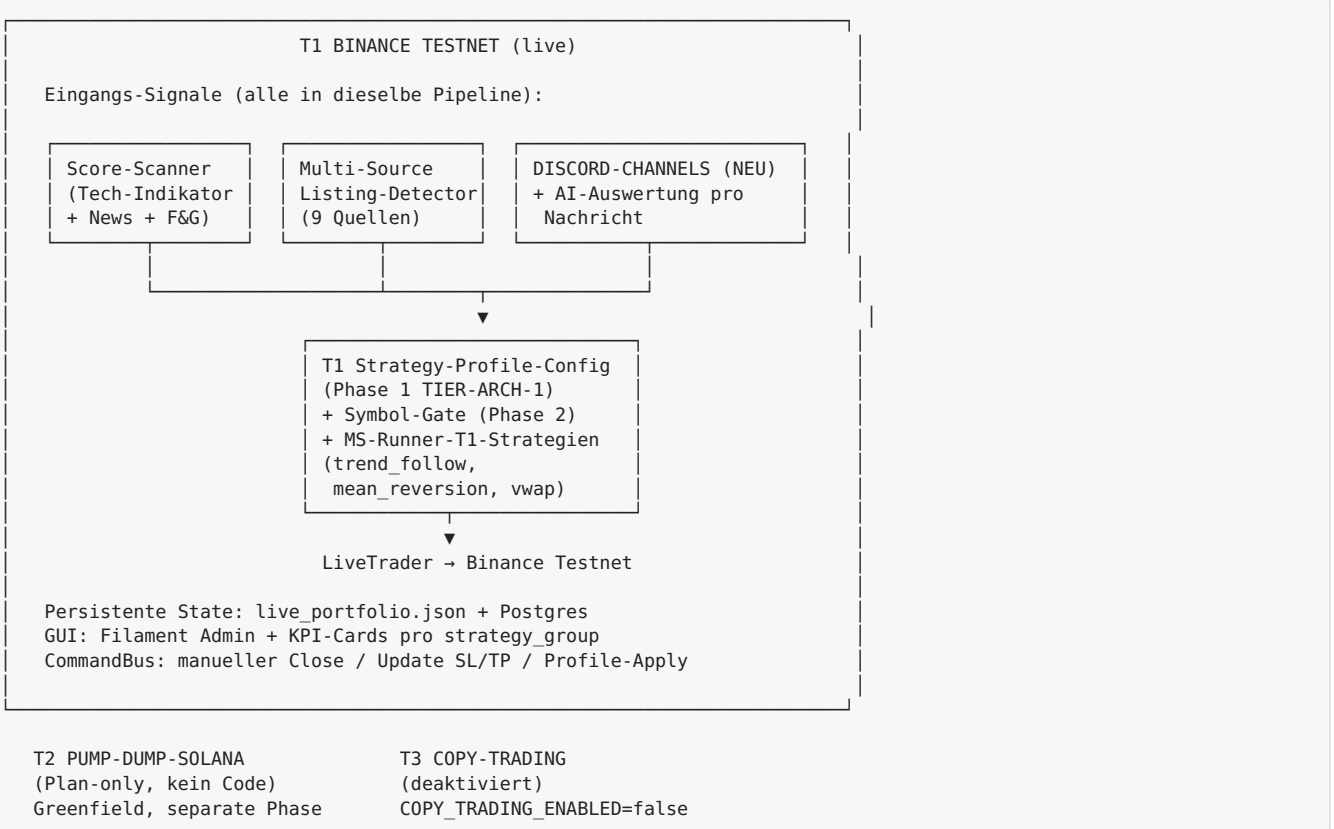
Steve-TradingBot — 2-Wochen-Roadmap: T1-Konsolidierung + Discord + AI

Erstellt: 2026-05-16 · Branch `master` · HEAD `a6a629d` **Zeitraum:** 2026-05-19 (Mo) bis 2026-05-31 (So) — 2 Kalenderwochen
Status: Plan, keine Implementation in diesem Dokument.

