
SISTEMA DE RECOMENDACIÓN COLABORATIVO DE ALQUILER DE PELÍCULAS

Alumno: Fernando Siles Fernández

Tutores: Luis Martínez López
Luis Gonzaga Pérez Cordon

Departamento: Informática



Universidad de Jaén

CAPITULO 1. PREFACIO	1
1.1 Introducción al Proyecto.....	2
1.2 Propósito.....	4
1.3 Objetivos.....	4
CAPITULO 2. ANTECEDENTES	7
2.1 Sistemas de Recomendación.....	8
2.1.1 Tipos de Sistemas de Recomendación.....	9
1. Sistemas de Recomendación Basados en Contenido.....	9
2. Sistemas de Recomendación Colaborativos.....	9
3. Sistemas de Recomendación Basados en Conocimiento.....	10
4. Híbridos.....	10
2.1.2 Utilidad de los Sistemas de Recomendación.....	11
2.1.3 Realimentación en Sistemas de Recomendación.....	11
1. Realimentación implícita.....	12
2. Realimentación explícita.....	12
2.1.4 Elección del conjunto de datos para Sistemas de Recomendación.....	12
1. ¿Análisis online u offline?.....	12
2. ¿Datos reales o sintetizados?.....	13
2.1.5 Ejemplos reales de Sistemas de Recomendación.....	13
a) Amazon.....	14
b) IMDB Recommendation Center.....	15
c) Entreé Chicago.....	16
2.2 Sistemas de Recomendación Colaborativos.....	17
2.2.1 Introducción a los Sistemas de Recomendación Colaborativos.....	17
2.2.2 Categorías de los algoritmos de Filtrado Colaborativo.....	19
2.2.3 Ejemplos reales de Sistemas de Recomendación Colaborativos.....	20
a) Tapestry.....	20
b) zagat.com.....	21
c) filmaffinity.com.....	22
d) MovieLens.....	23
e) last.fm.....	24

CAPITULO 3. PROYECTO	26
3.1 Estudio Comparativo.....	28
3.1.1 Notación.....	28
3.1.2 Algoritmo K-nn.....	29
a) Funcionamiento del algoritmo K-nn.....	30
b) Descripción algorítmica.....	30
c) Principales características.....	30
3.1.3 Medidas de Similaridad.....	31
1. Coeficiente Coseno.....	32
2. Distancia Euclídea.....	33
3. Coeficiente de Correlación de Pearson.....	34
3.1.4 Algoritmos de Predicción Basados en Ítem.....	34
1. item average + adjustment.....	35
2. weighted sum.....	36
3.1.5 Diseño de las Pruebas.....	36
1. Especificaciones software y hardware.....	36
2. Conjunto de datos.....	37
3. Métricas de evaluación.....	41
3.1.6 Implementación de las Pruebas.....	43
3.1.7 Evaluación de Resultados.....	43
3.1.7.1 Resultados de las Pruebas.....	45
3.1.7.2 Comparativa entre las Pruebas.....	64
3.2 Sistema de Recomendación Colaborativo.....	66
3.2.1 Especificación de Requerimientos.....	67
3.2.1.1 Requerimientos funcionales.....	68
3.2.1.2 Requerimientos no funcionales.....	70
A) Requerimientos del equipo informático.....	71
B) Requerimientos de la Interfaz.....	72
3.2.2 Análisis del Sistema.....	73
3.2.2.1 Casos de uso.....	74
3.2.2.2 Escenarios.....	86

3.2.3 Diseño del Sistema.....	88
3.2.3.1 Diseño de los datos.....	88
1) Esquema Conceptual.....	91
2) Esquema Conceptual Modificado.....	92
3) Tablas de la aplicación.....	94
3.2.3.2 Diseño de la Interfaz.....	96
A) Definir Guía de Estilo.....	97
B) Metáforas.....	99
3.2.4 Implementación.....	101
3.2.4.1 Tipo de arquitectura de la aplicación.....	101
3.2.4.2 Lenguajes de programación utilizados.....	102
3.2.4.3 Herramienta de desarrollo utilizada.....	103
3.2.4.4 Actualización del algoritmo de filtrado.....	104
3.2.4.5 Instalación del servidor y funcionamiento de moviesrecommender.....	105
CAPITULO 4. CONCLUSIONES	107
ANEXO I. MANUAL DE INSTALACIÓN DEL SERVIDOR	111
ANEXO II. MANUAL DE USUARIO	133
ANEXO III. CÓDIGO FUENTE DEL MÓDULO DE PRUEBAS	151
ANEXO IV. CÓDIGO FUENTE DEL MÓDULO DE FORMATEO DE LA BASE DE DATOS	171
BIBLIOGRAFÍA	181

CAPITULO 1.

PREFACIO.

1.1 Introducción al proyecto

El número de empresas que ofrecen sus productos o servicios en Internet se multiplica cada año. Este hecho indiscutible ha hecho que se produzca una tremenda competencia cada vez más encarnizada para asegurar su supervivencia. Debido a esta competencia, las empresas deben ofertar una serie de servicios diferenciadores que les permitan no sólo mantener su clientela sino atraer nuevos clientes desde los competidores.

Una de las estrategias más interesantes para conseguir ésto, es la de desarrollar servicios de marketing personalizado basados en **sistemas de recomendación**. Estos sistemas se encargan de suministrar a los usuarios información personalizada y diferenciada sobre determinados productos y/o servicios que pueden ser de interés para ellos. Es decir, se encargan de guiar a un usuario mediante recomendaciones en la búsqueda de aquellos servicios o productos que puedan ser más atractivos para él, modificando el proceso de navegación y búsqueda. Esto es sin duda una gran ventaja para los clientes, que encontrarán lo que necesitan de una forma más rápida, cómoda y fácil dentro de las enormes bases de datos que ofertan las tiendas electrónicas en Internet y además descubrirán nuevos productos o servicios que le puedan ser atractivos, que de otra manera les hubiese sido mucho más difícil y complejo encontrar.

Existen muy diversos tipos de sistemas de recomendación, siendo los dos más importantes y utilizados los siguientes:

1. Sistemas de recomendación basados en contenido

Se basan en la similitud entre objetos, es decir, predicen que para un usuario serán de interés aquellos objetos muy parecidos en su contenido con aquellos que ya sabemos que son de su agrado.

2. Sistemas de recomendación colaborativos

Son más cercanos a la forma de pensar de los seres humanos que los basados en contenido. Son aquellos en los que las recomendaciones se realizan basándose solamente en los términos de similitud entre los usuarios [1].

El objeto de este proyecto es el estudio y desarrollo de un **Sistema de Recomendación Colaborativo**. Se podría decir que el funcionamiento de este tipo de sistemas de

recomendación tiene los siguientes pasos [13]:

1. El sistema guarda un perfil de cada usuario con evaluaciones sobre objetos conocidos por él y que pertenezcan a la base de datos sobre la que se trabaje.
2. Se mide el grado de similitud entre los diferentes usuarios del sistema en base a sus perfiles y se crean grupos de usuarios con características afines.
3. El sistema utilizará toda la información obtenida en los pasos anteriores para realizar las recomendaciones. A cada usuario le recomendará objetos que no haya evaluado y que hayan sido evaluados de manera positiva por los miembros del grupo en el que este incluido.

Por lo tanto, este sistema no toma en consideración el contenido y las características de los productos que se recomiendan sino que sean del gusto de usuarios con gustos semejantes al usuario que solicita la recomendación.

Una vez comentados brevemente la situación actual del mercado de Internet y las características de los novedosos y útiles sistemas de recomendación podemos pasar a introducir brevemente en qué consiste el proyecto que se documenta en esta memoria.

En este proyecto se pretende realizar un sistema de recomendación con filtrado colaborativo que se aplicará al ámbito de la recomendación en tiendas de alquiler o venta de películas o videoclubs. Para ello utilizaremos los datos que proporciona la web <http://www.grouplens.org>, que se han utilizado de base para realizar estudios de investigación en el ámbito de los sistemas de recomendación colaborativos.

Estos datos están en formato texto por lo que nuestro primer paso será formatearlos hasta convertirlos a un formato que nos sea de utilidad. Nosotros hemos elegido un formato de base de datos **Microsoft Access** aunque hubiera resultado igualmente válido para nuestro propósito cualquier otro formato de base de datos manejable mediante sentencias **SQL** como MySQL o similares.

Una vez terminado el formateo de nuestra base de datos pasamos a realizar una aplicación **Java** (utilizando la herramienta **NetBeans 5.0**) en la que evaluamos y comparamos distintos tipos algoritmos de filtrado colaborativo basados en ítem para observar cuál o cuales

de las propuestas existentes en la literatura nos proporcionarán mejores resultados para el sistema que pretendemos realizar.

Una vez realizado este estudio comparativo elegiremos el mejor de estos algoritmos y lo utilizaremos para implementar un sistema de recomendación con una arquitectura cliente/servidor y con una interfaz web que lleva por nombre **moviesrecommender** y de la que se propondrá una versión prototipal o beta.

Por último se expondrán una serie de conclusiones tanto sobre los sistemas de recomendación en general como sobre el desarrollado para este proyecto y adjuntaremos diversos manuales y cualquier descripción que consideramos útil para el correcto entendimiento y funcionamiento de los modelos y aplicaciones desarrolladas en este proyecto. Finalmente, para aquellos lectores que estén interesados en el tema, se incluirá una amplia bibliografía.

1.2 Propósito

Hacer un estudio comparativo de algoritmos de filtrado colaborativo y sus parámetros de configuración. De esta manera, atendiendo a los resultados obtenidos de dicho estudio implementar un sistema de recomendación colaborativo basado en técnicas de inteligencia artificial para la gestión del alquiler de películas, que utilice el algoritmo que muestre un mejor comportamiento.

1.3 Objetivos

1. Búsqueda y revisión bibliográfica.
2. Formatear la base de datos de películas disponible en <http://www.grouplens.org> para poder desarrollar correctamente los pasos posteriores.
3. Evaluación y comparación de uno o varios algoritmos de filtrado colaborativo para encontrar grupos de usuarios con gustos similares.

4. Desarrollo de un proceso de extracción de conocimiento que permita predecir las valoraciones de un usuario con arreglo a las hechas por los miembros más similares al mismo.
5. Estudio de la efectividad de los distintos algoritmos de filtrado colaborativo y de los modelos de predicción mediante varios procesos de *hold out*.
6. Implementación de un sistema de recomendación basado en los resultados obtenidos en los puntos anteriores con una arquitectura cliente/servidor y una interfaz web amigable que permita al usuario interactuar con facilidad con el sistema.

CAPITULO 2.

ANTECEDENTES.

2.1 Sistemas de Recomendación

El ser humano necesita información para tomar decisiones de cualquier tipo pero muchas veces se encuentra con que la información que tiene disponible es demasiado amplia o inconexa, tenemos una sobrecarga de información y no somos capaces de extraer la que es verdaderamente relevante. Es necesario filtrar esta información relevante diseminada en grandes volúmenes de información y es en esta situación donde entran con fuerza al rescate del usuario los sistemas de recomendación.

Una definición formal de sistema de recomendación es que se trata de aquel sistema que tiene como principal tarea seleccionar ciertos objetos de acuerdo a los requerimientos del usuario [7].

Otra definición de un sistema de recomendación podría ser la del sistema que utiliza las opiniones de los usuarios de una comunidad para ayudar a usuarios de esa comunidad a encontrar contenidos de su gusto entre un conjunto sobrecargado de posibles elecciones [7].

Pero los sistemas de recomendación no se basan en ningún modelo novedoso sino que lo hacen en un acto que existe desde que el ser humano tiene consciencia e inteligencia: pedir consejo o recomendación a expertos en la materia o seguir a aquellos individuos que tienen gustos similares al del usuario o bien seleccionar objetos que tienen características similares a objetos que le hayan gustado anteriormente o que se parecen al que inicialmente buscaba. Estas dos tendencias milenarias han dado lugar a las dos ramas principales (aunque no las únicas) de los sistemas de recomendación: los colaborativos y los basados en contenido, respectivamente.

Tanto los sistemas de recomendación colaborativos como los basados en contenido necesitan una enorme cantidad de información sobre usuarios y objetos para poder realizar unas recomendaciones de calidad [6]. Debido a esta circunstancia han surgido otros tipos de sistemas de recomendación que pueden trabajar y ofrecer recomendaciones de calidad sin necesitar una cantidad de información tan grande: los sistemas de recomendación basados en conocimiento. Además, en los últimos tiempos se han desarrollado sistemas de recomendación híbridos, los cuales recogen los mejores aspectos de dos o más de los sistemas de recomendación ya nombrados para conseguir unos resultados todavía mejores a la hora de realizar sus recomendaciones.

2.1.1 Tipos de Sistemas de Recomendación

Los sistemas de recomendación se clasifican atendiendo a su funcionamiento, dando lugar a varios tipos de sistemas de recomendación:

1. Sistemas de Recomendación Basados en Contenido

Un sistema de recomendación basado en contenido es aquel en el cual 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 un pasado [1].

En otras palabras: extraen características de los objetos y las comparan con el perfil del usuario para predecir las preferencias de los usuarios sobre tales objetos. Lo que se pretende es recomendar objetos muy similares en su contenido a objetos que ya sabemos que son del agrado del usuario en cuestión, o sea, los que forman parte de su perfil.

Inicialmente, el filtrado basado en contenido era el más extendido de los sistemas de recomendación pero tiene un claro y, en muchos casos, grave problema como es la **sobre-especialización**. Esta sobre-especialización se da al reducir las recomendaciones a unos contenidos muy similares sin tener en cuenta la posible arbitrariedad de los gustos e intereses de los usuarios. Otro problema es que de un objeto solo se puede conocer una información parcial, normalmente textual, mientras que la información contextual, visual o semántica es más difícil de conocer y por lo tanto se pierden conexiones entre objetos similares de manera menos obvia.

Se han intentado múltiples soluciones para estos problemas como la incorporación de una cierta aleatoriedad a las búsquedas, la indexación semántica latente (LSI) de la información textual [5] o las medidas de similitud basadas en ontologías pero sin duda la mejor solución a tales problemas es que exista una buena retro-alimentación entre el sistema y sus usuarios.

2. Sistemas de Recomendación Colaborativos

Los sistemas de recomendación basados en un filtrado colaborativo son aquellos en los que las recomendaciones se realizan basándose solamente en los términos de similitud

entre los usuarios [1]. Es decir, los sistemas colaborativos recomiendan objetos que son del gusto de otros usuarios de intereses similares.

Para la realización de un buen sistema de recomendación colaborativo (es decir, un sistema que ofrezca recomendaciones de calidad) es necesario utilizar un buen algoritmo de filtrado colaborativo. Estos algoritmos se pueden encuadrar dentro de dos categorías: los algoritmos basados en memoria o usuario y los basados en modelos o ítem.

Conforme la utilización de estos sistemas de recomendación se ha ido popularizando se han ido descubriendo una serie de problemas como son la escasez, la escalabilidad y el problema del ítem nuevo [4]. Multitud de estudios y experimentos se han llevado a cabo en los últimos tiempos con la intención de minimizar estos problemas.

Este tipo de sistemas de recomendación son el eje sobre el que gira este proyecto por lo que se merecen un estudio más detallado en un epígrafe posterior.

3. Sistemas de Recomendación Basados en Conocimiento

Los sistemas de recomendación basados en conocimiento realizan inferencia entre las necesidades y preferencias de cada usuario para sugerir recomendaciones [6].

A diferencia de otros sistemas de recomendación, los basados en conocimiento no dependen de grandes cantidades de información sobre objetos puntuados (basados en contenido) y usuarios particulares (colaborativos) sino que lo único que necesitan es tener un conocimiento general sobre el conjunto de objetos y un conocimiento informal de las necesidades del usuario.

El principal problema de estos sistemas de recomendación es que aunque no requieren mucha información si que requieren un gran esfuerzo humano para realizar las recomendaciones mediante todo tipo de heurísticas de inferencia.

4. Híbridos

Todos los sistemas de recomendación vistos hasta ahora tienen sus puntos fuertes y sus talones de Aquiles por lo que es lógico pensar en intentar maximizar sus bonanzas y minimizar sus puntos débiles mediante la hibridación de dos o más de ellos.

Los sistemas híbridos entre los basados en contenido y los colaborativos guardan las preferencias del usuario y las combinan con los objetos más relevantes para realizar las recomendaciones [6].

También existen los sistemas híbridos entre los basados en conocimiento y los colaborativos, los basados en contenido y los basados en conocimiento e incluso entre los colaborativos y las redes sociales.

2.1.2 Utilidad de los Sistemas de Recomendación

Los sistemas de recomendación resultan de vital importancia para el marketing personalizado ya que reducen el tiempo de búsqueda de los productos, consiguen una mayor efectividad en las búsquedas y, por lo tanto, una mayor satisfacción en los clientes.

Para lograr estos objetivos, todos los sistemas de recomendación llevan a cabo dos tareas [8]:

- **Predecir:** los sistemas de recomendación predicen una serie de objetos, servicios o productos en los que un usuario o cliente particular podría estar interesado.
- **Recomendar los N-mejores objetos:** los sistemas de recomendación identifican los N objetos en los que el usuario estará más interesado.

2.1.3 Realimentación en Sistemas de Recomendación

Un sistema de recomendación no debe ser una entidad estática sino evolucionar en el tiempo en cuanto a la calidad de sus recomendaciones y pronósticos en base a la experiencia y nueva información adquiridas. Para conseguir este objetivo se utilizan mecanismos de realimentación entre el sistema y los gustos de los usuarios. Existen dos tipos de mecanismos de realimentación: los implícitos y los explícitos.

1. Realimentación implícita

Un mecanismo de realimentación implícito es aquel que proporciona información al sistema de recomendación acerca de los gustos de los usuarios sin que éstos sean conscientes de esta situación. Por lo tanto este tipo de realimentaciones no son directas sino que se realizan mediante diversos tipos de medidas como pueden ser el tiempo de visualización del objeto, el número de veces que el objeto es solicitado, etc.

Esta realimentación implícita tiene el problema de depender en demasía del contexto y de ser excesivamente hipotética (podemos suponer que solicitar la visualización de un objeto muchas veces indica un especial interés por parte del usuario pero no tiene porque ser de esa manera) por lo que no resulta ser la más apropiada para todas las situaciones de recomendación.

2. Realimentación explícita

Un mecanismo de realimentación explícito es aquel basado en la acción directa por parte del usuario para indicar que objetos determinados del sistema son de su interés. Esta interacción directa se puede realizar mediante votaciones numéricas o, más sencillo aún, que el usuario diga si el objeto es o no de su agrado.

Este mecanismo tampoco se encuentra exento de problemas como pueden ser la voluntariedad del cliente o el tiempo consumido.

2.1.4 Elección del conjunto de datos para Sistemas de Recomendación

La elección de un adecuado conjunto de datos es algo primordial para evaluar la calidad de un sistema de recomendación. Para que esta elección resulte la adecuada hay que contestar a las siguientes dos preguntas [7]:

1. ¿Análisis online u offline?

Es importante decidir como se va a trabajar sobre los datos: si de manera online u offline. En el análisis offline se usa una técnica o algoritmo para predecir ciertos valores retenidos de un conjunto de datos y los resultados son analizados mediante una o varias

métricas de error. Este análisis offline tiene la ventaja de ser rápido y económico pero también tiene dos desventajas importantes: el problema de la escasez de datos y el problema de obtener sólo como resultado la bondad de la predicción.

Por contra, el análisis online permite obtener otros resultados como son la actuación de los usuarios participantes, su satisfacción o su participación. En su defecto es más lento y caro que el análisis offline.

2. ¿Datos reales o sintetizados?

Otra elección importante es elegir entre un conjunto de datos reales (recopilados de usuarios reales sobre objetos reales) o un conjunto de datos sintetizados (creados específicamente para el sistema de recomendación, sin ninguna base real). Los datos sintéticos son más fáciles de crear que los reales ya que no hay que realizar encuestas ni otros métodos para conseguirlos del mundo real pero sin embargo es recomendable usar estos últimos y sólo utilizar los sintetizados en las primeras fases de desarrollo del sistema, siendo sustituidos por los reales cuando haya un número importante de éstos recopilados.

2.1.5 Ejemplos reales de Sistemas de Recomendación

En este epígrafe vamos a hacer un repaso a distintos sistemas de recomendación (salvo los colaborativos que se verán en su propio epígrafe) que se pueden encontrar en el mercado actualmente.

a) Amazon (<http://www.amazon.com>)



Figura 2.1. Página personalizada en Amazon

La poderosa empresa norteamericana de comercio electrónico que empezó siendo una librería online es un ejemplo paradigmático de sistema de recomendación que **mezcla los enfoques basado en contenido y colaborativo**. El sistema guarda las preferencias del usuario activo y las combina con objetos relevantes para generar recomendaciones (las ya celebres páginas “*People who bought this item... also bought these items...*” como la de la figura que acompaña este párrafo).

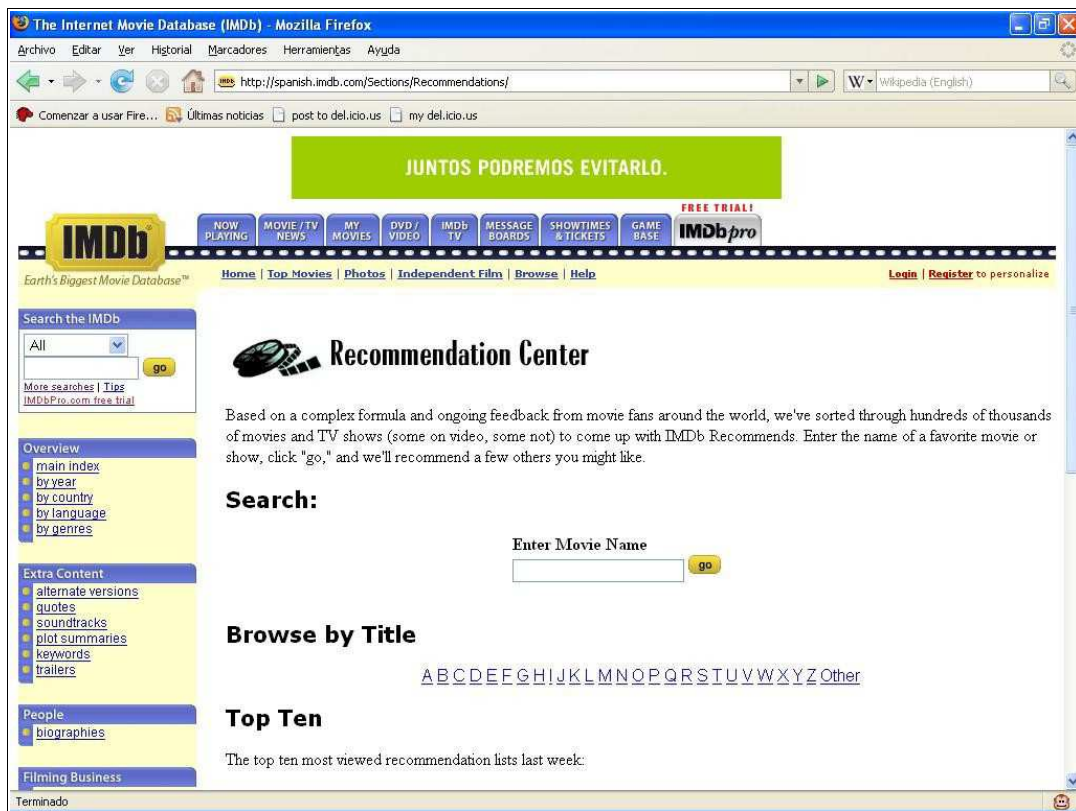
b) IMDB Recommendation Center (<http://spanish.imdb.com/Sections/Recommendations/>)

Figura 2.2. Interfaz del Centro de Recomendación de IMDB

La mayor base de datos de cine y televisión del mundo y uno de los sitios web más populares y visitados de todo Internet también ofrece a sus usuarios un sistema de recomendación: se llama **Recommendation Center** y esta **basado en contenido**. El usuario introduce la película o show televisivo que más le guste y el sistema le ofrece una lista con diez recomendaciones. Como método de realimentación o feedback con el sistema, el usuario puede señalar las recomendaciones con las que no este de acuerdo y proponer recomendaciones nuevas con lo que el algoritmo se va depurando con la interacción del usuario. De todas formas no es el servicio más exitoso que ofrece **IMDB** y eso se debe principalmente a que no es un sistema de recomendación especialmente bueno y acertado.

c) Entrée Chicago

Entrée Chicago (Guo; 2006) es un sistema de recomendación basado en conocimiento desarrollado por el Intelligent Information Laboratory (Infolab) de la Universidad de Chicago que ayuda al usuario a decidir entre más de 700 restaurantes de la ciudad de Chicago según sus restaurantes preferidos de otras ciudades norteamericanas. Además de ofrecer recomendaciones dispone de reviews de todos los restaurantes y de mapas para llegar a ellos.

2.2 Sistemas de Recomendación Colaborativos

Debido a que en este proyecto nos proponemos hacer un estudio de ciertos algoritmos de filtrado colaborativo para implementar posteriormente un sistema de recomendación colaborativo vamos a hacer una revisión más profunda y detallada sobre este tipo de sistemas de recomendación.

2.2.1 Introducción a los Sistemas de Recomendación Colaborativos

Un sistema de recomendación colaborativo es aquel en el que las recomendaciones se realizan basándose solamente en los términos de similitud entre los usuarios [1]. Es decir, los sistemas colaborativos recomiendan objetos que son del gusto de otros usuarios de intereses similares en vez de recomendar objetos similares a los que le gustaban en un pasado al usuario activo como sucedía con los basados en contenido. Se podría decir que este tipo de sistemas de recomendación se basan en el concepto del *boca a boca* entre sus usuarios para realizar sus recomendaciones.

La base teórica de los sistemas de recomendación colaborativos es bastante sencilla: se forman grupos de usuarios más cercanos, es decir, aquellos cuyos perfiles son más parecidos y a un usuario de un grupo se le recomiendan objetos que él no tenga puntuados pero que tengan unas puntuaciones positivas por parte del resto de usuarios de ese grupo, es decir, de los más similares a él.

Shardaham y Maes [13] distinguieron tres pasos fundamentales en el funcionamiento de los sistemas de recomendación colaborativos:

1. El sistema guarda un perfil de cada usuario con sus evaluaciones sobre objetos conocidos por él y que pertenecen a la base de datos sobre la que se trabaje.
2. Se mide el grado de similitud entre los diferentes usuarios del sistema en base a sus perfiles y se crean grupos de usuarios con características afines.
3. El sistema utiliza toda la información obtenida en los dos pasos anteriores para realizar las recomendaciones. A cada usuario le recomendará objetos que no haya evaluado

previamente y que hayan sido evaluados de manera positiva por los miembros del grupo en el que esté incluido.

Al igual que los basados en contenido, los sistemas de recomendación colaborativos tienen una serie de problemas que se han ido descubriendo conforme se han hecho más populares y más utilizados, como son los problemas de: la escasez, la escalabilidad y del ítem nuevo [4]. Comentemos las principales características de cada uno de estos problemas:

- **Escasez**

Los sistemas de recomendación colaborativos necesitan de una gran cantidad de datos, muchos usuarios puntuando muchos ítems similares para así poder calcular los grupos de vecinos y, en base a ellos, realizar las recomendaciones. Si en nuestra base de datos tenemos pocos usuarios o pocas puntuaciones por parte de cada usuario, nuestra matriz de puntuaciones será muy escasa y los cálculos de vecindad, predicción y recomendación no pueden ser realizados con la suficiente seguridad y exactitud obteniendo por lo tanto unas recomendaciones de baja calidad.

- **Escalabilidad**

Los sistemas de recomendación colaborativos usan por norma general algoritmos de cálculo de los k vecinos más cercanos (knn, K-nearest neighbors) para obtener la similaridad entre usuarios. Estos algoritmos son costosos computacionalmente y su coste crece linealmente cuanto mayor sea el número de usuarios y de ítems por lo que con bases de datos con millones de elementos, al aumentar el número de datos, el sistema sufrirá graves problemas de escalabilidad.

- **Problema del ítem nuevo**

En los sistemas de recomendación colaborativos los ítems nuevos, que tienen muy pocas o, incluso, ninguna puntuación no van a ser recomendados prácticamente nunca. De la misma forma, los nuevos usuarios en el sistema recibirán muy pobres predicciones debido a que ellos han puntuado muy pocos ítems y se hace difícil encuadrarlos en algún grupo de vecinos. Estos dos hechos nos hacen ver que estos

sistemas de recomendación requieren un cierto tiempo antes de empezar a hacer predicciones y recomendaciones ciertamente relevantes y acertadas.

Una gran cantidad de experimentos, estudios e investigaciones se han realizado en los últimos años para encontrar técnicas que reduzcan el impacto de estos problemas en los sistemas de recomendación con filtrado colaborativo.

Para reducir el problema de la escasez se han intentado la utilización de puntuaciones implícitas [10], la correlación entre ítems [12] y el filtrado híbrido. Mientras que para tratar de mejorar la escalabilidad se han propuesto la reducción de la dimensionalidad [11] y aproximaciones basadas en modelos. Finalmente, estudios han demostrado que las técnicas de web mining como los árboles de decisión son útiles para paliar el problema de los ítems y usuarios nuevos [3].

2.2.2 Categorías de los algoritmos de Filtrado Colaborativo

Para obtener un sistema de recomendación colaborativo de calidad es necesario elegir un buen algoritmo de filtrado colaborativo. Estos algoritmos pueden dividirse dentro de dos categorías:

- **Algoritmos basados en memoria o basados en usuario**

Estos algoritmos utilizan la base de datos completa para generar una predicción. El funcionamiento de estos algoritmos es el siguiente: se utilizan técnicas estadísticas para encontrar un conjunto de vecinos al usuario activo y posteriormente se utilizan una serie de algoritmos que combinan las preferencias de esta vecindad para realizar las predicciones y recomendaciones.

Estos algoritmos basados en usuario son muy populares y exitosos en la práctica pero son también los que con más ferocidad sufren los problemas de escasez y escalabilidad vistos anteriormente por lo que se hizo necesaria la aparición de otro tipo de algoritmos como los que vienen a continuación.

- **Algoritmos basados en modelos o basados en ítem**

Estos algoritmos proporcionan recomendaciones de ítems desarrollando primero un modelo (ya sea mediante redes bayesianas, clustering o modelos basados en reglas) de las puntuaciones de los usuarios sobre los ítems.

No se utilizan técnicas estadísticas sino una aproximación probabilística que calcula el valor esperado de una predicción del usuario dados sus puntuaciones sobre otros ítems. Es decir, estos algoritmos miran en el conjunto de ítems que el usuario activo ha puntuado o evaluado y calcula como de similar son estas puntuaciones con respecto al ítem activo con el fin de realizar una predicción para el mismo.

Los sistemas de recomendación colaborativos basados en modelo o ítem son los que centrarán el proyecto que se detalla en esta memoria por lo que veremos con más profundidad estos algoritmos y sus principales aspectos cuando entremos a analizar en profundidad el proyecto realizado.

2.2.3 Ejemplos reales de Sistemas de Recomendación Colaborativos

En este epígrafe vamos a hacer un repaso a distintos sistemas de recomendación colaborativos que se pueden encontrar en el mercado actualmente o que han sido de vital importancia para el desarrollo, arraigo y auge de los mismos.

a) Tapestry

El pionero. Tapestry, un proyecto de Xerox PARC, está considerado como el primer sistema de recomendación que implementaba filtrado colaborativo. Tapestry permitía a sus usuarios encontrar documentos basados en comentarios hechos previamente por otros usuarios. Al ser un experimento pionero surgieron muchos problemas ya que sólo funcionaba correctamente con pequeños grupos de personas y eran necesarias consultas de palabras específicas para obtener resultados lo que dificultaba en gran medida el propósito último del filtrado colaborativo. También tenía otras carencias como la falta de privacidad. A pesar de todo fue un sistema que resultó crucial para el posterior fulgurante crecimiento de los sistemas de recomendación colaborativos.

b) zagat.com (<http://www.zagat.com>)

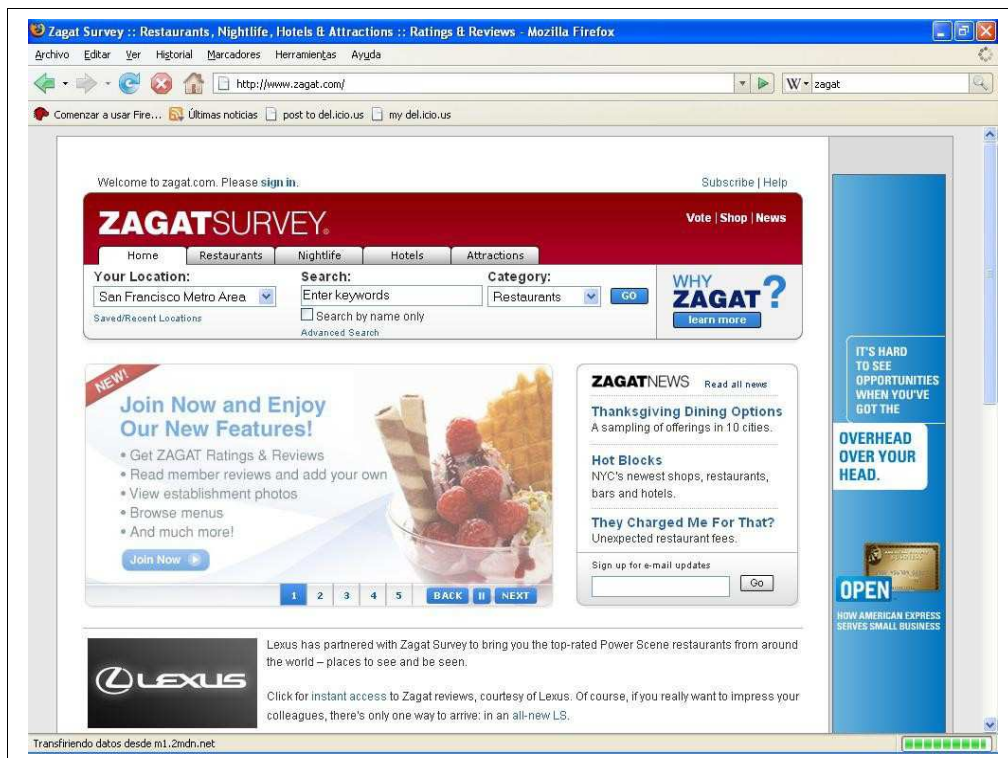


Figura 2.3. Interfaz de zagat.com

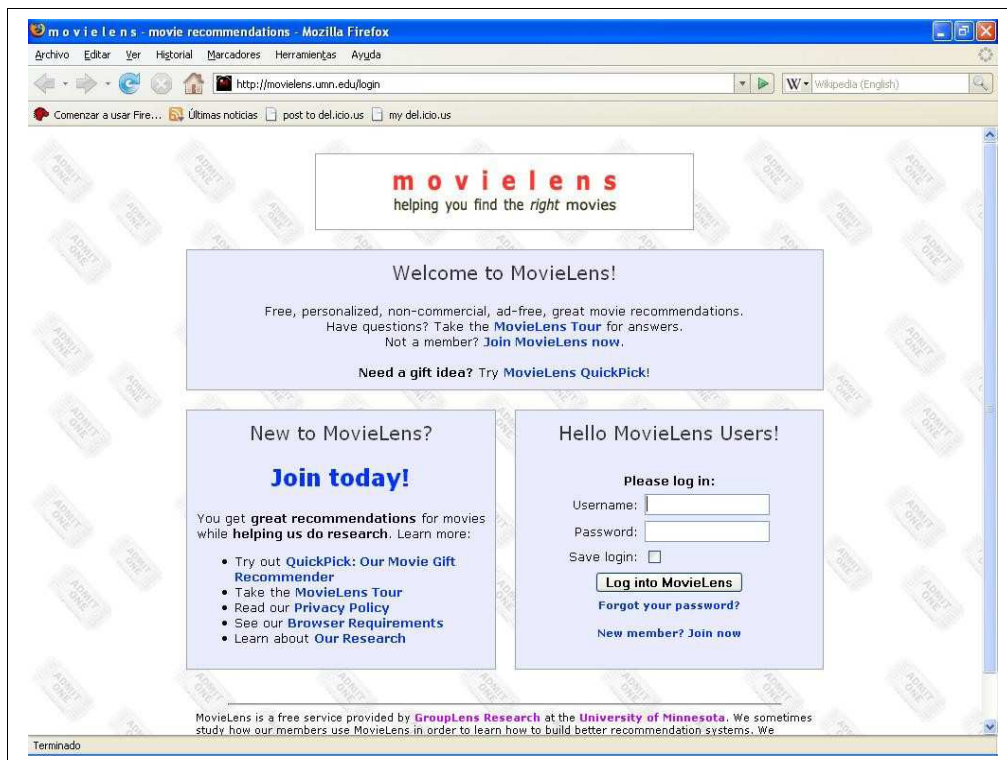
Zagat Survey es una empresa americana fundada en 1979 que se dedica a la edición de todo tipo de guías de restaurantes, hoteles, clubes o tiendas de distintas ciudades de los Estados Unidos y Canadá. En zagat.com los usuarios registrados pueden votar distintos aspectos (hasta 30) del local referido y, además, introducir pequeños comentarios con su experiencia. En base a estas votaciones los responsables de la empresa asignan sus puntuaciones en sus guías anuales y hacen recomendaciones individuales a sus usuarios a través de su web.

c) **filmaffinity.com** (<http://www.filmaffinity.com>)



Figura 2.4. Página principal de *filmaffinity.com*

Filmaffinity es un proyecto español de gran proyección internacional que desde 2002 se encarga de recibir puntuaciones de todo tipo de películas y dar recomendaciones a sus usuarios. Su funcionamiento es sencillo: el nuevo usuario puede empezar a votar las películas que haya visto con la ayuda de unos tours dirigidos que atemperan de manera inteligente los problemas de escasez y nuevo ítem; posteriormente el sistema calcula las almas gemelas de este nuevo usuario y le recomienda las películas favoritas de estas almas gemelas que no haya votado el usuario. También ofrece la posibilidad de puntuar series de televisión, de ver la información de tus almas gemelas y contactar con ellas, realizar críticas y comentarios sobre las películas y series vistas y consultar los diferentes tops por géneros o décadas o generales.

d) MovieLens (<http://movielens.umn.edu>)Figura 2.5. *Interfaz de MovieLens*

MovieLens es un sistema de recomendación de películas online basado en filtrado colaborativo. Desarrollado por el GroupLens Research de la **Universidad de Minnesota** (<http://www.grouplens.org>) recolecta puntuaciones sobre películas de sus usuarios y en base a esos datos agrupa los usuarios de similares gustos. Atendiendo a las puntuaciones de todos los usuarios dentro de un grupo se intenta predecir para cada usuario individual su opinión sobre películas que todavía no ha visto.

En un principio utilizaba algoritmos basados en usuario para realizar sus predicciones y recomendaciones pero desde hace un tiempo emplea algoritmos basados en ítem porque dan unos mejores resultados.

Los datos sobre sus usuarios y sus puntuaciones son privados pero los investigadores de GroupLens mantienen como públicas dos ejemplos de 100000 y un millón de puntuaciones respectivamente. Estos ejemplos se pueden descargar desde la propia página web (<http://www.grouplens.org>) siendo el de 100000 puntuaciones el usado en este proyecto como conjunto de datos de partida.

e) last.fm (<http://www.last.fm>)



Figura 2.6. *Página de recomendaciones de last.fm*

La popularidad adquirida en los últimos tiempos por los sistemas de recomendación colaborativos ha propiciado que se haya ampliado el rango de aplicación de los mismos. Una nueva generación de sistemas de recomendación colaborativos ha surgido en el ámbito de la radio musical vía Internet.

last.fm es un sistema de recomendación musical además de una red social y un radio vía Internet. Cada nuevo usuario va creando su propio perfil de manera muy sencilla: el usuario puede escuchar las radios personalizadas (canciones favoritas) del resto de usuarios y decidir si les gustan (pasan a formar parte del perfil del propio usuario) o si por el contrario no quiere volverlas a escuchar. Con este perfil y gracias a un algoritmo de filtrado colaborativo el sistema va cerrando el cerco sobre los usuarios con gustos más afines con el usuario activo (los denominados vecinos) y recomendando grupos y artistas que no se encuentran en su perfil pero que si son favoritos dentro de su vecinos. Estas acciones son continuas por lo que la calidad de las recomendaciones se refina de manera ciertamente importante cuando se es un usuario veterano dentro la aplicación.

