Statusbericht: Forex Agent "As-Is" Status

Datum: 21. Januar 2026 **Version:** 1.0 - MVP Analyse

1. Executive Summary

Der "Forex Agent" befindet sich aktuell in der **MVP (Minimum Viable Product) Phase**. Die grundlegende modulare Architektur für einen automatisierten Trading-Bot ist implementiert. Der Agent kann einen vollständigen Trading-Zyklus (Daten -> Signal -> Risiko -> Order) simulieren, nutzt dafür aber derzeit noch überwiegend generierte Dummy-Daten und einen Mock-Broker. Die Integration eines echten Brokers (IG Markets) wurde begonnen, ist aber technisch noch im Prototypen-Stadium.

2. Aktueller Projektstatus

- **Gesamtstatus:** Prototyp / MVP (Lauffähig im Simulations-Modus)
- **Code-Qualität:** Modularer Python-Code, klare Trennung der Verantwortlichkeiten.
- **Test-Abdeckung:** Manuelle Skripte (`verify_full_pipeline.py`) vorhanden; keine Unit-Tests.

3. Architektur-Komponenten Analyse

3.1 Data Layer (Daten-Ebene)

- **Status:** Initialisiert, aber leer.
- **Implementierung:** Ordner `data/raw` und `data/processed` existieren.
- **Funktionalität:** Es gibt aktuell **keinen** Live-Datenfeed. Die Datei `verify_full_pipeline.py` enthält eine Funktion `generate_dummy_data()`, die synthetische Preise für den Test generiert.
- **Gap:** Anbindung an eine echte API (z.B. IG, AlphaVantage, OANDA) fehlt.

3.2 Signal Engine (Signal-Logik)

- **Status:** Funktional (Basis).
- **Strategie:** Eine Strategie ist implementiert: `BaselineSMACross` (Simple Moving Average Crossover).
- **Logik:** Schnell (50) kreuzt Langsam (200) -> Signal Generierung.
- **Gap:** Keine komplexen Indikatoren oder Multi-Timeframe-Analysen.

3.3 Risk Management (Risiko-Modul)

- **Status:** Implementiert & Integriert.
  Komponenten:
  - `RiskConfig`: Definiert Limits (Max Daily Loss, Max Trades, Risk per Trade).
  - `risk_eval.py`: Prüft Signale gegen Kontostand und Limits.
- **Funktionalität:** Berechnet Positionsgrößen basierend auf Kontostand (Equity) und Risiko-Parametern. Stop-Loss und Take-Profit Logik ist vorhanden.

3.4 Execution Layer (Ausführung)

- **Status:** Hybrid (Mock fertig, Real in Arbeit).
  Mock Broker (`mock_broker.py`):
  - Voll funktionsfähig. Simuliert Order-Ausführung ("FILLED") und führt kein echtes Tracking durch.

  IG Broker (`ig_broker.py`):
  - Integration gestartet.
  - Verbindung (`IGService`) implementiert.
  - Balance Abfrage implementiert.
  - Order Ausführung implementiert, aber Logik zur Umrechnung von 'Units' zu 'Contracts' (Lots) ist noch rudimentär und hardcodiert (z.B. Annahme Mini-Contracts für USDJPY).

- Epics (Symbol-IDs) sind teilweise hardcodiert.

3.5 Orchestration (Steuerung)

- **Status:** Skript-basiert.
- **Steuerung:** `verify_full_pipeline.py` dient als Haupt-Runner, um den Prozess einmalig durchzuspielen. Keine dauerhafte "Loop" oder CRON-Job Logik sichtbar.

4. Identifizierte Lücken & Nächste Schritte

Priorität	Komponente	Aufgabe	Status
Hoch	Data Layer	Echten Marktdaten-Feed anbinden (Live Candle Data).	Offen
Hoch	Execution	IG Broker Logik finalisieren (Unit vs. Contract Mapping validieren).	In Arbeit
Mittel	System	End-to-End Test mit Demo-Account durchführen.	Offen
Mittel	Strategy	Implementierung weiterer Strategien (z.B. RSI, MACD).	Geplant
Niedrig	Infra	Datenbank (SQLite/Postgres) statt CSV/In-Memory.	Geplant

5. Fazit

Das Fundament ist solide. Der Code ist sauber strukturiert und erlaubt Erweiterungen. Der kritische Pfad zur Profitabilität ist nun die Anbindung echter Live-Daten und die Validierung der IG-Broker-Ausführung, um vom "Dummy-Modus" in den "Demo-Trading-Modus" zu wechseln.