Riepilogo scelte implementative per l'homework di Distributed Systems and Big Data

Applicazione: (a) Video Server

Variante: DB1 (MySQL), GW2 (Spring Cloud Gateway), STATS1 (API Gateway)

Homework 1

API GATEWAY

Per la realizzazione dell'API Gateway è stato usato Spring Cloud Gateway. Il forwarding delle richieste è stato realizzato dal Bean customRouteLocator che effettua il routing delle richieste modificandone l'URI (sia per le richieste rivolte al VideoManagementService che per quelle rivolte allo storage).

La raccolta delle statistiche è stata effettuata sull'API Gateway sfruttando Prometheus. Per farlo, è stato necessario importare le dipendenze necessarie (micrometer-core, micrometer-registry-prometheus e spring-boot-starter-actuator) e abilitare la raccolta delle statistiche nel file application.properties. Le statistiche possono essere visualizzate alla pagina http://localhost:8080/actuator/metrics/apigateway.requests

VIDEO MANAGEMENT SERVICE

Per la realizzazione dell'applicazione è stato usato Spring Boot, inoltre altre dipendenze usate sono Spring Security e Spring Data JPA. Questa si basa sul pattern MVC, sono presenti delle sottocartelle per distinguere i componenti.

Repository

- **Video Repository:** contiene le operazioni CRUD per la gestione dei Video (contiene un riferimento all'utente autenticato che ha caricato il video).
- **User Repository**: contiene le operazioni CRUD per la gestione degli User.

Services

- **User Service**: implementa l'operazione di registrazione dei nuovi utenti.
- **Storage Service**: implementa le operazioni per la memorizzazione dei file multimediali nello storage.
- **Video Processing Service**: ha la funzione di invocare la procedura remota di elaborazione video presente in Video Processing Service.
- **Video Service**: implementa le operazione per la creazione di nuovi video e modifica dello stato di un video.

Controller

- ***User Controller:*** espone un API POST per la registrazione di un nuovo utente.
- ***Video Controller:*** espone due tipologie di API:

- API POST per la creazione e l'upload di un video.
- API GET per l'accesso in lettura all'elenco dei video presenti e all'indirizzo web in cuoi è possibile trovare un video dato il suo Id.

Autenticazione

L'autenticazione dell'utente è richiesta solamente nelle API POST presenti in Video Controller. Il meccanismo di autenticazione è stato implementato attraverso Spring Security utilizzando come protocollo Basic Authentication. Ogni volta che il client vuole aggiungere o caricare un nuovo video deve inserire il suo username e la sua password nell'header HTTP.

VIDEO PROCESSING SERVICE

L'applicazione è stata realizzata con Spring Boot. Una classe controller, ProcessingController, espone una REST API POST all'url /video/process. La classe ProcessingService consente il processamento dei video.

Homework 2

Le applicazioni sono state adattate alle richieste, di seguito vengono riportate le modifiche effettuate. È stato scritto uno script per l'esecuzione di alcuni comandi (k8s/development/script.sh).

VIDEO MANAGEMENT SERVICE

Il servizio utilizza una coda Kafka per notificare al componente Video Processing Service l'upload di un nuovo video. Ogni volta che l'utente carica un video tramite la relativa API, il servizio crea un messaggio contenete l'id del video e lo inserisce nella coda del topic. Il video viene successivamente processato da Video Processing Service (consumer della coda). Inoltre, VMS si sottoscrive al topic di VPS per essere informato quando l'elaborazione del video è stata completata.

VMS utilizza il pattern Saga per gestire le transazioni che coinvolgono più servizi. L'upload di un video richiede l'esecuzione delle seguenti operazioni:

Operazione	Servizio	Azione	Transazione di compensazione
findByID	VideoService	Viene recuperata l'entity dal database	
storeVideo	StorageService	Viene memorizzato il video nello storage	Viene eliminato il video salvato precedentemente dallo storage
updateStatus	VideoService	Viene eseguito l'update dell'entity nel database	Viene annullata l'operazione di update nel database, ripristinando i valori precedenti.

Se la transazione è stata eseguita correttamente viene inserito il messaggio in Kafka, altrimenti lo stato del sistema risulterà immutato, come se l'operazione di upload non fosse mai stata eseguita.

VIDEO PROCESSING SERVICE

Il controller Rest di Video Processing è stato eliminato. La comunicazione tra VPS e VMS avviene mediante la coda Kafka. Video Processing si sottoscrive al topic di VMS per ricevere i video che devono essere processati. VPS processa i video in maniera asincrona, al termine crea un messaggio contenente l'id del video e l'esito (processed o notprocessed) e lo invia in un'altra coda Kafka (altro topic di cui VMS è il sottoscrittore).

SPOUT

È stato realizzato con Spring Boot, ogni 10 secondi recupera i record di Prometheus esposti alla pagina http://apigateway:8080/actuator/prometheus. Memorizza i record in una lista di Record, classe che tiene traccia delle informazioni sulla richiesta (tipo, URI, stato), del numero di questa specifica richiesta e del tempo totale di risposta. Se ci sono nuovi record, invia un messaggio sulla coda Kafka al componente Spark. Il messaggio è codificato in modo da rendere più semplice il filtraggio dei dati da parte di Spark: il numero di richieste è preceduto dal simbolo #, mentre il tempo di risposta è preceduto da @.

SPARK

È sviluppato in Scala, recupera i messaggi dalla coda Kafka in batch di 30 secondi, li elabora ricavandone la media. Tiene in memoria le ultime dieci medie ricavate per ciascuna metrica, se le ultime misurazioni hanno una variazione significativa (20% in più), viene mandato un alert in un canale Telegram: per farlo, un bot di Telegram controllato dal componente Spark è reso amministratore del canale e pubblica gli avvisi sotto forma di messaggi testuali.