Sin duda, last.fm y otras aplicaciones de similares características, están siendo una de las grandes revoluciones de los últimos tiempos en Internet gracias a su marcado carácter social, a su amplio catálogo musical (last.fm contaba con más de 100.000 canciones a noviembre de 2006) y a la calidad de sus recomendaciones.

Otros sistemas de recomendación colaborativos (tanto de ámbito comercial como no comercial) que pueden ser de interés para que el lector compruebe el funcionamiento y utilidad de los mismos son los siguientes: Pandora (<http://www.pandora.com>), Live365 (<http://www.live365.com>), TiVo (<http://www.tivo.com>), aNobii (<http://www.anobii.com/anobi>) y otros muchos como los que pueden ser consultados en la siguiente dirección (http://en.wikipedia.org/wiki/Collaborative_filtering).

CAPITULO 3.

PROYECTO.

Una vez presentado el proyecto a grandes rasgos con su propósito y objetivos y realizada la introducción teórica a los sistemas de recomendación en general y a los sistemas de recomendación colaborativos en particular llega el momento de pasar a detallar el desarrollo del proyecto que se ha realizado.

Este proyecto consta de dos partes bien diferenciadas, cada una correspondiente a una de las tareas básicas que deben realizar los sistemas de recomendación para conseguir sus objetivos de marketing personalizado, como son la predicción y recomendación.

1. La primera parte consiste en un estudio comparativo de los distintos tipos de algoritmos de filtrado colaborativo basados en modelos o ítem para la predicción, un diseño de pruebas y la evaluación de los resultados obtenidos en dichas pruebas.
2. La segunda parte consiste en realizar un prototipo de un sistema de recomendación basado en una arquitectura cliente/servidor con interfaz web implementando el mejor de los algoritmos previamente estudiados.

3.1 Estudio Comparativo

En este apartado vamos a encargarnos de detallar pormenorizadamente el desarrollo de un estudio comparativo de distintos algoritmos de filtrado colaborativo para elegir el mejor de ellos y, posteriormente, desarrollarlo en la práctica.

En la introducción teórica se presentaban los sistemas de recomendación colaborativos. El primer paso para su realización es formar grupos con los usuarios o ítems de la base de datos más similares entre si. Para formar estos grupos se aplicarán distintas medidas de similaridad y un algoritmo de clasificación K-nn.

Una vez creados estos grupos y evaluado el algoritmo K-nn mediante la técnica *hold-out* se estudiará el comportamiento de distintos algoritmos de predicción. Finalmente se realizarán una serie de pruebas basándose en distintas métricas de evaluación y se compararán sus resultados para de esta manera obtener el mejor de los algoritmos y pasar a su implementación práctica en la segunda parte del proyecto.

3.1.1 Notación

Antes de empezar a enumerar las numerosas formulas, expresiones y algoritmos que se van a detallar en este estudio resulta conveniente dejar clara la notación que se va a emplear para que no se produzcan equívocos entre los lectores:

- Un usuario será aquel elemento representado por $u_i \in U = \{u_1, \dots, u_n\}$
- Un ítem será aquel elemento representado por $i_i \in I = \{i_1, \dots, i_m\}$
- La similaridad entre dos ítems i_j e i_k se representará como $s(i_j, i_k)$
- Una evaluación o puntuación de un usuario u_i sobre un ítem i_j se representará como r_{u_i, i_j}

- Finalmente, una predicción sobre el ítem i_j del usuario u_i se representará como p_{u_i, i_j}

3.1.2 Algoritmo K-nn

Un paso imprescindible para la realización de un sistema de recomendación colaborativo de calidad es la formación de grupos de usuarios (si es un sistema de recomendación colaborativo basado en memoria) o de ítems (si es basado en modelos, como es el caso de este proyecto) de características similares. Esta actividad se puede ver como la primera parte de un **problema de clasificación** de la base de datos y existen multitud de técnicas y algoritmos, llamados clasificadores, que permiten resolverlo. Uno de los clasificadores de mayor difusión y mejores prestaciones es el **algoritmo K-nn**, que es el que se va a utilizar en este proyecto para formar los grupos de ítems más similares para cada uno de los ítems de la base de datos.

El algoritmo K-nn (k nearest neighbors, k vecinos más cercanos) es un tipo de clasificador basado en instancias. Estos clasificadores, que trabajan directamente sobre los datos sin construir ningún tipo de modelo sobre la base de datos, están basados en aprendizaje por analogía encuadrándose dentro del paradigma perezoso del aprendizaje. Este paradigma se caracteriza por:

- El trabajo se retrasa lo máximo posible.
- No se construye ningún modelo, el modelo es la propia base de datos sobre la que se trabaja.
- Se trabaja cuando llega un nuevo objeto a clasificar: se buscan los casos más parecidos y la clasificación se construye en función de la clase a la que dichos casos pertenezcan.

a) Funcionamiento del algoritmo K-nn:

Siendo i el objeto a clasificar debemos seleccionar los k objetos con $K = \{i_1, \dots, i_k\}$ tal que no existe ningún ejemplo i' fuera de K con $d(i, i') < d(i, i_j), j = 1, \dots, k$.

Una vez encontrados los k - vecinos se puede proceder a la clasificación de dos formas distintas:

- Voto por la mayoría: se clasifica el nuevo objeto en la clase mayoritaria entre los objetos de K .
- Voto compensado según la distancia: se clasifica el objeto según su distancia ponderada con el resto de objetos de K .

b) Descripción algorítmica

1. Se separan los datos en dos conjuntos disjuntos: entrenamiento (E) y test (T)
2. Llega un nuevo objeto i_a
3. Se obtienen los k objetos i_1, \dots, i_k del conjunto E mas cercanos a i_a
4. Se clasifica el objeto i_a de una de las dos maneras siguientes:¹
 1. Voto por la mayoría
 2. Voto ponderado según la distancia

c) Principales características

- Es un algoritmo robusto frente al ruido cuando el valor de k es moderado ($k > 1$).

¹ Este último paso del algoritmo K-nn no se va a implementar en este proyecto al limitarnos a encontrar grupos de vecinos sin necesidad de clasificar a los objetos.

- Es bastante eficaz cuando el número de clases posibles es alto y cuando los datos son heterogéneos o difusos.
- Tiene una complejidad temporal de $O(dn^2)$ siendo $O(d)$ la complejidad de la métrica de distancia utilizada.
- El hecho de no utilizar modelos sino la base de datos al completo provoca que sea ineficiente en memoria.
- Es válido tanto para clasificación como para predicción numérica.

Para este estudio se utilizará un algoritmo K-nn para seleccionar los k ítems más similares para cada uno de los ítems que componen nuestra base de datos. Lo que variará de una prueba a otra será la métrica de medida o similaridad empleada para calcularlo y el tamaño de los conjuntos de test y entrenamiento utilizados para su evaluación *hold-out*.

3.1.3 Medidas de similaridad

Un paso previo crucial para el filtrado colaborativo basado en ítem es el cálculo de la similaridad entre los distintos ítems y la selección de los k más similares. El concepto de similaridad se puede representar de muy diversas formas de entre las cuales hemos elegido para este proyecto la siguiente:

$s(x, y) : U \times U \rightarrow [0, 1]$ midiendo el grado de similaridad entre x e y siendo mayor cuanto más cerca de 1 se encuentre.

La similaridad cumple con las propiedades:

- Reflexiva: $(s(x, x) = 1)$
- Simétrica: $(s(x, y) = s(y, x))$.

Por lo tanto para calcular los ítems más similares a uno dado x , con una medida de similaridad entre ítems $s(x, y) : U \times U \rightarrow [0, 1], y \in U - x$, seleccionamos los k ítems y tal que la

función sea máxima.

Es conveniente aclarar que, en este proyecto, para calcular la similaridad entre dos ítems x e y , sólo se tendrán en cuenta a aquellos usuarios que hayan evaluado a ambos ítems y no siendo tomados en consideración el resto.

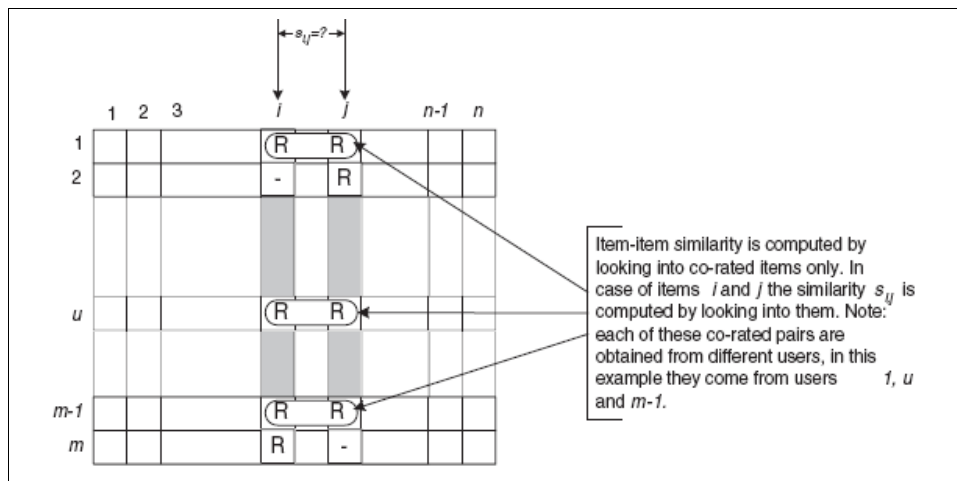


Figura 3.1.1. *Calculo de similaridad basado solo en ítems co-evaluados*

Existen multitud de funciones que nos pueden dar una medida de la similaridad entre dos elementos pero nosotros hemos elegido para este proyecto implementar solo las tres siguientes:

1. Coeficiente Coseno

En este método se supone que dos ítems x e y son vectores en el espacio y la similaridad entre ellos vendrá dada por el coseno que formen sus ángulos. La expresión para su cálculo es la siguiente:

$$s(x, y) = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n (x_i)^2 \sum_{i=1}^n (y_i)^2}}$$

Siendo x_i el valor del objeto x para el usuario i , y_i el valor del objeto y para el usuario i y n el número de usuarios que han evaluado tanto x como y .

2. Distancia Euclídea

Una forma de calcular la similaridad entre dos objetos puede ser calcular la distancia entre los mismos ya que cuanto menor sea esa distancia mayor será la similaridad. Una distancia cumple con las siguientes propiedades:

1. $d(x, y) = 0 \Leftrightarrow x = y$
2. $d(x, y) = d(y, x)$ (Propiedad simétrica)
3. $d(x, z) \leq d(x, y) + d(y, z)$

Existen múltiples funciones para el cálculo de distancias y una de las más importantes y utilizadas es la distancia euclídea cuya expresión genérica es la siguiente:

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Pero esta distancia plantea el inconveniente de que el valor resultante no está restringido a un intervalo entre $[0, 1]$ que es el intervalo en el que hemos comentado que vamos a representar la similaridad. Para remediar este inconveniente recurrimos a la normalización de la fórmula anterior, que quedaría de la siguiente manera:

$$d_n(x, y) = \frac{\sqrt{\sum_{i=1}^n \frac{(x_i - y_i)^2}{A_i^2}}}{\sqrt{n}}$$

Siendo A_i la amplitud del dominio del usuario i , la cual se calcula como: $A_i = b_i - a_i$ donde b y a son los límites superior e inferior respectivamente de tal dominio. Este dominio del usuario, en nuestro proyecto, es el comprendido por el intervalo de enteros entre 1 y 5, ambos inclusive.

La similaridad será por tanto: $s(x, y) = 1 - d_n(x, y)$

3. Coeficiente de Correlación de Pearson

Este coeficiente de correlación de Pearson apareció por primera vez en la literatura dentro del contexto del proyecto **GroupLens** donde se empleaba como la base para la asignación de pesos. Este coeficiente es un índice que mide la relación lineal entre dos variables cuantitativas, siendo independiente de la escala de medidas de dichas variables y estando acotado dentro del intervalo [-1, 1]. La expresión que permite calcular este coeficiente es la siguiente:

$$s(x, y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

Siendo \bar{x} e \bar{y} la media de todos los valores de x e y respectivamente.

3.1.4 Algoritmos de predicción basados en ítem

Una vez calculado el conjunto de los ítems más similares para cada uno de los ítems de la base de datos mediante alguna de las medidas de similaridad vistas anteriormente, llega el momento del paso más crucial en el filtrado colaborativo: elegir la técnica o algoritmo adecuado para realizar la predicción.

No existen algoritmos mejores o peores sino que existen algoritmos que se ajustan mejor o peor al conjunto de datos [7]. Esto se debe a que muchos de los algoritmos de filtrado colaborativo han sido diseñados para un conjunto de datos específico. Si este conjunto de datos específico tiene muchos más usuarios que ítems puede resultar inapropiada su ejecución sobre conjuntos de datos donde se tienen más ítems que usuarios y viceversa.

En este proyecto hemos tomado en consideración dos de los múltiples algoritmos que existen para resolver esta cuestión y vamos a comprobar cual se ajusta mejor a nuestra base de datos:

1. item average + adjustment

Esta técnica presupone que una predicción para un usuario concreto sobre un ítem es igual al valor medio de ese ítem más un ajuste que viene a ser la suma ponderada de las evaluaciones hechas por el usuario y su similitud con el ítem activo [9]. La expresión para dicha técnica es la siguiente:

$$p_{u_a, i_a} = \bar{r}_{i_a} + \frac{\sum_{h=1}^n s(i_a, i_h) (r_{u_a, i_h} - \bar{r}_{u_a})}{\sum_{h=1}^n |s(i_a, i_h)|}$$

Siendo u_a el usuario activo, i_a el ítem cuyo valor se quiere predecir y \bar{r}_{i_a} y \bar{r}_{u_a} las puntuaciones medias del usuario y el ítem, respectivamente. Estas medias se calculan de la siguiente manera:

$$\bar{r}_{i_a} = \frac{\sum_{h=1}^m r_{u_h, i_a}}{m} \qquad \bar{r}_{u_a} = \frac{\sum_{h=1}^n r_{u_a, i_h}}{n}$$

Donde n es el número de ítems que el usuario activo ha puntuado y m es el número de usuarios que han puntuado el ítem a predecir.

Existen dos enfoques diferenciados para afrontar esta técnica atendiendo al número de valores seleccionados para realizar la predicción:

- **Todos menos 1** -> se conocen todas las evaluaciones que ha hecho el usuario salvo la que se quiere predecir.
- **Dados n** -> solo se conocen n evaluaciones del total que ha hecho el usuario. Esta n suele ser un número potencia de 2.

2. weighted sum

Este método calcula la predicción de un ítem i por parte del usuario activo u_a como la suma de las evaluaciones del usuario u_a sobre ítems similares a i . Cada una de estas evaluaciones esta ponderada por la correspondiente similaridad $s(i, j)$ entre los ítems i y j [12]. Podemos denotar esta técnica de la siguiente manera:

$$p(u_a, i_a) = \frac{\sum_{h=1}^k s(i_a, i_h) * r_{u_a, i_h}}{\sum_{h=1}^k |s(i_a, i_h)|}$$

Indicando k los k ítems más similares al ítem i_a .

Básicamente, esta técnica intenta captar como evalúa el usuario activo a ítems similares al que se quiere predecir. Es necesario ponderar estas evaluaciones con la similaridad para asegurarnos de que la predicción entra dentro del rango previamente definido.

3.1.5 Diseño de Pruebas

Una vez introducidos los algoritmos de clasificación, las medidas de similaridad y las técnicas de predicción que se van a utilizar se puede pasar al diseño de las pruebas para comparar combinaciones de distintos valores de estos parámetros y decidir cual de ellas ofrece unos mejores resultados. En tal diseño de pruebas vamos a detallar las especificaciones software y hardware en las que se van a realizar las pruebas, el conjunto de datos sobre el que se van a realizar y las distintas métricas de evaluación que se van a utilizar para analizar los resultados obtenidos.

1. Especificaciones software y hardware

Resulta fundamental especificar los requerimientos tanto hardware como software con las que se van a realizar las pruebas ya que la variación de las mismas podría producir unos resultados sustancialmente diferentes a los que se han obtenido en este proyecto.

Las pruebas se han realizado en un PC de sobremesa con un microprocesador **AMD Athlon XP+ 3000** y 512 Mb de memoria RAM sobre un sistema operativo **Windows XP Professional Service Pack 2**.

El lenguaje de programación empleado para la implementación de estas pruebas ha sido **Java** mediante el entorno de desarrollo **NetBeans 5.0** y el sistema gestor de bases de datos empleado para la comunicación con el conjunto de datos de prueba ha sido **Microsoft Access** a través de **ODBC**.

2. Conjunto de datos

El conjunto de datos utilizado para evaluar los experimentos realizados en este proyecto es un ejemplo de la base de datos **MovieLens** (disponible en <http://www.grouplens.org>) formado por 943 usuarios, 1682 películas y 100000 puntuaciones habiendo puntuado cada uno de esos 943 usuarios un mínimo de 20 películas y habiendo sido puntuada cada película al menos una vez.

Esta base de datos se encuentra en un formato textual poco manejable y eficiente por lo que la hemos transformado a un formato de base de datos más aconsejable para su tratamiento por parte de un módulo desarrollado en Java a través de ODBC como es **MS Access**. La implementación en Java del módulo de transformación o formateo se encuentra disponible en el **Anexo IV**.

Una vez realizada esta transformación o formateo de la base de datos a un formato más adecuado a nuestros propósitos se ha construido una base de datos en MS Access llamada **movieranks** que contiene las tres tablas siguientes:

USUARIOS: una tabla de 943 filas en la que cada una de estas filas esta compuesta por los siguientes campos:

- ID_USER: entero. Llave primaria. Identificador numérico y unívoco del usuario.
- EDAD: entero. Edad del usuario.
- GENERO: cadena de 1 carácter. Género del usuario (M si es hombre, F si es mujer).
- PROFESION: cadena de 25 caracteres. Profesión del usuario.
- COD_POSTAL: cadena de 5 caracteres. Código postal del usuario.
- NUM_PUNTUACIONES: entero. Campo calculable. Número de películas puntuadas por el usuario.



Figura 3.1.2. Vista Diseño de la tabla USUARIOS

PELICULAS: tabla de 1682 filas con los siguientes campos cada una:

- ID_MOVIE: entero. Llave primaria. Identificador numérico y unívoco de la película.
- TITULO: cadena de 81 caracteres. Título de la película.
- FECHA: fecha. Fecha de estreno de la película.
- IMDB_URL: cadena de 134 caracteres. Enlace a la entrada en IMDB de la película.
- DESCONOCIDO: booleano. Verdadero si no se conoce el género de la película.
- ACCION: booleano. Verdadero si la película es de acción.
- AVENTURAS: booleano. Verdadero si la película es de aventuras.
- ANIMACION: booleano. Verdadero si la película es de animación.
- INFANTIL: booleano. Verdadero si es una película infantil.
- COMEDIA: booleano. Verdadero si la película es una comedia.
- CRIMEN: booleano. Verdadero si es una película de crimen.
- DOCUMENTAL: booleano. Verdadero si la película es un documental.
- DRAMA: booleano. Verdadero si la película es un drama.
- FANTASIA: booleano. Verdadero si la película es de fantasía.
- NEGRO: booleano. Verdadero si la película es de género negro.
- TERROR: booleano. Verdadero si la película es de terror.
- MUSICAL: booleano. Verdadero si la película es un musical.
- MISTERIO: booleano. Verdadero si la película es de misterio.
- ROMANTICO: booleano. Verdadero si la película es romántica.
- CIENCIA-FICCION: booleano. Verdadero si la película es de ciencia-ficción.
- THRILLER: booleano. Verdadero si la película es un thriller.
- GUERRA: booleano. Verdadero si la película es de guerra.
- WESTERN: booleano. Verdadero si la película es un western.

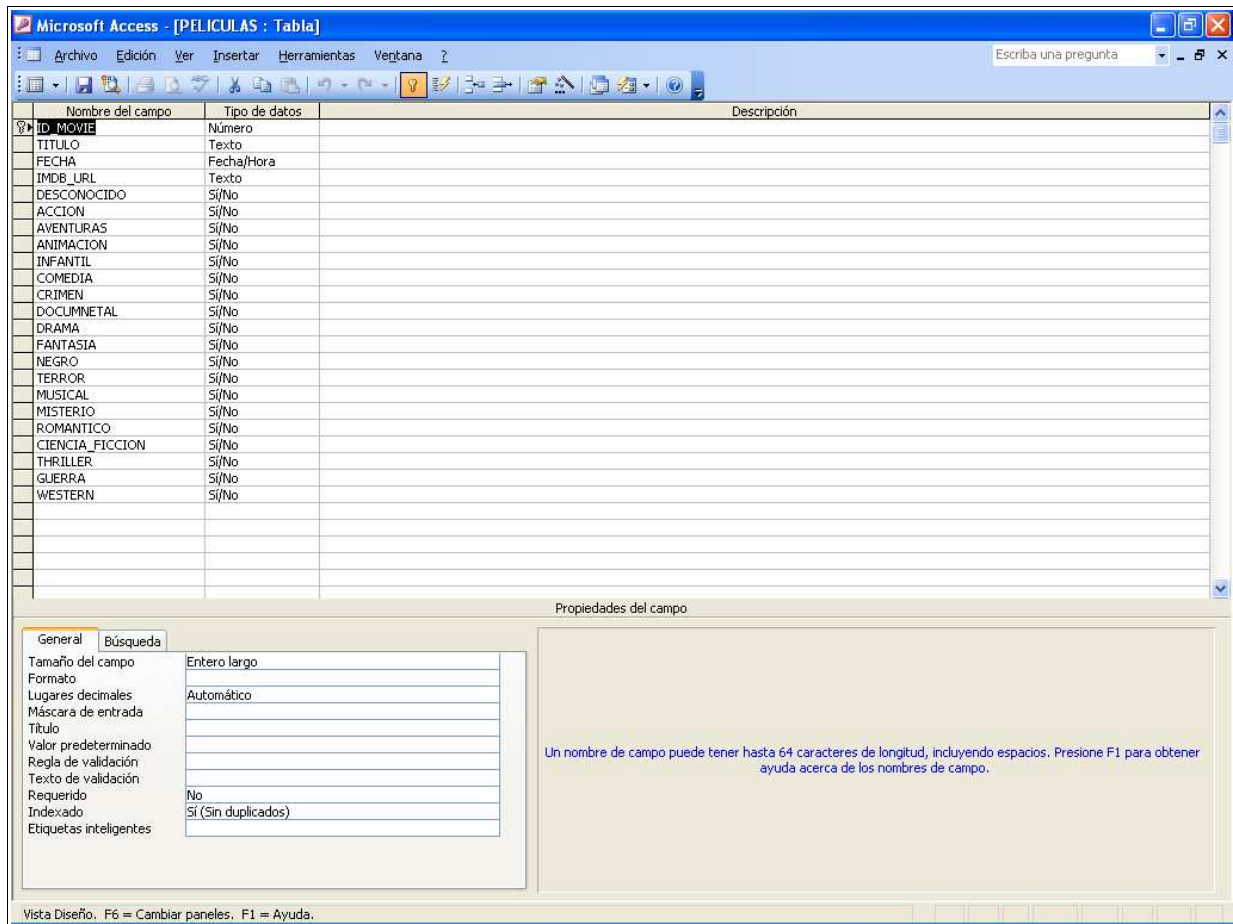


Figura 3.1.3. Vista Diseño de la tabla PELICULAS

PUNTUACIONES: tabla con 100000 filas con los siguientes campos cada una:

- ID_USER: entero. Llave primaria. Llave foránea. Identificador del usuario.
- ID_MOVIE: entero. Llave primaria. Llave foránea. Identificador de la película.
- RATING: byte. Puntuación (1, 2, 3, 4 o 5) del usuario sobre la película.

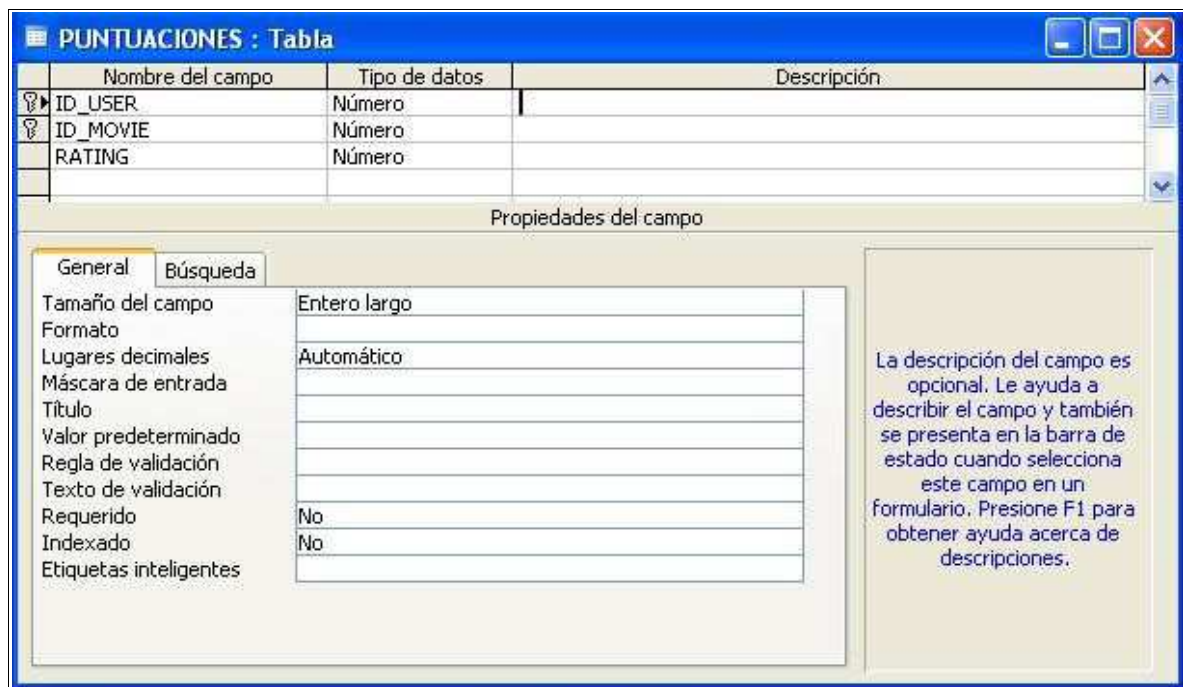


Figura 3.1.4. Vista Diseño de la tabla PUNTUACIONES

Por otra parte, en la base de datos **movieranks** existe otra tabla auxiliar llamada **ALQUILADAS** que guarda las películas alquiladas pero todavía no puntuadas de cada usuario. Esta tabla no tiene incidencia en este estudio sino para el posterior desarrollo del sistema de recomendación colaborativo por lo que será allí donde se detalle adecuadamente².

3. Métricas de evaluación

Para evaluar la bondad de los resultados de las pruebas realizadas existen multitud de métricas. Dentro de estas métricas, un tipo muy importante son las **métricas de precisión**, de las cuales existen, principalmente dos clases: las métricas de precisión estadística y las métricas de precisión de apoyo a la decisión [9].

² Concretamente en la sección 3.2.3.1) Diseño de los datos

- **Métricas de precisión estadística:** son aquellas que evalúan la precisión de un sistema de recomendación comparando las predicciones numéricas con las puntuaciones reales para cada ítem que tenga tanto puntuación como predicción. Algunas de estas métricas son el MAE (Mean Absolute Error), el RMSE (Root Mean Squared Error) o la correlación.
- **Métricas de precisión de apoyo a la decisión:** son aquellas que evalúan como de efectivamente las predicciones ayudan a los usuarios a seleccionar ítems adecuados. Algunas de las métricas de esta clase son la puntuación inversa, la sensibilidad ROC (Receiver Operating Characteristic) o la sensibilidad PRC (Precision Recall Curve).

Sin duda, elegir unas buenas métricas resulta fundamental. Para este proyecto hemos elegido dos cuyos resultados se pueden complementar bastante bien.

La primera de estas métricas es una métrica de precisión estadística llamada **MAE** (Mean Absolut Error) que es, con diferencia, la métrica de este tipo más utilizada. El MAE es una medida en valor absoluto de la desviación entre las puntuaciones reales (r) y sus predicciones (p) cuya expresión es la siguiente:

$$MAE = \frac{\sum_{h=1}^n |p_h - r_h|}{n}$$

Cuanto menor sea este MAE, que estará acotado por la amplitud del dominio de las puntuaciones, más exactas serán las predicciones permitiendo unas mejores recomendaciones.

La segunda métrica que vamos a utilizar es una métrica temporal ya que de nada nos sirve que el algoritmo de predicción proporcione predicciones muy exactas si el coste en tiempo para ofrecerlas es demasiado alto. Calcularemos el tiempo de ejecución de cada algoritmo en milisegundos (ms) siendo el mejor de estos algoritmos el que consiga ofrecer unos mejores predicciones en un menor tiempo.

3.1.6 Implementación de Pruebas

Como ya se ha comentado en varias ocasiones esta parte de la memoria esta dedicada al estudio comparativo de distintos algoritmos de filtrado colaborativo mediante la evaluación y análisis de los resultados de diversas pruebas realizadas. Pues bien, en este apartado se va a pasar a detallar el proceso de implementación de dichas pruebas.

El primer paso consiste en dividir el conjunto de datos detallado en el apartado anterior en dos conjuntos disjuntos independientes de usuarios: uno será el denominado **conjunto de entrenamiento** mientras que el otro, aconsejablemente más pequeño, será el denominado **conjunto de test**. A este enfoque se le conoce como una técnica de evaluación **hold-out** y suele emplearse para bases de datos relativamente grandes como la que se utiliza en este proyecto.

Sobre el conjunto de entrenamiento se aplicará el algoritmo **k-nn** para calcular el conjunto de vecinos más similares para cada uno de los ítems. Esta similitud será calculada mediante uno de las tres **medidas de similitud** presentadas anteriormente en esta memoria.

Este conjunto de k-vecinos es indispensable para el **cálculo de predicciones**, un cálculo que se realizará para los usuarios contenidos en el conjunto de test mediante la utilización de uno de los **algoritmos de predicción** ya estudiados.

Finalmente obtendremos una medida de la precisión de estas predicciones gracias a la medida de precisión estadística **MAE** (Mean Absolut Error) y una medida del **tiempo empleado** para realizarlas.

Todo el código realizado para la creación de este módulo de pruebas esta incluido en el material que acompaña a esta memoria y las partes más destacadas del mismo se pueden encontrar en el **Anexo III**.

3.1.7 Evaluación de Resultados

Se han realizado 6 pruebas diferentes, ejecutándose 20 iteraciones de cada una de ellas. Para cada una de estas pruebas se han variado los valores de uno o más de los siguientes parámetros:

- **Porcentaje Entrenamiento/ Test:** indica el porcentaje de la base de datos que se usará como conjunto de entrenamiento en tanto por uno siendo el conjunto de test el resto de la base de datos.
- **Número de vecinos:** proporciona el valor k del algoritmo **k-nn** para la construcción del conjunto de k ítems más similares para cada ítem.
- **Medida de similitud:** indica cual de las tres medidas de similitud consideradas (*distancia euclídea, coeficiente coseno o coeficiente de correlación de Pearson*) se ha utilizado.
- **Algoritmo de predicción:** indica cual de los dos algoritmos de predicción considerados se ha utilizado. El algoritmo *item+adjustment* se identifica en las tablas y gráficos de resultados como **Pr1** mientras que el algoritmo *weighted sum* se identifica como **Pr2**. Siempre que se utilice el algoritmo *item+adjustment* se emplearán los dos enfoques vistos previamente: *todos menos 1* (identificado como **TM1**) y *dados n* con los valores 2, 4 y 8 para dicha n (identificados respectivamente como **D2, D4, D8**).

Los valores de estos parámetros para cada una de las pruebas son los siguientes:

	PRUEBA1	PRUEBA2	PRUEBA3	PRUEBA4	PRUEBA5	PRUEBA6
ENT/TEST	0.8/0.2	0.8/0.2	0.6/0.4	0.8/0.2	0.8/0.2	0.8/0.2
Nº VEC.	20	20	20	20	10	40
MED. SIMILAR.	Coeficiente Coseno	Coeficiente Coseno	Coeficiente Coseno	Correlación de Pearson	Coeficiente Coseno	Coeficiente Coseno
ALG. PRED.	item+adjust.	weighted sum	item+adjust.	item+adjust.	item+adjust.	item+adjust.

Tabla 3.1.1. Valores de los parámetros de las distintas pruebas

Los resultados presentados en las siguientes tablas y gráficos se corresponden con los valores medios de las métricas de evaluación MAE y temporal para cada una de las ejecuciones ya comentadas.

3.1.7.1 Resultados de las Pruebas

PRUEBA 1

		Pr1			
		TM1	D2	D4	D8
Ej1	MAE	0,8399574	0,9091844	0,9091844	0,9091844
	Tiempo	0,052910052	6,095238	6,4179893	6,724868
Ej2	MAE	0,7961574	0,860809	0,8863211	0,91598016
	Tiempo	1,1640211	5,5079365	6,047619	5,0846562
Ej3	MAE	0,8235353	0,8792336	0,9336889	0,9691272
	Tiempo	0,6878307	5,714286	5,2486773	5,031746
Ej4	MAE	0,7579644	0,8274456	0,87336	0,9062245
	Tiempo	0,74603176	6,3650794	5,878307	5,5555553
Ej5	MAE	0,9014391	0,86159664	0,8818423	0,91135806
	Tiempo	0,7407407	5,9312167	5,9312167	4,6084657
Ej6	MAE	0,8413623	0,88274693	0,8983355	0,91679156
	Tiempo	2,1693122	6,6243386	6,571429	7,047619
Ej7	MAE	0,8255099	0,87925327	0,91216934	0,9460321
	Tiempo	0,6878307	5,724868	5,4497356	5,7777777
Ej8	MAE	0,847643	0,8573806	0,8834329	0,91131616
	Tiempo	1,1111112	6,148148	6,148148	5,6137567
Ej9	MAE	0,9025	0,893788	0,9013497	0,91731286
	Tiempo	0,52910054	5,6666665	5,820106	5,571429
Ej10	MAE	0,8256069	0,87137294	0,88979936	0,9165067
	Tiempo	0,74603176	5,5079365	5,2486773	5,5555553
Ej11	MAE	0,8389686	0,87945104	0,9008881	0,9398318
	Tiempo	1,4285715	6,4550266	6,7777777	6,4232802
Ej12	MAE	0,82146186	0,8927211	0,9204125	0,94389373
	Tiempo	0,84656084	4,978836	5,6719575	4,878307
Ej13	MAE	0,8818918	0,8852441	0,91919035	0,9437338
	Tiempo	0,74603176	6,3597884	5,3492064	5,878307
Ej14	MAE	0,9714764	0,85976565	0,88290125	0,91171217
	Tiempo	1,1640211	5,7830687	5,4497356	5,4074073
Ej15	MAE	0,8545881	0,87416464	0,916549	0,93262964
	Tiempo	0,7407407	5,4074073	5,2433863	4,825397
Ej16	MAE	0,79279286	0,88925016	0,92068464	0,9670446
	Tiempo	0,52910054	4,724868	5,296296	5,3439155
Ej17	MAE	0,84542274	0,86703396	0,8916349	0,9212311
	Tiempo	0,52910054	6,84127	5,724868	5,3439155
Ej18	MAE	0,9055007	0,85441107	0,8953181	0,91266114
	Tiempo	0,31746033	5,5185184	5,5132275	5,026455
Ej19	MAE	0,8242213	0,88827103	0,90811175	0,94462746
	Tiempo	0,8994709	6,4074073	6,835979	6,94709
Ej20	MAE	0,83496475	0,9076144	0,92021275	0,9413919
	Tiempo	0,47619048	5,301587	5,5132275	5,1904764

Tabla 3.1.2. Resultados de Prueba 1

Como una tabla de tal tamaño puede ser poco clarificadora a la hora de extraer cualquier tipo de conclusión, se van a presentar los datos en dos gráficos de líneas (uno para cada una de las métricas de evaluación empleadas) para comprobar su comportamiento a través de las distintas ejecuciones:

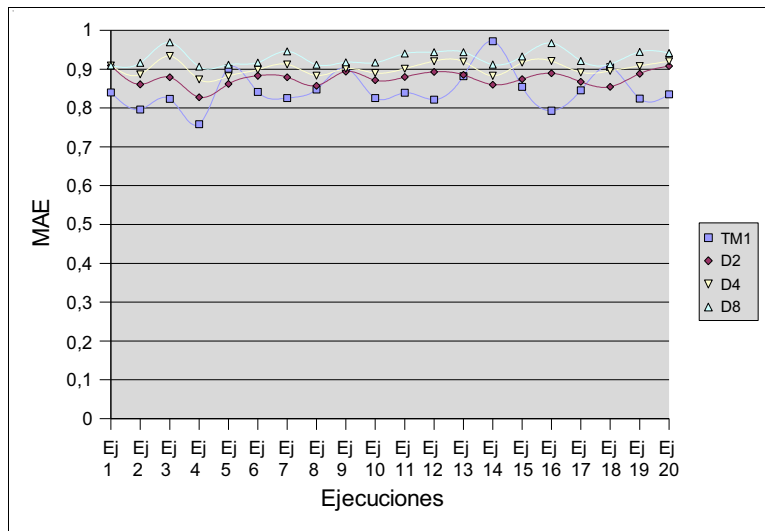


Figura 3.1.5. Gráfico de MAE de Prueba 1

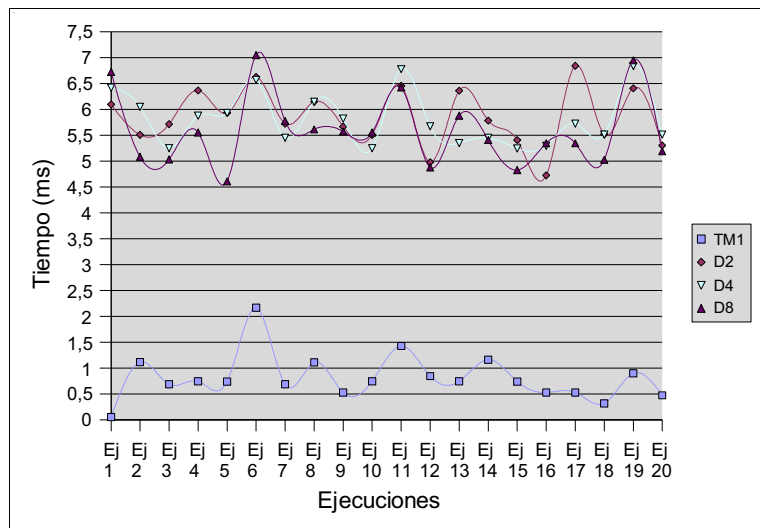


Figura 3.1.6 Gráfico de Tiempo de Prueba 1

Los gráficos anteriores ayudan a comprender el comportamiento de cada uno de los enfoques a lo largo de las ejecuciones pero no dejan claro cual de ellos es el mejor.

Para resolver esta situación se han calculado los valores medios de cada enfoque para las dos métricas empleadas y se presentan en los siguientes gráficos de barras:

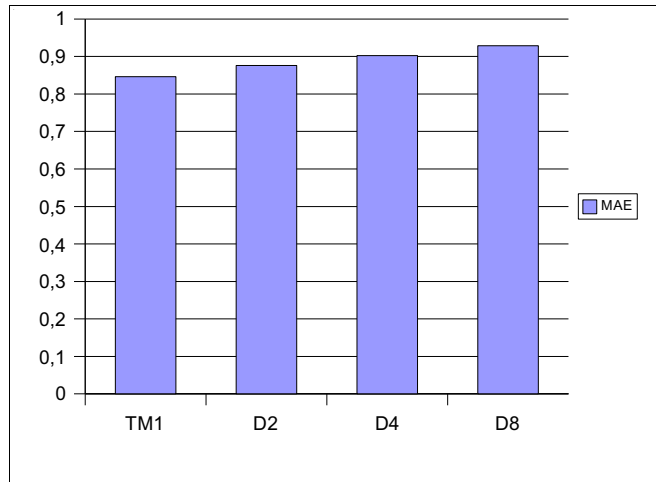


Figura 3.1.7. Media MAE de Prueba 1

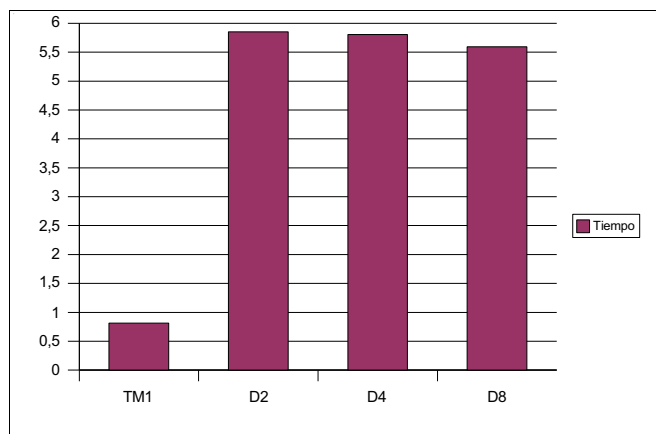


Figura 3.1.8. Media de Tiempo de Prueba 1

A la vista de estos dos últimos gráficos se observa claramente que el enfoque **Todos Menos 1** obtiene unos resultados mucho mejores en términos de tiempo que los demás enfoques y, aunque con unos márgenes mucho más igualados, también obtiene unos mejores resultados en cuanto a precisión de la predicción. Esta vez, las dos métricas no se contradicen y señalan al mismo enfoque como el mejor. Por lo tanto, a la hora de elegir cual prueba se ha saldado con mejores resultados, para esta primera prueba tomaremos en consideración el enfoque **Todos Menos 1**.

