📋 NEXUS CRM - Instrucciones de Desarrollo

> ***Archivo de Continuidad:*** *Usa este documento para retomar el desarrollo desde cualquier dispositivo.*
> ***Última actualización:*** *15 de Diciembre, 2025 - 2:30 PM*
> ***Estado actual:*** *Beta funcional con webhook activo*
> ***GitHub:*** *https://github.com/lunacodeabit/nexusrd*

🎯 Visión General

¿Qué es NEXUS CRM?

Un CRM (Customer Relationship Management) diseñado específicamente para **agentes inmobiliarios en República Dominicana**. Reemplaza las hojas de Excel con una aplicación web moderna, móvil y automatizada.

Problema que Resuelve

- Leads perdidos por falta de seguimiento
- Datos dispersos en WhatsApp, Excel, notas
- Sin visibilidad de qué actividades generan resultados
- Proceso de calificación inconsistente

Usuario Objetivo

- Agentes inmobiliarios independientes
- Pequeñas/medianas agencias
- Asesores de inversión inmobiliaria

🚀 Cómo Iniciar el Proyecto

Requisitos

- Node.js 18+ instalado
- npm o pnpm
- VS Code (recomendado)
- Docker (opcional, para n8n local)

Comandos para Iniciar

```
# 1. Navegar al proyecto
cd C:\Users\howar\OneDrive\Desktop\NEXUSRD

# 2. Instalar dependencias (si es primera vez o nuevo dispositivo)
npm install

# 3. Iniciar servidor de desarrollo
npm run dev

# 4. Abrir en navegador
# http://localhost:5173 (o el puerto que indique)
```

Estructura del Proyecto

```
NEXUSRD/
├── src/
│   ├── components/       # Componentes React
│   ├── services/         # Lógica de negocio (scoring, export, etc.)
│   ├── types/            # TypeScript interfaces
│   ├── App.tsx           # Componente principal
│   └── main.tsx          # Entry point
├── public/               # Assets estáticos, PWA files
├── NEXUS_CRM_ROADMAP.md  # Roadmap técnico completo
├── INSTRUCTIONS.md       # Este archivo
└── package.json          # Dependencias
```

✅ Pasos Completados

Fase 1: Setup Inicial ✓

- [x] Proyecto Vite + React + TypeScript creado
- [x] Tailwind CSS configurado con colores custom (nexus-base, nexus-surface, nexus-accent)
- [x] Estructura de carpetas organizada
- [x] PWA configurado (manifest.json, service worker)

Fase 2: Core CRM ✓

- [x] Dashboard con KPIs
- [x] Gestión de leads (crear, ver, editar)
- [x] Gestión de propiedades/inventario
- [x] Navegación móvil (bottom tabs)
- [x] Modales y formularios funcionales

Fase 3: Sistema de Calificación ✓

- [x] 10 preguntas de calificación implementadas
- [x] Algoritmo de scoring por puntos
- [x] Categorías HOT 🔥 / WARM 🌡️ / COLD ❄️
- [x] Indicadores visuales en tarjetas de leads

Fase 4: Actividades Diarias ✓

- [x] Checklist por horarios (8AM, 8:30AM, 3PM, 6PM)
- [x] Categorías: Posts, Subidas, Aprendizaje, Prospección
- [x] Progreso diario/semanal
- [x] Navegación por fechas

Fase 5: Seguimiento S1-S12 ✓

- [x] Barra de progreso de seguimientos
- [x] Registro de método (WhatsApp, Llamada, Email, Visita)
- [x] Estado de respuesta
- [x] Notas por seguimiento
- [x] Timeline cronológico

Fase 6: Analytics ✓

- [x] Gráficos de correlación actividad-resultados
- [x] Efectividad por número de seguimiento
- [x] Distribución de leads por estado/score
- [x] Insights generados
- [x] Exportar a CSV

Fase 7: Notificaciones ✓

- [x] Permisos de notificación del navegador
- [x] Sonidos via Web Audio API (sin archivos externos)
- [x] Sonido urgente para leads HOT
- [x] Sonido de éxito para completar tareas

