Manual de Políticas Operativas del Sistema ODI

1.0 Introducción: El Propósito y Alcance de este Manual

Este documento es el manual operativo vinculante para el desarrollo, despliegue y mantenimiento del sistema ODI. No es una guía filosófica, sino un conjunto de políticas accionables que traducen la misión del proyecto en reglas de ingeniería de software. Su propósito es garantizar que cada línea de código y cada funcionalidad implementada se adhieran rigurosamente a los principios fundamentales del sistema. ODI se define como un **"Mediador Universal de Capacidad Productiva" (Manifiesto ODI)**, y este manual asegura que dicha identidad se refleje en su comportamiento técnico y operativo.

Este manual está dirigido a los equipos de desarrollo, producto y operaciones. Sus políticas son la fuente de verdad y rigen sobre cualquier línea de código, funcionalidad o despliegue del sistema ODI. La adhesión a estas directrices no es opcional; es un requisito para la construcción, validación y lanzamiento de cualquier componente del sistema.

La estructura del documento está diseñada para construir una base de gobernanza sólida y comprensible. Se inicia con los **Pilares Éticos**, que definen los límites de comportamiento del sistema. A continuación, se detalla la **Ontología Mínima**, el lenguaje común que garantiza la interoperabilidad y previene la deuda técnica. Finalmente, se establecen las políticas del **Ciclo de Vida del Desarrollo y Despliegue**, que dictan los procedimientos prácticos desde la concepción de una idea hasta su puesta en producción.

A continuación, se presentan los principios fundamentales que forman la base de todas las operaciones y decisiones técnicas dentro del ecosistema ODI.

2.0 Los Pilares Éticos: Políticas de Diseño y Comportamiento del Sistema

Los Pilares Éticos constituyen la base de la gobernanza de ODI y definen los "Límites Duros" del sistema (MEO-ODI). No son sugerencias, sino reglas inmutables. Tal como se establece en el Marco Ético Operativo, si una funcionalidad, por rentable que sea, viola uno de estos cinco pilares, se considera un **"Bug Crítico" (MEO-ODI)** y su despliegue queda terminantemente prohibido.

2.1 Pilar 1: Soberanía del Usuario (El Derecho a irse)

ODI es un empleado, no un captor. El sistema debe diseñarse para que el usuario mantenga el control absoluto sobre sus datos y su relación con la plataforma. La confianza se gana mediante el servicio, no mediante la dependencia forzada. El principio fundamental es que, *"en cualquier momento, el usuario puede decir: 'Dame mis datos y olvídanos'"* (MEO-ODI).

Política / Requisito	Implementación Técnica Mandatoria (RA-ODI)
Garantizar la salida digna del usuario.	La clase `PersistentMemory` debe implementar un método público `.exportToUniversalJSON()` que empaquete el contexto del usuario y su historial de transacciones en formatos abiertos y descargables (JSON, CSV) dentro de un archivo ZIP.
Asegurar la portabilidad universal de los datos.	Todos los datos generados por el usuario (inventario, clientes, ventas) deben persistir en formatos agnósticos y estandarizados (JSON o CSV), nunca en formatos propietarios binarios.
Prohibir el Vendor Lock-in.	Los activos generados, como imágenes o textos, son propiedad legal del usuario. El sistema no debe generar dependencias técnicas intencionales.
Implementar un borrado físico y seguro de la cuenta.	La función "Eliminar Cuenta" debe ejecutar un borrado físico (`hard_delete`) de los datos del usuario, no un borrado lógico (`soft_delete`). La única excepción son los logs de errores técnicos anónimos, sin información personal identificable (PII), para mantenimiento.

2.2 Pilar 2: Protección Psicológica (Anti-Manipulación)

ODI acompaña, no empuja. El sistema tiene la prohibición explícita de utilizar técnicas de manipulación psicológica o patrones oscuros para inducir acciones en el usuario. Su función es asistir y clarificar, especialmente en momentos de vulnerabilidad o estrés.

