Jeg har valgt at lave et udkast til et portfolio-website. Jeg fokuserer primært på mulighederne i Strapi CMS i stedet for at fylde websitet med indhold. Noter at mit design også virker på mobilen!

Min kode kan ses og hentes fra github:

* Frontend: <https://github.com/RoWaBo/my-strapi-test-frontend.git>
* Strapi backend: <https://github.com/RoWaBo/my-strapi-test.git>

Min techstack:

* Nextjs – frontend framework
* Typescript – tilføjer syntax for typer i js. Gør JS strongly typed.
* Tailwind – CSS framework med classes

**Users, roles and auth in Strapi**

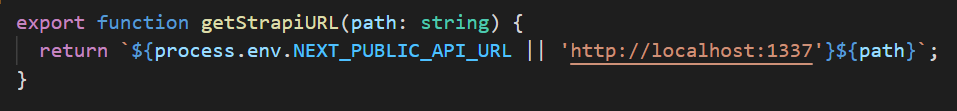
Da min side kun skal bruges af portfolio-websitet, undersøgte jeg hvordan man kan styre burgere og deres rettigheder i Strapi. Mit API kan kun tilgåes hvis brugerens rolle er authenticated. Dette satte jeg op i Strapi adminpanelet og tilføjede en bruger med username, email og password.

I min frontend håndtere jeg auth i en asynkron funktion som kun kører på serveren (dette er en del af nextjs). Det handler om performance og om at holde env’s ude af client koden. I denne funktion kører jeg et axios post kald med username og password og får returneret en jwt token som jeg gemmer i en const kaldet auth.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Derudover har jeg lavet en funktion som henviser til den rigtige url ligegyldigt hvilket miljø der opereres i. Oprindeligt havde jeg tænkt mig at deploye Strapi til Heroku, men nåede det ikke.



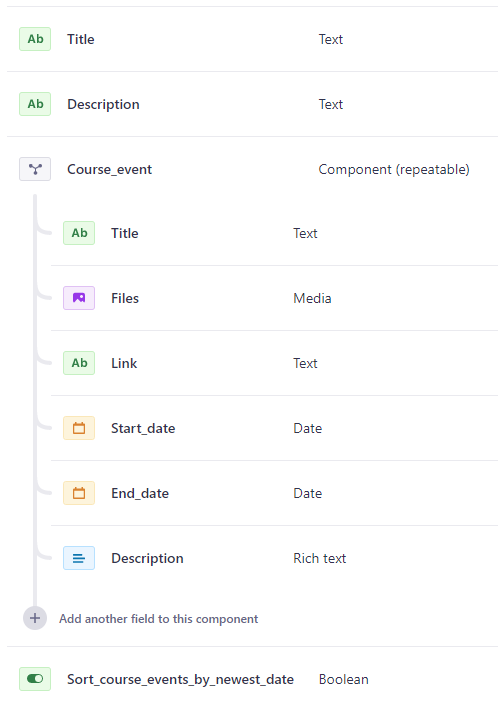
**Strapi query funktionen**

I Strapi dokumentation anbefaler de at bruge en npm-pakke som hedder qs. Den understøtter alle Strapis query funktioner såsom søgning, populating og filtrering. Vi havde tidligere problemer i klasse med at populate media data, men dette løser denne funktion. Der står at det handler om performance, så man skal spørge specifik til den data man vil populate. Jeg bruger funktionen til kun at hente den data jeg skal bruge med fields og populate.

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Tilsidst laver jeg et axios get kald med min query og jwt token og får kun ønsket data returneret.

**COURSE COLLECTION TYPE**

En type som bruges til at registrere kurser. Den består af en overskrift, en beskrivelse og et komponent der beskriver under-kursuser. Jeg har lavet det som et komponent så det nemt kan genbruges i en anden collection type. Da der kan være flere komponenter kan rækkefølgen derfor også ændres via drag n’ drag. Dette er smart, da det giver brugeren kontrol over rækkefølgen. Det bringer mig videre til min første feature der kan sortere kurser i frontenden. Det styres enten manuelt af brugeren via drag n’ drop eller via boolean feltet: ”Sort\_course\_events\_by\_newest\_date”. Hvis den er true vil kurserne blive sorteret med den nyeste dato først. Se funktion på næste side:

Datosorteringsfunktionen:

Et billede, der indeholder tekst, indendørs, skærmbillede

Automatisk genereret beskrivelse

Et billede, der indeholder tekst

Automatisk genereret beskrivelseDerudover har jeg også en funktion som omdanner datotallene fra strapi til måneder. Dette gør jeg med en switch der tjekker tallet og returnere den tilsvarende måned. (Hele funktion kan ikke ses på billedet. You get the point)

Sådan ser et kursus ud i browseren:

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

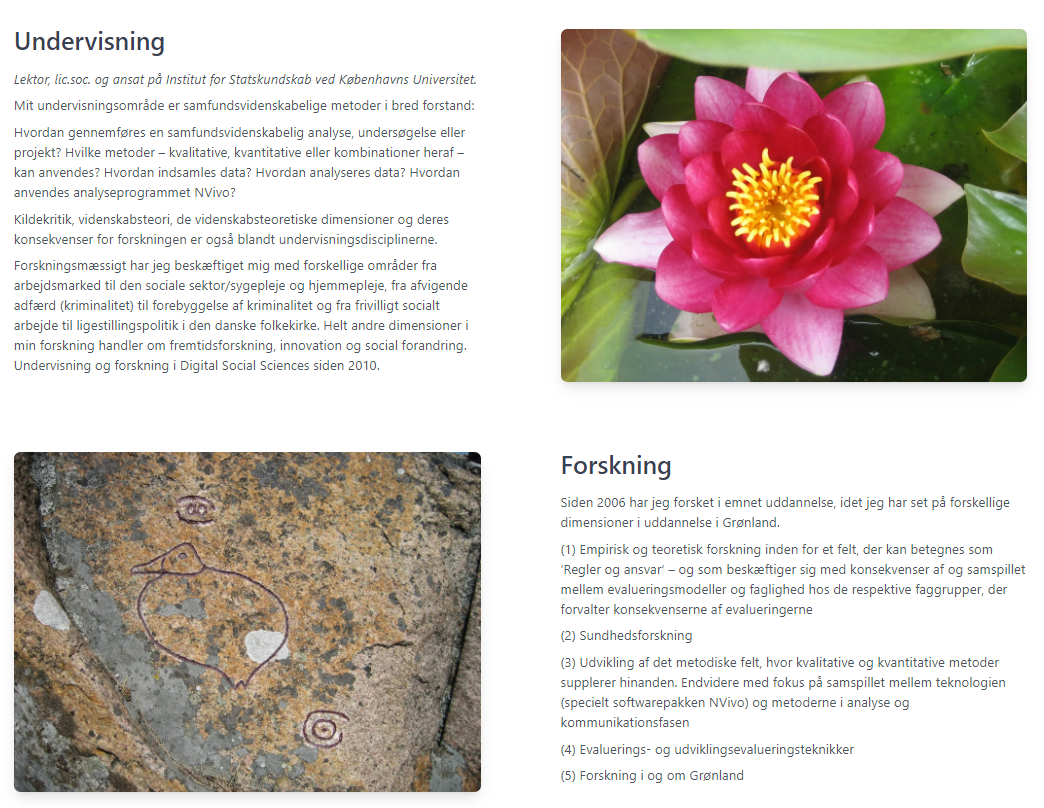
**Article collection type**

I min article collection type findes en overskrift, beskrivelse og et billede med alt-tekst og url.

Jeg bruger en funktion til at tjekke at den rigtige billede-url bliver returneret uanset miljø eller provider (cloudinary ect.).

Beskrivelsen i strapi bliver skrevet i markdown. Det gør det muligt for brugeren at style tekst og inkludere links og lister ect. Jeg bruger en npm-pakke der hedder ReactMarkdown til at konvertere markdown til html.

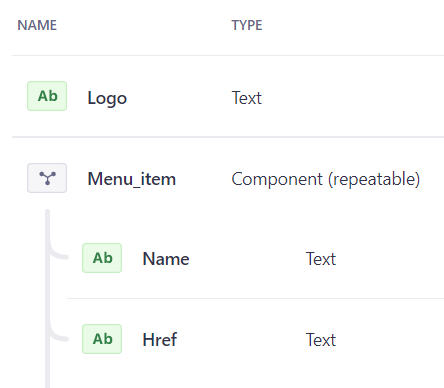
Jeg har lavet en feature funktion som sætter hver anden artikels billede til modsatte side. Det ser pænere ud når der er flere artikler under hinanden:



Funktionen tjekker artiklens index tal og hvis det tal er ulige returnere jeg flex-row-reverse klassen:

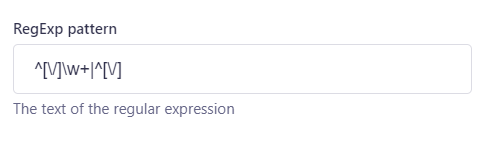
Et billede, der indeholder tekst

Automatisk genereret beskrivelse

**Navigation single type**

Jeg har forsøgt at gøre indhold på siden mulig for brugeren at ændre, så navigation måtte naturligvis følge med. Det er muligt at tilføje et logo og en menu item som har et name og en href.

Da hreffet skal skrives på en bestemt måde for at virke i min frontend har jeg tilføjet et regular expression der skal matche før at Strapi tillader den værdi.



I min frotend looper jeg over mine menu items og rendere dem på siden:

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Sådan ser navigation ud i browseren:

