Landrup-dans Documentation

Tech Stack

React - React er et komponent baseret ”framework” der tillader at kun dele af den virtuelle DOM bliver re-renderet.

React samt mange andre frameworks tilbyder indbygget hooks, som erstatning på mange vanilla javascript funktioner, en af de populære, useState bruges til at gemme data og opdatere/re-renderer knyttede komponenter. Som en af de eneste frameworks tilbyder React en udvidet udgave af states, useContext. useContext er designet til at dele data på tværs af komponenter, og på den måde undgå prop drilling.
Så derfor har jeg valgt at bruge react. Det. og så er det super udviklere venlig at arbejde med.

React-router - React router gør det muligt at give brugeren samme oplevelse som vi er vant til som f. eks med Nextjs sider og helt plain HTML sider, Dog under hjelmen rendere den jsx og viser brugeren tilsvarende indhold.
(“Det kaldes også en single page application”)

Tailwindcss - Et css framework med indbygget utilities, der kan kombineres og med få utilities danne den ønskede styling. Tailwind har også mulighed for at tilpasse en config fil der erstatter deres utilities med udvikleres egne brugerdefineret utilities. det gør det super nemt at kombinere reusable components og tailwind for at genbrug dele af en app.

Axios - der er ikke så meget at sige, det super nemt og hurtigt. forkorter syntaxen og konvertere selv dataen til JSON osv.

Framer-motion - Framer-motion et bibliotek der er bygget specielt til react, der gør det super nemt og effektivt at animere elementer i ens app, Framer-motionen ligner meget css animation med properties og property values, der gør det muligt at lave en flydende overgang og forståelse af biblioteket.

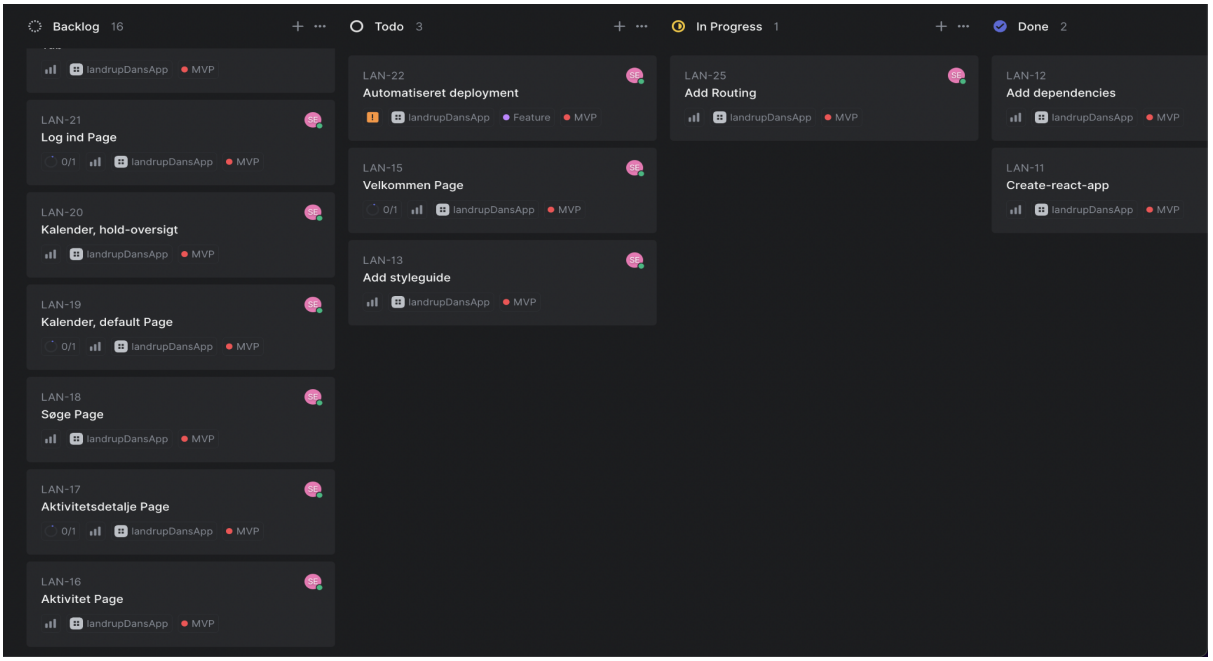
Yup - Et objekt baseret skema validerings pakke der stiller data op mod hinanden, og printer nogle fine beskeder. Yup er nemlig bygget på en måde der gør det muligt at bruge et skema flere gange på tværs af ens app. “Super smart ha?”

