

**OPPTUR – EN WEBAPPLIKASJON**  
PROSJEKT I TBA4250, 210416

Marie S. Sagedal, Nina T. Hermanrud & Kristina Jakobsen



# INNHOLDSLISTE

<b>1. INNLEDNING .....</b>	<b>3</b>
<b>1.1 Produktbeskrivelse.....</b>	<b>3</b>
1.1.1 Bakgrunn.....	3
1.1.2 Formål.....	3
1.1.2 Målgruppe .....	4
<b>1.2 Kravspesifikasjon .....</b>	<b>4</b>
1.2.1 Krav med høyes prioritet.....	4
1.2.2 Krav med lavest prioritet .....	4
<b>1.3 Arbeidsplan .....</b>	<b>5</b>
1.3.1 Oversikt.....	5
1.3.2 Utvikle web-applikasjon .....	5
1.3.3 Hente inn, tilpasse og vurdere data .....	6
1.3.4 Forberede presentasjon.....	6
1.3.5 Skrive rapport.....	6
<b>1.4 Data brukt i prosjektet.....</b>	<b>6</b>
<b>2.1 Git, Node.js, Heroku og MongoDB .....</b>	<b>7</b>
<b>2.2 HTML5, CSS, JavaScript og AngularJS .....</b>	<b>8</b>
<b>2.3 Biblioteker .....</b>	<b>9</b>
<b>3. PROSESS .....</b>	<b>9</b>
<b>3.1 Idémyldring og konseptutvikling .....</b>	<b>9</b>
<b>3.2 Utvikling .....</b>	<b>10</b>
3.2.1 Implementering av søk-funksjon .....	10
3.4.2 Implementering av API for Google Places Autocomplete .....	12
3.4.3 Leaflet .....	12
<b>4. DISKUSJON.....</b>	<b>12</b>
<b>4.1 Utfordringer .....</b>	<b>12</b>
4.1.1 Forkunnskaper .....	12
4.1.3 Server.....	13
4.1.4 GitHub .....	13
4.1.5 CORPS .....	13
4.1.6 Angular vs uten rammeverk .....	14
<b>5. RESULTAT .....</b>	<b>14</b>
<b>5.1 En oversikt over nettsidens ferdige oppbygging .....</b>	<b>14</b>
<b>5.2 Brukerveiledning .....</b>	<b>14</b>
<b>5.3 Veien videre .....</b>	<b>15</b>

