

Eksamens dokument

Gruppe 39

Github lenke:

https://github.com/CiprianAV/UIN_eksamensoppgave1

Vanskelighets grad valg:

Første diskusjoner i gruppa etter vi gikk eksamens tekst gjennom sammen var at vi sikter på Grad C først, men hvis vi kommer til det fort nok så har vi kanskje nok ferdigheter å prøve på B. Men etter at vi møtte noen utfordringer ganske tidlig i oppgaven allerede så fant vi ut at C er mer sannsynlig for oss å prøve å treffe.

Vi klarte det nesten, men noen problemer og vanskelige error meldinger i inspurten gjorde det sånn at vi treffer D kravene.

Github Alias:

CiprianAV (Ciprian-Andrei Vlad)

Marsojuuh (Marko Juhola)

Latifi010 (Aman Latif)

Redegjørrelse til eksamensoppgave:

Dette prosjektet er satt opp med Create React App (CRA)

Utfordringer:

API:

Api set up som generelt var en stor utfordring. Der var det mye kranling med det i starten. I første utgangspunkt så satt jeg api-key inn i en env.fil(kilde link nederst i dette dokument), jeg fulgte den metoden videre og fikk den til å funke på min side, men når hele gruppa startet å teste det så kommet det opp problemer. Da fant jeg ut fort at pga

dette .env filen så ble den skjult fra de andre i gruppa og ikke skjønnte jeg da at den skal da skape problemer til sensorene for å komme inn i appen. Så da måtte den endres.

Da ble den endret til at jeg lagret den inn i [config.js](#) fil som jeg da commit til GIT, da fikk alle tilgang til det og alle kunne få siden opp uten error meldinger. Link til inspo fant til dette finnes i kildeliste nederst.

Vi brukte en hook useTicketmaster.js til å satt opp TMI_API_KEY inn i den med BASE_URL til ticketmaster. Link til kilder i bruk av Hooks finnes i kildeliste nederst.

CategoryPage.js:

Hentet ut ifra Ticketmaster sin API-dokumentasjon, deretter bygget siden med tre seksjoner.

1. Arrangementer
2. Attraksjoner
3. Spillesteder.

Data:

Her har vi brukt useEffect for å hente ut alle tre samtidig, dermed lagret resultatene i tre ulike states.

```
const [events, setEvents] = useState([])
```

```
const [artists, setArtists] = useState([])
```

```
const [venues, setVenues] = useState([])
```

Så brukt fetch mot tre ulike endepunkter.

```
/events
```

```
/attractions
```

```
/venues
```

Her bruker alle kallene den oversatte *slug* verdien, slik at riktig innhold knyttet til hver valgt kategori vises.

Utfordringer:

- Spillesteder: Her vises det ingen bilde.

Kilder:

[How to Handle Multiple Fetch Requests With JavaScripts Promise.all](#)
[How to return the Promise.all fetch api json data? - Stack Overflow](#)
[Optional chaining \(?.\) - JavaScript | MDN](#)

Filterfunksjon:

Utfordringer:

- net::ERR_FAILED 400 (Bad Request)
- 429 Too Many Requests

Under utviklingen av filterfunksjon, støtte vi på et gjengående problem.

Etter å ha testet ut forskjellige fremgangsmåter, fant vi en mulig årsak til denne feilmeldingen.

Vi prøvde først å sende kun startdatetime, også i kombinasjon med enddatetime for å definere et tidsvindu. Begge tilnærmingene førte til 429 (too many request) feilmelding.

For å redusere sannsynligheten for denne feilen, implementerte vi inn `setTimeout` mellom hver av de tre fetch-kallene. Slik at vi kan være helt sikre på at vi ikke overstiger API-et sitt `rate limit` som er på 5 forespørsler per sekund.

Som en siste utvei, fjernet vi muligheten for å filtrere på dato helt. Noe som førte til at vi kunne filtrere på land og by, uten noen feilmeldinger.

Til tross for at vi har forsøkt å begrense antall forespørsler ved å bruke setTimeout og redusert antall kall, dukker likevel denne feilmeldingen opp noen ganger – når man navigerer raskt mellom sidene. Den forsvinner som regel ved å oppdatere siden.

(mulig) Løsning:

Isteden for å kjøre API-kallet etter hvergang en bruker endrer dato, by eller land, kunne vi brukt en enkel Søk knapp for å hente data etter at alle valgene er gjort. Dette ville nok ha redusert antall forespørsler betraktelig.

Søkefunksjon:

En bruker kan skrive inn et søkeord, dermed trykke på søk knappen. Når denne knappen trykkes, vil variabelen searchClick oppdatere seg.

Denne variabelen brukes i useEffect, slik at vi manuelt kan kjøre et nytt API kall. På denne måten vil vi også unngå (429 error) – Siden dette hindrer et nytt API kall for hvert input i søkefeltet.


[Date.prototype.toISOString\(\) - JavaScript | MDN](#)

[Given a DateTime object, how do I get an ISO 8601 date in string format? - Stack Overflow](#)

[How To Create Custom Select Menus](#)

<https://www.developersmonk.com/post/resolve-429-error>

[Combination of async function + await + setTimeout - Stack Overflow](#)

 [Make a Search Bar with React \(with API Calls\) | Beginners Tutorial](#)

Dashboard.js:

Her bruker vi en simpel useState variabel (loggedin), for å se om brukeren er logget inn. Er loggedin satt til false, vil det vises et innlogging skjema.

Etter at dette skjema blir sendt inn, vil onSubmit sørge for at siden ikke blir oppdatert og dermed setter loggedin til true.

Slik at vi “Min side” da vil vises.

Kilder:

- <https://stackoverflow.com/questions/71536244/check-username-password-login-form-using-react-hooks>

Kilder:

- Brukt dette source til å sette opp API key trygt i React appen, bestemt å bruke .env file for å "store" api key til og så satt den i .gitignore filen.
<https://medium.com/@oadaramola/a-pitfall-i-almost-fell-into-d1d3461b2fb8>
- CORS error
<https://stackoverflow.com/questions/67320806/getting-cors-error-rather-than-expected-429-response-when-rate-limit-is-reached>
- API key inn i en config.js fil.
<https://stackoverflow.com/questions/51601903/hiding-api-keys-in-js-setup>
- Hooks kilder:
https://www.w3schools.com/react/react_hooks.asp
<https://medium.com/@hafidkrntn/based-practice-api-call-with-react-hooks-d2b2a99b84b4>
- Ticketmaster develop, for å hente inn info om eventene:
<https://developer.ticketmaster.com/api-explorer/v2/>