0. Executive Summary

Aspekt	Wert
Fokus	T1 (Binance Spot, Testnet) — vollständig konsolidieren und erweitern
T2 / T3	bleiben **deaktiviert** für 2 Wochen (T2 Plan-Phase, T3 ganz pausiert)
Neue Capability	Discord-Signal-Channel-Integration + AI-gestützte Text-Auswertung
Strategischer Output	Refactor-vs-Rewrite-Plan-PDF als Operator-Entscheidungsgrundlage
Boundaries	Mainnet bleibt 0. MS-DRY_RUN=false bleibt 0. Solana-Trading bleibt 0.
Aufwand	~9-11 Arbeitstage netto über 2 Wochen verteilt

1. Big-Picture Vision (Ende Woche 2)



2. Phasenreihenfolge (operator-bestätigt 2026-05-16)

#	Phase	Aufwand	Woche
1	**TIER-ARCH-CONTRACT-1** — formale Tier-Architektur (T1 aktiv, T2/T3 deaktiviert) als Code-Konstanten + Tests	1.5 Tage	W1 Mo-Di
2	**T1-BINANCE-SYMBOL-GATE-1** — T1 nur Binance-USDT-Symbole, Solana-only-Tokens explizit ausgeschlossen	1 Tag	W1 Di-Mi
3	**EXEC-MODE-LABEL-3 / Paper-Cleanup** — `--paper`, `PaperTrader`, `paper_portfolio.json` retiren + Doku	1 Tag	W1 Mi-Do
4	**REFACTOR-VS-REWRITE-PLAN-PDF** — finales Architektur-Entscheidungs-PDF	0.5 Tag	W1 Do-Fr
5	**T1-DISCORD-SIGNAL-INTAKE-1** — Discord-Bot/Webhook + Channel-Whitelist + Persistence	2.5 Tage	W2 Mo-Mi
6	**T1-AI-SIGNAL-ANALYSIS-1** — LLM-basierte Auswertung der Discord-Nachrichten	2.5 Tage	W2 Mi-Fr
7	**T2-SOLANA-SHADOW-1 Plan** — Plan-PDF für T2 Greenfield (KEIN Code)	0.5 Tag	W2 Sa

Gesamt **~9 Tage Arbeit** mit Puffer für Review-Zyklen + 24-72h Live-Beobachtungs-Fenster pro Cutover.

---

3. Phase 1 — TIER-ARCH-CONTRACT-1

3.1 Ziel

Tier-Architektur als **Code-Vertrag** kodifizieren: welcher Tier ist aktiv, welcher deaktiviert, welche Beziehungen gelten. Damit kann zukünftiger Code beim Import gegen den Vertrag prüfen, statt ad-hoc-Boolean-Flags überall.

3.2 Konkretes Lieferziel

Neue Datei `trading/tier_arch.py`:

```
class TierStatus(Enum):
    ACTIVE         = "active"
    PLAN_ONLY      = "plan_only"      # Architekt-Phase, kein Code
    DEACTIVATED    = "deactivated"
    SHADOW         = "shadow"          # zukünftig: läuft, aber tradet nicht

TIER_ARCH = {
    "t1_core":      TierStatus.ACTIVE,
    "t2_pump_dump": TierStatus.PLAN_ONLY,
    "t3_copy_trading": TierStatus.DEACTIVATED,
}

def assert_tier_active(group: str) -> None:
    """Erzwingt: jeder Code-Pfad, der auf einem Tier Trades auslösen will,
    muss den ACTIVE-Status prüfen. T2/T3-Aufrufe fail-fast hier."""

def is_active(group: str) -> bool: ...
```

3.3 Wo wird das genutzt

Stelle	Vorher	Nachher
`main.py:run_scan_cycle` (T1-emit)	implizit aktiv	`assert_tier_active('t1_core')` am Top
`main.py:_process_t3_forwarded_signals`	aktiv	`if not is_active('t2_pump_dump'): return`
`multi_strategy_runner.py`	aktiv	`if not is_active('t1_core'): return []`
Wallet-Tracker (T3 Copy-Trading)	bereits `COPY_TRADING_ENABLED=false`	zusätzlich `assert_tier_active('t3_copy_trading')` fail-fast

3.4 Tests

Test	Was
`test_tier_arch_t1_is_active`	T1 = active
`test_tier_arch_t2_is_plan_only`	T2 = plan_only
`test_tier_arch_t3_is_deactivated`	T3 = deactivated
`test_assert_tier_active_raises_for_t2`	wenn Code-Pfad T2 erreicht → RuntimeError
`test_main_py_uses_assert_tier_active`	AST: main.py importiert tier_arch und ruft assert_tier_active
`test_msr_uses_is_active_check`	AST: multi_strategy_runner.py top-level check
`test_t3_forwarded_bridge_uses_is_active_check`	AST: T2 forward path gates on tier-status

3.5 Aufwand

Block	Zeit
Code-Datei `tier_arch.py`	1 h
Wiring an 4 Stellen	2 h
7 Tests	2 h
Build + Cutover + Verify	1 h
Doku-Update	30 min

Gesamt ~6 h (halb-Tag).

