Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación

PRÁCTICAS DE APLICACIONES EN INTERNET

Propuesta del Trabajo de Prácticas 2020/2021

Aplicación web de puntuación y recomendación de vídeos

Revisión 1.3

Profesores:
Esteban Egea López
Juan José Alcaraz Espín

Índice.

Índice. 2
1 Consideraciones generales. 3
1.1 Objetivos 3
1.2 Introducción 3
1.2.1 Qué deben hacer los alumnos 3
1.2.2 Cómo está organizada esta propuesta 3
2 Desarrollo de la aplicación 3
2.1 Lógica de la aplicación 4
2.2 Diseño de la base de datos 5
2.3 Interfaz con los algoritmos de análisis de la aplicación 6
2.4 Algoritmo de filtrado colaborativo 7
2.5 Desarrollo libre y extensión 7
3 Material a entregar y criterios de evaluación 7
3.1 Memoria y Código 8
3.2 Criterios de evaluación 8
3.3 Grupos de una persona 8
4 Notas adicionales para la implementación 9
4.1 Implementación de la lógica en PHP 9
4.2 Diseño alternativo de la base de datos 9
4.3 Uso de frameworks y librerías adicionales 9
5 Bibliografía 9
6 Anexo 1. Estructura de la base de datos 9
6.1 Tabla genre 9
6.2 Tabla movie 10
6.3 Tabla moviecomments 10
6.4 Tabla moviegenre 10
6.5 Tabla recs 10
6.6 Tabla users 10
6.7 Tabla user_score 11
7 Anexo 2. Uso de Java en Matlab 11

1 Consideraciones generales.

1.1 Objetivos

En este trabajo de prácticas los alumnos realizarán por parejas una aplicación de puntuación y recomendación de vídeos (películas). El objetivo es que el alumno se familiarice con las tecnologías web más empleadas en la actualidad (HTML, CSS, PHP, javascript) así como con el tratamiento de los datos y los algoritmos de análisis de información más relevantes.

1.2 Introducción

La aplicación a desarrollar imitará de manera simplificada el funcionamiento de aplicaciones como http://www.filmaffinity.com, por ejemplo. Se trata de una aplicación que dispone de un catálogo de películas, con información variada sobre la misma, como el póster, fecha de estreno y género, que cualquier usuario puede consultar. La aplicación permite a los usuarios registrarse. Los usuarios registrados pueden puntuar las películas, añadir comentarios y recibir una recomendación del sistema.
Para poder realizar la aplicación se ha recopilado un catálogo de películas y de usuarios y puntuaciones de distintas fuentes. Se ha utilizado el *dataset* de Movielens 10k (http://grouplens.org/datasets/movielens/) que se ha completado con información extraída de IMDB (http://www.imdb.com/) y con una lista de nombres ficticia generada automáticamente. Esta información base se pondrá a disposición de los alumnos para que sobre ella implementen su aplicación, tal y como se describe en los apartados posteriores.

1.2.1 Qué deben hacer los alumnos

Los alumnos deben implementar por parejas una aplicación web con la funcionalidad que se especifica en este documento. Para ello deberán:

1. Programar los scripts en PHP con la lógica de la aplicación.
2. Programar cualquier fichero adicional con código HTML, CSS o javascript necesario para la aplicación.
3. Programar los scripts de Matlab con el algoritmo de análisis de datos.

La aplicación debe ser completamente funcional y debe poder probarse. Para ello, se alojará la aplicación en la cuenta del servidor del laboratorio (labit601.upct.es) y se accederá a ella a través del servidor HTTP del laboratorio.

Se deberá entregar una memoria del trabajo realizado. El contenido de la memoria y los criterios de evaluación también están descritos en esta propuesta.

