Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Computación Paralela y Distribuída

DESARROLLO DE MÓDULOS DE EXTRACCIÓN Y PREPARACIÓN DE DATOS ALOJADOS EN UN CLÚSTER HADOOP MEDIANTE UNA APLICACIÓN WEB PARA LA GENERACIÓN DE UNA VISTA MINABLE

Trabajo Especial de Grado presentado ante la Ilustre Universidad Central de Venezuela por los

Br. Daniel Córcega
CI. 19.453.592
Br. Juan David Piza
CI. 18.911.785
Tutor: Jesús Lares
Tutora: Haydemar Nuñez

Caracas, Venezuela mayo 2017

Universidad Central de Venezuela.
Facultad de Ciencias
Escuela de Computación

ACTA

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado con el título "**DESARROLLO DE MODULOS DE EXTRACCIÓN Y PREPARACIÓN DE DATOS ALOJADOS EN UN CLÚSTER HADOOP MEDIANTE UNA APLICACIÓN WEB PARA LA GENERACIÓN DE UNA VISTA MINABLE**", el cual es presentado por el Br. Daniel Córcega, de Cédula de Identidad 19.453.592 y el Br. Juan Piza, de Cédula de Identidad 18.911.785, a los fines de optar al título de Licenciado en Computación, dejamos en constancia lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día 19 de Mayo de 2017, a las 9:00 am, para que sus autores lo defendieran en forma pública en la Sala 1 de la Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela, lo cual se realizó mediante una exposición oral de su contenido, y luego respondieron satisfactoriamente a las preguntas que les fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo con una nota de 20 puntos.

En fe de lo cual se levanta la presente acta, en Caracas a los 19 días del mes de Mayo del año 2017.

Prof. Jesus Lares
Tutor

Profa. Haydemar Nuñez
Tutora

Prof. Hector Navarro
Jurado

Profa. Ana Leguizamo
Jurado

RESUMEN

Título: Desarrollo de módulos de extracción y preparación de datos alojados en un clúster hadoop mediante una aplicación web para la generación de una vista minable.
Autores: Daniel Córcega y Juan David Piza
Tutores: Haydemar Núñez y Jesús Lares

En la actualidad se genera una gran cantidad de datos provenientes de aplicaciones y equipos, estos datos por sus dimensiones se hacen imposibles de manejar con los sistemas tradiciones y por tanto implica un problema de grandes volúmenes de datos (Big Data) donde se debe resolver el tema de procesamiento y almacenamiento de estos.

El valor de todos los datos mencionados anteriormente está en su contenido, por tanto estos deben ser analizados para obtener conocimiento, para realizar un análisis de datos se utiliza un proceso llamado KDD (Knowledge Discovery in Databases), este proceso establece los pasos para encontrar conocimiento a partir de repositorios de datos.

La escuela de computación actualmente no cuenta con una herramienta capaz de realizar el proceso mencionado anteriormente de forma centralizada, por lo tanto este trabajo especial de grado se dedicó a la construcción de una aplicación web capaz de realizar el procesamiento de conjuntos de datos almacenados en un clúster hadoop de forma centralizada, cubriendo la etapa de extracción y preparación de los datos.

Palabras Claves: Big Data, KDD, Web, Haddop, Drill, NodeJs, MeteorJs, JavaScript, MongoDB, NoSQL, REST, API.

DEDICATORIA

Dedicado a Yajaira, Celeste, Leopoldo, Moisés y Paola.

Daniel.

Dedicado a Marianne, Nora, Julia y Karina.

Juan David.

Agradecimientos

Primero agradecer a Dios, y nuestras familias por el apoyo.

Queremos agradecer a nuestros tutores Haydemar Nuñez y Jesús Lares por el apoyo durante la realización de todo este proyecto, un especial agradecimiento a los profesores(as) Hector Navarro, Vanesa Leguizamo y José Sosa por su apoyo y formación académica durante el proceso de este trabajo.

A la Universidad Central de Venezuela, a la Escuela de Computación y a todos y cada uno de los profesores que contribuyeron en nuestra formación académica.

Por separado:

Gracias a mi familia quienes siempre han estado detrás de mi apoyándome, especialmente a mi Abuela Yayi que desde que tengo memoria ha sido el pilar de esta familia y el motivo para seguir adelante.

Gracias a Juan Piza por ser un excelente compañero y amigo.

Gracias a mis amigos de toda la vida Francisco Roldan, Gabriel Noda, Juan Bautista Salazar, Angélica Blanco, Isis Montero, Javier Rey, Ayleen Núñez, Valentina Yaquer y Sadhanna Rodriguez.

Gracias a mis amigos y compañeros de carrera Alejandra Matos, Andreina Da Silva, Rafael Dominguez, Jorge Simoes, Leonardo Testa.

Gracias a Victoria Armenta y Reinid Valarino por ser un apoyo incondicional y una inspiración.

Daniel.

Gracias a mi familia los que están presentes y los que están en alma.

Gracias a mi compañero de Tesis y de Carrera Daniel Córcega.

Gracias a mis compañeros de universidad Andreina Da Silva, Alejandra Matos, Rafael Domínguez, Andrés Morales, Carlos Abreu, Oscar Valecillos. Por formar parte de este desarrollo como persona y como profesional.

Gracias a los profesores Mercy Ospina, Nestor Mendez, Franky Uzcategui y Ana Morales por formar profesionales altamente necesarios para la construcción del país.

Gracias a mis amigos de infancia Ricardo Toro, David Fuentes, Javier Fuentes, Arturo Pineda, Cristofer Sanchez, Víctor Fernandes, Keynneth Falcón, David Sandoval y Wilmer Vizcaya por todas las vivencias a su lado.

Juan David.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Las ciencias de la computación siempre han estado en la búsqueda de conocimiento a partir de los datos que han sido almacenados u obtenidos de distintas fuentes, para esto se han utilizado métodos científicos que han permitido el descubrimiento de diferentes fenómenos, uno de estos métodos es el proceso de descubrimiento de conocimiento a partir de bases de datos, conocido como KDD por sus siglas en inglés, donde divide en una serie de pasos secuenciales la obtención del conocimiento. Anteriormente la cantidad de datos almacenados podían ser manipulados de manera sencilla ya que no implicaban problemas de procesamiento ni almacenamiento de estos, pero hoy en día existe una alta generación de datos que son valiosos para cualquier análisis que se requiera realizar, de allí surgen las tecnologías de Big Data capaces de realizar la manipulación de todos estos datos.

Actualmente se requiere el análisis de estos grandes volúmenes de datos que permitan la generación de conocimiento a partir de estos, es por esto que en este trabajo especial de grado se plantea la construcción de una aplicación web capaz de manipular estas grandes cantidades de datos, enmarcada dentro del proceso anteriormente expuesto como KDD, partiendo de la extracción de los datos, realizando la preparación de los mismos hasta llegar a la obtención de una vista minable. En el capítulo 2 se encuentra detallado el planteamiento del problema, los capítulos 3 y 4 presentan el marco conceptual y las herramientas utilizadas dentro del proyecto, el capítulo 5 presenta el marco metodológico donde se explica la metodología

de desarrollo utilizada, el capítulo 6 detalla paso a paso el desarrollo de la aplicación y por último en el capitulo 7 las pruebas realizadas a la aplicación.

Capítulo 2

Planteamiento del Problema

2.1. Descripción del Problema

Actualmente existen diferentes herramientas que permiten realizar puntualmente cada uno de los procesos de análisis de datos cuando se trabaja con grandes volúmenes de datos. Estas herramientas de análisis de datos apoyan al analista durante los procesos de extracción, transformación y algunas permiten la aplicación de ciertos algoritmos a la vista minable generada. Estos algoritmos en su mayoría deben ser programados por el analista de datos, donde además debe hacer uso de una herramienta adicional que permita interpretar de manera gráfica los datos obtenidos, luego del proceso de modelado de estos.

Todo este procedimiento implica además del uso de las diferentes herramientas puntuales por proceso, el establecimiento del formato de los datos, que posiblemente debe ser cambiado constantemente entre las entradas y salidas de las herramientas y a su vez el analista debe ser capaz de manipular estos datos para evitar problemas de compatibilidad de formatos.

El uso de estas distintas herramientas implica que para llevar a cabo todo el proceso de análisis deba invertirse una cantidad de tiempo adicional tratando de comunicar de manera eficiente dichas herramientas, así como el tiempo de programación que debe invertirse en la aplicación de los algoritmos para realizar el proceso de minería y modelado de datos. Adicional

a esto existe el reto de que la cantidad de datos que se obtengan para realizar dicho análisis sea de grandes volúmenes de datos, donde no cualquier aplicación utilizada para realizar los procesos de KDD está en capacidades de realizar la manipulación de esta cantidad de datos.

La mayoría de las herramientas utilizadas para realizar todos estos procesos de manera conjunta son privativas y requieren de una licencia para su uso, dejando de lado la parte pedagógica, académica y de capacitación de analistas de datos.

Debido a la situación expuesta anteriormente surge la siguiente interrogante: ¿Cómo se puede desarrollar una aplicación web que permita a los estudiantes de la escuela de computación de la Universidad Central de Venezuela tener una herramienta centralizada capaz de cubrir las necesidades de realizar cada uno de los pasos enmarcados en el proceso de análisis de datos KDD?.

2.2. Justificación

KDD (Knowledge Discovery in Databases) se refiere al proceso no-trivial de descubrir conocimiento e información potencialmente útil dentro de los datos contenidos en algún repositorio de datos. No es un proceso automático, es iterativo, explora exhaustivamente grandes volúmenes de datos para determinar relaciones. Este proceso extrae información de calidad que puede usarse para dibujar conclusiones basadas en relaciones o modelos dentro de los datos.

Actualmente lo estudiantes de la escuela de computación de la Universidad Central de Venezuela no cuentan con una herramienta que permita realizar un análisis de datos basados en el proceso KDD de forma centralizada. Para cumplir con sus actividades académicas deben emplear diferentes herramientas por cada una de las tareas de dicho proceso, lo cual hace que el aprendizaje se vea afectado por el tiempo empleado en conocer y utilizar cada una de las herramientas.

A través del desarrollo de este proyecto se cubre la necesidad de los estudiantes en tener una herramienta centralizada con la cual puedan acceder a diferentes conjuntos de datos alojados en un clúster hadoop, preparar los datos contenidos, aplicar diferentes algoritmos de minería de datos, generar gráficos y obtener conocimiento de estos.

Por consiguiente los estudiantes podrán enfocarse directamente en el análisis de datos sin la necesidad de invertir tiempo en la investigación de herramientas puntuales que los apoyen en cada tarea del proceso antes descrito.

2.3. Alcance

El desarrollo se realizará enmarcado dentro del proceso Knowledge Discovery in Databases (KDD), puntualmente establecido en las etapas de selección, pre procesamiento y transformación de los datos obtenidos del clúster Hadoop.
La aplicación web a desarrollar partirá de la inserción de los datos en el clúster Hadoop que luego podrán ser extraídos por el usuario.
El módulo de preparación de datos se basará en la manipulación directa de cada uno de los atributos que componen el conjunto de datos, esto a través de la interfaz de usuario. El mismo tendrá precargado ciertas tareas esenciales de limpieza y transformación de datos.
Todos estos pasos tendrán la finalidad de construir una vista minable que sirva de entrada para la posterior aplicación de algoritmos de minería de datos.

2.4. Objetivos

2.4.1. General

Desarrollar los módulos de extracción y preparación de datos alojados en un clúster Hadoop mediante una aplicación web para la generación de una vista minable.

2.4.2. Específicos

- Seleccionar una metodología de desarrollo de software.
- Instalar y configurar un clúster Hadoop pseudo-distribuido que será tomado como parte del modelo de datos de la aplicación.
- Instalar y configurar la herramienta Apache Drill que realizará la función de extracción y transformación de los conjuntos de datos alojados en el clúster.
- Desarrollar las funciones de inserción de los conjuntos de datos al clúster de Hadoop.
- Implementar algunos métodos de limpieza y transformación del conjunto de datos.
- Diseñar las interfaces de usuario para cada una de las etapas del flujo de trabajo de la aplicación.
- Implementar las etapas del proceso KDD hasta la generación de la vista minable.
- Realizar pruebas del flujo de trabajo de la aplicación y rendimiento del procesamiento de grandes volúmenes de datos.

2.5. Antecedentes

Tanto Oracle como Microsoft han lanzado sus herramientas capaces de realizar todo el proceso de KDD (ver capítulo 3.8) utilizando un conjunto de datos alojado en HDFS. En el 2015 Microsoft Azure pone en funcionamiento su producto de Machine Learning, donde podemos destacar su interfaz con alta usabilidad capaz de acortar en gran medida la curva de aprendizaje de cualquier analista de datos ya que permite armar una especie de algoritmo de forma gráfica (con nodos) que representan instancias de los datos, transformaciones o algoritmos aplicados sobre el conjunto. Por otra parte Oracle Big Data Discovery lanzado también en el 2015, permite realizar todos los pasos del proceso de KDD de manera distinta separando los pasos de limpieza y transformación de la exploración de los datos de forma gráfica. Ambas herramientas ofrecen al usuario la abstracción del uso de los conjuntos de datos de forma local y no alojados en un clúster Hadoop de forma distribuida, lo que genera un valor agregrado ya que el usuario no debe preocuparse por la capacidad que pueda tener el equipo desde donde se esta usando la herramienta en cuanto al almacenamiento y procesamiento de dichos datos.
Cabe destacar que ambos productos son privativos y ofrecidos de forma empresarial ya que su licencia representa una alta inversión que un analista de datos o un estudiante universitario no puede costear para su uso o entrenamiento.

Capítulo 3

Marco Conceptual

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. [7]

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. [8]

3.3. Conocimiento

El conocimiento es un conjunto de representaciones abstractas que se almacenan mediante la experiencia o la adquisición de conocimientos o a través de la observación. En el sentido más extenso que se trata de la

tenencia de variados datos interrelacionados que al ser tomados por sí solos, poseen un menor valor cualitativo. [9]

3.4. Ciencia de Datos

Ciencias de datos es la generación de conocimiento a partir de grandes volúmenes de datos, aplicando técnicas de procesamiento paralelo y distribuido para implementar algoritmos que permitan predecir o detectar patrones sobre los datos almacenados. A partir de los resultados obtenidos se podrán construir herramientas que permitan analizar los resultados y realizar procesos de toma de decisiones. [10]

3.5. Grandes volúmenes de datos (Big Data)

Concepto aplicado para definir el conjunto de datos relacionados a un software donde estos superan la capacidad de su obtención y procesamiento, para generar una respuesta en un tiempo razonable. A su vez el término es aplicado cuando los datos son heterogéneos (estructurados, no estructurados y semiestructurados). Para realizar los repositorios de datos de estos software no se utilizan las bases de datos relacionales, ya que debido al volumen de dichos datos estas no serían capaces de responder en el tiempo adecuado para continuar con la ejecución del programa. [11]

El Big Data puede ser descrito por 5 características importantes que lo definen de manera clara y puntual, las mismas son conocidas como las 5 V's, a continuación serán explicadas detalladamente:

Figura 3.1: 5v's de Big Data [1]

- **Volumen:** Se refiere a la cantidad de datos que son manipulados por la aplicación, partiendo por unidades de Terabytes llegando hasta Zettabytes. Debido a esto el uso de bases de datos relacionales desmejora la eficiencia del programa, ya que el rendimiento de las mismas es deficiente y no es posible tener particiones de las mismas.
- **Velocidad:** Se refiere no solo a la alta frecuencia con la que se generan nuevos datos, sino a la necesidad de dar respuesta a la información en tiempo real.
- **Variedad:** Hace referencia a la naturaleza diversa de la información a manejar. Venimos de información estructurada que encajaba

perfectamente en el modelo relacional pero ahora nos encontramos con información no estructurada y semiestructurada que requiere de nuevos métodos de persistencia y consulta.

- **Veracidad:** Cuando se habla de dichas cantidades de datos la precisión y la calidad son difíciles de controlar, estas varían ampliamente dependiendo de distintos factores como lo son su fuente, su calidad y su certeza. Debido a esto en su análisis podría arrojar valores erróneos, para evitar esto en el proceso de realizar ciencia de datos se debe tener una etapa de preprocesamiento de datos.

- **Valor:** Cada dato representa un valor particular el cual puede ser leído sin ningún proceso y arrojar un significado, pero en big data va más allá de ese valor representativo ya que se estudia ese significado de valor oculto que puede poseer un dato, por ejemplo en big data el hecho de ausencia de datos también puede tener un significado importante en su análisis. [12]

3.6. Organización de los datos

Se refiere a cómo almacenar los datos usando un sistema manejador de bases de datos o cualquier otra tecnología para la administración de los mismos. Describe cosas como, por ejemplo, tablas relacionales y columnas, o clases y atributos orientado a objetos, entre otros.

3.6.1. Estructurados

Los tipos de datos estructurados organizan los elementos de datos y estandarizan como los elementos de datos se relacionan unos con otros, esto se hace siguiendo un modelo de datos específico que implica una serie de reglas que deben ser aplicadas a estos. Este modelo es utilizado en las bases de datos relacionales. Estos tipos de datos son representados en los lenguajes de programación en estructuras de datos como arreglos, listas, matrices, pilas y colas, las cuales son llenadas con datos numéricos, caracteres u otras estructuras de datos.

3.6.2. Semi-Estructurados

Estos son datos estructurados que no corresponden a un modelo de estructura formal como las bases de datos relacionales, ya que son definidos por su propia estructura, suelen tener etiquetas o marcas para identificar y separar los elementos semánticamente y estableciendo jerarquía entre estos elementos. El concepto de esta organización de datos viene dado por lenguajes de marcado como el Extensible Markup Language (XML) usado en el área web y convertido en un estándar por la W3C. Otro ejemplo de datos semi-estructurados viene dado por otro formato de texto que ha sido impulsado en los últimos años debido al fácil intercambio de datos presente en las aplicaciones web llamado Javascript Object Notation comúnmente conocido como JSON.

3.6.3. No Estructurados

Se refiere a los datos que no siguen un modelo de datos predefinidos y tampoco están organizados en alguna estructura específica. Los datos no estructurados se presentan en una variedad de diferentes tipos los cuales son difíciles de clasificar y por lo tanto difíciles de almacenar y ser recuperados con consultas. La solución para su almacenamiento son las bases de datos NoSQL debido a su estructura flexible y su rapidez en el acceso a los datos. [13]

3.7. Modelo de Datos

En Big Data el modelo de datos que se tiene para representarlos es disperso,distribuido, persistente y ordenado, el cual siempre está indexado por una clave de acceso que puede ser clave fila, clave columna y siempre lleva una marca de tiempo que lo hace única dentro del modelo. A continuación se enumeran los distintos modelos de datos conocidos para lograr una representación ordenada de los datos de un sistema.

3.7.1. Relacional

Es comúnmente conocido en el área de la computación ya que fue el primer modelo que fue capaz de organizar la data y hacerla persistente con los sistemas manejadores de bases de datos. Estos son capaces de ordenar la data en tablas que están relacionadas por uno o varios campos claves en las mismas que permiten la navegación entre ellas al momento de realizar las operaciones CRUD (Crear, Actualizar y Eliminar) de filas dentro de las tablas que componen al modelo. [14] Las propiedades que rigen cómo se realizan las transacciones en una base de datos relacional son llamadas propiedades ACID y establecen los siguientes parámetros:

Atomicidad:

Si una operación consiste en una serie de pasos, todos ellos ocurren o ninguno, es decir, ninguna operación se ejecuta de manera incompleta.

Consistencia:

Es la propiedad que asegura que todos los datos dentro de la base de datos son exactos y consistentes, para lograr esto también debe garantizar que las transacciones que se ejecutaron en la base de datos fueron válidas para llegar a un estado válido.

Aislamiento:

Propiedad que asegura que una operación dentro de la base de datos no puede afectar a otras, asegura que dos o varias transacciones realizadas sobre la misma dato sean independientes entre sí.

Durabilidad:

Esta propiedad asegura la persistencia sobre los datos contenidos en la base de datos, asegura que una vez realizada una operación sobre algún dato, ésta persistirá y no se podrá deshacer aunque falle el sistema. [15]

Aunque en este modelo de datos se mantiene a la perfección las características de consistencia y disponibilidad de los datos, el mismo no puede

ser implementado en el uso de Big Data, partiendo de la limitación que el Big Data está concebido para un ambiente distribuido, y dicha característica de distribución no puede ser ofrecida por este modelo de datos relacional.

