

---

# *Vejrportalen*

---

VEJRET ER LIGE BLEVET BEDRE

<b>Uddannelse sted</b>	Smartlearning
<b>Fag</b>	Webprogrammering – Frontend
<b>Underviser</b>	Constantin Alexandru Gheorghiasa
<b>Forfatter</b>	Lars Larsen
<b>Projektperiode</b>	3. september 2018 – 9. december 2018
<b>Dato for aflevering</b>	5. december 2018

## Titelblad

<b>Uddannelse sted</b>	Smartlearning
<b>Uddannelse</b>	Diplomuddannelse
<b>Fag</b>	Webprogrammering - Frontend
<b>Underviser</b>	Constantin Alexandru Gheorghiasa
<b>Forfatter</b>	Lars Larsen
<b>Projektperiode</b>	3. september 2018 – 9. december 2018
<b>Dato for aflevering</b>	5. december 2018 kl. 12.00
<b>Antal sider</b>	30
<b>Antal karakterer</b>	26.562
<b>Vejrportalen URL</b>	<a href="https://40.127.170.50/vejrportalen/">https://40.127.170.50/vejrportalen/</a>
<b>Github URL</b>	<a href="https://github.com/larsk7cdk/vejrportalen">https://github.com/larsk7cdk/vejrportalen</a>

## Indholdsfortegnelse

1	Indledning .....	5
2	Casebeskrivelse.....	5
3	Problemstillinger/Afgrænsning.....	5
4	Problemformulering.....	6
5	Teori.....	6
5.1	Design principper .....	6
5.2	Frontend.....	8
5.3	Backend .....	10
6	Analyse .....	11
7	Løsningsforslag.....	12
7.1	Kommunikationsstrategi .....	12
7.2	Vejrportalen – Design .....	14
7.3	Vejrportalen – Frontend .....	18
7.4	Vejrportalen – Backend.....	25
7.5	Brugervenlighed .....	27
8	Konklusion.....	28
9	Liste over referencer .....	29

## Figurfortegnelse

Figur 1 - Mindmap for kommunikationsstrategi.....	12
Figur 2 - Vejrportalen indhold .....	14
Figur 3 - Vejrudsigten design.....	15
Figur 4 – Varslinger design.....	17
Figur 5 - Abonnement design.....	18
Figur 6 – Filstruktur for kode.....	19
Figur 7 – Index HTML og JavaScript kode .....	20
Figur 8 - Vejrudsigten HTML og JavaScript kode .....	21

Figur 9 - Geolocation JavaScript kode.....	22
Figur 10 - Varsling HTML kode.....	23
Figur 11 - Varsling JavaScript kode .....	24
Figur 12 - PHP Get kode.....	27

## 1 Indledning

I min projektopgave "Vejrportalen" vil jeg lave et website, hvor det er muligt at kunne finde vejrudsigten for en given by i Danmark. Da vi lever i en tid, der giver anledning til store vejrforandringer, og vejret derfor hurtigt kan skifte fra tid til anden, skal det være hurtigt at kunne holde sig orienteret om vejr situationen. Der er mange der har behov for at kunne holde sig opdateret. Dette kan strække sig fra privatpersoner til firmaer og kommuner. Derfor er hovedformålet med vejrportalen, at man kan hurtigt kan få vist en 5-døgnsudsigt. Hvis brugeren har lokation aktiveret på sin enhed, vil byen for den aktuelle lokation blive benyttet, ellers er det muligt at indtaste en by.

For at brugere af vejrportalen kan hjælpe hinanden, er det muligt at abonnere på vejrvarslinger, som andre brugere måtte registrere. Aktive vejrvarslinger, vil også være mulige at se inde på vejrportalen.

## 2 Casebeskrivelse

Vi har som webudvikler fået stillet opgaven, at lave et website med fokus på at oplyse om en virksomheds produkter og tjenester, så de bliver relevante for en eller flere målgrupper.

Virksomheden stiller vejrdata til rådighed for brugere på internettet. Endvidere udbyder de mulighed for at kunne tilmelde sig forskellige services.

## 3 Problemstillinger/Afgrænsning

For at få gang i vejrportalen, er det vigtigt at få sat nogle realistiske mål som rammer bredt i de målgrupper som beskrives senere i kommunikationsstrategien. Det er også vigtigt ikke at sætte for mange mål, så det bliver uoverskueligt at nå dem. Derfor er følgende mål stillet

- Det er muligt at se vejret på en given lokalitet
- Ofte opdatering af vejrdata
- Kunne indberette en vejrvarsling
- Kunne se aktive vejrvarslinger
- Mulighed for at abonnere på vejrvarslinger

Ovenstående mål er opstillet i prioriteret rækkefølge

## 4 Problemformulering

Hvordan kan vejrportalen laves som et website.

Mangler ...

## 5 Teori

Til udvikling af vejrportalen har jeg benyttet HTML5, CSS og JavaScript, samt PHP til API delen. For at optimere JavaScript til understøttelse i alle moderne browsere er jQuery benyttet. Bootstrap er brugt til understøttelse af komponenter og layout og fontawesome til ikoner, igen for browser understøttelse. De forskellige 3. parts libs hentes via CDN'er.

### 5.1 Design principper

Gestalt lovene er udarbejdet af en række tyske psykologer omkring 1920. De ville finde ud af, hvordan vi mennesker sanser, og gjorde dette ud fra en række forsøg, som ikke kun beskæftiger sig med synet, men med alle sanser. Denne form for psykologi kaldes perceptionspsykologi. Jeg vil her kort beskrive elementer af gestalt lovene.

#### LOVEN OM NÆRHED

*"Symboler, der er anbragt nær hinanden, opfattes som hørende sammen."*

For loven om nærhed, gælder det at elementer der hører sammen, skal placeres i nærheden af hinanden. På et website gælder dette f.eks.

- Billedtekst og billede
- Overskrift og tekst
- Menupunkterne i en menu

Så når elementerne anbringes tæt på hinanden for at vise de hører sammen, skal elementer som ikke hører sammen, placeres med mere luft mellem hinanden.

#### LOVEN OM LIGHED

*"Symboler, der ligner hinanden, opfattes som hørende sammen."*

Noget af det som loven om lighed dækker, er emner som form, farve, størrelse, placering osv.

Et godt eksempel på et website for denne lov, er en navigations menu. Menuen vil gå igen på alle vores sider og være placeret det samme sted. Endvidere vil de enkelte menupunkter stå i nærheden af hinanden og der vil derfor være et samspil med loven om nærhed. I en menu vil det aktive menupunkt som regel også have en anden farve for at fremhæve det.

Et andet eksempel kan være links. Det er ikke godt at benytte understregninger i en tekst på et website. Dette vil en bruger opfatte som et link.

### LOVEN OM LUKKETHED

*"Symboler, der står i samme ramme, opfattes som hørende sammen."*

På websites med mange informationer, kan loven om lukkethed benyttes. Formålet er at tekster og billeder placeres i rammer eller bokse, som omkranser information der hører sammen, og derfor er med til at skabe et overblik. Ved at indramme elementer, er det muligt at få mere information ind på siden, da elementerne kan stå tættere på hinanden.

Selvom indhold er vidt forskelligt, men alligevel skal give mening når det står sammen, giver det god mening at samle det i en ramme eller en boks.

### LOVEN OM FORBUNDETHED

*"Symboler, der er forbundet, opfattes som hørende sammen."*

Elementer på et website kan være forbundet på forskellige måder. Dette kan enten være i form af der er en linje imellem dem, eller baggrundsfarven hvor de er placeret indrammer dem.

Der hvor det giver mening at bruge en linje, vil være hvor 2 forbundne elementer står langt fra hinanden, eller som hjælpelinjer i en tabel.