React-Cookie - Gør det muligt at gemme data der læses af serveren, det gør det muligt at gemme små mængder af data. Cookies kan tage brug af httponly, det betyder at man kan forhindre client-side scripts i at læse en cookie, så kun serveren kan tage brug af en cookie.
(“Dog tager jeg ikke brug af denne feature da det ikke er serveren jeg arbejdede med, men super smart nonetheless”)

Dag for Dag fremskridt

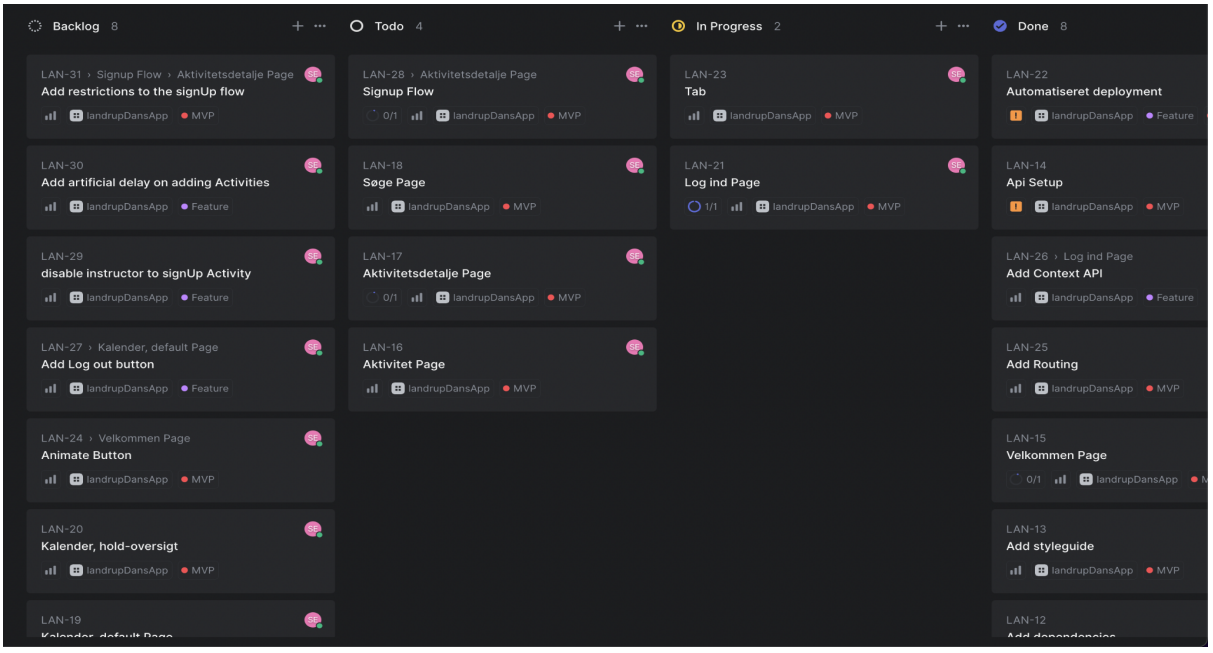
Dag 1

Første dag gik på at sætte Github, Linear og Vercel op. En style guide med Tailwinds config fil, og starten på forsiden.



