📋 Documentación Técnica - Sistema de Inspecciones Vehiculares PESV

📌 Información General

Nombre del Proyecto: Sistema de Inspecciones Vehiculares
Descripción: Aplicación web para gestión de inspecciones preoperacionales de vehículos según el Plan Estratégico de Seguridad Vial (PESV) de Colombia
Versión: 1.0.0
Última Actualización: Noviembre 2025
Repositorio: github.com/nelsonsanch/inspecciones-vehiculoso (https://github.com/nelsonsanch/inspecciones-vehiculoso)
URL Producción: inspeccionpesv.abacusai.app (https://inspeccionpesv.abacusai.app)

🏗️ Stack Tecnológico

Frontend

- **Framework:** Next.js 14.2.28 (App Router)
- **Lenguaje:** TypeScript 5.2.2
- **Styling:** Tailwind CSS 3.3.3
- **UI Components:**
- Radix UI (componentes accesibles)
- shadcn/ui (sistema de diseño)
- Lucide React (iconos)
- **State Management:**
- React Context API (autenticación)
- React Hooks (estado local)
- **Notificaciones:** Sonner (toast notifications)
- **Formularios:** React Hook Form + Zod (validación)

Backend & Servicios

- **Base de Datos:** Firebase Firestore (NoSQL)
- **Autenticación:** Firebase Authentication
- **Almacenamiento:** Firebase Storage
- **Hosting:** Netlify
- **Control de Versiones:** Git + GitHub

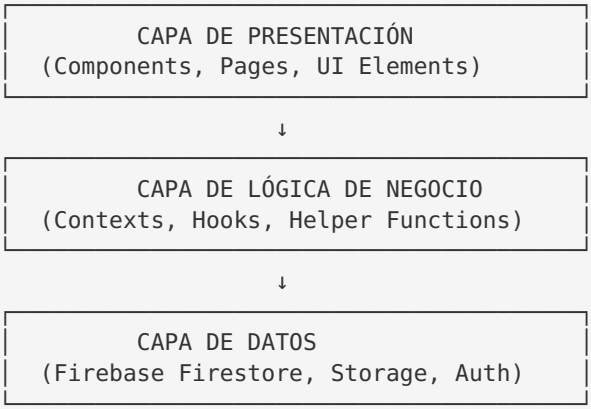
Herramientas de Desarrollo

- **Package Manager:** Yarn (v1.22+)
- **Linter:** ESLint
- **TypeScript Compiler:** tsc
- **Build Tool:** Next.js built-in

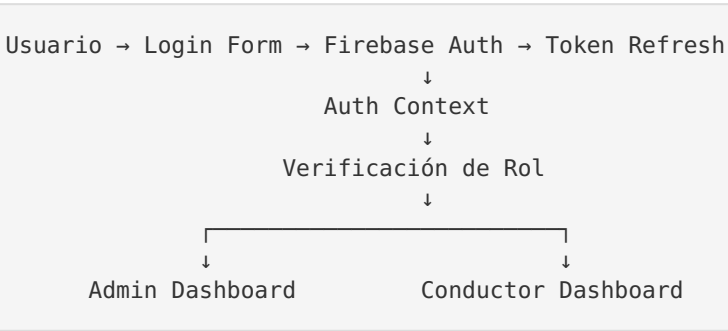
🏛️ Arquitectura de la Aplicación

Patrón Arquitectónico

La aplicación sigue una arquitectura de **3 capas**:



Flujo de Autenticación



Arquitectura de Componentes

```
app/
├── layout.tsx (Root Layout + Providers)
├── page.tsx (Landing/Auto-redirect)
├── login/
│   └── page.tsx (Login Page)
├── admin/
│   ├── layout.tsx (Admin Layout + Navigation)
│   ├── dashboard/
│   ├── vehiculos/
│   ├── conductores/
│   ├── inspecciones/
│   └── alertas/
└── conductor/
    ├── layout.tsx (Conductor Layout + Navigation)
    ├── dashboard/
    ├── inspecciones/
    └── nueva-inspeccion/
```