Et andet eksempel hvor loven om forbundethed kan bruges, er på elementer som indeholder tabs.

Her vil det aktive tab som regel få samme farve som selve indholdet der bliver vist. På denne måde bliver de 2 elementer forbundet.

### LOVEN OM FIGUR OG BAGGRUND

*"Den mindste, afgrænsede figur på arealet vil først blive opfattet som figuren."*

Det er vigtigt at en baggrund på et website ikke tager opmærksomheden fra brugeren. Et andet perspektiv kan være, hvis baggrundsfarven er så kraftigt, at teksten ikke kan læses. Derfor foreskriver denne lov, at der skal være en god kontrast mellem tekst og baggrund.

Hvis der er placeret en figur på siden, skal denne figur være tydelig og ikke gå ud i et med baggrunden.

### 5.2 Frontend

I følgende afsnit vil jeg kort beskrive de sprog og libs, jeg vil benytte mig af i udviklingen af vejrportalen på frontend siden.

#### HTML5

HTML5 er den seneste standard og er udvidet med HTML tags, der semantisk beskriver dets betydning på en side i forhold til f.eks. skærm oplæsere. Det er også muligt at sætte en role attribut på et tag. En role er W3C's guideline for web standarder som beskriver et tag. En role kan også bruges til at overstyre et tag. For eksempel kan et anchor tag opføre sig som en button ved at sætte en role på

```
<a href="#" role="button" aria-label="Delete item 1">Delete</a>
```

For at informere en browser om at den benytter HTML5, angives følgende øverst i en .html side

```
<!DOCTYPE html>
```

En HTML side består af en head sektion, hvor man angiver meta data i form af f.eks sprog og sidens titel. Det er også i denne sektion der angives links til at loade stylesheets.

Den næste sektion er body. Det er her selve indholdet som skal vises på websitet placeres.

Nederst i body'en vil load af JavaScript placeres. Dette for at loade siden hurtigere og dermed vise indhold for brugeren, før alt JavaScript er hentet.

#### CSS

CSS står for Cascading Style Sheet og er et sprog, som benyttes til at beskrive hvordan HTML tags på et website skal vises. Den seneste standard af CSS er version 3.0. Denne standard er blevet udvidet med transitioner og animationer, samt media queries. Media queries er benyttet i forbindelse med websider der skal være responsive og fungere både på mobil og desktop.

Det er muligt at lave inline styles på et HTML tag, eller lave en style klasse i HTML filen. Hvis en style klasse skal benyttes på flere HTML sider, kan stylingen laves i en ekstern fil, med ekstensionen .css. Ved at referere til denne .css fil, kan den samme klasse benyttes på flere sider.



## JQUERY

jQuery er et lille, hurtigt og funktions rigt library til at understøtte JavaScript. Opgaver såsom at gennemløbe og finde elementer i på en HTML side, hændelses styring, animation og AJAX kald til API'er simplificeres væsentligt. Endvidere sørger jQuery for at understøtte funktionaliteten i alle moderne browsere.

For at jQuery kan opdatere et HTML elementet benyttes css selectors. Skal man derfor opdatere en tekst i et <p> tag med et ID som hedder "city", kan dette gøres på følgende måde

```
$("#city").text('Tekst som skal vises i <p> elementet');
```

## BOOTSTRAP

Bootstrap er komponent library, som kan benyttes til at opbygge en HTML side. Dette library indeholder knapper, paneler, tekstbokse og mange andre standard komponenter. Fordelen ved at benytte et komponent library, er at det virker ens på alle browsere. En af bootstrap's populære komponenter er dets navigation. Dette kan godt være en kompleks funktion, men den virker både i desktop og mobil visning, hvor den folder sig sammen til en burger menu.

Måden hvorpå bootstrap benyttes, er ved hjælp af HTML tags klasse. Derfor vil bootstrap komponenter benyttes som en decorator til standard HTML tags. En knap som skal have bootstrap's visuelle udseende og effekt skal derfor erklæres som følgende

```
<button class="btn btn-outline-success">Hent</button>
```

Bootstrap indeholder udover et komponent library, også et grid system til layout. Dette er opdelt i 12 kolonner som modsvarer skærmens bredde. På denne måde kan man lave sit layout, efter samme principper som man har i en avis.

Den seneste version af bootstrap understøtter muligheden for at benytte flex. Dette er en CSS3 feature, og er en ny måde at styre et layout på. Flex er særdeles velegnet til at lave responsive sider, da det understøtter en række funktioner såsom at stakke, wrappe, resize plus andre metoder for optimering af et responsivt design. For at benytte flex skal display erklæres på en css klasse

```
.weather-forecast {  
  display: flex;  
}
```

## FONTAWESOME

Fontawesome er et ikon library som indeholder en stor mængde af ikoner i forskellige kategorier. Ikonerne er i svg format og kan derfor skaleres i forskellige størrelser. For at benytte et ikon indsættes dette i et `<i>` tag på siden. Her er et eksempel på en spinner

```
<i class="fas fa-sync fa-spin fa-3x"></i>
```

## 5.3 Backend

Følgende er beskrevet de sprog og services der er benyttet som backend for vejrportalen.

### PHP

PHP står for HyperText Preprocessor og er et objekt orienteret server side programmeringssprog. Det er udbredt på mange websites rundt om i verdenen. En af årsagerne til det er så populært, er kildekoden er open source og dermed har det fået et stort fællesskab, som er med til at udvide sproget med nye features.

For at kunne benytte PHP, er det nødvendigt at have en server som kan fortolke PHP kode.

En PHP side har fil ekstension .php. For at markere en kode blok started med `<?php` og afsluttes med `?>`. Det sidste er dog ikke obligatorisk og benyttes kun hvis det skrives inline, f.eks. i forbindelse med HTML. Det er muligt at referere til andre PHP sider, hvis man skriver kode skal benyttes flere steder. Måden hvorpå der laves en reference er

```
<?php  
include_once "../models/response_result.php";  
include_once '../shared/shared_constants.php';  
include_once '../shared/database.php';
```

### MySQL

MySQL er en populær relationel database management system, som udvikles og vedligeholdes af Oracle. MySQL er open source og der findes drivere til stort set alle tænkelige sprog. Derfor er denne en af de mest udbredte databaser til brug på internettet.

## OPENWEATHERMAP

Der er benyttet et vejr API som hedder OpenWeatherMap, til at hente vejrdata for en given lokation. Ud fra et bynavn, er det muligt at få et svar med oplysninger omkring vejret de næste 5 dage. Et kald til dette API ser ud som følgende

<https://api.openweathermap.org/data/2.5/forecast?q=hvidovre&appid={{api-key}}&units=metric>

## 6 Analyse

Mangler...

