Documento Técnico - Sistema de Asistencia ITS Cipolletti

Información del Proyecto

Proyecto: Sistema de Registro de Asistencias para Profesores
Institución: Instituto Técnico Superior (ITS) Cipolletti
Tecnología: Angular + Ionic + Firebase
Tipo: Aplicación Web Progresiva (PWA)
Versión: 1.0.0
Fecha: Mayo 2025

Tabla de Contenidos

---

1. Resumen Ejecutivo

1.1 Propósito del Sistema

El Sistema de Asistencia ITS Cipolletti es una aplicación web progresiva diseñada para digitalizar y automatizar el proceso de registro de asistencias de profesores. El sistema permite a los docentes registrar sus entradas y salidas mediante validación geográfica y captura fotográfica, proporcionando un control preciso y transparente de la asistencia laboral.

1.2 Objetivos Principales

- **Automatización:** Eliminar registros manuales de asistencia
- **Validación:** Asegurar que los registros se realicen desde el campus
- **Transparencia:** Proporcionar historial detallado y estadísticas
- **Eficiencia:** Reducir tiempos administrativos
- **Trazabilidad:** Mantener registro fotográfico y geográfico

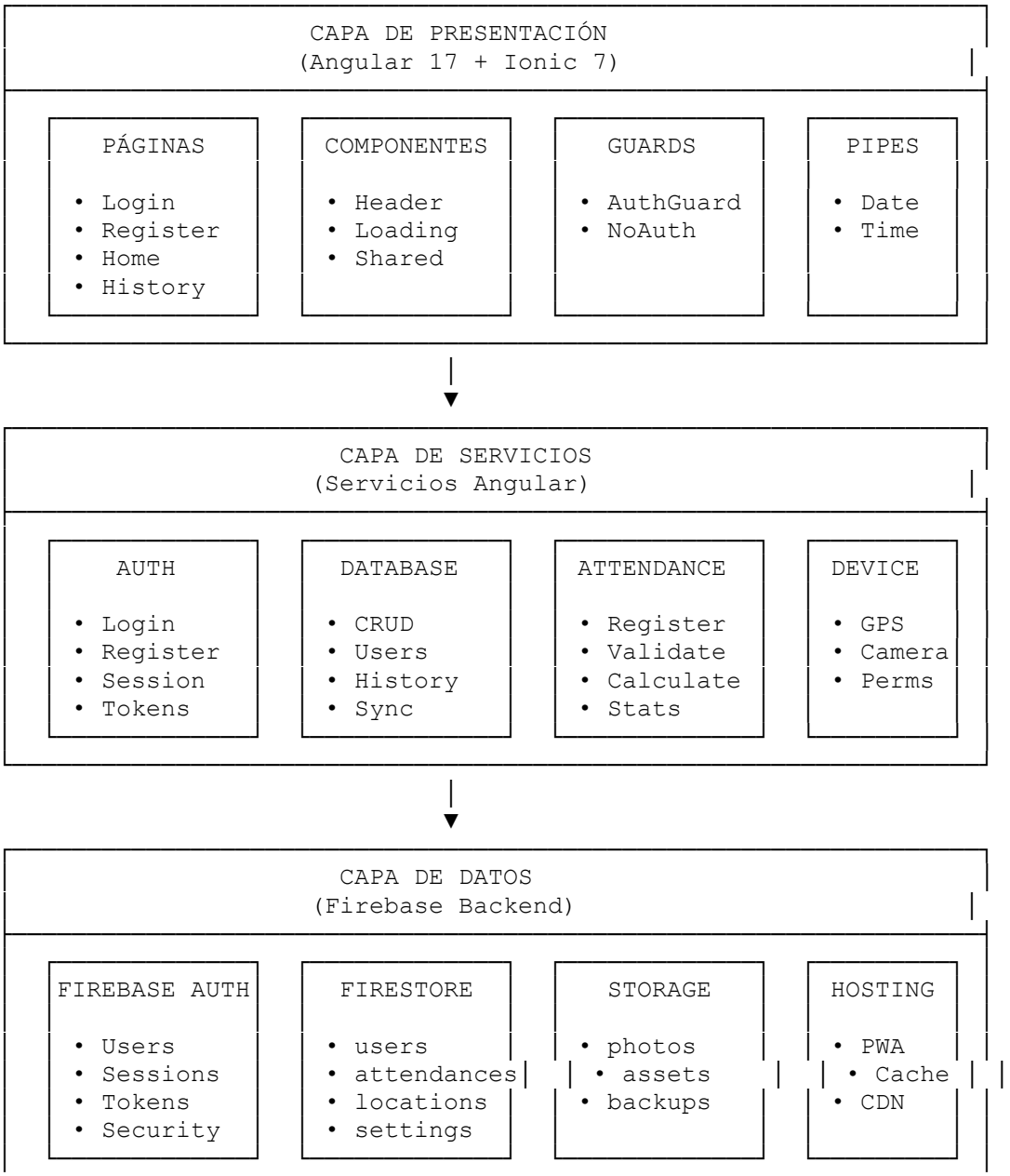
1.3 Beneficios Clave

- Control automático de ubicación (GPS)
- Registro fotográfico para verificación
- Historial completo con estadísticas
- Interfaz moderna y responsive
- Funcionamiento offline básico
- Integración con sistema académico existente

---

2. Arquitectura del Sistema

2.1 Arquitectura General



└──┘

2.2 Arquitectura de Componentes

Frontend (Angular + Ionic)

- **Framework:** Angular 17 con Standalone Components
- **UI:** Ionic 7 para componentes móviles
- **Estado:** RxJS BehaviorSubjects para manejo reactivo
- **Routing:** Angular Router con Guards de seguridad
- **PWA:** Service Worker para funcionamiento offline

Backend (Firebase)

- **Autenticación:** Firebase Auth con email/password
- **Base de Datos:** Firestore (NoSQL) para datos estructurados
- **Almacenamiento:** Firebase Storage para imágenes
- **Hosting:** Firebase Hosting para deployment
- **Reglas:** Security Rules para protección de datos

Servicios del Dispositivo

- **Geolocalización:** Capacitor Geolocation + Web Geolocation API
- **Cámara:** Capacitor Camera + Web Camera API
- **Permisos:** Gestión automática de permisos nativos

2.3 Patrones Arquitectónicos Implementados

Patrón de Servicios

- **AuthService:** Gestión centralizada de autenticación
- **DatabaseService:** Operaciones CRUD unificadas
- **AttendanceService:** Lógica de negocio de asistencias
- **GeolocationService:** Abstracción de servicios de ubicación

Patrón Observer (RxJS)

- **BehaviorSubjects:** Estado reactivo para usuario actual
- **Observables:** Flujo de datos en tiempo real
- **Subscriptions:** Gestión automática de memoria

Patrón Guard

- **AuthGuard:** Protección de rutas autenticadas
- **NoAuthGuard:** Redirección de usuarios logueados

---

3. Modelo de Datos

3.1 Estructura de la Base de Datos (Firestore)

Colección: `users`

```
interface User {
  id?: string;                    // Documento ID
  email: string;                  // Email institucional
  nombre: string;                 // Nombre del profesor
  apellido: string;               // Apellido del profesor
  telefono?: string;              // Teléfono de contacto
  employeeId?: string;            // ID de empleado institucional
  carrera?: string;               // Carrera asignada
  materia?: string;               // Materia que dicta
  fechaCreacion?: Date;           // Fecha de registro
  activo?: boolean;               // Estado del usuario
  uid?: string;                   // Firebase Auth UID
}
```

Colección: `attendances`

```
interface Attendance {
  id?: string;                    // Documento ID
  userId: string;                 // Referencia al usuario
  tipo: 'entrada' | 'salida';     // Tipo de registro
  fecha: Date;                    // Fecha del registro
  hora: string;                   // Hora en formato HH:MM
  ubicacion: {                    // Coordenadas GPS
    latitud: number;
    longitud: number;
  };
  fotoUrl: string;                // URL de la foto tomada
  ubicacionValida: boolean;       // Si está dentro del radio
permitido
}
```

Colección: `locations`

```
interface ValidLocation {
  id?: string;                    // Documento ID
  nombre: string;                 // Nombre del campus/edificio
  latitud: number;                // Latitud del centro
  longitud: number;               // Longitud del centro
  radioPermitido: number;         // Radio en metros
}
```

3.2 Configuración de Carreras y Materias

Enum de Carreras

```
enum Carrera {
  DESARROLLO_FULLSTACK = 'Tecnicatura Superior en Desarrollo de
Software Full Stack',
  DEVOPS = 'Tecnicatura Superior en DevOps',
  INFORMATICA = 'Tecnicatura Superior en Informática'
}
```

Materias por Carrera

Desarrollo Full Stack:

- Inglés Técnico I, II, III
- Matemática
- Laboratorio Full Stack I, II
- Arquitectura de las Computadoras
- Programación, Backend, Frontend
- Base de Datos
- Diseño UX/UI
- Práctica Profesionalizante

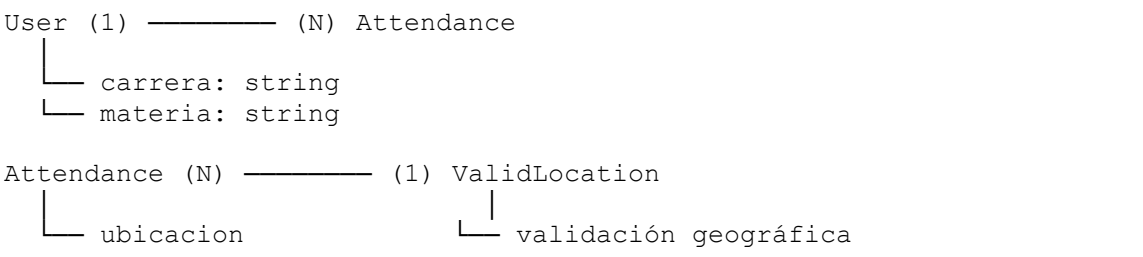
DevOps:

