Remix report

# Mit projekt

Jeg har valgt at lave en tidligere opgave kaldet ”Movibes” igen med ved brug af Remix denne gang. Grunden til jeg valge at bruge Remix istedet for nogen andre frameworks er fordi at vi har tidligere arbejded med React Router, som er udtrukket fra Remix. Derfor fik jeg en interresse for frameworked.

Movibes er en hjemmeside der minder lidt om Netlfix, da det er en film side hvor ideen er at man kan se film på. Der er 5 pages eller routes i alt på webpagen. Tre pages hvor man kan se de forskellige film, en log in page and en detail page. Detail pagen kommer jeg nærmere ind på i næste afsnit.   
Log in pagen er en statisk side, da jeg desvære ikke havde tid til at validere formen, p¨grund af jeg fokuserede mere på at fecthe dataen på de andre pages.   
De tre film pages er der forsiden, hvor man kan se et banner, en statisk sorter som man ville kunne sotere hvilke genre eller film selvskaber man vil se film fra og tre lister af forskellige film. Trending, upcoming og tv series.

# Dynamiske Routes i Remix

Når man skal lave et dynamisk Route i Remix, så er fil navngivningen det man skal have fokus på. Normale Routes i Remix kalder man bare hvad routet skal hedde, men når man skal lave et dynamisk route der er unikt med et ID, så skal man kalde filen noget specielt.   
Man skal kalde det for hvad routet skal hedde med ”$id.jsx” efter. I mit tilfælde kaldte jeg routet for ”details.$id.jsx” da det skal vise en detalje side af den påkaldte film, man har trykket på.

Hvis man skal fetche den individuelle film inde på sin ”$id.jsx” fil, så skal man have params med i sit fetch, som der står nedenstående;

export const loader = async({params}) => {

  let result = await fetch(`https://api.themoviedb.org/3/movie/${params.id}?api\_key=${APIKey}&language=en-US`)

  return result.json();

}

Derefter skal man bruge en useLoaderData til at fange sin data og udskrive det;

  let movies = useLoaderData();

  console.log(movies)

Når man har bade sit fetch med paramse og en useLoaderData, så kan man begynde at fetche de forsklellige elementer fra sit API og uskrive den på sin side.

# Hvordan er Remix bedre end React Router

Remix er bedre en react router på flere måder. React Router er et snippet taget ud fra Remix, så de to minder meget om hinanden, dog er der stadigvæk tor forskel for hvordan man laver sine routes og hvordan men eksempelvis fetcher noget fra et API.

I Remix har man en ”routes” mappe hvor ens index fil lægger i. Det er også denne mappe man skal lave sine andre pages i, da det er ”routes” mappen der connecter alle routesne sammen, så man kan linke dem til hinanden.   
Det er væsentligt nemmere at lave sine routes på denne måde, end at lave en fil hvor man skal skrive og impotere alle sine routes i.

Se, når man skal fetche i Remix er det en hel anden sag end i normal React. I normal React bruger men for eksempel en useEffect til at fetche fra sit API, men i Remix bruger man noget der hedder ”useLoaderData”. (insert mere her efter du har læst noget mere omkring useLoaderData)

# Ulemper og fordele med Remix

En stor ulempe i mine øjne omkring remix er at man ikke kan fetche i componenter. Normalt når man laver en React projekt, så har man en ”components” mappe, hvor man har de fleste til alle sine fetches. Det kan man desvære ikke i Remix. Jeg har prøvet at læse mig frem til problemet, men kunne desvære ikke finde nogen grund til hvorfor det er sådan. Jeg kunne heller ikke finde nogen alternative måder, istedet for components.   
Det vi sige at alle ens fetches skal være i den samme jsx fil. Det er måske muligt at gøre, men jeg kunne intet dokumentation finde på det.

En fordel ved Remix er dog at det har et godt sammerbejde med Tailwind CSS. Det vil sige at Remix fortrækker at man styler med tailwind og derfor er blevet lavet med tailwind i tankerne. Der er dog nogle steps man skal gå igennem for at instillere og implementere tailwind i sit Remix projekt.

# Remix vs Next

Remix minder lidt om Next, dog er de væsentlige forskellige. Remix har for eksempel hurtigere server rendering end Next har og Remix’es routing er nemmere og mere ligetil at forstå. Når man sætter Remix og Next op imod hinanden i en test for at se hvilket framework har hurtigere server site rendiering så er Remix 0.5 sekunder hurtigere end Next, hvilket gør en stor forskel for brugeren.

# Min personlige præfrerence

Min personlige præfrerence i forhold til at bruge Remix eller React er nok at bruge React. React giver mere mening i mit hovedet og min viden om det er meget større end Remix. Ja det er sandt at hvis jeg havde mere tid til at dykke ned i Remix, så ville jeg nok have bedre forstårelse for det, dog er jeg et vane dyr og bryder mig mest om det jeg er vandt til.

Der er meget jeg stadigvæk ikke forstår omring Remix, da jeg kun har haft to uger at dykke ned i det. Det har været sjovt at prøve noget nyt og lære et andet kodnings sprog, dog glæder jeg mig til at gå tilbage og have normalt React.