Adrian Ramirez Gonzalez
Adrian Alexander Benavides
Andrés Hernando Borda Muñoz
Brayan Alejandro Muñoz Pérez

Tarea 3 - Priorización y estimación de requerimientos

Priorización MOSCOW

- **Must have**
    a. Registro e inicio de sesión
    b. Lista de contactos dinámica
    c. Chat grupal
    d. Seguridad
    e. Disponibilidad
    f. Mantenibilidad
- **Should have**
    a. Chat individual
    b. Rendimiento
    c. Búsqueda de mensajes y contactos
    d. Notificaciones en tiempo real
- **Could have**
    a. Historial de mensajes
- **Won't have**
    a. Integración con calendario académico

Estimación de los requerimiento usando planning poker
Limitaciones:

- **Backend:** django
- **Frontend:** Vue
- **Base de datos:** Mongodb
- Cada punto equivale a 1 día

Justificaciones de las estimaciones

Adrian Ramirez Gonzalez

- **Registro e inicio de sesión**

  Estimación: 5

  Justificación: La creación de un módulo de registro e inicio es un proceso bastante sencillo el cual no debería durar mucho en realizarse, pero es necesario tomar un tiempo prudente para construirlo con el fin de asegurar la calidad de este.

- **Lista de contactos dinámica**

  Estimación: 3

Justificación: La creación de una lista que permita añadir nuevos contactos con su información es un proceso bastante simple el cual involucra funcionalidades básicas, por lo cual estimo que este requerimiento puede ser fácilmente construido en 3 días.

- **Chat individual**

  Estimación: 13

  Justificación: La implementación de un chat puede ser bastante complicada debido principalmente a la poca experiencia con las tecnologías definidas. Es por esto, que considero 13 días como un tiempo prudente para poder profundizar en las tecnologías mencionadas para posteriormente realizar la implementación de este requerimiento.

- **Chat grupal**

  Estimación: 13

  Justificación: Al igual que para la construcción del chat individual considero que el problema en este caso también está asociado a la inexperiencia con las tecnologías que se van a usar, por lo cual creo que 13 días también es un tiempo adecuado para poder satisfacer este requerimiento.

- **Historial de mensajes**

  Estimación: 8

  Justificación: El historial de mensajes es un requerimiento muy relacionado con el chat individual y grupal, por lo cual estimo que el tiempo de implementación de este requerimiento será un solo poco más pequeño que el tiempo estimado para el chat grupal e individual ya que es una funcionalidad más sencilla, pero al mismo tiempo sigue siendo un reto debido a la inexperiencia.

- **Notificaciones en tiempo real**

  Estimación: 5

  Justificación: Este conjunto de funcionalidades (notificaciones en tiempo real y silenciar notificaciones de chats) no parece tener un grado de complejidad muy alto ya que tienen que ver con algunos aspectos básicos de los chats, por lo cual creo que 5 días es suficiente para poder realizar la implementación.

- **Búsqueda de mensajes y contactos**

  Estimación: 1

  Justificación: Estimo que este requerimiento se puede realizar en un día ya que este no depende tanto de las tecnologías utilizadas, sino de la estrategias y algoritmos usados para realizar estas búsquedas de conversaciones y usuarios de manera eficiente.

- **Integración con calendario académico**

  Estimación: 21

  Justificación: Estimo que 21 días es suficiente para poder realizar esta implementación debido principalmente a que en este caso se debe conocer de manera un poco más profunda las tecnologías usadas para poder implementar todo un sistema de recordatorios de clase que se vaya actualizando constantemente, por lo cual será necesario realizar una investigación profunda de las tecnologías para poder cumplir con este requerimiento, obligando a tomar un poco más de tiempo de lo normal.

- **Seguridad**

  Estimación: 21

  Justificación: La seguridad es una parte fundamental de la aplicación ya que se estará tratando información de una comunidad académica, sin embargo, debido a la falta de experiencia con la implementación de mecanismos de seguridad, creo que la realización de este requerimiento tomará bastante tiempo ya que habrá que investigar la mejor forma de implementar mecanismos como el cifrado de mensajes y otros sistemas de seguridad para posteriormente realizar pruebas exhaustivas de estos mecanismos para asegurar su calidad.

- **Rendimiento**

  Estimación: 5

  Justificación: Se esperaría que las tecnologías usadas tuvieran un gran rendimiento gracias a todo el trabajo que hay detrás de estas, por lo cual creo que en 5 días se podría asegurar un buen rendimiento del software por medio la implementación de las buenas prácticas y recomendaciones asociadas a las tecnologías que se van a usar.

