



---

# Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA  
MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

---

## Análisis Real Time del mercado de valores con tecnologías Big Data

---

Alumno:  
Alberto Gutiérrez del Campo

Tutores:  
Benjamín Sahelices Fernández  
Cristian Vicente Álvarez



---

*«Somos lo que hacemos repetidamente.  
La excelencia, entonces, no es un acto;  
es un hábito.»*  
Aristóteles

*«Si quieres cambiar algo, cambia tú.»*  
Kase O



# Agradecimientos

Quiero dar las gracias a mis tutores Benjamín y Cristian por la ayuda prestada a lo largo del desarrollo de este proyecto, además de por darme la posibilidad de trabajar en este proyecto con tecnologías totalmente novedosas y dedicarme mucho tiempo de calidad de manera totalmente desinteresada. Además quería agradecer a Everis, en especial a Alberto Rico, por su ayuda, gestión, colaboración y consejos a lo largo del desarrollo de este proyecto así como a todos mis compañeros del departamento de Data Analytics.

Me gustaría dar las gracias también a todas las personas que han formado parte de mi etapa universitaria, a todos mis compañeros que a lo largo de los años han sido muchos, pero en especial a Kave Heidarieh por su solidaridad, a Rebeca Colinas por darme tanta luz, a Jesús Javier Fernández por su paciencia conmigo, a Francisco Carrero por su comprensión y a José Martín Gago por su amistad.

A toda mi familia, ya que sin ellos no habría podido llegar hasta aquí. En especial mi hermana que me enseña todos los días una cosa nueva, mi mujer Patricia que desde el cariño hace de mí alguien mejor, mi hijo Mateo que es mi ilusión de vivir y quien me mantiene en la lucha, mi padre que es mi espejo y sobre todo a mi madre; si consigo todo lo que me propongo porque ella me motiva desde donde esté, como un testimonio de eterno agradecimiento por el gran amor y la confianza que siempre me brindó.

Por todos ellos he llegado a realizar una de mis grandes metas lo cual constituye la herencia más valiosa que pudiera recibir.



# Resumen

El paradigma de la información ha cambiado por completo en la última década. El poder almacenar grandes cantidades de información con el fin de explotarlo para obtener un beneficio a presente o futuro es la realidad del sector empresarial. Esta situación está cambiando para todos la forma de ver el mundo, afectando a un gran número de aspectos de la vida tanto cotidiana como específicamente la economía. Todo este escenario es el caldo de cultivo para el crecimiento de las nuevas tecnologías que nos permitan gestionar y utilizar toda esa cantidad de datos. En este proyecto se va a incidir en el almacenamiento, tratamiento y gestión de datos sobre el mercado de valores.

El fin de este proyecto es dar a ver cómo un alumno recién salido del grado en ingeniería informática se adapta a una de las especialidades con mayor mercado en el sector (el desarrollador Big Data fue el perfil más demandado en 2018) así como de las mejor remuneradas en la actualidad y de en vistas al futuro (21.000 euros anuales, ascendiendo hasta los 60.000 en posiciones más sénior).

En el ámbito de este proyecto se van a manejar datos relativos a la segunda bolsa de valores automatizada y electrónica más grande de los Estados Unidos, el NASDAQ, donde se emite en tiempo real el valor de cada empresa que lo forma. Vamos a recoger esta cotización en tiempo real lo que en un caso real es de suma importancia tanto para las propias empresas como para la competencia, inversores y posibles terceros implicados, con el fin de mejorar resultados, inversiones, ganancias o incluso predecir tendencias en el mercado. Las empresas que apuesten por el Big Data podrán aumentar su producción en un 8%, según un informe de la consultora Indra.

Para gestionar esta información se ha decidido que a lo largo de este proyecto se diseñará e implementará un Data Lake. Los Data Lakes surgen debido al aumento de la necesidad de almacenar y gestionar grandes volúmenes de datos de diversos tipos por parte de las organizaciones. El uso de esos datos podría ayudar a mejorar la toma de decisiones de los implicados anteriormente citados, con el fin de llegar antes al mercado o de mejor manera.





# Índice general

|   |            |
|---|------------|
| <b>Agradecimientos</b>                                | <b>III</b> |
| <b>Resumen</b>  | <b>V</b>   |
| <b>Lista de figuras</b>                               | <b>XI</b>  |
| <b>Lista de tablas</b>                                | <b>XV</b>  |
| <b>1. Introducción</b>                                | <b>1</b>   |
| 1.1. Contexto . . . . .                               | 1          |
| 1.2. Objetivos . . . . .                              | 2          |
| <b>2. Estado del arte</b>                             | <b>3</b>   |
| 2.1. Big Data . . . . .                               | 4          |
| 2.1.1. Data Lake . . . . .                            | 6          |
| 2.1.2. Arquitectura <i>LAMBDA</i> . . . . .           | 19         |
| 2.1.3. Análisis del mercado Big Data . . . . .        | 21         |
| 2.2. Introducción al mercado de valores . . . . .     | 24         |
| 2.3. Naturaleza de los datos . . . . .                | 26         |
| 2.3.1. Cotización en tiempo real . . . . .            | 28         |
| 2.3.2. Cotización histórica de una compañía . . . . . | 31         |
| 2.4. Análisis de tareas a realizar . . . . .          | 33         |

|   |           |
|---|-----------|
| 2.5. Estimación planificación . . . . .                               | 34        |
| <b>3. Análisis</b>  | <b>35</b> |
| 3.1. Objetivos . . . . .  | 36        |
| 3.2. Limitaciones y restricciones de implementación . . . . .         | 37        |
| 3.3. Actores del Sistema . . . . .                                    | 38        |
| 3.4. Metodología . . . . .  | 39        |
| 3.5. Product Backlog . . . . .  | 42        |
| 3.5.1. Historias de Usuario . . . . .                                 | 42        |
| 3.6. Sprint Backlog . . . . .   | 46        |
| 3.7. Planificación . . . . .  | 48        |
| 3.8. Gestión de Riesgos . . . . .                                     | 51        |
| 3.9. Presupuesto Final . . . . .                                      | 57        |
| <b>4. Diseño</b>  | <b>59</b> |
| 4.1. Modelo de Datos . . . . .  | 60        |
| 4.1.1. Raw Data . . . . .   | 60        |
| 4.1.2. Refined Data . . . . .   | 65        |
| 4.1.3. Transformación de Datos . . . . .                              | 69        |
| 4.2. Arquitectura lógica . . . . .                                    | 73        |
| 4.3. Arquitectura Física . . . . .                                    | 74        |
| 4.4. Diseño de interfaces . . . . .                                   | 75        |
| 4.4.1. Interfaz para los datos en Real Time . . . . .                 | 75        |
| 4.4.2. Interfaz para los datos Históricos . . . . .                   | 76        |
| 4.4.3. Interfaz para la visualización de tipo Trading Chart . . . . . | 76        |
| <b>5. Implementación</b>  | <b>79</b> |
| 5.1. Descripción del sistema . . . . .                                | 80        |

|  |            |
|--|------------|
| 5.2. Descripción del entorno . . . . .                           | 81         |
| 5.2.1. Activación del Cloudera Manager . . . . .                 | 81         |
| 5.2.2. Instalación de Java 1.8 . . . . .                         | 82         |
| 5.2.3. Instalación de Spark 2. . . . .                           | 85         |
| 5.2.4. Instalación de NiFi . . . . .                             | 87         |
| 5.3. Herramientas utilizadas . . . . .                           | 89         |
| 5.3.1. Cloudera CDH 5.13.0 . . . . .                             | 89         |
| 5.3.2. IntelliJ . . . . .  | 89         |
| 5.3.3. Power BI . . . . .  | 90         |
| 5.3.4. Overleaf . . . . .  | 92         |
| 5.3.5. GanttPRO . . . . .  | 93         |
| 5.4. Tecnologías utilizadas . . . . .                            | 95         |
| 5.5. Implementación del Data Lake . . . . .                      | 102        |
| 5.5.1. Hive. . . . .   | 102        |
| 5.5.2. Impala. . . . .   | 105        |
| 5.5.3. HDFS. . . . .   | 106        |
| 5.5.4. NiFi. . . . .   | 107        |
| 5.5.5. HQL Scripts y Shell Scripts . . . . .                     | 115        |
| 5.6. Implementación del programa de cierre del mercado . . . . . | 118        |
| 5.6.1. Pruebas del programa de cierre del mercado . . . . .      | 120        |
| 5.7. Implementación del dashboard . . . . .                      | 121        |
| <b>6. Conclusiones</b>   | <b>125</b> |
| 6.1. Conclusiones . . . . .                                      | 125        |
| 6.2. Trabajo futuro . . . . .                                    | 127        |
| 6.3. Aprendizaje . . . . .                                       | 128        |
| <b>A. Contenidos del CD-ROM</b>                                  | <b>131</b> |



# Índice de figuras

|   |    |
|---|----|
| 2.1. Resumen y ejemplo de usos de un Data Lake. . . . .                             | 7  |
| 2.2. Diferencia entre ETL y ELT. . . . .  | 8  |
| 2.3. Arquitectura de un Data Lake. . . . .  | 10 |
| 2.4. Gobierno de Datos y Seguridad. . . . .   | 10 |
| 2.5. Funcionalidades de la Capa de gestión del ciclo de vida de la información. . . | 11 |
| 2.6. Capa de Metadatos. . . . .   | 12 |
| 2.7. Tipos de flujo de entrada. . . . .   | 13 |
| 2.8. Funcionalidades de <i>Transient Zone</i> . . . . .                             | 14 |
| 2.9. Funcionalidades de alto nivel asociadas a la <i>Raw Zone</i> . . . . .         | 15 |
| 2.10. Funcionalidades asociadas a la <i>Integration Zone</i> . . . . .              | 15 |
| 2.11. Funcionalidades asociadas a la <i>Enrichment Zone</i> . . . . .               | 16 |
| 2.12. Estructura del Nivel de Obtención. . . . .                                    | 17 |
| 2.13. Flujo a seguir por los datos en un Data Warehouse durante ETL. . . . .        | 18 |
| 2.14. Arquitectura <i>LAMBDA</i> . . . . .  | 19 |
| 2.15. Capa de Velocidad y Servicio . . . . .  | 20 |
| 2.16. Arquitectura Lambda completa . . . . .  | 20 |
| 2.17. Arquitectura de Spark repartiendo el trabajo. . . . .                         | 22 |
| 2.18. Arquitectura de SPARK on YARN. . . . .  | 23 |
| 2.19. Cómo funciona el mercado de valores. . . . .                                  | 24 |

|  |    |
|--|----|
| 2.20. Resultado bursátil de Google para AAPL. . . . .  | 26 |
| 2.21. Resultado bursátil de Yahoo para AAPL. . . . .   | 27 |
| 2.22. Documentación de la API de IEX. . . . .  | 28 |
| 2.23. Ejemplo de JSON de la API de IEX para AAPL. . . . .  | 29 |
| 2.24. Interfaz de Yahoo Finance para una cotización, en este caso la de Apple, Inc. . . . .                      | 31 |
| 2.25. Interfaz de Yahoo Finance para los datos históricos de una empresa, en este caso la de Apple, Inc. . . . . | 32 |
| 3.1. Jerarquía de las historias de usuario. . . . .  | 43 |
| 4.1. Ejemplo de CSV en bruto de Yahoo Finance para la empresa Apple. . . . .                                     | 61 |
| 4.2. Diagrama de Entidad-Relación. . . . .   | 66 |
| 4.3. ReplaceText de NiFi en desuso para nosotros. . . . .  | 70 |
| 4.4. Nuestro ConvertJSONToAvro de NiFi. . . . .  | 71 |
| 4.5. Detalle del esquema de Avro para los datos recogidos en tiempo real. . . . .                                | 71 |
| 4.6. Arquitectura lógica. . . . .  | 73 |
| 4.7. Arquitectura física utilizada. . . . .  | 74 |
| 4.8. Boceto de la ventana de visualización para los datos en tiempo real. . . . .                                | 75 |
| 4.9. Boceto de la ventana de visualización para los datos históricos. . . . .                                    | 76 |
| 4.10. Funcionamiento del gráfico de tipo Trading Chart. . . . .  | 77 |
| 4.11. Boceto de la ventana de visualización para el gráfico de velas Trading Chart. . . . .                      | 78 |
| 5.1. Ejemplo de consulta en Impala desde HUE. . . . .  | 82 |
| 5.2. Reinicio de Cloudera Management Service. . . . .  | 84 |
| 5.3. Parcel de SPARK2 en Cloudera Manager. . . . .   | 86 |
| 5.4. Consola de SPARK2 en nuestra máquina. . . . .   | 87 |
| 5.5. Ejemplo de orígenes de Power BI. . . . .  | 91 |
| 5.6. Ecosistema de Power BI. . . . .   | 92 |

|  |     |
|--|-----|
| 5.7. Ejemplo de mi aplicación vista desde la aplicación Android de Power BI. . . .   | 92  |
| 5.8. Captura de pantalla de mi panel de Overleaf. . . . .  | 93  |
| 5.9. Captura de pantalla de mi panel de GanttPRO. . . . .  | 94  |
| 5.10. Versiones de Spark a lo largo del tiempo. . . . .  | 95  |
| 5.11. Posibles orígenes de Apache Spark. . . . .   | 97  |
| 5.12. Replicación en clústers utilizando data nodes. . . . .   | 98  |
| 5.13. Gestión de acceso de los Name Nodes a los Data Nodes. . . . .  | 98  |
| 5.14. Datos sobre los que trabajaremos en el ejemplo. . . . .  | 99  |
| 5.15. Hive en el ecosistema hadoop. . . . .  | 100 |
| 5.16. Ecosistema de BBDD y tablas de nuestro Datalake. . . . .   | 103 |
| 5.17. Vista de directorios para albuguti en el HDFS. . . . .   | 106 |
| 5.18. Pantalla inicial de NiFi con nuestro flow. . . . .   | 107 |
| 5.19. Invalidate Metadata en Impala con ExecuteProcess de NiFi. . . . .  | 108 |
| 5.20. Nuestro flow de NiFi para aapl. . . . .  | 108 |
| 5.21. Planificación con CRON de InvokeHTTP para ejecutarse cada minuto de 6 a<br>13 horas y solo de lunes a viernes (cuando está abierto es NASDAQ). . . . .         | 109 |
| 5.22. Valores de las propiedades de configuración de InvokeHTTP para recoger los<br>JSONs de la API de IEX Trading. . . . .  | 109 |
| 5.23. Configuración del StandardRestrictedSSLContextService necesario para la eje-<br>cución del proceso InvokeHTTP. . . . .   | 110 |
| 5.24. Configuración del ConvertJSONToAvro donde se le indica el esquema Avro<br>que debe seguir para la conversión. . . . .  | 110 |
| 5.25. Configuración del proceso PutHiveStreaming de NiFi que insertará cada Avro<br>generado por el anterior proceso en la tabla de destino que se le configure. . . | 114 |
| 5.26. Configuración del proceso ExecuteProcess de NiFi que ejecuta el script que<br>genera la respectiva actualización a la tabla Parquet. . . . .                   | 115 |
| 5.27. Captura de pantalla que muestra el código de nuestra aplicación Spark-Scala<br>abierta en IntelliJ. . . . .  | 118 |
| 5.28. Captura de pantalla del Dashboard para los datos recogidos en tiempo real<br>con la empresa Facebook, Inc. seleccionada. . . . .                               | 121 |

|  |     |
|--|-----|
| 5.29. Captura de pantalla del Dashboard para los datos históricos con la empresa AMD, Inc. seleccionada y filtrando por fecha desde el 8 de mayo de 2017 al 28 de Junio de 2019. . . . . | 122 |
| 5.30. Captura de pantalla del Dashboard para el gráfico de velas de los 30 últimos días con la empresa Intel, Corp. seleccionada. . . . .  | 122 |



# Índice de cuadros

|   |    |
|---|----|
| 2.1. Diferencias entre Data Lake y Data Warehouse. . . . .                | 18 |
| 3.1. Actor Usuario General . . . . .                                      | 38 |
| 3.2. Actor albguti . . . . .  | 38 |
| 3.3. Explicación componentes de scrum . . . . .                           | 41 |
| 3.4. Product Backlog . . . . .  | 42 |
| 3.5. Épica 01 . . . . .   | 44 |
| 3.6. Épica 02 . . . . .   | 45 |
| 3.7. Sprint Backlog . . . . .   | 47 |
| 3.8. Listado de riesgos . . . . .   | 51 |
| 3.9. Listado de riesgos priorizados . . . . .                             | 52 |
| 3.10. Risk-01. Retrasos en la planificación . . . . .                     | 53 |
| 3.11. Risk-02. Desconocimiento de las tecnologías . . . . .               | 53 |
| 3.12. Risk-03. Cambios en la especificación de los requisitos . . . . .   | 54 |
| 3.13. Risk-04. Problemas hardware en los servidores disponibles . . . . . | 54 |
| 3.14. Risk-05. Falta de experiencia del equipo de desarrollo . . . . .    | 55 |
| 3.15. Risk-06. Problemas de salud en el equipo de desarrollo . . . . .    | 55 |
| 3.16. Risk-07. Llegar a un punto sin salida en la investigación . . . . . | 56 |
| 3.17. Risk-08. Desarrollo del mismo producto por la competencia . . . . . | 56 |
| 3.18. Coste componentes Hardware . . . . .                                | 57 |

|   |    |
|---|----|
| 3.19. Coste componentes Software . . . . .                          | 57 |
| 4.1. Campos asociados a los datos históricos . . . . .              | 61 |
| 4.2. Campos asociados a los datos captados en tiempo real . . . . . | 64 |
| 4.3. Entidad Real Time . . . . .                                    | 68 |
| 4.4. Entidad Historic . . . . .                                     | 68 |
| 4.5. Entidad Bakruptcy . . . . .                                    | 68 |
| 4.6. Entidad Correlation Mark . . . . .                             | 69 |
| 5.1. Lenguajes soportados por IntelliJ . . . . .                    | 90 |

# Capítulo 1

## Introducción

### 1.1. Contexto

La motivación inicial de este proyecto viene dada ante la oportunidad que me surge al entrar a trabajar en Everis en el departamento de Big Data en un contexto con cliente bancario (Banco Santander) y teniendo en ese momento en el horizonte la realización de este mi TFG, me planteo aprovechar esta oportunidad de adaptación a las tecnologías Big Data para aplicarlo a dicho proyecto.

Reunido con mi tutor de la empresa, Cristian Vicente Álvarez ingeniero informático especialista en Big Data con certificación en Cloudera & Spark, y tras una lluvia de ideas para enfocar el proyecto, llegamos a la primera idea y principal de este proyecto “abarcar en todo momento los lenguajes y las tecnologías más punteras en el ámbito empresarial para el BigData”. También se nos ocurrió lo que pensamos que es un buen escenario para este juego, el mercado de valores.

### 1.2. Objetivos

Trataremos de realizar procesos básicos en el Big Data como las ETLs, transformaciones, agregados, históricos y un panel final de visualización y consumo del dato. Pero siempre como hemos dicho aplicando tecnologías y herramientas punteras. Así describimos una primera idea con un escenario base (Ilustración XYZ) donde vemos que ingestamos datos históricos de empresas caídas en bolsa y de empresas que se revalorizan en bolsa como buenos ejemplos de inversión; ingestamos datos históricos de empresas que actualmente cotizan en el mercado (NASDAQ) e ingestamos en tiempo real datos de estas mismas empresas.

Tras procesos de tratamiento, limpieza, agregados, cierres, normalizaciones, creaciones de índices, etc. La idea es hacer comparaciones regresivas para ver cómo de mucho o poco estas empresas que actualmente cotizan se acercan a esos ejemplos (o índices referencia) de caída o ascenso bursátil.

Como resultado final tendremos un panel, explotado con herramientas de visualización BigData como puede ser QlikView o Power BI, donde veremos distintas opciones: un panel con un top5 de empresas que parecen caer, top5 de empresas que parecen subir notablemente, acceso al histórico de cada empresa, y diversos gráficos de interés.

## Capítulo 2

# Estado del arte

En la actualidad es un hecho que los datos son el motor del mundo, debido a esto, se han producido una gran cantidad de avances, tanto en arquitecturas como en tecnologías, con el fin de satisfacer las capacidades de análisis y almacenamiento derivadas de la cada vez más creciente generación de datos.

## 2.1. Big Data

Estos avances se engloban dentro del paraguas que ofrece Big Data. Se considera Big Data a todas las arquitecturas y tecnologías encargadas de almacenar y gestionar datos cuyo tamaño, complejidad y escalabilidad son elevados. Existen una gran cantidad de definiciones que tratan de explicar qué es Big Data, por ejemplo, Oxford English Dictionary (OED) ha definido Big data como data of a very large size, typically to the extent that its manipulation and management present significant logistical challenges". Pese a las definiciones tradicionales de Big Data, como la vista anteriormente, es prácticamente imposible definir Big Data sin pensar en las 3 Vs, término utilizado con el fin de identificar sus principales características. Normalmente se identifica a las 3 Vs como:

1. **Volumen:** En la actualidad es posible acceder a enormes conjuntos de datos, debido principalmente al gran volumen de información que es posible captar gracias a satélites o a las redes sociales. Este hecho supone un gran problema a la hora de almacenar esta información, debido, principalmente, a que los sistemas tradicionales tienen problemas para tratar la gran escalabilidad asociada a los datos actuales, además de problemas asociados a la capacidad de almacenamiento de los hardware. Ante la necesidad de grandes capacidades de almacenamiento, a poder ser con bajo coste, nace Apache Hadoop, plataforma tecnológica opensource utilizada para el tratamiento de grandes volúmenes de datos y su almacenamiento distribuido.
2. **Velocidad:** La posibilidad de obtener datos en streaming, es decir, en tiempo real, unido con la imperiosa necesidad de obtener datos en el menor tiempo posible, puede dar lugar a problemas a la hora de procesar los datos en las base de datos tradicionales. Big Data permite mejorar los flujos de entrada de información, así como procesar grandes volúmenes de información a gran velocidad, eliminando los cuellos de botella comunes en las bases de datos tradicionales.
3. **Variedad:** En la actualidad los datos no provienen únicamente de sistemas relacionales, sino que son enviados a través de otros sistemas distintos, como sensores o redes sociales. La gran variedad de datos da lugar a problemas de procesamiento de los mismos, es sabido que los datos estructurados son los más fáciles de procesar. Estos datos se dividen, según su estructura, en 3 tipos:
  - **Datos estructurados:** Se denominan datos estructurados a todos aquellos con una estructura totalmente definida, delimitando, por ejemplo, la longitud y el formato de los datos. Suelen estar sujetos a una gran cantidad de normas bastante estrictas. Estos datos suelen ser almacenados en bases de datos relacionales, en las que los datos se almacenan en tablas, con un formato previo.
  - **Datos semi-estructurados:** Estos datos, pese a no presentar una estructura perfectamente definida, si que cuentan con una ligera organización interna que facilita su tratamiento. Esta organización se basa principalmente en los metadatos, encargados de describir los objetos y sus relaciones. Dentro de este tipo de datos destacan XML y JSON.

- **Datos no estructurados:** Estos datos no se ajustan a los formatos convencionales, al no estar dotados de estructura alguna. La generación de este tipo de datos se lleva a cabo a gran velocidad, debido principalmente a las redes sociales, ya que los tweets, imágenes o vídeos entran dentro de este tipo de datos. En la actualidad la mayor parte de la información generada no tiene estructura interna.

Aunque las 3 Vs están mundialmente aceptadas, actualmente han surgido nuevos autores y empresas, como IBM, que sostienen que estas pueden ampliarse a 5 Vs, añadiendo para ello las siguientes características:

4. **Veracidad:** Uno de los problemas de las ingentes cantidades de datos que se pueden captar en la actualidad es su grado de veracidad. Esta característica está estrechamente relacionada con la limpieza de datos y el análisis de los datos obtenidos, tratando de mejorar la calidad de la información, con el objetivo de alcanzar una correcta toma de decisiones. A menudo los problemas de veracidad de los datos son compensados con el gran volumen de datos obtenido.
5. **Valor:** La posibilidad de almacenar grandes volúmenes de información no es de mucha utilidad, a no ser que sea posible dar valor a estos datos. Esta característica es fundamental en estos sistemas, ya que el hecho de almacenar información poco precisa puede dar lugar a graves problemas una vez sea utilizada. La importancia del valor de los datos se puede ver en las empresas que persiguen obtener un mayor conocimiento sobre su mercado, gracias al análisis de estos datos. En estos casos la toma de decisiones se puede ver afectada enormemente por la inexactitud de los datos captados. Las empresas están destinando cada vez más recursos al análisis de los datos obtenidos, ya que almacenar datos de los que no se puede extraer valor puede ser muy costoso.

Otro aspecto a destacar dentro de las soluciones Big Data es el Gobierno de Datos (*Data Governance*), capacidad de una organización para gestionar el conocimiento que tiene sobre su información. Este aspecto proporciona un enfoque completo para mejorar y administrar la información obtenida. Esto es de vital importancia en las organizaciones, ya que un mal gobierno de datos forma un camino directo al descontrol de los datos, el cual puede alcanzar grandes proporciones al trabajar con un gran volumen de información. Esta es una de las principales causas de muerte por éxito de los proyectos Big Data y las empresas que los llevan a cabo, ya que la gestión tradicional en proyectos de este tamaño es muy complicada.

Gracias al Big Data, junto con los grandes avances actuales en el campo de la Inteligencia Artificial, es posible crear soluciones a problemas que no nos habíamos planteado. Sin embargo el Big Data no es perfecto, como se puede ver en el artículo publicado por Forbes *3 Massive Big Data Problems Everyone Should Know About* [18], existe una creciente preocupación por la privacidad de los datos y su seguridad así como las posibles ambigüedades presentes en estos.

El auge del Big Data ha supuesto una revolución tecnológica de gran dimensión, favoreciendo al desarrollo de una gran cantidad de arquitecturas, como Data Warehouse y Data Lake, así como la creación de una gran cantidad de puestos de trabajo. En la siguiente sección se explicará la arquitectura que se propone como solución en este proyecto, el Data Lake.

### 2.1.1. Data Lake

La preocupación actual, tanto de empresas como de particulares, por la gestión de sus datos es enorme. Como previamente se ha comentado, el volumen de datos que se generan en la actualidad está sufriendo un crecimiento exponencial, principalmente debido al auge de las redes sociales. Este gran crecimiento está llevando de la mano la creación de nuevas arquitecturas encargadas de almacenar cualquier tipo de dato, estructurado, semi-estructurado y no estructurado. Cabe destacar el crecimiento de los datos no estructurados, el cual ronda un 23 % por año [20].

Es un hecho que gran parte de la información que se capta carece de un valor inmediato, es decir, se desconoce su utilidad inmediata. Sin embargo no conocer la utilidad actual de los datos no significa que en el futuro no adquieran una gran importancia en los procesos de negocio de las diferentes empresas. Ante esto, grandes empresas como Google almacenan toda la información que adquieren, tratando de buscar un valor a futuro. Esto es posible verlo gracias al artículo *¿Qué es un Data Lake? [19]*, en el cual se comenta que un ingeniero de Google dijo *En Google guardamos todos los datos aunque no les veamos ningún valor hoy en día. No sabemos si dentro de 1, 5 o 10 años se nos ocurrirá una idea para explotarlos y perderlos sería un grave error*".

Hechos como los anteriormente mencionados, unidos a la gran heterogeneidad de las nuevas fuentes de datos, han evidenciado la necesidad de crear nuevas arquitecturas para almacenar y gestionar la información obtenida. En este ámbito comienza a surgir una nueva arquitectura, llamada Data Lake, mediante la cual se persigue almacenar y procesar cualquier tipo de información, estructurada, semi-estructurada y no estructurada, tratando de mejorar el tratamiento de la información para prevenir problemas de amigüedades en los datos. En las siguientes secciones se explicará en detalle qué es un Data Lake.

#### Definición

Un Data Lake es un repositorio donde se almacena cualquier tipo de datos, estructurados, semi-estructurados y no estructurados, sin ningún tipo de pre procesamiento ni de esquema, ofreciendo además mecanismos para refinar y explorar estos datos. En los Data Lake los datos se almacenan en bruto, es decir, son cargados directamente desde las fuentes originales sin descartar ningún elemento, almacenándose en su estado original. Al no existir un pre procesamiento, no existe ningún tipo de agregación ni de transformación, lo que permite disponer de toda la información potencial contenida en los datos. En la figura 4.1 es posible ver un pequeño resumen de lo que es un Data Lake así como alguno de sus usos, complementando lo explicado anteriormente.





Figura 2.1: Resumen y ejemplo de usos de un Data Lake.

El concepto Data Lake suele encontrarse íntimamente ligado con Apache Hadoop, plataforma open-source que permite el despliegue de esta arquitectura con un coste reducido. Sin embargo no es la única vía de desarrollo de un Data Lake, ya que es posible desarrollar este tipo de arquitecturas sin necesidad de Hadoop, aunque normalmente es lo más utilizado.

Esta arquitectura tiene grandes puntos de ruptura con las arquitecturas tradicionales, tales como los Data Warehouse. Un aspecto a destacar, dentro de los Data Lake, es el momento en el que se realiza el formateo de los datos. Esta arquitectura rompe en este punto con las arquitecturas tradicionales, ya que en esta los datos son formateados, dotados de estructura, únicamente en el momento en que son necesarios para satisfacer una necesidad de análisis. Esta propiedad, conocida como "schema on read", es una de las más destacadas de esta arquitectura, al distinguirla totalmente de las arquitecturas tradicionales, las cuales utilizan un "schema on write". El schema on read ha supuesto una gran innovación en el mundo de Big Data, ya que el hecho de poder almacenar los datos sin esquemas permite añadir y modificar grupos de datos fácilmente, reduciendo los costos asociados a su transformación, aunque también es posible contar con repositorios de esquemas que nos faciliten el acceso a los datos.

Además de la anterior existe una gran cantidad de características asociadas a los Data Lakes, sin embargo ninguna tiene la gran relevancia de los ELT (Extract, Load, Transform). En este tipo de arquitectura se sigue este sistema, según el cual los datos son extraídos de las fuentes y cargados como Raw Data en un sistema de archivos distribuido, a diferencia de las arquitecturas tradicionales, las cuales siguen un proceso ETL (Extract, Transform, Load). Esta característica es de gran importancia, ya que como tal es la que permite el almacenamiento de raw data. En la figura 2.2 se puede ver la diferencia existente entre los dos sistemas previamente explicados.

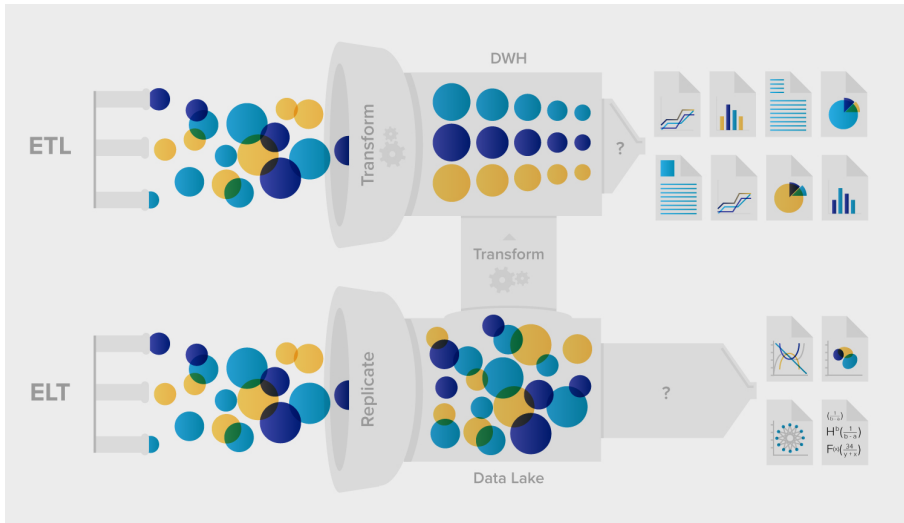


Figura 2.2: Diferencia entre ETL y ELT.

La posibilidad de almacenar la información sin haberla dotado de un esquema da lugar a diversas ventajas, siendo un elemento decisivo a la hora de afrontar problemas de escalabilidad y heterogeneidad de los datos. Algunas de las principales ventajas de este tipo de sistemas son:

- La posibilidad de cargar los datos en bruto, directamente desde las fuentes originales y sin ningún tipo de preprocesamiento, facilita a los analistas procesar los datos con una mayor flexibilidad.
- Los datos no estructurados y semi-estructurados son soportados fácilmente y cargados a gran velocidad.

Sin embargo no todo son ventajas en este tipo de sistemas, por ejemplo, el hecho de almacenar los datos sin ningún esquema imposibilita su análisis, es decir, no es posible analizar datos a no ser que se les dote de un esquema.

Una de las preocupaciones más extendidas asociadas al uso de los Data Lakes deriva de la posible acumulación de datos incomprensibles. Esta preocupación surge debido a la falta de esquema de los datos almacenados, siendo la principal causa de convertir un Data Lake en un pantano de datos. Para intentar mitigar esta preocupación se creó un repositorio de metadatos, el cual se encarga de almacenar información de alto nivel sobre las entidades de datos a almacenar, como por ejemplo el tipo, el tiempo o el creador.

Otro de los principales problemas que pueden tener lugar en este tipo de arquitecturas radican en la potencial ambigüedad de los datos. Este hecho deriva de la heterogeneidad de las fuentes encargadas de captar datos, las cuales pueden obtener los mismos datos y así dar lugar a problemas de ambigüedad. Ante este problema adquiere una enorme importancia

el gobierno de datos, ya que un buen gobierno de datos puede evitar en gran medida los problemas de ambigüedad en los datos.

El desarrollo de estos sistemas suele encontrarse muy ligado a la Inteligencia Artificial, la cual es capaz de potenciar las capacidades de los Data Lakes. Es muy común el desarrollo de inteligencias artificiales encargadas de facilitar la integración de las diversas fuentes de datos, analizar los conjuntos de datos, automatizando los procesos, así como utilizar los datos almacenados mediante *Machine Learning* para mejorar los diferentes flujos empresariales.

### Arquitectura

La arquitectura de un Data Lake suele estar compuesta por 3 capas, compuestas a su vez por 3 niveles. Las capas que conforman este tipo de arquitectura son la capa de Gobierno de Datos y Seguridad (Data Governance and Security layer), capa de Metadatos (Metadata layer) y la capa de gestión del ciclo de vida de la información (Information Lifecycle Management layer). Sin embargo esta no es la única interpretación de la arquitectura de un Data Lake, ya que existen múltiples interpretaciones sobre la misma en función de los requisitos tanto empresariales como tecnológicos. Algunos de los principales factores que afectan a la arquitectura de un Data Lake son los siguientes:

- La forma de obtención de datos utilizada, como por ejemplo en tiempo real.
- La forma de almacenamiento de los datos.
- El gobierno de datos que se desea llevar a cabo.
- La profundidad de los metadatos.

La arquitectura por capas, mencionada anteriormente, es la más tradicional que se puede encontrar en un Data Lake, además de la más utilizada en la actualidad. Esta arquitectura a su vez también se encuentra caracterizada por 3 niveles distintos, en las cuales se agrupan funcionalidades de mayor similitud, es decir, con un nivel de abstracción menor. El flujo existente entre los 3 niveles es secuencial, mientras que los datos son transmitidos entre los niveles, las capas se encargan del procesamiento de los mismos. Estos niveles son:

- Nivel de entrada (Intake Tier)
- Nivel de gestión (Management Tier)
- Nivel de consumo (Consumption Tier)

En la figura 2.3 se muestra en detalle la arquitectura previamente explicada.

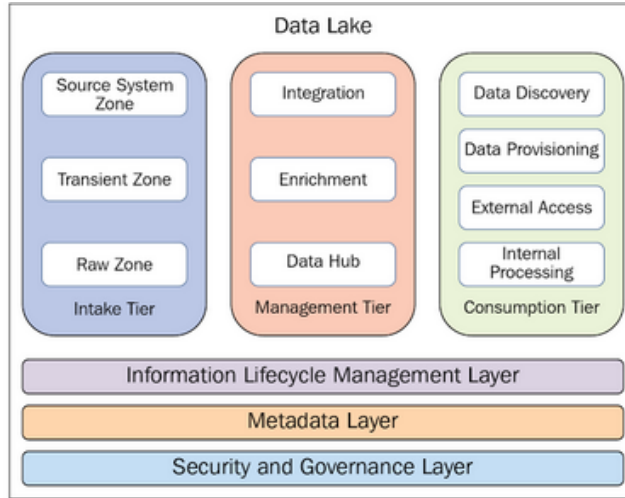


Figura 2.3: Arquitectura de un Data Lake.

