YouShareApp

Cambios de la api y descripción de las funcionalidades

3r Sprint - 31/05/2024

Grupo Delta 11
Albert Comas
Victor De Lamo
Lluc Gracia
Jose Miguel Santos

ÍNDICE

1. Cambios de la api

1.1. Problemas encontrados

1.1.1. Json del hilos/:id incompleto

Mientras programábamos la funcionalidad de mostrar un hilo en concreto nos dimos cuenta que necesitábamos mandar la información de cuántos likes, dislikes y el contador de cuantos comentarios tiene. También nos dimos cuenta que teníamos que hacer un get adicional para tener el username del usuario del hilo.

1.1.2. Ordenar comentarios incorrecto

En la anterior entrega no nos funcionaba correctamente la ordenación de comentarios. Nos dimos cuenta que era porque primero ordenábamos y luego se llamaba a listar comentarios sin la ordenación aplicada.

1.1.3. Json del hilos/:id sin información del magazine

Al necesitar mostrar la información del magazine al cual pertenece el hilo o link, nos dimos cuenta que solo con el id nos obligábamos a hacer una petición extra a la api por cada hilo, la cual pidiera toda la información del magazine.

1.1.4. Json del users/:id/hilos o comments o boosts incompleto

Al necesitar mostrar un usuario, nos dimos cuenta que al mostrar los hilos creados, commentarios creados y boosts de hilos que ha dado el usuario no teníamos toda la información necesaria para poder mostrar las mismas funcionalidades que se dan en hilos/ o hilos/:id.

1.1.5. No se puede subir imágenes al s3

Pensábamos que como funcionaba en la primera versión con el frontend en ruby on rails, la api para subir imágenes al perfil funcionaría también. No obstante no funcionaba.

1.1.6. Json search/ incompleto

Durante el proceso de desarrollo de hilos/ el jbuilder index había cambiado para obtener más información de los hilos. Esto hizo que la funcionalidad de buscar en hilos o links se descuadrase y no enviará el mismo json.

1.1.7. Json hilos/ información incompleta

Cuando realizamos la vista de un hilo o de varios vimos que necesitábamos mostrar el nombre del creador y su id, originalmente solo mostrabamos el id del usuario provocando que tuviéramos que realizar una petición extra por cada hilo para poder obtener el username.

1.1.8. Json magazines/ información incompleta

Cuando realizamos la demo del sprint 2 vimos que en el json de magazine no teníamos la información necesaria para saber quién era el propietario ni la cantidad de hilos, comentarios, publicaciones o suscripciones que tenía el magazine.

1.2. Soluciones aplicadas

1.2.1. Json del hilos/:id incompleto

Para poder solucionar este error, hemos modificado el jbuilder del show y el de _hilo de manera que mostrasen las variables :likes, :dislikes, @hilo.comments.count. y añadimos el username del propietario

1.2.2. Ordenar comentarios incorrecto

Para arreglar la ordenación tuvimos que ir al backend e implementar bien la ordenación. Para hacerlo modificamos la función de show en el controlador de hilos, quitando de ahí la ordenación y poniéndola justo después de listar los comentarios anidados (en el comentarios_helper).

1.2.3. Json del hilos/:id sin información del magazine

Para solucionar este problema decidimos editar los jbuilder de show y de _hilo de manera que mostrara un objeto magazine con las siguiente información :id, :name, :title.

1.2.4. Json del users/:id/hilos o comments o boosts incompleto

Para poder solucionar este problema tuvimos que crear dos funciones en el controlador de usuarios que generase un json exactamente igual que en el show de hilos o show de comentario. De esta manera ya podremos realizar las mismas funcionalidades en la vista de usuario que en hilo/:id.

1.2.5. No se puede subir imágenes al s3

Para solucionar este problema añadimos cover y avatar a los parámetros permitidos.

1.2.6. Json search/ incompleto

Para poder solucionar este problema tuvimos que modificar la función del controlador de search que generaba el json de manera que mandara la misma información que la de hilos/ o hilos/:id.

1.2.7. Json hilos/ información incompleta

Para poder solucionar este problema decidimos realizar un objeto propietario en los jbuilder de hilos de manera que obtuvieramos el id y el username con la misma llamada.

1.2.8. Json magazines/ información incompleta

Para poder solucionar este problema decidimos realizar cambios en el jbuilder magazine de manera que tuviéramos un objeto propietario con id y username. Por otro lado añadir cuatro atributos que mostraran cada uno el número de hilos, comentarios, publicaciones y suscriptores. De manera que podamos ver toda la información igual que en el sprint 1.

