Relazione DSBD Samperi_Volpe

Abstract

Il progetto da noi presentato propone lo sviluppo di un'applicazione distribuita il cui scopo è quello di fornire ai clienti la possibilità di sottoscriversi e di essere notificati su eventuali offerte su prezzi di voli di suo interesse o su voli per qualsiasi destinazione con prezzi particolarmente bassi in partenza da aeroporti da esso specificati.

Di seguito presentiamo il funzionamento:
-il **Client** si interfaccia con lo **User Controller** per registrarsi e per effettuare le richieste di iscrizione, una richiesta di iscrizione può essere del tipo iscrizione ad una tratta (in cui l'utente specifica l'aeroporto di destinazione, quello di origine, e un budget), oppure può richiedere di iscriversi per ricevere delle offerte speciali riguardo voli che partono da un determinato aeroporto (in questo caso l'utente inserisce solo l'aeroporto di origine e una soglia di prezzo).
-**User Controller** si occupa di registrare (o disiscrivere) l'utente e le sue "rules" nei relativi database; quindi, inoltrerà le richieste dell'utente rispettivamente ai microservizi: **UserInfo**, che ha accesso al database "users", e **Rules**, il quale ha accesso al database "rules", al cui interno troviamo le tabelle tratte e aeroporti.
User Controller ha inoltre il ruolo di inoltrare le tratte e gli aeroporti di interesse dei clienti al microservizio **Controller Tratte.**
-**Controller Tratte** ha quindi il ruolo, man mano che riceve tratte e aeroporti da **User Controller**, di inserire (o eliminare) questi dati nel proprio database "controllertratte", composto da due tabelle: tratte_salvate e aeroporti_salvati; ha inoltre il ruolo di inviare, quando richiesti, tali dati allo **Scraper** per permettergli di effettuare le richieste alle API esterne.
-Lo **Scraper** ogni 24h dovrà effettuare una richiesta a **Controller Tratte** per poter ricevere da esso le tratte e gli aeroporti , così da poter effettuare le richieste alle API esterne offerte da Amadeus (https://developers.amadeus.com) in particolare utilizziamo Flight Offers Search per la ricerca delle tratte data origine e destinazione, e Flight Inspiration Search per la ricerca delle offerte con sede di partenza l'aeroporto fornito dall'utente; quindi invierà i risultati ottenuti nei rispettivi topic Kafka: Tratte e Aeroporti.
-l'**Elaboratore** è il consumer di tali topic, quindi prende i dati da essi e si occuperà di effettuare una richiesta a **User Controller**, inviandogli i dati ottenuti, per ricevere le e-mail degli utenti interessati a quelle specifiche offerte; quindi, invierà al **Notifier** le e-mail e le informazioni da inviare agli utenti iscritti.
-il **Notifier** quindi, ottenute le informazioni dall'elaboratore, avviserà via e-mail il cliente.

