ACTIVIDAD EN CLASE ANÁLISIS DE REQUERIMIENTOS

Presentado por:
Stiven Aguirre Granada - *staguirreg@unal.edu.co*
Cesar Camilo Velandia Cubillos - *cevelandiac@unal.edu.co*
Manuel Santiago Mori Ardila - *mmori@unal.edu.co*
Andrés Felipe Perdomo Uruburu - *aperdomou@unal.edu.co*

Profesor:
Oscar Eduardo Alvarez Rodriguez
oalvarezr@unal.edu.co

Martes 26 de Noviembre

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería Sistemas e Industrial
2024

CONTEXTO

El objetivo del cliente es crear una red que permita facilitar la comunicación entre su comunidad académica. Dentro de la app se esperan dos roles, estudiantes y profesores, cada rol tiene formas de comunicación distintas. El presupuesto se tomará como infinito. (no se hará métodos de estimación por costo y riesgo)

Los estudiantes pueden buscar y agregar a otros estudiantes como contactos, enviando mensajes individuales o creando grupos entre ellos, sin restricción. También pueden interactuar con profesores, pero solo si existe una relación académica previa o están inscritos en un curso actual con ellos. En caso de no tener una relación académica el estudiante deberá enviar una solicitud al profesor para poder comunicarse con él.

Los profesores pueden contactar solo a estudiantes que estén o hayan estado inscritos en sus cursos. Tendrán un grupo de chat con los estudiantes asociados a las materias que estén dictando. Los profesores pueden agregar a otros profesores como contactos sin restricciones. Se notificarán a los estudiantes de las actividades académicas asociadas al curso. Encuestas a los estudiantes.

MÉTODO MoSCoW

1. Must Have (M):

- Estos son los requisitos que deben cumplirse para que el proyecto sea exitoso. Sin ellos, el proyecto no podrá continuar o completarse. negociables y deben ser entregados.

2. Should Have (S):

- Estos requisitos son importantes pero no críticos para la operación. Si no se entregan, el proyecto todavía puede ser un éxito, aunque con algunas limitaciones.

3. Could Have (C):

- Estos son requisitos deseables pero no necesarios. Son los primeros en ser eliminados si el tiempo y los recursos son limitados.

4. Won't Have (W):

- Estos son requisitos que se han acordado que no se entregarán. Este grupo puede incluir elementos que se consideren para futuras versiones o fases del proyecto.

PRIORIZACIÓN
(MoSCoW)

REQUERIMIENTOS FUNCIONALES:

- **Registro e inicio de sesión (M)**
  - Para el correcto uso de la aplicación es completamente necesario el inicio de sesión, ya que mediante a este los usuarios hacen uso de los roles de estudiante o docente, además de guardar contactos, registros de chat (en caso de tenerlos), y poder pertenecer a grupos (en caso de tenerlos).

- **Lista de contactos dinámica (S)**
  - Si bien es una opción bastante práctica para ofrecer al usuario de la aplicación, no es fundamental para el funcionamiento de esta aplicación pero aun asi es una característica deseable para su completa funcionalidad.

- **Chat individual (M)**
  - Al ser una aplicación enfocada a la comunicación por texto, el eje central es lograr conectar a las personas mediante el uso de chats, principalmente individuales.

- **Chat grupal (S)**
  - Este requerimiento ofrece una gran posibilidad para el usuario, sin embargo esto no es un requerimiento obligatorio para que la aplicación sea entregada al usuario, aun así sería un añadido que complementará muy bien la aplicación.

- **Historial de mensajes (M)**
  - Es algo que se debería tener obligatoriamente si se va a tener cualquier tipo de chat, la aplicación necesita esta funcionalidad para que esté completa,

- **Notificaciones en tiempo real (W)**
  - Es una característica que no es importante en absoluto, y no aportaría realmente nada sustancial al funcionamiento de la aplicación.

- **Búsqueda de mensajes y contactos (M)**
  - La búsqueda de mensajes es fundamental para el uso de la aplicación, ya que al ser un chat académico la información y como esta está organizada es importante, ya que la información adjuntada por docentes o trabajos hechos con compañeros deben ser alcanzados con facilidad en caso de ser necesarios para futuras ocasiones.

- **Integración con calendario académico (W)**
  - Este requerimiento parece más para una siguiente versión, el chat puede funcionar perfectamente sin este requerimiento.

REQUERIMIENTOS NO FUNCIONALES:

- **Seguridad (M)**
  - Es necesaria para proteger la información del usuario u otro tipo de información sensible.

- **Rendimiento (S)**
  - Es importante que la aplicación tenga un buen rendimiento para su utilización, pero no es algo fundamental.

