HACKER NEWS

Heroku Client Link

Nom	Bitbucket
Aleix Ventura Nasarre	Aleix Ventura
Ángel González Jurado	Ángel González Jurado
Elia de Biasi	DBE
Júlia Gasull i Navarro	Júlia Gasull
Roger Garcia Nevado	“-R“ al títol del issue - no deixava entrar al repositori

Índex

1. Canvis realitzats a la API-REST (back-end)

Solució als errors

Els primers canvis que vam haver de realitzar al *Back* en *Amazon Educate* van ser per solucionar errors que vam cometre a l'anterior entrega.

A diferència d'abans, ara es retorna l'objecte actualizat o creat en un *post* i no només l'*status*, i es mantè constància a l'hora de demanar un tipus de paràmetre per fer una búsqueda en els *Gets*.

També vam haver de migrar els vots a un controlador pròpi i assignar-lis les seves pròpies crides *gets, post* i *put* per tal de generar els *JBuilders* dels vots i poder accedir a ells des d'una crida, ja que abans estaven integrats en el *contribution_controller* i només es retornaven quan es feia una crida *get* a *contributions*.
Les crides per fer un *unvote* ara en comptes d'agafar l'identificador del *vote* agafa el identificador de la contribució a la que es vol desvotar.

Refactor de les request i responses

Una vegada corregit el feedback obtés, el *Front-End* va començar a fer ús de les operacions que hi havien a l'API però ens vam trobar que hi faltava informació. Aquests problemes van provocar uns altres canvis a algunes crides.

Ara ens calia retornar el *karma* d'un usuari junt amb totes les seves dades.
També afegir un atribut al resultat obtés a les *contributions* de l'usuari que l'ha creat, i afegir les relacions d'indexs amb els *comments* i *replies* d'una contribució, en comptes de només els id i després haver de fer un *Get* per cada un d'ells. De manera que per cada *contribution* hi ha un array de Contributions que corresponen als seus *comments*, cada un d'aquest *comments* també té el mateix atribut, i així successivament fins a obtenir l'arbre sencer de tots els *comments* i *replies*.

Un cop aquest refactor era funcional vam anar actualitzant les responses per tal de facilitar les consultes fetes de *Front* i reduir així el nombre de peticions necessàries per obtenir la informació.

Es va facilitar un *Get* de user per email, en comptes de per ID (que aquest es va mantenir), ja que des del client era més pràctic perquè és el que s'obté des d'un primer moment.

En els *Gets* de les contribucions, on abans es retornaven les contribucions i els vots realitzats al sistema, ara existeix una altra operació complementària on es retornen les contribucions amb un booleà per saber si l'usuari amb un id passat per paràmetre ha votat la contribució o no, en comptes d'haver de fer dues crides per cada contribució (una per obtenir la contribució i l'altre per obtenir els vots) i a més després implementar la lògica per a determinar si la contribució ha estat votada o no.

També s'ha afegit un endpoint que donada un *id* d'una contribució et retorna la contribució arrel del arbre.

JBuilder

Tots aquests canvis respecte el que es retorna a les crides es van fer mitjançant el *JBuilder* per a que des del servidor de AWS es retornessin les dades desitjades en format *JSON* ja que l'*scaffolding* que vam fer al inicialitzar el projecte no es va generar amb totes les dades i models finals, de manera que hi havia atributs (com per exemple de quin tipus es una contribution, o els vots) que no es retornaven en els *JSON*.

A part d'aquests canvis, com s'ha comentat en l'anterior apartat, tots aquells atributs que s'han hagut de facilitar per a reduir el nombre de crides també s'han hagut d'afegir als fitxers *JBuilder*.

També s'ha hagut de controlar el valor d'aquestes variables des dels controladors, que són les que es retornaran mitjançant les crides i el *JBuilder*.

