Nombre del Proyecto: SIG-CM (Sistema Integral de Gestión de Clasificados Multicanal)

1. Resumen Ejecutivo

Desarrollo de una plataforma tecnológica de alto rendimiento para la gestión, publicación y cobro de avisos clasificados. El sistema unifica tres canales de venta (Web, Mostrador Físico y Diario Impreso) bajo una única lógica de negocio centralizada, pero con interfaces específicas para cada tipo de usuario.

El núcleo del sistema destaca por su **flexibilidad taxonómica**, permitiendo a la administración modelar rubros, subrubros y operaciones comerciales de forma dinámica sin intervención de programadores.

---

2. Arquitectura Tecnológica (Stack Definido)

El proyecto abandona las soluciones “Low-Code” para adoptar una arquitectura robusta, compilada y de tipado estático, garantizando escalabilidad, seguridad y performance milimétrica.

A. Capa de Datos (Persistencia)

- **Motor: Microsoft SQL Server**.
- **Diseño:** Modelo Relacional Normalizado para integridad transaccional estricta (Pagos/Facturación) combinado con búsquedas full-text para el catálogo.

B. Backend (API & Lógica de Negocio)

- **Framework: .NET 10 (C#) - ASP.NET Core Web API**.
- **Acceso a Datos: Dapper** (Micro-ORM).
    - Justificación: A diferencia de Entity Framework, Dapper permite escribir SQL crudo optimizado, crucial para filtros complejos de clasificados y reportes de gran volumen, manteniendo la velocidad de ejecución casi nativa.
- **Autenticación:** JWT (JSON Web Tokens) con Roles (Admin, Cajero, Usuario, Diagramador).

C. Frontend (Interfaces de Usuario)

- **Core: React + Vite**.
- **Lenguaje: TypeScript** (Estricto).

- **Estado Global:** Zustand o Redux Toolkit.
- **Estilos:** Tailwind CSS (para desarrollo ágil y diseño consistente).

---

3. Módulos del Sistema (Frontends)

El sistema se compone de 4 aplicaciones cliente (SPA) que consumen la misma API:

I. Panel de Administración (El “Cerebro”)

Reemplazo total de Directus. Enfocado en la flexibilidad total. * **Gestor de Taxonomía Dinámica (Árbol de Categorías):** * Interfaz “Drag & Drop” para crear Rubros y Subrubros infinitos (Niveles ilimitados). * Capacidad de **Unir** dos categorías (Merge) o **Dividir** una categoría migrando los avisos existentes. * **Matriz de Operaciones:** * Definición de Operaciones (Venta, Alquiler, Pedido, Ofrecido). * **Asignación Condicional:** UI para tildar qué operaciones aplican a qué rubro. * Ejemplo: “Inmuebles” habilita [Venta, Alquiler]. “Empleos” habilita [Pedido, Ofrecido]. * **Moderación de Avisos:** Bandeja de entrada para aprobar/rechazar avisos con alertas de palabras prohibidas. * **Módulo de Diagramación (Exportación):** * Filtro por fecha de edición. * Generación de TXT/XML para Adobe InDesign. * Reporte de ocupación (cm² vendidos en papel).

II. Web Pública (Portal de Clasificados)

- **Motor de Búsqueda Facetado:** Filtros laterales dinámicos generados según el rubro seleccionado (si elijo Autos, aparecen filtros de Km y Año; si elijo Inmuebles, aparecen Ambientes).
- **Landing Page:** Diseño optimizado para SEO, carga instantánea (Vite).
- **Ficha de Aviso:** Galería de fotos, contacto (WhatsApp/Email), mapas.
- **Perfil de Usuario:** “Mis Avisos”, historial de pagos, renovación de avisos vencidos.

III. Web de Publicación (Wizard de Auto-gestión)

- **Flujo paso a paso:**
  1. Selección de Rubro (Cascada).
  2. Selección de Operación (Filtrada por Rubro).
  3. Carga de datos Web (Fotos, HTML rich text).
  4. **Upselling Papel:** Editor de texto plano con contador de caracteres estricto y selector de fechas (Calendario de ediciones).
  5. **Checkout:** Integración con Pasarela de Pagos (MercadoPago/ Stripe).

