Dokumentasjon

Mathias Voie Johansen – 06.02.25

Zod

Zod brukes for validering av input-data både på klientsiden (f.eks. i skjemaer) og serversiden (f.eks. i API-endepunkter). Dette sikrer at all data som sendes til serveren er i riktig format og følger spesifikke regler, som:

* Navn må være en tekststreng med minst 8 og maks 64 tegn.
* Beskrivelse er valgfri, men kan ikke overskride 250 tegn hvis den er tilstede.

Fordeler:

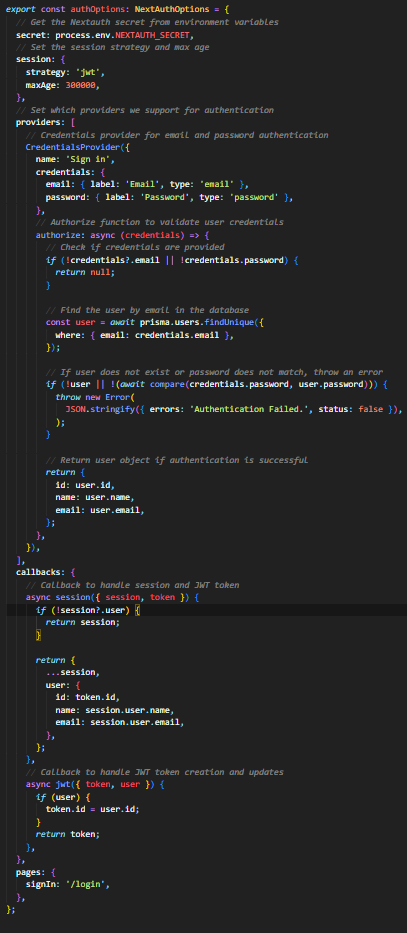
* Forhindrer ugyldig data fra å bli sendt videre til backend (API-kall stoppes automatisk hvis validering feiler).
* Reduserer belastning på serveren, siden unødvendige kall ikke blir gjort.
* Gir brukeren raskere tilbakemelding, da validering skjer allerede på klientsiden.
* Gjenbrukbar logikk – samme skjema kan brukes på begge sider for konsistent datavalidering.

A screen shot of a computer program

AI-generated content may be incorrect.

NextAuth

**Konfigurasjon og strategi**

For dette prosjektet er det brukt NextAuth.js for å håndtere autentisering via en modul kalt extauth. Dette er en kraftig autentiseringsløsning som støtter mange populære leverandører som Google, GitHub, Discord, med mer. Det er også enkelt å integrere i ulike prosjekter/løsninger.

* **Session-strategi:** JWT brukes for økt ytelse og fleksibilitet, spesielt i serverless-miljøer.
* **Maks sessionsvarighet:** Er satt til 3,5 dager (300 000 sekunder). Dette er for å sikre at brukeren forblir innlogget i en passende tidsperiode, samtidig som det beskytter mot langvarige sesjoner hvor brukere kan få uønsket tilgang uker/måneder senere.

**Egendefinert pålogging med e-post og passord**

Ved bruk av CredentialsProvider tillater løsningen lokal innlogging med e-post og passord. Dette skjer via en **authorize-funksjon**:

* Hvis ingen gyldige credentials er sendt, returneres null.
* Dersom e-posten ikke finnes, eller passordet er feil, stoppes autentiseringen og en feilmelding returneres.
* Dersom alt stemmer, returneres spesifikk brukerinfo som lagres i JWT/session: id, name, email.

**Sessions og JWT-callbacks**

* **Session-callback** brukes til å legge til ekstra informasjon i session-objektet.
* **JWT-callback** brukes for å tilpasse tokenet når en bruker logger inn – spesielt nyttig når man vil ha kontroll over ID-er og metadata.

Routing

A screen shot of a computer

AI-generated content may be incorrect.Next.js tilbyr et avansert og fleksibelt routing-system som støtter en rekke moderne navigasjonsmønstre som:

* **Dynamic Routes** – for variabel-baserte URL-er
* **Parallel Routes** – for flere samtidige views i samme rutehierarki
* **Intercepting Routes** – for å fange og vise nye ruter som "overlays", f.eks. i modaler

**Bruk av Intercepting Routes i prosjektet**

I dette prosjektet ble intercepting routes brukt for å håndtere modaler – på en elegant og isolert måte. Dette gir flere fordeler:

* **Modal-logikken** er separert fra hovedsiden, og trenger ikke spesialhåndtering i komponentene som gjengir hovedinnholdet.
* Modalene rendres i en egen rute og blir automatisk knyttet til layout-filen i samme katalog via Next.js sitt **routing-system**.
* Dette gir en svært modulær og skalerbar arkitektur, hvor nye modaler og views kan legges til uten å forstyrre eksisterende sider.

