Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

HP Service Activator Log Mining

Autor:
D. Carlos Tejo Sánchez

Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

HP Service Activator Log Mining

Autor:
D. Carlos Tejo Sánchez

Tutor:
D. Pablo de la Fuente Redondo

Trabajo Fin de Grado

HP Service Activator Log Mining

Mayo 2016

Resumen

En este trabajo nos centraremos en la creación de una aplicación web que nos permita transformar una base de datos en formato XML a una base de datos NoSQL. Este proyecto es una parte de una aplicación más compleja empleada por HP.

Inicialmente se ha realizado un estudio de diferentes alternativas de sistemas NoSQL en detalle para poder decidir el candidato más apropiado. Se han analizado las distintas tecnologías que habría que emplear en el proceso, así como el funcionamiento de alguna de ellas.

Hay que considerar el contexto de la aplicación, dado el uso de datos semejantes a los reales, y se ha establecido una planificación temporal ajustada a las necesidades de los clientes.

Palabras clave

HP Service Activator, NoSQL, bases de datos documentales, Solr, JAXB, Java, XML, aplicación web, Tomcat.

Abstract

In this assignment we will focus on creating a web application that allows us to transform a database in XML format to a NoSQL database. This project is a part of a more complex application used by HP.

Initially it conducted a study of different alternatives of NoSQL systems in detail in order to decide the most appropriate candidate. We analyzed the different technologies that should be used in the process and the operation of some of them.

We must consider the context of the application, given the use of such data to the real, and has established a temporary planning tailored to the needs of customers

Keyword

HP Service Activator, NoSQL, documental data bases, Solr, JAXB, Java, XML, web app, Tomcat.

Índice

Índice de imágenes

Índice de tablas

1. Introducción y contexto

HP Service Activator es una plataforma software que permite la activación remota de servicios propiedad de HP. Esta plataforma fue desarrollada en el año 2000 como una solución de centro de datos de internet en España. En el año 2001 acuño el nombre de Service Activator bajo la supervisión del grupo de comunicación, media y entretenimiento de HP junto con la organización de consultorio e integración de HP. Finalmente, en 2004, fue lanzada como una solución global de activación CSP. La última generación de esta herramienta fue lanzada en 2008.

Pongamos un ejemplo para entender mejor el contexto de esta aplicación. Un cliente de determinada compañía de telecomunicaciones desea activar un servicio nuevo, por ejemplo, una ampliación de su plan de internet. Si nos remontamos a años antes de que esta aplicación fuera lanzada al mercado, un técnico hubiera tenido que ir al domicilia del cliente para poder cambiar la configuración de su router y de los servicios ofrecidos. Gracias a aplicaciones como la que HP nos proporciona podremos activar el servicio desde la propia sucursal de la empresa de telecomunicaciones reduciendo costes, tiempo y esfuerzo.

Empresas como Movistar, Jazztel o Vodafone emplean esta herramienta.

Este trabajo se centrará en una parte más pequeña de la plataforma, en concreto con la aplicación que recoge toda la información de las incidencias, es decir, la que crea los logs de estas.

Estos logs se dividen en trabajos o jobs. Cada trabajo está formado por una serie de entradas relacionadas entre sí que corresponderían a la vida total de una incidencia durante todo su proceso.

Más técnicamente estos logs se encuentran en un formato XML. En el anexo I podemos ver una pequeña porción de uno de ellos.

Inicialmente este formato es poco útil para realizar cambios, búsqueda de información por lo que hay que pensar en otro sistema que nos permita realizarlo con más facilidad y rapidez. La solución propuesta a este problema es emplear una base de datos, en concreto una base de datos NoSQL, por lo que será necesario almacenar toda la información contenida en los XML en la base de datos que elijamos.

La información inicial, en algunos casos, no está totalmente completada, ni detallada y en algunas entradas existe una falta de información por lo que tendremos también que decorarlas.

Todo esto debe permitirse ser ejecutada a través de una interfaz web.

2. Análisis del problema a resolver

El problema que tiene la empresa HP es que la aplicación desarrollada en Holanda que se integra con Service Activator genera los archivos logs de incidencias en formato XML. ¿Cuál es el problema con este tipo de documentos? En primer lugar y siendo este uno de los grandes problemas es que el sistema de búsqueda no es muy complejo, permitiendo solamente búsquedas simples. Otro problema es la existencia de numerosos documentos con información relacionada, es decir, toda la información no está centralizada. Viendo estos dos problemas, lo lógico es emplear una base de datos potente que nos ayude a solucionarlos, en nuestro caso, una base de datos NoSQL.

Las opciones que existen son múltiples, y ya no solo sistemas sino tipo de sistemas. Si entramos en la página http://nosql-database.org/ veremos que la cantidad de bases de datos NoSQL ascienden a más de 225 por lo que la decisión de emplear una u otra tendrá que ser sometido a un examen exhaustivo.

Otro problema que nos proporciona la aplicación que genera los logs originales es que no genera todos los campos que necesitaremos para el análisis futuro. Uno de los campos más importantes es el campo ID que identifica un trabajo o una incidencia, es decir todos aquellos campos que tengan el mismo ID pertenecerán a la misma incidencia. Aquí está uno de los problemas que nos encontramos, existen zonas ciegas en los logs que deja la aplicación. Pongamos dos ejemplos para entender mejor cual es el problema:

- Un trabajo está formado por distintas entradas. Lo normal sería que si avanzamos a lo largo de todas las entradas de un trabajo (todas aquellas que tengan el mismo ID), encontremos todo el flujo que ha seguido. Pero esto no es así ya que podemos echar en falta ciertas entradas. Esto se debe a que la aplicación oculta o no escribe el ID en ciertos casos perdiendo estas entradas si seguimos el flujo.

- Otro problema es que pueden existir entradas con el mismo número de ID pero que no pertenecen al mismo job, sino que pertenece a un job hijo, que se considera diferente al padre por lo que tendremos que recuperar el ID del hijo para remplazarlo.

No todas las entradas con ID vacío o inexistente pertenecen a un trabajo, sino que son simplemente procesos internos de la aplicación. Esto es otra dificultad añadida ya que hay una gran mayoría de estas.

Como vemos la aplicación no crea la información tal cual nos gustaría que fuese el resultado final, por lo que tendremos que ser nosotros quien consiga ese objetivo.

Es necesario dejar todo preparado, de tal manera que sea sencillo nuevas implementaciones sin tener que modificar el código en gran medida.

Otro de los objetivos de este proyecto es dejar preparado todo para que funcione como una aplicación web dejando preparado todo para incorporaciones y funciones futuras a la interfaz. En

nuestro caso será tan sencillo como permitir la carga y la resolución de los problemas de la parte ciega de la base de datos con un simple botón y permitir una búsqueda simple en la base de datos.

Como vemos la parte principal que tendremos que desarrollar es la parte de back-end, dejando la brecha abierta para el desarrollo definitivo del fornt-end.

3. Planificación

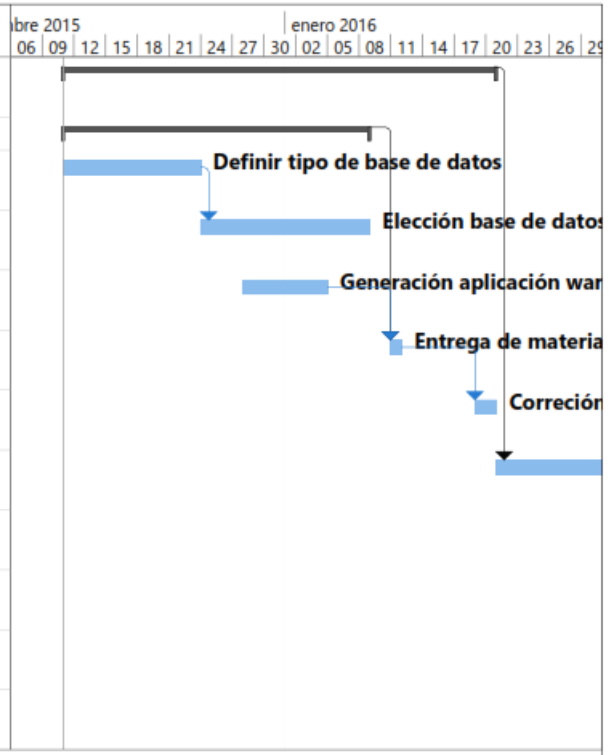
La metodología empleada es una metodología iterativa. Al inicio del proyecto se desarrolló una planificación general indicando las iteraciones que tendría la planificación, así como la longitud de cada una, aunque esta podrá estar sometida a cambios o incluso incorporaciones nuevas por nuevas exigencias del cliente. En nuestro caso sí ha habido cambios en el desempeño del proyecto con incorporación de nuevas tareas que provocaría cambios en la planificación.

Al comienzo de cada iteración se realizaba una reunión con los objetivos a cumplir durante esa iteración ya que el cliente no definía claramente lo que deseaba hasta ese momento.

El calendario utilizado para la planificación es el calendario por defecto, que incluye los días entre el lunes y el viernes. El número de horas diarias no es fijo, pero oscilan entre 2 y 3 horas.

La planificación final, tras haber planificado todas las iteraciones se muestra en las siguientes imágenes obtenidas de la aplicación MS Project.

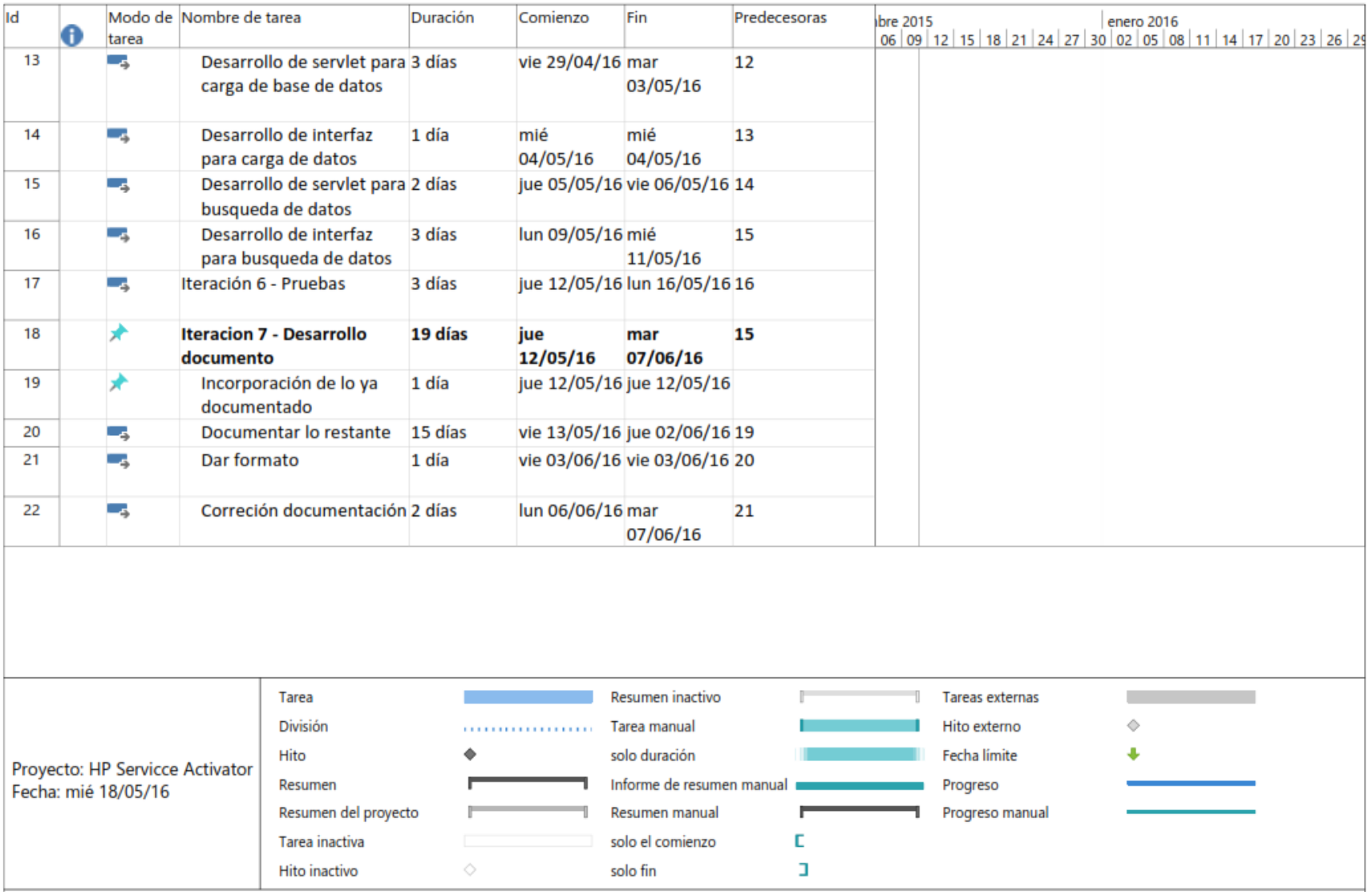
Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		**Iteración 1 - Documentación inicial**	**29 días**	**vie 11/12/15**	**mié 20/01/16**	
2		**Elección sistema NoSQL**	**21 días**	**vie 11/12/15**	**vie 08/01/16**	
3		Definir tipo de base de datos	9 días	vie 11/12/15	mié 23/12/15	
4		Elección base de datos concreta	12 días	jue 24/12/15	vie 08/01/16	3
5		Generación aplicación war	6 días	lun 28/12/15	lun 04/01/16	
6		Entrega de material	1 día	lun 11/01/16	lun 11/01/16	2;5
7		Correción documentación	2 días	mar 19/01/16	mié 20/01/16	6
8		Iteración 2 - Definir estructura	27 días	jue 21/01/16	vie 26/02/16	1
9		Iteración 3 - Desarrollo colector	21 días	lun 29/02/16	lun 28/03/16	8
10		Iteración 4 - Desarrollo decorador	21 días	mar 29/03/16	mar 26/04/16	9
11		**Iteración 5 - Desarrollo de la interfaz web**	**11 días**	**mié 27/04/16**	**mié 11/05/16**	**10**
12		Incorporación de los sistemas necesarios	2 días	mié 27/04/16	jue 28/04/16	

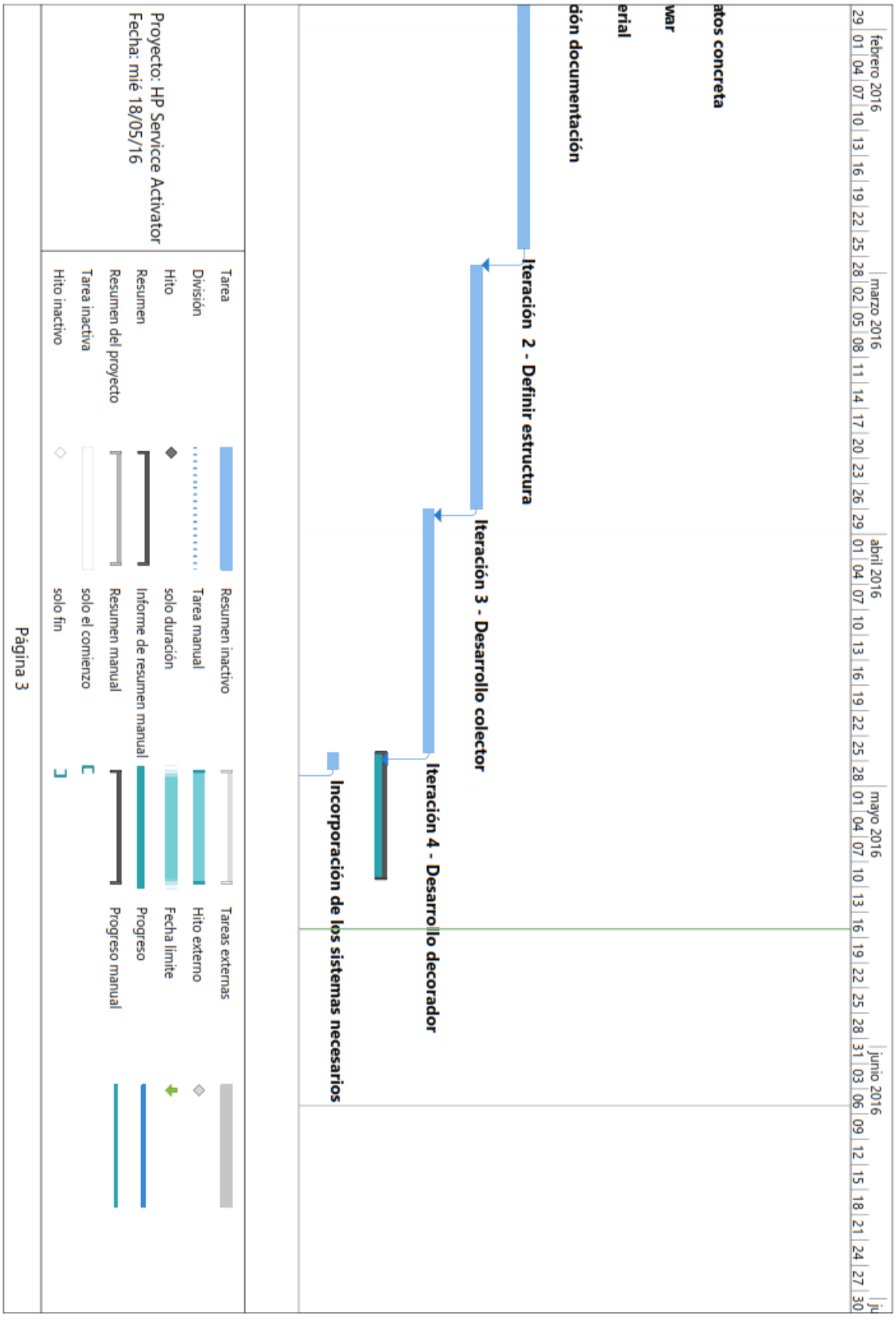


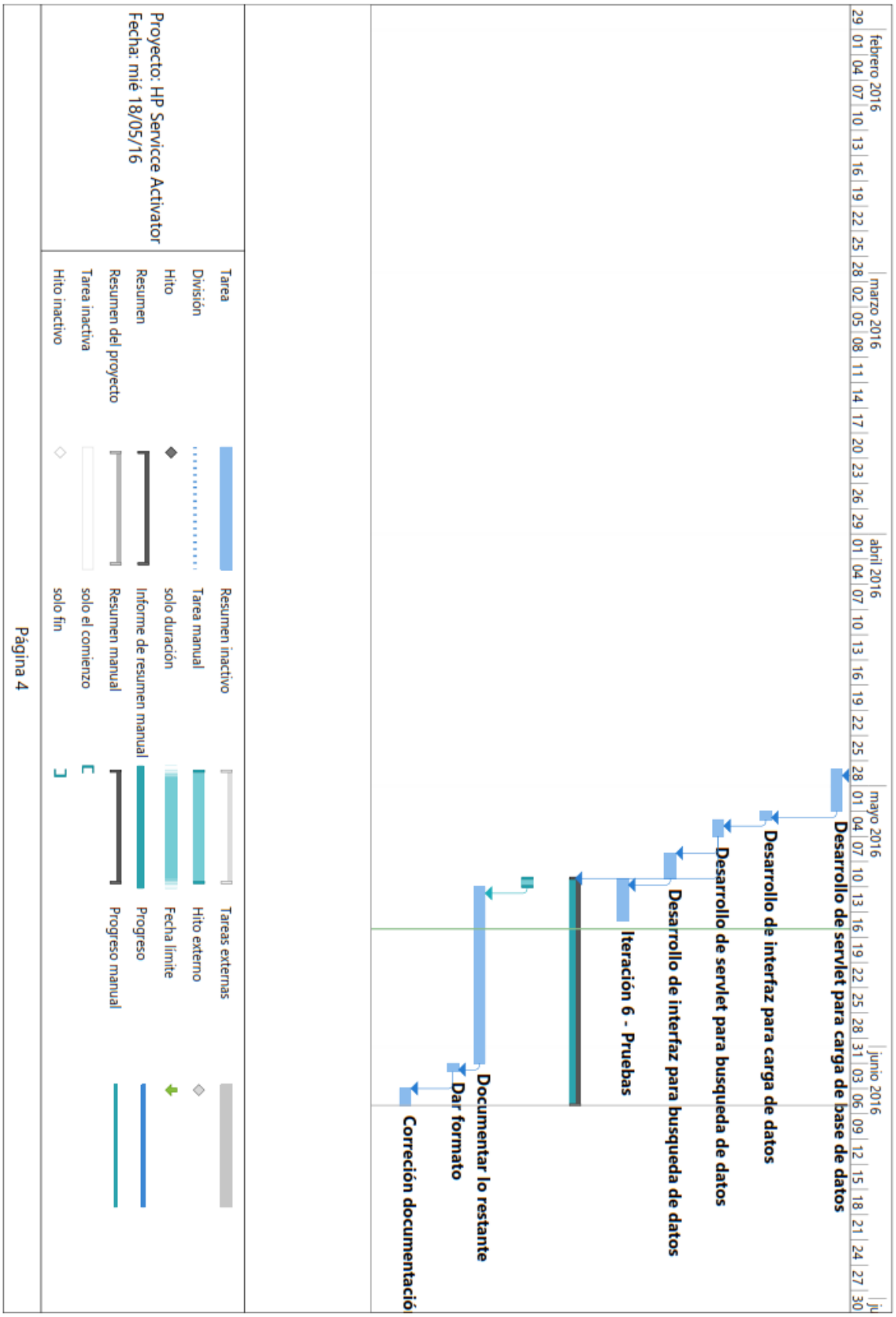
Proyecto: HP Servicce Activator
Fecha: mié 18/05/16

