REFUSED	El navegador se estaba intentando conectarse al servidor pero este no le escuchaba, esto pasa porque el node.js no se estaba ejecutando correctamente
ERR_CONNECTION_TIMED_OUT	Se estaba intentando conectar a través del pueto 3000 desde afuera de la máquina virtual, pero el bastión de isard solo deja por el puerto 80, lo que hacia que la peticion se quedarse esperando a que sea aceptada.
La sesión no se mantenía entre páginas	Cuando tu inicias sesión en la página web la sesion solo se quedaba iniciada en una pagine pero en las demas no se iniciaba la sesion, esto pasaba porque cada html es independiente y ninguna tenia el codigo para comprobar si el token se habia guardado.
Apache y Node tenían el mismo puerto	Apache ya usaba el 80 para servir la web, y cuando Node.js intentaba usarlo también, fallaba. Lo solucionamos con el proxy inverso, que hace que Apache reciba todo y reenvía las peticiones `/api/` a Node.js en el puerto 3000.

Errores De la base de datos

Errores	**Explicacion**
SyntaxError: Unexpected end of input	Al editar el server.js muchas veces el archivo se quedaba incompleto le faltaba el cierre final, esto causaba que cuando ejecutaba el node.js él lo leía pero cuando llegaba al final no encontraba el cierre y por eso fallaba.
Unknown column 'usuarios' in 'field list **Unknown column 'gmail' in 'field list'**	En el server.js la consulta de registro decía: INSERT INTO usuarios (usuarios, gmail, contraseña) pero en realidad tenia estas columnas: id | nombre | email | contraseña La columna de usuarios no se llamaba usuarios si no se llamaba nombre por eso daba ese error.
Cannot find module 'mysql2'	Este error es tanto del node.js como de la base de datos, ya que el paquete mysql 12 que le permite al node.js conectarse con mysql no estaba instalado.

Errores del Bot

Errores	Explicacion
Compatibilidad con las dependencias	Muchas de las funciones de un bot de discord requieren de dependencias. Estas necesitan que el código del bot estén enfocadas a estas, pero el grave problema es que si la versión de la dependencia esta actualizada o se actualiza la configuración del bot tiene que cambiar también
Error 400 - Solicitud incorrecta (Bad Request)	El bot envió datos mal formateados o incompletos a la AP de Discord
ERROR 3001 Dispositivo de audio	Si el bot gestiona audio, en nuestro caso tiene la capacidad de poner música, este error nos lo devuelve para mostrarnos que no es capaz de simplificar el formato de video y reproducirlo
Rate Limits	El bot está enviando demasiadas solicitudes demasiado rápido
Interacciones fallidas	El bot no puede leer mensajes, escribir, o gestionar roles (Ej. Missing Permissions)
Error 50006 (Cannot send an empty message)	Intentaste enviar un mensaje de texto, ni “embed” ni archivos adjuntos que pueda reconocer.
Error 4000 a 4009 (Gateway Close Codes)	Errores de conexión interna, Por ejemplo si el token del bot se caduca será incapaz de encontrar la ruta a los servidores de Discord
ECONNRESET	La conexión fue cerrada forzosamente por el servidor (el mínimo corte de internet puede provocar esto)
ETIMEDOUT	El bot intentó conectarse al servidor pero no respondió a tiempo
ENOTFOUND	El bot no puede resolver la dirección de Discord (generalmente por falta total de internet o por fallos de DNS)
API-KEY error 404	Estábamos tratando de conectar a Gemini 1.5 cuando ya no da soporte así que pasamos a Gemini 2.5 Pro
V1Beta	Era incompatible totalmente el bot
You are exceeded of limits in this API-Key	Las API-Key por desgracia al final son un acceso a las capacidades de una IA que no es propia y contiene una licencia. Así que debes pagar por ellas para poder utilizarlas no existe ninguna API-Key funcional gratuita ni en OpenSource

Otros capítulos

Durante el desarrollo de SyncronyBot también se tuvieron en cuenta otros aspectos importantes además de las funciones principales del bot. Uno de los puntos más importantes fue la seguridad, ya que el bot trabaja directamente con mensajes, configuraciones y datos de usuarios dentro de los servidores de Discord. Por este motivo, se pensó en limitar ciertas funciones únicamente a administradores o usuarios autorizados, evitando así posibles problemas o un mal uso del sistema. Además, se intentó que la información almacenada estuviera protegida y organizada correctamente para evitar pérdidas de datos o errores de funcionamiento.