1.2.2 Cómo está organizada esta propuesta.

En el apartado 2 se describe la funcionalidad mínima a implementar y las partes en las que se ha dividido el desarrollo, especificando cuáles serán opcionales.
En el apartado 3 se especifican los criterios de evaluación y la memoria y el material que se debe entregar.
Finalmente, en el apartado 4 se dan algunas sugerencias sobre la implementación y el uso de librerías adicionales.

2 Desarrollo de la aplicación

La **funcionalidad mínima** que debe proporcionar la aplicación es la siguiente:

1. **Catálogo**. Mostrar y permitir la navegación por todo el catálogo de películas. Para ello debe mostrar:
    a. Imagen con el póster de la película.
    b. Título de la película hiperenlazado a una nueva página que mostrará el contenido descrito en 4 (ver abajo).
    c. Descripción de la película.
    d. Fecha de estreno.
    e. Puntuación media de la película, número de puntuaciones recibidas y **puntuación ponderada de la película.**
    f. Se debe permitir la **reordenación** de los elementos en base, al menos, a los campos nombre y puntuación media.
2. **Login**. Mostrar un enlace o un formulario en la página de inicio que permita iniciar una sesión a los usuarios registrados o registrar un nuevo usuario.
3. **Registro**. Permite añadir un nuevo usuario. Los datos que puede aportar el usuario son:

   a. Nombre.
   b. Edad.
   c. Sexo.
   d. Ocupación.
   e. Clave de acceso.
   f. Foto de perfil.
4. **Película**. Mostrar los campos asociados a la película (ver 1) que considere y:
   a. Los géneros a los que pertenece.
   b. Los comentarios recibidos con el nombre del usuario que lo realizó.
   c. Puntuación recibida por el usuario (si la ha puntuado ya) si el usuario se ha identificado y ha iniciado una sesión.
   d. Posibilidad de puntuar o cambiar la puntación si el usuario se ha identificado y ha iniciado una sesión.
   e. Posibilidad de añadir un comentario si el usuario se ha identificado y ha iniciado una sesión.
5. **Usuario**. Mostrar los datos personales del usuario y permitir
   a. **Generar recomendaciones personalizadas**. Ejecutará el algoritmo de recomendación y generará una puntuación recomendada para cada una de las películas del catálogo.
   b. **Mostrar las recomendaciones generadas**. Se mostrarán o bien todas o bien un subconjunto, ordenadas por mayor puntuación.
   c. Modificar los datos personales.
6. **Formato**. Debe proporcionar al menos una hoja de estilo que dé formato a su aplicación. El formato es libre pero se valorará un uso adecuado.
7. **Contenido dinámico**. Debe utilizar javascript para generar el contenido, formato o responder a acciones de usuario.

Para implementar esta funcionalidad se le proporcionará una base de datos ya rellena con los datos que necesita. La base de datos se describe en 2.2 y en el Anexo 1. Para generar las recomendaciones deberá utilizar el algoritmo de filtrado colaborativo que realizó en prácticas y proporcionar una interfaz con el servidor Matlab que lo ejecutará, como se describe en 2.3. La parte fundamental de la aplicación es la lógica de aplicación en PHP, que se describe en 2.1.
La **calificación** del trabajo se reparte entre una **parte obligatoria con la funcionalidad mínima (75%), y una de temática libre y extensión (25%)**. La parte de temática libre y extensión se asignará a funcionalidad extra no solicitada en la especificación.

Se le proporciona una implementación de ejemplo de esta aplicación con la funcionalidad requerida en http://labit601.upct.es/video/ para guiarle en su trabajo. Puede iniciar sesión con cualquier usuario del 1 al 900 y clave ai04.
ATENCIÓN:

- **No es necesario que copie exactamente la estructura de la aplicación de ejemplo.**
- **No es necesario que imite el formato de la aplicación de ejemplo.**
- **No es necesario que use las mismas técnicas que en la aplicación de ejemplo.**

2.1 Lógica de la aplicación

La lógica de la aplicación.
Se implementará mediante **scripts PHP** que proporcionarán *al menos* la **funcionalidad mínima** requerida descrita en el apartado anterior. Organice la funcionalidad en scripts separados y utilice funciones para reutilizar partes de código que se repitan. Puede utilizar *require* y *require_once* para cargar librerías de funciones u otros scripts PHP. En particular, considere la posibilidad de agrupar el HTML común a todas las páginas en funciones/scripts de cabecera o pie.