PRUEBA 2

		Pr2
Ej1	MAE	0,7471493
	Tiempo	0,052910052
Ej2	MAE	0,76753604
	Tiempo	0,052910052
Ej3	MAE	0,7380677
	Tiempo	0,052910052
Ej4	MAE	0,758815
	Tiempo	0
Ej5	MAE	0,6601054
	Tiempo	0
Ej6	MAE	0,72152346
	Tiempo	0,052910052
Ej7	MAE	0,75190395
	Tiempo	0,052910052
Ej8	MAE	0,80621403
	Tiempo	0,052910052
Ej9	MAE	0,6928996
	Tiempo	0
Ej10	MAE	0,67935616
	Tiempo	0
Ej11	MAE	0,7982117
	Tiempo	0,52910054
Ej12	MAE	0,7453146
	Tiempo	0
Ej13	MAE	0,6918417
	Tiempo	0,105820104
Ej14	MAE	0,7857819
	Tiempo	0,05820106
Ej15	MAE	0,7051473
	Tiempo	0
Ej16	MAE	0,7463787
	Tiempo	0
Ej17	MAE	0,67296684
	Tiempo	0
Ej18	MAE	0,77536833
	Tiempo	0
Ej19	MAE	0,76340055
	Tiempo	0
Ej20	MAE	0,6837682
	Tiempo	0

Tabla 3.1.3. Resultados de Prueba 2

Una vez obtenida esta tabla se van a extraer los datos en diversos gráficos que mostrarán el comportamiento del algoritmo a lo largo de las ejecuciones y el valor medio de las métricas MAE y temporal para el mismo:

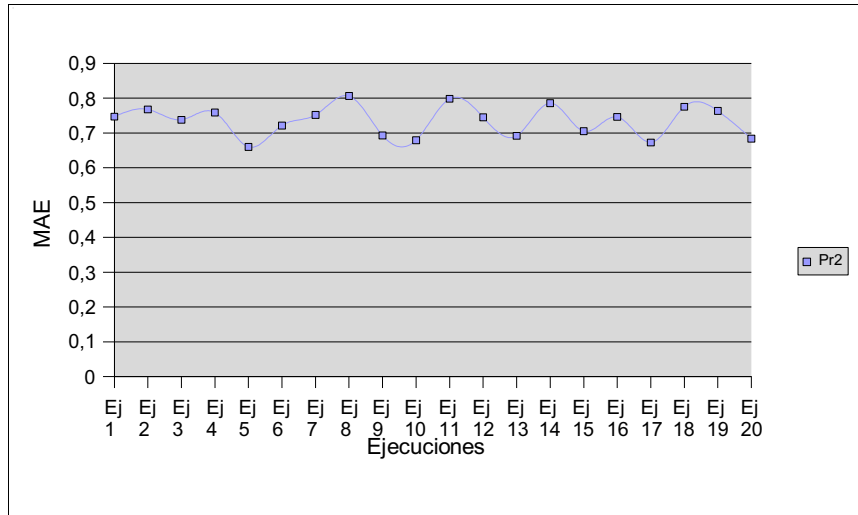


Figura 3.1.9. Gráfico de MAE de Prueba 2

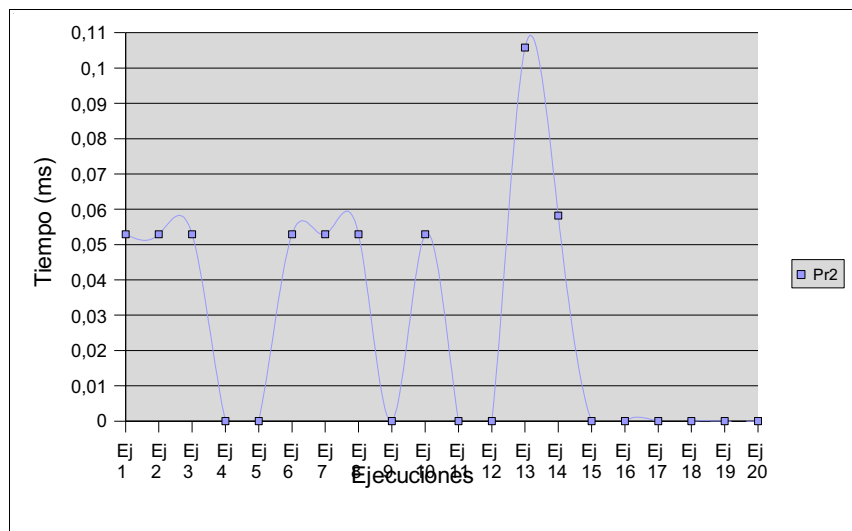


Figura 3.1.10. Gráfico de Tiempo de Prueba 2

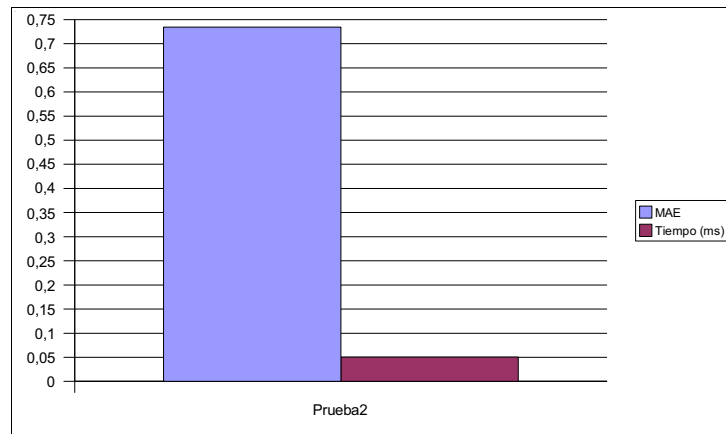


Figura 3.1.11. Media de MAE y Tiempo de Prueba 2

Se observa que esta prueba (en la que se tienen los mismos parámetros que para **Prueba 1** salvo el algoritmo de predicción) los resultados son bastante buenos, principalmente en la métrica de tiempos donde se puede decir que las ejecuciones se realizan de forma instantánea. Sin duda tiene muchas opciones de ser la prueba con mejores resultados.

PRUEBA 3

		Pr1			
		TM1	D2	D4	D8
Ej1	MAE	0,7891631	0,89330035	0,9152147	0,9312285
	Tiempo	0,85411143	5,4668436	5,363395	4,8647213
Ej2	MAE	0,879499	0,8856924	0,9203394	0,9207101
	Tiempo	0,4774536	6,1352787	5,97878	6,2175064
Ej3	MAE	0,82645065	0,87468016	0,906191	0,9188787
	Tiempo	0,6366048	5,156499	5,2864723	5,498674
Ej4	MAE	0,87647164	0,87846494	0,8963687	0,92968816
	Tiempo	1,2493368	6,376658	6,6896553	6,69496
Ej5	MAE	0,80260146	0,8706919	0,8935741	0,9066327
	Tiempo	0,928382	6,769231	6,888594	6,29443
Ej6	MAE	0,8005821	0,8560503	0,88483757	0,9089134
	Tiempo	0,90716183	6,259947	6,535809	6,509284
Ej7	MAE	0,8450893	0,8834645	0,9113812	0,94587654
	Tiempo	0,928382	6,424403	6,32626	7,0928383
Ej8	MAE	0,81564826	0,8848531	0,896048	0,9216531
	Tiempo	0,6366048	5,7161803	5,840849	5,4164457
Ej9	MAE	0,81764555	0,8686024	0,899244	0,9222548
	Tiempo	1,0079576	6,005305	6,2519894	6,4190984
Ej10	MAE	0,87638116	0,90519404	0,94469315	0,9581149
	Tiempo	0,6366048	6,4323606	6,427056	6,3448277
Ej11	MAE	0,8751021	0,8729384	0,9134295	0,93857694
	Tiempo	0,7161804	5,4960213	5,8488064	5,5782495
Ej12	MAE	0,8973061	0,88148755	0,91291827	0,94721955
	Tiempo	1,0079576	5,4668436	5,525199	5,5331564
Ej13	MAE	0,8463662	0,8999509	0,9231452	0,9404347
	Tiempo	0,7692308	6,4084883	5,6525197	6,474801
Ej14	MAE	0,8534515	0,8960156	0,9334678	0,9508041
	Tiempo	0,5835544	4,888594	5,474801	5,2572947
Ej15	MAE	0,8708212	0,90800893	0,9257292	0,944099
	Tiempo	0,7161804	5,7612734	5,204244	5,132626
Ej16	MAE	0,87350345	0,8512323	0,88089776	0,9048089
	Tiempo	2,0477455	6,7480106	6,1909814	6,2413793
Ej17	MAE	0,825637	0,88107586	0,91245776	0,91588527
	Tiempo	0,5039788	5,31565	5,2572947	5,2519894
Ej18	MAE	0,84777	0,8971592	0,9191528	0,9335893
	Tiempo	0,3183024	5,498674	5,949602	5,204244
Ej19	MAE	0,81878644	0,8740259	0,89465946	0,9238785
	Tiempo	0,4244032	5,973475	5,5305037	6,2122016
Ej20	MAE	0,8358831	0,881792	0,9054176	0,9201863
	Tiempo	0,7161804	6,4005303	6,244032	5,657825

Tabla 3.1.4. Resultados de Prueba 3

Al igual que para las pruebas anteriores se va a proceder a extraer los datos de esta tabla a distintos gráficos para, de esta manera, poder conocer mejor su comportamiento y determinar cual de los enfoques obtiene unos mejores resultados:

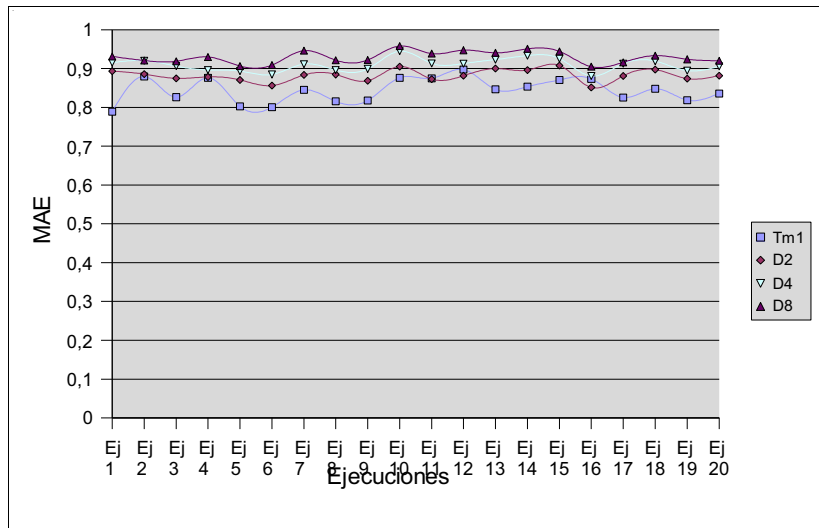


Figura 3.1.12. Gráfico de MAE de Prueba 3

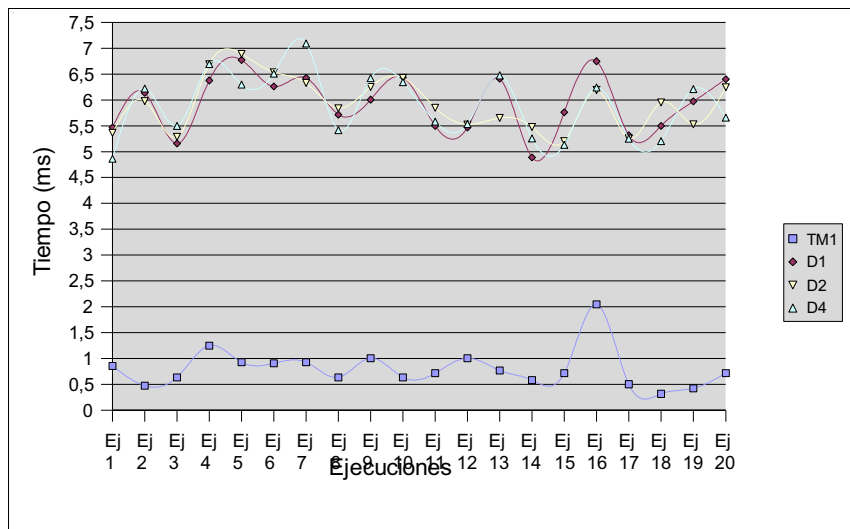


Figura 3.1.13. Gráfico de Tiempo de Prueba 3

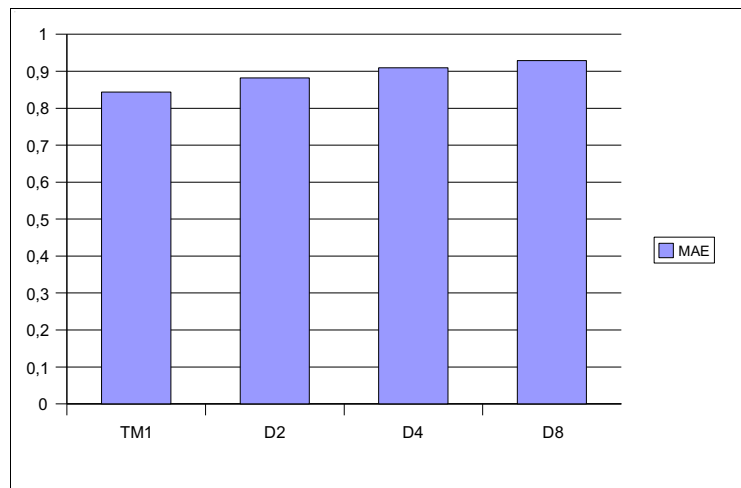


Figura 3.1.14. *Media MAE de Prueba 3*

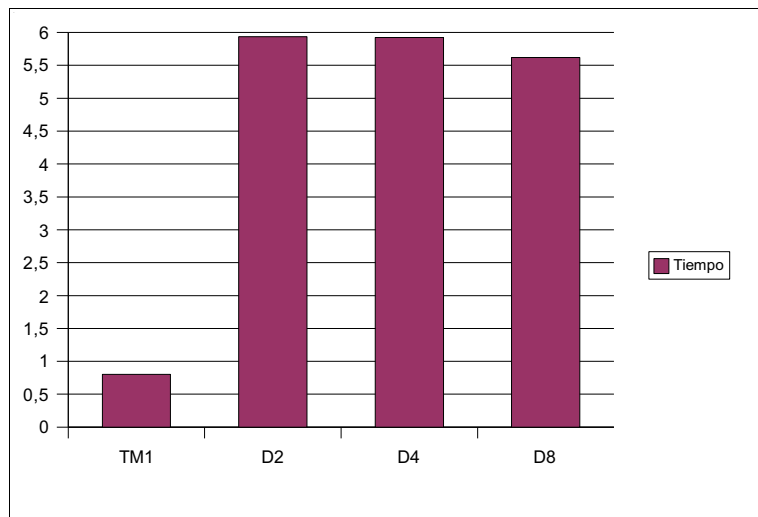


Figura 3.1.15. *Media de Tiempo de Prueba 3*

Con respecto a **Prueba 1** se ha variado un aspecto importante: el tamaño de los conjuntos de prueba y test. Ahora, el tamaño del conjunto de prueba es bastante menor con lo que es posible que se ha resentido la calidad del algoritmo **K-nn** y, por ende, el de la predicción. De todas formas esto se comprobará en la comparativa final para la cual para esta prueba se tomará en consideración el enfoque **Todos Menos 1** ya que resulta el que mejor resultados consigue tanto en tiempos como en precisión.

PRUEBA 4

		Pr1			
		TM1	D2	D4	D8
Ej1	MAE	0,8607042	0,8365449	0,8567084	0,8780762
	Tiempo	0,95238096	6,6719575	6,4074073	6,5185184
Ej2	MAE	0,91682166	0,85205007	0,8568071	0,8824493
	Tiempo	0,47619048	6,2063494	6,042328	6,1957674
Ej3	MAE	0,86716366	0,82911175	0,8424223	0,8596457
	Tiempo	0,26455027	5,4021163	5,883598	5,994709
Ej4	MAE	0,8683027	0,8712737	0,8861082	0,90868163
	Tiempo	0,63492066	5,7777777	4,9312167	5,6666665
Ej5	MAE	0,94376343	0,81840605	0,8287324	0,85602224
	Tiempo	1,3227513	5,7777777	5,714286	5,883598
Ej6	MAE	0,9014174	0,8534792	0,8649634	0,8837512
	Tiempo	0,64021164	5,6666665	5,031746	4,5608463
Ej7	MAE	0,7685422	0,86439955	0,87317735	0,90054226
	Tiempo	0,47619048	6,296296	6,4550266	5,7883596
Ej8	MAE	0,8442674	0,8608898	0,8669227	0,8894639
	Tiempo	0,21164021	6,142857	6,571429	5,4550266
Ej9	MAE	0,7600607	0,84827	0,86184424	0,87639487
	Tiempo	0,31746033	5,989418	6,095238	5,296296
Ej10	MAE	0,8615474	0,8364576	0,84677404	0,87195504
	Tiempo	0,31746033	5,5132275	5,883598	4,873016
Ej11	MAE	0,9248854	0,8581736	0,87718195	0,88356274
	Tiempo	0,63492066	5,4603176	5,132275	5,7830687
Ej12	MAE	0,9355303	0,8331444	0,85204023	0,86891615
	Tiempo	0,105820104	5,2433863	5,1957674	5,3492064
Ej13	MAE	0,8092929	0,8444746	0,8468639	0,87992805
	Tiempo	0,26455027	6,2010584	5,830688	6,571429
Ej14	MAE	0,89334834	0,86578757	0,8760373	0,8990298
	Tiempo	0,42328042	6,6772485	5,989418	5,984127
Ej15	MAE	0,912916	0,8239494	0,8368945	0,85647357
	Tiempo	0,4814815	5,6190476	6,2539682	5,132275
Ej16	MAE	0,8746802	0,8278694	0,8446043	0,86261165
	Tiempo	0,52910054	5,5079365	5,3650794	4,5502644
Ej17	MAE	0,8394401	0,8302364	0,8516222	0,88166857
	Tiempo	0,26455027	5,6772485	4,814815	6,037037
Ej18	MAE	0,8833269	0,8256317	0,8331051	0,8530259
	Tiempo	0,63492066	5,4603176	4,820106	5,031746
Ej19	MAE	0,9561469	0,85536635	0,875424	0,87770987
	Tiempo	0,52910054	5,291005	4,867725	5,4708996
Ej20	MAE	0,8289949	0,8306874	0,85026306	0,8605016
	Tiempo	0,47619048	6,73545	6,2486773	6,2486773

Tabla 3.1.5. Resultados de Prueba 4

Al igual que para las pruebas anteriores se va a proceder a extraer los datos de esta tabla a distintos gráficos para, de esta manera, poder conocer mejor su comportamiento y determinar cual de los enfoques obtiene unos mejores resultados:

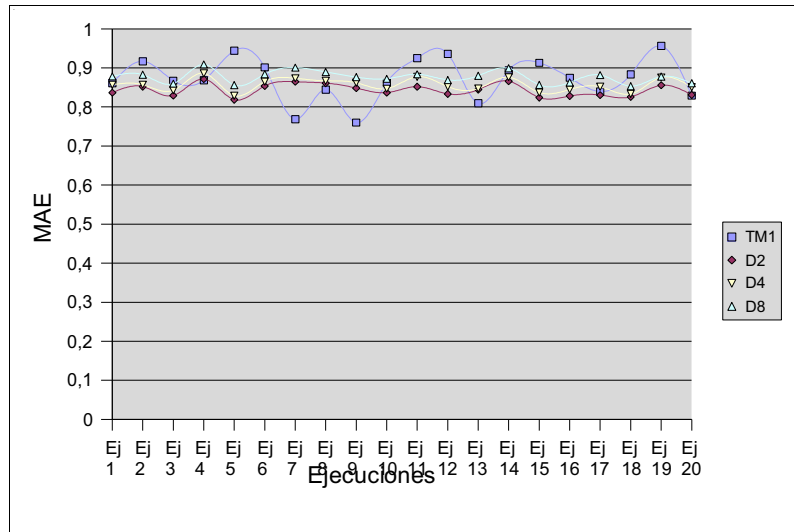


Figura 3.1.16. Gráfico de MAE de Prueba 4

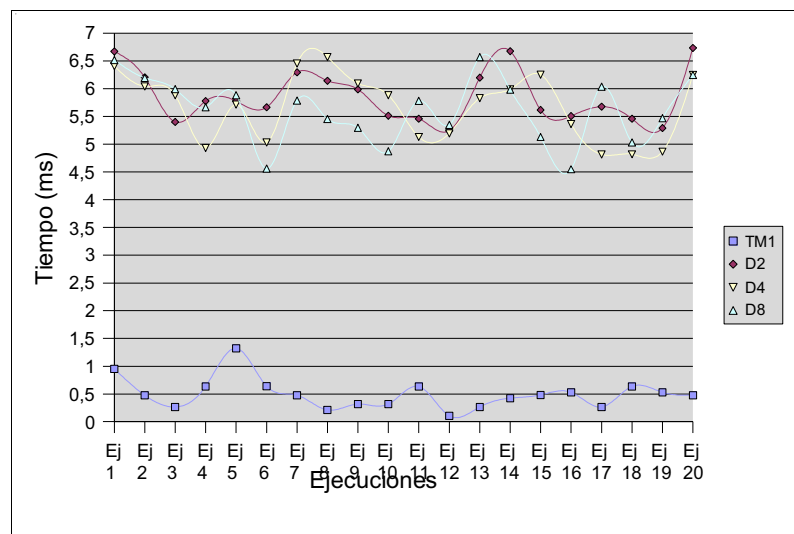


Figura 3.1.17. Gráfico de Tiempo de Prueba 4

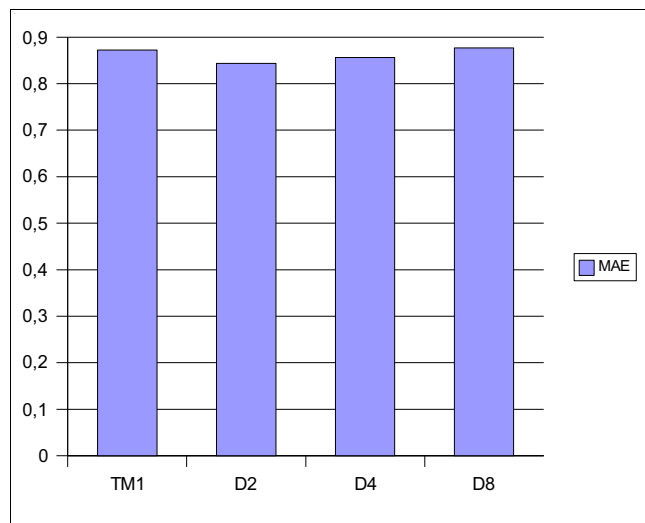


Figura 3.1.18. *Media MAE de Prueba 4*

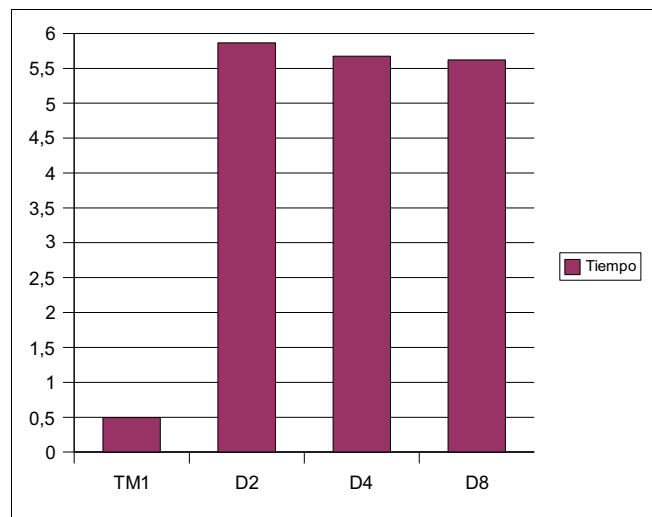


Figura 3.1.19. *Media de Tiempo de Prueba 4*

Para esta prueba se ha optado por implementar como métrica de similitud el coeficiente de **correlación de Pearson** manteniendo el resto de parámetros como en Prueba 1 y nos encontramos por primera vez con que las métricas de evaluación difieren en sus resultados: en cuanto al tiempo el enfoque Todos Menos 1 resulta claramente el mejor pero, sin embargo, en cuanto a la precisión MAE, el enfoque Datos 2 obtiene un valor medio mejor mientras que el Todos Menos 1 obtiene la peor media de todos los enfoques y su comportamiento sufre muchos altibajos.

Ante esta situación, se hace indispensable el seguir una norma pre-establecida para decidir cual de los dos enfoques elegir y se ha decidido que esta norma sea la siguiente: *se elegirá siempre el enfoque que obtenga unas mejores predicciones salvo que estas predicciones las obtenga en unos tiempos mucho mayores, del orden de las varias decenas de milisegundos, que el resto.*

Es siguiendo esta norma como se decide que sea el enfoque **Dados 2** el que se tenga en consideración para la comparativa final entre pruebas.

PRUEBA 5

		Pr1			
		TM1	D2	D4	D8
Ej1	MAE	0,9072662	0,86696905	0,8971425	0,9141342
	Tiempo	2,5978837	4,825397	5,984127	3,7089946
Ej2	MAE	0,8406394	0,845517	0,86411136	0,8763314
	Tiempo	3,074074	4,3439155	5,6243386	3,973545
Ej3	MAE	0,9131363	0,8430198	0,85652953	0,8997373
	Tiempo	2,2275133	5,0846562	4,984127	4,6031747
Ej4	MAE	0,76707053	0,82992387	0,8434733	0,85594904
	Tiempo	0,31746033	5,883598	5,185185	5,137566
Ej5	MAE	0,8459521	0,85896873	0,8873888	0,90763444
	Tiempo	1,0582011	5,2433863	4,9259257	4,6719575
Ej6	MAE	0,87210196	0,84033525	0,8577663	0,88309723
	Tiempo	3,8730159	5,2380953	4,5502644	5,095238
Ej7	MAE	0,9171906	0,84013665	0,8524127	0,88226414
	Tiempo	1,3280423	5,1904764	5,6666665	4,2380953
Ej8	MAE	0,922505	0,85781723	0,8730222	0,9017307
	Tiempo	1,3333334	5,031746	5,4074073	4,5502644
Ej9	MAE	0,93261915	0,84813744	0,89451057	0,90415
	Tiempo	0,8518519	6,142857	5,1904764	4,978836
Ej10	MAE	0,89275736	0,8629659	0,8885409	0,9142797
	Tiempo	2,867725	4,714286	5,4126983	4,4973545
Ej11	MAE	0,9269985	0,85142136	0,86582416	0,88185364
	Tiempo	2,3280423	5,089947	4,7671957	4,714286
Ej12	MAE	0,85118383	0,83779997	0,8591151	0,8734808
	Tiempo	3,8201058	5,4074073	5,2486773	4,708995
Ej13	MAE	0,8598805	0,87188923	0,89007473	0,9353956
	Tiempo	1,3227513	4,9206347	4,7671957	4,3544974
Ej14	MAE	0,88086855	0,86474746	0,8874667	0,90103316
	Tiempo	0,5873016	5,137566	5,4603176	4,978836
Ej15	MAE	0,80939966	0,8865382	0,90608037	0,91948307
	Tiempo	2,9629629	4,6137567	5,1957674	5,132275
Ej16	MAE	0,8320645	0,84064746	0,85337394	0,90312403
	Tiempo	3,915344	4,867725	5,2539682	5,3544974
Ej17	MAE	0,92881316	0,85982704	0,88465124	0,90902203
	Tiempo	2,6984127	5,296296	5,4708996	4,867725
Ej18	MAE	0,7908301	0,84678316	0,85996443	0,88195854
	Tiempo	2,4444444	5,5608463	5,3492064	4,3968253
Ej19	MAE	0,8606862	0,85465544	0,86832386	0,90531075
	Tiempo	2,6560845	5,1957674	4,6613755	5,031746
Ej20	MAE	0,9083502	0,8792419	0,8904219	0,923175
	Tiempo	1,7460318	5,291005	4,7671957	4,4126983

Figura 3.1.6. Resultados de Prueba 5

Al igual que para las pruebas anteriores se va a proceder a extraer los datos de esta tabla a distintos gráficos para, de esta manera, poder conocer mejor su comportamiento y determinar cual de los enfoques obtiene unos mejores resultados:

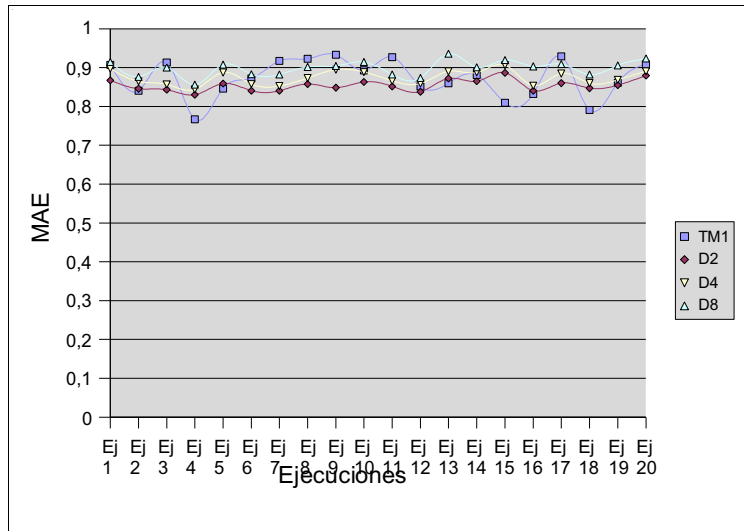


Figura 3.1.20. Gráfico de MAE de Prueba 5

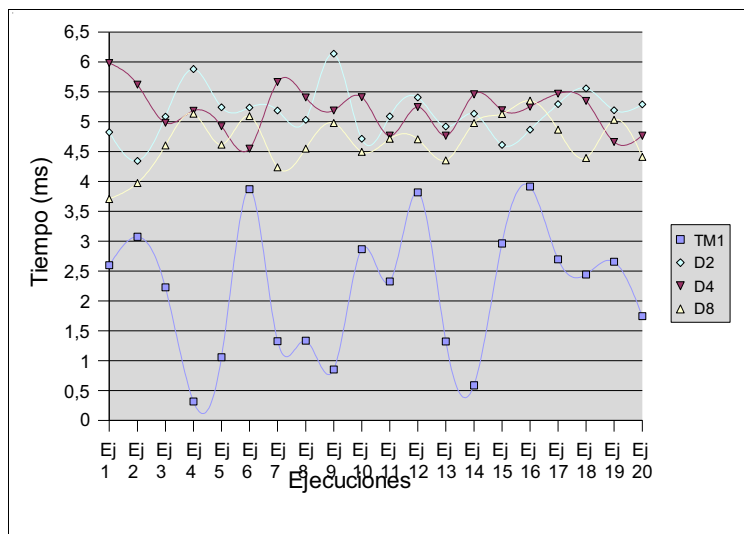


Figura 3.1.21. Gráfico de Tiempo de Prueba 5

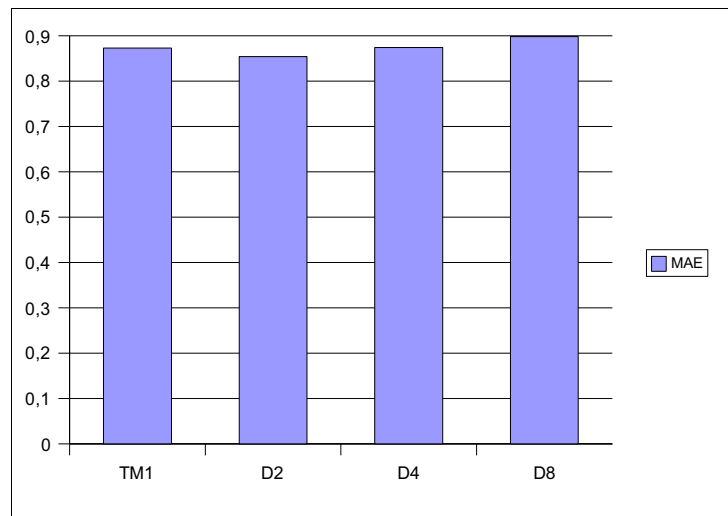


Figura 3.1.22. *Media MAE de Prueba 5*

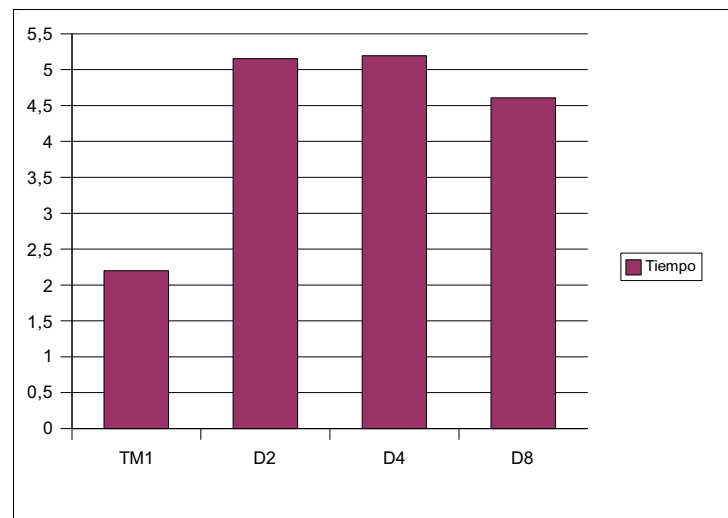


Figura 3.1.23. *Media de Tiempo de Prueba 5*

Tanto para esta prueba como para la siguiente hemos variado el **número de vecinos** que se calcularán para cada uno de los ítems del conjunto de entrenamiento para comprobar como afecta el tamaño de este conjunto de vecinos en la predicción. En esta prueba se ha pasado de 20 a 10 vecinos y nos encontramos con que las dos métricas de evaluación vuelven a diferir: el enfoque Datos 2 realiza mejores y más constantes predicciones mientras que Todos Menos 1 obtiene los mejores tiempos de ejecución. Como en la prueba anterior antepone la precisión al tiempo ya que las diferencias en milisegundos son inapreciables y tomamos en consideración para la comparativa final al enfoque **Datos 2**.

PRUEBA 6

		Pr1			
		TM1	D2	D4	D8
Ej1	MAE	0,8785218	0,90194535	0,9330488	0,9224521
	Tiempo	2,3862433	5,6137567	5,132275	5,148148
Ej2	MAE	0,8509358	0,9283263	0,93042356	0,93445486
	Tiempo	2,8042328	4,6666665	4,296296	4,2380953
Ej3	MAE	0,78445894	0,91019446	0,9588665	0,9601047
	Tiempo	2,862434	4,878307	5,1957674	5,3492064
Ej4	MAE	0,8758615	0,9023815	0,92250836	0,9398476
	Tiempo	2,910053	5,6243386	5,137566	5,5079365
Ej5	MAE	0,83165884	0,9200539	0,96138096	0,9507927
	Tiempo	5,5132275	5,4021163	5,4603176	5,291005
Ej6	MAE	0,8477021	0,92652965	0,9868059	0,966225
	Tiempo	2,3862433	4,285714	5,719577	5,148148
Ej7	MAE	0,7904137	0,91061544	0,9536638	0,9663543
	Tiempo	3,021164	5,0793653	5,5608463	5,2063494
Ej8	MAE	0,83304375	0,92507315	0,9536549	0,97372794
	Tiempo	4,5026455	5,2433863	5,4074073	5,301587
Ej9	MAE	0,876602	0,9170559	0,9275307	0,95398974
	Tiempo	2,8042328	4,7724867	5,4550266	4,984127
Ej10	MAE	0,8774581	0,96115494	0,97623384	0,9649713
	Tiempo	3,1216931	5,0846562	5,5079365	4,724868
Ej11	MAE	0,8377185	0,9487743	0,9399986	0,95247054
	Tiempo	3,132275	4,4444447	3,8148148	4,984127
Ej12	MAE	0,799156	0,9007503	0,9519072	0,96031195
	Tiempo	3,1746032	5,0846562	5,4074073	5,3544974
Ej13	MAE	0,82683885	0,928964	0,95691574	0,95505124
	Tiempo	3,2910054	5,1904764	5,1957674	5,3439155
Ej14	MAE	0,8056894	0,89666265	0,9058138	0,91789865
	Tiempo	3,7619047	5,5079365	4,142857	5,0846562
Ej15	MAE	0,85651004	0,87876725	0,9087125	0,9308502
	Tiempo	2,9682539	4,978836	5,0846562	5,301587
Ej16	MAE	0,81148046	0,90710205	0,9318752	0,9627555
	Tiempo	3,068783	5,0846562	4,724868	5,1904764
Ej17	MAE	0,8445794	0,88921064	0,9527363	0,9284814
	Tiempo	2,6455026	4,9259257	4,5132275	4,5555553
Ej18	MAE	0,8567844	0,92218	0,9311507	0,969939
	Tiempo	2,8042328	3,925926	4,5608463	4,5555553
Ej19	MAE	0,7830478	0,8537299	0,9024189	0,92815894
	Tiempo	2,915344	5,7777777	5,5026455	5,6243386
Ej20	MAE	0,918258	0,88576114	0,92661893	0,9053871
	Tiempo	2,915344	6	4,5026455	4,9206347

Tabla 3.1.7. Resultados de Prueba 6

Al igual que para las pruebas anteriores se va a proceder a extraer los datos de esta tabla a distintos gráficos para, de esta manera, poder conocer mejor su comportamiento y determinar cual de los enfoques obtiene unos mejores resultados:

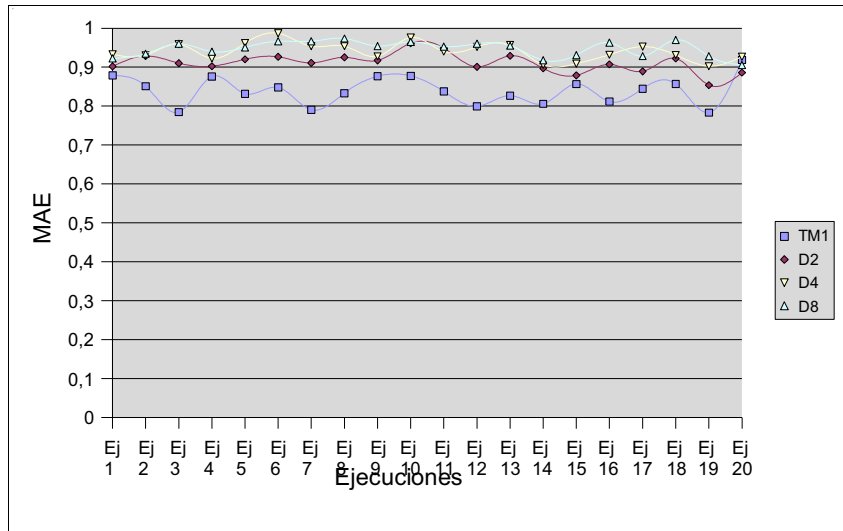


Figura 3.1.24. Gráfico de MAE de Prueba 6

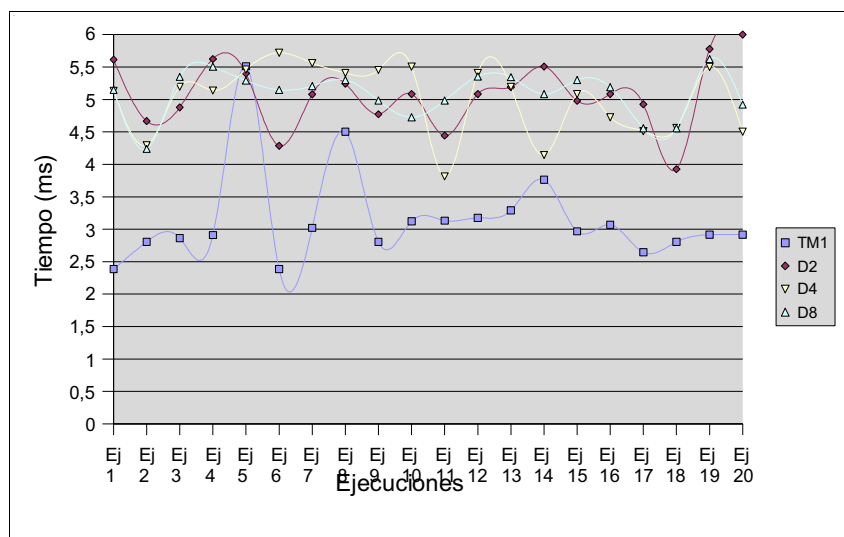


Figura 3.1.25. Gráfico de Tiempo de Prueba 6

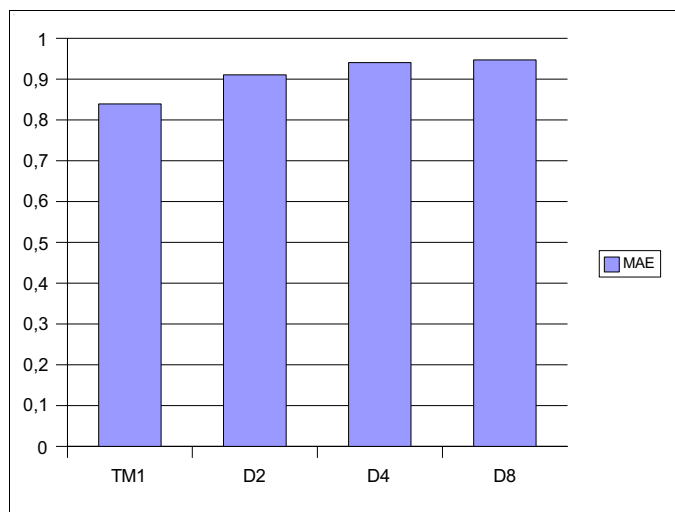


Figura 3.1.26. Media MAE de Prueba 6

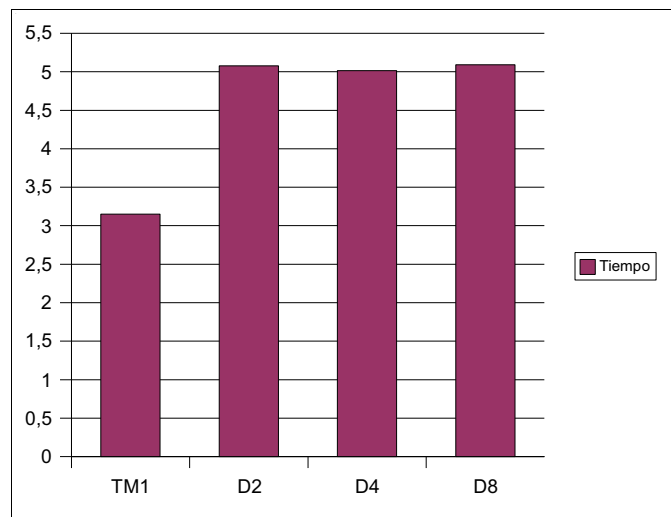


Figura 3.1.27. Media de Tiempo de Prueba 6

Para esta prueba se ha aumentado el **número de vecinos** de 20 a 40 y esta vez las dos métricas de evaluación coinciden en que, aunque por márgenes estrechos, el enfoque **Todos Menos 1** realiza las mejores predicciones en el mejor tiempo por lo que este enfoque será el que se tome en consideración para la comparativa final entre las distintas pruebas.