# 1. INNLEDNING

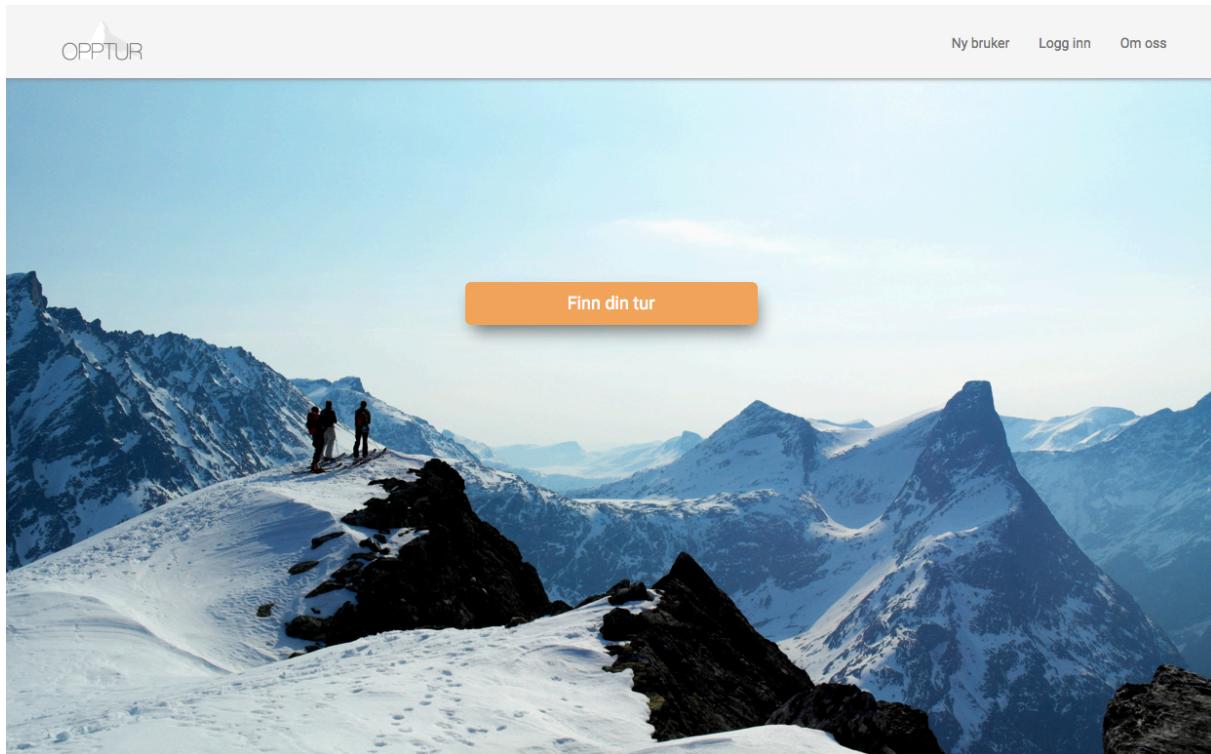


Figure 1 Skjermdump av startsiden til Opptur

## 1.1 Produktbeskrivelse

### 1.1.1 Bakgrunn

Opptur er en webapplikasjon som gir brukeren en oversikt over gode toppturmuligheter. Brukeren skal kunne søke på toppturer i hele Norge, og øyeblinkelig få informasjon om hvor turen går, turens varighet, vanskelighetsgrad, skredfaresoner, osv.. Dersom brukeren er logget inn, vil brukeren også få muligheten til å legge til sitt eget toppturforslag.

### 1.1.2 Formål

Formålet med webapplikasjonen er å gjøre toppturer lett tilgjengelige. Planlegging og gjennomføring av en topptur skal bli enklere og tryggere. På den måten kan toppturer bli tilgjengelig for et bredere publikum

### **1.1.2 Målgruppe**

Webapplikasjonens målgruppe er alle som kunne tenke seg å gå på toppturer, uavhengig om de har ski eller brett på bena. Fordi hver tur inneholder informasjon om aktsomhetsområder for skred, skredvarsling for området og vanskelighetsgrad, retter applikasjonen seg til hele spekteret fra nybegynnere til ekspert.

## **1.2 Kravspesifikasjon**

### **1.2.1 Krav med høyes prioritet**

Brukeren skal kunne ...

- Registrere ny bruker
- Logge av og på
- Søke etter turer
  - *Avstand til startpunkt fra adresse*
  - *Turens varighet*
  - *Vanskelighetsgrad*
- Få informasjon om tur
  - *Områder med skredfare*
  - *Estimert turvarighet*
  - *Vanskelighetsgrad*
  - *Rute*
  - *Beskrivelse*
  - *Aktsomhetsområder for skred*
- Registrere egen tur (må være logget inn, og admin må godkjenne turen før den er søkbar)

### **1.2.2 Krav med lavest prioritet**

Brukeren skal kunne ...

- Lese om webapplikasjonen og få brukerveiledning for denne

- Lese om hvem som har utviklet web-applikasjonen og få kontaktinformasjonen deres
- Få informasjon om tur
  - *Skredvarsling for området*
  - *Høydeprofil*
- Søke etter turer
  - *Toppturens navn*
  - *Avstand til startpunkt fra gjeldende posisjon*
- Endre brukerinformasjon (må være logget inn)
  - *Passord*
  - *Brukernavn*
  - *Legge til utstyr (randoné/telemark/alpint/snowboard/...)*
  - *Legge til profilbilde*
- Gi vurdering til andre turer (må være logget inn)
  - Kommentarer
  - Stjerne-vurdering
- Legge til turer til ”Mine turer” (må være logget inn)
  - *”Ønskelisten”: ønskeliste av turer brukeren ønsker å gå*
  - *”Mine turer”: oversikt over turer brukeren har gått*