- Inglés I, II, III
- Cultura DevOps y Adopción
- Metodologías Ágiles
- Control de Versiones
- Aplicaciones Cloud Nativas
- Sistemas Operativos
- Automatización y Scripting
- Laboratorio I, II, III

Informática:

- Inglés Técnico I, II, III
- Tecnología, Ciencia y Sociedad
- Arquitectura de las Computadoras
- Matemática I, II
- Laboratorio de Informática I, II, III
- Sistemas y Organizaciones

3.3 Relaciones entre Entidades



---

4. Casos de Uso

4.1 Actores del Sistema

Actor Principal: Profesor

- **Descripción:** Docente del ITS Cipolletti
- **Responsabilidades:** Registrar entrada/salida, consultar historial

- **Permisos:** Acceso a sus propios registros

Actor Sistema: Sistema Automático

- **Descripción:** Lógica automatizada del sistema
- **Responsabilidades:** Validar ubicación, capturar fotos, calcular estadísticas

4.2 Casos de Uso Principales

CU-001: Registro de Usuario

Actor: Profesor
Descripción: Un profesor se registra en el sistema por primera vez

Precondiciones:

- El profesor tiene email institucional válido
- No existe cuenta previa con ese email

Flujo Principal:

1. El profesor accede a la pantalla de registro
2. Completa información personal (nombre, apellido, email)
3. Selecciona carrera de las opciones disponibles
4. Selecciona materia específica según la carrera elegida
5. Ingresa contraseña segura y confirmación
6. El sistema valida formato de email y fortaleza de contraseña
7. Se crea la cuenta en Firebase Auth
8. Se guardan los datos del perfil en Firestore
9. Se muestra mensaje de confirmación exitosa
10. Se redirige automáticamente a pantalla de login

Flujos Alternativos:

- **6a. Datos inválidos:** Se muestran errores específicos de validación
- **7a. Email ya registrado:** Se informa que la cuenta ya existe
- **8a. Error de red:** Se muestra mensaje de error y opción de reintentar

Postcondiciones:

- Usuario creado en Firebase Auth
- Perfil guardado en Firestore
- Usuario puede iniciar sesión

CU-002: Iniciar Sesión

Actor: Profesor
Descripción: Un profesor accede al sistema con credenciales válidas

Precondiciones:

- El profesor tiene cuenta registrada
- Las credenciales son correctas

Flujo Principal:

1. El profesor ingresa email y contraseña
2. El sistema valida formato de credenciales
3. Firebase Auth autentica las credenciales
4. Se obtienen datos del perfil desde Firestore
5. Se inicializa sesión local con tokens
6. Se redirige a pantalla principal (Home)

Flujos Alternativos:

- **2a. Formato inválido:** Se muestran errores de validación
- **3a. Credenciales incorrectas:** Se muestra error de autenticación
- **4a. Error de conexión:** Se intenta usar cache local si está disponible

Postcondiciones:

- Usuario autenticado en sesión
- Tokens de acceso almacenados
- Acceso a funcionalidades protegidas

CU-003: Registrar Entrada

Actor: Profesor
Descripción: Un profesor registra su llegada al instituto

Precondiciones:

- Usuario autenticado en el sistema
- No ha registrado entrada en el día actual
- Dispositivo con permisos de ubicación y cámara

Flujo Principal:

1. El profesor presiona botón "Registrar Entrada"
2. El sistema solicita permisos de ubicación si no están concedidos
3. Se obtiene ubicación GPS actual del dispositivo
4. Se valida que la ubicación esté dentro del radio permitido del ITS
5. El sistema solicita permisos de cámara si no están concedidos
6. Se captura una fotografía del profesor
7. Se crea registro de asistencia con timestamp actual
8. Se guarda en Firestore con toda la información
9. Se muestra confirmación con detalles del registro
10. Se actualiza la interfaz con el nuevo estado

Flujos Alternativos:

- **2a. Permisos de ubicación denegados:** Se usa ubicación por defecto y se marca como inválida
- **4a. Ubicación fuera de rango:** Se registra pero se marca como ubicación inválida
- **5a. Permisos de cámara denegados:** Se continúa sin foto
- **6a. Error de cámara:** Se registra sin imagen
- **8a. Error de red:** Se guarda localmente para sincronizar después

Postcondiciones:

- Registro de entrada guardado
- Estado de asistencia actualizado
- Profesor puede registrar salida posteriormente

CU-004: Registrar Salida

Actor: Profesor
Descripción: Un profesor registra su salida del instituto

Precondiciones:

- Usuario autenticado en el sistema
- Ha registrado entrada previamente en el día
- No ha registrado salida en el día actual

Flujo Principal:

1. El profesor presiona botón "Registrar Salida"
2. Se ejecuta el mismo proceso de validación que en entrada
3. Se calcula automáticamente las horas trabajadas
4. Se muestra resumen completo del día laboral

Flujos Alternativos:

- Similares a CU-003 (Registrar Entrada)

Postcondiciones:

- Registro de salida guardado
- Horas trabajadas calculadas
- Día laboral completado

CU-005: Consultar Historial

Actor: Profesor
Descripción: Un profesor consulta su historial de asistencias

Precondiciones:

- Usuario autenticado en el sistema

- Existen registros previos de asistencia

Flujo Principal:

1. El profesor accede a la sección "Historial"
2. El sistema carga todas las asistencias del usuario
3. Se muestran registros agrupados por fecha (más recientes primero)
4. Se pueden aplicar filtros por período (semana, mes, todos)
5. Se muestran estadísticas del período seleccionado
6. El profesor puede ver detalles de cualquier registro específico

Flujos Alternativos:

- **2a. No hay registros:** Se muestra mensaje informativo
- **6a. Error cargando detalles:** Se muestra mensaje de error específico

Postcondiciones:

- Historial visualizado correctamente
- Estadísticas calculadas y mostradas

CU-006: Buscar en Historial

Actor: Profesor
Descripción: Un profesor busca registros específicos en su historial

Precondiciones:

- Usuario en pantalla de historial
- Existen registros para buscar

Flujo Principal:

1. El profesor presiona el botón de búsqueda en el header
2. Se muestra diálogo de búsqueda
3. Ingresa término de búsqueda (fecha, hora, tipo, etc.)
4. El sistema filtra registros en tiempo real
5. Se muestran solo los registros que coinciden
6. Se mantienen las estadísticas actualizadas para la búsqueda

Postcondiciones:

- Resultados filtrados mostrados
- Posibilidad de limpiar búsqueda

CU-007: Validación Automática de Ubicación

Actor: Sistema
Descripción: El sistema valida automáticamente si el registro se hace desde el campus

Precondiciones:

- Ubicaciones válidas configuradas en el sistema
- Coordenadas GPS obtenidas del dispositivo

Flujo Principal:

1. Se obtienen coordenadas GPS del dispositivo
2. Se cargan todas las ubicaciones válidas desde Firestore
3. Se calcula distancia a cada ubicación usando fórmula de Haversine
4. Se determina la ubicación válida más cercana
5. Se verifica si la distancia está dentro del radio permitido
6. Se marca el registro como válido o inválido según resultado

Flujos Alternativos:

- **1a. Error obteniendo GPS:** Se marca como ubicación inválida
- **2a. No hay ubicaciones configuradas:** Se permite por defecto

Postcondiciones:

- Estado de ubicación determinado
- Registro marcado apropiadamente

---

5. Documentación de Pantallas

5.1 Pantalla de Login

Descripción General

Pantalla de autenticación donde los profesores ingresan sus credenciales para acceder al sistema.

Elementos de Interfaz

- **Header simplificado:** Solo título sin botones de acción
- **Formulario de login:**
  - Campo email con validación de formato
  - Campo contraseña con validación de longitud mínima
  - Botón "Iniciar Sesión" con indicador de carga
- **Enlaces de navegación:**
  - Link a pantalla de registro
  - Mensajes de error contextuales

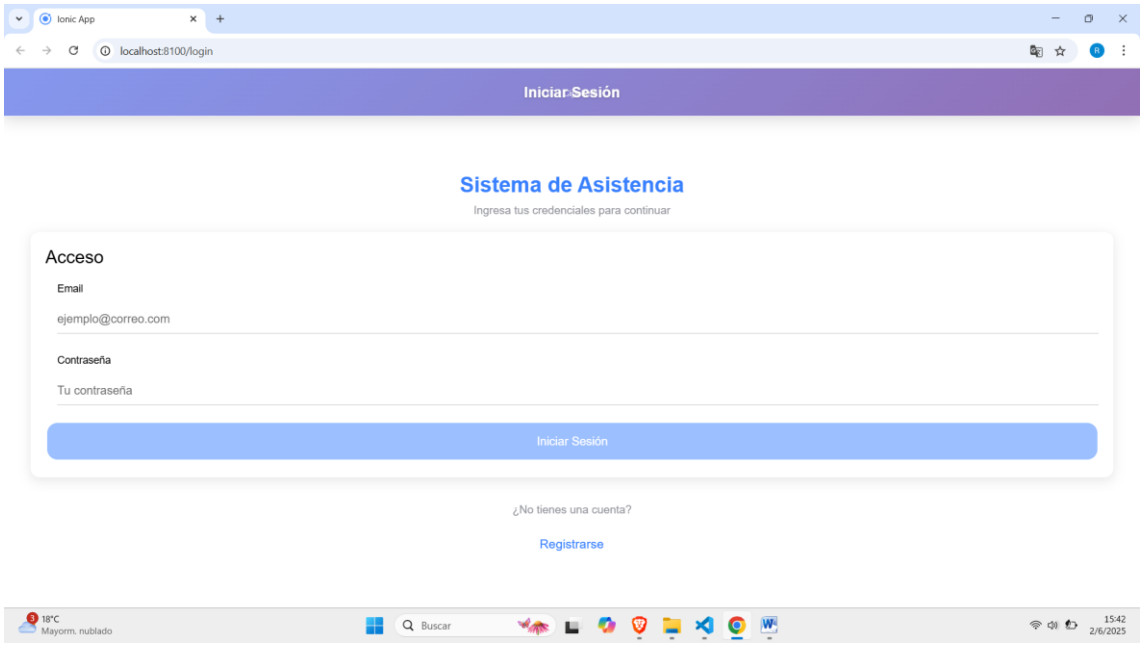
Validaciones Implementadas

- Email: requerido, formato válido

- Contraseña: requerida, mínimo 6 caracteres
- Mensajes de error específicos por tipo de validación

Estados de la Pantalla

- **Estado inicial:** Formulario vacío
- **Estado de carga:** Spinner durante autenticación
- **Estado de error:** Mensajes de error específicos
- **Estado de éxito:** Redirección automática



Flujo de Navegación

```
Login → [Credenciales válidas] → Home
Login → [Registro] → Register
Login → [Error] → Login (con mensaje)
```

5.2 Pantalla de Registro

Descripción General

Formulario completo para que nuevos profesores se registren en el sistema con toda su información académica.

