Trabajo de Fin de Grado

Sistema recomendador basado en contenidos.

Recommender system based in content

Katerine Cardona de la Pava

La Laguna, 8 de Septiembre de 2015

Dra. **Carina Soledad González González**, con N.I.F. 54.064.251-Z profesora Titular de Universidad adscrito al Departamento ingeniería informática y de sistemas de la Universidad de La Laguna, como tutor

Dr. **Yeray Del Cristo Barrios Fleitas**, con N.I.F. 54.106.27-R personal investigador del Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Sistema recomendador basado en contenidos."

Ha sido realizada bajo su dirección por D. **Katerine Cardona de la Pava**, con N.I.F. 51.206.290-X.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 8 de Septiembre de 2015.

Agradecimientos

Quisiera agradecer a los docentes implicados Carina y Silvana por la colaboración prestada para la realización del trabajo.

A Yeray por aguantarme y tener tanta paciencia conmigo y guiarme a llevar a buen fin el proyecto, por estar ahí siempre muchas gracias.

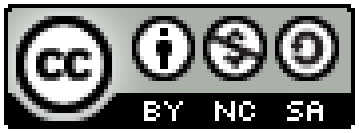
Sobre todo a mi familia por estar ahí apoyándome día a día y aguantando mi genio.

A los profesores que me han acompañado en todo estos años que me han enseñado e inculcado el valor de aprender y el cómo aprender para un mejor futuro.

Gracias a cada uno por aportar un granito de arena para formarme en una mejor ingeniera.

Licencia

* Si quiere permitir que se compartan las adaptaciones de tu obra mientras se comparta de la misma manera y NO quieres permitir usos comerciales de tu obra indica:



Resumen

El gran avance tecnológico que se ha incrementado sustancialmente al pasar del tiempo ha dado lugar a tener que reinventar las formas de poder almacenar tanta información y poder reestructurarla por ello es que a día de hoy contamos con muchas ayudas para poder llevar a cabo dicha tarea, por esta razón los sistemas recomendadores han tenido un papel muy importante ayudando a la simplificación de muchas de estas tareas a realizar, como la búsqueda de datos en cualquier campo.

Los sistemas recomendadores son ahora una opción muy usada para realizar sugerencias a los usuarios tanto para personas indecisas en que quieren como para personas que lo saben pero quieren más opciones de elección.

Se han creado dos versiones que enseñara a hacer una aplicación más dinámica y poco acoplada.

Palabras clave: sistema recomendador, docentes, contenido, mahout, aplicación.

Abstract

The technological breakthrough that has substantially increased the passage of time has resulted in having to reinvent the ways of storing information and to restructure it so that is why today we have much help to carry out this task, therefore recommender systems have played an important role in helping the simplification of many of these tasks, such as searching data in any field.

The recommender systems are now widely used option for users to make suggestions for both undecided people that want to know but people want more choice.

Two versions were created to teach to make a more dynamic and loosely coupled application.

Keywords: recommender systems, educational, contents, mahout, application.

Índice General

Item[1]: Cada uno de los elementos que forman parte de un dato

Item[1]: Cada uno de los elementos que forman parte de un dato

Índice de figuras

Item[1]: Cada uno de los elementos que forman parte de un dato

Índice de tablas

Capítulo 1. Introducción

1.1 Introducción.

Debido a la modernización en la que cada vez estamos más sumergidos en el mundo de las tecnologías se han creado una cantidad de estrategias para poner manejar toda la información que se ha venido creando en todos los ámbitos. Tal es así que cada día es más complejo el tratamiento de la información.

En todas las áreas de interés hay gran cantidad de información que se trata de distintas maneras y para diferentes cosas. Por ello, una buena técnica para este uso son las técnicas de recomendación, que son un proceso en el cual, basados o no en las características de un usuario, podemos ayudarle a encontrar lo que sea afín a él. Esta es una buena forma de ayudar a las personas a poder clasificar información que le sea útil en algún momento.

Por otro lado, los sistemas recomendadores están en auge para fines educativos, ya que se podrá hacer interacción con los estudiantes, su contenido y actividades entre ellos.

La implementación del sistema recomendador es una parte de un macro proyecto llamado EDhospi (aulas hospitalarias), donde hay involucradas una cantidad de personas para la realización de este y, con este sistema, se ha de aportar un valor añadido al proyecto.

1.2 Motivación.

El excesivo uso de las tecnologías por parte de nuestra sociedad, aso como la ingente cantidad de búsquedas diarias que se hacen en buscadores como Google, nos ha llevado a pensar que sería interesante investigar

Item[1]: Cada uno de los elementos que forman parte de un dato

acerca de los sistemas recomendadores y sus distintas variantes. Su uso en proyectos puede ser muy beneficioso de cara a mejorar la experiencia de usuario y la eficacia de los sistemas.

Por otro lado, la posibilidad de aprender nuevas e interesantes tecnologías para la integración de bases de datos en web y servidores, supone un añadido a mí formación académica de valor, ya que este tipo de sistemas son los más utilizados en el actual mercado de desarrollo web y sobre todo la aportación que se le puede hacer al proyecto EDhospi.

1.3 Estructura de la memoria.

Esta memoria se estructura con el inicio de un estudio previo sobre los tipos de sistemas recomendadores que hay, siendo este el estado del arte. Después de saber los tipos de recomendadores que hay, se decidió escoger uno a implantar para realizar el software. Luego se analizaron los tipos de requerimientos y casos de usos para dicha implantación. Por último el software utilizado e instalación de estos. Para resumir la estructura será la siguiente:

- **Estado del arte:** estudio sobre los sistemas de recomendación.
- **Recomendación escogida:** explicación del método de sistema recomendador implantado en el sistema.
- **Especificación de los requisitos:** explicación de la aplicación.
- **Desarrollo:** descripción de todo el proceso para creación de aplicación.
- **Evaluación:** revisión del sistema recomendador y aplicación.
- **Presupuesto.**
- **Conclusiones.**

1.4 Objetivo.

Los objetivos marcados para este trabajo final de grado son los listados a continuación:

- Creación de un sistema recomendador basado en contenido mediante perfiles de usuarios.
- Verificar la eficiencia del sistema recomendador mediante la evaluación de este.
- Creación de la una aplicación web para presentar el funcionamiento del sistema recomendador.

1.5 Metodología.

Para realizar el proyecto se dividirá en 4 fases las cuales dan una idea de cómo se llevaran a cabo las tareas a realizar:

Fase I: Análisis del proyecto.

Esta fase es previa al comienzo del desarrollo de la aplicación donde se creara un estudio de los sistemas que hay y poder analizarlos y optar por uno en particular.

- **Tarea 1:** Estudio de los distintos sistemas dando una duración de 28 horas para realizarlo y realizar un documento recogiendo dicha información.

Fase II: Interfaz de usuario.

Esta fase contempla todo lo relacionado con el desarrollo del sistema dando tareas para una mejor organización de lo que hay por hacer dándole una duración de unas 94 horas aproximadamente.

- **Tarea 1:** instalar el software necesario para realizar esta fase
- **Tarea 2:** Realizar el código para conseguir las vistas de usuario.
- **Tarea 3:** Realizar pruebas de uso.
- **Tarea 4:** documentar lo hecho.

Item[1]: Cada uno de los elementos que forman parte de un dato

Fase III: Sistema recomendador.

