Actividad Práctica 4 (UT3) – Aplicaciones interactivas

Logueo y registro de usuarios

1. Descripción de la actividad

Uno de los casos de uso más comunes de muchas aplicaciones web es la gestión de los usuarios. Cuando hablamos de la gestión de usuarios, en este caso, nos referimos a aquellos casos de uso que permiten a un usuario *loguearse* o *registrarse* en la aplicación o servicio que desea acceder.

A la hora de diseñar una aplicación, se pueden definir tres tipos de *home*, los cuales pueden condicionar la experiencia del usuario y por lo tanto se tiene que tener en cuenta su diseño desde el comienzo del desarrollo:

First User: Cuando el usuario entra por primera vez. Por ejemplo, vista/as del registro de un nuevo usuario (cuando no se sabe nada de él), vista de inicio de sesión etc.

Logger User: Suele considerarse la vista home, cuando el usuario ya está logueado o registrado y es la vista principal de la cual parte toda la usabilidad de la aplicación.

Last User: Vista de salida de la aplicación; cuando el usuario finaliza sesión o es derivado a terminar el flujo de la ejecución de la misma. Dentro de estas se deben tener en cuenta la vista de errores.

2. Requisitos de software

En esta actividad práctica, se solicita implementar una estructura básica que represente los tres tipos de home anteriormente mencionados, debiéndose acceder a la base de datos tanto para validar al usuario como para permitir a este registrarse por primera vez.

Se omitirá las operaciones de eliminación y modificación para simplificar esta actividad.

Para ello, se pone a disposición del alumno, un script denominado *commercedb.sql* con las tablas básicas que deberá implementar en una base de datos MySql (versión 5.7 o superior).

2.1. Flujo de los scripts de la aplicación

Ver documento *Anexo I – Flujo de scripts de aplicación.*

2.2. index.php

2.2.1. Lógica de negocio

1º) Si existe la cookie *ckdatauser* entonces redireccionará al script denominado *validateuser.php* por método get. En este caso, no será necesario enviar datos por get, dado que *validateuser.php* detectará la cookie y procederá a validar el usuario (ver apartado 2.3.).

2º) Si no existe la cookie *ckdatauser* entonces mostrará un formulario en el que se solicitará el *username* y *password* seguido de un botón denominado "*Entrar*". En la parte inferior se mostrará un enlace denominado "*Nueva cuenta*" que hará referencia al script *newuser.php*.

Además, si se detecta algún parámetro por método GET se entenderá que está recibiendo un código de error procedente de *validateuser.php* y se añadirá a la vista el mensaje de error de validación que crea oportuno.

2.2.2. Acciones/Casos de uso permitidos

a) Al hacer clic en "*Entrar*" enviará los datos por método post al script *validateuser.php*.

b) Al hacer clic en "*Nueva cuenta*" redireccionará por método get a newuser.php. No será necesario enviar datos por get.

2.3. validateuser.php

2.3.1. Lógica de negocio

1º) Si existe la cookie *ckdatauser* tomará de ella los datos del *customerid*.

2º) Si no existe la cookie *ckdatauser*, tomará los datos del *username* y *password* que reciba por el método post.

3º) Una vez disponga de los datos, bien sea por las anteriores condiciones 1º) o 2º), procederá a intentar crear el objeto *clsCliente* pasándole al constructor el *username* y *password* o pasandole al constructor el *customerid* (Ver apartado 3.2.2 y 3.2.4).

4º) Si con el método *IsError* se confirma un error en el objeto *clsCliente* entonces se obtendrá el código de error con *GetCodError* y se redireccionará por get enviando el error a *index.php*.

5º) Se deberá comprobar con el método *IsValidatedUser* si la validación ha sido satisfactoria.

Si ha sido satisfactoria, entonces creará la cookie *ckdatauser* (si no existe) con el dato del *customerid* y se creará la/s variables de sesión que estime oportuno asignando al menos a alguna de ellas el objeto *clsCliente* con los datos del cliente.

De lo contrario, si la validación no ha sido correcta, se eliminará la cookie *ckdatauser* y se redireccionará por método get a index.php informando del código de error oportuno.

Nota: Para más información de la clase *clsCliente* ver apartado 3.2.

2.4. home.php

2.4.1. Lógica de negocio

1º) Se tomará de la/s variable/s de sesión los datos del usuario (objeto *clsCliente*) y se mostrarán. Además, se añadirá un enlace o botón denominado "*Cerrar sesión*" que hará referencia al script *endsesion.php*.

2.4.2. Acciones/Casos de uso permitidos

a) Al hacer clic en "*Cerrar sesión*" redireccionará al script *endsesion.php* por el método get. No es necesario aportar datos por get dado que *endsesion.php* procederá automáticamente a eliminar la cookie y reiniciar o anular las variables de sesión (ver apartado 2.5).

2.5. endsesion.php

2.5.1. Lógica de negocio

1º) Eliminará la cookie *ckdatauser*
2º) Eliminará las variables de sesión del usuario.
3º) Redireccionará por get sin datos al script *index.php*

2.6. newuser.php

2.6.1. Lógica de negocio

1º) Eliminará la cookie *ckdatauser*.