Tarea	Resumen inactivo	Tareas externas
División	Tarea manual	Hito externo
Hito	solo duración	Fecha límite
Resumen	Informe de resumen manual	Progreso
Resumen del proyecto	Resumen manual	Progreso manual
Tarea inactiva	solo el comienzo	
Hito inactivo	solo fin	

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
13		Desarrollo de servlet para carga de base de datos	3 días	vie 29/04/16	mar 03/05/16	12
14		Desarrollo de interfaz para carga de datos	1 día	mié 04/05/16	mié 04/05/16	13
15		Desarrollo de servlet para busqueda de datos	2 días	jue 05/05/16	vie 06/05/16	14
16		Desarrollo de interfaz para busqueda de datos	3 días	lun 09/05/16	mié 11/05/16	15
17		Iteración 6 - Pruebas	3 días	jue 12/05/16	lun 16/05/16	16
18		**Iteracion 7 - Desarrollo documento**	**19 días**	**jue 12/05/16**	**mar 07/06/16**	**15**
19		Incorporación de lo ya documentado	1 día	jue 12/05/16	jue 12/05/16	
20		Documentar lo restante	15 días	vie 13/05/16	jue 02/06/16	19
21		Dar formato	1 día	vie 03/06/16	vie 03/06/16	20
22		Correción documentación	2 días	lun 06/06/16	mar 07/06/16	21




febrero 2016
marzo 2016
abril 2016
mayo 2016
junio 2016
atos concreta
war
erial
ión documentación
Iteración 2 - Definir estructura
Iteración 3 - Desarrollo colector
Iteración 4 - Desarrollo decorador
Incorporación de los sistemas necesarios
Proyecto: HP Servicce Activator
Fecha: mié 18/05/16
Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin
Tareas externas
Hito externo
Fecha límite
Progreso
Progreso manual
Página 3


febrero 2016
marzo 2016
abril 2016
mayo 2016
junio 2016
Desarrollo de servlet para carga de base de datos
Desarrollo de interfaz para carga de datos
Desarrollo de servlet para busqueda de datos
Desarrollo de interfaz para busqueda de datos
Iteración 6 - Pruebas
Documentar lo restante
Dar formato
Corrección documentació
Proyecto: HP Servicce Activator
Fecha: mié 18/05/16
Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin
Tareas externas
Hito externo
Fecha límite
Progreso
Progreso manual
Página 4

julio 2016 | agosto 2016 | septiembre 2016 | octubre 2016 | noviembre 2016

30 03 06 09 12 15 18 21 24 27 30 02 05 08 11 14 17 20 23 26 29 01 04 07 10 13 16 19 22 25 28 01 04 07 10 13 16 19 22 25 28 31 03 06 09 12 15 18 21 24 27 30 d

ación

Proyecto: HP Servicce Activator
Fecha: mié 18/05/16

Tarea
División
Hito
Resumen
Resumen del proyecto
Tarea inactiva
Hito inactivo

Resumen inactivo
Tarea manual
solo duración
Informe de resumen manual
Resumen manual
solo el comienzo
solo fin

Tareas externas
Hito externo
Fecha límite
Progreso
Progreso manual

La metodología empleada es una metodología iterativa. Al inicio del proyecto se desarrolló una planificación general indicando las iteraciones que tendría la planificación, así como la longitud de cada una, aunque esta podrá estar sometida a cambios o incluso incorporaciones nuevas por nuevas exigencias del cliente. En nuestro caso si ha habido cambios en el desempeño del proyecto con incorporación de nuevas tareas que provocaría cambios en la planificación.

Al comienzo de cada iteración se realizaba una reunión con los objetivos a cumplir durante esa iteración ya que el cliente no definía claramente lo que deseaba hasta ese momento.

El calendario utilizado para la planificación es el calendario por defecto, que incluye los días entre el lunes y el viernes. El número de horas diarias no es fijo, pero oscilan entre 2 y 3 horas.

La planificación final, tras haber planificado todas las iteraciones se muestra en las siguientes imágenes obtenidas de la aplicación MS Project.

El seguimiento de todo el proceso realizado es:

Fecha inicio	Fecha Fin	Actividad
11/12/2015	23/12/2015	Búsqueda de los distintos sistemas de bases de datos NoSQL así como elección de uno de ellos.
24/12/2015	23/12/201	Búsqueda de distintas bases de datos documentales y elección de varios candidatos y comienzo del despliegue de la aplicación war "Hello World".
03/01/2016	08/01/2016	Elección de un sistema NoSQL definitivo para su utilización a lo largo del proyecto. Finalización de la aplicación web.
11/01/2016	11/01/2016	Entrega de material
19/01/2016	20/01/2016	Corrección de pequeños errores en el documento entregado anteriormente

21/01/2016	06/02/2016	Estudio de las tecnologías que se van a emplear durante la implementación.
11/02/2016	18/02/2016	Creación de la estructura del proyecto para poder incorporar todas las tecnologías necesarias.
21/02/2016	26/02/2016	Corrección de la estructura para incorporar un patrón de diseño nuevo.
29/02/2016	28/03/2016	Desarrollo del colector junto con todos los métodos necesarios para la integración entre Solr y el colector.
04/04/2016	19/04/2016	Desarrollo del primer decorador junto con todos los métodos necesarios para la integración entre Solr y el decorador.
20/04/2016	26/04/2016	Desarrollo del segundo decorador. Proceso más corto que con el anterior decorador ya que los métodos para emplear Solr ya se encontraban desarrollados.
27/04/201	28/04/2016	Incorporación de los frameworks, como bootstrap o los JavaScripts necesarios, como JQuery
29/04/2016	04/05/2016	Implementación del sistema de búsqueda desde interfaz web (implementación de servlets y paginas jsp).
05/05/2016	13/05/2016	Inicio del documento final incorporando el primer informe y añadiendo nuevos apartados.
14/05/2016	17/05/2016	Aceleración del proceso. Finalización de la interfaz web para la búsqueda. Continuación del documento.

18/05/2016	20/05/2016	Pruebas sobre la aplicación y finalización del documento

Tabla 1: Seguimiento del proceso

Como vemos la planificación mostrada se acerca mucho a lo que finalmente hemos desarrollado, aunque como ya hemos comentado sería necesario realizar algunos cambios en esta para ajustarlo a los cambios del cliente. La diferencia se encuentra en la última parte del proyecto en la que, por diversos motivos, fue necesario acelerar el proceso, empleando todos los días de la semana incluidos sábados y domingos y entre 10 y 12 horas diarias. Debido a este motivo el proyecto se finalizó con mayor antelación de la prevista inicialmente.

4. Elección de un sistema NoSQL adecuado

4.1. Introducción

La aparición en escena de la web 2.0 dejo en duda la efectividad de las bases de datos relacionales. Hasta entonces existían webs con unos pocos contenidos publicados por el propietario de la web por lo que la extensión de las bases de datos no se extendía más allá de unas pocas tablas con relativamente pocas filas cada una.

Con la aparición de aplicaciones, como YouTube, Facebook, Dailymotion, Vine, etc., en las que cualquier usuario podía subir su propio contenido, el tamaño de las bases de datos se vio incrementado notablemente dejando al descubierto los problemas de escalabilidad que sufrían los sistemas relacionales.

Había que buscar una solución a estos inconvenientes, surgiendo así los sistemas NoSQL.

4.1.1. ¿Qué es un sistema NoSQL?

NoSQL o Not Only SQL (No Solo SQL), son un enfoque hacia la gestión de datos y el diseño de base de datos que es útil para grandes conjuntos de datos distribuidos, orientada a solucionar los problemas de escalabilidad y rendimiento, al acceder numerosos usuarios de manera concurrente, que supone el Big Data para los cuales los sistemas relacionales no fueron diseñados.

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

Como ya hemos comentado NoSQL significa Not Only SQL, es decir, que no prohíbe un lenguaje estructurado de consultas. Algunos sistemas son totalmente no relacionales, sin embargo, otros evitan ciertas funcionalidades relacionales como puede ser la utilización de tablas fijas y operaciones conjuntas, por ejemplo, la operación JOIN.

4.1.2. Diferencias con una base de datos relacional

NoSQL está orientado al uso de base de datos, al igual que ocurre con los sistemas SQL, sin embargo, estos cuentan con numerosas diferencias que en función de la situación ante la que nos harán decantarnos por un sistema u otro. Las diferencias de NoSQL frente a SQL son:

- No utilizan SQL como lenguaje de consultas: Por poner algunos ejemplos, Cassandra emplea el lenguaje CQL, MongoDB utiliza JSON, HyperGraphDB usa Java, etc.

- No emplea estructuras fijas como tablas para el almacenamiento de tablas: Más adelante veremos los distintos tipos de estructuras que emplean las bases de datos NoSQL.

- No permiten operaciones JOIN: El hecho de contar con una cantidad de datos exagerada provoca que estas operaciones puedan ser muy costosas computacionalmente debido a su complejidad.

4.1.3. Ventajas de NoSQL frente a SQL

Los sistemas NoSQL cuentan con numerosas ventajas frente a los SQL. Esas ventajas son:

- Arquitectura distribuida: Las bases de datos relacionales están almacenadas en una única máquina, pero los sistemas NoSQL permite estar almacenadas en diversas maquinas empleando tablas Hash distribuidas.

- Mayor velocidad: Muchos de estos sistemas realizan operaciones directamente sobre la memoria principal y solo vuelcan sus datos a disco cada cierto tiempo. Esta característica está presente en muchas de estas bases de datos, pero hay que puntualizar que el hecho de ser sistemas más recientes hace que muchos de ellos no estén lo suficientemente maduros por lo que la velocidad no es la óptima.

- Escalabilidad: Como ya hemos comentado anteriormente la escalabilidad vertical fue una de las necesidades que pretendía cubrir en NoSQL. En cuanto a la escalabilidad horizontal, permiten la adición de nuevos nodos teniendo que indicar cuales son los nodos que están disponibles.

- Se ejecutan en máquinas con pocos recursos: El costo computacional de estos sistemas es muy reducido por lo que se necesita máquinas tan potentes, lo que se puede traducir también en un coste menor.

  El hecho de poderse ejecutar también en clústeres, provoca que pueda reunirse la potencia computacional de varias máquinas reduciendo así también el coste.

- Puede manejar enormes cantidades de datos: Esto es gracias a la utilización de una estructura distribuida.

- No generan cuellos de botella: Uno de los principales problemas con los que contaba SQL es que necesitan transcribir cada sentencia para ser ejecutada y en casos de sentencias complejas este problema se amplifica. Esto es un punto de entrada común por lo que ante un número elevado de peticiones puede ralentizar el sistema.

- Existencia de diferentes sistemas: Esto permite una gran flexibilidad a la hora de elegir una base de datos que se adapte correctamente a un determinado proyecto.

4.1.4. Desventajas de NoSQL frente a SQL

Ya hemos citado las ventajas que suponen los sistemas NoSQL, pero esto no es la panacea ya que también cuenta con desventajas:

- Perdida de datos: Uno de los puntos a favor que hemos comentado anteriormente del NoSQL era la velocidad y para ellos ciertas operaciones se pueden realizar sobre la memoria principal, pasando a ser almacenadas en disco de forma no instantánea. Ante un apagón o cualquier fallo en el sistema podría perder operaciones de escritura y afectar a la consistencia de la propia base de datos.

- Esto se puede solucionar permitiendo que una operación de escritura haya de realizarse en más de un nodo antes de darla por válida. A pesar de esta solución siempre existirá el riesgo.

- Sistemas poco maduros: Muchos de los sistemas NoSQL tienen un tiempo de vida muy corto por lo que podrán contar con problemas de estabilidad frente a la gran madurez de la mayoría de las bases de datos SQL.

- Falta de experiencia: Los sistemas NoSQL no están completamente introducidas dentro del mundo universitario y son sistemas mucho más jóvenes por lo que es difícil encontrar expertos.

- Problemas de compatibilidad: Una de las ventajas es la existencia de numerosos sistemas para elegir el que mejor se adapta a tus necesidades, pero esto tiene un punto en contra y es los problemas de compatibilidad entre las distintas bases de datos. Cada una tiene su propia API, su propia interfaz de consultas y muy pocos estándares en común por lo que cambiar de un sistema a otro no es tan sencillo.

4.2. Tipos de bases de datos NoSQL

Existen numerosos sistemas NoSQL, más de 225, por lo que primero nos decidiremos por el tipo que mejor se adapte a nuestras expectativas. Estos tipos se diferencian unos de otros por la manera de almacenar la información. Estos son:

4.2.1. Bases de datos clave – valor

Las bases de datos clave valor son los sistemas más sencillos de almacenar la información. Gracias a la simplicidad de la estructura tiene un alto rendimiento en lo que a escalabilidad se refiere.

Estos sistemas están formados por pares clave – valor. En este caso el valor, es decir, la información está directamente relacionada con una clave que lo identifica. Esta clave es única, lo que permite la recuperación de información de forma muy rápida. El valor puede estar estructurado según las necesidades en forma de hash o JSON o un objeto binario

Estos sistemas son muy eficientes tanto en lectura como escritura, pero esta última solo se puede realizar directamente sobre la clave, no sobre el valor. Tampoco son útiles para operaciones complejas con los datos. Solo soportan operaciones simples de creación, lectura, actualización y eliminación.

Las bases de datos clave – valor se emplean para almacenar opciones de configuración, información de sesión, videojuegos o comercio electrónico. Son muy útil para el almacenamiento temporal de información que deba persistir en el tiempo o a través de procesos concatenados.

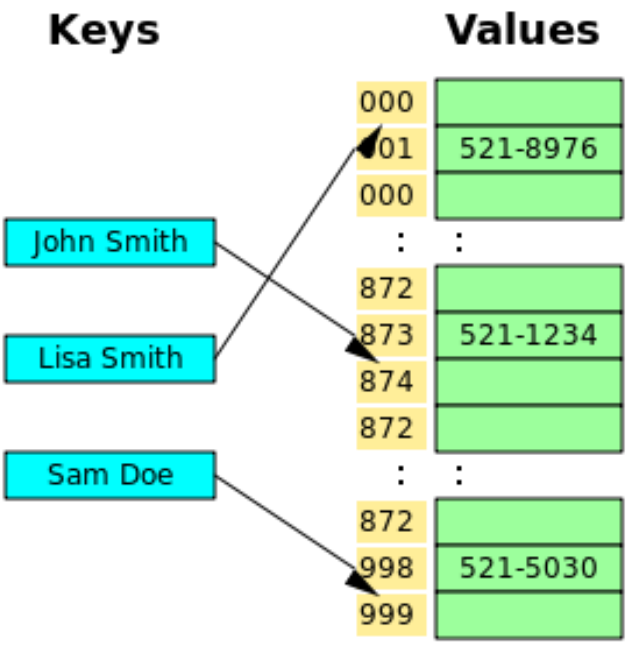


Imagen 1: Ejemplo gráfico de una base de datos calve - valor

4.2.2. Bases de datos documentales

El modelo de bases de datos documental se basa en la idea de que cada archivo o documento contiene información sobre un objeto concreto.

El concepto más importante y para poder entender las bases de datos documentales es el propio concepto de documento. Estos documentos encapsulan y codifican datos o información siguiendo algún formato estándar. Entre estos formatos se encuentran algunos como XML, YAML, JSON, BSON e incluso formatos binarios como FDF o documentos de Microsoft Office.

Cada documento está identificado por una clave única que además de realizar búsqueda por clave - valor, permite realizar consultas más avanzadas sobre el documento. Esta calve se compone de una simple cadena. En algunos casos se trata de una dirección URI que identifica el documento.

En estos sistemas cada objeto dentro del documento puede contener información sobre otros sub-objetos que guardan alguna relación. Gracias a esto se puede crear asociaciones y relaciones dentro de un mismo documento sin necesidad de tablas relacionales como ocurría en las bases de datos SQL. Esto se traduce en un ahorro de recursos al acceder a información relacionada ya que evita tener que pasar por varias tablas o esquemas de forma simultanea para poder llegar hasta ese sub-objeto.

Como ya hemos comentado la información se almacena en formato ASCII que carece de tipología por lo que no es necesario definir campos ni tipos de valor. La morfología de la información está creada en tiempo real sin predefinir las características de la información que vamos a almacenar. Esto se traduce en que en caso de que queramos añadir nueva información podremos hacerlo directamente sin tener que informar de esos cambios que se van a realizar ni indicar que tipo de dato es el que hemos añadido.

Las operaciones sobre una base de datos documental se realizan mediante un API que permite buscar objetos por clave, rango o cualquier otra condición empleada en bases de datos relacional sin incluir, como ya se comentó en las diferencias entre SQL y NoSQL, la operación JOIN.

Algunas incluso permiten la definición de índices compuestos, claves únicas, agrupaciones y otras características típicas de las bases de datos relacionales.

Gracias a su potencia y a su flexibilidad y a su potencia, se convierten en el tipo de bases de datos NoSQL más utilizadas. Se emplean en una amplia gama de aplicaciones incluso las que funcionarían sobre bases de datos relacionales. Son adecuadas para prácticamente cualquier situación y especialmente para aplicaciones online donde la variedad de información recolectada puede ser muy amplia.

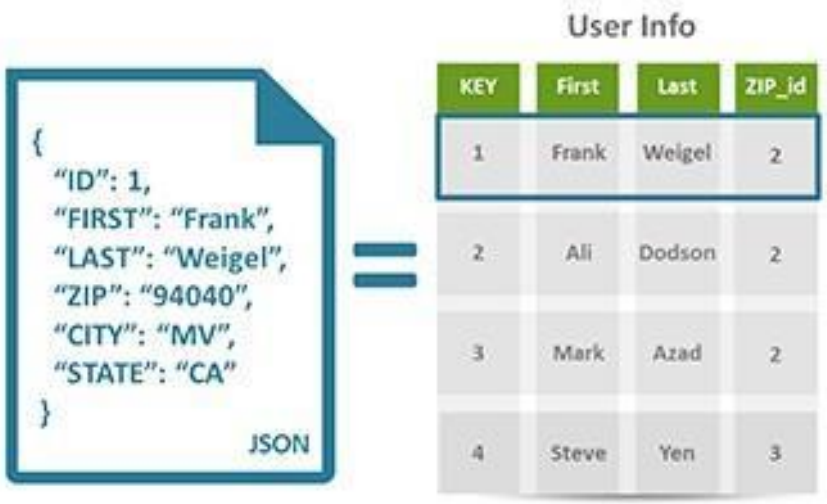


Imagen 2: Ejemplo gráfico de una base de datos documental

4.2.3. Bases de datos en grafo

Las bases de datos en grafos, como su propio nombre indica, hace uso de grafos para representar la información. Para que lo entendamos mejor un grafo está formado por nodos o vértices que representan la propia información unidos por vértices que relacionan unos con otros.

Para recorrerla hace uso de la teoría de grafos por lo que tiene una navegación más eficiente entre relaciones que incluso en el modelo relacional.

Este tipo de bases de datos es ampliamente utilizado en redes sociales gracias a su capacidad relacional de estas. Gracias a ello se puede imitar las relaciones sociales entre individuos, preferencias, gustos u otras personalizaciones del perfil de los miembros.

Los sistemas de datos almacenados en grafos constan de un sistema de minería de datos muy completo que permite hacer buen uso de las relaciones entre entidades e incluso cuantificarlas.

Para poder entender mejor este tipo de bases de datos vamos a utilizar como ejemplo una red de individuos y los comercios online que han visitado y la cantidad de dinero que han gastado y en que lo han gastado. En este caso los nodos son los compradores y están unidos por relaciones de amistad mediante las aristas. Esta red permite conocer quien son amigos directos de alguien y quien es amigo de amigos directos. También tendríamos representados mediante nodos los comercios online y los productos comprados que estarían relacionados con los individuos. De esta manera podemos saber que compra cada usuario y que tiendas visita. Esto puede ser muy interesante para obtener información sobre si personas con relaciones de amistad visitan las mismas tiendas, realizan compras similares o siguen patrones parecidos.

Este tipo de bases de datos es ampliamente utilizado en redes sociales gracias a su capacidad relacional de estas. Gracias a ello se puede imitar las relaciones sociales entre individuos, preferencias, gustos u otras personalizaciones del perfil de los miembros. Y como hemos visto en el ejemplo anterior son muy útiles para para las empresas de marketing ya que la información que nos permite extraer las relaciones puede ser muy útiles y ser utilizada con fines estadísticos o comerciales.

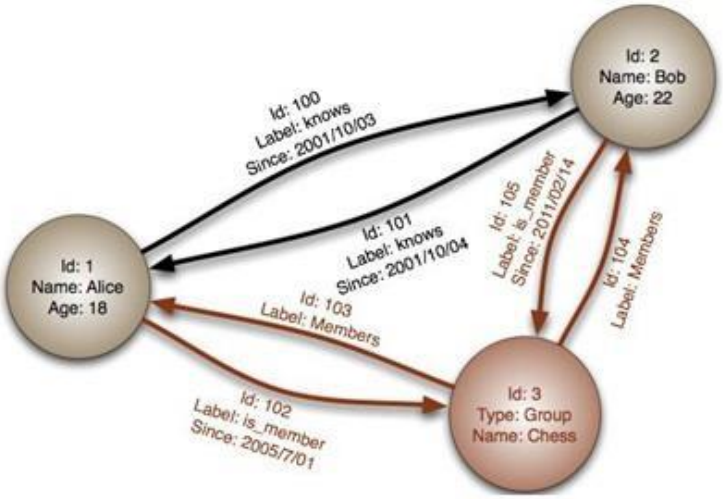


Imagen 3: Ejemplo gráfico de una base de datos basada en nodos

4.2.4. Base de datos orientada a objetos

Las bases de datos orientada a objetos representan la información mediante objetos como se hace en lenguajes de programación orientada a objetos. Almacena métodos y datos. Es muy útil para almacenar objetos complejos.

Estas bases de datos se diseñan para trabajar bien en conjunción con los lenguajes de programación orientados a objetos como lo son Java, C++, Visual Basic.Net, C#...

Cada objeto en estas bases de datos está asociado con:

- Un conjunto de variables que contienen los datos del objeto.
- Un conjunto de mensajes a los que responde. Estos mensajes se refieren a solicitudes entre los objetos.
- Un conjunto de métodos, siendo, cada uno, el código que implementa un mensaje.

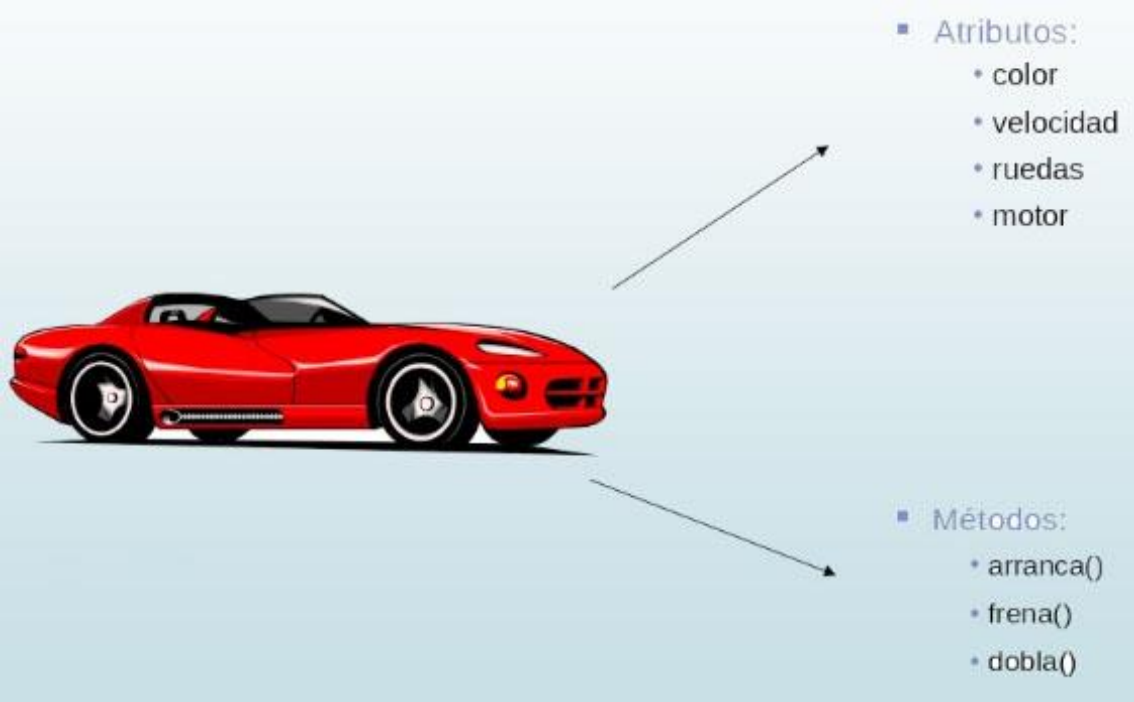


Imagen 4: Ejemplo gráfico de una base de datos orientada a objetos

4.2.5. Base de datos orientadas a columnas

Para poder entender las bases de datos orientadas a columnas tenemos que tener claro dos conceptos: el de base de datos clave – valor y el de las bases de datos relacionales.

Las bases de datos columnares utilizan el esquema clave – valor para almacenar la información, pero permite asignar distintos valores a una misma clave.

Para comprender de donde viene el nombre de bases de datos orientada a columnas emplearemos el concepto de SQL. Las bases de datos relacionales almacenan la información en tablas de forma horizontal, es decir, en las filas de la tabla. En el caso de las bases de datos que estamos tratando la información se almacena de forma vertical, es decir, en columnas. Para ver mejor esta diferencia fijémonos en las dos siguientes imágenes:

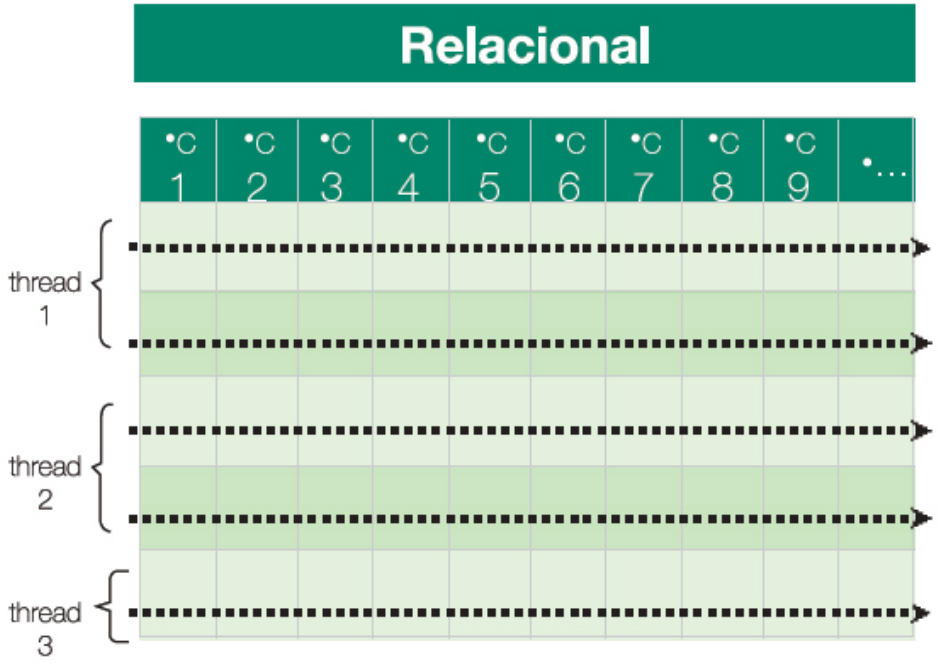


Imagen 5: Base de datos relacional basada en filas

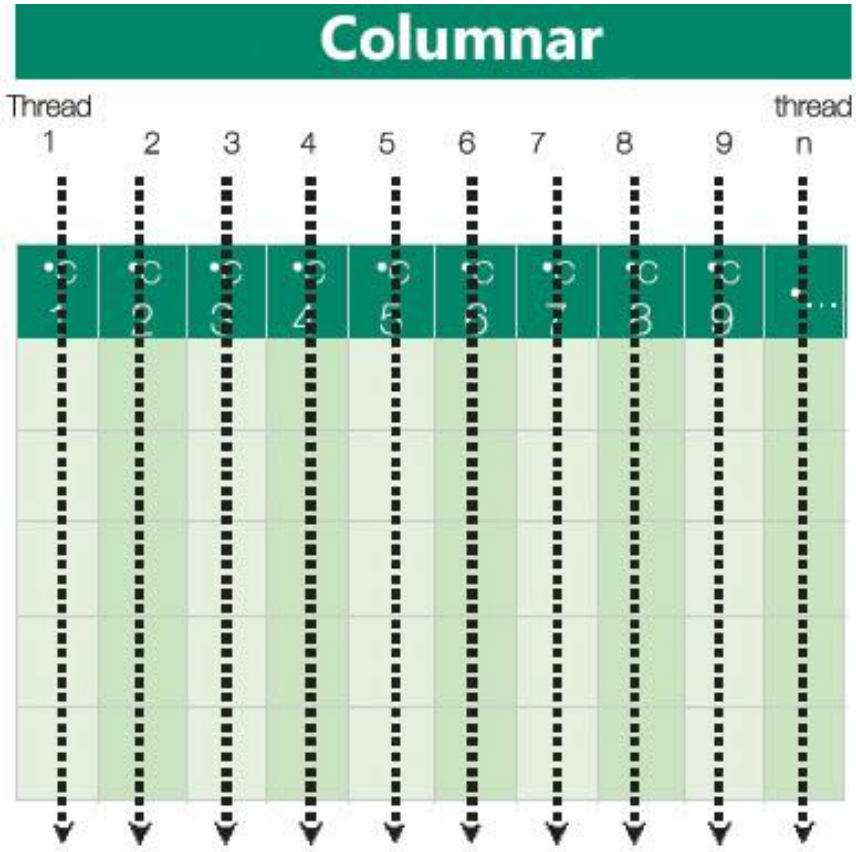


Imagen 6: Base de datos columnar basada en columnas

Las filas contenidas en esta base de datos pueden variar ya que hereda las propiedades de las bases de datos calve – valor. Las tablas de esta base de datos deben ser declarativas por lo que debemos definir su esquema para trabajar los datos.

Para ver la utilidad de estos sistemas pondremos como ejemplo una base de datos que almacena datos de personas y en el campo dos de cada columna, es decir, todos los nombres estarán almacenados en la segunda fila, lo que nos permite acceder de forma rápida y sencilla a ese campo y obtener los nombres de todos. Esto la hace especialmente útil en consultas analíticas.

4.2.6. Bases de datos multi-modelo

Las bases de datos multi-modelo son aquellas que son capaces de organizar los datos en diferentes tipos de estructuras. Por ejemplo, una de estas bases es OrientDB, que soporta grafos, documentos, calve – valor y orientada a objetos.

4.2.7. Elección de un tipo de base de datos

Para poder elegir un tipo de base de datos primero vamos a realizar una tabla resumen comparativa de estos:

Modelo de datos	Formato	Características	Aplicaciones	Ejemplos de bases de datos
Documental	Similar a JSON (JavaScript Object Notation), BSON, XML…	Intuitivo Manera natural de modelar datos cercana a la programación orientada a objetos Flexibles, con esquemas dinámicos Reducen la complejidad de acceso a los datos	Se pueden utilizar en diferentes tipos de aplicaciones debido a la flexibilidad que ofrecen	ElasticSearch - www.elastic.co/-products/elasticsearch Solr - http://lucene.apache.org/solr/ MongoDB - www.mongodb.org CouchDB - http://couchdb.apache.org/
Grafo	Nodos con propiedades (atributos) y relaciones (aristas)	Los datos se modelan como un conjunto de relaciones entre elementos específicos Flexibles, atributos y longitud de registros variables Permite consultas	Redes sociales, software de recomendación, geolocalización, topologías de red …	Neo4J - http://neo4j.com/ Infinite Graph - http://www.objectivity.com/products/infinitegraph/

		más amplias y jerárquicas		
Clave-Valor	Clave-valor: una clave y su valor correspondiente	Rendimiento muy alto Alta curva de escalabilidad Útil para representar datos no estructurados	Aplicaciones que solo utilizan consulta de datos por un solo valor de la clave	Amazon DynamoDB - http://aws.amazon.com/es/dynamodb/ Redis - http://redis.io/ LevelDB - http://leveldb.org/
Columnar	Columnar: variante que permite más de un valor (columna) por clave	Rendimiento muy alto Alta curva de escalabilidad Útil para representar datos no estructurados	Aplicaciones analíticas Aplicaciones que solo utilizan consulta de datos por un solo valor de la clave	Cassandra - https://cassandra.apache.org/ Hypertable - http://www.hypertable.com/ Amazon SimpleDB - http://aws.amazon.com/es/simpledb/
Orientada a objetos	Similar a la programación orientada a objetos: Java, C#, C++, Visual Basic.Net	Mayor capacidad de modelado Ampliabilidad. Crear nuevos tipos o propiedades a partir de los ya existentes	Trabajar bien en conjunción con los lenguajes de programación orientados a objetos	Objetivity - http://www.objectivity.com/ StarCounter - http://starcounter.com/ Versant - http://www.actian.com/products/operational-databases/versant/

Tabla 2: Comparativa tipos de bases de datos

En esta tabla no comparamos las bases de datos multi-modelo ya que esas dependen de las diferentes estructuras en que es capaz de manejar los datos. Y tendría las características de las anteriores.

En el caso que nos ocupa, tenemos una base de datos almacenada mediante archivos XML que transformaremos en una base de datos NoSQL para más tarde representar esos datos en una interfaz web.

La simplicidad de las bases de datos clave – valor hace que las tengamos que descartar ya que únicamente podremos almacenar un valor por cada clave.

También descartando la base de datos en grafo. Como vemos su aplicación va orientada a redes sociales, software de recomendación, geolocalización, topologías de red … Nuestro propósito no es ninguno de estos por lo que podemos descartar este tipo.

Para las columnares el hecho de que solo utilizan consulta de datos por un solo valor de la clave hace que tengamos que apartarlas de nuestro propósito.

Con respecto a los otros dos tipos, tenemos una gran ventaja si hablamos de bases de datos documentales, estas están escritas en un lenguaje muy similar al que tenemos en nuestros archivos iniciales que nos facilitaría mucho su transformación. Además, estas bases de datos son unas de las más utilizadas y cuentas con varios de los sistemas más maduros y eficientes en cuanto a NoSQL se refiere.

Scofield y Popescu resumieron muy bien las características de varios tipos de bases de datos en una tabla. Entre ellas aparecen varias de las citadas anteriormente. La tabla original de estos autores es:

	Performance	**Scalability**	**Flexibility**	**Complexity**	**Functionality**
Key – Value	High	High	High	None	Variable (none)
Column	High	High	Moderate	Low	Minimal
Document	High	Variable (high)	High	Low	Variable (low)
Graph	Variable	Variable	High	High	Graph theory
Relational	Variable	Variable	Low	Moderate	Relational algebra

Tabla 3: Clasificación – Categorización y Comparación por Scofield y Popescu

Como vemos las bases de datos documentales es de las que mejores características tiene por lo que finalmente nos quedaremos con este tipo para poder realizar nuestra elección final.

4.3. Bases de datos documentales

Como ya hemos indicado las bases de datos documentales son el sistema elegido para nuestro propósito. Dentro de este tipo, podemos encontrar numerosos creadores y productos para poder escoger uno. Vamos a comparar cuatro de los sistemas más empleados.

4.3.1. Conceptos previos

4.3.1.1. JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Está basado en un subconjunto del lenguaje de programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999.

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.

- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como vectores, listas o secuencias.

Para entender mejor el concepto de JSON vamos a ver la comparación de un archivo XML con su transformación en este formato.

XML:

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

JSON:

```
{"menu": {
   "id": "file",
   "value": "File",
   "popup": {
     "menuitem": [
       {"value": "New", "onclick": "CreateNewDoc()"},
       {"value": "Open", "onclick": "OpenDoc()"},
       {"value": "Close", "onclick": "CloseDoc()"}
     ]
   }
```

```
  }
}
```

Tenemos un elemento menú con id “file”. Dentro de este un popup con tres elementos de menú “menuitem”. Este podría ser un menú simplificado que encontramos en muchas aplicaciones para crear un nuevo documento, abrirle o cerrarle.

Como vemos es muy simple comprender como almacena la información JSON con este simple ejemplo.

4.3.1.2. BSON

BSON (Binary JSON) es un formato de intercambio de datos usado para su almacenamiento y transferencia. Es una representación binaria de estructuras de datos y mapas.

La forma de un archivo BSON es exactamente igual a un JSON por lo que el ejemplo empleado en el apartado anterior de JSON nos sirve también para este.

Aunque JSON y BSON sean muy similares hay alguna pequeña diferencia entre ellos. BSON está diseñado para tener un almacenamiento y velocidad más eficiente. Los elementos largos contienen el campo “tamaño” para facilitar su escaneo, lo que provoca que en algunos casos BSON use más espacio en memoria que JSON.

