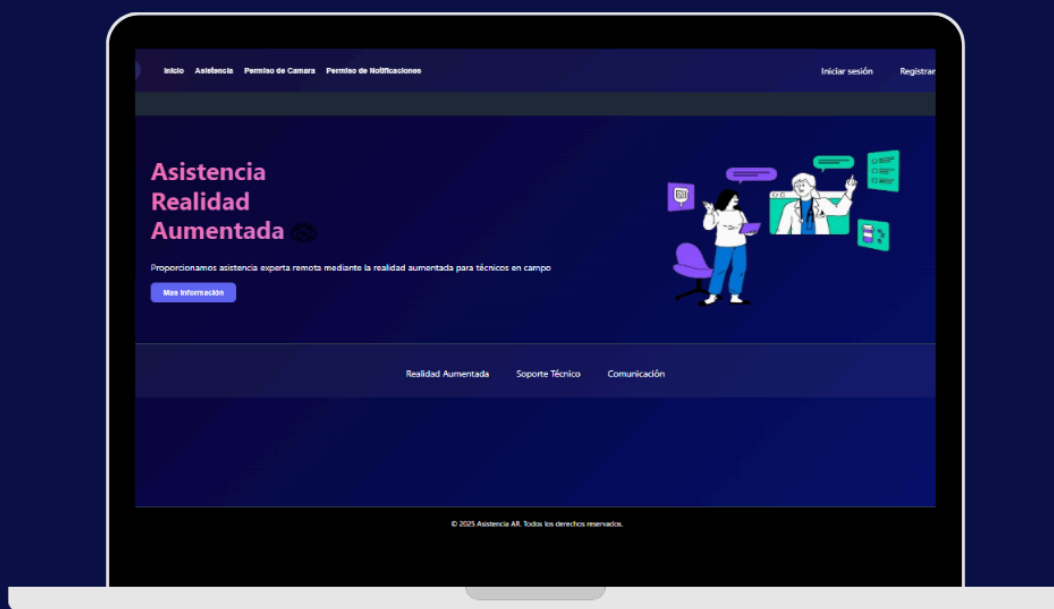




*Asistencia por Realidad
Aumentada*

MANUAL DE USUARIO



SITIO WEB

www.apra.com



ÍNDICE

| | |
|---|----------|
| Parte 1 - Manual del Usuario | 3 |
| 1. Introducción | 3 |
| 2. Requisitos del sistema | 3 |
| 3. Ingreso al sistema | 3 |
| 4. Uso general de la aplicación | 3 |
| 4.1 Registro de asistencia | 3 |
| 4.2 Consultar asistencias | 4 |
| 4.3 Modificar asistencias | 4 |
| 4.4 Eliminar asistencias | 4 |
| 5. Guía visual de la aplicación | 5 |
| 5.1. Pantalla de inicio | 5 |
| 5.2. Formulario de registro | 5 |
| 5.3. Tabla de asistencias | 5 |
| 5.4. Reconocimiento facial | 5 |
| 5.5 Vista de cámara | 5 |
| 6. Resolución de problemas | 6 |
| Parte 2 – Manual del Programador | 6 |
| 1. Tecnologías empleadas | 6 |
| 2. Estructura del código | 6 |
| 3. Descripción técnica de componentes y hooks | 7 |
| App.jsx | 7 |
| FormularioIniciarSesion.jsx | 7 |
| Asistencia.jsx | 8 |
| FormularioRegistrarse.jsx | 8 |
| BajaAsistencia.jsx | 9 |
| AltaAsistencia.jsx | 9 |
| Modificar Asistencia.jsx | 10 |
| Notificaciones.jsx | 10 |
| PermisoCamara.jsx | 11 |
| Reconocimiento.jsx | 11 |
| CameraComponent.jsx | 11 |
| ProtectedRoute.jsx | 12 |
| useAsistencias.jsx | 12 |
| useUsuario.jsx | 13 |
| 4. Flujo general de datos | 13 |
| 5. Estilo y convenciones | 14 |
| 6. Deploy | 14 |
| Créditos | 14 |

Parte 1 - Manual del Usuario

1. Introducción

La página **Asistencia por Realidad Aumentada** es una aplicación web desarrollada en **React** que permite registrar, visualizar y gestionar asistencias mediante una interfaz moderna e interactiva.

El objetivo principal es facilitar el control de asistencia de alumnos o personal usando una interfaz accesible desde navegador, integrando tecnologías modernas como **Firebase**, **React Router**, etc.

