Sistemas de recomendación sensibles al contexto con bases de datos NoSQL

Proyecto Fin de Máster en Sistemas Inteligentes

Autor: Eddy Rodriguez Vargas
Director: Pedro Antonio González Calero
Co-director:Guillermo Jiménez Díaz

Máster en Investigación en Informática
Facultad de Informática
Universidad Complutense de Madrid

Febrero 2013

Documento maquetado con TEXIS v.1.0+.

Este documento está preparado para ser imprimido a doble cara.

Sistemas de recomendación sensibles al contexto con bases de datos NoSQL

Informe técnico del departamento
Ingeniería del Software e Inteligencia Artificial
02/20103

Dirigida por el Doctor
Pedro Antonio González Calero y Guillermo Jiménez Díaz

Máster en Investigación en Informática
Facultad de Informática
Universidad Complutense de Madrid

Febrero 2013

ISBN 666-666-666-666

A mi familia y amigos.

Agradecimientos

De desagradecidos está el infierno lleno.
Refranero multilingüe CVC.

A mis padres

A mis papás Hugo y Alicia, a ellos en especial por haberme apoyado en todo momento en la distancia, por sus consejos por sus valores que me han transmitido y la educación que me han permitido tener, por hacer que siempre tenga una motivación constante en la vida que me ha permitido ser una persona de bien, pero más que nada, por su amor.

A todos mis amigos

A los que están siempre cuando se los necesita, a los que los veo y hablo en la distancia. A mis amigos de Madrid que con este proyecto me he llegado a perder con muchos de ellos. A mis amigos de Bolivia, los de infancia, los de colegio y los de la universidad. Finalmente a mis amigos de Chile a mis nuevo amigos en Barcelona, y los que se encuentran repartidos por todo el mundo y que mantengo contacto lejos y siempre han estado presente.

Quiero agradecer de igual manera a mi hermano mayor Franklin que en la distancia siempre me ha estado apoyando y finalmente a mis tutores Pedro y Guillermo por la ayuda y el apoyo en este proyecto.

Abstract

The present project represents a general architecture for a context-aware recommender system. This architecture pretends take advantages from the main features of a NoSql database, one important feature is the use of documents in collections, it will be used to store contextual information and achieve pre-filtering context information.

This work shows that through this architecture we pretend to have new way and alternative to contextualize rating information, with help of NoSql database and its collections. Also present an implementation of the architecture and design of a prototype that uses experimental comparative tests, these experiments are performed between two databases systems, NoSql versus a traditional one.

Keywords: Context-Aware Recommender Systems, NoSQL, Collaborative Filtering, Pre-filter, Architecture, CARS, non-relational databases.

Resumen

No somos ciudadanos, somos
consumidores de todo: de cosas, de
personas, de sentimientos, de intereses.
No disfrutamos, sólo devoramos,
poseemos y olvidamos.

Josep Marc Laporta

El presente trabajo muestra una arquitectura general para un sistema de recomendación contextual. Dicha arquitectura pretende sacar ventaja de las características de las bases de datos NoSql, una de esas características es la utilización de colecciones de documentos, las mismas que serán empleadas para poder almacenar información contextual, logrando así un pre-filtrado contextual.
En este trabajo se pretende mostrar que a través de dicha arquitectura se tenga una nueva forma y alternativa para poder contextualizar la información de rating, gracias a la ayuda de un gestor de base de datos NoSql y sus colecciones. También se presentara una implementación de dicha arquitectura, así como el diseño de un prototipo que sirve de experimentación, se realizan pruebas comparativas entre dos gestores de bases de datos, uno tradicional versus un NoSql.

Palabras clave: Sistema de recomendación sensible al contexto, NoSql, Filtrado Colaborativo, Pre-filtrado, Arquitectura, CARS, Bases de datos no relacionales.

Índice

Índice de figuras

Capítulo 1

Introducción

Tengo una pregunta que a veces me tortura: estoy loco yo o los locos son los demás.

Albert Einstein

1.1. Motivación

Existe hoy una gran cantidad de sitios especializados en ofrecer productos y servicios a través internet, ofertando millones de sus productos o servicios para su consumo convirtiéndose en un caos de información cuando se necesita realizar una adquisición eligiendo entre todas las opciones existentes. Los sistemas de recomendación surgen como solución a este problema.

Básicamente un sistema de recomendación recibe información del usuario sobre sus gustos y preferencias de un producto o servicio en particular que el usuario se encuentra interesado y le recomienda aquéllos cercanos a sus necesidades. Incorporar de una tecnología de recomendación puede aumentar las ventas en sistemas de comercio electrónico[21], pero dotar a los sistemas de recomendación de nuevas técnicas que personalicen las recomendaciones más allá de las simples peticiones de productos por parte del usuario, constituye un importante campo de investigación actualmente en el ámbito de mercados en red con grandes volúmenes de contenidos. Los sistemas de recomendación sensibles al contexto nos presentan un enfoque multidimensional, ayudando así en la calidad de las recomendaciones en ciertos entornos[15]. Ofreciéndonos nuevos enfoques para poder manejar información contextual y añadir capacidades contextuales a los sistemas de recomendación. Traen consigo unos paradigmas de inclusión de contexto a los sistemas de recomendación clásicos ayudando a poder realizar filtrados de contexto a la hora de hacer una recomendación[1]. Por otra parte para poder brindar de recomendaciones de calidad se debe tener una gran cantidad de información pero

por otro lado la mayoría de los algoritmos que se utilizan en estos sistemas realizan una gran cantidad de cálculos, por lo que el costo de procesamiento puede ser muy elevado.

Puede ocurrir un caso en que un usuario tenga que esperar un tiempo prolongado para obtener una recomendación y dependiendo de las características de hardware y de software del sistema se puede llegar a tener cuellos de botella. Propuestas como particionar el conjunto de elementos se han implementado[3], reduciendo el espacio de dimensiones de los ítems, pero traen consigo problemas de escalabilidad, ya que al aumentar el número de ítems en el sistema, se hace difícil mantener una performance razonable en las recomendaciones.

El continuo incremento de datos y la continua necesidad de procesar grandes cantidades de datos en tiempos cortos[29] hacen que herramientas como los sistemas almacenamiento NoSql dan una opción como ayuda a este tipo de problemas ya que pueden manejar enormes cantidades de información, son más simples, no generan cuellos de botella, son ejecutadas en máquinas con coste reducido y en mayor número gracias a su nivel de escalabilidad[31]. Una de las características importantes en estos sistemas NoSql y de la cual sacaremos ventaja para este proyecto es el hecho de poder crear colecciones de documentos en tiempo de ejecución, sin la necesidad de crear estructuras de datos como se haría como una tabla en un sistema de base de datos relacional. Esta ventaja será utilizada posteriormente para la creación de un pre-filtrado contextual.

Framework como Lenskit que nos ofrecen un amplia variedad de algoritmos de recomendación, de fácil uso e integración en cualquier sistema y manejo de grandes de datos[26], hacen que este proyecto tome un poco de lo mejor de cada tecnología para hacer un modelo y su implementación de un sistema de recomendación basado sensible al contexto.

1.2. Objetivo del trabajo de investigación

El principal objetivo de este trabajo es crear una arquitectura para un sistema de recomendación contextual, haciendo uso de un gestor de base de datos NoSql.

1.3. Objetivos específicos

Entre los objetivos específicos se tiene a implementar dicha arquitectura para su posterior experimentación y uso. También se pretende como un ob-

jetivo el demostrar que con la utilización de este tipo de sistemas NoSql se pueda proporcionar un sistema de recomendación más eficiente que con la utilización de un sistema de base de datos clásico.

Para poder cumplir con el objetivo planteado, primeramente se hará un estudio del arte, este estudio incluirá analizar temas sobre sistemas de recomendación y sistemas de recomendación sensibles al contexto. Por otra parte se estudiara sobre sistemas de gestión de bases de datos de tipo NoSQL.

También requerimos que la implementación de la arquitectura deba de ser adaptable a cualquier entorno, principalmente a entornos de comercio electrónico, esto será posible por su propia arquitectura que tendrá el mismo, esta arquitectura incluye usar una base de datos orientada a documentos, esto no implica tener documentos de texto, sino tener una estructura de documentos que empleen JSON[1] como representación de la información. La idea es usar estos documentos como información contextual. También el uso de esta estructura permitirá la creación dinámica de información contextual, lo que ayudara al sistema para poder separar la información de un determinado contexto en colecciones de documentos separados.

La selección de una base de datos NoSql para este proyecto está dada por sus características principales que tienen este tipo de gestores, entre ellos se encuentran el manejo de grandes cantidades de datos, escalabilidad y rendimiento entre otros[22]. Estos tipos de base de datos tienen las características necesarias para resolver este tipo de problemas. Este tema será tratado con más detalle en el capítulo 4.

Además al tratarse de información de rating y contextos, que no son estructuras de sistemas de tipo CRM[2] o ERP[3] o similares que dependen mucho de la integridad de los datos, sino que son estructuras simples, se aprovecha esta característica con las bases de datos NoSql. Por todo esto se pretende hacer una arquitectura que emplee una capa de datos simple, de fácil manejo y que posea características como el manejo de grandes cantidades de datos.

La implementación de la arquitectura será hecha principalmente con soft-

[1] Acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos.

[2] Término en inglés "customer relationship management". Es un modelo de gestión de toda la organización, basada en la orientación al cliente, el concepto más cercano es marketing relacional.

[3] Por sus siglas en inglés, "Enterprise Resource Planning". Son Sistemas de Información Gerenciales que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía en la producción de bienes o servicios.

ware Open-Source[4], tratando de engranar cada módulo que será propuesto en la arquitectura con el software seleccionado.

La puesta a prueba del sistema de recomendación se la realizara a través de experimentos de comparación entre dos gestores de bases de datos. Por un lado un gestor de bases de datos NoSql y por otro lado un gestor de base de datos tradicional entidad relación.

Se realizara un prototipo de pruebas de modo que se pueda ver gráficamente en tiempo de ejecución los tiempos de respuesta de ambos gestores de base de datos en las llamadas a recomendaciones.

1.4. Estructura de la memoria

La estructura de esta memoria se encuentra divida en seis capítulos, y cada capítulo estará dividida en secciones en las que nos iremos introduciendo en un principio en el estado del arte (capítulo 2) con todo lo referente a los estudios actuales que hay sobre los temas involucrados en la implementación de este sistema de recomendación contextual. En el capítulo 3, haremos la presentación del diseño general de la arquitectura del sistema, donde se presentan todas las capas que comprenden la arquitectura así como cada uno de sus módulos contenidos en cada capa. El capítulo 4 nos presenta la implementación de dicho diseño de la arquitectura, utilizando diferentes tipos de tecnología para la construcción y el desarrollo del mismo.

En el capítulo 5 se realizará una experimentación de comparativa del sistema de recomendación entre dos gestores de bases de datos, dicha experimentación se detallará y evaluará para su posterior conclusión en el capítulo final (capítulo 6).

[4]Es el término con el que se conoce al software distribuido y desarrollado libremente.

Capítulo 2

Estado del arte

El saber no ocupa lugar.

Refranero multilingüe - CVC

2.1. Sistemas de recomendación

Un sistema de recomendación se podría definir como "aquél sistema que tiene como principal tarea seleccionar ciertos objetos, de acuerdo a los requerimientos del usuario, dado que estos objetos están almacenados y caracterizados, en base a sus atributos". Desde la óptica del marketing, [21], exponen que el término de sistema de recomendación es el resultado de la evolución y la ampliación del algoritmo del filtrado colaborativo.

La automatización del proceso "boca a boca" es uno de sus objetivos principales, donde personas recomiendan productos y servicios a otros, esto parte por buscar opiniones de personas que tengas los mismos gustos o están acostumbradas a recibir recomendaciones cuando no se tiene opciones por elegir.

El procedimiento de los sistemas de recomendación se puede resumir en que primeramente los usuarios dan una valoración sobre elementos del sistema, estos elementos pueden ser productos de una tienda electrónica y la valoración puede ser numérica o binaria. Con esas valoraciones lo que se quiere es predecir nuevos elementos en función de los ya conocidos para un usuario activo. De tales predicciones calculadas se toman aquellas que tienen los valores más altos para dar una recomendación. El sistemas comercio electrónico estos sistemas recomiendan listas de productos a evaluar, sirven como soporte al cliente para la presentación de productos vendidos, permiten la creación de tiendas personalizadas para un cliente.

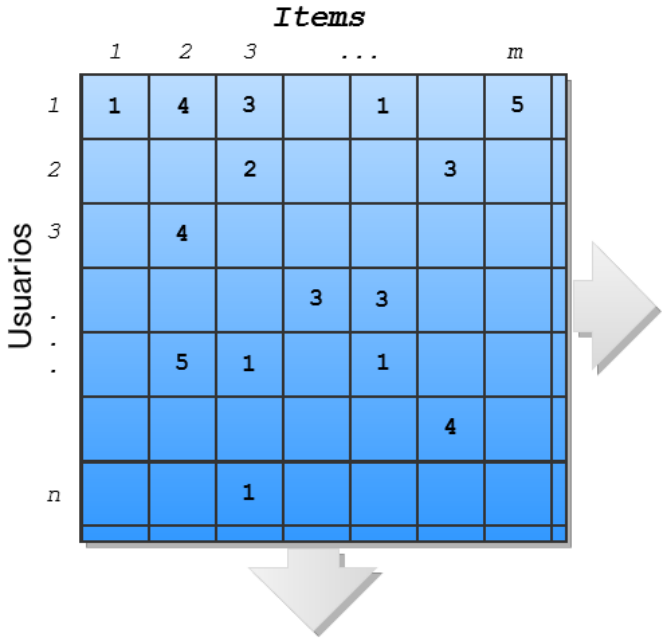

Figura 2.1: Matriz de ratings de usuario, donde cada celda R(u,i) corresponde a el rating del usuario u para un ítem i. La tarea principal es predecir rating R(a,i) para un usuario activo.

