# OHLCV-CACHE-1 — Plan-Review

**Datum:** 2026-05-15 17:18 UTC **Status:** Plan-Review, read-only — 0× Code, 0× Migration, 0× Restart **Master HEAD:** `fa7e450`
**Auftrag:** Read-only Audit + Architektur-Plan für persistenten OHLCV-Cache als Datenbasis für GUI-TRADE-CHART-1b/c/d **Mainnet-Relevanz:** sehr gering (read-only Public-Klines, keine API-Keys, keine Orders, BINANCE_TESTNET=true unverändert)

---

## Executive Summary

Der Bot liest OHLCV-Daten **bereits heute** über `ccxt.binance` aus dem **Mainnet (anonym, Public-Endpoints)** — das ist genau die Quelle, die wir für die Chart-Anzeige im GUI brauchen. Es gibt jedoch **keine Persistenz**: jeder Scanner-Lauf, jeder regime-Check, jeder backtest fetcht live. Für `GUI-TRADE-CHART-1b/c/d` (Trade-Detail-Replay mit OHLCV + Entry/Exit-Marker + Trailing-Replay) brauchen wir:

1. **Persistenten Cache** ( `ohlcv_cache` Tabelle) — sonst hängt jede Detail-Page von Binance-Latenz/Verfügbarkeit ab und brennt unnötige Weight bei jeder Operator-Navigation.
2. **Adaptiven Timeframe-Resolver** — kurze Scalps brauchen 1m, mehrtägige Swing-Trades 1h/1d. Median-Trade-Duration nicht ermittelbar (alle 56 trade_logs pre-DATA-LINK-1 haben `opened_at=NULL` ).
3. **Public-Klines-Client direkt in PHP/Laravel** — Bot-Python und GUI-PHP sind getrennte Stacks; ein gemeinsamer Cache in der `tradingbot_gui` DB ist der saubere Mittelweg.
4. **JSON-Route hinter Admin-Auth** — `/admin/trades/{id}/chart-payload` als Adapter zwischen Cache und TradingView-Lightweight-Charts (Operator-Decision schon getroffen).

**Empfehlung: GO als separate Implementations-Phase OHLCV-CACHE-1-IMPL (~4-6h Code + ~2h Tests + 1× GUI-Recreate).** Keine Bot-Code-Änderung, kein Worker-Touch, kein docker cp, keine Mainnet-Order-Surface.

Die Implementation kann **parallel zu SEC-1c-3c-Stabilitätsfenster** laufen (Konflikt-frei: nur neue Tabelle, kein Bot-Code, kein Auth-Flow-Touch, keine Migrator-Schema-Änderung an bestehenden Tabellen).

---

## Phase 1 — Live-State Snapshot

| Komponente | Status |
|---|---|
| Bot main.py | host-PID 1228771, ALIVE |
| Worker | host-PID 3430424 |
| GUI | host-PID 3431767, `/admin/login` HTTP 200 |
| BINANCE_TESTNET | `true` |
| Master HEAD | `fa7e450` (GUI-TRADE-EXPORT-1-FU1) |
| Bot-Code Volume-Mount | SOT-1d (clawbot-bot-data) |
| decision_logs latest | 2026-05-15 15:08:08 UTC |
| position_snapshots latest | 2026-05-15 15:10:18 UTC |
| trade_logs total | 56 |
| trade_logs latest closed | 2026-05-15 01:13:15 UTC (keine neuen seit DATA-LINK-1 11:33 Cutover) |
| Tracebacks seit 11:33 | 0 |
| OHLCV-Fehler in trading.log | 0 (10 INFO-Lines erfolgreich) |
| DB-Größe `tradingbot_gui` | 71 MB |
| Bot-Logs-Größe | 195 MB |

System gesund. Sicher um Plan-Review zu fahren.

