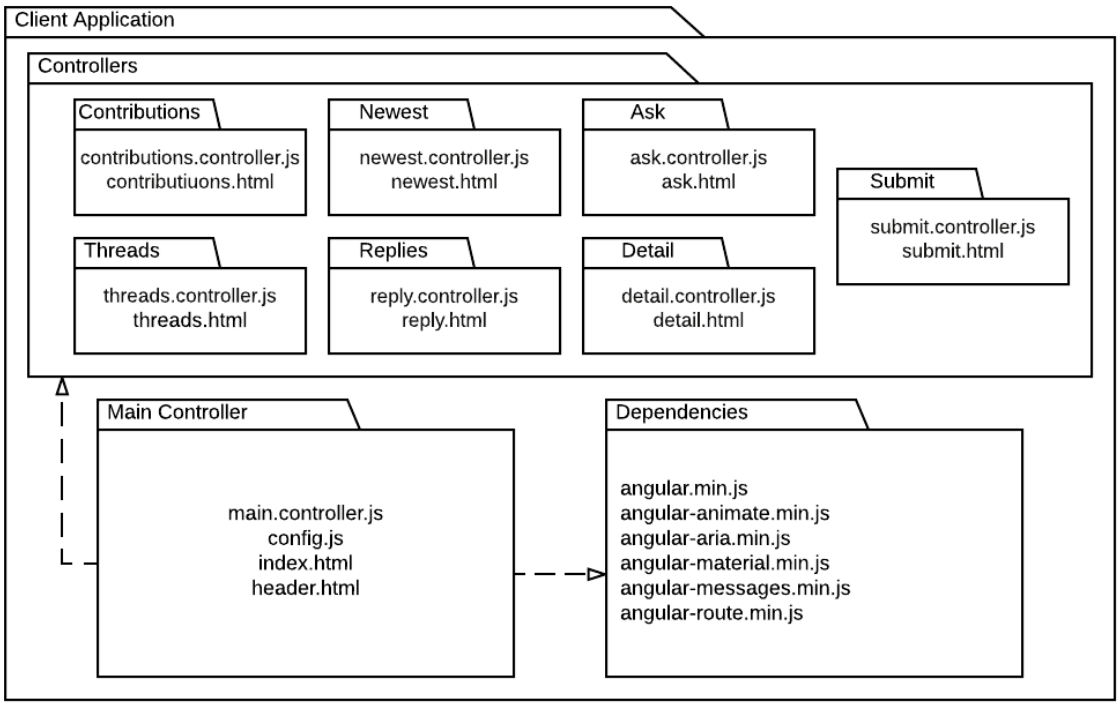
David Aleu
Jordi Guiu
Eduard Maura
Arnau Orriols
Aplicacions i Serveis Web

Disseny Intern

Descripció dels components

La nostra aplicació-client Angular (o *Client Application* en anglès) funciona bàsicament a través de controladors, que són arxius *javascript* que gestionen el funcionament general de les vistes en HTML, definides a banda. Així doncs, d'aquesta manera tenim fins a 7 controladors específics (*Contributions*, *Newest*, *Ask*, *Submit*, *Threads*, *Replies* i *Detail*) cada un associat a un dels recursos de la api. Aquest són els encarregats de efectuar les peticions HTTP a la API i d'omplir les vistes amb les dades que rebin d'aquesta última.

A banda d'això, tenim un altre controlador per sobre del nivell anterior, i aquest és l'anomenat *Main Controller.js*. Dins d'aquest no només ens referim al propi arxiu .js del controlador, sinó que també hi incloem el *index.html*, el *header.html* i el *config.js*, i que juntament amb el *main.controller.js* conformen el core de l'aplicació, que té com a objectiu ser el primer que s'executa al iniciar el servei web, i enllaçar tots els altres controladors de manera que funcionin correctament i quan els hi toca.
Finalment, el component de les *Dependencies* és necessari per a qualsevol aplicació que utilitzi el *framework* de Angular, que són totes les llibreries necessàries que fan falta per utilitzar els avantatges que presenta, tant *Angular Material*, com per a les crides a la API des dels controladors. (per a més informació, veure el següent dibuix).

