

## ■# Auditoria de Funcionamiento Trading Agentic

Fecha: 2026-02-25

Repositorio auditado: D:\OneDrive\GitHub\traiding-agentic

### 1) Alcance y metodologia

Se audito:

- Codigo de ejecucion de trades (Next.js + Python backend)
- Calculo de PnL (realizado/no realizado), comisiones y portfolio
- Esquema y consistencia de base de datos (migraciones vs runtime)
- Estado de ejecucion real sobre DB (propuestas, posiciones, riesgo, reconciliacion)
- Salud de build/typecheck/tests

Comandos y verificaciones ejecutadas:

- `npm run typecheck` (fallo)
- `npm run build` (fallo)
- `npm run lint` (ok)
- `python -m pytest -q` en `backend/` (fallo)
- Consultas directas a Supabase para validar formulas y conteos operativos

### 2) Resultado ejecutivo

Estado general: **NO listo para operar en dinero real.**

Resumen corto:

- El calculo de PnL guardado en las pocas posiciones historicas cerradas hoy es consistente con su formula interna.
- Pero el sistema operativo esta degradado: en 24h hubo **\*\*7223 propuestas en error\*\*** y **\*\*0 ejecuciones exitosas\*\*** en ese periodo.
- La ganancia mostrada no esta "mal calculada" en esos pocos registros, pero **\*\*no es confiable como metrica de rendimiento real del bot\*\*** porque casi no hay trades exitosos recientes y hay fallas masivas de ejecucion.

### 3) Hallazgos prioritarios

### Critico 1 - Ejecucion operativa fallando de forma masiva

Evidencia en DB (medido en esta auditoria):

- Ultimas 24h: `error=7223`, `executed=0`, `rejected=1` en `trade\_proposals`
- Riesgo ultimas 24h: `execution\_error=7232`, `order\_executed=0`

- Error dominante: `400 Bad Request` contra proxy Binance

Impacto:

- El bot no esta transformando senales en transacciones exitosas.
- El historial de ganancia reciente no representa operativa real.

### Critico 2 - Build/typecheck rotos (inconsistencia de API quant)

- [app/api/quant/backtest/benchmark/route.ts](D:/OneDrive/GitHub/traiding-agentic/app/api/quant/backtest/benchmark/route.ts:4) importa `runBacktestBenchmark` que no existe en [lib/trading/python-backend.ts](D:/OneDrive/GitHub/traiding-agentic/lib/trading/python-backend.ts:92).
- [app/api/quant/backtest/presets/route.ts](D:/OneDrive/GitHub/traiding-agentic/app/api/quant/backtest/presets/route.ts:3) importa `getBacktestPresets` que no existe.

Impacto:

- `npm run typecheck` y `npm run build` fallan, bloqueando despliegues confiables.

### Critico 3 - Secreto hardcodeado en repositorio

- [docs/02-2026/generate\_maxi\_pdf.py](D:/OneDrive/GitHub/traiding-agentic/docs/02-2026/generate\_maxi\_pdf.py:14) contiene una API key hardcodeada.

Impacto:

- Riesgo alto de seguridad (exposicion de credenciales).

### Alto 1 - Bug de PnL al cerrar por cantidad mayor a la entrada

En Python executor:

- Calcula `realized\_pnl` antes de clamp de cantidad en [backend/app/services/executor.py](D:/OneDrive/GitHub/traiding-agentic/backend/app/services/executor.py:182).
- El clamp ocurre despues en [backend/app/services/executor.py](D:/OneDrive/GitHub/traiding-agentic/backend/app/services/executor.py:185).

Impacto:

- Si llega `exit\_qty > entry\_qty`, el PnL queda mal calculado.

### Alto 2 - Manejo parcial de posiciones inconsistente

- Cierre busca solo `status=open` en [backend/app/services/executor.py](D:/OneDrive/GitHub/traiding-agentic/backend/app/services/executor.py:172).
- Si queda `partially\_closed`, ya no se vuelve a encontrar esa posicion para siguiente cierre.
- Portfolio tambien solo recalcula `open` en [backend/app/services/portfolio.py](D:/OneDrive/GitHub/traiding-agentic/backend/app/services/portfolio.py:22).

Impacto:

- Riesgo de posiciones parcialmente cerradas "atascadas" fuera del flujo normal.

#### ### Alto 3 - Riesgo de drift schema/codigo en estados y eventos

Migracion de `trade_proposals` define estados:

- [supabase/migrations/20260216\_create\_trading\_tables.sql](D:/OneDrive/GitHub/traiding-agentic/supabase/migrations/20260216\_create\_trading\_tables.sql:27)

- Solo: `draft`, `validated`, `approved`, `rejected`, `executed`, `error`

Codigo usa estados adicionales:

- ``dead_letter`` en [backend/app/services/executor.py](D:/OneDrive/GitHub/traiding-agentic/backend/app/services/executor.py:103)
- ``cancelled`` en [backend/app/routers/dead\_letter.py](D:/OneDrive/GitHub/traiding-agentic/backend/app/routers/dead\_letter.py:61)

Impacto:

- Segun el esquema aplicado, puede romper updates/inserts o dejar rutas nunca utilizables.

#### ### Alto 4 - Riesgo de drift schema/codigo en `risk_events.event_type`

`20260216_fix_schema_issues.sql` agrega tipos como:

- ``execution_blocked``, ``proposal_cancelled``, ``position_opened``, ``risk_warning`` en [supabase/migrations/20260216\_fix\_schema\_issues.sql](D:/OneDrive/GitHub/traiding-agentic/supabase/migrations/20260216\_fix\_schema\_issues.sql:31)

Pero `20260217_quant_engine_tables.sql` vuelve a redefinir el check sin varios de esos tipos en [supabase/migrations/20260217\_quant\_engine\_tables.sql](D:/OneDrive/GitHub/traiding-agentic/supabase/migrations/20260217\_quant\_engine\_tables.sql:218).