3.7.2. Familia de Columnas

Estas fueron introducidas por primera vez en 1970 en productos como Model 204 y ABABAS, este enfoque ha resurgido recientemente en Vertica y en cierta medida en QD Technology. Como es indicado en su nombre estas bases de datos son organizadas columna por columna en vez de por fila como lo establecen las relacionales, es decir, se agrupa la columna de todos los casos de un solo elemento de dato, por ejemplo: Nacionalidad. Esto presenta como ventaja la eficacia en consultas análiticas como lista de selecciones, que se necesitan todas las instancias de cierto elemento.
Cada columna es almacenada contiguamente en un lugar separado en disco, usando generalmente unidades de lectura grandes para facilitar el trabajo al buscar varias columnas en disco. Para mejorar la eficiencia de lectura, los valores se empaquetan de forma densa usando esquemas de compresión ligera cuando es posible. Los operadores de lectura de columnas se diferencian de los comunes (de filas) en que son responsables de traducir las posiciones de los valores en locaciones de disco y de combinar y reconstruir, si es necesario, tuplas de diferentes columnas.
Este cambio se traduce en el incremento de la velocidad en lecturas, ya que si se requiere consultar un número reducido de columnas, es muy rápido hacerlo pero no es eficiente para realizar escrituras. Por ello este tipo de soluciones es usado en aplicaciones con un índice bajo de escrituras pero muchas lecturas. Típicamente en data warehouses y sistemas de inteligencia de negocios, donde además resultan ideales para calcular datos agregados. Cabe resaltar que parte del auge actual que está provocando la tendencia de las bases de NoSQL se debe a la adopción de Cassandra (originalmente desarrollada por y para Facebook, luego donada a la fundación Apache) por parte de Twitter y Digg. Apache Cassandra es la base de datos orientada a columnas más conocida y utilizada actualmente. [16]

3.7.3. Clave/Valor

Son bases de datos que utilizan una estructura de tipo diccionario con tabla de hashes donde dependiendo de una clave que puede ser única o no (dependiendo del sistema manejador), puede accederse a datos de manera rápida y directa. La diferencia de este tipo de base de datos radica en la posibilidad de almacenar datos sin ningún esquema predefinido. El mayor potencial de estas bases de datos es la simplicidad de la lectura y escritura, así como la velocidad de acceso a la información.
Una de las bases de datos clave/valor mayormente usada y conocida es Redis, la misma es un servicio que se mantiene en memoria ram con la capacidad de mantener la persistencia de las operaciones. Esta puede manejar operaciones de búsqueda de listas y sets organizados por cierto valor. Los tiempos de respuesta de lectura y escritura son bastante bajos, por esta razón se ha vuelto popular su uso sistemas donde se priorice la respuesta al usuario. [17]

3.7.4. Orientado a Documentos

Las bases de datos orientadas a documentos son una subcategoría de las bases de datos NoSQL, las cuales están centradas en el almacenamiento de documentos. A diferencia de las bases de datos relacionales estos documentos se describen ellos mismos y por lo tanto son libres de esquema. Las codificaciones usadas por los objetos que las componen están XML, YAML, JSON y BSON.
En estas bases de datos cada registro es un documento que debe estar bien formado para auto-describirse adecuadamente. A su vez existe un nivel superior a los documentos que son las colecciones, ya que estas están compuestas por uno o muchos documentos a la vez.
Para realizar consultas cruzadas entre documentos, haciendo analogía a los JOIN en base de datos relacionales, se debe programar siguiendo el paradigma de MapReduce para recorrer todos los documentos asociados y encontrar las coincidencias que se requieran.
Estas bases de datos permiten la escalabilidad horizontal debido a su naturaleza distribuida. Para hacer la replicación de la misma se maneja un esquema de maestro-esclavo. También presentan excelente comportamiento de concurrencia de procesos.

Dentro de las más utilizadas actualmente están CouchDB y MongoDB siendo excelentes opciones debido a su estable comportamiento. En el caso de MongoDB es recomendable su uso en el área web debido a que los documentos que maneja son de formato JSON y por lo tanto son compatibles con el formato de comunicación entre aplicaciones web. [18]

3.7.5. Orientado a Grafos

Las bases de datos orientadas a grafos representan la información como nodos de un grafo y sus relaciones como las aristas del mismo. De esta forma, se puede usar la teoría de grafos para recorrer la base de datos, ésta describe atributos de los nodos (entidades) y de las aristas (relaciones). Una base de datos orientada a grafos debe estar absolutamente normalizada, lo que significa que cada tabla tendrá una sola columna y cada relación tan solo dos. Con esto, se consigue que cualquier cambio en la estructura de la información tenga un efecto tan solo local. Lo que nos indica que un sistema de gestión de base de datos gráficas se construye generalmente para el uso transaccional de sistemas, optimizar el rendimiento, integridad de los datos y disponibilidad operacional.
Este tipo de base de datos está diseñada para los datos cuyas relaciones son bien representadas en forma de grafo. Es decir, los datos son elementos interconectados con un número no determinado de relaciones entre ellos.
A pesar de que las estructuras de datos en forma de grafos son en teoría normalizables, incluso en sistemas relacionales, esto tendría serias implicaciones en el rendimiento de las consultas debido a las características de implementación de los RDBMS, donde cada operación sobre una relación resultaría en una operación de unión para el gestor de datos, lo cual es un proceso lento y no escalable ante un creciente número de tuplas en estas tablas.

a) **Nodos:** se utilizan para representar entidades, estas están contenidas por documentos formados en formato JSON donde se representan los datos.

b) **Relaciones:** organizan los nodos conectandolos entre sí. Una relación conecta dos nodos, un nodo de inicio y un nodo final. Un nodo también puede tener una relación consigo mismo. Al igual que los nodos,

> las relaciones pueden tener propiedades, estas permiten encontrar datos relacionados, organizan a los nodos en estructuras arbitrarias, lo que permite crear grafos que se asemejen a una lista, un árbol, un mapa, o una entidad compuesta. Cabe destacar que las relaciones manejadas dentro de las bases de datos orientadas a grafos son unidireccionales. [19]

El rendimiento de las consultas y la capacidad de respuesta son las primeras preocupaciones de muchas organizaciones con respecto a sus plataformas de datos. En particular, sistemas transaccionales en línea y grandes aplicaciones web, deben responder a los usuarios en milisegundos si se quiere tener éxito. En el mundo relacional, como el tamaño del conjunto de datos de una aplicación crece, los problemas debido a los join comienzan a manifestarse y el rendimiento se deteriora. El uso de adyacencia libre de índices, en una base de datos gráfica convierte los complejos joins en recorridos rápidos al grafo, manteniendo de ese modo el rendimiento en milisegundos, independientemente del tamaño global del conjunto de datos.
Otra ventaja que presenta el modelo de datos de un grafo es que reduce el desajuste de impedancia que ha plagado el desarrollo de software durante décadas, lo que a su vez reduce los gastos generales de desarrollo de traducción de ida y vuelta entre un modelo de objetos y un modelo relacional. Más importante aún, un modelo orientado a grafos reduce el desajuste de impedancia entre los dominios técnicos y comerciales: expertos en la materia, arquitectos y desarrolladores pueden hablar y representar el núcleo de dominio mediante un modelo compartido que se incorpora en la propia aplicación.
Dentro de las bases de datos orientadas a grafos surgen tres productos utilizados en la actualidad el primero es Neo4j, con la característica de ser robusta, escalable y de alto rendimiento. El segundo producto es Titan donde a parte de permitir la escalabilidad y alto rendimiento es capaz de soportar varios almacenes de datos en backend como Cassandra, HBase y Oracle BerkleyDB. Por último existe un producto que ha entrado mucho en el mercado de las bases de datos NoSQL llamado OrientDB, donde el mismo ofrece como característica principal SQL como el lenguaje utilizado para realizar la construcción de la base de datos y las consultas, esto da un valor agregado ya que las anteriores opciones debían tener cierto aprendizaje en los lenguajes de consulta utilizados para su funcionamiento.

3.8. Proceso KDD (Knowledge Discovery in Databases)

El proceso KDD se refiere al proceso no-trivial de descubrir conocimiento e información potencialmente útil dentro de los datos contenidos en algún repositorio de datos. No es un proceso automático, es un proceso iterativo que explora exhaustivamente grandes volúmenes de datos para determinar relaciones. Es un proceso que extrae información de calidad que puede usarse para dibujar conclusiones basadas en relaciones o modelos dentro de los datos.

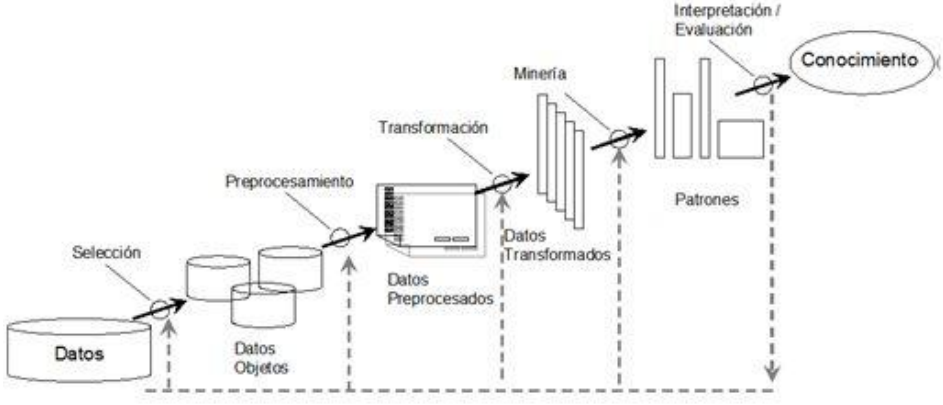

Figura 3.2: Etapas del proceso KDD [2]

1. **Selección de datos:** En esta etapa se determinan las fuentes de datos y el tipo de información a utilizar. Es la etapa donde los datos relevantes para el análisis son extraídos desde la o las fuentes de datos.

2. **Preprocesamiento:** Esta etapa consiste en la preparación y limpieza de los datos extraídos desde las distintas fuentes de datos en una forma manejable, necesaria para las fases posteriores. En esta etapa se utilizan diversas estrategias para manejar datos faltantes o en blanco, datos inconsistentes o que están fuera de rango, obteniéndose al final una estructura de datos adecuada para su posterior transformación.

3. **Transformación:** 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.

4. **Minería de datos:** Es la fase de modelamiento propiamente, en donde métodos inteligentes son aplicados con el objetivo de extraer patrones previamente desconocidos, válidos, nuevos, potencialmente útiles y comprensibles y que están contenidos u "ocultos" en los datos.

5. **Interpretación y Evaluación:** Se identifican los patrones obtenidos y que son realmente interesantes, basándose en algunas medidas y se realiza una evaluación de los resultados obtenidos. [20]

3.9. Lenguajes de programación

Un lenguaje de programación es un lenguaje formal diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras. [21]

3.9.1. HTML

HTML o HyperText Markup Language es un lenguaje de marcado para la descripción y definición visual de páginas web de forma bien estructurada. Determina el contenido de la página más no su funcionalidad. HTML es un estándar determinado así por la W3C (World Wide Web Consortium), organización dedicada a la estandarización de las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación.
HTML es el lenguaje que describe la estructura y el contenido semántico de un documento web mediante el uso de "etiquetas", las cuales están rodeadas por corchetes angulares (¡,¿,/). Estas etiquetas son interpretadas por los navegadores que luego genera la vista de la página web. [22]

3.9.2. CSS

Cascade Style Sheets u hojas de estilo en cascada es un lenguaje utilizado para describir la presentación de documentos HTML, este funciona como un mecanismo para definir cómo se mostrará por pantalla el documento, o cómo será pronunciado por dispositivos de lectura de pantalla. Así como con HTML, es un estándar definido por la W3C quien se encargó de formular las especificaciones de los documentos CSS.
Los estilos son aplicados individualmente, o por grupos, sobre los elementos (etiquetas) de los documentos HTML y estos son definidos mediante pares 'nombre:valor'. Los estilos pueden ser definidos de tres formas:

- En un documento fuera del HTML con extensión '.css' que se enlazara con el HTML mediante el uso de la etiqueta '¡link¿¡/link¿'.
- En la cabecera del documento HTML mediante el uso de la etiqueta '¡style¿¡/style¿'.
- Por último, en el elemento propiamente dentro de la etiqueta de apertura: ¡example style="name:value"¿¡/example¿.

Cual de los tres métodos definidos se escoja es lo que define qué propiedad se le aplicará a un elemento debido a que las propiedades del mismo nombre serán sobreescritas por la que esté más cercana al elemento HTML, de ahí proviene el nombre de "cascada". [23]

3.9.3. JavaScript

JavaScript (a veces abreviado como JS) 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.

Las 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. [24]

3.10. Frameworks

Un framework es una estructura conceptual y tecnológica de soporte definido que posee artefactos o módulos de software que permiten el desarrollo de software de manera organizada y concreta, para de esta forma facilitar y acelerar la construcción de las aplicaciones por parte de los programadores.

El uso de los frameworks por parte de los programadores tiene las siguientes ventajas:

- Proporciona un estándar de desarrollo, el cual permite mantener organizado el código y la estructura.
- El programador no necesita plantearse la estructura global de la aplicación sino simplemente desarrollar en las partes de dicha estructura establecida por el framework, la mayoría de estos sigue un patrón de diseño de software.
- Facilita el desarrollo colaborativo entre distintos programadores, siendo esta una tarea tediosa cuando se realiza un desarrollo en equipo, pero al tener una estructura definida se pueden definir las tareas y los módulos a desarrollar, sin necesidad de intervenir en el trabajo del otro.
- Se pueden agregar librerías, módulos o herramientas que agilicen el desarrollo de la aplicación. [25]

Capítulo 4

Herramientas tecnológicas utilizadas

Para el desarrollo de la solución se hace necesario el uso de un conjunto de tecnologías de software tanto para el manejo de las herramientas de hadoop, como para el desarrollo de la aplicación web del lado del servidor y del cliente, donde todas en conjunto cumplen con una serie de procesos que cumplen con determinadas tareas. En este capítulo se presentan las tecnologías utilizadas para el desarrollo de este Trabajo Especial de Grado.

4.1. Tecnologías de Ciencia de Datos

Son las tecnologías que ejecutan todo el control de las tareas del clúster de hadoop, tanto para el almacenamiento de los archivos distribuídos como para el acceso a dichos datos.

4.1.1. Apache Hadoop

Apache Hadoop es un framework tolerante a fallos que permite el procesamiento distribuido de grandes volúmenes de datos sobre un clúster usando modelos simples de programación. Está diseñado para ser escalable desde un simple servidor a miles de computadoras, donde cada una ofrece almacenamiento y capacidad de cómputo. Apache Hadoop fue desarrollado

por la comunidad ApacheTM inspirados en los papers de Google sobre MapReduce. Hadoop provee servicios para el almacenamiento, procesamiento, acceso, gobernanza, seguridad y operaciones de los datos.

Hadoop Distributed File System

El Hadoop Distributed File System (HDFS) es un sistema de archivos distribuido, escalable y portátil escrito en Java para el framework Hadoop. Cada nodo en una instancia Hadoop tiene un único NameNode, un clúster de datos forma el clúster HDFS. HDFS es altamente tolerante a fallos, provee un alto rendimiento en el acceso a los datos de la aplicación y es adecuado para aplicaciones que tienen un enorme conjunto de datos.

Entre sus características principales tenemos:

- **Acceso a datos en flujos:** la construcción de HDFS fue pensada para manejar datos de manera eficiente, normalmente se maneja sobre un patrón el cual se escribe una vez un dato, pero se leen múltiples veces. Al tener grandes volúmenes de información, HDFS permite ir leyendo datos bajo demanda lo cual es una ayuda para reducir el rendimiento debido a que no se debe esperar una copia de todo un conjunto de datos para funcionar.
- **Tolerancia a fallos:** HDFS permite la replicación de los datos en distintos nodos (normalmente 3). Para poder tener disponible siempre los datos en caso de ser requerido.
- **Manejo de grandes archivos:** cuando se hace referencia a grandes archivos, son aquellos que pueden pesar cientos de Megabytes, Gigabytes, Terabytes, Petabytes, etc.
- **Acceso a datos con baja latencia:** HDFS fue creado para manejar grandes volúmenes de datos, por lo tanto, está optimizado para tener altas tasas de transferencia.
- **Escritura múltiple:** los archivos deben de ser escritos por un único proceso, las escrituras siempre son realizadas al final de los archivos. No existe soporte para escribir en un mismo archivo por múltiples procesos al mismo tiempo.

- **Pequeños archivos:** HDFS tiene un límite para almacenar archivos debido a que crea metadata de los archivos almacenados en memoria, directorios y bloques, lo cual limita la cantidad de datos a tener almacenados dependiendo de la cantidad de memoria que posea el nodo.

Arquitectura

HDFS tiene una arquitectura maestro/esclavo. Un clúster HDFS está compuesto por un NameNode que cumple el rol de maestro, y un número mayor de DataNodes usualmente uno por cada nodo en el clúster como se puede observar en la figura 4.1.

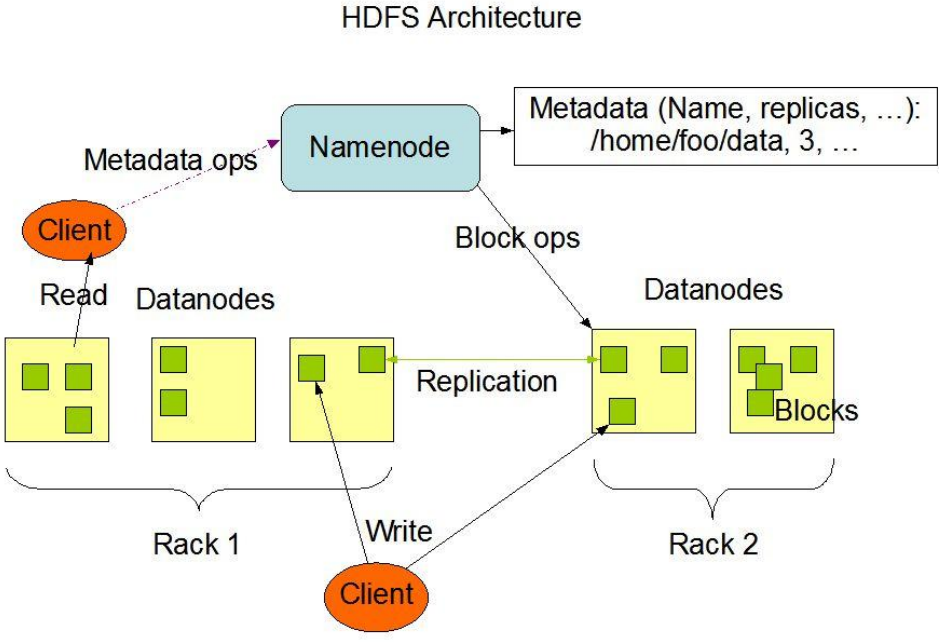

Figura 4.1: Arquitectura de HDFS [3]

- **NameNode:** Se encarga de la administración del sistema de archivos, mantiene el sistema de archivos y la metadata asociada a estos y a los directorios. El mantenimiento consiste en tener dentro de un

árbol todos los archivos y directorios para poder ser accedidos con mayor facilidad. Toda esta información es guardada en un medio de almacenamiento persistente en forma de log para ser leída luego. Almacena los datos relacionados con la posición de un bloque para un archivo en específico durante un periodo de tiempo, estos datos son actualizados al iniciar el sistema y cada cierto tiempo.

- **DataNode:** estos nodos se encargan de manejar el almacenamiento y distribución de los datos en el nodo al cual se encuentra unido. Una vez realizada la distribución, el DataNode realiza un reporte al NameNode, este reporte se realiza periódicamente y contiene los bloques los cuales están siendo almacenados en ese nodo en específico.

Bloque de Datos

HDFS está diseñado para dar soporte a archivos muy grandes y las aplicaciones compatibles con HDFS son aquellas que están diseñadas para lidiar con estos conjuntos de datos grandes. Estas aplicaciones escriben sus datos una vez pero la leen una o más veces. Debido a esto el tamaño del bloque típico usado por HDFS es de 64MB, un archivo de HDFS es picado en pedazos de 64MB y, de ser posible, cada pedazo es almacenado en diferentes DataNodes.

Namespace del sistema de archivos

HDFS soporta una organización jerárquica tradicional. Un usuario o una aplicación puede crear directorios y almacenar archivos dentro de estos directorios. El namespace de un clúster Hadoop es similar al de cualquier otro sistema de archivos, se pueden crear y remover archivos, reubicarlos de un directorio a otro o renombrar un archivos. La información del namespace es almacenada en el NameNode, cualquier cambio al namespace o a sus propiedades son guardados por el NameNode. Una aplicación puede especificar el número de réplicas que HDFS debe mantener de un archivo, su factor de replicación.

Accesibilidad

HDFS puede ser accedido por aplicaciones de distintas formas. Nativamente, HDFS provee un API para Java que las aplicaciones pueden usar, también esta disponible una librería para el lenguaje C. Así como para los navegadores HTTP, que pueden ser utilizados para navegar a través de los archivos de una instancia de HDFS. [26]

4.1.2. Apache Drill

Es un motor para realizar consultas SQL libres de esquema sobre Hadoop, bases de datos NoSQL y Almacenamiento en la Nube.

Posee 3 características importantes:

- Agilidad: Permite obtener una amplia perspectiva sin tener sobrecarga en la carga de datos, creación y mantenimiento de esquema, transformaciones, etc.
- Flexibilidad: Analiza los datos multi-estructurados y jerarquizados en los almacenes de datos no relacionales directamente sin transformar o restringir los datos.
- Familiaridad: Aprovecha sus conjuntos de habilidades SQL existentes y las herramientas de BI, incluyendo Tableau, QlikView, MicroStrategy, Spotfire y Excel.

Drill permite el tratamiento de los datos como una tablas, incluso cuando estos datos no sean tablas. Este cuenta con un modelo de datos JSON que permite realizar consultas sobre datos complejos .ªnidados", así como estructuras que evolucionan rápidamente, comúnmente encontradas en las aplicaciones modernas y almacenes de datos no relacionales.
Drill es compatible con el estándar SQL. Los usuarios, analistas y científicos de datos pueden utilizar las herramientas estándar de BI / analíticas tales como Tableau, Qlik, MicroStrategy, Spotfire, SAS y Excel para interactuar con los almacenes de datos no relacionales mediante el aprovechamiento de los drivers JDBC y ODBC. [27]

4.2. Tecnologías de Aplicaciones Web

4.2.1. NodeJs

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. [28]

Arquitectura y Funcionamiento

En la figura 4.2 podemos ver la arquitectura orientada a eventos de NodeJs, 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 NodeJS 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.

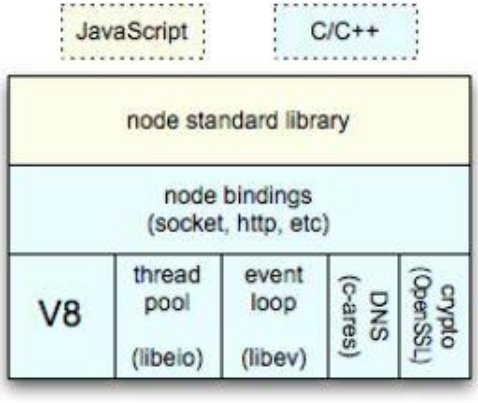

Figura 4.2: Arquitectura de NodeJS [4]

NodeJs 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 (ver figura 4.3), 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.
En el siguiente gráfico se presenta la interacción de una aplicación desarrollada en NodeJS:

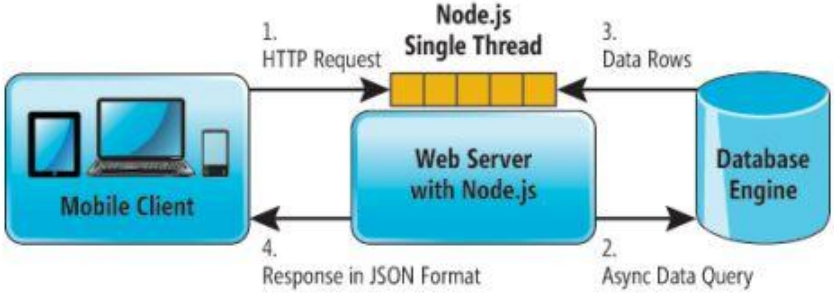

Figura 4.3: Interacción de aplicación NodeJS [5]

Como se puede observar todo se ejecuta en el servidor, sin la preocupación de la compatibilidad del código con los diferentes clientes.

