NexOS — Gameplan de Agentes y Subagentes (1 número hoy, base preparada para multi-número)

> Objetivo: Operar TODO el ecosistema NexOS con **un solo número de WhatsApp** (fase inicial), manteniendo el diseño canónico (Zero-UI, event-driven, SSOT, idempotencia, flags/estados) y dejando **preparada la base** para, en el futuro, añadir **números adicionales** por operatividad sin rediseñar el sistema.

---

0) Principios de diseño (no negociables)

1. **WhatsApp es canal, no memoria**: ninguna decisión crítica se toma "por lo que diga el chat".
2. **SSOT central**: Airtable/DB definida como verdad.
3. **Event-driven**: todo se dispara por eventos reales (mensaje, audio, imagen, webhook, fecha).
4. **Idempotencia**: un evento repetido nunca duplica acciones.
5. **Flags y estados**: todo lo relevante queda marcado (qué ya se hizo).
6. **Fix mínimo**: la futura expansión a multi-número será un cambio de canal, no de lógica.

---

1) Arquitectura lógica (un cerebro, muchos planetas)

1.1 Orquestador (Cerebro NexOS)

Responsable de: - Identificar **quién** escribe (contacto/cliente) y con qué **contexto**. - Detectar **intención** y elegir **planeta** (producto/módulo) + **fase**. - Llamar al **subagente** correcto. - Aplicar **políticas globales** (tono, límites, consentimientos, plantillas, escalado humano, horarios, etc.). - Registrar en SSOT: `estado`, `flags`, `event_id`, `acciones`.

1.2 Subagentes (Planetas)

Cada planeta es un agente especializado con: - **KB** y reglas del planeta (RentOS, IngresOS, Control Taxi PRO, etc.). - "Herramientas" (funciones) que ejecutan acciones: crear/actualizar registros, generar documentos, enviar mensajes, etc. - Estados propios (operativos/financieros/soporte), pero siempre escritos en SSOT.

> Importante: subagentes **no envían WhatsApp directamente** si no es necesario; pasan por el Orquestador para mantener consistencia y control.

---

2) Gameplan por fases (1 número hoy)

Fase 1 — Fundaciones (imprescindible)

Meta: Tener routing robusto + SSOT + idempotencia, funcionando con un único número.

Entregables: 1. **Modelo de conversación** en SSOT (tablas y campos mínimos). 2. **Registro de eventos** (message_id, media_id, timestamp, hash) para idempotencia. 3. **Router** (clasificación de planeta y fase). 4. **Contrato de respuesta** (formato estándar de salida para subagentes).

Fase 2 — 1er planeta en producción (p. ej. RentOS)

Meta: Un planeta (RentOS) con subagente ya "enchufado" al cerebro.

Entregables: - Subagente RentOS v1: intents principales, KB operativa, acciones seguras. - Flags y estados RentOS.

Fase 3 — Añadir planetas (IngresOS, etc.)

Meta: Añadir planetas sin tocar la lógica global.

Entregables: - Subagente IngresOS v1: intents financieros, estructura de estados. - "Plantillas de mensajes" y "eventos programados" del planeta.

---

3) SSOT: esquema mínimo para gobernar agentes

> Nota: aquí se define un esquema **genérico** de NexOS (no inventa nombres de tu base RentOS canónica). Es el "backbone" del cerebro.

3.1 Tabla: `Contacts`

- `contact_id` (PK)
- `phone_e164`
- `name`
- `org_id` (si aplica)
- `preferred_language`
- `consent_marketing` (bool)
- `created_at`, `updated_at`

3.2 Tabla: `Conversations`

- `conversation_id` (PK)
- `contact_id` (FK)
- `channel_id` (FK → Channels)
- `active_planet` (enum: RentOS, IngresOS, …)
- `phase` (enum: lead, onboarding, ops, support)
- `status` (enum: open, waiting_user, resolved, escalated)
- `last_event_id`
- `last_user_message_at`
- `locks` (json: p. ej. "no comercial en ops")

3.3 Tabla: `Events` (idempotencia)

- `event_id` (PK; recomendado: `${provider}:${message_id}`)
- `conversation_id`

- `provider` (WhatsApp)
- `message_id`
- `type` (text|audio|image|document|interactive)
- `payload_hash` (hash del contenido relevante)
- `received_at`
- `processed_at`
- `processing_status` (new|processed|ignored|failed)
- `error` (si aplica)

3.4 Tabla: `Actions` (trazabilidad)

- `action_id` (PK)
- `event_id` (FK)
- `planet`
- `action_type` (send_message|create_record|update_record|generate_pdf|...)
- `idempotency_key` (única por acción)
- `status` (queued|done|failed)
- `result_ref` (link/ID externo)
- `created_at`

3.5 Tabla: `Channels` (preparada para multi-número)

- `channel_id` (PK)
- `provider` (WhatsApp Cloud API)
- `phone_number_id` (Meta)
- `display_name`
- `mode` (brain|ops)
- `default_planet` (nullable)
- `active` (bool)

---

4) Router (cómo decide el Orquestador)

4.1 Inputs del router

- Texto del usuario (o transcripción si audio)
- Señales del historial (active_planet, phase)
- Canal (channel_id)
- Palabras clave / intents
- Reglas de “bloqueo” (locks)

4.2 Output del router (contrato)

Un objeto estándar: - `planet` - `phase` - `intent` - `confidence` - `needs_handoff` (humano) - `safe_reply_allowed` (bool)

4.3 Regla de persistencia

Si `confidence` es alta o el usuario confirma un planeta, se fija: - `active_planet = X`

Y queda hasta que: - el usuario pida explícitamente cambiar, o - se detecte una intención dominante de otro planeta con alta confianza.