3.6 Boundaries & Stop-Gates

- 0× DB-Migration
- 0× Strategie-Parameter-Änderung
- 0× Mainnet
- Bot-Restart via SOT-1d (kein Worker-Touch)

3.7 Operator-Decisions vor Start

1. Naming-Bestätigung: `tier_arch.py` ok?
2. T2-Status: `PLAN_ONLY` oder `DEACTIVATED`? (Empfehlung: PLAN_ONLY, damit Architektur-Vorbereitung klar ist)
3. T3-Status: bleibt definitiv `DEACTIVATED`?

---

4. Phase 2 — T1-BINANCE-SYMBOL-GATE-1

4.1 Ziel

T1 handelt **ausschließlich** Symbole, die auf Binance Spot Testnet tatsächlich handelbar sind. Heute können theoretisch Solana-only-Tokens (via T3-Forward oder web-channel) als T2-getaggte BUYs auf Binance landen — und scheitern dann, weil das Symbol auf Binance nicht existiert.

Mit dem Gate wird vor jedem `execute_buy` geprüft: - Symbol-Format korrekt (`<BASE>/USDT`) - Symbol in Binance-Testnet-Universe (cached, refresh stündlich) - Wenn nein: signal wird `rejected` mit Reason `symbol_not_on_binance`

4.2 Implementation-Skizze

Neue Methode in `LiveTrader`:

```
def is_tradable_strict(symbol: str) -> bool:
    """Markets-Cache + USDT-Quote + Spot-Only."""
```

In `run_scan_cycle` + `_process_t3_forwarded_signals` (+ MS-Runner) einmaliger Check vor `execute_buy`:

```
if not live_trader.is_tradable_strict(symbol):
    emit_decision(action='reject', reject_reason='symbol_not_on_binance', ...)
    continue
```

4.3 Tests

Test	Was
`test_binance_usdt_pair_is_tradable`	BTC/USDT → True
`test_solana_only_token_not_tradable`	PUMP/USDT (nicht auf Binance) → False
`test_invalid_symbol_format_rejected`	`BTC` (ohne /USDT) → False
`test_emit_decision_when_reject_for_symbol`	reject-row landet in decision_logs
`test_legacy_scanner_respects_gate`	T1-BUY für non-binance Symbol skipped
`test_t3_forward_respects_gate`	T3→T2 mit non-binance Symbol skipped

4.4 Aufwand

~6 h (1 Tag) — Code + Tests + Live-Verify mit dem aktuellen Universe.

4.5 Boundaries

- 0× DB-Migration
- 0× Mainnet
- T1-only Effekt; T2/T3 vom Gate unbeeinflusst (eh deaktiviert)

---

5. Phase 3 — EXEC-MODE-LABEL-3 / Paper-Cleanup

5.1 Ziel

Vollständige Retire von Paper-Trading-Resten: - `--paper` CLI-Flag aus main.py entfernen (heute deprecated no-op) - `settings.PAPER_TRADING` aus settings.py entfernen - `PaperTrader` Klasse umbenennen zu `BaseTrader` - `paper_portfolio.json` File-Path-Konstante umbenennen zu `state_portfolio.json` (Symlink für Backward-Compat) - `execution/balance_provider.PaperBalanceProvider` umbenennen zu `LocalBalanceProvider`

5.2 Sicherheits-Strategie

Da viele Tests und Sub-Module noch auf `PaperTrader` / `paper_*`-Namen verweisen: refactor in 2 Sub-Phasen.

3a — Backward-Compatible Rename: - Neuen Namen einführen, alten als Alias parallel halten - Tests grün halten - Cutover testen

3b — Alte Namen löschen (separate Operator-GO): - Wenn 3a stabil über 24h läuft - alle `PaperTrader`-Verweise zu `BaseTrader` ändern - alte Konstanten entfernen

5.3 Affected Files

File	Änderung
`trading/main.py`	`--paper` Flag-Parsing entfernen
`trading/config/settings.py`	`PAPER_TRADING` Field entfernen
`trading/execution/paper_trade.py`	Rename `PaperTrader` → `BaseTrader`, Datei-Rename `base_trader.py`
`trading/execution/live_trade.py`	Import-Update
`trading/execution/balance_provider.py`	Rename `PaperBalanceProvider` → `LocalBalanceProvider`
`trading/command_worker.py`	Imports + STATE_FILE-Konstante
`trading/tests/...`	Alle PaperTrader-Imports anpassen
`gui/app/Services/ModeLabels.php`	Mode-Label-Mapping erweitern

5.4 Aufwand

~8 h (1 Tag) für Phase 3a, +4 h für Phase 3b nach 24h-Beobachtung.

