Projecte d'ASW, 3r Sprint

Pàgina web inspirada en el client Kbin

Júlia Tena Domingo

Miquel Amorín Díaz

Alba López Ruiz

Agustí Costabella Moreno

TAULA DE CONTINGUTS

1. CANVIS REALITZATS A LA API

En aquest apartat, es detallen els errors detectats durant el procés d'integració del backend i el frontend de l'aplicació. A més, es descriuen les solucions que s'han implementat per resoldre aquests problemes.

1.1. Problemes trobats

1 . Centrant-nos en els posts i comments, s'ha vist que per marcar els botons del frontend, si l'usuari havia donat like, dislike, boost (només posts) ens faltava alguna informació per saber si marcar el botó o no. D'altra banda, a les magazines també faltava aquesta informació per saber si l'usuari estava subscrit o no.

2. També s'ha trobat que no es podia saber si l'usuari loguejat a l'aplicació era l'autor dels posts i comments per saber si mostrar el botó d'editar i el de delete.

1.2. Solucions implementades

1 . De cara als posts, a l'hora de fer el frontend, s'han hagut d'afegir una sèrie de paràmetres al controlador de posts indicant si l'usuari que fa la request a /posts, havia donat like, dislike o boost a cada post.

D'aquesta manera els botons de like, dislike, boost estan ja marcats i fan l'acció contrària que és treure el like, dislike o boost.

Aquest canvi s'ha aplicat al *GET* a */posts* i */posts/:id*. També, al *POST* i *DELETE* a */posts/:id/like*, */posts/:id/dislike*, */posts/:id/boost* i al *PUT* a */posts/:id*.

Un canvi molt semblant s'ha fet al comments controller, afegint si l'usuari havia donat like o dislike al comentari.

Aquest canvi s'ha implementat en el *GET* a */posts/:id/comments*, */posts/:id/comments/:id*, *POST* i *DELETE* a /posts/:id/comments/:id/like, */posts/:id/comments/:id/dislike* i *PUT* a */posts/:id/comments/:id.*

El canvi a les magazines ha sigut al fer *GET* a */magazines* i a */magazines/:id.*

Per altra banda, aquests dos canvis a posts i comments s'han hagut de fer a users controller quan obtenim els posts o comentaris d'un donat usuari. Concretament, *GET* a */users/:id/posts* i */users/:id/comments.*

2. S'ha afegit un paràmetre a posts i comments indicant si l'user és *l'owner* o no, per mostrar el botó d'editar i de delete.

2. DESCRIPCIÓ DE FUNCIONALITATS

A continuació es detallen les funcionalitats relacionades amb la visualització de la informació d'un usuari “loguejat” i la modificació de la seva biografia.

2.1. Visualitzar informació d’usuari “loguejat”

En aquesta funcionalitat, els usuaris “loguejats” han de poder veure el seu perfil per administrar-lo. Han de veure la informació del seu perfil, és a dir, el nom d’usuari i, en el cas que tinguin, la descripció, la imatge de cover i l’avatar. A més a més, han de poder veure un llistat dels threads i comments, fets per l’usuari, i el llistat boosts.

Per tal de veure la informació d’un usuari loguejat, el sistema ha de tenir guardada l’*api key* d’aquest.

La vista *User* conté el botó “View Profile” per anar al perfil de l’usuari *loguejat*. Així doncs, es fa un GET a l’API amb la url */users/{:user_id}*. A la petició s’afegeix com a header *l’apiKey* de l’usuari loguejat. La crida ens retorna la informació necessària, en format *json*, per mostrar per pantalla el perfil complet de l’usuari.

