Discord Bot manager

TRABAJO FINAL DE CICLO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Pelayo Palacio Suárez

curso académico 2024/2025

Oviedo, FECHA

Contenido

[**1. Descripción general del proyecto** 4](#_Toc177584882)

[ Introducción: Resumen de las características del proyecto 4](#_Toc177584883)

[ Presentación de las características 4](#_Toc177584884)

[ Alcance del proyecto: Detalle del problema planteado indicando, punto por punto, todos los detalles que se van a gestionar. 5](#_Toc177584885)

[ Justificación y análisis de la realidad 7](#_Toc177584886)

[ Marco legal 9](#_Toc177584887)

[ Marco teórico 9](#_Toc177584888)

[ Temporalidad 14](#_Toc177584889)

[**2. estudio de la viabilidad del sistema** 15](#_Toc177584890)

[ Análisis económico (DAFO) 15](#_Toc177584891)

[ Plan de marketing 15](#_Toc177584892)

[**3. Descripción del entorno tecnológico** 16](#_Toc177584893)

[ Descripción de todos los perfiles de usuario que habrá en la aplicación 16](#_Toc177584894)

[ Descripción de las tecnologías para cada perfil 17](#_Toc177584895)

[**4. Especificación de requisitos** 18](#_Toc177584896)

[ Requisitos funcionales 18](#_Toc177584897)

[ Requisitos no funcionales 19](#_Toc177584898)

[**5. Modelo entidad / relación** 21](#_Toc177584899)

[ Explicar por qué ha salido ese E/R, de dónde salen las Entidades y de dónde salen las Relaciones. 21](#_Toc177584900)

[ Plasmar el modelo E/R 21](#_Toc177584901)

[ Explicar el origen de los atributos 21](#_Toc177584902)

[ Explicar por qué se han elegido esas claves primarias. 21](#_Toc177584903)

[ Modelo Relacional 21](#_Toc177584904)

[**6. Diagrama de procesos** 21](#_Toc177584905)

[ Casos de Uso 21](#_Toc177584906)

[ Diagrama de clases 21](#_Toc177584907)

[**7. Diseño Interfaz** 21](#_Toc177584908)

[ Diagramación y prototipado del diseño gráfico de la interfaz 21](#_Toc177584909)

[ Guías de estilo 23](#_Toc177584910)

[ Mapa de navegación 24](#_Toc177584911)

[**8 Pruebas** 24](#_Toc177584912)

[ Pruebas unitarias 24](#_Toc177584913)

[ Pruebas de integridad 24](#_Toc177584914)

[**9. Manual de usuario** 24](#_Toc177584915)

[ Manual del cliente 24](#_Toc177584916)

[ Manual de instalación y despliegue 24](#_Toc177584917)

[**10. CONCLUSIONES** 24](#_Toc177584918)

[**11. Anexo** 24](#_Toc177584919)

[ Bibliografía 24](#_Toc177584920)

[ Referencias a texto y/o web de ayuda 24](#_Toc177584921)

[ Referencia de imágenes 24](#_Toc177584922)

# **1. Descripción general del proyecto**

* Introducción: Resumen de las características del proyecto

El proyecto pretende cumplir dos funciones principales. La primera siendo ser una forma de aplicar los contenidos estudiados en la FP de Desarrollo de Aplicaciones Multiplataforma en un caso real de uso sin cerrarse sólo a los propios contenidos proporcionados por los profesores. Y la segunda permitir el control de un Bot de Discord de forma local tanto en aplicación de escritorio como en una versión móvil permitiendo a los usuarios no depender de un host remoto y tener control total sobre los datos.

Dentro de las funciones disponibles se encontrará los comandos básicos de moderación, log de dichas acciones, manipulación de servidores, etc.…

* Presentación de las características

Aplicación de escritorio:

La aplicación de escritorio permitirá al usuario ejecutar el Bot, tener acceso al sistema de módulos de este permitiendo desactivar ciertas funciones en ejecución o previo a su lanzamiento mediante el uso de un archivo de texto para su configuración. Se pretende que la versión de escritorio sea la que permita la modificación de funcionalidades. Deberá ser capaz de mostrar información del servidor y sus usuarios mediante la id\_server o id\_user, así como las acciones de los moderadores quedando todo ello registrado en una base de datos.

Aplicación móvil:

La aplicación móvil tendrá la función de hostear de forma local un bot de Discord dando uso a dispositivos viejos o en desuso.

Bot de Discord:

El Bot deberá ser capaz de realizar funciones básicas de moderación, administración de servidores, modificación de estos, control de usuarios, obtención de información de ellos, …

* Alcance del proyecto: Detalle del problema planteado indicando, punto por punto, todos los detalles que se van a gestionar.

Alcance del proyecto:

El proyecto tiene como objetivo principal proporcionar una solución accesible y personalizable para la gestión de bots de Discord a través de una aplicación multiplataforma. Esta solución aborda varios problemas que enfrentan los usuarios al administrar y personalizar bots, como la dependencia de servicios de hosting remotos, la complejidad de configurar funcionalidades avanzadas, y la falta de control directo sobre los datos y el rendimiento del Bot.

