Workshop arbeidsprosess dokumentasjon

Dato: 07.06.2023

Skrevet av

Mathias Krogh

Innhold

[Om dette heftet 2](#_Toc136959711)

[Valgte Moduler Begrunnelse 2](#_Toc136959712)

[AuthJs (Next-auth) 2](#_Toc136959713)

[TailwindCSS 2](#_Toc136959714)

[Axios 3](#_Toc136959715)

[Next-Connect 3](#_Toc136959716)

[Formidable 3](#_Toc136959717)

[React-DatePicker 3](#_Toc136959718)

[React-Icons 3](#_Toc136959719)

[React-Toastify 3](#_Toc136959720)

[SWR 3](#_Toc136959721)

[Zod 3](#_Toc136959722)

[HMS 4](#_Toc136959723)

[Tirsdag 30.05.2023 4](#_Toc136959724)

[Resultat av dagen 6](#_Toc136959725)

[Hjelpemidler og lenker brukt i dag 6](#_Toc136959726)

[Onsdag 31.05.2023 6](#_Toc136959727)

[Resultat av dagen 8](#_Toc136959728)

[Hjelpemidler og lenker brukt i dag 8](#_Toc136959729)

[Torsdag 1.06.2023 8](#_Toc136959730)

[Resultat av dagen 13](#_Toc136959731)

[Hjelpemidler og lenker brukt i dag 13](#_Toc136959732)

[Fredag 2.06.2023 14](#_Toc136959733)

[Resultat av dagen 25](#_Toc136959734)

[Hjelpemidler og lenker brukt i dag 25](#_Toc136959735)

[Lørdag 3.06.2023 26](#_Toc136959736)

[Resultat av dagen 30](#_Toc136959737)

[Hjelpemidler og lenker brukt i dag 30](#_Toc136959738)

[Søndag 4.06.2023 31](#_Toc136959739)

[Resultat av dagen 42](#_Toc136959740)

[Hjelpemidler og lenker brukt i dag 42](#_Toc136959741)

[Mandag 5.06.2023 43](#_Toc136959742)

[Resultat av dagen 46](#_Toc136959743)

[Hjelpemidler og lenker brukt i dag 46](#_Toc136959744)

[Tirsdag 6.06.2023 47](#_Toc136959745)

# Om dette heftet

I dette dokumentasjonsheftet skal jeg beskrive og vise arbeidsprosessen og resultatene av arbeider jeg utfører. Jeg har laget en tidsplan i prosjektplanen som viser lett hva jeg arbeider med fra dag til dag, mens dette heftet vil ha mer beskrivende resultat av dagene.

Dette heftet vil være også basert på Loggen jeg skriver på slutten av dagene for å dokumentere arbeidet jeg har fullført. Innholdet i dette heftet vil også vise tankegangen min, milepæler og endringer jeg har underveis igjennom arbeidsprosessen.

HMS og Modul-beskrivelse vil være inkludert i dette hefte.

# Valgte Moduler - Begrunnelse

I løsningen min så har jeg valgt å bruke følgende moduler:

## AuthJs (Next-auth)

Jeg har valg tå bruke AuthJs for innlogging fordi de har mange forskjellige innloggingsløsninger å velge mellom. Jeg valgte å bruke Google og GitHub for innlogging.

## TailwindCSS

Jeg valgte å bruke TailwindCSS fordi det er et rammeverk som er fantastisk å bruke i NextJs prosjekter. Ved opprettelse av et nytt NextJs prosjekt så får man en liten liste med funksjonsvalg som man kan legge til hvis man ønsker. Det er 2 andre pakker/moduler som installeres sammen med TailwindCSS som er: PostCSS og Autoprefixer.

## Axios

Jeg valgte å bruke Axios fordi den håndterer JSON data automatisk i response, har

## Next-Connect

Next-Connect er en «promise-based» metode for ruting. Jeg ble introdusert til Next-Connect fra min kollega istedenfor å bruke NextJs sin egen løsning for routing. og jeg har brukt dette mest når jeg oppretter API Routes i mine tidligere prosjekter. Dette har jeg brukt tidligere og fått mest opplæring i.

Next-Connect forenkler syntaxen for ruting. Dette gjør håndtering av rutingen lettere enn med standard NextJs Routes.

## Formidable

Dette er en modul jeg ikke er så godt kjent med, men jeg bruker denne modulen for å håndtere bilde-filer som lastet opp til databasen. Modulen gjør det lettere å håndtere filopplastning til API ‘et.

## React-DatePicker

På innleggingssiden av nye gjenstander ville jeg ha en knapp hvor brukeren kan velge en dato i stedet for å skrive datoen inn som tekst. Dokumentasjonen til modulen var god, så jeg valgte å bruke denne i min løsning.

## React-Icons

Jeg valgte å bruke denne populære modulen for å bruke ikoner på nettsiden i løsningen min. Denne modulen har støtte til 27 forskjellige ikon-biblioteker.

## React-Toastify

Toastify er en module som er dedikert for brukervennlige tilbakemeldinger som vises på skjermen. Dette har jeg valgt å bruke til å gi brukeren tilbakemelding etter at ulike hendelser har blitt utført.

## SWR

Dette er en ny React Hook som jeg nylig ble introduser til av min kollega. Denne React hook’en brukes til å hente data fra databasen og oppdatere dataen som vises på nettsiden i sanntid. Det er en hook som jeg personlig begynner å like godt og har lært meg å bruke. Derfor har jeg valgt å bruke denne for å hente data fra databasen.

## Zod

Zod er et bibliotek for TypeScript skjemavalidering. Jeg bruker Zod på server-siden for å validere ID’er som blir mottatt fra URL ’en. Zod passer på at ID’en er den typen som jeg definerer.

# HMS

I denne delen beskriver jeg HMS etter at arbeidet er utført.