Fase 8: Webhook n8n ✓

- [x] Workflow en n8n configurado y ACTIVO
- [x] URL: `https://n8n.srv806559.hstgr.cloud/webhook/nexus-lead`
- [x] Email de notificación con formato HTML
- [x] Subject dinámico con nombre y fuente
- [x] Sección en app con botón de copiar URL

⏳ Pasos Pendientes (Por Orden de Prioridad)

🔴 Alta Prioridad

1. Google Sheets Integration (n8n)

Estado: Pendiente
Tiempo estimado: 30 minutos
Instrucciones:

1. Abrir n8n: `https://n8n.srv806559.hstgr.cloud`
2. Editar workflow "NEXUS CRM - Leads"
3. Agregar nodo "Google Sheets" después de "Edit Fields"
4. Conectar credenciales de Google
5. Configurar: Append Row a un spreadsheet nuevo
6. Mapear campos: name, phone, email, source, message, fecha
7. Guardar y probar

2. WhatsApp Business API

Estado: Pendiente (requiere setup de Meta)
Tiempo estimado: 2-4 horas
Prerrequisitos:

- [ ] Cuenta de Meta Business verificada
- [ ] Número de teléfono NO registrado en WhatsApp personal
- [ ] App creada en developers.facebook.com

Pasos cuando esté listo:

1. Crear app en Meta for Developers
2. Agregar producto WhatsApp
3. Obtener Phone Number ID y Access Token
4. Agregar nodo WhatsApp Cloud API en n8n
5. Crear template de mensaje de bienvenida
6. Conectar al workflow

3. Backend + Persistencia de Datos

Estado: Pendiente
Tiempo estimado: 4-6 horas
Opción recomendada: Supabase
Pasos:

1. Crear cuenta en supabase.com
2. Crear proyecto nuevo
3. Diseñar tablas: leads, properties, activities, follow_ups
4. Configurar Row Level Security (RLS)
5. Obtener API keys
6. Instalar @supabase/supabase-js en el proyecto
7. Reemplazar estado local con queries a Supabase

4. Autenticación de Usuarios

Estado: Pendiente
Tiempo estimado: 3-4 horas
Después de: Backend configurado
Pasos:

1. Habilitar Auth en Supabase
2. Crear componente de Login/Register
3. Implementar contexto de autenticación
4. Proteger rutas
5. Asociar leads al usuario autenticado

🟡 Media Prioridad

5. Despliegue a Producción

Opciones:

- Vercel (recomendado, gratis)
- Netlify (alternativa)
- Hostinger (si ya tienes)

Pasos para Vercel:

1. Push código a GitHub
2. Conectar repo en vercel.com
3. Deploy automático
4. Configurar dominio custom (opcional)

6. Importar Leads desde CSV

Tiempo estimado: 2 horas

- Componente de upload

- Parser de CSV
- Mapeo de columnas
- Validación de datos

7. Integración con Portales Reales

- Investigar API de SuperCasas
- Configurar Facebook Lead Ads webhook
- Conectar Instagram Lead Forms

🟢 Baja Prioridad (Nice to Have)

- [ ] Tema claro/oscuro toggle
- [ ] Multi-idioma (EN/ES)
- [ ] Calculadora de comisiones
- [ ] Calendario con citas
- [ ] Notas de voz
- [ ] Generación de PDF reports

🗄️ BACKEND COMPLETO CON SUPABASE (Guía Detallada)

¿Por qué Supabase?

Característica	Beneficio
PostgreSQL	Base de datos robusta y escalable
Auth integrado	Login con email, Google, Facebook
Row Level Security	Seguridad a nivel de fila
Realtime	Actualizaciones en tiempo real
Storage	Para fotos de propiedades
Edge Functions	Para lógica del servidor
Gratis	Hasta 500MB y 50k usuarios

Paso 1: Crear Cuenta y Proyecto

1. **Ir a:** https://supabase.com
2. **Sign up** con GitHub (recomendado)
3. **New Project:**
   - Name: `nexus-crm`
   - Database Password: (guardar en lugar seguro)
   - Region: `East US` o el más cercano
4. **Esperar** ~2 minutos mientras se crea

Paso 2: Obtener Credenciales

En el dashboard de Supabase:

1. Ir a **Settings** → **API**
2. Copiar:

- `Project URL` → será tu `VITE_SUPABASE_URL`
- `anon public` key → será tu `VITE_SUPABASE_ANON_KEY`

Paso 3: Crear Tablas (SQL)

Ir a **SQL Editor** y ejecutar este script:

```
-- ============================================
-- NEXUS CRM - DATABASE SCHEMA
-- ============================================

-- 1. TABLA DE USUARIOS (extends auth.users)
CREATE TABLE public.profiles (
  id UUID REFERENCES auth.users(id) PRIMARY KEY,
  full_name TEXT,
  phone TEXT,
  avatar_url TEXT,
  role TEXT DEFAULT 'agent', -- agent, admin, broker
  agency_name TEXT,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 2. TABLA DE LEADS
CREATE TABLE public.leads (
  id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
  user_id UUID REFERENCES auth.users(id) NOT NULL,

  -- Info básica
  name TEXT NOT NULL,
  phone TEXT,
  email TEXT,
  source TEXT, -- Facebook, Instagram, SuperCasas, Referido, etc.

  -- Calificación
  status TEXT DEFAULT 'NEW', -- NEW, CONTACTED, VISIT_SCHEDULED, NEGOTIATING, CLOSED_WON,
CLOSED_LOST
  score INTEGER DEFAULT 0, -- 0-100
  score_category TEXT, -- HOT, WARM, COLD
  qualification_answers JSONB, -- Respuestas de las 10 preguntas

  -- Preferencias
  budget_min DECIMAL,
  budget_max DECIMAL,
  preferred_zones TEXT[],
  property_type TEXT, -- Apartamento, Casa, Villa, etc.
  bedrooms INTEGER,

  -- Seguimiento
  current_follow_up INTEGER DEFAULT 0, -- S1-S12
  next_follow_up_date TIMESTAMP WITH TIME ZONE,
```

```sql
  -- Notas
  notes TEXT,

  -- Timestamps
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  closed_at TIMESTAMP WITH TIME ZONE
);

-- 3. TABLA DE SEGUIMIENTOS (S1-S12)
CREATE TABLE public.follow_ups (
  id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
  lead_id UUID REFERENCES public.leads(id) ON DELETE CASCADE,
  user_id UUID REFERENCES auth.users(id) NOT NULL,

  follow_up_number INTEGER NOT NULL, -- 1-12
  method TEXT NOT NULL, -- WHATSAPP, CALL, EMAIL, VISIT, OTHER
  response TEXT, -- RESPONDED, NO_ANSWER, PENDING
  notes TEXT,

  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 4. TABLA DE PROPIEDADES
CREATE TABLE public.properties (
  id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
  user_id UUID REFERENCES auth.users(id) NOT NULL,

  -- Info básica
  title TEXT NOT NULL,
  description TEXT,
  property_type TEXT, -- Apartamento, Casa, Villa, Solar, Local
  status TEXT DEFAULT 'AVAILABLE', -- AVAILABLE, RESERVED, SOLD, RENTED

  -- Ubicación
  address TEXT,
  city TEXT,
  zone TEXT,
  coordinates JSONB, -- {lat, lng}

  -- Características
  price DECIMAL NOT NULL,
  currency TEXT DEFAULT 'USD',
  bedrooms INTEGER,
  bathrooms INTEGER,
  parking_spots INTEGER,
  area_m2 DECIMAL,
  amenities TEXT[],

  -- Comisión
  commission_percentage DECIMAL DEFAULT 3.0,
  commission_split JSONB, -- {agent: 50, broker: 50}
```

```sql
  -- Dueño
  owner_name TEXT,
  owner_phone TEXT,
  owner_email TEXT,

  -- Media
  images TEXT[], -- URLs de imágenes
  video_url TEXT,
  virtual_tour_url TEXT,

  -- Timestamps
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 5. TABLA DE ACTIVIDADES DIARIAS
CREATE TABLE public.daily_activities (
  id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
  user_id UUID REFERENCES auth.users(id) NOT NULL,

  task_id TEXT NOT NULL, -- ID de la tarea del checklist
  date DATE NOT NULL,
  day_of_week TEXT NOT NULL,
  completed BOOLEAN DEFAULT FALSE,
  completed_at TIMESTAMP WITH TIME ZONE,

  UNIQUE(user_id, task_id, date)
);

-- 6. TABLA DE INTERACCIONES/LOG
CREATE TABLE public.interactions (
  id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
  user_id UUID REFERENCES auth.users(id) NOT NULL,
  lead_id UUID REFERENCES public.leads(id) ON DELETE CASCADE,
  property_id UUID REFERENCES public.properties(id) ON DELETE SET NULL,

  type TEXT NOT NULL, -- CALL, EMAIL, WHATSAPP, VISIT, NOTE, STATUS_CHANGE
  description TEXT,
  metadata JSONB,

  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 7. TABLA DE WEBHOOKS/LEADS ENTRANTES
CREATE TABLE public.webhook_leads (
  id UUID DEFAULT gen_random_uuid() PRIMARY KEY,

  -- Datos raw del webhook