Secciones del Formulario

Información Personal

- Nombre (requerido, solo letras)
- Apellido (requerido, solo letras)
- Email institucional (requerido, formato email)
- Teléfono (opcional, formato numérico)
- ID Empleado (opcional, alfanumérico mayúsculas)

Información Académica

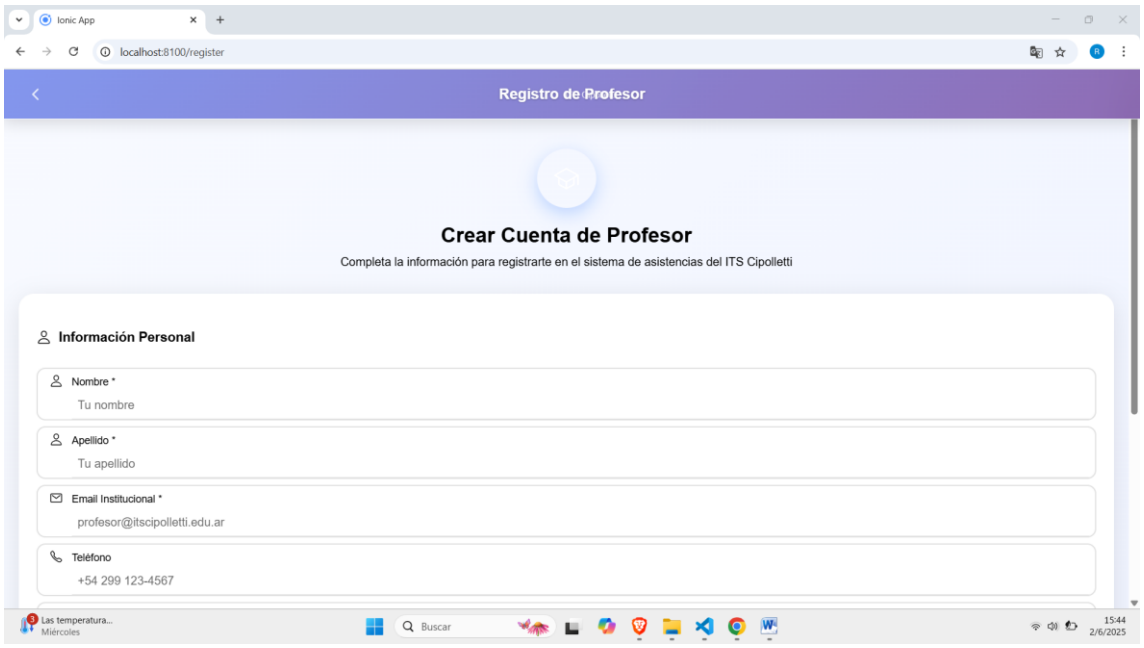
- Carrera (requerido, lista desplegable)
- Materia (requerido, dependiente de carrera seleccionada)

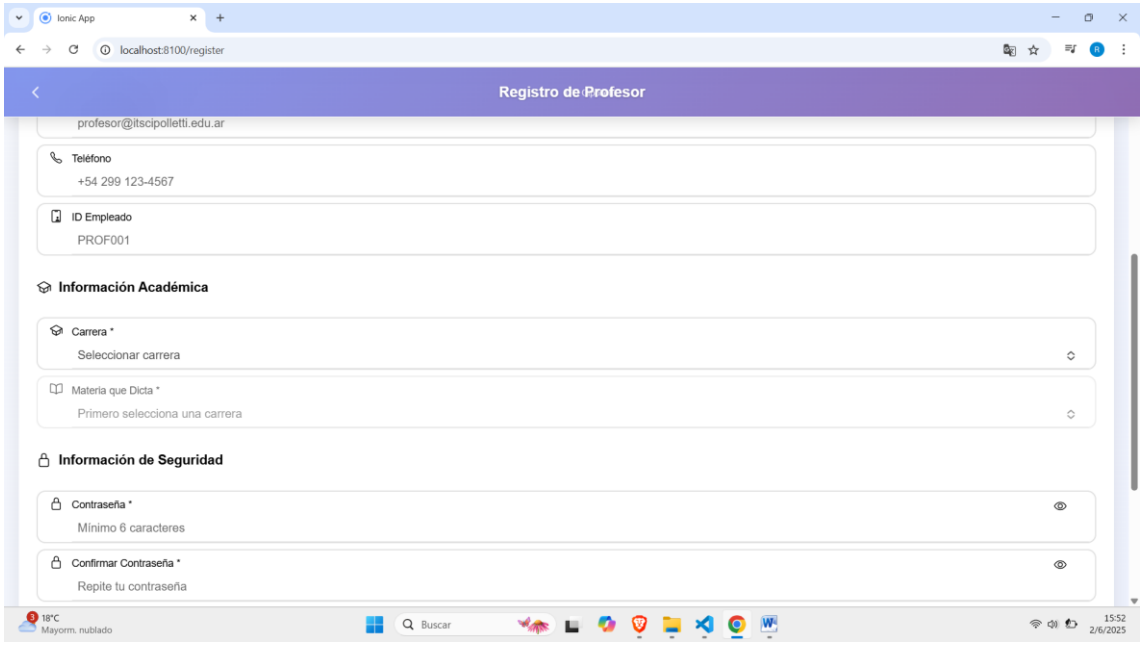
Información de Seguridad

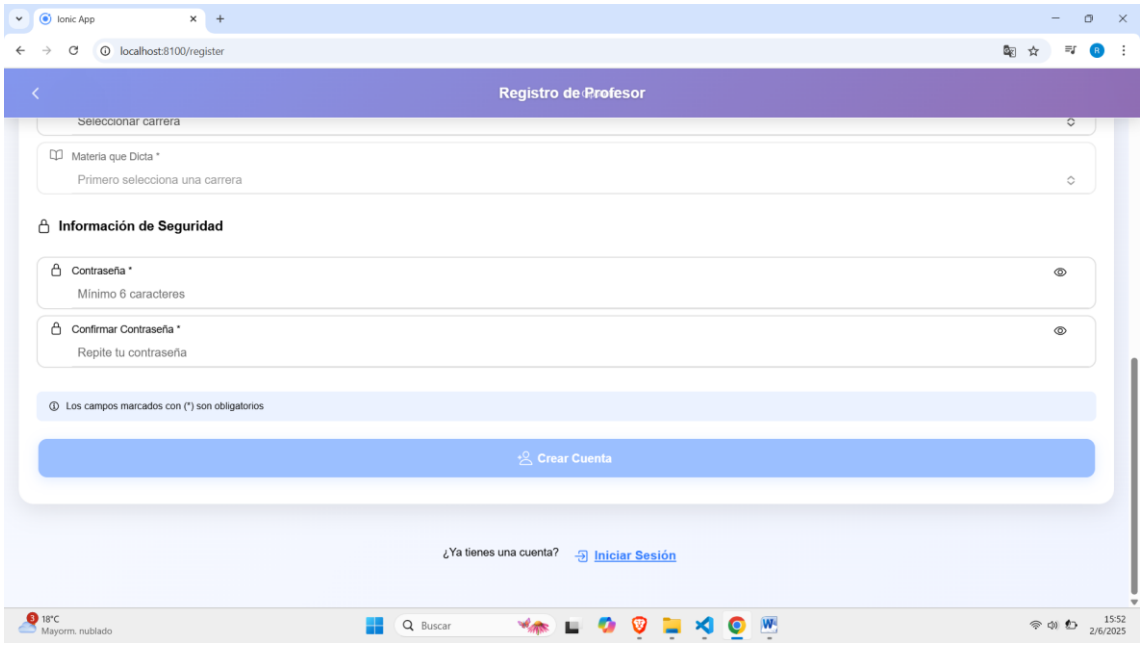
- Contraseña (requerido, mínimo 6 caracteres)
- Confirmar contraseña (requerido, debe coincidir)
- Botones para mostrar/ocultar contraseñas

Funcionalidades Especiales

- **Selección dependiente:** Materias se filtran según carrera
- **Validación en tiempo real:** Errores mostrados al escribir
- **Indicadores visuales:** Campos válidos/inválidos marcados
- **Prevención de envío:** Botón deshabilitado si formulario inválido







Validaciones por Campo

```
// Ejemplo de validaciones implementadas
nombre: [requerido, mínimo 2 caracteres, solo letras]
email: [requerido, formato email válido]
carrera: [requerido, debe estar en lista]
materia: [requerido, debe corresponder a carrera]
password: [requerido, mínimo 6 caracteres]
confirmPassword: [requerido, debe coincidir con password]
```

5.3 Pantalla Principal (Home)

Descripción General

Pantalla principal del sistema donde los profesores pueden ver su estado actual y registrar entrada/salida.

