

## Exercise\_03\_basic\_producer\_consumer

La entrega de la práctica debe ser en formato zip donde debe incluir:

- Funcionalidad de Exercise 03 en formato pdf.
- Ficheros de API Java con las modificaciones.

1.- Escribe el nombre del Topic donde el Producer está enviando los mensajes en la API Java

Topics / quotes-input

Produce Message

Overview

Messages

Consumers

Settings

Statistics

Partitions

1

Replication Factor

1

URP

0

In Sync Replicas

1 of 1

Type

External

Segment Size

62 KB

Segment Count

1

Clean Up Policy

DELETE

Message Count

463

Partition ID

0

Replicas

1

First Offset

0

Next Offset

463

Message Count

463

2.- Describe los mensajes que esta enviando el Producer en la API Java.

	Offset	Partition	Timestamp	Key	Preview	Value	Preview
+	0	0	16/4/2025, 19:27:32			Can one desire too much of a good thing?.	
+	1	0	16/4/2025, 19:27:32			Blow, blow, thou winter wind! Thou art not so unki...	
+	2	0	16/4/2025, 19:27:33			Can one desire too much of a good thing?.	
+	3	0	16/4/2025, 19:27:33			Can one desire too much of a good thing?.	
+	4	0	16/4/2025, 19:27:33			For ever and a day.	
+	5	0	16/4/2025, 19:27:33			How bitter a thing it is to look into happiness thro...	
+	6	0	16/4/2025, 19:27:33			For ever and a day.	
+	7	0	16/4/2025, 19:27:33			The fool doth think he is wise, but the wise man k...	
+	8	0	16/4/2025, 19:27:33			True is it that we have seen better days.	
+	9	0	16/4/2025, 19:27:33			For ever and a day.	

### 3.- Describe qué hace el código de WordCountConsumer

```
static void createWordCountStream(final StreamsBuilder builder) {
    final KStream<String, String> textLines = builder.stream(INPUT_TOPIC);
    final Pattern pattern = Pattern.compile("\\W+", Pattern.UNICODE_CHARACTER_CLASS);
    final KTable<String, Long> wordCounts = textLines
        .map(((key, value) -> {
            System.out.println("VALUE " + value);
            return KeyValue.pair(key, value);
        }))
        .flatMapValues(value -> Arrays.asList(pattern.split(value.toLowerCase())))
        .groupBy((keyIgnored, word) -> word)
        .count();

    wordCounts.toStream().to(OUTPUT_TOPIC, Produced.with(Serdes.String(), Serdes.Long()));
}
```

Este método permite crear una tabla de key value donde key es el identificador del contador de palabras y el value el número que cuenta cuantas palabras tiene la frase.

### 4.- Explica cómo el Consumer obtiene los mensajes según el código de la API Java.

```
try{
    while (true){
        ConsumerRecords<String, String> records = kafkaConsumer.poll(Duration.ofMillis(100));
        for (ConsumerRecord<String, String> record: records){
            System.out.println(" | " + record.key() + " | " + String.valueOf(record.value()));
        }
    }
} catch (Exception e){
    System.out.println(e.getMessage());
} finally {
    kafkaConsumer.close();
}
```

Definimos un ConsumerRecords y recorremos sus records en formato string donde imprimimos el key value del hashmap de tipo string, string del ConsumerRecords.

### 5.-Realiza un pequeño cambio en el código de la API Java de Producer modificando el mensaje que se envía.

```
try{
    for(int i = 0; i < 100000; i++){
        Thread.sleep(100);
        Esports es = faker.esports();
        System.out.println("Event: " + es.event() + " Game: " + es.game() + " League " + es.league() + " Team " +
es.team() + " Player " + es.player());
        String quote = faker.yoda().quote();
        System.out.println(quote);
        kafkaProducer.send(new ProducerRecord<String, String>("quotes-input", quote ));
    }
} catch (Exception e){
    e.printStackTrace();
} finally {
    kafkaProducer.close();
}
```

