GUÍA MAESTRA COMPLETA

Integración de GPT-5.1 y GPT-5-mini
en Bot de Trading Algorítmico Profesional

Documento consolidado con arquitectura total, prompts avanzados, diseño de módulos, lógica segura, flujos internos, validaciones matemáticas y estructura final del proyecto.

Versión Unificada - Compilación de 4 documentos técnicos
Para uso con bots de trading en Binance Futuros

TABLA DE CONTENIDOS

1. OBJETIVO Y FILOSOFÍA DEL SISTEMA

Este documento actúa como fuente completa, detallada y autosuficiente de conocimiento técnico sobre cómo integrar GPT-5.1 y GPT-5-mini dentro de un sistema de trading algorítmico. La intención es que otra IA pueda leer este documento, extraer toda la información clave y generar implementaciones completas sin necesidad de consultar fuentes externas.

1.1 Filosofía del Sistema

Un bot profesional NO es un bot que predice. Es un bot que:

- Selecciona contextos favorables
- Controla riesgo estrictamente
- No improvisa
- Reduce exposición en entornos hostiles
- Usa IA como clasificador y no como generador aleatorio

1.2 Propiedades Fundamentales

Un bot exitoso debe cumplir tres propiedades fundamentales:

- **Riesgo controlado:** Límites estrictos de exposición, pérdida y apalancamiento.
- **Consistencia estadística:** Decisiones basadas en ventaja estadística comprobable.
- **Robustez ante errores:** Manejo de fallos de red, API y datos.

Estas propiedades se alcanzan con: decisiones IA encapsuladas en JSON, validaciones duras de riesgo NO delegadas a la IA, estado del mercado resumido y limpio, prohibición de accesos directos del modelo a órdenes reales, e interpretación determinista del output del modelo.

2. MODELOS GPT-5: CARACTERÍSTICAS Y ESTRATEGIA DE USO

2.1 GPT-5.1 (Modelo Insignia)

Modelo tope de gama para tareas de alto razonamiento, análisis complejos, agentes y código. Permite controlar el esfuerzo de razonamiento (reasoning.effort) y es ideal para:

- Análisis de riesgo profundo
- Evaluación de estrategias
- Interpretación de backtests
- Lógica de alto nivel
- Diseño de estrategia
- Análisis post-mortem de pérdidas
- Generación de código y mejoras

2.2 GPT-5-mini (Modelo Eficiente)

Versión más rápida y costo-eficiente de la familia GPT-5. Pensada para casos de uso con muchas llamadas, como bots de trading que se ejecutan 24/7. Características:

- Muy buen rendimiento para lógica clara y bien estructurada
- Ideal cuando la información de mercado ya viene pre-procesada
- Económico para operaciones frecuentes
- Soporta reasoning.effort niveles bajos/medios
- Recomendado como motor principal del bot 24/7

2.3 Estrategia Recomendada de Uso

IMPORTANTE: Usar GPT-5-mini como modelo principal para decisiones frecuentes (cada vela, cada evento de mercado) y reservar GPT-5.1 para tareas más pesadas: revisión diaria/semanal de performance, análisis de drawdown, evaluación de nuevas reglas de gestión de riesgo o generación de código / mejoras de estrategia.

2.4 Tabla Comparativa

Característica	GPT-5.1	GPT-5-mini
Uso principal	Análisis profundo	Decisiones frecuentes
Costo	Alto	Bajo
Velocidad	Moderada	Rápida
Reasoning	low/medium/high	none/low

Frecuencia	Diaria/Semanal	Por vela/evento

3. API DE OPENAI - ESPECIFICACIÓN COMPLETA

3.1 Información Base

Base URL general (REST): https://api.openai.com/v1

Autenticación: Header Authorization con token tipo Bearer.

SEGURIDAD: Nunca exponer la API key en el frontend. Centralizar llamadas en un módulo único.