  raw_data JSONB NOT NULL,
  source TEXT, -- n8n, facebook, supercasas
```

```
  -- Datos normalizados
  name TEXT,
  phone TEXT,
  email TEXT,
  message TEXT,

  -- Estado de procesamiento
  processed BOOLEAN DEFAULT FALSE,
  assigned_to UUID REFERENCES auth.users(id),
  converted_lead_id UUID REFERENCES public.leads(id),

  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- ============================================
-- ÍNDICES PARA PERFORMANCE
-- ============================================

CREATE INDEX idx_leads_user_id ON public.leads(user_id);
CREATE INDEX idx_leads_status ON public.leads(status);
CREATE INDEX idx_leads_score_category ON public.leads(score_category);
CREATE INDEX idx_leads_next_follow_up ON public.leads(next_follow_up_date);
CREATE INDEX idx_leads_created_at ON public.leads(created_at DESC);

CREATE INDEX idx_follow_ups_lead_id ON public.follow_ups(lead_id);
CREATE INDEX idx_follow_ups_created_at ON public.follow_ups(created_at DESC);

CREATE INDEX idx_properties_user_id ON public.properties(user_id);
CREATE INDEX idx_properties_status ON public.properties(status);
CREATE INDEX idx_properties_zone ON public.properties(zone);

CREATE INDEX idx_daily_activities_user_date ON public.daily_activities(user_id, date);

CREATE INDEX idx_webhook_leads_processed ON public.webhook_leads(processed);

-- ============================================
-- ROW LEVEL SECURITY (RLS)
-- ============================================

-- Habilitar RLS en todas las tablas
ALTER TABLE public.profiles ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.leads ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.follow_ups ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.properties ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.daily_activities ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.interactions ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.webhook_leads ENABLE ROW LEVEL SECURITY;

-- Políticas: Usuarios solo ven sus propios datos
CREATE POLICY "Users can view own profile" ON public.profiles
  FOR ALL USING (auth.uid() = id);
```

```sql
CREATE POLICY "Users can manage own leads" ON public.leads
  FOR ALL USING (auth.uid() = user_id);

CREATE POLICY "Users can manage own follow_ups" ON public.follow_ups
  FOR ALL USING (auth.uid() = user_id);

CREATE POLICY "Users can manage own properties" ON public.properties
  FOR ALL USING (auth.uid() = user_id);

CREATE POLICY "Users can manage own activities" ON public.daily_activities
  FOR ALL USING (auth.uid() = user_id);

CREATE POLICY "Users can view own interactions" ON public.interactions
  FOR ALL USING (auth.uid() = user_id);

-- Webhook leads: cualquiera puede insertar, solo admin procesa
CREATE POLICY "Anyone can insert webhook leads" ON public.webhook_leads
  FOR INSERT WITH CHECK (true);

CREATE POLICY "Users can view unassigned or own webhook leads" ON public.webhook_leads
  FOR SELECT USING (assigned_to IS NULL OR auth.uid() = assigned_to);

-- ============================================
-- FUNCIONES Y TRIGGERS
-- ============================================

-- Auto-crear perfil cuando se registra usuario
CREATE OR REPLACE FUNCTION public.handle_new_user()
RETURNS TRIGGER AS $$
BEGIN
  INSERT INTO public.profiles (id, full_name)
  VALUES (NEW.id, NEW.raw_user_meta_data->>'full_name');
  RETURN NEW;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE TRIGGER on_auth_user_created
  AFTER INSERT ON auth.users
  FOR EACH ROW EXECUTE FUNCTION public.handle_new_user();

-- Auto-actualizar updated_at
CREATE OR REPLACE FUNCTION public.update_updated_at()
RETURNS TRIGGER AS $$
BEGIN
  NEW.updated_at = NOW();
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_leads_updated_at
  BEFORE UPDATE ON public.leads
  FOR EACH ROW EXECUTE FUNCTION public.update_updated_at();
```

```sql
CREATE TRIGGER update_properties_updated_at
  BEFORE UPDATE ON public.properties
  FOR EACH ROW EXECUTE FUNCTION public.update_updated_at();

CREATE TRIGGER update_profiles_updated_at
  BEFORE UPDATE ON public.profiles
  FOR EACH ROW EXECUTE FUNCTION public.update_updated_at();
```

