AURORA

Personal Investment Intelligence System (uso personale)

Data: 01/01/2026
Scopo: specifica di progetto + codex tecnico per sviluppo in locale e successiva migrazione su VPS.
Focus fase 0-3/4 anni: ETF-only con PAC 150-200 €/mese e profilo aggressivo.

Executive summary

AURORA e' un sistema operativo personale per investire: definisce una policy (IPS) versionata, seleziona e valuta ETF candidati tramite scoring riproducibile, propone un portafoglio coerente con vincoli e tolleranze, e gestisce un PAC mensile con ribilanciamento principalmente tramite contributi (contributi-only). Nella fase successiva (3/4 anni) AURORA estende l'universo a ETF + azioni, introducendo uno screener di Quality per settore.

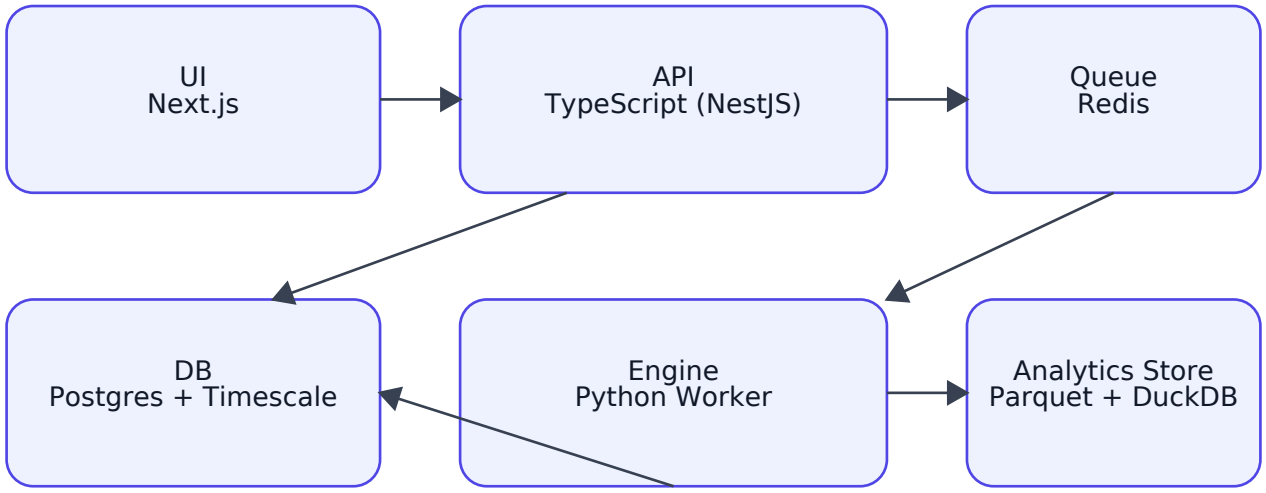
Principi chiave

- Dati auditabili e versionati (ogni proposta ha data_asof e run_id).
- Niente black box: ogni proposta include motivazione numerica e red flags.
- Decisioni guidate da policy + vincoli + costi (non da opinioni).
- Automazione prudente: alert utili, pochi ma rilevanti.

1. Che cos'e' AURORA

AURORA e' un sistema modulare composto da UI (dashboard), API (logica di prodotto) e un Engine analytics (calcoli finanziari e scoring). Il sistema gira in locale con Docker; la migrazione su VPS mantiene la stessa architettura.

Architettura logica (locale; poi identica su VPS)



Uso previsto: esclusivamente personale. Nessuna distribuzione a terzi e nessuna funzionalita' di consulenza per clienti.

2. Obiettivi e non-obiettivi

Obiettivi

• Costruire un portafoglio ETF-only iniziale coerente con profilo aggressivo e PAC 150-200 €/mese.

• Avere un processo ripetibile: IPS -> universe -> scoring -> proposta -> PAC -> monitor.

• Ridurre errori tipici: eccessiva complessita', trading emotivo, mancanza di regole.

