[**Indledning (Mads)**](#_8zjd6ipacydb) **2**

[Kort resume (AK)](#_umznt9qy5o26) 2

[**Problemformulering (Alle har været inde over i fællesskab)**](#_bmazap2h3nhk) **3**

[Problemstillinger](#_boo4b2bddbzx) 3

[**Afgrænsning (Mads)**](#_yt25ltq3nxwn) **3**

[**Metodeovervejelser (Mads)**](#_vty6zuskzp9s) **3**

[**Research (Mads)**](#_x1wo25o4jt5y) **4**

[EJS (Hanna)](#_wvlr0viupnd3) 4

[**Analyse (AK & Mads)**](#_sc0w587orl9o) **5**

[**Konstruktion (Alle har været inde over i fællesskab)**](#_inek19a8n85v) **5**

[Express (Sune)](#_6d6v5xrfytku) 5

[Ajax (AK)](#_45kfqf1pjfgv) 5

[Includes (AK)](#_xjvvplmqt8m) 6

[Mongoose Schemas (Hanna)](#_h1err1jhhem5) 6

[Handlers (Sune)](#_p0i99rmhg7rr) 6

[Menu](#_liuxth8gmiwa) 7

[**Evaluering**](#_8n28c6wwjhm4) **7**

[Proces (AK)](#_wbg8uaxvah0u) 7

[Udfordringer (Alle har været inde over)](#_aje2k6oxe0ng) 8

[**Konklusion (Alle har været inde over i fællesskab)**](#_13i5amcvjkql) **10**

[**Referencer**](#_j6p7anc5mwbh) **11**

# 

# Indledning (Mads)

I en database er det vigtigt at kunne tilgå den data som databasen indeholder. Dette kan gøres ved hjælp af CRUD princippet (Create, Retrieve, Update og Delete).   
I vores projekt tager vi udgangspunkt fra en NoSQL database lavet i mongoDB hvor vi kan hente data omkring forskellige kontinenter og byer. Til det har vi lavet en hjemmeside hvor det er muligt at se og redigere det data som vi har i vores database.

## Kort resume (AK)

I denne rapport vil vi komme ind på metodeovervejelser omkring platform og framework, research af EJS som template engine, analyse, konstruktion og opbygning af vores hjemmeside. Til sidst vil der være en evaluering af vores proces og en konklusion.

# Problemformulering (Alle har været inde over i fællesskab)

Hvordan kan vi udvikle et website i node.js med express og mongoDB, hvis sider viser dynamisk data ved hjælp af AJaX og henter data fra vores world database ved hjælp af mongoose.

## Problemstillinger

* Hvordan konvertere vi fra mongo db kald over til mongoose db kald?
* Hvordan kan vi via EJS eller Pug, skabe en dynamisk hjemmeside, som henter data fra mongoDB

# Afgrænsning (Mads)

Grundet den givne situation med Covid-19 har vi har valgt at bruge det meste af vores tid på at kode vores hjemmeside og få kommunikeret sammen så vi har været enige omkring de forskellige opgaver i projektet. Dette har gjort at vi har afgrænset vores tid til research omkring mongoose, EJS og andre mulighed for implementering af egne modeller på vores hjemmeside. Vi har blandt andet fravalgt at bruge et chart som viser information omkring de forskellige lande på forsiden da disse data vil blive vist i andre menuer. I stedet har vi valgt at lave verdenskort hvor man kan klikke sig ind på de forskellige kontinenter hvorefter man får en liste frem der viser info omkring de forskellige lande på det kontinent man har valgt.

# Metodeovervejelser (Mads)

Efter forskriftet fra opgave oplægget har vi gjort brug af node.js som platform og express som framework. Dertil har vi valgt at bruge ejs som template engine og fravalgt pug da vi som gruppe har mere erfaring med ejs.