**Automatisk integrasjon med Layouts**

Intercepting routes fungerer sømløst med Next.js sin layout-struktur

* Ruter som ligger i undermapper som **@modal** blir automatisk "fanget opp" av layout-filen i samme mappe **(dashboard/layout.tsx i eksempelet over)**.
* Dette gir en naturlig sammenkobling mellom layout og modalene, og åpner for enkel implementasjon av **rollestyring**, **beskyttede sider**, og tilpassede **UI-opplevelser** basert på **brukertype**.

**Oppsummering**

* Ved å bruke Intercepting Routes i Next.js oppnår man:
* Ren og adskilt kode for modaler
* Ingen behov for lokal tilstandshåndtering i sider
* Sømløs integrasjon med layout og route-segmentering
* Et fleksibelt system som er godt egnet for skalerbare og rollebaserte applikasjoner

Tailwind

I dette prosjektet er Tailwind CSS brukt som hovedverktøy for styling. Tailwind er et utility-first CSS-rammeverk som lar utviklere raskt og enkelt bygge tilpassede brukergrensesnitt – uten å skrive tradisjonelle .css-filer.

**Effektiv utvikling med ferdige klasser**

* **Tailwind** gjør det mulig å bruke ferdigdefinerte klasser direkte i **HTML (JSX/TSX)**, som **f.eks**. **p-4**, **bg-gray-100**, **text-lg**, **rounded-md**, etc. Dette gir flere fordeler:
* Ingen behov for **manuelle** CSS-filer
* Alt styles direkte i komponentene, som gir høyere **lesbarhet** og **vedlikeholdbarhet**
* Man kan designe **raskere** uten å forlate komponentfila
* **Responsivt** design og tilpasning for ulike skjermstørrelser er innebygd med enkle breakpoints (**sm**:, **md**:, **lg**:, osv.)

**Hvorfor Tailwind?**

* Det gir **full** kontroll over utseende uten å måtte opprette eller holde styr på mange CSS-klasser
* Det fjerner behovet for å bytte mellom **HTML/JSX og CSS-filer**
* Det gjør det **lettere** å gjenbruke mønstre og stilvalg
* Ved bruk sammen med komponentbaserte biblioteker **(som React/Next.js)**, gjør det utvikling **raskere** og mer **konsistent**

Prisma

I dette prosjektet benyttes **Prisma** som **Object-Relational Mapper (ORM)** for å håndtere databasen. **Prisma** gjør det enkelt og **intuitivt** å jobbe med databasen ved hjelp av en typesikker og deklarativ tilnærming.

**Enkel modellering med schema.prisma**

Alt av databasedefinisjon skjer i én fil: schema.prisma. Her beskrives datamodellen med tydelig støtte for:

* Datatyper og validerin
* Relasjoner mellom tabeller
* Unike constraints
* Standardverdier og tidsstempler

**Prisma** genererer automatisk TypeScript-typer og gir intellisense i utviklingsverktøyet (VS Code m.fl.), noe som reduserer feil og forbedrer utvikleropplevelsen betydelig.

**Modeller**

Et konkret eksempel fra prosjektet er Tools-modellen. Den representerer verktøy som brukere kan legge til i mapper, og inneholder:

**En relasjon til:**

* Images via imageI
* Folders via folderId

**Felter:**

* name: Navn på verktøyet (påkrevd)
* quantity: Antall, med en standardverdi på 1
* description: Valgfri beskrivelse

**Tidsstempler:**

* createdAt: Når verktøyet ble opprettet
* updatedAt: Når verktøyet sist ble endret

**Unik begrensning:**

* Kombinasjonen av id og folderId er unik, noe som forhindrer duplikate verktøy i samme mappe

A screen shot of a computer program

AI-generated content may be incorrect.

**Hente relaterte data med include**

Ved hjelp av Prisma sin include-funksjonalitet kan man hente ut relaterte entiteter i én enkelt spørring. Her er en oversikt over planen jeg fikk godkjent av prøvenemda også har jeg markert med grønn og rødt av endringene som ble gjort underveis. Dette lar oss hente alle verktøy i en gitt mappe og deres tilhørende bilder i ett API-kall, noe som forbedrer både ytelse og kodeforståelse.

