

README



¿Qué es NEO-ID?

NEO-ID es una aplicación web diseñada para gestionar la identificación y registro de neonatos (bebés recién nacidos) y sus madres en un entorno hospitalario. El sistema utiliza códigos **DataMatrix** para garantizar la seguridad y trazabilidad de los pacientes, permitiendo la relación entre madres y neonatos mediante códigos únicos.

Objetivo del Proyecto de Grado: Implementar un sistema de identificación mediante DataMatrix que relacione neonatos con sus madres, facilitando la gestión y trazabilidad en el entorno hospitalario.



¿Qué hace la aplicación?

La aplicación permite:

Gestión de Madres (Funcionalidad Principal)

- Registrar información de madres (nombre, documento, servicio, habitación)
- Ver lista de todas las madres registradas
- Editar y actualizar datos de madres
- Asociar códigos DataMatrix únicos a cada madre
- Relacionar madres con sus neonatos

Gestión de Neonatos (Funcionalidad Principal)

- Registrar información completa de neonatos (datos de nacimiento, medidas, servicio)
- Vincular neonatos con sus madres mediante DataMatrix

- Asociar códigos DataMatrix únicos a cada neonato
- Registrar observaciones y estado del neonato
- Almacenar información médica relevante

Sistema de DataMatrix (Funcionalidad Principal)

- Generar códigos DataMatrix en lotes para impresión
- Asociar códigos pre-impresos con registros de madres y neonatos
- Escaneo de códigos DataMatrix para identificar rápidamente madres y neonatos
- Relación bidireccional entre madres y neonatos mediante códigos

Escaneo de Códigos QR/DataMatrix

- Escanear códigos con la cámara del dispositivo
- Identificar automáticamente a madres o neonatos al escanear
- Visualizar información completa del registro escaneado

Administración y Estadísticas

- Ver estadísticas generales (total de madres, neonatos activos/inactivos)
- Acceder a la base de datos completa
- Gestionar registros almacenados en Firebase

Reconocimiento Biométrico de Orejas (En Desarrollo)

- Nota: Esta funcionalidad está en fase de desarrollo
- El frontend está implementado, pero el backend Python para categorizar y tokenizar las imágenes aún no está disponible
- Captura de fotos de las orejas del neonato (interfaz lista)
- Sistema de reconocimiento pendiente de implementación

Sistema de Vinculación con DataMatrix

El sistema utiliza dos tipos de vinculación para relacionar madres y neonatos mediante códigos DataMatrix:

1. Vinculación Directa en Base de Datos

Cada neonato está vinculado directamente con su madre mediante el campo

`id_madre`:

```
// Estructura de datos
Neonato {
  id_neonato: string;
  id_madre: string;           // ← Vinculación directa con la madre
  nombre_neonato: string;
  // ... otros campos
}
```

Esta relación permite:

- Ver todos los neonatos de una madre específica
- Navegar desde un neonato a su madre
- Mantener la integridad referencial de los datos

2. Códigos DataMatrix del Sistema

Cuando se genera un código DataMatrix desde la aplicación, este contiene información de ambos (madre y neonato) en un formato JSON:

```
{
  "madreId": "M-20241013-001",
  "madreDocumento": "12345678",
  "neonatoId": "N-20241013-001",
  "neonatoNombre": "Juan Pérez",
  "timestamp": "2024-10-13T10:30:00.000Z"
}
```

Ventajas:

- Un solo código identifica tanto a la madre como al neonato
- Al escanear el código, se obtiene información de ambos
- Relación bidireccional garantizada
- Trazabilidad completa del vínculo

3. Códigos DataMatrix Pre-impresos de la Clínica

El sistema también soporta códigos DataMatrix pre-impresos que la clínica puede asignar:

```
// Madre con código pre-impreso
Madre {
  id_madre: string;
  qr_id: "NE037001" // ← Código DataMatrix de la clínica
  // ... otros campos
}

// Neonato con código pre-impreso
Neonato {
  id_neonato: string;
  qr_id: "NAB38001" // ← Código DataMatrix de la clínica
  // ... otros campos
}
```

