



Análisis del Proyecto DynamicFin CRM y Plan de Mejoras



Estado Actual del Proyecto

Tecnologías Identificadas:

- **Frontend:** Next.js 14 + React 18 + TypeScript
- **UI:** Tailwind CSS + Shadcn/ui + Framer Motion
- **Backend:** Next.js API Routes + NextAuth.js
- **Base de Datos:** PostgreSQL + Prisma ORM
- **Autenticación:** NextAuth.js con roles granulares

Estructura del Proyecto:

```
dynamicfin-crm/
├── app/
│   ├── api/                # API Routes
│   ├── auth/              # Autenticación
│   ├── dashboard/        # Dashboard principal
│   │   ├── gerente/      # Funciones gerenciales
│   │   ├── prospectos/   # Gestión de leads
│   │   ├── inventario/   # Gestión de vehículos
│   │   ├── usuarios/     # Gestión de usuarios
│   │   └── reportes/     # Reportes y análisis
│   ├── components/       # Componentes reutilizables
│   ├── prisma/           # Esquema de base de datos
│   └── lib/              # Utilidades y configuración
```

Funcionalidades Existentes:

- ✓ **Sistema SPCC** - 15 pilares de calificación de leads
- ✓ **Dashboard Gerencial** - KPIs y métricas en tiempo real
- ✓ **Gestión de Prospectos** - CRUD completo con clasificación automática
- ✓ **Sistema de Roles** - Control granular de acceso
- ✓ **Grabación de Conversaciones** - Con análisis de IA (componente ya implementado)
- ✓ **Coaching de Vendedores** - Sesiones programadas
- ✓ **Forecasting** - Proyecciones de ventas
- ✓ **Reasignación de Leads** - Control gerencial
- ✓ **Reportes Avanzados** - Análisis detallados



Plan de Implementación de Mejoras Solicitadas

1. 🚗 Catálogo Estandarizado de Vehículos

Estado: ⚠️ **NECESITA MEJORA**

- **Problema:** El modelo `Vehículo` actual es básico y no tiene estandarización
- **Solución:** Crear catálogo maestro con especificaciones técnicas completas

Implementación:

```
-- Nuevas tablas a agregar al schema.prisma
model CatalogoVehiculo {
  id          Int      @id @default(autoincrement())
  marca       String
  modelo      String
  year        Int
  version     String
  categoria   String   // "Sedan", "SUV", "Hatchback", etc.
  combustible String   // "Gasolina", "Híbrido", "Eléctrico"
  transmision String   // "Manual", "Automática", "CVT"
  cilindros   Int?
  potencia    String?  // "150 HP"
  torque      String?  // "200 Nm"
  rendimiento String?  // "15 km/l ciudad"
  dimensiones Json?    // {largo, ancho, alto, distancia_entre_ejes}
  capacidades Json?    // {tanque, cajuela, pasajeros}
  equipamiento Json?   // {seguridad, confort, tecnologia}
  coloresDisponibles Json? // ["Blanco", "Negro", "Plata"]
  precioBase  Decimal  @db.Decimal(12, 2)
  activo      Boolean  @default(true)
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt
}
```

Archivos a crear/modificar:

- app/dashboard/inventario/catalogo/page.tsx - Gestión del catálogo maestro
- components/vehiculos/CatalogoVehiculos.tsx - Componente de catálogo
- app/api/vehiculos/catalogo/route.ts - API para catálogo

2. 🎤 Corrección del Flujo de Grabación

Estado: ✅ **PARCIALMENTE IMPLEMENTADO**

- **Problema:** El componente existe pero puede tener issues de flujo
- **Solución:** Revisar y optimizar el componente GrabacionConversacion.tsx

Mejoras identificadas:

- ✅ Componente ya existe y está bien estructurado
- ⚠️ Necesita integración con API real (actualmente simulado)
- ⚠️ Falta manejo de errores más robusto
- ⚠️ Necesita optimización de almacenamiento de archivos

Archivos a modificar:

- components/GrabacionConversacion.tsx - Optimizar flujo existente
- app/api/grabaciones/route.ts - Crear API real para grabaciones
- lib/audio-processing.ts - Utilidades para procesamiento de audio

3. 👤 Nuevo Rol de Recepción

Estado: ❌ **NO IMPLEMENTADO**

- **Problema:** Solo existen roles de ventas y gerencia
- **Solución:** Agregar rol RECEPCION con permisos específicos

Implementación:

```
-- Modificar enum en schema.prisma
enum TipoRol {
  DIRECTOR_GENERAL
  DIRECTOR_MARCA
  GERENTE_GENERAL
  GERENTE_VENTAS
  VENDEDOR
  RECEPCION          // NUEVO ROL
  DYNAMICFIN_ADMIN
}
```

Funcionalidades del rol Recepción:

- ☒ Registrar leads iniciales
- ☒ Asignar leads a vendedores
- ☒ Ver calendario de citas
- ☒ Gestionar información básica de clientes
- ☒ NO puede ver comisiones
- ☒ NO puede acceder a reportes gerenciales
- ☒ NO puede reasignar leads (solo asignación inicial)

Archivos a crear:






- `app/dashboard/recepcion/page.tsx` - Dashboard específico para recepción
- `components/recepcion/AsignacionLeads.tsx` - Componente de asignación
- `middleware/auth-recepcion.ts` - Middleware de permisos

4. Dashboard de Seguimiento para Gerentes

Estado: ☒ **PARCIALMENTE IMPLEMENTADO**

- **Problema:** Existe dashboard gerencial pero necesita mejoras específicas
- **Solución:** Ampliar funcionalidades existentes

Mejoras a implementar:

-  **KPIs en Tiempo Real:** Métricas actualizadas cada 5 minutos
-  **Seguimiento de Metas:** Progreso individual y por equipo
-  **Alertas Inteligentes:** Notificaciones automáticas de situaciones críticas
-  **Vista Mobile:** Dashboard optimizado para dispositivos móviles
-  **Actualizaciones en Vivo:** WebSockets para datos en tiempo real

Archivos a crear/modificar:

- `app/dashboard/gerente/seguimiento/page.tsx` - Dashboard mejorado
- `components/gerente/KPIsRealTime.tsx` - KPIs en tiempo real
- `components/gerente/AlertasInteligentes.tsx` - Sistema de alertas
- `hooks/useRealTimeData.ts` - Hook para datos en tiempo real

Orden de Implementación Recomendado

Fase 1: Fundación (Semana 1)

1. ☒ Verificar y corregir flujo de grabación
2. ☒ Implementar rol de Recepción
3. ☒ Crear middleware de permisos

Fase 2: Catálogo (Semana 2)

1. ☒ Diseñar y crear catálogo estandarizado de vehículos
2. ☒ Migrar datos existentes al nuevo formato
3. ☒ Crear interfaces de gestión

Fase 3: Dashboard Avanzado (Semana 3)

1. ☒ Implementar KPIs en tiempo real
2. ☒ Crear sistema de alertas inteligentes
3. ☒ Optimizar para dispositivos móviles

Fase 4: Integración y Testing (Semana 4)

1. ☒ Pruebas integrales de todas las funcionalidades
 2. ☒ Optimización de rendimiento
 3. ☒ Documentación y capacitación
-

Checklist de Tareas Específicas

Catálogo de Vehículos:

- ☐ Crear modelo `CatalogoVehiculo` en Prisma
- ☐ Migrar datos existentes
- ☐ Crear API endpoints
- ☐ Desarrollar interfaz de gestión
- ☐ Implementar búsqueda y filtros avanzados

Flujo de Grabación:

- ☐ Revisar componente existente
- ☐ Implementar API real para grabaciones
- ☐ Optimizar almacenamiento de archivos
- ☐ Mejorar manejo de errores
- ☐ Agregar compresión de audio

Rol de Recepción:

- ☐ Modificar enum de roles en Prisma
- ☐ Crear dashboard específico
- ☐ Implementar permisos granulares
- ☐ Desarrollar componente de asignación de leads
- ☐ Crear middleware de autorización

Dashboard Gerencial:

- [] Implementar WebSockets para tiempo real
- [] Crear componentes de KPIs avanzados
- [] Desarrollar sistema de alertas
- [] Optimizar para mobile
- [] Agregar exportación de reportes



Herramientas y Dependencias Adicionales

Para Tiempo Real:

```
{
  "socket.io": "^4.7.2",
  "socket.io-client": "^4.7.2"
}
```

Para Procesamiento de Audio:

```
{
  "multer": "^1.4.5-lts.1",
  "@types/multer": "^1.4.7",
  "fluent-ffmpeg": "^2.1.2"
}
```

Para Análisis de Datos:

```
{
  "d3": "^7.8.5",
  "@types/d3": "^7.4.0",
  "recharts": "^2.8.0"
}
```



Estado del Repositorio

✓ REPOSITORIO CLONADO Y LISTO

- 📁 Ubicación: `/home/ubuntu/github_repos/dynamicfin-crm`
- 🛠️ Tecnologías: Next.js 14 + TypeScript + Prisma + PostgreSQL
- 📊 Base de datos: Esquema completo con 20+ modelos
- 🎯 Funcionalidades: 80% implementadas, necesita mejoras específicas



LISTO PARA COMENZAR IMPLEMENTACIÓN

El proyecto está bien estructurado y tiene una base sólida. Las mejoras solicitadas son factibles y se pueden implementar de manera incremental sin afectar la funcionalidad existente.