Progetto Distributed System and Big Data

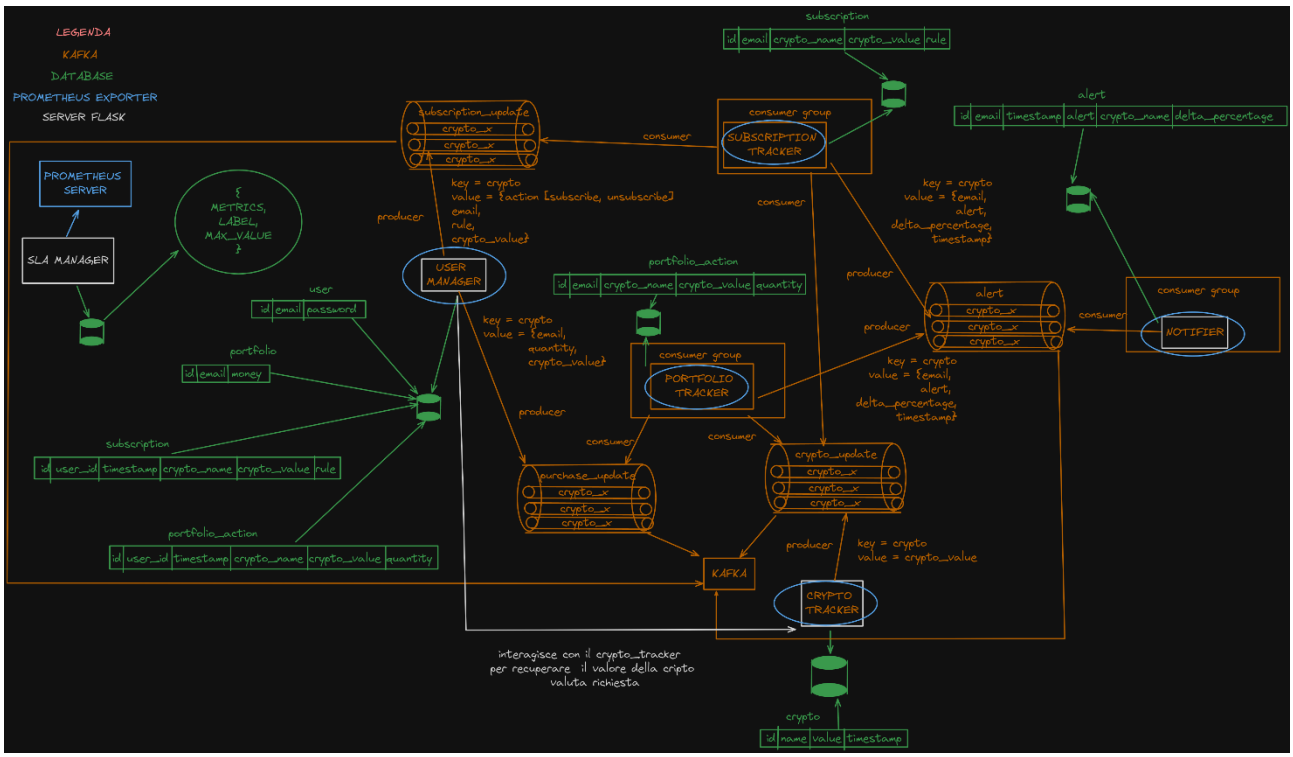
Portfolio tracker

Sviluppatori: Francesco Cerruto, matricola 1000005927 – Giovanni Traina, matricola 1000053629

ABSTRACT

Si intende realizzare un servizio web che emula l'acquisto, la vendita ed il monitoring di prodotti finanziari. Tramite apposite API REST, l'utente registrato ha accesso ad un proprio portafoglio digitale sul quale vengono memorizzate le azioni eseguite sui prodotti finanziari offerti dal sistema. L'utente avrà dunque la possibilità di manifestare il proprio interesse su alcuni dei prodotti finanziari offerti, specificandone contestualmente una soglia di variazione percentuale di interesse, acquistarli oppure venderli. Il sistema si occuperà di monitorare il valore reale dei prodotti scelti dell'utente e lo notificherà via email in caso di una variazione percentuale che soddisfa i requisiti applicativi da lui imposti. Le notifiche generate via email vengono inoltre memorizzate su di un database così da consentirne un successivo recupero da parte dell'utente.

DIAGRAMMA ARCHITETTURALE



Il sistema sviluppato è composto dai micro servizi

- User manager: si occupa dell'autenticazione dell'utente (previa registrazione) e ne gestisce i relativi dati
  - Autenticazione
    - (id, email, password)
  - Sottoscrizioni, e associate regole di notifica
    - (id, user_id, timestamp, crypto_name, crypto_value, rule)
  - Acquisti/vendite
    - (id, user_id, timestamp, crypto_name, crypto_value, quantity)
  - Portafoglio
    - (id, user_id, money)
- Notifier: sistema di notifica utente per la variazione percentuale del valore digitale del proprio portafoglio e delle cripto valute a cui si è sottoscritto. Gestisce i dati inerenti le notifiche
    - (id, email, timestamp, alert, crypto_name, delta_percentage)
- Crypto tracker: sistema di accesso e monitoraggio del valore reale delle cripto valute che, tramite un broker Kafka, notifica gli attori interessati. Gestisce i dati inerenti le cripto valute
    - (id, name, value, timestamp)
- Portfolio e Subscription tracker: sistema di elaborazione dei dati forniti dal Crypto tracker
- Prometheus: server di scraping delle metriche esposte dai precedenti micro servizi per consentire una gestione della Qos