2. Requisitos del sistema

Requisitos mínimos:

- Navegador moderno (Google Chrome, Edge o Firefox).
- Conexión a Internet.
- Cámara web activa (opcional, si se utiliza la función de realidad aumentada).
- Permisos de acceso a cámara y micrófono si son requeridos.

3. Ingreso al sistema


Pasos:

1. Abrir el sitio web o ejecutar el proyecto localmente (`npm start`).
2. En la pantalla inicial, iniciar sesión con usuario y contraseña o cuenta de Google.
3. Si el usuario no está registrado, crear una cuenta nueva.
4. Una vez autenticado, el sistema redirige a la pantalla principal de asistencias.

4. Uso general de la aplicación

4.1 Registro de asistencia

- Ir al módulo *Nueva Asistencia*.
- Completar los campos requeridos (curso, división, nombre y apellido, fecha y estado).
- Presionar el botón "Guardar Asistencia".


- Aparece un mensaje confirmando el registro exitoso “ Asistencia registrada correctamente”.

4.2 Consultar asistencias


En la sección “**Asistencias**”, se puede:

- Ingresar a la sección *Asistencias*.
- Se muestra una tabla con todos los registros.
- Se pueden ordenar o buscar registros por curso, división, fecha o estado..

4.3 Modificar asistencias

- Ir al módulo *Modificar Asistencia*.
- Haga clic en “Editar” según el estudiante a modificar.
- Completar los campos requeridos (curso, división, nombre y apellido, fecha y estado).
- Presionar el botón “Guardar Cambios”.
- Aparece un mensaje confirmando el registro exitoso “ Asistencia actualizada correctamente”.

4.4 Eliminar asistencias

- Ir al módulo *Eliminar Asistencia*.
- Haga clic en “Eliminar” según el estudiante a eliminar.
- Luego se muestra la siguiente pregunta “¿Seguro que deseas eliminar la asistencia de Estudiante?”.
- Presionar el botón “Aceptar”.
- Aparece un mensaje confirmando el registro exitoso “ Asistencia eliminada correctamente”.

5. Guía visual de la aplicación

5.1. Pantalla de inicio

La pantalla de inicio permite al usuario autenticarse. Desde aquí se accede a todas las funciones del sistema.

5.2. Formulario de registro

Formulario para ingresar nuevos usuarios, es decir, registrarlos. Incluye validación de campos obligatorios.

5.3. Tabla de asistencias

Muestra todos los registros existentes. Posee botones para registrar, eliminar y modificar los datos de los estudiantes..

5.4. Reconocimiento facial

Descripción funcional:

Esta vista permite **registrar asistencias utilizando la cámara del dispositivo y detección facial**.

El sistema utiliza el modelo `face-api.js` para reconocer rostros visibles en el cuadro y marcar los identificables.

El usuario puede:

- Permitir el acceso a la cámara.
- Ver su imagen en pantalla con un marco animado.
- Presionar el botón “**Tomar asistencia**” para capturar la imagen.
- Observar los resultados: rostros identificables (verde) y no identificables (rojo).
- Repetir el proceso con “Nueva foto”.

5.5 Vista de cámara

Descripción funcional:

Permite **visualizar la cámara en tiempo real** dentro de la aplicación.

Sirve como herramienta auxiliar para probar el acceso del navegador al dispositivo de video antes del uso del reconocimiento facial.

Pasos de uso:

1. Al abrir la vista, el navegador solicitará permiso para usar la cámara.

2. Una vez aceptado, la imagen del usuario se muestra en tiempo real.
3. Si la cámara no se puede acceder, aparece un mensaje en la consola.

6. Resolución de problemas

| Problema | Causa | Solución |
|-------------------------------------|---|---------------------------------------|
| No se carga la tabla de asistencias | Error de conexión a internet o Firebase | Verificar conexión o reiniciar sesión |
| No se guardan los datos | Falta completar campos requeridos | Revisar el formulario |
| Error "Permission denied" | Usuario sin autenticación activa | Cerrar sesión y volver a iniciar |

Parte 2 – Manual del Programador

1. Tecnologías empleadas

- React JS.
- Firebase (base de datos y auth).
- React Router DOM.
- Java Docs.
- Canva.