• Preparare l'evoluzione a ETF + azioni (entro 3/4 anni) con screener Quality per settore.

Non-obiettivi (fase iniziale)

• Real-time trading e intraday.

• Previsioni di mercato 'magiche' o segnali non spiegabili.

• Ottimizzazioni complesse senza dati e governance (si parte semplice e misurabile).

• Gestione fiscale automatica (dipende dal broker; AURORA e' broker-agnostica).

3. IPS v1 (Investment Policy Statement) - versione iniziale

L'IPS e' la 'costituzione' del sistema: regole, obiettivi, vincoli e parametri di ribilanciamento. Ogni proposta di portafoglio deve superare un controllo vincoli (constraint check) rispetto all'IPS attiva.

Parametro	Valore (IPS v1)
Profilo	Aggressivo (fase 4-5 anni)
Strumenti iniziali	Solo ETF (ETF-only)
Orizzonte fase 1	3/4 anni ETF-only; poi ETF + azioni (satellite)
PAC	150-200 EUR/mese
Frequenza acquisti	Mensile
Ribilanciamento	Contributi-only (vendite solo se drift estremo)
Banda asset class	±5% assoluto (configurabile)
Banda per strumento	±3% assoluto (configurabile)
Vincoli base	Min 1 ETF; max 3 ETF nella fase iniziale (consigliato 1-2)

Default portafoglio iniziale (scelta di partenza)

AURORA v1 propone come default un portafoglio estremamente semplice per massimizzare disciplina e ridurre costi cognitivi: Single ETF Core (100% azionario globale). In alternativa, se si desidera ridurre swing emotivi: Core + Stabilizzatore (90% azionario globale + 10% obbligazionario globale hedged EUR o monetario).

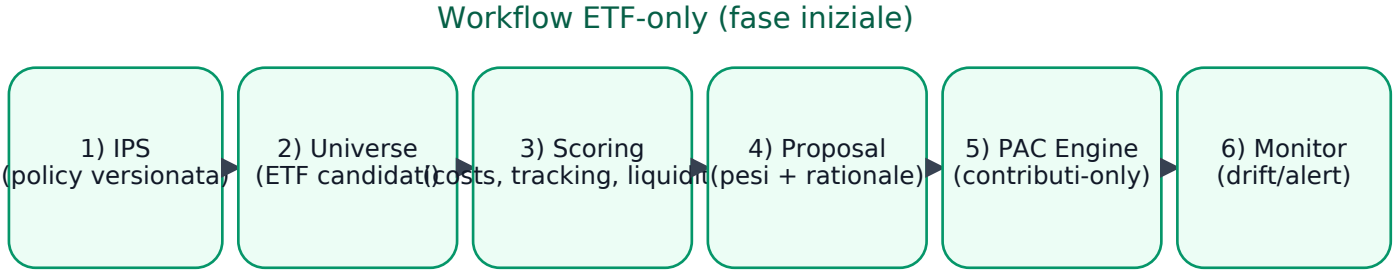
Esempio IPS v1 (JSON salvato in DB, versionato)

```
{
"version": "1.0.0",
"profile": "aggressivo",
"phase": "ETF-only",
"horizon_years": 4,
"pac_monthly_eur": { "min": 150, "max": 200, "default": 200 },
"targets": [
{ "bucket": "equity_global", "weight": 1.00 }
],
"rebalance": {
"mode": "contributi-only",
"frequency": "monthly",
"bands": {
"asset_class_abs": 0.05,
"instrument_abs": 0.03
},
"sell_only_if_extreme_drift": true
},
"constraints": {
"min_instruments": 1,
"max_instruments": 3,
"max_single_instrument_weight": 1.00
```

```
}
}
```

4. Workflow operativo (ETF-only)

Questo e' il flusso standard che AURORA esegue. Ogni step produce artefatti salvati (proposta, rationale, run_id).



