Vejrportalen

VEJRET ER LIGE BLEVET BEDRE

Uddannelse sted	Smartlearning
Fag	Webprogrammering – Frontend
Underviser	Constantin Alexandru Gheorghiasa
Forfatter	Lars Larsen
Projektperiode	3. september 2018 – 9. december 2018
Dato for aflevering	5. december 2018 kl. 12.00

Titelblad

Uddannelse sted	Smartlearning
Uddannelse	Diplomuddannelsen i webudvikling
Fag	Webprogrammering – Frontend
Underviser	Constantin Alexandru Gheorghiasa
Forfatter	Lars Larsen
Projektperiode	3. september 2018 – 9. december 2018
Dato for aflevering	5. december 2018 kl. 12.00
Antal sider	34
Antal karakterer	26.102
Vejrportalen URL	https://40.127.170.50/vejrportalen/
Github URL	https://github.com/larsk7cdk/vejrportalen

Indholdsfortegnelse

Figurfortegnelse

1 Indledning

Jeg har igennem de sidste 12 uger gennemgået emner i form af teori og praksis, som handler om frontend udvikling. I dette forløb har jeg afleveret opgaver, der danner grundlag for projektopgaven. Min projektopgave er en case baseret skriftlig opgave, med det formål at omsætte teori og praksis til et samlet projekt.

2 Casebeskrivelse

Vi lever i en tid, der giver anledning til store vejrforandringer. Der er mange der har behov for at holde sig opdateret omkring vejrforholdene, kaldet brugere. Det kunne f.eks. være private virksomheder og offentlige instanser, som skal have mulighed for at få adgang til en vejrudsigt. Brugerne kan have forskellige behov for at få information her og nu, som er blevet en del af vores hverdag.

3 Problemstillingen

Jeg 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. I dette tilfælde drejer det sig om et website, hvor vejrudsigten for en given by i Danmark kan vises.
For at brugere kan hjælpe hinanden, skal det være muligt at abonnere på vejrvarslinger, som andre brugere har registreret. Aktive vejrvarslinger skal kunne ses på websitet.
Hvis brugeren har lokation aktiveret på sin enhed, skal byen for den aktuelle lokation blive benyttes, ellers er det muligt at indtaste en by.
For at lave en MVP (minimum viable product) for websitet, laves en lav risiko baseret tilgang.
Derfor stilles følgende mål i prioriteret rækkefølge

- 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

4 Problemformulering

Hvordan kan jeg lave et website, så det bliver muligt for brugere at se og abonnere på vejrdata?

- Hvilken arkitektur og teknologier skal benyttes for bedst at løse opgaven?
- Hvordan opbygges et design, som giver en god brugeroplevelse?
- Hvordan kan websitet understøtte både PC og mobile enheder?

5 Teori

5.1 Design principper

Til design af websites kan Gestalt lovene benyttes. Gestalt lovene er udarbejdet af tyske psykologer omkring 1920. Psykologerne ville finde ud af menneskers sanser, og gjorde dette ud fra en række forsøg, som beskæftiger sig med alle sanser. Denne form for psykologi kaldes perceptionspsykologi. Jeg vil her kort beskrive elementerne 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 placeres i nærheden af hinanden. Elementer som ikke hører sammen placeres længere fra hinanden.

LOVEN OM LIGHED

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

Det loven om lighed dækker, er emner som form, farve, størrelse, placering osv.

En navigation menu vil gå igen på alle websider og være placeret det samme sted. De enkelte menupunkter står i nærheden af hinanden og er derfor i et samspil med loven om nærhed. I en menu vil det aktive menupunkt, som regel også have en anden farve for at fremhæve det.

Det er ikke godt at benytte understregninger i en tekst på et website. Det 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 indrammer 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 det skaber overblik.

LOVEN OM FORBUNDETHED

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

Elementer på et website kan være forbundet på forskellige måder. Enten i form af en linje imellem dem, eller baggrundsfarven hvor de er placeret indrammer det.
Tabs dækkes også af denne lov. Her vil det aktive tab få samme farve, som selve indholdet der bliver vist. På denne måde bliver de 2 elementer forbundet.

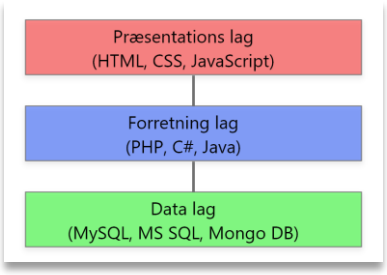
LOVEN OM FIGUR OG BAGGRUND

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

Det er vigtigt at en figur eller 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 figur/baggrund.

5.2 Arkitektur

Der findes forskellige arkitektur modeller afhængigt af hvilken opgave der skal løses. Her er 3-lags arkitekturen beskrevet.



Figur 1 - 3-lags arkitektur

PRÆSENTATIONSLAG

Præsentationslaget er det brugeren ser i en browser. Dette lag er baseret på HTML, CSS og JavaScript kode.

FORRETNINGSLAG

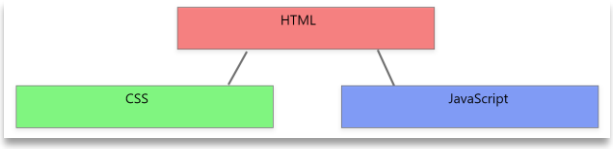
Indeholder forretningsregler og core funktionalitet, som vil blive stillet på baggrund af en opgave. Dette kan skrives i PHP, C# eller Java.

DATA LAG

Data laget sørger for opbevaring af data i en database. Dette kan være MySQL, MS SQL eller Mongo DB.

