

Unten ist ein **neu generierter Implementation Plan** (inkl. **SKILLS & RULEs**) – erweitert um:

- **ICS-Label “Wirtschafts-Event” für *alle* Events** (Outlook-Filterkriterium) via iCalendar-Property CATEGORIES (Outlook importiert beliebige Category-Werte). ([Microsoft Learn](#))
  - **Quellenstrategie:** Primär **Investing.com**, Sekundär **TradingView**, plus **tertiäre Quellen** (z. B. Reuters *wenn lizenziert*, außerdem offizielle Stellen) als „Verifier/Backfill“ unter klaren Prioritätsregeln.
- 

## 1) Ziel & UX (Web Designer View)

**One-Page App (klar, deterministisch, auditierbar):**

1. **Country Scope** per Checkbox (USA/EZ/UK/JP/CH/CA/AU/NZ)
2. Button „**Wochenausblick generieren**“ → Ergebnisliste im UI im *exakt* geforderten Schema
3. Button „**ICS herunterladen**“ → Outlook-kompatible ICS-Datei, alle Events mit Kategorie **Wirtschafts-Event**

## UI Layout

- Card „Country Scope“ (Checkbox Grid + „Alle / Keine“)
  - Card „Aktionen“ (Generate / Download ICS)
  - Card „Output“ (<pre> monospace, copyable) – Ausgabe exakt wie Spezifikation, ohne Extras
- 

## 2) Architektur (Codex-freundlich, testbar)

**Frontend:** Next.js (React) + TypeScript

**Backend:** Next.js API Routes (Server-side Fetching/Parsing, CORS-sicher)

**Parsing:** serverseitig (HTTP + optional Playwright-Fallback)

**ICS:** serverseitig generieren (CRLF, Folding, TZID Europe/Berlin)

## Endpoints

- GET /api/weekly?countries=USD,EUR,... → returns { renderedText, events[] }
  - GET /api/weekly.ics?countries=... → returns text/calendar download
- 

## 3) Datenmodell (intern)



## EconomicEvent

- source: "investing" | "tradingview" | "tertiary:<name>"
- country: USA | EZ | UK | JP | CH | CA | AU | NZ
- currency: USD | EUR | GBP | JPY | CHF | CAD | AUD | NZD
- titleRaw, titleNormalized
- categoryAF: A|B|C|D|E|F
- datetimeBerlinISO, timeHHMM
- isTopEvent: boolean
- hasExactTime: boolean
- provenance: { urlHash?, fetchedAtISO, parserVersion } (nur intern)

## RenderDay

- dayHeader: "### Mittwoch, 11. Februar"
  - lines: string[] **oder** note: string (exakt eine Hinweiszeile)
- 

## 4) Deterministische Pipeline (End-to-End)

1. **WeekResolver (Europe/Berlin)**
  - Mo–Fr der relevanten Woche (Sa/So → kommende Woche)
2. **HolidayEngine (Market-relevant)**
  - Feiertage pro Land prüfen; betroffene Länder-Events entfernen
  - Falls Tag danach leer → Hinweis „Feiertag“ (oder „Wochenende“)
3. **Fetch Primary: Investing.com (Pflicht)**
4. **Fetch Secondary: TradingView.com (Pflicht)**
5. **Normalize + Timezone Convert**
  - Alles nach Europe/Berlin
  - Drop: All Day, Tentative, fehlende Uhrzeit
6. **Filter & Classify (A–F)**
  - Keyword-basierte, versionierte Regeln (keine “LLM-Interpretation” im Core)
7. **Merge & Conflict Resolution**



- **Priorität:** Investing > TradingView > Tertiary
- Konflikt (Zeit/Titel): höhere Quelle gewinnt (Investing dominiert)