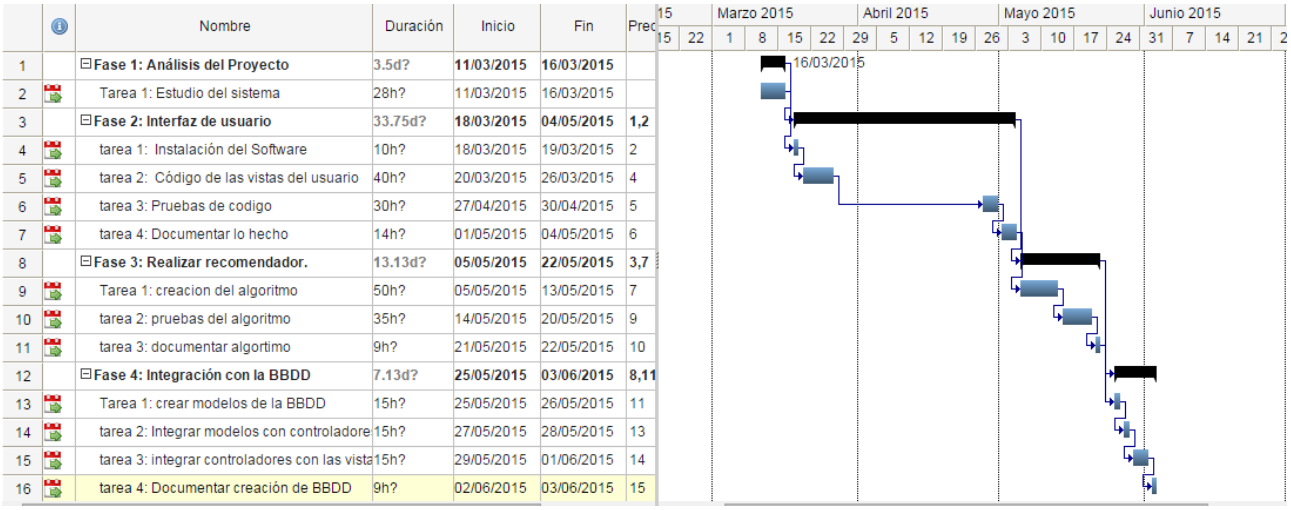
Esta fase se realizara todo lo concerniente al sistema recomendador con una duración de unas 94 horas.

- **Tarea 1:** creación del algoritmo de recomendación.
- **Tarea 2:** pruebas del algoritmo.
- **Tarea 3:** Documentar el sistema.

Fase IV: Integración con la base de datos.

En esta fase se hará la integración de la base de datos con toda la lógica del proyecto para guardar toda información con una duración de 54 horas.

- **Tarea 1:** Realizar modelos necesarios para crear base de datos.
- **Tarea 2:** Integrar los modelos con los controladores del sistema.
- **Tarea 3:** Integrar los controladores con las vistas de usuario.
- **Tarea 4:** Documentar integración.



	Nombre	Duración	Inicio	Fin	Pred
1	**Fase 1: Análisis del Proyecto**	3.5d?	**11/03/2015**	**16/03/2015**	
2	Tarea 1: Estudio del sistema	28h?	11/03/2015	16/03/2015	
3	**Fase 2: Interfaz de usuario**	33.75d?	**18/03/2015**	**04/05/2015**	**1,2**
4	tarea 1: Instalación del Software	10h?	18/03/2015	19/03/2015	2
5	tarea 2: Código de las vistas del usuario	40h?	20/03/2015	26/03/2015	4
6	tarea 3: Pruebas de codigo	30h?	27/04/2015	30/04/2015	5
7	tarea 4: Documentar lo hecho	14h?	01/05/2015	04/05/2015	6
8	**Fase 3: Realizar recomendador.**	13.13d?	**05/05/2015**	**22/05/2015**	**3,7**
9	Tarea 1: creacion del algoritmo	50h?	05/05/2015	13/05/2015	7
10	tarea 2: pruebas del algoritmo	35h?	14/05/2015	20/05/2015	9
11	tarea 3: documentar algortimo	9h?	21/05/2015	22/05/2015	10
12	**Fase 4: Integración con la BBDD**	7.13d?	**25/05/2015**	**03/06/2015**	**8,11**
13	Tarea 1: crear modelos de la BBDD	15h?	25/05/2015	26/05/2015	11
14	tarea 2: Integrar modelos con controladore	15h?	27/05/2015	28/05/2015	13
15	tarea 3: integrar controladores con las vista	15h?	29/05/2015	01/06/2015	14
16	tarea 4: Documentar creación de BBDD	9h?	02/06/2015	03/06/2015	15

Figura 1. Planificación del proyecto.

Capítulo 2. Antecedentes y estado actual del sistema.

En el capítulo anterior se ha explicado cómo será introducido el trabajo, pero en este capítulo se hará una introducción a los sistemas recomendadores y sus tipos para entender mejor su funcionamiento.

2.1 Sistemas recomendadores.

Los sistemas recomendadores (RS) son sistemas que ayudan a emparejar usuarios con productos. Tienen el potencial para apoyar y mejorar la calidad de la toma de decisiones de los usuarios, mientras que se realiza la búsqueda y selección de productos.

Cada usuario obtiene una lista de recomendación diferente según criterios específicos (ej. Los más valorados/consumidos/vistos), requiriendo a los recomendadores que tengan un modelo de usuario (ej. Historial de visitas, likes, etc.).

Existe un campo bastante amplio para adaptar estos recomendadores como lo pueden ser: la gestión del conocimiento, creación de perfiles, interfaces de usuario, aspectos sociológicos y psicológicos, etc.

Figura 2. Esquema de tipos de sistemas recomendadores.

Item[1]: Cada uno de los elementos que forman parte de un dato

2.2 Formas de adquisición de información de usuarios.

Para crear un perfil de un usuario con sus gustos, preferencias e intereses, se puede utilizar dos formas o métodos en la recolección de información: implícita o explícita.

- **La información implícita:** Es aquella que el sistema adquiere a partir del comportamiento e interacción del usuario con el sistema. Por ejemplo:
  - registro de los ítems que el usuario ha visto en una tienda online.
  - analizar el número de visitas que recibe un ítem[1].
  - analizar las redes sociales de las que el usuario forma parte y de esta manera conocer sus gustos y preferencias.
  - hacer un análisis del historial de compra del usuario.
  - comparar el tiempo que el usuario pasa leyendo sobre ciertos ítems en el sitio en contraste contra el tiempo que dedica a otros ítems.
  - introducción de palabras claves en una búsqueda.
  - analizar la cantidad de clic que hace el usuario sobre determinada sección del sitio.

- **La información explícita:** Es la información que el usuario ingresa al sistema en respuesta a las peticiones manifiestas del mismo. Por ejemplo:
  - solicitar al usuario que pondere un conjunto de ítems de una lista de ítems.
  - presentar al usuario dos ítems, y solicitarle que seleccione uno de ellos.
  - solicitar al usuario.

2.3 Tipos de sistemas recomendadores.

2.3.1 Recomendación colaborativa.

Si los usuarios han compartido algunos de sus intereses en el pasado, tendrán gustos similares en el futuro, A y B tienen una historia parecida. A compra un libro que B no ha visto. Lo más razonable es proponer ese libro también a B (enséñame lo que es popular entre mis vecinos).

Es la técnica más empleada sus ventajas, rendimiento y limitaciones son bien conocidos. La Idea es Utilizar lo que dice la gente para recomendar elementos:

- La suposición fundamental y motivación donde los usuarios asignan puntuaciones a los elementos disponibles (implícitas o explícitas).
- Usuarios que han tenido gustos similares en el pasado, tendrán gustos similares en el futuro.
- De entrada se cuenta con una Matriz de usuarios-elementos.
- De salida se obtiene una predicción numérica sobre el grado de interés de cada elemento, una lista de n-elementos recomendados, que por supuesto no contiene aquellos ya vistos por el usuario.

La principal desventaja que presenta ésta técnica es que requiere información de un número elevado de usuarios para que las recomendaciones sean precisas.

Además, el uso de ésta técnica, cuenta con otras problemáticas expuestas a continuación:

- **nuevo usuario:** Estos sistemas de recomendación suelen basarse en la puntuación que el usuario otorga a los ítems. Por ello, si el usuario es nuevo, probablemente tendrá pocos ítems puntuados, lo que provocará como consecuencia pocos ítems de comparación. Por lo tanto, los usuarios recién registrados necesitan puntuar suficientes ítems para que el sistema recomiende con una cierta garantía.

Item[1]: Cada uno de los elementos que forman parte de un dato

- **nuevo ítem:** El origen de éste problema se da cuando un nuevo ítem es dado de alta en el sistema, ya que cuenta con un bajo número de puntuaciones por parte del usuario. Por lo tanto, éste ítem será difícilmente recomendable hasta que su número de puntuaciones sea incrementado.
- **Dispersión de ratios:** El problema únicamente es aplicable en usuarios con gustos poco comunes, debido a que sus puntuaciones no coinciden con las de otros, lo que conlleva que el sistema encuentre pocos usuarios con los que compara o pocos elementos con los que buscar similitudes.

