UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA COMPUTACIÓN
CENTRO DE COMPUTACIÓN PARALELA Y DISTRIBUIDA

Desarrollo de módulos de procesamiento de datos y generación de gráficos de una herramienta web a partir de una vista minable alojada en un clúster Hadoop.

Trabajo Especial de Grado presentado ante la ilustre

Universidad Central de Venezuela por

Br. David Alejandro Padrino González

C.I. 20.328.774.

Br. Miguel Alejandro Figueira De Freitas

C.I. 20.309.037.

Tutores: Jesús Lares y Haydemar Núñez.

Caracas, Mayo 2017.

Acta

Acta

Quienes suscriben, miembros del Jurado designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por los bachilleres Miguel Alejandro Figueira de Freitas, portador de la Cédula de Identidad V-20.309.037 y David Alejandro Padrino González, portador de la Cédula de Identidad V-20.328.774, con el título: **"Desarrollo de módulos de procesamiento de datos y generación de gráficos de una herramienta web a partir de una vista minable alojada en un clúster Hadoop"**, a los fines de optar al título de Licenciados en Computación, dejan constancia de lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del jurado, se fijó el día 19 de Mayo de 2017, para que sus autores lo defiendan en forma pública, lo que hizo en la Sala 1 de la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondieron las preguntas formuladas.
Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo con la nota de 20 puntos.

En fe de lo cual se levanta la presente Acta, en Caracas a los diecinueve (19) días del mes de Mayo del año dos mil diecisiete (2017), dejando constancia de que actuó como Coordinador del Jurado el profesor Jesús Lares.

Prof. Jesús Lares - Tutor

Prof. Haydemar Núñez - Tutor

Prof. Mercy Ospina - Jurado

Prof. Héctor Navarro - Jurado

Dedicatoria

Dedicado a Carmen González, Lilia González y José Padrino, mis estrellas que brillan en el cielo...

- David Padrino

Dedicado a mis padres Cecilia Figueira y Francisco Figueira que son el pilar fundamental de mi vida.

- Miguel Figueira

Agradecimientos

Los resultados de este trabajo están dedicados a todos aquellos que, de alguna forma, son parte de su culminación, los cuales agradecemos a continuación.

David Padrino:

A la directiva Wikot Technologies por permitirnos asistir a reuniones de tesis en determinados momentos, por apoyarnos en nuestro crecimiento profesional y particularmente gracias a Camilo Iturra por creer en mí, por permitirnos, como presidente de Wikot, utilizar las instalaciones de la empresa para la realización de este proyecto tan ambicioso.

A mis tutores Prof. Jesús Lares y Prof. Haydemar Núñez por hacer de este trabajo, uno de características especiales, por ser un trabajo diferente y ambicioso que da pie para ser un gran y reconocido proyecto; por su paciencia, su colaboración, su aceptación como tutores y sus conocimientos transmitidos que se ven plasmados en los resultados obtenidos.

A Lourdes González, Nelson Padrino D., Nelson Padrino G., Ramona Malavé, Victor Vargas, Carlos Plaza, Sabrina Yépez, Ana Mejías, José Espinoza y demás familiares y amigos por apoyarme desde el inicio de mis estudios y mantenerme en constante motivación e interés por el progreso y hacer el bien desde la ciencia y la tecnología, por estar ahí cuando más los necesitaba y por el empuje emocional que se requiere para una carrera universitaria.

A mi casa de estudios, la Ilustre "Universidad Central de Venezuela" por permitir mi formación en sus instalaciones y ser testigo de mi crecimiento profesional.

A todos, Gracias!

Miguel Figueira:

A Cecilia Figueira , Francisco Figueira , Jessika Figueira , Alexander Figueira , Jose L De Freitas , Bryan Aleixo , Christian Betancourt y demás familiares y amigos por apoyarme incondicionalmente durante mi vida personal y profesional.

A mis tutores Prof. Jesús Lares y Prof. Haydemar Núñez por hacer de este

trabajo especial de grado , uno de características especiales, por ser un proyecto ambicioso que da pie para convertirse en una herramienta importante para la facultad de Ciencias; por su paciencia, su colaboración, su aceptación como tutores y sus conocimientos transmitidos que se ven plasmados en los resultados obtenidos.

A Wikot Technologies por permitirnos asistir a las reuniones de tesis, además de la posibilidad de usar sus instalaciones para la realización del proyecto.

A mi casa de estudios, la Ilustre "Universidad Central de Venezuela" y a todos los profesores que conocí a lo largo de mi vida universitaria que realizan una gran labor transmitiendo sus conocimientos a pesar de las dificultades.

Resumen

Título: Desarrollo de módulos de procesamiento de datos y generación de gráficos de una herramienta web a partir de una vista minable alojada en un clúster Hadoop.

Autores: David Padrino y Miguel Figueira.
Tutores: Prof. Jesús Lares y Haydemar Núñez.

En la actualidad las herramientas para el análisis de grandes volúmenes de datos se limita al uso de aplicaciones licenciadas y costosas, sin tener en consideración las prácticas académicas a través de las cuales los futuros profesionales pueden conocer el flujo de trabajo desde la preparación de datos crudos hasta los gráficos generados producto de aplicar algoritmos a los datos "limpios"basándose en el proceso KDD (Knowlegde Discovery in Databases, Descubrimiento de conocimiento en bases de datos, punto que sera tratado en la seccion 3.11.9) de minería de datos.

El objetivo del presente trabajo es resolver las carencias de ese conjunto de tecnologías referenciadas anteriormente, a través de una herramienta basada en una interfaz web que permita trabajar conjuntos de datos desde su preparación hasta la visualización de resultados que pueden ser interpretados para la generación de conocimiento.

Las herramientas usadas en conjunto para realizar este proyecto fueron la suite de Hadoop, Apache Spark, el framework web Meteor.js, jQueryUI para la realización de una interfaz usable, un servidor TCP/IP de R, Apache Drill y paquetes de Node.js.

Palabras clave: Big Data, Apache Spark, Meteor.js, jQueryUI, HDFS, Apache Drill, RStudio, KDD, Hadoop.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

Desde el surgimiento del fenómeno internet la velocidad de generación de datos, al igual que el número de fuentes que los ha ido creando han ido aumentando drásticamente con el pasar de los años debido al incremento de tecnologías y dispositivos que permiten cada vez a mayor cantidad de personas tener acceso, generar y compartir información a través de internet. Aunado a lo anteriormente expuesto, la mayor cantidad de esta información no tiene una estructura rígida por lo que han nacido nuevas tecnologías y otras han sido adaptadas a la nueva estructura de los datos, de manera que su carga, almacenamiento, procesamiento, visualización y análisis sea en un tiempo menor a pesar de que la cantidad de datos de la que se está hablando es enorme y diversa.

La ciencia de datos (conocido popularmente como Big Data) nace como solución a esta problemática y como el siguiente escalón de evolución de la inteligencia de negocios. Big data se refiere a las tecnologías e iniciativas que involucran datos que presentan un alto grado de diversidad y crecen en forma exponencial, como para ser procesados con tecnologías convencionales, además de las habilidades e infraestructura para hacerle frente de manera eficiente. Dicho de otra manera, el volumen, la velocidad en que se generan o la variedad de datos es demasiado grande. Es decir, Big Data describe conjuntos de datos tan grandes y complejos que son difíciles de manipular con las herramientas de software tradicionales.

Además, para agregar valor a Big data, existen tecnologías web y de almacenamiento de datos que permiten el manejo de grandes volúmenes de manera eficiente en tiempo, tales como bases de datos NoSQL y el framework web basado en JavaScript: Meteor, que en conjunto con elementos del ambiente Hadoop y bibliotecas de visualización de gráficos permiten la generación de información producto de datos, para posteriormente convertirlo en conocimiento bien sea por el análisis de resultados en tablas o gráficos.

El presente trabajo tiene una estructura bien definida separando el problema de investigación, donde se hace el planteamiento y justificación del mismo, objetivos de investigación y alcance de investigación, el marco teórico, en donde se explican las

tecnologías y herramientas investigadas para una posterior selección enmarcada en el desarrollo del proyecto, el marco metodológico, en donde se habla sobre la metodología de desarrollo XP en la cual se basó el equipo de desarrollo para aplicar una metodología Ad Hoc, el marco aplicativo, donde se hace referencia a cada una de las fases de la metodología XP que se utilizaron para la planificación y desarrollo del proyecto. Finalmente se concluye con contribuciones del trabajo, recomendaciones y trabajos futuros, previo a las referencias bibliográficas.

Al final del documento se presenta un apéndice que sirve como manual para extender el desarrollo del presente trabajo de grado.

CAPÍTULO 2

Problema de investigación

En el presente capítulo se detalla el problema de investigación, su justificación, además se realiza el planteamiento de los objetivos generales y especificos en conjunto con la definición del alcance del proyecto.

2.1. Planteamiento del problema

En la actualidad existen herramientas para el manejo de grandes volúmenes de datos, los cuales pueden venir de diferentes fuentes, en diferentes formatos y con una estructura distinta para cada una, por lo cual se requieren herramientas de extracción, limpieza y transformación de datos para luego realizar un procesamiento para generar una vista minable y a través de algoritmos de minería de datos, generar modelos que luego se podrán interpretar y evaluar a través de gráficos.

En su mayoría, dichas herramientas son privativas, requieren de una licencia para poder ser utilizadas y no estan orientadas a la formación y al ámbito académico. Aunado a lo anteriormente mencionado, dichas herramientas no permiten al usuario ejecutar el flujo completo de un proceso KDD (Knowlegde Discovery in Databases, Descubrimiento de conocimiento en bases de datos, punto que sera tratado en la seccion 3.11.9) directamente sobre una misma interfaz.

Para realizar el flujo completo del proceso KDD, se requiere interconectar un conjunto de herramientas y manejar distintos formatos de archivo, que en muchas ocasiones pueden ocasionar problemas de compatibilidad causando que el proceso se vea ralentizado o interrumpido.

Existe la necesidad de una herramienta integral y no privativa que se utilice a nivel académico y que permita a los estudiantes la realización del flujo ininterrumpido de un proceso de mineria sobre un conjunto de datos de gran volumen desde pre-procesamiento de los datos hasta la visualización de resultados.

En la Universidad Central de Venezuela, específicamente en la Facultad de Ciencias, Escuela de Computación se desea desarrollar una herramienta web personali-

zada que permitan integrar en una sola interfaz varias funcionalidades sin necesidad de estar explícitamente conectándose a otros servicios que podrían automatizarse en segundo plano sin que el usuario tenga que realizarlas.

Expuesto lo anterior surge la duda, ¿Cómo desarrollar una herramienta web que cumpla con las necesidades anteriormente expuestas, desarrollada con software libre y que facilite el proceso KDD de un científico de datos?.

2.2. Justificación

El desarrollo de este proyecto parte de dar solución a las dificultades y carencias explicadas anteriormente, donde el proceso KDD transcurre entre los módulos de procesamiento de grandes volúmenes de datos y visualizacion de resultados de una herramienta web, además de facilitar el proceso de formación de los estudiantes de la escuela de Computación que se forman en el área de ciencia de datos, donde los dichos estudiantes puedan acceder a diferentes conjuntos de datos alojados en un clúster Hadoop, puedan aplicar diferentes algoritmos de minería de datos, generar gráficos que permitan la generación de conocimiento a través de la evaluación e interpretación de resultados.

Este proyecto busca interconectar un número de herramientas que resuelven determinados problemas por separado y se pretende lograr la resolución de problemas de mayor envergadura logrando además facilitar que la comunidad universitaria tenga acceso a herramientas de libre uso en el área de ciencia de datos.

Con este tipo de herramientas se busca motivar al alumnado a participar en la construcción de otras herramientas aún más complejas y al uso de tecnologías no tan populares en la actualidad pero que en conjunto con otro tipo de tecnologías, permitan la generación de cada vez más herramientas que colaboren en la resolución de problemas complejos que tengan que ver con grandes volúmenes de datos y a su vez, unificar las tecnologías para trabajar ciencia de datos con interfaces usables de modo que se puedan realizar tareas complejas de manera más eficaz, con mayor accesibilidad y en menor tiempo que otras herramientas existentes.

Además, con una interfaz web se permite un fácil acceso al sistema y deja del lado del servidor toda la complejidad que conlleva el procesamiento y almacenamiento paralelo y distribuido de los datos.

Actualmente, el número de organizaciones que buscan generar conocimiento y dinero a través de la explotación de sus grandes volúmenes de datos va en aumento por lo que el momento es el indicado para ser pioneros en la construcción de herramientas integrales que resuelvan en poco tiempo problemas que actualmente son costosos en tiempo y dinero.

2.2.1. ¿Por qué es un problema?

Es un problema porque no se cuenta con una herramienta de uso libre que permita mostrar el flujo de trabajo del proceso KDD.

2.2.2. ¿Para quién es un problema?

Es un posible problema para los estudiantes de computación que desean complementar su formación en el área de ciencia de datos de la Escuela de Computación en la Universidad Central de Venezuela.

2.2.3. ¿Desde cuándo es un problema?

Es un problema desde que se imparten estudios sobre Ciencia de datos en la Escuela de Computación en la Universidad Central de Venezuela.

2.3. Objetivos de la investigación

Los módulos a desarrollar son **Modelado de vista minable** (vista minable se refiere a un conjunto de datos que ha sido preparado para su procesamiento) y **Visualización de resultados**; éstos forman parte de una herramienta que procura integrar todos los pasos de un proceso de minería de datos.

Dichos modulos hacen referencia a los últimos tres pasos del proceso KDD.

2.3.1. Objetivo General

Implementar módulos de una herramienta web, cuya entrada sea una vista minable, basados en los tres últimos pasos del proceso KDD los cuales permiten la aplicación de algoritmos de minería de datos a dicha vista minable y el despliegue de gráficos que permitan generación de conocimiento a través de análisis e interpretación de resultados.

2.3.2. Específicos

1. Seleccionar una metodología de desarrollo que cumpla con las necesidades de la herramienta.

2. Implementar interfaces de usuario usables e intuitivas para los diferentes módulos de la herramienta.

3. Implementar una funcionalidad que permita realizar lecturas de una vista minable desde un clúster Hadoop.

4. Implementar módulo que permita la aplicación de algoritmos de minería de datos a la vista minable.

5. Implementar módulo de visualización para la generación de gráficos una vez aplicados los algoritmos a la vista minable

6. Realizar pruebas de integración para cada una de las iteraciones.

2.4. Antecedentes

Hadoop resuelve el problema de grandes volúmenes de datos proporcionando un marco de trabajo (framework) que permite realizar procesamientos en paralelo con tolerancia a fallos, utilizando clusters. Algunas de sus funciones principales son el almacenamiento distribuido utilizando HDFS y el procesamiento distribuido mediante MapReduce. MapReduce es un paradigma limitado ya que posee problemas con muchos algoritmos iterativos (por ejemplo, PageRank). Para resolver el problema anteriormente mencionado, Apache Hadoop incorporó la herramienta de procesamiento en memoria, Spark.

Apache Spark posee un paquete llamado MlLib que posee un conjunto de algoritmos de *Machine Learning* y una biblioteca para R llamada *SparklyR* (Disponible desde Spark v1.5) que permite hacer llamadas de algoritmos de MlLib a través de R.

Meteor.js como framework de desarrollo web basado en Node.js tiene una gran facilidad para instalar paquetes NPM, integración nativa con MongoDB y paradigma de programación orientado a componentes o módulos, lo cual permite la construcción de una aplicación altamente escalable y con facilidad de integración con diversos elementos que puedan adaptarse y mejorar funcionalidades constantemente.

RIO (R Input/Output), conecta una aplicación a Rserve, el cual es un servidor TCP/IP que permite a otros programas a utilizar las facilidades de R. Soporta cualquier tipo de datos incluyendo, enteros, arreglos, caracteres, cadenas de caracteres, lógicos, incluso imágenes y archivos. Además soporta texto plano y encriptación de contraseñas para autenticación, gracias a las configuraciones realizadas en Rserve.

Para este proyecto se toma como referencia software existentes de grandes organizaciones como Oracle y Microsoft con sus productos Big Data Discovery y Microsoft Azure respectivamente permitiéndole al usuario final tener una abstracción del manejo de grandes volúmenes de datos que ofrecen independencia de las prestaciones de la máquina que se utilizaría para la ejecución, dado que vuelca todo el procesamiento y almacenamiento en un clúster de forma paralela y distribuida.
Azure ofrece un espacio de trabajo con nodos que se interconectan para generar el flujo de trabajo del proceso KDD de forma completa.
Por su parte Oracle Big Data Discovery ofrece una separacion en el proceso de limpieza y transformacion, del proceso de exploración de datos.

2.5. Alcance

Se plantea que los módulos a desarrollar en el proyecto abarquen las últimas tres etapas del proceso KDD, que son la aplicación de algoritmos de minería de datos y la generación de gráficos que facilite la interpretación y evaluación de resultados, producto de la aplicación de dichos algoritmos.

La aplicación web a desarrollar partirá de leer vistas minable de un clúster Hadoop , alojados en el sistema de archivos distribuidos Hadoop(HDFS).

La herramienta web permitirá la aplicación de algunos de los más importantes algoritmos de minería de datos y generación gráficos que permitirán un análisis de resultados.

CAPÍTULO 3

Marco teórico

En el presente capítulo se exponen los conceptos de investigación previos a la realización del proyecto.

3.1. Dato

Los datos son números, letras o símbolos que describen objetos, condiciones o situaciones. Son el conjunto básico de hechos referentes a una persona, cosa o transacción de interés para distintos objetivos, entre los cuales se encuentra la toma de decisiones. [1]

3.2. Información

La información es un sistema de control, en tanto que es la propagación de consignas que deberíamos de creer. En tal sentido la información es un conjunto organizado de datos capaz de cambiar el estado de conocimiento. Un grupo de datos ordenados y supervisados, que sirven para construir un mensaje basado en un cierto fenómeno o ente, la cual permite resolver problemas y tomar decisiones, es información, ya que su aprovechamiento racional es la base del conocimiento. [2]

3.3. Conocimiento

Fidias Arias (2004), define el conocimiento como un "proceso en el cual se relacionan el sujeto que conoce, que percibe mediante sus sentidos, y el objeto conocido y percibido". El conocimiento es el acto o efecto de conocer. Es la capacidad del hombre para comprender por medio de la razón la naturaleza, cualidades y relaciones de las cosas.
La ciencia considera que, para alcanzar el conocimiento, es necesario seguir un método. El conocimiento científico no sólo debe ser válido y consistente desde el punto

de vista lógico, sino que también debe ser probado mediante el método científico o experimental. [3]

3.4. Ciencia de datos (Data Science)

3.4.1. Definición:

La Ciencia de datos [4] es un campo interdisciplinario que involucra los procesos y sistemas para extraer conocimiento o un mejor entendimiento de grandes volúmenes de datos en sus diferentes formas (estructurado, semi estructurado y no estructurado) a través del uso de métodos automatizados que se encargan de limpiar, procesar, analizar y mostrar resultados que colaboren en la toma de decisiones. La ciencia de datos determina un nuevo enfoque que permite realizar diferentes descubrimientos a través de la combinación de distintos aspectos de estadística, matemáticas aplicadas, ciencia de la computación y técnicas de visualización, convirtiendo así vastos volúmenes de datos en nuevas ideas y conocimiento.
Algunas características de la ciencia de datos:

- Debe involucrar conocimientos de uno o más dominios (por ejemplo finanzas, medicina o geología).
- Debe tomar en cuenta aspectos computacionales.
- Debe incluir técnicas científicas tales como la prueba de hipótesis y la validación de resultados.
- Los resultados deben ser confiables.
- Debería involucrar más matemáticas y estadísticas que los enfoques anteriores.
- Debería incluir el aprendizaje automatizado (machine learning), inteligencia artificial o algoritmos de descubrimiento de conocimiento (knowledge discovery).
- Debería implicar la visualización y creación rápida de prototipos para el desarrollo de software.
- Debe satisfacer al menos uno de estos deberes en un nivel perturbador.

3.4.2. Grandes Volúmenes de Información (Big Data):

Es un término que describe un gran volumen de datos tanto estructurados como no estructurados que se generan en un negocio cada determinada cantidad de tiempo, sea diariamente, anualmente, entre otros. Big Data no sólo hace referencia a la cantidad de información, sino a lo que las organizaciones hacen con esta información para generar conocimiento y así tomar mejores decisiones y, de ser necesario, cambiar las estrategias del negocio convirtiéndose en la evolución de la inteligencia de

negocios, agregando el proceso de captura, almacenamiento, procesamiento, análisis y visualización de grandes cantidades de datos y a los procedimientos usados para encontrar patrones repetitivos dentro de esos datos.[5]
Algunas características de big data:

- Debe involucrar cómputo distribuido en múltiples servidores.
- Debe entremezclar gestión y procesamiento de datos.
- Debe ir más allá de las bases de datos relacionales y data
- Debe permitir resultados que no estaban disponibles con los enfoques anteriores, o que llevarían sustancialmente mucho más tiempo (tiempo de ejecución o latencia).

El término big data va ligado generalmente a 5 palabras que la caracterizan, todas estas palabras comienzan con la letra "V" por lo cual se habla de las 5 V's de Big Data:

- **Volumen:** Refiriéndose a la gran cantidad de datos generada cada segundo, la cual supera los Terabytes incluso alcanzando y sobrepasando los Zettabytes.
- **Variedad:** Se refiere a los distintos tipos de datos que se pueden utilizar. En el 2016 el 80 % aproximadamente de los datos está no estructurado (video, imágenes, voz, etc).
- **Velocidad:** Hace referencia a la velocidad en la que los datos son generados y a la velocidad en que estos datos se mueven. Por ejemplo, información en los medios sociales que se hace viral en cuestión de minutos, incluso segundos.
- **Veracidad:** Se refiere al grado de veracidad de los datos. El caso más popular son los medios sociales en los cuales se habla coloquialmente, analizar esta data es posible a través de ciertas técnicas relacionada con el texto.
- **Valor:** No tendría sentido realizar un análisis exhaustivo de big data sobre ciertos datos si no obtendrá un valor a nivel de conocimiento que colabore con el negocio.

Se podría agregar una sexta V que a pesar de no estar popularizada dentro del área, tiene un valor muy importante a nivel de alta gerencia como lo es la **Visualización** del dato. Dado un análisis, se requeriría una forma gráfica de ver todos los resultados para poder realizar comparaciones y así generar conocimiento proveniente del conjunto de datos.

Sin embargo, se habla de una séptima V: **Variabilidad**, que hace referencia al cambio el significado de los datos ya que para algunos elementos, por ejemplo las palabras, ya no significan lo mismo en todas partes del mundo, ni en todos los idiomas.

3.4.3. Campos en los cuales es utilizado

El manejo de grandes volúmenes de datos es utilizado y se puede utilizar en múltiples áreas, por ejemplo:

- Seguridad: Su potencial reside en la capacidad de análisis de volúmenes de datos antes impensable de una manera óptima y ágil. Existen, por ejemplo, modelos de análisis del comportamiento humano para prevenir atentados terroristas obtenidos mediante un análisis permanente de las cámaras, sensores y accesos secuenciales a un sistema.

- Investigación médica: La investigación médica puede mejorar muchísimo si es capaz de asimilar una enorme cantidad de datos (monitorización, historiales, tratamientos, etc.) y estructurarlos para el establecimiento de diagnósticos o la síntesis de medicamentos.

- Gobierno y toma de decisiones: Big Data ofrece una mejora y optimización en los procesos de análisis de empresas y gobiernos, permitiendo entre muchas otras, el soporte a la toma decisiones, siendo complementario a las plataformas de "Business Intelligence"(BI).

- Internet 2.0: genera una gran multitud de datos que difícilmente se podrían gestionar sin un Big Data. Las redes sociales cada vez se extienden a más ámbitos de nuestra sociedad

- Logística: El sector logístico mejora notablemente gracias a las posibilidades analíticas de un Big Data y su potencial para el despliegue de servicios específicos (movilidad, tracking, seguridad, etc.). El ejemplo más popular se encuentra en el control de flotas (la ruta óptima permite a los vehículos circular con la máxima capacidad de carga, pudiendo recorrer rutas mejorando tiempos, consumos y contaminación).

