

### 0.0.1 Komponent 3: Administrations hjemmeside

Denne komponent har til formål at håndtere alle de administrative opgaver i systemet. Dette består af 4 delkomponenter:

- Den første delkomponent gør det muligt at tilføje en bus til systemet, fjerne den, eller rediger i en bus der allerede findes i systemet.
- Derefter skal det være muligt at tilføje eller fjerne en bus fra en rute der findes i systemet.
- Den tredje delkomponent gør det muligt at kunne oprette en hel ny busrute i systemet, ændrer i en allerede eksisterende busrute, eller slette en fra systemet.
- Den sidste delkomponent består af muligheden for at kunne tilføje, ændre samt fjerne busstoppesteder fra systemet.

Alle disse delkomponenter udgøre tilsammen en vigtig del af systemet, da uden nogle af dem vil det ikke være muligt at kunne få vist nogle af overstående ting på mobil applikationen.

#### Design:

Hjemmesiden er blevet implementeret ved brug af Microsoft ASP.NET MVC 4 frameworket. Dette gør det nemt og hurtigt at implementere en sofistikeret og moderne hjemmeside, der følger gode design principper. MVC står for Model-View-Controller og følger de samme principper som MVVM angående 'separation of concerns'.

For at kunne indtegne busruter og stoppesteder skal der bruges et kort, til dette er der blevet brugt Google maps samt Google Directions API.

Hjemmesiden består af 4 view, først og fremmest et view til startsiden der linker til de 3 andre views, der består af et der håndterer alt vedrørende busser, et til stoppesteder samt et til busruter.

Det første view der håndterer alt om busserne består af 2 dele. Første del gør det muligt at tilføje en ny bus til systemet, fjerne en bus fra systemet og rediger ID'et for en bus. Dette er blevet implementeret ved at når view'et bliver loaded, bliver en JavaScript function kaldet, der kalder funktionen GetAllBusses() i controlleren, der henter alle busser der er i

MySQL databasen. Til at lave dette kald fra JavaScript til controlleren, bliver der brugt ajax. Ajax gør det muligt at udvæksle data med controlleren og updatere view'et uden at skulle reloade hele websiden.

#### Kodeudsnit 1: Ajax kald til controller funktionen 'GetAllBusses'

```
1      $.ajax({
2          type: "POST",
3          url: "Bus/GetAllBusses",
4          dataType: "json",
5          success: function (result) {
6              var select = document.getElementById("busses");
7              select.options.length = 0;
8              for (var i = 0; i < result.length; i++) {
9                  select.options.add(new Option(result[i]));
10                 ListOfAllBusses.push(result[i]);
11             }
12         }
13     });
```

Dette eksempel på et ajax kald, kalder GetAllBusses(), dette er en funktion der ligger i controlleren, som henter en liste af alle bussernes ID'er fra MySQL databasen. Se *afsnit 9.2.3 Implementering af persistens i online værktøjet* for nærmere beskrivelse af hvordan databasen bliver tilgaaet. Når controlleren er færdig returnerer den et json object, og callback funktionen der er defineret i success parameteren af ajax bliver kaldt. Result parameteren på callback funktionen er returnværdien fra controller funktionen, der i dette tilfælde er et json object, der indeholder en liste af alle bussernes ID'er, hentet fra MySQL databasen. Callback funktionen løber igennem listen af ID'er og tilføjer dem til et HTML select element. Dette gør det muligt for administratoren at se hvilke busser der er gemt i databasen. Administratoren har nu mulighed for at enten tilføje en ny bus, slette en bus, eller ændre ID'et på en bus.

For at tilføje en bus, skriver administratoren bussens ID ind i feltet: 'Bus ID' hvorefter han trykker på knappen 'Add'. Dette vil tilføje bussen til listen, administratoren kan blive ved med at tilføje busser til listen. Administratoren kan også fjerne en bus fra listen, ved at vælge en bus i listen og trykke på knappen 'Remove', der er også mulighed for at ændre navnet for en bus, ved at vælge en bus, og trykker på 'Rename' knappen.

Først ved tryk på 'Save' knappen vil ændringerne blive tilføjet til databasen. Dette sker igen gennem et ajax kald til controller, der kalder `SaveBusChanges()` funktionen. Denne funktion modtager listen af busser, med de nye busser administratoren har tilføjet, samt en liste af alle busserne på databasen. Funktionen sammenligner de 2 lister, finder de busser der er blevet tilføjet, de som er blevet fjernet og dem som har fået nyt ID. Efter alt er fundet, vil den slette de relevante busser fra databasen og tilføje de nye busser.

