Relazione progetto finale

Video Server

Distributed Systems and Big Data

C.d.L.M. Ingegneria Informatica

Prof.ssa: **Antonella Di Stefano**

Fugale Dario – *Mat. O55000394*

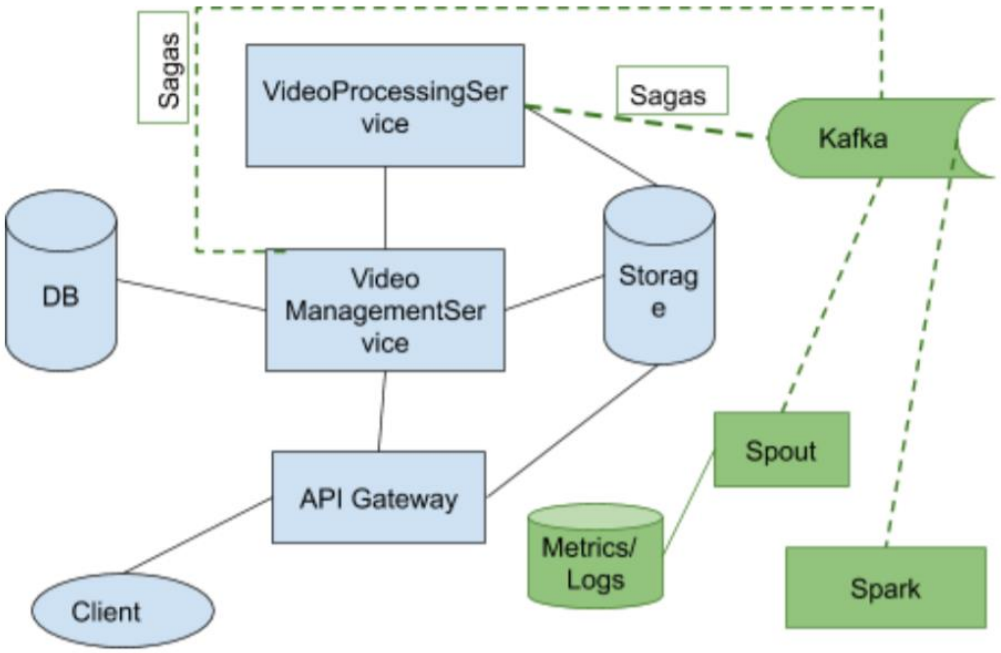
Castagnolo Giulia – *Mat. O55000389*

Introduzione

Per questo progetto finale, abbiamo scelto di implementare il progetto “A” il quale prevede la realizzazione di un **Video Server**, scegliendo di:

- Utilizzare un database ***MongoDB;***
- Implementare l’API Gateway attraverso un’applicazione con ***Spring Cloud Gateway***;

L’architettura generale che è stata realizzata è la seguente:



Homework #1

Per soddisfare le specifiche richieste nel primo Homework, sono stati realizzati dei container Docker al cui interno vengono messe in esecuzione delle applicazioni Spring Boot.

In un primo momento è stato realizzato lo scheletro dell'architettura attraverso il *deploy* dei container al cui interno è stata inserita un'applicazione Spring Boot minimale. È stato nostro interesse infatti, realizzare una *network* (denominata "video-network") sfruttando il driver di rete *bridge* il quale ha permesso di realizzare una rete di interconnessione tra i vari container in pochi passi.

Dopo aver appurato che ogni componente presente nell'architettura potesse comunicare con gli altri secondo lo schema designato, si è proceduto all'implementazione vera e propria dei vari componenti. Nella trattazione procederemo nell'ordine in cui tali componenti sono stati realizzati.

Api Gateway (Spring Cloud Gateway)

L'API Gateway, realizzato attraverso Spring Cloud Gateway, è stato implementato utilizzando una configurazione "Java Based". In particolare, si occupa di instradare le richieste agli opportuni micro-servizi in base alle richieste del client nascondendo così la struttura interna del sistema.

Per far ciò, è stata implementata un'opportuna classe di configurazione denominata *SpringCloudConfig* al cui interno è stato definito un metodo di tipo *RouteLocator* che, in base al predicato dell'URI, contatta l'opportuno servizio interno al sistema.

Il client contatta l'API Gateway (porta 8081) questo implica che ad esempio Video Management Service non viene contattato direttamente dal client ma riceve una richiesta da parte dell'API Gateway.