A continuación se explicarán las diversas capas y niveles que forman esta arquitectura.

**Gobierno de Datos y Seguridad (Data Governance and Security layer).** Capa encargada de regular el acceso y modificación a los datos de los Data Lakes. Para alcanzar la consecución de este objetivo esta capa documenta todo el proceso para el cambio y acceso a todos los datos. Este sistema realiza un seguimiento de todas las modificaciones llevadas a cabo en los distintos niveles de un Data Lake, combinándose con las normas de seguridad. Estas normas se encargan de garantizar la integridad de los datos, así como el control del acceso y autorización a los mismos. Un ejemplo de sistema que gestiona este tipo de capas es Kerberos, solución utilizada en Hadoop con el fin de garantizar el autocontrol de los usuarios. En la figura 2.4 se puede ver la funcionalidad de la capa de Gobierno de Datos y Seguridad, obtenida del libro *Data Lake development with Big Data: explore architectural approaches to building Data Lakes that ingest, index, manage, and analyze massive amounts of data using Big Data technologies* [21].

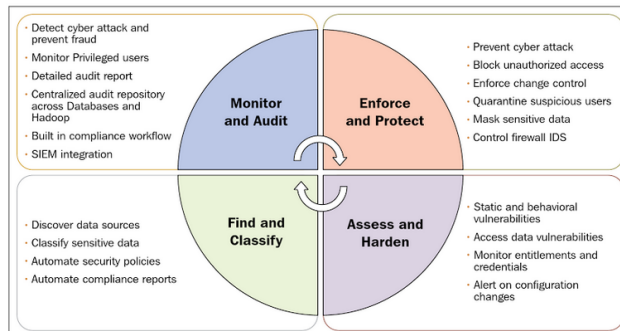


Figura 2.4: Gobierno de Datos y Seguridad.

**Capa de gestión del ciclo de vida de la información (Information Lifecycle Management layer).** Esta capa se encarga de asegurar las reglas que gobiernan el almacenamiento de datos en un Data Lake. Estas reglas suelen utilizarse para gestionar la información almacenada a lo largo de su ciclo de vida. Esto se debe a que la información suele perder valor con el paso del tiempo, lo que eleva el riesgo de uso de la misma.

Esta capa por tanto trata de definir la estrategia y políticas a utilizar, tratando así de clasificar los datos de valor y determinar cuando deberían dejar de ser almacenados. Las herramientas, encargadas de realizar estas tareas, son automáticas y permiten limpiar y archivar los datos en función de las políticas a seguir.

En la figura 2.5, obtenida también a gracias al libro escrito por Pradeep Pasupuleti y Beulah Salome Purra, se puede ver las funcionalidades asociadas a esta capa.

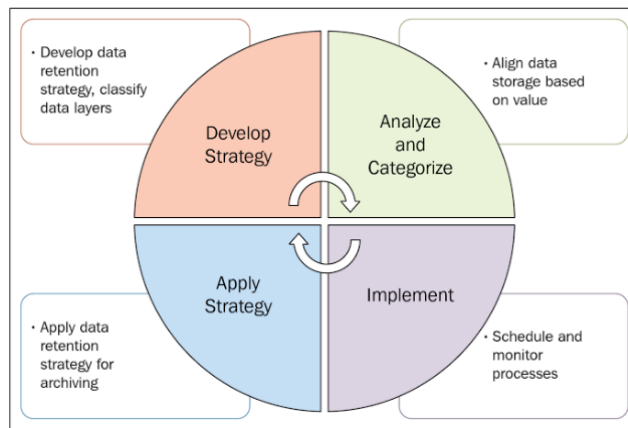


Figura 2.5: Funcionalidades de la Capa de gestión del ciclo de vida de la información.

**Capa de Metadatos (Metadata layer).** La capacidad de un Data Lake de almacenar cualquier tipo de datos se encuentra especialmente ligada con esta capa. Esta capa trata de ofrecer un mecanismo mediante el cual se puedan encontrar los vínculos entre la información que se almacena y quién puede acceder a esta. Esta capa, por tanto, suele considerarse el corazón de un Data Lake.

En esta capa se almacena la información sobre los datos, a medida que son obtenidos, y la indexa con el fin de poder acceder antes a los metadatos que a los propios datos, facilitando en gran medida la accesibilidad a estos. La posibilidad de indexar esta información es de gran importancia en un Data Lake, ya que posibilita el acceso a la información almacenada, por mucho que los datos se almacenen en su formato original o incluso no tengan estructura, permitiendo así almacenar los datos en su formato original para después ser transformados en el momento en que vayan a ser usados. Es por tanto muy importante para definir la estructura de los ficheros almacenados en el Raw Zone, describiendo también las entidades existentes en los ficheros.

Esta capa es capaz de proveer de información de vital importancia al Data Lake, como

por ejemplo sobre el significado de la información almacenada en el mismo. Una buena construcción de la capa de metadatos aumenta el potencial del Data Lake, permitiendo realizar gran cantidad de análisis mediante sistemas como Machine Learning as a Service (MLaaS) o Data as a Service (DaaS). En la figura 2.6 se pueden ver las algunas de las capacidades asociadas a la capa previamente explicada.

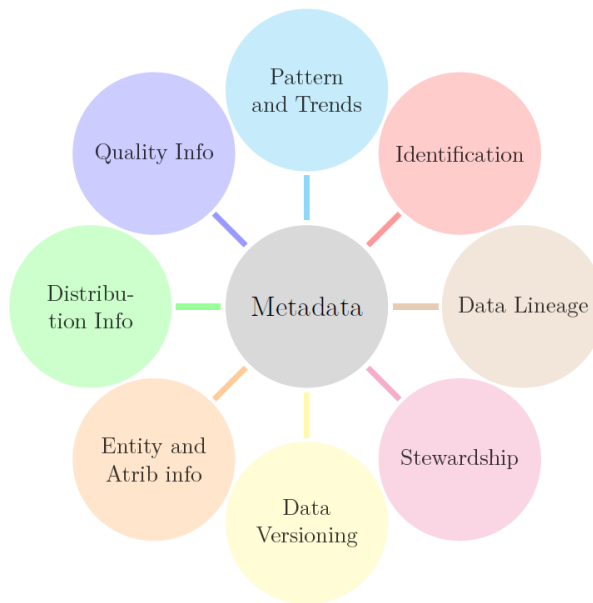


Figura 2.6: Capa de Metadatos.

**Nivel de Entrada (Intake tier).** Nivel encargado de gestionar todos los servicios que se conectan a fuentes externas, así como del área de almacenamiento de los datos obtenidos. A su vez este nivel se encuentra dividido en 3 zonas. Estas zonas son:

1. *The Source System Zone:* Zona encargada de gestionar las conexiones con las fuentes externas y la obtención de datos de dichas fuentes. Un hecho a tener en cuenta en esta zona es el tiempo de adquisición de la información, el cual depende de los requisitos de la aplicación a crear, pudiendo ser en tiempo real, en batch o en micro batch. En la figura 2.7 es posible ver los diferentes flujos de información que puede captar un Data Lake.

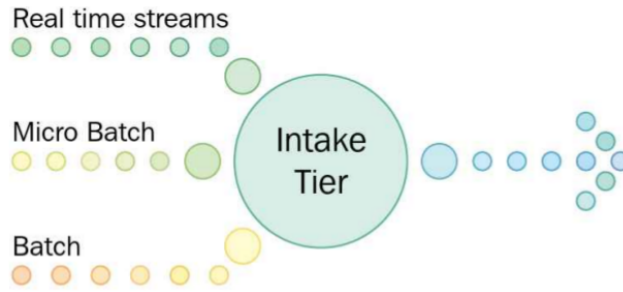


Figura 2.7: Tipos de flujo de entrada.

2. *The Transient Zone:* Zona intermedia entre The Source System Zone y The Raw Zone encargada de almacenar la información los datos obtenidos, por fuentes, antes de ser enviados a la Raw Zone. Esta zona además se encarga de calcular el tamaño de los datos obtenidos. Al realizar unas comprobaciones mínimas de validación de datos su inexistencia podría ocasionar bajadas en la calidad de los datos almacenados en la Raw Zone. En la figura 2.8 se pueden ver algunas de las funcionalidades asociadas a esta zona.

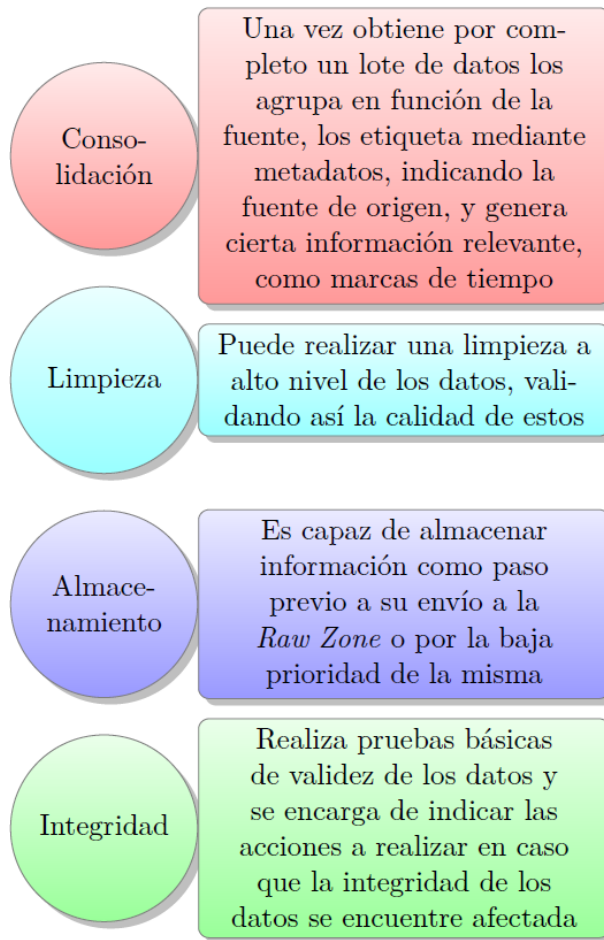


Figura 2.8: Funcionalidades de *Transient Zone*.

3. *Raw Zone*: Zona encargada de almacenar la información obtenida. Para implementar esta zona se utiliza un sistema de archivos distribuidos, normalmente HDFS, incluyendo un área en el que se conservan los datos en su formato original. Este área varía en función de si los datos son obtenidos en tiempo real o por lotes. En la figura 2.9 es posible ver a alto nivel el esquema de uso de la *Raw Zone*.



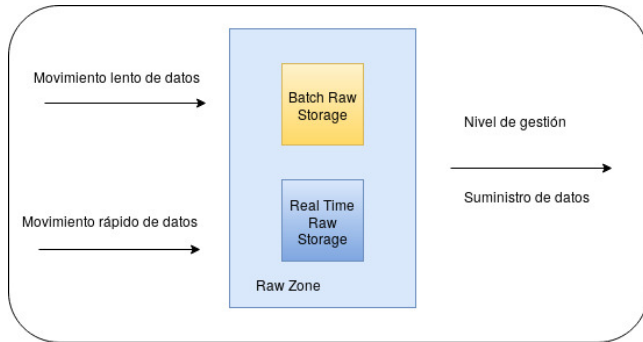


Figura 2.9: Funcionalidades de alto nivel asociadas a la *Raw Zone*.

**Nivel de Gestión (Management Tier).** Nivel encargado del tratamiento y preparación de los datos, previamente almacenado en su formato original, para su análisis y uso futuro. Este nivel, al igual que el anterior, se encuentra dividido en 3 zonas, a través de las cuales fluyen los datos de manera secuencial, hasta almacenarse en el *Data Hub*, zona donde se almacenan tanto datos estructurados como no estructurados. En este nivel se añaden y adjuntan los metadatos a cada fichero, creando así un identificador de los mismos, permitiendo seguir todos los cambios que tengan lugar en cada registro individual. La información asociada a los metadatos de cada fichero es de gran importancia, permitiendo mejorar la gestión de la calidad de los datos, realizar un seguimiento de la progresión de estos e incluso analizar las anomalías y correcciones a realizar a mayor velocidad. Las zonas que componen este nivel son las siguientes:

1. *The Integration Zone*: En esta zona se lleva a cabo la integración de las diversas fuentes de datos, sobre los que se aplican una serie de transformaciones creando una estructura estandarizada. En la figura 2.10 es posible ver las principales funcionalidades asociadas a esta zona, así como el flujo seguido por los datos en este nivel.

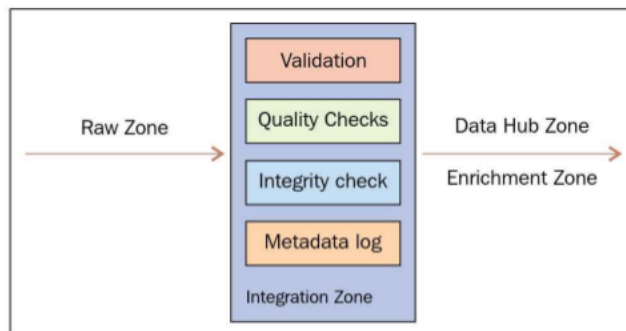


Figura 2.10: Funcionalidades asociadas a la *Integration Zone*.

2. *The Enrichment Zone*: Zona encargada de proporcionar procesos para el tratamiento y gestión de los datos, pudiendo incluir procesos que permitan añadir nuevos atributos

a los registros existentes a través de fuentes externas e internas. En esta zona se utiliza un sistema de archivos distribuido, como puede ser HDFS, permitiendo así alcanzar una mayor flexibilidad en el tratamiento de estos datos. Esta flexibilidad radica en que estos sistemas, principalmente HDFS, permiten el almacenamiento sin necesidad de esquemas o índices, por lo que no es necesario que los datos sean tratados antes de ser usados. En la figura 4.11 es posible ver las diferentes capacidades asociadas a esta zona:

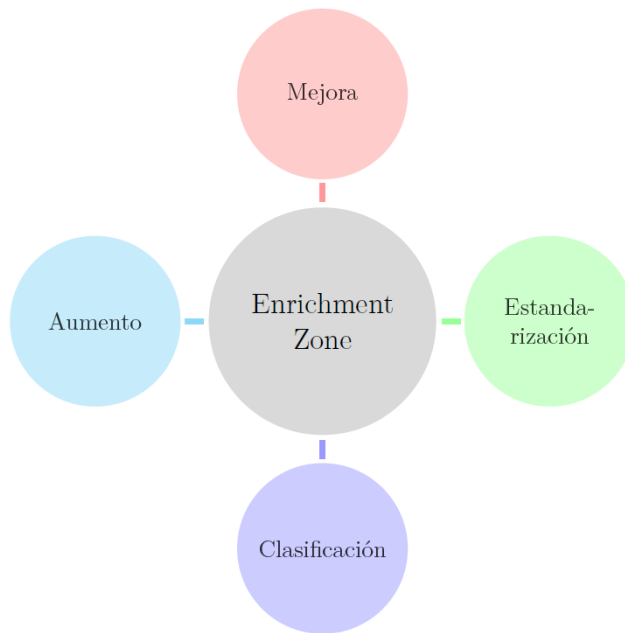


Figura 2.11: Funcionalidades asociadas a la *Enrichment Zone*.

3. *The Data Hub Zone*: Zona encargada del almacenamiento de los datos una vez han sido limpiados y procesados, constituyendo el final del flujo de datos dentro del Nivel de Gestión. En esta zona se llevan a cabo diversos procesos de búsqueda y recuperación de la información, a través de diversas herramientas, como puede ser Apache Lucene. Estos procesos hacen posible llevar a cabo algunos descubrimientos, basándose para ello en los grandes conjuntos de metadatos almacenados con anterioridad. En esta zona es posible almacenar la información utilizando bases de datos relacionales, como Oracle o MS SQL Server, así como utilizando arquitecturas no relacionales, tales como Mongo DB o HBase.

**Nivel de Obtención (The Data Consumption Tier).** En este nivel es posible acceder a los datos almacenados tanto en la *Data Hub Zone* como en la *Raw Zone*, siendo capaces de llevar a cabo una gran variedad de análisis sobre los mismos. Estos datos pueden ser usados también mediante herramientas de visualización, algo muy usado en la actualidad para realizar seguimientos de mercancías, o incluso por aplicaciones externas conectadas a través de servicios Web. El acceso a estos datos se encuentra altamente regulado, siguiendo

una gran cantidad de controles de seguridad cuya función es evitar el acceso a usuarios sin los suficientes permisos. Al igual que en los anteriores niveles, este nivel se encuentra dividido en diversas zonas:

1. *The Data Discovery Zone*: Zona considerada como la principal puerta de entrada para los usuarios externos que desean acceder al Data Lake. En esta zona se lleva a cabo un registro de los eventos que ocurren sobre los datos, el cual, unido al seguimiento que se realiza gracias a los metadatos, facilita a los usuarios el análisis de los mismos.
2. *The Data Provisioning Zone*: Zona en la que los usuarios pueden obtener los datos almacenados en el Data Lake, pudiendo suministrarse tanto del *Data Hub Zone* como de la *Raw Zone*.

En la figura 2.12 es posible ver la estructura que forma el Nivel de Obtención previamente explicado.

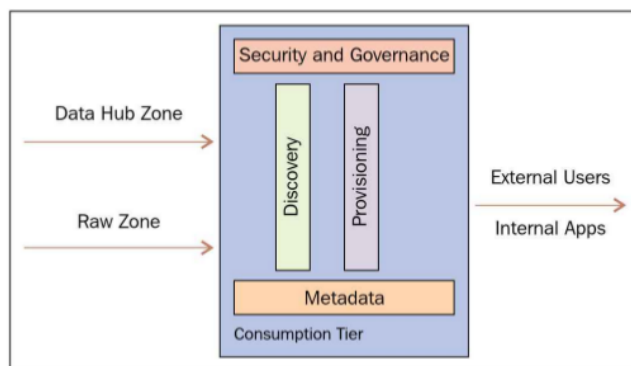


Figura 2.12: Estructura del Nivel de Obtención.

### Comparativa Data Lakes vs Data Warehouses

El concepto de Data Lake es utilizado, en algunas ocasiones, de manera equivocada como sustitutivo de los tradicionales Data Warehouses. Sin embargo existen grandes diferencias entre estas arquitecturas, las cuales radican principalmente en el tipo de datos que son capaces de almacenar, así como en el momento en que estos datos son formateados. Un Data Warehouse es una base de datos corporativa tradicional, la cual destaca por integrar y depurar la información antes de su almacenamiento, siguiendo procesos ETL, facilitando así su análisis desde un gran número de perspectivas a gran velocidad. Este hecho constituye una de las principales diferencias existentes entre los Data Lakes y los Data Warehouses, ya que de esta forma en los Data Warehouses los datos son almacenados con una estructura previa definida, mientras que los Data Lakes los datos son almacenados sin ser formateados, los datos se formatean únicamente en el momento en que son usados. En la figura 2.13 se puede ver el flujo que siguen los datos en los Data Warehouses.

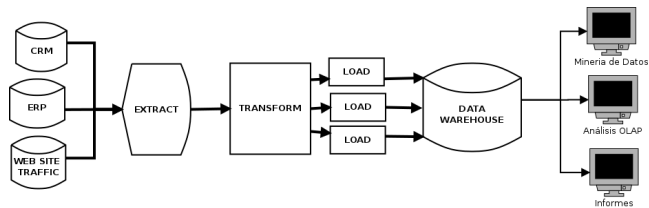


Figura 2.13: Flujo a seguir por los datos en un Data Warehouse durante ETL.

Otra gran diferencia existente entre estas arquitecturas, como previamente se ha comentado, radica en el tipo de datos que son capaces de almacenar. Este aspecto se encuentra fuertemente relacionado con la anterior diferencia, ya que el proceso ETL seguido en los Data Warehouse limita en gran parte los datos a almacenar, siendo almacenados de una única fuente y dotados de una estructura, perdiendo así gran cantidad de información. Frente a esto, los Data Lakes siguen un *schema on read*, pudiendo así almacenar cualquier tipo de datos. Este hecho constituye una gran ventaja de este tipo de arquitecturas, ya que en la actualidad el 80% de los datos que se recogen son no estructurados o semi-estructurados, por lo que no podrían ser almacenado en un Data Warehouse sin un procesamiento previo, el cual aumentaría los costes, reduciría la velocidad de carga de los datos e incluso podría ocasionar la pérdida de gran cantidad de información.

Otra de las principales diferencias que se pueden apreciar entre estas arquitecturas radica en las preguntas a realizar, es decir, si se conoce el valor de los datos previamente a su almacenamiento, como en los Data Warehouse, o si no se conoce y es necesario su estudio, como en los Data Lakes.

En cuanto al precio asociado al almacenamiento de estas arquitecturas cabe destacar el bajo coste de los Data Lakes, para los que es posible conseguir grandes capacidades de almacenamiento a bajo coste, gracias principalmente a soluciones open-source, como Hadoop, permitiendo un gran ahorro en licencias. En la tabla 2.1 se pueden ver algunas de las principales diferencias existentes entre estas arquitecturas.

| Data Warehouse                                | VS                      | Data Lakes   |
|---|-------------------------|--|
| Estructurados, procesados                     | DATOS                   | Estructurados, semi-estructurados y no estructurados |
| Orientado a escritura, <i>schema on write</i> | PROCESAMIENTO           | Orientado a lectura, <i>schema on read</i>           |
| Es caro almacenar grandes volúmenes de datos  | ALMACENAMIENTO          | Grandes capacidades de almacenamiento a bajo coste   |
| Normalmente una                               | FUENTES                 | Múltiples fuentes                                    |
| Estructura en estrella y en copo de nieve     | ESTRUCTURAS OPTIMIZADAS | No   |
| Madura  | SEGURIDAD               | Madurando (se encuentra en desarrollo)               |
| Profesionales de los negocios                 | USUARIOS                | Científicos de datos                                 |

Cuadro 2.1: Diferencias entre Data Lake y Data Warehouse.

En definitiva los Data Lakes no surgen como una arquitectura sustitutiva de los Data Warehouse, ya que como se ha podido observar pueden trabajar de manera conjunta, permitiendo a las empresas conseguir grandes mejoras en la gestión, almacenamiento y tratamiento de los enormes volúmenes de datos que son capaces de captar en la actualidad.

### 2.1.2. Arquitectura *LAMBDA*

Como se explica muy bien en el blog de Deusto Data [6] a la hora de procesar flujos de datos en tiempo real se puede no renunciar a la aproximación batch. Es decir, que podemos diseñar sistemas de Big Data que los integren a ambos, dando así una opción genérica y que para cada necesidad concreta, pueda emplear las tecnologías Batch o Tiempo Real. Nathan Marz publicó el libro «Big Data: Principles and best practices of scalable realtime data systems» en abril de 2015 para explicar todo esto y lo resumió en «la Arquitectura Lambda» que representamos a continuación:

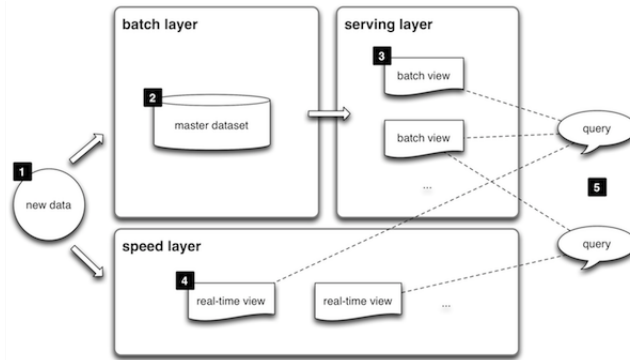


Figura 2.14: Arquitectura *LAMBDA*

El problema ante el que nos solemos encontrar al tratar con grandes volúmenes de datos es que no existe una técnica predefinida para hacerlo. Ya hemos visto con los paradigmas anteriores que el enfoque a adoptar para el procesamiento puede ser diferente. En esta ocasión, el creador de este paradigma Lambda, propone descomponer el problema en tres capas: Batch, Serving y Speed.

Supongamos que tenemos un proyecto de análisis de datos de una web con Google Analytics. Dejamos así preparada una función con todas las métricas que quisiéramos consultar (páginas vistas, visitantes únicos, búsquedas orgánicas, etc.) en una función con (URL, día). De esta manera, cuando queramos lanzar una consulta para un día determinado, solo necesitaríamos consultar la vista del rango de día donde hubiera caído el día concreto que nos interesa, y así, ágilmente, conseguir la información que nos interesa. En esta capa intervendrían, por ejemplo, Hadoop o Spark.

Posteriormente, tenemos la **capa de servicio**. La capa anterior, creaba esas vistas con los datos pre-computados. Pero, siempre necesitaremos una capa que cargue esas vistas en algún lugar que luego permita se puedan consultar. Esto se hace en la capa de servicio. Indexa las vistas creadas en la capa batch, las actualiza cada vez que llegan nuevas versiones de la capa batch. Dado que no recibe escrituras de datos aleatorias, esta capa es realmente robusta, predecible, fácil de configurar y operar. En esta capa, destaca *Cassandra*, por ejemplo.

Y, por último, aparece la **capa de velocidad**. Cuando alguien quiere acceder a una

consulta de datos, lo hace a través de una combinación de la capa de servicio y de la capa de velocidad. Esto lo podemos ver en el siguiente gráfico:

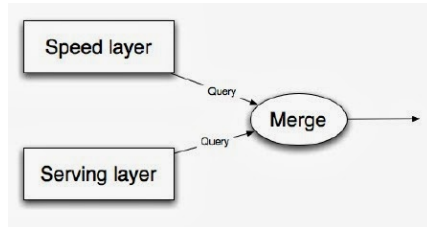


Figura 2.15: Capa de Velocidad y Servicio

La capa de velocidad es similar a la batch en el sentido que produce vistas a partir de los datos que recibe. Ahora bien, hay algunas diferencias clave. La más importante es que para conseguir altas velocidades, esta capa no mira a todos los nuevos datos de golpe. Solo actualiza aquellos nuevos datos que recibe, lo que le permite ofrecer de manera efectiva consultas de datos en tiempo real. Por eso se suele decir que esta capa actúa con actualizaciones incrementales, solo marcando como nuevo aquello que sea estrictamente necesario para ofrecer al usuario una vista en tiempo real.

Y todos esos módulos y funcionalidades es lo que nos permite disponer de una arquitectura Lambda que de manera completa representamos en la siguiente figura 2.16.

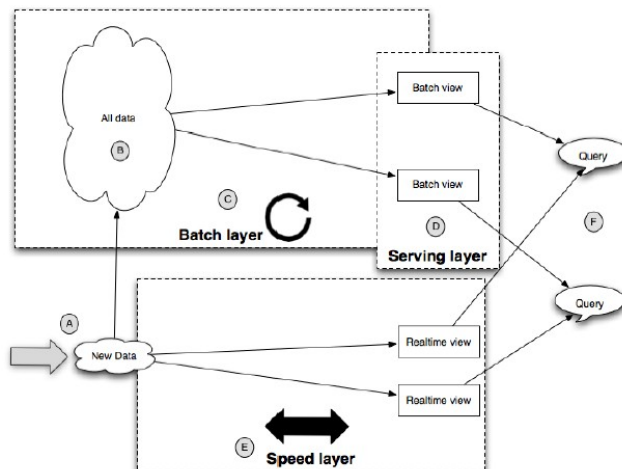


Figura 2.16: Arquitectura Lambda completa

### 2.1.3. Análisis del mercado Big Data

Con todas los elementos que ya vemos que veamos a necesitar para este desarrollo hemos realizado un breve análisis del mercado de Big Data en las empresas con proyectos de este tipo como es el caso de mi empresa *everis* con quien realizo este trabajo.

Cabe destacar de una manera muy importante la situación actual del Big Data. Como destacan muchos blogs tecnológicos el eterno debate sobre si el Big Data es una burbuja y sobre si esta explotará es un tema candente y edito estas lineas pocos días antes de la entrega de este proyecto para señalar que lo que a continuación se va a detallar ha cambiado por completo desde hace 3 meses al día en que esto se entrega y que a día de hoy el mercado me haría elegir otras herramientas y soluciones completamente diferentes como detallaré en la sección 6.2 de trabajo futuro.

En los proyectos Big Data que se están desarrollando en *everis* se están eligiendo soluciones open-source como *HortonWorks* o *Cloudera* (de hecho ambas se han fusionado en Octubre de 2018 [7] aunque de momento siguen ofreciendo sus servicios por separado, dicha fusión aun tardará en ser efectiva, o no ya que Cloudera está sufriendo un notable hundimiento de acciones con esta unión [8]).

Mientras que empresas como *Teléfonoica* indican que eligen *HortonWorks* para ofrecer Big Data as a service en España [9], desde *Everis* y su principal cliente en Big Data *Banco Santander* los proyectos se están montado sobre plataformas *Cloudera On Premise* ya que el nivel de madurez de la herramienta de gestión *Cloudera Manager* es superior a la de *HortonWorks*. Este hecho y estar trabajando en proyecto con *Cloudera* me hacen decantarme por *Cloudera* para este trabajo.

En otra posición están las soluciones que ofrencen *Amazon AWS* o *Microsoft Azure*, esta última algo inmadura pero creciendo y se tiene muy en cuenta a un futuro próximo.

Ya teniendo en más detalle de lenguajes y soluciones a utilizar, si entramos en portales de búsqueda de trabajo o en redes sociales profesionales como *LinkedIn* vemos que si buscamos trabajo en Big Data este perfil de profesionales [10] requieren unas capacidades y conocimientos concretos, habilidades técnicas y analíticas. En cuanto a las más técnicas, relacionadas con el conocimiento de ciertas herramientas que los profesionales deben manejar en cada momento, las más solicitadas por las empresas son: *Hadoop*, *Java*, *Cassandra*, *MongoDB*, *Kafka*, *Python*, *Spark*, *R*, *SAS* y *Hive*.

Elegido *Cloudera* tenemos incluido en su ecosistema otras herramientas básicas para nuestro proyecto, esto es que contamos con *HDFS*, *Hive* e *Impala* por defecto, así como con *Spark* que veremos a continuación en detalle por qué lo necesitamos.

Para el escenario del caso de uso que vamos a desarrollar nos es básico y fundamental el uso de un sistema de computación como *Apache Spark*. Este se basa en *Hadoop Map Reduce* y permite dividir o paralelizar el trabajo ya que normalmente se instala en un clúster de máquinas. La idea es que tengamos un número de máquinas, por ejemplo diez máquinas sobre las que se reparte el trabajo. Además trabaja en memoria, con lo que se consigue mucha mayor velocidad de procesamiento.

Spark nos proporciona API para Java, Scala, Python e incluso para R pero cabe destacar que en los perfiles que se demandan destaca el especialista o conocedor del lenguaje Scala ya que este lenguaje, a pesar de tener una mayor curva de aprendizaje frente a Python o tener una sintaxis más compleja y verbosa, es 10 veces más rápido [11] que procesando sobre Python además de que es un lenguaje funcional, es un hermano de Java que corre sobre la máquina virtual de Java pero está hecho por y para proporcionar acceso a las últimas funciones de Spark, ya que Apache Spark está escrito en Scala y resultaría más beneficioso para utilizar todo el potencial de Spark.

Otra de las ventajas de por qué lo elegimos es porque permite el procesamiento en tiempo real, con un módulo llamado Spark Streaming, que combinado con Spark SQL nos va a permitir el procesamiento en tiempo real de los datos. Conforme vayamos inyectando los datos podemos ir transformándolos y volcándolos.

El siguiente gráfico muestra cómo lo hace Spark para hacer transparente ese reparto de trabajo entre máquinas, funcionando una de ellas como el driver que divide el trabajo en tareas y las distribuye entre el resto de máquinas también conocidas en este ámbito como ejecutores.

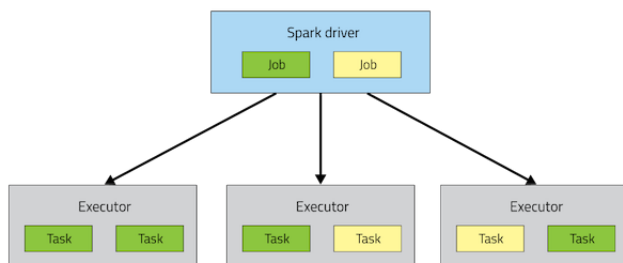


Figura 2.17: Arquitectura de Spark repartiendo el trabajo.

### SPARK on YARN

En concreto con Cloudera vamos a ejecutar Spark en *YARN* (Yet Another Resource Negotiator otro negociador de recursos”) es una tecnología de administración de clústeres. *YARN* combina un administrador central de recursos que reconcilia la forma en que las aplicaciones utilizan los recursos del sistema de Hadoop con los agentes de administración de nodo que monitorean las operaciones de procesamiento de nodos individuales del clúster.

Aquí cada ejecutor de Spark se ejecuta como un contenedor *YARN*. Donde MapReduce programa un contenedor y enciende una JVM para cada tarea, Spark hospeda múltiples tareas dentro del mismo contenedor. Este enfoque permite varios pedidos de magnitud más rápidos en el tiempo de inicio de la tarea. Esto queda definido en el siguiente gráfico que muestra la arquitectura de Spark en Yarn.



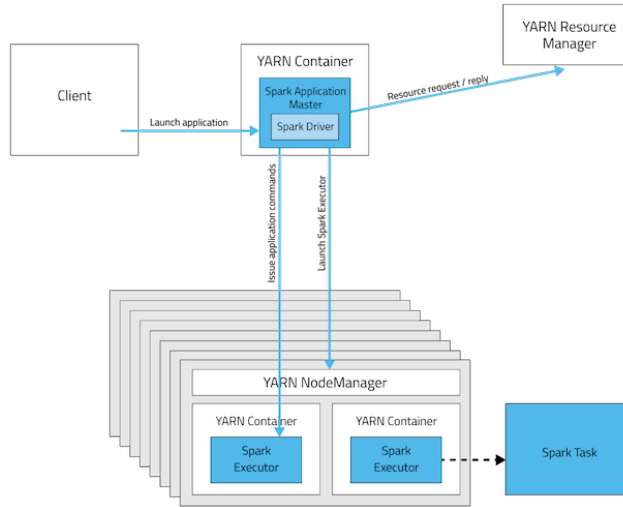


Figura 2.18: Arquitectura de SPARK on YARN.

## 2.2. Introducción al mercado de valores

Un mercado de valores, es una organización privada que brinda las facilidades necesarias para que sus miembros, atendiendo los mandatos de sus clientes, introduzcan órdenes y realicen negociaciones de compra y venta de valores, tales como acciones de sociedades o compañías anónimas, bonos públicos y privados, certificados, títulos de participación y una amplia variedad de instrumentos de inversión.

Los mercados de valores cumplen las siguientes funciones:

- Canalizan el ahorro hacia la inversión, contribuyendo así al proceso de desarrollo económico.
- Ponen en contacto a las empresas y entidades del Estado necesitadas de recursos de inversión con los ahorristas.
- Confieren liquidez a la inversión, de manera que los tenedores de títulos pueden convertir en dinero sus acciones u otros valores con facilidad.
- Certifican precios de mercado.
- Favorecen una asignación eficiente de los recursos.
- Contribuyen a la valoración de activos financieros.



Figura 2.19: Cómo funciona el mercado de valores.

Nosotros en concreto hemos elegido centrar este caso de uso a la segunda bolsa de valores automatizada y electrónica más grande de los Estados Unidos, el NASDAQ. EL porqué de escoger este y no al primero, el NYSE, es porque este comprende las empresas de alta

tecnología en electrónica, informática, telecomunicaciones, biotecnología además de muchas otras más.

Más de 7000 acciones de pequeña y mediana capitalización cotizan en la NASDAQ, nosotros afinaremos en 10 de estas empresas.

## 2.3. Naturaleza de los datos

Con todo lo anterior aterrizamos la idea en seleccionar los datos que necesitamos y explicaremos su naturaleza.

Primero haremos una diferenciación natural entre dos datos que vamos a necesitar, el dato en tiempo real de la cotización en bolsa de una compañía y el dato histórico de cada una de ellas.

Son muchas las herramientas y los portales web que difunden la información de todos los mercados de valores existente y en tiempo real. Los mas conocidos son el propio buscado de *Google* donde con poner, por ejemplo, el identificador abreviado de dicha compañía en su mercado (ejemplo: *AAPL* es la abreviatura para la compañía *Apple, Inc.* en el *NASDAQ*) este nos muestra todo sus datos relativos.

