Applicazione a microservizi

Ardovino Luca
Matricola: 0345008

Indice

- Introduzione
- Applicazione a microservizi
- Test e risultati
- Conclusioni


INTRODUZIONE

Introduzione: Algoritmo Top Trading Cycle

- L'algoritmo TTC è un metodo elegante e potente per risolvere problemi di matching con diritti di proprietà iniziali (es. House Allocation Problem)
- Ogni partecipante possiede inizialmente un bene, ha preferenze su tutti i beni e si cerca una ridistribuzione efficiente e giusta
- Fasi: costruzione del grafo, identificazione dei cicli, matching parziale, rimozione, iterazione

Introduzione: Algoritmo Top Trading Cycle (2)

- Nel progetto, l'algoritmo è stato usato per realizzare matching stabile in ambito lavorativo
- L'utente target è un laureato in ingegneria informatica a Roma
- L'idea è realizzare un matching stabile per assegnare ad un utente un ambito tra CyberSecurity, DataScience e Software

Introduzione: obiettivi del progetto

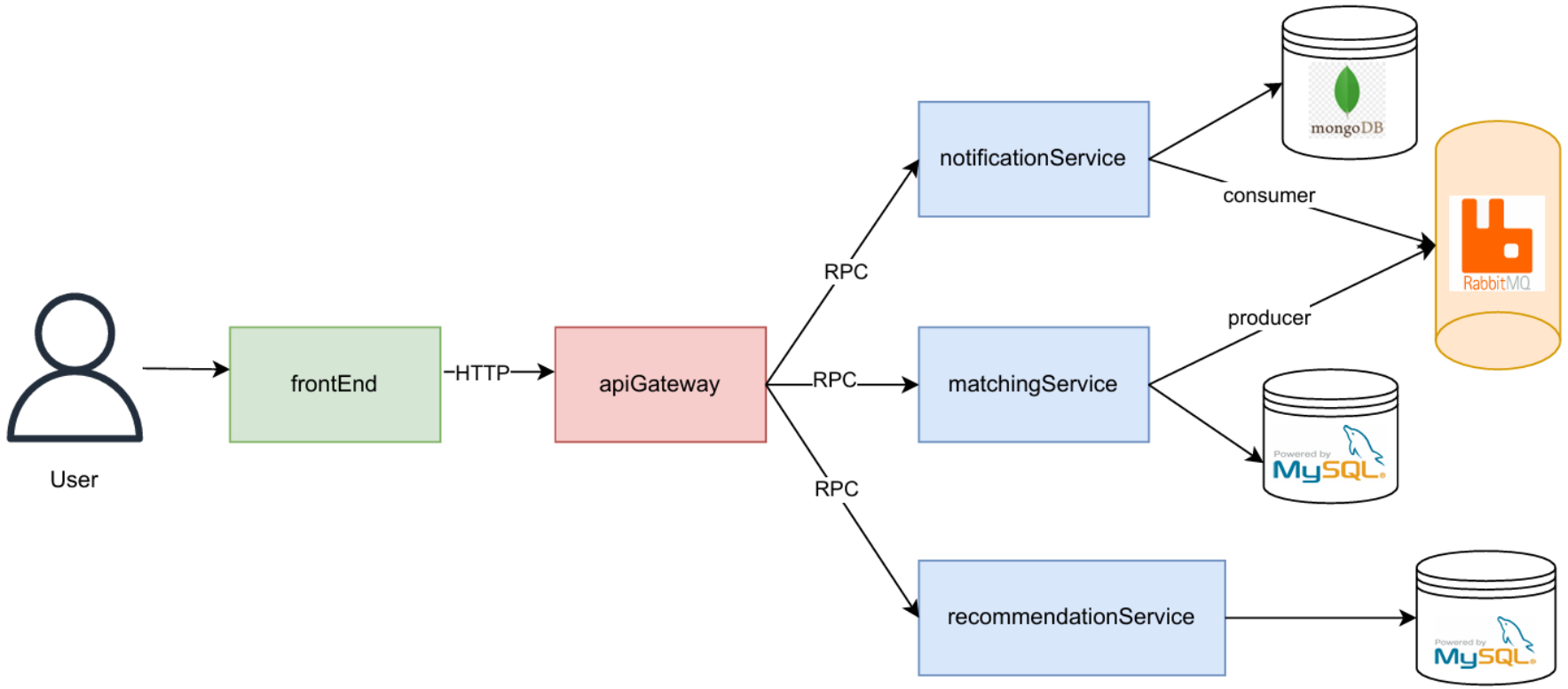
- Implementare un'applicazione a microservizi con almeno 3 servizi (non CRUD), ed uno implementa il TTC
- Applicare pattern affrontati durante il corso
- Analizzare le prestazioni ottenute dal sistema
- Dedurre conclusioni in base ai risultati ottenuti


ARCHITETTURA A MICROSERVIZI

Architettura a microservizi: struttura generale

- 4 microservizi in totale (+ 1 servizio di front-end)
- Go come linguaggio di programmazione
- Chiamate RPC per la comunicazione (chiamate sincrone)
- Docker per la containerizzazione dei servizi (Dockerfile)
- Docker Compose/Kubernetes per l'orchestrazione
- «Database per Service» e «Circuit Breaker» i pattern utilizzati

Architettura a microservizi: diagramma di flusso



Architettura a microservizi: ApiGateway

- Punto d’ingresso unico per le richieste
- Smista richieste ai servizi interni tramite RPC
- Applica il Circuit Breaker ad ogni chiamata
- Comunicazione completamente sincrona

