

# C.F.P Juan XXIII



Desarrollo de Aplicaciones Multiplataforma

## TRABAJO DE FIN DE GRADO

BIG DATA: IMPLEMENTACIÓN DE SISTEMA Y  
APLICACIÓN EN ANÁLISIS DE DATOS

David Martínez Martín  
Tutor: Aníbal Martín Serrano

Marzo de 2018



# **BIG DATA: IMPLEMENTACIÓN DE SISTEMA Y APLICACIÓN EN ANÁLISIS DE DATOS**

Autor: David Martínez Martín

Tutor: Aníbal Martín Serrano

C.F.P Juan XXIII

Marzo de 2018

# Resumen

**Resumen** — *Big data* ha sido uno de los términos tecnológicos que más importancia ha ganado durante los últimos años. Debido al proceso de digitalización de la sociedad y de la economía, que se ha producido y se está produciendo, gran cantidad de datos se generan cada día y son muchas las empresas y entidades que los almacenan para intentar sacar provecho de ellos. Estos datos surgen desde la propia actividad de las empresas hasta de mediciones en la naturaleza, pasando por los recogidos en smartphones, sensores, wearables o dispositivos conectados, generándose millones de terabytes de datos al día.

De la necesidad de organizar y procesar todos ellos, para obtener información valiosa que pueda generar valor surge el término *big data*, que se refiere a aquellos datos que son tan grandes o complejos que las aplicaciones de procesado de datos tradicionales no serían capaces de tratarlos. Por ello, las tecnologías y herramientas *big data* permiten el tratado de este tipo de datos de una manera más rápida y eficiente, ampliando las posibilidades de trabajo con dicha información.

El gran auge de este sector durante los últimos años ha permitido el desarrollo de importantes mejoras que van desde herramientas para el análisis en tiempo real hasta la creación de nuevas arquitecturas para optimizar el flujo y almacenamiento de los datos. Además, este desarrollo ha permitido el abaratamiento de la tecnología y la creación de empresas que ofrecen a otras servicios de *big data*, provocando el interés de un gran número de empresas de los diferentes sectores de la economía, que ven una oportunidad para sus negocios en el análisis de datos.

En este proyecto nos enfocaremos a realizar un análisis de diferentes fuentes de datos para que una empresa dedicada a la impartición de cursos dirigidos a programadores pueda decidir cuáles lanzar el próximo año, dónde lo harán, y en qué idioma.

**Palabras clave** — Datos, Big Data, Apache Hadoop, Apache Spark, Insights.

# Abstract

**Abstract** — *Big data* has been one of the technological terms which more relevance has gained during the recent years. This has been caused because of the digital transformation process that is taking place nowadays and which is the main reason of the great quantity of data that is being generated every day. Some organizations and business, increasingly each day, are storing and trying to get some benefit of them. The source of this data very diverse, it can come from the activity generated by the business, from the devices used by people, like smartphones, wearables and connected devices, to measures taken in the nature. Millions of terabytes of new data is generated every day.

Because the necessity of organizing and processing this data to obtain valuable information, the term *big data*, which refers to the kind of data which is so big or so complex to the traditional tools to threat it, comes up. So that, *big data* technologies and tools allow this kind of processing in a much efficient and faster way, increasing the working possibilities with it.

The growth of this sector during the last years has allowed the development of a wide range of improvements from tools that allow real time analysis to new file and system architectures to save space and increase the efficiency.

In this project, we are going to focus to analyze differents data sources for a company dedicated to the impartition of training courses for programmers must decide which courses they will launch next year, where they will do it, and in which language.

**Key words** — Data, Big Data, Apache Hadoop, Apace Spark, Insights.



# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Objetivos . . . . .	2
1.2. Estructura del documento . . . . .	3
<b>2. Estado del Arte</b>	<b>4</b>
2.1. Open data . . . . .	4
2.2. Big Data . . . . .	7
2.2.1. Introducción . . . . .	7
2.2.2. Definición . . . . .	7
2.2.3. Propósitos . . . . .	9
2.2.4. Visión de negocio y futuro . . . . .	10
2.3. Sistemas distribuidos . . . . .	12
2.3.1. Configuraciones . . . . .	13
2.3.2. Arquitectura utilizada . . . . .	15
2.3.3. Nuevos roles de trabajo . . . . .	16
2.4. Infraestructura . . . . .	18
2.4.1. Apache Spark . . . . .	18
2.4.2. Apache Hadoop . . . . .	23
2.4.3. Apache Flume . . . . .	25
2.4.4. Jupyter Notebook . . . . .	27
<b>3. Marco Regulador</b>	<b>29</b>
3.1. Legislación aplicable . . . . .	29
3.2. Estándares técnicos . . . . .	31
<b>4. Diseño</b>	<b>32</b>
4.1. Introducción . . . . .	32
4.2. Arquitectura del sistema . . . . .	32
4.2.1. Modo distribuido o multinodo . . . . .	32
4.3. Insights del proyecto . . . . .	36
4.3.1. Plan de proyecto . . . . .	36
<b>5. Instalación del clúster</b>	<b>37</b>
5.1. Introducción . . . . .	37
5.2. Preparación del entorno de trabajo . . . . .	37

5.2.1. Especificaciones del equipo . . . . .	38
5.2.2. Instalación del sistema operativo . . . . .	39
5.2.3. Instalación de Apache Spark . . . . .	39
5.2.4. Configuración de <i>Jupyter Notebook</i> . . . . .	45
5.2.5. Instalación de <i>Apache Hadoop</i> . . . . .	46
5.2.6. Instalación de <i>Apache Flume</i> . . . . .	53
5.3. Arranque y parada del clúster . . . . .	53
<b>6. Stack Overflow</b>	<b>55</b>
6.1. Introducción . . . . .	55
6.2. Aprovisionamiento de datos . . . . .	55
6.3. Procesado de datos . . . . .	56
6.4. Análisis de datos . . . . .	59
<b>7. Twitter</b>	<b>63</b>
7.1. Introducción . . . . .	63
7.2. Aprovisionamiento de datos . . . . .	64
7.2.1. Agente de <i>Apache Flume</i> . . . . .	64
7.3. Análisis de datos . . . . .	65
7.4. Procesado de datos . . . . .	67
7.4.1. Script Apache Spark . . . . .	67
7.4.2. Script Bash para normalizar resultados . . . . .	67
<b>8. GitHub</b>	<b>69</b>
8.1. Introducción . . . . .	69
8.2. Análisis de datos . . . . .	70
8.3. Aprovisionamiento de datos . . . . .	70
8.3.1. Aprovisionamiento de datos pre-análisis . . . . .	71
8.3.2. Aprovisionamiento de datos post-análisis . . . . .	71
8.4. Procesado de datos . . . . .	77
8.5. Graficado de datos . . . . .	81
<b>9. Conclusiones</b>	<b>82</b>
9.1. Introducción . . . . .	82
9.2. Conclusiones . . . . .	83
9.3. Informe final . . . . .	84
9.3.1. Lenguajes populares . . . . .	84
9.3.2. Regionalización de la empresa . . . . .	86
9.3.3. Idioma vehicular . . . . .	87
<b>Bibliografía</b>	<b>88</b>
<b>Apéndices</b>	<b>92</b>
<b>A. Estructura completa de un <i>tweet</i></b>	<b>93</b>

## ÍNDICE GENERAL

B. Estructura completa de un evento de GitHub	112
C. Fichero de configuración “.bashrc” completo	124

# Índice de tablas

5.1.	Especificaciones del maestro . . . . .	38
5.2.	Especificaciones del nodo1 . . . . .	38
6.1.	<i>DataFrame</i> de muestra de datos de <i>Stack Overflow</i> . . . . .	61
6.2.	Tabla con los 10 lenguajes más populares de <i>Stack Overflow</i> . . . . .	62
8.1.	Tabla de instrucciones de descarga de <i>Github Archive</i> [62]. . . . .	71
8.2.	Resultados extraídos de los errores en el proceso de aprovisionamiento. . .	78
8.3.	Tabla de resultados cruzados de GitHub. . . . .	80

# Índice de figuras

2.1.	Número de API de datos disponibles con respecto al tiempo y compromisos de abrir los datos de diferentes países del mundo [9] . . . . .	5
2.2.	Valor potencial de los datos abiertos en billones de dólares en Estados Unidos	6
2.3.	Evolución de las V's que definen el <i>big data</i> a lo largo del tiempo [18] . . . . .	8
2.4.	Panorama de las tecnologías <i>big data</i> en 2017 [20] . . . . .	10
2.5.	Proyección de los ingresos del <i>big data</i> [28] . . . . .	11
2.6.	Esquema de computación distribuida [30] . . . . .	12
2.7.	Esquema cliente-servidor [29] . . . . .	13
2.8.	Esquema peer-to-peer [32] . . . . .	14
2.9.	Esquema de un clúster [34] . . . . .	15
2.10.	Componentes de <i>Apache Spark</i> [40] . . . . .	19
2.11.	Funcionamiento de <i>Apache Spark</i> [40] . . . . .	20
2.12.	Transformaciones sobre un Resilient Distributed Dataset (RDD) representadas en un Direct Acyclic Graph (DAG) [40] . . . . .	21
2.13.	RDD distribuido en diferentes máquinas [43] . . . . .	21
2.14.	Arquitectura del Hadoop Distributed File System (HDFS) [4] . . . . .	24
2.15.	Arquitectura del HDFS [47] . . . . .	25
2.16.	Previsualización del entorno de Jupyter Notebook [49]. . . . .	27
4.1.	Modo distribuido o multinodo de <i>Spark</i> [34] . . . . .	33
4.2.	Arquitectura del clúster doméstico . . . . .	34
4.3.	Esquema del sistema de replicación HDFS . . . . .	35
5.1.	Estructura de carpetas para operativa del <i>namenode</i> . . . . .	47
5.2.	Estructura de carpetas para operativa del <i>datanode</i> . . . . .	47
6.1.	Esquema completo de <i>Stack Overflow</i> . . . . .	59
9.1.	Esquema de orígenes de datos del proyecto . . . . .	83
9.2.	Histórico Stack Overflow . . . . .	84
9.3.	Histórico de GitHub . . . . .	85
9.4.	Mapamundi de incidencia de programadores por país . . . . .	86
9.5.	Gráfica orientativa sobre el uso de idiomas . . . . .	87

# Índice de códigos

5.1.	Instalación de Java y Scala. . . . .	40
5.2.	Descarga e instalación de Anaconda. . . . .	40
5.3.	Creación de grupo común y adición del usuario del sistema a este. . . . .	41
5.4.	Instalación del cliente y el servidor de Secure Shell (SSH). . . . .	41
5.5.	Creación y autorización del certificado SSH. . . . .	41
5.6.	Líneas a añadir en el fichero “/etc/hosts” de cada nodo del clúster. . . . .	42
5.7.	Obtención de los certificados SSH de los esclavos por parte del maestro en el clúster. . . . .	42
5.8.	Líneas modificadas en el fichero “/etc/ssh/sshd_config” cada nodo del clúster. .	43
5.9.	Comando de consola para reiniciar el servidor SSH. . . . .	43
5.10.	Script de instalación de <i>Apache Spark</i> . . . . .	43
5.11.	Líneas a añadir en el fichero “spark-env.sh” para configurar <i>Apache Spark</i> en el modo distribuido. . . . .	44
5.12.	Contenido del fichero “spark-defaults.conf” de configuración de <i>Apache Spark</i> . .	44
5.13.	Líneas a añadir en el fichero “slaves” para establecer las máquinas a utilizar en el clúster. . . . .	44
5.14.	Comando para generar el kernel <i>Jupyter</i> para PySpark. . . . .	45
5.15.	Contenido del fichero “kernel.json” para la configuración con <i>Apache Spark</i> . .	45
5.16.	Script de instalación de <i>Apache Hadoop</i> . . . . .	46
5.17.	Código de creación de la estructura de carpetas del <i>namenode</i> . . . . .	46
5.18.	Código de creación de la estructura de carpetas del <i>datanode</i> . . . . .	47
5.19.	Líneas a añadir a “.bashrc” para el funcionamiento de <i>Apache Hadoop</i> . . . .	48
5.20.	Línea a añadir a “hadoop-env.sh”. . . . .	48
5.21.	Contenido del “core-site.xml”. . . . .	48
5.22.	Contenido del “hdfs-site.xml” de el nodo maestro. . . . .	49
5.23.	Contenido del “hdfs-site.xml” de los nodos esclavos. . . . .	50
5.24.	Contenido del “mapred-site.xml”. . . . .	50
5.25.	Contenido del “yarn-site.xml”. . . . .	51
5.26.	Línea a añadir a “master”. . . . .	52
5.27.	Líneas a añadir a “slaves”. . . . .	52
5.28.	Script de instalación de <i>Apache Flume</i> . . . . .	53
5.29.	Línea a añadir a “flume-env.sh”. . . . .	53
5.30.	Script de inicio del clúster implementado. . . . .	53
5.31.	Script de parada del clúster implementado. . . . .	54
5.32.	Comandos requeridos en la primera ejecución del clúster. . . . .	54

6.1.	Ficheros públicos sobre <i>Stack Overflow</i> en la web de <i>Data Dump</i> . . . . .	56
6.2.	<i>Imports</i> del script “ <i>xml2json.py</i> ” de procesado de Extensible Markup Language (XML). . . . .	56
6.3.	Fragmento de código del script “ <i>xml2json.py</i> ” de comprobación del estado de los ficheros <i>in/out</i> . . . . .	57
6.4.	Fragmento de código del script “ <i>xml2json.py</i> ” encargado de la persistencia de resultados. . . . .	57
6.5.	Fragmento de código del script “ <i>xml2json.py</i> ” encargado de la persistencia de resultados. . . . .	58
6.6.	Consulta Structured Query Language (SQL) para la extracción de la fecha, <i>tag</i> y <i>score</i> de las consultas en <i>Stack Overflow</i> . . . . .	60
6.7.	<i>Imports</i> y configuración de una sesión de <i>Apache Spark</i> . . . . .	61
6.8.	Código de declaración del <i>schema</i> para lectura del Comma Separated Values (CSV). . . . .	61
6.9.	Carga de un fichero CSV con un esquema predefinido con el módulo <i>Spark SQL</i> . . . . .	61
6.10.	Código de generación de la tabla temporal y la ejecución de una consulta SQL. . . . .	62
7.1.	Fichero de configuración del agente de <i>Apache Flume</i> para <i>Twitter</i> . . . . .	64
7.2.	Comando de lanzamiento del agente de <i>Apache Flume</i> para <i>Twitter</i> . . . . .	65
7.3.	Extracto del esquema de un <i>tweet</i> . . . . .	65
7.4.	Consulta SQL para la extracción de datos de Twitter. . . . .	67
7.5.	Comandos utilizados durante el proceso de unificación de datos. . . . .	67
7.6.	Código bash de evaluación. . . . .	68
8.1.	Comando para el preaprovisionamiento de datos de GitHub. . . . .	71
8.2.	Script para la extracción automática de los ficheros de GitHub. . . . .	71
8.3.	Clase <i>Main</i> de la aplicación de aprovisionamiento de datos de GitHub. . . . .	72
8.4.	Fragmento de la clase <i>Process</i> . Declaración de constantes y apertura de ficheros. . . . .	73
8.5.	Fragmento de la clase <i>Process</i> . Descarga, extracción y guardado. . . . .	74
8.6.	Fragmento de la clase <i>Extractor</i> . Declaración de constantes. . . . .	75
8.7.	Fragmento de la clase <i>Extractor</i> . Método principal. . . . .	75
8.8.	Fragmento de la clase <i>Extractor</i> . Comprobación de tipo. . . . .	76
8.9.	Fragmento de la clase <i>Extractor</i> . Método de extracción de lenguajes. . . . .	76
8.10.	Fragmento de la clase <i>Extractor</i> . Método de extacción de <i>commits</i> . . . . .	76
8.11.	Creación del <i>DataFrame</i> y su tabla temporal . . . . .	77
8.12.	Creación de un <i>DataFrame</i> diferencial de lenguajes y unión de ambas. . . . .	77
8.13.	SQL para cruzar datos de lenguajes y <i>commits</i> . . . . .	79
8.14.	Transformaciones del RDD para asegurar la calidad del dato. . . . .	79
A.1.	Esquema completo de la estructura de un <i>tweet</i> . . . . .	93
B.1.	Esquema completo de la estructura de GitHub. . . . .	112
C.1.	Fichero “ <i>.bashrc</i> ” completo del sistema. . . . .	124

# 1

## Introducción

A consecuencia del proceso de transformación digital que lleva produciéndose desde hace unos años de forma general en la sociedad y, especialmente, en el mundo empresarial, el tráfico y la generación de datos ha aumentado de forma muy abrupta, generándose millones de terabytes de nuevos datos al día [1].

La procedencia de estos datos es muy variada, sin embargo, la irrupción en la vida cotidiana de los dispositivos conectados a la red ha sido la principal razón del crecimiento exponencial que se ha producido en estos últimos años. Concretamente, son el crecimiento en el número de smartphones [2] y, más recientemente, la llegada del Internet de las cosas (IOT) y los wearable los principales responsables de este crecimiento en la cantidad de datos generados.

Este gran volumen de datos hace que la tarea del procesado de los mismos haya pasado a ser mucho más costosa. Además, el uso de diferentes fuentes de información hace que su estructura no sea homogénea, de forma que aumenta la complejidad del procesado. Esto es debido a la necesidad de su reestructuración para poder ser utilizados.

Los cambios en la manera de obtener los datos han conllevado a que nuevas herramientas hayan sido necesarias para suplir las carencias e incapacidades que las tradicionales tenían para procesar los datos. El desarrollo de estas nuevas tecnologías ha supuesto un desafío para las empresas y organizaciones, que buscan obtener ventajas competitivas mediante el estudio de toda la información.

Es decir, las organizaciones y empresas buscan, con el desarrollo de estos sistemas, obtener de los datos la información más valiosa posible para sus negocios y, así, lograr ventaja con respecto a sus competidores y poder trazar estrategias que aumenten los beneficios.

Por otro lado, hay organizaciones y entes públicos que están utilizando estas herramientas de procesamiento y análisis para mejorar el funcionamiento de diversos sistemas. Especial importancia está tomando procesado en tiempo real que permiten mostrar la información de forma instantánea y predecir tendencias en cortos espacios de tiempo, por ejemplo, en los sistemas de transporte, permitiendo la toma de medidas para optimizar su funcionamiento.

Este proyecto de fin de grado analizará diferentes fuentes de información de uso común para programadores de forma profesional y personal con el fin de obtener una conclusión que permita una toma de decisiones basada en los datos obtenidos, permitiendo una mejora de las decisiones así como una reducción en los costes y una mayor captación de clientes potenciales como resultado directo este proyecto.

En este documento se expondrán el diseño implementado para el sistema *big data*, que será desarrollado con las tecnologías *Apache Spark* [3], presente en todas las implementaciones realizadas, y con *Apache Hadoop* [4], cuyo sistema distribuido de ficheros será utilizado en conjunción con el primero en algún diseño.

## 1.1. Objetivos

El principal objetivo es realizar un informe donde se indiquen los lenguajes de programación más populares, el idioma vehicular de los programadores y los países donde estos tienen mayor impacto. Para alcanzar el objetivo, el proyecto contará con las siguientes fases de trabajo:

- Análisis y estudio del panorama y de las herramientas *big data* disponibles en la actualidad.
- Estudio de los insights a resolver y las fuentes de datos.
- Diseño del sistema *big data*.
- Implementación del sistema *big data* diseñado.
- Aprovisionamiento, análisis y procesado de datos.
- Obtención de las conclusiones.

## 1.2. Estructura del documento

En este apartado se presentará la estructura que seguirá el documento con el fin de ofrecer una visión general del mismo y facilitar su lectura y seguimiento.

1. **Introducción:** Donde se presenta el contexto en el que se realiza el proyecto realizado, así como la motivación del mismo, los objetivos y la estructura del documento. Apartado 1.
2. **Estado del arte:** Donde se analiza el contexto en el que se realiza el proyecto respecto a las tecnologías y herramientas utilizadas, en este caso el *big data* y el open data. Apartado 2.
3. **Marco regulador:** Detalla las leyes y estándares que se aplican en los proyectos *big data* en la realidad. Apartado 3.
4. **Diseño:** En este apartado se describe el proceso de diseño del sistema y los objetivos a cumplir. Apartado 4.
5. **Instalación:** Se describe el proceso de implementación de los sistemas diseñados, describiendo los pasos necesarios y realizados para hacer funcionar al sistema. Apartado 5.
6. **Stack Overflow:** Detalla los pasos realizados para el aprovisionamiento, análisis y procesado de datos de esta plataforma. Apartado 6.
7. **Twitter:** Detalla los pasos realizados para el aprovisionamiento, análisis y procesado de datos de esta plataforma. Apartado 7.
8. **GitHub:** Detalla los pasos realizados para el aprovisionamiento, análisis y procesado de datos de esta plataforma. Apartado 8.
9. **Conclusiones:** Se realiza una retrospectiva sobre el trabajo realizado, así como un informe final donde se da respuesta a los objetivos planteados. Apartado 9.
10. **Apéndices:** Información extra sobre el proyecto.

# 2

## Estado del Arte

En este apartado se analiza el contexto y la situación de las tecnologías involucradas en este proyecto. Por ello, en este capítulo trataremos la situación en la que se encuentran los datos abiertos, como los aquí utilizados, en cuanto a su acceso y beneficios.

Además también realizaremos un análisis sobre la situación actual del *big data*, las herramientas disponibles para crear sistemas de este tipo y, en especial, *Apache Spark*, *Apache Hadoop* y *Apache Parquet*, que han sido las utilizadas para la implementación de los sistemas diseñados.

### 2.1. Open data

En este proyecto, los datos que se van a utilizar están publicados en Internet, donde cualquiera que lo desee puede hacer y utilizarlos sin incurrir en ningún tipo de ley.

Este tipo de datos que está disponible de forma libre para todo el mundo, sin restricciones de ningún tipo, ya sean derechos de autor, patentes o cualquier otro mecanismo de control es conocidos como datos abiertos u open data, en inglés. Estos datos podrán ser utilizados y redistribuidos de forma libre por cualquier persona [5].

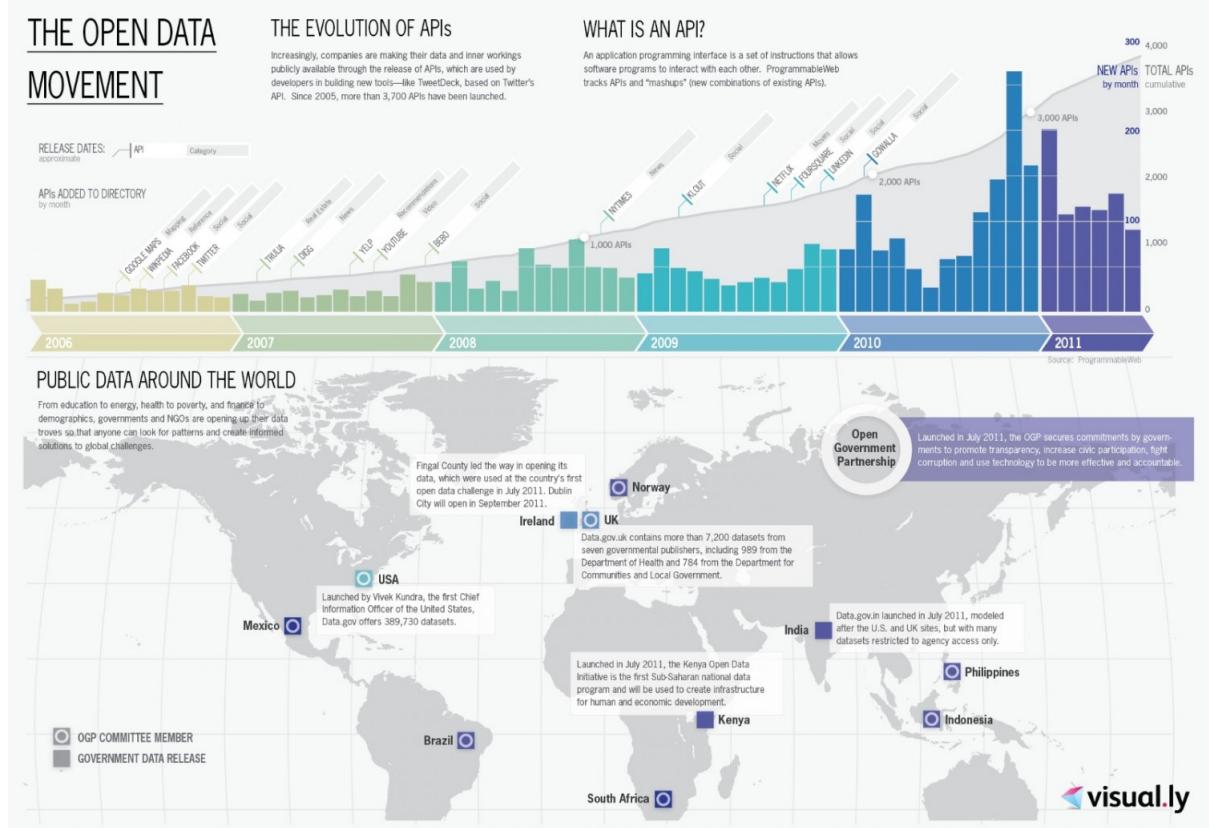
Aunque el término estaba definido desde el siglo pasado, en 1958 se crea la primera base de datos abiertos para la ciencia, el World Data Center (WDC) [6], ha sido con el proceso de digitalización de la sociedad y el reciente auge de las herramientas *big data* cuando más proliferación ha habido de estos movimientos.

Actualmente, un gran número de organismos gubernamentales y no gubernamentales tienen plataformas de acceso a datos abiertos sobre diferentes aspectos de la sociedad,

como datos demográficos o económicos. Así podemos encontrar datos globales, como en el portal de datos la Organización de las Naciones Unidas (ONU) [7], a datos locales, como en el portal de datos abiertos de la ciudad de Madrid [8].

Como se puede apreciar en la figura 2.1 el crecimiento del número de APIs para el acceso a datos ha aumentado de una forma importante desde 2006. También se puede observar como grandes países se han ido comprometiendo a liberar datos abiertos sobre diferentes asuntos de estado.

Figura 2.1: Número de API de datos disponibles con respecto al tiempo y compromisos de abrir los datos de diferentes países del mundo [9]

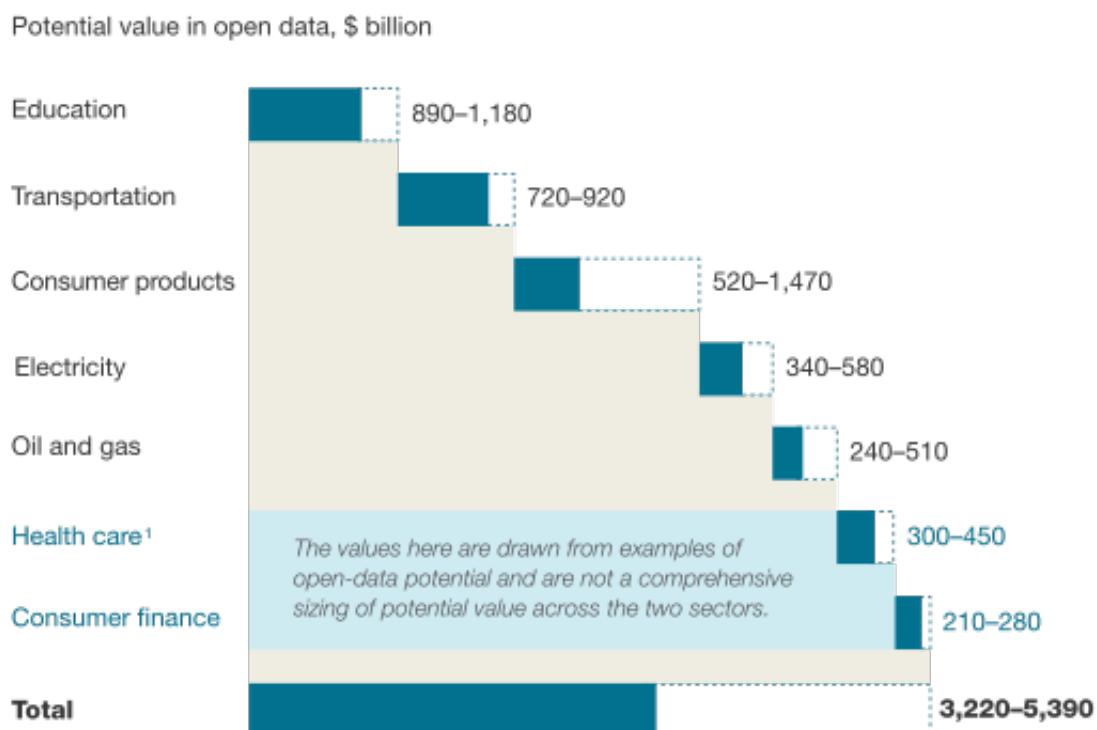


Se estima que la apertura de datos, solo en Estados Unidos, puede generar entre 3 y 5 billones de dólares [10]. El principal motivo es el gran potencial económico que puede suponer, ya que se podría aumentar la eficiencia en diferentes aspectos de la sociedad, como en la sanidad y educación, además, crearía nuevos servicios y productos, que irían acompañados de puestos de trabajo, y podrían mejorar la percepción para las personas sobre el mundo en general al aumentar la transparencia de la información.

Por otro lado en Europa, otros estudios prevén que el uso de datos abiertos reportará un ahorro de más de 1.7 billones de euros a las administraciones públicas, reducirá en un 16 % la energía consumida y el tiempo de espera en las carreteras. Aparte, indica que las compañías que siguen una filosofía de datos abiertos aumentan su valor en mayor medida que otras que no los abren [11].

Por tanto, se espera que en el futuro cercano más y más países y organizaciones se sumen a estas iniciativas de apertura de datos y, que, los países que ya están participando en ellas, mejoren la calidad de la información que suministran, cuyos formatos, en ocasiones, no pueden ser utilizados sin una transformación a otros adecuados para la lectura por las máquinas.

Figura 2.2: Valor potencial de los datos abiertos en billones de dólares en Estados Unidos



<sup>1</sup>Includes US values only.

Source: McKinsey Global Institute analysis

## 2.2. Big Data

### 2.2.1. Introducción

El término *big data* es relativamente reciente, su primera aparición parece datar de 1989 cuando Erik Larson lo utilizó en una artículo para la revista Harper's, aunque, se creé que se definió como tal en los años noventa durante las charlas que daba John Mashey, científico jefe de Silicon Graphics, donde explicaba el término para vender sus productos, como se puede ver en presentaciones suyas de 1998 [12].

Sin embargo, los pilares en los que se basa el *big data* no son nada nuevos. Los seres humanos nos hemos caracterizado por la idea de almacenar toda la información posible para crear una base de datos en continua expansión que pueda ser analizada y de donde se extraiga conocimiento valioso para la humanidad.

Ya en el año 18000 Antes de Cristo (A.C), había tribus paleolíticas que mediante muescas en palos y huesos hacían seguimientos de las reservas de suministros y de las actividades comerciales, permitiendo hacer predicciones sobre la duración de los alimentos almacenados. También, sobre el año 2400 A.C surge el ábaco en Babilonia que permitía realizar cálculos.

Como ejemplo de primer almacén de datos, tenemos a la Biblioteca Real de Alejandría, que fue la colección de conocimiento del antiguo mundo. Como primer ejemplo que definió el término "Business intelligence", contamos con el banquero Henry Furnese que utilizó en 1985 los datos recolectados de su negocio para obtener una ventaja competitiva frente a sus adversarios.

Pero es durante la segunda mitad del siglo XX y el XXI donde se produce los avances que han permitido llegar a la realidad actual. Hitos como el desarrollo de la computación, los sistemas de almacenamiento digital e Internet y las conexiones inalámbricas han sido las que han permitido crear y almacenar grandes cantidades de datos, unas cantidades que crecen exponencialmente con el tiempo.

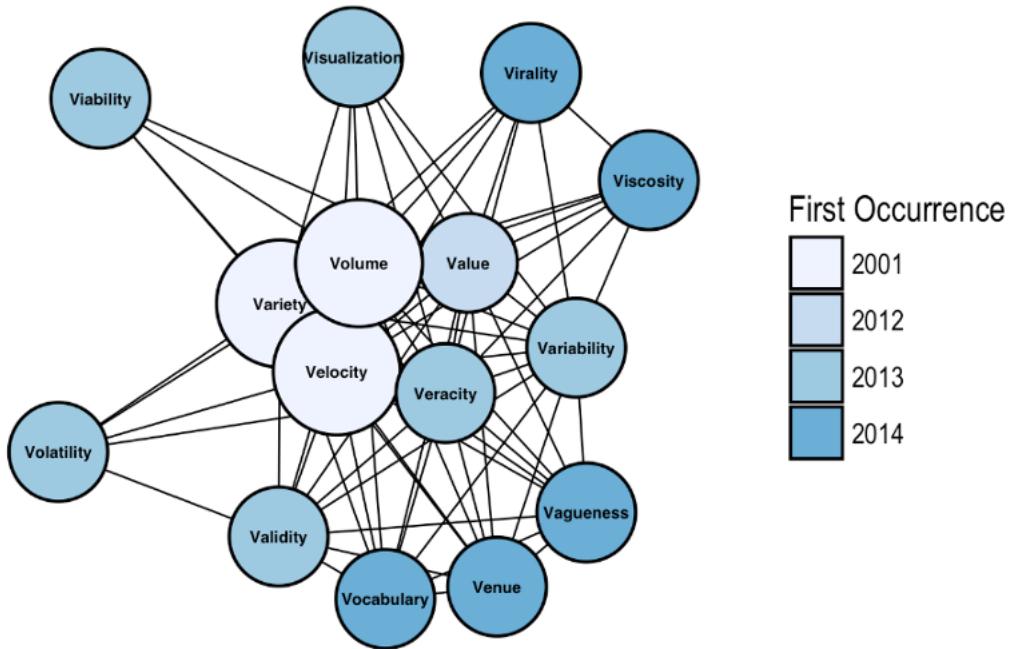
### 2.2.2. Definición

Si buscamos la definición de *big data* en Internet podemos encontrar diversas versiones de la misma [13], algunas más correctas y otras menos, pero no existe un consenso claro sobre el término. Sin embargo, la agencia de la ONU encargada de las tecnologías de la información, creadora del primer estándar *big data* lo define como un paradigma que permite la recolección, almacenamiento, manejo, análisis y visualización de extensos conjuntos de datos, con características heterogéneas y con restricciones de tiempo cercanas al tiempo real [14].

Otro aspecto que ejemplifica estos cambios en la definición del concepto se puede apreciar en la conocida forma de describir el *big data*, las V's. Estas comenzaron siendo tres [13], para pasar a ser 4 [15] y, luego, 5 [16], 7 [17] e, incluso, 10. En la figura 2.3 se

puede apreciar como ha sido este proceso a lo largo del tiempo.

Figura 2.3: Evolución de las V's que definen el *big data* a lo largo del tiempo [18]



Aunque, igual que en el caso de la definición, el número de V's variará dependiendo del experto al que consultes, podemos establecer cuatro aspectos básicos de un sistema para que este sea considerado *big data*. Estos son:

- **Volumen:** Referido como al tamaño y la cantidad de los datos, aunque no existe un límite establecido ya que estas cantidades cada vez aumentan a un mayor ritmo.
- **Velocidad:** Referido a la rapidez con la que se accede a los datos. Lo que se busca en un análisis en tiempo real.
- **Veracidad:** Referido a la certeza de que los datos contenidos en el sistema son reales. Un sistema *big data* debe procurar mantener a cero la información no veraz a razón de no perder eficiencia.
- **Variedad:** Referido al número de fuentes diferentes de donde provienen los datos. Los sistemas *big data* tienen que poder procesar datos de diversos orígenes y estructuras.

### 2.2.3. Propósitos

Muchas grandes compañías del sector tecnológico llevan años usando herramientas *big data* para mejorar su negocio, compañías pioneras en este campo como Google y Yahoo que desarrollaron primeras versiones de esta tecnología buscando mejorar el resultado de sus buscadores.

Como ya se ha comentado anteriormente, las posibles aplicaciones de estas herramientas son infinitas. En general, disponer de un conjunto de datos que contengan información sobre situaciones pasadas o sobre sucesos y elementos que puedan influenciar un proceso es beneficioso para saber como actuar de la forma más correcta posible.

En general, se pueden encontrar cuatro tipos de usar herramientas *big data* [19]. Por un lado, podemos encontrar el análisis prescriptivo, que es el más valioso y menos usado, que consiste la utilización de los datos para encontrar posibles caminos de actuación que permitan anticiparse al futuro y obtener ventaja. Un caso donde se está empezando a utilizar este tipo de análisis es en los sistemas de salud, donde conociendo los datos de un grupo, se pueden encontrar patrones para atajar enfermedades de la forma más efectiva.

Otro tipo de análisis es el predictivo, donde se busca detectar patrones que hayan ocurrido en el pasado para predecir el futuro. Este, está siendo especialmente utilizado en los procesos de venta, por ejemplo, para ajustar los precios en base a la oferta y la demanda, así como para preparar los “stocks” de los productos.