También se pensó desde el principio en que el proyecto pudiera crecer y seguir mejorando en el futuro. El bot está dividido en diferentes partes independientes, lo que permite añadir nuevas funciones sin tener que rehacer todo el sistema desde cero. Gracias a esta organización, el mantenimiento resulta más sencillo y las futuras actualizaciones pueden realizarse de manera más rápida y cómoda. Esto también facilita corregir errores o cambiar partes del proyecto sin afectar al resto de funciones.

Otro aspecto importante fue conseguir que el bot funciona de manera estable y continua. Para ello, se decidió utilizar servidores en la nube y herramientas que permiten mantener el sistema activo las 24 horas del día. De esta forma, el bot puede responder a los usuarios en cualquier momento sin necesidad de que un ordenador personal esté siempre encendido. Además, se realizaron diferentes pruebas para comprobar el rendimiento del sistema y detectar posibles fallos antes de su uso.

Durante el desarrollo del proyecto aparecieron varios problemas técnicos relacionados con la página web, la conexión con la base de datos y el propio funcionamiento del bot de Discord. Algunos errores estaban relacionados con conexiones incorrectas, permisos, problemas de puertos o configuraciones mal realizadas. También surgieron errores con dependencias y librerías necesarias para el funcionamiento del proyecto. Resolver todos estos problemas ayudó a mejorar el funcionamiento general del sistema y permitió aprender mucho más sobre programación, administración de sistemas y resolución de errores reales.

Por último, también se intentó que SyncronyBot fuera fácil de utilizar para cualquier persona. El objetivo era crear una herramienta sencilla, útil y cómoda tanto para administradores como para usuarios normales del servidor. Gracias a esto, el bot puede ayudar a ahorrar tiempo, organizar mejor la información y mejorar la experiencia general dentro de Discord.

4 Video sobre las funciones de SyncroniBot

Ya que el contenido de nuestro bot es demasiado para una simple explicación hemos decidido preparar un video para resumir todas las funcionalidades que podemos ofrecer a nuestros usuarios. Con explicaciones detalladas de qué hace cada comando y cómo funciona nuestro bot.

Enlace del video:

- https://youtu.be/msOiJJewKXE

En el video podemos ver un poco el funcionamiento del bot

PD: en la parte final falto colocar que estamos mostrando una funcionalidad que es la /wiki (fandom) or (wikipedia)

Nuestro bot tiene almacenado wikipedia y fandom para poder entrar he investigar información a una velocidad mas rapida. Así si un usuario quiere buscar información académica puede buscar en la wikipedia y si es para el entorno gaming sobre juego puede usar el fandom que es una popular página sobre juegos.

5 Conclusiones

4.1 Conclusiones generales del proyecto

Tras haber trabajado durante todo este tiempo en el desarrollo del bot, podemos concluir que el proyecto ha supuesto un gran aprendizaje tanto a nivel técnico como organizativo. La funcionalidad principal del proyecto ha sido la implementación de una inteligencia artificial integrada en un bot de Discord, algo poco común, ya que normalmente los bots disponibles suelen centrarse en una única función específica.

En nuestro caso, hemos logrado desarrollar un sistema con múltiples funcionalidades útiles, pulidas y funcionales, orientadas a facilitar la gestión de comunidades dentro de Discord. Durante el desarrollo hemos aprendido cómo funciona la creación de un bot, el uso de dependencias, la programación de comandos y la conexión entre procesos y bases de datos. Además, uno de los aspectos más importantes ha sido comprender cómo integrar diferentes herramientas dentro de un mismo proyecto, combinando las funcionalidades tradicionales de un bot con el uso de inteligencia artificial.

A lo largo del proceso también hemos entendido la importancia de la planificación previa, ya que una buena estructura inicial ha permitido que el desarrollo posterior fuera más ordenado y eficiente. La división del proyecto en módulos ha facilitado la organización del código y ha hecho posible trabajar en diferentes partes del sistema de forma independiente sin afectar al funcionamiento global.

