INFORME DE PROYECTO: MIGRACIÓN DE SISTEMA MONOLÍTICO A MICROSERVICIOS

**Integrantes** : Francisco Gómez

Ricardo Diaz

Joan Rojas

**Asignatura** : Desarrollo FullStack I

**Docente :** VÍCTOR ISIDRO ROSENDO LUGO

**Fecha de entrega : 01/04/2025**

### **1. Definición del Problema**

**1.1 Descripción del sistema monolítico actual**

La empresa chilena ConnectForo SPA dedicada a ofrecer servicios de moderación y generación de contenido a empresas e instituciones que construyen comunidades activas alrededor de sus marcas. Al no contar con una plataforma propia que permita a los clientes crear, administrar y moderar sus foros, actualmente los procesos se llevan a cabo con herramientas externas que carecen de control, escalabilidad y personalización. Esto genera dificultades para monitorear conversaciones, aplicar políticas de comunidad y ofrecer una experiencia fluida tanto como para moderadores y usuarios.

**1.2 Identificación del problema principal**

* La empresa ocupa un sistema monolítico que escasea de personalización.
* Sistema con falta de escalabilidad.
* Dificultad al implementar políticas de comunidad.
* Dificultad de usar tanto para moderadores como para usuarios.

**1.3 Impacto en la empresa**

Impacta negativamente en la empresa ya que limita las acciones que puede tener el usuario dentro de las funciones de la página web haciendo que las empresas rivales tengan ventaja sobre ella

**1.4 Objetivo General**

Superar las limitaciones de su sistema actual y soportar su continuo crecimiento cambiando el sistema a uno de microservicios

**1.5 Objetivos específicos del proyecto**

1. Desarrollar una plataforma web propia ConnectForo SPA, que permite a los usuarios/clientes crear, administrar y moderar sus propios foros, eliminando la dependencia de herramientas externas.
2. Implementar una arquitectura basada en microservicios, organizada por dominios de negocio, para garantizar escalabilidad, modularidad y mantenimiento eficiente del sistema.
3. Diseñar e implementar un sistema de gestión de usuarios y roles, permitiendo la administración de permisos para administradores, moderadores y usuarios finales.
4. Desarrollar un módulo de moderación de contenido que incluya herramientas para la revisión, aprobación y eliminación de publicaciones que infrinjan las normas de la comunidad.
5. Crear un sistema de notificaciones y anuncios institucionales que permita la publicación de avisos segmentados por categorías o perfil de usuario.
6. Integrar herramientas de seguridad y autenticación incluyendo gestión de contraseñas, autenticación multifactor (MFA) y protección contra spam o ataques de bots.
7. Optimizar la experiencia de usuario mediante un diseño intuitivo y responsivo garantizando accesibilidad desde distintos dispositivos y navegadores.
8. Implementar un sistema de reportes y métricas que permita monitorear la actividad de los foros, detectar tendencias y evaluar la efectividad de moderación.
9. Asegurar la disponibilidad y estabilidad de la plataforma mediante el uso de bases de datos optimizadas, backups periódicos y monitoreo de rendimiento.
10. Realizar pruebas y validaciones del sistema para garantizar que la plataforma cumpla con los requisitos funcionales y no funcionales antes de su lanzamiento.

### **2. Estrategia de Microservicios**

#### **2.1 Tipo de Estrategia Usada: Por Dominio**

La estrategia elegida para la arquitectura de microservicios es la **estrategia por dominio**. Esta estrategia implica la organización de microservicios de acuerdo a los distintos dominios de negocio que conforman la plataforma de ConnectForo SPA. Cada microservicio se diseñaría para manejar un aspecto específico del negocio, como la gestión de usuarios, la moderación de contenido, las categorías de foros, las interacciones sociales, entre otros.

