World

Anders Dall
Simon Lervad
Tinna María Richter
Rikke Domino Isaksen

Erhvervsakademi Kolding, IBA
Webudvikling, 2. semester

Niels Müller Larsen
Mile Ninkovic

Indholdfortengelse

Indledning 3
Problemformulering 3
Metodeovervejelser 3
    Agilt & plandrevet - Tinna 3
    GIT - Rikke 6
    Wireframe - Tinna 7
    Arbejdsproces - Anders & Tinna 7
Research 9
    Node.js - Rikke 9
    MongoDB - Simon 9
    AJAX & API - Rikke 9
Analyse 11
    Express Pug vs Dust - Anders 11
    Mongoose - Simon 13
    Opbygning af site - Simon 13
Konstruktion 14
    POST vs GET - Rikke 14
    Mongoose, CRUD - Simon 18
Evaluering af proces og produkt 20
Konklusion 22
Litteraturliste 23

Indledning

Der skal laves en dynamisk side, som har informationer omkring jordens verdensdele. For at gøre siden dynamisk bliver der lavet en database, der indeholder informationer omkring kontinenter, lande, byer, sprog og styreform. Siden skal kunne hente data fra databasen og gøre det muligt for brugerne at oprette, tilføje og slette informationer. Databasen er koblet sammen med en server, hvilket gør det muligt for brugerne at se indholdet. Projektet er udarbejdet gennem Node.js med frameworket Express, hvor den anvendte database er MongoDB med modulet Mongoose.

Problemformulering

Hvordan kan der laves en hjemmeside, der ved hjælp af Node.js og Express, kan vedligeholde en MongoDB database med modulet Mongoose?
Databasen skal kunne udføre CRUD på alle de kollektioner den indeholder.

Metodeovervejelser

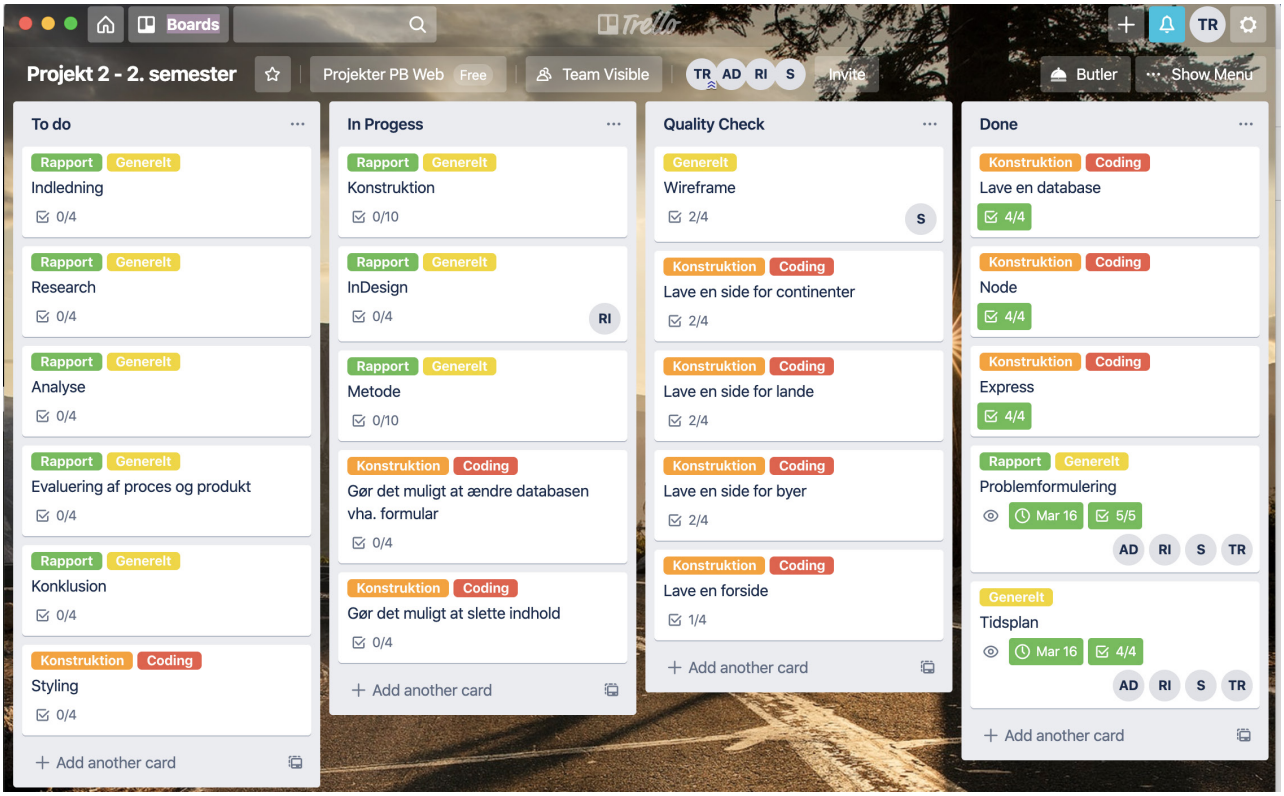
Agilt & plandrevet - Tinna

For at skabe en oversigt over hvilke opgaver der skulle laves i projektet og tidsestimere hver enkelte del, er der anvendt Gantt kort. I Gantt kortet blev der lavet oversigt over dagene med markeringer om, hvornår de enkelte opgaver skulle arbejdes med, samt hvilke tidspunkter Skype møderne/stand up møderne skulle afholdes. Opgaverne blev delt ind i følgende kategorier:

- Generelt
- Research
- Konstruktion

Opgaver	Fredag	Lørdag	Søndag	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
Mødes								
Anders								
Simon								
Tinna								
Rikke								
Arbejder på kode		(Rikke)	(Rikke)	Rikke	Simon	Anders	Tinna	
Skypemøde	kl 13.00			kl 8.30	kl 8.30	kl 8.30	kl 8.30	kl 8.30
Generelt								
Indledning								
Problemformulering	Alle							
Wireframe								
Metode								
Research								
Analyse								
Konstruktion								
Evaluering af proces og produkt								
Konklusion								
Referencer								
InDesign								
Programmering								
Research								
Brugervenlighed								
Mongoose								
Konstruktion								
Forside								
Lave en side med alle informationer								
Lave en side for continenter								
Lave en side for lande								
Lave en side for byer								
Gør det muligt at tilføje info								
Gør det muligt at slette indhold								
Gør det pænt								

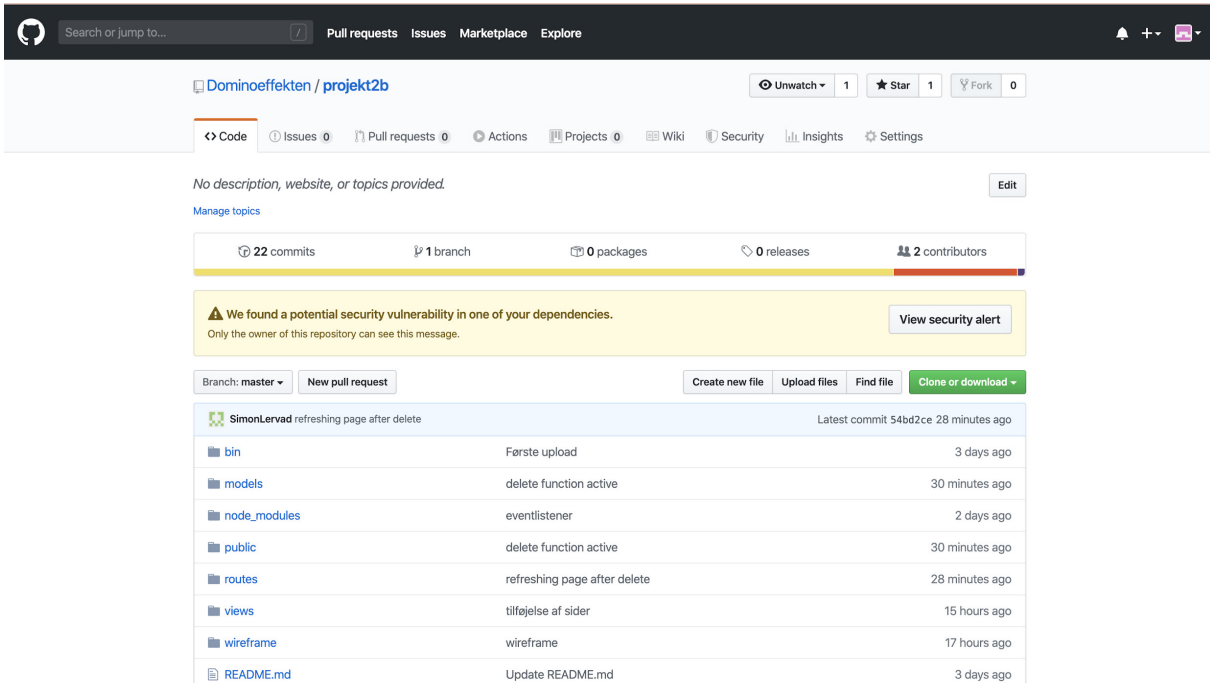
Derudover blev der anvendt Trello for at få et overblik over, hvilke opgaver der skulle laves, hvilke der blev arbejdet på, hvilke der skulle tjekkes igennem for eventuelle rettelser, samt hvilke der var færdige. Dertil blev der anført, hvilken kategori hver opgave hørte ind under med labels. Derfra kunne der sættes deadlines på opgaverne og tilføjes gruppemedlemmer, som skulle arbejde på given opgave.
Trello er blevet anvendt som en slags Product Backlog (SCRUM), hvor der er et board med tilhørende opgaver.



Ved at anvende disse værktøjer blev arbejdsprocessen både plandrevet og agil. Plandreven i den forstand at der blev udarbejdet og visualiseret en plan over projektets forløb. Agilt i den forstand at opgaverne skulle tjekkes igennem, før de blev vurderet færdige, og derfor blev de taget op igen og arbejdet med indtil, de kunne vurderes færdige og var blevet godkendt af alle gruppemedlemmerne.