- **Mantenibilidad (C)**
  - Sería una característica que aportaría funcionalidad a largo plazo de la aplicación, aun así no es necesario para el buen funcionamiento de esta.

- **Disponibilidad (S)**
  - Es importante y algo que se debería agregar en el proyecto, pero no es fundamental para el uso de la aplicación.

ESTIMACIÓN
(Días)

Manuel Santiago Mori Ardila

- **Must (M)**
  - **Registro e inicio de sesión (5)**
    Las tecnologías ofrecen buenas librerías para este requerimiento. El otro tiempo se puede utilizar validación, formularios de inicio de sesión y registro.
  - **Búsqueda de mensajes y contactos (8)**
    Es una tarea que puede requerir un buen tiempo por el uso de una base de datos no relacional (estructuración adecuada y creación de índices).
  - **Chat individual (13)**
    Requiere especial cuidado al ser un requerimiento principal, la comunicación en tiempo real puede presentar varios desafíos, por ejemplo el manejo de errores de conexión.
  - **Historial de mensajes (5)**
    El historial de mensajes implica la persistencia y retribución de la base de datos, lo cuál no parece tan dispendioso. Aunque sería bueno disponer de tiempo para el tema de interfaz.
  - **Seguridad (5)**
    Aquí se podría implementar una seguridad sencilla, requerimiento de contraseñas seguras, hash de contraseñas y autenticación de usuarios, para lo cual hay varias librerías.

- **Should (S)**
  - **Chat grupal (5)**
    Ya teniendo el chat individual, varias cosas se podrían adecuar al chat grupal, además se le puede añadir temas de gestión de grupo por administradores.
  - **Rendimiento (8)**
    Mucho tiempo podría utilizarse identificando los sitios donde se pierde eficiencia.
  - **Disponibilidad (3)**
    Sería necesario revisar todo lo referente a servidores, el haber realizado el requerimiento de rendimiento aliviaría bastante trabajo de disponibilidad.
  - **Lista de contactos dinámica (1)**
    Solo sería necesario organizar los contactos en una lista.

- **Could (C)**
  - **Mantenibilidad (5)**
    Se utilizará el tiempo para reestructurar código para que siga buenas prácticas y tener documentación adecuada, puede tomar bastante tiempo solo hasta este punto se revisa la calidad del código.

- **Won't (W)**
  - **Notificaciones en tiempo real (3)**
    Algunas dificultades que en este punto estarán resueltas por el chat individual se deben poder aplicar.
  - **Integración con el calendario académico (13)**
    Puede resultar difícil al tener que lidiar con el acceso a una API externa.

Cesar Camilo Velandia Cubillos

- **Must (M)**
  - **Registro e inicio de sesión (5)**
    Requiere el desarrollo de formularios funcionales para inicio de sesión y registro. Las tecnologías actuales facilitan esta tarea gracias a librerías preexistentes, pero se debe destinar tiempo adicional a validaciones y pruebas.
  - **Búsqueda de mensajes y contactos (13)**
    Incluir funcionalidad para encontrar mensajes específicos y contactos en un sistema no relacional requiere planificación y pruebas de rendimiento, considerando diferentes escenarios.
  - **Chat individual (13)**
    Al ser un requerimiento crítico, exige especial atención en la implementación de comunicación en tiempo real, manejo de errores de conexión y pruebas de alta carga.
  - **Historial de mensajes (5)**
    La persistencia y recuperación eficiente de mensajes en la base de datos son clave.

Se debe invertir tiempo en diseñar una interfaz de usuario intuitiva para acceder al historial.

- **Seguridad (5)**
  En este punto, es fundamental incluir prácticas como hash de contraseñas, autenticación robusta y validaciones. Se puede aprovechar librerías para acelerar la implementación.

- **Should (S)**
  - **Chat grupal (8)**
    Basado en el chat individual, la lógica grupal puede reutilizarse parcialmente. Se debe añadir funcionalidades como gestión de administradores y permisos.
  - **Rendimiento (13)**
    Requiere identificar cuellos de botella en la aplicación y optimizar procesos clave. Este esfuerzo también asegura una mejor experiencia del usuario.
  - **Disponibilidad (3)**
    La revisión de configuraciones de servidor y mecanismos de redundancia es crucial. Los avances realizados en rendimiento contribuirán significativamente.
  - **Lista de contactos dinámica (2)**
    Se basa en organizar contactos en una estructura eficiente, con actualizaciones en tiempo real. Su implementación no debería ser compleja.