#### **2.2 Justificación Técnica y de Diseño de la Estrategia Elegida**

1. **Alineación con el Negocio**: La organización por dominios permite que cada microservicio esté alineado con las funcionalidades específicas del negocio. Esto asegura que el desarrollo y las actualizaciones sean más coherentes con las necesidades del mismo, facilitando la implementación de nuevas funcionalidades de manera ágil.
2. **Escalabilidad**: Cada microservicio puede ser escalado de manera independiente en función de la demanda. Por ejemplo, si la moderación de contenido requiere más recursos debido a un aumento en la actividad de foros, se puede escalar ese servicio sin necesidad de afectar a otros componentes del sistema.
3. **Desarrollo Independiente**: Los equipos pueden trabajar en microservicios diferentes de manera simultánea y autónoma, lo que acelera el proceso de desarrollo. Esto también permite a los equipos adoptar tecnologías y herramientas que mejor se adapten a cada dominio sin imponer una uniformidad innecesaria en toda la aplicación.
4. **Mantenimiento y Despliegue Simplificado**: Al tener microservicios claramente definidos, el mantenimiento y despliegue se vuelven más manejables. Las actualizaciones pueden realizarse en un servicio específico sin necesidad de realizar despliegues masivos que podrían introducir errores en otras áreas del sistema.

#### **2.3 Beneficios de Adoptar esta Estrategia en Comparación al Sistema Actual**

1. **Flexibilidad y Adaptabilidad**: A diferencia del sistema actual, que utiliza herramientas externas y carece de personalización, la estrategia por dominio permite a ConnectForo SPA adaptar su plataforma a las necesidades cambiantes del mercado de manera más efectiva.
2. **Mejor Manejabilidad de Fallos**: La arquitectura de microservicios por dominio reduce el riesgo de que un fallo en un servicio afecte a todo el sistema. Esto significa que si ocurre un problema en el módulo de moderación, los foros pueden seguir operativos mientras se gestiona la incidencia.
3. **Optimización de la Experiencia del Usuario**: Al proporcionar un sistema más eficiente, que responde mejor al manejo de la carga y a las interacciones, se puede ofrecer una experiencia más fluida para moderadores y usuarios finales, lo cual es crítico en la construcción de comunidades digitales activas.
4. **Capacidad de Innovación**: La adopción de microservicios por dominio permite experimentar e innovar en diferentes áreas del negocio sin afectar la estabilidad del resto de la plataforma. Esto puede traducirse en implementar nuevas funcionalidades o mejoras basadas en feedback de los usuarios con mayor rapidez.

**3. Herramientas Utilizadas**

**3.1 Listado de herramientas**

1. **Lenguaje de Programación: Java**

**Funcionalidad**: Java es un lenguaje de programación orientado a objetos que permite crear aplicaciones robustas y escalables. Es ampliamente utilizado en aplicaciones empresariales y tiene un rico ecosistema de bibliotecas y herramientas que facilitan el desarrollo.

**Justificación**: Se elige Java por su estabilidad, rendimiento y robustez. Además, su amplia adopción en el sector empresarial y la gran comunidad de soporte disponible facilitan el desarrollo y la resolución de problemas.

1. **Framework: Spring Boot**

**Funcionalidad**: Spring Boot es un framework para crear aplicaciones Java que simplifica la configuración y la implementación. Permite desarrollar aplicaciones de microservicios de manera rápida y eficiente, proporcionando características como configuraciones automáticas, gestión de dependencias y una amplia variedad de módulos para diferentes funcionalidades (seguridad, acceso a datos, etc.).

**Justificación**: Se elige Spring Boot por su capacidad para facilitar el desarrollo de microservicios. Su modularidad, velocidad y facilidad de integración con servicios RESTful son esenciales para la arquitectura basada en microservicios de ConnectForo SPA.

1. **Base de Datos: MySQL**

**Funcionalidad**: MySQL es un sistema de gestión de bases de datos relacional que ofrece un alto rendimiento, escalabilidad y flexibilidad. Es adecuado para una variedad de aplicaciones, y su soporte para transacciones y consultas complejas lo hace muy eficiente.