Imágenes
Las imágenes utilizadas se han comprimido en un fichero disponible en http://labit601.upct.es/~eegea/aplicacion/.

Interfaz de consultas SQL.
La lógica de la aplicación realiza consultas a la base de datos mediante SQL, insertando las variables PHP necesarias. Para realizar la funcionalidad mínima, debe desarrollar un conjunto de consultas SQL como mínimo para:

- Seleccionar la información de un usuario dado y comprobar su clave de acceso.
- Insertar o modificar la información de un usuario.
- Seleccionar la información de una película dada.
- Seleccionar, insertar o modificar la puntuación de una película para un usuario dado.
- Seleccionar los géneros a los que pertenece una película.

- Seleccionar los comentarios asociados a una película.
- Añadir un comentario a una película por un usuario.
- Seleccionar, insertar o actualizar las recomendaciones para un usuario dado.
- Seleccionar las puntuaciones de todos los usuarios y todas películas.
- Seleccionar la información de todas las películas junto con la puntuación media de cada película y el número de puntuaciones de cada película.

Estilo y javascript.
Desarrolle una hoja de estilo y, si lo cree conveniente, código javascript para implementar la funcionalidad o el estilo.

Puntuación ponderada: Bayesian Ranking
Puede comprobar que este tipo de aplicaciones presentan el problema de cómo agregar puntuaciones para poder comparar una película con otra. Por ejemplo, ¿está mejor valorada una película con 356 votos y media 3.8 que una con 10 votos y media 4.8?
La respuesta a esta pregunta es relativamente compleja y puede encontrar una discusión muy interesante al respecto en [1, capítulo 5]. Si la lee recibirá una explicación sobre el método de ponderación que va a utilizar en este trabajo y que se describe a continuación. Para obtener una **puntuación ponderada**, se aplicará la siguiente fórmula a cada película i:

$$p_i = \frac{NR + n_i r_i}{N + n_i}$$

donde N es el número total de películas, R es la puntuación media de todas las películas, n_i es el número de puntuaciones de la película i y r_i es la puntuación media de la película i.

MATERIAL A ENTREGAR:

- **Scripts PHP, hojas de estilo, ficheros con javascript y HTML** Estarán localizados en su directorio *public_html* bajo el directorio *video*, de manera que la aplicación pueda ser probada.
- **Memoria** descriptiva de la lógica de la aplicación con una justificación de las decisiones de diseño. **No incluya el código completo**. Limítese a describir cómo se ha organizado la lógica, dónde se ha implementado la funcionalidad y comente las decisiones de diseño que crea más relevantes.

2.2 Diseño de la base de datos

Dispone de una base de datos con los datos necesarios para la aplicación. Para trabajar con ella, cada pareja cuenta con un usuario de la base de datos y una base de datos para desarrollo. Se le asignará un nombre de usuario y clave para trabajar con ellos. Puede utilizar phpmyadmin (http://labit601.upct.es/phpmyadmin/) o directamente el comando *mysql* para trabajar con su base de datos. Si realiza el trabajo localmente, deberá exportar la BBDD e importarla en su equipo.

La estructura de las tablas de la base de datos se detalla en el Anexo 1. No se ha establecido ninguna relación entre tablas, lo que quiere decir que aunque hay campos que obviamente hacen referencia a campos de otras tablas, la base de datos **no comprobará la integridad referencial automáticamente**.

