

# Prosjektrapport – Team 12

## Badenymfe

Institutt for informatikk, UiO

**Sydhagen, Kristin Aurora**

Informatikk: Programmering & systemakritik

[kristasy@student.matnat.uio.no](mailto:kristasy@student.matnat.uio.no)

**Skøien, Johannes**

Informatikk: Design, bruk & interaksjon

[johasko@student.matnat.uio.no](mailto:johasko@student.matnat.uio.no)

**Rafdal, Maren Zon**

Informatikk: Programmering & systemakritik

[marenzr@student.matnat.uio.no](mailto:marenzr@student.matnat.uio.no)

**Knudsen, Vilde Paschen**

Informatikk: Programmering & systemakritik

[vildepk@student.matnat.uio.no](mailto:vildepk@student.matnat.uio.no)

**Andersen, Ellen Marie**

Informatikk: Programmering & systemakritik

[ellemand@student.matnat.uio.no](mailto:ellemand@student.matnat.uio.no)

**Wik, Lars Erik**

Informatikk: Programmering & systemakritik

[larsewi@student.matnat.uio.no](mailto:larsewi@student.matnat.uio.no)

**Olstad, Jostein**

Veileder

[josteiol@student.matnat.uio.no](mailto:josteiol@student.matnat.uio.no)

29-05-2020

# **Innhold**

<b>1 Presentasjon</b>	<b>1</b>
1.1 Sammendrag . . . . .	1
1.2 Om teamet . . . . .	1
1.3 Om caset . . . . .	4
<b>2 Prosessdokumentasjon</b>	<b>5</b>
2.1 Bestemme case . . . . .	5
2.2 Strukturen i gruppearbeidet . . . . .	5
2.3 Valg av digitale kanaler og bruk av disse . . . . .	6
2.4 Gjennomføring av sprints . . . . .	9
<b>3 Brukerdokumentasjon</b>	<b>10</b>
3.1 Funksjonalitet og struktur . . . . .	11
3.2 Målgruppe . . . . .	13
3.3 Scenarios . . . . .	15
3.3.1 Scenario en . . . . .	15
3.3.2 Scenario to . . . . .	16
3.4 Applikasjonens tilgjengelighet . . . . .	16
<b>4 Brukerundersøkelser</b>	<b>16</b>
4.1 Første brukerundersøkelse . . . . .	16
4.1.1 Metode . . . . .	17
4.1.2 Presentasjon av undersøkelse med funn . . . . .	17
4.1.3 Bruk av funn for videre utvikling av Badenymfe . . . . .	18
4.2 Andre brukerundersøkelse . . . . .	19
4.2.1 Metode . . . . .	19
4.2.2 Presentasjon av undersøkelse med funn . . . . .	19
4.2.3 Bruk av funn og videre utforming av Badenymfe . . . . .	20
4.3 Oppsummering av designvalg og forkastede idéer . . . . .	21
<b>5 Kravspesifikasjon og modellering</b>	<b>23</b>
5.1 Opprinnelige krav . . . . .	23
5.1.1 Opprinnelige funksjonelle krav . . . . .	23
5.1.2 Opprinnelige ikke-funksjonelle krav . . . . .	23
5.2 Endelige funksjonelle krav og modellering . . . . .	24
5.2.1 Endelige funksjonelle krav . . . . .	24

5.2.2	Use Case - Modellering . . . . .	25
5.2.3	Use Case - tekstlig beskrivelse . . . . .	26
5.2.4	Klassediagram . . . . .	30
5.2.5	Sekvensdiagram . . . . .	32
5.3	Endringer og implementering av krav . . . . .	35
5.4	Universell utforming . . . . .	36
<b>6</b>	<b>Produktdokumentasjon</b>	<b>37</b>
6.1	Arkitektur . . . . .	37
6.1.1	Høy kohesjon & lav kobling . . . . .	38
6.1.2	Databinding . . . . .	38
6.1.3	Model-View-ViewModel . . . . .	39
6.2	Periodisk arbeider . . . . .	40
6.3	Live data . . . . .	41
6.4	Teknologier . . . . .	42
6.4.1	Android Plattformen . . . . .	42
6.4.2	Android Studio . . . . .	42
6.4.3	Kotlin . . . . .	42
6.4.4	API meteorologisk institutt . . . . .	43
6.4.5	Room . . . . .	43
6.4.6	Retrofit . . . . .	44
6.4.7	Google Maps . . . . .	44
<b>7</b>	<b>Testdokumentasjon</b>	<b>45</b>
7.1	Trusselpoker . . . . .	45
7.2	Testing av ikke-funksjonelle krav . . . . .	47
7.3	Instrumentelle tester . . . . .	48
7.4	Test av database . . . . .	50
7.5	Enhetstester . . . . .	51
<b>8</b>	<b>Refleksjon</b>	<b>51</b>
8.1	Samarbeid, kommunikasjon og Korona . . . . .	51
8.2	Digital arbeidshverdag, motivasjon og .... Korona . . . . .	53
8.3	Store ambisjoner, begrenset tid og enda mer begrenset kompetanse	54
<b>Referanser</b>		<b>56</b>
<b>A Teamavtale</b>		<b>59</b>

A.1	Tilstedeværelse . . . . .	59
A.2	Tidsbruk . . . . .	59
A.3	Forventninger . . . . .	59
A.4	Konflikthåndtering . . . . .	59
<b>B</b>	<b>Samtykkeerklæring</b>	<b>60</b>
<b>C</b>	<b>Innledende brukerundersøkelse</b>	<b>62</b>
C.1	Bruk . . . . .	62
C.2	Introduksjon . . . . .	62
C.3	Hoveddel . . . . .	63
C.4	Avslutning . . . . .	64
<b>D</b>	<b>Brukertest</b>	<b>65</b>
D.1	Mål for brukertesten . . . . .	65
D.2	Deltaker . . . . .	65
D.3	Tilnærming/metode . . . . .	65
D.4	Oppgaver . . . . .	65
D.5	Praktiske forhold . . . . .	66

## Figurer

1	Motivasjon til gruppen på Slack . . . . .	6
2	Standup på Slack . . . . .	7
3	Eksempel på hvordan backlogen relatert til appen så ut . . . . .	8
4	Eksempel på hvordan backlogen relatert til use case ID . . . . .	8
5	Eksempel på hvordan backlogen relatert til praktiske oppgaver så ut . . . . .	8
6	Sprint i uke 1 . . . . .	9
7	Mine favoritter . . . . .	11
8	Kart, slettefunksjon og om appen . . . . .	12
9	Værmelding for en lokasjon . . . . .	12
10	Use-case diagram . . . . .	26
11	Klassediagram . . . . .	31
12	Sekvensdiagram: bruker leser om appen . . . . .	32
13	Sekvensdiagram: bruker legger til ny posisjon . . . . .	33
14	Sekvensdiagram: bruker sjekker badeforhold på lagret lokasjon . . . . .	34
15	MVVM . . . . .	41

16	Trusselpoker på Zoom . . . . .	46
17	GitHub i starten av prosjektet . . . . .	52
18	GitHub mot slutten av prosjektet . . . . .	52

## Tabeller

1	Avstemning rundt scenarioene . . . . .	47
2	Avstemning rundt sannsynlighet . . . . .	47
3	Testplan . . . . .	48

# 1 Presentasjon

## 1.1 Sammendrag

Som en del av emnet IN2000 *Software Engineering med prosjektarbeid* ved Universitetet i Oslo (UiO) våren 2020, har studentene blitt delt inn i grupper. Hver gruppe har fått i oppgave å utvikle en applikasjon og skrive tilhørende rapport til appen, basert på et valgt case. Denne rapporten er skrevet av seks studenter ved UiOs Institutt for informatikk som har jobbet sammen i team gjennom apputviklingsprosjektet våren 2020. Vår gruppe danner Team 12, og sammen har vi utviklet en applikasjon vi kaller for “Badenymfe”. Badenymfe er utviklet i Android Studio med Kotlin og XML som programmeringsspråk. Denne rapporten beskriver prosjektet vi har vært gjennom, hva vi har utviklet, hvorfor vi har tatt visse valg, og hvordan de er utført. Vi har også lagt til refleksjoner og tanker rundt utfordringer og hva slags tilleggsfunksjoner og endringer vi ville jobbet med fremover dersom vi hadde mer tid, da vi har mange idéer om dette.

## 1.2 Om teamet

Vi i Team 12 er en engasjert gjeng på 6 personer. Fem av oss går studieretningen programmering og systemarkitektur, og det sjette teammedlemmet vårt går design, bruk og interaksjon. Under er litt mer informasjon om hver og en av oss i Team 12:

Elle:

- Alder: 28 år
- Bosted: Asker
- Interesser: teknologi, fjell- og skogstur, brettspill og kortspill.
- Hvorfor caset er interessant i mine øyne: appen vil kunne gi nyttig informasjon for mange brukere fordi de enkelt kan trykke på bestemt lokasjon og få spesifikk ønsket informasjon om en badeplass. Dette er interessant for meg fra et teknisk perspektiv når det gjelder å utvikle en løsning som mange kan bruke til sine forskjellige behov og for å få akkurat den informasjonen de ønsker på kort tid. Det å utvikle en app som er verdifull er interessant for meg fordi vi får et dypere innblikk i spesifikk

utviklingsmetode og app programmering i tillegg til at det er givende å lage noe som kan brukes og hjelpe mennesker i sin hverdag.

Lars:

- Alder: 26 år
- Bosted: Oslo, men kommer fra Nordens Paris.
- Interesser: Jeg har mange interesser, og får nye interesser hele tiden. Jeg har drevet med alt fra fluefisking til tredreiing, musikk og gaming. Det er nesten som en psykisk lidelse; jeg kan bli en smule inspirert av hva som helst, så er jeg obsessed. Tiden rekker aldri til å gjøre alt jeg skulle ønske. Programmering er en av mine favoritt interesser. Det har det vært i snart åtte år og jeg regner med det følger meg livet ut.
- Hvorfor caset er interessant i mine øyne: Nytten av caset som et produkt, treffer meg ikke så veldig godt, da jeg ikke er så glad i å bade. Det å bade synes jeg er kaldt, man blir våt og man får sand og salt over alt. Jeg sitter heller i skyggen - under en parasoll - og nyter en iskald øl. Kanskje kan jeg bruke appen til å finne grunner til ikke å bade? Selve utviklingen av produktet kommer nok derimot til å bli veldig gøy! Ser fram til å lære mer om arkitektur og forskjellige teknologier. Det er alltid gøy å skrive kode.  
Takk for meg!