* Jeg har brukt en komfortabel stol på jobb som jeg lånte fra min kollega, men byttet tilbake til min egen stol som også er behagelig.
* Jeg har hatt pauser ofte, men veldig lite pauser hvor jeg beveger meg rundt (utenom turer til kaffemaskinen)
* Luften inne har vært normal og dersom det var dårlig luft åpnet jeg bare vinduet for å få litt gjennomtrekk
* Jeg har valgt å jobbe overtid som har påvirket søvnen min, men kaffe redder dagen som vanlig

HMS Vurdering:

|  |  |  |
| --- | --- | --- |
| DÅRLIG | BRA | GOD |
|  | X |  |

# Tirsdag 30.05.2023

I dag har jeg mottatt oppgaven og utarbeidet en prosjektplan for arbeidet. I prosjektplanen skrev jeg hvilke verktøy og hjelpemidler jeg har tenkt å bruke i løsningen min. Verktøyene jeg har valgt å bruke:

* rammeverket NextJs.
* AuthJs (Autentisering),
* MongoDB,
* Prisma
* TailwindCSS (layout/design)

Jeg skrev også ned hvilke hjelpemidler jeg tenker å bruke igjennom arbeidsprosessen, som er kollegaer, bruk av internett, moduler/pakker og ChatGPT dersom det trengs.

Jeg sendte inn prosjektplanen da jeg ble ferdig og fikk den godkjent. Satte i gang med arbeidet med en gang og opprettet en BackLog hvor jeg dokumenterer kode og problemene jeg får underveis.

Senere på dagen har vi et lite møte hvor jeg spurte noen få spørsmål og fikk litt info om hvordan prøvenemnden vil bedømme arbeidet mitt.

På slutten av dagen laget jeg en visuell skisse av tabellene med bruk av nettsiden Figma som har verktøy for å visualisere databaser/modeller/tabeller. Etter at jeg lagde ferdig tabellene, fikk jeg et tips fra min kollega om å bruke eraser.io som er en AI som skaper visuelle skisser av prisma-kode.

Bilde av tabellene (laget manuelt i Figma)

Et bilde som inneholder tekst, skjermbilde, diagram, nummer

Automatisk generert beskrivelse

Bilde av tabellene (eraser.io AI)

Et bilde som inneholder tekst, diagram, Parallell

Automatisk generert beskrivelse

## Resultat av dagen

* Prosjektplan ble godkjent av prøvenemnda
* Nytt NextJS prosjekt er laget
* Ferdiglaget innlogging
* Opprettet MongoDB og laget ferdig tabellstruktur
* Dokumentering

## Hjelpemidler og lenker brukt i dag

* Fikk innspill fra kollega om at jeg nesten gjorde en design-feil i relasjonen mellom tabellene «items» og «image».
* Har brukt Figma for å lage en visuell struktur av database
* Fikk tips fra kollega om å bruke eraser.io til å skape visuell tabellstruktur av prisma.schema kode med bruk av AI

Lenke til eraser.io: https://www.eraser.io/

# Onsdag 31.05.2023

I dag har jeg opprettet routes for Items, Comments og Workshop. Siden jeg bare trenger en ID for å slette kommentarer eller verktøy, har jeg lagt delete routen i en egen fil innpakket med [] rundt seg. Dette vil da tillate meg i å skrive for eksempel «/api/comment/923892382» for å slette kommentaren som har id «923892382» i databasen. Jeg har ikke lagt til en slik fil for workshop fordi jeg ikke har noe behov for å slette workshop når jeg utvikler.

Bilde av mappestruktur (api)

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

På morgenen så la jeg merke til at jeg ikke hadde lagt til userId på «Items» tabellen og laget relasjon mellom dem. Jeg tenker at det kan være gunstig å identifisere hvilken bruker som oppretter nye verktøy og utstyr. Det må generelt ikke vises på nettsiden nå, men jeg vil ha muligheten til å videreutvikle en «logg» som lister opp hvem som sletter verktøy, legger til og oppdaterer innholdet.

Bilde av tabellene (Laget manuelt i Figma)

Et bilde som inneholder tekst, skjermbilde, diagram, nummer

Automatisk generert beskrivelse

Bilde av tabellene (eraser.io AI)

Et bilde som inneholder tekst, diagram, Parallell

Automatisk generert beskrivelse

Har ikke enda fått laget layout for fremvisning av data, så det gjør jeg i morgen etter at jeg har verifisert databasestruktur.

## Resultat av dagen

* Oppsett av routes
* Lagt inn testdata
* Testet routes
* Data er klar til å vises på nettside
* Oppdatert tabeller og skjema
* Dokumentering

## Hjelpemidler og lenker brukt i dag

* Har sett på tidligere kode for hvordan jeg satt opp SWR Hook og sett på dokumentasjonen til SWR

Link til SWR Dokumentasjon: https://swr.vercel.app/

# Torsdag 1.06.2023

I dag har jeg sett over databasestrukturen min og tenker det jeg har er bra nok for å utføre arbeidet. Jeg har derimot noen tanker for videreutvikling, som for eksempel et felt for å legge til beskrivelse på et verktøy og navn på verktøy/utstyr. Jeg tenker å muligens legge til navn og kanskje beskrivelse for et verktøy for å gi litt bedre beskrivelse for verktøy og utstyr.

Etter jeg var ferdig med å se over databasestrukturen, så satte jeg i gang med å sette opp paginering (pagination). Jeg hadde tanke om å bruke en pagination-hook fra Mantine fordi oppsettet for deres hook er lett å bruke, men jeg fikk tips fra kollega av å bare bruke SWR sin metode for paginering. Jeg valgte først å prøve ut hooken fra Mantine, men fikk problemer med reglene for React Hooks. Jeg valgte til slutt å droppe det og heller bruke SWR sin løsning.

SWR sin løsning var ikke så avansert fordi i eksemplet kan man se at logikken de går for er å sende sideantallet via query.