L'API Gateway ha anche la funzione di catturare le statistiche tramite l'utilizzo di un *GlobalFilter.* Le statistiche che vengono rilevate sono: API richiesta + URI, Payload size input, Payload size output, Tempi di risposta, Eventuali errori, Richieste al secondo. Queste statistiche sono state conservate su un file di testo (*metrics.log*) il quale è stato memorizzato all'interno di un volume. Per quanto riguarda le prime tre statistiche, queste sono state registrate sfruttando i metodi che permettono la lettura dell'header del pacchetto mentre, per quanto riguarda il numero di richieste in un secondo e i tempi di risposta, si è fatto uso di contatori che sono stati opportunamente incrementati.

Video Management Service

Esso rappresenta il micro-servizio principale del sistema (porta 8080). È utilizzato per effettuare le seguenti operazioni:

- Registrazione degli utenti memorizzati nel sistema;
- Visualizzazione degli utenti registrati;
- Inserimento delle informazioni di un nuovo video;
- Upload del video in formato mp4;

Il Video Management Service si occupa, inoltre, di contattare il Video Processing Service, il quale provvederà ad effettuare il processamento del video indicato. Anche questo micro-servizio si trova all'interno della rete "*video-network*" ed è strettamente legato al servizio *MongoDB* (depends_on).

Dal momento che esso rappresenta il nodo centrale del sistema, sono stati inseriti opportuni meccanismi di sicurezza tramite la suite fornita da *Spring Security*: i metodi di tipo POST che permettono l'inserimento delle informazioni dei video all'interno del database e l'upload dello stesso all'interno dello storage, necessitano di autenticazione.

Inoltre, il VMS è l'unico componente che comunica direttamente con *MongoDB* per cui è stata aggiunta una classe di configurazione (*MongoConfiguration*) che ne consente la connessione/comunicazione con esso.

Video Processing Service

Questo micro-servizio (porta 8083) si occupa di effettuare:

- Il processamento del video;
- Il salvataggio del video processato all'interno di un opportuno path nello Storage;
- La comunicazione dell'esito del processamento al video management service il quale si occuperà di aggiornare lo stato del video;

Per effettuare il processamento, si è scelto di salvare (a *runtime*) i comandi di processamento su file (anch'esso memorizzato all'interno di un opportuno path di un volume) e di eseguire codesto script tramite la funzione di java che permette di eseguire comandi shell.

La comunicazione tra i due componenti è sincrona e avviene mediante una richiesta HTTP/REST da parte del VMS verso il VPS (utilizzando *RestTemplate*). A partire dal JSON, viene estratto l'id del video il quale è fondamentale per tutte le operazioni quali controlli e creazione delle rispettive directory all'interno di "var/videofiles".

Storage (Volume)

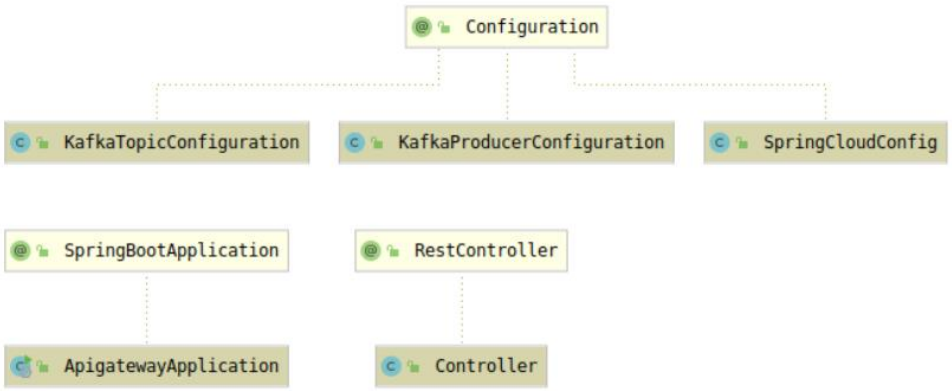
Il componente Storage è stato realizzato attraverso un *volume docker* che viene condiviso tra i 3 componenti software del sistema. All'interno di esso vengono memorizzati i video in formato mp4, i video processati dal VPS e gli script necessari per il funzionamento del sistema (tra cui *videoprocessing.sh*)

Homework #2

Rispetto all'homework 1, sono stati introdotti altri componenti quali Spark, Kafka e sono state introdotte alcune modifiche su quelli preesistenti.

API Gateway: una delle scelte implementative importanti che sono state effettuate è stata quella di predisporre questo componente per la comunicazione diretta con una coda Kafka nella quale vengono inviate nel topic *"metrics-topic*" le metriche relative al **tempo di risposta** e al **numero di richieste** (per secondo). Il *listener* di questo topic è il componente Spark il quale si occupa di effettuare il processamento real time dei dati raccolti. Rispetto all'architettura originale della consegna, la scelta effettuata è stata quella di dare gli oneri dello ***Spout*** direttamente all'API gateway. La motivazione di questa scelta risiede nel fatto che le statistiche vengono scritte su un file e pertanto, al fine di evitare di gestire complessi meccanismi di concorrenza tra i micro-servizi coinvolti, si è deciso di fare inoltrare direttamente da API gateway tali metriche di interesse subito dopo averle raccolte.

