Gruppo di lavoro: Salvatore Rinaudo 1000084774, Paolo Alfò 1000030219

Abstract

Il progetto riguarda la creazione di un sistema distribuito per il monitoraggio del traffico aereo. La soluzione si basa su un'architettura a microservizi (sviluppati come container Docker e orchestrati tramite Docker Compose), composta da due servizi principali: **User Manager** e **Data Collector**.

L'interazione con l'utente avviene tramite un client CLI (Command Line Interface) che comunica con i servizi tramite API REST. Il sistema offre funzionalità di gestione degli utenti (registrazione e cancellazione) e di monitoraggio di voli aerei, recuperando dati in tempo reale dalle API di **OpenSky Network**. Le operazioni di registrazione degli utenti sono state implementate garantendo una politica **"at-most-once"** tramite l'utilizzo di un identificativo univoco della richiesta (Request-ID) memorizzato su database, prevenendo duplicazioni in caso di ritrasmissioni.

Il **Data Collector** è un microservizio che gestisce gli interessi degli utenti (aeroporti da monitorare) ed esegue periodicamente, tramite un worker in background, il fetch dei dati sui voli. Il sistema implementa l'autenticazione tramite Client Credentials verso OpenSky e gestisce le limitazioni di rate-limiting per garantire la continuità del servizio. La persistenza dei dati è affidata a un database MySQL con schemi logici separati per utenti e dati di volo.

Diagramma Architetturale

Abbiamo scelto di suddividere il sistema in due microservizi distinti: **User Manager** e **Data Collector**.

Questi servizi sono supportati da un database MySQL containerizzato (condiviso a livello di container ma con database logici user_db e data_db separati) e comunicano tra loro sfruttando sia protocolli HTTP (REST) che **gRPC** per comunicazioni interne ad alta efficienza.

- **User Manager (Porta 5000 / 50051):** Espone API REST per la gestione anagrafica e un server gRPC (CheckUserExists) per permettere al Data Collector di verificare l'esistenza degli utenti.
- **Data Collector (Porta 5001):** Espone API REST per la gestione degli interessi (aeroporti) e le analisi statistiche. Contiene un worker thread che interroga ciclicamente OpenSky Network.
- **Database MySQL (Porta 3306):**
    - **user_db:** Contiene le tabelle users (email, username) e processed_requests (log delle richieste).
    - **data_db:** Contiene le tabelle interests (associazioni utente-aeroporto) e flights (storico dei voli).