- **Mantenibilidad**

  Estimación: 8

  Justificación: La cantidad de funcionalidades y características de este software no son muy grandes, por lo cual creo que 8 días es suficiente para poder construir el código de manera modular y realizar una correcta documentación de todo el software.

- **Disponibilidad**

  Estimación: 8

Justificación: Para garantizar este requerimiento será necesario hacer un análisis profundo de todo el código con el fin de buscar todos los posibles errores que podría ocurrir dentro del software los cuales lleven a problemas en la operatividad de este. Además, es necesario pensar y evaluar todo tipo de escenarios que podrían afectar la operatividad del software con el fin de poder plantear medidas que solucionen estos eventos. En conclusión, teniendo en cuenta que este software no es tan grande en términos de funcionalidades, estimo que 8 días es suficiente para poder hacer todo este tipo de análisis y planteamiento de medidas las cuales permitirán maximizar el tiempo que el software se encuentra operativo.

Adrian Alexander Benavides

- **Registro e inicio de sesión**

  Estimación: 5

  Justificación: Este módulo incluye el desarrollo de interfaces para el registro e inicio de sesión, así como la integración con un sistema de autenticación. Se deben implementar medidas de seguridad como encriptación de contraseñas (por ejemplo, usando bcrypt) y validación de entradas para evitar ataques como inyecciones SQL o fuerza bruta. También puede ser necesario integrar autenticación de terceros (como Google o Microsoft) para simplificar el acceso, lo que añade un poco de complejidad. El tiempo estimado considera pruebas básicas de funcionalidad y correcciones de errores menores.

- **Lista de contactos dinámica**

  Estimación: 8

  Justificación: La lista de contactos debe ser capaz de actualizarse en tiempo real cuando un usuario agrega, elimina o modifica contactos. Esto requiere integrar tecnologías como WebSockets para lograr sincronización en tiempo real, así como diseñar una base de datos eficiente que permita consultas rápidas sobre contactos relacionados. Además, es fundamental crear una interfaz de usuario intuitiva que permita búsquedas y filtros por nombre o rol (estudiante/profesor), lo que añade complejidad. El tiempo estimado incluye pruebas de rendimiento para asegurar la fluidez en el uso.

- **Chat individual**

  Estimación: 13

  Justificación: La implementación de un chat individual incluye establecer comunicación en tiempo real entre dos usuarios, utilizando tecnologías como WebSockets o servicios como Firebase. Además, se debe considerar la persistencia de los mensajes en la base de datos, garantizando que los mensajes enviados y recibidos estén disponibles incluso después de cerrar sesión. Esto también requiere lógica para manejar errores, como la reconexión automática en caso de pérdida de

conexión. Otro aspecto clave es la encriptación de mensajes para garantizar la privacidad. El tiempo estimado incluye pruebas exhaustivas para asegurar la estabilidad del sistema.

- **Chat grupal**

  Estimación: 15

  Justificación: Un chat grupal es significativamente más complejo que un chat individual, ya que requiere la creación de lógicas para la gestión de grupos, como agregar y eliminar participantes, asignar roles (administrador, miembro), y manejar notificaciones grupales. También es necesario implementar una estructura de mensajes que diferencie a los emisores y permita visualizar mensajes históricos de manera organizada. La sincronización en tiempo real debe ser robusta para evitar inconsistencias entre los usuarios. Este módulo también incluye pruebas extensas para identificar y corregir problemas relacionados con la escalabilidad y el manejo de grandes volúmenes de usuarios.

- **Historial de mensajes**

  Estimación: 8

  Justificación: El historial de mensajes implica desarrollar una funcionalidad que permita a los usuarios cargar mensajes antiguos de manera eficiente, sin afectar el rendimiento general del sistema. Esto requiere optimizar las consultas a la base de datos, utilizando paginación o técnicas de carga progresiva. Además, es fundamental que el historial sea accesible tanto para chats individuales como grupales, lo que añade complejidad. También se deben considerar aspectos de seguridad, como la protección de datos históricos frente a accesos no autorizados. El tiempo estimado incluye optimizaciones y pruebas para asegurar una experiencia de usuario fluida.

- **Notificaciones en tiempo real**

  Estimación: 13

  Justificación: Las notificaciones en tiempo real son esenciales para alertar a los usuarios sobre nuevos mensajes, solicitudes de contacto o cambios en eventos del calendario. Esto requiere integrar un sistema de notificaciones push y diseñar una lógica backend que envíe alertas según las actividades realizadas en la plataforma. También es necesario desarrollar una interfaz que permita a los usuarios configurar sus preferencias de notificación. El tiempo estimado considera la implementación de estas funcionalidades y pruebas para asegurar que las notificaciones sean precisas y no generen spam.