📁 Estructura de Directorios

```
nextjs_space/
├── app/                          # App Router (Next.js 13+)
│   ├── api/                      # API Routes
│   │   ├── upload-image/         # Subida de fotos de vehículos
│   │   ├── upload-signature/     # Subida de firmas digitales
│   │   └── generar-pdf/          # Generación de PDFs
│   ├── admin/                    # Panel de Administración
│   │   ├── dashboard/            # Dashboard principal admin
│   │   ├── vehiculos/            # Gestión de vehículos
│   │   │   ├── nuevo/            # Crear vehículo
│   │   │   └── [id]/             # Ver/Editar vehículo
│   │   ├── conductores/          # Gestión de conductores
│   │   │   ├── nuevo/            # Crear conductor
│   │   │   └── [id]/             # Ver/Editar conductor
│   │   ├── inspecciones/         # Gestión de inspecciones
│   │   └── alertas/              # Sistema de alertas
│   ├── conductor/                # Panel de Conductor
│   │   ├── dashboard/            # Dashboard conductor
│   │   ├── inspecciones/         # Ver inspecciones
│   │   │   └── [id]/             # Detalle de inspección
│   │   └── nueva-inspeccion/     # Crear inspección
│   │       └── [vehiculoId]/     # Formulario de inspección
│   ├── login/                    # Página de login
│   ├── layout.tsx                # Layout raíz
│   ├── page.tsx                  # Página principal
│   └── globals.css               # Estilos globales
│
├── components/                   # Componentes reutilizables
│   ├── auth/                     # Componentes de autenticación
│   │   ├── login-form.tsx        # Formulario de login
│   │   └── route-guard.tsx       # Protección de rutas por rol
│   ├── navigation/               # Componentes de navegación
│   │   ├── admin-nav.tsx         # Navegación admin
│   │   └── conductor-nav.tsx     # Navegación conductor
│   ├── dashboard/                # Componentes del dashboard
│   ├── inspeccion/               # Componentes de inspección
│   │   └── item-inspeccion.tsx   # Item individual de inspección
│   └── ui/                       # Componentes UI (shadcn/ui)
│       ├── button.tsx
│       ├── card.tsx
│       ├── dialog.tsx
│       ├── camera-capture.tsx    # Captura de cámara
│       ├── signature-pad.tsx     # Panel de firma digital
│       └── ... (40+ componentes)
│
├── contexts/                     # React Contexts
│   └── auth-context.tsx          # Contexto de autenticación
│
├── lib/                          # Utilidades y configuración
│   ├── firebase.ts               # Configuración Firebase
│   ├── firebase-admin.ts         # Firebase Admin SDK
│   ├── auth-types.ts             # Tipos de autenticación
│   ├── types.ts                  # Tipos de datos
│   ├── utils.ts                  # Funciones utilitarias
│   ├── alertas-helper.ts         # Lógica de alertas
│   ├── aws-config.ts             # Configuración AWS S3
│   ├── s3.ts                     # Utilidades S3
│   ├── db.ts                     # Helpers de base de datos
│   └── api-auth.ts               # Autenticación API
│
├── scripts/                      # Scripts de utilidad
│   └── seed.ts                   # Poblar base de datos
```

```
│   ├── fix-admin.ts              # Corregir usuario admin
│   ├── check-user-doc.ts         # Verificar documentos
│   └── ... (otros scripts)
│
├── public/                       # Archivos estáticos
│   ├── favicon.svg
│   ├── og-image.png
│   └── robots.txt
│
├── .env                          # Variables de entorno
├── .gitignore                    # Archivos ignorados por Git
├── package.json                  # Dependencias del proyecto
├── yarn.lock                     # Lock file de Yarn
├── tsconfig.json                 # Configuración TypeScript
├── tailwind.config.ts            # Configuración Tailwind
├── next.config.js                # Configuración Next.js
├── netlify.toml                  # Configuración Netlify
├── firestore.rules               # Reglas de seguridad Firestore
└── firestore.indexes.json        # Índices de Firestore
```

💾 Modelo de Datos (Firestore)

Colecciones Principales

1. `users` - Usuarios del Sistema

```typescript
interface User {
  uid: string;                    // UID de Firebase Auth
  email: string;
  nombre: string;
  role: 'administrador' | 'conductor';
  estado: 'activo' | 'inactivo';  // Control de acceso
  createdAt: string;              // ISO 8601
  updatedAt: string;
}
```

Documento de ejemplo:

```json
{
  "uid": "abc123xyz",
  "email": "admin@test.com",
  "nombre": "Administrador",
  "role": "administrador",
  "estado": "activo",
  "createdAt": "2025-11-27T10:00:00Z",
  "updatedAt": "2025-11-27T10:00:00Z"
}
```

2. `conductores` - Conductores

```
interface Conductor {
  id?: string;                    // ID del documento
  nombre: string;
  email: string;
  licencia: string;               // Número de licencia
  telefono?: string;
  estado: 'activo' | 'inactivo';
  userId: string;                 // Referencia a users/{userId}
  createdAt: string;
  updatedAt: string;
}
```

3. `vehiculos` - Vehículos

```
interface Vehiculo {
  id?: string;
  placa: string;                  // Ej: "ABC123"
  marca: string;
  modelo: string;
  año: number;
  tipoVehiculo: 'automovil' | 'camioneta' | 'camion' | 'bus' | 'motocicleta' | 'fur-
gon' | 'otro';
  color: string;
  kilometrajeInicial: number;
  kilometrajeActual: number;      // Se actualiza con cada inspección
  estado: 'activo' | 'inactivo' | 'mantenimiento';
  soatVencimiento?: string;       // Fecha ISO 8601
  tecnomecanicaVencimiento?: string;
  fotos?: {
    delantera?: string;           // Storage path
    lateralIzquierda?: string;
    lateralDerecha?: string;
    trasera?: string;
  };
  createdAt: string;
  updatedAt: string;
}
```

4. `vehiculos/{vehiculoId}/eventos` - Historial de Vehículo (Subcolección)

```
interface EventoVehiculo {
  id?: string;
  tipo: 'mantenimiento' | 'reparacion' | 'actualizacion_documento' | 'cambio_estado';
  descripcion: string;
  fecha: string;                  // ISO 8601
  realizadoPor: string;           // UID del usuario
  detalles?: {
    documentoActualizado?: 'soat' | 'tecnomecanica';
    nuevaFechaVencimiento?: string;
    estadoAnterior?: string;
    estadoNuevo?: string;
  };
  createdAt: string;
}
```

5. `inspecciones` - Inspecciones Realizadas

```typescript
interface Inspeccion {
  id?: string;
  vehiculoId: string;             // Referencia a vehiculos/{id}
  conductorId: string;            // Referencia a users/{uid}
  conductorNombre: string;
  vehiculoPlaca: string;
  fecha: string;                  // ISO 8601
  estado: 'aprobada' | 'rechazada' | 'pendiente';

  // Información del conductor
  estadoSalud: 'bueno' | 'regular' | 'malo';
  observacionesSalud?: string;

  // Información del viaje
  destino: string;
  kilometrajeActual: number;

  // Items de inspección (agrupados por categorías)
  itemsLuces: InspeccionItem[];
  itemsMotor: InspeccionItem[];
  itemsFrenos: InspeccionItem[];
  itemsCarroceria: InspeccionItem[];
  itemsSeguridad: InspeccionItem[];
  itemsDocumentacion: InspeccionItem[];

  // Fotos del vehículo
  fotosVehiculo?: {
    delantera?: string;
    lateralIzquierda?: string;
    lateralDerecha?: string;
    trasera?: string;
  };

  // Firma del conductor
  firmaDigital?: string;          // Storage path

  observaciones?: string;
  createdAt: string;
  updatedAt: string;
}

interface InspeccionItem {
  nombre: string;
  estado: 'bueno' | 'malo' | 'no_aplica';
  critico?: boolean;              // Si requiere acción inmediata
}
```

