Systemspesifikasjon

Introduksjon

Applikasjonen “planeTracker” er utviklet for å kunne gjøre det mulig å hente ut detaljert informasjon om fly som befinner seg i hele verden i sanntid. Informasjonen som kan hentes omfatter både det flytekniske, som for eksempel flyets modelltype eller motor, men også fart og høyde. Det er lagt stor vekt på enkelhet og tydelighet i brukergrensesnittet, og et kart er benyttet for å øke visualiseringen av området som blir søkt etter.

Formål

Målet med denne applikasjonen er at brukeren kan skaffe detaljert oversikt over alle fly i nærheten av ønsket lokasjon i hele verden. Det betyr at applikasjonen er laget for en spesifikk gruppe med mennesker, enten hobby flyentusiaster eller mer profesjonelle aktører innen luftfart. Hovedmålet med applikasjonen er at man på en enkel og brukervennlig måte kan fremskaffe detaljert informasjon om fly som befinner seg i luften.

Hvordan bruke planeTracker

Applikasjonen er utviklet for web, og kan aksesseres ved å gå inn på denne URL: *localhost:8080/planeTracker* når webserver kjører. Når siden er lastet inn, kan man gjøre følgende:

1. Trykk på “ search location” knappen
2. Skriv i input feltet, for eksempel London og trykk på “search” knappen
3. Den røde sirkelen markerer området som er valgt. Trykk “find planes” knappen
4. Trykk på et fly i tabellen og man får opp detaljert informasjon om det valgte fly. Vinduet kan lukkes for å sjekke ut andre fly i tabellen.

5. Hvis man ønsker å endre området, trykk på “search location” knappen på nytt og gjenta punkt 2 og 3.

Støtte for nettlesere

Applikasjonen er utviklet med bruk av kjente og oppdaterte web teknologier som HTML, CSS og JavaScript, og burde være støttet av de fleste nettlesere. Under utvikling og testing er Chrome benyttet som nettleser, men andre populære nettlesere som Firefox, Microsoft Edge og Safari skal fungere fint. API’et fra Google Maps har støtte for alle de siste versjonene av de nevnte nettlesere ovenfor.

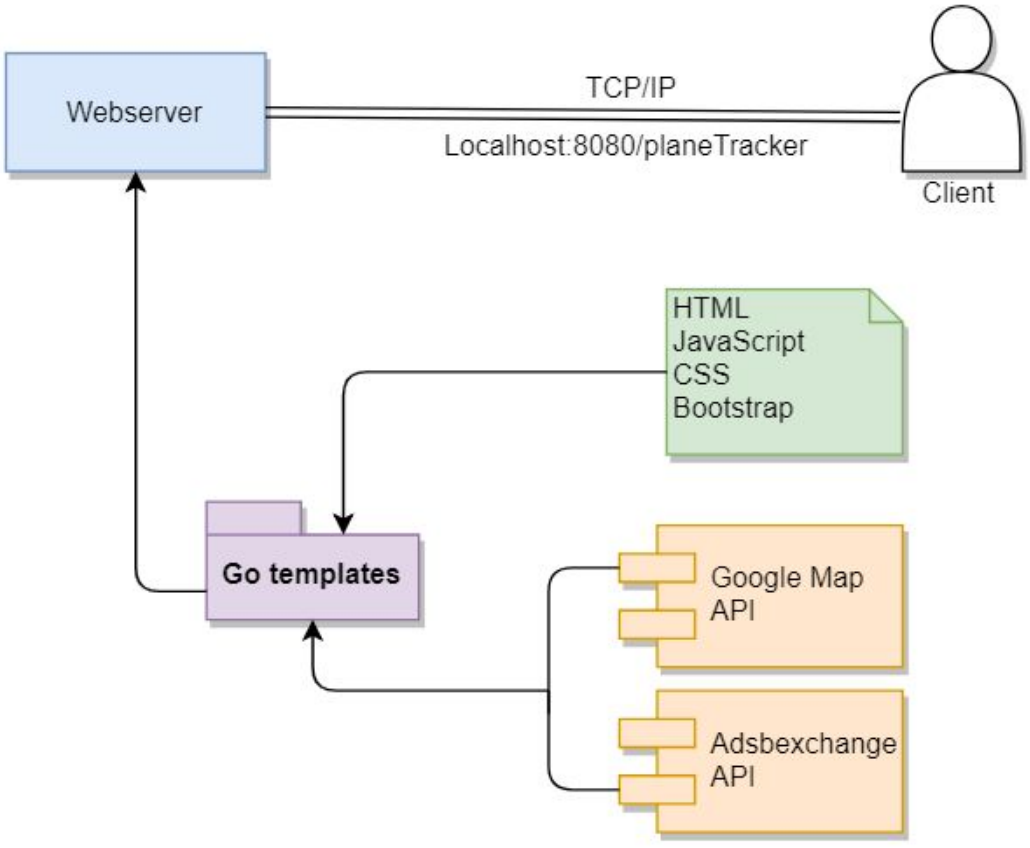
Kjente system begrensninger

Direkte flydata er hentet fra Adsbexchange (ADS-B Exchange, 2018), og er en side som drives på bakgrunn av donasjoner. Det betyr at siden potensielt kan bli fjernet når som helst. Videre, siden det er en gratis tjeneste, vil det noen ganger forekomme noe ustabile API kall.