- **Búsqueda de mensajes y contactos**

  Estimación: 5

Justificación: La búsqueda eficiente en la plataforma requiere indexar los datos de mensajes y contactos, posiblemente utilizando un motor de búsqueda como Elasticsearch o una solución basada en consultas SQL optimizadas. Además, la interfaz de búsqueda debe proporcionar sugerencias en tiempo real y permitir el uso de filtros avanzados. Este módulo también necesita manejar posibles errores, como resultados vacíos, y asegurar que los datos sean presentados de manera intuitiva. Las pruebas de usuario son clave para garantizar que la funcionalidad sea rápida y efectiva.

- **Integración con calendario académico**

  Estimación: 13

  Justificación: La integración con un calendario académico requiere sincronizar eventos y horarios con los sistemas existentes de la institución, lo que puede implicar el uso de APIs externas. También es necesario desarrollar interfaces que permitan a los usuarios visualizar y gestionar eventos directamente desde la app. La funcionalidad debe incluir notificaciones sobre eventos importantes y la posibilidad de crear recordatorios personalizados. Este módulo es más complejo si se considera la necesidad de gestionar conflictos de horarios y la sincronización en tiempo real, lo que justifica el tiempo estimado.

- **Seguridad**

  Estimación: 13

  Justificación: La seguridad es un componente crítico, especialmente en una plataforma académica donde se manejan datos sensibles de usuarios. Esto incluye implementar encriptación de extremo a extremo para mensajes, asegurar conexiones HTTPS y desarrollar mecanismos para prevenir accesos no autorizados. También se deben realizar pruebas de penetración para identificar vulnerabilidades y corregirlas. Además, es necesario cumplir con normativas de protección de datos aplicable, lo que añade un nivel adicional de complejidad.

- **Rendimiento**

  Estimación: 8

  Justificación: Optimizar el rendimiento implica identificar y eliminar cuellos de botella en el sistema, como consultas lentas a la base de datos o tiempos de carga elevados en la interfaz de usuario. Esto incluye realizar pruebas de carga para medir el comportamiento del sistema bajo condiciones de alto tráfico y ajustar la arquitectura en consecuencia. Además, se deben implementar estrategias de cacheo para reducir la dependencia de consultas frecuentes a la base de datos. Este tiempo también incluye ajustes en el frontend para mejorar la experiencia del usuario.

- **Mantenibilidad**

  Estimación: 5

Justificación: Asegurar la mantenibilidad del sistema requiere estructurar el código de manera modular, documentar todas las funcionalidades y establecer convenciones de desarrollo claras para el equipo. También es necesario crear pruebas automatizadas que permitan verificar rápidamente la estabilidad del sistema después de realizar cambios. Estas prácticas no solo reducen el tiempo requerido para futuras actualizaciones, sino que también facilitan la incorporación de nuevos desarrolladores al proyecto.

- **Disponibilidad**

  Estimación: 8

  Justificación: La alta disponibilidad implica implementar estrategias de redundancia, como la configuración de múltiples servidores o el uso de servicios en la nube que garanticen uptime continuo. También incluye configurar monitoreo en tiempo real para detectar y solucionar problemas antes de que afecten a los usuarios. Es importante establecer un plan de recuperación ante desastres (DRP) para minimizar el impacto de fallas críticas. Este tiempo incluye pruebas para validar la resistencia del sistema bajo diversas condiciones.

Andrés Hernando Borda Muñoz

- **Registro e inicio de sesión**

  Estimación: 5

  Justificación: Dado que la esencia del proyecto es crear código con toda la documentación requerida es pertinente tomar 5 días para la elaboración del “log in”, con ello se garantiza una buena elección de las tecnologías a implementar y un a posible mejora o mantenimiento en el futuro.

- **Lista de contactos dinámica**

  Estimación: 5

  Justificación: La complejidad de esta lista es la integración con la base de datos seleccionada, por ende es pertinente tomarse un par de días para instruirse en el proveedor de base de datos y sus herramientas. Testear posibles anomalías e ir identificando estilos para el despliegue de esta lista.

- **Chat individual**

  Estimación: 13

  Justificación: La complejidad de este ítem radica en la autenticación y seguridad de los datos compartidos a través del chat, luego se necesita probar su encriptación. Por otro lado, la sincronización en tiempo real es un ítem que puede demorar tiempo al implementar websockets.

