Sistemas Distribuidos e Internet

Curso 2023/2024

PRÁCTICA 2 DE ENTREGA – Node.js y Servicios web

MY SOCIALNETWORK

(Convocatoria ordinaria)

Consideraciones sobre la evaluación de requisitos

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

- Parte 1 "aplicación web" 5.0/11 puntos.
- Parte 2 "servicios web" 5.0/11 puntos.
- Documentación 1.0/11 puntos.

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 (*ver Tabla 1 Detalle de la puntuación de la práctica*) 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 todos los **requisitos obligatorios, más la documentación 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, en evaluación continua.**

Nota: Los trabajos que no desarrollen todos los requisitos obligatorios con sus correspondientes pruebas, serán calificados con un cero (0).

Los **requisitos opcionales (Ver *Tabla 1 Detalle de la puntuación de la práctica*) 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**.

Descripción			Puntos
PARTE1 - APLICACIÓN WEB			**5.00**
	REQUISITOS OBLIGATORIOS (WEB)		**4.00**
	W1	Usuario Anónimo: registrarse como usuario	0.20
	W2	Usuario Anónimo: iniciar sesión	0.10
	W3	Usuario Registrado: Fin de sesión	0.10
	W4	Usuario Administrador: Ver listado de usuarios del sistema	0.20
	W5	Usuario Administrador: Editar usuarios	0.30
	W6	Usuario Administrador: Borrar múltiples usuarios	0.40
	W7	Usuario Estándar: Ver el listado de usuarios de la red social	0.20
	W8	Usuario registrado: Buscar usuarios	0.20
	W9	Usuario Estándar: Enviar una invitación de amistad	0.40
	W10	Usuario Estándar: Listar las invitaciones de amistad recibidas	0.40
	W11	Usuario Estándar: Aceptar una invitación	0.30
	W12	Usuario Estándar: Ver listado de amigos	0.20
	W13	Seguridad	1.00
	REQUISITOS OPCIONALES (WEB)		**1.00**
	W14	Usuario Estándar: Crear una nueva publicación	0.40
	W15	Usuario Estándar: Ver el listado de publicaciones propias	0.30
	W16	Usuario Estándar: Ver perfil de un usuario amigo	0.30
PARTE2 - SERVICIOS WEB			**5.00**
	PARTE2A - REQUISITOS OBLIGATORIOS (API REST)		**1.50**
	S1	Usuario Anónimo: Identificarse como usuario vía token	0.30
	S2	Usuario Estándar: Listar todos los amigos	0.30
	S3	Usuario Estándar: Enviar un mensaje a un amigo	0.30
	S4	Usuario Estándar: Obtener los mensajes de una "conversación"	0.30
	S5	Usuario Estándar: Obtener el listado de conversaciones	0.30
	PARTE2A - REQUISITOS OPCIONALES (API REST)		**1.00**
	S6	Usuario Estándar: Eliminar una conversación	0.50
	S7	Usuario Estándar: Marcar mensaje como leído	0.50
	PARTE2B - REQUISITOS OBLIGATORIOS (CLIENTE REST)		**1.50**
	C1	Usuario Anónimo: Autenticación del usuario	0.30
	C2	Usuario Estándar: Mostrar listado de amigos	0.30
	C3	Usuario Estándar: Enviar un mensaje a un amigo	0.30
	C4	Usuario Estándar: Mostrar los mensajes de una conversación	0.30
	C5	Usuario Estándar: Mostrar el listado de conversaciones	0.30
	PARTE2B - REQUISITOS OPCIONALES (CLIENTE REST)		**1.00**
	C6	Usuario Estándar: Eliminar una conversación	0.30
	C7	Usuario Estándar: Marcar mensajes como leídos de forma automática	0.30
	C8	Usuario Estándar: Mostrar el número de mensajes sin leer	0.40
INFORME OBLIGATORIO			**1.00**
TOTAL			**11.00**
RESUMEN			
REQUISITOS OBLIGATORIOS			**7.00**
REQUISITOS OPCIONALES			**3.00**
INFORME OBLIGATORIO			**1.00**
TOTAL			**11.00**

Tabla 1 Detalle de la puntuación de la práctica

Evaluación continua:

- **Trabajo en equipo:** Esta práctica se deberá realizar en equipo (los equipos formados en los grupos de prácticas) y su entrega es obligatoria.
- **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 Likert 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, en caso de ser necesario.

- **Prueba de autoría:** 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. El listado de alumnos y fecha de dicha prueba se publicará en el CV con suficiente antelación.

Evaluación diferenciada:

- **Trabajo individual:** Los alumnos se hayan acogido a la evaluación diferenciada deberán realizar este trabajo de ***forma individual obligatoriamente*** y su entrega también es obligatoria.
- **Prueba de autoría:** Todos los alumnos que se hayan acogido a la evaluación diferenciada deberán realizar una defensa presencial del trabajo en el laboratorio de prácticas. Esta defensa consistirá 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.