Ejemplo de items de inspección:

```json
{
  "itemsLuces": [
    { "nombre": "Luces delanteras", "estado": "bueno", "critico": true },
    { "nombre": "Luces traseras", "estado": "bueno", "critico": true },
    { "nombre": "Direccionales", "estado": "malo", "critico": true }
  ]
}
```

6. `alertas` - Sistema de Alertas

```
interface Alerta {
  id?: string;
  tipo: 'documento_vencido' | 'documento_proximo_vencer' | 'falla_critica' | 'mantenimiento_requerido';
  prioridad: 'alta' | 'media' | 'baja';
  vehiculoId: string;
  vehiculoPlaca: string;
  mensaje: string;

  // Campos específicos según tipo
  documentoTipo?: 'soat' | 'tecnomecanica';
  fechaVencimiento?: string;
  diasParaVencer?: number;

  inspeccionId?: string;          // Si la alerta viene de una inspección
  itemFallido?: string;           // Nombre del item con falla

  estado: 'pendiente' | 'resuelta' | 'en_proceso';
  resueltaPor?: string;           // UID del usuario que la resolvió
  fechaResolucion?: string;
  notasResolucion?: string;

  createdAt: string;
  updatedAt: string;
}
```

🔐 Sistema de Autenticación y Autorización

Roles de Usuario

Administrador

- **Permisos:**
- ✅ Ver, crear, editar y eliminar vehículos
- ✅ Ver, crear, editar y gestionar conductores
- ✅ Ver todas las inspecciones
- ✅ Gestionar alertas y mantenimiento
- ✅ Ver estadísticas globales
- ✅ Acceder al sistema de alertas

Conductor

- **Permisos:**
- ✅ Ver sus propias inspecciones
- ✅ Crear nuevas inspecciones
- ✅ Ver información de vehículos asignados
- ❌ No puede editar vehículos
- ❌ No puede ver inspecciones de otros conductores
- ❌ No tiene acceso al panel de alertas

Flujo de Autenticación

1. **Login:**
   ```typescript
   // components/auth/login-form.tsx
   const userCredential = await signInWithEmailAndPassword(auth, email, password);
   await userCredential.user.getIdToken(true); // Forzar actualización de token
   await new Promise(resolve => setTimeout(resolve, 500)); // Propagación de token

const userDoc = await getDoc(doc(db, ‘users’, userCredential.user.uid));
const userData = userDoc.data();

// Verificar estado
if (userData.estado === ‘inactivo’) {
await signOut(auth);
throw new Error(‘Usuario inactivo’);
}

// Redirigir según rol
if (userData.role === ‘administrador’) {
router.push(‘/admin/dashboard’);
} else {
router.push(‘/conductor/dashboard’);
}
```

1. **Persistencia de Sesión:**
   ```typescript
   // contexts/auth-context.tsx
   useEffect(() => {
   const unsubscribe = onAuthStateChanged(auth, async (firebaseUser) => {
   if (firebaseUser) {
   await firebaseUser.getIdToken(true);
   const userDoc = await getDoc(doc(db, ‘users’, firebaseUser.uid));

```
if (userDoc.exists()) {
  const userData = userDoc.data();

  // Auto-logout si está inactivo
  if (userData.estado === 'inactivo') {
    await signOut(auth);
    return;
  }

  setUser(userData);
}
```

   }
   });
   return () => unsubscribe();
   }, []);
   ```

2. **Protección de Rutas:**
   ```typescript
   // components/auth/route-guard.tsx
   export function RouteGuard({ children, allowedRoles }) {
   const { user, loading } = useAuth();
   const router = useRouter();

   useEffect(() => {
   if (!loading && !user) {
   router.push('/login');
   } else if (!loading && user && !allowedRoles.includes(user.role)) {
   router.push('/login');
   }
   }, [user, loading]);

   if (loading || !user) return ;
   return <>{children};
   }
   ```

Reglas de Seguridad de Firestore

IMPORTANTE: Las reglas de Firestore **NO** verifican roles usando `get()` debido a problemas de rendimiento y timing. La verificación de roles se hace en el código de la aplicación.

```
// firestore.rules
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    function isAuthenticated() {
      return request.auth != null;
    }

    // Usuarios: Leer propios datos, escribir solo admin
    match /users/{userId} {
      allow read: if isAuthenticated();
      allow write: if isAuthenticated();
    }

    // Vehículos: Leer todos autenticados, escribir todos autenticados
    match /vehiculos/{vehiculoId} {
      allow read: if isAuthenticated();
      allow write: if isAuthenticated();

      // Eventos de vehículo
      match /eventos/{eventoId} {
        allow read: if isAuthenticated();
        allow write: if isAuthenticated();
      }
    }

    // Inspecciones: Verificación de ownership en el código
    match /inspecciones/{inspeccionId} {
      allow read: if isAuthenticated();
      allow create: if isAuthenticated();
      allow update, delete: if isAuthenticated() &&
                               request.auth.uid == resource.data.conductorId;
    }

    // Alertas: Solo lectura para autenticados
    match /alertas/{alertaId} {
      allow read: if isAuthenticated();
      allow write: if isAuthenticated();
    }

    // Conductores
    match /conductores/{conductorId} {
      allow read: if isAuthenticated();
      allow write: if isAuthenticated();
    }
  }
}
```

⚠️ **NOTA CRÍTICA:** La autorización por rol (admin vs conductor) se implementa en el código del cliente mediante:

1. `RouteGuard` para proteger páginas completas
2. Verificaciones en componentes individuales usando `useAuth().user.role`
3. Validaciones en acciones críticas antes de modificar datos

Ver `SEGURIDAD.md` para más detalles sobre la estrategia de seguridad.

