Apache Kafka – SpringBoot –Angular

Kerkesat per kete projekt ishin:

Duhet te kete 2 aplikacione spring-boot.

Aplikacioni i pare do sherbeje si kafka producer

Kerkesat:

1-Te beje tail 2 log file te ndryshem (pra te interceptoje cdo rresht te ri ne kto dy loge), me pas, keto rreshta ti beje publish ne 2 topics te ndryshem ne Kafka (topic te vecante per cdo log file.

Aplikacioni i dyte spring-boot do sherbeje si kafka consumer

Kerkesat:

1- Te beje subscribe ne njeren perj 2 topics qe kerkohet nga user-i (mund te realizohet nepermjet nje api-call nga front end drejt consumerit ku te kerkohet ndryshimi i subscription).

2- Te ruaje cdo rekord qe mer nga kafka ne database (MongoDB)

3- Te beje push nepermjet web socket ne FrontEnd rekordet qe vijne nga kafka.

Pjesa e fundit eshte FrontEnd-i

Kerkesat:

1- Te jete ndertuar me framework-un Angular(6+)

2- Ti shfaqe user-it ne real time rekordin e ri, me keto fusha;  
Content dhe Source (pra nga cili log file vjen ky rekord).   
Gjithashtu ne front duhet te kete nje filter mbi keto fusha.

* Tools qe u perdoren per kete project jane: SpringBoot, Apache Kafka, Kafka Tools, MongoDB, Angular.

# Kafka Producer

Kafka Producer eshte ndertuar si nje aplikacion I pavarur Spring-Boot, ku do te perfaqesohet nje producer qe do te beje publish rekordeve ne topic e tyre perkates. Kemi ndertuar 2 topic te ndryshem me emrat “Topic1” dhe “Topic2”, respektivisht per rekordet e marra nga file1 dhe file2.

## Logging ne file

Ne file *src/main/resources/log4j.properties*  jane bere konfigurimet per secilin log, ku njeri do te shkruaj ne file1 duke marre vetem logger.info dhe ne file2 shkruhen logger.debug .

Per shkrimin ne file eshte krijuar nje metode e cila thirret cdo 5s, pas momentit qe behet run aplikacioni dhe pas shkrimit ne secilin file behet edhe shkrimi ne topic perkates me ane te metodave produceToTopic1() dhe produceToTopic2().

Per te ndertuar producer duhen klasat:

* MessageModel  Kjo klase perfaqeson si do te interpretohet nje mesazh nga topic. Do te kete fushat content(permbajtja e mesazhit) , path(burimi i mesazhit) dhe rowNumber(numri i rreshtit qe ka mesazhi tek log file).
* KafkaConfiguration  Kjo klase permban konfigurime per producer ne menyre qe ne topic te shkruajme nje object te tipit List<MessageModel> dhe mesazhin dhe kthen nje template per secilin, sipas ketyre konfigurimeve.
* LogHandler  Kjo klase ka nje metode static messageList(String,int) e cila merr si parameter path-in e file dhe numrin e rreshtit te fundit te lexuar nga hera paraardhese dhe kthen listen me rreshtat e ri qe do te shkruhet ne topic.
* ProducerController  Ne kete klase ka dy KafkaListener ku secila lexon ne topic dhe ku marrim rekordin e fundit te ruajtur, ka nje metode returnMessage merr si parameter elementin e fundit ne topic ne forme string dhe na kthen nje objekt te tipit MessageModel.
* Log4j.properties  Ne kete file kemi percaktuar konfigurimet per logger-in si dhe krijojme 2 log file me path te percaktuar.

# Kafka Consumer

Eshte gjithashtu nje aplikacion Spring-Boot i pavarur nga producer, i cili do te beje subscribe ne topic per leximin ne real time pas berjes publish. Ky aplikacion behet run ne porten “8081”.

Per te ndertuar consumer duhen keto klasa:

* MessageModel  Kjo klase perfaqeson si do te interpretohet nje mesazh nga topic. Do te kete fushat content(permbajtja e mesazhit) , path(burimi i mesazhit) dhe rowNumber(numri i rreshtit qe ka mesazhi tek log file).
* DBModelList  Kjo klase permban nje liste me objekte te tipit MessageModel dhe perfaqeson nje rresht te dhenash qe do te dergohet ne databaze.
* KafkaConfiguration  Kjo klase permban konfigurime per consumer per te lexuar objekte dhe kthen nje template per secilin, sipas ketyre konfigurimeve.
* KafkaConsumerService  Ne kete klase ka dy KafkaListener te cilat lexojn pas cdo here qe behet publish ne topic1 apo topic2 dhe gjithashtu behet dergimi I mesazhit ne end-point e websocket si dhe shkruhet rekordi ne database. Keto veprime kryhen per te dy topic por jo ne te njejten kohe. Kjo do te varet nga kerkesa e bere ne Client per te zgjedhur topic1 apo topic2.
* ConsumerController  Eshte nje RestController, permban 2 metoda changeTopic dhe insertToDB. Kerkesa e bere nga client do te beje edhe ndryshimin e topic nga ku do te behet subscribe.
* FilterController  Eshte nje RestController nga ku do te merren kerkesat http nga clienti per te kryer nje filter mbi fushat e caktuara content dhe path.Metoda e cila ben filter sipas fushave te plotesuara na kthen listen perkatese.
* WebSocketconfiguration  Kjo klase permban konfigurimet e websocket i cili lidh brokerin me pjesen e front-end.

# Angular

Per te ndertuar pjesen e front-end, jane paraqitur keto klasa:

* App.component.html  Ndertimi I faqes qe do ti shfaqet klientit e cila permban butonate per zgjedhjen e topic nga ku do te beje Subscribe si dhe 2 fusha per filterin.
* App.module.ts  Ne kete klase bejme importet e meposhtme:

BrowserModule,

AppRoutingModule,

FormsModule,

HttpClientModule

* App.component.ts  Kjo klase permban te gjitha funksionet duke filluar nga ngarkimi i faqes deri tek perdorimi i butonave dhe filtrave.
* WebSocketAPI.ts  Kjo kalse permban funksionet per lidhjen dhe shkeputjen me webSocket dhe funksioin qe merr mesazhet e lexuara nga consumer, per tia shfaqur userit.