Contenido

[1. Autores del trabajo, planificación y entrega 2](#_Toc448254544)

[1.1 Autores 2](#_Toc448254545)

[1.2 Planificación 2](#_Toc448254546)

[1.3 Entrega 2](#_Toc448254547)

[2. Requisitos del prototipo a implementar 3](#_Toc448254548)

[2.1 Requisitos funcionales 3](#_Toc448254549)

[2.2 Otros requisitos 3](#_Toc448254550)

[3. Criterios de comparación en la implementación 4](#_Toc448254551)

[3.1 Criterio 1: Nombre del criterio 4](#_Toc448254552)

[3.2 Criterio 2: Nombre del criterio 4](#_Toc448254553)

[3.N Criterio N: Nombre del criterio 4](#_Toc448254554)

[4. Proyecto de implementación de un prototipo del sistema utilizando la tecnología A 5](#_Toc448254555)

[4.1 Documentación de diseño 5](#_Toc448254556)

[4.2 Documentación de construcción 5](#_Toc448254557)

[4.3 Documentación de pruebas 5](#_Toc448254558)

[4.4 Documentación de instalación 5](#_Toc448254559)

[4.5 Manual de usuario 5](#_Toc448254560)

[5. Proyecto de implementación de un prototipo del sistema utilizando la tecnología B 6](#_Toc448254561)

[5.1 Documentación de diseño 6](#_Toc448254562)

[5.2 Documentación de construcción 6](#_Toc448254563)

[5.3 Documentación de pruebas 6](#_Toc448254564)

[5.4 Documentación de instalación 6](#_Toc448254565)

[5.5 Manual de usuario 6](#_Toc448254566)

[6. Comparación de las dos implementaciones 7](#_Toc448254567)

[6.1 Evaluación de los criterios en la implementación usando la tecnología A 7](#_Toc448254568)

[6.2 Evaluación de los criterios en la implementación usando la tecnología B 7](#_Toc448254569)

[7. Comparación de la implementación de las tecnologías 8](#_Toc448254570)

[8. Conclusiones 10](#_Toc448254571)

# 1. Autores del trabajo, planificación y entrega

## 1.1 Autores

En este apartado se debe indicar el número de grupo y los nombres de los autores, poniendo en primer lugar al coordinador del grupo.

## 1.2 Planificación

En este apartado se debe incluir un enlace (URL) compartido a la planificación del trabajo utilizando una herramienta online de diagramación Gantt (por ejemplo, GanttPro, versión gratuita).

Hay que tener en cuenta que cada participante del grupo debe tener asignadas tareas que sumen al menos 45 horas. El peso de este trabajo en la calificación total de la asignatura es de un 30%, por tanto requiere de una dedicación de 45 horas del total de 150 horas de la asignatura.

## 1.3 Entrega

En este apartado debe incluirse un enlace (URL) a un repositorio en GitHub o en BitBucket creado para el trabajo.

En dicho repositorio debe encontrarse, al menos los siguientes archivos en la rama máster:

* Informe del trabajo: con el nombre TG3\_final.docx
* Presentación del trabajo: TG3\_final.pptx
* Prototipos obtenidos implementando cada una de las tecnologías (deben incluir el código fuente y todos los archivos necesarios para la instalación y uso de cada prototipo):
  + PrototipoTecnologiaA\_final.zip (o .rar)
  + PrototipoTecnologiaB\_final.zip (o .rar).

Dichos archivos serán los que se tendrán en cuenta para la calificación del trabajo.

# 2. Requisitos del prototipo a implementar

El objetivo del proyecto es comparar la implementación de un mismo prototipo de sistema utilizando dos tecnologías diferentes (A y B).

Es importante cumplimentar este apartado antes de empezar a implementar el prototipo de cada tecnología, porque ambos prototipos deben cumplir los requisitos que se establezcan en este apartado. Si se van a crear dos equipos de trabajo, uno para cada prototipo, el contenido de este apartado es lo que han de compartir ambos equipos como punto de partida.

Cuanto más detallados sean los requisitos, mayor será la precisión en la comparación que se realizará al final del trabajo. Se trata de conseguir dos prototipos con igual funcionalidad, pero utilizando diferentes tecnologías.

Se puede dar libertad a los equipos de desarrollo en cuanto al diseño, pero la funcionalidad debe ser lo más parecida posible. Por ejemplo, no es necesario que los colores utilizados en las pantallas sean exactamente los mismos en ambos prototipos, a no ser que los miembros del grupo lo hayan decidido así, en cuyo caso, esos detalles de colores deben incluirse en el catálogo de requisitos, para que ambos equipos los cumplan.

## 2.1 Requisitos funcionales

Los requisitos funcionales deben ser los mismos para las dos implementaciones.

En la siguiente tabla se indicará el catálogo de requisitos funcionales del sistema.

| **REQ.** | **DESCRIPCIÓN** |
| --- | --- |
| RF01 | …. |
| RF02 | …. |
|  |  |

## 2.2 Otros requisitos

Se pueden incluir aquí otros requisitos para el prototipo que no puedan considerarse como funcionales. Por ejemplo, requisitos de datos, de seguridad, de interfaz de usuario, de rendimientos, etc.

Se puede dejar libertad

En la siguiente tabla se indicará el catálogo de requisitos no funcionales del sistema.

| **REQ.** | **DESCRIPCIÓN** |
| --- | --- |
| R01 | …. |
| R02 | …. |
|  |  |

# 3. Criterios de comparación en la implementación

En el trabajo TG2 se definieron criterios de comparación de las dos tecnologías a nivel teórico.

En este trabajo hay que definir criterios para la comparación de la implementación de las tecnologías en la construcción del prototipo de sistema de ejemplo, cuyos requisitos son los establecidos en el apartado 2.

Se trata de criterios del tipo” “horas empleadas en el desarrollo del sistema”, “velocidad de funcionamiento del sistema”, “recursos necesarios”, etc.

## 3.1 Criterio 1: Nombre del criterio

Por cada criterio hay que indicar el nombre, una breve descripción, y el tipo de valor a asignar al criterio.

Por ejemplo, si se comparan dos herramientas CASE realizar el diseño UML de un mismo sistema, un criterio podría ser:

*Nombre del criterio: Tiempo de creación del diagrama de clases del sistema.*

*Descripción: Horas invertidas en la creación del diagrama de clases utilizando el editor de la herramienta.*

*Tipo de valor: Numérico (horas).*

## 3.2 Criterio 2: Nombre del criterio

## 3.N Criterio N: Nombre del criterio

# 4. Proyecto de implementación de un prototipo del sistema utilizando la tecnología A

Se trata de incluir en este apartado la documentación del desarrollo del proyecto de implementación, utilizando la tecnología A, del sistema cuyos requisitos funcionales se enumeraron en el apartado 2.