Applikasjonen har ikke et eget domene, og er avhengig av at en maskin kjører webserveren før man får tilgang til nettsiden gjennom et lokalt oppsett.

Systemarkitektur

Applikasjonen benytter to hovednoder, en webserver og en klient. Klienten aksesserer applikasjonen ved å benytte URL adressen *localhost:8080/planeTracker.* Når en klient kontakter webserveren, opprettes en TCP/IP forbindelse (Margaret Rouse, 2008). Et TCP/IP nettverk benytter små datapakker til å sende informasjon til nettverkslaget, samt sette sammen datapakker og sende dem til applikasjonslaget, websiden.



Figur 1.1 Enkel fremstilling av arkitektur for “planeTracker”

Figur 1.1 viser hvordan arkitekturen i applikasjonen er satt sammen på et overordnet nivå. Ved å benytte Go sin innebygde HTML template pakke, blir index.html rendret til webserveren. Mapper med filer for JavaScript og CSS blir håndtert av Go's HTTP pakke, hvor det benyttes en "fileServer" for å gi tilgang til filene.

Teknologi

På serversiden i applikasjonen benyttes en HTTP protokoll (Wikipedia a, 2018) som gjør det mulig for kommunikasjon mellom klient og server. Protokollen håndterer flere metoder, men de mest vanlige er kalt POST og GET. En GET metode burde alltid hente data, mens en POST metode forespør server om tillatelse til å sende data, for eksempel via et input felt på en webside. Et eksempel fra applikasjonen vises i figur 1.2, hvor det benyttes en POST metode for å sende koordinatene til et område, som blir hentet fra input feltet "searchInput", videre til webserveren via et input felt som er usynlig for bruker.

```
<input type="text" name="search" id="searchInput" placeholder="e.g. Manchester">
<button class="btn btn-dark" id="searchBtn" onclick="getAddress()">Search</button>
<button class="btn btn-success" id="subBtn" onclick="submit()">Go to planes</button>
<form action="/planeTracker" method="post" id="coordinates-form">
  <input type="hidden" name="coordinates" id="coordinates">
</form>
```

Figur 1.2 POST metode eksempel

I applikasjonen er det brukt flere teknologier for å lage innholdet på websiden. Det benyttes kun en HTML side, index.html, for å lage skallet til applikasjonen. Videre brukes Bootstrap (Bootstrap, 2018) for å lage et responsivt og mobilvennlig brukergrensesnitt. Bootstrap's grid system er benyttet for å holde kontroll over elementene på siden, mens "modal" brukes for pop-up vinduene.

JavaScript er brukt for å lage ulike funksjoner i applikasjonen, som for eksempel henting av detaljert flydata fra datasettet. I denne funksjonen er også jQuery (jQuery, 2018), et raskt og effektivt JavaScript bibliotek, tatt i bruk for å legge til data i flydata tabellen. JQuery gjør DOM (Document Object Model) manipulering enkelt og raskt.
Det er benyttet to ulike API'er i denne applikasjonen. Et API for å hente ned fly i sanntid kalt ADS-B Exchange som leverer all data i JSON format, og et Google Maps API for å vise frem kartet med området som er valgt. Et eksempel på kall for fly med 100 km omkrets rundt Bergen vil se slik ut:

http://public-api.adsbexchange.com/VirtualRadar/AircraftList.json?lat=60.397076&lng=5.324383&fDstL=0&fDstU=100

Beskrivelse av systemets funksjoner

I dette kapitlet vil systemets funksjoner bli presentert i form av en mer detaljert beskrivelse.

1. Når applikasjonen starter kjøres en http.ListenAndServe() metode som lytter etter koblinger mot port 8080. Når en klient gjør en forespørsel til stien "/planeTracker", blir det gjort et kall på handleFunc("/planeTracker, response).