# Research (Mads)

I forbindelse med Covid-19 har mange nyhedsmedier gjort brug af [world chart maps](https://www.datawrapper.de/_/RjSrj/) til at vise virus smitte og andet relevant info. Via henvisningerne til disse charts, har vi fundet hjemmesiden ([datawrapper](https://www.datawrapper.de/)) som vi har brugt til at lave vores egne versioner med. Dataen i vores egne versioner er hentet fra <http://dkexit.eu/webdev/downloads/world.zip>. (Et eksempel kan ses her: <https://datawrapper.dwcdn.net/AqXgz/1/>) Vi har valgt ikke at gøre brug af disse charts på vores website da vi har prioriteret andre opgaver og manglende viden omkring implementering af charts sammen med vores egen kode.

## EJS (Hanna)

Efter at have fundet frem til hvilken database vi skulle bruge, gik vi i gang med at kigge på hvordan vi skal opbygge vores website. Vi blev enig om i gruppen, at gøre brug af templating engine.

Express giver mulighed for at arbejde med andre package og værktøjer.

Templating engine er noget som express,js bruger for at kan omdanne vores views til et læsbart html side. Valget af ejs kommer i takt med det, også er den engine vi er bedst kendte med. Derudover ligger dens syntax meget op af html, så man ville hurtigere kunne lære denne.

Vores start fil skal repræsentere vores index og med den og vores undersider er målet at lave en html side. Templating engine vil omdanne det dynamiske content tin en statisk html nodes og elementer. Det ville sige alt javascript variabler eller diverse beregninger med ejs tags vil blive omdannet. Og slut resultatet ville være en html side, som diverse browser kan læse og vise korrekt til brugerne.

# Analyse (AK & Mads)

Som tidligere nævnt har vi har valgt at bruge EJS som template engine fremfor Pug da det er den view engine de fleste af os har arbejdet mest med, og derfor føler os mest komfortabel med. Derudover ligner syntaksen HTML, hvilket gør det nemmere for os at arbejde med, da vi ikke skal til at lære en hel ny syntaks, som ses ved for eksempel Pug.

På grund af DRY (Don’t repeat yourself) princippet og for at gøre det nemmere for os selv har vi gjort brug af blocks i vores kode. I vores views mappe har lavet en includes mappe med alle blocks. (F.eks. menu og source scripts som vi har inkluderet på alle sider)

# Konstruktion (Alle har været inde over i fællesskab)

Hele konstruktionen af hjemmesiden er lavet via node.js, Express & Ajax. Der har dog været en del udfordringer på vejen. Vi har valgt at benytte os af disse ting, da vi gerne ville have en asynkron og dynamisk hjemmeside.

## Express (Sune)

Vi har i vores opgave valgt at gøre brug af et Express framework, en af årsagerne til dette er for at man kan “få den samme struktur” over flere forskellige projekter. Vi har ud fra express valgt at arbejde med ejs. strukturen. Ejs mappestrukturen osv. har vi så genereret via express. Express er i bund og grund en minimal, open source, flexible måde at arbejde med kode på.

## Ajax (AK)

Ajax bliver brugt til at udveksle data fra vores world database, mellem klienten og serveren. På den måde gør vi vores hjemmeside mere interaktiv og dynamisk, når der skal vises data frem på siderne.

## Includes (AK)

Ejs har muligheden for at inkludere f.eks en menu på alle siderne, hvilket gør arbejdet lidt nemmere. Dette gøres ved at lave en include fil, som i vores tilfælde er inkluderet i toppen af alle vores sider.