5.5 Boundaries

- 0× Trading-Logik-Änderung (reine Renames)
- 0× DB-Mutation
- 0× Mainnet
- Bot-Restart via SOT-1d

---

6. Phase 4 — REFACTOR-VS-REWRITE-PLAN-PDF

6.1 Ziel

Finales Architektur-Entscheidungs-Dokument als PDF mit: - Kostenvergleich Refactor vs Rewrite (aus heutiger Sicht) - 6-7 Wochen Hybrid-Plan mit Aufwandsschätzung pro Sub-Phase - Stop-Gates pro Phase - Migration-Path für DB / GUI / Tests - Risiken + Mitigationen

6.2 Lieferung

Markdown + PDF wie heute gewohnt, auf files.gewerbespeicher-rechner.de.

6.3 Aufwand

~4 h. Read-only, kein Code.

---

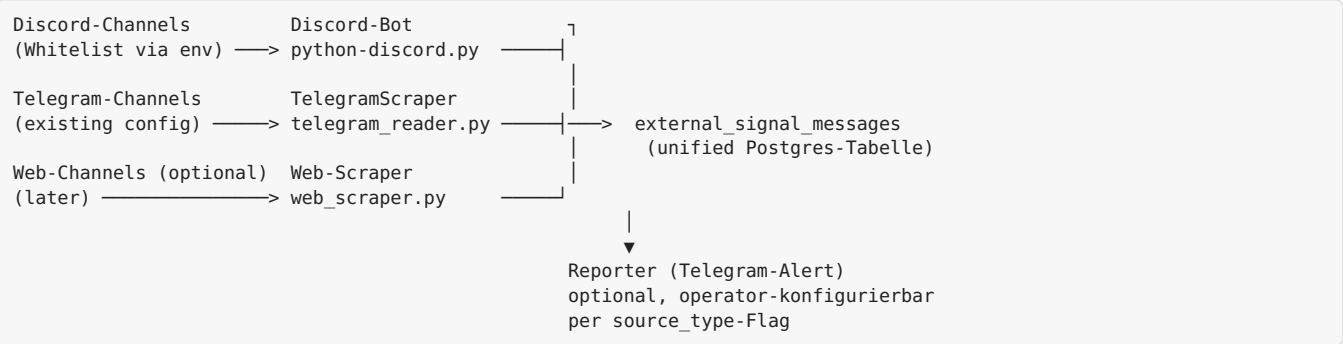
7. Phase 5 — T1-EXTERNAL-SIGNAL-CHANNELS-1 (Discord + Telegram unified)

7.1 Ziel

T1 bekommt **Discord-Channel-Integration** UND die **existierenden Telegram-Signal-Channels** werden in dieselbe Persistenz-Schicht überführt. Beide Quellen feeden in eine generische `external_signal_messages`-Tabelle, sodass die AI-Auswertung in Phase 6 **eine** Pipeline statt zwei separate brauchen.

Heute lesen die Telegram-Channels über `trading/news/telegram_*.py` (siehe `telegram_scraper.py`, `signal_channels.py`, `telegram_reader.py`) und werfen Signale direkt in den Bot-Loop. Künftig: alle Telegram- Channel-Nachrichten landen ZUSÄTZLICH in der unified-Tabelle, damit die AI sie sieht und bewerten kann. Die existierende Signal-Logik bleibt parallel intakt (Doppel-Persistenz vorerst, später konsolidieren).

7.2 Architektur (unified)



7.3 Auth + Setup

Quelle	Auth	Heute Status
Discord	`DISCORD_BOT_TOKEN` (neu, secret)	Operator legt Bot-Account an
Telegram-Channels	`TELEGRAM_API_ID` + `TELEGRAM_API_HASH` (existieren bereits in env)	bereits aktiv
Channel-Whitelists	env-Variablen `DISCORD_CHANNELS_WHITELIST` + `TELEGRAM_SIGNAL_CHANNELS` (existiert bereits in `signal_channels.py`)	teilweise da