- **Could (C)**
  - **Mantenibilidad (8)**
    Incluye la refactorización del código para seguir estándares, mejorar la legibilidad y crear documentación. Esto prepara el proyecto para futuras actualizaciones.

- **Won't (W)**
  - **Notificaciones en tiempo real (3)**
    A pesar de las dificultades iniciales, los avances logrados con el chat individual deben facilitar la implementación futura de esta funcionalidad.
  - **Integración con el calendario académico (21)**
    Es un requerimiento complejo por la necesidad de interactuar con APIs externas y manejar datos de terceros, lo que lo convierte en una tarea de baja prioridad actual.

Stiven Aguirre Granada

- **Must (M)**
  - **Registro e inicio de sesión (5)**
    Este tiempo cubre la implementación básica para que los usuarios puedan autenticarse, incluyendo manejo de errores y validación de datos para garantizar la funcionalidad esencial.
  - **Búsqueda de mensajes y contactos (13)**
    Incluir funcionalidad para encontrar mensajes específicos y contactos en un sistema

no relacional requiere planificación y pruebas de rendimiento, considerando diferentes escenarios.

- **Chat individual (21)**
  Este tiempo se destina a implementar un sistema robusto de comunicación en tiempo real, con manejo de conexiones, interfaces intuitivas y pruebas para evitar fallos.
- **Historial de mensajes (8)**
  Permitir que los usuarios accedan a mensajes pasados de forma ordenada y eficiente requiere desarrollo tanto en la parte del servidor como en la interfaz.
- **Seguridad (8)**
  Garantizar que los datos de los usuarios estén protegidos implica desarrollar medidas de protección, como cifrado y autenticación segura, además de realizar pruebas para verificar su efectividad.

- **Should (S)**
  - **Chat grupal (8)**
    Ampliar la funcionalidad del chat para permitir la interacción entre múltiples usuarios incluye diseño de roles, permisos, y pruebas para garantizar estabilidad.
  - **Rendimiento (13)**
    Optimizar el sistema para responder rápido incluso con alto tráfico requiere análisis detallado, ajustes en la arquitectura y validación bajo diferentes condiciones.
  - **Disponibilidad (5)**
    Configurar y probar el sistema para que esté accesible constantemente, incluso ante fallos, requiere implementar redundancia y pruebas de tolerancia a fallos.
  - **Lista de contactos dinámica (3)**
    Crear una lista actualizable que refleje los cambios en tiempo real implica vincular la interfaz con los datos del sistema de manera eficiente.

- **Could (C)**
  - **Mantenibilidad (8)**
    Mejorar la estructura del código y crear documentación detallada permitirá que futuras actualizaciones y correcciones sean más fáciles de implementar.

- **Won't (W)**
  - **Notificaciones en tiempo real (8)**
    Implementar un sistema para alertar a los usuarios de eventos en tiempo real requiere integración con servicios de notificaciones y validaciones adicionales.
  - **Integración con el calendario académico (21)**
    Conectar el sistema a una plataforma externa para sincronizar actividades académicas implica desarrollar una integración sólida y pruebas exhaustivas para asegurar la precisión.

Andrés Felipe Perdomo Uruburu

- **Must Have (M):**
  - **Registro e inicio de sesión (5 días):**
    Este módulo es esencial para la funcionalidad base de la aplicación. Implica gestionar el flujo inicial de usuarios, crear sesiones seguras y asegurar la identificación de roles desde el primer uso.
  - **Búsqueda de mensajes y contactos (13 días):**
    Una búsqueda eficiente implica no solo obtener resultados precisos sino también diseñar un sistema que soporte grandes volúmenes de datos, respondiendo rápidamente y evitando cuellos de botella.
  - **Chat individual (21 días):**
    Este componente requiere una arquitectura robusta para manejar múltiples conversaciones simultáneas, garantizar la entrega de mensajes incluso en condiciones de red inestables y ofrecer una experiencia fluida para el usuario.
  - **Historial de mensajes (8 días):**
    La capacidad de almacenar y recuperar mensajes históricos con precisión necesita un diseño cuidadoso de almacenamiento, manejo de grandes volúmenes y una interfaz que facilite la navegación en el tiempo.
  - **Seguridad (8 días):**
    Proteger la información del usuario implica implementar protocolos de seguridad estándar, prever posibles vulnerabilidades y garantizar que los datos permanezcan seguros en tránsito y en reposo.