General:

- **Mongo DB:** Es requisito indispensable utilizar una base de datos Mongo en local (versión 7), debido a la latencia que hay Mongo en la nube (Mongo Cloud). Esto último podría ocasionar fallos y errores en las pruebas automatizadas.
- **GitHub:** Es requisito indispensable subir el proyecto a un **repositorio PRIVADO en GitHub** e invitar al usuario **sdigithubuniovi** como colaborador. Sobre dichos repositorios deberán realizarse actualizaciones frecuentes para que se pueda hacer un seguimiento del ritmo de trabajo de todos los alumnos.

Fecha máxima de entrega

La fecha límite de entrega será el domingo ***05/05/2024 a las 23:55.***

Consideraciones importantes:

- ***No se aceptará ningún trabajo fuera de plazo.***
- ***Todos los trabajos deben entregarse por el campus virtual.***
- ***Solo entregará un alumno por equipo.***
- ***No se aceptará ningún trabajo que sea entregado por una vía diferente a la especificada en este documento.***

Parte1 – Aplicación web

El objetivo de la práctica es desarrollar una aplicación web de tipo red social. Existirán perfiles de usuario de tipo: Público (Anónimo), Usuario Registrado (Administrador y Usuario Estándar), usando Node.js y servicios web. En esta parte los ***requisitos del W1 hasta el W13 son obligatorios***. A continuación, se detallan los requisitos:

W1 Usuario Anónimo: Registrarse como usuario

Los usuarios deben poder registrarse en la aplicación aportando email, nombre, apellidos, fecha de nacimiento (DD/MM/AAAA), y una contraseña (que deberá repetirse dos veces y coincidir entre sí).

Consideraciones importantes:

- El email del usuario no podrá estar repetido en el sistema, se debe informar al usuario de los errores en el proceso de registro.
- Es obligatorio realizar las validaciones del lado del servidor.
- Una vez registrado un usuario será autenticado automáticamente redirigiéndole a la vista de login "Iniciar sesión" [Requisito Obligatorio W2].
- La fecha de nacimiento deberá ser una fecha válida.
- La contraseña se deberá guardar cifrada en la base de datos.
- El perfil por defecto cuando un usuario se registre será de "Usuario Estándar".

Pruebas Funcionales

[Prueba1] Registro de Usuario con datos válidos.

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

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

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

W2 Usuario Anónimo: Iniciar sesión

Suministrando su email y contraseña, un usuario registrado 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 redirigir al usuario a la ruta que contenga las opciones del perfil correspondiente.

Caso 1: Usuario con perfil de administrador

- **Para facilitar las pruebas deberá existir un usuario administrador en el sistema con las siguientes credenciales (email:admin@email.com y contraseña: @Dm1n1str@D0r).**
- Un usuario es administrador cuando está asignado a dicho rol en el sistema. Se puede crear el usuario administrador antes citado, desde código o directamente en la base de datos.

- En caso de que el inicio de sesión sea correcto se debe redirigir al usuario a la vista: "Ver listado de usuarios del sistema" [Requisito Obligatorio W4].

Caso 2: Usuario Estándar

- En caso de que el inicio de sesión sea correcto se debe redirigir al usuario a la vista "Ver el listado de usuarios de la red social" [Requisito Obligatorio W7].

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, email existente, pero contraseña incorrecta).

[Prueba8] Inicio de sesión con datos inválidos (campo email o contraseña vacíos).

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". En esta funcionalidad se deberán cumplir los siguientes criterios:

- Este botón solo se mostrará exclusivamente si un usuario se ha autenticado previamente.
- Al cerrar sesión se mostrará un mensaje indicando que el "Ha cerrado sesión correctamente".

Pruebas Funcionales

[Prueba9] Hacer clic en la opción de salir de sesión y comprobar que se muestra el mensaje "Ha cerrado sesión correctamente" y 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: Ver 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. En esta funcionalidad se deberán cumplir los siguientes criterios:

- Para cada usuario se mostrará su email, nombre, apellidos y un enlace/botón "Editar" que la hacer clic en el mismo, mostrará el formulario de editar usuario [Requisito Obligatorio W5].
- La lista debe incluir un sistema de paginación y mostrar al menos 3 páginas y 5 usuarios por página.

Pruebas Funcionales

[Prueba11] Mostrar el listado de usuarios y comprobar que se muestran todos los que existen en el sistema, ***incluyendo el usuario actual y los usuarios administradores***.

W5 Usuario Administrador: Editar usuarios

Se debe incluir una nueva vista con un formulario para que un usuario administrador pueda editar los datos de un usuario existente. Al editar un registro de usuario se deberán cumplir los siguientes criterios:

- Un usuario administrador podrá asignar o cambiar el rol de un usuario registrado en el sistema mediante una lista desplegable de roles, que deberá cargarse dinámicamente en el formulario de editar usuario desde un servicio de gestión de roles.
- Ningún usuario podrá cambiar el perfil o rol de otro usuario, a menos que el usuario autenticado sea administrador. En este caso, la lista desplegable de roles NO deberá aparecer la vista y siempre deberá validarse del lado del servidor.