4.3.1.3. AGPL

AGPL es el acrónimo de Affero General Public License y es una licencia copyleft derivada de la licencia GNU diseñada para asegurar la cooperación con la comunidad en el caso de software que corra en servidores de red.

A diferencia de la licencia GNU GPL, añade una clausula nueva que obliga a distribuir el software en caso de ejecutarse sobre la red.

El uso de AGPL v3 es recomendado por la Free Software Foundation (Fundación para el software libre) para cualquier servicio que corra sobre la red.

4.3.1.4. Apache License 2.0

La licencia Apache License está creada por Apache Software Foundation en 2004. Esta licencia obliga a la distribución del copyrigt y el disclaimer, pero no obliga a la redistribución del software en las versiones modificadas. Es decir, permite al usuario la libertad para modificar, distribuir, distribuir modificaciones del software, pero puede hacerlo bajo la licencia que el desee con la única condición de reconocerles como creadores de ese software.

Cualquier software de Apache Software Foundation está desarrollado bajo esta licencia.

4.3.1.5. Sharding

El sharding es un tipo de particionamiento de grandes bases de datos en fragmentos más pequeños Estos agrupamientos no se realizan de forma aleatoria, sino que se realiza de forma que tenga sentido y que permita un direccionamiento más rápido. Por ejemplo, organizar los usuarios en una base de datos por nombre encontrándose, en una determinada tabla, los usuarios cuyo nombre va de la A hasta la E.

Estas agrupaciones están localizadas en tablas en diferentes bases de datos y localizaciones físicas. El sharding puede ser a nivel de todas las tablas y no solamente a nivel de una. Por ejemplo, los datos de los usuarios asiáticos se encontrarían en el servidor de Asia, los de los europeos en Europa y así sucesivamente haciendo que el acceso sea mucho más rápido.

El sharding mejora la escalabilidad y el rendimiento al agrupar menos datos en tablas más pequeñas por lo que los accesos serán muchos más rápidos. En el ejemplo de la localización se mejora notablemente la latencia.

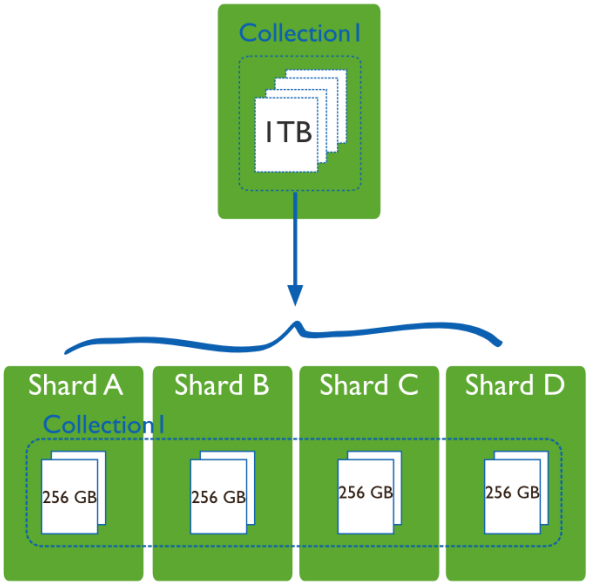


Imagen 7: Descripción gráfica de sharding

4.3.1.6. ACID

ACID es un conjunto de características o propiedades que garantizan que las transacciones en una base de datos son fiables.

En concreto ACID significa Atomicity (Atomicidad), Consistency (Consistencia), Isolation (Aislamiento) y Durability (Durabilidad). Cada una de estas propiedades significa:

- Atomicidad: Cualquier cambio de estado que produce una transacción es atómico. Esta propiedad asegura que una operación se realiza o no se realiza.

- Consistencia: Propiedad que asegura que una transacción no romperá con la integridad de una base de datos, pues respeta todas las reglas y directrices de ésta.

- Aislamiento: Propiedad que asegura que las transacciones no se afectarán entre sí. Dos transacciones podrán interactuar sobre los mismos datos sin afectarse.

- Durabilidad: Una vez que la transacción quedó aceptada no podrá deshacerse, aunque falle el sistema.

4.3.1.7. RESTful

REST (Representational State Transfer) es un modelo de arquitectura de software. REST consiste en una seria de directrices para mejorar las comunicaciones cliente-servidor.

En 1999 se comenzó su desarrollo entorno a los estándares HTTP y URI. Los principios que resumen REST son:

- Todo lo que viaja por internet es un recurso y cada recurso está representado por su formato.

- Cada recurso debe ser identificable por un identificador único accesible a través de la URI.

- Usa métodos HTTP estándar. Estos son GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE, CONNECT.

- Los recursos pueden tener múltiples representaciones. Por ejemplo, podemos tener un recurso XML, pero lo solicitamos en representación JSON.

- Comunicación sin estado: Cada petición al servidor es tratada de modo totalmente independiente. No se mantiene ningún tipo de persistencia. Devuelve el recurso completo en una petición, no podríamos solicitar una segunda parte de ese recurso en otra petición.

RESTful son los servicios web que siguen estos principios.

Las ventajas que nos permite seguir esta arquitectura son:

- Separación del recurso y su representación.

- Visibilidad

- Seguridad

- Escalabilidad

- Rendimiento

4.3.1.8. Indexado

La indexación consiste en registrar ordenadamente datos e informaciones para elaborar un índice facilitando así la búsqueda de información y sección de la información más persistente. Existen dos tipos de índices:

- Índice primario: Es aquel que guarda la clave primaria de un registro identificándolo inequívocamente de los demás.

- Índice secundario: Este tipo de índice se construye sobre el concepto de clave secundaria. Puede existir una única clave secundaria para varios registros, por lo que identifica un conjunto de ellos.

4.3.1.9. Lucene

Apache Lucene es una API de código abierto desarrollado por Apache Software Foundation que permite la recuperación de información. Como cualquier producto de Apache está construido bajo la licencia Apache License 2.0. Podríamos decir que Lucene, básicamente, pone la base para ciertas bases de datos orientadas a documentos creadas por Apache.

Esta API es útil para cualquier aplicación que requiera indexado y búsqueda a texto completo por lo que es ampliamente utilizado por su implementación de motores de búsqueda.

Lucene es independiente del formato del fichero como pueden ser textos en PDF, documentos de Word, etc. Cualquier archivo del que se puede extraer información puede ser indexado.

Las principales características de Lucene son:

- Más de 150 GB/hora en un hardware moderno.
- Pequeños requisitos de memoria principal (solamente 1 MB).
- Indexación incremental tan rápida como la indexación por lotes.
- Tamaño del índice aproximadamente un 20-30% del tamaño del texto indexado.
- Búsqueda por ranking.
- Gran cantidad de tipos de consultas.
- Búsqueda por campos.
- Ordenación por cualquier campo.
- Permite actualización y búsqueda simultánea.
- Motores de almacenamiento configurable.

- Múltiple búsqueda de índices con combinación de resultados.
- Autocompletado.

4.3.2. ElasticSearch

Imagen 8: Logo ElastacSearch

ElasticSearch es un servidor de búsqueda basado en Lucene. El proyecto fue iniciado en 2004 pero no fue hasta 2008 cuando vio la luz la primera versión. Esta construido bajo la licencia Apache License 2.0. Las principales características de este sistema son:

- Datos en tiempo real: Este sistema dispone de los cambios en tiempo real.
- Alta disponibilidad: ElascticSearch elimina os nodos que estén fallando y reorganizarse a sí mismos para asegurar que los datos sigan accesibles.
- Indexado: Puede alojar varios índices y permite consultar estos de manera independiente. También se puede definir índices online.
- Búsqueda full-text: ElascticSearch se basa en Lucene para sus búsquedas soportando, geolocalización, autocompletado…
- Uso de documentos JSON.
- Implementa las propiedades ACID por lo que no habrá conflictos ante escrituras simultaneas.
- API RESTful: ElascticSearch proporciona una API RESTful sobre JSON.

4.3.3. MongoDB

Imagen 9: Logo MongoDB

MongoDB guarda su estructura de datos en documentos tipo BSON (Binary JSON). El desarrollo de MongoDB empezó en octubre de 2007 por la compañía de software 10gen y en 2009 fue lanzada al mercado bajo la

licencia de código abierto AGLP. Las características principales de este sistema son:

- Consultas Ad hoc: MongoDB soporta la búsqueda por campos, consultas de rangos y expresiones regulares. Las consultas también pueden devolver una función JavaScript.

- Indexación: Cualquier campo en un documento puede ser indexado.

- Replicación: MongoDB soporta el tipo de replicación primario - secundario. El primario puede ejecutar comandos de lectura y escritura. Los secundarios replican los datos del primario y sólo se pueden usar para lectura o para copia de seguridad, pero no se pueden realizar escrituras.

- Balanceo de carga: MongoDB permite el sharding.

- Almacenamiento de archivos: Permite ser utilizada como sistema de archivos gracias al balanceo de carga y la replicación de datos. Esta base de datos tiene funciones para la manipulación de archivos y contenido a los desarrolladores. En un sistema con múltiples servidores, los archivos pueden ser distribuidos y copiados entre los mismos varias veces y de una forma transparente, de esta forma se crea un sistema eficiente que maneja fallos y balanceo de carga.

- Agregación: El sistema MongoDB proporciona un framework de agregación que permite realizar operaciones similares a la “GROUP BY” de SQL Este framework está construido como un pipeline en el que los datos van pasando a través de diferentes etapas en los que son modificados, agregados, filtrados y formateados hasta obtener el resultado deseado. También proporciona una función MapReduce que puede ser utilizada para el procesamiento por lotes de datos y operaciones de agregación.

- Ejecución de JavaScript del lado del servidor: Permite realizar consultas utilizando JavaScript siendo ejecutadas directamente en la base de datos.

Pero como cualquier sistema, este no es perfecto. Los principales problemas de MongoDB son:

- No implementa las propiedades ACID. Esto deriva en otros cuatro puntos:

    - Problemas de consistencia: Las lecturas pueden ver versiones obsoletas de documentos y pueden devolver lecturas de datos incorrectos.

    - Bloqueo a nivel de documento: Ante todas y cada una de las operaciones de escritura, MongoDB bloquea la base de datos a nivel de documento. Solo se permiten lecturas concurrentes.

- Escalabilidad: Como ya hemos dicho NoSQL surge para solventar los problemas de escalabilidad de SQL, pero como cualquier sistema todo tiene su límite y la de MongoDB es de 100GB. A partir de esta cuota sufre problemas de rendimiento.

MongoDB es una de las bases de datos más utilizadas actualmente y de las mejores situadas, aparte de contar con un gran soporte y cantidad de tutoriales.

4.3.4. Solr

Solr es un motor de búsqueda de código abierto basado, al igual que ElasticSearch, en Lucene. El proyecto se comenzó en 2004. Actualmente está construido bajo la licencia Apache License 2.0. Las características de este sistema son:

Imagen 10: Logo Apache Solr

- API RESTful: Solr cuenta con una API RESTful sobre XML y JSON.
- Caches internas: Este sistema incluye caches internas para devolver con mayor rapidez los resultados.
- Administración web: Incluye una administración web que permite consultar estadísticas de rendimiento, incluyendo el uso de cache, realizar búsquedas mediante un formulario, navegar por los términos más populares del índice, visualizar un desglose detallado de las matemáticas de puntuación y las fases de análisis de texto.
- Manejo de documentos ricos: Permite el manejo de estos documentos como pdf, docx…
- Geolocalización: Soporta geolocalización en los documentos, de modo que podemos realizar búsquedas con un filtro de distancia.
- Soporta como formato de respuesta CSV.
- Podemos restringir la navegación de facetas por rangos numéricos.
- Incorpora autocompletado
- Búsqueda de documentos similares: Permite buscar documentos similares.
- Spell check: Dispone de un plugin para revisión gramatical.
- Emplea las consultas básicas de Lucene con ciertas mejoras.

- Permite la configuración de la indexación y recuperación de documentos mediante ficheros de configuración XML.

4.3.5. CouchDB

CouchDB pertenece a la compañía Apache Software Foundation. Se lanzó en el año 2005 bajo la licencia Apache License 2.0. Utiliza JSON para almacenar los datos. Las principales características de CouchDB son:

Imagen 11: Logo CouchDB

- Respeta las características ACID lo que se traduce en poder manejar numerosos escritores y lectores.

- Vistas e índices Map/Reduce: Los datos almacenados se estructuran por medio de vistas. En CouchDB, cada vista se construye por medio de una función JavaScript que actúa como la mitad Map de una operación map/reduce. La función recibe un documento y lo transforma en un único valor, retornándolo. CouchDB puede indexar vistas y mantener actualizados esos índices a medida que se agregan, eliminan o actualizan documentos.

- Replicación: En el caso de CouchDB múltiples replicas pueden tener cada una sus propias copias de los mismos datos, modificarlas y luego sincronizar eses cambios en un momento posterior. Respetar las características ACID permite que todos los cambios se sincronicen al realizarlos.

- Interfaz REST: CouchDB sigue los principios REST.

- Consistencia eventual: CouchDB garantiza consistencia eventual para ofrecer tanto disponibilidad como tolerancia a la partición

- Hecha para operar offline: CouchDB es capaz de funcionar en dispositivos que se encuentran sin conexión y sincronizar los cambios realizados cuando este se encuentre online de nuevo.

4.3.6. Comparativa y decisión final

Ya hemos visto una pequeña explicación de los cuatro sistemas candidatos para nuestro proyecto. De entre todos los sistemas documentales hemos elegido estos cuatro por su gran extensión, su utilización siendo las más empleadas y su gran soporte. Para elegir uno necesitaremos comparar las características de unos y otros:

Nombre	CouchDB	ElascticSearch	MongoDB	Solr
Descripción	Un almacenamiento documental inspirado por Lotus Notes	Un motor de búsqueda empresarial moderno basado en Apache Lucene	Una de las bases de datos más populares	Un motor de búsqueda empresarial ampliamente utilizado basado en Apache Lucene
Sitio web	couchdb.apache.org	www.elastic.co/-products/-elasticsearch	www.mongodb.org	lucene.apache.org/-solr
Documentación técnica	wiki.apache.org/couchdb	www.elastic.co/-guide	docs.mongodb.org/manual	lucene.apache.org/-solr/-documentation.html
Desarrollador	Apache Software Foundation	Apache Software Foundation	MongoDB. Inc	Apache Software Foundation
Fecha de lanzamiento	2005	2010	2009	2004
Versión actual	1.6.1, Septiembre 2014	2.1.1, Diciembre 2015	3.2.0, Diciembre 2015	5.4.0, Diciembre 2015
Licencia	Open Source, Apache License 2	Open Source, Apache License 2	Open Source, AGPL v3	Open Source, Apache License 2
Puntuación según la web http://db-engines.com/	25.04	76.57	301.39	79.15
Ranking según la web http://db-engines.com/	#25	#13	#4	#12
Sistemas operativos compatibles	Android BSD Linux OS X Solaris Windows	Todos los sistemas operativos con una VM Java	Linux OS X Solaris Windows	Todos los sistemas operativos con una VM Java y un contenedor de Servlets

Esquema de datos	Esquema libre	Esquema libre	Esquema libre	Esquema libre
Tipado	no	si	si	si
Indexado secundario	si	si	si	si
APIs y otros métodos de acceso access methods	API RESTful HTTP/JSON	API Java API RESTful HTTP/JSON	Protocolo propietario usando JSON	API Java API RESTful HTTP
Lenguajes de programación soportados	C C# ColdFusion Erlang Haskell Java JavaScript Lisp Lua Objective-C OCaml Perl PHP PL/SQL Python Ruby Smalltalk	.Net Erlang Java JavaScript Perl PHP Python Ruby Scala	Actionscript C C# C++ Clojure ColdFusion D Dart Delphi Erlang Go Groovy Haskell Java JavaScript Lisp Lua MatLab Perl PHP PowerShell Prolog Python R Ruby Scala Smalltalk	.Net Erlang Java JavaScript any language that supports sockets and either XML or JSON Perl PHP Python Ruby Scala
Script del lado del servidor	Funciones en JavaScript	no	JavaScript	Plugins Java
ACID	si	si	no	si
Triggers	si	si	no	si

Método de particionamiento	Sharding	Sharding	Sharding	Sharding
Método de replicación	Replicación maestro – maestro Replicación maestro - esclavo	si	Replicación maestro - esclavo	Distribuida (vía Apache Zookeeper) Replicación maestro – esclavo
MapReduce	si	no	si	no
Consistencia	Consistencia eventual	Consistencia eventual	Consistencia eventual Consistencia inmediata	Consistencia eventual
Concurrencia	si	si	si	si
Durabilidad	si	si	si	si

Tabla 4: Comparativa bases de datos documentales

Los cuatro sistemas candidatos son quizá los sistemas NoSQL más empleados y que cuentan con un mayor soporte, pero debemos quedarnos únicamente con uno, aunque cualquiera de ellos podría ser válidos.

Los datos anteriores has sido extraídos de la página db-engines.com. Esta web cuenta con un ranking de las bases de datos más populares existentes (hay que tener en cuenta que también incluye bases de datos SQL Y otras NoSQL), siendo estas cuatro las que más arriba se encuentran en el ranking dentro de las bases de datos documentales.

La que se encuentra en una posición superior es MongoDB, pero la eliminaremos en favor de los sistemas que utilizan Lucene, ya que es uno de los mejores motores de búsqueda en la actualidad, mejor que el que utiliza MongoDB. Los otros sistemas candidatos si están basados en Lucene.

Las diferencias entre los tres lenguajes son mínimas. Todos cuentan con una API RESTful, aunque solamente ElasticSearch y Solr cuentan con una API Java.

En cuanto a los lenguajes de programación, CouchDB es la que más soporta de los tres, pero el único que nos interesaría en este caso en Java incluido en todos ellos.

En cuanto al resto de diferencias como MapReduce, solo CouchDB cuenta con ello. Esta opción puede ser muy interesante si el manejo de datos es muy grande pero no es nuestro caso por lo que no nos obliga a tomar una opción final solamente a tenerlo en cuenta.

La siguiente diferencia si nos hará tomar una decisión importante como es descartar ElasticSearch. Contar con script del lado del servidor nos permitirá generar páginas dinámicas a partir de las peticiones facilitando el acceso a la base de datos.

De entre los dos sistemas restantes el resto de características son prácticamente iguales por lo que centraremos nuestra decisión en puntos como la posición en el ranking con claro ganador por parte de Solr. Otro punto a favor de Solr es que, al ser una de las bases de datos más extendidas, cuenta con un gran soporte, mayor al de CouchDB.

La decisión final será emplear Solr, aunque cualquiera de los otros sistemas hubiera sido válido.

4.4. Conclusión

Como hemos podido comprobar existe una gran variedad en cuanto a bases de datos NoSQL se refiere. Contamos con muchas más opciones que con SQL. Esto permite que cualquier aplicación pueda adaptarse a una base de datos de este tipo solventando muchos de los problemas que tendríamos en caso de usar SQL. Esto no significa que SQL no sea indicado para muchos sistemas, de hecho, sigue siendo más utilizado en la actualidad, pero si lo hace inútil para otros muchos.

Incluso para un único proyecto tenemos la posibilidad de elegir numerosas bases de datos, aunque debemos tener cuidado, ya que una vez elegido una, la migración solamente es sencilla entre bases de datos del mismo tipo.

5. Diseño de a aplicación

Nuestra aplicación consta de dos tipos de elementos bien diferenciados:

- Colectores: Serán los encargados de insertar la información contenida en los archivos XML en la máquina de búsqueda Solr.

- Decoradores. Se encargarán de adaptar la información almacenada en la base de datos de la forma deseada. En el caso de los decoradores que necesitamos desarrollar se basan en la modificación o inserción de los identificadores adecuados.

