Indhold

Indledning og kort resume

Indledning

Formålet med opgaven er at, udvikle et website med dynamiske sider og menuvalg, hvis funktion er vedligeholdelse af en MongoDB database. Kravet til den data som databasen skal indeholde er tre collections med data over: lande fra hver af de 7 kontinenter, sprog og byer fra et land som allerede er oprettet i databasen.

Dataene skal kunne vises, oprettes og redigeres ved hjælp af Node.js, JavaScript og moduler.

Kort resume

I rapporten vil vi komme omkring punkterne metodeovervejelser, research, vores arbejdsproces, analyse og konstruktion af kode og website. Til sidst vil der være en konklusion, hvor vi laver en opsamling på hele opgaven og får svar på vores problemformulering og -stillinger, samt om kravet til opgaven er opfyldt.

Problemformulering

Hvordan kan vi lave et dynamisk website ved hjælp at Node.js og MongoDB, indeholdende global information, med fokus på funktionalitet som display og vedligeholdelse af data.

Problemstillinger

- Hvor finder vi vores data.
- Hvordan opbevarer vi den indsamlet data.
- Hvordan får vi data fremvist på siden.
- Hvordan kan vi oprette/redigere data.

Metodeovervejelser

Som start er vi nødt til at få fat i noget data, som vi kan bruge på vores website. Vi har brugt kvantitativ metode til indsamling af data. Vi har ikke selv været ude at måle og finde vores data. Vi har brugt Wikipedia[1] og The World Factbook[2] som vores kilder, hvor alle de nødvendige oplysninger allerede findes.

The World Factbook indeholder oplysninger om blandt andet om mennesker, samfund, regering og demografi.[3] Vi har primært brugt The World Factbook, hvor vi har brugt Wikipedia som en slags opbakning under vores dataindsamling.
Vi tænker at The World Factbook er en pålidelig kilde, da man kan se at dataene bliver holdt opdateret løbende.

Til opbevaring af den indsamlet data bruger vi MongoDB[4], hvor vi opretter en database, i dette tilfælde er den kaldt *world*, og tre collections, som indeholder data over lande, sprog og by.

Ved hjælp af Node.js kan vi bruge vores data fra databasen til at, lave et dynamisk infosite hvor dataen kan vises, oprettes eller redigeres.

[1] https://www.wikipedia.org/
[2] https://www.cia.gov/library/publications/resources/the-world-factbook/
[3] https://www.cia.gov/library/publications/resources/the-world-factbook/
[4] https://www.mongodb.com/

Research

Som nævnt tidligere tidligere, har vi arbejde med den kvantitativ metode. Det eneste data vi har fået opgivet på forhånd er, de syv verdens kontinenter. Ud fra dette skal vi så finde mere data. Vi skal finde byer, sprog og land, sådan vi kan få dækket alle kontinenterne. Vi nævnet før vi ikke selv leverede dataen, men fandt den udefra nemlig The World Factbook. En del af troværdigheden ved denne kilde ligger også i at, the factbook bliver udgivet at CIA som en ressource. Fra hver verdensdel valgte vi to lande, som kunne repræsentere dem. Lande er: Danmark, Frankrig, Kina, Japan, Australien, New Zealand, Etiopien, Ghana, Colombia, Venezuela, USA, Canada, og Antarktis. Factbook opgav som sagt de fleste oplysninger, derfor tog vi byer og de fleste sprog derfra.

Efter at have fundet data og lavede databasen manuelt, undersøgt vi hvordan vi kan opbevare den. For at arbejde med MongoDB, har vi kigget på de kode eksempler som vi har gennemgået i timerne. Det materiale vi har derfra har været en god forudsætning for at, arbejde med MongoDB. Hvis der opstod nogle problemer med programmering, som vi ikke kunne finde svar på ud fra timerne, så valgte vi at gøre brug af MongoDB[5]

[5] *https://docs.mongodb.com/manual/*

Arbejdsproces/Evaluering af proces

Vi har arbejdet systematisk med opgaven, hvor vi har lavet en liste med mål og arbejdet med en ting ad gangen. Vi har arbejdet på at få en ting til at virke, før vi er gået videre til næste punkt på listen. På den måde har vi sørget for at, nogle funktioner virker helt, fremfor at der er mange som kun virker halvt. Men selvom vi har fokuseret meget på en ting ad gangen, har vi, når vi sad fast med en del af opgaven, i stedet kigget på noget helt andet, indtil vi kunne få hjælp til at komme videre. Som det første var vores mål at få forbindelse til databasen og få trukket noget data ud som vises på siderne. Da det lykkedes os kunne vi derefter arbejde videre på at kunne redigere og tilføje data til databasen.

