Programmazione di sistemi distribuiti

ANNO 2024/25

Laboratorio 2

In questa attività di laboratorio, siete invitati a fare pratica con gRPC, un moderno framework open-source ad alte prestazioni che implementa il paradigma della chiamata di procedura remota (*RPC*).

L'attività consiste nell'implementazione di un client gRPC (in node.js) e nella sua integrazione all'interno del servizio *Film Manager* sviluppato nel Laboratorio 1. Questo client dovrà interagire con un server Java gRPC. Questo client dovrà interagire con un server Java gRPC. Le strutture di dati (chiamate messaggi) e i servizi coinvolti nella comunicazione tra il client e il server gRPC saranno descritti in un documento di presentazione.
File .proto (file del buffer di protocollo).

Gli strumenti consigliati per lo sviluppo della soluzione sono:

- *Visual Studio Code* (https://code.visualstudio.com/) per l'estensione del servizio *Film Manager* mediante l'implementazione di un client gRPC;
- *vscode-proto3,* estensione di Visual Studio Code, (https://marketplace.visualstudio.com/items?itemName=zxh404.vscode-proto3), per la definizione dei file del buffer di protocollo;
- *IDE Eclipse per sviluppatori Enterprise Java* per lo sviluppo del server gRPC;
- *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*.

Contesto dell'attività

Il servizio *Film Manager* è stato ampliato con una nuova funzionalità rispetto alla versione del servizio sviluppata per il Laboratorio 1, ovvero offre la possibilità di associare un insieme di *immagini* a ciascun filmato pubblico (ad esempio, screenshot, poster, volantini). Il servizio supporta i seguenti tre tipi di immagini: PNG (.png), JPEG (.jpg), GIF (.gif). Il servizio non ammette altri tipi di immagini.

Il servizio *Film Manager* espone le seguenti API REST, relative all'associazione delle immagini ai film, solo per gli utenti autenticati:

- Un utente può associare una nuova immagine a un film pubblico di sua proprietà, caricando il file dell'immagine sul servizio *Film Manager*. È possibile associare più immagini a un singolo film. Quando il servizio riceve un'immagine con un tipo di supporto accettato, memorizza localmente il file dell'immagine con il suo tipo di supporto e salva le informazioni relative (cioè il nome dell'immagine e l'associazione dell'immagine al film) in una struttura di dati, chiamata per semplicità *immagine*, nel database.
- Un utente può recuperare l'elenco di tutte le immagini associate a un film pubblico di cui è proprietario o di cui è recensore. In questo caso, il servizio restituirà un array codificato in json di strutture dati di immagini. Non è necessario un meccanismo di paginazione per il loro recupero, poiché il caso d'uso tipico affrontato dal progetto è quello di associare un numero limitato di immagini a ciascun film.
- Un utente può recuperare ogni singola immagine associata a un film pubblico di cui è proprietario o di cui è recensore. In questo caso, specificando il tipo di contenuto desiderato tramite l'intestazione *Accept*, l'utente può decidere se recuperare la struttura dei dati dell'immagine (tipo di contenuto json), che non contiene il file dell'immagine, o il file dell'immagine stesso, in uno dei tipi di contenuto dell'immagine supportati (image/png, image/jpg e image/gif). Se l'utente richiede un altro tipo di supporto, l'operazione fallisce.
- Un utente può cancellare un'immagine associata a un filmato pubblico di sua proprietà (in questo caso, il servizio cancella anche i file immagine corrispondenti).

Quando un utente richiede un file di immagine con un determinato tipo di supporto e il servizio *Film Manager* dispone di quel file di immagine già memorizzato localmente con quel tipo di supporto, invia immediatamente il file di immagine all'utente.

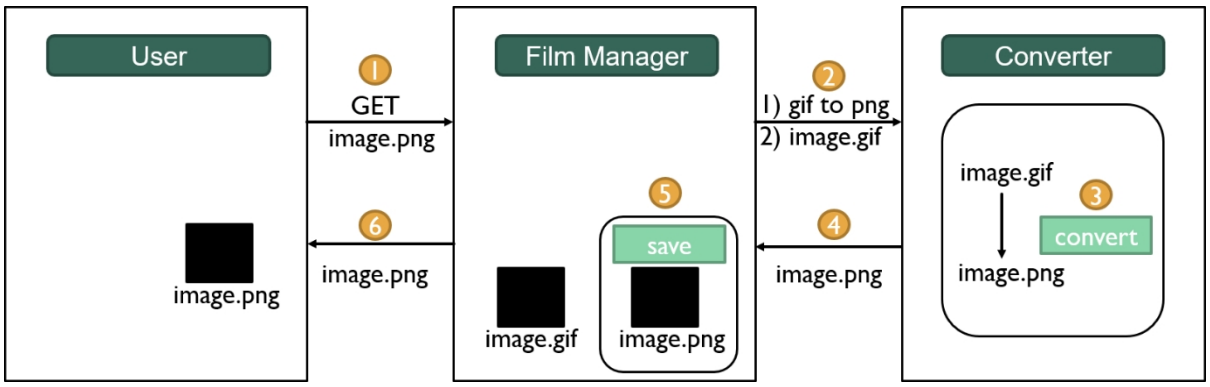
Se invece il file immagine è disponibile localmente solo con un altro tipo di supporto, allora il servizio *Film Manager* interagisce con un altro servizio, chiamato servizio *Converter*, delegando l'operazione di conversione del tipo di supporto. Più in dettaglio, il servizio *Film Manager* utilizza un canale gRPC verso il servizio *Converter* (cioè, il servizio *Film Manager* ricopre il ruolo di client gRPC, il servizio *Converter* ricopre il ruolo di server gRPC). La loro comunicazione è organizzata nel modo seguente:

- il servizio *Film Manager* invia un primo messaggio che descrive il tipo di supporto dell'immagine da convertire e il tipo di supporto di destinazione. Quindi, invia il file immagine, suddiviso in parti.
- il servizio *Converter* converte il file di immagine ricevuto nel tipo di supporto richiesto e lo rimanda al servizio *Film Manager*, se non si verificano problemi nel processo di conversione.

conversione. In caso contrario, notifica il motivo per cui la conversione non è riuscita. Il servizio *Converter* è stateless, cioè non salva il file immagine ricevuto né il risultato della conversione.

Dopo aver ricevuto i pezzi del file immagine convertito e averli memorizzati localmente, il *Film Manager* può finalmente inviare il file all'utente che lo ha richiesto.

L'immagine seguente illustra un esempio di richiesta da parte di un client di un file immagine (ad esempio, image.png) il cui tipo di supporto non è disponibile sul lato server (ad esempio, è disponibile solo image.gif):



Come vivere questa attività di laboratorio

Lo sviluppo completo dell'estensione comporta i seguenti compiti:

1. definizione delle nuove API REST esposte dal servizio *Film Manager*;
2. definizione del file .proto su cui si basa il servizio gRPC;
3. estensione dell'implementazione del servizio *Film Manager* di node.js, per la gestione delle immagini (implementazione delle API REST) e della comunicazione gRPC;
4. implementazione del servizio gRPC *Converter* in Java;

Tuttavia, accanto a questo documento, vi forniamo i seguenti elementi (già discussi nelle lezioni del corso):

- un file .proto che specifica una possibile organizzazione dei messaggi e dei servizi coinvolti nel servizio gRPC *Converter*;
- un'implementazione completa dello stub del server gRPC basato su Java per il servizio *Converter*.

Pertanto, il lavoro di laboratorio consiste nello sviluppo dei passi 1. e 3. Tuttavia, siete caldamente invitati a dare un'occhiata attenta non solo al file .proto, che dovrete utilizzare per implementare il lato client nel passo 3., ma anche al codice del servizio *Converter*, e a comprenderli entrambi. Il motivo è che nell'esame finale potrebbe esservi chiesto di estenderli.

Consigli utili

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

- è possibile utilizzare il modulo node.js *multer* (https://www.npmjs.com/package/multer) per gestire la ricezione e il salvataggio dei file immagine nel servizio *Film Manager*. Questo modulo fornisce un middleware per la gestione di *multipart/form-data*, che è il modo principale usato per caricare i file;
- è possibile utilizzare il modulo node.js *@grpc/grpc-js* (https://www.npmjs.com/package/@grpc/grpc-js) per implementare la funzionalità del client gRPC nel servizio *Film Manager*;
- si può usare il modulo node.js *@grpc/proto-loader* (https://www.npmjs.com/package/@grpc/proto-loader) per caricare i file .proto da usare con gRPC.

Ecco alcune informazioni utili relative al server Java gRPC:

- I file .proto devono essere inseriti nella cartella /src/main/proto del progetto Maven.
- Per generare automaticamente le classi richieste dal file .proto, in Eclipse è necessario sincronizzare la configurazione del progetto con l'impostazione di il pom.xml. Per farlo, seguire le seguenti istruzioni: Cliccare con il tasto destro del mouse sul nome del progetto Eclipse → Maven → Update Project... → OK.
- Il file pom.xml include tutte le dipendenze necessarie per utilizzare gRPC in Java.
- La conformità del compilatore deve essere impostata a 1.6 (o superiore). Il file pom.xml che si trova insieme all'implementazione fornita del server include già la definizione delle proprietà necessarie per impostare il corretto livello di conformità del compilatore a 1.11, che è superiore a quello minimo richiesto.