Un sistema de recomendación considera un conjunto de usuarios U y un conjunto de ítems I de cualquier tipo de producto ya sea películas, libros o música. Para generar recomendaciones, se necesita que el usuario u que pertenece al conjunto U tenga alguna iteración con un subconjunto de ítems I. Estas iteraciones pueden ser de muchos tipos, entre ellos pueden estar el rating, consumo o una combinación de ambos.

El rating, sucede cuando un usuario ha asignado una valoración sobre un ítem. Entre las más comunes existen las binarias ("me gusta", "no me gusta") o la gradual (las famosas estrellas de valoración), hoy por hoy es el caso más estudiado en la literatura, (ver figura 2.1).

El consumo, ocurre cuando un usuario ha consumido o adquirido algún producto, también sucede cuando se visita un video, un artículo o una página en particular.

2.1.1. Mecanismos de recomendación

Entre los principales mecanismos para generar recomendaciones se encuentran: el basado en contenido, el colaborativo y el basado en conocimiento. En las recomendaciones colaborativas se pueden realizar recomendaciones sorpresivas de elementos, todo en función de la información compartida en la base de datos; mientras que la recomendación basada en contenidos sólo se puede realizar recomendaciones de elementos de datos explorados en el pasado del usuario.

- Recomendación basada en contenido: Es un método de recomendación que parte de propiedades en común entre perfiles de usuario y se relaciona con características de contenido, es información tomada de registros históricos.

  "Aquél sistema en el que las recomendaciones son realizadas basándose solamente en un perfil creado a partir del análisis del contenido de los objetos, que el usuario ha evaluado en el pasado."[5]. En otras palabras, el sistema hará recomendaciones solamente de productos que tengan una importancia elevada para el usuario, de esa forma una recomendación va a depender de la representación de los datos y la descripción de las preferencias de los usuarios. Sólo recomendará aquellos elementos en que el usuario mostro interés en el pasado, lo cual generalmente se considera como una limitación al método.

  Este método aborda diferentes áreas dentro de la inteligencia artificial como ser la recuperación de la información con enfoques como los espacios vectoriales; el aprendizaje automático con las redes bayesianas y la clasificación con redes neuronales; y finalmente el clustering[4]. Entre todos estos el más preferido son los espacios vectoriales por su simplicidad.

- Recomendación colaborativa: Pueden definirse como: "aquél sistema en el que las recomendaciones son hechas basándose solamente en los términos de similitud entre los usuarios",[5]. La idea principal de este método es que la tarea de realizar las recomendaciones es llevada a cabo por los mismos usuarios. Con este método es posible afirmar que el mismo se adapta fielmente en el proceso natural de recomendar que es hecho por las personas, logrando automatizarlo[28].

  La manera en que se automatiza este proceso natural es haciendo comparaciones de las preferencias o gustos de los usuarios en un campo de

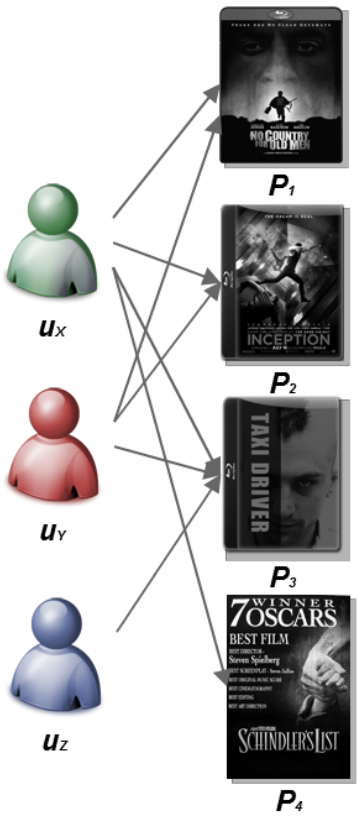

Figura 2.2: Mecanismo de recomendación colaborativa basada en usuario.

aplicación determinado, el propósito es de hallar grupo de usuarios con características similares o es lo mismo decir que se puede trabajar con un grupo de ítems y ver las características similares de los mismos.

El proceso se puede resumir en los siguientes pasos tomando en cuenta que el sistema mantiene el perfil del usuario, el mismo que contiene las evaluaciones positivas y negativas sobre los objetos que han sido de interés. Con este perfil se hace comparaciones con los perfiles de otros usuarios para determinar el grado de similitud que se tienen entre ellos. Una vez que se tiene los perfiles con mayor similitud se recomiendan objetos que el usuario no haya evaluado.

Para poder entender mejor este proceso tomemos un ejemplo de filtrado colaborativo basado en usuarios para un sistema de recomendación de películas, (ver figura 2.2). Suponiendo que el usuario "X" ha realizado una evaluación positiva de las siguientes películas: "P1", "P2", "P3"

y "P4". Ahora, por otra parte el usuario "Y" ha evaluado de la misma las películas que el usuario "X", con la excepción de "P4". Entonces, el sistema de ser capaz de notar que ambos usuarios tienen en cierta medida preferencias similares, porque podría recomendar "P4" al usuario "Y".

Para poder realizar toda esta tarea se pueden distinguir dos tipos generales de algoritmos de Filtrado Colaborativo. Aquellos basados en memoria, en el que para el cálculo de una predicción se toma en consideración la similitud del usuario actual y todos los demás usuarios del sistema. El segundo tipo de algoritmo son los basados en modelo, aquí se utilizan los ítems de los usuarios para entrenar un modelo probabilístico, este modelo es usado para generar predicciones. Este es un caso en que no se toma en cuenta de una manera explícita la similitud que existe entre el usuario activo y todos los demás usuarios.

- Recomendación basada en conocimiento: los sistemas de recomendación basados en conocimiento realizan inferencia entre las necesidades y preferencias de cada usuario para sugerir recomendaciones[16].

  Estos tipos de sistema no dependen de grandes cantidades de información sobre ítems y usuarios particulares, lo único que necesitan es tener un conocimiento general sobre el conjunto de objetos y un conocimiento informal de las necesidades del usuario. El problema principal en este tipo de sistemas es que requieren un gran esfuerzo humano para realizar las recomendaciones mediante todo tipo de heurísticas de inferencia.

- Híbridos: Para finalizar, los sistemas de recomendación híbridos brindan una solución a las limitaciones de las dos anteriores, lo que se quieres es aprovechar los beneficios de ambos métodos. Sus principales variantes son[9]: ponderado (weighted), switching, combinado, combinación de características, cascada, potenciación de características y meta nivel (meta-level).

2.2. Sistemas de recomendación sensibles al contexto

2.2.1. Definición de contexto

Es un concepto multifacético, estudiado a través de una gran variedad de investigaciones entre las cuales se hallan la inteligencia artificial y la compu-

tación ubicua, la psicología, la ciencia cognitiva y la lingüística entre otras. Dado que el contexto ha sido estudiado en múltiples disciplinas, cada una de ellas tiene su propia visión que es algo diferente de otras disciplinas y es más específica que la definición del diccionario estándar: " Entorno físico o de situación, ya sea político, histórico, cultural o de cualquier otra índole, en el cual se considera un hecho"[14]. Incluso, dada la complejidad y el carácter multifacético del concepto, se pueden llegar a presentar más de ciento cincuenta definiciones de contexto para los diferentes campos, por lo que es difícil encontrar una definición pertinente para satisfacer cualquier disciplina. De acuerdo con [12], el contexto es cualquier información que puede ser utilizado para caracterizar la situación de las entidades (persona, lugar u objeto) que se consideran relevantes para la interacción entre un usuario y una aplicación, incluyendo el usuario y el aplicación sí mismos. El contexto es normalmente la ubicación, identidad y el estado de personas, grupos y objetos físicos.

En la bibliografía relacionada con este tipo de sistemas, el contexto se definió inicialmente como la ubicación del usuario, el identidad de las personas cercanas al usuario, los objetos a su alrededor, y los cambios en estos elementos. Otros incluyen la fecha, la temporada, y la temperatura[10].

La mayoría de los enfoques existentes para los sistemas de recomendación se centran en lo más relevantes para los usuarios individuales y no así tomando en consideración cualquier información contextual como la hora, el lugar y la compañía de otras personas. Un ejemplo podría ser un sistema de recomendación para un portal de viajes[20], el mismo proporcionaría una recomendación de vacaciones en el invierno, que puede ser muy diferente en el verano.

Dado que nos centramos en los sistemas de recomendación en este trabajo y desde el general concepto de contexto es muy amplio, vamos a tratar de centrarnos en aquellos campos que son directamente relacionado con los sistemas de recomendación, como el comercio electrónico, bases de datos y sistemas sensibles al contexto móviles.

2.2.2. Modelado de la información contextual en sistemas de recomendación sensibles al contexto

Como en los sistemas de recomendación tradicionales, el proceso suele comenzar la especificación de puntuaciones que son hechas por los usuarios o implícitamente se deduce por el sistema. Una vez que estas puntuaciones iniciales se especifican, un sistema de recomendación trata de estimar la función de clasificación R: el usuario x ítem $\rightarrow$ rating[2]. Una vez que dicha función R se estima, un sistema de recomendación puede recomendar

el elemento de mayor audiencia o los ene elementos mejor valorados, aquí los elementos son un conjunto totalmente ordenados.

A estos sistemas tradicionales también se les suele llamar de dos dimensiones (2D) ya que consideran el conjunto de usuarios e ítems en el proceso de recomendación. Esto se reduce al problema de la estimación de los ítems que no se han visto por un usuario. Esta estimación se basa generalmente en las puntuaciones que fueron otorgadas por este usuario a otros ítems.

Mientras que una cantidad sustancial de la investigación se ha realizado en el área de sistemas de recomendación, la gran mayoría de los enfoques existentes se centran en recomendar artículos a los usuarios o usuarios a los artículos y no toman en la consideración de cualquier tipo de contexto adicional, tales como tiempo, lugar, la compañía de otras personas u otros. Es por esta razón que los sistemas de recomendación que son sensibles al contexto, que se ocupan de modelar y predecir los gustos del usuario y preferencias mediante la incorporación de información contextual. Esta información contextual puede ser de diferentes tipos, como el tiempo, la ubicación, compañero, el propósito de una compra, etc.[2] Además, cada tipo contextual puede tener una estructura complicada que refleja la complejidad de la información contextual.
Esta complejidad de la información contextual puede adoptar muchas formas diferentes, desde una estructura jerárquica que se puede ser representado como un árbol.

Con la presencia de la información contextual y siguiendo los diagramas de la figura 2.3, se comienza con los datos que tiene la forma de $U \times I \times C \times R$, donde U es el usuario, I los items, C es la información contextual y finalmente R es el valor de la recomendación. Más específicamente, el proceso de recomendación que es sensible al contexto puede tomar una de las tres formas.

- `Pre-filtrado`: En este paradigma, lo que se hace es que, sobre el contexto actual C, se utiliza para seleccionar o construir el conjunto pertinente de los registros de datos (es decir, las puntuaciones). A continuación, los ratings se pueden predecir con cualquier sistema de recomendación tradicional 2D con el conjunto de datos seleccionados.

  Un ejemplo de un pre-filtrado de datos contextual de un sistema de recomendación sería: dada una persona que desea ver una película en un día domingo, entonces solo los datos de tipo domingo o fin de semana son utilizados para recomendar películas.

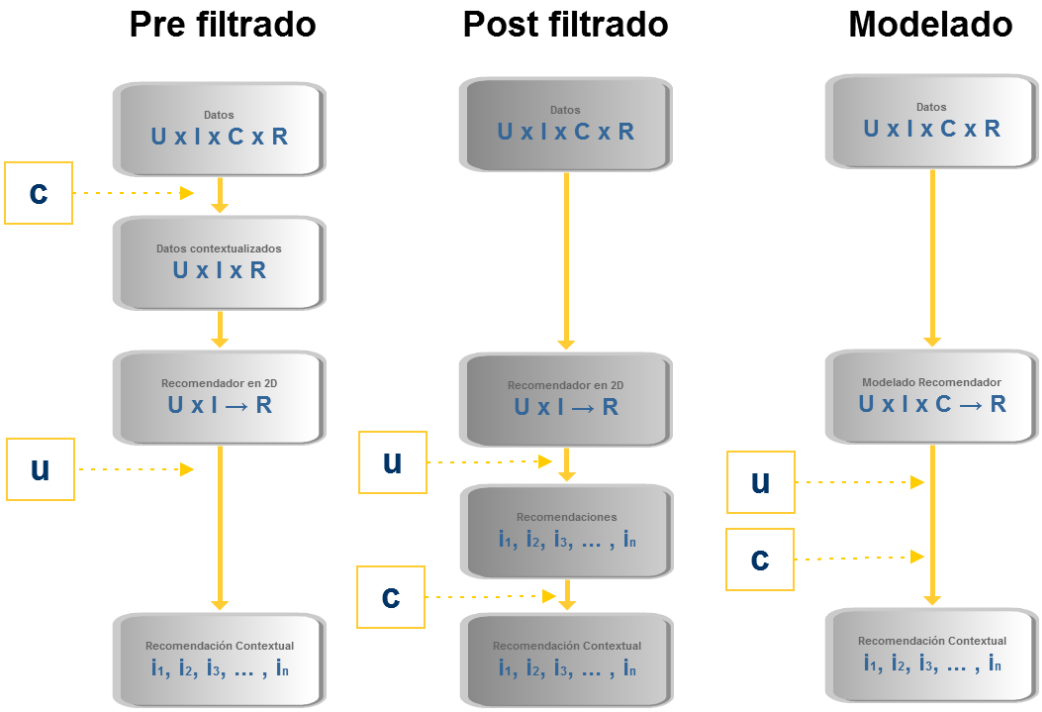

Figura 2.3: Paradigmas para la incorporación de contexto en sistemas de recomendación.

Por lo tanto, un beneficio importante sobre este paradigma es que este se basa en toda la investigación anterior, los sistemas de recomendación tradicionales(los que son en dos dimensiones) son directamente aplicables en este caso multi-dimensional. Por lo que la función de predicción multi-dimensional queda expresada a través de una función de predicción en 2D.