3.5. Minería de datos (Data Mining)

Es un campo de la estadística y las ciencias de la computación que tiene como objetivo detectar la información procesable o patrones sobre conjuntos grandes de datos. Normalmente, estos patrones no se pueden detectar mediante la exploración tradicional de los datos porque las relaciones son demasiado complejas o porque hay demasiados datos.[6]

La generación de un modelo de minería de datos forma parte de un proceso mayor que incluye desde la formulación de preguntas acerca de los datos y la creación de un modelo para responderlas, hasta la implementación del modelo en un entorno de trabajo. Este proceso se puede ejecutar siguiendo distintas metodologías de trabajo, siendo una de las mas conocidas CRISP-DM, explicada con mas detalle en la sección 3.11.10.

3.6. Aprendizaje automático (Machine Learning)

En ciencias de la computación el aprendizaje automático es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. El aprendizaje automático se divide en dos áreas principales: aprendizaje supervisado y aprendizaje no supervisado.[7]

El objetivo del aprendizaje supervisado es hacer predicciones a futuro basadas en comportamientos o caracterısticas que se han visto en los datos ya almacenados. El aprendizaje supervisado permite buscar patrones en datos históricos etiquetados. Por otro lado, el aprendizaje no supervisado usa datos históricos que no están etiquetados. El fin es explorarlos para encontrar alguna estructura o forma de organizarlos.

3.7. Inteligencia artificial (Artificial Intelligence)

Inteligencia artificial es considerada una rama de la computación que relaciona un fenómeno natural con una analogía artificial mediante programas de computador. Consiste en el diseño de procesos que, al ejecutarse sobre una arquitectura física, producen resultados que maximizan una cierta medida de rendimiento denominado comúnmente comportamiento inteligente.[8]

3.8. Inteligencia de negocios (Business Intelligence)

Inteligencia de negocios es el conjunto de conceptos y métodos para mejorar la toma de decisiones en los negocios, utilizando sistemas de apoyo basados en hechos (Howard Dresner, 1989). Sin embargo, en la actualidad este concepto incluye una amplia categoría de metodologías, aplicaciones y tecnologías que permiten reunir, acceder, transformar y analizar los datos con el propósito de ayudar a los usuarios de una organización a tomar mejores decisiones de negocio .[9]

3.9. Base de datos

Se define una base de datos como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular. [10]
Entre las principales características de los sistemas de base de datos podemos mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

3.9.1. Base de datos relacionales

Una base de datos relacional es una colección de elementos de datos organizados en un conjunto de tablas formalmente descritas desde la que se puede acceder a los datos o volver a montarlos de muchas maneras diferentes sin tener que reorganizar las tablas de la base.[11]

Estas bases de datos poseen un conjunto de tablas que contienen datos provistos en categorías predefinidas. Cada tabla (que a veces se llaman 'relación') contiene una o más categorías de datos en columnas. Cada fila contiene una instancia única de datos para las categorías definidas por las columnas.

3.9.1.1. Ventajas de las base de datos relacionales

- Está más adaptado su uso y los perfiles que los conocen son mayoritarios y más baratos.
- Debido al largo tiempo que llevan en el mercado, estas herramientas tienen un mayor soporte y mejores suites de productos para gestionar estas bases de datos.
- La atomicidad de las operaciones en la base de datos.
- Los datos deben cumplir requisitos de integridad tanto en tipo de dato como en compatibilidad.

3.9.1.2. Desventajas de las base de datos relacionales

- La atomicidad de las operaciones juegan un papel crucial en el rendimiento de las bases de datos.
- Escalabilidad, que aunque probada en muchos entornos productivos suele, por norma, ser inferior a las bases de datos NoSQL.

3.9.2. Base de datos no relacionales

Son un enfoque hacia la gestión de datos y el diseño de base de datos que es útil para grandes conjuntos de datos distribuidos. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad), y habitualmente escalan bien horizontalmente.[12]

Son especialmente útiles cuando una empresa necesita acceder y analizar grandes cantidades de datos no estructurados o datos que se almacenan de forma remota en varios servidores virtuales en la nube.

3.9.2.1. Ventajas de las base de datos no relacionales

- La escalabilidad y su carácter descentralizado. Soportan estructuras distribuidas.
- Suelen ser bases de datos mucho más abiertos y flexibles. Permiten adaptarse a necesidades de proyectos más fácilmente que los modelos de Entidad Relación.
- Se pueden hacer cambios de los esquemas sin tener que detener las bases de datos.
- Escalabilidad horizontal: son capaces de crecer en número de máquinas, en lugar de tener que residir en grandes máquinas.
- Se pueden ejecutar en máquinas con pocos recursos.
- Optimización de consultas en base de datos para grandes cantidades de datos.

3.9.2.2. Desventajas de las base de datos no relacionales

- No todas las bases de datos NoSQL contemplan la atomicidad de las instrucciones y la integridad de los datos. Soportan lo que se llama consistencia eventual.
- Problemas de compatibilidad entre instrucciones SQL. Las nuevas bases de datos utilizan sus propias características en el lenguaje de consulta y no son 100 % compatibles con el SQL de las bases de datos relacionales. El soporte a problemas con las queries de trabajo en una base de datos NoSQL es más complicado.

- Falta de estandarización. Existen muchas bases de datos NoSQL y aún no hay un estándar como lo hay en las bases de datos relacionales. Se presume un futuro incierto en estas bases de datos.
- Soporte multiplataforma. Aún quedan muchas mejoras en algunos sistemas para que soporten sistemas operativos que no sean Linux.
- Suelen tener herramientas de administración no muy usables o se accede por consola.

3.9.2.3. Tipos de base de datos no relacionales

Existe una amplia gamma de sistemas manejadores de bases de datos no relacionales a través de sus diferentes familias, en el siguiente cuadro 3.1 se resumen las características y aplicaciones de algunas de estas familias.

MODELO DE DATOS	FORMATO	CARACTERÍSTICAS	APLICACIONES
Documento.	Similar a JSON (JavaScript Object Notation).	- Intuitivo. - Manera natural de modelar datos cercana a la programación orientada a objetos. - Flexibles, con esquemas dinámicos. - Reducen la complejidad de acceso a los datos.	Se pueden utilizar en diferentes tipos de aplicaciones debido a la flexibilidad que ofrecen.
Grafo.	Nodos con propiedades (atributos) y relaciones (aristas).	- Los datos se modelan como un conjunto de relaciones entre elementos específicos. - Flexibles, atributos y longitud de registros variables. - Permite consultas más amplias y jerárquicas.	Redes sociales, software de recomendación, geolocalización, topologías de red, etc.
Clave-Valor y Columna.	Clave-valor: una clave y su valor correspondiente Columnas: variante que permite más de un valor (columna) por clave.	- Rendimiento muy alto. - Alta curva de escalabilidad. - Útil para representar datos no estructurados.	Aplicaciones que solo utilizan consulta de datos por un solo valor de la clave.

Tabla 3.1: Tipos de Base de Datos NoSql.

3.9.2.3.1. Orientada a columnas

Este tipo de bases de datos están pensadas para realizar consultas y agregaciones sobre grandes cantidades de datos. Funcionan de forma parecida a las bases de datos relacionales, pero almacenando columnas de datos en lugar de registros. Algunos ejemplos de base de datos orientada a columnas: Cassandra, HBase.

- Cassandra: incluida en esta sección, aunque en realidad sigue un modelo hí-

brido entre orientada a columnas y clave-valor. Es utilizada por Facebook y Twitter (aunque dejaron de usarla para almacenar tweets).

- HBase. Escrita en Java y mantenida por el Projecto Hadoop de Apache, se utiliza para procesar grandes cantidades de datos. La utilizan Facebook, Twitter o Yahoo.

3.9.2.3.2. Orientada a clave/valor

Son sencillas de entender. Simplemente guardan tuplas que contienen una clave y su valor. Cuándo se quiere recuperar un dato, simplemente se busca por su clave y se recupera el valor. Algunos ejemplos de base de datos clave/valor: DynamoDB, Redis.

- DynamoDB: desarrollada por Amazon, es una opción de almacenaje que puedemos usar desde los Amazon Web Services. La utilizan el Washington Post y Scopely.
- Redis: desarrollada en C y de código abierto, es utilizada por Craiglist y Stack Overflow (a modo de caché).

3.9.2.3.3. Orientada a documentos

Son aquellas que gestionan datos semi estructurados, en otras palabras documentos. Estos datos son almacenados en algún formato estándar como puede ser XML, JSON o BSON. Son las bases de datos NoSQL más versátiles. Se pueden utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales. Algunos ejemplos de base de datos orientada a documentos: MongoDB, CouchDB.

- MongoDB: probablemente la base de datos NoSQL más famosa del momento. En octubre del año pasado, MongoDB conseguía 150 millones de dólares en financiación, convirtiéndose en una da las startups más prometedoras. Algunas compañías que actualmente utilizan MongoDB son Foursquare o eBay.

  MongoDB ha sido creado para brindar escalabilidad, rendimiento y gran disponibilidad, escalando de una implantación de servidor único a grandes arquitecturas complejas de centros multidatos. MongoDB brinda un elevado rendimiento, tanto para lectura como para escritura, potenciando la computación en memoria. La replicación nativa de MongoDB y la tolerancia a fallos automática ofrece fiabilidad a nivel empresarial y flexibilidad operativa.
  [13]

  Algunas de las características de MongoDB son las siguientes: [14]

- **Consultas Ad hoc**
  MongoDB soporta la búsqueda por campos, consultas de rangos y expresiones regulares. Las consultas pueden devolver un campo específico del documento pero también puede ser una función JavaScript definida por el usuario.

- **Indexación**
  Cualquier campo en un documento de MongoDB puede ser indexado, al igual que es posible hacer índices secundarios. El concepto de índices en MongoDB es similar a los encontrados en base de datos relacionales.

- **Replicación**
  MongoDB soporta el tipo de replicación primario-secundario. Cada grupo de primario y sus secundarios se denomina replica set El primario puede ejecutar comandos de lectura y escritura. Los secundarios replican los datos del primario y sólo se pueden usar para lectura o para copia de seguridad, pero no se pueden realizar escrituras. Los secundarios tiene la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.

- **Balanceo de carga**
  MongoDB se puede escalar de forma horizontal usando el concepto de "shard". El desarrollador elige una clave de sharding, la cual determina cómo serán distribuidos los datos de una colección. Los datos son divididos en rangos (basado en la clave de sharding) y distribuidos a través de múltiples shard. Cada shard puede ser una replica set. MongoDB tiene la capacidad de ejecutarse en múltiple servidores, balanceando la carga y/o replicando los datos para poder mantener el sistema funcionando en caso que exista un fallo de hardware. La configuración automática es fácil de implementar bajo MongoDB y se pueden agregar nuevas servidores a MongoDB con el sistema de base de datos funcionando.

- **Almacenamiento de archivos**
  MongoDB puede ser utilizado como un sistema de archivos, tomando la ventaja de la capacidad que tiene MongoDB para el balanceo de carga y la replicación de datos utilizando múltiples servidores para el almacenamiento de archivos. Esta función se llama GridFS y es más bien una implementación en los drivers, no en el servidor, por lo que está incluida en los drivers oficiales que la compañía de MongoDB desarrolla. Estos drivers exponen funciones y métodos para la manipulación de archivos y contenido a los desarrolladores. En un sistema con múltiple servidores, los archivos pueden ser distribuidos y replicados entre los mismos y de una forma transparente, de esta forma se crea un sistema eficiente que maneja fallos y balanceo de carga.

- **Agregación**
  MongoDB proporciona un framework de agregación que permite realizar operaciones similares a las que se obtienen con el comando SQL "GROUP BY". El framework de agregación está construido como un pipeline en el que los datos van pasando a través de diferentes etapas en los cuales estos datos son modificados, agregados, filtrados y formateados hasta obtener el resultado deseado. Todo este procesado es capaz de utilizar índices si existieran y se produce en memoria. Asimismo, MongoDB proporciona una función MapReduce que puede ser utilizada para el procesamiento por lotes de datos y operaciones de agregación.

- **Ejecución de JavaScript del lado del servidor**
  MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.

- CouchDB: es la base de datos orientada a documentos de Apache. Una de sus interesantes características es que los datos son accesibles a través de una API Rest. Este sistema es utilizado por compañías como Credit Suisse y la BBC.

3.9.2.3.4. Orientada a grafo

Basadas en la teoría de grafos utilizan nodos y aristas para representar los datos almacenados. Son muy útiles para guardar información en modelos con muchas relaciones, como redes y conexiones sociales. Algunos ejemplos de base de datos orientada a grafos: Infinite Graph, Neo4j.

- Infinite Graph: escrita en Java y C++ por la compañía Objectivity. Tiene dos modelos de licenciamiento: uno gratuito y otro de pago.

- Neo4j: base de datos de código abierto, escrita en Java por la compañía Neo Technology. Utilizada por compañías como HP, Infojobs o Cisco.

3.10. Lenguajes de programación

En computación, un lenguaje de programación es cualquier lenguaje artificial, que intenta conservar una similitud con el lenguaje humano, el cual, se utiliza para definir adecuadamente una secuencia de instrucciones que puedan ser interpretadas y ejecutadas en una computadora.[15]

Establecen un conjunto de símbolos, reglas sintácticas y semánticas, las cuales rigen la estructura y el significado del programa, junto con sus elementos y expresiones. De esta forma, permiten a los programadores o desarrolladores, poder especificar

de forma precisa los datos sobre los que se va a actuar, su almacenamiento, transmisión y demás acciones a realizar bajo las distintas circunstancias consideradas. Usualmente se clasifican en interpretados y compilados, en el cual los compilados tienen un compilador específico que obtiene como entrada un programa y traduce las instrucciones las cuales pueden servir de entrada para otro interprete o compilado y los interpretados tienen un intérprete especifico que obtiene como entrada un programa y ejecuta las acciones escritas a medida que las va procesando.

3.10.1. R

R es un lenguaje y entorno de programación para análisis estadístico y gráfico, el cual proporciona un amplio abanico de herramientas estadísticas (modelos lineales y no lineales, tests estadísticos, análisis de series temporales, algoritmos de clasificación y agrupamiento, etc.) y gráficas. Se trata de un proyecto de software libre, resultado de la implementación GNU del premiado lenguaje S y se distribuye bajo la licencia GNU GPL y está disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux. Puede integrarse con distintas bases de datos y existen bibliotecas que facilitan su utilización desde lenguajes de programación interpretados como Perl y Python.[16]

R al estar orientado a las estadísticas, proporciona un amplio abanico de herramientas de cálculo numérico y a su vez para minería de datos, y posee una capacidad gráfica, que permite generar gráficos con alta calidad, con sólo utilizar las funciones de graficación.

3.10.1.1. R Studio

RStudio es un entorno de desarrollo integrado (IDE) construido exclusivo para R, para computación estadística y gráficos . Incluye una consola, editor de sintaxis que apoya la ejecución de código, así como herramientas para el trazado, la depuración y la gestión del espacio de trabajo. [17]

Algunas de sus características son:

- Ejecutar código R directamente desde el editor de código fuente.
- Salto rápido a las funciones definidas.
- Colaborativo.
- Documentación y soporte integrado.
- Administración sencilla de múltiples directorios de trabajo mediante proyectos.
- Navegación en espacios de trabajo y visor de datos.
- Potente autoría y depuración.
- Depurador interactivo para diagnosticar y corregir los errores rápidamente.

- Herramientas de desarrollo extensas.
- Autoría con Sweave y R Markdown.

3.10.2. Java

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Apple, etc. Esto lo consigue porque se ha creado una Máquina de Java para cada sistema que hace de puente entre el sistema operativo y el programa de Java y posibilita que este último se entienda perfectamente. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. [18]

El lenguaje Java se creó con cinco objetivos principales:

- Debería incluir por defecto soporte para trabajo en red.
- Debería usar el paradigma de la programación orientada a objetos.
- Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.
- Debería diseñarse para ejecutar código en sistemas remotos de forma segura.

3.10.3. JavaScript

JavaScript (a veces abreviado como JS, verdadero nombre ECMAScript) es un lenguaje ligero e interpretado más conocido como el lenguaje de script para páginas web, pero también usado en muchos entornos sin navegador, tales como Node.js.[19] Algunas características de JavaScript:

- Imperativo y estructurado.

- **Débilmente tipado:** Como en la mayoría de lenguajes de scripting, el tipo está asociado al valor, no a la variable.
- **Objetual:** JavaScript está formado casi en su totalidad por objetos. Los objetos en JavaScript son arrays asociativos, mejorados con la inclusión de prototipos. Los nombres de las propiedades de los objetos son claves de tipo cadena.
- **Evaluación en tiempo de ejecución:** JavaScript incluye la función 'eval' que permite evaluar expresiones como expresadas como cadenas en tiempo de ejecución.
- **Funcional:** A las funciones se les suele llamar ciudadanos de primera clase; son objetos en sí mismos. Como tal, poseen propiedades y métodos. Una función anidada es una función definida dentro de otra. Esta es creada cada vez que la función externa es invocada. Además, cada función creada forma una clausura; es el resultado de evaluar un ámbito conteniendo en una o más variables dependientes de otro ámbito externo, incluyendo constantes, variables locales y argumentos de la función externa llamante. El resultado de la evaluación de dicha clausura forma parte del estado interno de cada objeto función, incluso después de que la función exterior concluya su evaluación.
- **Prototipos:** es un estilo de programación orientada a objetos en el cual los objetos no son creados mediante la instanciación de clases sino mediante la clonación de otros objetos o mediante la escritura de código por parte del programador.

Mediante el uso de JavaScript se le añadió dinamismo a las páginas web que solían ser estáticas. JavaScript trajo consigo una forma de manipular el DOM y permitir realizar cambios sobre la página web al reaccionar a eventos que ocurren durante la visualización de la página. Algunos de los eventos mediante los cuales JavaScript realiza alguna acción son load, change, mouse over, mouse out, click, etc.

3.10.3.1. Node.js

Es un entorno en tiempo de ejecución asíncrono capaz de correr aplicaciones escritas en lenguaje Javascript del lado del servidor, y está diseñado para construir aplicaciones web altamente escalables de manera rápida reduciendo considerablemente el tiempo de desarrollo.[20]

Arquitectura y Funcionamiento

Posee una arquitectura orientada a eventos, basado en el motor V8 de Google. En contraste con el modelo de concurrencia más común de hoy en día donde se emplean hilos de ejecución del sistema operativo, consiste en que casi ninguna función de node.js realiza una operación de entrada/salida directamente por lo que nunca se bloquea de allí su asincronía, de allí parte su característica de eficiencia y está diseñado para ejecutarse bajo el protocolo HTTP.

Node.js y JavaScript

Como fue mencionado con anterioridad Node.js no separa la aplicación de diferentes lenguajes del lado del cliente o servidor, todo es desarrollado con javascript, esto trae como consecuencias excelentes ventajas dadas por el mismo lenguaje además del uso con anterioridad para el desarrollo web lo que reduce drásticamente el tiempo de aprendizaje de cierto lenguaje en particular.

Servidor Node.js

El servidor que se crea con Node.js es muy eficiente para aplicaciones web altamente escalables y se debe a su asincronía, comparando servidores que corren en lenguajes como Java o PHP cada conexión que se realiza a la aplicación web se crea un hilo de ejecución, el cual necesita aproximadamente 2MB de memoria para ser ejecutado y si tenemos un hipotético caso de un servidor con 8GB de memoria ram el mismo solo permite la atención simultánea de 4000 usuarios, el cual puede quedarse corto para soportar un alto tráfico de aplicaciones de hoy en día como lo son redes sociales. Siendo estas razones las cuales generan el cuello de botella de la arquitectura de las aplicaciones web.
Node resuelve el problema antes expuesto cambiando la forma en la que se realiza una conexión al servidor, ya que en lugar de tener un nuevo hilo por cada conexión y asignarle la cantidad de memoria correspondiente, se dispara una ejecución de evento dentro del proceso de Node, por lo tanto nunca se quedará en punto muerto porque no existen bloqueos. Node está orientado a aplicaciones intensivas de datos en tiempo-real que se ejecutan a través de dispositivos distribuidos, por lo tanto puede ejecutar decenas de miles de conexiones concurrentes a la vez.
La programación de las aplicaciones se realiza de forma asíncrona, es decir no bloquean la línea de ejecución del código con respecto a entradas y salidas, implementando callbacks, que son funciones que se indican con cada operación E/S para continuar.

Módulos Node.js

Los módulos de Node son librerías de código donde están contenidas funciones objetos o variables, las cuales pueden ser importadas para el proyecto que se esté utilizando. Estos módulos son desarrollados por la comunidad de programadores que impulsa esta plataforma de desarrollo, sin embargo existen algunos módulos básicos utilizados como por ejemplo el utilizado para crear el servidor http.

3.10.3.2. Framework: Meteor.js

Meteor es una plataforma para crear aplicaciones web y móviles en tiempo real construida sobre Node.js, la misma está localizada en el modelo de datos y la vista de la aplicación donde mantiene la sincronización entre estos dos entes. Al estar basada en Node.js la misma utiliza como lenguaje base Javascript del lado del cliente

y del servidor.[21]

Esquema

Este framework no está basado en el patrón de diseño de software llamado MVC (modelo, vista y controlador), a diferencia de la mayoría de los framework web lo que permite estructurar la aplicación según la necesidad del programador, esto lo realiza para mantener simplicidad de comunicación entre la capa vista y el modelo de datos utilizado, simplicidad es llamada "data on the wire" y significa que los datos viajan entre el modelo y la vista de manera simple y rápida. Meteor es flexible en cuanto a la forma de estructurar los archivos de la aplicación, se realiza automáticamente la carga de los archivos por lo que no hay necesidad del uso de etiquetas <script> ni <link> dentro del código HTML.

Plantillas

Todas las vistas dentro de Meteor están definidas en plantillas (templates). Una plantilla es un fragmento de código HTML que incluye datos que cambian dinámicamente todos manipulados con código JavaScript.

Paquetes

Muchas de las funcionalidades necesarias para el desarrollo de las aplicaciones están contenidas en paquetes modulares, algunos distribuidores de estos paquetes son: NPM, Atmosphere y GitHub.

Colecciones

Son estructuras de datos destinadas al almacenamiento de datos tipo documentales en formato JSON, para esto Meteor utiliza un manejador de base de datos por defecto que es MongoDB, sin embargo se puede trabajar con manejadores de base de datos relacionales como MySQL, Oracle o Postgres y la interacción entre el modelo y la base de datos se realiza con la sintaxis de MongoDB actuando como ORM (Object Relational Model) entre la aplicación y el manejador de base de datos.

3.10.4. Python

Python es un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma, cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Es administrado por la Python Software Foundation.[22] Posee una licencia de código abierto, denominada Python Software Foundation License. Algunas de las características de Python son:

- Orientado a Objetos : La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.
- Funciones y librerías: Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que se pueden importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos en .zip.
- Propósito general: Se pueden crear todo tipo de programas. No es un lenguaje creado específicamente para la web, aunque entre sus posibilidades sí se encuentra el desarrollo de páginas.
- Multiplataforma: Hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.
- Interpretado: Quiere decir que no se debe compilar el código antes de su ejecución. En realidad sí se realiza una compilación, pero esta lo hace de manera transparente para el programador. En ciertos casos, cuando se ejecuta por primera vez un código, se producen unos bytecodes que se guardan en el sistema y que sirven para acelerar la compilación implícita que realiza el intérprete cada vez que se ejecuta el mismo código.
- Interactivo: Python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudar a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.
- Sintaxis clara: Por último, destacar que Python tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. En muchos lenguajes, para separar porciones de código, se utilizan elementos como las llaves o las palabras clave begin y end. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

3.11. Apache Hadoop

Apache Hadoop es un framework que permite el procesamiento de grandes volúmenes de datos a través de clusters, usando un modelo simple de programación. En otras palabras, es un framework de software que soporta aplicaciones distribuidas bajo una licencia libre de la comunidad de Apache.[23] Algunas de sus ventajas se basan en sus principales cualidades:

- Velocidad: garantiza una alta eficiencia de procesamiento.
- Flexibilidad: se adapta a las necesidades del negocio, permite la utilización de diversas fuentes de datos y distintos tipos de datos.
- Escalabilidad: permite almacenar y distribuir conjuntos de datos inmensos en sus cientos de servidores que operan en paralelo, permitiendo olvidarse de los límites que otras alternativas imponen.
- Resistencia al fracaso: su tolerancia a errores es uno de sus atributos mejor valorados por los usuarios ya que toda la información contenida en cada nodo tiene su réplica en otros nodos del cluster. En caso de producirse un fallo siempre existirá una copia lista para ser usada.