Figura 3. Esquema de un sistema colaborativo.

2.3.2 Recomendación colaborativa según vecino más cercano:

La técnica básica: Dado un "usuario activo" y un elemento i no visto por el usuario, encontrar el grupo de usuarios (vecinos cercanos) a los que les han gustado los mismos elementos que a María en el pasado y que han valorado el elemento i, Utilizar las valoraciones para predecir si al usuario le gustará i, Luego, repetir esto para todos los elementos no vistos por el usuario y recomendar los mejores.

- Suposición fundamental e idea: Si los usuarios han tenido gustos similares en el pasado, tendrás gustos similares en el futuro.
- Las preferencias/gustos de un usuario permanecen estables y consistentes en el tiempo.

Para realizar los cálculos de similitud se utiliza la técnica de correlación de Pearson.

$$sim(a,b) = \frac{\sum_{p \in V_{a,b}} (V_{a,p} - \overline{V}_a)(V_{b,p} - \overline{V}_b)}{\sqrt{\sum_{p \in V_{a,b}} (V_{a,p} - \overline{V}_a)^2} \sqrt{\sum_{p \in V_{a,b}} (V_{b,p} - \overline{V}_b)^2}}$$

Figura 4. Fórmula de correlación de Pearson.

2.3.3 Recomendación colaborativa basado en elementos.

La idea general: Utilizar la similitud entre elementos (y no entre usuarios) para hacer las predicciones tomando como referencia la similitud del coseno para poder hallar predicción.

2.3.4 Recomendación basada en contenido.

Partiendo de una base de elementos con descripción (características) y un perfil con asignaciones de importancia a estas características. Emparejar aquellos elementos que mejor cumplan las preferencias del usuario. (Muéstrame más de lo que ya me gustado).

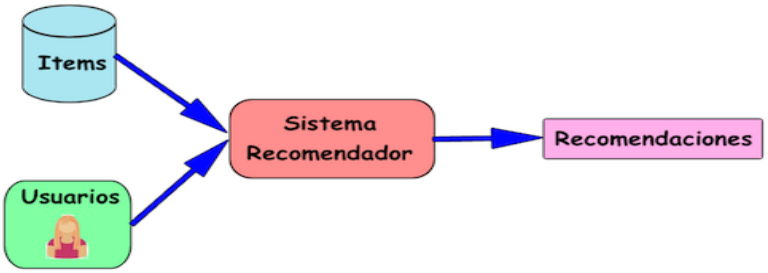


Figura 5. Esquema de sistema recomendador por contenido.

2.3.5 Recomendación basada en conocimiento:

En muchas ocasiones nos encontraremos con usuarios novatos. No tenemos historial ni preferencias. Son recomendadores que están más centrados en el dominio de aplicación y necesitan información adicional para

Item[1]: Cada uno de los elementos que forman parte de un dato

poder realizar las recomendaciones. Esto significa, información sobre el usuario y/o sobre los elementos (enséñame lo que se adapte a mis necesidades).

2.3.5.1 Basado en restricciones (Constraint-based):

A partir de un conjunto de reglas de recomendación explícitamente definidas, Satisfacen todas las reglas de recomendación

2.3.5.2 Basado en casos (Case-based):

Basado en diferentes métricas de similitud.

- Recuperar elementos que son similares a los requisitos especificados.

El usuario especifica sus requisitos, y el sistema trata de identificar soluciones, Si no se encuentra una solución, el usuario debe "cambiar" sus requisitos.

2.3.6 Recomendación Hibrida:

Combinación de varias de las técnicas anteriores para obtener mejores recomendaciones. La mayoría de sistemas que usan la técnica de recomendación híbrida, están compuestos por dos o más sistemas de recomendación, de los cuáles son combinados de distintas maneras para obtener un resultado más fiel.

Capítulo 3. Especificación de Requisitos.

La especificación de los requisitos son las características con las que se describe el comportamiento del sistema. En los requisitos funcionales es donde se describe toda la actividad que realizara el sistema, y los requisitos no funcionales son los que describirán las características en técnicas del sistema.

El objetivo fundamental es hacer una aplicación web en la que realicen las recomendaciones basadas en un perfil mediante la técnica de recomendación por contenido.

3.1 Requisitos Funcionales.

El sistema tendrá que cumplir los siguientes requisitos de funcionamiento:

1. **Registrar Usuarios:** El sistema registrara a cualquier usuario que quiera acceder a la plataforma para prestarle el servicio de recomendación, con esto dando lugar a ser dado de alta en el sistema, el cual luego podrá modificar sus datos o darse de baja si lo desea.
2. **Registrar Categorías:** El sistema registrara categorías que no estén ya en el sistema y de las que quiera el usuario hacer uso.
3. **Recomendar Ítem de las categorías:** La recomendación se hará mediante un algoritmo en el que se podrá sugerir datos a un usuario que sea de su interés.
4. **Almacenamiento de los datos:** Los datos se almacenaran en una base de datos, listando los usuarios con los perfiles que se harán a medida que se extraiga la información de su interés.

3.1.1 Actores

Los actores serán los usuarios que interactuaran con la aplicación. Por ello contamos con dos situaciones:

Item[1]: Cada uno de los elementos que forman parte de un dato

1. **Usuario no registrado:** Es el actor nuevo que llega por primera vez al sistema y del cual no se tiene ninguna información.
2. **Usuario Registrado:** Es el actor que ya está registrado en el sistema del cual se puede extraer la información para realizar una recomendación. Este usuario podrá acceder a todas las funciones de sistema (modificar, darse de baja, ver recomendaciones, etc.).
3. **Servidor:** este actor es el que guarda toda información de la base de datos, con una arquitectura cliente-servidor (el sistema solicita o envía información al servidor).

3.1.2 Casos de uso.

Los casos de uso lo que me proporcionan es una descripción de las iteraciones que realiza el actor con el sistema.

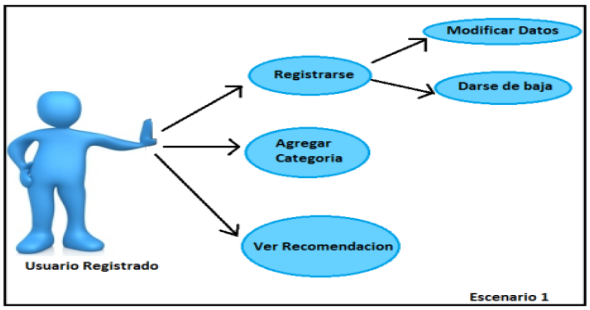
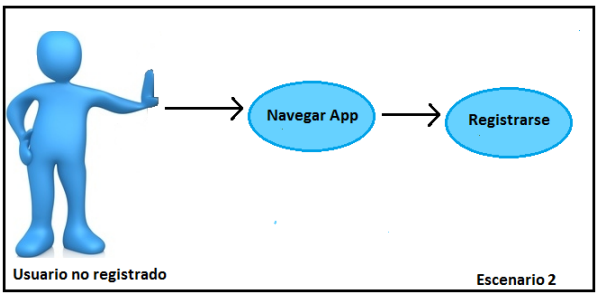
Usuario Registrado	Usuario no Registrado
 **Figura 6.** Diagrama de caso de uso para un usuario registrado.	 **Figura 7.** Diagrama de caso de uso de usuario no registrado.

Tabla 1. Imágenes de usuarios.

3.1.2.1 Descripción de los casos de uso:

Nombre	Registro de nuevo usuario.
Objetivo	Llevar a cabo el registro del usuario.

Actores	Usuario no registrado.
Descripción	El usuario no registrado navegara por la aplicación hasta encontrar la opción.
Post-Condición	El usuario ya puede hacer su perfil de usuario para mostrarle una recomendación.

Tabla 2. Caso de uso de un usuario sin registrar.

Nombre	Modificar datos de usuario.
Objetivo	Llevar a cabo la modificación de los datos de registro de un usuario.
Actores	Usuario registrado.
Descripción	El usuario registrado podrá acceder a los datos proporcionados y modificarlos.
Post-Condición	Se ha de visualizar los cambios.

Tabla 3. Caso de uso de modificación de datos de un usuario.

