Sistemas Distribuidos e Internet
Curso 2021/2022

PRÁCTICA 2 DE ENTREGA – Node.js y Servicios web
MY SOCIALNETWORK

Consideraciones sobre la evaluación de requisitos

A lo largo del enunciado se detallan los requisitos obligatorios y opcionales de la práctica de entrega. Para cada requisito, se detalla una funcionalidad y las pruebas automatizadas que deben implementarse correctamente para validarlo.

Para que el trabajo sea evaluado por el profesorado**,** todos los **requisitos obligatorios deben estar implementados, así como todas las pruebas automatizadas asociadas al mismo.** La puntuación máxima a la que se opta por implementar a la perfección los requisitos obligatorios **es de 8.0 puntos.** La puntuación establecida para cada requisito es una puntuación máxima, **pudiendo llegar a ser negativa**, afectando a la puntuación total de la práctica**.** La **responsabilidad** de que no se cumpla lo especificado en este párrafo es, **en cualquier tipo de caso, grupal y no individual.**

Los **requisitos opcionales permiten alcanzar una puntuación máxima de 11.0 puntos**, siempre y cuando se implementen todos a la perfección a nivel de funcionalidad y pruebas automatizas. En perfección, se incluyen aspectos tales como: validación, registro en log, etc. **La puntuación mínima de un ejercicio opcional será de 0.0 puntos**.

Prueba de autoría: Aquellos alumnos que sean requeridos por el profesorado, deberán realizar una defensa presencial del trabajo en el laboratorio de prácticas, consistente en la implementación de nuevos casos de uso. La nota de la práctica se verá condicionada a la correcta implementación de esos casos de uso. El listado de alumnos y fecha de dicha prueba se publicará en el CV con suficiente antelación.

Coevaluación holística

La calificación final asignada a cada integrante de un equipo (Nota Individual del Estudiante, NIE) se calculará mediante el método denominado coevaluación holística: coevaluación, porque cada alumno va a valorar el rendimiento y comportamiento de sus compañeros, así como de uno mismo, y holístico porque esa valoración deberá ser una medida global (actitud, cumplimiento de plazos, …) de sus compañeros en lo que compete al desarrollo de esta práctica.

Con el fin de simplificar ese cálculo cada alumno asignará una valoración entre varios niveles de posibles de Ciudadanía de Equipo (CE) a sus compañeros. Los valores de la escala Liker de CE van desde:

- "EXCELENTE" (100%): Contribución muy destacada y constante en el trabajo de equipo, con un rendimiento sobresaliente, hasta
- NO MOSTRADO (0%): No jugó un papel efectivo en el trabajo en equipo y/o asistencia y compromisos virtualmente inexistentes.

De esta forma cada alumno emitirá un listado de N valoraciones siendo un equipo de N miembros. Con la matriz de NxN valores de un equipo, el profesor correspondiente calculará para cada alumno de dicho equipo el Factor Individual de Coevaluación (FIC). De esta forma se calculará la Nota Individual de cada Estudiante (**NIE**) a partir del FIC y de la Nota obtenida por el Equipo en la práctica entregada (NE):

$$NIE = FIC * NE$$

Se enviará un mensaje desde el CV indicando cual será el procedimiento tanto para enviar los niveles de CE. Además, en clase se explicará con algo más de detalle cómo se realizará el cálculo del FIC.

Instrucciones generales

La práctica consta de dos partes más la documentación y el peso de cada parte es el siguiente:

- Parte 1 "aplicación web" 5/11 puntos.
- Parte 2 "servicios web" 5/11 puntos.
- Documentación 1/11 punto.

Para que el trabajo pueda ser evaluado los **Requisitos obligatorios** deben estar implementados, con sus correspondientes pruebas automatizadas ***(no se evaluarán los casos de uso que no contengan pruebas).*** La puntuación máxima a la que se opta por implementar todos los requisitos obligatorios de forma perfecta es de 7.0 sobre 11, mientras que los requisitos opcionales implementados de forma perfecta llegarán hasta 3.0 puntos sobre 11 (ver Tabla 1).