**Justificación**: La elección de MySQL se justifica por su madurez, estabilidad y facilidad de uso. Es uno de los sistemas de gestión de bases de datos más populares y cuenta con amplia documentación y soporte. Además, su capacidad para manejar grandes volúmenes de datos y su compatibilidad con diversas herramientas de administración de bases de datos contribuyen a una gestión eficaz de la información.

1. **Plataforma en la Nube: AWS (Amazon Web Services)**

**Funcionalidad**: AWS ofrece una amplia gama de servicios en la nube, incluyendo computación, almacenamiento, bases de datos y herramientas de integración. Se puede usar para desplegar aplicaciones en la nube, manejar la escalabilidad y almacenar datos de manera confiable.

**Justificación**: La elección de AWS se basa en su escalabilidad, seguridad y la diversidad de servicios que facilitan el desarrollo y la implementación de aplicaciones. Su infraestructura global mejora los tiempos de respuesta y el acceso de usuarios en diversas ubicaciones geográficas.

**4. Herramientas de Trabajo Colaborativo.**

Las Herramientas que ocuparemos para llevar a cabo nuestro objetivo/proyecto son las siguientes

* **Trello** Para organizar tareas de cada integrantes de equipo

(<https://trello.com/invite/b/67e06e62820792a02a6b981c/ATTId1be9f7c97cd223b3fa50e89f6f2a171EB6FC0D3/evaluacion-fullstack>)

* **Github** para Guardar Avances

(<https://github.com/francisco1354/ConnectForo-SPA>)

**5. Enfoque Ético en el Desarrollo**

**5.1 Evaluación de los Principales Desafíos Éticos**

1. **Privacidad y seguridad de datos:**

ConnectForo SPA debe garantizar la protección de los datos personales de sus usuarios, cumpliendo con normativas locales como la Ley de Protección de Datos en Chile. Esto incluye medidas de encriptación, almacenamiento seguro, anonimización y políticas claras de privacidad.

1. **Seguridad:**

Implementar mecanismos de autenticación segura (MFA), monitoreo constante de actividades sospechosas y protocolos de seguridad para proteger la plataforma contra ataques cibernéticos y accesos no autorizados.

1. **Responsabilidad en el despliegue:**

Asegurar un desarrollo responsable y transparente, donde los usuarios estén informados sobre cómo se usan sus datos y se respeten sus derechos, Además, realizar auditorías periódicas para evaluar el cumplimiento normativo.

1. **Impacto en los puestos de trabajo:**

La automatización y digitalización pueden afectar ciertos roles tradicionales. ConnectForo SPA debe ofrecer oportunidades de capacitación y reubicación para empleados afectados por la transformación digital.

**5.2 Soluciones y buenas prácticas adoptadas.**

Utilizar marcos éticos para el desarrollo de software, como el diseño centrado en el usuario y la toma de decisiones basada en la equidad y transparencia.

**5.3 Consideraciones de cumplimiento normativo.**

Asegurar el cumplimiento de la Ley 19.628 sobre Protección de la Vida Privada en Chile, que regula el tratamiento de datos personales y protege los derechos de los titulares.

**6. Análisis de Requerimientos**

**6.1 y 6.2 Requisitos Funcionales y No Funcionales**

* [**Requerimientos Funcional y No Funcional**](https://docs.google.com/spreadsheets/d/16-h_M7bd2HyBHRYlTCeOTWALVxiPh_-5Rx5XVfah1VA/edit?usp=sharing)

**6.3 Síntesis de Necesidades de Cada Perfil**

**Administrador de Sistemas**

* Necesita Gestionar usuarios, foros y configuraciones de seguridad, garantizando que el sistema sea accesible, seguro desde múltiples dispositivos.