## 4.1 Documentación de diseño

Hay que incluir la descripción del diseño del prototipo, incluyendo diagramas, y el diseño de la interfaz de usuario.

## 4.2 Documentación de construcción

Hay que incluir una descripción de la construcción del prototipo, incluyendo algún extracto de código fuente. No es necesario todo el código. Sólo algún extracto para ver cómo se ha comentado.

## 4.3 Documentación de pruebas

Casos de prueba establecidos y resultados de las pruebas y acciones de corrección. No es creíble que no hayan aparecido errores en los caso de prueba.

## 4.4 Documentación de instalación

Descripción suficiente para que una persona que no ha participado en el proyecto pueda instalar el prototipo.

## 4.5 Manual de usuario

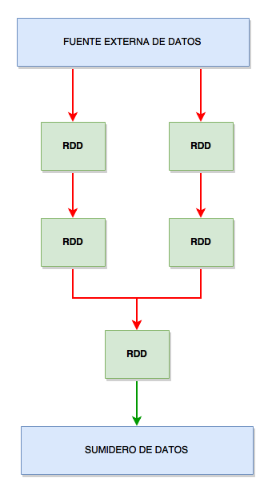
Descripción suficiente para que una persona que no ha participado en el proyecto pueda utilizar toda la funcionalidad que ofrece el prototipo. Que debe coincidir con los requisitos funcionales incluidos en el apartado 2.

# 5. Proyecto de implementación de un prototipo del sistema utilizando la tecnología Spark

Se trata de incluir en este apartado la documentación del desarrollo del proyecto de implementación, utilizando la tecnología B, del sistema cuyos requisitos funcionales se enumeraron en el apartado 2.

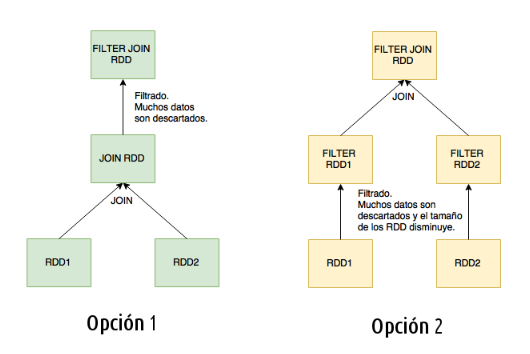
## 5.1 Documentación de diseño

El prototipo se ha diseñado de tal forma que el funcionamiento fundamental corresponde al mostrado en el siguiente diagrama, donde los nodos son objetos RDD y las operaciones entre ellos son operaciones de transformación (representadas por líneas rojas) u operaciones (línea verde):



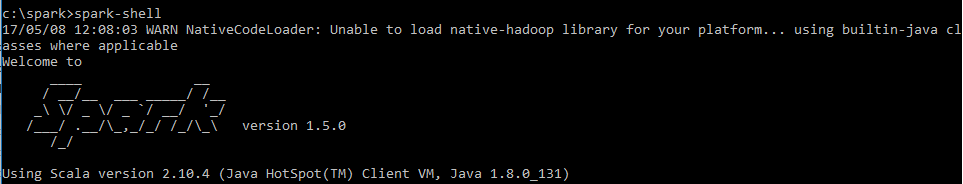
Los objetos RDD iniciales leen datos de ficheros, bases de datos u otras fuentes de información. Una vez creados los RDD iniciales, pasan por una serie de operaciones de trasformación en término de programación funcional, donde no se eliminan ni se alteran, simplemente crea otros RDD con los que trabajar más fácilmente, para, finalmente, converger en un mismo RDD.

La convergencia de los RDD puede ser de 2 formas:



La opción 2 es mucho más efectiva ya que filtra los datos de cada RDD antes de convergerlos, eliminando datos innecesarios a la hora de mezclar, no como la opción 1 que antes de filtrar los datos, realiza la convergencia con la totalidad de estos.

En cuanto la interfaz de usuario, como mencionaremos en el apartado de instalación, Spark trabaja desde el Símbolo del sistema.



Mediante una serie de comandos se cargarían los datos necesarios y el programa a utilizar para obtener el resultado final. Dichos comandos se mencionan más adelante.

## 5.2 Documentación de construcción

El prototipo diseñado está creado en Scala y su función principal es hacer un conteo exacto de las palabras existentes en un documento. Esto es muy útil y práctico a la hora de comprobar requisitos de trabajos o proyectos en los que exigen limitaciones de extensión en número de palabras.

A continuación, mostramos el código que se ha diseñado:

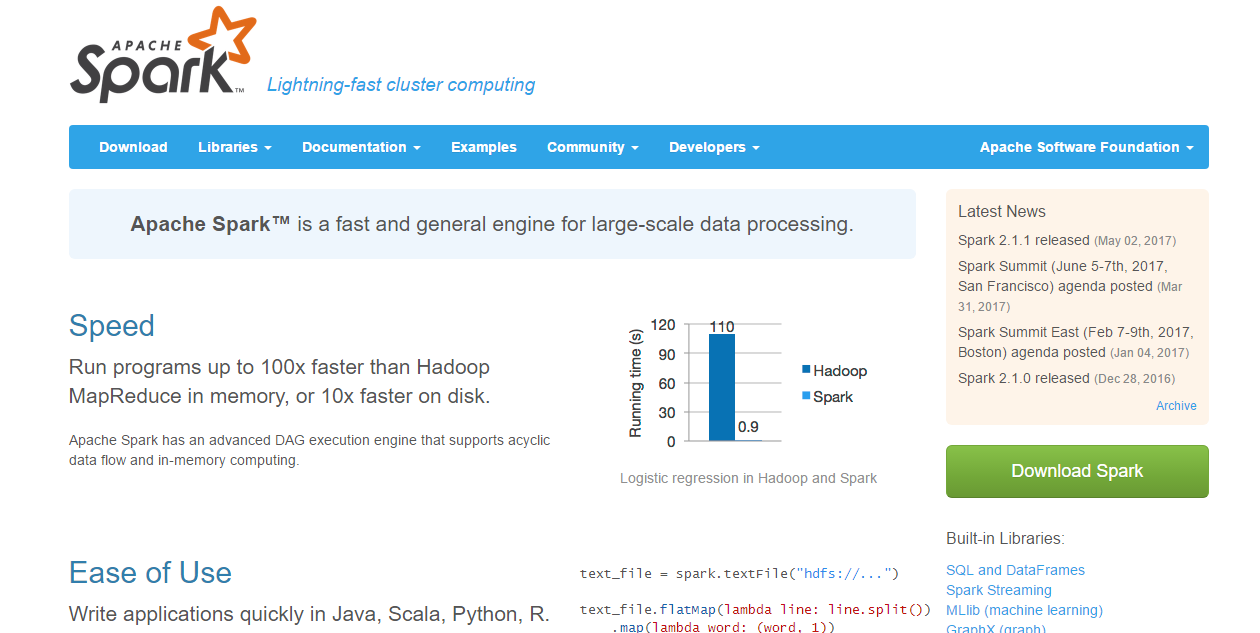
## 5.3 Documentación de pruebas

Casos de prueba establecidos y resultados de las pruebas y acciones de corrección. No es creíble que no hayan aparecido errores en los caso de prueba.