Tabla 1. Detalle de la puntuación de la práctica

			Puntos
PARTE1 - APLICACIÓN WEB			**5,00**
	REQUISITOS OBLIGATORIOS		**5,00**
	W1	Perfil Público: registrarse como usuario	0,20
	W2	Perfil Público: iniciar sesión	0,10
	W3	Usuario Admon/Registrado: Fin de sesión	0,10
	W4	Usuario Admon: Ver el listado de usuarios del sistema	0,20
	W5	Usuario Admon: Borrado múltiple de usuarios	0,40
	W6	Usuario registrado: Ver el listado de usuarios de la red social	0,20
	W7	Usuario registrado: buscar entre todos los usuarios de la aplicación	0,40
	W8	Usuario registrado: enviar una invitación de amistad a un usuario	0,40
	W9	Usuario registrado: listar las invitaciones de amistad recibidas	0,40
	W10	Usuario registrado: aceptar una invitación recibida	0,40
	W11	Usuario registrado: Ver listado de amigos	0,40
	W12	Usuario Estándar: Crear una nueva publicación	0,40
	W13	Usuario Estándar: Ver el listado de publicaciones propias	0,20
	W14	Usuario Estándar: Ver el listado de publicaciones de un amigo	0,20
	W15	Seguridad	1,00
PARTE2 - SERVICIOS WEB			**5,00**
	PARTE2A - REQUISITOS OBLIGATORIOS		**1,00**
	S1	Identificarse con usuario – token	0,40
	S2	Usuario identificado: listar todos los amigos	0,20
	S3	Usuario identificado: Crear un mensaje	0,20
	S4	Usuario identificado: Obtener mis mensajes de una "conversación"	0,20
	PARTE2A - REQUISITOS OPCIONALES		**0,40**
	*S5	Usuario identificado: Marcar mensaje como leído	0,40
	S6		
	S7		
	PARTE2B - REQUISITOS OBLIGATORIOS		**1,00**
	C1	Autenticación del usuario	0,40
	C2	Mostrar la lista de amigos	0,10
	C3	Mostrar los mensajes	0,10
	C4	Crear mensaje	0,40
	PARTE2B - REQUISITOS OPCIONALES		**2,60**
	*C5	Marcar mensajes como leídos de forma automática	1,00
	*C6	Mostrar el número de mensajes sin leer	0,60
	*C7	Ordenar la lista de amigos por último mensaje	1,00
INFORME OBLIGATORIO			**1,00**
TOTAL			**11,00**

RESUMEN	
REQUISITOS OBLIGATORIOS	**7,00**
REQUISITOS OPCIONALES	**3,00**
INFORME OBLIGATORIO	**1,00**
TOTAL	**11,00**

Notas:

- ***Es requisito indispensable utilizar una base de datos Mongo en la nube (Mongo Cloud).***

Parte1 "Aplicación Web"

Se trata de desarrollar una aplicación Web de tipo red social en el que existirán perfiles de usuario de tipo: Público (Anónimo) y Usuario Registrado (Administrador y Usuario Estándar). En esta parte todos los requisitos son obligatorios. A continuación, se detallan los requisitos:

W1 Público: Registrarse como usuario

Los usuarios deben poder registrarse en la aplicación aportando un email, nombre, apellidos y una contraseña (que deberá repetirse dos veces y coincidir entre sí).

El email del usuario no podrá estar repetido en el sistema, se debe informar al usuario de los errores en el proceso de registro.

Pruebas Funcionales

[Prueba1] Registro de Usuario con datos válidos.

[Prueba2] Registro de Usuario con datos inválidos (email vacío, nombre vacío, apellidos vacíos).

[Prueba3] Registro de Usuario con datos inválidos (repetición de contraseña inválida).

[Prueba4] Registro de Usuario con datos inválidos (email existente).

W2 Público: Iniciar sesión

Suministrando su email y contraseña, un usuario podrá autenticarse ante el sistema. Sólo los usuarios que proporcionen correctamente su email y su contraseña podrán iniciar sesión con éxito.

En caso de que el inicio de sesión fracase, será necesario mostrar un mensaje de error indicando el problema.

A continuación, se detalla cómo se debe proceder para cada uno de los perfiles existentes.

Caso 1: Usuario con perfil de administrador

- **Sólo existirá un usuario administrador en el sistema con email admin@email.com y contraseña admin**. Podemos indicar que un usuario es administrador incluyendo un campo extra en sus datos. Se puede crear un usuario administrador de prueba e insertarlo desde código o directamente en la base de datos. **No será posible el registro de usuarios con perfil de Administrador a través de la aplicación**.
- En caso de que el inicio de sesión sea correcto se debe redirigir al usuario a la vista: "lista todos los usuarios de la aplicación" [Requisito Obligatorio 4].