Header Moderno

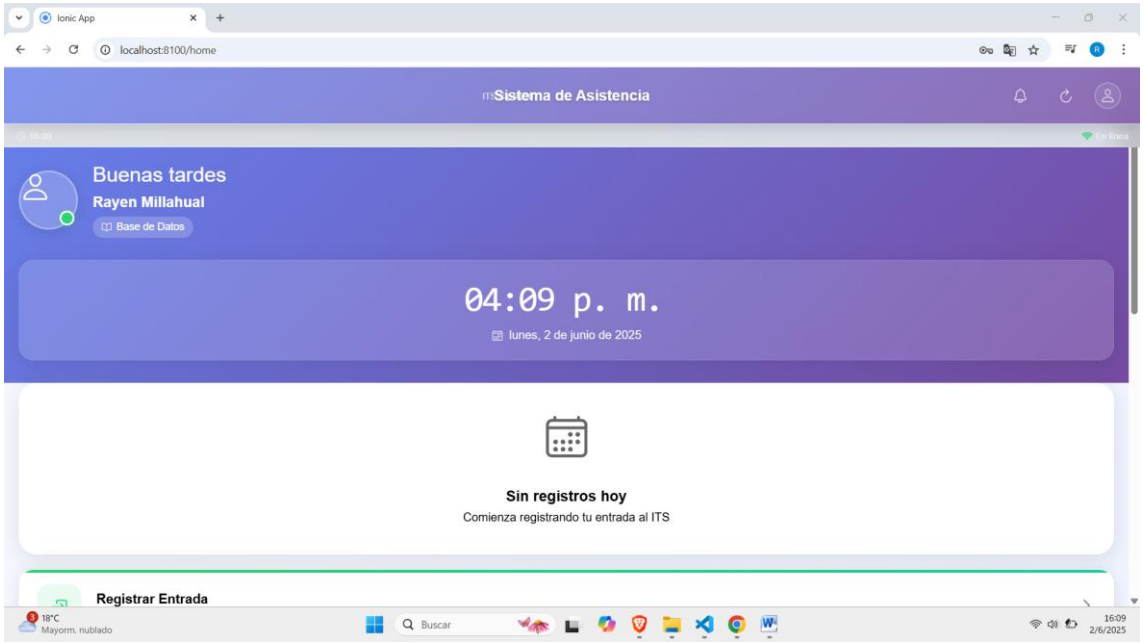
- **Título dinámico:** "Sistema de Asistencia - ITS Cipolletti"
- **Información de usuario:** Avatar, menú desplegable
- **Controles:** Notificaciones, refresh, configuración
- **Estado de conexión:** Indicador online/offline
- **Reloj en tiempo real:** Hora actual actualizada

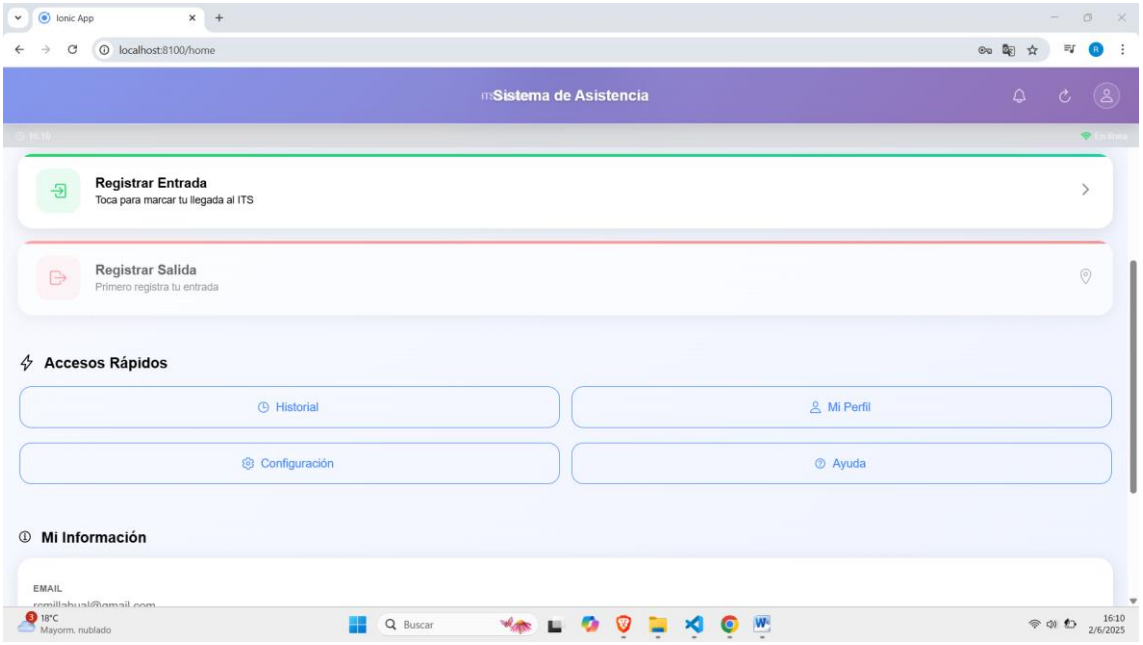
Sección de Bienvenida

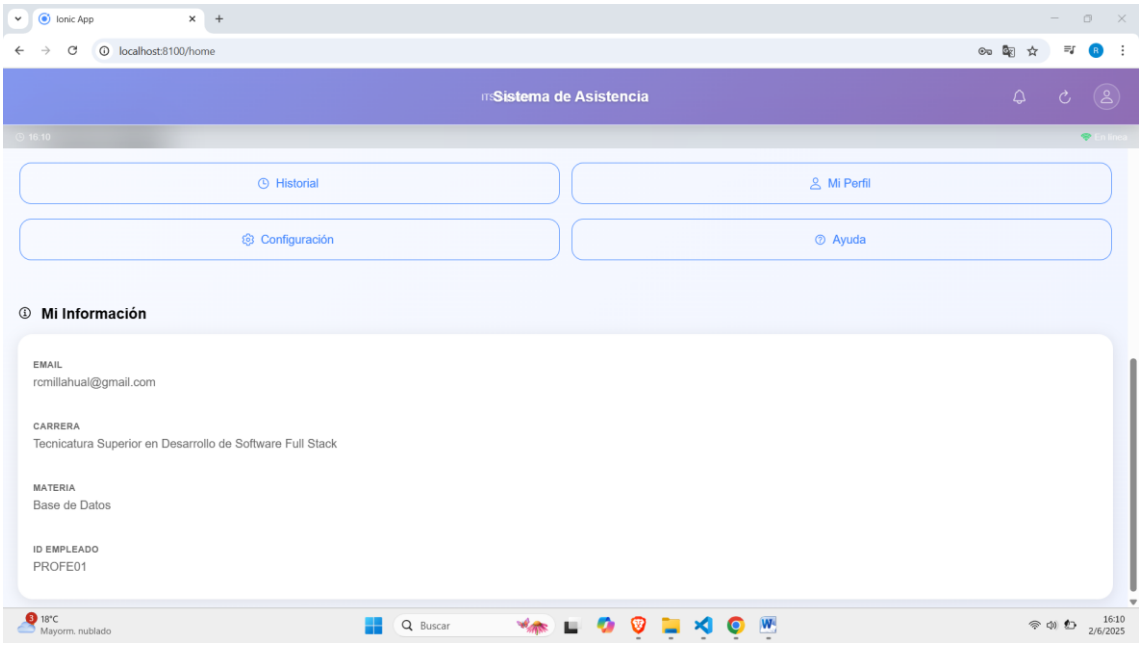
- **Saludo personalizado:** Buenos días/tardes/noches
- **Información del usuario:** Nombre completo, materia asignada
- **Reloj digital:** Hora actual con segundos animados
- **Fecha completa:** Día de la semana, fecha completa