5.3 Frontend

I forhold til 3-lags arkitekturen, består præsentation laget af 3 ansvarsområder, som vist på figuren



Figur 2 - Præsentation lag

HTML

HTML benyttes til at opbygge selve siden, som vises for brugeren. HTML5 er den seneste standard og er udvidet med HTML tags, der semantisk beskriver betydningen i forhold til f.eks. skærm oplæsere. Det er 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 tilføje en role

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

For at informere en browser om den benytter HTML5 angives følgende øverst

```
<!DOCTYPE html>
```

En HTML side består af en head sektion, hvor man angiver meta data, som sprog og titel. Det er også her der angives links til at hente stylesheets.
Næste sektion er body'en. Det er her selve indholdet der vises på websitet placeres.
Nederst i body'en vil hentning af JavaScript placeres. Dette for hurtigere at vise indhold for brugeren, før alt JavaScript er hentet.

CSS

CSS står for Cascading Style Sheet og beskriver hvordan HTML tags på et website vises. Seneste standard er version 3.0. Den er blevet udvidet med transitioner og animationer, samt media queries. Media queries benyttes 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 style klassen skal benyttes på flere HTML sider, kan den laves i en ekstern fil med ekstensionen .css.

jQuery

jQuery er et lille, hurtigt og funktions rigt library til at understøtte JavaScript. Opgaver til at gennemløbe og finde elementer 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 selektere. Skal en tekst i et <p> tag med et ID som hedder ”city” opdateres, kan dette gøres på følgende måde

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

BOOTSTRAP

Bootstrap er et komponent library til HTML sider. Det indeholder knapper, paneler, tekstbokse og andre standardkomponenter. Fordelen ved bootstrap er at det virker ens på alle browsere. En af de populære komponenter er navigation komponenten, som både virker i desktop og mobil visning. Måden hvorpå bootstrap benyttes, er ved hjælp af HTML tags klasser. 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 også et grid system til layout, der er opdelt i 12 kolonner. På denne måde kan man lave sit layout, efter samme principper som i en avis.

Den seneste version understøtter muligheden for at benytte flex. Flex er en CSS3 feature, og udgør en ny måde at lave layout på. Det er særdeles velegnet til at lave responsive sider, da det understøtter funktioner til at stakke, wrappe, resize og andre optimeringer ifm. responsivt design.

For at benytte flex skal det erklæres på en css klasse

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

FONTAWESOME

Fontawesome er et ikon library, som indeholder ikoner i forskellige kategorier. Ikonerne er i svg format og kan skaleres i forskellige størrelser. For at benytte et ikon indsættes dette i et <i> tag. Her er en spinner

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

5.4 Backend

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, skyldes 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 startes 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 koden skal benyttes flere steder. Referencer importeres som

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

MySQL

MySQL er et populært 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

OpenWeatherMap er et API 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

5.5 Kommunikationsstrategi

Kommunikationsstrategi er en plan for ”hvem gør hvad, hvornår og hvorfor”. Det er vigtigt at den konkret beskriver hvilke budskaber, målgrupper og kanaler der skal understøttes.

5.6 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. Brugerne placeres i et lokale og får udleveret en række opgaver, som de hver især skal gennemføre under observation. Der er 3 typer opgaver, som hver beskriver en række handlinger. Opgavetyperne er

- Indtryk opgave (Impression task)
- Eksplorativ opgave (Exploratory task)
- Instrueret opgave (Directed task)

6 Analyse

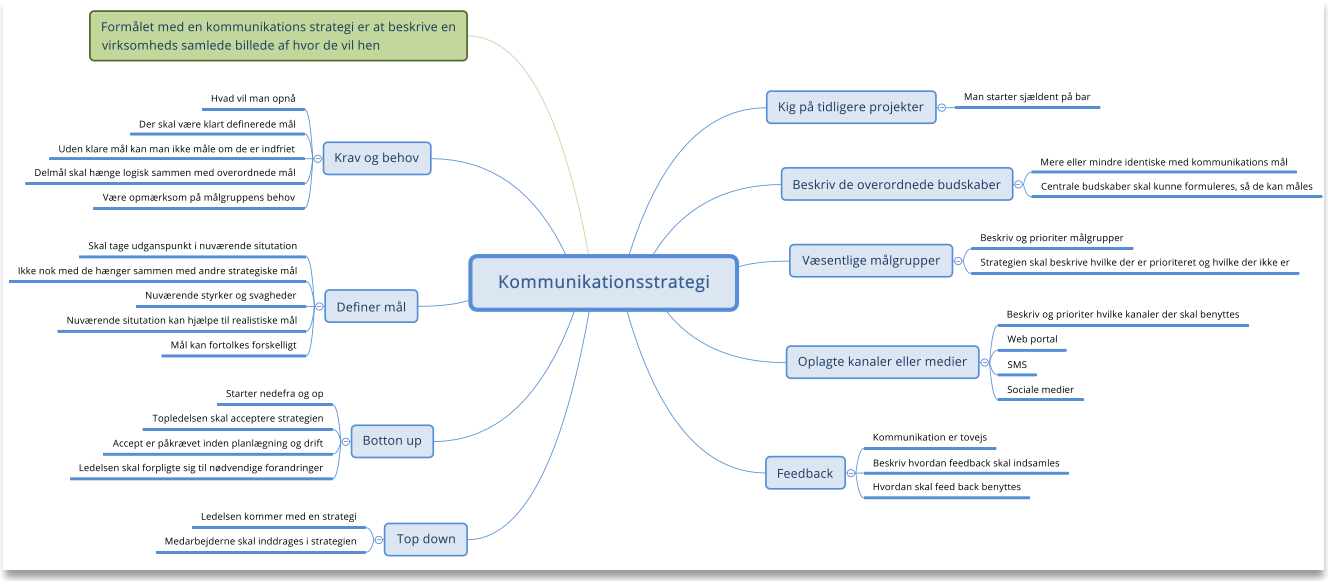
Der er forskellige måder at tilbyde vejrdata til brugere på. Valget er faldet på et website, da dette vil kunne tilgås fra såvel PC, som mobile enheder i form af IPads og mobiltelefoner.
Arkitekturen er på baggrund af tidligere erfaringer, valgt som en 3-lags arkitektur.
Til udvikling af frontend delen, findes i dag mange forskellige typer frameworks. Her kan nævnes Angular, React og Vue. Udfordringen ved at benytte disse frameworks er en høj indlæringskurve.
På denne baggrund er der valgt HTML5 til opbygning af HTML siderne. På styling delen, kan der ofte være udfordringer med at få komponenter til at se ens ud i de forskellige browsere. Derfor er der valgt en kombination af bootstrap og CSS3. På JavaScript delen, er jQuery valgt for nemmere at kunne manipulere med dynamiske data og udføre kald til API.