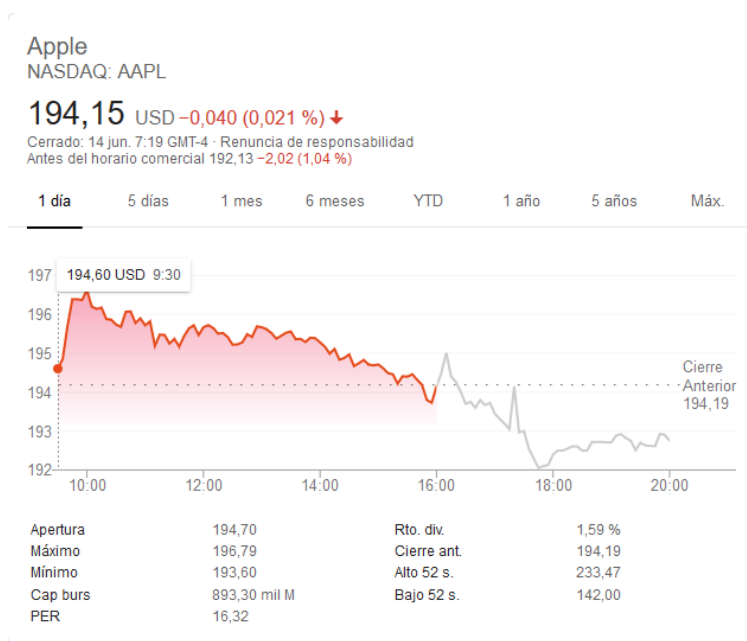


Figura 2.20: Resultado bursátil de Google para AAPL.

Otra de las más interesantes y que hemos barajado es la muy conocida y apoyada por los profesionales *Yahoo Finance*. Nos da información muy parecida a la anterior de Google pero nos permite navegar para obtener mucho más detalle, desde ese resumen hasta sus datos históricos pudiendo usar filtros pasando por Gráficos interactivos, estadísticas financieras, perfil de la empresa (sede, sector, organigrama detallado, etc), datos financieros con el estado de ingresos, el balance general y el flujo de caja, análisis financiero con beneficios, ingresos, BPA, crecimiento, etc. detalle de accionistas y un análisis detallado de su sostenibilidad.

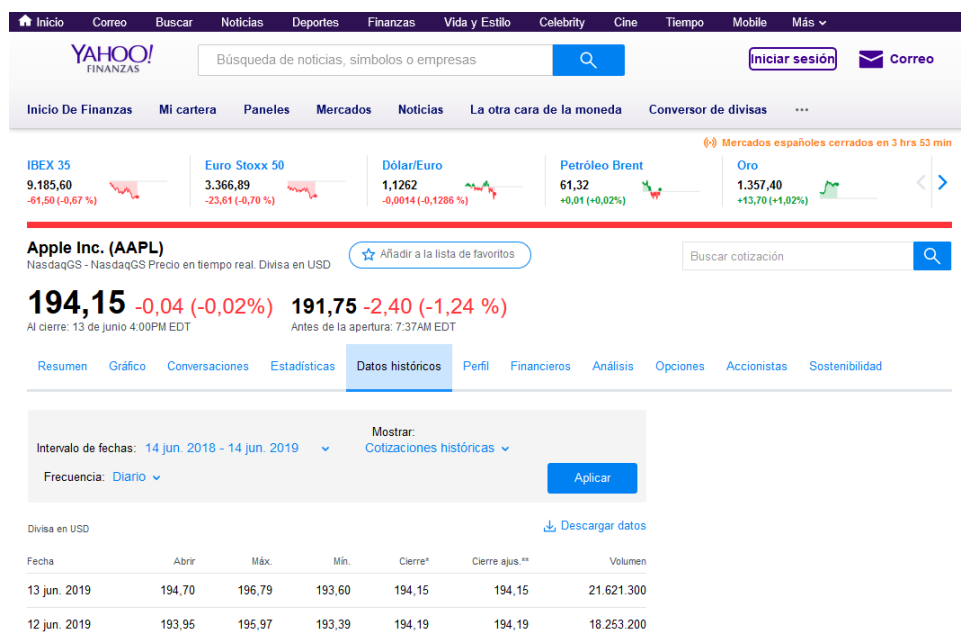


Figura 2.21: Resultado bursátil de Yahoo para AAPL.

Y entre las decenas de alternativas encontradas destacó una herramienta por encima del resto. La API de IEX es un conjunto de servicios ofrecidos por *The Investors Exchange (IEX)* para proporcionar acceso a los datos del mercado a desarrolladores e ingenieros. En los primeros meses del desarrollo de este trabajo el acceso a la API era gratuito, pero desde el 1 de Junio esta ha sufrido unos cambios tanto en el uso como en el coste pasando a necesitar contratar su plan básico que ellos llaman *LAUNCH* por \$19/mes.

La API está en continua mejora y desarrollo. Puede utilizarse para crear aplicaciones y servicios de alta calidad y cuenta con una herramienta que ellos llaman *IEX Stocks* que es un ejemplo de cómo se puede usar. Los datos proporcionados aquí provienen tanto del propio IEX como de múltiples fuentes de terceros. (Nota: IEX no ofrece ninguna garantía, representación o garantía en cuanto a la integridad o exactitud de los datos proporcionados).

Si navegamos por su documentación nos hacemos una idea del potencial de esta solución, de las numerosas alternativas que nos ofrece y sobre todo de la genericidad y simplicidad de uso.

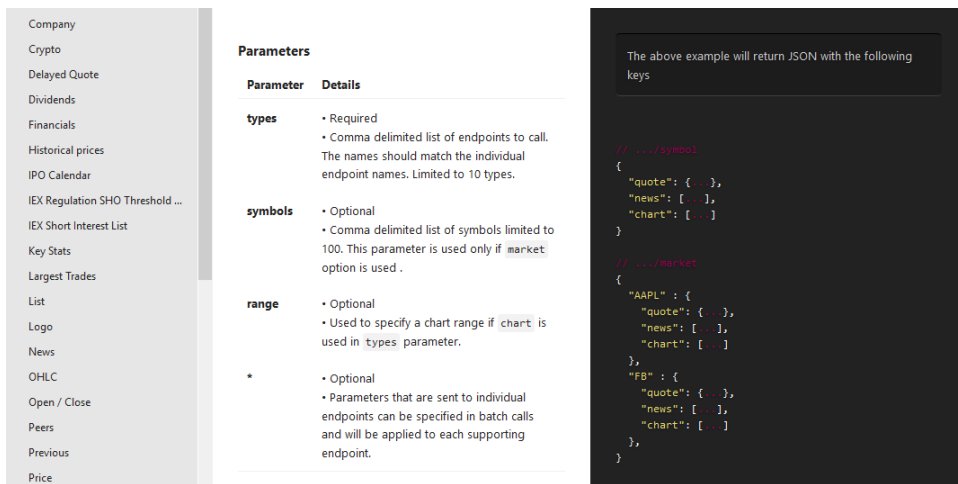


Figura 2.22: Documentación de la API de IEX.

### 2.3.1. Cotización en tiempo real

Con todas las alternativas que acabamos de ver, para cubrir las necesidades de lectura de la bolsa en tiempo real vemos que el uso de la API de *IEX Trading* nos cubre las necesidades de la siguiente manera.

Realizando una *HTTP request* haciendo uso de los parámetros:

- **“stock”** Petición a la API de stocks (valores).
- **“symbol=aapl”** Por ejemplo le pedimos la empresa *APPLE, INC.*
- **“type=quote”** De las múltiples opciones que tiene la API (cotización, gráfica, noticias, etc) elegimos la de cotización que es la que nos interesa hasta el momento.

Si lanzamos el siguiente ejemplo de petición HTTP [https://cloud.iexapis.com/stable/stock/aapl/quote?token=MY\\_TOKEN](https://cloud.iexapis.com/stable/stock/aapl/quote?token=MY_TOKEN), la API nos devolverá un fichero de tipo *JSON* (“notación de objeto de JavaScript” es un formato de texto sencillo para el intercambio de datos. Un subconjunto de la notación literal de objetos de JavaScript) con la siguiente información:

```

▼ quote:
  symbol: "AAPL"
  companyName: "Apple Inc."
  calculationPrice: "close"
  open: 194.73
  openTime: 1560432600280
  close: 194.15
  closeTime: 1560456000414
  high: 196.79
  low: 193.6
  latestPrice: 194.15
  latestSource: "close"
  latestTime: "June 13, 2019"
  latestUpdate: 1560456000414
  latestVolume: 21392544
  iexRealtimePrice: 0
  iexRealtimeSize: 0
  iexLastUpdated: 0
  delayedPrice: 194.15
  delayedPriceTime: 1560456000414
  extendedPrice: 193.64
  extendedChange: -0.51
  extendedChangePercent: -0.00263
  extendedPriceTime: 1560459597958
  previousClose: 194.19
  change: -0.04
  changePercent: -0.00021
  iexMarketPercent: 0
  iexVolume: 0
  avgTotalVolume: 30676573
  iexBidPrice: 0
  iexBidSize: 0
  iexAskPrice: 0
  iexAskSize: 0
  marketCap: 893298711250
  peRatio: 15.97
  week52High: 233.47
  week52Low: 142
  ytdChange: 0.2392380928842337
  
```

Figura 2.23: Ejemplo de JSON de la API de IEX para AAPL.

La explicación detallada de los datos que nos devuelve es la siguiente:

- **symbol:** Se refiere al símbolo bursátil. Es una cadena de caracteres.
- **companyName:** Se refiere al nombre de la compañía. Es una cadena de caracteres.
- **calculationPrice:** Se refiere al tipo de fuente del último precio. Puede ser tops”, sip”, previousclose” o close”). Es una cadena de caracteres.
- **open:** Se refiere al precio oficial de apertura. Es numérico.
- **openTime:** Se refiere al horario de intercambio de cotización oficial para la apertura. Es numérico.
- **close:** Se refiere al precio oficial de cierre. Es numérico.
- **closeTime:** Se refiere a la hora oficial de intercambio de cotización para el cierre. Es numérico.
- **high:** Se refiere al precio más alto del mercado. 15 minutos de retraso. Es numérico.
- **low:** Se refiere al precio más bajo del mercado. 15 minutos de retraso. Es numérico.
- **latestPrice:** Se refiere al precio más reciente. Es numérico.
- **latestSource:** Se refiere a la fuente de latestPrice. Puede ser Precio IEX en tiempo real”, Precio retrasado de 15 minutos”, Cierre” o Cierre anterior”. Es una cadena de caracteres.
- **latestTime:** Se refiere a una indicación temporal, legible por un humano, del último precio. El formato variará en función de latestSource. Es una cadena de caracteres.
- **latestUpdate:** Se refiere al tiempo de la actualización de latestPrice en milisegundos desde la medianoche del 1 de enero de 1970. Es numérico.
- **latestVolume:** Se refiere al volumen total de mercado de la acción. Es numérico.
- **iexRealtimePrice:** Se refiere al último precio de venta de las acciones en IEX. Es numérico.
- **iexRealtimeSize:** Se refiere al tamaño de la última venta de las acciones en IEX. Es numérico.
- **iexLastUpdated:** Se refiere a la última actualización de los datos en milisegundos desde la medianoche del 1 de enero de 1970 UTC o -1 o 0. Si el valor es -1 o 0, IEX no ha citado el símbolo en el día de negociación. Es numérico.
- **delayedPrice:** Se refiere al precio de mercado retrasado de 15 minutos durante las horas normales de mercado de 9:30 a 16:00. Es numérico.
- **delayedPriceTime:** Se refiere a la hora del precio de mercado diferido durante las horas normales de mercado de 9:30 a 16:00. Es numérico.
- **extendedPrice:** Se refiere al precio de mercado retrasado de 15 minutos fuera del horario normal de mercado de 8:00 a 9:30 y de 16:00 a 17:00. Es numérico.
- **extendedChange:** Se calcula utilizando extendedPrice desde computingPrice. Es numérico.
- **extendedChangePercent:** Se calcula utilizando extendedPrice desde computingPrice. Es numérico.
- **extendedPriceTime:** Se refiere a la hora del precio de mercado diferido fuera de los horarios normales de mercado de 8:00 a 9:30 y de 16:00 a 17:00. Es numérico.
- **change:** Se calcula usando el precio de cálculo de previousClose. Es numérico.
- **changePercent:** Se calcula utilizando calculationPrice de previousClose. Es numérico.
- **iexMarketPercent:** Se refiere al porcentaje de IEX del mercado en la acción. Es numérico.
- **iexVolume:** Se refiere a las acciones negociadas en las acciones de IEX. Es numérico.
- **avgTotalVolume:** Se refiere al volumen promedio de 30 días en todos los mercados. Es numérico.



- ***iexBidPrice***: Se refiere al mejor precio de oferta en IEX. Es numérico.
- ***iexBidSize***: Se refiere a la cantidad de acciones en la oferta en IEX. Es numérico.
- ***iexAskPrice***: Se refiere al mejor precio de venta en IEX. Es numérico.
- ***iexAskSize***: Se refiere a la cantidad de acciones en la solicitud en IEX. Es numérico.
- ***marketCap***: Se calcula en tiempo real mediante `calculationPrice`. Es numérico.
- ***peRatio***: Se calcula en tiempo real mediante `calculationPrice`. Es numérico.
- ***week52High***: Se refiere al precio más alto del mercado en las últimas 52 semanas. Es numérico.
- ***week52Low***: Se refiere al precio más bajo del mercado en las últimas 52 semanas. Es numérico.
- ***ytdChange***: Se refiere al porcentaje de cambio del precio desde el inicio del año hasta el cierre anterior. Es numérico.

Como puede verse es una gran cantidad de información y de mucho valor con la que con el suficiente conocimiento de la materia y tiempo se pueden desarrollar numerosas soluciones relacionadas. Nosotros de momento recogeremos todos estos datos y veremos hasta dónde podemos explotarlos en nuestro caso de uso.

### 2.3.2. Cotización histórica de una compañía

Para este segundo eje de nuestro escenario, de todas las alternativas que acabamos de ver, para cubrir la necesidad de tener el histórico de cotizaciones de las compañías de la bolsa vemos que el uso de la herramienta de *Yahoo Finance* nos cubre las necesidades de la siguiente manera.



Figura 2.24: Interfaz de Yahoo Finance para una cotización, en este caso la de Apple, Inc.

A la vista (en la figura superior 2.24) del panel que ofrece Yahoo Finance, la idea es que si vamos a tener lecturas en tiempo real de las empresas desde un día X, necesitaremos los

### 2.3. NATURALEZA DE LOS DATOS

datos históricos para esas mismas empresas desde su salida a bolsa hasta el día X-1, con el fin de después seguir formando ese histórico con el cierre de mercado de los datos en tiempo real.

Para ello haremos uso de la pestaña *Datos históricos* del menú superior, donde como vemos en la figura 2.25 podemos elegir un intervalo de fechas, y después de aplicar dicho filtro pulsando sobre *Descargar datos* tendremos un fichero de tipo CSV (comma-separated values) que podremos tratar con facilidad para ingestarlo en nuestro entorno y para nuestras futuras necesidades.

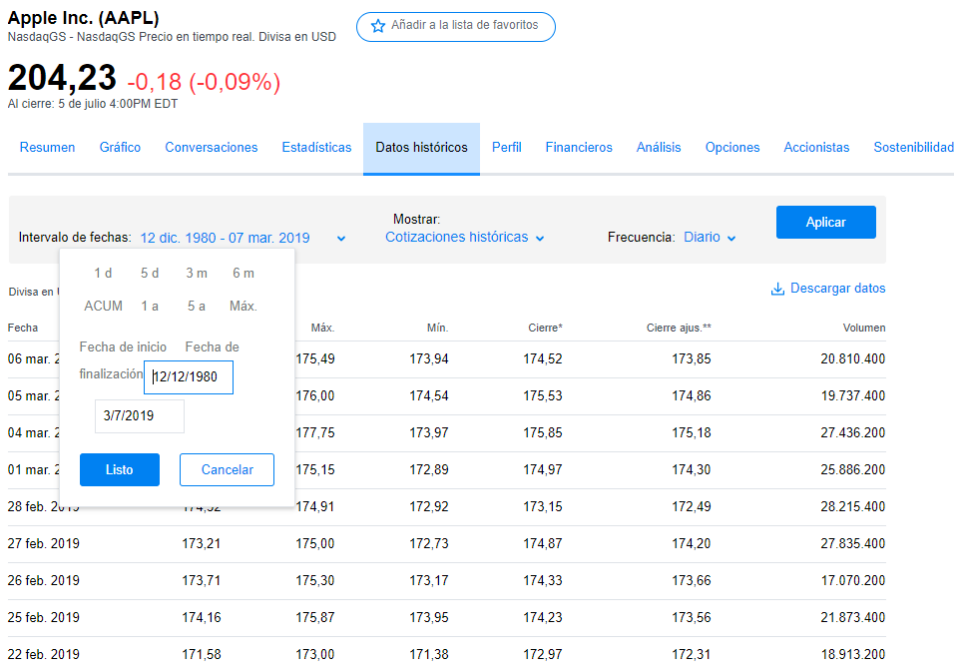


Figura 2.25: Interfaz de Yahoo Finance para los datos históricos de una empresa, en este caso la de Apple, Inc.

## 2.4. Análisis de tareas a realizar

Este proyecto que se pretende llevar a cabo requiere a groso modo de las tareas de:

- **Investigación:** Se identifica como una tarea esencial. Empezando por investigar el escenario, la disponibilidad de los datos, de hardware y de herramientas. Su fin es la obtención de los nuevos conocimientos necesarios para su aplicación a la solución del problema objetivo: este proyecto.
- **Formación:** Tarea muy común en el ámbito de los proyectos de investigación y en concreto de los proyectos tecnológicos. La necesidad en estos ámbitos de afrontar retos que son en su mayor parte nuevos genera la necesidad de formarse constantemente en esas nuevas herramientas o metodologías para su desarrollo.
- **Análisis:** Con esto se persigue realizar un análisis detallado sobre el sistema desarrollado en este proyecto, partiendo desde la captación de datos y continuando con el dashboard final donde consumiremos dichos datos. Se llevará a cabo una especificación de los diferentes tipos de requisitos existentes en un proyecto de desarrollo, así como el desarrollo de las historias de usuario asociadas a los requisitos de usuario especificados.
- **Diseño:** Trataremos de desarrollar el trabajo de diseño tanto del modelado de datos, de aplicación como del dashboard. Con esta tarea también se detallarán las arquitecturas lógica y física utilizadas para el desarrollo del proyecto, así como el diseño de las interfaces.
- **Desarrollo:** Aquí hablamos de la implementación del proyecto o del cómo. Se explicará cómo se ha implementado el sistema desarrollado a lo largo de este proyecto. Para ello se realizará una breve explicación sobre el desarrollo y la arquitectura Big Data asociada a este, así como de las diferentes herramientas y tecnologías que han sido utilizadas para el desarrollo del mismo.

## 2.5. Estimación planificación

En proyectos de investigación, como es el caso, es difícil dar una estimación certera de la planificación de este. Pero si bien es cierto que este proyecto empieza en Febrero, debe ser compaginado con mi actividad profesional en Everis, es susceptible a retrasos de planificación pero con el conocimiento que ya me ha aportado mi experiencia en la empresa sobre muchas de las técnicas que se van a utilizar, no debería haber problemas para conseguir los objetivos de este trabajo entre los meses de junio y julio, dando una estimación de unos 6 meses de trabajo.

Donde a priori caben menos dudas es en la elección de la metodología a utilizar. Aunque esto ya se detallará más adelante en su apartado correspondiente señalo desde un principio el uso de metodologías ágiles para la realización de este proyecto. Tanto por su auge como por estar acostumbrado a ellas en mi entorno laboral, por conocimiento de lo que suponen y porque la naturaleza de este proyecto aplica a la perfección a ese tipo de metodología: Los proyectos que siguen esta metodología parten de una visión general del producto que se desea desarrollar, a partir de la cual se va especificando y detallando la funcionalidad a realizar.

## Capítulo 3

# Análisis

En este capítulo se persigue realizar un análisis detallado sobre el sistema desarrollado en este proyecto, partiendo desde de captación de datos en tiempo real acabando con la aplicación de visualización desarrollada. Pese a estar desarrollando un proyecto más orientado a la investigación, el hecho de desarrollar una dashboard con el fin de facilitar la visualización y el consumo de los datos bursátiles y así contar con una herramienta más a la hora de analizar la posible gran cantidad de información de la que se dispone. Esto hace que sea plausible poder realizar una pequeña fase de análisis del proyecto. Por esto a lo largo de este capítulo se llevará a cabo una especificación de los diferentes tipos de requisitos existentes en un proyecto de desarrollo, así como el desarrollo de las historias de usuario asociadas a los requisitos de usuario especificados, al trabajar siguiendo una metodología ágil. El principal elemento en torno al que gira todo el análisis del proyecto son los requisitos de usuario, los cuales se encargan de describir las interacciones entre los usuarios y la aplicación desarrollada.

A continuación especificaremos en detalle los objetivos asociados al proyecto y posteriormente se pasará a iniciar la fase de análisis describiendo las diversas limitaciones, tanto hardware como software e incluso en la información disponible para trabajar, con las que nos encontraremos a lo largo del proyecto.

## 3.1. Objetivos

En nuestro afán por realizar procesos básicos en el Big Data como ETLs, transformaciones, agregados, históricos y un panel final de visualización y consumo del dato pero siempre, como hemos dicho, aplicando tecnologías y herramientas punteras en el mercado laboral, describimos como objetivo principal el análisis del mercado de valores NASDAQ para diez empresas: *Apple*”, *Advanced Micro Devices (AMD)*”, *Amazon.com*”, *Cisco Systems*”, *Facebook*”, *Alphabet (Google)*”, *International Business Machines (IBM)*”, *Intel*”, *Microsoft*”, y *QUALCOMM*”.

¿Y cómo se va a hacer? En los siguientes párrafos se van a narrar los objetivos en detalle acompañando a la manera en que se cumplirán.

En una primera fase se ha pensado por crear una aplicación de descarga de datos de la bolsa que realice una ingesta más automatizada sobre el Data Lake, pero tras ver la facilidad de descarga de los datos históricos con *Yahoo Finance* se ha optado por una ingesta manual de estos datos sobre sus correspondientes tablas Hive creadas acorde a las necesidades.

Estos datos cargados en batch les aplicaremos procesos de tratamiento, limpieza, agregados, cierres, normalizaciones, creaciones de índices, etc. Como la idea es hacer comparaciones regresivas para ver cómo de mucho o poco estas empresas que actualmente cotizan se acercan a esos ejemplos (o índices referencia) bursátiles no debemos olvidarnos de la carga, también en modo Batch, de los datos de referencia.

En cuanto a la ingesta en tiempo real de los datos de cotización de las empresas se analiza en detalle si con *Apache NiFi* y los datos de la API de *IEX Trading* cubrimos todas las necesidades. La idea es crear diez flows en NiFi, uno por cada empresa, que nos dejen cada lectura de Json ingestada en nuestro Data Lake.

Se desarrolla un modelo Entidad-Relación para todos estos tipos de datos que tendremos, los datos de referencia, los datos históricos y los datos Real Time.

Teniendo en cuenta que el NASDAQ tiene un horario de 9:30 a 16:00 (15:30 a 22:00 horario peninsular) de lunes a viernes, al finalizar el mercado esos datos en tiempo real pasarán a ser datos históricos. Aquí se ve necesaria una aplicación Spark-Scala que realice este cierre de mercado. La idea es que se planifique su ejecución cada día tomando la última de las lecturas Real Time que tengan el campo `latestSource` igual a `Close`”, cogiendo los datos relativos al modelo de datos para ser escritos en su correspondiente tabla histórica.

Teniendo en el Data Lake los datos de referencia, los datos históricos que se van actualizando en cada cierre de mercado del NASDAQ y los datos en Real Time ya nos quedaría el paso final. Un panel, explotado con *Power BI*, donde veremos distintas opciones: un panel con un top de empresas que parecen caer, top de empresas que parecen subir notablemente, acceso al histórico de cada empresa, y diversos gráficos y KPIs de interés.

## 3.2. Limitaciones y restricciones de implementación

En el desarrollo de este proyecto se han podido encontrar ciertas limitaciones, siendo la principal el hecho de que sólo se va a disponer de datos de 10 empresas en concreto y solo de su cotización en bolsa en tiempo real descartando otros posibles datos interesantes para completar la herramienta como pueden ser estadísticas avanzadas adicionales como EBITDA, ratios, datos financieros clave, etc o por limitaciones de solicitudes de la API en su plan más básico. El hecho de tener únicamente la información mencionado no constituye la única limitación del sistema, ya que unido a ese problema se encuentra así como por limitaciones derivadas del hardware del que se dispone, al disponer de un clúster con un único nodo con una capacidad de almacenamiento y procesamiento menor que la cantidad de datos obtenidos, impidiendo así realizar un seguimiento de todo un ecosistema financiero en tiempo real. Pese a las limitaciones ya descritas se ha decidido continuar con el desarrollo del proyecto pensando en las diferentes fases de captación y procesamiento de los datos por si en un futuro se puede tener acceso a otras licencias más completas.

De una manera más específica, sobre la aplicación se han podido encontrar ciertas restricciones, derivadas del software al que se tiene acceso, al contar con una versión CDH 5.13 Standalone de Cloudera como entorno altamente desactualizada, ya que las versiones posteriores son de pago. El hecho de contar con una versión Cloudera antigua lleva consigo el hecho de contar con una versión del sistema operativo CentOS 6.9, la cual a su vez también se encuentra desactualizada y que entre otras cosas no permite una correcta integración con GitHub, imposibilitando la realización de pruebas de integración continua, al ser necesario para el uso de GitHub. Una vez descritas las limitaciones, tanto a nivel de proyecto como a nivel de aplicación, se pasará a realizar la especificación, anteriormente mencionada, comenzando por realizar una descripción de los actores.

### 3.3. Actores del Sistema

En esta sección se realizará la especificación asociada a los diferentes actores que interactúan con la aplicación desarrollada. Se denominan como actores a todas las entidades externas al sistema que interactúan con el mismo. Siguiendo esta definición es posible determinar que los actores pueden ser tanto personas físicas como otros sistemas con los que interactúa el sistema a desarrollar.

| <b>ACT-01</b> | <b>Usuario General</b>   |
|---------------|--|
| Versión       | 2.0 (16/05/2019)   |
| Descripción   | Se considera como usuario general a todo usuario capaz de acceder a la aplicación y disfrutar de sus servicios de visualización, no existe la necesidad de ningún tipo de registro |
| Comentarios   | Ninguno  |

Cuadro 3.1: Actor Usuario General

| <b>ACT-02</b> | <b>albguti</b>   |
|---------------|--|
| Versión       | 2.0 (16/05/2019)   |
| Descripción   | Actor secundario a través del que se interactúa con el fin de obtener los datos de la bolsa y con el que se ejecutan los distintos scripts y procesos en la máquina.<br>Homónimo del usuario de cloudera con el que se ejecuta toda la creación de bases de datos, tablas, modificación, ingesta, cruces, etc. |
| Comentarios   | Ninguno  |

Cuadro 3.2: Actor albguti



## 3.4. Metodología

En esta sección se persigue describir y detallar la metodología seguida a lo largo del proyecto a desarrollar, así como los motivos de su utilización. Una vez descrita la metodología a seguir en el proyecto se detallará su planificación temporal y el coste asociado al desarrollo del mismo.

Cabe destacar que al tratarse de un proyecto de investigación la planificación será un poco especial, siendo muy complejo realizar una estimación de costes previa, al no conocer con exactitud con que nos encontraremos a lo largo del proyecto.

En la actualidad se está viendo un gran crecimiento en el número de proyectos que utilizan metodologías ágiles, viviendo el auge de la gestión de proyectos “DevOps”. El auge de estas técnicas de gestión de proyectos se encuentra muy relacionado con los grandes y extremadamente rápidos cambios realizados en el mundo de la tecnología. Estos cambios tecnológicos suelen ocasionar una gran cantidad de cambios en los requisitos de un proyecto. Ante esta situación, las técnicas de gestión de proyectos tradicionales se encuentran con una gran cantidad de problemas derivados de su reducida capacidad de adaptación a cambios. Los cambios tecnológicos, junto con la gran cantidad de empresas dedicadas a ofrecer soluciones informáticas, han aumentado la necesidad de seguir estrategias que permitan el lanzamiento temprano de productos funcionales, no completos, ocasionando este otro problema a las técnicas de gestión de proyectos tradicionales. Para poner solución a estos problemas surgen las metodologías ágiles, las cuales ofrecen una mayor adaptabilidad a cambios y siguen estrategias orientadas a obtener una versión inicial del proyecto a mayor velocidad. Las metodologías ágiles se fundamentan sobre 4 pilares:

- Valorar más el software funcional que la documentación extensiva: La posibilidad de anticipar cómo será el funcionamiento del producto final, antes de haberlo desarrollado por completo, puede ofrecer un feedback muy interesante y estimulante. Este feedback permite generar ideas que serían muy difíciles de generar en una especificación detallada de requisitos en el inicio del proyecto.
- Valorar más la colaboración con el cliente que la negociación contractual: Los proyectos ágiles se encuentran orientados a proyectos de evolución continua, en los que sería imposible definir un documento inicial de requisitos que indique cómo debería ser el producto final.
- Valorar más a los individuos y su interacción que a los procesos y las herramientas: En los proyectos que siguen metodologías ágiles se persigue alcanzar las cotas deseadas de calidad basándose en los conocimientos del equipo de trabajo, más que en la calidad de las herramientas disponibles para trabajar, las cuales no deben perder importancia.
- Valorar más las respuestas a los cambios que el seguimiento de un plan: En el desarrollo de productos con requisitos inestables, en los que es inherente el cambio y la evolución rápida y continua, es mucho más valiosa la capacidad de respuesta que la de seguimiento de planes.

### 3.4. METODOLOGÍA

---

En la actualidad existen una gran cantidad de metodologías ágiles de trabajo, entre las que destacan Kanban y Scrum. Para el desarrollo de este proyecto se ha decidido utilizar una metodología ágil, más en concreto se ha decidido utilizar Scrum. La elección de una metodología ágil para la gestión del proyecto radica en la inestabilidad inherente a un proyecto de investigación, además de que, al ser un proyecto también realizado durante las prácticas para una empresa, era la única manera de gestionar los avances del proyecto. Dentro de las metodologías ágiles se ha decidido utilizar Scrum, ya que es la metodología ágil más conocida por el equipo de trabajo. Scrum es una metodología ágil caracterizada por:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.
- Solapamiento de las diferentes fases del desarrollo, en lugar de realizarlas una tras otra en un ciclo secuencial o de cascada.

Los proyectos que siguen esta metodología parten de una visión general del producto que se desea desarrollar, a partir de la cual se va especificando y detallando la funcionalidad a realizar. En esta metodología cada ciclo de desarrollo, llamado sprint, finaliza con la entrega operativa del producto, siendo la duración recomendada de un sprint menor de un mes. En scrum se suelen realizar reuniones diarias de seguimiento del proyecto, de corta duración, con el fin de monitorizar el avance del mismo. En este proyecto, a diferencia de un scrum clásico, las reuniones de seguimiento son semanales, ya que los participantes en las reuniones, mi tutor de prácticas en la empresa, trabajan en otros proyectos, imposibilitando por tiempo las reuniones diarias de seguimiento.

Al finalizar cada sprint se suele llevar a cabo una reunión, entre todas aquellas personas involucradas en el desarrollo del proyecto, con el fin de poder revisar el resultado del sprint y valorar el incremento conseguido en el mismo.

El marco de scrum se encuentra formado por 3 componentes, roles, artefactos y eventos, siempre girando en torno al sprint. A continuación se puede ver una tabla en la que se describen estos componentes.

| Componente | Tipos                               | Descripción   |
|------------|-------------------------------------|---|
| Roles      | Equipo scrum                        | Grupo de profesionales que realizan el incremento en cada sprint  |
|            | Product Owner                       | Persona encargada de tomar las decisiones del cliente   |
|            | Scrum Master                        | Responsable del cumplimiento de las reglas de un marco de scrum técnico   |
| Artefactos | Product Backlog                     | La lista ordenada de todo aquello que el propietario de producto cree que necesita el producto  |
|            | Sprint Backlog                      | La lista de las tareas necesarias para construir las historias de usuario que se van a realizar en un sprint  |
|            | Incremento                          | La parte de producto producida en un sprint, y tiene como característica el estar completamente terminado y operativo   |
|            | Sprint                              | El evento clave de scrum para mantener un ritmo de avance continuo es el sprint, el periodo de tiempo acotado de duración máxima de 1 semana, durante el que se construye un incremento del producto        |
| Eventos    | Reunión de planificación del sprint | En esta reunión se toman como base las prioridades y necesidades de negocio del cliente, y se determinan cuáles y cómo van a ser las funcionalidades que se incorporarán al producto en el siguiente sprint |
|            | Scrum diario (daily)                | Reunión diaria breve, de no más de 15 minutos, en la que el equipo sincroniza el trabajo y establece el plan para las siguientes 24 horas   |
|            | Revisión del sprint                 | Reunión realizada al final del sprint para comprobar el incremento  |
|            | Retrospectiva del sprint            | Reunión en la que se realiza un autoanálisis de la forma de trabajar e identifica fortalezas y puntos débiles   |

Cuadro 3.3: Explicación componentes de scrum

## 3.5. Product Backlog

En esta sección se persigue especificar y describir las diferentes historias de usuario que deberán ser cumplidas al final del desarrollo de este proyecto. Se puede definir una historia de usuario como cualquier actividad que podrá realizar el usuario mediante interacciones con el sistema a desarrollar. En definitiva las historias de usuario materializan progresivamente los objetivos de negocio. Las historias de usuario se identificarán en el product backlog, el cual consiste en una lista priorizada de requisitos que representa la visión y expectativas del cliente respecto a los objetivos que tienen sobre el proyecto.

A continuación, en la tabla 3.4, se podrá ver el Product Backlog de las historias de usuario que se tendrán en cuenta en el desarrollo de la herramienta desarrollada en este proyecto, así como una breve descripción de las mismos.

| ID    | Descripción  |
|-------|--|
| US-01 | El usuario general podrá visualizar un listado de las diferentes empresas, de las que se tienen datos, en el que se mostrarán los datos históricos de cotización en bolsa. |
| US-02 | El usuario general podrá visualizar los datos de cotización de cada empresa en tiempo real con un intervalo de 60 segundos de actualización.                               |
| US-03 | El usuario general podrá visualizar, en un gráfico descriptivo, la previsión a futuro de la cotización de cada empresa.  |
| US-04 | El usuario general podrá filtrar las empresas a mostrar en el panel.   |
| US-05 | El usuario general podrá filtrar los datos de cotización a mostrar en función de una fecha.  |

Cuadro 3.4: Product Backlog

### 3.5.1. Historias de Usuario

En este apartado se realizará una especificación detallada de los historias de usuario, anteriormente listadas en el Product Backlog. Esta técnica es altamente utilizada en aquellos proyectos, que como este, son planificados utilizando metodologías ágiles. Una historia de usuario es la unidad de trabajo más pequeña que se puede identificar en el marco de las metodologías ágiles, constituyendo un objetivo final expresado desde el punto de vista de un usuario final del sistema a desarrollar. Este tipo de técnicas han adquirido una gran importancia en los últimos años, comenzando a ser puestas en práctica en una gran cantidad de proyectos.

Las historias de usuario se pueden agrupar en bloques, de mayor tamaño, surgiendo así

las épicas y las iniciativas. A continuación, en la figura 3.1, se podrá ver la jerarquía existente entre los elementos anteriormente descritos.

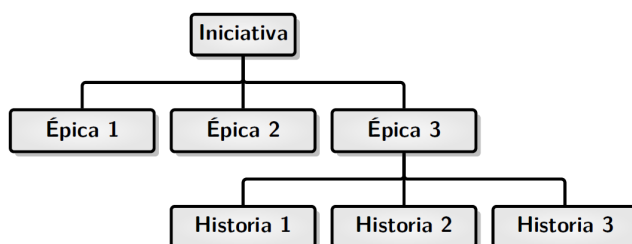


Figura 3.1: Jerarquía de las historias de usuario.

Una vez conocida la jerarquía existente entre los diferentes elementos utilizados para tratar de entender los requisitos de usuario utilizando historias de usuario se procederá a especificar en detalle los requisitos de usuario anteriormente descritos. El proyecto a desarrollar puede dividirse en 2 épicas con sus historias de usuario correspondientes asociadas a estas, las cuales se detallan a continuación, al igual que las historias de usuario.

