SDI – Sistemas Distribuidos e Internet

ENUNCIADO PRÁCTICA 2 – NODE.Js & Servicios Web

INFORME

Grupo 207-211

Nombre1:	Alejandro (50%participación)
Apellidos1:	González Meléndez
Email1:	uo265111@uniovi.es/alejan1579@gmail.com
Cód. ID GIT	207
Nombre1:	Clara (50%participación)
Apellidos1:	Miranda García
Email1:	UO264958@uniovi.es
Cód. ID GIT	211

Indice

1.Introducción

1.1 Breve descripción de la aplicación:

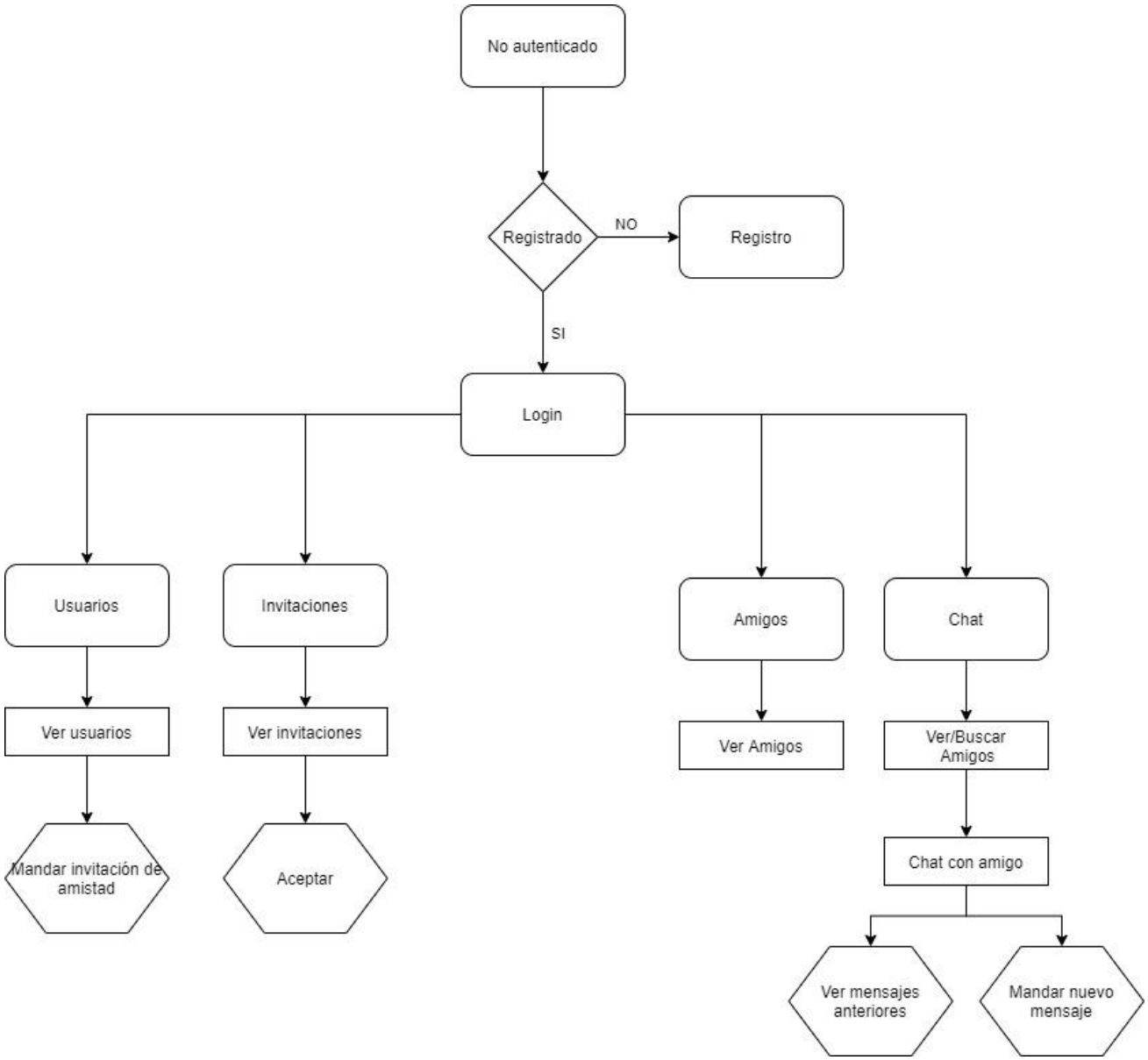
En esta entrega hemos desarrollado una red social utilizando NodeJS, que gestiona desde el registro, login, etc del usuario hasta el manejo de sus amigos. Además se implementaros y consumieron diferentes servicios web REST desarrollados con NodeJS para permitir un chat entre los distintos amigos existentes.