Paso 4: Instalar Supabase en el Proyecto

```
cd C:\Users\howar\OneDrive\Desktop\NEXUSRD
npm install @supabase/supabase-js
```

Paso 5: Crear archivo de configuración

Crear: `src/lib/supabase.ts`

```typescript
import { createClient } from '@supabase/supabase-js';

const supabaseUrl = import.meta.env.VITE_SUPABASE_URL;
const supabaseAnonKey = import.meta.env.VITE_SUPABASE_ANON_KEY;

if (!supabaseUrl || !supabaseAnonKey) {
  throw new Error('Missing Supabase environment variables');
}

export const supabase = createClient(supabaseUrl, supabaseAnonKey);
```

Paso 6: Crear archivo .env.local

```
VITE_SUPABASE_URL=https://xxxxx.supabase.co
VITE_SUPABASE_ANON_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Paso 7: Crear Hooks para datos

Crear: `src/hooks/useLeads.ts`

```typescript
import { useState, useEffect } from 'react';
import { supabase } from '../lib/supabase';
import type { Lead } from '../types';

export function useLeads() {
  const [leads, setLeads] = useState<Lead[]>([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<Error | null>(null);

  useEffect(() => {
```

```
  fetchLeads();

  // Suscripción a cambios en tiempo real
  const subscription = supabase
    .channel('leads')
    .on('postgres_changes', { event: '*', schema: 'public', table: 'leads' },
      () => fetchLeads()
    )
    .subscribe();

  return () => {
    subscription.unsubscribe();
  };
}, []);

const fetchLeads = async () => {
  try {
    const { data, error } = await supabase
      .from('leads')
      .select('*')
      .order('created_at', { ascending: false });

    if (error) throw error;
    setLeads(data || []);
  } catch (e) {
    setError(e as Error);
  } finally {
    setLoading(false);
  }
};

const addLead = async (lead: Omit<Lead, 'id'>) => {
  const { data, error } = await supabase
    .from('leads')
    .insert([lead])
    .select()
    .single();

  if (error) throw error;
  return data;
};

const updateLead = async (id: string, updates: Partial<Lead>) => {
  const { data, error } = await supabase
    .from('leads')
    .update(updates)
    .eq('id', id)
    .select()
    .single();

  if (error) throw error;
  return data;
```

```
  };

  const deleteLead = async (id: string) => {
    const { error } = await supabase
      .from('leads')
      .delete()
      .eq('id', id);

    if (error) throw error;
  };

  return { leads, loading, error, addLead, updateLead, deleteLead, refetch: fetchLeads };
}
```

Paso 8: Crear Contexto de Auth

Crear: `src/contexts/AuthContext.tsx`

```
import React, { createContext, useContext, useEffect, useState } from 'react';
import { supabase } from '../lib/supabase';
import type { User, Session } from '@supabase/supabase-js';

interface AuthContextType {
  user: User | null;
  session: Session | null;
  loading: boolean;
  signIn: (email: string, password: string) => Promise<void>;
  signUp: (email: string, password: string, fullName: string) => Promise<void>;
  signOut: () => Promise<void>;
}

const AuthContext = createContext<AuthContextType | undefined>(undefined);

export function AuthProvider({ children }: { children: React.ReactNode }) {
  const [user, setUser] = useState<User | null>(null);
  const [session, setSession] = useState<Session | null>(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    // Obtener sesión actual
    supabase.auth.getSession().then(({ data: { session } }) => {
      setSession(session);
      setUser(session?.user ?? null);
      setLoading(false);
    });

