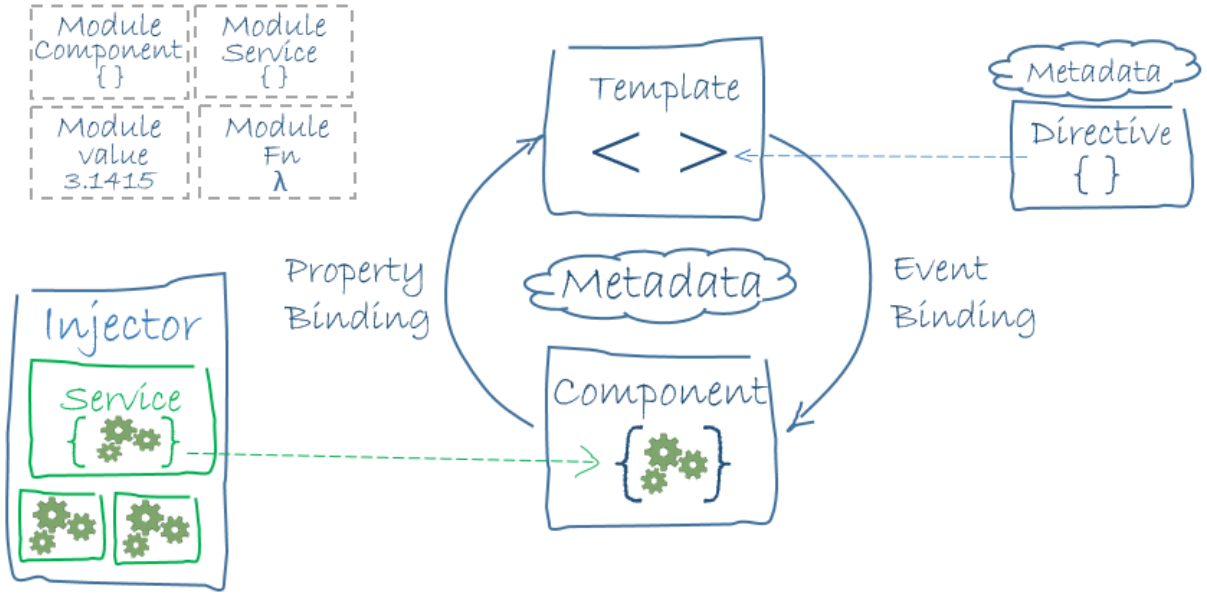
Per la tercera pràctica hem fet servir angular 5 un framework basat en javascript. Angular serveix per construir clients basats en HTML i Javascript amb l'objectiu de que el pes de la lògica i el renderitzat el porti el propi navegador, en comptes del servidor.

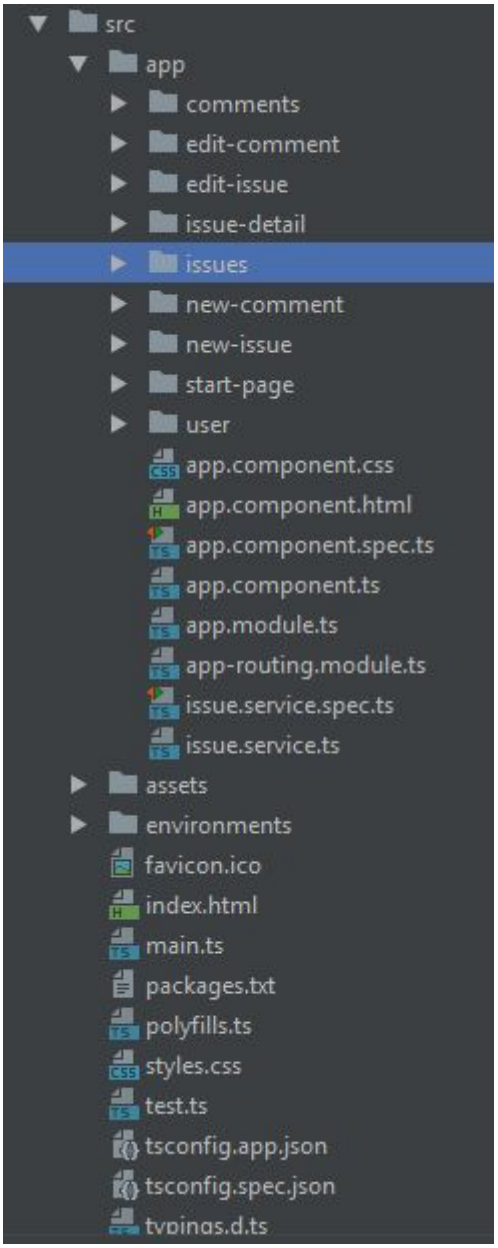
Per crear apps tenim diferents passos diferenciats:

- Composem plantilles HTML
- Escribim components per gestionas les plantilles i directives que afecten al comportament dels components
- Encapsulem la lògica de l'aplicació en serveis
- Definim un mòdul principal que li diu a Angular què és el que inclou la app (altres mòduls) i com compilar-ho

Tipic diagrama d'arquitectura d'una aplicació d'angular:



Disseny intern



Com podem apreciar a la imatge, Angular es basa en un disseny per components, cada component consta d'un css si volem per donar estil, un html que es la part visible i un ts que es la part lògica.

En el nostre cas tenim el app component, que es el nostre core, es qui te tot l'enrotament de laplicació i qui coneix els demés components que formen l'app.

Per començar ens trovem l'start-page, allà trobarem tot el relacionat amb la divisió d'espais i on tenim la navbar.

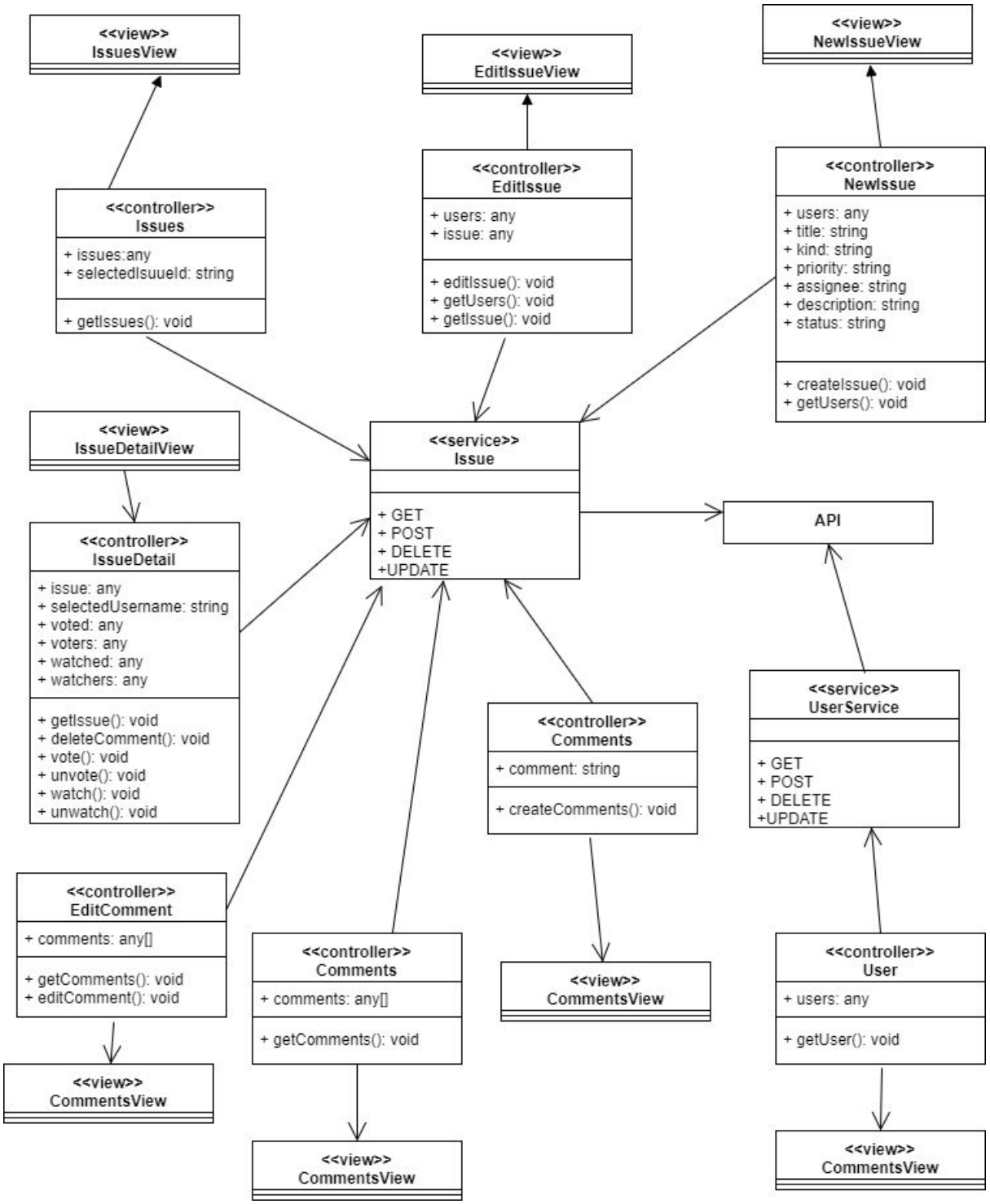
Després tenim el issues, encarregat de fer el get de totes les issues, mostrar-les i filtrar-les.