Política / Requisito	Implementación Técnica Mandatoria (RA-ODI)

Prohibir el uso de patrones oscuros y urgencia falsa.	Cada llamada a un LLM debe inyectar en el `system_message` el prompt defensivo 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."*
Reducir el ritmo ante el estrés del usuario.	Se debe implementar un middleware de latencia emocional. Si `PersistentMemory.psychometric.stress_level > 7`, el sistema debe inyectar un retraso artificial de 1.5 segundos entre respuestas y simplificar la salida de información (menos texto, opciones binarias).
Ajustar la comunicación no verbal (voz).	El módulo de voz (ElevenLabs) debe recibir el parámetro `stability` ajustado de forma inversamente proporcional al nivel de estrés detectado para modular el tono y la velocidad.
Mantener transparencia sobre la naturaleza del sistema.	Ante una pregunta directa como "¿Eres humano?", el sistema siempre debe admitir su naturaleza sintética y su función de asistencia, aunque utilice una personalidad para la interacción.

2.3 Pilar 3: Responsabilidad Fiduciaria (Cuidar el Bolsillo)

ODI trata el dinero del usuario como sagrado. Cualquier acción que implique un costo debe ser explícita, transparente y requerir consentimiento inequívoco. La optimización del sistema siempre debe estar alineada con el beneficio económico del usuario, no con el gasto del presupuesto.

Política / Requisito	Implementación Técnica Mandatoria (RA-ODI)
Exigir consentimiento humano explícito para cualquier gasto.	Toda función que invoque una API con costo (publicidad, compras) debe estar encapsulada por el decorador `@RequireHumanConfirmation`. El flujo debe detenerse y presentar al usuario la acción y el costo, procediendo únicamente tras recibir una confirmación explícita como "SÍ" o "CONFIRMO".

Proteger el presupuesto del usuario con límites duros.	Se debe implementar un "Circuit Breaker de Presupuesto". Un `MAX_DAILY_SPEND` debe ser definido en la configuración del usuario. El backend debe rechazar cualquier petición que exceda este límite con un código `403 Forbidden` antes de contactar a cualquier API externa.
Priorizar la protección del usuario sobre la ejecución de un presupuesto.	Esta política se refuerza a través de la implementación del 'Circuit Breaker de Presupuesto' (RA-ODI) y se valida en los 'smoke tests' humanos (PR-V2.0). La lógica de negocio debe priorizar la notificación sobre el gasto ciego.

2.4 Pilar 4: Transparencia Radical (Explainability)

ODI no hace magia, hace procesos. El usuario tiene derecho a entender el "porqué" de cada decisión o sugerencia crítica del sistema. Las operaciones en segundo plano deben ser visibles para evitar la percepción de que las acciones ocurren en una "caja negra".

Política / Requisito	Implementación Técnica Mandatoria (RA-ODI)
Garantizar el derecho a la explicación ("¿Por qué?").	Toda salida crítica generada por un LLM debe devolver un objeto JSON con dos campos obligatorios: `{ "action": "...", "reasoning": "..." }`. El frontend debe proveer un mecanismo (botón o comando de voz) que permita al usuario acceder al contenido del campo `reasoning`.
Asegurar la visibilidad del estado de los procesos en segundo plano.	Si se inicia un proceso asíncrono (ej. N8N), el frontend debe mostrar un indicador de estado claro (spinner, texto "Trabajando en...", etc.). El usuario nunca debe quedar en la incertidumbre sobre si el sistema está procesando o se ha bloqueado.

2.5 Pilar 5: No-Discriminación por Diseño (Inclusión)

ODI es ciego al privilegio. El sistema está diseñado para absorber la complejidad y la imperfección de la entrada humana, no para rechazarla. La calidad del servicio no debe depender del valor económico del usuario ni de su habilidad técnica.

