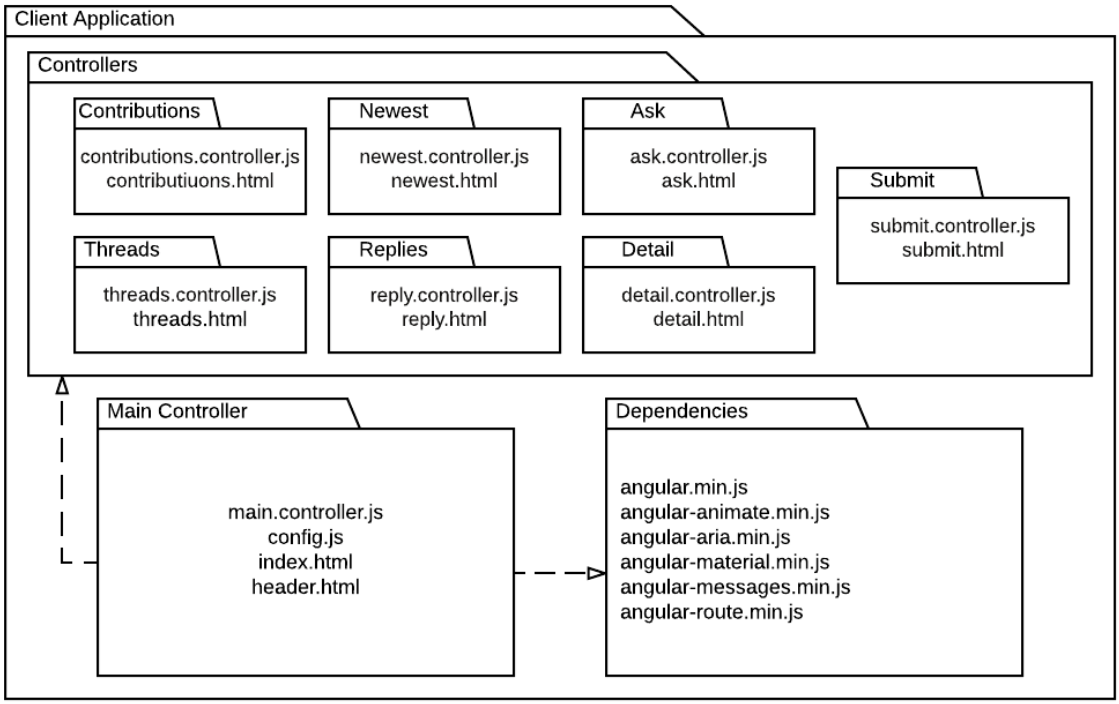
# Disseny Intern:

**Descripció dels components:**

La nostra aplicació-client Angular (o *Client Application* en anglès) funciona bàsicament a través de controladors, que són arxius *javascript* que "controlen" 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*) que tal com indica el seu nom s'encarreguen d'implementar la funcionalitat de tota la interfície gràfica de les vistes, com ara fer les peticions HTTP al servidor o controlar la lògica del servei web.
A banda d'això, tenim un altre controlador per sobre del nivell anterior, i aquest és l'anomenat Main Controller. 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, i és 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 dibuix *Client Application*)