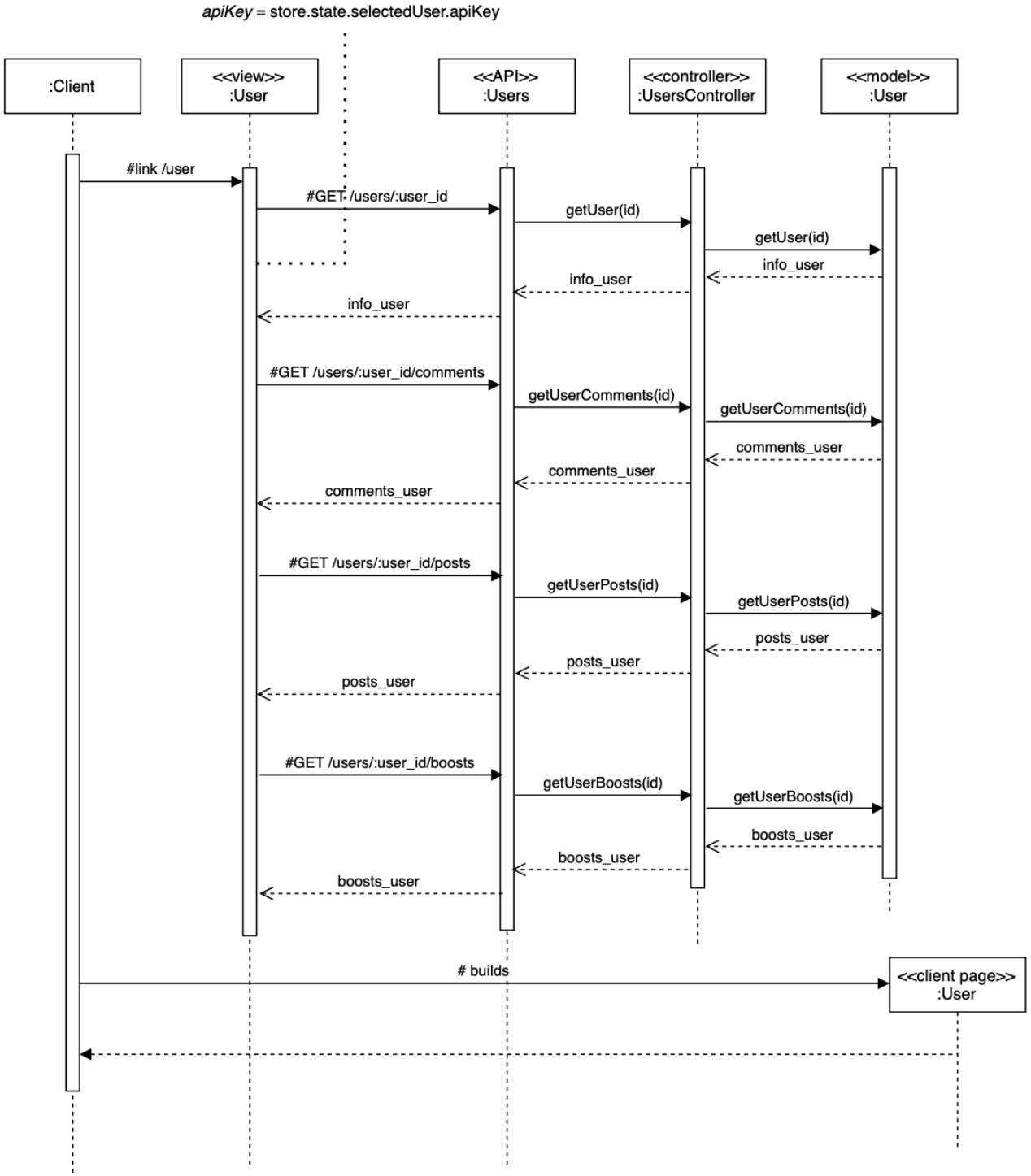

Diagrama 1: Diagrama de seqüència de la visualització d'un usuari loguejat

2.2. Modificar biografia d'usuari "loguejat"

En aquest cas, els usuaris loguejats han de poder editar les dades del seu perfil, en específic, la biografia.

De la mateixa manera que per veure la informació d'un usuari "loguejat", el sistema ha de tenir guardada l'*api key* d'aquest.