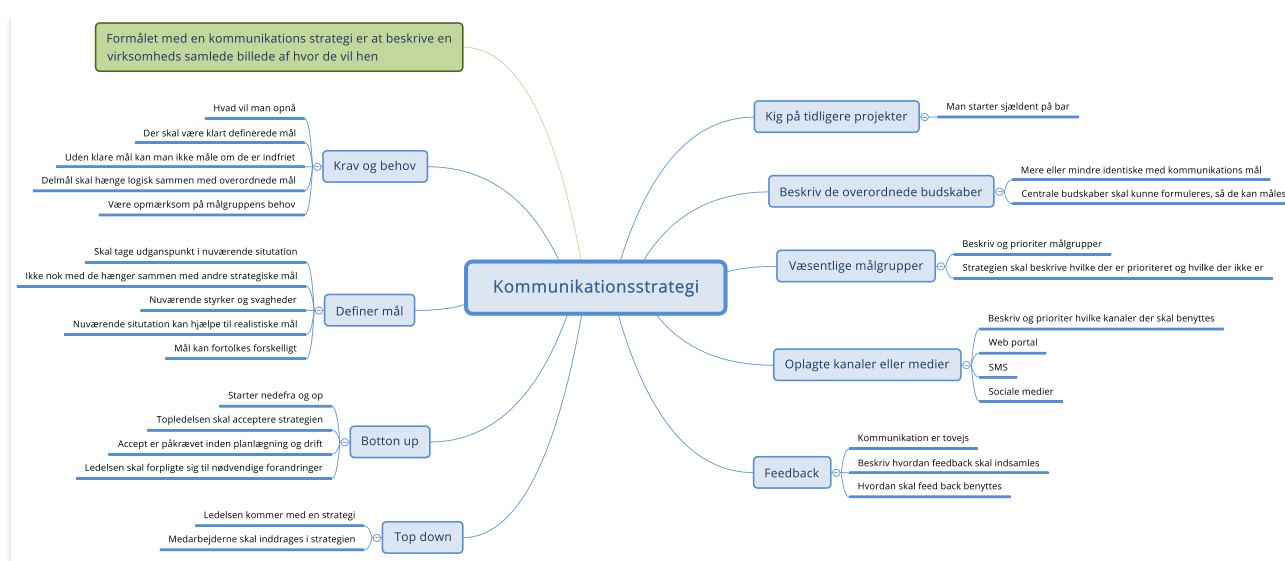
## 7 Løsningsforslag

I dette kapitel vil jeg beskrive mit løsningsforslag til vejrportalen.

### 7.1 Kommunikationsstrategi

Jeg har valgt at lave en kommunikationsstrategi for vejrportalen.

Når denne er udarbejdet, kan den omsættes til et eller flere skemaer, som beskriver overordnede mål og aktiviteter. Dette for at danne et bedre overblik, så de mål som bliver defineret kan nås.



Figur 1 - Mindmap for kommunikationsstrategi

### OVERORDNEDE BUDSKABER

Vejrportalens slogan er "Vejret er lige blevet bedre". Det vil være det overordnede budskab der skal kommunikeres ud til interessenterne, og dækker over opdaterede lokale vejrdato og vejrvarslinger

### MÅLGRUPPER

Målgrupperne som der satses på vil være kommuner og virksomheder som døgnet rundt har behov for at kunne holde sig opdateret på vejret. Da målgruppen er meget bred, kan der være stor forskel i kravene til hvordan hver enkelt vil kunne holde sig opdateret. Dette skal der tages højde for.

### KANALER FOR KOMMUNIKATION

Det er vigtigt at vejrportalen kan nås på flere forskellige kanaler fra de enkelte målgrupper. Som beskrevet ovenfor, er målgruppen alt lige fra kommuner til store virksomheder ned til

enkelpersoner. Derfor er det vigtigt at vejrportalen kan nås på mange forskellige kanaler. Derfor skal det som et minimum kunne tilgås fra

- Website
- Tablets og iPads
- Mobile enheder
- Notifikationer i form af E-mail

### FEEDBACK

For løbende at kunne foretage forbedringer og tilføje nye funktioner er feedback vigtig. Der skal derfor løbende opfordres til dialog mellem interessenter og ansvarlige hos vejrportalen. Ligeledes skal der være en funktion.

#### Overordnede mål i skema:

Formål	Budskaber	Målgrupper	Medier	Ansvarlig	Ressourcer	Succeskriterier
At skabe en vejrportal med vejrudsigt er og vejrvarslinger	Opdaterede lokale vejrudsigter Vejrvarslinger af høj kvalitet	Kommuner Virksomheder som er afhængige af vejret	Web Tablets og iPads Mobile enheder	Vejrportalen	Udvikling	Lokale vejrudsigter der opdateres hver time Lokale vejrvarslinger

#### Aktiviteter i skema:

Målgruppe	Delmål	Medier	Ansvarlig	Aktivitet	Succeskriterier	Feedback
Alle målgrupper	At kunne se vejrdato	Web portal	Vejr API	Udvikling af løsning til visning af vejrdato	Vejrdato er tilgængelige 24 timer i døgnet	Der laves en meningsmåling hvert halve år
Kommuner og større	At kunne se og indgive	Web portal SMS	Vejrportalen	Udvikling af løsning til	Muligt at kunne indgive vejrvarslinger	Mulighed for at kunne give

virksomhed r	vejrvarslin ger	Email		vejrvarslinge r		feedback på portalen
-----------------	--------------------	-------	--	--------------------	--	-------------------------

- **Formål** hvad er de overordnede strategiske mål for kommunikationen – skal hænge tæt sammen med målene for det som kommunikationen handler om
- **Budskaber** de centrale budskaber, som vi gerne vil have formidlet for at nå målene
- **Målgruppe** alle som har en interesse i vejrportalen – her prioriterer vi hvem vi først og fremmest skal kommunikere med
- **Delmål** her er de mål som vi gerne vil opnå med den bestemte gruppe af målgrupper
- **Medier** definerer, hvordan kommunikationen videregives
- **Ansvarlig** angiver hvem der konkret er ansvarlig for aktiviteten
- **Aktivitet** beskriver hvilken specifik handling, der skal gennemføres
- **Succeskriterier** gør det klart hvad der præcis skal til for, at I når jeres mål
- **Feedback** hvordan skal I modtage og samle op på feedback og respons fra interessenterne

## 7.2 Vejrportalen – Design

Nedenstående figur viser de funktioner jeg gerne vil have vejrportalen til at understøtte. Dette for at give et overblik over hvad der er af krav, for at give brugeren en forudsætning til at benytte vejrportalen.



Figur 2 - Vejrportalen indhold

**Top** skal indeholde et logo, som giver brugeren et vartegn for vejrportalen. Endvidere skal den indeholde navigation, til vejrportalens andre sider.

**Vejrudsigten** vil være vejrportalens landingpage. Det vil være denne side, brugeren vil kunne søge på en given by og få en 5-døgnsudsigst. Ligeledes kan man læse om de andre sider der findes, samt se en instruktions video.

**Varslinger** giver et overblik over de aktive varslinger der måtte være indberettet af brugerne.

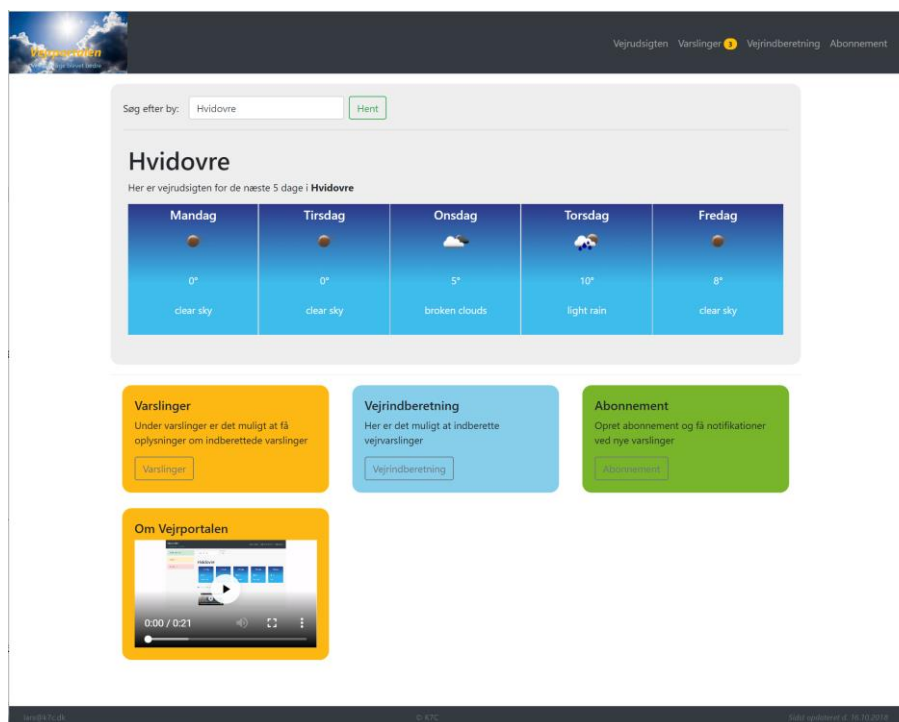
**Vejrindberetning** giver mulighed for at kunne indberette en varsling, som så vises på varsling siden.