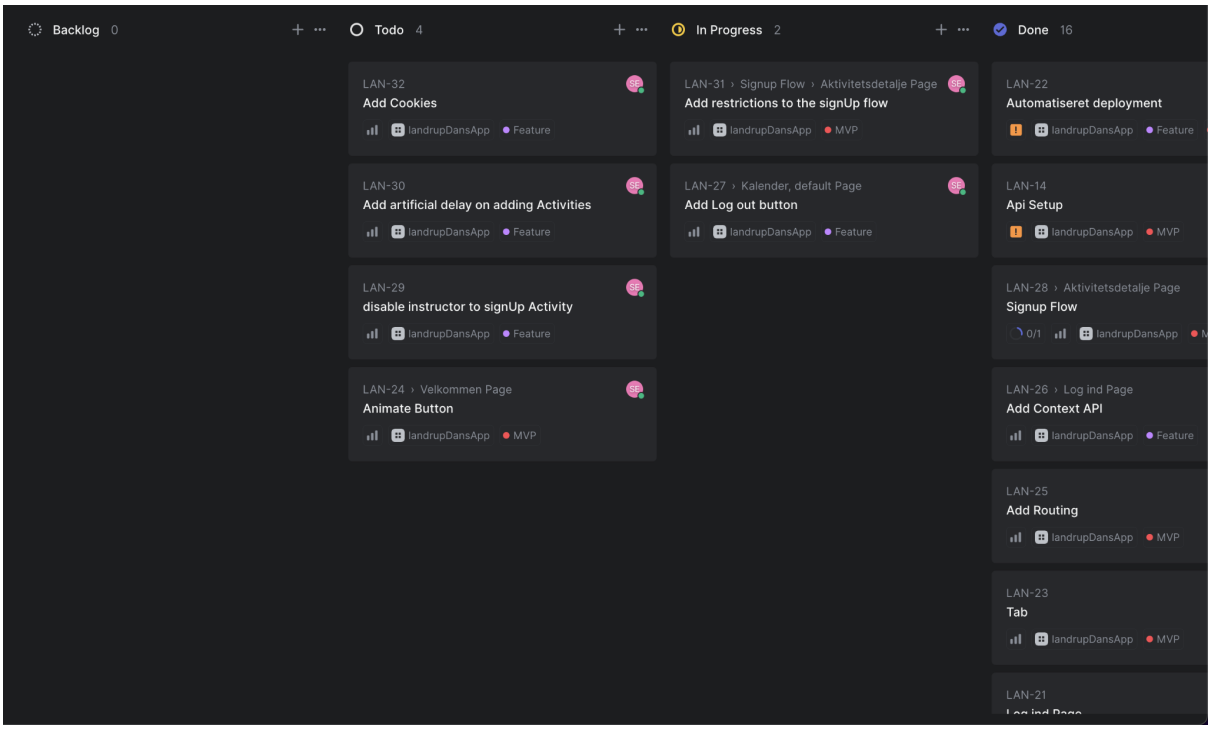
Dag 2

Dag 2 var en super produktiv dag hvor mange af mine issues kunne flyttes til Done. Alle min Routes/sider blev færdige og manglede kun småting.

