Programmazione di sistemi distribuiti

ANNO 2024/25

Laboratorio 5

In questa attività di laboratorio siete invitati a esercitarvi con MQTT, un protocollo di messaggistica publish/subscribe estremamente leggero che oggi viene utilizzato in un'ampia varietà di applicazioni grazie a un meccanismo interno di consegna affidabile dei messaggi e al supporto di reti non affidabili. Per la comunicazione con un browser web, i messaggi MQTT possono essere trasferiti in rete e incapsulati in frame WebSocket (MQTT Over Websockets).

In questo laboratorio, avrete anche l'opportunità di fare pratica con i meccanismi di sincronizzazione client a consistenza debole nelle applicazioni web.

L'attività comprende i seguenti compiti:

- modifica del servizio *Film Manager* sviluppato nel Laboratorio 4 con l'introduzione di un meccanismo publish/subscribe gestito da un broker MQTT per il trasferimento di informazioni sulle selezioni di film;
- adattamento del client React sviluppato nel Laboratorio 4 alle nuove caratteristiche del servizio, compresa l'interazione con il broker MQTT.

Gli strumenti consigliati per lo sviluppo della soluzione sono:

- *Visual Studio Code* (https://code.visualstudio.com/) per l'estensione del *Film Manager* e dell'applicazione React implementando il meccanismo richiesto per la comunicazione MQTT Over Websockets;
- *PostMan* (https://www.postman.com/) per testare la versione estesa di
  Servizio di *Film Manager*;
- *DB Browser for SQLite* (https://sqlitebrowser.org/) per la gestione del database del servizio *Film Manager*;
- un browser web (ad esempio, *Google Chrome*);
- Eclipse Mosquitto (https://mosquitto.org/) per istanziare un broker di messaggi che implementa il protocollo MQTT.

Contesto dell'attività

Nel Laboratorio 4, il servizio Film Manager è stato ampliato con la funzionalità di selezione dei film, in modo che ogni utente possa selezionare un film pubblico, tra i film per i quali è stata rilasciata una recensione a quell'utente, come suo film *attivo* (cioè il film per il quale sta attualmente lavorando alla stesura di una recensione). Per questa operazione necessario introdurre un nuovo vincolo: un film può essere *attivo* per **un solo** utente alla volta. Se un utente cerca di selezionare un film *attivo* per un altro utente, l'operazione fallisce.

Questo nuovo vincolo richiede una sincronizzazione tra i client. In questo caso, la coerenza eventuale è accettabile: si accetta che in alcuni intervalli di tempo un client possa avere una copia non aggiornata delle selezioni dei film, e anche che due utenti possano cercare di selezionare lo stesso film nello stesso momento. In questo caso, è necessario che alla fine, in assenza di altre operazioni, tutti i client siano d'accordo sulle selezioni dei film. Quando un utente tenta di selezionare un film ci sono due opzioni (e si può scegliere quella che si preferisce): (1) la selezione rimane in uno stato di attesa fino a quando non una conferma o un rifiuto della selezione da parte del server (2) la selezione appare immediatamente come attiva all'utente, ma se viene successivamente rifiutata dal server, viene annullata (approccio ottimistico). In entrambi i casi, l'utente deve essere informato del fallimento della selezione con un avviso.

Questa modifica del servizio potrebbe richiedere una modifica della sua API REST, perché l'operazione di selezione del film potrebbe fallire in caso di conflitti.

Inoltre, le informazioni sulle selezioni dei film vengono propagate dal server ai client del Film Manager utilizzando un broker MQTT (ad esempio, Eclipse Mosquitto), basato su un meccanismo di publish- subscribe. Tutti i client si iscrivono agli argomenti associati ai film pubblici (l'argomento di un film è denominato con l'id del film). Ogni volta che le selezioni dei film cambiano, il server pubblica le modifiche agli argomenti dei film. Questi cambiamenti sono pubblicati attraverso messaggi che trasmettono lo stato della selezione del film (cioè identificano l'utente per il quale un film è *attivo*) e la loro struttura è definita nello schema JSON contenuto nel file *mqtt_film_mes- sage_schema.json*. Più precisamente, ogni messaggio trasmette lo stato del filmato (*attivo* se un utente lo ha selezionato, *inattivo* se il filmato non è attualmente selezionato da alcun utente, *abbandonato* se il filmato non esiste più) e, se il filmato è *attivo*, le informazioni relative all'id e al nome dell'utente che lo ha selezionato.