GIT - Rikke

Udover at anvende Trello og Gantt kort, er Git også anvendt. Git er et moderne versionsstyringssystem, som bruges til programmeringsprojekter, hvor det ikke er nødvendigt at være geografisk nær hinanden. Ved at anvende Git kunne flere af gruppens medlemmer arbejde samtidig med koden - dog ikke med den samme fil. Git fokuserer på filens indhold, hvilket gør det muligt at se kodens udvikling, derudover er den fleksibel, da Git gør det muligt at lave branches, hvor koden kan testes uden at påvirke koden, som er der i forvejen. For at gruppens medlemmer kunne arbejde sammen, blev der lavet et repo på Github, hvor gruppemedlemmerne blev tilføjet, som samarbejdspartner. Det gjorde, at de individuelle medlemmer fik samme rettigheder til repoet (Atlassian n.d.)



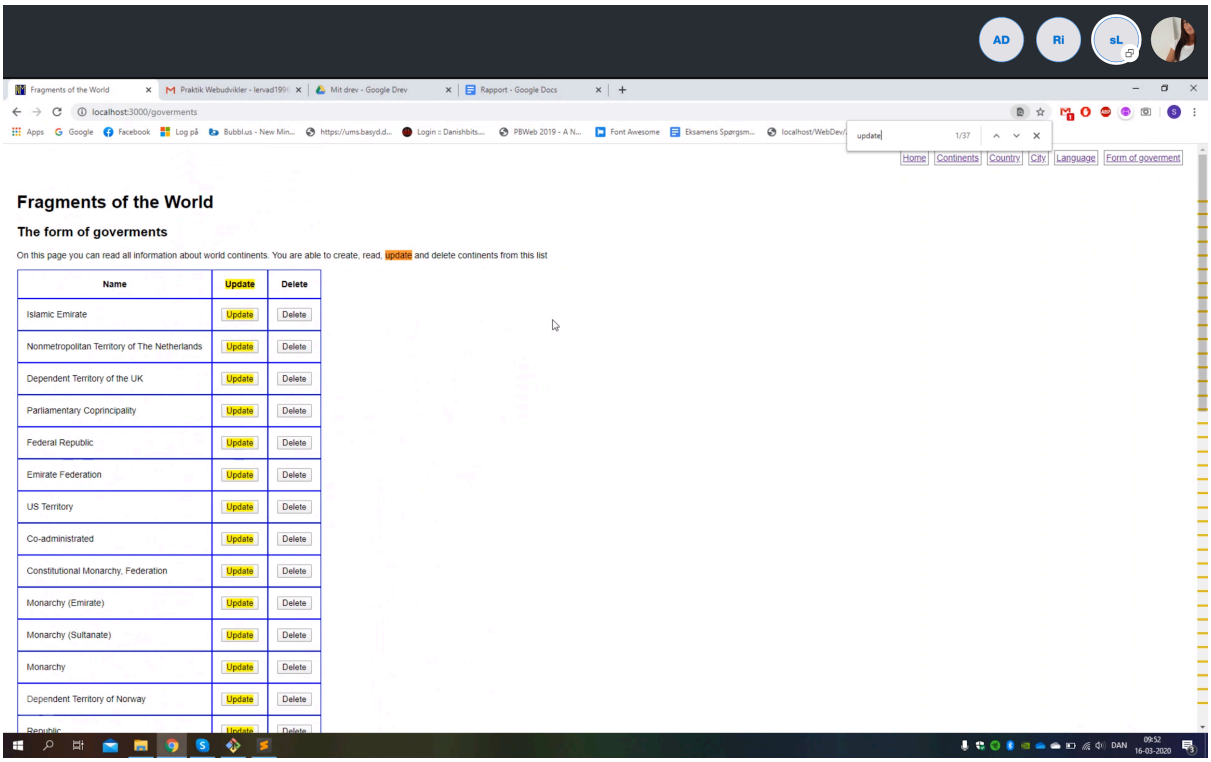
Wireframe - Tinna

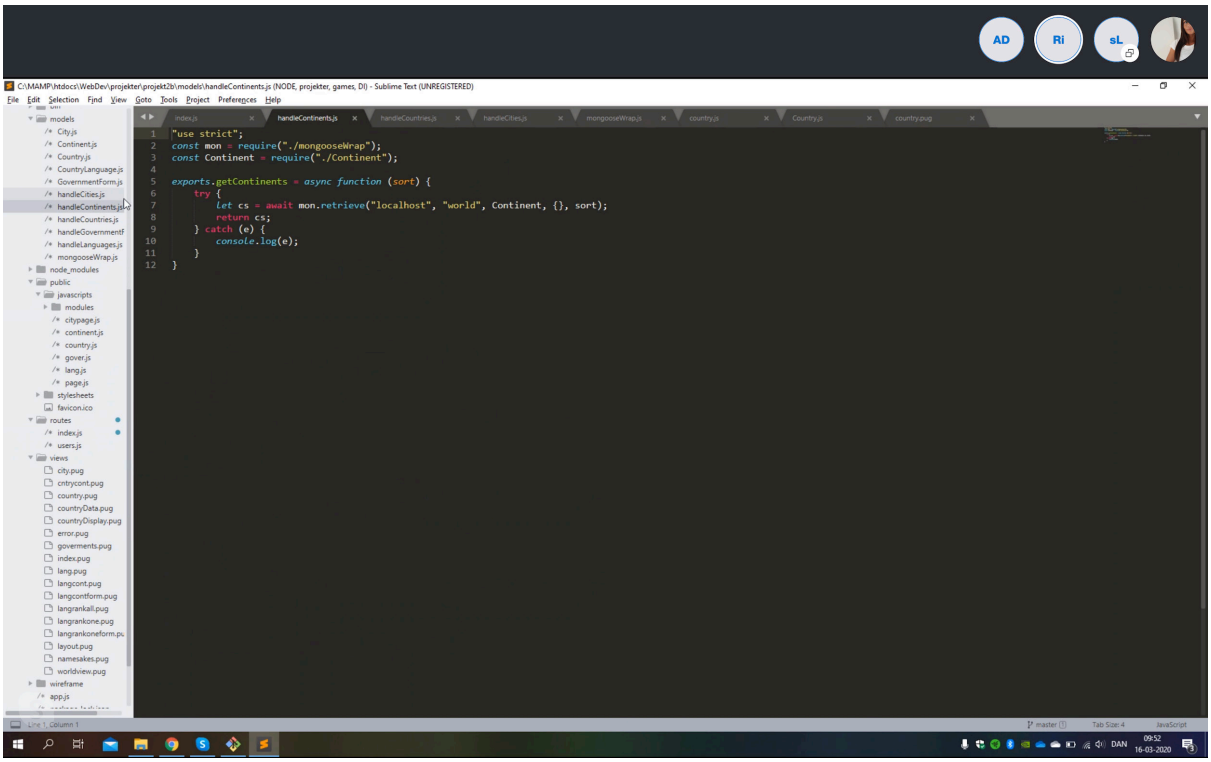
Da ideen om udarbejdelse af koden havde fundet sted, blev der lavet en wireframe for at visualisere hjemmesidens opbygning. Hvilket gav et indtryk af, hvor elementerne skulle placeres samt et helhedsindtryk af hjemmesiden. Wireframen blev holdt i low fidelity, da der udelukkende blev fokuseret på opbygning og funktionalitet, men ikke designet.

Arbejdsproces - Anders & Tinna

Tilgangen til projektet har været anderledes, sammenlignet med tidligere gruppeprojekter. På grund af situationen omkring corona virussen, har gruppen ikke haft mulighed for at mødes på skolen, men har udelukkende kommunikeret gennem daglige Skype møder og Messenger på Facebook.

Skype er blevet anvendt som kommunikationsmiddel, hvor der også er blevet benyttet skærmdeling for at gennemgå koden og snakke om eventuelle problematikker, der er opstået. Derudover blev der debatteret omkring løsningsforslag og planlægningen af projektets forløb. Facebook Messenger blev anvendt til at planlægge Skype møderne, samt dele relevante links til projektet. Hvor Google Drive blev anvendt til at dele filer blandt gruppemedlemmerne, såsom rapporten, projektbeskrivelsen og Gantt kortet.


