Immagine che contiene nero, oscurità

Descrizione generata automaticamente

**ELABORATO DISTRIBUITED SYSTEMS AND BIG DATA**

**Antonio Inveninato 1000055701**

**Angelo Cocuzza 1000055700**

Sommario

[Introduzione 3](#_Toc157594889)

[Architettura del sistema 4](#_Toc157594890)

[Implementazione microservizi 5](#_Toc157594891)

[Auth service 5](#_Toc157594892)

[Subscription service 7](#_Toc157594893)

[Api service 8](#_Toc157594894)

[Notifier service 9](#_Toc157594895)

[SLA manager 9](#_Toc157594896)

# Introduzione

Il presente progetto si propone di sviluppare un sistema innovativo e flessibile per la gestione delle informazioni relative ai voli e alle preferenze degli utenti, adottando un'architettura basata su microservizi. L'obiettivo principale è la creazione di un'infrastruttura scalabile e modularizzata, in grado di garantire una gestione efficiente e indipendente di funzionalità specifiche attraverso microservizi distinti.

L'architettura a microservizi è stata scelta strategicamente per consentire una maggiore flessibilità e adattabilità del sistema, permettendo lo sviluppo, il rilascio e la manutenzione indipendente di ciascun componente. Questa separazione modulare permette di affrontare le sfide legate alla complessità del sistema, facilitando la gestione e l'evoluzione delle singole funzionalità.

Gli obiettivi chiave di questo progetto includono la realizzazione di microservizi dedicati alle seguenti funzionalità:

**Auth Service:**

Gestione dell'iscrizione, accesso e informazioni degli utenti.

Interazione con un database per memorizzare in modo sicuro le informazioni degli utenti.

**Subscription Service:**

Gestione delle preferenze di volo degli utenti, compresa la scelta delle date e altri criteri di selezione.

Interazione con un database per immagazzinare e recuperare le preferenze degli utenti in modo efficiente.

**Api Service:**

Recupero regolare dei prezzi dei voli da servizi esterni attraverso l'utilizzo di API.

Filtraggio ed elaborazione dei dati in base alle preferenze di volo degli utenti.

Interazione con un database per immagazzinare e recuperare le preferenze degli utenti in modo efficiente e invio dei delle informazioni relative ai voli al microservizio di notifica.

**Notifier Service:**

Interazione con un database per memorizzare le informazioni relative ai voli ricevute.

Utilizzo di servizi esterni per inviare e-mail, garantendo una comunicazione tempestiva agli utenti.

Attraverso questa suddivisione delle funzionalità in microservizi, si mira a raggiungere una migliore scalabilità del sistema, facilitando l'espansione delle risorse in risposta alla crescita della domanda o all'introduzione di nuove funzionalità. Questo approccio consentirà inoltre una manutenzione agevole, permettendo di apportare modifiche specifiche a ciascun microservizio senza impattare il resto del sistema. Nel complesso, questo progetto si propone di realizzare un'architettura avanzata, incentrata sulla modularità e scalabilità, per soddisfare le esigenze dinamiche di un servizio di gestione voli efficiente e reattivo.

# Architettura del sistema

L’architettura è composta dai seguenti microservizi:

* auth\_service
* subscription\_service
* api\_service
* notifier\_service
* prometheus
* kafka
* sla\_manager
* gateway
* cAdvisor
* node-exporter

Immagine che contiene testo, schermata, diagramma, Blu elettrico

Descrizione generata automaticamente

# Implementazione microservizi

## Auth service

Il microservizio auth\_service è progettato per gestire l'autenticazione degli utenti attraverso operazioni di registrazione e login. Queste operazioni coinvolgono un database MySQL per la memorizzazione dei dati degli utenti e l'emissione di token JWT per l'autenticazione.

**Registrazione degli Utenti:**

Gli utenti possono registrarsi fornendo un nome utente e una password.

La password deve essere lunga almeno 8 caratteri.

Vengono registrati i tentativi di registrazione e se l'utente esiste già.

**Autenticazione e Login:**

Gli utenti possono effettuare il login fornendo le loro credenziali.

Vengono registrati i tentativi di login, inclusi quelli che falliscono.

Se le credenziali sono valide, viene generato un token JWT che verrà utilizzato per le successive richieste autenticate.

**Token di Autenticazione:**

Viene utilizzato JSON Web Token (JWT) per creare token di autenticazione sicuri.

I token contengono il nome utente e un tempo di scadenza.

I token vengono emessi durante la registrazione e il login.

**Metriche di Monitoraggio:**

Il microservizio registra metriche di monitoraggio, come il numero totale di utenti registrati, tentativi di login riusciti/falliti e metriche di utilizzo delle risorse del sistema.