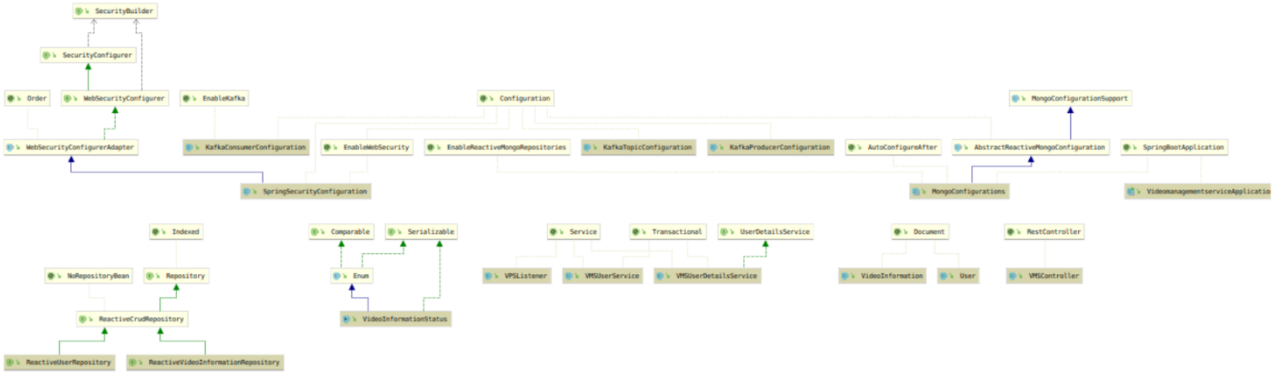
Dopo aver integrato tali funzioni, il componente API Gateway è caratterizzato dal seguente diagramma delle classi:

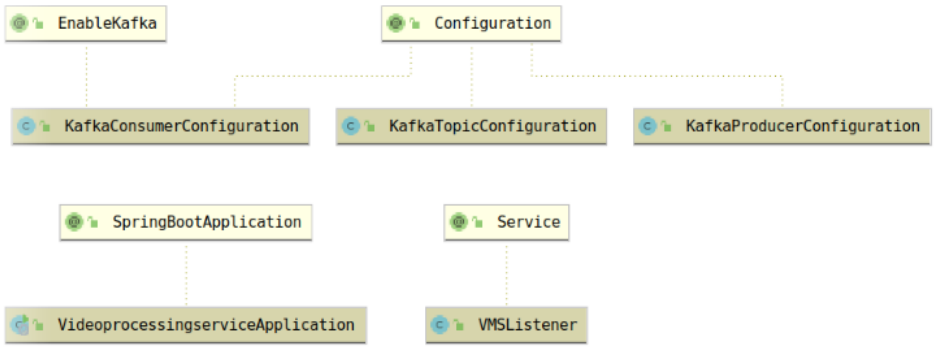


Kafka

Comunicazione VMS – VPS: a differenza dell'homework 1, la comunicazione tra i due componenti è diventata asincrona e viene gestita attraverso una comunicazione broker-based. Sia VPS che VMS fungono sia da Producer che da Consumer di una coda Kafka. Il VMS manda nel *main-topic* di Kafka le informazioni sul video di interesse per il VPS. Nel topic, vengono inviate da VPS informazioni relative all'esito del processamento di uno specifico "videoId". Tali informazioni vengono catturate dal VMS il quale provvederà a settare opportunamente lo stato del video in funzione di quanto appreso dalla coda Kafka.

Le features aggiuntive introdotte nel secondo homework ai componenti VPS e VMS possono essere osservate nei seguenti diagrammi UML delle classi:





Spark

Questo componente recupera i dati dal *metrics-topic* di Kafka e, sfruttando *Streaming Spark,* li divide in diversi batch di 30 secondi. Ogni riga di *DStream* letto viene divisa per il carattere "|" in due parti e vengono filtrate in base al tag della metrica e quindi viene calcolata la media. Il risultato viene memorizzato all'interno di un file contenuto in un volume dedicato a Spark nel path "*/streaming/data.txt*". Il file servirà dunque a contenere lo storico delle medie registrate e a poter effettuare i controlli sugli eventuali incrementi sia sul numero di richieste per secondo che sui tempi di risposta: se dovessero esserci incrementi sostanziali (superiori al 20%) viene attivato il meccanismo di notifica mediante un messaggio sul Bot Telegram.