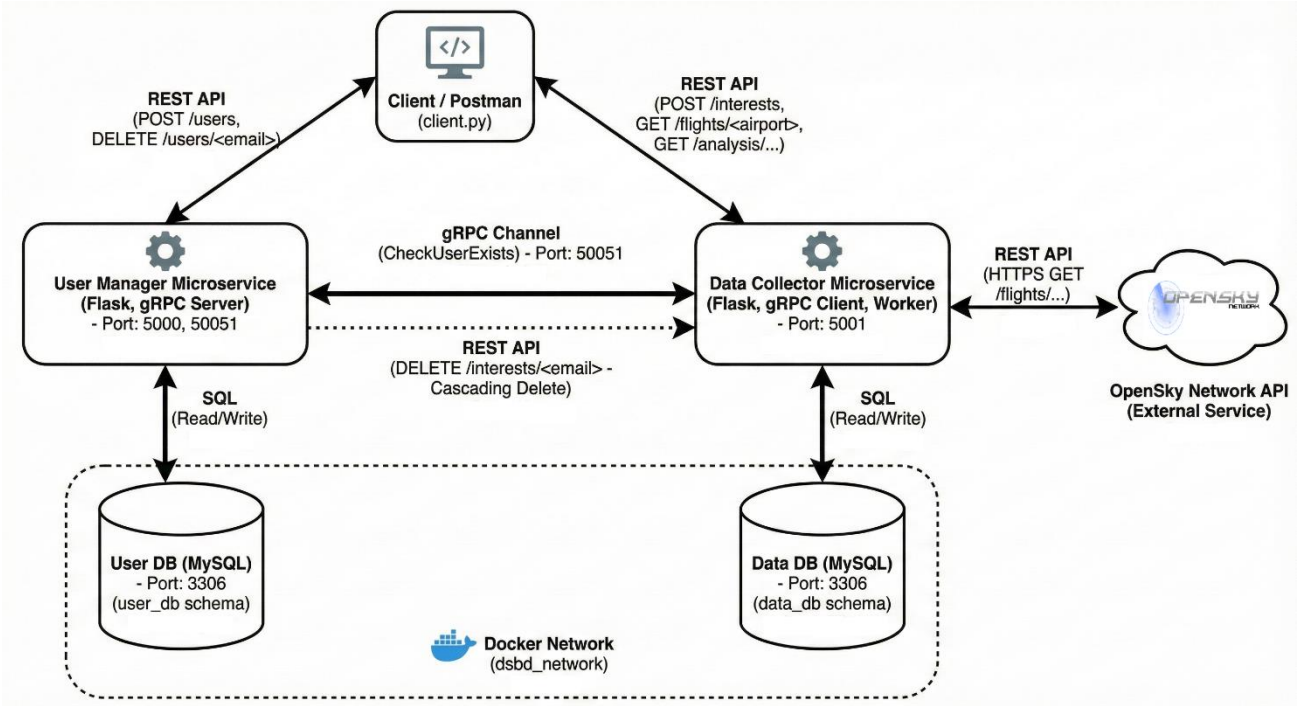


Diagramma delle interazioni

Registrazione Utente (Politica At-Most-Once)

Il sistema implementa una logica di **idempotenza** basata su token per gestire la registrazione degli utenti secondo la politica "at-most-once". Invece di affidarsi solo ai vincoli di unicità del database, il client genera un X-Request-ID univoco (UUID) per ogni tentativo di registrazione. Il server **User Manager** verifica prima nella tabella processed_requests se l'ID è già presente:

1. Se l'ID esiste: il server riconosce che l'operazione è già stata eseguita e restituisce un esito positivo senza effettuare nuove scritture.
2. Se l'ID non esiste: il server procede con l'inserimento dell'utente nella tabella users e, successivamente, registra l'ID della richiesta.

Questa strategia previene la creazione di utenti duplicati o errori in caso di ritrasmissione della stessa richiesta da parte del client a seguito di timeout di rete.

Cancellazione Utente (Cascading Delete Distribuita)

La cancellazione di un utente richiede un coordinamento tra i microservizi per mantenere la consistenza dei dati. Quando viene invocata l'API di cancellazione sullo **User Manager**:

1. Il servizio contatta preliminarmente il **Data Collector** tramite chiamata REST interna, richiedendo l'eliminazione di tutti gli interessi associati a quell'email.
2. Una volta inviata la richiesta di pulizia, lo User Manager procede all'eliminazione fisica del record utente dal user_db.

Aggiunta Interesse e Verifica gRPC

Quando un utente richiede di monitorare un nuovo aeroporto, il client invia una richiesta al **Data Collector**. Prima di salvare l'interesse nel database, il Data Collector deve verificare che l'utente sia

effettivamente registrato. Per fare ciò, agisce come client gRPC e invoca il metodo remoto CheckUserExists esposto dallo User Manager. Solo se la risposta gRPC conferma l'esistenza dell'email (exists=True), il Data Collector procede al salvataggio nella tabella interests.

Recupero e Analisi Dati (Ultimo Valore e Medie)

Il client permette di visualizzare i dati raccolti attraverso due funzionalità principali di analisi:

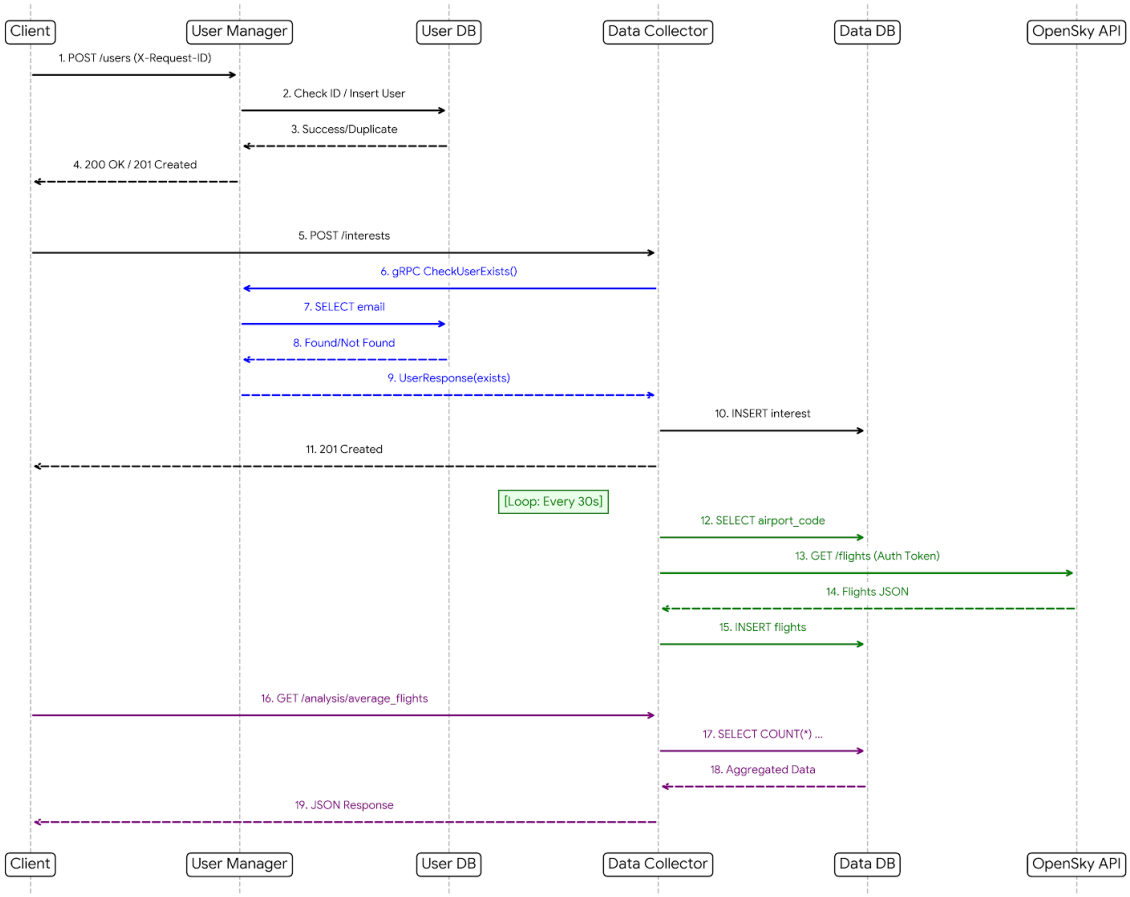
1. **Recupero Ultimo Volo:** Il server interroga la tabella flights filtrando per il codice aeroporto richiesto e ordinando per timestamp decrescente (ORDER BY time DESC LIMIT 1), restituendo i dettagli del volo più recente (Callsign, ICAO24, Orario).
2. **Calcolo della Media:** Il client specifica un numero di giorni X. Il server calcola il timestamp di inizio e conta il numero totale di voli (suddivisi in arrivi e partenze) registrati da quel momento in poi, restituendo la media giornaliera (total_flights / days).

Data Collector e Interazione con OpenSky

Per il popolamento del database è di fondamentale importanza l'interazione tra il Data Collector e le API esterne di OpenSky Network.

Il microservizio avvia all'accensione un **Worker Thread** ciclico che esegue le seguenti operazioni:

1. **Recupero Interessi:** Legge dalla tabella interests la lista degli aeroporti univoci monitorati dagli utenti.
2. **Autenticazione:** Ottiene un token di accesso OAuth/Bearer scambiando le credenziali (Client ID e Secret) con il server di autenticazione di OpenSky.
3. **Fetch dei Dati:** Per ogni aeroporto, scarica i voli in arrivo e in partenza.
4. **Gestione Rate Limiting:** Il sistema gestisce attivamente i codici di errore HTTP **429 (Too Many Requests)**. Se rilevati, il worker salta l'aeroporto corrente o attende per evitare il consumo di crediti per l'API di OpenSky, per questo motivo l'intervallo di raccolta è stato impostato a 1 ora.
5. **Persistenza:** I dati scaricati vengono salvati nella tabella flights utilizzando la clausola INSERT IGNORE per evitare duplicati basati sulla chiave univoca del volo (ICAO24 + Timestamp).


Client
User Manager
User DB
Data Collector
Data DB
OpenSky API
1. POST /users (X-Request-ID)
2. Check ID / Insert User
3. Success/Duplicate
4. 200 OK / 201 Created
5. POST /interests
6. gRPC CheckUserExists()
7. SELECT email
8. Found/Not Found
9. UserResponse(exists)
10. INSERT interest
11. 201 Created
[Loop: Every 30s]
12. SELECT airport_code
13. GET /flights (Auth Token)
14. Flights JSON
15. INSERT flights
16. GET /analysis/average_flights
17. SELECT COUNT(*) ...
18. Aggregated Data
19. JSON Response
Client
User Manager
User DB
Data Collector
Data DB
OpenSky API