Tarea 03

Presentado por:

Cristian Medina - crmedinab@unal.edu.co

Cristian Montañez - cmontanez@unal.edu.co

Justin Rodriguez - jusrodriguez@unal.edu.co

Sergio Ruiz – seruizh@unal.edu.co

Profesor:

Oscar Eduardo Alvarez Rodriguez

14/11/2024

Universidad Nacional de Colombia

Facultad de Ingeniería

Departamento de Ingeniería de Sistemas

Requerimientos funcionales:

- Registro e inicio de sesión
- Lista de contactos dinámica
- Chat individual
- Chat Grupal
- Historial de mensajes
- Notificaciones en tiempo real
- Búsqueda de mensajes y contactos
- Integración calendario académico

Requerimientos no funcionales:

- Seguridad
- Rendimiento
- Mantenibilidad
- Disponibilidad

Para requerimientos funcionales

Debe tener

Chat individual

- Justin: **(3 días)** Implementar sockets o utilizar una biblioteca como Flask-SocketIO.
- Sergio Ruiz **(8 días)** No he trabajado con funcionalidades similares o iguales, investigando al respecto, en tecnologías como node js es necesario manejar peticiones HTTP y conexiones en tiempo real. Además configura WebSockets con Socket.io, teniendo en cuenta la base de datos y el diseño front me parece adecuado 7 días.
- Daniel Montañez: **(13 días)** Similar a Sergio Ruiz, pero asumiendo que necesito algo más de tiempo para investigar o resolver posibles problemas con la integración de WebSockets y la base de datos, ya que hace bastante que no trabajo en algo relacionado.
- Cristian Medina(**8 días**): Utilizar WebSockets o bibliotecas similares como Socket.io para implementar un chat en tiempo real. Esto puede tomar más tiempo investigando sobre tecnologías como Node.js, peticiones HTTP y manejo de bases de datos.

Registro e inicio de sesión

- Justin: **(2 día)** Crear la base de datos para usuarios e implementar autenticación con contraseñas
- Sergio Ruiz **(3 dias)** Plantear la estructura de la base de datos inicial y usar web tokens para identificar a los usuarios, el diseño front es de los más simples

- Daniel Montañez **(3 días)**: Considero que es un tiempo razonable para estructurar la base de datos, implementar autenticación y asegurar compatibilidad básica en el frontend.
- Cristian Medina (**3 días**): Crear la base de datos de usuarios, implementar autenticación básica con contraseñas cifradas. Incluye el diseño de un formulario simple para el frontend.

Historial de mensajes

- Justin: **(2 días)** Almacenar mensajes en una base de datos y diseñar una interfaz para mostrar el historial.
- Sergio Ruiz **(2 dias)** Apartir del chat individual, guardar los mensajes de cada usuario en la base de datos y mostrarlos en la interfaz diseñada
- Daniel Montañez: **(2 días)** Similar a otros compañeros, enfocándose en almacenar mensajes en una base de datos y ajustando la interfaz para mostrar el historial.
- Cristian Medina(**2 días**): Guardar mensajes en una base de datos y mostrarlos de forma ordenada en la interfaz. Esta funcionalidad depende directamente del chat individual y grupal, por lo que será rápida si estas bases ya están implementadas.

Debería tener

Chat Grupal

- Justin: **(3 días)** extender el sistema de chat para soportar grupos y administrar permisos (quién puede enviar mensajes o unirse).
- Sergio Ruiz **(3 días)** Con la funcionalidad de chat individual ya completada, pensaría que es manejar la cantidad de usuarios en las salas.
- Daniel Montañez: **(5 días)** Aunque Sergio lo estima en 3 días, agregar manejo de múltiples usuarios y permisos puede requerir configuraciones adicionales si no tienes experiencia previa en esto (como es mi caso).
- Cristian Medina(**5 días**): Ampliar la lógica del chat individual para soportar salas de chat con múltiples usuarios. Manejar permisos y roles dentro de los grupos.