7.4 Tabellen-Schema (unified)

Eine Postgres-Tabelle `external_signal_messages` statt zwei separate (Discord-/Telegram-spezifische). Die `source_type`-Spalte trennt sie sauber, die AI in Phase 6 verarbeitet beide einheitlich.

```
CREATE TABLE external_signal_messages (
    id              BIGSERIAL PRIMARY KEY,
    source_type     VARCHAR(20) NOT NULL,  -- 'discord' | 'telegram_channel' | 'telegram_bot' | 'web'
    source_message_id VARCHAR(80) NOT NULL,  -- Discord-ID oder Telegram-Message-ID
    channel_id      VARCHAR(80) NOT NULL,
    channel_name    VARCHAR(120),
    author          VARCHAR(120),
    raw_message     TEXT NOT NULL,
    attachments     JSON,
    ai_analysis_id  BIGINT REFERENCES external_signal_analysis(id),
    received_at     TIMESTAMP NOT NULL,
    created_at      TIMESTAMP NOT NULL DEFAULT NOW(),
    UNIQUE (source_type, source_message_id)
);
CREATE INDEX idx_ext_signals_source_channel ON external_signal_messages(source_type, channel_id);
CREATE INDEX idx_ext_signals_received_at    ON external_signal_messages(received_at);
```

7.5 Komponenten

Datei	Rolle	LoC
`trading/signals/discord_bot.py` (NEU)	Discord-Channel-Listener, async	~200
`trading/signals/external_message_persistence.py` (NEU)	INSERT in `external_signal_messages`, source-agnostisch	~80
`trading/news/telegram_reader.py` (EDIT)	zusätzlicher Hook: jede Telegram-Channel-Nachricht via Persistence-Layer in DB schreiben (Doppel-Persistenz, parallel zur existierenden Signal-Logik)	~30
`trading/signals/__init__.py` (NEU)	Public API	~20
`gui/database/migrations/..._create_external_signal_messages.php` (NEU)	Schema migration	~40
`gui/app/Models/ExternalSignalMessage.php` (NEU)	Eloquent	~50
`gui/app/Filament/Resources/ExternalSignalMessageResource.php` (NEU)	Filament-Listing mit source_type-Filter	~220

7.6 Aufwand

~18 h (2.5-3 Tage), inkl. Discord-API + Telegram-Hook + Migration + GUI-Resource.

7.7 Boundaries

- **Operator-Entscheidung Auth**: Discord-Bot ist ein Account (gut: read-only kann begrenzt werden via Bot-Permissions)
- **Secret-Surface**: DISCORD_BOT_TOKEN → operator setzt, lese-only Permissions, nicht im Chat anzeigen
- 0× Mainnet

- Bot-Restart via SOT-1d für env-Update

7.8 Operator-Decisions vor Start

1. Discord-Server-Liste: welche Server sollen monitort werden?
2. Bot-Account-Anlage: Operator-Aktion (Discord Developer Portal)
3. Channel-IDs: Operator pflegt die Whitelist (`DISCORD_CHANNELS_WHITELIST`)
4. Telegram-Channels: welche der heute schon in `signal_channels.py` definierten Channels sollen für AI-Auswertung mit-erfasst werden?
5. Doppel-Persistenz für Telegram (alte Pipeline UND neue Tabelle) ok? → ja: kein Risiko alte Logik zu brechen, später konsolidieren
6. Reporter-Notification: pro RAW-Nachricht oder nur nach AI-Bewertung?

---

8. Phase 6 — T1-AI-SIGNAL-ANALYSIS-1 (unified)

8.1 Ziel

Jede neue **externe Signal-Nachricht** — egal ob aus Discord oder Telegram-Channels — wird durch einen LLM-Call analysiert. Der Worker arbeitet **source-agnostisch** auf der `external_signal_messages`-Tabelle, muss also nicht zwischen Discord und Telegram unterscheiden.

Output-Schema:

```
{
  "is_trading_signal": true,
  "signal_type": "buy" | "sell" | "tip" | "fud" | "spam" | "unknown",
  "symbols_mentioned": ["BTC/USDT", "SOL/USDT"],
  "confidence": 0.0,        // 0.0 - 1.0
  "scam_likelihood": 0.0,   // 0.0 - 1.0
  "summary_de": "Trader empfiehlt BTC-Long bei 78k mit SL 76k, TP 82k",
  "raw_signals_extracted": {
    "entry": 78000.0,
    "stop_loss": 76000.0,
    "take_profit": 82000.0
  },
  "source_quality_estimate": 0.5
}
```

8.2 LLM-Provider-Auswahl

Heute bereits in env vorhanden (laut SEC-LEAK-Inventar von heute morgen):

Provider	Stärken	Schwächen
Anthropic Claude (Sonnet 4.x)	Sehr gut für strukturierte Analyse, lange Kontexte, deutsche Sprache	API-Kosten ~$0.003 / 1k tokens (in)
OpenAI GPT-5 (operator-präferiert?)	Verbreitet, JSON-mode, schnell	dito Kosten
Google Gemini Flash	sehr günstig	weniger zuverlässig für strukturierten Output

Empfehlung: parallel mock-fähig, Provider via env-Flag wählbar. Für v1 reicht ein Provider; ich empfehle Claude (Anthropic-Key bereits in env, gute deutsche Sprache).