Caso 2: Usuario Estándar

- En caso de que el inicio de sesión sea correcto se debe redirigir al usuario a la vista "Ver listado de usuarios de la red social" [Requisito Obligatorio 6].

Pruebas Funcionales

[Prueba5] Inicio de sesión con datos válidos (administrador).

[Prueba6] Inicio de sesión con datos válidos (usuario estándar).

[Prueba7] Inicio de sesión con datos inválidos (usuario estándar, campo email y contraseña vacíos).

[Prueba8] Inicio de sesión con datos válidos (usuario estándar, email existente, pero contraseña incorrecta).

W3 Usuario Registrado: Fin de sesión

Incluir en el menú de navegación una opción que permita finalizar la sesión, enviando al usuario al formulario de Inicio de sesión. Este botón solo se mostrará exclusivamente si un usuario se ha autenticado previamente.

Pruebas Funcionales

[Prueba9] Hacer clic en la opción de salir de sesión y comprobar que se redirige a la página de inicio de sesión (Login).

[Prueba10] Comprobar que el botón cerrar sesión no está visible si el usuario no está autenticado.

W4 Usuario Administrador: Listado de usuarios del sistema

Un usuario identificado con perfil de Administrador debe poder acceder a una lista en la que figuren todos los usuarios de la aplicación. Para cada usuario se mostrará su email, nombre y apellidos.

No es necesario incluir sistema de paginación en este listado.

Pruebas Funcionales

[Prueba11] Mostrar el listado de usuarios y comprobar que se muestran todos los que existen en el sistema.

W5 Usuario Administrador: Borrado múltiple de usuarios

En la vista del [Requisito Obligatorio W4] donde figuran todos los usuarios del sistema se debe poder seleccionar múltiples usuarios (con un checkbox) y se dispondrá de un botón **"Eliminar"** para confirmar el borrado de todos aquellos seleccionados. Al pulsar el botón de Eliminar se deben eliminar todos los usuarios seleccionados, así como toda la información relativa a los mismos, véase: datos, publicaciones, amistades, etcétera. Un usuario Administrador no podrá borrarse a sí mismo.

Pruebas Funcionales

[Prueba12] Ir a la lista de usuarios, borrar el primer usuario de la lista, comprobar que la lista se actualiza y dicho usuario desaparece.

[Prueba13] Ir a la lista de usuarios, borrar el último usuario de la lista, comprobar que la lista se actualiza y dicho usuario desaparece.

[Prueba14] Ir a la lista de usuarios, borrar 3 usuarios, comprobar que la lista se actualiza y dichos usuarios desaparecen.

W6 Usuario registrado: Listado de usuarios

Se visualizará en una lista todos los usuarios de la aplicación (excepto Administrador y el usuario autenticado). Para cada usuario se mostrará su nombre, apellidos e email.

La lista debe incluir un sistema de paginación y mostrar al menos 3 páginas y 5 usuarios por página.

Esta funcionalidad deberá ser accesible mediante una opción de menú principal, visible exclusivamente para usuarios autenticados.

Pruebas Funcionales

[Prueba15] Mostrar el listado de usuarios y comprobar que se muestran todos los que existen en el sistema, excepto el propio usuario y aquellos que sean Administradores.

W7 Usuario registrado: Buscar entre todos los usuarios de la aplicación

Incluir un sistema que permita realizar una búsqueda por nombre, apellidos y email de usuario. El cuadro de búsqueda contendrá un único campo de texto. La cadena introducida en ese campo se utilizará para buscar coincidencias tanto en el campo nombre y apellidos como en el email de los usuarios. Por ejemplo, si escribimos la cadena "mar" deberá retornar usuarios en los que la cadena "mar" sea parte de su nombre, apellidos o su email.

El resultado de la búsqueda debe ser una lista que muestre las coincidencias encontradas, esta lista debe incluir un sistema de paginación y mostrar 5 usuarios por página.

Pruebas Funcionales

[Prueba16] Hacer una búsqueda con el campo vacío y comprobar que se muestra la página que corresponde con el listado usuarios existentes en el sistema.

[Prueba17] Hacer una búsqueda escribiendo en el campo un texto que no exista y comprobar que se muestra la página que corresponde, con la lista de usuarios vacía.

[Prueba18] Hacer una búsqueda con un texto específico y comprobar que se muestra la página que corresponde, con la lista de usuarios en los que el texto especificado sea parte de su nombre, apellidos o de su email.

