Систем за Анализа на Берзански Податоци

Овој систем претставува реално-временска и историска платформа за анализа на берзански податоци. Ги прибира, обработува, предвидува и прикажува податоците за берзански акции користејќи микросервисна архитектура. Сите сервиси се извршуваат во Docker контејнери и комуницираат преку PostgreSQL, Kafka и Flask.

Структура на Проектот

Структура на Проектот:

```
.
├── Data_Ingestion_Service.py         # Прибирање на историски податоци преку yFinance
├── Data_Analysis_Service.py          # Генерирање прогнози со Prophet
├── Dashboard.py                      # Flask сервер за API и dashboard
├── Real-Time_Processing_Service.py   # Зима Real-Time податоци од берзата
├── graph.html                        # Фронтенд HTML за dashboard
├── ticker_list.csv                   # CSV со листа на тикери
├── docker-compose.yml                # Docker Compose конфигурација
├── Dockerfile                        # Docker image за апликацијата
├── templates/
│   └── graph.html                    # HTML код за изглед на страницата и прикажување на графови
```

Опис на Сервисите

1. Сервис за Прибирање Податоци (Ingestion)
Фајл: Data_Ingestion_Service.py

Функција: Автоматски ги симнува последните 1 ден податоци за сите тикери од ticker_list.csv преку yfinance и ги зачувува во stock_history табелата во PostgreSQL.

Закажано Време: Секој ден во 00:20–00:25 по европско време (Скопје).

2. Сервис за Анализа на Податоци
Фајл: Data_Analysis_Service.py

Функција: Користи Prophet за да направи 7-дневна прогноза за цените на акциите.
База: Чита од stock_history, пишува во stock_analysis.
Извршување: Автоматски секој ден во 04:00 – 04:10.

3. Dashboard Сервис
Фајл: Dashboard.py

Функција: Flask сервер кој обезбедува API за:
- Историски податоци (/historical-data)
- Последна цена (/last-price)
- Живи податоци преку Server-Sent Events (/live-data)
- Прогноза (/forecast-data)

Интерфејс: graph.html – модерен и интерактивен frontend со Plotly график.

4. Kafka + Real-Time Сервис
Фајл: Real-Time_Processing_Service.py

Функција:
- Producer: Ги прибира податоците во живо преку yfinance и ги испраќа во Kafka тема (stock-data)
- Consumer (Spark): Чита податоци од Kafka, ги агрегира на 10 секунди и ги запишува во табелата

stock_realtime во PostgreSQL
- Автоматски ја чисти табелата stock_realtime секој ден во 23:55

Технологии: confluent-kafka, Apache Spark Structured Streaming, PostgreSQL
Работно време: секој ден од 14:30 до 21:00 (локално време – Европа/Скопје)

Функционалности:
- Rate-limit менаџмент (retry ако има грешки)
- Паралелна обработка на податоци со threading
- Автоматско стартување и стопирање на сервисот во соодветни времиња

Бази на Податоци

Бази на Податоци (PostgreSQL Табели):
- stock_history: историски податоци (date, open, close, volume...)
- stock_realtime: реално-временски податоци (timestamp, close_price)
- stock_analysis: прогнозирани вредности (yhat, yhat_upper/lower)
(Сите табели се хостирани на Azure)

Веб Интерфејс

Веб Интерфејс (graph.html):
- Селектор за тикер и временски период
- Приказ на:
  • Историски податоци (сини линии)
  • Живи податоци (портокалова линија)
  • Прогноза (зелена линија)
  • Последна цена (црвена точка)
  • Индикатор за статус на пазарот: LIVE, Loading, Market Closed

Docker Compose Конфигурација

Docker Compose Конфигурација:
Вклучува следниве сервиси:
- postgres: база на податоци (порт 5433 → 5432)
- kafka, zookeeper: за real-time податоци
- dashboard: Flask сервер на порт 8050
- ingestion: сервис за прибирање
- analysis: сервис за прогноза
- realtime: сервис за Kafka обработка

Како да стартуваш

1. Инсталирај Docker и Docker Compose
- За Windows/Mac: симни од https://www.docker.com/products/docker-desktop
- За Linux (пример Ubuntu):
  sudo apt install docker.io docker-compose

2. Клонирај го репозиториумот
```
git clone https://github.com/Jabolce/DASprojcet.git
cd DASprojcet/DASproekt
```

3. Градба и стартување на сите сервиси
```
docker-compose up --build
```

4. Провери дали сервисите се активни
- Dashboard треба да е достапен на: http://localhost:8050
- PostgreSQL работи на порт 5433
- Kafka/Zookeeper и останатите сервиси стартуваат автоматски

5. (Опционално) Провери логи
docker-compose logs -f

6. (Опционално) Рестартирај ги сервисите
docker-compose down
docker-compose up --build