---

## Phase 2 — OHLCV Source Recon

### 2.1 Call-Sites von `fetch_ohlcv`

| Datei | Zeile | Timeframe | Limit | Zweck |
|---|---|---|---|---|
| `scanner/universe.py` | 77 | `1h` | 2 | "letzter geschlossener Bar"-Filter (Universe-Heartbeat) |

| | | | | |
|---|---|---|---|---|
| `scanner/regime.py` | 71 | `1d` | (default 365) | BTC/USDT Regime-Detektion |
| `scanner/regime.py` | 231 | dyn | dyn | Multi-Symbol Regime-Probe |
| `scanner/screener.py` | 143 | dyn | dyn | Per-Symbol Indikatoren |
| `scanner/data_fetcher.py` | 192/309 | dyn | dyn | Core-Fetcher + CoinGecko-Helper |
| `proposal_engine.py` | 358 | `1d` | 30 | Proposal-Validierung |
| `backtest/backtester.py` | 32 | dyn | dyn | Backtest |

→ Zentralisiert über `DataFetcher.fetch_ohlcv()` (außer `universe.py:77` und `proposal_engine.py:358`, die direkt ccxt aufrufen).

### 2.2 Exchange-Konfig

```
# data_fetcher.py:_get_exchange('binance')
params = {'enableRateLimit': True}
params['options'] = {'defaultType': 'spot'}
# KEINE Keys an Mainnet-Instance (Testnet-Keys auf Mainnet -2008 Invalid Api-Key ID)
# Public OHLCV/Ticker/load_markets brauchen keine Auth.
inst = ccxt.binance(params)
```

→ **Mainnet anonym, Public-Endpoints only.** Genau die Quelle, die OHLCV-CACHE-1 nutzen wird. Keine Auth-Surface, keine Order-Surface.

Testnet-Instance existiert separat ( `binance_testnet` , `set_sandbox_mode(True)` ), aber **nur für Order-Routing + Balance** — nicht für OHLCV (Begründung im Code: "Testnet-Preise sind unbrauchbar fuer Analyse").

### 2.3 Symbol-Format

| Layer | Format | Beispiel |
|---|---|---|
| Bot/DB (trade_logs.symbol) | ccxt | `BTC/USDT` , `SOL/USDT` , `WLFI/USDT` |
| ccxt internal | ccxt | `BTC/USDT` |
| ccxt market.id (raw) | Binance | `BTCUSDT` |
| Binance API | Binance | `BTCUSDT` |

**Konsequenz für PHP-Client:** Mapping ist trivial: `str_replace('/', '', $ccxtSymbol)` . Aber: **nur 40 distinct Symbole im aktuellen trade_logs-Universum** — eine Whitelist-Validierung gegen `exchangeInfo` ist zusätzlich sinnvoll (kein blindes Pass-through unbekannter Strings).

### 2.4 Bestehende Cache-Mechanik

- Keine lokalen OHLCV-Dateien ( `find -name '*ohlcv*' -o -name '*klines*' -o -name '*candles*'` → 0 Treffer)
- Kein in-memory cache erkennbar
- Jeder Scanner-Lauf, jede regime-Check, jeder backtest fetcht live → **bei häufiger Operator-Chart-Navigation würde GUI das Mainnet-Weight-Budget belasten**

### 2.5 Rate-Limit / Retry

- ccxt `enableRateLimit=True` regelt Bot-side (sliding window)
- `trading/execution/exchange_errors.py` kategorisiert `RateLimitExceeded` / `DDoSProtection`
- Bot scheinbar problemlos auf Mainnet-Public (0 Errors in den letzten Bot-Cycles)
- GUI hat **keinen** ccxt → eigener Client + eigener Rate-Limit-Wrapper (Phase 5)

### 2.6 Aktuelle Fehler

- `grep -E "OHLCV|fetch_ohlcv|429|rate.limit"` in `trading.log` → **0 Fehler**, nur INFO-Lines "Binance Mainnet Modus (anonym)" alle ~2.5min (Scanner-Tick)

---

## Phase 3 — Source Decision: Mainnet vs Testnet

**Entscheidung (Operator-pinned):** Binance Mainnet Public Klines.

| Aspekt | Mainnet Public | Testnet |
|---|---|---|
| API-Key nötig | NEIN | NEIN für Klines |
| Order-Surface | NEIN (nur GET) | NEIN bei Public-Endpoints |
| Datenqualität | echt, alle Symbole | dünne Bücher, künstliche Spreads, viele Symbole |

| | | |
|---|---|---|
| | | fehlen |
| Bot nutzt heute | ✅ Ja (data_fetcher.py:107-117) | ❌ Nur für Order-Routing |
| Rate-Limit | 6000 Weight/min/IP | 6000 Weight/min/IP |
| Auth-Schicht | nicht erforderlich | nicht erforderlich |
| Symbol-Coverage | ~2500 Spot-Pairs | ~150 Spot-Pairs (oft veraltet) |
| Mainnet-Boundary-Risiko | **read-only, keine API-Keys, kein Order-Pfad** | n/a |