La base de datos de la que parte se ha diseñado a partir de un conjunto de ficheros con datos, disponibles en el Aula Virtual que se proporciona por si decide modificar la base de datos proporcionada o diseñar la suya propia. Cada línea del fichero contiene el registro de una película y cada campo está separado por un tabulador ("\t"). Examínelos, antes de realizar el diseño. Los campos de cada fichero son:

- u.movie5: identificador de película, título, fecha de estreno, URL en IMDB, nombre de la imagen asociada y descripción.
- u.user2: identificador de usuario, nombre, edad, ocupación, clave de acceso.
- u.genre: nombre de un género de la película e identificador asociado del género.
- u.moviegenre: identificador de la película y 19 campos que valen 1 si la película está etiquetada dentro del género (ver u.genre) y 0 en caso contrario.
- u.data: identificador de usuario, identificador de película, puntuación (del 1 al 5) y *timestamp* en segundos desde 1/1/1970 UTC.
- imagenes.rar: pósters disponibles para las películas (no todas). El nombre de la imagen corresponde con el nombre de la imagen en u.movie5 para cada película.

2.3 Interfaz con los algoritmos de análisis de la aplicación

Los algoritmos de análisis (recomendación, etc.) se implementarán mediante Matlab, por lo que es necesaria una interfaz entre PHP y Matlab. En una aplicación real de este estilo es muy poco probable que se hiciera de esta forma, sino que se reimplementarían los algoritmos en un lenguaje más apropiado para su uso en un entorno de producción. En nuestro caso, para simplificar el desarrollo y evitar reimplementarlos, utilizará directamente Matlab.

Para ello haremos lo siguiente:

1. Matlab se invocará mediante un servidor que se ejecuta en el laboratorio. El servidor recibirá peticiones TCP, al puerto 1111, de un script PHP que le enviará el **path absoluto en el que se encuentran los scripts de Matlab necesarios** así como **la función a ejecutar** incluyendo los parámetros necesarios. **No tiene que implementar este servidor TCP, ya se encuentra funcionando en el laboratorio.** El servidor invocará el script de Matlab correspondiente al algoritmo.
2. El algoritmo obtendrá los datos necesarios directamente de la BBDD mediante consultas SQL y actualizarán la BBDD con los datos generados.

Siendo más precisos, tiene que realizar lo siguiente:

- **Un script PHP** que se conecte mediante un socket TCP al servidor de Matlab y le pase:
  - la **ruta absoluta de los scripts de Matlab** que tiene que ejecutar
  - y el **nombre de la función que tiene que ejecutar**. Esa función es precisamente la que implementa el algoritmo de filtrado colaborativo.
- La función que implementa el algoritmo contendrá el **código modificado** de la práctica de Sistemas de Recomendación. En esa práctica, las matrices R e Y y movieList se leen de ficheros locales. Para la aplicación, esas matrices tienen que ser construidas a partir de los datos de la BBDD. Por tanto, tiene que **implementar una función *getData()*** que genera las matrices R, Y y movieList a partir de las tablas de la base de datos. Además, en la práctica los resultados simplemente se muestran por pantalla. En la aplicación, los datos se tienen que guardar en la BBDD, mediante la **función *updateRecommendation()* que tiene que implementar**. Estas funciones, por tanto, se ejecutan como parte del código de la función que implementa el algoritmo de filtrado colaborativo.

Implementación del servidor de recomendación con Java y Matlab

Una forma inmediata de implementar esta funcionalidad es utilizar código Java directamente sobre Matlab. Matlab ejecuta una máquina virtual de Java y permite ejecutar código Java. Es decir, se pueden utilizar clases de Java dentro de los scripts de Matlab. Simplemente hay que tener cuidado y utilizar la sintaxis apropiada, que es una mezcla de Java y Matlab (ver http://es.mathworks.com/help/matlab/matlab_external/bringing-java-classes-and-methods-into-matlab-workspace.html). En el Anexo 2 se le proporciona un ejemplo de implementación de un servidor TCP con Matlab, que utiliza las clases de java.net. Puede comprobar como la sintaxis es una mezcla de Java y Matlab.