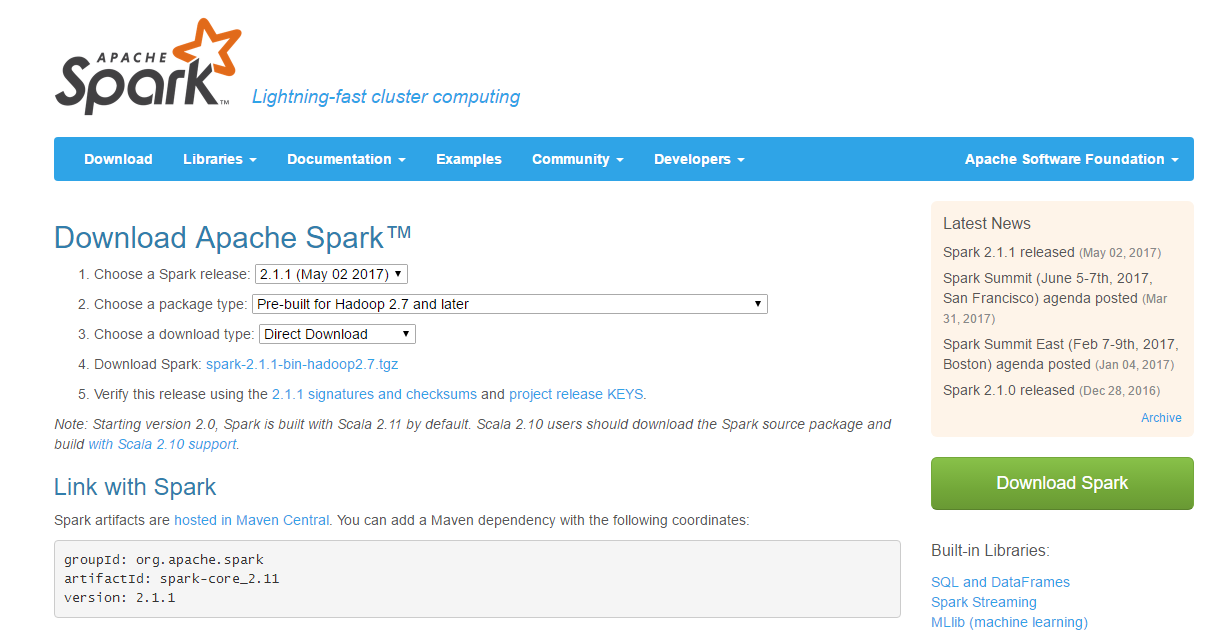
## 5.4 Documentación de instalación

Para comenzar, debemos descargar Apache Spark en la siguiente dirección: <http://spark.apache.org/>

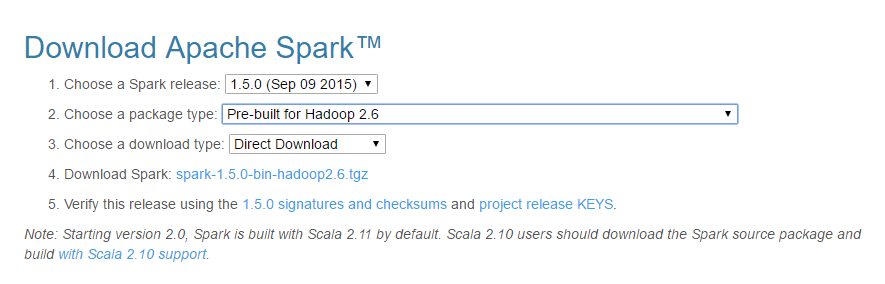
Una vez estemos en la página oficial de Apache Spark, nos dirigimos a la pestaña de descargas, como se muestra a continuación:



Una vez dentro, nos aparecerá la siguiente página:



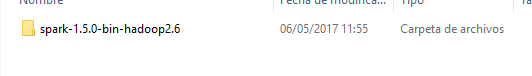
En ella, podemos ver una serie de desplegables que deberemos completar para descargar la versión de Apache Spark que deseemos:



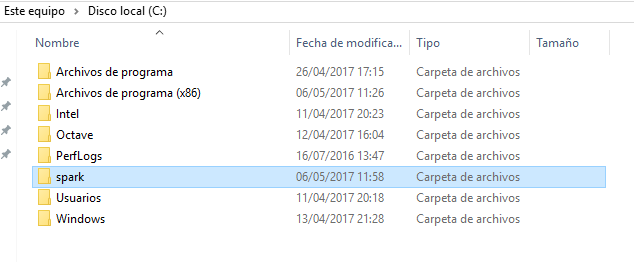
En nuestro caso, hemos elegido La versión 1.5.0 con la pre-construcción para Hadoop 2.6 en una descarga directa. Una vez hemos completado todos los campos, nos disponemos a descargar el software pulsando el hipervínculo que aparece en el paso 4 y comenzará la descarga.



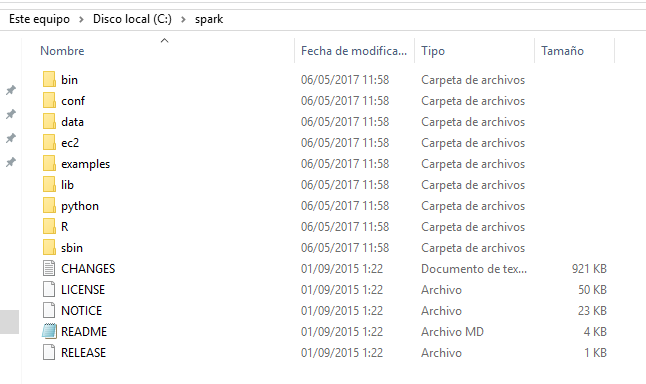
Una vez finalizada la descarga, procedemos a extraer en Spark la carpeta completa, y obtendríamos la siguiente carpeta:



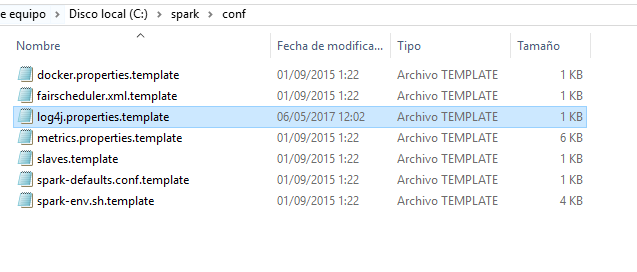
A continuación, por motivos de eficiencia y comodidad (como veremos más adelante), creamos una nueva carpeta Spark directamente en C:\.