3.1.7.2 Comparativa entre las Pruebas

Una vez realizadas las pruebas y analizados sus resultados llega el momento de comparar estos resultados y determinar cual prueba los ha obtenido mejores para la posterior implementación de un sistema de recomendación colaborativo con los valores de los parámetros de dicha prueba.

Como en las pruebas que han implementado el algoritmo de predicción **item average + adjustment** se presentan distintos resultados según el enfoque seguido vamos a recordar cual de estos enfoques es el tomado en consideración (es decir, el que mejores resultados haya obtenido) para cada prueba:

PRUEBA	ENFOQUE
PRUEBA1	Todos Menos 1
PRUEBA3	Todos Menos 1
PRUEBA4	Dados 2
PRUEBA5	Dados 2
PRUEBA6	Todos Menos 1

Tabla 3.1.8. *Enfoque elegido para cada prueba*

En el siguiente gráfico de barras se muestran con claridad los resultados comparados de todas las pruebas para las métricas MAE y temporal:

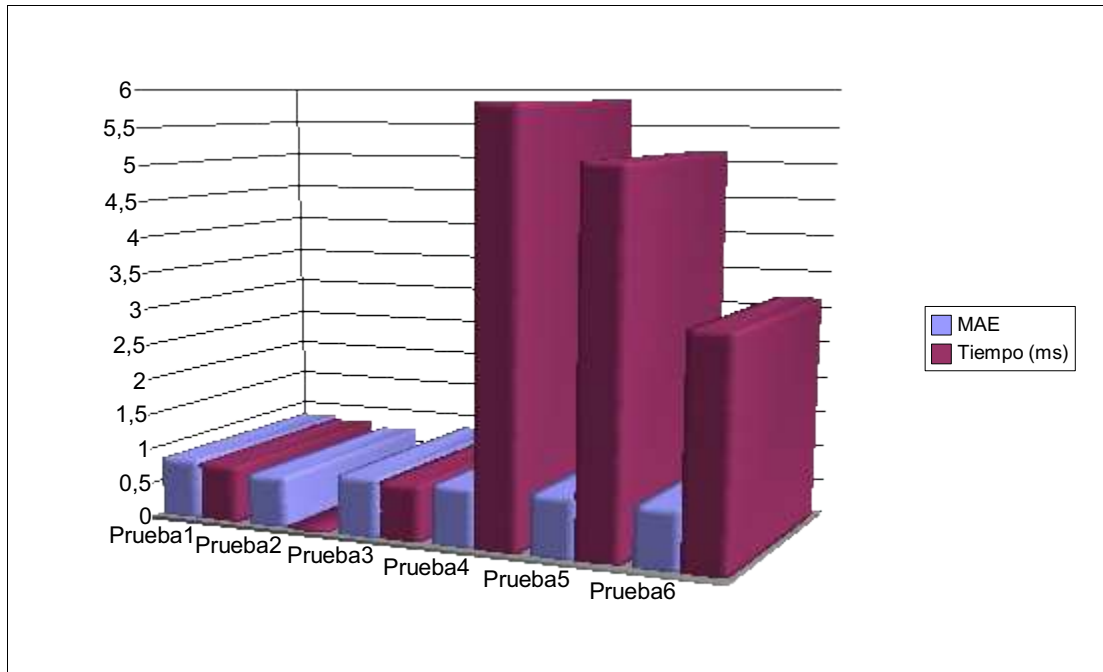


Figura 3.1.28. Comparativa de todas las pruebas

A la vista del gráfico queda claro que tanto para la métrica de precisión MAE como para la métrica temporal es la **Prueba 2** la que obtiene unos mejores resultados. Por lo tanto, en la segunda parte de este proyecto se implementará el prototipo de un sistema de recomendación colaborativo basado en una arquitectura cliente/servidor que utilizará un algoritmo de filtrado colaborativo formado por los siguientes valores de los parámetros:

	PRUEBA2
ENT/TEST	0.8/0.2
Nº VEC.	20
MED. SIMILAR.	Coeficiente Coseno
ALG. PRED.	weighted sum

Tabla 3.1.9. Valores del Sistema de Recomendación Colaborativo

3.2 Sistema de recomendación colaborativo

Una vez realizado el estudio comparativo de algoritmos de filtrado colaborativo que conformaba la primera parte de este proyecto, en este apartado se va a detallar el desarrollo del prototipo de un sistema de recomendación colaborativo basado en una arquitectura cliente/servidor con interfaz web implementando el mejor de los algoritmos previamente estudiados.

Por lo tanto, esta segunda parte es un proyecto de desarrollo software y, como tal, para su desarrollo deben seguirse las actividades de la **Ingeniería del Software**. No existe una definición única y estandarizada para la Ingeniería del Software pero las dos que se presentan a continuación pueden resultar perfectamente válidas para este cometido:

- Ingeniería del Software es la construcción de software de calidad con un presupuesto limitado y un plazo de entrega en contextos de cambio continuo.
- Ingeniería del Software es el establecimiento y uso de principios y métodos firmes de ingeniería para obtener software económico que sea fiable y funcione de manera eficiente en máquinas reales.

Las actividades que conforman la Ingeniería del Software son las siguientes:

- **Especificación de Requerimientos:** se obtienen el propósito del sistema y las propiedades y restricciones del mismo.
- **Análisis del Sistema:** se obtiene un modelo del sistema correcto, completo, consistente, claro y verificable.
- **Diseño del Sistema:** se definen los objetivos del proyecto y las estrategias a seguir para conseguirlos.
- **Implementación:** se traduce el modelo a código fuente.

En los puntos siguientes se profundizará en cada una de estas actividades y en como se han llevado a cabo en el ámbito nuestro proyecto.

3.2.1 Especificación de Requerimientos

El primer paso en la Ingeniería del Software debe ser determinar el **propósito último del proyecto**, las propiedades que debe satisfacer y las restricciones a las que está sometido. Este es, sin duda, un paso de vital importancia dentro del desarrollo de un proyecto software ya que, sin conocer el propósito del proyecto y todas las limitaciones de diversa índole a las que debe hacer frente, difícilmente se podrá realizar una aplicación software que cumpla dicho propósito.

En un proyecto de ámbito comercial destinado a una empresa real, para determinar el propósito del mismo se recurre a una serie de estudios como pueden ser entrevistas con los clientes, encuestas con posibles usuarios, estudios de la situación actual del sistema o estudios de viabilidad. En nuestro caso no nos encontramos ante un proyecto comercial sino ante uno académico por lo que el propósito es conocido desde el mismo momento de la concepción del mismo:

El desarrollo de un sistema de recomendación colaborativo para el alquiler de películas basado en una arquitectura cliente/servidor con interfaz web implementando el mejor algoritmo de filtrado colaborativo de los estudiados previamente.

Habiendo determinado el propósito último del proyecto, el siguiente paso consiste en especificar los requerimientos del mismo. Los **requerimientos de un proyecto software** son el conjunto de propiedades o restricciones definidas con total precisión, que dicho proyecto software debe satisfacer. Existen dos tipos bien diferenciados de tales requerimientos:

- **Requerimientos funcionales:** aquellos que se refieren específicamente al funcionamiento de la aplicación o sistema.
- **Requerimientos no funcionales:** aquellos no referidos al funcionamiento estricto sino a otros factores externos.

En los dos siguientes subapartados se pasarán a definir cuales son estos requerimientos (tanto funcionales como no funcionales) para el proyecto del que se ocupa esta memoria. Sin embargo, estas definiciones sólo serán previas ya que en la actividad de análisis del sistema, donde se crearán los casos de uso y sus escenarios, se descubrirán nuevas necesidades que no son observables en esta primera actividad y que permitirán refinar

completamente estos requerimientos.

3.2.1.1 Requerimientos funcionales

Los requerimientos funcionales de un sistema software son aquellos que se encargan de describir las funcionalidades que el sistema debe proporcionar a los usuarios del mismo para cumplir sus expectativas.

Normalmente, estos requerimientos se obtendrían de la interacción con el cliente mediante diversas entrevistas y/o encuestas. En nuestro caso, al tratarse de un proyecto académico, nos encontramos ante la situación de la no existencia de cliente alguno por lo que la información sobre las funcionalidades que debe disponer el sistema se ha obtenido investigando otros sistemas de recomendación colaborativos que ya se encuentran en el mercado, muy especialmente aquellos de reconocido éxito y gran número de usuarios.

En base a estas investigaciones realizadas se ha llegado a la conclusión de que las funcionalidades que el usuario potencial espera de un sistema de recomendación como el nuestro son las siguientes:

- Recibir recomendaciones de objetos que no ha probado y pueden ser de su gusto.
- Puntuar los objetos que ha probado.
- Poder cambiar las puntuaciones de objetos ya puntuados.
- Consultar sus datos personales.
- Modificar su datos en caso de que cambien o sean erróneos.
- Disponer de una lista con todas las objetos disponibles en el sistema y toda la información posible sobre los mismos.
- Disponer de mecanismos de ayuda.

Una vez definidas cuales son las funcionalidades que los usuarios reclaman a un sistema de recomendación, se hace necesario caracterizar de una manera más formal y concreta como va a responder a estas funcionalidades, dentro del ámbito del alquiler de películas, nuestro sistema:

1) Recibir recomendaciones de nuevas películas

El sistema debe proporcionar al usuario, basándose en las películas ya vistas por el mismo y sus puntuaciones, una lista con las películas que no ha visto o alquilado todavía y que pueden ser de su agrado.

2) Puntuar películas

El sistema debe permitir al usuario que puntúe películas, tanto aquellas que vaya viendo como aquellas que ya han sido puntuadas pero qué, por alguna razón, el usuario quiera volver a puntuar. Además, debe actualizar la base de datos con esta información.

3) Visualizar datos personales

El sistema debe proporcionar al usuario la posibilidad de visualizar sus datos personales.

4) Modificar datos personales

El sistema debe permitir al usuario modificar sus datos personales y actualizar esta información en la base de datos.

5) Visualizar todas las películas

El sistema debe proporcionar la posibilidad de visualizar un listado con todas las películas disponibles en la base de datos al usuario.

6) Visualizar sólo las películas ya puntuadas

El sistema debe proporcionar al usuario la posibilidad de visualizar un listado con las películas que ya ha puntuado y cuales son dichas puntuaciones.

7) Visualizar sólo las películas alquiladas pero no puntuadas

El sistema debe proporcionar al usuario la posibilidad de visualizar un listado con las películas que ha alquilado pero que todavía no ha puntuado.

8) Consultar ayuda

El sistema debe proporcionar al usuario algún medio de ayuda para que pueda conocer perfectamente el manejo de la aplicación o resolver cualquier duda puntual que pueda tener.

Estas son las funcionalidades que debe proporcionar nuestro sistema de recomendación teniendo en cuenta de que en este proyecto vamos a desarrollar una versión prototipal del mismo. En una versión final del sistema, este debería satisfacer otras funcionalidades, como podrían ser:

- Recálculo instantáneo de las recomendaciones después de una actualización de la base de datos.
- Posibilidad de dar de alta a nuevos usuarios y de incorporar nuevas películas a la base de datos.
- Tours guiados de puntuaciones para que los nuevos usuarios puedan obtener un perfil amplio de manera rápida y sencilla.
- Posibilidad de realizar comentarios textuales los usuarios sobre las distintas películas.

3.2.1.2 Requerimientos no funcionales

Los requerimientos no funcionales son aquellos que restringen los requerimientos funcionales. Son tan importantes como los propios requerimientos funcionales y pueden incluso ser críticos para la aceptación del sistema. Estos requerimientos normalmente especifican propiedades del sistema o del producto en si (plataforma, velocidad, rendimiento...) y del diseño de la interfaz gráfica con el usuario además de todas las restricciones impuestas por la organización (políticas de empresa, estándares, legalidad vigente...).

Al no ser este un proyecto comercial para ninguna organización o empresa real, no debemos someternos a restricciones organizacionales. Por lo tanto, los requerimientos no funcionales que se deben obtener y analizar son los referentes a las necesidades hardware y

software de los equipos informáticos para que estos proporcionen al usuario las funcionalidades requeridas de forma eficiente y los referentes a la interfaz gráfica entre la aplicación y el usuario.

A) Requerimientos del equipo informático

Al hablar de los requerimientos del equipo informático y debido a que el marco del desarrollo de la aplicación es una arquitectura cliente/servidor, debemos diferenciar los requerimientos de equipo que necesita el servidor y los que necesita el cliente.

Las necesidades de **equipo informático del cliente** son muy simples ya que tan solo le hace falta un computador conectado a Internet (preferiblemente de banda ancha) y tener instalado un navegador capacitado para visualizar de forma correcta la aplicación (se recomienda Firefox u Opera pero podría ser válido cualquier otro).

Los requerimientos del **equipo informático del servidor**, el cual se aconseja que sea un equipo dedicado, son más amplios y se dividen en dos tipos: los requerimientos de hardware y los requerimientos software.

1) Hardware

- *Velocidad:* el equipo debe ser lo suficientemente rápido como para ejecutar la aplicación en el menor tiempo posible y con la mayor fiabilidad. Cualquier microprocesador actual es capaz de cumplir con esta labor.
- *Memoria:* el equipo debe disponer de la suficiente memoria RAM libre para realizar las operaciones que se soliciten entre la aplicación y la base de datos.
- *Almacenamiento:* el equipo que haga la labor de servidor debe tener una capacidad de almacenamiento suficiente para almacenar la base de datos con la que trabaja la aplicación y permitir con holgura las transacciones entre ambas entidades.
- *Tarjeta gráfica:* las tarjetas gráficas de las que disponen los equipos informáticos actuales son de gran potencia por lo que no es necesario establecer ningún requerimiento en este aspecto.

- *Monitor:* el monitor debe soportar una resolución de 1024x768 y superiores.
- *Conexión a Internet:* el servidor debe encargarse de que la aplicación sea accesible a través de Internet para todos sus usuarios por lo que es indispensable que se encuentre conectado a Internet a través de banda ancha las 24 horas del día.

2) Software

- *Sistema Operativo:* el servidor de la aplicación trabaja sobre un sistema operativo Windows XP
- *Navegador:* la aplicación debe poder ser visualizada desde cualquier navegador web actual aunque se recomienda el uso de Firefox en sus últimas versiones.
- *Sistema Gestor de Bases de Datos:* la aplicación trabaja con una base de datos MS Access.
- El resto del software necesario (servidor Http Apache, lenguaje de programación PHP...) será proporcionado al administrador de la aplicación, el cual dispone de un manual para su instalación en el **Anexo I**.

B) Requerimientos de la interfaz

Los requerimientos de la interfaz gráfica entre la aplicación y el usuario están íntimamente ligados a la **usabilidad** y sus principios. La usabilidad se puede definir de varias formas [17]:

- Usabilidad se define coloquialmente como facilidad de uso, ya sea de una página web, una aplicación informática o cualquier otro sistema que interactúe con un usuario.
- Usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso.

- Usabilidad es la efectividad, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico.

A partir de estas tres definiciones se pueden obtener los principios básicos de la usabilidad, los cuales se asociarán a los requerimientos no funcionales que deberá cumplir la interfaz gráfica:

- **Facilidad de aprendizaje:** se refiere a la facilidad con la que nuevos usuarios pueden tener una interacción efectiva. Depende de los siguiente factores:
 - *Predecibilidad:* una vez conocida la aplicación, se debe saber en cada momento a que estado se pasará en función de la tarea que se realice.
 - *Síntesis:* los cambios de estado tras una acción deben ser fácilmente captados.
 - *Generalización:* las tareas semejantes se resuelven de modo parecido.
 - *Familiaridad:* el aspecto de la interfaz tiene que resultar conocido y familiar para el usuario.
 - *Consistencia:* siempre se han de seguir una misma serie de pasos para realizar una tarea determinada.
- **Flexibilidad:** relativa a la variedad de posibilidades con las que el usuario y el sistema pueden intercambiar información. También abarca la posibilidad de diálogo, la multiplicidad de vías para realizar la tarea, similitud con tareas anteriores y la optimización entre el usuario y el sistema.
- **Robustez:** es el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos o, también, la capacidad del sistema para tolerar fallos. Está relacionada con los siguiente factores:
 - *Observación:* el usuario debe poder observar el estado del sistema sin que esta observación repercuta de forma negativa en él.

- *Recuperación de información:* la aplicación debe poder deshacer alguna operación y permitir volver a un estado anterior.
- *Tiempo de respuesta:* es el tiempo necesario para que el sistema pueda mostrar los cambios realizados por el usuario.

3.2.2 Análisis del Sistema

Una vez conocido el propósito del proyecto software, las propiedades que debe cumplir y las restricciones a las que debe someterse, llega el momento de **analizar el sistema** y crear un **modelo** del mismo que sea correcto, completo, consistente, claro y verificable. Para conseguir ésto se crearán y definirán casos de uso en base a los requerimientos previamente obtenidos y se describirán ciertos escenarios de acción de dichos casos de uso.

3.2.2.1 Casos de uso

Un **caso de uso** representa una clase de funcionalidad dada por el sistema como un flujo de eventos. También se puede definir como la representación de una situación o tarea de interacción de un usuario con la aplicación.

Los casos de uso describen como se realiza una tarea de manera exacta y constan de los siguientes elementos:

- Nombre único e unívoco
- Actores participantes
- Condiciones de entrada
- Flujo de eventos
- Condiciones de salida
- Requerimientos especiales

Por lo tanto, es necesario determinar cuáles son los actores participantes en cada uno de los casos de uso. Un **actor** modela una entidad externa que se comunica con el sistema, es decir, es un tipo de usuario del sistema. Un actor, al igual que un caso de uso, debe tener un nombre único y puede tener una descripción asociada.

En nuestro sistema contamos con los tres actores siguientes:

- **Cliente:** se trata del usuario tipo de la aplicación, el que la va a utilizar para recibir recomendaciones, puntuar películas y demás.
- **Administrador:** se trata del responsable de la aplicación, el que se encarga de actualizar el algoritmo de filtrado colaborativo empleado en base a las nuevas puntuaciones que van realizando los usuarios del sistema.
- **BBDD:** se trata de la base de datos que proporciona los datos a la aplicación.

Una vez definidos cuales van a ser los actores del sistema, es el momento de crear los distintos casos de uso. A la hora de realizar esta acción es importante que cada uno de los requerimientos funcionales ya definidos aparezca en al menos uno de los casos de uso aunque, por otra parte, puede haber casos de uso nuevos, en los que no aparezca ninguno de los requerimientos, ya que estamos en una fase de refinamiento del sistema donde queremos construir un modelo detallado del mismo.

Un paso previo a la creación y descripción de los distintos casos de uso es la obtención de los diversos **diagramas de casos de uso** de nuestro sistema, al que vamos a llamar **moviesrecommender**.

El primero es un *diagrama frontera*, es decir, un diagrama que describe completamente la funcionalidad de un sistema:

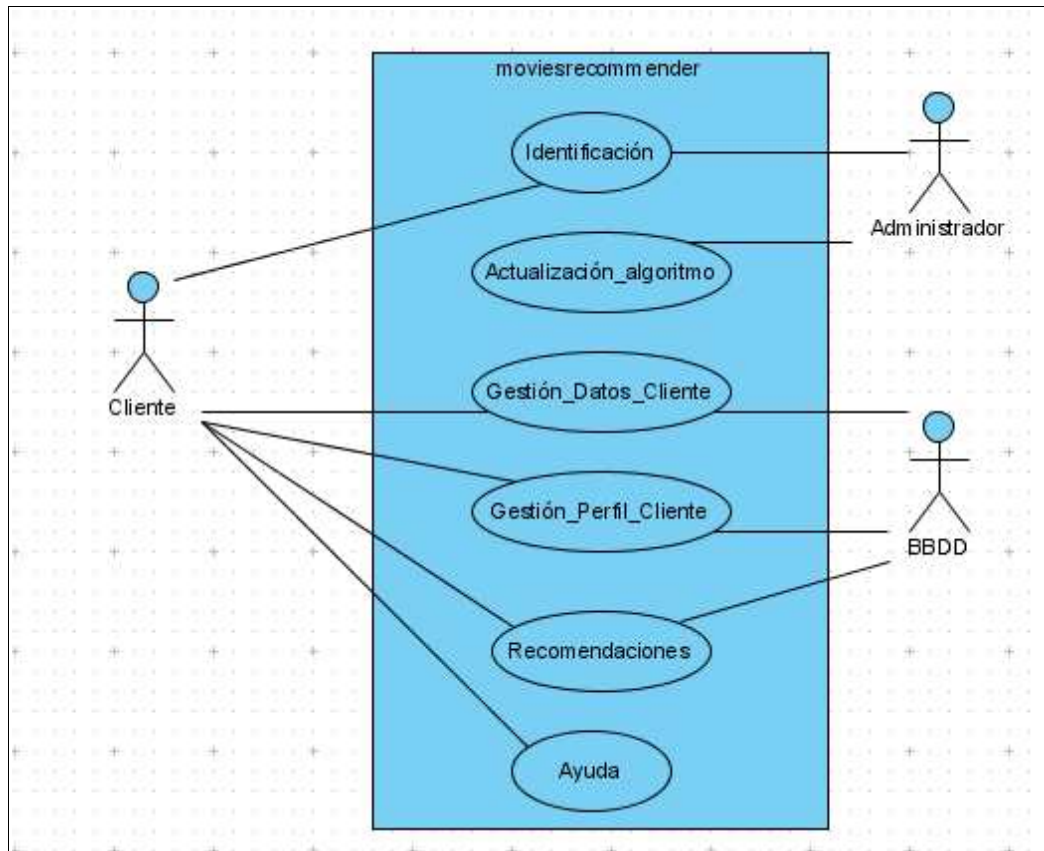


Figura 3.2.1. Diagrama frontera de moviesrecommender

Los casos de uso mostrados en un diagrama frontera pueden ser lo suficientemente exactos o, por el contrario, pueden ser concretados con un mayor detalle. A la hora de detallar un caso de uso se pueden emplear dos tipos de relaciones:

- `<<extend>>`: es una relación cuya dirección es hacia el caso de uso a detallar que representa comportamientos excepcionales del caso de uso.
- `<<include>>`: es una relación cuya dirección es contraria a la de la relación `<<extend>>` que representa un comportamiento común del caso de uso

En nuestro caso nos encontramos con que los casos de uso **Gestión_Datos_Cliente** y **Gestión_Perfil_Cliente** requieren ser detallados en más profundidad:

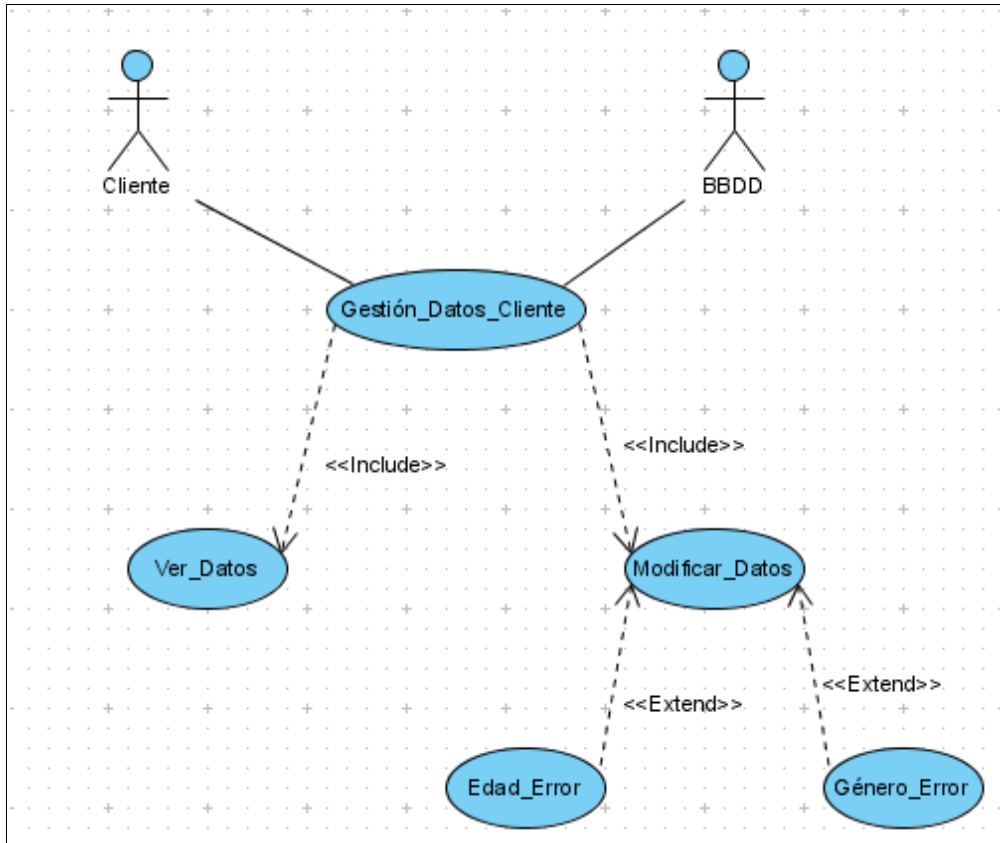


Figura 3.2.2. Diagrama del caso de uso *Gestión_Datos_Cliente*

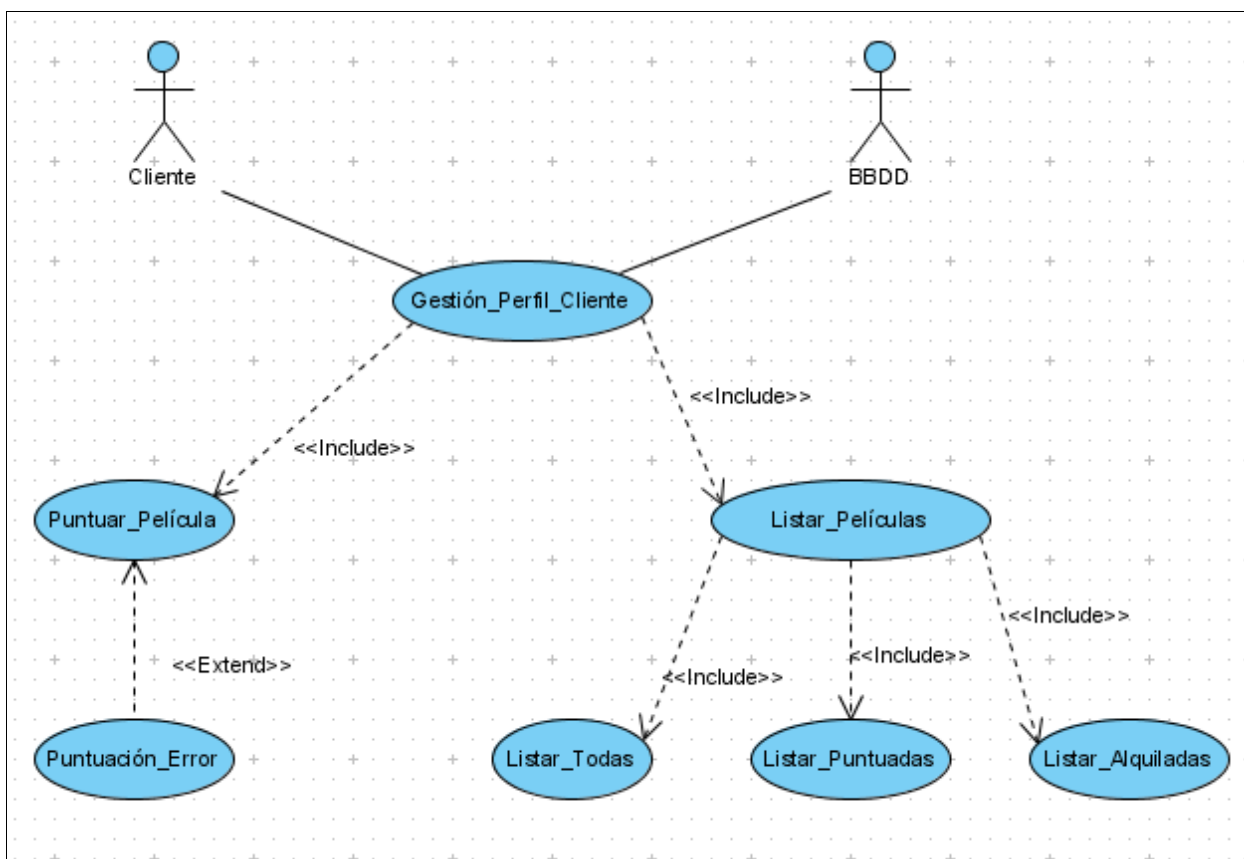


Figura 3.2.3. Diagrama del caso de uso Gestión_Perfil_Cliente

A continuación, se describen detalladamente cada uno de los casos de uso mostrados en las figuras anteriores:

➤ **Caso de Uso 1: Identificación**

Actores participantes: Cliente y Administrador

Condiciones de entrada: Que existan cuentas de usuario en la aplicación

Flujo de eventos:

1. El usuario (Cliente o Administrador) inicia la aplicación.
2. El sistema muestra un formulario de entrada.
3. El usuario introduce su identificador.
4. El sistema comprueba que el identificador es válido (E-1).
5. Según el identificador sea de:
 - 5.1. Cliente, entonces el usuario entra al sistema y este le muestra el *menú principal*.
 - 5.2. Administrador, entonces el usuario entra al sistema y este le muestra el *menú de administrador*.

Condiciones de salida: La contraseña ha sido comprobada.

Excepciones:

E-1:El identificador introducido por el usuario no es válido. El sistema informa al usuario de dicha situación. El usuario puede intentar introducir un identificador válido de Cliente o Administrador o salir del caso de uso.

➤ **Caso de Uso 2: Actualización algoritmo**

Actores participantes: Administrador

Condiciones de entrada: Administrador identificado correctamente.

Flujo de eventos:

1. El sistema muestra el menú de administrador.
2. Administrador elige la opción *Actualizar Algoritmo de Filtrado* del menú.
3. El sistema lanza un script para actualizar el algoritmo (E-1).
4. Una vez terminada la actualización, el sistema muestra la fecha en la que se ha realizado la misma y un mensaje informativo para Administrador.

Condiciones de salida: la actualización se produce satisfactoriamente y se solicita salir del caso de uso.

Excepciones:

E-1: el script lanzado por el sistema ha fallado al actualizar el algoritmo. El sistema se lo comunica a Administrador. Administrador puede volver a intentar lanzar el script o salir del caso de uso.

➤ **Caso de Uso 3: Gestión_Datos_Cliente**

Actores participantes: Cliente y BBDD

Condiciones de entrada: Existen Cliente y BBDD.

Flujo de eventos principal:

1. El sistema muestra el menú principal.
2. Cliente elige la opción *Datos Personales* del menú principal.
3. El sistema muestra un menú con tres opciones y le pide a Cliente que elija:
 - Si Cliente elige *Ver Datos Personales*, se realiza S-1.
 - Si Cliente elige *Modificar Datos Personales*, se realiza S-2.
 - Si Cliente elige *Salir*, se termina el caso de uso.

Subflujos de eventos:

S-1: Ver_Datos

- 1.1. El sistema se comunica con BBDD (E-1) para obtener los datos personales de Cliente.
- 1.2. El sistema muestra sus datos personales a Cliente.
- 1.3. El caso de uso se inicia de nuevo.

S-2: Modificar_Datos

- 2.1. El sistema le muestra a Cliente un formulario de entrada de datos.
- 2.2. Cliente introduce su edad (E-2), su género (E-3), su profesión y su código postal en el formulario.
- 2.3. El sistema actualiza BBDD (E-1) con los nuevos datos.
- 2.4. El caso de uso se inicia de nuevo.

Condiciones de salida: Se solicita la salida del caso de uso.

Excepciones:

E-1: Ha habido un error al comunicarse el sistema con BBDD. El sistema informa a Cliente de dicha situación. El caso de uso se inicia de nuevo.

E-2: Cliente ha introducido una edad inválida. El sistema informa a Cliente de dicha situación. Usuario puede intentar introducir de nuevo una edad o salir del caso de uso.

E-3: Cliente ha introducido un género inválido. El sistema informa a Cliente de dicha situación. Usuario puede intentar introducir de nuevo un género o salir del caso de uso.

➤ **Caso de Uso 4: Gestión_Perfil_Cliente**

Actores participantes: Cliente y BBDD

Condiciones de entrada: Existen Cliente y BBDD.

Flujo de eventos principal:

1. El sistema muestra el menú principal.
2. Cliente elige la opción *Perfil de Usuario* del menú principal.
3. El sistema muestra un menú con tres opciones y le pide a Cliente que elija:
 - Si Cliente elige *Puntuar Película*, se realiza S-1.
 - Si Cliente elige *Listados*, se realiza S-2.
 - Si Cliente elige *Salir*, se termina el caso de uso.

Subflujos de eventos:

S-1: Puntuar_Película

- 1.1. El sistema se comunica con BBDD (E-1) para obtener el listado de las películas.
- 1.2. El sistema muestra a Cliente el listado anterior.
- 1.3. Cliente elige la película que desea puntuar.
- 1.4. El sistema le muestra a Cliente un formulario de entrada de datos.
- 1.5. Cliente introduce la puntuación (E-2) que le quiere dar a la película en el formulario.
- 1.6. El sistema actualiza BBDD (E-1) con los nuevos datos.
- 1.7. El caso de uso se inicia de nuevo.

S-2: Listar_Películas

- 2.1. El sistema muestra un menú con cuatro opciones y le pide a Cliente que elija:
 - Si Cliente elige *Listar Todas*, entonces se realiza S-2-1.
 - Si Cliente elige *Listar Puntuadas*, entonces se realiza S-2-2.
 - Si Cliente elige *Listar Alquiladas*, entonces se realiza S-2-3.
 - Si Cliente elige *Salir*, entonces se inicia el caso de uso.

S-2-1: Listar_Todas

- 2.1.1. El sistema se comunica con BBDD (E-1) para obtener todas las películas.
- 2.1.2. El sistema muestra un listado con todas las películas a Cliente.
- 2.1.3. El caso de uso se inicia de nuevo.

S-2-2: Listar_Puntuadas

- 2.2.1. El sistema se comunica con BBDD (E-1) para obtener las películas ya puntuadas por Cliente.
- 2.2.2. El sistema muestra un listado con las películas obtenidas a Cliente.
- 2.2.3. El caso de uso se inicia de nuevo.

S-2-3: Listar_Alquiladas

- 2.3.1. El sistema se comunica con BBDD (E-1) para obtener todas las películas que ha alquilado Cliente pero todavía no ha puntuado.
- 2.3.2. El sistema muestra un listado con las películas obtenidas a Cliente.
- 2.3.3. El caso de uso se inicia de nuevo.

Condiciones de salida: Se solicita la salida del caso de uso.

Excepciones:

E-1: Ha habido un error al comunicarse el sistema con BBDD. El sistema informa a Cliente de dicha situación. El caso de uso se inicia de nuevo.

E-2: Cliente ha introducido una puntuación inválida. El sistema informa a Cliente de dicha situación. Cliente puede intentar introducir de nuevo una puntuación o salir del caso de uso.

➤ **Caso de Uso 5: Recomendaciones**

Actores participantes: Cliente y BBDD

Condiciones de entrada: Existen Cliente y BBDD. Cliente tiene puntuaciones de películas en su perfil.

Flujo de eventos:

1. El sistema muestra el menú principal.
2. Cliente elige la opción *Obtener Recomendaciones* del menú principal.
3. El sistema se comunica con BBDD (E-1) y obtiene las puntuaciones hechas por Cliente.
4. El sistema calcula las películas recomendadas para Cliente en base a sus puntuaciones previas utilizando el algoritmo de filtrado colaborativo implementado.
5. El sistema obtiene de BBDD (E-1) la información de las películas recomendadas.
6. El sistema muestra a Cliente una lista con las películas recomendadas.

Condiciones de salida: Cliente visualiza correctamente su lista de películas recomendadas y solicita salida del caso de uso.

Excepciones:

E-1: Que haya error al comunicarse el sistema con la base de datos. El sistema informa a Cliente de dicha situación. El caso de uso se inicia de nuevo.

➤ **Caso de Uso 6: Ayuda**

Actores participantes: Cliente

Condiciones de entrada: Cliente identificado correctamente.

Flujo de eventos:

1. Cliente elige *Ayuda* de la lista de opciones.
2. El sistema muestra el mecanismo de ayuda implementado a Cliente.

Condiciones de salida: Cliente visualiza correctamente la ayuda y solicita salida del caso de uso.

3.2.2.2 Escenarios

Un caso de uso es una representación abstracta, una abstracción, de una funcionalidad del sistema a realizar. La representación concreta de un caso de uso se realiza mediante la creación de uno o más **escenarios** que muestren todas las interacciones posibles entre el sistema y sus usuarios.

