Steve-TradingBot — Full-Stack-Architektur

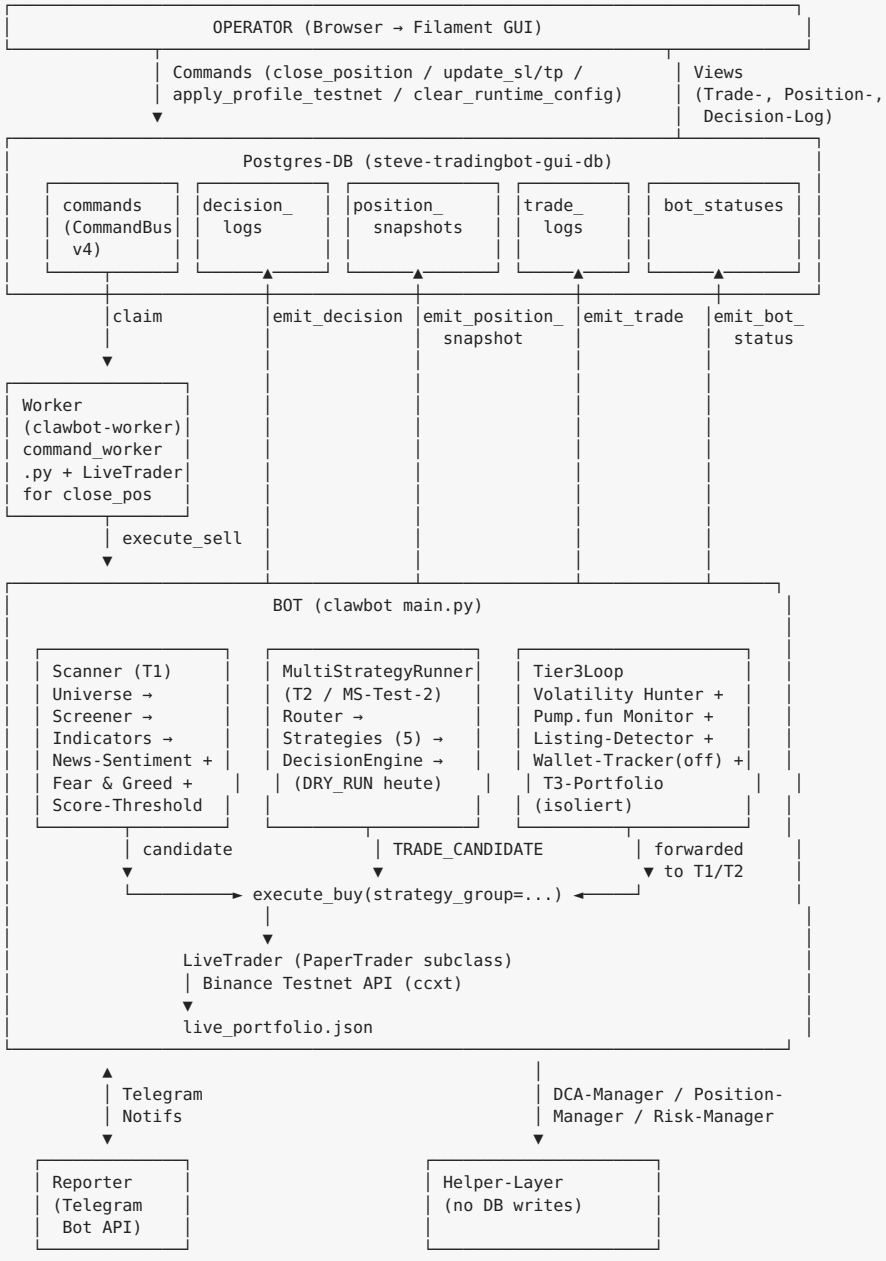
Stand: **2026-05-16 13:10 UTC** · Branch `master` · HEAD `7e5cf6e` · Read-only Architektur-Bericht aller drei Tiers + ihrer Zusammenhänge

0. Executive Summary

Der Bot besteht aus **drei parallel laufenden Trading-Tiers**, die unterschiedliche Risiko-Profile bedienen, aber denselben Execution-Backend (Binance Testnet via `LiveTrader`) und denselben Postgres-Audit-Trail teilen.