W8 Usuario registrado: Enviar una invitación de amistad a un usuario

Junto a la información de cada usuario presente en la vista de "listar todos los usuarios de la aplicación" debe aparecer un botón con el texto "agregar amigo". Al pulsar el botón, el usuario en sesión enviará una invitación de amistad a ese usuario. Un usuario no podrá enviarse una invitación de amistad así mismo. Hay que validar la invitación del lado del servidor (No vale sólo con ocultar el botón de envío).

Pruebas Funcionales

[Prueba19] Desde el listado de usuarios de la aplicación, enviar una invitación de amistad a un usuario. Comprobar que la solicitud de amistad aparece en el listado de invitaciones (punto siguiente).

[Prueba20] Desde el listado de usuarios de la aplicación, enviar una invitación de amistad a un usuario al que ya le habíamos enviado la invitación previamente. No debería dejarnos enviar la invitación, se podría ocultar el botón de enviar invitación o notificar que ya había sido enviada previamente.

W9 Usuario registrado: Listar las invitaciones de amistad recibidas

Se visualizará en una lista todas las invitaciones de amistad recibidas por el usuario identificado en sesión. Para cada invitación se mostrará el nombre y apellidos del usuario de la aplicación que solicitó la amistad.

Debe incluirse una opción en el menú principal, visible solo para usuarios en sesión, que permita acceder al listado de todas las invitaciones de amistad recibidas.

La lista de invitaciones debe incluir un sistema de paginación y mostrar 5 invitaciones por página.

Pruebas Funcionales

[Prueba21] Mostrar el listado de invitaciones de amistad recibidas. Comprobar con un listado que contenga varias invitaciones recibidas.

W10 Usuario registrado: Aceptar una invitación recibida

Junto a cada invitación mostrada en la lista de invitaciones de amistad recibidas, se debe incluir un botón/enlace con el texto "aceptar". Al pulsar este botón/enlace, la invitación de amistad debe desaparecer de la lista de invitaciones y el usuario que la envió pasará a ser un amigo del usuario en sesión y viceversa (A es amigo de B, B es amigo de A). Una vez creada la relación de amistad, no pueden existir peticiones pendientes entre A y B.

Pruebas Funcionales

[Prueba22] Sobre el listado de invitaciones recibidas. Hacer clic en el botón/enlace de una de ellas y comprobar que dicha solicitud desaparece del listado de invitaciones.

W11 Usuario registrado: Listar los usuarios amigos

Se visualizará en una lista todos los usuarios amigos del usuario en sesión. Para cada usuario se mostrará su nombre, apellidos y email.

La lista de amigos debe incluir un sistema de paginación y mostrar 5 usuarios por página.

Debe incluirse una opción en el menú principal, visible solo para usuarios en sesión, que permita acceder al listado de todos los amigos del usuario en sesión.

[Prueba23] Mostrar el listado de amigos de un usuario. Comprobar que el listado contiene los amigos que deben ser.

W12 Usuario registrado: Crear una nueva publicación

Se debe incluir una nueva vista con un formulario para que un usuario autenticado pueda crear una nueva publicación. Las publicaciones consistirán en un título y un texto. Al crear la publicación se debe almacenar: qué usuario realizo la publicación, la fecha en la que se creó, el título y el texto asociados a la publicación.

Debe incluirse una opción en el menú principal, visible solo para usuarios autenticados, que permita acceder al formulario de crear publicaciones.

Pruebas Funcionales

[Prueba24] Ir al formulario crear publicaciones, rellenarla con datos válidos y pulsar el botón Submit. Comprobar que la publicación sale en el listado de publicaciones de dicho usuario.

[Prueba25] Ir al formulario de crear publicaciones, rellenarla con datos inválidos (campo título vacío) y pulsar el botón Submit. Comprobar que se muestra el mensaje de campo obligatorio.

W13 Usuario registrado: Listado de publicaciones propias

Se visualizará en una lista todas las publicaciones realizadas por el usuario autenticado. Para cada publicación se mostrará: la fecha en la que se creó, el título y el texto de la publicación.

Es necesario incluir sistema de paginación en este listado.

Debe incluirse una opción en el menú principal, visible solo para usuarios autenticados, que permita acceder al a la lista personal de publicaciones.

Pruebas Funcionales

[Prueba26] Mostrar el listado de publicaciones de un usuario y comprobar que se muestran todas las que existen para dicho usuario.

W14 Usuario registrado: Listado de publicaciones de un amigo

Se modificará la vista que muestra la lista con los amigos del usuario autenticado para poder acceder a las publicaciones de un amigo. Al pulsar sobre el nombre del amigo, se redirigirá a una lista donde se visualicen todas sus publicaciones. Para cada publicación se mostrará: la fecha en la que se creó, el título y el texto de la misma.