Un escenario esta formado por los siguientes elementos:

- Un nombre único y unívoco
- Una descripción
- Los actores participantes
- El flujo de eventos

Como se ha indicado, para cada caso de uso puede haber varios escenarios. Para nuestro proyecto se han creado y descrito una cantidad importante de casos de uso por lo que no vamos a definir todos los escenarios de cada uno de ellos sino que vamos a definir unos pocos que puedan servir como ejemplo de las principales funcionalidades que el sistema va a desarrollar: realizar puntuaciones y obtener recomendaciones.

<p>Nombre: PuntuarPeliGoldenEye</p> <p>Descripción: El usuario con identificador 89 quiere puntuar con un 4 la película de título GoldenEye</p> <p>Actores: Cliente89 y BBDD</p> <p>Flujo de eventos:</p> <ol style="list-style-type: none"> 1. El usuario entra al sistema. 2. El sistema muestra el formulario de entrada. 3. El usuario introduce 89 en el formulario. 4. El sistema valida correctamente y el usuario entra a la aplicación como <i>Cliente89</i>. 5. El sistema muestra el menú principal de la aplicación. 6. El usuario elige la opción <i>Perfil de Usuario</i> del menú. 7. El sistema muestra un menú con 3 opciones: <i>Puntuar Película</i>, <i>Listados</i> y <i>Salir</i>. 8. El usuario elige la opción <i>Puntuar Películas</i>. 9. El sistema se comunica con BBDD y obtiene todas las películas disponibles. 10. El sistema muestra una lista con las películas obtenidas. 11. El usuario encuentra en la lista la película <i>GoldenEye</i> y la elige. 12. El sistema le muestra un formulario al usuario para que introduzca la puntuación que le va a conceder a la película <i>GoldenEye</i>. 13. El usuario introduce en el formulario un 4. 14. El sistema comprueba que 4 es una puntuación válida 15. El sistema actualiza BBDD con la nueva puntuación. 16. El sistema comunica a usuario que la operación se ha realizado con éxito
--

Figura 3.2.4. Escenario PuntuarPeliGoldenEye

<p>Nombre: ObtenerRecCliente345</p> <p>Descripción: El usuario con identificador 345 quiere recibir del sistema una lista de películas recomendadas</p> <p>Actores: Cliente345 y BBDD</p> <p>Flujo de eventos:</p> <ol style="list-style-type: none"> 1. El usuario entra al sistema. 2. El sistema muestra el formulario de entrada. 3. El usuario introduce 345 en el formulario. 4. El sistema valida correctamente y el usuario entra a la aplicación como <i>Cliente345</i>. 5. El sistema muestra el menú principal de la aplicación. 6. El usuario elige la opción <i>Obtener Recomendaciones</i> del menú. 7. El sistema se comunica con BBDD y obtiene las puntuaciones hechas por el usuario. 8. El sistema calcula las películas recomendadas para el usuario en base a sus puntuaciones previas y utilizando el algoritmo de filtrado colaborativo. 9. El sistema obtiene de BBDD información de las 10 películas recomendadas. 10. El sistema muestra al usuario una lista con sus 10 películas recomendadas.
--

Figura 3.2.5. Escenario ObtenerRecCliente345

3.2.3 Diseño del Sistema

Sin duda, realizar de manera adecuada cada una de las actividades que conlleva la **Ingeniería del Software** es indispensable para la realización de un proyecto software de calidad. Por lo tanto, no se puede decir que ninguna de estas actividades sea más importante que otra. Sin embargo, si podemos decir que la actividad de diseño es la más delicada y la más laboriosa de llevar a cabo.

Es delicada porque si no se lleva a cabo correctamente se hace imposible el codificar, de manera correcta, en la actividad de implementación el modelo obtenido en el análisis del sistema, lo que puede repercutir en el desperdicio de todo el esfuerzo realizado durante las primeras actividades de la Ingeniería del Software.

Y es laboriosa porque las estrategias a seguir para conseguir que esta traducción entre modelo y código se lleve a cabo correctamente son muy diversas y bastante complejas.

Se puede decir, por tanto, que el **diseño del sistema** es la actividad de la Ingeniería del Software en la que se identifican los objetivos finales del sistema y se plantean las diversas estrategias para alcanzarlos en la actividad de implementación.

Sin embargo, el sistema no se suele diseñar de una sola vez sino que hay que diferenciar entre el diseño y estructura de los datos que se van a manejar y el diseño de la interfaz entre la aplicación y el usuario. Estas dos fases del diseño no se realizan de forma consecutiva una detrás de la otra sino que lo normal es realizarlas de manera concurrente y finalizarlas a la vez.

3.2.3.1 Diseño de los datos

La intención de esta fase del diseño software es determinar la estructura que poseen cada uno de los elementos de información del sistema, es decir, la estructura de los datos sobre los que va a trabajar. Estos elementos son:

- Las *películas*, de las que conocemos su nombre, su año de producción, su fecha de estreno, el/los géneros a los que se adscribe y la URL de su entrada en IMDB.

- Los *usuarios*, de los que conocemos su edad, su género, su profesión y su código postal además de su identificador del sistema y el número de películas que ha puntuado.
- Las *puntuaciones*, de las que conocemos el usuario que las hace, las películas que las reciben y, obviamente, el valor numérico de las mismas.
- Las *películas alquiladas* por los usuarios pero que todavía no han puntuado.

Una vez determinados cuales son los elementos de información del sistema, se deben obtener sus representaciones en forma de tablas de una base de datos. Para ello, se debe realizar primeramente un diseño conceptual de la base de datos para, posteriormente, obtener las tablas requeridas. Para realizar este diseño conceptual se utilizará el modelo Entidad-Relación.

Modelo Entidad-Relación

El modelo Entidad-Relación (también conocido por sus iniciales: E-R) es una técnica de modelado de datos que utiliza **diagramas entidad-relación**. No es la única técnica de modelado pero si es la más extendida y utilizada.

Un diagrama entidad-relación está compuesto por tres tipos de elementos principales:

- **Entidades:** objetos (cosas, conceptos o personas) sobre los que se tiene información. Se representan mediante rectángulos etiquetados en su interior con un nombre. Una *instancia* es cualquier ejemplar concreto de una entidad.
- **Relaciones:** interdependencias entre uno o más entidades. Se representan mediante rombos etiquetados en su interior con un verbo. Si la relación es entre una entidad consigo mismo se denomina *reflexiva*, si es entre dos entidades se denomina *binaria*, *ternaria* si es entre tres y *múltiple* si es entre más (muy raro).
- **Atributos:** características propias de una entidad o relación. Se representan mediante elipses etiquetados en su interior con un nombre.

En los diagramas entidad-relación también hay que tener en cuenta otros aspectos como pueden ser:

- **Entidades débiles:** son aquellas que no se pueden identificar unívocamente solo con sus atributos, es decir, necesitan de estar relacionadas con otras entidades para existir. Se representan con dos rectángulos concéntricos de distinto tamaño con un nombre en el interior del más pequeño.
- **Cardinalidad de las relaciones:** existen tres tipos de cardinalidades de una relación según el número de instancias de cada entidad que involucren:
 - *Uno a uno:* una instancia de la entidad A se relaciona solamente con una instancia de la entidad B. (1:1)
 - *Uno a muchos:* cada instancia de la entidad A se relaciona con varias de la entidad B. (1:*)
 - *Muchos a muchos:* cualquier instancia de la entidad A se relaciona con cualquier instancia de la entidad B. (*:*)
- **Claves:** cada entidad de un diagrama entidad-relación debe tener una clave, debe estar formada por uno o más de sus atributos.

Una vez conocidos los elementos que forman parte de un diagrama entidad-relación podemos empezar a desarrollar el **modelo entidad-relación**. Los pasos a seguir son los siguientes:

1. Convertir el enunciado del problema (o, como es nuestro caso, los elementos del sistema software) en un **Esquema Conceptual** del mismo.
2. Convertir este Esquema Conceptual (o EC) en uno más refinado conocido como **Esquema Conceptual Modificado** (ECM).
3. Obtener las tablas de la base de datos a partir del Esquema Conceptual Modificado.

1) Esquema Conceptual

Necesitamos convertir nuestros elementos del sistema en entidades o relaciones. De manera obvia se puede llegar a la conclusión de que *películas* y *usuarios* se convierten en las entidades *PELICULAS* y *USUARIOS* respectivamente y que *puntuaciones* se transforma en la relación *PUNTUAR* que une las dos entidades. Por su parte, *películas alquiladas* se transforma en la relación *ALQUILAR* entre *PELICULAS* y *USUARIOS*. Nuestro EC quedaría de la siguiente forma:

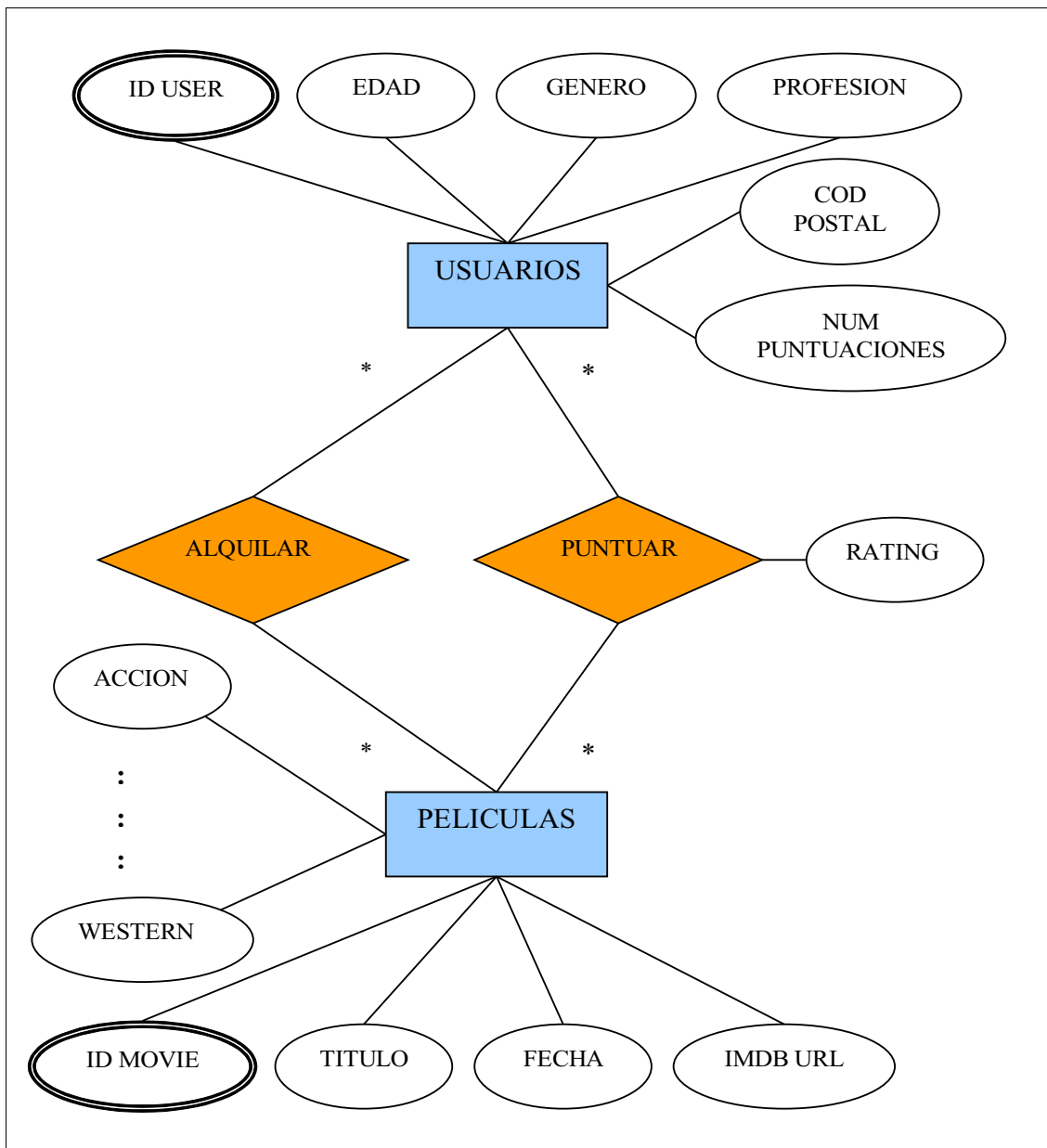


Figura 3.2.6. Esquema Conceptual

2) Esquema Conceptual Modificado

Para obtener el Esquema Conceptual Modificado debemos eliminar todas las entidades débiles, relaciones muchos a muchos y relaciones con atributos que haya en nuestro Esquema Conceptual. Por lo tanto, nuestro ECM queda como sigue:

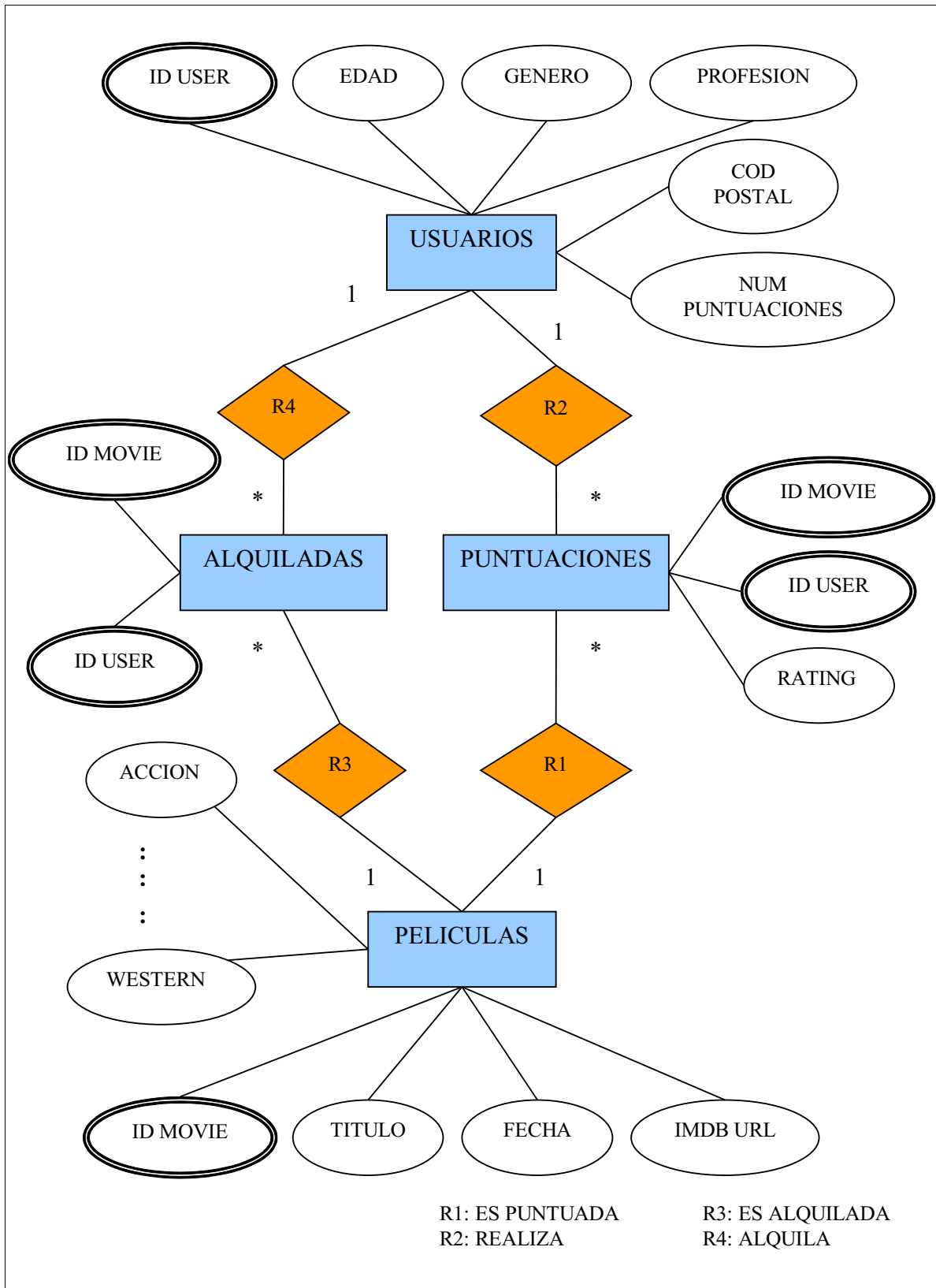


Figura 3.2.7. Esquema Conceptual Modificado

3) Tablas de la aplicación

A partir del ECM obtenido previamente podemos determinar las tablas de la base de datos, teniendo en cuenta que:

- Cada entidad del ECM se transforma en una tabla.
- Los atributos de una entidad se convierten en los campos de las tablas respectivas.

Por lo tanto, obtendremos las siguientes cuatro tablas: USUARIOS, PUNTUACIONES, PELICULAS y ALQUILADAS. A continuación se detallan cada una de estas tablas.

USUARIOS

Tabla que contiene la información sobre los usuarios de la aplicación. Esta formada por los siguientes campos:

CAMPO	TIPO	DESCRIPCIÓN	CLAVE
ID_USER	NUMBER(3)	Identificador unívoco de usuario	*
EDAD	NUMBER(3)	Edad del usuario	
GENERO	VARCHAR(1)	M si el usuario es hombre y F si es mujer	
PROFESION	VARCHAR(25)	Profesión del usuario	
COD_POSTAL	VARCHAR(5)	Código Postal del usuario	
NUM_PUNTUACIONES	NUMBER(3)	Número de películas que ha puntuado el usuario. Campo calculable.	

Tabla 3.2.1. Campos de la tabla USUARIOS

PELICULAS

Tabla que contiene la información de todas la películas de la aplicación. Esta formada por los siguientes campos:

CAMPO	TIPO	DESCRIPCIÓN	CLAVE
ID_MOVIE	NUMBER(4)	Identificador unívoco de la película	*
TITULO	VARCHAR(81)	Título de la película	
FECHA	DATE	Fecha de estreno de la película	
IMDB_URL	VARCHAR(134)	Dirección URL de la página en IMDB de la película	
DESCONOCIDO	BOOLEAN	La película es de género desconocido	
ACCION	BOOLEAN	La película es de acción	
AVENTURAS	BOOLEAN	La película es de aventuras	
ANIMACION	BOOLEAN	La película es de animación	
INFANTIL	BOOLEAN	La película es infantil	
COMEDIA	BOOLEAN	La película es de comedia	
CRIMEN	BOOLEAN	La película es de crimen	
DOCUMENTAL	BOOLEAN	La película es documental	
DRAMA	BOOLEAN	La película es de drama	
FANTASIA	BOOLEAN	La película es de fantasía	
NEGRO	BOOLEAN	La película es de género negro	
TERROR	BOOLEAN	La película es de terror	
MUSICAL	BOOLEAN	La película es musical	
MISTERIO	BOOLEAN	La película es de misterio	
ROMANTICO	BOOLEAN	La película es romántica	
CIENCIA-FICCION	BOOLEAN	La película es de ciencia-ficción	
THRILLER	BOOLEAN	La película es un thriller	
GUERRA	BOOLEAN	La película es bélica	
WESTERN	BOOLEAN	La película es un western	

Tabla 3.2.2. Campos de la tabla PELICULAS

PUNTUACIONES

Tabla que contiene la información sobre las puntuaciones que hacen los distintos usuarios de la aplicación sobre las distintas películas. Esta formada por los siguientes campos:

CAMPO	TIPO	DESCRIPCIÓN	CLAVE
ID_USER	NUMBER(3)	Identificador unívoco del usuario que realiza la puntuación	*
ID_MOVIE	NUMBER(4)	Identificador unívoco de la película puntuada	*
RATING	NUMBER(1)	Valor de la puntuación. Debe ser 1, 2, 3, 4 o 5. Cualquier otro valor es erróneo	

Tabla 3.2.3. Campos de la tabla PUNTUACIONES

ALQUILADAS

Tabla que contiene la información sobre las películas alquiladas pero todavía no puntuadas por parte de cada usuario de la aplicación. Esta formada por los siguientes campos:

CAMPO	TIPO	DESCRIPCIÓN	CLAVE
ID_USER	NUMBER(3)	Identificador unívoco del usuario que alquila la película	*
ID_MOVIE	NUMBER(4)	Identificador unívoco de la película alquilada	*

Tabla 3.2.4. Campos de la tabla ALQUILADAS

3.2.3.2 Diseño de la Interfaz

En esta fase del diseño del sistema software se define cual va a ser la **apariciencia visual de la aplicación**, es decir, se define la interfaz visual entre el usuario y la aplicación. Sin duda, realizar un buen diseño de la interfaz resulta primordial ya que está debe presentarse atractiva al usuario de la aplicación pero a la vez le debe de resultar fácil de entender y trabajar sobre ella.

Esta importancia se acrecienta aun más en nuestro caso ya que la interfaz de nuestro proyecto es una **interfaz web** y la usabilidad web es un tema candente, foco de cierta controversia. La cual se debe a que para las aplicaciones con interfaces web no existe una guía de estilo estándar como la puede haber, por ejemplo, para desarrollar interfaces para aplicaciones de escritorio de **Windows XP** y que resulten, a la vez, atractivas y familiares. Cada programador, desarrollador o diseñador web debe definir su propia guía de estilo y procurar que, en base a ella, la interfaz resultante consiga unas cotas dignas de atractivo visual, familiaridad y facilidad de uso.

En este apartado vamos a definir nuestra **guía de estilo** y describir y analizar las **metáforas** empleadas:

A) Definir Guía de Estilo

Antes de ponerse a diseñar una interfaz de usuario, se debe definir el estilo de la misma. Esto es de vital importancia cuando el diseño va a ser compartido entre varios diseñadores, ya que ayuda a mantener la coherencia interna de la interfaz.

Sin embargo, en contra de lo que pueda parecer en un principio, también es de mucha utilidad definir una guía de estilo cuando solo hay un diseñador encargado de la interfaz. Esto se debe a varias razones:

- A veces es posible que mantener la coherencia y consistencia de una interfaz, si esta es muy grande o muy ambiciosa, sea algo complicado incluso si solo hay un diseñador si no tiene una base.
- El diseñador primitivo puede, por las más diversas razones, abandonar el diseño y es de utilidad para sus sustitutos contar con una guía de estilo predefinida para no tener que empezar de cero otra vez. Lo mismo puede aplicarse si no es el diseñador original el que se encarga del mantenimiento o la actualización de la interfaz.

Quedando demostrada la utilidad del uso de guías de estilo podemos pasar a definir las reglas, normas y recomendaciones que contendrá la guía de estilo de nuestra interfaz:

– **Fuentes**

- *Tipo:* Verdana
- *Tamaño:*
 - Cabecera: 18px
 - Párrafo: 12px
 - Formulario: 12px
 - Otras circunstancias: 9px
- *Color:*
 - En el Cuerpo: Blanco
 - En el Pie: Gris oscuro

- **Enlaces:** sin subrayado. Fondo Negro. Los enlaces externos se abren en una nueva ventana o pestaña del navegador. Los internos a la aplicación lo hacen en la misma ventana o pestaña. Los colores empleados para los enlaces son:

- *Al exterior:* Fucsia
 - Sobre: Fucsia
 - Visitado: Morado
- *Internos:* Azul claro
 - Sobre: Rojo
 - Visitado: Amarillo

– **Colores de fondo**

- *Cabecera:* Gris claro
- *Cuerpo:* Verde oscuro
- *Pie:* Gris claro

- **Logotipo:** arriba a la izquierda (en la Cabecera). Esta presente en todas las páginas del sitio web y siempre sirve como enlace de vuelta a la página de inicio.

- **Ayuda:** abajo a la izquierda (en el Pie). Esta presente en todas las páginas del sitio web.

- **Opciones:** en menús desplegados.


B) Metáforas

Una metáfora es el empleo de un objeto con un significado o dentro de un contexto diferente al habitual. Al diseñar una interfaz gráfica, la utilización de metáforas resulta muy útil ya que permiten al usuario, por comparación con otro objeto o concepto, comprender de una manera más intuitiva las diversas tareas que la interfaz permite desarrollar.

Al igual que pasa en el ámbito de la literatura o la lingüística, para que una metáfora cumpla con su cometido, el desarrollador de la aplicación y el usuario final de ésta deben tener una base cultural similar. Es muy posible que el uso de un icono de manera metafórica sea entendido de una manera por el usuario occidental y de otra bien distinta por un usuario de extremo oriente. Hay que intentar, por lo tanto, que las metáforas empleadas sean lo más universales posibles para que así sean comprendidas a la perfección por la mayor parte del público potencial.

Las aplicaciones de escritorio de **Windows** suelen seguir la **Guía de Estilo XP** y utilizan una serie de metáforas con las que el usuario está plenamente familiarizado (por ejemplo, una lupa con un signo '+' en su interior establece que la función del icono es, inequívocamente, la de realizar un aumento de zoom). En el mundo de las aplicaciones web también existen una cantidad de metáforas de amplia difusión como puede ser, por ejemplo, el celebre *carrito de la compra* que emplean casi todos los comercios online.

Pero las metáforas no solo dependen del tipo de aplicación (escritorio o web) sino también del ámbito de la misma. Por ejemplo, el carrito de la compra es una metáfora conocida por todos pero si nuestra aplicación no va a vender nada al usuario no resulta conveniente utilizarla ya que puede confundir. Es por ello, que se ha realizado una revisión de los sitios webs de recomendación de características similares al que se pretende realizar en este proyecto para encontrar las metáforas más comunes entre los sistemas de recomendación y se ha llegado a la conclusión de usar la metáfora siguiente:

 **Información.** El usuario, a la vista de este icono, comprende inmediatamente que si pincha sobre el obtendrá una información sobre la película a la que acompaña. Además, al colocarse encima del mismo, se mostrará una dirección web en la barra inferior del

navegador por lo que el usuario sabrá rápidamente que se trata de un enlace externo de información.

Se pensó también en utilizar una metáfora para referirse al hecho de puntuar pero pocos son los sitios que lo utilizan al no haber ninguna lo suficientemente universal y reconocible por todos por lo que al final se deshechó la idea.

3.2.4 Implementación

La implementación es la actividad final de la **Ingeniería del Software**, aquella en la que el modelo obtenido en las actividades anteriores se debe transformar en código fuente. Para ello se debe ser cuidadoso en la elección del lenguaje de programación empleado para la codificación y de la herramienta utilizada para generarla.

3.2.4.1. Tipo de arquitectura de la aplicación

En nuestro caso, vamos a desarrollar una sistema de recomendación con una **arquitectura cliente/servidor** y una interfaz web de comunicación con los usuarios. El funcionamiento de las arquitecturas de este tipo es sencilla: la aplicación se encuentra en un servidor central al que los usuarios acceden a través de un software cliente, en nuestro caso un navegador web. Una vez que ha accedido a la aplicación, el usuario realiza peticiones que el servidor tiene que atender para generar una respuesta comprensible para el cliente.

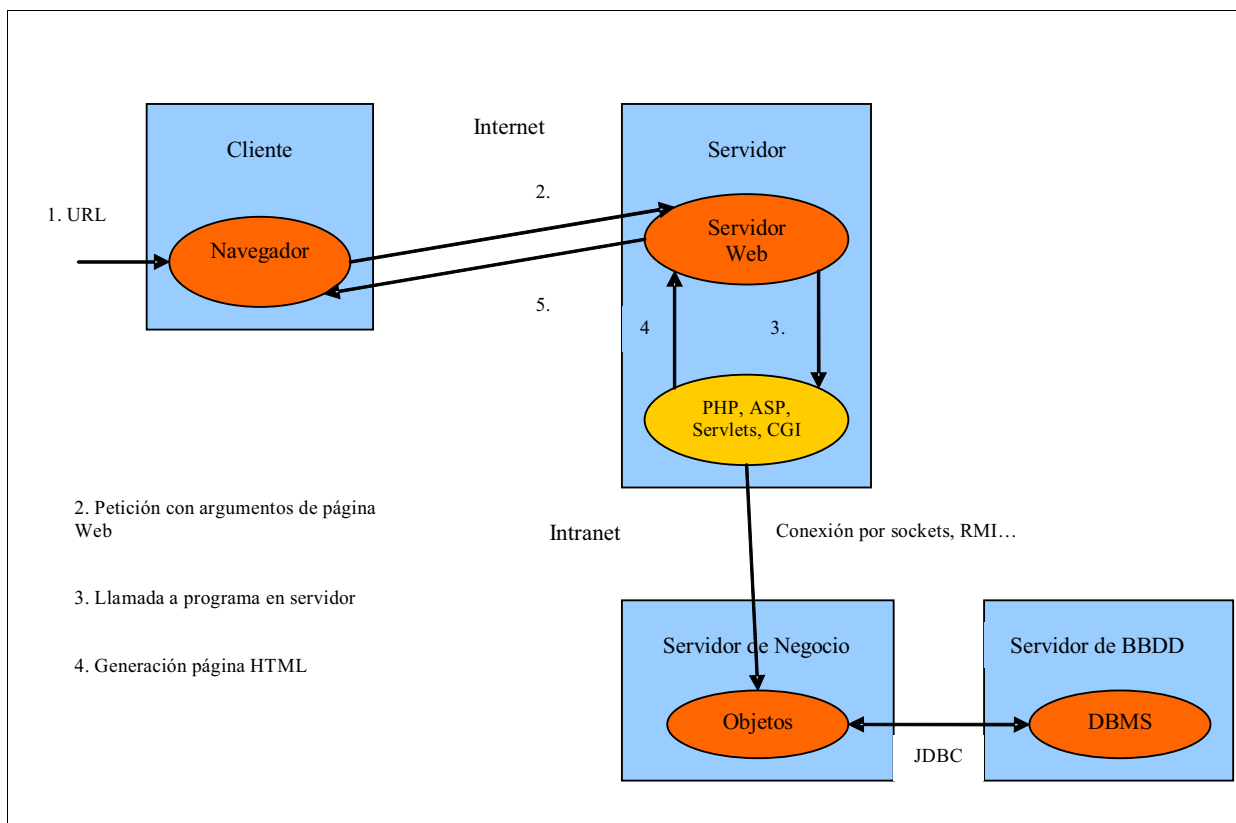


Figura 3.2.8. *Arquitectura Cliente/Servidor*

Una arquitectura cliente/servidor libera, por lo tanto, al usuario final de la aplicación de tener que instalarla en su máquina y consigue que cada usuario solo pueda acceder a la información que le corresponde. Además, este tipo de arquitectura, gracias a su diseño modular, es fácilmente escalable y ampliable tanto en nuevos clientes como en servidores añadidos.

3.2.4.2. Lenguajes de programación utilizados

Resulta obvio ante la arquitectura y el funcionamiento previsto de nuestra aplicación que el uso de **HTML** simple y llano no es adecuado sino que se necesita otro lenguaje capaz de generar contenido dinámico desde el servidor de manera transparente al usuario final. Existen varias alternativas para realizar esto, desde Perl a ASP (Active Server Pages) o JSP (Java Server Pages) pasando por el uso de CGIs (Common Gateway Interface). Sin embargo, finalmente, nos hemos decantado por el uso de **PHP**.

PHP, acrónimo de **PHP Hypertext Preprocessor**, es un lenguaje de programación interpretado, que se ejecuta del lado del servidor y genera contenido dinámico a petición del cliente. Es un lenguaje que tiene una importante serie de ventajas sobre otros lenguajes que realizan funciones parecidas como son las siguientes:

- Es libre, abierto y multiplataforma.
- Sintaxis similar a lenguajes estructurados como C.
- Capacidad de conexión con múltiples gestores de bases de datos.
- Cuenta con mecanismos para trabajar con ficheros, tratar textos, generar imágenes de manera dinámica y tratar documentos XML.
- Esta ampliamente documentado.
- Cuenta con un gran número de extensiones y módulos.
- No requiere declaración de variables.

Estas características le hacen ideal para nuestros propósitos:

- El cliente solicita cualquier funcionalidad.
- El servidor, mediante PHP, conecta con nuestra base de datos en Access y obtiene los datos pertinentes.
- También mediante PHP realiza los cálculos y acciones que sean necesarios sobre

esos datos.

- Finalmente, genera el código XHTML adecuado y se lo presenta al cliente de manera transparente.

Con PHP es suficiente para satisfacer las funcionalidades que debe presentar la aplicación a sus usuarios. Sin embargo, para realizar una implementación de la interfaz web adecuada se hace necesario el uso de otros dos lenguajes de programación: **CSS** y **Javascript**.

CSS, acrónimo de **Cascade Style Sheets**, es un lenguaje formal que ayuda a separar la estructura interna de un documento de su presentación externa. Las etiquetas de estilo CSS pueden presentarse tanto dentro de un documento HTML (encerradas dentro de las etiquetas `<style type="text/css"></style>` en la cabecera) como en un documento aparte (con extensión .css) al que el documento HTML se encarga de llamar cuando es necesario. De esta última manera no solo se consigue separar la estructura de la presentación sino que se consigue la centralización del estilo ya que una sola hoja de estilos CSS puede ser invocada por distintas páginas de la aplicación web lo que ayuda de manera muy importante al mantenimiento de la coherencia y consistencia del diseño de la aplicación.

En nuestra aplicación, el uso de hojas de estilo CSS es algo ineludible ya que así se consigue que las sentencias PHP del servidor generen, simplemente, el código XHTML necesario para responder a la petición del cliente sin entrar en temas del diseño o visualización de esta respuesta, de lo que se encargará el estilo CSS predefinido.

Por su parte, **Javascript**, lenguaje interpretado de sintaxis similar a lenguajes como Java o C que se ejecuta del lado del cliente, ayuda a PHP de otra manera: filtrando los datos de las peticiones de los clientes, dejando realizar la petición al servidor solo cuando estos son válidos. Si los datos son erróneos informan al cliente de su error mediante mensajes de error o alerta.

Al igual que ocurre con CSS, el código Javascript puede ir incrustado dentro del documento HTML (entre las etiquetas `<script type="text/javascript"></script>` en el cuerpo o la cabecera) o estar almacenado en ficheros aparte (con extensión .js) y ser invocados por el documento. Para nuestra aplicación, tanto para los estilos CSS como para el código Javascript, nos hemos decantado por la segunda opción.

3.2.4.3 Herramienta de desarrollo

Para generar código XHTML, CSS, PHP y Javascript no hace falta ninguna herramienta o entorno específico de desarrollo ya que con un simple editor de textos se pueden escribir las sentencias y etiquetas y guardar el resultado con la extensión correspondiente. Sin embargo, existen multitud de herramientas de desarrollo que facilitan de manera enorme esta tarea de codificación. Las hay tanto libres y gratuitas (como pueden ser **Quanta Plus** o **NVU**) como propietarias. Para este proyecto hemos elegido una de las de este último grupo, que además es la más popular de ellas y la que mejores prestaciones ofrece: **Macromedia Dreamweaver MX 2004**.

Dreamweaver es una herramienta con capacidad **WYSIWYG** (What You See Is What You Get) que permite crear y trabajar sobre documentos de HTML, PHP, CSS, Javascript, ASP, Java, C#, JSP, VisualBasic y otros muchos lenguajes de manera sencilla además de proporcionar numerosas plantillas ya prediseñadas. Dispone de previsualizador en distintos navegadores, vistas de código, de diseño e híbrida, validador de código W3C y otras muchas funcionalidades que la hacen una herramienta de desarrollo muy potente para entornos web.

3.2.4.4 Actualización del algoritmo de filtrado

La base primordial para desarrollar un sistema de recomendación colaborativo es contar con un buen **algoritmo de filtrado colaborativo**. Para ello es necesario refinarlo conforme los clientes vayan mejorando sus perfiles puntuando nuevas películas. En la aplicación final, este refinamiento o actualización debería realizarse de manera inmediata cada vez que hubiera una nueva puntuación. Sin embargo, al centrarse este proyecto en realizar un modelo prototipal, no contempla esta funcionalidad sino que el refinamiento se realizará de manera periódica a petición del administrador del servicio.

Para realizar esto se lanzará un **servlet de Java** en el servidor que se encargará de actualizar el algoritmo. Esta actualización requerirá de un tiempo importante para realizarse ya que se deben recalculan los grupos de vecinos, es por ello que se ha optado por la utilización de un servlet que se ejecute en segundo plano y devuelva el resultado cuando termine antes que realizar la actualización a través de la aplicación misma.

3.2.4.5 Instalación en el servidor y funcionamiento de moviesrecommender

La instalación de la aplicación **moviesrecommender** en el servidor viene documentada paso a paso en el **Anexo I**.

Por su parte, en el **Anexo II** se encuentra disponible un **manual de usuario** donde se detalla el funcionamiento desde el punto de vista del cliente de moviesrecommender.

CAPITULO 4.

CONCLUSIONES.

Este proyecto nació con la idea de desarrollar un *Sistema de Recomendación* que sirva para videoclubs o webs de venta de películas con la intención de tener un sistema centralizado que permitiera a sus clientes dar puntuaciones a las películas que fueran alquilando y/o comprando. En base a esas puntuaciones el sistema crearía un perfil de cada cliente, lo compararía con los perfiles de los demás clientes y ofrecería una lista de películas recomendadas entre las favoritas del grupo de clientes más afines a sus gustos. Con dicha mecánica de funcionamiento, el tipo de sistema de recomendación que mejor se adapta es, sin ninguna duda, el *Sistema de Recomendación Colaborativo*.

Los sistemas de recomendación colaborativos tienen como base el empleo de un *algoritmo de filtrado colaborativo*. Existen dos categorías dentro de estos algoritmos: los basados en usuario y los basados en ítem. Los primeros se caracterizan por utilizar toda la base de datos para realizar sus predicciones y recomendaciones. Los segundos por desarrollar un modelo de las puntuaciones de los clientes sobre los ítems.

Ante la imposibilidad de reunir un conjunto de datos reales lo suficientemente amplio, se optó por el empleo de un ejemplo de los datos disponibles en MovieLens. Este ejemplo proporciona 943 usuarios, 1682 películas y 100000 puntuaciones de los usuarios sobre las películas. Resulta, sin duda, una base de datos muy amplia para tratarla por completo como hacen los algoritmos basados en usuario por lo que finalmente se optó por implementar un *algoritmo de filtrado colaborativo basado en ítem*.

Algoritmos de filtrado colaborativo basados en ítem hay tantos como posibles combinaciones de los valores de los parámetros que los forman por lo que se hizo indispensable un estudio previo de estos algoritmos para elegir el que mejores prestaciones presentara sobre nuestra base de datos e implementarlo posteriormente.

Para este estudio implementamos un *algoritmo K-nn* para el cálculo de los usuarios más similares a cada uno de los usuarios de la base de datos y utilizamos distintos valores para los parámetros que dan forma al algoritmo como son las medidas de similitud, el tamaño de los conjuntos test y prueba del algoritmo K-nn y los algoritmos de predicción.

Una vez implementado un modelo para realizar las diversas pruebas y después de evaluar los resultados obtenidos llegamos a la conclusión de que el algoritmo de filtrado colaborativo que mejor se adapta a nuestras necesidades es el que está formado por grupos de 20 vecinos, utilizando el coeficiente coseno como medida de similitud, el algoritmo de

predicción weighted sum y unos porcentajes del 80% para el conjunto de prueba y el 20% para el conjunto de test.

Este algoritmo es el que utilizamos para la implementación del prototipo del sistema de recomendación colaborativo. Esta implementación se basa en una arquitectura cliente/servidor con una interfaz web que permite a los clientes conectarse remotamente desde su navegador web a la aplicación que se encuentra alojada en un servidor central.

Para el desarrollo de esta parte del proyecto seguimos las actividades de la Ingeniería del Software. Primero determinamos las propiedades que debe satisfacer la aplicación y las restricciones a las que se encuentra sometido. Luego se ha creado un modelo del sistema correcto, completo, consistente, claro y verificable. Finalmente hemos codificado este modelo en una versión prototipal y lo hemos instalado en el servidor.

Ha sido un trabajo arduo durante un largo tiempo pero no es un trabajo terminado. La aplicación ha quedado en un estado prototipal y si se pretendiera realizar una aplicación completa para el ámbito comercial existen muchas funcionalidades que deberían tenerse en cuenta como son el empleo de datos reales, la actualización inmediata del algoritmo de filtrado cada vez que se produce una variación en el perfil de un usuario o el empleo de etiquetas lingüísticas en vez de valores numéricos para la realización de puntuaciones.

Finalmente, comentar que, para el autor, la realización de este proyecto ha supuesto todo un reto ya que trataba sobre un ámbito de la informática que le era ajeno por completo y que le ha servido para poner en práctica muchas de las metodologías y habilidades adquiridas durante sus años de formación académica y otras muchas que ha tenido que adquirir durante su desarrollo.