8.3 Architektur (source-agnostic)

```
external_signal_messages Row (Discord OR Telegram)
        │
        ▼
ai_signal_worker (NEU, source-agnostisch)
   - claim row where ai_analysis_id IS NULL
   - LLM-Call (Anthropic Claude API)
   - JSON-Parse + Validation
   - INSERT external_signal_analysis
   - UPDATE external_signal_messages.ai_analysis_id
        │
        ▼
   if is_trading_signal AND confidence > 0.7 AND scam_likelihood < 0.3
      AND symbol IN binance_universe:
        bot main loop → integrate as T1 signal candidate
        source=`discord_ai` oder `telegram_ai` je nach source_type
        Reporter → Telegram alert
```

8.4 Komponenten

Datei	Rolle	LoC
`trading/signals/ai_analyzer.py` (NEU)	LLM-Call + JSON-Validation	~250
`trading/signals/ai_worker.py` (NEU)	Async-Worker (analog command_worker), source-agnostisch	~150
Neue Tabelle `external_signal_analysis`	siehe Schema unten	—
`gui/app/Models/ExternalSignalAnalysis.php` (NEU)	Eloquent	~50
Filament-Resource Analyse-Ansicht mit source_type-Filter	analog External-Resource	~220

```
CREATE TABLE external_signal_analysis (
    id                BIGSERIAL PRIMARY KEY,
    message_id        BIGINT REFERENCES external_signal_messages(id) NOT NULL,
    llm_provider      VARCHAR(20) NOT NULL,
    llm_model         VARCHAR(50) NOT NULL,
    is_trading_signal BOOLEAN NOT NULL,
    signal_type       VARCHAR(20),
    symbols_mentioned JSON,
    confidence        NUMERIC(4,3),
    scam_likelihood   NUMERIC(4,3),
    summary_de        TEXT,
    raw_signals_extracted JSON,
    source_quality_estimate NUMERIC(4,3),
    llm_raw_response  TEXT,                  -- Audit / Replay
    tokens_used       INTEGER,
    cost_usd          NUMERIC(10,6),
    duration_ms       INTEGER,
    error_message     TEXT,
    analyzed_at       TIMESTAMP NOT NULL DEFAULT NOW()
);
```

8.5 Wiring in T1

In `main.run_scan_cycle`: zusätzlich zu den 8 bestehenden Quellen, lese auch `external_signal_analysis JOIN external_signal_messages` rows mit: - `is_trading_signal=true` - `confidence > 0.7` - `scam_likelihood < 0.3` - `analyzed_at` innerhalb der letzten 60min - `symbol IN binance_universe`

→ Diese Signale werden in `STRATEGY_GROUP_T1` emittiert mit `source` = `discord_ai` (wenn source_type=discord) bzw. `telegram_ai` (wenn source_type=telegram_channel/telegram_bot).

→ Operator sieht in DecisionLog-Filter "Source" sauber getrennt: `discord_ai`, `telegram_ai`, `run_scan_cycle` (legacy), etc.

8.6 Cost-Management

- Per Nachricht ~500 input tokens + 300 output tokens = ~800 tokens
- Bei Anthropic Sonnet ≈ $0.003 input + $0.005 output ≈ **$0.008 / Nachricht**
- Bei 100 Nachrichten/Tag → ~$0.80/Tag, ~$24/Monat
- **Rate-Limit + Cap**: max 200 Analysen/Tag, danach skip + WARN

8.7 Tests

Test	Was
`test_ai_extracts_buy_signal_from_german_text`	"Long BTC bei 78k, SL 76k, TP 82k" → buy + symbols+levels
`test_ai_classifies_spam_high_likelihood`	"🚀🚀🚀 100x guaranteed!!" → scam_likelihood>0.8
`test_ai_handles_no_signal_text`	"Wie geht's heute Markt?" → is_trading_signal=false
`test_ai_caches_idempotently`	Same discord_id → re-use existing analysis row
`test_ai_fail_does_not_crash_signal_intake`	LLM-API down → row gets error_message, signal stays unanalyzed
`test_ai_respects_daily_cap`	201ste Analyse pro Tag → skipped
`test_bot_consumes_high_confidence_signals`	confidence=0.8 → reaches run_scan_cycle
`test_bot_skips_low_confidence_signals`	confidence=0.5 → no signal candidate

8.8 Aufwand

~18 h (2.5 Tage), inkl. Provider-Integration, Tests, Cost-Tracking.