**Endpoint-Spec:** - `GET https://api.binance.com/api/v3/klines` - Params: `symbol`, `interval` (1m/5m/15m/1h/4h/1d), `startTime` (ms), `endTime` (ms), `limit` (1..1000) - Weight: 2 per request - IP-Limit: 6000 Weight/min → ~3000 Anfragen/min worst case - Antwort: `[openTime, open, high, low, close, volume, closeTime, quoteVolume, trades, takerBuyBase, takerBuyQuote, ignore]` (Array of 12-Tupel)

**Mainnet-Boundary-Bestätigung:** GET-only, anonym, 0 Auth-Surface, 0 Order-Surface, 0 API-Key-Exfiltrationsrisiko. BINANCE_TESTNET=true bleibt unverändert (betrifft Bot's Order-Routing, nicht GUI-Read-Only).

---

## Phase 4 — Tabellen-Design `ohlcv_cache`

### 4.1 Schema-Vorschlag

```
CREATE TABLE ohlcv_cache (
    id           BIGSERIAL PRIMARY KEY,
    symbol       VARCHAR(32)  NOT NULL,                  -- ccxt-format z.B. 'BTC/USDT'
    timeframe    VARCHAR(8)   NOT NULL,                  -- 1m | 5m | 15m | 1h | 4h | 1d
    open_time    TIMESTAMPTZ  NOT NULL,                  -- UTC bar-open
    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(28,8) NOT NULL,
    source       VARCHAR(32)  NOT NULL DEFAULT 'binance_mainnet',
    fetched_at   TIMESTAMPTZ  NOT NULL DEFAULT NOW(),
    CONSTRAINT ohlcv_cache_unique UNIQUE (symbol, timeframe, open_time)
);

CREATE INDEX ohlcv_cache_lookup_idx
    ON ohlcv_cache (symbol, timeframe, open_time);

-- Optional, für zukünftige Maintenance-Jobs
CREATE INDEX ohlcv_cache_fetched_at_idx
    ON ohlcv_cache (fetched_at);
```

### 4.2 Sanity-Checks

- `tradingbot_gui_migrator` hat `CREATE` auf public-schema ( `has_schema_privilege` = `t` )
- `tradingbot_gui_app` hat **kein** `CREATE` ( `f` ) → Lockdown intakt
- Tabelle existiert noch nicht (0 Treffer für `ILIKE '%ohlcv%'` )
- SEC-1c-3c Befund: Migrator kann CREATE TABLE, aber kein ALTER auf existing → **neue Tabelle, kein ALTER, sauberer Pfad**

### 4.3 Why NUMERIC(20,8)?

Binance-Klines liefern bis zu 8 Nachkommastellen (z.B. SHIB ~0.00001234). `NUMERIC(20,8)` deckt Coins bis ~10^12 Preis pro Coin bei 8 Decimals — passt für sämtliches Spot-Universum.

### 4.4 Wachstumsabschätzung

Worst-Case-Annahme: 40 Symbole × 6 Timeframes × adaptiv aktivste Auflösung

| Timeframe | Bars/Tag/Symbol | 30d Retention, 40 Symbole |
|---|---|---|
| 1m | 1440 | ~1.7M |
| 5m | 288 | ~345k |
| 15m | 96 | ~115k |
| 1h | 24 | ~28k |
| 4h | 6 | ~7k |
| 1d | 1 | ~1.2k |

**Gesamt 30d**: ~2.2M Rows bei Worst-Case 6-TF-Coverage für alle 40 Symbole = **~250 MB** (Row inkl. Indizes ~110 Bytes).

Realistisch werden wir **nur die TFs cachen, die wir für tatsächlich offene/geschlossene Trades brauchen** (adaptive_timeframe). Erwartung: <50 MB.

→ Akzeptabel auf 71 MB DB-Total-Size.