    // Escuchar cambios de auth
    const { data: { subscription } } = supabase.auth.onAuthStateChange(
      (_event, session) => {
        setSession(session);
        setUser(session?.user ?? null);
```

```
      }
    );

    return () => subscription.unsubscribe();
  }, []);

  const signIn = async (email: string, password: string) => {
    const { error } = await supabase.auth.signInWithPassword({ email, password });
    if (error) throw error;
  };

  const signUp = async (email: string, password: string, fullName: string) => {
    const { error } = await supabase.auth.signUp({
      email,
      password,
      options: { data: { full_name: fullName } }
    });
    if (error) throw error;
  };

  const signOut = async () => {
    const { error } = await supabase.auth.signOut();
    if (error) throw error;
  };

  return (
    <AuthContext.Provider value={{ user, session, loading, signIn, signUp, signOut }}>
      {children}
    </AuthContext.Provider>
  );
}

export const useAuth = () => {
  const context = useContext(AuthContext);
  if (!context) throw new Error('useAuth must be used within AuthProvider');
  return context;
};
```

Paso 9: Actualizar n8n para guardar en Supabase

En tu workflow de n8n, después del nodo de Gmail, agregar:

1. **Nodo HTTP Request** para insertar en Supabase:
   - Method: POST
   - URL: `https://tu-proyecto.supabase.co/rest/v1/webhook_leads`
   - Headers:
     - `apikey` : tu anon key
     - `Authorization` : `Bearer tu_anon_key`
     - `Content-Type` : `application/json`
   - Body:

```
{
  "raw_data": {{ $json }},
  "source": "n8n",
  "name": "{{ $json.name }}",
  "phone": "{{ $json.phone }}",
  "email": "{{ $json.email }}",
  "message": "{{ $json.message }}"
}
```

📱 WHATSAPP BUSINESS API (Guía Completa)

Requisitos Previos (Checklist)

- [ ] Cuenta de Facebook personal activa
- [ ] Página de Facebook de tu negocio
- [ ] Cuenta de Meta Business Suite (business.facebook.com)
- [ ] Número de teléfono que NO esté en WhatsApp personal
- [ ] Verificación de negocio (puede tomar 1-7 días)

Paso 1: Crear App en Meta for Developers

1. Ir a: https://developers.facebook.com
2. Click en "My Apps" → "Create App"
3. Seleccionar: **Business** type
4. Llenar:
   - App Name: `NEXUS CRM WhatsApp`
   - Contact Email: tu email
   - Business Account: seleccionar o crear
5. Click "Create App"

Paso 2: Agregar WhatsApp a la App

1. En el dashboard, buscar "WhatsApp" en productos
2. Click "Set up"
3. Seleccionar tu Business Account
4. Te asignarán un **número de prueba gratuito**

Paso 3: Obtener Credenciales

En WhatsApp → API Setup, copiar:

- **Phone Number ID:** `1234567890123456`
- **WhatsApp Business Account ID:** `9876543210`
- **Temporary Access Token:** `EAAxxxxxx...` (válido 24h)

Para token permanente:

1. Ir a Business Settings → System Users
2. Crear System User
3. Generar token con permisos de WhatsApp

Paso 4: Crear Template de Mensaje

Los mensajes proactivos requieren templates aprobados:

1. WhatsApp Manager → Message Templates
2. Create Template:
   - Name: `nuevo_lead_bienvenida`
   - Category: `Marketing`
   - Language: `Spanish`
   - Body:

```
¡Hola {{1}}! 👋

Gracias por contactarnos. Soy {{2}} de NEXUS Real Estate.

Recibí tu mensaje sobre propiedades en República Dominicana.

¿Te gustaría agendar una llamada para conocer mejor tus necesidades?