ANEXO I.

MANUAL DE INSTALACIÓN DEL SERVIDOR.

Existen dos alternativas para poner en funcionamiento el software **moviesrecommender**. La primera consiste en alojar la aplicación en un servidor de hosting que de soporte al lenguaje **PHP** y al gestor de bases de datos **MS Access**. La otra alternativa, la cual ha resultado la elegida, consiste en montar nuestro propio servidor. En este anexo se especificará paso a paso como realizar esta operación en el entorno de un sistema operativo **Windows XP Professional Service Pack 2**.

Consideraciones previas y material necesario

Las únicas consideraciones previas que han de tenerse en cuenta son que durante todo este manual se ha supuesto que la unidad principal de disco duro es **C:**, que la unidad principal de disco flexible es **D:** y que los diferentes archivos necesarios se instalan en los directorios indicados.

Todo el material necesario para instalar y dejar operativo el servidor se encuentra disponible en el CD que acompaña a esta memoria. Vaya a **D:\moviesrecommender** y compruebe que en el directorio se encuentran los siguientes archivos:

- apache_2.2.3-win32-x86-no_ssl.ins
- apace-tomcat-5.5.20.ins
- bbdd.zip
- httpd.conf
- jdk-6-windows-i586.exe
- mod_jk.dll
- moviesrecommender_files.zip
- php5_2.zip

Si es así podemos proceder a la instalación de nuestro servidor inmediatamente. Si falta algún archivo o alguno de ellos se encuentra dañado pongase en contacto con el responsable de la aplicación para subsanar el percance.

Paso 1: Instalar Apache

Apache es un servidor HTTP de código abierto y multiplataforma desarrollado por la Apache Software Foundation, en cuya web (<http://www.apache.org>) se pueden conseguir la

última versión del servidor, sus múltiples módulos de desarrollo y ampliación y toda la documentación necesaria para su correcto funcionamiento. Se trata, con diferencia, del más popular de los servidores HTTP de la actualidad.

a) Ya dispone de la versión 2.2.3 o superior de Apache:

Dirigase a **Paso 2**.

b) No dispone de Apache o lo hace de una versión anterior a la 2.2.3:

Si dispone de una versión anterior sería conveniente que la eliminara y siguiera leyendo este paso para instalar esta versión, que es la que nos asegura que el servidor va a ser montado correctamente. Si no dispone de ninguna versión de Apache siga leyendo.

Vaya a **D:\moviesrecomender** y ejecute el archivo instalador **apache_2.2.3-win32-x86-no_ssl.ins**. Le aparecerá una pantalla de bienvenida como la siguiente:



Figura 1. *Bienvenida del instalador de Apache*

Pulse el botón **Next**. Aparecerá la pantalla de aceptación de la licencia del producto (Figura 2). En ella hay dos botones de radio, seleccione el de la confirmación de aceptación y pulse el botón **Next**.



Figura 2. Aceptación de la licencia de Apache

La siguiente pantalla que aparecerá ante usted es una pantalla de información sobre el servidor HTTP Apache (Figura 3). Una vez leído, pulse el botón **Next**.



Figura 3. Lectura de información sobre Apache

Una vez realizado este paso nos encontramos ante una de las pantallas más importantes de la instalación: la de la información del servidor (Figura 4). Se deben introducir el nombre del dominio, el nombre del servidor y el e-mail del administrador. Usted puede rellenar estos campos con los datos por defecto que se muestran en la figura o introducir los suyos propios. Si realiza esta segunda acción, no olvide apuntar los nuevos datos ya que le serán de utilidad en siguientes pasos de esta instalación.



Figura 4. Introducir información del servidor

En la misma pantalla hay dos botones de radio: el de arriba permite la utilización como servicio del servidor Apache a todos los usuarios del computador; el de abajo, por su parte, permite solo su utilización al usuario actual. Lo recomendado es seleccionar el la primera opción. Una vez hecho esto pulse el botón **Next**.

La siguiente pantalla (Figura 5) nos ofrece dos tipos de instalación: típica y personalizada por el usuario (custom). Se recomienda elegir la primera opción. Una vez hecho esto pulse el botón **Next**.



Figura 5. Elegir tipo de instalación

Una vez elegido el tipo de instalación llega el momento de elegir el directorio donde se va a instalar Apache (Figura 6). Pulse sobre el botón Change... e introduzca el siguiente directorio: **C:\Apache Software Foundation\Apache2.2**. Una vez elegido el directorio de instalación pulse el botón **Next**.

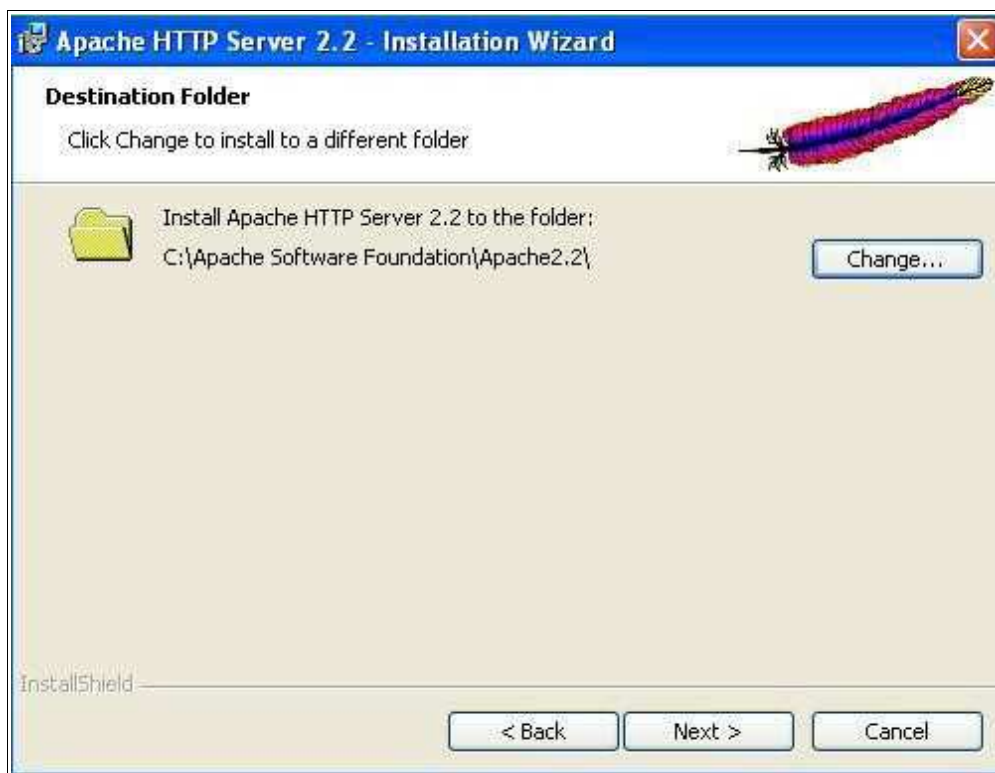


Figura 6. *Seleccionar directorio de destino*

La siguiente pantalla que aparecerá será una de confirmación (Figura 7). Si esta seguro de realizar la instalación pulse sobre el botón **Install**, si quiere cancelar la instalación pulse **Cancel** y si no esta seguro de alguno de los pasos anteriores pulse **Back** para realizar los cambios oportunos en los datos introducidos en las pantallas anteriores.

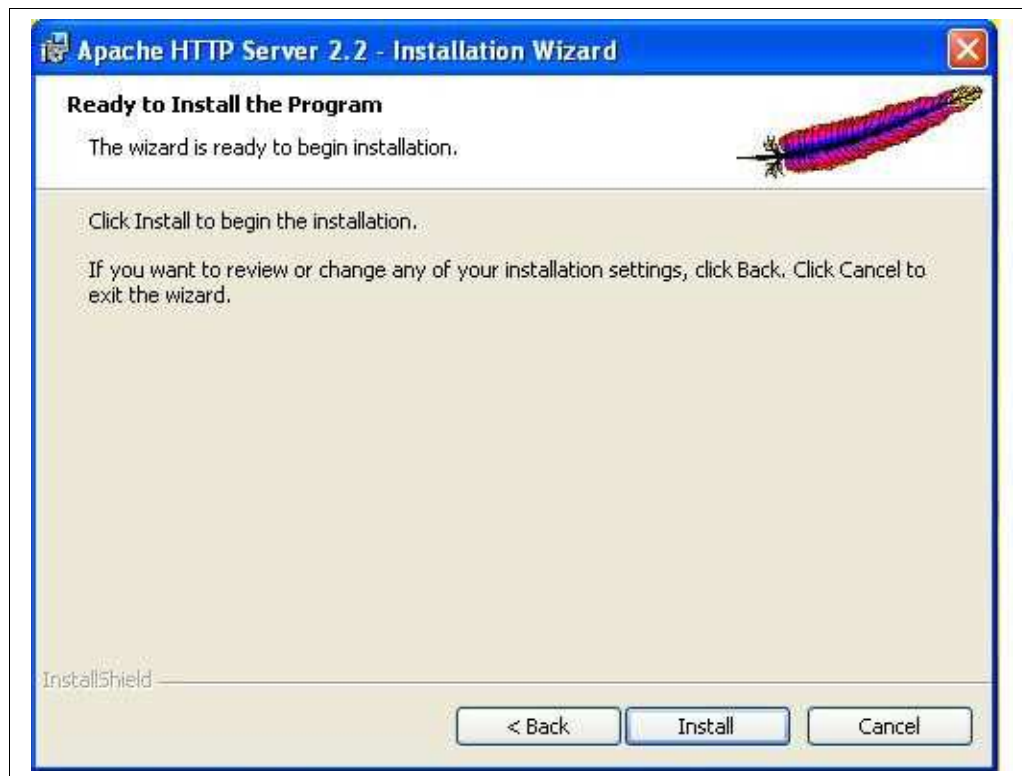


Figura 7. Confirmar la instalación

Si ha pulsado **Install** le aparecerá una pantalla donde se mostrará el progreso de la instalación (Figura 8).

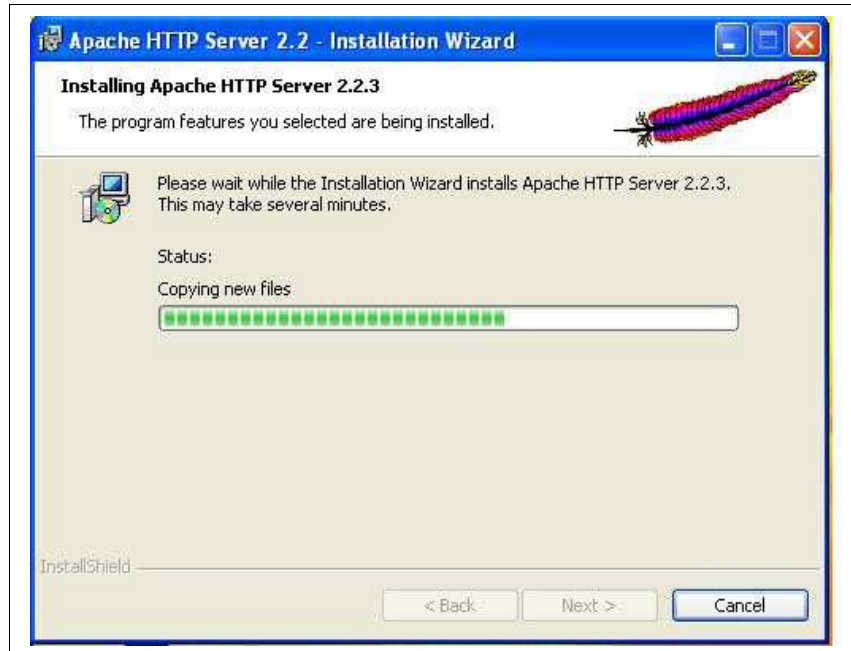


Figura 8. Progreso de la instalación

Una vez terminada la instalación, se le mostrará una pantalla (Figura 9) confirmando que la instalación se ha realizado de manera correcta y exitosa.



Figura 9. Instalación finalizada correctamente

Una vez pulsado el botón **Finish** podremos comprobar que la instalación ha sido un éxito con la aparición del siguiente icono en la barra de inicio:



Figura 10. *Icono de Apache 2.2*

Paso 2: Instalar PHP

La instalación del lenguaje PHP es mucho menos laboriosa y costosa en tiempo que la del servidor Apache.

a) Ya dispone de la versión 5.2 de PHP

Dirigase a **Paso 3**.

b) Dispone de una versión anterior o no dispone de ninguna

Vaya a **D:\moviesrecommender**, descomprima el archivo **php5_2.zip** y extraiga los archivos que se encuentran en su interior en **C:\php5_2**.

Paso 3: Configurar Apache y PHP

Una vez llegados a este punto, tanto Apache como PHP se encuentran instalados en la máquina pero la aplicación no puede ser puesta en funcionamiento debido a que no están conectados entre si. Para conseguir esta conexión entre el servidor y el lenguaje se hace necesario modificar el archivo de configuración de Apache para insertar a PHP como un módulo del servidor.

Lo primero antes de llevar a cabo esta configuración es asegurarse de que Apache esta detenido y, si no lo esta, detenerlo (Figura 11).



Figura 11. Parar la ejecución de Apache

Ahora sustituya el archivo de configuración **httpd.conf** que se encuentra en la carpeta **conf** del directorio donde este instalado Apache por el que se encuentra en **D:\moviesrecommender**. Los siguientes pasos a realizar dependerán de las circunstancias que le hayan llevado a este Paso 3:

a) Ha llegado aquí realizando los dos pasos anteriores:

La configuración ha terminado. Dirigase a **Paso 4**.

b) Ha llegado aquí sin realizar ninguno de los pasos anteriores:

Deberá ir a la carpeta donde tenga instalado PHP, abrir con cualquier editor de texto el archivo **php.ini** y buscar las siguientes líneas:

```
    ; Directory in which the loadable extensions (modules) reside.  
    extension_dir = "c:\php5_2\ext"
```

Sustituya la ruta marcada en negrita por la que corresponda a la carpeta **ext** dentro del directorio donde tenga instalado PHP y guarde los cambios.

Abra **httpd.conf** de con un editor de texto cualquiera y busque las siguientes líneas:

```
# ServerRoot: The top of the directory tree under which the server's  
# configuration, error, and log files are kept.  
#  
# Do not add a slash at the end of the directory path. If you point  
# ServerRoot at a non-local disk, be sure to point the LockFile directive  
# at a local disk. If you wish to share the same ServerRoot for multiple  
# httpd daemons, you will need to change at least LockFile and PidFile.  
#  
ServerRoot "C:/Archivos de programa/ Apache Software Foundation/ Apache2.2"
```


Sustituya el valor de **ServerRoot** por la ruta donde se encuentre instalado Apache teniendo en cuenta que dentro del archivo **httpd.conf** las barras van invertidas con respecto a las barras normales de las rutas de Windows (es decir, si hay una “\” deberá transformarla en una “/” al copiarla en httpd.conf).

Una vez realizado este cambio busque las siguientes líneas:

```
#LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule php5_module "c:/php5_2/php5apache2_2.dll"
#LoadModule proxy_module modules/mod_proxy.so
```

Sustituya el valor de **php5_module** por la ruta donde se encuentra el archivo **php5apache2_2.dll** teniendo en cuenta la misma consideración que en el párrafo anterior. Una vez realizado el cambio busque las siguientes líneas:

```
ServerAdmin yo@localhost

#
# ServerName gives the name and port that the server uses to identify itself.
# This can often be determined automatically, but we recommend you specify
# it explicitly to prevent problems during startup.
#
# If your host doesn't have a registered DNS name, enter its IP address here.
#
ServerName localhost:80
```

Sustituya los valores de **ServerName** y **ServerAdmin** por los de su configuración de Apache si es que son diferentes.

c) Ha llegado aquí realizando sólo Paso 2:

Abra httpd.conf con un editor de texto cualquiera y busque las siguientes líneas:

```
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# Do not add a slash at the end of the directory path. If you point
# ServerRoot at a non-local disk, be sure to point the LockFile directive
# at a local disk. If you wish to share the same ServerRoot for multiple
# httpd daemons, you will need to change at least LockFile and PidFile.
#
ServerRoot "C:/Archivos de programa/Apache Software Foundation/Apache2.2"
```

Sustituya el valor de **ServerRoot** por la ruta donde se encuentre Apache teniendo en cuenta que dentro del archivo **httpd.conf** las barras van invertidas con respecto a las barras normales de las rutas de Windows (es decir, si hay una “\” deberá transformarla en una “/” al copiarla en httpd.conf).

Una vez realizado el cambio busque las siguientes líneas:

```
ServerAdmin yo@localhost

#
# ServerName gives the name and port that the server uses to identify itself.
# This can often be determined automatically, but we recommend you specify
# it explicitly to prevent problems during startup.
#
# If your host doesn't have a registered DNS name, enter its IP address here.
#
ServerName localhost:80
```

Sustituya los valores de **ServerName** y **ServerAdmin** por los de su configuración de Apache si es que son diferentes.

d) Ha llegado aquí realizando sólo el Paso 1:

Deberá ir a la carpeta donde tenga instalado PHP, abrir con cualquier editor de texto el archivo **php.ini** y buscar las siguientes líneas:

```
    ; Directory in which the loadable extensions (modules) reside.
    extension_dir = "c:\php5_2\ext"
```

Sustituya la ruta marcada en negrita por la que corresponda a la carpeta **ext** dentro del directorio donde tenga instalado PHP y guarde los cambios. Una vez realizado el cambio busque las siguientes líneas en el archivo httpd.conf de Apache:

```
#LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule php5_module "c:/php5_2/php5apache2_2.dll"
#LoadModule proxy_module modules/mod_proxy.so
```

Sustituya el valor de **php5_module** por la ruta donde se encuentra el archivo **php5apache2_2.dll** teniendo en cuenta que dentro del archivo **httpd.conf** las barras van invertidas con respecto a las barras normales de las rutas de Windows (es decir, si hay una “\” deberá transformarla en una “/” al copiarla en httpd.conf).

Paso 4: Instalar Tomcat y conexión con Apache

El administrador de la aplicación **moviesrecommender** tendrá la potestad de actualizar el algoritmo de filtrado colaborativo en el que se basan las recomendaciones que ofrece la aplicación a sus usuarios. Esta acción la realizará mediante un **servlet Java** por lo que será necesario que la máquina disponga de una máquina virtual Java, un servidor Tomcat y una pasarela entre este servidor y el servidor Apache.

a) No dispone de Java:

Si no dispone de una máquina virtual de Java instalada puede conseguirla en la dirección <http://java.sun.com/javase/downloads/index.jsp> De cualquier forma, en el cd que acompaña a esta memoria puede encontrar la release JDK 6 (en **D:\moviesrecommender\jdk-6-windows-i586.exe**). Solo tiene que ejecutar el archivo y seguir los sencillos pasos de instalación para instalar las últimas versiones de no solo el kit de desarrollo Java (**JDK**) sino el entorno de ejecución Java (**JRE**) y la máquina virtual Java.

b) Dispone de Java pero no de Tomcat:

Si tiene instalado alguna herramienta de desarrollo Java como **NetBeans**, **JBuilder** o **Eclipse**, el servidor Tomcat viene de serie con la instalación de la misma por lo que sólo deberá ponerlo en funcionamiento desde la interfaz gráfica de la herramienta.

Si no tiene ninguna de estas herramientas de desarrollo instaladas pero si dispone de Java ejecute el archivo **D:\moviesecommender\apache-tomcat-5.5.20.ins**. Le aparecerá la siguiente pantalla de bienvenida:



Figura 12. Bienvenida de Tomcat

Acepte los términos de licencia y vaya pasando pantallas hasta que llegue a la siguiente:

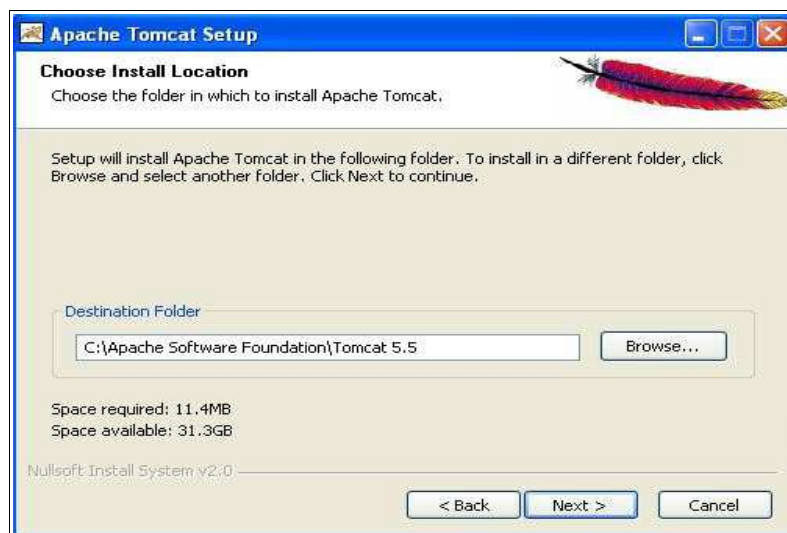


Figura 13. Carpeta destino de Tomcat

Puede elegir la ruta de destino de Tomcat que desee pero se aconseja la especificada en la figura anterior. Posteriormente llegará a la siguiente pantalla donde deberá especificar la

ruta donde se encuentre instalado el entorno de ejecución Java en su sistema:

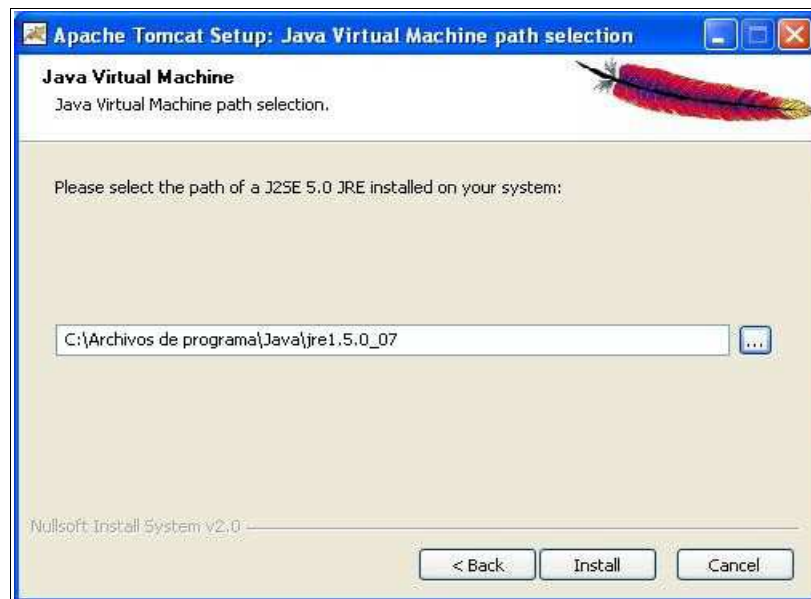


Figura 14. Especificar la ruta de JRE en su sistema

Pulse el botón **Install**, si esta seguro de que todos los pasos dados han sido correctos, para que la instalación de comienzo.

Cuando la instalación haya finalizado puede comprobar el éxito del proceso abiendo un navegador web, escribiendo <http://localhost:8080> en su barra de direcciones y comprobando que obtiene una página como la siguiente:

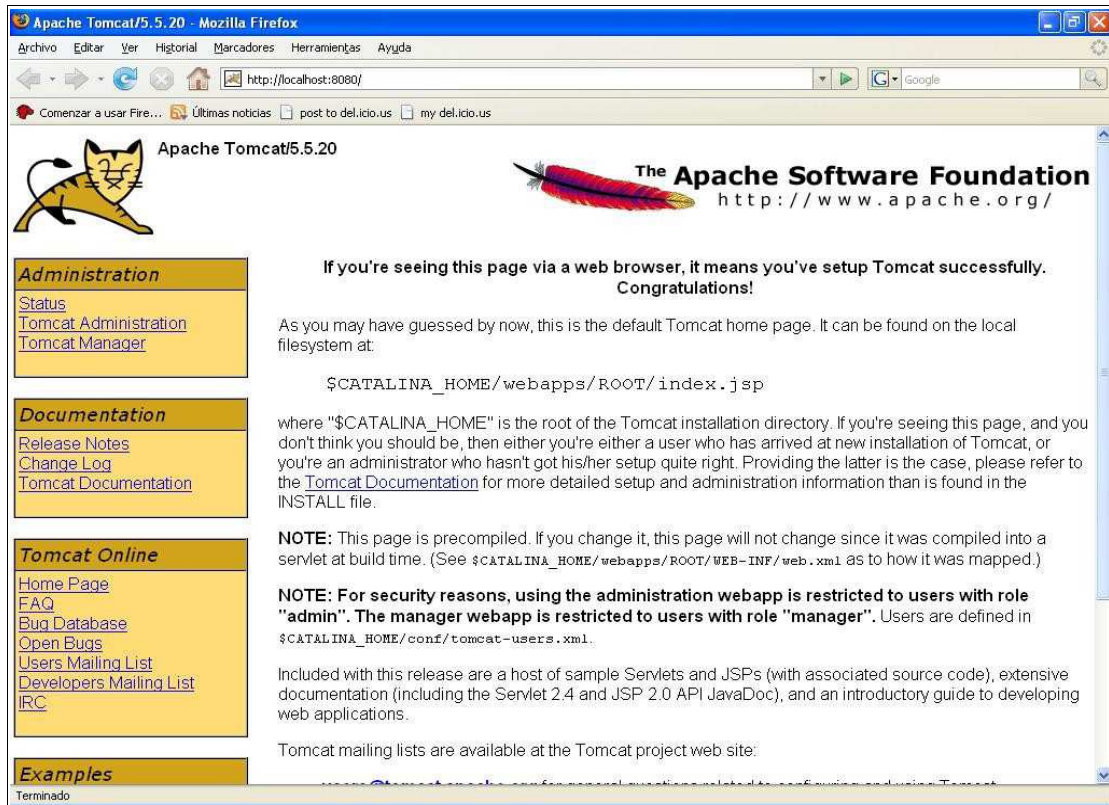


Figura 15. Página de inicio Tomcat

c) Dispone tanto de Java como de Tomcat:

Para poder ejecutar un servlet Java desde un servidor HTTP Apache se hace necesario conectarse a uno Tomcat y mandarle la petición. Para realizar esto se hace necesario un conector como el que esta disponible en `D:\moviesrecommender\mod_jk.dll`. Este conector simplemente hay que copiarlo en la carpeta **modules** de Apache y en la carpeta **conf** crear un documento **workers.properties** con el siguiente contenido:

- `workers.tomcat_home=Directorio en que instalamos Tomcat`
- `workers.java_home=Directorio en que instalamos el JDK`
- `ps=\`
- `worker.list=default`
- `worker.default.port=8009`
- `worker.default.host=localhost`

```
worker.default.type=ajp13
```

```
worker.default.lbfactor=1
```

Paso 4: Descomprimir archivos

Vaya a **D:\moviesrecommender** y descomprima el archivo **moviesrecommender_files.zip** extrayendo su contenido en el directorio **C:\moviesrecommender**. Luego vaya a su navegador Firefox (**moviesrecommender** es una aplicación web optimizada para este navegador) o en su defecto cualquier otro y copie en la barra de direcciones lo siguiente: <http://localhost/index.php> (sustituyendo “localhost” por el nombre de dominio que eligiera al instalar Apache en el **Paso 1**). Si obtiene la siguiente pantalla en su navegador la instalación habrá sido un éxito:

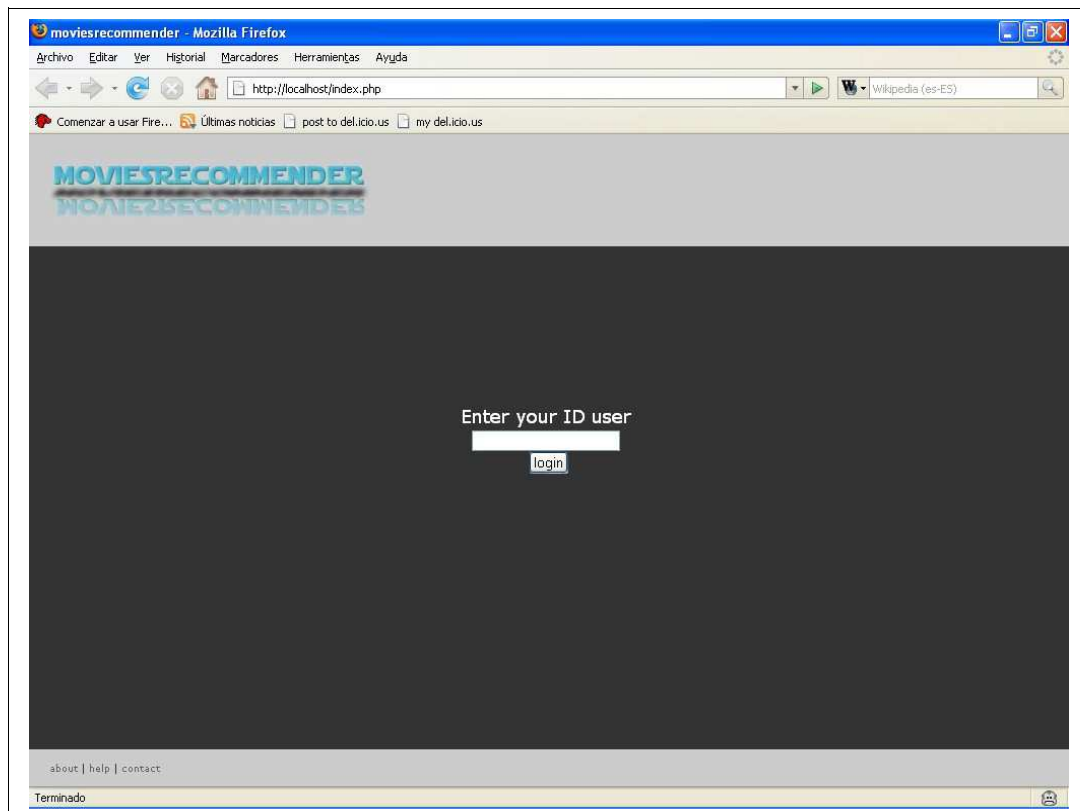


Figura 16. *Página de inicio de moviesrecommender*

Además, para que el servlet de Java sea operativo copie el archivo **.war** contenido en **C:\moviesrecommender\servlet\dist** en la carpeta **webapps** de Tomcat.

Paso 5: Conectar base de datos

La instalación del servidor web ya ha terminado pero todavía queda otro paso para que la aplicación sea operativa: conectarla con la bases de datos utilizada. Este tipo de conexión se llama **ODBC** y el primer paso es ir a **D:\moviesrecommender** y descomprimir el archivo **bbdd.zip** que en el se encuentra extrayendo el fichero *movieranks.mdb* en el directorio **C:\BasedeDatos**).

Luego debe ir a su **Panel de Control -> Herramientas administrativas -> Orígenes de datos ODBC -> DNS del Sistema** (Figura 17):



Figura 17. *Administrador de orígenes ODBC*

Pulse sobre el botón **Agregar** para añadir el origen de la primera de las bases de datos indicando que se trata de una base de datos MS Access:



Figura 18. *Crear nuevo origen de datos*

Pulse el botón **Finalizar** y en la siguiente pantalla (Figura 19) escriba **movierank** como **Nombre del origen de datos ODBC** y pulse el botón **Seleccionar**.



Figura 19. *Configurar ODBC para base de datos movierank*

Al pulsar **Seleccionar** se entra en una pantalla de selección (Figura 16) donde tiene

que buscar la base de datos principal (movieranks.mdb). Una vez localizada y seleccionada pulse **Aceptar** para volver a la pantalla anterior donde también debe pulsar **Aceptar**.



Figura 20. Seleccionar base de datos

Ahora la aplicación está conectada a la base de datos con la que debe trabajar y el servidor HTTP Apache está correctamente configurado con el lenguaje PHP como uno de sus módulos. La instalación ha terminado satisfactoriamente.

ANEXO II.

MANUAL DE USUARIO.

Este manual de usuario esta organizado como una visita guiada por la aplicación pero antes de embarcarse en ella es conveniente que el usuario tenga claros algunos aspectos:

- **moviesrecommender** es una aplicación web optimizada para su visualización en un navegador **Firefox** (a ser posible su versión más reciente, la cual se puede descargar en <http://www.mozilla-europe.org/es/>) y con una resolución no inferior a **1024x768** pixels. Si se utiliza otro navegador o una resolución inferior a la recomendada se pueden producir fallos de visualización aunque la funcionalidad de la aplicación esta completamente asegurada.
- En este punto de su desarrollo, **moviesrecommender** es sólo un prototipo que no permite el registro de nuevos usuarios ni la inserción de nuevas películas en la base de datos. Cuando la aplicación esté completamente desarrollada, éstas funcionalidades y otras muchas estarán disponibles para los usuarios.
- Debido a que los datos de la base de datos son sobre usuarios norteamericanos y se encuentran en inglés se ha decido implementar **moviesrecommender** en ese idioma. De cualquier forma se trata de un ingles muy básico y estandarizado por lo que no hace falta que el usuario sea un experto angloparlante para disfrutar de las funcionalidades que ofrece la aplicación.

Una vez aclarados los puntos anteriores puede empezar la visita guiada por la aplicación.

Lo primero que se encontrará el usuario al iniciar **moviesrecommender** (<http://localhost/index.php>) será una **página de inicio** como la que se puede observar en la figura:

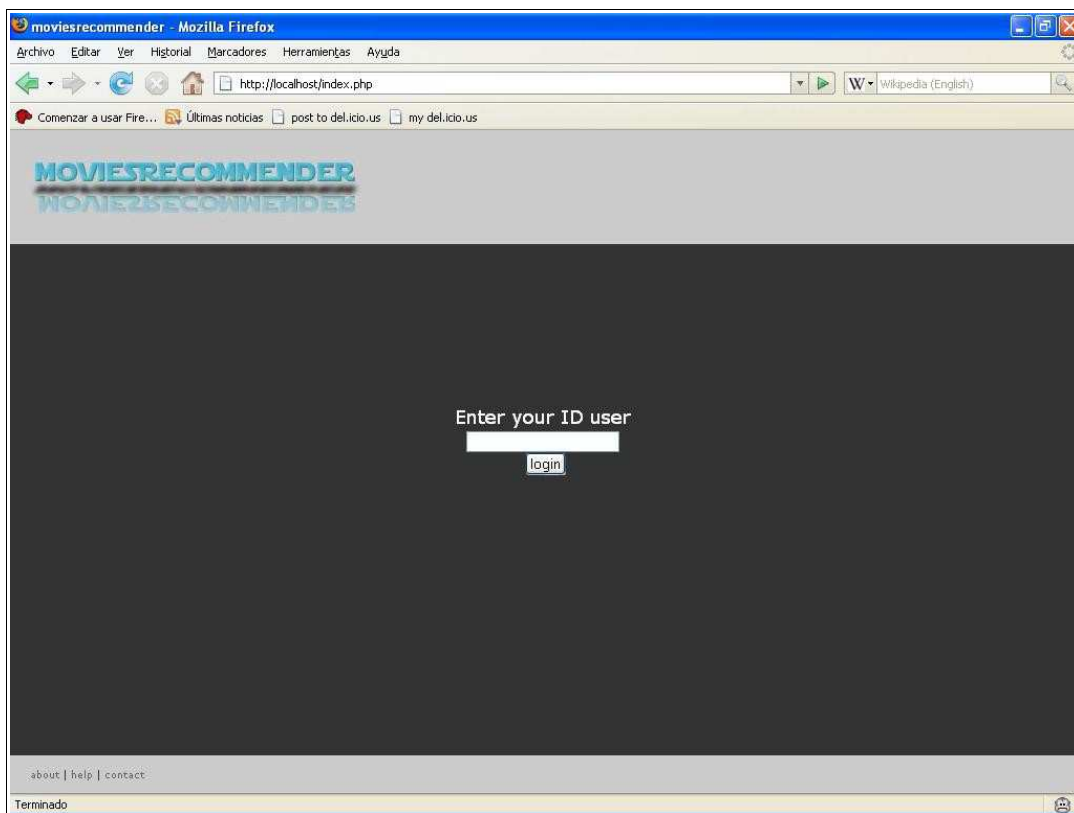


Figura 1. *Página de inicio de moviesrecommender*

El usuario deberá introducir su identificador de usuario y pulsar el botón **login**. Si pulsa el botón sin haber introducido nada se mostrará el siguiente aviso:



Figura 2. *Aviso para insertar una ID de usuario*

Si por contra pulsa el botón habiendo introducido un identificador incorrecto recibirá el siguiente aviso:



Figura 3. Aviso de ID de usuario incorrecta

Finalmente, si el usuario introduce una ID correcta accederá a su **página principal de usuario**:

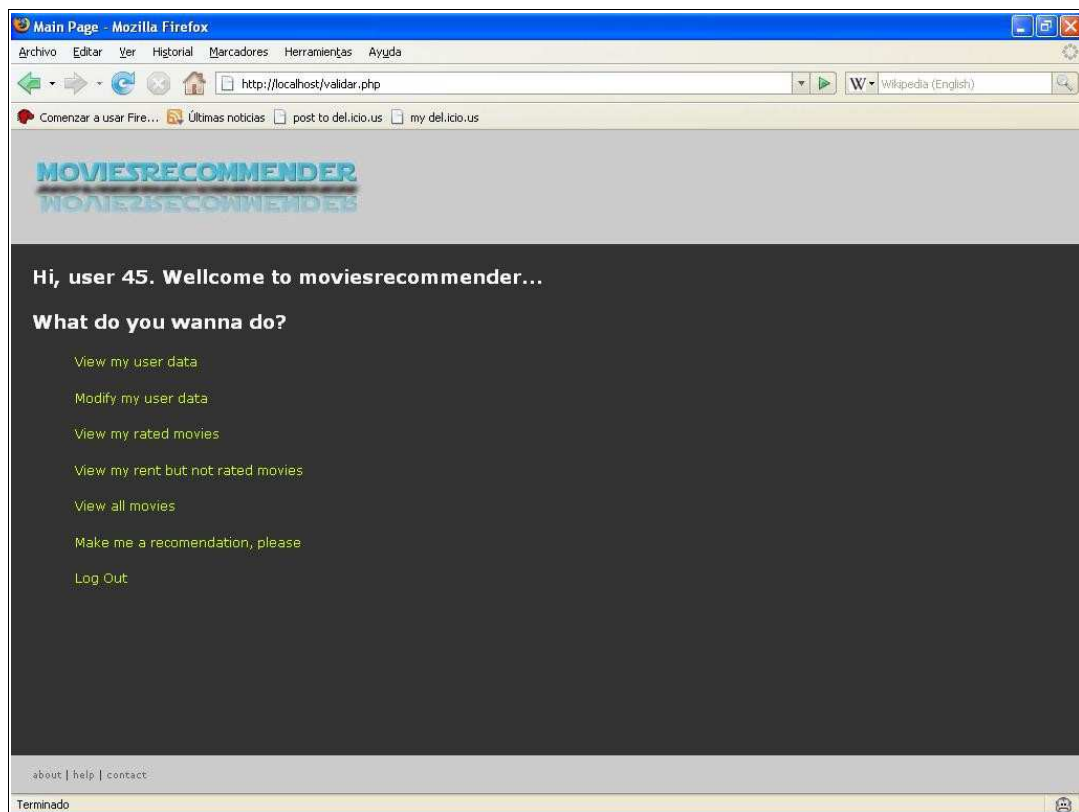


Figura 4. Página principal de usuario

En esta página principal el sistema dará la bienvenida al usuario y le ofrecerá siete posibles opciones:

1. View my user data

Si el usuario pincha sobre el enlace se desplegará un menú en el que podrá comprobar su edad, género (Male or Female), profesión, código postal y el número de películas que ha visto. Pinchando de nuevo sobre el enlace este menú se plegará.