AD
Ri
sL
Fragments of the World
localhost:3000/goverments
update
1/37
Home
Continents
Country
City
Language
Form of goverment
Fragments of the World
The form of goverments
On this page you can read all information about world continents. You are able to create, read, update and delete continents from this list
Name
Update
Delete
Islamic Emirate
Nonmetropolitan Territory of The Netherlands
Dependent Territory of the UK
Parliamentary Coprincipality
Federal Republic
Emirate Federation
US Territory
Co-administrated
Constitutional Monarchy, Federation
Monarchy (Emirate)
Monarchy (Sultanate)
Monarchy
Dependent Territory of Norway
Republic


AD
Ri
sL
handleContinents.js
"use strict";
const mon = require("./mongooseWrap");
const Continent = require("./Continent");
exports.getContinents = async function (sort) {
try {
let cs = await mon.retrieve("localhost", "world", Continent, {}, sort);
return cs;
} catch (e) {
console.log(e);
}
}

Research

Node.js - Rikke

I forbindelse med projektet er der anvendt Node.js, som er et open source server miljø, der kan køre på adskillige platforme. For at få serveren til at virke, anvendes der JavaScript. Førhen blev JavaScript kun kørt i en browser, men ved hjælp af Node.js kan der nu laves webapplikationer uden en browser. Ofte bliver en webserver anvendt til at åbne en fil på en server og returnere indholdet. Ved at anvende Node.js bliver ventetiden formindsket. Node.js er asynkron, hvilket vil sige, at den kan lave flere opgaver på samme tid, uden at blokere de forskellige opgaver (W3School n.d.).

MongoDB - Simon

For at lave en dynamisk hjemmeside er der gjort brug af MongoDB, som bruges til at styre databasen. MongoDB er et gratis online værktøj, som kan bruges til at administrere DB med Node.js. I projektet er der en “dump” mappe, som indeholder en database bestående af 5 kollektioner. Hvis der bliver kørt en restore på denne mappe, oprettes der en database, som indeholder content for hele verdens kontinenter, lande, byer, sprog og statsformer.