1.2 Separación de requisitos:

Para poder realizar esta aplicación, se dividió la carga de trabajo de la siguiente manera:

Alejandro	W-1,2,3,10 S-4,5 C-3,5,7
Clara	W-4,5,6,7,8,9 S-1,2,3 C-1,2,4,6

2.Mapa de navegación



3.Aspectos técnicos y de diseño relevantes

3.1 Peticiones de amistad.

Para realizar esta funcionalidad se decidió tener una tabla en mongodb llamada "invitations" que manejase las peticiones de amistad. Un usuario podría enviar peticiones de amistad a otros usuarios, y una vez aceptadas, se convertirían en amigos. Esta tabla contendría un id compuesto, formado por los emails que representan a ambos usuarios.

Se consideró que las peticiones de amistad eran bidireccionales, esto es, una vez que el usuario A enviase una petición de amistad al usuario B, existiría la petición genérica con ambos ids de los usuarios, y se trataría de la misma manera si el usuario A quisiera enviar la petición otra vez al usuario B , que si el usuario B intentase enviar una petición de amistad al usuario A.

En la tabla "invitations" habría una entrada por petición enviada, formada por los ids de ambos usuarios que conforman esta relación.

3.2 Amigos.

Esta funcionalidad siguió la misma línea de diseño que la anterior. De nuevo se decidió crear una tabla nueva en la base de datos, al considerarse que eran distintos los conceptos de amigo y de petición de amistad. Al aceptarse una petición de amistad, los usuarios involucrados se convierten en amigos, desapareciendo esa invitación de amistad.

4. Casos de uso.

W1 Público: Registrarse como usuario

Se añade la vista de registro junto con su respectiva opción en la barra de navegación. Contamos con validación en el servidor y la ofrecida por defecto en cliente. Guardamos la contraseña como un hash en la base de datos por motivos de seguridad.

W2 Público: Iniciar sesión

Creación de la vista para inicio de sesión junto con la debida opción en la barra de navegación. Igual que antes, validamos los datos y nos aseguramos de que el login es legítimo comparándolo contra la base de datos.

W3 Usuario Registrado: Fin de sesión

Añadimos la opción para desconectarnos en la barra de navegación, la cual nos aseguramos de que muestre la opción de desconexión cuando tenemos un usuario conectado y las opciones de registro y identificación cuando no sea así.

W4 Usuario registrado: Listar todos los usuarios de la aplicación

Para realizar esta funcionalidad se incluyó en la barra de navegación una opción para usuarios autenticados que mostrase paginadamente los distintos usuarios de la aplicación. Por usuario se mostraría la información del nombre, apellidos y el email.

W5 Usuario registrado: Buscar entre todos los usuarios de la aplicación

En la vista de la funcionalidad W4 donde se muestra el listado de usuarios, se añade una opción por la que mediante un cuadro de búsqueda se puede buscar dinámicamente a los usuarios en los que su nombre, email, o apellidos cumpliesen con el criterio de búsqueda.

W6 Usuario registrado: Enviar una invitación de amistad a un usuario

En la vista de la funcionalidad W4 donde se muestra el listado de usuarios, se añade un enlace "Añadir amigo" que permite enviarle al usuario en esa fila determinada una invitación de amistad. En el caso de que no se pueda enviar la petición, ya sea porque ya existe una petición o porque se intenta seleccionar al mismo usuario autenticado, se notifica mediante un mensaje en la parte superior de la vista.

W7 Usuario registrado: Listar las invitaciones de amistad recibidas

Se incluyó en la barra de navegación una opción para usuarios autenticados que mostrase paginadamente las distintas invitaciones que el usuario autenticado recibió. Si no hubiese ninguna invitación para el usuario, se muestra un mensaje notificándolo. Para cada invitación se muestra el nombre y apellidos del usuario que realizó la petición.

W7 Usuario registrado: Listar las invitaciones de amistad recibidas

Se incluyó en la barra de navegación una opción para usuarios autenticados que mostrase paginadamente las distintas invitaciones que el usuario autenticado recibió. Si no hubiese ninguna invitación para el usuario, se muestra un mensaje notificándolo. Para cada invitación se muestra el nombre y apellidos del usuario que realizó la petición.

W8 Usuario registrado: Aceptar una invitación recibida

En la vista que presenta las distintas invitaciones, se incluye un enlace "Añadir" por invitación que permite añadir al usuario como amigo. Se detalla más esta funcionalidad en el apartado 3.

W9 Usuario registrado: Listar los usuarios amigos

De nuevo, se añade una opción visible para el usuario autenticado en la barra de navegación que permite acceder al listado paginado de amigos del usuario autenticado. Para cada usuario amigo se muestra nombre, apellidos y email del mismo.

W10 Seguridad

Utilizamos routers para asegurarnos de que un usuario no puede acceder a urls indebidas y así evitamos que este tenga acceso a datos que no le pertenecen.

