|  |
| --- |
| Cecilie Seim, Stine Losnedahl, Julian Alsemmani |
| EKSAMEN UIN Vår 2023 |
| GRUPPE 5 |

|  |
| --- |
|  |

Innhold

[GitHub 2](#_Toc135079881)

[Vanskelighetsgrad 2](#_Toc135079882)

[Våre forutsetninger 2](#_Toc135079883)

[API-id vs slug 2](#_Toc135079884)

[Sanity Studio vs Sanity cli 2](#_Toc135079885)

[Schema: 3](#_Toc135079886)

[GROQ 3](#_Toc135079887)

[Skalering 4](#_Toc135079888)

[Tilgjengelighet & Responsivitet 4](#_Toc135079889)

[Utfordringer 5](#_Toc135079890)

[Oppgaver vi ikke prioriterte 5](#_Toc135079891)

[Kilder 6](#_Toc135079892)

## GitHub

@Julianalsemmani - **Julian Alsemmani**  
@tilen1976 – **Cecilie Heim**  
@slosne – **Stine Losnedahl**

## Vanskelighetsgrad

Vi har valgt å ta utgangspunkt i kravene for karakter A.

## Våre forutsetninger / utgangspunkt

### Branches

Vi har valgt å jobbe med GitHub issues, hvor vi lager en branch for hver issue vi tar til oss.   
Vi lager så PR på hver issue, godkjenner for hverandre og merger inn i main kontinuerlig.   
  
På den måten har vi alltid kjørbar kode i main, som oppdateres ofte. Vi får også flere jevnlige commits, som gir bedre oversikt, dersom man har nødt til å gå tilbake å se hva som er gjort i ettertid.   
I tillegg er det lettere å lære av hverandre, da vi må se over hverandres arbeid og eventuelt kommentere om man har tips til bedre løsninger eller forbedringer før merge.

### API-id vs slug

I oppgavens B-krav står det at: «På en side for ett spill, bruk feltet API-id for å hente ekstra informasjon (i tillegg til informasjon som ligger i Sanity) som skal vises om spillet.»

Her har vi tolket oppgavens krav slik:

Ved å bruke slug som lenke binder den sammen GameCard og GamePage, der GamePage kan hente slug med useParams. Fordelen med en slug i motsetning til en id med tall er at den er logisk og lesbar.

Det er ikke nødvendig å lese inn data fra Sanity i GamePage kun for å matche api-id. All informasjon vi trenger på siden er tilgjengelig med en spørring opp mot rawg api, og hos RAWG er slug likestilt med id.

Slug er alltid unik både hos Sanity og RAWG. Vi har dessuten skrevet det meste av dataene direkte fra RAWG til Sanity ved bruk av Sanity.cli, slik at det er garantert at slug er lik.

Men selv om både Sanity og rawg krever helt unike slugs, er det en risiko hvis man slugify’er url i Sanity studio med en regex som ikke dekker hele spekteret av muligheter.

For eksempel er det en del paranteser rundt årstall i rawg api, som med en standard slugify fra undervisning ikke ble fjernet ved testing.

### Sanity Studio vs Sanity cli

I Sanity studio kan man kun jobbe med et dokument om gangen og det egner seg lite til større jobber. Det er også en risiko for skrivefeil. Nå kan man kanskje ikke påstå at dette datasettet er det største, men å importere dataset med Sanity cli har sine fordeler og noen utfordringer.

Utover det å beherske kjøring i node, sette opp scripts eller bruke client i backend og transformere json til ndjson, så er nok den største utfordringen å tilpasse dataene til Schema. Det finnes mye hjelp å få i Sanity dokumentasjonen, men noe er likevel vanskelig. Særlig å skrive data til en «reference».