---

5) Contrato Orquestador ↔ Subagentes (imprescindible)

5.1 Entrada a subagente

- `conversation_context` (resumen + estado)
- `intent`
- `user_message`
- `attachments` (si hay)
- `constraints` (locks, policies)

5.2 Salida de subagente

- `assistant_message` (texto listo)
- `actions[]` (lista de acciones propuestas)
- `state_updates` (qué campos actualizar)
- `flags_to_set` (checkpoints)
- `requires_confirmation` (si hay acciones sensibles)

El Orquestador es quien **valida** y **ejecuta** acciones.

---

6) Idempotencia: cómo evitar duplicados

6.1 Evento entrante

- Si `Events.processing_status == processed` → **STOP**

6.2 Acción saliente

Cada acción debe llevar `idempotency_key` (único): - Ejemplo conceptual: `send_message:${conversation_id}:${template}:${yyyy-mm-dd}:${variant}`

Antes de ejecutar: - si existe `Actions.idempotency_key` en `done|queued` → **no repetir**

---

7) Operar con 1 número (hoy)

7.1 Un solo canal

- `Channels` tendrá 1 registro: `mode=brain`.
- Todo entra por ahí.
- Todo sale por ahí.

7.2 Separación “lógica” aunque sea el mismo número

Aunque físicamente sea un número, el sistema separa por: - `phase` (ops vs lead) - `locks` - `active_planet`

Esto previene que el agente “meta comercial” donde no toca.

---

8) Preparación para multi-número (mañana) — sin rediseñar

8.1 Qué cambia

- Se añade otro registro en `Channels` (p. ej. `mode=ops`, `default_planet=RentOS`).
- El inbound webhook ya trae `phone_number_id` → se mapea a `channel_id`.

8.2 Qué NO cambia

- Router, subagentes, SSOT, idempotencia, flags.
- El contrato Orquestador↔Subagentes.

8.3 Estrategia de migración visible al cliente

- Informar: “A partir de hoy, los avisos operativos llegan desde este número.”
- Mantener el número “brain” para soporte/solicitudes.

8.4 Reglas de envío (outbound) con múltiples canales

Cuando el Orquestador va a enviar un mensaje: 1. Si `phase=ops` y existe `channel.mode=ops` para ese planeta → enviar por OPS. 2. Si no, enviar por `brain`.

> Todo esto se controla por `channel_id` + reglas, no por duplicar workflows.

---

9) Workflows n8n (plantilla lógica)

WF-01 Inbound WhatsApp → Orquestador

1. Webhook (WhatsApp)
2. Normalización payload
3. Upsert Contact
4. Upsert Conversation (por contact+channel)
5. Insert Event (idempotencia)
6. Router → planeta/fase/intent
7. Llamada a subagente
8. Guardar `Actions` propuestas + `state_updates`
9. Ejecutar acciones (con idempotencia)
10. Marcar Event processed

WF-02 Scheduler (eventos de tiempo)

- Para recordatorios/avisos: disparo por fecha → genera un `Event` interno → Orquestador → acciones.

  Con esto, lo programado también es event-driven (evento de tiempo), con idempotencia.

---

10) Checklist de implementación (para guiarnos cuando lo hagamos)

Checklist A — SSOT listo

- [ ] Tablas: Contacts, Conversations, Events, Actions, Channels
- [ ] Índices/unique: phone_e164, event_id, idempotency_key
- [ ] Campos mínimos de estado (active_planet, phase)

Checklist B — Router estable

- [ ] Persistencia de planeta activo
- [ ] Locks por fase (ops no se mezcla con lead)
- [ ] Umbral de confidence

Checklist C — Subagentes por planeta

- [ ] RentOS v1 con intents base
- [ ] IngresOS v1 con intents base
- [ ] Contrato de salida (message + actions + flags)

Checklist D — Multi-número (futuro)

- [ ] Channels con 2+ números
- [ ] Envío outbound según mode
- [ ] Migración de comunicación al cliente

---

11) Decisiones tomadas (resumen)

- Hoy: **1 número** para todo.
- Diseño: **Orquestador NexOS** + **subagentes por planeta**.
- Base preparada: tabla `Channels` + routing por `channel_id`.
- Mañana: añadir números es **cambio de canal**, no rediseño.

---

12) Próximo paso cuando retomemos

1) Definir el **set exacto de planetas** a cubrir en el router (los que van en la landing). 2) Fijar el **diccionario de fases** (lead/onboarding/ops/support) y locks. 3) Implementar WF-01 con idempotencia completa.