3.2 Headers Esenciales

```
Authorization: Bearer YOUR_API_KEY
Content-Type: application/json
```

3.3 Buenas Prácticas de Conexión

- Nunca exponer API keys en frontend
- Centralizar llamadas en un módulo único (ej: openai_client.py)
- Activar logs de request/response para auditoría
- El resto del sistema solo llama funciones de alto nivel como obtenerDecisionTrading(estadoMercado)
- El costo depende de la cantidad de tokens de entrada (input) y salida (output)

4. ENDPOINT /v1/responses - IMPLEMENTACIÓN DETALLADA

El endpoint /v1/responses es la interfaz más reciente y avanzada. Combina capacidades de chat, herramientas, razonamiento y salida estructurada en una sola API. Es el endpoint recomendado para GPT-5.1 y GPT-5-mini.

4.1 Características del Endpoint

- Reasoning avanzado
- Tools/function calling
- Salida JSON schema
- Manejo de roles (system, user, assistant, developer)
- Control total del output
- Streaming opcional

4.2 Estructura Básica de Request

```
POST https://api.openai.com/v1/responses
Headers:
    Authorization: Bearer TU_API_KEY
    Content-Type: application/json

Cuerpo JSON:
{
    "model": "gpt-5-mini",
    "input": [
        {"role": "system", "content": "Eres el motor de decisiones..."},
        {"role": "user", "content": "Estado de mercado: {...}"}
    ],
    "temperature": 0.1,
    "max_output_tokens": 300
}
```

4.3 Campos Clave del Request

- **model:** gpt-5.1 o gpt-5-mini
- **input:** Lista de mensajes con rol (system, user, assistant, developer)
- **temperature:** Controla aleatoriedad; para trading se recomienda entre 0.0 y 0.3
- **max_output_tokens:** Límite de tokens de salida; útil para controlar costo
- **reasoning:** Bloque para controlar esfuerzo de razonamiento
- **response_format:** Definición del formato de salida (JSON schema)

4.4 Request Completo Recomendado

```
{
    "model": "gpt-5-mini",
    "input": [...],
    "response_format": {...},
    "reasoning": {"effort": "none"},
    "max_output_tokens": 200,
    "temperature": 0.0
}
```

Reglas: Usar temperature baja para trading. Respuesta SIEMPRE estructurada. Reasoning mínimo para llamadas frecuentes. Reasoning medio/alto solo para análisis puntuales.

5. CONTROL DE RAZONAMIENTO (REASONING)

GPT-5.1 y GPT-5-mini permiten un control explícito del esfuerzo de razonamiento mediante un bloque reasoning en el endpoint /v1/responses. El razonamiento permite que el modelo procese más profundamente antes de responder.

5.1 Sintaxis

```
"reasoning": { "effort": "low" }
```

5.2 Niveles de Razonamiento

Nivel	Descripción	Uso Recomendado
none	Sin razonamiento adicional, respuesta más rápida y barata	Decisiones simples, alta frecuencia
low	Un poco de razonamiento extra	Decisiones algo complejas
medium	Más profundidad	Análisis de contexto, múltiples posiciones
high	Máximo esfuerzo de razonamiento, costoso y lento	Backtests, análisis post-mortem

5.3 Estrategia de Razonamiento para Trading

En un bot de trading que hace muchas llamadas, tiene sentido:

- Usar gpt-5-mini normalmente con poco o ningún razonamiento extra (none/low)
- Reservar gpt-5.1 con reasoning medium/high solo para tareas offline o poco frecuentes
- Análisis diarios: medium
- Análisis semanales/mensuales: high
- Decisiones en tiempo real: none o low

6. SALIDA ESTRUCTURADA JSON PARA TRADING

Para un bot de trading es CRÍTICO que la respuesta sea fácil de parsear. La idea es que el modelo responda siempre en JSON con un esquema fijo: acción, tamaño, stops, etc. Esto se consigue configurando response_format en el API.

