



Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Computación Paralela y Distribuida (CCPD)

Desarrollo de una herramienta para la integración de repositorios digitales institucionales con plataformas de grandes volúmenes de datos (Big Data)

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
por los Bachilleres
Ysidro Alba C.I. 20613436
Rafael Piña C.I. 22760076
para optar al título de
Licenciado en Computación

Prof. Jesús Lares y Prof. José Sosa

Caracas, 24 / 10 / 2016

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACION

ACTA DEL VEREDICTO

Quienes suscriben, miembros del jurado designado por el Consejo de la Escuela de Computación para examinar el Trabajo Especial de Grado, presentado por los Bachilleres Rafael Alejandro Piña Yanza C.I: 22.760.076 e Ysidro Moisés Alba Maloney C.I: 20.613.436, con el título "Desarrollo de una herramienta para la integración de repositorios digitales institucionales con plataformas de grandes volúmenes de datos (Big Data)", a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los miembros del jurado, se fijó el día 24 de Octubre de 2016, a las 9:00 am, para que sus autores lo defendieran de forma pública, en el aula PAIII, lo cual estos realizaron mediante una exposición oral de su contenido, y luego respondieron satisfactoriamente a las preguntas que le 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.

En fe de lo cual se levanta la presente acta, en Caracas el 24 de Octubre de 2016, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Tutor Jesús Lares.



Prof. Jesús Lares
Tutor



Prof. José Sosa
Co-tutor



Prof. Hector Navarro
Jurado



Prof. Carlos Acosta
Jurado

DEDICATORIA

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

A nuestros padres (Rafael e Isidro), madres (Mariluz y Zulay), gracias por estar ahí en todo momento para que sigamos adelante.

A nuestros tutores Jesús Lares y José R. Sosa, por el apoyo y colaboración en esta investigación.

Resumen

Los repositorios digitales son una herramienta útil para la gestión de documentos digitales. En el caso de los repositorios digitales institucionales, se apoyan en la creación de objetos digitales para hacer referencia a archivos de cualquier tipo. En este trabajo se estudia la integración de herramientas basadas en big data y repositorios digitales para manejar grandes volúmenes de documentos con facilidad. Logramos esto aprovechando las capacidades de Hadoop para la computación distribuida sobre el sistema de almacenamiento de archivos HDFS gestionado por un repositorio digital Fedora Commons, el cual resulta útil en la prestación de nuevos tipos de servicios de objetos digitales y el mantenimiento de cantidades cada vez mayores de datos.

Palabras clave: Repositorio digital, metadatos, Hadoop, firma electrónica.

Índice general

Introducción	1
1. El problema	3
1.1. Planteamiento del problema	3
1.2. Justificación	4
1.3. Objetivos	4
1.3.1. General	4
1.3.2. Específicos	4
1.4. Alcance	5
2. Marco teórico	7
2.1. Ciencias de datos	7
2.1.1. Grandes volúmenes de información	7
2.1.2. Organización de los datos	9
2.1.2.1. Estructurados	9
2.1.2.2. Semi-Estructurados	9
2.1.2.3. No Estructurados	9
2.2. Apache Hadoop	10
2.2.1. Cloudera	10
2.2.2. Hadoop Distributed File System	11
2.2.2.1. Arquitectura	13
2.2.2.2. Escalabilidad	14
2.2.3. MapReduce	15
2.2.3.1. Conceptos Básicos de MapReduce	16
2.3. Sistema Operativo	19
2.4. Objeto digital	19

2.5.	Repositorio digital	20
2.5.1.	Fedora Commons	20
2.5.1.1.	Modelo Objeto Digital	21
2.5.1.2.	Datastreams	22
2.5.1.3.	FOXML	23
2.5.1.4.	Comunicación con el usuario	24
2.6.	Metadatos	25
2.6.1.	Estándares de metadatos	26
2.6.2.	Dublin Core	29
2.6.3.	OAI-PMH	31
2.6.4.	Open Harvester System	32
2.7.	CMS	33
2.7.1.	Drupal	33
2.7.1.1.	Islandora	34
2.8.	Lenguajes de Programación	34
2.8.1.	PHP	35
2.8.1.1.	Smarty	36
2.8.2.	Java	36
2.9.	Bases de datos	37
2.9.1.	Bases de datos relacionales	37
2.9.1.1.	MySQL	38
2.10.	Akubra	38
2.10.1.	Akubra Low Level Storage	39
2.11.	Amazon Web Services	39
2.11.1.	Costos	40
2.12.	Putty	42
2.13.	Interoperabilidad	42
2.13.1.	Servicio Web	43
2.13.1.1.	XML	44
2.13.1.2.	SOAP	45
2.14.	Firma electrónica	46
2.14.1.	Xolidosing	48
2.14.2.	OpenSSL	49
2.14.3.	Certificado electrónico	49

3. Método de desarrollo	51
3.1. Ad Hoc	51
4. Desarrollo de la solución	55
4.1. Arquitectura de la solución	55
4.2. Análisis y diseño	57
4.3. Configuración del ambiente	59
4.3.1. Instalación del Clúster Hadoop	59
4.3.1.1. Configuración de nodos del cluster	60
4.3.1.2. Instalación Cloudera Hadoop	63
4.3.2. Instalación y Configuración del Repositorio Digital Fedora Commons	68
4.3.3. Configuración Akubra HDFS	70
4.3.4. Integración con CMS	73
4.3.5. Cambios en la interfaz OHS	75
4.3.5.1. Listado de Objetos Digitales	77
4.3.6. Firma electrónica	79
5. Conclusiones	85
5.1. Aporte	86
5.2. Recomendaciones	86
5.3. Trabajos futuros	87
A. Anexos	89
A.1. Adición de módulos	89
A.1.1. Drupal Filter	89
A.1.2. Tuque	91
A.1.3. Islandora Core Module	91
A.1.4. Islandora Basic Collection Solution Pack	93
A.2. Casos de uso	95
A.2.1. Islandora	95
A.2.1.1. Autenticar usuario	96
A.2.1.2. Cargar documento	96
A.2.1.3. Eliminar objeto	97
A.2.1.4. Cargar archivo comprimido	98
A.2.2. OHS	101

A.2.2.1. Agregar repositorio	101
A.2.2.2. Buscar Documentos	102
A.3. Generar archivos de prueba	105

Índice de figuras	107
--------------------------	------------

Introducción

La gestión documental es un problema recurrente en los procesos de administración tanto pública como privada. Las tendencias actuales nos llevan a desarrollar esta gestión a través de medios digitales, es decir, usando las tecnologías de información. En este contexto, aparecen lenguajes de representación de datos y frameworks para apoyar la especificación y administración de la información digital, tales como XML, Arquitecturas Orientadas a Servicios y Servicios Web.

Para abordar la tarea de analizar, diseñar e implementar una solución de gestión documental, se requiere primero comprender qué conceptos están involucrados. Entre los más importantes están: documento electrónico, metadatos, objetos digitales, repositorios digitales. Junto a cada uno de estos conceptos, es necesario conocer las tecnologías existentes que permiten implementar una solución.

Un punto importante de este trabajo es el almacenamiento de información, cabe destacar que hoy en día almacenamos más y más información, este almacenamiento lo llevamos a cabo cada vez de formas más eficientes, rápidas y con menores costos. La cantidad de datos hoy en día es tan grande, compleja y dinámica que las herramientas convencionales no sirven ciento por ciento para captar, administrar y almacenarlos. Por esta razón, las herramientas que usamos deben poder integrarse de forma sencilla a una plataforma basada en big data, el aspecto mas deseable es la escalabilidad del sistema.

En el capítulo 1 se presenta el problema y la necesidad de implementar una plataforma capaz de soportar grandes volúmenes de datos usando un repositorio digital encargado de administrar e indexar objetos digitales. En el segundo se hace un repaso de todos conceptos claves para la lectura de este trabajo, mas específicamente, el uso de métodos y herramientas que permitirán responder las dudas planteadas en el capitulo anterior. En el capítulo 3 explicamos en que consiste la metodología ad hoc y como la aplicamos en nuestro trabajo, luego, en el capítulo 4 describimos todo lo referente al desarrollo de nuestra solución, la justificación sobre la elección de algunas herramientas y finalmente presentamos las conclusiones del trabajo en el capítulo 5.

Capítulo 1

El problema

1.1. Planteamiento del problema

Un repositorio digital es una herramienta que permite organizar, archivar, preservar y difundir determinados objetos digitales (imagen, vídeo, audio, documentos multipágina, PDF, presentaciones, etc.) resultado de las actividades realizadas por instituciones públicas, permitiendo que los usuarios tengan acceso al material digital de forma organizada. Tener solamente registros físicos siempre ha sido un problema pues es incómodo realizar búsquedas o ir de una institución a otra para hacer solicitudes, estamos en una época donde muchos procesos están siendo automatizados y tenemos la oportunidad aportar a esta tendencia de preservación digital.

Los contenidos digitales provenientes de diversas fuentes están aumentando, el intercambio de contenidos, la publicación de esos contenidos en repositorios y la reutilización de la información se realiza todo el tiempo. Dicha situación descrita, plantea a la sociedad, empresas e instituciones un gran reto relacionado con la gestión de grandes conjuntos de datos que requiere tener como base una infraestructura que sea eficiente.

Hasta no hace mucho, previas infraestructuras se han encontrado con el llamado “problema del big data”, el cual solo puede ser enfrentado con nuevas plataformas de tecnología y paradigmas de programación diferentes a las tradicionales.

1.2. Justificación

El beneficio de este trabajo es tener una plataforma que haga gestión de muchos repositorios y almacenamiento de documentos en un sistema distribuido. La ventaja de esto es que los repositorios solamente tienen la función de conectarse a otros y en caso de necesitar algún documento, deben buscar el objeto digital que contiene la dirección donde está almacenado.

Otra ventaja es que el usuario no necesita tener conocimiento sobre el funcionamiento de la plataforma. Por medio de un formulario se podría solicitar todos los documentos que hagan referencia a una determinada cédula, una fecha, nombre, etc; el resultado es una lista de enlaces a los archivos solicitados. En el caso de subir archivos, por medio una interfaz, selecciona que va a guardar en el sistema y agrega los metadatos correspondientes.

Es un primer paso para la interoperabilidad entre las instituciones del país, poder agilizar trámites por medio de consultas rápidas y guardar toda la información requerida sin preocuparse por el espacio de almacenamiento. Esta plataforma permite que todos puedan observar la información de cualquier repositorio de manera fácil y cómoda.

1.3. Objetivos

1.3.1. General

Desarrollar una aplicación de repositorios digitales apoyado en una plataforma distribuida que permita la consulta de información sobre documentos almacenados en un conjunto de nodos que actúan como balanceadores en las búsquedas realizadas por el usuario.

1.3.2. Específicos

- Implementar una plataforma robusta para el almacenamiento de grandes volúmenes de datos.
- Garantizar la interoperabilidad del sistema definiendo el formato de los metadatos.

- Proponer una arquitectura que busque maximizar la eficiencia en el trabajo con documentos electrónicos.
- Integrar la firma electrónica a los documentos compartidos, garantizando así su autenticidad.

1.4. Alcance

- Instalar una plataforma basada en Fedora Commons sobre Hadoop para el procesamiento de documentos digitales.
- Configurar esta plataforma para que sea ejecutada sobre Amazon EC2.
- Desarrollar un interfaz para poder buscar y subir documentos de forma intuitiva.

Capítulo 2

Marco teórico

2.1. Ciencias de datos

Es la generación de conocimiento a partir de grandes volúmenes de datos que pueden ser estructurados o no estructurados, aplicando técnicas de procesamiento paralelo y distribuido, para implementar algoritmos que permitan predecir o detectar patrones sobre los datos almacenados. Este proceso para obtener conocimiento es de vital importancia pues sirve como base para crear herramientas o realizar análisis para la toma de decisiones, el nivel superior del negocio, por ejemplo.

2.1.1. Grandes volúmenes de información

Es un término mejor conocido como Big Data, hace referencia a una cantidad de datos tal que supera la capacidad del software habitual para ser capturados, administrados y procesados en un tiempo razonable. El volumen de los datos masivos crece constantemente. Cuando hablamos de grandes volúmenes nos referimos a tratamientos de Terabytes o Petabytes. Esto permite incluir en este tipo de proyectos informaciones (por ejemplo logs) que hasta la fecha no se utilizaban porque la tecnología no permitía procesarlos en un tiempo razonable. El concepto de volumen es muy variable y cada día que pasa aumenta la cantidad que podemos considerar grandes volúmenes de datos.

Dicho concepto engloba infraestructuras, tecnologías y servicios que han sido creados para dar solución al procesamiento de enormes conjuntos de datos estructurados, no estructurados o semi-estructurados (mensajes en redes sociales, señales de móvil, archivos de audio, sensores, imágenes digitales, datos de formularios, emails, datos de encuestas, logs, etc) que pueden provenir de sensores, micrófonos, cámaras, escáneres médicos, imágenes.

Al hablar del término Big Data, se tienen relacionados distintos fenómenos los cuales ayudan a explicar mejor el concepto de Big Data. Estos fenómenos están separados en 5 variables, conocidas como las 5 V de Big Data:

- **Volumen:** Cantidad de data generada por unidad de tiempo. Dependiendo de las capacidades de procesamiento o almacenamiento, grandes volúmenes pueden ser Terabytes, como para otros grandes volúmenes pueden ser Zettabytes. Para el procesamiento de estas cantidades no se deben utilizar bases de datos tradicionales debido a que su rendimiento es deficiente y no proveen técnicas para el particionamiento de estas.
- **Velocidad:** Velocidad a la cual es generada la data. Existen herramientas las cuales permiten el análisis de estos datos sin tener siquiera que almacenarlos.
- **Variedad:** Distintos tipos de dato que se pueden utilizar. Los datos aparte de tener diferentes tipos, también puede ser estructurados, semi-estructurados y no estructurados (para más información de la Organización de los datos observar el punto 2.1.2).
- **Veracidad:** Credibilidad y correctitud de los datos. Algunas veces, dependiendo de la fuente de los datos, la calidad y certeza de estos no pueden ser controlados y estos casos para el momento de su análisis podrían entregar valores erróneos. Normalmente los datos erróneos son creados gracias al poco control que se puede tener sobre un humano, es decir, la fuente de los datos provienen de los humanos.
- **Valor:** Valor oculto en los datos. Así se tengan grandes volúmenes de

datos, si estos no poseen valor alguno, no se podría obtener información valiosa de estos.

2.1.2. Organización de los datos

La organización de los datos se refiere a cómo organizar los datos usando un sistema manejador de bases de datos o cualquier otra tecnología para la administración de dato

2.1.2.1. Estructurados

Organizan 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 los datos. Por ejemplo, las bases de datos relacionales utilizan este modelo para organizar los datos.

2.1.2.2. Semi-Estructurados

Son una forma de datos estructurados que no aplican a ningún modelo formal de estructura (generalmente asociados a las bases de datos relacionales), estos datos suelen contener etiquetas o algunas otras marcas para identificar y separar los elementos semánticos fortaleciendo una jerarquía dentro de los mismos datos.

Este concepto de datos semi-estructurados se genera de los populares lenguajes de marcado, como por ejemplo, eXtensible Markup Language (XML), muy usado en páginas web y en tecnologías orientadas a servicios web, o también JavaScript Object Notation (JSON), usado comúnmente por su facilidad de procesamiento y basado en el lenguaje de programación Javascript.

2.1.2.3. No Estructurados

Aquellos datos los cuales no siguen un modelo de datos predefinido o que no están organizados de una manera bien definida. También pueden ocurrir que la data es estructurada aunque no está formalmente definida en un modelo de datos. Los datos no estructurados suelen ser pesados

en texto, aunque pueden contener información como fechas, números y hechos. Esto resulta en ambigüedades que hacen que sea mucho más difícil el procesamiento utilizando algoritmos tradicionales, lo cual lleva también a utilizar mecanismos de almacenamiento más complejos.

2.2. Apache Hadoop

Es un framework desarrollado en Java y de licencia libre que permite el almacenamiento y procesamiento distribuido de grandes volúmenes de datos usando modelos de programación Map-Reduce. Es utilizado para almacenar, procesar y analizar grandes volúmenes de datos de manera eficiente a través de clusters, donde su diseño permite escalar de pocos nodos a miles de nodos de forma ágil. En lugar de depender de hardware de alta gama, la fortaleza de estos clusters se debe a la capacidad que tiene el software para detectar y manejar fallas a nivel de aplicaciones.

Si bien existen otros sistemas que realizan procesamiento de grandes volúmenes de datos en sistemas distribuidos, Hadoop tiene la ventaja de proveer un modelo de programación simple, el cual permite escribir y testear sistemas distribuidos de forma rápida. Además provee un sistema eficiente de distribución automática de datos y trabajo en el conjunto de nodos, y también dentro de cada nodo con sus respectivos núcleos.

2.2.1. Cloudera

La distribución open-source de Apache Hadoop, CDH (Cloudera Distribution Hadoop) se enfoca en el desarrollo de esta tecnología para empresas. Según la compañía, más del 50 % de las ganancias son donadas a diferentes proyectos open source (Apache Hive, Apache Avro, Apache HBase, entre otros) que se suman para formar la plataforma Hadoop. CDH contiene el núcleo, los principales elementos de Hadoop [7] que proporcionan un procesamiento de datos fiable y escalable (básicamente MapReduce y HDFS), así como otros componentes específicos para empresas que aportan seguridad, alta disponibilidad e integración con hardware y software.

2.2.2. Hadoop Distributed File System

Existen sistemas los cuales necesitan almacenar una cantidad de datos enorme, esta cantidad de datos puede que no pueda ser almacenada dentro de un solo nodo físico, por lo tanto, se ha recurrido al almacenamiento en sistemas de archivos distribuidos, los cuales permiten mediante una conexión de red y distintos computadores, compartir archivos de manera tal que para los nodos es transparente el lugar donde se almacenan dichos datos.

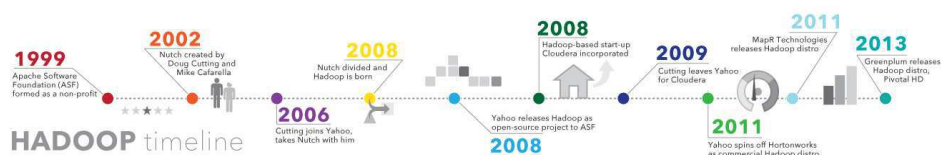


Figura 2.1: Línea de tiempo Hadoop (Fuente: SAS)

Hadoop Distributed File System (HDFS) es un sistema de archivos distribuido, escalable y portátil escrito en Java y creado especialmente para trabajar con ficheros de gran tamaño [19]. Su diseño está basado en el diseño de GFS, el sistema de archivos de Google, es el sistema primario de almacenamiento de datos usado por las aplicaciones de Hadoop, el cual replica los bloques de datos y los distribuye en nodos del clúster. Es esta distribución y redundancia la que permite el acceso rápido y la tolerancia a fallos en los nodos del clúster. Una de sus principales características es un tamaño de bloque muy superior al habitual (64 MB) que otros sistemas de archivos distribuidos, para no perder tiempo en los accesos de lectura.

Características

HDFS posee las siguientes características las cuales lo hacen relucir frente a otros sistemas de archivos: Manejo de grandes archivos: cuando se hace referencia a grandes archivos, son aquellos que pueden pesar cientos

de Mega-Bytes, Giga-Bytes, Tera-Bytes, Peta-Bytes, etc.

- **Compatibilidad con hardware:** Fue diseñado para aceptar hardware común, que se puede encontrar en cualquier tipo de servidores. No se necesitan marcas específicas, ni modelos específicos de discos para poder funcionar.
- **Acceso a datos en flujos:** La construcción de HDFS fue pensada para manejar datos de manera eficiente, se maneja sobre un patrón el cual se escribe una vez un dato, pero se leen múltiples veces. 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:** Para poder tener siempre disponible los datos en caso de ser requeridos, utiliza la replicación de los datos en distintos nodos (por defecto 3).

Pero, también tiene algunas desventajas:

- **Acceso a datos con baja latencia:** Fue creado para manejar grandes volúmenes de datos, por lo tanto, está optimizado para tener altas tasas de transferencia.
- **Escritura múltiple:** Los archivos deben 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 metadatos 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.

2.2.2.1. Arquitectura

Antes de hablar de la arquitectura básica de HDFS se debe hablar sobre la manera la cual HDFS almacena los datos. Estos datos son almacenados de una forma muy parecida a los sistemas de archivos convencionales donde se separan los discos en bloques de un tamaño predeterminado. HDFS adopta este concepto de bloques y toma una unidad mucho más grande que los discos convencionales, 64MB, mientras que los discos convencionales poseen bloques de 512B, dependiendo de su configuración. De tener un dato mayor de 64MB este es cortado en distintos trozos de 64MB (por defecto) los cuales permitirán almacenar y distribuir todos los trozos en un cluster y así poder realizar el almacenamiento de forma distribuidas.