🎯 Funcionalidades Implementadas

1. Gestión de Vehículos

Crear Vehículo

- Formulario completo con todos los campos requeridos
- Captura de 4 fotos del vehículo:
- Vista delantera
- Vista lateral izquierda
- Vista lateral derecha
- Vista trasera
- Subida a Firebase Storage
- Validación de placa (formato colombiano)
- Registro de kilometraje inicial
- Fechas de vencimiento de documentos (SOAT, Tecnomecánica)

Ubicación: `/app/admin/vehiculos/nuevo/page.tsx`

Editar Vehículo

- Actualización de información general
- Cambio de estado (activo/inactivo/mantenimiento)
- Actualización de fechas de vencimiento de documentos
- Registro automático de cambios en el historial

Ubicación: `/app/admin/vehiculos/[id]/editar/page.tsx`

Ver Detalle de Vehículo

- Información general completa
- Fotos del vehículo
- Estadísticas de inspecciones
- Historial de eventos (mantenimiento, actualizaciones, cambios)
- Alertas activas relacionadas

Ubicación: `/app/admin/vehiculos/[id]/page.tsx`

Eliminar Vehículo

- Confirmación de eliminación
- Eliminación de fotos asociadas en Storage
- Eliminación de subcolección de eventos

Ubicación: `/app/admin/vehiculos/page.tsx`

2. Gestión de Conductores

Crear Conductor

- Creación simultánea en Firebase Auth y Firestore
- Generación de contraseña temporal
- Sincronización de UID entre Auth y Firestore
- Estado inicial: ‘activo’

Flujo de creación:

```
// 1. Crear en Firebase Auth
const userCredential = await createUserWithEmailAndPassword(auth, email, tempPass-
word);
const uid = userCredential.user.uid;

// 2. Crear documento en 'users' con el UID
await setDoc(doc(db, 'users', uid), {
  uid,
  email,
  nombre,
  role: 'conductor',
  estado: 'activo'
});

// 3. Crear documento en 'conductores'
await addDoc(collection(db, 'conductores'), {
  nombre,
  email,
  licencia,
  telefono,
  userId: uid,
  estado: 'activo'
});
```

Ubicación: `/app/admin/conductores/nuevo/page.tsx`

Editar Conductor

- Actualización de información personal
- Cambio de número de licencia
- Actualización de teléfono
- Sincronización automática entre `users` y `conductores`

Ubicación: `/app/admin/conductores/[id]/editar/page.tsx`

Activar/Desactivar Conductor

- Cambio de estado entre ‘activo’ e ‘inactivo’
- Bloqueo automático de login para conductores inactivos
- Preservación de datos históricos
- Sincronización en ambas colecciones (`users` y `conductores`)

Ventajas sobre eliminación:
- ✅ Preservación de historial de inspecciones
- ✅ Auditoría completa
- ✅ Reversibilidad inmediata
- ✅ Reutilización de email

Ver `CONDUCTORES-ESTADO.md` para más detalles.

Eliminar Conductor

- **Proceso de 2 pasos:**
  1. Eliminar documento de Firestore (automático desde la app)
  2. Eliminar usuario de Firebase Auth (manual desde Firebase Console)

IMPORTANTE: La aplicación proporciona instrucciones paso a paso y abre automáticamente Firebase Console para facilitar la eliminación en Auth.

Ver `ELIMINACION-CONDUCTORES.md` para guía completa.

Ubicación: `/app/admin/conductores/page.tsx`

3. Sistema de Inspecciones

Crear Inspección (Conductor)

Flujo completo:

1. **Selección de Vehículo**
   - Vista de todos los vehículos activos
   - Información básica de cada vehículo

2. **Formulario de Inspección**
   - **Información de Salud del Conductor:**
     - Estado de salud (bueno/regular/malo)
     - Observaciones de salud

- **Información del Viaje:**
  - Destino
  - Kilometraje actual del vehículo
- **Items de Inspección (agrupados por categorías):**
  - 🔦 Luces (delanteras, traseras, direccionales, etc.)
  - ⚙️ Motor (nivel de aceite, refrigerante, baterías, etc.)
  - 🛑 Frenos (estado, líquido, freno de mano)
  - 🚗 Carrocería (espejos, limpiaparabrisas, neumáticos)
  - 🦺 Seguridad (cinturones, botiquín, extintor, kit carreteras)
  - 📄 Documentación (SOAT, tecnomecánica, licencia)
- **Cada item tiene 3 opciones:**
  - ✅ Bueno
  - ❌ Malo
  - 🚫 No Aplica
- **Items Críticos:** Marcados en rojo si son esenciales

1. **Captura de Fotos del Vehículo (4 ángulos)**
   - Vista delantera
   - Vista lateral izquierda
   - Vista lateral derecha
   - Vista trasera

2. **Firma Digital del Conductor**
   - Canvas interactivo para firma
   - Botón para limpiar y rehacer

3. **Observaciones Generales**
   - Campo de texto libre para notas adicionales

4. **Generación Automática de Alertas**
   - Al enviar la inspección, el sistema analiza:
     - Items marcados como “Malo” y críticos → Alerta de falla crítica
     - Documentos próximos a vencer (15 días) → Alerta de documento
     - Documentos vencidos → Alerta de alta prioridad

Ubicación: `/app/conductor/nueva-inspeccion/[vehiculoId]/page.tsx`

Lógica de alertas: `/lib/alertas-helper.ts`

Ver Inspecciones (Conductor)