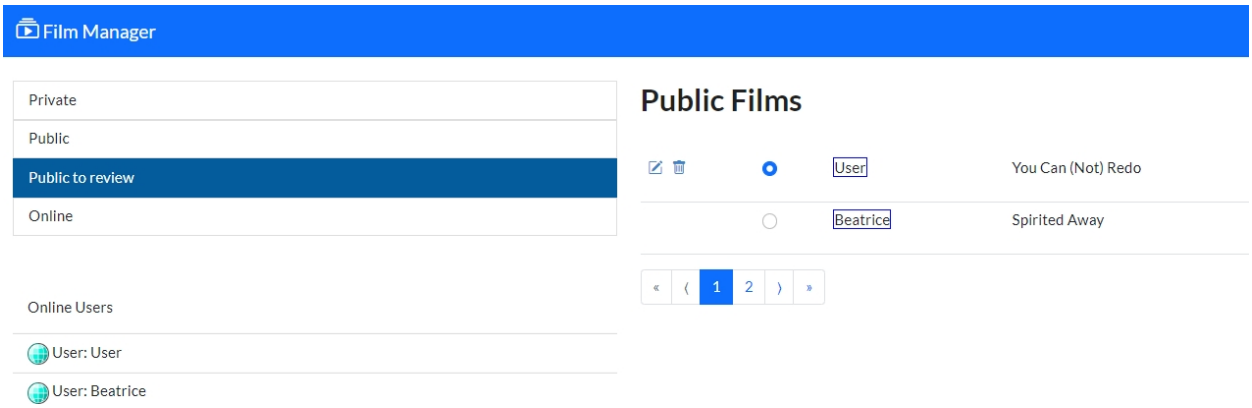
I messaggi MQTT vengono pubblicati dal servizio Film Manager nei seguenti casi:

- Dopo aver stabilito con successo una connessione con il broker, il servizio Film Manager pubblica un messaggio per ogni film pubblico esistente. Ogni messaggio deve avere

il flag retained impostato su true, in modo che il broker invii il messaggio retained per un determinato argomento a un client che si è abbonato a quell'argomento.

- Quando un film diventa attivo per un altro utente, il servizio Film Manager pubblica un nuovo messaggio conservato, comunicando lo stato *attivo* di quel film insieme all'id e al nome dell'utente che lo ha selezionato.
- Quando un film non è più attivo per un utente (ad esempio, un utente ha cambiato la sua selezione da questo film a un altro), il servizio Film Manager pubblica un nuovo messaggio di mantenimento, comunicando lo stato *inattivo* di quel film.
- Quando viene creato un film pubblico, il servizio Film Manager pubblica un nuovo messaggio conservato, comunicando lo stato di *inattività* del film.
- Quando un film pubblico viene cancellato, il servizio Film Manager pubblica un nuovo messaggio conservato per informare i client iscritti di questo evento (ad esempio, con lo stato del film impostato su *cancellato*).

client React, l'interfaccia grafica della pagina *Pubblico da rivedere* viene ampliata con un nuovo elemento in ogni riga di film, come illustrato nella figura seguente. Questo elemento identifica l'utente che ha selezionato quel film come film *attivo*, se presente.



Il client React riceve le informazioni necessarie per la visualizzazione di questo tipo di contenuti attraverso una comunicazione MQTT Over Websockets con lo stesso broker MQTT (cioè Eclipse Mosquitto), con il quale il servizio Film Manager stesso ha stabilito una comunicazione. In particolare:

- Il client React deve sottoscrivere gli argomenti corrispondenti all'id di ogni filmato pubblico per il quale è stata emessa una richiesta di revisione all'utente loggato e che è

attualmente mostrato nella pagina *Pubblico da rivedere* dell'interfaccia grafica. Successivamente, il client React riceverà un messaggio conservato per ogni filmato che soddisfa queste condizioni, contenente lo stato *attivo* di quel filmato insieme all'id e al nome dell'utente che lo ha selezionato, se presente.