Es necesario incluir sistema de paginación en este listado.

Pruebas Funcionales

[Prueba27] Mostrar el listado de publicaciones de un usuario amigo y comprobar que se muestran todas las que existen para dicho usuario.

[Prueba28] Utilizando un acceso vía URL u otra alternativa, tratar de listar las publicaciones de un usuario que no sea amigo del usuario identificado en sesión. Comprobar que el sistema da un error de autorización.

W15 Seguridad

Deberá diseñarse adecuadamente la política de seguridad con los mecanismos de seguridad de Node.Js para que no existan situaciones de vulnerabilidad en el acceso a recursos/acciones de usuarios registrados.

Pruebas Funcionales

[Prueba29] Intentar acceder sin estar autenticado a la opción de listado de usuarios. Se deberá volver al formulario de login.

[Prueba30] Intentar acceder sin estar autenticado a la opción de listado de invitaciones de amistad recibida de un usuario estándar. Se deberá volver al formulario de login.

[Prueba31] Intentar acceder estando autenticado como usuario standard a la lista de amigos de otro usuario. Se deberá mostrar un mensaje de acción indebida.

Parte 2 “servicios web”

Se deberá implementar los servicios web REST correspondientes a una aplicación de mensajería/chat similar al WhatsApp que permitirá enviar y recibir mensajes de otros usuarios amigos. Estos servicios web deberán de estar contenidos en la aplicación web anterior (no hay que desarrollar una nueva aplicación). Además de los servicios se implementará también una aplicación jQuery-AJAX que los consuma. Esta aplicación debe permitir el intercambio de mensajes en tiempo real. El cliente también se incluirá dentro del mismo proyecto, en la carpeta “public”.

Los casos de usos S1 – S4 y C1 – C4 son de carácter obligatorio, para que el trabajo pueda ser evaluado. Además, los casos de uso **C1-C4 deben incluir sus correspondientes pruebas automatizadas** (no se evaluarán los casos de uso que no contengan pruebas). La puntuación máxima a la que se opta por implementar estos 8 casos de forma perfecta es de 5 sobre 10 dentro de la parte 2 “servicios web”. Mientras que **los casos de uso S1 – S4 no requerirán pruebas automatizadas**.

Los casos de uso S5 y C5 – C7 son de carácter opcional, la puntuación máxima a la que se opta por implementar todos los casos de forma perfecta es de 5 sobre 10 dentro de la parte 2 “servicios web”

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

S1 Identificarse con usuario – token

El servicio recibe las credenciales de un usuario (email y contraseña). En caso de que exista coincidencia con algún usuario almacenado en la base de datos retornará un token de autenticación. Si no hay coincidencia, retornará un mensaje de error de inicio de sesión.

Pruebas Funcionales

En este caso de uso no será necesario realizar las pruebas funcionales.

S2 Usuario identificado: Listar todos los amigos

El servicio retornará una lista con **exclusivamente los identificadores, nombre y apellidos** de todos los amigos del usuario identificado. La lista se enviará ordenada alfabéticamente por nombre.

Para permitir listar los amigos, el usuario debe de estar identificado en la aplicación. Por lo tanto, la petición debe contener un token de seguridad válido.

Pruebas Funcionales

En este caso de uso no será necesario realizar las pruebas funcionales.

S3 Usuario identificado: Crear un mensaje

El servicio debe permitir crear nuevos mensajes.

Las propiedades del mensaje son: emisor, destinatario, texto, leído (true/false). Por defecto, el mensaje se crea como no leído.

Para permitir crear un nuevo mensaje, el usuario destinatario debe ser amigo del usuario identificado. Por lo tanto, la petición debe contener un token de seguridad válido y, posteriormente, se validará que exista la relación de amistad. La API debe devolver el mensaje de error correspondiente en caso de no cumplirse cualquiera de las dos condiciones anteriores.

Pruebas Funcionales

En este caso de uso no será necesario realizar las pruebas funcionales.

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

El servicio recibe el identificador de dos usuarios y retorna una lista con todos los mensajes en los que esos usuarios son emisor y receptor o viceversa (mensajes enviados en ambos sentidos para esos dos usuarios).

Para permitir obtener los mensajes de una conversación, el usuario identificado debe ser el emisor o receptor de los mensajes. Por lo tanto, la petición debe contener un token de seguridad valido.