- Lista de inspecciones propias
- Filtros por fecha y estado
- Vista de detalle de cada inspección

Ubicación: `/app/conductor/inspecciones/page.tsx`

Gestión de Inspecciones (Admin)

- Ver todas las inspecciones del sistema
- Filtros avanzados:
- Por fecha
- Por vehículo
- Por conductor
- Por estado (aprobada/rechazada/pendiente)
- Estadísticas generales
- Eliminar inspecciones

Ubicación: `/app/admin/inspecciones/page.tsx`

4. Sistema de Alertas y Mantenimiento

Generación Automática de Alertas

El sistema genera alertas automáticamente en 3 situaciones:

1. **Documentos Próximos a Vencer (15 días)**

```typescript
  if (diasParaVencer <= 15 && diasParaVencer > 0) {
    await addDoc(collection(db, 'alertas'), {
      tipo: 'documento_proximo_vencer',
      prioridad: 'media',
      vehiculoId,
      vehiculoPlaca,
      mensaje: `${docTipo} del vehículo ${placa} vence en ${dias} días`,
      documentoTipo,
      fechaVencimiento,
      diasParaVencer,
      estado: 'pendiente',
      createdAt: new Date().toISOString()
    });
  }
```

2. **Documentos Vencidos**
   - Prioridad: ALTA
   - Requiere acción inmediata

3. **Fallas Críticas en Inspección**
   - Detecta items marcados como “Malo” y que son críticos
   - Ejemplos: luces, frenos, neumáticos en mal estado
   - Genera alerta de mantenimiento requerido

Ubicación de lógica: `/lib/alertas-helper.ts`

Panel de Alertas (Admin)

Características:
- Lista de todas las alertas activas
- Filtros por:
- Prioridad (alta/media/baja)
- Estado (pendiente/en proceso/resuelta)
- Tipo de alerta
- Visualización con códigos de color:
- 🔴 Alta prioridad (rojo)
- 🟡 Media prioridad (amarillo)
- 🟢 Baja prioridad (verde)

Acciones disponibles:
- Marcar como “En proceso”
- Resolver alerta (con notas de resolución)
- Ver contexto completo (vehículo, inspección relacionada)
- Acceso directo al vehículo o inspección

Ubicación: `/app/admin/alertas/page.tsx`

5. Dashboards

Dashboard Administrador

Métricas principales:
- Total de vehículos por estado
- Total de conductores activos
- Inspecciones del mes (aprobadas/rechazadas)
- Alertas pendientes por prioridad

Gráficos y visualizaciones:
- Evolución de inspecciones (últimos 30 días)
- Distribución de vehículos por tipo
- Top 5 vehículos con más inspecciones
- Alertas críticas del día

Alertas destacadas:
- Documentos vencidos hoy
- Inspecciones rechazadas recientes
- Vehículos en mantenimiento

Ubicación: `/app/admin/dashboard/page.tsx`

Dashboard Conductor

Información visible:
- Resumen de inspecciones realizadas
- Última inspección
- Próximas inspecciones pendientes
- Acceso rápido a crear nueva inspección

Ubicación: `/app/conductor/dashboard/page.tsx`

6. Subida de Archivos (Firebase Storage)

Fotos de Vehículos

- Captura directa desde cámara o selección de archivo
- Compresión automática (opcional)
- Almacenamiento en Firebase Storage: `uploads/{timestamp}-{filename}`
- Path almacenado en Firestore (no URL completa)

Componente: `/components/ui/camera-capture.tsx`

Firmas Digitales

- Canvas HTML5 para firma manuscrita
- Conversión a imagen PNG
- Subida a Firebase Storage
- Asociación con inspección

Componente: `/components/ui/signature-pad.tsx`

API Endpoints:
- `/api/upload-image` - Subida de fotos
- `/api/upload-signature` - Subida de firmas

Utilidades: `/lib/s3.ts` (renombrado, ahora usa Firebase Storage)

⚙️ Configuración del Proyecto

Variables de Entorno

Archivo: `.env`

```
# Firebase Configuration
NEXT_PUBLIC_FIREBASE_API_KEY=your_api_key
NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=your_project.firebaseapp.com
NEXT_PUBLIC_FIREBASE_PROJECT_ID=your_project_id
NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET=your_project.appspot.com
NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID=123456789
NEXT_PUBLIC_FIREBASE_APP_ID=1:123456789:web:abcdef
NEXT_PUBLIC_FIREBASE_MEASUREMENT_ID=G-XXXXXXXXXX

# Netlify Configuration (opcional)
NEXTAUTH_URL=https://inspeccionpesv.abacusai.app
```

⚠️ **IMPORTANTE:**
- Las variables que comienzan con `NEXT_PUBLIC_` son accesibles en el cliente

- Nunca commitear el archivo `.env` al repositorio (está en `.gitignore`)
- En Netlify, configurar estas variables en: Site settings → Environment variables

Configuración de Firebase

1. Crear Proyecto en Firebase Console

1. Ir a Firebase Console (https://console.firebase.google.com/)
2. Crear nuevo proyecto
3. Habilitar Google Analytics (opcional)

2. Habilitar Servicios

Authentication:
- Ir a Authentication → Sign-in method
- Habilitar “Email/Password”

Firestore Database:
- Ir a Firestore Database
- Crear base de datos en modo “Producción”
- Seleccionar ubicación (ej: us-east1)

Storage:
- Ir a Storage
- Iniciar en modo de prueba (luego configurar reglas)

3. Configurar Reglas de Seguridad

Firestore Rules:

```
// Ver archivo firestore.rules completo arriba
```

Publicar reglas:
1. Firestore Database → Reglas
2. Pegar reglas
3. Publicar

Storage Rules:

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /uploads/{allPaths=**} {
      allow read: if request.auth != null;
      allow write: if request.auth != null
                   && request.resource.size < 5 * 1024 * 1024  // 5MB max
                   && request.resource.contentType.matches('image/.*');
    }
  }
}
```

4. Obtener Credenciales

1. Project Settings → General
2. Scroll hasta “Your apps”
3. Seleccionar “Web app” (icon)

4. Copiar el objeto `firebaseConfig`
5. Agregar valores al archivo `.env`

Instalación Local

```
# 1. Clonar repositorio
git clone https://github.com/nelsonsanch/inspecciones-vehiculoso.git
cd inspecciones-vehiculoso/nextjs_space

# 2. Instalar dependencias
yarn install

# 3. Configurar variables de entorno
cp .env.example .env
# Editar .env con tus credenciales de Firebase

# 4. Inicializar base de datos con datos de prueba
yarn tsx scripts/seed.ts

# 5. Ejecutar en desarrollo
yarn dev

# La app estará disponible en http://localhost:3000
```

Build de Producción

```
# Build
yarn build

# Iniciar servidor de producción
yarn start
```

Scripts Disponibles

```
{
  "scripts": {
    "dev": "next dev",              // Desarrollo
    "build": "next build",          // Build producción
    "start": "next start",          // Servidor producción
    "lint": "next lint"             // Linting
  }
}
```

Scripts de utilidad:

- `yarn tsx scripts/seed.ts` - Poblar BD con datos de prueba
- `yarn tsx scripts/fix-admin.ts` - Verificar usuario admin
- `yarn tsx scripts/check-user-doc.ts` - Verificar documento de usuario

🚀 Deployment (Netlify)

Configuración en Netlify

1. Conectar Repositorio

1. Ir a Netlify (https://www.netlify.com/)
2. New site from Git
3. Conectar con GitHub
4. Seleccionar repositorio: `nelsonsanch/inspecciones-vehiculoso`

2. Configurar Build Settings

Base directory: `nextjs_space`

Build command: `yarn build`

Publish directory: `nextjs_space/.next`

Archivo `netlify.toml`:

```
[build]
  base = "nextjs_space"
  command = "yarn build"
  publish = "nextjs_space/.next"

[build.environment]
  NODE_VERSION = "18"

[[redirects]]
  from = "/*"
  to = "/index.html"
  status = 200
```

3. Configurar Variables de Entorno

1. Site settings → Environment variables
2. Agregar todas las variables de `.env`:
   - `NEXT_PUBLIC_FIREBASE_API_KEY`
   - `NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN`
   - `NEXT_PUBLIC_FIREBASE_PROJECT_ID`
   - `NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET`
   - `NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID`
   - `NEXT_PUBLIC_FIREBASE_APP_ID`
   - `NEXT_PUBLIC_FIREBASE_MEASUREMENT_ID`
   - `NEXTAUTH_URL`

4. Deploy

- Cada push a `main` despliega automáticamente
- Preview deploys para pull requests
- Rollback disponible desde el dashboard

Dominio Personalizado

Actual: `inspeccionpesv.abacusai.app`

Para cambiar:
1. Site settings → Domain management
2. Add custom domain
3. Configurar DNS según instrucciones

🐛 Problemas Conocidos y Soluciones

1. Error: “Missing or insufficient permissions”

Causa: Reglas de Firestore muy restrictivas o token no propagado.

Solución:
1. Verificar reglas en Firebase Console
2. Asegurar que el código incluye:

```
typescript
   await user.getIdToken(true);
   await new Promise(resolve => setTimeout(resolve, 500));
```

Archivo afectado: `components/auth/login-form.tsx` , `contexts/auth-context.tsx`

2. Toast de “Subiendo foto...” no desaparece

Causa: No se cierra el toast después de la subida.

Solución:

```
const toastId = toast.loading('Subiendo foto...');
// ... subir foto
toast.dismiss(toastId);
```

Archivo afectado: `app/admin/vehiculos/nuevo/page.tsx`

3. Error: “email-already-in-use” al crear conductor

Causa: El email ya existe en Firebase Auth.

Solución:
1. La app muestra toast con instrucciones
2. Opción 1: Usar un email diferente
3. Opción 2: Eliminar el usuario existente en Firebase Console

Archivo: Ver `ELIMINACION-CONDUCTORES.md`

4. Netlify build falla: “yarn.lock not found”

Causa: `yarn.lock` era un symlink.

Solución: Reemplazado con archivo real en el proyecto. Ya está resuelto.

5. Conductor no puede iniciar sesión después de creación

Causa: Desincronización entre Firebase Auth y Firestore.

Solución:

```
# Ejecutar script de corrección
yarn tsx scripts/fix-existing-conductor.ts
```

Prevención: El código actual crea usuarios correctamente con UID sincronizado.

6. Errores de React Hydration

Causa: Diferencias entre renderizado servidor/cliente.

Solución:
- Evitar `new Date()` directo en componentes
- Usar `suppressHydrationWarning` en `<html>`
- Mover lógica dependiente del cliente a `useEffect`

Archivo afectado: `app/layout.tsx`

📚 Guías de Desarrollo

Agregar una Nueva Página

```tsx
// 1. Crear archivo en app/
// app/admin/nueva-pagina/page.tsx

'use client';

import { RouteGuard } from '@/components/auth/route-guard';
import { useAuth } from '@/contexts/auth-context';

export default function NuevaPaginaPage() {
  const { user } = useAuth();

  return (
    <RouteGuard allowedRoles={['administrador']}>
      <div>
        <h1>Nueva Página</h1>
        {/* Contenido */}
      </div>
    </RouteGuard>
  );
}
```

Agregar una Nueva Colección de Firestore