El servidor espera recibir:

1. Una cadena de caracteres con la ruta absoluta al directorio en el que se encuentran sus scripts de Matlab, terminada con retorno de carro. Por ejemplo, en PHP tendría que establecer un socket con el servidor y enviar una variable así *$ruta="/home/alumnos/ai/matlab\r\n";*
2. Una cadena de caracteres con la invocación de la función que ejecutará su algoritmo. Por ejemplo, si su función se ha declarado como *filtrado(id*), es decir, se le pasa como parámetro un identificador de usuario, deberá usar una variable en PHP *$fun="filtrado(".$userid.")\r\n"*.

Observe como se introduce el "\r\n" al final de las cadenas para que el servidor separe los dos campos, puesto que se usa la función *readLine()* de Java para recibir los datos en el servidor. Finalmente, antes de escribir dichas cadenas sobre su socket, concatene con *chr(0)* para asegurar que se fuerza el envío y se recibe correctamente. Con el ejemplo anterior, tendríamos que hacer

```
$info = $ruta.$fun.$chr(0);
$sent=socket_write($socket, $info, strlen($info));
```

El servidor TCP de Matlab ejecutándose escucha el **puerto 1111**. El servidor guardará en la ruta que se le proporciona para los scripts de Matlab (ver 1 arriba), un fichero **matlab.log** con un resumen de la ejecución. **Atención: para que el servidor pueda generar este fichero de log, el directorio de los scripts de Matlab debe tener permisos de escritura para otros**. Puede (y debe) consultar ese fichero para depurar errores.

RESUMEN. TIENE QUE IMPLEMENTAR:

- Scripts de Matlab con las funciones:
  - *getData*. Función que genera las matrices R, Y y movieList a partir de las tablas de la base de datos.

- *updateRecommendation*. Función que recibe una matriz de recomendaciones para un usuario y su identificador de usuario y actualiza la tabla de recomendaciones de la base de datos (*recs*).
  MUY IMPORTANTE: la distribución de Matlab que usamos en la Universidad, no tiene licenciado el toolbox Database de Matlab. **Por tanto, NO PUEDE UTILIZAR EL TOOLBOX DATABASE DE MATLAB para implementar las funciones anteriores.** Debe utilizar **Java JDBC** sobre Matlab para obtener y actualizar los datos de la BBDD en sus scripts de Matlab. **SI UTILIZA EL TOOLBOX DATABASE NO FUNCIONARÁ SU APLICACIÓN EN EL SERVIDOR DEL LABORATORIO Y NO SE LE PUNTUARÁ ESTA FUNCIONALIDAD.**

- Una función PHP que se conecta al servidor Matlab del laboratorio pasándole los datos para que ejecute el algoritmo de filtrado colaborativo como se describe en este apartado.

2.4 Algoritmo de filtrado colaborativo

Debe incorporar en la aplicación el código realizado en la práctica de Sistemas de Recomendación basada en filtrado colaborativo. Para obtener las matrices R e Y puede **emplear la función *getData()* del apartado anterior**. Después de entrenar el algoritmo y generar las puntuaciones (mediante el código realizado en la práctica), debe actualizarlas en la base de datos mediante la **función *updateRecommendation* del apartado anterior**.

2.5 Desarrollo libre y extensión

Un porcentaje de la nota de este trabajo (**25%**) se asignará a la innovación o extensión de la aplicación aquí descrita. Por tanto, se anima a los alumnos a incorporar la funcionalidad adicional que consideren necesaria o atractiva para la aplicación.

En la calificación se tendrán en cuenta tanto:

- las extensiones de la **funcionalidad**,
- como variaciones del **algoritmo de recomendación** o experimentos comparativos.