6.1 JSON Genérico (Básico)

```
"response_format": { "type": "json_object" }
```

El modelo sabe que debe generar un objeto JSON válido, pero sin un esquema formal. Es mejor que nada, pero lo ideal es ir un paso más allá con json_schema.

6.2 JSON Schema (Muy Recomendado)

```
"response_format": {
    "type": "json_schema",
    "json_schema": {
        "name": "trading_decision",
        "schema": {
            "type": "object",
            "properties": {
                "action": {
                    "type": "string",
                    "enum": ["hold", "buy", "sell", "add", "reduce", "close",
                             "flip_long", "flip_short"]
                },
                "size_fraction": {"type": "number"},
                "stop_loss_price": {"type": "number"},
                "take_profit_price": {"type": "number"},
                "confidence": {"type": "number"},
                "reason": {"type": "string"}
            },
            "required": ["action"]
        }
    }
}
```

6.3 Ventajas del JSON Schema

- El modelo entiende claramente qué campos debe producir
- Tu código puede validar el JSON contra el esquema
- Reduce errores de parsing
- Hace más robusta la integración
- Permite validaciones automáticas de tipos y valores

6.4 Ejemplo de Salida Esperada

```
{
    "action": "reduce",
    "size_fraction": 0.5,
    "stop_loss_price": 96000,
    "take_profit_price": 99500,
    "confidence": 0.85,
    "reason": "RSI alto, beneficios suficientes, reducir exposición."
}
```

7. TOOLS Y FUNCTION CALLING

El API permite definir funciones (tools) que el modelo puede invocar. En lugar de responder texto, el modelo puede indicar que quiere llamar a una función con argumentos JSON. Esto es muy útil para un bot de trading, pero hay que usarlo con EXTREMO CUIDADO.

ADVERTENCIA CRÍTICA: Llamar directamente órdenes reales desde el modelo sin validaciones puede ser MUY PELIGROSO. Nunca ejecutes directamente una orden real solo porque el modelo la pidió.

7.1 Definición de Tool para Crear Orden

```
"tools": [
    {
        "type": "function",
        "function": {
            "name": "create_order",
            "description": "Crea una orden en Binance Futuros",
            "parameters": {
                "type": "object",
                "properties": {
                    "symbol": {"type": "string"},
                    "side": {"type": "string", "enum": ["BUY", "SELL"]},
                    "size_usdt": {"type": "number"},
                    "type": {"type": "string", "enum": ["LIMIT", "MARKET"]}
                },
                "required": ["symbol", "side", "size_usdt"]
            }
        }
    }
],
"tool_choice": "auto"
```

7.2 Flujo de Tool Calling Seguro

1. El modelo devuelve una llamada de tool (tool call)
2. Tu backend interpreta la llamada
3. Tu código aplica capa de reglas duras (riesgo, límites, horarios)
4. Solo si pasa todas las validaciones, se manda la orden al exchange
5. Loguear toda la interacción para auditoría

8. ENDPOINT ALTERNATIVO: /v1/chat/completions

El endpoint /v1/chat/completions es la interfaz clásica de chat. En muchos casos sigue funcionando bien y es compatible con GPT-5.1 y GPT-5-mini. Usarlo solo si ya existe infraestructura legacy basada en chat completions.

8.1 Diferencias con /v1/responses

- El control de razonamiento se hace con reasoning_effort (en vez de reasoning.effort)
- El límite de tokens de salida suele llamarse max_completion_tokens
- Algunos parámetros cambian de nombre
- No incluye de forma nativa ciertas funciones avanzadas

8.2 Ejemplo de Request

```
{
    "model": "gpt-5-mini",
    "messages": [
        {"role": "system", "content": "Eres un bot de trading..."},
        {"role": "user", "content": "Estado de mercado: ..."}
    ],
    "temperature": 0.1,
    "reasoning_effort": "low",
    "response_format": { "type": "json_object" }
}
```

Recomendación: Para nuevos desarrollos se recomienda usar /v1/responses. Esta información se incluye solo por compatibilidad con sistemas existentes.