Funcionamiento:

- Los códigos pre-impresos se pueden escanear y asociar a registros existentes

- El sistema busca en ambas colecciones (madres y neonatos) cuando se escanea un código
 - Permite flexibilidad para usar códigos externos o del sistema
-

Proceso de Escaneo y Búsqueda

Cuando se escanea un código DataMatrix, el sistema sigue este flujo:

1 Decodificación del código

- Si es un código del sistema → Extrae `madreId` y `neonatoId` (JSON).
- Si es un código pre-impreso → Obtiene el `qr_id` (texto simple).

2 Búsqueda en la base de datos

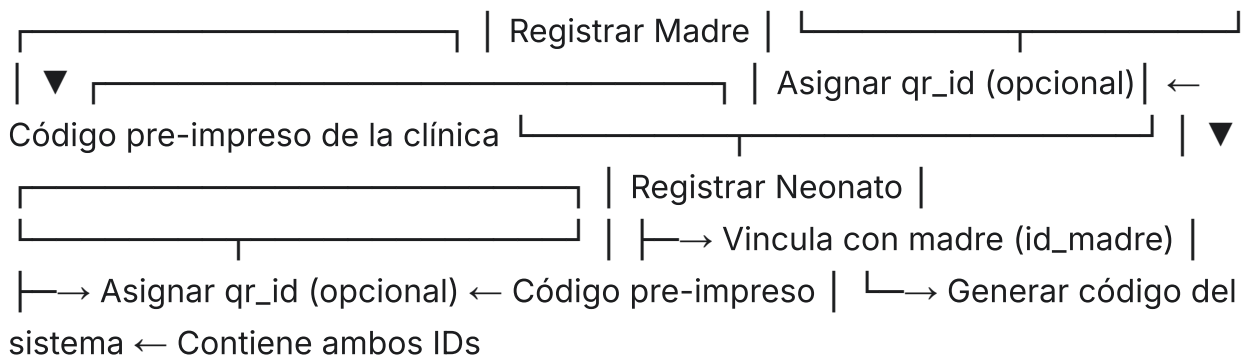
- Para códigos del sistema: busca directamente por IDs.
- Para códigos pre-impresos: busca en ambas colecciones por `qr_id`.

3 Resultado mostrado

- Información completa de la madre (si existe).
 - Información completa del neonato (si existe).
 - Relación entre ambos.
-

Flujo de Trabajo Completo (Registro y Generación)

Diagrama resumido del flujo:



Ejemplo práctico:

1. Se registra una madre con `id_madre: "M-20241013-001"`.
2. Se registra un neonato con `id_neonato: "N-20241013-001"` y `id_madre: "M-20241013-001"`.
3. El sistema genera un DataMatrix que contiene ambos IDs.
4. Al escanear ese código, se muestra información de la madre y del neonato.
5. La relación está garantizada tanto en la base de datos como en el código.

Cómo Funciona la Aplicación

Estructura General del Sistema

NEO-ID - Sistema Web (Vue 3 + TypeScript + Vite)



Firebase (Backend)

- Firestore (Base de Datos)
- Storage (Almacenamiento de Fotos)

Flujo de Funcionamiento Principal

Inicio de la Aplicación

- Usuario abre la aplicación → Dashboard
 - Dashboard muestra estadísticas:
 - Total de madres
 - Total de neonatos
 - Estados activos/inactivos
 - Accesos rápidos:
 - Gestión de Madres
 - Gestión de Neonatos
 - Escáner QR
 - Generador DataMatrix
-

Procesos Clave (Steppers)

Registro de Madre

1 Paso: Acceder a "Nueva Madre"

- El usuario abre el formulario "Nueva Madre".
- Campos: Nombre, Tipo Documento, Número Documento, Servicio, Habitación, Observaciones.
- Opcional: Escanear DataMatrix pre-impreso (qr_id de la clínica).