El objetivo ha sido demostrar que es posible centralizar distintas herramientas en un único sistema organizado y eficiente, manteniendo funcionalidades similares a las de otros bots pero añadiendo características más avanzadas, como la IA integrada. Esta centralización no solo mejora la experiencia del usuario, sino que también reduce la necesidad de depender de múltiples servicios externos, lo que simplifica la gestión general del sistema.

Durante el desarrollo también se han afrontado diversos retos técnicos, como la depuración de errores en comandos, la optimización de respuestas de la inteligencia artificial y la gestión de la comunicación entre distintos servicios. Estos problemas han permitido mejorar la capacidad de análisis y resolución de errores, habilidades fundamentales en cualquier proyecto de programación.

Otro aspecto relevante ha sido el aprendizaje relacionado con el uso de APIs externas y la integración de servicios de terceros. Esto ha permitido comprender mejor cómo funcionan las aplicaciones modernas, donde diferentes sistemas trabajan de manera conjunta para ofrecer una experiencia completa al usuario. También se ha aprendido la importancia de gestionar correctamente los datos y las respuestas para garantizar un funcionamiento estable y seguro.

Asimismo, el proyecto ha contribuido a mejorar la capacidad de trabajo en equipo.

Finalmente, estamos orgullosos del trabajo realizado y consideramos que se trata de un proyecto original con potencial de uso en un entorno profesional en el futuro. Aunque algunas funciones podrían seguir mejorándose, como la optimización del rendimiento, la ampliación de capacidades de la inteligencia artificial o la incorporación de nuevas herramientas de administración, estamos satisfechos con el resultado final obtenido. Este proyecto no solo ha servido como desarrollo técnico, sino también como una experiencia completa de aprendizaje y crecimiento en el ámbito de la programación y la gestión de proyectos.

4.2 Consecución de los objetivos

El objetivo principal del proyecto era crear un bot multifunción para Discord que ayudará a mejorar la organización y la comunicación dentro de los servidores. Este objetivo se ha conseguido en gran parte, ya que SyncronyBot es capaz de realizar diferentes funciones útiles que ayudan tanto a los administradores como a los usuarios normales del servidor. Gracias a estas funciones, muchas tareas que antes se hacían manualmente ahora pueden realizarse de forma automática y más rápida.

Entre los objetivos cumplidos destaca la creación de un sistema capaz de resumir conversaciones largas para facilitar la lectura de la información importante. También se desarrolló una función de moderación que permite detectar spam, mensajes inapropiados o palabras prohibidas, ayudando a mantener un entorno más limpio y seguro dentro de Discord. Además, el bot puede automatizar algunas tareas y gestionar información de manera más organizada.

Otro de los objetivos alcanzados fue crear una estructura organizada y modular para el proyecto. Esto significa que el bot está preparado para seguir creciendo en el futuro, permitiendo añadir nuevas funciones sin tener que rehacer todo el sistema desde el principio. Gracias a esto, el proyecto puede seguir mejorando con el tiempo de manera más sencilla.

Aun así, algunas funciones todavía necesitan mejoras o más tiempo de desarrollo. Las partes relacionadas con inteligencia artificial y algunas funciones de la página web todavía pueden ampliarse para ofrecer mejores resultados y más opciones a los usuarios. Sin embargo, el proyecto ya funciona correctamente y cumple con la mayoría de los objetivos planteados al inicio, demostrando que la idea principal del proyecto es viable y útil.

4.3 Valoración de la metodología y planificación

En este apartado, la organización del proyecto ha sido uno de los principales retos. Al principio decidimos alojar el bot y la página web en el mismo servidor para mejorar el rendimiento y simplificar el desarrollo. Separarlos habría requerido una arquitectura cliente-servidor más compleja, algo que no era viable por falta de tiempo, ya que el bot necesitaba un desarrollo largo y con margen de errores.

Durante las primeras fases, nos centramos demasiado en la documentación y en la página web, lo que provocó que el desarrollo del bot se retrasara y que la web aún presentara errores en la base de datos. Esto nos obligó a reorganizar el trabajo y cambiar la planificación inicial. A partir de ese momento, distribuimos mejor las tareas, alternando el desarrollo del bot y de la web, además de dividir también la documentación y la programación entre ambos desarrolladores

4.4 Visió de futuro

En el futuro, SyncronyBot puede seguir evolucionando y ampliándose con muchas mejoras y nuevas funciones que lo hagan más completo, útil y fácil de usar en el día a día. El proyecto todavía tiene bastante margen de crecimiento, ya que la idea principal no es que se quede como una herramienta cerrada, sino que pueda ir mejorando con el tiempo según las necesidades de las comunidades que lo utilicen.

