Università degli studi di Roma Tor Vergata

Facoltà di Ingegneria Informatica

Esercizio in Go

Manuale d'uso e breve relazione

Anno Accademico

2019/2020

Giuseppe Lasco

0286045

Breve relazione

L'esercizio consiste nella realizzazione di una coda di messaggi che implementa le due semantiche di delivery: at-least-once e quella basata su timeout di visibilità.

Per la prima è stato realizzato un broker in grado di supportare la ritrasmissione dei messaggi, la cui logica è implementata nel producer. Come da definizione il client può ricevere messaggi duplicati e non in ordine.

Per quanto riguarda la seconda semantica, è stato implementato il servizio basato su timeout di visibilità con approccio pull da parte del client, ovvero questo può ottenere un messaggio dalla coda inviando un ack alla ricezione di quest'ultimo.

Dettagli implementativi:

- L'applicazione supporta un producer e più consumer: nel caso at-least-once delivery, i messaggi vengono smaltiti dai consumer usando un approccio round robin nell'ottica di un'applicazione di computazione distribuita con bilancio del carico. Ogni messaggio è smaltito da un singolo consumer come da definizione di coda di messaggi. Nel caso di delivery basata su timeout di visibilità, più consumer possono leggere lo stesso messaggio se il timeout di visibilità è scaduto, rendendo visibile di nuovo il messaggio, ma l'ack non è ancora arrivato impedendone la cancellazione dalla coda, come da definizione; inoltre supporta idempotenza nel caso di ack duplicati;
- Il broker è interattivo nei confronti dell'utente e restituisce lo stato della coda sotto sua richiesta;
- I parametri di configurazione e la lista dei payload dei messaggi sono salvati in due file json e letti dal programma.
- Il valore del timeout, la semantica e le porte di producer e broker sono modificabili dal file config.JSON
- Per semplicità l'applicazione funziona in locale sull'indirizzo di loopback

Manuale d'uso

Aprire il file config.JSON e scegliere la semantica di delivery tra: "at-least" e "timeout-based" inserendola nell'oggetto in corrispondenza del campo "semantic".

Il file messages.JSON con percorso EsercizioGoLasco/Producer/massages.JSON contiene un array di oggetti json del tipo:

```
{
  "payload" : "payload"
}
```

È possibile inserire più oggetti di questo tipo nell'array e modificarne il contenuto a piacimento; questo array rappresenta la lista di messaggi che il producer invia dopo un'opportuna elaborazione.

Aprire il terminale e spostarsi nella directory Main con percorso /EsercizioGoLasco/Main. Ogni terminale aperto rappresenta un componente dell'applicazione.

Nel caso della semantica at-least-once bisogna:

- Far partire il broker usando il comando **$go run BrokerMain.go**
- Aprire uno o più ulteriori terminal per avviare uno o più consumer digitando il comando **$go run ConsumerMain.go**
- Aprire un ultimo terminale e avviare il producer tramite il comando **$go run ProducerMain.go**
- Per interagire con il broker premere Enter
- Per eliminare un consumer digitare Ctrl + C
- Per chiudere il broker digitare Ctrl + C

Nel caso della semantica timeout-based bisogna:

- Far partire il broker usando il comando **$go run BrokerMain.go**
- Aprire un terminale e avviare il producer tramite il comando **$go run ProducerMain.go**
- Aprire uno o più ulteriori terminali per avviare uno o più consumer digitando il comando **$go run ConsumerMain.go**, i quali adottano un approccio pull ed estraggono il messaggio in testa alla coda.
- Per interagire con il broker premere Enter
- Per chiudere il broker digitare Ctrl + C

Vi è un semplice caso di test per controllare l'effettiva consegna di un messaggio; per eseguirlo bisogna aprire un terminale, spostarsi nella directory Test con percorso /EsercizioGoLasco/Test e digitare il comando
$go run Test.go