Algoritmo PAC (v1): contributi-only

Dato un target di pesi e il valore attuale delle posizioni, AURORA calcola il drift per ciascun ETF e alloca il contributo mensile acquistando lo strumento piu' sotto target. Le vendite sono evitate salvo drift estremo (oltre 2x banda).

```
Input: targets w_i, current_values v_i, contribution C
Compute portfolio_value V = sum(v_i)
Compute current_weight_i = v_i / V
Compute drift_i = w_i - current_weight_i
Select k = argmax(drift_i) # più sotto target
Buy ETF_k for amount C (or as close as possible given broker min)
Store: proposal(trade_list, drift_snapshot, data_asof, run_id)
```

Alert utili (v1)

- Drift oltre banda (es. asset class > ±5%).
- Concentrazione involontaria (es. singolo ETF > soglia se hai 2-3 strumenti).
- Cambio policy (nuova versione IPS): richiede rigenerazione proposta e conferma.

5. Selezione ETF: filtri e scoring

AURORA non 'indovina' il mercato: seleziona strumenti robusti minimizzando rischi strutturali. Lo scoring serve a scegliere tra ETF simili (stesso bucket) in modo misurabile.

Filtro hard (must-have)

- UCITS; quotazione su mercato accessibile; ISIN valido e mapping corretto.
- Categoria corretta (es. equity globale, bond globale hedged EUR).
- AUM minimo e storico minimo (soglie configurabili).
- Dati sufficienti per calcolare metriche (missingness sotto soglia).

Scoring (0-100) - v1

Componente	Descrizione	Peso indicativo
Replica / tracking	Tracking difference / error vs benchmark	Alto
Costi	TER + costi impliciti stimati	Medio
Liquidita'	Spread/volume proxy, AUM	Medio
Robustezza	Provider, dimensione fondo, anzianita'	Basso
Preferenze	Accumulo vs distribuzione, replica fisica vs sintetica	Configurabile

Output per bucket: Top 3 candidati + motivazione numerica (score e breakdown) + red flags.

6. Tecnologie e motivazione (stack locale)

• UI: Next.js (dashboard professionale, tabelle/filtri, drill-down).

• API: TypeScript (NestJS) per logica prodotto, versioning IPS, alert, integrazioni.

• Engine: Python worker per scoring ETF e calcoli (data/finanza).

• DB: Postgres + Timescale per storage e time series prezzi/metriche.

• Queue/Cache: Redis (job, caching risultati).

• Orchestrazione (opz.): Dagster quando i job diventano numerosi/complessi.

Setup locale

Il sistema deve avviarsi con un singolo comando: docker compose up. La UI parla con API; l'API enqueua job su Redis; l'Engine consuma e scrive risultati in Postgres.

```
# Avvio (locale)
docker compose up -d

# Logs
docker compose logs -f api
docker compose logs -f engine

# Reset (dev)
docker compose down -v
```

7. Struttura repository (Codex)

```
aurora/
apps/
api/ # TypeScript (NestJS)
ui/ # Next.js
services/
engine/ # Python worker (analytics)
infra/
compose.yml # docker-compose
docker/ # Dockerfiles
packages/
shared/ # contratti (OpenAPI/JSON schema), types, validators
docs/ # note, ADR, guide
```

Convenzioni

- Ogni esecuzione engine produce: run_id, data_asof, input_hash, output_hash.
- Le policy sono versionate (ips_policy_version) e immutabili dopo creazione.
- Tutti i calcoli salvano la provenienza: sorgente dati, timestamp, e campo confidence.

8. Data model v1 (minimo ma robusto)

Questo modello copre IPS, strumenti, portafoglio paper, proposte e snapshot. E' sufficiente per partire e non dover rifare tutto quando aggiungi ETF multipli e poi azioni.