Issue-detail ens permetrà veura una issue mes datalladament i de forma mes clara, a part també ens permetrà comentar, votar i seguir la propia issue.

New-issue i edit-issue son dos components molt semblants, en el new issue sens mostra un formulari on podem crear una nova issue, mentre que al edit issue, sens mostra el mateix formulari amb la diferencia que ja ve ple.

Comment conté tota la lógica per mostrar els comentaris. Mentre que new-comment i new-comment, com en la part d'issue, fan pràcticament el mateix, en el primer ens mostra un formulari buit per crear un nou comentari, mentre que el segon ens mostra el mateix formulari amb el camp ple, que ens permet editar-lo.

En la següent pàgina podem observar el diagrama de clases:


<<view>>
IssuesView
<<view>>
EditIssueView
<<view>>
NewIssueView
<<controller>>
Issues
+ issues:any
+ selectedIsuueId: string
+ getIssues(): void
<<controller>>
EditIssue
+ users: any
+ issue: any
+ editIssue(): void
+ getUsers(): void
+ getIssue(): void
<<controller>>
NewIssue
+ users: any
+ title: string
+ kind: string
+ priority: string
+ assignee: string
+ description: string
+ status: string
+ createIssue(): void
+ getUsers(): void
<<view>>
IssueDetailView
<<service>>
Issue
+ GET
+ POST
+ DELETE
+UPDATE
API
<<controller>>
IssueDetail
+ issue: any
+ selectedUsername: string
+ voted: any
+ voters: any
+ watched: any
+ watchers: any
+ getIssue(): void
+ deleteComment(): void
+ vote(): void
+ unvote(): void
+ watch(): void
+ unwatch(): void
<<controller>>
Comments
+ comment: string
+ createComments(): void
<<service>>
UserService
+ GET
+ POST
+ DELETE
+UPDATE
<<controller>>
EditComment
+ comments: any[]
+ getComments(): void
+ editComment(): void
<<controller>>
Comments
+ comments: any[]
+ getComments(): void
<<view>>
CommentsView
<<controller>>
User
+ users: any
+ getUser(): void
<<view>>
CommentsView
<<view>>
CommentsView
<<view>>
CommentsView