**Moderador de foro**

* Requiere Herramientas para moderar el contenido y mantenerlo ordenado

**Cliente**

* requiere una experiencia fluida, segura y accesible en la plataforma

con un rendimiento bueno para participar en el foro

**Soporte Tecnico**

* Necesita asegurar la operatividad del sistema monitoreando el rendimiento, seguridad y disponibilidad para mantenerlo funcionando siendo un sistema escalable

**Gestor de anuncios**

* Necesita la facilidad de poder crear y gestionar anuncios en los foros

**7. Análisis del Sistema Actual.**

**7.1 Arquitectura General del Sistema Monolítico**

El sistema actual de ConnectForo SPA opera bajo un enfoque monolítico, lo que significa que todas las funcionalidades están integradas en una única aplicación. Este tipo de arquitectura puede presentar problemas de escalabilidad y flexibilidad, especialmente al crecer en volumen de usuarios o al requerir nuevas funcionalidades.

**7.2 Puntos Débiles Detectados**

* **Fallas Y Vulnerabilidades:** La dependencia de herramientas externas y la falta de una plataforma propia aumentan el riesgo de errores y caídas del sistema.
* **Cuellos de Botella:** La arquitectura monolítica dificulta el manejo eficiente de grandes volúmenes de datos y usuarios simultáneos, lo que impacta el rendimiento durante picos de tráficos.
* **Mantenimiento y Actualización:** Cualquier cambio o actualización implica modificar todo el sistema, lo que retrasa el desarrollo y aumenta el riesgo de introducir errores.
* **Escalabilidad Limitada:** La expansión de la plataforma se ve restringida, pues el crecimiento exponencial de usuarios o nuevas funcionalidades requiere un rediseño complejo.

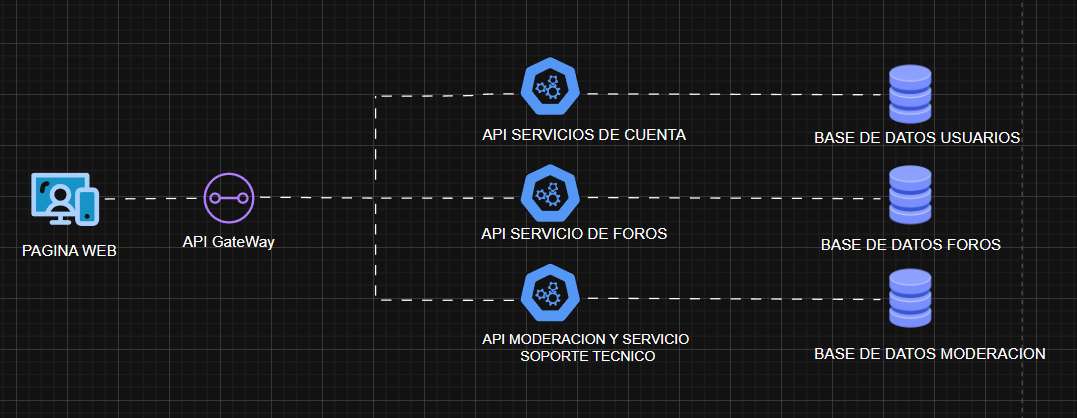
Estas limitaciones resaltan la necesidad de una solución más moderna y escalable, como una arquitectura de microservicios.