Servidor de NodeJS

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. La figura 4.4 presenta una estructura ilustrativa de cómo se realizan las peticiones asíncronas y cómo ocurre la concurrencia de los eventos:

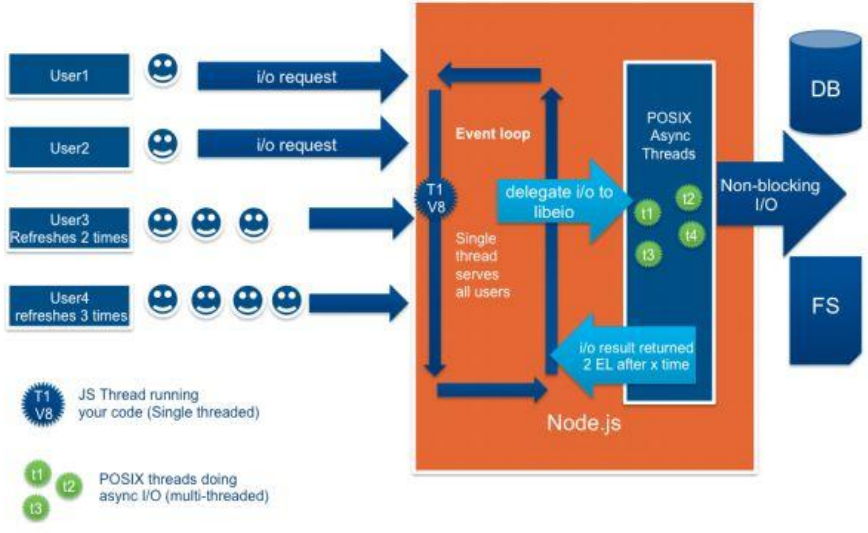

Figura 4.4: Peticiones Asíncronas de NodeJS [6]

Módulos de NodeJS

Los módulos de NodeJS 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, por ejemplo el que realiza creación del servidor http, en la figura 4.5 se muestra el código javascript que permite realizarlo:

```
1  const http = require('http');
2
3  const hostname = "127.0.0.1";
4  const port = 3000;
5
6  const server = http.createServer((req, res) => {
7      res.statusCode = 200;
8      res.setHeader('Content-Type', 'Text/plain');
9      res.end('Hello World\n');
10 });
11
12 server.listen(port, hostname, () => {
13     console.log(`Server running at http://${hostname}:${port}/`);
14 });
```

Figura 4.5: Código JavaScript para la creación de un servidor http en NodeJs

4.2.2. Meteor

Meteor es una plataforma para crear aplicaciones web y móviles en tiempo real construida sobre NodeJS, 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 NodeJS la misma utiliza como lenguaje base Javascript del lado del cliente y del servidor. [29]

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, esta simplicidad es llamada “data on the wire” y significa que los datos viajan entre el modelo y la vista de manera simple y rápida.

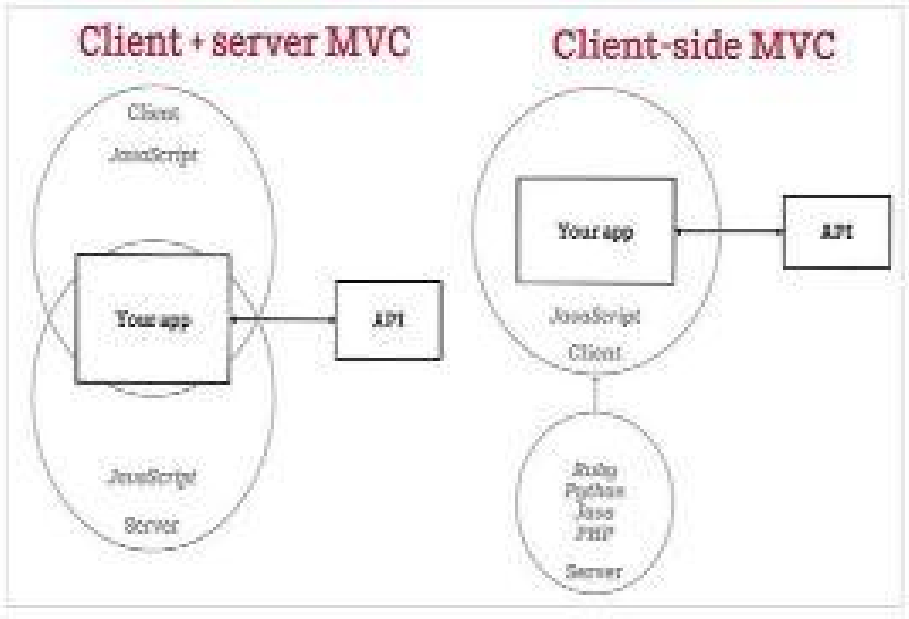

Figura 4.6: Arquitectura MVC vs. aplicación en Meteor

Como podemos observar en la figura 4.6 en la sección Client-side MVC es donde se muestra la arquitectura de una aplicación desarrollada en Meteor donde no hay un límite definido entre la vista y el modelo de datos, sino que se mantienen juntos y se relacionan directamente en su contenido, todo lo que está presentado en la vista se mantiene consistente en el modelo de datos.

Estructura de Archivos

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. A continuación se presenta un fragmento de código de un template:

```
<template name="intro">
    <div class="container intro">
        <h1><img class="bullet-intro" src="images/logo-black.png">Meteor</h1>
        <div class="container">
            {{#each this}}
                <p class="bod-pad">{{bod}}</p>
                <p class="bod-pad">{{integrantes}}</p>
            {{/each}}
            <button class="button"><span>Continuar </span></button>
        </div>
    </div>
</template>
```

Figura 4.7: Código de un template de Meteor

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.

4.2.3. MongoDB

Es una base de datos ágil que permite a los esquemas cambiar rápidamente cuando las aplicaciones evolucionan, proporcionando siempre la funcionalidad que los desarrolladores esperan de las bases de datos tradicionales, tales como índices secundarios, un lenguaje completo de búsquedas y consistencia estricta.
La misma pertenece a la familia de las bases de datos NoSQL orientadas a documentos, donde se pueden almacenar archivos en formato JSON donde se permite la anidación de los mismo.
Posee un lenguaje de consulta propio del sistema manejador de base de datos, el cual permite realizar cualquier tipo de consulta sin importar el esquema del documento almacenado.

Entre sus características principales tenemos:

- **Consultas AdHoc:** 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:** 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 nuevos 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 controladores, no en el servidor, por lo que está incluida en los controladores oficiales que la compañía de MongoDB desarrolla. Estos dricontroladoresers 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. [30]

Capítulo 5

Marco Metodológico

Para el logro de los objetivos planteados en este trabajo, fue necesario definir un esquema o metodología que permita el desarrollo rápido y eficiente de cada uno de los requerimientos del sistema. A continuación, se presenta la especificación de la metodología utilizada.

La mayoría de los métodos de desarrollo plantean un esquema de trabajo que se divide en 4 fases: Análisis, Diseño, Codificación y Pruebas, este debe ser ejecutado en forma secuencial para el desarrollo de la aplicación. Sin embargo, la adaptación de este enfoque no resulta ser el más adecuado en la actualidad, donde el entorno de desarrollo tiende a ser muy cambiante. La aparición de nuevos requerimientos durante la ejecución de cualquiera de las fases es muy probable, lo que conlleva la replanificación de las actividades y la necesidad de volver a ejecutar las fases del método de desarrollo.

Debido a esto, se decidió trabajar con un método ágil en lugar del enfoque tradicional, los cuales proponen dividir el desarrollo de la aplicación por iteraciones cortas, dividiendo los requerimientos de la aplicación por módulos, y desarrollando un módulo por iteración. A diferencia de los métodos tradicionales, los métodos ágiles no emplean una planificación estricta sino abierta y flexible, para aumentar la habilidad de respuesta a los cambios que puedan surgir, intentan hacer entregas frecuentes y promueven la comunicación con el cliente.

5.1. Programación Extrema XP

Entre los métodos ágiles más conocidos se encuentra XP (eXtreme Programming o Programación Extrema), centrado en potenciar las relaciones interpersonales como clave para el éxito del desarrollo de software. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y adaptabilidad. [31]

El objetivo principal que se persigue es la satisfacción del cliente. Esta metodología trata de dar al cliente el software que necesita y cuando lo necesita. Por tanto, se debe responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final del ciclo de programación. Por otro lado, trata de potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software.

El ciclo de desarrollo consiste en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos.

Como principios básicos en la programación extrema se tienen:

- **Planificación:** Hay una comunicación frecuente entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.

- **Entregas pequeñas:** Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más de 3 meses.

- **Sistema de metáforas:** El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema (conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema).

- **Diseño simple:** Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.

- **Pruebas:** La producción de código está dirigida por las pruebas unitarias. Éstas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.

- **Refactorización del código:** Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo.

- **Programación en parejas:** Toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores, etc.).
- **Propiedad colectiva del código:** Cualquier programador puede cambiar cualquier parte del código en cualquier momento.
- **Integración continua:** Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.
- **Ritmo sostenible:** Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo.
- **Relación con el cliente:** El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Éste es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita.
- **Estándares de programación:** XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

El mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras. La mayoría de las prácticas propuestas por XP no son novedosas ya que algunas habían sido propuestas en ingeniería del software e incluso demostrado su valor en la práctica. El mérito de XP es integrarlas de una forma efectiva y complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo.

5.1.1. Roles

Existen diferentes roles y responsabilidades en el proceso de desarrollo XP. Para este trabajo, los roles existentes son:

- **Desarrollador:** es el responsable de llevar a cabo la codificación, el diseño y realizar las pruebas unitarias. También define las tareas que conlleva cada historia de usuario, y estima el tiempo que requerirá cada una.
- **Cliente:** escribe las historias de usuario y las pruebas funcionales para validar su implementación. Asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.

5.1.2. Historias de Usuario

La historia de usuario es la técnica utilizada para especificar los requisitos y funcionalidades que debe cumplir el software, estas tienen el propósito de describir los requerimientos de los clientes. El contenido de las historias de usuario proviene de los clientes, tal y como ven ellos las necesidades del sistema, por lo tanto son descripciones cortas y escritas en el lenguaje de usuario, sin terminología técnica. En cada iteración se lleva a cabo un grupo de historias, es decir, en cada iteración se trabaja sobre uno o más requerimientos.

El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden ser desechadas, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario debe ser lo suficientemente comprensible y estar delimitada para ser desarrollada en un corto período de tiempo.

A continuación presentaremos una plantilla a utilizar en el desarrollo del proyecto para representar las historias de usuarios:

ID	Fecha	Descripción

Cuadro 5.1: Plantilla para la representación de tablas de usuario.

5.1.3. Actividades

La programación extrema define cuatro actividades básicas que se realizan dentro del proceso de desarrollo de software: Planificación, Diseño, Codificación y Pruebas. A continuación se detallan estas actividades.

Planificación

- Se escriben historias de usuario con el propósito de describir los requerimientos del sistema, conduciendo a la creación de las pruebas de aceptación y proporcionando a su vez una estimación del tiempo necesario para el desarrollo y planificación de las entregas. El tiempo ideal para cada historia de usuario es de 1 a 3 semanas.
- El equipo estima la duración de la implementación de cada historia de usuario y el cliente prioriza las historias teniendo en cuenta el valor que le aporta al sistema tenerla completa.
- A partir de las estimaciones, se planifican entregas de pequeñas versiones operativas al cliente, en donde este puede introducir nuevas funcionalidades.

Diseño

- Se debe implementar la solución más simple que pueda funcionar, la complejidad innecesaria y el código extra debe ser removido de forma

inmediata y no se deben agregar nuevas funcionalidades antes de que sean agendadas.

- El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo, la cual es una historia compartida que describe cómo debería funcionar el sistema. Esto con el n de solventar el hecho de no contar con una definición de la arquitectura desde el comienzo, ya que en XP la arquitectura se asume evolutiva.

Codificación

- El cliente está siempre disponible, ayudando al equipo desarrollador y formando parte de él. Gran parte del éxito del proyecto se debe a que es el cliente quien conduce de forma constante el trabajo hacia lo que aportará mayor valor de negocio. Todas las fases de XP requieren de la comunicación con el cliente, preferiblemente cara a cara.

- Durante todas las actividades el cliente ayuda con la estimación de tiempo para las historias de usuario, ayuda con la asignación de las prioridades, cerciora que las funcionalidades del sistema cubran todas las historias de usuario y participa en las reuniones de planificación para completar detalles de las tareas. Igualmente colabora con la elaboración de las pruebas.

- La producción de código está dirigida por las pruebas unitarias, las cuales son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema.

- Todo el código de producción es programado por parejas, aumentando la calidad del mismo, mejorando el diseño, disminuyendo el tamaño del código y solventando de forma más rápida los problemas de programación.

- Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día. Esto con el fin de que todos los participantes trabajen con la última versión y así evitar problemas de compatibilidad.

- El código es propiedad colectiva de los participantes. De esta forma, cualquier programador puede cambiar cualquier parte del código en cualquier momento y evita que algún programador sea imprescindible para realizar cambios en alguna porción de código.

Pruebas

- Todo el código debe tener pruebas unitarias asociadas y éste debe ser pasado por las pruebas antes de la entrega del sistema final.
- Las pruebas de aceptación se realizan con el fin de demostrar que hayan sido cubiertos todos los requerimientos expresados en las historias de usuario. Además se utilizará un formato para registrar cada una de las pruebas que se realicen, el mismo puede apreciarse en el siguiente cuadro:

ID	Descripción del caso de prueba	Resultado esperado	Resultado obtenido

Cuadro 5.2: Plantilla para pruebas.

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.

5.1.4. Requerimientos Generales del Sistema

El sistema a realizar radica principalmente en el desarrollo de una aplicación web capaz de conectarse con un clúster de hadoop y manipular los conjuntos de datos almacenados dentro de éste. Dicha manipulación estará enmarcada dentro del proceso de KDD en las etapas de extracción y transformación del conjunto de datos para la generación de una vista minable donde se apliquen posteriores algoritmos de minería de datos. La finalidad de dicho sistema es aplicarlo directamente en la escuela de computación de

la Universidad Central de Venezuela, con miras en apoyar la educación y preparación de los estudiantes en el área de ciencia de los datos.

Requerimientos Funcionales

Son declaraciones de los servicios que debe proporcionar el sistema para realizar los objetivos solicitados por el usuario. Se definen los siguientes requerimientos funcionales:

- Permitir al usuario añadir al clúster el conjunto de datos que necesite para realizar cualquier estudio.
- Ofrecer la opción al usuario de compartir o no el conjunto de datos que añadió al clúster.
- Establecer una política de seguridad en la aplicación donde no se permita que los otros usuarios accedan a conjuntos establecidos como privados.
- Ofrecer al usuario la opción de eliminar campos y registros.
- Ofrecer al usuario la opción de filtrar un campo por comparación.
- Ofrecer al usuario la opción de explorar la metadata de los campos.
- Ofrecer al usuario la opción de reemplazar valores que se encuentran en un campo por imputación.
- Implementar la funcionalidad de deshacer la acción realizada sobre el conjunto de datos si la misma no fue satisfactoria por parte del usuario.
- Desplegar una lista de las acciones realizadas sobre el conjunto de datos.

Requerimientos No Funcionales

Los requerimientos no funcionales abarcan aspectos del sistema visibles para el usuario, que no están relacionados de forma directa con el comportamiento funcional del sistema. Se definieron los siguientes requerimientos no funcionales:

- Ofrecer soporte multiplataforma referido al navegador en este caso por ser una aplicación web.
- Proveer control de errores y excepciones que proporcione robustez al sistema.
- Elaborar una aplicación que goze de usabilidad para los usuarios.
- Realizar una buena documentación de la aplicación para permitir futuros desarrollos de funcionalidades que permitan la extensión de la aplicación.
- Las tecnologías de desarrollo deben ser no propietarias.

Capítulo 6

Desarrollo

Siguiendo el proceso de desarrollo de XP, el desarrollo del sistema se encuentra dividido en iteraciones, el objetivo de cada iteración es obtener una versión del sistema que incluya la implementación de las historias de usuario.

6.1. Iteración 0: Diseño de la solución

En esta iteración se definen los requerimientos generales del sistema tomando como base la propuesta elaborada. Adicionalmente, los componentes del mismo son representados a través de un esquema gráco (metáfora), a partir del cual se desarrollan las demás iteraciones.

6.1.1. Planificación

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

ID	Fecha	Descripción
1	15/11/2016	Definición de requerimientos. El cliente quiere una aplicación web que realice la preparación de datos de conjuntos alojados 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.

Cuadro 6.1: Historias de usuario. Iteración 0.

6.1.2. Diseño

Se debe desarrollar una aplicación web que se conecte con un clúster de hadoop y manipule los conjuntos de datos almacenados allí aplicando el proceso de KDD de extracción y transformación, hasta generar una vista minable.
Para cumplir con los requerimientos establecidos se realizó un esquema general del sistema donde se identifican cada uno de los componentes que lo conforman y la interacción entre cada uno de ellos.

Figura 6.1: Esquema General del Sistema

Como se puede observar en la figura 6.1 se plantea un esquema de la aplicación donde el analista de datos accede a ésta y es capaz de conectarse al

clúster de hadoop para subir o acceder a conjuntos de datos alojados en él, donde estos a su vez pueden ser preparados para generar una vista minable. Todas las operaciones realizadas por el analista de datos son guardadas en una base de datos mongoDB, donde ademas se almacena los perfiles de usuario y toda la metadata operacional de la aplicación, cabe destacar que esta base de datos mongoDB esta embebida en la aplicación desarrollada en Meteor.

A continuación se presentan los diagramas de casos de uso que establecen el flujo de trabajo y la interacción del usuario en la aplicación:

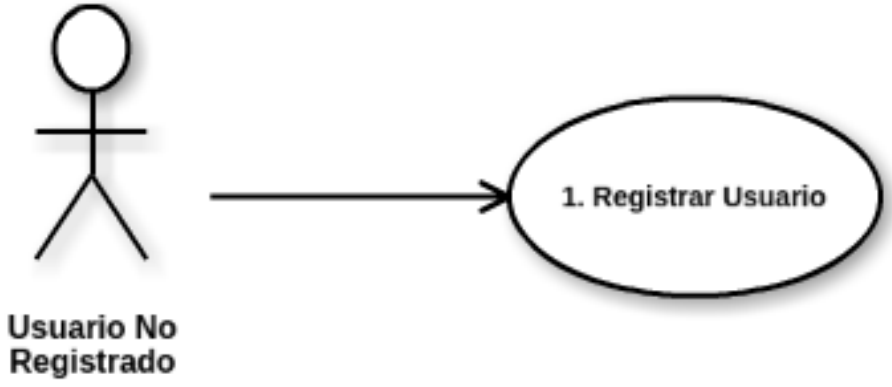

Figura 6.2: Casos de Uso de Usuario No Registrado

Figura 6.3: Casos de Uso de Usuario Registrado

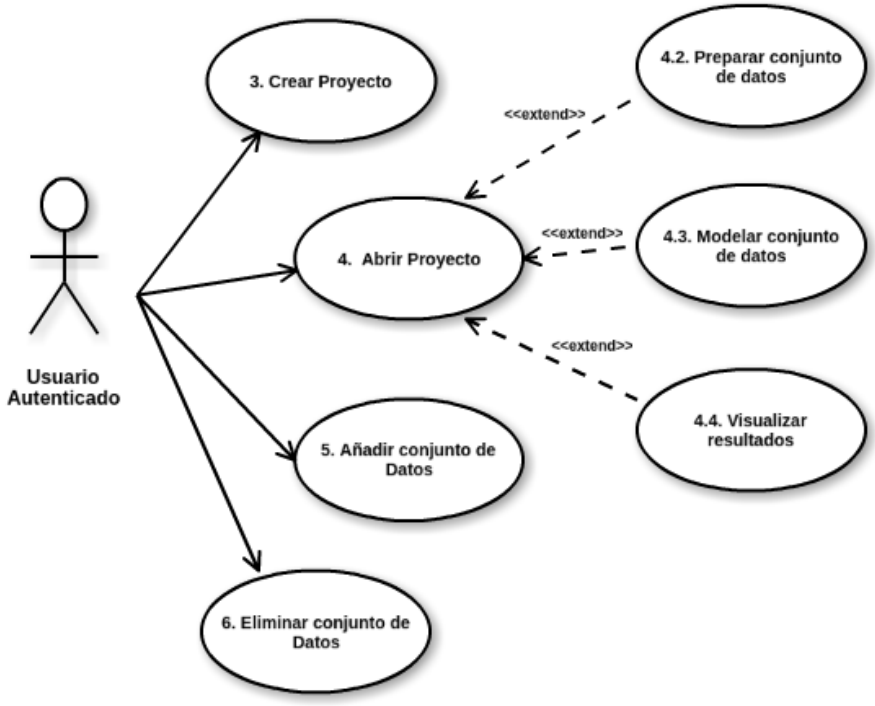

Figura 6.4: Casos de Uso de Usuario Autenticado

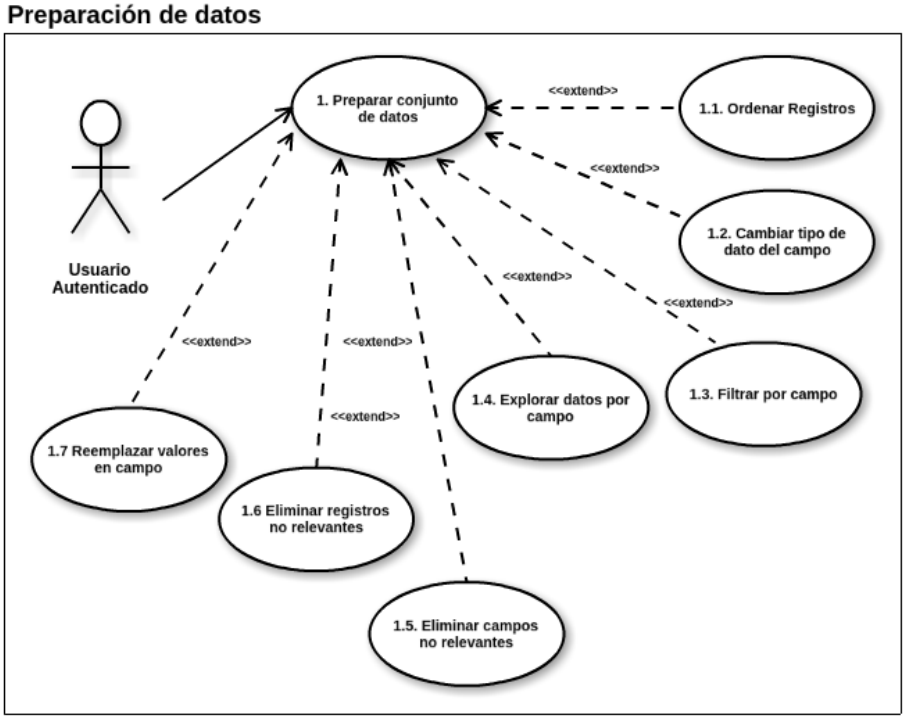

Figura 6.5: Casos de Uso de Módulo de Preparación de datos

6.2. Iteración 1: Instalación y Configuración del Clúster Hadoop

En esta iteración los desarrolladores realizan la instalación y configuración de Apache Hadoop, que se encargará del procesamiento y almacenamiento de los conjunto de datos en un ambiente distribuido. Además de documentarse con la REST API para la versión 2.7.3.