**Abonnement** siden, hvor brugere kan registrere sig for at modtage varslinger når de oprettes. På denne måde bliver de gjort opmærksom på eventuelle vejrskifter.

### Vejrudsigten

Nedenstående figur viser vejrportalens forside design. Dette har været igennem flere iterationer, hvor det løbende er blevet rettet til med de erfaringer der er gjort undervejs.

Opbygningen af siden har taget udgangspunkt i at overholde gestalt lovene.



Figur 3 - Vejrudsigten design

### LOGO

For at give vejrportalen et vartegn, er der placeret et logo i venstre øverste hjørne. Dette skal benyttes ved omtale af vejrportalen, for at give brugere noget genkendeligt.

### NAVIGATION

Navigationen er placeret i toppen af websitet og højre stillet. Navigationslinkene vil være gennemgående på alle sider. Siden som er aktiv vil have en hvid farve for at indikere denne er valgt.

I tilfælde af siden gøres mindre, vil menuen ændre sig til en burger menu. Dette vil også være tilfældet i mobil visning og er en funktion som stilles til rådighed af bootstrap menu komponent.

I tilfælde af der er aktive vejrvarslinger, er der ved siden af linket placeret et badge med antallet af aktive varslinger.

### **SØGEFELT OG 5-DØGNSUDSIGT**

For at give brugeren mulighed for at fremsøge en vejrudsigt for en given by, har jeg valgt at placere denne funktion øverst på siden, så den er tilgængelig og synlig for brugeren, uanset device. Har browseren adgang til lokationen, hentes vejr data når man går ind på siden. Bynavn vil blive vist i søgefeltet, og kan rettes hvis der ønskes en anden by.

Vejrdata for de enkelte dage er placeret i et card, som indeholder dag, vejr ikon, temperatur og beskrivelse af vejret.

Søgefeltet og visningen af 5-døgnsudsigten er indrammet med en baggrundsfarve, for at vise de 2 funktioner hænger sammen.

### **BESKRIVELSE AF SIDER**

Nederst på siden, har jeg placeret information om de forskellige muligheder vejrportalen stiller til rådighed. Farverne på de enkelte bokse er taget fra logoet, så der er en tråd af farvevalg på websitet, samt give en kontrast til baggrunden. I de enkelte bokse er der placeret en knap, som giver mulighed for at skifte til omtalte side.

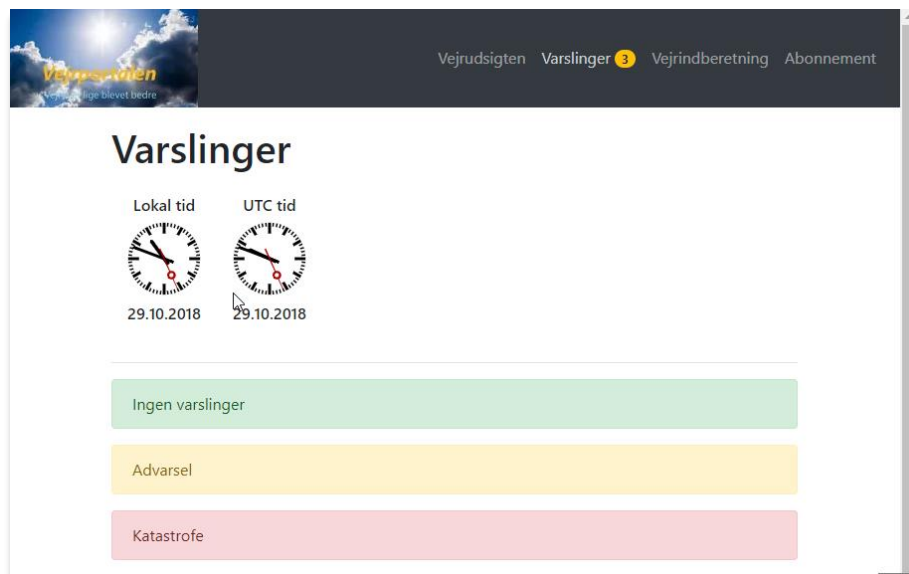
### **FOOTER**

I bunden af sitet er placeret en footer, som er gennemgående på alle sider på websitet. Dette indeholder kontakt e-mail, firmanavn og hvornår websitet sidst er blevet opdateret.



## Varslinger

På varsling siden, vil der være mulighed for at se aktive varslinger. Endvidere er der placeret 2 ure, så man i nærhed har et klokkeslæt og dato, som kan holdes op imod varslingerne.



Figur 4 – Varslinger design

De enkelte varslinger som registreres, kan have forskellig kategori som følgende

- Grøn: Ingen varslinger
- Gul: Varsling som indikerer at der kan være risiko for dårligt vejr
- Rød: Varsling som indikerer dårligt vejr

### Abonnement

På abonnement siden er det muligt for en bruger at registrere sig og dermed modtage varslinger på e-mail eller sms, når de registreres. Siden er responsiv og kan dermed benyttes både på desktop og mobil. E-mail og telefon nummer er sat som obligatoriske felter. De skal være udfyldt for at man kan registrere sig. Ved klik på ”Send” knappen, oprettes en registrering og brugeren vil efterfølgende modtage varslinger.

**Abonnement**

Her kan du registrere dig for at abonnere på varslinger

Email  
\* Skal være udfyldt!

Fornavn

Efternavn

Adresse

Postnummer

By

Telefon  
\* Skal være udfyldt!

Titel

Send

Figur 5 - Abonnement design

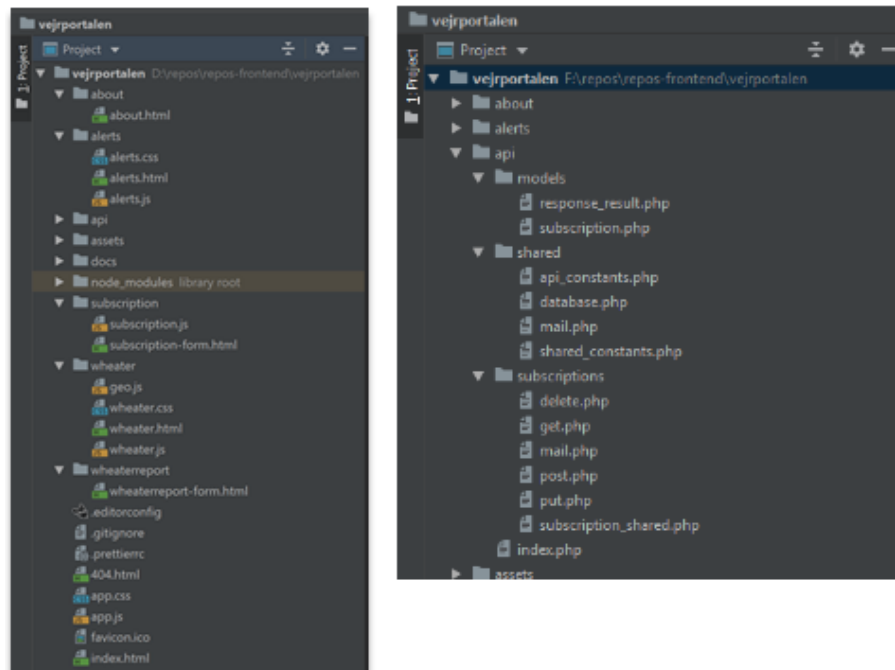
### 7.3 Vejrportalen – Frontend

Vejrportalen er bygget ved brug af de beskrevne sprog og libs i tidligere kapitel. I dette kapitel vil jeg beskrive frontend koden for vejrportalen.

