GUI-TRADE-CHART-1a — Plan-Review

Datum: 2026-05-15 14:19 UTC | Repo: `Steve-TradingBot` | master HEAD: `fa7e450` (GUI-TRADE-EXPORT-1-FU1) | Modus: **EXCHANGE_TESTNET** | Scope: **READ-ONLY** | Implementation: **NO-GO (Vorphasen)**

Executive Summary

Ziel: Trade-Detail-Tab mit OHLCV-Candles, Entry/Exit-Markers, SL/TP-Step-Lines, Trailing-Path und Decision-Attribution — Operator-Self-Review-Tool, kein Trading-Interface.

Architektur-Entscheidung: 5 Phasen über ~4 Wochen. **GUI-TRADE-CHART-1b** (Basic-Chart) ist **NICHT** sofort umsetzbar — beide Voraussetzungen **OHLCV-CACHE-1** + **TRAIL-EVENT-LOG-1** brauchen DB-Migration und müssen vor 1b laufen.

Operator-Decisions Q1-Q8 alle bestätigt → verbindlicher Implementation-Pfad. **Migration-Impact: 2 neue Tabellen** (`ohlcv_cache` + `trade_events`) plus 1 Frontend-Dep (`lightweight-charts` npm).

0. API/Tool-Abhängigkeiten — Operator-Input erforderlich?

> **Kurzantwort: NEIN — keine externen API-Keys oder Subscriptions vom Operator nötig.** Ich beschaffe alle benötigten Tools/Libraries selbst.

Komponente	Quelle	API-Key nötig?	Wer beschafft
OHLCV-Daten (Candles)	Binance Public Klines API `/api/v3/klines`	**NEIN** (public endpoint)	Backend nutzt bestehende `ccxt` -Library (im Bot bereits aktiv für scanner/regime.py)
Chart-Library	TradingView `lightweight-charts` (Apache-2.0)	NEIN	`npm install lightweight-charts` im GUI-Build, kostenlos, kein Account
Rate-Limit Binance public	1200 weight/min, klines=1 weight	—	56 Trades × ~1 fetch = unkritisch. Cache vermeidet repeated fetches
TradingView Account/Sub	—	NEIN	Lightweight Charts ist standalone — keine Verbindung zu TradingView-Servern
BINANCE_API_KEY/SECRET	bereits in `.env` für Testnet-Order-Execution	nicht für Chart	Existiert schon, wird NICHT für OHLCV-Cache verwendet (public endpoint)

> **Einzige Operator-Decision (optional)**: Mainnet-Candles vs. Testnet-Candles für Chart.
> - **Mainnet public klines**: echte Preise, höhere Liquidität, semantisch realistisch (Bot trade auf Testnet, aber Preise sind ähnlich)
> - **Testnet public klines**: matched 1:1 zu Bot-Orders, aber Testnet-Preise sind oft thin/synthetic
> **Empfehlung**: Mainnet-Klines (default Binance ccxt-config), weil sie die *echten* Market-Bewegungen zeigen, die zum Trade führten. Operator-Override möglich.

1. Live-State Preflight

Item	Wert
master HEAD	`fa7e450` (GUI-TRADE-EXPORT-1-FU1)
Bot host-PID	1228771 healthy
BINANCE_TESTNET	true
LABEL-1 / DATA-LINK-1 / DATA-CLEANUP-1 / SOT-1d / GUI-DESIGN-1a	alle ✓ **live**
Frontend Stack	Filament 5, Tailwind 4, Vite 6, Laravel 13
Chart-Library	**nicht installiert**
`routes/web.php`	nur `redirect('/', '/admin')` — eigener Route-Slot frei
OHLCV-Cache-Tabelle	**existiert nicht**
trade_events-Tabelle	**existiert nicht**
`logs/sl_adjustments.json`	**1.5 MB, 75 Symbole**, Format `{time, kind, old_sl, new_sl, sl_changed, old_tp, new_tp, tp_changed, reason}` — Legacy-Bridge für Backfill

2. Architektur — Current vs Target

```
CURRENT
trade_logs (56 rows) ──┐
decision_logs ─────────┤ (DATA-LINK-1: decision_id JOIN für neue Trades)
position_snapshots ────┤
sl_adjustments.json ───┘ ← JSON, 1.5MB, symbol-scoped, NICHT in DB

OHLCV ──────── ccxt fetch_ohlcv (ephemer, nicht persistiert)
               trading/scanner/universe.py
               trading/scanner/regime.py
```

```
TARGET
trade_logs (existing) ──┐
decision_logs (existing) ┤
trade_events (NEW) ──────┤── TradeChartController
ohlcv_cache (NEW) ───────┘    GET /admin/trades/{id}/chart-payload
                                        ↓
                                JSON-Payload
                                        ↓
                                Lightweight Charts (Vite-bundle)
```

