Tarea 3

Requerimientos funcionales:

- Chat individual
- Registro e inicio de sesión
- Lista de contactos dinámica: est añadir contactos mediante búsqueda por nombre o matrícula
- Chat grupal
- Historial de mensajes: almacenamiento de mensajes para consultas futuras, con opción de buscar por palabras clave
- Notificaciones en tiempo real
- Búsqueda de mensajes y contactos
- Integración con calendario académico

Requerimientos no funcionales:

- Seguridad: encriptación mensajes
- Rendimiento: velocidad de un servidor
- Disponibilidad: medible en segundos tiempo
- Mantenibilidad: código modular y documentado para facilitar actualizaciones

Método MoSCoW:

MUST: Chat individual, registro e inicio de sesión, historial de mensajes, seguridad, disponibilidad.

SHOULD: Lista de contactos dinámica, chat grupal, rendimiento, mantenibilidad.

COULD: Notificaciones en tiempo real, búsqueda de mensajes y contactos.

WON'T: Integración con calendario académico.

Requerimientos Funcionales

1. **Chat individual**
   - **Tiempo estimado**: 5 días.
   - **Justificación**: Implica la creación de una interfaz de usuario intuitiva y un sistema backend que permita la transmisión de mensajes entre usuarios de manera eficiente.
2. **Registro e inicio de sesión**
   - **Tiempo estimado**: 2 días .
   - **Justificación**: La implementación de un sistema de autenticación es relativamente rápida si ya existe una estructura backend sólida como FireBase.
3. **Lista de contactos dinámica**
   - **Tiempo estimado**: 12 días.
   - **Justificación**: Este requisito implica la creación de una interfaz de usuario donde los estudiantes pueden buscar a otros por nombre o matrícula, lo cual es relativamente sencillo en términos de funcionalidad. Sin embargo, la integración con una base de datos para realizar la búsqueda y la validación de datos puede requerir un poco más de tiempo.
4. **Chat grupal**
   - **Tiempo estimado**: 2-3 semanas.
   - **Justificación**: El chat grupal puede ser un poco más complejo que el chat individual, ya que debe gestionar múltiples usuarios en un solo canal de comunicación. La implementación de funcionalidades como la creación de grupos, gestión de miembros, y la sincronización en tiempo real puede requerir más trabajo en la parte de backend y frontend.
5. **Historial de mensajes**

- **Tiempo estimado**: 7 días.
- **Justificación**: La implementación de un sistema de historial de mensajes implica almacenamiento y recuperación de mensajes de manera eficiente.

6. **Notificaciones en tiempo real**
   - **Tiempo estimado**: 5 días.
   - **Justificación**: Implementar las notificaciones en tiempo real puede ser relativamente rápido si se utilizan servicios como Firebase Cloud Messaging o sistemas de WebSocket. Sin embargo, la integración con los mensajes y la gestión de notificaciones puede aumentar la complejidad.
7. **Búsqueda de mensajes y contactos**
   - **Tiempo estimado**: 1-2 semanas.
   - **Justificación**: Incluir una funcionalidad de búsqueda dentro de la aplicación se puede tornar particularmente compleja si se hace con una base de datos no relacional.
8. **Integración con calendario académico**
   - **Tiempo estimado**: 7 días.
   - **Justificación**: La integración con un calendario académico requiere conectarse a una API o gestionar eventos, lo cual puede ser un poco más complejo dependiendo de la fuente de los datos.

Requerimientos No Funcionales

1. **Seguridad (Encriptación de mensajes)**
   - **Tiempo estimado**: 1-2 semanas.
   - **Justificación**: La encriptación de los mensajes es esencial para proteger la privacidad de los usuarios. Esta es una tarea crítica que requiere pruebas exhaustivas y la implementación de varias tecnologías de encriptación y seguridad.
2. **Rendimiento (Velocidad del servidor)**
   - **Tiempo estimado**: 1 semana (tarea continua).
   - **Justificación**: Este es un requisito clave pero más a nivel de configuración e infraestructura. Puede que no sea un trabajo de desarrollo directo, pero se necesitarán pruebas de carga, optimización de la base de datos y configuración de servidores para garantizar que la aplicación maneje un número elevado de usuarios simultáneos.
3. **Disponibilidad (Tiempo de inactividad medido en segundos)**
   - **Tiempo estimado**: 1 semana (tarea continua).
   - **Justificación**: Asegurar alta disponibilidad generalmente implica la implementación de estrategias de tolerancia a fallos y monitoreo constante. Aunque la implementación básica puede ser rápida, la configuración y prueba del sistema son aspectos que requieren tiempo para asegurar que la aplicación tenga un tiempo de inactividad mínimo.
4. **Mantenibilidad (Código modular y documentado)**
   - **Tiempo estimado**: 1-2 semanas.
   - **Justificación**: La modularidad y documentación del código es importante para que el sistema sea escalable y fácil de actualizar. Esto no es tanto una funcionalidad directa, pero es necesario para facilitar el trabajo a largo plazo. Implementar buenas prácticas de desarrollo y documentar bien el código es importante para que la aplicación perdure en el tiempo.