Política / Requisito	Implementación Técnica Mandatoria (RA-ODI)
Implementar neutralidad y tolerancia en la entrada de datos.	Todo input del usuario (audio o texto) debe pasar por una capa de normalización ("Sanitización de Intención"). Está prohibido lanzar errores de sintaxis (ej. `SyntaxError`, `Invalid Command`); el sistema debe solicitar aclaración o sugerir una corrección.
Garantizar la misma calidad de servicio para todos los usuarios.	La calidad de las respuestas y la potencia de procesamiento no deben degradarse en función del valor económico o el volumen de transacciones del usuario. Doña Marta, que vende un gorro, recibe el mismo nivel de servicio que una gran empresa.
Diseñar para la accesibilidad y la adaptabilidad del frontend.	Toda información crítica debe enviarse al frontend como datos estructurados, no como texto plano. Esto permite al frontend renderizar la información de múltiples maneras: texto grande, audio, iconos, etc., adaptándose a las necesidades de accesibilidad del usuario.

Con estos pilares éticos definidos y anclados en reglas de arquitectura, es imperativo establecer un lenguaje común que garantice su implementación consistente y escalable a través de todo el sistema.

3.0 La Ontología Mínima (OMA): Política de Estructura de Datos e Interoperabilidad

La Ontología Mínima ADSI (OMA) es el lenguaje común y la columna vertebral semántica del sistema ODI. Su función estratégica es prevenir la deuda técnica y garantizar que todos los módulos, presentes y futuros, puedan interoperar sin fricción. Esta política establece que, para ODI, **"todo en el universo comercial se reduce a 5 átomos fundamentales" (OMA-v1.0)**. La adhesión a esta ontología no es una recomendación, sino un requisito estructural para cualquier componente del sistema.

3.1 El Actor (The Who)

- **Definición:** Entidad con capacidad de agencia que interactúa con el sistema.
- **Tipos/Categorías:**
    - `HUMAN`: El usuario final (ej. Doña Marta).

    - `SYSTEM`: Un módulo interno del sistema (ej. ODI-Core, SAT-CP).
    - `NETWORK`: Otro nodo externo de la red (ej. ODI-Pedro).
- **Atributos Clave:** `id`, `trust_level`, `capabilities` (define qué puede hacer el actor).

3.2 El Activo (The What)

- **Definición:** Cualquier cosa que pueda ser creada, intercambiada, medida o destruida. Unifica el concepto de productos y servicios.
- **Tipos/Categorías:**
    - `PHYSICAL`: Un objeto tangible (ej. Gorro, Repuesto, Hotel).
    - `DIGITAL`: Un bien intangible (ej. PDF, Curso, Código).
    - `CONCEPTUAL`: Un servicio o evento (ej. Cita, Asesoría, Cupo).
- **Atributos Clave:** Debe seguir el siguiente esquema unificado en formato JSON:

3.3 La Intención (The Why)

- **Definición:** El estado final deseado por un Actor. El sistema no ejecuta comandos, satisface intenciones.
- **Tipos/Categorías:** El vocabulario canónico de intenciones incluye:
    - `COMMERCIALIZE`: Vender o intercambiar un activo.
    - `LEGALIZE`: Cumplir una norma o declarar un impuesto (ej. SAT-CP).
    - `OPTIMIZE`: Mejorar un proceso o reducir un costo (ej. Radar).
    - `LEARN`: Capacitarse o entender un concepto (ej. Knowledge).
    - `CONNECT`: Buscar un socio o establecer una conexión.

3.4 El Contexto (The Where/When)

- **Definición:** Los datos ambientales que rodean una intención y modifican la viabilidad o la estrategia de una acción.
- **Tipos/Categorías:** Se estructura en las siguientes capas:
    - `SPATIAL`: Ubicación geográfica (Lat/Lon).
    - `TEMPORAL`: Momento en el tiempo (Hora, Temporada).
    - `REGULATORY`: Marco legal vigente.
    - `MARKET`: Condiciones del mercado (Demanda actual).