2. Estructura del código

```
src/
├ components/
│   ├── Asistencia.jsx
│   ├── PáginaInicio.jsx
│   ├── PermisoCamara.jsx
│   ├── FormularioIniciarSesion.jsx
│   ├── FormularioRegistrarse.jsx
│   ├── Notificaciones.jsx
│   ├── ProtectedRoute.jsx
│   ├── Reconocimiento.jsx
│   ├── CameraComponent.jsx
│   └── AuthContext.jsx
```

```
├ hooks/
│   └ useAsistencias.jsx
│   └ useUsuario.jsx
│   └ Modificar asistencia.jsx
│   └ BajaAsistencia.jsx
│   └ AltaAsistencia.jsx
├ css/
│   └ Asistencia.css
│   └ FormularioIniciarSesion.css
│   └ FormularioRegistrarse.css
│   └ Notificaciones.css
│   └ PáginaInicio.css
│   └ PermisoCamara.css
├ App.jsx
└ main.jsx
```

3. Descripción técnica de componentes y hooks

App.jsx

Ruta: `/src/App.jsx`

Función:

Define la estructura principal de la aplicación y gestiona las rutas mediante **React Router DOM**.

Redirige entre pantallas como *FormularioInicarSesion* y *Asistencia*.

Hooks usados:

- `BrowserRouter` y `Routes` para el enrutamiento.

Relaciones:

- Llama a `FormularioIniciarSesion`, `FormularioRegistrarse`, `Asistencia`, `PáginaInicio`, `Notificaciones`, `PermisoCamara`, `ProtectedRoute`, `AltaAsistencia`, `BajaAsistencia`, `ModificarAsistencia`.

FormularioIniciarSesion.jsx

Ruta: `/src/components/FormularioIniciarSesion.jsx`

Función:

Maneja la autenticación de usuarios mediante **Firestore Authentication** o Google Sign-In. Controla el inicio de sesión y redirige al usuario al módulo principal.

Hooks usados:

- `useState` para manejar email/contraseña.
- `useNavigate` para redirección.

Dependencias:

- Firebase (auth).

Relaciones:

- Llamado desde `App.jsx`.
- Redirige a `Asistencia.jsx` al iniciar sesión correctamente.

Asistencia.jsx

Ruta: `/src/components/Asistencia.jsx`

Función:

Vista principal donde se visualizan las asistencias registradas.
Contiene la tabla y los botones de acciones (editar/eliminar/registrar).

Hooks usados:

- `useAsistencias()` (hook personalizado).
- `useEffect` para cargar los datos al montar.
- `useState` para manejar estados locales.

Dependencias:

- Firebase (lectura de datos).

Relaciones:

- Llamado desde `App.jsx`.
- Invoca a `BajaAsistencia.jsx` para eliminar registros.
- Invoca a `AltaAsistencia.jsx` para agregar registros.
- Invoca a `ModificarAsistencia.jsx` para modificar registros.

FormularioRegistrarse.jsx

Ruta: `/src/components/FormularioRegistrarse.jsx`

Función:

Permite que un usuario cree una nueva cuenta en el sistema. Muestra un formulario con campos de correo, contraseña y confirmación. Realiza validaciones básicas y registra al usuario en Firebase Authentication.

Hooks usados:

- `useState` para los campos del formulario.
- `useNavigate` para redirigir al login tras registrarse.
- `useUsuario` (para manejar la sesión de usuario).

Dependencias:

- Firebase Authentication (`createUserWithEmailAndPassword`).

Relaciones:

- Llamado desde `FormularioIniciarSesion.jsx` o desde una ruta específica `/registrarse`.

BajaAsistencia.jsx

Ruta: `/src/components/BajaAsistencia.jsx`

Función:

Permite eliminar un registro de asistencia de la base de datos. Muestra una confirmación antes de ejecutar la eliminación.

Hooks usados:

- `useAsistencias()`
- `useNavigate()`

Dependencias:

- Firebase (función de borrado).

AltaAsistencia.jsx

Ruta: `/src/components/AltaAsistencia.jsx`

Función:

Permite **registrar una nueva asistencia** en el sistema. Incluye un formulario donde se ingresan los datos del alumno (nombre, fecha, estado, etc.). Al confirmar, guarda la información en la base de datos mediante el hook `useAsistencias`.

Hooks usados:

- `useState` para manejar los campos del formulario.
- `useNavigate` para redirigir tras guardar.
- `useAsistencias` para acceder a la función `guardarAsistencia`.

Dependencias:

- Firebase (creación de registros).
- `FormularioAsistencia` (si está separado).

Relaciones:

- Llamado desde `App.jsx` o un botón en `Asistencia.jsx` "Nueva Asistencia".
- Interactúa directamente con el hook `useAsistencias`.

ModificarAsistencia.jsx

Ruta: `/src/components/ModificarAsistencia.jsx`

Función:

Componente que permite **editar un registro existente** de asistencia. Carga los datos seleccionados mediante un ID, los muestra en un formulario editable y actualiza el registro al guardar.