Un Clúster HDFS tiene dos tipos de nodos, diferenciados completamente según el rol o la función que vayan a desempeñar a la hora de ser usados, son los siguientes:

- Namenode (JobTracker): Este tipo de nodo, del que solo hay uno por clúster, es el más importante ya que es responsable de la topología de todos los demás nodos y por consiguiente, de gestionar el espacio de nombres.
- Datanode (TaskTracker): Este tipo de nodos, de los que normalmente van a existir varios, son los que realizan el acceso a los datos propiamente dicho. En este caso, almacenan los bloques de información y los recuperan bajo demanda.

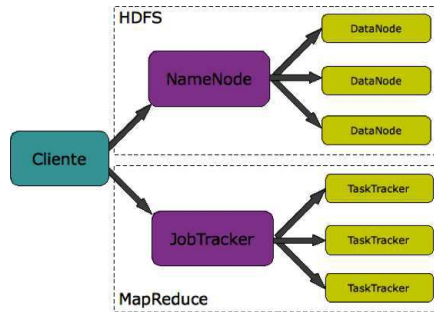


Figura 2.2: Esquema HDFS (Fuente: MadridSchool)

Como se mencionó anteriormente, existe una dependencia enorme con el namenode debido a que en este se encuentra toda la metadata necesaria para poder rearmar un dato, ya que estos están divididos en distintos bloques. Por eso, es muy importante tener mecanismos para poder mantener estos nodos siempre activos, Hadoop provee dos mecanismos clave para poder lograrlo:

1. Se realiza un respaldo de la metadata actual. Este respaldo normalmente es realizado en distintos sistemas de archivos, uno de los utilizados es el sistema de archivos local del namenode y otro de los utilizados es en algún nodo secundario el cual provea almacenamiento por red mediante NFS.
2. Se podría mantener un namenode secundario el cual se activaría al fallar el namenode principal. Mientras el namenode principal esté en funcionamiento, el secundario no se deja mostrar en el cluster como un namenode. Este se encarga de realizar las mismas funciones que el namenode principal, es decir, mantener la metadata de todos los directorios y archivos.

2.2.2.2. Escalabilidad

La manera la cual fue construido HDFS permite que sea un sistema de archivos escalable horizontalmente, el problema se puede presentar al

momento de ingresar nuevos nodos al sistema y estos lo saturan, en este apartado se hablará sobre las limitaciones de escalabilidad y a que están asociadas.

Uno de los problemas principales viene dado por el namenode, el cual mantiene la metadata del cluster en memoria RAM, lo cual es un problema debido a que los servidores tienen un límite de memoria RAM el cual puede ser instalado[24]. Este problema se presenta debido a que cada directorio, archivo o bloque almacenado ocupa alrededor de 150B, lo cual hace que al tener grandes volúmenes de datos, este problema se vea presente en un cluster Hadoop.

Una de las fallas existentes en un cluster Hadoop es que durante el uso de este, la relación bloque/archivo puede afectar al namenode, debido a que el tamaño de los bloques tiende a disminuir lo cual hace que se tenga más metadata y eso perjudica el sistema ya que se invierte mucho tiempo actualizando las tablas para mantener el sistema en orden.

Un problema relacionado con las tablas las cuales deben ser actualizadas es la cantidad de mensajes que deben ser enviados desde los datanodes hacia el namenode, al tener que enviar muchos mensajes, la red del cluster podría colapsar como también puede colapsar el namenode debido a que este tiene una capacidad de procesamiento limitada. La carga relacionada con los datanodes la cual es sometida al namenode es proporcional a la cantidad de datanodes que se encuentren en el cluster. Se puede decir que el cuello de botella principal de un cluster Hadoop es el namenode debido a que tanto clientes como datanodes dependen de este.

2.2.3. MapReduce

MapReduce es un modelo de programación con una implementación asociada al procesamiento y generación de grandes cantidades de datos. Los usuarios especifican una función de map que procesa pares clave/valor para generar un grupo intermedio de pares clave/valor y una función de reduce que combina todos los valores intermedios.

Programas escritos de esta manera son automáticamente paralelizados y ejecutados en un clúster Apache Hadoop. El sistema en tiempo de ejecución se encarga de cosas como los detalles de particionar los datos de entrada, la planificación del programa en ejecución a través de todas las maquinas, encargarse de manejar fallos y administrar la comunicación necesaria entre maquinas. Esto permite a programadores sin experiencia con sistemas paralelos y distribuidos a utilizar fácilmente los recursos del sistema.

MapReduce fue desarrollado por Google y expuesto al resto del mundo en una publicación hecha por ellos mismos en diciembre del 2004, Google usa MapReduce para indexar páginas web.

Específicamente la implementación de Apache Hadoop MapReduce es un framework con una serie de librerías para facilitar escribir aplicaciones usando este esquema.

2.2.3.1. Conceptos Básicos de MapReduce

Los programas que funcionan bajo este modelo están compuestos de dos procedimientos esenciales llamados map y reduce, básicamente entre esos dos procedimientos se puede resumir todo el flujo de datos que hace el programa. MapReduce también incorpora un esquema de tolerancia a fallos que permite responder de una forma eficaz a casi cualquier problema que se pueda presentar al momento de la ejecución del programa. Cada uno de estos conceptos será desarrollado más adelante.

Mapping Lists La primera fase de un programa que corre bajo un esquema de MapReduce es llamada mapping. Una lista con elementos de datos es dada en un instante de tiempo a una función llamada Mapper (Mapping function), la cual transforma cada elemento individualmente a un elemento de datos de salida

Reducing Lists Reducing permite juntar todos los valores. Una función reducir recibe un iterador de valores de entradas de una lista de entrada. Entonces combina estos valores y retorna un único valor de salida.

Reducing es usado frecuentemente para producir un resumen de los datos, cambiando un gran volumen de datos a una pequeña cantidad de los mismos

Flujo de Datos La entrada de datos para MapReduce viene típicamente de archivos cargados en el clúster en HDFS (ver 2.6.1). Estos archivos son igualmente distribuidos a través de todos los nodos del clúster. Un programa que corre con un esquema de MapReduce involucra correr tareas de mapping en varios o todos los nodos del clúster. Cada una de estas tareas de mapping es equivalente, es decir, no tienen identificadores particulares asociados. De esta manera, cualquier mapper puede procesar cualquier archivo de entrada. Cada mapper carga un conjunto de archivos locales a la máquina que los está procesando.

Cuando la fase de mapping haya completado, los pares (clave/valor) intermedios deben ser cambiados entre máquinas para mandar todos los valores de la misma clave a un reducer. Las tareas de reduce se esparcen a través de los mismos nodos que los mappers. Este es el único paso de comunicación en MapReduce. Las tareas de map individuales no cambian información una con otra, y ni están conscientes de la existencia de las otras tareas. Similarmente, tareas diferentes de reduce no se comunican una con otra. El usuario nunca tiene control de cómo es la transferencia de datos, toda la transferencia es manejada por la plataforma de Hadoop MapReduce; guiada implícitamente por las diferentes claves asociadas a los valores. Si los nodos del clúster fallan, las tareas deben poder ser reiniciadas.

Finalmente los componentes que están presentes en el flujo de datos de un programa que corre con un esquema de MapReduce son los siguientes:

- **Input reader:** se encarga de dividir los datos de entrada al tamaño apropiado (suele ser 64 MB hasta 128MB) y el framework asignará cada pedazo a una función de map. Estos datos generalmente son leídos del HDFS y generan pares clave/valor.
- **Función de Map:** la función de Map agarra una serie de pares clave/valor, procesa cada uno, y genera cero o más salidas de pares clave/valor.

- **Función de Partición:** cada salida de la función de map es colocada en un reducer en particular por la función de partición de la aplicación. A la función de partición se le da la clave y el número de reducers y retorna un índice para el reducer deseado.
- **Función de Comparación:** la entrada de cada función de reduce es obtenida de la maquina donde el map corrió y son ordenadas usando la función de comparación de la aplicación.
- **Función de Reduce:** el framework llama a la función de reduce de la aplicación una vez por cada clave única ordenada. Reduce itera por los distintos valores que están asociados con esa clave y puede producir cero o más salidas.
- **Escritor de salida:** la salida de la función de reduce es escrita en algún repositorio, usualmente el mismo HDFS.

Tolerancia a Fallos La forma en la que MapReduce o Hadoop en general logra la tolerancia a fallos es a través del reinicio de las tareas. Tareas individuales corriendo en los nodos (TaskTrackers) están en constante comunicación con la cabeza del nodo del sistema, llamado JobTracker. Si un TaskTracker falla al comunicarse con el JobTracker por un período de tiempo (por defecto en Hadoop, un minuto), el JobTracker va a asumir que el TaskTracker en cuestión falló. El JobTracker conoce que tareas de map y reduce fueron asignadas a cada TaskTracker. Si el trabajo todavía está en la fase de mapping, entonces otros TaskTrackers serán asignados de volver a ejecutar todas las tareas de map del TaskTracker en específico que falló. Si el trabajo está en su fase de reduce, entonces los otros TaskTrackers van a volver a ejecutar automáticamente las tareas del TaskTracker que falló.

Esta tolerancia a fallos necesita que los programas en ejecución sean los más individuales posible, es decir, si cada tarea map o reduce tuviese su identidad propia y se comunicara con el mundo exterior (a través de la red por ejemplo) o con otras tareas, entonces reiniciar esas tareas se vuelve un poco más complicado porque hay que tomar en cuenta el estado que tenía en el momento que falló. Este proceso es realmente complicado y propenso a errores. MapReduce simplifica este problema de cierta forma evitando las

identidades o que las tareas se comuniquen entre ellas. Una tarea individual solo puede trabajar con sus propios datos y conoce solo sus propias salidas, para hacer este fallo y proceso de reinicio limpio y no dependiente.

2.3. Sistema Operativo

Existen distintos tipos de sistemas operativos y estos pueden tener distintas definiciones dependiendo de su uso. Un Sistema Operativo puede ser un programa el cual administra el hardware de un computador. También provee una base para poder ejecutar programas sobre este hardware, actuando como intermediario entre el hardware y los programas a ejecutar [2].

También un sistema operativo puede proporcionar a los programadores de aplicaciones un conjunto abstracto de recursos simples, en vez de los complejos conjuntos de hardware [33]. Es decir, nos ayuda a observar el hardware con el cual vamos a interactuar de una forma mucho más sencilla, ya que se realizan abstracciones para el acceso a este.

2.4. Objeto digital

Un objeto digital es una instancia de un tipo de dato abstracto que tiene dos componentes, el dato y la metadata[25]. Hay que reconocer la importancia de la metadata, puesto que un archivo que está siendo creado, es un estado de ejecución, más no un objeto digital. Por ejemplo, un documento de un editor de texto no se le identifica como documento hasta que dicho documento es guardado, al guardarlo, se le asocia un autor, un título, una fecha de modificación, un formato de archivo, etc.

El Contenido de un objeto digital y sus metadatos forman una unidad o dicho de otra manera, un contenido sin metadatos o unos metadatos sin contenidos, no son un objeto digital, se forma por la mezcla de ambos. Por lo general el tamaño digital (bytes) de los metadatos es inferior (muy inferior) al tamaño del contenido.

2.5. Repositorio digital

Es un almacén que puede servir para varios propósitos según la información que contiene, su utilidad es la de otorgar acceso desde cualquier acceso a documentos, archivos, etc; por medio de la red. En otras palabras, su función es recolectar, dar acceso y preservar objetos digitales.

2.5.1. Fedora Commons

También llamado Flexible Extensible Digital Object Repository Architecture, es una arquitectura modular con licencia Apache y basada en el principio de que la interoperabilidad y extensibilidad se consiguen mejor mediante la integración de datos, interfaces, y mecanismos como módulos claramente definidos. Fedora posee una arquitectura de gestión de activos digitales (*Digital Asset Management*, DAM), sobre la cual se pueden construir muchos tipos de bibliotecas digitales, repositorios institucionales, archivos digitales, y sistemas de bibliotecas digitales.

En un repositorio de Fedora, todo el contenido se gestiona como objetos de datos, cada uno de los cuales está compuesto de componentes (datastreams) que contienen tanto el contenido como los metadatos sobre él. Cada datastream puede ser gestionado directamente por el repositorio o una ubicación externa, accesible desde la web para ser entregados a través del repositorio, según sea necesario. Un objeto de datos puede tener cualquier número de datos y componentes de metadatos[8].

Fedora soporta dos tipos de servicios de acceso: un cliente de gestión para ingestión, mantenimiento, y exportación de objetos; o una vía API para servicios de acceso basados en web construidos mediante HTTP o bien SOAP. Un repositorio Fedora proporciona una capa de gestión general para objetos digitales, y contenedores que agregan fuentes de datos MIME-typed (pueden ser imágenes digitales, archivos XML, metadatos). Fedora soporta importación y exportación de objetos digitales en variedad de formatos XML. Esto permite intercambios entre Fedora y otras aplicaciones basadas en XML y facilita las tareas de archivado.

2.5.1.1. Modelo Objeto Digital

Fedora utiliza un diseño 'objeto digital compuesto' que agrega uno o más elementos de contenido en el mismo objeto digital. Los elementos de contenido pueden ser de cualquier formato y también pueden ser almacenados localmente en el repositorio o externamente y referenciado por el objeto digital. Este modelo es simple y flexible de modo que pueden ser creados muchos objetos de varios tipos. El repositorio Fedora permite gestionar todos estos objetos de una manera consistente.

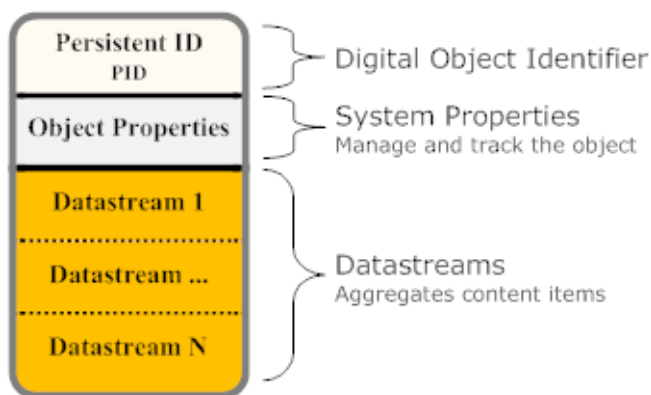


Figura 2.3: Componentes de un objeto digital. Fuente: FEDORA

Los componentes básicos de un objeto digital Fedora son:

- **PID:** Identificador único del objeto.
- **Object Properties:** Un conjunto de propiedades descriptivas definidas por el sistema que son necesarias para la gestión y seguimiento del objeto en el repositorio.
- **Datastream(s):** Es un elemento que representa el contenido del objeto.

2.5.1.2. Datastreams

Es el elemento de un objeto digital Fedora que representa al contenido. Un objeto digital Fedora puede tener uno o más Datastreams. Cada Datastream registra atributos útiles sobre el contenido que representa como el tipo MIME (para la compatibilidad Web)[21] y, opcionalmente, el URI que identifica formatos de contenido.

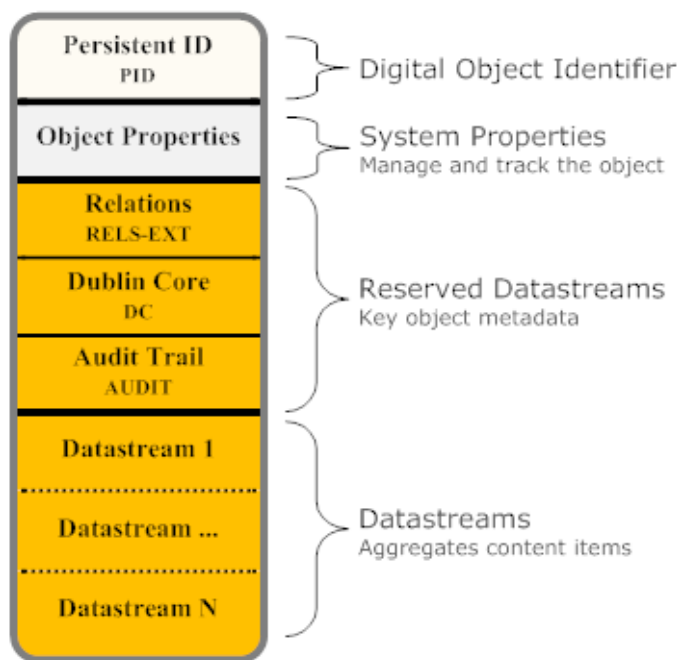


Figura 2.4: Datastreams reservados

A cada Datastream se le asigna un identificador único dentro del objeto digital. Fedora se reserva cuatro identificadores de Datastream para su uso, DC, AUDIT, RELS-EXT y RELS-INT. Cada objeto digital Fedora tiene un (Dublin Core) Datastream DC por defecto que se utiliza para contener metadatos sobre el objeto (y se creará automáticamente si no se proporciona ninguno). Fedora también mantiene un flujo de datos especial,

AUDIT, que registra todos los cambios realizados en el objeto, y no se puede modificar ya que sólo el sistema lo controla. El RELS-EXT se utiliza principalmente para proporcionar un lugar consistente para describir las relaciones con otros objetos digitales y RELS-INT para describir las relaciones internas de los Datastreams de objetos digitales. Además, un objeto digital Fedora puede contener cualquier número de Datastreams personalizados para representar el contenido definido por el usuario.

Las propiedades básicas que el modelo de objetos Fedora define para un Datastream son las siguientes:

- Datastream Identifier: Identificador único dentro del objeto digital.
- State: Estado del Datastream que puede ser Active, Inactive o Deleted.
- Created Date: Fecha en la que se creó el Datastream, es asignada por el servicio del repositorio.
- Modified Date: Fecha en la que fue modificado el Datastream, es asignada por el servicio del repositorio.

2.5.1.3. FOXML

FOXML (Fedora Object XML) es un sencillo formato XML que expresa directamente el modelo de objetos de Fedora digital. Los objetos digitales se almacenan internamente en un repositorio Fedora en el formato FOXML. Además, FOXML se puede utilizar para la ingestión y la exportación de objetos desde y hacia repositorios de Fedora.

La introducción de FOXML fue motivada por una serie de requisitos: simplicidad, optimización de rendimiento y flexibilidad en la evolución de Fedora. En cuanto a la sencillez, la retroalimentación de los usuarios sugiere que el mapeo de los conceptos de Fedora en un formato XML es más fácil, conceptualmente. Los usuarios querían una forma intuitiva de cómo crear objetos Fedora, especialmente para aquellos que no están familiarizados con formatos tales como METS[20]. En cuanto a la optimización y rendimiento, el esquema FOXML fue diseñado para mejorar el rendimiento del

repositorio, tanto en la ingesta y durante diseminaciones. El rendimiento de ingestar objetos fue afectado positivamente con la introducción de FOXML, especialmente en las fases de validación. En cuanto a la flexibilidad, estableciendo FOXML como formato de almacenamiento interno para los objetos de Fedora permite una evolución más fácil de la funcionalidad en el repositorio de Fedora, sin necesidad de extensiones en curso a otros formatos de la comunidad.

2.5.1.4. Comunicación con el usuario

Fedora provee un conjunto de servicios directamente asociados con los datos empaquetados en el objeto digital, de esta manera, cuando los servicios cambian, los objetos heredan automáticamente los cambios[31]. Estos servicios pueden ser invocados por REST o SOAP y se dividen en API-A, una interfaz para el acceso a los objetos digitales[14], el otro conjunto de servicios llamados API-M[15] define la interfaz para la administración del repositorio incluyendo crear, modificar y eliminar objetos digitales o componentes internos. El objetivo de ambos servicios es permitir al usuario desde una perspectiva abstracta la posibilidad de ver y manipular objetos digitales sin conocer sobre formatos de almacenamiento, tipos de archivos, esquemas para objetos, etc.