### 4.5 Retention

Empfehlung: **180-Tage-Retention** via einfachen Maintenance-Job ( `DELETE FROM ohlcv_cache WHERE fetched_at < NOW() - INTERVAL '180 days' AND open_time < NOW() - INTERVAL '180 days'` ). Kein Auto-Job in OHLCV-CACHE-1-IMPL — als P2-Backlog **OHLCV-CACHE-1-RETENTION** notieren.

---

## Phase 5 — Fetch / Upsert-Strategie

### 5.1 Adaptive Timeframe Resolver

Da alle bestehenden trade_logs `opened_at=NULL` haben, brauchen wir **2 Quellen** für die Trade-Dauer:

1. **Primär:** `trade_logs.opened_at` und `trade_logs.closed_at`
2. **Fallback** (für pre-DATA-LINK-1 trade_logs): parse `trade_id` Format `SYMBOL-ISO8601` (z.B. `ASTER/USDT-2026-05-15T01:13:14.469978+00:00` ) → ungefähre Open-Zeit. Wenn das fehlschlägt: Default-Fenster `[closed_at - 6h, closed_at + 30min]` .

```
function resolveTimeframe(Carbon $open, Carbon $close): string {
    $hours = $close->diffInSeconds($open) / 3600.0;
    return match (true) {
        $hours < 1   => '1m',
        $hours < 6   => '5m',
        $hours < 24  => '15m',
        $hours < 168 => '1h',
        default      => '1d',
    };
}
```

### 5.2 Window-Berechnung

```
window_start = open_time  - max(20 bars × tf, 30min)
window_end   = close_time + max(10 bars × tf, 30min)
```

Vorlauf gibt visuelle Pre-Entry-Kontext, Nachlauf zeigt Post-Exit-Verlauf.

### 5.3 Cache-Coverage-Check

```
SELECT MIN(open_time), MAX(open_time), COUNT(*)
FROM ohlcv_cache
WHERE symbol = :sym AND timeframe = :tf
  AND open_time BETWEEN :start AND :end;
```

Cache-Hit wenn: - `COUNT(*) = expected_bar_count(tf, start, end)` , ODER - `(MAX - MIN) / tf_seconds + 1 ≈ COUNT(*)` (toleriere ≤2 Bars Lücke für laufenden Bar)

Sonst → Lücken-Berechnung + Refill via BinanceKlineClient.

### 5.4 Upsert-Strategie

```
INSERT INTO ohlcv_cache (symbol, timeframe, open_time, open, high, low, close, volume, source)
VALUES (...)
ON CONFLICT (symbol, timeframe, open_time)
DO UPDATE SET
    high       = EXCLUDED.high,
    low        = EXCLUDED.low,
    close      = EXCLUDED.close,
    volume     = EXCLUDED.volume,
    fetched_at = NOW();
```