Nombre	Mostrar datos de categorías.
Objetivo	Ver todas las categorías que hay en el sistema.
Actores	Usuario registrado.
Descripción	El usuario registrado podrá ver todas las categorías que hay en nuestro sistema.
Post-Condición	Se ha de visualizar las categorías que hay y pudiendo añadir otro si no hay alguna de su interés.

Tabla 4. Caso de uso de mostrar datos de categorías.

Nombre	Ver recomendaciones.
Objetivo	Mostrar un listado con los ítems recomendados según datos del usuario.
Actores	Usuario registrado.
Descripción	El usuario registrado podrá visualizar las recomendaciones hechas según sus preferencias.

Item[1]: Cada uno de los elementos que forman parte de un dato

Post-Condición	Se ha de mostrar al usuario las recomendaciones que se han obtenido según su interés.

Tabla 5. Caso de uso de ver recomendaciones.

3.2 Requisitos no Funcionales

Estos requisitos como ya había explicado serán las características técnicas que tendrá el sistema.

3.2.1 Interfaz.

Ha de ser una interfaz sencilla de manera que se puedan visualizar bien los módulos de la aplicación con imágenes y colores agradables a la vista, y con un tamaño acorde para la correcta visualización.

3.2.2 Rendimiento.

Ha de tener un tiempo rápido de respuesta para todas las funcionalidades como lo son el registro de usuario o la actualización de datos hasta llegar a la recomendación.

La recomendación estará optimizada para garantizar al usuario un tiempo de respuesta corto.

3.2.3 Fiabilidad, seguridad.

El sistema ha de asegurar a los usuarios la fiabilidad y seguridad de sus datos teniendo en cuenta la ley orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal garantizando la no divulgación de estos.

3.2.4 Usabilidad.

El sistema tendrá que ser flexible y débilmente acoplada para garantizar su crecimiento sin tener que cambiar demasiado, así poder añadir nuevos requerimientos de futuro.

3.2.5 Restricciones

Para la primera versión del sistema recomendador, se han de tener disponibles los siguientes programas para poder probar la aplicación:

- Xampp: con el Apache y mysql para correr el servidor y la base de datos creada con phpMyadmin.
- Eclipse java EE IDE for web developer's version Luna.
- Java Lenguaje escogido para esta versión.

Para la segunda versión del sistema recomendador se ha de tener instalados los siguientes programas:

- **mongoDB:** para crear la base de datos.
- **Express:** framework server web.
- **AngularJS:** framework web cliente.
- **NodeJS:** Plataforma del servidor.
- **Javascript:** Lenguaje escogido para realizar esta versión.

Item[1]: Cada uno de los elementos que forman parte de un dato

Capítulo 4.
Arquitectura de la aplicación.

La arquitectura con la que hemos enfocado las dos versiones del sistema es el modelo-vista-controlador (**MVC**), enfocado de cierta forma según el caso de cada versión.

4.1 Patrón Arquitectónico utilizado.

Es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello se propone la construcción de tres componentes distintos que son el **modelo**, la **vista** y el **controlador.** Por un lado se definen componentes para la representación de la información, y por otro lado para la interacción del usuario.

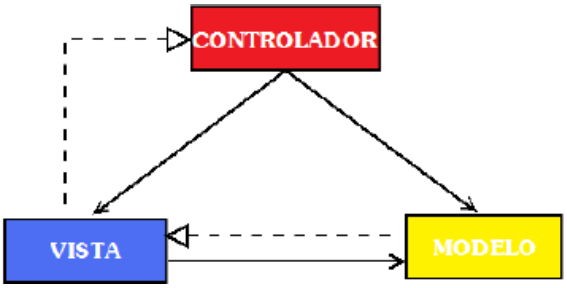


Figura 8. Esquema de MVC.

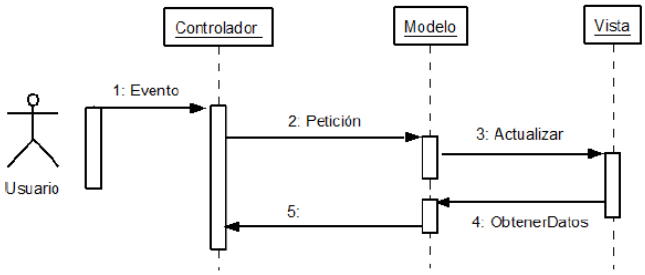


Figura 9. Ciclo del MVC.

4.1.1 Ventajas del MVC:

- Es posible tener diferentes vistas para un mismo modelo.
- Es posible modificar las vistas (o construir nuevas) sin modificar el modelo subyacente.
- Proporciona un mecanismo de configuración a componentes complejos mucho más tratable que el puramente basado en eventos.

4.1.2 MVC en aplicaciones de escritorio (Utilizada en Vers.1).

En una aplicación de escritorio, el Modelo, la Vista, y el Controlador se ejecutan en la maquina con la cual el usuario interactúa.

- **Modelo:** Lo realiza el desarrollador.
- **Vista:** Conjunto de clases que heredan de los componentes gráficos (Ventanas, etc.).
- **Controlador:** El thread de tratamiento de eventos, que captura y propaga los eventos a la vista y al modelo Clases de tratamientos de eventos, que implementan interfaces tipo EventListener.

4.1.3 MVC Programación Web (Utilizada en Versión 2).

El código que se ejecuta en el cliente incluye el navegador, La aplicación debe mostrar los datos que están en el servidor en un contexto ajeno en donde el cliente pueda visualizarlos correctamente, La información sobre los eventos del usuario debe volver al servidor y reflejar los cambios.

- **Vista:** La página HTML.
- **Controlador:** Código que obtiene datos dinámicamente y genera el contenido HTML.
- **Modelo:** La información almacenada en una base de datos, Junto con las reglas de negocio que transforman esa información (teniendo en cuenta las acciones de los usuarios).

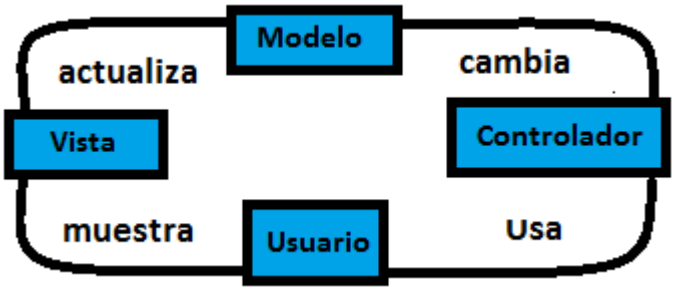


Figura 10. Estructura Web.

Item[1]: Cada uno de los elementos que forman parte de un dato

Capítulo 5.
Versión 1 del sistema recomendador

Esta versión del sistema recomendador es hecha con mahout, que nos proporciona una serie de librerías que calculan la recomendación según los datos proporcionados mediante un fichero.

El desarrollo de la vista para mostrar al usuario el funcionamiento del sistema recomendador se realizó con lenguaje java, utilizando el programa eclipse.

5.1 Fases de desarrollo de mi aplicación.

Para la realización de la aplicación se ha de tener encuentra que será un prototipo de cómo se podría ver de cara al usuario.

5.1.1 Creación de la Base de Datos:

Para la creación de la base de datos se utilizó el programa xampp que me proporciona la creación de la base de datos relacional en el localhost mediante el phpMyadmin.

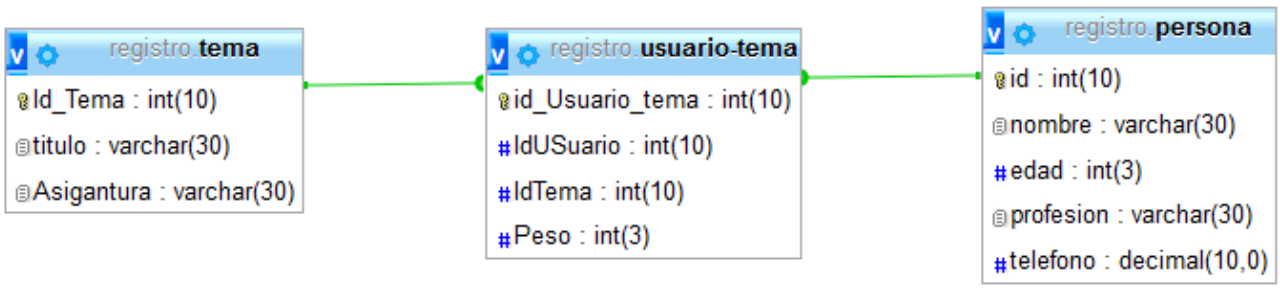


Figura 11. Esquema de BBDD.