7 Løsningsforslag

Websitet har jeg valgt at kalde ”Vejrportalen”, som benyttes fremadrettet.

7.1 Kommunikationsstrategi

På baggrund af teorien om kommunikationsstrategi, har jeg valgt at lave en 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 3 - 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 vejrdata og vejrvarslinger.

MÅLGRUPPER

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

KANALER FOR KOMMUNIKATION

Det er vigtigt at vejrportalen kan nås på forskellige kanaler fra de enkelte målgrupper. Det skal som et minimum kunne tilgås fra

- Website
- Tablets og IPads
- Mobile enheder
- Notifikationer i form af e-mail eller sms

FEEDBACK

For løbende at kunne foretage forbedringer og tilføje nye funktioner er feedback vigtig. Der skal løbende opfordres til dialog mellem brugere og ansvarlige hos vejrportalen.

OVENSTÅENDE KRAV OPSAT I SKEMA FORM

Overordnede mål i skema:

Formål	**Budskaber**	**Målgrupper**	**Medier**	**Ansvarlig**	**Ressourcer**	**Succeskriterier**
At skabe en vejrportal med vejrudsigter og vejrvarslinger	Opdaterede lokale vejrudsigter Vejrvarslinger af høj kvalitet	Private virksomheder Offentlige instanser	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 vejrdata	Web portal	Vejr API	Udvikling af løsning til visning af vejrdata	Vejrdata er tilgængelige 24 timer i døgnet	Der laves en meningsmåling hvert halve år
Private virksomheder Offentlige instanser	At kunne se og indgive vejrvarslinger	Web portal SMS E-mail	Vejrportalen	Udvikling af løsning til vejrvarslinger	Muligt at kunne indgive vejrvarslinger	Mulighed for at kunne give 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 viderebringes
- **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 vejrportalen skal understøtte.

Figur 4 - Vejrportalen indhold

Top indeholder et logo, som giver brugeren et vartegn. Endvidere indeholder den navigation til vejrportalens andre sider.

Vejrudsigten vil være vejrportalens landingpage. Her kan brugeren søge på en given by og få en 5-døgnsudsigt. Ligeledes kan man læse om de andre sider der findes, samt se en instruktions video.

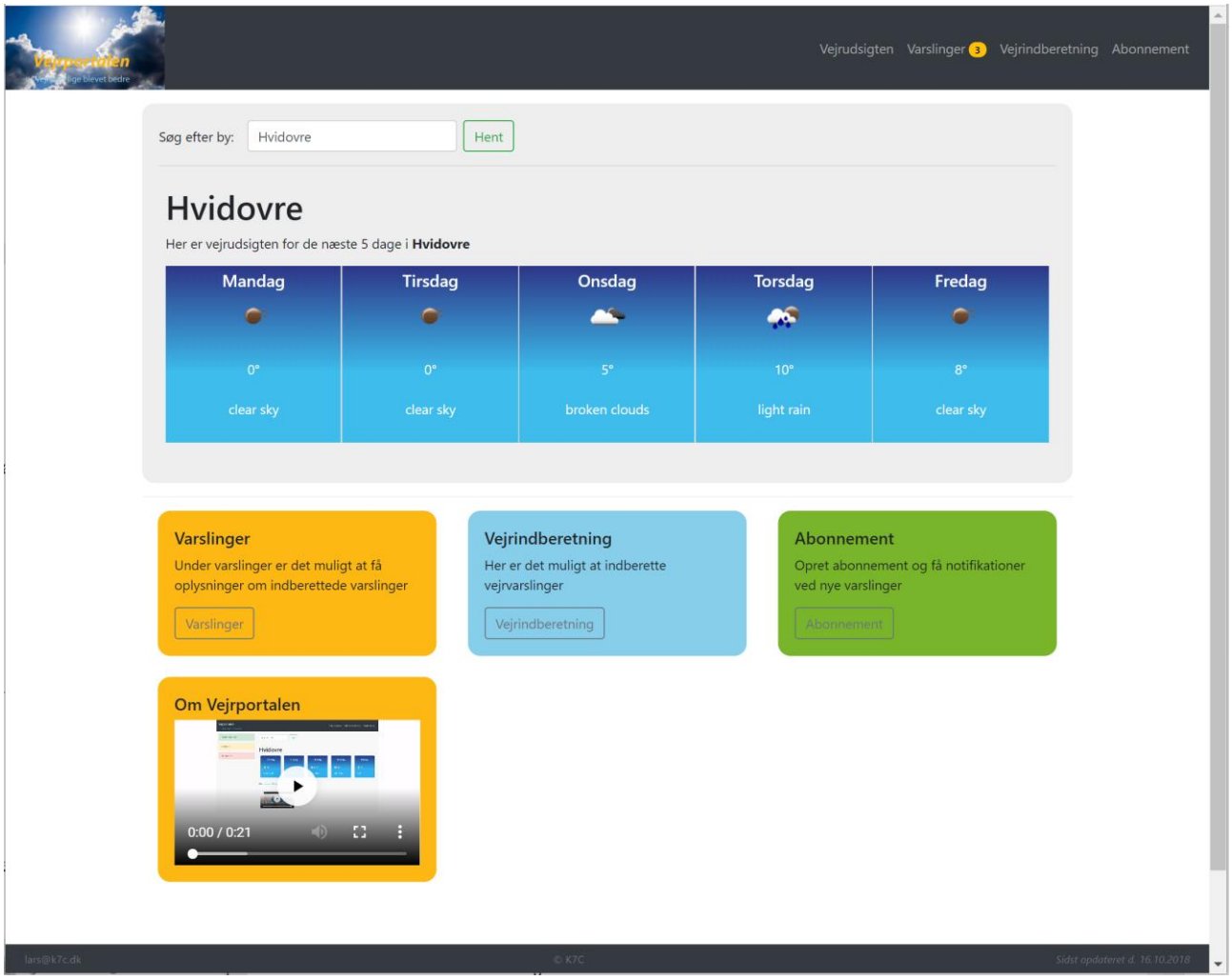
Varslinger giver et overblik over aktive varslinger der er indberettet af brugerne.