El análisis de diagnóstico es aquel que busca determinar porque se ha producido algún suceso, usando para ello los datos relacionados con el evento. Este es especialmente utilizado en marketing, para entender el efecto que han tenido las campañas, pero, también en sistemas de software, para conocer el motivo del error y poder evitar situaciones similares en el futuro.

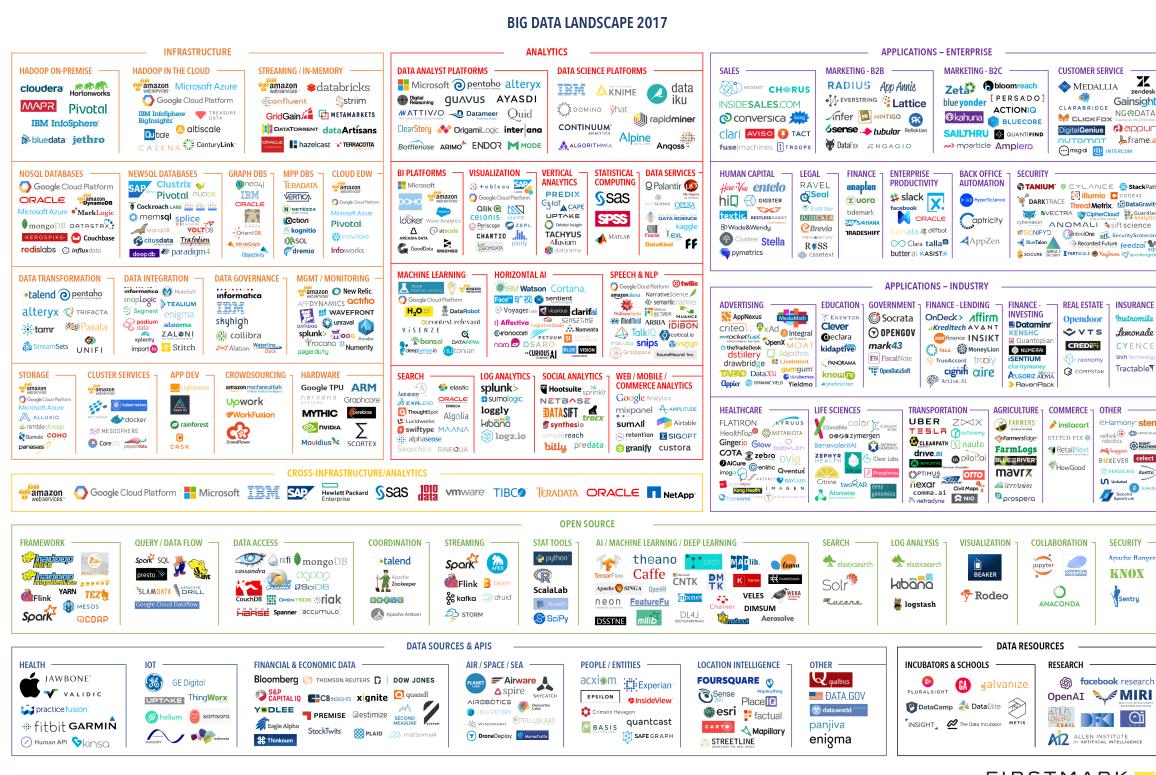
Finalmente, el análisis descriptivo o minería de datos, que busca información valiosa entre todos los datos almacenados. Un ejemplo de este tipo de análisis se da en el estudio de los riesgos al ofrecer un crédito, donde se puede buscar en el pasado financiero del cliente datos que puedan describir la capacidad económica del mismo.

#### **2.2.4. Visión de negocio y futuro**

La amplias posibilidades que las herramientas *big data* ofrecen junto con el conocido éxito de empresas y su, cada vez, más sencilla implementación está haciendo que cada vez más y más negocios implementen estas tecnologías.

El “landscape” también ha ido creciendo durante estos últimos años, debido al auge de estas tecnologías, haciendo que las herramientas y arquitecturas disponibles haya aumentado, disponiéndose de un amplio abanico de opciones que se adaptan a todo tipos de datos y análisis. En la figura 2.4 se puede observar el panorama actual.

Figura 2.4: Panorama de las tecnologías *big data* en 2017 [20]



Una de las características de este paradigma tecnológico es la naturaleza abierta de las herramientas, donde las más usadas son de código abierto y han surgido del resultado de la colaboración de diferentes empresas u organizaciones. Librerías como *Apache Hadoop* [4] o *Apache Spark* [3], bases de datos como *Apache Cassandra* [21] o *MongoDB* [22] y herramientas de análisis como *SciPy* [23] o *Pandas* [24] son de uso libre y no requieren de pago de licencias.

Esto ha hecho surgir empresas que se encargan de establecer y mantener un sistema *big data*, es decir, ha permitido a los negocios externalizar este proceso, ahorrándose el presupuesto del diseño y la implantación de estos sistemas. Empresas como Hortonworks [25], Cloudera [26] o Databricks [27], ofrecen sistemas de análisis de datos a empresas que utilizan las herramientas mencionadas anteriormente, centrando su negocio en el cobro por

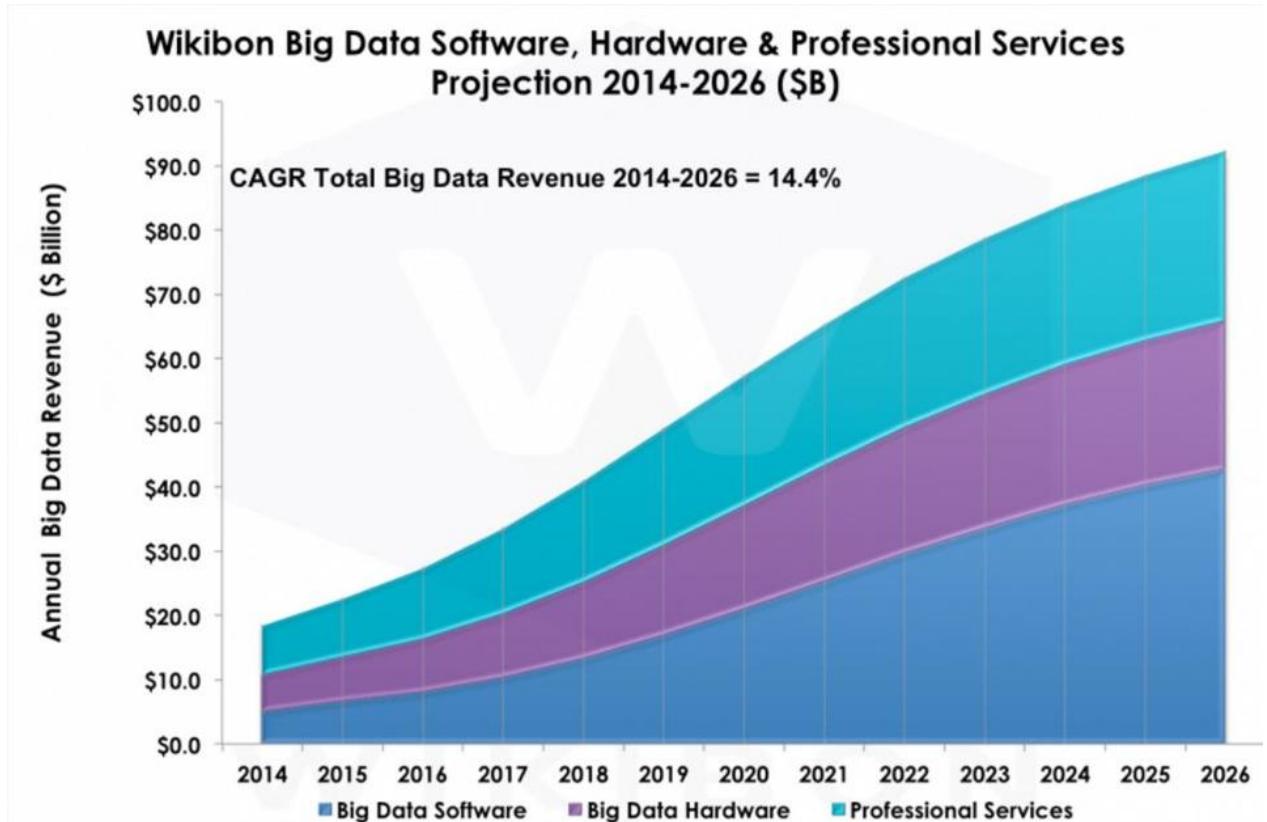
el uso de recursos computacionales y el servicio técnico.

A este tipo de servicio que ofrecen estos negocios se les conoce como Big Data as a Service (BDaaS) y se definen como servicios en la nube que permiten al usuario la capacidad de recoger, almacenar, analizar, visualizar y manejar los datos usando *big data*.

El futuro del *big data* parece prometedor, como se puede apreciar en la figura 2.5, proyectándose un crecimiento del 14.4 % anual en los ingresos generados por el *big data*, pasando de los 18 billones de dólares en 2014 a los 92 en 2022. En general, las estadísticas de crecimiento son muy favorables para este sector tecnológico, con predicciones de entre el 4 % al 15 % [28].

Sin embargo, las tecnologías *big data* no solo mejorarán los resultados económicos de las empresas. Estas, influirán directamente en los usuarios, mejorando las relaciones de estos con los servicios, principalmente, mediante la personalización de estos en base al cliente. También aumentarán la transparencia, al disponerse de más datos, permitiendo al usuario tener más conocimiento sobre las opciones disponibles.

Figura 2.5: Proyección de los ingresos del *big data* [28]

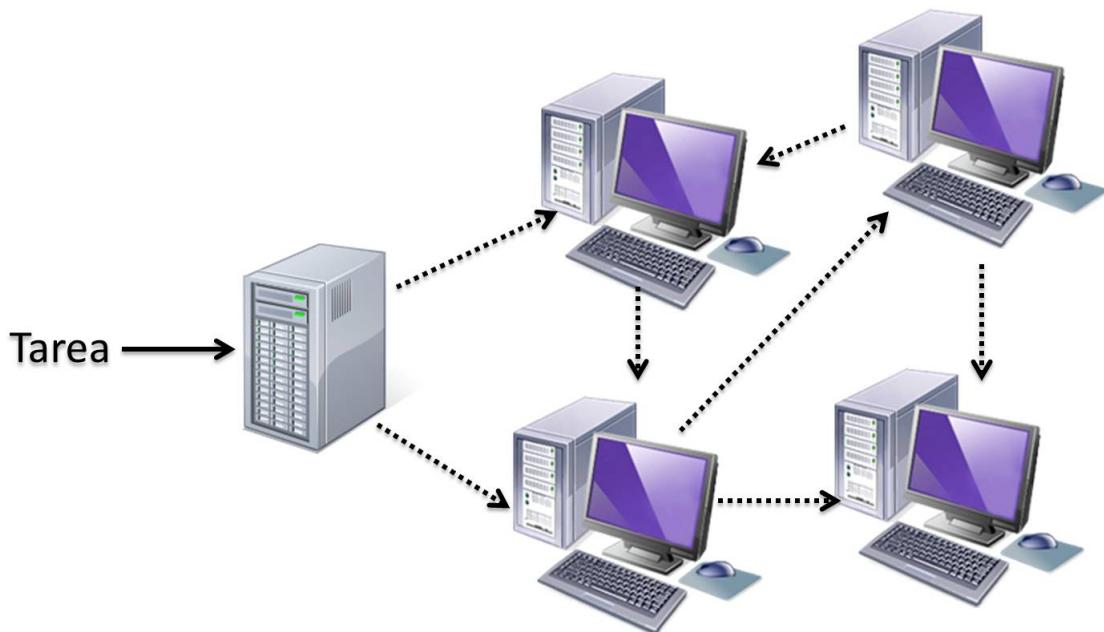


## 2.3. Sistemas distribuidos

La computación distribuida es un modelo de computación que consiste en la conexión de diferentes máquinas, que pueden estar en localizaciones físicas distintas, con el fin de realizar tareas en un tiempo menor del que necesitaría una sola máquina. Es decir, un sistema distribuido es una red de equipos autónomos que se comunican entre ellos para conseguir un objetivo. Estos son independientes en el sentido que no comparten memoria o procesadores físicamente [29].

Estos ordenadores se comunican entre ellos por “mensajes” que son fragmentos de información que se intercambian por la red. Estos mensajes sirven para ordenar diferentes tareas con argumentos particulares, para enviar y recibir paquetes de datos o para mandar ordenar comportamientos específicos de otros ordenadores. Estas máquinas pueden tener diferentes roles dentro del sistema, dependiendo de su objetivo.

Figura 2.6: Esquema de computación distribuida [30]



Las ventajas que ofrece este tipo de computación son [31]:

- **Compartición de recursos:** Este modelo de computación permite que las máquinas compartan recursos de hardware al estar conectadas en red.
- **Escalabilidad:** Gracias a la capacidad de compartir recursos, nuevos sistemas pueden ser añadidos a la red y aumentar la capacidad del conjunto.
- **Tolerancia a fallos:** Al contarse con varias máquinas, la caída de alguna de ellas de la red no será crítico, permitiendo la continuidad del sistema.

- **Concurrencia:** Este modelo permite que se ejecuten varios procesos al mismo tiempo dentro de la red.
- **Variedad:** Se pueden incorporar máquinas con diferente hardware y software debido al uso de protocolos estándar.

A pesar de estas ventajas, también cuenta con inconvenientes como los siguientes:

- **Complejidad:** Por el hecho de que se cuenta con varias máquinas, la organización y control de estos sistemas es más complicado que el de una sola máquina.
- **Seguridad:** Debido a que existen conexiones remotas que conectan a los equipos del sistema, se puede dar el caso de escuchas e interceptación de mensajes.
- **Disparidad en el comportamiento:** Se puede dar el caso de obtención de resultados diferentes dependiendo de las condiciones.

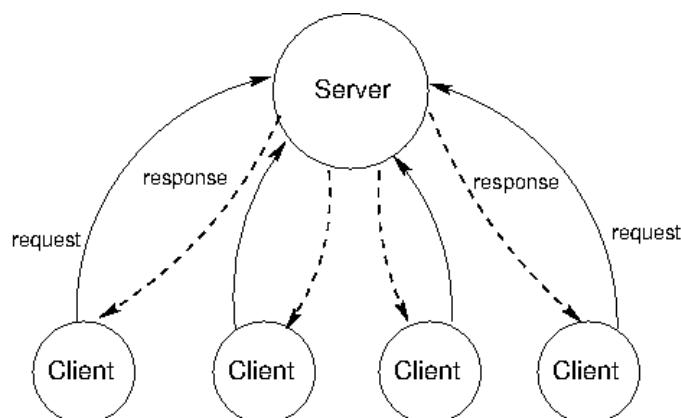
### 2.3.1. Configuraciones

Existen dos arquitecturas predominantes para la creación de sistemas distribuidos, la configuración cliente-servidor y la “peer-to-peer”.

#### Cliente-servidor

La arquitectura cliente-servidor es una forma de ofrecer servicios desde una fuente central, es decir, existe un solo servidor a los que múltiples clientes se conectan para consumir sus productos.

Figura 2.7: Esquema cliente-servidor [29]



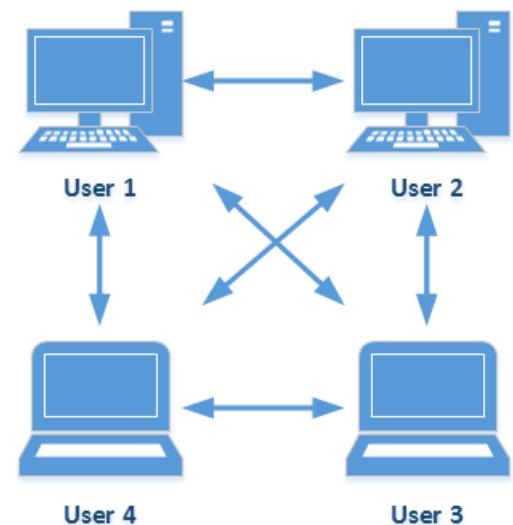
En este tipo los roles de los equipos son diferentes, mientras que el servidor es responder a las peticiones de los clientes, el trabajo de los clientes es usar los datos recibidos desde el servidor para realizar alguna tarea.

Internet es un ejemplo muy importante de este tipo de arquitectura ya que esta basada en ella, donde el servidor es una máquina que ofrece la página web a los sistemas que se conectan con ella. Sin embargo este sistema tiene dos importantes inconvenientes, por un lado, si se cae el servidor se paraliza el servicio totalmente. Por otro, este modelo no tiene capacidad de escalar, por lo que cuando hay muchos clientes conectados el rendimiento disminuye.

### Peer-to-peer

Son aquellos sistemas distribuidos en los que el trabajo está dividido entre todos los componentes del sistema. Todos las máquinas envían y reciben datos y aportan capacidad de computación y memoria al sistema, siendo este más potente cuanto más integrantes tenga la arquitectura.

Figura 2.8: Esquema peer-to-peer [32]



La división del trabajo entre todos las máquinas es el aspecto clave de esta arquitectura y, para mantenerla, mantener un comunicación fiable es básico. Por ello, para asegurar que los mensajes llegan entre los componentes, la red de un sistema peer-to-peer es estructurada y todos los nodos del mismo tienen que procurar en tener suficiente información de la misma para mantenerlo.

Existen variaciones de este sistema donde hay elementos que se preocupan de mantener la conexión entre todos los equipos de la red, de controlar la información del sistema y proporcionarla y organizar las tareas del conjunto. Este tipo de sistema es el que utilizaremos en este proyecto, donde habrá un maestro que organizará las tareas y los esclavos que realizarán el procesamiento.

Las aplicaciones más comunes de este tipo de arquitecturas son la transferencia de datos y almacenamiento. También aplicaciones como Skype funcionan con esta tecnología.

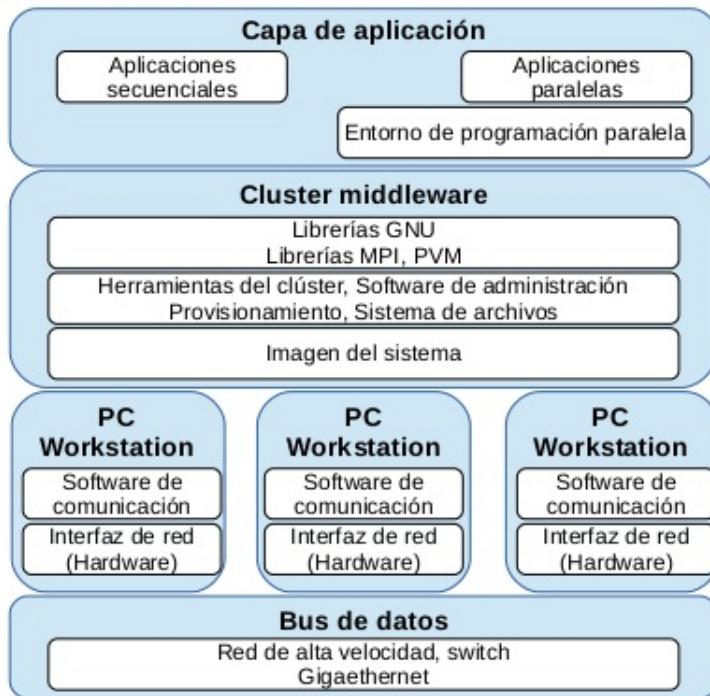
### 2.3.2. Arquitectura utilizada

Como hemos comentado anteriormente, en este proyecto vamos a usar una variación de la arquitectura de sistemas distribuidos peer-to-peer, donde tendremos una máquina que hará de maestro y será la que se encargue de establecer la conexión con el resto de elementos, repartir las tareas y recolectar los datos. El resto hará de esclavos de esta máquina, realizando las tareas encargadas.

#### Clúster o multinodo

Un clúster es un conjunto de ordenadores conectados entre sí por una red de alta velocidad y que se comportan como si fueran un único sistema [33].

Figura 2.9: Esquema de un clúster [34]



La tecnología de clústeres es usada en tareas de cómputos o supercómputos, en servidores web y comercio electrónico y en bases de datos de alta disponibilidad. Es decir, las características de que se esperan de un clúster son las siguientes [33]:

- **Alto rendimiento:** Debido a que se trata de un conjunto de ordenadores, la suma de su potencia hace que sean óptimos para tareas que requieran gran capacidad de computación y sean paralelizables.
- **Alta disponibilidad:** Por la misma razón, los clústeres deben ofrecer una rápida capacidad de recuperación ante fallos, por ejemplo, pudiendo recuperar los datos de

un nodo perdido al guardar copias en otros o evitando la caída del sistema por el fallo en un nodo, siendo este sustituido.

- **Balanceo de carga:** Un clúster debe distribuir el trabajo entre todos los nodos de la manera más equilibrada posible.
- **Escalabilidad:** El sistema tendría que tener facilidad para adaptarse a las diferentes situaciones, siendo posible añadir o reducir capacidad del sistema mediante la adición o extracción de nodos del mismo.

Los elementos de un clúster son:

- **Bus de datos:** Que conecta a los nodos del clúster y permite la comunicación entre ellos.
- **Nodos:** Es el conjunto de máquinas que forman el clúster, todas ellas tendrán que estar conectadas a la red y disponer de un software, el Sistema Operativo (S.O.) que deberá ser multiproceso y multiusuario.
- **Middleware:** Que permitirá el entendimiento entre las aplicaciones y el sistema operativo de los nodos. Es el encargado de hacer que el clúster sea visto por el usuario como una única entidad y, por otro lado, es el encargado de realizar el balanceo de tareas y permitir la escalabilidad del sistema.

### 2.3.3. Nuevos roles de trabajo

Los datos se han convertido en un nuevo modelo de negocio por derecho propio, esto propicia la aparición de empresas dedicadas a ellos y por tanto nuevos roles de trabajo. Estos precisan de gente especializada en diferentes áreas para poder llevar a cabo estas fases de análisis y procesos.

- **Data Scientist:** Este perfil es el encargado de la extracción de valor y conocimiento de los datos, donde debe analizar la información, realizar modelos predictivos y realizar los diferentes reportes que sean necesarios. Para ello se requieren conocimientos de estadística, matemáticas, programación, Machine Learning (ML), bases de datos y un largo etc.
- **Big Data Engineer:** Perfil técnico responsable del flujo de información “end-to-end” desde un punto de vista técnico, donde principalmente se debe de encargar de integrar las diferentes fuentes de datos y desarrollar procesos de transferencia. En definitiva desarrolla, construye, prueba y mantiene bases de datos y sistemas de procesamiento de datos a gran escala.
- **Big Data Architect:** Perfil técnico encargado del diseño, despliegue y gestión de la arquitectura *big data*. Crea sistemas de gestión integrados para centralizar, proteger y mantener las fuentes de datos, se les requiere un profundo conocimiento del ecosistema *Apache Hadoop* y *UNIX* [35].

- **Big Visualization Specialist:** Perfil responsable del diseño y creación de visualizaciones de datos de gran impacto, para ello se requiere conocimientos de arquitecturas webs, desarrollo de Interfaz de Programación de Aplicaciones (API), conocimientos de “full stack” [36], consultas a base de datos y herramientas comerciales de visualización.
- **Big Data Business Consultant:** Perfil responsable de la interlocución directa con el negocio, diseñar con el cliente la estrategia de implementación de la tecnología y el desarrollo de casos de uso analíticos para identificar las necesidades de negocio susceptibles de ser satisfechas con analítica avanzada de datos. Requiere conocimiento del sector y del negocio concreto, así como las tendencias de mercado y conocimientos sobre la tecnología *big data*.

## 2.4. Infraestructura

### 2.4.1. Apache Spark

*Apache Spark* [3] es un Framework de código abierto que permite el procesamiento de datos mediante la computación distribuida. Desarrollado originalmente por el departamento AMPLab [37] de la Universidad de California, fue, posteriormente, donado a la fundación Apache, que lo ha mantenido desde entonces.

Desarrollado en Scala [38] y con soporte para Java, Python y R, *Apache Spark* proporciona una API para programar clúster con paralelismo de datos implícito y tolerancia a fallos. Esta API se apoya en la estructura llamada RDD, donde se almacenan los datos particionados para permitir transformaciones en paralelo y que mantiene el índice de estas transformaciones para rehacerlas en caso de error.

Una buena característica de este Framework es el amplio soporte para la mayoría de los formatos de ficheros de datos, teniendo, además, integraciones con varios sistemas de almacenamiento como HDFS, *Apache Cassandra* [21] o Amazon S3 [39]. En la figura 2.10 podemos observar los diferentes componentes que conforman *Apache Spark*.

El Framework está formado por una base, sobre la que trabajan las diferentes librerías y paquetes del mismo. Dicha base está formada por el núcleo del programa, donde se crean y manipulan los RDD, y la *DataFrame* API, que es una abstracción a más alto nivel de los RDD, para simplificar su manipulación por el usuario.

Además *Apache Spark* cuenta con diversas librerías como Spark SQL, para realizar consultas sobre los datos, Spark Streaming, para permitir la introducción y análisis de datos en tiempo real, MLlib, para realizar aprendizaje automático, y Graphx, que permite la computación de grafos.

### Funcionamiento

*Apache Spark* basa su funcionamiento en dos conceptos, RDD y DAG. Una aplicación consiste en un controlador, llamado *SparkContext*, que utiliza el código de usuario para crear y transformar RDD y alcanzar el objetivo final. Estas transformaciones de los RDDs son traducidos a gráficos DAG para ser enviados al planificador y que este reparta las tareas.

**RDD: Resilient Distributed Dataset.** Esta es la solución clásica en *Apache Spark* para tratar los datos de una forma paralela y a prueba de errores. Además ofrece diferentes APIs para realizar transformaciones sobre los datos y permite el control sobre el modo de particionar los datos y el caché de los mismos.

Un RDD puede ser creado a partir de un fichero en un almacenamiento externo o a partir de otro RDD y es un sistema vago (*lazy*), es decir, se guardan las operaciones

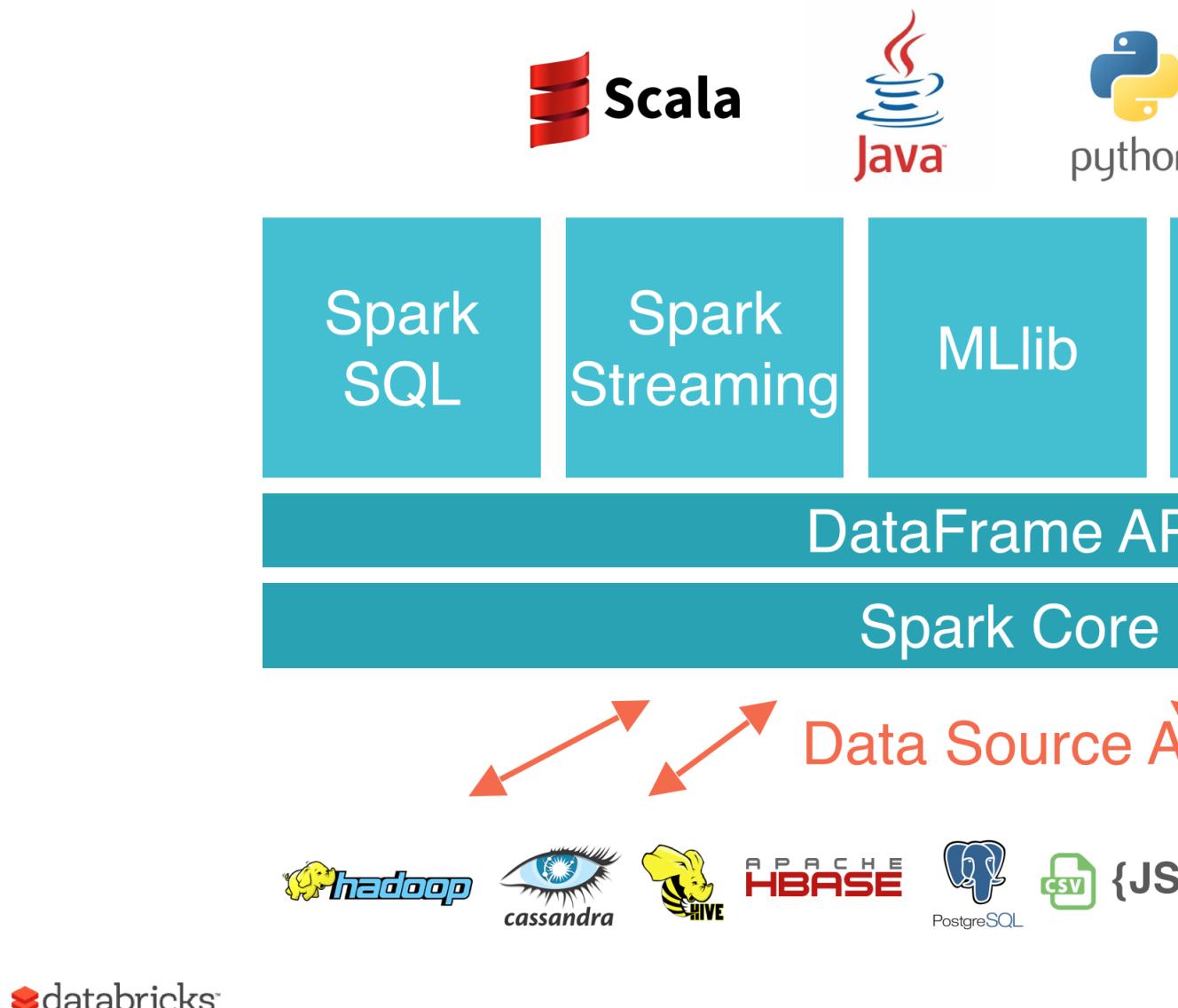
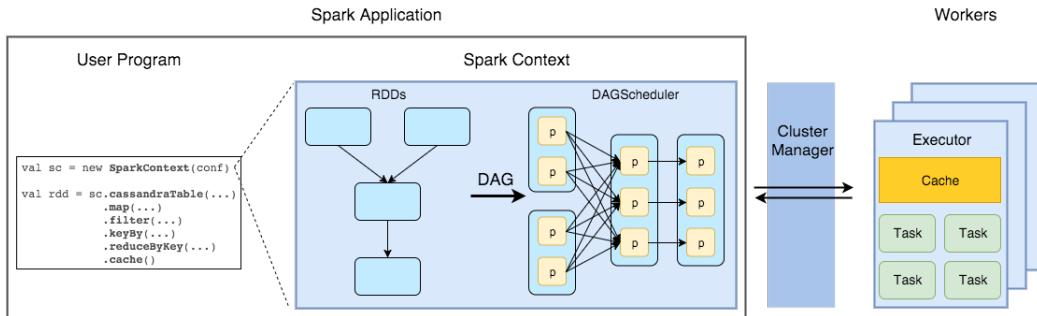
Figura 2.10: Componentes de *Apache Spark* [40]

Figura 2.11: Funcionamiento de *Apache Spark* [40]

que se realizarán sobre los datos, pero estas no son ejecutadas hasta que no se realiza la evaluación de los datos. Este sistema también es utilizado para la evitar los errores, donde el sistema sigue los pasos realizados desde el inicio para recomponer los datos correctamente.

Existen diferentes grupos de operaciones que se pueden realizar sobre un RDD, estos son:

- **Transformaciones:** Permite realizar las funciones codificadas por el usuario sobre todos los datos del sistema, permite operaciones de agrupación, ordenamiento y particionamiento del RDD. Estas operaciones son reflejadas en el grafo DAG.
- **Acciones:** Son las que inician el trabajo, es decir, hacen que se ejecuten las transformaciones estables
- **Persistencia:** Permiten almacenar el RDD en memoria o en disco de forma explícita.

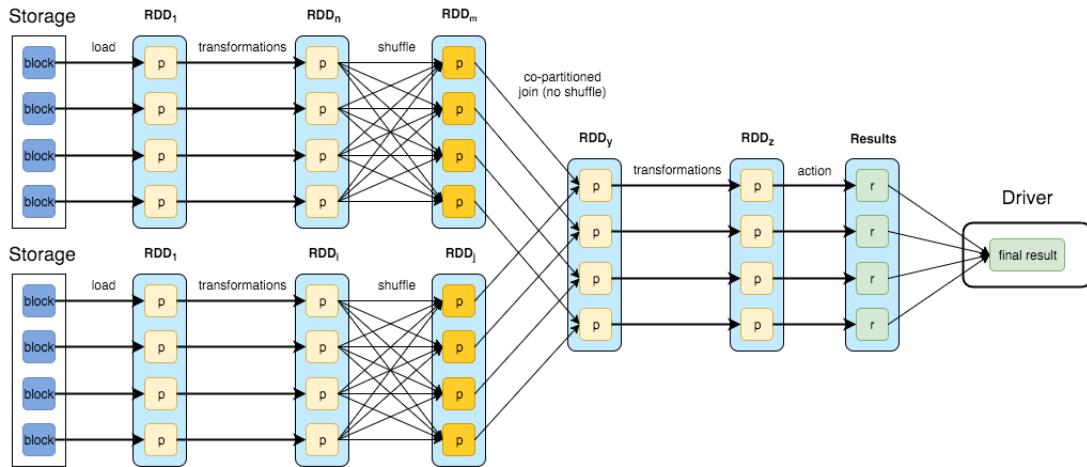
**DataFrame.** Esta es una variante de los RDD incluida en el modulo SQL que permite consultar los datos con una “query” como si se tratase de una base de datos.

En *Apache Spark*, un *DataFrame* es una colección distribuida de datos organizados en columnas nombradas. Es conceptualmente equivalente a una tabla en una base de datos relacional o un marco de datos en R/Python, pero con optimizaciones más ricas bajo el capó. Un *DataFrame* se puede construir a partir de una amplia gama de fuentes como: archivos de datos estructurados, tablas en Hive, bases de datos externas o RDD existente.

**DAG: Direct Acyclic Graph.** [41] Es la forma de codificación de los trabajos sobre los RDDs o *DataFrame* utilizada en *Apache Spark*. Indica el flujo que seguirá la aplicación durante la ejecución que, generalmente, se basa en leer los datos del origen, realizar las transformaciones oportunas y materializar los datos obtenidos.

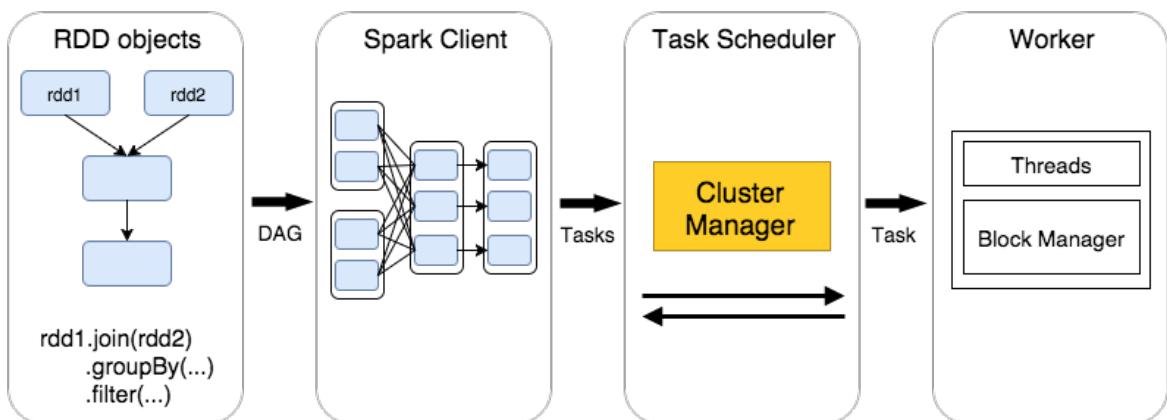
Este componente también es el que se encarga de establecer las diferentes tareas del trabajo que, posteriormente, serán repartidas entre los nodos del clúster para parallelizar el trabajo.

Figura 2.12: Transformaciones sobre un RDD representadas en un DAG [40]



Una de las características que diferencia *Apache Spark* es la forma en la que gestiona los datos, este Framework guarda los datos en la memoria Memoria de Acceso Aleatorio (RAM) del dispositivo, sin llegar escribir a disco, por lo que el acceso a estos es muy rápido. Por ello, hace que la velocidad en procesos iterativos, donde se consultan los mismos datos de forma continua, sea muy elevada, mejorando con creces los tiempos de otros sistemas como MapReduce [42].

Figura 2.13: RDD distribuido en diferentes máquinas [43]



## Librerías

Como se ha indicado anteriormente, *Apache Spark* incluye diferentes librerías o módulos que trabajan sobre la base del Framework. Estas son:

- **Spark SQL:** Permite el soporte para datos estructurados o semi-estructurados para realizar consultas SQL sobre ellos. Estos datos pueden ser modificados con funciones específicas de *Apache Spark* o mediante el lenguaje SQL.
- **Spark Streaming:** Permite el análisis en tiempo real de datos mediante la introducción de estos en mini lotes de datos. Este sistema aunque cumple su función no es tan potente como el ofrecido por otras alternativas como *Apache Flink* [44] o *Apache Storm* [45].
- **MLib:** Es un Framework de aprendizaje automático distribuido. Este permite diferente tipos de análisis sobre los datos: clasificaciones, regresiones, clusterizaciones, optimizaciones, etc.
- **GraphX:** Es un procesador de grafos distribuido. Es muy veloz para grafos que no tienen que ser actualizados, sin embargo, debido a la naturaleza inmutable de los RDD no es una herramienta válida para grafos que se modifiquen.