Una de las mejoras más importantes sería el desarrollo de una página web más avanzada, clara y sencilla de manejar. Esta web permitiría a los administradores controlar todas las opciones del bot de una forma más visual y cómoda, sin depender tanto de comandos dentro de Discord. De esta manera, tareas como la configuración, la gestión de permisos o la activación de funciones serían mucho más rápidas y accesibles, incluso para usuarios con menos conocimientos técnicos.

También se podría seguir trabajando en la parte de inteligencia artificial, que es uno de los puntos más interesantes del proyecto. La idea sería hacer que el bot entienda mejor las preguntas de los usuarios y pueda responder de una forma más natural, precisa y útil. Además, podría ayudar a resumir conversaciones largas, dar soporte dentro de los servidores o incluso actuar como asistente para tareas concretas dentro de la comunidad. Esto haría que el bot no solo fuera una herramienta de administración, sino también un apoyo activo para los usuarios.

Otra línea de mejora sería añadir más opciones de personalización. Cada servidor de Discord tiene su propia forma de funcionar y su propio estilo, por lo que sería interesante que SyncronyBot pudiera adaptarse a cada uno de ellos. Esto podría incluir cambiar la forma en la que responde, activar o desactivar módulos concretos, ajustar niveles de automatización o incluso modificar la apariencia de algunos mensajes.

Además, se podría ampliar la conexión del bot con otros servicios externos, como redes sociales, plataformas de contenido o herramientas de organización. Esto permitiría automatizar aún más tareas, como publicar contenido de forma automática, sincronizar información entre plataformas o gestionar notificaciones sin intervención manual. De esta forma, el bot se convertiría en un sistema mucho más completo y centralizado.

También sería importante mejorar el rendimiento general del sistema, especialmente pensando en servidores grandes con muchos usuarios activos al mismo tiempo. Optimizar el uso de recursos, reducir tiempos de respuesta y evitar sobrecargas serían aspectos clave para garantizar que el bot funcione de manera estable en cualquier situación.

Por otro lado, en el futuro también se podría mejorar la estructura interna del proyecto, haciendo el código más limpio, organizado y fácil de mantener. Esto facilitaría que otros desarrolladores pudieran trabajar en el bot o añadir nuevas funciones sin complicaciones, lo que ayudaría a que el proyecto creciera de forma colaborativa.

En general, SyncronyBot tiene un potencial muy grande de evolución. Con el tiempo, puede transformarse en una herramienta mucho más potente, flexible y estable, capaz de adaptarse a diferentes tipos de comunidades y necesidades dentro de Discord, convirtiéndose en una solución cada vez más completa y profesional.

6. Glosario

Término	Definición
Discord	Plataforma de comunicación donde funciona el bot SyncronyBot.
Bot	Programa automático que realiza tareas dentro de Discord.
Discord.js	Herramienta utilizada para programar el bot en JavaScript.
Inteligencia Artificial (IA)	Tecnología que permite al bot analizar textos y generar respuestas automáticas.
Python	Lenguaje de programación utilizado para las funciones de inteligencia artificial.
Base de datos	Sistema donde se guarda la información y configuraciones del bot.
MongoDB	Base de datos utilizada en el proyecto para almacenar datos.
API	Sistema que permite la comunicación entre diferentes programas o servicios.
Spam	Mensajes repetitivos o no deseados enviados dentro de un servidor.
Moderación	Control de mensajes y comportamiento de los usuarios dentro del servidor.
Servidor Cloud	Servidor en internet que permite que el bot esté activo las 24 horas.
Docker	Herramienta que facilita ejecutar el proyecto en diferentes sistemas sin problemas.
Panel web	Página de administración para controlar las funciones del bot.

Automatización	Realización automática de tareas sin intervención manual.
Traducción automática	Función que permite traducir mensajes entre diferentes idiomas.
NLP	Tecnología utilizada para que la IA pueda entender y analizar texto.
GitHub	Plataforma utilizada para guardar y organizar el código del proyecto.
Node.js	Entorno utilizado para ejecutar el bot y la parte web del proyecto.