Tidsplan

**Mandag**

* Lese og forstå oppgaven som er levert.
* Lage løsningsforslag
* Jobbe med den teoretiske delen av oppgaven
* Klargjøre GitHub repository.
* Oppsett backend (PostgreSQL)
* Sette av tid til egenvurdering og dokumentasjon

**Tirsdag**

* API ( POST, GET)
* Rolle-styrt adgang
* Bildeopplastning til server
* Autentisering
* Sikkerhet for brukere
* Sette av tid til egenvurdering og dokumentasjon

**Onsdag**

* API ( PUT, PATCH, DELETE )
* Bildeopplastning til server
* Utforming & design av nettside
* Utvikle forms til forskjellige handlinger
* Sikkerhet for brukere
* Sette av tid til egenvurdering og dokumentasjon

**Torsdag**

* Putte formene som ble utvikler på onsdag i modals for lett tilgjengelighet
* Ferdigstille bildeopplastningen.
* Sikkerhet for brukere
* Sette av tid til egenvurdering og dokumentasjon
* Utvikle forms til forskjellige handlinge
* Utforming & design av nettside

**Fredag**

* Sikkerhet for brukere
* Ferdigstille bildeopplastningen.
* Ferdigstille utviklingen av API
* Ferdigstille kommunikasjon mellom frontend og backend
* Sette av tid til egenvurdering og dokumentasjon

**Mandag**

* Vurdere HMS i henholdt til oppgaven
* Få prosjektet online med SSL sertifikat.
* Ferdigstille dokumentasjon og egenvurdering
* Sette av tid til egenvurdering og dokumentasjon

**Tirsdag**

Ferdigstille dokumentasjon og PC for fremføringen.

**Onsdag**

* Presentere fagprøven for prøvenemda

Fjernet punktet med å få prosjektet online med SSL sertifikat siden jeg ikke følte det var relevant for vurdering.

Helse, Miljø og Sikkerhet (HMS)

HMS ble tatt på alvor under gjennomføringen av prosjektet, med fokus på både fysisk og mental helse. Flere tiltak ble implementert for å sikre et trygt, komfortabelt og bærekraftig arbeidsmiljø:

**Ergonomi:**

* Arbeidsplassen var utstyrt med en ergonomisk kontorstol og riktig justert bordhøyde for å fremme god sittestilling og redusere belastning på rygg og nakke ved langvarig skjermarbeid.

**Luftkvalitet:**

* God ventilasjon og regelmessig lufting av rommet ble prioritert for å sikre frisk luft og redusere risikoen for hodepine og redusert konsentrasjon.

**Pauser og bevegelse:**

* Det ble satt av tid til korte pauser med frisk luft og lett fysisk aktivitet, som gåturer og tøying, for å opprettholde fokus, redusere stillesittende tid og forbedre mental klarhet.

**Oppsummering**

Ved å legge til rette for både fysisk komfort og mentale pauser ble arbeidshverdagen mer produktiv, helsevennlig og bærekraftig over tid. HMS-tiltakene bidro dermed positivt til både kvaliteten på arbeidet og den generelle trivselen under oppgaven.

Daglig arbeid

**Mandag 26.05.25:**

Fikk utlevert oppgaven kl. 13:00 og startet med å vurdere ulike prosjektideer. Etter en kort ide laging bestemte jeg meg for å utvikle en digital verktøykasse. Sattt også opp test databasen som skulle brukes under utvikling. Deretter lagde jeg en plan for de neste dagene, inkludert oppgaver som kravspesifikasjon, valg av teknologier, oppsett av utviklingsmiljø, samt design og implementasjon av brukergrensesnitt og funksjonalitet. Begynte også på kvelden å lage en skjelett som jeg skulle følge framover.

**Tirsdag 27.05.25:**

Fortsatt der jeg slapp dagen før med videreutvikling av skjelettet på siden. Begynte på skjelettene til modalene uten funksjonalitet. Fikk også ned autentiseringen med nextauth med innloggingsside og registreringsside som lagrer og benytter brukerinformasjonen på en effektiv og sikker måte. Begynte å lage bildeopplastning siden jeg følte det var noe jeg måtte prioritere, brukeren opplaster filen til backendet også gjør serveren det om til binær før det blir klargjort til lagring på serveren.

