UNIVERSIDAD COMPLUTENSE
MADRID

APLICACIONES WEB
2018-2019
Grado en Ingeniería del Software
Facultad de Informática

Práctica guiada - Parte 5

Fecha de entrega: 29-nov-2018

El objetivo de esta parte es incorporar los atributos de sesión en la aplicación web de gestión de tareas. Para realizar el ejercicio es imprescindible haber realizado el anterior ya que consiste en realizar modificaciones sobre el mismo.

Creación de una página de login

Codificar una página **login.html** que contenga un formulario para gestionar el login de los usuarios de la aplicación (ver figura 1). Además del formulario la página debe tener un elemento que permita mostrar un mensaje como el que se ve en la figura escrito en color rojo. Dicha página tiene el mismo aspecto que la página principal de la aplicación pero no aparece información del usuario identificado ya que no hay ninguno que lo esté en ese momento.

Guardar el fichero **login.html** en la carpeta **public** del proyecto y arrancar el servidor para comprobar que accediendo a la dirección http://localhost:3000/login.html el formulario de login se visualiza correctamente.

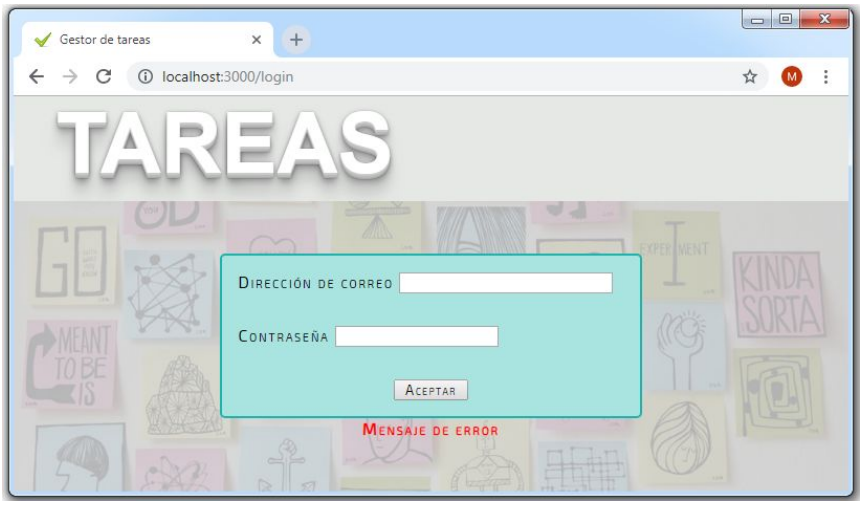

Figura 1: página de identificación de usuario

Introducir el middleware express-session

En primer lugar es necesario el middleware **express-session** para poder almacenar y obtener los atributos de sesión. Además, para almacenar de manera persistente los datos de sesión en la base de datos MySQL se necesita el middleware **express-mysql-session**.
Para utilizar y configurar ambos middlewares realizar los siguientes pasos:

1. Instalar en el proyecto los módulos **express-session** y **express-mysql-session.**
2. En el módulo principal **app.js** importar los módulos **express-session** y **express-mysql-session.**
3. Obtener la clase **MySQLStore**. Esta clase se obtiene llamando a la función exportada por el módulo **express-mysql-session**, pasándole como parámetro el objeto exportado por el módulo **express-session** (ver transaparencias).
4. Crear una instancia de **MySQLStore**, pasando a la constructora los datos de conexión a la BD. Estos datos se obtienen del módulo **config.js** (ver transaparencias).
5. Crear un middleware de sesión y añadirlo a la cadena de middlewares de la aplicación. Al especificar las opciones en la creación de este middleware se debe incluir la opción **store** con la instancia de **MySQLStore** obtenida en el paso anterior, para que los datos de sesión se guarden en la BD.

Tras hacer esto, reiniciar el servidor y comprobar que se haya creado una tabla vacía llamada **session** en la base de datos. En esta tabla se almacenarán los atributos de sesión de cada usuario.

Entrada y salida del sistema

La página de entrada al sistema (**login.html**) tiene que ser una página dinámica, ya que contiene un mensaje de error que deberá mostrarse o no en función de los datos de acceso que haya introducido el usuario. Por tanto, mover el fichero **login.html** al directorio de plantillas **views**, y renombrarlo a **login.ejs**. Modifica esta plantilla para que el mensaje de error dependa de una variable llamada **errorMsg**, que tomará el valor **null** en el caso en que no se quiera mostrar ningún mensaje de error.
Añadir un manejador de ruta GET **/login** a la aplicación, para que se muestre la vista **login** pasando el valor **null** a la variable **errorMsg**.
Cuando el usuario hace clic en el botón **[Aceptar]** del formulario de identificación, se realiza una petición POST a la ruta **/login** con la información introducida. Escribir el manejador de esta ruta para que se compruebe si la dirección de correo y contraseña introducidos se encuentran en la base de datos. Para realizar esta comprobación, utilizar el método **DAOUsers.isUserCorrect()** (recuperar el módulo **DAOUsers** de la parte 3 de esta práctica guiada).