Problemas planteados:

* + Dependencia de un hosting remoto: Los bots de Discord suelen necesitar servidores externos para ejecutarse, lo que implica costos adicionales, falta de control sobre el entorno y riesgos de seguridad.
  + Falta de accesibilidad a funciones avanzadas: Muchas requieren de servicios premium o gran conocimiento técnico para implementarlos.
  + Carencia de aplicaciones locales para el manejo de los bots: La mayoría de bots de Discord tienen una dashboard online a través de la cual se controlan.
  + Seguridad y privacidad: El hosting local permite evitar problemas de seguridad al no tener control sobre los protocolos con los que se tratan los datos.
  + Facilidad de uso: La mayoría de bots son de fácil uso, aunque no disponen de una estructura estándar de desarrollo.

Detalles que gestionar:

* + Creación de una aplicación de escritorio
    - Desarrollo de una interfaz que permita al usuario:
      * Configurar módulos y comandos del Bot.
      * Ejecutar el Bot localmente y actuar como host para otras aplicaciones.
      * Visualizar información relevante de servidores y usuarios (ID, acciones, registros, etc.).
    - Implementación de una base de datos para registrar actividades y datos del Bot.
  + Desarrollo de una aplicación móvil
    - Función principal como host para bots en dispositivos móviles.
    - Configuración del bot al igual que la de escritorio pero con menos acceso a datos.
    - Soporte para dispositivos antiguos o en desuso, maximizando el aprovechamiento del hardware existente.
  + Diseño y funcionalidad del Bot de Discord
    - Comandos básicos para moderación (gestión de usuarios, permisos, logs).
    - Herramientas de administración de servidores, como configuraciones automáticas y reportes.
    - Opciones de personalización para adaptarse a necesidades específicas de servidores de Discord.
  + Gestión de seguridad y datos
    - Hay que asegurar que toda la información manejada esté protegida contra accesos no autorizados.
    - Cumplir con las normativas de privacidad de datos aplicables a Discord y a los usuarios finales.
  + Soporte multiplataforma y escalabilidad
    - Garantizar una experiencia fluida y consistente entre las versiones de escritorio y móvil.
    - Facilitar actualizaciones y mejoras para mantener la compatibilidad con la API de Discord.
  + Facilidad de uso y documentación
    - Desarrollo de manuales detallados para instalación, configuración y uso.
    - Diseño de una interfaz intuitiva que reduzca la curva de aprendizaje, permitiendo a usuarios sin experiencia técnica aprovechar el producto.
* Justificación y análisis de la realidad

Justificación del proyecto:

* + Demanda creciente de bots personalizados:
    - Desarrollo A medida que las comunidades de Discord crecen en tamaño y complejidad, existe una necesidad cada vez mayor de bots que puedan adaptarse a las necesidades específicas de cada servidor. Sin embargo, muchas soluciones disponibles son genéricas o difíciles de personalizar sin conocimientos técnicos avanzados.
  + Control total del entorno y los datos:
    - La mayoría de los bots de Discord dependen de servicios de hosting remoto, lo que implica riesgos relacionados con la privacidad, la seguridad y la dependencia de terceros. Este proyecto permite a los usuarios gestionar el Bot localmente, ofreciendo control total sobre sus datos y funcionalidades.
  + Accesibilidad y facilidad de uso:
    - La posibilidad de usar el Bot tanto en aplicaciones de escritorio como móviles abre la puerta a una mayor flexibilidad. Además, el diseño multiplataforma facilita el acceso y uso de esta herramienta, incluso para aquellos con recursos limitados, como dispositivos antiguos o en desuso.
  + Aplicación práctica de conocimientos técnicos:
    - Este proyecto no solo resuelve un problema práctico, sino que también sirve como ejercicio para aplicar los conocimientos adquiridos en el ciclo formativo de Desarrollo de Aplicaciones Multiplataforma, integrando múltiples tecnologías y principios de diseño en un caso real de uso.

Análisis de la realidad:

* + Popularidad de Discord:
    - En los últimos años, Discord se ha posicionado como una de las plataformas de comunicación más utilizadas, con más de 300 millones de usuarios registrados y comunidades activas en todo el mundo. Esta popularidad genera un ecosistema ideal para el desarrollo de herramientas avanzadas como este proyecto.
  + Competencia y mercado actual:
    - Actualmente, existen varios bots con funcionalidades avanzadas (como MEE6, Dyno, o Carl-Bot), pero muchos de ellos operan bajo modelos de negocio freemium que limitan las funciones más atractivas a usuarios pagos. Este proyecto busca posicionarse como una alternativa flexible, de bajo costo y con posibilidades de personalización.
  + Avances tecnológicos y accesibilidad:
    - Con el auge de dispositivos móviles más potentes y el acceso a tecnologías multiplataforma, los usuarios esperan soluciones que puedan ser gestionadas desde cualquier lugar. Este proyecto se alinea con esa expectativa al ofrecer una integración completa entre escritorio y móvil.
  + Retos legales y éticos:
    - La gestión de datos personales y las actividades de moderación requieren un enfoque cuidadoso para cumplir con las normativas internacionales de privacidad, como el Reglamento General de Protección de Datos (GDPR) en Europa. El proyecto incluye mecanismos para garantizar que los datos se gestionen de manera segura y ética.
  + Crecimiento y evolución de la API de Discord:
    - Discord actualiza continuamente su API para incluir nuevas funciones y mejorar la experiencia del usuario. Este proyecto está diseñado con un enfoque escalable que facilita su mantenimiento y actualización frente a estos cambios.
* Marco legal
  + Reglamento General de Protección de Datos (GDPR):
  + Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos. Este reglamento establece las bases para la recopilación, almacenamiento y procesamiento de datos personales en la Unión Europea.
  + Ley Orgánica de Protección de Datos y Garantía de Derechos Digitales (LOPDGDD):
  + Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales, que adapta el GDPR al marco legal español y regula el tratamiento de datos personales en España.
  + Normas de uso de la API de Discord:
  + Discord API Terms of Service (Discord, Inc.). Este documento establece las condiciones legales para el uso de la API de Discord, incluyendo las limitaciones sobre el uso de datos, el respeto a los derechos de los usuarios y las políticas de privacidad.
  + Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico (LSSI-CE):
  + Ley 34/2002, de 11 de julio, que regula las actividades relacionadas con los servicios de la sociedad de la información en España, incluyendo la provisión de servicios digitales y el comercio electrónico.
* Marco teórico
  + 1. Introducción a los Bots de Discord
    - Discord se ha consolidado como una plataforma versátil para la comunicación y gestión de comunidades en diversos ámbitos, incluyendo la educación y el sector profesional. Gracias a su arquitectura abierta y la posibilidad de integrar bots, se ha convertido en una herramienta clave para la automatización de tareas, la gestión de usuarios y la facilitación de interacciones dentro de servidores especializados (López et al., 2021).
  + 2. Uso Educativo de los Bots de Discord
    - El empleo de bots en entornos educativos ha ganado popularidad debido a su capacidad para mejorar la participación estudiantil y automatizar procesos administrativos. Según estudios recientes, los bots en Discord permiten la creación de entornos de aprendizaje más interactivos mediante la integración de recordatorios, evaluaciones automáticas y foros de discusión virtuales (Smith & Johnson, 2022). Además, pueden actuar como tutores virtuales, proporcionando retroalimentación inmediata y facilitando el acceso a materiales de estudio.
    - Un análisis detallado de su aplicación en la educación revela que estos bots pueden aumentar la motivación de los estudiantes al incorporar elementos de gamificación y recompensas basadas en la participación activa(Kumar & Singh, 2021). Así, los bots no solo complementan la enseñanza tradicional, sino que también fomentan el aprendizaje colaborativo y la interacción entre estudiantes.
  + 3. Aplicaciones Profesionales de los Bots de Discord
    - En el sector profesional, los bots de Discord han demostrado ser herramientas eficaces para la gestión de equipos, la automatización de procesos y la organización de eventos virtuales. Según un estudio publicado por García et al. (2023), las empresas han comenzado a implementar bots para la asignación de tareas, el seguimiento de proyectos y la notificación de eventos importantes dentro de sus servidores corporativos.
    - Un caso de estudio señala que la automatización de respuestas y la integración con otras plataformas como Trello o Slack mejoran significativamente la productividad en entornos de trabajo colaborativo (Miller & Thompson, 2020). Además, el uso de inteligencia artificial en los bots permite ofrecer soporte técnico automatizado, reduciendo la carga de trabajo en departamentos de atención al cliente y optimizando la experiencia de usuario.
  + 4. Beneficios y Desafíos
    - Beneficios:
      * **Accesibilidad y disponibilidad**: Los bots operan 24/7, garantizando asistencia constante tanto en el ámbito educativo como en el profesional.
      * **Automatización**: Reducción de carga administrativa mediante la automatización de respuestas, recordatorios y evaluaciones.
      * **Interactividad y engagement**: Mejora de la participación a través de gamificación y respuestas personalizadas.
    - Desafíos:
      * **Seguridad y privacidad**: La recopilación de datos puede representar un riesgo si no se implementan medidas adecuadas de protección.
      * **Mantenimiento y actualización**: Los bots requieren revisiones periódicas para garantizar su funcionamiento y adaptación a nuevas necesidades.
      * **Curva de aprendizaje**: Los usuarios pueden enfrentar dificultades iniciales en la implementación y configuración de los bots.
  + 5. Conclusión
    - Los bots de Discord representan una solución innovadora tanto en el ámbito educativo como profesional. Su capacidad para automatizar tareas, mejorar la interacción y facilitar la gestión de comunidades los convierte en herramientas valiosas en la era digital. A pesar de los desafíos asociados, su implementación adecuada puede generar beneficios significativos, contribuyendo a una mayor eficiencia y participación en distintos entornos.
  + Referencias
    - García, R., López, M., & Pérez, J. (2023). *Implementation of AI-powered bots in professional environments*. En *Proceedings of ACM Digital Conference*. https://dl.acm.org/doi/abs/10.1145/3653707
    - Kumar, A., & Singh, R. (2021). *A Comprehensive Review on Bot Discord Bots for Interactive Learning*. En *International Conference on Computer Science and Engineering*. https://d1wqtxts1xzle7.cloudfront.net/89967831/CSEIT2172100-libre.pdf
    - López, M., Rodríguez, D., & Torres, L. (2021). *Aplicaciones de bots en entornos digitales*. En *LACCEI 2021 Virtual Edition*. https://laccei.org/LACCEI2021-VirtualEdition/full\_papers/FP9.pdf
    - Miller, C., & Thompson, B. (2020). *Automated task management and collaboration through Discord bots*. En *Proceedings of the ACM Conference on Human Factors in Computing Systems*. https://dl.acm.org/doi/abs/10.1145/3334480.3382960
    - Smith, J., & Johnson, K. (2022). *Uso de Discord en la educación y sus beneficios*. En *Proceedings of the LearnTechLib Conference*. https://www.learntechlib.org/p/220539/
* Temporalidad
  + Planteamiento de ideas iniciales: Septiembre
  + Inicio de la documentación: Septiembre-Octubre
  + Desarrollo de la lógica de la aplicación
    - Bot: Octubre-Abril
    - Base de datos: Abril-Mayo
    - Persistencia: Abril-Mayo
    - Eventos: Abril-Mayo
  + Github del proyecto: Octubre-Junio
  + Desarrollo de la interfaz:
    - Escritorio: Abril-Mayo
    - Móvil: Mayo
  + Testing con usuarios reales: Mayo
  + Revisiones de los resultados de las pruebas: Mayo

Tabla

El contenido generado por IA puede ser incorrecto.