Architettura a microservizi: MatchingService

- Implementa l'algoritmo Top Trading Cycle con l'obiettivo di ottenere un matching stabile tra utenti e aziende
- Suddivisione in categorie per l'esecuzione:
  - Utenti: Triennale, Magistrale < 100, Magistrale > 100
  - Aziende: Cybersecurity, Software, Data Science
- L'utente ordina gli ambiti in base alle proprie preferenze (si esegue TTC con n = 3)
- Il responso dell'algoritmo viene mostrato a schermo

Architettura a microservizi: NotificationService

- Invia la mail all'utente che ha eseguito il matching con l'elenco delle aziende di quel determinato ambito (client SMTP)
- Comunicazione asincrona tramite RabbitMQ:
  - MatchingService: produttore
  - NotificationService: consumatore
- Memorizzazione della mail inviata al proprio DB (MongoDB)
- Funzione di «rispedisci la mail» (disponibile solo dopo il primo matching)

Architettura a microservizi: RecommendationService

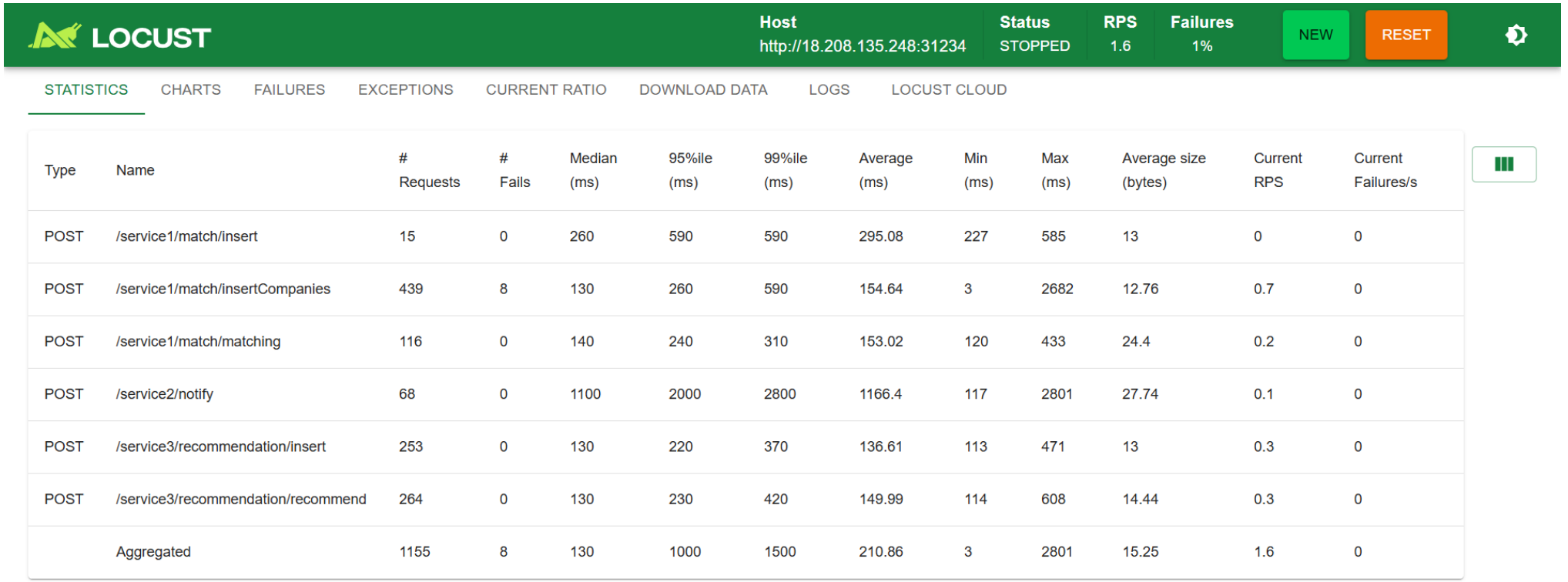
- Funzionalità a disposizione dell'utente solo dopo aver eseguito il matching una prima volta
- Algoritmo di raccomandazione:
  - Voti da 1 a 5 sulle aziende (Collaborative filtering)
  - Basato su similarità del coseno
- Input: tipo di laurea, voto, azienda
- Output: posizione dell'azienda richiesta all'interno della classifica delle aziende raccomandate per quell'utente. Se non presente, quell'azienda è non raccomandata per quell'utente

Architettura a microservizi: pattern

- Database per service:
  - MatchingService: MySQL
  - RecommendationService: MySQL
  - NotificationService: MongoDB
- Circuit Breaker:
  - dopo 3 fallimenti consecutivi il circuito va in stato «open»
  - dopo 10 secondi il circuito passo allo stato «half-open»
  - se il servizio risponde ritorna in «closed», altrimenti resta in «open»

TEST E RISULTATI

Test e risultati



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/service1/match/insert	15	0	260	590	590	295.08	227	585	13	0	0
POST	/service1/match/insertCompanies	439	8	130	260	590	154.64	3	2682	12.76	0.7	0
POST	/service1/match/matching	116	0	140	240	310	153.02	120	433	24.4	0.2	0
POST	/service2/notify	68	0	1100	2000	2800	1166.4	117	2801	27.74	0.1	0
POST	/service3/recommendation/insert	253	0	130	220	370	136.61	113	471	13	0.3	0
POST	/service3/recommendation/recommend	264	0	130	230	420	149.99	114	608	14.44	0.3	0
	Aggregated	1155	8	130	1000	1500	210.86	3	2801	15.25	1.6	0


CONCLUSIONI

Conclusioni

- Semplice ma efficace implementazione dell'algoritmo Top Trading Cycle nel contesto del mercato del lavoro
- Soluzione può essere facilmente estesa ad implementazioni più complesse
- Architettura a microservizi garantisce buone prestazioni