Vejrindberetning giver mulighed for at kunne indberette en varsling.

Abonnement siden, er hvor brugere kan registrere sig for at modtage varslinger når de oprettes.

Vejrudsigten

Nedenstående figur viser vejrportalens forside. Dette har været igennem flere iterationer, hvor det løbende er blevet rettet til med de erfaringer der er opnået undervejs.
Designet har taget udgangspunkt i Gestalt lovene.



Figur 5 - Vejrudsigten design

Logo

For at give vejrportalen et vartegn, er der placeret et logo i hjørnet øverst til venstre. Dette benyttes ved omtale af vejrportalen, for at give brugerne noget genkendeligt. **Her benyttes loven om figur og baggrund**.

NAVIGATION

Navigationen er placeret til højre i toppen af websitet. 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.
Ved aktive vejrvarslinger, er der ved siden af linket placeret et badge med antallet af aktive varslinger. **Her benyttes loven om lighed**.

SØGEFELT OG 5-DØGNSUDSIGT

For at fremsøge en vejrudsigt for en given by, placeres denne funktion øverst på siden, så den er tilgængelig og synlig for brugeren uanset enhed. Har enheden adgang til lokationen, hentes vejrdata når siden vises. Bynavnet vil blive vist i søgefeltet, og kan rettes hvis der ønskes en anden by.
Vejrdata for de enkelte dage er placeret i et kort. Kortet indeholder dag, vejr ikon, temperatur og beskrivelse af vejret.
Søgefeltet og visningen af 5-døgnsudsigten er indrammet med en baggrundsfarve, for at vise de hænger sammen. **Her benyttes loven om lighed og loven om lukkethed**.

BESKRIVELSE AF ANDRE SIDER

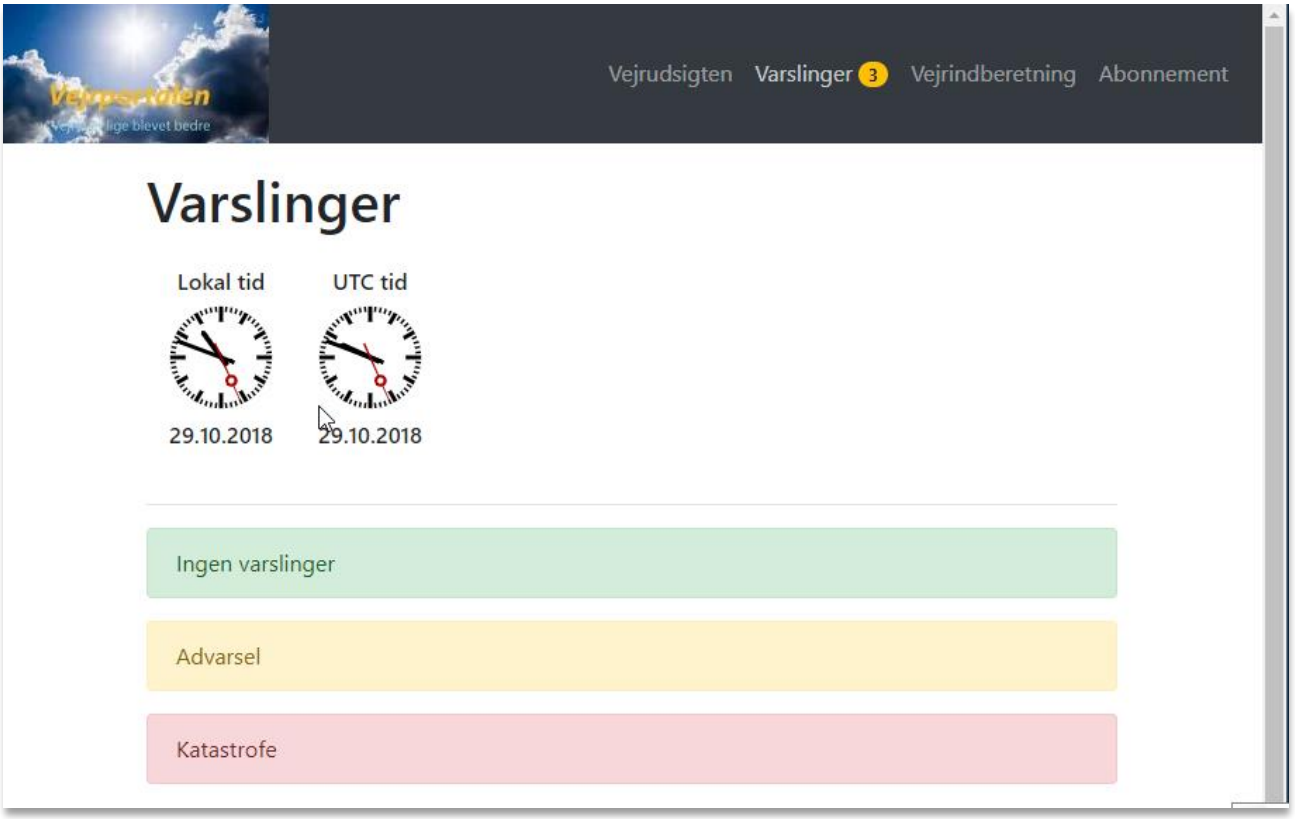
Under 5-døgnsudsigten er information om de forskellige sider der findes på vejrportalen placeret. Farverne på de enkelte bokse er taget fra logoet så der er genbrug af farvevalg. Det giver en kontrast til baggrunden. I de enkelte bokse er der et link, som giver mulighed for at skifte til den omtalte side. **Her benyttes loven om figur og baggrund**.