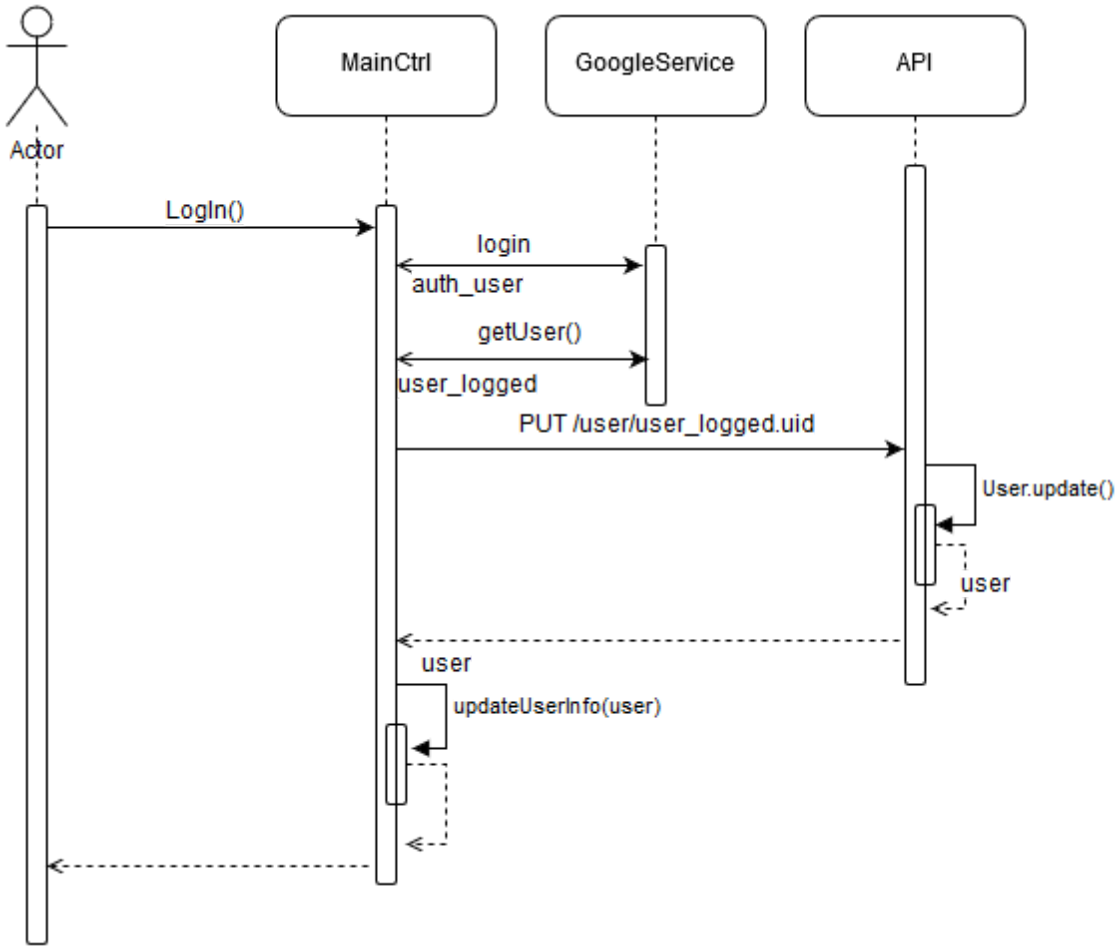


Funcionament del login

Després d'intentar nombroses vegades d'implementar el Log In a través del servei de Google, ens hem vist obligats a què el nostre client funcioni sense aquest.

Totes les accions que es duen a terme al client, les executa un usuari que existeix a la base de dades de Heroku, i per tant, totes les publicacions, comentaris i replies que es facin estaran amb el nom del mateix usuari. En definitiva, és com si estiguéssim sempre loggejats amb el mateix usuari.