Hadoop usa una arquitectura maestro-esclavo (Master-Slave), usando para almacenar datos Hadoop Distributed File System (HDFS) y algoritmos de MapReduce para hacer cálculos. Permite a las aplicaciones trabajar con miles de nodos y petabytes de datos. Hadoop trata de ser confiable, proveer alta disponibilidad y manejo de fallos.

Los servicios de Hadoop proporcionan almacenamiento, procesamiento, acceso, gestión, seguridad y operaciones de datos. Posee 4 módulos esenciales, Common, HDFS, YARN y MapReduce como se muestra a continuación: (Fig. 3.1).

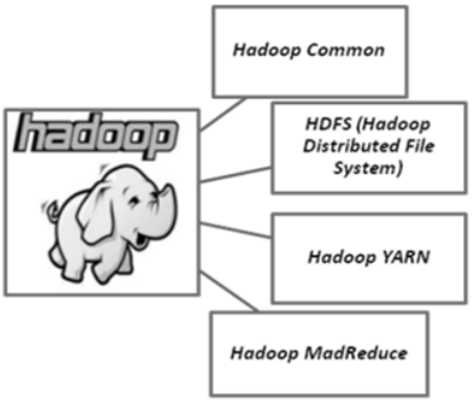

Figura 3.1: Apache Hadoop.

3.11.1. Common

Contiene bibliotecas y otras utilidades que necesitan los otros módulos de Hadoop. Provee un conjunto de utilidades que permiten el soporte a otros proyectos de Hadoop, estas utilidades son consideradas como el núcleo del framework que provee servicios esenciales para otras aplicaciones basadas en Hadoop.[24] Proporciona abstracciones a nivel de sistema de archivos o sistema operativo. Contiene los archivos .jar (Java ARchive) y scripts necesarios para iniciar Hadoop. Al igual que todos los demás módulos, Hadoop Common maneja los fallos de hardware comunes de forma automática. El proyecto Apache Commons se compone de tres partes:

- Commons Proper: Un repositorio de componentes Java reutilizables.
- Commons Sandbox: Un espacio de trabajo para el desarrollo de componentes de Java.
- Commons Dormant: Un repositorio de componentes que se encuentran actualmente inactivas.

3.11.2. Hadoop Distributed File System (HDFS)

Es un sistema de archivos distribuido que proporciona acceso de alto rendimiento a datos. Fue creado a partir del Google File System (GFS). HDFS se encuentra optimizado para grandes flujos y trabajar con ficheros grandes en sus lecturas y escrituras. La escalabilidad y disponibilidad son otras de sus claves, gracias a la replicación de los datos y tolerancia a los fallos. [25]

3.11.2.1. Características

HDFS posee características muy útiles para el manejo de grandes volúmenes de datos:

- Es adecuado para el almacenamiento y procesamiento distribuido.
- Permite el manejo de grandes archivos que pueden pesar cientos de Mega-Bytes, Giga-Bytes, Tera-Bytes, Peta-Bytes, etc.
- Proporciona permisos de archivo y autenticación.
- Los servidores ayudan a los usuarios a comprobar fácilmente el estado del clúster.
- Hadoop proporciona una interfaz de comandos para interactuar con HDFS.
- Tolerancia a fallos: para poder tener siempre disponible los datos en caso de ser requeridos, utiliza la replicación de estos en distintos nodos.
- Los nodos pueden hablar entre ellos para reequilibrar datos, mover copias, y conservar una alta replicación.

3.11.2.2. Arquitectura

HDFS sigue una arquitectura maestro-esclavo y contiene los siguientes elementos: (Fig. 3.2)

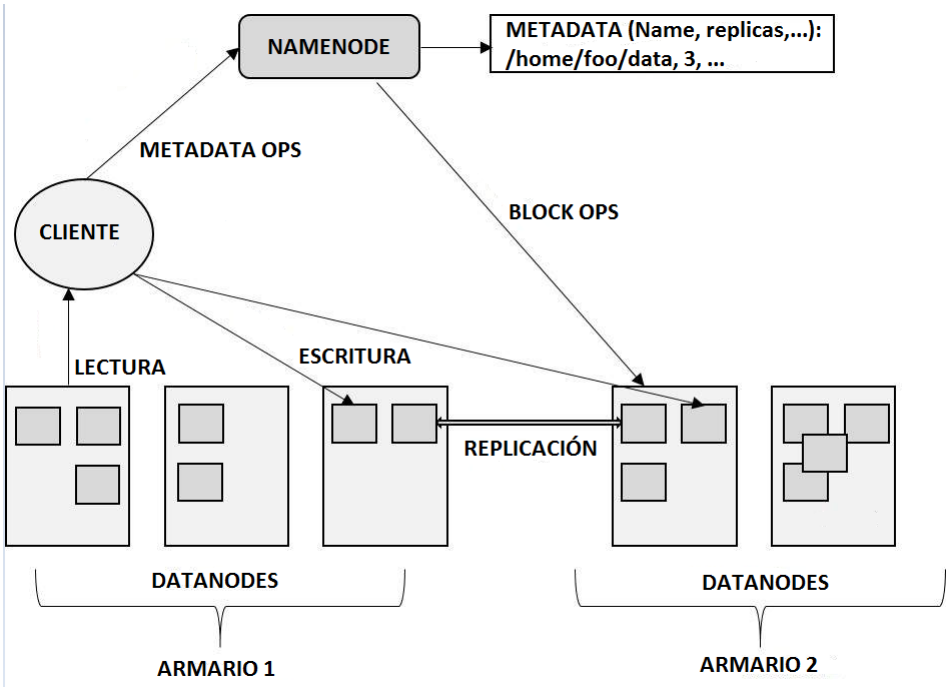

Figura 3.2: Arquitectura de HDFS.

- Namenode: Actúa como el servidor maestro y se encarga de la administración del espacio de nombres del sistema de archivos, regula el acceso a los ficheros por parte de los clientes, ejecuta las operaciones del sistema de archivos como el cambio de nombre, cierre y apertura de archivos y directorios y realiza el mantenimiento del sistema de archivos y la metadata asociada a todos estos. Regula el acceso a los ficheros por parte de los clientes. Mantiene en memoria la metadata del sistema de ficheros y control de los bloques de fichero que tiene cada DataNode.

- Datanode: Son los responsables de leer y escribir las peticiones de los clientes. Almacena y distribuye los bloques entre los distintos nodos. Permiten realizar operaciones tales como creación, supresión, con lo que la replicación de acuerdo con las instrucciones del namenode. Los ficheros están formados por bloques, estos se encuentran replicados en diferentes nodos. La distribución de los bloques es realizada cuando reciben un aviso desde un namenode o algún cliente (explicado en el siguiente punto) solicita una distribución. Una vez realizada la distribución, estos realizan un reporte al namenode, este reporte se realiza periódicamente y contiene los bloques los cuales están siendo almacenados en ese nodo en específico.

- Bloque: En general los datos de usuario se almacenan en los archivos de HDFS. El archivo se divide en uno o más segmentos y son almacenados en los nodos. Estos segmentos, es decir, la cantidad mínima de datos que HDFS puede leer o escribir se llama un bloque. El tamaño de bloque por defecto es de 64 MB, pero puede ser modificado.

- Clientes: son los usuarios del sistema de archivos

HDFS se gestiona a través de un servidor NameNode dedicado para alojar el índice de sistema de archivos y un NameNode secundario destinado a generar instantáneas de estructuras de memoria del NameNode principal. De esta manera, se evita la corrupción del sistema de archivos y la reducción de pérdida de datos.

3.11.3. Yet Another Resource Negotiator (YARN)

Apache Hadoop YARN (por las siglas en inglés de "otro negociador de recursos") es una tecnología de administración de clústeres. Facilita la planificación de tareas y gestión de recursos de clúster. Nace para dar solución a una idea fundamental: Dividir las dos funciones principales del JobTracker (NameNode), es decir, tener en servicios o demonios totalmente separados e independientes la gestión de recursos por un lado y, por otro, la planificación y monitorización de las tareas o ejecuciones.[26]

Gracias a YARN, Hadoop tiene un entorno de gestión de recursos y aplicaciones distribuidas dónde se pueden implementar múltiples aplicaciones de procesamiento de datos totalmente personalizadas y específicas para realizar una tarea en cuestión. 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. Ejecutándose en clústeres de hardware básicos, Hadoop ha atraído un interés particular como zona de espera y de almacenamiento de datos para grandes volúmenes de datos estructurados y no estructurados destinados al uso en aplicaciones de analítica. Separar HDFS de MapReduce con YARN hace al ambiente Hadoop más adecuado para las aplicaciones operativas que no pueden esperar para que terminen los trabajos por lotes.

3.11.3.1. Características

- Procesamiento: soporta distintos modelos de procesamiento y posee la capacidad de adaptarse a muchos más.
- Compatibilidad: las aplicaciones existentes de MapReduce pueden correr en YARN sin inconvenientes.
- Escalabilidad: YARN evita ciertos problemas de cuello de botella en grandes clúster.

3.11.3.2. Arquitectura

Los elementos que intervienen son:

- Resource Manager: asigna los recursos del clúster de manera abstracta a través de contenedores (Containers) los cuales incorporan elementos como memoria, CPU, disco, red, entre otros.

- Node Manager: hay uno por nodo esclavo, es el responsable de la monitorización y gestión de los recursos. Recoge las directrices del ResourceManager y crea contenedores en base a los requerimientos de la tarea.
- Application master: se despliega junto al NodeManager. Controla la monitorización y la ejecución de las tareas usando el contenedor. Puede ser alguna librería específica de algún framework que es la encargada de negociar recursos con el ResourceManager y coordinar las tareas con el NodeManager. Proporciona tolerancia a fallos a nivel de tarea.
- Container: es la unidad básica de la asignación. El contenedor se define con atributos como la memoria, CPU, disco, entre otros, aplicaciones como procesamiento gráfico y MPI.
- History Server: mantiene la historia de todos los Jobs.

  En la siguiente imagen 3.3 se explica la distribución de cada uno de los elementos mencionados anteriormente.

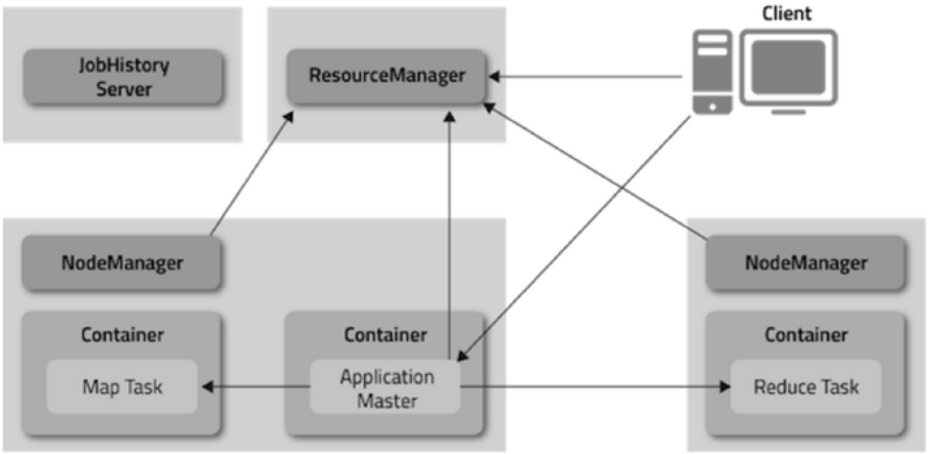

Figura 3.3: Arquitectura de YARN.

3.11.4. MapReduce

MapReduce fue desarrollado por Google y expuesto al resto del mundo en una publicación hecha por ellos mismos en diciembre del 2004, Google usa MapReduce para indexar páginas web. MapReduce es un modelo de programación con una implementación asociada al procesamiento y generación de grandes cantidades de datos. Los usuarios especifican una función de *Map* que procesa pares clave/valor para generar un grupo intermedio de pares clave/valor y una función de *Reduce* que combina todos los valores intermedios.[27]

Es un subproyecto del proyecto Hadoop. Posee una arquitectura maestro-esclavo y es un paradigma de programación que permite la escalabilidad masiva a través de cientos o miles de servidores en un clúster Hadoop. Cuenta con un servidor maestro

o JobTracker y varios servidores esclavos o TaskTrackers, uno por cada nodo del clúster. El JobTracker es el punto de interacción entre los usuarios y el framework MapReduce. Los usuarios envían trabajos MapReduce al JobTracker, que los pone en una cola de trabajos pendientes y los ejecuta en el orden de llegada.

El JobTracker gestiona la asignación de tareas y delega las tareas a los TaskTrackers. Los TaskTrackers ejecutan tareas bajo la orden del JobTracker y también manejan el movimiento de datos entre la fase Map y Reduce.

3.11.4.1. JobTracker

- Capacidad para manejar metadatos de trabajos.
- Estado de la petición del trabajo.
- Estado de las tareas que se ejecutan en TaskTracker.
- Decide sobre la programación.
- Hay exactamente un JobTracker por cluster.
- Recibe peticiones de tareas enviadas por el cliente.
- Programa y monitoriza los trabajos MapReduce con TaskTrackers.

3.11.4.2. TaskTracker

- Ejecuta las solicitudes de trabajo de JobTrackers.
- Obtiene el código que se ejecutará.
- Aplica la configuración especifica del trabajo.
- Comunicación con el JobTracker: Envíos de la salida, finalizar tareas, actualización de tareas, etc.

3.11.4.3. Map

La función Map recibe como parámetros un par clave-valor y devuelve una lista de pares. Esta función se encarga del mapeo y se aplica a cada elemento de la entrada de datos, por lo que se obtendrá una lista de pares por cada llamada a la función Map. Después se agrupan todos los pares con la misma clave de todas las listas, creando un grupo por cada una de las diferentes claves generadas. No hay requisito de que el tipo de datos para la entrada coincida con la salida y no es necesario que las claves de salida sean únicas.

3.11.4.4. Reduce

La función Reduce se aplica en paralelo para cada grupo creado por la función Map().La función Reduce se llama una vez para cada clave única de la salida de la función Map. Junto con esta clave, se pasa una lista de todos los valores asociados con la clave para que pueda realizar alguna fusión y producir un conjunto más pequeño de los valores.

Cuando se inicia la tarea Reduce, la entrada se encuentra dispersa en varios archivos a través de los nodos en las tareas de Map. Los datos obtenidos de la fase Map se ordenan para que los pares clave-valor sean contiguos (fase de ordenación, sort fase), esto hace que la operación Reduce se simplifique ya que el archivo se lee secuencialmente.

Si se ejecuta el modo distribuido estos necesitan ser primero copiados al filesystem local en la fase de copia. Una vez que todos los datos están disponibles a nivel local se adjuntan a una fase de adición, el archivo se fusiona (merge) de forma ordenado. Al final, la salida consistirá en un archivo de salida por tarea reduce ejecutada.

Por lo tanto, N archivos de entrada generará M mapas de tareas para ser ejecutados y cada mapa de tareas generará tantos archivos de salida como tareas Reduce hayan configuradas en el sistema.

3.11.5. Ecosistema Hadoop

El ecosistema de Hadoop posee un conjunto de herramientas muy diverso, que crece día tras día, por lo que es difícil saber de todos los proyectos que interactúan con Hadoop de alguna forma, las cuales a su vez poseen subconjuntos, en este capítulo se mencionará algunos de ellos y una breve descripción.

3.11.5.1. Administración de los datos

Son herramientas que permiten el almacenamiento y el manejo de datos, como HDFS y YARN que fueron mencionados anteriormente.

3.11.5.2. Acceso a los datos

Son herramientas que se encargan del acceso a los datos por parte de los usuarios y permiten la lectura, escritura y procesamiento de los datos.

3.11.5.2.1. Apache Accumulo

Apache Accumulo es un store clave/valor distribuido, escalable y de alto rendimiento. Se basa en el diseño de Google BigTable y se construye sobre Hadoop, Zookeeper y Thrift.

[28]

Accumulo permite el manejo de datos a nivel de celdas, lo cual es una funcionalidad muy importante debido a que se puede restringir el acceso a los datos a ciertos usuarios en específico. Permite también la mezcla de distintos datos los cuales pueden estar restringidos o no, las reglas que pueden ser aplicadas a los datos pueden llegar a ser muy específicas.

3.11.5.2.2. Apache Hbase

HBase, se trata de la base de datos de Hadoop. HBase es el componente de Hadoop a usar cuando se requiere escrituras/lecturas en tiempo real y acceso aleatorio para grandes conjuntos de datos. Es una base de datos distribuida orientada a columnas que funciona sobre HDFS, eso quiere decir que no sigue el esquema relacional. [29]

Características de HBase:

- Escalabilidad linear y modular.
- Soporte para caídas de nodos.
- Proporciona lectura coherente y escrituras.
- Se integra con Hadoop, tanto como un origen y un destino.
- Compatibilidad con trabajos MapReduce.
- Proporciona replicación de datos en clústeres.

Las aplicaciones de HBase:

- Apache HBase se utiliza para tener al azar y en tiempo real de acceso de lectura/escritura a los grandes datos.
- Alberga las tablas de gran tamaño en la parte superior de los grupos de hardware de productos básicos.
- Se usa cuando es necesario escribir aplicaciones pesadas.
- HBase se utiliza cada vez que se necesite proporcionar un rápido acceso aleatorio a los datos disponibles.
- Empresas como Facebook, Twitter, Yahoo y Adobe usan HBase internamente.

A continuación se presenta una comparación entre HDFS y HBase. (Tab. 3.2)

HDFS	HBase
HDFS es un sistema de ficheros distribuido adecuado para almacenar archivos de gran tamaño.	HBase es una base de datos creada en la parte superior de la HDFS.
HDFS no admite búsquedas rápidas registro individual.	HBase proporciona búsquedas rápidas tablas más grandes.
Proporciona una alta latencia procesamiento por lotes; un concepto de procesamiento por lotes.	Proporciona acceso de baja latencia a filas de miles de millones de registros (acceso aleatorio).
Sólo proporciona acceso secuencial de los datos.	HBase internamente usa tablas Hash y proporciona acceso aleatorio, y que almacena los datos en archivos indexados HDFS búsquedas más rápido.

Tabla 3.2: Comparación entre HBase y HDFS.

3.11.5.2.3. Apache Hive

Hive es un sistema de Data Warehouse para Hadoop que facilita el uso de la agregación de los datos, ad-hoc queries, y el análisis de grandes datasets almacenados en Hadoop. Hive proporciona métodos de consulta de los datos usando un lenguaje parecido al SQL, llamado HiveQL. Posee interfaces JDBC/ODBC, por lo que empieza a funcionar su integración con herramientas de inteligencia de negocios. [30] Características de Hive:

- Esquema que almacena en una base de datos y se procesan los datos en HDFS.
- Está diseñado para OLAP.
- Proporciona un lenguaje de consulta parecido a SQL, HiveQL (HQL).
- Es familiar, rápido, escalable y extensible.

3.11.5.2.4. Apache Storm

Apache Storm es un sistema que sirve para recuperar streams de datos en tiempo real desde múltiples fuentes de manera distribuida, tolerante a fallos y en alta disponibilidad. Storm está principalmente pensado para trabajar con datos que deben ser analizados en tiempo real, por ejemplo datos de sensores que se emiten con una alta frecuencia o datos que provengan de las redes sociales donde a veces es importante

saber qué se está compartiendo en este momento.
[31]

Se compone de dos partes principalmente. La primera es la que se denomina Spout y es la encargada de recoger el flujo de datos de entrada. La segunda se denomina Bolt y es la encargada del procesado o transformación de los datos.

3.11.5.2.5. Apache Mahout

Es una librería de los algoritmos más famosos de aprendizaje automático (Machine Learning), fue implementado sobre Hadoop utilizando el paradigma de MapReduce. Con los datos almacenados en HDFS, Mahout provee las herramientas necesarias para poder aplicar algoritmos sobre estos.
[32]

Proporciona bibliotecas de Java para las operaciones matemáticas comunes (centrado en álgebra lineal y estadística) y las colecciones de Java primitivos. También ha añadido un número de algoritmos matemáticos de bajo nivel que los usuarios pueden encontrar útil.

Mahout ayuda a descubrir patrones en grandes datasets. Tiene algoritmos de recomendación, clustering y clasificación. Algunos de los algoritmos que provee Mahout son los siguientes: (Tab. 3.3)

Algoritmo	Descripción breve	Caso de uso
Regresión logística, resuelta por gradiente estocástico descendiente (SGD).	Clasificador brillante, rápido, simple y secuencial, capaz de aprendizaje online en entornos exigentes.	Recomiende publicidad a los usuarios, clasifique texto en categorías.
Modelos ocultos de Markov (HMM).	Implementaciones secuenciales y paralelas del algoritmo clásico de clasificación diseñado para modelar procesos del mundo real cuando el proceso de generación subyacente es desconocido.	Etiquetado de texto parte-del-discurso; reconocimiento del discurso.
Descomposición de valor singular (SVD).	Diseñado para reducir el ruido en matrices grandes, haciendo con esto que sean más pequeñas y que sea más fácil trabajar con ellas.	Como precursor del almacenamiento en clúster, los recomendadores y la clasificación para realizar selección de recursos automáticamente.
Almacenamiento en clúster Dirichlet.	Enfoque de almacenamiento en clúster basado en modelo, que determina la propiedad con base en si los datos se ajustan al modelo subyacente.	Útil cuando los datos tienen sobreposición o jerarquía.
Almacenamiento en clúster espectral.	Es una familia de enfoques similares que usa un enfoque basado en gráficas para determinar la membresía a clúster.	Como todos los algoritmos de almacenamiento en clúster, es útil para explorar conjuntos de datos grandes y no vistos.
Almacenamiento en clúster Minhash.	Utiliza una estrategia de hash para agrupar elementos similares, produciendo así clústeres.	Igual a otros enfoques de clúster.
Numerosas mejoras de recomendador.	Co-ocurrencia distribuida, SVD, mínimos cuadrados alternantes.	Sitios de citas, e-commerce, recomendaciones de películas o de libros.
Colocaciones.	Implementación de colocación reducida por correlacionamiento.	Encontrando frases estadísticamente interesantes en texto.

Tabla 3.3: Algoritmos Mahout.

3.11.5.2.6. Apache Drill

Es un framework de código abierto que soporta la intensidad del manejo de datos típica de aplicaciones distribuidas encargadas del análisis interactivo de de grandes conjunto de datos. Se puede considerar a Drill como la versión de código abierto del sistema Dremel de Google que está disponible en una infraestructura de servicios llamada BigQuery. Drill es capaz de escalar 10.000 servidores o más para el procesamiento de petabytes de datos y trillones de registros en segundos, lo cual hace que sea un proyecto 'top' de Apache. Utiliza un modelo basado en JSON similar a MongoDB sin la necesidad de declarar un esquema formal, con una arquitectura que permite agregar múltiples funcionalidades a los almacenes de datos.[33] Su soporte se enfoca en almacenamiento no relacional incluyendo Hadoop, NoSQL y almacenamiento en la nube, algunos son:

- Todas las distribuciones de Hadoop incluyendo Apache Hadoop y MapR
- Bases de datos NoSQL como MongoDB y HBase
- Almacenamiento en la nube de Amazon S3 y Google Cloud Storage
- Soporta sistemas de administración de bases de datos relacionales usando el API JDBC (Java DataBase Connectivity).

3.11.5.3. Integración

Aquellas herramientas que facilitan la extracción, transformación, replicación, entre otros, de los datos, son las herramientas de integración.

3.11.5.3.1. Apache Falcon

Falcon es un sistema de procesamiento y manejo de carga de datos destinada a facilitar a los consumidores finales a bordo de sus procesamiento de datos. Establece relación entre los diversos elementos de procesamiento de datos y en un entorno de Hadoop. Posee soporte para el manejo de datos finales, integración con MetaStore/Hive/HCatalog. Proporciona notificaciones al cliente final sobre la base de la disponibilidad de los grupos de carga de datos.[34]

Básicamente, es una herramienta de software que simplifica la creación y el manejo de tuberias (pipelines) de procesamiento de datos.

3.11.5.3.2. Apache Flume

Apache Flume es un producto que forma parte del ecosistema Hadoop, y conforma una solución Java distribuida y de alta disponibilidad para recolectar, agregar y mover grandes cantidades de datos desde diferentes fuentes a un data store centralizado.