**8. DISEÑO DE LA NUEVA ARQUITECTURA**

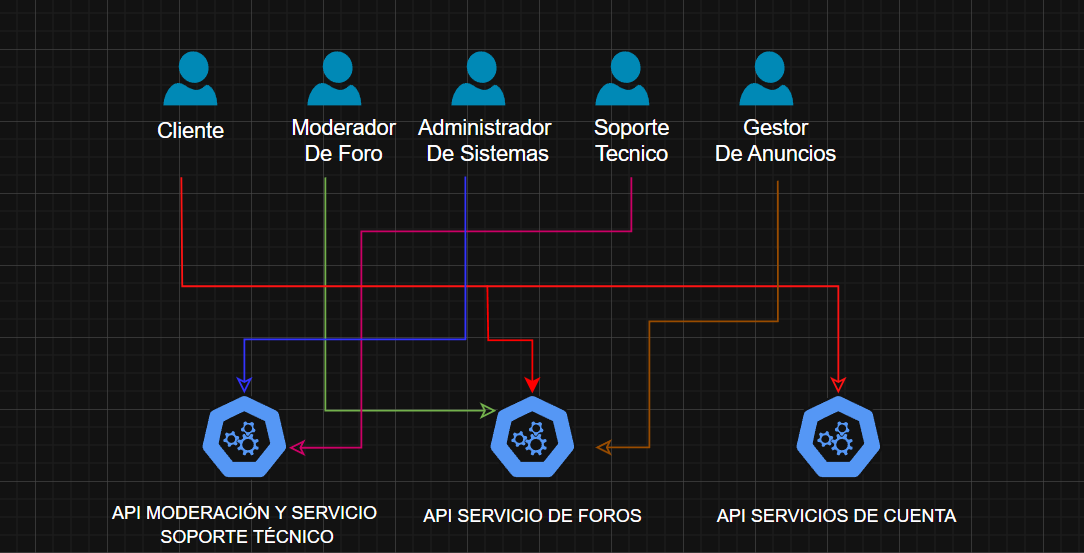
**8.1 Descripción general de la arquitectura basada en microservicios**

La nueva arquitectura está basada en microservicios que permiten dividir el sistema en componentes independientes. Cada componente es responsable de una funcionalidad específica (usuarios, foros, mensajes, etc.).

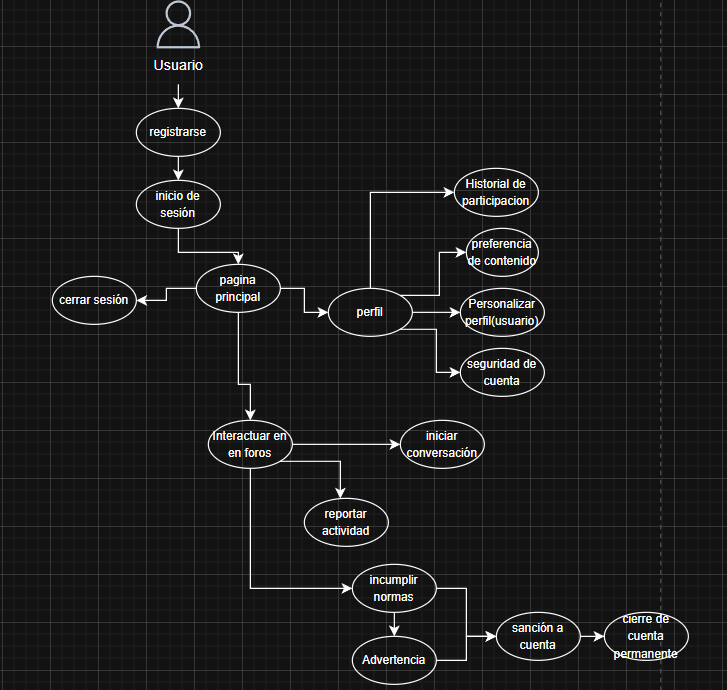
**8.2 Diagramas**



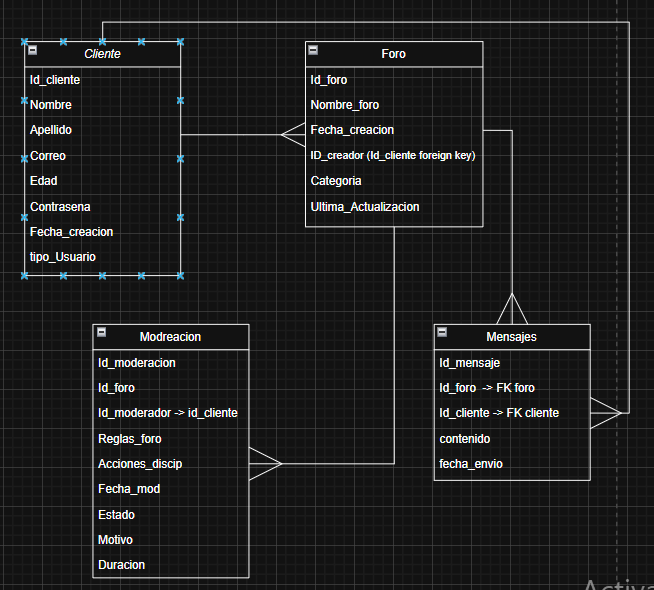
**8.2 Diagrama de Actores de Alto Nivel**