Cuadro 2.1: Métodos API-A

Acceso al repositorio	describeRepository
Acceso a objetos	findObjects resumeFindObjects getObjectHistory getObjectProfile
Acceso a datastreams	getDatastreamDissemination listDatastreams
Acceso a diseminacion de objetos	getDissemination listMethods

Cuadro 2.2: Métodos API-M

Manejo de datastreams	addDatastream compareDatastreamChecksum getDatastream getDatastreamHistory getDatastreams modifyDatastreamByReference modifyDatastreamByValue setDatastreamState setDatastreamVersionable purgeDatastream
Manejo de objetos	modifyObject purgeObject export getNextPID getObjectXML ingest validate

2.6. Metadatos

Una definición utilizada con frecuencia nos dice que los metadatos son "datos sobre datos", en general un objeto que describe o dice algo sobre otro objeto de información[26]. Este concepto puede describirse mejor haciendo analogía con el uso de índices para localizar objetos en vez de datos. Por ejemplo, en una biblioteca se usan fichas que especifican autores, títulos, casas editoriales y lugares para buscar libros. Así, los metadatos ayudan a ubicar datos.

Los beneficios de utilizar metadatos son diversos y dependen del área en que se utilicen. En términos generales:

- Adhieren contenido, contexto y estructura a los objetos de información, asistiendo de esta forma al proceso de recuperación de conocimiento desde colecciones de objetos.

- Permiten el intercambio de la información sin la necesidad de involucrar el intercambio de los recursos mismos. Esta particularidad facilita entre otras cosas las búsquedas sobre colecciones distribuidas.
- Por último, permiten una descripción precisa y discreta de los recursos permitiendo la creación de colecciones virtuales de descripciones donde agrupan los objetos de información para satisfacer requerimientos específicos. Un ejemplo podría ser una institución educacional que recolecta materias de cursos desde distintas instituciones del globo agrupadas por temas, sin importar el formato del material recolectado[26].

2.6.1. Estándares de metadatos

Un estándar de metadatos es un documento que identifica contenido que se debe proporcionar para describir recursos geospaciales como mapas, servicios de mapas, datos vectoriales, imágenes e incluso recursos no espaciales como tablas y herramientas que son relevantes a su trabajo espacial. Un estándar de metadatos también puede proporcionar un esquema XML que describe el formato en el que se debe almacenar el contenido. Por lo general, un formato XML estándar se define utilizando un esquema XML o una definición de tipo de documento (DTD). Los estándares por lo general se ratifican mediante conjuntos de estándares nacionales o internacionales.

La generación de estándares de metadatos es una inversión en cuanto a la futura interoperatividad ya que expande las posibilidades de las distintas partes para trabajar efectivamente en el largo plazo, sin importar el cambio de tecnología.

Si catalogamos un pequeño grupo de canciones, e-mails y especímenes biológicos, utilizando una relación persona-objeto, podemos decir que una canción tiene un autor y un título así como un e-mail tiene un destinatario y asunto. Siguiendo la misma lógica también podemos decir que un espécimen biológico tiene un 'recolector' y un nombre. Ahora bien, si una persona busca algo en este grupo de datos (con la relación persona-objeto en mente) es muy posible que encuentre lo que busca, sin embargo esta relación

persona-objeto no contempla detalles como el de que un 'recolector' no es quien crea el nombre del organismo al contrario de como pasa con un e-mail.

En materia de metadatos las comunidades no acuerdan consenso para establecer criterios y estándares, lo que es lógico ya que existen innumerables formas de organizar objetos. Hasta el día de hoy ningún estándar ha logrado aceptación global.

Para facilitar la comprensión global de los metadatos existentes es necesario clasificarlos. La clasificación sugerida se realiza mediante grupos o categorías de acuerdo a los propósitos generales de cada marco de metadatos.

Administrativos: Se refieren a información provista para facilitar la administración de los recursos. En este conjunto tienen cabida datos sobre cuando y como un objeto fue creado, quien es el responsable de controlar el acceso o registrar su contenido, que actividades de procesamiento fueron efectuados en relación al contenido y que restricciones de acceso o de uso son aplicables. Un ejemplo son los utilizados para la preservación que apuntan específicamente a apoyar la retención a largo plazo de los objetos digitales y dependiendo del contexto, a su reconstrucción en caso de pérdida.

Descriptivos y de descubrimiento: Se refieren a la información provista para encontrar, describir y distinguir cada uno de los objetos de información. Dublin Core es el ejemplo mas claro de este tipo de metadatos. En esta categoría tienen cabida también los metadatos encargados de describir recursos de dominios específicos del conocimiento. Ejemplos para el campo de las ciencias serian los metadatos de Darwin Core que proveen representación para la búsqueda y recuperación de colecciones de historia natural y los pertenecientes al *Data Documentation Initiative* (DDI) [11] el estándar que sirve para describir conjuntos de datos para uso en ciencias sociales.

Técnicos: Corresponden a los estándares de metadatos relacionados con los elementos que describen como un sistema funciona o debe ser interpretado. Un ejemplo de estos son los metadatos que describen el formato

de alguna imagen digital.

Modelos: Tienen relación con objeto de información compuesto, describe como se interrelacionan cada uno de sus componentes. Por ejemplo un metadato puede describir que, en el contexto de un libro, llegaremos a un tema deseado si seguimos el numero de pagina indicado en el indice y que ademas las paginas están ordenadas.

Es importante tomar en cuenta que los límites entre estas categorías tienden a ser difusos, por lo que muchos de los metadatos no caben en sólo una de las categorías. Así en un mismo esquema de metadatos se incluyen componentes con distintos propósitos y alcances.

Una clasificación formal en que se agrupan los metadatos en solo una de estas tres categorías no representa adecuadamente a la realidad, por lo que se puede utilizar un diagrama triangular para visualizar la clasificación.

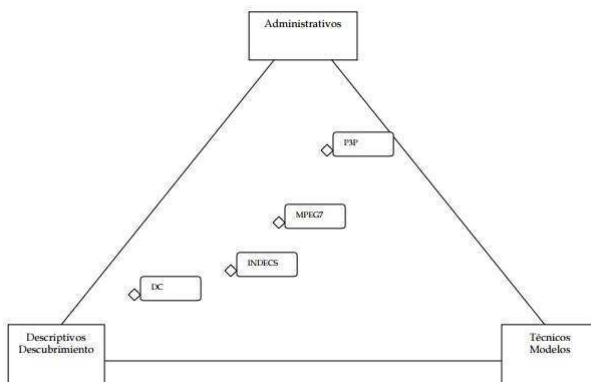


Figura 2.5: Estándares de metadatos

En el presente diagrama se encuentran clasificados cuatro (de innumerables) estándares, donde un metadato situado cerca una de las categorías indica un mayor numero de componentes del que tienen como finalidad

cumplir con dicho propósito general.

Así es como Dublin Core (DC) [12] sitúa en la categoría de metadatos descriptivos y de descubrimiento, y MPEG7 se sitúa cerca del centro del diagrama por poseer elementos que cumplen con los tres propósitos generales.

2.6.2. Dublin Core

Es un modelo de metadatos elaborado y auspiciado por la DCMI (*Dublin Core Metadata Initiative*), una organización dedicada a fomentar la adopción extensa de los estándares interoperables de los metadatos y a promover el desarrollo de los vocabularios especializados de metadatos para describir recursos para permitir sistemas más inteligentes el descubrimiento del recurso.

Dublin Core posee 15 definiciones descriptivas que pretenden transmitir un significado semántico para el usuario. Podemos clasificar el conjunto de elementos Dublin Core en 3 grupos que indican la clase o el ámbito de la información que contienen:

- Elementos relacionados con el contenido del recurso:
 - **DC.Title:** Nombre dado a un recurso, habitualmente por el autor.
 - **DC.Subject:** Temas del recurso. Típicamente, Subject expresará las claves o frases que describen el título o el contenido del recurso.
 - **DC.Description:** Una descripción o resumen, dependiendo del tipo de recurso.
 - **DC.Source:** Secuencia de caracteres que identifican un trabajo a partir del cual proviene el recurso actual.
 - **DC.Language:** Lenguajes del contenido del recurso.
 - **DC.Relation:** Identificador de un segundo recurso y su relación con el recurso actual.

- **DC.Coverage:** Es la característica de cobertura espacial y/o temporal del contenido intelectual del recurso. La cobertura espacial se refiere a una región física, utilizando por ejemplo coordenadas. La cobertura temporal se refiere al contenido del recurso, no se refiere a la fecha de creación (que ya lo encontramos en el elemento Date).
- Elementos relacionados con el recurso cuando es visto como una propiedad intelectual:
 - **DC.Creator:** La persona u organización responsable de la creación del contenido intelectual del recurso.
 - **DC.Publisher:** Entidad responsable de hacer que el recurso se encuentre disponible en la red.
 - **DC.Contributor:** Persona u organización que haya tenido una contribución intelectual significativa, pero que esta sea secundaria en comparación con los aportes de personas u organizaciones especificadas en el elemento Creator.
 - **DC.Rights:** Referencia (por ejemplo, una URL) para una nota sobre derechos de autor, para un servicio de gestión de derechos o para un servicio que dará información sobre términos y condiciones de acceso a un recurso.
- Elementos relacionados con la instanciación del recurso:
 - **DC.Date:** Fecha en la cual el recurso se pone a disposición del usuario.
 - **DC.Type:** Categoría del recurso. Por ejemplo, página personal, romance, poema, diccionario, etc.
 - **DC.Format:** Formato de datos de un recurso, usado para identificar el software y, posiblemente, el hardware que se necesitaría para mostrar el recurso.
 - **DC.Identifier:** Secuencia de caracteres utilizados para identificar unívocamente un recurso.

2.6.3. OAI-PMH

Es un protocolo basado en HTTP para emitir preguntas y obtener respuestas entre un servidor o archivo y un cliente o servicio recolector de metadatos. El segundo puede pedir al primero que le envíe metadatos según determinados criterios como por ejemplo la fecha de creación de los datos. En respuesta el primero devuelve un conjunto de registros en formato XML, incluyendo identificadores (URLs por ejemplo) de los objetos descritos en cada registro.

Este protocolo genera y promueve estándares de interoperabilidad que facilitan la difusión, intercambio y accesibilidad a documentos de diferente naturaleza, además, [17] OAI - PMH permite almacenar en un solo lugar los metadatos y es allí en donde se realizan las diferentes consultas, el protocolo no define la creación de los metadatos, ni da los parámetros para realizar una consulta, únicamente se ocupa de la gestión de información.

Un sencillo ejemplo es la búsqueda que un usuario realiza en un servidor Web, el usuario envía una petición a un proveedor de servicios, el cual solicita a un proveedor de datos que le envíe registros de metadatos de diferentes recursos con que este disponga.

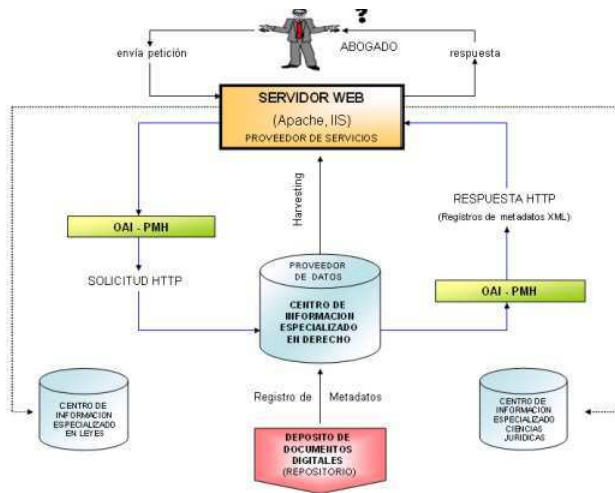


Figura 2.6: Ejemplo búsqueda OAI - PMH

2.6.4. Open Harvester System

Open Harvester System es un sistema de indexación de metadatos gratuito desarrollado por el Public Knowledge Project a través de sus esfuerzos financiados con fondos federales para ampliar y mejorar el acceso a la investigación. Ha sido diseñado pensando en la flexibilidad y soporta múltiples protocolos de recolección y formatos de metadatos con un énfasis en el rendimiento y simplicidad de uso. En concierto con la suite de software PKP, incluyendo *Open Journal Systems* y *Open Conference Systems*, el objetivo de Harvester2 es promover la publicación de acceso abierto y contribuir al bien público a escala global.

OHS permite crear un índice de búsqueda de los metadatos de la *Open Archives Initiative* (OAI), tales como los sitios que utilizan *Open Journal Systems* (OJS) o *Open Conference Systems* (OCS)[23].

2.7. CMS

Mejor conocido como un sistema de gestión de contenidos, un CMS (*Content Management System*) es un sistema con el cual se puede crear y editar contenidos en un medio digital principalmente en su sitio web mediante lenguajes de programación y base de datos.

Algunos lo consideran como Backend ya que es un sistema que se encuentra en el lado del servidor y es invisible para el visitante, únicamente el administrador mediante un acceso privado puede ingresar en la plataforma y gestionar el contenido.

2.7.1. Drupal

Drupal es un sistema de administración de contenidos Web especialmente versátil. En sus orígenes el sistema estaba dirigido a dar soporte a una comunidad de Weblog.

Drupal no está dirigido a un tipo de escenarios específico. El límite de este CMS lo impone el desarrollador; al igual que ocurre con muchos otros CMS, es necesario disponer de un buen conocimiento y experiencia en dicha solución para sacarle el máximo partido.

Dispone de un entorno de personalización robusto, tanto el contenido como la presentación pueden ser tratados de forma individual de acuerdo a unas preferencias definidas por el usuario. La gestión de contenido se realiza como objetos independientes, de forma que puede realizarse un tratamiento individualizado de la información, facilitando su inclusión en cualquier página o permitiendo comentarios específicos sobre cada uno de ellos.

Otros puntos importantes a su favor son el rendimiento y la escalabilidad. Cuenta con sistema de caché avanzado, replicación de base de datos, balanceo de carga, mecanismos de control de congestión configurable para habilitar o deshabilitar módulos.

2.7.1.1. Islandora

Es un framework de código abierto basado en Fedora Commons enfocado a la utilización de estándares abiertos para el acceso y descripción de los datos, manteniendo los altos estándares para la administración de datos y la seguridad en el tiempo. Islandora hace que sea posible crear, editar, descubrir, ver y administrar los objetos del repositorio.

El sistema se esfuerza por lograr un equilibrio entre la extensibilidad y facilidad de uso, proporcionando soporte para las colecciones, mientras que mantiene una arquitectura que se presta a la personalización desde otro software y flujos de trabajo. La base del modelo de administración de datos de Islandora es Fedora, si usted es un usuario de Fedora, usted sigue siendo capaz de acceder y manipular objetos como lo haría en cualquier instalación de Fedora.

El proyecto Islandora aprovecha la potencia del sistema de gestión de contenidos Drupal y los repositorios Fedora para crear un sistema de gestión de activos digitales robusto que se puede utilizar para cumplir con los requisitos de colaboración a corto y largo plazo de la administración de datos digitales.

2.8. Lenguajes de Programación

Un lenguaje de programación es un conjunto de palabras diseñadas para comunicar instrucciones a un computador. También se podría definir como aquel lenguaje que permite especificar de manera precisa sobre qué datos debe operar una computadora, como deben ser almacenados o transmitidos y que acciones debe tomar bajo una variada gama de circunstancias. La descripción de un lenguaje de programación usualmente se divide en dos componentes: la sintaxis (forma en la que se escribe) y la semántica (lo que significa).

Esos lenguajes suelen clasificarse en lenguajes interpretados y compilados. Los lenguajes interpretados tienen un intérprete específico que obtiene como entrada un programa y ejecuta las acciones escritas a medida que

las va procesando; mientras que los lenguajes compilados son llevados a un programa ejecutable utilizando un compilador, este obtiene como entrada un programa y traduce las instrucciones las cuales pueden servir de entrada para otro interprete o compilador.

2.8.1. PHP

PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor, y se trata de un lenguaje de scripting para la programación de páginas dinámicas de servidor. Es un lenguaje de tipo gratuito, y forma parte del software que se conoce como de código abierto (Open Source). Es decir que se le pueden introducir modificaciones y mejoras y ponerlas a disposición de los demás usuarios del mismo.