#### 8. Tertiary Sources (neu) – *nur unter strengen Bedingungen*

- Rolle: **Verifier/Backfill**, nicht “free-for-all”
- Trigger:
  - (a) Investing+TradingView liefern widersprüchliche Zeiten **oder**
  - (b) Event ist in Investing/TradingView vorhanden, aber Zeit fehlt/unklar (und Tertiary kann exakt bestätigen) **oder**
  - (c) Primary/Secondary temporär nicht erreichbar → Tertiary nur, wenn als „Approved“ markiert
- Beispiele “Approved Tertiary”:
  - Offizielle Statistikämter/Behörden (BLS, BEA, Eurostat, ONS, SNB/BoJ/BoE/ECB/Fed Kalender, Statistics Canada, ABS, Stats NZ)
  - Reuters **nur**, wenn legaler Zugriff (API/Lizenz). (Sonst deaktiviert)

#### 9. Dedupe + Grouping

- Dedupe-Key: country + datetimeBerlin + titleNormalized
- Grouping: gleiche Country+Zeit → 1 Zeile, Titel mit " / " verbinden

#### 10. TOP-EVENT Marker

- Zentralbank, CPI/PCE, NFP, BIP/GDP, PMI/ISM → suffix exakt: - **\*\*TOP-EVENT\*\***

#### 11. Strict Renderer (DE)

- Header + Mo–Fr Tagesblöcke
- Wenn leer/unsicher: exakte Hinweiszeile

---

### 5) ICS-Export (Outlook kompatibel) + Kategorie “Wirtschafts-Event”


**Wichtigster Punkt:** Jeder VEVENT bekommt:

- CATEGORIES:Wirtschafts-Event

CATEGORIES ist Standard-iCalendar Property (RFC 5545) ([datatracker.ietf.org](https://datatracker.ietf.org)) und Microsoft/Outlook kann beim Import beliebige Werte für CATEGORIES übernehmen (inkl. frei benannter Kategorien). ([Microsoft Learn](#))

#### ICS-Richtlinien



- VCALENDAR:
  - VERSION:2.0
  - PRODID:-//<DeinProdukt>//Wochenausblick//DE
  - CALSCALE:GREGORIAN
  - METHOD:PUBLISH
- VTIMEZONE für Europe/Berlin (empfohlen, um Outlook/DST stabil zu halten)
- VEVENT (pro *Einzel-Event* oder pro *Gruppenzeile* – empfehlenswert: pro Einzel-Event, aber Summary darf gruppiert sein):
  - UID: deterministisch (Hash aus weekStart|country|datetime|titleNormalized)
  - DTSTAMP: build-time UTC
  - DTSTART;TZID=Europe/Berlin:YYYYMMDDTHHMM00
  - DTEND;TZID=Europe/Berlin:... (+15 min Default)
  - SUMMARY: z. B. USA Verbraucherpreisindex (CPI) & Kernrate - TOP-EVENT
  - DESCRIPTION: optional (z. B. “Quelle: Investing/TradingView”) – **nur wenn du das willst**, UI-Ausgabe bleibt weiterhin quellenfrei
  - CATEGORIES:Wirtschafts-Event  **Filter-Kriterium**
- **CRLF** und **Line Folding (75 octets)** für Outlook-Interoperabilität (Microsoft empfiehlt Folding/Import tolerant, aber sauber ist besser). ([Microsoft Learn](#))

### Download “Wunschverzeichnis”

- Standard: Content-Disposition: attachment; filename="Wochenausblick\_YYYY-MM-DD.ics"
- Optional (Chromium): File System Access API → Directory Picker + Save As (Fallback auf Standard-Download)

---

## 6) SKILLS (Module/Capabilities)

1. **WeekResolver (Europe/Berlin)**
2. **CountryScopeManager (Checkbox → Scope)**
3. **InvestingCalendarFetcher (Primary)**
4. **TradingViewCalendarFetcher (Secondary)**