La tabla persona: me recoge toda la información del registro de usuario.

La tabla Categoría: me recoge todos los datos de las categorías en las que el usuario quiere que le recomienden.

La tabla usuario-tema: tendrá referencias las claves de las tablas persona y categoría como clave foráneas.

De estas tablas se recogerá: el id de la tabla persona, el id de la tabla Categoría, y el peso de la tabla usuario-Tema para insertarlos en un fichero que me pedirá el recomendador [Mahout] para hacer la recomendación.

5.1.2 Creación de las ventanas de interacción con el usuario:

La vista de la aplicación se creara por medio de un gestor de ventanas creadas con el programa eclipse mediante el lenguaje java.

Utilizando el patrón MVC (modelo, vista, controlador) [Capitulo 4.1.2]

Ventana principal en la que el usuario puede escoger si quiere registrarse o en caso de que ya este registrado:

Figura 12. Ventana de decisión de usuario.

Si el usuario escoge la opción usuario sin registrar se mostrara otra ventana para realizar el registro del usuario.

Figura 13. Ventana de Registro de Usuario.

Si el usuario ya está registrado tendrá las opciones de modificar usuario, crear categoría, modificar categoría.

Item[1]: Cada uno de los elementos que forman parte de un dato

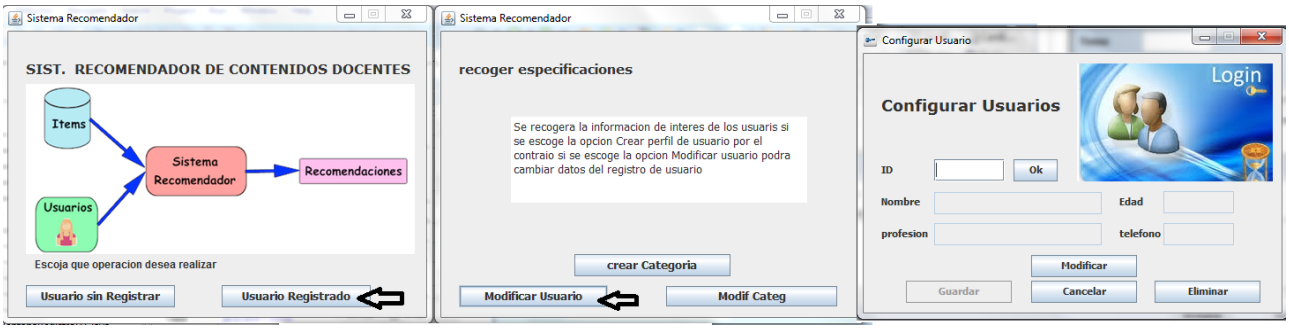


Figura 14. Ventana de configuración de usuario.

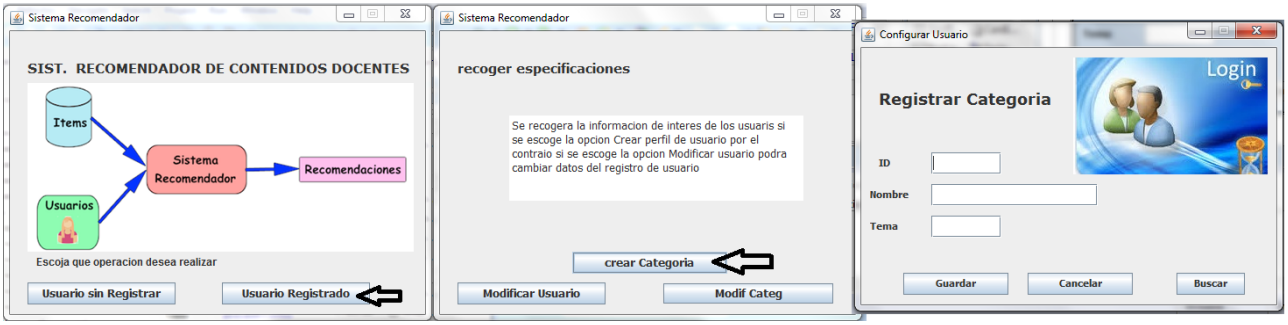


Figura 15. Ventana de Creación Categoría.

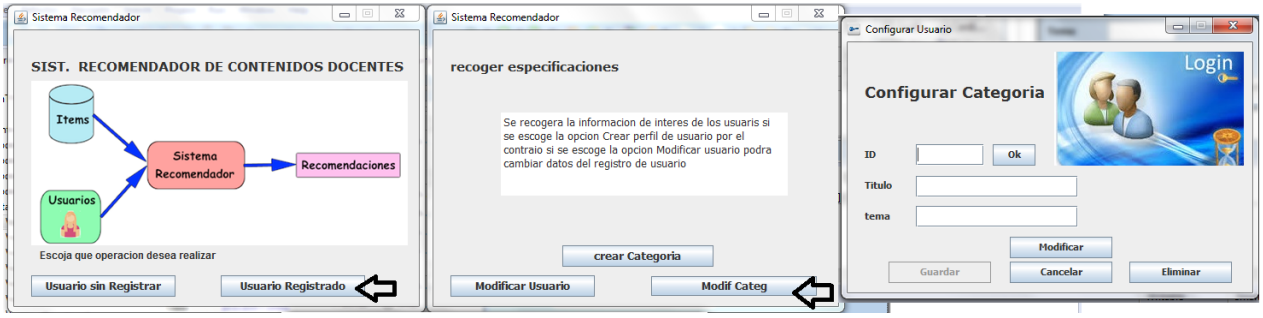


Figura 16. Ventana de configuración de categoría.

5.1.3 Recomendador con librería Mahout:

5.1.3.1 ¿Qué es?

Apache Mahout es un proyecto de la Fundación Apache Software para producir implementaciones libres de algoritmos de aprendizaje automático.

Se centraron principalmente en las áreas de colaboración filtrada, agrupación y clasificación. Muchas de las implementaciones utilizan la plataforma Hadoop.

Mahout también proporciona bibliotecas de Java para las operaciones matemáticas comunes (centrado en álgebra lineal y estadística) y las

colecciones de Java primitivos. Mahout es un trabajo en progreso, pero el número de algoritmos implementados ha crecido rápidamente, a pesar de que varios algoritmos siguen desaparecidos.

5.1.3.2 Pasos para implementación.

El uso de Mahout para la recomendación incluye los siguientes pasos:

1. Proporcionar el modelo de datos que consiste en un archivo de texto que representa las preferencias de los usuarios a los objetos de aprendizaje, cada línea está expresada de la siguiente manera: *id_usuario, id_Categoria, peso.* El peso es la puntuación que me dará cada usuario a una categoría en particular.

2. Seleccionar el algoritmo de similaridad, Mahout implementa varios algoritmos como Correlación de Pearson, Medida Coseno, Coeficiente de Tanimoto, Log Likelihood, entre otros; En este caso para realizar la recomendación de utilizo el coeficiente de Tanimito[Apéndice B].

```
TanimotoCoefficientSimilarity sim = new TanimotoCoefficientSimilarity
(dm);
```

3. Establecer el tamaño del vecindario del usuario U1 para determinar los usuarios con gustos similares a U1 y de esta manera obtener mejores recomendaciones.

4. Inicializar el recomendador en el que se especifica el modelo de datos, el algoritmo de similaridad y el vecindario.

```
GenericItemBasedRecommender recommender = new
GenericItemBasedRecommender (dm, sim);
```

Finalmente los objetos recomendados se obtienen al indicar sobre cuál usuario se desea realizar la recomendación.