Bilde fra SWR sin paginering dokumentasjon:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

I Prisma er det mulig å definere hvor mye data som skal hentes og hoppes over (skip, take). Paginering på denne måten har jeg utført veldig lite tidligere, men jeg skjønte litt hvordan det skulle settes opp. Jeg har valgt å hente 5 verktøy om gangen. Det som gjenstod, var å kunne bla mellom sidene og oppdatere dataen i sanntid. Det var dette som tok mest tid i dag.

Litt etter lunsj spurte jeg om litt hjelp fra min kollega før han måtte rekke et møte. Jeg spurte om hjelp til hvordan og hvor jeg burde legge «mutate» funksjonen. Denne «mutate» funksjonen skal oppdatere dataen som blir hentet fra API’et.

**Kjapp Mutate forklaring:**

Under ser vi bilde av SWR hooken som jeg har kaldt «GetItems» og importerer inn i filen som skal fremvise dataen:



Variabelen «items» inneholder all dataen og «mutate» er funksjonen som lar meg hente dataen fra databasen når funksjonen kjøres.

**Ferdig med forklaring.**

Planen var å få dataen til å «mutate» for hver gang brukeren trykkes på «tilbake» eller «neste» knappen, men vi klarte vi ikke.

Etter å ha brukt flere timer på å få dataen til å oppdatere seg i sanntid, så fant kollegaen min ut av det med bruk av useEffect og få «mutate» til å oppdatere for hver gang variablen for «skip» og «take» blir oppdatert. UseEffect hooken måtte ligge inni useSWR Hooken som vist på bilde under:

Et bilde som inneholder tekst, skjermbilde, programvare

Automatisk generert beskrivelse

Da det endelig fungerte gjensto det bare å få knappene «tilbake» og «neste» til å forsvinne dersom det ikke var flere verktøy å vise (for å gå frem og tilbake). Jeg hadde allerede skrevet en slik løsning for å få knappene til å forsvinne i en tidligere oppgave, så jeg kopierte kode derifra.

**Link hvor jeg hentet koden:** [**https://github.com/Mathias231/next-auth-fagprove-ovelse/blob/main/components/posts/listPosts.tsx**](https://github.com/Mathias231/next-auth-fagprove-ovelse/blob/main/components/posts/listPosts.tsx)

Det var bare et lite problem som jeg måtte endre, og det var å hente total lengde fra databasen for å finne ut når «neste»-knappen skulle forsvinne og vise totall sidetall. Dette var også et tips kollegaen min nevnte mens han satt ved meg.

Prisma har en .count() funksjon som gir total lengde av objekter i databasen, så jeg måtte bare få sendt dette sammen med dataen jeg allerede returnerte.

Her hentes totalt antall verktøy og utstyr som eksisterer i DB:



Neste var å returnere totalAmount sammen med item (dataen) som et objekt:



Deretter lagde vi en ny type som inneholder Item og totalAmount for å definere hva som kommer fra databsen.

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Og til slutt returnere pageLength, pageOffset, items og count tilbake til client.

Et bilde som inneholder tekst, skjermbilde, Font, design

Automatisk generert beskrivelse

Slik ble resultatet av pagineringen og under er et bilde av koden for knappene med sidetall imellom:

Et bilde som inneholder tekst, skjermbilde, Font, design

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, programvare

Automatisk generert beskrivelse

Bilde fra da jeg spurte ChatGPT om paginering:

Et bilde som inneholder tekst, skjermbilde, programvare, Multimedieprogramvare

Automatisk generert beskrivelse

Kalkulere totalt antall sider

Et bilde som inneholder tekst, skjermbilde, programvare, Font

Automatisk generert beskrivelse

På slutten av dagen før dokumentering så startet jeg på den nye siden hvor administrator får legge til nye verktøy og utstyr. Dette fortsetter jeg med i morgen.

## Resultat av dagen

* Problem og feilsøking
* Paginering fungerer
* Verifisert database
* Startet på innleggingsside for verktøy og utstyr
* Mye dokumentering

## Hjelpemidler og lenker brukt i dag

* ChatGPT – Kalkulere totall sidetantal og page (currentPage)
* Fått hjelp fra kollega med paginering

Lenke for prosjektet jeg gjenbrukte kode fra: [**https://github.com/Mathias231/next-auth-fagprove-ovelse/blob/main/components/posts/listPosts.tsx**](https://github.com/Mathias231/next-auth-fagprove-ovelse/blob/main/components/posts/listPosts.tsx)

ChatGPT: <https://chat.openai.com/auth/login>

TailwindUI Form: <https://tailwindui.com/components/application-ui/forms/form-layouts>

# Fredag 2.06.2023

Denne morgenen fortsatte jeg videre på innleggingsside av verktøy og utstyr ifølge planen. Jeg valgte også å legge til navn på Item modellen i prisma.schema fordi jeg vil ha navn for både utstyr og verktøy. Grunnen for at jeg ikke la det til tidligere med en gang var fordi jeg tenkte å bruke «itemType» som beskrivelse når brukere ser verktøyet på nettsiden.

Rundt klokken 10:15 viste jeg fram innleggingssiden til min kollega da han gikk forbi. Han så at jeg holdt på med innleggingssiden og jeg pratet om planene for denne siden. Jeg fikk litt styling tips som jeg valgte å bruke i utseendet på denne siden. Jeg lastet også ned en module som heter React-DatePicker. Den skal altså la brukeren velge dato for dagen verktøyet eller utstyret ble kjøpt.

Bilde av DatePicker:

Et bilde som inneholder tekst, skjermbilde, nummer, line

Automatisk generert beskrivelse

Ved trykk så åpnes kalenderen.

Med kollegaen ved min side diskuterte vi utseendet for innleggingsforumet (form). Vi testet ut mye forskjellig og kom fram til en konklusjon.