Johannes:

- Alder: 22 år ung
- Bosted: Oslo
- Interesser: Vintersport, løping og funfacts
- Hvorfor caset er interessant i mine øyne: Ingenting er mer irriterende enn å skulle dra ut til sjøen, og ikke vite hvor det er best å dra. Hvor er det varmt nok badevann for meg? Hvor er det mest sannsynlig at jeg kan unngå maneter? Hvilke steder egner seg for å fange makrell? Faktorene er mange, og kan gjøre valget vanskelig. Å skape en potensiell løsning og gjøre dette valget enklere, gjør dette caset spennende og motiverende i mine øyne.

Aurora:

- Alder: 23 år

- Bosted: Oslo
- Interesser: programmering, snowboard, tegning og musikk
- Hvorfor caset er interessant i mine øyne: Folk flest bader, folk flest hater brennmaneter, og folk flest bryr seg om temperatur i vannet. Selv er jeg oppvokst med mye strandliv, både i Norge og det såkalte syden. Denne appen er noe jeg selv kunne hatt glede av, og noe resten av min familie kunne dratt nytte av. Jeg kommer fra Hamar, med Mjøsa og ferskvann. Vi har imidlertid vært mye i Stavern, og for en ferskvannsfisk som meg er brennmaneter den store frykten. Får vi implementert en løsning for sannsynligheten for brennmaneter et gitt sted, kan mitt badeliv bli ett steg nærmere komplett.

Vilde:

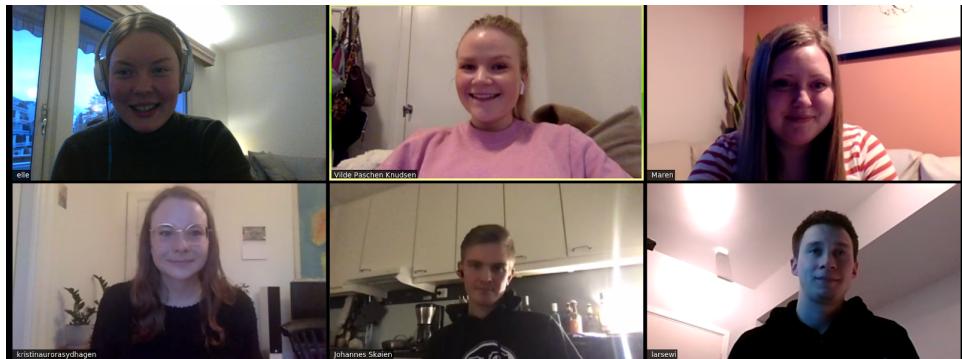
- Alder: 24 år
- Bosted : Oslo/Skøyen
- Interesser: Gå turer, synge og bade
- Hvorfor caset er interessant i mine øyne: Jeg er et sommermenneske, er oppvokst langs kysten, og tilbringer derfor mange timer i sjøen. Men er det en ting som legger en demper på en varm og god sommerdag, så er det sjansen for å bli brent av brennmaneter, da jeg selv kan skrive under på at det fører til et stort ubehag man gjerne skulle vært foruten. I tillegg er nærmest alle som bader interessert i å vite badetemperaturer der man skal. Å sy sammen informasjon om din kommende badetur i en app, tror jeg derfor er noe mange kan få bruk for og vil benytte seg av, fordi folk flest liker å vite hva de går til :-)

Maren:

- Alder: 32 år
- Bosted: Oslo
- Interesser: Bading, baking og brettspill.
- Hvorfor caset er interessant i mine øyne: Jeg elsker å bade, og foretrekker havet fremfor innlandsvann. Folk flest er bekymret for brennmaneter og temperatur, så denne appen vil kunne tilby en tjeneste av allmenn in-

teresse. I en drømmeverden kunne man også sett hvor mange mennesker som bader en bestemt plass, hvor man har gode sjanser ved fiske og mye mer. Jeg håper vi kan få til en app med mye spennende funksjoner, men uansett vil det være en lærerik prosess.

Her er et gruppebilde av oss fra et av våre mange Zoom møter:



### 1.3 Om caset

I dette prosjektet har vi tatt for oss et case om bevegelser i havet, mer spesifikt *Temperaturmålinger for offentligheten*. Caset vi har valgt er relevant for sluttbrukere som er interessert i å finne ut hvor det er behagelig å bade. Badenymfe viser relevant informasjon som vanntemperatur, sannsynlighet for brennmaneter og bølgehøyde på ønsket badepllass i Norge. Dette kan for eksempel være en person som skal sjekke vanntemperatur et spesifikt sted fordi vedkommende lurer på om det er relevant å pakke med seg badetøy før en tur, eller om forholdene ikke er optimale for bading. Andre brukere lurer kanskje på informasjon om bølgehøyde, som f.eks. relaterer til hvor trygt det er å bade en gitt plass. Noen er kanskje livredd for brennmaneter, eller motsatt, interesserte forskere som lurer på hvor det er sannsynlighet for brennmaneter.

Gjennom brukerundersøkelser, smidig metodikk og testing har vi utviklet en applikasjon vi mener er nyttig for sluttbrukere. Vi har produsert noe vi mener er en brukervennlig app som gir personer muligheten til å velge en posisjon i havet og se relevant hav- og værinformasjon om dette stedet. I tillegg kan brukeren legge flere ønskede posisjoner til sin favorittliste som f. eks utvalgte badeplasser, for så å enkelt gå inn på oversikt over disse hver gang brukeren ønsker å se hav- og værdata på de forskjellige lokasjonene. Vi beskriver applikasjonen i mer detalj

i seksjonene under.

## 2 Prosessdokumentasjon

Eventyret Badenymfe startet en tirsdag ettermiddag med et uformelt møte i kantina, der vi alle ble bedre kjent, og diskuterte hvilke forventninger og idéer vi hadde for prosjektet. Det viste seg fort at gruppa hadde god kjemi og at alle var engasjerte og gira på å komme i gang.

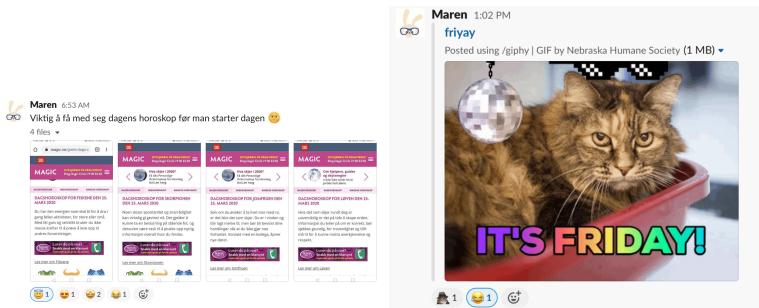
### 2.1 Bestemme case

Det første punktet for å komme i gang med prosjektet var å bli enige om hvilket case vi skulle jobbe ut ifra. Alle i gruppa var enige om at case nr.1 – “bevelgelser i havet” virket spennende, og valget falt naturligvis derfor på nettopp dette caset. Videre presenterte alle i gruppa de idéene vi satt inne med knyttet til caset, og idémyldringen om hvordan Badenymfe skulle bli var i full gang.

### 2.2 Strukturen i gruppearbeidet

Vi oppdaget fort at vi hadde mange idéer og god driv, men for å drive dette prosjektet videre på en god måte, trengte vi struktur. For å organisere oss, ble vi enige om å ha en fast rutine hver uke der vi hadde retrospektiv og sprintplanlegging hver tirsdag kl 12.00, samt et mer uformelt møte på torsdag kl 10.00, der vi kunne lufte tanker og problemer som hadde oppstått siden sist. I hver sprint har vi hatt en Scrum master, referent og en motivator hvor disse rollene rullerte. Selv om vi har anvendt Scrum som metode, var det ikke naturlig for oss å inkludere produkteier som en rolle, da vi ikke anså det som relevant i vårt prosjekt, ettersom vi alle i teamet “eier” prosjektet (Sommerville, 2020). Scrum masteren har hatt som ansvar å lage en agenda for tirsdagsmøtet, der vi har hatt en retrospektiv fra forrige sprint, samt planlagt hvilke oppgaver vi skulle gjøre videre i neste sprint. Referentens oppgave var naturlig nok å skrive referater fra hvert møte, slik at vi lett kunne finne tilbake til beslutninger som ble tatt i plenum, samt at de har vært fine å ha til rapportskriving. Vi ville også ha med en motivator, slik at moralen og engasjementet i gruppa ble opprettholdt på best mulig måte. Denne rollen ble svært viktig da Korona-situasjonen oppstod,

og moralen fort kunne dabbet av, og frustrasjonen lett kunne ta overhånd. I starten av prosjektet tok motivatoren gjerne med noe godt til gruppemøtene, for eksempel brus og O’boy. Dette ble naturlig nok umulig etter at Norge stengte ned. Allerede i starten av semesteret motiverte motivatoren ved å sende bilder, GIFs og liknende til gruppen på Slack. Dette ble en populær praksis for motivatorene resten av semesteret. Andre digitale måter motivatoren utførte sin rolle på var Kahoot over Zoom og en selvlagd liste over 10 grunner til å ikke fortvile over Korona. Motivatorens rolle var ikke lenger kun å motivere til å gjennomføre prosjektet, men også å skape en boost i en hverdag som føltes rar for oss alle (se eksempel i figur 1).



Figur 1: Motivasjon til gruppen på Slack

### 2.3 Valg av digitale kanaler og bruk av disse

For at alle skulle ha best mulig oversikt over de forskjellige oppgavene knyttet til prosjektet, brukte vi Slack til kommunikasjon, GitHub til kode, LaTeX til ferdigstilling av rapport, og Google drive til alt som var relatert til rapport, og diverse planer. Da Korona ikke lenger var til å unngå selv i trygge Norge, viste det seg også at Zoom var det som skulle redde Badenymfe, og forøvrig også team-medlemmernes sosiale stimuli. De faste tirsdags- og torsdagsmøtene ble dermed flyttet til Zoom, i stedet for i Ole Johan Dahls hus. Hver mandag, onsdag og fredag hadde vi “standup-møter” på Slack (figur 2) der alle fikk informert om hvordan vi lå an i prosjektet. Denne aktiviteten innebar de tre klassiske spørsmålene, *hva har du gjort siden sist?*, *hva skal du gjøre til neste gang?*, og om det var noen utfordringer. Denne måten å gjennomføre standup-møter på fungerte svært godt, og vi var ekstra glade for å ha implementert dette da Korona-situasjonen oppstod.



Figur 2: Standup på Slack

Utover standup-møter på slack, ble kanalen brukt til generell kommunikasjon, og hyggelige meldinger fra motivator som tidligere nevnt. Naturlig nok ble GitHub brukt til å organisere kode. Noen av medlemmene i gruppa synes GitHub var noe kronglete til å begynne med, men fikk mer og mer dreisen på det etter hvert. Mer om tanker rundt GitHub-prosessen kommer i seksjonen om Refleksjon.