AJAX & API - Rikke

For at få vist databasens indhold på siden er der blevet anvendt AJAX og API. AJAX er asynkron, og gør det muligt at læse data fra en webserver, efter at siden er loaded. Derudover kan den opdatere en side, uden at reloade, da AJAX sender data til webserveren i baggrunden. Ved at anvende AJAX kan der bruges browserens built-in XMLHttpRequest object, som beder om data fra en webserver, hvorefter der kan anvendes JavaScript og HTML DOM til at vise den pågældende data (AJAX Introduction n.d.).

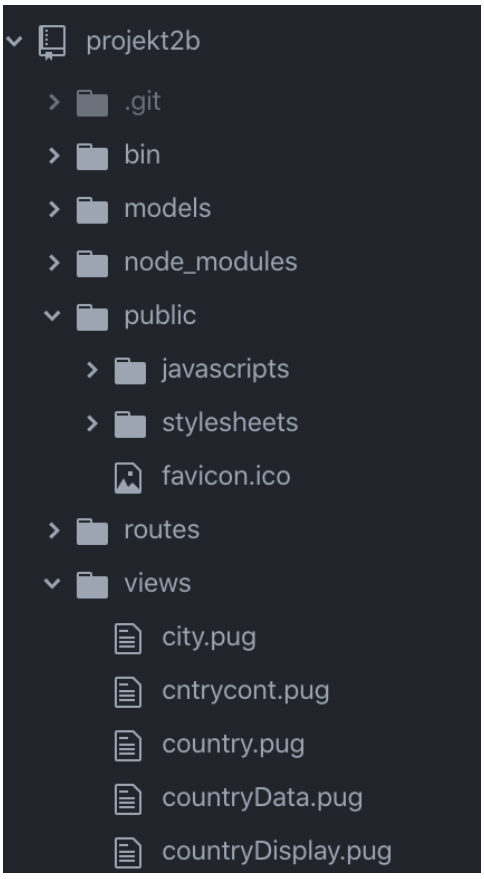
Application Programming Interfaces (API) giver muligheden for at gøre indviklede funktionaliteter mere ligetil, ved at lave en brugervenlig syntaks. JavaScript har mange API'er tilgængelige, det er dog ikke en del af selve JavaScript sproget, men det kan være med til at give JavaScript superkræfter, i den forstand, at der kan tilføjes noget ekstra på en nemmere måde. API kan anvendes til at manipulere dokumenter, som bliver loadet i browseren, men den kan også hente data fra en server, som kan opdatere små dele af hjemmesiden (Introduction to Web APIs n.d.).

Analyse

Express Pug vs Dust - Anders

Projektet anvender Express, som er et web framework i Node.js, der gør det nemmere at programmere en webapplikation. Det er et modul, der strukturerer applikationen. Fordelen er, at applikationen ikke skal bygges op fra bunden. Express installeres gennem pakken Express.js i Node.js.

Express gør det muligt at arbejde med templating engines, som har hver deres templating language. En templating engine er den måde, som Express arbejder med de filer, der ligger i views folderen og konverterer dem til HTML format, som kan læses af browseren. Gruppen har primært arbejdet med to forskellige templating engines i tidligere projekter, Dust.js og Pug.js. Der skulle tages stilling til, hvilken der skulle anvendes i projektet. Dust.js syntaks og semantik minder om HTML, hvor Pug.js minder om Pythons syntaks.



Den valgte templating engine er Pug.js, da den har nogle fordele i forhold til Dust.js. Pug.js har en layout.pug fil, hvor koden til head og header skrives en gang. På den måde undgår man at skrive den samme kode gentagne gange i forskellige filer.

layout.pug

```
doctype html
html
    head
        title= 'Fragments of the World'
        link(rel='stylesheet', href='/stylesheets/style.css')
        script(type='module' src=scriptLink)
    body
        header
            nav
                ul
                    li
                        a(href="/") Home
                    li
                        a(href="/worldview") Continents
                    li
                        a(href="/country") Country
                    li
                        a(href="/city") City
                    li
                        a(href="/lang") Language
                    li
                        a(href="/goverments") Form of goverment
```

De andre pug sider, der blev oprettet i views folderen, skulle starte med ”extends layout” for at påbegynde læsningen af layout.pug. Denne metode gør det nemmere at vedligeholde koden, da det kun skal skrives en gang (Wexler 2019).

namesakes.pug

```
extends layout
```

Syntaks og semantik i Pug.js adskiller sig fra HTML format, da den:

- ikke anvender tags < >
- ikke anvender closing tags </>
- skal bruge indentation
- id=”” erstattes med #
- class=”” erstattes med punktum

(Kraft and Chalkley n.d.)

Mongoose - Simon

Siden er blevet gjort dynamisk vha. udvidelsen til MongoDB, Mongoose. Denne udvidelse arbejder med JSON schemas, som hjælper med at oprette og vedligeholde kollektioner i databasen. Ved hjælp af disse schemas kan man fortælle serveren, hvilke oplysninger der skal vedligeholdes, eller hvad et objekt, som minimum, skal indeholde. På den måde sikres en mere ensartet struktur gennem databasen. Der følger nogle standardfunktioner til Mongoose, som blev anvendt i projektet. Heriblandt “DeleteOne”, “findOneAndUpdate” og “find”, som hjælper med at lave en global funktion, der kan slette, finde eller oprette/opdatere indhold i databasen.

