Universidad del Valle

Facultad de Ingeniería

Escuela de Ingeniería de Sistemas y Computación

750018C PROYECTO INTEGRADOR I 2025-2

Mini proyecto #3 - Plataforma de Videoconferencias

1. Propósito general

Desarrollar y desplegar una plataforma web de videoconferencia que permita la creación de reuniones, chat en tiempo real, transmisión de voz y vídeo entre 2 y 10 participantes. El sistema ofrecerá autenticación multicanal, una interfaz accesible y responsiva, y servicios de comunicación en tiempo real basados en Socket.io, Peer.js y servidores STUN.

2. Alcance por Sprint

Sprint	Enfoque funcional	Entregables en producción
Sprint 1: Gestión de usuarios + GUI	• Registro, login, logout, recuperación de contraseña, edición y borrado de cuenta (manual + 2 proveedores OAuth). • Creación de reunión y exploración de la GUI de la plataforma.	• Frontend en Vercel (menú, inicio + mapa del sitio, sobre nosotros, pie de página; 2 heurísticas, 1 WCAG). • Backend - usuarios en Render (Node/TS). • BD Firestore (users). • Video e informe de la prueba de usuario.
Sprint 2: Chat en tiempo real	• Conexión de 2-10 usuarios a una reunión mediante ID. • Chat en tiempo real.	• Frontend actualizado (4 heurísticas, 2 WCAG). • Backend - usuarios + chat (Socket.io). • Firestore (users + meetings). • Videos + informes S1-S2.
Sprint 3: Transmisión de voz	• Chat + audio (full-duplex, activar/desactivar micrófono).	• Frontend con 7 heurísticas y 3 WCAG (operable, comprensible, perceptible). • Backend - usuarios + chat + voz (Socket.io + Peer.js, STUN). • Firestore (users + meetings + AI resumen chat). • Videos + informes S1-S3.
Sprint 4: Transmisión de video	• Chat + audio + video (activar/desactivar cámara).	• Frontend con 10 heurísticas y 4 WCAG (añade robusto). • Backend - usuarios + chat + voz + video (2 STUN, Peer.js). • Firestore (users + meetings + AI resúmenes). • Videos + informes S1-S4.

3. Requisitos técnicos

Frontend

- Vite.js · React · TypeScript · SASS.
- Consumo vía Fetch API.
- Diseño responsivo, heurísticas y WCAG progresivas.
- Variables de entorno, código en inglés, estilo limpio, JSDoc.

Backend

- Node.js + Express con TypeScript (1→4 micro-servicios).
- Render deploy, variables de entorno, estilo limpio, JSDoc.
- Servicios en tiempo real con Socket.io y Peer.js.
- Servidores STUN propios (voz y video).

Base de datos

- Firebase Authentication + Firestore.
- Colecciones: users, meetings, chat, summaries.

DevOps & Gestión

- Taiga: planificación y cierre de cada sprint.
- GitHub: rama dedicada por integrante, commits pequeños, *Pull Request* sprint-X-release.

4. Roles del equipo

El profesor asignará un equipo de 5 estudiantes. Cada equipo tendrá un líder (Product Owner) encargado de coordinar el trabajo de sus compañeros según el rol correspondiente, de los cuales los asigna el equipo de trabajo:

Rol	**Responsabilidades para este mini proyecto**
Frontend	Encargado de implementar y desplegar la interfaz gráfica del usuario usando principios de UX/UI, HTML5, CSS3, Typescript, React y SASS. Debe asegurar una experiencia visual y funcional fluida, integrando también componentes de diseño desde herramientas como Figma.
Backend	Diseña, desarrolla y despliega la lógica del servidor usando Node.js, Express y Firebase. Además, implementa tecnologías de comunicación en tiempo real como WebSockets y WebRTC con su respectivo servidor STUN, esenciales para funciones como chats y transmisiones. Maneja rutas, autenticación, controladores y la comunicación entre el cliente y la base de datos. Su labor garantiza un funcionamiento seguro, eficiente y dinámico de la aplicación.
Base de datos	Gestiona el almacenamiento, consulta y persistencia de la información. Emplea Firebase para crear esquemas, colecciones o documentos, asegurando el correcto funcionamiento de operaciones CRUD en los proyectos.

Gestión de proyectos & VCS	Administra el avance del equipo utilizando metodologías ágiles como SCRUM con la plataforma TAIGA. Documenta tareas, actualiza el tablero de sprints y colabora en la organización del trabajo con control de versiones en GitHub, asegurando trazabilidad y colaboración efectiva.
Pruebas	Evalúa la experiencia del usuario final mediante pruebas prácticas de uso. Identifica barreras de interacción, problemas de diseño o funcionalidad, y sugiere mejoras en base a retroalimentación real para incrementar la accesibilidad, eficiencia y satisfacción del usuario.

5. Entrega

- Pull Request final con tag sprint-X-release antes de la fecha límite.
- URL pública de Vercel y Render funcionando.
- Video de pruebas e informe PDF cargados en el drive del equipo de trabajo.
- Sustentación pública con el profesor y el curso completo

6. Penalizaciones

El estudiante que no trabaje en equipo y sea retirado por el profesor o se retire del grupo por decisión propia, tendrá que trabajar solo en el mini proyecto que está por presentar, donde la penalidad será de -1.0 para la nota final del mini proyecto que está trabajando.