8.9 Boundaries

- LLM-Provider-Key NICHT im Chat anzeigen (Lesson Learned heute)
- Per-Day-Cost-Cap als Env (`AI_DAILY_COST_USD_CAP=5.00`)
- Telegram-Alert wenn Cap erreicht
- AI-Output ist **nur Signal**, keine direkte Order-Trigger ohne T1-Risk-Manager-Check
- 0× Mainnet (T1 bleibt Binance Testnet)

8.10 Operator-Decisions vor Start

1. Provider: Anthropic Claude, OpenAI GPT-5, oder Google Gemini? (Empfehlung: Claude wegen deutscher Sprache + bereits vorhandener Key)
2. Daily-Cost-Cap in USD?
3. Confidence-Threshold für T1-Signal-Übernahme (Default: 0.70)
4. Scam-Likelihood-Threshold (Default: 0.30)

---

9. Phase 7 — T2-SOLANA-SHADOW-1 PLAN (Plan-Only, Code in W3+)

9.1 Ziel

Plan-PDF für T2-Solana-Greenfield. Anforderungen aus deinem Briefing: - T2 läuft auf Solana-Chain (echtes Mainnet, kein Devnet existiert für Pump.fun) - Pump-and-Dump-Charakter mit eigenem Risk-Profil - **Shadow-Phase zuerst**: T2-Module läuft, schreibt nur in DB, führt KEINE Orders aus - Erst nach Operator-explizitem GO + Wallet-Funding wird live

9.2 Inhalt des Plan-PDFs (Lieferung Ende W2)

- Wallet-Architektur (dedicated SOL-Wallet, Hot-/Cold-Separation)
- Jupiter SDK Integration (Quote+Swap-Flow)
- T2-Risk-Profile (1.5×ATR SL, 6h time-stop, kein DCA)
- Symbol-Universe (Pump.fun Bonding Curve >85% + Raydium-Pairs mit min Volume)
- Shadow-Phase-Konstante als 4-stufiges Gate (signals→decision→quote-only→live)
- Operator-Boundaries für Mainnet-Aktivierung
- Cost-Budget für Test-Phase
- Phasen-Zeitplan für Implementation (W3-W6 vermutlich)

9.3 Aufwand

~4 h Plan-Schreiben + PDF-Render.

---

10. Tagesplan-Tabelle

Tag	Phase	Aktivitäten
W1 Mo (19.05.)	1 (TIER-ARCH)	Recon + tier_arch.py + Wiring in main.py
W1 Di (20.05.)	1 → 2	Tier-Arch Tests + Cutover; Start Symbol-Gate
W1 Mi (21.05.)	2 (SYMBOL-GATE)	is_tradable_strict + Wiring + Tests
W1 Do (22.05.)	3 (EXEC-MODE-3)	Backward-Compat-Renames Phase 3a + Tests + Cutover
W1 Fr (23.05.)	4 (PDF)	Refactor-vs-Rewrite Plan PDF erstellen
W1 Sa (24.05.)	Review	Operator review + 24h Live-Observation
W1 So (25.05.)	Buffer	Rest / Bugfixes wenn nötig
W2 Mo (26.05.)	3b + 5	Phase 3b (alte Paper-Namen löschen) + Discord+Telegram-Recon
W2 Di (27.05.)	5 (DISCORD+TG)	DB-Migration unified + discord_bot.py + Telegram-Hook
W2 Mi (28.05.)	5 → 6	Tests + Cutover; Start AI-Analyzer (source-agnostisch)
W2 Do (29.05.)	6 (AI)	LLM-Provider-Integration + Worker + Tests
W2 Fr (30.05.)	6 (AI)	AI-Wiring in run_scan_cycle (discord_ai + telegram_ai sources) + Cutover
W2 Sa (31.05.)	7 (T2-PLAN)	T2-Solana-Shadow-1 Plan-PDF
W2 So (01.06.)	Review	Operator review + alle Phasen Live-Beobachtung

---

11. Risiken & Mitigationen

Risiko	Wahrscheinlichkeit	Impact	Mitigation
Discord API rate-limit	mittel	mittel	Bot-Permissions read-only + Backoff
LLM-Cost-Spike	niedrig	niedrig	Daily-Cap-Env + Alert
LLM-Hallucination (fake Symbol)	mittel	mittel	Binance-Universe-Cross-Check vor Signal-Übernahme
LLM-Provider-Outage	niedrig	niedrig	error_message in DB, Signal bleibt unanalyzed
DB-Migration-Lock	niedrig	niedrig	Migration testen in Staging-Container vor Live
Tier-Architektur-Vertrag bricht alte Tests	mittel	niedrig	Backward-Compat-Phase, alle Tests durchlaufen pre-cutover
Scope-Creep durch Operator-Wünsche	hoch	mittel	strict Phasen-Grenzen, separate Operator-GOs