### 2.4.2. Apache Hadoop

*Apache Hadoop* [4] es otro Framework de código abierto que permite el tratamiento de grandes cantidades de datos de forma distribuida. Desarrollado en Java, permite la gestión de clústeres de un nodo hasta cientos de ellos de forma sencilla. Al establecer sistemas distribuidos estos tienen una alta tolerancia a los fallos.

*Apache Hadoop* cuenta con cuatro módulos principales, aunque cuenta con muchos más proyectos totalmente compatibles:

- **Hadoop Common:** Utilidades comunes que soportan el resto de módulos.
- **Hadoop Distributed File System (HDFS)** Sistema de ficheros distribuido que permite la replicación de los datos en los nodos del sistema.
- **Hadoop YARN:** Es un Framework que se encarga de la gestión de los trabajos y los recursos del clúster.
- **Hadoop Mapreduce:** Sistema basado en YARN para el procesamiento paralelo de grandes cantidades de datos.

En este proyecto, debido a que utilizamos *Apache Spark* para la gestión del clúster, no utilizaremos la mayoría de los módulos de *Apache Hadoop*. El módulo que se utilizará será el sistema de ficheros HDFS por la configuración distribuida del clúster doméstico.

Con este sistema realizaremos la replicación de los ficheros de datos en los diferentes nodos del sistema y, de esa forma, ahorrar la transmisión de estos durante la ejecución de los trabajos y, así, mejorar los tiempos de procesamiento.

#### **Hadoop Distributed File System (HDFS)**

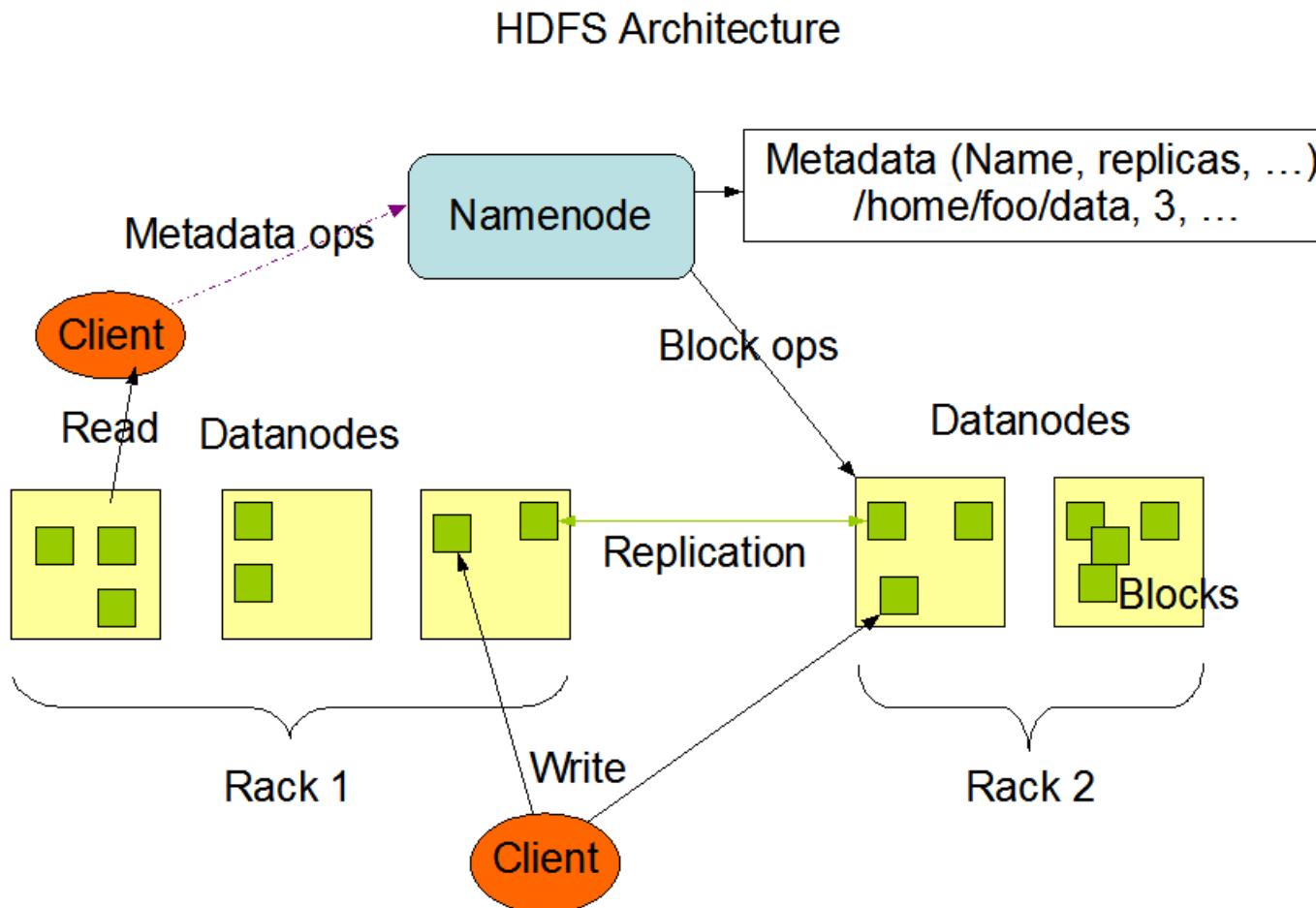
HDFS es un sistema de ficheros distribuido, escalable y portable escrito en Java. Este permite el almacenado de grandes cantidades de datos mediante la distribución de estos en diferentes máquinas. HDFS es muy tolerante a los fallos y está diseñado para correr en hardware de bajo coste.

HDFS fue construido con las siguientes asunciones y objetivos:

- **Errores de hardware:** Los fallos de hardware son más una norma que una excepción en la realidad, por ello, HDFS es muy tolerante a los fallos, distribuyendo y replicando los datos en diferentes máquinas del clúster para no perder el acceso a ellos por la caída de algún nodo.
- **Acceso al streaming de datos:** HDFS fue concebido para aplicaciones que necesitan un buen rendimiento en el acceso a los datos, más que para su manipulación continua por los usuarios. Por ello, se centra más en el procesamiento en lotes.

- **Grandes volúmenes de datos:** HDFS está afinado para manejar ficheros de datos que van desde los cientos de gigabytes hasta terabytes de tamaño.
- **Modelo de coherencia simple:** Las aplicaciones que usan este sistema necesitan el acceso a la lectura del fichero más que sus modificaciones. Por ello, el sistema del HDFS usa un modelo “escribe una vez y lee el resto”, por lo que una vez que se crea un fichero, se escribe y guarda, este no vuelve a modificarse excepto para reducir la cantidad de datos o añadir más.
- **“Mover la capacidad de computación es más barato que mover los datos”:** Una aplicación es más eficiente si la computación se realiza cerca de los datos, ya que ahorras el tiempo de transporte de estos. Esta es la premisa que sigue HDFS y por ello se replican los datos en todos los sistemas para realizar las operaciones sobre estos en los propios nodos.
- **“Portabilidad entre hardware y software heterogéneo”:** HDFS ha sido diseñado para ser fácilmente portado entre sistemas y plataformas.

Figura 2.14: Arquitectura del HDFS [4]



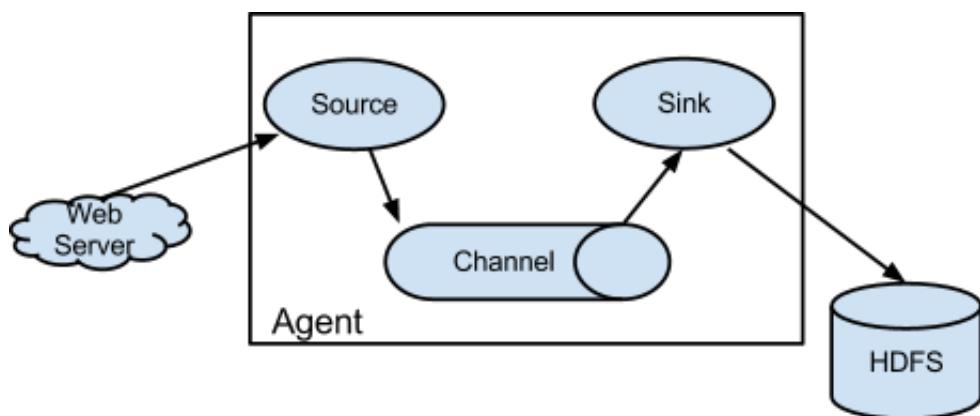
En la figura 2.14 podemos ver la arquitectura de un sistema HDFS, este esta basado en la estructura maestro/esclavo, donde existe una *Namenode* que hace de maestro y gestiona el sistema de ficheros y regula el acceso a los ficheros por los clientes. Por otro lado, están los *Datanodes*, uno por nodo, que es donde se almacenan y gestionan los datos del sistema.

Por otro lado, dentro de estos *Datanodes* encontramos los bloques, que son partes o fragmentos de los datos originales y que se replican por los diferentes nodos.

### 2.4.3. Apache Flume

*Apache Flume* [46] es un sistema distribuido, seguro y eficiente para recoger, agregar y mover grandes volúmenes de datos provenientes de *logs* desde distintas fuentes a un almacen de datos centralizado. *Apache Flume* trabaja con agentes, que son procesos independientes que alojan los componentes de *Apache Flume*.

Figura 2.15: Arquitectura del HDFS [47]



En la figura 2.15 se pueden ver los componentes un agente de *Apache Flume*. Estos están conformados de la siguiente forma:

- **Evento:** Un *payload* [48] de bytes con encabezados opcionales que representan la unidad de datos que Flume puede transportar desde su punto de origen hasta su destino final.
- **Cliente:** Implementación que opera en el punto de origen de los eventos y los entrega a un agente Flume. Por ejemplo, Log4J appender, es un cliente de Flume.
- **Fuente (*source*):** Implementación que puede consumir eventos entregados a él a través de un mecanismo (por ejemplo, la monitorización de un directorio). Cuando una fuente recibe un evento, este se lo entrega a uno o más canales.
- **Canal (*channel*):** Es un almacenamiento temporal para eventos, donde estos se entregan a través de las fuentes que operan con el agente. Un evento permanece en el canal hasta que es recogido por un sumidero (*sink*).

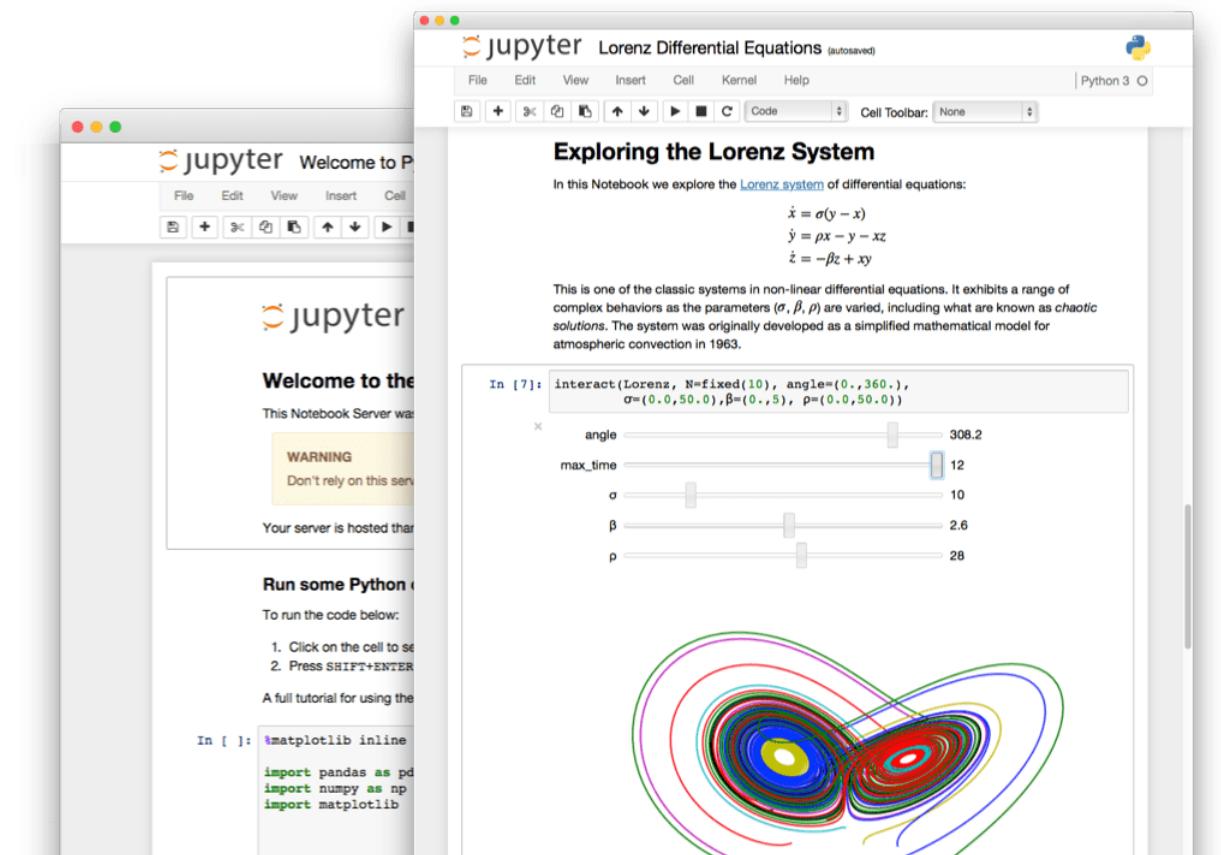
- **Sumidero (*sink*):** Implementación que puede eliminar eventos de un canal y transmitirlos al siguiente agente en el flujo, o hasta el destino final del evento.

#### 2.4.4. Jupyter Notebook

Jupyter [49] es un nuevo entorno de trabajo *open source* oriente a científicos de datos que soporta los lenguajes R y Python, entre otros.

Jupyter surge en 2014 como una evolución del proyecto IPython [50], una potente consola vitaminada para Python. Sin embargo, Jupyter es mucho más ambicioso que IPython, se pretende construir una plataforma agnóstica del lenguaje que ofrezca a los científicos un conjunto de potentes herramientas para trabajar con datos, visualizarlos y poder compartir los resultados.

Figura 2.16: Previsualización del entorno de Jupyter Notebook [49].



Jupyter nos ofrece una shell interactiva vía web, a la que podemos acceder desde un navegador. La shell está organizada en pequeños bloques, cada bloque puede contener texto arbitrario formateado en Markdown [51], fórmulas matemáticas en LaTeX [52], código en multitud de lenguajes, resultados, gráficos, vídeos, widgets o cualquier elemento multimedia.

Podemos escribir código de programación en estas celdas e ir ejecutándolo paso a paso o todo de golpe, obteniendo todos los resultados parciales. También podemos usar los bloques de texto para documentar el código o añadir las explicaciones oportunas, que pueden contener enlaces, imágenes, vídeos u otros elementos.

Esta serie de piezas de código, notas y resultados se guardan en un notebook, que es un fichero que contiene toda esta información. Uno de los principales objetivos de Jupyter es fomentar y simplificar la compartición de conocimiento y resultados a través de los notebooks. Plataformas como GitHub [53] o Databricks Community [27] facilitan esta tarea. De esta manera los notebooks pueden ser fácilmente difundidos y los resultados pueden ser reproducidos y validados en diferentes entornos. Por supuesto esto es muy útil para la divulgación y la formación o en entornos educativos.

Jupyter soporta integración con más de 40 lenguajes de programación en los que podemos escribir el código de nuestro notebooks, por ejemplo Python, R, Scala, Ruby o Go. Pero también es fácilmente integrable con herramientas y plataformas de *big data* como *Apache Spark*, lo que permite abstraerse de la complejidad de estas herramientas, aprovechando todo su potencial desde un entorno muy amigable.

# 3

## Marco Regulador

### 3.1. Legislación aplicable

Como se ha comentado anteriormente, la tecnología *big data* permite el procesado de grandes cantidades de datos de forma más rápida que con tecnologías anteriores, con el propósito de obtener información valiosa de ella y generar conocimiento.

Este gran volumen de datos disponibles proviene de diferentes fuentes de información, que abarcan desde reportes anónimos hasta informes de uso de usuarios, donde estos están perfectamente identificados. Son este último tipo de datos los que pueden producir problemas legales, debido a la legislación vigente respecto a la privacidad de las personas.

La normativa nacional establece que “un dato de carácter personal es cualquier información que permita identificarte o hacerte identifiable” y, por ello, “reconoce al ciudadano la facultad de controlar sus datos personales y la capacidad para disponer y decidir sobre los mismos” mediante el derecho fundamental a la protección de datos [54].

En España, es la Agencia Española de Protección de Datos (AEPD) “la autoridad de control independiente que vela por el cumplimiento de la normativa sobre protección de datos”. Además, “garantiza y tutela el derecho fundamental a la protección de datos personales” [54].

En la actualidad, es la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal [55] la que afecta al procesamiento de información que se realiza en los sistemas *big data*. Esta, tiene que objetivo “garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar” [55]. Los derechos que incluye esta ley son [56]:

- Derecho de información: En el momento en que se procede a la recogida de los datos personales, el interesado debe ser informado previamente.
- Derecho de acceso: permite al ciudadano conocer y obtener gratuitamente información sobre sus datos de carácter personal que han sido tratados.
- Derecho de rectificación: permite corregir errores, modificar los datos que resulten ser inexactos o incompletos y garantizar la certeza de la información tratada.
- Derecho de cancelación: permite que se supriman los datos que resulten ser inadecuados o excesivos.
- Derecho de oposición: permite al afectado que el tratamiento de sus datos de carácter personal no se realice o el cese de los mismo.

Con respecto a la Unión Europea, la ley que actualmente regula la protección y privacidad de los datos de carácter personal es el Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos [57].

Esta legislación europea ha sido reformada recientemente y se espera que para 2018 se aplique totalmente en todos los países de la zona euro. En esta reforma destacan, entre otras, la reforma al Derecho al Olvido, las grandes multas por el incumplimiento de las leyes de privacidad y el endurecimiento de los controles parentales [58]. En general, esta reforma ha sido un endurecimiento de las medidas ya existentes para salvaguardar los derechos de los ciudadanos en esta nueva época digital.

Por tanto, con respecto al marco regulador que afecta a las aplicaciones del *big data* se han de tener en cuenta aspectos como la seguridad, privacidad y correcta conservación de los datos personales de los usuarios, para proteger los derechos de los ciudadanos. De la misma forma, también debe asegurarse la transparencia de su uso frente a las autoridades.

### 3.2. Estándares técnicos

La utilización del *Big Data* de forma masiva para el análisis de datos en el mundo empresarial, como ya se ha comentado, es una técnica relativamente reciente, por lo que no se ha producido un gran desarrollo con respecto a estándares de uso.

El primer estándar que se desarrolló data de noviembre de 2015, cuando la International Telecommunication Union (ITU), agencia que depende de la ONU y que es responsable de los problemas que conciernen a las tecnologías de información y comunicación, desarrolló y publicó el documento ITU-T Y.3600 (11/2015) [14] titulado “Big data - Cloud computing based requirements and capabilities”.

En este documento “se presentan los requisitos, las capacidades y los casos de uso de los volúmenes masivos de datos (*big data*) de computación en la nube, así como su contexto de sistema” [14]. Además, en este documento se establece la definición de *big data* comentada en el apartado 2.2.2.

# 4

## Diseño

### 4.1. Introducción

Tras el análisis del panorama y las tecnologías *big data* y, en especial, de las que van a ser utilizadas en este proyecto, en este apartado, se procederá con la explicación del diseño de la arquitectura *big data* que, posteriormente, se implementará en un clúster doméstico.

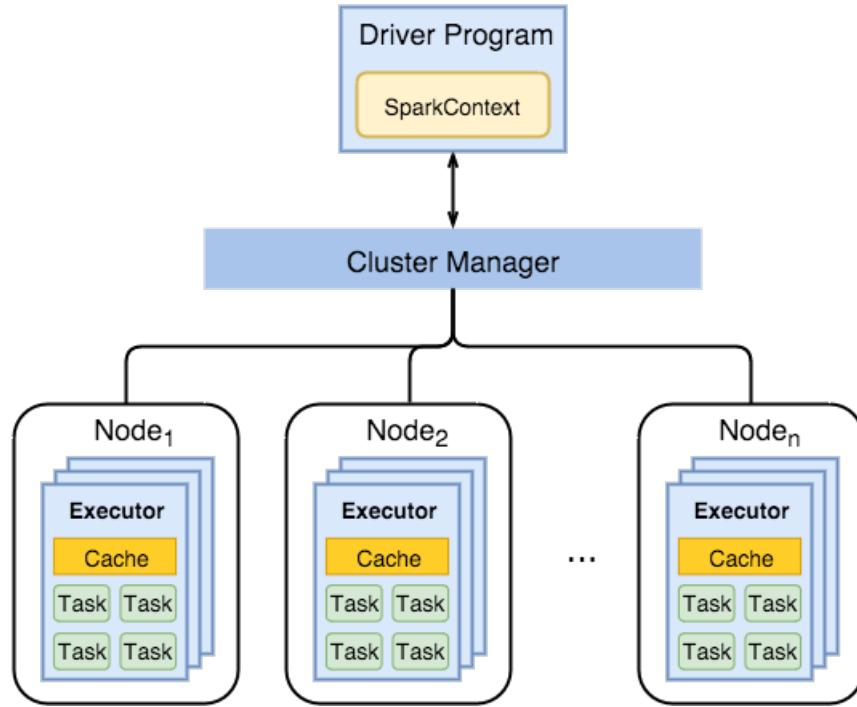
En cuanto a la estructura de este apartado, en primer lugar procederemos a presentar la arquitectura del clúster, posteriormente se analizarán los *insights* del proyecto y las fuentes que se utilizarán.

### 4.2. Arquitectura del sistema

#### 4.2.1. Modo distribuido o multinodo

En este modo de ejecución se cuenta con más de una máquina para ser utilizada por el sistema, siendo la configuración utilizada en la gran mayoría arquitecturas *big data* que se utilizan.

En este tipo de ejecución se cuenta con una máquina que hará de maestro y distribuirá las tareas sobre el resto de nodos, los esclavos, y ella misma, para lograr la máxima eficacia de procesamiento. Es decir, el maestro creará un objeto *SparkContext* que será el que distribuya los trabajos entre los nodos conectados y recogerá los resultados, como se muestra en la figura 4.1 [34].

Figura 4.1: Modo distribuido o multinodo de *Spark* [34]

En este caso, la ventaja de esta configuración es la gran capacidad de procesamiento que se puede obtener, especialmente si las tareas requieren gran cantidad de potencia de procesamiento, al disponer de la suma de los procesadores y de la memoria RAM de los nodos del sistema.

La principal desventaja de este, que es su principal punto débil, se encuentra en la conexión entre los nodos, cuya calidad y velocidad afectará al rendimiento del sistema de forma muy importante. Es decir, el problema en los sistemas distribuidos proviene de la conexión de red entre los equipos, donde se pueden producir cuellos de botella en el envío de datos entre ellos. Estos pueden hacer que aunque la potencia teórica del sistema sea muy alta se obtengan malos rendimientos de procesamiento que hagan que el sistema no resulte rentable.

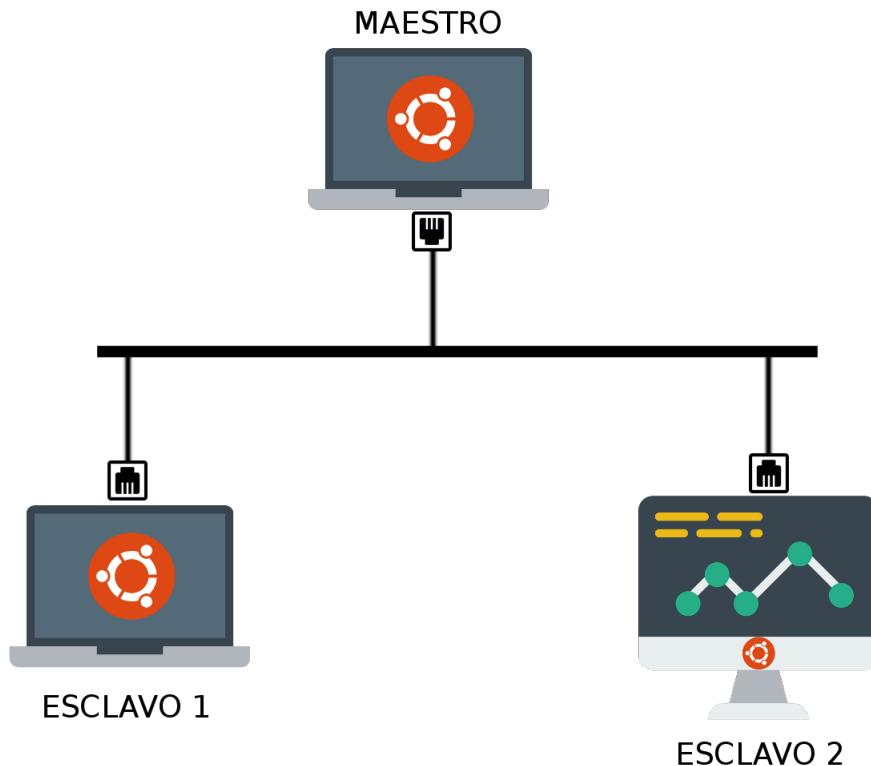
Otro de las desventajas de esta configuración es el coste de mantenimiento del conjunto de las máquinas y los dispositivos necesarios para mantener el clúster en funcionamiento.

## Clúster

El número de máquinas que formarán este clúster serán dos, una de sobremesa que hará de maestro y esclavo y un portátil que hará de esclavo, un portátil y un sobremesa, como se puede observar en la figura 4.2. Todos estos nodos estarán conectados a la red mediante cables Ethernet RJ45 Cat.5e UTP y un switch (TP-LINK TL-SG108) y contarán con Ubuntu 16.10 [59] como versión del sistema operativo.

*Apache Hadoop* es la tecnología utilizada para montar este mecanismo de replicación

Figura 4.2: Arquitectura del clúster doméstico

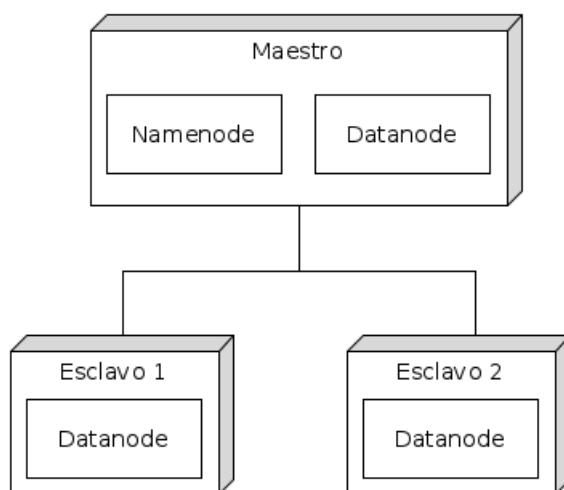


mediante el uso de su sistema distribuido de ficheros (HDFS) que resulta óptimo para esta configuración. Este mecanismo nos permitirá tener los datos en cada nodo del clúster y, así, lo único que circulará por la red serán las tareas que el maestro asigne a los esclavos y los resultados que estos obtengan tras su procesamiento.

Esto evitará el tráfico que generaría transmitir los datos a procesar con la tarea asignada a los nodos esclavos, mejorando el rendimiento de la arquitectura.

Como se aprecia en la figura 4.3 que muestra el esquema del sistema de replicación, para su funcionamiento, en cada nodo se necesita un proceso *datanode* ejecutándose, que será el que se encargue de el almacenamiento de los datos en la máquina. Además, para controlar el estado de los *datanodes* se necesita un proceso *namenode*, que se ejecutará en el nodo maestro.

Figura 4.3: Esquema del sistema de replicación HDFS



## 4.3. Insights del proyecto

En este trabajo debido a que se realiza un proyecto real de análisis para una supuesta empresa de impartición de cursos, hay ciertas preguntas que se desean resolver a lo largo del proyecto, estas “preguntas” en ámbito de *big data* se denominan *insights* que iremos resolviendo hasta llegar al informe final.

- ¿Cuáles son los lenguajes de programación más populares?
- En caso de regionalización ¿Qué países enfocar?
- ¿Qué idioma utilizan los programadores?

El primer paso para enfrentarnos a estos *insights* será definir el término “popularidad”, para luego proceder a la resolución de estas cuestiones.

### 4.3.1. Plan de proyecto

El proyecto constará de diferentes fases donde primero implementaremos el clúster anteriormente mencionado. Realizando un pequeño estudio preliminar encontramos que los programadores utilizan tres plataformas de forma predominante.

- **Stackoverflow [60].** Esta plataforma está dedicada a que los programadores planteen dudas, que son resueltas por otro profesionales. Cada *post* esta etiquetado con el lenguaje o tecnología sobre la que se realiza la cuestión.
- **GitHub [53].** Es un repositorio de versiones basado en *Git* [61], el cual siguiendo la filosofía de *open data* [5] proporciona una API desde la que se pueden descargar los eventos almacenados desde 2015 a través de *GitHub Archive* [62].
- **Twitter [63].** Es una plataforma de comunicación bidireccional con naturaleza de red social (porque permite elegir con quien te relacionas). Cuyos mensajes están limitados a 280 caracteres. Aquí los usuarios publican sus opiniones personales o anuncios que consideran importantes. Proporciona una API que permite obtener estos mensajes (*tweets*) en tiempo real.

# 5

## Instalación del clúster

### 5.1. Introducción

En este apartado se detallaran los aspectos relevantes para la implementación de la arquitectura *big data* diseñada en el apartado 4.

Como se indica en dicho apartado, se plantean una configuracion para el sistema multinodo, donde una máquina es el maestro y, el resto, los esclavos.

Por tanto, se comentaran todos los detalles y pasos para montar la arquitectura diseñada y, de esta forma, conseguir un sistema *big data* funcional y completo que sea capaz de almacenar, procesar y analizar los datos para obtener las respuestas de las consultas planteadas.

### 5.2. Preparación del entorno de trabajo

Como se expuso en el apartado 2.4.1, donde se habla de *Apache Spark*, este, resulta compatible con las tres sistemas operativos más importantes para ordenadores, aunque la solución escogida ha sido Linux para ejecutarlo. Se va a utilizar Ubuntu Desktop [59] para el master y Ubuntu Server para los slaves.

Aunque se podría ejecutar todo el clúster exclusivamente en un entorno Ubuntu Server, se ha decidido utilizar la variante Desktop en el master para poder realizar los análisis directamente desde el nodo.

### 5.2.1. Especificaciones del equipo

Tabla 5.1: Especificaciones del maestro

<b>GENERAL</b>	
<b>Nombre:</b>	david-hdp
<b>Sistema Operativo:</b>	Linux
<b>Versión:</b>	Ubuntu 16.10 (64 bits)
<b>Usuario:</b>	david
<b>SISTEMA</b>	
<b>Procesador:</b>	Intel(R) Core(TM) i7-3820 CPU @ 4.30GHz
<b>Cores:</b>	8
<b>Threads por core:</b>	2
<b>RAM:</b>	16384
<b>ALMACENAMIENTO</b>	
<b>Dispositivo:</b>	Kingston SUV400S37480G
<b>Tamaño:</b>	1 Tb
<b>RED</b>	
<b>Dispositivo:</b>	Realtek PCIe GBE Family Controller
<b>Nombre:</b>	Master

Tabla 5.2: Especificaciones del nodo1

<b>GENERAL</b>	
<b>Nombre:</b>	node1
<b>Sistema Operativo:</b>	Linux
<b>Versión:</b>	Ubuntu 16.10 (64 bits)
<b>Usuario:</b>	david
<b>SISTEMA</b>	
<b>Procesador:</b>	Intel(R) Celeron(TM) T1600 @ 2GHz
<b>Cores:</b>	2
<b>Threads por core:</b>	1
<b>RAM:</b>	8192
<b>ALMACENAMIENTO</b>	
<b>Dispositivo:</b>	Samsung HM320JI
<b>Tamaño:</b>	320 Gb
<b>RED</b>	
<b>Dispositivo:</b>	Realtek PCIe GBE Family Controller
<b>Nombre:</b>	node1

### 5.2.2. Instalación del sistema operativo

Esta instalación solo fue necesaria de llevar a cabo en los esclavos que utilizaríamos para el entorno doméstico. Como se ha comentado anteriormente, el S.O. elegido será Ubuntu, concretamente, la versión 16.10.

#### Requisitos previos

Para proceder con la instalación del sistema operativo en los ordenadores había que cumplir algunos requisitos muy básicos. Lo primero era contar con máquinas que superasen los requisitos mínimos [59] necesarios para su correcta ejecución, cosa que las utilizadas cumplían (especificaciones en el apartado 5.2.1).

También sería necesario un soporte de instalación del S.O., que en este caso fue un Digital Video Disc (DVD)), en el que se grabó una imagen con Ubuntu, que permitió el inicio de este sistema operativo en las máquinas y su posterior instalación en ellas, de forma local en el disco duro.

### 5.2.3. Instalación de Apache Spark

Para la realización de este proyecto utilizaremos la versión 2.2.0 *Apache Spark*, que era la versión disponible cuando se comenzó a trabajar en él. Como lo que se deseaba era una versión estable para realizar todo el trabajo, se descartó la creación del paquete a partir del código fuente y, también, la actualización a la versión 2.2.1 que salió en diciembre.

Por tanto, para la instalación de *Apache Spark* en los equipos que se utilizarían se recurrió a la versión pre-compilada del Framework [64], en concreto la compilada para funcionar con *Apache Hadoop 2.9* que se utilizaría en la configuración multinodo del clúster.

#### Prerrequisitos

Para que *Apache Spark* funcione de forma correcta en el sistema se necesita cumplir una serie de prerrequisitos:

- Tener instalado Linux en el sistema.
- Tener instalado Java en el sistema.
- Tener instalado Scala [38] en el sistema.
- Tener instalado Python en el sistema.
- Tener instalado SSH en el sistema.
- Disponer de un usuario y grupo común en todas las máquinas.

## Instalación de Java y Scala

La instalación de estos dos lenguajes es esencial para el funcionamiento de *Apache Spark* debido a que está escrito en Scala. Por otro lado, Java es necesario ya que Scala [38] necesita de la Máquina Virtual de Java (JVM) para su funcionamiento.

La instalación de estos dos lenguajes de programación se realiza a través de la terminal de forma muy sencilla, primero se añade el repositorio oficial de Oracle [65] y posteriormente se introduce los comandos de instalación. Los comandos necesarios se encuentran en el bloque de código 5.1.

Código 5.1: Instalación de Java y Scala.

```
sudo add-apt-repository ppa:webupd8team/java  
sudo apt-get update  
sudo apt-get install oracle-java8-installer  
sudo apt install scala
```

## Instalación de Anaconda (Python)

Python va a ser esencial para el desarrollo de este proyecto ya que se utilizará la API en este lenguaje para escribir todo el código que procese *Apache Spark*, conocida como *PySpark*. Además como Integrated Development Environment (IDLE) utilizaremos *Jupyter Notebook* [49], que es ampliamente conocido en el entorno de la ciencia de datos.

Para instalar tanto Python como *Jupyter Notebook* vamos a utilizar una distribución, que contiene estos y además muchas otras librerías útiles, llamada *Anaconda* [66]. La instalación se realizará en la carpeta “*/opt*” del sistema. Los comandos necesarios para la instalación se encuentran en el bloque de código 5.2.

Código 5.2: Descarga e instalación de Anaconda.

```
sudo wget https://repo.continuum.io/archive/Anaconda3-5.0.1-Linux-  
        ↪ x86_64.sh  
sudo bash Anaconda3-5.0.1-Linux-x86_64.sh
```

## Creación de usuario y grupo común

Para evitar posibles problemas de permisos durante la ejecución de las tareas, es buena práctica tener un grupo de usuarios donde estén las máquinas incluidas en el clúster.

Por tanto, para abordar este problema, durante la instalación del S.O. en las máquinas del clúster se estableció el mismo nombre de usuario para todas. Además, tras esta se creó un nuevo grupo de usuarios, donde se incluyó a dichos usuarios. El proceso para la creación del grupo y adición del usuario a este se puede encontrar en el fragmento de código 5.3

Código 5.3: Creación de grupo común y adición del usuario del sistema a este.

```
sudo addgroup spark  
sudo adduser --ingroup spark david
```

Una vez realizados estos pasos, las máquinas de la red tendrían acceso a los ficheros de cualquier otra si el grupo tuviese permisos de acceso en dichos archivos. En este caso, todo el conjunto de ficheros utilizados por el sistema *big data* tendrá acceso de lectura y escritura por el grupo de usuarios *spark*, como se explicará en la configuración de *Apache Spark*.

### Instalación de SSH y creación del certificado

El cliente de SSH viene instalado por defecto en Ubuntu y Debian, sin embargo, para que los nodos devuelvan la llamada del maestro y establecer la conexión cuando se inicie el clúster, será necesario que estos dispongan del servidor SSH.

Estos programas también se encuentran en el repositorio oficial de Ubuntu, por lo que la instalación se realiza escribiendo el nombre del paquete en la terminal. Con respecto a los equipos de la universidad, estos están ya instalados para su uso. El comando de consola para instalarlo se encuentra en el fragmento de código 5.4.