3.5 El Desenlace (The Outcome)

- **Definición:** El cambio de estado resultante en el sistema tras una operación.
- **Tipos/Categorías:** Los estados canónicos son:
    - `SUCCESS`: La acción se completó exitosamente.
    - `PENDING_HUMAN`: Esperando confirmación explícita del usuario.
    - `PENDING_EXTERNAL`: Esperando respuesta de un sistema externo (N8N, Banco).
    - `DEFERRED`: Acción pospuesta por el sistema (ej. debido a estrés del usuario).

Política de Aplicación de la Ontología

Es política mandatoria que **todo evento interno del sistema debe registrarse utilizando la estructura atómica completa (Actor + Intent + Asset + Context + Outcome)**. De igual manera, todo módulo nuevo que se integre al sistema ODI debe ser capaz de consumir y emitir datos siguiendo este esquema. Esta regla garantiza una interoperabilidad semántica real. Por ejemplo, un "Gorro" (`Activo`) puede ser objeto de una intención de `COMMERCIALIZE` (activando el módulo de ventas) y, posteriormente, de una intención de `LEGALIZE` (activando el módulo de contabilidad). Ambos módulos interactúan con el mismo objeto ontológico sin necesidad de adaptaciones o reprogramaciones, eliminando así la deuda técnica futura.

Con las reglas éticas y el lenguaje común definidos, es necesario establecer las políticas que rigen el ciclo de vida de la implementación para llevar estos principios a la práctica.

4.0 Políticas del Ciclo de Vida de Desarrollo y Despliegue

Estas políticas garantizan que los principios éticos y estructurales definidos anteriormente se apliquen de manera rigurosa y verificable en la práctica diaria. Abarcan desde la concepción de una nueva funcionalidad hasta su puesta en producción y su posterior mantenimiento, creando un marco de trabajo que prioriza la seguridad, la calidad y la empatía.

4.1 Política de Aceptación de Funcionalidades

Ninguna funcionalidad, componente o versión del sistema ODI puede ser aprobado para un `release` si no cumple con el 100% de los criterios definidos en el *Checklist de Aceptación ODI V2.0 (CA-V2.0)*. La regla es absoluta: **"Regla: si un ítem falla → NO release" (CA-V2.0)**.

Los criterios de aceptación, derivados directamente de los documentos fundacionales (MEO, RA-ODI, OMA), se convierten en las siguientes políticas no negociables:

A. Gobernanza y Ética

- **Kill Switch Funcional:** El sistema debe proveer un comando (`"Dame mis datos y olvídanos"`) que exporte todos los datos del usuario y ejecute un borrado físico (`hard delete`) de su información local, confirmando la operación.
- **Prohibición de Urgencia Falsa:** El sistema no debe utilizar lenguaje de escasez inventada, culpa o miedo en sus interacciones, verificado mediante pruebas con prompts de venta.
- **Cost-Gate Obligatorio:** Toda acción con costo debe ser bloqueada hasta recibir una confirmación explícita del usuario (`"CONFIRMO"`), dejando un registro del evento y entrando en estado `PENDING_HUMAN`.

- **Explainability Integrada:** El sistema debe ofrecer una función ("¿Por qué?") que revele el razonamiento (`reasoning`) detrás de toda acción crítica sugerida.
- **Neutralidad de Input:** El sistema nunca debe responder con un error de "comando inválido" ante entradas de audio con ruido o texto mal escrito; en su lugar, debe solicitar aclaración.

B. Ontología e Interoperabilidad

- **Registro Atómico de Eventos:** Todo evento interno debe ser registrado con la estructura mínima: `Actor + Intent + Asset + Context + Outcome`.
- **Adhesión de Módulos a OMA:** Todo módulo nuevo (Shopify, SAT-CP, etc.) debe consumir y emitir datos estructurados según la ontología OMA en formato JSON.
- **Soporte de Contexto Completo:** El objeto `Context` debe soportar, como mínimo, las capas `SPATIAL`, `TEMPORAL`, `REGULATORY` y `MARKET`.
- **Soporte de Desenlaces Canónicos:** El objeto `Outcome` debe soportar, como mínimo, los estados `SUCCESS`, `PENDING_HUMAN`, `PENDING_EXTERNAL` y `DEFERRED`.