Surge para subir datos de aplicaciones al HDFS de Hadoop. Su Arquitectura se basa en flujos de streaming de datos, ofrece mecanismos para asegurar la entrega y mecanismos de failover y recuperación. Ofrece una gestión centralizada. [35]

3.11.5.3.3. Apache Sqoop

Apache Sqoop ("Sql-to-Hadoop"), es una herramienta diseñada para transferir de forma eficiente bulk data entre Hadoop y sistemas de almacenamiento con datos estructurados, como bases de datos relacionales.[36] Algunas de sus características son:

- Permite importar tablas individuales o bases de datos enteras a HDFS.
- Genera clases Java que permiten interactuar con los datos importados.
- Además, permite importar de las bases de datos SQL a Hive.

3.11.5.4. Operaciones

Son las herramientas que permiten monitorear, administrar y realizar operaciones sobre un clúster Hadoop.

3.11.5.4.1. Apache Ambari

Es un proyecto que pertenece a la distribución Hortonworks, que facilita la gestión de Hadoop. Ofrece una interfaz web intuitiva y fácil de usar para la gestión de Hadoop y además proporciona una API REST.[37] Ambari permite a los administradores del sistema:

- Monitorizar el clúster Hadoop.
- Ofrece un panel de control para vigilancia de la salud y el estado del cluster Hadoop.
- Se encarga de la instalación de los paquetes de Hadoop en el clúster.
- Un asistente paso a paso para la instalación de servicios de Hadoop a través de múltiples equipos.
- Proporciona la forma de gestionar gestión central para iniciar, detener y volver a configurar los servicios de Hadoop en todo el clúster.

3.11.5.4.2. Apache Zookeeper

Zookeeper es un proyecto de Apache que proporciona una infraestructura centralizada y de servicios que permiten la sincronización del cluster. ZooKeeper mantiene objetos comunes que se necesiten en grandes entornos de cluster. [38]

3.11.5.4.3. Apache Oozie

Apache Oozie es un sistema de programación de flujo de trabajo basado en servidor para administrar los trabajos de Hadoop. Los flujos de trabajo en Oozie se definen como una colección de flujos de control y nodos de acción en un grafo dirigido acíclico.[39]

Los nodos de flujo de control definen el comienzo y el final de un flujo de trabajo (inicio, final y los nodos de fallo), así como un mecanismo para controlar la ruta de ejecución del flujo de trabajo. Los nodos de acción son el mecanismo por el cual un flujo de trabajo provoca la ejecución de una tarea de cálculo/procesamiento.

Oozie se implementa como una aplicación web en Java que se ejecuta en un contenedor de servlets Java y se distribuye bajo la licencia Apache 2.0.

3.11.5.5. Seguridad

Apache Hadoop provee herramientas de monitoreo y administración que aportan seguridad a los datos, mediante autorización, autenticación, protección, auditoría, entre otros.

3.11.5.5.1. Apache Knox

El Apache Knox Gateway es una puerta de enlace API REST para interactuar con los racimos de Hadoop. El Knox Gateway proporciona un único punto de acceso para todas las interacciones REST con racimos de Hadoop. En esta capacidad, Knox es capaz de proporcionar la funcionalidad valiosa de ayudar en el control, la integración, el seguimiento y la automatización de las necesidades administrativas y analíticas que son críticas en la empresa.[40]

- Autenticación (LDAP y proveedor de autenticación de Active Directory).
- Federación/SSO (encabezado HTTP basado federación de identidades).
- Autorización (Service Level Autorización).
- Revisión de cuentas.

3.11.5.5.2. Apache Ranger

Ranger es un framework que permite supervisar y gestionar la seguridad de datos completa a través de la plataforma Hadoop.[41]

La visión con Ranger es proporcionar seguridad integral en todo el ecosistema Hadoop. Con la llegada de Apache Yarn, la plataforma Hadoop ahora puede soportar una verdadera arquitectura de conjunto de datos. La seguridad de los datos dentro de Hadoop necesita evolucionar para soportar múltiples casos de uso para el acceso de datos, mientras que también proporciona un marco para la administración central de las políticas y el seguimiento de acceso de los usuarios de seguridad.

3.11.6. Apache Spark

Apache Spark es una alternativa para el procesamiento de grandes volúmenes de datos (Big Data) que ha suplantado a muchas herramientas. Es una plataforma de computación de código abierto para análisis y procesos avanzados, que tiene muchas ventajas sobre Hadoop. Desde el principio, Spark fue diseñado para soportar en memoria algoritmos iterativos que se pudiesen desarrollar sin escribir un conjunto de resultados cada vez que se procesaba un dato. Esta habilidad para mantener todo en memoria es una técnica de computación de alto rendimiento aplicado al análisis avanzado, la cual permite que Spark tenga unas velocidades de procesamiento que sean 100 veces más rápidas que las conseguidas utilizando MapReduce.[42]

Spark posee un framework integrado para implementar análisis avanzados que incluye la librería MLlib, GraphX, Spark Streaming, y la herramienta de consulta Shark. Esta plataforma asegura a los usuarios la consistencia en los resultados a través de distintos tipos de análisis.
A continuación se muestra la estructura de Apache Spark. (Fig. 3.4)

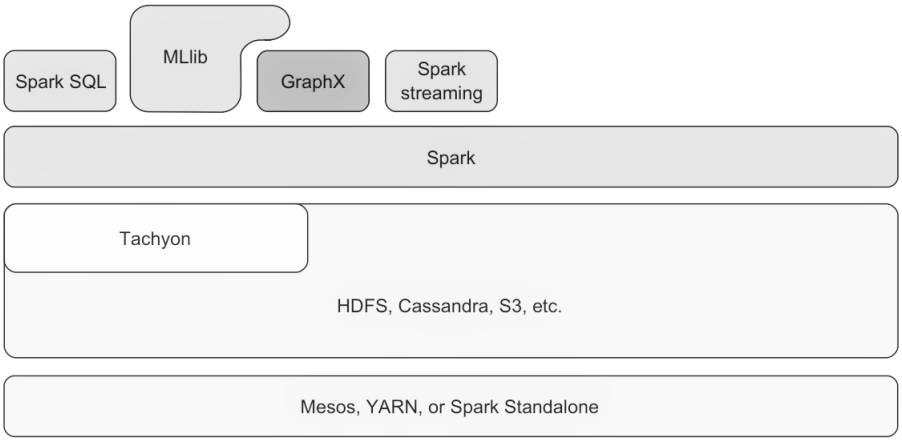

Figura 3.4: Apache Spark.

Al igual que Hadoop, Spark provee el API MapReduce, la diferencia es que Spark almacena los datos en memoria RAM del clúster y Hadoop en el disco duro. También proporciona procesamiento de datos en paralelo lo que lo hace ideal para los problemas de Big Data en el mundo real, ya que a menudo se requiere tanto procesamiento de grafos en paralelo como de datos. Spark mantiene la escalabilidad

lineal y la tolerancia a fallos de MapReduce, pero amplía sus bondades gracias a varias funcionalidades: DAG y RDD.

3.11.6.1. Spark SQL

Spark SQL trae un soporte nativo de SQL para Spark y agiliza el proceso de consulta de datos almacenados tanto en RDD como en otras fuentes. Difumina convenientemente las líneas entre RDD y las tablas relacionales. La unificación de estas potentes abstracciones hace que sea fácil para los desarrolladores entremezclar comandos de consulta SQL sobre datos externos para análisis complejos, todo dentro de una sola aplicación. Concretamente, permite a los desarrolladores:

- Importar datos relacionales a partir de archivos de Parquet y tablas de Hive.
- Ejecutar consultas SQL sobre los datos importados y RDD existentes.
- Escribir facilmente sobre RDD fuera de Hive o archivos Parquet.

3.11.6.2. MLlib

MLlib es un subproyecto de Spark que provee primitivas de aprendizaje automático escalable y sencillo. Esta compuesto por los algoritmos más comunes, como los de clasificación, clusterización, regresión, filtrado colaborativo, reducción de dimensionalidad, entre otros. Se encuentra divido en 2 paquetes:

- spark.mllib contiene el API original realizado sobre los RDDs.
- spark.ml provee un APi de alto nivel realizado sobre los DataFrames para construir ML pipelines.

Usualmente se utiliza spark.ml debido a que este API es más flexible y versátil manejando DataFrames.

3.11.6.3. Ventajas de Spark

- Spark tiene más potencia que hadoop:

  Para empezar, Spark es un framework de análisis distribuido en memoria y permite ir más allá de las operaciones en batch de Hadoop MapReduce: procesamiento de streaming, machine learning (MLlib), cálculo de grafos (GraphX), integración con lenguje R (Spark R) y análisis interactivos.

  Con todo esto, ahora se puede desarrollar nuevos proyectos de big data con menos presupuesto y soluciones más completas.

- Spark es rápido, muy rápido:

  Spark puede ejecutar análisis de varios órdenes de magnitud más rápido que los despliegues de Hadoop existentes. Esto significa una mayor interactividad, la experimentación más rápido y mayor productividad para los analistas.

- Spark puede coexistir con una arquitectura Big Data:

  Puede coexistir con las instalaciones existentes de Hadoop y añadir nuevas funcionalidades. Spark se integra perfectamente con Hadoop y en muchos de los proyectos se utiliza/almacenan los datos que están en el sistema de fichero de Hadoop HDFS y/o se ejecutan los procesos de Spark usando YARN de Hadoop 2.0. Además puede funcionar con muchos otros productos de Big Data como: CassandraDB, Google Big Query, almacenamiento de Amazon S3, Elastic Search, etc.

- Spark entiende SQL:

  El módulo Spark Sql es capaz de usar fuentes de datos existentes (HIVE, CassandraDB, MongoDB, JDBC, etc), se puede usar para gestionar las fuentes internas de datos (RDDs y/o DataFrames) como si fueran tablas estructuradas. Aunque Spark SQL no es la implementación más robusta y completa del mercado ya está lista para ser usada.

- Spark mima a los desarrolladores:

  Cuando una tecnología encanta a los desarrolladores se convierten en early adopters y empiezan a usarla y disfrutarla. Spark es un ejemplo de esto, cuando usan Spark solo tiene que dedicarse a resolver el problema. Spark se ha programado con el lenguaje Scala que un nuevo lenguaje funcional y orientado a objetos. Gracias a Scala son capaces de programar de manera muy concisa y fluida soluciones que antes requerían cientos de lineas. Además se puede programar en python, R e incluso en Java.

- Spark empieza a ser el motor de Big Data:

  Ahora mismo Apache Spark forma muchos proyectos de Big Data y empresas como IBM, Microsoft, Amazon, Google lo intengran con sus productos de Big Data.

Apache Spark es similar a Apache Hadoop en el sentido que almacenan los datos de forma distribuidas utilizando clusters, la diferencia es que Apache Spark los almacena en memoria (RAM) y Hadoop en disco. Hadoop tiene 2 inconvenientes, procesar consultas interactivas y algoritmos iterativos, por lo tanto nace Apache Spark para solventar estos problemas.
La siguiente imagen muestra la comparación entre Hadoop y Spark en uso de recursos. (Fig.3.5)

Figura 3.5: Apache Hadoop vs Apache Spark

3.11.7. Distribuciones Hadoop

3.11.7.1. Cloudera

Cloudera fue fundada en 2008 por algunas de las mentes más brillantes de las empresas líderes de Silicon Valley, incluyendo Google (Christophe Bisciglia), Yahoo (Amr Awadallah), Oracle (Mike Olson) y Facebook (Jeff Hammerbacher). Se fundó con el objetivo de ayudar a las empresas a usar Hadoop para obtener más valor de todos sus datos. Ofrece una plataforma de datos basada en Hadoop, rápida, fácil de usar y segura. Basada al 100 % en software de código abierto y estándares abiertos, la plataforma de Cloudera proporciona una mayor flexibilidad, más control de los costes y unos mejores resultados para la empresa. [43]

Cloudera Enterprise ofrece 3 opciones para utilizar Apache Hadoop, local, en una infraestructura basada en un entorno local o en la nube:

- QuickStart VM o Docker Image: Permite la utilización de Cloudera mediante la utilización de máquinas virtuales y contenedores Docker. Provee un entorno de pruebas (Sandbox) con diversos ejemplos y herramientas.
- Cloudera Manager: Provee una interfaz unificada para gestionar centros de datos empresariales.
- Cloudera Director: Provee un entorno en la nube que facilita la implementación y administración de Cloudera.

3.11.7.2. Hortonworks

Hortonworks es una compañía de software empresarial con sede en Santa Clara, California. La compañía se centra en el desarrollo y soporte de Apache Hadoop, posee un conjunto de procesos y herramientas que permiten el procesamiento distribuido de grandes conjuntos de datos a través de clusters.

Hortonworks se formó en junio de 2011, financiado por Yahoo. Utiliza Apache Ambari como administrador del sistema y HDFS como sistema de archivos, algunas herramientas desarrolladas por Hortonworks son:

- Slider: framework el cual permite la gestión de recursos en tiempo real, según los requerimientos de las herramientas ejecutadas. Permite administrar de una mejor manera el clúster debido a que permite asignar mayor o menor cantidad de recursos según los requerimientos de las herramientas, esto anteriormente no era posible dependiendo solo de YARN.
- Tez: framework diseñado para trabajar sobre YARN, permite la manipulación de datos producidos en lotes.
- Phoenix: provee una interfaz SQL sobre HBase para así poder tratar este almacén de datos con un lenguaje muy parecido a SQL. Este realiza una optimización sobre distintas consultas a realizar lo cual provee un mayor rendimiento.
- YARN: administrador de recursos de Hadoop.

3.11.7.3. MapR

MapR es una empresa de software con sede en San José, California, que ofrece soluciones sobre el ecositema Hadoop. La empresa contribuye a proyectos Apache Hadoop como HBase, Pig, Apache Hive, y Apache ZooKeeper. [44]

MapR es una empresa privada con financiación original de 9 millones de dólares por Lightspeed Venture Partners y New Enterprise Associates en el 2009. Utiliza de un sistema de archivos, una base de datos y un sistema de administración del clúster creados especialmente para esta distribución cuyos nombres son: MapR-FS, MapR-DB y MapR-Control System respectivamente.

- MapR-Control System. Es un sistema para poder controlar el ecosistema de la distribución, es equiparable a Apache Ambari. Provee sistemas de alarmas, una serie de medidores para observar el estado del clúster de una forma rápida. Este también funciona mediante una interfaz web que se comunica mediante un API el cual realiza todas las funcionalidades de esta herramienta.
- MapR-DB: Es una base de datos de calidad empresarial, fue creada basándose en el enfoque NoSQL. Es totalmente compatible con el ecosistema Hadoop, incluye soporte para distintas características como: modelo de datos basado

en grandes columnas (al estilo Apache HBase), escalabilidad al estilo de Hadoop, localidad de datos con trabajos MapReduce, alta consistencia de datos, transacciones ACID a nivel de filas, entre otras.

- MapR-FS: Es un sistema de archivos distribuido basado en el funcionamiento de HDFS pero mejorando ciertos puntos de fallo que este tiene. A diferencia de HDFS, MapR-FS no es estructurad, no depende de una arquitectura en específico, este diseño busca solventar la problemática del punto de fallo simple que tiene HDFS relacionado con el NameNode.

3.11.7.4. Cloudera vs Hortonworks

Cloudera, así como Hortonworks están ambos construidos sobre el mismo núcleo de Apache Hadoop. Como tales, tienen más similitudes que diferencias. Ambos ofrecen distribuciones de Hadoop listas para la empresa. Las distribuciones han superado la prueba del tiempo generadas por los consumidores, y garantizan seguridad y estabilidad.

Además, proporcionan entrenamientos y servicios pagos para familiarizar a los nuevos usuarios que pisan el camino de grandes volúmenes de datos y análisis. Ambos poseen una arquitectura maestro-esclavo, soportan YARN y MapReduce, y han establecido comunidades que participan de manera activa.

Cloudera ha anunciado que su objetivo a largo plazo es convertirse en una empresa especializada en data hub (colección de datos procedentes de distintas fuentes, organizados para ser posteriormente distribuidos y compartidos), disminuyendo así la necesidad de utilizar almacenes de datos. Hortonworks, por el contrario, mantiene su firme posición como proveedor de una distribución de Hadoop, y se ha asociado con la compañía de almacenes de datos Teradata.

Mientras Cloudera CDH se puede ejecutar en un servidor Windows, HDP está disponible como un componente nativo de Windows. Un clúster Hadoop basado en Windows puede utilizar Windows Azure a través del servicio HDInsight. Cloudera tiene un software de gestión propio llamado Cloudera Manager, una interfaz para consultas SQL (Impala), y Cloudera Search que permite un acceso fácil y en tiempo real de los productos. Hortonworks no tiene software propietario, utiliza Ambari para la gestión y Stinger para el manejo de consultas, y Apache Solr para las búsquedas de datos. Cloudera tiene uso comercial, mientras que Hortonworks tiene licencia de código abierto.

Cloudera también permite el uso de sus proyectos de forma gratuita, pero no incluye Cloudera Manager o cualquier otro software propietario. Cloudera tiene una versión de prueba de 60 días, Hortonworks es completamente libre. Cloudera ha sido el jugador de más edad en el mercado, con más de 350 clientes. Pero Hortonworks está igualando rápidamente y ha hecho más innovaciones en el ecosistema Hadoop en el pasado reciente.

3.11.8. D3.js

D3.js (o simplemente D3 por las siglas de Data-Driven Documents) es una librería de JavaScript distribuida en Agosto de 2011, para producir, a partir de datos, infogramas dinámicos e interactivos en navegadores web. Básicamente, la librería permite manipular documentos basados en datos usando estándares abiertos de la web y los navegadores pueden crear visualizaciones complejas sin depender de un software propietario. Hace uso de tecnologías bien sustentadas como SVG, HTML5, y CSS. Esta librería es sucesora de la librería Protovis. En contraste con muchas otras librerías, D3.js permite tener control completo sobre el resultado visual final. [45]

3.11.9. KDD: Knowledge Discovery in Databases

KDD es un proceso que se puede definir como "el proceso no trivial de identificar patrones válidos, novedosos y potencialmente útiles y en última instancia, comprensible a partir de los datos".[46] Este proceso también es conocido por diferentes nombres que podrían ser sinónimos del mismo, entre los cuales se encuentran Data Archeology, Dependency Function Analysis, Information Recollect, Pattern Data Analysis ó Knowledge Fishing.
Generalmente se consideran las siguientes etapas en el proceso: (Fig. 3.6)

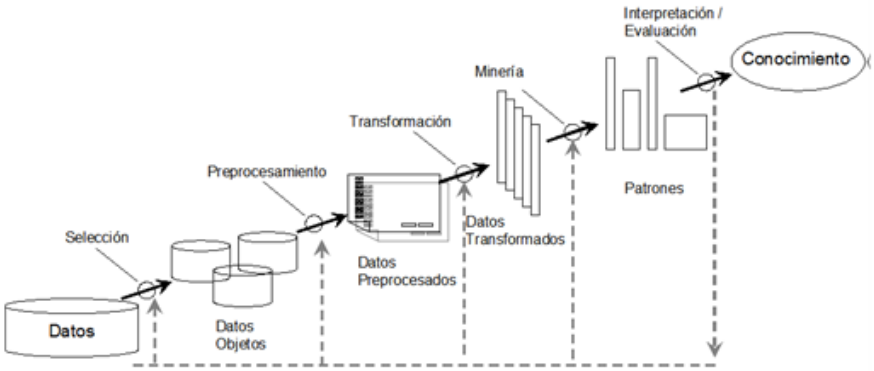

Figura 3.6: Proceso KDD.

- **Selección de datos:** Consiste en buscar el objetivo y las herramientas del proceso de minería, identificando los datos que han ser extraídos, buscando los atributos apropiados de entrada y la información de salida para representar la tarea. Esto quiere decir, primero se debe tener en cuenta lo que se saber lo que se quiere obtener y cuáles son los datos que nos facilitarán esa información para poder llegar a nuestra meta, antes de comenzar el proceso en sí. Básicamente ocurren dos acciones:
  1) Se determina fuente de datos y el tipo de información a utilizar.
  2) Se toman los dato relevantes y se descartan los que no lo son.

- **Preprocesamiento:** En este paso se limpian los datos 'sucios', incluyendo los datos incompletos (donde hay atributos o valores de atributos perdidos), el ruido (valores incorrectos o inesperados) y datos inconsistentes (conteniendo valores y atributos con nombres diferentes). Los datos sucios en algunos casos deben ser eliminados ya que pueden contribuir a un análisis inexacto y resultados incorrectos.

- **Transformación de datos:** Consiste en el tratamiento preliminar de los datos, transformación y generación de nuevas variables a partir de las ya existentes con una estructura de datos apropiada. Aquí se realizan operaciones de agregación o normalización, consolidando los datos de una forma necesaria para la fase siguiente.
  Las transformaciones discretas de los datos tienen la ventaja de que mejoran la comprensión de las reglas descubiertas al transformar los datos de bajo nivel en datos de alto nivel y también reduce significativamente el tiempo de ejecución del algoritmo de búsqueda. Su principal desventaja es que se puede reducir la exactitud del conocimiento descubierto, debido a que puede causar la pérdida de alguna información.
  Reducir el tamaño de los datos, encontrando las características más significativas dependiendo del objetivo del proceso.
  Se pueden utilizar métodos de transformación para reducir el número efectivo de variables a ser consideradas, o para encontrar otras representaciones de los datos tales como:

  - Reducción de dimensiones (la extracción irrelevante y débil de atributo), compresión de datos (reemplazando valores de datos con datos alternativos codificados).
  - Reducción de tamaño (reemplazando valores de datos con representación alternativa más pequeña)
  - Una generalización de datos (reemplazando valores de datos de niveles conceptuales bajos con niveles conceptuales más altos).

- **Minería de Datos:** Consiste en la búsqueda de los patrones de interés que pueden expresarse como un modelo o simplemente que expresen dependencia de los datos. Se tiene que especificar un criterio de preferencia para seleccionar un modelo de un conjunto de posibles modelos. También se tiene que especificar la estrategia de búsqueda a utilizar (normalmente está determinado en el algoritmo de minería).

- **Evaluación de los patrones:** Se identifica verdaderamente patrones interesantes que representan conocimiento usando diferentes técnicas incluyendo análisis estadísticos y lenguajes de consultas.

3.11.10. CRISP-DM

Para implementar una tecnología en un negocio, se requiere de una metodología. Estos métodos son definidos a partir de sus experiencias y tomando lo mejor de

los procedimientos más exitosos o populares. Para el caso de proyectos de implementación de minería de datos, hay una en particular; CRISP-DM, que describe los enfoques comunes que utilizan los expertos en minería de datos. El estándar incluye un modelo y una guía, estructurados en seis fases, algunas de estas fases son bidireccionales, lo que significa que algunas fases permitirán revisar parcial o totalmente las fases anteriores.[47]

A continuación (Fig. 3.7) se describen los pasos para la ejecución de esta metodología:

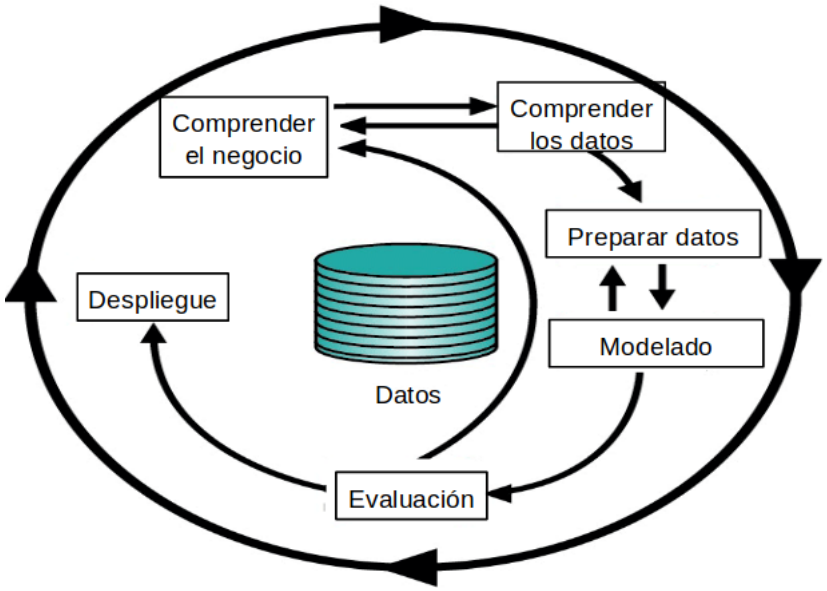

Figura 3.7: Metodología CRISP-DM.