Código 5.4: Instalación del cliente y el servidor de SSH.

```
sudo apt install openssh-client  
sudo apt install openssh-server
```

Una vez instalados los programas, el siguiente paso será crear el certificado SSH para cada equipo y, así, facilitar las conexiones entre los nodos. Es decir, una vez creados los certificados SSH de cada máquina, si estos son compartidos entre ellas, los equipos pasarán a estar en la lista de dispositivos seguros y no será necesaria la autentificación con contraseña para establecer la conexión.

El proceso de creación de certificados se puede encontrar en el fragmento de código 5.5 y se realizará en todos los máquinas.

Código 5.5: Creación y autorización del certificado SSH.

```
ssh-keygen -t rsa -P ""  
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

Tras estos pasos, ya tenemos los requisitos completos para iniciar la instalación del clúster.

## Instalación y configuración del modo distribuido

Este apartado se tratará la configuración de *Apache Spark* para la configuración distribuida o multinodo, donde existe una máquina que hará de maestro y  $n$  máquinas que harán de esclavos.

**Asignación de IPs y nombres de sistema en la red.** Este proceso se realizará para simplificar la conexión entre los nodos de los sistemas utilizados, permitiendo con ellos, realizar las conexiones SSH mediante el uso de los nombres asignados, en vez de las direcciones IP locales.

En este caso se realizará modificando el archivo “/etc/hosts” en cada nodo utilizado. Como se tenemos dos equipos el *master* recibirá el nombre de “david-hdp” que es el nombre que tenía anteriormente y el esclavo será *nodo n* donde  $n$  será su número de esclavo en el sistema. La modificación del archivo “hosts” será la misma para todos los nodos, añadiendo las líneas que se encuentran en el fragmento de código 5.6

Código 5.6: Líneas a añadir en el fichero “/etc/hosts” de cada nodo del clúster.

```
192.168.1.15 david-hdp  
192.168.1.17 node1
```

**Compartición de certificados SSH.** Como se indicó en el apartado 5.2.3, la creación de los certificados SSH permitirá que, al compartirse entre los equipos que formen el clúster, no sea necesaria la autentificación con contraseña, simplificando el proceso de conexión entre las máquinas. Además este proceso es requerido tanto por *Apache Spark* como por *Apache Hadoop* para realizar la interoperatividad.

En el caso del entorno doméstico, será el maestro el que necesite los certificados de los esclavos para realizar la conexión, por lo que el proceso seguido es la copia de estos en el nodo maestro. El proceso inverso y la compartición entre esclavos no será necesaria, ya que, es el maestro el que inicia la conexión y porque los esclavos no están conectados entre sí.

Es exclusivamente el maestro el que requiere certificarse en los esclavos para realizar la conexión, por lo que el proceso seguido es la copia de estos en el nodo maestro. El proceso inverso y la adición entre esclavos no será necesaria, ya que, es el maestro el que inicia la conexión y los esclavos no están interconectados.

Los comandos a ejecutar en la terminal del nodo maestro para la obtención de los certificados de los esclavos se puede encontrar en el fragmento de código 5.7.

Código 5.7: Obtención de los certificados SSH de los esclavos por parte del maestro en el clúster.

```
ssh-copy-id -i $HOME/.ssh/id_rsa.pub david@node1
```

Por último, para evitar que el servidor SSH rechace las peticiones de forma automática editaremos el fichero “/etc/ssh/sshd\_config”, la modificación será reflejada en el fragmento de código 5.8.

Código 5.8: Líneas modificadas en el fichero “/etc/ssh/sshd\_config” cada nodo del clúster.

```
PasswordAuthentication No
```

Posteriormente para hacer efectivo el cambio se ejecutará el comando que se puede encontrar en el fragmento de código 5.9 o simplemente reiniciando el equipo.

Código 5.9: Comando de consola para reiniciar el servidor SSH.

```
sudo service sshd restart
```

**Instalación de Apache Spark.** La instalación de *Apache Spark* en este caso se tiene que hacer de forma manual en cada máquina del sistema. Para la realización de este proceso de forma automática en cada nodo se creó un script que se ejecutaría en cada nodo del sistema *big data*. El código de este se puede encontrar en el fragmento de código 5.10.

Código 5.10: Script de instalación de *Apache Spark*.

```
#!/bin/bash
cd /opt
wget http://apache.uvigo.es/spark/spark-2.2.0/spark-2.2.0-bin-hadoop2
→ .7.tgz
tar -xvzf spark-2.2.0-bin-hadoop2.7.tgz
mv spark-2.2.0-bin-hadoop2.7 spark
rm -rf spark-2.2.0-bin-hadoop2.7.tgz
chmod 1777 -R /opt/spark
```

**Configuración de Apache Spark.** El primer paso a realizar es indicar al sistema la ruta de instalación de *Apache Spark* y de Java, modificando el archivo “.bashrc” de cada nodo. Debido a que el nombre de usuario es el mismo en todos los equipos, el fragmento de código a añadir en el fichero es el mismo para todos, siendo este el que se puede encontrar en el código 5.11. Se puede ver el fichero “.bashrc” completo en el anexo C.

Por otro lado, también hay que modificar los archivos de configuración de *Apache Spark* para su correcto funcionamiento los cuales se pueden encontrar en “/opt/spark/conf”. Primero se modificará el archivo “spark-env.sh” en todos los nodos, para establecer que máquina será la maestra y donde se ha instalado java y python. Estas últimas líneas no serían necesarias, dado que debería haberse añadido esta variable de sistema durante la instalación de java.

Código 5.11: Líneas a añadir en el fichero “spark-env.sh” para configurar *Apache Spark* en el modo distribuido.

```
export SPARK_MASTER_IP=david-hdp # Nombre del maestro del cluster
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export HADOOP_CONF_DIR=/opt/hadoop/etc/hadoop

export SPARK_WORKER_INSTANCES=1
export SPARK_DRIVER_MEMORY=2G
export PYSPARK_PYTHON=/opt/anaconda3/bin/python3.6
export PYSPARK_DRIVER_PYTHON=/opt/anaconda3/bin/python3.6
```

Por último, a partir de la versión 2.0 de *Apache Spark* las configuraciones genéricas ya no se realizan en el fichero “spark-env.sh” sino en el fichero “spark-defaults.conf”, en nuestro caso el contenido de dicho fichero es el especificado en el fragmento de código 5.12.

Código 5.12: Contenido del fichero “spark-defaults.conf” de configuración de *Apache Spark*.

spark.master	spark://david-hdp:7077
spark.eventLog.dir	hdfs:///spark-history
spark.eventLog.enabled	true
spark.history.fs.logDirectory	hdfs:///spark-history
spark.history.provider	org.apache.spark.deploy.history.
→ FsHistoryProvider	
spark.history.ui.port	18080
spark.executor.memory	2G
spark.executor.cores	2

Tras realizar esto, el siguiente paso, será establecer que máquinas de la red formarán parte del clúster. Esto se indicará en el archivo “/opt/spark/conf/slaves”, donde se escribirá el nombre de red de los nodos que se utilizarán. En este caso, al ser el maestro también un esclavo se le incluirá en dicho fichero, en un clúster de producción el maestro no sería en ningún caso esclavo, es más, existirían dos maestros para prevenir la caída de uno de ellos. Las líneas a añadir en el fichero se puede encontrar en el fragmento de código 5.13 que solo será necesario escribir en el nodo maestro, que será el que realice las conexiones.

Código 5.13: Líneas a añadir en el fichero “slaves” para establecer las máquinas a utilizar en el clúster.

```
localhost # Es la maquina maestra
node1
```

### 5.2.4. Configuración de *Jupyter Notebook*

Como se ha comentado anteriormente en la sección 2.4.4 se va a utilizar como IDLE *Jupyter Notebook*. Para poder utilizar PySpark en *Jupyter* se ha de crear un nuevo kernel, para ello se hará uso del código 5.14 y se escribirán las líneas reflejadas en el fragmento 5.15.

Código 5.14: Comando para generar el kernel *Jupyter* para PySpark.

```
sudo mkdir -p /opt/anaconda3/share/jupyter/kernels/pyspark
sudo nano /opt/anaconda3/share/jupyter/kernels/pyspark/kernel.json
```

Código 5.15: Contenido del fichero “kernel.json” para la configuración con *Apache Spark*.

```
{
  "display_name": "PySpark",
  "language": "python",
  "argv": [
    "/opt/anaconda3/bin/python3",
    "-m",
    "ipykernel",
    "-f",
    "{connection_file}"
  ],
  "env": {
    "SPARK_HOME": "/opt/spark/",
    "PYTHONPATH": "/opt/spark/python:/opt/spark/python/lib/py4j
      ↪ -0.10.4-src.zip",
    "PYTHONSTARTUP": "/opt/spark/python/pyspark/shell.py"
  }
}
```

La variable asociada al “pythonpath” del fragmento 5.15 puede variar según la versión de *Apache Spark* por lo que sería necesario comprobar si corresponde a la instalada. En este caso, debido a la versión que hemos utilizado la numeración del fichero es la correcta.

### 5.2.5. Instalación de *Apache Hadoop*

Como se ha indicado en el diseño de la arquitectura *big data*, en el apartado 4, para la configuración distribuida del clúster se necesita un mecanismo de replicación de datos para su funcionamiento. Por tanto, como se explicó, se decidió usar *Apache Hadoop*, en concreto su sistema de ficheros distribuido, HDFS, que permitirá la disponibilidad de las trazas en todos los nodos del clúster.

En este apartado se va a proceder a explicar el proceso de instalación y configuración de este sistema. La versión de *Apache Hadoop* utilizada en este proyecto es la versión 2.9.0 precompilada, que está disponible para su descarga desde la página oficial [67].

Al tratarse de un sistema distribuido, al igual que con la instalación de *Apache Spark*, esta deberá realizarse en cada sistema. Para agilizar este proceso, se ha creado un script para que realice la descarga del fichero “.tar.gz”, lo descomprima y lo mueva a la ubicación seleccionada para su instalación. Esta ubicación será similar a la de *Apache Spark*, es decir, será instalado en la ruta “/opt/hadoop/”. El script ejecutado se encuentra en el fragmento de código 5.16.

Código 5.16: Script de instalación de *Apache Hadoop*.

```
#!/bin/bash
wget http://www-us.apache.org/dist/hadoop/common/stable2/hadoop
      ↳ -2.9.0.tar.gz
tar -xzvf hadoop-2.9.0.tar.gz
mv hadoop-2.9.0/ /opt/hadoop
rm -rf hadoop-2.9.0.tar.gz
chmod 1777 -R /opt/hadoop
```

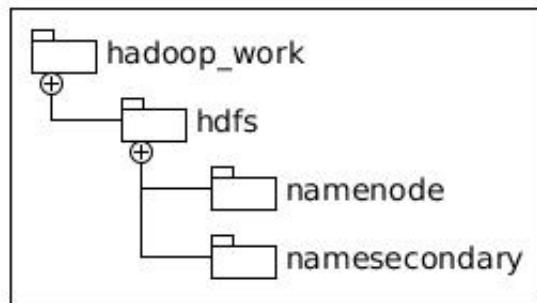
**Estructura de carpetas para *Apache Hadoop*.** Antes de iniciar la configuración de *Apache Hadoop* vamos a crear la estructura de directorios de trabajos de HDFS que configuraremos posteriormente. Como se ha explicado anteriormente en la sección 2.4.2, existen dos tipos de nodo, *datanode* y *namenode*, el primero contiene los datos de replicación y no existe limitación en cuanto a su número, mientras el segundo tipo es un nodo de control o nodo maestro.

Esta diferencia es importante ya que se van a realizar dos estructuras diferentes según el tipo de nodo. Para el nodo maestro vamos a realizar la estructura que aparece reflejada en la figura 5.1, los comandos para su realización se pueden ver en el fragmento de código 5.17.

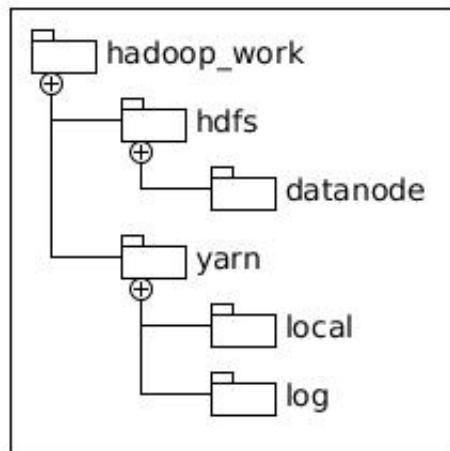
Código 5.17: Código de creación de la estructura de carpetas del *namenode*.

```
sudo mkdir -p /opt/hadoop_work/hdfs/namenode
sudo mkdir -p /opt/hadoop_work/hdfs/namesecondary
```

Para los nodos esclavos se va a realizar la estructura de carpetas reflejada en la figura 5.2. Para la realización de este proceso de forma automática en cada nodo se creó un

Figura 5.1: Estructura de carpetas para operativa del *namenode*.

script que se ejecutaría en cada nodo del sistema *big data*. El código de este se puede encontrar en el fragmento de código 5.18.

Figura 5.2: Estructura de carpetas para operativa del *datanode*.Código 5.18: Código de creación de la estructura de carpetas del *datanode*.

```

#!/bin/bash
mkdir -p /opt/hadoop_work/hdfs/datanode
mkdir -p /opt/hadoop_work/yarn/local
mkdir -p /opt/hadoop_work/yarn/log
chown david:david -R /opt/hadoop_work/
  
```

**Configuración de *Apache Hadoop*.** Tras la descarga e instalación de *Apache Hadoop*, para su funcionamiento, será necesario modificar diferentes archivos de configuración. Lo primero será modificar los archivos de sistema para establecer correctamente las rutas de instalación y, posteriormente, los ficheros propios del Framework.

Como en el caso de la instalación de *Apache Spark* el fichero de sistema a modificar es “`bashrc`”, al que habrá que añadirle unas líneas indicando diferentes rutas de instalación de los módulos de *Apache Hadoop* para su funcionamiento. Las líneas a añadir se pueden encontrar en el fragmento de código 5.19.

Código 5.19: Líneas a añadir a “`.bashrc`” para el funcionamiento de *Apache Hadoop*.

```
export HADOOP_HOME=/opt/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export CLASSPATH=$CLASSPATH:$HADOOP_HOME/lib/*
```

Posteriormente, se modificarán los diferentes ficheros de configuración de *Apache Hadoop* para hacerlo funcionar en el clúster doméstico. Todos estos archivos residen en la ruta “`/opt/etc/hadoop`”. Los ficheros a modificar serán los siguientes:

- **`hadoop-env.sh`**: Donde se indicará la ruta de instalación de Java en la máquina.

Código 5.20: Línea a añadir a “`hadoop-env.sh`”.

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

- **`core-site.xml`**

Código 5.21: Contenido del “`core-site.xml`”.

```
<configuration>
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://david-hdp:9000</value>
</property>

<property>
    <name>ipc.maximum.data.length</name>
    <value>134217728</value>
</property>
<property>
    <name>io.file.buffer.size</name>
```

```

    <value>131072</value>
</property>
</configuration>

```

#### ■ hdfs-site.xml

Código 5.22: Contenido del “hdfs-site.xml” de el nodo maestro.

```

<configuration>
<property>
<name>dfs . replication</name>
<value>2</value>
<description>Replication factor , 1 no copy , 2 two copies each
    ↳ fragment</description>
</property>
<property>
<name>dfs . namenode . name . dir</name>
<value>file : / opt / hadoop _ work / hdfs / namenode</value>
<description>Carpeta correspondiente al NameNode</description>
    ↳ >
</property>
<property>
<name>dfs . datanode . data . dir</name>
<value>file : / opt / hadoop _ work / hdfs / datanode</value>
<description>Carpeta correspondiente al DataNode</description>
    ↳ >
</property>
<property>
<name>dfs . namenode . checkpoint . dir</name>
<value>file : / opt / hadoop _ work / hdfs / namesecondary</value>
<description>Carpeta correspondiente al NameSecondary</
    ↳ description>
</property>
<property>
<name>dfs . block . size</name>
<value>134217728</value>
<description>Tamanno maximo de bloque de archivo (128Mb)</
    ↳ description>
</property>
<property>
<name>dfs . webhdfs . enabled</name>
<value>true</value>
</property>
</configuration>

```

Código 5.23: Contenido del “hdfs-site.xml” de los nodos esclavos.

```
<configuration>
  <property>
    <name>dfs . replication</name>
    <value>2</value>
    <description>Replication factor , 1 no copy , 2 two copies each
      ↳ fragment</description>
  </property>
  <property>
    <name>dfs . datanode . data . dir</name>
    <value>file : / opt / hadoop _ work / hdfs / datanode</value>
    <description>Carpeta correspondiente al DataNode</description>
      ↳ >
  </property>
  <property>
    <name>dfs . namenode . checkpoint . dir</name>
    <value>file : / opt / hadoop _ work / hdfs / namesecondary</value>
    <description>Carpeta correspondiente al NameSecondary</
      ↳ description>
  </property>
  <property>
    <name>dfs . block . size</name>
    <value>134217728</value>
    <description>Tamanno maximo de bloque de archivo (128Mb)</
      ↳ description>
  </property>
</configuration>
```

- **mapred-site.xml**

Código 5.24: Contenido del “mapred-site.xml”.

```
<configuration>
  <property>
    <name>mapreduce . framework . name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>mapreduce . jobhistory . address</name>
    <value>david-hdp:10020</value>
    <description>Direccion de acceso al historial</description>
  </property>
  <property>
    <name>mapreduce . jobhistory . webapp . address</name>
    <value>david-hdp:19888</value>
    <description>Acceso a la app web</description>
  </property>
  <property>
    <name>jobtracker . thrift . address</name>
    <value>0.0.0.0:9290</value>
  </property>
```

```

<description>Block size</description>
</property>
<property>
  <name>mapred.jobtracker.plugins</name>
  <value>org.apache.hadoop.thriftfs.ThriftJobTrackerPlugin</value>
<description>Comma-separated list of jobtracker plug-ins to
  <!-- be activated --></description>
</property>
</configuration>

```

- **yarn-site.xml**

Código 5.25: Contenido del “yarn-site.xml”.

```

<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>david-hdp</value>
  </property>
  <property>
    <name>yarn.resourcemanager.bind-host</name>
    <value>0.0.0.0</value>
  </property>
  <property>
    <name>yarn.nodemanager.bind-host</name>
    <value>0.0.0.0</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
    <name>yarn.log-aggregation-enable</name>
    <value>true</value>
  </property>
  <property>
    <name>yarn.nodemanager.local-dirs</name>
    <value>file:/opt/hadoop_work/yarn/local</value>
  </property>
  <property>
    <name>yarn.nodemanager.log-dirs</name>
    <value>file:/opt/hadoop_work/yarn/logs</value>
  </property>
  <property>

```

```
<name>yarn . nodemanager . remote-app-log-dir</name>
<value>hdfs://david-hdp:9000/var/log/hadoop-yarn/apps</value>
<description>logs en hdfs</description>
</property>
</configuration>
```

- **master:** Máquina en la que resida el *namenode*.

Código 5.26: Línea a añadir a “master”.

```
david-hdp
```

- **slaves:** Máquinas en la que resida el *datanode*, es decir, donde se repliquen los datos.

Código 5.27: Líneas a añadir a “slaves”.

```
david-hdp
node1
```

### 5.2.6. Instalación de *Apache Flume*

Al tratarse de un sistema distribuido, al igual que con la instalación de *Apache Spark*, esta deberá realizarse en cada sistema. Como en los casos anteriores se utilizará una versión precompliada de *Apache Flume* que puede encontrarse en la página de descarga oficial [68]

Para agilizar este proceso, se ha creado un script para que realice la descarga del fichero “.tar.gz”, lo descomprima y lo mueva a la ubicación seleccionada para su instalación. Esta ubicación será similar a la de *Apache Spark*, es decir, será instalado en la ruta “/opt/flume/”. El script ejecutado se encuentra en el fragmento de código 5.28.

Código 5.28: Script de instalación de *Apache Flume*.

```
#!/bin/bash

wget http://apache.rediris.es/flume/1.8.0/apache-flume-1.8.0-bin.tar.gz
tar -xzvf apache-flume-1.8.0-bin.tar.gz
mv apache-flume-1.8.0-bin/ /opt/flume
rm -rf apache-flume-1.8.0-bin.tar.gz
chmod 1777 -R /opt/flume
```

**Configuración de *Apache Flume*.** Además se deberá editar el fichero de configuración de *Apache Flume* para indicarle la localización de la instalación de java, para ello se modificará el fichero en la ruta “/opt/flume/conf/flume-env.sh” como se encuentra en el fragmento de código 5.29.

Código 5.29: Línea a añadir a “flume-env.sh”.

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export JAVA_OPTS="-Xms2000m -Xmx10000m -Dcom.sun.management.jmxremote
↪ "
```

## 5.3. Arranque y parada del clúster

Con todos los pasos anteriores realizados correctamente necesitamos iniciar el clúster, para ello se realizará con una serie de comandos iniciando cada parte de forma independiente. Para automatizar este proceso se han creado dos scripts de arranque y parada, se puede ver el código de estos scripts en las figuras 5.30 para el inicio y en la figura 5.31 la parada del mismo.

Código 5.30: Script de inicio del clúster implementado.

```
#!/bin/bash
printf "\n\nLaunching Hadoop"
printf "\n-----\n"
```

```

printf "\nStarting DFS system:\n"
$HADOOP_HOME/sbin/start-dfs.sh
printf "\nStarting yarn daemons"
$HADOOP_HOME/sbin/start-yarn.sh
printf "\nStarting mapreduce history"
$HADOOP_HOME/sbin/mr-jobhistory-daemon.sh start historyserver

printf "\n\nLaunching Apache Spark"
printf "\n-----\n"
$SPARK_HOME/sbin/start-all.sh
printf "\n\nLaunching Apache Spark History Server"
$SPARK_HOME/sbin/start-history-server.sh

```

Código 5.31: Script de parada del clúster implementado.

```

#!/bin/bash
printf "\n\nStoping Hadoop"
printf "\n-----\n"
printf "\nStoping DFS system"
$HADOOP_HOME/sbin/stop-dfs.sh
printf "\nStoping yarn daemons"
$HADOOP_HOME/sbin/stop-yarn.sh
printf "\nStoping mapreduce history server"
$HADOOP_HOME/sbin/mr-jobhistory-daemon.sh stop historyserver

printf "\n\nStoping Apache Spark"
printf "\n-----\n"
$SPARK_HOME/sbin/stop-all.sh
printf "\n\nStoping Apache Spark History Server"
$SPARK_HOME/sbin/stop-history-server.sh

```

Una vez iniciado el clúster se deben generar las carpetas y ficheros que formarán parte del mismo, esto lo hace *Apache Hadoop* de forma automática utilizando el comando 5.32 y posteriormente se debe crear la carpeta del usuario que se realiza con el segundo comando del fragmento 5.32

Código 5.32: Comandos requeridos en la primera ejecución del clúster.

```

hadoop namenode -format
hdfs dfs -mkdir -p /user/david

```

# 6

## Stack Overflow

### 6.1. Introducción

Empezó en 2008 por *Joel Spolsky* con su blog “*Coding Horror*” con la idea de crear un sitio de preguntas y respuestas. Un año más tarde con la ayuda de *Jeff Atwood* apareció el sitio conocido como *Stack Overflow* [60] donde los especialistas de Information Technology (IT) pueden plantear sus dudas y ser respondidos por otros profesionales del sector.

Esta plataforma ha ido evolucionando rápidamente desde sus inicios hasta convertirse en una web de referencia, como es actualmente, para los profesionales del sector.

### 6.2. Aprovisionamiento de datos

La empresa propietaria de la web *Stack Overflow* [60], *Stack Exchange* [69], siguiendo la cultura de *open data* [5] hace públicos los datos relacionados con sus comunidades entre las que se encuentra la que vamos a analizar para este proyecto.

Estos datos se encuentran publicados en la web de *Stack Exchange Data Dump* [70], distribuidos libremente bajo licencia *Creative Commons* [71]. Para este proyecto vamos a descargar exclusivamente los ficheros relacionados, que son los que se pueden ver en la figura 6.1

Código 6.1: Ficheros públicos sobre *Stack Overflow* en la web de *Data Dump*.

stackoverflow.com-Badges.7z	180.9M
stackoverflow.com-Comments.7z	3.4G
stackoverflow.com-PostHistory.7z	19.7G
stackoverflow.com-PostLinks.7z	62.8M
stackoverflow.com-Posts.7z	11.3G
stackoverflow.com-Tags.7z	709.2K
stackoverflow.com-Users.7z	328.0M
stackoverflow.com-Votes.7z	826.8M

Estos archivos vienen comprimidos en 7z [72], por lo tanto, el primer paso es descomprimirlos, dicho proceso se completa en un tiempo total de 4 horas. Los ficheros descomprimidos están en formato XML, al abrirlos comprobamos que la estructura de estos ficheros consiste en un elemento por etiqueta. Dado *Apache Spark* no permite de forma nativa la lectura de este tipo de ficheros, se deberá crear un script que realice la conversión de formato.

### 6.3. Procesado de datos

Los datos vienen en XML [73], un formato no soportado de forma nativa en *Apache Spark* por lo que el primer paso es convertir estos datos en uno soportado como es JavaScript Object Notation (JSON). Se ha creado un script llamado “xml2json.py” que hará esta labor.

La primera parte del script son los *imports* necesarios que se pueden apreciar en el fragmento de código 6.2, donde se pueden ver tres principales:

- **OS:** Será utilizado para comprobar la existencia de los ficheros de entrada y salida.
- **json:** Utilizado para generar las cadenas de texto resultantes del proceso.
- **lxml:** Librería que realizara el *parseo* para la extracción de los datos necesarios del XML.

Código 6.2: *Imports* del script “xml2json.py” de procesado de XML.

```

1 #!/usr/bin/env python3.6
2
3 """
4 Este programa solo extrae los atributos de cada elemento,
5 y ademas da por hecho que cada elemento esta alojado en una unica
6 → linea
7 Usage:
8 ./xml2json.py inputFile outputFile
9 ./xml2json.py /opt/Posts.xml /opt/Posts.json
"""

```

```

10 import sys
11 import json
12 import tqdm # Solo para ver como avanza
13 from time import time
14 from hurry filesize import size
15 from os import path, remove
16 from lxml import etree

```

Se realiza la comprobación sobre la existencia del fichero de entrada, en caso afirmativo el script continua y en caso contrario se informa al usuario y termina la ejecución del mismo. Posteriormente se comprueba la existencia del fichero de salido, en caso de que este exista se le informa al usuario preguntándole si desea sobreescibirlo, ejecutando las acciones necesarias según la decisión de este. Esta parte se puede ver en el fragmento de código 6.3.

Código 6.3: Fragmento de código del script “xml2json.py” de comprobación del estado de los ficheros *in/out*.

```

24 # Se comprueba si el fichero de entrada existe
25 if path.isfile(_input) == False:
26     print('El fichero no existe')
27     sys.exit() # Si no existe se interrumpe la ejecucion
28
29 # Se comprueba si el fichero de salida existe
30 # Si existe, se le pregunta al usuario si desea sobreescibirlo
31 if path.isfile(_output):
32     isOverwrite = input('El fichero {} ya existe, Desea sobreescibirlo
33     ↪ ?[Y/n]: '.format(_output))
34     if isOverwrite.lower() == 'y':
35         remove(_output)

```

En el fragmento de código 6.4 se declara una función encargada de la persistencia del JSON. Aunque la la librería “lxml” retorna un diccionario, este no es de tipo básico de python y por tanto la librería “json” no es capaz de procesarlo, para solucionar este inconveniente se recorre la lista de pares clave-valor y se almacenan consecutivamente en un diccionario de tipo básico. Por último se abre el fichero de destino en modo “append” y se guardan los JSON resultantes del proceso.

Código 6.4: Fragmento de código del script “xml2json.py” encargado de la persistencia de resultados.

```

36 # Metodo para guardar los atributos en json
37 def save_json(atributos, count):
38     # Se convierten los atributos (etree.Attrib) a tipo diccionario
39     dict_json = dict()
40     for k, v in atributos.items():
41         dict_json[k] = v
42     # Se convierte el diccionario en un .json y se le agrega un salto
43     # de linea final
43     str_json = json.dumps(dict_json) + '\n'

```

```

44 # Se abre el fichero en modo 'append' y se escribe el .json
45 with open(_output, 'a') as f:
46     f.write(str_json)

```

La función principal (fragmento de código 6.5 del script) consiste en abrir el fichero de entrada en modo lectura y comenzar a leer línea a línea. Cada línea es procesada y guardada antes de continuar con la siguiente.

Cada línea leída es convertida en un nodo de tipo árbol por la librería “lxml” y de cada nodo se extraen sus atributos. Por si dicho nodo perteneciese a una etiqueta que no deseemos, como es la etiqueta de definición, se comprueba que esta posea el atributo “Id” entre ellos; si todo es correcto se llama al método “*save\_json*” para su almacenamiento. La función principal del script corresponde al fragmento de código 6.5.

Código 6.5: Fragmento de código del script “xml2json.py” encargado de la persistencia de resultados.

```

48 # Se abre el fichero de entrada
49 with open(_input, 'r') as f:
50     #Creamos un progress bar donde el total de proceso es el tamaño del
      ↵ fichero
51     progress = tqdm.tqdm(unit='bytes', leave=False, total=path.getsize(
      ↵ _input), \
52     bar_format='{l_bar}{bar}| {percentage:3.0f}% [{elapsed}<{remaining
      ↵ }, {rate_fmt}{postfix}]')
53     # Se va leyendo linea a linea
54     for line in f:
55         # Se va actualizando la barra con el tamaño en bytes de la linea
            ↵ leida.
56         progress.update(sys.getsizeof(line))
57     try:
58         node = etree.fromstring(line)    # Se convierte el texto en un
            ↵ arbol
59         atributos = node.attrib        # Se obtienen los atributos
60         if atributos.has_key('Id'):
61             save_json(atributos, count)
62             count+=1
63     except Exception as f: # Algunas lineas como la descripcion de
            ↵ formato dan error
64         print(line)
65     progress.close()

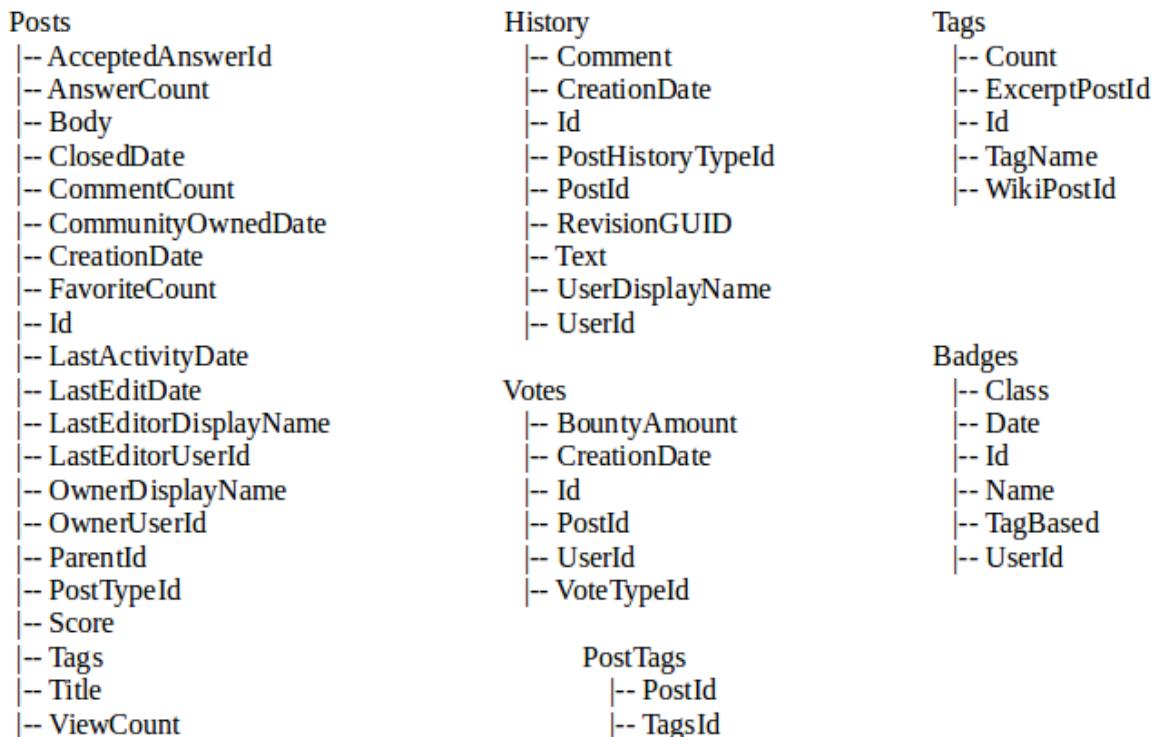
```

## 6.4. Análisis de datos

La fase de análisis de datos consiste en documentar el valor de los datos, asegurar su calidad y veracidad. Para este estadio no es necesario usar el total de los datos, pues ello supondría mucho tiempo de procesado, a tal efecto se coge una muestra pequeña, pero significativa, de cada origen. En este caso los ficheros mencionados al inicio de la sección.

El primer paso realizado es la obtención de un esquema de estructura de datos de cada fichero, la cual se puede apreciar en la figura 6.1

Figura 6.1: Esquema completo de *Stack Overflow*.



Llegados a este punto deberemos establecer que datos serán los que utilicemos y qué valor poseen estos. El primer problema al que nos enfrentemos será definir el término “popularidad”; para ello nos encontramos con varias formas de resolverlo: qué pregunta recibe más votos, cuál recibe más *badges*, cuál es el más visitado, sobre cuál se crean más *posts*, en cuál comenta más gente, etc...

Realizando un pequeño análisis descubrimos varias cosas:

- No tenemos información sobre le número de visitas.
- Los tipos de votos son demasiado dispares para definir un valor exacto de cada en relación al *tag*.

- Los *badges* varían a lo largo del tiempo, asignándose más a principios y finales de año, debido a la naturaleza de los mismos.
- Los comentarios no indican el interés absoluto de los profesionales del sector, únicamente la capacidad, conocimiento o participación de los usuarios activos.
- Los *posts* se crean cuando los usuarios tienen una duda no resuelta anteriormente o cuando la duda no se resuelve fácilmente.

Este sencillo análisis deja solamente una opción dirigida directamente a resolver los *insights* establecidos, los *posts*.

Una vez que hemos decidido que serán los *posts* quienes no van a proporcionar los datos necesarios para resolver la popularidad, continuaremos analizando cuáles de estos datos serán los más relevantes.

Los datos que a *priori* consideramos más relevantes son el *tag*, el *creationDate* y el *score*. Además, para obtener el *tag* de cada *post*, deberemos cruzarlos con la tabla de *tags* a través de *PostTags*, que nos permitirán conocer que lenguaje o tecnología pertenece cada *posts*.

Realizaremos una primera consulta a través del módulo de *Spark SQL* y almacenaremos los resultados. En el fragmento de código 6.10 podemos ver como se realiza una consulta a través de este módulo. Pero de momento la consulta SQL usada para la relación se puede observar en fragmento de código 6.6.

Código 6.6: Consulta SQL para la extracción de la fecha, *tag* y *score* de las consultas en *Stack Overflow*.

```
SELECT t.tag ,
       date_format(creation_date , '%%-%m') AS month ,
       COUNT(*) AS count ,
       SUM(q.score) AS score
  FROM posts p
 INNER JOIN tags t on p.id = t.id
 WHERE score > 0
 GROUP BY month , t.tag ;
```

La consulta del código 6.6 nos devolverá la puntuación que le han dado los usuarios ordenada por mes y lenguaje, almacenaremos los datos, para análisis futuros. A partir de estos datos, podemos realizar la búsqueda de los lenguajes más populares.

Establecemos una búsqueda de los lenguajes atendiendo al número de apariciones de los lenguajes en el último año, a través de *Spark SQL*. Estas consultas son gestionadas por el *Spark Core*. Para empezar a trabajar con *Apache Spark* debemos declarar algunas variables importantes como son el *SparkContext* o el *SparkSession*.

En primer lugar, realizaremos los *imports* y la inicialización tanto de *SparkContext*, como del *SparkSession* tal y como se puede apreciar en el fragmento de código 6.7, esto será común a todos los scripts de *Apache Spark* por lo que se evitará repetir código a lo largo del proyecto.

Código 6.7: Imports y configuración de una sesión de Apache Spark.

```
from pyspark import SparkConf, SparkContext
from pyspark.sql.functions import col, asc, desc
from pyspark.sql.types import StructType, StructField, StringType,
                             IntegerType

conf = SparkConf().setAppName("Jupyter: Stack Posts")
conf = conf.setMaster("spark://david-hdp:7077")
sc = SparkContext(conf=conf)
spark = SparkSession(sc)
```

Dado que los resultados del código 6.6 se han exportado a un CSV sin cabecera, deberemos crear un *schema* de manera manual, para ello *Apache Spark* proporciona unos tipos de datos propios para relacionar los *schemas* con su origen de datos. Como se puede observar en el código 6.8, se ha creado un *schema* de cuatro campos.

