**AInvestorAgent 架构设计**

**1) 目标与原则**

* **目标**：用专业量化指标在**每周≤3次**频率给出组合建议，并可用历史数据做**低频回测**验证。
* **原则**：模块清晰、依赖最小、接口稳定；先跑通主链路，细节可渐进增强。

**2) 技术栈（建议）**

* **后端**：Python + FastAPI（REST），Pydantic（Schema）
* **数据存储**：**SQLite**（本地，SQLAlchemy ORM）
* **任务调度**：APScheduler（可选，遵守≤3次/周）
* **LLM**：GPT-5 / Claude（仅用于新闻摘要与情绪分类）
* **前端**：React + ECharts + Tailwind（深色专业风格）

说明：SQLite 足够支撑原型；未来需要并发/多用户时可平滑迁移到 Postgres。

**3) 模块划分（最小可用集）**

backend/

api/ # HTTP 路由与请求/响应模型

ingestion/ # 数据拉取：AlphaVantage(日线/概览)、news API(标题/摘要)

factors/ # 指标计算与标准化：价值/质量/动量/风险/情绪

scoring/ # 因子加权聚合 → 0~100 分 + version\_tag

portfolio/ # 组合构建：权重分配 + 约束(单票/行业/数量)

backtest/ # 周度调仓回测(≤3次/周)、净值/回撤/Sharpe

sentiment/ # 新闻清洗、LLM摘要与极性(-1~1)，轻量去重

storage/ # SQLAlchemy 模型与DAO（SQLite）

orchestrator/ # 流程编排：数据→因子→评分→组合→模拟

core/ # 配置、日志、限频(每周≤3次)

frontend/

pages/ Dashboard | Stock | Portfolio | Simulator

components/ 图表与卡片

**4) 数据模型（SQLite 表，精简且够用）**

* **symbols**：symbol(PK), name, exchange, sector, industry, currency
* **prices\_daily**：symbol, date(PK组合), open, high, low, close, adj\_close, volume
* **fundamentals**：symbol(PK), as\_of, pe, pb, roe, net\_margin, market\_cap, source
* **news\_raw**：id(PK), symbol, title, summary, url, source, published\_at
* **news\_scores**：id(PK), news\_id(FK), sentiment(-1..1), topic TEXT NULL
* **factors\_daily**：symbol, date, value, quality, momentum, risk, sentiment
* **scores\_daily**：symbol, date, score(0..100), weights JSON, version\_tag
* **portfolios**：portfolio\_id(PK), name, constraints JSON, created\_at
* **portfolio\_snapshots**：snapshot\_id(PK), portfolio\_id, date, holdings JSON, explain JSON
* **backtest\_results**：backtest\_id(PK), portfolio\_id, params JSON, date, nav, drawdown, benchmark\_nav
* **run\_control**：job\_name(PK), last\_run\_at *(限频用：≤3次/周)*

说明：JSON 字段存小对象（权重、参数、解释），SQLite 读写快速、足够灵活。

**5) 核心数据流（端到端）**

1. **Analyze(symbol)**  
   ingestion 拉或更新 → factors 计算（标准化/分位） → sentiment 近7–30天新闻打分 →  
   scoring 加权合成分数(+version\_tag) → 返回统一分析对象（行情+因子+分数+情绪）。
2. **Propose Portfolio(symbols[])**  
   过滤（流动性/市值/分数阈值）→ 初始权重(按分数) → 约束修正（单票≤30%、行业≤50%、持仓5–15）→  
   输出权重+入选理由（因子贡献TOP2）。
3. **Backtest(组合规则/权重, 窗口)**  
   周度调仓（**最多3次/周**），套用历史日线 → 计算 NAV/回撤/Sharpe/胜率/换手 → 存 backtest\_results。  
   可与基准（如 SPY）对比。