- Sla manager: server che consente l'aggiunta di un SLO ed il monitoraggio delle metriche suddette

Lo User manager, oltre che essere un sistema di gestione dati utente, si comporta anche come producer kafka per attivare il sistema di monitoraggio delle proprie sottoscrizioni e de l proprio portafoglio. Il Crypto tracker, oltre che essere un producer kafka, consente allo User manager di accedere alle informazioni sui valori aggiornati delle cripto valute (informazioni non possedute da lui localmente). Il Notifier, oltre ad essere un consumer kafka, consente all'utente di recuperare le proprie notifiche

Si è scelto di separare il sistema di notifica e di elaborazione dati dal micro servizio User manager così da rendere il lavoro svolto dal suddetto micro servizio il più omogeneo possibile.

Per quanto riguarda lo strumento di memorizzazione permanente dei dati, vista la natura strutturata degli stessi, si è scelto come tipologia di database un database relazionale MySQL. Ogni micro servizio accede ad un database dedicato che ospita le tabelle di interesse per il micro servizio. Si è scelto di duplicare le informazioni possedute dallo User manger inerenti alle sottoscrizioni e agli acquisti/vendite sui servizi Portfolio e Subscription tracker, rispettivamente, così da rendere il sistema più scalabile (altrimenti lo User manager sarebbe diventato un collo di bottiglia). Scelta diversa è stata applicata per il database utilizzato dallo Sla manager in quanto si presta meglio alla memorizzazione di dati strutturati sotto forma di JSON

Il sistema di monitoraggio delle cripto valute consiste in un'interrogazione continua e periodica (opportunamente configurato) di un API esterna (www.coingecko.com). Viste le limitazioni imposte dall'API (rate limit di interrogazione) questo micro servizio non può essere duplicato. Per ottenere dei dati più o meno aggiornati si è scelto di interrogare il sistema esterno ogni 10 minuti.

La pubblicazione dei valori aggiornati delle cripto valute avviene tramite un broker kafka. In particolare sono stati predisposti 4 topic

- crypto_update: su questo topic vengono pubblicati dal Crypto tracker i valori aggiornati delle cripto valute
    - i consumer interessati sono Portfolio e Subscription tracker, ciascuno appartenente ad un consumer group distinto
- purchase_update e subscription: su questo topic vengono pubblicate dallo User manager le azioni di registrazione/cancellazione e acquisto/vendita sulle cripto valute effettuate dall'utente
- alert: su questo topic vengono pubblicate dal Portfolio e dal Subscription tracker gli eventi di notifica per l'utente sulla variazione percentuale del valore del proprio portafoglio e sulla variazione percentuale del valore della cripto valuta stessa, rispettivamente

Poiché tutti i micro servizi che si occupano consumare i dati dai vari topic predisposti non presentano fonte di errore, si è scelto di utilizzare l'auto commit del consumo del messagio.

Per aggiungere un ulteriore spunto di scalabilità per il sistema, ogni topic è stato predisposto con un opportuno numero di partizioni (pari al numero di cripto valute monitorate dal sistema). Per semplicità di è scelto di monitorare soltanto 11 cripto valute su 100 offerte dal sistema. In particolare le cripto valute monitorate sono

