TALKIFY (alpha11)

Integrantes: Miguel Goñi, Ivan López, Sara Chillón y Ferran Martínez

Cambios Realizados en la API desde el sprint 2:

post_controller:

Información añadida al devolver un post en las operaciones: index, show, search

Añadir en la respuesta del GET Post, un objeto magazine y objeto user:

- Descripción: Se ha añadido información adicional a los objetos Post. Ahora, los objetos Post incluyen los objetos magazine y user creador del Post, permitiendo pasar tanto el nombre como el ID de cada uno de estos objetos.

- Objetivo: Permitir que los usuarios puedan ver y acceder a la información de la revista y del creador del post. En la aplicación se ha de mostrar el nombre y usamos el ID para poder redirigir al usuario a la página correspondiente al pulsar tanto en el nombre del creador como en el nombre de la magazine.

Añadir Booleanos is_Upvoted, is_Downvoted y is_Boosted a Post

- Descripción: Se han añadido tres nuevos campos booleanos a los objetos Post para indicar si el usuario actual ha votado positivamente, negativamente o ha impulsado el post.

- Objetivo: Facilitar la interacción del usuario con los posts, permitiendo ver de inmediato si ya han votado o impulsado un post. Esto se muestra con colores en el caso de upvote y downvote y en boost se muestra que el usuario como boost(1) si como usuario ya había impulsado este post.

Ejemplo de post:

1:

- **body: "Un body de un post"**
- **comments_count: 0**
- **created_at: "2024-05-27T09:03:14.295Z"**
- **downvotes_count: 0**
- **id: 29**
- **is_boosted: true**
- **is_downvoted: false**
- **is_upvoted: false**
- **link: false**
- **magazine: {id: 1, title: 'HOLA'}**
- **title: "Un post"**
- **updated_at: "2024-05-27T09:03:14.295Z"**
- **upvotes_count: 0**
- **url: " "**
- **user: {id: 4, email: 'sara.chillon@estudiantat.upc.edu', full_name: 'Sara Chillón Domínguez'}**

user_controller:

Añadir el Post con toda la información a user_posts y user_boosts

- Descripción: Se ha actualizado la API para que al pedir o los posts o los boosts de un usuario también devuelva toda la información que se ha añadido para el post_controller.

- Objetivo: Tener posts funcionales y poder visualizar que acciones el usuario iniciado ha realizado a estos posts.

comment_controller:

Añadir Objeto user y post a Comment

- Descripción: Se ha enriquecido el objeto Comment para incluir los detalles del usuario que creó el comentario y el post al que pertenece.

- Objetivo: Facilitar la visualización y navegación entre comentarios y sus respectivos autores y posts.

Añadir Booleanos is_Upvoted, is_Downvoted y is_Author a Comment

- Descripción: Se han añadido tres nuevos campos booleanos a los objetos Comment para indicar si el usuario actual ha votado positivamente, negativamente o es el autor del comentario.

- Objetivo: Mejorar la interacción del usuario con los comentarios, permitiendo ver de inmediato si ya han votado o si son los autores del comentario.

Incluir Todas las Reviews Sin Límite en Comment

- Descripción: Se ha modificado la API para que al solicitar los comentarios de un post, se incluyan todas las reviews sin límite.

- Objetivo: Proporcionar una visión completa de todas las interacciones y feedback relacionados con un comentario.

Diferenciar entre un solo comentario o una array de ellos:

- Descripción: Se ha modificado la API para contemplar si lo que devuelve es un solo comentario o más de uno, de ésta manera se distingue entre un objeto o una array

- Objetivo: Facilitar el manejo de comentarios en frontend y no tener que analizar si es un array o no lo es.

Ejemplo de comentario:

- **body: "Comment 2!"**
- **created_at: "2024-05-28T17:45:07.522Z"**
- **downvote: 0**
- **id: 18**
- **is_author: true**
- **is_downvoted: false**
- **is_upvoted: false**
- **parent_comment_id: 9**

- **post: {id: 16, title: "Un post"}**
- **replies: []**
- **updated_at: "2024-05-28T17:45:07.522Z"**
- **upvote: 0**
- **user: {id: 2, email: "miguel.goni@estudiantat.upc.edu", full_name: "Miguel Goñi Fusté"}**

magazines_controller:

Añadir parámetros 'threads' y 'comments', y booleano 'isSubscribed' a Magazine:

- Descripción: se han añadido estos tres parámetros más al json de respuesta de los GETs de /magazines y /magazines/:id que muestran el número de threads que contiene la magazine, su número de comentarios y si el usuario que tiene la sesión iniciada está suscrito o no a la magazine.
- Objetivo: facilitar la interacción del usuario con las magazines, ofreciendo el número de threads y comentarios totales que contiene la magazine, además de un booleano indicando si el usuario está suscrito o no a la magazine en cuestión, información que anteriormente no estaba disponible de manera directa a los usuarios.