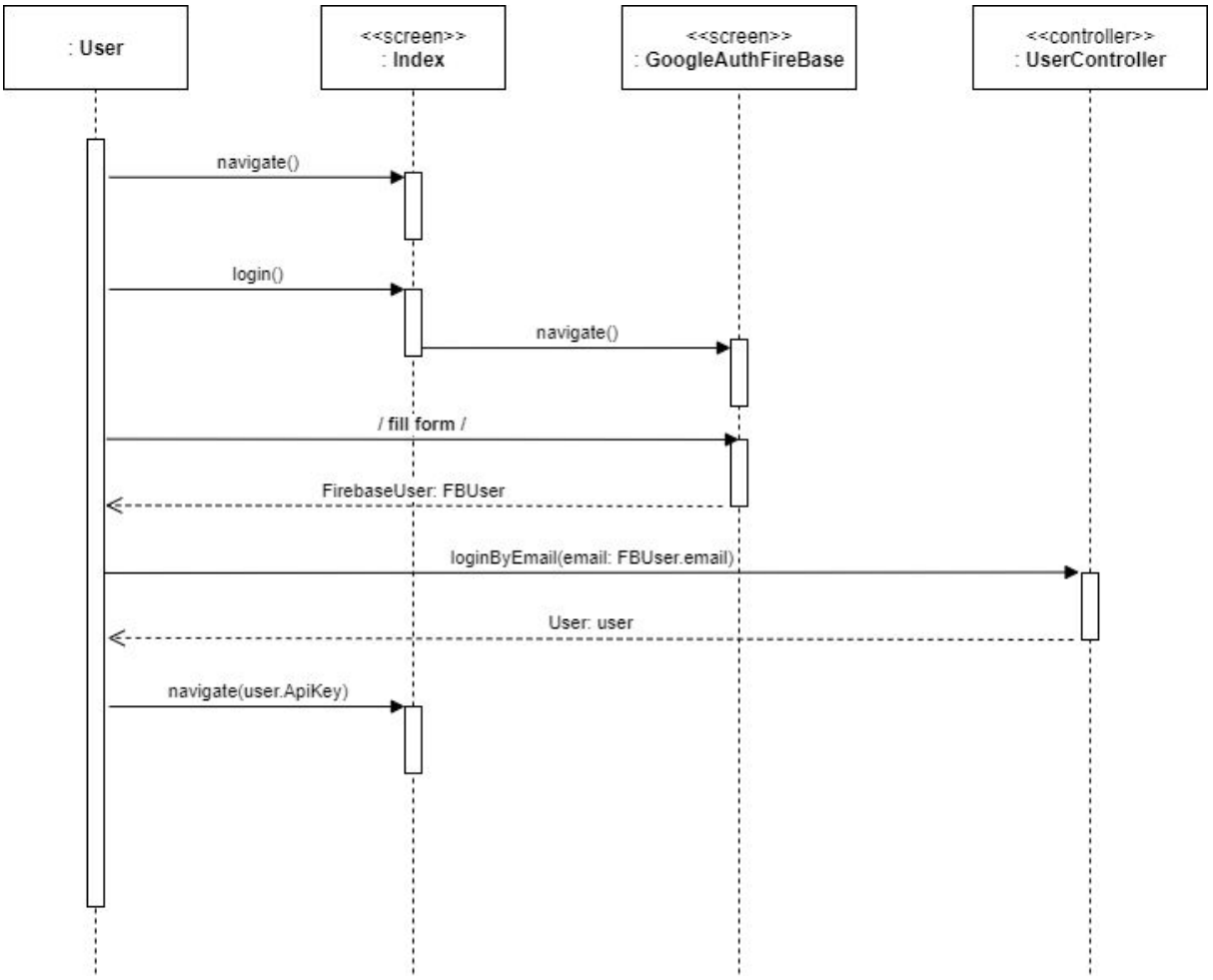
Manteniment del client des de AWS

Alguns d'aquest canvis realitzats han afectat a les antigues *Views* de *Front-End* de *Amazon* pel que hem hagut d'actualitzar-les lleugerament.