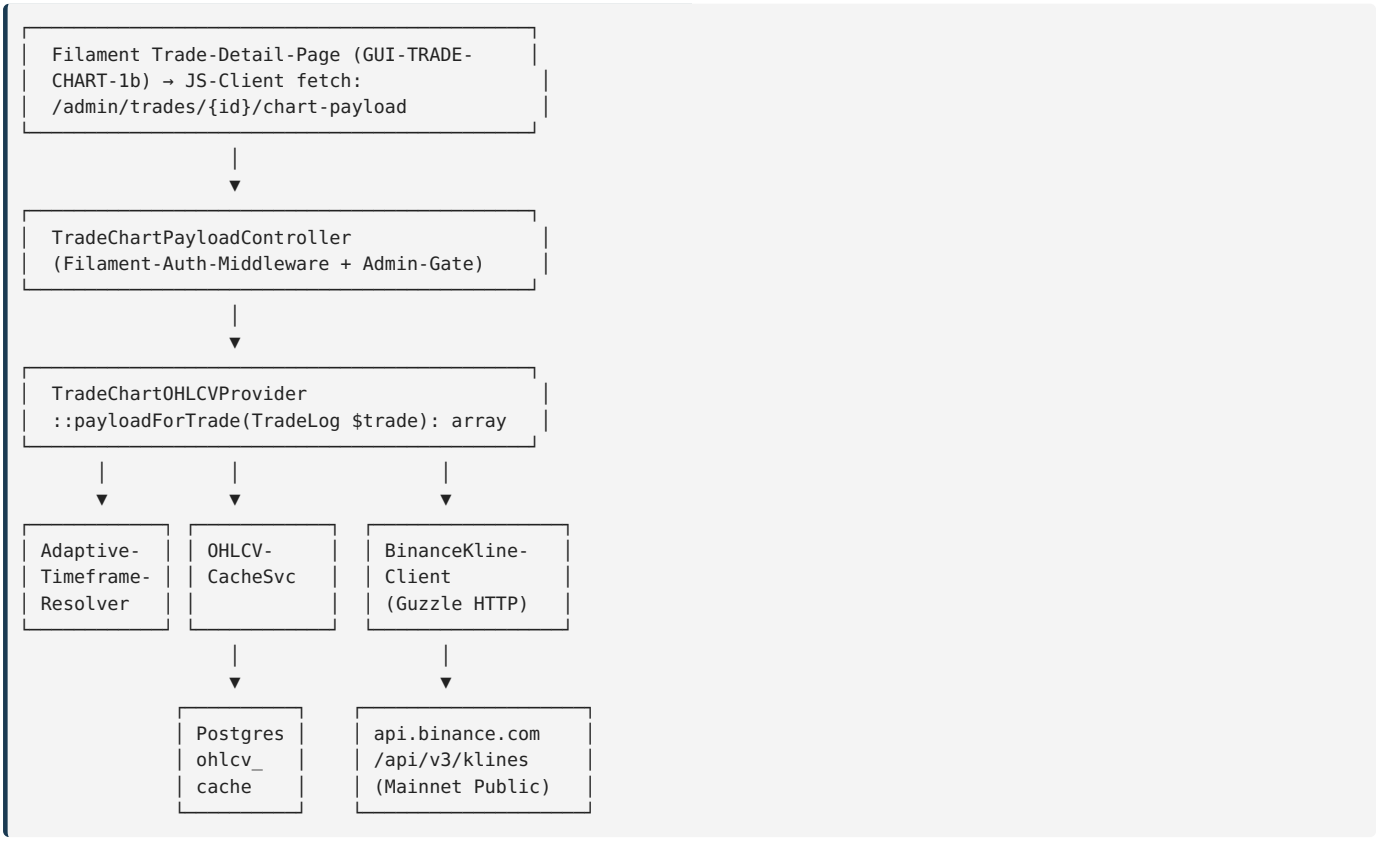
`open` wird beim Upsert **nicht überschrieben** (Binance-Closed-Bars sind final; nur Volumes können bei replikations-lag geringfügig variieren). Für **laufende Bars** (open_time + tf > NOW): nicht cachen — direct fetch + return-only.

### 5.5 Idempotenz

- Mehrfacher Aufruf mit identischem Window → 0 neue Inserts (alle Bars unique-constraint-blockiert oder ON CONFLICT)
- Unterbrechung mittendrin → nächster Aufruf liefert die fehlenden Bars (Window-Coverage-Check)

---

## Phase 6 — Service-Design (PHP / Laravel / Filament)




**Komponenten (alle PHP/Laravel, alle in `gui/app/Services/Ohlcv/`):**

| Klasse | Zweck | LoC-Schätzung |
|---|---|---|
| `AdaptiveTimeframeResolver` | Trade-Duration → TF (5 buckets) | ~40 |
| `OHLCVCacheService` | Read/Write `ohlcv_cache`, Coverage-Check, Upsert | ~120 |
| `BinanceKlineClient` | Guzzle GET `/api/v3/klines`, Rate-Limit-Wait, 429-Backoff | ~100 |
| `TradeChartOHLCVProvider` | Orchestrator: Resolver → Cache → Client → JSON-Payload | ~80 |
| `TradeChartPayloadController` | Route + Admin-canAccess()-Gate | ~30 |

**Gesamtcode:** ~370 LoC + Tests.

### 6.1 BinanceKlineClient Details

- Base-URL `https://api.binance.com`
- Endpoint `/api/v3/klines`
- HTTP-Timeout 10s
- Retry: max 3, Backoff 1s/2s/4s bei 5xx oder Connection-Errors
- 429-Handling: respect `Retry-After` header
- Symbol-Konversion: `str_replace('/', '', $symbol)` (validiert gegen statische Whitelist von 6-stelligen `[A-Z0-9]+/USDT`-Patterns)
- Antwort-Decoding: Array → DTO `[open_time, open, high, low, close, volume]`
- **0 API-Keys**, **0 Auth-Header**, **0 POST/PUT/DELETE**