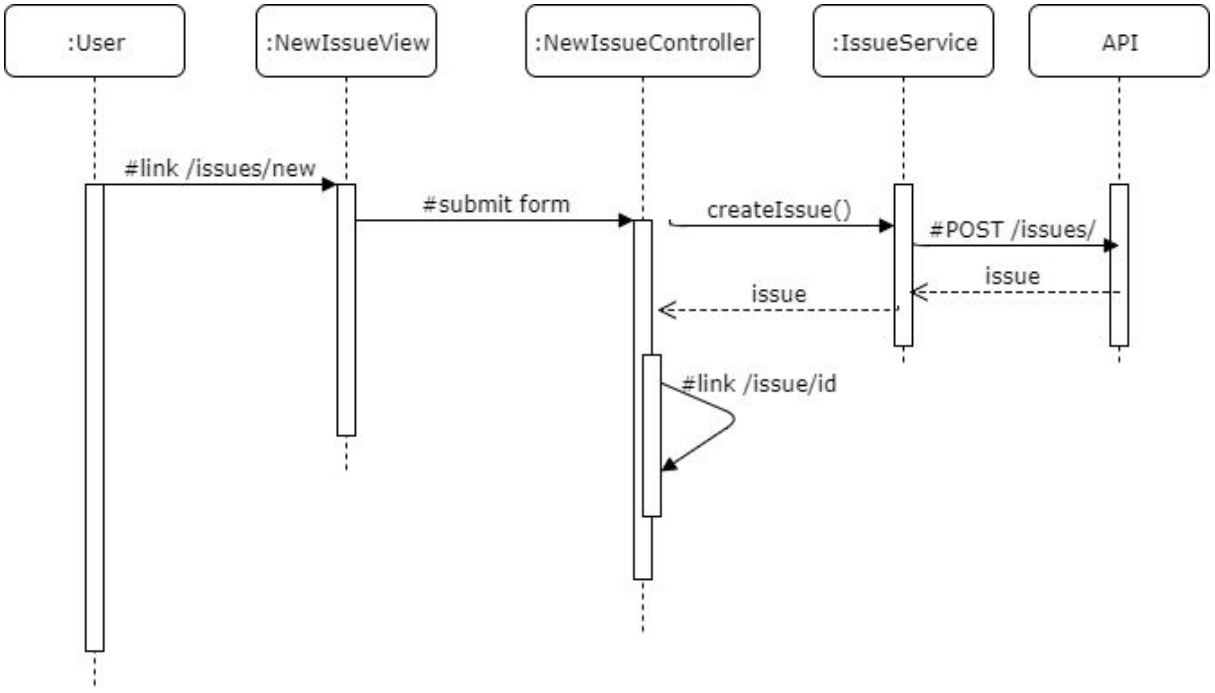
Processament d'una petició

Per processar qualsevol petició, es comença sempre amb una acció d'un usuari, ja sigui al clicar un botó o al accedir a una direcció en concret.