Som første led i vores arbejdsproces har vi brainstormet ideer omkring opbygning af website, hvilke funktioner vi skulle have med, også i forhold til de formelle krav til opgaven, og hvordan vi på bedste vis kunne præsentere data på websitet.

Næste led var at finde den data som vi skulle bruge og oprette i vores database. Dette tog meget af vores tid, da vi først satte data ind en af gangen. Vi fandt senere ud af at, vi kunne smide mere data ind af gangen, som endte med at spare os tid.

Vi har haft et fælles GIT repository hvor vi har delt kode med hinanden. Vi havde i starten af projektet en del problemer med GIT, især når vi redigerede i den samme fil eller lavede push på samme tid. Det krævede en del koordinering af hvem der pushede først og vi endte med aftale hvem af os der redigerede i koden hvornår, så vi ikke lavede rod i vores filer.

Vi har alle arbejdet med kodning på samme tid, hvor en af os har delt skærmen med de andre. På den måde kunne vi alle følge med i processen og komme med input og forslag til løsninger på problemer. Det eneste minus der måske har været ved denne metode er at, ikke alle har haft fingrene lige meget i koden.

Undervejs har vi haft nogle problemer og frustrationer omkring selve koden.
Vi havde som det første problemer med at få forbindelse til vores MongoDB database, så vores data kunne vises på sideren. Vi testede først med data på landene. Det var derefter nemt at få det til at virke på de andre sider, da det jo er den samme løsning vi har brugt.[6]
Vi havde svært ved at få trukket dataen ud. Grunden til det var at, vi ikke havde snakket ordentligt sammen omkring hvilket sprog vi ville benytte os af. Derfor endte det med at være en blanding af engelsk og dansk, hvilket jo ikke var optimalt da vi senere skulle trække dataen ud. Vi opdagede også andre småting som stavefejl, nogle af vores nøgler og værdier var skrevet forskelligt, glemt komma eller anførselstegn. Hvis vi havde været lidt bedre til at kommunikere lige omkring den del, så kunne vi have undgået disse problemer.
På vores forside har vi valgt at, præsentere de 7 kontinenter. Hvis der trykkes på et af de 7 kontinenter, vil der blive vist data over landene som findes i kontinentet og i databasen. Hvordan vi lige skulle løse dette havde vi problemer med.[7]
Hele delen med at kunne redigere og tilføje data til databasen, har vi også haft nogle problemer med og derfor det sidste som vi har kunne nå at tilføje. Vi har valgt at tilføje en ekstra side på websitet, som fungerer som en slags admin side, hvor der kan redigeres og tilføjes data til de tre collections. Grunden til at det kun virker på by og sprog er fordi, vi ikke har lagt kontinenterne ind i en collection. Det betyder at der ikke kan laves det samme check som der laves på by og sprog.[8]

[6] *Løsning til dette kan findes under afsnittet konstruktion*
[7] *Løsning til dette kan findes under afsnittet analyse og konstruktion*
[8] *Løsning til dette kan findes under afsnittet konstruktion*

Analyse

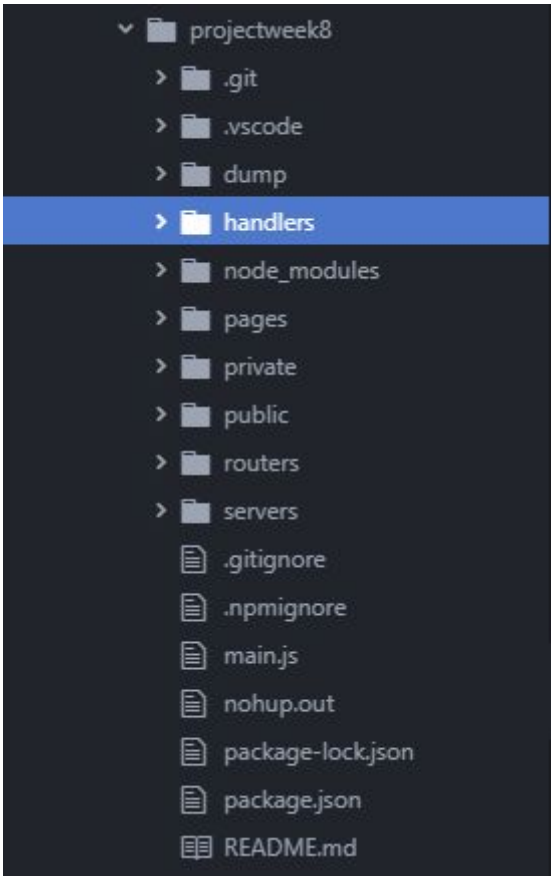
Da vi fik opgaven, fik vi stillet en masse krav på forhånd, som den skulle indkludere. I vores gruppe snakkede vi om hvad der ellers skulle være af krav til vores site. Her kom vi frem til nogle krav til vores forside. Vi har valgt ikke at, stille for mange krav. Dette skyldes at vi valgte at lægge vægt på opgavens formelle krav. De andre krav blev kun en realitet, hvis vi havde tid til overs. Sådan det ikke påvirkede vores arbejde og gav os ekstra press. Vi har valgt at lave en dynamisk forside, hvor folk klikke rundt på et verdenskort. Derefter kan de få informationer omkring det enkelte kontinent.