# **2. estudio de la viabilidad del sistema**

* Análisis económico (DAFO)
* Debilidades
  + El éxito de la aplicación depende de que la plataforma Discord se mantenga estable.
  + Tener aplicación de escritorio y de móvil en lugar de web puede suponer mayores costes de mantenimiento y desarrollo.
  + Curva de aprendizaje alta en caso de querer modificar el código de forma manual.
* Amenazas
  + Competencia: Existen otros bots de Discord con funcionalidades similares que pueden captar a los usuarios.
  + Seguridad: La gestión de datos personales de los usuarios debe ser especialmente cuidada.
  + Evolución tecnológica: La api utilizada y el propio Discord puede ir requiriendo actualizaciones periódicas completas de la aplicación
* Fortalezas
  + Ofrecer de forma condensada gran cantidad de funcionalidades puede atraer a muchos usuarios.
  + La existencia de aplicación móvil puede dar flexibilidad a los usuarios por poder editar funciones en cualquier lugar.
  + El ser un Bot de Discord permite que gran cantidad de usuarios puedan acceder al producto al estar ya dentro de la plataforma.
  + Gran cantidad de información comunitaria existente.
* Oportunidades
  + Crecimiento de la plataforma Discord.
  + Personalización de las funciones según las necesidades del cliente.
  + Posibles alianzas con servidores que promocionen el producto.
  + Monetización en funcionalidades premium.

A diagram of a company

AI-generated content may be incorrect.

* Plan de marketing
  + Análisis de mercado:
    - Público objetivo:
      * Administradores de servidores.
      * Streamers y creadores de contenido con comunidades en Discord.
      * Empresas y grupos de trabajo que utilicen Discord como medio de soporte.
      * Desarrolladores de bots que quieran aprender o entender ciertas funciones.
    - Análisis de la competencia:
      * Existen bots populares que ofrecen todas las funciones, pero de forma individualizada, ninguno proporciona el conjunto completo.
      * Existen frameworks que permiten el desarrollo de bots de forma personal.
      * Existen aplicaciones para el manejo de bots, pero son todas o en su gran mayoría aplicaciones webs.
    - Diferenciación propuesta:
      * Gestión modular.
      * Interfaz simple e intuitiva.
      * Automatización de tareas y control avanzado de usuarios.
      * Adaptable a multitud de sistemas.
  + Estrategias de lanzamiento:
    - Desarrollo y pruebas (beta cerrada para el equipo de desarrollo o en comunidades pequeñas para tener control de todas las variables)
    - Lanzamiento en GitHub como proyecto personal y educativo, sin dar licencia comercial.
    - Promoción y expansión en comunidades ya existentes intentando entrar en contacto con creadores de contenido emergentes que puedan requerir estos servicios.
  + Posibles canales de marketing y promoción:
    - Comunidades existentes de desarrolladores de bots.
    - Redes sociales mediante playlists del desarrollo de esta o promocionando directamente las funciones de la aplicación.
    - Crear un servidor de soporte para agrupar a todos los posibles clientes que vengan desde otros canales.
    - Colaboraciones con otros proyectos para dar el nombre a conocer.
  + Monetización y modelo de negocio:
    - Plan Freemium:
      * Versión gratuita con funcionalidades básicas.
      * Versión premium con acceso a mayor funcionalidad y soporte.
    - Donaciones.
    - Venta de módulos específicos.