Algunas ideas para extender la aplicación son las siguientes:

- Por ejemplo, observe que los usuarios en la BBDD se identifican mediante ID, no nombre o correo electrónico. Una extensión evidente es permitir la identificación mediante otra de esas opciones.
- Puede permitir que los usuarios asignen *tags* o etiquetas a las películas (tanto de una lista predefinida, como sus propios tags). Esos tags se pueden utilizar para realizar búsquedas o para establecer similaridad entre las mismas. El tag no tiene por qué coincidir con el género.
- Puede utilizar el API (https://developers.themoviedb.org/3/getting-started) de The Movie Database (https://www.themoviedb.org/) para añadir funcionalidad adicional. Por ejemplo, puede mostrar una imagen de fondo relacionada con la película que consulta un usuario o completar la información de la película.
- Puede permitir que el usuario se cree listas personalizadas: de películas que le han gustado, de aquellas que tiene pendientes de ver o le interesan, etc.
- Puede hacer todo lo anterior de manera interactiva, mediante javascript, por ejemplo, permitiendo que el usuario arrastre la foto de una película a su lista para añadirla a la misma.
- Puede hacer que su aplicación muestre películas similares cada vez que el usuario consulte la ficha de una película. O que se le muestren usuarios similares cuando consulte su perfil.
- Puede ajustar el parámetro de regularización. Puede pedirle al usuario que puntúe su recomendación y usar esas puntuaciones para seleccionar el parámetro de regularización y comprobar si es efectivo este método. Puede comprobar si las recomendaciones varían al cambiar las puntuaciones.

3 Material a entregar y criterios de evaluación

Los alumnos deberán entregar una **memoria breve** del trabajo, además de **alojar el código** de la aplicación en su cuenta del laboratorio, de manera que quede accesible mediante el servidor HTTP. La entrega y evaluación se realizará en Enero. Se avisará con suficiente antelación de la fecha límite de entrega. Las **memorias** se subirán como entregable al **aula virtual**, por duplicado**,** es decir, cada miembro del grupo debe subir su propia copia de la memoria. **No se admitirá ningún trabajo en fechas posteriores a la fecha límite**. Es posible que en algún caso los profesores necesiten reunirse con los alumnos de un grupo para evaluar su trabajo. Para esos casos se publicará una lista de los grupos que deben entrevistarse con los profesores para explicar su trabajo.

3.1 Memoria y Código

La **memoria** debe contener:

- Nombre completo de los componentes del grupo, correo electrónico y, al menos, un teléfono de contacto.
- Índice
- URL de acceso a la aplicación.
- La memoria descrita en el apartado 2.1, describiendo la implementación de la aplicación.
- Una lista con todos los scripts PHP implementados, describiendo la funcionalidad que implementan.
- Una lista con todos los ficheros adicionales que haya utilizado (CSS, javascript), describiendo la funcionalidad que implementan.
- Si ha utilizado un framework para desarrollo PHP, CSS o Javascript, una descripción del uso que hace.
- Una descripción de cualquier funcionalidad adicional implementada.

El código se alojará en el directorio *public_html*, bajo un directorio *video* para que sea accesible al servidor HTTP del laboratorio, y **deberá ser completamente funcional, es decir, debe poder probarse la aplicación mediante un navegador**.

3.2 Criterios de evaluación

Aunque se ha mencionado antes, se resumen de nuevo los criterios de evaluación. La puntuación total del trabajo se ha repartido entre las distintas partes:

Apartado	Puntuación máxima
Funcionalidad mínima	7.5
Libre	2.5

Para la evaluación se tendrá en cuenta:

- Que se hayan implementado correctamente las funcionalidades necesarias.
- Se valorará la interpretación dada de los resultados del algoritmo de recomendación y se apreciará especialmente que los alumnos demuestren haber consultado bibliografía, Internet, etc, para mejorar sus conclusiones.
- Se valorará la claridad, rigor y concisión en las explicaciones aportadas en la memoria.
- Se valorará la claridad y orden en el código.
- Se valorará que se haya separado la funcionalidad en ficheros y funciones apropiadamente.
- El uso de estilos CSS. Aunque la calidad estética en sí de la aplicación es subjetiva y no se evaluará como tal, sí que se evaluará el uso adecuado de hojas de estilos y técnicas asociadas.