Item[1]: Cada uno de los elementos que forman parte de un dato

```
List<RecommendedItem>recommendations =
recommender.mostSimilarItems (itemId, 2);

System.out.println ("id:"+ itemId + "," + "recomendacion:" +
recommendation.getItemID () + "," + "valor:" + recommendation.getValue ());
```

Resultado: para cada id de usuario tenemos 2 recomendaciones [código completo en el Apéndice A]

```
id: 1489 ,  recomendacion: 1484 , valor: 1.0
id: 1489 ,  recomendacion: 390 , valor: 1.0
id: 1492 ,  recomendacion: 1484 , valor: 1.0
id: 1492 ,  recomendacion: 390 , valor: 1.0
id: 1493 ,  recomendacion: 1484 , valor: 1.0
id: 1493 ,  recomendacion: 390 , valor: 1.0
id: 1495 ,  recomendacion: 1162 , valor: 0.5
id: 1495 ,  recomendacion: 1151 , valor: 0.5
id: 1497 ,  recomendacion: 1484 , valor: 1.0
id: 1497 ,  recomendacion: 390 , valor: 1.0
id: 1498 ,  recomendacion: 1484 , valor: 1.0
id: 1498 ,  recomendacion: 390 , valor: 1.0
id: 1500 ,  recomendacion: 1484 , valor: 1.0
id: 1500 ,  recomendacion: 390 , valor: 1.0
id: 1503 ,  recomendacion: 704 , valor: 0.666(
id: 1503 ,  recomendacion: 1459 , valor: 0.5
id: 1508 ,  recomendacion: 398 , valor: 1.0
id: 1508 ,  recomendacion: 373 , valor: 1.0
id: 1518 ,  recomendacion: 1520 , valor: 1.0
id: 1518 ,  recomendacion: 1276 , valor: 1.0
id: 1520 ,  recomendacion: 1518 , valor: 1.0
id: 1520 ,  recomendacion: 1276 , valor: 1.0
id: 1529 ,  recomendacion: 643 , valor: 0.5
id: 1529 ,  recomendacion: 469 , valor: 0.125
```

Figura 17. Recomendaciones con mahout.

Este sistema produce recomendaciones para un usuario, teniendo en cuenta los temas que ha marcado como favoritos, y cuyos gustos son similares. Esto se ha parametrizado con las combinaciones de las interfaces Ítem Similarity y Ítem neighbourhood.

Existen varias implementaciones para esto con mahout como lo son: el coeficiente de Pearson, el coseno, etc.

El concepto de neighbourhood o vecindario determina cuantos ítems han de ser tomados en consideración para producir recomendaciones. Una vez que la distancia entre ítems está calculada, las recomendaciones pueden producirse tomando los n-ítem más cercanos dentro del umbral de similitud.

5.1.4 Inconvenientes.

A la hora de realizar esta versión nos encontramos con varios inconvenientes, por los cuales, esta implementación fue desechada, ya que no era la idónea para realizar la mejor implementación posible.

Los inconvenientes por los cuales fue desechada esta versión son:

- Al trabajar con los datos ya pasados a un fichero, se hace necesario la carga y actualización de un fichero cada vez que se inicie o exista un cambio en la aplicación.
- Los parámetros internos por los cuales el algoritmo de recomendación asigna pesos a los distintos ítems, no son accesibles, lo que supone que no se puede personalizar el algoritmo de recomendación.
- Trabajar con una aplicación standalone de Java, implica tener que crear clases por cada tabla que contenga la base de datos. Esto supone una complejidad enorme de manipulación de ficheros, además de un trato ineficiente de los datos.

Estas son las razones más contundentes por las cuales se ha desechado esta primera implementación. Por otro lado, con lo aprendido hasta el momento, se puede mejorar mucho la eficacia del sistema con nuevas y mejoradas más dinámicas y que permitan su manipulación en tiempo real, tal como si fuera un proyecto web basado en la tecnología AngularJs.

Item[1]: Cada uno de los elementos que forman parte de un dato

Capítulo 6.
Versión 2 de sistema recomendador con AngulasJS.

En el capítulo anterior hablamos lo que fue hacer el sistema recomendador con mahout con una interfaz de ventanas hecha en java para la visión de los usuarios, ahora veremos cómo montar el sistema recomendador utilizando como interfaz una aplicación web utilizando AngularJS.

6.1 Explicación de los programas a usar.

En esta sección se trabajara mediante el sistema Mean Full Stack, que significa desarrollar una aplicación con un único lenguaje de programación, en este caso, JavaScript.

Esta parte del trabajo se hará con un conjunto de programas que serán indispensables para la buena cohesión de toda la aplicación. Los programas a utilizar serán los siguientes:

- **MongoDB:** para la creación de la base de datos.

Def. Es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto.

MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

- **Express:** para la creación del framework server web.

Def. Es un framework web mínimo y flexible para Node.js que proporciona un conjunto robusto de características para aplicaciones web y

móviles, con una gran variedad de métodos de utilidad HTTP y middleware a su disposición, la creación de una potente API es rápida y fácil.

- **NodeJs:** Plataforma de servidor.

Def. Es una plataforma construida en tiempo de ejecución de JavaScript de Chrome para construir fácilmente aplicaciones rápidas de red escalables. Node.js utiliza una, el bloqueo no-modelo de E / S basada en eventos que lo hace ligero y eficiente, ideal para aplicaciones en tiempo real de datos intensivos que se ejecutan a través de dispositivos distribuidos.

- **AngularJs:** Framework de web cliente.

Def. Es un framework de JavaScript de código abierto, mantenido por Google, que ayuda con la gestión de lo que se conoce como aplicaciones de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript. Los valores de las variables de JavaScript se pueden configurar manualmente, o recuperados de los recursos JSON estáticas o dinámicas.

Maneja 3 clases de archivos: los ?.HTML para manejar los archivos web, los ?.Css para manejar los estilos de la web y los ?.Js para manejar las funciones script para manejar la lógica del sistema.

- **JavaScript:** Lenguaje de programación utilizado.

Def. Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS).

6.2 Desarrollo de la aplicación.

6.2.1 Creación estructura jerárquica:

Para el desarrollo de la aplicación se ha creado una aplicación mean, para trabajar la estructura de archivos en la cual se verá la lógica, los datos o modelos y las vistas de la aplicación.

Para iniciar la aplicación, después de haber instalado cada uno de los programas anteriormente descritos se ha de ejecutar la sentencia "yo angular", que dará lugar a la siguiente estructura.

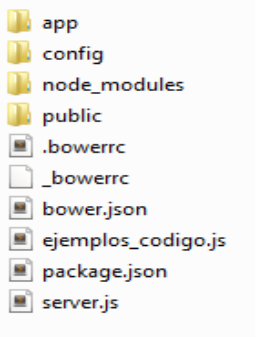


Figura 18. Jerarquía de archivos.

- **App:** carpeta donde está la lógica de la aplicación, están los controladores, modelos, vistas, y rutas de enlaces entre las carpetas.
- **Config:** carpeta de configuración donde estarán las dependencias express(npm) utilizadas.
- **Node modules:** carpeta de enlaces entre las aplicaciones Mean.
- **Public:** carpeta donde se carga toda la aplicación cliente.
- **Ficheros:** .json, .bowercc, .js para manejar los diferentes módulos de la aplicación.

6.2.2 Creación de la BBDD:

Es una base de datos NOSQL creada con mongoDB, para su instalación se descargó el ficherito de la página oficial mongoDB y se instaló, para dar inicio a la creación de está, se navegó mediante la consola hasta donde se alojaba y allí con la sentencia mongodb se dio inicio a la creación de las colecciones y documentos necesarios de la base de datos.

Las colecciones serán **usuarios** para el inicio de sección de los usuarios y **categorías** para registrar las categorías que se crean para realizar la recomendación.

6.2.3 Crear recomendador:

Para la creación del recomendador se pueden tener encuentra varias ayudas que nos da ya la tecnología, como son las librerías que me proporciona node.js (por ej. raccoon [Apéndice]), pero al ya haber utilizado una librería en la versión anterior mejor se ha decidido utilizar funciones en JavaScript para ir realizando las recomendaciones, emparejando las categorías creadas con los usuarios registrados y ponderando las categorías.

Las funciones se basaron en recoger los id's de las colecciones (usuario y categoría) y darle un peso promedio a cada categoría del usuario; si el usuario fuera de nuevo ingreso habrá una función que calculara nuevamente la repartición de pesos quedando equitativos.