IV. Panel de Mostrador (Oficina/Sucursal)

Diseñado para velocidad y uso con teclado (Hotkeys). * **Carga Rápida:** Formulario optimizado para empleados que transcriben avisos dictados por clientes. * **Caja Diaria:** Gestión de pagos en Efectivo. Apertura y Cierre de

caja. * **Impresión de Ticket:** Comprobante de pago y constancia de publicación para el cliente. * **Gestión de Clientes Frecuentes:** Búsqueda rápida de anunciantes recurrentes (Inmobiliarias, Gestores).

---

4. Diseño de Base de Datos (SQL Server) - Conceptos Clave

Para lograr la flexibilidad que pides, la base de datos no será estática. Usaremos un modelo de **Entidad-Atributo-Valor (EAV)** híbrido o tablas relacionales flexibles.

1. **Tabla `Categories` (Recursiva):**
   - `Id`, `ParentId` (permite n-niveles de subrubros), `Name`, `Slug`, `Active`.
2. **Tabla `Operations`:**
   - `Id`, `Name` (Venta, Alquiler).
3. **Tabla `CategoryOperations` (Intermedia):**
   - Define qué operaciones son válidas para qué categoría.
4. **Tabla `Ads` (Avisos):**
   - Datos fijos: `Title`, `Price`, `Category_Id`, `Operation_Id`, `UserId`.
   - Datos papel: `PrintText`, `PrintStartDate`, `PrintDuration`.
5. **Tabla `AdAttributes` (La flexibilidad):**
   - Permite guardar datos específicos sin cambiar la base de datos.
   - `AdId`, `AttributeKey` (ej: “Kilometraje”), `AttributeValue` (ej: “50000”).

---

5. Funcionalidades Faltantes a Desarrollar (Gap Analysis)

Respecto al prototipo anterior, estas son las funciones nuevas que se desarrollarán desde cero en C#:

1. **Lógica de Atributos Dinámicos:** Que al crear un rubro “Autos”, el admin pueda definir que ese rubro lleva el campo “Kilometraje”, y que el frontend renderice ese input automáticamente.
2. **Sistema de Cola de Impresión:** Algoritmo que calcule exactamente qué avisos entran en la edición de mañana (considerando feriados o días sin diario).
3. **Sistema de Renovación Automática:** Emails automáticos a usuarios cuando su aviso está por vencer.
4. **Micro-ORM Dapper:** Escritura de Stored Procedures en SQL Server para búsquedas complejas (ej: “Autos entre $5M y $10M en La Plata con menos de 50k km”).

---

6. Plan de Trabajo (Roadmap)

Fase 1: Cimientos (Backend & DB)

- Diseño del Esquema SQL Server.
- Configuración de la Solución .NET 8 (Clean Architecture).
- Implementación de Autenticación (JWT) y Roles.
- CRUD de Taxonomía (API para crear/editar rubros y operaciones).

Fase 2: El Panel de Administración (React)

- Desarrollo del UI para gestionar el árbol de categorías.
- Desarrollo de la matriz de Operaciones/Rubros.
- Gestión de Usuarios internos.

Fase 3: Publicación y Web Pública

- Wizard de publicación con validación de reglas de negocio.
- Integración SDK MercadoPago (Backend C#).
- Home Page y Listados con filtros generados dinámicamente desde SQL.

Fase 4: Módulo de Papel & Mostrador

- Desarrollo del Panel de Mostrador (Caja y Carga Rápida).
- Desarrollo del algoritmo de exportación a TXT/XML para diagrama.

Fase 5: QA y Despliegue

- Pruebas de carga (Stress testing) con Dapper.
- Configuración de CI/CD (Dockerización de la API .NET y Nginx para React).
- Deploy en Servidor VPS/Cloud.

---

7. ¿Por qué esta tecnología?

1. **SQL Server + Dapper:** Es el estándar de oro para sistemas donde la integridad de los datos (pagos, fechas de publicación) es crítica. Dapper ofrece un rendimiento superior a cualquier ORM tradicional, necesario si el diario tiene miles de avisos históricos.
2. **C# .NET 10:** Tipado estático y compilado. Reduce errores en tiempo de ejecución drásticamente comparado con Node.js. Es mucho más fácil de mantener a largo plazo en sistemas empresariales complejos.
3. **React + TypeScript:** Garantiza que el frontend sea modular. Al usar TypeScript en ambos extremos (aunque el backend sea C#, los DTOs se pueden sincronizar), se minimizan los bugs.