RELAZIONE – HOMEWORK 1 - Prova in Itinere DSBD aa 2024-2025

Studenti:
Bonafede Salvatore Luca (1000067612)
Bontempo Gaetano (1000067613)

ABSTRACT

Per l'Homework 1, è stato sviluppato un sistema distribuito progettato per la raccolta e l'elaborazione di dati finanziari utilizzando la libreria **yfinance**.

Il sistema è costituito da tre componenti principali, gestiti tramite container Docker separati:

1. Un database **MySQL**, che memorizza informazioni sugli utenti e sui dati finanziari.
2. Un server **gRPC**, responsabile della gestione delle richieste degli utenti.
3. Un **Data Collector**, che raccoglie, per ogni utente registrato, i ticker azionari e, tramite **yfinance**, recupera il valore dei relativi titoli azionari.

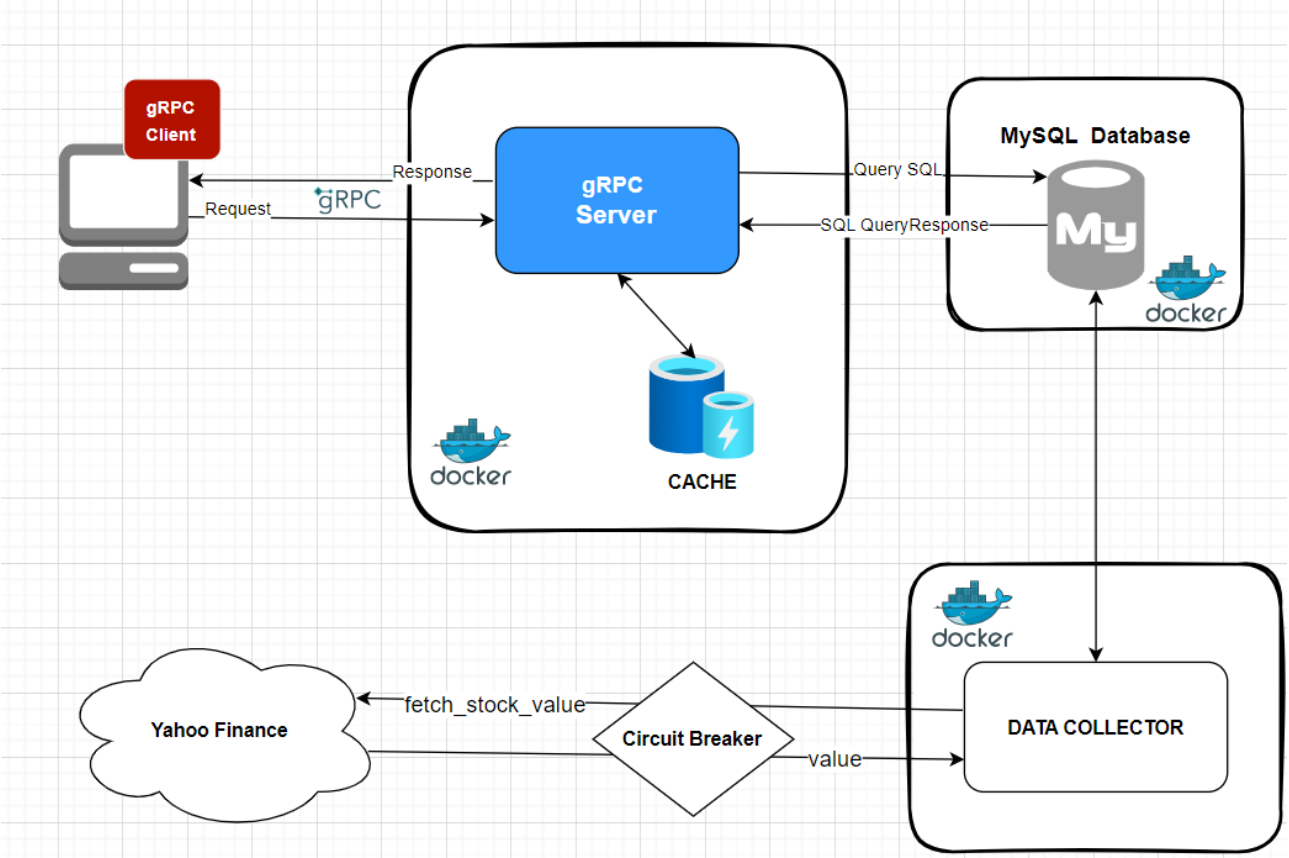
La politica **"at-most-once",** per la registrazione e l'aggiornamento degli utenti, è stata implementata attraverso una cache che memorizza identificatori univoci generati tramite hash. Per la registrazione, viene utilizzato l'hash dell'indirizzo e-mail dell'utente, mentre per l'aggiornamento vengono concatenati e-mail e nuovo ticker, generando l'hash. Questi hash vengono salvati nel database, e in caso di riavvio del server, mediante due query al database, le informazioni vengono ripristinate nella memoria cache, evitando la gestione di richieste duplicate. Sono inoltre presenti controlli che impediscono la gestione di richieste duplicate nel caso in cui la cache non dovesse funzionare per un qualsiasi motivo.

Per le operazioni eseguite dal **Data Collector**, è stato implementato un **Circuit Breaker** che, in caso di fallimento durante il recupero dei dati, incrementa un contatore di errori. Quando il numero di errori supera un limite prefissato, tutte le richieste vengono bloccate per un periodo definito. Al termine di tale timeout, il circuito entra nello stato "Half Open" e tenta di eseguire una chiamata:

- Se la chiamata ha esito positivo, il circuito si resetta e torna allo stato "Closed".
- In caso contrario, lo stato torna a "Open", prolungando il blocco.

Per garantire la robustezza del **Circuit Breaker**, è stato adottato un meccanismo di sincronizzazione basato su **thread locking**, evitando conflitti quando più richieste vengono gestite contemporaneamente. Infine, per ottimizzare le operazioni, è stato introdotto un controllo sull'orario di apertura del mercato azionario. Gli aggiornamenti vengono eseguiti solo durante gli orari in cui il mercato è aperto, riducendo così le richieste non necessarie. Inoltre, per evitare computazioni extra, nel caso in cui più utenti siano interessati allo stesso ticker, il Data Collector, effettua una sola chiamata all'API di yfinance (per ticker) e aggiorna il database per tutti gli utenti interessati.

Di seguito è riportato un diagramma che illustra i micro-servizi coinvolti e le loro interazioni:



Lista delle API implementate

- LoginUser

Request: LoginUserRequest (email)
Response: UserResponse (success, message)

- RegisterUser

Request: RegisterUserRequest (email, ticker, message_id)
Response: UserResponse (success, message)

- UpdateUser

Request: UpdateUserRequest (email, ticker, message_id)
Response: UserResponse (success, message)

- DeleteUser

Request: DeleteUserRequest (email)
Response: UserResponse (success, message)

- GetTickerValue

Request: GetTickerRequest (email)
Response: TickerResponse (success, message, value)

- GetTickerAverage

Request: GetTickerAverageRequest (email, lastXValues)
Response: TickerResponse (success, message, value)