6.2.1. Planificación

En el siguiente cuadro se muestran las historias de usuario desarrollada en esta iteración.

ID	Fecha	Descripción
4	10/12/2016	Se debe instalar y configurar Apache Hadoop.
5	10/12/2016	Se debe usar una herramienta que ofrezca un REST API para ser manipulada desde la aplicación web a desarrollar.

Cuadro 6.2: Historias de usuario. Iteración 1.

6.2.2. Diseño

Para el almacenamiento de los conjuntos de datos se debe utilizar una herramienta que automatice dicho procedimiento en un ambiente distribuido.
Apache Hadoop se encarga de esto utilizando el Sistema de Archivos Distribuidos de Hadoop (HDFS), además provee un REST API mediante el cual se puede realizar las creación, lectura, actualización y eliminación (CRUD) de los archivos y carpetas dentro de HDFS.

6.2.3. Instalación y Configuración

Para este proyecto Apache Hadoop será instalado en modo pseudo-distribuido, donde se crean dos procesos Java uno que se encarga del rol de Datanode y otro del Namenode, a continuación se presentan las características del equipo:

Paso 1: Configuración SSH y la generación de claves

Se generan un par de claves SSH, se agrega la clave pública a la lista de claves autorizadas y se asignan permisos de lectura y escritura al archivo

Componente	Especificación
Sistema Operativo	Ubuntu 16.04 64 bits
Memoria	4 GB
Procesador	Intel Core i5-3470 CPU @ 3.20GHz x 4
Disco Duro	500 GB

Cuadro 6.3: Especificaciones de equipo Apache Drill

de claves autorizadas haciendo uso de los siguientes comandos:

```
ssh-keygen -t rsa

cat ~/.ssh/id_rsa.pub >>~/.ssh/authorized_keys

chmod 0600 ~/.ssh/authorized_keys
```

Paso 2: Instalación de Java

Se realiza la descarga del JDK desde la pagina de Oracle o haciendo uso del siguiente comando:

```
wget http://download.oracle.com/otn-pub/java/jdk/8u131-b11/
d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.tar.gz
```

Se decomprime el archivo .tar.gz en el directorio desde donde se necesite ser ejecutado, en este caso en `/usr/local/hadoop` con el siguiente comando:

```
tar -xvzf jdk-8u131-linux-x64.tar.gz
```

Por último debemos añadir las siguientes líneas al archivo `~/.bashrc`:

```
export JAVA_HOME=/usr/local/jdk1.8.0_131
export PATH=PATH:$JAVA_HOME/bin
```

Paso 3: Descarga de Apache Hadoop

Se realiza la descarga desde la página de Apache Hadoop a través de la terminal de comandos

```
wget http://www-us.apache.org/dist/hadoop/common/
hadoop-2.7.3/hadoop-2.7.3.tar.gz
```

Paso 4: Descomprimir el archivo .tar.gz de Apache Hadoop

Se descomprime el archivo .tar.gz en el directorio desde donde se necesite ser ejecutado, en este caso en `/usr/local/hadoop` con el siguiente comando:

```
tar -xvzf hadoop-2.7.3.tar.gz
```

Paso 5: Configuración del entorno

Añadir al archivo ~/.bashrc las siguientes líneas:

```
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
```

Paso 6: Configuración de Apache Hadoop

En la carpeta `/usr/local/hadoop/etc/hadoop` se encuentran los archivos de configuración de Hadoop.

Lo primero es modificar en el archivo `hadoop-env.sh` la dirección a la implementación de Java que debe utilizar Hadoop:

```
export JAVA_HOME=/usr/local/jdk1.8.0_131
```

Lo segundo es dentro del archivo `core-site.xml` añadir dentro de las etiquetas <configuration></configuration> lo que sigue:

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

Lo tercero es dentro del archivo `hdfs-site.xml` añadir dentro de las etiquetas <configuration></configuration> lo que sigue:

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

   <property>
<name>dfs.name.dir</name>
<value>file:///usr/local/hdfs/namenode</value>
</property>

   <property>
<name>dfs.data.dir</name>
<value>file:///usr/local/hdfs/datanode</value>
</property>

   <property>
<name>dfs.webhdfs.enabled</name>
<value>true</value>
</property>
```

Lo cuarto es dentro del archivo `yarn-site.xml` añadir dentro de las etiquetas <configuration></configuration> lo que sigue:

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

Lo quinto es dentro del archivo `mapred-site.xml` añadir dentro de las etiquetas `<configuration></configuration>` lo que sigue:

```
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
```

Paso 7: Instalación del Namenode

En la consola debemos ejecutar el siguiente comando:

```
hdfs namenode -format
```

Paso 8: Iniciar el Sistema de Archivos de Hadoop

En la consola debemos ejecutar el siguiente comando:

```
start-dfs.sh
```

6.2.4. Pruebas

Las pruebas realizadas en esta iteración se hicieron luego de que todas las configuraciones anteriores fueron realizadas exitosamente y con la finalidad de comprobar su correcto funcionamiento. A continuación se detallan las mismas en siguiente Cuadro:

ID	Descripción del caso de prueba	Resultado Esperado	Resultado Obtenido
1	Se utilizan el comando `hadoop fs -ls /` de Hadoop para realizar consultar a la raíz del sistema de archivos.	Se espera que liste los elementos que se encuentran en la raíz del sistema de archivos.	En principio no hubo una respuesta visible dado que el sistema de archivos está vacío.
2	Se creará un directorio dentro de la raíz del sistema de archivos de Hadoop a través del REST API de Hadoop.	Se espera que se cree un directorio en la raíz del sistema de archivos de Hadoop.	Al ejecutar el comando obtuvimos respuesta HTTP 200, que indica que se creó de manera exitosa. (Ver figura 6.2)
3	Se elimina el directorio dentro de la raíz del sistema de archivos de Hadoop a través del REST API de Hadoop creado en la prueba anterior.	Se espera que se elimine el directorio en la raíz del sistema de archivos de Hadoop.	Al ejecutar el comando obtuvimos respuesta HTTP 200, que indica que se eliminó de manera exitosa. (Ver figura 6.3)

Cuadro 6.4: Casos de Prueba Apache Hadoop

Figura 6.6: Respuesta de Prueba #2 de Apache Hadoop

Figura 6.7: Respuesta de Prueba #3 de Apache Hadoop

6.3. Iteración 2: Instalación y Configuración del Apache Drill

En esta iteración los desarrolladores realizan la instalación y configuración de la herramienta Apache Drill, capaz de manipular los conjuntos de datos alojados en el clúster. Además de documentarse con su REST API para su posterior uso desde la aplicación web.

6.3.1. Planificación

En el siguiente cuadro se muestran las historias de usuario desarrollada en esta iteración.

ID	Fecha	Descripción
6	13/01/2017	El usuario debe modificar los conjuntos de datos alojados en el clúster.
7	13/01/2017	Se debe usar una herramienta que ofrezca un REST API para ser manipulada desde la aplicación web a desarrollar.

Cuadro 6.5: Historias de usuario. Iteración 2.

6.3.2. Diseño

Para realizar la manipulación de los conjuntos de datos alojados en el clúster se debe utilizar una herramienta que mediante comandos likeSQL realice consultas sobre dichos conjuntos y extraiga la información de la manera que lo solicite el analista de datos.
Apache Drill resuelve este problema utilizando un lenguaje de consulta para extraer la data de los conjuntos de datos, ademas estos comandos de consulta pueden ser ejecutados desde la aplicación web mediante el REST API que proporciona Apache Drill.

6.3.3. Instalación y Configuración

Para este proyecto Apache Drill será instalado en modo embebido, donde el mismo se ejecuta en el sistema operativo donde se realizó la instalación a continuación se presentan las características del equipo:

Componente	Especificación
Sistema Operativo	Ubuntu 16.04 64 bits
Memoria	4 GB
Procesador	Intel Core i5-3470 CPU @ 3.20GHz x 4
Disco Duro	500 GB

Cuadro 6.6: Especificaciones de equipo Apache Drill

Paso 1: Descarga de Apache Drill

Se realiza la descarga desde la página de Apache Drill a través de la terminal de comandos

```
wget http://apache.mirrors.hoobly.com/drill/drill-1.10.0/
apache-drill-1.10.0.tar.gz
```

Paso 2: Descomprimir el archivo .tar.gz de Apache Drill

Se descomprime el archivo .tar.gz en el directorio desde donde se necesite ser ejecutado, en este caso en `/usr/local/drill` con el siguiente comando:

```
tar -xvzf apache-drill-1.10.0.tar.gz
```

Paso 3: Ejecutar Apache Drill embebed mode

Para esto se debe estar posicionado en el directorio donde se descomprimió el paquete .tar.gz, en este caso en `/usr/local/drill` y allí se ejecuta el siguiente comando:

```
./bin/drill-embedded
```

Al ejecutar este comando se abre la línea de comandos de apache drill como podemos observar en la siguiente imagen:

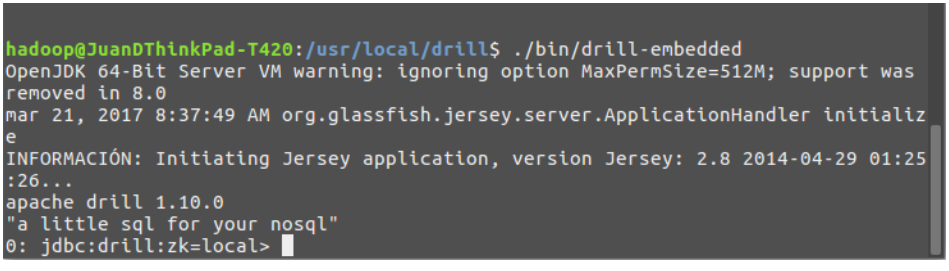

Figura 6.8: Ejecución de Apache Drill desde la línea de comandos

Paso 4: Acceder a la interfaz web de Apache Drill

Se abre una instancia del navegador y le coloca la siguiente dirección: `localhost:8047` allí se despliega la interfaz siguiente:

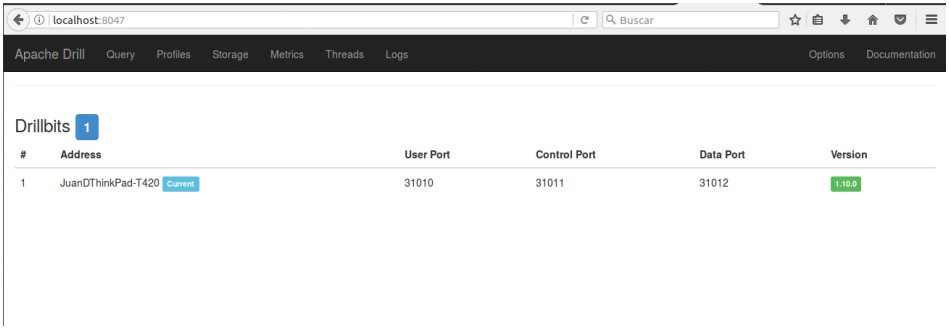

Figura 6.9: Interfaz web de Apache Drill

Paso 5: Configurar el plugin de almacenamiento de Apache Drill

La configuración de este plugin de almacenamiento establece todas las directrices del tratamiento de los archivos, desde las extensiones que van a ser permitidas como los directorios del clúster hadoop donde puede acceder, escribir o eliminar.
Para esto se debe acceder a la sección de storage como se puede observar en la siguiente imagen:

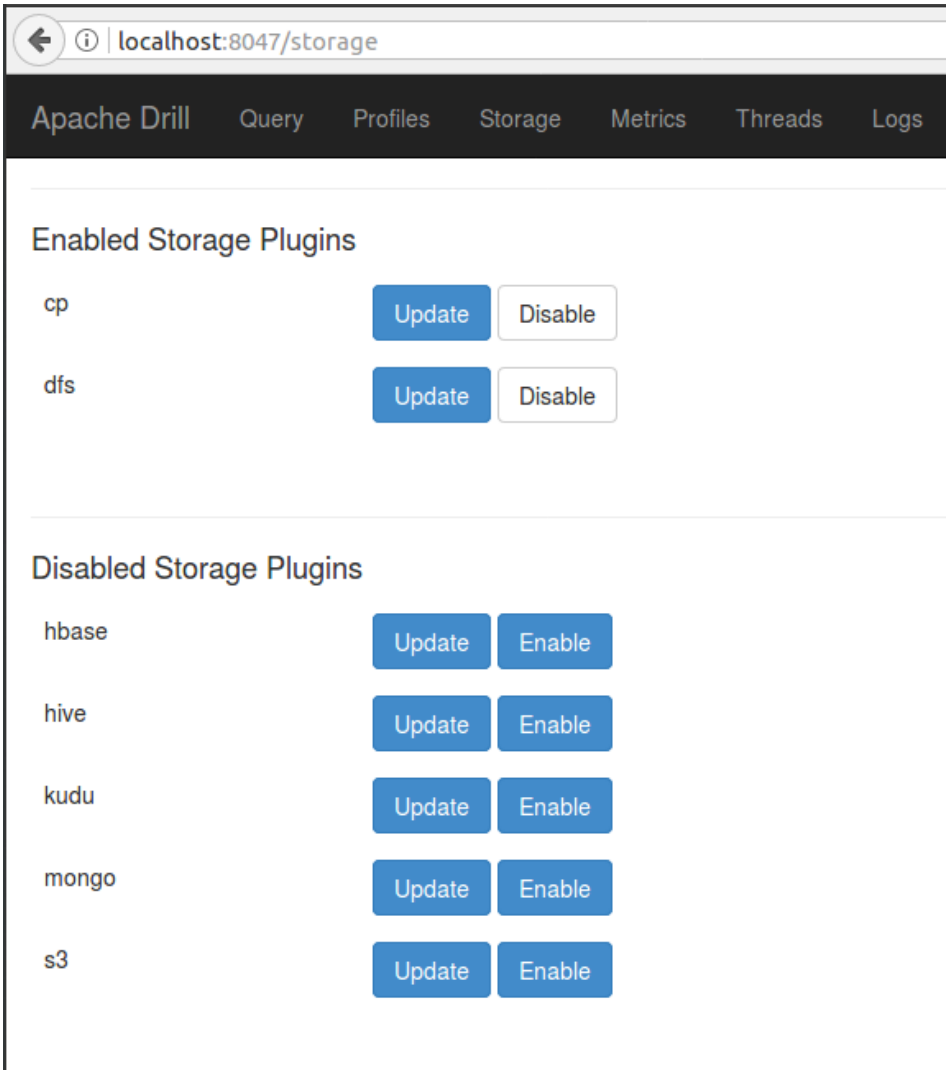

Figura 6.10: Sección de Storage de Interfaz web de Apache Drill

Allí se selecciona realizar Update sobre el plugin dfs para trabajar sobre el clúster de haddop, luego de esto se abrirá una interfaz como la siguiente imagen:

Figura 6.11: Editar la configuración del plugin dfs

La configuración realizada sobre el plugin se presenta a continuación en este archivo .json:

```
{
  "type": "file",
  "enabled": true,
  "connection": "hdfs://localhost:9000/",
  "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": ","
    }
  }
}
```

Como podemos observar se colocó en 'connection' la dirección y el puerto donde está contestando el clúster de hadoop, también dentro de los 'workspaces' y 'root' se colocó la propiedad 'writable' como true para poder leer y escribir en esos directorios de HDFS. Por último como se consideró el trabajo con archivos planos CSV con headers para colocar los nombres de los campos, se le colocó la propiedad 'extractHeader' como true.
Por último se estableció el uso de dfs como plugin predeterminado, a continuación podemos observar como se realizó este paso vía línea de comandos:

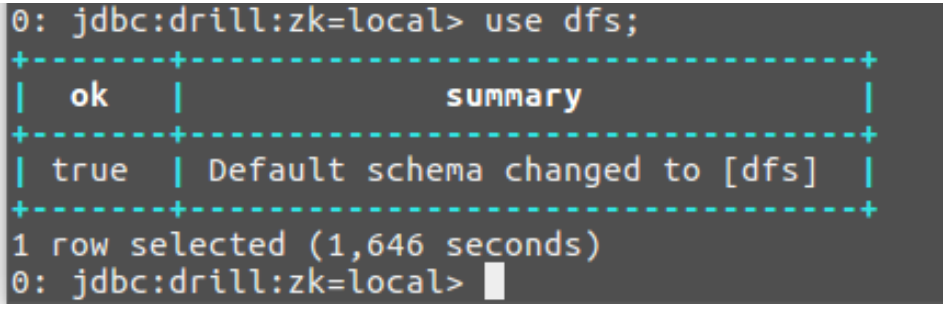

Figura 6.12: Establecer dfs como plugin de uso por defecto

6.3.4. Pruebas

Las pruebas realizadas en esta iteración se hicieron luego de que todas las configuraciones anteriores fueron realizadas exitosamente y con la finalidad de comprobar su correcto funcionamiento. A continuación se detallan las mismas en siguiente cuadro:

ID	Descripción del caso de prueba	Resultado Esperado	Resultado Obtenido
1	Se realiza una consulta a través del REST API de Apache Drill a un conjunto de datos alojados en el clúster.	Se espera que responda la consulta con los datos solicitados.	Respuesta con los datos solicitados. Como prueba de esto demuestra en la imagen a presentar a continuación. (Ver figura 6.9)

Cuadro 6.7: Casos de Prueba Apache Drill

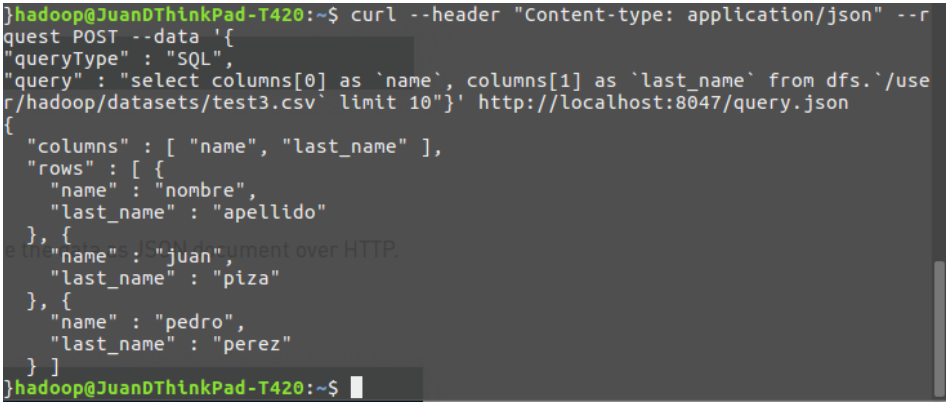

Figura 6.13: Respuesta de Prueba de Apache Drill

6.4. Iteración 3: Definición de Modelo de Datos de la Aplicación Web

Luego de tener el ambiente de trabajo instalado con el clúster podemos comenzar a desarrollar la aplicación con el framework Meteor y definir como será el modelo de datos que tendrá la aplicación web.

6.4.1. Planificación

En el siguiente cuadro se muestran las historias de usuario desarrollada en esta iteración.

ID	Fecha	Descripción
8	27/01/2017	El usuario puede añadir cualquier conjunto de datos desde su computadora local al clúster, siempre y cuando el mismo tenga formato CSV y tenga en los encabezados los nombres de los campos.
9	27/01/2017	El usuario puede crear proyectos para trabajar con el conjunto de datos que añadió al clúster, pueden existir varios proyectos por conjunto de datos y en un proyecto solo puede ser tratado un conjunto de datos.
10	27/01/2017	El usuario decide si el conjunto de datos es privado o público, de ser público cualquier otro analista puede crear un proyecto para trabajar con el conjunto.

Cuadro 6.8: Historias de usuario. Iteración 3.

6.4.2. Diseño

Para definir el modelo de datos a trabajar con la aplicación se debe comenzar tomando en cuenta que la base de datos que se utilizará para guardar la metadata de la aplicación es una base de datos NoSQL orientada a documentos, por lo tanto no se rige por un modelo relacional, en este caso las tablas que se pudieran presentar para guardar los objetos son colecciones (archivos .json), que gracias al framework permite la definición previa de un esquema de la colección, donde se pueden definir los tipos de datos de cada uno de sus campos como otras características de valores de asignación automáticos y si el mismo puede ser o no un valor nulo.

6.4.3. Implementación

A continuación se presentan los esquemas que fueron definidos según las colecciones a utilizar en la aplicación:

- **Colección: DataSets**

```
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**

```
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',
        },
});
```

- **Colección: Columns**

```
ColumnsSchema = new SimpleSchema({
        datasetId:{
                type: String,
                label: 'DatasetId'
        },
        name:{
                type: String,
                label:'Name'
        },
        dataType:{
                type: String,
                label:'DataType'
        },
        createdAt:{
                type: Date,
                label: 'CreatedAt',
                autoValue: function(){
                        return new Date()
                },
        }
});
```

6.5. Iteración 4: Cuentas de Usuario y Permisología

En esta iteración se establece la permisología de los usuarios dentro de la aplicación.

