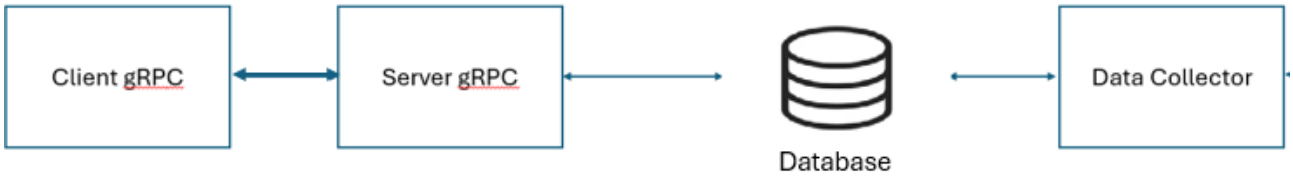
Relazione homework 1

Componenti del gruppo: Giorgia Grazia Mucciarella, Angelo Barbasola.
Il progetto si propone di sviluppare un sistema distribuito che consente la gestione e l'elaborazione di dati finanziari tramite un'architettura basata su microservizi.
Il server gRPC interagisce con un database per gestire le operazioni sugli utenti, come la registrazione, l'aggiornamento e la cancellazione.
Un componente separato, il DataCollector, recupera i dati finanziari da yfinance e li memorizza nel database; le chiamate a yfinance sono protette da un circuit breaker.
Il server gRPC e il Data Collector sono sviluppati come container docker e gestiti con docker compose.
Di seguito è riportato lo schema architetturale del progetto.



Il client è implementato come un'interfaccia a riga di comando che interagisce con il server gRPC.
Ogni funzionalità è mappata su un metodo specifico che invia richieste al server e gestisce le risposte.
L'interfaccia è basata su un menu ciclico, l'utente sceglie un'opzione da 1 a 6 e il ciclo continua finché l'utente non sceglie di uscire.

```
 Funzionalità di gestione degli utenti
1: Registrare un nuovo utente
2: Aggiornare il ticker di un utente
3: Cancellare un utente
4: Recuperare l'ultimo valore del ticker
5: Calcolare la media degli ultimi X valori
6: Esci
Scegli un'opzione:
```

Nel file server.py sono state implementate le diverse funzionalità relative alla registrazione dell'utente, aggiornamento del ticker dell'utente, cancellazione di un utente, il recupero dell'ultimo valore del titolo azionario scaricato da yfinance per l'utente specifico, calcolo della media degli ultimi X valori del titolo azionario per un utente.
Le funzionalità di registrazione dell'utente e l'aggiornamento del ticker di un utente sono implementate con la politica at most once: è stata utilizzata una cache per memorizzare le risposte inviate dal server per le specifiche richieste; prima di elaborare una nuova richiesta, il server verifica nella cache se esiste già una risposta associata all'identificatore unico della richiesta, in questo caso l'email dell'utente e se la risposta esiste il server restituisce il risultato memorizzato senza eseguire di nuovo l'operazione.
Per il progetto è stato utilizzato il database relazionale MySQL, insieme a XAMPP.
È stata utilizzata una tabella users che memorizza le informazioni degli utenti registrati ed è costituita da email (chiave primaria) e ticker. I dati finanziari sono memorizzati nella tabella financial_data costituita dal ticker, timestamp e value e la chiave primaria è la coppia ticker e timestamp.

email	ticker
abccc@gmail.com	AAPL
ciao@gmail.com	AMZN
esempio@hotmail.it	TSLA
user4@gmail.com	AAPL
user5@gmail.com	AAPL
user6@gmail.com	AMZN

ticker	timestamp	value
AAPL	2024-11-25 16:00:00	232.87
AAPL	2024-11-26 16:00:00	235.06
AAPL	2024-11-27 16:00:00	234.93
AAPL	2024-11-29 13:00:00	237.33
AMZN	2024-11-25 10:03:00	200.085
AMZN	2024-11-25 10:05:00	200.33
AMZN	2024-11-25 10:07:00	200.134
AMZN	2024-11-25 13:31:35	200.61
AMZN	2024-11-25 13:33:36	200.75
AMZN	2024-11-25 13:37:36	200.845
AMZN	2024-11-25 13:39:38	200.68

Il Data Collector recupera ogni due minuti la lista degli utenti registrati e i relativi ticker associati dalla tabella users, utilizza la libreria yfinance per scaricare i dati finanziari aggiornati (come il prezzo di chiusura) per ciascun ticker e memorizza i dati nella tabella financial_data. Le chiamate verso yfinance sono protette attraverso un Circuit Breaker per gestire eventuali errori o ritardi nelle risposte da yfinance.
Di seguito è riportato un diagramma che mostri le interazioni, sia tra componenti dell'applicazione che tra l'applicazione e il mondo esterno.