Homework 1 - Prova in itinere DSBD 2024/25

Matteo Santanocito - 1000069999

November 26, 2024

Contents

1 Abstract

Questo documento presenta l'architettura e le funzionalità di un sistema di gestione finanziaria basato su microservizi. Il progetto ha l'obiettivo di gestire in modo efficiente i dati degli utenti e le loro informazioni finanziarie. In generale il sistema è composto da cinque componenti principali: Client, Server gRPC, Data Collector, Database PostgreSQL e Data Cleaner.

Il cuore del sistema è il Server gRPC che funge da nucleo centrale e permette la gestione delle operazioni CRUD (Create, Read, Update, Delete) sugli account degli utenti e il recupero dei dati finanziari per ogni utente. Tutte le operazioni che comportano una modifica dello stato del sistema sono implementate con una politica **at-most-once**. Questo significa che ogni richiesta inviata dal client al server viene elaborata **al più una sola volta,** evitando duplicazioni anche in presenza di ritentativi dovuti a errori temporanei.

Il sistema è alimentato dal Data Collector, un microservizio dedicato alla raccolta periodica di dati finanziari utilizzando la libreria yfinance per ottenere informazioni aggiornate sui titoli. Ogni chiamata alla libreria yfinance è protetta attraverso il Circuit Breaker, fondamentale per la gestione di errori e/o ritardi nelle risposte.

Tutti i valori finanziari e le informazioni degli utenti vengono memorizzati in un database relazionale implementato attraverso PostgreSQL.

Per ottimizzare lo spazio di archiviazione, il microservizio Data Cleaner esegue ogni 24 ore scansioni per identificare i ticker inattivi — quelli non più associati a nessun utente — e rimuove i relativi dati finanziari dal database. Questo processo di pulizia automatizzato garantisce che il sistema rimanga efficiente e privo di dati ridondanti.

Complessivamente, questa architettura a microservizi mira a ottenere scalabilità, resilienza e manutenibilità, permettendo a ciascun componente di operare in modo indipendente. L'integrazione di meccanismi di caching e un robusto sistema di logging migliora ulteriormente le prestazioni e l'affidabilità del sistema.

2 Componenti del Sistema

2.1 Client gRPC

Il **Client gRPC** è l'interfaccia utente che consente agli utenti finali di interagire con il sistema tramite chiamate RPC (Remote Procedure Call). Può essere implementato come un'applicazione desktop, web o mobile che utilizza gRPC per comunicare con il Server gRPC.

- **Interazione con l'Utente**
  - Fornisce un'interfaccia intuitiva per eseguire operazioni come registrazione, login, aggiornamento dei dettagli, recupero dei dati finanziari, ecc.
- **Invio delle Richieste:**
  - Invia richieste al Server gRPC per eseguire operazioni CRUD e per ottenere dati finanziari.
- **Gestione delle Risposte**
  - Riceve e visualizza le risposte dal Server gRPC, inclusi messaggi di successo o errori.

2.2 Server gRPC

Il **Server gRPC** è il fulcro del sistema, responsabile della gestione delle operazioni CRUD (Create, Read, Update, Delete) sugli utenti e della fornitura di servizi per il recupero dei dati finanziari. Utilizza gRPC (Google Remote Procedure Call) per consentire una comunicazione efficiente e ad alte prestazioni tra i vari componenti del sistema e i client.

- **Gestione Utenti**
  - **Creazione**: Registrazione di nuovi utenti.
  - **Lettura**: Recupero delle informazioni degli utenti.
  - **Aggiornamento**: Modifica dei dettagli degli utenti, come il ticker associato.
  - **Cancellazione**: Rimozione degli utenti dal sistema.
- **Servizi Finanziari**
  - **Recupero Ultimo Valore**: Fornisce l'ultimo valore disponibile per un ticker specifico associato a un utente.
  - **Calcolo Media Valori**: Calcola la media degli ultimi X valori per un ticker specifico.
- **Autenticazione e Autorizzazione:**
  - Gestisce il login degli utenti e la generazione di token di autenticazione per garantire che solo gli utenti autorizzati possano accedere ai servizi.
- **Caching:**
  - Utilizza TTLCache per migliorare le prestazioni delle operazioni frequenti, riducendo il carico sul database.

2.3 Data Collector

Il **Data Collector** è un microservizio dedicato alla raccolta periodica di dati finanziari. Utilizza librerie come `yfinance` per ottenere dati aggiornati dai mercati finanziari e li inserisce nel database PostgreSQL per essere utilizzati dal Server gRPC e da altri servizi.