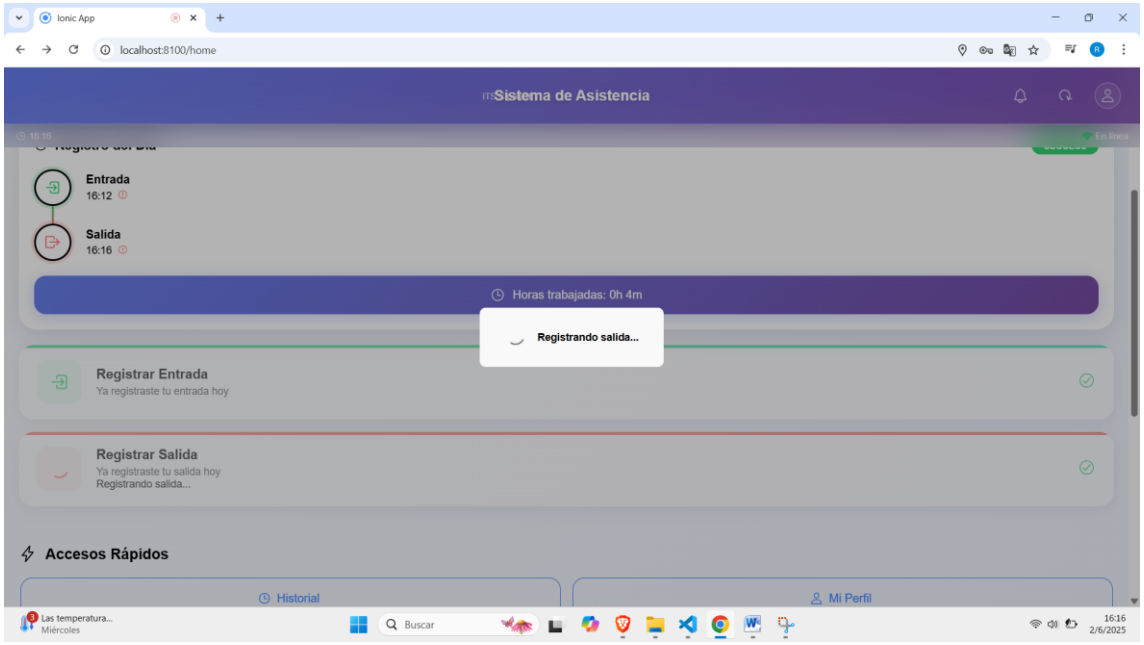
Estado de Asistencias del Día

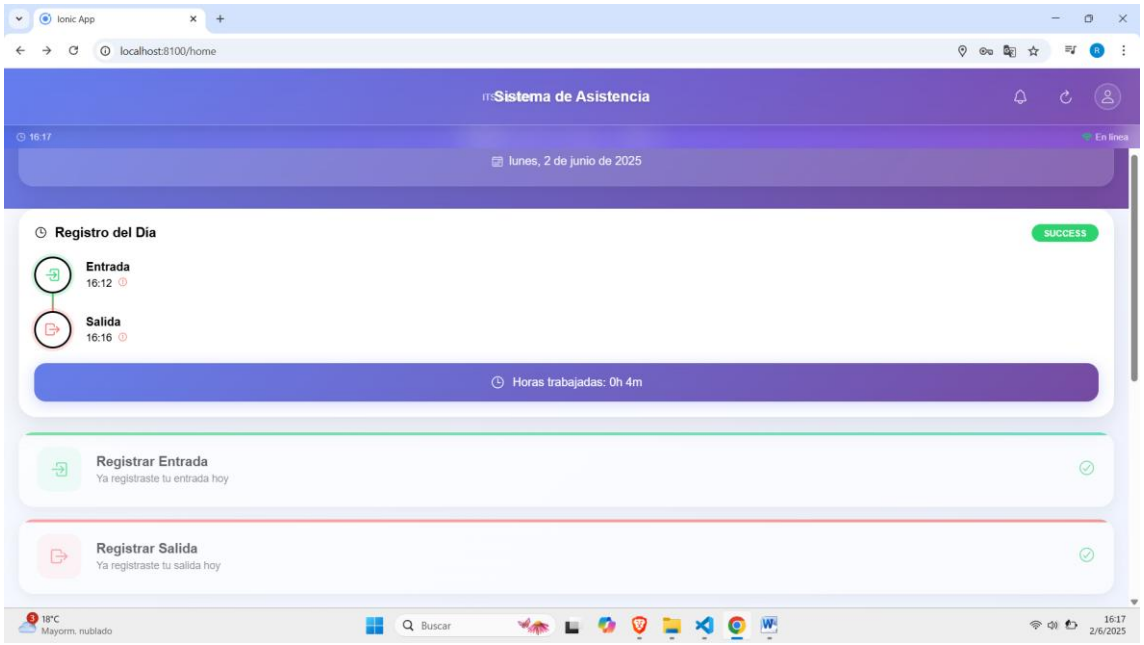
- **Timeline visual:** Progreso entrada → salida
- **Información de registros:**
  - Hora de entrada (si existe)
  - Hora de salida (si existe)
  - Estado de ubicación para cada registro
  - Horas trabajadas (si ambos registros existen)




Ionic App
localhost:8100/home
Sistema de Asistencia
Registrar Entrada
Toca para marcar tu llegada al ITS
Registrar Salida
Primero registra tu entrada
Accesos Rápidos
Historial
Mi Perfil
Configuración
Ayuda
Mi Información
EMAIL


Ionic App
localhost:8100/home
Sistema de Asistencia
Historial
Mi Perfil
Configuración
Ayuda
Mi Información
EMAIL
rcmillahual@gmail.com
CARRERA
Tecnicatura Superior en Desarrollo de Software Full Stack
MATERIA
Base de Datos
ID EMPLEADO
PROFE01





Botones de Acción Principales

- **Registrar Entrada:**
  - Estado habilitado: Si no hay entrada del día
  - Estado deshabilitado: Si ya se registró entrada
  - Indicador de carga durante proceso
- **Registrar Salida:**
  - Estado habilitado: Si hay entrada pero no salida
  - Estado deshabilitado: Si no hay entrada o ya hay salida
  - Indicador de carga durante proceso

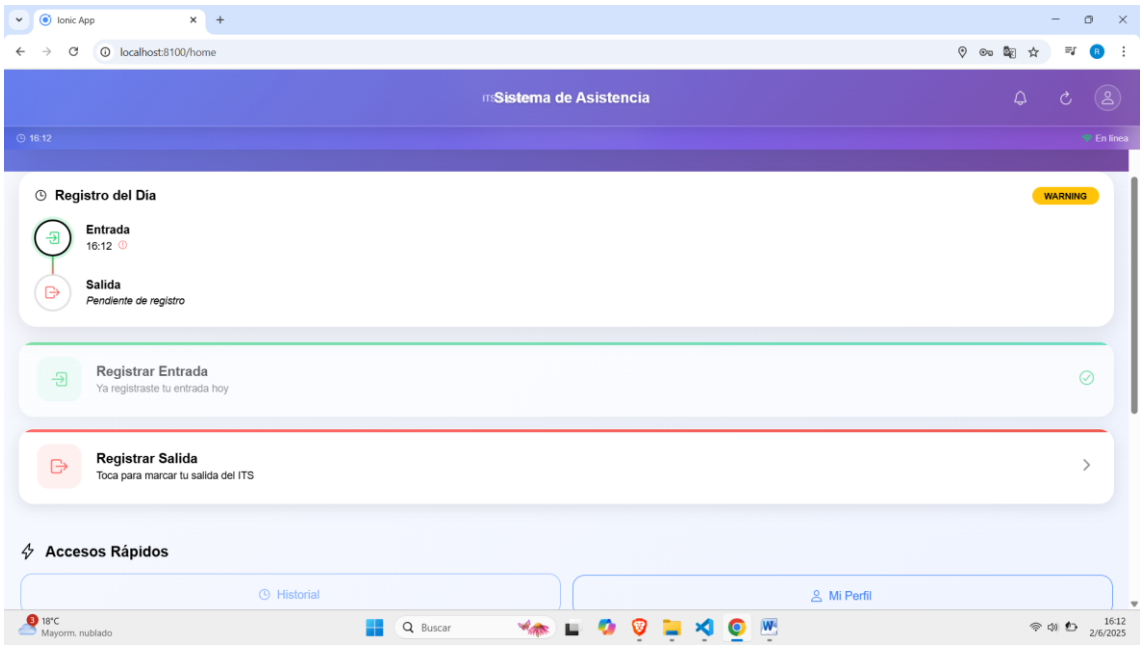
Accesos Rápidos

- **Grid de botones:** 2x2 en móvil, horizontal en desktop

- **Opciones disponibles:**
  - Ver Historial → Navegación a pantalla de historial
  - Mi Perfil → (Funcionalidad futura)
  - Configuración → (Funcionalidad futura)
  - Ayuda → Diálogo con información del sistema

Información del Usuario

- **Tarjeta desplegable** con datos del perfil:
  - Email institucional
  - Carrera asignada
  - Materia que dicta
  - ID de empleado (si existe)



Estados Dinámicos de la Pantalla

Sin Registros del Día

- Mensaje motivacional
- Solo botón de entrada habilitado
- Timeline vacío

Con Entrada Registrada

- Timeline con entrada marcada
- Botón de salida habilitado
- Indicador de tiempo transcurrido

Día Completo (Entrada + Salida)

- Timeline completo
- Cálculo de horas trabajadas
- Ambos botones deshabilitados
- Resumen del día

5.4 Pantalla de Historial

Descripción General

Pantalla donde los profesores pueden consultar todo su historial de asistencias con filtros y estadísticas.

Header Contextual

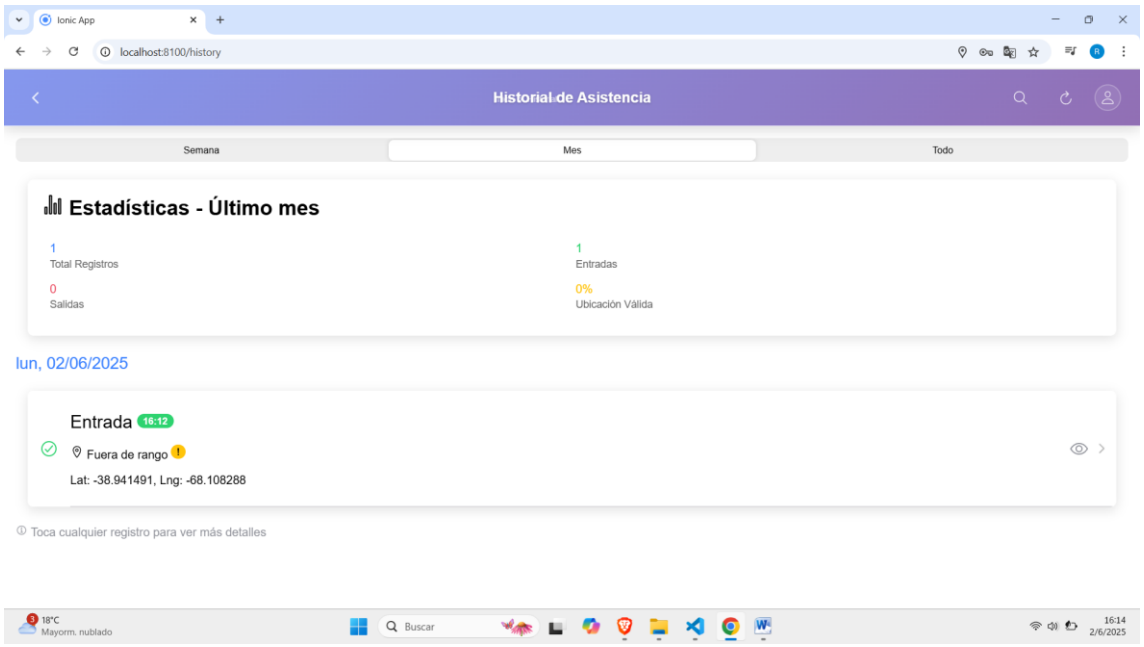
- **Breadcrumb:** "Inicio > Historial"
- **Botón de retroceso:** Navegación a Home
- **Búsqueda:** Filtrado por texto
- **Refresh:** Actualización manual
- **Barra de progreso:** Durante carga de datos