Tier	strategy_group	Pipeline	Live-Status
T1 Standard	`t1_core`	Legacy Scan-Loop in `main.run_scan_cycle` — alle 120 s; Multi-Source-Scoring + Fear & Greed + News-Sentiment + Regime → Threshold-Check → Binance Testnet Spot-BUY	aktiv (Default-Pipeline)
T2 Pump & Dump	`t2_pump_dump`	(a) T3-forwarded Signals (Large-Cap Pump-Detect) → Binance Testnet via T1-Risk-Layer (b) Multi-Strategy-Runner DRY-RUN — 5 Strategien (trend_follow / breakout / mean_reversion / vwap / volatility_sweep) — heute MS-ACTIVATION-TEST-2 dry-run aktiv	(a) aktiv passiv, (b) Dry-Run-Beobachtung
T3 (Subsystem)	`t3_copy_trading` (DB-Tag, geplant)	Solana-Pump.fun-Monitor + Multi-Source-Listing-Detector. Heute Daten-/Signal-Layer (kein echtes Solana-Trading). Forwarded Signale an T1/T2 mit `strategy_group=t2_pump_dump`	aktiv als Signal-Quelle (Helius-Failover seit 12:19 UTC)

1. System-Übersicht



2. Tier 1 — Legacy Multi-Source Scan-Loop

2.1 Funktion

Klassischer 120-Sekunden Scan-Cycle. Pro Iteration: scannt das Binance-USDT-Universe, holt News-Sentiment + Fear & Greed + Market-Regime, kombiniert in Score, vergleicht gegen dynamisch berechneten Threshold, kauft passende Kandidaten als Binance-Testnet Spot-Orders. Verwaltet Position-Lifecycle (SL / TP / DCA / Trailing / 60h-Breakeven / 96h-Time-Stop).

2.2 Code-Flow

```
main.py:main() loop:
  ↓
  scanner.universe.UniverseScanner.get_candidates()
    ↓ (volume + spread filter)
  scanner.indicators.IndicatorCalculator  + scanner.regime.MarketRegime
    ↓ (ATR, EMA, MACD, BB, RSI, regime classification)
  scanner.screener  → 50-150 Kandidaten/Cycle mit raw score
    ↓
  news.fetcher + news.analyzer (NewsFetcher + SentimentAnalyzer)
  fear_greed_index (CoinGecko / Alternative.me)
  ↓
  for each candidate:
    combined_score = score×0.7 + news_sentiment×0.2 + fear_greed×0.1
    apply: market-hours modifier, fg_bias gate, cooldown, quiet_market,
           volatility (ATR/price > 5%), N7.2 stable-/peg-asset block
    ↓
    if combined_score >= threshold AND action_ok:
      → emit_decision(action='buy', strategy_group='t1_core', ...)
      → live_trader.execute_buy(
            symbol, price, quantity,
            stop_loss, take_profit,
            strategy_group='t1_core',
            decision_id=<generated>,
            strategy_id=None)
    else:
      → emit_decision(action='reject', reject_reason=..., metadata=...)

  live_trader.update_prices(prices, dca_mgr=...)  →  closed_by_sl_tp list
    ↓
  for each closed trade:
    → emit_trade(source='live_trader.update_prices', ...)
    → emit_position_snapshot(status='closed', ...)
    → reporter.send_trade_closed_alert(...)

  emit_bot_status(...)
  sleep(120s) — except every 10s: tier3.check_exits_only()
```

2.3 Komponenten

Modul	Klasse	Rolle
`scanner/data_fetcher.py`	`DataFetcher`	Binance / Mainnet OHLCV via ccxt
`scanner/indicators.py`	`IndicatorCalculator`	EMA / MACD / RSI / BB / ATR auf OHLCV-DataFrame
`scanner/regime.py`	`MarketRegime`	Klassifiziert Markt-Regime (BULL / BEAR / CHOP / WEAK / STRONG_TREND etc.) + Stabilitäts-Filter
`scanner/screener.py`	`Screener`	Score-Berechnung pro Symbol (Score 0-10)
`news/fetcher.py`	`NewsFetcher`	Multi-Source-News (CryptoCompare / NewsAPI / Reddit)
`news/analyzer.py`	`SentimentAnalyzer`	Sentiment-Score aus News-Liste
`execution/risk_manager.py`	`RiskManager`	Position-Sizing aus ATR-Stop-Distance
`execution/position_sizer.py`	`PositionSizer`	Cash- + Max-Position-Cap
`execution/dca_manager.py`	`DCAManager`	DCA-Rescue Logik vor SL (Nachkauf statt Verkauf)
`execution/position_manager.py`	`PositionManager`	SL-Adjustment (Trailing, Breakeven, News, 96h-Hard-Exit) — schreibt `pos['_sl_kind']` für LABEL-1-Klassifikation
`execution/safety_check.py`	`SafetyChecker`	N7.2 Stable-/Peg-Asset-Block + andere Sicherheitsgates
`reports/reporter.py`	`Reporter`	Telegram-Notifs (Erstkauf / Pyramid / DCA-Rescue / Trade-Closed / Daily-Report)