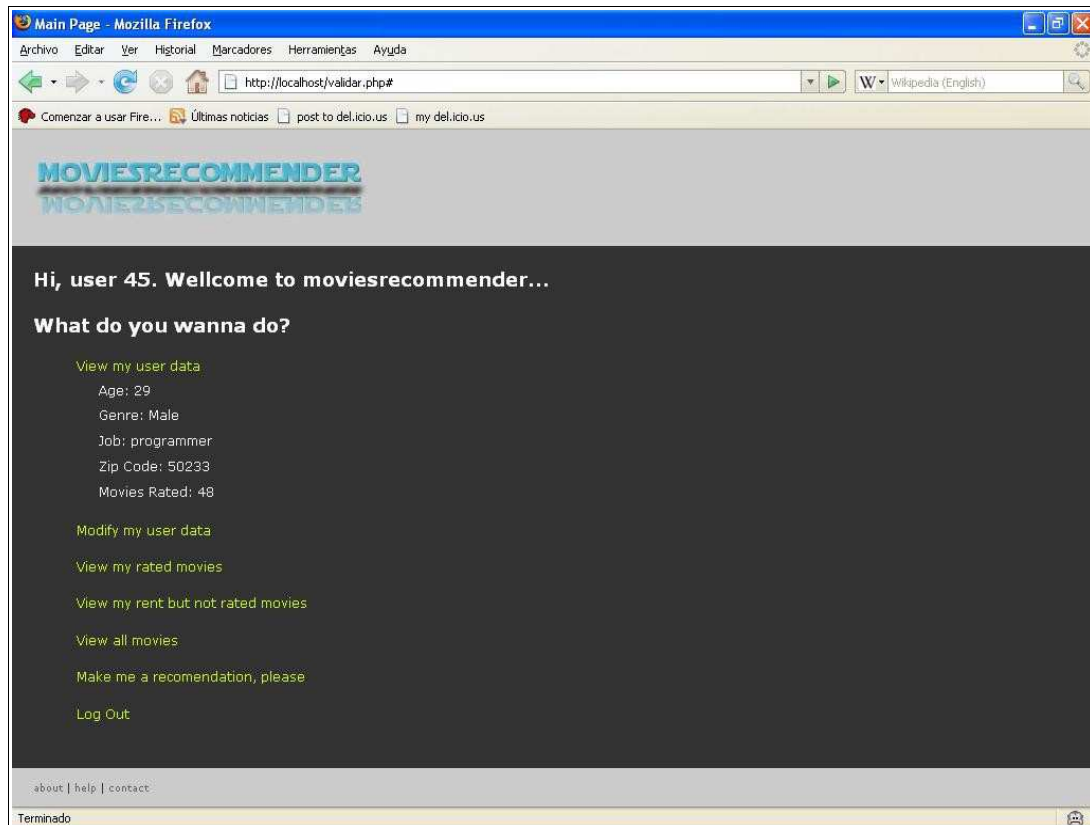


Figura 5. Datos del usuario

2. Modify my user data

Si el usuario pincha sobre el enlace se desplegará un menú con un formulario que le permitirá cambiar sus datos de usuario.

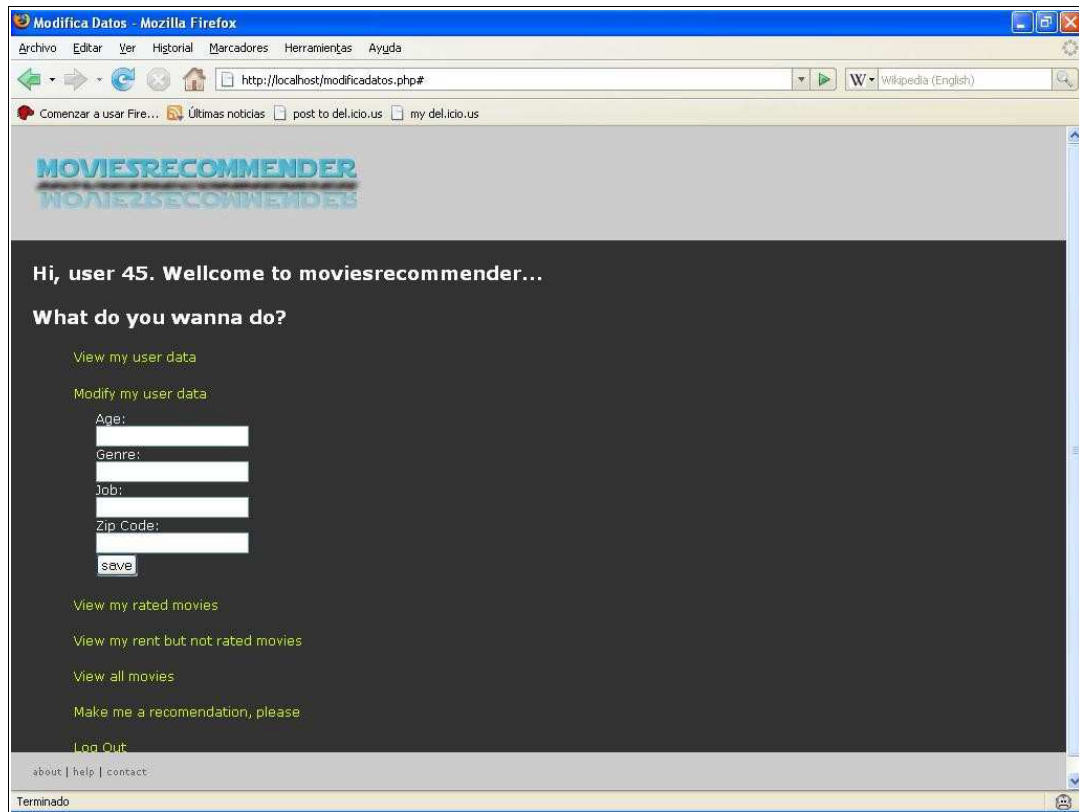


Figura 6. Modificar datos del usuario

Si envía el formulario (con el botón **save**) sin haberlo rellenado, la base de datos no se actualiza pero si introduce datos en los campos estos deben correctos. Por ejemplo, si introduce una edad incorrecta recibirá el aviso siguiente:



Figura 7. Aviso de edad incorrecta

Y si introduce algo que no sea "M" o "F" en el campo de género el aviso será como el siguiente:



Figura 8. Aviso de género incorrecto

Si los datos introducidos son correctos y la operación con la base de datos se desarrolla de manera exitosa será conducido a la siguiente pantalla:

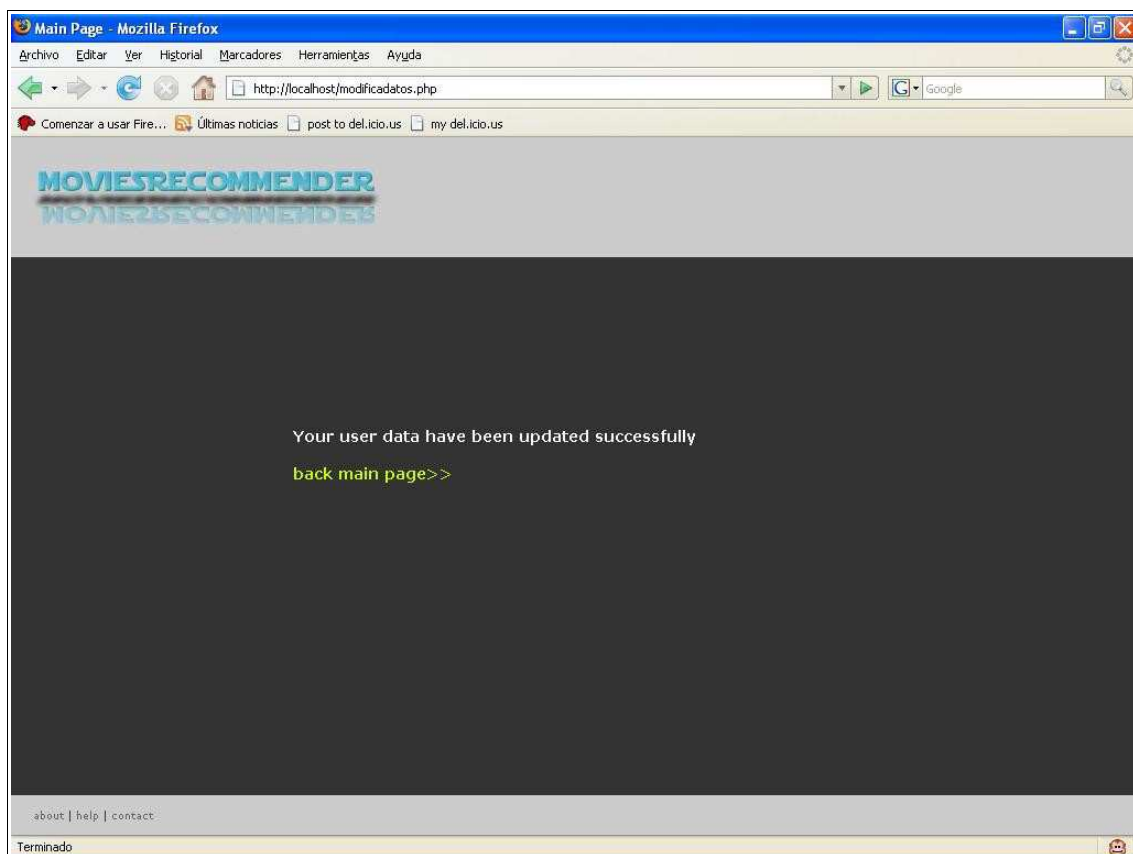



Figura 9. Éxito al modificar datos

Para volver a su página principal de usuario pulse el enlace.

3. View my rated movies

En esta opción se despliegan todas las películas que el usuario ha puntuado con su título, su año de estreno, la puntuación recibida y un icono  a la izquierda que nos dirige a la entrada en **IMDB** (<http://us.imdb.com>) de la película en cuestión.

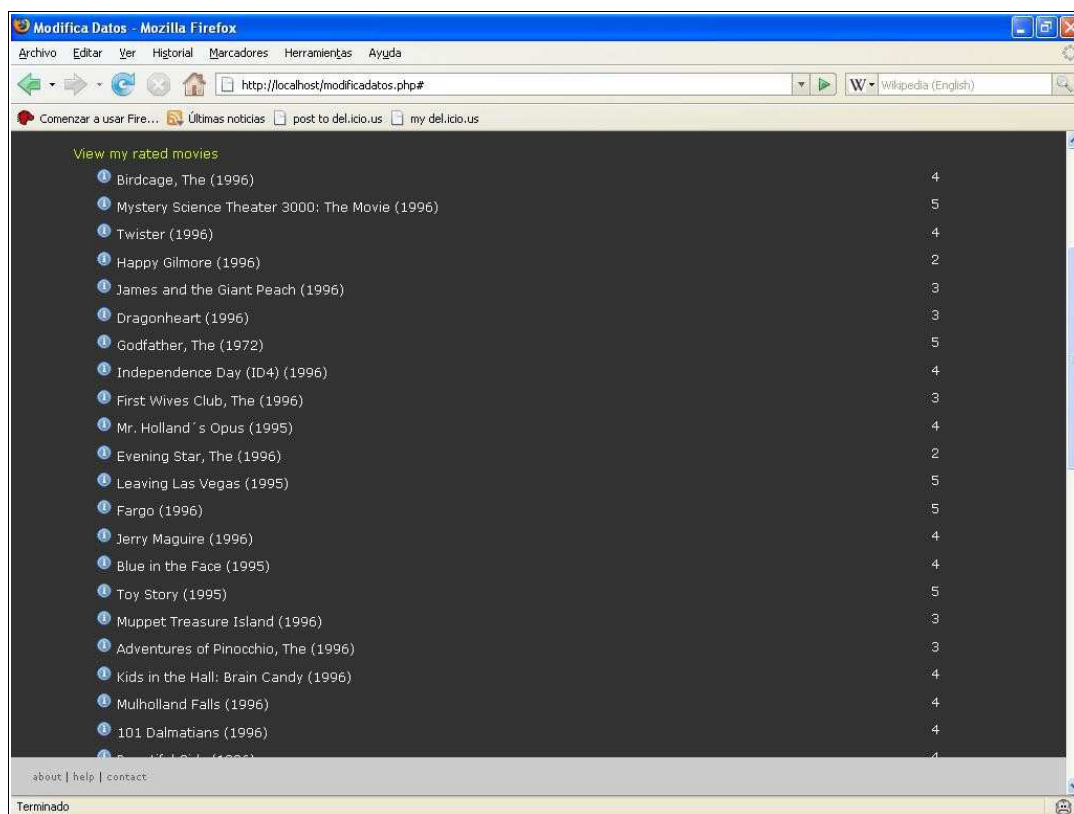


Figura 10. Películas puntuadas por el usuario

4. View my rent but not rated movies

En esta opción el usuario puede visualizar aquellas películas que ha alquilado pero que todavía no ha puntuado además de poder puntuarlas:

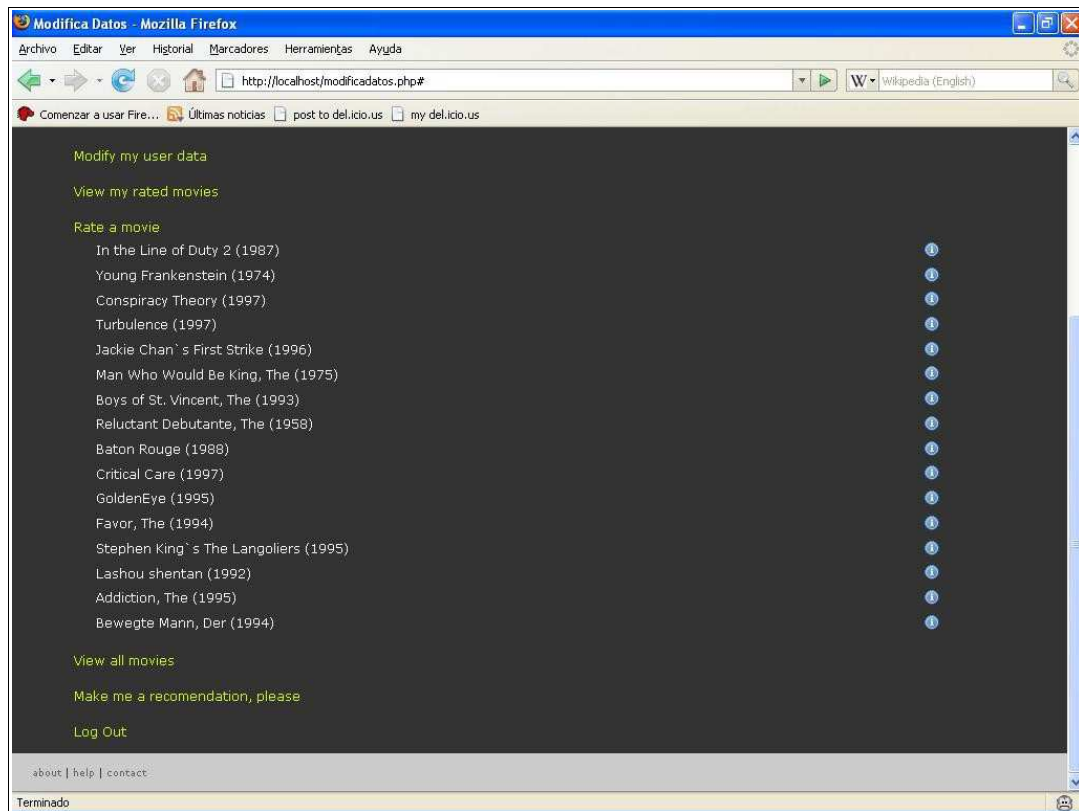


Figura 11. Películas alquiladas pero no puntuadas

Es conveniente que el usuario sea conocedor de la importancia de realizar puntuaciones ya que esto hace crecer y mejorar el perfil de usuario lo cual permite mejorar las predicciones del sistema de recomendación.

Para poder realizar una puntuación el usuario deberá pulsar sobre el título de la película y accederá a una nueva página que le permitirá realizar la acción:

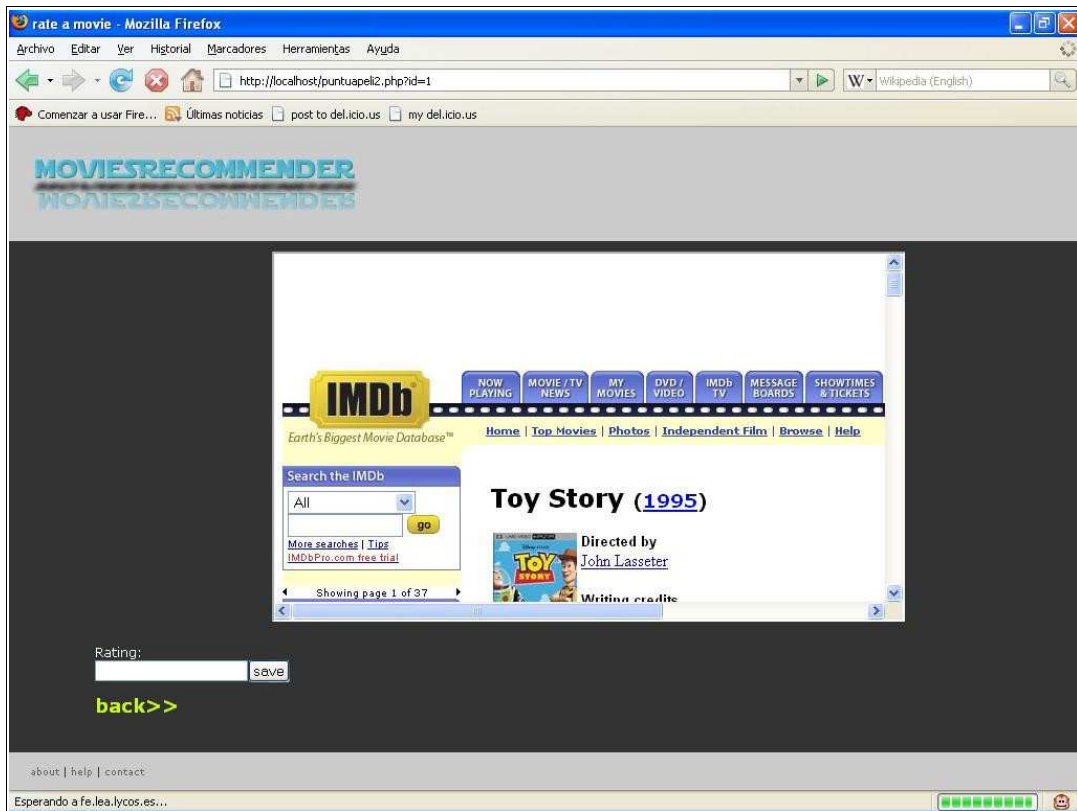


Figura 12. Página para puntuar película

En esta página el usuario encontrará una ventana o *frame* con la entrada en **IMDb** de la película (sobre la cual si pincha irá a la entrada misma) y un pequeño formulario con un campo para introducir la puntuación numérica entre 1 y 5, un botón (**save**) para aceptar esta puntuación y añadirla a la base de datos y otro botón (**back**) para volver a la página principal del usuario.

Si la operación de puntuación se realiza correctamente, el usuario será conducido a la pantalla siguiente (donde tendrá que pulsar sobre el enlace si desea volver a su página principal):

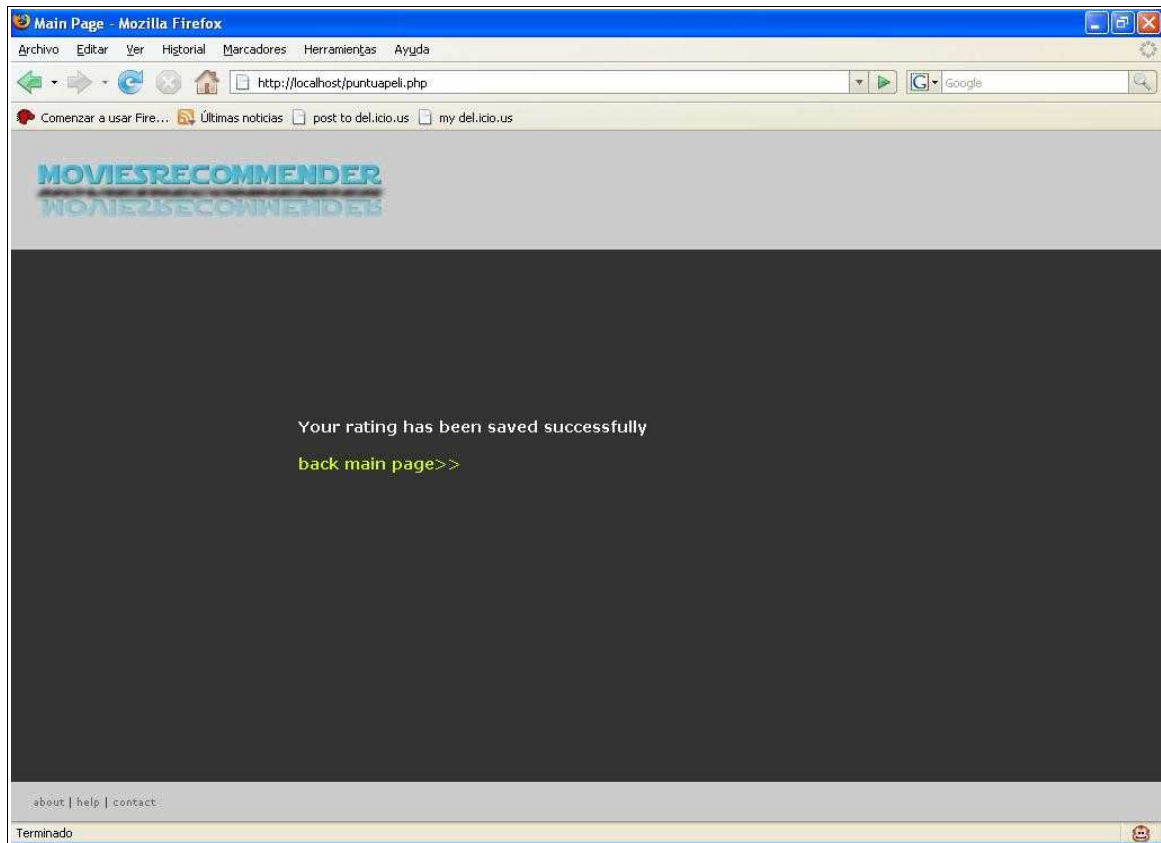


Figura 13. Éxito al realizar una puntuación

5. View all movies

Si el usuario pincha sobre este enlace se desplegará un menú con todas las películas disponibles en la base de datos:

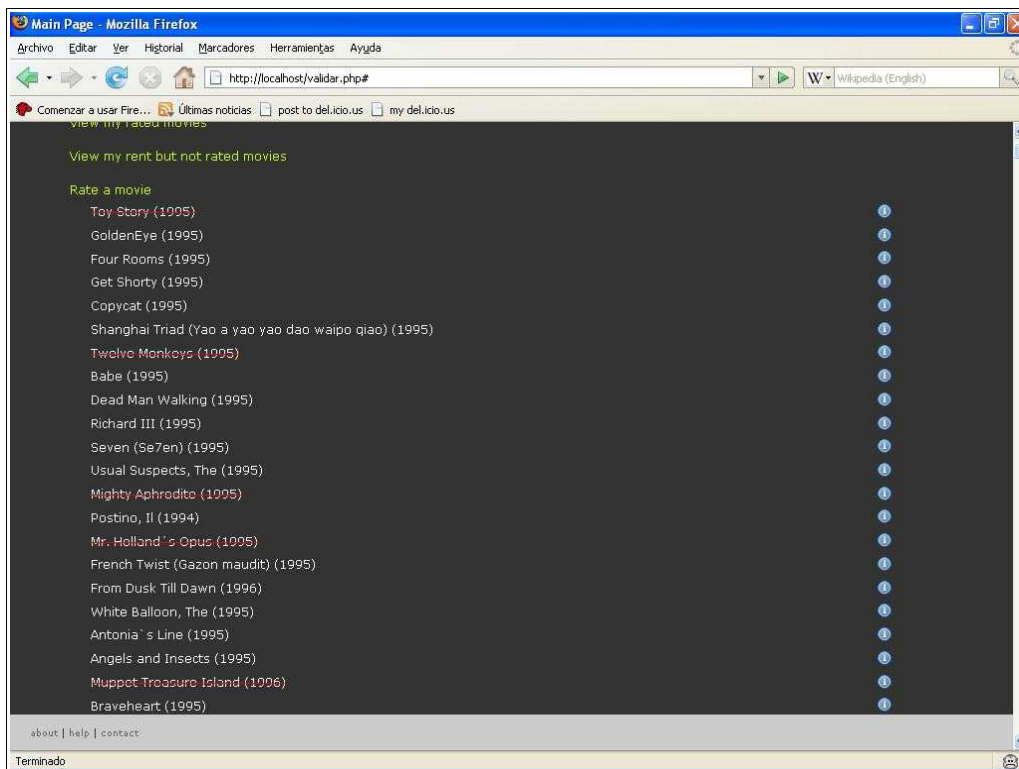


Figura 14. Todas las películas de la base de datos

Esta opción se ofrece porque es muy posible que el usuario haya visto más películas de la lista además de las que ha ido alquilando (ya sea en el cine, en la televisión o alquilada previamente a la introducción del sistema de recomendación) y es muy posible que resulte interesante conocer sus puntuaciones sobre esas películas.

Además con esta opción se da la oportunidad al usuario de en un momento dado cambiar la puntuación otorgada a una película ya sea este cambio de postura debido a cualquier circunstancia. Las películas ya puntuadas se diferencian del resto al estar tachado su título por una línea roja horizontal:



Figura 15. Película ya puntuada

La forma de puntuar es idéntica a la del apartado anterior: se pulsa sobre el título de la película y se accede a la página específica para puntuaciones operando en ella de la manera ya vista.

6. Make me a recommendation, please

Esta opción es la más importante dentro de la aplicación: cuando el usuario pinche sobre el enlace se le desplegará ante él un menú con las diez películas todavía no vistas ni alquiladas por el usuario que el sistema considera que van a ser más de su agrado ordenadas de mayor y menor y con un enlace hacia la entrada en el **IMDB** de cada de ellas para que el usuario pueda recabar toda la información que necesite:

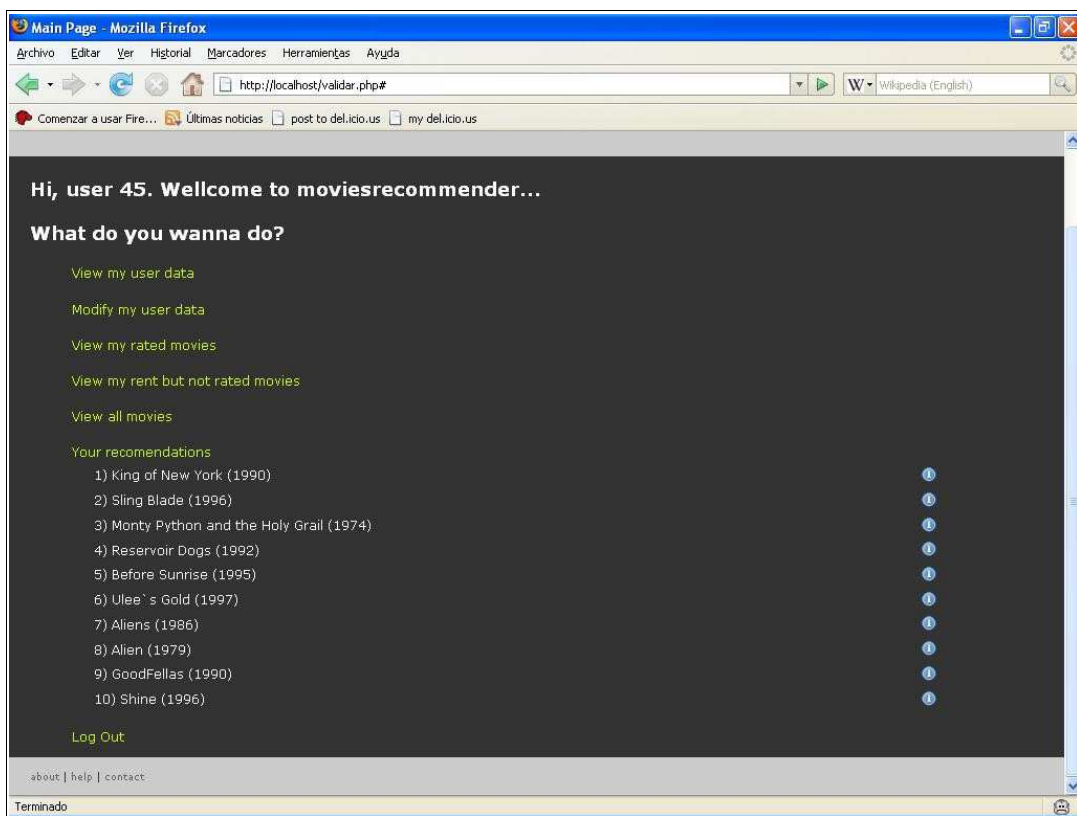


Figura 16. Películas recomendadas para el usuario dado

7. Logout

Cuando el usuario pulse sobre este enlace, terminará su sesión y volverá a la **página de inicio**.

Con lo visto hasta el momento en esta visita guiada el usuario ya debe haberse familiarizado con la funcionalidad principal de la aplicación pero todas las páginas de la aplicación integran una cabecera y un pie de página comunes que dotan de otras características a la aplicación completa y que sería conveniente que el usuario conociera.

La **cabecera** esta formada por un *banner* creado con la tipografía de la saga Star Wars y con diversos efectos de sombras y reflejos. Este banner a su vez es un enlace que lleva desde cualquier parte de la aplicación hasta la **página de inicio** terminando la sesión en curso si la hubiera.



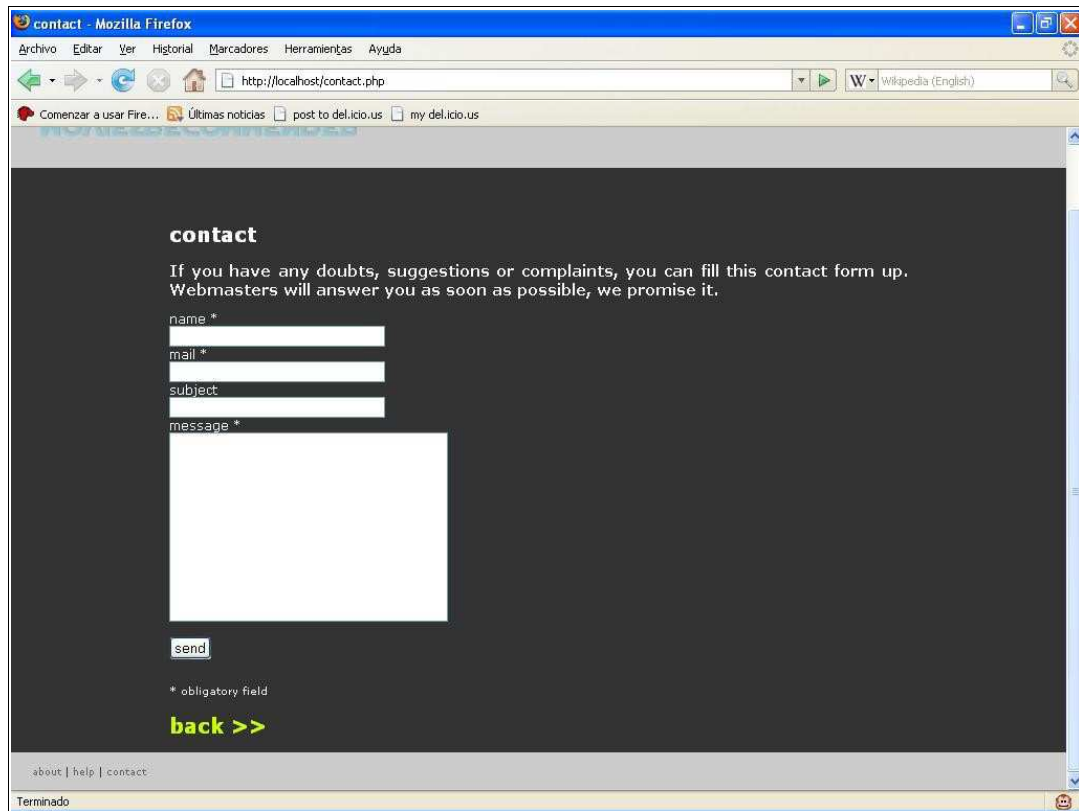
Figura 17. *Banner de la cabecera*

Por su parte, el **pie de página** nos muestra tres enlaces: uno a la página de contacto con el administrador de la aplicación (**contact**); otro a una página acerca de **moviesrecommender** (**about**) y el último a una página de ayuda (**help**):



Figura 18. *Pie de página*

La **página de contacto** presenta un formulario para que el usuario (logueado o no) pueda ponerse en contacto con la administración de la aplicación:



The image shows a screenshot of a web browser window titled "contact - Mozilla Firefox". The address bar shows "http://localhost/contact.php". The page content is as follows:

contact

If you have any doubts, suggestions or complaints, you can fill this contact form up. Webmasters will answer you as soon as possible, we promise it.

name *

mail *

subject

message *

* obligatory field

[back >>](#)

about | help | contact

Terminado

Figura 19. *Formulario de contacto*

Por su parte, la **página acerca de moviesrecommender** es una breve semblanza de la aplicación y de los autores de la misma.

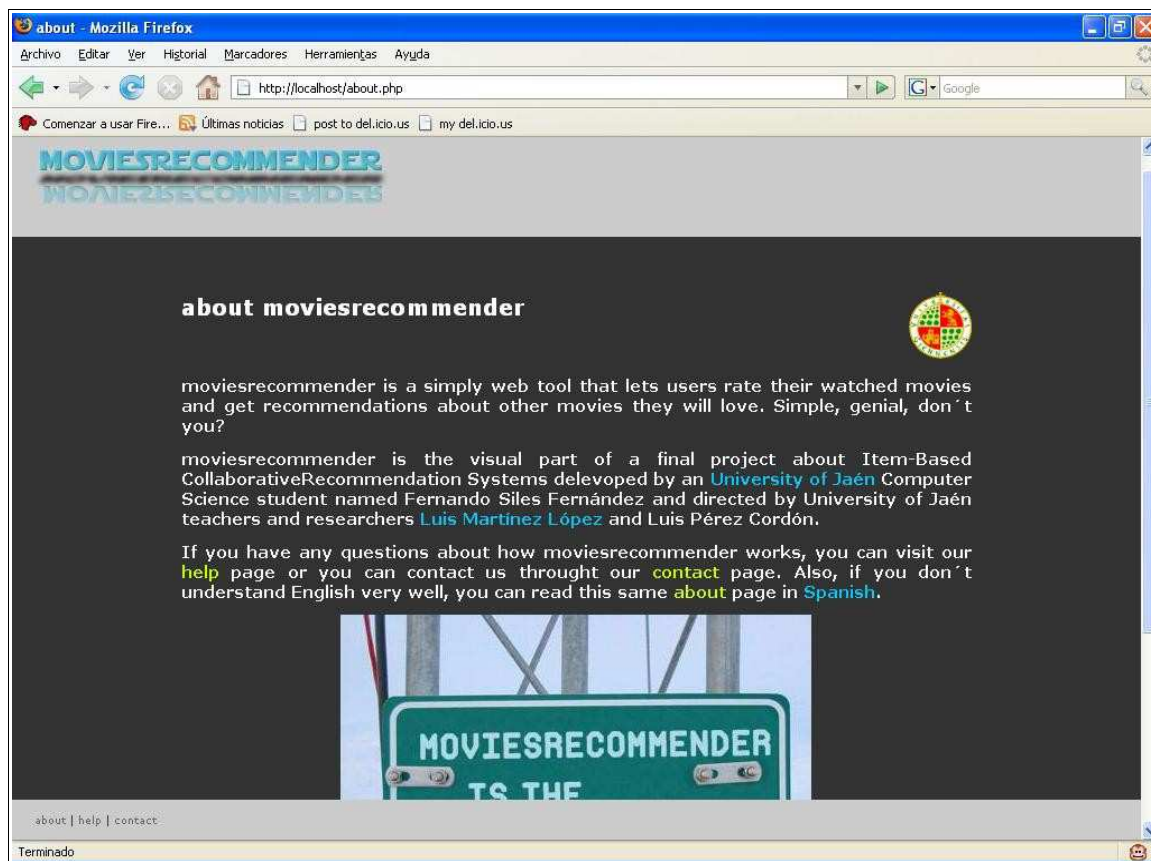


Figura 20. *Página acerca de moviesrecommender*

Esta página se puede leer también en español.

Finalmente, la **página de ayuda** se encarga de informar de que no se permiten nuevos usuarios en la aplicación y de dar un enlace en el que el usuario pueda descargar esta misma **guía de usuario** con la que aclarar cualquier duda que le surja:

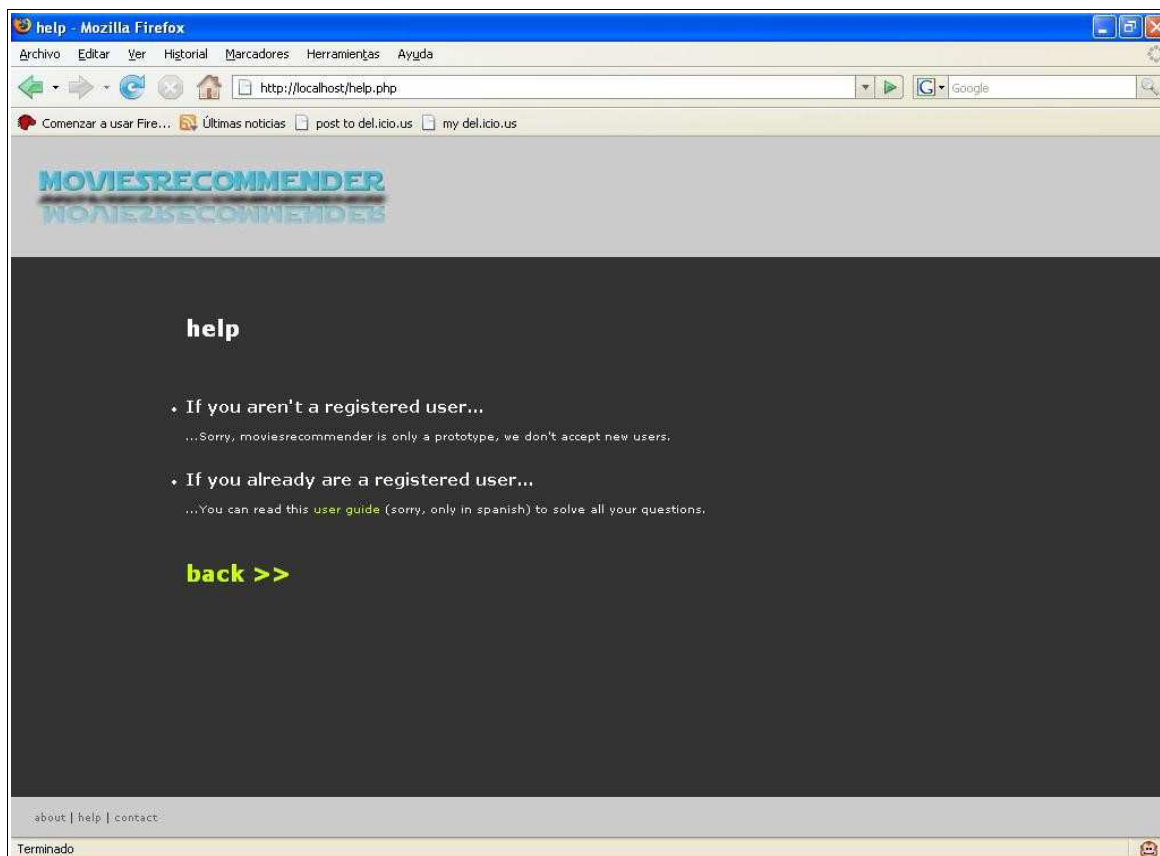


Figura 21. *Página de ayuda de moviesrecommender*

Estas tres páginas accesibles desde el pie de página tienen en común un enlace que, si no hay ninguna sesión iniciada (es decir, ningún usuario logeado), nos devuelve a la **página de inicio** o, si el usuario está logeado, a la **página principal del usuario**.

Una vez llegado a este punto el usuario debe ser capaz de manejarse con soltura por **moviesrecommender** y disfrutar de sus características y funcionalidades.

ANEXO III.

CÓDIGO FUENTE DEL MÓDULO DE PRUEBAS.