SIDEFOD

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

Varslinger

På varsling siden vises aktive varslinger. Endvidere er der placeret 2 ure, til visning af klokkeslæt og dato.



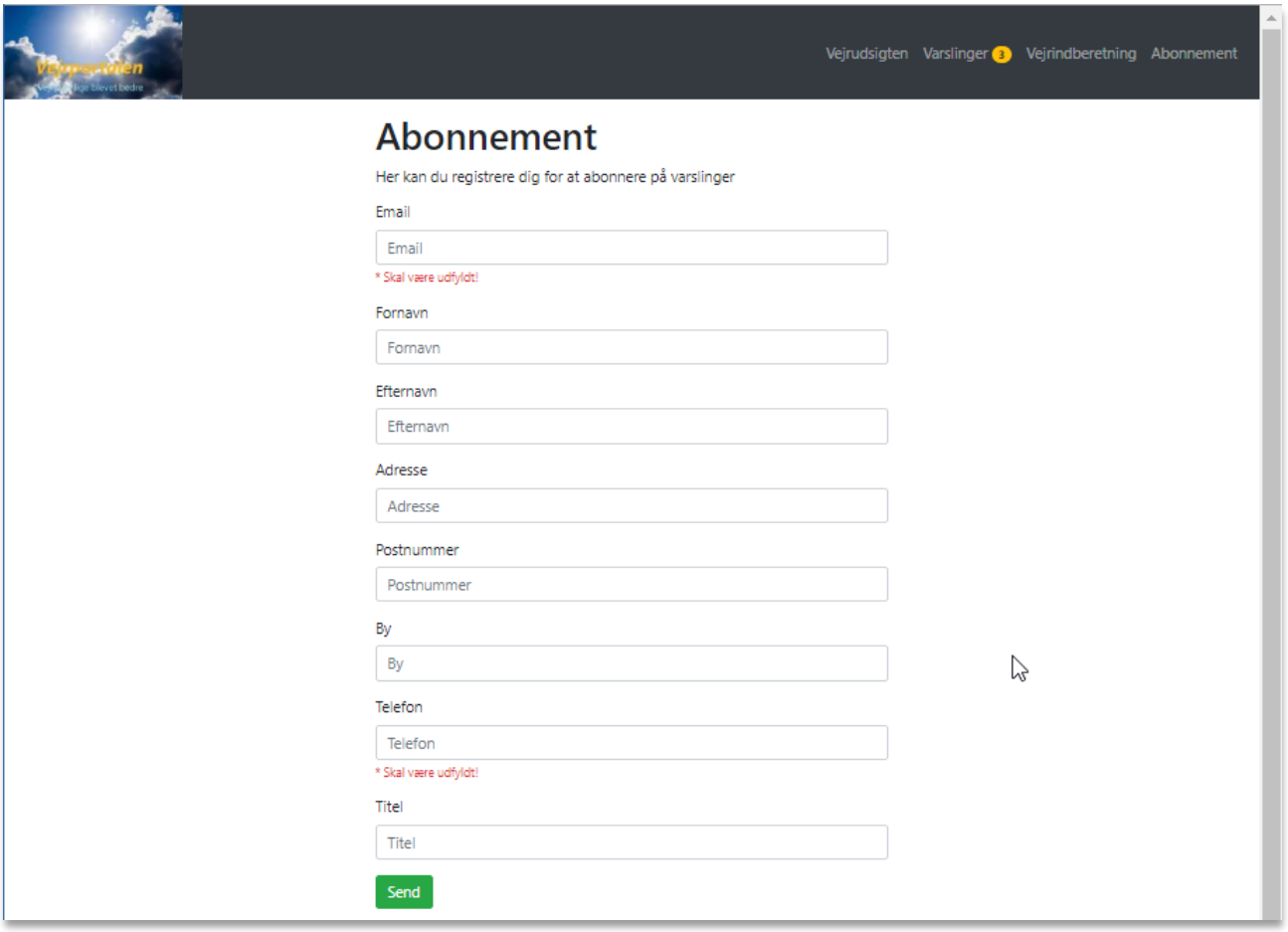
Figur 6 – Varslinger design

De enkelte varslinger har 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 kan en bruger registrere sig og dermed modtage varslinger på e-mail eller sms. Siden er responsiv og kan benyttes på desktop og mobil. E-mail og telefon nummer er 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.