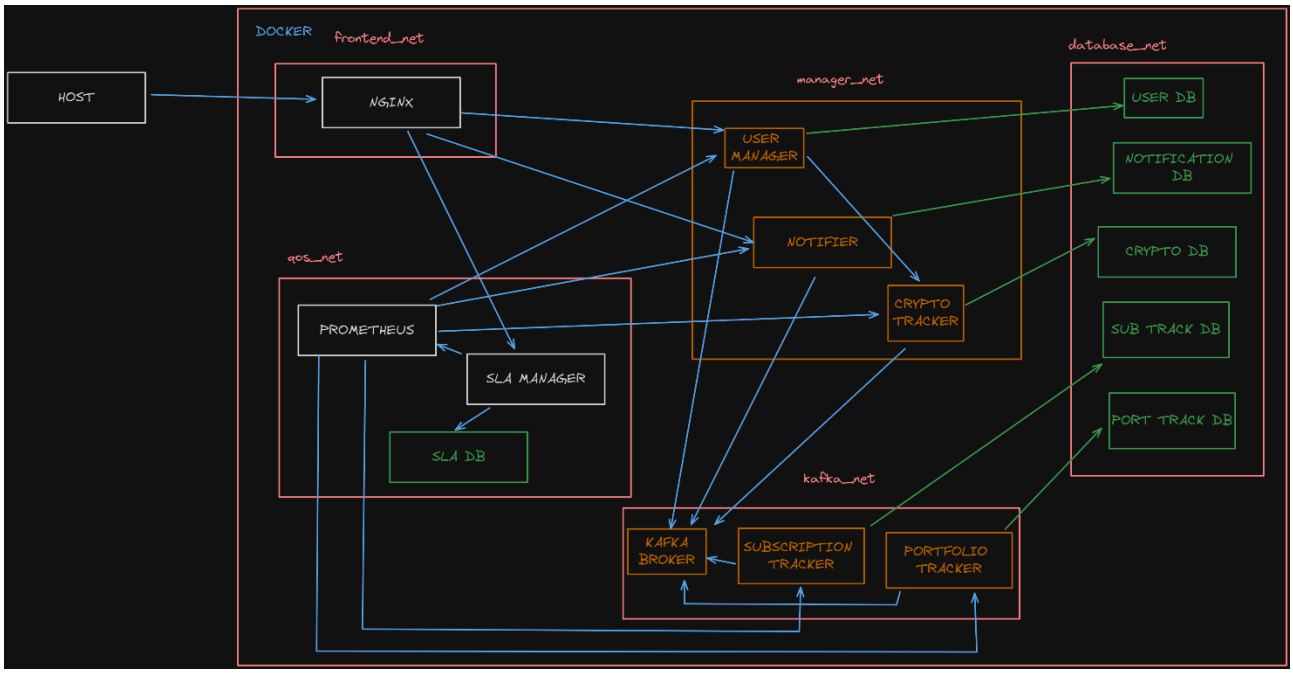
- Bitcoin
- Ethereum
- Tether
- BNB
- Solana
- XRP
- USDC
- Lido Staked Ether
- Cardano
- Avalanche
- Dogecoin

Così facendo su ogni partizione vengono inserite le informazioni inerenti quella specifica cripto valute ed ogni consumer kafka esegue le proprie operazioni sui dati su una porzione dell'intero database a disposizione

Per quanto riguarda la QoS, visto che il cuore centrale dell'intero sistema è kafka, si è scelto di monitorare il tempo di consegna dei messaggi. Ulteriore metrica di interesse, vista la presenza di numerosi db, è l'error rate del singolo micro serivzio. Ogni micro servizio dunque espone due metriche di tipo Counter, *number_error* e *number_operation*. Ad ogni azione eseguita dal micro servizio (sia questa il consumo di un messaggio o una richiesta alle API del servizio) le metriche sopra citate vengono incrementate. In particolare, la metrica number_error viene incrementata ogni qual volta i database non sono raggiungibili. Inoltre, tutti i micro servizi espongo una metrica di tipo Gauge, *error_rate*. Ogni qual volta i micro servizi aggiornano i counter, aggiornano di conseguenza il valore di questa richiesta. I micro servizi che svolgono anche il ruolo di producer kafka espongono un'ulteriore metrica di tipo Gauge, *delivery_time*. Ogni micro servizio è corredato di una *label* di pubblicazione per le metriche. Nello specifico si

ha il mapping (User manager, user), (Notifier, notifier), (Cryto tracker, crypto), (Portfolio tracker, portfolio), (Subscription tracker, subscription)