Vi endte derfor opp på en mellomløsning: Det meste skrives inn, men noe fikses i Sanity Studio i etterkant. Det hele har foregått slik:

* Et «script» i filen «fetchGame.js»
* Sanity.cli.ts satt opp med «writeClient» og token
* Terminal i Bash mode
* Node.exe fetchGame.js > filename (resultatet er i mappen ndjson\_files i mappen Sanity)
* Sanity dataset import filename.json production

Dataene som er lest inn er større enn behovet vårt, men en liste med bilde-url’er tar ikke stor plass, selv om vi i denne omgangen kun benyttet et bilde. Og den gjennomsnittelige playtime fra rawg kunne vært brukt til en visualisering i forhold til hvordan bruker ligger an med sin egen spilling: en progressbar eller lignende.

### Schema:

* genre – liste over sjangere hentet fra rawg api – title, apiId, slug
* game – title, apiId, avgPlaytime, array med bilde-url og en array med references til genre
* userGame – Object med brukerens spilletid og en reference til et spill (game)
* user – username, e-mail, array med favoritter som er references til game og userGames som er en array med userGames

Bortsett fra arbeidet som har blitt gjort med ‘user’ mangler det i de andre Schema bruk av regler og vurdering av hvilke felter som er «required». Dette er viktige ting som bør være med i prosjekter som skal deployes. Per nå kan man for eksempel skrive minustimer i spilletid, eller flere steder legge inn samme spill på nytt i samme liste o.l. Dette er vi klar over.

Bruken av «references» i et så lite datasett kan diskuteres. På den ene siden er det nyttig for å unngå redundans, men det medfører også en del overhead og etter hvert ganske komplekse nøstede spørringer. Det andre er bruken av «strong/weak» references. Hvis man ikke setter dette, er referansen «strong». Det kan medføre noen utfordringer ved skalering i en applikasjon med lenger levetid enn eksamensperioden som man bør ta stilling til, men som det ikke var plass til i denne omgangen.

### GROQ

Å sette opp gode og nyttige GROQ-spørringer kan være både effektiviserende og brukervennlig.

Med projections kan man for eksempel regulere hvilken data man henter inn etter behov. Dette er riktignok mer komplisert å sette opp når det er tre lag å nøste seg gjennom. Navngivning i projections kan hjelpe til å flate ut resultatet, men igjen med mange lag kan det være vanskelig. Og ha mange spørringer medfører flere kall til Sanity og det kan være mindre effektivt.

Underveis i prosessen har vi måtte endre spørringene en del både i forhold til opprettelse av «user», endringer i «user» og forsøk på å forme projections. Å sette count inn i fetch av games og favoritter sparer også kode i frontend.

Å beholde fetch av userGames og favoritter som to ulike kall var et bevisst valg da de i tillegg til at dataene skal ut i ulike komponenter, kan endres uavhengig av hverandre.

## Skalering

Vi har tenkt skalering i utforming av vår nettside. Dersom dette var et prosjekt som skulle lanseres ut i markedet, hvordan ville dette gått og har vi lagt noen låser for oss selv? Dette har vi justert i form av f.eks. følgende:

* ??
* Vi har laget en egen fil kalt Fetch i «functions»-mappa. I den er det laget en dynamisk fetch, som du kan benytte i andre filer om ønskelig. Da er det bare å importere funksjonen og kalle den med de parameterne man behøver. Det er litt begrenset hvor mye bruk vi har for den nå, men for et stort prosjekt sparer vi tid og «blekk og kalorier» ved å ha separert denne i en egen fil.

## Tilgjengelighet & Responsivitet

Vi har sørget for at følgende er i hht krav for UU & responsivitet:

* **Kontraster** på farger passerer tester for UU
* **Lenker og buttons**, bortsett fra logo, er understreket eller har tydelig knappeform.
* **Beskrivende tekst** på lenker og knapper
* **Breadcrumb** for lettere navigering
* **Breakpoints** for stor, mellomstor og liten skjerm med mobile first approach
* **Overskriftsnivåer** er overholdt

Vedr **TagCloud** kan det diskuteres hvor tilgjengelig denne er som utgangspunkt, men man kan ikke behandle en «ordsky» som et dekorativt bilde, siden det faktisk inneholder nyttig informasjon. Siden vi bruker en ferdig komponent, får vi ikke analysert «ordskyen» opp mot skjermleser. Dette er riktignok noe man absolutt bør gjøre hvis man vil benytte slik funksjonalitet på nettsiden.

TagCloud kan produsere random farger, i tillegg til å skalere ordene etter antall i games\_count. Men det ferdige oppsettet oppnår ikke god nok fargekontrast i test med Lighthouse.

Hvis man har bedre tid kan man style TagCloud ned på detaljnivå, men basert på det som er tilgjengelig i komponentet, ble det beste valget å sette opp minstestørrelse på tekst i tillegg til å benytte oss av ‘colorOptions’ i tagCloud, som har to key/value par. Så lenge bakgrunnen er lys er det best å velge «dark» på luminosity og faktisk var det kun «blue» på hue som oppnådde kravet til farge-kontrast. Vi har derfor prioritert kontrast fremfor utseende på ordskyen.

## Utfordringer

## Oppgaver vi ikke prioriterte

* **Lav performance**  
  Bildene vi henter inn fra RAWG er veldig store. Dette gjør at vi scorer litt lavt på performance.

Vi har vurdert det hen at vi ikke har mulighet til å prioritere det, men vi er klar over at skalering av bilder ville gjort en forskjell der.

* **Lagring av favoritter**  
  Vi ønsket å få på plass muligheten for lagring av favoritter til Sanity, men ettersom det er en kombinasjon av RAWG-spill og spill fra Sanity ble dette litt for tidkrevende til å prioritere i denne omgang. Vi har derfor valgt å gå for en LocalStorage-variant, slik at du i hvert fall midlertidig kan se favorittene du legger til.
* **Tastaturnavigasjon**  
  Vi har ikke prioritert full tastaturnavigering som en del av UU. Vi er klar over at denne hopper over overskrifter og går rett til knapper. Med mer tid ville dette vært noe vi hadde prioritert.
* «Buy this game» ?

## Kilder

Gillen, M. (20 september, 2021). HOW TO: Create Accessible Word Clouds.  
<https://accessiblewebsiteservices.com/how-to-create-accessible-word-clouds/>   
  
Hdoro (April 2021). *Learn GROQ in 45 minutes.*  
<https://hdoro.dev/learn-groq>   
  
Lindquist, J. for Egghead.io (2022). *Introduction to GROQ Query Language.*  
<https://egghead.io/courses/introduction-to-groq-query-language-6e9c6fc0>   
  
Madox2 (2022). *Tag cloud for React.*   
<https://madox2.github.io/react-tagcloud/>   
  
Nick for StackOverflow (6 mai, 2020). *javascript new array without duplicates*  
<https://stackoverflow.com/questions/61625182/javascript-new-array-without-duplicates>   
  
RAWG (u.å.). *RAWG Video Games Database API (v1.0)*.  
<https://api.rawg.io/docs/>   
  
Sanity.io (7. desember, 2022). *Importing Data.*  
<https://www.sanity.io/docs/importing-data>   
  
Sanity.io (28. august, 2019). *Importing data from external sources.*  
<https://www.sanity.io/guides/guide-importing-data-from-external-sources>   
  
Sanity.io (u.å.). *Welcome to Sanity's Documentation*.  
<https://www.sanity.io/docs>   
  
W3Schools (u.å). *How TO – CSS Loader.*  
<https://www.w3schools.com/howto/howto_css_loader.asp>

W3Schools (u.å). *JavaScript Array some()*  
<https://www.w3schools.com/jsref/jsref_some.asp>   
  
Wzker for StackOverflow (3. mars, 2021). *Javascript Filter and Some on Array of Objects*  
<https://stackoverflow.com/questions/66455933/javascript-filter-and-some-on-array-of-objects>