6.5.1. Planificación

En el siguiente cuadro se muestran las historias de usuario desarrollada en esta iteración.

ID	Fecha	Descripción
11	03/02/2017	El usuario debe estar registrado y autenticado en la aplicación para poder ingresar a todas sus funcionalidades.

Cuadro 6.9: Historias de usuario. Iteración 4.

6.5.2. Diseño

Para llevar a cabo el registro y la autenticación del usuario se partió de la premisa que el mismo será identificado en la aplicación a través de su correo electrónico, donde el mismo debe ser único. A continuación se presentan los prototipos diseñados para realizar las secciones de autenticación y registro en la herramienta:

- **Autenticación**

Figura 6.14: Autenticación de Usuario

- **Registro**

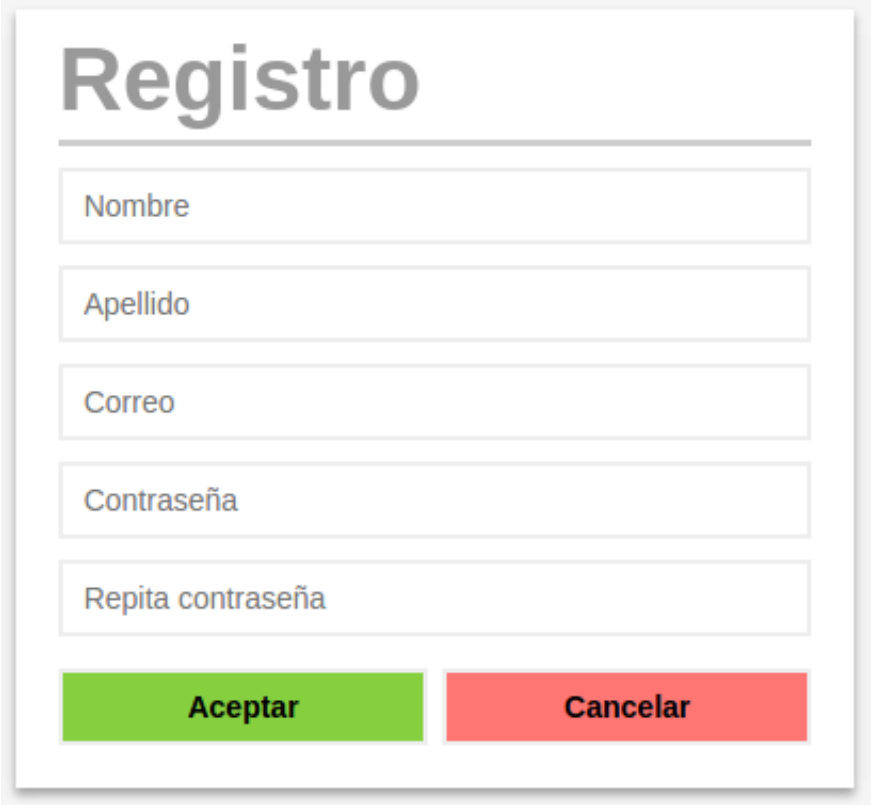

Figura 6.15: Registro de Usuario

6.6. Iteración 5: Manipulación de Conjuntos de Datos y Proyectos

En esta iteración se establece el flujo de trabajo de la manipulación de los conjuntos de datos y proyectos creados por la aplicación.

6.6.1. Planificación

En el siguiente cuadro se muestran las historias de usuario desarrollada en esta iteración.

ID	Fecha	Descripción
12	03/02/2017	El usuario solo puede acceder a los conjuntos de datos públicos o creados por él mismo.
13	03/02/2017	El usuario solo puede acceder a los proyectos creados por él mismo.
14	03/02/2017	El usuario para crear un proyecto debe elegir un conjunto de datos para trabajarlo.
15	03/02/2017	El usuario puede modificar la información de los conjuntos de datos y proyectos.

Cuadro 6.10: Historias de usuario. Iteración 5.

6.6.2. Diseño

Para añadir, modificar y eliminar conjuntos de datos y proyectos se elaboraron interfaces para esta manipulación. A continuación se muestran los prototipos relacionados a la manipulación de los conjuntos de datos y los proyectos realizados por el usuario:

- **Listado**

Figura 6.16: Listado de Proyectos

Figura 6.17: Listado de Conjuntos de datos

- **Creación**

Figura 6.18: Creación de Proyectos

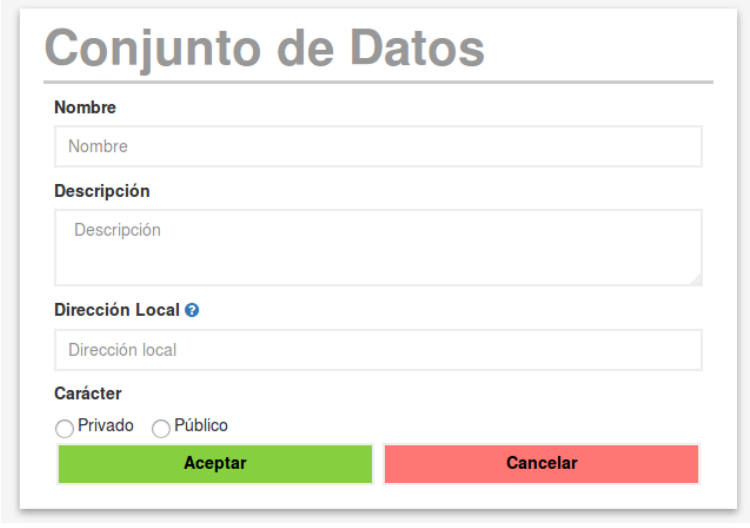

Figura 6.19: Creación de Conjuntos de datos

- **Editar**

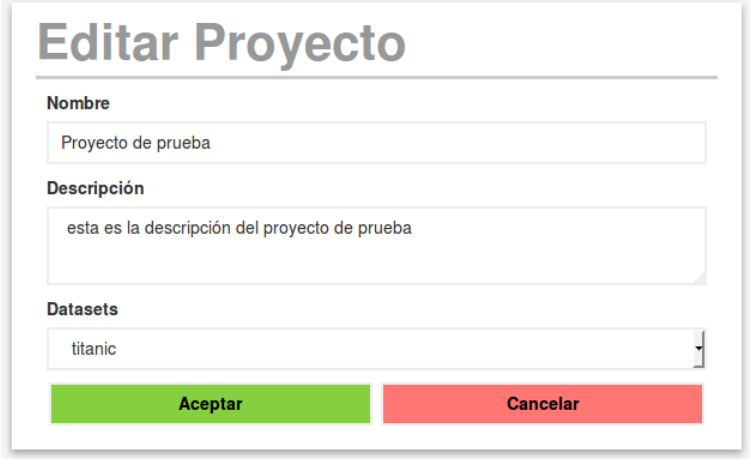

Figura 6.20: Edición de Proyectos

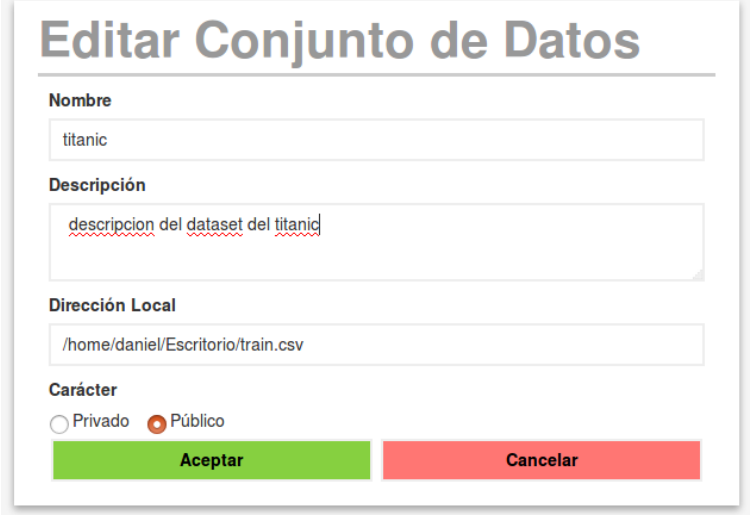

Figura 6.21: Edición de Conjuntos de datos

- **Eliminar**

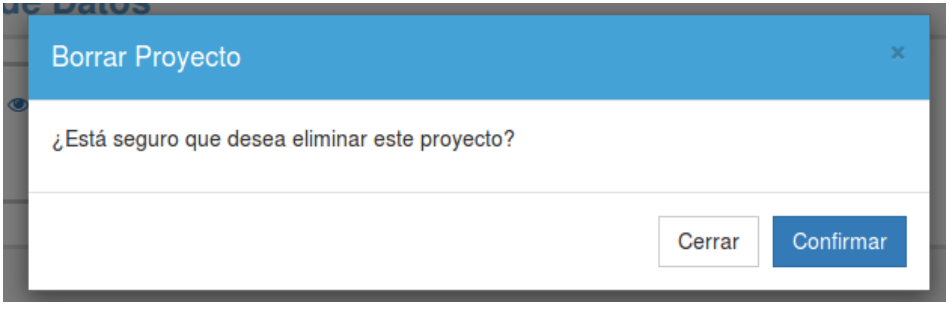

Figura 6.22: Eliminación de Proyectos

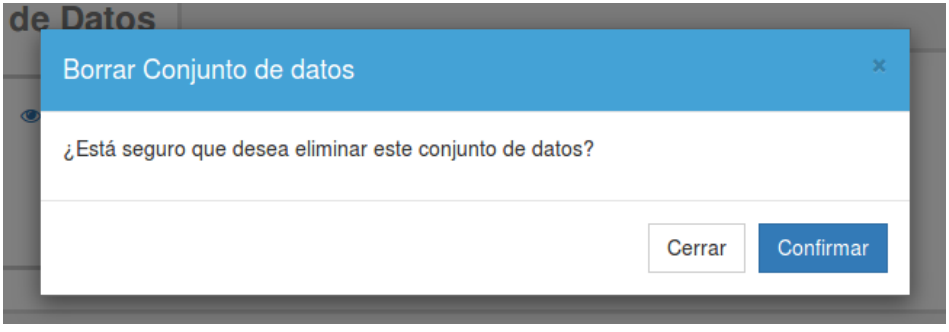

Figura 6.23: Eliminación de Conjuntos de datos

- **Detalle**

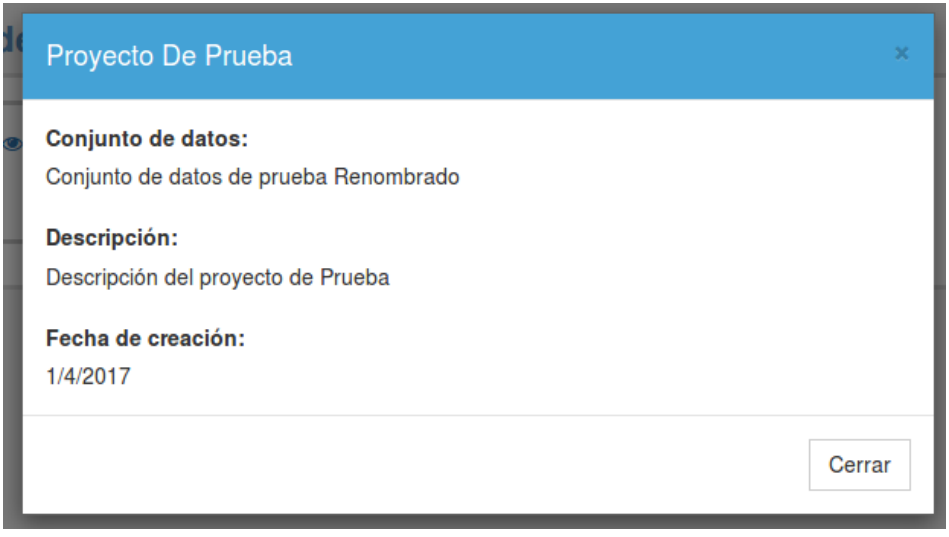

Figura 6.24: Detalle de un Proyecto

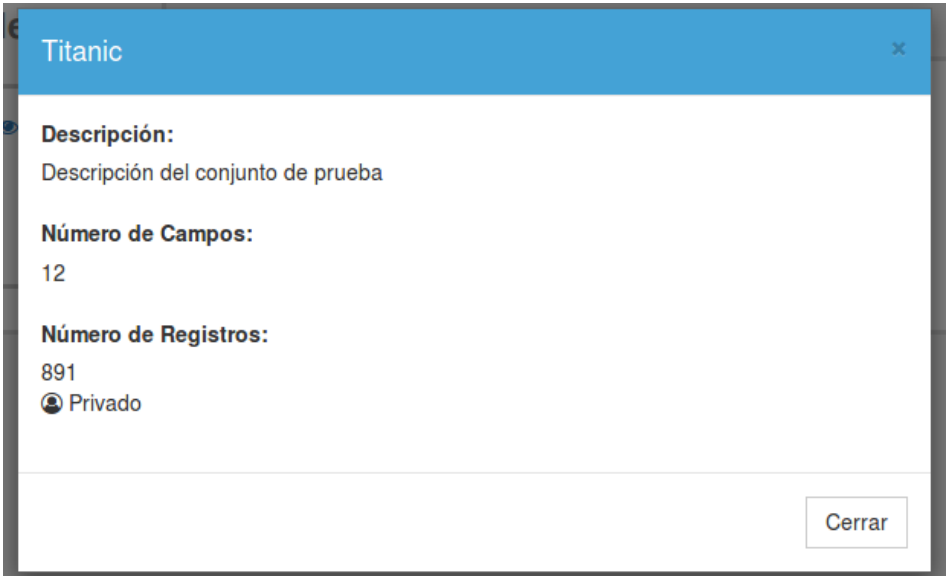

Figura 6.25: Detalle de un Conjuntos de datos

6.6.3. Codificación

En el siguiente código se muestra la función que se encarga de añadir el conjunto de datos que se encuentra en el sistema de archivos local a HDFS:

Añadir un conjunto de datos a HDFS

```
createDatasetInHDFS: (nameFile,dirLocal,dirHdfs) =>
{
  check(nameFile, String);
  var WebHDFS = require('webhdfs');
  var fs = require('fs');
  var hdfs = WebHDFS.createClient({
    user: cluster_user,
    host:  $host,
    port: $port
  });

  var localFileStream = fs.createReadStream(dirLocal);
  var remoteFileStream = hdfs.createWriteStream(dirHdfs);
  var response = true;

  localFileStream.pipe(remoteFileStream);

  // Handle errors
  remoteFileStream.on('error',
    function onError (err) {
      // Do something with the error
      console.log(err);
      return false;
  });
  // Handle finish event
  remoteFileStream.on('finish',
    function onFinish (res) {
      // Upload is done
      console.log('uploaded');
      // response = true;
      return true;
  });
```

```
  return response;
}
```

A continuación se muestra el código para crear el proyecto partiendo del conjunto de datos alojados en HDFS:

Crear Proyecto en HDFS

```
copyDatasetInFolderProject: function (folder_project,
dataset_address){
var result = {'error': true};
try {
        var result0 = HTTP.call("POST",
        "http://localhost:8047/query.json",
        { data: {
                "queryType" : "SQL",
                "query" : "use dfs"
                }
        });
        console.log(result0.data.rows[0].ok);
        if (result0.data.rows[0].ok == 'true'){
                var result1 = HTTP.call("POST",
                "http://localhost:8047/query.json",{
                data: {
                        "queryType" : "SQL",
                        "query" : "alter session set
                        `store.format`='parquet'"
                }
                });

                if(result1.data.rows[0].ok == 'true'){
                        var queryCopy = "create table
                        dfs.root.`"+ folder_project+
                        "/raw` as select row_number()
                        over(partition by 1) as
                        row_num,* from dfs.`"+
                        dataset_address+"`";
                        var result2 = HTTP.call("POST",
```

```
                        "http://localhost:8047/
                        query.json",{
                        data: {
                                "queryType" : "SQL",
                                "query" : queryCopy
                        }
                        });
                        result = result2;
                }
        }
        return result;

} catch (e) {
        console.log(e);
        return e;
}
}
```

6.6.4. Pruebas

Las pruebas realizadas con la finalidad de comprobar el correcto funcionamiento de esta etapa del proceso de trabajo. A continuación se detallan las mismas en siguiente cuadro:

ID	Descripción del caso de prueba	Resultado Esperado	Resultado Obtenido
1	Se realiza la creación de un conjunto de datos de prueba.	Se espera que desde la página de creación se redirija a la página del listado y se muestre el nuevo conjunto de datos en el listado.	Se creó el conjunto de datos de manera exitosa como se puede observar en la Figura 6.23 del detalle del conjunto de datos.
2	Se realiza la modificación de la información de un conjunto de datos de prueba.	Se espera que desde la página de edición se redirija a la página del listado y se muestre el conjunto de datos con la información modificada.	Se actualizo el conjunto de datos de manera exitosa como se puede observar en la Figura 6.25 del detalle del conjunto de datos editado.
3	Se realiza la creación de un proyecto de prueba.	Se espera que desde la página de creación se redirija a la página de Preparación de datos.	Se creó el proyecto de manera exitosa como se puede observar el la Figura 6.27 donde se muestra la vista de Preparación de datos.

Cuadro 6.11: Casos de Prueba CRUD de conjunto de datos y proyectos

Figura 6.26: Respuesta de prueba #1-1

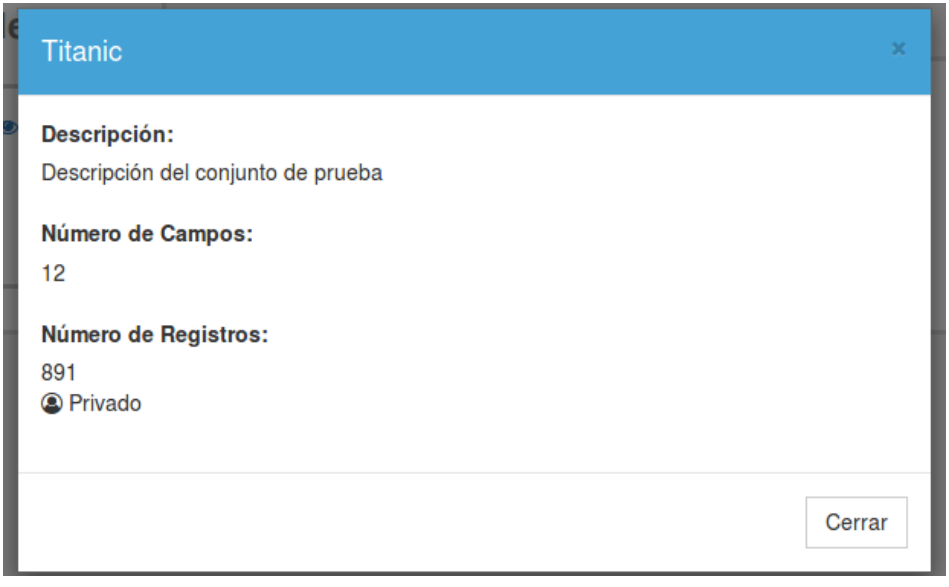

Figura 6.27: Respuesta de prueba #1-2

Figura 6.28: Respuesta de prueba #2-1

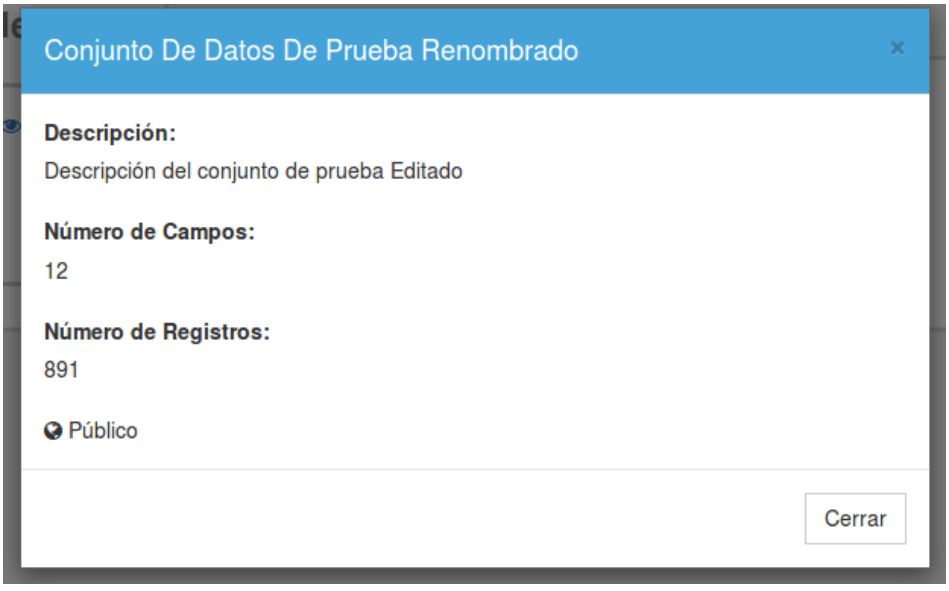

Figura 6.29: Respuesta de prueba #2-2

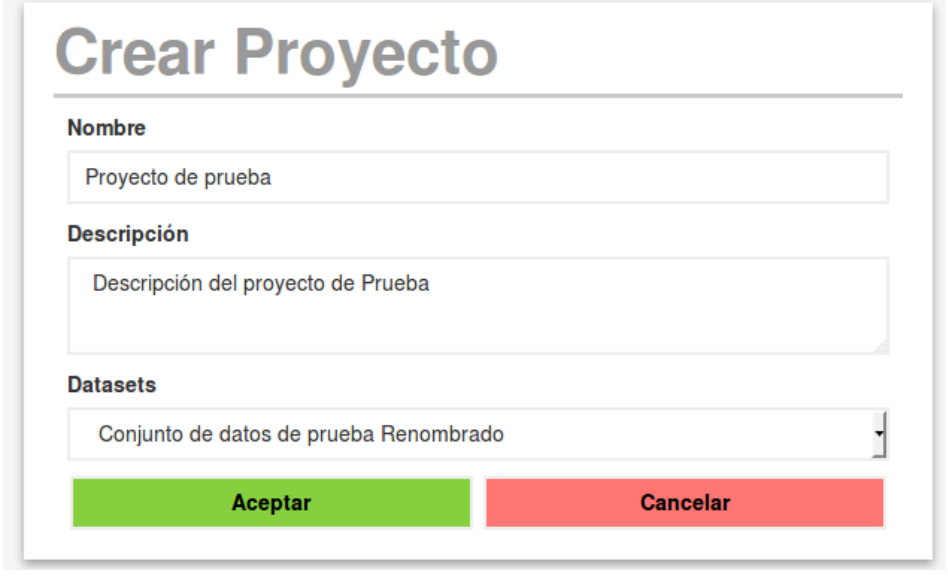

Figura 6.30: Respuesta de prueba #3-1

Figura 6.31: Respuesta de prueba #3-2

6.7. Iteración 6: Módulos de la aplicación

En esta iteración se establecen los módulos de la aplicación.