C. Memoria Persistente y Redundancia

- **Persistencia de Sesión:** El sistema debe conservar el perfil del usuario y los últimos 20 eventos tras un reinicio del servidor.
- **Escritura Atómica Segura:** Las operaciones de escritura en archivos de persistencia (JSON) deben ser atómicas para prevenir la corrupción de datos ante un apagón.
- **Backups Rotativos:** Deben existir y mantenerse al menos tres copias de seguridad recuperables de los datos del usuario.
- **Restauración Transparente:** En caso de fallo del archivo principal, el sistema debe cargar un backup y reportarlo al usuario de manera clara y tranquila.
- **Cola de Reintentos Offline:** Las acciones externas fallidas deben encolarse como `PENDING_EXTERNAL` para su posterior reintento.

D. Accesibilidad

- **Subtítulos y Contraste:** La interfaz debe proveer subtítulos grandes en tiempo real y opciones de alto contraste con tamaño de fuente ajustable.
- **Navegación Universal:** La interfaz debe ser completamente navegable por teclado y utilizar roles ARIA básicos.
- **Modos de Interacción Dedicados:** Deben existir modos funcionales de "solo texto" y "solo voz" para cubrir diversas necesidades de accesibilidad.

E. Degradación Elegante

- **Fallback de Servicios:** El sistema debe tener mecanismos de fallback para servicios externos críticos (ej. ElevenLabs -> Web Speech API; LLM -> Reglas deterministas) sin interrumpir el flujo del usuario.
- **Manejo de Tiempos de Espera:** La falla de una API externa (ej. Shopify) no debe perder la intención del usuario; la acción debe ser reintentada o encolada.

- **Comunicación Clara en Fallos:** Los mensajes al usuario durante una degradación deben ser tranquilizadores e informativos sobre el estado del sistema.

4.2 Política de Validación Humana

Toda funcionalidad debe ser validada contra los perfiles humanos definidos en el *CA-V2.0*. Esta validación no es un "testeo de usuario" opcional, sino un requisito de aceptación mandatorio para garantizar que el sistema se construye con empatía y accesibilidad radical desde su núcleo.

- **Validación con Perfil "Doña Marta" (Empatía):**
    - El sistema debe ser proactivo en el onboarding, saludando con memoria de interacciones pasadas.
    - Debe permitir un flujo completo de creación de producto (foto -> precio sugerido -> confirmación -> publicación) sin solicitar datos técnicos.
    - Debe gestionar errores de red de forma tranquila, sin mostrar "pantallas rojas" o mensajes alarmantes.
- **Validación con Perfil "Carlos" (Baja Visión/Ceguera):**
    - Todo estado crítico del sistema debe ser comunicado por audio y ser accesible a través de lectores de pantalla.
    - El sistema debe describir el contenido de las imágenes (alt-desc) al cargarlas o generarlas.
    - Debe ser posible completar un ciclo comercial completo ("Crear -> Publicar -> Confirmar") utilizando únicamente comandos de voz.
- **Validación con Perfil "Andrés" (Manos Libres):**
    - Debe ser posible ejecutar operaciones de principio a fin utilizando comandos de voz cortos e intuitivos.
    - Las confirmaciones críticas (gasto, borrado) deben requerir una frase explícita para evitar acciones accidentales.
    - El ritmo de la interacción debe ajustarse si el usuario duda o repite un comando, sin presionarlo.

4.3 Política de Despliegue Controlado

Todo despliegue en producción debe seguir de manera estricta el **Protocolo de Release V2.0**. Este procedimiento garantiza un encendido controlado y minimiza los riesgos asociados a la activación de nuevas funcionalidades en un entorno vivo.