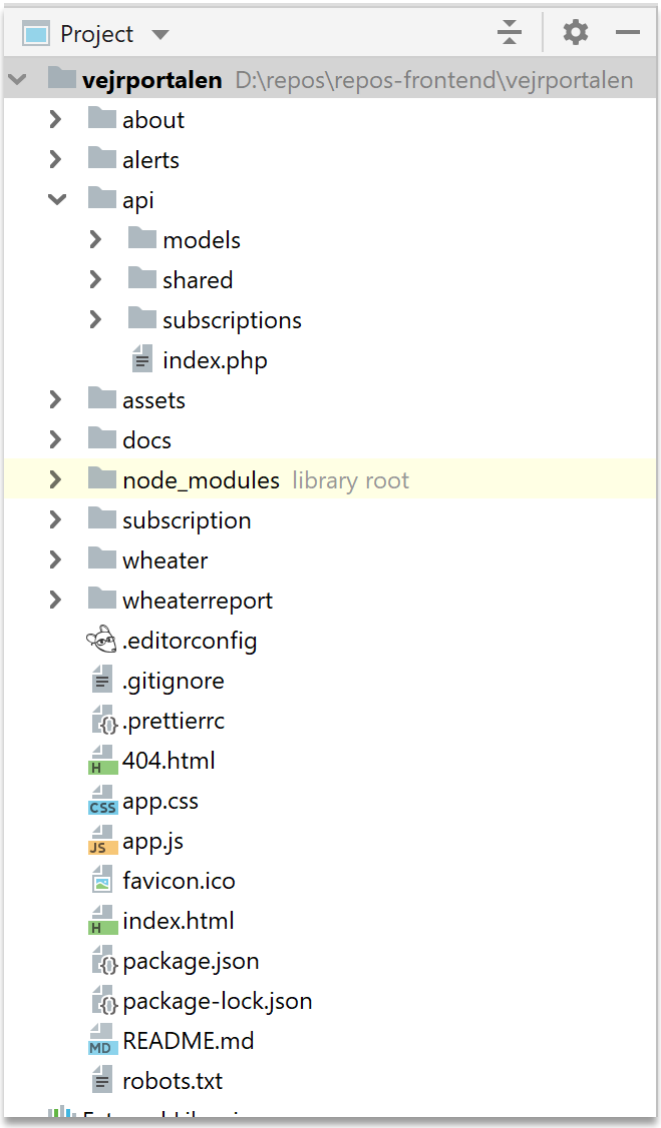
Figur 7 - Abonnement design

7.3 Vejrportalen – Frontend

FILSTRUKTUR

Kildekode for både frontend og backend er samlet i samme løsning. Backend koden er placeret under mappen api.

For frontend 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 8 – 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.

```
<!DOCTYPE html>
<html lang="da">
<head...>
<body>
<header class="container-fluid p-0" role="banner">
  <nav class="navbar navbar-expand-md navbar-dark bg-dark p-0 pr-3" role="navigation">
    <div>
      <img src="./assets/images/vejrportalen_logo.jpg" height="100px">
    </div>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse"
            aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarCollapse">
      <ul id="nav-items" class="navbar-nav ml-auto text-right">
        <li class="nav-item">
          <a id="nav-wheater#wheater" data-target="nav-click" class="nav-link" href="#">Vejrudsigten</a>
        </li>
        <li class="nav-item"...>
        <li class="nav-item"...>
        <li class="nav-item">
          <a id="nav-subscription#subscription-form" data-target="nav-click" class="nav-link" href="#">Abonnement</a>
        </li>
      </ul>
    </div>
  </nav>
</header>

<article class="container py-3 fill">
  <section>
    <main id="content" class="fill" role="main">
    </main>
  </section>
</article>


<footer class="container-fluid bg-dark fixed-bottom" role="contentinfo">
  <section class="d-flex justify-content-between p-2">
    <a href="mailto:lars@k7c.dk" class="text-muted tex">lars@k7c.dk</a>
    <span class="text-muted">© K7C</span>
    <span class="text-muted font-italic">Sidst opdateret d. 16.10.2018</span>
  </section>
</footer>

<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src="https://code.jquery.com/jquery-3.3.1.min.js" integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV
        crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-ChfqqxuZUCnJSK3+
        crossorigin="anonymous"></script>
<script src="./app.js" defer></script>

</body>
```

Figur 9 – index.html markup

Til brug for navigation er der lavet en unordered list. Hver list item har et tag med hver deres unikke ID. ID'et benyttes til jQuery's selekter, så ved brug af jQuery load funktion, asynkront kan vise html.

Dette har flere fordele. Index.html siden hentes kun en gang og ved sideskift, er der ikke oplevelsen af at siden laver et blink. Ved sideskift er animation på opacity. Dette giver fornemmelsen af en glidende overgang, når der navigeres til en ny side.
Generelt på alle siderne, er benyttet tags der semantisk beskriver de enkelte sektioner af HTML strukturen. Dette øger læsevenligheden af koden når man skimmer den igennem. Det medfører at man hurtigt kan danne sig et overblik over header, indhold af siden (article) og sidefod sektionerne.
Nederst på siden hentes 3. parts JavaScript filerne fra CDN's. Fordelen ved CDN's er, når filerne først er hentet en gang bliver de cachet i browseren og skal dermed ikke hentes igen.
Egen udviklede JavaScript filer, hentes med en "defer" egenskab, hvilket muliggør asynkron hentning og dermed øger brugervenligheden i form af siden vises hurtigere.

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

```
$(document).ready(function () {
  loadAlerts();
  changePage( target: "wheater#wheater");
  // changePage("wheater#tabs");
  // changePage("subscription#subscription-form");
  setupEvents();

  window.cp = changePage;

  // functions
  function loadAlerts() {
    $("#alerts").load("./alerts/alerts.html");
  }

  function changePage(target) {
    var feature = target.split("#")[0];
    var page = target.split("#")[1];

    $("#content").load(feature + "/" + page + ".html", function () {
      animateOpacity( selector: "#content", speed: 500, startOpacity: 0.5);

      var id = "nav-" + feature;
      setActive(id);
    });
  }

  function setActive(id) {
    $("#nav-items li a").removeClass("active");
    $("#" + id).addClass("active");
  }

  function animateOpacity(selector, speed, startOpacity) {
    var s = $(selector);
    s.css({opacity: startOpacity});
    s.animate( keyframes: {opacity: "1"}, speed);
  }

  // setup events
  function setupEvents() {
    $("a[data-target='nav-click']").click(function (event) {
      var id = event.currentTarget.id;
      changePage(id.substring(4, id.length));
    });
  }
});
```

Figur 10 - app.js kode

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.

```
<section id="search-section" class="section-style">
  <div class="form-group">
    <form id="search-form" class="form-inline" role="search">
      <label for="search-text" class="pr-3">Søg efter by:</label>
      <input id="search-text" class="form-control mr-2" type="text" placeholder="f.eks. Hvidovre"...>
      <button id="search-btn" type="submit" class="btn btn-outline-success my-2 my-sm-0">Hent</button>
    </form>
  </div>
  <hr>

  <article id="weather-loading-content" class="p-2"...>


  <article id="weather-success-content" class="p-2">
    <header...>

    <section id="weather-forecast" class="weather-forecast text-white">
      <article class="wheater-card">
        <h5 id="weekday-0-name"></h5>
        <img id="weekday-0-img"><br>
        <span id="weekday-0-temp"></span><br>
        <span id="weekday-0-desc"></span><br>
      </article>
      <article class="wheater-card"...>
      <article class="wheater-card"...>
      <article class="wheater-card"...>
      <article class="wheater-card"...>
    </section>
  </article>

  <article id="weather-error-content" class="pb-2">
    <p>Søg efter vejret i en by...</p>
  </article>

</section>
<hr>
<section id="info-section" >
  <div class="section-info-style">
    <div class="info-style" style="...">
      <h5>Varslinger</h5>
      <p>Under varslinger er det muligt at få oplysninger om indberettede varslinger</p>
      <button class="btn btn-outline-secondary" onclick="window.cp('alerts#alerts')">Varslinger</button>
    </div>
    <div class="info-style" style="...">
      <h5>Vejrindberetning</h5>
      <p>Her er det muligt at indberette vejrvarslinger</p>
      <button class="btn btn-outline-secondary" onclick="window.cp('wheaterreport#wheaterreport-form')">Vejrindberetning</button>
    </div>
    <div class="info-style" style="...">
      <h5>Abonnement</h5>
```