2.4 Schreibt nach

- `decision_logs` — 7 Call-Sites pro Cycle (evaluating + 6 reject/buy-Pfade)
- `trade_logs` — bei jedem Close
- `position_snapshots` — bei jedem Close (status=closed) + periodisch (status=open) im Bot-Loop
- `bot_statuses` — 1× pro Cycle (Heartbeat)
- `live_portfolio.json` — Filesystem-State (LiveTrader-_save_state)

3. Tier 2 — Multi-Strategy Runner

3.1 Funktion

Alternative Pipeline (Phase N7), die parallel zum T1-Scanner läuft **nur wenn** `MULTI_STRATEGY_ENABLED=true`. Pro Symbol holt ein Router basierend auf Regime die "aktiven" Strategien aus einer Registry von 5 spezialisierten Strategien. Jede Strategie produziert einen score; eine `FinalDecisionEngine` kombiniert die Werte und entscheidet TRADE_CANDIDATE oder REJECT.

Heute aktiv als **MS-ACTIVATION-TEST-2 Dry-Run** (seit 11:49 UTC). DRY_RUN=true → MS-Runner evaluiert, schreibt JSONL nach

`logs/multi_strategy.log`, aber löst KEINE Binance-Orders aus.

3.2 Code-Flow

```
main.py:run_scan_cycle  (wenn MULTI_STRATEGY_ENABLED=true):
  ↓
  MultiStrategyRunner.run(symbols, data_by_symbol)
    ↓ pro Symbol:
    MarketRegime.detect_extended(df, symbol)
    StrategyRouter.route(regime)  →  active_strategies
    ↓ pro Strategy-ID in active_strategies:
    Strategy.score_signal(df, regime)  →  {strategy_id, signal, score, entry, sl, tp}
    FinalDecisionEngine.evaluate(strategy_result, regime, router_result, ...)
      ↓
      DECISION_TRADE_CANDIDATE  oder  DECISION_REJECT
    ↓ wenn TRADE_CANDIDATE AND is_execution_allowed():
    _attempt_execution(decision, df, regime):
      Safety-Gates (N7.2 / DCA-Block / doppel-pos / N8.1 balance / sizer)
      ↓ alle pass:
      decision_log_id = generate_decision_id()       ← DATA-LINK-1-FU2
      emit_decision(action='buy',
                    decision_id=decision_log_id,
                    strategy_id=decision['strategy_id'],
                    strategy_group='t1_core',           ← T-SPLIT-2 origin
                    mode='testnet' if BINANCE_TESTNET else 'live')   ← EXEC-MODE-LABEL-2
      ↓
      LiveTrader.execute_buy(..., decision_id=..., strategy_id=...,
                             strategy_group='t1_core', allow_add=False)
    ↓ alle entscheidungen:
    _log_decision(...)  →  JSONL nach multi_strategy.log (eigene Forensik)
```

3.3 Strategie-Registry

strategy_id	Klasse	Regime-Match
`trend_follow`	`TrendFollowStrategy`	STRONG_TREND (bullish bias)
`breakout`	`BreakoutStrategy`	STRONG_TREND + niedrige Volatility
`mean_reversion`	`MeanReversionStrategy`	WEAK_TREND, RSI extrem
`vwap_mean_reversion`	`VWAPStrategy`	RANGE / WEAK_TREND
`volatility_sweep`	`VolatilitySweepStrategy`	HIGH_VOLATILITY + base_regime != CHOP

3.4 Schreibt nach

- `multi_strategy.log` (JSONL) — eigene Forensik mit allen Roh-Werten
- `decision_logs` — nur bei TRADE_CANDIDATE + Execution-Erfolg (DATA-LINK-1-FU2 seit heute; `source='multi_strategy_runner'`; Reject-/Evaluating-Pfade noch nicht emittet → Backlog *MS-DECISION-OBSERVABILITY-1*)
- `trade_logs` — via gemeinsamen LiveTrader.execute_buy-Pfad
- `position_snapshots` — analog