Impacto:

- Inserciones de eventos pueden fallar segun orden/estado real de migraciones.

#### ### Alto 5 - Validacion de estrategias inconsistente

- Trading agent filtra ``validation_status=approved`` en [lib/agents/trading-agent.ts](D:/OneDrive/GitHub/traiding-agentic/lib/agents/trading-agent.ts:242).
- Migracion define valores permitidos ``pending``, ``validated``, ``rejected``, ``needs_review`` en [supabase/migrations/20260216\_fix\_schema\_issues.sql](D:/OneDrive/GitHub/traiding-agentic/supabase/migrations/20260216\_fix\_schema\_issues.sql:98).
- Otro endpoint usa ``validated`` en [app/api/pipeline/status/route.ts](D:/OneDrive/GitHub/traiding-agentic/app/api/pipeline/status/route.ts:53).

Impacto:

- Puede dejar al agente sin estrategias aunque existan en DB.

#### ### Alto 6 - Seguridad de backend deshabilitable por config vacia

- Si `BACKEND\_SECRET` no esta seteado, se salta auth global en [backend/app/main.py](D:/OneDrive/GitHub/traiding-agentic/backend/app/main.py:24).

Impacto:

- En configuracion incorrecta, endpoints sensibles quedan expuestos.

#### ### Medio 1 - Modelo de comisiones incompleto para PnL multi-asset

- Se resta comision como si siempre fuera misma unidad del PnL en [backend/app/services/executor.py](D:/OneDrive/GitHub/traiding-agentic/backend/app/services/executor.py:141) y [backend/app/services/portfolio.py](D:/OneDrive/GitHub/traiding-agentic/backend/app/services/portfolio.py:35).
- `commission\_asset` no siempre se propaga correctamente (historico con `null`).

Impacto:

- En real (fees no cero y assets mixtos), el net PnL puede desviarse.

#### ### Medio 2 - Infra de tests backend incompleta

pytest falla en async tests por plugin faltante (pytest-asyncio) y hay incompatibilidades de fixtures/versión pandas.

Impacto:

- Menor confianza para cambios en motor cuantitativo/riesgo.

## 4) Verificacion puntual: transacciones y ganancias

#### ### 4.1 Transacciones

- Hay **42721** proposals historicas.
- Solo **4** en estado `executed` (historico) y **0** ejecutadas en ultimas 24h.
- Ultima ejecucion exitosa registrada: **2026-02-17**.

Conclusion:

- La canalizacion de ejecucion hoy no esta funcionando de forma estable.

#### ### 4.2 Ganancia (PnL mostrado)

Se recalculo sobre los registros actuales de `positions`:

- `positions\_total=3` (2 abiertas, 1 cerrada)
- Mismatches formula interna vs guardado:

- open\_unrealized\_mismatches=0

- closed\_realized\_mismatches=0

- closed\_realized\_pct\_mismatches=0

Conclusion:

- **\*\*Para esos registros puntuales\*\***, la ganancia mostrada coincide con la formula implementada.
- **\*\*No alcanza para validar rendimiento real del sistema\*\***, porque el volumen de trades exitosos recientes es casi nulo y hay miles de errores operativos.

## 5) Estado de ejecucion y base de datos

Tablas clave verificadas en DB: presentes (trade\_proposals, positions, risk\_events, reconciliation\_runs, klines\_ohlcv, etc.).

Observaciones:

- `reconciliation\_runs` registra actividad sana y frecuente (sin divergencias recientes), pero no compensa el problema principal de ejecucion de proposals.
- `account\_snapshots` historico muy escaso y desactualizado para soporte de controles diarios robustos.
- `performance\_metrics` vacio (sin metricas rolling utiles para decision de go-live).

## 6) Como pasarlo a dinero real (ruta segura)

**No recomendable pasar hoy a real.**

Secuencia minima obligatoria:

1. Corregir causa raiz de 400 en ejecucion (proxy/firma/parametros/permissions).
2. Llevar error rate de ejecucion a <1% sostenido (minimo 2 semanas).
3. Resolver build/typecheck/tests para asegurar deploys confiables.
4. Unificar un solo motor de ejecucion (evitar drift entre fallback Next y backend Python).
5. Corregir bugs de PnL/cierre parcial y normalizar comisiones por asset en USDT equivalente.
6. Endurecer seguridad:
  - remover secretos hardcodeados
  - BACKEND\_SECRET obligatorio en produccion
  - rotar claves expuestas
7. Implementar adapter real (binance-mainnet) separado de testnet con feature flags estrictos.
8. Fase de "shadow mode":
  - generar senales y validar sin ejecutar real
  - reconciliar expected vs fill real de forma pasiva
9. Fase de capital minimo:
  - tamano nominal ultrabajo
  - limites duros de perdida diaria y kill switch externo
10. Requisito de graduacion antes de escalar:
  - Sharpe > 0

- drawdown bajo control
- reconciliación limpia
- sin dead letters pendientes

## 7) Conclusion final

- **\*\*Transacciones:\*\*** actualmente no son correctas a nivel operativo (miles de fallos de ejecución recientes).
- **\*\*Ganancia reflejada:\*\*** fórmula consistente en los pocos registros históricos cerrados, pero no representativa para confiar en performance real del bot hoy.
- **\*\*Paso a real:\*\*** bloqueado hasta resolver ejecución, consistencia técnica, seguridad y validación estadística sostenida.