### 3.5. PRODUCT BACKLOG

| ID Épica                |  | Descripción Épica |           |              |             |
|-------------------------|--|-------------------|-----------|--------------|-------------|
| EP-01                   | El usuario general podrá visualizar todos los datos disponibles de las diferentes empresas de las que se tienen datos. Se mostrarán los datos históricos de cotización en bolsa así como los datos en tiempo real. |                   |           |              |             |
| ID Historia             | Descripción  | Estatus           | Prioridad | Estimaciones | Propietario |
| US-01                   | El usuario general podrá visualizar un listado de las diferentes empresas, de las que se tienen datos, en el que se mostrarán los datos históricos de cotización en bolsa.   |                   |           |              |             |
| Criterios de aceptación |  |                   |           |              |             |
| 1                       | Mostrar histórico  |                   |           |              |             |
| Dado que                | Utilizo el dashboard   |                   |           |              |             |
| Cuando                  | Accedo al panel correspondiente  |                   |           |              |             |
| Entonces                | Se muestra un gráfico descriptivo con los datos históricos de la empresa   |                   |           |              |             |
| ID Historia             | Descripción  | Estatus           | Prioridad | Estimaciones | Propietario |
| US-02                   | El usuario general podrá visualizar los datos de cotización de cada empresa en tiempo real con un intervalo de 60 segundos de actualización  |                   |           |              |             |
| Criterios de aceptación |  |                   |           |              |             |
| 1                       | Mostrar datos real time  |                   |           |              |             |
| Dado que                | Utilizo el dashboard   |                   |           |              |             |
| Cuando                  | Accedo al panel correspondiente  |                   |           |              |             |
| Entonces                | Se muestra un gráfico descriptivo con los KPIs en tiempo real de la empresa y se actualizan  |                   |           |              |             |
| ID Historia             | Descripción  | Estatus           | Prioridad | Estimaciones | Propietario |
| US-03                   | El usuario general podrá visualizar, en un gráfico descriptivo, la previsión a futuro de la cotización de cada empresa.  |                   |           |              |             |
| Criterios de aceptación |  |                   |           |              |             |
| 1                       | Mostrar previsión  |                   |           |              |             |
| Dado que                | Utilizo el dashboard   |                   |           |              |             |
| Cuando                  | Accedo al panel correspondiente  |                   |           |              |             |
| Entonces                | Se muestra un gráfico descriptivo con la previsión de la cotización de la empresa  |                   |           |              |             |

Cuadro 3.5: Épica 01

| ID Épica                |   | Descripción Épica |           |              |             |
|-------------------------|---|-------------------|-----------|--------------|-------------|
| EP-02                   | El usuario general podrá aplicar filtros por empresa y por fecha a los paneles de visualización.    |                   |           |              |             |
| ID Historia             | Descripción   | Estatus           | Prioridad | Estimaciones | Propietario |
| US-04                   | El usuario general podrá filtrar las empresas a mostrar en el panel.                                |                   |           |              |             |
| Criterios de aceptación |   |                   |           |              |             |
| 1                       | Filtrar por empresa   |                   |           |              |             |
| Dado que                | Utilizo el dashboard  |                   |           |              |             |
| Cuando                  | Accedo al panel correspondiente   |                   |           |              |             |
| Entonces                | Se muestran los datos tanto de gráficos como KPIs en dicho panel solo para la empresa seleccionada  |                   |           |              |             |
| ID Historia             | Descripción   | Estatus           | Prioridad | Estimaciones | Propietario |
| US-05                   | El usuario general podrá filtrar los datos de cotización a mostrar en función de una fecha.         |                   |           |              |             |
| Criterios de aceptación |   |                   |           |              |             |
| 1                       | Filtrar por fechas  |                   |           |              |             |
| Dado que                | Utilizo el dashboard  |                   |           |              |             |
| Cuando                  | Accedo al panel correspondiente   |                   |           |              |             |
| Entonces                | Se muestran los datos tanto de gráficos como KPIs en dicho panel solo para las fechas seleccionadas |                   |           |              |             |

Cuadro 3.6: Épica 02

Una vez especificadas las diferentes historias de usuario que se desarrollarán en nuestro proyecto se pasara a realizar un listado de las diferentes tareas realizadas en cada sprint del proyecto, gracias al Sprint Backlog.

### 3.6. Sprint Backlog

En esta sección se persigue detallar las diferentes tareas que se han llevado a cabo en cada uno de los sprint del proyecto. Esta lista hace las veces de plan mediante el que se persiguen completar los objetivos, así como demostrar su consecución a los clientes en cada iteración.

En este proyecto los sprint tendrán una duración de 2 semanas. Sin embargo cabe destacar la existencia de un gran lapsus de tiempo en el que el proyecto estuvo parado, principalmente por su sincronización con el paso de beca a contraro en la empresa y por problemas personales graves. La idea inicial era comenzar el proyecto en febrero y se estableció un sprint inicial el 1 de marzo siendo un primer sprint de cuatro semanas y no de dos, debido a la necesidad de tener datos disponibles para posteriormente realizar el proyecto. A continuación se muestra una tabla en la que se especificarán las diversas tareas que componen el Sprint Backlog.

| Sprint Backlog |       |  |            |             |            |
|----------------|-------|--|------------|-------------|------------|
| Sprint         | Tarea | Descripción  | Est. Horas | P. Historia | Estado     |
| 1              | T-01  | Investigar el sistema bursátil del mercado de valores                              | 12         | 1           | Completado |
|                | T-02  | Analizar y documentar las aplicaciones que utilizan y proveen de datos de la bolsa | 30         | 1           | Completado |
|                | T-03  | Crear una aplicación de descarga de datos de la bolsa                              | 16         | 0,5         | Completado |
|                | T-04  | Configurar el servidor   | 4          | 0,5         | Completado |
| 2              | T-05  | Investigación sobre Data Lake, fase inicial  | 40         | 2           | Completado |
|                | T-06  | Documentación introducción   | 11         | 0,5         | Completado |
|                | T-07  | Investigación y documentación en profundidad de la API IEX Trading                 | 25         | 1           | Completado |
| 3              | T-08  | Investigación sobre Data Lake, fase final  | 50         | 2           | Completado |
|                | T-09  | Búsqueda de información asociada al NASDAQ   | 10         | 0,25        | Completado |
|                | T-10  | Análisis y documentación de las fuentes de información (Raw Data)                  | 40         | 1           | Completado |
|                | T-11  | Desarrollo de un modelo entidad relación y el diccionario de datos asociado        | 30         | 1           | Completado |
|                | T-12  | Instalación y breve formación en el uso de Cloudera                                | 30         | 0,75        | Completado |
|                | T-13  | Breve formación en HDFS, Hive e Impala   | 15         | 0,5         | Completado |



|    |      |   |    |      |            |
|----|------|---|----|------|------------|
| 4  |      |   |    |      |            |
|    | T-14 | Instalación de Java 1.8, Spark 2 y formación                                  | 30 | 1,5  | Completado |
| 5  | T-15 | Primer análisis de la información bursátil usando Hive                        | 45 | 2    | Completado |
|    | T-16 | Finalizar la primera fase de limpieza de información                          | 10 | 2    | Completado |
| 6  | T-17 | Instalación de NiFi y formación   | 25 | 0,5  | Completado |
|    | T-18 | Configuración de NiFi para lectura real time y volcado en Data Lake           | 20 | 0,5  | Completado |
|    | T-19 | Instalación de Power BI en local, configurar conexiones con Cloudera          | 6  | 2    | Completado |
|    | T-20 | Documentación y análisis inicial del dashboard a desarrollar                  | 25 | 0,5  | Completado |
|    | T-21 | Gestión de riesgos  | 5  | 0,25 | Completado |
| 7  | T-22 | Diseño de interfaces  | 10 | 0,5  | Completado |
|    | T-23 | Creación arquitectura   | 16 | 1    | Completado |
|    | T-24 | Formación en Power BI   | 10 | 0,5  | Completado |
|    | T-25 | Desarrollo del dashboard con Power BI   | 10 | 1    | Completado |
|    | T-26 | Test aplicación   | 1  | 0,25 | Completado |
|    | T-27 | Análisis II aplicación  | 5  | 1    | Completado |
|    | T-28 | Diseño II aplicación  | 5  | 0,5  | Completado |
|    | T-29 | Conexión de los datos de Cloudera a Power BI                                  | 5  | 0,25 | Completado |
| 8  | T-30 | Creación de tablón de históricos  | 12 | 0,25 | Completado |
|    | T-31 | Creación de tablón con KPIs real time   | 20 | 0,25 | Completado |
|    | T-32 | Implementación de los filtros   | 10 | 0,25 | Completado |
|    | T-33 | Pruebas sobre la aplicación desarrollada hasta ahora                          | 20 | 1    | Completado |
| 9  | T-34 | Formación avanzada en NiFi  | 15 | 1    | Completado |
|    | T-35 | Segunda fase de limpieza  | 20 | 1,5  | Completado |
|    | T-36 | Carga de datos batch  | 15 | 1,5  | Completado |
|    | T-37 | Análisis métricas y KPIs  | 5  | 0,5  | Completado |
|    | T-38 | Diseño métricas y KPIs  | 10 | 0,5  | Completado |
|    | T-39 | Actualizar la implementación de la visualización de métricas en la plataforma | 10 | 1    | Completado |
| 10 | T-40 | Pruebas finales aplicación  | 10 | 1    | Completado |
|    | T-41 | Revisiones finales aplicación y base de datos                                 | 15 | 1    | Completado |
|    | T-42 | Documentación final   | 65 | 1    | Completado |

Cuadro 3.7: Sprint Backlog

## 3.7. Planificación

Atrás en este capítulo se han descrito las diferentes historias de usuario de la plataforma, así como las tareas realizadas con el fin de desarrollar las diferentes herramientas del proyecto a realizar, descritas en el sprint backlog. Es posible utilizar herramientas de gestión de proyectos en las que puedan insertar las diferentes historias de usuario y tareas que componen los product y sprint backlog, generando a partir de ellas gran cantidad de gráficos que permiten realizar un seguimiento del proyecto. En la sección de planificación se detallarán los diferentes sprint del proyecto, mediante una serie de documentos y gráficos obtenidos a través de la herramienta en línea de gestión de proyectos *GanttPRO* que puede visitarse en su web <https://ganttpro.com/>.

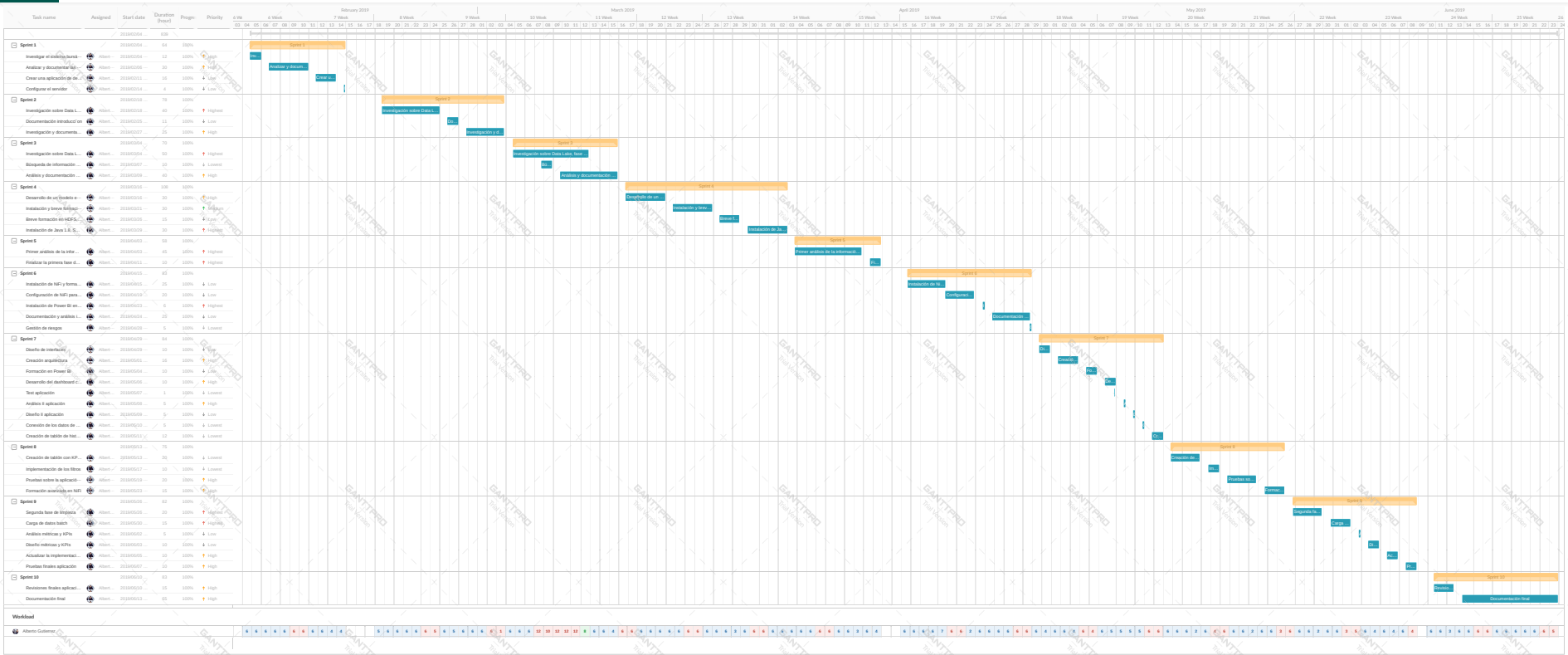
Como previamente se ha comentado, la planificación del proyecto se ha realizado utilizando una metodología ágil llamada Scrum. La planificación de este proyecto se ha desarrollado en 10 sprints, cuyas tareas están especificadas en el sprint backlog (ver tabla 3.7). Una vez creado el sprint backlog y habiendo dotado a las tareas de una estimación temporal a cada una, así como los puntos de historia de usuario asociados a estas, es posible desarrollar gráficos que muestren el avance temporal del proyecto. Como con anterioridad se ha comentado, la planificación en este proyecto es problemática. Este hecho deriva del carácter de investigación del proyecto al no tener como funcionalidad final el desarrollo de una herramienta software al uso, sino el análisis de la información bursátil del mercado de valores utilizando herramientas Big Data. El carácter de investigación asociado al proyecto, junto con la complejidad de las herramientas utilizadas, ha derivado en que la mayor parte de la planificación está destinada a la investigación del entorno bursátil y el Data Lake, así como a la formación tecnológica necesaria para analizar los datos de los que se dispone. Esto dificulta en gran parte la planificación del proyecto.

Este proyecto estará dividido en 10 sprints, siendo la duración media de estos de 2 semanas. Cabe destacar que el primer sprint en verdad ha sido más largo de las 2 semanas indicadas.

Se ha utilizado una licencia gratuita de la herramienta Online GanttPRO, acumulando varias cuentas gratuitas cuando estas iban venciendo sus 15 días de prueba, con el fin de realizar la gestión y planificación del proyecto. Esta aplicación permite obtener varios datos de seguimiento, destacando el gráfico de Gantt de Tareas y Sprints.

Gracias a este gráfico es posible ver la velocidad de desarrollo del proyecto en función de los puntos de historia estimados en el sprint backlog. Este gráfico supone una alternativa al gráfico burndown, permitiendo analizar el conjunto de sprints del proyecto y ayudando a conocer la evolución del proyecto. A continuación se mostrará al completo una página en formato apaisado dicho gráfico. Cabe destacar en ese histograma que los datos asociados a la carga de trabajo (Workload) del recurso *Alberto Gutierrez* de los fines de semana que aparecen en rojo es debido a que la herramienta no permite establecer estos días como días laborables. Además el hecho de que la mayoría de días aparezcan en azul con 6 horas es por el hecho de habernos decantado por una planificación *pesimista* en el sentido en que se pretende llevar una holgura para tratar posibles retrasos, además del hecho que el horario establecido para el recurso *Alberto Gutiérrez* es variable por su disponibilidad al salir de su

jornada laboral en Everis y por otras obligaciones personales e indisposiciones que le pueden surgir.



### 3.8. Gestión de Riesgos

Es un hecho que los proyectos software, al igual que en casi todos los aspectos de la vida, se encuentran expuestos a una enorme cantidad de riesgos que pueden influir en gran medida a la consecución de los objetivos que se espera conseguir con el desarrollo del proyecto. Este hecho dota por tanto de una gran importancia a la identificación, análisis y tratamiento de los diferentes riesgos que pueden afectar a los proyectos software. El análisis y gestión de riesgos trata de ofrecer herramientas que faciliten al equipo software la comprensión y el entendimiento de la incertidumbre generada por los riesgos. Esta gestión adquiere una enorme importancia debido a que un riesgo no identificado es capaz de, en el mejor de los casos, producir un impacto negativo significativo en la fecha de entrega del proyecto o en su presupuesto. Este hecho conlleva que cada vez se destinen más recursos al establecimiento de un buen Plan de Gestión de Riesgos que se encarga de definir:

- Un organigrama para la gestión de riesgos.
- El proceso de identificación y análisis de riesgos.
- Las herramientas y técnicas a utilizar.
- Plantillas estandarizadas para la identificación y gestión de riesgos.
- Actividades de control de riesgos y periodicidad de las mismas.

Al ser este un Trabajo de Fin de Grado alguno de los aspectos del Plan de Gestión de Riesgos, anteriormente descritos, no serán tenidos en cuenta, como el control de riesgos, ya que al ser un proyecto de corta duración temporal no habrá mucho espacio a cambios en los riesgos. A continuación se llevará a cabo la identificación de los principales riesgos susceptibles de afectar a este proyecto.

| ID      | Título   |
|---------|--|
| RISK-01 | Retrasos en la planificación                     |
| RISK-02 | Desconocimiento de las tecnologías a utilizar    |
| RISK-03 | Cambios en la especificación de los requisitos   |
| RISK-04 | Problemas hardware en los servidores disponibles |
| RISK-05 | Falta de experiencia del equipo de desarrollo    |
| RISK-06 | Problemas de salud en el equipo de desarrollo    |
| RISK-07 | Llegar a un punto sin salida en la investigación |
| RISK-08 | Competencia                                      |

Cuadro 3.8: Listado de riesgos

La identificación de riesgos en un proyecto suele ser muy problemática, ya que es prácticamente imposible identificar todos los riesgos a los que se enfrenta el desarrollo del proyecto.

Una vez identificados los principales riesgos que se tendrán en cuenta en el proyecto, se debe pasar a realizar un análisis cualitativo de los riesgos. Una de las herramientas más

### 3.8. GESTIÓN DE RIESGOS

---

utilizadas con este fin es la Matriz de Probabilidad Impacto. Esta herramienta facilita la priorización de riesgos en función de su probabilidad de ocurrencia, así como del impacto asociado a los diferentes riesgos. A continuación se mostrarán los diferentes riesgos priorizados utilizando la Matriz de Probabilidad Impacto.

El siguiente paso a seguir, una vez realizada la priorización de riesgos, consiste en llevar a cabo actividades y medidas que permitan gestionar los riesgos. Con el fin de gestionar los riesgos se establecen dos tipos de planes:

- **Plan de mitigación:** Definida en el *PMBOK* [12] como el conjunto de acciones a través de las cuales se pretende reducir la probabilidad de ocurrencia de un riesgo.
- **Plan de contingencia:** Conjunto de acciones realizadas como respuesta a la producción de un riesgo. Es el ejemplo perfecto de un plan reactivo de gestión de riesgos, es decir, trata de reducir el impacto del riesgo una vez este ya ha ocurrido.

A continuación se detallaran los planes de mitigación y contingencia asociados a los riesgos anteriormente priorizados.

| ID       | Título   | Impacto  | Probabilidad | Prioridad |
|----------|--|----------|--------------|-----------|
| RISK-01  | Retrasos en la planificación                     | Alto     | Media        | 4         |
| RISK-02  | Desconocimiento de las tecnologías a utilizar    | Alto     | Alta         | 2         |
| RISK-03  | Cambios en la especificación de los requisitos   | Medio    | Baja         | 7         |
| RISK-04  | Problemas hardware en los servidores disponibles | Medio    | Media        | 6         |
| RISK-05  | Falta de experiencia del equipo de desarrollo    | Alto     | Media        | 5         |
| RISK-06  | Problemas de salud en el equipo de desarrollo    | Medio    | Muy Baja     | 8         |
| RISK-07  | Llegar a un punto sin salida en la investigación | Alto     | Media        | 3         |
| RISK-081 | Desarrollo del mismo producto por la competencia | Muy Alto | Media        | 1         |

Cuadro 3.9: Listado de riesgos priorizados

| ID                        | Título                       | Descripción  |
|---------------------------|------------------------------|--|
| RISK-01                   | Retrasos en la planificación | Es posible que existan retrasos, en cuanto a la planificación, a lo largo del desarrollo del proyecto. Estos retrasos pueden derivarse de una mala planificación inicial   |
| <b>Gestión del riesgo</b> |                              |  |
| Estrategia                |                              | Se persigue establecer una estrategia mediante la cual se evite el riesgo  |
| Plan de mitigación        |                              | Para evitar problemas derivados de una mala planificación temporal se ha decidido añadir una variable de holgura a la planificación, es decir, se ha hecho una planificación pesimista en la que se trata de dar tres días a mayores al final de cada sprint |
| Plan de contingencia      |                              | Incrementar el número de horas diarias de trabajo  |

Cuadro 3.10: Risk-01. Retrasos en la planificación

| ID                        | Título  | Descripción  |
|---------------------------|---|--|
| RISK-02                   | Desconocimiento de las tecnologías a utilizar | Al realizar un proyecto de investigación en el que tanto el ámbito del proyecto, como el entorno tecnológico es eminentemente novedoso es posible que existan problemas derivados del trabajo con tecnologías novedosas, al no haber sido estudiadas en la carrera |
| <b>Gestión del riesgo</b> |   |  |
| Estrategia                |   | Se persigue establecer una estrategia mediante la cual se prevenga el riesgo   |
| Plan de mitigación        |   | Con el fin de prevenir estos problemas antes de iniciar el proyecto se ha tratado de tomar una pequeña formación en las tecnologías a utilizar, gracias a plataformas de cursos online   |
| Plan de contingencia      |   | Incrementar el número de horas diarias de trabajo y contactar con los tutores del proyecto, ya que ellos se encuentran familiarizados con estas tecnologías al haberrealizado proyectos de similares características   |

Cuadro 3.11: Risk-02. Desconocimiento de las tecnologías

### 3.8. GESTIÓN DE RIESGOS

| ID                        | Título                                     | Descripción   |
|---------------------------|--|---|
| RISK-03                   | Cambios en la especificación de requisitos | Al realizar un proyecto con un cliente real nos encontramos con la posibilidad de la ocurrencia de cambios en los requisitos del proyecto a desarrollar, más aún tratándose de un proyecto de investigación |
| <b>Gestión del riesgo</b> |  |   |
| Estrategia                |  | Se persigue establecer una estrategia mediante la cual se prevenga el riesgo  |
| Plan de mitigación        |  | Con el fin de prevenir y mitigar estos problemas se ha utilizado una metodología ágil, al destacar estas por su gran adaptabilidad a modificaciones en los requisitos                                       |
| Plan de contingencia      |  | Realizar los cambios pertinentes en los requisitos del proyecto   |

Cuadro 3.12: Risk-03. Cambios en la especificación de los requisitos

| ID                        | Título   | Descripción   |
|---------------------------|--|---|
| RISK-04                   | Problemas hardware en los servidores disponibles | Para el desarrollo de este proyecto son utilizados dos servidores ofertados por la Universidad de Valladolid. La caída o pérdida de estos servidores puede derivar en una pérdida irreparable en la información obtenida o en las aplicaciones encargadas de procesarla y utilizarla  |
| <b>Gestión del riesgo</b> |  |   |
| Estrategia                |  | Se persigue establecer una estrategia mediante la cual se prevenga el riesgo  |
| Plan de mitigación        |  | Con el fin de prevenir este riesgo, además de contar con los sistemas de backup con los que trabaja la universidad, se ha decidido almacenar en la nube, en aplicaciones como Google Drive o Dropbox, los avances realizados, así como en varios discos duros externos. Este hecho facilita la recuperación de información en caso de problema hardware |
| Plan de contingencia      |  | Realizar los cambios pertinentes en los requisitos del proyecto   |

Cuadro 3.13: Risk-04. Problemas hardware en los servidores disponibles



| ID                        | Título  | Descripción  |
|---------------------------|---|--|
| RISK-05                   | Falta de experiencia del equipo de desarrollo | La posible falta de experiencia del equipo de desarrollo del proyecto puede suponer retrasos y sobrecostos en el proyecto, así como una implementación poco eficaz del mismo |
| <b>Gestión del riesgo</b> |   |  |
| Estrategia                |   | Se persigue establecer una estrategia mediante la cual se prevenga el riesgo   |
| Plan de mitigación        |   | Con el fin de prevenir este tipo de problemas se incluye tiempo dedicado a la formación del personal dentro de la planificación del proyecto                                 |
| Plan de contingencia      |   | Incrementar los recursos temporales y económicos orientados a la formación del personal  |

Cuadro 3.14: Risk-05. Falta de experiencia del equipo de desarrollo

| ID                        | Título  | Descripción   |
|---------------------------|---|---|
| RISK-06                   | Problemas de salud en el equipo de desarrollo | Al igual que en cualquier trabajo la planificación temporal realizada puede verse truncada debido a la posibilidad de que el personal de trabajo tenga problemas de salud, entre otro tipo de problemas, que deriven en la baja del trabajador. |
| <b>Gestión del riesgo</b> |   |   |
| Estrategia                |   | Se ha decidido aceptar el riesgo, dado a la baja probabilidad de que el personal de trabajo del proyecto, en el corto periodo temporal en que se desarrolla, tenga que recurrir a solicitar alguna baja   |
| Plan de mitigación        |   | Ante los problemas que pueden dar lugar a bajas laborales no es posible disponer de ningún plan que las prevenga, más allá de promover hábitos de vida saludables   |
| Plan de contingencia      |   | Aceptar los retrasos producidos por la baja laboral   |

Cuadro 3.15: Risk-06. Problemas de salud en el equipo de desarrollo

### 3.8. GESTIÓN DE RIESGOS

| ID                        | Título   | Descripción   |
|---------------------------|--|---|
| RISK-07                   | Llegar a un punto sin salida en la investigación | Al realizar un proyecto de investigación en el que el objetivo final puede ser cambiante puede terminar dando lugar a un punto muerto, hecho muy frecuente en los proyectos de investigación debido a la falta de herramientas con las que desarrollarlos |
| <b>Gestión del riesgo</b> |  |   |
| Estrategia                |  | La estrategia seguida en este caso sería la aceptación, ya que ante un proyecto que ha llevado a un callejón sin salida no hay otra opción que aceptarlo y tratar de reenfocar el proyecto  |
| Plan de mitigación        |  | Es imposible prevenir que la investigación a seguir vaya a alcanzar un punto muerto   |
| Plan de contingencia      |  | Tratar de buscar una reorientación del proyecto, tratando así de reciclarlo y que sea útil en otro ámbito   |

Cuadro 3.16: Risk-07. Llegar a un punto sin salida en la investigación

| ID                        | Título   | Descripción  |
|---------------------------|--|--|
| RISK-08                   | Desarrollo del mismo producto por la competencia | En un mundo cada vez más informatizado y globalizado es cada vez más complejo que dos personas se encuentren desarrollando soluciones distintas a un mismo problema, un ejemplo de esto se puede ver entre Telegram y WhatsApp. Este riesgo es muy alto en este proyecto, al tener constancia de que otro equipo de trabajo en everis Madrid se encuentra desarrollando una solución similar |
| <b>Gestión del riesgo</b> |  |  |
| Estrategia                |  | Se persigue establecer una estrategia mediante la cual se evite el riesgo  |
| Plan de mitigación        |  | Es imposible saber si el producto desarrollado por la competencia es superior al tuyo o si se encuentra más desarrollado, para evitar problemas de competencia lo mejor es publicitar la herramienta y tratar de sacar una versión demo del producto antes que la competencia  |
| Plan de contingencia      |  | Tratar de desarrollar algo que nos diferencie del producto ya disponible   |

Cuadro 3.17: Risk-08. Desarrollo del mismo producto por la competencia

Una vez especificados los diferentes planes de mitigación y contingencia diseñados para cada riesgo se habrá completado la gestión de riesgos del proyecto a desarrollar.

### 3.9. Presupuesto Final

Al ser un proyecto de investigación es prácticamente imposible realizar una estimación inicial sobre el presupuesto asociado al desarrollo del proyecto a realizar, por ello se ha decidido pasar directamente a explicar el presupuesto final que supone el desarrollo de este proyecto, una vez finalizado. Para la realización del presupuesto del proyecto se cuenta con una duración de 6 meses. A continuación se podrá ver un desglose del presupuesto final del proyecto.

- **Presupuesto Hardware:** El presupuesto asociado al hardware del proyecto se prorrateará en función de la duración temporal del proyecto y la vida útil de los componentes. A continuación se podrá ver una tabla con el desglose del coste asociado al hardware.

| Componente              | Coste Total (euros) | Vida Útil (años) | % de uso | Coste Real (euros) |
|-------------------------|---------------------|------------------|----------|--------------------|
| Ordenador personal      | 800                 | 5                | 10 %     | 80                 |
| Servidor VPS Cloudera   | 360                 | 1                | 50 %     | 180                |
| Servidor Almacenamiento | 0                   | 1                | 50 %     | 0                  |
| Conexión a Internet     | 600                 | 1                | 50 %     | 300                |
| <b>TOTAL</b>            |                     |                  |          | <b>560</b>         |

Cuadro 3.18: Coste componentes Hardware

- **Presupuesto Software:** En la tabla 3.19 se muestran los costes asociados a los productos Software utilizados para el desarrollo del proyecto realizado. Los valores de los costes software se calcularán siguiendo el mismo patrón que se ha seguido para realizar el presupuesto hardware. Cabe destacar que para la realización de este proyecto se ha optado por utilizar tecnología open-source, por lo que en gran parte es gratuito.

| Componente                     | Coste Total (euros) | Vida Útil (años) | % de uso | Coste Real (euros) |
|--------------------------------|---------------------|------------------|----------|--------------------|
| Debian 9                       | 0                   |                  |          | 0                  |
| CentOS 6.9                     | 0                   |                  |          | 0                  |
| Cloudera CDH                   | 0                   |                  |          | 0                  |
| API IEX Trading                | 239,88              | 1                | 50 %     | 119,94             |
| IntelliJ IDEA Community        | 0                   |                  |          | 0                  |
| Overleaf ( $\backslash$ LaTeX) | 0                   |                  |          | 0                  |
| <b>TOTAL</b>                   |                     |                  |          | <b>119,94</b>      |

Cuadro 3.19: Coste componentes Software

- **Presupuesto de mano de obra:** Para realizar el presupuesto asociado a la mano de obra del proyecto se ha contado con el salario correspondiente un único ingeniero informático, de 25.000 euros/año [13]. Para calcular el sueldo total a cobrar sería necesario multiplicar el número de horas invertidas en el proyecto por el salario por hora a

### 3.9. PRESUPUESTO FINAL

---

cobrar por la persona encargada de su desarrollo. A continuación se puede ver el coste asociado al personal:

$$25000\text{euros/año}; 1780\text{horas/año} \implies 14,04\text{euros/hora} \quad (3.1)$$

Una vez tenemos precio por hora asociado al personal, se podrá calcular el presupuesto asociado a la mano de obra final.

$$\text{SueldoTotal} = \text{precio/hora} * \text{hora} \implies \text{SueldoFinal} = 14,04\text{euros/hora} * 890\text{horas} \quad (3.2)$$

El salario final a percibir por la persona encargada del desarrollo del proyecto será de 12.495,60 euros.

Una vez se han calculado los diferentes costes asociados al proyecto es posible obtener el presupuesto total del proyecto realizado. A continuación puede verse el cálculo del coste total del proyecto.

$$\text{PresupuestoTotal} = C.\text{Hardware} + C.\text{Software} + C.\text{Personal} \quad (3.3)$$

$$\text{PresupuestoTotal} = 560 + 119,94 + 12495,60 = 13175,54\text{euros}. \quad (3.4)$$

## Capítulo 4

# Diseño

En este capítulo se persigue desarrollar el trabajo de diseño previo al desarrollo del dashboard. En este capítulo por tanto se detallará un modelo de datos que es la pidera angular del diseño de este proyecto a partir del cual podrán ser establecidas las arquitecturas lógica y física utilizadas para el desarrollo de dicho proyecto, así como el diseño de los paneles creados para el consumo y visualización de los datos modelados.

### 4.1. Modelo de Datos

En este capítulo se persigue realizar un seguimiento completo de la información desde su extracción hasta su posterior carga en una aplicación final. Para ello se analizarán y describirán en detalle las diversas fuentes de información a partir de las cuales son obtenidos los datos que se utilizarán a lo largo de este proyecto.

Una vez analizadas esas fuentes de información se desarrollará el modelo conceptual, con el cual se pretende representar la realidad en función de los datos previamente analizados. Por último se llevará a cabo un análisis de los diferentes cambios que se realizan sobre los datos con el fin de facilitar su uso en las aplicaciones finales que los utilicen, utilizando para ello un Mapa de Datos, con el fin de mantener un historial de cambios sobre los mismos.

#### 4.1.1. Raw Data

En esta sección se persigue describir las diversas fuentes de información, en su formato original (Raw Data), de las cuales se van a obtener datos que son susceptibles de ser utilizados para dotar de contenido a la arquitectura a desarrollar. Una vez analizada la información de la que se dispone se seleccionarán los campos cuya información será utilizada en este proyecto, justificando su elección.

Las fuentes de información gracias a las que es posible dotar de contenido a dicha arquitectura pueden ser divididas en información asociada a los datos históricos (tanto de las empresas a analizar como de las empresas referencia) e información recogida en Real Time.

#### Información asociada a los datos históricos

Los datos históricos asociados a las empresas son obtenidos a través de *Yahoo Finance* (un servicio de Yahoo! que proporciona información financiera y comentarios con enfoque en los mercados de los Estados Unidos. A febrero de 2010 contaba con más de 23 millones de visitantes). La elección de esta fuente de información se fundamenta en su simplicidad y precio al ser gratuita y totalmente accesible la descarga de datos históricos. El archivo que se descarga es un fichero de tipo CSV (*comma-separated values*, son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea) que tiene el siguiente aspecto:

```

1 Date,Open,High,Low,Close,Adj Close,Volume
2 1980-12-12,0.513393,0.515625,0.513393,0.513393,0.022919,117258400
3 1980-12-15,0.488839,0.488839,0.486607,0.486607,0.021723,43971200
4 1980-12-16,0.453125,0.453125,0.450893,0.450893,0.020129,26432000
5 1980-12-17,0.462054,0.464286,0.462054,0.462054,0.020627,21610400
6 1980-12-18,0.475446,0.477679,0.475446,0.475446,0.021225,18362400
7 1980-12-19,0.504464,0.506696,0.504464,0.504464,0.022520,12157600
8 1980-12-22,0.529018,0.531250,0.529018,0.529018,0.023616,9340800
9 1980-12-23,0.551339,0.553571,0.551339,0.551339,0.024613,11737600
10 1980-12-24,0.580357,0.582589,0.580357,0.580357,0.025908,12000800
11 1980-12-26,0.633929,0.636161,0.633929,0.633929,0.028300,13893600
12 1980-12-29,0.642857,0.645089,0.642857,0.642857,0.028698,23290400
13 1980-12-30,0.629464,0.629464,0.627232,0.627232,0.028001,17220000
14 1980-12-31,0.611607,0.611607,0.609375,0.609375,0.027204,8937600
15 1981-01-02,0.616071,0.620536,0.616071,0.616071,0.027503,5415200
16 1981-01-05,0.604911,0.604911,0.602679,0.602679,0.026905,8932000
17 1981-01-06,0.578125,0.578125,0.575893,0.575893,0.025709,11289600
18 1981-01-07,0.553571,0.553571,0.551339,0.551339,0.024613,13921600
19 1981-01-08,0.542411,0.542411,0.540179,0.540179,0.024115,9956800
20 1981-01-09,0.569196,0.571429,0.569196,0.569196,0.025410,5376000
21 1981-01-12,0.569196,0.569196,0.564732,0.564732,0.025211,5924800
22 1981-01-13,0.546875,0.546875,0.544643,0.544643,0.024314,5762400
23 1981-01-14,0.546875,0.549107,0.546875,0.546875,0.024414,3572800
24 1981-01-15,0.558036,0.562500,0.558036,0.558036,0.024912,3516800
25 1981-01-16,0.555804,0.555804,0.553571,0.553571,0.024712,3348800
26 1981-01-19,0.587054,0.589286,0.587054,0.587054,0.026207,10393600
27 1981-01-20,0.571429,0.571429,0.569196,0.569196,0.025410,7520800
28 1981-01-21,0.580357,0.584821,0.580357,0.580357,0.025908,3976000
29 1981-01-22,0.587054,0.591518,0.587054,0.587054,0.026207,8887200
30 1981-01-23,0.587054,0.589286,0.584821,0.584821,0.026108,2805600

```

Figura 4.1: Ejemplo de CSV en bruto de Yahoo Finance para la empresa Apple.