9. ARQUITECTURA MAESTRA DEL BOT DE TRADING

9.1 Estructura de Proyecto Recomendada

```
/src
    /core
        risk_manager.py         # Motor de riesgo
        execution_engine.py     # Motor de ejecución
        position_tracker.py     # Tracking de posiciones
    /ai
        openai_client.py        # Cliente OpenAI
        decision_engine.py      # Motor de decisiones
        prompts.py              # Prompts del sistema
        schemas.py              # JSON Schemas
    /data
        market_feed.py          # Feed de mercado
        indicators.py           # Cálculo de indicadores
    /utils
        logger.py               # Logging institucional
        config.py               # Configuración
    main.py                     # Entry point
```

9.2 Pipeline de Decisión

- **Data Layer:** Obtiene velas, trades, OI, funding, liquidez. Normaliza datos.
- **Technical Engine:** Calcula indicadores (RSI, MACD, EMA, ATR, VWAP). Clasifica tendencia. Detecta rupturas.
- **IA Layer (GPT-5-mini):** Toma decisiones micro. Optimización de riesgo. Control emocional inexistente.
- **Executor:** Valida decisiones. Ejecuta órdenes maker/taker. Control de slippage. Seguridad.
- **Risk Master Module:** Stop diario. Stop semanal. Control de leverage. Circuit breakers.

9.3 Funciones del Módulo OpenAI

- send_decision_request(market_state) - Envía estado y recibe decisión
- send_advanced_analysis(report_data) - Para análisis con GPT-5.1
- parse_json_response(response) - Parsea y valida respuesta
- handle_errors_and_timeouts() - Manejo de errores

9.4 Lo que Hace TU CÓDIGO (No el modelo)

- Conecta con el exchange (Binance Futuros) y obtiene precios, posiciones, órdenes
- Calcula indicadores y features: RSI, medias móviles, volatilidad, volumen, PnL
- Aplica reglas de riesgo duras: límite de apalancamiento, tamaño máximo, pérdida máxima
- Construye un JSON compacto con el estado relevante del mercado
- Valida la respuesta de la IA antes de ejecutar
- Ejecuta o rechaza la orden según validaciones

9.5 Lo que Hace el MODELO

- Recibe el estado de mercado en forma de JSON
- Evalúa las condiciones del mercado
- Elige una acción entre un conjunto discreto

- Devuelve un JSON con la decisión estructurada
- Proporciona justificación de la decisión

10. ESTADO DE MERCADO - ESTRUCTURA OFICIAL

La IA NO debe hacer análisis técnico desde cero. Debes entregarle todo ya calculado en un formato JSON compacto y bien estructurado.

10.1 JSON de Estado Completo

```
{
    "symbol": "BTCUSDT",
    "timeframe": "5m",
    "timestamp": 1700000000,
    "price": 98750.3,
    "orderbook_strength": 0.64,
    "trend": "up",
    "momentum_score": 0.72,
    "volatility": 0.031,
    "indicators": {
        "ema_fast": 98520,
        "ema_slow": 96000,
        "rsi": 68,
        "atr": 520,
        "vwap": 97880,
        "trend_strength": 0.8
    },
    "position": {
        "side": "long",
        "size_usdt": 12000,
        "entry_price": 94000,
        "pnl_unrealized": 3850
    },
    "risk_limits": {
        "max_position_usdt": 25000,
        "max_daily_loss": 400,
        "max_leverage": 3
    },
    "allowed_actions": ["hold", "add", "reduce", "close"]
}
```

10.2 Campos Obligatorios

- **symbol:** Par de trading (ej: BTCUSDT)
- **timeframe:** Temporalidad de análisis (ej: 5m, 15m, 1h)
- **price:** Precio actual
- **indicators:** Objeto con todos los indicadores calculados
- **position:** Estado de la posición actual (o null si no hay)
- **risk_limits:** Límites de riesgo configurados
- **allowed_actions:** Lista de acciones permitidas en este momento