# **3. Descripción del entorno tecnológico**

* Descripción de todos los perfiles de usuario que habrá en la aplicación
  + La aplicación contará con los siguientes perfiles de usuario, cada uno con roles y funciones específicas:
    - Administrador del sistema:
    - Perfil destinado a los propietarios del servidor de Discord o usuarios avanzados.
    - Funciones principales:
      * Configuración avanzada del Bot (módulos, comandos personalizados, permisos).
      * Gestión de los datos almacenados en la base de datos.
      * Monitorización y auditoría de los registros de acciones del Bot.
  + Moderador del servidor:
    - Usuarios encargados de la moderación dentro de los servidores de Discord.
    - Funciones principales:
      * Uso de comandos de moderación (ban, kick, mute, etc.).
      * Acceso limitado a los registros de moderación.
      * Generación de reportes básicos sobre actividades en el servidor.
  + Usuario final:
    - Miembros regulares del servidor de Discord que interactúan con el Bot.
    - Funciones principales:
    - Uso de comandos básicos definidos por los administradores (consultas, juegos, encuestas).
    - Interacción con funciones no administrativas del Bot.
* Descripción de las tecnologías para cada perfil
  + Administrador del sistema
    - Aplicación de escritorio (JavaFX):
    - Herramienta principal para configurar y personalizar el Bot.
    - Interfaz gráfica diseñada para acceder a funciones avanzadas como:
      * Gestión de módulos y comandos.
      * Visualización de registros almacenados en la base de datos SQL.
      * Control del estado del Bot en tiempo real.
    - Base de datos SQL (MySQL o PostgreSQL):
      * Almacenamiento seguro de configuraciones, registros y datos de usuarios.
      * Soporte para consultas avanzadas y reportes.
  + Moderador del servidor
    - Aplicación móvil (Android en Java):
      * Proporciona acceso rápido y remoto a las funcionalidades del Bot.
      * Funciones clave:
        + Consultar registros básicos de moderación.
        + Ejecutar comandos de moderación desde dispositivos móviles.
    - Conexión al servidor mediante API RESTful:
    - Uso de endpoints seguros para consultar información o ejecutar comandos en el Bot.
  + Usuario final
    - Bot de Discord (JDA - Java Discord API):
      * Interacción directa con los usuarios dentro de Discord.
      * Comandos personalizados accesibles desde la interfaz de Discord (por ejemplo, !ayuda, !stats).
      * Respuestas inmediatas mediante WebSockets.
    - Servidor local o en la nube:
      * Ejecuta el Bot y procesa las solicitudes de los usuarios finales.
      * Escalable para soportar múltiples usuarios y peticiones concurrentes.

# **4. Especificación de requisitos**

* Requisitos funcionales

Los requisitos funcionales definen las funcionalidades que el sistema debe ser capaz de realizar. Estos requisitos están directamente relacionados con las operaciones y tareas que el Bot y las aplicaciones asociadas deben ejecutar.

1. Gestión del Bot
   1. El sistema debe permitir ejecutar el Bot de Discord tanto en una aplicación de escritorio como en una móvil.
   2. El Bot debe ser capaz de realizar acciones de moderación, como banear, silenciar o kickear usuarios.
   3. Debe ser posible agregar, editar y eliminar comandos personalizados del Bot.
   4. El Bot debe registrar todas las acciones de moderación (logs de acciones) en una base de datos SQL.
2. Interfaz de usuario
   1. La aplicación de escritorio debe tener una interfaz gráfica para permitir la configuración del Bot, incluida la personalización de comandos y módulos.
   2. Los administradores deben poder ver un registro de eventos del servidor de Discord y acceder a los detalles de las acciones realizadas por los moderadores.
   3. La aplicación móvil debe permitir a los usuarios conectarse al Bot y ver las configuraciones y registros almacenados en la versión de escritorio.
3. Base de datos
   1. El sistema debe almacenar los datos del Bot (configuraciones, logs, registros de usuarios) en una base de datos SQL.
   2. Los administradores deben poder consultar y actualizar las configuraciones del Bot a través de la aplicación de escritorio.
   3. La base de datos debe ser capaz de manejar consultas optimizadas y mantener la integridad de los datos.
4. Comunicaciones en tiempo real
   1. El sistema debe permitir la comunicación en tiempo real entre el Bot y los usuarios a través de WebSockets.
   2. La aplicación móvil debe conectarse al servidor para ejecutar comandos de moderación y obtener información relevante.
5. Seguridad y privacidad
   1. El sistema debe garantizar la seguridad de los datos almacenados y transmitidos mediante el uso de cifrado SSL/TLS.
   2. Solo los administradores y moderadores deben tener acceso a funciones avanzadas del Bot.
6. Multiplataforma
   1. La aplicación de escritorio debe ser compatible con Windows, macOS y Linux.
   2. La aplicación móvil debe ser compatible con dispositivos Android.
   3. El sistema debe permitir la modificación y personalización de las funciones del Bot a través de ambas plataformas (escritorio y móvil).

* Requisitos no funcionales

Los requisitos no funcionales especifican las características del sistema que no están relacionadas con funciones específicas, sino con el rendimiento, la seguridad y la usabilidad. Estos requisitos son fundamentales para garantizar la calidad y la eficiencia del sistema.

1. Rendimiento
   1. El sistema debe ser capaz de manejar múltiples solicitudes simultáneas sin afectar el rendimiento.
   2. La aplicación debe ser capaz de procesar y almacenar eventos en tiempo real sin demoras notables en la respuesta del Bot.
2. Escalabilidad
   1. La aplicación debe ser escalable para permitir la adición de nuevas funcionalidades sin comprometer el rendimiento o la estabilidad.
   2. La base de datos SQL debe ser capaz de manejar grandes volúmenes de datos a medida que el número de usuarios y servidores gestionados crece.
3. Usabilidad
   1. La aplicación debe ser fácil de usar, con interfaces intuitivas que permitan a los administradores configurar y gestionar el Bot sin necesidad de conocimientos técnicos avanzados.
   2. El diseño de la interfaz debe ser coherente y proporcionar retroalimentación clara al usuario sobre las acciones realizadas.
4. Seguridad
   1. Los datos sensibles, como contraseñas y tokens de Discord, deben ser almacenados de forma segura mediante cifrado.
   2. El sistema debe seguir las mejores prácticas de seguridad para proteger los datos personales de los usuarios conforme a las normativas de privacidad (GDPR, LOPDGDD).
5. Mantenimiento y soporte
   1. El sistema debe ser fácilmente mantenible, con documentación detallada para la instalación, configuración y actualización del Bot y las aplicaciones asociadas.
   2. Se deben ofrecer actualizaciones periódicas para mantener la compatibilidad con las últimas versiones de la API de Discord y mejorar las funcionalidades del Bot.
6. Compatibilidad
   1. El sistema debe ser compatible con versiones recientes de los sistemas operativos de escritorio (Windows, macOS, Linux) y dispositivos móviles (Android).
   2. Las aplicaciones deben ser compatibles con versiones de Java recientes y las bibliotecas necesarias para el funcionamiento del Bot y la base de datos SQL.