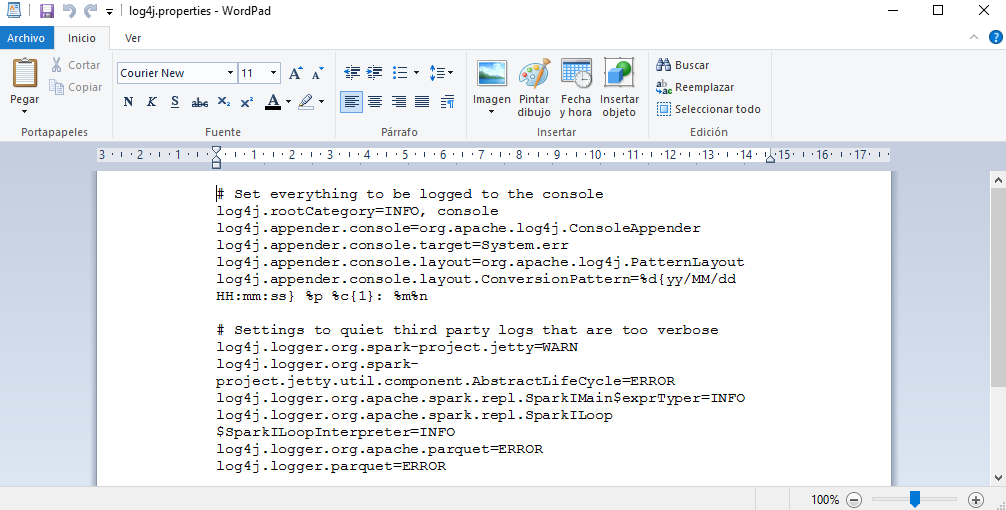
Cuando hayamos creado la carpeta, procedemos a copiar todos los archivos obtenidos en la carpeta extraída de la descarga anterior y los pegamos en la nueva carpeta spark que hemos creado en C:\.

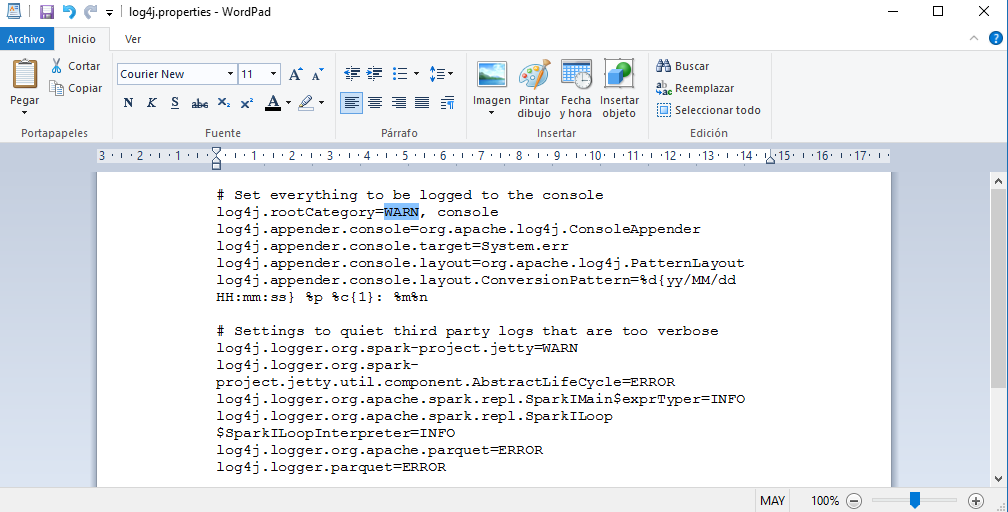


En la carpeta conf, nos disponemos a modificar el log4j para evitar futuros problemas de registro en la consola, en lugar de mostrar en todo momento como información, mostrarlo como advertencia sólo cuando sea necesario.

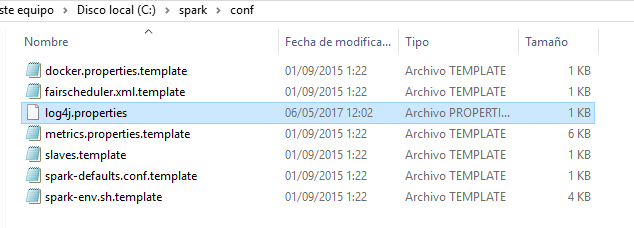


Para ello, abrimos el archivo log4j con el WordPad, por ejemplo, y en lugar de donde pone INFO, lo sustituimos por WARN, como se muestra en las siguientes imágenes y, para finalizar este paso, guardamos y cerramos el archivo.

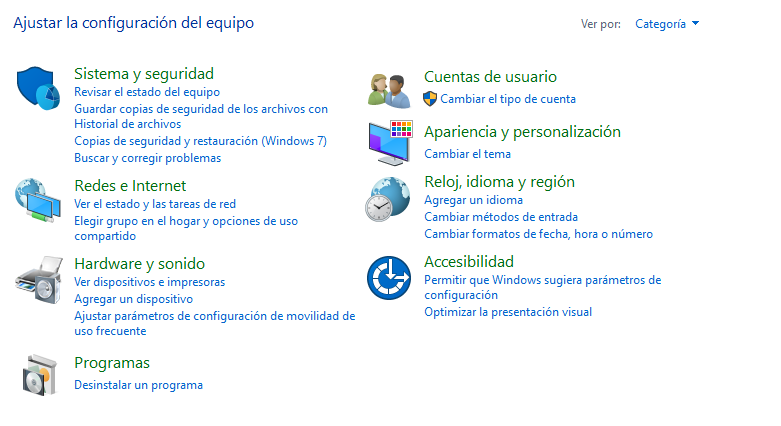




Para conseguir que este cambio tenga algún efecto, es importante no olvidar la extensión .template y dejarlo únicamente con la extensión .properties, como podemos ver a continuación:



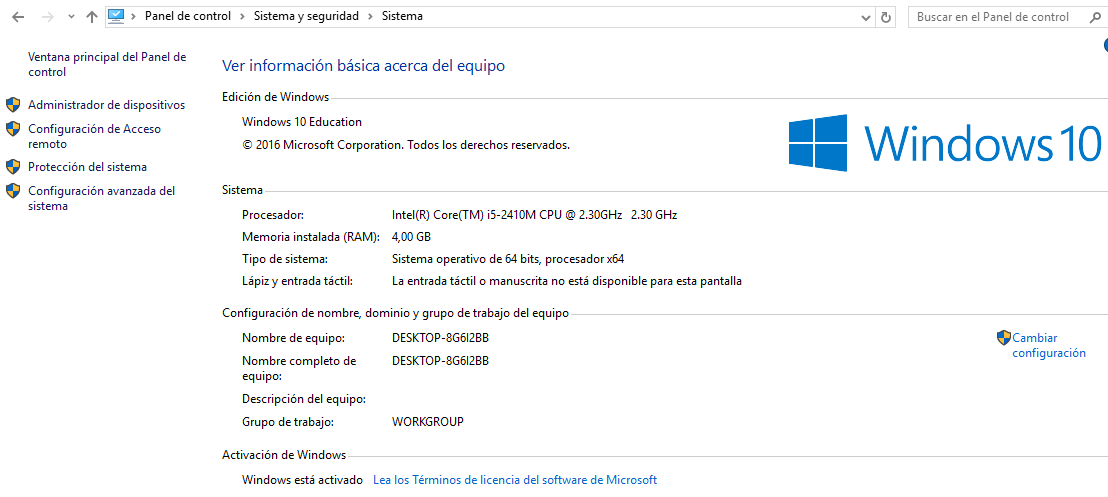
Ahora, debemos añadir Spark a las variables del entorno del sistema. Para ello nos dirigimos al Panel de Control, a la sección de Sistema y seguridad.



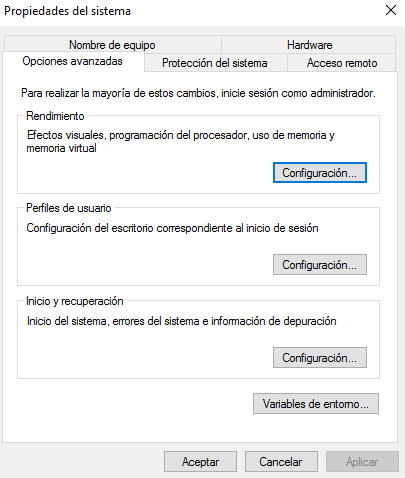
Una vez ahí, vamos a la parte de Sistema:



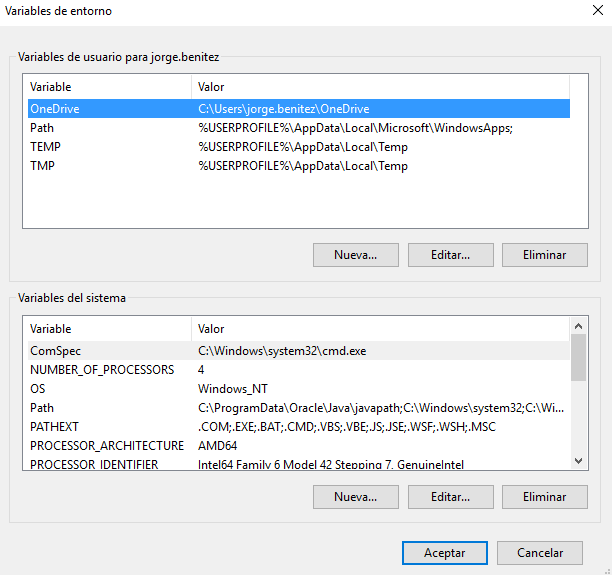
Una vez dentro, nos fijamos en la parte de la izquierda y vamos a la parte de Configuración avanzada del sistema:



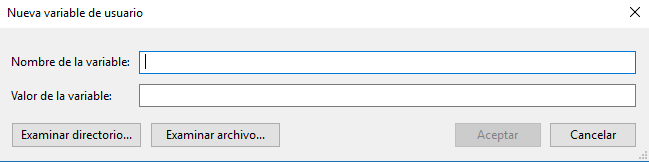
Nos aparecerá la siguiente ventana donde tendremos que pulsar en el botón Variables del entorno…:



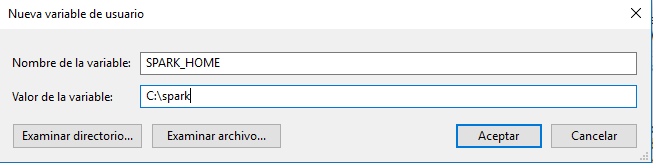
Nos aparecerá la siguiente ventada, donde debemos pulsar el botón Nueva… para poder añadir una nueva variable de entorno para Spark.



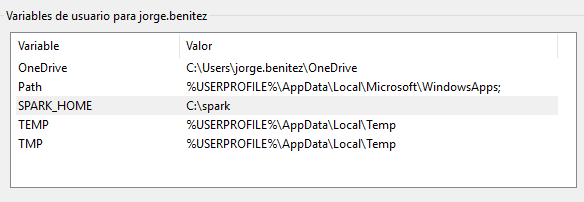
Nos aparecerá la siguiente ventana, que deberemos completar:



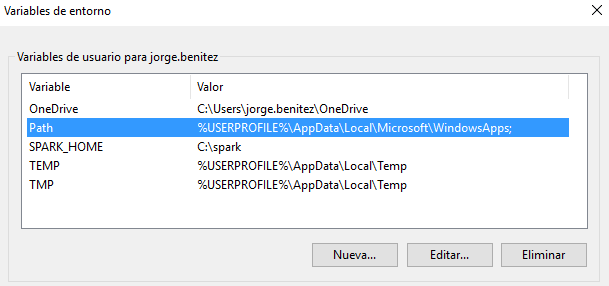
La completamos de la siguiente manera, introduciendo un nombre para la variable y la dirección donde se encuentra su valor (la localización de la carpeta creada en C:\), como se muestra a continuación:



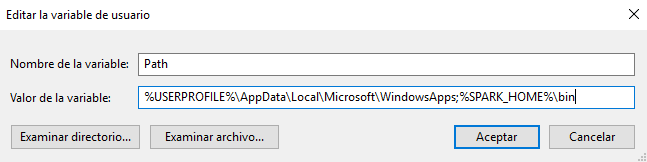
Una vez creada, se verá la nueva variable junto las existentes anteriormente.



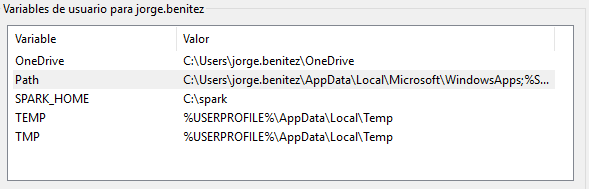
A continuación, también debemos modificar Path para que Windows pueda llamar a la nueva variable que hemos creado.



Para ello, añadimos lo siguiente:

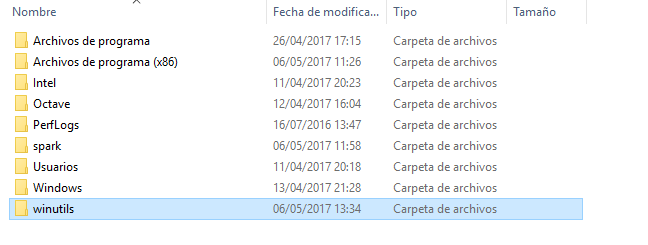


Y pulsaremos Aceptar para que se guarden los nuevos cambios realizados.