Hooks usados:

- `useParams` para obtener el ID del registro.
- `useAsistencias` para acceder a `obtenerAsistencia` y `actualizarAsistencia`.
- `useNavigate` para volver a la vista principal tras la modificación.

Dependencias:

- Firebase (actualización de datos).
- `FormularioAsistencia` (para reutilizar el mismo diseño del alta).

Relaciones:

- Llamado desde `Asistencia.jsx` mediante botón “Modificar”.

Notificaciones.jsx

Ruta: `/src/components/Notificaciones.jsx`

Función:

Muestra mensajes visuales de confirmación para permitir notificaciones.

Hooks usados:

- `useState` para manejar el estado visible del mensaje.
- `useEffect` para temporizar su desaparición.

PermisoCamara.jsx

Ruta: `/src/components/PermisoCamara.jsx`

Función:

Verifica y solicita permisos para usar la cámara del dispositivo (por ejemplo, si se usa realidad aumentada).

Hooks usados:

- `useEffect` para solicitar el permiso al montar el componente.
- `useState` para guardar el estado del permiso (concedido/denegado).

Relaciones:

- Puede ser usado antes de acceder al módulo de realidad aumentada o de escaneo.

Reconocimiento.jsx

Función:

Implementa el módulo principal de reconocimiento facial con `face-api.js`.

Carga los modelos necesarios desde una CDN y controla el flujo de captura, detección, y visualización de resultados.

Hooks usados:

- `useEffect`, `useRef`, `useState`, `useNavigate`
- Lógica de video con `navigator.mediaDevices.getUserMedia`
- Detección facial con `faceapi.detectAllFaces(...)`

Dependencias:

- `face-api.js`
- `react-router-dom`

CameraComponent.jsx

Función:

Componente simple que inicializa la cámara del usuario y muestra su flujo de video.

No realiza análisis ni guarda imágenes.

Hooks usados:

- `useRef` y `useEffect`

Dependencias:

- API `navigator.mediaDevices.getUserMedia`

ProtectedRoute.jsx

Ruta: `/src/components/ProtectedRoute.jsx`

Función:

Protege rutas que requieren autenticación.

Verifica si el usuario está logueado (a través de `useUsuario`).

Si no hay sesión activa, redirige al `FormularioIniciarSesion`.

Hooks usados:

- `useUsuario` para verificar el estado de sesión.
- `Navigate` de React Router para redirigir.

Dependencias:

- `react-router-dom`

Relaciones:

- Usado en `App.jsx` para proteger rutas como `/asistencias` o `/alta`.

useAsistencias.jsx

Ruta: `/src/hooks/useAsistencias.js`

Función:

Hook personalizado que abstrae toda la lógica de Firebase (CRUD).
Facilita obtener, agregar y eliminar asistencias sin repetir código.

Funciones principales:

- `agregarAsistencias()`
- `actualizarAsistencia()`
- `borrarAsistencia()`
- `fetchAsistencias ()`
- `reordenarNumeros()`

useUsuario.jsx

Ruta: `/src/hooks/useUsuario.js`

Función:

Hook personalizado que centraliza toda la lógica de **autenticación de usuario**.
Permite obtener el usuario actual, iniciar sesión, cerrar sesión y registrar nuevos usuarios.

Funciones principales:

- `iniciarSesion(email, password)`
- `cerrarSesion()`
- `registrarUsuario(email, password)`
- `usuario` (estado global del usuario actual).

Hooks usados:

- `useState` y `useEffect`.
- `onAuthStateChanged` de Firebase.

Dependencias:

- Firebase Authentication.

Relaciones:

- Usado por [Login](#), [FormularioRegistrarse](#), y [ProtectedRoute](#).

4. Flujo general de datos

- Usuario interactúa con la UI.
- El componente ejecuta la función del hook.
- Hook actualiza datos en Firebase.
- Estado React actualiza la vista automáticamente.

5. Estilo y convenciones

- Archivos CSS separados por componente.
- Comentarios tipo JavaDoc para cada función.

6. Deploy

1. Ejecutar `npm run build` para generar la versión productiva.
2. Subir la carpeta `/dist` a un hosting como **Vercel**, **Netlify** o **Firebase Hosting**.
3. Verificar que las rutas funcionen correctamente.

Créditos

- **Autor:** Gastón Frigo
- **Profesor:** Exequiel Wiedermann
- **Materia:** Programación Web Dinámica
- **Institución:** EPET 20
- **Creadores del sitio web:** Gastón Frigo, Joaquín Lema, Facundo Carrasco, Mateo Acuña y Aquiles Font