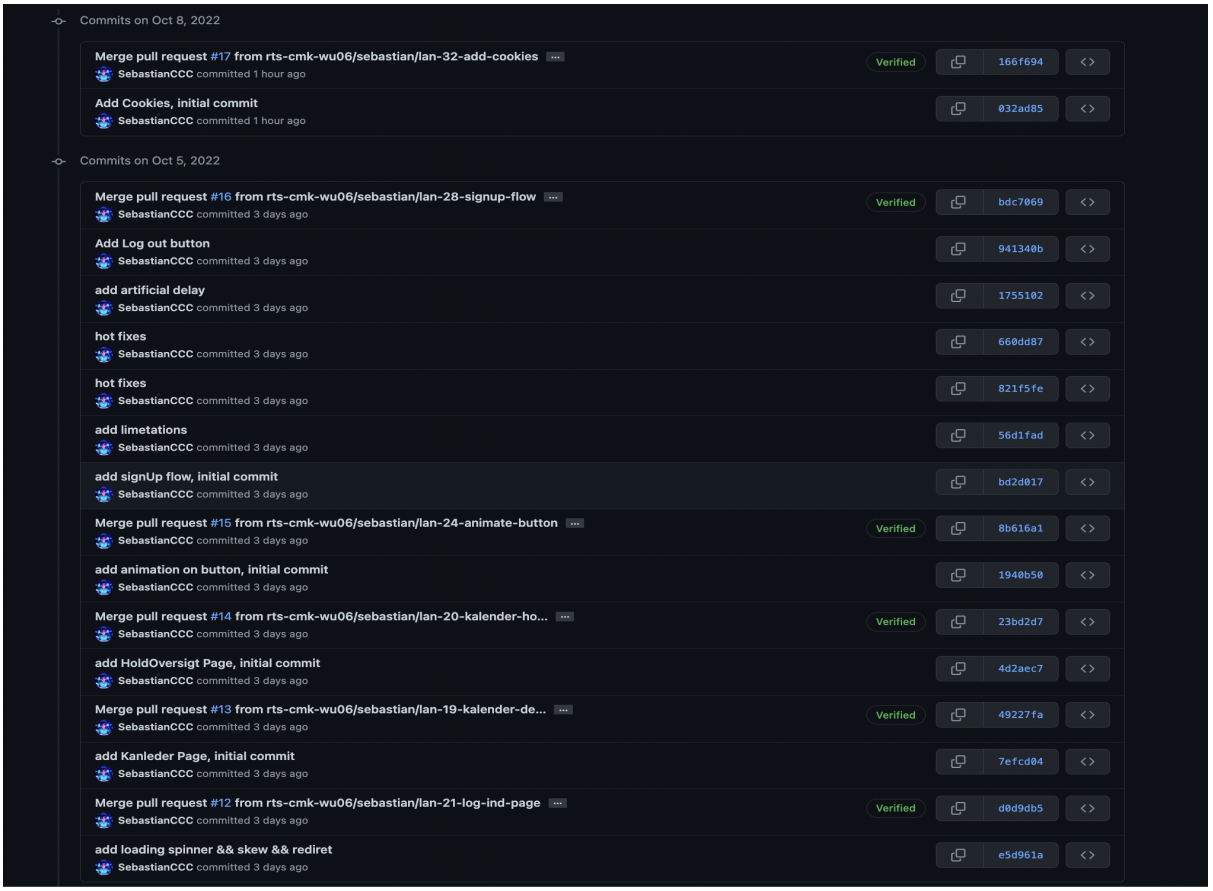


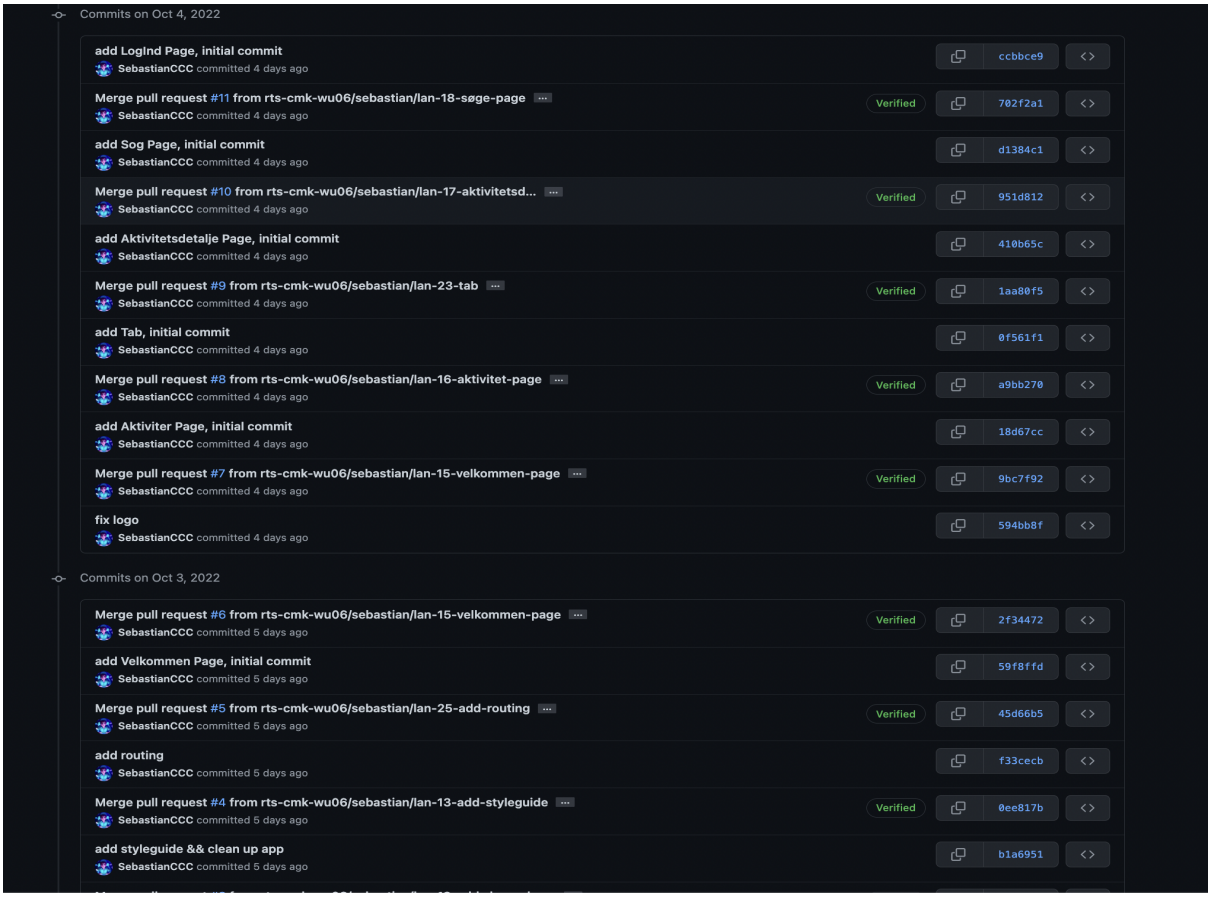
Dag 3

En stille dag hvor de sidste ting blev færdiggjort. Og finpudset.



