HOMEWORK 2

Prova in itinere DSBD aa 2024-2025

Giorgio Di Bartolo, matricola 1000001998
Silvia Garozzo, matricola: 1000069223

INDICE

ABSTRACT

Il presente progetto descrive un sistema avanzato per la gestione degli utenti e l'elaborazione di dati finanziari, sviluppato con un'architettura a micro-servizi containerizzati utilizzando Docker e coordinati attraverso un Docker Compose.

Al centro del sistema vi è un server gRPC che funge da interfaccia tra gli utenti e il database. Il server consente operazioni come registrazione, aggiornamento, eliminazione degli utenti e recupero dei dati finanziari. Per garantire la coerenza e prevenire duplicazioni o inconsistenze, tutte le operazioni di modifica dello stato del sistema seguono una politica **at-most-once**, assicurando l'idempotenza.

Il sistema adotta il pattern **CQRS (Command Query Responsibility Segregation)** per separare la gestione delle operazioni di scrittura (*command*) da quelle di lettura (*query*) per migliorare la scalabilità e la manutenibilità.

Un componente autonomo, il *DataCollector*, esegue ciclicamente il recupero dei dati finanziari. Questo modulo legge la lista degli utenti dal database e recupera, tramite la libreria *yfinance*, i dati finanziari relativi ai ticker azionari di interesse. Ogni richiesta verso la libreria è protetta da un *Circuit Breaker*, necessario a gestire eventuali errori o ritardi nelle risposte.

Per notificare gli utenti, il sistema invia email ogni volta che il valore di un ticker supera o scende al di sotto di una soglia definita. Questo meccanismo asincrono è implementato attraverso **Apache Kafka** che gestisce lo scambio di messaggi su due differenti topic: *to-alert-system* e *to-notifier*.

I dati finanziari e le informazioni degli utenti sono memorizzati in un database relazionale **PostgreSQL**. Per facilitare il testing e la verifica delle funzionalità del sistema, è stato sviluppato un client specifico.

Il sistema è progettato per garantire robustezza e affidabilità grazie ad una gestione efficace degli errori e l'implementazione di meccanismi di protezione avanzati.

DIAGRAMMA DEI MICRO-SERVIZI

Lista delle funzionalità di ognuno dei componenti:

Server gRPC

Gestisce le richieste degli utenti per la registrazione, l'aggiornamento, la cancellazione e il recupero dei dati finanziari.

DataCollector

Recupera periodicamente i dati finanziari dalla libreria *yfinance* relativi ai ticker azionari degli utenti registrati. Inoltre, invia un messaggio al topic *to-alert-system* ogni volta che una raccolta dati è completata.

Database

Memorizza le informazioni degli utenti e i dati finanziari.

Zookeeper

Coordina e gestisce Kafka.

Kafka

Funge da broker e gestisce la comunicazione tra i componenti del sistema.

AlertSystem

Legge i messaggi dal topic *to-alert-system* e, dopo aver scandito il database, invia messaggi al topic *to-notifier*.

AlertNotifierSystem

Legge i messaggi dal topic *to-notifier* e invia email agli utenti che necessitano di essere notificati.

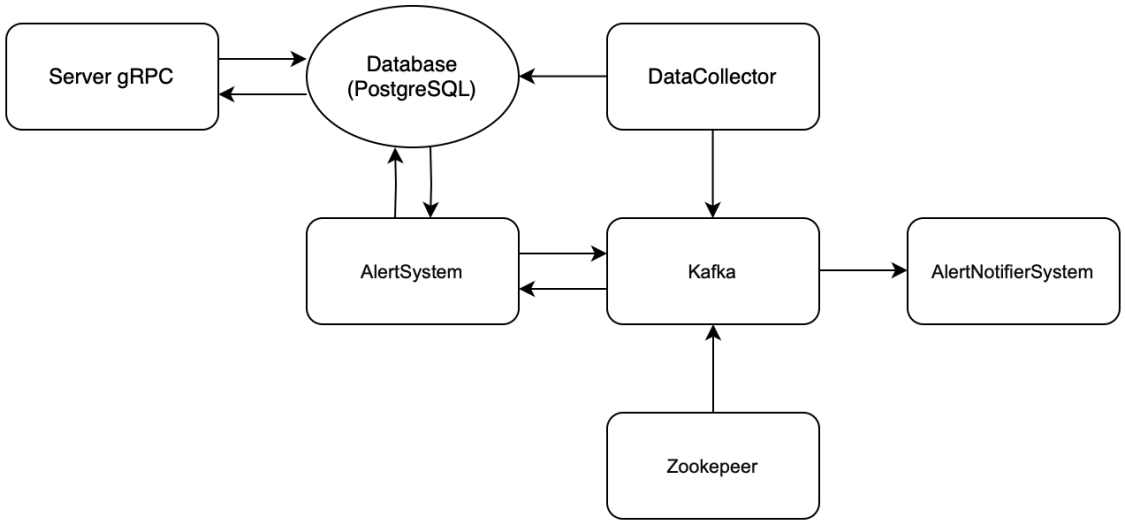

DIAGRAMMA DELLE INTERAZIONI

Utente - Client:

L'utente finale interagisce con il Client per inviare richieste al Server gRPC, includendo operazioni come registrazione, aggiornamento o eliminazione di un utente, oltre alla possibilità di ottenere dati finanziari.

Client - Server gRPC:

Il Client invia richieste al Server gRPC tramite chiamate gRPC. Il Server elabora tali richieste, interagisce con il Database e invia risposte adeguate.

Server gRPC - Database:

Il Server comunica con il Database per le operazioni di gestione degli utenti (registrazione, aggiornamento ed eliminazione) e recupero dei dati finanziari.

DataCollector - yfinance:

Per ciascun ticker, il DataCollector utilizza la libreria yfinance per ottenere gli ultimi dati finanziari. Tutte le chiamate a yfinance sono protette dal Circuit Breaker per gestire errori o ritardi.

DataCollector - Database:

Il DataCollector accede al Database per ottenere dalla lista degli utenti registrati i rispettivi ticker azionari. Utilizzando la libreria yfinance, il DataCollector acquisisce i dati finanziari aggiornati per ciascun ticker e li memorizza nel Database.

Zookeeper - Kafka:

Zookeeper avvia Kafka e si occupa di gestire e coordinare il cluster.

DataCollector - Kafka (to-alert-system):

Dopo aver aggiornato i dati relativi ai ticker, il DataCollector invia un messaggio sul topic to-alert-system per notificare che la fase di aggiornamento è stata completata.

Kafka (to-alert-system) - AlertSystem:

Il componente AlertSystem si occupa di leggere i messaggi dal topic to-alert-system e procede ad una scansione del Database.

AlertSystem - Database:

Il componente AlertSystem scandisce il Database per confrontare i nuovi valori aggiornati dei ticker con le soglie (minima e/o massima) di ciascun utente.

AlertSystem - Kafka (to-notifier):

In caso di superamento di almeno una delle due soglie, il componente AlertSystem invia un messaggio sul topic to-notifier contenente i seguenti parametri: email, ticker, condizione di superamento, soglia e prezzo del ticker.

Kafka (to-notifier) - AlertNotifierSystem:

Il componente AlertNotifierSystem legge i messaggi dal topic to-notifier e provvede ad inviare una mail con i parametri contenuti in ogni messaggio (destinatario: email; oggetto: ticker; testo email: condizione di superamento, soglia e prezzo del ticker).