## 5. **SourceRegistry & Governance (neu)**

- Registry „Approved Sources“, ToS/Lizenz-Flags, Rate Limits

## 6. **TertiaryVerifier/Backfill Engine (neu)**

- Offizielle Kalender/Release-Seiten, optional Reuters API Adapter

## 7. **EventNormalizer (Time parsing + Berlin TZ)**

## 8. **EventClassifier (A–F, keyword rules, versioned)**

## 9. **ConflictResolver (Priority Investing > TV > Tertiary)**

## 10. **DedupeAndGrouper**

## 11. **TopEventMarker**

## 12. **HolidayEngine (Market Holidays)**

## 13. **StrictRendererDE (byte-exakt)**

## 14. **IcsGeneratorOutlook (CATEGORIES + TZID + CRLF + Folding)**

## 15. **Determinism Layer (cache + snapshots + parserVersion pinning)**

## 16. **QA Harness (fixtures + snapshot tests + golden outputs)**

## 17. **Observability & Rate Limiting (prod-safe)**

---

## 7) **RULEs (harte Acceptance Criteria)**

### **RULE 1 — Source Priority & Conflict**

- Investing dominiert immer. TradingView nur Ergänzung. Tertiary nur Verifier/Backfill.
- Konflikte: Investing > TradingView > Tertiary.

### **RULE 2 — Exact Time Only**

- Nur Events mit exakter Uhrzeit, sonst Drop.

### **RULE 3 — Output Strictness**

- UI-Textausgabe exakt im Schema, ohne Links/Quellen/Extras.

### **RULE 4 — Country Scope**

- Nur ausgewählte Länder aus erlaubter Liste.

### **RULE 5 — Kategorien A–F**

- Nur Events, die eindeutig matchen; sonst Drop.



## **RULE 6 — Feiertagslogik**

- Wochenende/Feiertag/Unsicherheit → exakt definierte Hinweiszeilen.

## **RULE 7 — Sortierung & Gruppierung**

- Chronologisch; gleiche Country+Zeit → eine Zeile mit " / ".

## **RULE 8 — TOP-EVENT Suffix**

- Exakt - **\*\*TOP-EVENT\*\*** (Spacing/Markdown exakt).

## **RULE 9 — ICS Kategorie (NEU, kritisch)**

- **JEDES** VEVENT enthält exakt: CATEGORIES:Wirtschafts-Event ([Microsoft Learn](#))
- ICS wird mit CRLF + Folding erzeugt.

## **RULE 10 — Determinism**

- Gleiche Inputs + gleiche Rohdaten-Snapshots + gleiche parserVersion → identischer Output & identische UUIDs.

---

## **8) Codex-Workflow (AI Code Generation Plan)**

### **Codex “Tickets” (klein, testbar)**

1. weekResolver.ts + Tests
2. UI Checkbox Grid + Query Builder
3. investing.fetch+parse.ts + HTML fixtures
4. tradingview.fetch+parse.ts + fixtures
5. classifier.rules.ts (A–F) + Tests
6. merge.conflict.ts + Tests
7. holiday.engine.ts + Tests
8. renderer.strictDE.ts + Snapshot Tests (Golden Strings)
9. ics.generator.ts + Snapshot Tests (Golden ICS; prüfe CATEGORIES)
10. sourceRegistry.ts + tertiaryAdapters/\* (offizielle Quellen, Reuters nur via API-Key)

### **Codex RULES**

- Core-Logik als pure functions, keine UI-Abhängigkeiten
- Jede Regel als Konstante/Enum, versioniert
- Parser über Fixtures testen (keine “Live”-Tests in CI)



- Golden-Output Snapshots für **exakte** Formatkonformität
- 

## 9) Praktische Hinweise zu “weitere Quellen” (ohne Regelbruch)

- Implementiere Tertiary als **opt-in “Approved Sources”**:
  - *Default*: nur Investing + TradingView (maximale Prompt-Treue)
  - *Enhanced*: offizielle Stellen + optional Reuters API, **ohne** Investing zu überschreiben