- **Raccolta Dati**
  - Connessione a fonti di dati finanziari (Yahoo Finance tramite API `yfinance`) per il recupero dei dati finanziari come prezzi delle azioni, volumi, ecc.
- **Inserimento Dati**
  - Inserimento dei dati raccolti nella tabella `financial_data` del database.
- **Schedulazione**
  - Operazioni periodiche (es. 3 minuti) per garantire che i dati siano sempre aggiornati.
- **Gestione degli Errori**
  - Gestione delle eccezioni durante la raccolta e l'inserimento dei dati.
  - Implementazione di meccanismi di retry in caso di fallimenti temporanei.

2.4 Database - PostgreSQL

Il **Database PostgreSQL** è il sistema di gestione di database relazionali (RDBMS) che memorizza tutte le informazioni necessarie per il funzionamento del sistema, inclusi i dati degli utenti e i dati finanziari raccolti.

- **Memorizzazione Dati Utenti**
  - Conserva le informazioni sugli utenti, come email e ticker associati.
- **Memorizzazione Dati Finanziari**
  - Archivia i dati finanziari raccolti dal Data Collector, come prezzi delle azioni e timestamp.

2.5 Data Cleaner

Il **Data Cleaner** è un microservizio incaricato di mantenere il database pulito e aggiornato. Ogni 24 ore, verifica quali ticker sono ancora attivi nella tabella `users` e rimuove i dati finanziari associati ai ticker che non sono più in uso.

- **Verifica dei Ticker Attivi**
  - Scansiona la tabella `users` per identificare tutti i ticker attualmente in uso.
- **Eliminazione dei Dati Obsoleti**
  - Rimuove i record dalla tabella `financial_data` che non sono più associati a nessun ticker attivo.

- **Schedulazione**
  - Esegue queste operazioni automaticamente ogni 24 ore per garantire che il database non si riempia con dati inutili.

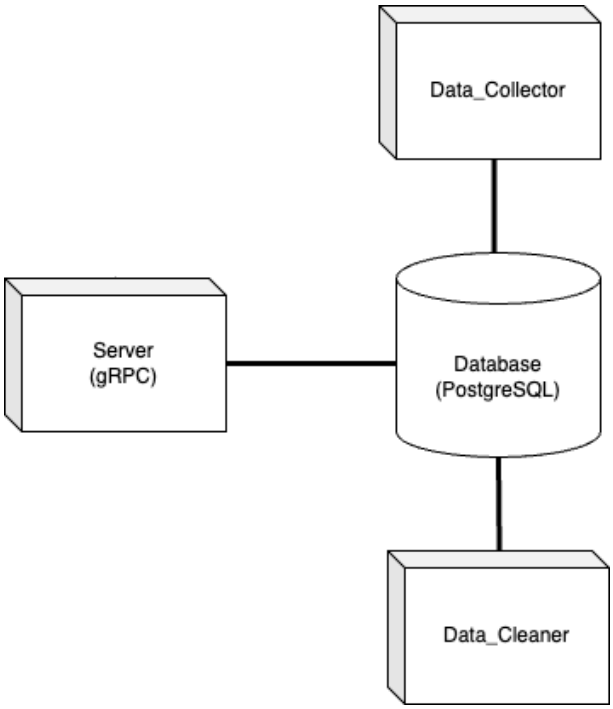

Figure 1: Diagramma architetturale dei microservizi

2.6 Diagramma Interazioni

2.6.1 Client - Server gRPC

Il **Client** invia richieste al **Server gRPC** al fine di permettere operazioni come la registrazione, l'aggiornamento del ticker, la cancellazione dell'account, la possibilità di ottenere l'ultimo valore del ticker e il calcolo della media di n valori del ticker.

2.6.2 Server gRPC - Database

Il server comunica con il database per le operazioni CRUD (Creazione di un nuovo utente, modifica del ticker per quell'utente, cancellazione utente), recupera l'ultimo valore del ticker registrato per quell'utente e calcola la media degli ultimi n valori associati al ticker (dove n è un numero passato dall'utente).

2.6.3 Data Collector-yfinance

Il DataCollector sfrutta la libreria yfinance per reciperare gli ultimi valori finanziari per ciascun ticker. Ogni chiamata a yfinance è protetta dal **CircuitBraker**, un sistema che permette la gestione di errori e ritardi nella richiesta.

2.6.4 DataCollector-Database

Il DataCollector accede al Database, guarda nella tabella degli utenti registrati e ottiene i ticker azionari. Sfrutta la libreria yfinance per ottenere i valori per quei ticker e li memorizza nel database. Questa operazione avverrà in modo parallelo agli altri microservizi ogni 3 minuti.

