Esta práctica se podrá realizar en equipos de hasta 2 alumnos (pertenecientes al mismo grupo de prácticas) y su entrega es obligatoria.

La práctica consta de dos partes con el siguiente peso:

- Parte 1 "aplicación web" 3/11 puntos
- Parte 2 "servicios web" 7/11 puntos
- Documentación 1/11 punto

*Defensa del trabajo presentado: aquellos alumnos que sean requeridos 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.

Es requisito indispensable utilizar una base de datos Mongo en la nube mLab.

Instrucciones parte 1 "aplicación Web"

Los casos de usos 1 - 8 son de carácter obligatorio, para que el trabajo pueda ser evaluado esos 8 casos 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 estos 8 casos de forma perfecta es de 10 sobre 10 dentro de la parte 1 "aplicación web"

Casos de uso

1 Público: registrarse como usuario

Los usuarios deben poder registrarse en la aplicación aportando un email, nombre 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.

2 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.

En caso de que el inicio de sesión sea correcto se debe dirigir al usuario a la vista que "lista todos los usuarios de la aplicación"

3 Usuario registrado: listar todos los usuarios de la aplicación

Se visualizará en una lista todos los usuarios de la aplicación. Para cada usuario se mostrará su nombre y email.

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

Debe incluirse una opción de menú principal, visible solo para usuarios en sesión que permita acceder al listado de todos los usuarios de la aplicación.

4 Usuario registrado: buscar entre todos los usuarios de la aplicación

Incluir un sistema que permita realizar una búsqueda por nombre y email de usuario. El cuadro de búsqueda contendrá un único campo de texto, la cadena introducida en ese campo utilizará para buscar coincidencias tanto en el campo nombre 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 o su email.

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

5 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 esta enviando una invitación de amistad a ese usuario.

No se debe poder enviar una invitación a un usuario que ya es amigo o a un usuario al que ya se la ha enviado otra previamente.

6 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 del usuario de la aplicación que solicito la amistad.

Debe incluirse una opción de 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.

7 Usuario registrado: aceptar una invitación recibida

Junto a cada invitación mostrada en la lista de invitaciones de amistad recibidas se debe incluir un botón con el texto "aceptar", al pulsar este botón 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).

8 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 y email.

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

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

Instrucciones 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 interior (no hay que desarrollar una nueva aplicación). Además de los servicios 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 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 6 sobre 10 dentro de la parte 2 "servicios web"

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 10 sobre 10 dentro de la parte 2 "servicios web"

Casos de uso

Servicio Web - Implementación de la API de Servicios Web REST

S.1 Identificarse con usuario – token

El servicio recibe un nombre de usuario y una contraseña, en caso de que exista coincidencia con algún usuario almacenado en la base de datos retornará un token de autenticación. Sí no hay coincidencia retornará un mensaje de error de inicio de sesión no correcto.

S.2 Usuario identificado: listar todos los amigos

El servicio retorna una lista con los identificadores de todos los amigos del usuario identificado.

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 valido.

S.3 Usuario identificado: Crear un mensaje

El servicio debe permitir crear nuevos mensajes.

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

Para permitir crear un nuevo mensaje, el usuario destino debe ser amigo del usuario identificado, por lo tanto, la petición debe contener un token de seguridad valido y el usuario identificado con el token debe ser amigo del usuario destino.

S.4 Usuario identificado: Obtener mis mensajes de una "conversación",

El servicio recibe el identificador de dos usuarios y retorna una lista con todos los mensajes para cuales 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.

* S.5 Usuario identificado: Marcar mensaje como leído

Este servicio permite 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.

Cliente - Aplicación jQuery

C.1. Autenticación del usuario

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

C.2. Mostrar la lista de amigos

Una vez autenticado se debe redirigir al usuario a su lista de amigos (no aplicar ningún sistema de paginación). Dentro de esta lista se debe ofrecer la posibilidad de filtrar los amigos por su nombre (en lugar del nombre se admite también que la lista se filtre únicamente por email). Por ejemplo, con un cuadro para insertar texto situado la parte superior de la lista.

C.3. Mostrar los mensajes

Al pulsar sobre el nombre de un usuario de la lista de amigos se debe 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 (Funcionalidad en caso de uso C.4)

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 cada N segundos.

C.4. 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 destino.

*C.5. 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 automáticamente. 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 de los 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 / o cambios en los estados leído.

*C.6 Mostrar el número de mensajes sin leer

Se debe mostrar junto al nombre de cada amigo (en la lista de amigos) cuantos 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.

*C.7 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 (teniendo en cuenta mensajes tanto enviados como recibidos). Las conversaciones más recientes deben tener prioridad respecto a las menos.

Pruebas automatizadas

Se deberá suministrar un proyecto Java JUnit con un mínimo de pruebas unitarias empleando el framework Selenium (Se suministrará en el campus virtual un proyecto plantilla de ejemplo).