- Una vez que a un usuario se le cambie el rol, automáticamente tendrá todos los permisos de acceso asignados a dicho rol.
- Se deberán validar todos los datos del registro que se está modificando, siguiendo los criterios de registro de usuarios [Requisito Obligatorio W1].
- Al editar en usuario no se modificará la contraseña del usuario.

Pruebas Funcionales

[Prueba12] Autenticarse como administrador, editar un usuario estándar, cambiando su rol a administrador, email, nombre y apellidos, comprobar que los datos se han actualizados correctamente. Salir de sesión como administrador y autenticarse como el usuario modificado y acceder a la funcionalidad de listado de usuarios del sistema para probar el nuevo rol de administrador.

[Prueba13] Editar un usuario introduciendo datos inválidos (email existente asignado a otro usuario del sistema, nombre y apellidos vacíos), comprobar que se devuelven los mensajes de error correctamente y que el usuario no se actualiza.

W6 Usuario Administrador: Borrar múltiples usuarios.

En la vista del [Requisito Obligatorio W4] donde figuran todos los usuarios de la aplicación 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.

Consideraciones importantes:

- En este listado no debería aparecer el usuario administrador o en su defecto no debería poder borrarse (validación tanto en lado cliente como en lado servidor).

Pruebas Funcionales

[Prueba14] Ir a la lista de usuarios, borrar el primer usuario de la lista, comprobar que la lista se actualiza y dicho usuario desaparece.

[Prueba15] Ir a la lista de usuarios, borrar el último usuario de la lista, comprobar que la lista se actualiza y dicho usuario desaparece.

[Prueba16] Ir a la lista de usuarios, borrar 3 usuarios, comprobar que la lista se actualiza y dichos usuarios desaparecen.

[Prueba17] Intentar borrar el usuario que se encuentra en sesión y comprobar que no ha sido borrado (porque no es un usuario administrador o bien, porque, no se puede borrar a sí mismo, si está autenticado).

W7 Usuario Estándar: Ver listado de usuarios de la red social.

Se visualizará en una lista todos los usuarios de la aplicación (***excepto Administrador y el usuario autenticado***). En esta funcionalidad se deberán cumplir los siguientes criterios:

- Para cada usuario se mostrará su email, nombre y apellidos.
- 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

[Prueba18] 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.

W8 Usuario registrado: Buscar usuarios

Incluir un sistema que permita realizar una búsqueda por email, nombre y apellidos de usuario. El cuadro de búsqueda contendrá un único campo de texto. La cadena introducida en ese campo se utilizará para buscar coincidencias parciales en cualquiera de los campos. Por ejemplo, si escribimos la cadena "mar" deberá devolver aquellos usuarios en los que la cadena "mar" sea parte de su nombre, apellidos o su email. En esta funcionalidad se deberán cumplir los siguientes criterios:

- 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.
- El sistema de paginación debe mantener el criterio de búsqueda y mostrar las páginas de acuerdo con la búsqueda realizada.

Pruebas Funcionales

[Prueba19] Hacer una búsqueda con el campo vacío y comprobar que se muestra la página que corresponde con el listado usuarios existentes en el sistema.

[Prueba20] 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.

[Prueba21] 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.

W9 Usuario Estándar: Enviar invitación de amistad

Junto a la información de cada usuario presente en la vista de "listado de usuarios de la red social" [Requisito Obligatorio W7], debe aparecer un ***botón/enlace*** con el texto "agregar amigo", siempre y **cuando no sean ya amigos**. Al pulsar el botón el usuario en sesión enviará una invitación de amistad a ese usuario. En esta funcionalidad se deberán cumplir los siguientes criterios:

- Se deberá registrar la fecha de envío de la solicitud.
- Si los usuarios ya son amigos, el enlace "agregar amigo" no deberá aparecer en la vista de usuario.
- Si se envía una invitación de amistad a un usuario amigo o hay una solicitud de amistad en curso entre ambos usuarios, el sistema validará del lado del servidor y mostrará un mensaje de error indicando que no se puede realizar dicha solicitud, indicando cuál es la causa.

Pruebas Funcionales

[Prueba22] 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).

[Prueba23] 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.

W10 Usuario Estándar: Listar las invitaciones de amistad recibidas

Se visualizará un listado de todas las invitaciones de amistad que ha recibido el usuario autenticado. En esta funcionalidad se deberán cumplir los siguientes criterios:

- Para cada invitación se mostrará la fecha de la solicitud, el email, el nombre completo del usuario de la aplicación que solicitó la amistad.
- Debe incluirse una opción en el menú principal, visible solo para usuarios autenticados 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

[Prueba24] Mostrar el listado de invitaciones de amistad recibidas. Comprobar con un listado que contenga varias invitaciones recibidas.

W11 Usuario Estándar: Aceptar una invitación

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). En esta funcionalidad se debe se deberán cumplir los siguientes criterios:

- Una vez creada la relación de amistad, no pueden existir peticiones pendientes entre A y B.
- Se deberá registrar la fecha de aceptación de la solicitud de amistad.

Pruebas Funcionales

[Prueba25] 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.