**Onsdag 28.05.25**

Starter utviklingen av API slik at modal formene har et backend dem kan kommunisere med. Lager også et design som er naturlig, tilgjengelig og appelerende for alle. Dette er da en form for design jeg fulgte hele veien igjennom prosjektet.

**Fredag 30.05.25**

Videre utviklet og ferdigstillte formsene i modalene for å kommunisere med backend ved bruk av axios. Ferdigstillte også bildeopplastningen slik at den er trygg og sikker for trafikk fra klienter. Ferdigstillte utviklingen av API og kommunikasjonen fra klienten til serveren. I dag tok jeg meg et steg tilbake for å få en liten puste pause og reflektere skikkelig over valg som er blitt gjort de siste dagene. Da begynte jeg å ekspirimentere med js classes for å se om det er noe som kan effektivisere framgangen i prosjektet som var en fantastisk ide. Gikk tilbake og endret ting til å bruke klassene jeg lagde. Ble mye bedre orden i koden.

**Mandag 02.06.25**

Videre utviklet & kommenterte / ryddet opp i koden som jeg har produsert de tidligere dagen for at det skal være mere lesbart for folk som ikke har holdt på med prosjektet. Ferdigstilt alt av kode og dokumentasjon slik at det bare er fremføringen som gjenstår.

**Tirsdag 03.06.25**

Ferdigstille dokumentasjon og forberede PC for fremføring for prøvenemda.

Rolle styrt tilganger

**A white rectangular object with black text

AI-generated content may be incorrect.**

Når brukeren er innlogget og er administrator kan man se forskjellige actions som man kan utføre på siden. Dette er gjort på serversiden under compiling av siden. Også replikert på api endepunktene for maksimal sikkerhet.

Modals

A screenshot of a chat

AI-generated content may be incorrect.Med bruk av en component som HeroUI har gjort lett tilgjengelig tilfredsstiller utseende og animasjonene brukeren. Det er også lett å navigere igjennom dem for å få ønsket resultat.

**Activation**

Måten jeg har valgt for å finne ut om modalen skal være aktiv eller ikke er å bruke searchparams.

Grunnen for dette valget er at hvis en bruker skal vise noen som sitter på en annen pc hva de ser, og de sender linken til siden. Vil modalen og alle instillingene deres komme opp på skjermen til brukeren de skal vise det til.



Altvernativet kunne vært å bruke useState men jeg synes det blir for gammeldags med valg som er lagret i memory istedenfor i url når det kommer til nettsider.

Her er da måten jeg henter ut searchParams på. Veldig enkel og fin måte også har jeg en isOpen variabel som da sjekker om modalen skal være aktiv eller ikke.

A screen shot of a computer code

AI-generated content may be incorrect.

isOpen verdien kan også passeres til Form komponenten hvis du skal hente data som er relevant til modalen slik at de ikke blir fetchet med en gang du kommer inn på siden men når du spør etter den dataen.

Database

A screen shot of a computer program

AI-generated content may be incorrect.Her ser du en oversikt over schemaet jeg har laget for databasen. Ved videreutvikling kan man legge ved så mange punkter / informasjon man har lyst på mappene og verktøyene. Her er et veldig simpelt setup

Det som er rødt betyr relations. Da skaper vi en relasjon mellom de to modellene slik at man kan lett inkludere den modellen mens du henter et verktøy. Da velger man keyen som skal matche i relasjonen.

Prisma funker også med mange type databaser ikke bare postgres.

Classes

I denne delen av prosjektet benyttes JavaScript sin innebygde class-syntaks for å strukturere logikken på en ryddig og oversiktlig måte. Selv om det ikke implementeres full objektorientert programmering (OOP) med arv og instansiering, brukes class-strukturen hovedsakelig for:

* Organisering av funksjoner i én samlet enhet
* Kapsling (encapsulation) via private og public metoder

Dette gjør koden mer vedlikeholdbar og forutsigbar – spesielt når funksjonene hører naturlig sammen, men ikke nødvendigvis trenger å være knyttet til objekter eller tilstand.