Filtros de Período

- **Segmento de selección:**
  - Última semana
  - Último mes
  - Todos los registros
- **Aplicación automática:** Cambio inmediato al seleccionar

Tarjeta de Estadísticas

- **Métricas del período seleccionado:**
  - Total de registros
  - Número de entradas
  - Número de salidas
  - Porcentaje de ubicaciones válidas



Lista de Asistencias

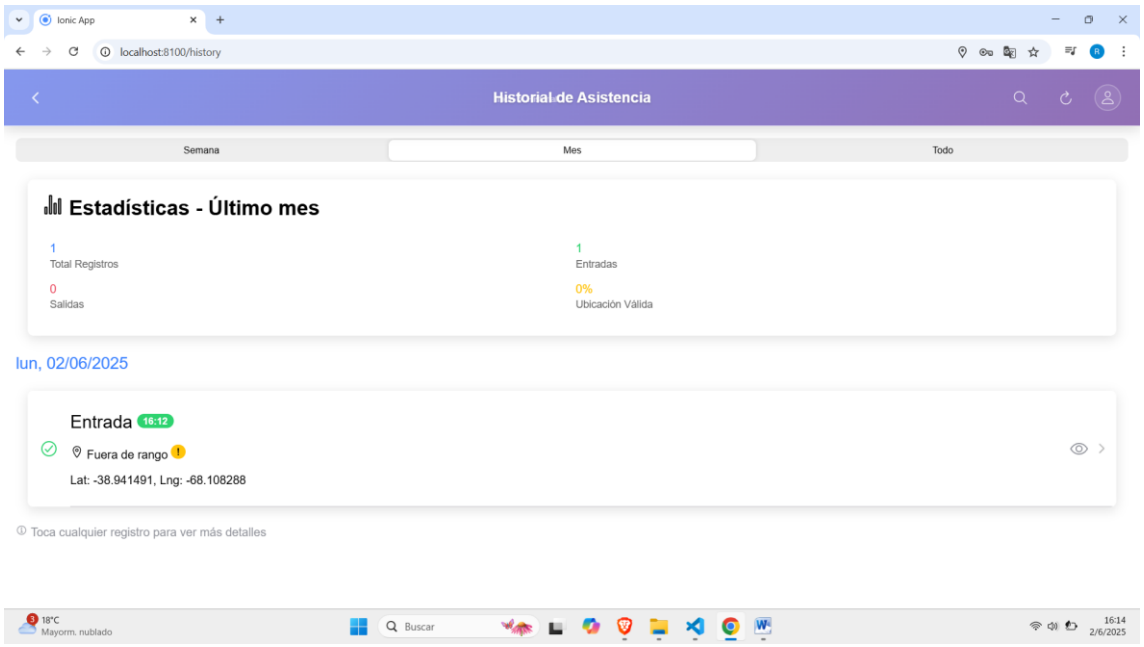
- **Agrupación por fecha:** Registros organizados por día
- **Información por registro:**
  - Tipo (entrada/salida) con iconos colorados
  - Hora exacta en badge
  - Estado de ubicación con indicadores
  - Coordenadas GPS precisas
- **Interactividad:** Tap para ver detalles completos

Funcionalidad de Búsqueda

- **Búsqueda por texto libre:**
  - Fecha en cualquier formato
  - Hora (ej: "08:30")
  - Tipo ("entrada", "salida")
  - Estado ("válida", "fuera de rango")
- **Resultados en tiempo real**
- **Opción de limpiar búsqueda**

Detalles de Registro Individual

- **Modal informativo con:**
  - Fecha y hora completas
  - Coordenadas GPS exactas
  - Estado de ubicación detallado
  - Información de foto (si existe)
  - Opciones para compartir/copiar detalles



Estados de la Pantalla

Cargando Datos

- Spinner central con mensaje
- Barra de progreso en header
- Botones deshabilitados

Sin Registros

- Mensaje informativo central
- Icono descriptivo
- Sugerencia de acción

Con Datos

- Lista completa de registros
- Estadísticas calculadas
- Filtros funcionales

5.5 Componente Header (Transversal)

Descripción General

Componente reutilizable que proporciona navegación y funcionalidades comunes a todas las pantallas.

Configurabilidad por Pantalla

- **Elementos mostrados/ocultos según contexto:**
  - Botón de retroceso (solo en pantallas secundarias)
  - Menú hamburguesa (solo en pantalla principal)

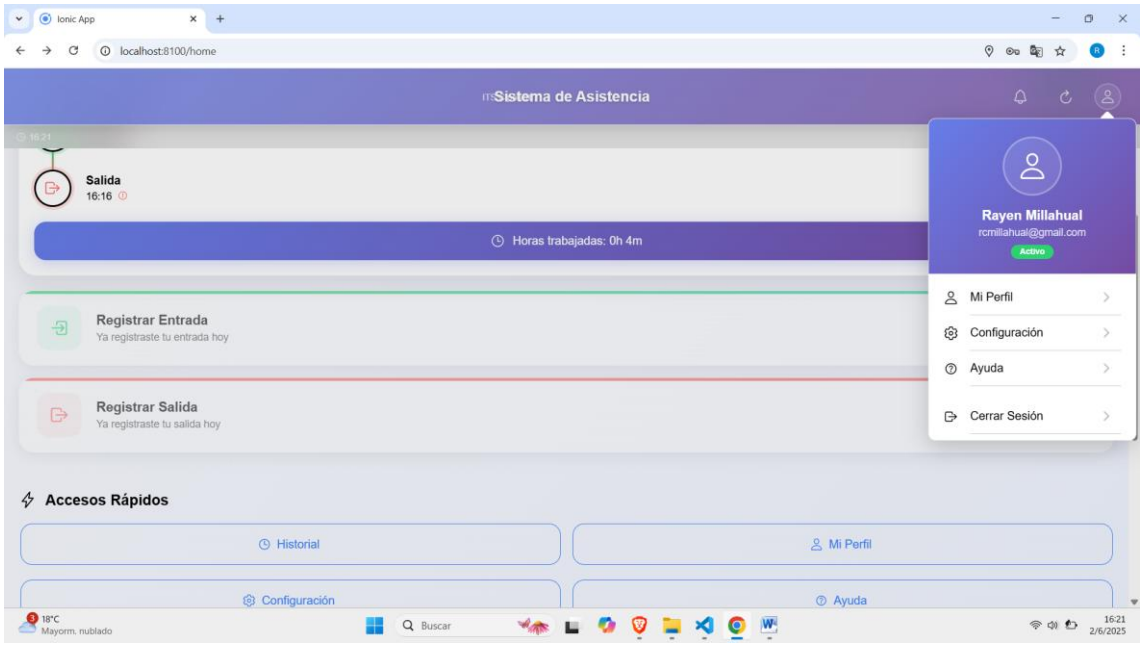
- Búsqueda (solo donde aplica)
- Notificaciones (solo para usuarios logueados)
- Barra de progreso (durante operaciones largas)

Menú de Usuario

- **Avatar personalizable** (con fallback a icono)
- **Información contextual:**
  - Nombre completo del usuario
  - Email institucional
  - Badge de estado (Activo)
- **Opciones del menú:**
  - Mi Perfil (navegación futura)
  - Configuración (navegación futura)
  - Ayuda (diálogo informativo)
  - Cerrar Sesión (con confirmación)

Indicadores de Estado

- **Conexión a internet:** Icono wifi/offline
- **Hora actual:** Actualizada en tiempo real
- **Notificaciones:** Badge con contador
- **Operaciones:** Spinner en botón de refresh



Adaptabilidad Responsive

- **Desktop:** Todos los elementos visibles
- **Tablet:** Elementos principales visibles
- **Móvil:** Elementos esenciales, menús colapsados

---

6. Servicios y Componentes

6.1 Servicios Principales

AuthService

Propósito: Gestión centralizada de autenticación y sesiones de usuario

Responsabilidades:

- Autenticación con Firebase Auth
- Manejo de estado de usuario (BehaviorSubject)
- Persistencia de sesión
- Renovación automática de tokens
- Logout seguro

Métodos principales:

```
login(email: string, password: string): Promise<User>
register(userData: User): Promise<User>
logout(): Promise<void>
currentUser: Observable<User | null>
isAuthenticated(): boolean
refreshUserData(): Promise<void>
```

DatabaseService

Propósito: Abstracción de operaciones con Firestore

Responsabilidades:

- Operaciones CRUD en Firestore
- Manejo de conexiones y errores
- Configuración de persistencia
- Gestión de colecciones

Métodos principales:

```
// Usuarios
registerUser(userData: User): Promise<User>
loginUser(email: string, password: string): Promise<User>
logoutUser(): Promise<void>

// Asistencias
saveAttendance(attendance: Attendance): Promise<Attendance>
getUserAttendances(userId?: string): Promise<Attendance[]>
hasUserRegisteredToday(userId: string, tipo: string): Promise<boolean>

// Ubicaciones
getLocations(): Promise<ValidLocation[]>
addLocation(location: ValidLocation): Promise<void>
```