- **Chat grupal**

Estimación: 21

Justificación: Dado que el requerimiento del cliente es generar una comunidad académica este requerimiento es importantísimo, luego se debe tomar un tiempo en generar los roles propios de la app (admin, estudiante, profesor), además la escalabilidad debe tenerse en cuenta en un futuro, ya que no es lo mismo dar soporte a una comunidad de 100 personas que a una de 7000 personas.

Por último, la revisión de la sincronización para todos los miembros es complejo y escribir la documentación de todo esto es una tarea ardua por ello es pertinente abarcar los 21 días.

- **Historial de mensajes**

  Estimación: 2

  Justificación: Este ítem debió ser cubierto en los dos anteriores requerimientos, por ende en este punto se debe preocupar nuestro equipo de desarrollo por testear posibles fallas de ahí el poco tiempo requerido.

- **Notificaciones en tiempo real**

  Estimación: 5

  Justificación: Aquí es fundamental el encontrar servicios adecuados que le proporcionen a nuestros clientes baja latencia (entrega inmediata de notificaciones) y un manejo para miles de usuarios, luego la infraestructura es clave. También se debe tener en cuenta el rendimiento de las baterías de los dispositivos a los que se entregan las notificaciones.

- **Búsqueda de mensajes y contactos**

  Estimación: 1

  Justificación: Este tiempo es reducido, ya que en los anteriores requerimientos debió abordarse y completado a cabalidad.

- **Integración con calendario académico**

  Estimación: 21

  Justificación: Esta tarea es compleja, ya que se necesita conocer las tecnologías propias del calendario académico y las posibles herramientas que permitan una sincronización con nuestro proyecto, ello implica un estudio minucioso y autodidacta para garantizar que esta implementación mantendrá nuestros servicios disponibles y no generará errores.

- **Seguridad**

  Estimación: 13

Justificación: Es imperativo generar una ventana de tiempo prudente que permita generar un escenario de ataque a nuestro desarrollo, con ello identificar vulnerabilidades y proceder a robustecer nuestro sistema, ya que el manejo de datos personales debe garantizarse con la mayor seguridad posible.

- **Rendimiento**

  Estimación: 8

  Justificación: La optimización del desarrollo realizado debe ser una etapa de nuestro proyecto donde aquello funcional mejore sus tiempos de respuesta sin caer el servicio, luego se deben identificar ciertos requerimientos críticos para la experiencia del usuario a los cuales focalizar esfuerzos. Documentar esta parte puede requerir reescribir código ya implementado y ello toma un tiempo.

- **Mantenibilidad**

  Estimación: 13

  Justificación: En este punto siendo un grupo de desarrolladores ético debemos pensar en generar una app sostenible en el tiempo sin la intervención directa de sus “creadores”, luego en base al presupuesto del proyecto se debe generar ciertas expectativas de tráfico y almacenamiento de información sin que la app fallé. Otra cosa diferente es si en el contrato se ha estipulado una revisión periódica,luego se deben definir puntos críticos a inspeccionar.

- **Disponibilidad**

  Estimación: 8

  Justificación: Todo el desarrollo se enfoca en dar funcionalidad al proyecto para todos los usuarios, luego la elección de la infraestructura es crucial para garantizar la disponibilidad de los servicios, luego se debe documentar al detalle los pros y contras de las elecciones realizadas, así como garantizar la disponibilidad de back ups or rescues que garanticen la operación de nuestra app.

Brayan Alejandro Muñoz Pérez

- **Registro e inicio de sesión**

  Estimación: 5

  Justificación: Yo considero que pese a que dos días es suficiente para hacer un login, los primeros tres días se pueden aprovechar para documentarse un poco de las tecnologías que se van a usar, también es de esperar que el login tenga una buena encriptación de contraseñas y pues que la UI y UX sea agradable.

- **Lista de contactos dinámica**

  Estimación: 5

Justificación: Solo consiste en hacer un CRUD sencillo, el hecho de programarlo no es complicado, bajo mi punto de vista desplegarlo se hace en una o dos tardes, la complejidad está en tener en cuenta ya un buen diseño de la base de datos para que, una lista de contactos que en realidad es una tabla sencilla, no ocasione problemas en el futuro por simplemente no haber tenido en cuenta su relación con otras entidades.

- **Chat individual**

  Estimación: 13

  Justificación: Pese a que tiene cierto nivel de complejidad, se partirá de la base del chat grupal, por lo que pienso que es totalmente posible realizarlo en la mitad del módulo base.