Ejemplo Hola mundo con PHP embebido en HTML:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <title> Ejemplo básico PHP</title>
  </head>
  <body>
    <?php
      echo 'Hola mundo';
    ?>
  </body>
</html>
```

El intérprete de PHP solo ejecuta el código que se encuentra entre sus delimitadores php. El propósito de estos delimitadores es separar el código PHP del resto de código, como por ejemplo el HTML.

Una aplicación web basada en PHP necesita dos tipos de software. El primero es un servidor web que va a atender las peticiones de los usuarios y devolverá las páginas solicitadas. El servidor Apache, tanto su versión Windows como Linux es el más utilizado. El segundo software es el propio

PHP, es decir el módulo que se va a encargar de interpretar y ejecutar los scripts que se soliciten al servidor.

Al utilizar una tecnología del tipo pre-procesado en el servidor es necesario visualizar las páginas generadas con PHP utilizando el protocolo http. Al contrario de lo que ocurre con las páginas de la tecnología cliente, en las que se puede visualizar mediante la opción “Archivo - Abrir“ en cualquier navegador, las páginas generadas con PHP necesitan ser servidas por un servidor web para que sean procesadas y luego enviadas al navegador del usuario.

2.8.1.1. Smarty

Smarty es un motor de plantillas para PHP. Mas específicamente, esta herramienta facilita la manera de separar la aplicación lógica y el contenido en la presentación. Es común que en grandes proyectos el rol de diseñador gráfico y el de programador sean cubiertos por personas distintas, sin embargo la programación en PHP tiene la tendencia de combinar estas dos labores en una persona y dentro del mismo código, lo que trae consigo grandes dificultades a la hora de cambiar alguna parte del diseño de la página, pues se tiene que escarbar entre los scripts para modificar la presentación del contenido, Smarty tiene como objetivo solucionar este problema.

2.8.2. Java

El lenguaje Java es de propósito general el cual es orientado a objetos, concurrente, basado en clases, portable y especialmente diseñado para tener pocas dependencias en su implementación como sea posible. Se basa en principios como WORA (write once, run everywhere) donde básicamente cualquier programa compilado en este lenguaje puede correr en cualquier equipo sin necesidad de recompilar (portabilidad). Todo esto es posible ya que Java corre en una máquina virtual llamada Java Virtual Machine (JVM) lo cual lo hace independiente de la arquitectura de la computadora donde esté corriendo. Este lenguaje de programación deriva mucho de lenguajes como C y C++, pero con menos funcionalidades de bajo nivel. Una de las características más importantes de Java (aparte de su gran

portabilidad) es su manejo automático de la memoria. Java tiene una implementación de recolección de basura automática la cual se encarga de manejar la memoria en el ciclo de vida de un objeto. Uno de los puntos importantes de Java en el mundo de Big Data es que Hadoop, el framework por excelencia de este mundo, es totalmente dependiente de Java debido a que una gran parte de este framework fue desarrollado sobre Java.

2.9. Bases de datos

Una base de datos es un conjunto estructurado de datos que representa entidades y sus interrelaciones. La representación será única e integrada, a pesar de que debe permitir utilizaciones varias y simultáneas[30].

La particularidad definitiva que convierte a un conjunto de datos en una base de datos es la siguiente: una base de datos se controla por medio de Sistemas de Gestión de Bases de Datos(SGBDs). Ellos definen los modelos con lo que se van a organizar las bases de datos.

El modelo mas usado es el relacional, esta organización ofrece la mayor flexibilidad ya que los datos se almacenan en tablas diferentes, conformadas así mismo por filas y columnas. Una tabla se denomina relación. En una tabla las filas contienen los registros. Las columnas representan los campos. Las tablas relacionadas poseen un campo común, el campo clave, mediante el cual la información almacenada en una tabla puede enlazarse con la información almacenada en otra[29].

2.9.1. Bases de datos relacionales

Una base de datos relacional es una colección de datos organizados en un grupo de tablas formalmente descritas a través de un esquema (Schema), estas tablas pueden ser accedidas o re-ensambladas. La manera estándar para acceder a una base de datos y que suele servir de interfaz para los usuarios de estas es el *Structured Query Language* (SQL). Las sentencias SQL suelen usarse para obtener información de las bases de datos, agregar

información, borrarla o modificarla.

En terminología de bases de datos relacionales, cuando se refieren a las tablas, las llaman relaciones; a las columnas, atributos y a las filas, tuplas.

Una *relación* es un grupo de tuplas que contienen los mismos atributos. Una tupla suele representar un objeto y la información acerca de ese objeto. En las bases de datos relacionales, cada tupla tiene una clave primaria que es simple si la representa un atributo o es compuesta si mas de un atributo la representa. Además cada tupla puede tener una clave foránea que hace referencia a otra relación, donde en dicha relación la clave foránea se convierte en una clave primaria. Un ejemplo de algunas bases de datos relacionales: MySQL, Oracle, PostgreSQL o MariaDB.

2.9.1.1. MySQL

MySQL es un sistema de gestión de base de datos relacional [22](RDBMS) de código abierto, basado en lenguaje de consulta estructurado (SQL).

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL, como base de datos relacional, utiliza multiples tablas para almacenar y organizar la información. MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interacción con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.

También es muy destacable, la condición de open source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente.

2.10. Akubra

El Proyecto Akubra es un plugin que provee de una interfaz de almacenamiento de archivos que se puede adaptar a casi cualquier sistema

de almacenamiento[1]. Es compatible con sistemas de almacenamiento tradicionales y transaccionales. Se encarga de facilitar la simplificación del manejo del sistema de archivos con el fin de lograr un alto nivel de interoperabilidad entre distintos sistemas de almacenamiento, este define:

- Blob es un flujo de bits de longitud finita con un id (URI).
- Almacenador de Blob se encarga principalmente de proporcionar acceso de lectura/escritura a los Blob.

2.10.1. Akubra Low Level Storage

También llamado "LLStore", la interfaz de almacenamiento de bajo nivel es un componente crítico de Fedora Commons. Almacena y proporciona acceso a la copia autorizada de todos los objetos XML (FOXML) y flujos de datos gestionados por un repositorio de Fedora.

El módulo LLStore [9] almacena por defecto en Fedora un objeto digital en formato XML y flujos de datos como archivos individuales en un sistema de archivos convencional, existe un archivo de configuración

2.11. Amazon Web Services

Amazon Web Services (AWS) es una plataforma de servicios de nube que ofrece potencia de cómputo, almacenamiento de bases de datos, entrega de contenido y otra funcionalidad para ayudar a las empresas a escalar y crecer.

Amazon Simple Storage Service (S3) es un servicio de almacenamiento en Internet. Provee una interfaz simple de servicios Web que puede ser utilizada para almacenar y recuperar cualquier cantidad de datos en cualquier momento, en cualquier lugar de la Web. Está diseñado para facilitar el diseño de aplicaciones escalables en Internet. Amazon Elastic Cloud Computing (EC2) es un servicio web que provee una flexible capacidad de procesamiento para aplicaciones en la nube. Este servicio provee un ambiente de cómputo virtual, sobre el cual pueden usarse servicios Web para levantar instancias de una variedad de sistemas operativos cargados con un

ambiente de aplicaciones predefinidas.

2.11.1. Costos

El costo es uno de los factores decisivos para la adopción de cualquier tecnología. Esto es especialmente cierto cuando se habla de instituciones con presupuesto ajustado. La ejecución en hardware de bajo costo es una de las características a las cuales Hadoop hace mención. Hadoop está diseñado para manejar los fallos, y no requiere de hardware tolerante a fallos de alto costo.

El objetivo principal del clúster Hadoop en nuestro caso es el de proporcionar un servicio de almacenamiento escalable y robusto. Para un clúster optimizado hacia almacenamiento, cada nodo debe tener varios discos de almacenamiento de alto volumen con suficientes recursos de memoria y procesamiento. Aunque ciertos servidores pueden soportar un gran número de discos de almacenamiento, hay factores que pueden limitar el número de discos por nodo. Haciendo que la distribución de datos se vea afectada por el exceso de almacenamiento por nodo.

Hadoop Storage Cost/GB (For Replication Factor: 3)

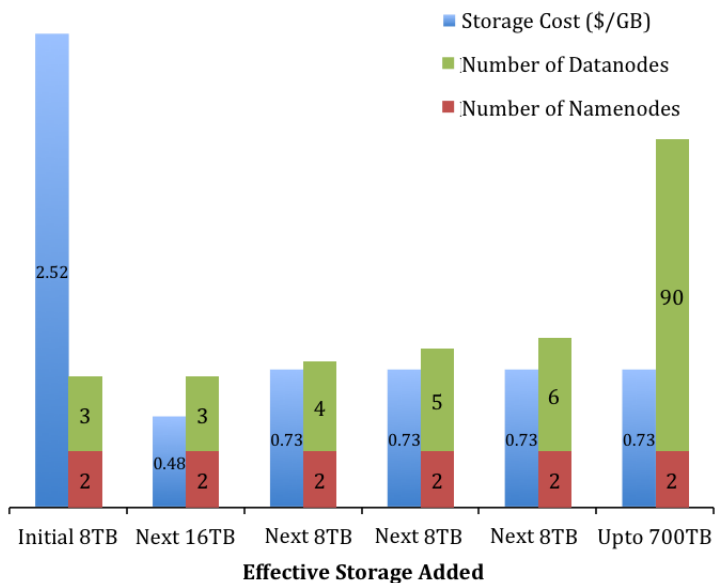


Figura 2.7: Costos por TB Hadoop.

Las instancias en Amazon Web Services se definen por varios modelos de compra: bajo demanda, instancias reservadas e instancias de subasta, con las instancias bajo demanda que fueron las utilizadas, se paga horas de capacidad informática sin necesidad de asumir compromisos a largo plazo ni realizar pagos iniciales[5]. Donde provee la capacidad de aumentar o reducir la capacidad informática en función de las exigencias de la aplicación y pagar únicamente la tarifa por hora especificada de las instancias que se utilizaron.

A continuación, un breve resumen para sistemas operativos RHEL CentOS los costos listados:

	vCPU	ECU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage
General Purpose - Current Generation					
t2.nano	1	Variable	0.5	EBS Only	\$0.0065 per Hour
t2.micro	1	Variable	1	EBS Only	\$0.013 per Hour
t2.small	1	Variable	2	EBS Only	\$0.026 per Hour
t2.medium	2	Variable	4	EBS Only	\$0.052 per Hour
t2.large	2	Variable	8	EBS Only	\$0.104 per Hour
m4.large	2	6.5	8	EBS Only	\$0.12 per Hour
m4.xlarge	4	13	16	EBS Only	\$0.239 per Hour
m4.2xlarge	8	26	32	EBS Only	\$0.479 per Hour
m4.4xlarge	16	53.5	64	EBS Only	\$0.958 per Hour

Figura 2.8: Costos por hora Instancias bajo Demanda

2.12. Putty

PuTTY es un cliente SSH, Telnet, rlogin, y TCP raw con licencia libre. Disponible originalmente sólo para Windows, ahora también está disponible en varias plataformas Unix, y se está desarrollando la versión para Mac OS clásico y Mac OS X. Otra gente ha contribuido con versiones no oficiales para otras plataformas, tales como Symbian para teléfonos móviles. Es software beta escrito y mantenido principalmente por Simon Tatham, open source y licenciado bajo la Licencia MIT.

Este cliente permite configurar la conexión a través de SSH de forma segura a una instancia de linux levantada en AWS, con la definición del usuario, el DNS público y la clave privada de la instancia[28].

2.13. Interoperabilidad

El incremento de sistemas, instituciones u organismos le da un carácter de urgencia a garantizar la capacidad de intercambiar información que ayude tanto a desarrolladores como a integradores de sistemas. La conside-

ración principal de los propietarios de sistemas es que se generaron 'islas', con esto se refieren a la falta de coordinación y manejo ineficiente cuando se comparten información (más allá de los recursos disponibles).

Por esto aparece el concepto interoperabilidad o interoperatividad, es la condición que permite que sistemas o productos diferentes puedan relacionarse entre sí, sin ambigüedad, para coordinar procesos o intercambiar datos. La interoperabilidad se fundamenta en que las informaciones precisas para llevarla a cabo estén disponibles como normas o estándares.

2.13.1. Servicio Web

Un servicio web o webservice es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones por medio de interfaces que se pueden definir, describir y descubrir mediante documentos XML. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos.

La principal razón para usar servicios Web es que se pueden utilizar con HTTP sobre TCP (Transmission Control Protocol) en el puerto 80. Dado que las organizaciones protegen sus redes mediante firewalls -que filtran y bloquean gran parte del tráfico de Internet-, cierran casi todos los puertos TCP salvo el 80, que es, precisamente, el que usan los navegadores. Los servicios Web utilizan este puerto, por la simple razón de que no resultan bloqueados. Es importante señalar que los servicios web se pueden utilizar sobre cualquier protocolo, sin embargo, TCP es el más común.

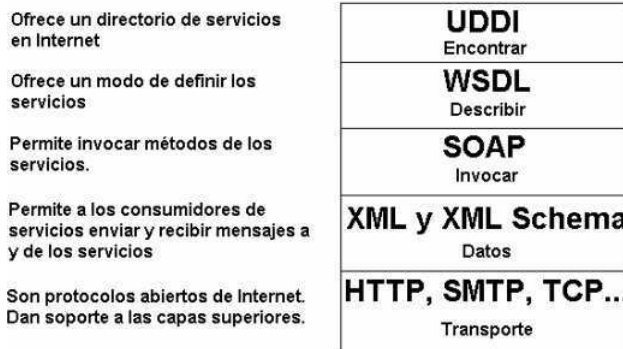


Figura 2.9: Pila de protocolos de los servicios web

Cuando se expone un servicio web, se publica un archivo wsdl en el servidor web, donde se muestran los métodos, parámetros, tipos de retorno, dirección para invocar el servicio.

2.13.1.1. XML

XML es un acrónimo para eXtensible Markup Language (lenguaje de marcado extensible o ampliable). XML es un lenguaje abierto que se ha desarrollado como un subconjunto de SGML, estandarizado por la W3C. Al igual que el HTML, se basa en documentos de texto plano en los que se utilizan etiquetas para delimitar los elementos de un documento. Sin embargo, XML define estas etiquetas en función del tipo de datos que está describiendo y no de la apariencia final que tendrán en pantalla o en la copia impresa, además de permitir definir nuevas etiquetas y ampliar las existentes [32].

Aunque a primera vista, un documento XML puede parecer similar a HTML, hay una diferencia principal: un documento XML contiene exclusivamente datos que se autodefinen. En cambio, un documento HTML contiene datos (cuya definición es, al menos, ambigua), mezclados con elementos de formato. En XML se separa el contenido de la presentación de forma total. Una forma rápida de entender la estructura de un documento

XML es viendo un ejemplo:

```
<?xml version='1.0'?>
<nacimiento>
<fecha>03-02-2006</fecha>
  <perfilSanguineo grupo='AB' factorRH='+' />
<nombre>
  <nombrePropio lugar='primero'>Ivan</nombrePropio>
  <nombrePropio lugar='segundo'>Luis</nombrePropio>
  <apellido parentesco='paterno'>Zamorano</apellido>
  <apellido parentesco='materno'>Albero</apellido>
</nombre>
</nacimiento>
```

Algunas de las características más destacables de XML son las siguientes:

- Las etiquetas y sus atributos pueden ser personalizadas.
- La sintaxis es estricta. La especificación XML determina claramente una serie de reglas que especifican cuándo un documento está “bien formado”.
- Es posible definir familias de documentos con una estructura que se considerará “válida”. Los principales tipos de documentos usados para especificar estructuras son Document Type Definition (DTD) y XML Schema (XSD).

2.13.1.2. SOAP

Simple Object Access Protocol es un protocolo estándar que define el formato de los mensajes. También detalla la forma en que las aplicaciones deben tratar determinados aspectos del mensaje, tales como los elementos del “encabezado”, lo que le permitirá crear aplicaciones en las que un mensaje pasa entre múltiples intermediarios antes de llegar a su destino final.

Básicamente SOAP es un paradigma de mensajería de una dirección sin estado, que puede ser utilizado para formar protocolos más complejos y completos según las necesidades de las aplicaciones que lo implementan

WSDL

Web Services Description Language, resumido como WSDL, es una especificación basada en XML, describe qué mensajes deben ser intercambiados para que la interacción con un Servicio Web sea exitosa. Permite describir la interfaz pública de los servicios web; eso significa que detalla los protocolos y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen y se unen después al protocolo concreto de red y al formato del mensaje.

Un programa cliente se conecta a un servicio web y puede leer el WSDL, determinando así las funciones disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

La estructura del WSDL tiene los siguientes elementos, esta estructura no tiene información para saber la función global del servicio web pero si dispone la información de las funciones definidas en el:

- **types:** Este elemento define los tipos de datos usados en los mensajes. Se utilizan los tipos definidos en la especificación de esquemas XML.
- **message:** Define los elementos del mensaje. Cada mensaje puede estar formado por un conjunto de partes lógicas en serie.
- **portType:** Esta etiqueta define las operaciones permitidas y los mensajes intercambiados en el Servicio.
- **binding:** Define el protocolo de comunicación que fue usado.
- **service:** Indica los puertos y las direcciones de los servicios web.

2.14. Firma electrónica

La firma electrónica es un concepto jurídico, equivalente electrónico al de la firma manuscrita, donde una persona acepta el contenido de un mensaje electrónico a través de cualquier medio electrónico válido. No debe

confundirse con la firma digital.

La función de esta firma además de identificar al firmante de manera inequívoca, es asegurar la integridad del documento firmado, el documento firmado es exactamente el mismo que el original y no ha sufrido alteración o manipulación. También los datos que utiliza el firmante para realizar la firma son únicos y exclusivos y, por tanto, posteriormente, no puede decir que no ha firmado el documento.

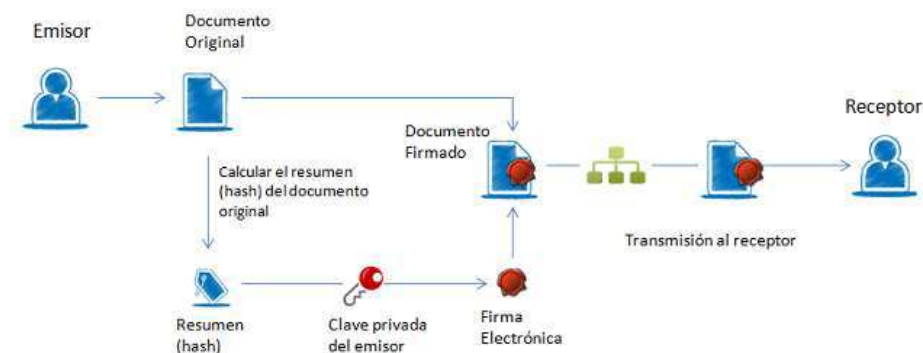


Figura 2.10: Proceso Básico de Firma Electrónica

El proceso básico que se sigue para la firma electrónica es el siguiente[27]:

- El usuario dispone de un documento electrónico (una hoja de cálculo, un pdf, una imagen, incluso un formulario en una página web) y de un certificado que le pertenece y le identifica.
- La aplicación o dispositivo digital utilizados para la firma realiza un resumen del documento. El resumen de un documento de gran tamaño puede llegar a ser tan solo de unas líneas. Este resumen es único y cualquier modificación del documento implica también una modificación del resumen.

- La aplicación utiliza la clave contenida en el certificado para codificar el resumen.
- La aplicación crea otro documento electrónico que contiene ese resumen codificado. Este nuevo documento es la firma electrónica.

El resultado de todo este proceso es un documento electrónico obtenido a partir del documento original y de las claves del firmante. La firma electrónica, por tanto, es el mismo documento electrónico resultante.

2.14.1. Xolidosing

Es una aplicación que permite de forma sencilla e intuitiva la firma electrónica de todos los documentos deseados por el usuario. También permite aplicar sello de tiempo digital a sus documentos, tanto independiente como incrustado en las firmas digitales, usando para ello un servidor compatible RFC 3161 determinado por el usuario.



Figura 2.11: Logo Xolidosing

Durante el proceso, la aplicación tiene en cuenta las medidas de control y seguridad apropiadas, como chequeos de revocación de los certificados y comprobación de integridad (verificar hash).

Esta destinada para ser usado por cualquier tipo de profesional o ciudadano facilitando el acercamiento de las nuevas tecnologías de firma electrónica, verificación y sellado de tiempo. La ventaja de esta aplicación es que reduce el tiempo en sus procedimientos documentales y los costos de envíos (sellos, sobres, papel) empleando archivos electrónicos y manteniendo la seguridad en los tramites.

2.14.2. OpenSSL

Es una librería de propósito general es de código abierto y hecha en lenguaje C. Esta biblioteca implementa operaciones criptogramas simétricas, encriptacion de claves publicas, firma electrónica, funciones hash, etc.

Estas operaciones ayudan a implementar el *Secure Sockets Layer* (SSL), así como otros protocolos relacionados con la seguridad, como el Transport Layer Security (TLS). OpenSSL también permite crear certificados digitales que pueden aplicarse a un servidor

2.14.3. Certificado electrónico

Es un fichero informático generado por una entidad de servicios de certificación que asocia unos datos de identidad a una persona física, organismo o empresa confirmando de esta manera su identidad digital.

El Certificado electrónico es el único medio que permite garantizar técnica y legalmente la identidad de una persona en Internet. Se trata de un requisito indispensable para que las instituciones puedan ofrecer servicios seguros a través de Internet. Además permite la firma electrónica de documentos. El receptor de un documento firmado puede tener la seguridad de que éste es el original y no ha sido manipulado y el autor de la firma electrónica no podrá negar la autoría de esta firma[34].

Capítulo 3

Método de desarrollo

Las metodologías de desarrollo de software son marcos o modelos de trabajos que se utilizan para construir, planificar y controlar el proceso de desarrollo de sistemas.

Hoy en día existen infinidad de metodologías para desarrollar software. Entre ellas encontramos las Metodologías Tradicionales, las Metodologías Iterativas/Evolutivas, las Metodologías basadas en Tecnología Web, y las Metodologías Ágiles.

3.1. Ad Hoc

Usamos esta metodología porque define las actividades como iteraciones, realizamos una integración de varias herramientas y eso requirió un tiempo de estudio, configuración y pruebas de cada una. Cada iteración es una actividad que no tiene orden lógico con respecto a las demás, como no hay conexión entre las actividades, entonces, es posible continuar con cualquier otra actividad, detener una actividad y continuar con otra. El orden de ejecución se establece durante la ejecución del proceso según las condiciones de trabajo y el tiempo establecido para finalizar todas las actividades.

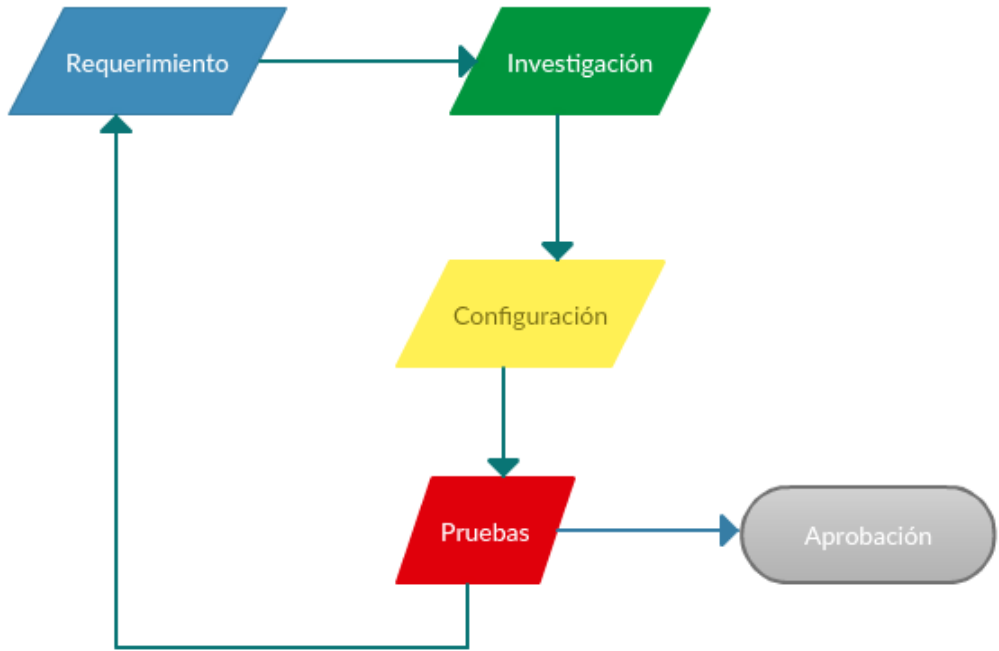


Figura 3.1: Pasos para la integración de herramientas

Usamos la metodología de la siguiente forma:

- **Requerimiento:** Nuestros tutores nos plantearon un problema a resolver y nosotros dividimos el problema definiendo las funcionalidades a implementar.
- **Investigación:** En esta fase estudiamos las herramientas que podríamos utilizar, estudiamos las funcionalidades que ofrecen y evaluamos los requerimientos mínimos de hardware o software.
- **Configuración:** Ya elegimos las herramientas a usar, entonces debemos descargar dependencias (si es necesario), instalar la herramienta

en un ambiente previamente configurado y modificar archivos de configuración.

- **Pruebas:** Nosotros realizamos pruebas de cada herramienta en un ambiente local, luego de eso las instalamos en el ambiente de producción. En esta etapa es posible que surjan sugerencias, por ejemplo, modificar la interfaz de alguna herramienta.
- **Aprobación :** Si las pruebas fueron satisfactorias entonces documentamos los pasos para la instalación, configuración y uso de la herramienta seleccionada. Cuando esta etapa finaliza, pasamos a una nueva iteración en el desarrollo de la solución.

Capítulo 4

Desarrollo de la solución

4.1. Arquitectura de la solución

Fedora Commons es flexible en cuanto a la configuración de almacenamiento de bajo nivel. Los objetos y los datastream se pueden configurar para utilizar diferentes plataformas de almacenamiento. Esta flexibilidad se ve reflejada a partir de la versión 3.2 de Fedora Commons y ha sido la interfaz de almacenamiento por defecto desde la versión 3.4. En la versión utilizada (3.6.2), akubra está optimizado para permitir a Fedora conectarse a cualquier plataforma de almacenamiento que tenga una implementación compatible. Por ejemplo, existen plataformas populares como iRods o Dell DX que incluyen interfaz compatible con Akubra [1]. Esta configuración permite almacenar ambos los objetos y datastreams en el FileSystem de un Cluster multinodos de Hadoop.

Ahora, en la arquitectura de Fedora, Akubra se encuentra como una capa intermitente de abstracción entre el módulo de gestión y su sistema de almacenamiento (que en la configuración estándar de Fedora Commons es un sistema de archivos POSIX). Este módulo gestiona peticiones a través de la interfaz de akubra para obtener contenido binario o los objetos almacenados en Fedora.

Al tener una implementación que permita conectar el API de Akubra al HDFS de Hadoop a través de Map Reduce, se hace factible el procesa-

miento de datos con el fin de realizar tareas de cálculo intensivo de manera distribuida. El desarrollo consistió en clases de Java apoyadas sobre el API de Hadoop en Java para el almacenamiento de los objetos digitales y datstreams de Fedora en el Hadoop File System. A través de métodos, siguiendo un estándar CRUD de lectura y escritura para, y desde archivos, iterando sobre directorios para descubrir contenido, así como la posibilidad de mapear los índices de Fedora a rutas de HDFS.

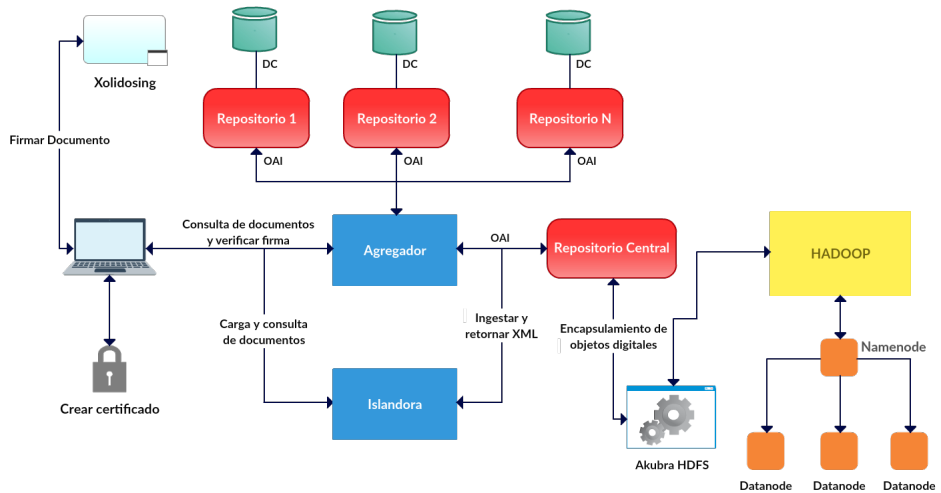


Figura 4.1: Arquitectura de la solución

Típicamente, un clúster Hadoop consiste en un único Namenode y múltiples DataNodes. Hadoop puede manejar fallos en los DataNode de muy buena manera, pero si falla el NameNode todo el clúster se cae. El NameNode gestiona la metadata de HDFS, mientras que los DataNodes almacenan todo el contenido de los objetos de Fedora almacenados. Las solicitudes de un cliente a HDFS siempre son procesados primero por el NameNode, y este se encarga de redirigir los clientes a la información almacenada en un DataNode. Hadoop utiliza la redundancia para lograr la tolerancia a fallos, almacenando copias de los mismos datos en diferentes

nodos de modo que cuando uno falla los datos todavía pueden ser recuperados desde otro Datanode disponible. Este numero de copias que genera Hadoop es configurable. Hadoop utiliza un marco de configuración basado en XML, donde se definen todos los nodos de Hadoop. Estos nodos necesitan estar configurados a través de archivos XML de manera correcta para el buen funcionamiento del cluster. Utilizamos Cloudera Enterprise versión 5 [7], una herramienta fácil de usar para implementar y administrar clústers Hadoop. Cloudera proporciona una plataforma escalable, flexible e integrada que hace que sea fácil de gestionar el rápido aumento de los volúmenes y variedades de los datos.

Generalmente, en un objeto digital, el datastream compone mayor parte del almacenamiento, siendo la metadata una pequeña porción de un objeto, y sabemos que HDFS esta optimizado para el manejo de objetos de gran tamaño. Sin embargo, para archivos pequeños, la eficiencia en rendimiento puede ser afectada cuando existe sobrecarga de red. Pero este problema se relaciona directamente con la arquitectura del framework y HDFS. Los archivos en HDFS son organizados en bloques de un tamaño específico de 128MB, y los archivos de mayor tamaño a estos bloques se distribuyen en partes del tamaño del bloque. El namenode en un Sistema de archivos HDFS es la maquina privilegiada encargada de el manejo y distribución de acceso a los archivos en un ambiente distribuido, entonces cuando se tiene cada archivo, directorio y bloque cargado en Namenode ocupan espacio en memoria, lo cual pone un limite en cuanto a la cantidad de archivos que se puede tener simultáneamente en memoria y hace a HDFS dependiente de la cantidad de memoria disponible.

4.2. Análisis y diseño

Para realizar una prueba de concepto, se configuró una instalación de Hadoop sobre un clúster de computadores levantados bajo demanda con un servicio de IaaS (Infrastructure as a Service) llamado Amazon Elastic Cloud Service (EC2) [3] que proporciona capacidad de computación escalable en la nube de Amazon Web Services. Donde se configuro la cantidad de 4 instancias bajo la distribucion de Linux Centos 7.2 RHEL (AWS) [4].

Una instancia de Amazon EC2 es un servidor virtual en la nube de AWS, donde existe la facilidad de correr y configurar aplicaciones en nuestro caso con Linux.

Para configurar cada instancia en Amazon Web Services, se definió desde consola de Amazon Web Services la Amazon Machine Image (AMI) donde esta despliegan las distintas configuraciones básicas estándar de sistemas operativos funcionales. Configurando la perteneciente al AMI Centos RHEL 7.2.

La ventaja del uso de Amazon Web Services como servicio en la nube y su sistema Elastic Block Store es la facilidad que provee a un sistema de escalabilidad, ya que podemos incrementar el espacio de almacenamiento de un volumen EBS o cambiar el tipo de instancia existente a una que posea mas capacidad de computo y memoria RAM sin perder la información de nuestra instancia, esto da una ventaja enorme en ahorro de tiempo y costos de configuración, esto gracias a los llamados Snapshot de una instancia podemos hacer backup de la información sin perder nada en esta.

Cabe acotar, que Amazon Web Services provee de un sistema de precios un poco complejo, y el costo es uno de los factores decisivos para adoptar cualquier tecnología. Y mas, tomando en cuenta la dificultad para conseguir divisas, contamos con la ventaja de aprovechar un programa llamado Amazon Educate, el cual provee a instituciones y estudiantes de pregrado y postgrado de herramientas de e-learning así como mayores oportunidades en el uso de la infraestructura de AWS. Gracias al programa, se pudo configurar instancias que permitan hacer pruebas considerables a nivel de hardware, ya que una de las debilidades de hadoop es el ejecutarse en hardware de bajo costo, aunque esta diseñado para manejar fallos por defecto y no depende de que el hardware provea de esta bondad, con un clúster bien planificado se puede lograr un balance entre costo y rendimiento.

Instance: **i-82c01316 (HadoopNewNameNode_FC)** Public DNS: **ec2-52-33-177-191.us-west-2.compute.amazonaws.com**

Description | Status Checks | Monitoring | Tags

Instance ID	i-82c01316	Public DNS	ec2-52-33-177-191.us-west-2.compute.amazonaws.com
Instance state	running	Public IP	52.33.177.191
Instance type	m4.large	Elastic IPs	

Figura 4.2: Descripción instancia EC2

4.3. Configuración del ambiente

4.3.1. Instalación del Clúster Hadoop

Para el desarrollo de la solución se instaló un clúster multinodo bajo la distribución Cloudera versión 5, utilizando la distribución Cloudera mediante 4 instancias conectadas por DNS donde cada nodo del clúster es representado por cada una de las instancias de AWS. Y para el tipo de instancia se configuro de la siguiente forma:

Cuadro 4.1: Instancia Namenode

Tipo de Instancia	m4.large
CPU	Intel Xeon® E5-2676 v3 (Haswell) de 2,4 GHz
MEMORIA	8GB
HDD	50GB

Cuadro 4.2: Instancia Datanode

Tipo de Instancia	T2.medium
CPU	Intel Xeon De alta frecuencia con Turbo hasta 3,3 GHz
MEMORIA	4GB
HDD	40GB

Al escoger el tipo de instancia, podemos seleccionar la configuración de hardware disponible para cada una de las instancias en la nube, existen muchos tipos pero para nuestro enfoque académico se escogió de tipo m4.large el namenode y de tipo t2.medium en los datanode.

La instancia de tipo m4.large que fue utilizada para el namenode esta compuesta por las siguientes características:

- Procesadores Intel Xeon® E5-2676 v3 (Haswell) de 2,4 GHz
- Optimizados para EBS [13] de manera predeterminada sin costos adicionales
- Soporte para redes mejoradas
- Equilibrio entre recursos de informática, memoria y red

La instancia de tipo t2.medium que fue utilizada para el namenode esta compuesta por las siguientes características:

- Procesadores Intel Xeon de alta frecuencia con Turbo hasta 3,3 GHz
- CPU en ráfagas, que se rige por créditos de CPU y desempeño de base constante
- Equilibrio entre recursos de informática, memoria y red

4.3.1.1. Configuración de nodos del cluster

Para la creación del cluster se implemento un modelo maestro-esclavo, donde contamos con un nodo maestro y tres nodos esclavos, y para establecer la comunicación hacia el servidor se realizó a través del cliente PUTTY, donde requiere de DNS o Ip Pública.

Cuadro 4.3: Dirección IPv4 y DNS públicos

	DNS Publico	Ip Publico
Namenode	ec2-52-42-145-135.us-west-2.compute.amazonaws.com	52.42.145.135
Datanode1	ec2-54-68-238-96.us-west-2.compute.amazonaws.com	54.68.238.96
Datanode2	ec2-52-87-227-189.compute-1.amazonaws.com	52.87.227.189
Datanode3	ec2-52-40-159-86.us-west-2.compute.amazonaws.com	54.191.152.190

Configuración de Hosts

Es necesario contener información persistente sobre los nombres de los equipos y direcciones IP de los nodos para poder desplegar el agente correctamente al instalar Cloudera, por lo tanto hay que tener las DNS públicas disponibles de cada node en el fichero host de cada máquina de la siguiente manera:

```
$ sudo vi /etc/hosts
```

```
127.0.0.1    localhost localhost.localdomain localhost4
::1         localhost localhost.localdomain localhost6 localhost6.
52.42.145.135 ec2-52-42-145-135.us-west-2.compute.amazonaws.com
54.68.238.96 ec2-54-68-238-96.us-west-2.compute.amazonaws.com
52.87.227.189 ec2-52-87-227-189.compute-1.amazonaws.com
52.40.159.86 ec2-52-40-159-86.us-west-2.compute.amazonaws.com
```

Para los protocolos de seguridad en Amazon Web services, se creo un grupo con las opciones predeterminadas y se habilito una regla para SSH en el puerto para 22. Esto permite hacer ping, SSH y otros comandos similares entre los servidores y de cualquier otra máquina en Internet. También se requiere estos protocolos y puertos para permitir la comunicación entre los nodos del clúster.

Asignando el hostname

Es necesario identificar cada máquina dentro de una red, por lo tanto se ejecuta el siguiente comando por cada nodo, donde dns_publico es el DNS

listado arriba asignado por Amazon Web Services a la instancia.

```
$ sudo hostname <dns_publico>
```

Desactivación de SELinux

SELinux actúa como un módulo de seguridad a nivel de kernel que provee un mecanismo que soporta el acceso a políticas de seguridad. Este debe ser desactivado para lograr una comunicación óptima.

```
vi /etc/sysconfig/selinux
SELINUX = disabled
```

Configuración del SSH

SSH (Secure Shell) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos. SSH soporta múltiples formas para autenticar a los usuarios. Para conectarse a cada nodo del cluster Hadoop Cloudera debe existir acceso SSH para así realizar la instalación y despliegue de servicios.

Modificación del archivo `sshd_config` en cada uno de los nodos.

```
sudo vi /etc/ssh/sshd_config
port 22
PermitRootLogin yes
RSAAuthentication yes
PubKeyAuthentication yes
service sshd restart
```

Para lograr la comunicación en un cluster multinodo de forma adecuada, es necesario proveer permisos a cada una de las instancias. La primera era obtener la clave privada de Amazon EC2 la cual usa criptografía de claves públicas para encriptar y desencriptar la información de autenticación. La criptografía de clave pública se usa para encriptar la información, como password, y luego el cliente usa la clave privada de Amazon para desencriptar esta data. También se le conoce como key pair. La cual se agrega

al nodo maestro:

```
>$ eval 'ssh-agent'  
>$ ssh-add DaveAws-rafaproHadoop.pem  
Identity added: DaveAWS-rafaproHadoop.pem (DaveAWS-rafaproHadoop.pem)
```

Finalmente se prueban las conexiones ssh, donde el nodo maestro puede acceder a los nodos esclavos sin contraseña:

```
$ssh slave1  
>logout
```

```
$ssh slave2  
>logout
```

```
$ssh slave3  
>logout
```

4.3.1.2. Instalación Cloudera Hadoop

Ya teniendo configurado las opciones de comunicación y las configuraciones anteriores se procede a la instalación de la distribución Cloudera en cada uno de los nodos, se puede configurar de distintas maneras, pero se procedió a hacerla de forma manual a través de los paquetes.

Descargamos el archivo rpm cloudera y instalamos el repositorio rpm.

```
$sudo yum nogpgcheck localinstall clouderacdh50.x86_64.rpm
```

Si la instancia de la instalación es un namenode:

```
$sudo yum clean all; sudo yum install hadoop-0.20 -mapreduce-jobtracker
```

Si la instancia de la instalación es un datanode:

```
$sudo yum clean all; sudo yum install hadoop-0.20 -mapreduce-tasktracker  
hadoop-hdfs-datanode
```

Creamos la configuración personalizada y además le damos prioridad para que Hadoop reconozca primero el archivo `conf.my_cluster`

```
sudo cp -r /etc/hadoop/conf.empty /etc/hadoop/conf.my_cluster
$ sudo alternatives --install /etc/hadoop/conf hadoop-conf
/etc/hadoop/conf.my_cluster 50}
$ sudo alternatives --set hadoop-conf /etc/hadoop/conf.my_cluster}
```

Además, para el correcto funcionamiento de Hadoop es necesario modificar distintos archivos de configuración en los que se identifica el nodo maestro, nodos esclavos, permisos y alta disponibilidad.

El archivo `core-site.xml`, este es uno de los más importantes porque aquí se define la ruta en HDFS del nodo maestro

```
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://ec2-52-42-145-135.us-west-
  2.compute.amazonaws.com:8020</value>
</property>
```

En el archivo `Hdfs-site.xml`, se define el balanceo de nodos, las carpetas locales donde se almacena los objetos de HDFS así como permisos:

```
<property>
  <name>dfs.permissions.superusergroup</name>
  <value>hadoop</value>
</property>
```

Además en el mismo archivo definimos las carpetas locales donde se almacena HDFS, para el namenode:

```
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:///data/1/dfs/nn,file:///nfsmount/dfs/nn</value>
</property>
```

En las instancias datanode, se definen las carpetas de almacenamiento de la siguiente manera:

```
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:///data/1/dfs/dn,file:///data/2/dfs/dn,
  file:///data/3/dfs/dn,file:///data/4/dfs/dn</value>
</property>
```

Además debemos crear las carpetas localmente en cada uno de los nodos y darle permisos al usuario hdfs como propietario. En el caso de la instancia Namenode:

```
sudo mkdir -p /data/1/dfs/nn /nfsmount/dfs/nn
sudo chown -R hdfs:hdfs /data/1/dfs/nn /nfsmount/dfs/nn}
```

En el caso de las instancias Datanode:

```
sudo mkdir -p /data/1/dfs/dn /data/2/dfs/dn /data/3/dfs/dn /data/4/dfs/
sudo chown -R hdfs:hdfs /data/1/dfs/dn /data/2/dfs/dn /data/3/dfs/dn
/data/4/dfs/dn
```

Además por defecto Hadoop no reconoce la versión de JAVA instalada en el sistema operativo, y debemos definirla en el archivo YARN-ENV.sh

```
export JAVA_HOME= /usr/bin/java/jdk1.8.0_91
```

Al tener toda la configuración completa en nuestro cluster, pasamos a copiar la carpeta de configuración desde el nodo maestro y todos los esclavos a través de línea de comando.

```
scp -r /etc/hadoop/conf.my\_cluster
ec2-user@<DNS_publico_instancia>:/etc/hadoop/conf.my\_cluster
```

Y le damos prioridad a nuestra carpeta nueva de configuración:

```
\$ sudo alternatives --verbose --install /etc/hadoop/conf hadoop-conf /etc
\$ sudo alternatives --set hadoop-conf /etc/hadoop/conf.my\_cluster
```

Luego damos formato al cluster con la siguiente línea de comando

```
$ sudo -u hdfs hdfs namenode -format
```

En este punto logramos configurar un cluster totalmente operable y distribuido con un maestro y 3 esclavos, para iniciarlo debemos ejecutar la siguiente línea en cada uno de los nodos.