Pseudocódigo pesos:

```
si (suma < umbralBajo || suma > umbralAlto)
    "error"
entons
    si (suma > 1)

        categoria -= suma -1

    si (suma < 1)

        categoria += (suma -1) * -1
```

Figura 19. Pseudocódigo de cálculo de pesos.

Pseudocódigo de recomendador:

```
comprobar que la puntuacion sea ~ 1

sumo pesos
  1/ # categorias -> media
  categoriaSubida += media

  categoria[i] = (media/(#categorias -1))

comprobar suma vale 1
    si no
      resto = 1 - suma
        para todas las categorias
            categoria += resto /#categorias
        fin para
    fin si
fin comprobar suma
```

Figura 20. Pseudocódigo cálculo recomendación.

Así se haría un cálculo promedio (ejemplo):

- **Media**= 1 / numCategoria => 1/4= 0,25.
- **Resta** = media / (numCategoria- 1) => 0,25/(4-0,25)= 0.06667.
- **Peso nuevaCategoria** = Categoría – resta = >0.25-0.06667 = 0.0833.

Usuario	Categoría	Peso de usuario de nuevo ingreso	Peso después seleccionar una categoría.
1	Peluquería	0.25	0.501
1	Moda	0.25	0.167
1	Bisutería	0.25	0.167
1	Joyería	0.25	0.167
2	Peluquería	0.40	0.65
2	Moda	0.35	0.2667
2	Bisutería	0.15	0.0667
2	Joyería	0.10	0.00167

Tabla 6. Explicación de asignación de pesos.

Este algoritmo lo que hace es dar pesos a las categorías que un usuario escoge y así hacerle la recomendación según lo que esa persona le guste en ese momento mediante la categoría escogida.

6.2.4 Creación de las vistas:

Se han de crear las vistas con las que interactuara el cliente, tanto la del login de usuarios, creación de categorías como la de realizar una recomendación.

Figuras	Imágenes
Figura 21. Vista de registro de usuario.	 el usuario se puede registrar
Figura 22. Vista de identificación de usuario.	 El usuario se identifica.
Figura 23. Acciones de un Administrador.	 Lo que el administrador

Item[1]: Cada uno de los elementos que forman parte de un dato

	puede hacer que sea la creación de las categorías.
Figura 24. Crear categorías	 El administrador crea categorías.
Figura 25. Listar Categorías.	 Todos los usuarios pueden ver las categorías que tiene el sistema.
Figura 26. Recomendación de categorías	 Consiste en listar las categorías que hay existentes y el usuario escogerá una de su preferencia y en base a eso se recomienda lo que este relacionado.

Tabla 7. Vistas de usuario.

Capítulo 7.
Conclusiones y líneas futuras.

El objetivo inicial de este trabajo era la creación de un sistema recomendador basado en contenido, en el que usuarios y profesores podrían tener una recomendación basada en sus perfiles, para lograr esto se han creado dos versiones una poco productiva(v1) y otra con mucho más alcance (v 2).

Como se explicó en el marco teórico para dar una mejor idea al inicio del trabajo se desarrolló un análisis de las técnicas de recomendaciones existentes, de las cuales se decidió por el filtrado colaborativo en el que según un relación usuario-ítem y su proximidad se realiza la recomendación, esto fue de mucha utilidad para poder seguir el desarrollo del mismo.

Se presentó una forma clásica de hacerlo mediante lenguaje java y un MVC muy estático que nos ayuda a hacer recomendaciones pero con muchas limitaciones a la hora de interactuar con grandes tamaños de datos, por esto se vio la necesidad de poder implantar una nueva versión con tecnologías mucho más acordes del gran avance que da cada día la informática.

Al implantar la nueva versión en lenguaje JavaScript siendo más dinámico, escalable y sobre todo mucho mejor a vista del usuario ya que es un sistema web dado que el anterior trabajaba sobre gestores de ventana y era más tedioso.

La creación de la aplicación ha sido muy satisfactorio, ya que se puede llegar a ver la complejidad y el gran alcance que se puede llegar a tener según las decisiones que tengamos, el equivocarse no siempre es malo, ya que de ello podemos aprender por esto hacer una primera aplicación no muy buena dio lugar a algo mucho mejor como lo fue trabajar con la tecnología MEAN una herramienta completamente nueva de la cual se puede aprender muchísimo.

Como líneas futuras se puede entrever que el recomendador al ser parte de un gran proyecto como lo es EDhospi (proyecto de aulas hospitalarias) se puede llegar a crear las recomendaciones más complejas y sofisticadas (ej. la publicaciones de videos, pdfs, recomendadores híbridos, etc.), también se puede implementar una opción que sea de que el usuario me diga que no le

gusta y para realizar la recomendación de lo que verdaderamente le puede interesar ya que las personas suelen saber que no les gusta de lo que le gusta.

Capítulo 8.
Summary and Conclusions.

The initial objective of this work was the creation of a content-based recommender system, where users and teachers could have a recommendation based on their profiles, to achieve this have created two versions (version 1) unproductive and one with more scope (version 2).

As he explained in the theoretical framework to give a better idea to start the paper analyzes existing techniques developed recommendations, of which the collaborative filtering is decided in which according to a user-item relationship and proximity it is performed developed recommendation, this was very useful to continue the development.

A classic way to do this using Java language and a static MVC that helps us to make recommendations but with many limitations when interacting with large data sizes, so was the need to introduce a new version with more chords breakthrough that gives everyday computing technologies.

By implementing the new version in JavaScript language being more dynamic, scalable and above all much better view of the user as it is a given that the previous worked on window managers and web system was tedious.

The creation of the implementation has been very satisfactory, since you can get to see the complexity and powerful that can have depending on the decisions that we, the mistake is not always bad, since it can learn why not make a very good first application resulted in much better as it was working with technology MEAN a completely new tool that you can learn a lot.

As future can glimpse the recommender to be part of a large project such as EDhospi (draft hospital wards) you can get to create the most complex and sophisticated recommendations (e.g. The publication of video, PDFs, hybrid recommenders, etc.), you can also implement an option that is for the user to tell me you do not like and to make a recommendation of what really would be interested because people usually know they do not like of what he likes.

Recommender show what you can earn if they continue to advance in new technologies, making it ever more flexible and accessible to all.

Item[1]: Cada uno de los elementos que forman parte de un dato

Capítulo 9.
Presupuesto.

Este capítulo veremos el total de lo que valdrá mi aplicación (valor estimado.)

9.1 Aplicaciones Informáticas:

Concepto	Nº Horas	Precio	Total
Análisis del proyecto	28h	15 €/h	420 €
Interfaz de usuario	94h	15 €/h	1410€
Sistema recomendador	94h	15 €/h	1410€
Integración con la BBDD	54h	15 €/h	810€

Total: 4050€

Tabla 8. Presupuesto Aplicaciones Informáticas.

9.1.1 Bienes, Equipos y servicios informáticos:

Concepto	Cantidad/unidades	Precio Unitario	Total
Ordenador Portátil	1	359 €	359 €
Acceso Internet	3 mes	39	117€
		Total:	476€

Tabla 9. Presupuesto de Bienes Informáticas.

Apéndice A. Algoritmos:

A.1. Algoritmo de un recomendador con Mahout.

Este algoritmo trabaja con un fichero de datos en los cuales estará el id de un usuario, el id de un ítem a recomendar y el peso dado a este ítem, y según la valoración de los ítem realiza una recomendación.

```
public class itemRecommender {

    public static void main(String[] args) {
        try {
            DataModel dm = new FileDataModel(new File("datos/categorias.csv"));

            TanimotoCoefficientSimilarity sim = new TanimotoCoefficientSimilarity(dm);

            GenericItemBasedRecommender recommender = new GenericItemBasedRecommender(dm, sim);

            int x=1;
            for(LongPrimitiveIterator items = dm.getItemIDs(); items.hasNext();) {
                long itemId = items.nextLong();
                List<RecommendedItem>recommendations = recommender.mostSimilarItems(itemId, 2);

                for(RecommendedItem recommendation : recommendations) {
                    System.out.println(" id: "+ itemId + " , " + " recomendacion: " + recommendation.getItemID() + " , " + "valor: " + recommenda
                }
                x++;
            }


        } catch (IOException e) {
            System.out.println("hay un error.");
            e.printStackTrace();
        } catch (TasteException e) {
            System.out.println("habia una excepcion");
            e.printStackTrace();
        }
```

Figura 27. Recomendador de Ítems.

