Relazione progetto DSBD

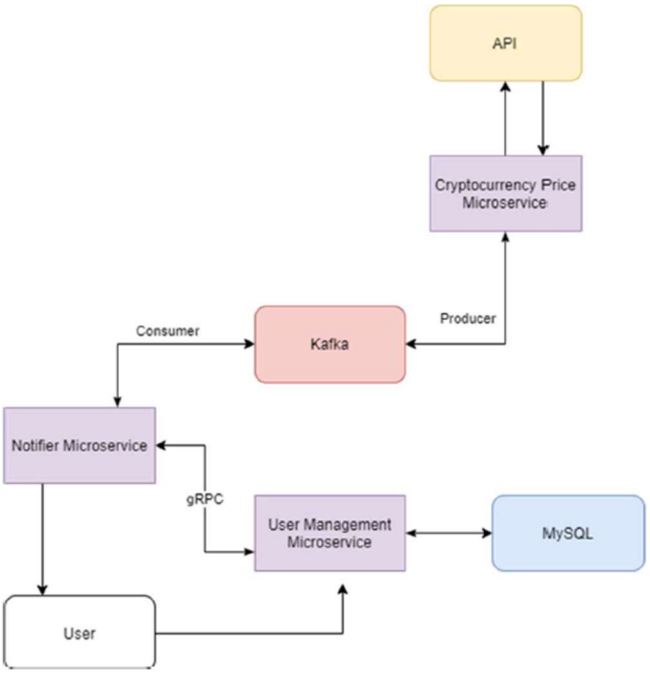
Introduzione

L'obiettivo del progetto è quello di realizzare un sistema distribuito che permetta di inviare agli utenti notifiche in tempo reale in relazione alla variazione del prezzo di diverse criptovalute a cui si è sottoscritti.
L'architettura del sistema è a microservizi, gestiti tramite docker.
Sono stati identificati tre microservizi:

- Retrieval
  L'obiettivo di questo microservizio è quello di realizzare, ad intervalli di tempo regolari, delle chiamate verso l'endpoint della API di coingecko.com in modo da ottenere un JSON che contiene le informazioni relative alle varie criptovalute. Il microservizio si occupa anche di eseguire un semplice scraping. Esso agisce da producer Kafka, pubblicando i dati nel relativo topic Kafka.
- Management
  L'obiettivo di questo microservizio è quello di interagire con l'utente tramite terminale per permettergli di sottoscriversi ad uno o più topic e fornire dei vincoli. Contestualmente, si occupa di memorizzare i dati in un database.
- Notifier
  L'obiettivo di questo microservizio è quello di agire da consumer Kafka, verificare se si sono verificate delle violazioni dei vincoli ed eventualmente avvertire l'utente mediante una mail dedicata.

Schema architetturale

Di seguito è riportato lo schema architetturale relativo al sistema realizzato:



Si è cercato di definire il campo di lavoro dei vari microservizi per isolarne le funzionalità ed ottenere un approccio che seguisse i concetti di separazione delle responsabilità tipici dei sistemi distribuiti. Nonostante la ricerca dell'isolamento dei vari microservizi, sono comunque presenti delle relazioni di dipendenza in quanto l'obiettivo del sistema è raggiunto dalla loro cooperazione. Proprio per questo motivo la comunicazione risulta essere un tema centrale. Sono state utilizzate due tecnologie: Kafka per permettere una comunicazione asincrona e indiretta tra Retrieval e Notifier; gRPC per permettere una comunicazione sincrona tra Management e Notifier all'interno di una rete bridge privata chiamata rete_container.

Microservizi

Retrieval

Questo microservizio si occupa di realizzare delle chiamate al seguente endpoint API di coingecko.com
https://api.coingecko.com/api/v3/coins/markets?vs_currency=eur&order=market_cap_desc&per_page=3&page=1&sparkline=false&locale=en

In risposta otterrà un file JSON contenente diversi campi per ogni criptovaluta. Il microservizio si occupa di effettuare uno scraping sui dati per generale una stringa con formattazione JSON contenente un sottoinsieme di campi del JSON originale (nome della criptovaluta, prezzo al momento della chiamata, prezzo massimo giornaliero, prezzo minimo giornaliero) e un valore calcolato dal microservizio stesso, la variazione percentuale. Dopo aver generato la stringa, tramite un Producer, si occupa di pubblicare i valori nei rispettivi topic Kafka.

Management

Il microservizio si occupa di interagire con l'utente attraverso il terminale e della persistenza dei dati mediante l'utilizzo di un database. Questo microservizio presenta un menù testuale che permette all'utente di sottoscriversi ad un topic o di inserire/aggiornare dei vincoli. Management agisce anche da client gRPC, utilizzando le Remote Procedure Call esposte dal server gRPC Notifier.
Il database Mysql contiene le informazioni associate all'utente (nome, congome, email, numero di telefono), al topic (nome), alle sottoscrizioni (email, topic) e ai vincoli. Di seguito un resoconto delle tabelle:

- UTENTE (NOME, COGNOME, EMAIL, TELEFONO)
- TOPIC (NOME)
- SOTTOSCRIZIONE(EMAIL_UTENTE, TOPIC_UTENTE)
- VINCOLI(EMAIL_UTENTE, TOPIC_UTENTE, PREZZO, PREZZO_MAX, PREZZO_MIN, VARIAZIONE_PERCENTUALE)

Il database è inizializzato al momento dell'avvio del relativo container attraverso la copia del file "inizializzazione.sql" all'interno della cartella del container docker-entrypoint-initdb.

Notifier

Il microservizio Notifier si occupa di monitorare i vincoli degli utenti sottoscritti nei vari topic Kafka, in caso di violazioni avviserà il relativo utente attraverso l'invio di una email.
A Notifier viene affidato il compito di gestire i vari Consumer che si sottoscrivono ai topic Kafka.
Mediante il metodo "sottoscrivi_utenti" viene generato un nuovo Consumer con un nuovo group_id che viene sottoscritto ad una lista di topic. A ciascun consumer viene poi associato un thread, mantenuto in una lista, che si occupa di consumare il messaggio ottenuto dal broker Kafka. Per ciascun messaggio relativo a quel consumer vengono confrontati i valori dei vincoli immessi dall'utente con i dati ricevuti dal broker e in caso di violazione viene invocato il metodo "invia_mail" che mediante l'utilizzo del protocollo SMTP invierà una mail all'utente.
Si è scelto di assegnare un group_id univoco in quanto l'idea alla base è quella che tutti gli utenti sottoscritti ad un topic debbano poter consumare il messaggio. Se due o più utenti sono sottoscritti allo stesso topic e condividono lo stesso **group_id**, Kafka garantirà la consegna a uno solo dei consumatori all'interno del gruppo, a meno di eventuali repliche. In questo modo, ogni utente non riceverà una copia del messaggio. Se invece gli utenti hanno **group_id** diversi, ognuno riceverà una propria copia dei messaggi pubblicati nel topic a cui sono sottoscritti.
Notifier agisce anche da server gRPC, esponendo le seguenti Remote Procedure Call:

rpc SendData (RequestData) returns (ResponseData);
Permette di ricevere i vincoli forniti dall'utente.
rpc SendNewSubscriber (Subscriber) returns (ResponseData);
Permette di ricevere i nuovi utenti che devono essere sottoscritti ad un topic.