AttendanceService

Propósito: Lógica de negocio para registro de asistencias

Responsabilidades:

- Coordinar proceso completo de registro
- Validar lógica de entrada/salida
- Integrar servicios de GPS y cámara
- Calcular estadísticas y métricas
- Mantener estado local reactivo

Métodos principales:

```
registerAttendance(tipo: 'entrada' | 'salida'): Promise<Attendance>
canCheckIn(): boolean
canCheckOut(): boolean
getTodayAttendances(): Observable<{entrada?: Attendance, salida?:
Attendance}>
getUserAttendanceHistory(): Promise<Attendance[]>
forceRefresh(): Promise<void>
```

Flujo de registro:

1. Validar precondiciones (entrada/salida)
2. Obtener ubicación GPS con fallbacks
3. Validar ubicación contra zonas permitidas
4. Capturar fotografía con fallbacks
5. Crear registro con timestamp
6. Guardar en Firestore
7. Actualizar estado local

GeolocationService

Propósito: Abstracción de servicios de geolocalización

Responsabilidades:

- Obtener ubicación GPS del dispositivo
- Validar ubicación contra zonas permitidas
- Cálculo de distancias geográficas
- Manejo de permisos de ubicación
- Fallbacks para diferentes plataformas

Métodos principales:

```
getCurrentPosition(): Observable<Position>
isWithinValidLocation(lat: number, lon: number): Promise<{
  isValid: boolean,
  distance: number,
  location: ValidLocation | null
}>
requestPermissions(): Promise<void>
calculateTravelTime(fromLat, fromLon, toLat, toLon): number
```

Validación de ubicación:

- Fórmula Haversine para cálculo de distancias
- Comparación contra múltiples ubicaciones válidas
- Tolerancia configurable por ubicación
- Logs detallados para debugging

CameraService

Propósito: Captura de imágenes multiplataforma

Responsabilidades:

- Captura de fotos en dispositivos móviles
- Fallback a web camera en navegadores
- Manejo de permisos de cámara
- Compresión y optimización de imágenes
- Gestión de errores específicos

Métodos principales:

```
takePicture(): Promise<string>
selectFromGallery(): Promise<string>
requestPermissions(): Promise<void>
isCameraAvailable(): Promise<boolean>
resizeImage(dataUrl: string, maxWidth, maxHeight): Promise<string>
```

6.2 Guards de Seguridad

AuthGuard

Propósito: Proteger rutas que requieren autenticación

Implementación:

```
canActivate(): Observable<boolean> {
  return this.authService.currentUser.pipe(
    map(user => {
      if (user) {
        return true;
      } else {
        this.router.navigate(['/login']);
        return false;
      }
    })
  );
}
```

NoAuthGuard

Propósito: Redirigir usuarios ya autenticados

Implementación:

```
canActivate(): boolean {
  const isAuthenticated = this.authService.isAuthenticated();
  if (isAuthenticated) {
    this.router.navigate(['/home']);
    return false;
  }
  return true;
}
```

6.3 Pipes Personalizados

DateFormatPipe

Propósito: Formateo consistente de fechas en español

```
transform(value: any): string {
  const date = new Date(value);
  return date.toLocaleDateString('es-ES', {
    day: '2-digit',
    month: '2-digit',
    year: 'numeric'
  });
}
```

TimeFormatPipe

Propósito: Formateo de horas con fallbacks

```
transform(value: any): string {
  if (typeof value === 'string' && /^\d{1,2}:\d{2}$/.test(value)) {
    return value;
  }
  const date = new Date(value);
  return date.toLocaleTimeString('es-ES', {
    hour: '2-digit',
    minute: '2-digit',
    hour12: false
  });
}
```

6.4 Componentes Compartidos

HeaderComponent

Propósito: Header reutilizable y configurable

Inputs configurables:

```
// Contenido
@Input() title: string = 'Sistema de Asistencia'
@Input() subtitle: string = ''
@Input() breadcrumb: string[] = []

// Elementos visibles
@Input() showBackButton: boolean = false
@Input() showNotifications: boolean = true
```

```
@Input() showSearch: boolean = false
@Input() showRefresh: boolean = true
@Input() showUserMenu: boolean = true

// Estados
@Input() isRefreshing: boolean = false
@Input() notificationCount: number = 0
@Input() progressValue: number = 0
```

Eventos emitidos:

```
@Output() menuClick = new EventEmitter<void>()
@Output() notificationsClick = new EventEmitter<void>()
@Output() searchClick = new EventEmitter<void>()
@Output() refreshClick = new EventEmitter<void>()
@Output() profileClick = new EventEmitter<void>()
@Output() settingsClick = new EventEmitter<void>()
```

---

7. Funcionalidades Implementadas

7.1 Sistema de Autenticación

Características

- **Registro completo** con información académica
- **Login seguro** con validación de credenciales
- **Persistencia de sesión** automática
- **Logout seguro** con confirmación
- **Validaciones robustas** en frontend y backend

Seguridad Implementada

- Contraseñas hasheadas por Firebase Auth
- Tokens JWT con expiración automática
- Validación de formato de email
- Protección contra ataques de fuerza bruta
- Sesiones seguras con renovación automática

7.2 Registro de Asistencias

Proceso Completo

1. **Validación de precondiciones:**
    - Usuario autenticado
    - Lógica de entrada/salida respetada
    - Permisos de dispositivo
2. **Captura de datos:**
    - Timestamp preciso del servidor
    - Coordenadas GPS con alta precisión
    - Fotografía del usuario
    - Metadata del dispositivo

3. **Validaciones automáticas:**
    - Ubicación dentro del radio permitido
    - Verificación de duplicados
    - Integridad de datos
4. **Almacenamiento seguro:**
    - Guardado en Firestore
    - Backup en Storage (fotos)
    - Logs de auditoría

Validación Geográfica

- **Radio configurable** por ubicación (default: 500m)
- **Múltiples ubicaciones** soportadas
- **Cálculo preciso** con fórmula Haversine
- **Tolerancia para errores** de GPS
- **Logging detallado** para debugging

7.3 Historial y Estadísticas

Funcionalidades de Consulta

- **Filtros por período:** semana, mes, todos
- **Búsqueda textual** en todos los campos
- **Agrupación por fecha** con orden cronológico
- **Estadísticas automáticas** por período
- **Exportación de datos** (compartir/copiar)

Métricas Calculadas

- Total de registros en período
- Cantidad de entradas vs salidas
- Porcentaje de ubicaciones válidas
- Horas trabajadas por día
- Promedios y tendencias

7.4 Interfaz de Usuario

Diseño Moderno

- **Material Design** con Ionic components
- **Glassmorphism** en elementos destacados
- **Animaciones fluidas** y transiciones
- **Feedback visual** para todas las acciones
- **Estados de carga** informativos

Responsive Design

- **Móvil first** con adaptación a desktop
- **Breakpoints específicos** para diferentes pantallas
- **Touch optimizado** para dispositivos táctiles

- **Accesibilidad** con contraste y tamaños adecuados

UX Optimizada

- **Flujos intuitivos** sin pasos innecesarios
- **Validación en tiempo real** con mensajes claros
- **Confirmaciones** para acciones importantes
- **Recuperación de errores** con opciones de reintento

7.5 Funcionalidades Técnicas

PWA (Progressive Web App)

- **Service Worker** para cache offline
- **Manifest** para instalación en dispositivos
- **Cache estratégico** de recursos críticos
- **Sincronización background** cuando sea posible

Manejo de Errores

- **Try-catch** comprehensivo en todas las operaciones
- **Fallbacks** para servicios críticos
- **Logging** detallado para debugging
- **Mensajes user-friendly** sin tecnicismos

Optimización de Performance

- **Lazy loading** de módulos y componentes
- **OnPush** change detection donde sea posible
- **Unsubscribe** automático para prevenir memory leaks
- **Compresión** de imágenes antes de upload

---

8. Configuración Técnica

8.1 Tecnologías y Versiones

Frontend

```
{
  "angular": "^17.0.0",
  "ionic": "^7.0.0",
  "typescript": "^5.0.0",
  "rxjs": "^7.8.0",
  "capacitor": "^5.0.0"
}
```

Backend (Firebase)

```
{
```

```
  "@angular/fire": "^17.0.0",
  "firebase": "^10.0.0"
}
```

Herramientas de Desarrollo

```
{
  "ionic-cli": "^7.0.0",
  "angular-cli": "^17.0.0",
  "capacitor-cli": "^5.0.0"
}
```

8.2 Configuración de Firebase

Configuración del Proyecto

```
export const firebaseConfig = {
  apiKey: "AIzaSyBF8Vf1wz9Qz_lfeKq1vAOxB7g76c6h-IQ",
  authDomain: "sistema-asistencias-418ad.firebaseapp.com",
  projectId: "sistema-asistencias-418ad",
  storageBucket: "sistema-asistencias-418ad.firebasestorage.app",
  messagingSenderId: "81431533973",
  appId: "1:81431533973:web:00ca4345cbf77752b5be9c"
};
```

Reglas de Seguridad de Firestore

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    // Usuarios solo pueden leer/escribir sus propios datos
    match /users/{userId} {
      allow read, write: if request.auth != null && request.auth.uid
== userId;
    }

    // Asistencias solo pueden ser creadas y leídas por el propio
usuario
    match /attendances/{attendanceId} {
      allow read, write: if request.auth != null &&
        request.auth.uid == resource.data.userId;
      allow create: if request.auth != null &&
        request.auth.uid == request.resource.data.userId;
    }

    // Ubicaciones válidas son de solo lectura para usuarios
autenticados
    match /locations/{locationId} {
      allow read: if request.auth != null;
    }
  }
}
```

Reglas de Seguridad de Storage

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
```

```
    match /attendance-photos/{userId}/{fileName} {
      allow read, write: if request.auth != null && request.auth.uid
== userId;
    }
  }
}
```