Pruebas Funcionales

En este caso de uso no será necesario realizar las pruebas funcionales.

* S5 Usuario identificado: Marcar mensaje como leído

Crear un servicio REST que permita marcar un mensaje como leído. La propiedad leído debe tomar el valor true. El servicio recibe el identificador del mensaje.

Para permitir cambiar el estado de un mensaje, el usuario identificado debe ser el receptor del mensaje. Por lo tanto, la petición debe contener un token de seguridad valido.

Pruebas Funcionales

En este caso de uso no será necesario realizar las pruebas funcionales.

Parte 2B - Cliente - Aplicación jQuery

C1. Autenticación del usuario

Debe permitir la autenticación a través de un formulario donde se solicite el correo y la contraseña de usuario. Se debe informar si la autenticación no se realiza con éxito.

Pruebas Funcionales

[Prueba32] Inicio de sesión con datos válidos.

[Prueba33] Inicio de sesión con datos inválidos (usuario no existente en la aplicación).

C2. Mostrar la lista de amigos

Una vez autenticado, se debe redirigir al usuario a su lista de amigos (no aplicar ningún sistema de paginación).

Pruebas Funcionales

[Prueba34] Acceder a la lista de amigos de un usuario, que al menos tenga tres amigos.

[Prueba35] Acceder a la lista de amigos de un usuario, y realizar un filtrado para encontrar a un amigo concreto, el nombre a buscar debe coincidir con el de un amigo.

C3. Mostrar los mensajes

Debajo de cada amigo se deberá incluir un enlace con la fecha, hora y texto del último mensaje, en caso de que haya mensajes anteriores, en otro caso no se deberá mostrar el enlace. Y al pulsar sobre dicho enlace se deberá mostrar el chat, el cual incluye:

- La lista con todos los mensajes relacionados con ese usuario y el usuario identificado en la aplicación.
- Formulario para enviarle un nuevo mensaje a ese usuario

La lista de mensajes debe actualizarse en tiempo real, sin necesidad de recargar la página ni de pulsar ningún botón. Es decir, debe haber un proceso automático que compruebe cada segundo si hay nuevos mensajes en esa conversación.

Pruebas Funcionales

[Prueba36] Acceder a la lista de mensajes de un amigo, la lista debe contener al menos tres mensajes.

C4. Crear mensaje

Desde la vista de chat, el usuario debe poder crear un nuevo mensaje para ese amigo. Siendo el usuario el emisor y el amigo el destinatario.

Pruebas Funcionales

[Prueba37] Acceder a la lista de mensajes de un amigo y crear un nuevo mensaje. Validar que el mensaje aparece en la lista de mensajes.

*C5. Marcar mensajes como leídos de forma automática

Al entrar en un chat de un amigo, todos los mensajes no leídos deben ser marcados como leídos. Junto a cada mensaje mostrado en el chat debe incluirse un <leído> al final (si tiene el estado leído).

Al igual que la lista de mensajes, el estado debe actualizarse en tiempo real sin necesidad de recargar la página ni de pulsar ningún botón. Es decir, debe haber un proceso automático que compruebe cada segundo si hay nuevos mensajes en esa conversación / o cambios en los estados leído.

Pruebas Funcionales

[Prueba38] Identificarse en la aplicación y enviar un mensaje a un amigo. Validar que el mensaje enviado aparece en el chat. Identificarse después con el usuario que recibió el mensaje y validar que tiene un mensaje sin leer. Entrar en el chat y comprobar que el mensaje pasa a tener el estado leído.

*C6 Mostrar el número de mensajes sin leer

Se debe mostrar junto al nombre de cada amigo (en la lista de amigos) cuántos mensajes sin leer hay en esa conversación.

El número de mensajes sin leer debe actualizarse en tiempo real, sin necesidad de recargar la página ni de pulsar ningún botón. Es decir, debe haber un proceso automático que compruebe cada segundo si hay nuevos mensajes.

Pruebas Funcionales

[Prueba39] Identificarse en la aplicación y enviar tres mensajes a un amigo. Validar que los mensajes enviados aparecen en el chat. Identificarse después con el usuario que recibido el mensaje y validar que el número de mensajes sin leer aparece en la propia lista de amigos.