Opbygning af site - Simon

I forbindelse med opbygningen af sitet og overvejelser i forhold til udseendet, blev der tegnet en low fidelity wireframe, som skulle give et overblik over, hvordan sitet skulle se ud, og hvordan de forskellige funktioner blev implementeret. Kort beskrevet blev der lavet et site med en menubar med tilhørende link til hver kollektion fra databasen. På hver side ville alt data fra den gældende kollektion blive fremvist i tabeller, på nær country siden, hvor der er blevet udvalgt få informationer om det enkelte land. For enden af hver “tr” i tabellen vil der blive fremvist to ikoner - Det ene for at opdatere informationerne og det andet for at slette dem. Hvis man anvender opdater-knappen, kan oplysningerne opdateres, hvor informationen for den valgte land/by bliver indsat og giver muligheden for at redigere det og efterfølgende sende det til databasen. Hvis krydset blev valgt, vil informationerne blive slettet fra databasen. På sider, som language og city, vil man øverst på siden kunne sortere efter kontinent og land for at få fremvist mindre data og gøre det mere overskueligt.

Konstruktion

POST vs GET - Rikke

Efter sidens opbygning er blevet dannet, kan kodningen påbegyndes, så der kan interageres med siden og dens indhold. For at man kunne tilgå siden, skulle siden tilføjes i routeren, så den vidste, at siden fandtes, og hvad den evt. skulle vise. Der er for hver side lavet en landingpage med metoden GET, hvor der blev anvendt res.render, som kunne tilpasses alt efter urlen. I res.render er der tilføjet, en overskrift, subtitle samt et JavaScript fil. Et eksempel er, at der på siden omkring Government form er tilføjet en subtitle, som fortæller, hvad siden handler om, samt en JavaScript fil, som indeholder sidens indhold. For at få databasens indhold ind på en side blev der lavet en ny url, hvor der er sorteret efter navn. På nogle sider er der tilføjet en post metode, hvilket er gjort, fordi der i nogle tilfælde bliver sendt data fra siden og over til serveren.

```
// government start
router.get('/goverments', async function(req, res, next) { // load the site
    res.render('goverments', {
        scriptLink:'/javascripts/gover.js',
        subtitle: 'The form of goverments'
    });
});
router.get('/gover', async function(req, res, next) {  // loads the db content
    //console.log(req.params.city)
    let gover = await modGover.getGovernmentForms({}, {sort: {name: 1}});
    res.json(gover);
});
router.post('/goverments', async function(req, res, next) { // deletes country from db
    let delGover = modGover.delGover({name: req.body.name});
    console.log("Yah du kom herind");
    res.render('goverments', {
        scriptLink:'/javascripts/gover.js',
        subtitle: 'The form of goverments'
    });
});
// government slut
```

I JavaScript filen, som viste sidens indhold, blev der linket til en API, som skal vise informationer fra sidens database. Dette blev gjort for at aflaste sidens server. API blev udført, alt efter sidens URL, i dette tilfælde "/gover". Et eksempel på government form siden, hvor siden skulle køre funktionen showGovernment med den pågældende URL.

```
/*
 * Event handler for button – create ajax object and get data
 */
const getGoverment = function (ev) { //country
    let req = Object.create(Ajax);
    req.init();
    req.getFile(`/gover`, showGoverment);
};
```

I funktionen, showGoverment, blev HTML DOM manipuleret for at skabe det ønskede resultat. Der blev her lavet en tabel med det pågældende indhold, samt et objekt hvor API'ens indhold befandt sig. I nogle tilfælde blev der linket til flere API'er på siden.

```
//Opret forbindelse til api continent indholdet
let goverments = JSON.parse(e.target.responseText);

let div = document.createElement("div");
let tabel = document.createElement("table");

let th1 = document.createElement('th');
    let name = document.createTextNode("Name");
let th2 = document.createElement('th');
    let del = document.createTextNode("Delete");

th1.appendChild(name);
th2.appendChild(del);
tabel.appendChild(th1);
tabel.appendChild(th2);
```

For at få adgang til de forskellige government forms blev der lavet en foreach funktion, så informationerne lettere kunne tilgås. De forskellige informationer blev indsat i en tabel, for at gøre siden overskuelig. Derudover blev der lavet en knap for at slette et objekt fra databasen. Der blev først lavet en knap, som skulle sende informationen tilbage til routeren, men efter flere forsøg, hvor det ikke kunne lykkes blev der udarbejdet en anden ide. Knappen blev sat ind i en form, som har fået metoden POST og et usynligt input felt med en værdi. Når der bliver trykket på knappen, bliver informationen sendt over til routeren gennem input værdien. Der blev valgt at lave et formelement med et usynligt input, for nemmere at sende input værdien tilbage til serveren med POST metoden.

```
goverments.forEach(function (goverment) {

    let tr = document.createElement('tr');

    let td = document.createElement('td');
        let name = document.createTextNode(goverment.name);

    let td2 = document.createElement('td');
        let form = document.createElement('form');
        form.setAttribute("method", "POST");
        form.setAttribute("action", "/goverments");

    let input = document.createElement('input');
    input.setAttribute("value", goverment.name);
    input.setAttribute("name", "name");
    input.setAttribute("type", "hidden");

    let delButton = document.createElement('button');
    let delI = document.createElement("I");
    delI.setAttribute("class", "fas fa-times");
    delButton.appendChild(delI);
```