Lista de contactos dinámica

- Justin: **(2 días)** Implementar una API o lógica para actualizar contactos en tiempo real y mostrar cambios en la interfaz de usuario.
- Sergio Ruiz **(3 días)** Obtener los contactos de la base de datos y mostrarla en tiempo real cada vez que se haga un cambio.
- Daniel Montañez **(3 días)**: Ajustó un día adicional para asegurar que la sincronización en tiempo real se integre bien con la interfaz.
- Cristian Medina(**3 días**): Hacer uso de una API para obtener contactos desde la base de datos y sincronizar cambios en tiempo real. Esto incluye actualizar la interfaz al agregar, eliminar o modificar contactos.

Podría tener

Búsqueda de mensajes y contactos

- Justin: **(3 días)** Configurar índices en la base de datos para búsqueda eficiente y diseñar una interfaz para ingresar términos de búsqueda.
- Sergio Ruiz **(2 días)** Creación de endpoints para cada tipo de búsqueda.
- Daniel Montañez: **(2 días)** Crear endpoints y configurar búsquedas optimizadas con índices en la base de datos es sencillo, pero va a depender del diseño actual de la base de datos.
- Cristian Medina: (**2 días**): Configurar índices en la base de datos para acelerar las búsquedas y crear una interfaz para filtrar resultados.

No tendrá

Integración calendario académico

- Justin: (**5 días):** Conectar con APIs de calendario (como Google Calendar) y sincronizar eventos académicos con la aplicación.
- Sergio Ruiz **(5 días)** Gracias a la existencia de APIs como Google Calendar, la integración se puede realizar de manera rápida. Esto incluye la utilización de botones predefinidos para solicitar los permisos necesarios, permitiendo una implementación más sencilla.
- Daniel Montañez: **(5 días)** Asumo que usarías APIs existentes como Google Calendar y que necesitarías tiempo para comprender las autorizaciones necesarias.
- Cristian Medina(**5 días**): Integrar con un sistema externo como Google Calendar. Requiere aprendizaje de APIs y manejo de eventos programados.

Notificaciones en tiempo real

- Justin: **(4 días)** Utilizar websockets o servicios como Firebase para push notifications y configurar alertas para nuevos mensajes y eventos del calendario.
- Sergio Ruiz **(5 días)** Modificar los websockets para notificaciones en tiempo real e implementar otras tecnologías como OneSignal para notificaciones cuando no se esté conectado.
- Daniel Montañez: **(5 días):** Implementar WebSockets para mensajes y utilizar servicios como Firebase implica ajustes tanto en backend como en frontend.
- Cristian Medina (**5 días**): Implica aprender a integrar sistemas de notificaciones como Firebase Cloud Messaging. Es necesario manejar eventos del sistema, como mensajes nuevos, y configurar notificaciones para los dispositivos que vayan a ser usados.

Para requerimientos no funcionales

Sergio, Daniel y Cristian **(Constantemente):** A nuestro parecer estos requerimientos funcionales son difíciles asignarles días, ya que estos deben estar presente en todo momento del proyecto. Cada funcionalidad implementada debe ser monitoreada y probada constantemente para asegurar que sea segura, ofrezca un buen rendimiento, sea accesible y fácil de mantener.

Debe tener

Seguridad

- Justin: (**3 días**) Implementar encriptación de extremo a extremo (usando bibliotecas como cryptography o PyCrypto) y gestión de claves (generación, intercambio seguro y almacenamiento).

Debería tener

Rendimiento

- Justin: (**4 días**) Mejorar el rendimiento requiere identificar cuellos de botella (procesamiento de datos) y optimizarlos, lo que involucra pruebas intensivas.

Disponibilidad

- Justin: (**5 días**) Garantizar que el sistema esté siempre accesible requiere trabajo en la infraestructura y manejo de errores.

Podría tener

No tendrá

Mantenibilidad

- Justin: (**4 días**) Facilitar el mantenimiento y futuras ampliaciones implica invertir en documentación, pruebas y estructura modular.

Consenso General

Requerimientos funcionales

Chat individual

- **Estimaciones:** Justin (3 días), Sergio Ruiz (8 días), Daniel Montañez (13 días), Cristian Medina (8 días).
- **Promedio:** (3 + 8 + 13 + 8) / 4 = 8 días.
- **Explicación:** El rango de días es amplio debido a la experiencia de cada participante. Justin estima menos tiempo al estar familiarizado con tecnologías como Flask-SocketIO, mientras que Daniel y Cristian consideran tiempo adicional para investigar e integrar WebSockets con una base de datos y manejar posibles errores.