Til alt annet relatert til prosjektet som typisk dokumenter til rapporten, planer, og backlog lagret vi disse i en felles mappe på Google drive, slik at alle kunne ha god oversikt over alle filer til enhver tid. Ettersom vi fort fikk mange oppgaver relatert til prosjektet, lagde vi to backloger, en til oppgaver relatert til appen (med tilhørende user stories i eget skjema) og en til praktiske og rapport-relaterte ting, samt andre oppgaver som måtte gjøres (figur 3, 4 og 5). Disse ble aktivt brukt i starten av prosjektet, men skled noe ut etter hvert som vi fant ut at det var like greit å lage en enkel TODO-liste med oppgaver, hvor vi lett kunne krysse av de oppgavene vi tok på oss dersom vi følte behov for å liste de opp. Vi førte også en tilhørende oversikt på use case, der ID tilsvarer ID use case i backlogen slik at vi hadde god kontroll på at det vi gjorde hadde en nyttig funksjon for bruker

Product backlog items	ID use case	Tasks	ID task	Dependencies	Started	Doing	Done
Research tema for osse	0	research wintertemp research shoreminger research mineraler research risk	0.0 0.1 0.2 0.3	marenzr marenzr marenzr marenzr	13.03.20 13.03.20 13.03.20 13.03.20		17.03.20
Se kart i app	1	Lage prosjekt i git Laste opp kartfunksjon rydd i koden markere et punkt på kartet legg til seleksjonsfun se på hvordan ta i bruk viewModel Mango med mester	1.0 1.1 1.2 1.3 1.4 1.5 1.6	kristsay kristsay kristsay/marenzr kristsay/marenzr 1.2, 1.1, 1.3 1.2 1.1, 1.2	10.03.20 10.03.20 12.03.20		12.03.20
Finne data om temperatur i havet for gitt posisjon	2	Hent API Lage database hente temperatur Vise teknisk beskrivelse Vise temperatur	2.0 2.1 2.3 2.6 2.5	larsewi larsewi 2.1 2.1 2.3	10.03.20 10.03.20		
Finne data om havstrømmingen for gitt posisjon	3	Hent havstrømminger Vise havstrømminger	3.0 3.1	2.1 3.0			
Brukernedleskaps nr 1	4	Bestemme metode Parallelg innsending av data Lage spennslid Lage sandkasse Lage forsidesdesign Lage spennsjema Gjennomføres innsamling Vurdere resultater fra innsamling	4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7	team12 team12 johasko johasko team12 team12 4.2 4.5	17.03.20 18.03.20 17.03.20 18.03.20 18.03.20 19.03.20 	17.03.20	
Testing	5	Vurdere testmetoder	5.1	allemand			

Figur 3: Eksempel på hvordan backlogen relatert til appen så ut

ID	Use Case beskrivelse
0	Som utvikler av appen ønsker jeg at dataen jeg presenterer er basert på kilder slik at brukere kan være fortrolig med at daten vi presenterer sannsynligvis er presis.
1	Som bruker ønsker jeg å kunne navigere meg med et kart slik at det er lett å finne riktig posisjon jeg ønsker å finne data på
2	Som badenymfe vil jeg kunne sjekke temperatur der jeg ønsker å bade for å se om det er varmt nok
3	Som badenymfe vil jeg kunne sjekke havstrømninger der jeg ønsker å bade for å sjekke om det er trygt å legge ut på svøm
4	Som utvikler ønsker jeg at appens funksjonaliteter og utseende er i tråd med brukerens behov/ønsker
5	Som utvikler ønsker jeg at koden jeg skriver er solid og robust slik at appen ikke kræsjer.
6	Som badenymfe ønsker jeg å kunne få en oversikt over været for å se om det er passelig å bade i dag

Figur 4: Eksempel på hvordan backlogen relatert til use case ID

Task category	ID category	Task	ID task	Dependencies	Started	Doing	Done
Rapport-relatert		0 Skrive presentasjon av team 0 Tekstlig beskrivelse av krav 0 skrive funksjonelle/ ikke skrive funksjonelle krav	0.0 0.1 0.2	allemand allemand allemand	17.03.20 13.03.20	24.03.20 17.03.20	
		0 ordne punkt 1 om prosessdokumentasjon 0 ordne punkt 2 om prosessdokumentasjon	0.3 0.4	vildepk vildepk	24.03.20		
Rydding av kode		1 Skriv kommentarer til kode	1.0	larsewi	18.03.20	18.03.20	
Organisering for oversikt		2 Lage overordnet prosjektplan 2 Sette inn tasks i backlog 2 Lage praktisk backlog	2.0 2.1 2.2	marenzr vildepk vildepk	19.03.20 12.03.20 19.03.20	15.03.20 19.03.20	
Oppdatere oss på relevant info		3 Gjennomgang av github 3 Se udacity videoer	3.0 3.1	larsewi team12	17.03.20 17.03.20		17.03.20
Diverse/finner ikke kategori 8-)		4 Levere oblig	4.0	johasko	10.03.20	12.03.20	

Figur 5: Eksempel på hvordan backlogen relatert til praktiske oppgaver så ut

## 2.4 Gjennomføring av sprints

De første ukene bestemte vi oss for hvilke oppgaver vi skulle ta ut av de to backlogene, og førte disse inn i en egen sprint-oversikt for gjeldende uke, slik at vi hadde full oversikt over hvilke oppgaver som ble gjort, og hvem som gjorde de.

Oppgaver relatert til praktisk backlog				
ID	Task	Doing	Done	Brukernavn
4.0	Levere oblig	10.03.20	12.03.20	johasko
2.1	skirve inn tasks i backlogs	12.03.20	15.03.20	vildepk
0.2	Skrive funksjonelle/ikke funksjonelle krav	13.03.20	17.03.20	ellemand
0.1	Tekstlig beskrivelse av use case	13.03.20	17.03.20	ellemand

Oppgaver relatert til product backlog				
ID	Task	Doing	Done	Brukernavn
0.0	Gjøre research på maneter	13.03.20	17.03.20	marenzr
0.1	Gjøre research på temperaturer	13.03.20	17.03.20	marenzr
0.2	Gjøre reseach på fisker			marenzr
0.3	Gjøre research på strømninger	13.03.20	17.03.20	marenzr
1.0	Lage android-prosjekt og laste opp til git	10.03.20	12.03.20	kristasy
1.1	Legge inn kartfunksjon	12.02.20	12.03.20	kristasy
2.0	Hente data fra API	10.03.20		larsewi
2.1	Lage database	10.03.20		larsewi

Figur 6: Sprint i uke 1

Dette ble utgangspunktet for hvilke oppgaver alle skulle gjøre for den kommande uken. På torsdagsmøtene på Zoom oppklarte vi eventuelle usikkerheter ved arbeidet vi satt med, og eventuelt ba om hjelp fra noen andre på gruppa.

Etterhvert som tiden gikk og oppgavene vokste i omfang, ble sprint-oversiktene mindre nytte. De fleste satt med samme oppgave over flere uker, og motivasjonen for å lage stadig nye nesten-kopier av forrige sprint falt. De store oversiktene over backlog og rapport-oppgavene ble dermed et bedre verktøy i prosessen. Vi åpnet for at de som ønsket en mer detaljert plan uke for uke kunne opprettholde sprint-dokumentet, mens resten kunne holde seg til å oppdatere hovedoversikten. Her kunne vi kanskje vært flinkere til å bryte ned oppgavene til små bestanddeler, men ettersom alle disse uansett ble gjennomført av samme personer, var ikke nytteverdien åpenbar.

Vi fortsatte med sprintplanlegging på tirsdagsmøtene, men uten å lage et eget sprint-dokument. Dermed ble hver enkelt ansvarlig for å ha oversikt over sine

oppgaver og oppdatere hovedoversiktene. Siden alle i gruppen jobbet jevnt og godt, ble ikke arbeidet forsinket av manglende sprint-press. Scrum master kunne i tillegg få en tilsvarende konsistens oversikt ved å lese svarene fra daily standup på slack.

### 3 Brukerdokumentasjon

Badenymfe er en enkel applikasjon som gir brukeren mulighet til å velge en lokasjon i havet for å få informasjon om badeforhold ved valgt sted. Brukeren kan se vær- og havinformasjon for seks dager frem i tid. Som bruker kan man legge til lokasjoner i ”Mine favoritter”, og slette dem dersom de ikke er aktuelle lenger. Informasjonen om vær- og havdata ved lokasjonen oppdateres hver time, slik at bruker raskt kan gå inn på applikasjonen og sjekke forholdene før dagens bad.

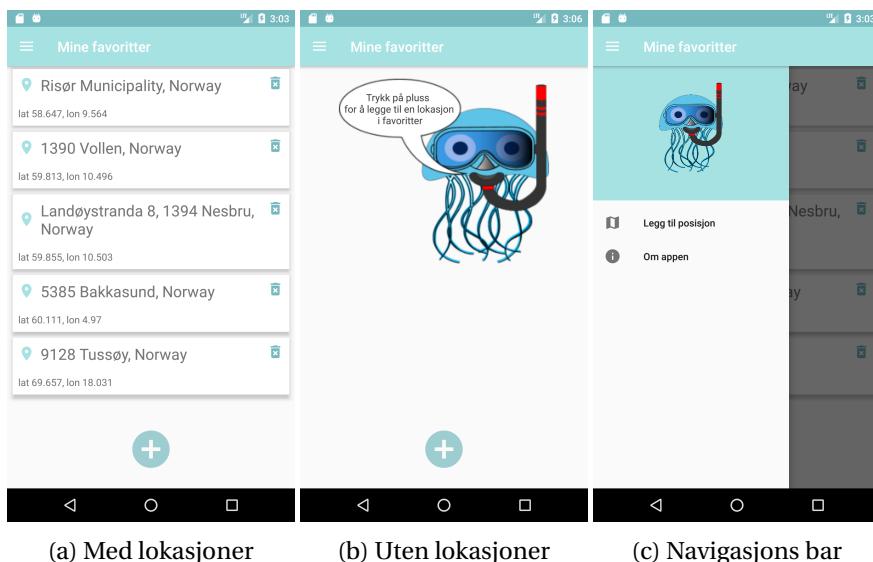
Havmålingene fra API’et Oceanforecast til Meterologisk Institutt er begrenset til målinger fra hav, det vil si det er ikke mulig å få værinformasjon fra lokasjoner på land eller i ferskvann (*Ocean Forecast API*, 2020). Siden posisjonene blir avrundet i til tre desimalplasser er de ikke like nøyaktige som ved vanlig bruk av Google Maps, derfor kan det oppstå en situasjon hvor man trykker på havet, men ikke får målinger for posisjonen. Da kan det være nødvendig å plassere markøren noe lenger fra land. Dette er for å unngå at man legger til flere nesten identiske lokasjoner. Dersom man la til flere nesten identiske lokasjoner fører det mer belastning på Meterologisk Institutt sine servere. Målingene er hovedsaklig knyttet til norsk farvann.