- Ogni volta che il client React riceve un messaggio MQTT relativo a uno di questi film, aggiorna lo stato dei film nella pagina *Public to Review* dell'interfaccia grafica.
- Ogni volta che la selezione di un film eseguita dall'utente connesso fallisce (ad esempio, il film è *attivo* per un altro utente), sullo schermo deve essere visualizzato un messaggio di avviso.

Si noti che le informazioni sugli utenti connessi vengono ancora trasferite tramite Websocket, come nel Laboratorio 4.

Come vivere questa attività di laboratorio

Siete invitati a completare entrambi i compiti richiesti per svolgere completamente questa attività di laboratorio (cioè la modifica del servizio Film Manager e del client React con il supporto per la comunicazione MQTT).

Tuttavia, accanto a questo documento, vi :

- il file *mosquitto.conf*, contenente la configurazione del broker Mosquitto MQTT necessaria per abilitare MQTT Over Websockets. Per lanciare Mosquitto con questa configurazione, si deve usare il seguente comando:
  mosquito -v -c mosquitto.conf
- lo schema JSON mqtt_film_message_schema.json, che descrive la struttura dei messaggi scambiati nella comunicazione MQTT;
- una versione estesa del client React fornito come soluzione per il Laboratorio 4, che contiene già l'interfaccia utente estesa necessaria per questo Laboratorio. Questa versione del client React deve essere estesa solo con il supporto per la comunicazione MQTT e con la funzione di aggiornamento dinamico della pagina in cui vengono visualizzate le informazioni ricevute dal broker MQTT. Tutti gli aspetti grafici sono invece già implementati. Si tenga presente che il client React fornito è compatibile con 'implementazione del servizio *Film Manager* fornita come soluzione per La- boratorio 4. Ciò implica che, se si desidera utilizzare questo client con la propria soluzione, è necessario adattarlo al proprio design dell'API REST.

Consigli utili

Di seguito sono riportati alcuni consigli che hanno lo scopo di aiutarvi a completare i compiti richiesti da questa sessione di laboratorio:

- Si invita a utilizzare il modulo node.js *mqtt* (https://www.npmjs.com/package/mqtt) che implementa e supporta il protocollo MQTT. Per abilitare MQTT Over Websockets, l'URL da specificare per la connessione al broker MQTT deve avere lo schema "ws", che identifica il protocollo Websocket (ad esempio, "ws://127.0.0.1:8080").
- Nel client React, la gestione della comunicazione MQTT può essere introdotta nel file *App.js*. Nelle callback che definirete per la comunicazione MQTT, potete utilizzare la seguente riga di codice per aggiornare lo stato relativo alla selezione dei film attivi da parte degli utenti del servizio Film Manager: *displayFilmSelection(topic, parsedMessage);*
  Quando questa funzione viene eseguita, il contenuto della pagina *Pubblico da rivedere* viene aggiornato automaticamente.
  In questa riga di codice, *topic* è una stringa che rappresenta l'id del film (cioè l'argomento per cui è stato ricevuto il messaggio MQTT), mentre *parsedMessage* è l'oggetto JSON recuperato dopo l'analisi del messaggio MQTT. Ogni messaggio MQTT è specificato come descritto nello schema JSON *mqtt_film_message_schema.json*, che viene fornito.

Lavoro opzionale

Nell'attività del Laboratorio 4, la comunicazione tra il servizio Film Manager e il client React sullo stato degli utenti connessi è stata implementata utilizzando Websockets. Ora, come lavoro opzionale, siete invitati a implementare tale comunicazione in MQTT Over Websockets (ad esempio, potete definire argomenti *online* per gli utenti del sistema, a cui ogni istanza client si iscrive, e il servizio Film Manager pubblica gli aggiornamenti).