Comprensión del negocio: (Objetivos y requerimientos desde una perspectiva no técnica)

- Establecimiento de los objetivos del negocio (Contexto inicial, objetivos, criterios de éxito).
- Evaluación de la situación (Inventario de recursos, requerimientos, supuestos, terminologías propias del negocio).
- Establecimiento de los objetivos de la minería de datos (objetivos y criterios de éxito).
- Generación del plan del proyecto (plan, herramientas, equipo y técnicas).

Comprensión de los datos: (Familiarizarse con los datos teniendo presente los objetivos del negocio)

- Recopilación inicial de datos.
- Descripción de los datos.

- Exploración de los datos.
- Verificación.

Preparación de los datos: (Obtener la vista minable o un conjunto de datos)

- Selección de los datos.
- Limpieza de datos.
- Construcción de datos.
- Integración de datos.
- Formateo de datos.

Modelado: (Aplicar las técnicas de minería de datos al conjunto de datos)

- Selección de la técnica de modelado.
- Diseño de la evaluación.
- Construcción del modelo.
- Evaluación del modelo.

Evaluación: (De los modelos de la fase anteriores para determinar si son útiles a las necesidades del negocio)

- Evaluación de resultados.
- Revisar el proceso.
- Establecimiento de los siguientes pasos o acciones.

Despliegue: (Explotar utilidad de los modelos, integrándolos en las tareas de toma de decisiones de la organización)

- Planificación de despliegue.
- Planificación de la monitorización y del mantenimiento.
- Generación de informe final.
- Revisión del proyecto.

CAPÍTULO 4

Marco Metodológico

Para lograr la realización de los objetivos planteados dentro de este trabajo en el tiempo estimado se necesitaba de una tecnología de desarrollo que permitiera una rápida adaptación a cambios de requerimientos y un desarrollo constante de manera ágil y ordenada.

Dentro de las metodologías estudiadas, XP resaltó una por los siguientes puntos:

- Es una de las metolodogías en las que se recomienda el trabajo en pareja de desarrolladores.
- Permite involucrar de manera constante al cliente con el producto.
- Se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.
- Se aplica de manera dinámica durante el ciclo de vida del software.
- Es capaz de adaptarse a los cambios de requisitos.

4.1. XP: Xtreme Programming

Esta metodología es posiblemente la mas destacada de las metodologías ágiles y esto se debe a su gran capacidad de adaptación ante cualquier tipo de imprevisto que surja. Pues la idea no es mantener ciertos requisitos desde que se está elaborando el proyecto, sino que durante el proceso, estos vayan cambiando o vayan evolucionando gradualmente sin complicaciones. Básicamente los creadores de esta metodología XP, consideran que es mejor adaptarte en el proceso a los requisitos que vayan apareciendo, que iniciar con requisitos y desarrollar un proyecto en base a eso.[48]
Los programadores trabajan en parejas y desarrollan pruebas para cada tarea antes de escribir el código. Todas las pruebas se deben ejecutar satisfactoriamente cuando el código nuevo se integra al sistema. Existe un pequeño espacio de tiempo entre las entregas del sistema.

El desarrollo incremental se lleva a través de entregas pequeñas y frecuentes del sistema y por medio de un enfoque que sirve para la descripción de requerimientos basado en las historias del clientes o escenarios que pueden ser la base para el proceso de planificación.

Esta metodología sigue una serie de pasos que deben cumplirse para el desarrollo de los objetivos que se plantean.

- **Desarrollo iterativo e incremental:** Pequeñas mejoras, unas tras otras.
- **Pruebas unitarias continuas:** Son frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- **Programación en parejas:** Se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera es más importante que la posible pérdida de productividad inmediata.
- **Frecuente integración del equipo de programación con el cliente o usuario:** Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- **Corrección de todos los errores antes de añadir nueva funcionalidad:** Hacer entregas frecuentes.
- **Refactorización del código:** Reescribir ciertas partes del código para aumentar su legibilidad y Mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- **Propiedad del código compartido:** en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- **Simplicidad del código:** es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

4.1.1. Fases de XP

La metodología XP se distribuye en las siguientes fases:

4.1.1.1. Fase 1: Planificación del proyecto

- **Historias de usuario:** El primer paso de cualquier proyecto que siga la metodología XP es definir las historias de usuario con el cliente. Las historias de usuario tienen la misma finalidad que los casos de uso pero con algunas diferencias: Constan de 3 ó 4 líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados, etc. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario. Cuando llega la hora de implementar una historia de usuario, el cliente y los desarrolladores se reúnen para concretar y detallar lo que tiene que hacer dicha historia. El tiempo de desarrollo ideal para una historia de usuario es entre 1 y 3 semanas. A continuación presentaremos una plantilla a utilizar en el desarrollo del proyecto para representar las historias de usuarios: (Tab. 4.1)

ID	Fecha	Descripción

Tabla 4.1: Plantilla para la representación de historias de usuario.

- **Planificación de publicaciones:** Después de tener ya definidas las historias de usuario es necesario crear un plan de publicaciones, en inglés Release plan", donde se indiquen las historias de usuario que se crearán para cada versión del programa y las fechas en las que se publicarán estas versiones. Un Release plan.[es] una planificación donde los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario, la prioridad con la que serán implementadas y las historias que serán implementadas en cada versión del programa. Después de un Release plan"tienen que estar claros estos cuatro factores: los objetivos que se deben cumplir (que son principalmente las historias que se deben desarrollar en cada versión), el tiempo que tardarán en desarrollarse y publicarse las versiones del programa, el número de personas que trabajarán en el desarrollo y cómo se evaluará la calidad del trabajo realizado.

- **Iteraciones:** Todo proyecto que siga la metodología X.P. se ha de dividir en iteraciones de aproximadamente 3 semanas de duración. Al comienzo de cada iteración los clientes deben seleccionar las historias de usuario definidas en el -elease planning"que serán implementadas. También se seleccionan las historias de usuario que no pasaron el test de aceptación que se realizó al terminar la iteración anterior. Estas historias de usuario son divididas en tareas de entre 1 y 3 días de duración que se asignarán a los programadores.

- **La Velocidad del Proyecto:** Es una medida que representa la rapidez con la que se desarrolla el proyecto; estimarla es muy sencillo, basta con contar el

número de historias de usuario que se pueden implementar en una iteración; de esta forma, se sabrá el cupo de historias que se pueden desarrollar en las distintas iteraciones. Usando la velocidad del proyecto controlaremos que todas las tareas se puedan desarrollar en el tiempo del que dispone la iteración. Es conveniente reevaluar esta medida cada 3 ó 4 iteraciones y si se aprecia que no es adecuada hay que negociar con el cliente un nuevo Release Plan".

- **Programación en Parejas:** La metodología X.P. aconseja la programación en parejas pues incrementa la productividad y la calidad del software desarrollado. El trabajo en pareja involucra a dos programadores trabajando en el mismo equipo; mientras uno codifica haciendo hincapié en la calidad de la función o método que está implementando, el otro analiza si ese método o función es adecuado y está bien diseñado. De esta forma se consigue un código y diseño con gran calidad.
- **Reuniones Diarias:** Es necesario que los desarrolladores se reúnan diariamente y expongan sus problemas, soluciones e ideas de forma conjunta. Las reuniones tienen que ser fluidas y todo el mundo tiene que tener voz y voto.

4.1.1.2. Fase 2: Diseño

- **Diseños Simples:** La metodología XP sugiere que hay que conseguir diseños simples y sencillos. Hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible e impleméntable que a la larga costará menos tiempo y esfuerzo desarrollar.
- **Glosarios de Términos:** Usar glosarios de términos y una correcta especificación de los nombres de métodos y clases ayudará a comprender el diseño y facilitará sus posteriores ampliaciones y la reutilización del código.
- **Riesgos:** Si surgen problemas potenciales durante el diseño, XP sugiere utilizar una pareja de desarrolladores para que investiguen y reduzcan al máximo el riesgo que supone ese problema.
- **Funcionabilidad extra:** Nunca se debe añadir funcionalidad extra al programa aunque se piense que en un futuro será utilizada. Sólo el 10 % de la misma es utilizada, lo que implica que el desarrollo de funcionalidad extra es un desperdicio de tiempo y recursos.
- **Refactorizar:** Refactorizar es mejorar y modificar la estructura y codificación de códigos ya creados sin alterar su funcionalidad. Refactorizar supone revisar de nuevo estos códigos para procurar optimizar su funcionamiento. Es muy común rehusar códigos ya creados que contienen funcionalidades que no serán usadas y diseños obsoletos.

4.1.1.3. Fase 3: Codificación

Como ya se dijo en la introducción, el cliente es una parte más del equipo de desarrollo; su presencia es indispensable en las distintas fases de XP. A la hora de codificar una historia de usuario su presencia es aún más necesaria. No olvidemos que los clientes son los que crean las historias de usuario y negocian los tiempos en los que serán implementadas. Antes del desarrollo de cada historia de usuario el cliente debe especificar detalladamente lo que ésta hará y también tendrá que estar presente cuando se realicen los test que verifiquen que la historia implementada cumple la funcionalidad especificada. La codificación debe hacerse ateniendo a estándares de codificación ya creados. Programar bajo estándares mantiene el código consistente y facilita su comprensión y escalabilidad.

4.1.1.4. Fase 4: Pruebas

Uno de los pilares de la metodología XP es el uso de test para comprobar el funcionamiento de los códigos que vayamos implementando. El uso de los test en XP es el siguiente:

- Se deben crear las aplicaciones que realizarán los test con un entorno de desarrollo específico para test.
- Hay que someter a tests las distintas clases del sistema omitiendo los métodos más triviales.
- Se deben crear los test que pasarán los códigos antes de implementarlos;en el apartado anterior se explicó la importancia de crear antes los test que el código.
- Un punto importante es crear test que no tengan ninguna dependencia del código que en un futuro evaluará.
- Como se comentó anteriormente los distintos test se deben subir al repositorio de código acompañados del código que verifican.
- Test de aceptación. Los test mencionados anteriormente sirven para evaluar las distintas tareas en las que ha sido dividida una historia de usuario.
- Al ser las distintas funcionalidades de nuestra aplicación no demasiado extensas, no se harán test que analicen partes de las mismas, sino que las pruebas se realizarán para las funcionalidades generales que debe cumplir el programa especificado en la descripción de requisitos.

Se utilizará el siguiente formato para registrar las pruebas realizadas: (Tab. 4.2)
Es posible que en algunas iteraciones no se desarrollen las cuatro actividades. Esto se debe a que determinadas historias de usuario no involucran todas las actividades durante una iteración.

Número	Descripción del caso de prueba	Resultado esperado	Resultado obtenido

Tabla 4.2: Plantilla para pruebas.

4.1.2. Ventajas y desventajas

- Ventajas
  - Programación organizada.
  - Menor taza de errores.
  - Satisfacción del programador.
- Desventajas
  - Es recomendable emplearlo solo en proyectos a corto plazo.
  - Altas comisiones en caso de fallar

CAPÍTULO 5

Marco aplicativo

Posterior a la fase investigativa de este trabajo especial de grado se procedió a la fase de desarrollo del proyecto en el cuál se contó con un equipo de desarrollo conformado por dos integrantes.

5.1. Proyecto

El presente Trabajo Especial de Grado está basado en el desarrollo de una solución web que permitirá al usuario trabajar en las últimas dos etapas del proceso KDD, los cuales consisten en aplicación de algoritmos a una vista minable y posterior presentación en forma de gráficos.

5.1.1. Selección de tecnologías

Para este trabajo especial de grado se seleccionaron las siguientes tecnologías:

- Web: jQueryUI, Meteor.js
- Ciencia de datos: Apache Hadoop (HDFS, YARN), Apache Spark (MlLib), Apache Drill.
- Integración: Node RIO [49] (Javascript y R)
- Manejo de versiones: Git (Repositorio Github)

Se seleccionó Meteor.js como framework web para el manejo del lado del servidor por facilidades como integración nativa con MongoDB, basado en Node.js, gracias al uso de una caché de MongoDB no hace falta refrescar el navegador para el pase de información, permitiéndole al usuario tener actualización de datos en tiempo real, trabajar con plantillas, permitiendo la creación de un sitio web modular, entre otros. Además posee otras bondades como las que tiene un automatizador de tareas como *Grunt.js* o *Gulp.js*, que a nivel de desarrollo permiten refrescar el navegador automáticamente con cada cambio en el código, minificar y unificar todo el codigo

CSS y todo el JS en un solo archivo dependiendo de cada tipo, entre otros.

Además, como tecnología web, se utilizó jQuery-UI para la creación de cajas al estilo *Drag and Drop* que permitieran seleccionar una vista minable para la aplicación de un algoritmo usando una caja como metáfora para el manejo tanto de entradas y salidas como de algoritmos.

Se utilizará Spark por dos principales razones, la primera es que es una herramienta basada en memoria lo cual reduce los tiempos de ejecución y la segunda es que existe una biblioteca llamada *Sparklyr* en el entorno R que permite invocar algoritmos de Spark MlLib para ser aplicados a la vista minable.

Para almacenar los proyectos, conjunto de datos y vistas intermedias, se utilizará HDFS, que permite establecer conexiones con Spark, y MongoDB para los metadatos de cada uno de los elementos en cuestión. Para administrar los recursos, YARN, ya que es el módulo por defecto de Hadoop. Apache Drill es la herramienta encargada de realizar consultas al estilo SQL a HDFS desde Meteor.js. Finalmente se implementará una infraestructura de procesamiento distribuido y paralelo, mediante un clúster con un único nodo físico.

5.2. Etapas del Proyecto

El proyecto se dividió en 5 etapas o iteraciones de acuerdo a la metodología utulizada:

- Levantamiento de requerimientos
- Desarrollo de platafoma web
- Integración con HDFS
- Integración con Drill
- Integración con R
- Integración con Spark

5.2.1. Iteración 0: Levantamiento de requerimientos

5.2.1.1. Planificación:

La planificación de esta fase es la siguiente:

- Definición de Historias de usuario.
- Diseño de estructura del proyecto.

- Estructurar los casos de uso.
- Levantar requerimientos funcionales y no funcionales.

En el siguiente cuadro se presentan las historias de usuario desarrolladas en esta iteración (Tab. 5.1):

ID	Fecha	Descripción
1	15/11/2016	Definir requerimientos iniciales. El cliente quiere una aplicación web que realice modelado y posterior visualización de resultados de conjuntos de datos de gran volumen alojado en un clúster Hadoop.
2	15/11/2016	El cliente quiere que el usuario pueda subir su propio conjunto de datos al clúster.
3	15/11/2016	El cliente quiere que el usuario cree un proyecto para manipular el conjunto de datos.
4	12/12/2016	El cliente requiere de un modelo de casos de uso para evaluar el correcto entendimiento del requerimiento.

Tabla 5.1: Historias de usuario. Iteración 0.

5.2.1.2. Diseño:

Para el diseño del sistema se utilizó un diagrama de arquitectura (Fig. 5.1) con los elementos que se utilizaron de acuerdo al trabajo de investigación realizado previamente, el cual tiene la siguiente forma:

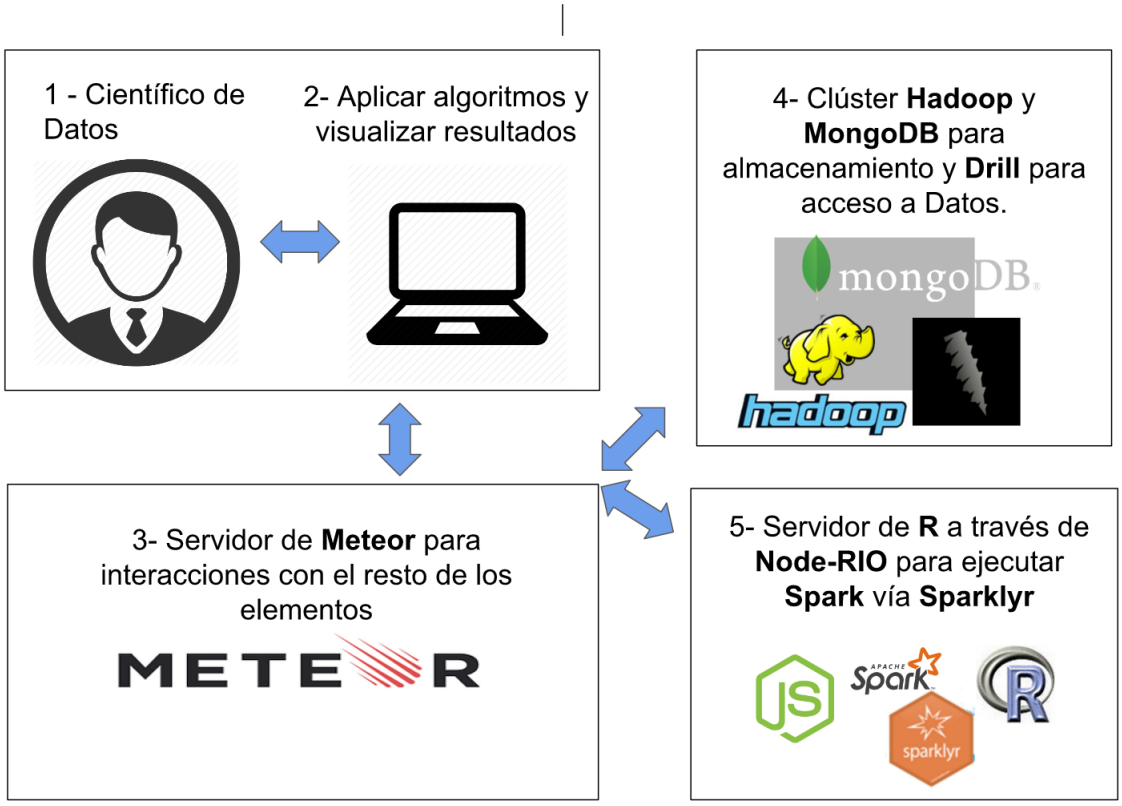

Figura 5.1: Arquitectura de la herramienta.

La siguiente estructura tiene como objetivo mostrar como se interconectan los componentes dentro de la herramienta (Fig. 5.2)

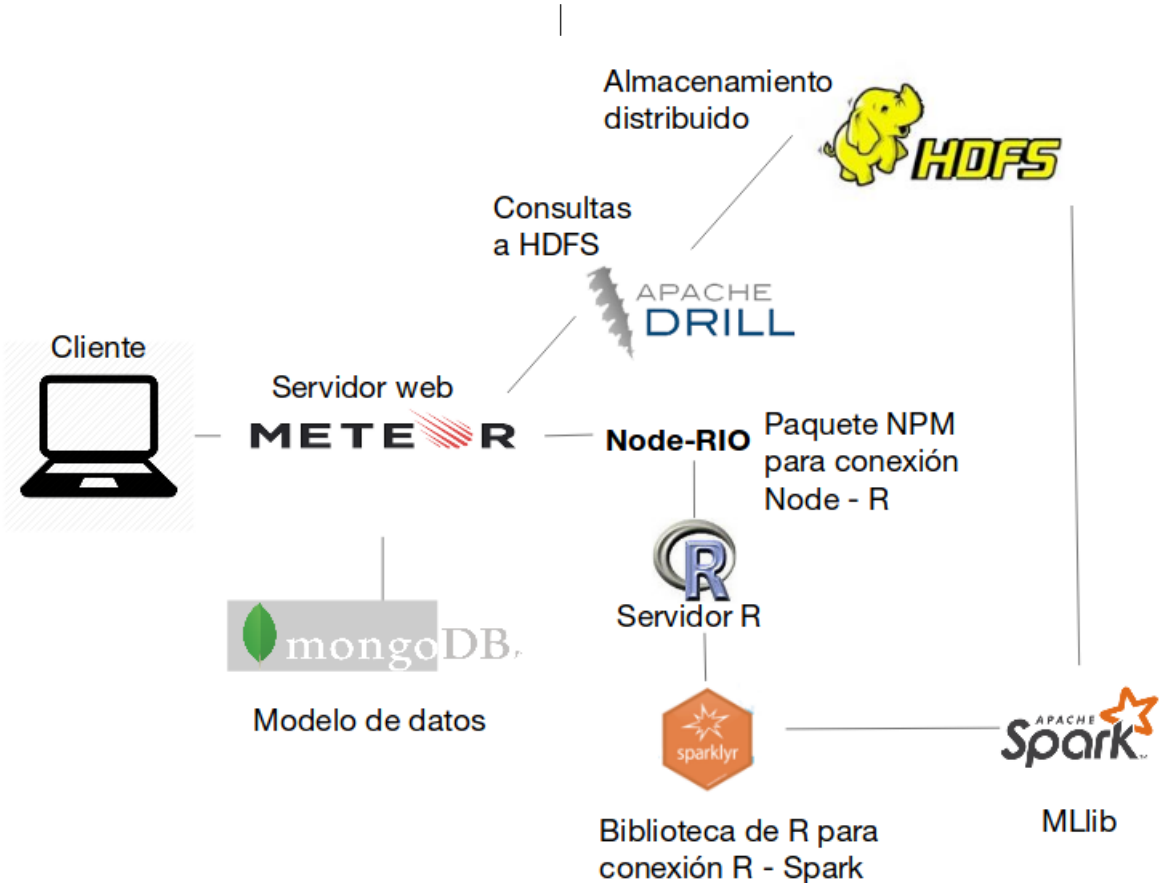

Figura 5.2: Interconexión de elementos en la herramienta.

Los diagramas de caso de uso que exponen el funcionamiento de la aplicación son los siguientes

- **Registro de usuario** (Fig. 5.3) el cual es necesario para diferenciar la pertenencia de un conjuntos de datos y/o proyectos dentro de la herramienta

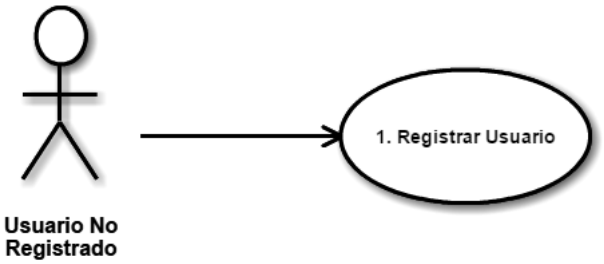

Figura 5.3: Registro de usuario.

- **Autenticación de usuario** (Fig. 5.4) ya que en todo sistema es necesario garantizar el resguardo de la información de cada usuario

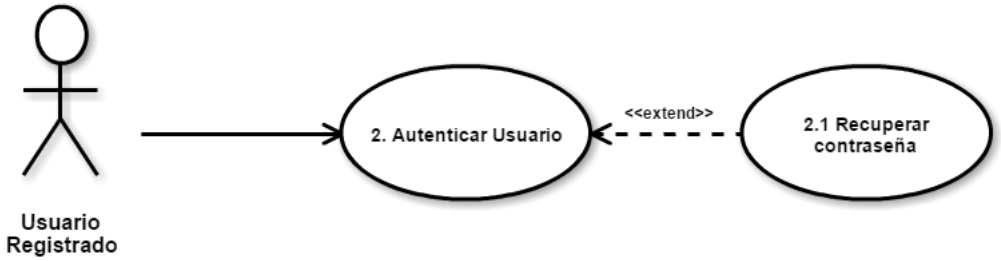

Figura 5.4: Autenticación de usuario.

- **Exploración de datos y creación de proyecto** (Fig. 5.5) que permitirá al usuario elegir una vista minable para la aplicación de algoritmos

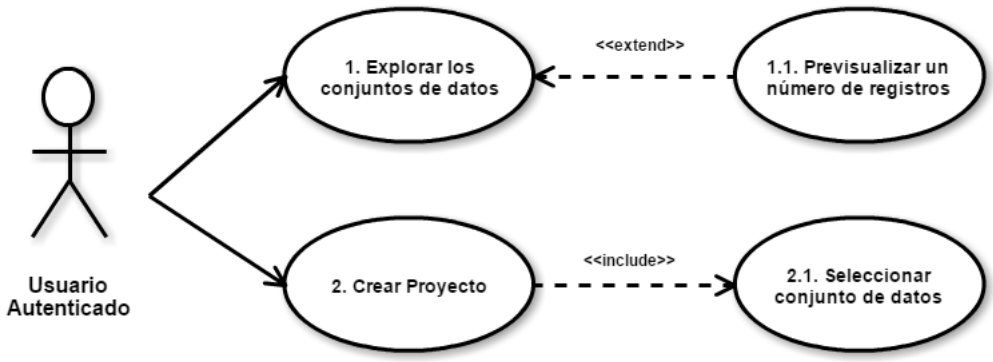

Figura 5.5: Exploración de datos y creación de proyecto.