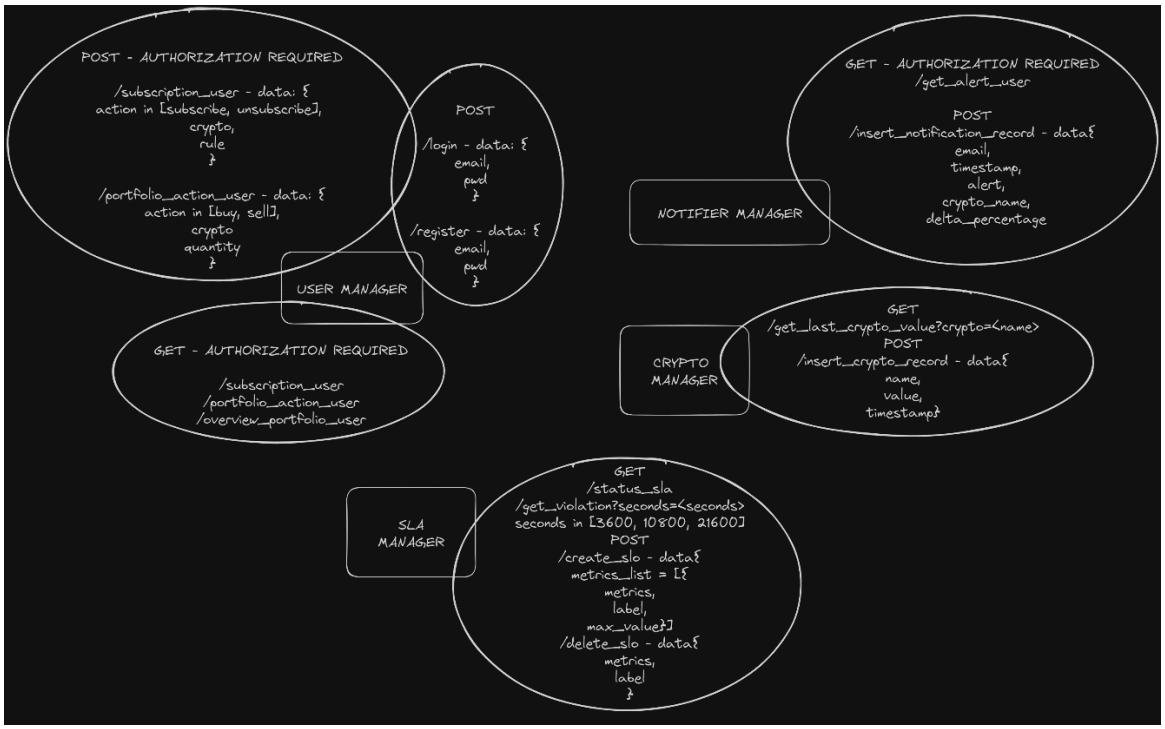
DIAGRAMMA ARCHITETTURALE DOCKER



Sono state predisposte 5 docker network

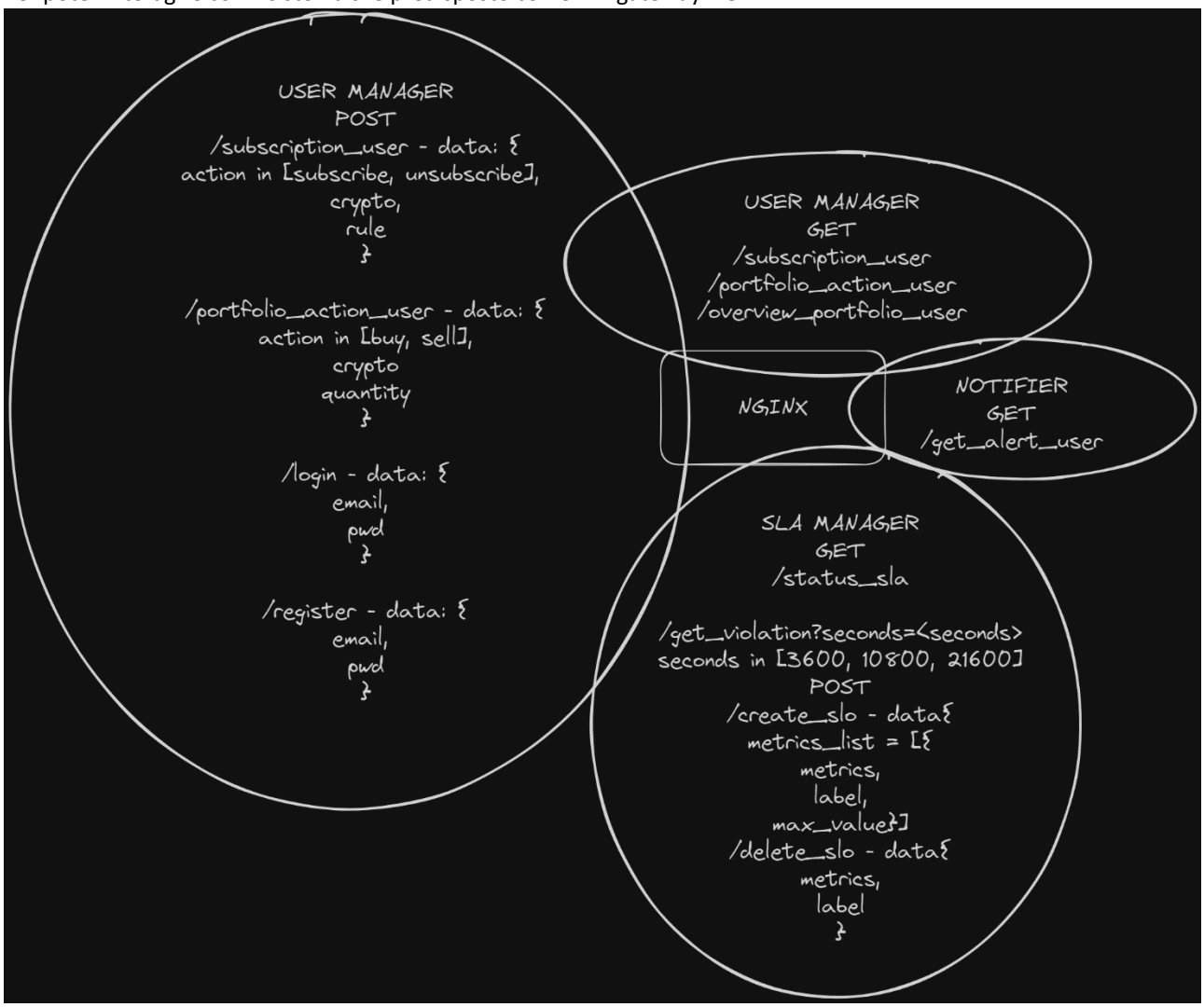
- frontend_net: i container esposti su questa rete sono quelli con cui interagisce l'utente
- manager_net: i container esposti su questa rete sono quelli che gestiscono i dati di interesse dell'utente
- kafka_net: i container esposti su questa rete sono quelli che interagiscono con il broker kafka
- database_net: su questa rete sono esposti tutti i database MySQL
- qos_net: i container esposti su questa rete sono quelli che si occupano della QoS e il database per la memorizzazione permanente del SLA