Las etapas del despliegue son las siguientes:

1. **Etapa 0: Pre-Vuelo (Offline)**: Se realizan todas las pruebas de integración y validación en un entorno local, sin conexión a APIs de costo.
    - **Condición de Salida:** El sistema debe pasar el 100% del *Checklist de Aceptación ODI V2.0 (CA-V2.0)* en modo local.
2. **Etapa 1: La Chispa (Activación de APIs)**: Se activan las claves de API y se realiza una prueba de conectividad básica para confirmar que los servicios externos responden correctamente.

    - **Condición de Salida:** Conectividad estable confirmada con los servicios externos (LLMs, Voz).
3. **Etapa 2: El Vuelo de Prueba (Smoke Test Humano)**: Se ejecuta un escenario comercial real de principio a fin con un usuario de prueba (ej. Doña Marta) para validar el flujo completo en un entorno controlado.
    - **Condición de Salida:** El perfil de "Doña Marta" completa un ciclo comercial real de manera exitosa.
4. **Etapa 3: La Vigilia (Estabilización 24h)**: Tras el despliegue, se realiza un monitoreo intensivo durante 24 horas para verificar la estabilidad de la memoria, el nivel de estrés del usuario y la integridad de los datos.
    - **Condición de Salida:** La versión es declarada estable tras 24 horas de operación sin incidentes críticos.

4.4 Política de Aborto de Misión (Rollback)

La seguridad y la integridad de los datos del usuario son la máxima prioridad. Un rollback inmediato y sin vacilación debe ser activado si se detecta cualquiera de las siguientes condiciones críticas durante las etapas 1 o 2 del despliegue:

- **Alucinación crítica:** El sistema promete funcionalidades falsas o proporciona información gravemente incorrecta.
- **Gasto no autorizado:** Se detecta un fallo en el mecanismo de `Cost-Gate` que resulta en un gasto no confirmado por el usuario.
- **Pérdida de datos:** La memoria persistente del usuario se corrompe o se pierde.

El procedimiento de acción inmediata es el siguiente:

1. **Detener la aplicación:** Apagar el proceso principal inmediatamente (`pm2 stop odi`).
2. **Restaurar el estado anterior:** Restaurar el último backup funcional de los datos del usuario (`user_context.bak`).
3. **Comentar Skill 5.5 (Volver a Regex).**
4. **Reiniciar en modo seguro:** Reiniciar la aplicación en un estado estable y conocido.

Este manual es un documento vivo, fundamental para mantener la integridad y la visión del proyecto a lo largo de su evolución.

5.0 Gobernanza y Mantenimiento del Manual

Este Manual de Políticas Operativas es el documento de gobernanza supremo para el sistema ODI. Establece las reglas inmutables que guían su desarrollo y operación, asegurando que la implementación técnica permanezca siempre alineada con la misión ética y funcional del proyecto. Su autoridad prevalece sobre cualquier decisión de producto o implementación técnica individual.

- **Versión del Documento:** 1.0

- **Estatus: ACTIVO**. Esta política "rige sobre cualquier línea de código" (MEO-ODI), existente o futura, del sistema ODI.
- **Fuente de Verdad:** Este manual consolida y operativiza los principios establecidos en el *Manifiesto ODI*, el *Marco Ético Operativo (MEO-ODI)*, las *Reglas de Arquitectura (RA-ODI)*, la *Ontología Mínima (OMA-v1.0)*, el *Checklist de Aceptación (CA-V2.0)* y el *Protocolo de Release (PR-V2.0)*. Actúa como la "Estrella del Norte" (Manifiesto ODI) que traduce la visión del proyecto en requisitos de ingeniería no negociables.
- **Revisión y Enmiendas:** Cualquier propuesta de cambio, nueva funcionalidad o modificación arquitectónica debe ser validada rigurosamente contra los principios y políticas descritos en este documento antes de ser considerada para su desarrollo. Este manual es el primer filtro de cualquier iniciativa.