```
$ for x in `cd /etc/init.d ; ls hadoop-hdfs-*` ;
do sudo service \$x start ; done
```

NameNode 'ec2-52-42-117-41.us-west-2.compute.amazonaws.com:8020' (active)

Started:	Sat Oct 08 18:15:17 EDT 2016
Version:	2.6.0-cdh5.8.0_042da8b668a212c843bcfb3594519dd26e816e79
Compiled:	2016-07-12T23:03Z by jenkins from Unknown
Cluster ID:	CID-1bc5da86-6384-496b-9c41-63bfb137099c
Block Pool ID:	BP-671999816-172.31.28.179-1475964895585

[Browse the filesystem](#)
[NameNode Logs](#)

Cluster Summary

Security is OFF

Rolling upgrades in progress. There are 2 versions of datanodes currently live: 2.6.0-cdh5.8.0(2), 2.6.0-cdh5.8.2(1)
 39 files and directories, 22 blocks = 61 total.

Heap Memory used 37.20 MB is 9% of Committed Heap Memory 388.50 MB. Max Heap Memory is 889 MB.

Non Heap Memory used 43.04 MB is 97% of Committed Non Heap Memory 44.03 MB. Max Non Heap Memory is -1 B.

Configured Capacity		319.86 GB			
DFS Used		37.49 MB			
Non DFS Used		27.68 GB			
DFS Remaining		292.14 GB			
DFS Used%		0.01%			
DFS Remaining%		91.34%			
Block Pool Used		0 B			
Block Pool Used%		0.00%			
DataNodes usages		Min %	Median %	Max %	stdev %
		0.01%	0.01%	0.01%	0.00%
Live Nodes		3 (Decommissioned: 0)			
Dead Nodes		0 (Decommissioned: 0)			
Decommissioning Nodes		0			

Figura 4.3: Interfaz de Cluster Hadoop y sus nodos

Facilidad de Acceso

El fácil acceso al almacenamiento es un requerimiento importante para cualquier sistema de archivos. Las colecciones digitales o usuarios pueden

necesitar acceder directamente al HDFS para realizar operaciones de manejo de data del día a día. Y las funcionalidades para montar nativamente el HDFS a usuarios en un cliente es limitado. Pero, existen servicios que permiten instalarse del lado del servidor y proveen de mayor facilidad de acceso al sistema de archivos distribuido.

Entre las opciones disponibles existen:

- **Hue File Browser:** Hue [18], por las siglas Hadoop User Experience, es una interfaz gráfica de usuario open source para Hadoop desarrollado por Cloudera. Este es un explorador de archivos Hue y expone una interfaz web accesible a HDFS. Es rico en funcionalidad y bastante intuitivo. La interfaz se puede utilizar para cargar, descargar o eliminar archivos y carpetas en HDFS.
- **FUSE-HDFS:** Filesystem in Userspace [16], permite a los sistemas de archivos externos tener un mount point en una maquina Unix. Proporciona a los usuarios la comodidad de acceder a sistemas de archivos remotos de manera similar a los sistemas de archivos locales. El módulo FUSE-HDFS se puede utilizar para montar HDFS a las máquinas de Linux. Aunque presenta una serie de desventajas, y es preferible no usar a nivel de producción. Ya que presenta problemas de rendimiento generando cuellos de botella y no tiene soporte para montar en un sistema operativo Windows.
- **HttpFS:** Es un servicio de proxy que permite a los clientes acceder a HDFS través de un API HTTP Rest. Las operaciones de transferencia de archivos en HttpFS se pueden realizar utilizando una herramienta de línea de comandos como CURL [10].

En esta investigación, se hicieron pruebas para acceder y subir archivos directamente al HDFS con el servicio FUSE-HDFS, el cual a partir de la versión Cloudera 4 se da la opción de instalar a través del siguiente comando:

```
$sudo yum install hadoop-hdfs-fuse
```

Para las pruebas se hizo el mount point en el folder ubicado en /home/ec2-user/mountfuse

```
$sudo hadoop-fuse-dfs dfs://ec2-52-43-160-6.us-west-2.compute.amazonaws.com:8020 /home/ec2-user/mountfuse
```

Por ultimo, al tener nuestro mount point de HDFS se permite utilizar el clúster HDFS como si se tratara de un sistema de archivos tradicional en Linux. Habilitando opciones de escritura y lectura de objetos digitales de Fedora Commons en el mount point de HDFS. Pero esto conlleva ciertas desventajas, primero, el servicio de almacenamiento de HDFS sobre FUSE solo soporta I/O secuencial, lo que significa que un cliente puede fallar en algún punto en tiempo. Segundo, Fuse no provee de consistencia mínima requerida para una aplicación (Un cliente no puede leer siempre lo que acaba de escribir). Tercero, debe instalarse un componente cliente en cada instancia, y solo es soportado por sistemas Linux.

Para desmontar FUSE HDFS se realiza de alguna de las siguiente maneras:

```
$ fusermount -u /home/ec2-user/mountfuse
$ sudo umount -l /home/ec2-user/mountfuse
```

4.3.2. Instalación y Configuración del Repositorio Digital Fedora Commons

El repositorio Digital de Fedora Commons fue configurado en la instancia donde corre el Namenode Cliente, para su instalación en la versión 3.6.2, se utilizaron las siguientes dependencias:

- Java SE Development Kit (JDK)
- Base de datos MySQL, El instalador incluye una instancia de Derby SQL Database, pero no es recomendado su uso a menos que sea solamente para evaluación y desarrollo.
- Servidor de aplicaciones Tomcat 6.0

- Maven

Para instalar Fedora y ejecutar comandos sobre el repositorio, definimos las siguientes variables de entorno:

- **JAVA_HOME**: Define la ruta de la versión de Java utilizada por los servicios que la requieran, en nuestro caso es `/usr/java/jdk1.8.0_91`.
- **FEDORA_HOME**: Es necesaria para ejecutar ordenes desde línea de comando. Opcional para ejecutar el instalador y es ignorada cuando se ejecuta el servidor Fedora.
- **PATH**: En esta variable incluimos los directorios bin de Java y Fedora. Nosotros definimos en UNIX las variables de la siguiente forma, `$FEDORA_HOME/server/bin`, `$FEDORA_HOME/client/bin` y `$JAVA_HOME/bin`.

Con las variables de entorno definidas, instalamos el repositorio al ubicarnos en el directorio donde está el instalador que descargamos previamente. Antes de que ejecutemos el instalador, verificamos que el usuario tenga permisos para escribir en los directorios donde se va a realizar la instalación. Iniciamos el proceso de la siguiente forma:

```
java -jar fcrepo-installer-3.6.2.jar
```

Al finalizar la instalación ejecutamos este comando para iniciar Fedora Commons:


```
$FEDORA_HOME/tomcat/bin/startup.sh
```

Finalizada la instalación, abrimos un navegador web e ingresamos a la ruta

```
http://ec2-52-42-145-135.us-west-2.compute.amazonaws.com:8080/fedora\describe\end
```

Donde se muestra la siguiente interfaz, de esta forma finalizamos la instalación de Fedora Commons a este punto con el sistema de almacenamiento

de bajo nivel local que incorpora por defecto.



Fedora

Repository Information View

Repository Name: Fedora Repository

Base URL:	http://localhost:8080/fedora
Version:	3.8.1
PID Namespace:	changeme
PID Delimiter:	:
Sample PID:	changeme:100
Retain PID Namespace:	*
OAI Namespace:	example.org
OAI Delimiter:	:
Sample OAI Identifier:	oai:example.org:changeme:100
Sample Search URL:	http://localhost:8080/fedora/objects
Sample Access URL:	http://localhost:8080/fedora/objects/demo:5
Sample OAI URL:	http://localhost:8080/fedora/oai?verb=Identify
Admin Email:	bob@example.org
Admin Email:	sally@example.org

Figura 4.4: Información del repositorio Fedora

4.3.3. Configuración Akubra HDFS

Para el almacenamiento de los documentos en el HDFS usamos una implementación de Fedora que usa el Hadoop Filesystem llamada Akubra HDFS bajo licencia Apache 2.0. Es una capa de abstracción del sistema de archivos que se utiliza por Fedora.

Primero debemos clonar el repositorio, luego que lo tenemos local, debemos construir el proyecto con Maven

```
$ git clone https://github.com/rafaalb/fedora-hadoop
$ cd ./fedora-hadoop
$ mvn install
```

Esto nos genera un directorio llamado `target`, donde se encuentra la dependencia `akubra-hdfs-0.0.1-SNAPSHOT.jar` que se encarga de la comunicación entre el API de `akubra` y el API de `Hadoop`.

Para que el entorno de `Apache Tomcat` donde esta corriendo el Repositorio `Fedora Commons` reconozca las librerías de `Hadoop`, es necesario copiar las dependencias de `Hadoop` al siguiente directorio de `Fedora`:

```
$FEDORA_HOME/tomcat/webapps/fedora/WEB-INF/lib
```

- `akubra-hdfs-0.0.1-SNAPSHOT.jar`
- `hadoop-core-1.0.3.jar` ubicado en `$HADOOP_HOME/`
- `hadoop-client-1.0.3.jar` ubicado en `$HADOOP_HOME/`
- `commons-configuration-1.6.jar` ubicado en `$HADOOP_HOME/lib/`
- `commons-lang-2.4.jar` ubicado en `$HADOOP_HOME/lib/`
- `htrace-core4-4.0.1-incubating.jar`
- `hadoop-hdfs-2.6.0-cdh5.8.0.jar`
- `hadoop-auth-2.6.0-cdh5.8.0.jar`

Para hacer la configuración, editamos el archivo:

```
$FEDORA_HOME/server/config/spring/akubra-llstore.xml
```

Donde editamos las bean de maven de forma que se reciba el parámetro de ruta del `HDFS` configurando los parámetros `fsObjectStore` y `fsDataStreamStore` para que usen la clase:

```
de.fiz.akubra.hdfs.HDFSBlobStore
```

De la misma manera, configurar los parámetros `fsObjectStoreMapper` y `fsDatastreamStoreMapper` para que usen la clase:

```
de.fiz.akubra.hdfs.HDFSIdMapper
```

Al editar nuestro archivo de configuración queda de la siguiente manera:

```
<bean name="fsObjectStore"
class="de.fiz.akubra.hdfs.HDFSBlobStore" singleton="true">
  <constructor-arg value="hdfs://localhost:8020/fedora/objects"/>
</bean>

<bean name="fsDatastreamStore" class="de.fiz.akubra.hdfs.HDFSBlobStore"
singleton="true">
  <constructor-arg value="hdfs://localhost:8020/fedora/datastreams"/>
</bean>

<bean name="fsObjectStoreMapper" class="de.fiz.akubra.hdfs.HDFSIdMapper"
singleton="true">
  <constructor-arg ref="fsObjectStore"/>
</bean>

<bean name="fsDatastreamStoreMapper" class="de.fiz.akubra.hdfs
.HDFSIdMapper" singleton="true">
  <constructor-arg ref="fsDatastreamStore"/>
</bean>
```

Culminada la configuración del archivo `akubra-llstore.xml`, procedemos a ubicarnos en la carpeta `$/CATALINA_HOME/bin/` y ejecutamos el script `startup.sh`, a este punto cambiamos la configuración de almacenamiento de objetos local de Fedora a sistema de archivos distribuido de Hadoop.

Contents of directory /fedora

Goto : /fedora

[Go to parent directory](#)

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
datastreamsfedora	dir				2016-10-09 09:09	rwxr-xr-x	root	supergroup
objectsfedora	dir				2016-10-09 09:15	rwxr-xr-x	root	supergroup

[Go back to DFS home](#)

Local logs

[Log directory](#)

[Hadoop](#), 2016.

Figura 4.5: Objetos digitales almacenados en el cluster

4.3.4. Integración con CMS

Es necesario usar un manejador de contenido para visualizar y administrar los objetos digitales del repositorio, elegimos usar Drupal porque entre los módulos que dispone esta Islandora, y provee integración con el repositorio de Fedora Commons. Donde se requirió de una configuración de:

- Servidor web Apache.
- Base de datos MySQL 5.0.15
- PHP 5.2.5

Desde línea de comandos (Linux) se indica donde se guardarán los archivos comprimidos de drupal.

```
$ cd /opt/downloads
$ wget http://ftp.drupal.org/files/projects/drupal-x.5.tar.gz
$ tar -xzvf drupal-x.5.tar.gz
```

Creamos el directorio destino donde se va a instalar Drupal. Un lugar recomendado para hacer esto es:

```
$ mkdir /var/www/drupal
```

Fue necesario mover el contenido del directorio drupal-x.x que fue descargado anteriormente en el directorio que acabamos de crear. Nos aseguramos que el archivo '.htaccess', un archivo oculto, sea transferido con éxito al directorio destino.

```
$ mv -v /opt/downloads/drupal-x.x/* /var/www/drupal
```

Con esto ya fue instalado Drupal y debemos realizar la respectiva configuración, es necesario hacer una copia del archivo default.settings.php en el directorio sites/default y nombrar la copia como settings.php, además de otorgarle privilegios de escritura al directorio sites/default y al archivo que acabamos de copiar:

```
$ cp sites/default/default.settings.php sites/default/settings.php
$ chmod a+w sites/default/settings.php
$ chmod a+w sites/default
```

Configuramos manualmente la base de datos MySQL para Drupal.

```
$ mysql -u root -p
mysql> create database drupal2;
mysql> GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER,
CREATE TEMPORARY TABLES ON
drupal.* TO 'druser'@'localhost' IDENTIFIED BY 'adminFedora2';
mysql> flush privileges;
mysql> exit
```

Como el enfoque es subir archivos de gran tamaño a nuestro repositorio, se modificaron los archivos de base configuración de php para que nos permita:

```
$sudo vi /etc/php5/apache2/php.ini
upload_max_filesize = 2048M
post_max_size = 2048M
memory_limit = 256M
```



Figura 4.6: Interfaz Drupal

4.3.5. Cambios en la interfaz OHS

El OHS es la interfaz donde realizamos las búsquedas sobre los repositorios que hayan sido agregados anteriormente. Los requerimientos para usarlo son:

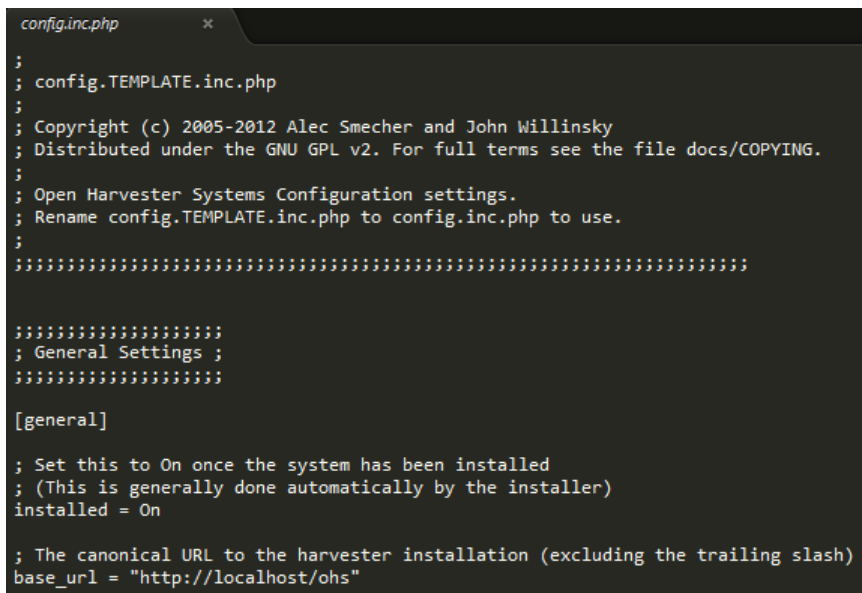
- PHP 4.2.x o superior.
- MySQL 3.23.23 o superior. Otra opción es PostgreSQL 7.1 o superior.
- Apache 1.3.2x, 2.0.4x o superior.
- No hay restricciones de sistema operativo. Ha sido probado en Linux, BSD, Solaris, Mac OS X, Windows.

Lo descargamos desde esta dirección

<https://pkp.sfu.ca/harvester2/download/ohs-2.3.2.tar.gz>.

Descomprimos el archivo y lo movimos al directorio de nuestro servidor web, en este caso es `/var/www/html/`, luego quitamos el número de la versión en la carpeta resultante.

Para iniciar el sistema tuvimos que iniciar el servidor web, y en el archivo `config.inc.php` indicamos la ruta de acceso, entonces, desde el navegador ingresamos a esa dirección, por ejemplo, `http://localhost/ohs/index.php`.



```
config.inc.php x
;
; config.TEMPLATE.inc.php
;
; Copyright (c) 2005-2012 Alec Smecher and John Willinsky
; Distributed under the GNU GPL v2. For full terms see the file docs/COPYING.
;
; Open Harvester Systems Configuration settings.
; Rename config.TEMPLATE.inc.php to config.inc.php to use.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; General Settings ;
;
[general]

; Set this to On once the system has been installed
; (This is generally done automatically by the installer)
installed = On

; The canonical URL to the harvester installation (excluding the trailing slash)
base_url = "http://localhost/ohs"
```

Figura 4.7: Configuración de ruta al sistema

Para iniciar al OHS debemos ejecutar el archivo `httpd` ubicado en el servidor donde fue instalado.



Figura 4.8: Intefaz Open Harvester System

4.3.5.1. Listado de Objetos Digitales

Inicialmente cuando hacemos una consulta, solo se muestra en la interfaz con el título del objeto y su identificador. Es necesario que esta vista brinde la opción de tener una vista previa (imagen) o descargar el archivo (PDF), así como los metadatos referentes al objeto.



Figura 4.9: Interfaz de consulta (Sin cambios)

Para mostrar mas información tuvimos que hacer llamadas al api-A de Fedora. Para separar funcionalidades, creamos un archivo llamado dataFedora.php donde llamamos al método ListDatastreams para obtener todos los datastreams del objeto y luego llamamos al método del api-M, getDatastream para obtener los datos consultados por el usuario.

```
$client = new SoapClient(
    'http://' . urlencode($login) . ':' . urlencode($password) . '@' . $hostname . ':8080/fedora/wsd1?api=API-M',
    array(
        'login' => $login,
        'password' => $password
    )
);
$response = $client->getDatastream($options);
```

Figura 4.10: Llamado SOAP al método getDatastream

La principal modificación que hicimos a la interfaz que permite ver los registros de los objetos consultados, la ruta del archivo que modificamos es /var/www/html/ohs/plugins/schemas/dc/record.tpl.

Ejecutamos código php desde ese archivo para llamar a los métodos de dataFedora.php:

```
{php}
    $myVars = $this->get_template_vars('record');
    $myVars = $myVars->getParsedContents();

    foreach($myVars as $key => $value) {
        if($key=="identifier"){
            $ID = $value;
        }
    }
    $ID=(string)$ID[0];
    $response=lookIDS($ID);
    $this->assign('myVars',$response);
{/php}
```

Con los datos necesarios para hacer referencia al objeto, en el archivo `record.tpl` solamente hacemos llamadas `rest` al repositorio central para obtener la información de los objetos solicitados. Acá dejamos un ejemplo de la invocación al recurso:

```
<a href="http://{&i->hostname}:8080/fedora/objects/  
{&i->pid}/datastreams/{&i->DsID}/content" download="{&i}">
```

4.3.6. Firma electrónica

Otra de las funcionalidades requeridas es poder verificar la validez de un documento digital desde el portal de manera rápida y sencilla. Existen herramientas cliente que proveen de esta funcionalidad, pero la idea era integrar una herramienta que sea centralizada, fácil de utilizar y segura.

Para firmar los documentos primero se debe generar un certificado, para este certificado se necesita de una clave privada creada desde línea de comando:

```
openssl genrsa -out <Llave>.key <longitud>
```

- Llave: Nombre que proporcionara el usuario.
- Longitud: El tamaño de la clave, puede ser de 1024, 2048 o 4096 bytes.

Lo siguiente es generar un CSR (*Certificate Signing Request*), es la base para un certificado SSL:

```
openssl req -new -key <Llave>.key -out <Request>.csr  
-config <Ruta ssl>\openssl.cnf
```

Con este comando se van a solicitar datos como el dominio, organización, ubicación, información de contacto, entre otros. La llave solicitada es la clave privada que fue creada anteriormente.

Es importante destacar que estos pasos también son necesarios cuando vas a adquirir un certificado SSL de un proveedor autorizado, durante la

gestión del mismo, el proveedor va a solicitar este archivo para crear tu certificado. Por lo tanto, debemos tener mucho cuidado en que la información que ingresamos sea correcta.

```
Country Name (2 letter code) [AU]:VE
State or Province Name (full name) [Some-State]:distrito capital
Locality Name (eg, city) []:caracas
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:ucv
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:cert@ejemplo.com
```

Figura 4.11: Solicitud de datos

- Country Name (2 letter code): Código de país en formato ISO de dos letras.
- State or Province Name (full name): Estado o provincia.
- Locality Name: Localidad o ciudad.
- Organization Name: Nombre de la organización.
- Organizational Unit Name: Sector de la organización.
- Common Name: Nombre del dominio ó FQDN. Muy importante, hay una diferencia entre `www.ejemplo.com` a `ejemplo.com` sin `www`. Si registras tu certificado a una de estas opciones, no funcionará para el otro.
- Email Address: Dirección de correo de contacto.

El siguiente comando es:

```
openssl x509 -req -days 365 -in <CSR>.csr -signkey <Llave>.key
-out <CERT>.crt
```

- El parámetro `days` sirve para definir la fecha de expiración del certificado.
- `CSR`: Certificado CSR creado en el paso anterior.
- `Llave`: Clave privada creada en el paso inicial.
- `CERT`: Nombre del certificado publico.

El paso final es crear el certificado final en formato PKCS#12 (con extensión `.pfx` ó `.p12`). El formato Intercambio de información personal (PFX, también denominado PKCS#12) admite el almacenamiento seguro de certificados, claves privadas y todos los certificados en una ruta de certificación. El formato PKCS#12 es el único formato de archivo que se puede usar para exportar un certificado y su clave privada.