Figur 11 – weather.html markup

Ved indlæsning af vejrudsigt siden, laves et kald til getTown. Denne funktion henter et bynavn på en given lokation. Hvis det ikke er muligt at fremfinde et bynavn, vises et felt til indtastning af et bynavn. Klik på "Hent" udfører et submit på siden, som laver et kald til getWeather funktionen. Ved brug af jQuery's getJSON laves et AJAX request til OpenWeatherMap API'et.

```
Geo.getTown()
  .then(function(data) {
    getWeather(data);
  })
  .catch( onrejected: function(err) {
    console.log(err);
  });

$("#search-form").submit(function() {
  event.preventDefault();

  $("#weather-no-content").hide();
  $("#search-btn").trigger("blur");

  const city = $("#search-text").val();

  getWeather(city);
});

function getWeather(city) {
  const url = API_URL + "forecast?q=" + city + "&appid=" + API_KEY + "&units=" + API_UNITS;

  const cityUpper = city.charAt(0).toUpperCase() + city.slice(1);
  $("#search-text").val(cityUpper);
  $("[data-target=city]").text(cityUpper);

  $("#weather-loading-content").show();
  $("#weather-success-content").hide();
  $("#weather-error-content").hide();

  setTimeout( handler: function() {
    $.getJSON(url, function(data) {
      const filteredWeatherData = filterWeatherData(data);
      const parsedWeatherData   = parseWeatherData(filteredWeatherData);
      setWeatherData(parsedWeatherData);
    })
      .done(function() {
        $("#weather-success-content").show();
      })
      .fail(function() {
        $("#weather-error-content").show();
        console.log("error");
      })
      .always(function() {
        $("#weather-loading-content").hide();
      });
  }, timeout: 500);
}
```

Figur 12 - wheater.js kode

Geo.js udstiller funktionen getTown der henter bynavn ud fra den lokation man befinder sig på, hvis man er på en mobil enhed. Benyttes en PC, vil det være IP-adressen der benyttes. Her benyttes HTML5 API'et navigator.geolocation, der returnerer længde og breddegrader.

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

```
var Geo = {};

$(document).ready(function() {
  const API_URL = "https://eu1.locationiq.com/v1/";
  const API_KEY = "6147b594e590c1";
  const API_FORMAT = "json";

  Geo.getTown = function() {
    return new Promise( executor: function(resolve, reject) {
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(
          successCallback: function(pos) {
            resolve(pos);
          },
          errorCallback: function(err) {
            reject(err);
          }
        );
      }
    }).then( onfulfilled: function(pos) {
      return new Promise( executor: function(resolve, reject) {
        const url =
          API_URL +
          "reverse.php?key=" +
          API_KEY +
          "&lat=" +
          pos.coords.latitude +
          "&lon=" +
          pos.coords.longitude +
          "&format=" +
          API_FORMAT;

        $.getJSON(url, function(data) {
          resolve(data.address.town);
        }).fail(function(err) {
          reject(err);
        });
      });
    });
  };
});
```

Figur 13 – geo.js kode

VARSLINGER (ALERTS)

På denne side er der lavet 2 ure til at vise aktuel 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

```
<article>
  <header>
    <h1>Varslinger</h1>
  </header>

  <section class="d-flex justify-content-start">
    <div class="m-3">
      <h6 class="text-center">Lokal tid</h6>
      <div>
        <svg xmlns="http://www.w3.org/2000/svg"
             xmlns:xlink="http://www.w3.org/1999/xlink"
             viewBox="-1024 -1024 2048 2048">
          <title>Swiss Railway Clock</title>
          <defs...>
          <circle class="bg" r="1024"/>
          <use xlink:href="#face" class="fc"/>
          <use xlink:href="#handh" id="hour" class="h1" transform="rotate(0)"/>
          <use xlink:href="#handm" id="minute" class="h1" transform="rotate(0)"/>
          <use xlink:href="#hands" id="second" class="h2" transform="rotate(0)"/>
        </svg>
      </div>
      <h6 id="date" class="pt-2"></h6>
    </div>
    <div class="m-3"...>
  </section>
  <hr>
  <ul id="alerts" class="list-group">
    <li class="alert alert-success" role="alert">
      Ingen varslinger
    </li>
    <li class="alert alert-warning" role="alert">
      Advarsel
    </li>
    <li class="alert alert-danger" role="alert">
      Katastrofe
    </li>
  </ul>
</article>

<script src="./alerts/alerts.js"></script>
```

Figur 14 – alerts.html markup

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 opdateres viserne ved at der udregnes en position af viserne.