Este trabajo se basara en la utilización de este tipo de paradigma para la construcción del sistema de recomendación contextual. El prefiltrado contextual será empleado a través de las colecciones que trae consigo la base de datos NoSql.

- `Post-filtrado`: Para este tipo de paradigma, la información contextual es inicialmente ignorada, el proceso es comenzar como en un sistema de recomendación tradicional en 2D, tras obtener el conjunto resultante de las recomendaciones, este es ajustado, es decir que el resultado será contextualizada utilizando la información contextual.

- `Modelado contextual`: En este paradigma recomendación, la información contextual se usó directamente en la técnica de modelado como parte de la estimación de valoración, es decir que la función se convierte en una modelo multidimensional, donde no hay pre-filtrado ni pos-filtrado, la información contextual es usada como parte del todo.

2.2.3. Obtención de la información contextual

- `De manera explícita`: Esto implica, el ir directamente a los usuarios y obtener la información textual, ya sea preguntando directamente u obtener esta información por otros medios.
  Por ejemplo, en un sitio web puede obtener información contextual al pedir a una persona que llene un formulario o responder algunas preguntas concretas antes de proporcionarle acceso a la web determinada[17].

- `De manera implícita`: A partir de los datos o del entorno, estos pueden ser, como un cambio en la ubicación del usuario que puede ser detectado por un GPS[1] en un dispositivo móvil. Se pueden llegar a deducir el contexto a través de métodos como la minería de datos o métodos estadísticos.
  Con el fin de dar con esa información contextual, es necesario construir un modelo predictivo, un clasificador, y entrenar a los datos correspondientes. El contexto también puede ser obtenido por parte de logs[2], links que va visitando o hábitos de navegación[30].

2.2.4. El factor del tiempo

Dependiendo de si los factores contextuales cambian en el tiempo o no, tenemos las siguientes dos categorías: estática y dinámica.

- `Estático`: La información contextual y su estructura sigue siendo la misma en el tiempo. Como ejemplo, en el caso de recomendar una compra de un determinado producto, tal como una camiseta, que puede incluir información contextual como, el propósito de compra, tiempo. Esta información quedaran estáticas durante la toda la vida útil de la recomendación de compra.

- `Dinámica`: Es el caso en que la información contextual puede cambiar de alguna forma. Por ejemplo, el sistema de recomendación puede darse cuenta que la marca de un producto ya no es relevante para la compra,

[1] Global Positioning System: sistema de posicionamiento global.

[2] Para los profesionales en seguridad informática es usado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué (who, what, when, where y why) un evento ocurre para un dispositivo en particular o aplicación.

entonces este podría descartar dicha información contextual. Además la estructura de la información contextual puede variar en el tiempo, como la adición de nuevas categorías.

La mayor parte de los trabajos previos sobre sistemas de recomendación sensibles al contexto siguen una obtención de información explicita con un factor de tiempo estático, como lo es este actual trabajo.

2.3. Lenskit (Framework para sistemas de recomendación)

Lenskit[3] es una herramienta de algoritmos de filtrado colaborativo y un conjunto de herramientas para la evaluación. Lenskit está basado en Java, proporciona un API[4] común para algoritmos de recomendación que tiene una implementación altamente modular.
Ofrece un marco común, reutilizable en aplicaciones, y claro, legible con la implementación de algoritmos que emplean las mejores prácticas en lo que respecta a las estrategias de implementación, puesta a punto, y normalizaciones. Lenskit está bajo un desarrollo activo. Está construido y desplegado a través de Maven, el código fuente también se puede extraer y utilizar en la mayoría de entornos de desarrollo Java. LensKit es capaz de procesar grandes cantidades de datos, ampliamente los datos disponibles, tales como los conjuntos de Movielens 10M. Al estar implementado en Java su código es fácilmente legible por una amplia audiencia de científicos de la computación, y desde que se ejecuta en la JVM[5], LensKit también se puede utilizar en muchos otros lenguajes de programación como el Groovy, Scala y Ruby[13].

2.3.1. Diseño de Lenskit

La modularidad en muchos algoritmos de recomendación, naturalmente, compuesta por varias piezas, tales como los normalizadores, las funciones de similitud y las reglas de predicción. Muchos de estos componentes no son específicos para un determinado algoritmo pero pueden ser reutilizados en otros algoritmos.

[3] http://lenskit.grouplens.org

[4] Application Programming Interface(Interfaz de programación de aplicaciones): es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

[5] Java Virtual Machine(Una máquina virtual Java): es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial, el cual es generado por el compilador del lenguaje Java.

2.3.2. Core API

Construida por las interfaces que son fundamentales en LensKit, las fundamentales son el ItemScorer y el ItemRecommeder, que proporcionan apoyo para las tareas de predicción y recomendación respectivamente[13].

2.3.3. Capa de acceso de datos LensKit

Incorpora una simple abstracción para el acceso de datos, las implementaciones pueden ser fácilmente respaldadas por archivos planos, bases de datos, arquitecturas de persistencia de datos como Hibernate, y almacenes de datos no tradicionales.

2.3.4. Implementaciones y componentes

LensKit proporciona tres implementaciones de uso común: el user-user, ítem-item y el SVD. Estos algoritmos están partidos en componentes que pueden ser recombinados y remplazados cada uno de ellos de manera independiente.[13]

2.4. Bases de datos NoSql

2.4.1. Definición

El termino NoSql no es perfecto, es algo que la mayoría de los proveedores de soluciones de este tipo está de acuerdo. La mayoría concuerda que el "No" significa "no sólo", es decir compensar las limitaciones técnicas que llevan consigo la mayoría de bases de datos relacionales. No significa un rechazo a un software en particular.[8]

Cuando hablamos de NoSQL no se refiere únicamente a un tipo de bases de datos en concreto, sino más bien a diferentes soluciones para almacenar datos, cuando las bases de datos relacionales generan problemas. Nos referimos a una multitud de bases de datos que intentan solventar las limitaciones que el modelo relacional se encuentra en entornos de almacenamiento masivo de datos, y concretamente en las que tiene en el momento de escalar, donde es necesario disponer de servidores muy potentes y de balanceo de carga.

Las bases de datos NoSQL son sistemas de almacenamiento, que no cumplen con el esquema tradicional entidad-relación, no imponen una estructura de datos en forma de tablas y relaciones entre ellas, en ese sentido son más flexibles, ya que suelen permitir almacenar información en otros formatos como clave-valor, Mapeo de Columnas, Documentos o Grafos.
Es necesario saber que este tipo de base de datos no sustituye a los actuales

gestores de base de datos, sino que surgen por otra necesidad. Una necesidad de rendimiento extremo.

Básicamente los gestores de bases de datos comunes se han caracterizado siempre por ser transaccionales llevados por el acrónimo ACID [6], mientras que NoSQL es a veces caracterizado por el acrónimo BASE[7].[24]

NoSQL se caracterizan por: responder a las necesidades de escalabilidad horizontal que tienden cada vez más empresas, manejan grandes cantidades de datos, no generan cuellos de botella, el escalamiento es sencillo y además existe una variedad este tipo de sistemas que pueden ser utilizados para diferentes tipo de proyectos. Además utilizan sobre todo el uso de memoria en vez del disco como la principal ubicación de escritura, y los almacenes de datos NoSQL aprovechan típicamente particiones horizontales[29].

En una base de datos relacional, realizar una consulta implica: convertir, preparar, optimizar, procesar, leer del disco, ejecutar, etc. Varios pasos hasta poder llevar a cabo una consulta. En NoSQL se evita hacer todo esto, puesto que en tiempo de ejecución (según el tipo y el API) este se encargue de acceder directamente a los datos, accediendo al más bajo nivel y teniendo lo que nos interesa en la memoria y no en el disco obtenemos tasas de rendimiento muy elevadas, sobre todo en volúmenes de datos grandes, como cuando se trabaja información de recomendación.

2.4.2. Un poco de historia

Bases de datos relacionales se desarrollaron en una época diferente, con diferentes tecnológica, dando lugar a un diseño que era óptimo para el despliegue típico de prevalencia en ese momento. Durante los últimos años las bases de datos relacionales dan soporte a la gran mayoría de las aplicaciones. Con los años se han ido mejorando, normalizándolas en todo lo posible, escalándolas según la demanda e incorporando sistemas de persistencia, en general se ejecutan en una organización de hardware más caro.

Hasta no hace mucho, cuando se tenía un tráfico elevado de visitas por segundo y la infraestructura no soportaba ese tráfico, se optaba por arquitecturas en las que cada cierto tiempo, se generaba ficheros HTML para que los mismos fueran despachados por servidores web básicos que no tenían q

[6](Atomicity, Consistency, Isolation, and Durability) Atomicidad, Consistencia, Aislamiento y Durabilidad: Son conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción.

[7]Basically Available (básicamente disponible), Soft state(permite que la consistencia de datos no sea altamente requerida), Eventually consistent(NoSQL asegura que en algún punto en el tiempo los datos asumen un estado de consistencia)

acceder a gestores de bases de datos ni consumir tiempo de procesador ni memoria.

En la actualidad, empresas como Twitter, Facebook tienen un tráfico millones de veces superior, existen aplicaciones RIA[8] que en las que se accede simultáneamente por infinidad de usuarios, es así que la tecnología existente se queda un poco corta.

Las bases de datos NoSQL rompen con la barrera del rendimiento, ya sea según su tipo, su estrategia de ejecución es distinta y aunque parece que es una tecnología nueva, este concepto ha existido durante muchos años atrás, es solo que NoSQL ha atraído mucha atención en los últimos años, debido principalmente a la producción de las implementaciones de gran renombre y soporte para algunas investigaciónes[18]. Ejemplos como Dynamo de Amazon y BigTable de Google son algunas de las mejores implementaciones conocidas.

2.4.3. Rendimiento

Hasta ahora los problemas de rendimiento se han intentado solucionar mediante escalabilidad vertical, es decir que se añaden más recursos a un solo nodo en particular, esto puede ser el añadir memoria o un disco duro más rápido a un servidor. Entonces se pensó que la solución puede ser la escalabilidad horizontal, una manera de agregar más nodos a un sistema, pero los actuales gestores de bases de datos relacionales no se adaptan del todo a este modelo.

A diferencia de lo anterior, una de las principales característica de las bases de datos NoSql es que están pensadas para manipular grandes cantidades de información de manera muy rápida. Almacenando la información directamente en memoria y usando el disco como herramienta de persistencia, están preparadas para escalar horizontalmente sin perder rendimiento. Las tablas o entidades del modelo relacional, no existen, la información es almacenada de forma diferente, generalmente como clave-valor, como una tabla en la que las columnas son dinámicas, las mismas pueden cambiar sin perder la agrupación de la información, se puede cambiar la estructura de la información dinámicamente sin tener que hacer un rediseño del mismo.

[8] Rich Internet applications(aplicaciones de Internet enriquecidas): son aplicaciones web que tienen la mayoría de las características de las aplicaciones de escritorio tradicionales. Estas aplicaciones utilizan un navegador web estandarizado para ejecutarse y por medio de complementos o mediante una máquina virtual se agregan las características adicionales.

2.4.4. Tipos de bases de datos NoSql

El tipo depende mucho de cómo se almacena la información, existen cinco tipos de bases de datos NoSQL:

- `Key-Value`: Es la forma más típica, es como un HashMap[9] en el cual cada elemento está identificado de manera única por una llave, lo que lleva a una recuperación de información rápida. Son muy eficientes para lecturas y escrituras.

- `Basada en documentos`: En este tipo de base de datos los datos son almacenados como un documento, su estructura varía según el tipo, pero pueden ser JSON o XML. Tienen similitud con las bases de datos Key-value, pero con la diferencia que el valor es un fichero que puede ser entendido.

- `Orientadas a Grafos`: Aquí la información es almacenada como grafos donde las relaciones entre los nodos son lo más importante. Su utilidad sirve para representar información en redes sociales. Este tipo de bases de datos sólo son aprovechables si la información en cuestión se puede representar de una manera fácil como una red. Este tipo de base de datos es capaz de obtener rendimientos altísimos en consultas sobre información relacionada.

- `Orientadas a columnas`: Los datos son almacenados en columnas en lugar de filas. Con este cambio se gana velocidad en la lectura, ya que si se requiere consultar un número reducido de columnas, es muy rápido hacerlo. Este tipo de base de datos son deficientes a la hora de realizar escrituras. Su uso está orientado en aplicaciones con un índice bajo de escrituras pero muchas lecturas.[11]

2.4.5. Software

Actualmente destacan los siguientes, todos son open-source.

- `MongoDB`: Creada por 10gen, está orientada a documentos, es posible persistir objetos, usa un formato propio para el almacenamiento de datos BSON[10]. Tiene soporte de índices y referencias entre documentos, lenguaje de consultas en otras.

- `Redis`: Implementa el paradigma key-value, para entender mejor este sistema se puede comparar como un arreglo gigante en memoria para

[9] Son una colección de objetos que no tienen orden y que están identificados por algún identificador único

[10] El nombre BSON está basado en el término JSON y significa Binary JSON (JSON Binario).

almacenar datos y esos datos pueden ser de diferentes tipos, strings, hashes, conjuntos y listas; con la ventaja de que sus operaciones son atómicas y persistentes. No tiene una forma de realizar consultas solo puede insertar y obtener datos, además de las operaciones comunes sobre conjuntos: diferencia, unión, intersección. Creado en ANSI C, compatible en sistemas Unix, Linux, Solaris, OS/X sin embargo no existe soporte oficial para plataformas Windows.

- `Casandra`: Es una base de datos orientada a columnas, en Cassandra una fila puede tener un conjunto diferente de columnas. Dispone de un lenguaje propio para realizar consultas CQL (Cassandra Query Language). Cassandra es una aplicación Java por lo que puede correr en cualquier plataforma que cuente con la JVM.