W12 Usuario Estándar: Ver listado de amigos

Se visualizará en una lista todos los usuarios amigos del usuario en sesión. En esta funcionalidad se deberán cumplir los siguientes criterios:

- Para cada usuario se mostrará su email, nombre, apellidos y la fecha de inicio de la relación de amistad.
- Para cada usuario se mostrará su última publicación realizada [Requisito Obligatorio W13].
- Si el usuario no ha realizado ninguna publicación se mostrará un mensaje ***"Este usuario aún no ha realizado ninguna publicación"***
- 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 autenticados que permita acceder al listado de todos los amigos del usuario autenticado.

Pruebas Funcionales

[Prueba26] Mostrar el listado de amigos de un usuario. Comprobar que el listado contiene los amigos que deben ser.

[Prueba27] Mostrar el listado de amigos de un usuario. Comprobar que se incluye la información relacionada con la última publicación de cada usuario y la fecha de inicio de amistad.

W13 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. Además, se evaluará en este apartado que:

- Se emplea la técnica de autentificación/autorización más adecuada al contexto.

- La existencia de comprobación basada en roles para el acceso a funcionalidades dependientes de rol específico.
- Registro de la actividad de los usuarios en un Logger (véase log4js), incluyendo la fecha, la acción y la información que el grupo considere relevante para cada caso.

  Casos:

  - Registro de todas y cada una de las peticiones recibidas en el controlador correspondiente.
  - Registro de las altas de usuarios en el sistema.
  - Registro de los inicios de sesión tanto exitosos como fallidos.
  - Registro de los cierres de sesión.
- Almacenar en base de datos la información de los cuatro tipos de logs indicados en el punto anterior (peticiones, altas, inicio de sesión con éxito y sin éxito y cierres de sesión.). La información mínima que se deberá almacenar en la base de datos será:
  - Para cada petición:
    - Tipo de log: "PET".
    - Fecha-Hora: Fecha y hora en que se registró el log en formato timestamp.
    - Texto descriptivo: Un texto que incluya el mapping del controlador que recibe la petición, el método http, y los parámetros recibidos si los hubiera.
  - Para cada alta:
    - Lo mismo que para cada petición, pero con tipo de log = "ALTA". En este caso el log aparecerá duplicado: uno por la petición y otro por el alta.
  - Para cada inicio de sesión con éxito:
    - Tipo de log: "LOGIN-EX".
    - Fecha-Hora: Fecha y hora en que se realizó el login en formato timestamp.
    - Texto descriptivo: el username del usuario que ha entrado en sesión.
  - Para cada inicio de sesión sin éxito:
    - Tipo de log: "LOGIN-ERR".
    - Fecha-Hora: Fecha y hora en que se realizó el intento de login en formato timestamp.
    - Texto descriptivo: el username del usuario que intentó entrar en sesión.
  - Para cada vez que el usuario sale de sesión:
    - Tipo de log: "LOGOUT".
    - Fecha-Hora: Fecha y hora en que se realizó el logout en formato timestamp.
    - Texto descriptivo: el username del usuario que salió de sesión.

En esta funcionalidad se deberán cumplir los siguientes criterios:

- El usuario administrador podrá visualizar todos los logs ordenados por fecha-hora de más reciente a más antiguo.

- El listado podrá ser filtrado por cada tipo de log indicado arriba y el resultado siempre será ordenado por fecha de más reciente a más antiguo.
- Para filtrar la información se utilizará una lista desplegable de tipos de logs.
- El usuario administrador podrá borrar todos los logs con una opción de menú, teniendo en cuenta el tipo de logs que está seleccionando en la lista desplegable.

Pruebas Funcionales

[Prueba28] Intentar acceder sin estar autenticado a la opción de listado de usuarios. Se deberá volver al formulario de login.

[Prueba29] 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.

[Prueba30] Estando autenticado como usuario estándar intentar acceder a una opción disponible solo para usuarios administradores (Añadir menú de auditoria (visualizar logs)). Se deberá indicar un mensaje de acción prohibida.

[Prueba31] Estando autenticado como usuario administrador visualizar todos los logs generados en una serie de interacciones. Esta prueba deberá generar al menos dos interacciones de cada tipo y comprobar que el listado incluye los logs correspondientes.

[Prueba32] Estando autenticado como usuario administrador, ir a visualización de logs y filtrar por un tipo, pulsar el botón/enlace borrar logs y comprobar que se eliminan los logs del tipo seleccionado, de la base de datos.

W14 OPCIONAL - Usuario Estándar: 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. En esta funcionalidad se deberán cumplir los siguientes criterios:

- Todos los datos son obligatorios y deberá validarse del lado del servidor.
- La fecha de creación de la publicación será la fecha del sistema.
- Debe incluirse una opción de menú principal, visible solo para usuarios autenticados que permita acceder al formulario de crear publicaciones.

Pruebas Funcionales

[Prueba33] 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.

[Prueba34] Ir al formulario de crear publicaciones, rellenarla con datos inválidos (campos título y descripción vacíos) y pulsar el botón Submit. Comprobar que se muestran los mensajes de campo obligatorios.

W15 OPCIONAL - Usuario Estándar: Ver listado de publicaciones propias

Se visualizará en una lista todas las publicaciones realizadas por el usuario autenticado.