Tabella	Scopo
instrument	Security master: ISIN/ticker, currency, category, provider, metadata.
etf_metrics	Metriche ETF per scoring: TER, AUM, tracking, liquidita' proxy, ecc.
ips_policy	Identita' policy (utente).
ips_policy_version	Versioni IPS (JSON), immutabili; una e' attiva.
portfolio	Portafogli: paper e (futuro) real.
transaction	Operazioni (paper): buy/sell, fee, timestamp, instrument.
position_snapshot	Snapshot mensili: qty, value, weights, drift.
proposal	Output engine: trade list mensile o proposta iniziale + rationale.
alert	Eventi: drift, violazione vincoli, variazione policy, ecc.

9. Contratti: API endpoints e job payload (Codex operativo)

Questi contratti sono l'interfaccia tra UI/API/Engine. Servono per lavorare in IDE senza ambiguita'.

API (REST) - endpoints minimi

- POST /ips -> crea IPS
- POST /ips/{id}/versions -> crea nuova versione (JSON)
- POST /ips/{id}/activate/{version_id} -> attiva versione
- POST /universe/buckets -> definisce bucket (equity_global, bond_hedged, ecc.)
- POST /universe/instruments -> aggiunge ETF candidati al bucket
- POST /engine/run -> enqueua job (scoring/proposal/pac) e ritorna run_id
- GET /proposals?run_id=... -> legge proposta e rationale
- POST /portfolio/paper/transactions -> aggiunge transazione paper
- GET /portfolio/paper/snapshots -> ritorna snapshot e drift

Job payload (Redis queue)

```
{
"run_id": "uuid",
"type": "ETF_SCORING | PORTFOLIO_PROPOSAL | MONTHLY_PAC",
"data_asof": "YYYY-MM-DD",
"ips_version_id": "uuid",
"params": {
"bucket": "equity_global",
"max_candidates": 50
}
}
```

Engine output (salvato in proposal)

```
{
"run_id": "uuid",
"type": "MONTHLY_PAC",
"data_asof": "YYYY-MM-DD",
"trade_list": [
{ "instrument_id": "uuid", "side": "BUY", "amount_eur": 200.0 }
],
"rationale": {
"targets": [{ "bucket": "equity_global", "weight": 1.0 }],
"drift_snapshot": [{ "instrument_id": "uuid", "current_weight": 0.92,
"target_weight": 1.0 }],
"constraints_passed": true
},
"hashes": { "input": "sha256...", "output": "sha256..." }
}
```

10. Roadmap (sprint-based)

Sprint 0 (2-3 giorni): Boot

- Monorepo + docker compose (Postgres/Timescale, Redis, API, Engine, UI).
- Healthcheck end-to-end: UI -> API -> enqueue -> engine -> DB -> UI.
- Schema DB base + migrazioni.

Sprint 1 (1 settimana): IPS + Paper Portfolio

- IPS Wizard (UI) + versioning (API/DB).
- Creazione paper portfolio + transazioni paper.
- PAC engine v1 (contributi-only) + snapshot mensili.

Sprint 2 (1-2 settimane): Universe + ETF scoring

- Bucket ETF (equity_global, bond_hedged) e gestione candidati.
- Scoring ETF v1 + Top 3 per bucket con rationale.
- Proposta portafoglio iniziale (Single ETF core o 90/10).

Fase 2 (anno 3/4+): ETF + Azioni

- Security master azioni + fundamentals pipeline.
- Quality screener within-sector (ROIC/ROE, accruals, leverage, stability).
- Satellite azioni 10-20% con vincoli di concentrazione e ribilanciamento.

11. Migrazione su VPS (quando pronto)

La migrazione e' una replica del locale: stessa compose, ma con hardening (backup DB, secrets, reverse proxy, monitoring). Obiettivo: zero cambi architetturali, solo configurazione.

- Backup automatici Postgres (giornalieri) + retention.
- Reverse proxy (Caddy/Nginx) + TLS.
- Secrets via env file protetto o vault leggero.
- Observability: logging strutturato, metriche base, alert su job falliti.

Fine documento. Questo Codex e' pensato per essere riutilizzato come base di lavoro nel tuo IDE: contiene scope, moduli, contratti e roadmap. Aggiorna il numero di versione e conserva le vecchie versioni.