Risikoen for brennmaneter er basert på informasjon om havtemperatur. Forskning om brennmaneter viser at det er tre hovedfaktorer som påvirker hvor manetene befinner seg (Fosette, Gleiss, Karpytchev & Hays, 2015; Nielsen, Pedersen & Riisgård, 1997; *Er badevannet ditt fullt av brennmaneter?*, 2019). Andre faktorer som kan påvirke er retning på havstrømninger og retning på vind. Sistnevnte er imidlertid relevant fordi vind bort fra land fører kjøligere vann opp i overflaten, noe som fører til flere brennmaneter (*Er badevannet ditt fullt av brennmaneter?*, 2019). Maneter kan til en viss grad styre om de ønsker å følge strømningene i havet eller ikke ((Fosette et al., 2015)). Den enkleste indikatoren for å vurdere risikoen er havtemperatur. Dermed er det kun havtemperatur den første versjo-

nen av Badenymfe baserer seg på. Versjon 1.0.1 av Badenymfe vil implementere havstrømninger i algoritmen.

### 3.1 Funksjonalitet og struktur

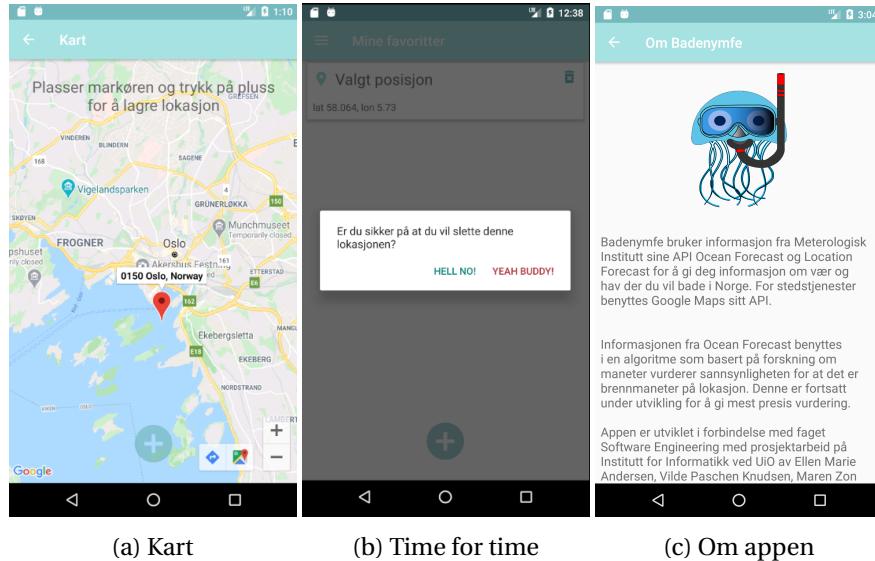
Når man starter Badenymfe kommer man inn på "Mine favoritter" (figur 7). Her ligger alle tidligere lagrede badeplasser (figur 7a), og man kan navigere til kartet for å velge nye lokasjoner ved å trykke på plussstegnet eller via navigasjonsbaren i venstre hjørne (figur 7c). Dersom du ikke har noen lagrede lokasjoner, for eksempel ved første bruk, vil det være et bilde med en snakkeboble som gir instruksjoner til hvordan man kan legge til en lokasjon (figur 7b).



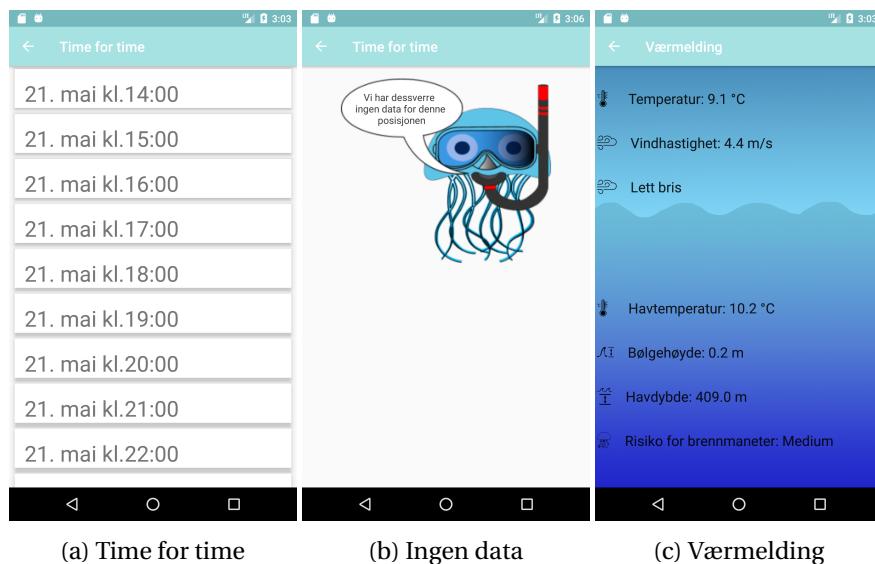
Figur 7: Mine favoritter

Dersom du tillater appen å finne din posisjon vil kartet zoome inn i nærheten av der du er. Hvis du ikke tillater posisjonsdeling må du selv zoome inn på området du er interessert i. Inne på kartet kan du markere en posisjon. Dersom du trykker på posisjonen vil en adresse dukke opp, og du kan velge å bli sendt til Google Maps-appen for å få veibeskrivelse til badeplassen. Dersom du trykker på plussstegnet vil posisjonen legges inn i "Mine favoritter" med den nærmeste tekstlige adressen Google Maps klarer å finne (figur 8a). For å navigere tilbake til "Mine favoritter" uten å legge til en posisjon kan man trykke på pilen i øverste venstre hjørne. Når du har lagt til favoritter er det mulig å slette dem ved å trykke

på knappen i øverste høyre hjørne (figur 7a). Du vil måtte bekrefte slettingen, så det er ingen fare dersom du trykker på delete-knappen ved en feiltakelse (figur 8b).



Figur 8: Kart, slettefunksjon og om appen



Figur 9: Værmelding for en lokasjon

Når en lokasjon er lagt inn i "Mine favoritter", kan man trykke på lokasjonen for å få opp værmeldingen. Ved å trykke på lokasjonen kommer man først til

en liste over tidspunkter med værmelding (figur 9a), med mindre det ikke finnes data for denne lokasjonen (figur 9b). Sistnevnte kan for eksempel oppstå dersom lokasjonen er på land. For å få opp værmeldingen (figur 9c) trykker man på et tidspunkt (figur 9a).

Via navigasjonsbaren kan du også lese litt om applikasjonen (figur 8c).

### 3.2 Målgruppe

Basert på resultater fra brukerundersøkelsene våre har vi stor tro på at vår målgruppe består av varierte personas produktet vårt kan nå ut til. Som bruker av applikasjonen vår er det en selvfølge at man er teknologisk kyndig, som naturligvis vil utelukke noen deler av markedet. Vi innser også at noen personer kanskje ikke er i målgruppen nå, men vil utvikle seg til å bli det. Et eksempel på dette er en nyfødt som ikke har utviklet seg stor nok enda til å begynne å anvende teknologi, men som sannsynligvis gjør det etter noen år. Selv om brukerne våre må kunne bruke og forstå teknologi til å åpne applikasjonen, har vi gjennom hele utviklingsløpet hatt brukervennlighet og universell utforming i bakhodet, slik at vi møter flest mulige behov. Som et resultat av vår oppfattelse av markedet vårt har vi utviklet følgende oversikt over de forskjellige brukerne vi ser for oss som representative for målgruppen:

#### Siri Svømmeentusiast:

Daglig leder på bakeri på Storo, 39 år og bor på Grefsen.

**Interesser:** Siri jobber for å leve. Det vil si at hun elsker fritid og å være aktiv. Svømming og bading er en av yndlingsaktivitetene hennes og hun tar ofte med seg venner og familie på badetur på forskjellige strender i Oslo-området. Hun er spesielt glad i å gå tur på Bygdøy, og å ende turen med en dukkert. I sine yngre dager var hun idrettsutøver i svømmeklubb og ekstra god på butterfly. Hun trives med å tenke tilbake på denne tiden og er glad i å holde seg i form. Når hun drar ut for å bade hender det ofte hun legger opp til konkurranser, enten med seg selv eller med venner og familie. Det kan være alt fra stafettunder i vannet til en lek hvor raskeste person vinner. Hun er et konkurransemenneske og liker helst å vinne. Siri trives som singel og ønsker seg ikke barn men er veldig glad i å leke med nevøene sine.

**Frustrasjoner/bekymringer:** Når ting ikke går som planlagt blir Siri frustrert. Hun blir også litt lei når hun ikke vinner men jobber hardt for å forbedre seg til neste gang. Siri sjekker ofte Yr-appen når hun planlegger turer til havet for å se hvor det er best mulig forhold.

**Bestefar Bengt:**

Pensjonist og ivrig bestefar, 69 år og bor i Lillestrøm.

**Interesser:** Bengt er en pensjonert ungdomsskolelærer som synes han plutselig fikk veldig mye fritid etter han gikk inn i pensjonisttilværelsen. Sammen med sin kone, Bodil, er de begge opptatt av å være aktive og bruker mye tid med sine tre barnebarn på 3, 5, og 8 år. De er opptatt av å ta med barnebarna ut i friluft og fokuserer på å skape gode minner sammen. For tiden har de oppdaget at flere av barnebarna liker sjøen godt, og har blitt inspirert til å lære dem å svømme over og under vann. Trygghet er et viktig aspekt og de vurderer alltid forholdene før de beslutter om det er ansvarsfullt å ha med barnebarna i vannet.

**Frustrasjoner/bekymringer:** Bengt er selv aktiv og blir frustrert når andre er negative om hans livsstil eller om det å være aktiv. Han har lært hvor viktig det er å holde seg i form etter mange års erfaring og blir bekymret når han ikke opplever at han strekker til ved å hjelpe eller overbevise andre om at det er viktig.

**Tøffe Tom:**

Student på videregående og ekstrahjelp på Anton Sport, 16 år og bor på Høvik.

**Interesser:** Tom er en aktiv ungdom som liker nye utfordringer. Han spiller fotball på fritiden og har nylig blitt inspirert til å prøve seg på triatlon. For å klare det ønsker han å forbedre svømmeferdighetene sine. Derfor har han begynt å øve men syns svømmehallen koster mye penger, og han trives best alene når han trener. Derfor liker han å svømme utendørs, og gjerne i havet. Han prøver å balansere skole, jobb, fotballtrening og øvelse til triatlon så godt som mulig.