*C7 Ordenar la lista de amigos por último mensaje

Se debe incluir un mecanismo de ordenación automático de la lista de amigos, de forma que los amigos se ordenen por la antigüedad en la creación del último mensaje, de más recientes a más antiguos (teniendo en cuenta mensajes tanto enviados como recibidos).La ordenación automática quiere decir que cada vez que se acceda a la opción C2/C3 se deberá recibir la lista ordenada por el criterio indicado desde el servidor.Pruebas Funcionales

[Prueba40] Identificarse con un usuario A que al menos tenga 3 amigos. Ir al chat del último amigo de la lista y enviarle un mensaje. Volver a la lista de amigos y comprobar que el usuario al que se le ha enviado el mensaje está en primera posición. Identificarse con el usuario B y enviarle un mensaje al usuario A. Volver a identificarse con el usuario A y ver que el usuario que acaba de mandarle el mensaje es el primero en su lista de amigos.

Informe y catálogo de casos de prueba

Se deberá entregar un informe técnico en formato PDF, así como un catálogo de casos de prueba en formato XLSX (que a su vez es formulario de autoevaluación). **Todos estos ficheros son OBLIGATORIOS**. Se suministran dos plantillas que deberán tomarse como base para la elaboración de dichos documentos:

- Una plantilla Word para el informe (sdi-entrega2-n.docx). (Se entregará en formato PDF).
- Una plantilla Excel para los casos de prueba (sdi-entrega2-n.xlsx). (Se entregará en formato Excel).

Ambos documentos deberán ser renombrados cambiando la n por el código del equipo asignado al equipo (ej. sdi-entrega2-21.pdf).

El contenido del informe deberá ser el siguiente:

- En la portada se deberá incluir:
  - Los datos personales de los autores: Nombre y apellidos, Emails de UNIOVI, Código ID GIT de cada uno y código del equipo.
- Diagrama del modelado de las entidades y sus relaciones. Debe incluirse también un texto que explique las relaciones y las decisiones más relevantes que ha tomado el equipo en el modelaje.
- Diagrama de navegabilidad con texto explicativo. Un diagrama de navegabilidad no se compone de capturas de pantalla de la interfaz. Pueden usarse como apoyo en el texto explicativo, pero nunca como sustitutivo.
- Una descripción clara y detallada aquellos aspectos técnicos y/o diseño que se consideren relevantes a nivel de servicios, controladores, repositorios, vistas y sus relaciones. El objetivo no es explicar qué son las capas y otros aspectos vistos en teoría. El objetivo es desarrollar qué decisiones se han tomado en base a lo visto en teoría.
- Cualquier otra información necesaria para una descripción razonablemente detallada de lo entregado y su correcto despliegue y ejecución (versiones de Node.js, mongoDB, etc**.). Obligatorio usar las versiones del software usado en clase.**
- Conclusión individual de cada miembro del equipo. Esto incluye la aportación en el desarrollo de la práctica, así como las dificultades encontradas durante el mismo. También qué ventajas y desventajas se han detectado a nivel técnico y a nivel de trabajo en equipo.

La plantilla suministrada para cumplimentar los casos de prueba realizados presenta dos hojas:

- Una (denominada Pruebas) con una serie de tablas, de las cuales el alumno deberá rellenar aquella indicada en amarillo (Casos de Prueba) y donde deberá reflejar los casos de prueba que haya diseñado, el estado de dicho caso de prueba y una explicación/aclaración sobre el mismo en caso de fallo o no haber rellenado el caso. También en el caso de incluir casos de prueba extra deberá incluir en la columna explicación/aclaración en que consiste dicha prueba.
- Una segunda hoja (Instrucciones) con las instrucciones para cumplimentar la hoja primera.

Pruebas automatizadas

Se deberá suministrar un proyecto Java JUnit5 con un mínimo de pruebas unitarias empleando el framework Selenium (Se suministra un proyecto plantilla de ejemplo .\sdi-entrega2-test-n).

Para cada una de estas pruebas se debe añadir al menos un caso de prueba y en caso de querer añadir más casos se añadirá numeración extra al nombre de la prueba. Por ejemplo. para la prueba PR06, el primer caso de prueba será el método: PR06, y los extra que se deseen añadir se enumerarán como PR06_1, PR06_2,… .

En caso de desear incluir más se deberán incluir al final de la clase de pruebas JUnit.

El protocolo de prueba

Para probar cada proyecto el profesor realizará los siguientes pasos:

a) Importar y ejecutar la aplicación Node.Js

b) Importará y ejecutar el proyecto JUnit en Intellij IDE.