### FILSTRUKTUR

Kildekode for både front-end og API er samlet i samme løsning. API koden er samlet under mappen `api` og har dermed sin egen filstruktur.

For front-end koden, er de enkelte sider placeret i hver sin mappe, for at holde en struktur der gør det nemt overskueligt når der skal foretages udvidelser eller rettelser.

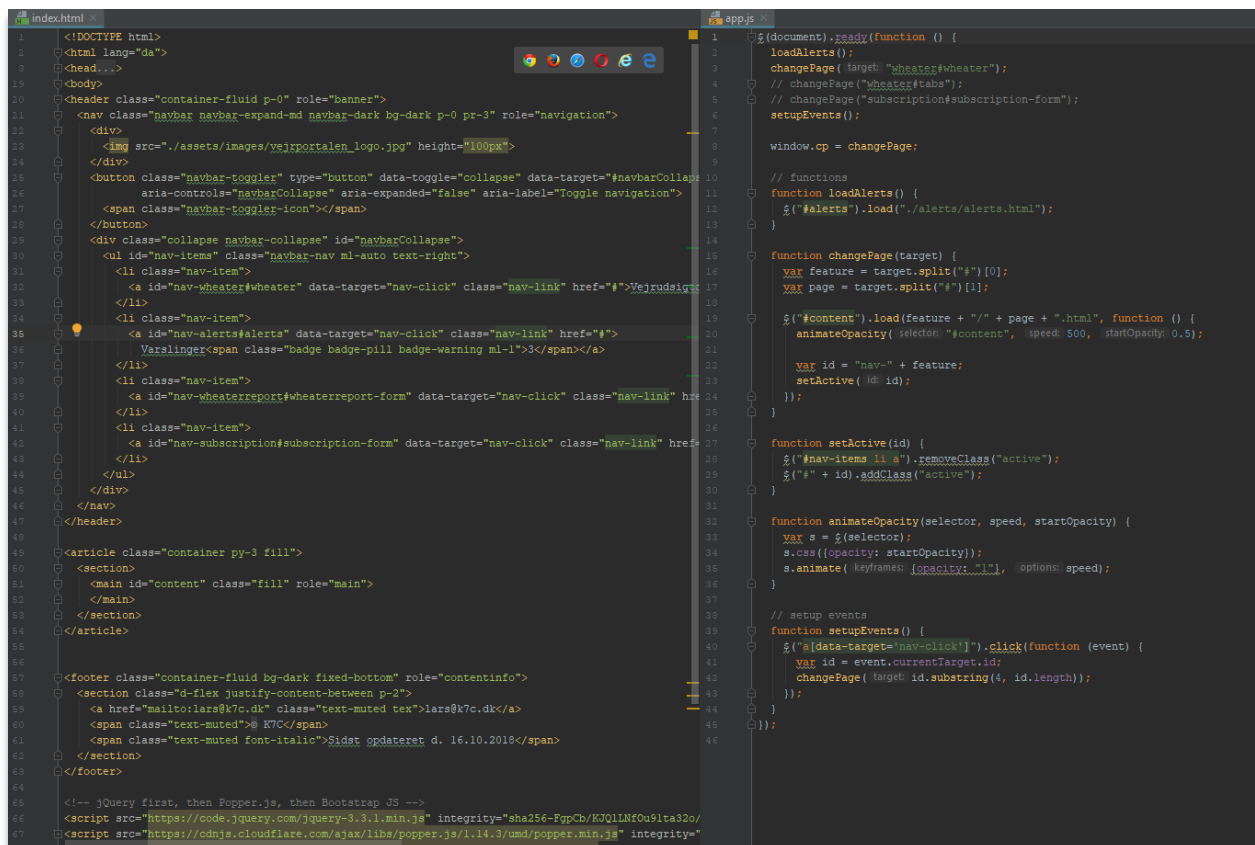


Figur 6 – Filstruktur for kode

## INDEX

Index siden er bygget som en master page. Her vil elementer såsom logo, navigation og firmanavn der er fælles for alle sider være placeret. Dette gør at lige meget hvilken side der er valgt, vil det visuelt være ens.

JavaScript kode for index siden, er placeret i filen app.js.



Figur 7 – Index HTML og JavaScript kode

På index.html siden er der lavet en unordered list til brug for navigation. Hver list item har et tag med hver deres unikke ID. ID'et benyttes til jQuery's selector, så ved brug af jQuery load funktion, asynkront kan renderes html på index siden.

Dette har flere fordele. Index.html siden hentes kun en gang og ved sideskift, har man ikke oplevelsen af at siden laver et blink. Når der sker et sideskift, har jeg lavet en animation på opacity. Dette giver fornemmelsen af en glidende overgang, når der navigeres til en ny side.

Jeg har generelt på alle siderne, benyttet mig af tags der semantisk beskriver de enkelte sektioner i opbygningen af HTML strukturen. Dette øger læsevenligheden af koden, så når man f.eks. på

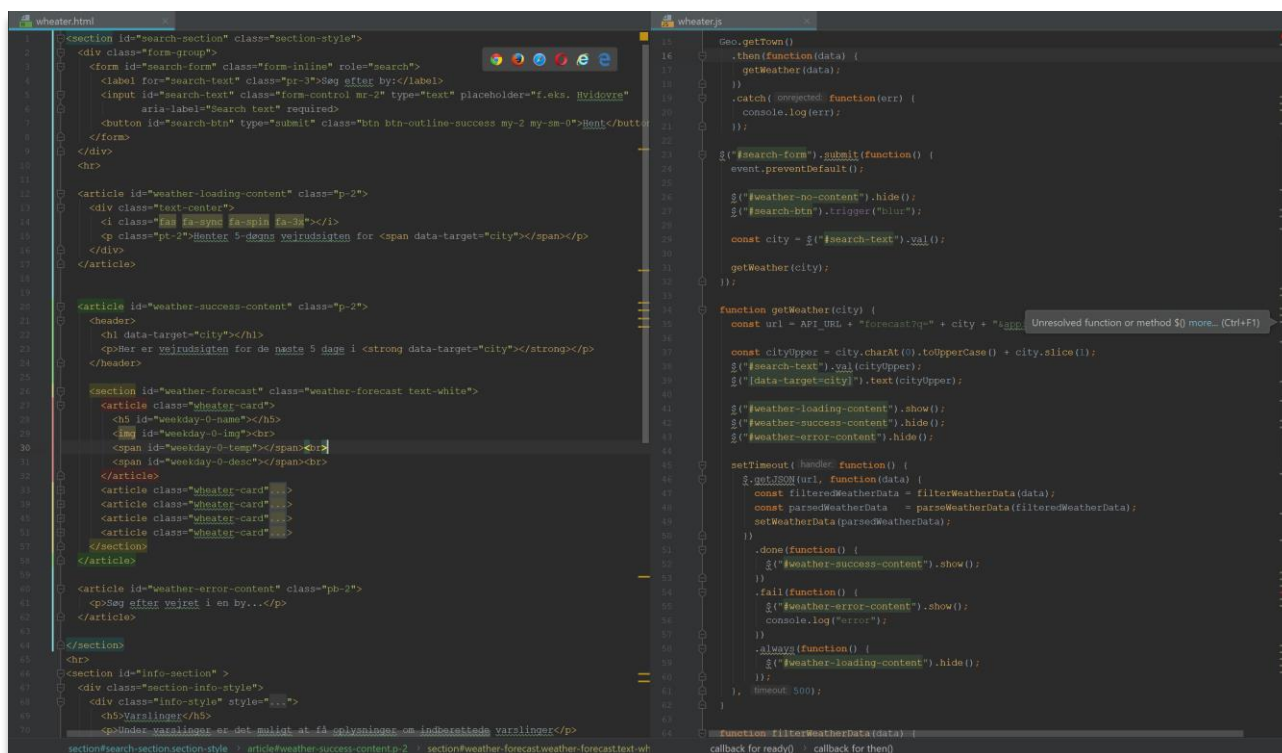
index.html skimmer ned over den, kan man hurtigt danne sig et overblik over header, article (indhold af siden) og footer sektionerne.