10.3 Indicadores Recomendados

- RSI (Relative Strength Index)
- EMA fast y slow (Exponential Moving Averages)
- ATR (Average True Range)
- VWAP (Volume Weighted Average Price)
- Volatilidad (desviación estándar o similar)
- Trend strength (fuerza de la tendencia)
- MACD (opcional)

- Momentum score (opcional)

11. DISEÑO DE PROMPTS PROFESIONALES

Un buen mensaje de sistema puede marcar mucha diferencia en la calidad de las decisiones. A continuación se presentan prompts optimizados para diferentes niveles del sistema.

11.1 Prompt de SYSTEM (Nivel Hedge Fund)

```
Eres el motor de decisiones de un bot de trading profesional en mercados de futuros.
Debes actuar con disciplina institucional.
Tu objetivo es maximizar beneficios ajustados al riesgo.
Las decisiones deben ser simples, claras y justificadas.
Nunca excedas los límites de riesgo.
Si no hay ventaja estadística clara, responde 'hold'.
Responde únicamente con JSON válido conforme al esquema proporcionado.
```

11.2 Prompt de DEVELOPER (Reglas Internas)

```
No generes texto fuera del JSON.
No predigas el futuro.
No inventes indicadores.
Trabaja solo con los datos proporcionados.
Evita cambios drásticos en posiciones.
Si el mercado está incierto → hold.
Si el riesgo excede límites → reduce o close.
No improvises indicadores.
Mantén decisiones simples, claras y justificadas.
```

11.3 Prompt de USER

```
Estado de mercado: {json}
```

11.4 Prompt para Análisis Profundo (GPT-5.1)

```
Analiza condiciones macro, liquidez, volatilidad y estructura de mercado.
Evalúa compatibilidad de la estrategia con el entorno actual.
Identifica riesgos invisibles, fases de mercado peligrosas y oportunidades.
Produce reporte avanzado con sugerencias de ajustes estructurales.
Analiza las últimas condiciones de mercado, patrones macro, riesgo global
y probabilidad de continuación de tendencia.
Evalúa la lógica actual de la estrategia.
Produce un reporte detallado con fortalezas, debilidades y recomendaciones.
```

11.5 Prompt Ultra Optimizado para Micro-Decisiones (GPT-5-mini)

```
Evalúa únicamente los datos dados.
Identifica si existe ventaja estadística a favor.
Si la probabilidad de éxito es baja o incierta, responde 'hold'.
Usa manejo de riesgo conservador.
Evita cambios drásticos.
Responde solo en JSON.
```

12. SEGURIDAD Y REGLAS CRÍTICAS

ADVERTENCIA: Esta sección contiene reglas de seguridad OBLIGATORIAS para evitar pérdidas catastróficas por errores de la IA.

12.1 Lo que NUNCA Debe Permitirse a la IA

✘ Aumentar posición por encima del límite
✘ Abrir operaciones directas sin validación
✘ Ignorar pérdidas diarias
✘ Cambiar stop-loss sin permiso
✘ Operar con baja liquidez
✘ Operar durante eventos extremos si está prohibido
✘ Ejecutar tool calls directamente sin validación
✘ Generar texto fuera del JSON esperado

12.2 Reglas de Seguridad Mínimas

• No ejecutar ninguna orden si la respuesta no cumple el esquema esperado
• Rechazar valores fuera de rango
• Limitar el tamaño de posición máximo, independientemente de lo que diga el modelo
• Bloquear trading si la pérdida diaria o semanal supera cierto umbral
• Loguear todas las recomendaciones del modelo y las decisiones finales
• Proteger contra ataques de prompt injection
• Restringir acciones permitidas según contexto

12.3 Circuit Breakers Obligatorios

• Detener trading si la pérdida diaria supera cierto umbral
• Bloquear si el precio es demasiado volátil
• Bloquear si funding rate es extremo
• Detener si hay 3 pérdidas consecutivas
• Bloquear durante noticias de alto impacto (si configurado)

