SDI – Sistemas Distribuidos e Internet

ENUNCIADO PRÁCTICA 2 – NODE.JS Y SERVICIOS WEB

INFORME

Nombre/Apellidos, ID-GIT#1	Miguel Cuesta Martínez – IDGIT 2122-1002 – UO258220@uniovi.es
Nombre/Apellidos, ID-GIT#2	Pablo Díaz Rubio – IDGIT 2122-1003 – UO271245@uniovi.es
Nombre/Apellidos, ID-GIT#3	Pablo Rodríguez Rodríguez – IDGIT 2122-1011 – UO271246@uniovi.es
Nombre/Apellidos, ID-GIT#4	Miguel Díaz-Pache Alonso - IDGIT 2122-1004 - UO270628@uniovi.es
Nombre/Apellidos, ID-GIT#5	
Nombre/Apellidos, ID-GIT#6	
Cód. ID EQUIPO	101
Repositorio Github	https://github.com/UO271245/sdi-entrega2-101

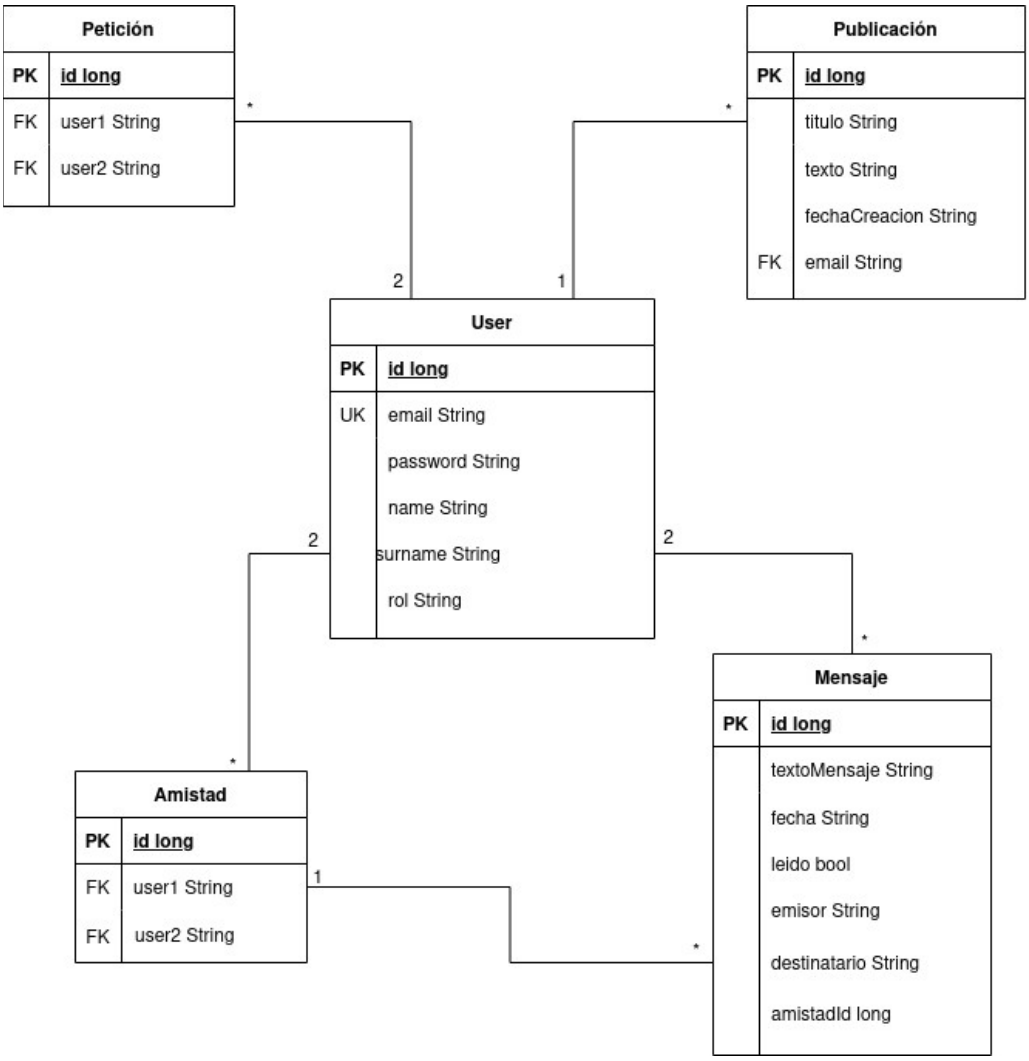
Índice

Introducción

SocialNetwork es una aplicación de interacción social basada en un sistema de publicaciones. Para emplear la aplicación, deberemos registrarnos o autenticarnos como usuario, teniendo uno de dos roles: USER (usuario corriente) o ADMIN (administrador de la aplicación).

Un usuario corriente podrá, en primer lugar, añadir a su perfil nuevas publicaciones y acceder a ellas. También podrá hacerse amigo de otros usuarios mediante un sistema de peticiones de amistad, en el que