Este anexo muestra diversos trozos del código fuente de las distintas clases que se han implementado para la realización del modulo de pruebas para elegir el algoritmo de filtrado colaborativo con mejores prestaciones para nuestros intereses.

training.java

Clase que divide la base de datos en los conjuntos disjuntos de *test* y *training* y realiza el entrenamiento para calcular la matriz de similitudes.

```
import java.lang.*;
import java.util.*;
import java.io.*;

public class Training {
    Matrix m;
    int[] test;
    int med;
    registro[][] arrknn;

    /** Creates a new instance of Training */
    public Training(float x, int med, int k) {
        m = new Matrix();
        test = calConjuntoTest(m, x);
        arrknn = new registro[m.numColumnas()][k];

        arrknn = entrenar(m, test, med, k);
    }

    public int[] calConjuntoTest(Matrix m, float x) {
        int[] test;
        int aux, i=0;
        Random r;

        //x es la proporcion en tanto por 1 entre el tamaño
        //de los conjuntos de prueba y de test
        aux = Math.round(m.numFilas() * x);
        test = new int[m.numFilas()-aux];

        //elegimos aleatoriamente los integrantes del conjunto de test
        r = new Random();
        while (i<test.length) {
            aux = r.nextInt(m.numFilas());
            test[i] = aux;
            i++;
        }

        return test;
    }
}
```

```

public registro[][] entrenar(Matrix m, int[] test, int med, int k) {
    int i, j, fila, aux, aux2;
    boolean valor = true;
    float sim;
    ArrayList cola;
    ArrayList colb;
    ArrayList vecinos;
    registro[][] trainmat = new registro[m.numColumnas()][k];

    i = 0;
    while (i<m.numColumnas()) {
        //obtengo la primera columna
        cola = new ArrayList();
        fila = 0;
        valor = true;
        while (fila<m.numFilas()) {
            j = 0;
            while (j<test.length && valor == true) {
                if (fila == test[j]) valor=false;
                j++;
            }
            if(valor == true) {
                cola.add(m.getCell(fila, i));
            }
            fila++;
            valor=true;
        }

        //obtengo las demas columnas
        valor = true;
        vecinos = new ArrayList();
        aux = 0;
        while (aux<m.numColumnas()) {
            colb = new ArrayList();
            if (aux!=i) {
                fila = 0;
                while (fila<m.numFilas()) {
                    j = 0;
                    while (j<test.length && valor == true) {
                        if (fila == test[j]) valor=false;
                        j++;
                    }
                    if(valor == true) {
                        colb.add(m.getCell(fila, aux));
                    }
                    fila++;
                    valor=true;
                }
                //calculamos similaridad
                sim = aplicarMedida(cola, colb, med);

                //calculamos si entra en la k-vecindad
                registro raux = new registro(aux+1, sim);
                vecinos = comprobarVecinos(vecinos, raux, k);
            }
            else {
                sim = 0;
            }
        }
    }
}

```

```

        aux++;
    }

    //introducimos en la matriz de resultados
    aux2 = 0;
    while (aux2<k) {
        trainmat[i][aux2]=(registro)vecinos.get(aux2);
        aux2++;
    }
    i++;
    System.out.println(i);
}

return trainmat;
}

/** Metodo que aplica la medida de similaridad requerida a los dos dos
 * conjuntos dados */
public float aplicarMedida(ArrayList a, ArrayList b, int med) {
    double f = 0, f1 = 0, f2 = 0, aux;
    double ma = 0, mb = 0;
    float sim=0;
    int i = 0, cont = 0;

    if (a.size()!=b.size()) System.out.println("Error");
    else {

        switch (med) {
            case 1: //distancia euclidea
                while (i<a.size()) {
                    if ((Short)a.get(i)!=0 && (Short)b.get(i)!=0) {
                        aux = (Short)a.get(i) - (Short)b.get(i);
                        f = f + (Math.pow(aux, 2)/5);
                        cont++;
                    }
                    i++;
                }

                f = Math.sqrt(f) / Math.sqrt(cont);

                sim = 1 - (float)f;
                break;
            case 2: //coeficiente coseno
                while (i<a.size()) {
                    if ((Short)a.get(i)!=0 && (Short)b.get(i)!=0) {
                        aux = (Short)a.get(i) * (Short)b.get(i);
                        f = f + aux;
                    }
                    i++;
                }

                i=0;
                while (i<a.size()) {
                    if ((Short)a.get(i)!=0) {
                        aux = Math.pow((Short)a.get(i), 2);
                        f1 = f1 + aux;
                    }
                    i++;
                }

```



```

    }
    //System.out.println(f1);
    i=0;
    while (i<b.size()) {
        if ((Short)b.get(i)!=0) {
            aux = Math.pow((Short)b.get(i), 2);
            f2 = f2 + aux;
        }
        i++;
    }
    if (f1==0 || f2==0) f = 0;
    else f = f / Math.sqrt(f1 * f2);

    sim = (float)f;
    break;
case 3: //coeficiente Pearson
    while (i<a.size()) {
        if ((Short)a.get(i)!=0) {
            ma = ma + (Short)a.get(i);
            cont++;
        }
        i++;
    }
    ma = ma / cont;

    i = 0;
    cont = 0;
    while (i<b.size()) {
        if ((Short)b.get(i)!=0) {
            mb = mb + (Short)b.get(i);
            cont++;
        }
        i++;
    }
    mb = mb / cont;

    i=0;
    while (i<a.size()) {
        if ((Short)a.get(i)!=0 && (Short)b.get(i)!=0) {
            aux = ((Short)a.get(i) - ma) * ((Short)b.get(i) - mb);
            f = f + aux;
        }
        i++;
    }

    i=0;
    while (i<a.size()) {
        if ((Short)a.get(i)!=0) {
            aux = Math.pow(((Short)a.get(i) - ma), 2);
            f1 = f1 + aux;
        }
        i++;
    }
    f1 = Math.sqrt(f1);

    i=0;
    while (i<b.size()) {
        if ((Short)b.get(i)!=0) {

```

```

        aux = Math.pow(((Short)b.get(i) - mb), 2);
        f2 = f2 + aux;
    }
    i++;
}
f2 = Math.sqrt(f2);

if (f1==0 || f2==0) f = 0;
else f = f / (f1 * f2);
sim = (float)f;
break;
default: return 0;
}
}
return sim;
}

/** Metodo que devuelve la matriz de resultados del entrenamiento */
public registro[][] getKnn() {
    return arrknn;
}

/** Metodo que devuelve el conjunto de test */
public int[] getConjuntoTest() {
    return test;
}

/** Metodo que comprueba que el registro dado pertenece a la k-vecindad */
private ArrayList comprobarVecinos(ArrayList a, registro r, int k) {
    //si a no esta llena, insertamos al final y ordenamos
    if (a.size()<k) {
        a.add(r);
        burbuja(a);
    }
    //si a esta llena, comprobamos la similaridad y ordenamos
    else {
        registro tmp = (registro)a.get(k-1);
        if (r.leeValor()>=tmp.leeValor()) {
            a.set(k-1, r);
            burbuja(a);
        }
    }
    return a;
}

/* ordenamos mediante el metodo de la burbuja el array
* de los vecinos mas cercanos */
private void burbuja(ArrayList a) {
    int i, j;
    float aux1, aux2;
    registro r1, r2, tmp;

    for(j=a.size()-1; j>0; j--){
        for (i=0; i<j; i++) {
            r1 = (registro)a.get(i);
            r2 = (registro)a.get(i+1);
            if (r1.leeValor()<r2.leeValor()) {
                //intercambio

```

```
        tmp = (registro)r1;  
        a.set(i, r2);  
        a.set(j, tmp);  
    }  
}  
}
```

predict.java

Clase que realiza predicciones para los usuarios del conjunto test usando el algoritmo de predicción *item + adjustment*.

```
import java.sql.*;
import java.io.*;
import java.util.*;

public class Predict {
    static final String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
    static final String url = "jdbc:odbc:movieranks";
    // los vectores donde vamos a almacenar los resultados
    float[] mat1;
    float[] mat2;
    float[] mat3;
    float[] mat4;
    float[] mederr1, mederr2, mederr3, mederr4;
    long[] temp1;
    long[] temp2;
    long[] temp3;
    long[] temp4;
    float[] medtemp1, medtemp2, medtemp3, medtemp4;

    /** Creates a new instance of Predict */
    public Predict(registro[][] train, int[] test, Matrix m) {
        Connection conexion = null;
        Statement sentencia = null;
        ResultSet resultado;
        int i = 0;
        float mae, mae2, mae4, mae8;
        long tiempo;
        // inicializamos los vectores de resultados
        mat1 = new float[test.length];
        mat2 = new float[test.length];
        mat3 = new float[test.length];
        mat4 = new float[test.length];
        mederr1 = new float[1];
        mederr2 = new float[1];
        mederr3 = new float[1];
        mederr4 = new float[1];
        // inicializamos los vectores de tiempos
        temp1 = new long[test.length];
        temp2 = new long[test.length];
        temp3 = new long[test.length];
        temp4 = new long[test.length];
        medtemp1 = new float[1];
        medtemp2 = new float[1];
        medtemp3 = new float[1];
        medtemp4 = new float[1];

        //cargamos el controlador
        try {
            Class.forName(driver);
```

```

} catch (Exception ex) {
    System.err.println("No se pudo encontrar el driver");
    return;
}

//conectamos con la base de datos
try {
    conexion = DriverManager.getConnection(url);
    sentencia = conexion.createStatement();

    while (i<test.length) {
        //obtenemos el numero de puntuaciones de un usuario concreto
        int tmp = test[i]+1;

        resultado = sentencia.executeQuery("SELECT NUM_PUNTUACIONES "+
            "FROM USUARIOS WHERE ID_USER = "+tmp);
        while (resultado.next()) {
            int n_punt = resultado.getInt( "NUM_PUNTUACIONES" );

            System.out.println("Usuario: "+tmp);
            System.out.println("Numero de puntuaciones: "+n_punt);

            //aplicamos la tecnica de todos menos 1
            tiempo = 0;
            tiempo = System.currentTimeMillis();
            mae = todosMenos1(test[i], n_punt, train, m);
            tiempo = System.currentTimeMillis() - tiempo;
            mat1[i] = mae;
            temp1[i] = tiempo;

            //aplicamos la tecnica de datos 2
            tiempo = 0;
            tiempo = System.currentTimeMillis();
            mae2 = dadosN(2, test[i], n_punt, train, m);
            tiempo = System.currentTimeMillis() - tiempo;
            mat2[i] = mae2;
            temp2[i] = tiempo;

            //aplicamos la tecnica de datos 4
            tiempo = 0;
            tiempo = System.currentTimeMillis();
            mae4 = dadosN(4, test[i], n_punt, train, m);
            tiempo = System.currentTimeMillis() - tiempo;
            mat3[i] = mae4;
            temp3[i] = tiempo;

            //aplicamos la tecnica de datos 8
            tiempo = 0;
            tiempo = System.currentTimeMillis();
            mae8 = dadosN(8, test[i], n_punt, train, m);
            tiempo = System.currentTimeMillis() - tiempo;
            mat4[i] = mae8;
            temp4[i] = tiempo;
        }

        i++;
    }
}

```

```

//medias de los errores
float sum = 0, sum1 = 0, sum2 = 0, sum3 = 0;
int j = 0;
while (j < mat1.length) {
    sum = sum + mat1[j];
    sum1 = sum1 + mat2[j];
    sum2 = sum2 + mat3[j];
    sum3 = sum3 + mat4[j];
    j++;
}
sum = sum / mat1.length;
sum1 = sum1 / mat1.length;
sum2 = sum2 / mat1.length;
sum3 = sum3 / mat1.length;

mederr1[0] = sum;
mederr2[0] = sum1;
mederr3[0] = sum2;
mederr4[0] = sum3;

//medias de los tiempos
sum = 0; sum1 = 0; sum2 = 0; sum3 = 0;
j = 0;
while (j < temp1.length) {
    sum = sum + temp1[j];
    sum1 = sum1 + temp2[j];
    sum2 = sum2 + temp3[j];
    sum3 = sum3 + temp4[j];
    j++;
}
sum = sum / temp1.length;
sum1 = sum1 / temp1.length;
sum2 = sum2 / temp1.length;
sum3 = sum3 / temp1.length;

medtemp1[0] = sum;
medtemp2[0] = sum1;
medtemp3[0] = sum2;
medtemp4[0] = sum3;

} catch (SQLException e) {
    System.err.println("No se puede conectar a esta base de datos");
    System.err.println(e);
}

//cerramos la conexion y la sentencia
try {
    conexion.close();
    sentencia.close();
} catch (SQLException e) {
    System.err.println("Imposible de cerrar");
}
}

/** Metodo que aplica la tecnica de prediccion de todos menos 1
 * y devuelve el error MAE cometido
 */
public float todosMenos1(int user, int n_punt, registro[][] train, Matrix punt) {

```

```

Random r;
int item = 0, i = 0, j = 0;
int cont = 0, punt_user_item = 0;
float media_punt_item = 0, media_punt_user = 0;
float sim_total = 0, produc_total = 0;
registro tmp;
float sim = 0, pbi;
boolean encontrado, valido;
int[] matpunt = new int[n_punt];
float[] matpred = new float[n_punt];

//elegimos aleatoriamente la columna del item a predecir
valido = false;
r = new Random();
while (valido == false) {
    item = r.nextInt(punt.numColumnas());
    if (punt.getCell(user, item)!=0) {
        punt_user_item = punt.getCell(user, item);
        valido = true;
    }
}

//guardamos la puntuacion de este item dada por el usuario dado
matpunt[0]=punt_user_item;

//calculamos la media de las puntuaciones del user dado
i = 0;
while (i<punt.numColumnas()) {
    if (punt.getCell(user, i)!=0) {
        media_punt_user = media_punt_user + punt.getCell(user, i);
    }
    i++;
}
media_punt_user = media_punt_user / n_punt;

//calculamos la media de las puntuaciones de todos los usuarios con respecto
//al item a predecir
i = 0;
cont = 0;
while (i<punt.numFilas()) {
    if (punt.getCell(i, item)!=0) {
        media_punt_item = media_punt_item + punt.getCell(i, item);
        cont++;
    }
    i++;
}
media_punt_item = media_punt_item / cont;

//calculamos el dividendo y el divisor de la formula
//de la prediccion basada en item
i = 0;
cont = 0;
while (i<punt.numColumnas() && cont<n_punt-1) {
    //comprobamos que no incluimos en los calculos al item a predecir
    if (i!=item) {
        if (punt.getCell(user, i)!=0) {
            j = 0;
            encontrado = false;

```

```

        while (j<train[item].length && encontrado == false) {
            tmp = train[item][j];
            if (i+1 == tmp.leelIndex()) {
                sim = tmp.leeValor();
                encontrado = true;
            }
            j++;
        }
        if (encontrado == false) sim = 0;
        //divisor
        sim_total = sim_total + Math.abs(sim);
        //dividendo
        produc_total = produc_total +
            sim * (punt.getCell(user, i) - media_punt_user);
        cont++;
    }
    }
    i++;
}

//obtenemos la predicción basada en item
if (sim_total==0) pbi = media_punt_item;
else pbi = media_punt_item + (produc_total / sim_total);

matpred[0] = pbi;
float mae = calculaMAE(matpred, matpunt, 1);

return mae;
}

/** Metodo que aplica la tecnica de prediccion de datos n
 * y devuelve el error MAE cometido
 */
public float dadosN(int dados, int user, int n_punt, registro[][] train, Matrix punt) {
    Random r;
    int item = 0, i = 0, j = 0, aux, val;
    int cont = 0, cont2;
    float media_punt_item = 0, media_punt_user = 0;
    float sim_total = 0, produc_total = 0;
    registro tmp;
    float sim = 0, pbi;
    boolean encontrado;
    registro[] matdados = new registro[dados];
    int[] matpunt = new int[n_punt];
    float[] matpred = new float[n_punt];

    //elegimos aleatoriamente los items dados y los guardamos en matdados
    r = new Random();
    while (i<dados) {
        item = r.nextInt(punt.numColumnas());
        if (punt.getCell(user, item)!=0) {
            registro reg = new registro(item, punt.getCell(user, item));
            matdados[i] = reg;
            i++;
        }
    }
}

```



```

//realizamos la prediccion para los items no dados
aux = 0;
val = 0;
while (aux < punt.numColumnas()) {
    if (punt.getCell(user, aux) != 0) {
        i = 0;
        encontrado = true;
        while (i < matdados.length) {
            if (aux == matdados[i].leeIndex()) encontrado = false;
            i++;
        }

        if (encontrado == true) {
            //puntuacion del item por parte del user
            int punt_user_item = punt.getCell(user, aux);
            matpunt[val] = punt_user_item;

            //buscamos puntuaciones de item de cualquier usuario
            i = 0;
            cont = 0;
            while (i < punt.numFilas()) {
                if (punt.getCell(i, aux) != 0) {
                    media_punt_item = media_punt_item + punt.getCell(i, aux);
                    cont++;
                }
                i++;
            }

            //obtenemos la media de las puntuaciones del item
            media_punt_item = media_punt_item / cont;

            //buscamos puntuaciones de user sobre cualquier item
            i = 0;
            cont = 0;
            while (i < punt.numColumnas()) {
                if (punt.getCell(user, i) != 0) {
                    media_punt_user = media_punt_user + punt.getCell(user, i);
                    cont++;
                }
                i++;
            }

            //obtenemos la media de las puntuaciones de user
            media_punt_user = media_punt_user / cont;

            //calculamos el dividendo y el divisor de la formula
            //de la prediccion basada en item
            i = 0;
            sim_total = 0;
            produc_total = 0;
            while (i < matdados.length) {
                int aux2 = matdados[i].leeIndex();
                j = 0;
                encontrado = false;
                while (j < train[aux].length) {
                    tmp = train[aux][j];
                    if (aux2+1 == tmp.leeIndex()) {
                        sim = tmp.leeValor();
                    }
                }
            }
        }
    }
}

```

```

        encontrado = true;
    }
    j++;
}

if (encontrado == false) sim = 0;
//divisor
sim_total = sim_total + Math.abs(sim);
//dividendo
produc_total = produc_total +
    sim * (punt.getCell(user, aux2) - media_punt_user);

    i++;
}

if (sim_total==0) pbi = media_punt_item;
else pbi = media_punt_item + (produc_total / sim_total);
matpred[val] = pbi;
val++;
}

}

    aux++;
}

float mae = calculaMAE(matpred, matpunt, n_punt-dados);
return mae;
}

/** Metodo que calcula la medida de error MAE */
public float calculaMAE(float[] pbi, int[] p, int n) {
    float mae = 0;
    int i = 0;
    int cont = 0;

    while (i<pbi.length) {
        if(pbi[i]!=0) {
            mae = mae + Math.abs(p[i] - pbi[i]);
            cont++;
        }
        i++;
    }

    if(cont!=0) {
        mae = mae / cont;
    }

    return mae;
}

/** Metodo que devuelve el vector con los resultados de Todos Menos 1 */
public float[] devTM1() {
    return mederr1;
}

```

```
/** Metodo que devuelve el vector con los resultados de Datos 2 */
public float[] devD2() {
    return mederr2;
}

/** Metodo que devuelve el vector con los resultados de Datos 4 */
public float[] devD4() {
    return mederr3;
}

/** Metodo que devuelve el vector con los resultados de Datos 8 */
public float[] devD8() {
    return mederr4;
}

/** Metodo que devuelve el vector con los tiempos de Todos Menos 1 */
public float[] devT1() {
    return medtemp1;
}

/** Metodo que devuelve el vector con los tiempos de Datos 2 */
public float[] devT2() {
    return medtemp2;
}

/** Metodo que devuelve el vector con los tiempos de Datos 4 */
public float[] devT3() {
    return medtemp3;
}

/** Metodo que devuelve el vector con los tiempos de Datos 8 */
public float[] devT4() {
    return medtemp4;
}
}
```

predict2.java

Clase que realiza predicciones para los usuarios del conjunto test usando el algoritmo de predicción *weighted sum*.

```
import java.sql.*;
import java.io.*;
import java.util.*;

public class Predict2 {
    static final String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
    static final String url = "jdbc:odbc:movieranks";
    // los vectores donde vamos a almacenar los resultados
    float[] mat1;
    float[] mederr;
    // los vectores donde vamos a almacenar los tiempos de ejecucion
    long[] temp1;
    float[] medtemp;

    /** Creates a new instance of Predict2 */
    public Predict2(registro[][] train, int[] test, Matrix m) {
        Connection conexion = null;
        Statement sentencia = null;
        ResultSet resultado;
        int i = 0;
        float mae;
        long tiempo;
        // inicializamos los vectores de resultados
        mat1 = new float[test.length];
        mederr = new float[1];
        // inicializamos los vectores de tiempos
        temp1 = new long[test.length];
        medtemp = new float[1];

        //cargamos el controlador
        try {
            Class.forName(driver);
        } catch (Exception ex) {
            System.err.println("No se pudo encontrar el driver");
            return;
        }

        //conectamos con la base de datos
        try {
            conexion = DriverManager.getConnection(url);
            sentencia = conexion.createStatement();

            while (i<test.length) {
                //obtenemos el numero de puntuaciones de un usuario concreto
                int tmp = test[i]+1;

                resultado = sentencia.executeQuery("SELECT NUM_PUNTUACIONES "+
                    "FROM USUARIOS WHERE ID_USER = "+tmp);
                while (resultado.next()) {
```

```

        int n_punt = resultado.getInt( "NUM_PUNTUACIONES" );

        //aplicamos la tecnica de prediccion
        tiempo = 0;
        tiempo = System.currentTimeMillis();
        mae = tecPred(test[i], n_punt, train, m);
        tiempo = System.currentTimeMillis() - tiempo;
        mat1[i] = mae;
        temp1[i] = tiempo;

    }

    i++;
}

//calculamos la media de los errores cometidos
float sum = 0;
int j = 0;
while (j < mat1.length) {
    sum = sum + mat1[j];
    j++;
}

sum = sum / mat1.length;
mederr[0] = sum;

//calculamos la media de los tiempos empleados
sum = 0;
j = 0;
while (j < temp1.length) {
    sum = sum + temp1[j];
    j++;
}
sum = sum / temp1.length;

medtemp[0] = sum;

} catch (SQLException e) {
    System.err.println("No se puede conectar a esta base de datos");
    System.err.println(e);
}

//cerramos la conexion y la sentencia
try {
    conexion.close();
    sentencia.close();
} catch (SQLException e) {
    System.err.println("Imposible de cerrar");
}
}

/** Metodo que aplica la tecnica de prediccion
 * y devuelve el error MAE cometido
 */
public float tecPred(int user, int n_punt, registro[][] train, Matrix punt) {
    Random r;
    registro tmp;
    int i, item=0, punt_user_item=0, aux, p;

```

```

boolean valido;
float sim = 0, sim_total = 0, produc_total = 0, pbi;
int[] matpunt = new int[n_punt];
float[] matpred = new float[n_punt];

//elegimos aleatoriamente la columna del item a predecir
valido = false;
r = new Random();
while (valido == false) {
    item = r.nextInt(punt.numColumnas());
    if (punt.getCell(user, item)!=0) {
        punt_user_item = punt.getCell(user, item);
        valido = true;
    }
}

//guardamos la la puntuacion de este item dada por el usuario dado
matpunt[0]=punt_user_item;

//calculamos el dividendo y el divisor de la formula
//de la prediccion basada en item
i=0;
while (i<train[item].length) {
    tmp = train[item][i];
    sim = tmp.leeValor();
    aux = tmp.leeIndex();
    p = punt.getCell(user, aux-1);

    if (p != 0) {
        //dividendo
        produc_total = produc_total + (sim * p);
        //divisor
        sim_total = sim_total + Math.abs(sim);
    }

    i++;
}

//obtenemos la prediccion basada en item
if (sim_total==0) pbi = 0;
else pbi = produc_total / sim_total;

matpred[0] = pbi;
float mae = calculaMAE(matpred, matpunt, 1);

return mae;
}

/** Metodo que calcula la medida de error MAE */
public float calculaMAE(float[] pbi, int[] p, int n) {
    float mae = 0;
    int i = 0;
    int cont = 0;

    while (i<pbi.length) {
        if(pbi[i]!=0) {
            mae = mae + Math.abs(p[i] - pbi[i]);
            cont++;
        }
    }
}

```

```
    }  
    i++;  
  }  
  
  if(cont!=0) {  
    mae = mae / cont;  
  }  
  
  return mae;  
}  
  
/** Metodo que devuelve el valor medio de los errores cometidos*/  
public float[] devMedTemp() {  
  return medtemp;  
}  
  
/** Metodo que devuelve el valor medio de los tiempos empleados*/  
public float[] devMedErr() {  
  return mederr;  
}  
}
```


ANEXO IV.

CÓDIGO FUENTE DEL MÓDULO DE FORMATEO DE LA BASE DE DATOS.

Este anexo muestra el código fuente realizado para la conversión de los datos en texto plano de MovieLens en una base de datos relacional de MS Access preparada para su comunicación con el sistema de recomendación implementado.

MoviesRank.java

```
import java.sql.*;
import java.io.*;

public class MoviesRank {

    static final String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
    static final String url = "jdbc:odbc:movieranks";

    /** Creates a new instance of MoviesRank */
    public MoviesRank() {
        Connection conexion = null;
        Statement sentencia = null;
        ResultSet resultSet;
        String s, insert, aux;
        int i = 0, p = 0, a = 0, cont = 0, id = 0;

        //cargamos el controlador
        try {
            Class.forName(driver);
        } catch (Exception ex) {
            System.err.println("No se pudo encontrar el driver");
            return;
        }

        //conectamos con la base de datos
        try {
            conexion = DriverManager.getConnection(url);
            sentencia = conexion.createStatement();

            //borramos las tablas preexistentes
            try {
                //sentencia.execute( "DROP TABLE PUNTUACIONES ON DELETE CASCADE" );
                //sentencia.execute( "DROP TABLE USUARIOS" );
                //sentencia.execute( "DROP TABLE PELICULAS" );

            } catch (Exception e) {}

            //creamos las tablas
            sentencia.execute( "CREATE TABLE USUARIOS (" +
                " ID_USER INTEGER PRIMARY KEY, " +
                " EDAD INTEGER, " +
                " GENERO VARCHAR(1)," +
                " PROFESION VARCHAR(25)," +
                " COD_POSTAL VARCHAR(5)," +
                " NUM_PUNTUACIONES INTEGER" );

            sentencia.execute( "CREATE TABLE PELICULAS (" +
                " ID_MOVIE INTEGER PRIMARY KEY, " +
                " TITULO VARCHAR(81), " +
```

```

" FECHA DATE," +
" IMDB_URL VARCHAR(134)," +
" DESCONOCIDO BIT," +
" ACCION BIT," +
" AVENTURAS BIT," +
" ANIMACION BIT," +
" INFANTIL BIT," +
" COMEDIA BIT," +
" CRIMEN BIT," +
" DOCUMNETAL BIT," +
" DRAMA BIT," +
" FANTASIA BIT," +
" NEGRO BIT," +
" TERROR BIT," +
" MUSICAL BIT," +
" MISTERIO BIT," +
" ROMANTICO BIT," +
" CIENCIA_FICCION BIT," +
" THRILLER BIT," +
" GUERRA BIT," +
" WESTERN BIT)" );

sentencia.execute( "CREATE TABLE PUNTUACIONES (" +
" ID_USER INTEGER, " +
" ID_MOVIE INTEGER, " +
" RATING BYTE NOT NULL," +
" PRIMARY KEY (ID_USER, ID_MOVIE)," +
" FOREIGN KEY (ID_USER) REFERENCES USUARIOS," +
" FOREIGN KEY (ID_MOVIE) REFERENCES PELICULAS)" );

//sacamos la informacion de los usuarios
try {
    BufferedReader us = new BufferedReader(new FileReader("u.user"));
    try {
        while ((s = us.readLine()) != null) {
            insert = nuevoUsuario(s);
            sentencia.executeUpdate(insert);
        }
    } catch (IOException ioe) {}
} catch (FileNotFoundException fnfe) {
    System.out.println("Archivo no encontrado");
}

//sacamos la informacion de las peliculas
try {
    BufferedReader pel = new BufferedReader(new FileReader("u.item"));
    try {
        while ((s = pel.readLine()) != null) {
            insert = nuevotem(s);
            sentencia.executeUpdate(insert);
        }
    } catch (IOException ioe) {}
} catch (FileNotFoundException fnfe) {
    System.out.println("Archivo no encontrado");
}

```

```

//sacamos las puntuaciones de los usuarios sobre las películas
try {
    BufferedReader dat = new BufferedReader(new FileReader("u.data"));
    try {
        while ((s = dat.readLine()) != null) {

            insert = nuevaPuntuacion(s);
            sentencia.executeUpdate(insert);

            //calculamos el campo NUM_PUNTUACIONES de la tabla USUARIOS
            i=0;
            while (i<insert.length() && (insert.substring(i, i+1).compareTo("") != 0)) i++;
            i++;
            int ini = i;
            while (i<insert.length() && (insert.substring(i, i+1).compareTo(",") != 0)) i++;
            aux = insert.substring(ini, i);
            id = Integer.parseInt(aux);

            sentencia.executeUpdate("UPDATE USUARIOS SET " +
                "NUM_PUNTUACIONES=NUM_PUNTUACIONES+1 " +
                "WHERE ID_USER="+id+";");
        }
    } catch (IOException ioe) {}

    } catch (FileNotFoundException fnfe) {
        System.out.println("Archivo no encontrado");
    }

    } catch (SQLException e) {
        System.err.println("No se puede conectar a esta base de datos");
        System.err.println(e);
    }
}

//cerramos la conexión y la sentencia
try {
    conexion.close();
    sentencia.close();
} catch (SQLException e) {
    System.err.println("Imposible de cerrar");
}
}

/*creamos la sentencia SQL para insertar un nuevo Usuario */
public String nuevoUsuario(String s) {
    String sid, edad, sgen, sprof, scod, saux;
    int iid, iedad, icod, ipun;
    int i = 0, p = 0;

    while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
        i++;
    sid = s.substring(p, i);

    i++;
    p = i;
    while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
        i++;
    edad = s.substring(p, i);
}

```

```

    i++;
    p = i;
    while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
        i++;
    sgen = s.substring(p, i);
    sgen = ""+sgen+"";

    i++;
    p = i;
    while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
        i++;
    sprof = s.substring(p, i);
    sprof = ""+sprof+"";

    i++;
    p = i;
    while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
        i++;
    scod = s.substring(p, i);
    scod = ""+scod+"";

    iid = Integer.parseInt(sid);
    iedad = Integer.parseInt(sedad);
    ipun = 0;

    String insert = "INSERT INTO USUARIOS VALUES("
        +iid+", "+iedad+", "+sgen+", "+sprof+", "+scod+", "+ipun+");"

    return (insert);
}

/*creamos la sentencia SQL para insertar un nuevo Item */
public String nuevotem(String s) {
    String sid, stit, sfec, surl, sdes, sac;
    String sav, sani, sinf, scom, scri, sdoc;
    String sdra, sfan, sne, ste, smu, smis;
    String sro, scfi, sthri, sgu, swes, sfec2, saux;
    boolean bdes, bac, bav, bani, binf, bcom;
    boolean bcri, bdoc, bdra, bfan, bne, bte;
    boolean bmu, bmis, bro, bcfi, bthri, bgu, bwes;
    int iid;
    int i = 0, p = 0;

    while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
        i++;
    sid = s.substring(p, i);

    i++;
    p = i;
    while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
        i++;
    stit = s.substring(p, i);

    stit = quitaComillas(stit);
    stit = ""+stit+"";

    i++;
    p = i;

```

```
while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
    i++;
sfec = s.substring(p, i);
if (sfec.compareTo("")==0) sfec2 = "01/01/1801";
else sfec2 = formateoFecha(sfec);
sfec2 = ""+sfec2+"";

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
    i++;
saux = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
    i++;
surl = s.substring(p, i);
surl = quitaComillas(surl);
surl = ""+surl+"";

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
    i++;
sdes = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
    i++;
sac = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
    i++;
sav = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
    i++;
sani = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
    i++;
sinf = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
    i++;
scom = s.substring(p, i);

i++;
```

```

p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("") != 0))
    i++;
scri = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("") != 0))
    i++;
sdoc = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("") != 0))
    i++;
sdra = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("") != 0))
    i++;
sfan = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("") != 0))
    i++;
sne = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("") != 0))
    i++;
ste = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("") != 0))
    i++;
smu = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("") != 0))
    i++;
smis = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("") != 0))
    i++;
sro = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("") != 0))
    i++;
scfi = s.substring(p, i);

```

```
i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
    i++;
sthri = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
    i++;
sgu = s.substring(p, i);

i++;
p = i;
while (i<s.length() && (s.substring(i, i+1).compareTo("|") != 0))
    i++;
swes = s.substring(p, i);

iid = Integer.parseInt(sid);
if (sdes.compareTo("1")==0) bdes=true;
else bdes=false;
if (sac.compareTo("1")==0) bac=true;
else bac=false;
if (sav.compareTo("1")==0) bav=true;
else bav=false;
if (sani.compareTo("1")==0) bani=true;
else bani=false;
if (sinf.compareTo("1")==0) binf=true;
else binf=false;
if (scom.compareTo("1")==0) bcom=true;
else bcom=false;
if (scri.compareTo("1")==0) bcri=true;
else bcri=false;
if (sdoc.compareTo("1")==0) bdoc=true;
else bdoc=false;
if (sdra.compareTo("1")==0) bdra=true;
else bdra=false;
if (sfan.compareTo("1")==0) bfan=true;
else bfan=false;
if (sne.compareTo("1")==0) bne=true;
else bne=false;
if (ste.compareTo("1")==0) bte=true;
else bte=false;
if (smu.compareTo("1")==0) bmu=true;
else bmu=false;
if (smis.compareTo("1")==0) bmis=true;
else bmis=false;
if (sro.compareTo("1")==0) bro=true;
else bro=false;
if (scfi.compareTo("1")==0) bcfi=true;
else bcfi=false;
if (sthri.compareTo("1")==0) bthri=true;
else bthri=false;
if (sgu.compareTo("1")==0) bgu=true;
else bgu=false;
if (swes.compareTo("1")==0) bwes=true;
else bwes=false;
```



```

String res = "INSERT INTO PELICULAS VALUES("+
    iid+","+stit+","+sfec2+","+surl+","+
    bdes+","+bac+","+bav+","+bani+","+
    binf+","+bcom+","+bcri+","+bdoc+","+
    bdra+","+bfan+","+bne+","+bte+","+bmu+","+
    bmis+","+bro+","+bcfi+","+bthri+","+
    bgu+","+bwes+)";

return (res);
}

/* metodo que devuelve la sentencia SQL para insertar
 * una nueva puntuacion*/
public String nuevaPuntuacion(String s){
    String sidu, sidm, srat;
    int iidu, iidm;
    short irat;
    int i = 0, p = 0;

    while (i<s.length() && (s.substring(i, i+1).compareTo("\t") != 0))
        i++;
    sidu = s.substring(p, i);

    i++;
    p = i;
    while (i<s.length() && (s.substring(i, i+1).compareTo("\t") != 0))
        i++;
    sidm = s.substring(p, i);

    i++;
    p = i;
    while (i<s.length() && (s.substring(i, i+1).compareTo("\t") != 0))
        i++;
    srat = s.substring(p, i);

    iidu = Integer.parseInt(sidu);
    iidm = Integer.parseInt(sidm);
    irat = Byte.parseByte(srat);

    //comprobamos que la puntuacion vaya entre 1 y 5
    if (irat>5) irat = 0;

    String insert = "INSERT INTO PUNTUACIONES VALUES("
        +iidu+","+iidm+","+irat+)";
    return (insert);
}

/* metodo que da el formato correcto a las fechas */
private String formateoFecha(String s) {
    String smes="00", res;

    String sdia = s.substring(0, 2);
    if ((s.substring(3, 6).compareTo("Jan"))==0) smes="01";
    if ((s.substring(3, 6).compareTo("Feb"))==0) smes="02";
    if ((s.substring(3, 6).compareTo("Mar"))==0) smes="03";
    if ((s.substring(3, 6).compareTo("Apr"))==0) smes="04";
    if ((s.substring(3, 6).compareTo("May"))==0) smes="05";

```

```
        if ((s.substring(3, 6).compareTo("Jun"))==0) smes="06";
        if ((s.substring(3, 6).compareTo("Jul"))==0) smes="07";
        if ((s.substring(3, 6).compareTo("Aug"))==0) smes="08";
        if ((s.substring(3, 6).compareTo("Sep"))==0) smes="09";
        if ((s.substring(3, 6).compareTo("Oct"))==0) smes="10";
        if ((s.substring(3, 6).compareTo("Nov"))==0) smes="11";
        if ((s.substring(3, 6).compareTo("Dec"))==0) smes="12";

        String sanio = s.substring(7);

        res = smes+"/"+s.dia+"/"+sanio;
        return res;
    }

    /* metodo que cambia las comillas simples por acentos en
    * los titulos de las peliculas*/
    private String quitaComillas(String s) {
        int j = 0;
        boolean encontrado = false;

        //pasada para eliminar la primera comilla
        while (j<s.length() && !encontrado){
            if (s.substring(j, j+1).compareTo("'")==0){
                encontrado=true;
            }
            j++;
        }
        if (encontrado==true) {
            s=s.substring(0, j-1)+" "+s.substring(j);
        }

        //pasada para eliminar la segunda comilla
        j=0;
        encontrado = false;
        while (j<s.length() && !encontrado){
            if (s.substring(j, j+1).compareTo("'")==0){
                encontrado=true;
            }
            j++;
        }
        if (encontrado==true) {
            s=s.substring(0, j-1)+" "+s.substring(j);
        }

        return s;
    }

    //fin
}
```

BIBLIOGRAFÍA

1) Bibliografía específica sobre Sistemas de Recomendación

- [1] Balabanovic, M. y Shoham, Y. (1997), "Content-based, collaborative recommendation" en *Communications of ACM* 40.
- [2] Breese, J. S., Heckerman, D. y Kadie, C. (1998), "Empirical analysis of predictive algorithms for collaborative filtering" en *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Madison, Wisconsin, USA.
- [3] Cho, Y. H., Kim, J. K. Y Kim, S. H. (2002), "A personalized recommender system based on web usage mining and decision tree induction" en *Expert Systems with Applications*, vol. 23, pp. 233-342.
- [4] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. y Sartin, M. (1999), "Combining content-based and collaborative filters in an online newspaper" en *Proceedings of ACM SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley, CA.
- [5] Dai, H. y Mobasher, B., "Integrated semantic knowledge with web usage mining and personalization" en *Web Mining: Applications and Techniques*, Ed. IRM Press
- [6] Guo, X. (2006), *Personalized Government Online Services with Recommendation Techniques*, tesis Phd, University Graduate School, University of Technology Sydney.
- [7] Herlocker, J., Konstan, J., Terveen, L. y Riedl, J. (2004), "Evaluating Collaborative Filtering Recommender Systems" en *ACM Vol. Transactions on Information Systems*, vol. 22, pp. 5-53.
- [8] Karypis, G. (2001), "Evaluation of item-based top-N recommendation algorithms" en *Proceedings of ACM 10th International Conference on Information and Knowledge Management*, Atlanta, Georgia.
- [9] Papagelis, M., Plexousakis, D., Rousadis, I. y Theorapoulos, E. (2004), "Qualitative Analysis of User-based and Item-based Prediction Algorithms for Recommendation Systems", *Proceedings of the 3rd Hellenic Data Management Symposium*.

- [10] Sarwar, B., Konstan, J., Riedl, J., Borchers, A., Herlocker, J. y Miller, B. (1998), "Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system" en *Proceedings of the 1998 ACM Conference on Computer Support Cooperative Work*, Seattle, Washington, USA.
- [11] Sarwar, B., Konstan, J., Terveen, L. y Riedl, J. (2000), "Analysis of recommendation algorithms for e-commerce" en *Proceedings of the 2nd ACM Conference on Electronic Commerce*, Minneapolis, Minnesota, USA.
- [12] Sarwar, B., Konstan, J., Terveen, L. y Riedl, J. (2001), "Item-Based Collaborative Filtering Recommendation" en *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, China.
- [13] Shardanan, U. y Maes, P. (1995), "Social information filtering: algorithms for automating 'word of mouth'", en *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*.

2) Bibliografía general para la realización del proyecto

[14] Dawson, C. W. y Martin, G. (2002), *El Proyecto Fin de Carrera en Ingeniería Informática: Una guía para el Estudiante*, Prentice Hall, Madrid.

[15] Gutiérrez, A. y Bravo, G. (2005), *PHP5 a través de ejemplos*, Editorial Ra-Ma

[16] Krug, S. (2000), *No me hagas pensar. Una aproximación a la usabilidad en la Web*, Prentice Hall, Madrid.

[17] Manchón, E. (2002), *¿Qué es usabilidad?*, disponible en Internet (http://www.ainda.info/que_es_usabilidad.htm).

[18] García Gómez, J. C. y Saorín Pérez, T. (2006), *Usabilidad para principiantes*, disponible en Internet (<http://usalo.es/?p=117>).

[19] *scriptaculous wiki*, disponible en Internet (<http://wiki.script.aculo.us/scriptaculous>).