B.A.I. SYSTEMS: CONTEXTO MAESTRO V2.0

Tema: Arquitectura Multi-Tenant, Nuevas Funcionalidades y Protocolo Skeleton. **Versión del Sistema:** 2.0 (Post-Refactorización "Modular Mind").

1. ARQUITECTURA NUCLEAR: EL PROTOCOLO "SKELETON"

B.A.I. ha evolucionado de una colección de páginas sueltas a una plataforma **Monorepo Unificado** bajo el principio de "*Radical Simplicity*".

- **Filosofía:** "Escribe una vez, despliega verticalmente". Un solo código base sirve para múltiples productos verticales (*Cannabiapp, Restaurantiapp*, etc.) mediante inyección de configuración.

- **Estructura de Rutas:**
  - **`/dashboard`**: Centro de mando unificado que dirige a los 3 servicios principales.
  - **`/software`**: Galería visual ("App Store") de soluciones verticales.
  - **`/data-mining`**: Dashboard de inteligencia de mercado con simulación de IA.
  - **`/demos/[appId]`**: Ruta dinámica que simula la experiencia de cualquier app del catálogo cambiando su identidad y contenido según la URL.

2. EL CEREBRO MULTI-TENANT (CHATBOTaaS)

La joya de la corona es la nueva arquitectura del Chatbot, diseñada para funcionar como **SaaS (Software as a Service)**.

2.1. Estrategia: "Cerebro Central, Ventana Remota"

En lugar de instalar un backend para cada cliente, B.A.I. utiliza un único servidor central (`api.baibussines.com`) que atiende a múltiples clientes externos simultáneamente.

- **El Cerebro (Backend):** Un motor en Python (FastAPI + Gemini 2.5) que gestiona la lógica y la personalidad.
- **La Ventana (Widget):** Un script ligero (`bai-widget.js`) que los clientes pegan en su propia web. Este script abre un chat flotante conectado a nuestro servidor.

2.2. Lógica de "Personalidad Dinámica"

El comportamiento del bot cambia automáticamente según quién lo llame, utilizando un `client_id`.

- **Detección:** El backend lee el `client_id` del widget.
    - Si es `bai_internal` → Actúa como "El Sabio Amigable" (Soporte B.A.I.).
    - Si es `inmo-cliente-001` → Actúa como "Agente Inmobiliario Vendedor".
- **Inyección de Datos (RAG Ligero):** El sistema carga un archivo JSON específico del cliente (ej: `inventories/inmo-001.json`) y se lo inyecta al prompt de Gemini. Esto permite que el bot conozca y venda el inventario específico de ese cliente sin reentrenar el modelo.

2.3. Gestión de Memoria

El sistema ahora soporta historial de conversación en el widget externo, permitiendo al bot recordar datos (presupuesto, zona) y realizar preguntas deductivas sin repetirse.

3. NUEVAS CARACTERÍSTICAS DE LA WEB

3.1. Software Studio (La Galería)

Una interfaz visual tipo "App Store" donde los clientes exploran las soluciones verticales.

- **Tarjetas Inteligentes:** Muestran características, sector y un botón de "Ver Demo".
- **Objeto Legendario:** Se ha añadido la tarjeta **"B.A.I. Neural Core"**, destacada en dorado con efectos de brillo, representando el servicio de IA pura.
- **Caso de Éxito:** La tarjeta de "RealEstate AI" abre directamente una web externa (`test-inmo.html`) para demostrar la integración real del widget.

3.2. Dashboard de Data Mining (La "Ilusión de Magia")

Una experiencia diseñada para vender el plan Enterprise mediante **Product-Led Growth**.

- **Flujo de Usuario:**
    1. **Estado Bloqueado:** El usuario ve un dashboard borroso con un candado ("Inteligencia Protegida").

2. **Entrevista:** Un chatbot realiza 4 preguntas clave (Idea, Sector, Edad, Redes) para personalizar el reporte.

3. **Efecto Matrix:** Una terminal falsa (`ProcessingTerminal`) muestra logs de "hackeo" y búsqueda para generar valor percibido.

4. **Revelación:** Se muestra el dashboard completo con gráficos (Recharts) generados a medida de las respuestas del usuario (Radar Charts, Demografía, Sentimiento).

- **Gancho de Venta:** Un botón destacado "Desbloquear Estrategia para llegar al 100/100".

3.3. Infraestructura de Enrutamiento

Se ha configurado **Caddy** con una lógica de "Doble Carril":

- Tráfico a `baibussines.com` → Frontend (Next.js).
- Tráfico a `baibussines.com/api/*` → Backend (Python), permitiendo que los widgets externos se conecten a la API de forma segura.

4. ESTRUCTURA DE CÓDIGO (BACKEND MODULAR)

El backend ha sido refactorizado bajo la arquitectura **"The Modular Mind"**:

- `app/services/brain/`: Carpeta que contiene la lógica modular.
  - `core.py`: Motor neuronal puro (conexión con Gemini).
  - `prompts.py`: Gestor de personalidades y carga de JSONs.
  - `tools.py`: Herramientas externas (Brave Search, n8n).

    - `memory.py`: Gestión de historial de chat.

- `bai_brain.py`: Actúa solo como orquestador, coordinando los módulos sin contener lógica de negocio.