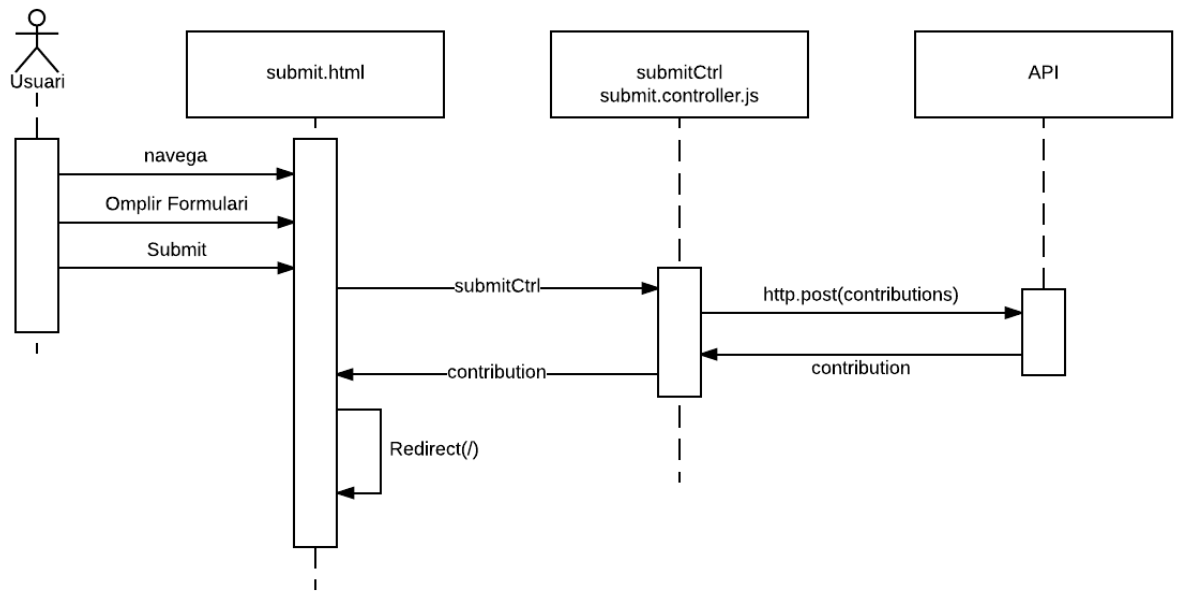
**Funcionament del login:**

Després d'intentar nombroses vegades d'implementar el login no hi ha hagut manera, i per tant, ens hem vist obligats a què el nostre client funciones sense aquest.
Totes les accions que es duen a terme al client, les du a terme 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.

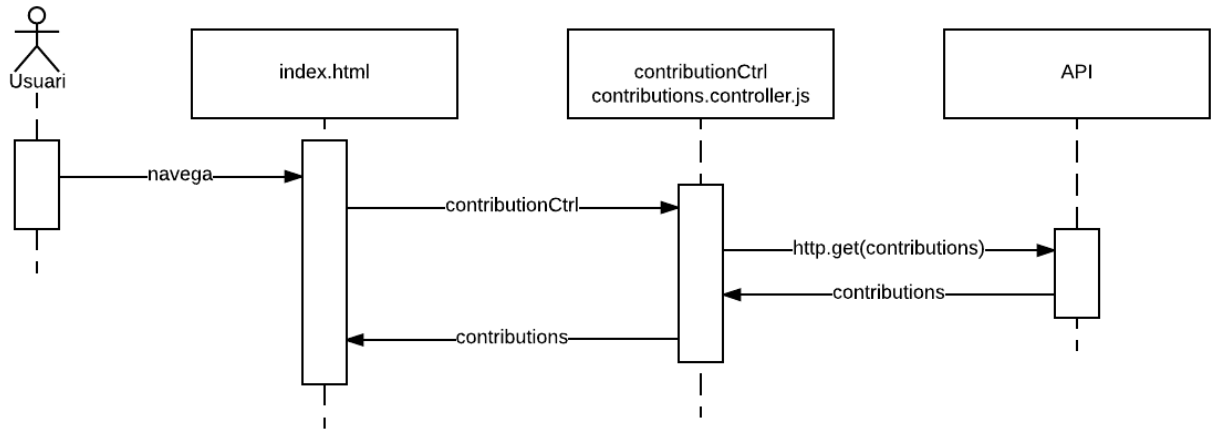
**Descripció de les peticions:**

Per tal d'explicar tot el procés d'una petició en el nostre client, des del moment en què l'usuari genera l'esdeveniment (clica, per exemple) fins a la crida a la nostra API, ens basarem en el cas d'una petició GET de totes les contribucions, tal com podem veure al primer diagrama de seqüència.
En primer lloc, l'usuari genera l'event a la pàgina web clicant el botó corresponent (navega). Llavors és quan index.html redirigeix la petició cap a contribution.html i per tant al controlador corresponent, en aquest cas contributionCtrl, que és qui gestiona la petició realment. Aquest fa una crida a la nostra API a través d'una HTTP GET request de contributions, i la nostra API, ja implementada en anteriors pràctiques, ja sap el que ha de fer i retornar quan rep aquest tipus de petició, i per tant aquesta retorna un llistat de contributions. Llavors és quan la cadena va cap enrere, i així doncs contributionCtrl rep el llistat de contribucions, es comunica amb contributions.html per a mostrar-los un per un i ben estilitzats, i finalment aquest llistat passa a la pàgina web principal o index.html, que és el que al final veu l'usuari per pantalla.
En el segon exemple, podem veure el mateix exemple però amb un POST de contributions a través de submitCtrl.

GET:



POST:


Usuari
index.html
contributionCtrl
contributions.controller.js
API
navega
contributionCtrl
http.get(contributions)
contributions
contributions