- **Should Have (S):**
  - **Chat grupal (8 días):**
    Los grupos requieren funcionalidades adicionales como administración de miembros, configuraciones de privacidad y sincronización en tiempo real, lo que añade complejidad al sistema base.
  - **Rendimiento (13 días):**
    Mejorar el rendimiento implica identificar y optimizar los procesos más demandantes, asegurando tiempos de respuesta rápidos incluso bajo alta carga, lo cual es clave para la usabilidad general.
  - **Disponibilidad (5 días):**
    Garantizar que el sistema esté siempre operativo incluye configuraciones de redundancia, monitoreo y ajustes para evitar interrupciones inesperadas en el servicio.
  - **Lista de contactos dinámica (3 días):**
    Una lista interactiva que se actualice automáticamente según las acciones del usuario aporta comodidad, pero su implementación es sencilla al aprovechar datos ya existentes.

- **Could Have (C):**
  - **Mantenibilidad (8 días):**
    Diseñar un sistema que facilite su actualización y depuración futura requiere estructurar el código de forma clara, implementar buenas prácticas y agregar documentación adecuada.

- **Won't Have (W):**
  - **Notificaciones en tiempo real (8 días):**
    Aunque es útil para mantener a los usuarios informados al instante, su implementación requiere configurar sistemas externos, ajustar sincronización y realizar pruebas específicas.
  - **Integración con el calendario académico (21 días):**
    Sincronizar con herramientas externas agrega la complejidad de entender y manejar APIs de terceros, además de diseñar procesos de actualización y manejo de errores en tiempo real.

CONSENSO

- **Must Have (M):**
  - **Registro e inicio de sesión (5 días):**
    Consideramos que este módulo es fundamental para la funcionalidad básica de la aplicación. Su desarrollo incluye la creación de formularios funcionales, manejo de errores y validaciones. Confiamos en que el uso de librerías actuales nos permitirá optimizar este proceso.
  - **Búsqueda de mensajes y contactos (13 días):**
    Acordamos que implementar una búsqueda eficiente requiere no solo estructurar adecuadamente los datos, sino también optimizar índices y garantizar tiempos de respuesta rápidos. Este tiempo refleja la importancia de brindar precisión y fluidez a los usuarios.
  - **Chat individual (21 días):**
    Al ser el núcleo de la aplicación, entendemos que este componente debe ser robusto, con soporte para múltiples conversaciones simultáneas y manejo confiable de conexiones en tiempo real. Nuestro consenso incluye tiempo para pruebas exhaustivas, asegurando la calidad.
  - **Historial de mensajes (8 días):**
    Hemos considerado que recuperar mensajes almacenados de manera eficiente requiere un diseño equilibrado entre backend e interfaz, priorizando accesibilidad y orden. Este tiempo nos permitirá enfocarnos en detalles como paginación y visualización.

- **Seguridad (8 días):**
  La protección de datos sensibles es esencial. Hemos acordado destinar este tiempo a implementar buenas prácticas de cifrado, autenticar usuarios y realizar pruebas que refuercen la confiabilidad del sistema frente a vulnerabilidades.

- **Should Have (S):**
  - **Chat grupal (8 días):**
    Ampliar las funcionalidades del chat individual para permitir interacción grupal es un objetivo importante para nosotros. Este tiempo incluye desarrollar herramientas como gestión de roles y permisos, garantizando sincronización y estabilidad.
  - **Rendimiento (13 días):**
    Todos estamos de acuerdo en que mejorar el rendimiento requiere un análisis minucioso de cuellos de botella y pruebas bajo condiciones de alta carga. Este tiempo asegura que el sistema responda ágilmente incluso en situaciones exigentes.
  - **Disponibilidad (5 días):**
    Nos parece crucial que el sistema esté accesible de manera constante. Este tiempo refleja el esfuerzo necesario para configurar redundancia, realizar pruebas de fallos y asegurar alta disponibilidad.
  - **Lista de contactos dinámica (3 días):**
    Consideramos que implementar esta funcionalidad no es complejo, ya que se basa en aprovechar datos existentes y reflejar cambios en tiempo real. Este tiempo es adecuado para cumplir con las expectativas del usuario.

- **Could Have (C):**
  - **Mantenibilidad (8 días):**
    Como equipo, valoramos la calidad del código. Este tiempo está destinado a estandarizar la estructura, refactorizar según sea necesario y generar documentación que facilite futuras actualizaciones y mantenimiento.

- **Won't Have (W):**
  - **Notificaciones en tiempo real (8 días):**
    Aunque relevante, hemos acordado priorizar otros módulos. Este tiempo refleja las dificultades de integrar servicios externos y asegurar sincronización sin fallos.
  - **Integración con el calendario académico (21 días):**
    Todos coincidimos en que este es un requerimiento complejo debido a la necesidad de interactuar con APIs de terceros. Acordamos posponerlo dada su baja prioridad actual en comparación con otros módulos críticos.