```
openssl pkcs12 -keypbe PBE-SHA1-3DES -certpbe PBE-SHA1-3DES -export  
-in <CSR>.crt -inkey <Llave>.key -out <Salida>.pfx -name "<Nombre>"
```

Este comando genera un archivo PFX listo para ser usado al momento de firmar. Finalizada su ejecución, ya creamos el certificado y lo siguiente es firmar, usamos `Xolidosing` para este proceso.

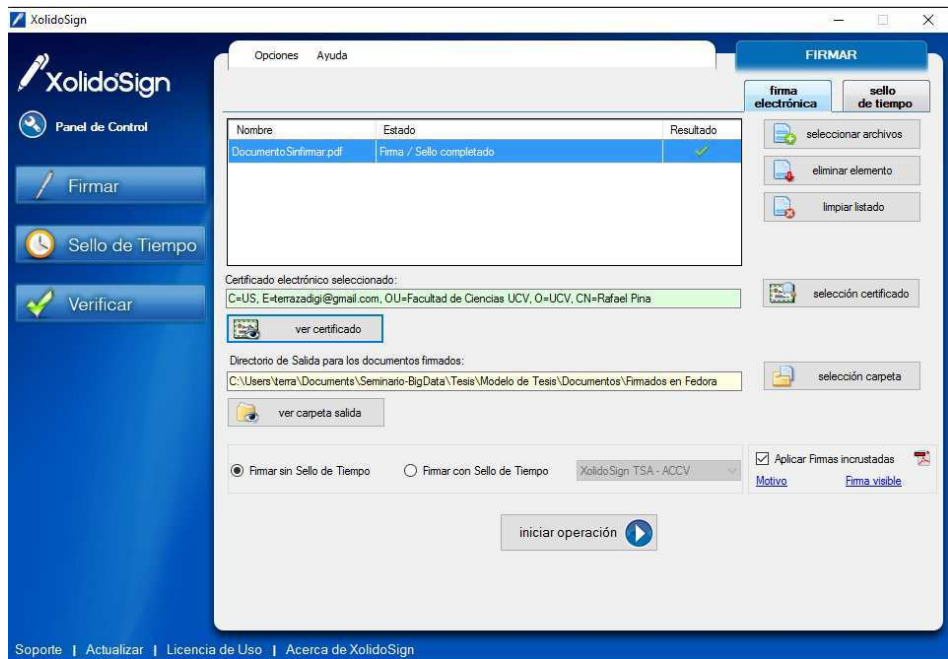


Figura 4.12: Interfaz Xolidosing

Primero seleccionamos el archivo a firmar, lo siguiente fue buscar el certificado que creamos anteriormente y usamos el botón 'Iniciar operación'. Esto genera un nuevo archivo PDF con la firma del electrónica.

Se agregó un modulo de validación de firma al OHS, donde por medio de un botón se elige el documento firmado y se carga en el portal. Y se crea una llamada POST al servicio *Secure Information Technology Center* - Austria, que provee de validación en Linea de un archivo digital.

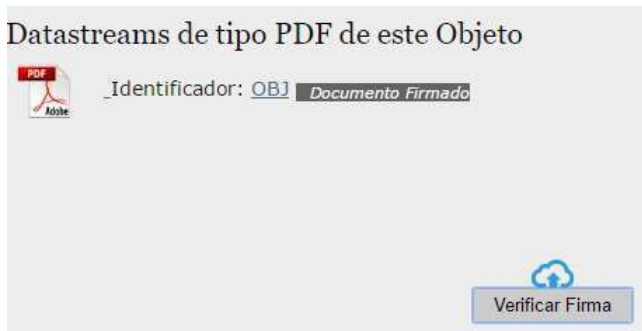


Figura 4.13: Modulo Validación Firma

Este valida la firma y el certificado emitido, muestra desde la interfaz un reporte donde se refleja el tipo de firma, el hash y si pasa la prueba de validación.

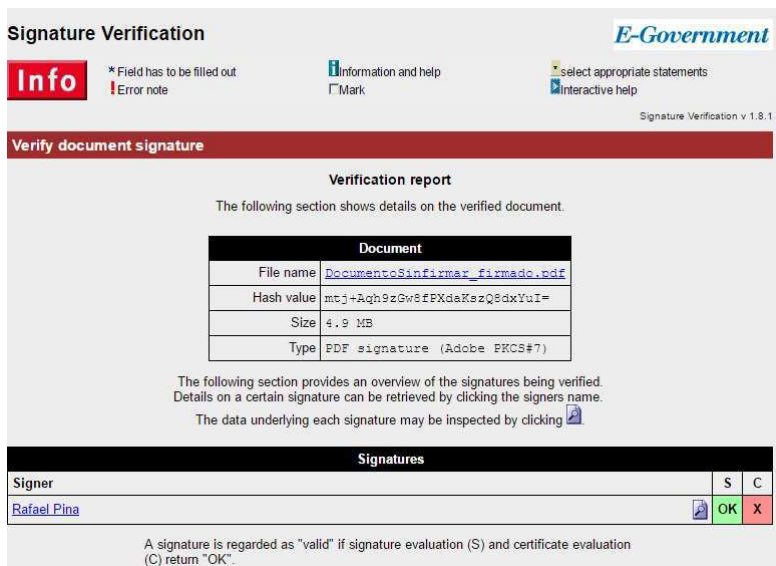


Figura 4.14: Respuesta Servicio Firma

Ademas, nos da la opción de ver la información del firmante y los datos asociados al certificado personal emitido, para que el servicio reconozca el Certificado generado debe haber sido emitido por un ente publico y institución registrada con certificado autentico, como la investigación tiene un enfoque académico se generaron certificados selfsigned que tienen validez local pero permiten agregar firma electrónica a un archivo.

Checks	
Time of signature and verification resp. (UTC)	2016-10-09T01:45:26Z
Signature	The verification of the signature value was successful.
Certificate	Unable to build a valid certificate chain up to a trusted root certificate.
Further information	
Type of signature	Adobe PKCS#7
	show signed data
Signer	
Name	Rafael Pina
Organizational unit	Facultad de Ciencias UCV
Organization	UCV
Country	US
E-Mail	terrazadigi@gmail.com
Issuer	
Name	Rafael Pina
Organizational unit	Facultad de Ciencias UCV
Organization	UCV
Country	US
E-Mail	terrazadigi@gmail.com
Certificate information	
Serialnumber	dec.: -467618164306674230811893, hex.: 9c:fa:5e:57:36:4c:12:f7:23:0b
Quality	non qualified certificate
Validity period	valid from 2016-10-01T18:30:32Z to 2021-10-01T18:30:32Z The given time of verification is within the validity period.
Keyusage	digitalSignature, dataEncipherment, keyAgreement

Figura 4.15: Información del Certificado Emitido

Capítulo 5

Conclusiones

Una vez finalizada toda la explicación de nuestro trabajo podemos afirmar que el sistema desarrollado es completamente funcional. Tenemos un cluster donde se guardan documentos que son gestionados por un repositorio digital. Para realizar las búsquedas usamos un indexador de metadata llamado *Open Harvester Systems* (OHS) y provee una interfaz que permite realizar búsquedas según los valores proporcionados por el usuario, Los metadatos deben estar en formato OAI, el resultado contiene todos los objetos digitales donde algún valor coincida con los recibidos en la interfaz. También para visualizar y manejar los objetos digitales del repositorio usamos Islandora, la ventaja que nos dio es que permite subir grandes cantidades de archivos al repositorio, esto es un punto importante porque hacer búsquedas es algo fácil pero una de las principales razones de este trabajo de grado es el manejo de grandes cantidades de documentos, por lo tanto esta plataforma debe ser capaz de permitir la carga masiva de documentos.

Iniciamos el trabajo configurando instancias en Amazon Web Services para tener un cluster Hadoop, luego instalamos el repositorio digital, usamos Fedora Commons porque tiene la facilidad de gestionar cualquier tipo recurso digital, gracias al modelo de objeto digital utilizado como mecanismo de representación.

La integración entre el repositorio y file system (HDFS) se hace con Akubra HDFS, una implementación de bajo nivel para el almacenamiento

en HDFS de objetos provenientes de Fedora. El siguiente paso fue trabajar con Islandora para manejar los objetos en el repositorio y OHS funcionando como interfaz para los usuarios.

Los tiempos de respuesta son óptimos, esto es porque OHS busca en todos los repositorios que hayan sido agregados, se enfoca en buscar en la metadata de todos los objetos para mostrar los resultados. En caso de querer verlos solo hace falta hacer click en la ruta que tiene el objeto para ver el documento.

Queremos finalizar mencionando que podemos usar la firma electrónica sobre los archivos antes de subirlos a la plataforma. El problema actual es que un documento digital no tiene la misma validez que un documento físico firmado. Esta implementación le otorgaría un carácter legal al sistema.

5.1. Aporte

Consideramos que este trabajo tiene un gran potencial porque integramos un conjunto de herramientas que brindan muchas ventajas, garantizamos una plataforma capaz de manejar grandes volúmenes de datos, búsquedas eficientes porque se realizan sobre metadatos presentes las etiquetas y tenemos interoperabilidad porque todos los repositorios pueden comunicarse con otros por una interfaz común donde internamente el repositorio expone todos los objetos solicitados sin distinguir el tipo de archivo.

Quizá es difícil adquirir el hardware para realizar una implementación lista para estar en producción, entonces nuestra solución fue trabajar en la nube. Esto reduce costos y brinda disponibilidad al servicio en todo momento, finalmente, permite enfocarse en el producto final.

5.2. Recomendaciones

- Estudiar la configuración de las herramientas para utilizar versiones mas recientes.

- Tener conocimientos básicos en PHP y el motor de plantillas Smarty para realizar modificaciones o agregar funcionalidades al OHS.
- Sugerimos tener conocimientos de sistemas operativos para realizar la instalación del cluster, específicamente recomendamos tener conocimientos sobre linux.
- Disponer de un cluster Hadoop para el entorno de pruebas, ya que las pruebas este ambiente entregan tiempos mucho menores que en computadores caseros.

5.3. Trabajos futuros

Hadoop ha crecido hasta constituir una gran familia de soluciones para el almacenamiento, gestión, interacción y análisis de grandes datos, integradas en un rico ecosistema de código abierto creado por la comunidad de la Apache Software Foundation. Por esta razón sugerimos que los futuros trabajos sean la inclusión de nuevas herramientas del ecosistema Hadoop.

Un punto a mejorar es estudiar que tan viable es usar una base de datos NOSQL para reducir la carga en la plataforma, es posible que se suban archivos de poco tamaño y el HDFS no esta diseñado para trabajar con archivos pequeños. En nuestra solución todos los datos van al HDFS pero sugerimos usar Apache HBase[6] para manejar esos archivos, de manera que se pueda integrar Documentos Firmados a una Base de datos NoSQL. Hadoop Solr es otra herramienta interesante para probar. Es una implementación de Apache Solr, el popular sistema de búsquedas, construido para trabajar directamente en el HDFS. Hadoop Solr utiliza el cluster para proveer escalabilidad y alta disponibilidad en el servicio de búsquedas.

Apache Mahout es otra herramienta con mucho potencial, es un proyecto para crear aprendizaje automático y data mining usando Hadoop. Es decir, Mahout nos puede ayudar a descubrir patrones en grandes datasets. Tiene algoritmos de recomendación, clustering y clasificación. Seria útil su inclusión a nuestra solución porque brindaría una forma sencilla de visualizar el trafico de documentos y sirve de apoyo para la toma de decisiones.

Apéndice A

Anexos

A.1. Adición de módulos

A.1.1. Drupal Filter

Luego de instalar Drupal, continuamos, añadiendo un modulo llamado Islandora, primero instalamos el Drupal Filter, una biblioteca que se instala en Fedora ubicado en el servidor de aplicaciones Tomcat. Es el lado de Fedora de la comunicación entre la biblioteca de Tuque de Islandora y el repositorio de Fedora.

Descargamos la ultima versión de Islandora Drupal Filter y copiarla en el directorio `$FEDORA_HOME/tomcat/webapps/fedora/WEB-INF/lib`. Lo hicimos ejecutando:

```
$ wget https://github.com/Islandora/islandora_drupal_filter/releases/download/v7.1.3/fcrepo-drupalauthfilter-3.8.0.jar
$ cp -v fcrepo-drupalauthfilter-3.8.0.jar $FEDORA_HOME/tomcat/webapps/fedora/WEB-INF/lib
```

El siguiente paso fue hacer que Fedora reconozca al filtro que añadimos. Lo hicimos ubicándonos en el directorio `$FEDORA_HOME/server/config` y abrimos el archivo `jaas.conf` en un editor de texto.

Para permitir que el Drupal Servlet Filter pueda en la base de datos de Drupal, reemplazamos la entrada "fedora-auth" con las siguientes líneas que hacen referencia a las clases del Drupal Servlet Filter:

```
fedora-auth
{
org.fcrepo.server.security.jaas.auth.module.XmlUsersFileModule required
debug=true;
ca.uepi.roblib.fedora.servletfilter.DrupalAuthModule required
debug=true;
};
```

El filtro todavía no está configurado para conectarse a Drupal, lo hicimos creando un archivo llamado filter-drupal.xml en el directorio \$FEDORA_HOME/server/config con el siguiente contenido:

```
<FilterDrupal_Connection>
  <connection server="localhost" dbname="[drupal_database]"
  user="[drupal_db_user]" password="[drupla_db_password]" port="3306">
    <sql>
      SELECT DISTINCT u.uid AS userid, u.name AS Name,
      u.pass AS Pass, r.name AS Role FROM (users u LEFT
      JOIN users_roles ON
      u.uid=users_roles.uid) LEFT JOIN role r ON r.rid=users_roles.rid
      WHERE u.name=? AND u.pass=?;
    </sql>
  </connection>
</FilterDrupal_Connection>
```

Modificamos los atributos de la etiqueta 'connection' para que coincida con el servidor, puerto, nombre de la base de datos, nombre de usuario y la contraseña de Drupal. Si existieran varios sitios Drupal entonces debemos agregar una etiqueta 'connection' por cada sitio.

Y cambiamos al propietario que ejecuta el servidor web porque Islandora necesita modificar el archivo.

```
# chown www-data:www-data filter-drupal.xml
```

Para iniciar el Drupal Servlet Filter tuvimos que detener y reiniciar el servidor Tomcat.

```
# $FEDORA_HOME/tomcat/bin/shutdown.sh
# $FEDORA_HOME/tomcat/bin/startup.sh
```

A.1.2. Tuque

El siguiente paso es instalar Tuque, el cual simplemente funciona como plugin que valida la conexión a un repositorio Fedora.

```
# mkdir -p /var/www/drupal/sites/all /var/www/drupal/sites/all/libraries
# cd /var/www/drupal/sites/all/libraries
# wget https://github.com/islandora/tuque/archive/1.5.zip
# unzip 1.5.zip
# mv tuque-1.5 tuque
```

Para el funcionamiento mínimo de Islandora se requieren dos módulos esenciales:

- Islandora Core Module
- Islandora Basic Collection Solution Pack

A.1.3. Islandora Core Module

Primero explicaremos como configuramos Islandora Core Module que lo descargamos desde:

```
https://github.com/islandora/islandora/archive/7.x-1.5.zip
```

El archivo comprimido lo guardamos en /opt/downloads. Lo colocamos en la carpeta 'modules' y removemos el numero de versión en el nombre del archivo. Frecuentemente se realiza la instalación en el directorio /var/www/drupal/sites/all/modules. Todo lo que mencionamos se realiza con estos comandos:

```
# cd /var/www/drupal/sites/all/modules
# unzip /opt/downloads/islandora-7.x-1.5.zip
# mv islandora-7.x-1.5 islandora
# chown -R www-data:www-data islandora
```

Solamente falta cambiar los permisos de los subdirectorios de Drupal para que coincida con el demonio del servidor web y reiniciar Tomcat.

```
# cd /var/www/drupal/sites/all
# chown -R www-data:www-data *

# $FEDORA_HOME/tomcat/bin/shutdown.sh
# $FEDORA_HOME/tomcat/bin/startup.sh
```

Debemos activar el modulo navegando a la dirección 'URL-Drupal'/admin/modules y buscamos en la lista de módulos y seleccionar Islandora Core Module (en la categoría de Islandora. Luego salvamos la configuración.



Figura A.1: Islandora Core Module

Una vez instalado, las opciones de configuración para el módulo Islandora se pueden encontrar en su sitio en <http://URL-Drupal/admin/islandora/configure>.

General Configuration

Namespaces

Excluded DSID

Fedora base URL *
http://localhost:8080/fedora
The URL to use for REST connections
✔ Successfully connected to Fedora Server (Version 3.8.1).

Root Collection PID *
islandora:root
The PID of the Root Collection Object

Generate/parse datastream HTTP cache headers
HTTP caching can reduce network traffic, by allowing clients to use cached copies.

Defer derivative generation during ingest
Prevent derivatives from running during ingest, useful if derivatives are to be created by an external service.

Show print option on objects
Displays an extra print tab, allowing an object to be printed

UUID PID Generation
Generate Fedora object PIDs with v4 UUIDs.

Save configuration

Figura A.2: Panel de configuración

A.1.4. Islandora Basic Collection Solution Pack

Este modulo lo descargamos desde:

https://github.com/islandora/islandora_solution_pack_collection/archive/7.x-1.5.zip

El archivo lo descomprimos y borramos el numero de versión en su nombre. Este directorio resultante lo copiamos a `/var/www/drupal/site-s/all/modules` y cambiamos los permisos del usuario a `www-data`.

```
# cd /opt/downloads
# unzip islandora_solution_pack_collection-7.x-1.5.zip
# mv islandora_solution_pack_collection-7.x-1.5
islandora_solution_pack_collection
```