Las clases de equivalencia mínimas (válidas e invalidas) que deberán realizarse serán:

1.1 [RegVal] Registro de Usuario con datos válidos.

1.2 [RegInval] Registro de Usuario con datos inválidos (repetición de contraseña invalida).

2.1 [InVal] Inicio de sesión con datos válidos.

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

3.1 [LisUsrVal] Acceso al listado de usuarios desde un usuario en sesión.

3.2 [LisUsrInVal] Intento de acceso con URL desde un usuario no identificado al listado de usuarios desde un usuario en sesión. Debe producirse un acceso no permitido a vistas privadas.

4.1 [BusUsrVal] Realizar una búsqueda valida en el listado de usuarios desde un usuario en sesión.

4.2 [BusUsrInVal] Intento de acceso con URL a la búsqueda de usuarios desde un usuario no identificado. Debe producirse un acceso no permitido a vistas privadas.

5.1 [InvVal] Enviar una invitación de amistad a un usuario de forma valida.

5.2 [InvInVal] Enviar una invitación de amistad a un usuario al que ya le habíamos invitado 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.

6.1 [LisInvVal] Listar las invitaciones recibidas por un usuario, realizar la comprobación con una lista que al menos tenga una invitación recibida.

7.1 [AcepInvVal] Aceptar una invitación recibida.

8.1 [ListAmiVal] Listar los amigos de un usuario, realizar la comprobación con una lista que al menos tenga un amigo.

C1.1[[CInVal] Inicio de sesión con datos válidos.

C1.2 [CInInVal] Inicio de sesión con datos inválidos (usuario no existente en la aplicación).

C.2.1 [CListAmiVal] Acceder a la lista de amigos de un usuario, que al menos tenga tres amigos.

C.2.2 [CListAmiFil] 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.

C3.1 [CListMenVal] Acceder a la lista de mensajes de un amigo "chat", la lista debe contener al menos tres mensajes.

C4.1 [CCrearMenVal] Acceder a la lista de mensajes de un amigo "chat" y crear un nuevo mensaje, validar que el mensaje aparece en la lista de mensajes.

C5.1 [CMenLeidoVal] 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 recibido 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.

C6.1 [CListaMenNoLeidoVal] 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.

C7.1 [COrdenMenVall] Identificarse con un usuario A que al menos tenga 3 amigos, ir al chat del ultimo 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 esta 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 lisa de amigos.

Para cada una de estas pruebas se debe añadir al menos un caso de prueba y en caso querer añadir más casos se numerará la prueba (por ejemplo, RegVal1, RegVal2, RegVal 3…).

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

Todas las pruebas deben incluir el click en las opciones de menú correspondientes, etc, etc, igual que lo haría un usuario real.

El protocolo de prueba

Para probar cada proyecto el profesor realizará los siguientes pasos:

a) Importar y ejecutar la aplicación Node

b) Importará y ejecutar el proyecto sdi2-test en eclipse.

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 siguiente 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 la aplicación en un controlador, acceder a un repositorio desde un controlador, configurar una vista desde un servicio, etc.).

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.

Entrega

El número n de cada trabajo deberá ser el UO del alumno (en caso de un integrante) o la concatenación de los UO(en caso de dos ej: UO0000UO0001) Según ese número se deberá crear los proyectos eclipse

y las pruebas unitarias con los nombres **sdi1-n** y **sdi1-test-n** respectivamente. Se deberá subir a la tarea correspondiente un archivo ZIP (usando el formato ZIP) con el nombre sdi1-n.zip (en minúsculas) y que deberá contener en su raíz:

- Un documento PDF (con el mismo nombre que el proyecto sdi2-n.pdf) que contenga:
  - Una descripción clara y detallada de la implementación de cada uno de los casos de uso implementados.
  - Un catálogo de las pruebas unitarias realizadas y descripción sencilla de cada una
  - Cualquier otra información necesaria para una descripción razonablemente detallada de lo entregado y su correcto despliegue y ejecución.
- El proyecto Node en formato carpeta (no comprimido) con el nombre .**sdi2-n**
  - Un único proyecto Node debe contener la aplicación web y el cliente jQuery-Ajax
- El proyecto Java eclipse con los casos de prueba en formato carpeta (no comprimido) con el nombre **.\sdi2-test-n**
- En resumen, el zip deberá contener en su raíz:
  - sdi2-n.pdf
  - sdi2-n
  - sdi2-test-n

Fecha máxima de entrega

La fecha de entrega será dependiente del grupo de laboratorio, coincidiendo con 12 días **DESPUES** de a la impartición de la práctica 11 en el el grupo al que pertenece el alumno, la hora límite serán las 20:00.

Por ejemplo, el grupo del L1 del Miércoles, tiene la práctica 11 el 25/04 por lo tanto deberá entregar el: 7/05 a las 20:00

Evaluación

Se penalizará:

- Que se presenten problemas durante el despliegue.