DataModel me carga los datos, TanimitoCoefficienteSimilary me da los ítems que sean similares, GenericItemBasedRecommender me inicia el gestor de la recomendación, luego me calcula por cada usuario me calcule dos ítems similares.

Item[1]: Cada uno de los elementos que forman parte de un dato

Apéndice B.
Explicaciones.

B.1. ¿Porque Apache Mahout?

Apache Mahout es una biblioteca de aprendizaje de máquina con soporte para muchos algoritmos diferentes, incluyendo varios algoritmos de recomendación; tiene un amplio soporte para la computación distribuida, Apache Mahout proporciona cierta capacidad de configuración de sus algoritmos, la función de elemento de similitud puede ser reemplazado, por ejemplo, pero tiene relativamente pocos puntos de configuración; lo más cerca que podemos decir, la normalización de datos necesita ser construida en tanto al modelo de datos (por lo que el algoritmo ve normalizado de datos) o en cada uno propio componente algoritmo.

Mahout mide el rendimiento de un algoritmo de predicción o top-N métricas, pero proporcionan una línea de comandos o una interfaz de programación Java.

Con Mahout, el programador debe proporcionar constructores de recomendación que construyen recomendadores comprobables, esta librería fue escogida también por ser de programable en lenguaje java bajo un entorno de eclipse.

Feature	LensKit	Apache Mahout	MyMediaLite
Platform	Java	Java	C#/.NET
User-user CF	Yes	Yes	Yes
Item-item CF	Yes	Yes	Yes
Matrix factorization CF	FunkSVD	Yes	Many
Distributed algorithms	No	Yes	No
Visualization of configurations	Yes	No	No
Algo-independent lifecycle separation	Yes	No	No
Rating data support	Yes	Yes	Yes
Implicit feedback support	Partial	Yes	Yes
Distinct data normalizations	Yes	No	No
Offline evaluation	Yes	Yes	Yes
Reuses shared components in eval	Yes	No	No

Figura 28. Comparación con otros sistemas.

B.2. Métrica Loglikelihood (Librería de mahout):

Métrica Loglikelihood (logaritmo de verosimilitud) En estadística, la estimación por máxima verosimilitud (conocida también como EMV y, en ocasiones, MLE por sus siglas en inglés) es un método habitual para ajustar un modelo y encontrar sus parámetros.

El método de máxima verosimilitud fue introducido primero por R. A. Fisher, genetista y experto en estadística, en la década de 1920. La mayoría de los expertos en estadística recomiendan este método, al 20 menos cuando el tamaño maestral es grande, porque los estimadores resultantes tienen ciertas propiedades deseables de eficiencia.

• Sea X una variable aleatoria con función de probabilidad f (x| θ), donde θ es un parámetro desconocido. Sean X1,.., Xn los valores observados en una muestra aleatoria de tamaño n. La función de verosimilitud de la muestra es:

$$L(\theta) = \prod_{i=1}^{n} f(x_i|\theta) \quad \text{(Ec. 1)}$$

Debemos considerar que la ecuación 1 es la función de densidad con junta de la muestra aleatoria.
Notemos que la función de verosimilitud es una función del parámetro desconocido θ

.

• El estimador de máxima verosimilitud de θ es el valor de θ que maximiza la función de verosimilitud L (θ).

En ocasiones es más simple maximizar la función log-verosimilitud que la ecuación 1, dada por:

$$l(\theta) = \log\big(L(\theta)\big) = \sum_{i=1}^{n} \log f(x_i|\theta) \quad \text{(Ec. 2)}$$

El método de máxima verosimilitud puede emplearse en situaciones donde existen varios parámetros desconocidos, θ1, θ2,..., θk, que es necesario estimar. En tales casos, la función de verosimilitud es una función de los k parámetros desconocidos θ1,θ2,..., θk y los 21 estimadores de máxima

Item[1]: Cada uno de los elementos que forman parte de un dato

verosimilitud k se obtienen al igualar a cero las k derivadas parciales, dadas por:

$$\frac{\partial L(\theta_1, \theta_2, \ldots, \theta_k)}{\partial \theta_i}, i = 1,2, \ldots, k$$

Y resolver el sistema resultante.

B.3. Coeficiente de Tanimoto (Librería de Mahout):

Coeficiente de Tanimoto es la relación entre el tamaño de la intersección, o superposición entre los artículos preferidos de dos usuarios, para el total de los artículos preferidos de los usuarios. TanimotoCoefficientSimilarity es una implementación, derivado del coeficiente de Tanimoto. También se conoce como el coeficiente de Jaccard. También se puede definir como el número de elementos, para que los dos usuarios expresen alguna preferencia, dividido por el número de artículos, para los que cualquiera de los usuarios expresar alguna preferencia.

Sobre la base de la definición anterior, la pregunta que surge es, ¿qué es exactamente Tanimoto hace y cómo podría descubrir que estos dos usuarios son similares?

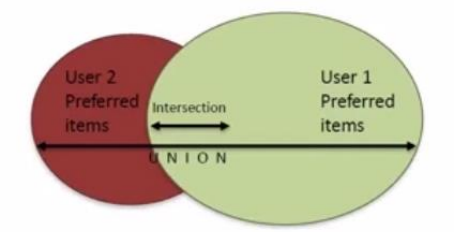


Figura 29. Diagrama de Tanimoto.

B.4. Raccoon (librería de nodeJS):

Es un motor de fácil uso basado en filtrado colaborativo es un módulo de npm de Node.js con servidor Redis. El motor utiliza el coeficiente de Jaccard para determinar la similitud entre los usuarios y k-vecinos más cercanos para crear recomendaciones. Este módulo es útil para cualquier persona con una base de datos de los usuarios, una base de datos de productos / películas / artículos y el deseo de dar a sus usuarios la posibilidad de recibir / disgusta y recibir recomendaciones sobre la base de usuarios similares. Raccoon se encarga de toda la lógica recomendación y calificación. Se puede combinar con cualquier base de datos, ya que no hace un seguimiento de la información de usuario / artículo, además de un identificador único.

Install Raccoon:

```
npm install raccoon
```

Install / Boot Redis:

```
npm install redis
redis-server
```

Require raccoon in your node server:

```
var raccoon = require('raccoon');
```

Connect raccoon to your redis instance:

```
raccoon.connect(port, url, auth);
// example of localhost:
// raccoon.connect(6379, '127.0.0.1');
// auth is optional, but required for remote redis instances
```

Add in ratings:

```
raccoon.liked('garyId', 'movieId');
raccoon.liked('garyId', 'movie2Id');
raccoon.liked('chrisId', 'movieId');
```

Ask for recommendations:

```
raccoon.recommendFor('chrisId', 10, function(results){
  // results will be an array of x ranked recommendations for
  // in this case it would contain movie2
```

Figura 30. Ejemplo de uso de librería raccoon.

Bibliografía

[1] Eclipse. https://eclipse.org/

[2] Java. https://www.java.com/es/download/

[3] Estudio de sistemas recomendadores: http://www.upf.edu/hipertextnet/numero-6/recomendacion.html

[4] http://www.upf.edu/hipertextnet/numero-2/recomendacion.html

[5] http://gaia.fdi.ucm.es/files/people/almudena/seminario/recsys-dia1.pdf

[6] http://www.gradiant.org/es/actualidad/noticias/727-sisitemas-recomendadores-en-la-web-20.html

[7] http://www.monografias.com/trabajos104/sistemas-recomendadores-recursos-informacion/sistemas-recomendadores-recursos-informacion.shtml

[8] NodeJS. https://nodejs.org/

[9] Mongo dB. https://www.mongodb.org/

[10] ExpressJs, http://expressjs.com/es/

[11] AngulasJs. https://angularjs.org/

[12] Librería mahout: http://www.ibm.com/developerworks/ssa/library/j-mahout-scaling/

[13] Librería Raccon: https://www.npmjs.com/package/raccoon

[14] Modelo-Vista-Controlador: http://www.desarrolloweb.com/articulos/que-es-mvc.html.