🏠 Tenemos opciones increíbles para ti.
```

3. Submit para aprobación (24-48h)

Paso 5: Configurar en n8n

1. Agregar nodo "HTTP Request" después de Gmail
2. Configurar:
   - Method: POST
   - URL: `https://graph.facebook.com/v18.0/YOUR_PHONE_NUMBER_ID/messages`
   - Headers:
     - Authorization: `Bearer YOUR_ACCESS_TOKEN`
     - Content-Type: `application/json`
   - Body:

```
{
  "messaging_product": "whatsapp",
  "to": "{{ $json.phone.replace(/[^0-9]/g, '') }}",
  "type": "template",
  "template": {
    "name": "nuevo_lead_bienvenida",
    "language": { "code": "es" },
    "components": [
      {
        "type": "body",
        "parameters": [
          { "type": "text", "text": "{{ $json.name }}" },
          { "type": "text", "text": "Howard" }
        ]
      }
    ]
  }
}
```

Costos de WhatsApp Business API

Tipo	Costo (USD)
Business-initiated (templates)	~$0.05-0.15/msg
User-initiated (respuestas 24h)	Gratis
Primeros 1000 mensajes/mes	Gratis

📊 GOOGLE SHEETS INTEGRATION (n8n)

Paso 1: Crear Spreadsheet

1. Ir a https://sheets.google.com
2. Crear nuevo: "NEXUS CRM - Leads Webhook"
3. En la primera fila, agregar headers:

A	B	C	D	E	F	G
Fecha	Nombre	Teléfono	Email	Fuente	Mensaje	Procesado

Paso 2: Configurar en n8n

1. Abrir workflow en n8n
2. Después de "Edit Fields", agregar nodo "Google Sheets"
3. Configurar:
   - Credential: Conectar cuenta Google
   - Operation: **Append Row**
   - Document: Seleccionar "NEXUS CRM - Leads Webhook"
   - Sheet: Sheet1
4. Mapear columnas:
   - A (Fecha): `{{ $now.toISO() }}`
   - B (Nombre): `{{ $json.name }}`
   - C (Teléfono): `{{ $json.phone }}`
   - D (Email): `{{ $json.email }}`
   - E (Fuente): `{{ $json.source }}`
   - F (Mensaje): `{{ $json.message }}`
   - G (Procesado): `No`

Paso 3: Workflow Final

```
[Webhook] → [Edit Fields] → [Google Sheets] → [Gmail] → [WhatsApp (opcional)]
```

🔧 Configuraciones Importantes

URLs y Credenciales

Servicio	URL/Valor
n8n Dashboard	`https://n8n.srv806559.hstgr.cloud`

Webhook URL	`https://n8n.srv806559.hstgr.cloud/webhook/nexus-lead`
Dev Server	`http://localhost:5173` (default Vite)
Email notificaciones	howard@alveare.do

Variables de Entorno (Cuando agregues backend)

```
# .env.local (crear este archivo)
VITE_SUPABASE_URL=tu_url_aqui
VITE_SUPABASE_ANON_KEY=tu_key_aqui
VITE_WEBHOOK_URL=https://n8n.srv806559.hstgr.cloud/webhook/nexus-lead
```

💡 Sugerencias para Mejorar el Proyecto

Corto Plazo (Esta semana)

1. **Agregar localStorage** - Persistir datos básicos sin backend
2. **Mejorar iconos PWA** - Diseñar iconos profesionales
3. **Validación de formularios** - Mensajes de error claros
4. **Loading states** - Skeletons mientras carga

Mediano Plazo (Este mes)

1. **Tests unitarios** - Jest + React Testing Library
2. **Error boundaries** - Manejo de errores graceful
3. **Analytics reales** - Google Analytics o Plausible
4. **SEO básico** - Meta tags para compartir

Largo Plazo (Próximos 3 meses)

1. **App nativa** - React Native o Capacitor
2. **AI Assistant** - Sugerencias de siguiente acción
3. **White-label system** - Multi-tenant para revender
4. **Marketplace de integraciones** - Plugins de terceros

🧪 Comandos Útiles

```
# Desarrollo
npm run dev              # Iniciar servidor dev
npm run build            # Crear build de producción
npm run preview          # Preview del build

# Testing (cuando se agregue)
npm run test             # Correr tests
npm run test:coverage    # Tests con coverage

# Linting
npm run lint             # Revisar código
```

```
# Git (recomendado configurar)
git add .
git commit -m "descripción del cambio"
git push origin main
```

🐛 Problemas Conocidos y Soluciones

Error: "Cannot find module..."