Slik ble utseendet til innleggingsformet ut:

Et bilde som inneholder tekst, skjermbilde, nummer, Font

Automatisk generert beskrivelse

Den er også responsive på telefon:

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse

Ble ferdig med utseendet klokken 12:06.

Litt etterpå da klokken var 12:28, nevnte min kollega en viktig del som jeg ikke hadde lagt til enda. Det var nemlig sikkerhet på backend. Jeg hadde ikke lagt til enda en funksjon som sjekker om brukeren er logget inn eller er administrator før man skal håndtere dataen i databasen.

En annen dum ting jeg hadde gjort var å legge til «userId» på verktøyene og sender dem via req.body, som man egentlig ikke skal gjøre. Grunnen for at man ikke skal gjøre det er for å hindre andre brukere/hackere i å få tak i andre brukere’s ID. På startsiden hvor alle verktøy og utstyr vises, kjøres en GET Request som henter disse gjenstandene ut fra databasen. Hvis man åpner «Developer Tools Interface» (f12) og navigerer seg til Network, kan man se kallet som henter verktøyene. I responsen så var «userId» også sendt. Dette er nå fjernet.

Et bilde som inneholder tekst, skjermbilde, programvare, Dataikon

Automatisk generert beskrivelse

Nå vises bare ID’en til verktøyet og resten av innholdet om et verktøy.

Ved opprettelse av nye verktøy så hadde jeg gjort noe like dumt som er å sende «userId» fra client til backend via req.body. Dette fjernet jeg og spurte min kollega om hvordan jeg da skulle legge til «userId» på verktøy. Jeg fikk svar tilbake om at jeg burde hente «userId» fra session og håndtere dette i backend med bruk av GetServerSession, som er en funksjon fra AuthJs. Jeg har ikke brukt det tidligere, så jeg fikk hjelp med å sette opp et middleware som legger session på «NextApiRequest» (req) i API’et.

Vi lagde først en test i [itemId].ts hvor vi brukte getServerSession og la på user.session i req, men flyttet dette og lagde det som et middleware for å gjenbruke dette for alle API rutene.

Bilde av Middleware:

Et bilde som inneholder tekst, skjermbilde, programvare

Automatisk generert beskrivelse

Tidligere nevnte jeg at jeg sendte «userId» via req.body fra client ved opprettelse av verktøy. Nå er dette endret.

Før:

/api/item/index.ts

Et bilde som inneholder tekst, skjermbilde, display, Font

Automatisk generert beskrivelse

Etter:

Et bilde som inneholder tekst, skjermbilde

Automatisk generert beskrivelse

Det neste vi gjorde sammen var å fremvise bildet på skjermen før bildet sendes og lagres på nettsiden. Vi fant en løsning som viser hvordan det settes opp i React.

Link til løsning: <https://stackoverflow.com/questions/38049966/get-image-preview-before-uploading-in-react>

Bilde av resultat:

Et bilde som inneholder tekst, skjermbilde

Automatisk generert beskrivelse

Klokken 16:20 ble vi ferdig med fremvisningen av bildet som vist over.

Det neste jeg jobbet med var opplastning av verktøy før jeg dro. Jeg måtte først få lastet opp bilde slik at jeg får «imageId» som jeg kan legge til på et verktøy for å referere bilder med verktøy. Jeg har tidligere jobbet veldig lite med håndtering av bilder og opplastning av bilder til database, så jeg valgte å bruke en løsning fra et tidligere prosjekt og hente koden derfra. Jeg valgte å skrive koden selv for å en forståelse og lære meg hvordan det fungerer. Det er 2 plasser jeg må håndtere bildet på, og det er på client og backend. Først skrev jeg en funksjon på client og kalte den for handleImageChange.

Bilde av handleImageChange:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Når brukeren har valgt bildet, kjøres funksjonen og legger bildet inn i en useState.

Når bildet velges så er bildet et array eller null, så hvis filen er et array, så tar jeg det første bildet i array. Hvis ikke blir variabelen «file» null.

Det neste var å legge til en sjekk for hvorvidt størrelsen på bildet var over 10MB eller ikke:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Hvis alt er greit blir bildet lagt til i «image»



Det neste jeg måtte gjøre var å sende bildet til API’et /api/image/index.ts.

For dette måtte jeg formatere dataen som skal sendes. Jeg brukte et grensesnitt («Interface») som heter FormData som kommer med react. Grunnen for at jeg legger bildet i et grensesnitt er fordi jeg ikke får sendt bilder direkte via axios. Bildet må derfor settes som et grensesnitt.

Bilde av grensesnitt-håndtering, sending av data og mottak av «imageId»:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Det neste og siste som trengtes var å lagre bildet i databasen og i en mappe. Deretter returneres «imageId» slik at opprettelse av nye verktøy fungerer.

Tidligere nevnt så har jeg ikke operert mye med opplastning og lagring av bilder, så jeg valgte å se på eksemplet jeg hadde fra en øvelse jeg hadde jobbet med. Dette eksemplet hadde min kollega hjulpet meg tidligere å sette opp, men jeg har ikke helt lært meg hvordan logikken fungerer med bruk av npm pakken «Formidable».

Jeg valgte likevel å skrive koden selv og teste underveis for lære meg logikken i det. Før jeg startet å kode selv måtte jeg sette opp en funksjon som analyserer grensesnittet som blir mottatt fra klienten. Denne funksjonen har vi tidligere lagt i «utils» mappe i eksemplet jeg ser på.

Bilde av funksjonen:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Det neste jeg gjorde var å skrive koden i /api/image/index.ts. Som tidligere nevnt testet jeg koden mens jeg utviklet for å lære meg hva som skjer igjennom prosessen. Jeg fant ut at etter at funksjonen FormidableParseAsync var kjørt ferdig, returneres filen som et Array. Dette fant jeg ut etter jeg prøvde å returnere filen (som en test), da jeg fikk feilmeldingen om at filen er et array. For å fikse dette så hadde vi i eksemplet laget en If-statement som velger det første bilde i arrayet.