Partim del diagrama anterior, d'on hem obtingut la informació de l'usuari loguejat. Així doncs, en la vista *EditUser* veiem els camps de l'usuari loguejat que es poden editar. Aquesta vista, conté una àrea de text per poder editar la biografia (en el nostre model de user l'anomenem description, però és exactament el mateix). Un cop modificada la biografia, s'ha de prémer al botó "Update Profile" i la vista invoca un PUT a l'api amb l'url */users/{:user_id}* enviant un .json amb la nova biografia. A la petició s'afegeix com a header *l'apiKey* de l'usuari loguejat. Un cop arribada la petició, el UserController la modifica i guarda els canvis.

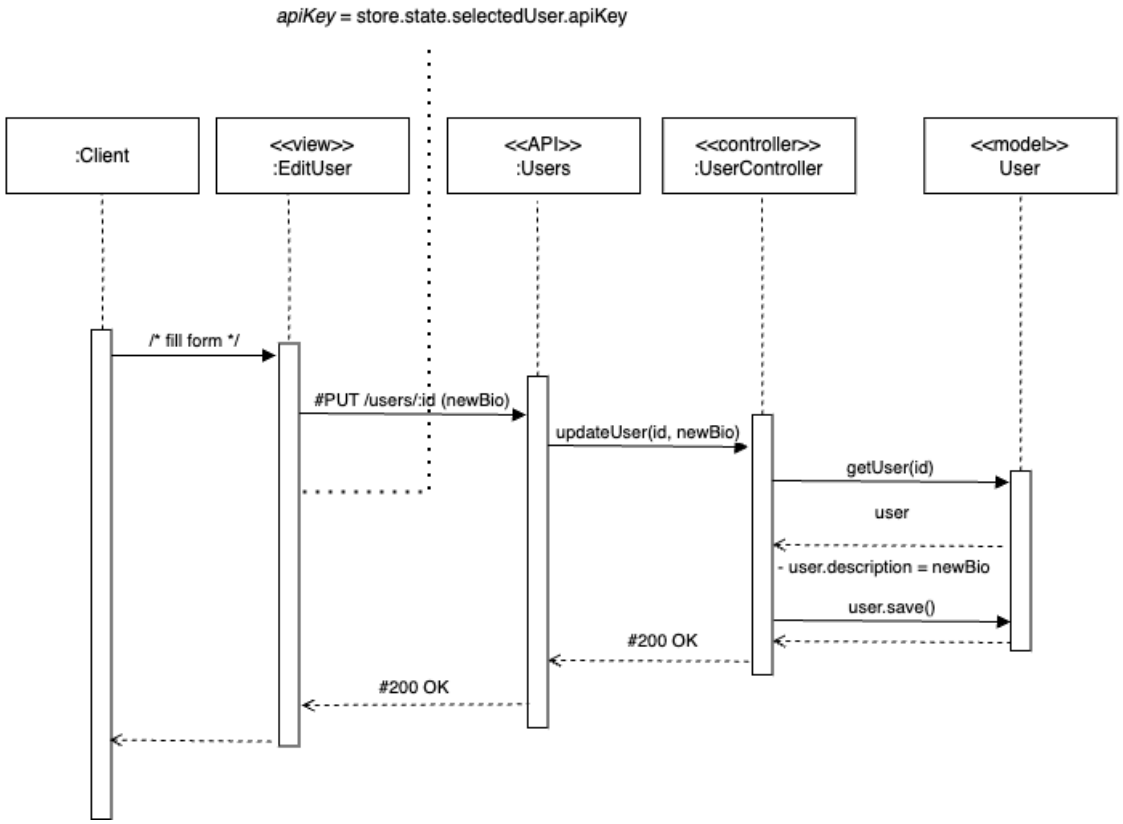

Diagrama 2: Diagrama de seqüència de la modificació de la biografia d'un usuari loguejat