Når alerts.js indlæses startes en anonym funktion i form af en IIFE. IIFE står for Immediately Invoke Function Expression, og svarer til et funktionskald der kaldes med det samme. Dette sætter en timer i gang, som trigger runTheClock funktionen med et sekund interval. Ved hvert funktions kald opdateres viserne på urene.

```
$(document).ready(function () {
  function getPositions(utc) {
    let date = new Date();

    let hr = utc ? date.getUTCHours() : date.getHours();
    let min = utc ? date.getUTCMinutes() : date.getMinutes();
    let sec = utc ? date.getUTCSeconds() : date.getSeconds();

    let fullDate = utc
      ? date.getUTCDate() + "." + date.getUTCMonth() + "." + date.getUTCFullYear()
      : date.getDate() + "." + date.getMonth() + "." + date.getFullYear();

    return {
      date: fullDate,
      hrPosition: (hr * 360) / 12 + (min * (360 / 60)) / 12,
      minPosition: (min * 360) / 60 + (sec * (360 / 60)) / 60,
      secPosition: (sec * 360) / 60
    };
  }

  function runTheClock(utc) {
    let postfix = utc === true ? "-utc" : "";
    let positions = getPositions(utc);

    let HOURHAND = $("#hour" + postfix);
    let MINUTEHAND = $("#minute" + postfix);
    let SECONDHAND = $("#second" + postfix);

    $("#date" + postfix).text(positions.date);

    hrPosition = positions.hrPosition + 3 / 360;
    minPosition = positions.minPosition + 6 / 60;
    secPosition = positions.secPosition + 6;

    HOURHAND.css({transform: "rotate(" + hrPosition + "deg)"});
    MINUTEHAND.css({transform: "rotate(" + minPosition + "deg)"});
    SECONDHAND.css({transform: "rotate(" + secPosition + "deg)"});
  }

  function runTheClocks() {
    runTheClock( utc: false);
    runTheClock( utc: true);
  }

  (function () {
    setInterval(runTheClocks,  timeout: 1000);
  })();
});
```

Figur 15 – alerts.js kode

ABONNEMENT (SUBSCRIPTION)

Indtastningsfelterne på abonnement siden er indlejret i en form. Submit knappen trigger en funktion der laver et json request ud fra værdierne i indtastningsfelterne. Dette request sendes til post operationen i REST API'et, der sørger for at oprette en post i databasen. Brugeren bliver registreret til at modtage varslinger.

```
<div class="form-row mb-4" >
  <div class="col-md-6 offset-md-3">
    <article>
      <header>
        <h1>Abonnement</h1>
        <p>Her kan du registrere dig for at abonnere på varslinger</p>
      </header>
    </article>

    <form id="form" role="form">
      <div class="form-group">
        <label for="email">Email</label>
        <input type="email" class="form-control" id="email" name="email" placeholder="Email">
        <span class="text-danger small">* Skal være udfyldt!</span>
      </div>
      <div class="form-group">
        <label for="firstname">Fornavn</label>
        <input type="text" class="form-control" id="firstname" name="firstname" placeholder="Fornavn">
      </div>
      <div class="form-group">
        <label for="lastname">Efternavn</label>
        <input type="text" class="form-control" id="lastname" name="lastname" placeholder="Efternavn">
      </div>
      <div class="form-group"...>
      <div class="form-group"...>
      <div class="form-group"...>
      <div class="form-group"...>
      <div class="form-group"...>
      <div class="form-group">
        <button type="button" id="btnSubmit" class="btn btn-success">Send</button>
      </div>
    </form>
  </div>
</div>

<script src="./subscription/subscription.js"></script>
```

Figur 16 – subscripton.html markup

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

Konstanter for API'et.

MODELS MAPPEN

Der er 2 forskellige model klasser. 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 MySQL.
Klassen subscription er 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 efter den samme skabelon og selve afviklingen af hvad der skal ske ligger i subscription_shared.php.

Nedenstående figur viser 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.

```php
class subscription_shared
{
    private $table_name = "subscriptions";

    public function __construct()
    {
        $this->database = new Database;
    }

    public function read($id)
    {
        $conn = $this->database->getConnection();

        $sql = "SELECT * FROM {$this->table_name}";

        if (!empty($id)) {
            $sql .= " WHERE subscription_id = {$id}";
        }

        $result = mysqli_query($conn, $sql);

        if (mysqli_num_rows($result) > 0) {
            $subscriptions_array = array();

            while ($row = $result->fetch_object()) {
                $subscriptions_item = new Subscription($row);
                array_push($subscriptions_array, $subscriptions_item);
            }

            $result = $subscriptions_array;
        } else {
            $result = new ResponseResult(SUCCESS, READ_NO_ROWS);
        }

        mysqli_close($conn);
        return $result;
    }
```

Figur 17 - subscription_shared.php kode

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

```php
<?php
include_once '../shared/api_constants.php';
error_reporting(ERROR_LEVEL);

header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");

include_once 'subscription_shared.php';
include_once '../models/subscription.php';

if (strcasecmp($_SERVER['REQUEST_METHOD'], 'GET') != 0) {
    throw new Exception('Request method must be GET!');
}

$subscription_shared = new Subscription_shared();
$result              = null;
$subscription_id     = null;

if (isset($_GET['id'])) {
    $subscription_id = $_GET['id'];
}

$result = $subscription_shared->read($subscription_id);

echo json_encode($result);
```

Figur 18 - get.php kode

7.5 Brugervenlighed

Følgende er en række opgaver, som en bruger bliver stillet, for at brugerteste vejrportalen. Jeg har samlet dem fortløbende, men overfor en bruger er de på hver deres side, uden nummerering så de ikke ved hvor mange opgaver der er.

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

Jeg har lavet et website, som kan vise en 5-døgns vejrudsigt. Hvis en browser har lokation aktiveret, hentes vejrdata ud fra denne lokation, ellers kan et bynavn indtastes i et søgefelt og på denne måde kan vejrdata hentes. Vejrkortene som viser vejrinformation for de enkelte dage, er opbygget i et responsivt design hvilket gør det muligt at se på PC eller mobil enhed. Designet på websitet er lavet med henblik på enkelhed, for at opnå en høj brugervenlighed for dem som skal benytte det. Dette gælder også, når en bruger vil registrere sig for at modtage vejrvarslinger på e-mail eller sms. For at fastholde brugervenligheden, er der udarbejdet en række brugervenlighedstest. Jeg har lavet en kommunikationsstrategi, så der ligger en plan for, hvordan man kan få vejrportalen ud til brugerne.

9 Liste over referencer

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-Genevese (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
- 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/