Código 6.8: Código de declaración del *schema* para lectura del CSV.

```
schema = StructType([
    StructField("tag", StringType(), True),
    StructField("fecha", StringType(), True),
    StructField("count", IntegerType(), True),
    StructField("score", IntegerType(), True),
    StructField("answers", IntegerType(), True)])
```

El siguiente paso será importar el fichero que contiene los datos indicandole el *schema* del fragmento de código 6.8 y lo guardamos en una variable llamada “df” de la forma que aparecen el código 6.9 y posteriormente se visualiza una muestra de los datos para comprobar como están almacenados los datos. Esta muestra se puede ver en la tabla 6.1 donde vemos los 6 primeros elementos del *dataframe*.

Código 6.9: Carga de un fichero CSV con un esquema predefinido con el módulo *Spark SQL*.

```
df = spark.read.csv('stof/clean/mysql_clean.csv', schema=schema)
```

Tabla 6.1: *DataFrame* de muestra de datos de *Stack Overflow*.

tag	fecha	count	score	answers
.net	2010-03	1073	6385	2768
.net	2011-02	1036	4408	2101
.net	2011-03	1030	5493	2192
.net	2010-08	1018	5003	2443
.net	2011-05	1004	4015	1906
.net	2011-04	987	3718	1977

Una vez que tengamos los datos cargados, ya se puede empezar a trabajar con ellos. El primer hito a resolver son los lenguajes más populares. Para ello vamos a realizar una consulta SQL que nos devuelva los diez primeros lenguajes y/o tecnologías más populares

del último año. El primer paso será generar una tabla temporal para poder operar con SQL, dicha función la cumple la primera línea del fragmento de código 6.10.

Código 6.10: Código de generación de la tabla temporal y la ejecución de una consulta SQL.

```
# Nueva tabla temporal
df.createOrReplaceTempView("temp_table")

# Ejecucion de consulta SQL
top10 = spark.sql("\
    SELECT tag , \
        SUM(count) AS count , \
        SUM(score) AS score \
    FROM temp_table \
    WHERE \
        fecha >= "2016-12" \
    GROUP BY tag \
    ORDER BY count DESC \
    LIMIT 10")
```

El resultado de esta consulta se muestra en la tabla 6.2, esta es la lista de las diez tecnologías más populares o, dicho de otra manera, sobre qué tecnologías están preguntando los profesionales del sector de IT.

Tabla 6.2: Tabla con los 10 lenguajes más populares de *Stack Overflow*.

<b>tag</b>	<b>count</b>	<b>score</b>
javascript	482970	876320
c#	436792	810352
java	354105	644255
android	352392	720040
python	267364	498320
c++	232816	677424
php	213584	328408
ios	181032	383838
c	91936	219344
html	91788	163830

Con estos datos se puede iniciar el aprovisionamiento de datos en twitter utilizando estos resultados como *keywords*.

# 7

## Twitter

### 7.1. Introducción

Twitter [63] es una plataforma de comunicación bidireccional con naturaleza de red social (porque permite elegir con quien te relacionas). Cuyos mensajes están limitados a 280 caracteres. Aquí los usuarios publican sus opiniones personales o anuncios que consideran importantes. Proporciona una API que permite obtener estos mensajes (*tweets*) en tiempo real.

La idea nació en 1992, en la mente de *Jack Dorsey*, inspirada en el *software* de rastreo de los taxistas. Finalmente el proyecto se concretó en 2006 como un servicio interno entre los empleados de *Odeo* y fue lanzado exitosamente en 2007 gracias a la popularidad de los *smartphones*.

Hoy día Twitter es uno de los sistemas de comunicación más utilizados, no solo para información intrascendente, social, sino como herramienta de comunicación entre profesionales.

## 7.2. Aprovisionamiento de datos

Para realizar el aprovisionamiento de datos en Twitter, el primer movimiento consiste en informarse sobre la API de Twitter [74], rápidamente se puede observar que ofrece tres niveles de acceso: al archivo completo, a los datos de la última semana y a los datos en tiempo real. Debido a que los dos primeros niveles requieren reuniones, pago y aprobación por parte de la empresa, vamos a utilizar los datos en tiempo real.

La captación de *tweets* en tiempo real y su ingestión en HDFS se va a realizar a través de *Apache Flume*, para ello se va a configurar un agente que nos permita obtenerlos y almacenarlos en HDFS.

### 7.2.1. Agente de *Apache Flume*

Para este caso específico *Cloudera* [26] proporciona una librería de java que permite obtener los *tweets* en formato JSON, lo que simplificará enormemente el análisis y el filtrado de datos.

El agente de *Apache Flume* está completo en el código 7.1, se definen el origen (*source*), el canal (*channel*) y el sumidero de datos (*sink*), la primera parte de la configuración corresponde a Twitter, donde se deberán indicar las cuatro claves que proporciona el API del mismo y los *keywords* que se desean obtener, los cuales han sido obtenidos anteriormente en el análisis de *Stack Overflow* y aparecen en la tabla 6.2.

La siguiente parte del código corresponde al sumidero o *sink*, en inglés, donde se le indica que los datos se mantienen en memoria y que se almacenan en HDFS, también se especifican los datos necesarios para la conexión con el clúster de *Apache Hadoop*; así como datos más específicos dependientes de la configuración del clúster.

Código 7.1: Fichero de configuración del agente de *Apache Flume* para *Twitter*.

```
TwitterAgent.sources = Twitter
TwitterAgent.channels = MemChannel
TwitterAgent.sinks = HDFS

TwitterAgent.sources.Twitter.type = com.cloudera.flume.source.
    ↪ TwitterSource
TwitterAgent.sources.Twitter.channels = MemChannel
TwitterAgent.sources.Twitter.consumerKey = <ConsumerKey>
TwitterAgent.sources.Twitter.consumerSecret = <ConsumerSecret>
TwitterAgent.sources.Twitter.accessToken = <AccessToken>
TwitterAgent.sources.Twitter.accessTokenSecret = <AccessTokenSecret>
TwitterAgent.sources.Twitter.keywords = javascript, androiddev,
    ↪ python, c++, php, iosdev, html, kotlin, php, typescript,
    ↪ powershell, git, github, groovy, maven, cplusplus, springboot,
    ↪ haskell, androidstudio, intellij, netbeans, oracle, mysql,
    ↪ cprogramming, c#
```

```

TwitterAgent.sinks.HDFS.channel = MemChannel
TwitterAgent.sinks.HDFS.type = hdfs
TwitterAgent.sinks.HDFS.hdfs.path = hdfs://david-hdp:9000/user/david/
    ↳ flume/tweets/%Y-%m-%d/
TwitterAgent.sinks.HDFS.hdfs.fileType = DataStream
TwitterAgent.sinks.HDFS.hdfs.writeFormat = Text
TwitterAgent.sinks.HDFS.hdfs.batchSize = 100
TwitterAgent.sinks.HDFS.hdfs.rollSize = 100000
TwitterAgent.sinks.HDFS.hdfs.rollCount = 10000

TwitterAgent.channels.MemChannel.type = memory
TwitterAgent.channels.MemChannel.capacity = 10000
TwitterAgent.channels.MemChannel.transactionCapacity = 100

```

**Lanzamiento del agente.** Para el inicio del agente de *Apache Flume* se utiliza el comando del código 7.2 y se deja funcionar en segundo plano para que se vayan almacenando una cantidad significativa de *tweets*.

Código 7.2: Comando de lanzamiento del agente de *Apache Flume* para *Twitter*.

```

flume-ng agent --conf /opt/flume/conf --conf-file /home/david/scripts
    ↳ /Flume-TwitterAgent.conf

```

### 7.3. Análisis de datos

Para el análisis de datos se configurará una sesión de *Apache Spark* como se ha indicado en el código 6.7 y cargar los datos del aprovisionamiento de Twitter, la primera parte consiste en identificar el *schema* para visualizar la estructura de los datos proporcionados por Twitter. La estructura completa está compuesta por un total de 1241 campos, puede verse en el apéndice A, aquí se muestra un pequeño extracto en el fragmento de código 7.3.

Código 7.3: Extracto del esquema de un *tweet*.

```

|-- created_at: string (nullable = true)
|-- geo: struct (nullable = true)
|   |-- coordinates: array (nullable = true)
|       |-- element: double (containsNull = true)
|   |-- type: string (nullable = true)
|-- lang: string (nullable = true)
|-- place: struct (nullable = true)
|   |-- country: string (nullable = true)
|   |-- country_code: string (nullable = true)
|   |-- full_name: string (nullable = true)
|   |-- id: string (nullable = true)

```

```
|     |-- name: string (nullable = true)
|     |-- place_type: string (nullable = true)
|     |-- url: string (nullable = true)
|-- text: string (nullable = true)
```

Para resolver algunos de los *insights* planteados en la sección 4.3, en este caso aprovecharemos los datos que proporciona Twitter sobre el idioma en el que está escrito un *tweet*, así como la ubicación desde la que se emite el mismo.

Tras un periodo de análisis de los campos existentes se descubren varios hechos:

- Existe un campo llamado “*geo*” el cual corresponde a un campo antiguo (*deprecated*), el cual ya no debería estar en uso, por tanto se descarta para precisar la ubicación.
- Una estructura llamada “*place*” que proporciona varios campos referentes a la ubicación del usuario, cuyo dato se rige por la International Organization for Standardization (ISO) 3166-1 alpha-2 [75].
- “*created\_at*” proporciona la fecha de creación del *tweet* en cuestión, aunque para este análisis no resultará relevante.
- El campo “*lang*” indica el idioma en el que está escrito el *tweet*, bajo la normativa de estandarización ISO 639-1:2002 [76].

Con esto en mente vamos a realizar un script en *Apache Spark* que nos permita extraer estos datos y almacenarlos. En un clúster más grande y para una empresa *data driven* estos datos se filtrarían de una forma menor para futuros análisis, es más, prácticamente no recibirían ninguna transformación.

## 7.4. Procesado de datos

### 7.4.1. Script Apache Spark

Como en casos anteriores, se declaran las variables necesarias y se inicia una sesión de *Apache Spark* y se cargan los datos (ver código 6.7). Por motivos de espacio en el clúster se filtrará periódicamente y se almacenará el resultado en HDFS.

Código 7.4: Consulta SQL para la extracción de datos de Twitter.

```
SELECT
    lang ,
    place.country_code ,
    text
FROM
    tweets
WHERE
    place IS NOT NULL;
```

Se realiza el filtrado con la consulta SQL que aparece en el código 7.4 de la forma ya mostrada en el código 6.10. Esta consulta extraerá el país, el lenguaje que se menciona en el *tweet* y el idioma en el que está escrito. El segundo filtro comprueba que aparezca alguno mencionado, y esto es debido a que la API de Twitter no filtra las palabras por contenido literal, sino por relación.

Por último se almacenan los resultados en HDFS para su posterior uso, este resultado se graba en una carpeta con múltiples archivos. Para simplificar el contenido y por utilizarlo en la generación de un mapa posterior, se creará un CSV, pero no es posible añadirle cabecera al fichero, por tanto para resolver esto y automatizar el proceso se creará un pequeño script *bash*.

### 7.4.2. Script Bash para normalizar resultados

El script tiene la función principal de unir todos los ficheros generados por el proceso anterior y darle una cabecera al fichero CSV.

Este contiene muchos comandos que están explicados en el código 7.5, además se realizan dos extracciones, una con los ficheros que contienen datos de geolocalización y otra sin ellos. No existe variación en los comandos a excepción de la cabecera asignada y la estructura de carpetas.

Código 7.5: Comandos utilizados durante el proceso de unificación de datos.

```
1 #!/bin/bash
2 # Comprueba si existe el fichero (0) o si no existe (1)
3 check_nongeo="hdfs dfs -test -e twitter/single/nongeo.tweets.csv"
4 # Crea un csv con todos los datos de la carpeta twitter/nongeo/*
```

```

5  create_nongeo="hdfs dfs -getmerge twitter/nongeo/* nongeo.csv"
6  # Agrega cada linea del fichero nongeo.csv al fichero nongeo.tweets .
7  #   ↪ csv
8  merge_nongeo_local="cat nongeo.csv >> nongeo.tweets.csv"
9  # Suma el fichero 'nongeo.csv' del local al 'twitter/single/nongeo .
10 #   ↪ tweets.csv' del hdfs
11 merge_nongeo_dfs="hdfs dfs -appendToFile nongeo.csv twitter/single/
12 # Sube el fichero a hdfs
13 upload_nongeo="hdfs dfs -put nongeo.tweets.csv twitter/single/"

```

La segunda fase de este script consiste en comprobar si este proceso se ha realizado anteriormente y en caso afirmativo ya no sería necesario construir una cabecera, tan solo sería necesario agregar los nuevos registros al fichero existente. Dado que este proceso requiere un tiempo se han agregado varias salidas por pantalla para mantener al usuario informado del desarrollo del script. Esta segunda fase se puede ver en el fragmento de código 7.6.

Código 7.6: Código bash de evaluación.

```

40 echo "Evaluando fichero geo"
41 if [ '$exist_geo' == '1' ]; then
42   echo "No existe fichero ../geo.tweets.csv"
43   echo "Creando cabecera de csv"
44   echo 'lang, country, text' > geo.tweets.csv
45   echo "Exportando datos del hdfs"
46   eval $create_geo
47   echo "Creando fichero local con cabecera"
48   eval $merge_geo_local
49   echo "Subiendo fichero ../geo.tweets.csv"
50   eval $upload_geo
51   echo "fichero ../geo.tweets.csv subido"
52   echo "Eliminando ficheros geo auxiliares . . ."
53   eval $remove_geo_local
54   eval $remove_geo_tweets
55   echo "Ficheros auxiliares geo, eliminados!"
56 else
57   echo "Existe el fichero ../geo.tweets.csv"
58   echo "Exportando datos del hdfs"
59   eval $create_geo
60   echo "Uniendo fichero local con hdfs"
61   eval $merge_geo_dfs
62   echo "Eliminando fichero geo auxiliar . . ."
63   eval $remove_geo_local
64   echo "Fichero geo auxiliar, eliminado!"
65 fi

```

# 8

## GitHub

### 8.1. Introducción

GitHub es un repositorio de versiones basado en *Git* [61], el cual siguiendo la filosofía de *open data* [5] proporciona una API desde la que se pueden descargar los eventos almacenados desde 2015 a través de *GitHub Archive* [62].

Nacido en 2008 en una oficina del Valle de San Francisco (como tantos otros proyectos), GitHub tiene hoy más de 9 millones de usuarios registrados, más de 200 millones de visitas al mes y ha sido valorado en más de 2000 millones de dólares (por lo que está en la Unicorn List [77] de Fortune). Utilizan GitHub para gestionar y almacenar su código desde Google a la Casa Blanca pasando por Facebook o incluso el Ayuntamiento de Madrid [78].

## 8.2. Análisis de datos

Para el análisis de datos se configurará una sesión de *Apache Spark* como se ha indicado en el código 6.7 y cargaremos una pequeña muestra de datos. El aprovisionamiento de datos se explicará más adelante.

La primera parte consiste en identificar el *schema* para visualizar la estructura de los datos proporcionados por GitHub. La estructura completa está compuesta por un total de 752 campos, esta se puede consultar en el apéndice B.

Lo ideal sería obtener los *commits* diarios de cada lenguaje. Para ello analizamos los 38 tipos de eventos [79] que proporciona la API de GitHub. Como resultado se decide que el evento que vamos a utilizar como medida de referencia va a ser el “PushEvent”, el cual contiene los *commits* realizados por los usuarios de la plataforma.

Se observa que dicho evento efectivamente proporciona el número de *commits* realizados por un usuario, así como la fecha y hora en la que se realiza pero, no aporta ningún dato referente al lenguaje que contiene dicho *commit*.

Para obtener los lenguajes se evalúa la posibilidad de utilizar la API para realizar la solicitud del lenguaje correspondiente al repositorio sobre el que se realiza el *commit*. En una primera revisión de dicha API visualizamos que existe un límite de peticiones de la misma, siendo esta de 60 solicitudes por hora para usuarios no identificados y en caso de identificarse el límite sube apenas a 5000. Teniendo en cuenta el volumen de los datos se descarta esta opción.

Por todo ello se decide realizar un segundo análisis de la estructura para intentar identificar si existe la posibilidad de extraer estos datos de otro tipo de evento. En el transcurso de la misma se identifican dos eventos que proporcionan la información del lenguaje identificado en el repositorio, estos eventos son “PullRequestReviewCommentEvent” y “PullRequestEvent”.

Como resultado de lo comentado anteriormente se decide generar dos conjuntos de datos, uno con las fechas, número e *id* del repositorio sobre el que se realiza el commit y otro con el *id* y los lenguajes asociados al mismo, para posteriormente cruzar los datos y obtener los resultados deseados.

## 8.3. Aprovisionamiento de datos

En esta fase debido a las modestas dimensiones del clúster y la gran cantidad de datos (más de 600Gb/día) se realizara un pequeño aprovisionamiento para poder realizar el análisis y luego el aprovisionamiento completo consecuencia del análisis realizado. Como se ha comentado anteriormente GitHub dispone de una página web desde la que se pueden descargar los eventos generados en el sistema, organizados por año, mes, día y hora; esta pagina llamada *GitHub Archive* [62] permite descargar estos datos mediante el comando “wget” de linux tal y como se muestra en la tabla 8.1.

Tabla 8.1: Tabla de instrucciones de descarga de *GitHub Archive*[62].

Query	Command
Activity for 1/1/2015 @ 3PM UTC	wget http://data.githubarchive.org/2015-01-01-15.json.gz
Activity for 1/1/2015	wget http://data.githubarchive.org/2015-01-01-{0..23}.json.gz
Activity for all of January 2015	wget http://data.githubarchive.org/2015-01-{01..30}-{0..23}.json.gz

### 8.3.1. Aprovisionamiento de datos pre-análisis

Para poder realizar el análisis ya que no podemos obtener los datos completos debido a las limitaciones vamos obtener una pequeña muestra, esta se obtiene con la ejecución del comando mostrado en el código 8.1. Todos estos ficheros vienen comprimidos en formato “.gz”, para automatizar la extracción se ha creado un pequeño script (ver código 8.2).

Código 8.1: Comando para el preaprovisionamiento de datos de GitHub.

```
mkdir git
cd git
wget http://data.githubarchive.org/2015-01-01-\{0..23\}.json.gz
```

Código 8.2: Script para la extracción automática de los ficheros de GitHub.

```
#!/bin/bash
for filename in $HOME/git/*.gz; do
    printf "\nExtracting $filename..."
    gzip -d $filename
    printf "...SUCCESFULLY"
done
```

### 8.3.2. Aprovisionamiento de datos post-análisis

Una vez identificados los tres tipos de eventos que deseamos recuperar, procedemos a codificar una solución para paliar la escasez de capacidad del clúster.

La aplicación va a generar tres ficheros dónde se guardarán los *commits*, los lenguajes y los eventos que hayan reportado problemas. Para hacer la ejecución más eficiente se ejecutarán 16 hilos en paralelo, se mantendrán los ficheros abiertos y accesibles de forma única en cada hilo para evitar problemas de concurrencia.

Lo primero será generar los procesos, los enlaces de descarga y balancearlos entre los primeros. Esta parte se hace en el método “create\_balance\_threads()” (ver línea 26 en código 8.3). Una vez creado los enlaces y los procesos se ejecutan.

Código 8.3: Clase *Main* de la aplicación de aprovisionamiento de datos de GitHub.

```

7  public class Main {
8      private static final int HILOS = 16;
9      // URL sobre la que se construye la petición (año, mes, dia y hora
10     ↵ )
10     private static final String URL_BASE = "http://data.githubarchive.org/20%02d-%02d-%02d.json.gz";
11
12     //Lista con todos los hilos que se van a lanzar
13     private static final ArrayList<Process> process_list = new
14         ↵ ArrayList<>();
15
15     public static void main( String[] args ) {
16         long start = System.nanoTime();
17         create_balance_threads();
18         launch_threads();
19         long time = System.nanoTime() - start;
20         System.out.printf("Took %.3f seconds", time / 1e9);
21     }
22
23     /**
24      * Genera un numero de hilos y distribuye uniformemente los ficheros
25      ↵ que se descargaran
26     */
26     private static void create_balance_threads(){
27         int balanced = 0;
28         for (int i = 0; i < HILOS; i++) process_list.add(new Process());
29         URL url;
30         for (int anno = 15; anno <= 17; anno++) {
31             for (int mes = 1; mes <= 12; mes++) {
32                 for (int dia = 1; dia <= 31; dia++) {
33                     for (int hora = 0; hora < 24; hora++) {
34                         try {
35                             url = new URL(String.format(URL_BASE, anno, mes, dia,
36                                         ↵ hora));
37                             process_list.get(balanced).addURLs(url);
38                             balanced++;
39                             if (balanced == HILOS) balanced = 0;
40                         } catch (MalformedURLException e) {
41                             e.printStackTrace();
42                         }
43                     }
44                 }
45             }
46         }
47     }
48
49     /**

```

```

50 * Lanza todos los hilos y espera a que se termine de ejecutar
51 */
52 private static void launch_threads(){
53     process_list.forEach(Thread::start);
54     process_list.forEach(hilo -> {
55         try {
56             hilo.join();
57         } catch (InterruptedException e) {
58             e.printStackTrace();
59         }
60     });
61     Process.close_files();
62 }
63 }
```

Como se puede apreciar en el fragmento de código 8.3 cada hilo pertenece a una clase llamada “Process”. Esta clase realiza la descarga de datos, y hace la lectura del documento línea a línea, cada una de estas se envía a la tercera clase que es la encargada de comprobar su contenido y extraer los campos necesarios según el tipo de evento.

Vamos a ver la clase “Process”, la primera parte (ver fragmento de código 8.4) consiste en la declaración de las constantes y variables que se van a utilizar, así como la apertura de los ficheros de salida.

Código 8.4: Fragmento de la clase *Process*. Declaración de constantes y apertura de ficheros.

```

16 private static final String OUTPUT_LANG = "git_langs.json";
   ↵ // Fichero de salida con los lenguajes
17 private static final String OUTPUT_COMMITS = "git_commits.json";
   ↵ // Fichero de salida con los commits
18 private static final String OUTPUT_ERROR = "git_errors.json";
   ↵ // Fichero de salida con los errores
19
20 private ArrayList<URL> urls = new ArrayList<>();
21 static AtomicInteger processed_links = new AtomicInteger(0);
22 private Logger log;
23 private Extractor extractor = new Extractor();
24 private static PrintWriter out_lang; //Write para usar el disco
25 private static PrintWriter out_commit; //Write para usar el disco
26
27
28 static {
29     try {
30         out_lang = new PrintWriter(new FileWriter(OUTPUT_LANG, true));
31         out_commit = new PrintWriter(new FileWriter(OUTPUT_COMMITS, true))
32             ↵ );
33     } catch (IOException e) {
34         e.printStackTrace();
```

```
35  }
36 }
```

El fragmento de código 8.5 realiza la descarga y extracción de los datos, posteriormente se procesa cada línea y se guarda el resultado en el fichero correspondiente. El procesado se realiza mediante la clase “Extractor” que se verá más adelante, esta devuelve un *array* con dos campos, donde la posición 1 devuelve el tipo de evento al que corresponde la información extraída, esta se haya en la posición 0 de dicho *array*.

Código 8.5: Fragmento de la clase *Process*. Descarga, extracción y guardado.

```
56 private void downloadUsingNIO(URL url) {
57     String line = "";
58     String result [] ;
59     try {
60         GZIPInputStream gzip = new GZIPInputStream( url .openStream() );
61         BufferedReader bf = new BufferedReader( new InputStreamReader( gzip
62             → , "UTF-8" ) );
63         while (( line = bf .readLine()) != null) {
64             if ( line .length() > 0) {
65                 result = extractor .extract( line );
66                 if ( result != null) {
67                     if ( result [1] .equals( String .valueOf( LANG_TYPE )) write_lang
68                         → ( result [0] );
69                     else if ( result [1] .equals( String .valueOf( COMMIT_TYPE )) )
70                         → write_commit( result [0] );
71                 }
72             }
73         }
74     } catch (PathNotFoundException e) {
75         log .warning("Fallo al encontrar el path: " + line );
76         log_error( line );
77     }
78     processed_links .incrementAndGet();
79 }
```

Como ya se ha comentado, la clase “Extractor” es la encargada de procesar cada evento, comprobar si este pertenece a alguno de los eventos que deseamos guardar, en caso de pertenecer comprueba si será usado para extraer los lenguajes o los *commits*. Realiza la extracción de los datos seleccionados para cada caso y devuelve un json con los campos deseados, así como la identificación de contenido del evento.

En el fragmento 8.6 se declaran las constantes utilizadas para el proceso, entre las que se incluyen los literales de los eventos útiles y los XML Path Language (XPath)

Código 8.6: Fragmento de la clase *Extractor*. Declaración de constantes.

```

10 // Declaracion de eventos
11 private static final String EVENT_TYPE1 = "
    ↪ PullRequestReviewCommentEvent";
12 private static final String EVENT_TYPE2 = "PullRequestEvent";
13 private static final String EVENT_TYPE3 = "PushEvent";
14 // XPath necesarios
15 private static final String ID_EXPR = "$.repo.id";
16 private static final String HEAD_EXPR = "$.payload.pull_request.head.
    ↪ repo.language";
17 private static final String BASE_EXPR = "$.payload.pull_request.base.
    ↪ repo.language";
18 private static final String TYPE_EXPR = "$.type";
19 private static final String TIME_EXPR = "$.created_at";
20 private static final String SIZE_EXPR = "$.payload.size";
21
22 public static final int LANG_TYPE = 0;
23 public static final int COMMIT_TYPE = 1;

```

El método principal (ver fragmento 8.7) recibe el JSON completo, comprueba que existan los eventos deseados y actúa en consecuencia de la devolución. Si el JSON posee datos de lenguaje o *commits*, los extrae, en cualquier otro caso devuelve *null*.

Código 8.7: Fragmento de la clase *Extractor*. Método principal.

```

30 public String[] extract(String line) throws PathNotFoundException {
31     line = line.replaceAll("\n", ""); // Normalizar linea
32     String result[] = new String[2];
33     int type = check_type(line); //Comprobar type event
34
35     if (type == LANG_TYPE) {
36         result[0] = extract_langs(line);
37         result[1] = String.valueOf(LANG_TYPE);
38     } else if (type == COMMIT_TYPE) {
39         result[0] = extract_commits(line);
40         result[1] = String.valueOf(COMMIT_TYPE);
41     } else result = null;
42
43     return result;
44 }

```

Para la comprobación del tipo de evento se realiza la extracción del campo “*type*” y se comprueba con los literales declarados en el fragmento 8.6, haciendo uso del XPath definido a tal efecto. El método encargado de esto se puede ver en el fragmento de código 8.8.

Código 8.8: Fragmento de la clase *Extractor*. Comprobación de tipo.

```

54 private int check_type(String line) throws PathNotFoundException {
55     int type_num = -1;
56     String type = JsonPath.read(line, TYPE_EXPR);
57
58     if (type.equalsIgnoreCase(EVENT_TYPE1) || type.equalsIgnoreCase(
59         ↪ EVENT_TYPE2)) type_num = LANG_TYPE;
60     else if (type.equalsIgnoreCase(EVENT_TYPE3)) type_num = COMMIT_TYPE
61         ↪ ;
62     else type_num = -1;
63
64     return type_num;
65 }
```

Una vez que el tipo de evento ha sido identificado, se realiza la extracción de los datos elegidos durante el análisis. Para ello se hace uso de los XPath definidos a tal efecto, cada JSON recibido por esta clase es reducido a los campos indispensables, por lo tanto una vez extraídos los campos de lenguaje (ver fragmento 8.9) o los campos de relacionados con los *commits* (ver fragmento 8.10) estos son añadidos a un *HashMap* y devolviendo un JSON generado a partir del mapa.

Código 8.9: Fragmento de la clase *Extractor*. Método de extracción de lenguajes.

```

71 private String extract_langs(String line) throws
72     ↪ PathNotFoundException {
73     HashMap<String, String> language_event = new HashMap<>();
74     language_event.put("id", JsonPath.read(line, ID_EXPR).toString());
75     language_event.put("repo_head_lang", JsonPath.read(line, HEAD_EXPR)
76         ↪ );
77     language_event.put("repo_base_lang", JsonPath.read(line, BASE_EXPR)
78         ↪ );
79
80     return new JSONObject(language_event).toJSONString();
81 }
```

Código 8.10: Fragmento de la clase *Extractor*. Método de extacción de *commits*.

```

86 private String extract_commits(String line) throws
87     ↪ PathNotFoundException {
88     HashMap<String, String> language_event = new HashMap<>();
89     language_event.put("id", JsonPath.read(line, ID_EXPR).toString());
90     language_event.put("size", JsonPath.read(line, SIZE_EXPR).toString
91         ↪ ());
92     language_event.put("time", JsonPath.read(line, TIME_EXPR).toString
93         ↪ ().substring(0, 10));
94
95     return new JSONObject(language_event).toJSONString();
96 }
```

## 8.4. Procesado de datos

Llegados a este punto, con el aprovisionamiento de datos completo solo queda relacionar cada repositorio con su lenguaje correspondiente.

Como habíamos previsto algunos eventos han generado errores, vamos a descubrir que ha fallado en primer lugar, para ello se importa el fichero JSON (ver código 8.11) y se eliminan los registros corruptos.

Código 8.11: Creación del *DataFrame* y su tabla temporal

```
errors = spark.read.json('git/git_errors.json')
errors = errors.filter(errors._corrupt_record.isNull()).drop(errors._corrupt_record)
errors.createOrReplaceTempView("errors")

langs = spark.read.json('git/git_langs.json')
langs.createOrReplaceTempView("langs")

commits = spark.read.json('git/git_commits.json')
commits.createOrReplaceTempView("commits")
```

Estos ficheros corresponden al resultado del aprovisionamiento de datos, para comprobar cual es el problema intentamos realizar la extracción de datos mediante *Apache Spark* y nos da como resultado la tabla 8.2, en la cual podemos apreciar que el error procede de un resultado nulo al identificar el lenguaje de la cabecera. Los errores suponen información de 13673 repositorios, por lo tanto, se recuperará esta información.

Para recuperar estos datos y tener una base de datos de los lenguajes de cada repositorio más completa, vamos a extraer los repositorios que no existan en el *dataset* de lenguajes y vamos a unir ambos. Código 8.12.

Código 8.12: Creación de un *DataFrame* diferencial de lenguajes y unión de ambas.

```
lang_error = spark.sql("SELECT * FROM lang_error WHERE id NOT IN (
    SELECT id FROM langs)")
langs = langs.unionAll(lang_error)
```

Llegados a este punto poseemos dos *datasets*, uno con lenguajes asociados a un repositorio y otro con *commits* asociados a un repositorio. Lo único que resta es cruzar estos datos para ello se hará uso la consulta SQL del código 8.13. El resultado final se puede apreciar en la tabla 8.3.

Tabla 8.2: Resultados extraídos de los errores en el proceso de aprovisionamiento.

<b>id</b>	<b>repo_base_lang</b>	<b>repo_head_lang</b>
13746419	C#	null
2464908	PHP	null
117358	Perl6	null
8869463	Haxe	null
2625205	Python	null
10941409	JavaScript	null
9426192	Python	null
23259263	JavaScript	null
206341	Java	null
1141571	JavaScript	null
7850354	C	null
27539466	JavaScript	null
1975671	PHP	null
17312031	JavaScript	null
10653734	Scheme	null
25968039	Java	null
5782574	Lua	null
17293434	JavaScript	null
5022564	JavaScript	null
905738	Objective-C	null

Código 8.13: SQL para cruzar datos de lenguajes y *commits*.

```
SELECT
  c.time ,
  c.size ,
  l.repo_base_lang ,
  l.repo_head_lang
FROM commits c
INNER JOIN langs l ON c.id = l.id ;
```

Dado qué vamos a utilizar el campo “repo\_base\_lang” para graficar, hay que asegurar que no haya campos nulos en esta columna ni duplicados vamos a convertir este *dataframe* en un RDD para poder tratar los datos.

En el código 8.14 se muestra una función simple donde en caso de que el campo “base” sea nulo se le da el valor del campo “head”. Además como se ha comentado en la sección 2 los *dataframes* son objetos inalterable, por lo que el primer paso es obtener un RDD, realizar la transformación señalada y eliminar los campos nulos. Posteriormente se agrupan los datos por fecha y lenguaje y se suman los resultados para obtener un peso por fecha y lenguaje.

Código 8.14: Transformaciones del RDD para asegurar la calidad del dato.

```
def complete(x):
    if x[2] is None and x[3] is not None:
        return (x[0], x[1], x[3], x[3])
    return x

rstRDD = result.rdd.map(complete).filter(lambda x: x[2] is not None)
result = rstRDD.toDF().groupBy(result.time, result.repo_base_lang,
                                result.size).sum()
```

Tabla 8.3: Tabla de resultados cruzados de GitHub.

<b>date</b>	<b>size</b>	<b>id</b>	<b>repo_base_lang</b>	<b>repo_head_lang</b>
2017-01-01	1	15615485	JavaScript	JavaScript
2017-01-01	1	77792086	Shell	Shell
2017-01-01	1	75845481	Python	Python
2017-01-01	2	76744795	TypeScript	TypeScript
2017-01-01	1	19610165	JavaScript	JavaScript
2017-01-01	1	68236778	Arduino	Arduino
2017-01-01	1	19610165	JavaScript	JavaScript
2017-01-01	24	53193142	CSS	CSS
2017-01-01	2	72273997	null	Python
2017-01-01	14	28008709	VimL	VimL
2017-01-01	1	76200114	HTML	HTML
2017-01-01	12	76391154	JavaScript	JavaScript
2017-01-01	2	5542032	JavaScript	JavaScript
2017-01-01	2	56255411	HTML	HTML
2017-01-01	2	77799473	PHP	Rust
2017-01-01	1	77365497	PowerShell	PowerShell
2017-01-01	3	74260508	JavaScript	JavaScript
2017-01-01	1	60346491	PHP	PHP
2017-01-01	1	29375063	JavaScript	JavaScript
2017-01-01	3	77765458	Java	Java

## 8.5. Graficado de datos

# 9

## Conclusiones

### 9.1. Introducción

En este capítulo se expondrán las conclusiones extraídas como resultado del proyecto, además de los problemas encontrados a la hora de realizar el mismo y la forma en que se han abordado.

Para su realización ha sido necesario un curso de formación para el aprendizaje sobre el uso de las herramientas pertenecientes al ecosistema, así como la instalación manual del mismo en los equipos de mi domicilio que han realizado la función de clúster.

La proyección de este ha requerido de tres meses para el planteamiento, implementación y resolución; desarrollando para ello varios *scripts* y aplicaciones que han sido necesarios para la resolución de los diferentes problemas, más un informe final con las recomendaciones para el supuesto cliente en el que se ha basado este desarrollo.

## 9.2. Conclusiones

La primera fase de este proyecto consistió en la selección de las herramientas que se utilizarían, para ello se decidió prescindir de las más antiguas, aunque actualmente se encuentren en proyectos en producción, a favor de *Apache Spark* que posee el potencial no solo de sustituirlas sino de ofrecer mejor rendimiento sobre el mismo *hardware*.

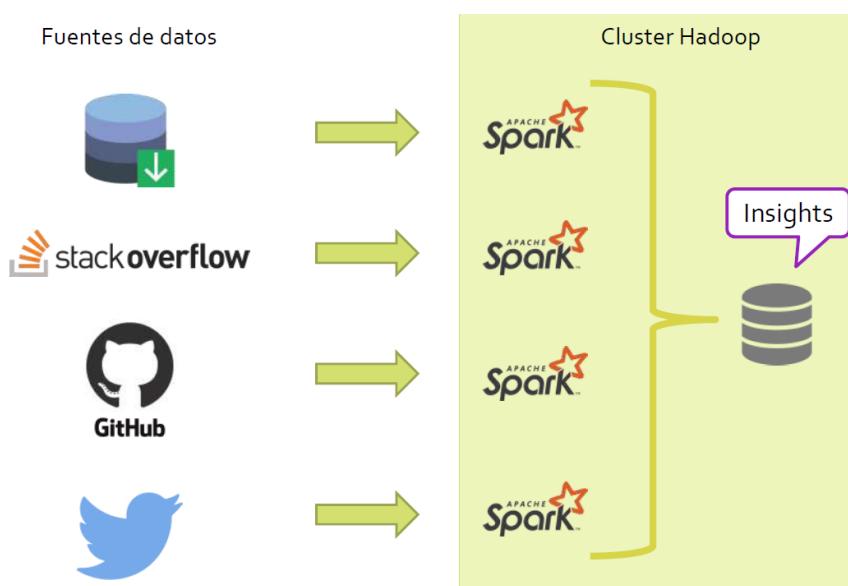
La instalación del sistema *big data* y su adaptación al *hardware* disponible requirió dos semanas; posteriormente se establecieron los objetivos para la demostración del uso de esta tecnología en un proyecto real.

La siguiente fase consistió en la selección y estudio de las fuentes de datos disponibles, así como establecer el valor y veracidad de los mismos. Algunos de los problemas que se encontraron durante esta fase fue por un lado que algunas de las empresas habían establecido una API de pago y por otro que los datos proporcionados eran intencionadamente incompletos. Para sortear estos inconvenientes se hizo uso de la API gratuita disponible y cruzando los datos para asegurar tanto el linaje como su veracidad.