- `CouchDB`: Los datos son almacenados en columnas en lugar de filas. Con este cambio se gana velocidad en la lectura, ya que si se requiere consultar un número reducido de columnas, es muy rápido hacerlo. Este tipo de base de datos son deficientes a la hora de realizar escrituras. Su uso está orientado en aplicaciones con un índice bajo de escrituras pero muchas lecturas.Es una base de datos orientada a documentos, usa JavaScript como lenguaje de interacción principal, el intercambio de datos se realiza mediante JSON. Estas características la hacen extremadamente interoperable y prácticamente es posible acceder a ella desde cualquier ambiente. CouchDB está compuesta por colecciones de documentos donde estos pueden contener datos en formato JSON. Escrita en el lenguaje Erlang, especialmente desarrollado para construir sistemas concurrentes distribuidos. Funciona en la mayoría de los sistemas POSIX, incluyendo a Linux y OS/X; por otra parte, no hay soporte oficial para Windows.[7]

Capítulo 3

Diseño general

Un agente inteligente diseñó la vida.
Michael Behe

3.1. Descripción general

Esta sección abarca el planteamiento general de la arquitectura junto a todos sus capas y los módulos que incluye cada una de ellas. Cada capa será detallada y explicada en cuanto a su funcionamiento y la relación que tienen con todo el modelo.

3.2. Arquitectura del sistema

En la figura (figura 3.1) se encuentra el diagrama que esquematiza la arquitectura del sistema de recomendación contextual que proponemos.
De manera general, el sistema de recomendación se divide en las cuatro capas principales:

- Capa de Datos.
- Motor de recomendación.
- Capa de Servicios.
- Capa de Cliente.

En las siguientes secciones vamos a ir detallando cada una de las capas, para ver su iteración con todo el sistema de recomendación y los módulos que contiene cada una de las capas.

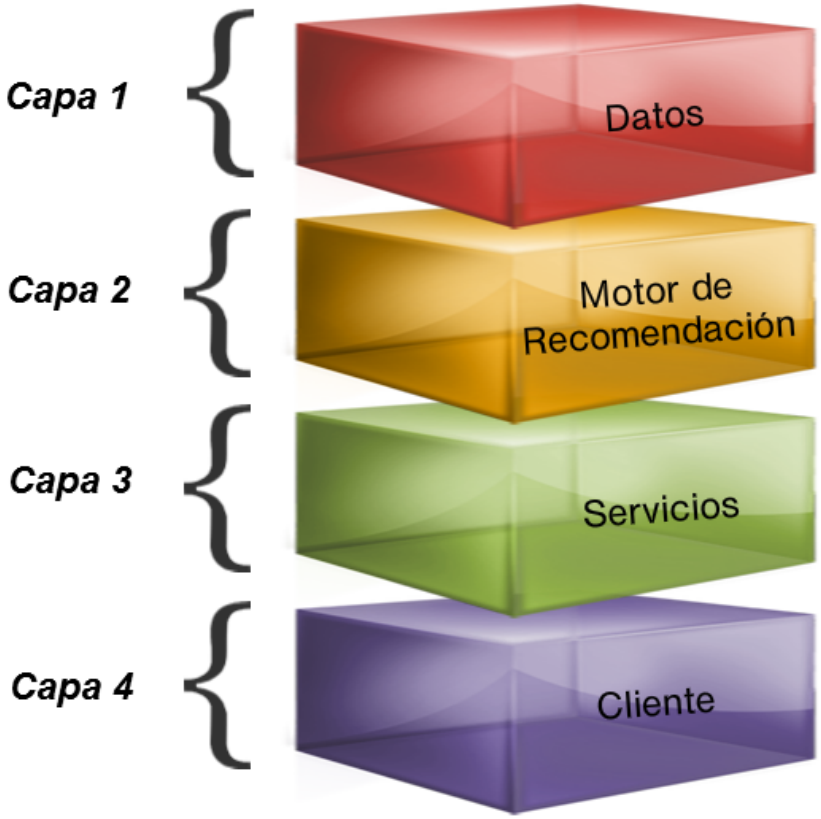

Figura 3.1: Arquitectura general del sistema

3.3. Capa de Datos

Esta capa tiene como objetivo principal del almacenamiento de todas las recomendaciones, los usuarios, ítems y la información contextualizada del todo el sistema de recomendación. El almacenamiento de tal información será realizada por un tipo de base de datos NoSql orientado a documentos.

La selección de dicha base de datos y el tipo de almacenamiento esta propuesta en este trabajo por tratarse de un modelo de almacenamiento que ayuda en el rendimiento general del sistema, como está expuesto en [22] donde se realizan comparaciones entre sistemas de gestión de base de datos tradicionales una base de datos NoSQL. Además de contar con un escalamiento en horizontal(figura 3.2) lo cual permite el manejo de grandes cantidades de datos afectando muy poco al rendimiento general del sistema.

Por todo esto dicha capa está pensada para que pueda crecer en horizon-

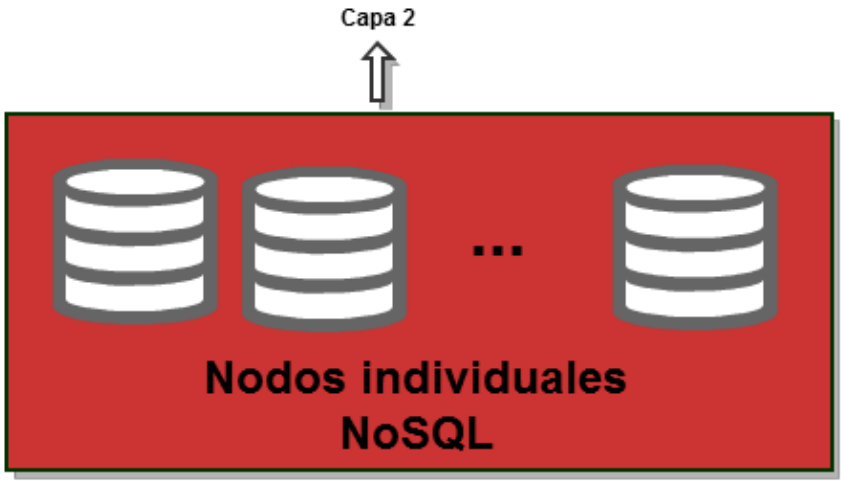

Figura 3.2: Escalabilidad de nodos en horizontal.

tal con varios nodos, cuando se pueda llegar a tener una gran cantidad de información de recomendación.

3.3.1. Almacenamiento de datos

El almacenamiento de toda la información del sistema estará dotado por un modelo orientado a colecciones de documentos, donde cada colección de una base de datos NoSql es el equivalente a una tabla o entidad en un modelo entidad-relación. Como se puede ver observar en la figura 3.3 existen tres tipos de documentos necesarios para el funcionamientos(Usuarios, ítems y rating) y uno opcional, este último además de ser opcional es dinámico en su construcción. Esto quiere decir que se pueden tener N colecciones de documentos de *Rating contextualizado*. La creación de estas N colecciones son una característica del gestor NoSql, ya que cada colección representa a un contexto en particular y se pueden ir creando dinámicamente varios contextos de manera automática en colecciones. Este proceso de creación automático será explicado a más detalle en la capa del *Motor de recomendación*, es una característica importante de los tipos de base de datos NoSQL.

3.4. Motor de Recomendación

El *Motor de recomendación* es una de las mas importantes de todo el sistema de recomendación, en esta capa se encuentran los algoritmos de recomendación, así como el tratamiento de las recomendaciones, las diferentes

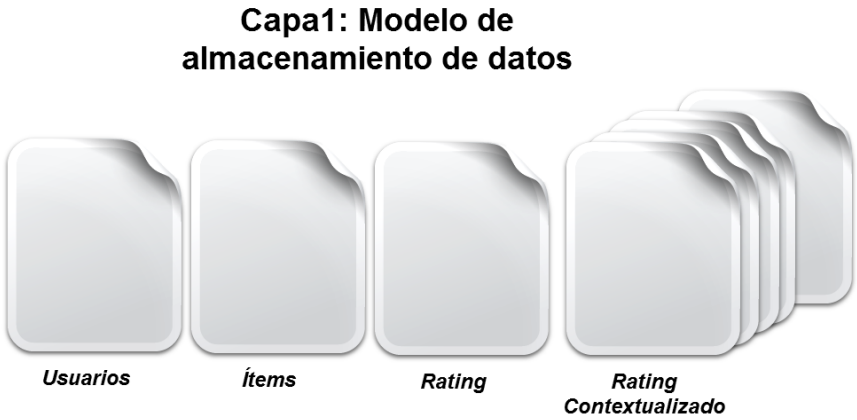

Figura 3.3: Almacenamiento de la información en documentos.

opciones que presenta el sistema y su comportamiento con cada una de ellas, la parametrización del sistema y las reglas del contexto.

La tarea principal de esta capa es la de manejar todas las interacciones que existe con la capa de datos y las capas superiores a esta. Es la de actuar como un controlador que realiza diferentes tareas en funciones de las entradas, como por ejemplo tareas de registro de rating, o tareas para la obtención de una recomendación para un usuario en particular.

Esta capa está compuesta por cinco principales módulos (figura 3.4):

- Algoritmos de recomendación.
- Motor de recomendación.
- Controlador.
- Reglas de contexto.
- Parametros

3.4.1. Algoritmos de recomendación

Este módulo tiene como objetivo principal dotar de los algoritmos de recomendación, aquellos que hasta ahora han sido estudiados en la literatura[25]. Para este sistema de recomendación en particular se emplearan aquellos que son de filtrado colaborativo.

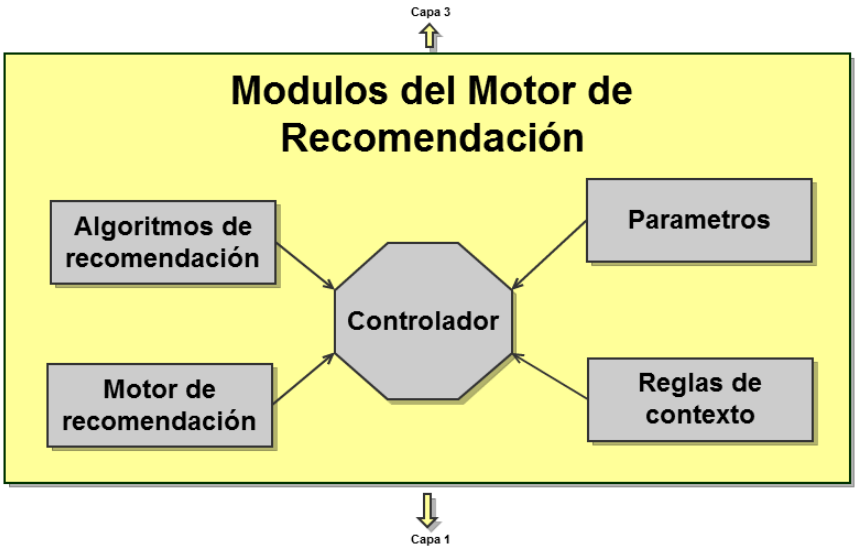

Figura 3.4: Módulos del Motor de Recomendación

3.4.2. Motor de recomendación

El motor de recomendación esta encargado de realizar el proceso de recomendación, dicho proceso está condicionado por otros módulos como los parámetros de entrada, la parametrización del sistema y la reglas del contexto. En función de estos tres parámetros importantes se puede realizar el proceso de recomendación. Como ejemplo se podría tener como entrada un usuario U, que requiere una recomendación para un determinado contexto C, y se tiene como parámetro el uso del algoritmo de recomendación basado en *Item-Item*[13].

3.4.3. Controlador

El controlador es el encargado de fusionar los demás módulos y poder llevar a cabo diferentes tareas. Estas tareas pueden ser la de generar una recomendación, almacenar un rating, verificar la existencia de un contexto, manejar el módulo de parámetros para actuar en función de los mismos. Por este controlador pasan todos los datos de capa de servicios y de la capa de datos.

3.4.4. Reglas del contexto

Este módulo tiene como objetivo principal de comprobar si un contexto C, es parte de una estructura, reglas para el tratamiento de contextos. Es decir cómo nos muestra en [27], [23], [19] y [6], el contexto puede ser manejado de diferentes formas, desde una simple entidad como por ejemplo *semana*, hasta entidades compuestas como por ejemplo: *fin de semana*, puede estar compuesto de otros contextos como *sábado y domingo*. Por todo esto la función de este módulo es de preguntar si el contexto actual corresponde a un contexto de nivel superior o no, dependiendo del tipo de modelo que maneje las reglas de contexto.

3.4.5. Parametros

El módulo de parámetros tiene como función principal de parametrizar características del todo el sistema de recomendación. Estos parámetros pueden ser: el tipo de algoritmo empleado para las recomendaciones, la cantidad de recomendaciones que retornara el mismo, normalización de los datos entre otros.

Este módulo está abierto a que se puedan ir añadiendo características importantes, para que el sistema llegue a ser más flexible a la hora de implementar el mismo en diferentes tipos de entornos con diferentes requerimientos.

3.5. Capa de Servicio

Esta capa ofrece servicios para que puedan ser consumidas por la capa de cliente, estos servicios están orientados a ofrecer recomendaciones y al almacenamiento de ratings, también incluye un servicio adicional para la importación de datos. Como se puede apreciar en la figura 3.5, esta capa consta de tres módulos importantes.

- Registro de ratings.
- Recomendador.
- Importación de datos.

3.5.1. Registro de ratings

Este módulo se encarga de recibir grupos de datos de la forma $U \times I \times R$ o de la forma $U \times I \times C \times R$, para ser almacenados en los documentos correspondientes, donde U es el usuario, I es el ítem, C es el contexto y R es el rating

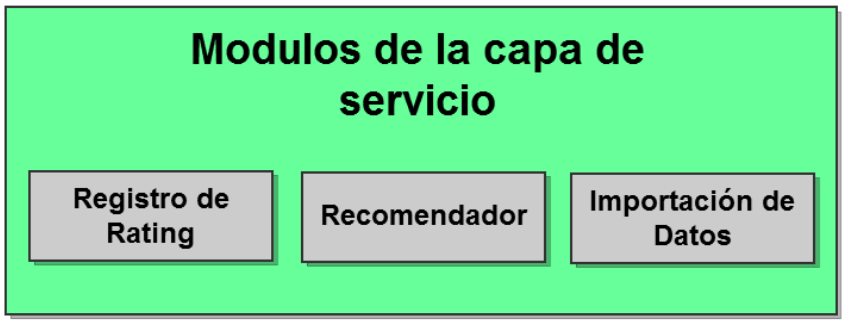

Figura 3.5: Modulos de la capa de servicio.