tendrá la posibilidad tanto de comenzar una nueva petición a otro usuario como de aceptar la que le haya sido enviada. Cuando una relación de amistad se ha formado entre dos usuarios, estos obtendrán acceso al listado de publicaciones del otro.

Un usuario administrador (rol que generalmente será único) poseerá capacidades relacionadas con la gestión de usuarios. Podrá obtener un listado de los usuarios existentes en la aplicación, así como borrarlos.



Además de la aplicación web, proveeremos una API que incluya tanto servicios web REST como la aplicación AJAX necesaria para consumirlos. Permitiremos el login de usuarios, y el listado de amistades una vez en sesión. Adicionalmente, incluimos aquí un sistema de mensajería en el que el usuario logueado podrá crear mensajes a otro usuario, listar su conversación total (conjunto de mensajes) con él, y marcar como leído un mensaje del que es el destinatario.

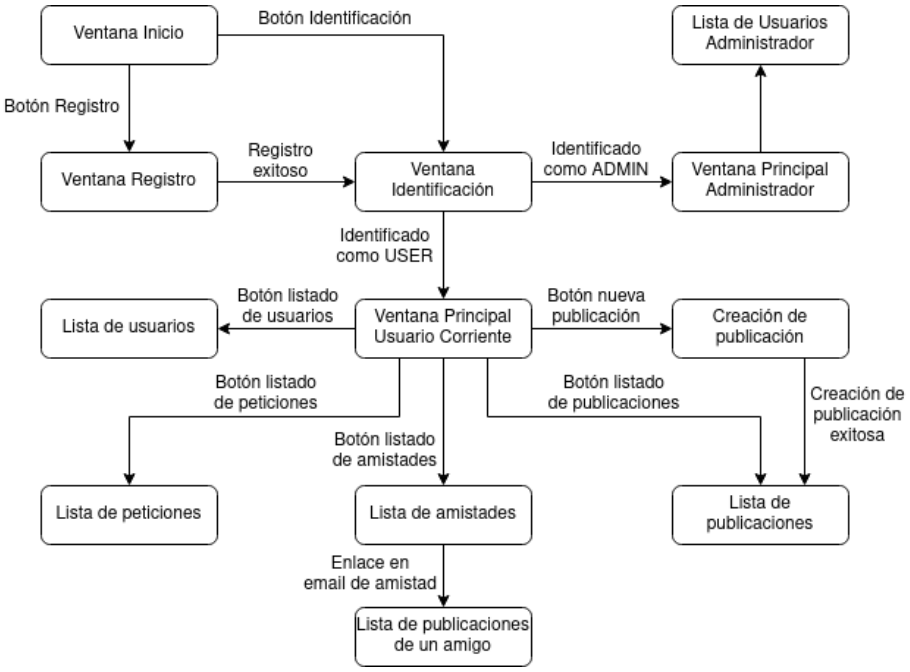
Mapa de navegación

Nuestra aplicación esta compuesta de dos elementos que actúan de forma independiente: la aplicación web y los servicios REST empleados por Jquery.