3.5 Hard-Gates

- `MULTI_STRATEGY_ENABLED=true`
- `MULTI_STRATEGY_DRY_RUN=false` (heute true — kein Order-Trigger)
- `BINANCE_TESTNET=true` (Mainnet-Cutover NICHT in N7)
- `LIVE_TRADING_ENABLED=true`
- Decision = TRADE_CANDIDATE
- Symbol tradable (LiveTrader._is_tradable)
- Bestehende Safety-Checks bestanden
- Position für Symbol nicht bereits offen
- DCA / Pyramid im MS-Modus explizit aus

4. Tier 3 — Volatility Hunter (Solana Pump.fun)

4.1 Funktion

Spezialisiertes Subsystem für die Suche nach Solana-Memecoin-Listings und Pump.fun-Token-Launches. Heute reines **Daten-/Signal-Layer** — führt keine echten Solana-Orders aus. Wirkt als Multi-Source-Signal-Aggregator mit Forwarding-Brücke zu T1/T2.

Siehe separates Dokument *tier3-architecture-20260516T123000Z.pdf* für tiefe Details (Modul-Tabelle, Datenfluss, Risiko-Bewertung, mögliche pumpportal.fun-Integration).

4.2 Externe Quellen (9 aktiv, 2 deaktiviert)

Quelle	Methode	Status
Pump.fun Bonding-Curve / Graduation	WebSocket via Helius RPC `logsSubscribe` auf Pump.fun-Programm	aktiv (HELIUS-KEY-FAILOVER seit 12:19 UTC)
Binance neue Listings	Public REST API	aktiv
Binance Announcements	HTTP-Scraping	aktiv
CoinGecko Trending	Free Public API, 10-min Cache	aktiv
Birdeye new Solana tokens	Free tier	aktiv
DexScreener boosts	Free Public API	aktiv
Reddit r/CryptoMoonShots RSS	15-min Cache, max confidence 0.45	aktiv
CryptoPanic Hot	Free tier	aktiv
Wallet-Tracker (Whale-Buys)	Solscan API	deaktiviert (`COPY_TRADING_ENABLED=false`)
Twitter / X Trending	—	deaktiviert (API-Kosten)

4.3 Market-Cap-Gate (Brücke zu T1/T2)

Wenn das Signal-Symbol entweder in `T3_BLACKLIST` ist (BTC, ETH, SOL, BNB, ...) oder eine Market-Cap > 25 Mrd USD hat (CoinGecko-Lookup), klassifiziert T3 den Coin als "Large-Cap" und gibt das Signal an T1/T2 weiter, statt es in das eigene isolierte T3-Portfolio zu schreiben.

Forwarding-Mechanismus: `Tier3Loop.forwarded_signals`-Liste (pro Cycle reset). `main.py:_process_t3_forwarded_signals` iteriert die Liste und ruft für jedes Signal `live_trader.execute_buy(strategy_group='t2_pump_dump', ...)`.

5. Cross-Cutting-Subsysteme (alle Tiers nutzen)

5.1 Execution-Layer

Klasse	Zweck	Wer nutzt
`LiveTrader`	Subclass von `PaperTrader`, switcht STATE_FILE auf `live_portfolio.json` + wired ccxt-Binance-Testnet-Exchange + handelt echte Testnet-Orders	T1, T2-MS-Runner, T2-Forwards, Command-Worker (CLOSE-TESTNET-3)
`PaperTrader`	State-Management-Basis: positions / closed_trades / cash / SL-Cooldowns. `execute_buy` / `execute_sell` / `update_prices`. *Seit heute mit STATE-RELOAD-Guard (HISTORY-1 Fix B) → erkennt externe Mutationen via mtime-cookie*	Parent-Klasse
`DcaManager`	DCA-Rescue (Nachkauf statt SL-Verkauf wenn Gap < 3% unter SL)	via LiveTrader.update_prices()
`PositionManager`	SL-Adjustment (Trailing / Breakeven / News-Tightened / 96h-Hard-Exit). Schreibt LABEL-1-Forensik-Tags (`_sl_kind`, `_tp_kind`) auf pos-Dict	T1-Loop + LiveTrader
`RiskManager`	Position-Sizing aus ATR-Stop-Distance + Max-Open-Positions-Cap	T1, T2-Forwards
`MultiStrategyPositionSizer`	N7/N8: eigenständiger Sizer für MS-Runner mit eigener Tier-Logik	T2-MS-Runner
`SafetyChecker`	N7.2 Stable-Coin- und Peg-Asset-Block	T1, T2-MS-Runner
`CircuitBreaker`	B-OUTAGE-RESILIENCE: Buy-Pause während Exchange-Outages	LiveTrader
`BalanceGuard`	N8.1: Hard-fail VOR Sizer wenn Exchange-Balance nicht verfügbar oder zu klein	T2-MS-Runner