Anden del af dette view gør det muligt at tilføje busser til en busrute og fjerne busser fra en busrute. Denne del består af 3 lister, hvor den ene indeholder alle busruter, hentet fra databasen, en der indeholder alle busser, der ikke er på nogle busruter samt en der viser hvilke busser der kører på en valgt busrute. I dette views `Onload` funktion bliver der, ud over den overnævnte `GetAllBusses()` funktion, også kaldt 2 andre funktioner, dette forgår igen gennem 2 ajax kald til controlleren, den første henter navnene på alle busruter fra databasen, den anden henter en liste af ID'er for alle de busser der ikke er tilknyttet en rute endnu. Disse 2 ajax kald er magen til ajax kaldet vist i kodeudsnit: 1, den eneste forskel er hvilken controller funktion der bliver kaldt, samt hvilken HTML select element der bliver tilføjet til. Det er nu muligt for administratoren at vælge en af busruterne, fra listen. Dette vil trigger et 'onchange' event, der laver endnu et ajax kald til controller for at hente alle de busser der kører på den valgte rute, og vise dem i listen 'Busses on route'. Der kan nu tilføjes busser fra listen 'Available busses' over til listen 'Busses on route' og ved tryk på knappen 'Save' vil de busser der er blevet flyttet til listen 'Busses on route' blive opdateret i databasen, således at de nu er knyttet til den valgte rute.

Det næste view gør det muligt at oprette en ny busrute, ændrer i en der allerede findes, samt slette en givet busrute fra systemet. For at indtegne en busrute, kræver det et kort, hertil er der blevet brugt Google maps API og Google Directions API. For at få vist kortet på hjemmesiden, kræves det at kortet bliver initialiseret. Først og fremmest skal man have lavet plads til det på siden.

#### Kodeudsnit 2: Div til google maps

```
1 <section id="Map">
2   <div id="map-canvas"></div>
3 </section>
```

Når vores HTML body element er loaded bliver dens OnLoad() event kaldt, dette kalder en JavaScript function, der initializere kortet samt Google directions service. Først bliver der defineret en style, som kortet skal bruge, denne fjerner 'Points of interest'. Dernæst bliver der oprettet et mapOptions object, der definerer forskellige options for kortet, her bliver kortets start position defineret til at vise Aarhus, kort typen bliver sat til ROADMAP, da dette vil vise kortet som et simpelt vej kort. StreetViewControl bliver sat til false, da det er en feature der ikke er relevant for systemet.

#### Kodeudsnit 3: Map opsætning

```
1 var featureOpts = [{
2   featureType: 'poi',
3   stylers: [
4     { visibility: 'off' }]
5   ]];
6 var Aarhus = new google.maps.LatLng(56.155955, 10.205011);
7 var mapOptions = {
8   zoom: 13,
9   mapTypeId: google.maps.MapTypeId.ROADMAP,
10  center: Aarhus,
11  streetViewControl: false,
12  styles: featureOpts
13 };
```

Efter at have defineret mapOptions bliver oprettet et map object. Dette object skal have det overnævnte HTML map-canvas div element, samt mapOptions som constructor parameter.

#### Kodeudsnit 4: Map opsætning

```
1 map = new google.maps.Map(document.getElementById('map-canvas'), ↵
    mapOptions);
```

kortet er nu blevet initialiseret og bliver vist på siden. Det næste der bliver initialiseret er Google direction renderer, dette bliver brugt til at vise en rute på kortet mellem 2 givet punkter. Først bliver der defineret de options som ruten skal bruge. Dette indebære om det skal være muligt at trække i ruten, for at ændre på den vej den skal tage, og om det skal være muligt at klikke på de markers der repræsenterer start og slut punktet for ruten. Dette `renderOptions` object bliver derefter brugt i constructoren for `DirectionsRenderer`, der laver et nyt `DirectionsRenderer` object der senere bliver brugt til at tegne ruten på kortet.

#### Kodeudsnit 5: Map opsætning

```
1 rendererOptions = {
2   map: map,
3   draggable: true,
4   markerOptions: {
5     clickable: true
6   },
7   suppressInfoWindows: true
8 };
9 directionsDisplay = new google.maps.DirectionsRenderer(↵
    rendererOptions);
```

efter kortet og direction renderen er blevet initialiseret, bliver der sat en listener på kortet der lytter efter om der bliver trykket på kortet.

#### Kodeudsnit 6: Map opsætning

```
1 google.maps.event.addListener(map, 'click', function (event) {
2   if (startPoint == null && endPoint == null) //No markers, set ↵
       first
3     startPoint = new google.maps.Marker({
4       map: map,
5       draggable: true,
6       position: event.latLng
7     });
8   else if (startPoint != null && endPoint == null) { //if 1 ↵
       markers, set last markers
9     endPoint = new google.maps.Marker({
10      map: map,
11      draggable: true,
```

```
12     position: event.latLng
13   });
14   calcRoute(startPoint, endPoint);
15   ClearMarkers();
16 }
```