```
// 1. Definir tipo en lib/types.ts
interface NuevaColeccion {
  id?: string;
  campo1: string;
  campo2: number;
  createdAt: string;
}

// 2. Actualizar reglas en firestore.rules
match /nueva_coleccion/{docId} {
  allow read: if isAuthenticated();
  allow write: if isAdmin();
}

// 3. Usar en componentes
import { collection, addDoc, getDocs } from 'firebase/firestore';
import { db } from '@/lib/firebase';

const nuevosRef = collection(db, 'nueva_coleccion');
const nuevosSnap = await getDocs(nuevosRef);
const nuevos = nuevosSnap.docs.map(doc => ({
  id: doc.id,
  ...doc.data()
}));
```

Agregar un Nuevo Componente UI

```
// components/ui/nuevo-componente.tsx

import * as React from 'react';
import { cn } from '@/lib/utils';

interface NuevoComponenteProps
  extends React.HTMLAttributes<HTMLDivElement> {
  variant?: 'default' | 'secondary';
}

const NuevoComponente = React.forwardRef<
  HTMLDivElement,
  NuevoComponenteProps
>(({ className, variant = 'default', ...props }, ref) => {
  return (
    <div
      ref={ref}
      className={cn(
        'base-styles',
        variant === 'secondary' && 'secondary-styles',
        className
      )}
      {...props}
    />
  );
});
NuevoComponente.displayName = 'NuevoComponente';

export { NuevoComponente };
```

Agregar una Nueva Alerta Automática

```
// lib/alertas-helper.ts

export async function generarNuevaTipoAlerta(
  vehiculoId: string,
  vehiculoPlaca: string,
  // otros parámetros
) {
  try {
    await addDoc(collection(db, 'alertas'), {
      tipo: 'nuevo_tipo_alerta',
      prioridad: 'media',  // alta, media, baja
      vehiculoId,
      vehiculoPlaca,
      mensaje: 'Mensaje descriptivo de la alerta',
      estado: 'pendiente',
      createdAt: new Date().toISOString(),
      updatedAt: new Date().toISOString()
    });
  } catch (error) {
    console.error('Error generando alerta:', error);
  }
}
```

Luego llamar esta función desde donde sea necesario (ej: después de guardar una inspección).

Testing

Actualmente no hay tests unitarios implementados.

Recomendación para el futuro:
- Jest + React Testing Library
- Tests de integración con Firebase Emulator
- E2E tests con Playwright o Cypress

📖 Recursos y Documentación Adicional

Documentos del Proyecto

- `SEGURIDAD.md` - Estrategia completa de seguridad
- `CONDUCTORES-ESTADO.md` - Sistema de activación/desactivación
- `ELIMINACION-CONDUCTORES.md` - Guía de eliminación de conductores
- `FIREBASE-ADMIN-SETUP.md` - Configuración de Firebase Admin
- `GITHUB-ACTIONS-SETUP.md` - CI/CD con GitHub Actions
- `DESPLIEGUE-COMPLETO.md` - Guía completa de deployment
- `GESTION-CONDUCTORES.md` - Manual de gestión de conductores

Documentación Externa

Next.js:
- Next.js Docs (https://nextjs.org/docs)
- App Router Guide (https://nextjs.org/docs/app)

Firebase:
- Firebase Docs (https://firebase.google.com/docs)
- Firestore Security Rules (https://firebase.google.com/docs/firestore/security/get-started)
- Firebase Auth (https://firebase.google.com/docs/auth)

Tailwind CSS:
- Tailwind Docs (https://tailwindcss.com/docs)
- Tailwind UI (https://tailwindui.com/)

shadcn/ui:
- shadcn/ui Docs (https://ui.shadcn.com/)
- Radix UI Primitives (https://www.radix-ui.com/)

Comunidad y Soporte

Stack Overflow:
- Tag: `next.js`, `firebase`, `typescript`

Discord:
- Next.js Discord (https://nextjs.org/discord)
- Firebase Discord (https://discord.gg/firebase)

🔄 Flujos de Trabajo Principales

Flujo de Inspección Completa

```
1. Conductor inicia sesión
   ↓
2. Dashboard del conductor
   ↓
3. "Nueva Inspección" → Seleccionar vehículo
   ↓
4. Completar formulario:
   - Estado de salud
   - Destino y kilometraje
   - Inspección de items (6 categorías)
   - Capturar 4 fotos del vehículo
   - Firma digital
   - Observaciones
   ↓
5. Guardar inspección en Firestore
   ↓
6. Sistema analiza automáticamente:
   - ¿Hay items críticos en "Malo"? → Generar alerta de falla
   - ¿Documentos próximos a vencer? → Generar alerta de documento
   - ¿Documentos vencidos? → Generar alerta de alta prioridad
   ↓
7. Actualizar kilometraje del vehículo
   ↓
8. Confirmar éxito y redirigir
```

Flujo de Gestión de Alerta

```
1. Alerta generada automáticamente
   ↓
2. Aparece en dashboard del admin
   ↓
3. Admin accede al panel de alertas
   ↓
4. Revisar contexto de la alerta
   ↓
5. Marcar como "En proceso"
   ↓
6. Realizar acción correctiva:
   - Programar mantenimiento
   - Actualizar documento vencido
   - Reparar item fallido
   ↓
7. Resolver alerta con notas
   ↓
8. Alerta archivada como "Resuelta"
```

Flujo de Creación de Conductor

```
1. Admin → "Nuevo Conductor"
   ↓
2. Completar formulario
   ↓
3. Sistema crea usuario en Firebase Auth
   ↓
4. Obtener UID del usuario creado
   ↓
5. Crear documento en 'users' con el UID
   ↓
6. Crear documento en 'conductores' con referencia al userId
   ↓
7. Mostrar contraseña temporal al admin
   ↓
8. Admin entrega credenciales al conductor
   ↓
9. Conductor inicia sesión y cambia contraseña
```

🎨 Convenciones de Código

Nomenclatura

Archivos:
- Páginas: `kebab-case.tsx` (ej: `nueva-inspeccion.tsx`)
- Componentes: `PascalCase.tsx` (ej: `LoginForm.tsx`)
- Utilidades: `kebab-case.ts` (ej: `firebase-admin.ts`)

Variables y Funciones:

```
// Variables: camelCase
const userName = 'John';
const vehicleList = [];

// Constantes: UPPER_SNAKE_CASE
const MAX_FILE_SIZE = 5 * 1024 * 1024;
const API_ENDPOINT = '/api/upload';

// Funciones: camelCase
function handleSubmit() {}
async function fetchVehicles() {}

// Componentes React: PascalCase
function LoginForm() {}
export default function DashboardPage() {}
```

Tipos e Interfaces:

```
// PascalCase
interface User {}
type InspectionStatus = 'aprobada' | 'rechazada' | 'pendiente';
```

Estructura de Componentes

```
'use client';  // Si usa hooks o estado

// 1. Imports de React y Next.js
import { useState, useEffect } from 'react';
import { useRouter } from 'next/navigation';

// 2. Imports de Firebase
import { collection, getDocs } from 'firebase/firestore';
import { db } from '@/lib/firebase';

// 3. Imports de componentes UI
import { Button } from '@/components/ui/button';
import { Card } from '@/components/ui/card';

// 4. Imports de iconos
import { Car, User } from 'lucide-react';

// 5. Imports de utilidades
import { cn } from '@/lib/utils';

// 6. Tipos e interfaces
interface ComponentProps {
  // ...
}

// 7. Componente principal
export default function ComponentPage() {
  // Estado
  const [data, setData] = useState([]);

  // Hooks
  const router = useRouter();

  // Effects
  useEffect(() => {
    // ...
  }, []);

  // Handlers
  const handleClick = () => {
    // ...
  };

  // Render
  return (
    <div>
      {/* JSX */}
    </div>
  );
}
```

Estilos con Tailwind

```
// Usar cn() para combinar clases
import { cn } from '@/lib/utils';

<div className={cn(
  'base-classes',
  condition && 'conditional-classes',
  props.className
)} />

// Orden de clases:
// 1. Layout (flex, grid, etc.)
// 2. Tamaño (w-, h-, max-, min-)
// 3. Espaciado (p-, m-, gap-, space-)
// 4. Colores (bg-, text-, border-)
// 5. Tipografía (text-, font-)
// 6. Bordes (border-, rounded-)
// 7. Efectos (shadow-, opacity-, transition-)
// 8. Responsive (sm:, md:, lg:, xl:)

<div className="
  flex items-center justify-between
  w-full h-16
  px-4 py-2 gap-4
  bg-white text-gray-900
  text-sm font-medium
  border border-gray-200 rounded-lg
  shadow-sm hover:shadow-md transition-shadow
  md:px-6 lg:h-20
" />
```

Manejo de Errores

```
try {
  // Código que puede fallar
  await someAsyncOperation();
  toast.success('Operación exitosa');
} catch (error: any) {
  console.error('Error detallado:', error);

  // Manejo específico de errores de Firebase
  if (error.code === 'permission-denied') {
    toast.error('No tienes permisos para esta operación');
  } else if (error.code === 'not-found') {
    toast.error('Recurso no encontrado');
  } else {
    toast.error('Ocurrió un error. Intenta nuevamente.');
  }
} finally {
  setLoading(false);
}
```

🚧 Mejoras Futuras Sugeridas

Corto Plazo (1-2 meses)

1. **Testing:**
   - Implementar tests unitarios con Jest
   - Tests de integración con Firebase Emulator
   - E2E tests para flujos críticos

2. **Performance:**
   - Implementar paginación en listas largas
   - Lazy loading de imágenes
   - Optimización de queries a Firestore
   - Service Worker para PWA

3. **UX:**
   - Modo offline con sincronización
   - Notificaciones push
   - Búsqueda avanzada en todas las secciones
   - Exportación de reportes a PDF/Excel

Mediano Plazo (3-6 meses)

1. **Funcionalidades:**
   - Sistema de reportes avanzados
   - Gráficos y estadísticas detalladas
   - Programación de mantenimientos
   - Historial completo de cambios (audit log)
   - Sistema de notificaciones por email

2. **Roles Adicionales:**
   - Supervisor (acceso intermedio)
   - Mecánico (acceso a mantenimiento)
   - Visor (solo lectura)

3. **Integraciones:**
   - API REST para integraciones externas
   - Webhooks para eventos importantes
   - Integración con sistemas de flota

Largo Plazo (6+ meses)

1. **Escalabilidad:**
   - Migración a Cloud Functions para lógica backend
   - Implementar caching con Redis
   - CDN para assets estáticos

2. **Mobile:**
   - App nativa con React Native
   - Captura de fotos mejorada
   - Modo offline completo

3. **IA/ML:**
   - Detección automática de anomalías en fotos

- Predicción de mantenimientos
- Análisis de patrones de fallas

👥 Contacto y Soporte

Desarrollador Original: [Nelson Sanchez]

Repositorio: github.com/nelsonsanch/inspecciones-vehiculoso (https://github.com/nelsonsanch/inspecciones-vehiculoso)

Issues: Reportar bugs o sugerencias en GitHub Issues

Última Actualización: Noviembre 2025

📄 Licencia

[Especificar licencia del proyecto]

🎉 ¡Gracias por usar el Sistema de Inspecciones Vehiculares PESV!

Esta documentación está diseñada para servir como guía completa para cualquier desarrollador que continúe con el proyecto. Si encuentras errores o áreas que necesitan más clarificación, por favor actualiza este documento.