# **5. Modelo entidad / relación (pendiente de revisión de la versión más reciente de la API)**

* Explicar por qué ha salido ese E/R, de dónde salen las Entidades y de dónde salen las Relaciones.
  + El modelo E/R surge de la necesidad de almacenar los usuarios, su experiencia(la cantidad de interacciones que tienen en el servidor) y las penalizaciones aplicadas a los mismos.
  + Las entidades del modelo son:
    - Usuario(id\_usuario, tipo\_usuario)
    - Experiencia(id\_experiencia, id\_usuario, nivel, puntos\_xp)
    - Penalización(id\_penalizacion, id\_usuario, id\_admin\_mod, tipo, fecha, razón, duración)
  + Relaciones:
    - Usuario con Experiencia (1 a 1).
    - Usuario con Penalizaciones (1 a muchos, un usuario puede tener varias penalizaciones).
    - Experiencia con Usuario (1 a 1).
    - Penalización con Usuario (1 usuario puede recibir varias penalizaciones).
    - Penalización con Usuario (como id\_admin\_mod, indica quién aplicó la sanción).
* Plasmar el modelo E/R

Diagrama

El contenido generado por IA puede ser incorrecto.

* Explicar el origen de los atributos
  + USUARIO
  + Esta entidad representa a los usuarios del bot de Discord y contiene los siguientes atributos:
    - id\_usuario (PK): Identificador único para cada usuario en la base de datos.
      * Se usa en todas las relaciones para identificar a un usuario de forma única.
    - nombre: Nombre del usuario en Discord.
      * Permite identificar a los usuarios de manera más amigable.
    - discord\_id: ID único proporcionado por Discord.
      * Es un identificador global que garantiza que el usuario es único en Discord.
    - rol: Define si el usuario es user, mod o admin.
      * Se necesita para determinar qué acciones puede realizar el usuario dentro de la aplicación.
  + EXPERIENCIA
  + Esta entidad gestiona el sistema de experiencia de los usuarios y tiene los siguientes atributos:
  + id\_experiencia (PK): Identificador único de la experiencia de cada usuario.
    - Evita duplicidad y permite extender los datos si se necesita más información en el futuro.
  + id\_usuario (FK): Relaciona la experiencia con un usuario específico.
    - Garantiza que cada usuario tenga su propio registro de experiencia.
  + nivel: Nivel del usuario basado en su experiencia.
    - Se usa para calcular progresión y recompensas.
  + puntos\_xp: Puntos de experiencia acumulados por el usuario.
    - Determina el avance del usuario en el sistema de niveles.
* Modelo Relacional

Imagen de la pantalla de un video juego

El contenido generado por IA puede ser incorrecto.

