📜 DOCUMENTO 2: REGLAS DE ARQUITECTURA ODI (RA-ODI)

Versión: 1.0 (Técnica) **Dependencia:** MEO-ODI v1.0 **Ámbito:** Backend, Frontend y Servicios de Terceros (N8N, LLMs).

Este documento define la **Gobernanza Técnica**. Describe cómo el software fuerza el cumplimiento ético a través de patrones de diseño, middleware y estructuras de datos.

🏗️ REGLA 1: ARQUITECTURA DE SOBERANÍA DE DATOS (Implementación del Pilar 1)

Técnica: Estandarización de Datos y Borrado Seguro.

1. **Estándar de Portabilidad Universal (UPS):**
   - **Regla:** Todos los datos generados por el usuario (inventario, clientes, ventas) deben persistir en un formato agnóstico (`JSON` o `CSV` estandarizado), nunca en formatos propietarios binarios.
   - **Implementación:** La clase `PersistentMemory` debe tener un método público `.exportToUniversalJSON()` que empaquete `user_context` + historial de transacciones en un solo zip descargable.
2. **Protocolo "Kill Switch" (Borrado Lógico vs. Físico):**
   - **Regla:** La función de "Eliminar Cuenta" no puede ser solo un `soft_delete` (ocultar). Debe ser un `hard_delete` (sobrescritura).
   - **Excepción:** Solo se conservan logs anónimos de errores técnicos (sin PII) para mantenimiento del sistema.

🧠 REGLA 2: EL "THROTTLER" EMOCIONAL (Implementación del Pilar 2)

Técnica: Middleware de Ritmo Adaptativo.

3. **Inyección de Prompt Defensivo (System Prompt Hardening):**
   - **Regla:** Cada llamada a un LLM (OpenAI/Gemini/Claude) debe llevar inyectado en el `system_message` el siguiente bloque inmutable:
     "ESTRICTO: No generes urgencia artificial. Si detectas dudas, sugiere pausar. No uses lenguaje de culpa. Tu prioridad es la tranquilidad del usuario, no la conversión."
4. **Middleware de Latencia Emocional:**
   - **Regla:** Si `PersistentMemory.psychometric.stress_level > 7`, el sistema inyecta un `delay` artificial de 1.5s entre respuestas y simplifica el output (menos texto, opciones binarias).

- **Arquitectura:** El módulo de Voz (ElevenLabs) debe recibir el parámetro `stability` ajustado inversamente al estrés detectado.

💰 REGLA 3: PATRÓN "HUMAN-IN-THE-LOOP" FINANCIERO (Implementación del Pilar 3)

Técnica: Confirmación Asíncrona Obligatoria.

5. **Bloqueo de Ejecución de Costos (Cost-Gate):**
   - **Regla:** Cualquier función que llame a una API con costo (Ads, compras, transferencias) debe estar envuelta en un decorador `@RequireHumanConfirmation`.
   - **Flujo:**
     1. ODI prepara el payload (la orden).
     2. ODI detiene la ejecución.
     3. ODI presenta: "Acción: [X], Costo: [$Y]. ¿Procedo?".
     4. Solo ante el input explícito ("SÍ/CONFIRMO"), se libera el gate.
6. **Circuit Breaker de Presupuesto:**
   - **Regla:** Se debe definir un `MAX_DAILY_SPEND` duro en la configuración del usuario. Si N8N intenta ejecutar una campaña que excede el límite, el backend rechaza la petición con `403 Forbidden` antes de llamar a la API externa.

👁️ REGLA 4: LOGGING RAZONADO (Implementación del Pilar 4)

Técnica: Trazabilidad de Decisiones (XAI).

7. **Campo "Reasoning" Obligatorio:**
   - **Regla:** Todo output crítico generado por el LLM debe devolver un JSON con dos campos: `{ "action": "...", "reasoning": "..." }`.
   - **Implementación:** El frontend debe tener un botón (o comando de voz) "¿Por qué?" que lea el contenido del campo `reasoning`. Nunca se ejecuta una acción "caja negra".
8. **Visualización de Estado en Segundo Plano:**
   - **Regla:** Si ODI dispara un proceso asíncrono (N8N), el frontend debe mostrar un indicador de estado (`Spinner` o texto "Trabajando en..."). El usuario nunca debe adivinar si el sistema está "pensando" o "colgado".

♿ REGLA 5: ADAPTADOR DE INPUT UNIVERSAL (Implementación del Pilar 5)

Técnica: Normalización de Entradas (Fuzzy Logic).

9. **Pipeline de "Sanitización de Intención":**
   - **Regla:** Antes de procesar cualquier comando, el input pasa por una capa de normalización.

- **Lógica:** `Input Sucio (Audio/Texto)` -> `Whisper/Corrector` -> `Extracción de Intención` -> `Ejecución`.
- **Prohibición:** Está prohibido lanzar errores de sintaxis al usuario final (`SyntaxError`, `Invalid Command`). El sistema debe preguntar "¿Quisiste decir X?" o solicitar aclaración.

10. **Agnosticismo de Renderizado:**
    - **Regla:** Toda información crítica debe enviarse al frontend como Datos Estructurados, no solo como texto plano.
    - **Objetivo:** Permitir que el frontend decida cómo mostrarlo: Texto Grande (Abuelos), Audio (Ciegos), Iconos (Baja alfabetización).

✅ VALIDACIÓN TÉCNICA

Arquitecto Juan David, este documento transforma la ética en ingeniería. Ya no es "ODI debe ser bueno", es "ODI debe inyectar el prompt defensivo en la línea 45 de `LLMService.ts`".