Elenco microservizi e relative comunicazioni

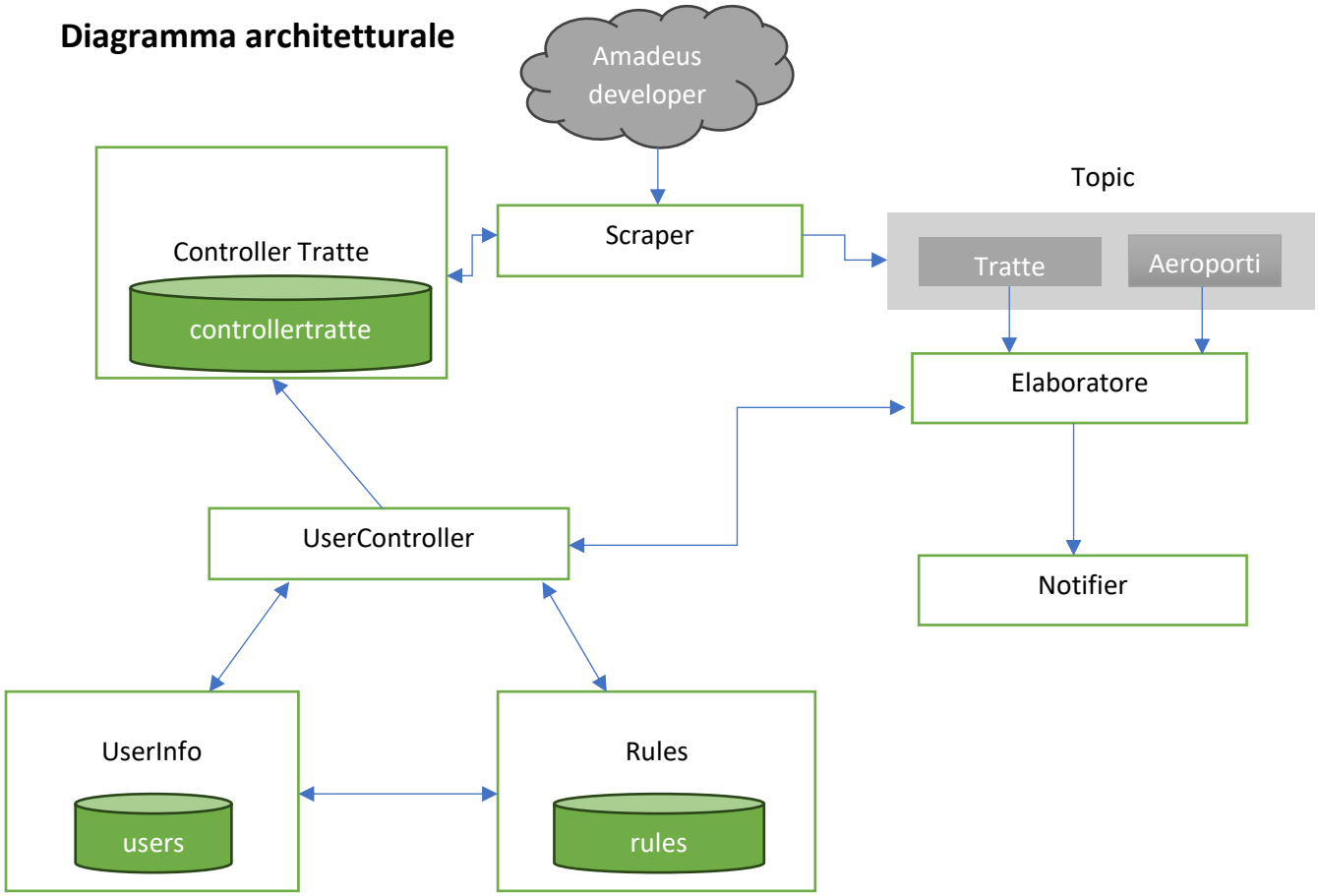
- UserController
- UserInfo
- Rules
- Controller_tratte
- Scraper
- Elaboratore
- Notifier

Nella nostra applicazione, il componente Scraper agisce come produttore dei topic **Kafka** “Tratte” e “Aeroporti”, per ottenere i dati da inserire in questi topic, esso esegue richieste periodiche all'API esterna fornita da **amadeus developers** ogni 24 ore, tali richieste consentono di raccogliere informazioni aggiornate sui prezzi delle tratte e sulle offerte disponibili. Dall'altro lato, il componente Elaboratore funge da consumatore, sottoscrivendosi ai medesimi topic.

L'utilizzo di Kafka come broker consente una comunicazione affidabile e asincrona tra i due componenti, garantendo una gestione efficiente dei dati.

I restanti microservizi interagiscono tra loro attraverso il servizio **Flask**. Ciascun microservizio è implementato come un'applicazione Flask indipendente, e le comunicazioni avvengono mediante richieste HTTP POST e GET. Questa scelta di design offre una flessibilità significativa, permettendo ai microservizi di scambiare dati in modo rapido ed efficiente.

Diagramma architetturale



Installazione e utilizzo

Dal repository Github è possibile effettuare il docker-compose up e verranno creati tutti i container, per le richieste è possibile farle comodamente tramite Curl da terminale, in particolare:

- **Per la registrazione:**

  curl "http://localhost:5000/registrazione" \

-X POST \

-d "{\"email\": \"prova_notifier@hotmail.com\", \"nome\": \"prova\", \"cognome\": \"1\"}" \

-H "Content-Type: application/json"

- **Per l'iscrizione a un topic tratta:**

  curl "http://localhost:5000/Insert_tratta" \

  -X POST \

  -d "{\"email\": \"prova_notifier@hotmail.com\", \"origine\": \"CTA\", \"budget\": 80}" \

  -H "Content-Type: application/json"

