**SCELTE PROGETTUALI HW1:**

Nel nostro progetto abbiamo implementato un sistema distribuito che include:

● Un server gRPC.

● Un servizio di recupero dati (DataCollector) da yfinance.

● Un database a supporto delle operazioni.

I servizi sono stati sviluppati come container docker e gestiti ed eseguiti con docker compose. Noi abbiamo implementato 3 container docker che sono: il server, il database e infine il dataCollector. Il server Grpc rappresenta l'interfaccia tra l'utente e l'intero sistema, riceve le richieste dall'utente e interagisce con il database per fornire le funzionalità richieste quindi vi è una comunicazione tra database e server. Questo è un sistema che permette di dare all'utente la possibilità di manifestare il suo interessamento ad una azione. Questo è il primo servizio. Questo servizio, come richiesto, offre le seguenti funzionalità: registrazione, aggiornamento, cancellazione, recupero ultimo valore, calcolo media degli ultimi x valori. Quando un utente si registra devo rendere questi dati persistenti ed è qui che entra in gioco il volume quindi il sistema interagisce con il database e il volume rende persistente i dati. Ovviamente nel server, nelle funzionalità di registrazione e aggiornamento dell’utente, è stata implementata una politica at most once tramite l’utilizzo di una cache che conserva il dato e non mi fa elaborare nuovamente l’informazione se è gia stata elaborata. L’ulteriore servizio che abbiamo implementato è il servizio di DataCollector. Esso è un servizio ovviamente indipendente, ha un timer, si attiva quando? Ogni 60 secondi. Esso si prende la lista degli utenti, si prende la lista delle azioni a cui gli utenti sono interessati e va a fare tutte le chiamate in fila a Yahoo finance(quindi abbiamo utilizzato una libreria specifica per poter accedere ad esso ed è stata inclusa in DataCollector) e si prende tutti i dati. I dati che prende li mette in un'altra tabella (chiamata dati\_finanziari) dove mette il time\_stamp, l'utente, l'azione e il valore. Per sopperire ad un eventuale guasto di Yahoo Finance, prima di chiamare Yahoo Finance, abbiamo inserito un circuit breaker nel mezzo che serve a fare la seguente cosa: faccio le chiamate, se yahoo finance funziona le faccio tutte, se lo stesso non funziona le faccio fallire velocemente. Naturalmente questi dati sono stati inseriti nel database e inoltre ovviamente questi dati devono essere mandati all’utente e qui entrano in gioco due funzionalità del server che sono il recupero dell’ultimo valore e il recupero media ultimi x valori. il dataCollector è un componente un po' particolare perché è il componente che guida tutto ma è un componente in un modo o nell'altro completamente indipendente dal server, infatti, facendoci caso, sono due componenti che sono fondamentali per la mia applicazione e non si parlano. Il dataCollector ovviamente è su un docker container differente. Infine abbiamo il terzo servizio che è il database che starà in un docker container differente ed esso contiene due tabelle che sono: utenti (che contiene tutte le informazioni degli utenti(email(primary key),ticker)), dati\_finanziari (che contiene tutte le info inerenti al valore del ticker di cui il client è interessato ed ha come attributi: email,ticker, valore, timestamp). Abbiamo una rete my\_network che consente ai container di comunicare tra loro. Questa rete, configurata con il driver bridge, collega i tre container: il **server**, il **DataCollector**, e il **database**. Abbiamo inoltre implementato un client per testare tutte le funzionalità da noi progettate. Ovviamente nelle tabelle del database implementato non abbiamo inserito il concetto di anonimizzazione dei dati privati visto che non richiesto però è di fondamentale importanza per seguire le norme del GDPR.

Di seguito è riportato un diagramma architetturale che mostra i micro-servizi coinvolti:

Client

localhost

loca



DataCollector

Container

Database

Container

Server

container



YahooFinance

Servizio esterno

Di seguito è riportato un diagramma che mostra le interazioni, sia tra componenti dell’applicazione che tra l’applicazione e il mondo esterno:

CLIENT



TIMER (ogni 60 s)

DATACOLLECTOR



DATABASE

SERVER

RICHIESTA UTENTE ATTIVAZIONE

RISPOSTA O ERRORE (open)

PRELEVA UTENTI

RICHIESTA

INSERIMENTO DATI FINANZIARI

CIRCUIT BREAKER

RISULTATI

CALL

OPERAZIONI UTENTI (REGISTRAZIONE, AGGIORNAMENTO, CANCELLAZIONE E RECUPERO DATI)

RISPOSTA O “ERRORE” (se è down)



YFINANCE



**VINCENZO COSTANZA**

**JOSE’ LUIS AQUILA**