### 6.2 OHLCVCacheService Details

- `getForWindow(symbol, tf, start, end): Collection<OHLCVBar>` — Read-or-fetch
- `upsertBars(symbol, tf, array $bars): int` — Bulk-INSERT mit ON CONFLICT
- `coverageGaps(symbol, tf, start, end): array<[gap_start, gap_end]>` — gibt fehlende Sub-Windows zurück
- Transaktions-Boundary: ein Upsert pro Symbol-TF-Fenster

### 6.3 JSON-Payload-Format

```
{
  "trade": {
    "id": 42,
    "symbol": "BTC/USDT",
    "opened_at": "2026-05-15T10:00:00Z",
    "closed_at": "2026-05-15T14:30:00Z",
    "entry_price": 67000.5,
    "exit_price": 68500.0,
    "side": "long"
  },
  "chart": {
    "symbol": "BTC/USDT",
    "timeframe": "5m",
    "window": {"start": "2026-05-15T09:30:00Z", "end": "2026-05-15T15:00:00Z"},
    "bars": [
      [1747300800000, 66980.0, 67020.5, 66950.0, 67000.5, 12.34],
      ...
    ]
  },
  "markers": {
    "entry": {"time": "2026-05-15T10:00:00Z", "price": 67000.5},
    "exit":  {"time": "2026-05-15T14:30:00Z", "price": 68500.0}
  },
  "meta": {
    "source": "binance_mainnet",
    "cache_hit": true,
    "fetched_at": "2026-05-15T17:18:00Z"
  }
}
```

Trailing-Stop-Replay (GUI-TRADE-CHART-1c/d) wird über separate Felder ergänzt — abhängig von TRAIL-EVENT-LOG-1.

---

## Phase 7 — Migration-Plan

### 7.1 Laravel-Migration

Pfad: `gui/database/migrations/2026_05_15_NNNNNN_create_ohlcv_cache_table.php`

```
public function up(): void {
    Schema::create('ohlcv_cache', function (Blueprint $t) {
        $t->bigIncrements('id');
        $t->string('symbol', 32);
        $t->string('timeframe', 8);
        $t->timestampTz('open_time');
        $t->decimal('open',   20, 8);
        $t->decimal('high',   20, 8);
        $t->decimal('low',    20, 8);
        $t->decimal('close',  20, 8);
        $t->decimal('volume', 28, 8);
        $t->string('source', 32)->default('binance_mainnet');
        $t->timestampTz('fetched_at')->useCurrent();
        $t->unique(['symbol','timeframe','open_time'], 'ohlcv_cache_unique');
        $t->index(['symbol','timeframe','open_time'], 'ohlcv_cache_lookup_idx');
        $t->index('fetched_at', 'ohlcv_cache_fetched_at_idx');
    });
}

public function down(): void {
    Schema::dropIfExists('ohlcv_cache');
}
```

### 7.2 Migrator-Path

- Migrator-Connection ( `tradingbot_gui_migrator` ) hat `CREATE` auf public-schema → CREATE TABLE läuft sauber.
- Owner-Konflikt: bestehende Tables sind owned by `tradingbot_gui` (Legacy-SUPERUSER). Migrator legt **neue** Tabelle an → Owner = `tradingbot_gui_migrator` . Das ist OK für die App-Connection ( `tradingbot_gui_app` ) solange explizite `GRANT SELECT, INSERT, UPDATE, DELETE ON ohlcv_cache TO tradingbot_gui_app` Teil der Migration ist.

→ **Add to migration**: `DB::statement('GRANT SELECT, INSERT, UPDATE, DELETE ON ohlcv_cache TO tradingbot_gui_app');`

(SEC-1c-3c hat `ALTER DEFAULT PRIVILEGES` für beide Issuer schon eingerichtet — aber für neue Tabellen sicherheitshalber explizit

grants.)

### 7.3 Migration-Lauf

- Pre-Check: `php artisan migrate:status` (read-only)
- Execute: `php artisan migrate` (1 neue Tabelle, ~30s)
- Rollback (falls nötig): `php artisan migrate:rollback --step=1`

Migrationsfenster: <1 min. **Kein Bot-Touch, kein Worker-Touch, kein Restart**.

---

## Phase 8 — Test-Plan

### 8.1 Unit-Tests (~150 LoC)

| Test | Coverage |
|---|---|
| `AdaptiveTimeframeResolverTest` | 5 buckets: <1h→1m, <6h→5m, <24h→15m, <168h→1h, ≥168h→1d |
| `BinanceKlineClientTest` | Mock Guzzle: 200 happy-path, 429 retry-after, 5xx retry, timeout |
| `BinanceKlineClientSymbolMappingTest` | `BTC/USDT` → `BTCUSDT`, invalid symbols rejected |

### 8.2 Feature-Tests (~200 LoC)

| Test | Coverage |
|---|---|
| `OHLCVCacheServiceTest::cache_miss_then_hit` | Cold cache → fetch → second call → cache_hit=true |
| `OHLCVCacheServiceTest::partial_coverage_gap_fill` | Cache hat 50%, restliche 50% werden gefetcht |
| `OHLCVCacheServiceTest::upsert_idempotent` | Doppel-Upsert → 0 neue Rows, kein Constraint-Error |
| `TradeChartOHLCVProviderTest::payload_format` | Vollständiger JSON-Payload, alle Felder |
| `TradeChartOHLCVProviderTest::null_opened_at_fallback` | trade_logs ohne opened_at → trade_id-Parse oder Default-Window |
| `TradeChartPayloadControllerTest::admin_200` | Admin-User → 200 |
| `TradeChartPayloadControllerTest::viewer_403` | Viewer-User → 403 |
| `TradeChartPayloadControllerTest::guest_redirect` | Unauthenticated → /admin/login |

Alle Feature-Tests mit `RefreshDatabase` + Mock-HTTP (kein echter Binance-Call in CI).

### 8.3 Integration-Smoke (manuell, post-deploy)

- 1× echter Trade-Detail-Aufruf via Browser → Payload sichtbar
- 2× selber Aufruf → Cache-Hit ( `fetched_at` unverändert)
- DB-Check: `SELECT COUNT(*) FROM ohlcv_cache` > 0

---

## Phase 9 — Risikoanalyse

| # | Risiko | Wahrscheinlichkeit | Impact | Mitigation |
|---|---|---|---|---|
| R1 | Binance IP-Rate-Limit (6000 Weight/min) bei viel Operator-Navigation | niedrig | mittel | Cache schützt; `BinanceKlineClient` respektiert `Retry-After`; max 3 retries |
| R2 | Symbol-Mapping-Fehler ( `SOL/USDT` → unbekanntes Format) | niedrig | niedrig | Whitelist-Regex `[A-Z0-9]+/USDT`, sonst HTTP 400 |
| R3 | `trade_logs.opened_at = NULL` (alle pre-DATA-LINK-1 56 Rows) | **hoch** | mittel | Fallback: trade_id-Parse + Default-6h-Window |
| R4 | DB-Wachstum unkontrolliert | niedrig | niedrig | 180d-Retention als P2-Backlog (OHLCV-CACHE-1-RETENTION) |
| R5 | Mainnet-Boundary-Bruch | sehr niedrig | hoch | nur GET, keine Keys, keine Order-Routen; Reviewbar in Code-Audit |
| R6 | Binance gibt revidierte Bars zurück (Trade-Re-Sequencing) | sehr niedrig | sehr niedrig | ON CONFLICT DO UPDATE für h/l/c/v (open bleibt) |
| R7 | Migrator-Owner ungleich App-User → GUI kann nicht INSERT/UPDATE | mittel | hoch | **Explizite GRANTs in Migration** (Phase 7.2) |

| | | | | |
|---|---|---|---|---|
| R8 | Race-Conditions bei Parallel-Refill desselben Windows | niedrig | sehr niedrig | unique constraint serialisiert; ON CONFLICT no-op |
| R9 | TradingView Lightweight Charts Lizenz | n/a | n/a | Apache-2.0, kein Risiko |
| R10 | Stale Bar-Daten (offene Bar in der Vergangenheit) | sehr niedrig | niedrig | Binance liefert finals nach close; bei offenem aktuellen Bar nicht cachen |
| R11 | psycopg2 / Bot-Telemetry-Outage (TELE-1b-Echo) | sehr niedrig | n/a | Bot komplett unangetastet in dieser Phase |
| R12 | Hat Konflikt mit GUI-DESIGN-1b-Widget-Branch | niedrig | niedrig | Branch ist in worktree, kein File-Overlap mit `app/Services/Ohlcv/` |

**Höchstes Risiko: R3 (NULL opened_at)** — wird durch Fallback-Logik in `TradeChartOHLCVProvider` aufgefangen. Realistisch betroffen sind nur die 56 pre-cutover trade_logs; alle neuen Trades haben `opened_at` korrekt gefüllt (DATA-LINK-1).

