Progetto #1 - Prova in itinere DSBD aa2025-2026

17/11/2025

Descrizione delle attività.

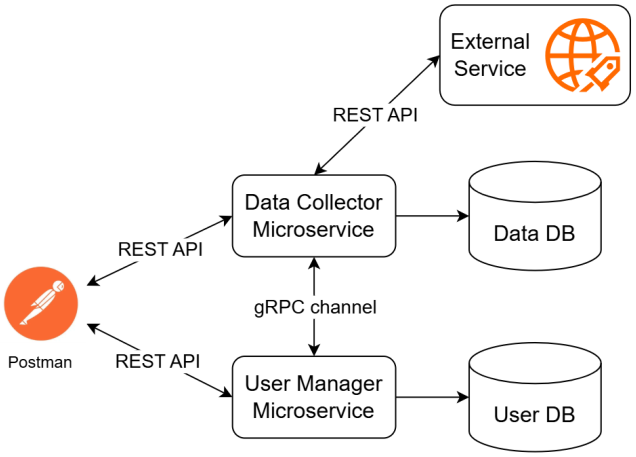
Obiettivo

Progettare e sviluppare un sistema dockerizzato a microservizi distribuito che includa:

- Un server per la gestione degli utenti (User Manager).
- Un servizio di recupero dati (Data Collector) da openSky Network.
- Almeno un servizio database per il supporto delle operazioni.

Il sistema deve essere in grado di gestire la registrazione e la cancellazione di utenti, nonché il recupero e l'elaborazione di dati su voli aerei da openSky Network. Le operazioni di registrazione degli utenti nello User Manager devono essere implementate con una politica "at-most-once" (attenzione questo requisito riguarda il messaging).
I *service* dovranno essere sviluppati come container docker e gestiti ed eseguiti con *docker compose*.

Per accedere alle API di openSky Network è conveniente registrarsi a openSky Network.
Qui di seguito trovate il diagramma architetturale (è solo una proposta, lo studente è libero di modificare l'architettura spiegandola opportunamente) che vi aiuterà nella realizzazione del homework:



User Manager: registra l'utente e memorizza i dati nello User DB
Data Collector: quando riceve una richiesta da parte di un utente prima verifica che l'utente esiste sfruttando il canale gRPC con lo User Manager. Secondariamente aggiunge l'interesse dell'utente nel Data DB.
Ciclicamente contatta il servizio esterno per ottenere i dati di interesse degli utenti e li memorizza nel Data DB.

I 2 DB si possono implementare sia come 2 container differenti oppure uno stesso container, ma con due database separati.

Descrizione del Progetto

User Manager
Riceve le richieste di registrazione dall'utente e interagisce con lo User DB per registrare l'utente con chiave primaria l'email e altre informazioni a scelta dello studente (es. codice fiscale, coordinate bancarie ecc…).

- Aggiunta di un Utente:
  - Permette l'inserimento di un nuovo utente.
- Cancellazione di un Utente:
  - Permette la cancellazione di un utente esistente.

Data Collector

L'utente contatta il Data Collector indicando uno o più aeroporti di suo interesse. L'utente può sempre aggiornare la lista dei suoi interessi.
Quando riceve una richiesta da parte dell'utente il Data Collector deve verificare se l'utente esiste contattando lo User Manager.
Il Data Collector è un servizio indipendente di monitoraggio che, in modo ciclico (es. una volta al giorno, ogni 12 ore), legge dal Data DB la lista degli aeroporti indicati dagli utenti, recupera le informazioni sui voli in partenza e/o in arrivo da ciascun aeroporto e scrive sul Data DB tali informazioni (a scelta dello studente è possibile sfruttare anche API diverse offerte da openSky Network).

L'utente può sempre richiedere al Data Collector tutte o alcune informazioni (a scelta dello studente) che ha memorizzato per gli aeroporti di suo interesse.

Funzionalità aggiuntive di recupero delle informazioni:

- Recupero dell'ultimo volo in partenza e in arrivo da un dato aeroporto:
  - Fornisce una funzione per ottenere l'ultimo volo in partenza e/o in arrivo da un dato aeroporto: il DataCollector interroga il database per recuperare il dato.
- Calcolo della Media degli ultimi X giorni:
  - Fornisce una funzione per calcolare e restituire la media degli ultimi X giorni sul numero di voli in partenza e/o in arrivo da un dato aeroporto: il DataCollector restituisce la media di quanti voli in partenza e/o quanti in arrivo ci sono stati negli ultimi X giorni da un dato aeroporto.

Lo studente può aggiungere l'utilizzo di altre funzionalità a suo piacere.

I Database

Gestione degli utenti

- Una o più tabelle per memorizzare le informazioni degli utenti.

Memorizzazione dei dati su voli aerei:

- Una o più tabelle per memorizzare i dati su voli aerei ottenuti dal DataCollector.

Requisiti Tecnici:

gRPC: Per la comunicazione tra Data Collector e User Manager.
Database: Relazionale (ad esempio, PostgreSQL, MySQL) o NoSQL, a scelta dello studente.

Criteri di Valutazione:

Correttezza Funzionale.

- Implementazione completa delle funzionalità richieste.
- Comportamento conforme alla descrizione del progetto.

Gestione degli Errori e Robustezza

- Robustezza nella gestione di eccezioni
- Implementazione della Politica "At-Most-Once".

Note di progetto

Le scelte progettuali (database utilizzati, decomposizione in microservizi, eventuali pattern di comunicazione) sono oggetto di valutazione e devono essere opportunamente motivate. Ogni gruppo può liberamente personalizzare il progetto, considerando che la valutazione delle scelte progettuali prevale sull'implementazione del codice. **NON** si richiede alcuna interfaccia grafica o qualsiasi forma di frontend e non incide in alcun modo sulla valutazione.

Modalità e tempi di consegna

Il progetto completo va consegnato improrogabilmente entro il 02/12/2025 (hard deadline).

Lo studente (ovvero il gruppo) deve esporre il progetto su GitHub e mandare il link tramite email al gruppo di docenti. (antonella.distefano@unict.it, giovanni.morana@unict.it, tutor alessandro.genovese@phd.unict.it).

Il codice del progetto dovrà essere consegnato come repository Git (es. su Github). Assieme al progetto va consegnata:

- una relazione che illustri le scelte progettuali (e.g., descrizione dell'applicazione, possibili variazioni della schema architetturale - (micro)servizi e relative comunicazioni - lista delle API implementate).La relazione deve includere:
    - un diagramma architetturale che mostri i micro-servizi coinvolti;
    - un diagramma che mostri il dettaglio delle interazioni, sia tra componenti dell'applicazione che tra l'applicazione e il mondo esterno.
- un documento con le informazioni utili (se presenti) per build & deploy, da consegnare in formato Markdown o PDF.

Il progetto può essere presentato anche se incompleto.