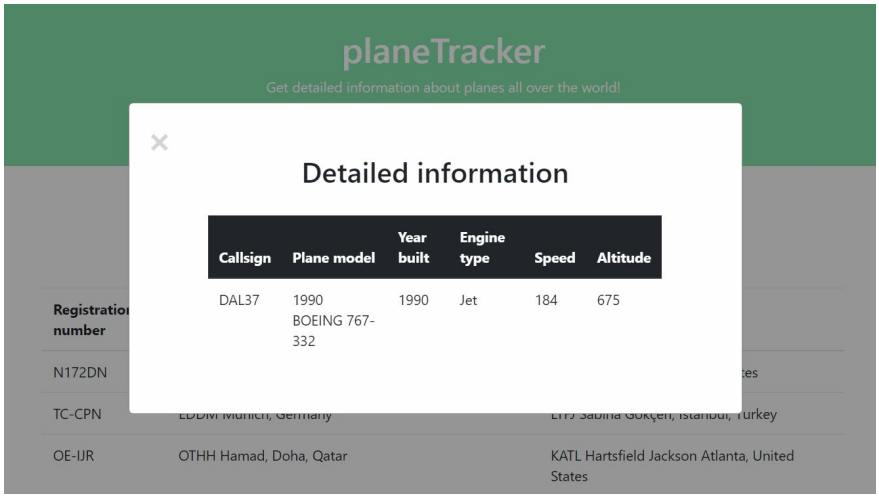
2. Når response() blir kalt sørger den for å parse HTML-filen "index.html" og vise frem nettsiden til bruker. Videre vil respone() lytte etter input fra bruker. Når en bruker skriver inn et sted i søkefeltet under "search location" vil denne informasjonen bli sendt via en POST metode (se figur 1.2) og lagret i en variabel kalt "coordinates" i response metoden. Informasjoen blir fetchet ved hjelp av en http.Request metode kalt parseForm(). Videre hentes de aktuelle verdiene fra dataen ned ved hjelp av FormValue().

3. Data lagret i variabelen “coordinates” benyttes videre til å lage API URL’en for ADS-B Exchange. Dette gjøres via en metode kalt getAPIUrl(), som benytter flere hjelpemetoder for å sette koordinatene på riktig plass i URL’en. Kall på API’et styrer data som representeres i hovedtabellen på landingssiden, og blir alltid oppdatert etter at bruker har trykket “find planes” knappen. Disse dataene blir hentet direkte fra index.html. Figur 1.3 viser hvordan data fra structet “planes” blir loopet gjennom og aktuell data blir hentet ut og lagt til i tabellen.

```
{{range $.AcList}}
  <tr onclick = "getPlaneInfo({{.Call}}, {{.Mdl}}, {{.Year}}, {{.EngType}}, {{.Spd}}, {{.Alt}})">
    <td>{{.Reg}}</td>
    <td>{{.From}}</td>
    <td>{{.To}}</td>
  </tr>
{{end}}
```

Figur 1.3 Looper gjennom struct data

4. I figur 1.3 ser man også hvordan detaljert informasjon blir hentet ut. På hver rad i tabellen er det lagt inn en “click event” som kaller på en metode kalt “getPlaneInfo()” med aktuelle flydata som parametere. Denne metoden håndterer de ulike dataene som blir sendt med, lagrer den i et JavaScript objekt, og benytter jQuery til å lage en ny tabell med informasjonen (se figur 1.4).



Figur 1.4 Snippet fra detaljert informasjon fra et fly

Testing

Det er skrevet unit tester (Wikipedia b, 2018) for metoder knyttet til webserver og response(). En unit test er ment til å teste funksjonalitet til en del av kildekoden, typisk en modul eller metode. Testene er implementert ved hjelp av Go's egen test-pakke kalt "testing". Alle testene er det man kaller positive tester, det vil si at man forventer at testen skal gå gjennom fordi input i testen skal være korrekt.

Den første testen sjekker at response handler fungerer, og at den returnerer en HTTP statuskode 200 hvis input er korrekt i forhold til ønsket resultat. Resten av testene sjekker de ulike metodene som er knyttet til opprettelse av API URL'en.

Det burde vært skrevet flere tester til andre funksjoner i programmet, spesielt knyttet til JavaScript funksjonene. I tillegg burde det vært skrevet negative unit tester til funksjonene på server siden, altså tester som skal returnere feil. Google Maps API har innebygde håndteringer av feil, og har derfor ikke vært fokusert på i forhold til testing i denne oppgaven.

References

Margaret Rouse (2008) TCP/IP. Retrieved from
https://searchnetworking.techtarget.com/definition/TCP-IP

Wikipedia a (2018) Hypertext Transfer Protocol. Retrieved from
https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

Bootstrap (2018) Introduction. Retrieved from
https://getbootstrap.com/

jQuery (2018) What is jQuery. Retrieved from
https://jquery.com/

ADS-B Exchange (2018) Accessing Data Collected by ADS-B Exchange. Retrieved from
https://www.adsbexchange.com/data/

Wikipedia b (2018) Unit testing. Retrieved from
https://en.wikipedia.org/wiki/Unit_testing