Utilizza Prometheus per esporre queste metriche, consentendo il monitoraggio delle prestazioni e delle risorse del microservizio.

**Scheduler di Metriche:**

Un job schedulato periodicamente raccoglie metriche sulle risorse di sistema come l'utilizzo della CPU, la memoria e lo spazio su disco.

Queste metriche sono utili per il monitoraggio delle prestazioni e la risoluzione dei problemi.

**Sicurezza:**

La password deve essere almeno di 8 caratteri per garantire una certa sicurezza.

I dati sensibili come la password sono memorizzati nel database in modo sicuro.

**Gestione degli Errori:**

L'applicazione gestisce gli errori durante il processo di registrazione e login, fornendo messaggi appropriati agli utenti.

**CORS abilitato:**

Abilita il Cross-Origin Resource Sharing (CORS) per consentire alle richieste di provenire da domini diversi.

**Database:**

Interagisce con un database MySQL per memorizzare le informazioni degli utenti.

## Subscription service

Il microservizio è progettato per gestire le sottoscrizioni di viaggio degli utenti, consentendo agli utenti di registrare le loro preferenze di volo.

Il microservizio segue un'architettura a microservizi, utilizzando Flask come framework principale per la creazione dell'applicazione web. La persistenza dei dati viene gestita tramite SQLAlchemy, mentre la comunicazione asincrona avviene attraverso Kafka per l'invio di dati a sistemi esterni.

**Sottoscrizione al volo**

Gli utenti autenticati possono sottoscriversi ai voli, specificando città di partenza e destinazione, date di viaggio, ritorno e intervalli di prezzo.

**Sicurezza e Autenticazione**

L'autenticazione è gestita tramite l'utilizzo di token JWT, garantendo che solo utenti validi possano registrare le proprie preferenze di volo.

Le informazioni sensibili degli utenti sono protette durante la trasmissione e la persistenza.

**Invio dei Dati a Kafka**

Dopo la sottoscrizione, i dati vengono convertiti in formato JSON e inviati a un sistema di messaggistica Kafka tramite il produttore Kafka integrato.

**Monitoraggio e Metriche**

Il microservizio espone metriche Prometheus per il monitoraggio delle prestazioni, inclusi il numero di sottoscrizioni riuscite/fallite, il tempo di elaborazione delle sottoscrizioni e le metriche di utilizzo delle risorse del sistema.

**Scheduler per le Metriche**

Un scheduler periodico esegue la raccolta di metriche sulle risorse di sistema, consentendo un monitoraggio continuo delle prestazioni.

## Api service

Questo microservizio, implementato in Flask utilizzando Python, funge da consumatore e produttore in un'architettura basata su Kafka, con l'obiettivo di gestire le preferenze degli utenti per i voli e fornire dati sui voli.

**Kafka Consumer:**

Si connette a un topic Kafka chiamato "users" come consumatore per ricevere messaggi contenenti le preferenze degli utenti.

Salva le preferenze degli utenti nel database MySQL utilizzando SQLAlchemy. Il modello SubPreferences rappresenta la struttura delle preferenze.

**API Tequila Kiwi Integration:**

Utilizza l'API Tequila Kiwi per ottenere i codici IATA (International Air Transport Association) delle città specificate dagli utenti.

Integra i dati ottenuti dall'API nei messaggi successivi che verranno inviati a Kafka.

**Kafka Producer:**

Agisce come produttore inviando i dati dei voli su un altro topic Kafka chiamato "flights".

Utilizza l'API Tequila Kiwi per ottenere informazioni sui voli in base alle preferenze degli utenti.

**Monitoraggio e Metriche:**

Raccoglie metriche di sistema, come l'utilizzo della memoria, l'utilizzo della CPU e lo spazio su disco, utilizzando psutil e Prometheus.

Espone le metriche su un endpoint HTTP per il monitoraggio attraverso Prometheus.

**Attività Periodiche e Pianificazione:**

Esegue ciclicamente attività periodiche tramite la funzione schedule\_jobs:

Consuma messaggi da Kafka usando la funzione consume\_messages.

Controlla l'orario corrente e, se è le ore 8:00, esegue la funzione flights per ottenere e inviare dati sui voli a Kafka.

Misura le metriche di sistema tramite la funzione measure\_metrics.

Ripete il ciclo con una pausa di 45 secondi.

**Gestione degli Errori:**

Gestisce errori durante l'invio di dati su Kafka, l'ottenimento dei dati dall'API e il salvataggio nel database.

**Esposizione delle Metriche:**

Espone le metriche di sistema su un endpoint HTTP per il monitoraggio tramite Prometheus.