6.7.1. Planificación

En el siguiente cuadro se muestran las historias de usuario desarrollada en esta iteración.

ID	Fecha	Descripción
16	03/02/2017	La aplicación está dividida en 3 módulos que engloban pasos del proceso de KDD. (Preparación, Modelado y Visualización)
17	03/02/2017	El módulo de preparación debe presentar un dashboard con las funcionalidades necesarias para realizar la limpieza y transformación de los datos.

Cuadro 6.12: Historias de usuario. Iteración 6.

6.7.2. Diseño

Para realizar todo el proceso KDD el mismo debe dividirse en cada una de sus etapas, la aplicación debe mantener este flujo para ofrecer una herramienta que sea eficiente al analista de datos, como se establece en el alcance del proyecto el mismo va desde la extracción de la data, pasando por su preparación hasta llegar a obtener una vista minable, a continuación se presenta el prototipo diseñado para la interfaz de usuario durante la tarea de preparación de datos:

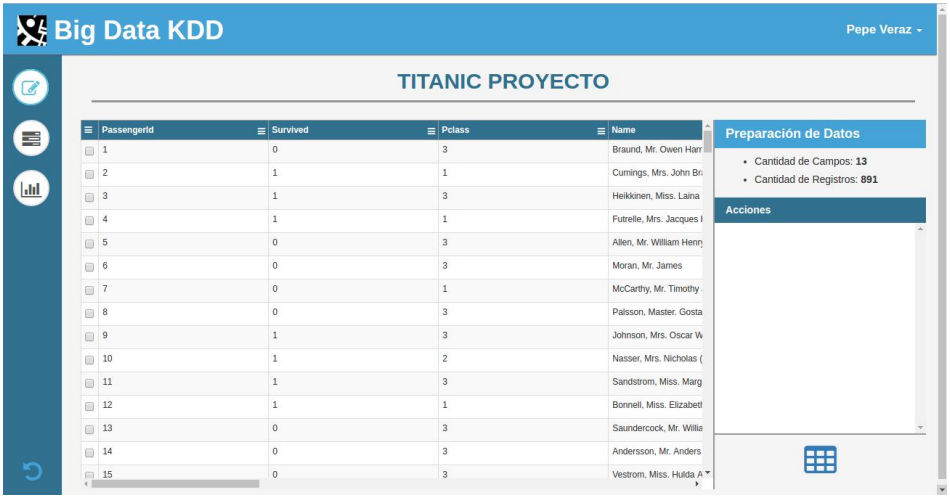

Figura 6.32: Interfaz de Usuario de Módulo de Preparación de Datos

Se puede observar la simplicidad de la interfaz donde se tiene el nombre del proyecto y el área principal de trabajo con los datos del conjunto, ademas a la derecha aparecen la cantidad de campos y registros que actualmente tiene el proyecto y la lista de las acciones de preparación que se van ejecutando.

6.8. Iteración 7: Tipos de Datos del Conjunto

En esta iteración se establece el manejo de los tipos de datos de los campos del proyecto en la etapa de preparación.

6.8.1. Planificación

En el siguiente cuadro se muestran las historias de usuario desarrollada en esta iteración.

ID	Fecha	Descripción
18	10/02/2017	La aplicación debe detectar automáticamente el tipo de dato de cada campo del conjunto a trabajar en el proyecto.
19	10/02/2017	El usuario debe tener la capacidad de elegir o cambiar el tipo de dato del campo en la aplicación.
20	10/02/2017	Los tipos de datos para trabajar en esta primera versión de aplicación serán Entero, Decimal o Caracteres.

Cuadro 6.13: Historias de usuario. Iteración 7.

6.8.2. Diseño

Se agregaron etiquetas a los campos al desplegar el menú donde se identifica el tipo de dato, a continuación se puede observar diferentes tipos de datos:

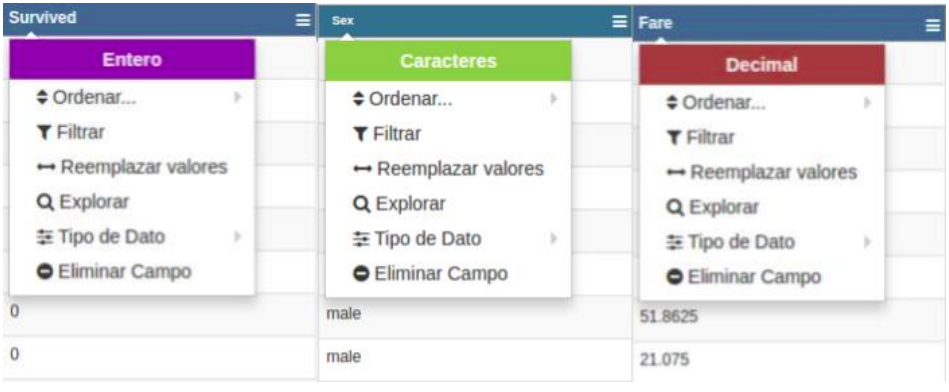

Figura 6.33: Etiqueta de tipo de dato del campo.

Para que el usuario realice el cambio en el tipo de dato según desee se agregó al menú una opción en el menú llamada "tipo de dato", donde el

usuario puede elegir el tipo de dato como quiere que sea tratado el campo, a continuación se presenta el prototipo con el menú desplegado para el tipo de dato a elegir:

Figura 6.34: Cambiar el tipo de dato de un campo.

6.8.3. Codificación

En el siguiente código se muestra la función que clasifica el tipo de dato de cada campo del conjunto:

Clasificar el tipo de dato de cada campo del conjunto

```
function compareDataTypes(columns, array, factor, dsId){
        let results = [];
        let c_to_insert = [];
        for(let i in columns){
                let int = isInt(columns[i],array);
                let double = isDouble(columns[i],array);

                if (int >= factor) {
                        results[columns[i]] = "Entero";
                } else if (double >= factor) {
```

```
                        results[columns[i]] = "Decimal";
                } else {
                        results[columns[i]] = "String";
                }

                let column =
                {
                        datasetId: dsId,
                        name: columns[i],
                        dataType: results[columns[i]]
                }

                c_to_insert.push(column);
        }

        console.log(c_to_insert);

        return c_to_insert;
};

function isInt(column, array) {
        let pattern = /^[\d]*$/;
        let results = 0;
        for (let i in array) {
                if (pattern.test(array[i][column])){
                        results++;
                }
        }
        return results;
};

function isDouble(column, array) {
        let pattern = /^([\d]+[\.,][\d]+|[\d]*)$/;
        let results = 0;
        for (let i in array) {
                if (pattern.test(array[i][column])){
```

```
47                              results++;
48                      }
49              }
50              return results;
51      };
```

6.8.4. Pruebas

Las pruebas realizadas con la finalidad de comprobar el correcto funcionamiento de todo el manejo de los tipos de datos de los conjuntos. A continuación se detallan las mismas en siguiente cuadro:

ID	Descripción del caso de prueba	Resultado Esperado	Resultado Obtenido
1	Se crea un proyecto a partir de un conjunto de datos para verificar que los tipos de datos asignados automáticamente sean correctos	Todos los campos se le asigne el tipo de dato correctamente	Se realizó la asignación de tipo de dato de todos los campos de forma correcta
2	Se realiza la modificación del tipo de dato de un campo de Decimal a String	Modificado correctamente	Fue modificado correctamente a String

Cuadro 6.14: Casos de Prueba de tipos de datos de los campos de un conjunto

6.9. Iteración 8: Funciones de preparación de datos

En esta iteración se desarrollan las funciones que se realizan sobre los datos para realizar la preparación de la vista minable.

6.9.1. Planificación

En el siguiente cuadro se muestran las historias de usuario desarrollada en esta iteración.

ID	Fecha	Descripción
21	24/02/2017	El usuario puede ordenar el conjunto de datos por un campo de menor a mayor o de mayor a menor.
22	24/02/2017	El usuario puede filtrar los registros por cierto campo y dependiente de su tipo de dato.
23	24/02/2017	El usuario puede reemplazar valores de un campo.
24	24/02/2017	El usuario puede eliminar campos o registros.
25	24/02/2017	El usuario puede explorar un campo conociendo la cantidad de repetición de datos o máximos y mínimos valores.

Cuadro 6.15: Historias de usuario. Iteración 8.

6.9.2. Diseño

Todas las tareas y funciones necesarias para la preparación de los datos fueron centralizadas en un menú, donde el usuario ingresa por campo y selecciona la función a aplicar sobre el mismo:

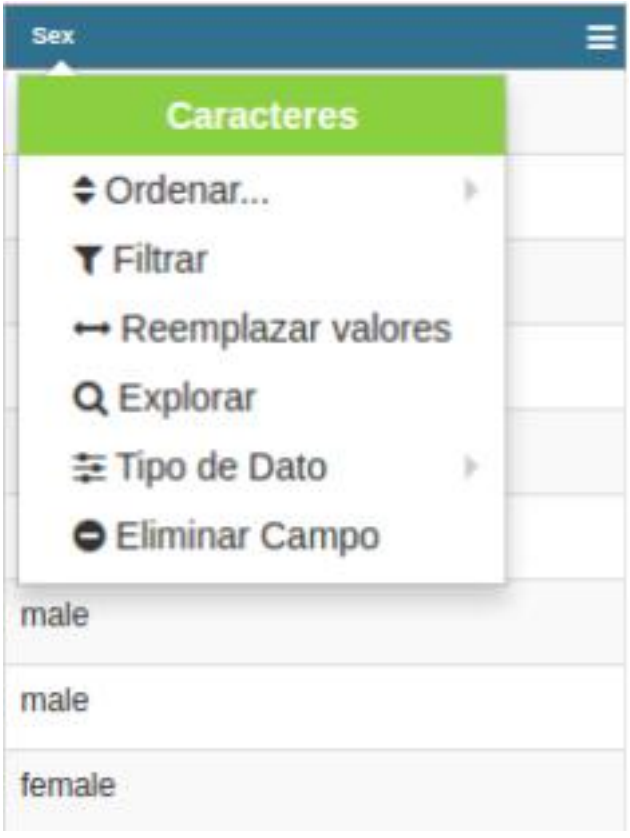

Figura 6.35: Menú de funciones para la preparación de los datos.

6.9.3. Codificación

A continuación se mostrará el código de las funciones de preparación sobre el conjunto de datos, las mismas programadas en JavaScript y con el uso del REST API de Apache Drill y Apache Hadoop WebHDFS:

- **Ordenamiento:**

Ordenamiento por campo

```
queryDataDrillOrderBy: function(column,data_type,
order,data_address){
        if (order == 'asc'){
            var queryCopy="select * from dfs.`"
            +data_address+"/*` order by
            (case when "+column+" = '' then
            CAST(NULL AS "+data_type+")
            else CAST("+column+" AS "+data_type+")
            end) "+order+" nulls first";
        }else{
            var queryCopy = "select * from dfs.`"
            +data_address+"/*` order by
            (case when "+column+" = '' then
            CAST(NULL AS "+data_type+")
            else CAST("+column+" AS "+data_type+")
            end) "+order+" nulls last";
        }

        try {
            var result = HTTP.call("POST",
            "http://localhost:8047/query.json",{
            data: {
                    "queryType" : "SQL",
                    "query" : queryCopy
            }
            });
            return result;

        } catch (e) {
            console.log(e);
            return e;
        }
```

```
},
```

- **Filtrado:**

Filtrado por campo

```
queryDataDrillFilterBy:function(new_version_address,
old_version_address,column,filter,value) {
        var queryCopy = "create table dfs.root.`"+
        new_version_address+
        "` as select * from dfs.`"+
        old_version_address+"/*` where "
        +column+" "+filter+" "+value+"";
        try {
                var result = HTTP.call("POST",
                "http://localhost:8047/query.json",
                {
                data: {
                        "queryType" : "SQL",
                        "query" : queryCopy
                }
        });
                return result;

        } catch (e) {
                console.log(e);
                return e;
        }

},
```

- **Reemplazar Valores:**

Reemplazo de Valores

```
replaceValues:(new_version_address,
old_version_address, column,
initVal, postVal, order) => {
try{

        var queryCopy = "select * from dfs.`"+
        old_version_address+"`";

        var result = HTTP.call("POST",
        "http://localhost:8047/query.json",{
                data: {
                        "queryType" : "SQL",
                        "query" : queryCopy
                }
        });

        var rows = result.data.rows;
        for (let i in rows){
                if (rows[i][column] == initVal) {
                        rows[i][column] = postVal;
                }
        }

        result = HTTP.call("PUT",
        "http://"+hadoop_host+"/webhdfs/v1"+
        cluster_root+"/tmp/0_0_0.json?op=CREATE&
        overwrite=true&user.name="+cluster_user);
        result = HTTP.call("PUT",
        result.headers.location);
        result = HTTP.call("POST",
        "http://"+hadoop_host+"/webhdfs/v1"+
        cluster_root+"/tmp/0_0_0.json?op=APPEND&
        user.name="+cluster_user);
```

```
        result = HTTP.call("POST",
        result.headers.location,
        {data: rows});


        queryCopy = "create table dfs.root.`"+
        new_version_address+"` as  select "+order+
        " from dfs.`"+cluster_root+
        "/tmp/0_0_0.json`";

        result = HTTP.call("POST",
        "http://localhost:8047/query.json",{
                data: {
                        "queryType" : "SQL",
                        "query" : queryCopy
                }
        });

        return result;

} catch (e){
        console.log(e);

        return e;
}
}
```

- **Explorar Datos:**

Explorar datos por Campo

```
queryExploreString: function (column,
project_address){
        var queryCopy = "select "+column+", count(*)
        as Cantidad from dfs.`"+project_address+
        "` group by "+column+" order by Cantidad
        desc limit 30";
```

```
        try {
                var result = HTTP.call("POST",
                "http://localhost:8047/query.json",
                {
                        data: {
                        "queryType" : "SQL",
                        "query" : queryCopy
                        }
                });

                return result;

        } catch (e) {
                console.log(e);
                return e;
        }
},

queryDataDrillMaxValue: function (field,data_type,
data_address){
        var queryCopy = "select max(cast("+field+
        " as "+data_type+")) from dfs.`"+
        data_address+"/*` limit 1";
        try {
                var result = HTTP.call("POST",
                "http://localhost:8047/query.json",
                {
                data: {
                        "queryType" : "SQL",
                        "query" : queryCopy
                }
        });
                return result;

        } catch (e) {
                console.log(e);
```

```
43                  return e;
44          }
45  },
46
47  queryDataDrillMinValue: function (field,data_type,
48  data_address){
49          var queryCopy = "select min(cast("+field+
50          " as "+data_type+")) from dfs.`"+
51          data_address+"/*` limit 1";
52          console.log(queryCopy);
53          try {
54                  var result = HTTP.call("POST",
55                  "http://localhost:8047/query.json",
56                  {
57                  data: {
58                          "queryType" : "SQL",
59                          "query" : queryCopy
60                  }
61          });
62                  return result;
63
64          } catch (e) {
65                  console.log(e);
66                  return e;
67          }
68  },
69
70  queryDataDrillAVGValue: function (field,data_type,
71  data_address){
72          var queryCopy = "select avg(cast("+field+
73          " as "+data_type+")) from dfs.`"+
74          data_address+"/*` limit 1";
75          try {
76                  var result = HTTP.call("POST",
77                  "http://localhost:8047/query.json",
78                  {
```

```
                data: {
                        "queryType" : "SQL",
                        "query" : queryCopy
                }
        });
                return result;

        } catch (e) {
                console.log(e);
                return e;
        }
},
```

- **Eliminar Campo:**

Eliminar Campo

```
removeField: function (new_version_address,
old_version_address,fields){
        var queryCopy = "create table dfs.root.`"+
        new_version_address+"` as select "+fields+
        " from dfs.`"+old_version_address+"`";
        try {
                var result = HTTP.call("POST",
                "http://localhost:8047/query.json",
                {
                data: {
                        "queryType" : "SQL",
                        "query" : queryCopy
                }
                });

                return result;

        } catch (e) {
                console.log(e);
                return e;
```

```
        }
},
```

- **Eliminar Registros:**

Eliminar Registros

```
removeRows: function (new_version_address,
old_version_address,rows_to_remove){
        var queryCopy = "create table dfs.root.`"+
        new_version_address+"` as
        select * from dfs.`"+old_version_address+
        "` where row_num not in
        ("+rows_to_remove+")";

        try {
                var result = HTTP.call("POST",
                "http://localhost:8047/query.json",
                {
                data: {
                        "queryType" : "SQL",
                        "query" : queryCopy
                }
                });

                return result;

        } catch (e) {
                console.log(e);
                return e;
        }
},
```

6.9.4. Pruebas

Las pruebas realizadas con la finalidad de comprobar el correcto funcionamiento de todas las funciones de preparación del conjunto de datos. A continuación se detallan las mismas en siguiente cuadro:

ID	Descripción del caso de prueba	Resultado Esperado	Resultado Obtenido
1	Se creó un proyecto con un campo 'Age' que representa la edad de un individuo, donde el mismo va ser ordenado de mayor a menor.	Se espera que coloque las edades mayores de primero	Los individuos de mayor edad son presentados de primeros (Ver figura 6.32).
2	Se realiza un filtrado en el campo 'Survived' de tipo entero con valores 0 o 1 colocando como filtro el valor 1.	Se obtienen todos registros que tienen como valor en campo 'Survived' = 1.	Se muestran en el proyecto solo los registros con 'Survived' = 1 (ver figura 6.33).
3	Se realiza un reemplazo de los datos del campo 'Pclass' con valor 3 por el valor 5.	Todos los registros con valor 3 en el campo 'Pclass' son cambiados al valor 5.	A todos los registros se les aplicó el cambio de valor (Ver figura 6.34).
4	Se realiza la exploración del campo 'Pclass' para conocer la cantidad de repeticiones que aparece de cada uno de los valores.	Una tabla con los valores y la cantidad de repeticiones en el conjunto.	Tabla con valores y cantidad de repeticiones en el conjunto (Ver figura 6.35).

Cuadro 6.16: Casos de Prueba de funciones de preparación sobre los campos del conjunto de datos

Figura 6.36: Ordenamiento de mayor a menor del campo 'Age'.

	PassengerId	Survived	Pclass	Name
	2	1	1	Cumings, Mrs. John Br
	3	1	3	Heikkinen, Miss. Laina
	4	1	1	Futrelle, Mrs. Jacques
	9	1	3	Johnson, Mrs. Oscar W
	10	1	2	Nasser, Mrs. Nicholas (
	11	1	3	Sandstrom, Miss. Marg
	12	1	1	Bonnell, Miss. Elizabeth
	16	1	2	Hewlett, Mrs. (Mary D K
	18	1	2	Williams, Mr. Charles E
	20	1	3	Masselmani, Mrs. Fatim
	22	1	2	Beesley, Mr. Lawrence
	23	1	3	McGowan, Miss. Anna
	24	1	1	Sloper, Mr. William Tho
	26	1	3	Asplund, Mrs. Carl Osc
	29	1	3	O'Dwyer, Miss. Ellen "N

Figura 6.37: Filtro 'Survived' = 1.

Figura 6.38: Reemplazo de campo 'Pclass' de valor 3 a valor 5.

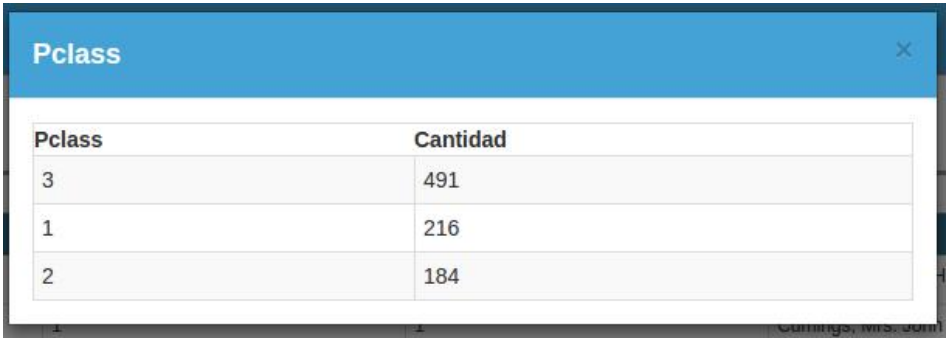

Figura 6.39: Exploración realizada sobre el campo 'Pclass'.

6.10. Iteración 9: Acciones de preparación sobre los datos

En esta iteración se establece que las acciones realizadas sobre los datos deben ser listadas y las mismas pueden ser deshechas por el usuario.

6.10.1. Planificación

En el siguiente cuadro se muestran las historias de usuario desarrollada en esta iteración.

ID	Fecha	Descripción
26	17/03/2017	Se debe mostrar una lista de las acciones de preparación de datos realizadas sobre el conjunto.
27	17/02/2017	El usuario puede deshacer estas acciones por separado.