API IMPLEMENTATE



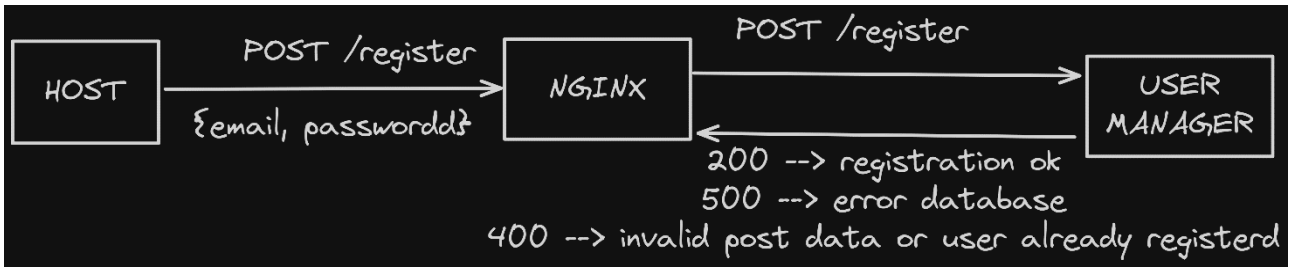
Come si può notare dall'immagine, l'autenticazione dell'utente viene eseguita tramite un JWT. Il token viene rilasciato dallo User manager e contiene, come informazioni di autenticazione, lo user_id (utilizzato nelle route dello User manager) e l'email (utilizzata nella route del Notifier). Per consentire la decifratura del token da parte di più micro servizi è stata predisposta una chiave di cifratura condivisa tra lo User manager ed il Notifier. La chiave, prodotta dallo User manager all'istante di avvio, viene successivamente scritta su di un file condiviso tra i due container (docker volume)

Per poter interagire con il sistema si è predisposto come API gateway NGINX

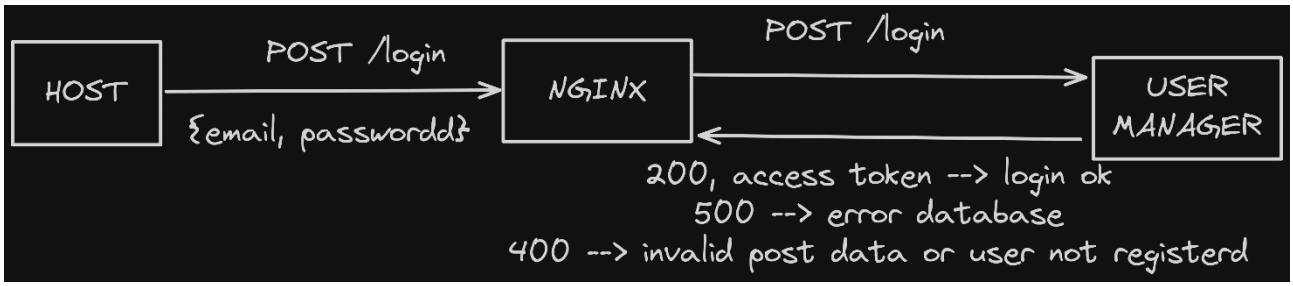


DETTAGLI API ED INTERAZIONI

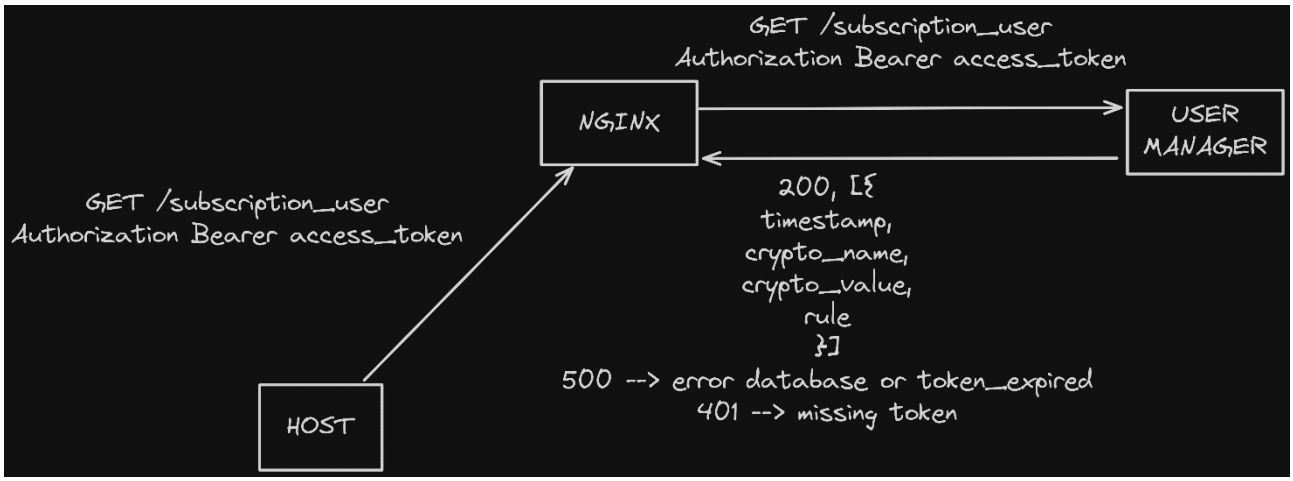
- User manager
    - Register: consente la registrazione dell'utente