Por desgracia, al ser un trabajo hipotético no se ha tenido acceso a los datos internos de la empresa que habría supuesto uno de los pilares principales sobre los que trabajar. Ver figura 9.1

Además, la mayoría de los datos eran semiestructurados o directamente no poseían estructura alguna. Para resolver esto se utilizaron *parsers* y un filtrado sumario, siendo recomendable la realización de un proyecto de *Data Text Mining* para la extracción de información, pero del que se prescindió debido a las limitaciones de tiempo para el desarrollo del proyecto.

Figura 9.1: Esquema de orígenes de datos del proyecto



## 9.3. Informe final

### 9.3.1. Lenguajes populares

El primer *insight* planteado corresponde a los lenguajes populares, para ello se han analizado las plataformas de Stack Overflow [60] (ver apartado 6) y GitHub [53] (ver apartado B), que nos proporcionan información sobre qué preguntan los programadores y en qué trabajan respectivamente.

Como se puede observar en la figura 9.3 referente a la analítica de GitHub de los últimos tres años hay varios lenguajes de sobra conocidos en el sector, pero han aparecido algunos como Rust, Go, Scala o Kotlin que han adquirido una gran importancia en relativamente poco tiempo, equiparándose o superando lenguajes con varios años de bagaje, por lo que sería altamente recomendable la realización de cursos relacionados con estos lenguajes debido a su potencial y rápida aceptación por los profesionales del sector.

En la figura 9.2 se puede observar cómo algunos de los lenguajes han tenido una menor incidencia en el último año con respecto a otros anteriores, con la salvedad de Python, el cual ha aumentado su presencia en este último año pese a ser un lenguaje establecido, por lo que se prevee un relanzamiento de este en los próximos años.

En definitiva la recomendación en función de los datos sería mantener los cursos en Javascript, Java, PHP, C++, Android e Ios y aumentar o crear cursos sobre Rust, Go, Scala, Kotlin y Python.

Figura 9.2: Histórico Stack Overflow

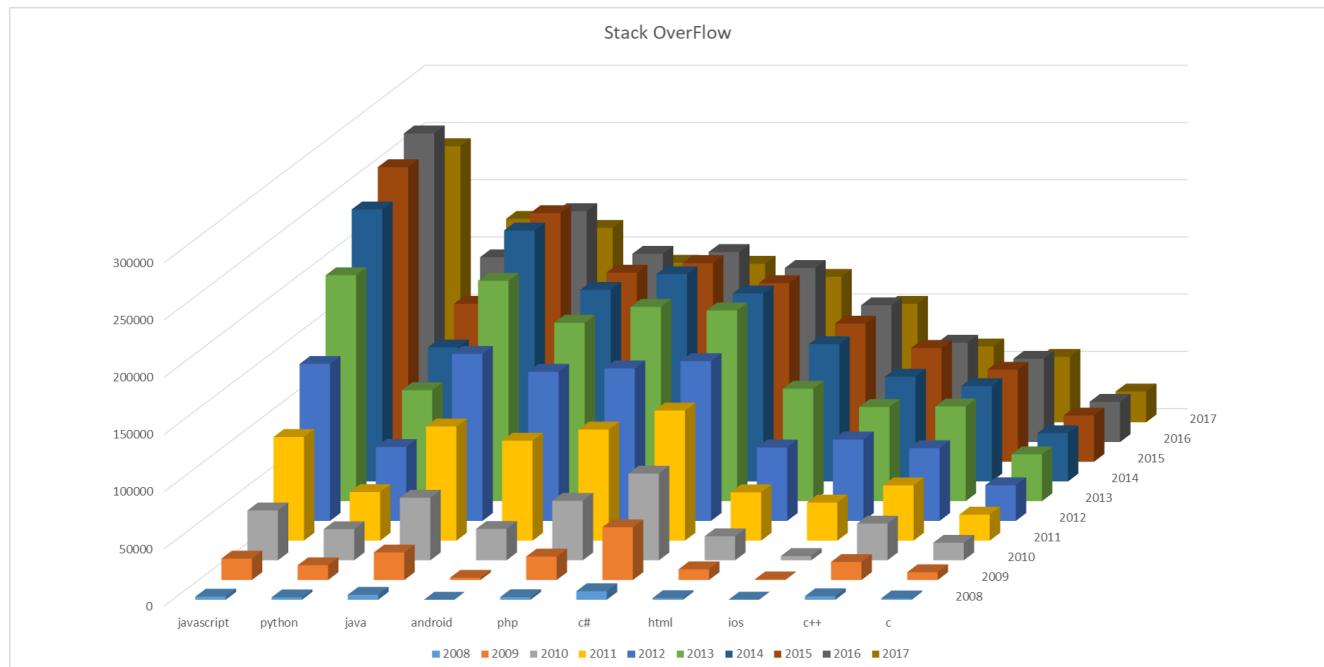
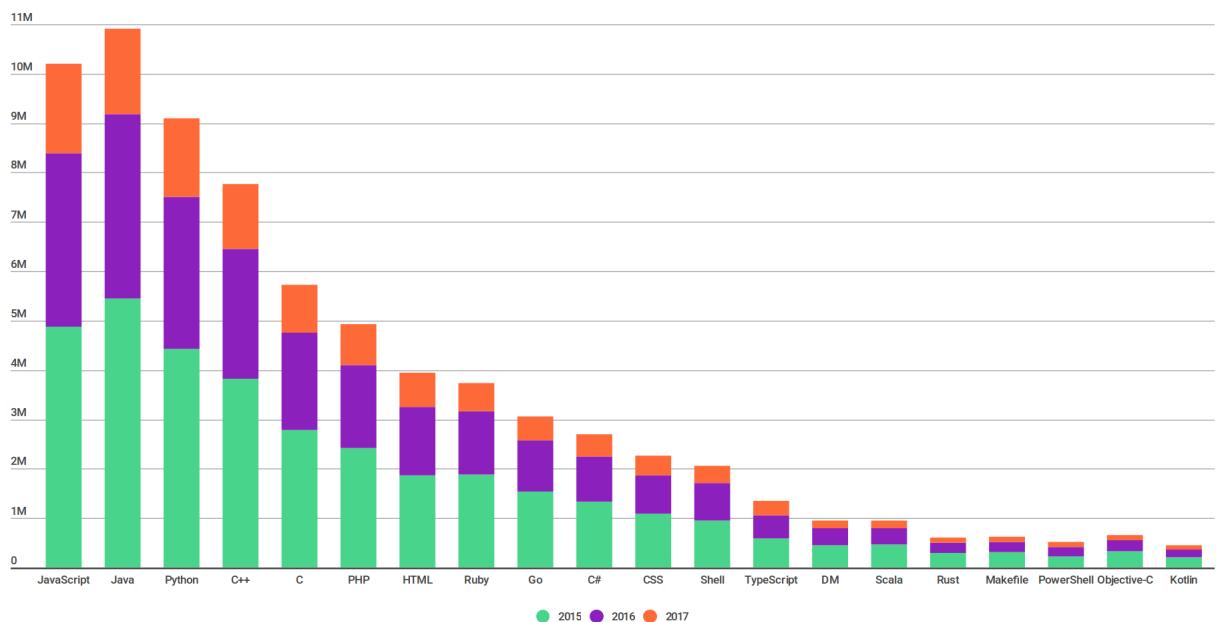


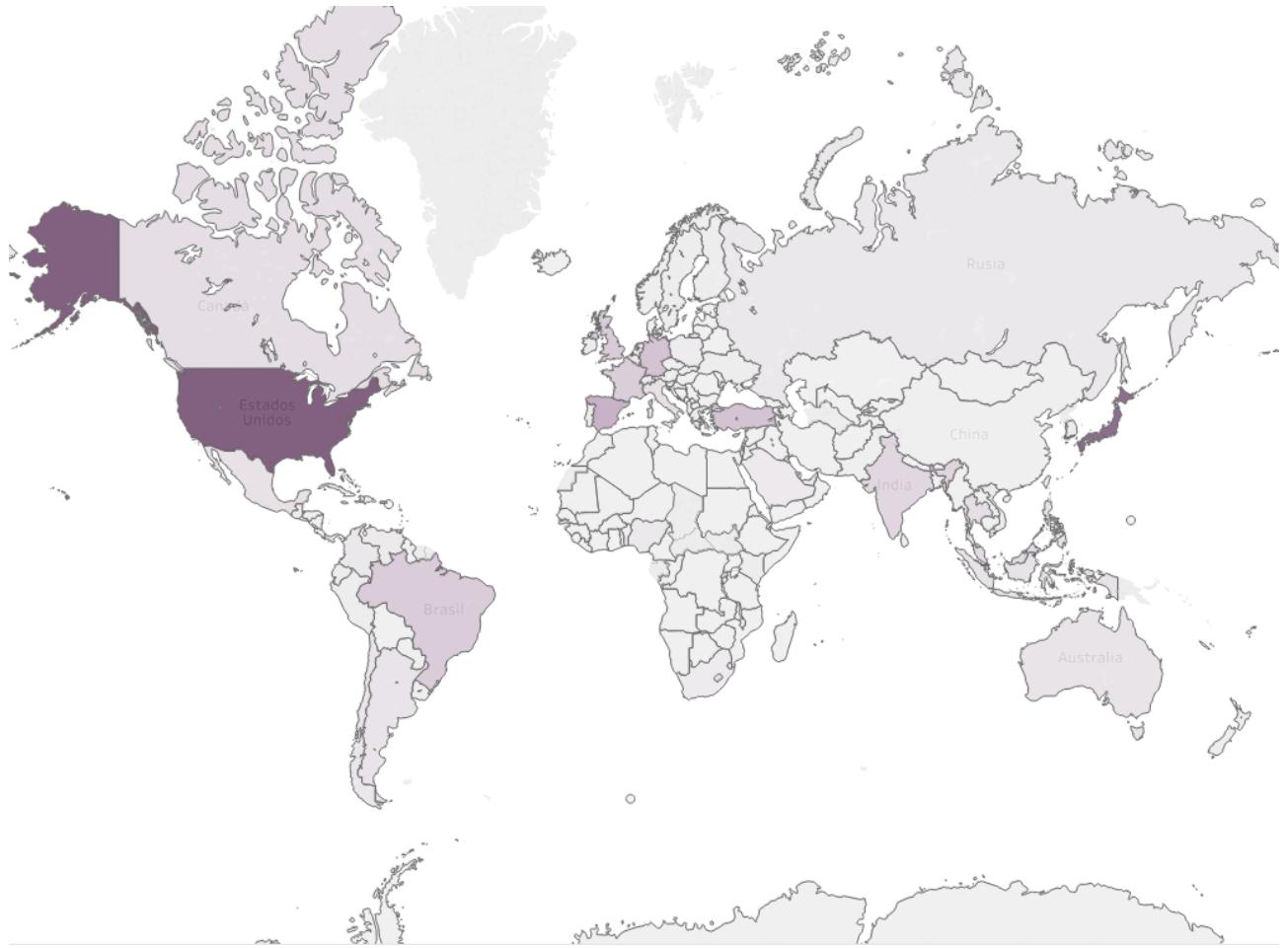
Figura 9.3: Histórico de GitHub



### 9.3.2. Regionalización de la empresa

Otro de los *insight* que solicitaba la empresa era un análisis sobre los países en los que enfocarse en caso de decidir regionalizar los cursos.

Figura 9.4: Mapamundi de incidencia de programadores por país

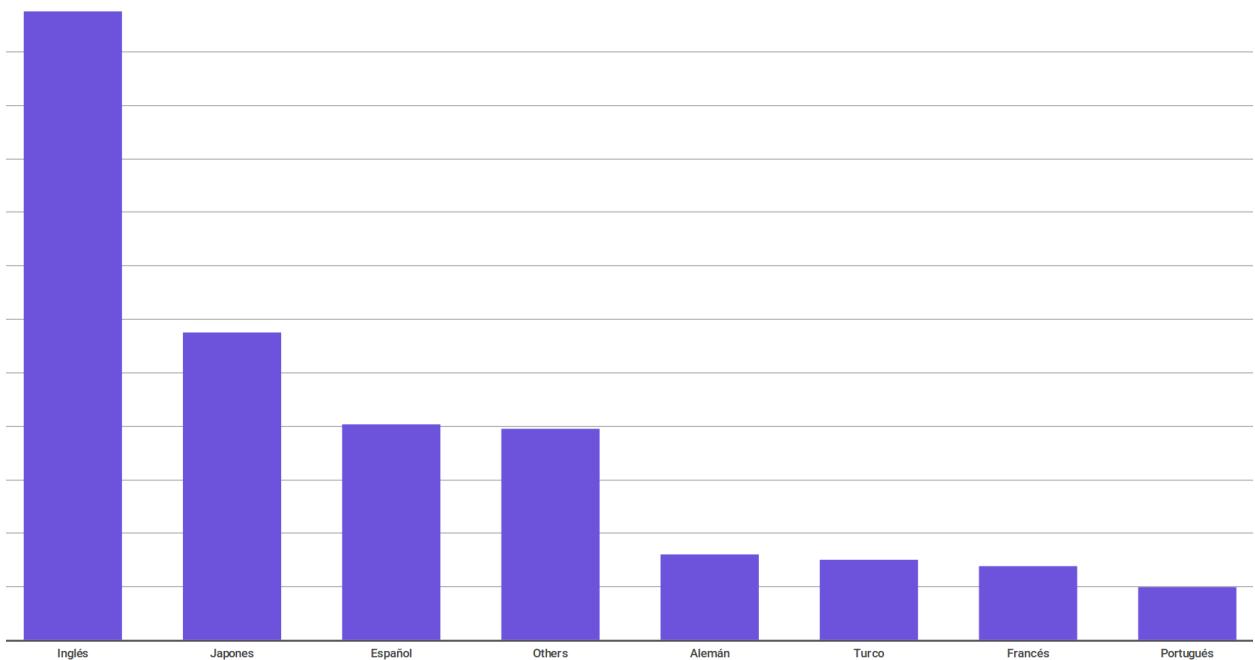


En la figura 9.4 se puede observar que la mayor incidencia se encuentra en Estados Unidos y Japón, seguido por Brasil, España, Francia, Turquía e India. La recomendación en este punto sería aumentar la presencia en estos dos países principalmente y luego expandirse hacia Europa y Sudamérica.

Estos datos han sido extraídos de Twitter [63], este punto es importante ya que explica la ausencia completa de datos en China, país donde se usa la plataforma Weibo [80] en sustitución de Twitter. Pero es un mercado de fácil acceso desde Japón sin tener que establecerse en el país.

### 9.3.3. Idioma vehicular

Figura 9.5: Gráfica orientativa sobre el uso de idiomas



En este punto no se aprecian grandes sorpresas, el idioma principal es el inglés que además es la segunda lengua de la mayoría de los profesionales del sector, por lo que se recomienda el uso de esta para los cursos. Además, el segundo puesto lo ocupa el japonés seguido del español, este primer idioma sería interesante a la hora de acercarse a una población con gran impacto tecnológico y el segundo debido al gran volumen de hispanohablantes.

El resto de idiomas tiene un impacto mucho menor, teniendo los idiomas minoritarios en conjunto un impacto similar al español.

La recomendación sería utilizar el inglés como idioma principal, asegurando la existencia de subtítulos o contenido en japonés y español para maximizar la cartera de clientes potenciales manteniendo un bajo costo por curso.

# Bibliografía

- [1] Statista Inc. *Statistics and facts about Big Data*. Feb. de 2017. URL: <https://www.statista.com/topics/1464/big-data/>.
- [2] Gartner. *Global smartphone sales*. Feb. de 2017. URL: <http://tinyurl.com/h77kptz>.
- [3] Apache Foundation. *Página web de Apache Spark*. 2018. URL: <https://spark.apache.org/>.
- [4] Apache Foundation. *Página web de Apache Hadoop*. 2018. URL: <https://hadoop.apache.org/>.
- [5] Open Knowledge International. *Definición de Open Data*. 2018. URL: <http://opendatahandbook.org/guide/es/what-is-open-data/>.
- [6] Wikipedia. *Página en Wikipedia dell World Data Center*. 2018. URL: [https://en.wikipedia.org/wiki/World\\_Data\\_Center](https://en.wikipedia.org/wiki/World_Data_Center).
- [7] *Portal de datos de la ONU*. 2018. URL: <http://data.un.org/>.
- [8] *Portal de datos abiertos de la ciudad de Madrid*. 2018. URL: <http://datos.madrid.es>.
- [9] *Gráfica sobre datos abiertos*. 2018. URL: <http://tinyurl.com/y72r44b8>.
- [10] James Manyika y col. *Open data: Unlocking innovation and performance with liquid information*. Inf. téc. McKinsey Global Institute, oct. de 2013. URL: <http://tinyurl.com/hrszsmf>.
- [11] Open Data Institute. *The value of open data*. 2018. URL: <https://theodi.org/the-value-of-open-data>.
- [12] Steve Lohr. *The Origins of "Big Data": An Etymological Detective Story*. Ene. de 2013. URL: <http://tinyurl.com/y93dwh3z>.
- [13] Juan Antonio Ayuso Rodríguez. «Proyecto Fin de Carrera: Landscape del Big Data». Tesis de mtría. Universidad Carlos III de Madrid, 2015.
- [14] *Big data - Cloud computing based requirements and capabilities. ITU-T Y.3600 (11/2015)*. International Telecommunication Union, nov. de 2015. URL: <http://www.itu.int/itu-t/recommendations/rec.aspx?rec=12584>.
- [15] IBM. *The Four V's of Big Data*. Jun. de 2017. URL: <http://www.ibmbigdatahub.com/infographic/four-vs-big-data>.

- [16] Montserrat Murillo González. «Sistema big data para el análisis de rutas de taxis en NYC». Tesis de mtría. Universidad Carlos III de Madrid, 2016.
- [17] Francisco París Soriano. «Análisis de la contratación en la administración estadounidense para la predicción de futuras licitaciones». Tesis de mtría. Universidad Carlos III de Madrid, 2016.
- [18] *Imagen de la evolución de las v's del big data*. 2018. URL: <http://www.elderresearch.com/company/blog/42-v-of-big-data>.
- [19] *Four Types of Big Data Analytics and Examples of Their Use*. 2018. URL: <http://tinyurl.com/ya7vzd79>.
- [20] Matt Turck. *Firing on All Cylinders: The 2017 Big Data Landscape*. 2018. URL: <http://mattturck.com/bigdata2017/>.
- [21] Apache Foundation. *Página web de Apache Cassandra*. 2018. URL: <https://cassandra.apache.org/>.
- [22] *Página web de MongoDB*. 2018. URL: <https://www.mongodb.com/>.
- [23] *Página web de SciPy*. 2018. URL: <https://www.scipy.org/>.
- [24] *Página web de Pandas*. 2018. URL: <https://databricks.com/>.
- [25] *Página web de Hortonworks*. 2018. URL: <https://es.hortonworks.com/>.
- [26] *Página web de Cloudera*. 2018. URL: <https://www.cloudera.com/>.
- [27] *Página web de Databricks*. 2018. URL: <https://databricks.com/>.
- [28] Louis Columbus. *Roundup Of Analytics, Big Data & BI Forecasts And Market Estimates, 2016*. 2018. URL: <http://tinyurl.com/ycag4feo>.
- [29] Berkeley University of California. *Chapter 4: Distributed and Parallel Computing*. 2018. URL: <http://wla.berkeley.edu/~cs61a/fa11/lectures/communication.html>.
- [30] *Imagen de un sistema distribuido*. 2018. URL: <http://tinyurl.com/ycn4zbwr>.
- [31] Juan Pablo Bustos Thamés. *Arquitectura de sistemas distribuidos*. 2018. URL: <http://tinyurl.com/y7ez4vlw>.
- [32] *Imagen peer-to-peer*. 2018. URL: [https://askleo.com/what\\_is\\_peer\\_to\\_peer\\_file\\_sharing/](https://askleo.com/what_is_peer_to_peer_file_sharing/).
- [33] Wikipedia. *Definición de clúster*. 2018. URL: [https://es.wikipedia.org/wiki/Cl%C3%BAster\\_\(inform%C3%A1tica\)](https://es.wikipedia.org/wiki/Cl%C3%BAster_(inform%C3%A1tica)).
- [34] *Imagen de clúster*. 2018. URL: <http://tinyurl.com/ybrhzawc>.
- [35] *Página de Unix Open Group*. 2018. URL: <http://opengroup.org/unix>.
- [36] *Being a Full Stack Developer*. 2018. URL: <https://tinyurl.com/y7ztcx23>.
- [37] *Página web del AMPLab de la Universidad de Berkeley*. 2018. URL: <https://AMPLab.cs.berkeley.edu/>.
- [38] *Página web del lenguaje Scala*. 2018. URL: <http://www.scala-lang.org/>.
- [39] *Página de Amazon Simple Storage Service*. 2018. URL: <https://aws.amazon.com/s3/>.

- [40] *RDDs are the new bytecode of Apache Spark*. 2018. URL: <http://tinyurl.com/oq89fb>.
- [41] *Introducing DataFrames in Apache Spark for Large Scale Data Science*. 2018. URL: <https://tinyurl.com/mweatbx>.
- [42] Saggi Neumann. *Spark vs. Hadoop MapReduce*. 2018. URL: <http://tinyurl.com/lobmnj7>.
- [43] *Data processing platforms architectures with SMACK: Spark, Mesos, Akka, Cassandra and Kafka*. 2018. URL: <https://tinyurl.com/yb8kcf23>.
- [44] Apache Foundation. *Página web de Apache Flink*. Jun. de 2017. URL: <https://flink.apache.org/>.
- [45] Apache Foundation. *Página web de Apache Storm*. Jun. de 2017. URL: <https://storm.apache.org/>.
- [46] Apache Foundation. *Página web de la wiki de Apache Flume*. 2018. URL: <https://tinyurl.com/y8shw6gs>.
- [47] Apache Foundation. *Página web de Apache Flume*. 2018. URL: <https://flume.apache.org/>.
- [48] *payload*. 2018. URL: [https://en.wikipedia.org/wiki/Payload\\_\(computing\)](https://en.wikipedia.org/wiki/Payload_(computing)).
- [49] *Página del proyecto jupyter*. 2018. URL: <http://jupyter.org/index.html>.
- [50] *Página web de IPython Interactive Computing*. 2018. URL: <https://ipython.org/>.
- [51] Javier Cristóbal. *Página web de Markdown Spain*. 2018. URL: <https://markdown.es/>.
- [52] *Página web de LaTeX Project*. 2018. URL: <https://markdown.es/>.
- [53] *Página de GitHub*. 2018. URL: <https://github.com/>.
- [54] *Portal web de la Agencia Española de Protección de Datos*. 2018. URL: <https://www.agpd.es/>.
- [55] *Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal*. Dic. de 1999. URL: <https://boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf>.
- [56] *Agencia Española de Protección de Datos. Principales derechos*. 2018. URL: <http://tinyurl.com/y9d3snor>.
- [57] *Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo de 27 de abril de 2016*. 2018. URL: <https://tinyurl.com/y7mxsdoy>.
- [58] New York Times Mark Scott. *Europe Approves Tough New Data Protection Rules*. Dic. de 2015. URL: [www.nytimes.com/2015/12/16/technology/eu-data-privacy.html](http://www.nytimes.com/2015/12/16/technology/eu-data-privacy.html).
- [59] *Enlace a la descarga de Ubuntu*. 2018. URL: <https://www.ubuntu.com/download/desktop>.
- [60] *Página de stackoverflow*. 2018. URL: <https://stackoverflow.com/>.
- [61] *Página de Git*. 2018. URL: <https://git-scm.com/>.

- [62] *Página de GitHub Archive*. 2018. URL: <https://www.githubarchive.org/>.
- [63] Inc. Twitter. *Página de Twitter*. 2018. URL: <https://twitter.com/>.
- [64] Apache Foundation. *Página de descarga de Apache Spark*. 2017. URL: <https://spark.apache.org/downloads.html>.
- [65] Oracle Corporation. *Página de descarga de Java SE Development Kit 8*. 2018. URL: <https://tinyurl.com/puhhbtr>.
- [66] Anaconda Distribution. *Página de Anaconda*. 2018. URL: <https://www.anaconda.com/>.
- [67] Apache Foundation. *Página de descarga de Apache Hadoop*. 2018. URL: <https://hadoop.apache.org/releases.html>.
- [68] Apache Foundation. *Página de descarga de Apache Flume*. 2018. URL: <https://flume.apache.org/download.html>.
- [69] *Página web de Stack Exchange*. 2018. URL: <https://stackexchange.com/>.
- [70] Inc Stack Exchange. *Stack Exchange Data Dump*. 2018. URL: <https://archive.org/details/stackexchange>.
- [71] *Página web de Creative Commons*. 2018. URL: <https://creativecommons.org>.
- [72] Igor Pavlov. *Página web de 7z format*. 2018. URL: <http://www.7-zip.org/7z.html>.
- [73] *Página web de Wikipedia sobre XML*. 2018. URL: <https://en.wikipedia.org/wiki/XML>.
- [74] Inc. Twitter. *Página web del API de Twitter*. 2018. URL: <https://developer.twitter.com/en/docs>.
- [75] International Organization for Standardization. *Página web de la ISO 3166-1 alpha-2*. 2018. URL: <https://www.iso.org/iso-3166-country-codes.html>.
- [76] International Organization for Standardization. *Página web de la ISO 639-1:2002*. 2018. URL: <https://www.iso.org/iso-639-language-codes.html>.
- [77] Inc. Time. *Página web de Unicorn List*. 2018. URL: <http://fortune.com/unicorns/>.
- [78] Ayuntamiento de Madrid. *Página web del repositorio oficial del ayto. De Madrid en GitHub*. 2018. URL: <https://github.com/AyuntamientoMadrid>.
- [79] *Página web de la API de GitHub*. 2018. URL: <https://tinyurl.com/mhbt733>.
- [80] *Página web de Weibo*. 2018. URL: <https://www.weibo.com/>.

# Apéndices

# A

## Estructura completa de un *tweet*

Código A.1: Esquema completo de la estructura de un *tweet*.

```
root
|--- contributors: string (nullable = true)
|--- coordinates: struct (nullable = true)
|   |--- coordinates: array (nullable = true)
|   |   |--- element: double (containsNull = true)
|   |--- type: string (nullable = true)
|--- created_at: string (nullable = true)
|--- display_text_range: array (nullable = true)
|   |--- element: long (containsNull = true)
|--- entities: struct (nullable = true)
|   |--- hashtags: array (nullable = true)
|   |   |--- element: struct (containsNull = true)
|   |   |   |--- indices: array (nullable = true)
|   |   |   |   |--- element: long (containsNull = true)
|   |   |   |--- text: string (nullable = true)
|--- media: array (nullable = true)
|   |--- element: struct (containsNull = true)
|   |   |--- description: string (nullable = true)
|   |   |--- display_url: string (nullable = true)
|   |   |--- expanded_url: string (nullable = true)
|   |   |--- id: long (nullable = true)
|   |   |--- id_str: string (nullable = true)
|   |   |--- indices: array (nullable = true)
|   |   |   |--- element: long (containsNull = true)
|   |   |--- media_url: string (nullable = true)
|   |   |--- media_url_https: string (nullable = true)
|   |   |--- sizes: struct (nullable = true)
|   |   |   |--- large: struct (nullable = true)
|   |   |   |   |--- h: long (nullable = true)
|   |   |   |   |--- resize: string (nullable = true)
|   |   |   |   |--- w: long (nullable = true)
|   |   |   |--- medium: struct (nullable = true)
|   |   |   |   |--- h: long (nullable = true)
|   |   |   |   |--- resize: string (nullable = true)
|   |   |   |   |--- w: long (nullable = true)
|   |   |   |--- small: struct (nullable = true)
|   |   |   |   |--- h: long (nullable = true)
|   |   |   |   |--- resize: string (nullable = true)
|   |   |   |   |--- w: long (nullable = true)
```

```

        |   |   |   |-- thumb: struct (nullable = true)
        |   |   |   |   |-- h: long (nullable = true)
        |   |   |   |   |-- resize: string (nullable = true)
        |   |   |   |   |-- w: long (nullable = true)
        |   |   |-- source_status_id: long (nullable = true)
        |   |   |-- source_status_id_str: string (nullable = true)
        |   |   |-- source_user_id: long (nullable = true)
        |   |   |-- source_user_id_str: string (nullable = true)
        |   |   |-- type: string (nullable = true)
        |   |   |-- url: string (nullable = true)
        |   |-- symbols: array (nullable = true)
        |   |   |-- element: struct (containsNull = true)
        |   |   |   |-- indices: array (nullable = true)
        |   |   |   |   |-- element: long (containsNull = true)
        |   |   |   |-- text: string (nullable = true)
        |   |-- urls: array (nullable = true)
        |   |   |-- element: struct (containsNull = true)
        |   |   |   |-- display_url: string (nullable = true)
        |   |   |   |-- expanded_url: string (nullable = true)
        |   |   |   |-- indices: array (nullable = true)
        |   |   |   |   |-- element: long (containsNull = true)
        |   |   |   |-- url: string (nullable = true)
        |   |-- user_mentions: array (nullable = true)
        |   |   |-- element: struct (containsNull = true)
        |   |   |   |-- id: long (nullable = true)
        |   |   |   |-- id_str: string (nullable = true)
        |   |   |   |-- indices: array (nullable = true)
        |   |   |   |   |-- element: long (containsNull = true)
        |   |   |   |-- name: string (nullable = true)
        |   |   |   |-- screen_name: string (nullable = true)
        |-- extended_entities: struct (nullable = true)
        |   |-- media: array (nullable = true)
        |   |   |-- element: struct (containsNull = true)
        |   |   |   |-- description: string (nullable = true)
        |   |   |   |-- display_url: string (nullable = true)
        |   |   |   |-- expanded_url: string (nullable = true)
        |   |   |   |-- id: long (nullable = true)
        |   |   |   |-- id_str: string (nullable = true)
        |   |   |   |-- indices: array (nullable = true)
        |   |   |   |   |-- element: long (containsNull = true)
        |   |   |   |-- media_url: string (nullable = true)
        |   |   |   |-- media_url_https: string (nullable = true)
        |   |   |   |-- sizes: struct (nullable = true)
        |   |   |   |   |-- large: struct (nullable = true)
        |   |   |   |   |   |-- h: long (nullable = true)
        |   |   |   |   |   |-- resize: string (nullable = true)
        |   |   |   |   |   |-- w: long (nullable = true)
        |   |   |   |   |-- medium: struct (nullable = true)
        |   |   |   |   |   |-- h: long (nullable = true)
        |   |   |   |   |   |-- resize: string (nullable = true)
        |   |   |   |   |   |-- w: long (nullable = true)
        |   |   |   |   |-- small: struct (nullable = true)
        |   |   |   |   |   |-- h: long (nullable = true)
        |   |   |   |   |   |-- resize: string (nullable = true)
        |   |   |   |   |   |-- w: long (nullable = true)
        |   |   |   |   |-- thumb: struct (nullable = true)
        |   |   |   |   |   |-- h: long (nullable = true)
        |   |   |   |   |   |-- resize: string (nullable = true)
        |   |   |   |   |   |-- w: long (nullable = true)
        |   |   |   |-- source_status_id: long (nullable = true)
        |   |   |   |-- source_status_id_str: string (nullable = true)
        |   |   |   |-- source_user_id: long (nullable = true)
        |   |   |   |-- source_user_id_str: string (nullable = true)
        |   |   |   |-- type: string (nullable = true)
        |   |   |   |-- url: string (nullable = true)
        |   |   |-- video_info: struct (nullable = true)
        |   |   |   |-- aspect_ratio: array (nullable = true)
        |   |   |   |   |-- element: long (containsNull = true)
        |   |   |   |-- duration_millis: long (nullable = true)
        |   |   |   |-- variants: array (nullable = true)
    
```

## APÉNDICE A. ESTRUCTURA COMPLETA DE UN TWEET

```
|-- element: struct (containsNull = true)
|  |-- bitrate: long (nullable = true)
|  |-- content_type: string (nullable = true)
|  |-- url: string (nullable = true)
|-- extended_tweet: struct (nullable = true)
|-- display_text_range: array (nullable = true)
|  |-- element: long (containsNull = true)
|-- entities: struct (nullable = true)
|-- hashtags: array (nullable = true)
|  |-- element: struct (containsNull = true)
|    |-- indices: array (nullable = true)
|      |-- element: long (containsNull = true)
|    |-- text: string (nullable = true)
|-- media: array (nullable = true)
|  |-- element: struct (containsNull = true)
|    |-- display_url: string (nullable = true)
|    |-- expanded_url: string (nullable = true)
|    |-- id: long (nullable = true)
|    |-- id_str: string (nullable = true)
|    |-- indices: array (nullable = true)
|      |-- element: long (containsNull = true)
|    |-- media_url: string (nullable = true)
|    |-- media_url_https: string (nullable = true)
|    |-- sizes: struct (nullable = true)
|      |-- large: struct (nullable = true)
|        |-- h: long (nullable = true)
|        |-- resize: string (nullable = true)
|        |-- w: long (nullable = true)
|      |-- medium: struct (nullable = true)
|        |-- h: long (nullable = true)
|        |-- resize: string (nullable = true)
|        |-- w: long (nullable = true)
|      |-- small: struct (nullable = true)
|        |-- h: long (nullable = true)
|        |-- resize: string (nullable = true)
|        |-- w: long (nullable = true)
|      |-- thumb: struct (nullable = true)
|        |-- h: long (nullable = true)
|        |-- resize: string (nullable = true)
|        |-- w: long (nullable = true)
|    |-- source_status_id: long (nullable = true)
|    |-- source_status_id_str: string (nullable = true)
|    |-- source_user_id: long (nullable = true)
|    |-- source_user_id_str: string (nullable = true)
|    |-- type: string (nullable = true)
|    |-- url: string (nullable = true)
|    |-- video_info: struct (nullable = true)
|      |-- aspect_ratio: array (nullable = true)
|        |-- element: long (containsNull = true)
|      |-- duration_millis: long (nullable = true)
|      |-- variants: array (nullable = true)
|        |-- element: struct (containsNull = true)
|          |-- bitrate: long (nullable = true)
|          |-- content_type: string (nullable = true)
|          |-- url: string (nullable = true)
|-- symbols: array (nullable = true)
|  |-- element: struct (containsNull = true)
|    |-- indices: array (nullable = true)
|      |-- element: long (containsNull = true)
|    |-- text: string (nullable = true)
|-- urls: array (nullable = true)
|  |-- element: struct (containsNull = true)
|    |-- display_url: string (nullable = true)
|    |-- expanded_url: string (nullable = true)
|    |-- indices: array (nullable = true)
|      |-- element: long (containsNull = true)
|    |-- url: string (nullable = true)
|-- user_mentions: array (nullable = true)
|  |-- element: struct (containsNull = true)
|    |-- id: long (nullable = true)
```

```

    |-- id: string (nullable = true)
    |-- indices: array (nullable = true)
    |-- element: long (containsNull = true)
    |-- name: string (nullable = true)
    |-- screen_name: string (nullable = true)
    |-- extended_entities: struct (nullable = true)
    |-- media: array (nullable = true)
        |-- element: struct (containsNull = true)
            |-- display_url: string (nullable = true)
            |-- expanded_url: string (nullable = true)
            |-- id: long (nullable = true)
            |-- id_str: string (nullable = true)
            |-- indices: array (nullable = true)
            |-- element: long (containsNull = true)
            |-- media_url: string (nullable = true)
            |-- media_url_https: string (nullable = true)
            |-- sizes: struct (nullable = true)
                |-- large: struct (nullable = true)
                    |-- h: long (nullable = true)
                    |-- resize: string (nullable = true)
                    |-- w: long (nullable = true)
                |-- medium: struct (nullable = true)
                    |-- h: long (nullable = true)
                    |-- resize: string (nullable = true)
                    |-- w: long (nullable = true)
                |-- small: struct (nullable = true)
                    |-- h: long (nullable = true)
                    |-- resize: string (nullable = true)
                    |-- w: long (nullable = true)
                |-- thumb: struct (nullable = true)
                    |-- h: long (nullable = true)
                    |-- resize: string (nullable = true)
                    |-- w: long (nullable = true)
            |-- source_status_id: long (nullable = true)
            |-- source_status_id_str: string (nullable = true)
            |-- source_user_id: long (nullable = true)
            |-- source_user_id_str: string (nullable = true)
            |-- type: string (nullable = true)
            |-- url: string (nullable = true)
            |-- video_info: struct (nullable = true)
                |-- aspect_ratio: array (nullable = true)
                    |-- element: long (containsNull = true)
                |-- duration_millis: long (nullable = true)
                |-- variants: array (nullable = true)
                    |-- element: struct (containsNull = true)
                        |-- bitrate: long (nullable = true)
                        |-- content_type: string (nullable = true)
                        |-- url: string (nullable = true)
            |-- full_text: string (nullable = true)
    |-- favorite_count: long (nullable = true)
    |-- favorited: boolean (nullable = true)
    |-- filter_level: string (nullable = true)
    |-- geo: struct (nullable = true)
        |-- coordinates: array (nullable = true)
            |-- element: double (containsNull = true)
        |-- type: string (nullable = true)
    |-- id: long (nullable = true)
    |-- id_str: string (nullable = true)
    |-- in_reply_to_screen_name: string (nullable = true)
    |-- in_reply_to_status_id: long (nullable = true)
    |-- in_reply_to_status_id_str: string (nullable = true)
    |-- in_reply_to_user_id: long (nullable = true)
    |-- in_reply_to_user_id_str: string (nullable = true)
    |-- is_quote_status: boolean (nullable = true)
    |-- lang: string (nullable = true)
    |-- place: struct (nullable = true)
        |-- bounding_box: struct (nullable = true)
            |-- coordinates: array (nullable = true)
                |-- element: array (containsNull = true)
                    |-- element: array (containsNull = true)