**Frustrasjoner/bekymringer:** Tom mislikter brennmaneter sterkt. Dette fordi det påvirker treningseffektiviteten hans. Derfor forsøker han å finne steder å trenere det er lite brennmaneter og optimale forhold, og blir irritert når han ikke får det som han ønsker.

### **Anne Aleneforsørger:**

Jobber som helsesøster på helsestasjon i Drammen, 51 år og bor i Lier.

**Interesser:** Anne har en sønn på 12 og en datter på 14 år. Barna hennes er første prioritet og hun bruker mesteparten av tiden sin ved siden av jobb på å oppdra dem og være med dem. Hun ønsker det beste for barna sine og er glad i kvalitetstid med dem. Kvalitetstid for Anne er når hun og barna opplever noe sammen. Dette kan være en reise, å se en spennende film, eller det å være aktive sammen. Sistnevnte resulterer ofte i at de drar på orientering eller til strender og tjern for å bade og ha piknik. Siden barna synes det er flaut å bli sett med moren sin på populære strender hvor de kjenner mange har familien begynt å utforske andre steder å bade. Anne forstår barna og forsøker sitt ytterste for å gjøre dem fornøyde samtidig som de tilbringer kvalitetstid og skaper minner sammen.

**Frustrasjoner/bekymringer:** Anne blir bekymret når hun ser at barna ikke har det godt, og blir frustrert om hun opplever at hun ikke vet hvordan å løse dette eller ikke kommer gjennom til barna sine. Hun har god kommunikasjon med dem og bruker det som et effektivt verktøy, men opplever også at kvalitetstid bringer dem nærmere sammen og at barna åpner seg mer når de får til dette.

### **3.3 Scenarios**

Underveis i designprosessen har vi utviklet scenarios for å danne oss et tydeligere bilde av hvordan bruken av applikasjonen kan være.

#### **3.3.1 Scenario en**

Da søsteren til Siri skal på helgetur til København, skal Siri være barnevakt for sine kjære nevøer Per og Ole. Både Per og Ole er rastløse sjeler, så Siri planlegger å dra en tur til Huk for å bade. For å gjøre badeturen litt morsommere, legger Siri opp en bade-stafett for de to nevøene sine. Siri har fått høre at Ole ikke liker å bade dersom bølgene er for store, og må derfor vite om vannet på Huk er forholdsvis rolig den dagen hun planlegger badeturen. Samtidig er Per en pyse som ikke liker å bade dersom vannet er under 18 grader. For å få gjennomført en vellykket badetur er Siri avhengig av å sjekke badeholdene på Huk.

### **3.3.2 Scenario to**

Tom har meldt seg på Oslo Triatlon som går av stabelen lørdag 8. august, og skal ut på en svømmeøkt for å trenere seg opp til den harde konkurransen. Været er fint, han tar på seg raske briller og bestemmer seg for å ta turen ut til Kadettangen. Det Tom ikke kunne forutse, er at vannet kryr av brennmaneter. På bakgrunn av dette blir treningsøkten lite optimal, da Tom stadig må se seg for etter brennmaneter som gjør at tempo blir dårlig. Han blir svært frustrert da Oslo triatlon er hans største drøm, og svømmeformen ikke er i nærheten av der den bør være på dette tidspunktet om han skal ha sjans til å hevde seg blant topp 10. I full frustrasjon ringer faren sin for å få ut sitt sinne, og for å spørre om spons til klippekort i Nadderudhallen, da han anser dette som eneste løsning for å nå målet i august.

## **3.4 Applikasjonens tilgjengelighet**

Badenymfe er utviklet til Android-mobilenheter med API 23 og oppover. Den er tilgjengelig via GitHub eller som zip-fil fra utviklerne i Team 12.

# **4 Brukerundersøkser**

## **4.1 Første brukerundersøkelse**

For å kunne komme raskt i gang med prosjektet ble det tidlig avgjort at vi skulle begynne programmering med grunnfunksjonaliteter umiddelbart. Siden vi valgte et case som tok utgangspunkt i havdata, ble arbeidet med å hente inn API fra Ocean Forecast og Google Maps påbegynt fra første uke. Da vi hadde en oversikt over de mulige dataene vi kunne innhente fra MET sine API og, et utgangspunkt for arbeid, ønsket vi input fra brukere for å se om våre tanker var i tråd med deres. Dette gjaldt både med tanke på informasjon de kunne hente fra appen, og design. På dette stadiet kunne man ønsket å få interaksjon med et bredt utvalg brukere hvor man presenterte en lavoppløselig prototype, men på grunn av Korona var alle teammedlemmer nå begrenset til fysisk interaksjon med et lite antall andre mennesker. For ordens skyld inkluderte vi samtykkeskjemaer slik at brukerne var informerte om hva undersøkelsen skulle

brukes til, samtidig som det også opplyste brukerene om deres rett til å trekke seg dersom det skulle være ønskelig (se vedlegg B).

#### **4.1.1 Metode**

Da vi ønsket idéer for bruk og design ut over det vi selv hadde fantasi til å komme på, gjennomførte vi et semistrukturert intervju med de vi hadde tilgjengelig i husholdningen (se vedlegg C).

Det må derfor tas i betraktning at brukerne har en tett relasjon til de som gjennomførte brukerundersøkelsene. Dette kan dermed ha vært med på å påvirke resultatene fra undersøkelsen, og tilbakemeldingene kan derfor bære preg av bias. Likevel anså vi det som nyttig å gjennomføre undersøkelsen, til tross for nære relasjoner, da dette var eneste mulighet vi hadde for å gjennomføre en slik undersøkelse på gitt tidspunkt. Vi valgte å gjennomføre en kvalitativ undersøkelse siden dette i større grad ga respondenten mulighet til å komme med frie innspill og idéer. Ved kvantitative undersøkelser som f.eks. ved bruk av spørreskjemaer, kan noe av friheten i svarene utgå, siden alternativene primært er valgt av oss. Semi-strukturererte intervjuer gir også deltakeren mulighet til å styre intervjuet selv innenfor visse rammer, og kan føre til friere og mer utdypende svar. Dermed skaffet vi oss detaljerte data på kort tid.

#### **4.1.2 Presentasjon av undersøkelse med funn**

Det ble gjennomført semi-strukturerete intervjuer av syv brukere i alderen 23 til 62 år, fem menn og to kvinner. Kun en av deltakerne var student, de andre i arbeid, og varierende erfaring med teknologi. Med utgangspunkt i informasjon tilgjengelig gjennom MET API ble deltakerne spurta hvilken informasjon de kunne tenke seg i en bade-applikasjon. Alle ønsket seg informasjon om havtemperatur og temperaturen i luften/været generelt. Andre ting de ønsket seg var informasjon om tidevann (antall: 4), havstrømmer (4), dybde i vannet (4), vindhastighet/retning (3), bølgehøyde (3), nedbør (1), is (1), UV-stråler (1), vurdering av trygghet (1). Informasjon utover den vi hadde tilgjengelig i MET sitt API som ble nevnt, var stort sett relatert til badesteder, blant annet om det var sandstrand, fasiliteter, om det var trygt å spise blåskjell osv.

Med tanke på stedstjenester var det et flertall av deltakerne som ønsket at det

ble gitt mulighet for å bruke «min posisjon» som utgangspunkt, og at man kunne lagre steder som favoritter i appen. Flere deltakere ønsket også mulighet til å ha en liste over populære badesteder i området.

Designmessig fikk deltakerne mulighet til å beskrive hvordan de ønsket at appen skulle se ut ved åpning, ved visning av havdata og hvordan en eventuell meny skulle se ut. De fikk ta utgangspunkt i en skisse dersom de ønsket det. Flesteparten av deltakerne ønsket at appen åpnet seg i kartfunksjonen, og at man hadde en liste med favoritter/badesteder man kunne få tilgang til i en meny nederst på siden. Ved visning av værdata ble det gitt mange gode idéer, og Yr-appen ble referert til av flere som et godt eksempel. Bruk av symboler/farger for å oppgi badevær, vind eller trygghet ble nevnt av flere. To av deltakerne hadde flere innspill til å kunne stille på egne kriterier for å vurdere «godt badevær», «middels badevær» og «dårlig badevær», som da kun ble illustrert til bruker gjennom en fargekode for å raskt vurdere forholdene.

#### **4.1.3 Bruk av funn for videre utvikling av Badenymfe**

I teamets drøfting av resultatene ble det vurdert at man kunne ta utgangspunkt i vær- og havdataen som ble hyppigst nevnt, nemlig havtemperatur og temperatur i luften. Siden dette førte til at Weather Forecast API måtte integreres i appen, vurderte vi det også relevant å ha med generell værdata om vind siden det påvirker badeopplevelsen betydelig når man stiger opp av vannet. Med tanke på stedstjenester hadde teamet allerede et ønske om å benytte «min posisjon» og lagring av favoritter, dette var i tråd med det flertallet av deltakerne ønsket, men mulighet for å ikke gi tilgang til stedstjenester ble vurdert som veldig relevant med tanke på design og testing videre. Det ble avgjort å ta utgangspunkt i et enkelt design med et recycler view som startside for å vise lagrede favoritter. Kartet åpnes når bruker skal legge til en badeplass, og åpner på brukerens posisjon. Designmessig ble det vurdert å ta med forslag videre i utvikling av appen, for eksempel ønsker om å ha en “om appen” side, bruke farger for å visualisere om noe er bra eller dårlig, og generelt holde seg til et enkelt design.

## **4.2 Andre brukerundersøkelse**

På tampen av prosjektet anså vi det som vesentlig med ny input fra brukere, da vi trengte å teste hvorvidt applikasjonen var intuitiv å ta i bruk, samt å få innspill på endelige designvalg.

### **4.2.1 Metode**

Da vi hadde utarbeidet en høyoppløselig prototype av appen, anså vi det som mest nyttig å gjennomføre en kvalitativ brukerundersøkelse. Vi bestemte oss for å gjennomføre brukbarhetstesting i kontrollerte omgivelser som inkluderte testing, spørre brukerne, samt observere dem underveis som testingen ble utført (se vedlegg D). På denne måten kunne vi få konkrete tilbakemeldinger på design-egenskapene til prototypen slik den var på gitt tidspunkt, og gjøre endringer på grunnlag av tilbakemeldingene vi fikk. Ettersom tiden begynte å bli knapp, og Korona gjorde det vanskelig å finne brukere, bestemte vi oss for å gjøre undersøkelsen på kun to brukere. I en annen setting ville vi hatt med flere brukere, samt tatt i bruk triangulering for å verifisere prototypen ytterligere. På samme måte som ved første brukerundersøkelse, må det også tas i betrakning at relasjonen mellom bruker og vi som gjennomførte undersøkelsene er tett, noe som kan medføre bias.