2.6.5 DataCleaner-Database

Il Cleaner accede al Database, guarda nella tabella degli utenti registrati e ottiene i ticker azionari. Accede al database, guarda nella tabella con i valori finanziari ed elimina tutti i ticker azionari che non sono più attivi. Questa operazione avverrà in modo parallelo agli altri microservizi ogni 24 ore.

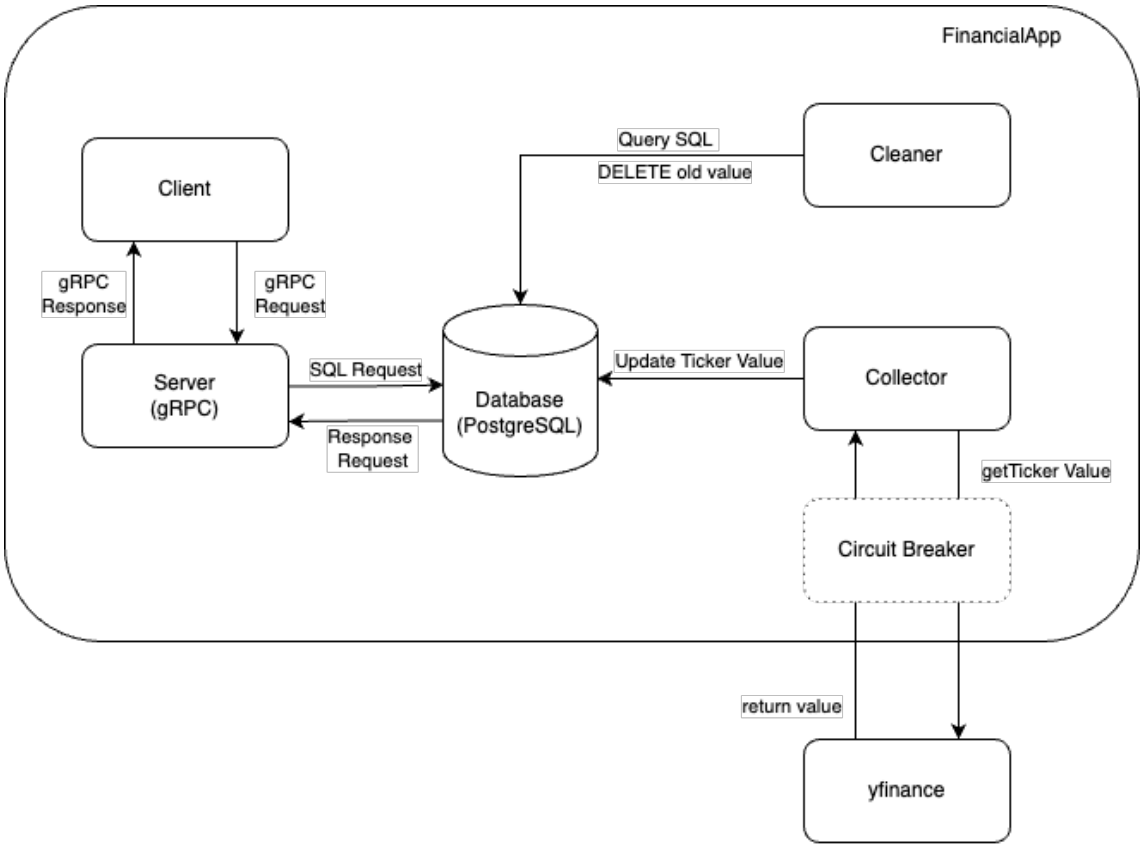

Figure 2: Diagramma che mostra le interazioni, sia tra componenti dell'applicazione che tra l'applicazione e il mondo esterno.

3 Panoramica delle API

Le API implementate all'interno del sistema consentono di:

- Registrare nuovi utenti.
- Effettuare il login degli utenti esistenti.
- Modificare il ticker (simboli di borsa) dell'utente.
- Recuperare l'ultimo valore del ticker dell'utente.
- Calcolare la media di n valori finanziari per il ticker specifico dell'utente.

3.0.1 Dettaglio delle API

Metodo	Input Richiesta	Risposta
`RegisterUser()`	• `request_id` (string) • `email` (string) • `ticker` (string)	• `message` (string)
`LoginUser()`	• `request_id` (string) • `email` (string)	• `message` (string) • `success` (bool)
`UpdateUser()`	• `request_id` (string) • `email` (string) • `ticker` (string)	• `message` (string)
`DeleteUser()`	• `request_id` (string) • `email` (string)	• `message` (string)
`GetLastValue()`	• `email` (string)	• `email` (string) • `ticker` (string) • `value` (double) • `timestamp` (string)
`GetAverageValue()`	• `email` (string) • `count` (int32)	• `email` (string) • `ticker` (string) • `average_value` (double)