2 Paso: Firebase Service

- Genera ID único (formato M-YYYYMMDD-###).
- Valida datos.
- Guarda en Firestore (colección: madres).

3 Resultado en Firestore

Documento en `madres` con campos como:

- `id_madre`, `qr_id` (opcional), datos personales, estado, fecha.
-

Registro de Neonato

1 Paso 1: Acceder a "Nuevo Neonato"

- Formulario Paso 1:
 - Seleccionar Madre (o escanear DataMatrix, opcional).

2 Paso 2: Datos Básicos

- Formulario Paso 2:
 - Datos básicos: fecha/hora de nacimiento, sexo.

3 Paso 3: Medidas y Datos Médicos

- Formulario Paso 3:
 - Medidas (talla, peso, PC, PA, PT), servicio, observaciones.

4 Guardado en Firebase y Generación de DataMatrix

- Firebase Service genera ID único (N-YYYYMMDD-###).
 - Vincula con madre (`id_madre`) y guarda en `neonatos`.
 - Genera DataMatrix con info de madre y neonato; almacena URL en `qr_code`.
-

Escaneo de Códigos (Detalle del flujo)

1 Captura y Decodificación

- Componente QRScanner captura el código y lo decodifica.
- QRService analiza el contenido y determina el tipo.

2 Búsqueda en Firebase

- Si es código del sistema: busca por `madreId` y `neonatoId`.
- Si es código pre-impreso: busca por `qr_id` en `madres` y `neonatos`.

3 Presentación de Información

- Muestra la información disponible de madre y/o neonato.
 - Ofrece acciones: ver, editar, vincular.
-

Generación de Códigos DataMatrix

1 Selección de Prefijo y Modo

- El usuario selecciona prefijo (p. ej. NEO37, NAB38).
- Puede generar individualmente o en lote (10, 50, 100).

2 Generación y Control

- DataMatrix Utils genera códigos únicos evitando duplicados.
- Crea imágenes de los códigos.

3 Visualización y Exportación

- Grid de códigos para impresión.
 - Opciones: Imprimir, Exportar.
-

Estructura de Datos en Firebase

Al exportar los datos de Firebase se genera un objeto JSON con:

```
{
  "estadisticas": { ... },
  "madres": [ ... ],
  "neonatos": [ ... ],
  "relaciones": [ ... ],
  "fechaExportacion": "2025-11-11T22:27:48.791Z"
}
```

Colección: **madres** (ejemplo de documento)

```
{
  "id_madre": "M-20251015-003",
  "nombre_madre": "Silvia Acuña",
  "tipo_documento": "PPT",
  "numero_documento": "13568999",
  "servicio": "cirugia",
  "habitacion": "201",
  "observaciones": "Ninguna",
  "estado": true,
  "fecha": "2025-10-15T15:55:19.248Z",
  "qr_id": "1A2B"
}
```

Campos principales:

- id_madre, nombre_madre, tipo_documento, numero_documento, servicio, habitacion (opcional), observaciones, estado, fecha, qr_id (opcional).

Colección: **neonatos** (ejemplo de documento)

```

{
  "id_neonato": "D5RfI53JvzwSnYeuSTyL",
  "id_madre": "q7bqAWK0DM9T37qADUfp",
  "nombre_neonato": "Adrian Baeza",
  "fecha_nacimiento": "2025-10-08",
  "hora_nacimiento": "21:52",
  "sexo": "M",
  "talla": 432,
  "peso": 435,
  "pc": 234,
  "pa": 234234,
  "pt": 324,
  "permeabilidad_rectal": "Si",
  "servicio": "324",
  "se_encuentra_en": "234",
  "observaciones": "23423",
  "estado": true,
  "fecha": "2025-10-09T02:52:53.677Z",
  "qr_id": "1A2B",
  "qr_code": "https://...",
  "fotos_oreja_derecha": [
    "https://neoidapp.infinityfreeapp.com/uploads/neonatos/.../oreja_derecha_1760475315_4192440d.jpg"
  ],
  "fotos_oreja_izquierda": [
    "https://..."
  ]
}

```

Campos principales:

- id_neonato, id_madre, nombre_neonato, fecha_nacimiento, hora_nacimiento, sexo, talla, peso, pc, pa, pt, permeabilidad_rectal, servicio, se_encuentra_en, observaciones, estado, fecha, qr_id (opcional), qr_code (opcional), fotos_oreja_* (opcional).