- **Chat grupal**

  Estimación: 21

  Justificación: Para mí es el core de la aplicación, la mayoría del esfuerzo debe estar concentrada en este módulo, ya que de alrededor de esta orbitarán las demás características, baso mi estimación en unos 3 días de diseño e investigación, alrededor de 8 días para realizar el backend y paralelamente 13 días para el front (soy fiel creyente de que una buen interfaz de usuario, que además cuente con buena experiencia de usuario debe ser un must para cualquier software, unos trece días para integrar el back y el frond, testear y corregir errores.

- **Historial de mensajes**

  Estimación: 2

  Justificación: Ya para este punto, el historial solo es una consecuencia directa del chat grupal e individual, si el modelo de la base de datos se realizó correctamente, dos días son razonables para realizar este módulo.

- **Notificaciones en tiempo real**

  Estimación: 5

  Justificación: Una vez más lo importante es seleccionar la tecnología y documentarse sobre la misma, integrar este módulo con el backend y ajustar la base de datos para eventos desencadenantes, el frontend no es nada complicado de realizar, solo es asegurar que las notificaciones sean claras y accesibles

- **Búsqueda de mensajes y contactos**

  Estimación: 1

  Justificación: Solo pongo 1 día porque toca en números fibonacci, pero en menos de 1 hora se puede realizar, solo es realizar una consulta en la base de datos, la

demora estaría en el front, diseñar un botón y barra de búsqueda, desde la parte del back simplemente es el C(R)UD.

- **Integración con calendario académico**

  Estimación: 21

  Justificación: Esto ya es más complicado, toca documentarse sobre con cuál calendario se debe implementar, diseñar la arquitectura e identificar los riesgos, por ejemplo el manejo de tokens, sincronización bidireccional, etc. Para implementar el back toca configurar el OAuth, implementar la lógica para leer y escribir los eventos; y lo más importante un webhook para que sea en tiempo real, por ejemplo si una clase se cancela, debe desaparecer del calendario; por último pues el testeo y la optimización

- **Seguridad**

  Estimación: 13

  Justificación: Se debe hacer un buen análisis de requisitos de seguridad, las bases de datos contará con datos sensibles, información personal de profesores y alumnos que bajo ningún concepto pueden fugarse, debido a la inexperiencia es razonable documentarse mucho sobre el tema para evitar la mayor cantidad de riesgos posibles.

- **Rendimiento**

  Estimación: 8

  Justificación: Considero que para asegurar un buen rendimiento se necesitan al menos unos 8 días. Lo principal es optimizar las consultas a la base de datos y la interacción con la API externa para evitar latencias molestas. También toca hacer pruebas de carga para ver cómo responde la app con varios usuarios simultáneos y ajustar lo necesario para mantener la rapidez. No es complicado, pero es muy importante asegurar que todo funcione bien sobre todo en horas pico.

- **Mantenibilidad**

  Estimación: 21

  Justificación: Aquí creo que el tiempo es más largo porque toca hacer todo pensando a futuro. Desde el inicio hay que diseñar un código limpio, bien modularizado y fácil de entender para que cualquier cambio o actualización sea manejable. Además, es vital documentar bien cada parte del proyecto y realizar pruebas unitarias para evitar problemas más adelante. Esto no se ve tan complicado en el corto plazo (y no lo es, todos sabemos que es más tedioso que difícil, y siempre la falta de documentación es por pereza más que otra cosa), pero en el largo plazo, no haberlo hecho puede costar mucho tiempo y recursos, así que vale la pena invertir estos 21 días.

- **Disponibilidad**

  Estimación: 5

  Justificación: Para mí, 5 días son razonables para garantizar que la app esté disponible casi todo el tiempo, incluso si algo falla. Hay que implementar sistemas de redundancia y balanceo de carga para que, si un servidor se cae, la app siga funcionando. Además, toca configurar monitoreo continuo para detectar problemas antes de que afecten a los usuarios. No es tan complicado, pero sí se necesita algo de tiempo para dejar todo bien ajustado.

Consenso final sobre las estimaciones

Requerimientos	Consenso
Registro e inicio de sesión	5
Lista de contactos dinámica	5
Chat individual	13
Chat grupal	21
Historial de mensajes	5
Notificaciones en tiempo real	8
Búsqueda de mensajes y contactos	1
Integración con calendario académico	21
Seguridad	21
Rendimiento	5
Mantenibilidad	13
Disponibilidad	8