Cuadro 6.17: Historias de usuario. Iteración 9.

6.10.2. Diseño

Para listar cada una de las acciones se realizó un cuadro dentro de la interfaz de preparación, el mismo contiene cada una de las acciones realizadas sobre el conjunto de datos, además se añadió un botón para deshacer una a una las acciones, a continuación se muestra el prototipo realizado:

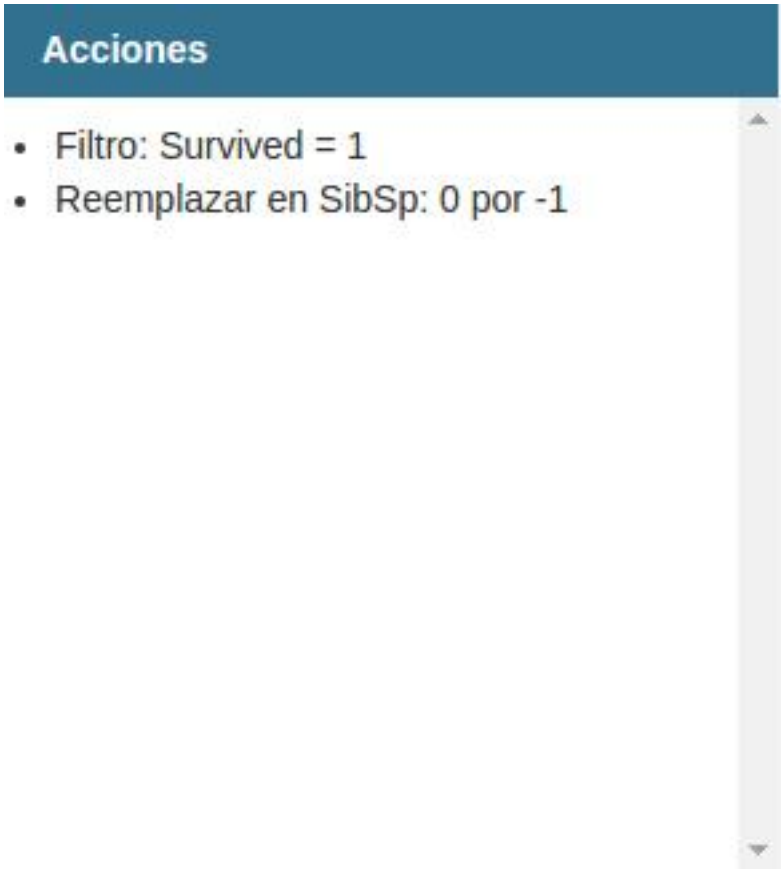

Figura 6.40: Prototipo de lista de acciones.

6.10.3. Codificación

A continuación se mostrará el código que permite deshacer las acciones realizadas sobre el conjunto de datos:

Deshacer acciones

```
'click .undo-btn'(event, template) {
    event.preventDefault();
    var project_id = FlowRouter.getParam('id');
    var versions = Projects.findOne({_id:project_id}).
    prepair_versions;
    var actions = Projects.findOne({_id:project_id}).
    actions;

    if (versions.length > 1 && actions.length > 0){
      var version_to_remove = versions[0];
      var current_version = versions[1];
      versions.shift();
      actions.pop();
      Meteor.call('removeHdfsFolder',version_to_remove,
      function(err,res){
        if(res.statusCode == 200){
            var old_version = Projects.update(
            {_id:project_id},
            {$set:{current_version_address:
            current_version,prepair_versions:
            versions,actions:actions}});

            if (old_version){
              Meteor.call('queryDataDrill',
              current_version, function(err,res){
                if(res.statusCode == 200){
                  Session.set('data_project',
                  res.data.rows);
                  Session.set('data_keys',
                  res.data.columns);
```

```
31                    Session.set('num_rows',
32                    res.data.rows.length);
33                    Session.set('num_fields',
34                    res.data.columns.length);
35                    Session.set('project_actions',
36                    actions);
37                    $(".backdrop").css('display','none');
38
39                  }
40                  if(err){
41                    alert("no data");
42                  }
43                });
44              }else{
45                $(".backdrop").css('display','none');
46
47                alert('No se ha podido revertir ninguna
48                acción!!!');
49              }
50          }else{
51            $(".backdrop").css('display','none');
52
53            alert('No se ha podido revertir ninguna
54            acción!!!');
55          }
56          if(err){
57            $(".backdrop").css('display','none');
58
59            alert('No se ha podido revertir ninguna
60            acción!!!');
61          }
62        });
63      }else{
64        $(".backdrop").css('display','none');
65
66        alert('No existe una versión previa para
```

```
67          revertir!!!!');
68        }
69      },
70
71  removeHdfsFolder: function (hdfs_address) {
72          check(hdfs_address, String);
73          try {
74                  var result = HTTP.call("DELETE",
75                  "http://"+hadoop_host+"/webhdfs/v1"+
76                  hdfs_address+"?op=DELETE&
77                  recursive=true&user.name="+
78                  cluster_user);
79                  return result;
80          } catch (e) {
81                  console.log(e);
82                  return e;
83          }
84  },
```

6.10.4. Pruebas

Las pruebas realizadas con la finalidad de comprobar el correcto funcionamiento de todas las funciones de deshacer las acciones de preparación sobre el conjunto de datos. A continuación se detallan las mismas en siguiente cuadro:

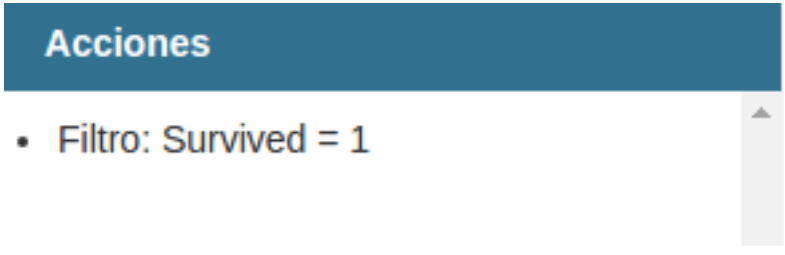

Figura 6.41: Prueba sobre acciones del conjunto de datos.

ID	Descripción del caso de prueba	Resultado Esperado	Resultado Obtenido
1	Se realizaron dos acciones sobre el conjunto de datos, la primera se filtró 'Survived' = 1 y la segunda se realizó un reemplazo de valores en 'SibSp' de 0 a -1, luego de esto se deshizo la última de las acciones realizadas.	Solo queda una acción en la lista que es la primera que se mencionó en la descripción.	Queda solo una acción en la lista de acciones: el filtrado de 'Survived' = 1 (Ver figura 6.).

Cuadro 6.18: Casos de Prueba de acciones de preparación realizadas sobre el conjunto de datos

Capítulo 7

Pruebas

En este capítulo se presentan las pruebas realizadas y los resultados obtenidos con el fin de evaluar el desempeño de la aplicación y su correcto funcionamiento.

7.1. Diseño de la prueba

Las pruebas fueron realizadas en tres máquinas cuyas especificaciones se encuentran en la tabla 7.1, donde a cada una se le instaló y configuró Apache Hadoop, Apache Drill y Meteor.

Especificaciones
Sistema Operativo: Ubuntu 16.04 64 bits Memoria: 4 GB Procesador: Intel Core i5-3470 CPU @ 3.20GHz Disco Duro: 500 GB
Sistema Operativo: Ubuntu 16.04 64 bits Memoria: 4 GB Procesador: Intel Core i3 CPU M 330 @ 2.13GHz Disco Duro: 320 GB
Sistema Operativo: Fedora 24 64 bits Memoria: 12 GB Procesador: Intel Core i7-3630QM @ 2.4ghzGHz Disco Duro: 720 GB

Cuadro 7.1: Especificaciones de equipos de prueba

7.2. Pruebas

Las pruebas realizadas se basan en evaluar el correcto funcionamiento de la aplicación al seguir el flujo de trabajo esperado.

Los pasos que se siguieron en cada prueba fueron:

1. Cargar los conjuntos de datos: Iris.csv, titanic.csv y menu.csv. Donde todos poseen un número distinto de registros y campos.

2. Revisar que se hayan creado de forma correcta en HDFS haciendo uso de la interfaz web de Apache Hadoop.

3. Crear un proyecto para cada uno de los conjuntos de datos subidos durante la prueba.

4. Realizar para cada proyecto algunas acciones de preparación de datos.

5. Generar una vista minable para cada proyecto y revisar que se haya generado en HDFS, junto con las diferentes versiones a lo largo del proceso de preparación haciendo uso de la interfaz web de Hadoop.

7.3. Resultados y conclusiones de las pruebas

En lineas generales se quiso evaluar el funcionamiento de la creación de conjuntos de datos, la creación de un proyecto y la aplicación de diferentes técnicas de preparación de datos y por último la generación de una vista minable que sirve de entrada para los módulos siguientes.

1. Creación de conjuntos de datos

   Para la creación de los conjuntos de datos se hizo uso de la aplicación desde el navegador, se repitió el proceso para cada uno de los conjuntos de datos definiendo una nombre y una descripción del conjunto que se estaba subiendo. En la figura 7.1 se puede ver un ejemplo de este proceso.

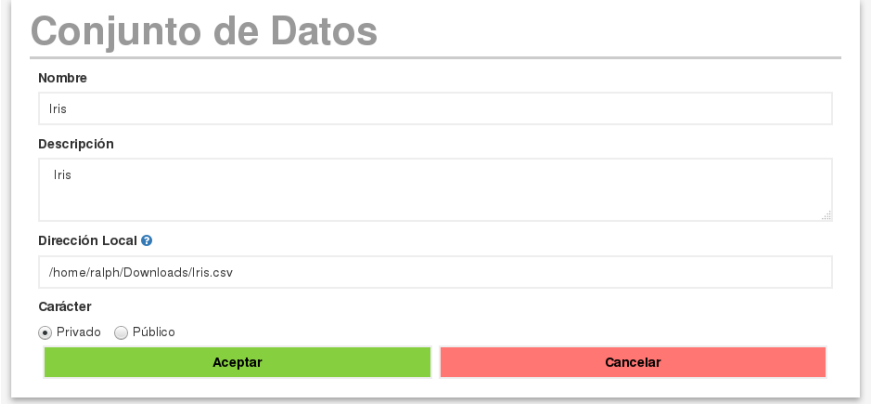

Figura 7.1: Paso 1 de pruebas: crear conjunto de datos

2. Comprobar creación de conjuntos de datos
   Para la comprobación de la creación de los conjuntos se hizo uso de

la interfaz web que provee Hadoop, la cual tiene una función para explorar el sistemas de archivos distribuidos de Hadoop desde esta. En la ruta '/user/hadoop/datasets/' podemos observar en la figura 7.2 la correcta creación de los conjuntos de datos.

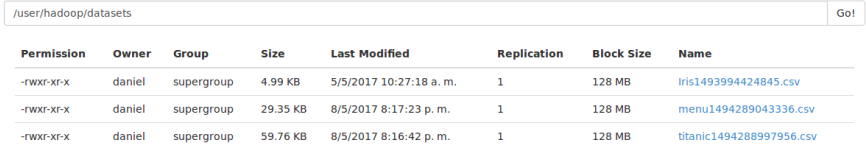

Figura 7.2: Paso 2 de pruebas: revisión de la creación de conjuntos de datos: interfaz web de Hadoop

3. Crear proyecto
   Para la creación de los proyectos se hizo uso de la aplicación desde el navegador, se repitió el proceso para cada uno de los proyectos definiendo una nombre y una descripción para el proyecto y escogiendo el conjunto de datos que utilizaría. En la figura 7.3 se puede ver un ejemplo de este proceso.

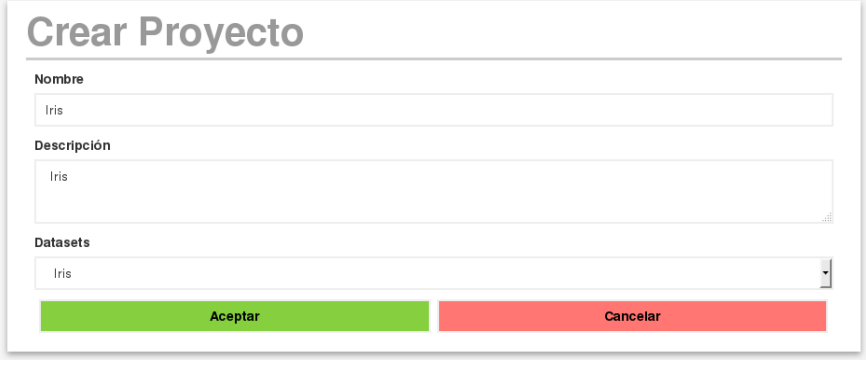

Figura 7.3: Paso 3 de pruebas: crear proyecto

4. Aplicación de acciones de preparación de datos
   Una vez creado los proyectos se aplicaron diferentes acciones a cada conjunto de datos dependiendo de la forma de los datos con la finalidad de llevarlos a la forma ideal para su uso durante las siguientes

etapas. En la figura 7.4 se puede observar un ejemplo de la forma del conjunto de datos luego de aplicar distintas acciones.

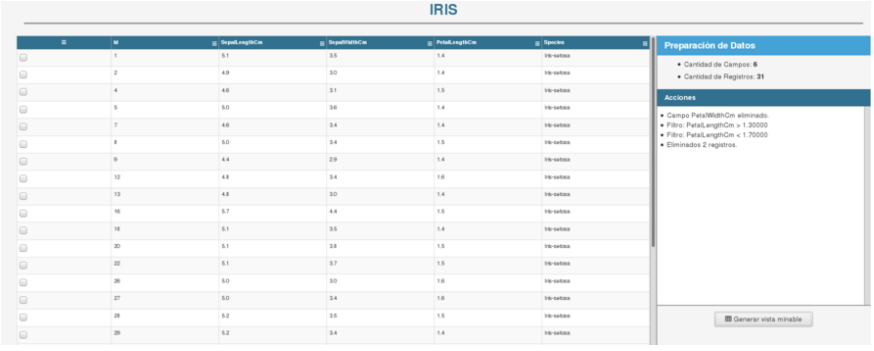

Figura 7.4: Paso 4 de pruebas: realizar múltiples acciones durante la etapa de preparación

5. Generar vista minable y comprobar su creación
   Por último, la generación de la vista minable se realizó haciendo uso de la aplicación utilizando el botón 'Generar vista minable' en la pantalla de preparación de datos. Y como se puede observar en la figura 7.5 dentro de HDFS se generaron las tablas intermedias para cada una de las versiones, una por acción que amerite un cambio en HDFS, y una para la vista minable.

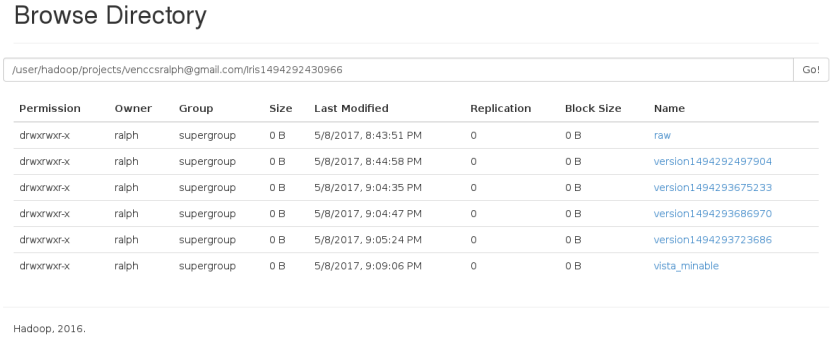

Figura 7.5: Paso 5 de pruebas: generar vista minable

Podemos concluir que la aplicación durante el flujo de datos que abarca de las etapas del proceso KDD, no presento ningún inconveniente mayor durante su funcionamiento. La creación de conjuntos de datos y proyectos, así como la aplicación de algunas acciones para la preparación de los datos y la generación de la vista minable se llevo a cabo de forma exitosa considerando los resultados esperados, como se pudo observar en las figuras anteriores.

Capítulo 8

Conclusiones

En un principio se partió con la idea de que el clúster Hadoop formaría parte del modelo de datos, idea que fue lograda al realizar correctamente la instalación de Apache Hadoop en modo pseudo-distribuido, seguidamente se logró instalar la herramienta Apache Drill que permitió realizar la extracción de los datos desde el clúster hasta la aplicación web de manera correcta y haciendo pleno uso de las facultades del sistema distribuido.

Se logró desarrollar una primera versión de la aplicación web que realiza la manipulación de los conjuntos datos alojados en el clúster Hadoop para la preparación de una vista minable siguiendo las etapas de selección, preprocesamiento y transformación de los datos enmarcados dentro del proceso KDD. Para dicha preparación se logró la implementación de las siguientes funciones: eliminar campos y registros, filtrar un campo por comparación, explorar la metadata de los campos y reemplazar valores por imputación. Todas estas permiten la preparación de los datos para la generación de la vista minable.

8.1. Contribución

Este trabajo especial de grado contribuye principalmente con la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de

Venezuela, para la formación de los estudiantes en el área de ciencia de datos y minería de datos.

8.2. Recomendaciones

Para la ejecución de esta aplicación debe estar configurado el clúster Hadoop y la herramienta Apache Drill ambos en el servidor, donde estas deben tener la configuración realizada en el capítulo de desarrollo. La aplicación puede ejecutarse independientemente si el clúster es de un solo nodo o multinodo y variará su desempeño en cuanto a velocidad dependiendo de la capacidad de procesamiento y memoria RAM del clúster Hadoop.

8.3. Trabajos Futuros

Sobre el trabajo realizado queda abierta la ventana para el desarrollo de los siguientes pasos del proceso KDD, como la minería de datos y la visualización de los mismos, además de esto a partir de esta primera versión de la aplicación pudiera realizarse el trabajo con tipo de dato fecha y una pre-visualización de los datos para la exploración de estos en la etapa de preparación, así como la expansión de las acciones que se realizan en la etapa de preparación.

Apéndice A

Manual de Usuario

BIG DATA KDD

Dominio: http://localhost:3000

1. **Acceso**
   Correo: correo electrónico con el cual creó la cuenta.
   Contraseña: contraseña escogida al momento de creación de la cuenta.

Figura A.1: Formulario de acceso

1.1. **Registrarse**

Nombre: el nombre del usuario.
Apellido: el apellido del usuario.
Correo: correo electrónico con el cual identificará la cuenta.
Contraseña: contraseña de acceso para esta cuenta, longitud mínima de 6 caracteres.

Figura A.2: Formulario de acceso

1.2. **Olvidé mi contraseña**

Correo: Debe ingresar el correo electrónico con el cual creó la cuenta.

Figura A.3: Formulario de recuperar contraseña

Una vez ingresado el correo se le enviara un correo con el enlace para restablecer su contraseña.

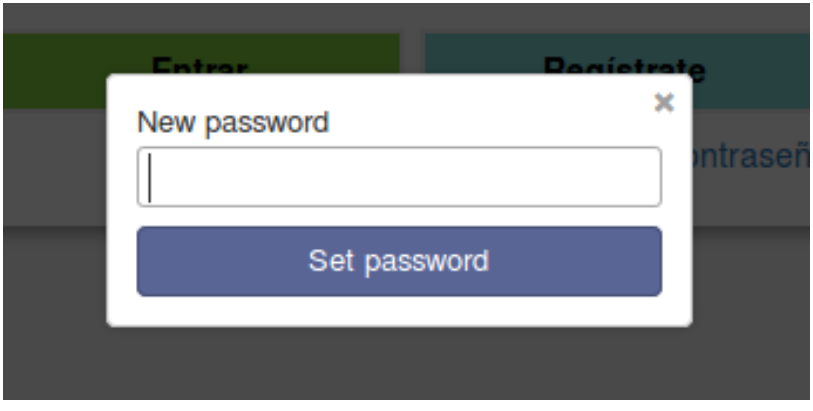

Figura A.4: Formulario de recuperar contraseña

2. Conjuntos de datos

2.1. Listado

Iconos:

- Ojo: ver detalle.
- Lápiz: editar conjunto de datos.
- Papelera: eliminar permanentemente conjunto de datos.

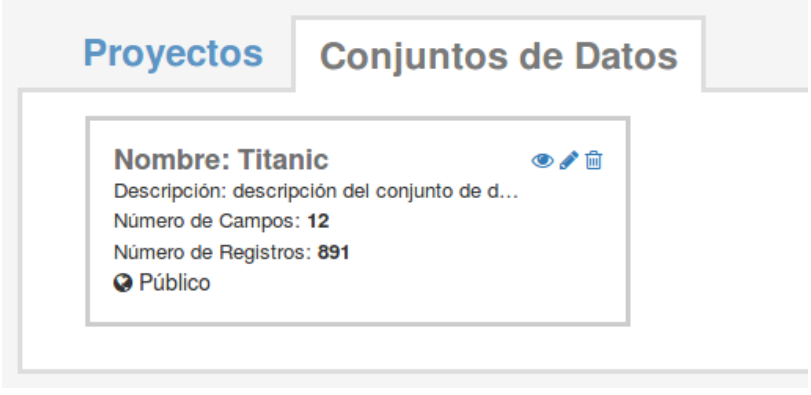

Figura A.5: Listado de conjuntos de datos

2.2. **Detalle**

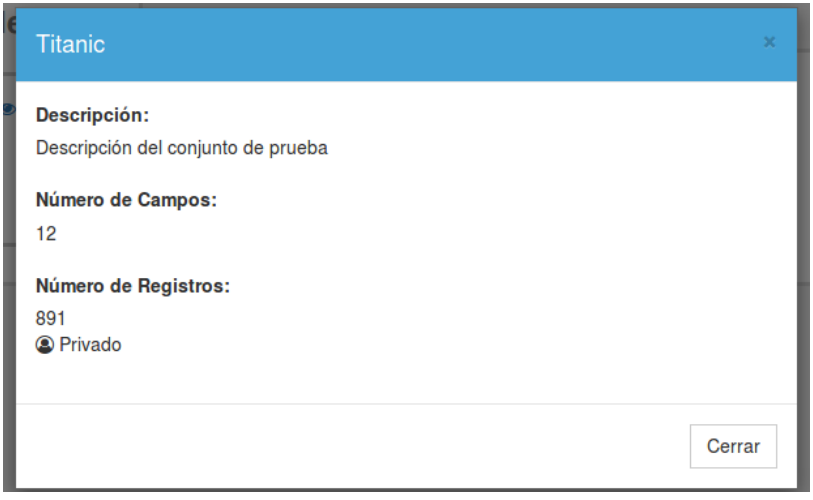

Figura A.6: Vista detallada de un conjunto de datos

2.3. **Añadir**

Nombre: nombre con el que desea identificar el conjunto de datos.
Descripción: una breve descripción acerca del conjunto de datos.
Dirección local: la dirección dentro del sistema de archivos Unix, escrito desde la raíz (/).
Caracter: debe ser el conjunto identificado como público o privado.