Et bilde som inneholder tekst, Font, skjermbilde, nummer

Automatisk generert beskrivelse

Videre var det en sjekk som verifiserer om bildet er av type «image»:

Et bilde som inneholder tekst, Font, skjermbilde, line

Automatisk generert beskrivelse

Dersom bildet var av typen «image», opprettes bilde i databasen og returnerer «imageId» tilbake til clienten:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Til slutt måtte jeg legge til en konfigurasjon som slår av «BodyParser». Grunnen for dette er at i NextJs er «BodyParser» aktivert som standard. Jeg innser nå at det er derfor funksjonen formidableParseAsync eksisterer.

Bilde av config:

Et bilde som inneholder tekst, Font, skjermbilde, design

Automatisk generert beskrivelse

Da jeg skulle teste nettsiden trodde jeg at alt skulle fungere, men det gjorde det ikke. Jeg fikk en feil fra Axios der meldingen var: AxiosError: Request failed with status code 500. Jeg forsto ikke hva det gjaldt, men sjekket terminalen på serveren for å finne ut hva som kunne ha skjedd. Terminalen viste: «TypeError: resolver is not a function». Jeg søkte etter en løsning på nett uten hell. Jeg satt deretter opp routen i API’et på nytt for å sakte, men sikkert teste om dataen når serveren. Da det endelig begynte å fungere og jeg så at dataen jeg sendte nådde API’et, skrev jeg samme funksjonalitet som sist, men mens jeg fulgte med på eksemplet på min andre skjerm. Dette fungerte etter 2 timer med feilsøking.

Jeg lastet også ned Toastify som er en module for å vise tilbakemeldinger på nettsiden. Under er bilde av Toastify som forteller brukeren at gjenstanden er blitt lagt til i databasen:

Et bilde som inneholder tekst, skjermbilde, Font, line

Automatisk generert beskrivelse

Det siste jeg gjorde før jeg dro var å endre Enum defineringen for Category i prisma.schema fordi tidligere så hadde jeg definert kategoriene for «TOOL» og «EQUIPMENT». Endret til norsk «VERKTOY» og «UTSTYR». Likevel har jeg lagt inn data med kategorien «TOOL» og «EQUIPMENT», så jeg kaller dette for en «blind bevissthet».

Før:

Et bilde som inneholder Font, tekst, skjermbilde, Grafikk

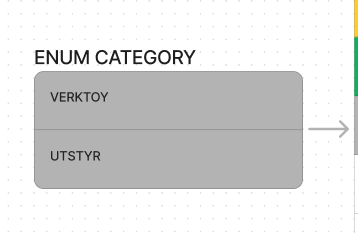
Automatisk generert beskrivelse

Nå:

Et bilde som inneholder Font, tekst, Grafikk, skjermbilde

Automatisk generert beskrivelse

Bilde av endret Enum Category i form av skisse



## Resultat av dagen

* Problem og feilsøking
* API og Rutes
* Bildefremvisning på Frontend
* Bildehåndtering
* Fungerende innleggingsmetode av verktøy og utstyr
* Sikkerhet på Backend
* Endring i Prisma.Schema
* Mye dokumentering

## Hjelpemidler og lenker brukt i dag

* Fått tips og hjelp fra kollega

Bildefremvisning løsning: <https://stackoverflow.com/questions/38049966/get-image-preview-before-uploading-in-react>

Lenke for prosjektet jeg gjenbrukte kode fra: <https://github.com/Mathias231/garage>

Toastify Module: <https://www.npmjs.com/package/react-toastify>

Formidable Løsning: <https://stackoverflow.com/questions/30128701/parse-form-value-with-formidable-to-filename>

DatePicker: <https://www.npmjs.com/package/react-datepicker>

# Lørdag 3.06.2023

I dag har jeg brukt mesteparten av dagen på å dokumentere resten av hendelsen i går og oppdatert «Prosjektplan (ettertid)» som skal vise endringene som skjedde underveis.

Etter jeg var ferdig med dokumenteringen, begynte jeg med å forbedre utformingen for gjenstandene.

Jeg startet først med å lage en «border» rundt gjenstandene og gi litt størrelse:

Et bilde som inneholder line, skjermbilde, tekst

Automatisk generert beskrivelse

Det neste jeg gjorde var å få bildene til å fungere.

For å vise frem bildet så måtte jeg hente ut «internalName» fra tabellen Image. Tidligere i fetch-kallet så hentet jeg «imageId». Jeg byttet ut «imageId» til å hente «internalName» fra Image tabellen siden de har relasjon med hverandre.

Før:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Etter:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Jeg la også på «internalName» på typen IItem:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Deretter la jeg til stien på elementet <Image />:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Det neste jeg satt opp var en ny fil i /api/image/ som skal hente bildet med bruk av navnet til bildet som sendes fra client. Jeg har tidligere gjort noe lignende, så jeg valgte å gjenbruke kode.

