2ª ENTREGA SDI – RED SOCIAL

Daniel Duque Barrientos – UO245553

Contenido

Implementación de los casos de uso

Caso 1: Registrarse como usuario

Implementé en este caso de uso una vista que consta con un formulario en el que hay que introducir los siguientes campos: Nombre, Email, Contraseña, Repetición contraseña.

Realicé un validador que comprueba que ninguno de los campos está vacío, que el email tiene el formato correcto (*alguien@ejemplo.com*) y que las contraseñas tienen También se valida que no exista el email introducido en la base de datos y que la contraseña se la misma que la confirmación.

Una vez validados los campos se crea el usuario y se almacena en la base de datos y se redirige al login para que el usuario pueda iniciar sesión.

Caso 2: Iniciar sesión

Implementé una vista con dos campos: Email y contraseña. Al introducir los datos se realiza una petición *POST* en la que se añaden al cuerpo de la petición el valor de los dos campos. En el módulo *gestorBD* cree una función que retorna de la base de datos un usuario a partir de un criterio pasado como parámetro. En caso de que se devuelva un usuario se realiza el inicio de sesión, en caso contrario se envía a la vista un mensaje de error.

Caso 3: Listar todos los usuarios de la aplicación

Esta vista se carga después de realizar un inicio de sesión correcto. Se carga una vista la cuál consta de una tabla en la que se muestran los usuarios del sistema a excepción del usuario que inició sesión. Se puede acceder a esta vista también desde un enlace de la barra de navegación. Los usuarios se muestran paginados, con un total de 5 usuarios por página.

Para esto se realiza una petición *GET* en la cual se retorna la lista de usuarios dependiendo de la página actual que se envía como parámetro de la petición y con la función creada previamente en el módulo *gestorBD* en el caso 2.

Una vez obtenida la lista de usuarios se muestra en la vista, añadiendo además un botón de solicitud de amistad.

Caso 4: Buscar entre todos los usuarios de la aplicación

Se añade a la vista del caso 3 un nuevo formulario que añade a la petición un nuevo parámetro que consiste en la cadena con la que se quiere realizar la búsqueda de los usuarios. En el criterio que se le pasa la función del módulo *gestorBD* se le añade un atributo en el cual se especifica que el nombre o el email tiene que contener esa cadena.

La función retorna la lista de usuarios que cumplen es criterio y posteriormente se muestra en la vista del caso 3.

Caso 5: Enviar una petición de amistad a un usuario

Al pulsar el botón “Añadir amigo” creado en la vista del caso 3 se crea una petición *GET* en la que se envía el id del usuario al que se quiere añadir como parámetro de la petición.

Mediante una función en el módulo *gestorBD* se almacena una relación entre el usuario que está en sesión y el usuario con el id que se recibió en la petición (previamente obteniéndolo de la base de datos). Además, se crea un campo *status* que indica el tipo de relación que tienen dos usuarios.

REQUEST -> Petición enviada

FRIEND -> Amigos (recíproco)

Caso 6: Listar peticiones de amistad recibidas

Implementé una vista la cual muestra en una tabla una lista de peticiones de amistad paginada, con cinco peticiones por página. Para ello cree una función que recibe una petición *GET* que se encarga de añadir a la vista la lista de peticiones de amistad.

Para obtener las peticiones implemente una función en el módulo *gestorBD* que, en base a un criterio, retorna una lista de relaciones paginada, dependiendo de la página actual pasada como parámetro en la petición. En dicho criterio se especifica que el usuario en sesión tiene que coincidir con el campo *recipient* y el campo *status* tiene que coincidir con REQUEST.

Caso 7: Aceptar una petición recibida

En la vista implementada en el caso 6 se añade un botón “Aceptar” a cada petición. Al hacer clic se crea una petición *GET* en la que se pasa como parámetro el id del usuario de la petición de la amistad. En la función que trata la petición se llama a una función del módulo *gestorBD* que, en base a un criterio se modifica una relación de la base de datos. Para el usuario en sesión y el usuario de la petición de amistad se modifica el campo “status” de la relación a FRIEND. Además, se crea una nueva relación si no existe en el otro sentido, es decir, entre el usuario de la petición de amistad y el usuario en sesión. Cuando no se produce ningún error se muestra un mensaje en la vista de que se ha añadido a un nuevo amigo y el botón de la vista del caso 3 cambia y ya no se puede pulsar.