2º) Se mostrará el formulario de recogida de datos que estime oportuno (en base al modelo de datos proporcionado) correspondiente a los campos de la tabla *customer*:

Usuario (username), Contraseña (password), Nombre (name), Apellido 1 (firstlastname), Apellido 2 (secondlastname), Fecha de nacimiento (birthdaydate), Calle (streetdirection), N.º (streetnumber), CP (provincecode), Municipio (cityid), Provincia (provinceid), país (countryid), Teléfono 1 (telephone1), teléfono 2 (telephone2), email.

Los componentes del formulario relativos a la ciudad, provincia o país previamente han de cargarse de los datos obtenidos de la base de datos. Es decir, el dominio de datos posible viene definido por las tablas city, country y province. El usuario no ha de escribirlos, sino seleccionar un valor permitido. (Ejemplo: combobox).

3º) Si se detecta por paso de parámetros get un error, se añadirá el mensaje al formulario anterior.

2.6.2. Acciones/Casos de uso permitidos

a) Al hacer clic en "*Enviar*" redireccionará al script *register.php* por método post.

b) Al hacer clic en "*Cancelar*" se redireccionará al script *index.php* por método get. No es necesario enviar parámetros por get.

2.7. register.php

2.7.1. Lógica de negocio

1º) Tomará los datos recibidos por el método post. Si hay algún dato que falta, entonces se redireccionará nuevamente a *newuser.php* informando del error de validación.

2º) Se creará el objeto *clsCliente* pasándole al constructor todos los datos recibidos. (Ver apartado 3.2.3).

3º) Si al invocar el método *IsError* se confirma un error del objeto, entonces redireccionará a *newuser.php* informando del código de error de validación oportuno que devolverá el método GetError.

Nota: Tal como se indica en el apartado 3.2.3, uno de los errores posibles es que el usuario ya exista.

4º) Si se ha superado todas las comprobaciones anteriores, entonces se entenderá que el usuario se ha registrado correctamente (IsValidatedUser = true e IsError = false):

- Se creará la cookie ckdatauser con el *customerid.*
- Se creará la/s variable/s de sesión que estime oportuno, con al menos en una de ellas el objeto clsCliente.

5º) Se redireccionará al script *home.php* por el método get sin parámetros.

3. Requisitos adicionales

Para dar soporte a los requisitos indicados en el apartado 2, se requiere crear y hacer uso al menos dos clases.

3.1. Creación de la clase clsConnection

Esta clase permitirá centralizar el acceso a la base de datos. En la aplicación, no podrá realizarse ningún acceso a la base de datos sin hacer instancia a un objeto de esta clase.

Deberá implementar como mínimo los métodos y atributos que se adjuntan en el fichero *ClsConnection.php*. Podrá añadir los métodos y atributos adicionales que necesite, como por ejemplo para realizar consultas o inserciones en la base de datos. O mejorar los métodos proporcionados.

3.2. Creación de la clase clsCliente

Esta clase permitirá centralizar los datos del cliente a modo de entidad de la tabla customer.

Deberá implementar como mínimo los métodos y atributos que se indican a continuación. Podrá añadir los métodos y atributos adicionales que necesite, como por ejemplo para acceder a los valores de los atributos privados, destructor, etc.

Para el acceso a datos, esta clase deberá instanciar a objetos de la clase *clsConnection* y llevar a cabo el paradigma de acceso a datos; *abrir conexión, ejecutar y cerrar conexión*.

Nota: Dada las especificaciones del objeto PDO de acceso a datos de PHP, bastará con asignar a nulo el objeto *clsConexion* para indicar a PHP que finaliza el ámbito del objeto PDO que implementa dentro de la clase *clsConexion*.

3.2.1. Atributos privados

a) Atributos relacionados con los datos del cliente/usuario

customerid, username, password, name, firstlastname, secondlastname, birthdaydate, streetdirection, streetnumber, provincecode, cityid, provinceid, countryid, telephone1, telephone2, email

b) Atributos de control

Error: de tipo booleano, indicará cualquier error.

CodError: de tipo string, indicará el código de error.

msgError: de tipo string, indicará el detalle del error.

Validated: de tipo booleano, indicará si se ha podido validar el usuario o creado su registro correctamente.

3.2.2. Constructor I

Esta sobrecarga del constructor, recuperará los datos de un usuario si existe de la tabla *customer*.

1º) Recibirá por parámetros el *username y password*.

2º) Haciendo uso del objeto *clsConnection* realizará una consulta a la base de datos para comprobar si el usuario y password son correctos. En la misma consulta, obtendrá los datos del usuario.

3º) Si el usuario existe, cargará todos los atributos privados correspondiente a los datos del cliente además de establecer el atributo *Validated* a true. Si no existiera el usuario, entonces establecerá a false el atributo *Validated*.