# **6. Diagrama de procesos (pendiente de implementación)**

* Casos de Uso

1. UC1: Consultar su perfil
   1. Actores: Usuario, Moderador, Administrador
   2. Descripción: El usuario puede ver información completa de su perfil dentro del servidor
   3. Precondiciones: Usuario registrado en el sistema
   4. Flujo básico:
      1. Usuario ejecuta comando de perfil
      2. Sistema consulta información del usuario
      3. Sistema muestra datos del perfil
   5. Resultado esperado: Usuario visualiza su información personal, nivel, experiencia y estadísticas
2. UC2: Consultar nivel y experiencia
   1. Actores: Usuario, Moderador, Administrador
   2. Descripción: El usuario puede verificar su nivel actual y puntos de experiencia
   3. Precondiciones: Usuario registrado en el sistema
   4. Flujo básico:
      1. Usuario ejecuta comando de nivel
      2. Sistema consulta progreso de experiencia
      3. Sistema calcula y muestra nivel
   5. Resultado esperado: Usuario visualiza su nivel actual, experiencia acumulada y puntos restantes para el siguiente nivel
3. UC3: Interactuar con comandos básicos
   1. Actores: Usuario
   2. Descripción: El usuario utiliza comandos generales del bot para diferentes funcionalidades
   3. Precondiciones: Usuario con permisos básicos
   4. Flujo básico:
      1. Usuario escribe comando con prefijo establecido
      2. Bot procesa el comando
      3. Bot ejecuta la acción solicitada
   5. Resultado esperado: Bot responde apropiadamente al comando ejecutado
4. UC4: Recibir experiencia por actividad
   1. Actores: Usuario
   2. Descripción: El usuario gana puntos de experiencia automáticamente al participar en el servidor
   3. Precondiciones: Usuario registrado en el sistema
   4. Flujo básico:
      1. Usuario realiza actividad (mensaje, tiempo en voz)
      2. Sistema detecta actividad y calcula EXP
      3. Sistema actualiza experiencia del usuario
   5. Resultado esperado: La experiencia del usuario aumenta y puede subir de nivel si alcanza el umbral requerido
5. UC5: Aplicar penalizaciones
   1. Actores: Moderador, Administrador
   2. Descripción: Aplicar sanciones a usuarios que infringen las normas del servidor
   3. Precondiciones: Usuario con rol de moderador o administrador
   4. Flujo básico:
      1. Moderador selecciona usuario a sancionar
      2. Moderador elige tipo de penalización y motivo
      3. Sistema aplica la sanción y la registra
   5. Resultado esperado: Usuario sancionado recibe la penalización correspondiente y se guarda en el historial
6. UC6: Consultar registros de usuarios
   1. Actores: Moderador, Administrador
   2. Descripción: Revisar el historial de actividad y penalizaciones de cualquier usuario
   3. Precondiciones: Usuario con rol de moderador o administrador
   4. Flujo básico:
      1. Moderador solicita historial de un usuario
      2. Sistema consulta registros en base de datos
      3. Sistema muestra información completa
   5. Resultado esperado: Moderador visualiza el historial completo de actividad y sanciones del usuario solicitado
7. UC7: Gestionar roles básicos
   1. Actores: Moderador, Administrador
   2. Descripción: Asignar o remover roles a usuarios del servidor
   3. Precondiciones: Usuario con permisos para gestionar roles
   4. Flujo básico:
      1. Moderador selecciona usuario
      2. Moderador indica rol a asignar/remover
      3. Sistema actualiza roles del usuario
   5. Resultado esperado: Los roles del usuario son modificados según lo solicitado
8. UC8: Revocar penalizaciones
   1. Actores: Moderador, Administrador
   2. Descripción: Eliminar sanciones activas aplicadas a usuarios
   3. Precondiciones: Usuario con permisos para gestionar penalizaciones
   4. Flujo básico:
      1. Moderador consulta penalizaciones activas
      2. Moderador selecciona penalización a revocar
      3. Sistema cancela la penalización
   5. Resultado esperado: La penalización queda registrada como revocada y sus efectos son eliminados
9. UC9: Configurar parámetros del bot
   1. Actores: Administrador
   2. Descripción: Modificar la configuración general y comportamiento del bot
   3. Precondiciones: Usuario con rol de administrador
   4. Flujo básico:
      1. Administrador accede a configuración
      2. Administrador modifica parámetros
      3. Sistema aplica y guarda cambios
   5. Resultado esperado: La configuración del bot se actualiza según los nuevos parámetros establecidos
10. UC10: Gestionar módulos y comandos
    1. Actores: Administrador
    2. Descripción: Activar, desactivar o configurar módulos y comandos específicos
    3. Precondiciones: Usuario con rol de administrador
    4. Flujo básico:
       1. Administrador selecciona módulo/comando
       2. Administrador cambia estado o configuración
       3. Sistema actualiza disponibilidad
    5. Resultado esperado: Los módulos y comandos se comportan según la nueva configuración establecida
11. UC11: Gestionar permisos de moderadores
    1. Actores: Administrador
    2. Descripción: Asignar o revocar permisos específicos a usuarios moderadores
    3. Precondiciones: Usuario con rol de administrador
    4. Flujo básico:
       1. Administrador selecciona moderador
       2. Administrador modifica permisos
       3. Sistema actualiza permisos del moderador
    5. Resultado esperado: El moderador tiene nuevos permisos asignados o removidos en el sistema
12. UC12: Ver estadísticas del servidor
    1. Actores: Administrador
    2. Descripción: Consultar estadísticas globales de actividad y uso del servidor
    3. Precondiciones: Usuario con rol de administrador
    4. Flujo básico:
       1. Administrador solicita estadísticas
       2. Sistema recopila y procesa datos
       3. Sistema presenta información estadística
    5. Resultado esperado: Administrador visualiza datos estadísticos completos sobre el uso del servidor

Diagrama

El contenido generado por IA puede ser incorrecto.

* Diagrama de clases

# **7. Diseño Interfaz (modificable en la fase de implementación)**

* Diagramación y prototipado del diseño gráfico de la interfaz(pendiente de decisión final sobre tecnologías de visualización a usar)

Interfaz de usuario gráfica, Aplicación, Teams

El contenido generado por IA puede ser incorrecto.Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

* Guías de estilo (pendiente de agregar códigos de colores, etc.)

Teniendo en cuenta que es una aplicación dedicada al manejo de bots utilizados en la red social Discord se intentará en la medida de lo posible imitar los colores de su propia interfaz, es decir, una escala de grises en su modo oscuro con colores de los botones destacados en rojo para las opciones importantes como eliminar o borrar y azul para las que se puedan revertir o sean de movimiento entre ventanas.

