PLATAFORMA DE VIDEOCONFERENCIAS
Manual de Usuario e Informe Técnico

DAVID ALEXANDER CARVAJAL MOLINA 2329524-2724
DANIEL ESCOBAR BAHOZ 2371732-2724
BRANDON ARLEY FERNANDEZ ROSERO -2724

Proyecto Integrador I – Mini Proyecto #3
Universidad del Valle
2025-2

Contenido

📘 MANUAL DE USUARIO – Plataforma de Videoconferencias

1. Introducción

Este manual describe el uso de la plataforma web de videoconferencias desarrollada como parte del proyecto académico. El sistema permite a los usuarios **crearse una cuenta, ingresar a reuniones, enviar mensajes en tiempo real y transmitir voz y video**.

El objetivo es brindar una experiencia accesible, clara y segura, en acuerdo con WCAG 2.1 y las heurísticas de Nielsen.

---

2. Acceso y Autenticación

2.1 Registro de Cuenta

Para crear una cuenta:

1. Ingrese a la página principal.
2. Seleccione **“Sign Up / Register”**.
3. Complete el formulario:
    - Nombre
    - Email
    - Contraseña
4. Confirme.
5. Su cuenta quedará creada en Firebase Authentication.

También puede registrarse mediante:

- **Google OAuth**
- **Github OAuth**

2.2 Inicio de Sesión

1. Seleccione **“Login”**.
2. Ingrese su correo y contraseña.
3. Puede ingresar también con cuentas OAuth.
4. La sesión se mantiene mediante Firebase + tokens internos.

2.3 Recuperación de Contraseña

1. En la pantalla de login, elija **“Forgot password?”**.
2. Ingrese su email.
3. Recibirá un enlace seguro desde Firebase.

2.4 Cierre de Sesión

Presione el botón **Logout** desde el menú superior.

---

3. Navegación General del Sistema

La plataforma cuenta con:

- **Navbar responsivo**: Inicio, Reuniones, Perfil, Logout.
- **Footer accesible** con enlaces institucionales.
- **Modo adaptable móvil/escritorio**.

---

4. Funcionalidades Principales

4.1 Creación de Reunión

1. Ingrese al dashboard con su sesión activa.
2. Seleccione **“Create Meeting”**.
3. El sistema genera un **ID único** almacenado en *meetings/ Firestore*.
4. Comparta el enlace con otros usuarios.

4.2 Ingreso a una Reunión

1. Seleccione **“Join meeting”** desde el menú.
2. Ingrese el **Meeting ID**.
3. Accederá a la sala con chat, audio y video.

4.3 Chat en Tiempo Real

Dentro de la sala:

- Escriba mensajes en el panel derecho.
- Los mensajes se sincronizan mediante **Socket.io**.
- El historial se almacena en Firestore.

4.4 Transmisión de Voz

- Active/desactive el micrófono con el botón correspondiente.
- La comunicación punto a punto usa **Peer.js** y servidores **STUN propios**.

4.5 Transmisión de Video

- Active/desactive la cámara.
- La plataforma soporta entre **2 y 10 usuarios simultáneos**.
- La transmisión se gestiona con WebRTC + STUN/Peer.js.

4.6 Perfil de Usuario

Permite:

- Editar nombre, foto, datos básicos
- Eliminar cuenta
- Cambiar contraseña (desde Firebase)

---

5. Accesibilidad (WCAG 2.1)

- Contrastes adecuados (AA)
- Teclado navegable
- Labeling correcto en botones
- Diseño perceptible/operable/comprensible/robusto según sprint

---

6. Soporte

En caso de fallos:

- Refresque la página
- Verifique su conexión
- Contacte al equipo técnico proporcionando capturas y descripción del error

---

📗 INFORME TÉCNICO – Plataforma de Videoconferencias

1. Propósito General

Desarrollar una plataforma web de videoconferencias con:

- Autenticación (manual + OAuth)
- Reuniones 2–10 usuarios
- Chat en tiempo real
- Audio y video con WebRTC (Peer.js)
- STUN propio
- BD en Firestore
- Accesibilidad progresiva por Sprint

---

2. Arquitectura General del Sistema

Frontend

- **Vite.js + React + TypeScript**
- Estilos: **SASS**, CSS Modules y Tailwind opcional
- Componentización limpia
- Comunicación vía **Fetch API**
- Despliegue: **Vercel**
- Enfoque UX/UI acorde a heurísticas y WCAG

Backend

- **Node.js + Express + TypeScript**
- 1 a 4 microservicios según sprint:
    - Auth
    - Meetings
    - Chat
    - Streaming (Peer.js + STUN)
- **Socket.io** para chat y señalización
- **Peer.js** para WebRTC
- STUN propio (voz y video)
- Despliegue: **Render**

Base de Datos — Firestore

Colecciones:

- **users**
- **meetings**
- **chat**
- **summaries** (sprint 3–4)

DevOps

- Taiga → planificación de sprint
- GitHub → ramas por integrante + PR sprint-X-release

---

3. Desarrollo por Sprint

Sprint 1 – Usuarios + GUI

Funcionalidades:

- Registro, login, logout
- Recuperación de contraseña
- Edición/borrado de cuenta
- Creación básica de reunión
- Frontend inicial en Vercel
- Backend usuarios en Render
- Firestore users
- 2 heurísticas + 1 WCAG
- Informe + video de pruebas

Sprint 2 – Chat en Tiempo Real

- Conexión de 2 a 10 usuarios con un Meeting ID
- Chat en vivo vía Socket.io
- Firestore: users + meetings
- 4 heurísticas + 2 WCAG
- Informe S1–S2 + video

Sprint 3 – Voz

- Chat + audio full-duplex
- Activar/desactivar micrófono
- STUN para voz + Peer.js

- Firestore: summaries automáticos por IA
- 7 heurísticas + 3 WCAG
- Informe S1–S3 + video

Sprint 4 – Video

- Chat + audio + video
- Activar/desactivar cámara
- STUN para video
- Firestore meetings + AI summaries
- 10 heurísticas + 4 WCAG
- Informe S1–S4 + video

---

4. Diagrama de Arquitectura (Descripción textual)

Cliente (React/Vite)
⬇ Fetch API
⬅ Socket.io (Tiempo real)
⬅ WebRTC (Audio/Video)

Backend (Node/Express)

- Auth Microservice
- Meetings Microservice
- Chat Microservice
- WebRTC Signaling

Firebase Firestore

- Persistencia de usuarios, reuniones, mensajes y resúmenes

STUN Servers (propios)

- Negociación WebRTC para voz/video

---

5. Accesibilidad y UX/UI

Se garantizó:

- Cumplimiento progresivo de **heurísticas de Nielsen** (2 → 10)
- Cumplimiento progresivo **WCAG 2.1 AA** (1 → 4)
- Interfaz responsiva
- Uso de patrones consistentes en navegación
- Señales de estado visibles (mute, cam off, conectado, desconectado)

---

6. Pruebas de Usuario

Las pruebas incluyen:

- Flujo completo de registro → login → reunión
- Verificación de accesibilidad
- Análisis de satisfacción de uso
- Grabaciones en video (obligatorias por sprint)

---

7. Conclusiones Técnicas

- El sistema cumple con los requisitos de video, voz, chat y autenticación.
- El uso de Socket.io + Peer.js permite una comunicación robusta y en tiempo real.
- La arquitectura es escalable mediante microservicios.
- Firestore facilita actualizaciones en vivo (listeners).
- El despliegue en Vercel + Render permite CI/CD rápido.