I routeren ser den på metoden, som er ændret efter knappen. Derfor ved den nu, at den skal gøre noget andet.

```
router.post('/goverments', async function(req, res, next) { /
    let delGover = modGover.delGover({name: req.body.name});
    console.log("Yah du kom herind");
    res.render('goverments', {
        scriptLink:'/javascripts/gover.js',
        subtitle: 'The form of goverments'
    });
});
// government slut
```

I routeren får den fortalt, at input værdien skal sendes videre til en bestemt funktion med en bestemt værdi. Den skal sendes videre til en funktion, som sletter government elementet, med den pågældende værdi. Der var her udfordringer, da funktionen skulle slette elementer, men i stedet valgte at slette hele kollektionen. Derfor skulle den nu udelukkende søge efter et bestemt id, som MongoDB selv laver. Efter flere mislykkede forsøg, blev opgaven sendt videre til et andet gruppemedlem, som ændrede funktionen. I stedet for at søge efter et id, som ikke står i Schemaet, blev der ledt efter noget andet som var unikt. I forbindelse med government siden, var det navnet.

```
exports.remove = async function(url, dbn, obj, name) {
    const constr = `mongodb://${url}:27017/${dbn}`;
    await mongoose.connect(constr, conparam);
    const db = mongoose.connection;
    let stuff = null;
    try {
        stuff = await obj.deleteOne(name, (err) => {});
        console.log("Successful deletion");
    } catch(err) {
        console.log(error);
    } finally {
        db.close();
        return stuff;
    }
}
```

Mongoose, CRUD - Simon

Udover at kunne slette elementerne, skal de også opdateres og tilføjes til databasen. Der er blevet oprettet en formular, som indsætter nye lande ind i databasen. Denne formular anvender: “findOneAndUpdate” funktionen fra Mongoose. I formen er der to dropdown menuer, som henter data fra databasen for at gøre valgmulighederne mere dynamiske. Det er blevet gjort ved at lave tre API kald - Et til contient, government form og country. Continent og government form blev anvendt til dropdown menuer, hvor country blev anvendt til at fremvise data og sende data til databasen. Der blev lavet et kald, som indsætter en ny URL i headeren, hvilket betyder, at de tre kald ikke bliver lavet på samme tid, men kommer hver for sig.

Der var problemer med at få API’erne til at virke, som de skulle, uden at miste den data, som blev hentet fra API’en. Når siden blev loadet, skulle der fremvises dynamisk data, i samme form vha. 3 forskellige API’er. Problemet var, at dataen i en lokal funktion gjorde, at den lokale funktion ikke kunne anvendes uden for funktionen. Det blev løst ved at tildele “#id” til de nødvendige input fields, for at kunne kalde på dem igen senere i scriptet, når den rigtige API blev håndteret.

I forbindelse med at opdatere en given kollektion, er der blevet lavet en funktion, som læser alle data for et specifikt objekt i kollektionen. Hvis man f.eks. vælger at klikke på Danmark, vil den tage alle de informationer, som Danmark indeholder og sende til en side med POST. Den side vil derefter generere en form, som indeholder alle værdier givet for Danmark. Brugeren vil herefter få mulighed for at ændre og manipulere disse oplysninger. De afslutter ved at sende formen og opdatere dataen i objektet for Danmark og sende det til country kollektionen i world databasen.

Strukturen på denne form er opbygget i .pug, hvor der vha. en POST funktion bliver indlæst data for det land, brugeren har klikket på. Herefter placerer funktionen Danmark objektet i en variabel. Efterfølgende indlæser den alle kontinenter og government forms, da de bliver brugt til at generere mulighederne for dropdown i formen, på den måde bliver dataen dynamisk for brugeren. Til sidst i funktionen bliver der udført en res.render(); hvor vi indlæser en side, som genererer formen baseret på de informationer, den har fået. I det her tilfælde vil den generere en form for country, og indsætte alt information for det givne land, og sende det med ind på siden sammen med alle kontinenterne og government forms.

```
router.post('/countryRead', async function(req, res, next) {
    let result = await modCountry.getCountries({name: req.body.name}, {sort: {continent: 1}});
    let continent = await modContinent.getContinents({}, {sort: {continent: 1}});
    let governmentform = await modGover.getGovernmentForms({}, {sort: {continent: 1}});
    console.log(result);
    res.render('countryData', {
        title: "You are about to edit selected country: " + req.body.name,
        country: result,
        cont: continent,
        gov: governmentform
    });
});
```

Der har undervejs været problemer med at indlæse det land, som blev trykket på og få det sendt til en side. Vi løste problemet med ovenstående funktion ved at gemme alt informationen i objekter og sende det ind til siden, når den var genereret.