Caso 8: Listar usuarios amigos

A partir de una petición *GET* y una función del módulo *gestorBD* se obtiene una lista con los amigos para el usuario en sesión. En esa función se le pasa como parámetro un criterio en el cual se especifica que el usuario en sesión tiene que coincidir con el campo “sender” y que el campo “status” tiene que coincidir con FRIEND. La lista obtenida de envía a la vista en la que se muestra de forma paginada, con cinco usuarios por página.

Caso S.1: Autenticación del usuario

Implementé una función que trata una petición *POST* en la cual se recibe un email y una contraseña en el cuerpo de la petición. Con una función del módulo *gestorBD* compruebo si existe un usuario con ese email y contraseña, y dependiendo de si existe o no la respuesta es 401(Unauthorized) o 200(correcto). En la respuesta también incluyo un token que va a identificar al usuario en sesión. El token está formado a partir del email del usuario y de la fecha actual. Este token tiene que formar parte de la cabecera de cualquier petición.

Caso S.2: listar todos los amigos

Implementé una función que trata una petición *POST* en la cual se recibe un campo de texto en el cuerpo que trata de la cadena de filtro de usuarios. Dependiendo de si el campo del filtro está vacío o no se declara un criterio que se le pasa como parámetro a una función del módulo *gestorBD* que, en base del criterio, retorna la lista de usuarios correspondiente (todos o los usuarios que coincidan con el filtro). Además, en dicho criterio se especifica que el campo "status" tiene que coincidir con FRIEND. En caso de que se procese correctamente la función se envía como respuesta la lista de usuarios.

Caso S.3: Crear un mensaje

Implementé una función que trata una petición *POST* en la cual se recibe el usuario destino del mensaje y el texto de este. Primero compruebo que el usuario en sesión y el de destino son amigos mediante una función del módulo *gestorBD.* Si son amigos se llama a otra función del módulo *gestorBD* que recibe como parámetro el mensaje creado (emisor, destino, texto, leído) y lo almacena en la base de datos.

Caso S.4: Obtener mis mensajes de una "conversación"

Implementé una función que trata una petición *GET* en la cual se recibe el usuario como parámetro de la petición. Dado que no puede haber mensajes de entre dos usuarios que no son amigos gracias al caso S.3, únicamente, mediante una función del módulo *gestorBD* y en base a un criterio en el cual se especifica el emisor (usuario en sesión o usuario destino) y destino (usuario en sesión o usuario destino) se retorna la lista de mensajes correspondiente.

Caso C.1: Autenticación del usuario

Implementé una vista que consta de dos campos: email y contraseña. También hay un botón que al pulsarlo se genera una petición ajax, la cuál se trata en el caso S.1. En el cuerpo de la petición ajax de incluye el email y la contraseña de los campos de la vista. Si el resultado es correcto (200) se carga la vista de la lista de usuarios amigos (Caso C.2). En caso contrario se muestra un mensaje de error. El token generado se guarda en una cookie para poder añadirlo en la cabecera de cada petición que se realice.

Caso C.2: Mostrar lista de amigos

Implementé una vista que muestra una tabla con la lista de usuarios amigos del usuario en sesión. Para ello se realiza una petición ajax en la cual se recibe la lista de

usuarios amigos. En el cuerpo de la petición ajax se incluye el campo del valor del filtro. En caso de que se procese correctamente la petición (Caso S.2) se recibe la lista de usuarios amigos. Mediante una función auxiliar se añade al código HTML para poder mostrar cada usuario de la lista junto a un botón que permite acceder a la conversación con ese usuario.

Caso C.3: Mostrar los mensajes

Implementé una vista que se encarga de mostrar los mensajes de una conversación entre el usuario en sesión y el usuario seleccionado de la lista de amigos. Para ello se realiza una petición ajax en la cual se incluye como parámetro de la petición el usuario de la conversación (Caso S.3). Si la petición se trata correctamente se recibe la lista de mensajes de la conversación. Mediante una función auxiliar se crea bloques diferentes dependiendo de quien es el emisor del mensaje.