**Diagrama caso uso**



**Diagrama de clases**



### **9. Planificación de la Migración**

### **9.1 Fases de migración (por módulo o servicio)**

#### **Fase 1: Modulo De Usuarios**

Se migrará el módulo de autenticación y gestión de usuarios a un servicio independiente.

**Acciones:**

* Integración con el sistema monolítico actual para garantizar la compatibilidad y realizar las pruebas necesarias.
* Asegurarse de que las funcionalidades de autenticación, gestión de contraseñas y roles de usuarios funcionen correctamente.
* Verificar que el sistema siga en funcionamiento antes de pasar a la siguiente fase.

**Pruebas:** Se realizará un conjunto de pruebas unitarias y de integración para asegurar que el módulo de usuarios esté completamente funcional.

#### **Fase 2: Módulo de Foros**

Se extraerá la lógica de creación y gestión de foros, convirtiéndola en un microservicio independiente.

**Acciones:**

* Implementación del microservicio de foros que incluirá funcionalidades de creación, edición, y eliminación de foros.
* Conexión con el módulo de usuarios para permitir la gestión de los creadores de foros.
* Se probará la integración con el módulo de usuarios para asegurar el flujo correcto de la plataforma.

**Pruebas:** Asegurarse de que la migración no genere problemas de rendimiento ni errores en la creación y administración de foros.

#### **Fase 3: Módulo de Mensajes**

Se separará la funcionalidad de publicación y consulta de mensajes a un nuevo microservicio.

**Acciones:**

* Implementación del microservicio de mensajes, con la capacidad de publicar, editar y consultar mensajes.
* Conexión con el módulo de foros y usuarios para asegurar que los mensajes estén correctamente relacionados con sus respectivos foros y autores.

**Pruebas:** Se realizarán pruebas exhaustivas para verificar que la comunicación entre el microservicio de mensajes y los otros módulos sea fluida y sin errores.

#### **Fase 4: Módulo de Moderación**

Se trasladará la lógica de moderación a un servicio independiente.

**Acciones:**

* Implementación de un microservicio dedicado a la moderación, que incluya la gestión de reglas, sanciones y filtrado de contenido.
* Verificar que los moderadores puedan gestionar correctamente los foros y aplicar las sanciones a los infractores.

**Pruebas:** Realización de pruebas de usabilidad y funcionalidad para asegurarse de que la moderación se lleva a cabo correctamente en el sistema.

#### **Fase 5: Desmontar el sistema monolítico**

Una vez todos los módulos se hayan migrado y verificado su correcto funcionamiento, se procederá a eliminar el sistema monolítico.

**Acciones:**

* Eliminar el código del sistema monolítico que ya no es necesario.
* Realizar pruebas finales de todo el sistema de microservicios para garantizar que todas las interacciones y comunicaciones entre los módulos sean efectivas y sin problemas.
* Optimización de las conexiones entre microservicios para garantizar la eficiencia y escalabilidad del sistema.

#### **9.2 Consideraciones de Compatibilidad**

* **Interoperabilidad:** Durante la migración, se asegurará que los microservicios sean compatibles con el sistema monolítico existente hasta que todos los módulos hayan sido completamente migrados. Se utilizarán interfaces de programación de aplicaciones (APIs) para garantizar la comunicación entre los servicios nuevos y el sistema antiguo durante la transición.
* **Datos y Migración:** Se llevará a cabo una migración de datos a medida que los módulos se van separando, garantizando que toda la información de usuarios, foros, mensajes y moderación se mantenga consistente y segura. La migración de bases de datos debe realizarse con cuidado para evitar pérdida de datos.
* **Integración de servicios:** Se adoptarán enfoques como contenedores Docker y orquestación con Kubernetes para facilitar la integración entre los microservicios, evitando dependencias entre los módulos y asegurando su fácil despliegue.