5.2 State & Persistence

Surface	Wo	Wer schreibt / liest
`live_portfolio.json`	File `logs/live_portfolio.json`	**Schreibt**: Bot's LiveTrader (T1, T2-Forward, T2-MS-Runner), Worker's LiveTrader (CLOSE-TESTNET-3). **Liest**: gleich. *Bot reload-guard seit HISTORY-1 Fix B*
`paper_portfolio.json`	File	Legacy seit Phase L; LiveTrader-Init switcht STATE_FILE weg → wird heute nicht mehr geschrieben/gelesen außer durch reine PaperTrader-Tests
`tier3_portfolio.json`	File	T3-Subsystem isoliertes Portfolio (eigene Cash-Quote 20% des Bot-Kapitals); heute nicht real-trading-aktiv
`commands`	Postgres	GUI / Operator → Worker
`decision_logs`	Postgres	Bot (7 emit-sites in main.py + 1 in MS-Runner since FU2)
`position_snapshots`	Postgres	Bot (open + closed) + Worker (closed since HISTORY-1 Fix A)
`trade_logs`	Postgres	Bot (close-flow in main) + Worker (HISTORY-1 Fix A)
`bot_statuses`	Postgres	Bot heartbeat 1× pro Cycle
`multi_strategy.log`	JSONL file	MS-Runner Forensik (auch wenn Dry-Run)
`trading.log`	logfile, rotiert	Globale stdout/stderr-Sammel
`baseline_holdings.json`	File (optional)	RECON-2.2: pre-bot-existing Holdings; "frozen" für Bot-Sicht

5.3 CommandBus / Worker

Asynchroner Operator-Channel. GUI fügt eine Row in `commands`-Tabelle ein; der `clawbot-worker` Container claimed sie via Postgres `SELECT … FOR UPDATE`-Lock und führt die Operation aus.

- **Aktiv (v4)**: noop, close_position (paper+testnet seit CLOSE-TESTNET-1), update_stop_loss, update_take_profit, cancel_command, apply_profile_testnet, clear_runtime_config
- **Worker-Handler-vorhanden aber Registry nicht aktiviert**: apply_baseline_holdings, clear_baseline_holdings (v5/v6-pending)
- **v6-Repo aber Worker-Handler fehlt** (8 managed_proposal-Commands)

Worker nutzt seit CLOSE-TESTNET-3 (commit `3288b09`) den **LiveTrader** für close_position (vorher PaperTrader → leerer State). Seit HISTORY-1 (commit `7baccfb`) emittiert er auch `trade_logs` + `position_snapshots` post-close.

5.4 GUI (Filament Admin)

Read-mostly Operator-Frontend (Laravel 13 + Filament 5).

- Auth via TOTP/MFA
- Resources: TradeLog, PositionSnapshot, DecisionLog, Command, ConfigProfile, ManagedHolding (gepflegt aber nicht aktiv)
- Widgets: FreshOpenPositionsWidget, LatestExitSubtypeWidget, BotMonitoringSnapshot
- Dashboard: MonitoringDashboard (Recent Closed Trades / Bot Logs / Telemetrie)
- Trade-Chart: Lightweight-Charts v5 mit OHLCV-Cache + DCA / SL / TP Overlays
- **Display-Layer**: ModeLabels (paper → TESTNET LIVE), TradeLog::labelForExitReason (manual_close → Manuell geschlossen, fixed_stop_loss → Stop Loss (Fix), ...) + Color-Buckets info/success/danger/warning
- **Action-Layer**: ViewPosition page mit Close-at-Market + Update-SL/TP-Aktionen → POST in `commands`-Tabelle. Idempotency-Key `close_position:<pid>` mit Retry-Logik nach Terminal-State (GUI-CLOSE-POSITION-COMMAND-CREATE-1)