## 1.3 Arbeidsplan

### 1.3.1 Oversikt

- Lære verktøy som skal brukes i utviklingen
  - GitHub, Git
  - Bootstrap, HTML, CSS
  - Heroku, node.js, MongoDB
  - Javascript, Angular

### 1.3.2 Utvikle web-applikasjon

- Sette opp databasen
- Utvikle brukergrensesnitt

- Controller og back-end

### **1.3.3 Hente inn, tilpasse og vurdere data**

- Hente inn data om skredfaresoner

### **1.3.4 Forberede presentasjon**

- Forberede demonstrasjon av brukergrensesnitt

### **1.3.5 Skrive rapport**

## **1.4 Data brukt i prosjektet**

Data vi har lastet ned i forbindelse med prosjektet, er skredfaresoner fra Norges vassdrags- og energidirektorat. Disse dataene har vi brukt til å visualisere skredfarene i de aktuelle toppturområdene.

## 2. METODE

### 2.1 Git, Node.js, Heroku og MongoDB

Vi har valgt å lage Opptur som en node.js- applikasjon. Applikasjonen inneholder samling av kode, rammeverk og beskrivelse av avhengigheter. For å kjøre nodeapplikasjonen vår i ‘nettskyen’ har vi brukt en server fra Heroku. På den måten har vi kunnet implementere appen ved bruk avhengighetscaching.<sup>1</sup>

For å kunne samarbeide på koden opprettet vi et Git-repository, som er et versjonskontrollsyste<sup>m</sup><sup>2</sup> som gjør det mulig for flere utviklere å sammenflette kodeenringer i et delt repository/mappestruktur. Git er også brukt til å oppdatere applikasjonen til server. Vi skriver `git push heroku master` i terminalen, og det som er lagret i vårt Gitrepository, blir da også lagret på server, og kan kjøres derfra istedenfor lokalt på datamaskinen.

Ved bruk av Heroku har vi lagt til MongoDB-database som en add-on for applikasjonen vår. MongoDB er en database for webapplikasjoner som kan håndtere romlige data, noe som er viktig for en kartapplikasjon<sup>3</sup>. Det er også en noSQL-database<sup>4</sup>, noe som er blitt mer og mer populært for programvareutviklere ettersom man kan behandle big data som må aksesseres raskt av mange brukere på en bedre måte enn ‘vanlige’ relasjonsdatabaser. For vårt formål har det vært bra med en ikke-relasjonsdatabase fordi vi underveis i utviklingen av programmet vårt har endret på objekt-modellene vi har sendt inn til databasen, og dette blir ikke et problem for spørringene våre til databsen. Objektene som lagres er JSON-objekter. De to modellene vi hadde er vist i figur 2 og 3, og hvordan de aksesseres og lagres er nøyere forklart i del 3. Prosess. Modellene i MongoDB kalles collections, og hver JSON som blir lagret er et dokument i en collection. For eksempel er brukeren ‘Kalle’ registrert som et JSON i collectionen ‘user’.

---

<sup>1</sup> <https://devcenter.heroku.com/categories/nodejs> [20.04.16]

<sup>2</sup> <https://no.wikipedia.org/wiki/Git> [21.04.16]

<sup>3</sup> <https://mongodb.github.io/node-mongodb-native/api-articles/nodekoarticle1.html> [19.04.16]

<sup>4</sup> <https://www.mongodb.com/nosql-explained> [19.04.16]

## 2.2 HTML5, CSS, JavaScript og AngularJS

Fordi vi har valgt at Opptur skal være en webapplikasjon, var HTML5, CSS og JavaScript naturlige valg av verktøy. HTML er et markeringsspråk for formattering av nettsider, CSS er språket som brukes til å style HTML-filene og JavaScript er et skriptspråk for å tilføye dynamiske elementer til nettsidene<sup>5</sup>. Fordi applikasjoner skrevet i HTML og JavaScript er plattformuavhengige, og fordi alt man trenger å ha installert for å bruke applikasjonen er en nettleser, så vi HTML og JavaScript som åpenbare verktøy.