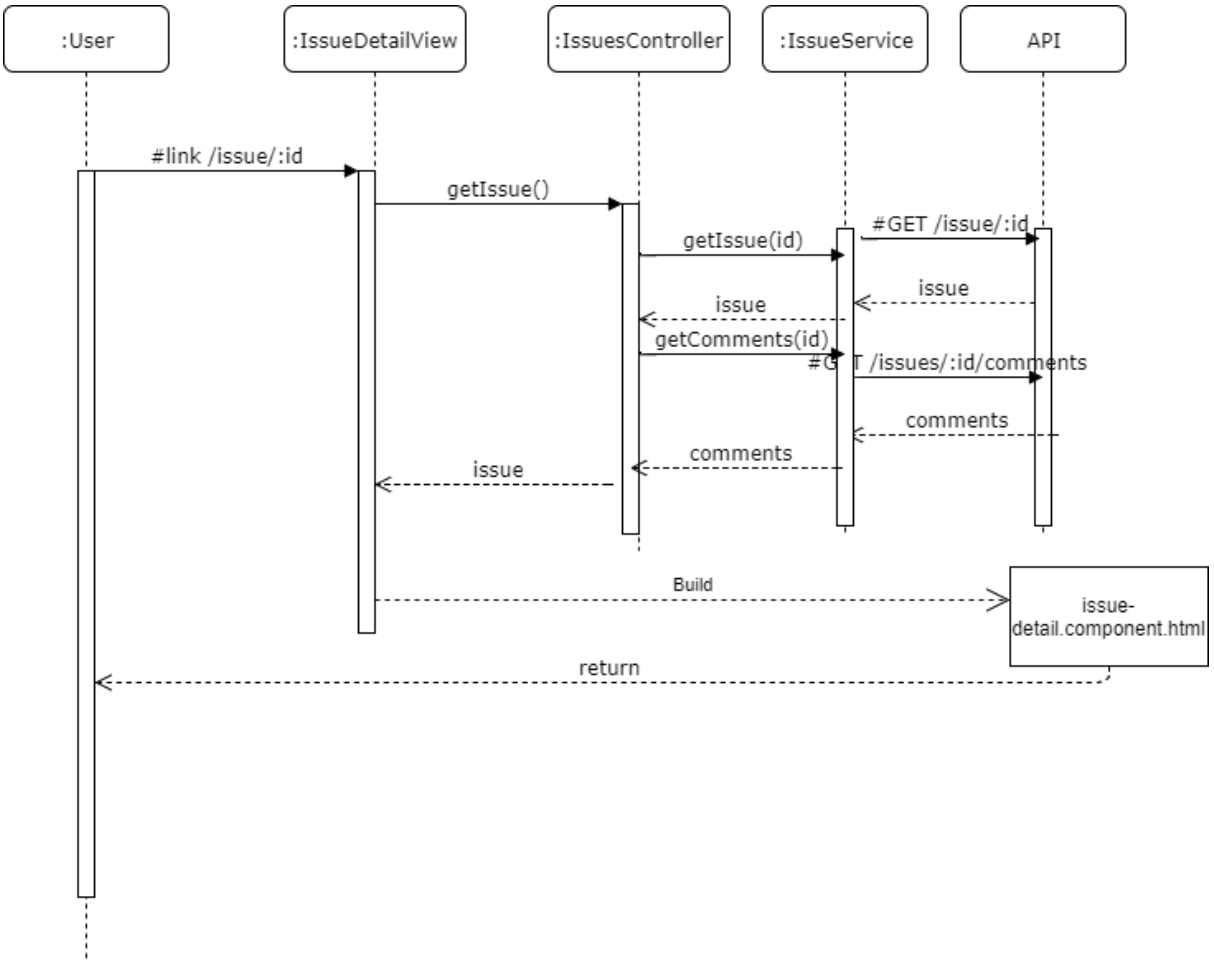
El nostre component consta de dues parts, la part visible, que es on l'usuari interacciona, i la part de darrera, que es on es computa la part lògica.

Al accedir a qualsevol part visible que necessiti d'una petició, la part visible demana mostrar un contingut, el component, per poder-ho fer, fa una crida a l'APi, a traves d'un servei, i quan aquesta li respon, l'infla el contigut.

Aqui tenim un exemple de creació de nova issue:



Com veiem en la imatge, a traves d'un formulari omplim els camps necessaris per la issue, despres el backend s'encarrega de gestionar el POST. AL finalitzar aquest es torna al controlador on fem un redirect cap a la issue acabada de crear:



En aquesta segona imatge podem observar com es procesa un get d'una issue en concret, primer es va a buscar a la APi el get de la issue, per després demanar els commentaris.