Hvad har vi så gjort for at nå frem til dette? Det første vi gik i gang med for, at nå dette var at finde et verdenskort. Vi valgte at finde et SVG billede af kontinenterne, sådan der ville være mulighed for at arbejde med hver enkelt land eller kontinent. Som start var planen at du skulle klikke på et land og derved få information omkring det land. Dog valgt vi at lave dette om til de enkelte kontinenter, som krævede vi lige rettede i billedet. Så vi fik grupperet landene til det givne kontinent. I vores index.html har vi ved hver kontinent, linket til de forskellige verdensdele. Som der er forbundet med vores continents.js fil, som var givet på forhånd. Derudover har vi også lavet en JavaScript fil, som der hedder af.js hvilket der skal indeholde lande i det valgte kontinent. I den fil vælger vi hvilket information der skal vises og hvordan den vises. Vi har valgt en klassisk tabel, hvor navn, kontinent, areal, befolkning og styreform for landene kommer frem. I vores handler.js fil, require vi filen oppe i toppen. Længere nede i filen har vi lavet en funktion findCountryByContinent, som der bliver registreret til en router, sådan at den fanges.

Konstruktion

Moduler

Vi har valgt i vores projekt at gøre brug af moduler. Grunden til vi har valgt at gøre det på denne måde, er fordi vi mener at det gør hele mappestrukturen mere overskuelig, samt det er nemmere at finde rundt i hvordan tingene hænger sammen.



Hvordan har vi gjort det

Selve konstruktionen af vores “hjemmeside” startede vi med at lave et skelet ud fra det vi havde lavet i undervisningen med Niels. Vi startede samtidigt også med at finde en masse data, som vi kunne indsætte i nogle collections i vores database world. Dette gjorde vi fordi at vores udgangspunkt i starten af processen, var at vi skulle have vores prototype til at skrive data ud fra databasen på vores dynamiske sider. Vi begyndte med at lave en “server” ved hjælp af node, sådan så vi kunne arbejde med routing og handling af vores html, på en dynamisk måde. Dette gjorde vi ved hjælp, af en en server fil & main.js. Disse ser ud & starter som set her under.

```
"use strict";

var server = require("./servers/server");              //Gør server modulet tilgængeligt for resten
var router = require("./routers/router");           // Router modulet required, sådan så den altid tjekker om routing er gjort rigtigt

server.start(router);                              // start server
                                                   // callback to route
```

```
"use strict";

const handlers = require("../handlers/handler");    // handlers module
const httpStatus = require("http-status-codes");
const contentTypes = {
    "text": { "Content-Type": "text/plain; charset=utf-8" },
    "start": { "Content-Type": "text/html; charset=utf-8" },
    "admin": { "Content-type": "text/html; charset=utf-8"},
    "js": { "Content-Type": "application/js" },
    "css": { "Content-Type": "text/css" },
    "png": { "Content-Type": "image/png" },
    "jpg": { "Content-Type": "image/jpg" },
    "gif": { "Content-Type": "image/gif" },
    "ico": { "Content-Type": "image/x-icon" },
    "svg": { "Content-Type": "image/svg+xml" }
};

const routes = {                                    // register handles to route
    "GET": {
        "/start": handlers.getAndRespond,
        "/country": handlers.findCountry,
        "/city": handlers.findCities,
        "/language": handlers.findLanguage,
        "/admin": handlers.getAndRespond,
        "/admincity": handlers.getAndRespond,
        "/adminlanguage": handlers.getAndRespond,
        "/af": handlers.findCountryByContinent,
        "/ant": handlers.findCountryByContinent,
        "/as": handlers.findCountryByContinent,
        "/eu": handlers.findCountryByContinent,
        "/na": handlers.findCountryByContinent,
        "/oc": handlers.findCountryByContinent,
        "/sa": handlers.findCountryByContinent,
        "js": handlers.getAndRespond,
        "css": handlers.getAndRespond,
        "png": handlers.getAndRespond,
        "jpg": handlers.getAndRespond,
        "gif": handlers.getAndRespond,
        "ico": handlers.getAndRespond,
        "svg": handlers.getAndRespond
    },
```