Añadir objetos "magazine" y "user" a los posts devueltos en la respuesta a GET /magazines/:id/posts:

- Descripción: se han añadido estos dos objetos al json de respuesta del GET de /magazines/:id/posts, con tanto el id como el nombre de ambos objetos.
- Objetivo: permitir que los usuarios tengan acceso al usuario creador y la magazine a la que pertenecen los posts, de la misma manera que se presenta la información en los GETs de /posts y /posts/:id

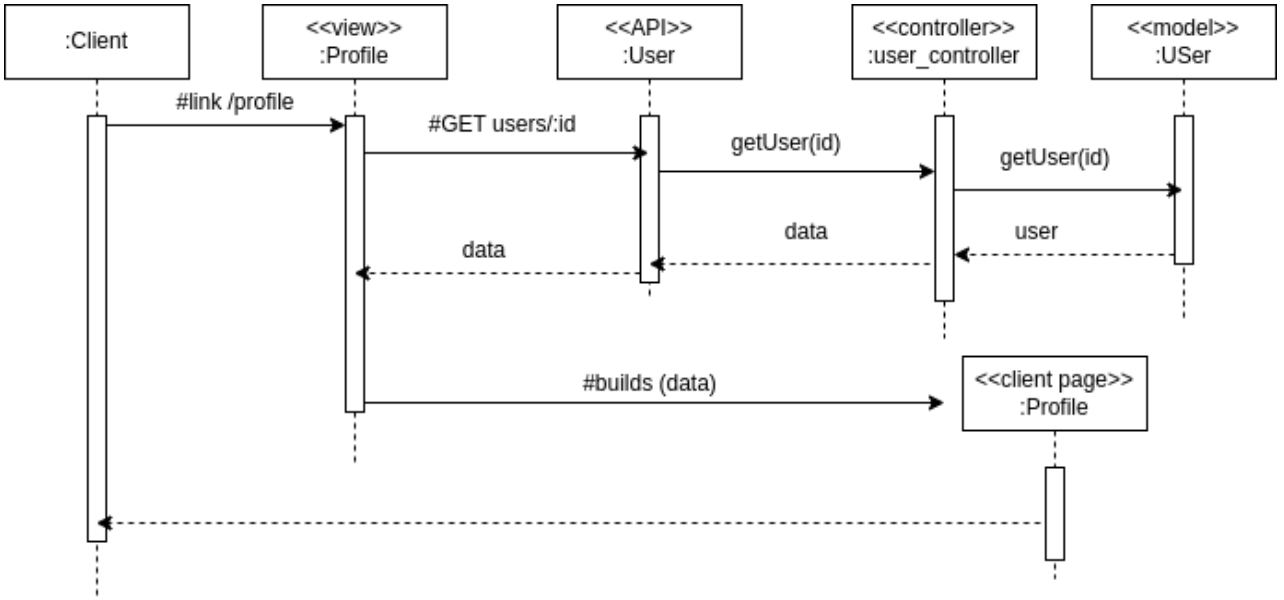
Ejemplo de magazine:

- **"id": 1**
- **"name": "Magazine"**
- **"title": "HOLA"**
- **"description": "dsadasdsad"**
- **"rules": "dsad"**
- **"created_at": "2024-05-16T10:33:44.196Z"**
- **"updated_at": "2024-05-16T10:33:44.196Z"**
- **"threads": 5**
- **"comments": 12**
- **"subscribers": 2**
- **"isSubscribed": true**

Diagramas de secuencia

Acceso al Perfil

Nuestra web, al iniciarse, automáticamente inicia sesión con un perfil determinado. Para cambiar de usuario logueado, se puede escoger el usuario deseado a través de un menú desplegable ubicado en la parte derecha de la barra de navegación. Para visualizar el perfil del usuario logueado, el usuario puede hacer clic en el botón de perfil que lo redirige a la ruta /profile. Esta ruta accede al ID del usuario logueado almacenado en el localStorage y realiza una solicitud GET al servidor para obtener toda la información necesaria. Una vez recibida la información, se despliega la vista del perfil, permitiendo al usuario realizar todas las funcionalidades disponibles.



Editar Perfil

Este diagrama de secuencia comienza donde termina el diagrama de visualización del perfil, con el perfil del usuario ya desplegado, y apretando el botón de editar perfil. También se puede acceder a esta funcionalidad utilizando el menú desplegable ubicado en la parte derecha de la barra de navegación. Al hacer clic en el botón de "Editar Perfil", el usuario es redirigido a una interfaz de perfil con el formulario de edición desplegado. La ruta de edición accede al ID del usuario logueado almacenado en el localStorage y carga la información actual del perfil en un formulario editable. Después de realizar los cambios deseados, el usuario puede guardar las modificaciones, lo que envía una solicitud PUT al servidor con la información actualizada. El servidor procesa la solicitud y actualiza los datos en la base de datos. Una vez completada la actualización, la vista del perfil se actualiza con la nueva información, reflejando los cambios realizados por el usuario.