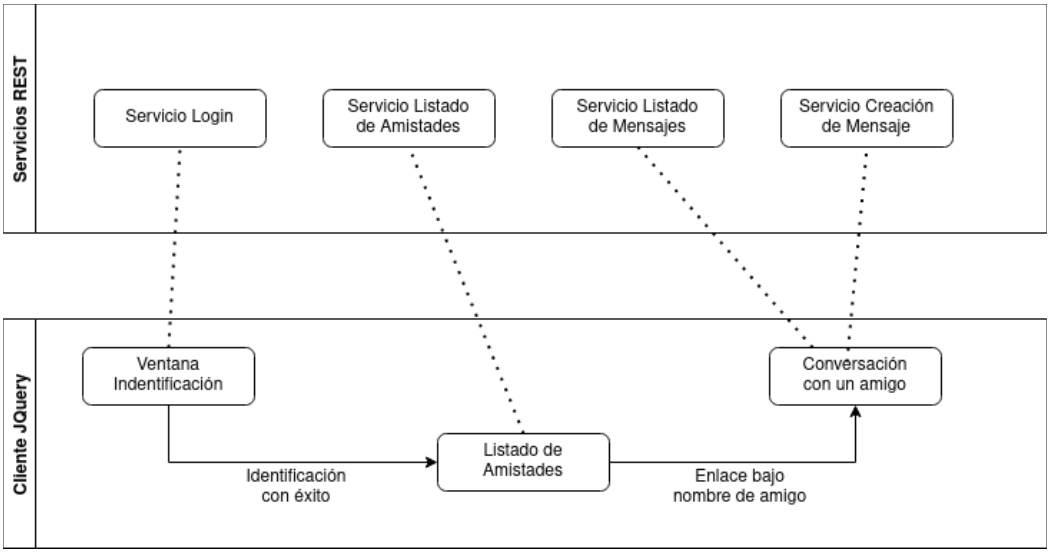
En la aplicación web, al abrir encontraremos una página de inicio desde la que será posible tanto identificarse como un usuario ya existente o registrarse como uno nuevo. Un registro exitoso nos enviará a la ventana de autenticación para comprobar que recordamos los datos introducidos.

Una vez identificados, accedemos a la ventana principal de la aplicación. Si nuestro rol es usuario corriente, tenemos acceso a las 5 funcionalidades principales de la aplicación: listado de usuarios, listado de amistades, listado de peticiones, listado de publicaciones y formulario de creación de nueva publicación. En el listado de usuarios, es posible enviar una petición de amistad a un usuario que no fuese nuestro amigo. Desde el listado de amistades, podremos a acceder a la lista de publicaciones de un usuario amigo. La creación exitosa de una nueva publicación nos devuelve al listado de publicaciones para comprobar su cambio de estado.

Si nuestro rol al identificarnos es el de usuario administrados, tendremos acceso únicamente a la lista de usuarios de toda la aplicación, desde el cual también se puede efectuar el borrado múltiple de estos.



En la parte de servicios web, existen dos elementos a distinguir. Por una parte, proveemos servicios REST que permiten la identificación del usuario; y una vez identificado recuperar su listado de amigos, mostrar sus mensajes con un amigo y crear un nuevo mensaje. Por el otro lado, una aplicación Jquery consumirá estos servicios en una aplicación lineal. Primero, el usuario deberá identificarse. Tras un logueo exitoso, entrará en su lista de amigos. Clicar en el enlace bajo el nombre de un amigo lo enviará a su conversación (listado de mensajes). Por último, desde la vista de listado de mensajes se permitirá la creación de nuevos mensajes.