Se valorarán las extensiones de la funcionalidad y/o propuestas de otros algoritmos y el estudio de la influencia de distintos parámetros en el rendimiento. Las aportaciones se valorarán por su **originalidad, su calidad**, y su argumentación, **no por su cantidad ni su extensión**. Estas contribuciones pueden ser útiles para mejorar la **nota final de prácticas** de la asignatura.

3.3 Grupos de una persona

Al inicio de esta propuesta se indica que el trabajo debe realizarse en parejas, pero en ocasiones las circunstancias impiden formar o mantener un grupo y se queda una sola persona. Como no es justo aplicar los mismos criterios de evaluación, en estos casos se aplicará el siguiente reparto de puntos:

Grupos de una persona:

Apartado	Puntuación máxima
Funcionalidad mínima	8.5
Libre	1.5

4 Notas adicionales para la implementación

4.1 Implementación de la lógica en PHP

- Se recomienda realizar la implementación de la lógica de manera incremental por unidades funcionales. Por ejemplo, puede comenzar realizando el/los scripts que permiten la visualización y navegado por el catálogo de películas. A continuación, el/los scripts que permiten ver los detalles de una película y comentarla. Y así sucesivamente.
- Puede dejar el formato y estilo de su aplicación para el final, pero se recomienda que antes de comenzar la implementación decida al menos un *layout* u organización visual aproximada. De esa manera podrá agrupar en bloques los elementos y realizar unas cabeceras y pies comunes que se mostrarán en todas las páginas. Por ejemplo, puede decidir que su aplicación solo contendrá un bloque de cabecera y otro de contenidos.

4.2 Diseño alternativo de la base de datos

- Si considera que el diseño de la base de datos que se le proporciona no es adecuado, es libre de utilizar un diseño alternativo o añadir las tablas que considere necesario. Deberá describir y justificar brevemente su diseño alternativo en la memoria.

4.3 Uso de frameworks y librerías adicionales

- Si el alumno lo prefiere, se permite el desarrollo de la lógica de la aplicación mediante *frameworks* de desarrollo, como pueden ser *CodeIgniter*, *Zend*, o *CakePHP*, siempre y cuando estén basados en PHP. **SU USO ESTÁ SUJETO A APROBACION POR PARTE DE LOS PROFESORES**. Deberá enviar un correo electrónico al profesor Esteban Egea indicando el *framework* o librerías que se pretende utilizar. Tenga en cuenta que en algunos casos su uso puede ser contraproducente. Tenga en cuenta también que algunos *frameworks* proporcionan mecanismos específicos para la persistencia de datos, por lo que deberá describir en la memoria también la estructura de la base de datos en ese caso.
- Si se utiliza un *framework* la memoria debe incluir una además una descripción de: **instalación, configuración** y **funcionamiento y forma de uso** del *framework* así como una descripción de cómo se ha implementado la lógica de la aplicación en el contexto del framework.
- Se permite también el uso de **librerías para estilo y formato**, como puedan ser *jQuery* u otras, pero deberá **documentarse su uso** en la memoria.

5 Bibliografía

[1] Mung Chiang. ”Networked Life. 20 Questions and Answers”, Cambridge University Press, Cambridge, UK, 2012

6 Anexo 1. Estructura de la base de datos

A continuación se describen las tablas usadas en la base de datos. Los campos que aparecen en negrita actúan como clave primaria de la tabla. En las tablas **recs** y **user_score** hay dos campos que actúan como clave primaria, es decir, que en esas columnas pueden aparecer valores repetidos pero la combinación de los valores de ambas siempre tiene que ser única en la tabla.

6.1 Tabla genre

Almacena el género de las películas.

Column	Type	Null	Default
name	text	No	
id	tinyint(3)	No	

6.2 Tabla movie

Almacena la información de cada película.