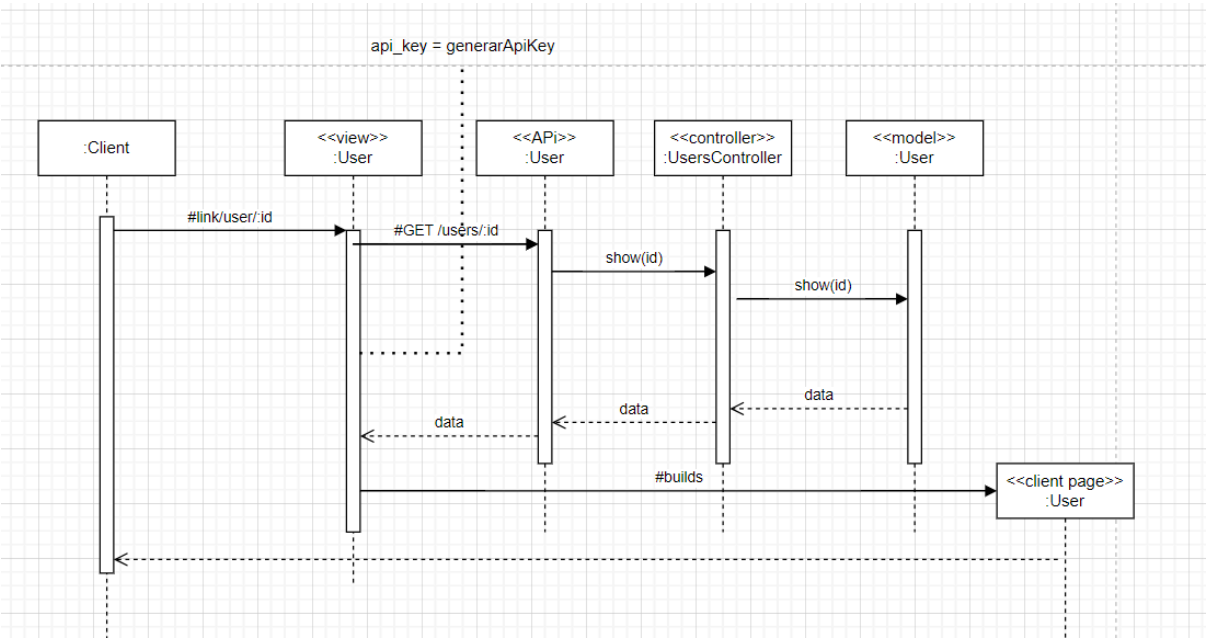
2. Descripción de las funcionalidades

A continuación se van a mostrar las funcionalidades de visualizar información del usuario logueado y modificar la biografía del usuario logueado mediante un diagrama de secuencia por cada funcionalidad

2.1. Visualizar información del usuario logueado

Para poder ver toda la información del usuario logueado, es necesario que el sistema se guarde una api key para que luego la podamos pedir, sino, no se podrán visualizar los boosts del usuario logueado. Esta api se obtiene al seleccionar un usuario del desplegable de usuarios que se encuentra en el navigation bar.

Al intentar hacer un link hacia user/:id (por ejemplo seleccionar el autor de un hilo), el router delega el trabajo hacia la vista User de nuestro sistema. Esta vista hace un GET a la API del backend con la url /users/{:id}.json, el cual este id ya se obtiene automáticamente cuando el usuario hace el link. Seguidamente, la API devuelve toda la información necesaria para mostrar la pàgina (username, email, description, avatar_url, cover_url, user.comments,user.hilos,user.boosts) y se renderiza.



2.2. Modificar la biografía del usuario logueado

En este diagrama de secuencia, partimos del diagrama anterior donde se nos muestra la información del usuario logueado.
Para poder editar el usuario, primero se tiene que presionar el botón de editar usuario, que es un link que nos va a llevar a una vista para poder editar el usuario. Una vez en este vista podemos editar los valores del usuario como el username, el email o las imágenes. Una vez editado los campos al presionar el botón Guardar Cambios, este invoca un método de tipo PUT a la url /users/{:id}.json y rellenar un archivo .json que contiene los datos modificados. A la petición se le pone como header la api_key del usuario seleccionado con el navbar(sino es el correcto, saldrá un error de Unauthorized). Una vez procesada la petición el UsersController se guarda este nuevo data y hace un update al modelo User. Al modificar el modelo, se renderiza otra vez la página del perfil de usuario.