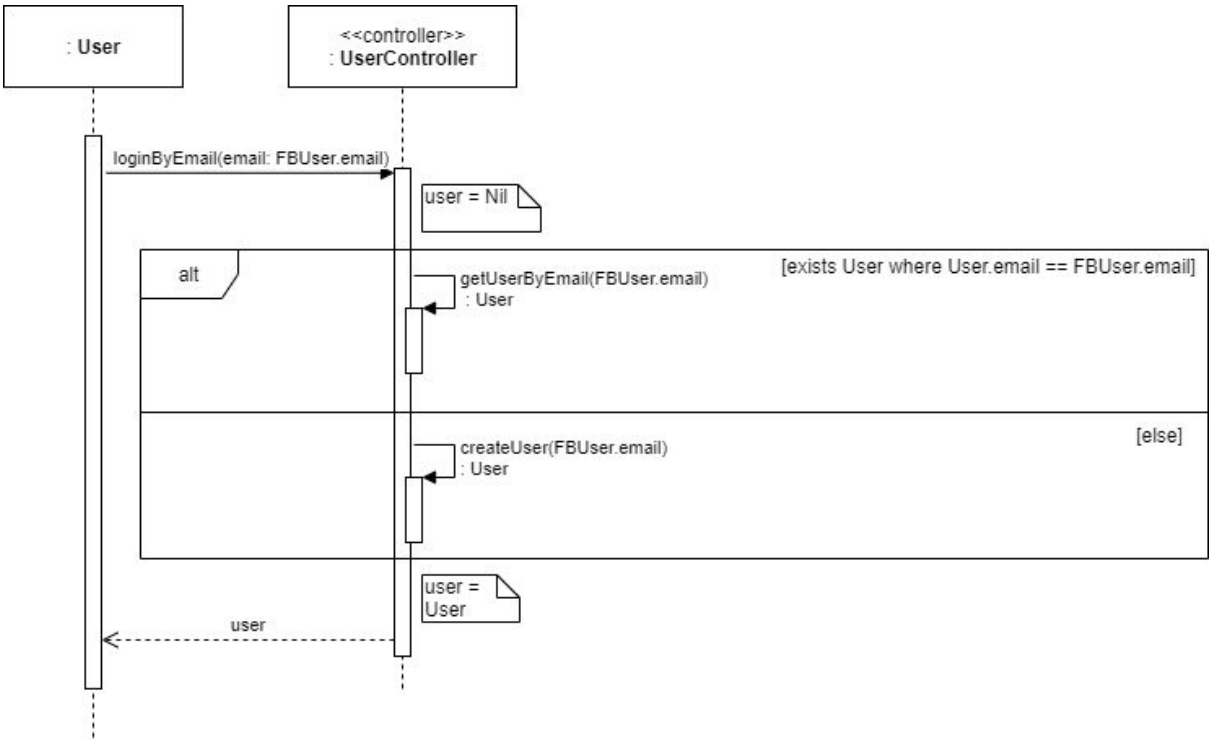
Tot el que correspon als vots s'ha hagut de canviar ja que, com ja s'ha comentat anteriorment, s'han migrat a un controlador pròpi i les crides anteriors ja no funcionen, per tant s'ha hagut de canviar a tots els *html* la ruta corresponent als *upvotes* i els *downvotes*.

2. Funcionament del mecanisme d'autenticació per *Firebase*

Desde la *Front API* de *Angular* es podrá iniciar el procès de login. Aquest farà connexió amb l'API de Firebase de Google i mostrarà una pantalla per escollir amb quin correu electrònic fer login. Una vegada escollit l'usuari desitjat, *Front* haurà obtingut informació d'un usuario de *Firebase* amb el email i altra informació corresponent dels usuaris. Entre aquestes dades es troba la informació necessària per fer posteriors logins a la base de dades de Firebase.



Fet aquest primer pas, s'envia una petició al controlador de *Back* per fer un login amb el correu electrònic corresponent. En aquest punt *Back-End* realitzarà uns processos segons l'estat de la base de dades en aquest punt. En el cas que l'usuari existeix llavors es recull i es retorna, en cas contrari es crea amb el email passat per paràmetre.



Una vegada recollit l'usuari i amb aquest, la seva API-KEY, la informació es guarda a *Front* per realitzar consultes com un usuari loggejat.

Si després de realitzar aquest procés s'intenta fer login desde el client d'*Amazon*, aleshores l'operació login farà un *redirect* a l'operació de registre de google i al insertar el mateix correu, buscará l'usuari corresponent per guardar els tokens necessaris per a un posterior login desde el mateix client.