7. Bibliografia

Webs consultadas en el proceso del proyecto que hemos desarrollado.

Recurso		***Descripción***	***Enlace***	***Fecha de consulta***
E1	BotGhost	Plataforma para crear bots de Discord	https://botghost.com/	18/01/2026
E2	Autocode	Herramienta para desarrollar bots y automatizaciones	https://autocode.com/	02/02/2026
E3	Discord Bot Studio	Programa para crear bots de forma visual (descartado: mejor uso de terminal)	https://discordbotstudio.org/	14/02/2026
E4	GPTBots	Plataforma de bots con inteligencia artificial	https://gptbots.ai/	27/02/2026
E5	Quickchat AI	Creación de bots conversacionales con IA	https://quickchat.ai/	05/03/2026
E6	Discord Developer Portal	Documentación oficial para el desarrollo de bots	https://discord.com/developers/docs	11/03/2026
E7	Discord.js	Librería para crear bots en JavaScript	https://discord.js.org/	19/03/2026
E8	Node.js	Entorno de ejecución del proyecto	https://nodejs.org/	28/03/2026
E9	Express.js	Framework para backend y página web	https://expressjs.com/	04/04/2026

E10	MongoDB	Base de datos utilizada en el proyecto	https://www.mongodb.com/docs/	09/04/2026
E11	MySQL	Documentación oficial de MySQL	https://dev.mysql.com/doc/	15/04/2026
E12	HTML	MDN Web Docs – HTML	https://developer.mozilla.org/es/docs/Web/HTML	18/04/2026
E13	CSS	MDN Web Docs – CSS	https://developer.mozilla.org/es/docs/Web/CSS	22/04/2026
E14	JavaScript	MDN Web Docs – JavaScript	https://developer.mozilla.org/es/docs/Web/JavaScript	26/04/2026
E15	Bootstrap	Framework utilizado para el diseño de la página web	https://getbootstrap.com/	30/04/2026
E16	GitHub	Plataforma para el control de versiones y almacenamiento del código	https://docs.github.com/	03/05/2026
E17	Trello	Herramienta de organización y planificación del proyecto	https://support.atlassian.com/trello/	05/05/2026
E18	Railway	Plataforma cloud utilizada para el despliegue del proyecto	https://docs.railway.com/	07/05/2026
E19	Render	Servicio utilizado para alojar aplicaciones web	https://render.com/docs	09/05/2026
E20	freeCodeCamp	Tutorial para crear un bot de Discord con Node.js	https://www.freecodecamp.org/espanol/news/crea-un-bot-de-discord-con-javascript-nodejs/	10/05/2026

E21	Hostinger	Guía para alojar un bot de Discord	https://www.hostinger.com/es/tutoriales/como-alojar-un-bot-de-discord	11/05/2026
E22	Back4App	Pasos para crear e implementar un bot de Discord	https://blog.back4app.com/es/pasos-para-crear-e-implementar-un-bot-de-discord/	12/05/2026

8 Anexos

Nuestro bot realmente parece no tener tanto trabajo pero detrás del bot requiere de mucho esfuerzo, entender cómo funciona y saber bien montar los archivos. Algo interesante a comentar es que el bot para funcionar debe tener una estructura concreta de directorios.

Anexo		Contenido
An1	Ejemplo de comandos	/resumen /ask /mute /ban /encuesta /wiki /resumen /rango
An2	Estructura de datos	Usuarios, configuraciones, permisos y actividad de bot para personalizar
An3	Problemas encontrados	Errores de conexión,permisos,configuración, dependencias, API y conexión.
An4	Instalación basica	Descargar proyecto, instalar dependencias, configurar base de datos, añadir token +API, programar cada comando y estructura del bot. Actualizar diariamente las dependencias y agregar un método de despliegue 24/7
An5	Herramientas usadas	Discord.js,Node.js,Python,html,MySQL,GitHub, Docker,discord.dev.portal, redit, vs code, java y np2
An6	Directorios	Syncronybot-Commands/events/config
An7	Archivos imprescindibles	Ready.js InteractionCreate.js Config.js deploy-commands.js .env package.jason index.js
An8	Dependencias	google.ai@latest music mongoose pm2 ente muchisimas mas son mas de 120 instaladas ya que dependen de que vayas a hacer con el bot.