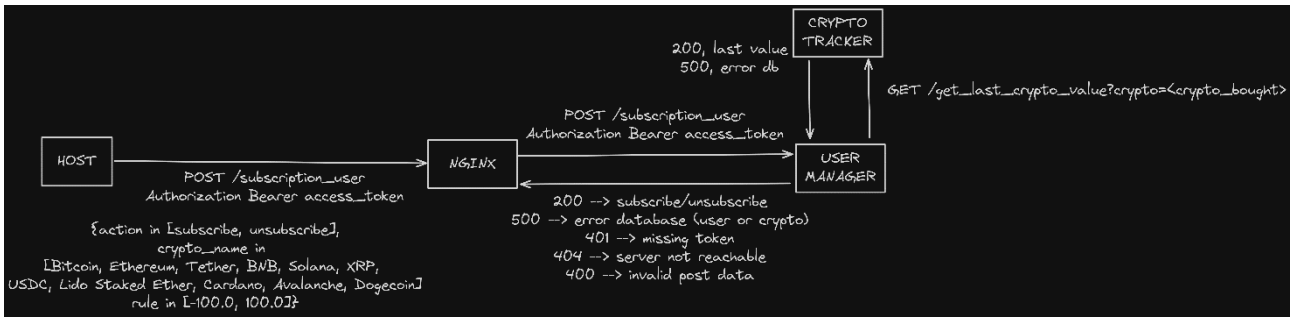


    - login: consente il login dell'utente ed il successivo rilascio del token

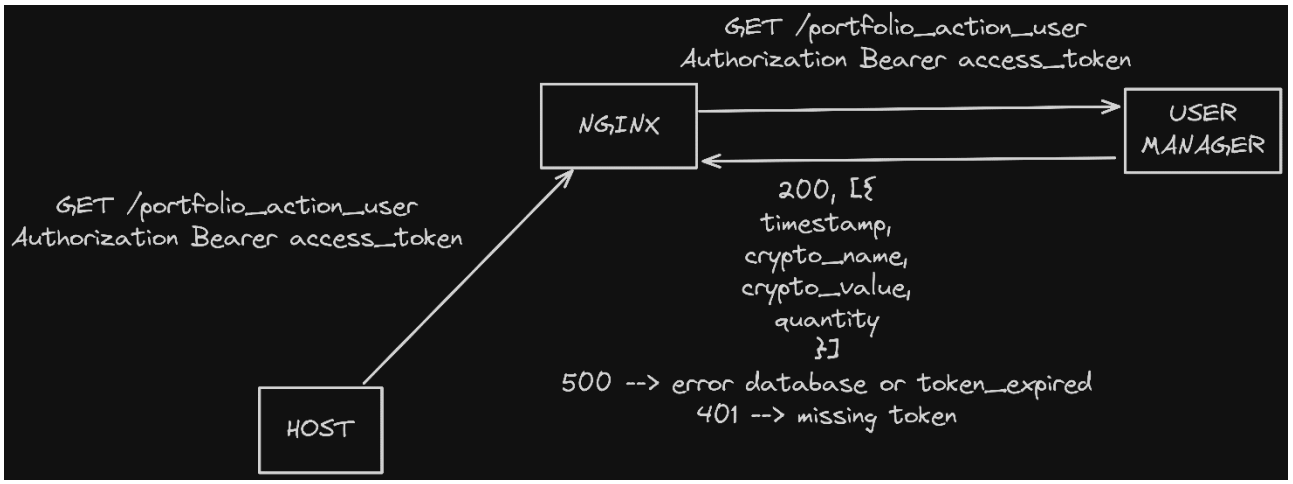


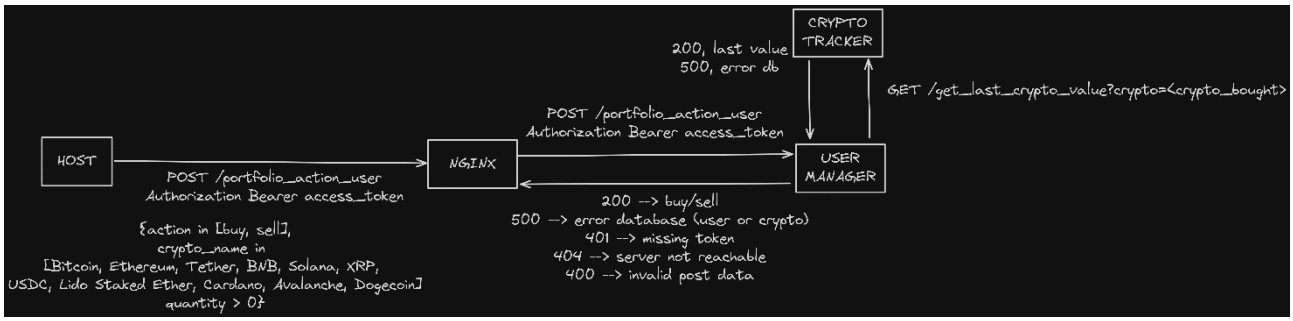
- subscription_user: consente il recupero e la modifica delle proprie sottoscrizioni