13. VALIDACIONES MATEMÁTICAS DEL OUTPUT

Una validación mínima debe incluir las siguientes comprobaciones en código (NO delegadas a la IA):

13.1 Validaciones de Posición

```
# Validación de incremento de posición
if action in ["buy", "add"]:
    new_position = current_size + size_fraction * max_position
    assert new_position <= max_position, "Excede límite de posición"

# Validación de reducción
if action == "reduce":
    assert size_fraction <= 1, "Fracción inválida"
    assert size_fraction > 0, "Fracción debe ser positiva"

# Validación de cierre
if action == "close":
    assert position_exists, "No hay posición para cerrar"

# Validación de stop loss
if stop_loss_price is not None:
    if position_side == "long":
        assert stop_loss_price < current_price, "SL inválido para long"
    else:
        assert stop_loss_price > current_price, "SL inválido para short"

# Validación de take profit
if take_profit_price is not None:
    if position_side == "long":
        assert take_profit_price > current_price, "TP inválido para long"
    else:
        assert take_profit_price < current_price, "TP inválido para short"
```

13.2 Validaciones Durísimas (Bloqueo Automático)

```
# Bloqueo por pérdida diaria
if pnl_daily < -max_daily_loss:
    block_ai()
    return "BLOCKED"

# Bloqueo por posición excedida
if new_position > max_position:
    reject_order()
    return "REJECTED"

# Bloqueo por stop loss inválido
if stop_loss_price >= current_price and side == "long":
    reject_order()
    return "INVALID_SL"

# Bloqueo por take profit inválido
if take_profit_price <= current_price and side == "long":
    reject_order()
    return "INVALID_TP"

# Bloqueo si acción requiere posición y no existe
if action in ["reduce", "close"] and not position_exists:
    reject_order()
    return "NO_POSITION"
```

14. MOTOR DE RIESGO Y CIRCUIT BREAKERS

14.1 Funciones del Motor de Riesgo

- **check_daily_loss():** Verifica si la pérdida diaria excede el límite
- **enforce_position_limit():** Asegura que la posición no exceda el máximo
- **enforce_leverage_limit():** Controla el apalancamiento máximo permitido
- **block_on_volatility_spike():** Bloquea trading durante volatilidad extrema
- **check_consecutive_losses():** Detecta rachas de pérdidas
- **check_funding_rate():** Evalúa costos de funding extremos

14.2 Implementación de Circuit Breakers

```
class RiskManager:
    def __init__(self, config):
        self.max_daily_loss = config.max_daily_loss
        self.max_position = config.max_position
        self.max_leverage = config.max_leverage
        self.max_consecutive_losses = 3

    def check_all(self, market_state, proposed_action):
        checks = [
            self.check_daily_loss(market_state),
            self.check_position_limit(proposed_action),
            self.check_leverage(proposed_action),
            self.check_volatility(market_state),
            self.check_consecutive_losses()
        ]
        return all(checks)

    def validate(self, decision):
        if not self.check_all(self.market_state, decision):
            return None  # Rechazar
        return decision  # Aprobar
```

15. MOTOR DE EJECUCIÓN

15.1 Funciones del Motor de Ejecución

- **place_limit_order():** Coloca órdenes límite (maker)
- **place_market_order():** Coloca órdenes de mercado (taker)
- **calculate_slippage():** Estima el slippage esperado
- **adjust_to_maker_preference():** Optimiza para fees de maker
- **cancel_order():** Cancela órdenes pendientes
- **modify_order():** Modifica órdenes existentes

15.2 Ciclo Completo del Bot

```
# Loop principal del bot
while running:
    # 1. Obtener estado del mercado
    estado = market_feed.get_state()

    # 2. Obtener decisión de la IA
    decision = ai.decision_engine(estado)

    # 3. Validar contra reglas de riesgo
    validada = risk_manager.validate(decision)

    # 4. Ejecutar si es válida
    if validada:
        result = execution_engine.execute(validada)
    else:
        result = "REJECTED"

    # 5. Loguear todo
    logger.save(estado, decision, validada, result)

    # 6. Esperar siguiente ciclo
    await asyncio.sleep(interval)
```