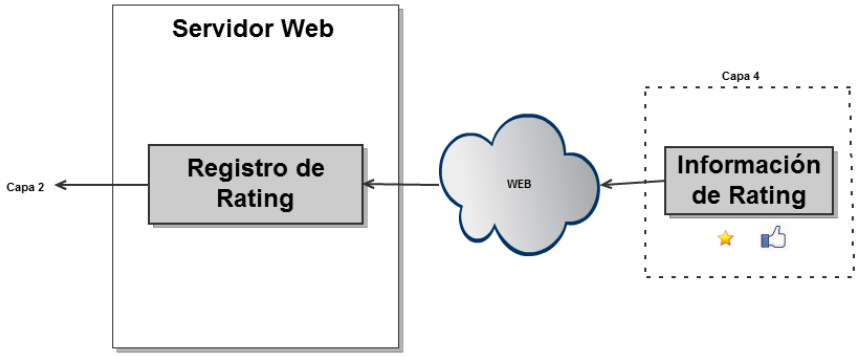

Figura 3.6: Registro de un rating a través del consumo del servicio.

o valoración. En la figura 3.6 se puede ver como ingresa la información del rating desde la capa 4 pasando por toda la nube de internet y llegando al servicio.

3.5.2. Recomendador

El servicio es responsable de ofrecer las listas de recomendación, dichas listas son el producto de la entrada de datos de la forma de $U \times I$ o $U \times I \times C$, donde U es el usuario, I el ítem y C el contexto. Se puede observar en 3.7, como la información llega por la nube al servicio, el cual luego de hacer todo el proceso de recomendación devuelve un lista top N, de ítems para el usuario activo.

3.5.3. Importación de datos

Este módulo fue incorporado con el propósito importante de que el sistema de recomendación sea adaptable a cualquier sistema de comercio elec-

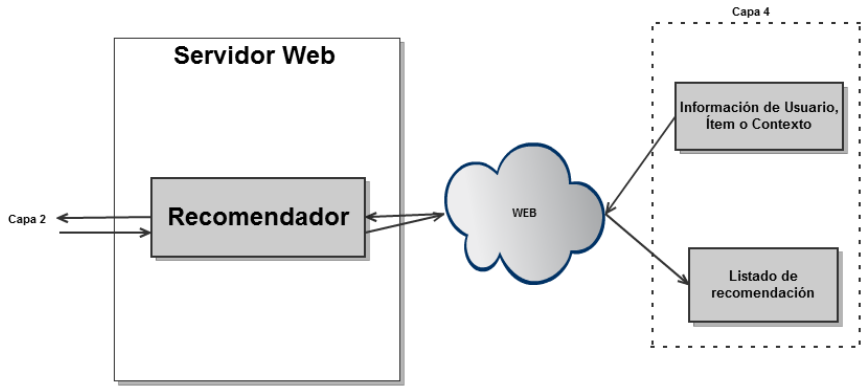

Figura 3.7: Obtención de recomendaciones a través del consumo del servicio.

trónico.
El propósito es obtener de un sistema existente, que tiene tablas de usuarios e ítems, se puedan llegar a importar toda esa información a los documentos de la base de datos NoSQL, para que luego puedan ser usados en las recomendaciones del sistema.

3.6. Capa de Cliente

Esta capa tiene como su objetivo mas importante de integrar el sistema de recomendación, en cualquier interface o formulario o página web de un sistema externo para que pueda interactuar con el sistema de recomendación. En la figura 3.8, se puede ver los dos principales módulos, *API Web y API dispositivos móviles*. El primero se trata de un conjunto de procedimientos y funciones que ayuden a mostrar de manera dinámica en cualquier entorno web las recomendaciones y también ayuden a que se puedan consumir los servicios de la capa 3. El segundo, orientado a dispositivos móviles, tiene el mismo propósito que el primero, pero con la característica de actuar autónomamente para la detección de contexto en dispositivos móviles como ejemplo podemos ver en [32].

Figura 3.8: Módulos de la capa de cliente.

Capítulo 4

Implementación del diseño del sistema

Soy pintor y clavo mis cuadros.

Kurt Schwitters

4.1. Descripción general

En parte vamos a llegar convertir la arquitectura propuesta en el capítulo anterior en un sistema de recomendación sensible al contexto, aplicando la tecnología necesaria para su funcionamiento. Vamos a llegar a ver y describir las tecnologías utilizadas en la implementación y el funcionamiento en las diferentes capas de todo el sistema de recomendación contextual.

Para poder llegar a desarrollar el sistema de recomendación se han utilizado una serie de lenguajes de programación, entre ellos se encuentran, JAVA, JAVA SCRIPT en particular JQUERY. También se incluyen tecnologías para el intercambio de información como JSON y el lenguaje de consultas propio de MongoDB la base de datos empleada para el sistema de recomendación.

4.2. Modelo de Arquitectura general del sistema

La figura 4.1, presenta un modelo similar al presentado 3.1. En este caso se detallan cada una de las tecnologías implementadas en cada una de las capas y los módulos de los mismos. De esta manera vamos a ir presentando cada una de las capas y la implementación desarrollada.

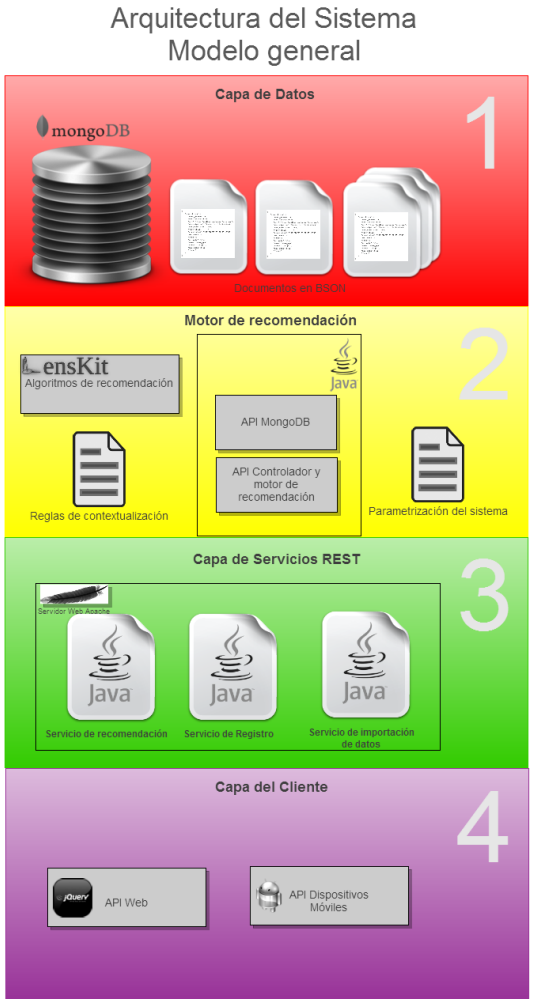

Figura 4.1: Modelo de Arquitectura general del sistema

4.3. Capa de Datos

En esta parte se ha incorporado un sistema de base de datos NoSQL llamado MongoDB, el mismo incluye varias características anteriormente dichas sobre este tipo de bases de datos, incluyendo su tipo de almacenamiento de datos basado en documentos.

La implementación de la base de datos fue hecha bajo un el entorno del sistema operativo Windows 7, ya que MongoDB ofrece distribuciones para diferentes plataformas incluyendo Linux, Solaris y OS X.

El modelo de la base de datos en Mongo DB está presentando por colec-

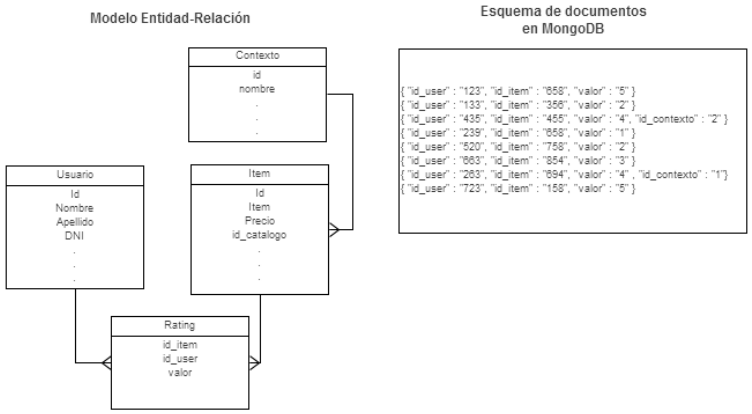

Figura 4.2: Comparación modelo entidad relación y una colección de documentos.

ciones de documentos como se puede observar en el ejemplo de la figura 4.1 en la parte 2, que a diferencia de un modelo clásico entidad relación se hace uso de las tablas para entidades y además el empleo de relaciones entre las mismas entidades para hacer una normalización de datos.

Una de las principales diferencias entre ambos modelos se centra en la obtención de la información. Con el uso de tablas, es necesaria la realización de un producto cartesiano entre tablas para poder obtener información. En cambio con una base de datos NoSql se tiene toda la información en una sola colección, no hay relaciones.

Para el presente proyecto dichas colecciones de documentos pueden llegar a creer en función de la cantidad de contextos que se vayan agregando al sistema. Dicho de otra manera una colección en MongoDB representara información de rating para un contexto en particular. Visto desde los paradigmas estaríamos hablando de un pre-filtrado contextual.

En el caso particular de un modelo entidad relación, para poder obtener información de los ratings, generalmente (aunque no necesariamente) se realiza una consulta a la base de datos, haciendo productos cartesianos entre tablas llegando así al resultado buscado. Es decir, si queremos los ratings de los ítems de un contexto en particular, debemos de relacionar los contextos con los ítems, posteriormente los usuarios, los ítems con los ratings.

Hasta este punto como se puede observar, ya se han realizado varias operaciones dentro el servidor de bases de datos que se pueden interpretar como tiempo de procesamiento. Ahora, mirando el lado derecho de la figura 4.2

se encuentran un esquema en MongoDB donde cada línea es un documento que contiene los ratings de usuarios e ítems, además se puede observar que hay documentos que tiene códigos de contextos. De esta manera, si se quiere llegar a obtener el mismo resultado con el esquema en MongoDB el tiempo de respuesta tiende a ser un poco más eficiente, ya que no necesita hacer varias operaciones dentro del servidor. Solo es necesario consultar una sola colección y no tiene que hacer relaciones con otras colecciones. Además se puede apreciar que la información del usuario como el nombre, apellido, etc. no se encuentra dentro de los documentos, que aunque pueden estar incluidos y repetirse en cada documento, no se encuentran, **porque no se necesitan** ya que no son relevantes para el sistema, el sistema solo se limita a obtener recomendaciones de usuarios e ítems con algún filtrado en particular, pero los demás datos como el nombre, o nombre del ítems, no son necesarios en absoluto para el sistema.

Con el ejemplo anterior, podemos afirmar que cuando queremos hacer una recomendación de filtrado colaborativo, vamos a tomar la información de una sola colección, o el filtrado de esa colección. Esto nos lleva tomar otra de las ventajas de MongoDB sobre la creación dinámica de colecciones[1], que podemos crear colecciones en tiempo de ejecución.

Ahora, suponiendo que dentro de la capa 2 específicamente en las reglas de contexto, se ha añadido un nuevo contexto que tiene por nombre *Posición*, este contexto dentro de la base de datos será creada como una colección independiente, es decir si vamos a registrar una valoración de un usuario e ítem en un determinado contexto, esta información contextual será almacenada en una colección independiente, con el nombre del contexto como nombre de la colección.

Es por eso que desde un principio se ha ido hablando de la contextualización de la información dentro de este sistema de recomendación. Por lo que esta información contextualizada quedaría en colecciones como se muestra en la figura 4.3, ahora en el momento de hacer una recomendación contextual, tomaría una sola colección y no así una colección que podría tener miles de documentos o tomar varias colecciones con muchos documentos. Esto supone una ventaja a la hora de hacer un pre-filtrado de contextos para hacer una recomendación.

[1] http://www.mongodb.org/display/DOCS/Tutorial#Tutorial-InsertingDataintoACollection

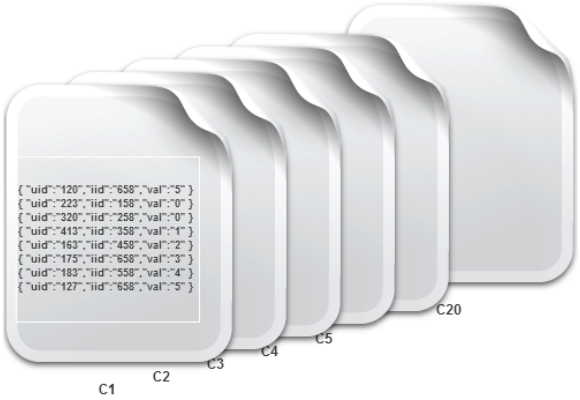

Figura 4.3: Información contextualizada en colecciones de documentos en MongoDB

4.3.1. Datos de prueba

Para realizar pruebas, y simular datos reales se ha empleado en conjunto de datos llamado *Book-Crossing Dataset*[2]. Este conjunto de datos está compuesto por tres tablas importantes, los usuarios, los libros (ítems) y la tabla de ratings. Esta última es nuestra tabla de interés que contiene 1,149,780 registros de rating.

Para poder tener los datos en funcionamiento dentro del sistema de recomendación contextual se ha tenido que hacer una migración, dicha migración ha incluido una transformación de ciertos datos a código entendible por el interpretador de MongoDB.

Las principales transformaciones de datos para la migración fueron en cuanto a tipo de datos que contiene la tabla de libros, ya que el campo principal del dataset de libros contiene un código identificador único de tipo de caracteres que luego fue llevado a conversión a un tipo numérico. Estos datos una vez transformados fueron importados dentro de la base de datos de MongoDB para que luego puedan ser utilizados por el sistema. La importación de datos se la realizo a través del comando propio de MongoDB llamado *mongoimport*.

