Posibles enfoques y aclaraciones práctica 2

Algunos de los casos de uso planteados no están totalmente resueltos en las practicas guiadas, requieren un poco de investigación por vuestra parte y consultar el material visto en clase de teoría. No vamos a poner ejemplos completos de estos casos, pero dado que varios de vosotros habéis preguntado dudas recurrentemente sobre algunos casos vamos a plantear posibles enfoques que podríais seguir en ellos, aunque debe quedar claro que no hay una única solución.

Pruebas: Iniciar base de datos

Lo mejor para poder partir siempre de un esquema de datos limpios es que borréis los datos de la base de datos antes de iniciar una batería de pruebas, una opción muy sencilla (aunque no es la única) sería incluir una URL /borrarTodo que eliminara todas las colecciones que estáis manejando, llamando desde Selenium a esta URL en primer lugar nos aseguramos de partir de una base de datos vacía.

Pruebas: Tiempos de espera

Tened en cuenta que las consultas con la base de datos mLab son más lentas que en la práctica anterior, probablemente tendréis que aumentar los timeouts de las pruebas.

Servicios: esquema RESTful.

Recordar que tal y como se especifica en el guión se debe utilizar una arquitectura RESTful, para los URL que nos dan acceso a los recursos, en caso de no utilizar la arquitectura de forma adecuada habrá una penalización.

En primer lugar debéis *identificar los recursos* con los que trabaja la API, muy probablemente *mensajes* y *usuarios,* no podemos "inventar" recursos que no existan realmente en la aplicación como "amigo" que realmente es un "usuario". Para cada recurso hay que determinar las operaciones posibles y utilizar los métodos HTTP adecuados. Por ejemplo, para obtener todos los mensajes se utiliza **GET /mensaje** o **GET /mensajes** (admitimos el uso de singular o plural). Para obtener un mensaje concreto se puede incluir su identificador como path param: **GET /mensaje/:id** , el identificador de un elemento es el único dato que puede ser enviado como path param, si quisiéramos hacer filtrados o selección de mensajes estos criterios deberían ser enviados como parámetros get (query params), por ejemplo **GET /mensaje?leidos=false** podría retornar todos los mensajes sin leer (este ejemplo solo incluye un parámetro pero podría admitir más).

Para crear un nuevo mensaje usaríamos **POST / mensaje**, enviando todos los datos en el cuerpo de la petición, para modificar un mensaje existente usaríamos **PUT** o **UPDATE /mensaje/:id** enviando los datos que queremos modificar del mensaje seleccionado en el cuerpo de la petición.

*** S.5 Usuario identificado: Marcar mensaje como leído**

El que marca un mensaje como leído es el "receptor" del mensaje la propiedad es "leído por el receptor", no tiene sentido que el emisor marque un mensaje como leído, lo que si tiene sentido es que en el cliente, vea si el receptor de los mensajes los ha leído o no.

C.2 Mostrar lista de amigos

En el documento dice que se puede realizar una búsqueda por nombre de usuario, pero podéis realizar en su lugar una búsqueda por **email**. La idea es simplificar la obtención de nuestros amigos, simplemente con una llamada obtenemos todos los email/id de nuestros amigos y poder cargarlo en la vista.

Depurar Javascript el Cliente/Navegador

Se ha incluido un documento en la práctica 10 donde se explica como depurar clientes desde el navegador.