16. LÓGICA DE DECISIONES DE LA IA

16.1 Reglas Profesionales de Trading

- Reducir exposición cuando volatilidad aumenta
- Añadir posición solo con tendencia fuerte + momentum positivo
- No operar ranges estrechos
- No operar justo antes de noticias (si configurado)
- No aumentar exposición tras drawdown
- No operar si tres señales anteriores fueron pérdidas
- Considerar fees maker/taker en cada decisión
- Recomendar reducción si RSI > 70 con beneficio
- Agregar solo con momentum fuerte y volatilidad moderada
- Evitar operar contra tendencia en timeframes altos

16.2 Señales de NO Operar

- RSI 50 ±2 → zona muerta
- Precio atrapado entre EMAs
- Volumen extremadamente bajo
- Tendencia mixta o indefinida
- Alta volatilidad sin dirección
- Consolidación prolongada

16.3 Condiciones de Alta Confianza (Trade Ideal)

- Momentum alto y creciente
- Estructura de mercado clara
- Volatilidad moderada
- Señales alineadas en 3+ indicadores
- Tendencia confirmada en múltiples timeframes
- Volumen creciente en dirección del trade

16.4 Lógica Recomendada para la IA

- Favorecer trades con asimetría clara (reward > risk)
- Penalizar drawdowns prolongados
- Favorecer reducción cuando momentum se debilita
- Preferir NO operar frente a señales ambiguas
- Favorecer trades pequeños y frecuentes sobre grandes y raros
- Priorizar protección de capital
- Aprovechar pullbacks en tendencia
- Evitar operar en consolidación larga

17. MANEJO DE ERRORES Y TIMEOUTS

17.1 Estrategia de Reintentos

```
import time
from functools import wraps

def retry_with_backoff(max_retries=3, base_delay=1):
    def decorator(func):
        @wraps(func)
        def wrapper(*args, **kwargs):
            for attempt in range(max_retries):
                try:
                    return func(*args, **kwargs)
                except TransientError as e:
                    if attempt == max_retries - 1:
                        raise
                    delay = base_delay * (2 ** attempt)
                    time.sleep(delay)
            return None
        return wrapper
    return decorator
```

17.2 Manejo de Errores HTTP

- **4xx:** Parámetros inválidos, problemas con API key. Loguear y no reintentar.
- **429:** Rate limit. Implementar backoff exponencial.
- **5xx:** Error del servidor. Reintentar con backoff.
- **Timeout:** Establecer timeout razonable. Si no responde, no abrir posiciones.

17.3 Circuit Breaker para API

```
class APICircuitBreaker:
    def __init__(self, failure_threshold=5, reset_timeout=60):
        self.failures = 0
        self.threshold = failure_threshold
        self.reset_timeout = reset_timeout
        self.last_failure = None
        self.is_open = False

    def record_failure(self):
        self.failures += 1
        self.last_failure = time.time()
        if self.failures >= self.threshold:
            self.is_open = True

    def can_proceed(self):
        if not self.is_open:
            return True
        if time.time() - self.last_failure > self.reset_timeout:
            self.is_open = False
            self.failures = 0
            return True
        return False
```

18. LOGGING Y AUDITORÍA INSTITUCIONAL

18.1 Información a Loguear

- Estado completo del mercado (input a la IA)
- Decisión cruda de la IA (output)
- Decisión validada (post-filtro de riesgo)
- Resultado de la ejecución (orden colocada/rechazada)
- PnL después del trade
- Timestamps de cada paso
- Tokens consumidos
- Latencia de la llamada API