En esta funcionalidad se deberán cumplir los siguientes criterios:

- Para cada publicación se mostrará: la fecha en la que se creó, el título y el texto de la publicación.
- Este listado debe incluir un sistema de paginación y mostrar 5 publicaciones por página.
- Debe incluirse una opción de menú principal, visible solo para usuarios autenticados que permita acceder al a la lista personal de publicaciones.

Pruebas Funcionales

[Prueba35] Mostrar el listado de publicaciones de un usuario y comprobar que se muestran todas las que existen para dicho usuario.

W16 OPCIONAL - Usuario Estándar: Ver perfil de un usuario amigo

Se modificará la vista que muestra la lista con los amigos del usuario autenticado para poder acceder a las al perfil de un amigo. Al pulsar sobre el nombre del amigo se redirigirá al perfil de dicho usuario, donde se mostrarán sus datos personales y una lista donde se visualicen todas sus publicaciones.

En esta funcionalidad se deberán cumplir los siguientes criterios:

- Se debe mostrar el perfil del usuario con sus datos personales (email, nombre y apellidos).
- Se deberá incluir un listado de sus publicaciones y para cada publicación se mostrará la fecha en la que se creó, el título y el texto de esta.
- Este listado debe incluir un sistema de paginación y mostrar 5 publicaciones por página**.**

Pruebas Funcionales

[Prueba36] Mostrar el perfil del usuario y comprobar que se muestran sus datos y el listado de sus publicaciones.

[Prueba37] Utilizando un acceso vía URL u otra alternativa, tratar de acceder al perfil de un usuario que no sea amigo del usuario identificado en sesión. Comprobar que el sistema da un error de autorización.

Parte 2 "API Servicios web REST" + Cliente ligero JQuery/AJAX

Dentro de la aplicación de la Parte 1 se deberán implementar un cliente ligero JQuery/AJAX cuyo backend será una suite de servicios web REST. Este nuevo cliente abordará parte de las funcionalidades de la aplicación anterior, y además incorporará algunas nuevas funcionalidades opcionales tales como un sencillo chat, similar al WhatsApp que permitirá enviar y recibir mensajes de otros usuarios amigos. Conceptualmente las funcionalidades, tanto obligatorias como opcionales, en esta parte de la práctica se organizarán de la siguiente manera. Una vez identificado, un usuario dispondrá de las siguientes opciones:

- Ver listado de amigos del usuario autenticado. Desde este listado para cada amigo aparecerá un botón o enlace para **iniciar una conversación**. Además, se deberá validar en el servidor que un usuario no pueda enviarse un mensaje a sí mismo.
- Ver listado de conversaciones ya iniciadas. Esta opción mostrará las conversaciones en las que el usuario participa. Para cada conversación se podrá:
    - Enviar mensajes a dicha conversación.
    - Borrar la conversación completa.

Parte 2A – API de Servicios Web REST

S1 Usuario Anónimo: Identificarse como usuario vía 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. Sí no hay coincidencia retornará un mensaje de error de inicio de sesión no correcto.

Pruebas Funcionales

[Prueba38] Inicio de sesión con datos válidos.

[Prueba39] Inicio de sesión con datos inválidos (email existente, pero contraseña incorrecta).

[Prueba40] Inicio de sesión con datos inválidos (campo email o contraseña vacíos).

Comprobar en las tres pruebas el estado HTTP retornado y la estructura JSON retornada por el servicio REST de autenticación. El token no es necesario comprobarlo ya que no es viable. Emplear la API RestAssured u otra similar (ver ejemplos en el proyecto plantilla compartido).

S2 Usuario Estándar: Listar todos los amigos

Realizar un servicio REST que retorne una lista con **exclusivamente los identificadores, email, 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

[Prueba41] Mostrar el listado de amigos para dicho usuario y comprobar que se muestran los amigos del usuario autenticado. Esta prueba implica invocar a dos servicios: S1 y S2.

S3 Usuario Estándar: Enviar un mensaje a un amigo

Crear un servicio REST que permita que un usuario pueda enviar mensajes (tipo chat) a una conversación entre dos amigos. En esta funcionalidad se deberán cumplir los siguientes criterios:

- Para permitir enviar un nuevo mensaje, el usuario tiene que estar identificado, por lo tanto, la petición debe contener un token de seguridad válido.
- Para permitir enviar un nuevo mensaje, el usuario destinatario debe ser amigo del usuario identificado.
- Un usuario identificado no puede enviarse un mensaje así mismo.
- Se abrirá una conversación por cada par de usuarios amigos.
- La primera vez que un usuario envíe un mensaje a un amigo se creará dicha conversación y, a partir de ahí, los mensajes enviados por ambos sentidos y se irán incorporando a la conversación.
- Para cada mensaje se mostrará y almacenará: el autor del mensaje, la fecha/hora del mensaje, el texto, y si el mensaje ha sido leído (true/false).
- Por defecto el mensaje se crea como no leído.
- No será posible enviar un mensaje vacío a una conversación.
- Los únicos usuarios que pueden participar en una conversación son el usuario que inició la conversación y el usuario destinatario.