En ciertas situaciones como el contemplado en la prueba 22, solucionamos el problema eliminando parámetros de urls y obteniendo la información necesaria desde la sesión, esto implica que un usuario nunca podrá acceder a por ejemplo una lista de amigos de otro usuario.

Como mencionamos previamente guardamos las contraseñas como hashes en la base de datos.

Parte 2A - Implementación de la API de Servicios Web REST

S1 Identificarse con usuario - token

Para poder permitir al usuario autenticarse en la aplicación, además de la parte de la vista donde se especifican los campos a rellenar y se realiza la llamada a la petición POST de autenticar, se crea una respuesta a dicha petición que, de ser los campos correctos, genera un token que es devuelto como respuesta y que va a ser utilizado para el resto de los métodos de esta sección de la aplicación. Se crea el router RouterUsuarioToken para verificar el token.

S2 Usuario identificado: Listar todos los amigos

Además del componente del widget-friendships que compone la vista de este apartado, donde se muestra la información de los amigos del usuario autenticado, se realiza una petición GET al servicio web para poder mostrar todos los amigos. Si hay algún problema se cambiará el estado de la petición y se notificará.

S3 Usuario identificado: Crear un mensaje

Se responde a esta petición POST creando un mensaje con leído por defecto a "false" y el resto de los parámetros obteniéndolos del cuerpo de la petición y del usuario en sesión. Se realizan varias comprobaciones, como que son amigos los usuarios que establecen esta comunicación y si de verdad existen en la base de datos.

S4 Usuario identificado: Obtener mis mensajes de una "conversación"

Creamos un servicio que permite a un usuario obtener todos los mensajes de una conversación. Estos mensajes serán aquellos en los cuales tanto emisor como receptor se correspondan con los parámetros. En caso de que solo se mande un parámetro, se asumirá que el segundo es el usuario en sesión.

S5 Usuario identificado: Marcar mensaje como leído

Servicio que recibe un identificador de un mensaje para cual el usuario identificado debe ser el receptor. Marca el mensaje como leído.

Parte 2B - Cliente - Aplicación jQuery

C1. Autenticación del usuario

En esta sección, se utiliza el objeto $.ajax para identificar al usuario con un email y una contraseña, esta petición devuelve un token de autenticación que es usado para comprobar que el usuario está autenticado.

C2. Mostrar la lista de amigos

Al autenticarse el usuario es redirigido a esta vista, la cual contiene una tabla con cuerpo vacío. Se realiza la función de cargar el listado de amigos en el script y se actualiza la tabla con los datos recibidos.

C3. Mostrar los mensajes

En conjunción con S4 implementamos una simple vista que muestre los resultados de la petición, mostrando los detalles de los mensajes. Esta lista se actualiza en tiempo real, por lo cual no es necesario el refresco.

C4. Crear mensaje

Para esta sección se añade en la vista donde se muestra el chat determinado un cuadro de texto donde se podrá poner lo que se quiere comentar y un botón de enviar que llamará al correspondiente método de gestionar la creación del comentario, que realiza una petición POST a la url correspondiente donde se pasa en el cuerpo el texto del comentario y el usuario al que se realiza.

C5. Marcar mensajes como leídos de forma automática

Cuando en la vista C3 un usuario vea los mensajes pertenecientes a una conversación estos serán marcados automáticamente como leídos. Solo se marcarán como leídos aquellos para los que el usuario en sesión es el receptor.

C6 Mostrar el número de mensajes sin leer

En la vista donde aparecen todos los usuarios amigos del usuario autenticado se muestra a su lado una columna "Sin leer" donde se verían los mensajes sin leer con otro usuario. Al cargar los datos de la tabla, se llamaría un POST a la url que permite el amigo con un campo de mensaje sin leer que se corresponde con los mensajes sin leer del usuario, Por cada usuario se actualiza la tabla. Dicha tabla se actualizaría cada poco para poder ver nuevos mensajes sin leer.

5. Información necesaria para el despliegue y ejecución

Se importa el proyecto desde la línea de comandos con:

git clone https://github.com/ClaraMirandaGarcia/sdi1920-entrega2-207-211.git

Tras ello, se importa desde WebStorm:

File -> Open

Y se selecciona la url a donde se ha importado el proyecto.

Para abrir el proyecto de tests':

File -> Import -> Existing proyect

Se necesita importar la librería Selenium desde:

Lib-> Selenium

También se deberá cambiar la ruta en el test SocialNetworkTest de Firefox

Tras ello se podrán ejecutar los tests correctamente.

6 Conclusión

En este proyecto se ha creado una aplicación de red social utilizando las distintas tecnologías previamente mencionadas siguiendo pautas similares a las utilizadas en clase, permitiendo profundizar en los conceptos aprendidos.

También se han tenido en cuenta distintos problemas a la hora de desarrollar la aplicación, como puede ser la seguridad de la aplicación, y gracias a ello se ha conseguido tener un mejor entendimiento de la seguridad de las aplicaciones web y otros aspectos de la misma.