- **Selección de algoritmo** (Fig. 5.6) que permitirá al usuario aplicar dichos algoritmo a la vista minable y permitiendo luego la visualización de resultados de forma gráfica en un paso posterior.

Figura 5.6: Selección y aplicación de algoritmo.

- **Requerimientos**
  La herramienta web debe ser capaz de conectarse con un clúster Hadoop para realizar lecturas de vistas minables alojadas en él, para luego aplicar algoritmos de minería de datos que permitan hacer un análisis gráfico posterior.

  - **Requerimientos funcionales**

    - Permitir al usuario agregar al clúster Hadoop un conjunto de datos de su preferencia.
    - Los conjuntos de datos creados pueden ser privados, es decir, opcionalmente pueden ser solo visibles a su creador.
    - Permitir al usuario crear proyectos a partir de conjuntos de datos creados previamente.
    - Ofrecer algoritmos de minería de datos en conjunto con técnicas de división en datos de prueba y datos de entrenamiento, selección de características y visualización de gráficos.

  - **Requerimientos no funcionales**

    - Notificar al usuario en caso de éxito o error a través del manejo y captura de excepciones.
    - El software debe soportar ejecución en múltiples plataformas previamente configuradas, por el hecho de estar hecho en un entorno web.
    - La aplicación debe ser usable e intuitiva con elementos que hagan el flujo de trabajo de forma natural para el usuario.
    - Las tecnologías del proyecto deben ser de licencia libre.

5.2.2. Iteración 1: Desarrollo de plataforma web

5.2.2.1. Planificación:

La planificación de esta fase es la siguiente:

- Definición de Historias de usuario.
- Preparación del entorno de desarrollo.
- Instalación de dependencias requeridas.
- Diseño de maquetas.
- Diseño de interfaces.
- Creación del modelo de datos.
- Creación de funcionalidades de usuario (Crear perfil,editar perfil).
- Creación de funcionalidades de conjuntos de datos y proyectos (Crear,editar, listar y eliminar).
- Creación de interfaz de modelado y visualización.

Para el trabajo con Meteor.js se requiere la instalación del entorno de trabajo y ciertas dependencias .
En el cuadro 5.2 se presentan las historias de usuario desarrolladas en esta iteración:

ID	Fecha	Descripción
1	18/01/2017	Generar maquetas.
2	18/01/2017	Comenzar trabajo de maquetación.
3	18/01/2017	Crear vistas funcionales para registro, autenticación y gestion de conjuntos de datos y proyectos.
4	18/01/2017	Crear modelo de datos de la aplicación.

Tabla 5.2: Historias de usuario. Iteración 1.

5.2.2.2. Diseño

Instalación de Meteor.js
Para la instalación de Meteor.js se siguió el comando de instalación ofrecido en su sitio web (*https://www.meteor.com/*) el cual se reduce a una línea que permite descargar un paquete a través de curl escribiendo en la línea de comandos:
curl https://install.meteor.com/ | sh

En conjunto con el entorno de trabajo de Meteor también se instala un motor de plantillas llamado **Blaze**, el manejador de paquetes de Node.js (NPM) y la base de datos no relacional **MongoDB**, con los que se trabajó en el proyecto.

Trabajar con Meteor.js
Una vez instalado el entorno de trabajo, basta con buscar un directorio y escribir en la terminal el comando **meteor create <nombre-proyecto>** para que se dibuje una estructura de directorio simple que gracias a la *minificación* de código (unión de todos los arhivos con la misma extensión generando un archivo único al momento del compilado para cada tipo de extensión), permitió generar una estructura personalizada acorde a las necesidades del desarrollador.

La estructura inicial es la siguiente:

- **client/main.js:** El archivo principal de entrada de Javascript del lado cliente.
- **client/main.html:** Un archivo HTML que define las plantillas con las cuales se va a trabajar.
- **client/main.css:** Un archivo CSS que define los estilos de la aplicación.
- **server/main.js:** El archivo principal de entrada de Javascript del lado servidor.
- **package.json:** Un archivo de control para instalar paquetes NPM.
- **.meteor:** Archivos internos de Meteor.
- **.gitignore:** El archivo de control por defecto de Git.

Solo basta ingresar en el directorio recién creado con el nombre del proyecto y tipear en la línea de comandos **meteor** para iniciar el servidor local que por defecto utiliza el puerto 3000.

Instalación de paquetes para el desarrollo
Para instalar paquetes dependientes de Node.js en Meteor se utiliza el comando **meteor npm install <nombre-paquete>** –save.
Algunos de los paquetes utilizados en el proyecto fueron:

- **meteor-base@1.0.4:** Paquetes base para cualquier aplicación de Meteor.js.
- **mongo@1.1.15:** La base de datos que soporta Meteor.js en esta versión
- **blaze-html-templates@1.0.4:** Compilador de archivos con extensión HTML, convirtiéndolos en vistas del motoro Blaze.
- **jquery@1.11.1:** Una de las bibliotecas mas utilizadas de JavaScript.
- **standard-minifier-css@1.3.3:** Minificador par aarchivos CSS.
- **standard-minifier-js@1.2.2:** Minificador par aarchivos JavaScript.

- **es5-shim@4.6.15:** Compilador para ECMAScript 5
- **twbs:bootstrap:** Biblioteca Bootstrap de CSS.
- **mizzao:jquery-ui:** Biblioteca que permite usar cajas arrastrables con el efecto Drag and Drop.
- **webhdfs:** API REST para consultas a HDFS con interfaz web.
- **rio:** Biblioteca que permite realizar peticiones a un servidor R (Rserve) desde Node.js a través de la integración con un servidor TCP/IP.

Etapas de desarrollo

Las etapas de desarrollo fueron las siguientes:

- **Maquetación:** Permitió realizar un trabajo de generación de elementos usables a nivel de diseño web y un análisis para decidir la forma en la que se presentan las secciones de modelado y visualización.
  Previo a comenzar el diseño de interfaces se utilizaron wireframes (maquetas de estructura) para definir la forma como se verían determinados módulos.

  La vista de conjuntos de datos fue planteada como en la imagen 5.7

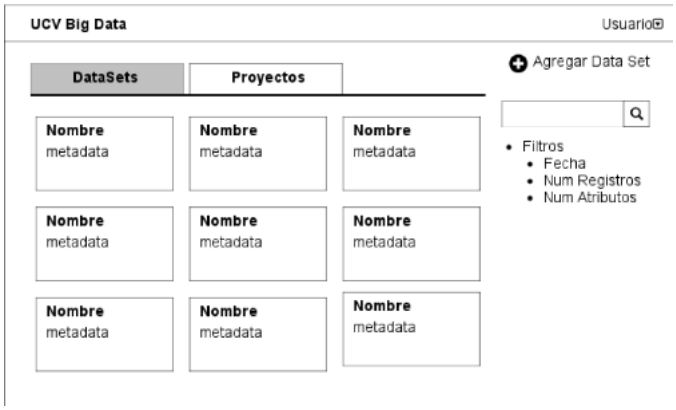

Figura 5.7: Estructura vista de conjunto de datos.

La vista de modelado de datos fue planteada como en la imagen 5.8

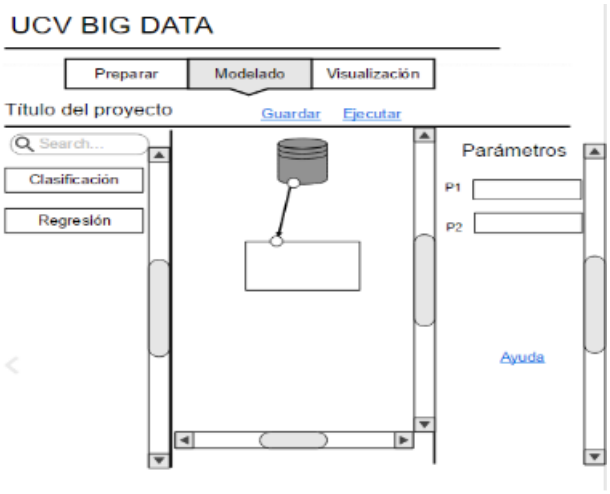

Figura 5.8: Estructura vista de modelado de datos.

Se utilizó la biblioteca Bootstrap de Twitter para realizar el maquetado de las secciones.
La primera pantalla es la del registro (Fig. 5.9) / inicio de sesión (Fig. 5.10) , que permite adicionalmente recuperar la contraseña del usuario.

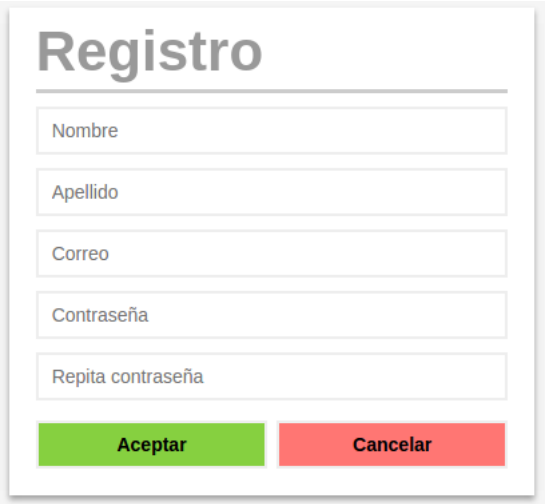

Figura 5.9: Registro de usuario.

Figura 5.10: Inicio de sesión.

Una vez autenticado, el usuario puede proceder a crear un Dataset (Conjunto de datos) (Fig. 5.11) , lo que se traduce en cargarlo desde una ubicación en HDFS para posteriormente, en una pestaña alterna, poder crear un Proyecto que es desde donde se realiza el proceso KDD.

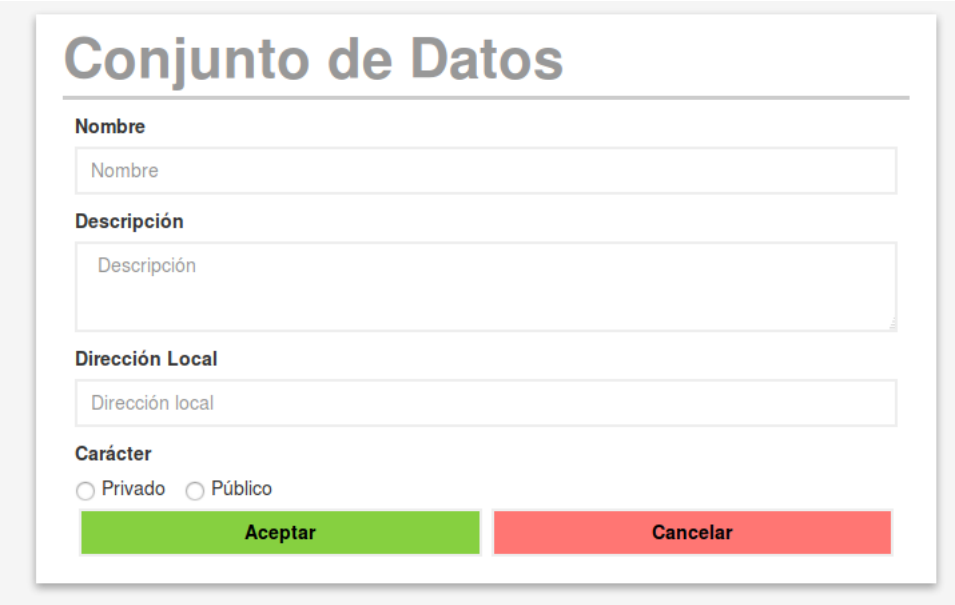

Figura 5.11: Formulario de carga de un conjunto de datos.

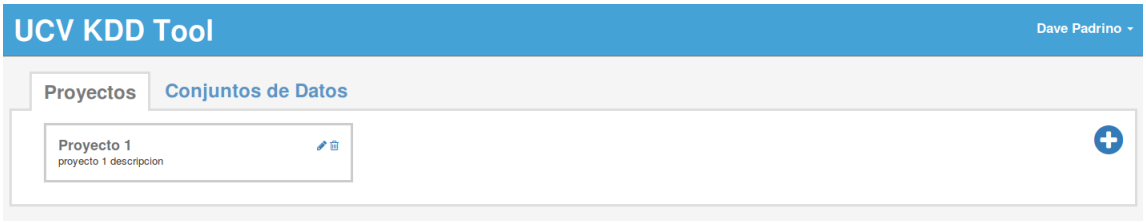

Figura 5.12: Vista inicial de un usuario autenticado, con un proyecto ya creado a partir de un conjunto de datos existente.

Para trabajar con el mencionado proceso se navega hasta una vista en donde se presenta una barra lateral con 3 fases del proceso KDD, preparación de datos, modelado de datos y visualización de datos.
La vista mas compleja de este proyecto es la de modelado (Fig. 5.13), que presenta una estructura de 3 columnas: la primera haciendo referencia a un conjunto de datos minable, algoritmos y elementos que expresados en forma de caja que permitan la aplicación de algoritmos y división del conjunto de datos en *pruebas* y *entrenamiento*, la segunda columna es el punto en donde se realizan las conexiones entre las cajas, permitiendo el pase de información y la tercera columna es aquella que permite seleccionar características de un conjunto de datos, pasar parametros a los algoritmos, activar la ejecución de la lista de cajas (Fig. 5.14) y visualizar resultados en una ventana emergente (Fig. 5.15).

Figura 5.13: Vista del modelado de datos.

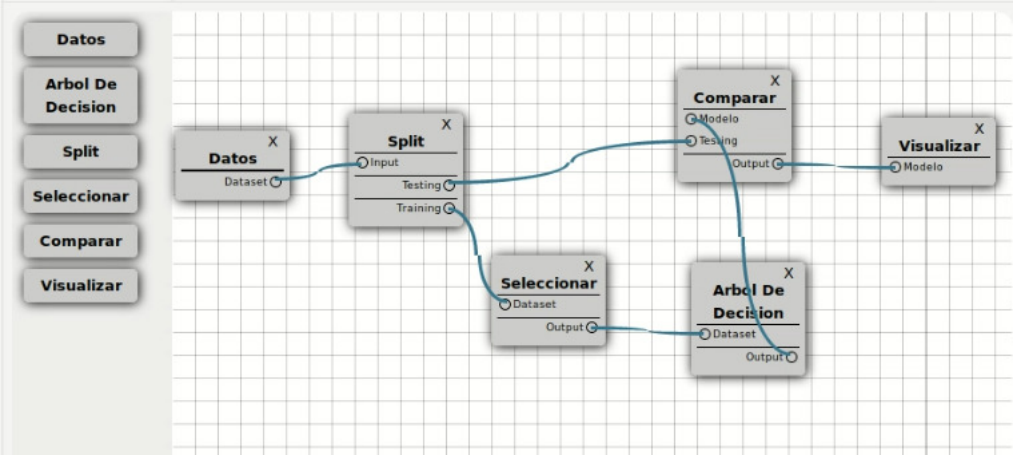

Figura 5.14: Vista del modelado de Cajas.

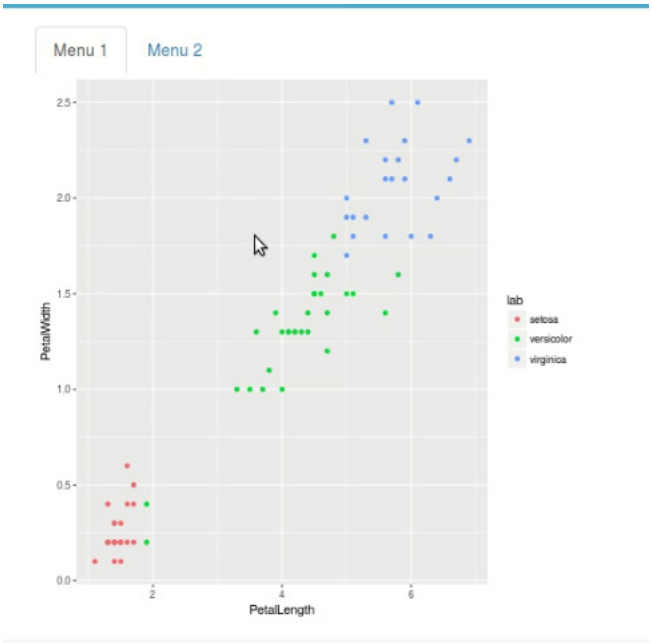

Figura 5.15: Vista del resultado del modelado.

La herramienta no está pensada para que sea adaptable a diferentes anchos de monitor, es decir, no cuenta con una característica *responsive*.

5.2.2.3. Codificación

- **Lado del cliente (Frontend):** Para el comportamiento del lado del cliente se utilizó Blaze como sistema de renderización, el cual permite construir plataformas usables y mantenibles, además que viene por defecto en la instalación de Meteor.js.

Para el comportamiento de las cajas, se utilizó una biblitoeca llamada **Node Editor** [50] que hace uso de jQuery-UI para el pase de información entre nodos. Las cajas iniciales se clonan para permitir el uso de varias instancias y cada una de ellas almacena sus datos de forma local del lado cliente en estructuras de datos que posteriormente se trabajarán del lado del servidor con R y Spark.

- **Modelo de datos MongoDB:** Se generó un modelo de datos para almacenar información de los proyectos y conjunto de datos.
  Por defecto Meteor.js genera un modelo interno para el manejo de información de usuarios por lo que no fue necesario crear uno.
  El modelo de datos se presenta en la imagen 5.16

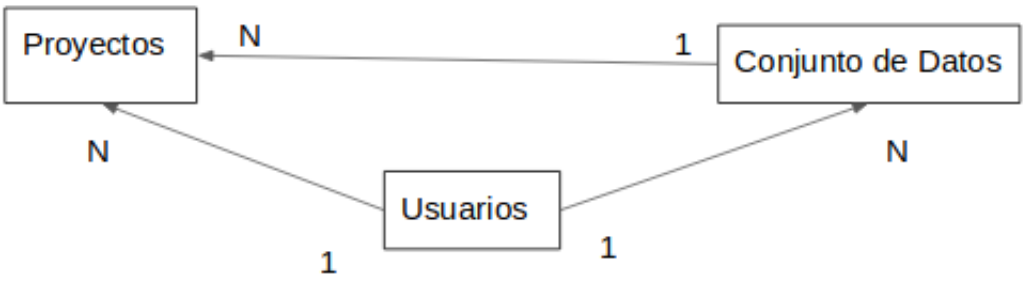

Figura 5.16: Modelo de datos.

A continuación se presentan las colecciones utilizadas para definir el modelo de datos utilizado en la aplicación:

Colección: DataSets Para la gestión de conjunto de datos.

```
DataSetsSchema = new SimpleSchema({
        name:{
                type: String,
                label:'Name'
        },
        desc:{
                type: String,
                label:'Description'
        },
        num_rows:{
                type: Number,
                label:'NumberRows'
        },
        num_fields:{
                type: Number,
```

```
                label:'NumberFields'
        },
        local_address:{
                type: String,
                label:'LocalAddress'
        },
        hdfs_address:{
                type: String,
                label:'HDFSAddress'
        },
        dataset_type:{
                type: String,
                label:'Tipo de dataset'
        },
        author:{
                type: String,
                label:'Author',
                autoValue: function(){
                        return this.userId
                },
        },
        createdAt:{
                type: Date,
                label: 'CreatedAt',
                autoValue: function(){
                        return new Date()
                },
        }
});
```

Colección: DataTypes

```
DataTypesSchema = new SimpleSchema({
        name:{
                type: String,
                label:'Name'
        },
        type:{
                type: String,
                label:'type'
        },
        active:{
                type: Boolean,
                label:'active'
        }
});
```

Colección: Projects Para la gestión de proyectos.

```
ProjectsSchema = new SimpleSchema({
        name:{
                type: String,
                label:'Name'
        },
        desc:{
                type: String,
                label:'Description'
        },
        num_rows:{
                type: Number,
                label:'Number of Rows',
                optional: true
        },
        num_fields:{
                type: Number,
                label:'Number of Fields',
                optional: true
        },
        dataset:{
                type: String,
                label:'Dataset'
        },
        address:{
                type: String,
                label:'Address in hdfs',
        },
        last_stage:{
                type: String,
                label: 'Last Stage in KDD Process'
        },
        author:{
                type: String,
                label:'Author',
                autoValue: function(){
                        return this.userId
                },
        },
        createdAt:{
                type: Date,
                label: 'CreatedAt',
                autoValue: function(){
                        return new Date()
                },
```

```
        },
        modifiedAt:{
                type: Date,
                label: 'ModifiedAt',
                optional: true
        },
        actions:{
                type: [String],
                label: 'acciones de preparacion',
                optional: true
        },
        data_types:{
                type: [DataTypesSchema],
                label: 'tipos de datos',
                optional: true
        },
        prepair_versions:{
                type: [String],
                label: 'versiones del proyecto en preparacion',
        },
        mining_view_address:{
                type: String,
                label: 'direccion de vista minable',
        },
        current_version_address:{
                type: String,
                label: 'direccion de version actual',
        },
});
```

- **Desarrollo Backend:** El backend se trabajó en JavaScript para cada sección. Trabajando en paralelo el frontend con el backend permitio ir teniendo secciones finalizadas en un menor tiempo.
  Cabe destacar que las llamadas al servidor de R y las consultas al sistema de archivos de Hadoop se hacen desde métodos declarados dentro de los archivos desde el lado del servidor.

5.2.3. Iteración 2: Integración con HDFS:

5.2.3.1. Planificación:

La planificación de esta fase es la siguiente:

- Definición de Historias de usuario.
- Preparación del entorno de desarrollo Hadoop (Pre-instalación).

- Instalación de HDFS.
- Configuración de HDFS.
- Integración de HDFS con entorno web.
- Pruebas de funcionamiento de HDFS.

Al finalizar la primera etapa de desarrollo web se procedió a integrar éste entorno con Hadoop, a través de su sistema de archivos (HDFS), para el almacenamiento proyectos de grandes volúmenes; para esto se hizo uso del servicio que ofrece Hadoop a través de una ruta dentro del sistema de archivos que es usada dentro del entorno web para la gestión de archivos. Se levantó un clúster con un único nodo para trabajar localmente en modo semi distribuido, el cual es una simulación distribuida en una sola máquina.

En el cuadro 5.3 se presentan las historias de usuario desarrolladas en esta iteración:

ID	Fecha	Descripción
1	05/02/2017	Levantar ambiente para ejecutar HDFS.
2	05/02/2017	Ejecutar pruebas en HDFS.

Tabla 5.3: Historias de usuario. Iteración 2.

5.2.3.2. Diseño

La máquina utilizada para el desarrollo cuenta con las siguientes características (Tab. 5.4):

Componente	Especificación
Sistema Operativo	Lubuntu 16.10
Memoria	3,7 GiB
Procesador	Intel Core i5-3470 CPU @ 3.20GHz x 4
OS Type	64-bit
GNOME	Versión 2.32.1-5ubun all
Disco	500 GB

Tabla 5.4: Descripción de la máquina utilizada para el desarrollo.