[2]http://www.informatik.uni-freiburg.de/ cziegler/BX/

4.4. Motor de Recomendación

Para el *Motor de recomendación* se ha implementado el framework gratuito de recomendación llamado Lenskit. De este hemos tomado los principales algoritmos de filtrado colaborativo que vienen a ser los correspondientes al módulo de algoritmos de recomendación presentados en la figura 3.4. También se ha incorporado la API de MongoDB[3] para realizar la conexión de los datos con los algoritmos de recomendación.

El resultado de este motor de recomendación es un archivo compilado de tipo JAR[4] que puede ser utilizado en cualquier plataforma que tenga la máquina virtual de Java (JVM)[5].

Este archivo compilado será utilizado posteriormente para la realización de los servicios web, dichos servicios web emplearan los métodos principales para la obtención de recomendaciones y para poder grabar información de rating.

Dado que se está haciendo uso de Lenkit para el motor de recomendación hacemos uso de las principales clases para la recomendación que se detallan a continuación:

- **Rating:** clase base para el manejo de información de rating, contiene el ítem, usuario y valor.

- **itemRecomender:** Es una de las interfaces base que provee de los principales métodos para dar recomendaciones. Su principal idea es la de recomendar ítems a un usuario, donde los ítems recomendados son tomados des un conjunto de ítems de otros usuarios.

- **Recommender:** Interfase que se emplea para el uso de los principales métodos de recomendación (getItemRecommender() y getItemScorer())

[3] http://www.mongodb.org/display/DOCS/Java+Tutorial

[4] Un archivo JAR (por sus siglas en inglés, Java ARchive) es un tipo de archivo que permite ejecutar aplicaciones escritas en el lenguaje Java.

[5] Una máquina virtual Java (en inglés Java Virtual Machine, JVM) es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el bytecode Java), el cual es generado por el compilador del lenguaje Java

4.5. Capa de Servicio

4.5.1. Características principales

Esta capa viene a ser relacionada con el servidor web, esta implementada en Apache Tomcat [6] bajo el sistema operativo Windows. Para que los servicios presentados en la figura 3.1 puedan tener funcionalidad se ha utilizado el lenguaje de programación Java[7] y la tecnología REST[8] para la creación de los servicios web..

Para la creación de los servicios web se ha utilizado una tecnología de tipo arquitectura para sistemas distribuidos que nos muestra unos principios en los cuales se pueden diseñar servicios web que funcionan a través HTTP [9]. La implementación de servicios web de tipo REST sigue cuatro principios básicos de diseño:

- Usa los métodos HTTP explícitamente.
- No tiene estado.
- Expone las URLs en un formato de directorios.
- Utiliza XML, JSON o ambos.

4.5.2. Modelo

El modelo utilizado para la creación de los servicios web se encuentra la figura 4.4, donde se ve una representación de diagramas de clases y sus relaciones.

En el modelo de la figura 4.4 se puede observar que la clase Rating y SaveRatingResponse son las clases padre para el uso de las otras dos clases. La clase RatingDAO es la que contiene los principales métodos para obtener datos del gestor de base datos y llevarlos al tipo Rating. Finalmente la clase RatingResource es la clase que contiene cada uno de los métodos del servicio web, contiene los métodos para entregar información de rating y métodos para registrar información de rating.

[6] El servidor HTTP Apache Tomcat 7.0 que es un servidor web HTTP de código abierto, multiplataforma

[7] Es un lenguaje de programación orientado a objetos

[8] Representational State Transfer, es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. Un servicio web REST principalmente sigue los siguientes cuatro principios: Utiliza los métodos HTTP de manera explícita, no mantiene estado, expone URIs con forma de directorios y transfiere XML, JavaScript Object Notation (JSON), o ambos.

[9] Hypertext Transfer Protocol , es un protocolo de transferencia de hipertexto que es usado en cada transacción de la world wide web

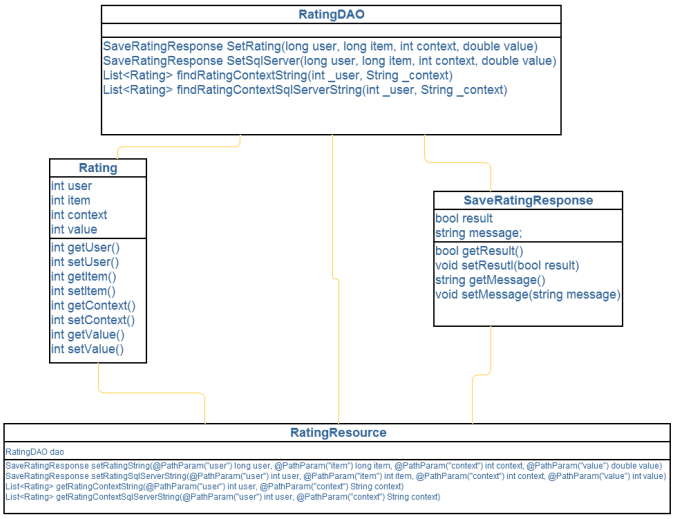

Figura 4.4: Modelo de clases Servicios Web

4.5.3. Consumición de servicios

El módulo de recomendación, aquí es representado como un servicio web que puede ser consumido mediante un URL[10], a dicha URL se le asocian una serie de parámetros siguiendo el principio de REST de exponer las URLs en forma de directorios.

Para la implementación de este proyecto se presentan las siguientes estructuras de URL para los servicios WEB:

Para el servicio web que va contra la base de datos MongoDB:
http://localhost:8080/WsBookRecommenderRest/rest/rating/mongoContext

Para el servicio web que va contra la base de datos MongoDB:
http://localhost:8080/WsBookRecommenderRest/rest/rating/sqlserverContext

El paso de parámetros esta dado de la siguiente forma:.
Url: $url + / + user + / + context$,.

Dónde:.
url: Es la URL del servicio web ya sea por MongoDB o por MsSqlServer.
user: Es el código de usuario.
context: Es el código de contexto.

[10]El URL es la cadena de caracteres con la cual se asigna una dirección única a cada uno de los recursos de información disponibles en la Internet.

El resultado de dicha consumición del servicio es una lista de recomendaciones con un formato JSON o XML que puede ser utilizado de cualquier forma por parte del cliente para su uso.

Para el almacenamiento de información contextual, también se tiene un par de URLs por tipo de gestor de base de datos. Estos servicios web encargados del registro de información contextual retornan un XML o JSON de un valor booleano para la comprobación de éxito o fracaso del mismo. El servicio de importación viene a ser una aplicación en JAVA, que tiene como objetivo la de hacer conexión con el sistema externo, ya sea de comercio electrónico o algún otro sistema que contenga información de usuarios e ítems. La idea principal de este servicio es tener una especie de Wizard[11] o asistente de importación de datos de rating. Este servicio se encuentra incompleto y falto de desarrollo en este proyecto, ya que no llega a ser relevante para las pruebas finales.

4.6. Capa de Cliente

Esta capa fue implementada en dos parte importantes, un API desarrollado gracias a JQuery para su incorporación en cualquier aplicación web ya sea en entorno de escritorio como en un entorno de dispositivos móviles.

Gracias a que JQuery es una biblioteca de java script, esta nos permite usar tecnología como AJAX[12] para que se puedan consumir los servicios anteriormente explicados de la capa anterior. Un ejemplo de incorporación de este tipo módulo son los utilizaos por Google Analytics[13], que ofrecen un script para que pueda ser insertado en una aplicación web para que este pueda ser rastreado[14]. De esa misma manera incorporamos un script dentro de la de la aplicación web para los siguientes usos:

- Consumir el servicio de Rating, script que será colocado principalmente para valorar los ítems.

[11] Un asistente de software o el asistente de configuración es un tipo de interfaz de usuario que presenta a un usuario con una secuencia de cuadros de diálogo que llevan al usuario a través de una serie de pasos bien definidos. Un ejemplo es la instalación de un programa en particular, donde en dicha instalación al usuario se le es guiado por una serie de pasos hasta culminar la instalación.

[12] Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications).

[13] .es un servicio gratuito de estadísticas de sitios web.

[14] https://developers.google.com/analytics/devguides/collection/gajs/gaTrackingOverview

- Para consumir el servicio de recomendación, y de esta forma obtener la lista de recomendaciones, el resultado como se explicó anteriormente esta en JSON y gracias a JQUERY se puede mostrar de cualquier forma que el cliente disponga.

Para el consumo de cada uno de los servicios web a través de JQuery se ha hecho uso del método Ajax(). La forma de utilización del mismo esta implementada y de manera general está representada de la siguiente forma:

```
$.ajax({
        type: 'get',
        url: _url + "/" + _user + "/" + _context,
        timeout: 30000,
        contentType: "application/json; charset=utf-8",
        dataType: 'json',
        success: function (data, textStatus, jqXHR) {
         //Trabajar con la información retornada por el servicio
        }
      });
```

Donde en el propiedad *url* se coloca la dirección URL del servicio web, *dataType* defina el tipo de dato que retorna el servicio web y que luego se utilizara para trabajar en la propiedad *success*.

Capítulo 5

Experimentos realizados

Acá hay tres clases de gente: las que se matan trabajando, las que deberían trabajar y las que tendrían que matarse.

Mario Benedetti

5.1. Descripción general

El objetivo de los experimentos de este capítulo es mostrar si la arquitectura presentada en este presente trabajo con bases de datos NoSql es o no más eficiente que una base de datos relacional.

Se mostraran los experimentos realizados con el sistema de recomendación contextual. Dichos experimentos serán solicitudes de recomendación por medio de un prototipo el cual consume los servicios web del sistema de recomendación haciendo uso de dos gestores de bases de datos, en este caso MongoDB con tres colecciones de ejemplo que serán detalladas más adelante y MsSqlServer con su modelo entidad relación.

5.2. Dominio de los experimentos

Para poder realizar una experimentación y comparación entre dos gestores de bases de datos se ha seleccionado el dataset presentado en el capítulo 5.4. Este dataset ha tenido un paso previo de importación, tanto para el gestor de MongoDB como para el gestor de MsSqlServer. La cantidad total de registros son de 612057 (figura 5.1), la división para cada gestor de base de datos queda distribuido de la siguiente manera:

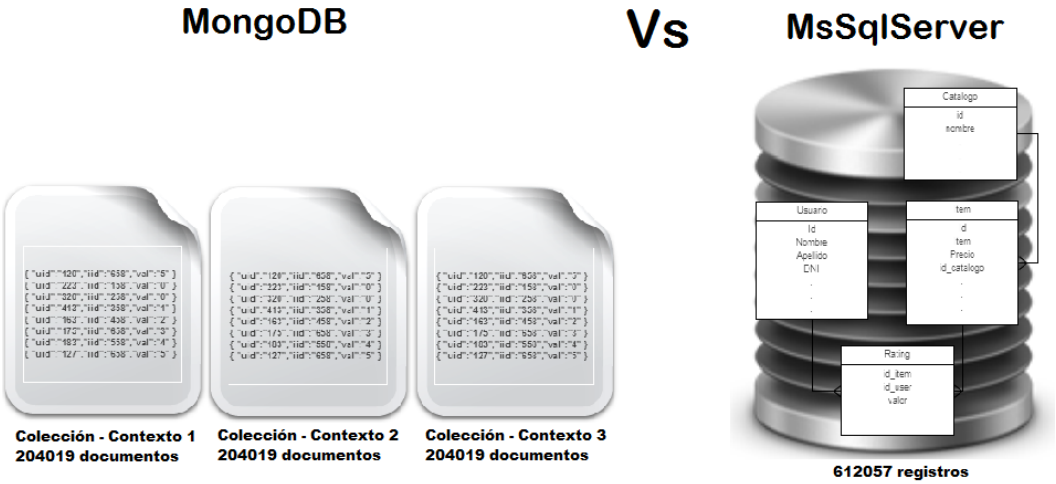

Figura 5.1: Dominio de datos

MongoDB:

- Colección 1 ->Contexto 1 [Posición] : 204019 documentos
- Colección 2 ->Contexto 2 [Entre Semana] : 204019 documentos
- Colección 3 ->Contexto 3 [Fin de Semana] : 204019 documentos

MsSqlServer:

- Tabla ->Tabla de rating único con un campo IdContexto: 612057 registros.

5.3. Entorno de pruebas

Para poder realizar dichos experimentos se ha montado un entorno de trabajo que funciona en un sistema operativo Windows 7. Cada una de las partes del conjunto será explicado a continuación en los siguientes puntos:

5.3.1. Composición general

Como se puede observar en la figura 5.2, se tiene una representación del entorno de experimentación. En primer lugar se encuentra el prototipo, lugar donde se podrá ver en tiempo de ejecución los dos experimentos que se

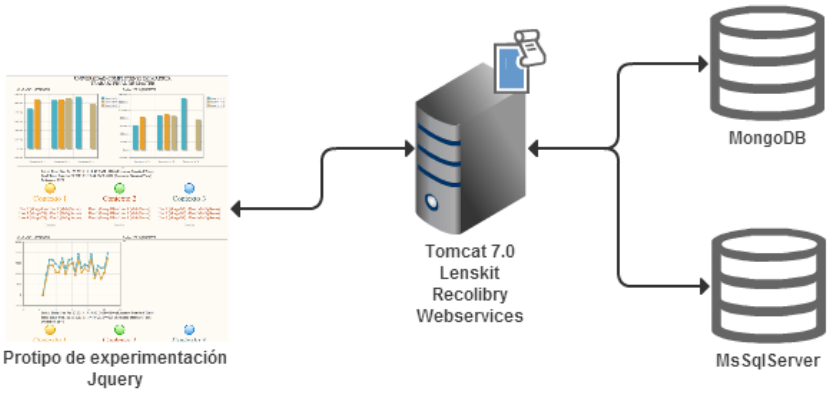

Figura 5.2: Estructura general del experimento

tendrán para este trabajo.