Pruebas Funcionales

[Prueba42] Enviar un mensaje a un amigo. Esta prueba consistirá en comprobar que el servicio almacena correctamente el mensaje para el mensaje enviado. Por lo tanto, el usuario tendrá que identificarse (S1), enviar un mensaje para un amigo (S3) y comprobar que el mensaje ha quedado registrado (S4).

S4 Usuario Estándar: Obtener los mensajes de una "conversación"

Crear un servicio REST que dado una conversación entre dos usuarios se retorna una lista con todos los mensajes de esta. 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

[Prueba43] Obtener los mensajes de una conversación. Esta prueba consistirá en comprobar que el servicio retorna el número correcto de mensajes para una conversación. El ID de la conversación deberá conocerse a priori. Por lo tanto, se tendrá primero que invocar al servicio de identificación (S1), y solicitar el listado de mensajes de una conversación de ID conocido a continuación (S4), comprobando que se retornan los mensajes adecuados.

S5 Usuario Estándar: Obtener el listado de conversaciones

Crear un servicio REST que dado un usuario identificado retorne todas las conversaciones vinculadas a dicho usuario:

- Por un lado, aquellas conversaciones en las que el usuario aparezca como emisor.
- Y por otro lado, aquellas conversaciones donde el usuario aparezca como receptor.

En esta funcionalidad se deberán cumplir los siguientes criterios:

- Para permitir obtener los mensajes de una conversación la petición debe contener un token de seguridad válido.

Pruebas Funcionales

[Prueba44] Obtener la lista de conversaciones de un usuario. Esta prueba consistirá en comprobar que el servicio retorna el número correcto de conversaciones para dicho usuario. Por lo tanto, se tendrá primero que invocar al servicio de identificación (S1), y solicitar el listado de conversaciones a continuación (S5) comprobando que se retornan las conversaciones adecuadas.

S6 OPCIONAL - Usuario Estándar: Eliminar una conversación

Crear un servicio REST que permita eliminar una conversación. El servicio recibe el identificador de la conversación. En esta funcionalidad se deberán cumplir los siguientes criterios:

- Al eliminar una conversación se deberán eliminar todos los mensajes relacionados.
- Para eliminar una conversación, el usuario identificado debe ser el *emisor o receptor de los mensajes de dicha conversación.*
- La petición debe contener un token de seguridad válido.

Pruebas Funcionales

[Prueba45] Eliminar una conversación de ID conocido. Esta prueba consistirá en comprobar que se elimina correctamente una conversación concreta. Por lo tanto, se tendrá primero que invocar al servicio de identificación (S1), eliminar la conversación ID (S6) y solicitar el listado de conversaciones a continuación (S5), comprobando que se retornan las conversaciones adecuadas.

S7 OPCIONAL - Usuario Estándar: Marcar mensaje como leído

Crear un servicio REST que permita marcar un mensaje como leído, la propiedad leída debe tomar el valor true. El servicio recibe el identificador del mensaje. En esta funcionalidad se deberán cumplir los siguientes criterios:

- Para permitir cambiar el estado de un mensaje, el usuario identificado puede ser tanto el emisor o el receptor
- La petición debe contener un token de seguridad válido.

Pruebas Funcionales

[Prueba46] Marcar como leído un mensaje de ID conocido. Esta prueba consistirá en comprobar que el mensaje marcado de ID conocido queda marcado correctamente a true como leído. Por lo tanto, se tendrá primero que invocar al servicio de identificación (S1), solicitar el servicio de marcado (S7), comprobando que el mensaje marcado ha quedado marcado a true como leído (S4).

Parte 2B – Cliente ligero JQuery/Ajax

En esta parte hay que a desarrollar una aplicación web jQuery-AJAX que consuma los servicios web REST desarrollados en el punto anterior. Esta aplicación debe permitir el intercambio de mensajes en tiempo real.

C1 Usuario Anónimo: Autenticación del usuario

Haciendo uso de la API REST, la aplicación web 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

[Prueba47] Inicio de sesión con datos válidos.

[Prueba48] Inicio de sesión con datos inválidos (email existente, pero contraseña incorrecta).

[Prueba49] Inicio de sesión con datos inválidos (campo email o contraseña vacíos).

C2 Usuario Estándar: Mostrar listado de amigos

Haciendo uso de la API REST, la aplicación web deberá disponer de una opción que le muestre en una vista HTML, el listado de amigos del usuario autenticado, mostrando para cada amigo los datos: email, nombre y apellidos [Requisito Obligatorio S2].

En esta funcionalidad se deberán cumplir los siguientes criterios:

- Para mostrar el listado de amigos, el usuario tiene que estar identificado, por lo tanto, la petición debe contener un token de seguridad válido.
- No es necesario incluir sistema de paginación en este listado.

Pruebas Funcionales

[Prueba50] Mostrar el listado de amigos para dicho usuario y comprobar que se muestran los amigos del usuario autenticado.