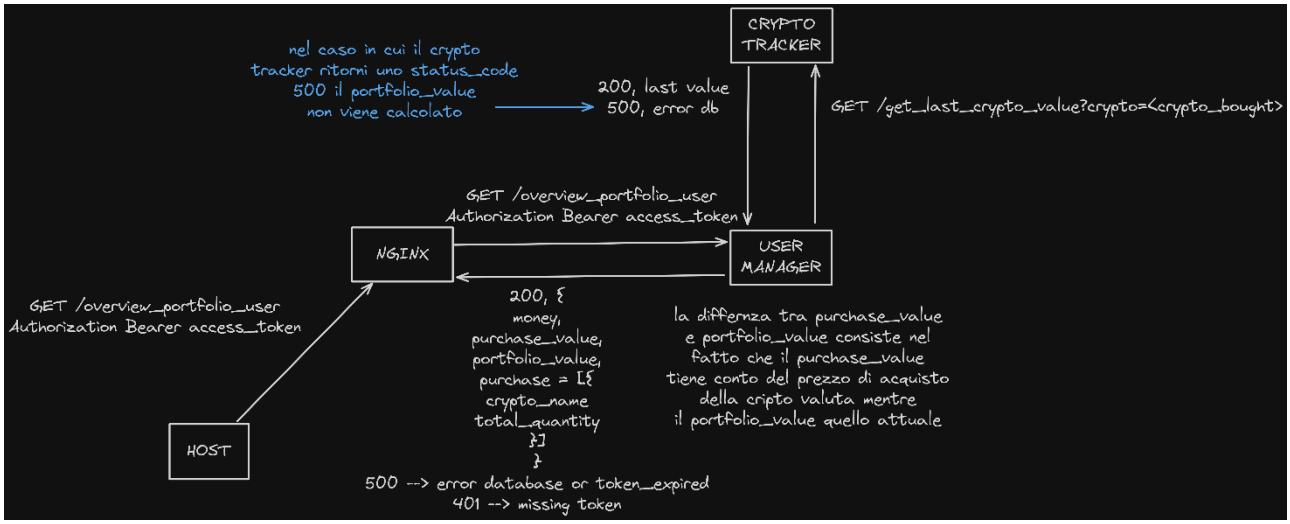


- portfolio_action_user: consente il recupero e la modifica dei propri acquisti (i record sono sotto forma di storico)

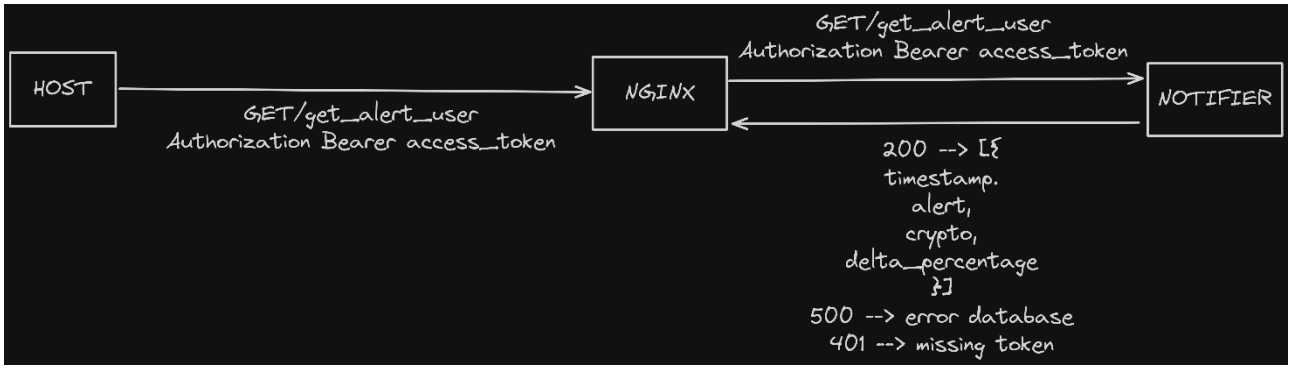




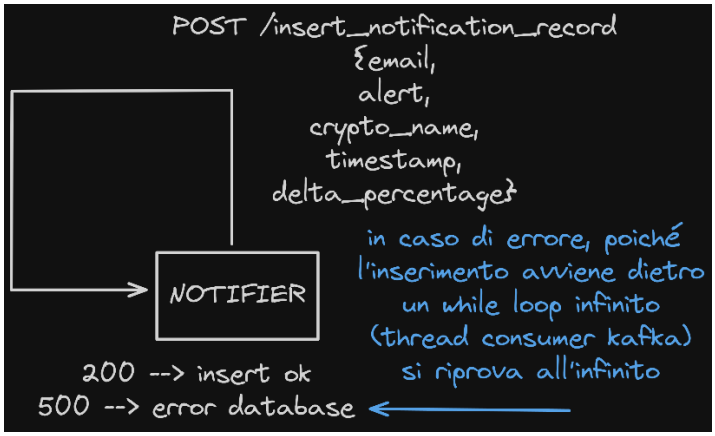
- overview_portfolio_user: mostra un quadro generale aggregato del portafoglio dell'utente