El funcionamiento de nuestra aplicación se muestra en el siguiente diagrama.

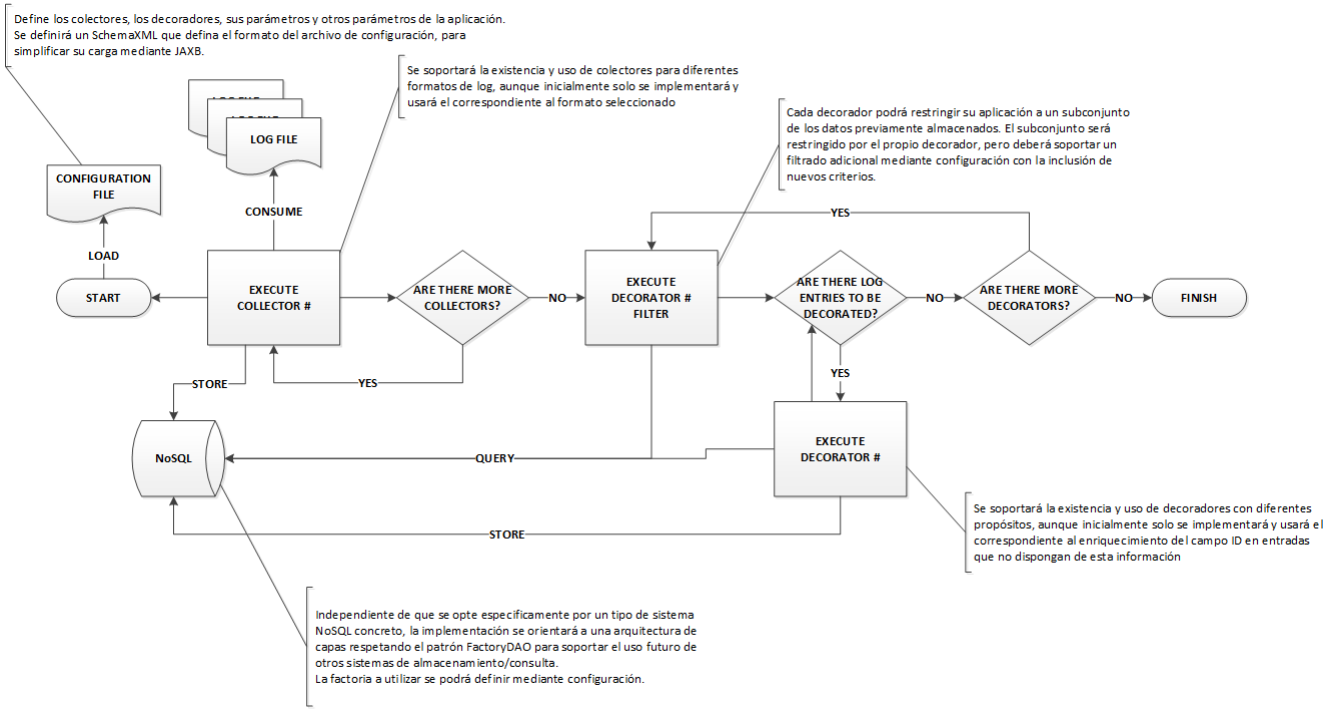


Imagen 12: Diseño de aplicación para carga enriquecido

La aplicación puede contar con un número variable de colectores y decoradores para la carga enriquecida. Estos pueden variar entre 0 e infinito y la ejecución de estos tiene que ser automática. En primer lugar, se ejecutarán los colectores y a continuación los decoradores en el mismo proceso.

Veamos primero cual es la estructura de los XML a partir de su XSD. El diagrama generado por Eclipse de este es:

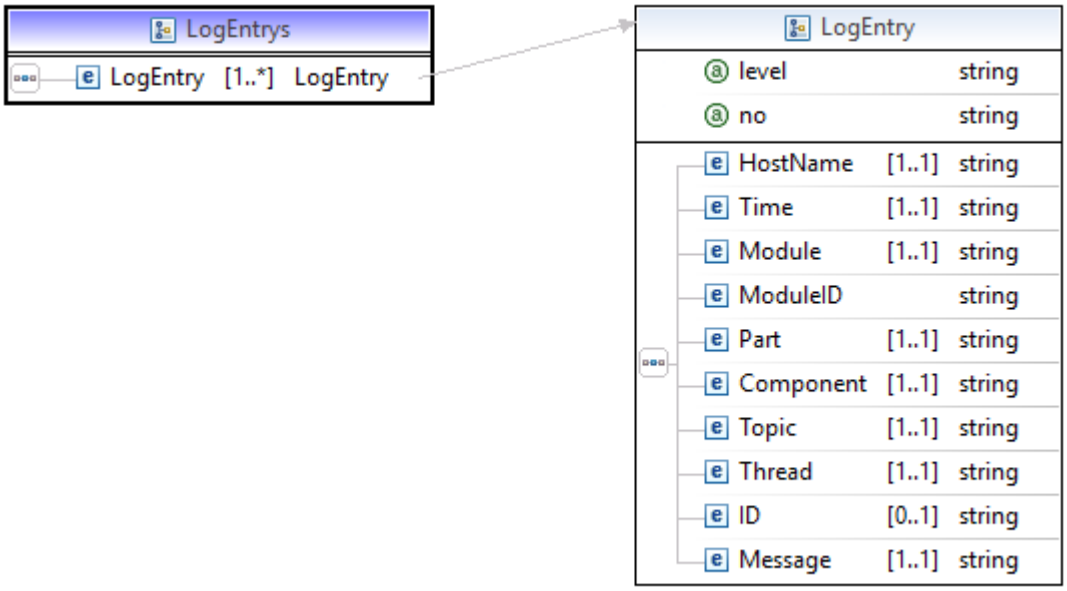


Imagen 13: Esquema de los XML

Como vemos tenemos un elemento raíz (LogEntrys) que no se encuentra en los XML originales, pero en el siguiente apartado explicaremos más a fondo en funcionamiento y el motivo de este elemento raíz. Ese elemento raíz está formado por otros elementos con etiquete LogEntry cuyos elementos a su vez, son:

- HostName
- Time
- Module
- ModuleID
- Part
- Component
- Topic
- Thread
- ID
- Message

También cuenta con dos atributos:

- level
- no

En nuestro caso construiremos un solo colector que recopilará la información de los archivos XML para escribirlos en la base de datos que hemos elegido anteriormente. La lectura de los ficheros XML se realizará mediante JAXB, cuya utilización podemos ver en el anexo V. Su estructura de archivos necesarios para su empleo es:

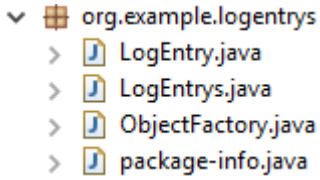


Imagen 14: Estructura JAXB

Los archivos XML se situarán en una carpeta exclusiva para ellos cuya lectura y carga se realizará de uno en uno.

El acceso a la base de datos será necesario realizarla empleando un patrón FactoryDAO.

Una vez, hemos acabado la recolección de datos podemos empezar con la decoración. Los decoradores serán estos:

- Uno de ellos nos permitirá escribir los IDs de ciertas entradas a partir del campo ModuleID. Para verlo más mejor pongamos un ejemplo de ello. Veamos el código siguiente:

```
<LogEntry level="DEBUG" >
   <HostName>MAQUINAVIRTUAL</HostName>
   <Time>19-dic-2014 13:45:59</Time>
   <ModuleID>12345678</ModuleID>
   <Part>FRAMEWORK</Part>
   <Component>WFMEngineWorker</Component>
   <Topic></Topic>
   <Thread>RMI TCP Connection(60)-192.168.88.128</Thread>
  <Message>blab la blab la bla bla</Message>
</LogEntry>
```

```
<LogEntry level="DEBUG" >
   <HostName>MAQUINAVIRTUAL</HostName>
   <Time>19-dic-2014 13:45:59</Time>
   <ModuleID>12345678</ModuleID>
   <Part>FRAMEWORK</Part>
   <Component>WFMEngineWorker</Component>
   <Topic></Topic>
   <Thread>RMI TCP Connection(60)-192.168.88.128</Thread>
   <ID>5747</ID>
  <Message>blab la blab la bla bla</Message>
</LogEntry>
```

En el segundo elemento LogEntry encontramos que este cuenta tanto con un elemento ModuleID y un elemento ID. En el momento en que encontremos algo con estas condiciones todos los elementos cuyo ModuleID sea este contendrá ese ID, por lo que podemos añadirlo a esa entrada. Un ejemplo de candidato a este cambio es el primer elemento LogEntry ya que cuenta con el mismo ModuleID, pero no cuenta con un ID propio.

- El segundo decorador consistirá en buscar todos los elementos LogEntry cuyo Message sea algo similar “MASTER_CHILD_JOBS=[43060]”. Este mensaje lo que nos indica realmente es el ID real de ese documento siendo el ID que con el que anteriormente contábamos, el ParentID, es decir el ID del trabajo padre. Para verlo mejor pongamos también un ejemplo:

```
<LogEntry level="INFORMATIVE" no="76" >
   <HostName>vodafone-broadband.esm.cpqcorp.net</HostName>
   <Time>Feb 11, 2016 9:53:12 AM</Time>
   <Module>mwfm</Module>
   <Part>COMPONENT</Part>
   <Component>WFTransactionHandler</Component>
   <Topic></Topic>
   <Thread>MWFM Worker 4</Thread>
   <ID>43059</ID>
  <Message>MASTER_CHILD_JOBS=[43060]</Message>
</LogEntry>
```

En este caso el verdadero ID es 43060 y el ParentID es 43059. Por lo que el resultado final sería.

```
<LogEntry level="INFORMATIVE" no="76" >
   <HostName>vodafone-broadband.esm.cpqcorp.net</HostName>
   <Time>Feb 11, 2016 9:53:12 AM</Time>
   <Module>mwfm</Module>
   <Part>COMPONENT</Part>
   <Component>WFTransactionHandler</Component>
   <Topic></Topic>
   <Thread>MWFM Worker 4</Thread>
   <ParentID>43059</ParentID>
   <ID>43060</ID>
  <Message>MASTER_CHILD_JOBS=[43060]</Message>
</LogEntry>
```

Estos ejemplos los hemos realizado sobre los archivos XML, pero en realidad esto se realizaría directamente sobre la base de datos después de que esta haya sido cargada completamente.

Cada decorador tendrá dos métodos principales:

- Filtrado de datos: Este será el primero en ejecutar. Obtendremos todos los documentos candidatos a ser modificados.
- Decoración de datos: Realizaremos todas las modificaciones necesarias en los documentos y volveremos a escribirlos en la base de datos con los cambios necesarios realizados.

6. Implementación

Las tecnologías empleadas y de las que dependerá nuestra implementación son:

- Eclipse como plataforma de desarrollo.
- Bases de datos Solr cuya instalación e integración con Tomcat se explica en el anexo III. La creación de colecciones en el anexo IV.
- Tomcat como servidor de aplicaciones cuya instalación se explica en el anexo II.
- JAXB para implementar la lectura de XML cuya integración con Eclipse se desarrolla en el anexo V.
- Java como lenguaje de programación.
- Paginas jsp para la parte web.
- Bootstrap para la presentación.
- Javascript para ciertas funciones de la parte web.
- API reflection para la búsqueda dinámica de clases
- API SolrJ para usar Solr con Java. Su integración en Eclipse se incluye en el anexo VI.

Comencemos por la forma de ejecutar los colectores y decoradores. Ya que la cantidad de estos es susceptible a cambios su búsqueda debe ser automática, por ese motivo todos los colectores se encontrarán en un paquete expresamente creado para ellos, al igual que los decoradores con su propio paquete. Con la API reflection nos ayudaremos a obtener todas esas clases y ejecutarlas a través del nombre de estas. De esta forma obteniendo un Array de String con los nombres de las clases podemos ejecutarlas dinámicamente. De otro modo no sería posible ejecutar una clase únicamente teniendo un String que guarda el nombre de esta.

Continuemos con los detalles importantes de la implementación del colector. La lectura de los XML se realizará archivo a archivo buscando en una ruta predeterminada todos aquellos archivos XML. Los archivos originales no cuentan con un elemento raíz y como hemos comentado en el apartado anterior se necesita un elemento raíz para emplear JAXB por lo que necesitamos añadirle. Para evitar posibles errores en la escritura o posibles paradas inesperadas del sistema o de la aplicación no haremos ninguna de estas modificaciones sobre los originales siendo estos únicamente de lectura. Todas estas modificaciones las originaremos sobre archivos temporales que crearemos al leer los archivos originales y borraremos tras escribir los datos en la base de datos textual Solr. Las líneas extra a incluir serán en el inicio y final del archivo para abrir y cerrar el elemento LogEntrys respectivamente. Es decir, la primera línea del archivo es:

```
<LogEntrys xmlns=\"http://www.example.org/LogEntrys\">
```

Seguido de todo el contenido de los XML originales y finalizando con:

```
</LogEntrys>
```

Otra decisión importante de diseño es el momento en el que debemos realizar un commit en la base de datos. Para este proceso se han realizado pruebas con distintas implementaciones:

- La primera es realizando el commit por cada elemento LogEntry, es decir por lo que equivaldría a un solo documento en la base de datos XML.
- La segunda haciendo el commit por cada archivo XML que leemos.
- Por último, ejecutando el commit al final de todo el proceso de obtención de datos.

Veamos los tiempos de ejecución en cada caso. Estas pruebas se han realizado sobre uno, cinco y doce archivos con una media de 990 entradas, con la base de datos textual vacía y en varias ejecuciones obteniendo un tiempo medio:

Implementación	Tiempo para 1 archivo	Tiempo para 5 archivos	Tiempo para 12 archivos
Primera	25 seg	110 seg	Time Out de servidor
Segunda	3.3 seg	15 seg	29 seg
Tercera	3.2 seg	12 seg	24 seg

Tabla 5: Tiempos de ejecución del colector

Como vemos la primera opción es descartable automáticamente debido a sus altos tiempos de ejecución por lo que nos deja dos opciones. En cuanto a tiempo de ejecución el último es el que mejores tiempos obtiene, pero no son excesivamente bajos con respecto a la segunda opción, aunque la diferencia será cada vez mayor conforme aumente la cantidad de archivos.

Otro aspecto a tener en cuenta es la robustez. En caso de un error en el sistema, un apagado inesperado, todo el proceso realizado con la implementación tres no quedaría guardado y habría que comenzar el proceso de nuevo. En el caso de la segunda implementación solo quedaría inconsistentes los datos del archivo que se estuviera leyendo en ese momento.

La única manera de conseguir diferenciar unas entradas de otras es comparar todos sus campos, ya que no existe una clave única en estos, pero hay un gran problema, una vez introducidos en la base de datos, estas entradas pueden ser candidatas a ser modificadas por lo que la comparación directa campo a campo no sería efectiva, aparte de ser mucho más lenta que comparar solamente un simple campo.

La solución radicaría en crear un campo que contenga la suma de todos los demás campos de la entrada antes de que se realice ninguna modificación de esta. La primera idea puede pasar por crear un String con la concatenación de estos, pero nos encontramos ante otro problema, algunos caracteres son especiales para Solr, creando errores si queremos hacer una query con estos. Eliminar todos estos caracteres especiales requiere conocer primero todos ellos y después reemplazarlos o borrarlos lo que al final hace que tengamos una clave única poco fiable.

Una solución correcta y que además nos haría ahorrar espacio es el empleo de un resumen criptográfico sobre la concatenación de todos los elementos. De esta manera tenemos una cadena aceptable para su almacenamiento.

Dentro de los resúmenes criptográficos tenemos diversas opciones, la primera de ellas y la más simple es MD5. El problema de este algoritmo es su vulnerabilidad ante colisiones, que, aunque la posibilidad es baja, no podemos permitir el riesgo de dejar sin almacenar alguna entrada porque se produzca algún error de este tipo por lo que queda descartado. Las colisiones se solucionan introduciendo SHA, en nuestro caso, SHA-256.

Cada vez que hagamos una nueva inserción con el colector en la base de datos debemos comprobar que el nuevo documento no se encuentre ya insertado. Una opción sería acceder a la base de datos cada vez que leamos un elemento LogEntry en los archivos XML, pero nos encontramos ante un problema de tiempo ya que, ante 1000 nuevos documentos, tendríamos que acceder a la base de datos 1000 veces lo que se hace inconsistente. La solución se encuentra en acceder una única vez a la base de datos, al iniciar la ejecución y guardar todos los identificadores únicos que hemos nombrado anteriormente en una estructura, como, por ejemplo, un ArrayList. De esa manera cada vez comprobemos si un documento se encuentra ya añadido únicamente tendremos que comprobar si está en dicho ArrayList. En caso de que no lo encontremos, tendremos que añadirlo a la base de datos y al ArrayList para futuras comprobaciones evitando que una entrada duplicada en los archivos de entrada, sea guardada. Una vez creado este identificador único no volverá a ser modificado.

Otra decisión de implementación es la indexación en e motor de y aunque lo más correcto no es indexar todos los campos, esta aplicación podrá sufrir nuevas incorporaciones de colectores y decoradores por parte de HP por lo que será necesario dejar todas las puertas posibles abiertas para nuevas implementaciones. Solr cuenta con un gran sistema de indexación siendo los tiempos de inserción despreciables en relación del número de índices que hayamos creado teniendo en cuenta el tiempo que aumentarán las búsquedas posteriores. Un aspecto que tampoco varía enormemente es el tamaño. En la siguiente tabla podemos ver los resultados del tiempo necesario para realizar la recolección de datos de doce archivos XML de 990 entradas de media, así como el espacio ocupado con todos los campos indexados y con todos ellos sin indexar.

	Tiempo	Tamaño
Indexado	29 seg	105.87 MB
Sin indexar	24 seg	93.6 MB

Tabla 6: Tiempo y tamaño indexación

Al ser un requisito necesario que todos los campos estén indexados, no tendremos en cuenta los datos obtenidos para nuestra decisión e indexaremos todos ellos.

Con respecto a los decoradores primero será necesario filtrar para después realizar la decoración de los datos y almacenarlos. Ya hemos explicado en el apartado anterior cual debe ser el comportamiento de cada colector, ahora centrémonos en las decisiones tomadas para la implementación.

En el caso del primero de ellos, es decir, el que busca en el cuerpo del elemento “Message” si comienza por “MASTER_CHILD_JOBS”, es necesario pensar que pasaría si este documento ha sido ya modificado. En este caso volver a hacer los cambios nos llevaría a sobrescribir en la base de datos algo que ya se encontraba guardado lo que se traduce en aumento del tiempo. Por lo que tendremos que comprobar que documentos no tienen un “ParentID”, es decir, aquellos cuyo “ParentID” tenga su valor por defecto, es decir, EMPTY. Una vez realizada la búsqueda bajo estas condiciones podemos realizar los cambios.

En el otro decorador realizado, el que se encarga de encontrar los ModuleID que contengan ya un ID distinto del establecido por defecto, también EMPTY. En este caso, en el filtrado, crearemos tres ArrayList. El primero guardará la lista de todos los documentos que deberemos modificar. Los otros dos guardarán una lista de los id de los modules y el id del documento. Existe una correspondencia entre estos tres, es decir, el elemento en la misma posición de cada ArrayList serán los empleados para hacer la correlación entre ellos.

Para la interfaz web tendremos dos partes bien diferenciadas, una que ejecutará todo el proceso de almacenado y decoración de datos en el motor de busqueda presionando un botón. Una vez pulsamos este botón se iniciará todo el proceso. Debido a que este puede ser un poco largo e impacientar al cliente, se desactivarán los botones, evitando que el usuario inicie nuevas acciones continuadas sobrecargando el sistema. Una vez finalice el proceso nos redirigiremos a una página que nos confirmará la finalización correcta de esta acción.