C3 Usuario Estándar: Enviar un mensaje a un amigo

Haciendo uso de la API REST y tomando como punto de partida la vista que muestra el listado de amigos [Requisito Obligatorio C2], el usuario identificado debe poder enviar un nuevo mensaje para ese amigo.

En esta funcionalidad se deberán cumplir los siguientes criterios:

- Al lado de cada amigo añadir un botón o enlace "Conversación" que permita enviar mensajes al amigo correspondiente. Una vez se pulse el botón o enlace se deberá abrir una nueva vista donde se muestre un "chat".
- Las conversaciones son únicas y solo debe existir una conversación por cada par de amigos.
- Una propuesta de interfaz sería un pequeño formulario donde escribir el mensaje y una tabla donde se muestran los mensajes de uno y otro (similar a WhatsApp).
- En este "chat" se visualizarán los mensajes de la conversación. Esta vista deberá tener al menos:
  i) La lista con todos los mensajes correspondientes al emisor y el receptor.
  ii) Formulario para enviar un nuevo mensaje a esta conversación.
- La lista de mensajes de una conversación que está viendo un usuario, 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 y los incorpore a la lista.

- Deberá tenerse en cuenta las validaciones realizadas en el servidor [Requisito Obligatorio S3):
    - El usuario autenticado no podrá iniciar una conversación consigo mismo.
    - No será posible enviar un mensaje vacío para iniciar una conversación.
    - Los únicos usuarios que pueden participar en una conversación son: el usuario que inició la conversación y el receptor.

Pruebas Funcionales

[Prueba51] Sobre listado de amigos (a elección de desarrollador), enviar un mensaje a un amigo concreto. Se abriría dicha conversación por primera vez. Comprobar que el mensaje aparece en el listado de mensajes.

[Prueba52] Sobre el listado de conversaciones enviar un mensaje a una conversación ya abierta. Comprobar que el mensaje aparece en el listado de mensajes.

C4 Usuario Estándar: Mostrar los mensajes de una conversación

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

[Prueba53] Acceder a la lista de mensajes de un amigo, la lista debe contener al menos tres mensajes.

C5 Usuario Estándar: Mostrar el listado de conversaciones

Haciendo uso de la API REST un usuario podrá ver un listado de las conversaciones abiertas y podrá reanudar cada una de las conversaciones enviando un nuevo mensaje y viendo los anteriores. El listado mostrará el email, nombre y apellidos del amigo, además de un botón/enlace para reanudar la conversación.

Pruebas Funcionales

[Prueba54] Mostrar el listado de conversaciones ya abiertas. Comprobar que el listado contiene la cantidad correcta de conversaciones.

C6 OPCIONAL - Usuario Estándar: Eliminar una conversación

Haciendo uso de la API REST y sobre el listado anterior se podrá borrar una conversación incluyendo un botón/enlace "Eliminar" para cada conversación. Al pinchar sobre el botón/enlace se eliminará dicha conversación y todos los datos relacionados [Requisito opcional S6].

Pruebas Funcionales

[Prueba55] Sobre el listado de conversaciones ya abiertas. Pinchar el enlace Eliminar en la primera y comprobar que el listado se actualiza correctamente.

[Prueba56] Sobre el listado de conversaciones ya abiertas. Pinchar el enlace Eliminar en la última y comprobar que el listado se actualiza correctamente.

C7 OPCIONAL - Usuario Estándar: Marcar mensajes como leídos de forma automática

Haciendo uso de la API REST y al entrar en un chat (conversación) 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) [Requisito opcional S7].

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

[Prueba57] 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.

C8 OPCIONAL - Usuario Estándar: Mostrar el número de mensajes sin leer

Haciendo uso de la API REST se debe mostrar junto a cada conversación 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.

Pruebas Funcionales

[Prueba58] 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 amigo y validar que el número de mensajes sin leer aparece.

Informe de entrega y plantilla 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 (sdi2324-entrega2-n.docx). (Entregar en formato PDF).
- Una plantilla Excel para los casos de prueba sdi2324-entrega2-n.xlsx). (Entregar en formato Excel).

Ambos documentos deberán ser renombrados cambiando la n por el código del grupo asignado al equipo (ej. sdi2324-entrega2-22.pdf, sdi2324-entrega2-22.xlsx) o en su defecto por el IDGIT del alumno que entrega de forma individual (evaluación diferenciada).

El contenido del informe deberá ser el siguiente:

- En la portada se deberá incluir:
  - Los datos personales de los autores: Nombre y apellidos, IDGIT, Emails de UNIOVI, Código ID GIT de cada uno y código del equipo.
- Diagrama de navegabilidad con texto explicativo. Un diagrama de navegabilidad no se compone de capturas de pantalla de la interfaz. Pueden usarse estas como apoyo en el texto explicativo, pero nunca como sustitutivo.
- Una descripción clara y detallada aquellos aspectos técnicos y/o diseño que considere relevantes. No se trata de describir qué es Node.js o Servicios web. Se trata de detallar la toma de decisiones a la hora de implementar la práctica en los aspectos técnicos y/o de diseño relevantes.
- Cualquier otra información necesaria para una descripción razonablemente detallada de lo entregado y su correcto despliegue y ejecución. **Obligatorio usar las versiones de los softwares usados en clase.**
- Una tabla que indique que funcionalidades desarrolladas por cada uno de los integrantes del equipo (a grandes rasgos esto debería coincidir con el repositorio de GitHub). Por cada miembro del equipo, se deberá incluir un pantallazo de los commits de GitHub (filtrado por cada alumno).
- Conclusión individual de cada miembro del grupo. Esto incluye la aportación en el desarrollo de la práctica, así como las dificultades encontradas durante el mismo. También que ventajas y desventajas se han detectado a nivel técnico y a nivel de trabajo en grupo

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 primera hoja.