Herefter startede vi med at lave en forside, som vi syntes skulle være udgangspunktet. Plus at vi lavede en header med en menu, sådan så vi kunne gå nemmere imellem vores sider. Det kan ses her under:

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8"/>
        <title>ROOT</title>
        <link rel="stylesheet" href="./style.css">
    </head>
    <body>
        <header>
            <h1>Map</h1>

            <nav>
                <ul>
                    <li><a href="/">Home</a></li>
                    <li><a href="/country">Country</a></li>
                    <li><a href="/city">City</a></li>
                    <li><a href="/language">Language</a></li>
                    <li><a href="/admin">Admin Area</a></li>
                </ul>
            </nav>

        </header>

            <div class="map">

                <svg class="svg" version="1.1"xmlns="http://www.w3.org/2000/svg" viewBox="0 0 950 620">=

            </div>
        <main>
        </main>
        <footer>
        </footer>
    </body>
</html>
```

Vi valgte ud fra denne index, at vi skulle arbejde med dynamisk routing & handling af vores undersider. Sådan så vi rent faktisk bare “refresher” indholdet på vores index/side.html.

En del af denne routing som sker kan ses her:

```
        if (req.url.charAt(req.url.length - 1) === "/") {
            asset = "/start";
            routedUrl = "pages/index.html";
            type = contentTypes.html;
        } else if (req.url === "/start") {
            asset = req.url;
            routedUrl = "pages/index.html";
            type = contentTypes.html;
        } else if (req.url === "/country") {
            asset = req.url;
            routes[req.method][asset](req, res);
            return;
        } else if (req.url === "/af" || req.url === "/ant" || req.url === "/as" || req.url === "/eu" || req.url === "/na" || req.url === "/oc" || req.url === "/sa" ) {
            asset = req.url;
            routes[req.method][asset](req, res, asset);
            return;
        } else if (req.url === "/city") {
            asset = req.url;
            routes[req.method][asset](req, res);
            return;
        } else if (req.url === "/language") {
            asset = req.url;
            routes[req.method][asset](req, res);
            return;
```

Denne router bruger man til at fortælle hvilken funktion der skal ske når man kommer ind på en bestemt url, eksempelvis /country bliver sendt ind på følgende side, som så tager funktionen findCountry, som ser ud som set her under.

```
201     findCountry(req, res) {
202         const mongo = require('mongodb');
203         const dbname = "world";
204         const constr = `mongodb://localhost:27017`;
205 
206         mongo.connect(constr, { useNewUrlParser: true, useUnifiedTopology: true}, function (error, con) {
207             if (error) {
208                 throw error;
209             }
210             const db = con.db(dbname);                  // make dbname the current db
211             /* Retrieve,
212              * reads cities from the database
213              */
214             db.collection("country").find().toArray(function (err, land) {
215                 if (err) {
216                     throw err;
217                 }
218                 res.writeHead(httpStatus.OK, {
219                    "Content-Type": "text/html; charset=utf-8" } ); /* yes, write relevant header */
220                 res.write(experimental1.country(land));           // home made templating for native node
221                 con.close();
222                 res.end();
223             });
224         });
225     },
```

Denne funktion connector til vores mongodb og finder alle data i collectionen country, og sender dem videre til experimental1.country som er siden myCountries i mappen private og som har en function country. myCountries ser sådan her ud:

```
/* myCountries.js Home made experimental templating */
"use strict";

const country = function(obj) {
    let htmltop = `<!doctype html>
<html>
    <head>
        <meta charset="utf-8"/>
        <title>Country</title>
        <link rel="stylesheet" href="./style.css"/>
    </head>
    <body>
        <header>
            <h1>Country</h1>
            <nav>
                <ul>
                    <li><a href="/">Home</a></li>
                    <li><a href="/country">Country</a></li>
                    <li><a href="/city">City</a></li>
                    <li><a href="/language">Language</a></li>
                    <li><a href="/admin">Admin Area</a></li>
                </ul>
            </nav>
        </header>
        <div>`;

    let htmlbot = `        </div>
    </body>
</html>`;