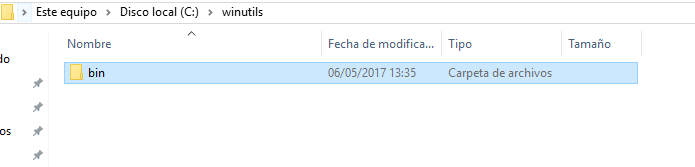


Como ya sabemos, Spark puede trabajar en conjunto con Hadoop, hasta el punto que necesitamos Hadoop para conseguir una correcta instalación. Pero como la presente práctica trata del comparar el funcionamiento por separado de ambas herramientas, debemos descargar un winutils.exe para evitar la necesidad de Hadoop en la instalación de Spark.

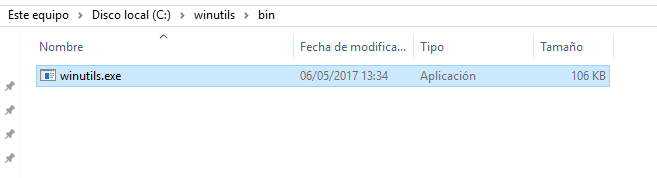
Para ello, creamos otra carpeta que llamaremos winutils en C:\, el mismo lugar donde anteriormente creamos una carpeta para Spark.



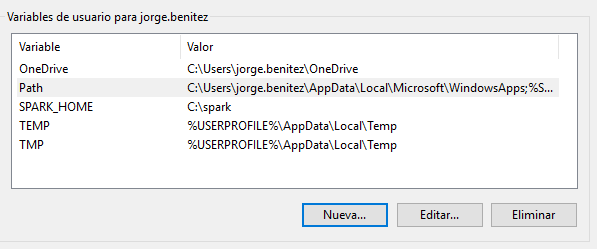
Dentro de esa carpeta, creamos otra que llamaremos bin.

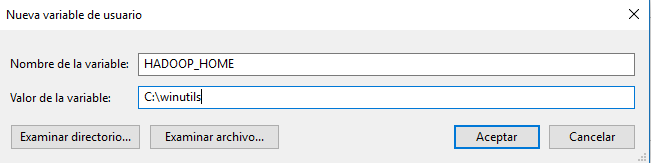


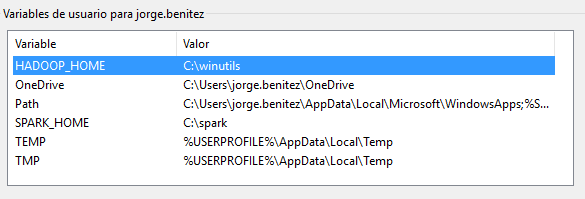
Y dentro de la carpeta bin, guardaremos winutils.exe.



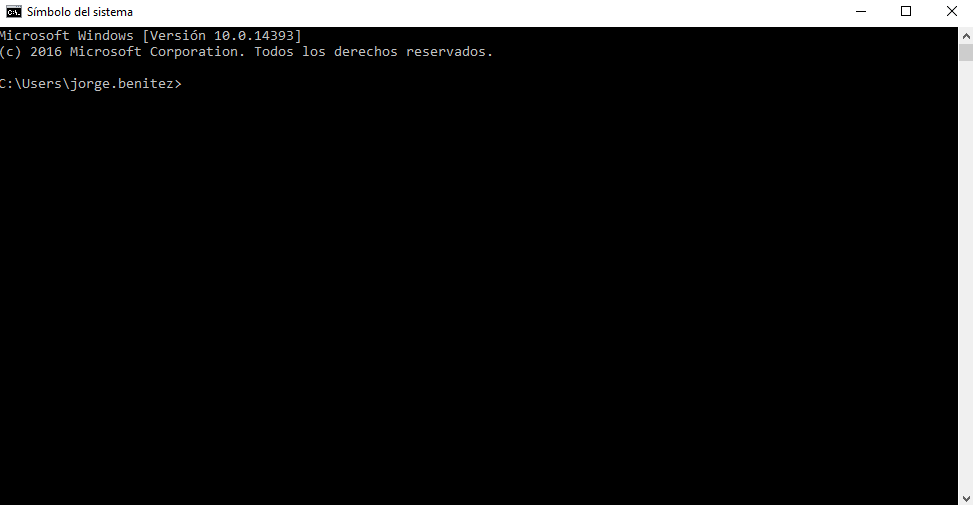
Una vez finalizados estos pasos, debemos crear una variable de entorno para winutils, al igual que hicimos anteriormente con Spark y siguiendo el mismo procedimiento, como se muestra a continuación:







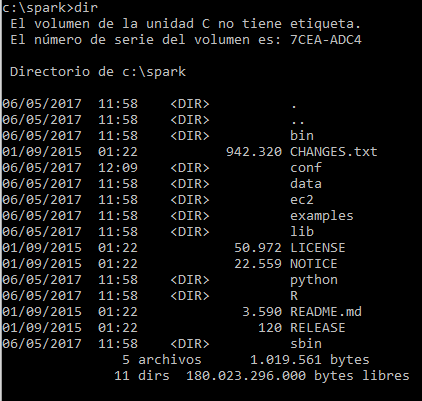
Una vez hayamos realizado todos los preparativos para la ejecución de Spark, abrimos una ventana de símbolo del sistema.



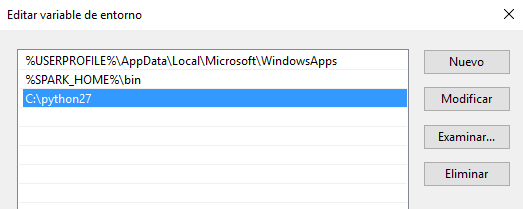
Ahora, utilizando el comando cd, nos movemos hasta la carpeta creada de Spark en C:\.