AngularJS er et JavaScript-rammeverk for å lage applikasjoner med kun én side, og alt innholdet lastes til denne siden. Det vil si at de forskjellige views'ene (sidene) routes (aksessereres) fra index-filen som virker som 'basisen', eller startsiden, for applikasjonen vår. Dette er illustrert i figur 4 og 5.

```
var tripSchema = new
Schema({
  tripName: { type: String,
required: true},
  latitude: {type: String},
  longitude: {type: String},
  place: {type: String, required:
true},
  difficulty : Number,
  description: String,
  duration : Number,
  center : Object,
  zoom : Number,
  path : Object,
  godkjent: Boolean
```

```
var userSchema = new Schema({
  email: { type: String, required:
true},
  password: { type: String,
required: true },
  nickname: String
});
```

Figure 3 Trip-schema, slik det sendes inn i databasen.  
'center' og 'path' er JSON-objekter som er variabler for kartet som gir informasjon om kartet til hver tur.

Figure 2 User-schema med brukerinformasjonen som lagres i databasen

<sup>5</sup> <https://no.wikipedia.org/wiki/JavaScript>

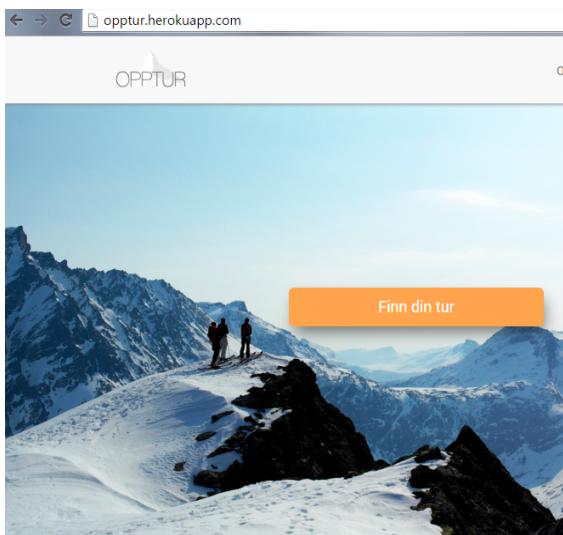


Figure 4 headeren vår, med fast navigasjonsbar i topp ("Opptur", "Om", "Ny bruker", "Logg inn")

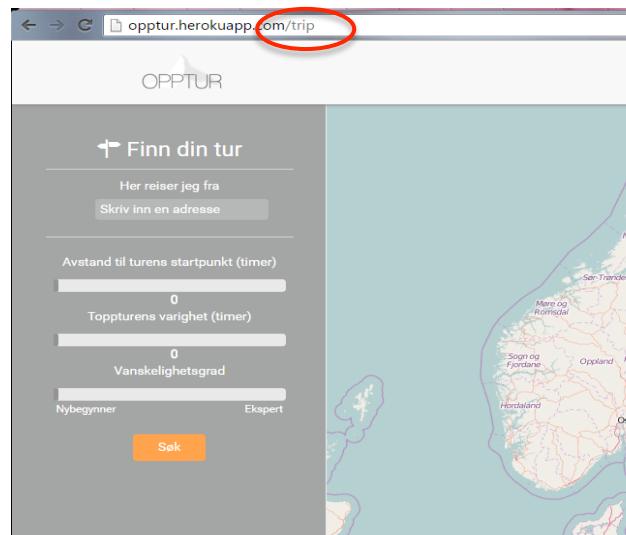


Figure 5 /trip -viewet, fortsatt fast navigasjonsbar fra headeren

## 2.3 Biblioteker

- **jQuery**  
JavaScript-bibliotek for å lette klientskriftning av HTML
- **Leaflet**  
JavaScript-bibliotek for interaktive kart
- **Bootstrap**  
Et front-end-rammeverk for websider og webapplikasjoner
- **Entypo**  
En samling av pictogrammer

# 3. PROSESS

## 3.1 Idémyldring og konseptutvikling

Første del av prosessen omhandlet idémyldring og konseptutvikling. Vi ble enige om hvilken tjeneste kartapplikasjonen vår skulle tilby, og om vi skulle lage en applikasjon for nettbrett/mobil eller for web. Neste steg var å finne ut hvilke verktøy vi ønsket å benytte oss av for å utvikle kartapplikasjonen. Verktøyene vi kom frem til er gjort rede for i metode-avsnittet og diskutert i diskusjons-avsnittet.