Figura A.7: Formulario para añadir conjunto de datos

2.4. **Modificar**

Nombre: nombre con el que desea identificar el conjunto de datos.
Descripción: una breve descripción acerca del conjunto de datos.
Dirección local: la dirección dentro del sistema de archivos Unix, escrito desde la raíz (/) (No puede ser modificada).
Caracter: debe ser el conjunto identificado como público o pri-

vado.

Figura A.8: Formulario para editar conjunto de datos

2.5. **Eliminar**

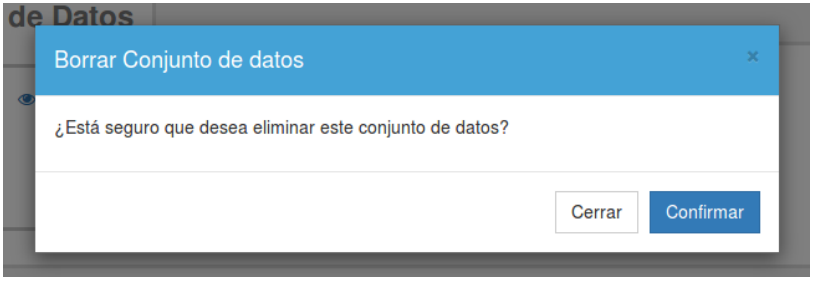

Figura A.9: Confirmación para eliminar conjunto de datos

3. **Proyectos**

3.1. **Listado**

Iconos:

- Ojo: ver detalle.
- Lápiz: editar proyecto.
- Papelera: eliminar permanentemente el proyecto.

Figura A.10: Listado de proyectos

3.2. Detalle

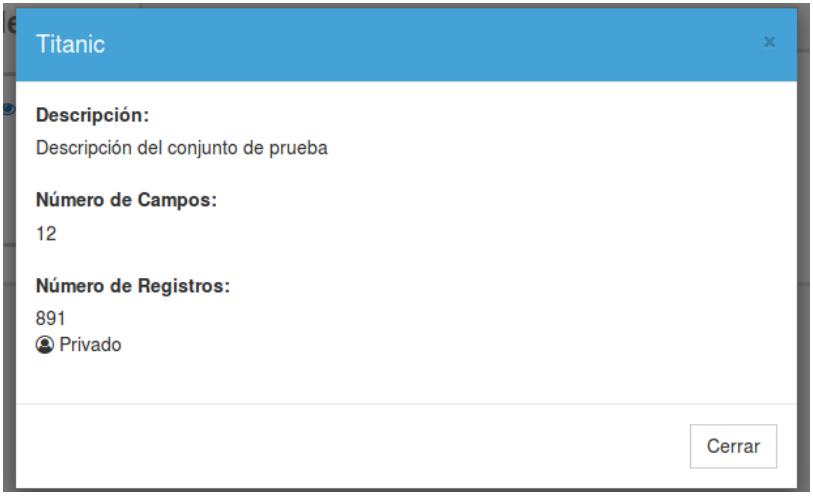

Figura A.11: Vista detallada de un proyecto

3.3. Crear

Nombre: nombre con el que desea identificar el conjunto de datos.
Descripción: una breve descripción acerca del conjunto de datos.
Conjunto de datos: conjunto de datos que se han añadido a la aplicación.

Figura A.12: Formulario para crear un proyecto

3.4. **Modificar**

Nombre: nombre con el que desea identificar el proyecto.
Descripción: una breve descripción acerca del proyecto.
Conjunto de datos: conjunto de datos que utiliza el proyecto (no puede ser modificado).

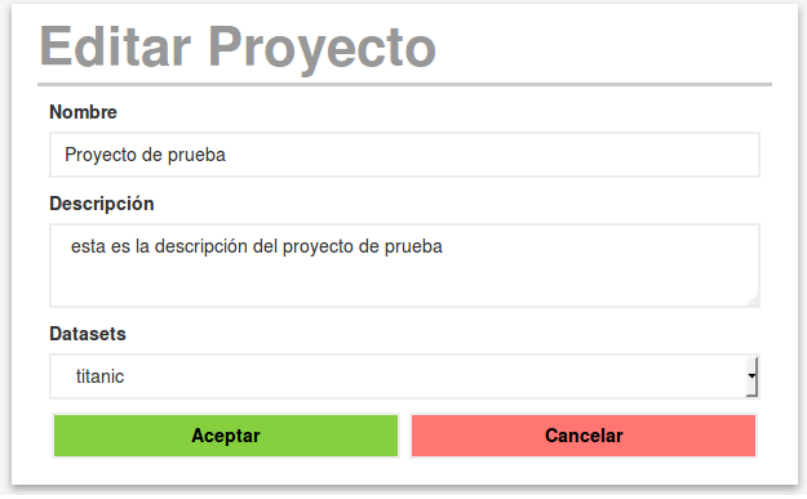

Figura A.13: Formulario para editar proyecto

3.5. **Eliminar**

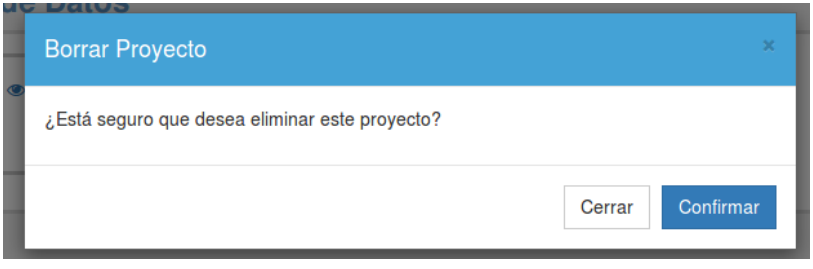

Figura A.14: Confirmación para eliminar proyecto

4. **Preparación de datos**
   En la parte superior se verá el nombre del proyecto en el centro.
   En el borde izquierdo veremos las opciones para avanzar a las siguientes etapas del proceso KDD, así como la opción para deshacer.
   En el extremo superior derecho se verá información relacionada al

conjunto de datos.
En el extremo inferior derecho veremos información con respecto a las acciones realizadas en esta instancia.

Figura A.15: Interfaz para la preparación de datos

4.1. **Tabla de datos**

En la tabla se observarán los datos para cada registro por cada columna.

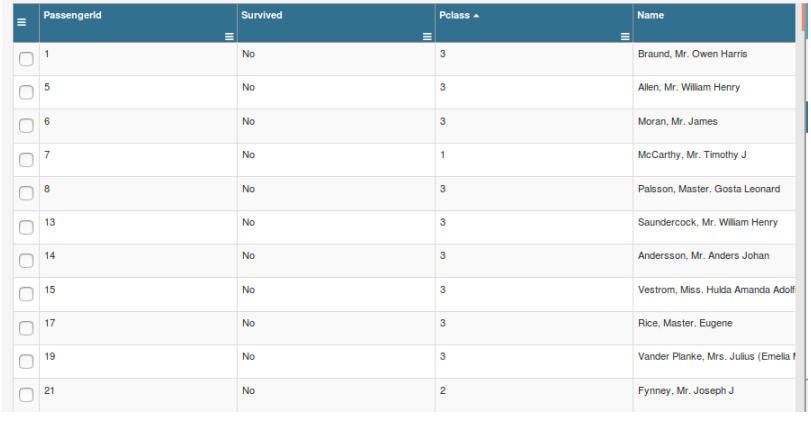

Figura A.16: Sección de la tabla de datos en la interfaz de preparación

4.2. **Opciones para una columna**
Al seleccionar las opciones para una columna se expande el listado de las acciones disponibles para esa columna.

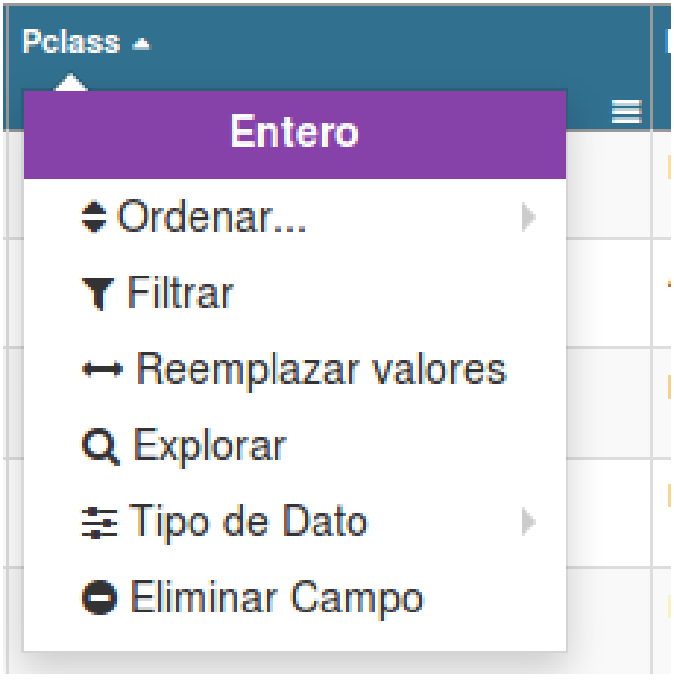

Figura A.17: Opciones aplicables a una columna

4.3. **Acciones**
Al realizar una acción que involucre la creación de una tabla intermedia se listarán en esta sección.

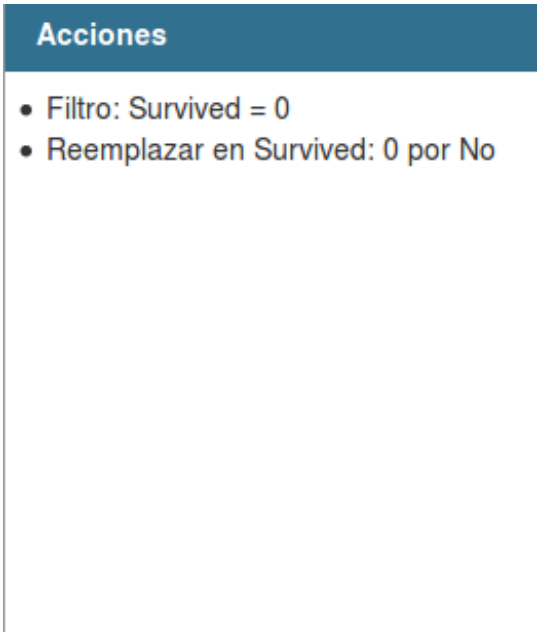

Figura A.18: Sección de acciones realizadas

Para deshacer la última acción realizada debe seleccionar el icono en la esquina inferior izquierda.

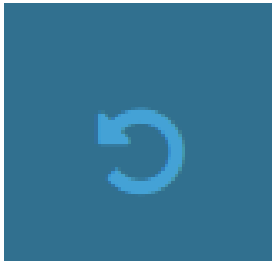

Figura A.19: Botón de deshacer

4.3.1. Ordenar

En la figura B.20 se observan las opciones para ordenar: `Menor a Mayor, Mayor a Menor`.

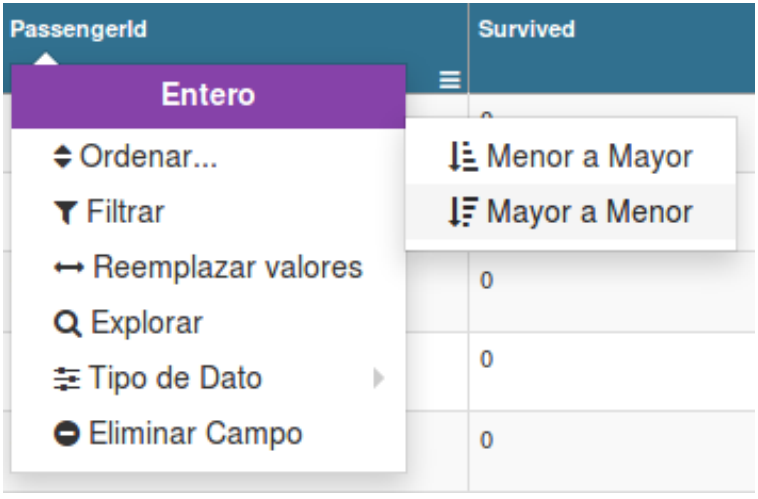

Figura A.20: Opción de ordenar los registros por una columna

4.3.2. Filtrar

Se pueden filtrar los valores que sean: `Igual a, Mayor que o Menor que`.

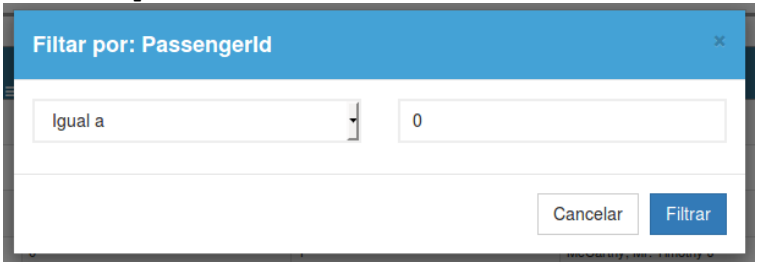

Figura A.21: Opción de filtrar

4.3.3. Reemplazar

Encuentra los valores que sean iguales al `Valor Inicial` y los reemplaza por el valor en `Reemplazar con`.

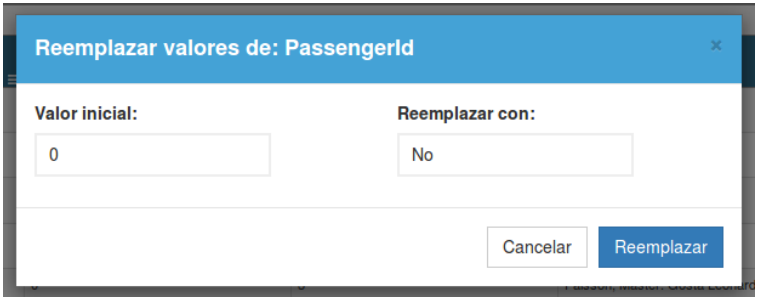

Figura A.22: Opción para reemplazar los valores de una columna

4.3.4. Explorar

Información sobre una columna.

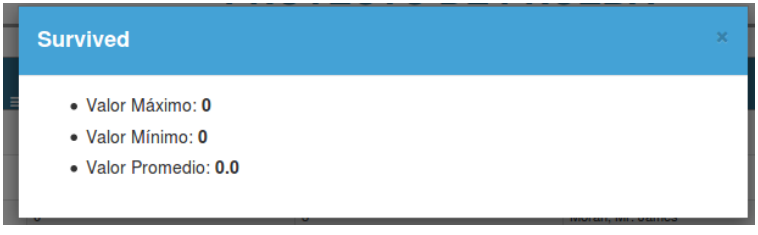

Figura A.23: Opción para detallar información sobre una columna

4.3.5. Tipo de Dato

En esta opción se puede modificar de forma manual el tipo de datos escogido al momento de la creación del conjunto de datos. Se puede cambiar a: `Entero, Decimal o String`

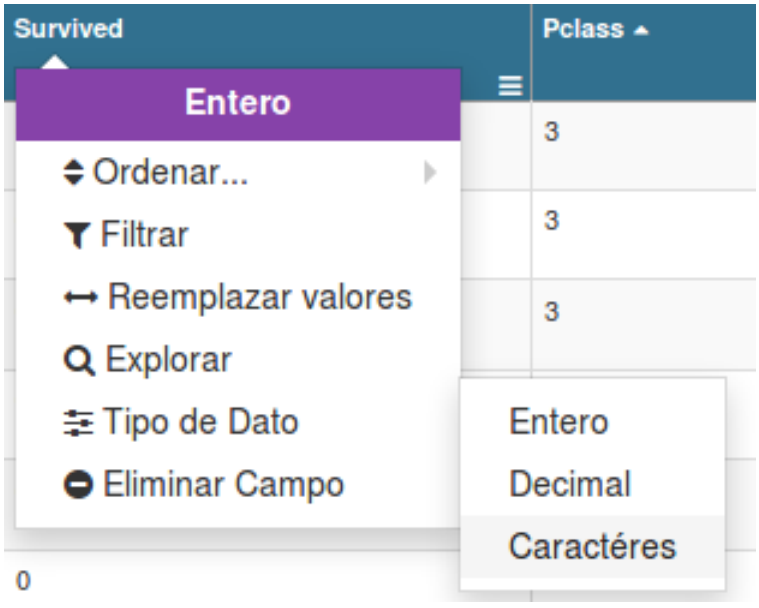

Figura A.24: Opción de cambiar tipo de dato

4.3.6. **Eliminar Campo**

Al escoger la opción se elimina el campo del conjunto de datos.

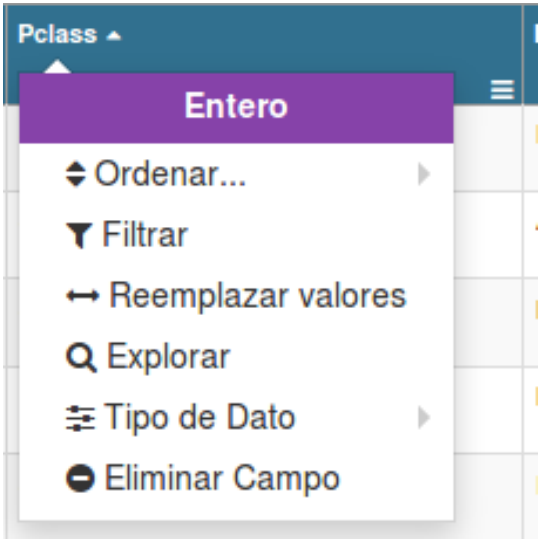

Figura A.25: Opción para eliminar una columna del conjunto de datos

4.4. **Generar Vista Minable**
Este botón genera la vista minable que es el producto final del modulo de preparación de datos.

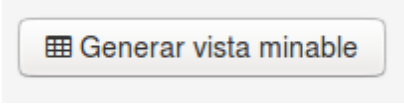

Figura A.26: Botón para generar vista minable

Bibliografía

[1] Las cinco v's de big data. https://cdn.app.compendium.com/, mar 2017.

[2] Etapas del proceso kdd. http://1.bp.blogspot.com, mar 2017.

[3] Arquitectura de hdfs. https://elentornodehadoop.files.wordpress.com/, mar 2017.

[4] Arquitectura de nodejs. http://www.nodehispano.com, mar 2017.

[5] Interacción de nodejs. https://i-msdn.sec.s-msft.com, mar 2017.

[6] Modo no bloqueante de nodejs. https://image.slidesharecdn.com, mar 2017.

[7] Definición de datos. http://definicion.de/datos/, mar 2017.

[8] Definición de información. http://www.definicionabc.com/tecnologia/informacion.php, mar 2017.

[9] Definición de conocimiento. http://conceptodefinicion.de/conocimiento/, mar 2017.

[10] M Rouse. Definición de ciencia de datos. http://conceptodefinicion.de/conocimiento/, mar 2017.

[11] Rubén Casado. Definición de big data. http://www.dataprix.com/blog-it/data-science/big-data-volumen-velocidad-variedad, mar 2017.

[12] Bernard Marr. Why only one of the 5 vs of big data really matters. http://www.ibmbigdatahub.com/blog/why-only-one-5-vs-big-data-really-matters, mar 2017.

[13] Juan Vidal. Big data: Gestión de datos no estructurados. http://www.dataprix.com/blog-it/big-data/big-data-gestion-datos-no-estructurados, mar 2017.

[14] Modelo relacional. http://ict.udlap.mx/people/carlos/is341/bases03.html, mar 2017.

[15] Leonardo De Seta. Acid en las bases de datos. https://dosideas.com/noticias/base-de-datos/973-acid-en-las-bases-de-datos, mar 2017.

[16] Adrián Garcete. Bases de datos orientadas a columnas. http://jeuazarru.com/wp-content/uploads/2014/10/dbco.pdf, mar 2017.

[17] José Alberto. Bases de datos clave/valor. http://softinspain.com/desarrollo/bases-de-datos-clave-valor/, mar 2017.

[18] Anthony Sotolongo. Bases de datos orientadas a documentos. https://es.slideshare.net/asotolongo/bases-de-datos-nosql-orientadas-a-documentos, mar 2017.

[19] Juan Cía. Neo4j: qué es y para qué sirve una base de datos orientada a grafos. https://bbvaopen4u.com/es/actualidad/neo4j-que-es-y-para-que-sirve-una-base-de-datos-orientada-grafos, mar 2017.

[20] Francisco Ibarra. Descubrimiento del conocimiento (kdd) : "el proceso de minería". http://mineriadatos1.blogspot.com/2013/06/descubrimiento-del-conocimiento-kdd-el.html, mar 2017.

[21] Julian Pérez. Lenguaje de programación. http://definicion.de/lenguaje-de-programacion/, mar 2017.

[22] Mozilla Developer Network. Html. https://developer.mozilla.org/en-US/docs/Web/HTML, mar 2017.

[23] W3C. Css. http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo, mar 2017.

[24] Damian Pérez. Javascript. http://www.maestrosdelweb.com/que-es-javascript, mar 2017.

[25] Javier Gutierrez. Frameworks web. http://www.lsi.us.es/, mar 2017.

[26] Hadoop. http://hadoop.apache.org/, mar 2017.

[27] Apache drill. https://drill.apache.org/, mar 2017.

[28] Nodejs. https://nodejs.org/es/, mar 2017.

[29] Meteor. https://www.meteor.com/, mar 2017.

[30] Mongodb. https://www.mongodb.com/es, mar 2017.

[31] extreme programming. http://www.extremeprogramming.org/, mar 2017.