## Mongoose Schemas (Hanna)

Ved hjælp af mongoose package, som er et værktøj der giver et syntax lag mellem ens applikation og databasen. Igennem mongoose gør det også at vi kan omdanne vores data, sådan det tilpasser sig en model struktur. Model struktur er del vi har gjort meget brug af i vores opgave, sådan vi kan hente data fra de forskellige steder. Ved at bruge schema kan man sige vi ligger et wireframe, for hvordan vi ville have vores data skal struktureres. I forhold til de forskellige objects der i vores applikation. Schemas er bygget på key:properties, keys er de ting som der defineret i vores database på forhånd. Selve properties er hvordan de vises, dette er en del af selve jsons schemes

## Handlers (Sune)

Handlers har vi valgt at gøre brug af pga. at vi syntes det er nemmere at dele de forskellige database kald op. Handlers bruger vi til at fortælle om de enkelte ting der sker i databasen skal være Create, update eller delete. samt det er dem som efterspørger sorteringen på vores prototype. I vores handlers har vi valgt at gøre brug af de mongoose kald i mongoosewrap(Retrieve, Upsert & Remove), som vi har lavet i lektionerne med niels. I handlers har vi valgt at lave en if til at sortere, som er lavet sådan at vis dataen der bliver trukket ud og sort er = null. Så skal den sortere dem efter navne, her efter har vi givet alle vores funktioner en try {} som indeholder selve kaldet til mongodb databasen. Dette bruges så i handlers til at lave forskellige funktioner som bliver eksporteret til routeren(index.js)

## Menu

I alt har vi 5 menupunkter på vores hjemmeside. World Map, Create, Country Data, City Data og Language.

På vores Create side får man mulighed for at tilføje data til hver collection i vores database. Det er kun muligt at tilføje data til en collection af gangen hvilket er repræsenteret af en “Create” knap efter hver indtastnings afsnit.

På vores Country Data side har man mulighed for at vælge et land i en dropdown menu hvor man derefter får vist information omkring det land man har valgt. Vi har valgt at gøre brug af tabels til visning af data.  
Samme fremgangsmåde har vi gjort brug af på vores City data side med information omkring de forskellige byer i et land. Her får man en ekstra dropdown hvor man har mulighed for at vælge en by efter man har valgt et land i den første dropdown menu.

Som en ekstra funktion har vi på vores startside lavet et worldmap i SVG format hvor de forskellige kontinenter er et link videre til en side med en liste til hvert kontinents lande.

# Evaluering

## Proces (AK)

På grund af den nuværende verdenssituation, og for vores eget og andres helbred skyld, valgte vi ikke at mødes i virkeligheden. Vi har i stedet brugt programmet Discord, som vores kommunikationsform. Her holdt vi møde hver dag hvor vi har haft mulighed for at kunne snakke og kode sammen live. På den måde kunne vi alle følge med og samarbejde om en løsning på problemerne.

Vi har som det første lavet en tidsplan med mål for hvad vi skulle nå, og hvornår vi skulle være færdig med opgaverne. Vi har arbejdet med en opgave ad gangen og fokuseret på at få en ting til at virke, før vi er gået videre til næste opgave. På den måde ender man ikke med ufærdige opgaver som kun virker halvt.

Vi har brugt Git som versionsstyringssystem til vedligeholdelse af vores kode, hvor vi har haft et fælles repository, som vi alle har bidraget til.

Hele situationen med Covid-19 og muligheden for at mødes i virkeligheden, har selvfølgelig været skyld i nogle udfordringer under projektet, som har resulteret i de mangler der er i vores kode. Situationen har også været skyld i nogle komplikationer i forhold til kommunikationen ved vejledning, hvilket betød at vi sad fast med en opgave. Alt dette har resulteret i en del frustrationer i gruppen og komplikationer i forhold til vores kode, og vi er derfor ikke kommet helt i mål med projektet.

## Udfordringer (Alle har været inde over)

Vores oprindelig idé var at, når der vælges et af de 7 kontinenter, som er vist på forsiden, vil man derefter blive dirigeret over på siden Country Data. Her skulle der være en dropdown menu, indeholdende landene i det valgte kontinent, hvorefter det så skulle være muligt at få vist data tilhørende det valgte land og derefter for en by. Vi var nødt til at droppe ideen, da vi fandt ud af at funktionerne er afhængige af hinanden før de virker. Vi har istedet valgt at lave en separat side til Country, Cities og Language, med hver deres script som nærmest er identiske. På alle siderne starter man med at vælge et kontinent og derefter et land.