Bilde av hentet kode fra tidligere prosjekt:

Et bilde som inneholder tekst, skjermbilde, programvare, display

Automatisk generert beskrivelse

Det som skjer først, er at navnet til bildet blir sendt til API’et og mottat av denne filen fordi filen heter [id].ts

Når filer har [ ] (Square Brackets) rundt seg, så leser/mottar filen tekst som blir sendt til stien. For eksempel /api/image/**23817313128937.png ->** leste **23817313128937.png.**

Jeg henter teksten med bruk av req.query og legger teksten i variabelen «id»

Videre så sjekker jeg om variabelen er av typen string:

Et bilde som inneholder Font, tekst, skjermbilde, line

Automatisk generert beskrivelse

Deretter henter bildet hvor navnet tilsvarer samme navn som er i variabelen:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Jeg bruker fs.readFileSync som er en Node Module som lar meg lese bildet. Bildet inneholder masse tall som ser slik ut:



Slik ser det ut nå:

Et bilde som inneholder tekst, skjermbilde, diagram, Plottdiagram

Automatisk generert beskrivelse

Har også lagt til at admin kan slette en gjenstand:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

## Resultat av dagen

* Image routes
* Bildefremvisning på Frontend
* Bildehåndtering
* Forbedret utseendet for gjenstander
* Testing som ny bruker
* Mye Dokumentering

## Hjelpemidler og lenker brukt i dag

* Gjenbruk av tidligere kode: https://github.com/Mathias231/garage/tree/main/pages/api/image
* TailwindCSS: https://tailwindcss.com/
* DatePicker: <https://www.npmjs.com/package/react-datepicker>

# Søndag 4.06.2023

I dag har jobbet hardt for å ta igjen det jeg ikke rakk tidligere ifølge timeplanen. Jeg har videreutviklet en kommentarside for verktøyene, testet nettsiden og sjekket funksjonalitet. Fram til nå så har jeg funnet noen få feil «bugs» og mangler i funksjonaliteten for å endre innhold og slette kommentarer. Det er mulig å slette verktøy, så det er noe som var galt i DELETE Routen for kommentarer.

Det første jeg jobbet med om morgenen var å lage kommentarsiden for verktøy. Mens jeg utviklet så innså jeg rundt 12:30 at jeg heller burde ha generert en egen ID for gjenstandene slik at jeg kan bruke den istedenfor «itemId» i URL’en. Det er med tanke på sikkerhet og for å hindre «itemId» fra å vises offentlig.

Derimot, så sjekker jeg allerede i API ‘et om DELETE forespørselen kommer fra en innlogget bruker som har Admin rollen:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Frem til klokken 18:35 har jeg fått utviklet kommentarsiden og laget skisser for funksjonalitet.

Første er funksjonaliteten for å opprette en ny gjenstand:

Et bilde som inneholder tekst, diagram, skjermbilde, Post-it-lapp

Automatisk generert beskrivelse

Andre er funksjonaliteten for å slette en gjenstand:

Et bilde som inneholder skjermbilde, tekst, diagram, Font

Automatisk generert beskrivelse

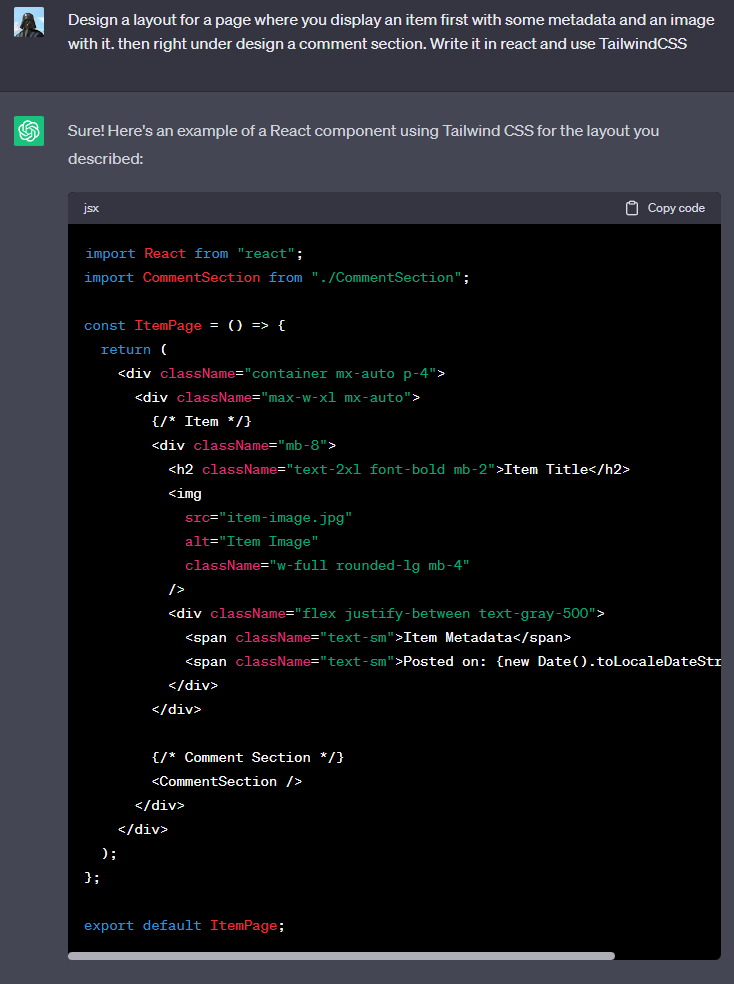
Bilde av kommentarsiden:

Et bilde som inneholder verktøy, skrutrekker

Automatisk generert beskrivelse

Jeg valgte å bruke ChatGPT til å generere utseendet for kommentarsiden. Grunnen for dette var å spare tid slik at jeg kunne ha mer tid til å dokumentere.

Bilde av hentet kode:



Mens jeg utviklet på kommentarsiden, trengte jeg en custom hook som henter et verktøy og kommentarene knyttet til verktøyet. Jeg valgte å bruke en tidligere laget custom hook som hentet Workshop og verktøyene. Siden jeg ikke brukte den lenger valgte jeg å endre innholdet og bruke filen for kommentarsiden.

Bilde av custom hook:

Et bilde som inneholder tekst, skjermbilde, display, programvare

Automatisk generert beskrivelse

Det er ikke så mye som er endret utenom stien til API ‘et og la til en ny type til spørringen.