Column	Type	Null	Default
id	int(10)	No	
title	text	No	
date	date	Yes	NULL
url_imdb	text	No	
url_pic	text	Yes	NULL
desc	text	Yes	NULL

6.3 Tabla moviecomments

Almacena los comentarios realizados a las películas.

Column	Type	Null	Default
movie_id	int(11)	No	
user_id	int(11)	No	
comment	text	No	

6.4 Tabla moviegenre

Almacena los géneros de cada película

Column	Type	Null	Default
movie_id	int(10)	No	
genre	tinyint(3)	No	0

6.5 Tabla recs

Almacena las recomendaciones para cada usuario, junto con la fecha en la que se creó la recomendación

Column	Type	Null	Default
user_id	int(11)	No	
movie_id	int(11)	No	
rec_score	double	No	
time	timestamp	No	CURRENT_TIMESTAMP

6.6 Tabla users

Almacena la información de cada usuario. **ATENCIÓN:** el campo *passwd* almacena un *hash* SHA1 de la clave del usuario.

Column	Type	Null	Default
id	int(10)	No	
name	text	No	
edad	tinyint(3)	Yes	NULL
sex	enum('M', 'F')	Yes	NULL
ocupacion	enum('administrator', 'artist', 'doctor', 'educator', 'engineer', 'entertainment', 'executive', 'healthcare', 'homemaker', 'lawyer', 'librarian', 'marketing', 'none', 'other', 'programmer', 'retired', 'salesman', 'scientist', 'student', 'technician', 'writer')	Yes	NULL
pic	text	Yes	NULL
passwd	varchar(40)	No	

6.7 Tabla user_score

Almacena las puntuaciones de los usuarios a las películas.

Column	Type	Null	Default
id_user	int(11)	No	
id_movie	int(11)	No	
score	tinyint(3)	Yes	NULL
time	timestamp	Yes	NULL

7 Anexo 2. Uso de Java en Matlab

Se proporciona aquí el código de un servidor como ejemplo de uso de Java en Matlab. También puede utilizar este script desde su propio Matlab para pruebas locales, es decir, sin utilizar el servidor del laboratorio.

```
import java.net.*;
import java.io.*;

serverSocket = ServerSocket(4450);

display('Iniciando servidor');

try
    while(1)
        socket = serverSocket.accept();
        display('Nueva conexion entrante');
        in = BufferedReader(InputStreamReader(socket.getInputStream()));
%Leemos el path del usuario
         pathstr = in.readLine();
%Leemos la function a ejecutar del usuario

        funcstr = in.readLine();
         out
=PrintWriter(BufferedWriter(OutputStreamWriter(socket.getOutputStream())),true)
;
          %Codigo que atiende la peticion
          %Establecemos el path del usuario con el valor dice el cliente
          userpath(char(pathstr));
            %Evaluamos la funcion que nos dice el cliente
                 status=eval(char(funcstr));
           %Devolvemos un valor de estado y cerramos
                 out.println(strcat(int2str(status),char(0)));
                out.flush();
                socket.shutdownOutput();
           %Restablecemos el path
                userpath('reset');

        display('cerrando conexion con cliente');
        socket.close()
    end
catch e
    e.message
    if(isa(e, 'matlab.exception.JavaException'))
        ex = e.ExceptionObject;
        ex.printStackTrace;
    end
    display('excepcion')
    socket.close();
    serverSocket.close();
end
serverSocket.close();
```

Observe que en el ejemplo anterior ha utilizado la sintaxis de Matlab para crear el ServerSocket (no aparece el operador *new* en la llamada al constructor) así como en el bucle *while* y ha utilizado la sintaxis de Java para invocar métodos del objeto Java, el ServerSocket. Observe como se convierten objetos String de Java a string de Matlab mediante *char*. Observe finalmente que, debido a ciertas limitaciones en el uso de Java sobre Matlab, el servidor no permite atender peticiones concurrentes.