### **4.2.2 Presentasjon av undersøkelse med funn**

Det ble gjennomført brukertest med to brukere, en mann og en kvinne som begge er innenfor målgruppen. Målet med brukertesten var å finne ut om appen var intuitiv å navigere seg i, og få innspill fra brukerne om hva som kunne gjøre den lettere å bruke eller gi bedre visuell fremstilling av informasjonen. Videre var sentrale brukbarhetsmål i denne undersøkelsen tilfredshet, og effektivitet, da det er dette vi streber mot. Det var dermed fokus på brukervennlighet, oppsett, visuell utforming, relevans av data og forbedringspotensiale. Brukerne fikk fem oppgaver.

Brukerne fikk først oppgaven å gå inn i appen, og finne hav-data fra et ønsket sted. Begge brukerne brukte litt tid på å finne ut hvordan de kunne legge til posisjon, og oppfattet at de bare skulle markere musen på kartet for å legge til po-

sisjon. Etter noen sekunders bruk hadde de navigert seg gjennom alle sidene på appen, og fant frem til pluss-knappen på kartet. Det var dermed ikke intuitivt for noen av brukerne hvordan man skulle legge til første lokasjon, men det tok ikke lang tid å bli kjent med appen. Andre spørsmål i brukertesten var innspill når det gjelder hvilke vær- og havdata de syntes var relevant. Den ene deltakeren syntes det var mest aktuelt med følgende informasjon fra API'ene: lufttemperatur, vindhastighet, havtemperatur, sjødybde, og sannsynlighet for maneter, mens den andre deltakeren ville gjerne se lufttemperatur, vindhastighet, vanntemperatur og bølgehøyde. Tredje spørsmål gjaldt tidsrammen for informasjonen i appen. På gitt tidspunkt viste den alle timer den finner informasjon om i API'ene. En av brukerne mente det var mest aktuelt med 12-24t frem i tid, og så for seg at man uansett ville gått inn samme dag for å sjekke badetemperatur. Den andre brukeren ønsket informasjon fem dager frem i tid, slik som Yr har. Fjerde oppgave var å forklare mulighetene for å navigere seg i appen. Begge brukerne ble fort fortrolige med navigeringen, og klarte å forklare navigeringen. Begge savnet fortsatt mer forklaring om hvordan man legger til posisjon i kartet, selv om det opplevdes mer intuitivt etter litt bruk. Siste punkt var for brukeren å komme med innspill til endringer. I tillegg til det de allerede hadde nevnt, kom det innspill på formatet av hvordan tiden ble vist i time-for-time da det på dette tidspunktet var en tid og dato på følgende format: 2020-05-10 12:00:00. Det kom også innspill på tittelen på "hjemmesiden" som het "mine lokasjoner", at det hadde vært mer forklarende med "mine favoritter". Brukerne var i hovedsak ellers fornøyd med både funksjonalitet og visuell fremstilling.

#### 4.2.3 Bruk av funn og videre utforming av Badenymfe

Da begge brukerne slet med å legge til lokasjon fra kartet fra prototypen, bestemte vi oss for å legge inn en tekst øverst i kartet som eksplisitt ber bruker om å trykke på pluss når brukeren vil legge til en lokasjon.

I prototypen hadde vi inkludert all vær- og havdata som finnes fra API'ene, og ønsket å kun inkludere den dataen som virket mest relevant for brukerne, og holde antall variabler presentert til ca 7 objekter. Denne avgjørelsen er tatt på grunnlag av at teamet kom frem til at flere variabler ville bli rotete, noe som er i henhold med Millers lov som sier at mennesker kan kun ha ca 7 objekter i kortidshukommelsen om gangen, og mer informasjon kan overbelaste

brukeres minne, noe som er tegn på et mindre godt design (Preece, Sharp & Yvonne, 2015). Da begge brukerne var ganske samstemte her om hva de ønsket av vær- og havdata, resulterte det i å vise data som inkluderer: lufttemperatur, vindstyrke, beskrivelse av vindstyrke, havtemperatur, sjødybde, bølgehøyde og sannsynlighet for maneter (som ikke var inkludert i prototypen). Ettersom vindstyrke ikke er like intuitivt for alle å forstå, ønsket vi å inkluderte en tekstlig beskrivelse av vindstyrke da vi hadde det tilgjengelig fra Weather Forecast API'et.

Vi ønsket også å finne ut hvor langt frem i tid brukerne ønsket å kunne få tilgang på vær- og havdata. På dette punktet var det uenigheter fra brukerne, og drøftingen av dette punktet ble utsatt, da vi ikke anså oppgaven som kritisk for å få til en fungerende app, men handlet mer om brukernes behov. Dersom vi hadde hatt bedre tid til å utforme appen, ville vi som nevnt inkludert flere brukere i undersøkelsen for å kunne undersøke hva slags tidsperspektiv flere kunne tenke seg. Tidsperioden for tilgang på vær-og havdata ble derfor uendret.

Med tanke på navigering og endringer i appen, gjorde vi noen små justeringer på bakgrunn av tilbakemelding fra brukerne. Utover å presisere med tekst at brukerne på trykke på pluss-knappen for å legge til en lokasjon, savnet de også en tilbake-knapp fra kartet som vi implementerte, samt mer beskrivende overskrifter. Vi endret derfor "Mine lokasjoner" til "Mine favoritter". I tillegg opplevde brukerne det noe forvirrende å få opp en lokasjon uten tilhørende data, da det ikke var tydelig om de hadde trykket feil, om appen ikke fungerte som den skulle, eller om det ikke finnes data på området de hadde trykket på. Om vi skulle videreutviklet denne appen, ville vi gjerne implementert en toast slik at brukeren fikk beskjed med en gang markøren ble plassert, dersom det ikke finnes informasjon på denne lokasjonen. Da dette ble en oppgave som tok mer tid enn vi forventet, løste vi det med å vise en melding på time-for-time fragmentet, med Badenymfe-ikonet som gir en beskjed til bruker om at det ikke finnes data på valgt lokasjon.

### 4.3 Oppsummering av designvalg og forkastede idéer

Siden formuleringen i valgt case rettet seg mot offentligheten, var det naturlig for oss å ta utgangspunkt i gjennomføring av prosjektet med brukersentrert design som tilnærming. Vi involverte brukere tidlig i prosessen, for å kartlegge behov og krav ut ifra mulighetene vi hadde i API'ene, og brukte dette som utgangspunkt

for videre arbeid. Det var underveis elementer av genious design, ettersom noen valg ble tatt av oss i teamet uten gjennomføring av undersøkelser med brukere. Likevel har vi primært basert valg på resultater fra brukerundersøkelsene så langt det har latt seg gjøre.

Brukerundersøkelsene ga oss inntrykk av at deltakerne ønsket et enkelt design, hvor kun det mest relevante av data skulle vises. Det ble av flere vist til designet i Yr-appen, og sagt at de kunne tenke seg noe lignende. Vi hentet derfor inspirasjon fra denne i utforming av egen app. Når det kom til fargevalg, tok vi utgangspunkt i en generator for fargekombinasjoner, og testet diverse fargepaletter. Her ble det ikke gjennomført en formell brukerundersøkelse, men tre forskjellige fargepaletter ble presentert for én bruker, der brukerens ønsker var i tråd med teamets ønsker.

Vi vurderte innledningsvis å følge opp første datainnsamling med en kvantitativ undersøkelse, i form av et spørreskjema. Etter innledende intervjurunder kunne en ny undersøkelse verifisere responsene vi hadde fått. Vi kom imidlertid, gjennom intern diskusjon og samtale med veileder, frem til at det kunne holde med kvalitativ innsamling, ettersom dette ga oss gode data å jobbe videre med.

En stor utfordring med prosjektet var tidsbegrensningen vi måtte forholde oss til mtp. innleveringsfrist. Vi var av denne grunn nødt til å velge bort funksjoner og idéer som brukerne hadde foreslått, og heller prioritere det flest hadde uttrykt ønske om. Også utfordringer med API'ene skapte hindringer for oss. Et eksempel på dette var et ønske fra flere brukere om å kunne søke etter stedsnavn. Dette viste seg å være en betalt løsning i Google sitt API, og måtte derfor forkastes i denne sammenheng. Også gruppens egne ønsker og idéer måtte forkastes underveis i prosjektet. Vi ønsket f.eks. å implementere en egenprodusert loading-bar, men innså at dette ville være for omfattende, og forkastet idéen til fordel for annet arbeid. Det var også lenge en plan om å lage en settings-aktivitet, hvor brukeren selv kunne velge hvilke data som skulle vises. På grunn av tidsbegrensningen ble dette etter hvert nedprioritert, slik at vi kom i mål med andre, viktigere aspekter for en fungerende app. Det er et forbedringspotensiale vi kunne implementert ved bedre tid, og vi lot derfor data fremdeles ligge i prosjektet.

## **5 Kravspesifikasjon og modellering**

### **5.1 Opprinnelige krav**

Med utgangspunkt i første brukerundersøkelse, kom vi frem til følgende krav:

#### **5.1.1 Opprinnelige funksjonelle krav**

1. Appen skal gi brukeren mulighet til å se lufttemperatur ved en gitt posisjon.
2. Appen skal gi brukeren mulighet til å se vindhastighet ved en gitt posisjon.
3. Appen skal gi brukeren mulighet til å se vindretning ved en gitt posisjon.
4. Appen skal gi brukeren mulighet til å se vanntemperatur ved en gitt posisjon.
5. Appen skal gi brukeren mulighet til å se vannstrømning ved en gitt posisjon
6. Appen skal gi brukeren mulighet til å se bølgehøyde ved en gitt posisjon.
7. Appen skal gi brukeren mulighet til å se sannsynlighet for brennmaneter ved en gitt posisjon.
8. Appen skal bruke Google Maps sin kartfunksjon så bruker kan velge ønsket posisjon på kartet.
9. Appen skal gi brukeren mulighet til å velge posisjon på kartet eller skrive inn longitude og latitude som tekst.

#### **5.1.2 Opprinnelige ikke-funksjonelle krav**

1. Appen skal hente vanntemperaturdata og visualisere dette innen 3 sekunder fra brukeren klikker på at han/hun ønsker dette, i 90% av tilfellene.
2. Appen skal hente lufttemperaturdata og visualisere dette innen 3 sekunder fra brukeren klikker på at han/hun ønsker dette, i 90% av tilfellene.
3. Appen skal hente vindhastighetsdata og visualisere dette innen 3 sekunder fra brukeren klikker på at han/hun ønsker dette, i 90% av tilfellene.