5.5 Reporter / Telegram

Operator-Notifications (deutsch) via Telegram Bot API. Methoden:

- send_erstkauf_alert / send_pyramid_alert / send_rebuy_alert
- send_dca_rescue_alert (heute mit HTML-escape seit `59489a5`)
- send_trade_closed_alert
- send_sl_tp_adjustment_alert
- send_daily_report
- send_outage_*_alert (B-OUTAGE-RESILIENCE-1 / 7)

6. Zusammenhänge zwischen den Tiers

6.1 T3 → T2 Forward-Brücke

Der einzige direkte Übergang zwischen Tiers. T3 detektiert einen Large-Cap Pump (z.B. SOL/USDT via Pump.fun-Graduation + CoinGecko-Trending-Convergence). Statt eines T3-Solana-Buys (nicht implementiert) gibt T3 das Signal weiter. Der Bot ruft `execute_buy(strategy_group='t2_pump_dump')` auf der Binance Testnet — also derselbe Execution-Pfad wie T1, aber mit T2-Tag in DB. Operator kann später per `strategy_group`-Filter in der GUI die T2-Trades von den T1-Trades unterscheiden.

6.2 T1 vs T2-MS-Runner (parallel)

Heute koexistieren beide Pipelines im selben Bot-Loop. Wenn `MULTI_STRATEGY_ENABLED=true`: T1 Legacy-Scanner UND T2-MS-Runner laufen pro Cycle. Beide nutzen denselben `universe` + `data_by_symbol`, aber unterschiedliche Score-Pipelines.

Beide pushen Signale an denselben `LiveTrader` → potenzielle Doppel-Buys vermieden via `execute_buy(allow_add=False)`-Gate (MS-Runner setzt es immer false; T1 nutzt true für DCA/Pyramid).

6.3 Shared State-Konflikt-Risiko

Risiko	Mitigation
T1 + T2-MS-Runner kaufen gleiches Symbol gleichzeitig	execute_buy prüft `positions[symbol]` + `allow_add=False` blockt → effektiv erste-kommt-erste-mahlt
Bot überschreibt Worker-Close auf live_portfolio.json	**HISTORY-1 Fix B**: Bot's LiveTrader stat()ed STATE_FILE vor update_prices, reloaded bei mtime-Bump
T3-Forward triggert Position die T1 schon hält	execute_buy doppel-pos-check; T3-Forward bekommt None-Return + log
Worker close_position auf Position die Bot grade tradet	kein File-Lock heute; Race-Window klein durch reload-Guard + Worker-LiveTrader-Pfad

6.4 Operator-Sicht in der GUI

Alle drei Tiers schreiben in dieselben Postgres-Tabellen. T-SPLIT-2 hat `strategy_group` als VARCHAR-Column ergänzt (T-SPLIT-1 Schema). Operator kann pro Resource filtern:

- TradeLogResource → Filter `strategy_group` mit Optionen "T1 Standard" / "T2 Pump & Dump" / "T3 Copy Trading" / "Legacy / Unknown" (StrategyGroupLabels-Service)
- PositionSnapshotResource → analog
- DecisionLogResource → analog mit zusätzlich `strategy_id`-Filter (Multi-Strategy)

7. Tagesveränderungen (Cross-Tier)

Seit gestern Abend 16 Commits, alle Tiers betroffen:

Commit	Affected Tier(s)	Kurzbeschreibung
`2baada3`	T1	DECISION-LOG-METADATA-EMIT — 7 emit_decision sites in main.py mit metadata_json (candidate_reason, news, FG)
`f290031`	T3-Forward	T3-T1-FORWARD-1 NameError fix (sig vs signal)
`59489a5`	Cross	NOTIFIER-DCA-RESCUE-HTML-ESCAPE — DCA-Rescue Telegram-400 fix
`27e9eee`	CommandBus	CLOSE-TESTNET-1 — close_position erlaubt für testnet env
`4eaf327`	GUI	MODE-DISPLAY-1 — paper → "TESTNET LIVE" in Position/TradeLog views
`32c95c7`	T2-MS-Runner	DATA-LINK-1-FU2 — MS emittiert decision_logs + propagiert decision_id/strategy_id
`ed9096d`	CommandBus	CLOSE-TESTNET-2 — Worker sys.path Shim + namespaced import
`ee491e9`	Repo-Hygiene	BACKLOG.md In-Repo Pin
`41fd5bf`	GUI	GUI-CLOSE-POSITION-COMMAND-CREATE-1 — terminal-state retry mit `:attempt:<uuid>`
`3288b09`	CommandBus	CLOSE-TESTNET-3 — Worker switch zu LiveTrader, mainnet defense-in-depth
`a58f9a1`	T1 + T2 + Watchdog	EXEC-MODE-LABEL-2 + WATCHDOG-PAPER-FLAG-CLEANUP — _G61_MODE truth fix
`7baccfb`	CommandBus + State	HISTORY-1 — Worker emit_trade post-close + Bot state-reload-guard
`395db06`	GUI	GUI-MANUAL-CLOSE-LABEL-1 — manual_close → "Manuell geschlossen"
`7e5cf6e`	T3-Pump.fun-Monitor	HELIUS-KEY-FAILOVER — automatische Rotation bei 429