På nogle sider kan man ikke ændre eller tilføje data, hvilket er et bevidst valgt. Det skal ikke være muligt for brugeren at tilføje eller ændre kontinent, fordi der kun er 7 på verdensplan, hvilket der også er på siden. Derudover er der ikke en opdaterer knap på government siden, fordi der er så få informationer, at det er nemmere for brugeren at slette dem og skrive det ind på ny. Brugerne kan ændre data på siderne omkring lande og byer, men for at gøre det mere overskueligt for brugerne, kan de trykke på en opdaterer knap, hvor informationerne, omkring den pågældende land eller by, bliver sat ind i en form.

Evaluering af proces og produkt

Ved påbegyndelse af projektet var gruppen blevet enige om at oprette to former for tidsplaner. Hvor der blev anvendt Gantt og Trello. Valget er blevet taget, fordi det er anvendt i tidligere projekter, hvor det fungerede godt for arbejdsprocessen. Derudover er der altid mulighed for at se en oversigt over, hvilke opgaver der mangler at blive lavet, og hvilke der kan deles ud blandt gruppens medlemmer.

Pga. omstændighederne har gruppen skullet vænne sig til ikke at kunne mødes fysisk, derfor har det været en anden måde at arbejde og kommunikere på. Som nævnt tidligere i rapporten er der kommunikeret dagligt gennem Skype og messenger. Ved at anvende skærmdeling på Skype har man kunne gennemgå koden med hinanden og været fælles om at løse eventuelle problematikker, der er opstået i forløbet. Det har fungeret godt i den forstand, at når der ikke er mulighed for at mødes fysisk, så kan der stadigvæk blive kommunikeret samt dele idéer og problemløsninger over et digitalt medie. For at gemme og dele filer er der anvendt både Google Drev og GIT, hvilket fungerer fint, når man ikke kan mødes. Ved deling af filer på disse medier, bliver der sørget for at hele gruppen har alle de informationer, der er nødvendige for, at de kan være en del af projektet. Ved at anvende Google Drev og dens værktøjer er der adgang til at live redigere og opdatere de filer, der er i den pågældende mappe, hvor der også er mulighed for at skrive kommentarer ved de elementer, der eventuelt skal rettes til. Ved anvendelse af GIT bliver der sørget for, at alle gruppemedlemmer har adgang til et fælles repo, hvor alt kode bliver delt i. Ved at gøre dette har alle gruppens medlemmer adgang til den nyeste version af koden. Derfor har det også været vigtigt for os at kommunikere med hinanden for at sørge for, at der ikke bliver arbejdet i de samme filer samtidig og undgå merge konflikter.

Der har været lidt problemer med kommunikation mellem underviser og gruppen, da det er sværere at præcisere, hvilken hjælp der er brug for til de problemer, der opstår på skrift frem for at kunne forklare det direkte.
I forhold til konstruktionen af sitet var der nogle udfordringer med CRUD. I starten kunne man se, indsætte og slette indholdet på siden, men man kunne ikke opdatere indholdet. Når der opstod problemer, som fx dette. Gik opgaven igennem forskellige gruppemedlemmer, som hver prøvede sig frem, evt. ved at søge efter løsninger på Google.

Konklusion

For at lave en dynamisk hjemmeside, der kan vise informationer om kontinenter, lande, byer, sprog og styreform, er der lavet en database i MongoDB, som indeholder informationerne. I MongoDB er modulet Mongoose anvendt, da den anvender JSON Schemas, som sikrer at informationerne, der bliver sendt i databasen altid, står ens.
Selve hjemmesiden er oprettet ved brug af Express med templating engine Pug, som er en hjælp til at tilgå databasen og udskrive informationer på siden.
Hjemmesiden har på forsiden en menu med 5 undersider, der hver repræsenterer en kollektion fra databasen. På nogle af siderne kan man oprette, rette og slette informationer i databasen. Hjemmesiden bliver dynamisk på den måde, at man som bruger ikke møder en statisk side, men en side der har muligt for at ændre sig, enten ved at brugeren selv tilføjer eller sletter information, eller ved at ændre indholdet gennem valgte kontinent og lande.

Litteraturliste

AJAX Introduction (n.d.) available from <https://www.w3schools.com/js/js_ajax_intro.asp> [16 March 2020]

Atlassian (n.d.) What Is Git: Become a pro at Git with This Guide | Atlassian Git Tutorial [online] available from <https://www.atlassian.com/git/tutorials/what-is-git> [16 March 2020]

Introduction to Web APIs (n.d.) available from <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction> [20 March 2020]

Kraft, J. and Chalkley, A. (n.d.) What Is Pug? [online] available from <https://teamtreehouse.com/library/what-is-pug> [20 March 2020]

W3School (n.d.) Node.Js Introduction [online] available from <https://www.w3schools.com/nodejs/nodejs_intro.asp> [17 February 2020]

Wexler, Y. (2019) Get Programming With Node.Js. Shelter Island, NY: Manning Publications Co