12. Boundaries (durchgehend)

- **0× Mainnet** (sowohl Binance als auch Solana)
- **0× MS-DRY_RUN=false** Aktivierung
- **0× T2-Solana-Live** in diesen 2 Wochen (nur Plan)
- **0× T3-Copy-Trading** Aktivierung
- **0× Strategie-Parameter-Änderung** ohne explizites Operator-GO
- **0× DB-Mass-Mutation** (nur INSERT in neue Tabellen, keine UPDATE auf alte)
- **0× Secrets im Chat anzeigen** (Lesson Learned heute)
- **0× CommandBus-V6/MH-1** Aktivierung

13. Stop-Gates pro Phase

Phase	Stop-Gate vor Cutover	Stop-Gate post-Cutover
1 TIER-ARCH	alle 7 Tests grün	24h Beobachtung 0 Tracebacks
2 SYMBOL-GATE	alle 6 Tests grün; reject-Logs in DB	24h: kein non-binance Symbol mehr in trade_logs
3 EXEC-MODE-3	3a alle Tests grün; alte Imports parallel	24h: kein --paper-Watchdog-Spawn-Log
4 PDF	Operator-Review	—
5 DISCORD	DB-Migration testbar; Channel-Whitelist gesetzt	24h: erste Nachrichten in DB; keine 429
6 AI	LLM-Provider-Test grün; Daily-Cap-Logik testbar	24h: erste Analysen in DB; Cost < Cap
7 T2-PLAN	Operator-Review	—

14. Erwartetes Resultat nach 2 Wochen

Komponente	Stand
T1 Binance Testnet	konsolidiert, Symbol-Gate aktiv, Multi-Source (Discord + Telegram-Channels), AI-bewertete Signale
Tier-Architektur	im Code-Vertrag dokumentiert, T1=active, T2=plan_only, T3=deactivated
PaperTrader-Legacy	retired, sauber als BaseTrader umbenannt
GUI	+ ExternalSignalMessage Resource (unified) + ExternalSignalAnalysis Resource (mit source_type-Filter)
Postgres	+ 2 neue Tabellen (external_signal_messages, external_signal_analysis) — unified für Discord + Telegram + zukünftige Quellen
Operator-Workflows	Discord-Channel-Whitelist via env, Telegram-Channels über `signal_channels.py`, LLM-Provider via env
Refactor-Plan-PDF	erstellt, Operator-Entscheidungsgrundlage für W3+
T2-Plan-PDF	erstellt, Implementation-Ready für separate Phase
T2-Code	bleibt 0 LoC (Plan-only)

15. Operator-Entscheidungen vor Start (Sammelliste)

1. **Phase 1**: T2-Status `PLAN_ONLY` oder `DEACTIVATED`?
2. **Phase 1**: Naming `tier_arch.py` ok?

3. **Phase 3**: Backward-Compat-Aliase über 24h beobachten vor Phase 3b?
4. **Phase 5**: Welche Discord-Server / Channel-IDs?
5. **Phase 5**: Discord-Bot-Account anlegen (Operator-Action)
6. **Phase 5**: Welche Telegram-Channels in AI-Pipeline mit-erfassen? (Liste aus `signal_channels.py`)
7. **Phase 5**: Doppel-Persistenz Telegram (alte Pipeline + neue Tabelle) ok?
8. **Phase 5**: Reporter-Notification pro RAW-Nachricht oder nur AI-Hits?
9. **Phase 6**: LLM-Provider — Claude (Empfehlung), GPT, Gemini?
10. **Phase 6**: Daily-Cost-Cap in USD? (Empfehlung: $5/Tag)
11. **Phase 6**: Confidence-Threshold? (Empfehlung: 0.70)
12. **Phase 6**: Scam-Likelihood-Threshold? (Empfehlung: 0.30)

---

16. STOP

Plan-Dokument. Keine Implementation.

Operator-Entscheidung erforderlich für jede einzelne Phase mit explizitem **GO <Phase-Name>**-Prompt. Boundaries hart, Sicherheitsgates konservativ.

Empfohlener nächster Schritt: Operator gibt **GO TIER-ARCH-CONTRACT-1** mit Bestätigung der 3 Decisions aus Sektion 3.7, dann starte ich Mo 19.05.