3. Operator-Decisions Q1–Q8 (verbindlich)

ID	Entscheidung	Implikation
Q1	OHLCV: **DB-backed** `ohlcv_cache`	Postgres-Migration nötig, joinbar mit trade_logs
Q2	**Dedicated JSON-Route** `/admin/trades/{id}/chart-payload` hinter Filament/Auth	`Route::get` in web.php, eigener Controller, Filament-Middleware-Stack
Q3	`trade_logs.id` für closed, **+** `position_id` für open Positions	Controller akzeptiert beide ID-Typen, Resolver entscheidet
Q4	**Neue** `trade_events` **-Tabelle**	Migration nötig, strukturiertes Event-Stream-Model
Q5	**Sub-Marker** unter einem Trade, kein parent_trade_id jetzt	DCA/Pyramid-Adds = `event_type='dca_add'` / `'pyramid_add'` in trade_events, gleicher trade_id
Q6	**"Replay limited"-Banner** für Legacy, Tab nutzbar	Controller liefert `payload.replay_limitations[]`, Frontend rendert Banner
Q7	**Adaptive Timeframe** (1m/5m/15m/1h je Hold-Time)	Backend wählt timeframe basierend auf `closed_at - opened_at`
Q8	**Gemeinsame OHLCV/Event-Basis** für Chart UND Backtest	Wiederverwendbarer Service-Layer (`OHLCVCacheService` + `TradeEventService`)

4. Daten-Verträge

4.1 `ohlcv_cache` — NEU (OHLCV-CACHE-1 Migration)

```
CREATE TABLE ohlcv_cache (
    id              BIGSERIAL PRIMARY KEY,
    symbol          VARCHAR(50)   NOT NULL,
    timeframe       VARCHAR(10)   NOT NULL,    -- '1m'|'5m'|'15m'|'1h'|'1d'
    open_time       BIGINT        NOT NULL,    -- UNIX ms epoch (Binance-kline)
    close_time      BIGINT        NOT NULL,
    open            NUMERIC(20,8) NOT NULL,
    high            NUMERIC(20,8) NOT NULL,
    low             NUMERIC(20,8) NOT NULL,
    close           NUMERIC(20,8) NOT NULL,
    volume          NUMERIC(24,8) NOT NULL,
    fetched_at      TIMESTAMPTZ   NOT NULL DEFAULT NOW(),
    UNIQUE (symbol, timeframe, open_time)
);
CREATE INDEX ohlcv_cache_symbol_tf_time ON ohlcv_cache (symbol, timeframe, open_time DESC);
```

4.2 `trade_events` — NEU (TRAIL-EVENT-LOG-1 Migration)

```
CREATE TABLE trade_events (
    id              BIGSERIAL PRIMARY KEY,
    trade_id        VARCHAR(255),              -- nullable (open positions noch ohne trade_id)
    position_id     VARCHAR(255),              -- T-SPLIT-2/N8.2 stabile ID
    symbol          VARCHAR(50)   NOT NULL,
    event_time      TIMESTAMP(6) WITH TIME ZONE NOT NULL,
    event_type      VARCHAR(50)   NOT NULL,
    price           NUMERIC(18,8),
    qty             NUMERIC(24,12),
    old_sl, new_sl  NUMERIC(18,8),
    old_tp, new_tp  NUMERIC(18,8),
    sl_kind         VARCHAR(50),               -- LABEL-1 subtype
    tp_kind         VARCHAR(50),
    pnl             NUMERIC(18,6),
    reason          VARCHAR(255),
    metadata        JSONB,
    created_at      TIMESTAMPTZ NOT NULL DEFAULT NOW()
);
CREATE INDEX trade_events_trade_id     ON trade_events (trade_id);
CREATE INDEX trade_events_position_id  ON trade_events (position_id);
CREATE INDEX trade_events_symbol_time  ON trade_events (symbol, event_time DESC);
```

4.3 Event-Types & Hook-Stellen im Bot

event_type	Bot-Quelle
entry	paper_trade.py:execute_buy (allow_add=False)
exit	paper_trade.py:execute_sell · inkl. LABEL-1 exit_reason
stop_loss_set	paper_trade.py:execute_buy init
take_profit_set	paper_trade.py:execute_buy init
trailing_update	position_manager.py:apply_trailing()
break_even_update	position_manager.py:apply_breakeven() 60h
dca_add	execute_buy(allow_add=True) DCA-Branch
pyramid_add	execute_buy(allow_add=True) Pyramid-Branch
news_tightening	position_manager.py:apply_news_adjustment() BEARISH

4.4 Legacy-Backfill (optional)

`logs/sl_adjustments.json` enthält bereits 75 Symbole × n Events im Format `{time, kind, old_sl, new_sl, sl_changed, old_tp, new_tp, tp_changed, reason}`. `kind` ist eines von `trailing` / `breakeven` / `news` / `hard_exit`. Eine einmalige Backfill-Migration könnte historische Adjustments retroaktiv in `trade_events` schreiben — aber symbol-scoped, Mapping zu `trade_id` ist heuristisch (time-range JOIN).
Empfehlung: optional, niedrige Priorität.