```

```
|      |      |      |-- element: double (containsNull = true)
|-- type: string (nullable = true)
|-- country: string (nullable = true)
|-- country_code: string (nullable = true)
|-- full_name: string (nullable = true)
|-- id: string (nullable = true)
|-- name: string (nullable = true)
|-- place_type: string (nullable = true)
|-- url: string (nullable = true)
|-- possibly_sensitive: boolean (nullable = true)
|-- quote_count: long (nullable = true)
|-- quoted_status: struct (nullable = true)
|   |-- contributors: string (nullable = true)
|   |-- coordinates: struct (nullable = true)
|       |-- coordinates: array (nullable = true)
|           |-- element: double (containsNull = true)
|           |-- type: string (nullable = true)
|   |-- created_at: string (nullable = true)
|   |-- display_text_range: array (nullable = true)
|       |-- element: long (containsNull = true)
|   |-- entities: struct (nullable = true)
|       |-- hashtags: array (nullable = true)
|           |-- element: struct (containsNull = true)
|               |-- indices: array (nullable = true)
|                   |-- element: long (containsNull = true)
|                   |-- text: string (nullable = true)
|   |-- media: array (nullable = true)
|       |-- element: struct (containsNull = true)
|           |-- description: string (nullable = true)
|           |-- display_url: string (nullable = true)
|           |-- expanded_url: string (nullable = true)
|           |-- id: long (nullable = true)
|           |-- id_str: string (nullable = true)
|           |-- indices: array (nullable = true)
|               |-- element: long (containsNull = true)
|           |-- media_url: string (nullable = true)
|           |-- media_url_https: string (nullable = true)
|           |-- sizes: struct (nullable = true)
|               |-- large: struct (nullable = true)
|                   |-- h: long (nullable = true)
|                   |-- resize: string (nullable = true)
|                   |-- w: long (nullable = true)
|               |-- medium: struct (nullable = true)
|                   |-- h: long (nullable = true)
|                   |-- resize: string (nullable = true)
|                   |-- w: long (nullable = true)
|               |-- small: struct (nullable = true)
|                   |-- h: long (nullable = true)
|                   |-- resize: string (nullable = true)
|                   |-- w: long (nullable = true)
|               |-- thumb: struct (nullable = true)
|                   |-- h: long (nullable = true)
|                   |-- resize: string (nullable = true)
|                   |-- w: long (nullable = true)
|           |-- source_status_id: long (nullable = true)
|           |-- source_status_id_str: string (nullable = true)
|           |-- source_user_id: long (nullable = true)
|           |-- source_user_id_str: string (nullable = true)
|           |-- type: string (nullable = true)
|           |-- url: string (nullable = true)
|   |-- symbols: array (nullable = true)
|       |-- element: struct (containsNull = true)
|           |-- indices: array (nullable = true)
|               |-- element: long (containsNull = true)
|               |-- text: string (nullable = true)
|   |-- urls: array (nullable = true)
|       |-- element: struct (containsNull = true)
|           |-- display_url: string (nullable = true)
|           |-- expanded_url: string (nullable = true)
|           |-- indices: array (nullable = true)
```

## APÉNDICE A. ESTRUCTURA COMPLETA DE UN TWEET

```

        |   |-- element: long (containsNull = true)
        |-- media_url: string (nullable = true)
        |-- media_url_https: string (nullable = true)
        |-- sizes: struct (nullable = true)
            |-- large: struct (nullable = true)
                |-- h: long (nullable = true)
                |-- resize: string (nullable = true)
                |-- w: long (nullable = true)
            |-- medium: struct (nullable = true)
                |-- h: long (nullable = true)
                |-- resize: string (nullable = true)
                |-- w: long (nullable = true)
            |-- small: struct (nullable = true)
                |-- h: long (nullable = true)
                |-- resize: string (nullable = true)
                |-- w: long (nullable = true)
            |-- thumb: struct (nullable = true)
                |-- h: long (nullable = true)
                |-- resize: string (nullable = true)
                |-- w: long (nullable = true)
        |-- source_status_id: long (nullable = true)
        |-- source_status_id_str: string (nullable = true)
        |-- source_user_id: long (nullable = true)
        |-- source_user_id_str: string (nullable = true)
        |-- type: string (nullable = true)
        |-- url: string (nullable = true)
        |-- video_info: struct (nullable = true)
            |-- aspect_ratio: array (nullable = true)
                |-- element: long (containsNull = true)
            |-- duration_millis: long (nullable = true)
            |-- variants: array (nullable = true)
                |-- element: struct (containsNull = true)
                    |-- bitrate: long (nullable = true)
                    |-- content_type: string (nullable = true)
                    |-- url: string (nullable = true)
        |-- symbols: array (nullable = true)
            |-- element: struct (containsNull = true)
                |-- indices: array (nullable = true)
                    |-- element: long (containsNull = true)
                |-- text: string (nullable = true)
        |-- urls: array (nullable = true)
            |-- element: struct (containsNull = true)
                |-- display_url: string (nullable = true)
                |-- expanded_url: string (nullable = true)
                |-- indices: array (nullable = true)
                    |-- element: long (containsNull = true)
                |-- url: string (nullable = true)
        |-- user_mentions: array (nullable = true)
            |-- element: struct (containsNull = true)
                |-- id: long (nullable = true)
                |-- id_str: string (nullable = true)
                |-- indices: array (nullable = true)
                    |-- element: long (containsNull = true)
                |-- name: string (nullable = true)
                |-- screen_name: string (nullable = true)
    -- extended_entities: struct (nullable = true)
        |-- media: array (nullable = true)
            |-- element: struct (containsNull = true)
                |-- display_url: string (nullable = true)
                |-- expanded_url: string (nullable = true)
                |-- id: long (nullable = true)
                |-- id_str: string (nullable = true)
                |-- indices: array (nullable = true)
                    |-- element: long (containsNull = true)
                |-- media_url: string (nullable = true)
                |-- media_url_https: string (nullable = true)
                |-- sizes: struct (nullable = true)
                    |-- large: struct (nullable = true)
                        |-- h: long (nullable = true)
                        |-- resize: string (nullable = true)

```

## APÉNDICE A. ESTRUCTURA COMPLETA DE UN TWEET

```
|    |    |-- w: long (nullable = true)
|    |    |-- medium: struct (nullable = true)
|    |    |    |-- h: long (nullable = true)
|    |    |    |-- resize: string (nullable = true)
|    |    |    |-- w: long (nullable = true)
|    |    |-- small: struct (nullable = true)
|    |    |    |-- h: long (nullable = true)
|    |    |    |-- resize: string (nullable = true)
|    |    |    |-- w: long (nullable = true)
|    |    |-- thumb: struct (nullable = true)
|    |    |    |-- h: long (nullable = true)
|    |    |    |-- resize: string (nullable = true)
|    |    |    |-- w: long (nullable = true)
|    |    |-- source_status_id: long (nullable = true)
|    |    |-- source_status_id_str: string (nullable = true)
|    |    |-- source_user_id: long (nullable = true)
|    |    |-- source_user_id_str: string (nullable = true)
|    |    |-- type: string (nullable = true)
|    |    |-- url: string (nullable = true)
|    |    |-- video_info: struct (nullable = true)
|    |    |    |-- aspect_ratio: array (nullable = true)
|    |    |    |    |-- element: long (containsNull = true)
|    |    |    |-- duration_millis: long (nullable = true)
|    |    |    |-- variants: array (nullable = true)
|    |    |    |    |-- element: struct (containsNull = true)
|    |    |    |    |    |-- bitrate: long (nullable = true)
|    |    |    |    |    |-- content_type: string (nullable = true)
|    |    |    |    |    |-- url: string (nullable = true)
|    |-- full_text: string (nullable = true)
|-- favorite_count: long (nullable = true)
|-- favorited: boolean (nullable = true)
|-- filter_level: string (nullable = true)
|-- geo: struct (nullable = true)
|    |-- coordinates: array (nullable = true)
|    |    |-- element: double (containsNull = true)
|    |-- type: string (nullable = true)
|-- id: long (nullable = true)
|-- id_str: string (nullable = true)
|-- in_reply_to_screen_name: string (nullable = true)
|-- in_reply_to_status_id: long (nullable = true)
|-- in_reply_to_status_id_str: string (nullable = true)
|-- in_reply_to_user_id: long (nullable = true)
|-- in_reply_to_user_id_str: string (nullable = true)
|-- is_quote_status: boolean (nullable = true)
|-- lang: string (nullable = true)
|-- place: struct (nullable = true)
|    |-- bounding_box: struct (nullable = true)
|    |    |-- coordinates: array (nullable = true)
|    |    |    |-- element: array (containsNull = true)
|    |    |    |    |-- element: array (containsNull = true)
|    |    |    |    |    |-- element: double (containsNull = true)
|    |    |-- type: string (nullable = true)
|    |-- country: string (nullable = true)
|    |-- country_code: string (nullable = true)
|    |-- full_name: string (nullable = true)
|    |-- id: string (nullable = true)
|    |-- name: string (nullable = true)
|    |-- place_type: string (nullable = true)
|    |-- url: string (nullable = true)
|-- possibly_sensitive: boolean (nullable = true)
|-- quote_count: long (nullable = true)
|-- quoted_status_id: long (nullable = true)
|-- quoted_status_id_str: string (nullable = true)
|-- reply_count: long (nullable = true)
|-- retweet_count: long (nullable = true)
|--retweeted: boolean (nullable = true)
|-- scopes: struct (nullable = true)
|    |-- followers: boolean (nullable = true)
|-- source: string (nullable = true)
|-- text: string (nullable = true)
```

## APÉNDICE A. ESTRUCTURA COMPLETA DE UN TWEET

---

```
-- truncated: boolean (nullable = true)
-- user: struct (nullable = true)
|-- contributors_enabled: boolean (nullable = true)
|-- created_at: string (nullable = true)
|-- default_profile: boolean (nullable = true)
|-- default_profile_image: boolean (nullable = true)
|-- description: string (nullable = true)
|-- favourites_count: long (nullable = true)
|-- follow_request_sent: string (nullable = true)
|-- followers_count: long (nullable = true)
|-- following: string (nullable = true)
|-- friends_count: long (nullable = true)
|-- geo_enabled: boolean (nullable = true)
|-- id: long (nullable = true)
|-- id_str: string (nullable = true)
|-- is_translator: boolean (nullable = true)
|-- lang: string (nullable = true)
|-- listed_count: long (nullable = true)
|-- location: string (nullable = true)
|-- name: string (nullable = true)
|-- notifications: string (nullable = true)
|-- profile_background_color: string (nullable = true)
|-- profile_background_image_url: string (nullable = true)
|-- profile_background_image_url_https: string (nullable = true)
|-- profile_background_tile: boolean (nullable = true)
|-- profile_banner_url: string (nullable = true)
|-- profile_image_url: string (nullable = true)
|-- profile_image_url_https: string (nullable = true)
|-- profile_link_color: string (nullable = true)
|-- profile_sidebar_border_color: string (nullable = true)
|-- profile_sidebar_fill_color: string (nullable = true)
|-- profile_text_color: string (nullable = true)
|-- profile_use_background_image: boolean (nullable = true)
|-- protected: boolean (nullable = true)
|-- screen_name: string (nullable = true)
|-- statuses_count: long (nullable = true)
|-- time_zone: string (nullable = true)
|-- translator_type: string (nullable = true)
|-- url: string (nullable = true)
|-- utc_offset: long (nullable = true)
|-- verified: boolean (nullable = true)
-- quoted_status_id: long (nullable = true)
-- quoted_status_id_str: string (nullable = true)
-- reply_count: long (nullable = true)
-- retweet_count: long (nullable = true)
--retweeted: boolean (nullable = true)
--retweeted_status: struct (nullable = true)
|-- contributors: string (nullable = true)
|-- coordinates: struct (nullable = true)
| |-- coordinates: array (nullable = true)
| | |-- element: double (containsNull = true)
| |-- type: string (nullable = true)
|-- created_at: string (nullable = true)
|-- display_text_range: array (nullable = true)
| |-- element: long (containsNull = true)
|-- entities: struct (nullable = true)
| |-- hashtags: array (nullable = true)
| | |-- element: struct (containsNull = true)
| | | |-- indices: array (nullable = true)
| | | | |-- element: long (containsNull = true)
| | | |-- text: string (nullable = true)
|-- media: array (nullable = true)
| |-- element: struct (containsNull = true)
| | |-- display_url: string (nullable = true)
| | |-- expanded_url: string (nullable = true)
| | |-- id: long (nullable = true)
| | |-- id_str: string (nullable = true)
| | |-- indices: array (nullable = true)
| | | |-- element: long (containsNull = true)
| | |-- media_url: string (nullable = true)
```

```

    |--- media_url_https: string (nullable = true)
    |--- sizes: struct (nullable = true)
    |   |--- large: struct (nullable = true)
    |   |   |--- h: long (nullable = true)
    |   |   |--- resize: string (nullable = true)
    |   |   |--- w: long (nullable = true)
    |   |--- medium: struct (nullable = true)
    |   |   |--- h: long (nullable = true)
    |   |   |--- resize: string (nullable = true)
    |   |   |--- w: long (nullable = true)
    |   |--- small: struct (nullable = true)
    |   |   |--- h: long (nullable = true)
    |   |   |--- resize: string (nullable = true)
    |   |   |--- w: long (nullable = true)
    |   |--- thumb: struct (nullable = true)
    |   |   |--- h: long (nullable = true)
    |   |   |--- resize: string (nullable = true)
    |   |   |--- w: long (nullable = true)
    |--- source_status_id: long (nullable = true)
    |--- source_status_id_str: string (nullable = true)
    |--- source_user_id: long (nullable = true)
    |--- source_user_id_str: string (nullable = true)
    |--- type: string (nullable = true)
    |--- url: string (nullable = true)
-- symbols: array (nullable = true)
|--- element: struct (containsNull = true)
|   |--- indices: array (nullable = true)
|   |   |--- element: long (containsNull = true)
|   |--- text: string (nullable = true)
-- urls: array (nullable = true)
|--- element: struct (containsNull = true)
|   |--- display_url: string (nullable = true)
|   |--- expanded_url: string (nullable = true)
|   |--- indices: array (nullable = true)
|   |   |--- element: long (containsNull = true)
|   |--- url: string (nullable = true)
-- user_mentions: array (nullable = true)
|--- element: struct (containsNull = true)
|   |--- id: long (nullable = true)
|   |--- id_str: string (nullable = true)
|   |--- indices: array (nullable = true)
|   |   |--- element: long (containsNull = true)
|   |--- name: string (nullable = true)
|   |--- screen_name: string (nullable = true)
-- extended_entities: struct (nullable = true)
-- media: array (nullable = true)
|--- element: struct (containsNull = true)
|   |--- display_url: string (nullable = true)
|   |--- expanded_url: string (nullable = true)
|   |--- id: long (nullable = true)
|   |--- id_str: string (nullable = true)
|   |--- indices: array (nullable = true)
|   |   |--- element: long (containsNull = true)
|--- media_url: string (nullable = true)
|--- media_url_https: string (nullable = true)
|--- sizes: struct (nullable = true)
|   |--- large: struct (nullable = true)
|   |   |--- h: long (nullable = true)
|   |   |--- resize: string (nullable = true)
|   |   |--- w: long (nullable = true)
|   |--- medium: struct (nullable = true)
|   |   |--- h: long (nullable = true)
|   |   |--- resize: string (nullable = true)
|   |   |--- w: long (nullable = true)
|   |--- small: struct (nullable = true)
|   |   |--- h: long (nullable = true)
|   |   |--- resize: string (nullable = true)
|   |   |--- w: long (nullable = true)
|   |--- thumb: struct (nullable = true)
|   |   |--- h: long (nullable = true)

```

```

    |      |      |-- resize: string (nullable = true)
    |      |      |-- w: long (nullable = true)
    |-- source_status_id: long (nullable = true)
    |-- source_status_id_str: string (nullable = true)
    |-- source_user_id: long (nullable = true)
    |-- source_user_id_str: string (nullable = true)
    |-- type: string (nullable = true)
    |-- url: string (nullable = true)
    |-- video_info: struct (nullable = true)
        |-- aspect_ratio: array (nullable = true)
            |-- element: long (containsNull = true)
        |-- duration_millis: long (nullable = true)
        |-- variants: array (nullable = true)
            |-- element: struct (containsNull = true)
                |-- bitrate: long (nullable = true)
                |-- content_type: string (nullable = true)
                |-- url: string (nullable = true)
    |-- extended_tweet: struct (nullable = true)
    |-- display_text_range: array (nullable = true)
        |-- element: long (containsNull = true)
    |-- entities: struct (nullable = true)
        |-- hashtags: array (nullable = true)
            |-- element: struct (containsNull = true)
                |-- indices: array (nullable = true)
                    |-- element: long (containsNull = true)
                |-- text: string (nullable = true)
        |-- media: array (nullable = true)
            |-- element: struct (containsNull = true)
                |-- description: string (nullable = true)
                |-- display_url: string (nullable = true)
                |-- expanded_url: string (nullable = true)
                |-- id: long (nullable = true)
                |-- id_str: string (nullable = true)
                |-- indices: array (nullable = true)
                    |-- element: long (containsNull = true)
                |-- media_url: string (nullable = true)
                |-- media_url_https: string (nullable = true)
                |-- sizes: struct (nullable = true)
                    |-- large: struct (nullable = true)
                        |-- h: long (nullable = true)
                        |-- resize: string (nullable = true)
                        |-- w: long (nullable = true)
                    |-- medium: struct (nullable = true)
                        |-- h: long (nullable = true)
                        |-- resize: string (nullable = true)
                        |-- w: long (nullable = true)
                    |-- small: struct (nullable = true)
                        |-- h: long (nullable = true)
                        |-- resize: string (nullable = true)
                        |-- w: long (nullable = true)
                    |-- thumb: struct (nullable = true)
                        |-- h: long (nullable = true)
                        |-- resize: string (nullable = true)
                        |-- w: long (nullable = true)
                |-- source_status_id: long (nullable = true)
                |-- source_status_id_str: string (nullable = true)
                |-- source_user_id: long (nullable = true)
                |-- source_user_id_str: string (nullable = true)
                |-- type: string (nullable = true)
                |-- url: string (nullable = true)
                |-- video_info: struct (nullable = true)
                    |-- aspect_ratio: array (nullable = true)
                        |-- element: long (containsNull = true)
                    |-- duration_millis: long (nullable = true)
                    |-- variants: array (nullable = true)
                        |-- element: struct (containsNull = true)
                            |-- bitrate: long (nullable = true)
                            |-- content_type: string (nullable = true)
                            |-- url: string (nullable = true)
    |-- symbols: array (nullable = true)

```

```

    |-- element: struct (containsNull = true)
    |   |-- indices: array (nullable = true)
    |   |   |-- element: long (containsNull = true)
    |   |   |-- text: string (nullable = true)
    |-- urls: array (nullable = true)
        |-- element: struct (containsNull = true)
            |-- display_url: string (nullable = true)
            |-- expanded_url: string (nullable = true)
            |-- indices: array (nullable = true)
                |-- element: long (containsNull = true)
            |-- url: string (nullable = true)
    |-- user_mentions: array (nullable = true)
        |-- element: struct (containsNull = true)
            |-- id: long (nullable = true)
            |-- id_str: string (nullable = true)
            |-- indices: array (nullable = true)
                |-- element: long (containsNull = true)
            |-- name: string (nullable = true)
            |-- screen_name: string (nullable = true)
    |-- extended_entities: struct (nullable = true)
        |-- media: array (nullable = true)
            |-- element: struct (containsNull = true)
                |-- description: string (nullable = true)
                |-- display_url: string (nullable = true)
                |-- expanded_url: string (nullable = true)
                |-- id: long (nullable = true)
                |-- id_str: string (nullable = true)
                |-- indices: array (nullable = true)
                    |-- element: long (containsNull = true)
                |-- media_url: string (nullable = true)
                |-- media_url_https: string (nullable = true)
                |-- sizes: struct (nullable = true)
                    |-- large: struct (nullable = true)
                        |-- h: long (nullable = true)
                        |-- resize: string (nullable = true)
                        |-- w: long (nullable = true)
                    |-- medium: struct (nullable = true)
                        |-- h: long (nullable = true)
                        |-- resize: string (nullable = true)
                        |-- w: long (nullable = true)
                    |-- small: struct (nullable = true)
                        |-- h: long (nullable = true)
                        |-- resize: string (nullable = true)
                        |-- w: long (nullable = true)
                    |-- thumb: struct (nullable = true)
                        |-- h: long (nullable = true)
                        |-- resize: string (nullable = true)
                        |-- w: long (nullable = true)
                |-- source_status_id: long (nullable = true)
                |-- source_status_id_str: string (nullable = true)
                |-- source_user_id: long (nullable = true)
                |-- source_user_id_str: string (nullable = true)
                |-- type: string (nullable = true)
                |-- url: string (nullable = true)
                |-- video_info: struct (nullable = true)
                    |-- aspect_ratio: array (nullable = true)
                        |-- element: long (containsNull = true)
                    |-- duration_millis: long (nullable = true)
                    |-- variants: array (nullable = true)
                        |-- element: struct (containsNull = true)
                            |-- bitrate: long (nullable = true)
                            |-- content_type: string (nullable = true)
                            |-- url: string (nullable = true)
                |-- full_text: string (nullable = true)
    |-- favorite_count: long (nullable = true)
    |-- favorited: boolean (nullable = true)
    |-- filter_level: string (nullable = true)
    |-- geo: struct (nullable = true)
        |-- coordinates: array (nullable = true)
            |-- element: double (containsNull = true)

```

```

    |   |--- type: string (nullable = true)
    |--- id: long (nullable = true)
    |--- id_str: string (nullable = true)
    |--- in_reply_to_screen_name: string (nullable = true)
    |--- in_reply_to_status_id: long (nullable = true)
    |--- in_reply_to_status_id_str: string (nullable = true)
    |--- in_reply_to_user_id: long (nullable = true)
    |--- in_reply_to_user_id_str: string (nullable = true)
    |--- is_quote_status: boolean (nullable = true)
    |--- lang: string (nullable = true)
    |--- place: struct (nullable = true)
    |   |--- bounding_box: struct (nullable = true)
    |   |   |--- coordinates: array (nullable = true)
    |   |   |   |--- element: array (containsNull = true)
    |   |   |   |   |--- element: array (containsNull = true)
    |   |   |   |   |   |--- element: double (containsNull = true)
    |   |   |   |--- type: string (nullable = true)
    |   |--- country: string (nullable = true)
    |   |--- country_code: string (nullable = true)
    |--- full_name: string (nullable = true)
    |--- id: string (nullable = true)
    |--- name: string (nullable = true)
    |--- place_type: string (nullable = true)
    |--- url: string (nullable = true)
    |--- possibly_sensitive: boolean (nullable = true)
    |--- quote_count: long (nullable = true)
    |--- quoted_status: struct (nullable = true)
    |   |--- contributors: string (nullable = true)
    |--- coordinates: struct (nullable = true)
    |   |--- coordinates: array (nullable = true)
    |   |   |--- element: double (containsNull = true)
    |   |--- type: string (nullable = true)
    |--- created_at: string (nullable = true)
    |--- display_text_range: array (nullable = true)
    |   |--- element: long (containsNull = true)
    |--- entities: struct (nullable = true)
    |   |--- hashtags: array (nullable = true)
    |   |   |--- element: struct (containsNull = true)
    |   |   |   |--- indices: array (nullable = true)
    |   |   |   |   |--- element: long (containsNull = true)
    |   |   |   |--- text: string (nullable = true)
    |--- media: array (nullable = true)
    |   |--- element: struct (containsNull = true)
    |   |   |--- description: string (nullable = true)
    |   |   |--- display_url: string (nullable = true)
    |   |   |--- expanded_url: string (nullable = true)
    |   |--- id: long (nullable = true)
    |   |--- id_str: string (nullable = true)
    |--- indices: array (nullable = true)
    |   |--- element: long (containsNull = true)
    |--- media_url: string (nullable = true)
    |--- media_url_https: string (nullable = true)
    |--- sizes: struct (nullable = true)
    |   |--- large: struct (nullable = true)
    |   |   |--- h: long (nullable = true)
    |   |   |--- resize: string (nullable = true)
    |   |   |--- w: long (nullable = true)
    |   |--- medium: struct (nullable = true)
    |   |   |--- h: long (nullable = true)
    |   |   |--- resize: string (nullable = true)
    |   |   |--- w: long (nullable = true)
    |   |--- small: struct (nullable = true)
    |   |   |--- h: long (nullable = true)
    |   |   |--- resize: string (nullable = true)
    |   |   |--- w: long (nullable = true)
    |--- thumb: struct (nullable = true)
    |   |--- h: long (nullable = true)
    |   |--- resize: string (nullable = true)
    |   |--- w: long (nullable = true)
    |--- source_status_id: long (nullable = true)

```

```

    |-- source_status_id: long (nullable = true)
    |-- source_user_id: string (nullable = true)
    |-- source_user_id_str: string (nullable = true)
    |-- type: string (nullable = true)
    |-- url: string (nullable = true)
    |-- symbols: array (nullable = true)
        |-- element: struct (containsNull = true)
            |-- indices: array (nullable = true)
                |-- element: long (containsNull = true)
            |-- text: string (nullable = true)
    |-- urls: array (nullable = true)
        |-- element: struct (containsNull = true)
            |-- display_url: string (nullable = true)
            |-- expanded_url: string (nullable = true)
            |-- indices: array (nullable = true)
                |-- element: long (containsNull = true)
            |-- url: string (nullable = true)
    |-- user_mentions: array (nullable = true)
        |-- element: struct (containsNull = true)
            |-- id: long (nullable = true)
            |-- id_str: string (nullable = true)
            |-- indices: array (nullable = true)
                |-- element: long (containsNull = true)
            |-- name: string (nullable = true)
            |-- screen_name: string (nullable = true)
    -- extended_entities: struct (nullable = true)
    |-- media: array (nullable = true)
        |-- element: struct (containsNull = true)
            |-- description: string (nullable = true)
            |-- display_url: string (nullable = true)
            |-- expanded_url: string (nullable = true)
            |-- id: long (nullable = true)
            |-- id_str: string (nullable = true)
            |-- indices: array (nullable = true)
                |-- element: long (containsNull = true)
            |-- media_url: string (nullable = true)
            |-- media_url_https: string (nullable = true)
            |-- sizes: struct (nullable = true)
                |-- large: struct (nullable = true)
                    |-- h: long (nullable = true)
                    |-- resize: string (nullable = true)
                    |-- w: long (nullable = true)
                |-- medium: struct (nullable = true)
                    |-- h: long (nullable = true)
                    |-- resize: string (nullable = true)
                    |-- w: long (nullable = true)
                |-- small: struct (nullable = true)
                    |-- h: long (nullable = true)
                    |-- resize: string (nullable = true)
                    |-- w: long (nullable = true)
                |-- thumb: struct (nullable = true)
                    |-- h: long (nullable = true)
                    |-- resize: string (nullable = true)
                    |-- w: long (nullable = true)
            |-- source_status_id: long (nullable = true)
            |-- source_status_id_str: string (nullable = true)
            |-- source_user_id: long (nullable = true)
            |-- source_user_id_str: string (nullable = true)
            |-- type: string (nullable = true)
            |-- url: string (nullable = true)
            |-- video_info: struct (nullable = true)
                |-- aspect_ratio: array (nullable = true)
                    |-- element: long (containsNull = true)
                |-- duration_millis: long (nullable = true)
                |-- variants: array (nullable = true)
                    |-- element: struct (containsNull = true)
                        |-- bitrate: long (nullable = true)
                        |-- content_type: string (nullable = true)
                        |-- url: string (nullable = true)
    -- extended_tweet: struct (nullable = true)

```

```

|-- display_text_range: array (nullable = true)
|   |-- element: long (containsNull = true)
|-- entities: struct (nullable = true)
|   |-- hashtags: array (nullable = true)
|       |-- element: struct (containsNull = true)
|           |-- indices: array (nullable = true)
|               |-- element: long (containsNull = true)
|               |-- text: string (nullable = true)
|   |-- media: array (nullable = true)
|       |-- element: struct (containsNull = true)
|           |-- display_url: string (nullable = true)
|           |-- expanded_url: string (nullable = true)
|           |-- id: long (nullable = true)
|           |-- id_str: string (nullable = true)
|           |-- indices: array (nullable = true)
|               |-- element: long (containsNull = true)
|           |-- media_url: string (nullable = true)
|           |-- media_url_https: string (nullable = true)
|           |-- sizes: struct (nullable = true)
|               |-- large: struct (nullable = true)
|                   |-- h: long (nullable = true)
|                   |-- resize: string (nullable = true)
|                   |-- w: long (nullable = true)
|               |-- medium: struct (nullable = true)
|                   |-- h: long (nullable = true)
|                   |-- resize: string (nullable = true)
|                   |-- w: long (nullable = true)
|               |-- small: struct (nullable = true)
|                   |-- h: long (nullable = true)
|                   |-- resize: string (nullable = true)
|                   |-- w: long (nullable = true)
|               |-- thumb: struct (nullable = true)
|                   |-- h: long (nullable = true)
|                   |-- resize: string (nullable = true)
|                   |-- w: long (nullable = true)
|   |-- source_status_id: long (nullable = true)
|   |-- source_status_id_str: string (nullable = true)
|   |-- source_user_id: long (nullable = true)
|   |-- source_user_id_str: string (nullable = true)
|   |-- type: string (nullable = true)
|   |-- url: string (nullable = true)
|   |-- video_info: struct (nullable = true)
|       |-- aspect_ratio: array (nullable = true)
|           |-- element: long (containsNull = true)
|       |-- duration_millis: long (nullable = true)
|       |-- variants: array (nullable = true)
|           |-- element: struct (containsNull = true)
|               |-- bitrate: long (nullable = true)
|               |-- content_type: string (nullable = true)
|               |-- url: string (nullable = true)
|-- symbols: array (nullable = true)
|   |-- element: struct (containsNull = true)
|       |-- indices: array (nullable = true)
|           |-- element: long (containsNull = true)
|           |-- text: string (nullable = true)
|-- urls: array (nullable = true)
|   |-- element: struct (containsNull = true)
|       |-- display_url: string (nullable = true)
|       |-- expanded_url: string (nullable = true)
|       |-- indices: array (nullable = true)
|           |-- element: long (containsNull = true)
|           |-- url: string (nullable = true)
|-- user_mentions: array (nullable = true)
|   |-- element: struct (containsNull = true)
|       |-- id: long (nullable = true)
|       |-- id_str: string (nullable = true)
|       |-- indices: array (nullable = true)
|           |-- element: long (containsNull = true)
|       |-- name: string (nullable = true)
|       |-- screen_name: string (nullable = true)

```

```

-- extended_entities: struct (nullable = true)
|--- media: array (nullable = true)
|--- |--- element: struct (containsNull = true)
|--- |--- |--- display_url: string (nullable = true)
|--- |--- |--- expanded_url: string (nullable = true)
|--- |--- |--- id: long (nullable = true)
|--- |--- |--- id_str: string (nullable = true)
|--- |--- |--- indices: array (nullable = true)
|--- |--- |--- |--- element: long (containsNull = true)
|--- |--- |--- media_url: string (nullable = true)
|--- |--- |--- media_url_https: string (nullable = true)
|--- |--- sizes: struct (nullable = true)
|--- |--- |--- large: struct (nullable = true)
|--- |--- |--- |--- h: long (nullable = true)
|--- |--- |--- |--- resize: string (nullable = true)
|--- |--- |--- |--- w: long (nullable = true)
|--- |--- |--- medium: struct (nullable = true)
|--- |--- |--- |--- h: long (nullable = true)
|--- |--- |--- |--- resize: string (nullable = true)
|--- |--- |--- |--- w: long (nullable = true)
|--- |--- |--- small: struct (nullable = true)
|--- |--- |--- |--- h: long (nullable = true)
|--- |--- |--- |--- resize: string (nullable = true)
|--- |--- |--- |--- w: long (nullable = true)
|--- |--- |--- thumb: struct (nullable = true)
|--- |--- |--- |--- h: long (nullable = true)
|--- |--- |--- |--- resize: string (nullable = true)
|--- |--- |--- |--- w: long (nullable = true)
|--- source_status_id: long (nullable = true)
|--- source_status_id_str: string (nullable = true)
|--- source_user_id: long (nullable = true)
|--- source_user_id_str: string (nullable = true)
|--- type: string (nullable = true)
|--- url: string (nullable = true)
|--- video_info: struct (nullable = true)
|--- |--- aspect_ratio: array (nullable = true)
|--- |--- |--- element: long (containsNull = true)
|--- |--- duration_millis: long (nullable = true)
|--- |--- variants: array (nullable = true)
|--- |--- |--- element: struct (containsNull = true)
|--- |--- |--- |--- bitrate: long (nullable = true)
|--- |--- |--- |--- content_type: string (nullable = true)
|--- |--- |--- |--- url: string (nullable = true)
|--- full_text: string (nullable = true)
-- favorite_count: long (nullable = true)
-- favorited: boolean (nullable = true)
-- filter_level: string (nullable = true)
-- geo: struct (nullable = true)
|--- coordinates: array (nullable = true)
|--- |--- element: double (containsNull = true)
|--- type: string (nullable = true)
-- id: long (nullable = true)
-- id_str: string (nullable = true)
-- in_reply_to_screen_name: string (nullable = true)
-- in_reply_to_status_id: long (nullable = true)
-- in_reply_to_status_id_str: string (nullable = true)
-- in_reply_to_user_id: long (nullable = true)
-- in_reply_to_user_id_str: string (nullable = true)
-- is_quote_status: boolean (nullable = true)
-- lang: string (nullable = true)
-- place: struct (nullable = true)
|--- bounding_box: struct (nullable = true)
|--- |--- coordinates: array (nullable = true)
|--- |--- |--- element: array (containsNull = true)
|--- |--- |--- |--- element: array (containsNull = true)
|--- |--- |--- |--- |--- element: double (containsNull = true)
|--- |--- type: string (nullable = true)
|--- country: string (nullable = true)
|--- country_code: string (nullable = true)
|--- full_name: string (nullable = true)

```

```

    |--- id: string (nullable = true)
    |--- name: string (nullable = true)
    |--- place_type: string (nullable = true)
    |--- url: string (nullable = true)
    --- possibly_sensitive: boolean (nullable = true)
    --- quote_count: long (nullable = true)
    --- quoted_status_id: long (nullable = true)
    --- quoted_status_id_str: string (nullable = true)
    --- reply_count: long (nullable = true)
    --- retweet_count: long (nullable = true)
    ---retweeted: boolean (nullable = true)
    --- source: string (nullable = true)
    --- text: string (nullable = true)
    --- truncated: boolean (nullable = true)
    --- user: struct (nullable = true)
        |--- contributors_enabled: boolean (nullable = true)
        |--- created_at: string (nullable = true)
        |--- default_profile: boolean (nullable = true)
        |--- default_profile_image: boolean (nullable = true)
        |--- description: string (nullable = true)
        |--- favourites_count: long (nullable = true)
        |--- follow_request_sent: string (nullable = true)
        |--- followers_count: long (nullable = true)
        |--- following: string (nullable = true)
        |--- friends_count: long (nullable = true)
        |--- geo_enabled: boolean (nullable = true)
        |--- id: long (nullable = true)
        |--- id_str: string (nullable = true)
        |--- is_translator: boolean (nullable = true)
        |--- lang: string (nullable = true)
        |--- listed_count: long (nullable = true)
        |--- location: string (nullable = true)
        |--- name: string (nullable = true)
        |--- notifications: string (nullable = true)
        |--- profile_background_color: string (nullable = true)
        |--- profile_background_image_url: string (nullable = true)
        |--- profile_background_image_url_https: string (nullable = true)
        |--- profile_background_tile: boolean (nullable = true)
        |--- profile_banner_url: string (nullable = true)
        |--- profile_image_url: string (nullable = true)
        |--- profile_image_url_https: string (nullable = true)
        |--- profile_link_color: string (nullable = true)
        |--- profile_sidebar_border_color: string (nullable = true)
        |--- profile_sidebar_fill_color: string (nullable = true)
        |--- profile_text_color: string (nullable = true)
        |--- profile_use_background_image: boolean (nullable = true)
        |--- protected: boolean (nullable = true)
        |--- screen_name: string (nullable = true)
        |--- statuses_count: long (nullable = true)
        |--- time_zone: string (nullable = true)
        |--- translator_type: string (nullable = true)
        |--- url: string (nullable = true)
        |--- utc_offset: long (nullable = true)
        |--- verified: boolean (nullable = true)
    --- quoted_status_id: long (nullable = true)
    --- quoted_status_id_str: string (nullable = true)
    --- reply_count: long (nullable = true)
    --- retweet_count: long (nullable = true)
    ---retweeted: boolean (nullable = true)
    --- scopes: struct (nullable = true)
        |--- followers: boolean (nullable = true)
    --- source: string (nullable = true)
    --- text: string (nullable = true)
    --- truncated: boolean (nullable = true)
    --- user: struct (nullable = true)
        |--- contributors_enabled: boolean (nullable = true)
        |--- created_at: string (nullable = true)
        |--- default_profile: boolean (nullable = true)
        |--- default_profile_image: boolean (nullable = true)
        |--- description: string (nullable = true)