---

## Phase 10 — Executive Output + GO/NO-GO

### 10.1 Empfehlung

**GO** für OHLCV-CACHE-1-IMPL als separate Implementations-Phase. **NO-GO sofort**, weil:

1. Operator hat **OHLCV-CACHE-1 explizit als read-only Plan-Review** beauftragt
2. Implementation wartet auf Operator-Decisions D1-D4 unten
3. GUI-DESIGN-1b-Widget-Branch sollte vorher gereviewt/gemerged werden um Test-Konflikte zu vermeiden

### 10.2 Operator-Decisions

| ID | Frage | Empfehlung |
|---|---|---|
| **D1** | Cache-Retention sofort oder als Backlog? | **Backlog** (OHLCV-CACHE-1-RETENTION P2) — Wachstum erstmal beobachten |
| **D2** | Symbol-Whitelist hartcoded oder dynamisch via `exchangeInfo` ? | **Hartcoded Regex** `[A-Z0-9]+/USDT` + DB-Lookup im `trade_logs.symbol` -Set — kein externer Call beim Validate |
| **D3** | `BinanceKlineClient` direkt in PHP oder ccxt-Bridge zum Bot? | **PHP direct** (Guzzle) — sauberer Stack-Cut, Bot bleibt unangetastet |
| **D4** | Migration-Fenster jetzt, nach GUI-DESIGN-1b-Merge, oder nach SEC-1c-3c-Stabilitätsfenster (frühestens 2026-05-21)? | **Nach GUI-DESIGN-1b-Merge** — kein Konflikt mit SEC-1c-3c |

### 10.3 Aufwandsschätzung

| Block | Aufwand |
|---|---|
| Migration (Phase 7) | 15 min |
| Services (Phase 6, 5 Klassen, ~370 LoC) | 3-4 h |
| Tests (Phase 8, ~350 LoC) | 2 h |
| Filament-Integration (Trade-Detail-Page-Slot) | 1-2 h (in GUI-TRADE-CHART-1b) |
| Container-Recreate (GUI only) | 5 min |
| **Gesamt OHLCV-CACHE-1-IMPL** | **~6-8 h** |

### 10.4 Boundaries

- 0× Bot-Code-Touch
- 0× Worker-Touch
- 0× docker cp
- 0× Mainnet-Order-Surface (read-only Public-Endpoints)
- 0× API-Key-Surface
- 0× Restart des Bots
- 0× Push zu remote
- 0× Secret-Output
- BINANCE_TESTNET=true unverändert

### 10.5 Nächste Schritte nach OHLCV-CACHE-1-IMPL

1. **TRAIL-EVENT-LOG-1** (separate Phase) — persistiert Trailing-Stop-Events für Replay
2. **GUI-TRADE-CHART-1b** — Filament-Page-Slot rendert TradingView-Lightweight-Charts mit Payload

3. **GUI-TRADE-CHART-1c** — Entry/Exit-Marker + Pre-/Post-Trade-Zone
4. **GUI-TRADE-CHART-1d** — Trailing-Replay (depends on TRAIL-EVENT-LOG-1)

### 10.6 Mainnet-Relevanz (MH-7-Pfad)

OHLCV-CACHE-1 ist **kein Blocker für MH-7**, aber **stark empfohlen davor**: - Operator wird Trade-Auswertung vor Mainnet-Go-Live brauchen → Charts auf Testnet-Trades essentiell - Vermeidet zur Mainnet-Phase weitere Architekturarbeit unter Druck

### 10.7 STOP

**Plan-Review abgeschlossen.** Operator-GO für OHLCV-CACHE-1-IMPL nach D1-D4-Antwort + GUI-DESIGN-1b-Review.

---

*Generated 2026-05-15 17:18 UTC · Read-only Plan-Review · 0 Code/Migration/Restart · Master HEAD fa7e450*