UNIVERSITÀ
degli STUDI
di CATANIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

DISTRIBUTED SYSTEMS AND BIG DATA

Andrea Azzaro • Leonardo Russo

RELAZIONE HOMEWORK 1

Anno Accademico 2024/2025

Abstract
Questo progetto consiste in un sistema distribuito per la gestione di utenti e il recupero di dati finanziari utilizzando la libreria **yfinance**. L'architettura è basata su un approccio a microservizi, che consente una maggiore modularità e scalabilità. Ogni componente del sistema è containerizzato tramite **Docker**, garantendone portabilità, e l'orchestrazione avviene attraverso **Docker Compose**, semplificando la gestione dell'intero sistema.
Il sistema comprende diversi moduli principali: un **server gRPC** che offre funzionalità quali registrazione, aggiornamento, cancellazione di utenti e gestione di dati azionari; un **Data Collector**, responsabile dell'interrogazione periodica dei dati azionari, protetto da un Circuit Breaker per aumentarne la robustezza; un database **MySQL**, utilizzato per la persistenza e la gestione delle informazioni.
La comunicazione tra client e server, per ciò che concerne la gestione degli utenti, avviene seguendo la politica "at-most-once", un approccio che garantisce l'idempotenza delle operazioni, evitando duplicazioni o inconsistenze nei dati. Questo meccanismo è fondamentale per mantenere l'integrità delle transazioni, soprattutto in contesti distribuiti. Mentre, per il recupero del valore azionario da parte di un utente, la comunicazione avviene secondo la politica "at least once", essendo tali operazioni già idempotenti.

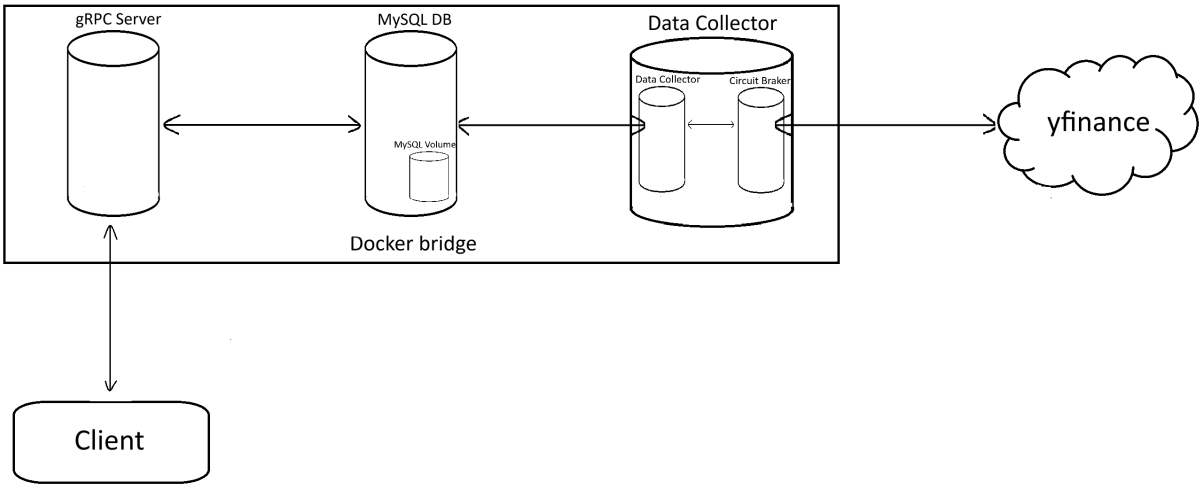
Architettura del Sistema
Il sistema è composto da tre principali componenti containerizzati, mentre il client esegue direttamente nel sistema host:

1. **Database MySQL**:
    - Memorizza le informazioni degli utenti e i dati recuperati dal servizio Data Collector. In particolare contiene due tabelle: la prima "users" memorizza l'email e il ticker azionario associato all'utente. L'email viene memorizzata come chiave primaria in modo tale da avere un'ulteriore protezione dalla presenza di duplicati. La seconda tabella "data" contiene il ticker azionario, il valore associato a tale ticker e il timestamp, che viene generato automaticamente dal database.
    - La persistenza dei dati è garantita attraverso l'uso di un Docker Volume.
2. **Data Collector**:
    - Include l'implementazione di un **Circuit Breaker** per gestire errori e ritardi nelle risposte.
    - Recupera ogni 30 minuti i dati azionari tramite **yfinance** per ogni utente registrato.
3. **Server gRPC**:
    - Implementa le seguenti API:
        - **Registrazione Utenti**: Inserisce nuovi utenti nel database.
        - **Aggiornamento Utenti**: Aggiorna i ticker azionari associati agli utenti.
        - **Cancellazione Utenti**: Rimuove utenti dal database.
        - **Recupero Valori Azionari**: Fornisce l'ultimo valore azionario disponibile o la media degli ultimi X valori.
    - Le prime tre API seguono la semantica "at-most-once" per prevenire duplicati o inconsistenze, mentre l'ultima utilizza "at-least-once" dato

che comprende soltanto operazioni di tipo “get” e quindi non possono causare i problemi sopra riportati.
    - Utilizziamo un port-mapping in modo tale da renderlo visibile dall’esterno
4. **Client gRPC**:
    - Esegue in un sistema host e comunica con il server tramite port-mapping.
    - Permette di testare tutte le funzionalità esposte dal server gRPC.