```

```

    --- favourites_count: long (nullable = true)
    --- follow_request_sent: string (nullable = true)
    --- followers_count: long (nullable = true)
    --- following: string (nullable = true)
    --- friends_count: long (nullable = true)
    --- geo_enabled: boolean (nullable = true)
    --- id: long (nullable = true)
    --- id_str: string (nullable = true)
    --- is_translator: boolean (nullable = true)
    --- lang: string (nullable = true)
    --- listed_count: long (nullable = true)
    --- location: string (nullable = true)
    --- name: string (nullable = true)
    --- notifications: string (nullable = true)
    --- profile_background_color: string (nullable = true)
    --- profile_background_image_url: string (nullable = true)
    --- profile_background_image_url_https: string (nullable = true)
    --- profile_background_tile: boolean (nullable = true)
    --- profile_banner_url: string (nullable = true)
    --- profile_image_url: string (nullable = true)
    --- profile_image_url_https: string (nullable = true)
    --- profile_link_color: string (nullable = true)
    --- profile_sidebar_border_color: string (nullable = true)
    --- profile_sidebar_fill_color: string (nullable = true)
    --- profile_text_color: string (nullable = true)
    --- profile_use_background_image: boolean (nullable = true)
    --- protected: boolean (nullable = true)
    --- screen_name: string (nullable = true)
    --- statuses_count: long (nullable = true)
    --- time_zone: string (nullable = true)
    --- translator_type: string (nullable = true)
    --- url: string (nullable = true)
    --- utc_offset: long (nullable = true)
    --- verified: boolean (nullable = true)
    |--- withheld_in_countries: array (nullable = true)
    |   |--- element: string (containsNull = true)
    --- source: string (nullable = true)
    --- text: string (nullable = true)
    --- timestamp_ms: string (nullable = true)
    --- truncated: boolean (nullable = true)
    --- user: struct (nullable = true)
        --- contributors_enabled: boolean (nullable = true)
        --- created_at: string (nullable = true)
        --- default_profile: boolean (nullable = true)
        --- default_profile_image: boolean (nullable = true)
        --- description: string (nullable = true)
        --- favourites_count: long (nullable = true)
        --- follow_request_sent: string (nullable = true)
        --- followers_count: long (nullable = true)
        --- following: string (nullable = true)
        --- friends_count: long (nullable = true)
        --- geo_enabled: boolean (nullable = true)
        --- id: long (nullable = true)
        --- id_str: string (nullable = true)
        --- is_translator: boolean (nullable = true)
        --- lang: string (nullable = true)
        --- listed_count: long (nullable = true)
        --- location: string (nullable = true)
        --- name: string (nullable = true)
        --- notifications: string (nullable = true)
        --- profile_background_color: string (nullable = true)
        --- profile_background_image_url: string (nullable = true)
        --- profile_background_image_url_https: string (nullable = true)
        --- profile_background_tile: boolean (nullable = true)
        --- profile_banner_url: string (nullable = true)
        --- profile_image_url: string (nullable = true)
        --- profile_image_url_https: string (nullable = true)
        --- profile_link_color: string (nullable = true)
        --- profile_sidebar_border_color: string (nullable = true)
        --- profile_sidebar_fill_color: string (nullable = true)

```

## APÉNDICE A. ESTRUCTURA COMPLETA DE UN TWEET

---

```
|-- profile_text_color: string (nullable = true)
|-- profile_use_background_image: boolean (nullable = true)
|-- protected: boolean (nullable = true)
|-- screen_name: string (nullable = true)
|-- statuses_count: long (nullable = true)
|-- time_zone: string (nullable = true)
|-- translator_type: string (nullable = true)
|-- url: string (nullable = true)
|-- utc_offset: long (nullable = true)
|-- verified: boolean (nullable = true)
|-- withheld_in_countries: array (nullable = true)
|   |-- element: string (containsNull = true)
```

---

# B

## Estructura completa de un evento de GitHub

Código B.1: Esquema completo de la estructura de GitHub.

```
root
|--- actor: struct (nullable = true)
|   |--- avatar_url: string (nullable = true)
|   |--- display_login: string (nullable = true)
|   |--- gravatar_id: string (nullable = true)
|   |--- id: long (nullable = true)
|   |--- login: string (nullable = true)
|   |--- url: string (nullable = true)
|--- created_at: string (nullable = true)
|--- id: string (nullable = true)
|--- org: struct (nullable = true)
|   |--- avatar_url: string (nullable = true)
|   |--- gravatar_id: string (nullable = true)
|   |--- id: long (nullable = true)
|   |--- login: string (nullable = true)
|   |--- url: string (nullable = true)
|--- payload: struct (nullable = true)
|   |--- action: string (nullable = true)
|   |--- before: string (nullable = true)
|   |--- comment: struct (nullable = true)
|       |--- links: struct (nullable = true)
|           |--- html: struct (nullable = true)
|               |--- href: string (nullable = true)
|           |--- pull_request: struct (nullable = true)
|               |--- href: string (nullable = true)
|           |--- self: struct (nullable = true)
|               |--- href: string (nullable = true)
|   |--- body: string (nullable = true)
|   |--- commit_id: string (nullable = true)
|   |--- created_at: string (nullable = true)
|   |--- diff_hunk: string (nullable = true)
|   |--- html_url: string (nullable = true)
|   |--- id: long (nullable = true)
|   |--- issue_url: string (nullable = true)
|   |--- line: long (nullable = true)
|   |--- original_commit_id: string (nullable = true)
|   |--- original_position: long (nullable = true)
|   |--- path: string (nullable = true)
|   |--- position: long (nullable = true)
```

```

--- pull_request_review_id: long (nullable = true)
--- pull_request_url: string (nullable = true)
--- updated_at: string (nullable = true)
--- url: string (nullable = true)
--- user: struct (nullable = true)
    |--- avatar_url: string (nullable = true)
    |--- events_url: string (nullable = true)
    |--- followers_url: string (nullable = true)
    |--- following_url: string (nullable = true)
    |--- gists_url: string (nullable = true)
    |--- gravatar_id: string (nullable = true)
    |--- html_url: string (nullable = true)
    |--- id: long (nullable = true)
    |--- login: string (nullable = true)
    |--- organizations_url: string (nullable = true)
    |--- received_events_url: string (nullable = true)
    |--- repos_url: string (nullable = true)
    |--- site_admin: boolean (nullable = true)
    |--- starred_url: string (nullable = true)
    |--- subscriptions_url: string (nullable = true)
    |--- type: string (nullable = true)
    |--- url: string (nullable = true)
--- commits: array (nullable = true)
    |--- element: struct (containsNull = true)
        |--- author: struct (nullable = true)
            |--- email: string (nullable = true)
            |--- name: string (nullable = true)
        |--- distinct: boolean (nullable = true)
        |--- message: string (nullable = true)
        |--- sha: string (nullable = true)
        |--- url: string (nullable = true)
--- description: string (nullable = true)
--- distinct_size: long (nullable = true)
--- forkee: struct (nullable = true)
    |--- archive_url: string (nullable = true)
    |--- assignees_url: string (nullable = true)
    |--- blobs_url: string (nullable = true)
    |--- branches_url: string (nullable = true)
    |--- clone_url: string (nullable = true)
    |--- collaborators_url: string (nullable = true)
    |--- comments_url: string (nullable = true)
    |--- commits_url: string (nullable = true)
    |--- compare_url: string (nullable = true)
    |--- contents_url: string (nullable = true)
    |--- contributors_url: string (nullable = true)
    |--- created_at: string (nullable = true)
    |--- default_branch: string (nullable = true)
    |--- deployments_url: string (nullable = true)
    |--- description: string (nullable = true)
    |--- downloads_url: string (nullable = true)
    |--- events_url: string (nullable = true)
    |--- fork: boolean (nullable = true)
    |--- forks: long (nullable = true)
    |--- forks_count: long (nullable = true)
    |--- forks_url: string (nullable = true)
    |--- full_name: string (nullable = true)
    |--- git_commits_url: string (nullable = true)
    |--- git_refs_url: string (nullable = true)
    |--- git_tags_url: string (nullable = true)
    |--- git_url: string (nullable = true)
    |--- has_downloads: boolean (nullable = true)
    |--- has_issues: boolean (nullable = true)
    |--- has_pages: boolean (nullable = true)
    |--- has_wiki: boolean (nullable = true)
    |--- homepage: string (nullable = true)
    |--- hooks_url: string (nullable = true)
    |--- html_url: string (nullable = true)
    |--- id: long (nullable = true)
    |--- issue_comment_url: string (nullable = true)
    |--- issue_events_url: string (nullable = true)

```

```

    --- issues_url: string (nullable = true)
    --- keys_url: string (nullable = true)
    --- labels_url: string (nullable = true)
    --- language: string (nullable = true)
    --- languages_url: string (nullable = true)
    --- merges_url: string (nullable = true)
    --- milestones_url: string (nullable = true)
    --- mirror_url: string (nullable = true)
    --- name: string (nullable = true)
    --- notifications_url: string (nullable = true)
    --- open_issues: long (nullable = true)
    --- open_issues_count: long (nullable = true)
    --- owner: struct (nullable = true)
        |--- avatar_url: string (nullable = true)
        |--- events_url: string (nullable = true)
        |--- followers_url: string (nullable = true)
        |--- following_url: string (nullable = true)
        |--- gists_url: string (nullable = true)
        |--- gravatar_id: string (nullable = true)
        |--- html_url: string (nullable = true)
        |--- id: long (nullable = true)
        |--- login: string (nullable = true)
        |--- organizations_url: string (nullable = true)
        |--- received_events_url: string (nullable = true)
        |--- repos_url: string (nullable = true)
        |--- site_admin: boolean (nullable = true)
        |--- starred_url: string (nullable = true)
        |--- subscriptions_url: string (nullable = true)
        |--- type: string (nullable = true)
        |--- url: string (nullable = true)
    --- private: boolean (nullable = true)
    --- public: boolean (nullable = true)
    --- pulls_url: string (nullable = true)
    --- pushed_at: string (nullable = true)
    --- releases_url: string (nullable = true)
    --- size: long (nullable = true)
    --- ssh_url: string (nullable = true)
    --- stargazers_count: long (nullable = true)
    --- stargazers_url: string (nullable = true)
    --- statuses_url: string (nullable = true)
    --- subscribers_url: string (nullable = true)
    --- subscription_url: string (nullable = true)
    --- svn_url: string (nullable = true)
    --- tags_url: string (nullable = true)
    --- teams_url: string (nullable = true)
    --- trees_url: string (nullable = true)
    --- updated_at: string (nullable = true)
    --- url: string (nullable = true)
    --- watchers: long (nullable = true)
    --- watchers_count: long (nullable = true)
    --- head: string (nullable = true)
    --- issue: struct (nullable = true)
        |--- assignee: struct (nullable = true)
            |--- avatar_url: string (nullable = true)
            |--- events_url: string (nullable = true)
            |--- followers_url: string (nullable = true)
            |--- following_url: string (nullable = true)
            |--- gists_url: string (nullable = true)
            |--- gravatar_id: string (nullable = true)
            |--- html_url: string (nullable = true)
            |--- id: long (nullable = true)
            |--- login: string (nullable = true)
            |--- organizations_url: string (nullable = true)
            |--- received_events_url: string (nullable = true)
            |--- repos_url: string (nullable = true)
            |--- site_admin: boolean (nullable = true)
            |--- starred_url: string (nullable = true)
            |--- subscriptions_url: string (nullable = true)
            |--- type: string (nullable = true)
            |--- url: string (nullable = true)

```

```

-- assignees: array (nullable = true)
|--- element: struct (containsNull = true)
|     |--- avatar_url: string (nullable = true)
|     |--- events_url: string (nullable = true)
|     |--- followers_url: string (nullable = true)
|     |--- following_url: string (nullable = true)
|     |--- gists_url: string (nullable = true)
|     |--- gravatar_id: string (nullable = true)
|     |--- html_url: string (nullable = true)
|     |--- id: long (nullable = true)
|     |--- login: string (nullable = true)
|     |--- organizations_url: string (nullable = true)
|     |--- received_events_url: string (nullable = true)
|     |--- repos_url: string (nullable = true)
|     |--- site_admin: boolean (nullable = true)
|     |--- starred_url: string (nullable = true)
|     |--- subscriptions_url: string (nullable = true)
|     |--- type: string (nullable = true)
|     |--- url: string (nullable = true)
-- body: string (nullable = true)
-- closed_at: string (nullable = true)
-- comments: long (nullable = true)
-- comments_url: string (nullable = true)
-- created_at: string (nullable = true)
-- events_url: string (nullable = true)
-- html_url: string (nullable = true)
-- id: long (nullable = true)
-- labels: array (nullable = true)
|--- element: struct (containsNull = true)
|     |--- color: string (nullable = true)
|     |--- default: boolean (nullable = true)
|     |--- id: long (nullable = true)
|     |--- name: string (nullable = true)
|     |--- url: string (nullable = true)
-- labels_url: string (nullable = true)
-- locked: boolean (nullable = true)
-- milestone: struct (nullable = true)
|--- closed_at: string (nullable = true)
|--- closed_issues: long (nullable = true)
|--- created_at: string (nullable = true)
|--- creator: struct (nullable = true)
|     |--- avatar_url: string (nullable = true)
|     |--- events_url: string (nullable = true)
|     |--- followers_url: string (nullable = true)
|     |--- following_url: string (nullable = true)
|     |--- gists_url: string (nullable = true)
|     |--- gravatar_id: string (nullable = true)
|     |--- html_url: string (nullable = true)
|     |--- id: long (nullable = true)
|     |--- login: string (nullable = true)
|     |--- organizations_url: string (nullable = true)
|     |--- received_events_url: string (nullable = true)
|     |--- repos_url: string (nullable = true)
|     |--- site_admin: boolean (nullable = true)
|     |--- starred_url: string (nullable = true)
|     |--- subscriptions_url: string (nullable = true)
|     |--- type: string (nullable = true)
|     |--- url: string (nullable = true)
|--- description: string (nullable = true)
|--- due_on: string (nullable = true)
|--- html_url: string (nullable = true)
|--- id: long (nullable = true)
|--- labels_url: string (nullable = true)
|--- number: long (nullable = true)
|--- open_issues: long (nullable = true)
|--- state: string (nullable = true)
|--- title: string (nullable = true)
|--- updated_at: string (nullable = true)
|--- url: string (nullable = true)
-- number: long (nullable = true)

```

```

-- pull_request: struct (nullable = true)
|--- diff_url: string (nullable = true)
|--- html_url: string (nullable = true)
|--- patch_url: string (nullable = true)
|--- url: string (nullable = true)
-- repository_url: string (nullable = true)
-- state: string (nullable = true)
-- title: string (nullable = true)
-- updated_at: string (nullable = true)
-- url: string (nullable = true)
-- user: struct (nullable = true)
|--- avatar_url: string (nullable = true)
|--- events_url: string (nullable = true)
|--- followers_url: string (nullable = true)
|--- following_url: string (nullable = true)
|--- gists_url: string (nullable = true)
|--- gravatar_id: string (nullable = true)
|--- html_url: string (nullable = true)
|--- id: long (nullable = true)
|--- login: string (nullable = true)
|--- organizations_url: string (nullable = true)
|--- received_events_url: string (nullable = true)
|--- repos_url: string (nullable = true)
|--- site_admin: boolean (nullable = true)
|--- starred_url: string (nullable = true)
|--- subscriptions_url: string (nullable = true)
|--- type: string (nullable = true)
|--- url: string (nullable = true)
-- master_branch: string (nullable = true)
-- member: struct (nullable = true)
|--- avatar_url: string (nullable = true)
|--- events_url: string (nullable = true)
|--- followers_url: string (nullable = true)
|--- following_url: string (nullable = true)
|--- gists_url: string (nullable = true)
|--- gravatar_id: string (nullable = true)
|--- html_url: string (nullable = true)
|--- id: long (nullable = true)
|--- login: string (nullable = true)
|--- organizations_url: string (nullable = true)
|--- received_events_url: string (nullable = true)
|--- repos_url: string (nullable = true)
|--- site_admin: boolean (nullable = true)
|--- starred_url: string (nullable = true)
|--- subscriptions_url: string (nullable = true)
|--- type: string (nullable = true)
|--- url: string (nullable = true)
-- number: long (nullable = true)
-- pages: array (nullable = true)
|--- element: struct (containsNull = true)
|--- action: string (nullable = true)
|--- html_url: string (nullable = true)
|--- page_name: string (nullable = true)
|--- sha: string (nullable = true)
|--- summary: string (nullable = true)
|--- title: string (nullable = true)
-- pull_request: struct (nullable = true)
|--- links: struct (nullable = true)
|--- |--- comments: struct (nullable = true)
|--- |--- |--- href: string (nullable = true)
|--- |--- commits: struct (nullable = true)
|--- |--- |--- href: string (nullable = true)
|--- |--- html: struct (nullable = true)
|--- |--- |--- href: string (nullable = true)
|--- |--- issue: struct (nullable = true)
|--- |--- |--- href: string (nullable = true)
|--- |--- review_comment: struct (nullable = true)
|--- |--- |--- href: string (nullable = true)
|--- |--- review_comments: struct (nullable = true)
|--- |--- |--- href: string (nullable = true)

```

```

    |-- self: struct (nullable = true)
    |   |-- href: string (nullable = true)
    |-- statuses: struct (nullable = true)
    |   |-- href: string (nullable = true)
    -- additions: long (nullable = true)
    -- assignee: struct (nullable = true)
        |-- avatar_url: string (nullable = true)
        |-- events_url: string (nullable = true)
        |-- followers_url: string (nullable = true)
        |-- following_url: string (nullable = true)
        |-- gists_url: string (nullable = true)
        |-- gravatar_id: string (nullable = true)
        |-- html_url: string (nullable = true)
        |-- id: long (nullable = true)
        |-- login: string (nullable = true)
        |-- organizations_url: string (nullable = true)
        |-- received_events_url: string (nullable = true)
        |-- repos_url: string (nullable = true)
        |-- site_admin: boolean (nullable = true)
        |-- starred_url: string (nullable = true)
        |-- subscriptions_url: string (nullable = true)
        |-- type: string (nullable = true)
        |-- url: string (nullable = true)
    -- assignees: array (nullable = true)
        |-- element: struct (containsNull = true)
            |-- avatar_url: string (nullable = true)
            |-- events_url: string (nullable = true)
            |-- followers_url: string (nullable = true)
            |-- following_url: string (nullable = true)
            |-- gists_url: string (nullable = true)
            |-- gravatar_id: string (nullable = true)
            |-- html_url: string (nullable = true)
            |-- id: long (nullable = true)
            |-- login: string (nullable = true)
            |-- organizations_url: string (nullable = true)
            |-- received_events_url: string (nullable = true)
            |-- repos_url: string (nullable = true)
            |-- site_admin: boolean (nullable = true)
            |-- starred_url: string (nullable = true)
            |-- subscriptions_url: string (nullable = true)
            |-- type: string (nullable = true)
            |-- url: string (nullable = true)
    -- base: struct (nullable = true)
        |-- label: string (nullable = true)
        |-- ref: string (nullable = true)
        |-- repo: struct (nullable = true)
            |-- archive_url: string (nullable = true)
            |-- assignees_url: string (nullable = true)
            |-- blobs_url: string (nullable = true)
            |-- branches_url: string (nullable = true)
            |-- clone_url: string (nullable = true)
            |-- collaborators_url: string (nullable = true)
            |-- comments_url: string (nullable = true)
            |-- commits_url: string (nullable = true)
            |-- compare_url: string (nullable = true)
            |-- contents_url: string (nullable = true)
            |-- contributors_url: string (nullable = true)
            |-- created_at: string (nullable = true)
            |-- default_branch: string (nullable = true)
            |-- deployments_url: string (nullable = true)
            |-- description: string (nullable = true)
            |-- downloads_url: string (nullable = true)
            |-- events_url: string (nullable = true)
            |-- fork: boolean (nullable = true)
            |-- forks: long (nullable = true)
            |-- forks_count: long (nullable = true)
            |-- forks_url: string (nullable = true)
            |-- full_name: string (nullable = true)
            |-- git_commits_url: string (nullable = true)
            |-- git_refs_url: string (nullable = true)

```

```
-- git_tags_url: string (nullable = true)
-- git_url: string (nullable = true)
-- has_downloads: boolean (nullable = true)
-- has_issues: boolean (nullable = true)
-- has_pages: boolean (nullable = true)
-- has_wiki: boolean (nullable = true)
-- homepage: string (nullable = true)
-- hooks_url: string (nullable = true)
-- html_url: string (nullable = true)
-- id: long (nullable = true)
-- issue_comment_url: string (nullable = true)
-- issue_events_url: string (nullable = true)
-- issues_url: string (nullable = true)
-- keys_url: string (nullable = true)
-- labels_url: string (nullable = true)
-- language: string (nullable = true)
-- languages_url: string (nullable = true)
-- merges_url: string (nullable = true)
-- milestones_url: string (nullable = true)
-- mirror_url: string (nullable = true)
-- name: string (nullable = true)
-- notifications_url: string (nullable = true)
-- open_issues: long (nullable = true)
-- open_issues_count: long (nullable = true)
-- owner: struct (nullable = true)
  -- avatar_url: string (nullable = true)
  -- events_url: string (nullable = true)
  -- followers_url: string (nullable = true)
  -- following_url: string (nullable = true)
  -- gists_url: string (nullable = true)
  -- gravatar_id: string (nullable = true)
  -- html_url: string (nullable = true)
  -- id: long (nullable = true)
  -- login: string (nullable = true)
  -- organizations_url: string (nullable = true)
  -- received_events_url: string (nullable = true)
  -- repos_url: string (nullable = true)
  -- site_admin: boolean (nullable = true)
  -- starred_url: string (nullable = true)
  -- subscriptions_url: string (nullable = true)
  -- type: string (nullable = true)
  -- url: string (nullable = true)
-- private: boolean (nullable = true)
-- pulls_url: string (nullable = true)
-- pushed_at: string (nullable = true)
-- releases_url: string (nullable = true)
-- size: long (nullable = true)
-- ssh_url: string (nullable = true)
-- stargazers_count: long (nullable = true)
-- stargazers_url: string (nullable = true)
-- statuses_url: string (nullable = true)
-- subscribers_url: string (nullable = true)
-- subscription_url: string (nullable = true)
-- svn_url: string (nullable = true)
-- tags_url: string (nullable = true)
-- teams_url: string (nullable = true)
-- trees_url: string (nullable = true)
-- updated_at: string (nullable = true)
-- url: string (nullable = true)
-- watchers: long (nullable = true)
-- watchers_count: long (nullable = true)
-- sha: string (nullable = true)
-- user: struct (nullable = true)
  -- avatar_url: string (nullable = true)
  -- events_url: string (nullable = true)
  -- followers_url: string (nullable = true)
  -- following_url: string (nullable = true)
  -- gists_url: string (nullable = true)
  -- gravatar_id: string (nullable = true)
  -- html_url: string (nullable = true)
```

```

    |--- id: long (nullable = true)
    |--- login: string (nullable = true)
    |--- organizations_url: string (nullable = true)
    |--- received_events_url: string (nullable = true)
    |--- repos_url: string (nullable = true)
    |--- site_admin: boolean (nullable = true)
    |--- starred_url: string (nullable = true)
    |--- subscriptions_url: string (nullable = true)
    |--- type: string (nullable = true)
    |--- url: string (nullable = true)
--- body: string (nullable = true)
--- changed_files: long (nullable = true)
--- closed_at: string (nullable = true)
--- comments: long (nullable = true)
--- comments_url: string (nullable = true)
--- commits: long (nullable = true)
--- commits_url: string (nullable = true)
--- created_at: string (nullable = true)
--- deletions: long (nullable = true)
--- diff_url: string (nullable = true)
--- head: struct (nullable = true)
    |--- label: string (nullable = true)
    |--- ref: string (nullable = true)
    |--- repo: struct (nullable = true)
        |--- archive_url: string (nullable = true)
        |--- assignees_url: string (nullable = true)
        |--- blobs_url: string (nullable = true)
        |--- branches_url: string (nullable = true)
        |--- clone_url: string (nullable = true)
        |--- collaborators_url: string (nullable = true)
        |--- comments_url: string (nullable = true)
        |--- commits_url: string (nullable = true)
        |--- compare_url: string (nullable = true)
        |--- contents_url: string (nullable = true)
        |--- contributors_url: string (nullable = true)
        |--- created_at: string (nullable = true)
        |--- default_branch: string (nullable = true)
        |--- deployments_url: string (nullable = true)
        |--- description: string (nullable = true)
        |--- downloads_url: string (nullable = true)
        |--- events_url: string (nullable = true)
        |--- fork: boolean (nullable = true)
        |--- forks: long (nullable = true)
        |--- forks_count: long (nullable = true)
        |--- forks_url: string (nullable = true)
        |--- full_name: string (nullable = true)
        |--- git_commits_url: string (nullable = true)
        |--- git_refs_url: string (nullable = true)
        |--- git_tags_url: string (nullable = true)
        |--- git_url: string (nullable = true)
        |--- has_downloads: boolean (nullable = true)
        |--- has_issues: boolean (nullable = true)
        |--- has_pages: boolean (nullable = true)
        |--- has_wiki: boolean (nullable = true)
        |--- homepage: string (nullable = true)
        |--- hooks_url: string (nullable = true)
        |--- html_url: string (nullable = true)
        |--- id: long (nullable = true)
        |--- issue_comment_url: string (nullable = true)
        |--- issue_events_url: string (nullable = true)
        |--- issues_url: string (nullable = true)
        |--- keys_url: string (nullable = true)
        |--- labels_url: string (nullable = true)
        |--- language: string (nullable = true)
        |--- languages_url: string (nullable = true)
        |--- merges_url: string (nullable = true)
        |--- milestones_url: string (nullable = true)
        |--- mirror_url: string (nullable = true)
        |--- name: string (nullable = true)
        |--- notifications_url: string (nullable = true)

```

```

    |-- open_issues: long (nullable = true)
    |-- open_issues_count: long (nullable = true)
    |-- owner: struct (nullable = true)
    |   |-- avatar_url: string (nullable = true)
    |   |-- events_url: string (nullable = true)
    |   |-- followers_url: string (nullable = true)
    |   |-- following_url: string (nullable = true)
    |   |-- gists_url: string (nullable = true)
    |   |-- gravatar_id: string (nullable = true)
    |   |-- html_url: string (nullable = true)
    |   |-- id: long (nullable = true)
    |   |-- login: string (nullable = true)
    |   |-- organizations_url: string (nullable = true)
    |   |-- received_events_url: string (nullable = true)
    |   |-- repos_url: string (nullable = true)
    |   |-- site_admin: boolean (nullable = true)
    |   |-- starred_url: string (nullable = true)
    |   |-- subscriptions_url: string (nullable = true)
    |   |-- type: string (nullable = true)
    |   |-- url: string (nullable = true)
    |-- private: boolean (nullable = true)
    |-- pulls_url: string (nullable = true)
    |-- pushed_at: string (nullable = true)
    |-- releases_url: string (nullable = true)
    |-- size: long (nullable = true)
    |-- ssh_url: string (nullable = true)
    |-- stargazers_count: long (nullable = true)
    |-- stargazers_url: string (nullable = true)
    |-- statuses_url: string (nullable = true)
    |-- subscribers_url: string (nullable = true)
    |-- subscription_url: string (nullable = true)
    |-- svn_url: string (nullable = true)
    |-- tags_url: string (nullable = true)
    |-- teams_url: string (nullable = true)
    |-- trees_url: string (nullable = true)
    |-- updated_at: string (nullable = true)
    |-- url: string (nullable = true)
    |-- watchers: long (nullable = true)
    |-- watchers_count: long (nullable = true)
    |-- sha: string (nullable = true)
    |-- user: struct (nullable = true)
    |   |-- avatar_url: string (nullable = true)
    |   |-- events_url: string (nullable = true)
    |   |-- followers_url: string (nullable = true)
    |   |-- following_url: string (nullable = true)
    |   |-- gists_url: string (nullable = true)
    |   |-- gravatar_id: string (nullable = true)
    |   |-- html_url: string (nullable = true)
    |   |-- id: long (nullable = true)
    |   |-- login: string (nullable = true)
    |   |-- organizations_url: string (nullable = true)
    |   |-- received_events_url: string (nullable = true)
    |   |-- repos_url: string (nullable = true)
    |   |-- site_admin: boolean (nullable = true)
    |   |-- starred_url: string (nullable = true)
    |   |-- subscriptions_url: string (nullable = true)
    |   |-- type: string (nullable = true)
    |   |-- url: string (nullable = true)
    |-- html_url: string (nullable = true)
    |-- id: long (nullable = true)
    |-- issue_url: string (nullable = true)
    |-- locked: boolean (nullable = true)
    |-- maintainer_can_modify: boolean (nullable = true)
    |-- merge_commit_sha: string (nullable = true)
    |-- mergeable: boolean (nullable = true)
    |-- mergeable_state: string (nullable = true)
    |-- merged: boolean (nullable = true)
    |-- merged_at: string (nullable = true)
    |-- merged_by: struct (nullable = true)
    |   |-- avatar_url: string (nullable = true)

```

```
    |-- events_url: string (nullable = true)
    |-- followers_url: string (nullable = true)
    |-- following_url: string (nullable = true)
    |-- gists_url: string (nullable = true)
    |-- gravatar_id: string (nullable = true)
    |-- html_url: string (nullable = true)
    |-- id: long (nullable = true)
    |-- login: string (nullable = true)
    |-- organizations_url: string (nullable = true)
    |-- received_events_url: string (nullable = true)
    |-- repos_url: string (nullable = true)
    |-- site_admin: boolean (nullable = true)
    |-- starred_url: string (nullable = true)
    |-- subscriptions_url: string (nullable = true)
    |-- type: string (nullable = true)
    |-- url: string (nullable = true)
-- milestone: struct (nullable = true)
    |-- closed_at: string (nullable = true)
    |-- closed_issues: long (nullable = true)
    |-- created_at: string (nullable = true)
    |-- creator: struct (nullable = true)
        |-- avatar_url: string (nullable = true)
        |-- events_url: string (nullable = true)
        |-- followers_url: string (nullable = true)
        |-- following_url: string (nullable = true)
        |-- gists_url: string (nullable = true)
        |-- gravatar_id: string (nullable = true)
        |-- html_url: string (nullable = true)
        |-- id: long (nullable = true)
        |-- login: string (nullable = true)
        |-- organizations_url: string (nullable = true)
        |-- received_events_url: string (nullable = true)
        |-- repos_url: string (nullable = true)
        |-- site_admin: boolean (nullable = true)
        |-- starred_url: string (nullable = true)
        |-- subscriptions_url: string (nullable = true)
        |-- type: string (nullable = true)
        |-- url: string (nullable = true)
    |-- description: string (nullable = true)
    |-- due_on: string (nullable = true)
    |-- html_url: string (nullable = true)
    |-- id: long (nullable = true)
    |-- labels_url: string (nullable = true)
    |-- number: long (nullable = true)
    |-- open_issues: long (nullable = true)
    |-- state: string (nullable = true)
    |-- title: string (nullable = true)
    |-- updated_at: string (nullable = true)
    |-- url: string (nullable = true)
-- number: long (nullable = true)
-- patch_url: string (nullable = true)
-- review_comment_url: string (nullable = true)
-- review_comments: long (nullable = true)
-- review_comments_url: string (nullable = true)
-- state: string (nullable = true)
-- statuses_url: string (nullable = true)
-- title: string (nullable = true)
-- updated_at: string (nullable = true)
-- url: string (nullable = true)
-- user: struct (nullable = true)
    |-- avatar_url: string (nullable = true)
    |-- events_url: string (nullable = true)
    |-- followers_url: string (nullable = true)
    |-- following_url: string (nullable = true)
    |-- gists_url: string (nullable = true)
    |-- gravatar_id: string (nullable = true)
    |-- html_url: string (nullable = true)
    |-- id: long (nullable = true)
    |-- login: string (nullable = true)
    |-- organizations_url: string (nullable = true)
```

```

    |--- received_events_url: string (nullable = true)
    |--- repos_url: string (nullable = true)
    |--- site_admin: boolean (nullable = true)
    |--- starred_url: string (nullable = true)
    |--- subscriptions_url: string (nullable = true)
    |--- type: string (nullable = true)
    |--- url: string (nullable = true)
-- push_id: long (nullable = true)
-- pusher_type: string (nullable = true)
-- ref: string (nullable = true)
-- ref_type: string (nullable = true)
-- release: struct (nullable = true)
|--- assets: array (nullable = true)
|--- element: struct (containsNull = true)
|--- browser_download_url: string (nullable = true)
|--- content_type: string (nullable = true)
|--- created_at: string (nullable = true)
|--- download_count: long (nullable = true)
|--- id: long (nullable = true)
|--- label: string (nullable = true)
|--- name: string (nullable = true)
|--- size: long (nullable = true)
|--- state: string (nullable = true)
|--- updated_at: string (nullable = true)
|--- uploader: struct (nullable = true)
|--- avatar_url: string (nullable = true)
|--- events_url: string (nullable = true)
|--- followers_url: string (nullable = true)
|--- following_url: string (nullable = true)
|--- gists_url: string (nullable = true)
|--- gravatar_id: string (nullable = true)
|--- html_url: string (nullable = true)
|--- id: long (nullable = true)
|--- login: string (nullable = true)
|--- organizations_url: string (nullable = true)
|--- received_events_url: string (nullable = true)
|--- repos_url: string (nullable = true)
|--- site_admin: boolean (nullable = true)
|--- starred_url: string (nullable = true)
|--- subscriptions_url: string (nullable = true)
|--- type: string (nullable = true)
|--- url: string (nullable = true)
|--- url: string (nullable = true)
-- assets_url: string (nullable = true)
-- author: struct (nullable = true)
|--- avatar_url: string (nullable = true)
|--- events_url: string (nullable = true)
|--- followers_url: string (nullable = true)
|--- following_url: string (nullable = true)
|--- gists_url: string (nullable = true)
|--- gravatar_id: string (nullable = true)
|--- html_url: string (nullable = true)
|--- id: long (nullable = true)
|--- login: string (nullable = true)
|--- organizations_url: string (nullable = true)
|--- received_events_url: string (nullable = true)
|--- repos_url: string (nullable = true)
|--- site_admin: boolean (nullable = true)
|--- starred_url: string (nullable = true)
|--- subscriptions_url: string (nullable = true)
|--- type: string (nullable = true)
|--- url: string (nullable = true)
-- body: string (nullable = true)
-- created_at: string (nullable = true)
-- draft: boolean (nullable = true)
-- html_url: string (nullable = true)
-- id: long (nullable = true)
-- name: string (nullable = true)
-- prerelease: boolean (nullable = true)
-- published_at: string (nullable = true)

```

## APÉNDICE B. ESTRUCTURA COMPLETA DE UN EVENTO DE GITHUB

---

```
|      |      |-- tag_name: string (nullable = true)
|      |      |-- tarball_url: string (nullable = true)
|      |      |-- target_commitish: string (nullable = true)
|      |      |-- upload_url: string (nullable = true)
|      |      |-- url: string (nullable = true)
|      |      |-- zipball_url: string (nullable = true)
|      |      |-- size: long (nullable = true)
|      |-- public: boolean (nullable = true)
|      |-- repo: struct (nullable = true)
|          |-- id: long (nullable = true)
|          |-- name: string (nullable = true)
|          |-- url: string (nullable = true)
|      |-- type: string (nullable = true)
```

---

# C

## Fichero de configuración “.bashrc” completo

Código C.1: Fichero “.bashrc” completo del sistema.

---

```
# Java
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export PATH=$PATH:$JAVA_HOME/bin
export PATH=$PATH:$JAVA_HOME

# Hadoop
export HADOOP_HOME=/opt/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin

export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME

export HADOOP_YARN_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME

export HADOOP_COMMON_LIB_NATIVE=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export CLASSPATH=$CLASSPATH:$HADOOP_HOME/lib/*

# Spark
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin
export LD_LIBRARY_PATH=$HADOOP_COMMON_LIB_NATIVE:$LD_LIBRARY_PATH
export PYSPARK_PYTHON=/opt/anaconda3/bin/python3.6
export PYSPARK_DRIVER_PYTHON=/opt/anaconda3/bin/python3.6

# Flume
```

---

---

## APÉNDICE C. FICHERO DE CONFIGURACIÓN “.BASHRC” COMPLETO

---

```
export FLUME_HOME=/opt/flume
export PATH=$PATH:$FLUME_HOME/bin
export CLASSPATH=$CLASSPATH:$FLUME_HOME/lib/*

# Alias
alias h="hdfs dfs $@"
alias hls="hdfs dfs -ls -h $@"
alias hrm="hdfs dfs -rm -r $@"
alias hreport="hdfs dfsadmin -report"
alias hsafe="hdfs dfsadmin -safemode leave"

# added by Anaconda3 installer
export PATH="/opt/anaconda3/bin:$PATH"
```

---