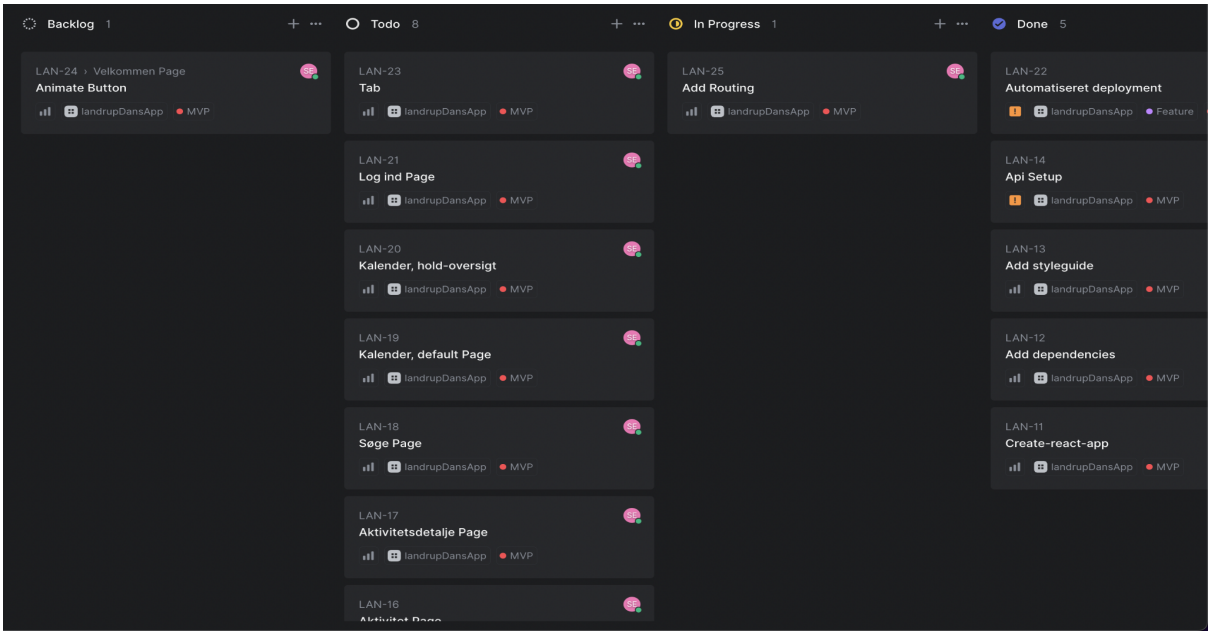
Commit History





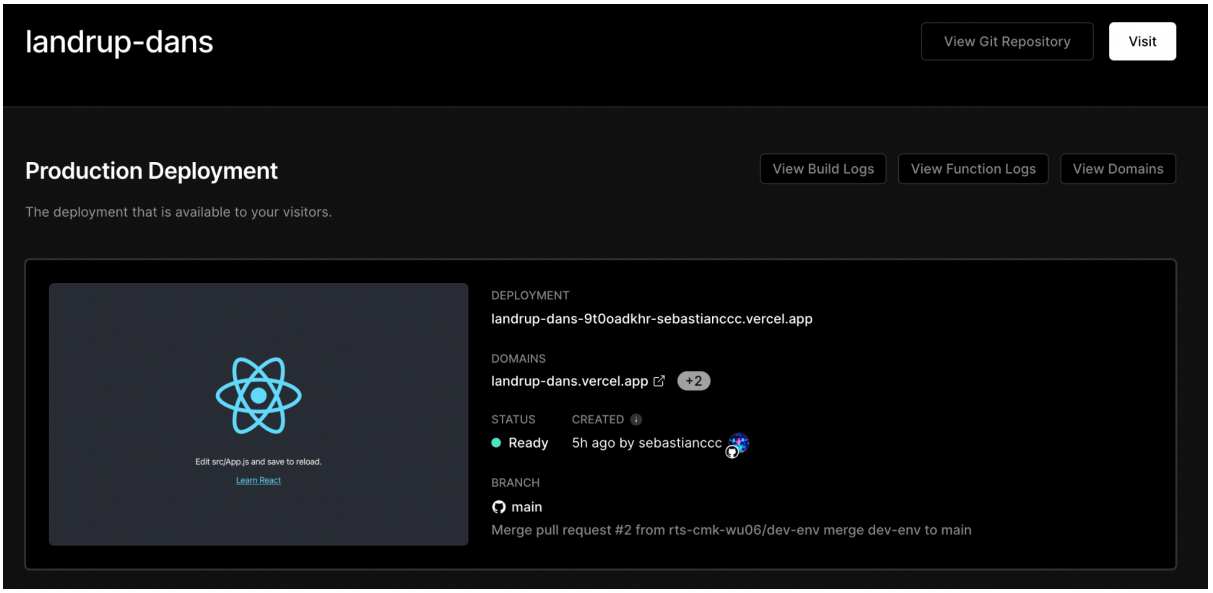
Kanban

For at holde styr på min opgave bruger jeg Linear, en platform der minder meget om Trello. Det smarte med Linear er muligheden for at knytte andre platforme sammen med Linear som f. eks Github, Figma og Slack, Og på den måde rykke på issues automatisk.



Continuous deployment

Som Valgfri opgave valgte jeg Continuous deployment. Som platform bruger jeg Vercel som er tilknyttede mit Github repo.



Cookies

Har tilføjet cookies til min app, som består af at gemme user objektet som en cookie. Det tilladere at brugere kan komme tilbage til app'en og stadig være logget ind.

```
3     .then((response) => response.json())
2     .then((data) => {
1       setLoaded(false)
41      setCookie('user', data)
1       navigate(-1, { replace: true })
2     })
```

Versionsstyring

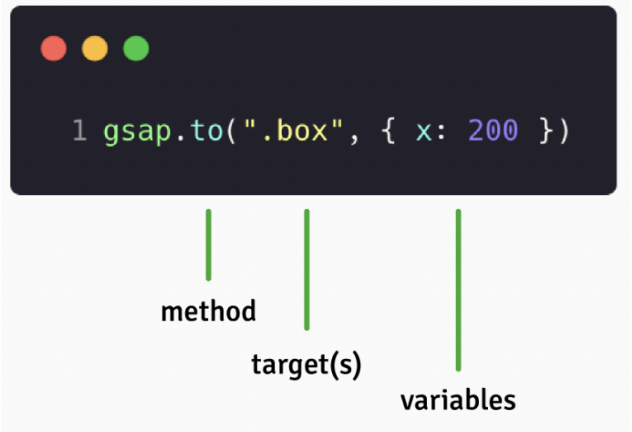
Som Versionsstyring bruger jeg Git. Git er nok den mest populære derude da den bliver brugt af Github, og gør det muligt at holde styr på ændringere i ens kode. Det smarte med Git er **(“Som jeg kunne forstå, som en af de eneste Versionsstyring”)** er muligheden for branches og kunne arbejde på flere ting med andre uden det hele ender i en merge conflict, hehe.

Perspektivering af Tech-Stack