Previo a la instalación de HDFS se deben realizar las siguientes acciones:

Pre - Instalación de HDFS:

1- Crear un usuario:

Para comenzar, se recomienda crear un usuario separado Hadoop para aislar HDFS del sistema de archivos de Unix

```
$ su
password:
# useradd hadoop
# passwd hadoop
New passwd:
Retype new passwd:
```

2- Generación de llave SSH :
Esta generación es importante debido a que permitirá realizar acciones como inicar o detener el clúster.
Para autenticar diferentes usuarios de Hadoop es necesario proveer un par de llaves para el usuario Hadoop y compartirla con diferentes usuarios.
Copiar las llaves desde **id_rsa.pub** hacia **authorized_keys** y luego autorizar a leer y escribir al dueño en este archivo.
Los siguientes comandos son usados para generar un par de clave/valor usando SSH.

```
$ ssh-keygen -t rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

3- Instalar Java :
Java es un prerequisito para instalar Hadoop, primero se debe verificar la existencia de Java en el sistema a través del comando:

```
$ java -version
```

4- *Configuración* de variables de entorno :
En caso que no esté instalado se debe descargar y extraer en el directorio /**usr**/**local**/ para posteriormente hacer la *configuraciónn* de las variables de entorno PATH y JAVA_HOME en el archivo /**.bashrc**:

```
export JAVA_HOME=/usr/local/jdk1.XX
export PATH=$PATH:$JAVA_HOME/bin
```

Posteriormente se deben aplicar los cambios:

```
$ source ~/.bashrc
```

Instalación de HDFS:
Para la instalación de HDFS basta con descargar la distribución deseada de Hadoop y posicionarla en una ruta determinada dentro del sistema de archivos, en el caso actual se posicionó en /**usr**/**local**/.

1- *Configuración* de variables de entorno :
Posteriormente a extraer la carpeta y renombrarla como Hadoop se deben agregar los siguientes valores para configurar las variables a utilizar por Hadoop en el archivo /**.bashrc**:

```
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_INSTALL=$HADOOP_HOME
```

Luego se deben aplicar los cambios:

```
$ source ~/.bashrc
```

Para poder desarrollar elementos de Hadoop en Java se necesita configurar la variable de entorno JAVA_HOME dentro del archivo **hadoop-env.sh** con la ubicación de Java en el sistema anfitrión, es decir:

```
export JAVA_HOME=/usr/local/jdk1.XX
```

2- Configuración archivos internos de Hadoop :
Para hacer la configuración del modo semi distribuido se deben configurar tres archivos:

- **core-site.xml:** este archivo contiene información tal como el número del puerto usado para la instancia de Hadoop, la cantidad de memoria asignada para el sistema de archivos y memoria limite para almacenamiento de datos, entre otros.
  Para el caso de el proyecto en cuestion se utilizó la siguiente configuración:

```
<configuration>
    <property>
        <name>fs.default.name </name>
        <value> hdfs://localhost:9000 </value>
    </property>
</configuration>
```

- **hdfs-site.xml:** Este archivo contiene informacion tal como el valor del factor de replicación, la dirección del namenode y datanode en el sistema de archivos local, es decir, el lugar donde se desea almacenar la infraestructura de Hadoop.
  Para el caso de el proyecto en cuestion se utilizó la siguiente configuración:

```
<configuration>
        <property>
            <name>dfs.replication </name>
            <value>1</value>
        </property>
        <property>
```

```
            <name>dfs.name.dir</name>
            <value>
file:///home/hadoop/hadoopinfra/hdfs/namenode
</value>
        </property>
        <property>
            <name>dfs.data.dir</name>
              <value>
file:///home/hadoop/hadoopinfra/hdfs/datanode
</value>
         </property>
         <property>
              <name>fs.defaultFS</name>
              <value>hdfs://127.0.1.1:40010/</value>
         </property>
 </configuration>
```

- **yarn-site.xml:** Este archivo es utilizado para almacenar la configuracion del YARN en Hadoop.
  Para el caso de el proyecto en cuestion se utilizó la siguiente configuración:

```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

5.2.3.3. Pruebas

A continuación las pruebas realizadas en esta iteración

Número de Caso de Prueba: 1
Número de Historia de Usuario: 1
Descripción: Para verificar la instalación de Hadoop basta con el comando **hadoop version** el cual nos arroja un resultado como el siguiente (Fig. 5.17):

```
vit@vit-W54xEU:~$ hadoop version
Hadoop 2.7.3
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r baa91f7c6bc9cb9
2be5982de4719c1c8af91ccff
Compiled by root on 2016-08-18T01:41Z
Compiled with protoc 2.5.0
From source with checksum 2e4ce5f957ea4db193bce3734ff29ff4
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-2
.7.3.jar
vit@vit-W54xEU:~$
```

Figura 5.17: Versión de hadoop para verificar la correcta instalación.

Número de Caso de Prueba: 2
Número de Historia de Usuario: 1
Descripción: Para verificar el correcto funcionamiento de **Hadoop dfs** se utiliza el comando **start-dfs.sh** (Fig. 5.18) para iniciar los nodos, el cual nos arroja el siguiente resultado:

```
vit@vit-W54xEU:~$ start-dfs.sh
17/05/14 12:37:16 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Starting namenodes on [vit-W54xEU]
vit-W54xEU: starting namenode, logging to /usr/local/hadoop/logs/hadoop-vit-name
node-vit-W54xEU.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-vit-datan
ode-vit-W54xEU.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-vi
t-secondarynamenode-vit-W54xEU.out
17/05/14 12:37:34 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
vit@vit-W54xEU:~$
```

Figura 5.18: Iniciar datanode y namenodes de HDFS.

Número de Caso de Prueba: 3
Número de Historia de Usuario: 1
Descripción: Se puede verificar en el navegador a través de la dirección **http://localhost:8088/** el inicio del clúster (fig. 5.19).

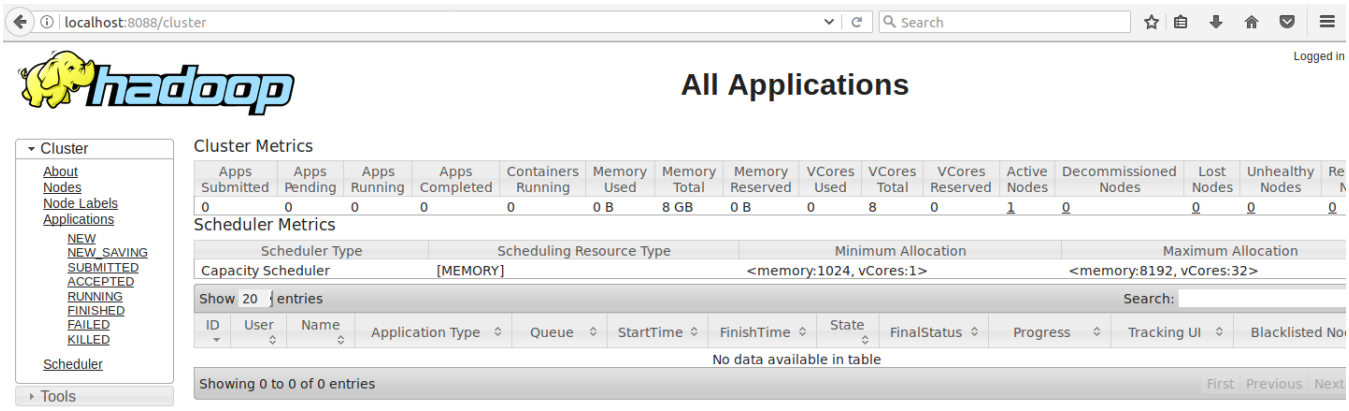

Figura 5.19: Dirección del clúster en el navegador.

Número de Caso de Prueba: 4
Número de Historia de Usuario: 1
Descripción: Se puede verificar en el navegador (Fig. 5.20) a través de la dirección **http://localhost:50070/** el inicio del sistemas de archivos (Hadoop DFS).

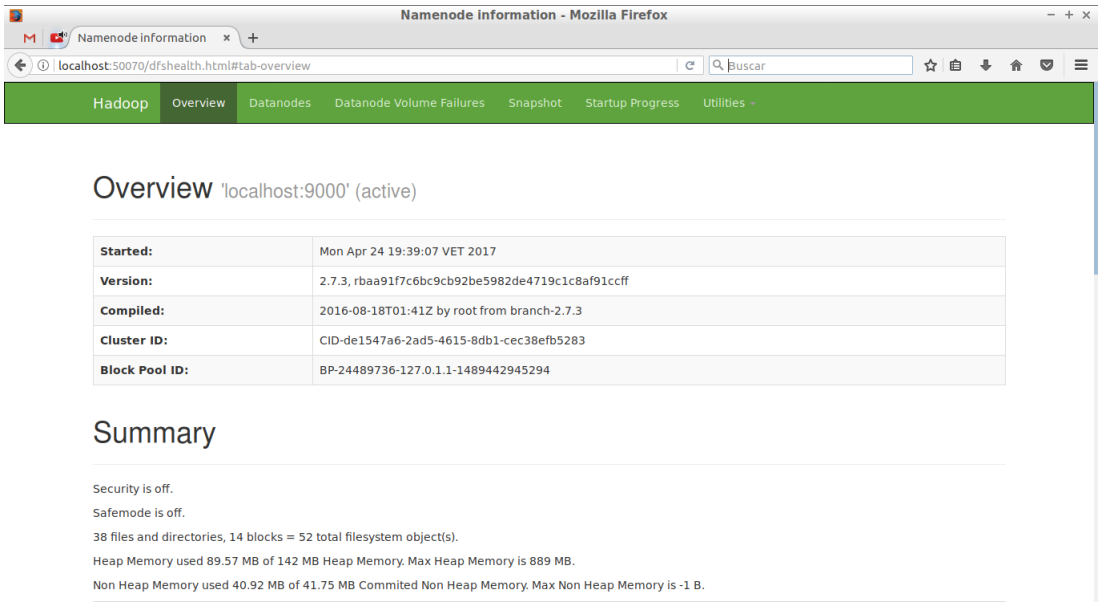

Figura 5.20: Verificar inicio de HDFS en el navegador.

Número de Caso de Prueba: 5
Número de Historia de Usuario: 1
Descripcion: Posteriormente se utiliza el comando start-yarn.sh (Fig. 5.21) para iniciar YARN, el cual nos arroja el siguiente resultado:

```
vit@vit-W54xEU:~$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-vit-resourceman
ager-vit-W54xEU.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-vit-node
manager-vit-W54xEU.out
vit@vit-W54xEU:~$
```

Figura 5.21: Iniciar nodemananger y resourcemananger de YARN.

5.2.4. Iteración 3: Integración con Drill:

La planificación de esta fase es la siguiente:

- Definición de Historias de usuario.
- Preparación del entorno de desarrollo Apache Drill (Instalación / actualización de Java).
- Instalación de Apache Drill.
- Configuración de Apache Drill.
- Integracion de Apache Drill con HDFS y entorno web.
- Pruebas de funcionamiento de Apache Drill.

Luego de tener algunos elementos dentro de HDFS se necesita una forma de acceder a estos elementos de forma sencilla, por lo que Drill es la ideal para realizar las consultas que se realizan.
Basta con realizar métodos del lado del servidor que permitan a través de métodos HTTP realizar consultas, inserciones, actualizaciones o borrado del conjunto de datos mediante código SQL.

5.2.4.1. Planificación:

En el siguiente cuadro 5.5 se presentan las historias de usuario desarrolladas en esta iteración:

ID	Fecha	Descripción
1	08/03/2017	Levantar ambiente para ejecutar Drill.
2	08/03/2017	Integrar tecnología Drill con entorno web y HDFS.

Tabla 5.5: Historias de usuario. Iteración 3.

5.2.4.2. Diseño:

Pre-Requisitos de Drill:
Para sistemas basados en Unix, sólo se requiere el JDK corriendo en el sistema.
Instalación de Drill:
Se ofrecen dos tipos de instalaciones para Drill, en modo embebido o en modo distribuido. Debido al entorno de desarrollo en una única máquina se utilizó el modo embebido para la instalación, la cual consiste en descargar la distribución correspondiente y extraer la carpeta descargada en un directorio seleccionado, en este caso se seleccionó /**usr**/**local** para mantener consistencia con el resto de los elementos con los que se trabajaron.

5.2.4.3. Pruebas:

A continuación las pruebas realizadas en esta iteración

Número de Caso de Prueba: 1
Número de Historia de Usuario: 1
Descripción: Trabajar con Drill: Para iniciar Drill basta con posicionarse en el directorio seleccionado para la extracción de la descarga y utilizar el comando **bin/drill-embeded**. (Fig. 5.22)

```
vit@vit-W54xEU:~$ cd /usr/local/apache-drill-1.10.0/
vit@vit-W54xEU:/usr/local/apache-drill-1.10.0$ sudo bin
bin/  bind
vit@vit-W54xEU:/usr/local/apache-drill-1.10.0$ sudo bin
bin/  bind
vit@vit-W54xEU:/usr/local/apache-drill-1.10.0$ sudo bin/drill-embedded
[sudo] password for vit:
OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize=512M; support was
removed in 8.0
may 17, 2017 11:13:15 AM org.glassfish.jersey.server.ApplicationHandler initialize
INFORMACIÓN: Initiating Jersey application, version Jersey: 2.8 2014-04-29 01:25:26...
apache drill 1.10.0
"a drill in the hand is better than two in the bush"
0: jdbc:drill:zk=local>
```

Figura 5.22: Iniciar Drill por consola.

Se puede verificar en el navegador a través de la dirección **http://localhost:8047/**, en donde debe realizarse la configuración a través de un JSON en al direccion **http://localhost:8047/storage/dfs** como el siguiente:

```
      {
"type": "file",
"enabled": true,
"connection": "hdfs://127.0.1.1:40010/",
"config": null,
"workspaces": {
  "root": {
  "location": "/",
  "writable": true,
  "defaultInputFormat": null
},
"tmp": {
  "location": "/tmp",
  "writable": true,
  "defaultInputFormat": null
}
},
"formats": {
  "psv": {
    "type": "text",
    "extensions": [
            "tbl"
          ],
  "delimiter": "|"
},
"csv": {
  "type": "text",
  "extensions": [
          "csv"
  ],
  "extractHeader": true,
  "delimiter": ","
},
"tsv": {
  "type": "text",
  "extensions": [
          "tsv"
  ],
  "delimiter": "\t"
},
"httpd": {
  "type": "httpd",
  "logFormat": "%h %t \"%r\" %>s %b \"%{Referer}i\"",
  "timestampFormat": null
},
```

```
            "parquet": {
                    "type": "parquet"
            },
            "json": {
              "type": "json",
              "extensions": [
                      "json"
              ]
            },
            "avro": {
              "type": "avro"
            },
            "sequencefile": {
              "type": "sequencefile",
              "extensions": [
                      "seq"
              ]
            },
            "csvh": {
              "type": "text",
              "extensions": [
                      "csvh"
              ],
              "extractHeader": true,
              "delimiter": ","
        }
    }
}
```

Número de Caso de Prueba: 2
Número de Historia de Usuario:
Descripción: Una vez iniciado se pueden hacer consultas usando SQL (Fig. 5.23) a la fuente de datos almacenada en HDFS, como se muestra a continuación (Fig. 5.24):

```
queryDataDrill: function (data_address){
    console.log('queryDataDrill');
    var queryCopy = "select * from dfs.`"+data_address+"/raw/*` limit 100";
    console.log(queryCopy);
    try {
        var result = HTTP.call("POST", "http://localhost:8047/query.json",{ data: {
                "queryType" : "SQL",
                "query" : queryCopy
            }
        });
        return result;

    } catch (e) {
        // Got a network error, time-out or HTTP error in the 400 or 500 range.
        console.log(e);
        return e;
    }
},
```

Figura 5.23: Llamada desde Meteor a Drill para realizar una consulta a HDFS.

```
0: jdbc:drill:zk=local> SELECT * FROM dfs.`/user/hadoop/datasets/iris1493583755334.csv` LIMIT 10;
+-----+-------------+------------+-------------+------------+----------+
| id  | SepalLength | SepalWidth | PetalLength | PetalWidth | Species  |
+-----+-------------+------------+-------------+------------+----------+
| 1   | 5.1         | 3.5        | 1.4         | 0.2        | setosa   |
| 2   | 4.9         | 3          | 1.4         | 0.2        | setosa   |
| 3   | 4.7         | 3.2        | 1.3         | 0.2        | setosa   |
| 4   | 4.6         | 3.1        | 1.5         | 0.2        | setosa   |
| 5   | 5           | 3.6        | 1.4         | 0.2        | setosa   |
| 6   | 5.4         | 3.9        | 1.7         | 0.4        | setosa   |
| 7   | 4.6         | 3.4        | 1.4         | 0.3        | setosa   |
| 8   | 5           | 3.4        | 1.5         | 0.2        | setosa   |
| 9   | 4.4         | 2.9        | 1.4         | 0.2        | setosa   |
| 10  | 4.9         | 3.1        | 1.5         | 0.1        | setosa   |
+-----+-------------+------------+-------------+------------+----------+
10 rows selected (0,493 seconds)
```

Figura 5.24: Consulta hecha con Drill.

5.2.5. Iteración 4: Integración con R

La planificación de esta fase es la siguiente:

- Definición de Historias de usuario.
- Preparación del entorno de desarrollo de R (Instalación R, RServe y RStudio).
- Descarga e integración de biblioteca Node-RIO.
- Integracion con entorno web.
- Pruebas de funcionamiento de Apache Drill.

Se utilizó una biblioteca llamada **Node-RIO** la cual permite la conexión con **Rserve** que es un servidor TCP/IP para R, y a través de esa conexión poder ejecutar comandos en lenguaje R y archivos con extensión **.R**.
Descargando el código desde el siguiente repositorio **https://github.com/albertosantini/node-rio** permite introducirlo manualmente en el código, o a través del comando **meteor npm install rio –save**.

El servidor puede iniciarse a través del siguiente comando:

```
require("Rserve")
Rserve()
```

En el repositorio del autor se encuentran ejemplos de como usar este paquete.

5.2.5.1. Planificación:

En el siguiente cuadro (Tab. 5.6) se presentan las historias de usuario desarrolladas en esta iteración:

ID	Fecha	Descripción
1	31/03/2017	Levantar ambiente para ejecutar el servidor de R.
2	31/03/2017	Integrar ambiente R con entorno web.

Tabla 5.6: Historias de usuario. Iteración 4.

5.2.5.2. Diseño:

Instalación de entorno R:
El entorno R para este proyecto no fue mas que R y su IDE RStudio, los cuales se instalan de la manera convencional a través de línea de comandos y descarga del sitio web de R Studio respectivamente.

Trabajar con entorno R:
Para utilizar las prestaciones de R en este proyecto se inicia el servidor (Fig. 5.25) con el siguiente comando **Rserve(args='–vanilla')**

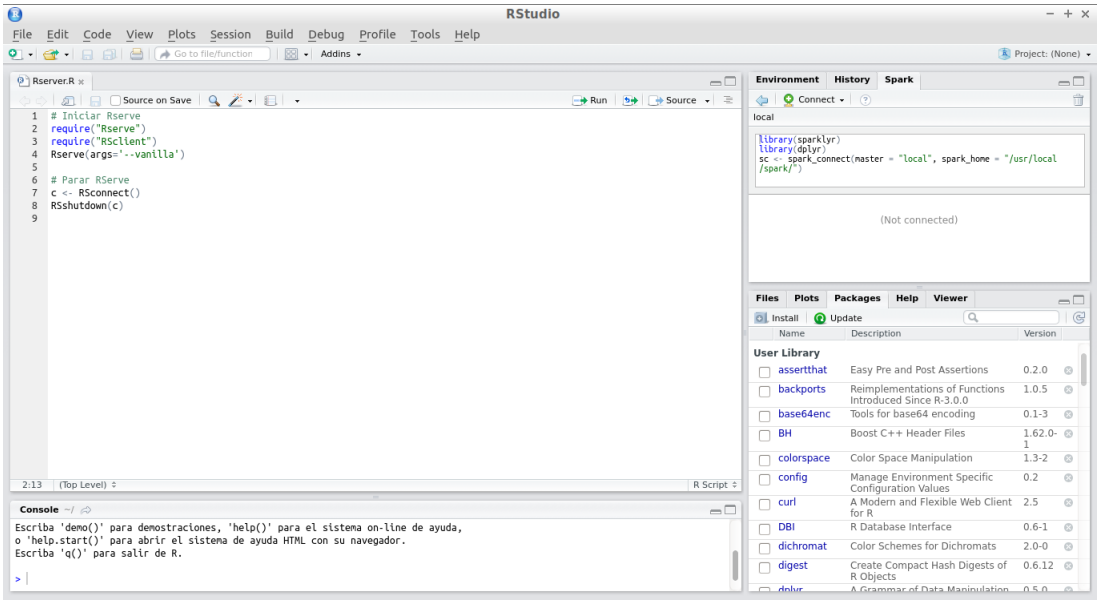

Figura 5.25: Llamada desde Rstudio para encender el servidor de R.

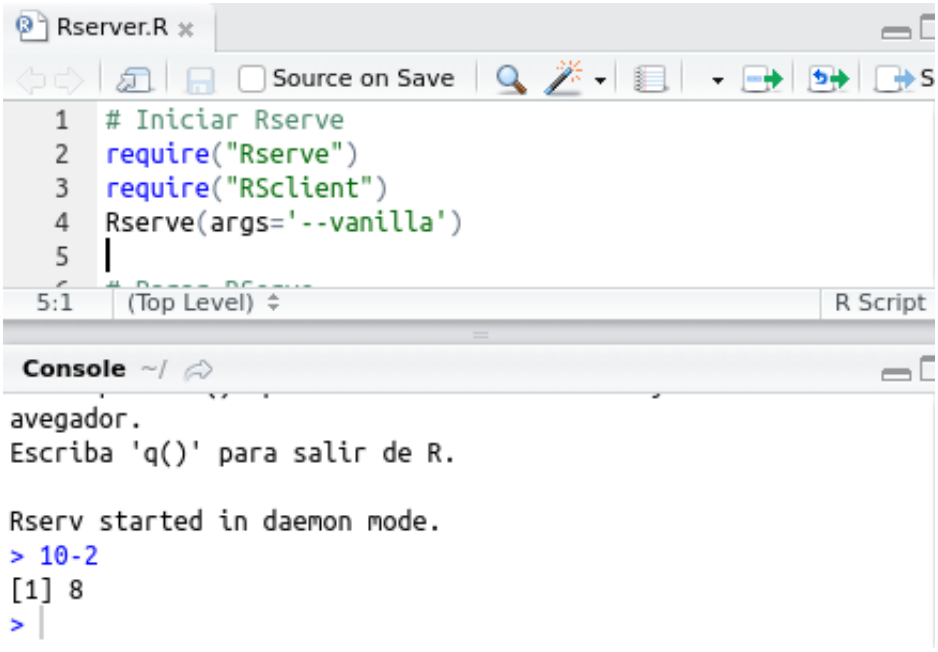

Figura 5.26: Prueba de funcionamiento del servidor de R.

5.2.5.3. Codificación:

A través de comandos de **Node-RIO** se hace llamada a un archivo con extensión .R (Fig. 5.27) que ejecutará cualquier código con los parametros pasados desde el proyecto en Meteor.

```
example10: async function(input){
        const promisedResult = rio.$e({
            filename: route + "example10.R",
            entrypoint: "test",
            data: input
        })
        .then(function (res) {
            return JSON.parse(res);
        })
        .catch(function (err) {
            return (err);
        });
        return promisedResult;
},
```

Figura 5.27: Llamada desde el servidor Meteor a R usando Node-RIO .

Siendo *filename* el nombre y la ruta del archivo, *entrypoint* el nombre de la función dentro del archivo dado y *data* los parametros que se le quieren pasar, lo mas recomendado es hacerlo en formato JSON.

5.2.5.4. Pruebas:

Las pruebas realizadas para esta iteración fueron las siguientes:(Tab. 5.7)

ID	Caso de prueba	Resultado Deseado	Resultado Obtenido
1	Iniciar Rserve con el comando respectivo.	Se espera que se inicie el servidor de R sin inconvenientes.	Correcto. (Fig. 5.25)
2	Usar comandos de JavaScript para hacer operaciones básicas de R	Se espera los comandos funcionen de manera correcta permitiendo realizar operaciones básicas en R usando JavaScript.	Al ejecutar los comandos se obtuvo respuesta positiva. (Fig. 5.26)

Tabla 5.7: Casos de Prueba R

5.2.6. Iteración 5: Integración con Spark:

La planificación de esta fase es la siguiente:

- Definición de Historias de usuario.
- Preparación del entorno de desarrollo de Apache Spark.
- Integracion con entorno R.

- Pruebas de funcionamiento de Apache Spark.

La conexión con Spark se realizó a través de una biblioteca del lenguaje R llamada **Sparklyr** que permite utilizar los algoritmos de MlLib de Spark, en conjunto con la biblioteca Dplyr. Adicionalmente esta biblioteca permite crear extensiones para invocar el API de Spark y tener acceso a sus paquetes a través de interfaces provistas por la misma.

5.2.6.1. Planificación:

En el siguiente cuadro (Tab. 5.8) se presentan las historias de usuario desarrolladas en esta iteración:

ID	Fecha	Descripción
1	12/04/2017	Levantar ambiente para ejecutar Spark.
2	12/04/2017	Integrar ambiente Spark con R y el entorno web.

Tabla 5.8: Historias de usuario. Iteración 5.