Først lagde jeg en type som skal brukes til spørringen:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Deretter lagde jeg en type for kommentarer:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Jeg knyttet sammen IComment til IItemWithComments fordi IComment innehold typene jeg trengte for en kommentar, så jeg gjenbrukte dette istedenfor å skrive mer. Noe jeg burde også nevne er hvorfor jeg har navngitt hver type med bokstaven «i» først. Det er referert som «Interface» og navnet interfacet.

Etter at jeg hadde typedefinert custom hooken lagde jeg en ny GET Route som ligger i fil-stien /api/item/comment/itemWithComment/[itemId].ts.

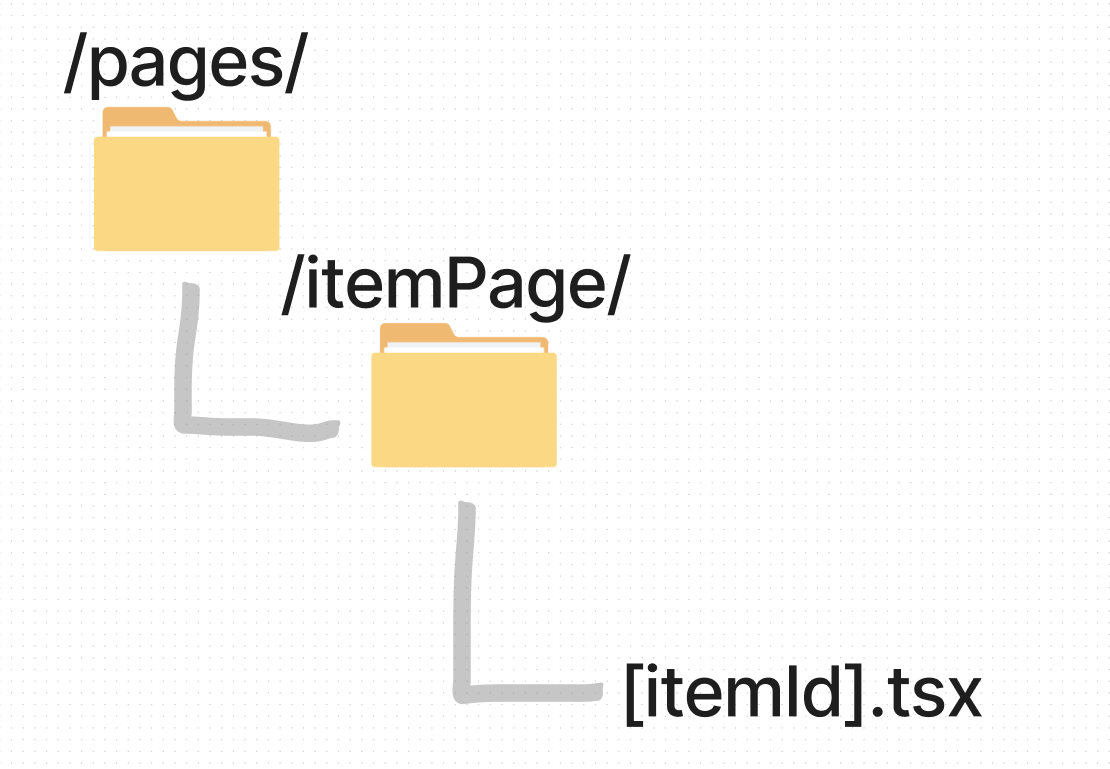
Filen henter et verktøy med kommentarene tilknyttet til verktøyet sammen med brukeren som opprettet kommentaren.

Bilde av filen [itemId].ts som ligger i stien /api/item/comment/itemWithComment/[itemId].ts:

Et bilde som inneholder tekst, skjermbilde

Automatisk generert beskrivelse

Det neste jeg gjorde var å opprette siden hvor kommentarene skal vises. Jeg ville at verktøyet skulle vises på toppen og ha kommentarene under, så jeg lagde en ny fil i /pages/ som jeg kalte for /itemPage/ med en fil som heter [itemId.tsx]:



Filen er ikke så stor fordi jeg har laget en komponent som inneholder meste av innholdet som vises på nettsiden.

Bilde av [itemId].tsx:

Et bilde som inneholder tekst, skjermbilde, programvare, display

Automatisk generert beskrivelse

I koden over så henter jeg itemId fra URL ‘en og sender den videre til komponenten <SingleItemPage />

Første del av koden i singleItemPage.tsx:

Et bilde som inneholder tekst, skjermbilde, display, programvare

Automatisk generert beskrivelse

Har lagt til en «form» hvor bruker kan opprette kommentarer på verktøy og utstyr.

Andre del av singleItemPage.tsx

Et bilde som inneholder tekst, skjermbilde, programvare, display

Automatisk generert beskrivelse

Utsendet er laget av ChatGPT og jeg har modifisert den litt.

Bilde av hentet kode generert av ChatGPT:

Et bilde som inneholder tekst, elektronikk, skjermbilde, programvare

Automatisk generert beskrivelse

Jeg valgte å bruke ChatGPT fordi jeg ville spare tid og jobbe videre med funksjonalitet.

Til slutt så har jeg lagt til en ny komponent som heter <Comment /> som fremviser kommentarene knyttet til verktøyet.

Bilde av komponenten Comment:

Et bilde som inneholder tekst, skjermbilde, programvare

Automatisk generert beskrivelse

Utseendet har jeg laget selv og jeg jobber med å få funksjonen som sletter kommentarer til å fungere.

Funksjonaliteten for å slette kommentarer via client fungerer ikke helt som den skal enda. Har hatt problemer med å motta «commentId» i API ‘et fordi jeg får et problem med zod.