4.5 Chart-Payload-Format

```
{
  "trade":   { "id", "symbol", "entry_price", "exit_price", "opened_at",
               "closed_at", "realized_pnl", "exit_reason", "decision_id" },
  "decision":{ "score", "regime", "source", "strategy_id" },
  "candles": [{ "time", "open", "high", "low", "close", "volume" }],
  "events":  [{ "type", "time", "price", "old_sl", "new_sl", "reason" }],
  "replay_limitations": [ "no_trail_events_pre_TRAIL_EVENT_LOG", "decision_link_missing" ],
  "meta": { "timeframe", "candles_from", "candles_to", "is_legacy" }
}
```

5. Phasen-Plan

Phase	Inhalt	Aufwand	DB-Mig.	Risiko
GUI-TRADE-CHART-1a	Plan-Review (dieser)	DONE	nein	—
OHLCV-CACHE-1	`ohlcv_cache`-Tabelle + Migration + `OHLCVCacheService` + Binance public-klines fetch + adaptive-tf-Resolver	~8–12h	**ja**	Medium
TRAIL-EVENT-LOG-1	`trade_events`-Tabelle + Migration + Hook-Insert in paper_trade/live_trade/position_manager (additiv, kein behaviour change)	~10–15h	**ja**	Medium
TRAIL-EVENT-LOG-1-BACKFILL (optional)	`sl_adjustments.json` retroaktiv mapping zu trade_id	~3–5h	nein	niedrig
GUI-TRADE-CHART-1b	TradeChartController + JSON-Route + lightweight-charts npm + Vite-bundle + Basic Chart (Candles + entry/exit + Summary)	~12–15h	nein	Low-Medium
GUI-TRADE-CHART-1c	Advanced Overlays — SL/TP-Step-Lines + trailing path + DCA/Pyramid-Marker + Timeline-Tab	~6–10h	nein	Medium
GUI-TRADE-CHART-1d	Decision/Risk-Panel (regime/score/source/strategy_id aus DATA-LINK-1)	~4–6h	nein	Low

Gesamt: ~43–63 h Code + 2 Migrationen.

6. Frontend-Integration

- `npm install lightweight-charts` (~50 KB minified, Apache-2.0) → `resources/js/trade-chart.js` als isoliertes Bundle
- `vite.config.js` Input erweitern: `'resources/js/trade-chart.js'`
- AdminPanelProvider unverändert — Route registriert via `routes/web.php`
- Filament-Blade-View bekommt `<x-stb.trade-chart :trade-id="$record->id" />` Component, lädt JSON via fetch() vom Endpoint, mountet Chart in `<div>`-Container
- CSP-konform: kein inline-Script, nur `data-*`-Attribute auf Container

7. Backend-Pattern

```
// routes/web.php
Route::middleware(['auth', 'verified'])
    ->get('/admin/trades/{trade}/chart-payload', [TradeChartController::class, 'show']);

App\Services\TradeChart\
├── TradeChartPayloadBuilder      // composer: trade + decision + candles + events
├── OHLCVCacheService             // fetch + insert + read (Binance public klines via ccxt)
├── TradeEventQuery               // filter + join trade_id/position_id
└── AdaptiveTimeframeResolver     // Q7: hold-time → timeframe
```

Authorization: `TradeChartController` ruft `TradeLogResource::canAccess()` (Admin-only, GUI-TRADE-EXPORT-1). 403 wenn nicht Admin.

8. Bot-Code-Hooks (TRAIL-EVENT-LOG-1)

File:Stelle	Hook	events
`paper_trade.py:execute_buy` nach _new_pos	INSERT entry + stop_loss_set + take_profit_set	3
`paper_trade.py:execute_sell` nach trade_record	INSERT exit	1
`paper_trade.py:update_prices` DCA-Rescue	INSERT dca_add	1
`position_manager.py:apply_trailing`	INSERT trailing_update	1
`position_manager.py:apply_breakeven` 60h	INSERT break_even_update	1
`position_manager.py:apply_breakeven` 96h-Hard-Exit	INSERT trailing_update reason=hard_exit_96h	1
`position_manager.py:apply_news_adjustment`	INSERT news_tightening	1

Pattern: DB-Write via `db_emitter.py` Pattern — neue Function `emit_trade_event()` analog `emit_trade` / `emit_decision`. **Additiv**, kein Verhalten geändert.