5.2.6.2. Diseño:

Pre-Instalación de Spark:
Previo a la instalación de Spark se requiere revisar lo siguiente:

1- Verificar la versión de Java
Esta verificación se hace a través del comando:

```
$ java -version
```

2- Verificar la versión de Scala
Esta verificación se hace a través del comando:

```
$ scala -version
```

Sino está instalado, se hace descargando a través de **http://www.scala-lang.org/download/** y descomprimiendo el archivo descargado en el directorio **/usr/local/** y renombrando a **scala**.
Posterior a la extracción se debe exportar la variable de entorno dentro del archivo **/.bashrc** con el comando:

```
export PATH = $PATH:/usr/local/scala/bin
```

Posteriormente se deben aplicar los cambios:

```
$ source ~/.bashrc
```

Instalación de Spark:
Al igual que las otras dependencias del entorno Apache, se debe descargar la distribución correspondiente bien sea a través de terminal o desde el sitio web oficial y luego extraer la carpeta descargada en el directorio local, el cual es /**usr**/**local**.
Una vez dentro del directorio se debe configurar el archivo **spark-env.sh** con la variable JAVA_HOME y la memoria del *WORKER* de Spark.
Adicionalmente debe agregarse la variable de entorno al archivo /**.bashrc** con el comando:

```
export PATH = $PATH:/usr/local/spark/bin
```

Posteriormente se deben aplicar los cambios:

```
$ source ~/.bashrc
```

5.2.6.3. Codificación:

Trabajar con Spark:
Al usar Spark desde R, se necesita invocar la biblioteca a través del **library(sparklyr)** para luego hacer una conexión con la biblioteca del comando **spark_connect(master = 'local', spark_home = '/usr/local/spark'** (Fig. 5.28), siendo *master* el tipo de conexión y *spark_home* la dirección del sistema de archivos locla en donde se posiciona la instalación de Spark.

```
> library(sparklyr)
> library(dplyr)

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

> sc <- spark_connect(master = "local")
>
```

Figura 5.28: Iniciar sparklyr desde R.

La siguiente imagen muestra como se utiliza el codigo sparklyr desde el archivo R (Fig. 5.29).

```
require(RJSONIO)
library(sparklyr)
library(dplyr)
library(DBI)
library(ggplot2)
library(tidyr);


test <- function (data) {

        resp = fromJSON(data)
        dataset = paste(resp[[1]]$parametros[[1]][2])
        sc <- spark_connect(master = "local",spark_home = '/usr/local/spark/')
        import_iris <- spark_read_csv(sc, name = "iris", path = dataset, header = TRUE)
```

Figura 5.29: Llamada a sparklyr desde R, para posteriormente ser leido por Meteor.js.

5.2.6.4. Pruebas:

Las pruebas realizadas para esta iteración fueron las siguientes: (Tab. 5.9)

ID	Caso de prueba	Resultado Deseado	Resultado Obtenido
1	Usar comandos R para iniciar Spark y a través de llamadas a su biblioteca sparklyr hacer operaciones basicas	Se espera los comandos funcionen de manera correcta permitiendo realizar operaciones básicas en Spark - R usando sparklyr.	Al ejecutar los comandos se obtuvo respuesta positiva. (Fig. 5.28)

Tabla 5.9: Casos de Prueba Spark - R

Todas las tecnologías usadas en este proyecto son de uso libre.

5.3. Flujo de trabajo

A continuación se explica el flujo de trabajo de la herramienta desde que lee la vista minable hasta que produce el gráfico:

- La primera caja que se agrega debe ser la vista minable.
- A medida que se van agregando cajas, se va creando un arreglo de objetos con determinadas propiedades.
- Se envía dicho arreglo al servidor de Meteor cuando se presiona “ejecutar”.

- El servidor de Meteor se conecta a través de un paquete NPM llamado Node-RIO al servidor de R (RServe) para iniciar la ejecución de un archivo con código R que recibe como parámetro un JSON de objetos donde cada objeto contiene la información de cada caja. Dicho archivo con código R ofrece todas las bibliotecas necesarias para la ejecución así como todos los algoritmos ofrecidos en la herramienta.

- Una vez conectado a RServe, se configura la conexión a Spark, en este proyecto se usa Spark de manera local, pero bien podría ser un nodo Spark en otro nodo distinto de aquel donde el servidor de R se ejecuta, esto se realiza mediante la biblioteca sparklyr. Las ventajas más significativas de ejecutar el código de esta manera es que se realiza una sola conexión a Spark (ya que la primera conexión es un poco lenta) y que los datos se guardan en Spark lo que permite manejar grandes volúmenes de datos.

- Posteriormente un ciclo se encarga de recorrer el JSON enviado desde el servidor de Meteor, y va realizando ejecuciones de acuerdo al orden en que fueron almacenadas las cajas. Este JSON contiene el orden , los parámetros y las columnas seleccionadas del conjunto de datos, en caso de que sea necesario.

- Al llegar al final de la ejecución, se genera un gráfico que se envía desde R al servidor de Meteor donde finalmente se almacena en un directorio de imágenes ubicado en el proyecto para que posteriormente, pueda ser accedido desde el cliente y mostrado desde una ventana modal.

CAPÍTULO 6

Conclusiones

Luego de un amplio trabajo de investigación en búsqueda de diferentes tecnologías que pudieran facilitar la realización del proyecto y de realizar pruebas para interconectar dichas herramientas, se realizaron pruebas con resultados positivos, pero tambien algunas pruebas fueron fallidas, lo que llevó a que el tiempo de pruebas fuese mayor durante el proyecto y que el equipo volviera a investigar en búsqueda de nuevas soluciones.
Cuando finalmente se logró encontrar y probar satisfactoriamente un conglomerado de herramientas, dichas herramientas facilitaron el cumplimiento de los objetivos específicos planteados, permitiendo así, a través de un trabajo modular y cíclico, lograr el objetivo principal teniendo una aplicación altamente escalable, con una interfaz que permite una interacción sencilla e intuitiva entre el usuario final y el sistema, que a través de optimización en algoritmos permiten que la ejecución sea bastante fluida y logre cumplir las tareas de lectura de vista minable, aplicación de algoritmos, separación de datos en pruebas y entrenamiento, visualización de resultados y gestión de perfil de usuario, proyectos y conjunto de datos (datasets) de manera simple.
Las tecnologías utilizadas permiten que la interacción sea fluida dado que procura mostrar cambios en tiempo real sin necesidad de refrescar el navegador y vuelca parte de la lógica del sistema sobre el lado cliente siendo además, una herramienta basada en eventos que son los que disparan las peticiones al servidor reduciendo así el número de interacciones innecesarias entre cliente y servidor que pudieran ralentizar el proceso.
Este proyecto pretende ser el primero de muchos que puedan a través de interfaces web usables, lograr resolver grandes problemas en poco tiempo y con pocos clicks.

6.1. Contribución

Este trabajo de investigación contribuye con la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela, debido a que ofrece varios factores innovadores dentro de esta institución y a nivel nacional por ser una solución web orientada a la ciencia de datos donde se utilizaron elementos del ambiente Hadoop (HDFS y YARN) y de la suite Apache (Drill y Spark) en conjunto

con herramientas web que son tendencias en la actualidad tales como Node.js y el motor de base de datos no relacional MongoDB.

Finalmente con el proyecto desarrollado se deja la posibilidad de implementar diversas soluciones, tales como el modelado de un conjunto de datos minable a través de la aplicación de diferentes tipos de algoritmos de aprendizaje automático (Machine Learning) provistos por la biblioteca MlLib de Apache Spark y la etapa de visualización en donde podrán realizarse distintos análisis en base a gráficos resultado de la aplicación de los algoritmos mencionados previamente.

6.2. Recomendaciones

Para ejecutar este proyecto es recomendable instalar alguna distribución Hadoop, o en su defecto instalar los componentes individualmente (HDFS, YARN), Apache Spark, Apache Drill, R y RStudio. Además se recomienda tener un ambiente de desarrollo para una aplicación basada en Node.js. Poseer un clúster con multi nodos físicos para tener un buen rendimiento, con un clúster de un solo nodo o con multi nodos virtuales es posible realizar el procesamiento, sin embargo la velocidad de ejecución podría ser mejor en el primer escenario.
Tener conocimientos del ecosistema Hadoop, programación web y entorno Linux para poder solventar cualquier problema que pueda ocurrir al momento de realizar configuraciones, implementaciones o instalaciones adicionales.
Tambien se recomienda que la máquina que ejecute el proyecto tenga prestaciones tales como un procesador superior al Intel i5 y memoria RAM superior a 6GB

6.3. Trabajos Futuros

Sobre el presente trabajo existen distintas modificaciones que pueden ser realizadas para mejorar su utilidad. En primer lugar instalar el clúster en máquinas que posean características superiores a las usadas en este proyecto (Procesador intel i5, 6Gb RAM) y añadirles más nodos tanto al clúster físico de ejecución como a la aplicación para poder contar con un número mayor de algoritmos aplicables a la vista minable con la que se vaya a trabajar.
Adicionalmente se propone un trabajo mas dedicado al área de visualización con alguna biblioteca del lado cliente como D3.js, la cual permitiría la generación de gráficos dinámicos en una vista aparte de la de modelado.
Para lo anteriormente expresado se deben poder almacenar estados intermedios de ejecuciones o grafos construidos en caso que se desee cambiar los parámetros de ejecuciones anteriores en caso de probar con diferentes parámetros o atributos.
Se propone que a futuro se implementen métodos para medir la eficiencia de un algoritmo aplicado, por ejemplo matrices de confusión o curvas ROC.

Bibliografía

[1] Definición de datos. http://definicion.de/datos/, 2016. Recuperado el 12 de Octubre de 2016.

[2] Definición de Información. http://www.definicionabc.com/tecnologia/informacion.php, 2016. Recuperado el 12 de Octubre de 2016.

[3] Sobreconceptos. http://sobreconceptos.com/conocimiento, 2016. Recuperado el 3 de Diciembre de 2016.

[4] What is Data Science? https://datascience.berkeley.edu/about/what-is-data-science/, 2016. Recuperado el 12 de Octubre de 2016.

[5] What is big data? http://searchcloudcomputing.techtarget.com/definition/big-data-Big-Data, 2016. Recuperado el 12 de Octubre de 2016.

[6] Jure Leskovec Anand Rajaraman and Jeffrey D. Ullman. *Mining of Massive Datasets*. Palo Alto, CA, 2014.

[7] Machine Learning | Stanford Online. http://online.stanford.edu/course/machine-learning, 2016. Recuperado el 3 de Diciembre de 2016.

[8] Intro to Artificial Intelligence Course and Training Online. https://www.udacity.com/course/intro-to-artificial-intelligence--cs271, 2016. Recuperado el 5 de Diciembre de 2016.

[9] What is business intelligence (BI)? http://searchdatamanagement.techtarget.com/definition/business-intelligence, 2016. Recuperado el 5 de Noviembre de 2016.

[10] What is database? http://searchsqlserver.techtarget.com/definition/database, 2016. Recuperado el 5 de Diciembre de 2016.

[11] ¿Qué es Base de datos relacional? http://searchdatacenter.techtarget.com/es/definicion/Base-de-datos-relacional, 2016. Recuperado el 5 de Diciembre de 2016.

[12] NOSQL Databases. http://nosql-database.org/, 2016. Recuperado el 5 de Diciembre de 2016.

[13] MongoDB. https://www.mongodb.com/es, 2017.

[14] Características de MongoDB. https://es.wikipedia.org/wiki/MongoDB, 2012.

[15] Lenguajes de programación. http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml, 2016. Recuperado el 7 de Noviembre de 2016.

[16] R: The R Project for Statistical Computing. https://www.r-project.org/, 2016. Recuperado el 7 de Noviembre de 2016.

[17] RStudio. https://www.rstudio.com/, 2016. Recuperado el 7 de Noviembre de 2016.

[18] What is Java? http://searchsoa.techtarget.com/definition/Java, 2016. Recuperado el 7 de Noviembre de 2016.

[19] JavaScript. https://es.wikipedia.org/wiki/JavaScript, 2016. Recuperado el 7 de Noviembre de 2016.

[20] Node.js. https://nodejs.org/en/about/, 2016. Recuperado el 7 de Noviembre de 2016.

[21] Meteor.js. https://guide.meteor.com/, 2016. Recuperado el 7 de Noviembre de 2016.

[22] What is Python? http://searchenterpriselinux.techtarget.com/definition/Python, 2016. Recuperado el 7 de Noviembre de 2016.

[23] Welcome to Apache Hadoop. http://hadoop.apache.org/, 2016. Recuperado el 21 de Noviembre de 2016.

[24] Apache Commons. https://commons.apache.org/, 2016. Recuperado el 21 de Noviembre de 2016.

[25] HDFS Architecture Guide. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html, 2016. Recuperado el 21 de Noviembre de 2016.

[26] Apache Hadoop 2.7.2 – Apache Hadoop YARN. http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html, 2016. Recuperado el 22 de Noviembre de 2016.

[27] MapReduce Tutorial. https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html, 2016. Recuperado el 22 de Noviembre de 2016.

[28] Apache Accumulo. https://accumulo.apache.org/, 2016. Recuperado el 22 de Noviembre de 2016.

[29] Apache HBase. https://hbase.apache.org/, 2016. Recuperado el 22 de Noviembre de 2016.

[30] Apache Hive TM. `https://hive.apache.org/`, 2016. Recuperado el 22 de Noviembre de 2016.

[31] Apache Storm. `http://storm.apache.org/`, 2016. Recuperado el 22 de Noviembre de 2016.

[32] Apache Mahout: Scalable machine learning and data mining. `http://mahout.apache.org/`, 2016. Recuperado el 22 de Noviembre de 2016.

[33] Apache Drill. `https://en.wikipedia.org/wiki/Apache_Drill/`, 2016. Recuperado el 22 de Noviembre de 2016.

[34] Falcon - Feed management and data processing platform. `https://falcon.apache.org/`, 2016. Recuperado el 23 de Noviembre de 2016.

[35] Welcome to Apache Flume . `https://flume.apache.org/`, 2016. Recuperado el 23 de Noviembre de 2016.

[36] Sqoop. `http://sqoop.apache.org/`, 2016. Recuperado el 23 de Noviembre de 2016.

[37] Ambari. `https://ambari.apache.org/`, 2016. Recuperado el 23 de Noviembre de 2016.

[38] Apache ZooKeeper. `http://zookeeper.apache.org`, 2016. Recuperado el 23 de Noviembre de 2016.

[39] Apache Oozie Workflow Scheduler for Hadoop. `http://oozie.apache.org/`, 2016. Recuperado el 23 de Noviembre de 2016.

[40] Knox Gateway – REST API Gateway for the Apache Hadoop Ecosystem. `https://knox.apache.org/`, 2016. Recuperado el 23 de Noviembre de 2016.

[41] Apache Ranger - Introduction. `http://ranger.apache.org/`, 2016. Recuperado el 23 de Noviembre de 2016.

[42] Mike Frampton. *Mastering Apache Spark*. Packt, 2016.

[43] The modern platform for data management and analytics - Cloudera. `https://www.cloudera.com/`, 2016. Recuperado el 8 de Diciembre de 2016.

[44] MapR: Converged Data Platform. `https://www.mapr.com/`, 2016. Recuperado el 8 de Diciembre de 2016.

[45] D3.js - Data-Driven Documents. `https://d3js.org/`, 2017.

[46] KDD: Extracción de conocimiento en Bases de datos. `http://mineriadatos1.blogspot.com/2013/06/descubrimiento-del-conocimiento-kdd-el.html`, 2016. Recuperado el 8 de Diciembre de 2016.

[47] Crisp-DM: una metodología para proyectos de Mineria de Datos. https://anibalgoicochea.com/2009/08/11/crisp-dm-una-metodologia-para-proyectos-de-mineria-de-datos/, 2016. Recuperado el 8 de Diciembre de 2016.

[48] XP. http://blogs.unellez.edu.ve/dsilva/files/2014/07/Metodologia-XP.pdf,https://okhosting.com/blog/metodologias-del-desarrollo-de-software/, http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm, 2017. Recuperado el 10 de Febrero de 2017.

[49] RIO. https://github.com/albertosantini/node-rio, 2016.

[50] jQuery Node Editor. https://github.com/MalcolmDwyer/jquery-ui-nodeEditor, 2012.

CAPÍTULO 7

Apéndices

Manual Técnico

7.0.1. ¿Cómo ejecutar la aplicación?

1) Asumiendo que esta instalado Meteor con sus dependencias, el entorno Apache (Spark y Drill), RStudio y el entorno Hadoop (HDFS, YARN), se deben seguir los siguientes pasos para realizar el encendido de los servicios y configuraciones previas:

- **Iniciar HDFS y YARN:**
  Para iniciar los nodos se utiliza el comando:

  ```
  start-dfs.sh
  ```

  Para iniciar YARN el comando:

  ```
  start-yarn.sh
  ```

- **Iniciar y configurar Drill:**
  Dentro del directorio de instalación se utiliza el siguiente comando con permisos de administrador:

  ```
  sudo bin/drill-embeded
  ```

  Por ejemplo (**user@machine:**/usr/local/drill/$ sudo bin/drill-embeded).
  Para la configuración de Drill, se debe ingresar un JSON en la direccion web de Drill como se referencia en la sección 5.2.4.3.

- **Iniciar R Serve:**
  Se necesita iniciar RStudio para realizar el encendido del servidor como se expresa en la sección 5.26.

- **Iniciar Meteor:**
  Finalmente debe posicionarse en la carpeta raíz del directorio y ejecutar en la línea de comandos:

```
meteor
```

El comando anterior permite instalar la herramienta para ejecución local en conjunto con las dependencias que se encuentren en el archivo **package.json**.

2) Para instalar las dependencias que hagan falta, se utiliza a través de la línea de comandos la siguiente llamada:

```
meteor npm install <nombre-módulo> --save
```

7.0.2. ¿Cómo agregar nodos a la aplicación?

Antes de hacer el ejercicio de agregar un nuevo nodo, es importante tener conocimiento de los siguientes campos que componen a cada nodo:

label (Obligatorio)

String. Hace referencia al nombre que aparece en el nodo desde pantalla.

inputs

Arreglo. Hace referencia a las entradas del nodo si las tiene. Se maneja como un arreglo de objetos con campos **id** como identificador unívoco de la entrada y **label** para utilizar un nombre que aparecerá en el campo de entrada cuando el nodo esté en el espacio de trabajo.

type (Obligatorio)

String. Hace referencia a los tipos de nodos, existen 4: **dataset** para el conjunto de datos que se leerá desde HDFS, **algoritmoCS** para algoritmos con seleccion de características, **algoritmoSS** para algoritmos sin seleccion de características y **algoritmoML** que se comporta idéntico que algoritmoCS con la diferencia que permite la selección de características objetivo para la predicción.

parámetros

Arreglo. Este campo ayuda a almacenar a aquellos elementos que sirvan de parámetros para determinado algoritmo.
Cuenta con los campos **name** para el nombre del parámetro, **value** para agregar un valor por defecto (que podrá ser cambiado haciendo click sobre el nodo y editando el campo en la columna de la derecha), este campo también puede ser un arreglo de objetos con los campos **name** para el nombre de la opcion y **selected** para indicar con 1 que el valor es seleccionado por defecto (y con 0 que no).
Adicionalmente a existe un tercer campo **type**, que básicamente va a definir el tipo de dato de cada parámetros, la inclusión de este campo es por temas de validación.

target

Arreglo. Hace referencia a aquellas características (Features) que serán objeto de predicción. Sólo disponible en el tipo **algoritmoML**.

output

Arreglo. Este campo cuenta con **label** que es el nombre que aparecerá en el nodo en caso de tener salida y **fn** el cual es una función que permite la transmisión de los datos desde la salida hasta la entrada de un nodo apenas se conecte un cable.

properties

Arreglo. Hace referencia a las propiedades, características o columnas que se seleccionarán del dataset. Solo se aplica para algoritmos que requieran selección de características, va ligado directamente al campo "type"(Solo con **algoritmoSC**).

1)

Abrir el archivo **node.js** ubicado en /**client**/**modeling**, el cual presenta una estructura sencilla basada en una variable que se exporta.

```
module.exports.nodes =
[
    {
    label: 'arbol de decision',
    type : 'algoritmoML',
    inputs: [
        {
            id: 'dataset',
            label: 'Entrada',
        }
    ],
    parametros : [
        {
            name: 'max.bins',
            value : '32L',
            type: 'text'
        },
        {
            name: 'max.depth',
            value : '5L',
            type: 'text'
        },
        {
```

```
                    name: 'type',
                    value : [
                        {
                            name:"auto",
                            selected: 1,

                        },
                        {
                            name:"regression",
                            selected: 0,

                        },
                        {
                            name:"classification",
                            selected: 0,

                        },
                    ],
                    type:''
                }
            ],
            properties: [],
            target: [],
            outputs: [
                {
                    label: 'Salida',
                }

            ]
        },
    ]
```

En el ejemplo anterior tenemos un nodo llamado **Arbol de decisión** que tiene una **Entrada**, un tipo **algoritmoML** por lo que tiene campo **properties** y **target** en vacío para posteriormente ser rellenado con las características seleccionadas por el usuario.
Adicionalmente sus parámetros son **max.bins** con un valor tipo string de '32L', **max.bins** con un valor string '5L' y **type** que es un arreglo de opciones, en donde la opcion 'auto' es la seleccionada por defecto.
En la **Salida**, apenas se conecte a una entrada de otro nodo se hará el pase de información en un JSON.

2)

Una vez se tenga el nodo en pantalla, se procede a hacer el trabajo de backend en el cual se hace llamada a un archivo con extensión **.R** que sirve de índice para hacer llamadas a determinadas funciones dependiendo del tipo de algoritmo que se vaya a ejecutar.

```
example: async function(input){
        const promisedResult = rio.$e({
            filename: route + "example.R",
        entrypoint: "test",
        data: input
    })
    .then(function (res) {
            return JSON.parse(res);
    })
    .catch(function (err) {
            return (err);
    });
    return promisedResult;
}
```

El código anterior es un ejemplo de código del lado servidor en la ruta **/lib/api_calls.js** donde se invoca a R cuando se ejecutan las conexiones del espacio de trabajo, siendo **filename** la ruta al archivo **.R** (incluyendo al archivo) que se ejecutará (en nuestro caso se ubican en **/r_code/**), **entrypoint** la función en específico que llamará dentro del archivo dado y **data** un JSON que se pasará desde JavaScript hacia R, que en nuestro caso, para no realizar múltiples conexiones a R, se envía un arreglo con el orden de ejecución de las cajas y ese será el orden de ejecución del algoritmo para que al final arroje un resultado en forma de gráfico.

3)

Escribir una funcion dentro del archivo .R que fue invocado (en el ejemplo .example.R"), la cual tendrá la ejecución del algoritmo desde Spark cuyo nodo fue creado con anterioridad. Este archivo funciona como indice, y permite ir validando la existencia de determinados nodos dentro de un ciclo alrededor del JSON que se construye a partir de las cajas.

```
#1
library(sparklyr)
library(dplyr)
library(ggplot2)
require(RJSONIO)
```

```
#2
Sys.setenv("JAVA_HOME" = "/usr/local/jdk1.8.0_131")


#3
test <- function (data) {
        #3.a
        resp = fromJSON(data)

    #3.b
        sc <- spark_connect(master = "local")
        ruta.base = "hdfs://127.0.1.1:40010"

    #3.c
        for(i in resp ) {
                        (...)
      #3.d
          if(i$label == "arbol de decision"){
            features = i$properties
            model_data <- tidy_data %>% ml_decision_tree (response="Species", featu
          }
                (...)
        }
}
```

En el ejemplo anterior ocurre lo siguiente:

1. Carga de bibliotecas necesarias y previamente instaladas de R.
2. Se llama la ubicación de JAVA para la ejecución de Spark.
3. Comienzo de la función llamada desde el servidor de Javascript. Esta función sirve como índice para la ejecución secuencial de los elementos en el JSON recibido.

    a) Se inicializa una variable con el JSON recibido

    b) Se realiza la conexión a SPARK y se hace la inicialización de la ruta que se usará para las consultas.

    c) Se transforma la variable del punto 3.a) en un JSON de R y se escribe en un JSON llamado 'example10.R' que posteriormente será leido para realizar la ejecución.

    d) Inicio del ciclo que recorrerá el JSON

    e) Tanto antes como despues de este punto se encuentran validaciones sobre el nombre de los elementos del JSON, específicamente en su atributo 'label', el cual definirá cuál elemento se ejecutará primero y cual despues,

en el caso del ejemplo, se valida si es 'arbol de decision' para luego aplicar algoritmos de Machine Learning de la biblioteca Splarklyr de R.