Por otro lado, tendremos un buscador que nos permitirá buscar por distintos campos. Una opción que podremos seleccionar será si queremos realizar una búsqueda exacta o una búsqueda conteniendo el término. Por ejemplo, si marcamos este checkbox y el ID que deseamos encontrar es 5747 nos mostrará todas las entradas con ID 5747 en una tabla, junto a toda su información. Pero si desmarcamos esta casilla y buscamos con el mismo ID, tendríamos una búsqueda de todos los documentos cuyo ID siga este patrón *5747*. En caso de no encontrar ningún elemento con los

parámetros de búsqueda, simplemente se mostrará un mensaje avisando de este problema junto con la posibilidad de realizar las mismas acciones que en el index.

El objetivo final del sistema de búsqueda es llegar a encontrar los resultados que deseas. Una vez has localizado el trabajo, lo normal es permanecer en esa página revisando todos lo resultados. Incluir un formulario de búsqueda en este apartado es ocupar espacio de pantalla ya que probablemente no se vaya a usar. En caso de que la búsqueda no encuentre nada, si es posible que el cliente desee realizar una nueva. Por ese motivo en caso de que sea nula, la redirección se hará a otra página con esta posibilidad.

7. Funcionamiento de la aplicación

Recordemos que el proyecto consiste en el desarrollo del back-end. El front-end debe ser algo muy simple que nos permita realizar todas las acciones desde esta interfaz. El motivo de haber desarrollado una interfaz es la continuación de este proyecto por parte de HP.

Si accedemos a la página principal de nuestra aplicación, nos encontramos con dos apartados bien diferenciados, que como ya hemos comentado con anterioridad son la carga de la base de datos textual y la búsqueda. En la siguiente imagen vemos las dos opciones.

Imagen 15: Index de la aplicación

Vamos a comenzar cargando al base de datos. En primer lugar, debemos indicar una ruta. Por defecto viene establecida como ruta, la ruta “c:/logTfg”. En el momento en que pulsamos sobre el botón cargar, este cambia su contenido a cargando y se deshabilita (imagen 16) al igual que el botón de borrado (imagen 17) y el de búsqueda (imagen 18). El proceso tardará unos segundos en ejecutarse, en función del número de logs que tenga que cargar. Este es el principal motivo para deshabilitar estos botones.

Imagen 16: Botón de carga deshabilitado

Imagen 17: Botón de borrado deshabilitado

Imagen 18: Botón de búsqueda deshabilitado

Una vez haya finalizado todo el proceso, nos enviará a otra página indicando que el proceso ha finalizado, sin darnos la opción de volver a cargar, ya que si por accidente volvemos a pulsar ese botón el proceso se repetirá de nuevo, no aportándonos nada ya que la carga acaba de ser realizada y haciéndonos perder un tiempo. En caso de desear volver a cargar por diversos motivos, por ejemplo, ha pasado mucho tiempo desde que hicimos la carga y seguimos en la misma página tendríamos que volver al inicio, pero ese no es el objetivo de este sistema ya que está pensado para que cuando hagas una carga de la base de datos, sea para hacer una búsqueda posterior.

En caso de que la ruta introducida sea errónea, la aplicación te notificará de ello con un mensaje que aparecerá en la siguiente página con la posibilidad de realizar la carga de nuevo.

Hasta aquí tendríamos realizada la carga, ahora es el momento de realizar una búsqueda. El funcionamiento es exactamente igual en todas las pantallas de búsqueda por lo que nos centraremos en el índex. Como hemos visto en la primera imagen de este apartado, esta función aparece en la parte derecha de la página.

Tendremos un desplegable con todos los campos sobre los que podemos realizar la búsqueda (imagen 19). Aunque se podría haber realizado sobre todos, algunos de ellos no tenían sentido ya que en muchos casos estaban en blanco o era información relevante para el objetivo que se quiere cubrir aquí.

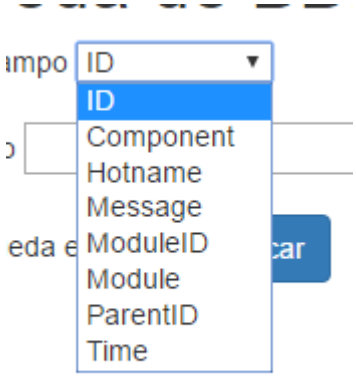


Imagen 19: Opciones de búsqueda

A continuación, introduciremos el contenido que deseamos buscar. No se podría enviar el formulario sin haber rellenado el campo contenido mostrando un error (imagen 20).

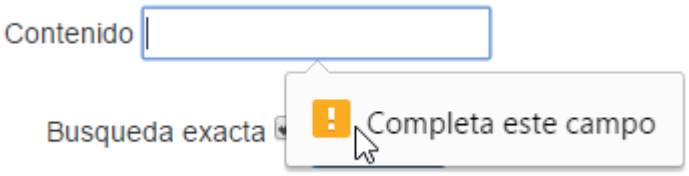


Imagen 20: Error de búsqueda sin contenido

Tendremos la opción de marcar el checkbox de "Búsqueda exacta". Este checkbox lo que hace es buscar en el campo marcado, el contenido exacto, es decir, todo el campo completo debe ser ese contenido. Tener esta opción marcada es muy útil por ejemplo para buscar los elementos correspondientes a un determinado ID de trabajo. En caso de tener este checkbox desmarcado, se buscará el campo indicado que contenga el contenido indicado. Esta opción es muy útil, por ejemplo, si deseamos buscar un Message que incluya cierto patrón. Incluso podremos indicar con un asterisco en medio del contenido, que se empleará para realizar la búsqueda para cualquier carácter o expresión.

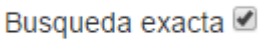


Imagen 21: Checkbox de búsqueda exacta

Finalmente, una vez hemos pinchado el botón de buscar, seremos redirigido a una página que contendrá una tabla con toda la información respectiva a la petición realizada. La siguiente imagen muestra el resultado de buscar un ID igual a 5747.

ID	Hostname	Component	Module	ModuleID	ParentID	Part	Thread	Time	Topic	Message
5747	MAQUINAVIRTUAL		mwfm	null	EMPTY	COMPONENT	RMI TCP Connection(60)-192.168.88.128	19-dic-2014 13:45:59		value of maxNodesPerThread is 5
5747	MAQUINAVIRTUAL	WFMEngineWorker	mwfm	null	EMPTY	FRAMEWORK	RMI TCP Connection(60)-192.168.88.128	19-dic-2014 13:45:59		starting in node 'Asignar Subaplicacion Deteccion Temprana Alarma'.
5747	MAQUINAVIRTUAL	WFMEngineWorker	mwfm	null	EMPTY	FRAMEWORK	RMI TCP Connection(60)-192.168.88.128	19-dic-2014 13:45:59		context set.
5747	MAQUINAVIRTUAL	WFMEngineWorker	mwfm	null	EMPTY	FRAMEWORK	RMI TCP Connection(60)-192.168.88.128	19-dic-2014 13:45:59		engine case packet variables set for job #5747
5747	MAQUINAVIRTUAL	Asignar Subaplicacion Deteccion Temprana Alarma	mwfm	null	EMPTY	COMPONENT	RMI TCP Connection(60)-192.168.88.128	19-dic-2014 13:45:59		In getAttribute ----- Type of case packet PRIORITY is Integer
5747	MAQUINAVIRTUAL	Asignar Subaplicacion Deteccion Temprana Alarma	mwfm	null	EMPTY	COMPONENT	MWFM Worker 5	19-dic-2014 13:45:59		attempting to set attribute 'sSubaplicacion' to 'Deteccion Temprana Alarma'.
5747	MAQUINAVIRTUAL	Asignar Subaplicacion Deteccion Temprana Alarma	mwfm	null	EMPTY	COMPONENT	MWFM Worker 5	19-dic-2014 13:45:59		attempting to set attribute 'syncTimeout' to '280000'.
5747	MAQUINAVIRTUAL	Asignar Subaplicacion Deteccion Temprana Alarma	mwfm	null	EMPTY	COMPONENT	MWFM Worker 5	19-dic-2014 13:45:59		In getAttribute ----- Type of case packet RUNTIME is Object
5747	MAQUINAVIRTUAL	Iniciar Deteccion Temprana Alarma	mwfm	null	EMPTY	COMPONENT	MWFM Worker 5	19-dic-2014 13:45:59		Job ID inside _updateState() is 5747

Imagen 22: Ejemplo de resultado de búsqueda

Cualesquiera de las columnas mostradas pueden ser ordenadas simplemente pinchando en el nombre de la columna.

Llegados abajo y si la tabla del resultado encontrado es muy extensa, nos encontramos ante el problema de regresar arriba, por ejemplo, si queremos realizar una ordenación nueva de la columna, etc. En un ordenador estaría fácil ya que contamos con la tecla "Inicio" en el teclado, pero si el acceso se está realizando desde otro dispositivo posiblemente no contemos con esa ventaja por lo que se ha colocado un botón abajo que nos permite volver al inicio de la página.

8. Pruebas

Durante el proceso de prueba se han realizado las siguientes pruebas con su consiguiente estado, siendo necesaria su corrección o no:

Prueba	Versión inicial	Estado final
Carga de la base de datos con ruta correcta	Sin errores	Sin errores
Carga de base de datos con ruta errónea	Sin implementación	Sin errores
Búsqueda de un ID concreto	Sin errores	Sin errores
Búsqueda de un ID sin búsqueda exacta	Sin errores	Sin errores
Búsqueda de un ID erróneo o que no se encuentra	Sin errores	Sin errores
Búsqueda de un Hostname concreto	Error (Campo no encontrado)	Sin errores
Búsqueda de un Hostname sin búsqueda exacta	Error (Campo no encontrado)	Sin errores
Búsqueda de un Hostname erróneo o que no se encuentra	Error (Campo no encontrado)	Sin errores
Búsqueda de un ParentID concreto	Sin errores	Sin errores
Búsqueda de un ParentID sin búsqueda exacta	Sin errores	Sin errores
Búsqueda de un ParentID erróneo o que no se encuentra	Sin errores	Sin errores
Búsqueda de un Time concreto	Sin errores	Sin errores
Búsqueda de un Time sin búsqueda exacta	Sin errores	Sin errores
Búsqueda de un Time erróneo o que no se encuentra	Sin errores	Sin errores
Búsqueda de un Component concreto	Sin errores	Sin errores
Búsqueda de un Component sin búsqueda exacta	Sin errores	Sin errores

Búsqueda de un Component erróneo o que no se encuentra	Sin errores	Sin errores
Búsqueda de un Message concreto	Sin errores	Sin errores
Búsqueda de un Message sin búsqueda exacta	Sin errores	Sin errores
Búsqueda de un Message erróneo o que no se encuentra	Sin errores	Sin errores
Búsqueda de un ModuleID concreto	Sin errores	Sin errores
Búsqueda de un ModuleID sin búsqueda exacta	Sin errores	Sin errores
Búsqueda de un ModuleID erróneo o que no se encuentra	Sin errores	Sin errores
Búsqueda de un ModuleID concreto	Sin errores	Sin errores
Búsqueda de un ModuleID sin búsqueda exacta	Sin errores	Sin errores
Búsqueda de un ModuleID erróneo o que no se encuentra	Sin errores	Sin errores
Ordenación de todos los campos en ambas direcciones	Sin implementar	Sin errores
Borrado de base de datos	Sin implementar	Sin errores

Tabla 7: Batería de pruebas

Todas las cargas, borrados y búsquedas se han realizado desde todas las posibles pantallas

En el caso de los elementos que en la versión inicial estaban sin implementar, fueron añadidos por el cliente en una reunión posterior, ampliando estos cambios que posteriormente implementaríamos.

En el caso de los errores provocados por la búsqueda sobre el campo Hostname, era un simple error de escritura, en el que el valor del campo recogido por el formulario estaba mal escrito con el nombre de “Hotname” y no de “Hostname”.

9. Conclusión y trabajo futuro

El mundo de internet está avanzando constantemente y tenemos que desarrollar soluciones que nos permita ir a la par con este crecimiento de las nuevas tecnologías. Es ahí donde surge el concepto de bases de datos NoSQL. A día de hoy las más empleadas y las más conocidas son las bases de datos relacionales siendo las otras un mundo aún por descubrir por mucha gente. Uno de los motivos de este desconocimiento puede ser la juventud de estas o incluso la desconfianza al poder elegir un sistema relacional con muchos más años de experiencia.

Cuando comienzas a usarlas te das cuenta de que no son sistemas tan jóvenes y que tienen por detrás un gran desarrollo mejorando en muchos conceptos a las bases de datos relacionales que ya conocíamos. Contamos con más de 100 sistemas de este este tipo y aunque si es cierto que a muchas de ellas les falta todavía tiempo de desarrollo para llegar a ser sistemas fiables, podemos elegir un sistema que se adapte completamente a nuestro objetivo.

Solr tiene una gran integración con Java y usando Lucene como sistema base, lo convierte en un sistema de búsqueda potente.

El estudio realizado durante este trabajo ha sido cuantioso descubriendo no solo las tecnologías empleadas durante el desarrollo de la aplicación, sino también tecnologías relacionadas y que, si bien podrían haber sido válidas, no eran las más correctas.

Este proyecto ha sido realizado para una empresa externa, en concreto, HP por lo que se acerca en cierto modo a un entorno profesional en los que es la empresa los que te pone las condiciones y establece los plazos, donde muchas de las decisiones de diseño son tomadas por la propia empresa. Esto me ha preparado para enfrentarme a situaciones relativamente reales y a emplear sistemas totalmente desconocidos para mí.

En cuento al trabajo futuro, como ya se ha indicado en alguna ocasión, HP tiene la intención de seguir ampliando las funcionalidades con nuevos colectores o decoradores. Por este motivo se ha dejado una implementación que permita una fácil incorporación de nuevos sistemas. También se continuará con la parte de front-end.

Bibliografía

1. SearchDataCenter en Español. ¿Qué es NoSQL (No Solo SQL)? - Definición en WhatIs.com [Internet]. 2014 [accedido el 8 de Diciembre 2015]. Disponible en: http://searchdatacenter.techtarget.com/es/definicion/NoSQL-No-Solo-SQL

2. Edlich P. NOSQL Databases [Internet]. Nosql-database.org. 2015 [accedido el 7 de Diciembre 2015]. Disponible en: http://nosql-database.org/

3. Es.wikipedia.org. NoSQL [Internet]. 2015 [accedido el 7 de Diciembre 2015]. Disponible en: https://es.wikipedia.org/wiki/NoSQL

4. Onewebsql.com. NoSQL - Do I Really Need It? | OneWebSQL [Internet]. [accedido el 8 de Diciembre 2015]. Disponible en: http://onewebsql.com/blog/no-sql-do-i-really-need-it

5. Paramio C. El concepto NoSQL, o cómo almacenar tus datos en una base de datos no relacional [Internet]. Genbetadev.com. 2011 [accedido el 7 de Diciembre 2015]. Disponible en: http://www.genbetadev.com/bases-de-datos/el-concepto-nosql-o-como-almacenar-tus-datos-en-una-base-de-datos-no-relacional

6. Garcete A. Base de Datos Orientadas a Columnas [Internet]. 1st ed. 2015 [accedido el 10 de Diciembre 2015]. Disponible en: http://jeuazarru.com/wp-content/uploads/2014/10/dbco.pdf

7. Genbetadev.com. Bases de datos NoSQL. Elige la opción que mejor se adapte a tus necesidades [Internet]. 2014 [accedido el 8 de Diciembre 2015]. Disponible en: http://www.genbetadev.com/bases-de-datos/bases-de-datos-nosql-elige-la-opcion-que-mejor-se-adapte-a-tus-necesidades

8. Ondho. Claves para elegir tu base de datos NoSQL | Ondho [Internet]. 2015 [accedido el 12 de Diciembre 2015]. Disponible en: https://www.ondho.com/claves-para-elegir-tu-base-de-datos-nosql/

9. QuantumMode. Las Bases de Datos NoSQL (II): El Modelo de Datos - QuantumMode [Internet]. 2014 [accedido el 13 de Diciembre 2015]. Disponible en: http://quantummode.com/las-bases-de-datos-nosql-ii-el-modelo-de-datos/

10. Es.slideshare.net. Big data y las apis (big data spain) [Internet]. 2016 [accedido el 15 de Diciembre 2015]. Disponible en: http://es.slideshare.net/MarcoAntonioSanzMoli/big-data-y-las-apis-big-data-spain

11. Shermin M. An Access Control Model for NoSQL Databases [Tesis de Master]. University of Western

Ontario; 2013. [accedido el 14 de Diciembre 2015]. Disponible en: http://ir.lib.uwo.ca/etd/1797/

12. Bases de datos NoSQL. Qué son y tipos que nos podemos encontrar [Internet]. 1st ed. 2014 [accedido el 14 de Diciembre 2015]. Disponible en: http://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf

13. Osorio Gonzalez J. NoSql-Clave-Valor [Internet]. prezi.com. 2013 [accedido el 16 de Diciembre 2015]. Disponible en: https://prezi.com/fdafxecgah4u/nosql-clave-valor/

14. Gonzalez Castro V. Bases de Datos Columnares [Internet]. Software Guru. 2015 [accedido el 18 de Diciembre 2015]. Disponible en: http://sg.com.mx/content/view/829

15. Valero N. Bases de Datos Columnares | Business Intelligence, Data Warehouse, Monterrey, México : Gravitar [Internet]. Gravitar.biz. 2015 [accedido el 18 de Diciembre 2015]. Disponible en: http://gravitar.biz/bi/base-datos-columnar/

16. Es.wikipedia.org. Base de datos documental [Internet]. 2015 [accedido el 20 de Diciembre 2015]. Disponible en: https://es.wikipedia.org/wiki/Base_de_datos_documental

17. Wikipedia. Document-oriented database [Internet]. 2007 [accedido el 21 de Diciembre 2015]. Disponible en: https://en.wikipedia.org/wiki/Document-oriented_database

18. Db-engines.com. CouchDB vs. Elasticsearch vs. MongoDB vs. Solr Comparison [Internet]. [accedido el 3 de Enero 2016]. Disponible en: http://db-engines.com/en/system/CouchDB%3BElasticsearch%3BMongoDB%3BSolr

19. Es.wikipedia.org. Base de datos orientada a objetos [Internet]. 2015 [accedido el 22 de Diciembre 2015]. Disponible en: https://es.wikipedia.org/wiki/Base_de_datos_orientada_a_objetos

20. Solano J. Base De Datos Orientada A Objetos [Internet]. Es.slideshare.net. 2015 [accedido el 23 de Diciembre 2015]. Disponible en: http://es.slideshare.net/joseivanestradasolano/base-de-datos-47357883

21. Modelos de BD. Bases de datos orientados a objetos [Internet]. 2012 [accedido el 23 de Diciembre 2015]. Disponible en: https://modelosbd2012t1.wordpress.com/2012/01/27/bases-de-datos-orientados-a-objetos/

22. de la Torre A. Base de datos orientada a objetos - Monografias.com [Internet]. Monografias.com. [accedido el 23 de Diciembre 2015]. Disponible en: http://www.monografias.com/trabajos79/base-datos-orientadas-objetos/base-datos-orientadas-objetos.shtml

23. Gomez Parede J. Bases de datos Multi-Modelos, Shut Up And Take My Money [Internet]. Medium. 2015

[accedido el 22 de Diciembre 2015]. Disponible en: https://medium.com/@DezkaReid/bases-de-datos-multi-modelos-shut-up-and-take-my-money-c2b8502333c2#.9vifautdo