La nostra idea a implementar era accedir al servei de Google per obtenir el token d'autentificació fem servir per autoritzar als usuaris a fer servir la API. Un cop aconseguit, hauríem d'actualitzar a la base de dades de Heroku l'usuari en concret que s'acaba de registrar o iniciar sessió perquè les peticions estiguessin a nom seu. El següent diagrama pretén explicar el procés.

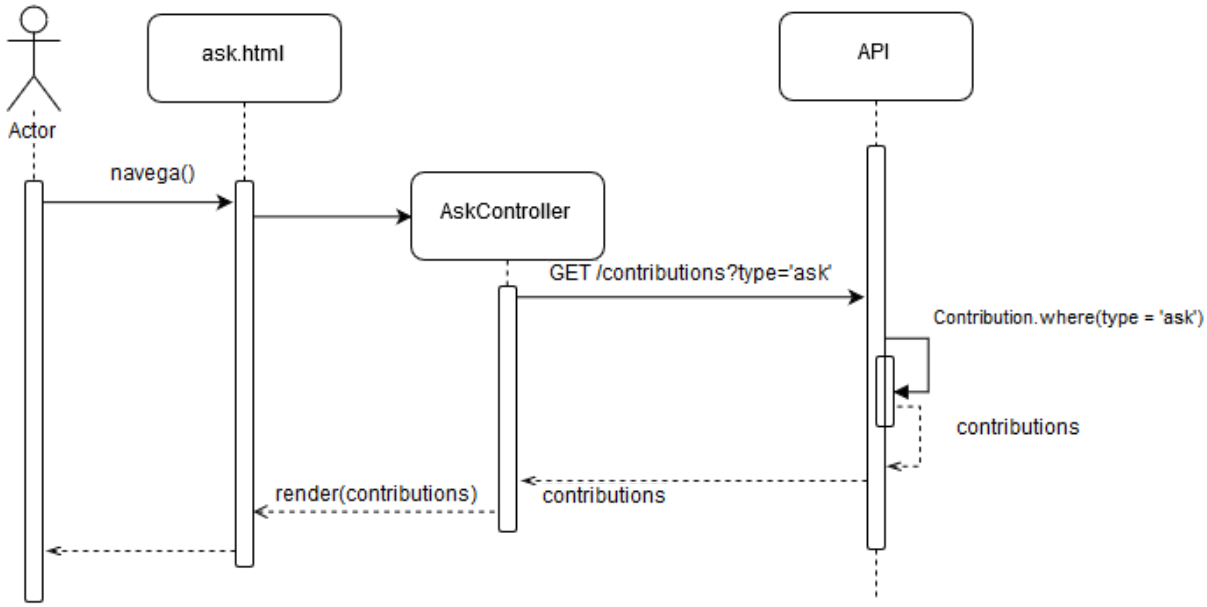


Descripció de les peticions

GET /contributions?type=’ask’

El client navega per la pàgina web fins arribar a les peticions del tipus ask (per exemple, clicant al menú *Ask*). Allà l’índex el redirigeix a la vista i en aquest moment el controlador de contribucions efectua la petició GET /contributions?type=’ask’. Fins que aquesta no respon, la vista resta carregant-se i en el moment que el controlador obté la resposta JSON omple la vista amb la informació.

