# 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. Il deploy dell’applicazione è stato eseguito anche su kubernetes.

Sono stati identificati quattro 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 esponendo delle API per permettergli di sottoscriversi ad una o più criptovalute e fornire dei vincoli. Contestualmente, si occupa di memorizzare i dati in un database. Agisce da consumer Kafka.
* Notifier  
  L’obiettivo di questo microservizio è quello di avvertire l’utente mediante una mail dedicata in caso di violazione dei vincoli forniti. Sfrutta il protocollo SMTP.
* SLA Manager  
  L’obiettivo di questo microservizio è quello di permettere la gestione di un SLA. Fornisce la possibilità di modificare l’insieme delle metriche che compongono SLA, modificarne i valori limite; verificare se i vincoli sono attualmente violati e calcolare la probabilità di violazione nei prossimi minuti attraverso l’utilizzo di un modello ARIMA (auto ARIMA).

## Schema architetturale

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

Immagine che contiene testo, schermata, diagramma

Descrizione generata automaticamente

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 Management; 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 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). Dopo aver generato la stringa, tramite un Producer, si occupa di pubblicare i valori nel rispettivo topic Kafka.

### Management

Il microservizio si occupa di interagire con l’utente esponendo delle API e della persistenza dei dati mediante l’utilizzo di un database. Questo microservizio permette all’utente di sottoscriversi ad una (o più) criptovalute 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 avvisare gli utenti sottoscritti alle criptovalute attraverso l’invio di una email.

Notifier agisce anche da server gRPC, esponendo la seguente Remote Procedure Call:

rpc SendNewSubscriber (Subscriber) returns (ResponseData);

Permette di ricevere l’utente a cui deve essere inviata una mail.

### SLA Manager

Il microservizio SLA Manager si occupa di gestire e interagire con un SLA.

Questo microservizio interagisce con un server Prometheus che a sua volta effettua lo scraping delle metriche esposte da cAdvisor. SLA Manager garantisce la persistenza delle informazioni mediante l’utilizzo di un database mysql chiamato ‘metriche’. SLA Manager è implementato come un’app Flask che espone degli endpoint che permettono all’utente di:

* Visualizzare le metriche attuali che compongono SLA ed eventualmente modificarle (aggiungere nuove metriche o rimuoverne).
* Aggiornare i valori della metrica (valore desiderato, valore minimo e massimo)
* Visualizzare lo stato del SLA (SLA violato o non violato per ogni metrica)
* Visualizzare i valori desiderati delle metriche nel SLA
* Visualizzare le violazioni del SLA nelle passate 1,3,6 ore
* Visualizzare la probabilità di violazione nei prossimi X minuti attraverso la previsione dei valori sfruttando un modello autoARIMA.

Le metriche che sono state scelte sono:

* container\_cpu\_load\_average\_10s'
* 'container\_fs\_io\_time\_seconds\_total'
* 'container\_memory\_usage\_bytes'
* 'container\_memory\_failcnt'
* 'container\_network\_receive\_errors\_total'
* 'container\_network\_transmit\_errors\_total'

Si è scelto di analizzare le seguenti metriche per ogni servizio presente nell’applicazione distribuita.

Le metriche relative all’utilizzo della CPU e quelle relative alla memoria hanno lo scopo di spiegare e sintetizzare come il sistema sta gestendo le risorse, tenere traccia di questi valori da indicazioni su eventuali blocchi o rallentamenti nell’esecuzione. Le metriche relative agli errori di ricezione e trasmissione invece sono state scelte proprio perché l’applicazione distribuita fa largo uso di internet per interagire con l’esterno: il microservizio ‘Retrieval’ riceve dei dati mediante API; i servizi ‘SLA Manager’ e ‘Management’ basano le proprie funzionalità sull’esposizione di API. Quindi un’analisi degli errori di rete permette di capire se il sistema sta svolgendo i propri doveri in maniera corretta.