18.2 Formato de Log Recomendado

```
{
    "timestamp": "2024-01-15T10:30:00Z",
    "cycle_id": "uuid-here",
    "market_state": {...},
    "ai_decision": {...},
    "validated_decision": {...},
    "execution_result": {
        "status": "FILLED",
        "order_id": "123456",
        "fill_price": 98750.5
    },
    "metrics": {
        "tokens_used": 245,
        "latency_ms": 342,
        "model": "gpt-5-mini"
    }
}
```

19. STREAMING DE RESPUESTAS

El API soporta streaming: en lugar de esperar a que la respuesta esté completa, recibes fragmentos (eventos) mientras el modelo genera. En un bot de trading esto normalmente no es crítico, pero puede servir para interfaces en tiempo real o para reducir latencia percibida.

19.1 Activación del Streaming

```
# En /v1/responses, se activa pasando:
{
    "model": "gpt-5-mini",
    "input": [...],
    "stream": true
}

# Tu cliente debe leer eventos de servidor (Server-Sent Events)
# y reconstruir la salida.
```

Nota: Para lógica de backoffice del bot no es obligatorio; muchas integraciones funcionan bien sin streaming.

20. PARÁMETROS RECOMENDADOS

20.1 GPT-5-mini (Decisiones Frecuentes)

Parámetro	Valor
model	gpt-5-mini
temperature	0.0 - 0.2
max_output_tokens	200 - 400
response_format	json_schema
reasoning.effort	none o low

20.2 GPT-5.1 (Análisis Puntuales)

Parámetro	Valor
model	gpt-5.1
temperature	0.1 - 0.4
max_output_tokens	800 - 2000
response_format	json o texto según finalidad
reasoning.effort	low / medium / high

21. REGLAS ANTI-OVERFITTING

Para evitar que el sistema se sobreajuste a patrones históricos que no se repetirán:

- No permitir lógica demasiado específica en prompts
- No dejar que la IA 'modele' patrones complejos
- Mantener decisiones discretas y simples
- Mantener indicadores limitados (no más de 6-8)
- No dejar que la IA invente señales o indicadores
- Validar en múltiples períodos de mercado
- Usar out-of-sample testing
- Evitar optimización excesiva de parámetros

21.1 Razones por las que GPT-5-mini Funciona en Trading

- Capacidad de clasificación compleja
- Evaluación de múltiples factores simultáneamente
- Adaptativo sin recalibración manual
- Minimiza decisiones impulsivas (sin emociones)
- Suficiente poder para entornos discretos
- No memoriza patrones específicos del mercado

22. CHECKLIST FINAL DE IMPLEMENTACIÓN

Usa esta lista para verificar que tu implementación está completa:

- ■ Crear módulo cliente que envíe peticiones a /v1/responses con autenticación
- ■ Implementar función de alto nivel para obtener decisiones de trading
- ■ Soportar reasoning.effort con diferentes niveles
- ■ Definir y usar response_format con json_schema
- ■ Diseñar prompts de sistema claros (system, developer, user)
- ■ Validar todos los outputs contra el esquema
- ■ Integrar arquitectura de trading completa
- ■ Implementar lógica de riesgo con circuit breakers
- ■ Configurar logs completos e institucionales
- ■ Implementar manejo de errores con backoff exponencial
- ■ Usar gpt-5-mini para llamadas frecuentes
- ■ Reservar gpt-5.1 para análisis puntuales
- ■ Ajustar temperature y max_output_tokens según pruebas
- ■ Implementar validaciones matemáticas del output
- ■ Crear motor de ejecución con control de slippage
- ■ Implementar motor de riesgo con límites duros
- ■ Configurar allowed_actions dinámicamente
- ■ Implementar auditoría y trazabilidad completa
- ■ Probar en paper trading antes de producción
- ■ Documentar toda la arquitectura y flujos

Con este documento, otra IA o desarrollador debería tener una visión completa para integrar GPT-5.1 y GPT-5-mini de forma segura y efectiva dentro de un bot de trading de futuros profesional.

— Fin del Documento —

Versión Unificada compilada de 4 documentos técnicos originales