```
# cd /var/www/drupal/sites/all/modules
# cp -R /opt/downloads/islandora_solution_pack_collection .
# chown -R www-data:www-data islandora_solution_pack_collection
```

Lo siguiente fue entrar al menú de módulos de Drupal. En la parte inferior, seleccione el paquete Basic Collection Solution pack, y guardar la configuración. Asegúrese de que el paquete está habilitado sin errores.

Continuamos dirigiéndonos a la página principal del sitio de Drupal. Haga clic en el enlace en la parte inferior izquierda de la pantalla que dice Islandora Repository. Debería ver un enlace de 'Islandora repository' en el panel de exploración y una ventana titulada "Top-level Collection".

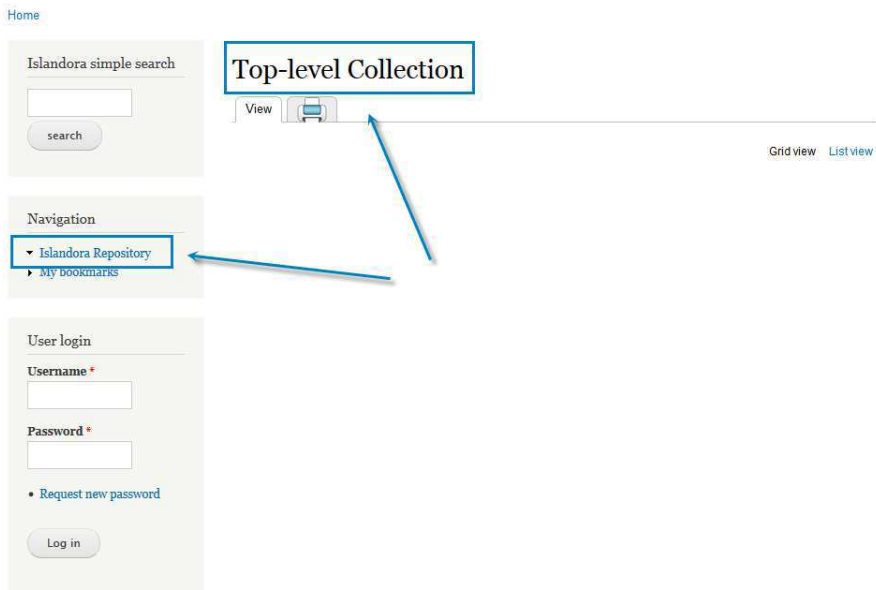


Figura A.3: Intefaz Islandora

A.2. Casos de uso

En esta sección vamos a explicar el funcionamiento de los casos de uso que consideramos mas complejos.

A.2.1. Islandora

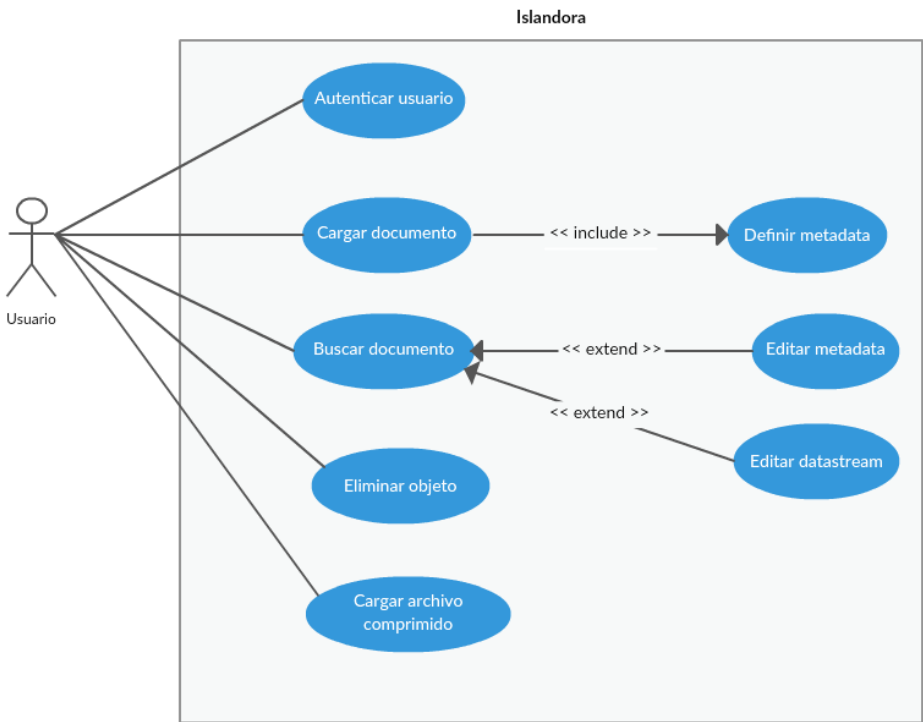


Figura A.4: Casos de uso - Islandora

A.2.1.1. Autenticar usuario

La autenticación se hace cuando ingresamos al sitio Drupal. Ese usuario debe tener las mismas credenciales para acceder a la base de datos de Drupal.

A.2.1.2. Cargar documento

Entramos a manage y luego a 'Add an object to this Collection' para crear el objeto.

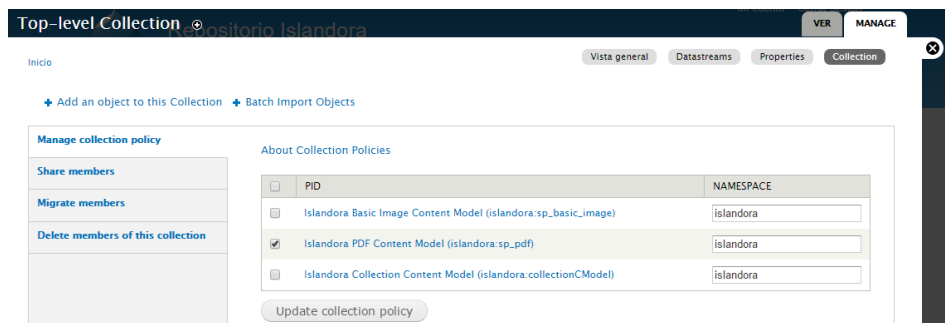
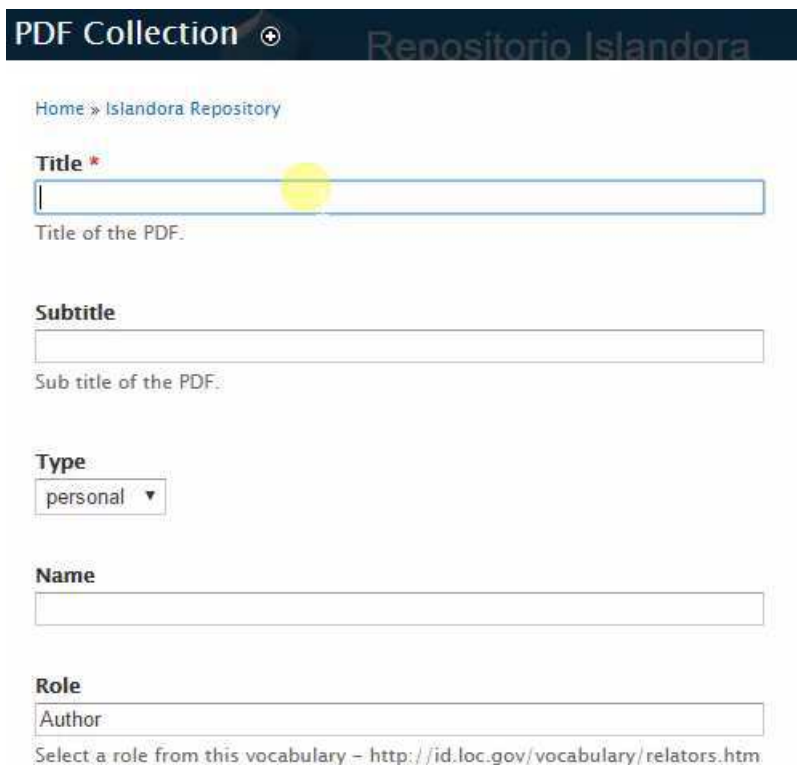


Figura A.5: Top-Level Collection

En la siguiente pantalla nos solicitan los valores que tendrá el objeto, luego de aceptar, debimos seleccionar el archivo a subir desde nuestra máquina.



The image shows a web form for defining metadata for a PDF collection. The header includes 'PDF Collection' with a plus icon and 'Repositorio Islandora'. Below the header is a breadcrumb trail: 'Home » Islandora Repository'. The form consists of several fields:

- Title ***: A text input field with a yellow highlight. Below it is the label 'Title of the PDF.'.
- Subtitle**: A text input field. Below it is the label 'Sub title of the PDF.'.
- Type**: A dropdown menu with 'personal' selected and a downward arrow.
- Name**: A text input field.
- Role**: A dropdown menu with 'Author' selected. Below it is the text 'Select a role from this vocabulary - <http://id.loc.gov/vocabulary/relators.htm>'.

Figura A.6: Definir metadatos

A.2.1.3. Eliminar objeto

Podemos eliminar objetos desde Islandora, solo es necesario buscar el objeto o navegar por alguna colección del repositorio. Al seleccionar el objeto, hicimos click en 'Properties' y luego en el boton 'Permanently remove from repository'.

A.2.1.4. Cargar archivo comprimido

Islandora permite cargar varios archivos de forma sencilla, tuvimos que crear un archivo ZIP que contenga los archivos a cargar, además, por cada archivo debe estar acompañado por un archivo XML con el mismo nombre, este archivo contiene los metadatos que permiten buscar al respectivo documento en el repositorio.

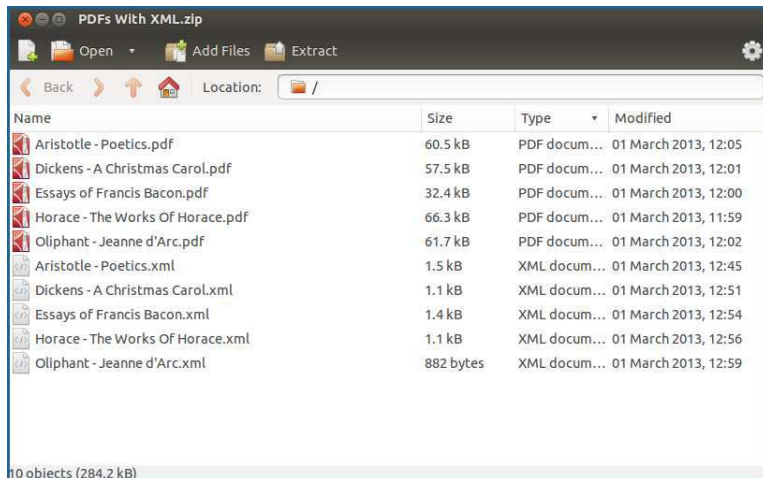


Figura A.7: Ejemplo del archivo ZIP requerido

```
<?xml version="1.0"?>
<mods xmlns="http://www.loc.gov/mods/v3" xmlns:mods="http://www.loc.gov/mods/v3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <titleInfo>
    <title>Title</title>
    <subTitle/>
  </titleInfo>
  <titleInfo>
    <name types="personal">
      <namePart>Aristotle</namePart>
      <role>
        <roleTerm authority="marcrelator" type="text">Author</roleTerm>
      </role>
    </name>
    <typeOfResource>text</typeOfResource>
    <abstract>Aristotle's</abstract>
    <genre>article</genre>
    <note/>
    <subject>
      <topic/>
      <geographic/>
      <temporal/>
    </subject>
    <location>
      <url/>
    </location>
  </mods>
```

Figura A.8: Estructura cada archivo XML

Luego nos dirigimos a la pestaña 'Collection', luego entramos en Batch Import Objects donde aparece la siguiente interfaz.

ZIP BATCH IMPORTER

Select the file containing the assets and metadata to import. Assets and metadata will be matched together based on the portion of the filename without the extension, so my_file.xml and my_file.pdf will be combined into a single object.

Zip file of files to import

Choose File | PDFs Without XML.zip | Upload **1**

CONTENT MODEL

The content model(s) to assign to the imported objects.

<input type="checkbox"/>	NAME
<input type="checkbox"/>	Islandora Collection Model - islandora:collectionCModel
<input checked="" type="checkbox"/> 2	Islandora PDF Content Model

Object Namespace

islandora **3**

The namespace in which the imported objects will be created.

Import **4**

Figura A.9: Zip Batch Importer

1. Seleccionamos el archivo Zip.
2. Elegimos los modelos que aplican para los objetos creados.
3. Seleccionamos el namespace.
4. Hicimos click en el boton 'import'.

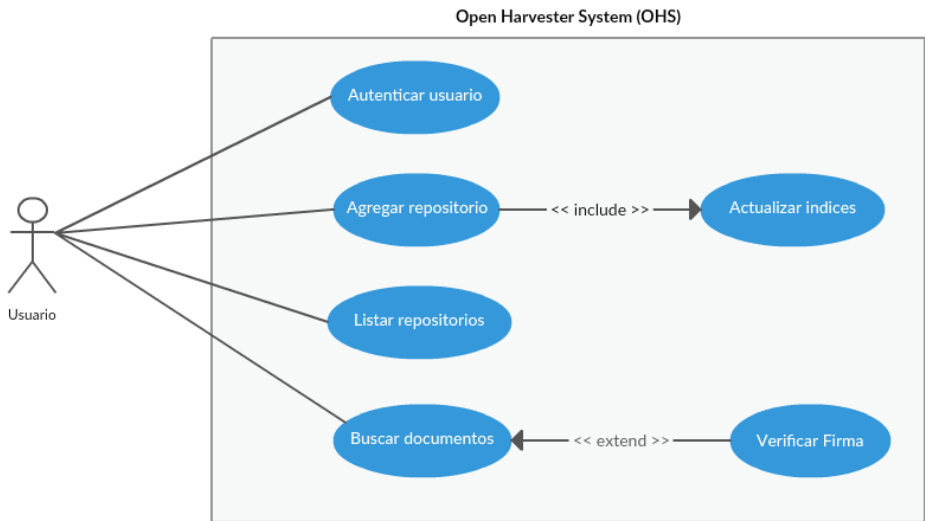
A.2.2. OHS

Figura A.10: Casos de uso - OHS

A.2.2.1. Agregar repositorio

En esta pantalla ingresamos toda la información requerida, los campos mas importantes son la ruta del repositorio y la dirección donde están alojados los metadatos del repositorio.

The image shows a web form for adding a repository. The fields and their values are as follows:

Título	Repositorio Central
URL*	http://ec2-52-42-117-41.us-west-2.compute.am Ej: http://www.repositorio.com
Identificador	repositorio central
Tipo*	OAI ▼
OAI Base URL*	http://ec2-52-42-117-41.us-west-2.compute.am Ej: http://www.repositorio.com:8080/fedora/oai
Definicion Repositorio	<input type="checkbox"/> ##Repositorio Estatico##
Metodo de Indexacion	ListRecords ▼
Formato de la Metadata*	Dublin Core ▼ Refresh

Figura A.11: Agregar repositorio

A.2.2.2. Buscar Documentos

Acá solamente tenemos que rellenar los campos correspondientes para buscar el documento.

Buscar Documentos

All

Repositorios

Resumen

Contribuyentes

Cobertura

Creador

Descripcion

Formato

Identificador

Idioma

Autor

Relacion

Derechos

Fuente

Motivo

Titulo

Tipo

Buscar Documentos

Figura A.12: Buscar documentos

Luego seleccionamos el documento que buscamos y se visualiza toda la información que contiene el objeto digital. También se puede observar la opción de verificar firma, consiste en seleccionar el certificado generado para firmar, validar la identidad del firmante.

Repositorio Central

[VER INFORMACION DEL REPO](#)

FIELD	VALUE
Titulo	New Object
Identificador	islandora:4 islandora:4

Datastreams de tipo PDF de este Objeto

 [_Identificador: OBJ](#) **Documento Firmado**

 [Verificar Firma](#)

Figura A.13: Información del documento



Figura A.14: Objeto desplegado con datastreams

A.3. Generar archivos de prueba

Es posible subir grandes cantidades de archivos al sistema, para simular este comportamiento creamos un proceso desarrollado en Java que genera archivos PDF y por cada uno, también crea su respectivo XML. Fue necesario incluir la librería `iText-5.0.5` para la creación de los archivos.

El archivo Jar que desarrollamos se ejecuta de la siguiente forma:

```
java -jar FileToXml.jar -pdf numero
```

El parámetro 'numero' es la cantidad de archivos requeridos por el

usuario. Acá hacemos una demostración de este proceso. Ejecutamos desde consola:

```
java -jar FileToXml.jar -pdf 10
```

Y genera la cantidad de archivos PDF y XML indicados por el usuario, la estructura de cada XML esta descrita en los anexos, casos de uso Islandora.























 lib	Carpeta de archivos	
 archivo_0	Archivo PDF	4.978 KB
 archivo_0	Documento XML	1 KB
 archivo_1	Archivo PDF	4.978 KB
 archivo_1	Documento XML	1 KB
 archivo_2	Archivo PDF	4.978 KB
 archivo_2	Documento XML	1 KB
 archivo_3	Archivo PDF	4.978 KB
 archivo_3	Documento XML	1 KB
 archivo_4	Archivo PDF	4.978 KB
 archivo_4	Documento XML	1 KB
 archivo_5	Archivo PDF	4.978 KB
 archivo_5	Documento XML	1 KB
 archivo_6	Archivo PDF	4.978 KB
 archivo_6	Documento XML	1 KB
 archivo_7	Archivo PDF	4.978 KB
 archivo_7	Documento XML	1 KB
 archivo_8	Archivo PDF	4.978 KB
 archivo_8	Documento XML	1 KB
 archivo_9	Archivo PDF	4.978 KB
 archivo_9	Documento XML	1 KB
 FileToXml	Executable Jar File	9 KB

Figura A.15: Directorio con los archivos generados

Si el directorio ya posee los archivos PDF, tambien es posible generar solamente los archivos XML. Ejecutando de la siguiente forma:

```
java -jar FileToXml.jar -xml 'rutaDestino'
```

Índice de figuras

2.1. Línea de tiempo Hadoop (Fuente: SAS)	11
2.2. Esquema HDFS (Fuente: MadridSchool)	14
2.3. Componentes de un objeto digital. Fuente: FEDORA	21
2.4. Datastreams reservados	22
2.5. Estándares de metadatos	28
2.6. Ejemplo búsqueda OAI - PMH	32
2.7. Costos por TB Hadoop.	41
2.8. Costos por hora Instancias bajo Demanda	42
2.9. Pila de protocolos de los servicios web	44
2.10. Proceso Básico de Firma Electrónica	47
2.11. Logo Xolidosing	48
3.1. Pasos para la integración de herramientas	52
4.1. Arquitectura de la solución	56
4.2. Descripción instancia EC2	59
4.3. Interfaz de Cluster Hadoop y sus nodos	66
4.4. Información del repositorio Fedora	70
4.5. Objetos digitales almacenados en el cluster	73
4.6. Interfaz Drupal	75
4.7. Configuración de ruta al sistema	76
4.8. Intefaz Open Harvester System	77
4.9. Interfaz de consulta (Sin cambios)	77
4.10. Llamado SOAP al método getDatatsream	78
4.11. Solicitud de datos	80
4.12. Interfaz Xolidosing	82

4.13. Modulo Validación Firma	83
4.14. Respuesta Servicio Firma	83
4.15. Información del Certificado Emitido	84
A.1. Islandora Core Module	92
A.2. Panel de configuración	93
A.3. Intefaz Islandora	94
A.4. Casos de uso - Islandora	95
A.5. Top-Level Collection	96
A.6. Definir metadatos	97
A.7. Ejemplo del archivo ZIP requerido	98
A.8. Estructura cada archivo XML	99
A.9. Zip Batch Importer	100
A.10.Casos de uso - OHS	101
A.11.Agregar repositorio	102
A.12.Buscar documentos	103
A.13.Información del documento	104
A.14.Objeto desplegado con datastreams	105
A.15.Directorio con los archivos generados	106

Bibliografía

- [1] *Akubra Project*. URL: <https://wiki.duraspace.org/display/AKUBRA/Akubra+Project>.
- [2] A. Silberschatz y et al. *Operating System Concepts*. 9th ed. Wiley, 2013.
- [3] *Amazon EC2 Cloud Computing*. URL: <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>.
- [4] *Amazon Web Services*. URL: <https://aws.amazon.com/es/>.
- [5] *Amazon Web Services Pricing*. URL: <https://aws.amazon.com/es/ec2/pricing/>.
- [6] *Apache Hbase*. URL: <http://hbase.apache.org/>.
- [7] *Cloudera CDH Hadoop 5*. URL: <http://www.cloudera.com/content/cloudera/en/home.html>.
- [8] Fedora Commons. *The Fedora Basics*. URL: <https://wiki.duraspace.org/display/FEDORA36/Getting+Started+with+Fedora>.
- [9] *Configuring Low Level Storage*. URL: <https://wiki.duraspace.org/display/FEDORA34/Configuring+Low+Level+Storage>.
- [10] *CURL*. URL: <https://curl.haxx.se/>.
- [11] *Data Documentation Initiative*. URL: https://en.wikipedia.org/wiki/Data_Documentation_Initiative.
- [12] *Dublin Core*. URL: <http://searchsoa.techtarget.com/definition/Dublin-Core>.
- [13] *Elastic Block System*. URL: <https://aws.amazon.com/es/ebs/>.

- [14] *Fedora API-A*. URL: <https://wiki.duraspace.org/display/FEDORA36/API-A>.
- [15] *Fedora API-M*. URL: <https://wiki.duraspace.org/display/FEDORA36/API-M>.
- [16] *Filesystem in Userspace*. URL: <http://fuse.sourceforge.net/>.
- [17] Laureano Felipe Gómez Dueñas. “La Iniciativa de Archivos Abiertos (OAI), un nuevo paradigma en la comunicación científica y el intercambio de información”. En: Universidad de la Salle. Cap. 4, pág. 25.
- [18] *Hadoop User Experience*. URL: <http://cloudera.github.io/hue/>.
- [19] *HDFS Architecture*. URL: <http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>.
- [20] *METS*. URL: <http://www.loc.gov/standards/mets/>.
- [21] *MIME Types*. URL: <https://www.sitepoint.com/web-foundations/mime-types-complete-list/>.
- [22] *My SQL Bible*. URL: <https://imcs.dvfu.ru/lib.int/docs/Databases/MySQL/MySQL%20Bible.pdf>.
- [23] *Open Harvester System*. URL: <https://pkp.sfu.ca/ohs/>.
- [24] OReilly. *Hadoop The Definitive Guide 4th Edition 2015*. URL: <https://www.iteblog.com/downloads/OReilly.Hadoop.The.Definitive.Guide.4th.Edition.2015.3.pdf>.
- [25] KahnyWilensky Palavitsinis. 2013, pág. 19.
- [26] Cristian Vasquez Paulus. *METADATOS: Introducción e historia*. URL: <https://users.dcc.uchile.cl/~cvasquez/introehistoria.pdf>.
- [27] *Proceso básico de firma electrónica*. URL: http://firmaelectronica.gob.es/Home/Ciudadanos/Firma-Electronica.html#proceso_basico.
- [28] *Putty SSH Client*. URL: <http://www.chiark.greenend.org.uk/~sgtatham/putty/>.
- [29] *RENa*. URL: <http://www.rena.edu.ve/cuartaEtapa/Informatica/Tema9.html>.

- [30] E. C. Foster y S. V. Godbole. *Database Systems. A pragmatic approach*. 1ra. ed. APress, 2014, 2009.
- [31] Alexander Barón Salazar. “Estudio de herramientas para la gestión de repositorios digitales”. En: *Annalen der Physik* 2 (2012), pág. 31.
- [32] Claudio Guitierrez Sergio Ochoa Cecilia Bastarrica. *DOCUMENTACIÓN ELECTRÓNICA e INTEROPERABILIDAD de la INFORMACIÓN*. Universidad de Chile, 2009, pág. 36. ISBN: 978-956-19-0633-4.
- [33] A. S. Tanenbaum. *Modern Operating Systems*. 3ra. ed. Pearson, Prentice Hall, 2007.
- [34] Universitat Politecnica de Valencia. *¿Qué es un Certificado Digital?*
URL: <http://www.upv.es/contenidos/CD/info/711545normalc.html>.