- **Per la disiscrizione al topic tratta:**

  curl "http://localhost:5000/Disiscrizione_tratta" \

  -X POST \

  -d "{\"email\": \"prova_notifier@hotmail.com\", \"origine\": \"CTA\", \"destinazione\": \"FCO\"}" \

  -H "Content-Type: application/json"

- **Per la discrizione al topic aeroporto:**

  curl "http://localhost:5000/Disiscrizione_aeroporto" \

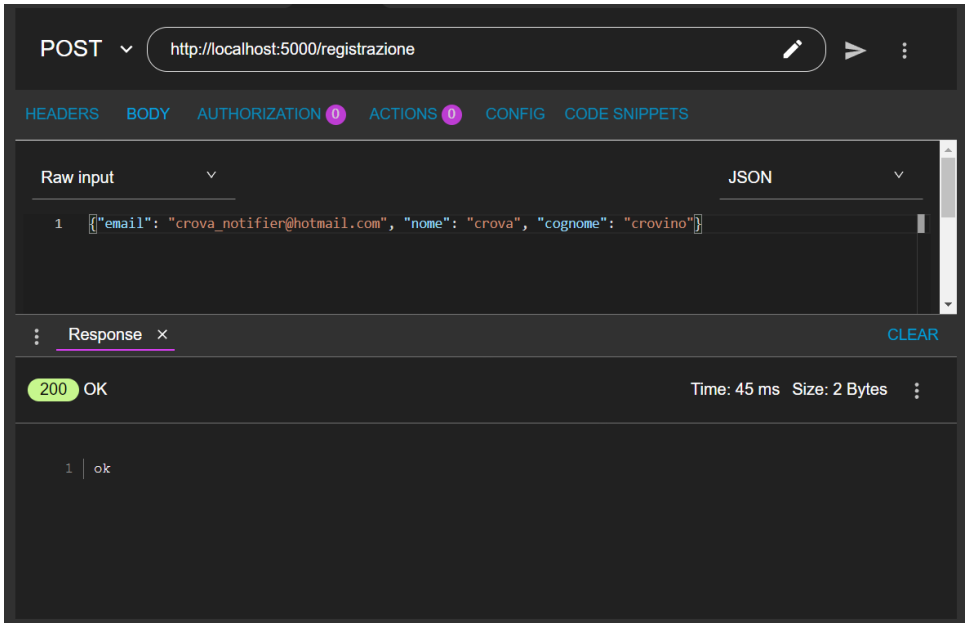
  -X POST \

  -d "{\"email\": \"prova_notifier@hotmail.com\", \"origine\": \"MAD\"}" \

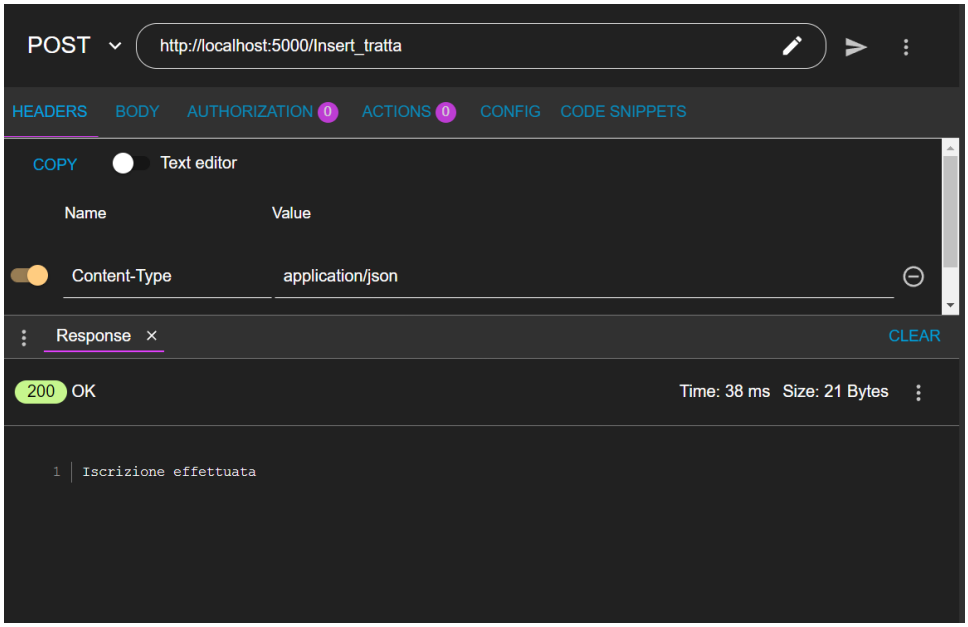
  -H "Content-Type: application/json"