**Oppsummering**

Selv om class-syntaksen i JavaScript er laget for objektorientert programmering, brukes den her hovedsakelig som et verktøy for å organisere logikk og kontrollere synlighet via private og public metoder. Dette gir bedre struktur og reduserer kompleksiteten i koden.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

API endepunktene

**A screenshot of a computer program

AI-generated content may be incorrect.POST /folders**

Først sjekker vi om at brukeren som benytter seg av endpunktet har en valid session igjennom funksjonene modulen nextauth tilbyr. Om de ikke er innlogget sender vi en 401 unauthorized melding som sier at de ikke har tilgang til å utføre handlingen. Om de har en session men brukeren deres ikke er en administrator vil den også si til brukeren at dem ikke har tilgang til å utføre handlingen. Etter alt av autentisering er godkjeft går vi videre til å prosessere formdataen som er sendt inn hvor vi henter ut navn og beskrivelse for mappen. Så sjekker vi med zod om form dataen som er sendt inn er OK. Om den ikke er OK sender vi en form validation failed feilmelding til brukeren.

Så sjekker vi om at mappenavnet er opptatt og sender en feilmelding om det oppstår en konflikt.

Når alt er gjort bruker vi prisma til å opprette mappen i databasen og revaliderer urlen slik at endringene som ble utført blir dynamisk hentet umiddelbart av klientene.

Sender da en sukse melding til brukeren.

**A screen shot of a computer program

AI-generated content may be incorrect.DELETE /folders/[folderId]**

Her gjør vi som alle viktige eksponerte endepunkter, sjekker at dem er innlogget og har type som administrator før vi prosesserer dataen som er innsendt for å minimere stress på database / server.

Når det er gjort henter vi ut hvilken mappe det er snakk om som vi da sjekker om en mappe med den ID’en finnes. Om den ikke finnes returnerer vi en melding til brukeren som sier at vi ikke fant den forespurte mappen.

Vi sletter mappen med bruk av prisma forså å revalidere urlen slik at brukeren henter inn endringene dem selv har utført.

Sender da en sukse melding til brukeren.

**A screenshot of a computer program

AI-generated content may be incorrect.POST /tools**

Her gjør vi som alle viktige eksponerte endepunkter, sjekker at dem er innlogget og har type som administrator før vi prosesserer dataen som er innsendt for å minimere stress på database / server.

Henter ut all data fra formdataen som klienten sende over for å validere det med zod. Om formdataen inneholder et bilde sender vi det til Image klassen som da genererer en UID (Sjekker også at denne uiden ikke er tatt av et annet bilde) forså å converte det til binær og skrive det til /public/uploads (i produksjon ville dette vært en blob storage osv...) Så tar vi vare på imageId og tilknytter den til mappen når vi oppretter den med prisma.

Så revaliderer vi endringene slik at brukeren får se endringene de har utført.

Sender da en sulse melding til brukeren.

A screen shot of a computer program

AI-generated content may be incorrect.**DELETE /folders/[folderId]**

Sjekker tilganger som vanlig på starten av koden. Også sender vi en oppgave til Folders klassen som da håndterer sletting av mappen.

Klassen folders sletter ikke bare mappen men alle verktøy som ligger inni den. Bildene som er lagt ved verktøyene blir også slettet for å ikke beholde på irrelevant data som ikke trengs lengre. Gjør også tjenesten mye mere effektiv og billigere å kjøre.

A screenshot of a computer program

AI-generated content may be incorrect.**DELETE /tools/[toolId]**

Håndterer session og sjekker om brukeren er autentisert. Om brukeren er autentisert sjekker vi om at brukeren er en administrator. Om brukeren er en administrator sjekker vi om verktøyet dem prøver å slette er et som eksisterer. Om det eksisterer sender vi det inn til custom classen vi har laget som håndterer prisma callet og sletter bildet som verktøyet har.

Om alt går fint sender vi en sukse melding til brukeren.

Forms

A screen shot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.For forms har jeg valgt en modul som heter useForm siden den lar deg lett legge in zod schemas for form validation. useForm ligger bare på klient nivå så zod schemaet må sjekkes manuelt på backend.

A screen shot of a computer

AI-generated content may be incorrect.Jeg eksporterer en modul i samme directory som modalen som eksponerer forskjellige form komponenter som trengs for å klistre alt sammen.

Her kan man se at jeg importerer gruppen av forskjellige komponenter fra useformen slik at jeg kan bruke den inni modalen som å registrere inputs. A computer screen shot of a program code

AI-generated content may be incorrect.

Hjelpemidler

Nextauth dokumentasjon:

<https://next-auth.js.org/providers/credentials>

Avfallsportalen.no (Mitt prosjekt hos iris)

<https://github.com/mathiasvoie/avfallsportalen.iris-salten.no>

Egetprosjekt med autentisering

<https://github.com/mathiasvoie/Playground>