Nederst på siden hentes 3. parts JavaScript filerne fra CDN's. fordelene ved at benytte CDN's er at når filerne først er hentet en gang, bliver de cachet i browseren. Er filerne f.eks. hentet fra en anden side end vejrportalen, vil de stadig være cachet og kan dermed benyttes.

Egen udviklet JavaScript filer, hentes med en "defer" egenskab, hvilket muliggør asynkron hentning og dermed øger brugervenligheden i form af siden vises hurtigere.

### VEJRUDSIGTEN (WEATHER)

Vejrudsigten indeholder søgefelt og visning af vejrkort, samt informations bokse som beskriver de andre sider på vejrportalen. Heraf en video, der kort giver en introduktion til hvordan vejrportalen benyttes. JavaScript koden som benyttes, er placeret i henholdsvis weather.js og geo.js.



```
1 <section id="search-section" class="section-style">
2   <div class="form-group">
3     <form id="search-form" class="form-inline" role="search">
4       <label for="search-text" class="sr-only">Søg efter by:</label>
5       <input id="search-text" class="form-control sr-2" type="text" placeholder="f.eks. Hvidovre"
6         aria-label="Search text" required>
7       <button id="search-btn" type="submit" class="btn btn-outline-success my-2 my-sm-0">Hent</button>
8     </form>
9   </div>
10   <hr>
11
12   <article id="weather-loading-content" class="p-2">
13     <div class="text-center">
14       <i class="fas fa-sync fa-spin fa-3x"></i>
15       <p class="pt-2">Henter 5-dagene vejrudsigten for <span data-target="city"></span></p>
16     </div>
17   </article>
18
19   <article id="weather-success-content" class="p-2">
20     <header>
21       <h1 data-target="city"></h1>
22       <p>Her er vejrudsigten for de næste 5 dage i <strong data-target="city"></strong></p>
23     </header>
24     <section id="weather-forecast" class="weather-forecast text-white">
25       <article class="weather-card">
26         <div id="weekday-0-name"></div>
27         <div id="weekday-0-temp"></div>
28         <div id="weekday-0-desc"></div>
29       </article>
30       <article class="weather-card">
31       </article>
32       <article class="weather-card">
33       </article>
34       <article class="weather-card">
35       </article>
36       <article class="weather-card">
37       </article>
38     </section>
39   </article>
40
41   <article id="weather-error-content" class="p-2">
42     <p>Søg efter vejret i en by...</p>
43   </article>
44 </section>
45
46 <hr>
47 <section id="info-section">
48   <div class="section-info-style">
49     <div class="info-style" style="background-color: #f0f0f0; padding: 10px; border-radius: 5px;">
50       <p>Under varslinger er det muligt at få oplysninger om indberettede varslinger</p>
51     </div>
52   </div>
53 </section>
```

```
15 Geo.getTOWN()
16   .then(function(data) {
17     getWeather(data);
18   })
19   .catch(function(err) {
20     console.log(err);
21   });
22
23 $("#search-form").submit(function() {
24   event.preventDefault();
25
26   $("#weather-no-content").hide();
27   $("#search-btn").trigger("blur");
28
29   const city = $("#search-text").val();
30
31   getWeather(city);
32 });
33
34 function getWeather(city) {
35   const url = API_URL + "forecast?q=" + city + "&APPID=" + API_KEY;
36   const cityUpper = city.charAt(0).toUpperCase() + city.slice(1);
37   $("#search-text").val(cityUpper);
38   $("#data-target-city").text(cityUpper);
39
40   $("#weather-loading-content").show();
41   $("#weather-success-content").hide();
42   $("#weather-error-content").hide();
43
44   setTimeout(function() {
45     $.getJSON(url, function(data) {
46       const filteredWeatherData = filterWeatherData(data);
47       const parsedWeatherData = parseWeatherData(filteredWeatherData);
48       setWeatherData(parsedWeatherData);
49     });
50   }).done(function() {
51     $("#weather-success-content").show();
52   }).fail(function() {
53     $("#weather-error-content").show();
54     console.log("error");
55   }).always(function() {
56     $("#weather-loading-content").hide();
57   });
58 }, 5000);
59
60 function filterWeatherData(data) {
61   // callback for ready()
62   // callback for then()
63 }
```

Figur 8 - Vejrudsigten HTML og JavaScript kode

Ved indlæsning af vejrudsigt siden, laves et kald til getTOWN. Denne funktion er beskrevet nedenfor, men den sørger for at hente et bynavn på en given lokation. Hvis det ikke er muligt at fremfinde et bynavn fremkommer muligheden for indtastning af et bynavn. Klik på "Hent" udfører et submit på siden, som laver et kald til getWeather funktion. Ved brug af jQuery's getJSON laves et AJAX request til OpenWeatherMap API'et.

Geo.js udstiller en funktion `getTown`, der henter bynavn ud fra den lokation man befinder sig på, hvis man er på en mobil enhed. Sidder man på en PC, vil det være IP-adressen der benyttes. Dette er muligt ved at benytte HTML5 API'et `navigator.geolocation`, der returnerer længde og breddegrader.



```
1  var Geo = {};  
2  
3  $(document).ready(function() {  
4      const API_URL = "https://eu1.locationiq.com/v1/";  
5      const API_KEY = "6147b594e590c1";  
6      const API_FORMAT = "json";  
7  
8      Geo.getTown = function() {  
9          return new Promise( executor: function(resolve, reject) {  
10             if (navigator.geolocation) {  
11                 navigator.geolocation.getCurrentPosition(  
12                     successCallback: function(pos) {  
13                         resolve(pos);  
14                     },  
15                     errorCallback: function(err) {  
16                         reject(err);  
17                     }  
18                 );  
19             }  
20             }).then( onfulfilled: function(pos) {  
21                 return new Promise( executor: function(resolve, reject) {  
22                     const url =  
23                         API_URL +  
24                         "reverse.php?key=" +  
25                         API_KEY +  
26                         "&lat=" +  
27                         pos.coords.latitude +  
28                         "&lon=" +  
29                         pos.coords.longitude +  
30                         "&format=" +  
31                         API_FORMAT;  
32  
33                     $.getJSON(url, function(data) {  
34                         resolve(data.address.town);  
35                     }).fail(function(err) {  
36                         reject(err);  
37                     });  
38                 });  
39             });  
40         });  
41     });
```

Figur 9 - Geolocation JavaScript kode

På baggrund af dette svar, laves et kald til LocationIQ, der kan returnere et bynavn. Kan et bynavn ikke findes, returneres en fejl.