Følgende feilmelding:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

## Resultat av dagen

* Problem og feilsøking
* Kommentarside for verktøy
* Gjort litt testing av nettsiden
* Laget skisser for funksjonalitet
* Utseendet for kommentarside
* Mye dokumentering

## Hjelpemidler og lenker brukt i dag

* ChatGPT: <https://chat.openai.com>
* TailwindCSS: https://tailwindcss.com/

# Mandag 5.06.2023

I dag startet jeg morgenen med å dokumentere i cirka 2-3 timer om arbeidsprosessen i går. Jeg har også vist fram min kollega arbeidet jeg har utført igjennom helgen og fikk litt tips og tilbakemelding på det. Han syns alt så bra ut, men det var noen plasser som kunne ha vært bedre, så jeg spurte med en gang hva jeg kunne utbedre.

Det første var å rydde opp i mappestrukturen i API ‘et fordi det så litt rotete ut. Jeg var enig i dette. Noe jeg ikke tenkte på da jeg utviklet sent på kvelden, var at i stedet for å lage en sti til /api/item/comment/itemWithComment/[itemId].ts, så kunne jeg heller ha laget en GET Route i /api/item/[itemId].ts. Etter at jeg flyttet filene fikk jeg en «a-ha» opplevelse og så med en gang at strukturen ble bedre. Tidligere så flyttet jeg mappen /comments/ inn i mappen /item/ fordi jeg tenkte det gir mening å ha comment i item. Kollegaen min sa at det heller er bedre å ha /comment/ som en egen mappe fordi det er «best practice».

Bilde av ny mappestruktur.

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

(Har ikke bilde av tidligere mappestruktur)

Etter det, så satt jeg i gang med å teste nettsiden ifølge tidsplanen fra planleggingen. Jeg testet nettsiden som en vanlig bruker og administrator. Jeg fikk også testet nettsiden med nye og flere brukere. Jeg fant noen bugs, deriblant tilgang som vanlige brukere ikke skulle ha. Jeg la derfor inn en sjekk på /api/item[itemId].ts og et if-statement for sletteikonet som vises på /pages/itemPage.tsx.

Et bilde som inneholder tekst, skjermbilde, display, Font

Automatisk generert beskrivelse

Jeg har også klart å få slettefunksjonen for kommentarer til å fungere. Feilen jeg gjorde tidligere var at jeg hadde skrevet req.query.itemId istedenfor req.query.commentId

Feil:

Et bilde som inneholder tekst, Font, skjermbilde

Automatisk generert beskrivelse

Løsning:

Et bilde som inneholder tekst, skjermbilde, Font, line

Automatisk generert beskrivelse

Etter at jeg testet nettsiden satte jeg i gang med å lage skisser for funksjonalitet som fungerer på nettsiden. Jeg har ikke implementert funksjonen for å endre innhold på gjenstander og kommentarer, men planlegger å gjøre det enten senere i dag eller i morgen når jeg har tid.

Skisse for opprettelse av nye gjenstander:

Et bilde som inneholder tekst, skjermbilde, Post-it-lapp, diagram

Automatisk generert beskrivelse

Skisse for sletting av eksisterende gjenstander:

Et bilde som inneholder skjermbilde, diagram, line, Font

Automatisk generert beskrivelse

Skisse for funksjonaliteten som henter gjenstandene fra databasen:

Et bilde som inneholder tekst, skjermbilde, diagram, Font

Automatisk generert beskrivelse

Skisse for sletting av kommentarer:

Et bilde som inneholder skjermbilde, tekst, diagram, Font

Automatisk generert beskrivelse

Skisse for oppretting av kommentarer:

Et bilde som inneholder tekst, skjermbilde, diagram, Post-it-lapp

Automatisk generert beskrivelse

Skisse for henting av gjenstand med kommentarer og skaperen av kommentaren:

Et bilde som inneholder tekst, skjermbilde, Font, diagram

Automatisk generert beskrivelse

## Resultat av dagen

* Laget skisser av funksjonalitet
* Brukertillatelser for å slette kommentar
* Smått testet nettsiden etter bugs
* Fikset slettefunksjonalitet på kommentarer
* Rydding i mappestruktur
* Mye dokumentering

## Hjelpemidler og lenker brukt i dag

* Figma: www.figma.com/
* Tips fra kollega

# Tirsdag 6.06.2023

I dag startet jeg dagen med å dokumentere videre og teste nettsiden fram til lunsj. Etter lunsj skulle jeg vise min kollega en bug som jeg nylig la merke til på startsiden. Kollegaen min la merke til at startsiden lastet inn tregere enn normalt. Vi sjekket «Network» i utvikler-konsollen og så at gjenstander ble hentet 8 – 10 ganger for hver gang jeg oppdaterte siden.

Jeg feilsøkte i cirka 15 minutter til jeg fant ut at i filen /components/items/item.tsx hadde jeg importert custom hooken GetItems.ts for å kjøre «mutate».

Jeg fant ut at custom hooken GetItems.ts var importert i filen /components/items/item.tsx når GetItems.ts allerede var importert i index.tsx (som er «parent» til item.tsx). Jeg endret komponenten slik at «mutate» blir sendt til «child component».

Før:

Et bilde som inneholder tekst, skjermbilde, Font, line

Automatisk generert beskrivelse

Etter:

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

I filen /component/items/item.tsx

La til type

Et bilde som inneholder tekst, skjermbilde, Font, display

Automatisk generert beskrivelse

Jeg fortsatte videre med dokumentering og lagde siste skisse som viser nettsiden i sin helhet som ikke innlogget, innlogget og innlogget med Admin rettigheter:

Et bilde som inneholder tekst, diagram, skjermbilde, plan

Automatisk generert beskrivelse

Rundt klokken 17:20 leste jeg og min kollega igjennom heftet og rettet skrivefeil.

Klokken 22:29 ble jeg ferdig med README filen og lastet opp dokumentene til GitHub.

## Resultat av dagen

* Laget skisser av funksjonalitet
* Testet nettsiden og fikset bugs
* Lastet opp dokumentene til GitHub
* Skrevet README
* Mye dokumentering

## Hjelpemidler og lenker brukt i dag

* Figma: www.figma.com/
* Tips fra kollega