- Si la dirección y contraseña son correctos, se debe establecer un atributo nuevo de sesión llamado **currentUser**, cuyo valor es la dirección de correo introducida. Después debe redirigirse a la ruta **/tasks**.
- En caso contrario, se debe volver a la vista **login** con el mensaje de error **Dirección de correo y/o contraseña no válidos**.

Dentro de la vista de tareas existe un enlace **[Desconectar]** que salta a la dirección **/logout.** Implementar el manejador de esta ruta para que se eliminen los atributos de sesión (utilizar, para ello, **request.session.destroy()**), y se redirija a la ruta **/login.**

Si quieres comprobar que los atributos de sesión se establecen correctamente, hacer login con un nombre de usuario y contraseña válidos. Debería añadirse una fila nueva a la tabla **sessions** de la base de datos. Tras hacer logout, esta fila debería eliminarse.

Middleware de control de acceso

Implementar un middleware que compruebe la existencia de un atributo llamado **currentUser** dentro de los atributos de sesión.

- Si se encuentra, el middleware debe añadir un nuevo atributo al objeto **response.locals** que contenga la dirección de correo del usuario actual. Por ejemplo, se puede llamar a este nuevo atributo **userEmail**. El objetivo de este atributo es poder hacer uso de la variable **userEmail** en todas las plantillas sin necesidad de pasarla explícitamente como modelo en cada llamada a la función **response.render()**. Todos los atributos que se inserten en el objeto **response.locals** estarán automáticamente disponibles en todas las vistas. Tras hacer esto, se debe saltar al siguiente middleware de la cadena.
- Si no se encuentra, se debe redirigir a la página **/login** sin saltar al siguiente middleware.

Una vez implementado este middleware, identificar todas aquellas rutas de la aplicación que requieran identificación: **/tasks**, **/finish**, **/addTas**k, **/deleteCompleted**. Añadir este middleware a todas estas rutas, de modo que sólo se permita su acceso a aquellos usuarios que estén identificados en el sistema.

Información específica de usuario

En la parte anterior de esta práctica guiada, todas las llamadas a **getAllTasks()**, **insertTask()** y **deleteCompleted()** se realizaban sobre las tareas del usuario concreto **usuario@ucm.es**. Modificar estas llamadas para que el usuario sobre el cual se apliquen sea el que esté actualmente identificado en el sistema.
Además, modificar la vista de tareas para que muestre el nombre del usuario actualmente identificado utilizando la variable **userEmail**.

Imagen de perfil

La imagen de perfil del usuario actualmente identificado se debe mostrar en la zona de datos de usuario. Las imágenes de perfil se guardan en la carpeta **profile_imgs** situada dentro de la carpeta de proyecto (crearla en este punto). Insertar en dicha carpeta imágenes de perfil para los usuarios que figuren en la base de datos y actualizar de manera adecuada la base de datos. Dejar algún usuario sin imagen de perfil.

A continuación, en la vista **tareas.ejs**, en el elemento <img> que contiene la imagen de perfil, sustituir el atributo src="..." por **src="/imagenUsuario"**.

Añadir a la aplicación un nuevo manejador de ruta **/imagenUsuario**. Este manejador debe:

- Buscar en la BD el nombre de la imagen de perfil correspondiente al usuario actualmente identificado. Utilizar para ello la función **getUserImageName** del módulo **DAOUsers**.
- Si el usuario no tiene imagen de perfil, la operación anterior devolverá **null**. En este caso llamar al método **response.sendFile()** pasando como argumento la ruta al fichero **NoPerfil.jpg** que se encuentra en la carpeta **img** de los recursos estáticos.
- Si el usuario tiene imagen de perfil, utilizar el nombre de fichero devuelto para llamar al método **response.sendFile()**.

Entrega

La fecha límite para la entrega de esta práctica es el jueves 29 de noviembre de 2018 (16:00). Se entregará a través del CV en un fichero pxx.zip (siendo xx el número de pareja de laboratorio) que contenga todos los ficheros necesarios para reconstruir el proyecto.