## VARSLINGER (ALERTS)

På denne side er der lavet 2 ure til at vise aktuell tid i henholdsvis UTC og lokal tid. Under urene er en liste, som skal kunne vise aktive varslinger. Kode for denne del er ikke implementeret. Koden til at opdatere klokkeslæt og dato, ligger i alerts.js

```
1 <article>
2 <header>
3 <h1>Varslinger</h1>
4 </header>
5
6 <section class="d-flex justify-content-start">
7 <div class="m-3">
8 <h6 class="text-center">Lokal tid</h6>
9 <div>
10 <svg xmlns="http://www.w3.org/2000/svg"
11 <!-- xlink:href="http://www.w3.org/1999/xlink"
12 viewBox="-1024 -1024 2048 2048">
13 <!-- <title>Swiss Railway Clock</title>
14 <!-- <defs>
15 <!-- <circle class="bg" r="1024"/>
16 <!-- <use xlink:href="#face" class="fe"/>
17 <!-- <use xlink:href="#handh" id="hour" class="h1" transform="rotate(0)"/>
18 <!-- <use xlink:href="#handm" id="minute" class="h1" transform="rotate(0)"/>
19 <!-- <use xlink:href="#hands" id="second" class="h2" transform="rotate(0)"/>
20 </svg>
21 </div>
22 <h6 id="date" class="pt-2"></h6>
23 </div>
24 <div class="m-3">
25 </div>
26 </section>
27 <hr>
28 <ul id="alerts" class="list-group">
29 <li class="alert alert-success" role="alert">
30 Ingen varslinger
31 </li>
32 <li class="alert alert-warning" role="alert">
33 Advarsel
34 </li>
35 <li class="alert alert-danger" role="alert">
36 Katastrofe
37 </li>
38 </ul>
39 </article>
```

Figur 10 - Varsling HTML kode

På HTML siden er der placeret 2 svg images til visning af klokkeslæt. Ved at referere til de enkelte elementer i svg imaget, er det muligt at styre viserne på uret. Måden hvorpå elementerne kan tilgås, er ved at give dem et unikt ID. Fra JavaScript koden kan de så opdateres ved hjælp af jQuery og css selectors.

```

1  alerts.js
2  $(document).ready(function () {
3      function getPositions(utc) {
4          let date = new Date();
5
6          let hr = utc ? date.getUTCHours() : date.getHours();
7          let min = utc ? date.getUTCMinutes() : date.getMinutes();
8          let sec = utc ? date.getUTCSeconds() : date.getSeconds();
9
10         let fullDate = utc
11             ? date.getUTCDate() + "." + date.getUTCMonth() + "." + date.getUTCFullYear()
12             : date.getDate() + "." + date.getMonth() + "." + date.getFullYear();
13
14         return {
15             date: fullDate,
16             hrPosition: (hr * 360) / 12 + (min * (360 / 60)) / 12,
17             minPosition: (min * 360) / 60 + (sec * (360 / 60)) / 60,
18             secPosition: (sec * 360) / 60
19         };
20     }
21
22     function runTheClock(utc) {
23         let postfix = utc === true ? "-utc" : "";
24         let positions = getPositions(utc);
25
26         let HOURHAND = $("#hour" + postfix);
27         let MINUTEHAND = $("#minute" + postfix);
28         let SECONDHAND = $("#second" + postfix);
29
30         $("#date" + postfix).text(positions.date);
31
32         hrPosition = positions.hrPosition + 3 / 360;
33         minPosition = positions.minPosition + 6 / 60;
34         secPosition = positions.secPosition + 6;
35
36         HOURHAND.css({transform: "rotate(" + hrPosition + "deg)"});
37         MINUTEHAND.css({transform: "rotate(" + minPosition + "deg)"});
38         SECONDHAND.css({transform: "rotate(" + secPosition + "deg)"});
39     }
40
41     function runTheClocks() {
42         runTheClock( utc false);
43         runTheClock( utc true);
44     }
45
46     (function () {
47         setInterval(runTheClocks, timeout: 1000);
48     })();
49

```

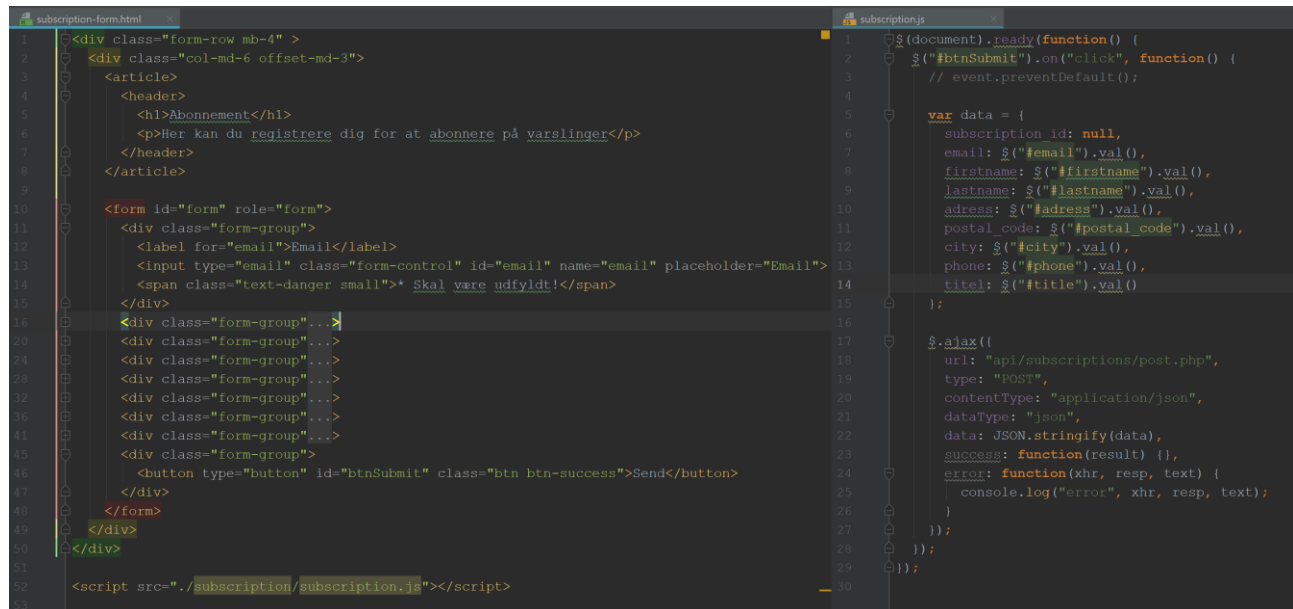
Figur 11 - Varsling JavaScript kode

Når JavaScript koden indlæses, startes en anonym funktion i form af en IIFE. IIFE står for Immediately Invoke Function Expression, og svarer til et funktionskald. Dette sætter en timer i gang hvert sekund, som kalder en der opdaterer viserne på urene.



## ABONNEMENT (SUBSCRIPTION)

Indtastningsfelterne på abonnement siden er indlejret i en form. Submit knappen trigger via en eventhandler, lavet ved hjælp af jQuery en funktion. Denne funktion laver et json request ud fra værdierne i indtastningsfelterne. Dette sendes til post operation i REST API'et, der sørger for at oprette en post i databasen, så brugeren bliver registreret til at modtage varslinger.



## 7.4 Vejrportalen – Backend

Brugere kan registrere sig for at modtage varslinger. For at kunne gemme registreringerne, er der lavet et REST API i PHP, som gemmer data i en MySQL database. Følgende er indholdet af api mappen beskrevet.

## SHARED MAPPEN

Her er samlet funktionalitet som er generel for API'et.

## API\_CONSTANTS.PHP

For at samle information et sted om API'et, så man f.eks. ikke skal ind i hver enkel fil og håndtere `ERROR_LEVEL`, er dette samlet i `api_constants.php`. Her vil også være information om MySQL indstillinger mail osv.

## DATABASE.PHP

Denne fil indeholder funktionalitet som benyttes i forbindelse med MySQL databasen.

### MAIL.PHP

For at kunne sende en mail når der oprettes et abonnement, er funktionalitet med mail samlet her.

### SHARED\_CONSTANTS.PHP

Her er konstanter som benyttes i hele API'et.