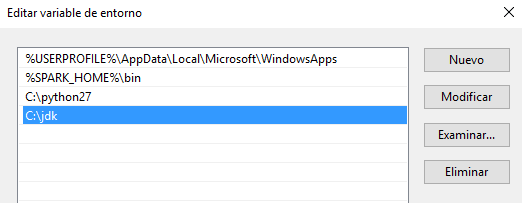


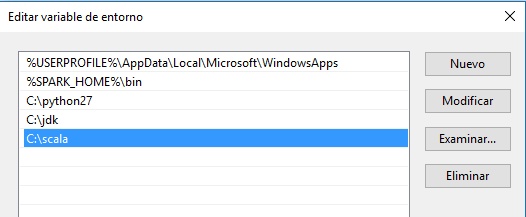
Una vez ahí, si utilizamos el comando dir, podemos ver su contenido. El resultado lo podemos ver en la siguiente imagen:



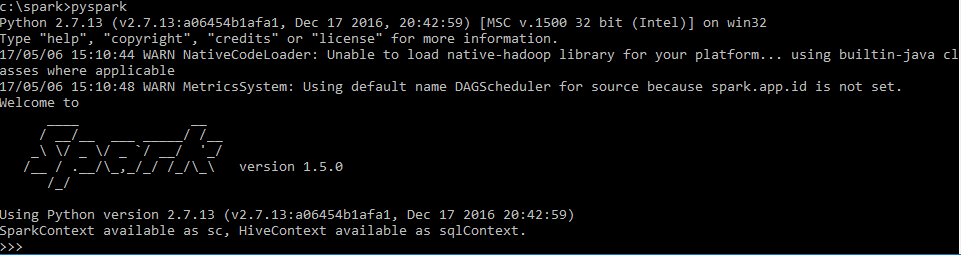
Previamente deberemos haber instalado Python 2.7 y haber hecho lo siguiente (lo mismo con el JDK y con Scala):



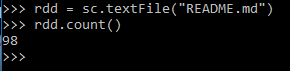




A continuación, iniciamos Spark:



Realizamos una pequeña comprobación para ver si funciona correctamente:

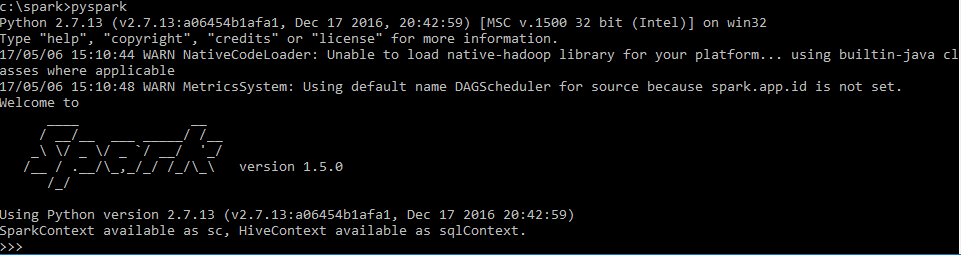


Si al introducir los 2 comandos y devuelve un número (no tiene por qué ser el 98, ya que cuenta el número de líneas que tiene el archivo) sin recibir ningún mensaje de error, podemos concluir con que Spark funciona correctamente.

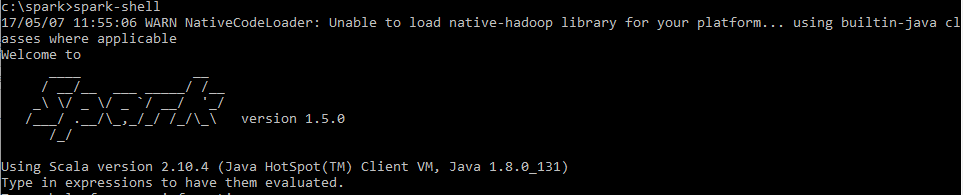
## 5.5 Manual de usuario

En primer lugar y, ante todo, como hemos mencionado en el documento de instalación, una vez instalado Spark, debemos abrir el símbolo de sistema.

Para comenzar a utilizar Spark utilizando el lenguaje de programación Python, debemos escribir el comando ***pyspark***, como se muestra a continuación:



Por otra parte, si deseamos utilizar Spark con el lenguaje de programación Scala, debemos introducir el comando ***spark-shell***:



En la presente práctica vamos a utilizar fundamentalmente el lenguaje de programación Scala para la creación del prototipo.

Pasamos a mostrar unos comandos clave para la utilización de Spark:

A continuación, hay una lista de transformaciones RDD.

|  |  |
| --- | --- |
| Nº | Transformaciones y Significado |
| 1 | **map(func)**  Devuelve un nuevo conjunto de datos distribuida, formado haciendo pasar cada elemento de la fuente a través de una función **func .** |
| 2 | **filter(func)**  Devuelve un nuevo conjunto de datos formado por la selección de aquellos elementos de la fuente en la que **func devuelve verdadero.** |
| 3 | **flatMap(func)**  Al igual que en el mapa, pero cada elemento de entrada se puede asignar a 0 o más elementos de salida (so *func* should return a Seq rather than a single item) . |
| 4 | **mapPartitions(func)**  Similar al mapa, pero se ejecuta por separado en cada partición (block) de la RDD, por lo **func debe ser de tipo Iterator <T> ⇒ Iterator <U> cuando se ejecuta en un RDD de tipo T.** |
| 5 | **mapPartitionsWithIndex(func)**  Similar al mapa de particiones, sino que también proporciona **func con un valor entero que representa el índice de la partición, de modo func debe ser de tipo (Int, Iterator<T>) ⇒ Iterator <U> cuando se ejecuta en un RDD de tipo T.** |
| 6 | **sample(withReplacement, fraction, seed)**  Pruebe una **fraction de los datos, con o sin sustitución, utilizando una semilla generador de números aleatorios dado.** |
| 7 | **union(otherDataset)**  Devuelve un nuevo conjunto de datos que contiene la unión de los elementos en el conjunto de datos de origen y el argumento. |
| 8 | **intersection(otherDataset)**  Devuelve un nuevo RDD que contiene la intersección de elementos en el conjunto de datos de origen y el argumento. |
| 9 | **distinct([numTasks])**  Devuelve un nuevo conjunto de datos que contiene los distintos elementos del conjunto de datos fuente. |
| 10 | **groupByKey([numTasks])**  Cuando se llama en un conjunto de datos de (K, V) pares, devuelve un conjunto de datos de (K, Iterable<V>) pares.  **Note - Si está agrupando a fin de realizar una agregación (such as a sum or average) sobre cada tecla, usando reduceByKey o aggregateByKey producirá un rendimiento mucho mejor.** |
| 11 | **reduceByKey(func, [numTasks])**  Cuando se llama en un conjunto de datos de (K, V) pares, devuelve un conjunto de datos de (K, V) pares donde los valores para cada clave se agregan utilizando el dado reducir la función *func* , que debe ser de tipo (V, V) ⇒ V . al igual que en groupByKey, el número de reducir las tareas se puede configurar a través de un segundo argumento opcional. |
| 12 | **aggregateByKey(zeroValue)(seqOp, combOp, [numTasks])**  Cuando se llama en un conjunto de datos de (K, V) pares, devuelve un conjunto de datos de (K, U) pares donde los valores para cada clave se agregan utilizando las funciones dadas combinar y un neutral "zero" valor. Permite un tipo de valor agregado que es diferente del tipo de valor de entrada, evitando al mismo tiempo las asignaciones innecesarias. Al igual que en groupByKey, el número de reducir las tareas se puede configurar a través de un segundo argumento opcional. |
| 13 | **sortByKey([ascending], [numTasks])**  Cuando se llama en un conjunto de datos de (K, V) pares donde implementos K pidió, devuelve un conjunto de datos de (K, V) pares ordenados por teclas en orden ascendente o descendente, como se especifica en el argumento ascendente booleana. |
| 14 | **join(otherDataset, [numTasks])**  Cuando se llama en conjuntos de datos de tipo (K, V) y (K, W) , devuelve un conjunto de datos de (K, (V, W) pares con todos los pares de elementos para cada tecla). Las combinaciones externas son apoyadas a través leftOuterJoin, rightOuterJoin y fullOuterJoin. |
| 15 | **cogroup(otherDataset, [numTasks])**  Cuando se llama en conjuntos de datos de tipo (K, V) y (K, W), devuelve un conjunto de datos de (K, (Iterable<V>, Iterable<W>) ) tuplas. Esta operación también se denomina grupo de Con. |
| 16 | **cartesian(otherDataset)**  Cuando se llama en conjuntos de datos de tipos T y U, devuelve un conjunto de datos de (T, U) pares (all pairs of elements) . |
| 17 | **pipe(command, [envVars])**  Pipe cada partición de la RDD a través de un comando shell, por ejemplo, un Perl o escritura del golpe. elementos RDD se escriben en la entrada estándar y las líneas de salida del proceso a su salida estándar se devuelven como un RDD de cadenas. |
| 18 | **coalesce(numPartitions)**  Disminuir el número de particiones en el RDD a numPartitions. Útil para operaciones en funcionamiento más eficiente después de la filtración por un gran conjunto de datos. |
| 19 | **repartition(numPartitions)**  Reordenar los datos de la RDD al azar para crear ya sea más o menos particiones y el equilibrio que a través de ellos. Esto siempre baraja todos los datos por la red. |
| 20 | **repartitionAndSortWithinPartitions(partitioner)**  Reparticionar el RDD de acuerdo con el particionador dado y, dentro de cada partición resultante, ordenar los registros de sus claves. Esto es más eficiente que llamar reparto y luego la clasificación dentro de cada partición, ya que puede empujar la clasificación hacia abajo en la maquinaria de reproducción aleatoria. |