In alternativa è possibile fare richieste tramite un'applicazione come Postman, facendo richieste al localhost sulla porta 5000, inserendo nell'header Content-Type : application/json, come negli esempi sotto:

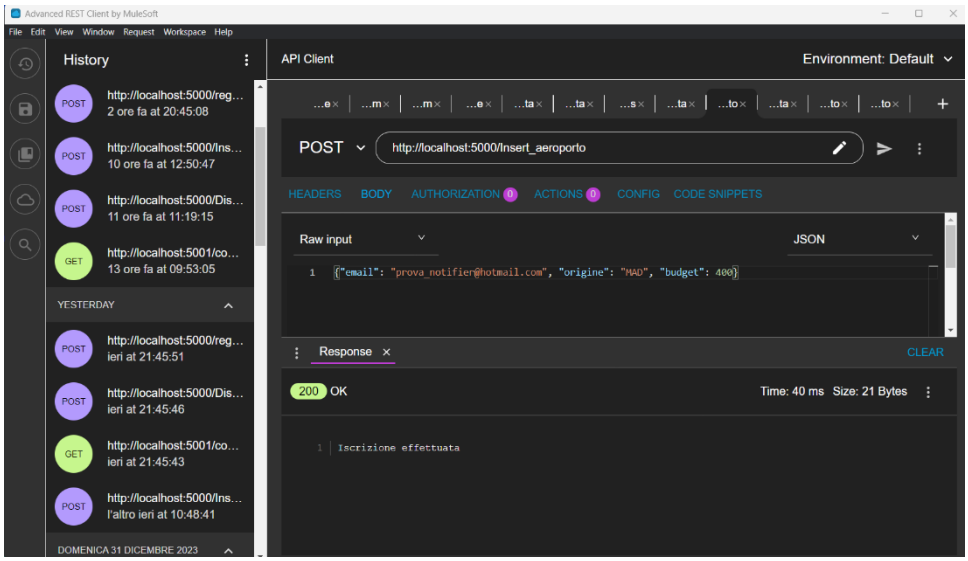
Registrazione



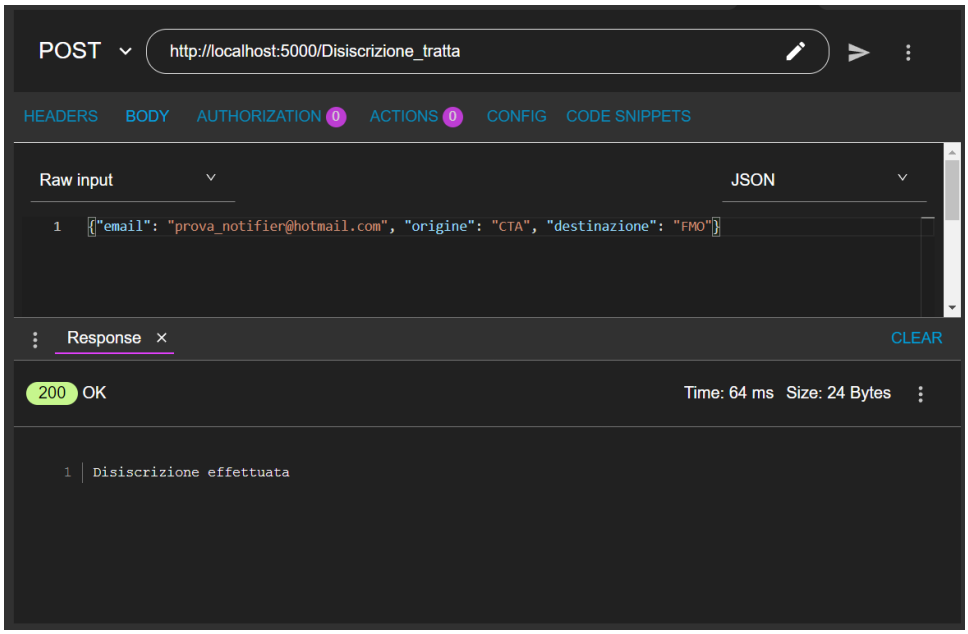
Iscrizione a una tratta



Iscrizione a un aeroporto



Disiscrizione ad una tratta



Disiscrizione ad un aeroporto