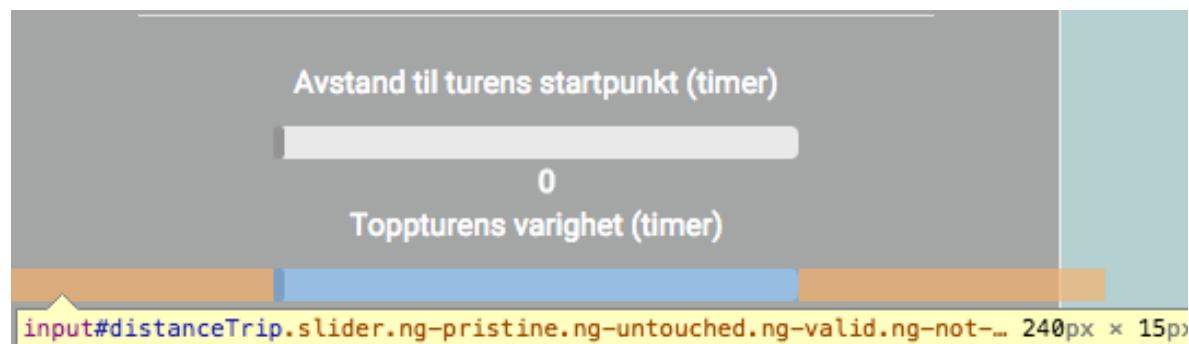
## 3.2 Utvikling

Vi har i dette avsnittet gjort greie for et utvalg av implementeringene vi har gjort.

### 3.2.1 Implementering av søk-funksjon

Søket ble implementert ved å først hente ut et json objekt fra databasen med alle turer som har en vanskelighetsgrad mindre eller lik verdiene skrevet av brukeren i infofeltene #distanceTrip og #difficult. Disse verdiene blir hentet opp i front-end, og sendt til backend. Der blir verdiene hentet ved req.body.difficulty og req.body.duration. Queryen for å hente opp disse turene er vist under, hvor verdien \$lte: tilsvarer mindre eller lik.

```
var query = {
  'difficulty': { $lte: req.body.difficulty },
  'duration': { $lte: req.body.duration },
  'godkjent': true
};
```



Videre implementerte vi Google Maps sitt "distance and duration"-API<sup>6</sup>. API'et krever lengde- og breddegrad til startposisjonen til brukeren, samt en liste over destinasjoner med lengde- og breddegrader, og en unik API-nøkkel for prosjektet. Startposisjon ble hentet ut fra input fra bruker i #txtautocomplete text-inputen. Figur 6 viser hvordan API-stringen er bygget opp.

<sup>6</sup> <https://developers.google.com/maps/documentation/distance-matrix/intro>

## Figur 6

### API for å finne korteste vei mellom origin og turer:

```
String API = "https://maps.googleapis.com/maps/api/distancematrix/json?units=imperial&origins=" +  
origin.latitude + "," + origin.longitude + "&destinations=";  
For(tur:muligeTurer){  
    API += tur.latitude + "%2C" + tur.longitude + "%7C";  
}  
API += "&key=AlzaSyCVws1GADOVAu0rPTddKuC0gHka0F32_VA" <- Vår nøkkel
```

API'et leverer et json objekt på formatet vist i figur 7

```
{  
    "destination_addresses" : [  
        "Vikabakken 10-12, 8310 Kabelvåg, Norway",  
        "Skolegata 4, 8300 Svolvær, Norway"  
    ],  
    "origin_addresses" : [ "Kirkeveien 11, 8312 Henningsvær, Norway" ],  
    "rows" : [  
        {  
            "elements" : [  
                {  
                    "distance" : {  
                        "text" : "12.0 mi",  
                        "value" : 19323  
                    },  
                    "duration" : {  
                        "text" : "24 mins",  
                        "value" : 1417  
                    },  
                    "status" : "OK"  
                },  
                {  
                    "distance" : {  
                        "text" : "15.0 mi",  
                        "value" : 24184  
                    },  
                    "duration" : {  
                        "text" : "29 mins",  
                        "value" : 1720  
                    },  
                    "status" : "OK"  
                }  
            ]  
        }  
    ],  
    "status" : "OK"  
}
```