#### **9.3 Identificación de Riesgos y su Plan de Mitigación**

1. **Riesgo: Pérdida de Datos durante la Migración**
   * **Mitigación:** Realizar copias de seguridad completas de la base de datos antes de cada fase de migración. Utilizar migraciones controladas y graduales para minimizar los riesgos de pérdida de información.
2. **Riesgo: Incompatibilidad entre el sistema monolítico y los nuevos microservicios**
   * **Mitigación:** Implementar una fase de pruebas de integración para validar que las APIs y microservicios funcionan correctamente con el sistema monolítico, antes de migrar completamente. Además, usar un enfoque gradual para que las transiciones sean menos abruptas.
3. **Riesgo: Problemas de escalabilidad**
   * **Mitigación:** Asegurarse de que los microservicios estén diseñados con escalabilidad en mente. Realizar pruebas de carga para detectar posibles cuellos de botella en el rendimiento de los microservicios antes de su despliegue en producción.
4. **Riesgo: Fallos en la comunicación entre los microservicios**
   * **Mitigación:** Adoptar tecnologías de mensajería como Kafka o RabbitMQ para asegurar que la comunicación entre microservicios sea eficiente y sin pérdidas de mensajes. Establecer sistemas de monitoreo para detectar fallos de comunicación rápidamente.
5. **Riesgo: Resistencia al cambio por parte de los usuarios internos (empleados y administradores)**
   * **Mitigación:** Proporcionar formación continua y documentación adecuada para los usuarios y administradores del sistema. Además, implementar fases de pruebas piloto y retroalimentación para mejorar la transición.

### **10. Planificación Ágil y Cronograma**

#### **10.1 Enfoque ágil usado**

* **Scrum:** Enfoque ágil que permitirá un desarrollo iterativo y la entrega de valor en fases

#### **10.2 Roles asumidos por los integrantes**

* **Francisco Gómez:** Líder de equipo y encargado del diseño de la arquitectura.
* **Ricardo Díaz:** Desarrollador principal y encargado de la base de datos.
* **Joan Rojas:** Coordinador de pruebas y responsable de la integración.

#### **10.3 Carta Gantt**

* [CARTA GANTT](https://docs.google.com/spreadsheets/d/1bWtM9fCiG3onetHeRdCULc6zN07pJ44gWDx2lF29weI/edit?usp=sharing)

### **11. Conclusiones y Recomendaciones**

#### **11.1 Lecciones aprendidas**

* Importancia de planificar cuidadosamente la migración y probar cada componente.

#### **11.2 Potencial escalabilidad del sistema**

* El sistema de microservicios permitirá escalar de forma eficiente a medida que crecen los usuarios y funcionalidades.

#### **11.3 Posibles mejoras futuras**

* Mejorar la inteligencia artificial para la moderación automática.
* Implementar un sistema de analítica más avanzado para medir la interacción de los usuarios.

### **Conclusión**

La migración del sistema monolítico a microservicios mejora significativamente la escalabilidad, flexibilidad y mantenimiento de la plataforma de ConnectForo SPA. La adopción de una arquitectura por dominio, junto con herramientas como Spring Boot, MySQL y AWS, permite una mayor modularidad y eficiencia. Este enfoque no solo optimiza el rendimiento y la experiencia del usuario, sino que también facilita la integración de nuevas funcionalidades de manera ágil y segura. La implementación de buenas prácticas éticas y el enfoque ágil en el desarrollo contribuyen a minimizar riesgos y asegurar el éxito a largo plazo del proyecto. Con esta migración, ConnectForo SPA está mejor posicionada para seguir creciendo y adaptándose a las necesidades del mercado.