Pruebas automatizadas

Se deberá suministrar un proyecto Java **JUnit** con un mínimo de pruebas unitarias empleando el framework Selenium. **Se suministrará un proyecto plantilla de ejemplo**.

Las clases de equivalencia mínimas (válidas e inválidas) que deberán realizarse se encuentran al final de cada requisito.

El grupo puede añadir más pruebas además de las planteadas en el enunciado, pero deberán incluirla en los ficheros correspondientes y en la documentación.

El desarrollo de las pruebas estará basado en el framework Selenium y JUnit5 vistos durante la asignatura.

Las pruebas realizadas en la primera actividad podrán utilizarse durante el desarrollo de esta actividad.

Aspectos transversales en la evaluación

La calificación máxima que se puede obtener será de 11 puntos, en base al siguiente reparto: 8.0 puntos pertenecen a la funcionalidad obligatoria y la documentación, y 3.0 puntos a la parte opcional

Si bien existen aspectos que se evaluarán de forma transversal a todos los requisitos establecidos en el presente enunciado. **Como normal general se considera que, si no se cumplen, penalizan a la nota obtenida en la práctica.**

Arquitectura

La aplicación deberá estar obligatoriamente diseñada siguiente el patrón arquitectónico visto en clase. Usando siempre de forma correcta controladores, modelos y vistas. La utilización incorrecta de elementos de esta arquitectura ***será fuertemente penalizada***.

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

A continuación, se enumeran otra serie de buenas prácticas (o requisitos) que penalizarán a la nota total del grupo en caso de incumplimiento:

- Claridad y calidad de la implementación del código JavaScript.
- Calidad de la implementación de las vistas y usando todas las funcionalidades vistas en clase.
- Validaciones en los formularios y los distintos endpoints de los servicios web.
- Uso de un formato de URLs consistente entre las diferentes rutas de los controladores.
- La NO documentación correcta del código, siguiendo los estándares.
- Nombrado de ficheros de entrega y formatos correctos.
- Que la compilación del código genere "*warnings*".
- Que se presenten problemas durante el despliegue.

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: @Dm1n1str@D0r
- 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: Us3r@1-PASSW, Us3r@2-PASSW, ….
- 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, debe pensarse en cómo poblar adecuadamente la base de datos para que cada prueba parta 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 equipo, se deberán crear los proyectos Node.js y las pruebas unitarias (Selenium) con los nombres **sdi2324-entrega2-n** y **sdi2324-entrega2-test-n** respectivamente.

Según lo anterior la entrega consistirá en los siguientes tres puntos:

1) Subir el proyecto a un **repositorio PRIVADO en GitHub** con nombre **sdi2324-entrega2-n**. E invitar al usuario **sdigithubuniovi** como colaborador. Sobre dichos repositorios deberán realizarse actualizaciones frecuentes para que se pueda hacer un seguimiento del ritmo de trabajo de todos los alumnos.
2) Subir a la tarea del Campus Virtual correspondiente un archivo ZIP (usando el formato ZIP) con el nombre **sdi2324-entrega2-n.zip** (en minúsculas) y que deberá contener en su raíz:
   - El INFORME OBLIGATORIO en formato PDF con nombre **sdi2324-entrega2-n.pdf y el archivo Excel con el catálogo de casos de prueba (sdi2324-entrega2-n.xlsx).**
   - El proyecto Node.js en formato carpeta (no comprimido) con el nombre **sdi2324-entrega2-n** (debe ser una copia literal de la carpeta del proyecto y **NO** de lo subido a Github).
   - El proyecto Java eclipse con los casos de prueba en formato carpeta (no comprimido) con el nombre **.\sdi2324-entrega2-test-n** (debe ser una copia literal de la carpeta del proyecto y **NO** de lo subido a Github).
   - En resumen, el zip deberá contener en su raíz:
     - **sdi2324-entrega2**-n.pdf (Archivo PDF suministrado y renombrado cambiado la n).
     - **sdi2324-entrega2**-n.xlsx (Archivo Excel suministrado y renombrado cambiado la n).
     - **sdi2324-entrega2**-n (Carpeta proyecto Node.js renombrada cambiado la n)
     - **sdi2324-entrega2-test**-n (Carpeta proyecto Java Intellij renombrada cambiado la n)

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 IDEA.

Nota importante: Es requisito indispensable utilizar una base de datos Mongo en local y que se generen automáticamente todos los datos necesarios para ejecutar las pruebas.