Figur 7 json - objekt fra Google maps sitt Distance and Duration API

En for-løkke ble tilslutt brukt til å hente ut et json-objekt med alle turene med avstand (distance.value) <= verdien i #inputDistanceToStart text-inputen. Verdien blir gitt i sekunder.

### **3.4.2 Implementering av API for Google Places Autocomplete**

I både trip og makeTrip ønsket vi at brukeren skulle kunne fylle inn en adresse, hvor det automatisk skulle komme opp alternative adresser samtidig som brukeren skrev. Dette løste vi ved å implementere Google Places Autocomplete APlet til Google development.<sup>7</sup> APlet er implementert med hjelp av et Angularjs directive<sup>8</sup> app/public/Shared/googleplace/googleplace

### **3.4.3 Leaflet**

For å vise kart og kartdata, har vi benyttet oss av Leaflet-biblioteket. Vi bruker funksjoner Leaflet tilbyr til å tegne GeoJSON-objekter, samt bildefiler fra wms-server. Datalaget om aktsomhetsområder hentet vi eksempelvis fra WMS-serveren til NVE, og vi la det til i kartet som et tileLayer.wms ved hjelp av Leaflet-biblioteket.

## **4. DISKUSJON**

### **4.1 Utfordringer**

#### **4.1.1 Forkunnskaper**

Vi har støtt på et par komplikasjoner underveis. De fleste av disse grunner i utfordringen med manglende forkunnskaper om webapplikasjonsprogrammering. En av tre har noe erfaring i HTML og CSS fra videregående, er ellers grunnlaget vårt det vi har fått fra tidligere semestre på studiet. Det har tatt tid å sette seg inn i alt, men vi har fordelt oppgavene slik at alle ikke må bruke like lang tid på å forstå absolutt alt av alt. På denne måten jobbet vi mer effektivt på applikasjonen.

---

<sup>7</sup> [https://developers.google.com/places/web-service/autocomplete#place\\_autocomplete\\_requests](https://developers.google.com/places/web-service/autocomplete#place_autocomplete_requests)

<sup>8</sup> <https://gist.github.com/VictorBjelholm/6687484>

#### **4.1.2 Innlogging**

En viktig implementasjon i programmet vårt er at du som logget inn får tilgang til flere funksjonaliteter enn om du ikke er innlogget. Å opprette bruker som lagres til databasen og hente ut denne informasjonen til konsollen har vært vanskelig, men det har gått greit etter noen tutorials på nettet og hjelp av studass, men det å gi brukeren en ‘state’ som pålogget var vanskeligere. Dette løste vi ved å bruke rammeverket AngularJS som er beskrevet i xxxx

#### **4.1.3 Server**

Opptur kjøres som nevnt på skyserveren Heroku. På grunn av en feilmelding etter søk på turer, krasjet hele appen med feilkoden ‘H10 – app crashed’ noe som skjer når web dynoen krasjer eller fordi det er en oppstartsstans på den<sup>9</sup>. Vi fant en quick-fix for dette, og det var å skrive `heroku restart`, for da kjører appen på nytt. Dette er derimot ikke en løsning for det lange løp, så vi fant ut at noen turer som blir hentet fra databasen har udefinerte elementer i kartlaget sitt som gjorde at beregningen av korteste veg i «Finn din tur» gav feil, og stanset programmet. Dette ble løst ved å fjerne turene med disse udefinerte elementene fra resultat lista over mulige turer.

#### **4.1.4 GitHub**

Vi har opplevd dette verktøyet som både hjelpsomt og frustrerende. Det har vist seg å være en utfordring å klare å merge/sammenflette endrete filer uten å overse små detaljer i koden som ikke skulle blitt endret. I tillegg har det vært komplikasjoner på Git'en som oppdaterer til server, slik vi har hatt vanskeligheter med å oppdatere til `heroku master`. Løsningen på dette problemet har vært å klone mappen på nytt, og be til høyere makter om at det ikke skal skje igjen.