**6) 稳定 API（最少但覆盖主链路）**

* GET /symbols?q=：搜索代码
* GET /prices/{symbol}?range=1M|3M|1Y：价格序列
* GET /fundamentals/{symbol}：基本面快照
* GET /news/{symbol}?days=7|30：新闻标题/摘要与情绪
* POST /analyze/{symbol}：**统一分析**（行情+因子+分数+情绪）
* POST /portfolio/propose：输入 symbols[] + 可选 weights\_config/constraints → 返回权重+解释
* POST /backtest/run：输入组合/窗口/成本 → 返回净值曲线与指标

**7) 因子与评分（契约）**

* **标准化**：对“越低越好”（PE、波动）取反向分；统一到 0–1 或 z-score。
* **基线权重（可配置）**：value 0.25 | quality 0.20 | momentum 0.35 | sentiment 0.20
* **得分**：score = 100 \* Σ(w\_i \* factor\_i) → 落表 scores\_daily，记录 version\_tag。
* **限频**：run\_control 记录策略运行时间，调度前校验**每周≤3次**。

**8) 前端信息架构（可视化最小集）**

* **Stock**：行情卡片、价格折线(MA5/20/60)、雷达(价值/质量/动量/风险/情绪)、新闻情绪时间轴
* **Portfolio**：权重饼图、行业条形图、持仓表(分数/理由/权重)
* **Simulator**：净值曲线、最大回撤图、指标表（年化/Sharpe/胜率/换手）
* **Dashboard**：最近一次组合快照 + 与基准对比的累计收益

统一深色主题；涨绿跌红；所有图表标注“数据时间戳/来源/版本”。

**9) LLM 边界（可控与可替换）**

* **只用于**：新闻摘要与情绪分类（短上下文、结构化输出）。
* **不用于**：权重/信号直接决策。
* **可替换**：后续可用更轻量的情绪词典或本地模型替代。

**10) 可靠性与扩展点**

* **可靠性**：缺数据时使用上次有效数据（Graceful Degradation）；所有外部调用超时与重试；关键步骤日志。
* **可迁移**：将 SQLite 的 ORM 模型保持无方言特性 → 未来换 Postgres 无需改业务层。
* **可扩展**：新增因子=新增列/函数；优化器可替换（线性→二次）；回测细节逐步增强（费率/停牌/分红）。

**基于多智能体的更新：**

**（A）总体架构图更新**

* 三层结构：Agents / Orchestrator / Shared Services。
* API Gateway（FastAPI）对外统一暴露：/api/v1/ingest, /research, /backtest, /portfolio, /report, /orchestrator/dispatch, /trace/{id}, /metrics。

**（B）消息协议（建议落地到 docs/protocols/ 下，但先在设计文档中描述）**

{

"trace\_id": "uuid",

"session\_id": "string",

"user\_id": "string",

"scene": "ingest|research|backtest|portfolio|report",

"agent": "ingestor|cleaner|researcher|backtest|risk|pm|report",

"payload": { "…与该Agent职责相关的参数…" },

"tool\_hints": { "provider": "alpha\_vantage|yfinance|newsapi|llm", "timeout\_s": 30 },

"meta": { "ts": "ISO8601", "retry": 0 }

}

**（C）编排策略**

* 串行链：Ingest → Clean → Research → Backtest → Risk → PM → Report
* 并行节点：不同alpha/feature组的研究与回测可并行，Orchestrator聚合最优解。
* 重试与降级：某外部源（如AlphaVantage）失败则自动切换备用源（如yfinance），并标记到trace。

**（D）可观测性**

* /trace/{trace\_id}：返回该次任务全链路时序与各Agent入参/出参摘要（脱敏）。
* /metrics：QPS、成功率、耗时直方图、数据缺口率、回测覆盖率等。

**（E）数据流**

* 原有 SQLite/DB 表不动；新增最小表以支持多智能体轨迹与审计（建议后续在你同意后再建表）：
  + agent\_runs(trace\_id, agent, start\_ts, end\_ts, status, input\_hash, output\_hash, error)
  + orchestrations(trace\_id, plan, dag\_json, status)