En la tabla 4.1 se podrán ver los diferentes campos que componen la información que recoge la descarga de datos históricos de Yahoo Finance y que son susceptibles de ser utilizados.

| ID | Campo     | Long | Nombre          | Tipo   | Descripción  |
|----|-----------|------|-----------------|--------|--|
| C1 | Date      | 10   | Fecha           | String | Fecha de la observación                            |
| C2 | Open      | 8    | Apertura        | Float  | Precio oficial de apertura del mercado en el día   |
| C3 | High      | 8    | Máximo          | Float  | El precio más alto del mercado en el día           |
| C4 | Low       | 8    | Mínimo          | Float  | El precio más bajo del mercado en el día           |
| C5 | Close     | 8    | Cierre          | Float  | Precio oficial de cierre del mercado en el día     |
| C6 | Adj Close | 8    | Cierre Ajustado | Float  | Precio de cierre ajustado por dividendos y splits. |
| C7 | Volume    | 8    | Volumen         | Float  | El volumen total de mercado de la acción en el día |

Cuadro 4.1: Campos asociados a los datos históricos

### Información recogida en Real Time

La información asociada a la lectura del mercado de valores en Real Time es obtenida a gracias a la API que proporciona *IEX Trading*. Es un conjunto de servicios ofrecidos por The

#### 4.1. MODELO DE DATOS

Investors Exchange (IEX) para proporcionar acceso a los datos del mercado a desarrolladores e ingenieros. Esta API, que desde el 1 de Junio ha sufrido cambios, al contratar su plan básico *LAUNCH* por \$19/mes nos dan un máximo de 5.000.000 mensajes al mes que nos es suficiente. Con ella es posible acceder a gran cantidad de información sobre las empresas que cotizan, pero nos centraremos, como ya se ha comentado, solo en los datos relativos a la cotización en tiempo real.

En la tabla 4.2 se podrán ver los diferentes campos que componen la información dinámica relativa a las lecturas en tiempo real de la bolsa.

| ID  | Campo            | Nombre                    | Tipo   | Descripción   |
|-----|------------------|---------------------------|--------|---|
| C8  | symbol           | Símbolo                   | String | Es el símbolo bursátil  |
| C9  | companyName      | Nombre de la compañía     | String | Es el nombre de la compañía   |
| C10 | calculationPrice | Precio de Cálculo         | String | Es el tipo de origen del último precio. Puede ser Tops, Sip, Previousclose o Close  |
| C11 | open             | Apertura                  | Float  | Es el precio oficial de apertura  |
| C12 | openTime         | Hora de Apertura          | Float  | Es el horario de intercambio de cotización oficial para la apertura   |
| C13 | close            | Cierre                    | Float  | Es el precio oficial de cierre  |
| C14 | closeTime        | Hora del cierre           | Float  | Es la a hora oficial de intercambio de cotización para el cierre  |
| C15 | high             | Máximo                    | Float  | Es el precio más alto del mercado para el día   |
| C16 | low              | Mínimo                    | Float  | Es el precio más bajo del mercado para el día   |
| C17 | latestPrice      | Último Precio             | Float  | Es el precio más reciente. El campo más importante para nosotros en el ámbito del tiempo real                                 |
| C18 | latestSource     | Último Origen             | String | Es la la fuente de latestPrice. Puede ser Precio IEX en tiempo real, Precio retrasado de 15 minutos, Cierre o Cierre anterior |
| C19 | latestTime       | Hora de la Última Lectura | String | Es una indicación temporal, legible por un humano, del último precio. El formato variaría en función de latestSource          |



|     |                       |                                |       |   |
|-----|-----------------------|--------------------------------|-------|---|
| C20 | latestUpdate          | Timestamp de la Última Lectura | Float | Es el tiempo de la actualización de latestPrice en milisegundos desde la medianoche del 1 de enero de 1970  |
| C21 | latestVolume          | Volumen Actual                 | Float | Es el volumen total de mercado de la acción   |
| C22 | iexRealttimePrice     | Precio Actual del IEX          | Float | Es el último precio de venta de las acciones en IEX   |
| C23 | iexRealttimeSize      | Volumen Actual del IEX         | Float | Es el tamaño de la última venta de las acciones en IEX  |
| C24 | iexLastUpdated        | Hora Actual del IEX            | Float | Es la última actualización de los datos en milisegundos desde la medianoche del 1 de enero de 1970 UTC -1 0. Si el valor es -1 o 0 IEX no ha citado el símbolo en el día de negociación |
| C25 | delayedPrice          | Precio Retrasado               | Float | Es el precio de mercado retrasado de 15 minutos durante las horas normales de mercado de 9:30 a 16:00   |
| C26 | delayedPriceTime      | Hora del Precio Retrasado      | Float | Es la hora del precio de mercado diferido durante las horas normales de mercado de 9:30 a 16:00   |
| C27 | extendedPrice         | Precio Extendido               | Float | Es el precio de mercado retrasado de 15 minutos fuera del horario normal de mercado de 8:00 a 9:30 y de 16:00 a 17:00   |
| C28 | extendedChange        | Cambio Extendido               | Float | Se calcula utilizando extendedPrice desde computingPrice  |
| C29 | extendedChangePercent | Porcentaje de Cambio Extendido | Float | Se calcula utilizando extendedPrice desde computingPrice  |
| C30 | extendedPriceTime     | Hora del Precio Extendido      | Float | Es la marca horaria de la última actualización de extendedPrice   |
| C31 | previousClose         | Cierre Anterior                | Float | Es la medida del anterior cierre  |

#### 4.1. MODELO DE DATOS

|     |                  |  |       |   |
|-----|------------------|--|-------|---|
| C32 | change           | Cambio                                   | Float | Se calcula usando el precio de cálculo de previousClose                               |
| C33 | changePercent    | Porcentaje de Cambio                     | Float | Se calcula utilizando calculationPrice de previousClose                               |
| C34 | iexMarketPercent | Porcentaje de Mercado del IEX            | Float | Es el porcentaje de IEX del mercado en la acción                                      |
| C35 | iexVolume        | Volumen del IEX                          | Float | Son las acciones negociadas en las acciones de IEX                                    |
| C36 | avgTotalVolume   | Volumen Medio del IEX                    | Float | Es el volumen promedio de 30 días entodos los mercados                                |
| C37 | iexBidPrice      | Oferta del IEX                           | Float | Es el mejor precio de oferta en IEX   |
| C38 | iexBidSize       | Tamaño de Oferta del IEX                 | Float | Es la la cantidad de acciones en la oferta en IEX                                     |
| C39 | iexAskPrice      | Mejor Precio de Venta del IEX            | Float | Es el mejor precio de venta en IEX  |
| C40 | iexAskSize       | Cantidad de Acciones Solicitadas del IEX | Float | Es la cantidad de acciones en la solicitud en IEX                                     |
| C41 | marketCap        | Capitalización del Mercado               | Float | Se calcula en tiempo real mediante calculationPrice                                   |
| C42 | peRatio          | Relación precio ganancias                | Float | Se calcula en tiempo real mediante calculationPrice                                   |
| C43 | week52High       | Máximo de las 52 Semanas                 | Float | Es el precio más alto del mercado en las últimas 52 semanas                           |
| C44 | week52Low        | Mínimo de las 52 Semanas                 | Float | Es el precio más bajo del mercado en las últimas 52 semanas                           |
| C45 | ytdChange        | Cambio del Año Hasta la Fecha            | Float | Es el porcentaje de cambio del precio desde el inicio del ao hasta el cierre anterior |

Cuadro 4.2: Campos asociados a los datos captados en tiempo real

Una vez analizados los campos que se pueden leer a través de esta plataforma se ha

decidido utilizar todos los campos en el desarrollo de este proyecto, aunque si bien es cierto que para el objetivo principal de este trabajo no son necesarios, pero no suponen un gran aumento de la volumetría y, sobre todo, así queda preparado para el trabajo futuro que aprovechará más campos.

### 4.1.2. Refined Data

En esta sección se lleva a cabo el diseño del Modelo Conceptual, a partir del cual se pretende describir el mini-mundo que afecta a este proyecto. Este objetivo lleva a que en esta sección se trate de representar el ámbito de las cotizaciones bursátiles de manera independiente a la implementación de la arquitectura a desarrollar. El diseño del Modelo Conceptual se encuentra soportado por documentación que incluye tanto el diagrama Entidad-Relación como el Diccionario de Datos generado. En las siguientes subsecciones se llevará a cabo una aproximación a estos dos grandes bloques del Modelado Conceptual.

#### Modelo Entidad-Relación

El Modelo Entidad-Relación persigue representar el mundo real, de una manera sencilla y libre de ambigüedades, con independencia de los aspectos físicos relacionados con la implementación de la arquitectura a desarrollar. Este modelo sigue una estructura top-down para realizar el diseño conceptual de cualquier base de datos.

Las primeras labores de análisis del problema han derivado en la creación del siguiente modelo entidad-relación, ver figura 4.2, gracias al cual se persigue mostrar de manera gráfica y sencilla el problema a solventar.



## Diccionario de Datos

El Diccionario de Datos trata de proveer de una descripción detallada de los datos gestionados por la Base de Datos, los metadatos. Es imposible pensar en una base de datos, o cualquier arquitectura Big Data, sin los metadatos, ya que sin un buen uso de ellos es muy complicado caracterizar los datos y conocer su significado. A continuación se puede ver el diccionario de datos asociado a este proyecto dividido por entidades.

- E1. Entidad Real Time:** Entidad encargada de representar los datos que describen a los datos leídos en tiempo real, correspondiéndose estos con los campos seleccionados asociados a la información de los datos anteriormente descritos. Para acceder a la información asociada a la descripción de este tipo de datos ver la tabla 4.2 en la sección Raw data.

| ID    | Nombre                | Tipo   | Restricc. | Fuente de información | ID Fuente |
|-------|-----------------------|--------|-----------|-----------------------|-----------|
| E1.1  | symbol                | string | Nullable  |                       | C8        |
| E1.2  | companyName           | string | Nullable  |                       | C9        |
| E1.3  | calculationPrice      | string | Nullable  |                       | C10       |
| E1.4  | open                  | double | Nullable  |                       | C11       |
| E1.5  | openTime              | long   | Nullable  |                       | C12       |
| E1.6  | close                 | double | Nullable  |                       | C13       |
| E1.7  | closeTime             | long   | Nullable  |                       | C14       |
| E1.8  | high                  | double | Nullable  |                       | C15       |
| E1.9  | low                   | double | Nullable  |                       | C16       |
| E1.10 | latestPrice           | double | Nullable  |                       | C17       |
| E1.11 | latestSource          | string | Nullable  |                       | C18       |
| E1.12 | latestTime            | string | Nullable  |                       | C19       |
| E1.13 | latestUpdate          | long   | Nullable  |                       | C20       |
| E1.14 | latestVolume          | long   | Nullable  |                       | C21       |
| E1.15 | iexRealtimePrice      | double | Nullable  |                       | C22       |
| E1.16 | iexRealtimeSize       | long   | Nullable  |                       | C23       |
| E1.17 | iexLastUpdated        | double | Nullable  |                       | C24       |
| E1.18 | delayedPrice          | double | Nullable  |                       | C25       |
| E1.19 | delayedPriceTime      | long   | Nullable  |                       | C26       |
| E1.20 | extendedPrice         | double | Nullable  |                       | C27       |
| E1.21 | extendedChange        | double | Nullable  |                       | C28       |
| E1.22 | extendedChangePercent | double | Nullable  |                       | C29       |
| E1.23 | extendedPriceTime     | long   | Nullable  |                       | C30       |
| E1.24 | previousClose         | double | Nullable  |                       | C31       |
| E1.25 | change                | double | Nullable  |                       | C32       |
| E1.26 | changePercent         | double | Nullable  |                       | C33       |
| E1.27 | iexMarketPercent      | double | Nullable  |                       | C34       |
| E1.28 | iexVolume             | long   | Nullable  |                       | C35       |
| E1.29 | avgTotalVolume        | long   | Nullable  |                       | C36       |
| E1.30 | iexBidPrice           | double | Nullable  |                       | C37       |
| E1.31 | iexBidSize            | long   | Nullable  |                       | C38       |

|       |             |        |          |     |
|-------|-------------|--------|----------|-----|
| E1.32 | iexAskPrice | double | Nullable | C39 |
| E1.33 | iexAskSize  | long   | Nullable | C40 |
| E1.34 | marketCap   | long   | Nullable | C41 |
| E1.35 | peRatio     | double | Nullable | C42 |
| E1.36 | week52High  | double | Nullable | C43 |
| E1.37 | week52Low   | double | Nullable | C44 |
| E1.38 | ytdChange   | double | Nullable | C45 |

Cuadro 4.3: Entidad Real Time

- E2. Entidad Historic:** Entidad encargada de representar los datos que describen las observaciones históricas de cada empresa. Para acceder a la información asociada a la descripción de este tipo de datos ver la tabla 4.1 en la sección Raw data.

| ID   | Nombre    | Tipo   | Restricciones      | Fuente de información | ID Fuente |
|------|-----------|--------|--------------------|-----------------------|-----------|
| E2.1 | date      | string | PK<br>Not Nullable | Yahoo Finance         | C1        |
| E2.2 | open      | double | Nullable           |                       | C2        |
| E2.3 | high      | double | Nullable           |                       | C3        |
| E2.4 | low       | double | Nullable           |                       | C4        |
| E2.5 | close     | double | Nullable           |                       | C5        |
| E2.6 | adj_close | double | Nullable           |                       | C6        |
| E2.7 | volume    | bigint | Nullable           |                       | C7        |

Cuadro 4.4: Entidad Historic

- E3. Entidad Bankruptcy:** Entidad encargada de representar los datos que describen las observaciones de las empresas de referencia para la identificación de la banarota. Para acceder a la información asociada a la descripción de este tipo de datos ver la tabla 4.5 en la sección Raw data.

| ID   | Nombre    | Tipo   | Restricciones      | Fuente de información | ID Fuente |
|------|-----------|--------|--------------------|-----------------------|-----------|
| E3.1 | date      | string | PK<br>Not Nullable | Yahoo Finance         | C1        |
| E3.2 | open      | double | Nullable           |                       | C2        |
| E3.3 | high      | double | Nullable           |                       | C3        |
| E3.4 | low       | double | Nullable           |                       | C4        |
| E3.5 | close     | double | Nullable           |                       | C5        |
| E3.6 | adj_close | double | Nullable           |                       | C6        |
| E3.7 | volume    | bigint | Nullable           |                       | C7        |

Cuadro 4.5: Entidad Bakruptcy

- E4. Entidad Correlation Mark:** Entidad encargada de almacenar los datos relativos a la conversión de la correlación de cada empresa con las empresas de referencia para la identificación de la banarota.

| ID   | Nombre           | Tipo   | Restricciones | Fuente de información | ID Fuente |
|------|------------------|--------|---------------|-----------------------|-----------|
| E3.1 | correlation_mark | double | Not Nullable  | Impala                | C46       |

Cuadro 4.6: Entidad Correlation Mark

- **R1. Relación Generates at end of market:** Esta relación se produce entre las entidades ‘Real Time’ e ‘Historic’ y surge para permitir asociar el paso de la observación del cierre de mercado de una empresa a su correspondiente histórico. La cardinalidad es la siguiente: muchos Real Time pueden generar 1 o ninguna Historic (0,1), mientras que una Historic puede ser generada por muchos Real Time (1,N).
- **R1. Relación Decides:** Esta relación se produce entre las entidades ‘Historic’ y ‘Bankruptcy’, generando una entidad ‘Correlation Mark’, La cardinalidad existente en esta relación es la siguiente: una Historic con una Bankruptcy solo puede formar una Correlation Mark (1,1), al igual que una Correlation Mark solo puede pertenecer a una relación entre Bankruptcy e Historic (1,1).

### 4.1.3. Transformación de Datos

En esta sección se persigue describir las transformaciones que afectarán a los datos y que conllevarán la transformación de estos de Raw Data a Refined Data, permitiendo así ahondar un paso más en el proceso ETL previamente descrito.

En los siguientes apartados se realizará una aproximación detallada sobre cada una de las transformaciones realizadas sobre el conjunto de datos originales de los que disponemos, siendo estas divididas por fase de limpieza.

#### Fase 0

Fase inicial de procesamiento de datos en la que se persigue, principalmente, generar los ficheros históricos que ingestaremos en nuestro Data Lake. Además de la generación de estos ficheros se persigue eliminar caracteres raros existentes a lo largo de todos los ficheros, así como algunos valores nulos en los registros que no dispongan de ningún valor. Esta fase inicial de limpieza solo se llevará a cabo sobre los datos procedentes de Yahoo Finance, ya que son los datos que como hemos indicado recogemos para la creación de nuestras tablas históricas en el Data Lake. A continuación se detallan las dos transformaciones realizadas sobre los datos en esta fase de procesamiento.

- **Eliminación de la cabecera:** El CSV que se descarga de Yahoo Finance trae una primera línea que define la cabecera. A nosotros nos sobra ya que la tabla Hive sobre la que se montan estos datos ya tiene definidas las columnas.
- **Cambio de separador:** En el entorno de trabajo en el que estoy tenemos como buena práctica que siempre que se pueda el carácter que define la separación de campos sea el conocido como carácter no imprimible o *SOH* definido por su valor Unicode como

*u0001*. Esto implica esta fase de procesamiento de los datos donde cambiamos las comas por dicho carácter.

### Fase 1

En esta segunda fase de procesamiento de la información se persigue realizar la adaptación entre el *JSON* que nos devuelve por defecto la *API* a la que consultamos en tiempo real y el fichero que necesitamos para ser insertado en su correspondiente tabla en Hive.

Esta fase se realiza íntegra desde *NiFi*, herramienta con la que realizamos la lectura en tiempo real y que describiremos en detalle en la sección 5.5.4. Así a continuación se podrán ver en detalle las diferentes transformaciones realizadas en esta fase ordenadas en función de la fuente de información a la que pertenecen y la entidad en la que se desean almacenar.

En una primera versión de la *API* nos era necesario el uso del *Processor* de *NiFi* llamado *ReplaceText* que vemos en la figura 4.3, ya que este *JSON* nos venía con una jerarquía inicial de más identificada por el texto *quote* y que conlleva también un caracter final de } sobrante para nosotros. Esto ya no nos es necesario.

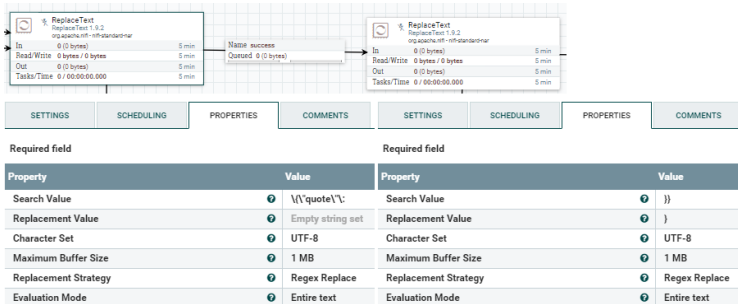


Figura 4.3: *ReplaceText* de *NiFi* en desuso para nosotros.

Sin embargo sí que tenemos que hacer uso del *Processor* de *NiFi* llamado *ConvertJSON-ToAvro* que vemos en la figura 4.4, aquí le pasamos un esquema de tipo *Avro* (es un marco de serialización de datos orientado a filas desarrollado dentro del proyecto Hadoop de Apache. Utiliza *JSON* para definir los tipos de datos y protocolos, y serializa los datos en un formato binario compacto) que necesita para poder hacer la transformación desde el *JSON* que recibimos al fichero a insertar que necesita *NiFi*.



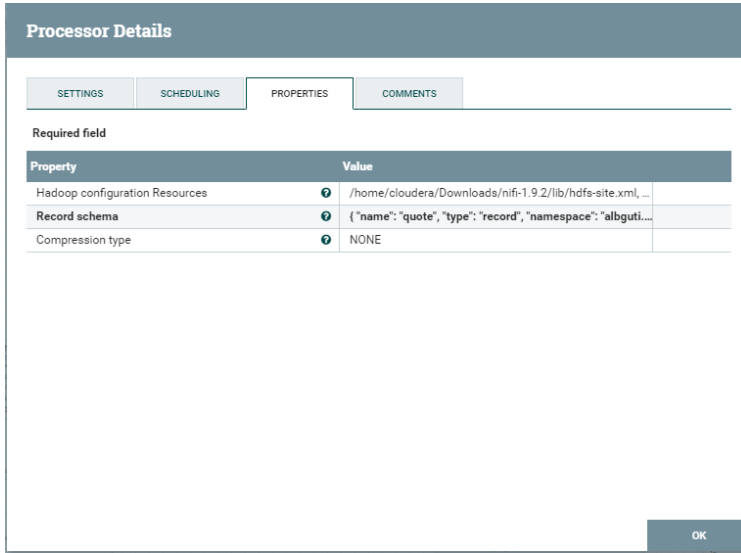


Figura 4.4: Nuestro ConvertJSONToAvro de NiFi.

En la siguiente figura 4.5 vemos una parte del detalle del esquema Avro que le pasamos para la transformación de estos ficheros.

```

1  {
2    "name": "quote",
3    "type": "record",
4    "namespace": "albguti.avro",
5    "fields": [
6      {
7        "name": "symbol",
8        "type": [
9          "string",
10         "null"
11       ]
12     },
13     {
14       "name": "companyName",
15       "type": [
16         "string",
17         "null"
18       ]
19     },
20     {
21       "name": "calculationPrice",
22       "type": [
23         "string",
24         "null"
25       ]
26     },
27     {
28       "name": "open",
29       "type": [
30         "double",
31         "null"
32       ]
33     },
34     {
35       "name": "openTime",

```

Figura 4.5: Detalle del esquema de Avro para los datos recogidos en tiempo real.

### Fase 2

En esta fase se persigue extraer valor sobre los datos ya almacenados, permitiendo así entre otras cosas generar nuevos atributos o eliminar mensajes duplicados o erróneos. A continuación se describe la única transformación que se realiza en esta fase.

Los datos en tiempo real que ingestamos con NiFi son ingestados, por restricciones del *Processor* de NiFi llamado *PutHiveStreaming*, en formato ORC (Apache ORC es un formato de almacenamiento de código abierto orientado a columnas del ecosistema de Apache Hadoop). Esto nos es una limitación ya que debemos hacer una transformación de nuestros datos en ORC a Parquet, lo cual llevamos a cabo a través de unos scripts que también ejecuta NiFi y que se verán en detalle en el capítulo de Implementación. En cuanto a la transformación que nos atañe aquí indicar que lo que hacemos con este script es una transformación con Hive del campo C20 del cuadro 4.2 *latestUpdate* que viene como un timestamp del tipo *1562356800197* a formato fecha con hora convertido a la hora de la máquina en la que trabajamos que es la de la ciudad de Los Ángeles. Esto lo conseguimos con la ejecución de la siguiente sentencia en Hive como hemos dicho:

```
to_utc_timestamp(from_unixtime(latestupdate DIV 1000),'America/Los_Angeles')
AS dateTime
```

## 4.2. Arquitectura lógica

En esta sección se persigue detallar los diferentes componentes lógicos que forman el sistema, así como la relación existente entre los mismos. La arquitectura lógica de este proyecto se corresponderá con todas las herramientas con las que nos vamos a apoyar para llegar desde la ingesta en tiempo real y la ingesta de datos históricos en batch hasta la explotación de los datos en un panel de visualización, pasando por todos los procesos de transformación, modelado, etc. A continuación se podrá ver la arquitectura lógica de este proyecto, descrita en la imagen 4.6.

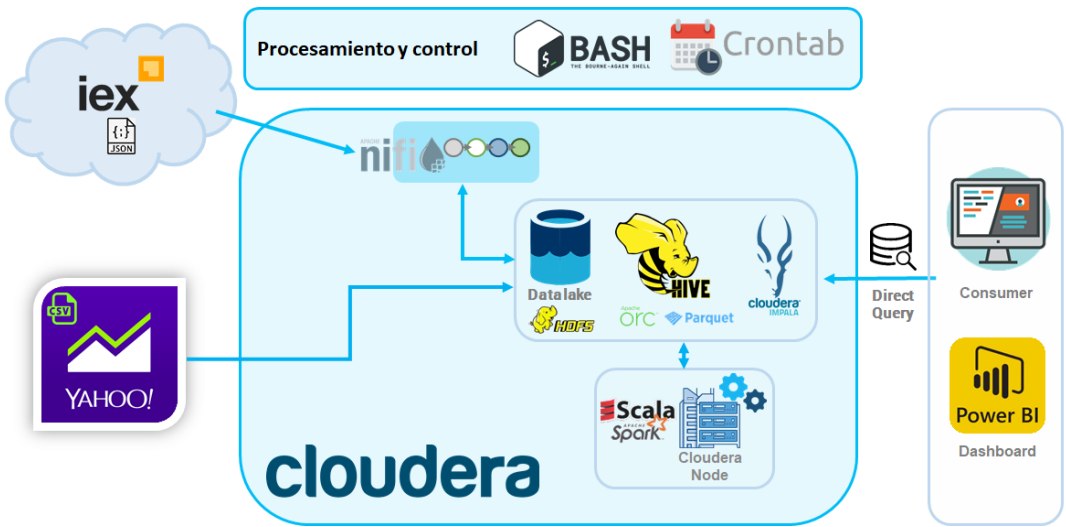


Figura 4.6: Arquitectura lógica.

Se ha realizado como un modelo de referencia de la arquitectura *LAMBDA* basado en la arquitectura de Big Data Open-Source (Cloudera Distribution) debido a la utilización de tecnologías Batch y Real Time. Esta arquitectura está explicada en detalle en la sección 2.1.2 Arquitectura *LAMBDA*.

La elección de todas estas herramientas está basada en el previo y breve análisis de mercado realizado en correspondencia a esa “adaptación de un alumno recién salido del grado en ingeniería informática al entorno Big Data en el mercado laboral”.

### 4.3. Arquitectura Física

En esta sección se persigue describir la arquitectura física utilizada a lo largo del proyecto.

Cabe destacar que dentro de la arquitectura física debería de encontrarse la arquitectura del Data Lake, ampliamente detallada en la subsección 2.1.1. Otro hecho a tener en cuenta son las limitaciones en cuanto a recursos con los que se ha realizado el proyecto, siendo imposible desarrollar una arquitectura física que facilite las mejores capacidades de gestión de concurrencia o de integridad y recuperación de la información.

El proyecto se ha realizado utilizando un único servidor virtualizado de la Universidad de Valladolid, este es capaz de captar en crudo de la información y con un entorno Cloudera utilizado para el procesamiento de la información.

A continuación es posible ver la simple arquitectura física resultante con la que se ha desarrollado este proyecto.

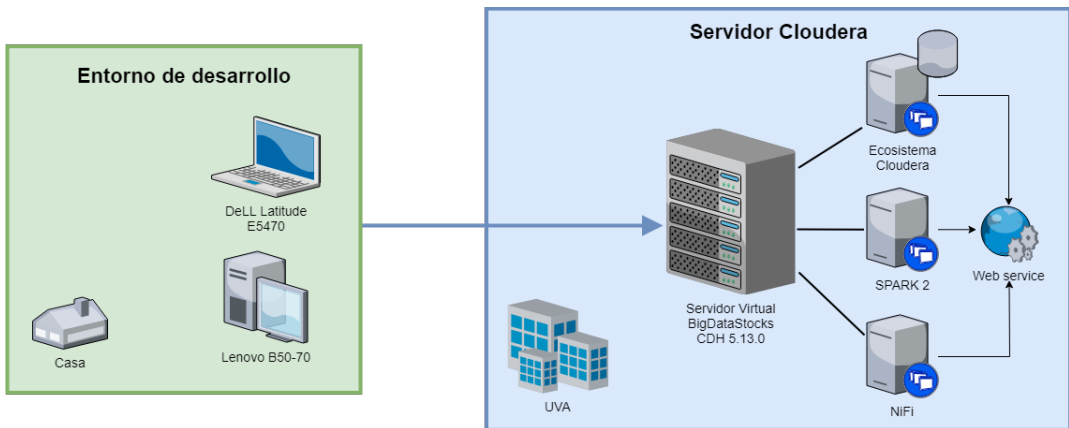


Figura 4.7: Arquitectura física utilizada.

## 4.4. Diseño de interfaces

En esta sección se persigue detallar las diferentes interfaces que compondrán la aplicación desarrollada a modo de dashboard. Por limitaciones de tiempo no se ha llegado a desarrollar todos los puntos que se tenían previstos y se han caído de la planificación, prefiriendo dar calidad a las tres pantallas que se van a describir a continuación:

### 4.4.1. Interfaz para los datos en Real Time

- **Nombre:** Real Time
- **Descripción:** Pestaña de la aplicación de Power BI en la que podremos visualizar la cotización en bolsa de las empresas en tiempo real. Tendrá un filtro por empresa y mostrará un gráfico lineal que se irá rellenando a medida que vaya avanzando el día y una tarjeta con el valor actual de cotización.
- **Activación:** Abrir la aplicación distribuida (archivo *.pbix*), seleccionar en el menú inferior la pestaña correspondiente que será la primera de ellas y filtrar por empresa.

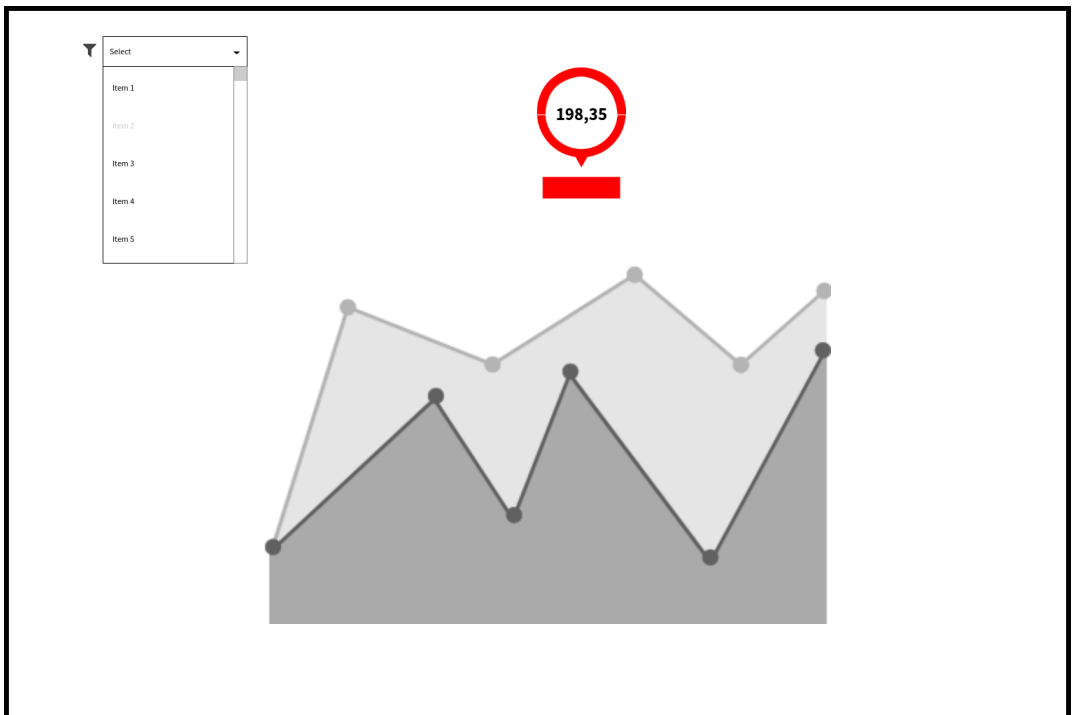


Figura 4.8: Boceto de la ventana de visualización para los datos en tiempo real.

### 4.4.2. Interfaz para los datos Históricos

- **Nombre:** Historic
- **Descripción:** Pestaña de la aplicación de Power BI en la que podremos visualizar la cotización histórica en bolsa de las empresas disponibles en el Data Lake. Tendrá un filtro por empresa, un filtro por intervalo de fechas y mostrará un gráfico linear donde veremos esos valores de cierre de mercado de cada día que la empresa ha estado cotizando en bolsa.
- **Activación:** Abrir la aplicación distribuida (archivo *.pbix*), seleccionar en el menú inferior la pestaña correspondiente que será la segunda de ellas y filtrar por empresa.

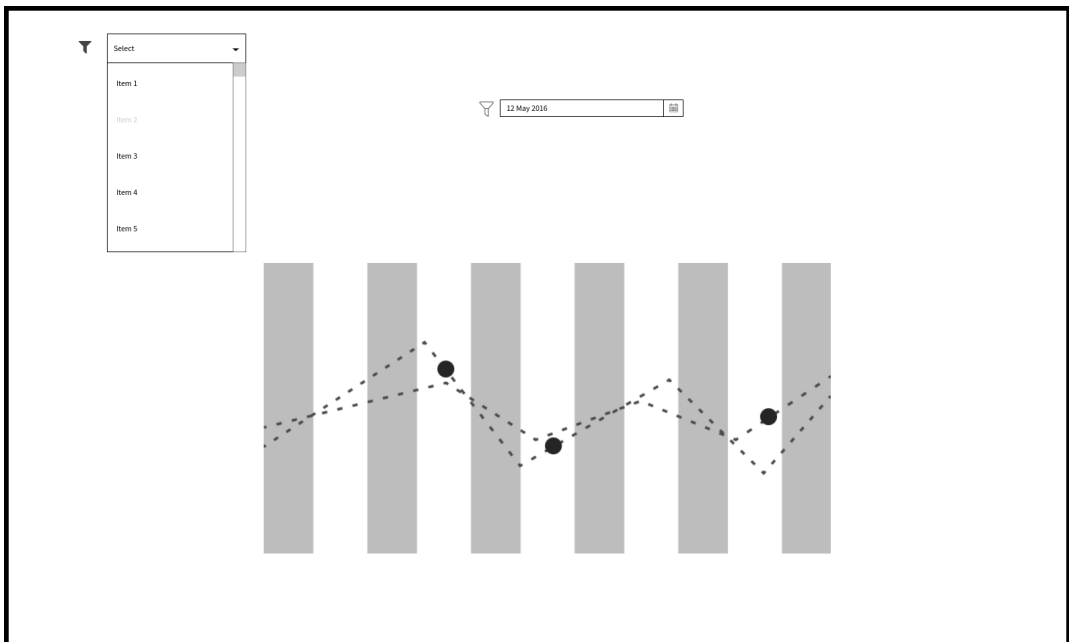


Figura 4.9: Boceto de la ventana de visualización para los datos históricos.

### 4.4.3. Interfaz para la visualización de tipo Trading Chart

Estos gráficos se tratan de un tipo de gráfico muy utilizado en el análisis técnico bursátil. Se representan como gráficos de velas y refleja:

- El precio de apertura de un valor.
- El precio de cierre de un valor.

- El máximo y el mínimo que ha encontrado ese valor.

El funcionamiento es básico, cada vela puede ser roja (negra) o verde (blanca), será roja para cuando el precio de una acción, futuro, índice o divisa haya caído, en cambio será verde si el precio ha subido respecto al inicio de esa misma vela. El rectángulo es el cuerpo de la vela, nos indica la variación desde la apertura al cierre y una línea que lo cruza (sombra) nos indica el máximo y el mínimo.

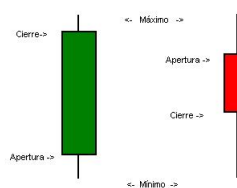


Figura 4.10: Funcionamiento del gráfico de tipo Trading Chart.

Por norma general una vela nos informa de los parámetros anteriormente mencionados en una sesión bursátil (cada vela es un día de bolsa) sin embargo se pueden utilizar velas de 30 minutos, 10 minutos, semanales, mensuales... Cualquier período, todas ellas tienen su utilidad en función de lo que analicemos y lo minuciosos que queramos ser.