El prototipo para este trabajo está desarrollado en HTML y JQuery para la consumición de los servicios web del sistema recomendador. En la parte central de la figura 5.2 tenemos el servidor web, que utiliza como ente principal Tomcat 7.0, Lenskit como motor de recomendación, Recolibry que contiene las principales funciones del recomendador y finalmente se encuentran los servicios Web REST. En la capa de datos se encuentran los dos motores de base de datos, cada uno de ellas se comunica con el servidor web por medio de APIS en Java.

5.3.2. Características técnicas del entorno de experimentación

Entrando más a detalle a cada uno de los componentes del entorno de experimentación tenemos:

- Sistema operativo: Windows 7 - 64
- Servidor Web: Apache Tomcat 7.0
- Motor de Recomendación: Lenskit.
- Motores de Base de datos: MongoDB y MsSqlserver 2005.
- Procesador: Intel Core i7-2600 3.4 GHz
- Memoria RAM: 6 GB.

5.4. Proceso de experimentación

El proceso de experimentación sigue los siguientes pasos:

- El sistema de recomendación contextual parte de que tiene tres contextos
- Se selecciona 3 usuarios, dichos usuarios son seleccionados partiendo de que tengan documentos en cada una de las colecciones de los contextos.
- Para comenzar una prueba del experimento se selecciona primero un contexto.
- Para cada usuario del contexto seleccionado se hacen llamadas a los servicios web del sistema de recomendación. Una llamada para MongoDB y otra para MsSqlServer todo esto por medio de Ajax.
- Se registran los tiempos de cada una de las llamadas anteriores para luego poder evaluarlas y mostrarlas en las gráficas de comparación (graficas de barras) o en las gráficas evolutivas.

5.5. Experimento realizado

El experimento realizado en este punto sigue el proceso explicado en la sección anterior. El objetivo es determinar los tiempos que se tarda en solicitar una recomendación por usuario para cada tipo de gestor de base de datos y poder ver gráficamente tanto por barras como evolutivamente la diferencia en milisegundos del tiempo. Las características principales de este experimento son las siguientes:

5.5.1. Información seleccionada

Los datos seleccionados para este experimento vienen a ser los siguientes:

- Tres usuarios, cada usuario fue seleccionado con la condición de que se encuentre en los tres contextos presentados en este experimento. Los códigos de los usuarios seleccionados son: 162639, 210485, 161752.
- Tres contextos, cada contexto viene a representar tres contextos reales:
  - Contexto 1: Posición
  - Contexto 2: Fin de semana (Sábado, Domingo)
  - Contexto 3: Entre semana (De lunes a viernes).

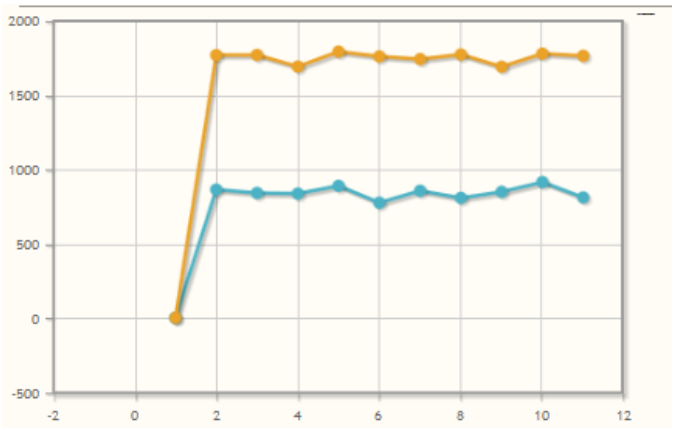

Figura 5.3: Grafica Usuario 1 en Contexto 1

5.5.2. Evaluación

Para la evaluación de los experimentos se realiza un total de 10 iteraciones para un usuario por contexto, haciendo un total de 60 solicitudes de recomendación contextual para un usuario, esto porque cada iteración hace 2 llamadas en paralelo, una para MongoDB y la otra para MsSqlServer.

De cada iteración, se va registrando los tiempos en milisegundos de la solicitud de recomendación, se marca el inicio desde el inicio de la llamada al servicio web por medio de Ajax, y se finaliza cuando se recibe las recomendaciones.

5.5.3. Resultado final del experimento

Como se puede observar los resultados evolutivos de cada usuario, se puede llegar a diferenciar visualmente en las gráficas 5.3, 5.5, 5.7 para el Usuario 1; 5.9, 5.11 para el Usuario 2; 5.13, 5.15 para el Usuario 3. Donde en el eje de horizontal se tiene el número de iteraciones por usuario y en el vertical se tiene los tiempos en milisegundos. Los puntos y líneas de color naranja son los tiempos de solicitud de recomendación utilizando el servidor MsSqlServer, mientras el color azul es para representar los tiempos de la base de datos MongoDB.

Las tablas 5.4, 5.6, 5.8 para el Usuario 1; 5.10, 5.12 para el Usuario 2; 5.14, 5.14 para el Usuario 3. Como se explicó, para la evaluación se contendrá un número de diez iteraciones. En la primera columna de cada tabla se tiene el número de iteración, en la segunda columna son los tiempos para MongoDB, la tercera columna es para MsSqlServer. Al final de la columna

Usuario: 1	Contexto: 1		
Tiempo de respuesta (milisegundos)			
Iteración	MongoDB	MsSqlServer	Diferencia
1	860	1762	902
2	837	1763	926
3	833	1685	852
4	885	1785	900
5	772	1753	981
6	852	1735	883
7	804	1766	962
8	845	1684	839
9	910	1772	862
10	807	1757	950
Promedio	840.5	1746.2	905.7

Figura 5.4: Tabla de tiempos Usuario 1 en Contexto 1, MongoDB vs MsSqlServer

se tiene la diferencia en milisegundos de cada gestor de base de datos, y en la última fila de cada tabla se tiene el promedio de las 10 iteraciones para cada gestor con su diferencia.

Los resultados para las gráficas de barras se puede apreciar en las figura 5.17. En este último caso las figuras muestran una comparativa que por el lado izquierdo de las gráficas tenemos a MongoDB que utiliza el modelo de Contextualización por colección y por el lado izquierdo tenemos a MsSqlServer con un modelo entidad relación clásico. Cabe recalcar que el Usuario 3 no existe en contexto 1, y el Usuario 2 no existe en el contexto 3.

5.5.4. Conclusión del experimento

Analizando las tablas 5.4, 5.6, 5.8 para el Usuario 1; 5.10, 5.12 para el Usuario 2; 5.14, 5.14 para el Usuario 3, se puede llegar a observar los promedios de tiempos para cada usuario en cada contexto y se puede ver una diferencia sustancial de casi 1000 milisegundos en cada uno de los 3 casos, llevando la delantera en un mejor tiempo de respuesta utilizando un motor de base de datos NoSql por parte del sistema de recomendación.

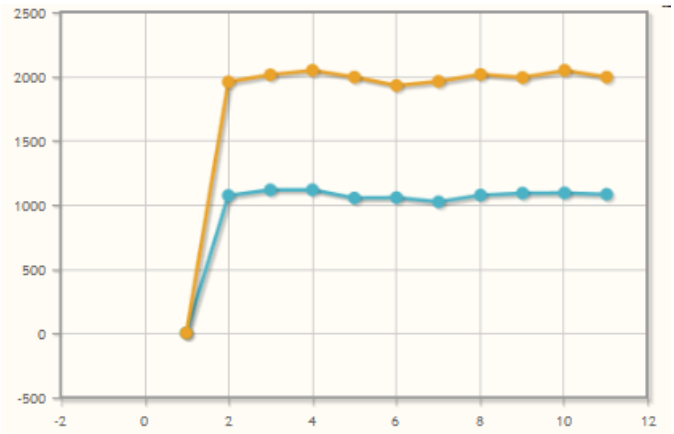

Figura 5.5: Grafica Usuario 1 en Contexto 2

Usuario: 1 **Contexto: 2**

Tiempo de respuesta (milisegundos)

Iteración	MongoDB	MsSqlServer	Diferencia
1	1071	1956	885
2	1115	2012	897
3	1116	2046	930
4	1052	1994	942
5	1055	1928	873
6	1022	1960	938
7	1074	2015	941
8	1089	1991	902
9	1091	2046	955
10	1080	1994	914
Promedio	1076.5	1994.2	917.7

Figura 5.6: Tabla de tiempos Usuario 1 en Contexto 2, MongoDB vs MsSqlServer

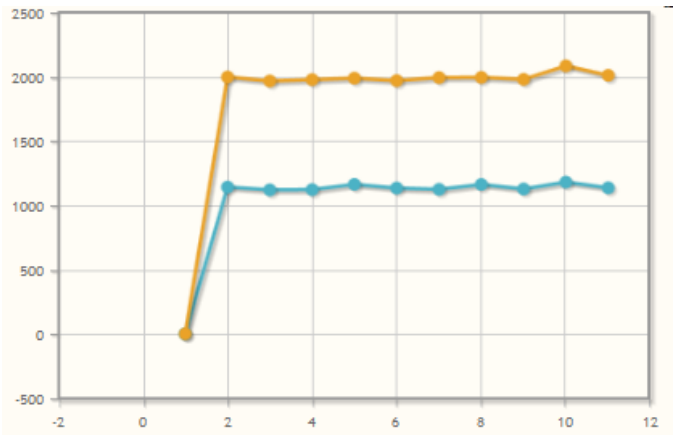

Figura 5.7: Grafica Usuario 1 en Contexto 3

Usuario: 1 **Contexto: 3**

Tiempo de respuesta (milisegundos)

Iteración	MongoDB	MsSqlServer	Diferencia
1	1142	1998	856
2	1120	1967	847
3	1122	1978	856
4	1163	1989	826
5	1134	1970	836
6	1124	1994	870
7	1162	1995	833
8	1127	1980	853
9	1180	2085	905
10	1136	2011	875
Promedio	1141	1996.7	855.7

Figura 5.8: Tabla de tiempos Usuario 1 en Contexto 3, MongoDB vs MsSqlServer

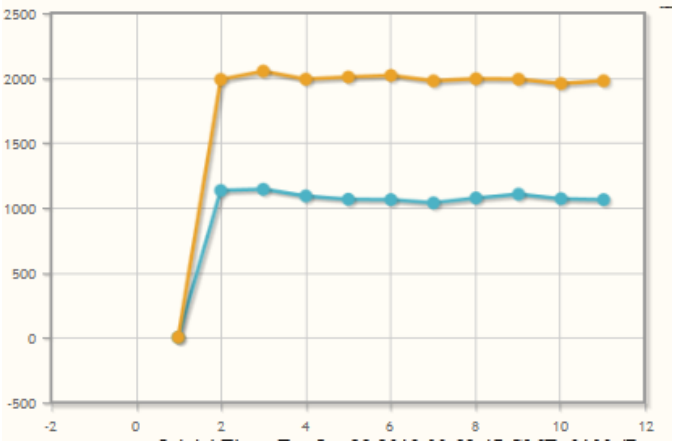

Figura 5.9: Grafica Usuario 2 en Contexto 1

Usuario: 2 **Contexto: 1**

Tiempo de respuesta (milisegundos)

Iteración	MongoDB	MsSqlServer	Diferencia
1	1132	1988	856
2	1142	2052	910
3	1089	1990	901
4	1064	2007	943
5	1061	2019	958
6	1037	1977	940
7	1074	1994	920
8	1103	1990	887
9	1068	1956	888
10	1061	1978	917
Promedio	1083.1	1995.1	912

Figura 5.10: Tabla de tiempos Usuario 2 en Contexto 1, MongoDB vs MsSqlServer

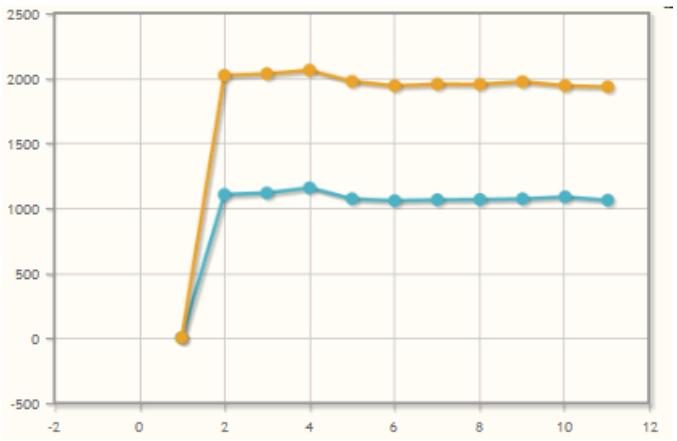

Figura 5.11: Grafica Usuario 2 en Contexto 2

Usuario: 2 **Contexto: 2**

Tiempo de respuesta (milisegundos)

Iteración	MongoDB	MsSqlServer	Diferencia
1	1103	2022	919
2	1115	2034	919
3	1155	2061	906
4	1070	1974	904
5	1056	1941	885
6	1063	1953	890
7	1065	1952	887
8	1070	1972	902
9	1085	1942	857
10	1060	1933	873
Promedio	1084.2	1978.4	894.2

Figura 5.12: Tabla de tiempos Usuario 2 en Contexto 2, MongoDB vs MsSqlServer

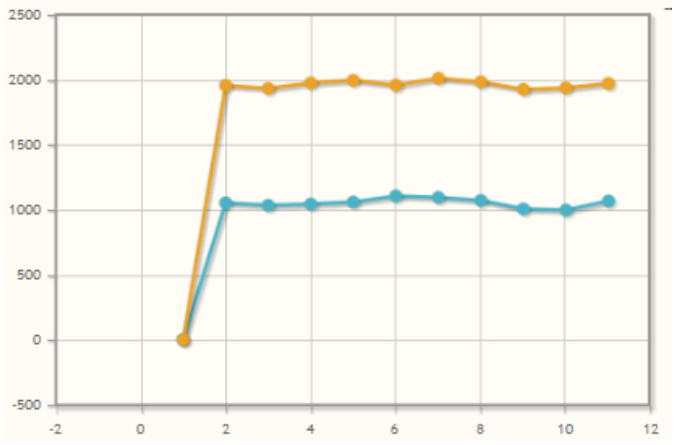

Figura 5.13: Grafica Usuario 3 en Contexto 2

Usuario: 3 **Contexto: 2**

Tiempo de respuesta (milisegundos)