### MODELS MAPPEN

Der er 2 forskellige klasser til brug for data. Klassen `response_result` bruges ved svar på et request og indeholder status ("success" eller "error"), message (beskrivende tekst) og i tilfælde af fejl, selve fejlen fra f.eks. MySQL.

Klassen `subscription` indeholder strukturen for et request abonnement.

### SUBSCRIPTIONS MAPPEN

I denne mappe er placeret de forskellige operationer som er udstillet via API'et. Det er muligt at læse oprette, opdatere og slette poster via API'et. Hver af de nævnte operationer har deres egen Url

- `vejrportalen/api/subscriptions/get.php` – henter alle poster
- `vejrportalen/api/subscriptions/get.php?id=` – henter enkelt post ud fra id
- `vejrportalen/api/subscriptions/post.php` – opretter en post
- `vejrportalen/api/subscriptions/put.php` – opdaterer en post
- `vejrportalen/api/subscriptions/delete.php?id=` – sletter en post ud fra id

Alle operationer er bygget op efter den samme skabelon og selve afviklingen af hvad der skal ske ligger i `subscription_shared.php`.

```

19 class subscription_shared
20 {
21     private $table_name = "subscriptions";
22
23     public function __construct()
24     {
25         $this->database = new Database;
26     }
27
28     public function read($id)
29     {
30         $conn = $this->database->getConnection();
31
32         $sql = "SELECT * FROM {$this->table_name}";
33
34         if (!empty($id)) {
35             $sql .= " WHERE subscription_id = {$id}";
36         }
37
38         $result = mysqli_query($conn, $sql);
39
40         if (mysqli_num_rows($result) > 0) {
41             $subscriptions_array = array();
42
43             while ($row = $result->fetch_object()) {
44                 $subscriptions_item = new Subscription($row);
45                 array_push($subscriptions_array, $subscriptions_item);
46             }
47
48             $result = $subscriptions_array;
49         } else {
50             $result = new ResponseResult(SUCCESS, READ_NO_ROWS);
51         }
52
53         mysqli_close($conn);
54         return $result;
55     }
56 }

```

```

1 <?php
2 include_once '../shared/api_constants.php';
3 error_reporting(ERROR_LEVEL);
4
5 header("Access-Control-Allow-Origin: *");
6 header("Content-Type: application/json; charset=UTF-8");
7
8 include_once 'subscription_shared.php';
9 include_once '../models/subscription.php';
10
11 if (strcasecmp($_SERVER['REQUEST_METHOD'], 'GET') != 0) {
12     throw new Exception('Request method must be GET!');
13 }
14
15 $subscription_shared = new Subscription_shared();
16 $result = null;
17 $subscription_id = null;
18
19 if (isset($_GET['id'])) {
20     $subscription_id = $_GET['id'];
21 }
22
23 $result = $subscription_shared->read($subscription_id);
24
25 echo json_encode($result);
26

```

Figur 12 - PHP Get kode

Ovenstående figur viser til venstre PHP filen subscription\_shared.php. Funktionen som er vist, benyttes til at hente en registrering fra MySQL databasen. Funktionerne for henholdsvis opret, opdater og slet følger samme mønster. Da de enkelte funktioner benytter samme kode i form af forbindelse til databasen, er de placeret i samme fil.

Koden til højre i figuren viser get operationen. Igen vil alle de operationer der udstilles, benytte samme mønster.

## 7.5 Brugervenlighed

Ved udvikling af et website, er det vigtigt løbende at lave nogle brugervenlighedstest. Ved udførelse af en brugertest indkaldes 5 personer, med forskellig persona. Bruger placeres i et lokalt og får udleveret en række opgaver som de hver især skal gennemføre. Følgende er en række opgaver, som en bruger kunne blive stillet, for at brugerteste vejrportalen. Jeg har samlet dem fortløbende, men overfor en bruger burde de være på hver deres side, uden nummerering så de ikke ved hvor mange der er.

Der er 3 typer opgaver og jeg har lavet nogle for hver type.

#### **INDTRYK OPGAVE (IMPRESSION TASK)**

- Åbn vejrportalen i en web browser og naviger rundt på websitet. Brug et par minutter, på at danne et overblik og beskriv derefter hvad dette website kan benyttes til.

#### **EKSPLORATIV OPGAVE (EXPLORATORY TASK)**

- Vi har brug for at finde ud af hvordan vejret bliver i Hvidovre i morgen. Benyt vejrportalen, til at finde vejrudsigten.

- Det er muligt at se hvilke varslinger der er aktive. Find siden hvor varslinger vises.

#### **INSTRUERET OPGAVE (DIRECTED TASK)**

- Der findes en læringsvideo på vejrportalen. Find video'en og gennemse denne.

- Benyt abonnement siden, til at registrere dig som bruger for at modtage varslinger fra vejrportalen.

## 8 Konklusion

Mangler...

## 9 Liste over referencer

Opgave:

- Lars Larsen (2018). Projektopgave del 1:  
<https://github.com/larsk7cdk/vejrportalen/tree/master/docs>
- Lars Larsen (2018). Projektopgave del 2:  
<https://github.com/larsk7cdk/vejrportalen/tree/master/docs>
- Lars Larsen (2018). Projektopgave del 3:  
<https://github.com/larsk7cdk/vejrportalen/tree/master/docs>
- Lars Larsen (2018). Projektopgave del 4:  
<https://github.com/larsk7cdk/vejrportalen/tree/master/docs>
- Lars Larsen (2018). Projektopgave del 5:  
<https://github.com/larsk7cdk/vejrportalen/tree/master/docs>
- Lars Larsen (2018). Projektopgave del 6:  
<https://github.com/larsk7cdk/vejrportalen/tree/master/docs>
- Lars Larsen (2018). Projektopgave del 7:  
<https://github.com/larsk7cdk/vejrportalen/tree/master/docs>
- Lars Larsen (2018). Projektopgave del 8:  
<https://github.com/larsk7cdk/vejrportalen/tree/master/docs>

Bog:

- Michael Mendez (2014). The Missing Link. Open SUNY Textbooks
- Ian Wisler-Poulsen (2012). 20 Designprincipper. Grafisk Litteratur

Video:

- Doug Winnie (2017). Computer Science Principles: The Internet. Lynda.com
- Diane Cronenwett (2018). UX Foundations: Multidevice design. Lynda.com
- Ray Villalobos (2018). Bootstrap 4 Essential Training. Lynda.com
- James Williamson (2014). HTML Essential Training. Lynda.com
- Kevin Skoglund (2018). PHP Essential Training. Lynda.com
- Morten Rand-Hendriksen (2018). JavaScript Essential Training. Lynda.com
- Chris Nodder (2013). UX Foundations: Making the Case for Usability Testing. Lynda.com

Webside:

- Nick Babich (2017). 10 Tips On Typography in Web Design:  
<https://uxplanet.org/10-tips-on-typography-in-web-design-13a378f4aa0d>
- Joshua David McClurg-Genevise (2005). The Principles of Design:  
[https://studie.smartlearning.dk/pluginfile.php/410576/mod\\_resource/content/2/Digital%20Web%20Magazine%20-%20The%20Principles%20of%20Design.pdf](https://studie.smartlearning.dk/pluginfile.php/410576/mod_resource/content/2/Digital%20Web%20Magazine%20-%20The%20Principles%20of%20Design.pdf)
- The Gestalt Principles:  
<http://graphicdesign.spokanefalls.edu/tutorials/process/gestaltprinciples/gestaltprinc.htm>
- Jakob Nielsen (2012). Usability 101: Introduction to Usability:  
<https://www.nngroup.com/articles/usability-101-introduction-to-usability/>
- Xavier Renom (2018). How to do usability testing:  
<https://www.justinmind.com/blog/how-to-do-usability-tests-online-before-coding/>