Cualquier error grave o problema de acceso a datos se ha de indicar estableciendo el atributo *Error* a true e indicando el código de error (*CodError*) y mensaje de error (*msgError*). Si no se produjera error, el atributo *Error* siempre tendrá asignado false y el atributo *CodError* y *msgError* tendrán la cadena vacía.

3.2.3. Constructor II

Esta sobrecarga del constructor, registrará un nuevo usuario en la tabla *customer*.

1º) Recibirá por parámetros el *username, password, name, firstlastname, secondlastname, birthdaydate, streetdirection, streetnumber, provincecode, cityid, provinceid, countryid, telephone1, telephone2, email*

2º) Haciendo uso del objeto *clsConnection* realizará una consulta a la base de datos para comprobar si el *username* o el *email* existen. Si existiera al menos alguno de los dos, se asignará con valor false a la variable *Validated* y a true la variable *Error*, además de indicar el *CodError* y mensaje de error en *msError* indicando que el usuario que se intenta crear ya existe.

Si no existe el username o el *email*, se realizará un insert en la tabla *customer*. Se asignará a true el atributo *Validated*. El atributo Error será false así como vacía la cadena asignada a *CodError* y *msgError*.

Cualquier error grave o problema de acceso a datos se ha de indicar estableciendo el atributo *Error* a true e indicando el código de error (*CodError*) y mensaje de error (*msgError*). Si no se produjera error, el atributo Error siempre tendrá asignado false y el atributo *CodError* y *msgError* tendrán la cadena vacía.

3.2.4. Constructor III

Esta sobrecarga del constructor, recuperará los datos de un usuario si existe de la tabla *customer* dado el identificador *customerid*.

1º) Recibirá por parámetros el identificador único de usuario*; customerid.*

2º) Haciendo uso del objeto *clsConnection* realizará una consulta a la base de datos para comprobar si el usuario existe. En la misma consulta, obtendrá los datos del usuario.

3º) Si el usuario existe, cargará todos los atributos privados correspondiente a los datos del cliente además de establecer el atributo *Validated* a true. Si no existiera el usuario, entonces establecerá a false el atributo *Validated*.

Cualquier error grave o problema de acceso a datos se ha de indicar estableciendo el atributo *Error* a true e indicando el código de error (*CodError*) y mensaje de error (*msgError*). Si no se produjera error, el atributo *Error* siempre tendrá asignado false y el atributo *CodError* y *msgError* tendrán la cadena vacía.

3.2.5. Método público IsError

Devolverá el valor del atributo Error.

3.2.6. Método público GetCodError

Devolverá el valor del atributo CodError.

3.2.7. Método público GetError

Devolverá el valor del atributo msgError.

3.2.8. Método público IsValidatedUser

Devolverá el valor del atributo Validated.

3.3. Funciones en bibliotecas

Todas las funciones auxiliares que se implementen en esta actividad o clases, deberán estar definidas al menos en una o varias bibliotecas (Pueden segmentar en varios ficheros las funciones según su familia etc).

3.3. Lenguaje

- *Lenguaje de desarrollo*: PHP. Las pruebas se realizarán con PHP v8 pero se puede desarrollar con la versión 7.4. Se recomienda en este último caso tener en cuenta posibles incompatibilidades:

https://www.php.net/manual/es/migration70.php
https://www.php.net/manual/es/migration80.php

4. Formato de entrega

La entrega deberá realizarse por el aula virtual empaquetado en un fichero .zip cuyo nombre de fichero debe corresponderse a las dos alternativas siguientes:

UT3_AP4_DWS_NombreApellidos.zip

El contenido del anterior fichero deberá estar estructurado con las siguientes carpetas:

- **www**: Carpeta donde estará todo el código desplegado y recursos de la aplicación, tal que copiando su contenido en un directorio virtual, deberá correr la aplicación web en el servidor apache2. Nota: tener en cuenta usar rutas relativas en el proyecto, por ejemplo, a la hora de hacer referencia a imágenes, .css u otros scripts .php.

- **source**: Código fuente completo del proyecto (estructura generado por el IDE de desarrollo).

- **doc**: Carpeta donde se podrá alojar de forma opcional algún documento (manual) o aclaración para su ejecución (pj: si necesita alguna parametrización de algún fichero, o sobre como abrir el proyecto en el IDE, ejecutarlo etc.).

- **resumen.txt**: Indicar la tecnologías y sus versiones utilizadas (Versión PHP, versión MySql, etc).

5. Criterios de evaluación

Los criterios de evaluación de esta actividad práctica tendrá en cuenta la siguiente calificación:

- Hasta 3 puntos la correcta implementación del apartado 2.2, 2.3, 2.4 y 2.5.

- Hasta 2 puntos, la correcta implementación del apartado 2.6 y 2.7.

- Hasta 2 puntos, la correcta implementación del apartado 3.1.

- Hasta 2 punto, la correcta implementación del apartado 3.2.

- Hasta 1 punto la maquetación y estilos.

6. Anexos

Anexo I: Flujo de scripts de aplicación.pdf