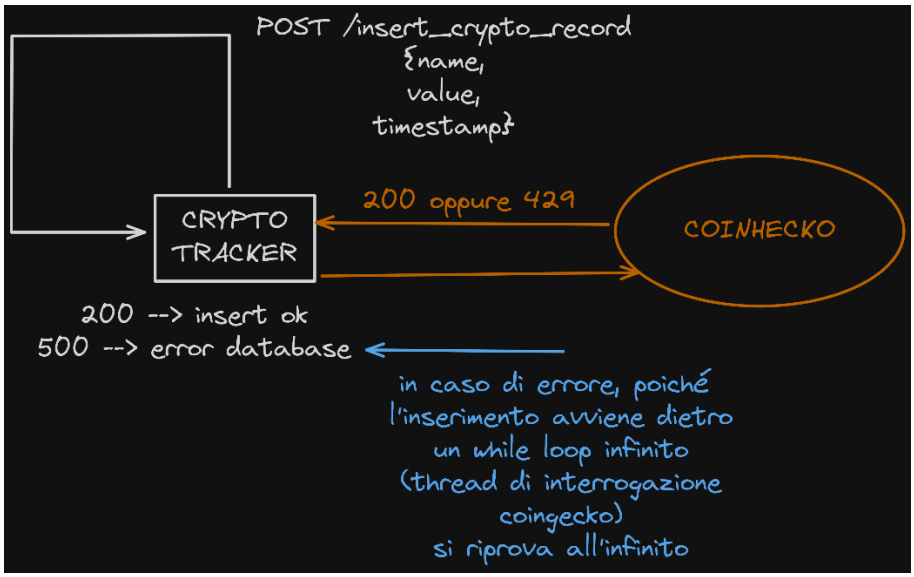
- Notifier
  - get_alert_user: consente il recupero delle notifiche emesse dal sistema



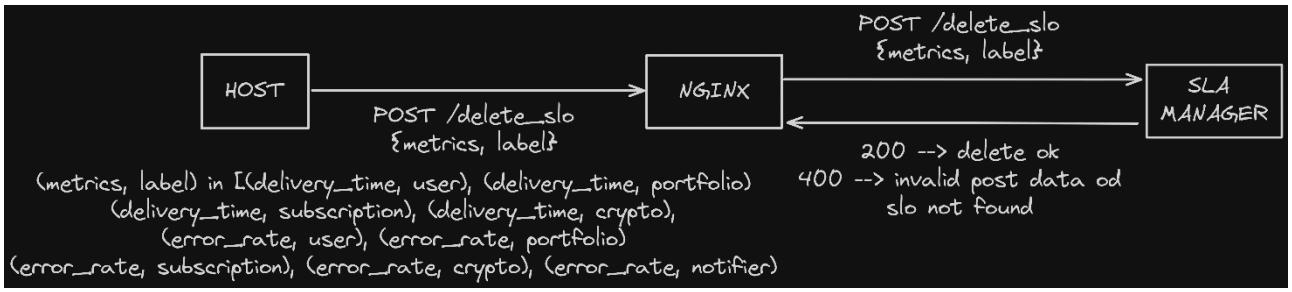
  - insert_notification_record: consente la memorizzazione permanente della notifica



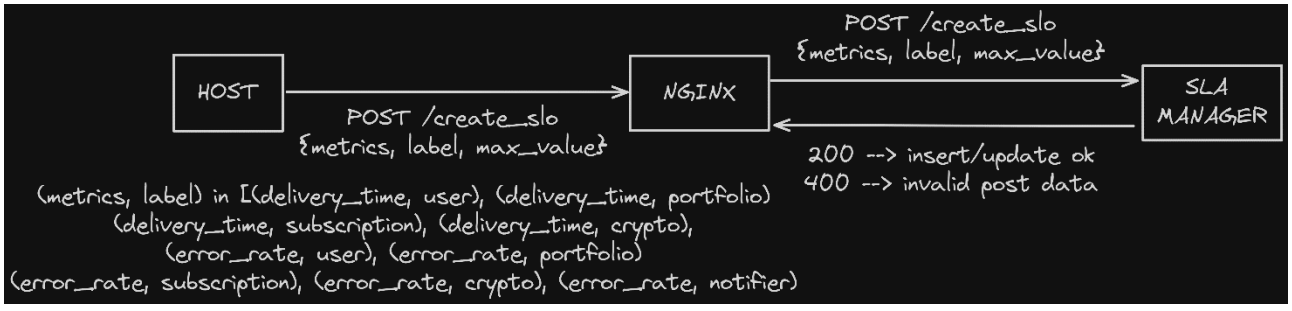
- Crypto manager
    - insert_crypto_record: consente la memorizzazione permanente della cripto valuta (i record sono sotto forma di storico)



- Sla manager
    - create_slo: consente l'aggiunta/modifica di un slo



    - delete_slo: consente la rimozione di un slo dal sla



    - status_sla e violation: consentono il recupero delle informazioni sullo stato del sla (stato attuale e numero di violazioni, rispettivamente)