Aspectos técnicos y de diseño relevantes

Como aspectos técnicos relevantes, mencionaremos el empleo de Maven para la correcta configuración del proyecto de testing. Debido al uso de una base de datos no relacional (MongoDB), la consistencia de esquema dentro de una misma colección ha dejado de ser necesaria. Aún así, hemos conseguido que todas las entidades en cada colección respeten el mismo diseño. Aparece con respecto a la primera práctica de Spring una nueva entidad, Mensaje, la cual está fuertemente vinculada tanto por Emisor y Receptor (emails de usuario) como Id de amistad entre ellos. Estas restricciones ayudaron a reforzar la seguridad del sistema.

Se consideró temporalmente la creación de una entidad conversación para almacenar los múltiples mensajes entre dos usuarios, pero al darse cuenta de que solamente podrían tener una conversación si tenían una relación de amistad, rechazamos la idea en favor del modelo actual.

Asimismo se consideró la reutilización de las plantillas en varios casos, como el listado de publicaciones para el amigo, pero finalmente se mantuvieron ficheros TWIG diferentes para no complicar el diseño con demasiadas condicionales.

Un problema adicional que apareció durante la realización del proyecto fueron las limitaciones del cluster de MongoDB. A pesar de que la nuestra es una aplicación de baja

carga, las peticiones casi consecutivas empleadas en el proyecto de testing superaban el límite establecido para nuestro perfil (gratis) de mongo. Esto nos ha hecho perder mucho tiempo en la recta final del desarrollo, optando finalmente por cerrar las conexiones tras cada test con el fin de reducir la carga. Durante los últimos días de desarrollo, quedó patente lo inconstente que puede llegar a resultar una aplicación con una dependencia al vendedor.

A la hora de testear la parte de servicios web, encontramos varios problemas al ejecutarlos, a pesar de que funcionasen correctamente al margen de Selenium. Finalmente, nos dimos cuenta de que para las pruebas deberíamos acceder a direcciones http y no https, con lo que pasaron a tener el comportamiento esperado.

Información necesaria para el despliegue y ejecución

Ejecución de la aplicación:

- Para acceder a la base de datos y así poder observar su correcta evolución, debemos autenticarnos en MongoDB **con cuenta de Gmail**, por lo que antes de entrar a la web de mongo debemos entrar en la cuenta de Gmail con las siguientes credenciales:
  - Usuario: sdi212210@gmail.com
  - Contraseña: Pa$$word123
- Después de esto, podemos ejecutar la aplicación y acceder a ella en el puerto 4000.
- Para acceder a los servicios web, debemos ir a localhost:4000/api/client.html

Login como administrador:

- Usuario: admin@email.com
- Contraseña: admin

Login como usuario corriente:

- Usuario: userXX@email.com
- Contraseña: userXX

(donde XX correponde a un número de 01 a 15)

Para la ejecución de pruebas, es necesario configurar el proyecto como Maven, arrancar la aplicación, y después ejecutar los test. Las rutas de Firefox y Gecko pueden configurarse al inicio del mismo archivo que contiene los casos de prueba.

Conclusión

Mi contribución consiste en el sistema de publicaciones (W12, W13 y W14), además de los servicios REST de listado de amigos (S2), y añadido y listado de mensajes (S3 y S4). Me encargué además de diseñar los test correspondientes a estas partes. Además de esto, fui el encargado de redactar la documentación y elaborar los diagramas a añadir.

Mi experiencia de trabajo en equipo ha sido positiva. Con respecto al primer entregable, he podido aprovechar más mi tiempo y planificar de antemano mis tareas. Reuniones con otros ayudaron a resolver los problemas encontrados con bastante flexibilidad.

En retrospectiva, dividir las tareas de servidor y cliente de una misma funcionalidad en servicios web no fue la mejor idea. Varias veces encontramos que resultaba más fácil ocuparse del “full stack” de una funcionalidad que combinar dos mitades de esta. Adicionalmente, los problemas aparecidos en el testeo fueron bastante preocupantes, por lo que podría afirmar sin duda que servicios web es la sección que más problemas nos supuso.