## Notifier service

Questo microservizio, implementato in Flask con Python, è progettato per ricevere notifiche relative a nuove offerte di volo da un topic Kafka, salvarle nel database MySQL e inviarle agli utenti interessati tramite email.

**Kafka Consumer:**

Si connette a un topic Kafka chiamato "flights" come consumatore per ricevere messaggi contenenti dati relativi alle offerte di volo.

Salva le informazioni nel database MySQL utilizzando SQLAlchemy. Il modello BestFlights rappresenta la struttura del database.

**Salvataggio nel Database:**

Utilizza SQLAlchemy per salvare le nuove offerte di volo nel database MySQL. Verifica se l'offerta è già presente nel database e la salva solo se è una nuova offerta o se il prezzo è migliore.

**Invio di Email di Notifica:**

Utilizza un account email configurato per inviare notifiche agli utenti. Le credenziali dell'account sono ottenute da variabili d'ambiente.

Costruisce un'email contenente i dettagli delle nuove offerte di volo e la invia all'utente interessato.

**Monitoraggio e Metriche:**

Raccoglie metriche di sistema, come l'utilizzo della memoria, l'utilizzo della CPU e lo spazio su disco, utilizzando psutil e Prometheus.

Espone le metriche su un endpoint HTTP per il monitoraggio attraverso Prometheus.

## SLA manager

Il microservizio offre un'infrastruttura robusta per il monitoraggio delle metriche, consentendo un'analisi approfondita delle prestazioni e facilitando la previsione di potenziali problemi.

**Configurazione del Database:**

Utilizza SQLAlchemy per la configurazione e l'interazione con un database MySQL, consentendo una gestione efficiente delle metriche.

**Modello del Database "Metrics":**

Definisce un modello Metrics con colonne per il nome della metrica, i valori minimi e massimi. Questo modello riflette la struttura delle metriche monitorate.

**Endpoint "/api/add":**

Implementa un endpoint POST che consente l'aggiunta di nuove metriche. I dati vengono ricevuti attraverso un modulo di form nella richiesta e vengono quindi memorizzati nel database per ulteriore monitoraggio.

**Endpoint "/api/status":**

Fornisce un endpoint GET per recuperare lo stato corrente di tutte le metriche. Viene effettuata un'interrogazione a Prometheus per ottenere i valori attuali delle metriche, e si verifica se rientrano nei limiti specificati.

**Endpoint "/api/singlestatus/":**

Implementa un endpoint POST per ottenere lo stato di una singola metrica. Interroga Prometheus per recuperare il valore attuale della metrica specificata.

**Endpoint "/api/violations":**

Fornisce un endpoint GET per ottenere il conteggio delle violazioni delle metriche in periodi di tempo specificati (1, 3, 6 ore). Utilizza i dati storici delle metriche e rileva violazioni in base ai limiti definiti.

**Endpoint "/api/probability":**

Implementa un endpoint POST per calcolare la probabilità di violazione in un periodo futuro specificato (6 ore). Utilizza i dati storici delle metriche e una funzione di previsione per stimare la probabilità di violazione.

## Prometheus

Viene utilizzato Prometheus per monitorare i diversi microservizi

I microservizi auth\_service, subscription\_service, api\_service, e notifier\_service espongono endpoint metrici che forniscono informazioni sulle prestazioni e lo stato del servizio.

Prometheus esegue lo scraping periodico di questi endpoint per raccogliere metriche e mantenerle in un database.

In particolare è stato definito un file .YML che descrive la configurazione di un server Prometheus per il monitoraggio di diversi job tramite la raccolta di metriche. L'intervallo di raccolta delle metriche è di 15 secondi per tutti i job. Ogni job raccoglie metriche da servizi specifici attraverso le rispettive porte specificate.

## Gateway

Viene creato il file di configurazione che consente a Nginx di definire un server block che ascolta sulla porta 80 e gestisce il reverse proxy per tre upstream:

Upstream "auth": Questo upstream è configurato per instradare le richieste verso il servizio auth\_service sulla porta 5000. Puoi facilmente aggiungere altri server a questo upstream se necessario.

Upstream "subscription": Similarmente, questo upstream instrada le richieste verso il servizio subscription\_service sulla porta 5000.

Upstream "sla": Questo upstream instrada le richieste verso il servizio sla\_manager sulla porta 5000.

Il blocco di server gestisce quindi i percorsi delle richieste per ciascun upstream specificando il prefisso di percorso tramite la direttiva location.

location /auth/: Tutte le richieste con il prefisso "/auth/" vengono instradate verso l'upstream "auth".

location /subscription/: Tutte le richieste con il prefisso "/subscription/" vengono instradate verso l'upstream "subscription".

location /sla/: Tutte le richieste con il prefisso "/sla/" vengono instradate verso l'upstream "sla".