    let dynamic = "<table><tr><th>Navn</th><th>Kontinent</th><th>Areal</th><th>Befolkningstal</th><th>Styreform</th></tr>";
    for (let i= 0; i < obj.length; i++) {
    dynamic += `<tr><td>${obj[i].navn}</td>`;
    dynamic += `<td>${obj[i].kontinent}</td>`;
    dynamic += `<td>${obj[i].areal}</td>`;
    dynamic += `<td>${obj[i].befolkningstal}</td>`;
    dynamic += `<td>${obj[i].styreform}</td></tr>`;
};

dynamic += `</table>`;

    return htmltop + dynamic + htmlbot;
}

exports.country = country;
```

Alt dette vil så resultere i at vores side vil se ud som ses på billedet her under:

Country

Home Country City Language Admin Area

Navn	Kontinent	Areal	Befolkningstal	Styreform
Etiopien	Afrika	1.104.300 km2	108.400.000	federal parliamentary republic
Ghana	Afrika	238.533 km2	29,340,248	presidential republic
Antarktis	Antarktis	14.200.00 km2	-	-
Kina	Asien	9.596.960	1.400.000.000	communist party-led state
Japan	Asien	377.915	126.200.000	parliamentary constitutional monarchy
Danmark	Europa	43.094 km2	5.869.410	konstitutionelt monarki
Frankrig	Europa	643.801	49.300.000	semi-presidential republic
USA	North America	9.833.517 km2	332.639.102	Repræsentativt demokrati
Canada	North America	9.984.670 km2	37.694.085	Konstitutionelt monarki med et parlamentarisk system
Autralien	Oceania	7.741.220 km2	23.500.000	parliamentary democracy (Federal Parliament) under a constitutional monarchy, a Commonwealth realm
New Zealand	Oceania	268.838 km2	4.925.477	parliamentary democracy under a constitutional monarchy, a Commonwealth realm
Columbia	South America	1.138.910 km2	49.084.841	Præsident Republik
Venezuela	South America	912.050 km2	28.644.603	Forbundspræsident Republik
Norge	Europa	385.203	5.328.212	parlamentarisk
Sverige	Stockholm	450.295	10.065.389	konstitutionelt monarki
Finland	Europa	338.424	5.180.000	Republik
Tyskland	Europa	357.021	81.770.900	Forbundsrepublik

Konklusion

Vi har ved brug af JavaScript, Node.js og MongoDB, skabt et website med dynamiske sider, og løst opgaven omkring display og vedligeholdelse af data.

Før at vi overhovedet kunne lave websitet, var vi nødt til at starte med research af information. Sådan vi kunne få et overblik omkring hvilke lande, der skulle indkluderes, da vi havde en række krav til landets informationer. Informationerne der fremgår på vores side, blev fundet ved hjælp af The World Factbook. Derudover supplerede vi med Wikipedia, hvis noget data manglede eller tallene var gamle. Den data som vi fandt, skrev vi ned som en JSON struktur i separate filer. JSON giver os lov til at bruge key:value parring, hvilket er vigtigt i denne sammenhæng. Ved hjælp af det kunne vi indsætte disse data i en database. Vi skabte databasen *world*, ved hjælp af MongoDB, som indeholder de collections vi havde brug for.

I forbindelse med at få fremvist data på siden, var vi nødt til at køre det hele igennem en node “server”(localhost), sådan så vi kunne få det gjort på en dynamisk måde. Dette gjorde at vi kunne benytte en router, til at fortælle hvilke funktioner der skulle bruges på de forskellige /xxx url’er. Da vi så skulle have data’en fra vores database trukket ud, benyttede vi os af funktionen findCountry på countries siden. som trak dataen ud, dette gjorde den ved hjælp af:

```
db.collection("country").find().toArray(function (err, land) {
    if (err) {
        throw err;
    }
```

som bare tager alt i collection country og skriver det ud.

I forbindelse med at vi skulle kunne oprette og redigere data i vores database, startede vi med at benytte os af insertOne, men måtte så konkludere at det var bedre for os at benytte updateOne. Da den tillader at man opretter en ny værdi i databasen, hvis den ikke allerede findes i kollektionen. Findes den allerede i databasen, vil den til gengæld blot opdatere de ændrede værdier i den nuværende.

Referencer

Wikipedia:

https://www.wikipedia.org/

CIA Fact Book:

https://www.cia.gov/library/publications/resources/the-world-factbook/docs/one_page_summaries.html

Niels' noter:

http://dkexit.eu/,

http://dkexit.eu/webdev/site/ch18s06.xhtml,

http://dkexit.eu/webdev/site/ch19.xhtml

MongoDB:

https://www.mongodb.com/

https://docs.mongodb.com/manual/