Estadísticas (estructura)

```
{
  "estadisticas": {
    "madres": { "total": 4, "activas": 4, "inactivas": 0 },
    "neonatos": { "total": 4, "activos": 4, "inactivos": 0 },
    "llanto": { "total": 0 }
  }
}
```

Relaciones (exportación)

Al exportar se genera un array de relaciones que agrupa cada madre con sus neonatos:

```
{
  "relaciones": [
    {
      "madre": { ... },
      "neonatos": [ ... ]
    }
  ]
}
```

Nota: En Firebase, las relaciones se mantienen mediante el campo `id_madre` en cada documento de `neonatos`.

Relaciones entre Colecciones

- Colección `madres` (PK: `id_madre`).
- Colección `neonatos` (PK: `id_neonato`, FK: `id_madre`).
- Relación 1:N (una madre puede tener varios neonatos).
- La búsqueda de neonatos por madre se realiza filtrando por `id_madre`.



Arquitectura de la Aplicación

Frontend:

- Framework: Vue 3 con TypeScript
- Build Tool: Vite
- Estilos: Tailwind CSS
- Routing: Vue Router

Backend:

- Firebase (Firestore + Storage)

Backend de Reconocimiento (en desarrollo):

- API Python separada para procesar imágenes de orejas (pendiente)
-

Casos de Uso Principales (resumen paso a paso)

1

Registrar Madre y Neonato Nuevos

1. Usuario → "Nueva Madre".
2. Completa formulario.
3. Sistema genera ID: M-YYYYMMDD-###.
4. Guarda en Firebase.
5. Usuario → "Nuevo Neonato".
6. Selecciona madre.
7. Completa datos del neonato.
8. Sistema genera ID: N-YYYYMMDD-###.
9. Vincula con madre (`id_madre`).
10. Genera DataMatrix con ambos IDs.
11. Guarda en Firebase.

2

Escanear Código para Identificar

1. Usuario → "Escáner QR".
2. Activa cámara.
3. Escanea código DataMatrix.
4. Sistema decodifica JSON con `madreId` y `neonatoId`.
5. Busca en Firebase y muestra información completa.
6. Usuario puede editar o ver detalles.

3

Usar Código Pre-impreso de la Clínica

1. Clínica tiene códigos pre-impresos (p. ej. NEO37001).
2. Usuario registra madre y escanea código pre-impreso → asigna `qr_id`.
3. Usuario registra neonato y escanea su código.
4. Al escanear cualquiera de estos códigos, el sistema busca por `qr_id` en ambas colecciones y muestra el registro correspondiente.

Diagrama de Componentes Principales (resumen)

- Vistas: Dashboard, MadresList, NeonatosList, QRScannerView, etc.
 - Componentes compartidos: QRScanner, QRDisplay, MadreSelector, PageHeader, MobileNav.
 - Services: FirebaseService (CRUD), QRService (generación/decodificación), PhotoService (upload/download).
 - Backend: Firestore + Storage.
-

Estados y Transiciones (resumen)

- Inicial (Dashboard)
 - Registrar Madre → Madre Creada
 - Registrar Neonato → Neonato Creado → Genera DataMatrix
 - Escanear QR → Busca en DB → Muestra Resultado
 - Generar DataMatrix → Códigos Listos
-

Tecnologías Utilizadas

Dependencias principales:

- Vue 3, TypeScript, Vite, Firebase, Vue Router, Axios, html5-qrcode, qrcode

Herramientas de desarrollo:

- Tailwind CSS, PostCSS, vue-tsc
-

Estructura del Proyecto (resumen)

```
NeoID/
├─ src/
│   ├── components/
│   ├── views/
│   ├── services/
│   ├── router/
│   ├── types/
│   ├── config/
│   ├── App.vue
│   └─ main.ts
├─ public/
├─ dist/
├─ package.json
├─ vite.config.ts
└─ tailwind.config.js
```



¿Qué es Vite y cómo funciona?