React vs - Når det på et tidspunkt bliver relevant er et nyt framework opstået. Qwik, et front end framework uden hydration og mulighed for complete lazy loading. React vs Qwik består af Qwik's mulighed for dele din javascript filer op i små bider, det tillader at en udviklere ikke længere skal tænke på performance men derimod det der nu engang gør en app lækker at navigere.

Framer-motion vs - Et alternativ til Framer-motion, som også står stærk er GSAP, et javascript animations bibliotek. De minder utrolig meget om hinanden og synes derfor det kommer and på personlig præference og hvad man skal bruge det til. Dog læner GSAP til Vanilla Javascript use, som gør det muligt at bruge uanset Javascript framework.

"Derfor synes jeg det var perfekt at sammenligne lige præcis de to biblioteker"



Perspektivering

Hvis jeg skulle skalere min app, vil jeg som starters lave en plan over min filstruktur og dele opgaver op. Jeg vil på nuværende tidspunkt ikke ændre noget da mit udgangspunkt består af dette. Hvis jeg skulle ændre noget, ville jeg overveje NextJs som framework, som kan levere static pages relativt hurtigt. **("men er ikke helt sikker på dette, bare en tanke")** Hvad jeg er sikker på er React Query som addition til fetch requests, og så er det er super nemt at bruge. Og så noget som pagination er en lej.