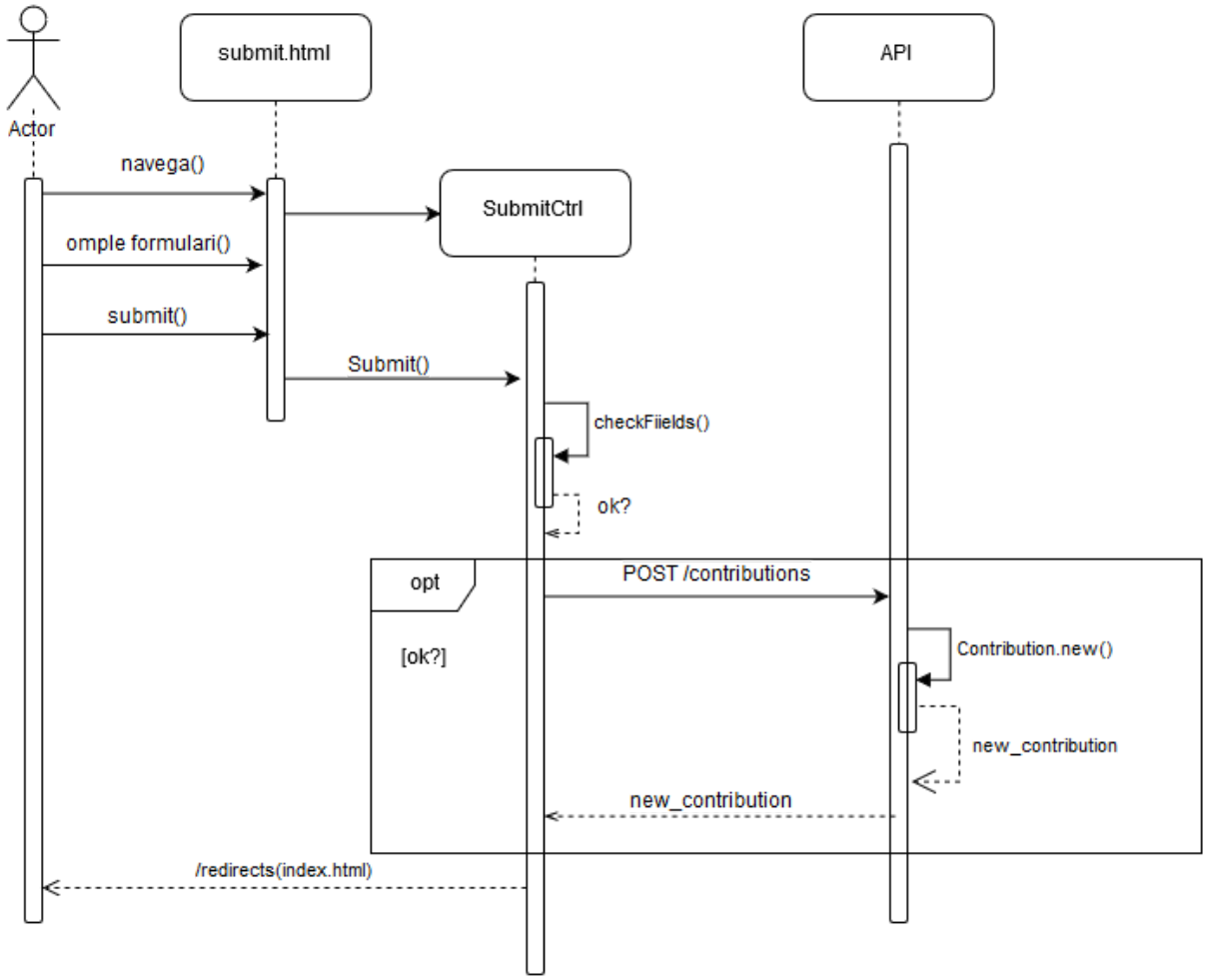
Com que la vista és general és estàtica i el que fem és omplir-la d’informació, la pàgina ask.html no és pas generada pel controlador, sinó que ja existeix prèviament.



POST /contribution

En el segon exemple, podem veure com creem una nova contribució. Suposem que el client navega fins a la vista de *submit.html*. Aquesta vista la renderitza el MainController. Allà el client omplirà el formulari i efectuarà l’acció de salvar la nova contribució. A partir d’aquí és on el SubmitContribution entra en joc, efectuant comprovacions de correctesa per a la API (una contribució no pot tenir url i text alhora) i finalment efectua la crida HTTP POST /contributions. Un cop la API respon amb codi 201 (CREATED), el controlador redirigeix al usuari directament cap a la vista principal perquè vegi la seva contribució.

El següent diagrama mostra el flux dels esdeveniments.


Actor
submit.html
API
navega()
SubmitCtrl
omple formulari()
submit()
Submit()
checkFiields()
ok?
opt
POST /contributions
[ok?]
Contribution.new()
new_contribution
new_contribution
/redirects(index.html)