Næste skridt var at få vist noget data frem på siderne. Den største udfordring her var routing og hvordan dataen skulle vises ved hjælp af Ajax. Vi forsøgte først at få det til at virke ordentligt på Country Data, herefter var det simpelt at implementere på City Data og Language, da det vises på samme måde på alle tre sider.

Eftersom vi har arbejdet med create delen i tidliger opgaver har der ikke været mange udfordringer med denne del. Den del af koden som der gav mest problemer var at gøre “select options” i en “form” afhængig af data fra en bestemt collection i mongoDB sammen med EJS. Sagt på en simplere måde: “Få data fra vores database til at spille sammen med resten af koden”

Da det kom til slettefunktion havde vi nogle problemer. Som start prøvede vi en masse forskellige af, men kunne finde en sammenhæng imellem de forskellige dele. Til sidste kom frem til at vi skulle tilføje vores slet knap til tabellen, hvor vores data kommer ud. Det blev gjort igennem en delCountries i vores handler, her skulle vi tage vores getCountries funktion og rette den om. Således at den tilpassede at den skulle hente på selve land navnet. Her opstod der nogen problemer, da man ikke havde fjernet sort dele fra funktion. Efterfølgende skulle vi lave en knap, som hang ved tabellen til når dataene blev vist. Knappen fik ved ved hængt, men efterfølgende skulle den også kunne slette. Her kom vi frem til at lave en form og et input, sådan at vi har mulighed for at trække på noget data. Vi brugte POST method, til at få vores slet funktion til at virke. I selve routeren var vi laver vores post havde vi også problemer, med at finde de helt rigtige parameter at indsætte i funktionen.

# 

# Konklusion (Alle har været inde over i fællesskab)

Vi har i forbindelse med denne opgave konkluderet at under omstændighederne med corona, og denne form for arbejdsmetode med at skulle gøre alt online, har vanskeliggjort arbejdsprocessen for os som gruppe. Det har krævet ekstra disciplin og ekstra overskud fra gruppen. Derfor har vi valgt at gøre brug af Express framework til node.js for at gøre det hele med at arbejde online nemmere for os selv. hvilket giver os mulighed for at kalde sider igennem routeren. Ejs er som nævnt tidligere et templating engine. Denne engine har vi brugt i forhold til at opbygge vores views. Ejs gør at vi kan undgå, at skulle gentage os selv. Ved hjælp af vores include mappe, kan vi linke til de forskellige sider, til den givne side der skal fremvises. Dette køres så igennem en router som redirecter til de respektive sider og funktioner der er behov for. I vores handlers har vi de givne funktioner, som der skal udføres på vores side eller undersider. Ved hjælp af vores router, kan vi kalde disse funktion, sådan de kan loades på den givne side. Ajax bruger vi herefter til at konverterer data fra henholdsvis vores database og server, den mulighed gøre vi kan skabe en dynamisk hjemmeside. Hvorpå den går ind og trækker data ud fra vores world database.Ved hjælp af mongoose schemas, har vi mulighed for at trække data ud på de forskellige collections i vores mongoDB.

# 

# Referencer

Template engine:

[*https://ejs.co/*](https://ejs.co/)

[*https://alligator.io/nodejs/express-template-engines/*](https://alligator.io/nodejs/express-template-engines/)

Generelt:

[*http://dkexit.eu/webdev/site/pt02.xhtml*](http://dkexit.eu/webdev/site/pt02.xhtml)

[*https://www.datawrapper.de/*](https://www.datawrapper.de/) *(*<https://datawrapper.dwcdn.net/AqXgz/1/>*)*

*Get programming with Node.Js*