#### **4.1.5 CORPS**

Da vi ønsket å hente inn API'ene for kjøretid og skredvarsel fra front-end, møtte vi på et uventet problem: CORPS<sup>10</sup>. ”Cross-origin resource sharing” hindrer nettsider fra å

---

<sup>9</sup> <https://devcenter.heroku.com/articles/error-codes#h10-app-crashed> [21.04.16]

<sup>10</sup> [https://en.wikipedia.org/wiki/Cross-origin\\_resource\\_sharing](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing)

lage AJAX-spørninger til et annet domene. Dette er for å opprettholde lesersikkerheten på nettsiden. Etter å ha prøvd å komme rundt dette problemet i front-end, endte vi tilslutt opp med å bruke serveren til å få kontakt med APlet, for deretter å sende dataen man vil ha ut av APlet til front-end.

#### 4.1.6 Angular vs uten rammeverk

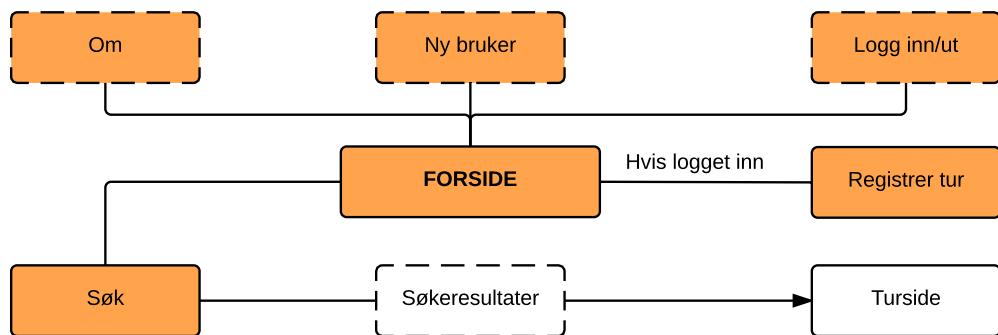
Da vi som tidligere nevnt hadde lite erfaring med programmering av webapplikasjoner ved prosjektstart, var vår opprinnelige plan å ikke bruke rammeverk på prosjektet. Da vi midtveis i prosjektet valgte å implementere Angular likevel for å få til innlogging, møtte vi på en rekke problemer.

## 5. RESULTAT

### 5.1 En oversikt over nettsidens ferdige oppbygging

Side     Popup     Side som til en hver tid kan aksesseres

#### OPPTUR



### 5.2 Brukerveiledning

Vedlagt finnes brukerveiledning for å:

- Registrere bruker
- Logge inn
- Registrere tur
- Søke på tur

### 5.3 Veien videre

Under idémyldringen og underveis i utviklingen har vi kommet på en rekke funksjoner vi kunne supplere med for å få et enda bedre produkt som er lettere å bruke og som har flere bruksområder. Veien videre ville vært å implementere følgende funksjoner:

Brukeren skal kunne ...

- Få informasjon om tur
  - *Høydeprofil*
  - *Opptur og nedrenn*
- Søke etter turer
  - *Toppturens navn*
  - *Avstand til turens startpunkt fra gjeldende posisjon*
- Få alle registrerte turer opp på kartet på søker-siden (før søkeret er gjort)
- Endre og legge til brukerinformasjon (må være logget inn)
- Gi vurdering til andre turer (må være logget inn)
  - Kommentarer
  - Stjerne-vurdering
- Legge til turer til ”Mine turer” (må være logget inn)
  - *”Ønskelisten”*: ønskeliste av turer brukeren ønsker å gå
  - *”Mine turer”*: oversikt over turer brukeren har gått
    - *Fylle ut egen informasjon om eksempelvis turvarighet*

- Logge geografisk posisjon (lat, long) når de går en topptur for så å kunne registrere den når toppturen er ferdig
- Bruke opptur-appen når de er på topptur for å finne riktig vei