Hay velas que, por sí solas, son indicadores adelantados de comportamientos de la acción, pero lo más común es utilizar figuras de tres velas. Esto es, 3 velas que por sus características nos informan del comportamiento del valor.

- **Nombre:** Trading Chart
- **Descripción:** Pestaña de la aplicación de Power BI en la que podremos visualizar el gráfico de velas asociado a una empresa que cotiza en bolsa y que tenemos en nuestro Data Lake. Este solo mostrará los datos de los últimos 30 días en los que dicha empresa ha cotizado (el cambio de este valor sería muy fácil de implementar a futuro). Tendrá un filtro de selección de empresa y mostrará dicho gráfico de velas donde se puede analizar la cotización de la empresa seleccionada.
- **Activación:** Abrir la aplicación distribuida (archivo *.pbix*), seleccionar en el menú inferior la pestaña correspondiente que será la tercera de ellas y filtrar por empresa.



Figura 4.11: Boceto de la ventana de visualización para el gráfico de velas Trading Chart.



## Capítulo 5

# Implementación

En este capítulo se explicará cómo se ha implementado el sistema desarrollado a lo largo de este proyecto. Para ello se realizará una breve explicación sobre el desarrollo de la aplicación y la arquitectura Big Data asociada a esta, así como de las diferentes herramientas y tecnologías que han sido utilizadas para el desarrollo del mismo.

### 5.1. Descripción del sistema

La herramienta desarrollada gira en torno a la captura de datos de la bolsa en tiempo real a través de NiFi y la API de IEX Trading, explicada con anterioridad. En un inicio no se contemplaba el desarrollo de ninguna aplicación de visualización, dashboard, sin embargo con el fin de tratar de mostrar el trabajo realizado y comprender mejor los datos captados se ha decidido implementar un simple dashboard que permita su visualización aunque este no fuera uno de los objetivos iniciales.

El hecho de contar con dicho volumen de datos (reducido por limitaciones ya explicadas), junto con el interés principal del proyecto de diseñar y desarrollar, aunque fuera en parte, un Data Lake para gestionar el mercado de valores, ha llevado al uso de novedosas herramientas y tecnologías Big Data, que al final es el objetivo principal de este proyecto. Por tanto la implementación de este proyecto cuenta con tres partes totalmente diferenciadas, que se unen para componer el sistema final desarrollado.

En el inicio del proyecto, con el fin de poder captar la información ofrecida por IEX Trading, se desarrolló una pequeña aplicación Scala encargada de realizar conexiones cada 60 segundos a la API de IEX Trading, a través de una url, y descargar los datos disponibles en la misma en el momento de la conexión. Pero esto cambió por completo al descubrimiento de la tan demandada herramienta de automatización de flujo de datos *Apache NiFi*. Los datos obtenidos a través de esta aplicación se almacenan de manera constante mientras el mercado de valores está abierto en un servidor de la Universidad de Valladolid.

Una vez se dispone de los datos en crudo, se ha desarrollado una arquitectura Big Data en la que se trata de almacenar y procesar los datos asociados al mercado de valores, así como la información asociada a las empresas, previamente explicada. Esta arquitectura Big Data es conocida como Data Lake y montada en un entorno Cloudera, utilizando Hive y Spark con el fin de procesar la información disponible, además de HDFS, para almacenar los datos captados en tiempo real.

Paralelamente al desarrollo del Data Lake, anteriormente mencionado, se ha desarrollado una aplicación, diseñada a modo de dashboard, con el objetivo de tratar de dotar de mayor valor a los datos almacenados. Esta aplicación se desarrolla de manera íntegra utilizando Power BI, herramienta de visualización muy utilizada y demandada en la actualidad.

Esta herramienta de visualización se conecta con el Data Lake, anteriormente mencionado, utilizando su conector con *Cloudera Impala* (motor de consultas SQL open source de Cloudera para el procesamiento masivo en paralelo de los datos almacenados en un clúster de computadoras corriendo Apache Hadoop).

El desarrollo de este sistema ha supuesto un gran reto, ya que, además de la utilización de gran cantidad de tecnologías y herramientas no utilizadas con anterioridad, se ha tenido que modificar el entorno Cloudera sobre el que se desarrolla, con el fin de incluir más herramientas no incluidas por defecto o actualizaciones de las que sí incluye. Además del reto tecnológico del Big Data en Real Time. A lo largo de este capítulo se explicará en mayor detalle el desarrollo de estas herramientas.

## 5.2. Descripción del entorno

Como se ha comentado en la sección anterior el proyecto se ha desarrollado utilizando un entorno Cloudera, más concretamente utilizando CDH 5.13 standalone. Este entorno utiliza como sistema operativo base un sistema CentOS 6.9, bifurcación de la distribución linux Red Hat Enterprise Linux RHEL. En este entorno nos encontramos con gran cantidad de herramientas Big Data, como Hive, Hue o Impala, ya instaladas, así como con la opción de acceder a Cloudera Manager, herramienta que permite gestionar los servicios ejecutados en el clúster con el que se trabaja. Sin embargo no se dispone de todos los elementos necesarios ya instalados, por ello para el desarrollo de todo este proyecto es necesario instalar y actualizar gran cantidad de paquetes y herramientas, con el fin de que este pueda ser desarrollado, además de que esto es una realidad en el día a día de los proyectos de Big Data en el mercado laboral.

La instalación de estas herramientas ha dado lugar a diferentes problemas derivados del sistema operativo utilizado, su versión, al estar anticuada, etc. La modificación de este entorno, mediante la instalación de *parcels*, paquetes y librerías, se divide en 4 fases, descritas en las siguientes sub-secciones:

### 5.2.1. Activación del Cloudera Manager

Para tener acceso a *Parcels* de Cloudera, como veremos en el tercer apartado donde explicamos la instalación de Spark2, necesitamos habilitar Cloudera Manager. Para poder habilitar Cloudera Manager necesitamos habilitar 2 cores en la MV y nosotros disponemos de 4.

Con la MV iniciada, ejecutamos el script del icono del escritorio de la MV con el nombre “Launch Cloudera Express” y esperamos a que termine con un mensaje del tipo:

```
Success! You can now log into Cloudera Manager from the
QuickStart VM's browser:
http://quickstart.cloudera:7180
```

```
Username: cloudera
Password: cloudera
```

```
Press [Enter] to exit...
```

Podemos presionar *Enter* para cerrar la terminal, y abrimos una nueva para comenzar con el upgrade de Java de los siguientes apartados.

Hemos pedido a los técnicos la redirección de este puerto al exterior para poder acceder a nuestro Cloudera Manager desde la URL `http://virtual.lab.inf.uva.es:20013`

Debemos indicar que tenemos acceso al HUE de Cloudera: Hue es un editor de consultas interactivo basado en web que permite interactuar con los almacenes de datos. Por ejemplo, la

## 5.2. DESCRIPCIÓN DEL ENTORNO

siguiente imagen muestra una representación gráfica de los resultados de la consulta Impala SQL que puede generar con Hue:

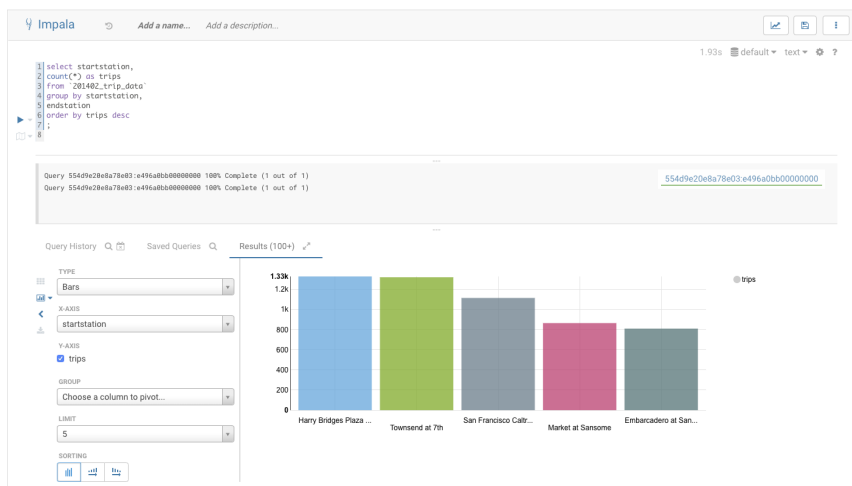


Figura 5.1: Ejemplo de consulta en Impala desde HUE.

También se puede usar Hue para explorar los datos a través de la navegación guiada en el panel izquierdo de la página. Podremos navegar por las bases de datos, profundizar en tablas específicas, ver directorios del HDFS, descubrir índices y tablas HBase o Kudu, encontrar documentos, consulte sus datos con queries en Impala o en Hive creando un panel personalizado o programar trabajos repetitivos.

Esto lo hacemos desde el exterior por la redirección al exterior de su puerto por lo que podemos acceder a este desde la URL <http://virtual.lab.inf.uva.es:20012>

### 5.2.2. Instalación de Java 1.8

Lo primero es conocer nuestra versión de *CentOS* que en nuestro caso lo sabemos ejecutando en un terminal el siguiente comando:

```
$ cat /etc/centos-release
CentOS release 6.7 (Final)
```

Descargamos el archivo *.tar.gz* de la versión de 64 bits de Oracle JDK 1.8 de la página de descargas de Oracle [16].

Nos conectamos por SSH a la máquina con el usuario *root* y ejecutamos los siguientes comandos:

```
$ wget --continue --no-check-certificate -O
/home/cloudera/Downloads/jdk-8u191-linux-x64.tar.gz
--header "Cookie: oraclelicense=a"
https://download.oracle.com/otn-pub/java/jdk/8u201-b09
/42970487e3af4f5aa5bca3f542482c60/jdk-8u201-linux-x64.tar.gz
```

Extraemos el .tar.gz descargado con el siguiente comando:

```
$ tar xvfz /home/cloudera/Downloads/jdk-8u191-linux-x64.tar.gz -C /usr/java/
```

Ahora configuraremos la ubicación del JDK en los hosts del clúster. Para ello abrimos la Consola de Administración del Cloudera Manager. En la barra de navegación principal, hacemos click en la pestaña Hosts. (Si está configurando la ubicación JDK solo en un host específico, haga clic en el enlace de ese host).

Hacemos click en la pestaña Configuration de la barra superior. En el menú de la izquierda seleccionamos la categoría Advanced. Establecemos la propiedad “Java Home Directory” a nuestra ubicación personalizada. En nuestro caso escribiremos “/usr/java/jdk1.8.0\_191”. Y hacemos click en Save Changes.

Ahora abrimos el fichero “/etc/default/cloudera-scm-server” con un editor de texto. Editamos la línea que empieza con “export JAVA\_HOME” (si no existe tal línea la añadimos) y cambiar la ruta a la nueva ruta del JDK, en nuestro caso quedaría algo así:

```
export JAVA_HOME="/usr/java/jdk1.8.0_191"
```

Guardamos el archivo y reiniciamos el Cloudera Manager Server con el siguiente comando:

```
$ sudo service cloudera-scm-server restart
```

También reiniciamos el Cloudera Management Service, para ello en la página principal del Cloudera Manager, en la comuna de la izquierda el último elemento es el Cloudera Management Service (figura 5.2). Click en la flecha de su derecha y click en Restart.

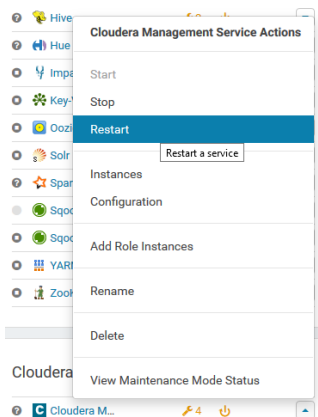


Figura 5.2: Reinicio de Cloudera Management Service.

Renombraremos los archivos de la instalación Java anterior. Si no eliminamos estos archivos, Cloudera Manager y otros componentes pueden continuar utilizando la versión anterior del JDK. Para ello ejecutamos el siguiente comando:

```
$ mv jdk1.7.0_67-cloudera/ OLD_jdk1.7.0_67-cloudera_OLD
```

Editamos el archivo “.bash\_profile” con:

```
$ vim ~/.bash_profile
```

Quedando de la siguiente manera para incluir el JAVA\_HOME y este mismo al path:

```
# User specific environment and startup programs
```

```
JAVA_HOME=/usr/java/jdk1.8.0_191  
export JAVA_HOME
```

```
PATH=$PATH:$HOME/bin:$JAVA_HOME  
export PATH
```

También debemos editar con el siguiente comando Para cambiar el JAVA\_HOME al nuestro:

```
$ sudo vim /etc/profile
```

Ejecutamos el siguiente comando:

```
$ sudo update-alternatives --install  
/usr/bin/java java /usr/java/jdk1.8.0_191/bin/java
```

Reiniciamos la máquina. Tras este reinicio volveremos a entrar en el Cloudera Manager (paciencia, tarda en estar disponible) y hacemos click en los servicios sobre el icono de “Restale configuration” y seguimos los pasos por defecto pero seleccionando el check de la opción “Re-deploy client configuration” para que todos los servicios actualicen su JDK.

### 5.2.3. Instalación de Spark 2.

Nuestro CDH incluye Spark, pero en su versión 1.6 lanzada en enero de 2016, hace ya más de 3 años. Pretendemos hacer uso de las últimas tecnologías y en nuestro intento por ir a lo más avanzado necesitaríamos tener Spark 2.

Tras ver las dependencias y necesidades de las nuevas versiones de Spark2 instalaremos la versión 2.3.0 de Marzo de 2018 aunque esta no sea la más nueva (ya estaría disponible, a fecha de 9 de julio de 2019 en que este párrafo está siendo redactado, la versión 2.4.3 del 2 de Noviembre de 2018).

Nuestra máquina es compatible con esa versión 2.3.0 de Spark, incluso puede estar co-existiendo en la máquina con la versión 1.6.0 que nuestro CDH ya incluía. Pero debemos haber actualizado el JDK de nuestra máquina (por defecto está en la versión 1.7) a la versión 1.8 (Cloudera recomienda y da soporte en concreto de la versión 1.8\_162) como ya hemos explicado en el punto anterior.

Una vez ya tenemos Java 1.8 en nuestra máquina, nos ayudaremos de los llamados *Parcels* de Cloudera: Los parcels son un formato de distribución binario alternativo, admitido por primera vez en Cloudera Manager 4.5, desarrollado originalmente para proporcionar una mejor gestión del ciclo de vida para CDH. Más información puede encontrarse en el punto [15] de la bibliografía.

Este procedimiento ha requerido de mucha investigación, de mucho tiempo y de muchas pruebas, pidiendo a los técnicos de la escuela que me realizaran imágenes de backup de la máquina y a cada fallo que me las restaurasen, ya que los fallos en despliegues erróneos en estas máquinas acaban ensuciando mucho el funcionamiento de otros componentes.

Como hemos dicho vamos a utilizar la opción de instalación por Parcels. Para ello accedemos desde el Cloudera Manager a la gestión de Parcels haciendo click en el icono con forma de paquete de regalo de la parte superior.

Primero actualizamos el CDH. Para ello seleccionamos de Location la opción “Available Remotely” y en Filters seleccionamos el Parcel CDH 5 y hacemos click en Download a su derecha. Una vez descargado el Parcel “CDH 5.13.0-1.cdh5.13.0.p0.29” cambiamos el Filter a “Cloudera QuickStart” y seleccionamos nuevamente el Parcel CDH 5, ahora podremos hacer click en *Distribute* a su derecha. Una vez desplegado hacemos click en *Activate*.

Ahora aquí pulsaremos en la pestaña Configuration del menú superior. En el apartado

## 5.2. DESCRIPCIÓN DEL ENTORNO

---

“Remote Parcel Repository URLs” añadiremos uno nuevo con el siguiente valor para nuestro caso: “<http://archive.cloudera.com/spark2/parcels/latest/>” y click en *Save Changes*.

Ahora veremos que en el menú de la izquierda nos sale SPARK2 para ser elegido. Seleccionamos de Location la opción “Available Remotely” y en Filters seleccionamos SPARK2 (figura 5.3).

Available Remotely

| Parcel Name | Version                               | Status             | Actions                                 |
|-------------|---------------------------------------|--------------------|---|
| SPARK2      | 2.3.0.cloudera3-1.cdh5.13.3.p0.458809 | Available Remotely | <input type="button" value="Download"/> |

Figura 5.3: Parcel de SPARK2 en Cloudera Manager.

Haremos click sobre Download y, una vez descargado, cambiamos al filtro “Cloudera QuickStart” de Location y nos aparecerá la opción Distribute sobre SPARK2 que haremos nuevamente click. Acabado este despliegue nos aparecerá una nueva opción, Activate, nuevamente hacemos click.

Una vez activado en la página principal del Cloudera Manager tenemos que volver a hacer Restale de la configuración. Para ello hacemos click en los servicios sobre el icono de “Restale configuration” y seguimos los pasos por defecto pero seleccionando el check de la opción “Re-deploy client configuration” para que todos los servicios actualicen su CDH.

Ahora necesitamos añadir el servicio de Spark2 al cluster. Para ello en la página principal del Cloudera Manager, en la columna de la izquierda donde nos pone el nombre de nuestro Host Cloudera QuickStart (CDH 5.13.0 Parcels) a su derecha pulsamos el botón desplegable y seleccionamos *Add Service*. En el primer paso seleccionamos la opción Spark. En el segundo paso seleccionamos las dependencias y seleccionamos la opción con más dependencias (HBase, HDFS, YARN (MR2 Included), ZooKeeper). En el siguiente paso seleccionamos como Host nuestro quickstart.cloudera y de igual manera para el Gateway. En el tercer paso no tocamos nada y pulsamos continuar. En el cuarto paso arranca los servicios, esperamos a que termine y ya lo tendremos listo.

Por ultimo debemos actualizar los ficheros *profile* y *.bash\_profile* añadiendo a ambos la siguiente línea:

```
export SPARK_DIST_CLASSPATH=$(hadoop classpath)
```

Reiniciamos la máquina. Una vez volvamos a tener todos los servicios necesarios levantados y disponibles, en un terminal lanzamos el spark2-shell y veremos que efectivamente usa la versión 2.3.0 de Spark como vemos en la siguiente ilustración 5.4.





## 5.2. DESCRIPCIÓN DEL ENTORNO

---

Para parar el servicio únicamente se debe ejecutar el siguiente comando:

```
sh -x ./bin/nifi.sh stop
```

En caso de tener arrancado el servicio de Apache NiFi se accede a la plataforma a través de la siguiente url: <http://localhost:8080/nifi/>. En nuestro caso hemos pedido a la escuela la redirección de este puerto al exterior por lo que entrando en <http://virtual.lab.inf.uva.es:20014/nifi/> veremos nuestro NiFi funcionando.

## 5.3. Herramientas utilizadas

En esta sección se describirán las diferentes herramientas que se han utilizado en el desarrollo del proyecto.

### 5.3.1. Cloudera CDH 5.13.0

Plataforma de código abierto de Cloudera sobre la que se despliegan soluciones Hadoop, al incorporar su núcleo y diversos proyectos de la fundación Apache, como Hive, HBase, Mahout o Pig. Una de las características más destacadas de Cloudera es que cuenta con una interfaz gráfica propietaria, llamada Cloudera Manager, para la administración y gestión de los nodos y servicios del clúster. La versión de Cloudera utilizada en este proyecto cuenta con un clúster de un único nodo y está desplegada sobre un sistema operativo CentOS 6.9.

En definitiva Cloudera puede definirse como una solución ya desarrollada de Hadoop en la que se pueden desarrollar proyectos que necesiten el uso de tecnologías Big Data.

### 5.3.2. IntelliJ

Es un entorno de desarrollo integrado (IDE) para el desarrollo de programas informáticos. Es desarrollado por JetBrains y está disponible en dos ediciones: edición para la comunidad (gratuita) y edición comercial. IntelliJ IDEA no está basada en Eclipse como MyEclipse u Oracle Enterprise Pack para Eclipse.

La primera versión de IntelliJ fue liberada en enero 2001, y en aquel momento fue uno de los primeros IDE Java disponibles con navegación avanzada de código y capacidades de refactorización de código integrado. En un informe de *Infoworld* en 2010, IntelliJ recibió la puntuación más alta entre las cuatro mejores herramientas de programación de Java: *Eclipse*, *IntelliJ IDEA*, *NetBeans* y *Oracle JDeveloper*.

En diciembre 2014, Google anunció la versión 1.0 de Android Studio, un IDE de código abierto para aplicaciones de Android basado en el código abierto de la edición comunitaria de IntelliJ. Otros entornos de desarrollo se basaron en IntelliJ como PhpStorm, PyCharm, RubyMine o WebStorm.

| Languages               | IntelliJ IDEA<br>Community Edition | IntelliJ IDEA<br>Ultimate Edition |
|-------------------------|------------------------------------|-----------------------------------|
| ActionScript/MXML       | No                                 | Sí                                |
| Clojure (vía plugin)    | Sí                                 | Sí                                |
| CoffeeScript            | No                                 | Sí                                |
| Dart (vía plugin)       | Sí                                 | Sí                                |
| Erlang (vía plugin)     | Sí                                 | Sí                                |
| Go (vía plugin)         | Sí                                 | Sí                                |
| Groovy                  | Sí                                 | Sí                                |
| Haskell (vía plugin)    | Sí                                 | Sí                                |
| Haxe (vía plugin)       | Sí                                 | Sí                                |
| HTML/XHTML/CSS          | No                                 | Sí                                |
| Java                    | Sí                                 | Sí                                |
| JavaScript              | No                                 | Sí                                |
| Kotlin                  | Sí                                 | Sí                                |
| Lua (vía plugin)        | Sí                                 | Sí                                |
| Perl (vía plugin)       | Sí                                 | Sí                                |
| PHP (vía plugin)        | No                                 | Sí                                |
| Python (vía plugin)     | Sí                                 | Sí                                |
| Ruby/JRuby              | No                                 | Sí                                |
| Scala (vía plugin)      | Sí                                 | Sí                                |
| SQL                     | No                                 | Sí                                |
| TypeScript (vía plugin) | No                                 | Sí                                |
| XML/XSL                 | Sí                                 | Sí                                |

Cuadro 5.1: Lenguajes soportados por IntelliJ

### 5.3.3. Power BI

Es una solución de análisis empresarial que permite visualizar los datos y compartir información con toda la organización, o insertarla en su aplicación o sitio web.

Es una colección de servicios de software, aplicaciones y conectores que funcionan conjuntamente para convertir orígenes de datos sin relación entre sí en información coherente,

interactiva y atractiva visualmente. Sus datos pueden ser una hoja de cálculo de Excel o una colección de almacenes de datos híbridos locales y basados en la nube. Power BI permite conectarse a los orígenes de datos, visualizar y descubrir lo que es importante y compartirlo con quien desee fácilmente.



Figura 5.5: Ejemplo de orígenes de Power BI.

Permite la colaboración con usuarios de todos los roles de una organización. Publicar contenido en tiempo real al que los usuarios pueden acceder tanto si están en la oficina como fuera. Administrar de forma centralizada la inteligencia empresarial de una organización.

Power BI puede ser sencilla y rápida, capaz de crear información rápida de una hoja de cálculo de Excel o una base de datos local. Pero Power BI también es estable y empresarial, listo para una amplia modelado y análisis en tiempo real, así como desarrollo personalizado. Puede ser el personal informe y la herramienta de visualización y actuar también como el motor de análisis y de decisión para proyectos de grupo, divisiones o empresas enteras.

Power BI consta de:

- Una aplicación de escritorio de Windows llamado Power BI Desktop
- Un SaaS en línea (Software como servicio) servicio denominado el servicio Power BI
- Aplicaciones móviles para dispositivos Android, iOS y Windows

### 5.3. HERRAMIENTAS UTILIZADAS

---

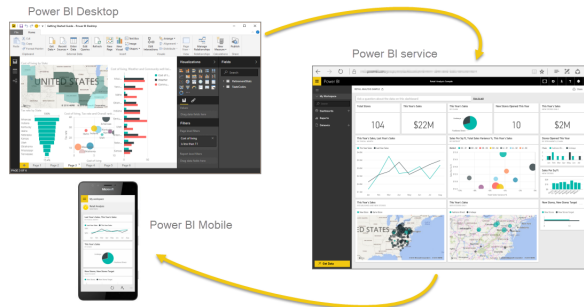


Figura 5.6: Ecosistema de Power BI.

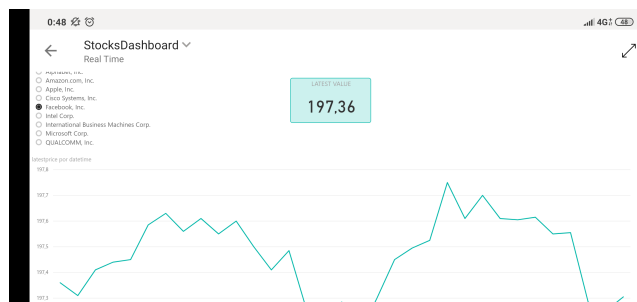


Figura 5.7: Ejemplo de mi aplicación vista desde la aplicación Android de Power BI.

Estos tres elementos están diseñados para permitir a los usuarios crear, compartir y consumir información empresarial de la manera en que resulte más apropiada a cada rol.

#### 5.3.4. Overleaf

Overleaf es un servicio de  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  colaborativo en línea con el cual se pueden realizar trabajos académicos, artículos técnicos, realizar posters y diapositivas utilizando  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  en la nube.

Overleaf provee además servicios Premium para revisión de artículos por personas de lengua inglesa para verificar el lenguaje y que no sea rechazado por tecnicismos. Peer review de artículos antes de ser enviados a revistas, servicios de edición para libros y publicación directa en varios journals Open Access.

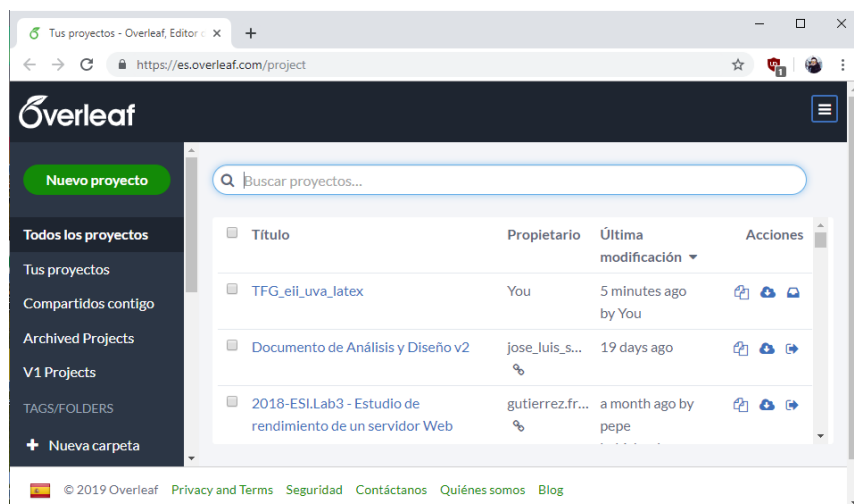


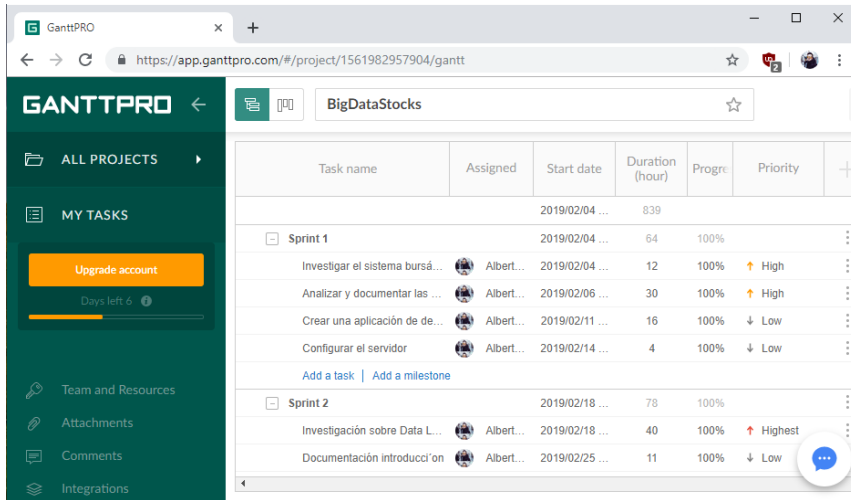
Figura 5.8: Captura de pantalla de mi panel de Overleaf.

Lo único que necesitamos para empezar a trabajar con Overleaf es un navegador web y conexión a internet. Ofrece algunos tutoriales para empezar a trabajar con él. Hay cientos de diferentes plantillas (para libros, tesis, informes, artículos, presentaciones, etc.) que por la naturaleza de  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  pueden ser modificados y adaptados a las necesidades. En este caso se ha utilizado la plantilla de la Universidad de Valladolid que hay disponible en su web y que está adaptada para cumplir con los requisitos de forma que este documento requiere.

### 5.3.5. GanttPRO

Es una herramienta online para la gestión de proyectos que sigan metodologías ágiles. Permite planificar, crear y gestionar tareas de TI, construcción, desarrollo web, eventos, diseño y muchos otros proyectos. Útil para programar tareas y asignarlas, establecer sus duraciones y dependencias entre ellos. Compartir el proyecto y exportarlo a otros formatos populares, dejar comentarios en las tareas y adjuntar archivos a ello, incluso permite notificaciones en tiempo real.

### 5.3. HERRAMIENTAS UTILIZADAS



The screenshot displays the GanttPRO web application interface. The browser address bar shows the URL: <https://app.ganttpro.com/#/project/1561982957904/gantt>. The page title is 'BigDataStocks'. The interface features a dark green sidebar on the left with navigation options: 'ALL PROJECTS', 'MY TASKS', 'Upgrade account' (with 'Days left: 6'), 'Team and Resources', 'Attachments', 'Comments', and 'Integrations'. The main content area shows a Gantt chart and a table of tasks. The table has columns for 'Task name', 'Assigned', 'Start date', 'Duration (hour)', 'Progre', and 'Priority'. The tasks are organized into two sprints: 'Sprint 1' and 'Sprint 2'. Each task is assigned to 'Albert...' and has a progress bar and priority indicator.

| Task name                      | Assigned  | Start date     | Duration (hour) | Progre | Priority |
|--------------------------------|-----------|----------------|-----------------|--------|----------|
| 2019/02/04 ...                 |           |                | 839             |        |          |
| Sprint 1                       |           |                | 2019/02/04 ...  | 64     | 100%     |
| Investigar el sistema bursá... | Albert... | 2019/02/04 ... | 12              | 100%   | High     |
| Analizar y documentar las ...  | Albert... | 2019/02/06 ... | 30              | 100%   | High     |
| Crear una aplicación de de...  | Albert... | 2019/02/11 ... | 16              | 100%   | Low      |
| Configurar el servidor         | Albert... | 2019/02/14 ... | 4               | 100%   | Low      |
| Add a task   Add a milestone   |           |                |                 |        |          |
| Sprint 2                       |           |                | 2019/02/18 ...  | 78     | 100%     |
| Investigación sobre Data L...  | Albert... | 2019/02/18 ... | 40              | 100%   | Highest  |
| Documentación introducci'on    | Albert... | 2019/02/25 ... | 11              | 100%   | Low      |

Figura 5.9: Captura de pantalla de mi panel de GanttPRO.



## 5.4. Tecnologías utilizadas

En esta sección se persigue describir las diferentes tecnologías y lenguajes de programación utilizados en el desarrollo de este proyecto.

- Spark 2.** Apache Spark es un framework de computación en clúster open-source. Fue desarrollada originariamente en la Universidad de California, en el AMPLab de Berkeley. El código base del proyecto Spark fue donado más tarde a la Apache Software Foundation que se encarga de su mantenimiento desde entonces. Spark proporciona una interfaz para la programación de clusters completos con Paralelismo de Datos implícito y tolerancia a fallos.

Apache Spark se puede considerar un sistema de computación en clúster de propósito general y orientado a la velocidad. Proporciona APIs en Java, Scala, Python y R. También proporciona un motor optimizado que soporta la ejecución de grafos en general. También soporta un conjunto extenso y rico de herramientas de alto nivel entre las que se incluyen Spark SQL (para el procesamiento de datos estructurados basada en SQL), MLlib para implementar machine learning, GraphX para el procesamiento de grafos y Spark Streaming.

Spark fue desarrollado en sus inicios por Matei Zaharia en el AMPLab de la UC Berkeley en 2009. Fue liberado como código abierto en 2010 bajo licencia BSD.

En 2013, el proyecto fue donado a la Apache Software Foundation. En noviembre de 2014, la empresa de sus fundadoras, M. Zaharia Databricks y Carmen Pérez Universidad Complutense obtuvo un nuevo récord mundial en la ordenación a gran escala usando Spark. Hacia 2015, Spark tenía más de 1000 contribuidores convirtiéndose en uno de los proyectos más activos de la Apache Software Foundation y en uno de los proyectos de big data open source más activos.

| Versión | Fecha de la Distribución Original | Última Versión | Fecha de la Distribución |
|---------|-----------------------------------|----------------|--------------------------|
| 0.5     | 2012-06-12                        | 0.5.1          | 2012-10-07               |
| 0.6     | 2012-10-14                        | 0.6.2          | 2013-02-07 <sup>6</sup>  |
| 0.7     | 2013-02-27                        | 0.7.3          | 2013-07-16               |
| 0.8     | 2013-09-25                        | 0.8.1          | 2013-12-19               |
| 0.9     | 2014-02-02                        | 0.9.2          | 2014-07-23               |
| 1.0     | 2014-05-30                        | 1.0.2          | 2014-08-05               |
| 1.1     | 2014-09-11                        | 1.1.1          | 2014-11-26               |
| 1.2     | 2014-12-18                        | 1.2.2          | 2015-04-17               |
| 1.3     | 2015-03-13                        | 1.3.1          | 2015-04-17               |
| 1.4     | 2015-06-11                        | 1.4.1          | 2015-07-15               |
| 1.5     | 2015-09-09                        | 1.5.2          | 2015-11-09               |
| 1.6     | 2016-01-04                        | 1.6.3          | 2016-11-07               |
| 2.0     | 2016-07-26                        | 2.0.2          | 2016-11-14               |
| 2.1     | 2016-12-28                        | 2.1.1          | 2017-05-02               |
| 2.2     | 2017-07-11                        | 2.2.1          | 2017-12-01               |

Legenda: ■ Versión antigua ■ Versión antigua, con servicio técnico ■ Última versión ■ Última versión prevista

Figura 5.10: Versiones de Spark a lo largo del tiempo.

Apache Spark tiene la base de su arquitectura en el llamado RDD o Resilient Distributed DataSet que es un multiset de solo lectura de ítems de datos distribuidos a lo largo de un clúster de máquinas que se mantiene en un entorno tolerante a fallos. Spark y sus RDDs fueron desarrollados en 2012 en respuesta a las limitaciones del paradigma de computación en clúster *MapReduce* que fuerza a la utilización de una estructura lineal *dataflow* en particular en los programas distribuidos: Los programas basados en *MapReduce* leen los datos de entrada desde disco, que mapea una función a lo largo de los datos, reduce los resultados del mapa y almacena los resultados de la reducción en disco. Los RDDs de Spark funcionan como un *working set* para los programas distribuidos que ofrecen una forma deliberadamente restringida de la memoria compartida distribuida en el cluster.

Para la gestión del clúster, Spark soporta las opciones siguientes:

- Spark Standalone (Cluster Spark Nativo)
- Hadoop YARN
- Apache Mesos

Para el almacenamiento distribuido, Spark presenta interfaces hacia una gran variedad de plataformas:

- Hadoop Distributed File System (HDFS)
- MapR File System (MapR-FS)
- Cassandra
- OpenStack Swift
- Amazon S3
- Kudu
- Incluso soporta una solución personalizada.

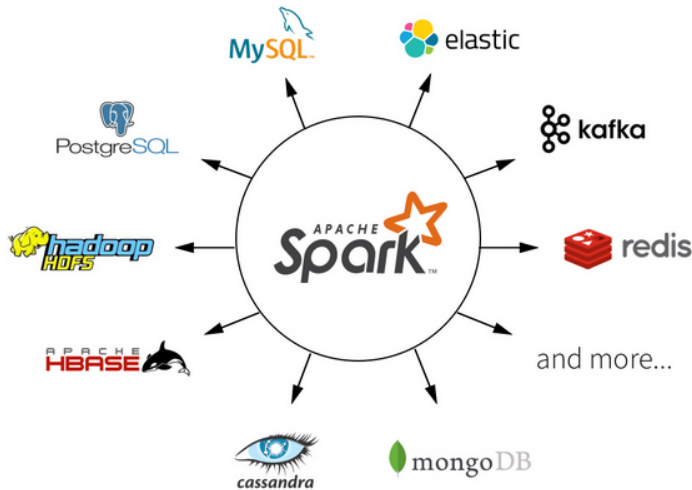


Figura 5.11: Posibles orígenes de Apache Spark.