24. Andres Campos F. Curso completo de Elasticsearch [Internet]. Es.slideshare.net. [accedido el 27 de Diciembre 2015]. Disponible en: http://es.slideshare.net/FedericoAndrsOcampo/curso-completo-de-elasticsearch

25. García L. Un poco de ElasticSearch [Internet]. Un poco de Java. 2013 [accedido el 27 de Diciembre 2015]. Disponible en: https://unpocodejava.wordpress.com/2013/10/01/un-poco-de-elasticsearch/

26. Es.wikipedia.org. MongoDB [Internet]. 2015 [accedido el 28 de Diciembre 2015]. Disponible en: https://es.wikipedia.org/wiki/MongoDB

27. Genbetadev.com. MongoDB: empezando por el principio. Insertando datos [Internet]. 2014 [accedido el 29 de Diciembre 2015]. Disponible en: http://www.genbetadev.com/bases-de-datos/mongodb-empezando-por-el-principio-insertando-datos

28. Sánchez Suárez J. Introducción a Apache Solr. | Adictos al Trabajo [Internet]. Adictosaltrabajo.com. 2014 [accedido el 29 de Diciembre 2015]. Disponible en: http://www.adictosaltrabajo.com/tutoriales/apache-solr-intro/#01

29. Es.wikipedia.org. CouchDB [Internet]. 2015 [accedido el 30 de Diciembre 2015]. Disponible en: https://es.wikipedia.org/wiki/CouchDB

30. Es.wikipedia.org. CouchDB [Internet]. 2015 [accedido el 30 de Diciembre 2015]. Disponible en: https://es.wikipedia.org/wiki/CouchDB#Empresas_que_usan_CouchDB

31. Conde J. ¿Qué es REST y RESTful? [Internet]. YouTube. 2016 [accedido el 27 de Diciembre 2015]. Disponible en: https://www.youtube.com/watch?v=pVAMOielOJQ

32. Es.wikipedia.org. JSON [Internet]. 2015 [accedido el 27 de Diciembre 2015]. Disponible en: https://es.wikipedia.org/wiki/JSON

33. Json.org. JSON [Internet]. [accedido el 29 de Diciembre 2015]. Disponible en: http://www.json.org/json-es.html

34. Es.wikipedia.org. Lucene [Internet]. 2015 [accedido el 2 de Enero 2016]. Disponible en: https://es.wikipedia.org/wiki/Lucene

35. Wikipedia. BSON [Internet]. 2015 [accedido el 27 de Diciembre 2015]. Disponible en: https://en.wikipedia.org/wiki/BSON

36. Es.wikipedia.org. GNU Affero General Public License [Internet]. 2015 [accedido el 27 de Diciembre 2015]. Disponible en: https://es.wikipedia.org/wiki/GNU_Affero_General_Public_License

37. Vicente-navarro.com. Lo hice y lo entendí | La licencia Affero (AGPL) y su relación con la GPL [Internet]. 2008 [accedido el 27 de Diciembre 2015]. Disponible en: http://www.vicente-navarro.com/blog/2008/03/01/la-licencia-affero-agpl-y-su-relacion-con-la-gpl/

38. Rouse M. What is sharding? - Definition from WhatIs.com [Internet]. SearchCloudComputing. [accedido el 29 de Diciembre 2015]. Disponible en: http://searchcloudcomputing.techtarget.com/definition/sharding

39. Alegsa L. Definicion de ACID [Internet]. Alegsa.com.ar. 2010 [accedido el 30 de Diciembre 2015]. Disponible en: http://www.alegsa.com.ar/Dic/acid.php

40. Es.wikipedia.org. ACID [Internet]. 2015 [accedido el 30 de Diciembre 2015]. Disponible en: https://es.wikipedia.org/wiki/ACID

41. Brigomp.blogspot.com.es. Pensamientos ágiles: Sharding,¿sí o no? [Internet]. 2009 [accedido el 29 de Diciembre 2015]. Disponible en: http://brigomp.blogspot.com.es/2009/08/sharding-si-o-no.html

42. Es.wikipedia.org. Apache License [Internet]. 2015 [accedido el 28 de Diciembre 2015]. Disponible en: https://es.wikipedia.org/wiki/Apache_License

Anexo I (Ejemplo archivos originales)

El código siguiente es un ejemplo de dos entradas de los archivos originales. Como vemos no todos contienes los mismos campos.

```
<LogEntry level="INFORMATIVE" no="0" >
   <HostName>vodafone-broadband.esm.cpqcorp.net</HostName>
   <Time>Feb 11, 2016 9:53:12 AM</Time>
   <Module>mwfm</Module>
   <Part>COMPONENT</Part>
   <Component>WFTransactionHandler</Component>
   <Topic></Topic>
   <Thread>MWFM Worker 4</Thread>
   <ID>43059</ID>
  <Message>SUBSTEP=</Message>
</LogEntry>
<LogEntry level="DEBUG2" no="1" >
   <HostName>vodafone-broadband.esm.cpqcorp.net</HostName>
   <Time>Feb 11, 2016 9:53:12 AM</Time>
   <Module>mwfm</Module>
   <Part>COMPONENT</Part>
   <Component>WFTransactionHandler</Component>
   <Topic></Topic>
   <Thread>MWFM Worker 4</Thread>
   <ID>43059</ID>
  <Message>getting null attribute 'EX_STEP_NAME', default value
   ""</Message>
</LogEntry>
```

Anexo II (Instalación y ejecución de Tomcat)

Antes de comenzar debemos tener instalado Java SE Runtime Environment (jre). En caso de no tenerlo debemos descargarlo de http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html e instalarlo posteriormente.

Tomcat es un contenedor de aplicaciones desarrollado Apache Software Foundation. La página oficial de este es: http://tomcat.apache.org/ . Para descargarlo debemos acceder a la versión que deseemos en la sección “Download” del menú principal. En nuestro caso la versión que instalaremos es la 8.

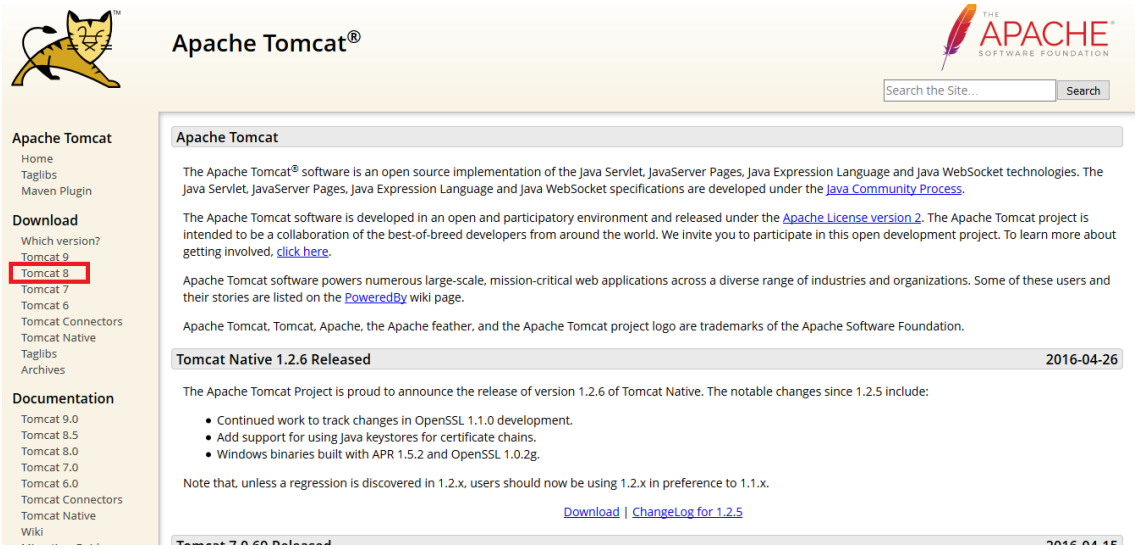


Imagen 23: Descarga Tomcat

Tenemos disponibles varias versiones para descargar para sistemas operativos Windows. Los cuatro primeros contienen el código fuente del servidor tomcat, mientras que el último nos permite descargar un instalador con extensión exe. Este será el que utilicemos.

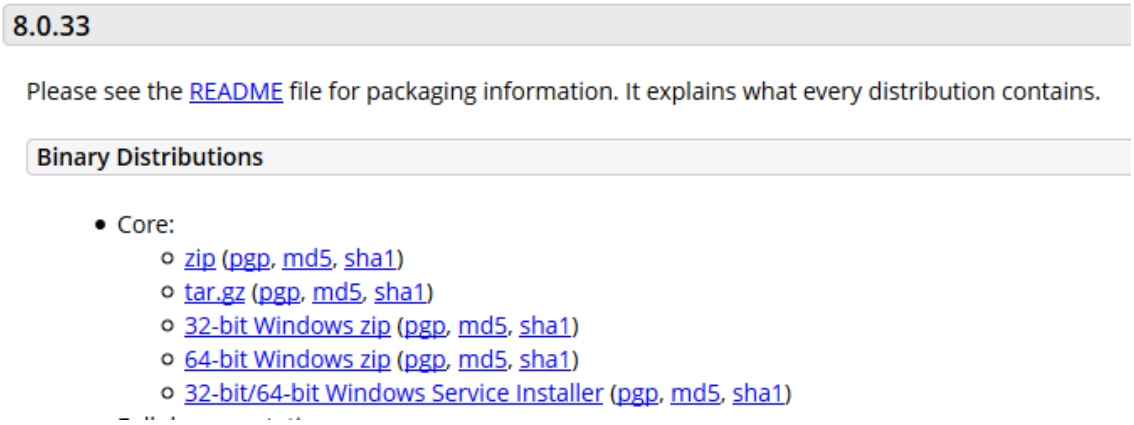


Imagen 24: Descarga versión Tomcat

Una vez tenemos descargado el instalador, lo ejecutamos y seguimos los pasos de la instalación. Durante el proceso podremos cambiar la configuración de los puertos, así como del usuario como se muestra en la siguiente imagen. En nuestro caso los dejaremos por defecto sin ningún usuario.

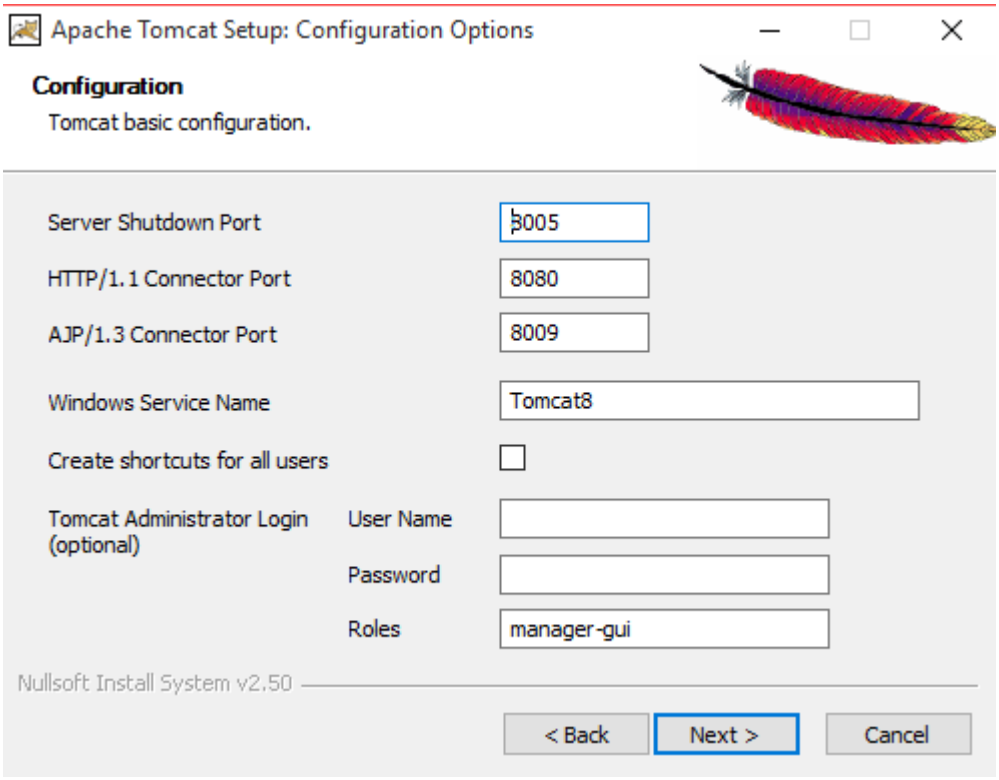


Imagen 25: Configuración de puertos Tomcat

También podremos cambiar la ruta de instalación por defecto que también la dejaremos por defecto, siendo esta: “C:\Program Files\Apache Software Foundation\Tomcat 8.0”.

Una vez hemos finalizado deberemos encontrar en la carpeta de instalación una estructura similar a esta:

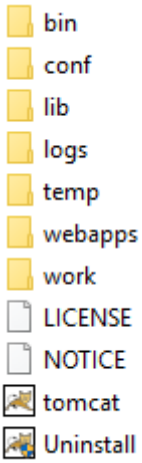


Imagen 26: Estructura de archivos creada en Tomcat

Dentro de esta estructura tenemos dos carpetas que tenemos que tener en cuenta durante nuestro proyecto:

- Bin: Guarda los archivos necesarios para ejecutar (startup.bat) y apagar (shutdown.bat) el contenedor de aplicaciones aparte de configurar la ejecución de este (archivo Tomcat8w.exe). Para iniciar también podemos emplear el archivo Tomcat8.exe que cerrando la ejecución de este conseguimos apagar el servidor.

- Webapps: Esta carpeta contendrá todas las aplicaciones web que creemos. Para su ejecución desde el servidor es imprescindible que estas se encuentren localizadas en este directorio.

Ya podemos ejecutar nuestro servidor. Para comprobar su funcionamiento basta con dirigirnos a http://localhost:8080/ a través de cualquier navegador web. Si todo va bien debe aparecer la ventana de inicio:

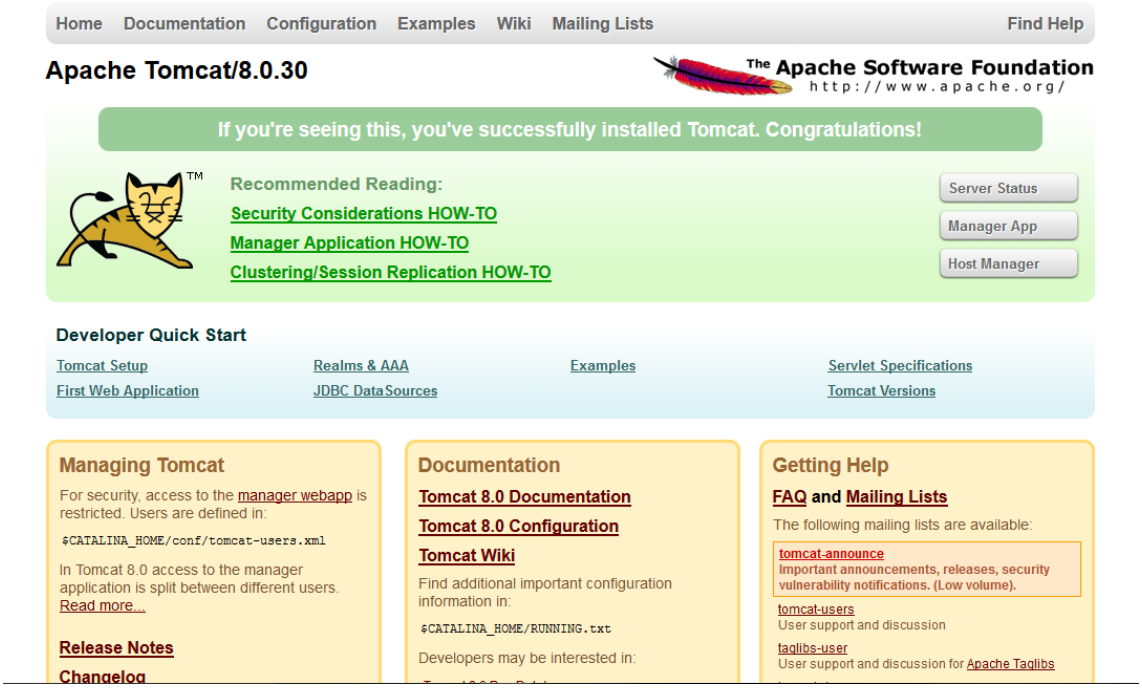


Imagen 27: Índex servidor Tomcat

Anexo III (Desplegar Solr sobre Tomcat)

Solr deja de ser compatible con Tomcat a partir de la versión 5 de Solr, por lo que en este caso realizaremos la instalación con la versión 4, en concreto, la 4.10.4. Durante el proceso de instalación explicado a continuación daremos los pasos necesarios para versiones 4.2 e inferiores y los pasos para versiones desde la 43 hasta la 4.10.4 ya que estos difieren en unos pequeños puntos.

Primero comenzaremos descargando la versión del servidor de búsqueda Solr desde https://www.apache.org/dist/lucene/solr/4.10.4/. En este caso descargaremos el archivo con extensión zip.

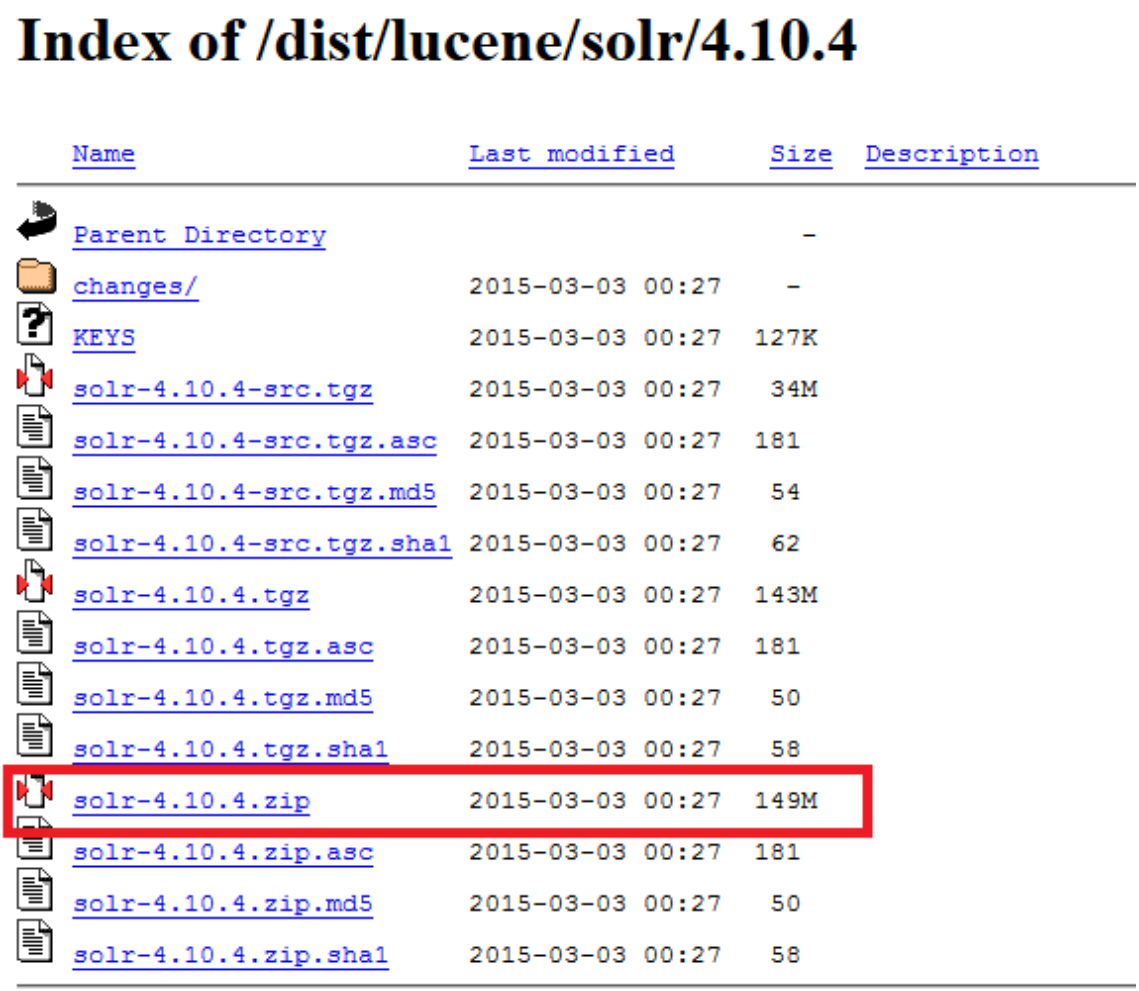


Imagen 28: Descarga Solr

Una vez lo tenemos descargado, lo abrimos y nos copiamos el archivo solr-4.10.4\dist\solr-4.10.4.war, renombrado a solr.war, al directorio webapps situado donde hemos instalado el servidor Tomcat (C:\Program Files\Apache Software Foundation\Tomcat 8.0\webapps).