Iteración	MongoDB	MsSqlServer	Diferencia
1	1053	1956	903
2	1034	1933	899
3	1044	1975	931
4	1059	1996	937
5	1105	1958	853
6	1094	2011	917
7	1071	1983	912
8	1006	1925	919
9	997	1937	940
10	1067	1971	904
Promedio	1053	1964.5	911.5

Figura 5.14: Tabla de tiempos Usuario 3 en Contexto 2, MongoDB vs MsSqlServer

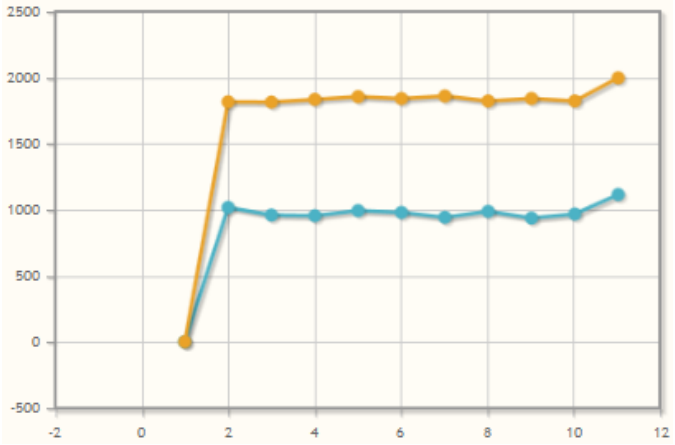

Figura 5.15: Grafica Usuario 3 en Contexto 3

Usuario: 3 **Contexto: 3**

Tiempo de respuesta (milisegundos)

Iteración	MongoDB	MsSqlServer	Diferencia
1	1017	1817	800
2	959	1815	856
3	954	1835	881
4	993	1856	863
5	979	1841	862
6	942	1861	919
7	987	1823	836
8	937	1842	905
9	967	1823	856
10	1115	1999	884
Promedio	985	1851.2	866.2

Figura 5.16: Tabla de tiempos Usuario 3 en Contexto 3, MongoDB vs MsSqlServer

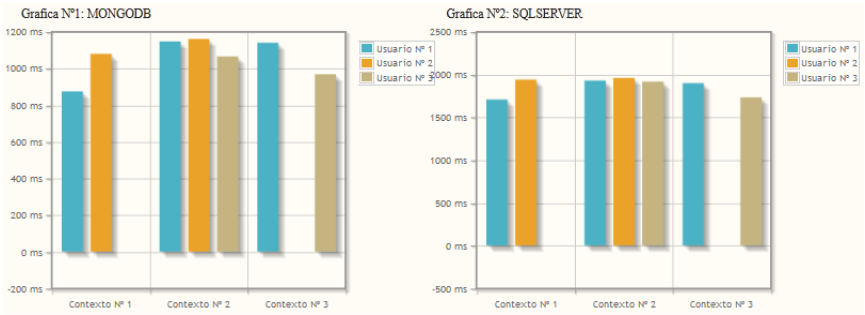

Figura 5.17: Resultado iteración MongoDB vs MsSqlServer

Capítulo 6

Conclusiones

Saltar rápidamente a conclusiones rara vez conduce a felices aterrizajes.

S. Siporin

6.1. Descripción general

En este capítulo vamos a presentar las conclusiones relevantes alcanzadas como consecuencia de este presente trabajo. Además del trabajo se proponen una seria de líneas de mejora y de investigación para futuros trabajos.

6.2. Conclusiones

En relación a los objetivos planteados en el capítulo 1, podemos ver el grado de cumplimiento en los siguientes puntos:

- Para poder culminar este trabajo, se realizó un estudio sobre los sistemas de recomendación y sistemas de recomendación sensibles al contexto, con el objetivo de tener un mejor entendimiento sobre el funcionamiento de cada uno de estos temas y así poder aplicarlos sobre el conjunto de la arquitectura propuesta.
- El estudio sobre las bases de datos NoSQL en el estado del arte también ha permitido incorporar la idea de manejar información contextual en forma de documentos para su procesamiento. Del mismo modo se ha llegado a seleccionar MongoDB como base de datos para implementación, y así cumplir con uno de los objetivos de la arquitectura y del sistema en si, que abarca temas de escalabilidad y el rendimiento.
- Hemos incorporado en la implementación del sistema de recomendación contextual software OpenSource para su construcción y desarrollo. De

esta manera mantenemos la idea de que el sistema pueda ser reutilizado para cualquier otro estudio de este tipo de sistemas.

- Se ha empleado Lenskit, un poderoso framework de recomendación para ayudar a la implementación de la capa del *motor de recomendación*. Es que de esta manera se pueden tener varios tipos de algoritmos para el funcionamiento del sistema, que luego son configurados en los archivos de configuración vistos en el capítulo 3-4.

- Con el conjunto de datos de prueba mencionado en el capítulo 5.4, se ha podido demostrar la viabilidad de la arquitectura mediante una implementación expuesta en el capítulo 4.

- Como se han podido observar en los experimentos realizados se ha podido llegar a ver gráficamente y por resultado de tiempos en milisegundos representados en tablas la diferencia entre ambos gestores de bases de datos. La diferencia en promedio de los diferentes experimentos no llega a los mil milisegundos. Se puede llegar a concluir que por la arquitectura el sistema de recomendación con MongoDB llega a responder más rápido en comparación de MsSqlServer en todos y cada uno de los experimentos individuales por usuario y por contexto.

- Se ha sacado ventaja por parte de MongoDB en la creación de colecciones dinámicas para el almacenamiento de información contextual en la arquitectura propuesta.

6.3. Trabajo futuro

Por la arquitectura que tiene el sistema de recomendación, se pueden llegar a realizar muchas mejoras y continuar investigando en diferentes módulos de la arquitectura. Se pueden realizar mejoras e investigaciones en los siguientes puntos:

- Hacer análisis en diferentes dominios de sistemas de comercio electrónico, pudiendo ser estos de venta de libros, música, videos, ropa, etc. De esta forma poder medir el nivel de adaptabilidad en los diferentes sistemas.

- Realizar más pruebas de comparación utilizando diferente algoritmos contenidos en el framework Lenskit o utilizar otro framework de recomendación en la capa de motor de recomendación.

- Hacer del módulo de reglas de contexto, algo más complejo e ir adaptando todos estudios realizados hasta ahora en la literatura científica para así poder manejar contextos y de esta forma extender el sistema a nuevos estudios.

- Realizar tipos de experimentación con registro de recomendaciones o información contextual, en este tema se han escrito varias comparativas en internet, y es en este tipo de experimentación donde este tipo de gestores NoSql toman mucha mayor ventaja en cuanto los gestores de base de datos tradicionales.

En resumen la arquitectura presentada en este trabajo para el sistema de recomendación contextual, sirve para poder implementarla con diferentes herramientas y tecnologías, por ejemplo en este trabajo en particular se han empleado tecnologías tipo MongoDB como motor de base de dato y Lenskit como framework de recomendación, pero se podría utilizar otra serie de opciones ofrecidas en el mercado y adaptarlas a las necesidades que se vayan a requerir.

Este proyecto y su correspondiente modelo de arquitectura para un sistema de recomendación contextual, pretende ser una herramienta y opción para el manejo de recomendaciones contextuales utilizando un gestor de base de datos NoSql en la forma de trabajar un pre-filtrado almacenado en colección de MongoDB. Aprovechando las muchas ventajas que nos ofrecen este tipo de sistemas, entre los que se encuentran su escalabilidad y performance. Con la experimentación presentada en el capítulo anterior podemos observar que de alguna manera se ha llegado a superar a un gestor de base de datos tradicional. Hay que afirmar que son solo un par de pruebas de las muchas más que se podrían llegar a hacer para comparar estos dos tipos de gestores de base de datos y además sacar ventaja de las muchas características que conlleva un gestor de base de datos NoSql.

Parte I

Apéndices

Apéndice A

Instalación principales componetes

El éxito en la vida no se mide por lo que has logrado, sino por los obstáculos que has tenido que enfrentar en el camino.

Valerie B.

A.1. Descripción general

En esta sección se describirán la instalación del software empleado en la implementación del sistema de recomendación.

A.2. Instalación de los principales componentes

Para que el sistema llegue a estar operativo se deben de instalar los principales componentes requeridos.

A.2.1. Instalación MongoDB

Para la instalación de esta base de datos se revisó la documentación presentada en el sitio web del producto[1]. No se tuvieron problemas o incidentes en el momento de la instalación. Una vez que la base de datos entra en funcionamiento como se puede ver en la figura A.1, se pasa a la creación de una colección principal para el almacenamiento de los documentos que contendrán los ratings de los usuarios y los ítems.

[1] http://docs.mongodb.org/manual/tutorial/install-mongodb-on-windows/

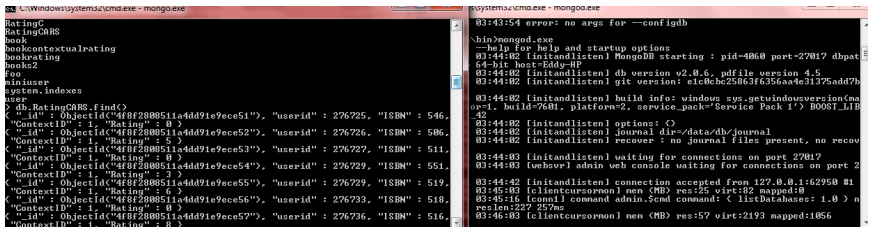

Figura A.1: Funcionamiento de MongoDB.

A.2.2. Instalación Apache Tomcat 7.0

Par el funcionamiento de los servicios es necesario tener instalado un servidor web, en este caso se utilizó Apache Tomcat 7.0 también llamada Jakarta Tomcat [2]. Para la instalación de este servidor web se utilizó la versión 7.0 obtenida su página web oficial. http://tomcat.apache.org/download-70.cgi. Toda la instalación se la realizo dentro de un entorno Windows creando así sus variables de entorno dentro del sistema.

[2] Tomcat funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Este implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

Bibliografía

[1] G. Adomavicius and A. Tuzhilin. Incorporating context into recommender systems using multidimensional rating estimation methods. *In Proceedings of the 1st International Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces (WPRSIUI 2005)*, 2005.

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 335–336, New York, NY, USA, 2008. ACM.

[3] R. D. A.K. Jain. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[4] M. Balabanovic. Exploring versus exploiting when learning user models for text recommendation. *User Model. User-Adapt. Interact.*, 8(1-2):71–102, 1998.

[5] M. y. S. Balabanovic. Content-based, collaborative recommendation. *Communications of the ACM*, 40(3), 1997.

[6] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2(4):263–277, June 2007.

[7] J. Benitez. 5+ base de datos nosql para mejorar la performance de tus aplicaciones, Mar 2012.

[8] G. Burd. Nosql, Oct 2011.

[9] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov. 2002.

[10] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, Hanover, NH, USA, 2000.

[11] Codecriticon.com. ¿qué es nosql?, Nov 2011.

[12] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact.*, 16(2):97–166, Dec. 2001.

[13] M. D. Ekstrand, M. Ludwig, J. A. Konstan, and J. T. Riedl. Rethinking the recommender research ecosystem: reproducibility, openness, and lenskit. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 133–140, New York, NY, USA, 2011. ACM.

[14] D. D. L. L. ESPAÑOLA. Rae.

[15] S. S. G. Adomavicius, R. Sankaranarayanan and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23(1):103–145, 2005.

[16] X. Guo and J. Lu. Intelligent e-government services with personalized recommendation techniques. *International Journal of Intelligent Systems*, 22(5):401–417, 2007.

[17] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Jan. 2004.

[18] A. I. E. Juan Juli´an Merelo Guerv´os. Usando bases de datos nosql para algoritmos evolutivos paralelos.

[19] K. Kim, H. Ahn, and S. Jeong. Context-aware Recommender Systems using Data Mining Techniques.

[20] T. Mahmood, F. Ricci, and A. Venturini. Improving recommendation effectiveness: Adapting a dialogue strategy in online travel planning. *Information Technology and Tourism*, 11(4):285–302, 2009.

[21] C. C. C. C. F. J. M. L. V. M. Moreno. Sistemas de recomendación en el comercio electrónico y la educación. *Criterio Libre*, 8(12):161–182, 2010.

[22] D. G. Nuno Santos, Óscar M. Pereira. Context storage using nosql. Technical report, Coimbra, Portugal, 2011.

[23] J. K. H. K. Panu Korpipää, Jani Mäntyjärvi and E.-J. Malm. Managing context information in mobile devices. Technical report, Coimbra, Portugal, 2003.

[24] A. O. W. Paper. Oracle nosql database. Sep 2011.

[25] V. S. Prem Melville. *Recommender Systems.* Encyclopedia of Machine Learning, 2010.

[26] G. Research. Lenskit.

[27] C. Schmidt. Context-aware computing. *Computer Engineering Bachelor of Science Berlin Institute of Technology.*

[28] U. Shardanand and P. Maes. Social information filtering: algorithms for automating word of mouth. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, 1995.

[29] C. Strauch. Nosql databases, 2011.

[30] E. Vozalis and et al. Analysis of recommender systems algorithms.

[31] Wikipedia. Nosql.

[32] T. Yamabe, A. Takagi, and T. Nakajima. Citron: A context information acquisition framework for personal devices. In *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, RTCSA '05, pages 489–495, Washington, DC, USA, 2005. IEEE Computer Society.

Autorización de Difusión

Eddy Rodriguez Vargas

Febrero de 2013

El abajo firmante, matriculado en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "Sistemas de recomendación sensibles al contexto con bases de datos NoSQL", realizado durante el curso académico 2012-2013 bajo la dirección de Pedro Antonio González Calero y Guillermo Jiménez Díaz en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.