4. Appen skal hente vindretningsdata og visualisere dette innen 3 sekunder fra brukeren klikker på at han/hun ønsker dette, i 90% av tilfellene.
5. Appen skal hente vannstrømningsdata og visualisere dette innen 3 sekunder fra brukeren klikker på at han/hun ønsker dette, i 90% av tilfellene.
6. Appen skal hente bølgehøydedata og visualisere dette innen 3 sekunder fra brukeren klikker på at han/hun ønsker dette, i 90% av tilfellene.
7. Appen henter brukers lokasjon og zoomer inn på nåværende lokasjon ved hjelp av Google Maps, med en radius på 2km slik at bruker kan velge ønsket posisjon på kartet med utgangspunkt i egen posisjon. Dette vises innen 1 sekund.
8. Appen skal hente data om sannsynlighet for brennmaneter og visualisere dette innen 3 sekunder fra brukeren klikker på at han/hun ønsker dette, i 90% av tilfellene.
9. Appen skal ikke lagre og selge brukers posisjon(er) til tredjepart.
10. Appen skal fungere både i landskap- og portrettmodus.
11. Appen skal ikke krasje på API-level 23.

## 5.2 Endelige funksjonelle krav og modellering

Gjennom utviklingsprosessen har noen krav måttet vike, og andre har kommet til. I denne seksjonen vises de funksjonelle kravene som har blitt implementert, samt tekstlige beskrivelser og modellering av applikasjonen slik den ser ut i dag. Mer om hvilke opprinnelige krav som ble forkastet, og hvorfor, kommer i seksjonen *Endringer og implementering av krav*.

### 5.2.1 Endelige funksjonelle krav

1. Appen skal gi brukeren mulighet til å se lufttemperatur ved en gitt posisjon.
2. Appen skal gi brukeren mulighet til å se vindhastighet ved en gitt posisjon.
3. Appen skal gi brukeren mulighet til å se en tekstlig beskrivelse av vind ved en gitt posisjon

4. Appen skal gi brukeren mulighet til å se vanntemperatur ved en gitt posisjon.
5. Appen skal gi brukeren mulighet til å se vanndybde ved en gitt posisjon.
6. Appen skal gi brukeren mulighet til å se bølgehøyde ved en gitt posisjon.
7. Appen skal gi brukeren mulighet til å se sannsynlighet for brennmaneter ved en gitt posisjon.
8. Appen skal bruke Google Maps sin kartfunksjon så bruker kan velge ønsket posisjon på kartet.
9. Appen skal gi brukeren mulighet til å lagre en posisjon så bruker slipper å finne en gitt posisjon på nytt hver gang appen åpnes

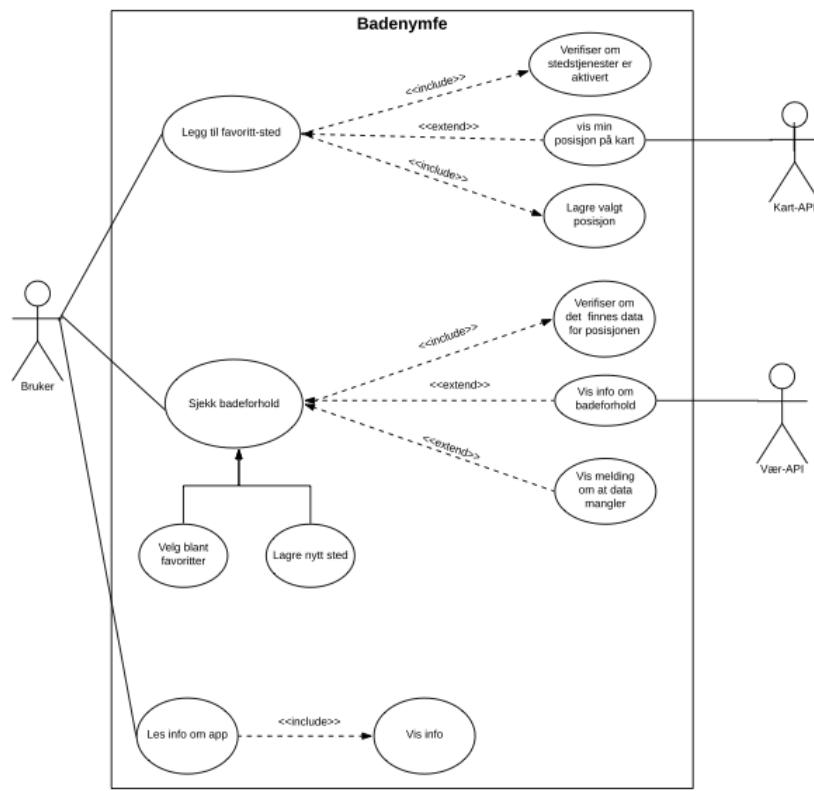
### **5.2.2 Use Case - Modellering**

I følgende diagram har vi modellert de tre use-casene “Legg til favoritt-sted”, “sjekk badeforhold” og “Les Info om app” (*se figur 10*).

I tilfellet “Legg til favoritt-sted” vil appen navigere til kartet, og appen vil alltid sjekke hvorvidt stedstjenester er aktivert i appen. Dersom de er aktivert, vil kartet zoome inn på brukerens posisjon. Når brukeren har valgt et punkt og bedt appen om å legge den til, vil posisjonen bli lagret i “Mine favoritter”

I tilfellet “Sjekk badeforhold” kan brukeren enten velge å sjekke værmeldingen for en allerede lagret favoritt-plass, eller legge til en ny posisjon. Dersom bruker ønsker å sjekke en ny posisjon, utføres stegene i “Legg til favoritt-sted” først. Når bruker har valgt en lokasjon, vil appen sjekke om det finnes data på posisjonen. Dersom det finnes data, vil brukeren kunne navigere til værmeldingen og data om badeforhold vises på skjermen. Hvis ikke, vil en melding om at data ikke finnes dukke opp.

I tilfellet “Les info om app” vil bruker navigere til denne delen av appen, og en tekst med info om appen vises på skjermen.



Figur 10: Use-case diagram

### 5.2.3 Use Case - tekstlig beskrivelse

#### Lagre favorittlokaliteter

**Use case:** Som ny appbruker ønsker jeg å lagre badestedene jeg vanligvis drar til ved å trykke på kartet for å raskt kunne sjekke badeforhold ved en senere anledning

**Aktør:** Appbruker

**Prebetingelser:** Må ha en smarttelefon med Android OS og appen lastet ned

**Postbetingelser:** Valgt favorittposisjon lagres og vises i «Mine favoritter»

#### **Hovedflyt:**

1. Bruker åpner appen
2. Bruker trykker på plussstegnet
3. Appen navigerer til kartet

4. Bruker markerer en posisjon på kartet
5. Bruker trykker på plussknappen
6. Posisjonen lagres blant favoritter
7. Appen navigerer til «Mine favoritter»

**Alternativ flyt:**

- 1.1 Appen fryser
- 1.2 Bruker åpner appen på nytt
- 1.3 Fortsett fra punkt 2 i hovedflyt

- 2.2 Bruker navigerer til kartet via sidemenyen
- 2.3 Fortsett fra punkt 3 i hovedflyt

**Sjekke badeforhold på et nytt sted**

**Use case:** Som badeturist vil jeg kunne sjekke badeforhold på et nytt badested for å kunne forsikre meg om at det blir en varm og behagelig dag på stranden

**Aktør:** Badeturist (bruker)

**Prebetingelser:** Må ha en Android smarttelefon og appen lastet ned

**Postbetingelser:** Data om badeforhold vises for ønsket posisjon

**Hovedflyt:**

1. Bruker åpner appen
2. Bruker trykker på plussstegnet i "Mine favoritter"
3. Appen navigerer til kartet
4. Bruker trykker på en posisjon på kartet
5. Bruker trykker på plussknappen
6. Posisjonen lagres med tilhørende data
7. Appen navigerer til «Mine favoritter»
8. Bruker trykker på posisjonen
9. Bruker trykker på et gitt tidspunkt
10. Data om badeforhold vises for valgt posisjon

**Alternativ flyt:**

- 1.1 Appen fryser
- 1.2 Bruker åpner appen på nytt
- 1.3 Fortsett fra punkt 2 i hovedflyt
  
- 2.6 Kan ikke hente data for gitt posisjon fordi markert sted angir et sted på land
- 2.7 Appen navigatorer til «Mine favoritter»
- 2.8 Bruker trykker på lagret posisjon
- 2.9 Melding om at det ikke finnes data for den gitte posisjonen vises på skjermen
- 2.10 Bruker navigatorer tilbake til «Mine favoritter»
- 2.11 Fortsett fra punkt 2 i hovedflyt
  
- 3.4 Bruker trykker på feil posisjon
- 3.5 Trykker på annen, korrekt, posisjon
- 3.6 Fortsett fra punkt 5 i hovedflyt
  
- 4.2 Bruker navigatorer til kartet via sidemenyen
- 4.3 Fortsett fra punkt 3 i hovedflyt

**Sjekke badeforhold blant «Mine favoritter»**

**Use case:** Som badeturist vil jeg kunne sjekke badeforhold på et allerede lagret favorittsted for å kunne forsikre meg om at det blir en varm og behagelig dag på favorittstranden

**Aktør:** Badeturist (bruker)

**Prebetingelser:** Må ha en smarttelefon, appen må være lastet ned og favorittposisjoner må være lagret tidligere

**Postbetingelser:** Data om badeforhold vises for ønsket posisjon.

**Hovedflyt:**

1. Bruker åpner appen
2. Bruker velger en av sine favorittlokasjoner
3. Bruker velger et ønsket tidspunkt
4. Data om badeforhold vises for valgt posisjon

**Alternativ flyt:**

- 1.1 Appen fryser
- 1.2 Bruker åpner appen på nytt

- 1.3 Fortsett fra punkt 2 i hovedflyt
- 2.2 Favorittlokalisasjonen har forsvunnet
- 2.3 Bruker trykker på plussstegnet
- 2.4 Appen navigerer til kartet
- 2.5 Bruker markerer ønsket posisjon på kartet
- 2.6 Bruker trykker på plussstegnet
- 2.7 Posisjonen med tilhørende data lagres
- 2.8 Appen navigerer til «Mine favoritter»
- 2.9 Fortsett fra punkt 2 i hovedflyt
- 3.2 Bruker trykker på feil favorittsted
- 3.3 Bruker trykker på riktig favorittsted
- 3.4 Fortsett fra punkt 3 i hovedflyt

#### **Sjekke badehold på brukerens posisjon**

**Use case:** Som en badegladd person som befinner seg på en badeplass ønsker jeg å sjekke badeholdene for å vite om vannet er behagelig før jeg kaster meg uti

**Aktør:** Badeturist (bruker)

**Prebettinger:** Må ha en Android smarttelefon og appen lastet ned

**Postbettinger:** Data om badehold vises for brukerens nåværende posisjon

#### **Hovedflyt:**

1. Bruker åpner appen
2. Bruker trykker på plussstegnet i "Mine favoritter"
3. Appen navigerer til kartet
4. Bruker tillater posisjonsdeling
5. Kartet zoomer inn på brukerens posisjon
6. Bruker markerer ønsket posisjon på kartet
7. Bruker trykker på plussstegnet
8. Appen navigerer til «Mine favoritter»
9. Bruker trykker på posisjonen
10. Bruker velger ønsket tidspunkt