Registro e inicio de sesión

- **Estimaciones:** Justin (2 días), Sergio Ruiz (3 días), Daniel Montañez (3 días), Cristian Medina (3 días).
- **Promedio:** (2 + 3 + 3 + 3) / 4 = 2.75 ≈ 3 días.
- **Explicación:** La similitud en las estimaciones sugiere que esta funcionalidad es bien entendida por todos, dado que solo requiere una base de datos para usuarios, autenticación básica y un formulario simple para el frontend.

Historial de mensajes

- **Estimaciones:** Justin (2 días), Sergio Ruiz (2 días), Daniel Montañez (2 días), Cristian Medina (2 días).
- **Promedio:** (2 + 2 + 2 + 2) / 4 = 2 días.
- **Explicación:** Todos coinciden en que almacenar mensajes en una base de datos y mostrarlos en una interfaz es un proceso directo, especialmente si ya se implementaron las bases del chat individual y grupal.

Chat grupal

- **Estimaciones:** Justin (3 días), Sergio Ruiz (3 días), Daniel Montañez (5 días), Cristian Medina (5 días).
- **Promedio:** (3 + 3 + 5 + 5) / 4 = 4 días.
- **Explicación:** Las opiniones varían según el nivel de experiencia con la administración de permisos y múltiples usuarios en salas. Los tiempos más altos consideran problemas potenciales en la configuración de roles y permisos.

Lista de contactos dinámica

- **Estimaciones:** Justin (2 días), Sergio Ruiz (3 días), Daniel Montañez (3 días), Cristian Medina (3 días).
- **Promedio:** (2 + 3 + 3 + 3) / 4 = 2.75 ≈ 3 días.
- **Explicación:** Los tiempos son consistentes, ya que la funcionalidad depende principalmente de una API para manejar contactos en tiempo real, algo que los participantes consideran manejable con tecnologías estándar.

Búsqueda de mensajes y contactos

- **Estimaciones:** Justin (3 días), Sergio Ruiz (2 días), Daniel Montañez (2 días), Cristian Medina (2 días).
- **Promedio:** (3 + 2 + 2 + 2) / 4 = 2.25 ≈ 2 días.
- **Explicación:** La mayoría coincide en que esta funcionalidad es sencilla, dependiendo de consultas optimizadas con índices en la base de datos y una interfaz básica para el usuario.

Integración calendario académico

- **Estimaciones:** Justin (5 días), Sergio Ruiz (5 días), Daniel Montañez (5 días), Cristian Medina (5 días).
- **Promedio:** (5 + 5 + 5 + 5) / 4 = 5 días.

- **Explicación:** Todos están de acuerdo en el tiempo necesario, ya que el uso de APIs existentes como Google Calendar simplifica la implementación, aunque se necesita tiempo para comprender autorizaciones y sincronización.

Notificaciones en tiempo real

- **Estimaciones:** Justin (4 días), Sergio Ruiz (5 días), Daniel Montañez (5 días), Cristian Medina (5 días).
- **Promedio:** (4 + 5 + 5 + 5) / 4 = 4.75 ≈ 5 días.
- **Explicación:** Aunque Justin estima menos tiempo, la mayoría considera que ajustar WebSockets o servicios como Firebase para notificaciones, tanto en frontend como backend, requiere tiempo adicional.

Requerimientos no funcionales

En cuanto a los requerimientos no funcionales, consideramos que estos son elementos fundamentales que deben estar presentes a lo largo de todo el proyecto, ya que cada funcionalidad debe ser monitoreada y probada de forma constante para garantizar seguridad, buen rendimiento, accesibilidad y facilidad de mantenimiento. En este contexto, seguridad implica dedicar tiempo a implementar encriptación de extremo a extremo y una gestión adecuada de claves, mientras que el rendimiento demanda identificar y optimizar cuellos de botella mediante pruebas intensivas. Asimismo, la disponibilidad requiere un enfoque en la infraestructura y manejo de errores para asegurar el acceso continuo al sistema. Aunque no se contempla como prioridad, la mantenibilidad también es clave, ya que invertir en documentación, pruebas y una estructura modular facilitará futuras ampliaciones y el mantenimiento del sistema.