Aspecto Generales

Seguridad

Deberán tenerse en cuenta los siguientes aspectos de seguridad:

- Emplear la técnica de autentificación/autorización más adecuada a este contexto.
- En caso necesario, comprobar el permiso de cada usuario para acceder a las URL o recursos solicitados.
- Registrar el acceso de los usuarios y la actividad en un Logger (por ejemplo, log4js).

Arquitectura

La aplicación deberá estar obligatoriamente diseñada siguiendo el patrón arquitectónico visto en clase. La utilización incorrecta de elementos de esta arquitectura será fuertemente penalizada (por ejemplo, implementar parte de la lógica de negocio de una entidad en un controlador que gestiona otra entidad diferente.).

Los servicios web REST deben seguir la arquitectura RESTful. Se deben utilizar URLs y métodos HTTP adecuados en cada caso.

Otros aspectos que serán evaluados

- Claridad y calidad de la implementación del código JavaScript
- Calidad de la implementación de las vistas y usando todas las funcionalidades vistas en clase.

Datos en la base de datos para las pruebas y tiempo de ejecución de las pruebas

Se deberá poblar la base datos al arrancar la aplicación con un mínimo de datos que permite ejecutar las pruebas solicitadas:

- Un único usuario administrador con las siguientes credenciales:
  - Login: **admin@email.com**
  - Password: **admin**
- Usuarios registrados: al menos 15 (para que permitan al menos un listado de usuarios con 3 páginas de 5 usuarios por páginas). Seguir el patrón de nombrado siguiente:
  - Login user01@email.com, user02@email.com, ….
  - Password: user01
- Publicaciones por usuario: al menos 10 publicaciones por usuario que permitan un listado con páginas 5 publicaciones.

Un aspecto a tener en cuenta es que las pruebas no deben depender del resultado las anteriores. Por ello deben pensarse en cómo poblar adecuadamente la base de datos para que cada prueba parte de un conjunto de datos conocido. Por otro lado, debe buscarse un compromiso entre este aspecto y el tiempo de ejecución de las pruebas.

Proceso de entrega y protocolo de prueba

Según el número asignado a cada grupo, se deberá crear dos proyectos en IntelliJ: un proyecto Node.Js y un proyecto Java/JUnit5 con los respectivos nombres **sdi-entrega2-n y sdi-entrega2-test-n,** donde el número n de cada trabajo será el identificador del equipo (publicado en el CV).

Según lo anterior la entrega consistirá en los siguientes puntos:

1. Subir los proyectos Node.Js y JUnit a un repositorio GIT con nombre **sdi-entrega2-n**. E invitar al usuario **sdigithubuniovi** como colaborador. Sobre dicho repositorio deberán realizarse actualizaciones frecuentes para que se pueda hacer un seguimiento del ritmo de trabajo.
2. Subir a la tarea del Campus Virtual correspondiente un archivo ZIP (usando el formato ZIP) con el nombre **sdi-entrega2-n.zip** (en minúsculas) y que deberá contener en su raíz:
    - El INFORME OBLIGATORIO en formato PDF con nombre **sdi-entrega2-n.pdf,** que contenga:
        - Una descripción clara y detallada de la implementación de cada uno de los casos de uso implementados.
        - Cualquier otra información necesaria para una descripción razonablemente detallada de lo entregado y su correcto despliegue y ejecución.
    - El CATÁLOGO de CASOS DE PRUEBA con nombre sdi-entrega2-test-n.xlsx y rellenado con los casos de pruebas realizados, fallados y no realizados, así como los comentarios oportunos.
    - El proyecto **Node.Js** en formato carpeta (no comprimido) con el nombre **.\sdi-entrega2-n**
    - El proyecto **Java/Junit5** exportado desde STS (no comprimido) con el nombre **.\sdi-entrega2-test-n.**
    - En resumen, el zip deberá contener en su raíz:
        - sdi-**entrega2**-n.pdf (Archivo PDF suministrado y renombrado cambiado la n).
        - sdi-**entrega2**-n.xlsx (Archivo Excel suministrado y renombrado cambiado la n).
        - sdi-**entrega2**-n (Carpeta del proyecto Node.Js renombrada cambiado la n)
        - sdi-**entrega2**-test-n (Carpeta del proyecto Junit renombrada cambiado la n)

Fecha máxima de entrega

La fecha límite de entrega es el día 09 de mayo a las 23:55. No se recibirá ningún trabajo fuera de plazo.

Evaluación

Se penalizará:

- Que la compilación del código genere "*warnings*".
- Que se presenten problemas durante el despliegue.