8.3 Configuración de Capacitor

capacitor.config.ts

```
import { CapacitorConfig } from '@capacitor/cli';

const config: CapacitorConfig = {
  appId: 'ar.edu.itscipolletti.asistencias',
  appName: 'Sistema de Asistencia ITS',
  webDir: 'dist',
  bundledWebRuntime: false,
  plugins: {
    Geolocation: {
      permissions: ['ACCESS_FINE_LOCATION', 'ACCESS_COARSE_LOCATION']
    },
    Camera: {
      permissions: ['CAMERA', 'WRITE_EXTERNAL_STORAGE']
    }
  }
};

export default config;
```

8.4 Estructura de Archivos del Proyecto

```
src/
├── app/
│   ├── directives/
│   │   └── auto-focus.directive.ts
│   ├── guards/
│   │   ├── auth.guard.ts
│   │   └── no-auth.guard.ts
│   ├── models/
│   │   ├── attendance.model.ts
│   │   ├── location.model.ts
│   │   └── user.model.ts
│   ├── pages/
│   │   ├── history/
│   │   │   ├── history.page.html
│   │   │   ├── history.page.scss
│   │   │   ├── history.page.ts
│   │   │   └── history-routing.module.ts
│   │   ├── home/
│   │   │   ├── home.page.html
│   │   │   ├── home.page.scss
│   │   │   ├── home.page.ts
│   │   │   └── home-routing.module.ts
│   │   ├── login/
│   │   │   ├── login.page.html
│   │   │   ├── login.page.scss
│   │   │   ├── login.page.ts
│   │   │   └── login-routing.module.ts
```

```
│   │   └── register/
│   │       ├── register.page.html
│   │       ├── register.page.scss
│   │       ├── register.page.ts
│   │       └── register-routing.module.ts
│   ├── pipes/
│   │   ├── date-format.pipe.ts
│   │   └── time-format.pipe.ts
│   ├── services/
│   │   ├── attendance.service.ts
│   │   ├── auth.service.ts
│   │   ├── camera.service.ts
│   │   ├── database.service.ts
│   │   ├── firebase-config.ts
│   │   └── geolocation.service.ts
│   ├── shared/
│   │   ├── header/
│   │   │   ├── header.component.html
│   │   │   ├── header.component.scss
│   │   │   └── header.component.ts
│   │   ├── loading/
│   │   │   ├── loading.component.html
│   │   │   ├── loading.component.scss
│   │   │   └── loading.component.ts
│   │   └── shared.module.ts
│   ├── utils/
│   │   ├── constants.ts
│   │   ├── helpers.ts
│   │   └── validators.ts
│   ├── app.component.html
│   ├── app.component.scss
│   ├── app.component.ts
│   ├── app.config.ts
│   └── app.routes.ts
├── assets/
├── environments/
└── main.ts
```

8.5 Scripts de NPM Disponibles

```
{
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test",
    "lint": "ng lint",
    "ionic:build": "ng build",
    "ionic:serve": "ng serve",
    "ionic:cap:run": "ionic capacitor run",
    "ionic:cap:sync": "ionic capacitor sync"
  }
}
```

8.6 Comandos de Desarrollo

Desarrollo Local

```
# Instalar dependencias
npm install

# Servidor de desarrollo
ionic serve

# Servidor con live reload
ionic serve --lab
```

Build y Deploy

```
# Build para producción
ionic build --prod

# Sincronizar con Capacitor
ionic cap sync

# Deploy a Firebase Hosting
firebase deploy --only hosting
```

Desarrollo Móvil

```
# Agregar plataforma iOS
ionic cap add ios

# Agregar plataforma Android
ionic cap add android

# Ejecutar en dispositivo
ionic cap run android --livereload
ionic cap run ios --livereload
```

---

9. Anexos

9.1 Archivos de Configuración Importantes

angular.json (Extracto relevante)

```
{
  "projects": {
    "app": {
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "assets": [
              "src/assets",
              "src/manifest.json"
            ],
            "styles": [
              "src/theme/variables.css",
```

```
              "src/global.scss"
            ]
          }
        }
      }
    }
  }
}
```

tsconfig.json (Configuración TypeScript)

```
{
  "compilerOptions": {
    "target": "ES2022",
    "lib": ["ES2022", "DOM"],
    "module": "ES2022",
    "moduleResolution": "bundler",
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "experimentalDecorators": true,
    "resolveJsonModule": true
  }
}
```

9.2 Variables de Entorno

environment.ts (Desarrollo)

```
export const environment = {
  production: false,
  firebase: {
    // Configuración de Firebase
  },
  api: {
    timeout: 30000,
    retryAttempts: 3
  },
  features: {
    offlineMode: true,
    debugMode: true
  }
};
```

environment.prod.ts (Producción)

```
export const environment = {
  production: true,
  firebase: {
    // Configuración de Firebase para producción
  },
  api: {
    timeout: 15000,
    retryAttempts: 2
  },
  features: {
    offlineMode: false,
    debugMode: false
  }
```

```
};
```

9.3 Dependencias del Proyecto

Dependencias Principales

```
{
  "dependencies": {
    "@angular/animations": "^17.0.0",
    "@angular/common": "^17.0.0",
    "@angular/core": "^17.0.0",
    "@angular/fire": "^17.0.0",
    "@angular/forms": "^17.0.0",
    "@angular/platform-browser": "^17.0.0",
    "@angular/router": "^17.0.0",
    "@capacitor/android": "^5.0.0",
    "@capacitor/app": "^5.0.0",
    "@capacitor/camera": "^5.0.0",
    "@capacitor/core": "^5.0.0",
    "@capacitor/geolocation": "^5.0.0",
    "@capacitor/haptics": "^5.0.0",
    "@capacitor/ios": "^5.0.0",
    "@capacitor/keyboard": "^5.0.0",
    "@capacitor/status-bar": "^5.0.0",
    "@ionic/angular": "^7.0.0",
    "firebase": "^10.0.0",
    "ionicons": "^7.0.0",
    "rxjs": "^7.8.0",
    "tslib": "^2.3.0",
    "zone.js": "^0.14.0"
  }
}
```

Dependencias de Desarrollo

```
{
  "devDependencies": {
    "@angular-devkit/build-angular": "^17.0.0",
    "@angular/cli": "^17.0.0",
    "@angular/compiler": "^17.0.0",
    "@angular/compiler-cli": "^17.0.0",
    "@capacitor/cli": "^5.0.0",
    "@ionic/cli": "^7.0.0",
    "@types/jasmine": "^5.0.0",
    "@types/node": "^20.0.0",
    "jasmine-core": "^5.0.0",
    "karma": "^6.4.0",
    "karma-chrome-headless": "^3.1.0",
    "karma-coverage": "^2.2.0",
    "karma-jasmine": "^5.0.0",
    "karma-jasmine-html-reporter": "^2.0.0",
    "typescript": "^5.2.0"
  }
}
```

9.4 Consideraciones de Seguridad

Datos Sensibles Protegidos

- Contraseñas hasheadas por Firebase Auth
- Tokens JWT con expiración automática
- Coordenadas GPS almacenadas de forma segura
- Fotos almacenadas con permisos específicos por usuario

Validaciones de Seguridad

- Autenticación requerida para todas las operaciones
- Validación de ubicación en servidor y cliente
- Sanitización de inputs del usuario
- Protección contra inyección de código

Mejores Prácticas Implementadas

- Principio de menor privilegio en reglas de Firestore
- Separación de datos por usuario
- Logging de operaciones críticas
- Manejo seguro de errores sin exposición de información

9.5 Roadmap de Mejoras Futuras

Funcionalidades Pendientes

1. **Página de Perfil de Usuario**
    - Edición de datos personales
    - Cambio de contraseña
    - Configuración de notificaciones
2. **Página de Configuración**
    - Preferencias de la aplicación
    - Configuración de notificaciones
    - Gestión de permisos
3. **Sistema de Notificaciones**
    - Notificaciones push para recordatorios
    - Alertas de registros pendientes
    - Notificaciones administrativas
4. **Dashboard Administrativo**
    - Vista de administradores
    - Reportes consolidados
    - Gestión de usuarios y ubicaciones

Mejoras Técnicas Sugeridas

1. **Modo Offline Completo**
    - Sincronización automática al reconectar
    - Queue de operaciones pendientes
    - Resolución de conflictos
2. **Optimizaciones de Performance**
    - Virtual scrolling en listas largas
    - Lazy loading de imágenes
    - Compresión adicional de datos

3. **Integraciones Externas**
    - API REST para sistemas externos
    - Exportación a Excel/PDF
    - Integración con sistemas de RR.HH.

9.7 Contacto y Soporte

Información de Desarrollo

- **Desarrolladores:** Beltran Enzo y Millahual Rayen
- **Email:** beltranenzocontacto@gmail.com rayencelestemillahual@gmail.com
- **Repositorio:** https://github.com/RayenCMillahual/AsistenciaGeolocalizada

Información del Cliente

- **Institución:** Instituto Técnico Superior Cipolletti

Conclusión

El Sistema de Asistencia ITS Cipolletti representa una solución moderna y completa para la gestión digital de asistencias de profesores. Con una arquitectura sólida basada en Angular e Ionic, integración robusta con Firebase, y un diseño centrado en el usuario, el sistema proporciona todas las funcionalidades necesarias para automatizar y mejorar el control de asistencias institucional.

La implementación incluye validación geográfica automática, captura fotográfica, historial detallado con estadísticas, y una interfaz moderna responsive que garantiza una excelente experiencia de usuario tanto en dispositivos móviles como en desktop.

El proyecto está listo para deployment en producción y cuenta con una base sólida para futuras mejoras y expansiones funcionales.