9. Adaptive Timeframe (Q7)

Hold-Time	Timeframe	~Candles
< 1h	1m	~60
1–6h	5m	~72
6–24h	15m	~96
24–96h (Hard-Exit)	1h	~96
> 96h	1h	cap 200

Service: `AdaptiveTimeframeResolver::resolve($openedAt, $closedAt): string`.

10. Read-only-Boundaries

Boundary	Status
Kein Live-Order vom Chart aus	✓ nur GET
Kein Strategy-Param-Change	✓
Keine historische DB-Mutation	✓ insert-only Tables
Keine synthetischen Trades	✓
Kein Mainnet (Order)	✓ BINANCE_TESTNET=true gepinnt
Keine Secrets in Logs/Reports	✓ AuditMetadataScrubber re-use
Keine Mockdaten	✓ Empty-State → "replay_limited"-Banner
Kein TradingView-Buy/Sell-Action	✓ Lightweight Charts hat keine Actions

11. Akzeptanzkriterien

Bereich	Kriterium
Basic chart	Candles um opened_at±buffer, entry/exit markers
Price labels	Exakte Werte aus trade_logs (nicht candle-approx)
Exit label	LABEL-1 Subtype
Decision link	regime/score/source IF decision_id; sonst "legacy/no decision link"
Trailing	Step-Lines IF events; sonst "trailing history unavailable"
OHLCV	Cached, nicht repeatedly fetched
Mobile	Horizontal-Scroll-Container, breaks GUI nicht
Security	Keine Secrets/env/trade-actions/Mainnet

12. Risiko-Matrix

Risiko	Schwere	Mitigation
2 neue DB-Tabellen (Migration)	mittel	OHLCV-CACHE-1 + TRAIL-EVENT-LOG-1 jeweils eigener Migration-Plan-Review vor Code
Bot-Code-Touch in TRAIL-EVENT-LOG-1 (7 Insert-Hooks)	**HOCH**	Strikt additiv, kein behaviour change, Tests pro Hook, Build via SOT-1d-Pfad
OHLCV-Backfill für legacy trades	niedrig	Optional, kann nach 1b/1c
Lightweight-Charts JS-Library-Vertrauen	niedrig	tradingview/lightweight-charts: open source, Apache-2.0, >9k stars, vom TradingView selbst
Adaptive Timeframe bei sehr kurzen Trades (<5 min)	niedrig	Default 1m, falls Hold <5min: cap candles
Open Positions ohne exit-Event	niedrig	Frontend rendert "Position open" overlay statt exit-marker
Decision-Link für pre-DATA-LINK-1-Trades	niedrig	"Replay limited"-Banner + Tab grayed-out für decision panel
Mobile-Chart-UX	mittel	Lightweight Charts native-touch-support + stb-* mobile defenders aus FU3
Backtest gemeinsame Datenbasis (Q8)	niedrig	Service-Layer-Pattern abstrahiert Datenquelle

13. GO / NO-GO

Phase	Status
GUI-TRADE-CHART-1a Plan-Review	**DONE**
GUI-TRADE-CHART-1b Implementation	**NO-GO sofort** — 2 Vorphasen nötig

14. Empfohlene Sequenz

Reihenfolge	Phase	Voraussetzung	Aufwand
1	DATA-LINK-1-MONITOR Wartephase (passive, läuft schon)	—	24–72 h
2	LABEL-1-MONITOR Wartephase (passive, läuft schon)	—	3–5 d
3	**OHLCV-CACHE-1 Plan-Review** (Migration + Service)	—	~1 h
4	OHLCV-CACHE-1 Impl	Plan-GO	~8–12 h
5	**TRAIL-EVENT-LOG-1 Plan-Review** (Migration + Bot-Hooks)	—	~1 h
6	TRAIL-EVENT-LOG-1 Impl + Backfill (optional)	Plan-GO	~13–20 h
7	GUI-TRADE-CHART-1b Impl (Basic Chart)	OHLCV + TRAIL-EVENT live	~12–15 h
8	GUI-TRADE-CHART-1c Impl (Advanced Overlays)	1b live	~6–10 h
9	GUI-TRADE-CHART-1d Impl (Decision/Risk)	1c live + DATA-LINK-1 stabil	~4–6 h

15. Boundaries (Plan-Phase eingehalten)

Boundary	Status
Kein Code	✓
Keine DB-Migration	✓
Keine Bot-/Worker-Restart	✓
Kein Mainnet	✓
Kein Push	✓ master `fa7e450` unverändert
Keine Secrets	✓

STOP — Operator-Decision für nächste Phase (siehe Sequenz oben):

- **A**: GO OHLCV-CACHE-1 Plan-Review als nächstes
- **B**: GO TRAIL-EVENT-LOG-1 Plan-Review zuerst (kein OHLCV-Dependency)
- **C**: erst 24–72 h DATA-LINK-1-MONITOR + LABEL-1-MONITOR abwarten, dann A