**Pregunta 2**

**Sesiones**

scala> :type sc

org.apache.spark.SparkContext

scala> :type spark

org.apache.spark.sql.SparkSession

scala> import org.apache.spark.sql.SparkSession //importamos las librerías necesarias

import org.apache.spark.sql.SparkSession

scala> import org.apache.spark.sql.DataFrame

import org.apache.spark.sql.DataFrame

scala> import org.apache.spark.sql.Dataset

import org.apache.spark.sql.Dataset

//creamos la sesión maestro en la dirección local con el nombre simple-app

scala> val spark: SparkSession = SparkSession.builder().master("local[\*]").appName("simple-app").getOrCreate()

spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@1d60ee76

//cargamos los datos del archivo ejemplo.csv en un dataset del tipo string

scala> val dataSet: Dataset[String] = spark.read.textFile("ejemplo.csv")

dataSet: org.apache.spark.sql.Dataset[String] = [value: string]

//creamos un dataframe para extraer los datos

scala> val df: DataFrame = dataSet.toDF()

df: org.apache.spark.sql.DataFrame = [value: string]

**Streaming**

scala> import org.apache.spark.streaming.StreamingContext

import org.apache.spark.streaming.StreamingContext

scala> import org.apache.spark.SparkContext

import org.apache.spark.SparkContext

scala> import org.apache.spark.streaming.Seconds

import org.apache.spark.streaming.Seconds

**//creamos un contexto y se refresca cada 20 segundos en espera de mensaje**

scala> val streamingContext: StreamingContext = new StreamingContext(spark.sparkContext, Seconds(20))

streamingContext: org.apache.spark.streaming.StreamingContext = org.apache.spark.streaming.StreamingContext@1c585232

scala> import org.apache.spark.streaming.dstream.ReceiverInputDStream

import org.apache.spark.streaming.dstream.ReceiverInputDStream

**//escuchamos el puerto 9090 en espera de algún mensaje**

scala> val lines: ReceiverInputDStream[String] = streamingContext.socketTextStream("localhost",9090)

lines: org.apache.spark.streaming.dstream.ReceiverInputDStream[String] = [org.apache.spark.streaming.dstream.SocketInputDStream@45ae224f](mailto:org.apache.spark.streaming.dstream.SocketInputDStream@45ae224f)

**RDD**

**// creamos un Array de tipo String con tres valores**

scala> val cadenas = Array("Docentes","IntelArtificial","quefinal")

cadenas: Array[String] = Array(Docentes, IntelArtificial, quefinal)

**//generamos una cadenasRDD de tipo String**

scala> val cadenasRDD = sc.parallelize(cadenas)

cadenasRDD: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[0] at parallelize at <console>:34

**//vemos el contenido del RDD**

scala> cadenasRDD.collect()

res0: Array[String] = Array(Docentes, IntelArtificial, quefinal)

**//aplicamos un filtro para que nos muestre en que líneas se repite la palabra “quefinal”**

scala> val filtro = cadenasRDD.filter(line => line.contains("quefinal"))

filtro: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[3] at filter at <console>:33

**//intentamos cargar un fichero que no existe y le pedimos que lo divida en 6 partes**

scala> val fileNotFound = sc.textFile("/7añljdlsjd/alkls/", 6)

fileNotFound: org.apache.spark.rdd.RDD[String] = /7añljdlsjd/alkls/ MapPartitionsRDD[5] at textFile at <console>:33

**//como era de esperarse nos muestra un error, ya que el RDD no existe**

scala> fileNotFound.collect()

org.apache.hadoop.mapred.InvalidInputException: Input path does not exist: file:/7añljdlsjd/alkls

at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:304)

at org.apache.hadoop.mapred.FileInputFormat.listStatus(FileInputFormat.java:244)

at org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:332)

at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:208)

at org.apache.spark.rdd.RDD.$anonfun$partitions$2(RDD.scala:292)

at scala.Option.getOrElse(Option.scala:189)

at org.apache.spark.rdd.RDD.partitions(RDD.scala:288)

at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:49)

at org.apache.spark.rdd.RDD.$anonfun$partitions$2(RDD.scala:292)

at scala.Option.getOrElse(Option.scala:189)

at org.apache.spark.rdd.RDD.partitions(RDD.scala:288)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2293)

at org.apache.spark.rdd.RDD.$anonfun$collect$1(RDD.scala:1021)

at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)

at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:112)

at org.apache.spark.rdd.RDD.withScope(RDD.scala:406)

at org.apache.spark.rdd.RDD.collect(RDD.scala:1020)

... 48 elided

Caused by: java.io.IOException: Input path does not exist: file:/7añljdlsjd/alkls

at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:278)

... 64 more