Resumen:

- Herramienta de construcción moderna (desarrollo rápido con HMR).
- En desarrollo procesa módulos bajo demanda y usa ES Modules.
- En producción usa Rollup para optimizar, minificar y generar /dist.

Scripts en package.json típicos:

```
{
  "scripts": {
    "dev": "vite",
    "build": "vue-tsc -b && vite build",
    "preview": "vite preview"
  }
}
```



Cómo Empezar

Prerrequisitos

- Node.js (v16+)
- npm (incluido con Node.js)

Instalación

1. Clonar o descargar el proyecto.
2. Instalar dependencias:

```
npm install
```

3. Configurar Firebase en `src/config/firebase.ts` con tus credenciales.
4. Iniciar servidor de desarrollo:

```
npm run dev
```

La app se abre normalmente en `http://localhost:5173`

Comandos disponibles

```
# Desarrollo
npm run dev

# Producción
npm run build
npm run preview
```



Funcionalidades Principales (rutas)

- Dashboard: vista principal con estadísticas
 - Gestión de Madres: /madres
 - Gestión de Neonatos: /neonatos
 - Escáner QR/DataMatrix: /qr-scanner
 - Generador de DataMatrix: /herramientas/datamatrix
 - Reconocimiento de Orejas (en desarrollo): /ear-recognition
-



Configuración

Firestore (ejemplo)

Archivo `src/config/firebase.ts`:

```
import { initializeApp } from 'firebase/app';
import { getFirestore } from 'firebase/firestore';
import { getStorage } from 'firebase/storage';

const firebaseConfig = {
  // ... tus credenciales
};

export const app = initializeApp(firebaseConfig);
export const db = getFirestore(app);
export const storage = getStorage(app);
```

API de Reconocimiento de Orejas (En Desarrollo)

- Frontend implementado (captura/visualización).
- Backend Python para categorización y tokenización de imágenes pendiente.

- Configuración en `src/config/python-api.ts`.
-

Conceptos Importantes (rápido)

- Vue 3: componentes `.vue` con `template`, `script` `setup` y `style`.
 - TypeScript: tipos en variables y funciones.
 - Vue Router: configuración de rutas en `router/index.ts`.
 - Services: encapsulan lógica de negocio (p. ej. `firebase.service.ts`).
-

Solución de Problemas Comunes

- Error "Cannot find module":

```
rm -rf node_modules
npm install
```

- Error de Firebase:
 - Verifica credenciales en `src/config/firebase.ts`.
 - Revisa reglas de Firestore.
 - Puerto en uso:
 - Vite sugiere otro puerto automáticamente si 5173 está ocupado.
-

Notas para Desarrolladores

- La app es responsive y usa Tailwind CSS.
- Datos almacenados en Firebase (requiere conexión).
- Funcionalidad principal: DataMatrix para relacionar madres y neonatos.

- En desarrollo: reconocimiento biométrico de orejas (frontend listo; backend pendiente).
-

Objetivos del Proyecto de Grado

1. Implementar un sistema de identificación mediante DataMatrix que relacione neonatos con sus madres.
2. Gestionar el registro completo de madres y neonatos en un entorno hospitalario.
3. Facilitar la trazabilidad mediante códigos únicos asociados a cada registro.
4. Proporcionar una interfaz web moderna para la gestión y consulta de información.

Funcionalidades adicionales (en desarrollo):

- Reconocimiento biométrico de orejas (frontend implementado, backend pendiente).
-

Licencia

Proyecto privado destinado para uso hospitalario como proyecto de grado.