La siguiente tabla muestra una lista de acciones, que devuelven valores.

|  |  |
| --- | --- |
| Nº | Acción y Significado |
| 1 | **reduce(func)**  Agregar los elementos del conjunto de datos utilizando una función **func (which takes two arguments and returns one).** La función debe ser conmutativa y asociativa de modo que pueda ser calculado correctamente en paralelo. |
| 2 | **collect()**  Devuelve todos los elementos del conjunto de datos como un conjunto en el programa del conductor. Esto suele ser útil después de un filtro u otra operación que devuelve un suficientemente pequeño subconjunto de los datos. |
| 3 | **count()**  Devuelve el número de elementos en el conjunto de datos. |
| 4 | **first()**  Devuelve el primer elemento del conjunto de datos (similar to take (1) ). |
| 5 | **take(n)**  Devuelve una matriz con los primeros **n elementos del conjunto de datos.** |
| 6 | **takeSample (withReplacement,num, [seed])**  Devuelve una matriz con una muestra aleatoria de **num elementos del conjunto de datos, con o sin sustitución, opcionalmente pre-especificando una semilla generadora de números aleatorios.** |
| 7 | **takeOrdered(n, [ordering])**  Devuelve los primeros **n elementos de la RDD utilizando ya sea su orden natural o un comparador personalizado.** |
| 8 | **saveAsTextFile(path)**  Escribe los elementos del conjunto de datos como un archivo de texto (or set of text files) en un directorio dado en el sistema de archivos local, HDFS o cualquier otro sistema de archivos compatible Hadoop. Chispa llama a toString en cada elemento para convertirlo en una línea de texto en el archivo. |
| 9 | **saveAsSequenceFile(path) (Java y Scala)**  Escribe los elementos del conjunto de datos como una Hadoop SequenceFile en una trayectoria dada en el sistema de ficheros local, HDFS o cualquier otro sistema de archivos compatible-Hadoop. Esto está disponible en DDR de pares de valores clave que implementan la interfaz grabable de Hadoop. En Scala, que también está disponible en los tipos que son convertir implícitamente a escribible (Spark includes conversions for basic types like Int, Double, String, etc) . |
| 10 | **saveAsObjectFile(path) (Java y Scala)**  Escribe los elementos del conjunto de datos en un formato simple usando serialización Java, que luego puede ser cargado usando SparkContext.objectFile(). |
| 11 | **countByKey()**  Sólo disponible en DDR de tipo (K, V). Devuelve un mapa hash de (K, Int) se empareja con el recuento de cada tecla. |
| 12 | **foreach(func)**  Ejecuta una función **func a cada elemento del conjunto de datos.** Esto es por lo general, hecho por efectos secundarios tales como la actualización de un acumulador o interactuar con los sistemas de almacenamiento externos.  **Note - la modificación de variables distintas de Acumuladores fuera del foreach() puede resultar en un comportamiento indefinido.** Ver cierres de entendimiento para más detalles. |

# 6. Comparación de las dos implementaciones

Se trata de dar valores a los criterios de comparación definidos en el apartado 3 sobre la implementación de cada uno de los prototipos.

## 6.1 Evaluación de los criterios en la implementación usando la tecnología A

Debe incluir al menos una tabla con la siguiente estructura.

| **CRITERIO** | **EVALUACIÓN** |
| --- | --- |
| Criterio 1 |  |
| Criterio 2 |  |
| … |  |
| Criterio N |  |

Y algunos comentarios aclaratorios sobre aquellos criterios cuyo valor indicado en la tabla no sea suficiente para entenderlo.

## 6.2 Evaluación de los criterios en la implementación usando la tecnología B

# 7. Comparación de la implementación de las tecnologías

Debe incluir al menos una tabla resumen, en sección de página horizontal, cruzando los criterios y los valores de cada tecnología. Con una columna de comentarios sobre la comparación

| **CRITERIOS** | **TECNOLOGÍA A** | **TECNOLOGÍA B** | **COMENTARIOS** |
| --- | --- | --- | --- |
| 1 |  |  |  |
| 2 |  |  |  |
| … |  |  |  |
| N |  |  |  |

# 8. Conclusiones

A partir de la información incluida en el apartado 7 y de la experiencia al realizar el trabajo, el grupo debe estar en condiciones de manifestar su opinión sobre la implementación del sistema utilizando ambas tecnologías, y debe plasmarla en este apartado, indicando las ventajas e inconvenientes más relevantes de utilizar una u otra tecnología para implementar el sistema.

---------------------------

(Hay que cumplir la estructura básica indicada de secciones. Pero si se desea se pueden añadir otras secciones como anexos. Por ejemplo, alguna encuesta de opinión realizada sobre las tecnologías, etc.)