- Miguel Cuesta Martínez

La realización del trabajo me ha ayudado a comprender y afianzar mejor los conceptos descritos en las clases de teoría y prácticas de la asignatura, igual que con el primer trabajo. Considero también que el uso de las siguientes tecnologías me ha resultado muy interesante, por lo que voy a comentar el porqué:

La utilización de Node.js y de Express me ha resultado una experiencia muy interesante, y considero que el uso de estas tecnologías es muy cómodo para realizar aplicaciones web, mucho mejor que construir html y scripts para después juntarlos, pudiendo introducir variables en cualquier parte de una vista entre otras herramientas que proporcionan.

El uso de la herramienta session de Express me ha ayudado mucho a cambiar la apariencia de la aplicación en las distintas situaciones, además de facilitar el desarrollo de la aplicación en sí. Gracias al uso de esta simple pero poderosa extensión, pude hacer uso de Templates Javascript que me permiten enviar información que luego va a ser usada mediante bloques de programación en el propio código html. Al tratarse de la fase de desarrollo, poder contar con una herramienta como Nodemon, que arranca un servidor y lo actualiza automáticamente ha sido muy cómodo, ya que nos ha ahorrado mucho tiempo.

Para tanto la realización de las pruebas unitarias como para el propio trabajo de la aplicación en node, he hecho un uso extensivo de diferentes métodos de comunicación con la base de datos no relacional MongoDB. Esto ha afianzado los conocimientos que tenía en un principio de como se hace una conexión entre node y mongo, al darle uso a los iterables que proporciona la librería de mongo, recuperar diferentes tipos de colecciones para luego trabajar con ellos, y por último pero no por ello menos importante emplear Documents para el intercambio de información entre la parte de la aplicación con MongoDB. Esto último me ha proporcionado una forma muy intuitiva para realizar el intercambio de información para las pruebas unitarias, ya que mediante las instrucciónes insertMany, deleteMany, insertOne, deleteOne he podido crear una base de usuarios para la realización de las pruebas, los cuales se borran una vez finalizadas. Esto nos deja la base de datos en un estado

consistente por muchas pruebas unitarias que hiciéramos.

Contribuciones: (Creacion de la base de la aplicacion, Creacion de InitDB y AES para meter datos en mongo desde java en los test, Logger,)(Login y SignUp: twig,js,repository: (W1, W2, W3)),(Login del usuario con cookie y token en servicio API (S1)) (Cliente de la aplicación JQuery: js, widget y Tests: (C1 ,C2 ,C3 , C4)), MensajesRepository.

- Pablo Díaz Rubio

La contribución en este segundo trabajo grupal ha consistido, en su gran mayoría, en la programación relativa a el sistema de peticiones y al sistema de amistades (W8, W9, W10, W11). Aparte de esto, he realizado tareas relacionadas con el listado de usuarios (W6). He creado también varias Bases de Datos en MongoDB. El trabajo me ha resultado bastante satisfactorio al acabarlo, pero sí es verdad que en un principio se me hizo más cuesta arriba que el anterior (quizás porque no se me daba tan bien, tener más trabajos y exámenes...). Entiendo el por qué Node se utiliza mucho hoy en día ya que las aplicaciones se hacen mucho más rápido que, por ejemplo, en SpringBoot; y son más sencillas de ampliar, aunque a veces nuestro conocimiento de JavaScript se ponga por el medio. Me siento satisfecho con mi trabajo y en general también con el de mis compañeros también. Lo peor que me llevo es el último día de trabajo, ya que al ejecutar todos los tests de seguido no funcionaban. Tras mucho pensar en cuál era el problema nos dimos cuenta que no era nuestro trabajo, sino el Cluster el cual nos decía que nos estábamos acercando mucho al límite de conexiones, lo cual no supimos como arreglar de manera sencilla.

- Pablo Rodríguez Rodríguez