Captura de pantalla de computadora

Descripción generada automáticamente

# **8. Pruebas (pendiente de implementación)**

* Pruebas unitarias

|  |  |  |
| --- | --- | --- |
| Componente | Prueba | Descripción |
| Usuario.java | Creación de usuario | Verificar la correcta creación de un usuario con parámetros válidos |
| Usuario.java | Gestión de experiencia | Comprobar la asignación y obtención de experiencia |
| Usuario.java | Gestión de penalizaciones | Validar la adición de penalizaciones a un usuario |
| Experiencia.java | Inicialización | Verificar la inicialización correcta de experiencia |
| Experiencia.java | Subida de nivel | Comprobar el cálculo correcto al subir de nivel |
| Experiencia.java | Relación con Usuario | Validar la relación bidireccional con Usuario |
| Penalizacion.java | Creación de penalización | Verificar la creación correcta de penalizaciones |
| Penalizacion.java | Conversión de duraciones | Comprobar la conversión correcta de duraciones |
| Penalizacion.java | Relaciones | Validar la relación con Usuario y AdminMod |
| Repositorios | Operaciones CRUD | Verificar operaciones básicas (create, read, update, delete) |
| Repositorios | Consultas personalizadas | Comprobar consultas como findByIdUsuario() o findByTipoUsuario() |
| Repositorios | Manejo de errores | Validar el manejo correcto de errores de SQL |
| UsuarioServiceImpl | actualizarExperiencia() | Verificar el funcionamiento con diferentes valores |
| UsuarioServiceImpl | agregarPenalizacion() | Comprobar con distintos tipos de penalización |
| UsuarioServiceImpl | Búsquedas por tipo | Validar búsquedas por tipo de usuario |
| ExperienciaServiceImpl | aumentarExperiencia() | Verificar incluyendo casos de subidas de nivel |
| ExperienciaServiceImpl | subirNivel() | Comprobar con diferentes niveles iniciales y finales |
| ExperienciaServiceImpl | Búsquedas por nivel | Validar métodos de búsqueda por nivel |
| PenalizacionServiceImpl | tienePenalizacionActiva() | Verificar con penalizaciones válidas e inválidas |
| PenalizacionServiceImpl | revocarPenalizacion() | Comprobar funcionamiento correcto de revocaciones |
| PenalizacionServiceImpl | Búsquedas por fecha | Validar búsquedas por rango de fechas |
| Controladores | Delegación a servicios | Verificar la delegación correcta a los servicios |
| Controladores | Manejo de respuestas | Comprobar el manejo de respuestas y errores |
| CommandManager | Registro de comandos | Verificar el registro correcto de comandos |
| CommandManager | Ejecución de comandos | Comprobar la ejecución con diferentes permisos |
| CommandManager | Activación/desactivación | Validar la activación/desactivación de comandos |
| Utils | Validación de IDs | Verificar validación de IDs de usuario |
| Utils | Formateo de mensajes | Comprobar formateo de mensajes de penalización |
| Utils | Conversión de duraciones | Validar conversión entre diferentes formatos de tiempo |

* Pruebas de integridad

|  |  |  |
| --- | --- | --- |
| Componente | Prueba | Descripción |
| DatabaseManager | Inicialización | Verificar la inicialización correcta de la base de datos |
| DatabaseManager | Carga de esquema | Comprobar la carga correcta del esquema SQL |
| DatabaseManager | Conexiones | Validar conexión y cierre de conexiones |
| Servicios/Repos | Integración básica | Verificar que los servicios utilizan correctamente sus repositorios |
| Servicios/Repos | Transacciones complejas | Comprobar transacciones que involucran múltiples repositorios |
| Servicios/Repos | Operaciones en cascada | Validar el manejo de cascadas (eliminación de usuarios, etc.) |
| Controladores/Servicios | Integración básica | Verificar que los controladores utilizan correctamente sus servicios |
| Controladores/Servicios | Flujos completos | Comprobar flujos como registro, actualización de exp y penalización |
| ModuleManager/Bot | Registro de módulos | Verificar el registro y activación correcta de módulos |
| ModuleManager/Bot | Interacción entre módulos | Comprobar la interacción entre diferentes módulos |
| ModuleManager/Bot | Ciclo de vida | Validar el ciclo de vida completo de comandos |
| Integración JDA | Comandos slash | Verificar respuesta a comandos slash |
| Integración JDA | Eventos del servidor | Comprobar manejo de eventos del servidor |
| Integración JDA | Interacciones | Validar interacciones de usuarios |

# **9. Manual de usuario (Pendiente de creación de instalador y aplicación final)**

* Manual del cliente
* Manual de instalación y despliegue

# **10. CONCLUSIONES**

El desarrollo de la aplicación para manejar bots de Discord ha representado un proyecto integral que ha permitido aplicar los conocimientos aprendidos en el curso de desarrollo de aplicaciones multiplataforma y mejorar ciertas habilidades del desarrollo del software como uso de otros patrones de programación, interfaces más complejas a las dadas en clase, entendimiento del uso y necesidad de la persistencia en aplicaciones comerciales.

Se ha conseguido implementar con éxito un sistema completo de gestión de usuarios con múltiples funcionalidades como el seguimiento de experiencia, gestión de penalizaciones y roles administrativos. La arquitectura basada en comandos ha demostrado ser flexible y escalable, permitiendo añadir nuevas funcionalidades sin modificar el núcleo del sistema.

La integración con la API de Discord (JDA) supuso un reto inicial debido a su complejidad, pero la estructura modular implementada ha permitido un desarrollo incremental. La gestión de estados asíncronos y eventos en tiempo real ha requerido implementar patrones de diseño que garanticen la consistencia de los datos y la respuesta adecuada a los usuarios.

Este proyecto ha reforzado la importancia de una buena planificación y diseño de arquitectura previo a la implementación. La separación en capas (controladores, servicios, repositorios y modelos) ha facilitado el mantenimiento y las pruebas. Además, la gestión de dependencias con Gradle ha simplificado la integración de bibliotecas externas.

Dentro de las posibles mejoras a futuro se encuentra la creación de nuevos módulos de comandos más complejos como serían los de reproducción de audio, inserción de gifs en base a contextos o incluso un chat de ia que aprenda del propio chat de Discord del servidor donde se ejecute

# **11. Anexo**

* Bibliografía
* Referencias a texto y/o web de ayuda
* Referencia de imágenes