Nosotros vamos a hacer uso la versión 2 de Spark y de su API en Scala que como ya se ha comentado es la más demandada y en cierto modo la más óptima por su naturaleza como lenguaje funcional. Gestionado por *Hadoop YARN* y sobre el almacenamiento distribuido *Hadoop Distributed File System (HDFS)*.

- **Scala.** Scala es un lenguaje de programación multi-paradigma diseñado para expresar patrones comunes de programación en forma concisa, elegante y con tipos seguros. Integra sutilmente características de lenguajes funcionales y orientados a objetos. La implementación actual corre en la máquina virtual de Java y es compatible con las aplicaciones Java existentes.

Scala posee características propias de los lenguajes funcionales. En Scala las funciones son valores de primera clase, soportando funciones anónimas, orden superior, funciones anidadas y currificación. Scala viene integrado de fábrica con la técnica de *pattern matching* para modelar tipos algebraicos usados en muchos lenguajes funcionales. Está equipado con un sistema de tipos expresivo que refuerza a que las abstracciones de tipos se usen en forma coherente y segura en contraposición a Python por ejemplo.

- **Apache HDFS.** HDFS (Hadoop Distributed File System) es el software encargado del almacenamiento distribuido de datos en un clúster de servidores, especialmente creado para trabajar con grandes volúmenes de datos. En este tipo de sistema de archivos los datos son divididos en pequeñas partes, denominadas bloques. El tamaño de los bloques es superior al habitual, lo que permite reducir el tiempo en los accesos para lectura de datos. Estos bloques se almacenan de forma distribuida a través de un clúster siguiendo el patrón “Write once read many”, muy útil para almacenar grandes ficheros de logs

#### 5.4. TECNOLOGÍAS UTILIZADAS

que serán consultados en un gran número de veces. Esta forma de almacenamiento ayuda a que las funciones MapReduce, de las que hablaremos posteriormente, puedan ser ejecutadas sobre pequeños subconjuntos de bloques, alcanzando así una mayor escalabilidad. Por tanto un fichero es almacenado en un conjunto de pequeños bloques, que a su vez son replicados en los distintos servidores de todo el clúster de Hadoop. Por defecto cada bloque citado se ha replicado en 3 ocasiones, sin embargo se puede realizar un mayor número de réplicas.

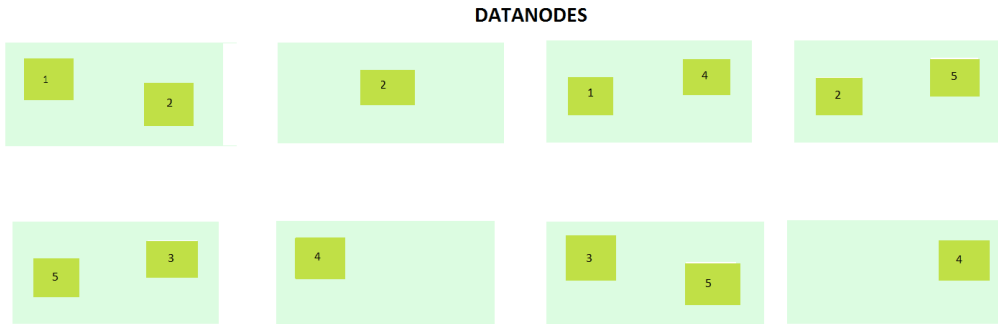


Figura 5.12: Replicación en clústeres utilizando data nodes.

HDFS sigue una arquitectura maestro/esclavo, en la que el nodo maestro es denominado NameNode y el resto de nodos se denominan DataNodes.

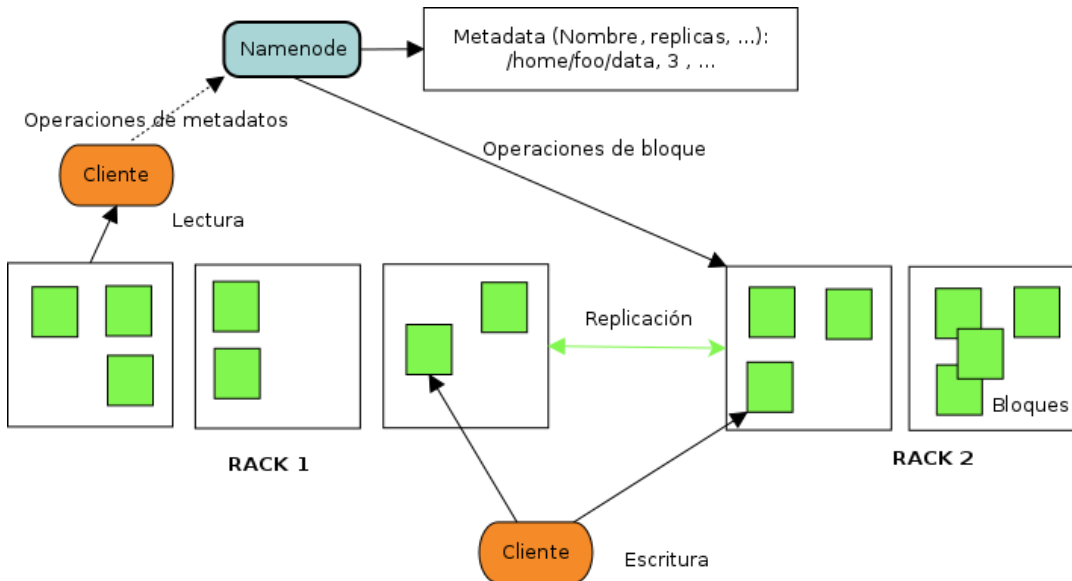


Figura 5.13: Gestión de acceso de los Name Nodes a los Data Nodes.

Los NameNode son servidores que gestionan el acceso a los archivos almacenados en

los DataNodes, para ello almacenan la información de qué bloques forman un archivo y donde se localizan (los metadatos de los archivos). Además de esto los NameNode ejecutan operaciones del sistema de fichero, tales como la apertura, el cierre o el renombrado de ficheros y directorios, así como el mapeado de los bloques en DataNodes. En cambio los DataNodes son los responsables de servir los requerimientos de lectura y escritura de los clientes del sistema de ficheros, además realiza la creación, eliminado y replicación de los bloques siguiendo las instrucciones ofertadas por el NameNode.

- **Map Reduce.** MapReduce es un paradigma de programación utilizado para dar soporte a la computación paralela sobre grandes colecciones de datos en clústers de servidores. Este paradigma es usado para la resolución práctica de algoritmos susceptibles de ser paralelizados. Normalmente son utilizados para abordar problemas con grandes volúmenes de datos suelen ejecutarse en HDFS. Gracias a MapReduce es posible ofrecer una escalabilidad enorme a través de cientos o miles de servidores de Apache Hadoop, estando formado por la operativa de las funciones Map () y Reduce (). Ambas funciones se ejecutan de forma distribuida en diversas máquinas. La función Map() tiene como objetivo la transformación de una entrada de datos, por filas de clave valor, en una salida clave/valor, como se puede ver en el ejemplo 1 siguiente.
- **Ejemplo 1** Tenemos un fichero con varias filas, con datos de ciudades y su temperatura registrada. El resultado de realizar Map() sobre los datos de este fichero de ejemplo sería: (Toronto, 20) (Whitby, 25) (New York, 22) (Rome, 32), (toronto,4) (Rome, 33) (New York, 18).

|              |
|--------------|
| Toronto, 20  |
| Whitby, 25   |
| New York, 22 |
| Rome, 32     |
| Toronto, 4   |
| Rome, 33     |
| New York, 18 |

Figura 5.14: Datos sobre los que trabajaremos en el ejemplo.

La función Reduce() es aplicada en paralelo para cada grupo de valores, produciendo una colección de valores para cada dominio, es decir, toma todos los valores de una clave específica y genera una nueva lista reducida como salida (Reduce (k2, list(v2)) -¿list(v3). Esto se puede ver en el ejemplo 2.

- **Ejemplo 2.** Sobre la lista anterior, realizamos una reducción para quedarnos solo con las temperaturas máximas de cada ciudad. El resultado de realizar Reduce() sobre las tuplas de datos obtenidas al realizar la operación Map() es: (Toronto, 20) (Whitby, 25) (New York, 22) (Rome, 33).

- Apache Hive.** Apache Hive surge como una tecnología de tipo data warehousing que permite gestionar grandes volúmenes de datos almacenados en HDFS, así como en otros tipos de almacenamiento como HBase. Hive dispone de un lenguaje de consulta declarativo, llamado HiveQL, muy similar a SQL. Se suele considerar que Hive es una abstracción de alto nivel de Map Reduce, ya que su motor de ejecución se encarga de traducir las consultas HQL en configuraciones de jobs Map Reduce, ejecutados directamente sobre la infraestructura Hadoop. El hecho de utilizar una sintaxis similar a SQL y su forma de trabajo hace que se asemeje a los sistemas de bases de datos tradicionales. Sin embargo Hive se encuentra pensado para entornos de data warehouse (OLAP), en los que los datos suelen ser estáticos y se demanda llevar a a cabo tareas analíticas de manera exhaustiva, siendo lentas las consultas en caso de compararlo con los sistemas tradicionales. En definitiva Hive es altamente utilizado para implementar ETL's, construir informes u obtener análisis específicos. Es considerada una de las principales tecnologías Big Data para el procesamiento y análisis de grandes volúmenes de información. En la siguiente figura es posible ver la posición que ocupa Hive en el ecosistema Hadoop.

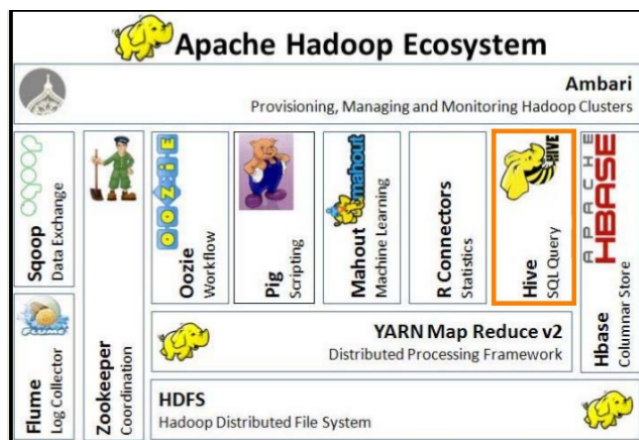


Figura 5.15: Hive en el ecosistema hadoop.

- Cloudera Impala.** Es un motor de consulta que corre en *Apache Hadoop*. El proyecto fue anunciado en octubre de 2012 con una distribución pública beta y se encuentra disponible para el público en general desde mayo de 2013. Con licencia Apache lleva la tecnología de base de datos escalable en paralelo a *Hadoop*, permitiendo a los usuarios realizar consultas SQL de baja latencia a los datos almacenados en *HDFS* y *Apache HBase* sin necesidad de movimiento o transformación de los datos. Impala está integrada con *Hadoop* para utilizar los mismos archivos y formato de datos, metadatos, seguridad y frameworks de gestión de recursos utilizados por *MapReduce*, *Apache Hive*, *Apache Pig* u otro software de Hadoop.

Sus características incluyen:

- Soporte de HDFS y almacenamiento Apache HBase

- Lee formatos de archivos de Hadoop, incluyendo texto, LZO, SequenceFile, Avro, RCFile, y Parquet
  - Soporta seguridad Hadoop (autenticación Kerberos)
  - Autorización fine-grained, basada en roles con Sentry
  - Utiliza metadata, controlador ODBC, y sintaxis SQL de Apache Hive
- **Bash Shell Script.** Es un programa diseñado para ser ejecutado por el shell de Unix, un intérprete de línea de comandos. Los diversos dialectos de los scripts de shell se consideran lenguajes de scripting. Las operaciones típicas realizadas por los scripts de shell incluyen la manipulación de archivos, la ejecución de programas y la impresión de texto.

La mayoría de los scripts de shell se utilizan en la parte de ETL de los proyectos de Business Intelligence, por lo que el aprendizaje será realmente útil para cuando se está trabajando en tales proyectos. Ahora que la mayoría de los servidores de Informática se ejecutan en el cuadro de Unix se debe comprender cómo programar las tareas, cómo acceder a los directorios de Unix, etc. Resulta de una herramienta muy útil y esencial en el ámbito del BI.

- **Crontab.** Crontab es un simple archivo de texto que guarda una lista de comandos a ejecutar en un tiempo especificado por el usuario. Crontab verificará la fecha y hora en que se debe ejecutar el script o el comando, los permisos de ejecución y lo realizará en el background. Cada usuario puede tener su propio archivo crontab. Es la manera más sencilla de administrar tareas de cron en sistemas multiusuario, ya sea como simple usuario de sistema o usuario root.

En nuestro caso además de ser usado a través de NiFi con la planificación de la lectura en tiempo real cada 60 segundos, tenemos un registro crontab en la máquina para el usuario “albguti”:

```
30 13 * * 1-5 /home/cloudera/scripts/StocksEndMarket_spark-submit.sh
```

Este nos ejecuta la aplicación desarrollada de cierre de mercado a las 13:30 de lunes a viernes (horario de la máquina, en Los Ángeles), media hora después de que cierre el NASDAQ. Este script que ejecuta será explicado más adelante en la subsección 5.5.5 HQL Scripts y Shell Scripts.

## 5.5. Implementación del Data Lake

En esta sección se persigue explicar la implementación de los diferentes programas y scripts utilizados para la implementación del Data Lake. La implementación del Data Lake se hace utilizando cinco tecnologías, previamente explicadas, NiFi, HDFS, Scripts, Hive e Impala. Con el objetivo de no hacer demasiado tediosa la explicación de esta sección se explicará lo que se ha implementado con cada tecnología, el motivo de su implementación y un breve ejemplo, en código, de cada una de ellas. A continuación se explicarán los diferentes procesamientos realizados sobre los datos, en función de la tecnología utilizada.

### 5.5.1. Hive.

Mediante Hive se persigue realizar las siguientes fases de procesamiento sobre la información, desde un nivel más elevado de abstracción. A continuación se mostrará las bases de datos y tablas que se tienen en concordancia con el origen de datos explicado en la sección 4.1 del Modelo de Datos, así como un ejemplo de Script de creación de uno de cada tipo de ellos.



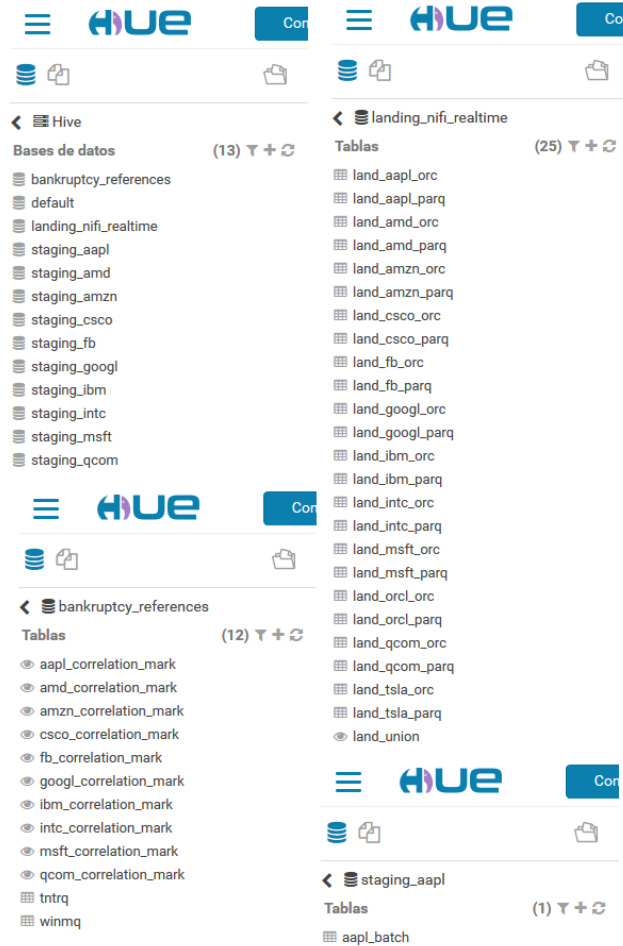


Figura 5.16: Ecosistema de BBDD y tablas de nuestro Datalake.

```
CREATE TABLE landing_nifi_realtime.land_aapl_orc(
  symbol string,
  companynname string,
  calculationprice string,
  open double,
  opentime bigint,
  close double,
  closetime bigint,
  high double,
  low double,
  latestprice double,
  latestsource string,
  latesttime string,
  latestupdate bigint,
```

```
latestvolume bigint,
iexrealtimeprice double,
iexrealtimesize string,
iexlastupdated string,
delayedprice double,
delayedpricetime bigint,
extendedprice double,
extendedchange double,
extendedchangepercent double,
extendedpricetime bigint,
previousclose double,
change double,
changepercent double,
iexmarketpercent string,
iexvolume string,
avgtotalvolume bigint,
iexbidprice string,
iexbidsize string,
iexaskprice string,
iexasksize string,
marketcap bigint,
peratio double,
week52high double,
week52low double,
ytdchange double)
CLUSTERED BY (
  latestupdate)
INTO 5 BUCKETS
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat'
LOCATION
  'hdfs://quickstart.cloudera:8020/user/albguti/
  landing_nifi_realtime/land_aapl_orc'
TBLPROPERTIES (
  'orc.create.index'='true',
  'transactional'='true');
```

\* El hecho de que esta tabla sea de tipo ORC es porque como veremos en la sección posterior de NiFi es un requerimiento del proceso que nos inserta los datos en tiempo real en esta tabla.

```
CREATE TABLE landing_nifi_realtime.land_aapl_parq
```

```

STORED AS PARQUET
LOCATION '/user/albguti/landing_nifi_realtime/land_aapl_parq'
AS SELECT * FROM landing_nifi_realtime.land_aapl_orc;

```

\*Es una versión de la tabla land\_[empresa]\_orc pero en formato parquet, para ser compatible con Apache Impala.

```

CREATE EXTERNAL TABLE staging_aapl.aapl_batch(
  date string,
  open double,
  high double,
  low double,
  close double,
  adj_close double,
  volume bigint)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
  'field.delim'='\u0001',
  'serialization.format'='\u0001')
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
LOCATION
  'hdfs://quickstart.cloudera:8020/user/albguti/staging/aapl_batch';

```

\* El hecho de que esta tabla sea externa es porque la información la ingestamos a mano con esos CSVs que hemos detallado en la sección 4.1 del Modelo de Datos, así como ya hemos explicado por qué elegimos ese carácter no imprimible SOH como separador.

Dentro de la base de datos *bankruptcy\_references* tenemos las tablas *tntrq* y *winmq* que son los datos históricos de empresas que han caído en el NASDAQ para esa comparación que se iba a realizar pero a la que no se ha llegado a tiempo más que a crear los indicadores en Impala con la creación en Impala de las vistas que se observan en la figura 5.16, que no son mas que la correlación lineal de las empresas con cada una de estas dos empresas que cayeron en banca rota.

### 5.5.2. Impala.

En Impala creamos los indicadores de bancarota con la creación en Impala de las vistas que se observan en la figura 5.16, que no son mas que la correlación lineal de las empresas con cada una de estas dos empresas que cayeron en banca rota, con una ponderación de 2 para *Tintri, Inc.* y de 1 para *Windstream Holdings, Inc.*:

```

CREATE VIEW bankruptcy_references.aapl_correlation_mark AS

```

```
SELECT 'Apple, Inc.' companyname, (2 * ((avg(IND1.close * DEP.close)
- avg(IND1.close) * avg(DEP.close)) / (1.00000000 * stddev_pop(IND1.close)
* stddev_pop(DEP.close)))) + ((avg(IND2.close * DEP.close) - avg(IND2.close)
* avg(DEP.close)) / (1.00000000 * stddev_pop(IND2.close) * stddev_pop(DEP.close)))
correlation_mark FROM bankruptcy_references.tntrq ind1
INNER JOIN staging_aapl.aapl_batch dep
INNER JOIN bankruptcy_references.winmq ind2;
```

### 5.5.3. HDFS.

Aquí tendremos la abstracción entendida como una organización en directorios de todo nuestro ecosistema explicado como vemos en la figura 5.17.

| Nombre                                     | Tamaño    | Usuario | Grupo   | Permisos   | Fecha                       |
|--|-----------|---------|---------|------------|-----------------------------|
| ./   |           | albguti | albguti | drwxrwxrwx | June 20, 2019 08:07 AM      |
| ./Trash                                    |           | albguti | albguti | drwxrwxrwx | July 09, 2019 12:01 PM      |
| ./sparkStaging                             |           | albguti | albguti | drwxrwxrwx | July 09, 2019 07:29 AM      |
| ./staging                                  |           | albguti | albguti | drwx---    | July 09, 2019 12:06 PM      |
| 2015_11_18                                 |           | albguti | albguti | drwxrwxrwx | August 23, 2018 03:16 AM    |
| 2015_11_19                                 |           | albguti | albguti | drwxrwxrwx | August 23, 2018 03:16 AM    |
| 2015_11_20                                 |           | albguti | albguti | drwxrwxrwx | August 23, 2018 03:16 AM    |
| 2015_11_21                                 |           | albguti | albguti | drwxrwxrwx | August 23, 2018 03:16 AM    |
| bankruptcy_references                      |           | albguti | albguti | drwxrwxrwx | June 20, 2019 09:11 AM      |
| example.json                               | 910 bytes | albguti | albguti | -rwxrwxrwx | August 29, 2018 05:31 AM    |
| json-serde-1.3.7-jar-with-dependencies.jar | 80,2 KB   | albguti | albguti | -rwxrwxrwx | May 25, 2019 03:07 AM       |
| json-serde-1.3.8-jar-with-dependencies.jar | 83,5 KB   | albguti | albguti | -rwxrwxrwx | May 25, 2019 03:10 AM       |
| landing_nifi_realtime                      |           | albguti | albguti | drwxrwxrwx | July 09, 2019 08:23 AM      |
| oozie-oozi                                 |           | albguti | albguti | drwxrwxrwx | September 20, 2018 04:01 AM |
| staging                                    |           | albguti | albguti | drwxrwxrwx | June 18, 2019 03:53 PM      |

Figura 5.17: Vista de directorios para albguti en el HDFS.

Dentro de la carpeta “landing\_nifi\_realtime” tenemos, como hemos indicado en los scripts de creación, las tablas “land-[empresa]\_orc” y “land-[empresa]\_parq” que se ingestarán en el próximo paso con NiFi y con scripts de Hive HQL que ya explicaremos en su subsección futura.

En la carpeta “staging” tenemos, también como hemos indicado en los scripts de creación, las tablas “[empresa]\_batch” ingestadas en batch que contienen la información histórica de las empresas. Estas son ingestadas en primera instancia a mano con los CSVs y después

mantenidas con la inserción cada día de los datos de cierre de mercado con la ejecución de la aplicación Spark-Scala que hemos desarrollado y que también veremos en su futura sección.

Así mismo dentro de la carpeta “bankruptcy\_references” tenemos los históricos de esas dos empresas referencia de bancarota que hemos ingestado.

### 5.5.4. NiFi.

Con todo el ecosistema de tablas montado en nuestro Data Lake con Hive e Impala y los archivos ingestados sobre el HDFS procederemos a preparar nuestro flow de lectura de JSONs con NiFi en tiempo real.

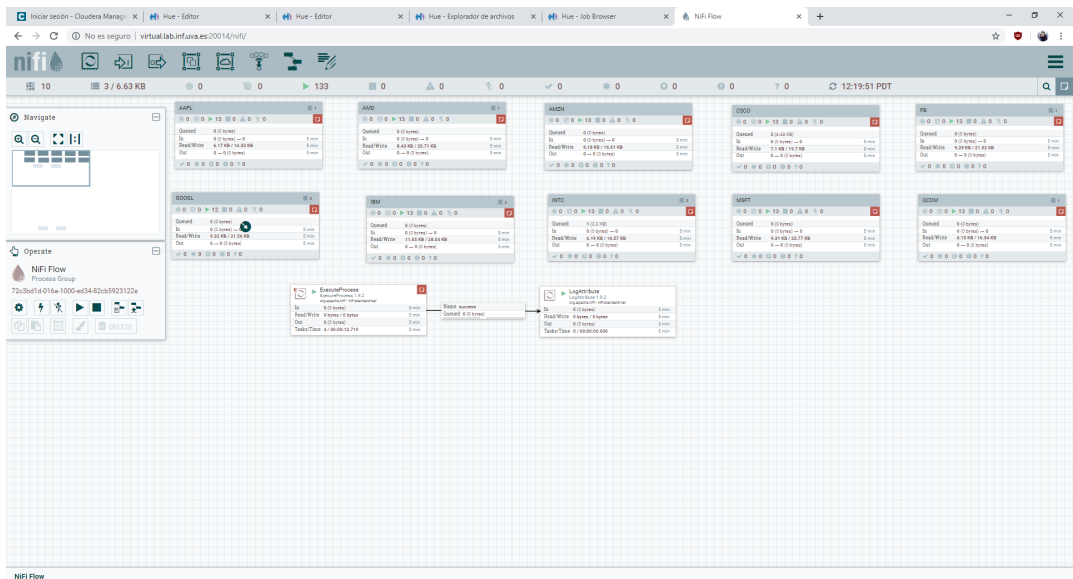


Figura 5.18: Pantalla inicial de NiFi con nuestro flow.

En esa figura 5.18 vemos la pantalla inicial de nuestro NiFi que contiene todos los 10 grupos de procesos que tenemos, uno por cada empresa, y el proceso especial “ExecuteProcess” de la figura 5.19 que nos realiza un invalidate metadata cada 60 segundos en Impala.

Este proceso “ExecuteProcess” ejecuta un comando del sistema operativo especificado por el usuario y escribe la salida de ese comando en un FlowFile. Si se espera que el comando sea de larga ejecución, el procesador puede generar los datos parciales en un intervalo específico. Cuando se usa esta opción, se espera que la salida esté en formato de texto, ya que normalmente no tiene sentido dividir los datos binarios en intervalos arbitrarios basados en el tiempo. Nosotros ejecutamos un Shell Script que se conecta con Impala como veremos en la próxima subsección 5.5.5 HQL Scripts y Shell Scripts.

## 5.5. IMPLEMENTACIÓN DEL DATA LAKE

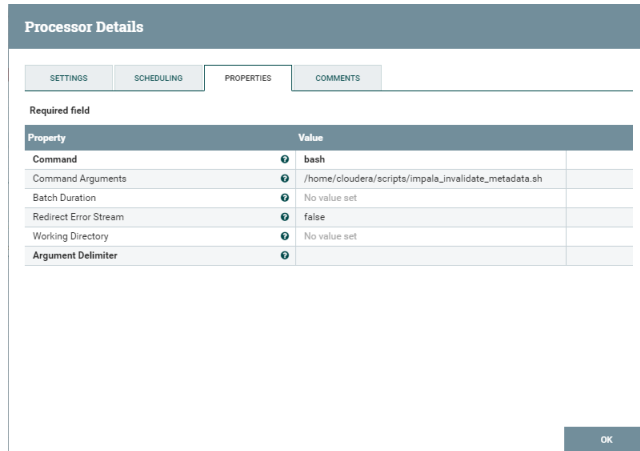


Figura 5.19: Invalidate Metadata en Impala con ExecuteProcess de NiFi.

Veremos un ejemplo de uno de esos 10 procesos, ya que el resto son exactamente iguales excepto que cambia las referencias correspondientes a cada empresa. Veremos el ejemplo de le empresa "Apple, Inc." que tenemos en nuestro Data Lake.

A la vista de la figura 5.20 el flow de cada empresa es sencillo aunque llegar a tal sencillez ha sido un arduo trabajo. Consta de la consecución de 3 procesos más un proceso extra que nos ejecuta un script de transformación de nuestra ingesta Real Time en ORC a una tabla Parquet con las transformaciones que necesitamos.

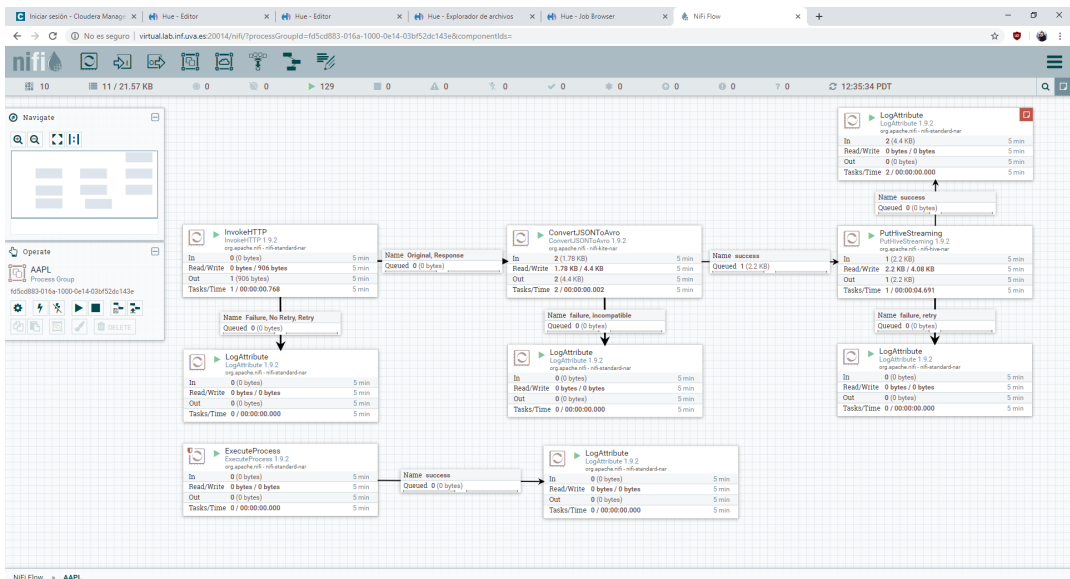


Figura 5.20: Nuestro flow de NiFi para aapl.

## InvokeHTTP

Es el primero de nuestros procesos de lectura en Real Time del mercado de valores. Es un procesador de cliente HTTP que puede interactuar con un punto final HTTP configurable. La URL de destino y el método HTTP son configurables. Los atributos de FlowFile se convierten en encabezados HTTP y los contenidos de FlowFile se incluyen como el cuerpo de la solicitud (si el método HTTP es PUT, POST o PATCH).

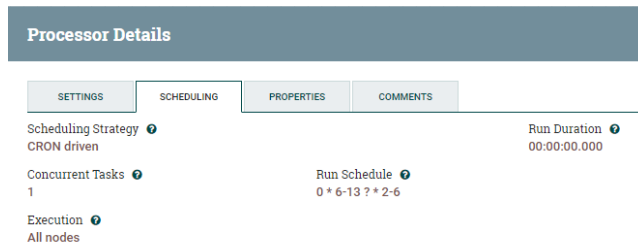


Figura 5.21: Planificación con CRON de InvokeHTTP para ejecutarse cada minuto de 6 a 13 horas y solo de lunes a viernes (cuando está abierto es NASDAQ).

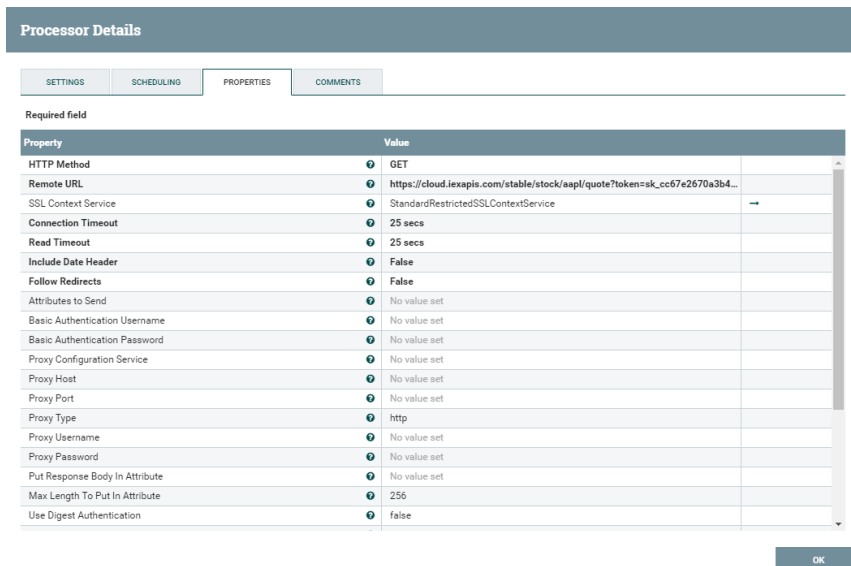


Figura 5.22: Valores de las propiedades de configuración de InvokeHTTP para recoger los JSONs de la API de IEX Trading.

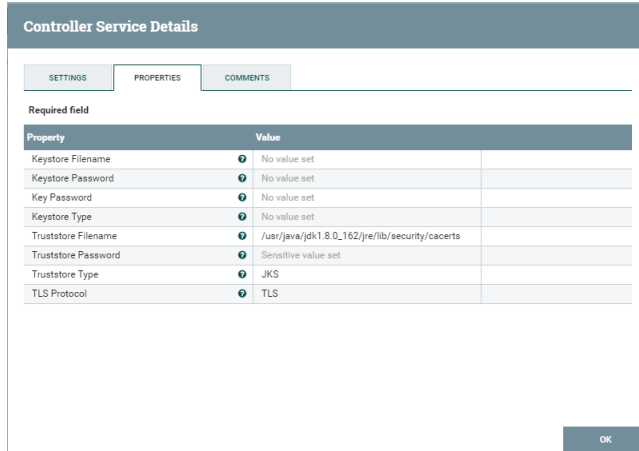


Figura 5.23: Configuración del StandardRestrictedSSLContextService necesario para la ejecución del proceso InvokeHTTP.

En las anteriores figuras 5.21, 5.22 y 5.23 vemos en detalle cómo lo tenemos configurado para acceder a la API de IEX Trading con SSL y recogerlos el JSON en crudo que esta devuelve.

## ConvertJSONToAvro

Este proceso de NiFi convierte archivos de tipo JSON a Avro según un esquema de Avro que se le pasa.

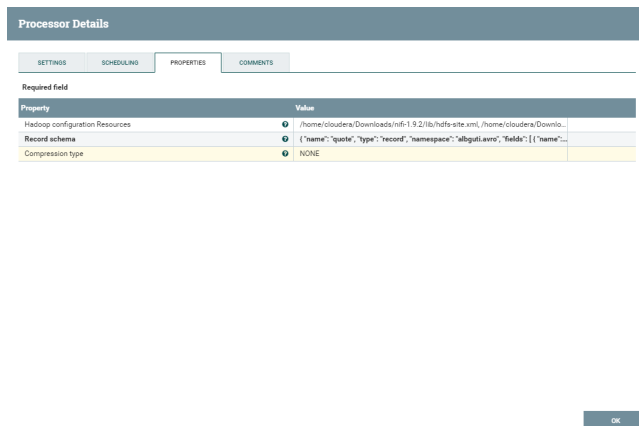


Figura 5.24: Configuración del ConvertJSONToAvro donde se le indica el esquema Avro que debe seguir para la conversión.