```
# Solución: Reinstalar dependencias
rm -rf node_modules
npm install
```

Error: TypeScript enums

Problema: `verbatimModuleSyntax` no permite enums
Solución: Usar `const objects as const` en lugar de enums

Webhook no recibe datos

Verificar:

1. Workflow activo en n8n (toggle verde)
2. URL correcta (production, no test)
3. Content-Type: application/json en el request

Notificaciones no suenan

Verificar:

1. Navegador permite audio (click primero en la página)
2. Volumen del sistema no está en mute
3. Permisos de notificación otorgados

📞 Retomar Desarrollo

Checklist para Continuar

1. ☐ Abrir VS Code en `C:\Users\howar\OneDrive\Desktop\NEXUSRD`
2. ☐ Ejecutar `npm run dev`
3. ☐ Leer este archivo para recordar contexto
4. ☐ Revisar NEXUS_CRM_ROADMAP.md para detalles técnicos
5. ☐ Verificar que n8n workflow sigue activo
6. ☐ Continuar con el siguiente paso pendiente

Contexto para IA (Copilot/ChatGPT)

Si necesitas ayuda de una IA, copia este contexto:

```
Estoy desarrollando NEXUS CRM, un CRM inmobiliario en React + TypeScript + Vite.
Stack: React 19, TypeScript 5.9, Tailwind CSS, Recharts, Lucide icons.
```

```
Backend pendiente (planificado Supabase).
Automatización con n8n (webhook activo).

Características completadas:
- Lead management con scoring (HOT/WARM/COLD)
- Daily activities tracker (checklist por horarios)
- Follow-up tracking S1-S12
- Analytics con correlación
- Notificaciones con sonido
- Webhook para recibir leads externos

Pendiente:
- Google Sheets integration
- WhatsApp Business API
- Backend con Supabase
- Autenticación
- Deploy a producción

El código está en: C:\Users\howar\OneDrive\Desktop\NEXUSRD
Documentación en: NEXUS_CRM_ROADMAP.md e INSTRUCTIONS.md
```

🗓️ Historial de Sesiones

Sesión 1 - 15 Dic 2025

Duración: ~4 horas
Logros:

- Setup completo del proyecto desde Google AI Studio
- Todas las fases 1-8 completadas
- Sistema de calificación de leads (10 preguntas)
- Tracker de actividades diarias
- Seguimiento S1-S12
- Analytics con correlación
- Notificaciones con sonido
- Webhook n8n funcionando con email
- Documentación completa
- Push a GitHub: https://github.com/lunacodeabit/nexusrd

Pendiente para próxima sesión:

1. Google Sheets en n8n (30 min)
2. WhatsApp Business API (2-4 horas)
3. Backend Supabase (4-6 horas)
4. Deploy a Vercel

🎯 RESUMEN EJECUTIVO - Estado del Proyecto

✅ Lo que FUNCIONA ahora:

- App completa en localhost
- Webhook recibe leads 24/7

- Email de notificación instantáneo
- Sistema de scoring y seguimiento

⚠️ Lo que FALTA para producción:

1. **Backend (Supabase)** - Sin esto, los datos se pierden al refrescar
2. **Auth** - Sin esto, cualquiera puede ver los datos
3. **Deploy** - Sin esto, solo funciona en tu computadora
4. **WhatsApp** - Opcional pero muy útil

📊 Estimación de tiempo total pendiente:

Tarea	Tiempo	Prioridad
Google Sheets	30 min	Alta
Supabase setup	2 horas	Crítica
Auth implementation	3 horas	Crítica
Conectar app a Supabase	4 horas	Crítica
Deploy Vercel	30 min	Alta
WhatsApp API	3 horas	Media
TOTAL	**~13 horas**	

Actualiza este archivo cada vez que termines una sesión de desarrollo.