Para la actualización automática de la conversación creé un temporizador que cada 5 segundos realiza una llamada a la petición ajax. Además, guardo el número de mensajes de la conversación, así si no se producen cambios en la conversación no se generan de nuevo los bloques de los mensajes.

Caso C.4: Crear mensaje

En la vista del caso C.3 añadí un nuevo campo para poder introducir el texto del mensaje a crear. También consta de un botón que al pulsarlo se crea una petición ajax el cuál incluye en su cuerpo el texto del mensaje y el usuario destino (Caso S.4). Como hay una actualización automática de los mensajes, y al producirse una inserción se actualizan los bloques de los mensajes.

Implementación de las pruebas

1.1 RegVal

Registro de un usuario con datos válidos y comprobar que se introduce en sesión y se muestra la vista principal

1.2 RegInVal

Se prueba que no se permite registrar un usuario cuando la contraseña y la repetición de la contraseña no coinciden. Se comprueba que se muestra el error en la vista.

2.1 InVal

Se comprueba que se realiza el correcto inicio de sesión con un usuario existente en la base de datos y proporcionando la contraseña correcta. Se comprueba que se muestra la vista principal una vez se realiza el login.

2.2 InInVal

Se comprueba que no se puede realizar el inicio de sesión con un usuario que no existe en la base de datos. Se comprueba que se muestra el mensaje de error.

3.1 LisUsrVal

Se comprueba que se puede acceder a la lista de usuarios desde la opción de la barra de navegación. Se comprueba que se muestra el número correcto de usuarios.

3.2 LisUsrInVal

Se comprueba que no se puede acceder a la lista de usuarios introduciendo la *URL* que redirecciona a dicha lista. Se comprueba que automáticamente el sistema nos redirecciona a la vista de inicio de sesión.

4.1 BusUsrVal

Se comprueba que se realiza la correcta búsqueda de un usuario en el buscador de la vista de la lista de usuarios. Se comprueba que la tabla cuenta con las filas correctas.

4.2 BusUsrInVal

Se comprueba que no se puede acceder a la vista que contiene el buscador. Se comprueba que automáticamente el sistema nos redirecciona a la vista de inicio de sesión.

5.1 InvVal

Se comprueba que se realiza una petición de amistad correctamente. Se comprueba que una vez realizada la petición el botón se deshabilita.

5.2 InvInVal

Se comprueba que una vez realizada la petición el botón se deshabilita.

6.1 LisInvVal

Se realiza primero una petición de amistad a un usuario. A continuación, se inicia sesión con dicho usuario. Se comprueba que accediendo al listado de peticiones de amistad se obtiene una tabla con una fila.

7.1 AcepInvVal

Se realiza primero una petición de amistad a un usuario. A continuación, se inicia sesión con dicho usuario. Se comprueba que accediendo al listado de peticiones de amistad se obtiene una tabla con una fila. Se comprueba que al aceptar la petición ésta desaparece de la tabla y se muestra un mensaje de que no hay peticiones de amistad pendientes.

8.1 ListAmiVal

Se realiza el proceso de añadir un amigo. Una vez añadido se accede a la lista de usuarios amigos y se comprueba que la tabla consta de una fila. Se comprueba con el otro usuario que también contiene una fila la tabla en la vista de los usuarios amigos.

C.1.1 CInVal

Se comprueba que se realiza el correcto inicio de sesión con un usuario existente en la base de datos y proporcionando la contraseña correcta. Se comprueba que se muestra la vista principal una vez se realiza el login.

C.1.2 CInInVal

Se comprueba que no se puede realizar el inicio de sesión con un usuario que no existe en la base de datos. Se comprueba que se muestra el mensaje de error.

C.2.1 CListAmiVal

Añado 3 amigos a un usuario para comprobar que se muestran correctamente en la lista de usuarios amigos.

C.2.2 CListAmiFil

Añado 3 amigos a un usuario y realizo un filtrado por un usuario, comprobando que efectivamente se muestra un usuario.

C.3.1 CListMenVal

Compruebo que en una conversación entre el usuario en sesión y un usuario amigo se pueden ver los mensajes entre ellos.

C.4.1 CCrearMenVal

Compruebo que en una conversación entre el usuario en sesión y un usuario amigo se pueden crear mensajes y se muestran en la conversación.