Y este es el esquema Avro concreto que hemos desarrollado y que utilizamos para nuestro caso de uso. Es muy sencillo y autodestructivo, contiene un mapeo de cada uno de los campos que nos devuelve la API a su formato.

```
{
    "name": "quote",
    "type": "record",
    "namespace": "albguti.avro",
    "fields": [
        {
            "name": "symbol",
            "type": [
                "string",
                "null"
            ]
        },
        {
            "name": "companyName",
            "type": [
                "string",
                "null"
            ]
        },
        {
            "name": "calculationPrice",
            "type": [
                "string",
                "null"
            ]
        },
        {
            "name": "open",
            "type": [
                "double",
                "null"
            ]
        },
        {
            "name": "openTime",
            "type": [
                "long",
                "null"
            ]
        },
        {
            "name": "close",
            "type": [
                "double",
                "null"
            ]
        },
        {
            "name": "closeTime",
            "type": [
                "long",
                "null"
            ]
        },
        {
            "name": "high",
            "type": [
                "double",
                "null"
            ]
        },
        {
            "name": "low",
            "type": [
                "double",
                "null"
            ]
        },
        {
            "name": "latestPrice",
            "type": [
                "double",
                "null"
            ]
        },
        {
            "name": "latestSource",
            "type": [
                "string",
                "null"
            ]
        },
        {
            "name": "latestTime",
            "type": [
                "string",
                "null"
            ]
        }
    ]
}
```

```
    "null"
  ]
},
{
  "name": "latestUpdate",
  "type": [
    "long",
    "null"
  ]
},
{
  "name": "latestVolume",
  "type": [
    "long",
    "null"
  ]
},
{
  "name": "iexRealtimePrice",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "iexRealtimeSize",
  "type": [
    "long",
    "null"
  ]
},
{
  "name": "iexLastUpdated",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "delayedPrice",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "delayedPriceTime",
  "type": [
    "long",
    "null"
  ]
},
{
  "name": "extendedPrice",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "extendedChange",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "extendedChangePercent",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "extendedPriceTime",
  "type": [
    "long",
    "null"
  ]
},
{
  "name": "previousClose",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "change",
  "type": [
    "double",
    "null"
  ]
},
}
```

```

{
  "name": "changePercent",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "iexMarketPercent",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "iexVolume",
  "type": [
    "long",
    "null"
  ]
},
{
  "name": "avgTotalVolume",
  "type": [
    "long",
    "null"
  ]
},
{
  "name": "iexBidPrice",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "iexBidSize",
  "type": [
    "long",
    "null"
  ]
},
{
  "name": "iexAskPrice",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "iexAskSize",
  "type": [
    "long",
    "null"
  ]
},
{
  "name": "marketCap",
  "type": [
    "long",
    "null"
  ]
},
{
  "name": "peRatio",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "week52High",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "week52Low",
  "type": [
    "double",
    "null"
  ]
},
{
  "name": "ytdChange",
  "type": [
    "double",
    "null"
  ]
}
}

```

## PutHiveStreaming

Este procesador utiliza *Hive Streaming* para enviar datos de archivos de flujo a una tabla de *Apache Hive*. Se espera que el archivo de flujo entrante esté en formato Avro y la tabla debe existir en Hive. Los valores de partición se extraen del registro de Avro en función de los nombres de las columnas de partición que se especifican en el proceso.

Este procesador que nos soluciona mucho tiene unos requisitos muy estrictos [17] y estos son:

- La transmisión de Hive es compatible con tablas que cumplen lo siguiente:
  - ORC es el único formato compatible actualmente. Entonces la tabla destino debe tener “stored as orc” en su declaración como nosotros ya tenemos.
  - Se debe establecer transactional = “true” en la declaración de creación de esta tabla destino.
  - En cubos pero no ordenado. Por lo tanto, la tabla debe tener “clustered by (colName) into (n) buckets” como nosotros tenemos “CLUSTERED BY (latestupdate) INTO 5 BUCKETS”.
- Además, Hive debe tener las siguientes propiedades establecidas en su archivo de configuración:
  - hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager
  - hive.compactor.initiator.on = true
  - hive.compactor.worker.threads >0

| Processor Details            |                   |   |          |
|------------------------------|-------------------|---|----------|
| SETTINGS                     | SCHEDULING        | PROPERTIES  | COMMENTS |
| Required field               |                   |   |          |
| Property                     | Value             |   |          |
| Hive Metastore URI           | <a href="#">?</a> | thrift://quickstart.cloudera:9083   |          |
| Hive Configuration Resources | <a href="#">?</a> | /etc/hive/conf/hive-site.xml,/home/cloudera/Downloads/nifi-1.9.2/lib/hdfs-site... |          |
| Database Name                | <a href="#">?</a> | landing_nifi_realtime   |          |
| Table Name                   | <a href="#">?</a> | land_aapl_orc   |          |
| Partition Columns            | <a href="#">?</a> | No value set  |          |
| Auto-Create Partitions       | <a href="#">?</a> | true  |          |
| Max Open Connections         | <a href="#">?</a> | 8   |          |
| Heartbeat Interval           | <a href="#">?</a> | 30  |          |
| Transactions per Batch       | <a href="#">?</a> | 100   |          |
| Records per Transaction      | <a href="#">?</a> | 10000   |          |
| Call Timeout                 | <a href="#">?</a> | 0   |          |
| Rollback On Failure          | <a href="#">?</a> | false   |          |
| Kerberos Credentials Service | <a href="#">?</a> | No value set  |          |
| Kerberos Principal           | <a href="#">?</a> | No value set  |          |
| Kerberos Keytab              | <a href="#">?</a> | No value set  |          |

Figura 5.25: Configuración del proceso PutHiveStreaming de NiFi que insertará cada Avro generado por el anterior proceso en la tabla de destino que se le configure.

## ExecuteProcess

Este segundo proceso de tipo “ExecuteProcess” ejecuta un comando del sistema operativo especificado en su configuración (figura 5.26). Nosotros ejecutamos un Shell Script que se conecta con Impala como veremos en la próxima subsección 5.5.5 HQL Scripts y Shell Scripts.

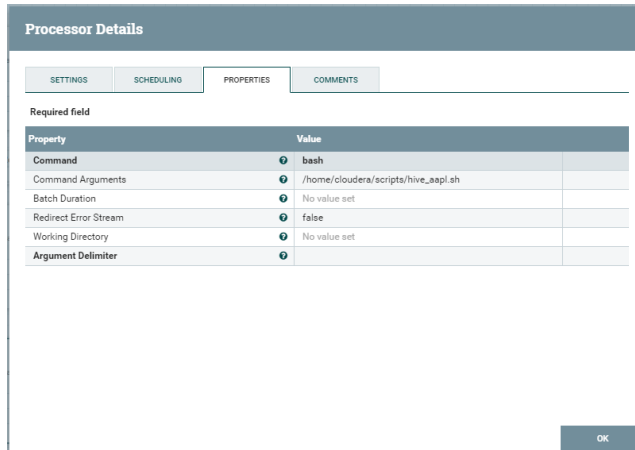


Figura 5.26: Configuración del proceso ExecuteProcess de NiFi que ejecuta el script que genera la respectiva actualización a la tabla Parquet.

### 5.5.5. HQL Scripts y Shell Scripts

Como en cualquier proyecto de *Business Intelligence* dentro de los que puede entrar la naturaleza del Big Data, es difícil que no se tenga que utilizar Scripts de Shell para ciertas tareas.

Los Scripts de Shell son programas que permiten a los usuarios interactuar con el sistema, procesando las órdenes que se le indican. En nuestro caso tenemos los siguientes scripts:

- hive\_[empresa].sh:** Son 10 scripts, uno por cada empresa que tenemos en nuestro Data Lake, donde lo único que hacemos es invocar a Hive a través del intérprete *Beeline* para ejecutar un Script de Hive .hql de la siguiente manera por ejemplo para el caso de “Apple, Inc.”:

```
#!/bin/bash
beeline -u jdbc:hive2://localhost:10000/
landing_nifi_realtime -n albguti -p *****
-f /home/cloudera/scripts/hql/hive_aapl.hql
```

Este script se ejecuta a través de NiFi cada 60 segundos mientras que el NASDAQ está abierto.

- **hive\_[empresa].hql:** Son scripts de tipo HQL. Hive proporciona una CLI (Interfaz de línea de comandos) para escribir consultas de Hive mediante el lenguaje de consulta de Hive (HiveQL). En general, la sintaxis HQL es similar a la sintaxis SQL. Estos ficheros .hql contienen sentencias a ejecutar en Hive como es el ejemplo de la siguiente para el caso de “Apple, Inc.”:

```
INSERT OVERWRITE TABLE landing_nifi_realtime.land_aapl_parq
SELECT
    symbol, companyname, calculationprice, open, opentime, close,
    [...],
    week52high, week52low, ytdchange,
    to_utc_timestamp(from_unixtime(latestupdate DIV 1000),
    'America/Los_Angeles') AS dateTime
FROM landing_nifi_realtime.land_aapl_orc;
```

Este script se ejecuta a través de la invocación por un Script de Shell que ejecuta NiFi cada 60 segundos mientras que el NASDAQ está abierto.

- **impala\_invalidate\_metadata.sh:** Parecido al caso de los primeros scripts que explicamos aquí, este invoca a Impala a través de su intérprete *impala-shell* para ejecutar un *INVALIDATE METADATA*, que marca los metadatos de una o todas las tablas como obsoletos. Este comando se requiere después de crear una tabla a través de Hive, para hacer la sincronización de la metadata entre ambos y hacer que la tabla esté disponible para las consultas de Impala. La próxima vez que el nodo Impala realice una consulta en una tabla cuyos metadatos se invaliden, Impala vuelve a cargar los metadatos asociados antes de continuar con la consulta. Nuestro Script es el siguiente:

```
#!/bin/bash
impala-shell -q 'INVALIDATE METADATA;'
```

Este script se ejecuta a través de NiFi cada 60 segundos mientras que el NASDAQ está abierto.

- **StocksEndMarket\_spark-submit.sh:** Este script de shell tiene que ver con la siguiente sección 5.6 del programa de cierre del mercado. Es el script que lanza al nodo la ejecución del proceso Spark que realiza el cierre del mercado una vez al día, pasando los datos en tiempo real a un dato histórico. El Script es de la siguiente manera, y ejecuta el lanzamiento al nodo del propio proceso Spark invocando a la clase “*everisStocks.StocksEndMarket*” que lo implementa, seguido de una regeneración y limpieza de los permisos de las carpetas del HDFS que este proceso toca y acabando de nuevo con un “*INVALIDATE METADATA*” cuya función se ha descrito en el ítem anterior de este listado:

```
#!/bin/bash
echo "RUNNING SPARK SUBMIT"
spark2-submit --class everisStocks.StocksEndMarket
```

```
--master yarn --deploy-mode cluster
/home/cloudera/IdeaProjects/ReadStocksStreaming/
target/scala-2.11/readstocksstreaming_2.11-0.1.jar
echo "REGENERATING HDFS PERMISSIONS AFTER TRUNCATED DATA"
hdfs dfs -chmod -R 777 /user/albguti/landing_nifi_realtime
echo "RUNNING INVALIDATE METADATA"
impala-shell -q 'INVALIDATE METADATA;'
```

\* Todos estos Scripts está incluidos en los contenidos del CD-ROM donde se entrega este proyecto.

## 5.6. Implementación del programa de cierre del mercado

Se ha desarrollado una sencilla aplicación “*ReadStocksStreaming*” desarrollada en *Spark-Scala* y compilada y empaquetada con *sbt*. El fin de esta no es más que realizar el cierre de mercado de nuestro proyecto.

Esto es que los datos que leemos en tiempo real cada día durante el funcionamiento del NASDAQ, al cierre de este estos datos en tiempo real pasan a ser históricos, y esto se hace con la última lectura donde el campo C18 del 4.1 Modelo de Datos “latestSource” que nos devuelve la API de IEX Trading está en valor “Close”.

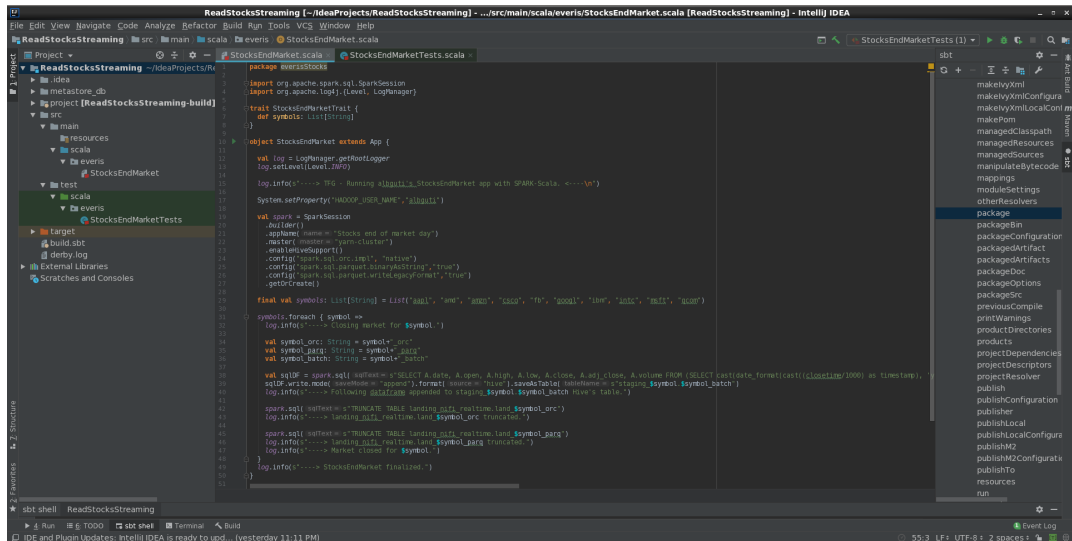


Figura 5.27: Captura de pantalla que muestra el código de nuestra aplicación Spark-Scala abierta en IntelliJ.

- Lo primero de todo es establecer una sesión Spark con la máquina preparada que vaya a ejecutar la aplicación,
- a esta sesión se le da un nombre “Stocks end of market day”,
- se le indica que debe ser ejecutada en el cluster a través de Yarn (Como ya hemos comentado en este documento, Spark-On-Yarn se encargaría en un entorno multi-nodo del reparto de todo este trabajo entre los nodos “workers”, para nosotros esto pasa a ser transparente a parte de que contamos con una máquina de laboratorio mono-nodo).
- Se le establece una propiedad (enableHiveSupport()) para que Spark2 pueda entenderse con el intérprete de Hive.
- Se le establece la propiedad “spark.sql.orc.impl”. Esto es el nombre de la implementación de ORC. Puede ser nativo o nativo a través de Hive. Nativo significa que el soporte



ORC se basa en Apache ORC 1.4. mientras que sobre Hive significa que el soporte se basa en la biblioteca ORC en Hive 1.2.1.

- Se le establece la propiedad “spark.sql.parquet.binaryAsString=true”. Algunos otros sistemas que producen ficheros parquet, en particular Impala, Hive y versiones anteriores de Spark SQL, no distinguen entre datos binarios y cadenas al escribir sobre el esquema de Parquet. Esta propiedad le dice a Spark que interprete los datos binarios como una cadena.
- También se le establece la propiedad “spark.sql.parquet.writeLegacyFormat=true”. Si es verdadero, los datos se escribirán de forma Spark 1.4 y anteriores. Por ejemplo, los valores decimales se escribirán en el formato de matriz de bytes de longitud fija de Apache Parquet. Si es falso, se usará el formato más nuevo en Parquet. Por ejemplo, los decimales se escribirán en formato basado en int. Si la salida de Parquet está pensada para su uso con sistemas que no admiten este formato más reciente la documentación recomienda establecerlo en verdadero.

Por cada una de las 10 empresas que tenemos presentes en nuestro Data Lake se realiza el mismo proceso:

- De su correspondiente tabla “landing\_nifi\_realttime.land\_[empresa]\_orc” se selecciona la última de las lecturas cuyo campo “latestSource” sea “Close” como ya se ha explicado con anterioridad para recoger el cierre del mercado.
- solo se seleccionan los campos que cuadran con la tabla histórica de destino, estos son:
  - date: Campo C1 del modelo de datos. Se transforma recogiendo el timestamp del C14 del modelo de datos “closeTime” transformado a fecha compatible con la tabla histórica de destino.
  - open: Campo C2 del modelo de datos. Se recoge directo desde el campo C11 del modelo de datos.
  - high: Campo C3 del modelo de datos. Se recoge directo desde el campo C15 del modelo de datos.
  - low: Campo C4 del modelo de datos. Se recoge directo desde el campo C16 del modelo de datos.
  - close: Campo C5 del modelo de datos. Se recoge directo desde el campo C13 del modelo de datos.
  - adj\_close: Campo C6 del modelo de datos. Se recoge directo desde el campo C13 del modelo de datos.
  - volume: Campo C7 del modelo de datos. Se recoge directo desde el campo C21 del modelo de datos.
- Se borran los datos de las tablas de lectura en tiempo real del tipo “landing\_nifi\_realttime.land\_[empresa]\_orc” y “landing\_nifi\_realttime.land\_[empresa]\_parq” para dejarlas limpias para la siguiente jornada.

Así tendríamos el día actual historificado y el Data Lake preparado y optimizado para una nueva jornada del mercado de valores.

\* El detalle de esta implementación, su código, puede verse a fondo en los contenidos del CD-ROM donde se entrega este proyecto.

### 5.6.1. Pruebas del programa de cierre del mercado

Como la aplicación desarrollada es muy sencilla además de constar de un caracter muy imperativo, el posible testing a realizar queda muy reducido pasando a ser la metodología del ensayo-error la más apropiada. Aun asi bien es cierto que se han implementado dos sencillos casos de prueba:

1. **numberOfStocks:** Test que comprueba que la longitud de la lista de empresas que utiliza la aplicación coincide con el número entero 10. Esto es que sabemos que tenemos ingestadas 10 empresas en nuestro Data Lake y no queremos que haya algún tipo de desfase.
2. **numberOfStocks2:** Este test por su lado lo que comprueba es que, en conjunto con el anterior test, coincide el numero de elementos de la lista que contiene las 10 empresas con el número de elementos distintos que tiene la lista. Eso es que además de ser 10, están todos porque ninguno es repetido.

\* Los detalles de este testing pueden verse a fondo en los contenidos del CD-ROM donde se entrega este proyecto, dentro del código de la aplicación.

## 5.7. Implementación del dashboard

Se ha implementado un Dashboard para el consumo y visualización de todos los datos, transformaciones y procesos descritos a lo largo de este documento. Es de vital importancia la visualización de los datos, no es nada sorprendente recapacitar sobre que por muy buen Data Lake que gestiones, por muchos datos y grandes que sean estos, si al final no hay una aplicación final que los consuma de poco sirve todo ese arduo trabajo. Más aun si hablamos del mercado laboral, donde los clientes y sobre todo los cargos altos de las empresas que ejercen de cliente pasan desapercibidos ante todos estos detalles de la implementación y basan un gran porcentaje de su percepción sobre la implementación en cómo de atractivos y de qué manera útil para él se recogen esos datos.

Sobre la elección de Power BI ya se ha hablado en apartados anteriores, resumiendo su crecimiento en el mercado es continuo y con un especial repunte en el último año, además de que su versión gratuita cubre todos los requisitos para este proyecto.

A partir de los mockups establecidos en la fase de Diseño en la sección 4.4 de este documento se han desarrollado los tres paneles que pueden verse en las siguientes tres figuras.

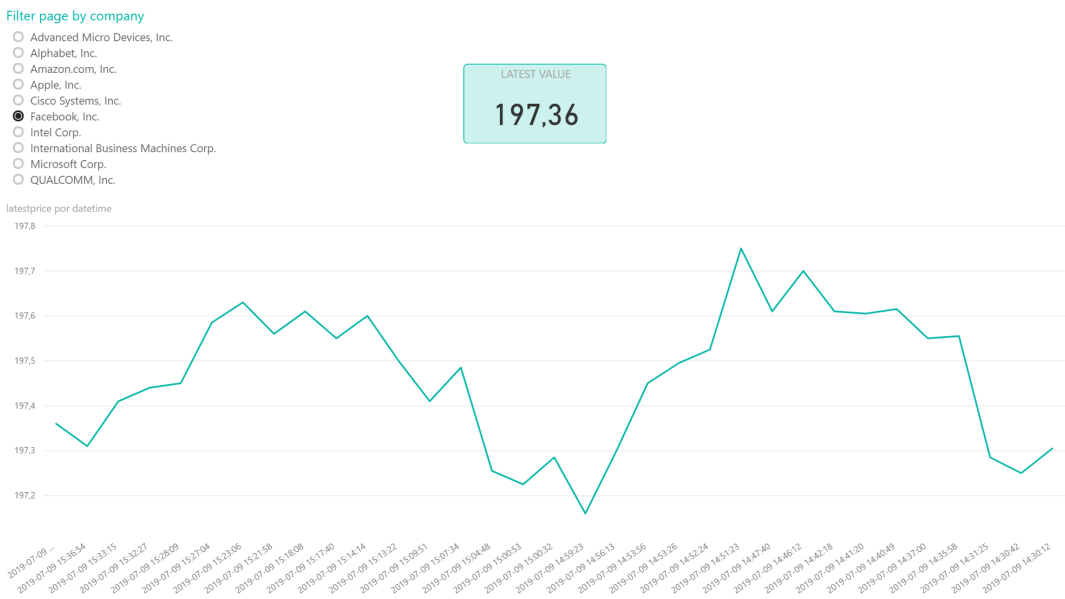


Figura 5.28: Captura de pantalla del Dashboard para los datos recogidos en tiempo real con la empresa Facebook, Inc. seleccionada.

## 5.7. IMPLEMENTACIÓN DEL DASHBOARD



Figura 5.29: Captura de pantalla del Dashboard para los datos históricos con la empresa AMD, Inc. seleccionada y filtrando por fecha desde el 8 de mayo de 2017 al 28 de Junio de 2019.

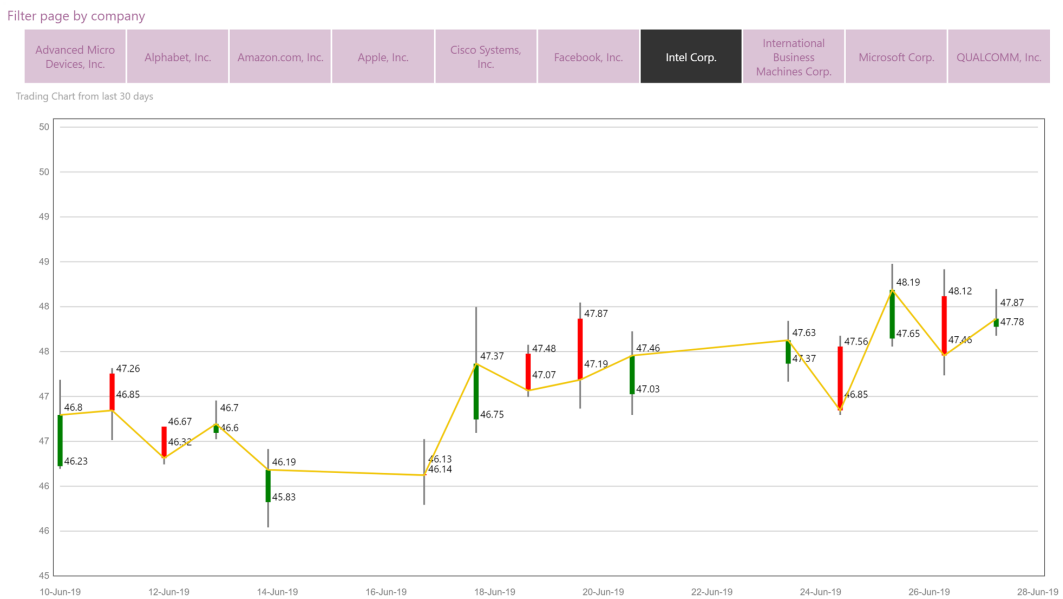


Figura 5.30: Captura de pantalla del Dashboard para el gráfico de velas de los 30 últimos días con la empresa Intel, Corp. seleccionada.

Esta aplicación está publicada y disponible accediendo al siguiente enlace <https://app.powerbi.com/groups/me/reports/ab106d3e-0c69-49c9-a908-b7adb97e327c?ctid=3048dc87-43f> aunque es necesario disponer de una cuenta de Power BI.

\* Esta aplicación de Power BI también está incluida en los contenidos del CD-ROM donde se entrega este proyecto, como un archivo con formato *.pbix*.



## Capítulo 6

# Conclusiones

### 6.1. Conclusiones

En este último capítulo del proyecto se presentarán las conclusiones derivadas de la realización de este proyecto, así como las principales metas alcanzadas. Además de esto se plantearán posibles mejoras y ampliaciones del proyecto desarrollado. A lo largo de este proyecto me he encontrado con gran cantidad de dificultades, siendo la principal de éstas el desconocimiento total tanto de la tecnología a utilizar como del sistema bursátil y el mercado de valores. Esta dificultad nos obligó a comenzar a investigar dichos elementos a nivel de usuario, para luego tratar de comprender el funcionamiento del mercado, el lenguaje bursátil o los diferentes agentes que influyen en las cotizaciones de una empresa. Una vez investigado este ecosistema se pudo desarrollar el modelo de datos del proyecto, el cual es la piedra angular de este.

Otra gran dificultad a la que me enfrenté fue el desconocimiento de las diferentes arquitecturas Big Data, así como la dificultad a la hora de acceder a fuentes fiables de información con las que entender qué es un Data Lake, ya que a pesar de estar ya en un entorno laboral trabajando con ello llega a ser un concepto muy abstracto.

Además de las dificultades iniciales asociadas a la investigación, anteriormente descritas, me he encontrado con gran cantidad de problemas derivados de la falta de recursos para tratar grandes volúmenes de información. Estos problemas se deben a la falta de memoria RAM de la máquina virtual con la que se realiza el proyecto, así las diferentes restricciones hardware que afectan a la máquina sobre la que se monta el ecosistema Big Data.

Por otro lado el trabajo con grandes volúmenes de datos y con tecnologías muy novedosas, como las tecnologías Big Data utilizadas para el almacenamiento y procesamiento de la información asociadas a la arquitectura propuesta, me ha supuesto gran cantidad de problemas, al tener que afrontar un gran número de complicaciones no planteadas con anterioridad y derivadas del gran volumen de información al que se puede tener acceso. Se ha tenido que recortar a 10 las empresas con las que realizar este caso de uso.

A lo largo de este proyecto se han dado gran cantidad de problemas asociados con el sistema operativo en el que se ha montado el ecosistema Big Data, como por ejemplo la necesidad del JDK 1.8 entrando en conflicto con la configuración de todas las herramientas que Cloudera lleva por defecto sobre JDK 1.7. Estos problemas me han permitido refrescar los conocimientos adquiridos en la carrera sobre sistemas, así como a dominar un nuevo lenguaje de programación, de muchísimo auge en la actualidad y sobre el que me estoy especializando en mi carrera profesional, como es Scala.

La principal conclusión derivada del desarrollo de este proyecto es que este tipo de tecnologías son ya el presente de la informática, ya que en la actualidad el mundo se mueve en torno a grandes volúmenes de información. El tratamiento de estos volúmenes de información supone un gran cambio que se engloba dentro de la transformación digital a la que se enfrentan gran cantidad de empresas en la actualidad. Es por eso que el desarrollo de este proyecto me ha supuesto un aprendizaje continuo y de gran valor para mi presente y futuro laboral, ya que el aprendizaje de estas tecnologías supone la incursión en un mundo laboral falto de profesionales en este ámbito.

Otra conclusión que se extrae de este TFG es la gran necesidad de adaptación al cambio necesaria a la hora de desarrollar un proyecto real. Este hecho se asocia principalmente a la gran velocidad de evolución de la tecnología, así como a la diferente visión, sobre el producto a desarrollar. Es el hecho de que durante la realización de este proyecto se ha hablado mucho sobre la explosión de la burbuja del Big Data, concepto que subjetivamente opino que es tan cierto como positivo ya que muchos de los proyectos que se vendían como necesidad de Big Data en verdad eran fallidos porque la volumetría y naturaleza en verdad no lo requería. Al mismo hilo que durante estos meses de elaboración de este proyecto ha evolucionado todo tan rápido que a día de hoy plantearía otras tecnologías que emergen y que son muy distintas a las escogidas.

Sin embargo, la conclusión más interesante que he podido extraer de este TFG es el gran nicho de mercado que supone la aplicación de tecnologías Big Data al ámbito bursátil. La implantación de este tipo de soluciones es muy numerosa pero la facilidad sobre el manejo de estas herramientas permitirá el desarrollo de soluciones muy ajustadas a necesidades muy concretas con costes bajos.



## 6.2. Trabajo futuro

Aunque la herramienta desarrollada en este proyecto es completamente funcional y escalable existen diversos aspectos que se pueden mejorar con el paso del tiempo.

- **Uso de bases de datos NOSQL:** Se podrían utilizar bases de datos NOSQL con el fin de agilizar la visualización de la información en el Dashboard desarrollado.
- **Ampliar los datos:** Pese a que para esta fase inicial del proyecto los datos con los que se ha contado han sido suficientes se podría adquirir mejor hardware que permitieran el seguimiento de más empresas, así como el mejor análisis de estos en tiempo real. Esto además facilitaría el desarrollo de gran cantidad de métricas de interés que al final no ha dado tiempo a cubrir.
- **Optimizar fases de limpieza:** Pese al desarrollo de diferentes fases de limpieza debido a los problemas tecnológicos a los que nos hemos enfrentado, así como a la dificultad de aprendizaje de estas tecnologías, ha hecho que no nos sea posible desarrollar todas las fases de limpieza de datos del proyecto. Es cierto que hay datos redundantes y sobrantes que ante una escalabilidad de estos podría generar problemas de almacenamiento y memoria.
- **Mejora de Scripts:** Se podrían mejorar los scripts que se utilizan parametrizándolos, así como generar Scripts parametrizados para la preparación del Data Lake de cara a futuras empresas que entren en el sistema. La idea sería generar Scripts sencillos a los que pasándoles como parámetro el símbolo bursátil o abreviatura de la empresa, este prepare todo lo necesario en el entorno para funcionar y así reducir el impacto de la entrada de nuevas empresas.
- **Securización del entorno:** No ha sido en ningún momento el objetivo de este proyecto, pero es cierto que cualquier solución que use estos entornos Cloudera debe ser securizada. Esto podría llevarse a cabo de una manera no muy complicada a través de Kerberos (protocolo de autenticación de redes de ordenador creado por el MIT que permite a dos ordenadores en una red insegura demostrar su identidad mutuamente de manera segura) cuya mayor ventaja en su integración con Cloudera es la compatibilidad y transparencia ante el escenario de que el cluster esté formado por una mayor cantidad de máquinas.
- **Migración a la nube:** Como se ha dicho en varias ocasiones, durante estos meses de desarrollo de este proyecto, el escenario del Big Data se ha modificado notablemente. En concreto las nuevas soluciones de *Microsoft Azure* para el Big Data están en auge y es una realidad que los proyectos Big Data que van entrando en la empresa llevan a esta nube de Microsoft como requisito.

## 6.3. Aprendizaje

Una vez finalizado el desarrollo de este proyecto he adquirido una gran cantidad de conocimientos que serán de gran ayuda en mi futuro laboral.

En primer lugar el utilización de herramientas Big Data antes del desarrollo de este proyecto era para mi un sueño prácticamente inalcanzable, en gran parte por su complejidad y por la inexistente formación adquirida en estas tecnologías. Por ello considero que la evolución en el aprendizaje de estas tecnologías ha sido muy positiva. Más allá de que mi situación en la empresa con la que realizo este proyecto haya sido de gran ayuda.

De manera externa al ámbito tecnológico este proyecto en conjunto con mi primera experiencia labora en Big Data me ha permitido conocer cómo nace un proyecto real, gracias a mis tutores y a su dedicación extra hacia mí. Además de esto me ha supuesto un gran desarrollo en mis capacidades de comunicación, así como un cambio en mi manera de pensar sobre cómo se desarrollan los proyectos informáticos.

Este proyecto además me ha permitido conocer una metodología de trabajo independiente. Esto se debe a la gran libertad existente a la hora de realizar un TFG, en la que los plazos te los marcas tú, a diferencia de las asignaturas desarrolladas a lo largo de la carrera, en la que te debes ajustar a unos criterios y plazos determinados con anterioridad.

Además de los anteriores aprendizajes, el desarrollo de este proyecto me ha permitido mejorar mi capacidad crítica, a la hora de investigar y descartar información. Este proyecto me ha supuesto un aprendizaje continuo, ya que además de todas las tecnologías y lenguajes no vistos con anterioridad, me ha permitido ampliar mis conocimientos sobre Data Lakes, además de conocer en profundidad el mercado de valores y los diferentes agentes que lo componen.

Además en gran cantidad de ocasiones me ha sido necesario aplicar los diferentes conocimientos adquiridos en asignaturas de la carrera para la elaboración de este proyecto:

- Aplicación de recolección: Para el uso de la aplicación recolectora de datos ha sido de gran valor los conocimientos sobre el lenguaje de programación Java y la programación multihilo adquiridos en Programación Orientada a Objetos y en Sistemas Distribuidos.
- Dashboard: Para el desarrollo del dashboard web de este proyecto ha sido importante tener los conocimientos adquiridos en las asignaturas que te enseñan a utilizar estructuras de datos así como de desarrollo web e Interacción Persona-Computadora.
- Aplicación de cierre de mercado: Nuevamente todas las asignaturas de programación y paradigmas. Estos conocimientos me han permitido conocer lo que deseo desarrollar y por tanto únicamente debería adaptarme al lenguaje con el que se desarrollará la aplicación.
- Arquitectura Big Data: Para el diseño y creación del Data Lake me han sido de gran utilidad los conocimientos adquiridos en las asignaturas de Bases de Datos, facilitándome también el desarrollo del modelo conceptual.

- **Instalación del entorno:** Para la instalación de los componentes necesarios para desarrollar la aplicación web y su conexión con Cloudera, han sido de gran utilidad los conocimientos adquiridos en las asignaturas de Sistemas Operativos, (sobre todo por interactuar con un sistema Unix) y de redes.
- **Documentación:** Para llevar a cabo la documentación de este proyecto, así como la gestión del mismo, han sido de gran utilidad los conocimientos adquiridos en asignaturas como Planificación Y Gestión De Plataformas Informáticas, las prácticas de Evaluación De Sistemas Informáticos y de Diseño, Integración Y Adaptación De Software con José Manuel Marqués.



## Apéndice A

# Contenidos del CD-ROM

Se entrega un CD-ROM duplicado que contiene los programas y scripts de creación y limpieza de datos, así como el código fuente de la aplicación desarrollada y una copia de esta memoria.

El contenido del CD es el que se detalla a continuación y sigue esta estructura de directorios:

- La versión en PDF del documento completo, en un archivo único con el nombre “memoria.pdf”.
- La aplicación de Power BI que contiene el Dashboard desarrollado con el nombre “StocksDashboard.pbix”.
- Directorio “ReadStocksStreaming”: Carpeta que contiene todo el software desarrollado en versión fuente.
- Directorio “scripts”: Carpeta que contiene los scripts utilizados.



# Referencias Web

- [1] La realidad del perfil de informático junior en España según los informes  
Revisado a 10 de Julio de 2019
- [2] Desarrollador en 'big data', el perfil más demandado en 2018  
Revisado a 10 de Julio de 2019
- [3] Se dispara la demanda de profesionales de Big Data  
Revisado a 10 de Julio de 2019
- [4] Analista de datos, el puesto más demandado (y mejor pagado) por las empresas  
Revisado a 10 de Julio de 2019
- [5] IEX Cloud - API Reference  
Revisado a 10 de Julio de 2019
- [6] ARQUITECTURA LAMBDA PARA SISTEMAS BIG DATA  
Revisado a 10 de Julio de 2019
- [7] Cloudera y Hortonworks se fusionan  
Revisado a 10 de Julio de 2019
- [8] Cloudera announces CEO departure, stock plunges  
Revisado a 10 de Julio de 2019
- [9] Cloudera, MapR, Hortonworks. . . ¿Qué distribución Hadoop necesitas?  
Revisado a 10 de Julio de 2019
- [10] Así es el perfil de Big Data más buscado por las empresas  
Revisado a 10 de Julio de 2019
- [11] Scala vs. Python for Apache Spark  
Revisado a 10 de Julio de 2019
- [12] Qué es la guía PMBOK y cómo influye en la administración de proyectos  
Revisado a 10 de Julio de 2019
- [13] Analista de datos, el puesto más demandado (y mejor pagado) por las empresas  
Revisado a 10 de Julio de 2019

- [14] OVH Servers  
Revisado a 10 de Julio de 2019
- [15] Cloudera Manager Extensions  
Revisado a 10 de Julio de 2019
- [16] Java 1.8 Downloads  
Revisado a 10 de Julio de 2019
- [17] Hortonworks Community Connection: How to use puthivestreaming  
Revisado a 10 de Julio de 2019
- [18] 3 Massive Big Data Problems Everyone Should Know About  
Revisado a 10 de Julio de 2019
- [19] ¿Qué es un Data Lake?  
Revisado a 10 de Julio de 2019
- [20] Crece el almacenamiento de datos no estructurados  
Revisado a 10 de Julio de 2019
- [21] Data Lake development with Big Data : explore architectural approaches to building Data Lakes that ingest, index, manage, and analyze massive amounts of data using Big Data technologies  
Revisado a 10 de Julio de 2019