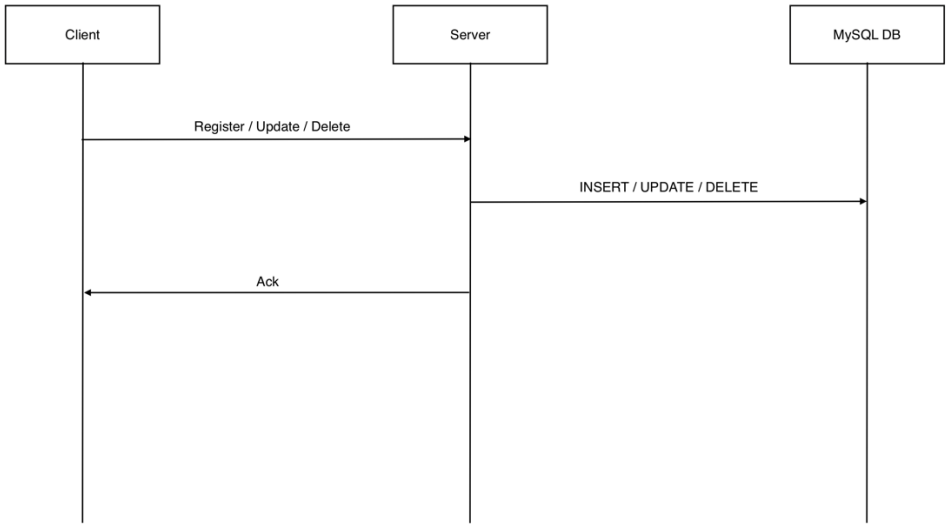
I tre container sono inseriti all’interno del Docker Biridge, creato automaticamente dal Docker Compose.

Diagramma Architetturale

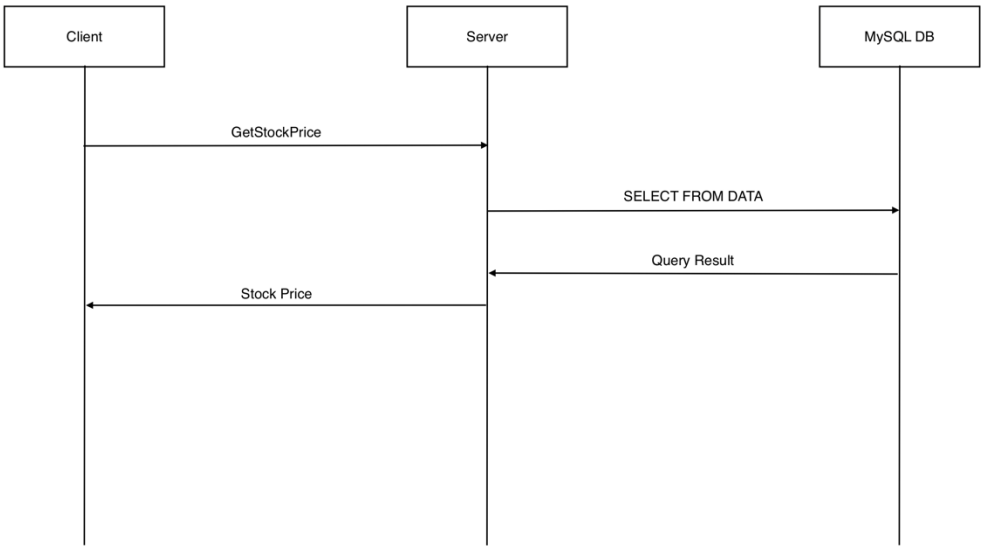


Interazioni tra Componenti

- Gestione Utenti
    - Il **Client gRPC** invia richieste al **Server gRPC** per creare, modificare o eliminare il proprio account.
    - Il **Server gRPC** riceve la richiesta e interagisce col **database**



- Stock Service
    - Il **Client gRPC** richiede al server il valore più aggiornato di un ticker o la media degli ultimi N valori
    - Il **Server gRPC**, in base alla richiesta effettua un'oppurtuna query al **Database**
    - Il **Database** ritorna al **Server gRPC** il valore richiesto
    - Il **Server gRPC** ritorna al **Client gRPC** il valore richiesto



- Data Collector
    - Il **Data Collector**, operando in modo indipendente, legge periodicamente dal database tutti I ticker associati a ciascun utente
    - Il **Data Collector** interagisce con il **Circuit Braker** per recuperare I dati azionari da **yfinance**
    - Il **Circuit Braker**, se chiuso, inoltra la richiesta a **yfinance**
    - Il **Data Collector**, ottenuti I dati richiesti, li memorizza nel **Database**