2ª Entrega sdi – Red social

Daniel Duque Barrientos – UO245553

Contenido

[Implementación de los casos de uso 3](#_Toc509768530)

[Caso 1: Registrarse como usuario 3](#_Toc509768531)

[Caso 2: Iniciar sesión 3](#_Toc509768532)

[Caso 3: Listar todos los usuarios de la aplicación 3](#_Toc509768533)

[Caso 4: Buscar entre todos los usuarios de la aplicación 3](#_Toc509768534)

[Caso 5: Enviar una petición de amistad a un usuario 4](#_Toc509768535)

[Caso 6: Listar peticiones de amistad recibidas 4](#_Toc509768536)

[Caso 7: Aceptar una petición recibida 4](#_Toc509768537)

[Caso 8: Listar usuarios amigos 5](#_Toc509768538)

[Caso 9: Crear una nueva publicación 5](#_Toc509768539)

[Caso 10: Listar mis publicaciones 5](#_Toc509768540)

[Caso 11: Listar las publicaciones de un usuario amigo 5](#_Toc509768541)

[Caso 12: Crear una publicación con una foto adjunta 6](#_Toc509768542)

[Caso 13: Iniciar sesión como administrador: 6](#_Toc509768543)

[Caso 14: Listar todos los usuarios de la aplicación 6](#_Toc509768544)

[Caso 15: Eliminar usuario 6](#_Toc509768545)

[Implementación de las pruebas 7](#_Toc509768546)

[1.1 RegVal 7](#_Toc509768547)

[1.2 RegInVal 7](#_Toc509768548)

[2.1 InVal 7](#_Toc509768549)

[2.2 InInVal 7](#_Toc509768550)

[3.1 LisUsrVal 7](#_Toc509768551)

[3.2 LisUsrInVal 7](#_Toc509768552)

[4.1 BusUsrVal 7](#_Toc509768553)

[4.2 BusUsrInVal 7](#_Toc509768554)

[5.1 InvVal 7](#_Toc509768555)

[5.2 InvInVal 7](#_Toc509768556)

[6.1 LisInvVal 8](#_Toc509768557)

[7.1 AcepInvVal 8](#_Toc509768558)

[8.1 ListAmiVal 8](#_Toc509768559)

[9.1 PubVal 8](#_Toc509768560)

[10.1 LisPubVal 8](#_Toc509768561)

[11.1 LisPubAmiVal 8](#_Toc509768562)

[11.2 LisPubAmiInVal 8](#_Toc509768563)

[13.1 AdInVal 8](#_Toc509768564)

[13.2 AdInInVal 8](#_Toc509768565)

[14.1 AdLisUsrVal 8](#_Toc509768566)

[15.1 AdBorUsrVal 9](#_Toc509768567)

[15.2 AdBorUsrInVal 9](#_Toc509768568)

# 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*](mailto: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

Implemente 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 click 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 C.1: Autenticación del usuario

# 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.