Si Tomcat se está ejecutando, se creará automáticamente un directorio “solr”, sino, no pasa nada ya que se creará la siguiente vez que lo ejecutemos.

Los pasos que vienen a continuación son únicos para versiones 4.3 o posteriores:

- Copiar los archivos jar desde solr-4.10.4\example\lib\ext del archivo descargado en el directorio lib de la carpeta de instalación de Tomcat.

- Debemos copiar el archivo solr-4.10.4\example\resources\log4j.properties del archivo descargado con el código fuente de Solr al mismo directorio que en el punto anterior

Ya podemos continuar con la instalación normal, es decir, la compatible con cualquier versión de Solr.

El siguiente paso es crear el Home de Solr. En el caso que nos ocupa lo crearemos en la raíz del disco. Copiamos el directorio completo “solr” del archivo descargado a C:\.

Ahora debemos indicar a Tomcat que el Home se encuentra en el directorio donde nosotros lo hemos situado. Para ello debemos ejecutar Tomcat8w.exe, cuya ruta ya hemos indicado con anterioridad (C:\Program Files\Apache Software Foundation\Tomcat 8.0\bin\Tomcat8w.exe). Una vez abierto, vamos a la pestaña Java y añadimos en Java Options la siguiente línea: -Dsolr.solr.home=c:\solr.

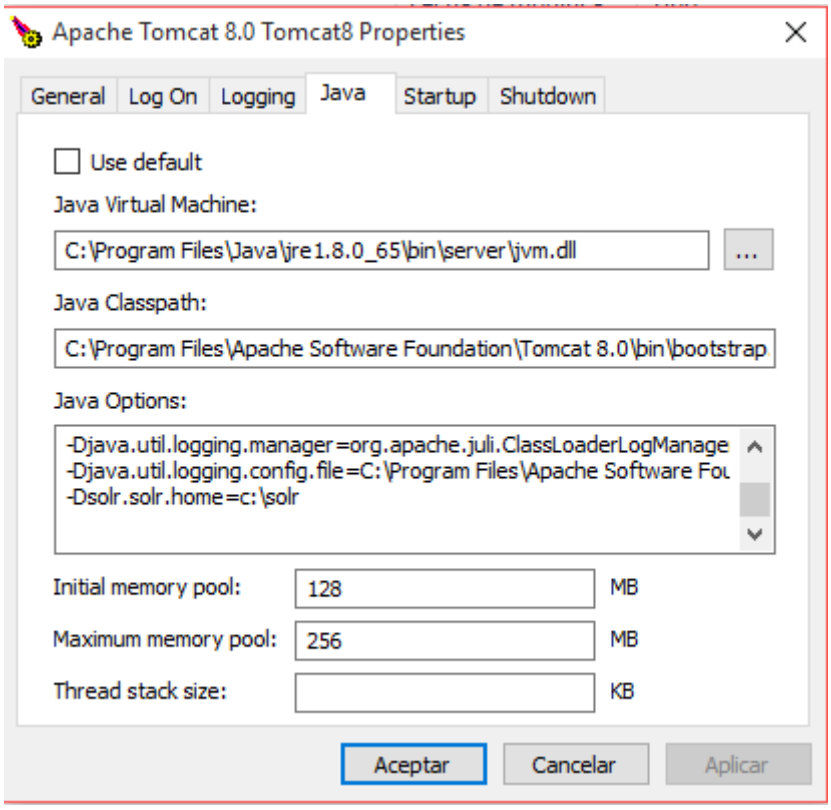


Imagen 29: Configuración HOME de Solr en Tomcat

Aplicamos cambios y solo faltaría probar que se ha realizado todo correctamente accediendo dentro de nuestro navegador a la ruta: http://localhost:8080/solr. En caso de ser todo correcto deberíamos ver la pantalla principal del panel de Solr que será similar a la siguiente captura:

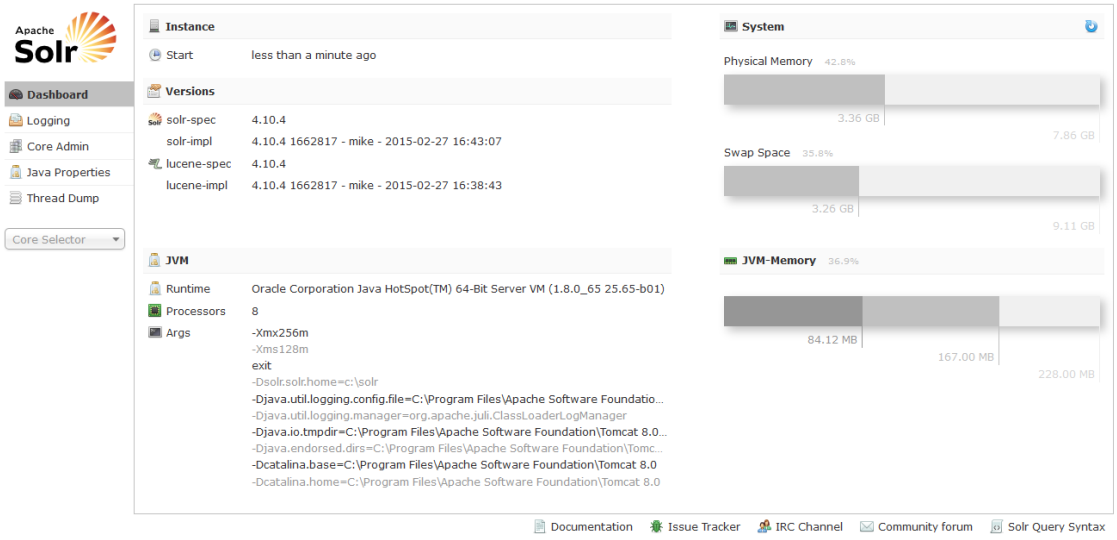


Imagen 30: Panel de control Solr

Nota: Es necesario tener iniciado el servidor Tomcat para poder acceder al panel de control.

Anexo IV (Creación de colecciones en Solr)

Podemos crear una colección desde cero, creando todos los archivos necesarios, pero es mucho más sencillo utilizar una colección ya existente ya que muchos de los archivos necesarios para la colección los preservaremos intactos. Si no hemos creado ninguna colección hasta el momento, existe una colección que viene por defecto ya creada al instalar solr. Esta colección se llama “collection1”. Duplicaremos esta carpeta asignándolo un nombre nuevo (“LogMining”). Realizaremos los siguientes cambios:

- Archivo LogMining/conf/schema.xml que contendrá el esquema de nuestra base de datos. Los datos de este archivo son:

```
<?xml version="1.0" encoding="UTF-8" ?>
   <schema version="1.5">
        <fields>
             <field    name="Check"    type="string"    indexed="true"
        stored="true" required="true"/>
             <field name="Hostname" type="string" indexed="true"
        stored="true" required="true"/>
             <field name="Time" type="string" indexed="true"
        stored="true" required="true"/>
             <field name="Module" type="string" indexed="true"
        stored="true" required="true"/>
             <field name="ModuleID" type="string" indexed="true"
        stored="true" required="false"/>
             <field    name="Part"    type="string"    indexed="true"
        stored="true" required="true"/>
             <field name="Component" type="string" indexed="true"
        stored="true" required="true"/>
             <field name="Topic" type="string" indexed="true"
        stored="true" required="true"/>
             <field name="Thread" type="string" indexed="true"
        stored="true" required="true"/>
             <field name="ID" type="string" indexed="true"
        stored="true" required="false" default="EMPTY"/>
```

```
        <field name="Message" type="string" indexed="true"
    stored="true" required="true"/>
        <field name="level" type="string" indexed="true"
    stored="true" required="false"/>
        <field name="no" type="string" indexed="true"
    stored="true" required="false"/>
        <field name="ParentID" type="string" indexed="true"
    stored="true" required="false" default="EMPTY"/>
    </fields>
    <uniqueKey>Check</uniqueKey>
    <types>
        <fieldType name="string" class="solr.StrField" />
    </types>
</schema>
```

Cada uno de los elementos “field” indican cada uno de los campos de los documentos. Debemos indicar:

- nombre (campo name)
- tipo (campo type)
- si esta indexado (indexed)
- si el valor exacto de este campo debe poder ser recuperado (stored)
- si es obligatorio (required) y el valor que toma por defecto (default).

Estos son los atributos empleados en el caso que nos concierne, pero existen otras muchas posibilidades como:

- docValues que en caso de ser true este campo debe tener doc values
- multiValued que si es true puede haber más de un valor asignado a este campo en un documento
- omitNorms que se pone a true para omitir las normas asociadas con este campo, aunque por defecto se omiten

- termVectors se pone a true para almacenar el vector de términos de un campo
- termPositions almacena información de posición en el vector de términos
- termOffsets que almacena información de desplazamiento en el vector de términos.

- Archivo LogMining/conf/solrconfig.xml. Los datos que contendrá este archivo son:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <config>
       <luceneMatchVersion>LUCENE_43</luceneMatchVersion>
       <requestDispatcher handleSelect="false">
       <httpCaching never304="true" />
       </requestDispatcher>
       <requestHandler name="/select" class="solr.SearchHandler" />
       <requestHandler name="/update" class="solr.UpdateRequestHandler"
  />
       <requestHandler name="/admin" class="solr.admin.AdminHandlers" />
       <requestHandler name="/analysis/field" class="solr.
FieldAnalysisRequestHandler" startup="lazy" />
    </config>
```

- Archivo LogMining/core.properties que almacenará el nombre de la colección. Esta es la única línea que tendrá:

```
name=LogMining
```

Una vez hemos realizado estos cambios, solo necesitamos comprobar si la colección ha sido creada en el panel de control de solr.

Anexo V (Emplear JaxB con Eclipse)

JaxB (Java Architecture for XML Binding) nos permite leer archivos XML con aplicaciones Java. Podremos crear todas las clases necesarias de forma automática gracias a la herramienta suministrada con JDK, XJC. Para ello tendremos que crear primero un Schema XML de la estructura de los XML que deseemos cargar. Eclipse permite un sistema fácil e intuitivo para crearlo.

El código de este archivo es:

```
<schema                          xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.example.org/LogEntrys"
   xmlns:tns="http://www.example.org/LogEntrys"
   elementFormDefault="qualified">


    <element name="LogEntrys" type="tns:LogEntrys"></element>


    <complexType name="LogEntrys">
        <sequence>
            <element name="LogEntry" type="tns:LogEntry"
                maxOccurs="unbounded">
            </element>
        </sequence>
    </complexType>


    <complexType name="LogEntry">
        <sequence>
            <element name="HostName" type="string" maxOccurs="1"
                minOccurs="1">
            </element>
            <element name="Time" type="string" maxOccurs="1"
                minOccurs="1">
            </element>
            <element name="Module" type="string" maxOccurs="1"
```

```
                    minOccurs="1">
                </element>
                <element name="ModuleID" type="string"></element>
                <element name="Part" type="string" maxOccurs="1"
                    minOccurs="1">
                </element>
                <element name="Component" type="string" maxOccurs="1"
                    minOccurs="1">
                </element>
                <element name="Topic" type="string" maxOccurs="1"
                    minOccurs="1">
                </element>
                <element name="Thread" type="string" maxOccurs="1"
                    minOccurs="1">
                </element>
                <element name="ID" type="string" maxOccurs="1"
                    minOccurs="0">
                </element>
                <element name="Message" type="string" maxOccurs="1"
                    minOccurs="1">
                </element>
            </sequence>
            <attribute name="level" type="string"></attribute>
            <attribute name="no" type="string"></attribute>
        </complexType>
    </schema>
```

Como ya hemos comentado Eclipse permite crear los XSD de forma muy sencilla y esta es gráficamente, de manera que se puede apreciar su estructura claramente. Solo tendremos que crear los elementos principales y toda su estructura haciendo las conexiones necesarias entre ellos. Añadiremos también los elementos escribiendo todos los elementos que pertenecen a un elemento padre, la multiplicidad de este así como su tipo.

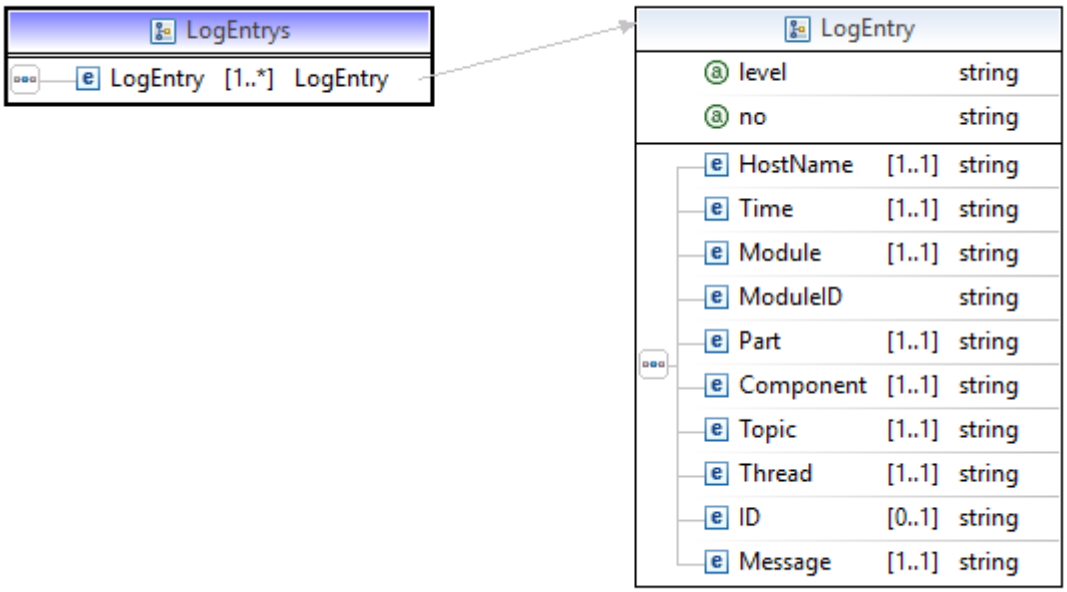


Imagen 31: Esquema de los archivos XML

En nuestro caso si se ha creado con Eclipse, pero podría haber sido creado con cualquier otra herramienta e incluso a mano con cualquier editor de texto.

Una vez los hemos finalizado crear todas las clases necesarias es muy sencillo ya que solo tendremos que ejecutar una herramienta sobre este archivo XSD. Esta es XJC que viene incluida con el jdk. Su ejecución se realiza sobre línea de comandos. La sintaxis de esta orden es:

```
Ruta_del_ejecutable_de_XJC -d ruta_de_destino_de_las_clases ruta_del_XSD.
```

Por ejemplo:

```
C:\Program Files\Java\jdk1.8.0_65\bin\xjc.exe -d .\src
   src\xml\LogEntrys.xsd
```

El árbol de clases creadas será:

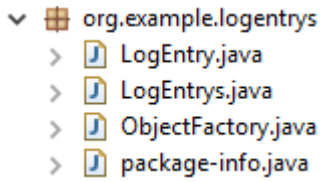


Imagen 32: Árbol de clases creadas por XJC

Uno de los problemas de esta herramienta es que con estas clases no detecta un elemento raíz por lo que no funcionaria. La solución es muy simple y consiste en añadir una línea al principio de la clase “LogEntrys.java”. Esta es:

```
@XmlRootElement (name="LogEntrys")
```

Anexo VI (Incorporar Solr con Eclipse)

Solr nos proporciona un APIs para integrar de forma fácil este sistema con Java. Este API se llama SolrJ. Para poderlo emplear en Eclipse necesitamos incluir ciertos archivos jar a la librería en Eclipse. Ya hemos indicado de donde en el anexo III de donde descargar Solr y es que SolrJ viene incluido dentro de este paquete fuente. Los archivos que deberemos incluir son:

- Del directorio dist:
  - solr-solrj-4.10.4.jar
- Del directorio dist/solrj-lib:
  - commons-io-2.3.jar
  - httpclient-4.3.1.jar
  - httpcore-4.3.jar
  - httmine-4.3.1.jar
  - noggit-0.5.jar
  - wstx-asl-3.2.7.jar
  - zookeeper-3.4.6.jar
  - No incluimos el archivo slf4j ya que los descargaremos más tarde ya que este produce errores en la creación del log.

Ahora necesitaremos descargar los archivos necesarios de slf4j. Podremos encontrarlos en esta dirección: http://mvnrepository.com/artifact/org.slf4j/slf4j-api. En nuestro caso descargamos la versión más moderna disponible en ese momento, es decir, la versión 1.7.15. Una vez descargado los archivos que deberemos incluir a la librería de eclipse son:

- slf4j-api-1.7.15.jar
- slf4j-simple-1.7.15.jar

También deberemos descargar el commons logging de la siguiente página https://commons.apache.org/proper/commons-logging/. Deberemos incorporar también el archivo “commons-logging-1.2.jar” que encontraremos dentro del comprimido.

Anexo VII (Contenido del CD)

Memoria del TFG:

Memoria.pdf: Documento en pdf de la memoria del proyecto.

Software desarrollado:

Solr: Carpeta que contiene el HOME de Solr. Dentro de ella se encuentra la colección para la utilización de la aplicación.

LogMining.zip: Archivo zip que contiene el código fuente de nuestra aplicación. Es una exportación del proyecto de Eclipse.

LogMining.war: Aplicación web en formato war de nuestra aplicación creada.

Instalador

Apache-tomcat-8.0.35.exe: Ejecutable para la instalación de Tomcat.

Bootstrap-3.3.6-dist.zip: Archivo comprimido con el código fuente de bootstrap v3

Commons-logging-1.2-bin.zip: Archivo comprimido con el código fuente de Apache commons logging.

Reflections-0.9.9-RC1-uberjar.jar: API para la utilización de reflections con Eclipse.

Slf4j-1.7.15.zip: Archivo comprimido con el código fuente de slf4l para el logging del servidor.

Solr-4.10.1.zip: Archivo comprimido con todo el código fuente de solr descargado directamente desde la web.

Manual

 Manual de instalación.pdf: Manual en pdf con los pasos a seguir para la instalación de todos los sistemas.

 **Archivos de log:** Esta carpeta contendrá un archivo ejemplo de log empleado. Estará modificado ya que son archivos confidenciales de HP.