8. Live-State (Snapshot 2026-05-16 13:10 UTC)

Aspekt	Wert
T1-Open-Positions	BABY, ENA, ORCA, SUI (+1000CHEEMS heute 13:01 geschlossen via fixed_stop_loss)
Cash	~9041 USDT (post-BABY-DCA)
T2-MS-Runner	Dry-Run aktiv seit 11:49 UTC; 24+ Cycles, 0 TRADE_CANDIDATE / 0 REJECT (Stability-Filter blockt alle Regimes)
T3-Pump.fun-Monitor	verbunden seit 12:19 UTC (Helius primary key '06dcec99-...' frische 1M credits)
Tracebacks seit Cutover 12:18 UTC	0
Heute closed trades	2 KITE/USDT (08:06 trailing_stop_profit + 10:09 manual_close via CommandBus), 1 1000CHEEMS/USDT (13:01 fixed_stop_loss)
DCA-Rescues heute	1000CHEEMS 12:04, BABY 13:08 — beide ohne Telegram-400 (HTML-escape funktioniert)

9. Verbleibende offene Punkte

- **COMMAND-BUS-V6 "(fix)"** — Operator-Klärung welche Interpretation (4 Optionen)
- **SEC-OUTPUT-LEAK-ROTATION** — 6 echte Secrets + 2 Helius-Keys gehören rotiert (heute morgen + jetzt im Chat)
- **MS-DECISION-OBSERVABILITY-1** — MS-Runner reject/evaluating in Postgres
- **MS-NOTIFIER-1** — Telegram für MS-Trades
- **pumpportal.fun-Integration** — Tier3 als parallele Source (Plan in *tier3-architecture-20260516T123000Z.pdf*)
- **EXEC-MODE-LABEL-3** — Full-Retire von `--paper` / `PaperTrader` / `paper_portfolio.json`
- **BOT-WORKER-STATE-LOCK** — heute Mitigation via mtime-Reload; tieferer Fix wäre Datei-Lock oder Bus-vermittelte Mutation
- **DCA-RISK-1** — Operator-Pause
- **GUI-DESIGN-1b-FIX-CONTRACT** — TelemetryStatus / CurrentRegime / EffectiveCap Datenvertrag

10. Nicht Teil dieses Bots

- Echte Solana-Execution (Jupiter-Swap, Raydium-Swap)
- Mainnet-Binance-Trading (BINANCE_TESTNET ist die Boundary)
- Wallet-Tracker / Copy-Trading (COPY_TRADING_ENABLED=false)
- Managed-Proposal-Subsystem (MH-1, v6 nicht aktiviert)
- Backtest-Framework — heute kein dediziertes Backtest-Modul

Erstellt read-only auf Basis von: `trading/main.py` (1700+ LoC), `trading/strategies/multi_strategy_runner.py`, `trading/strategies/{base,trend_follow,breakout,mean_reversion,vwap_mean_reversion,volatility_sweep,decision_engine,router,registry}.py`, `trading/execution/{paper,live}_trade.py`, `trading/execution/{dca,position,risk}_manager.py`, `trading/execution/multi_strategy_position_sizer.py`, `trading/scanner/` + `trading/news/` + `trading/reports/`, `trading/tier3/`, `trading/command_worker.py`, `gui/app/Filament/Resources/` + `gui/app/Services/`, BACKLOG.md, MEMORY.md, git log. Keine Code-Mutation, keine Container-Aktion für diese Analyse.