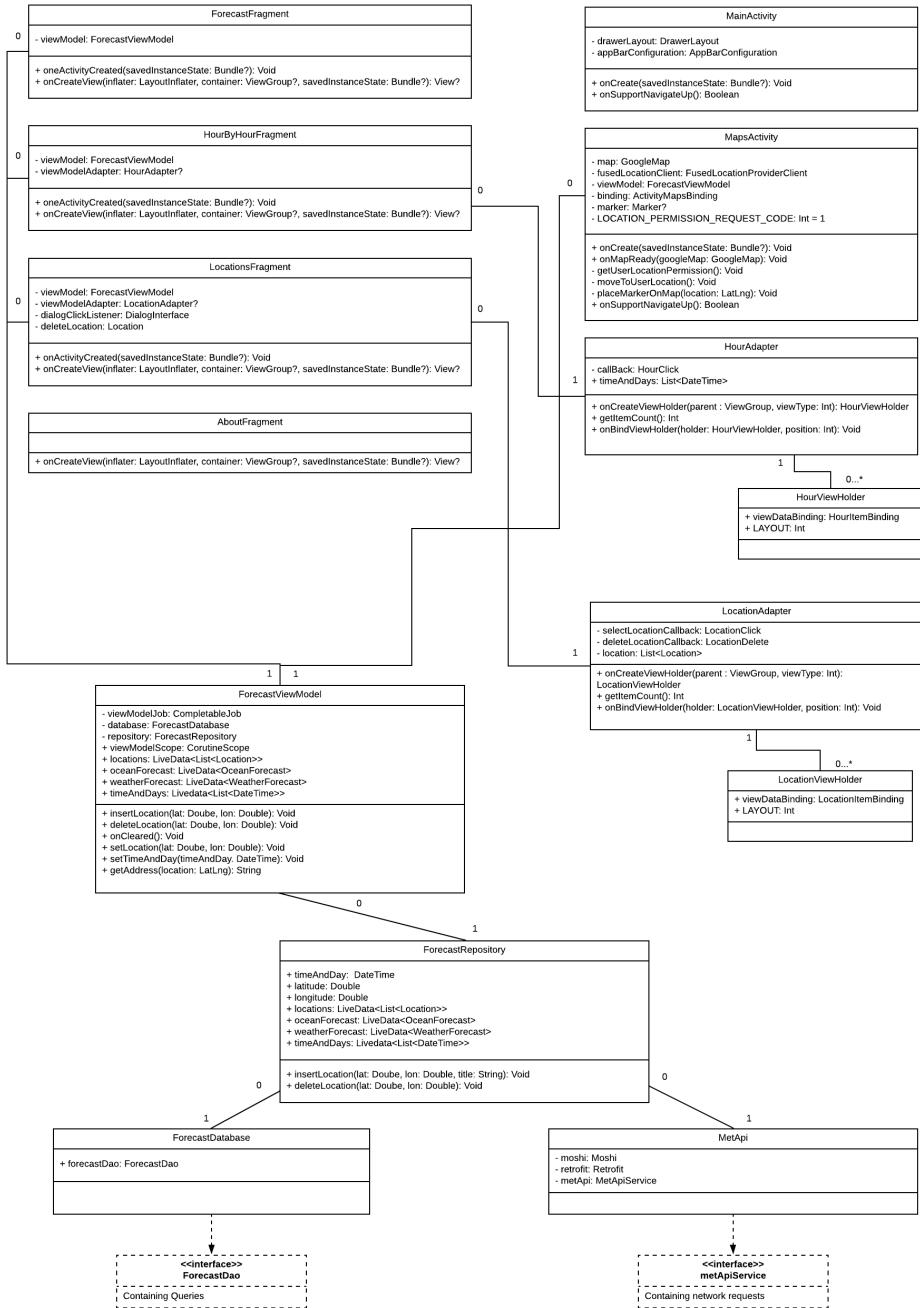
## 11. Informasjon om badeforhold vises for valgt posisjon

### **Alternativ flyt:**

- 1.4 Bruker avslår posisjonsdeling
- 1.5 Bruker zoomer inn mot sin posisjon
- 1.6 Fortsett fra punkt 6 i hovedflyt
  
- 2.2 Bruker nавигerer til kart via sidemeny
- 2.3 Fortsett fra punkt 4 i hovedflyt

### **5.2.4 Klassediagram**

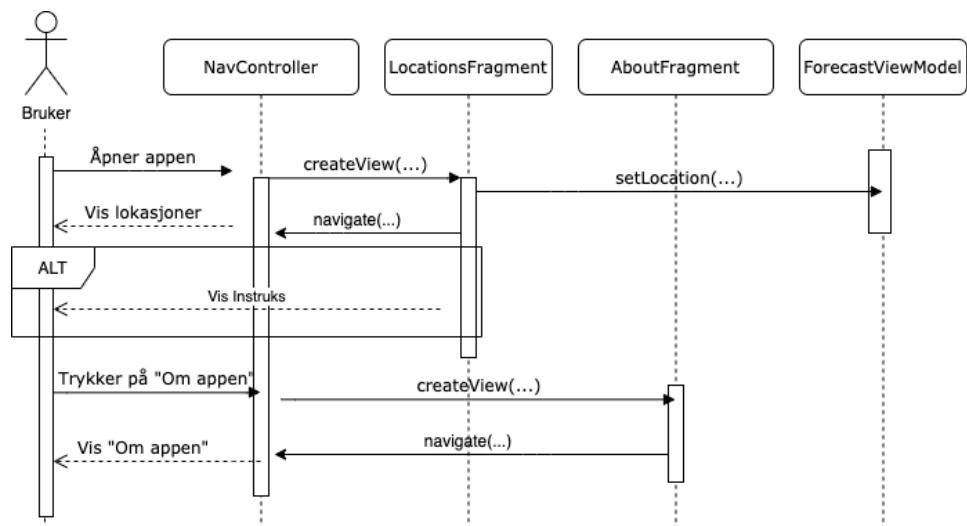
Klassediagrammet viser forholdet mellom klassene i applikasjonen. Koblingene kan tolkes som en rettet asykisk graf i henhold til MVVM arkitekturen, ved at relasjonene mellom klassene går en vei. Retningen kan tolkes ved at noen klasser har [0], det vil si ingen, referanser til den klassen som har en referanse til den, selv om de i prinsippet har en relasjon. Vi har valgt å bruke denne måten for å illustrere det på for å tydeliggjøre arkitekturen. For eksempel kan ForecastViewModel ha en ForecastRepository, mens sistnevnte ikke har en referanse til ForecastViewModel. De er derfor relatert, men det er en enveisrelasjon. Klasser som ikke er essensielle for å forstå strukturen i applikasjonen er ikke tatt med i klassediagrammet.



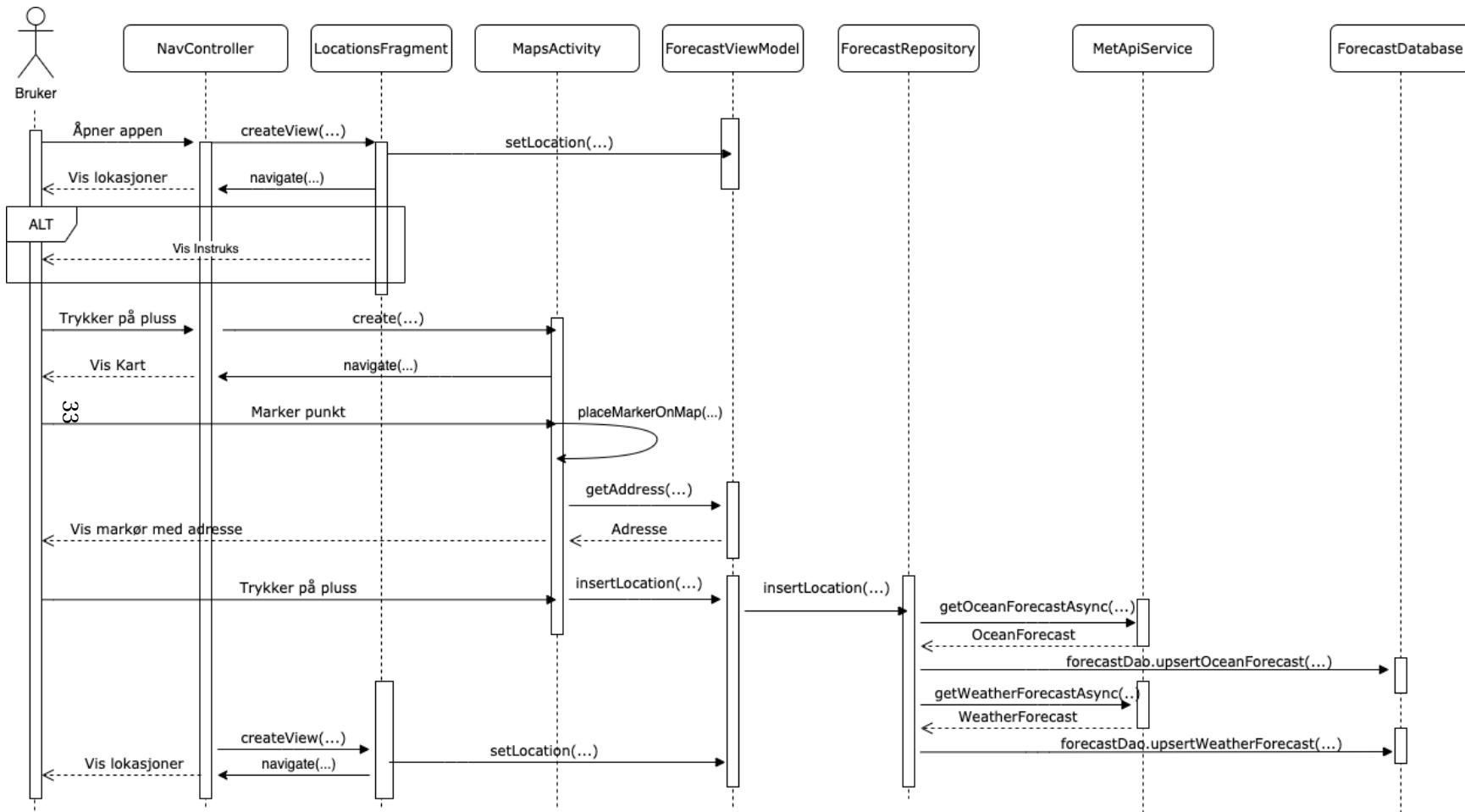
Figur 11: Klassediagramm

### 5.2.5 Sekvensdiagram

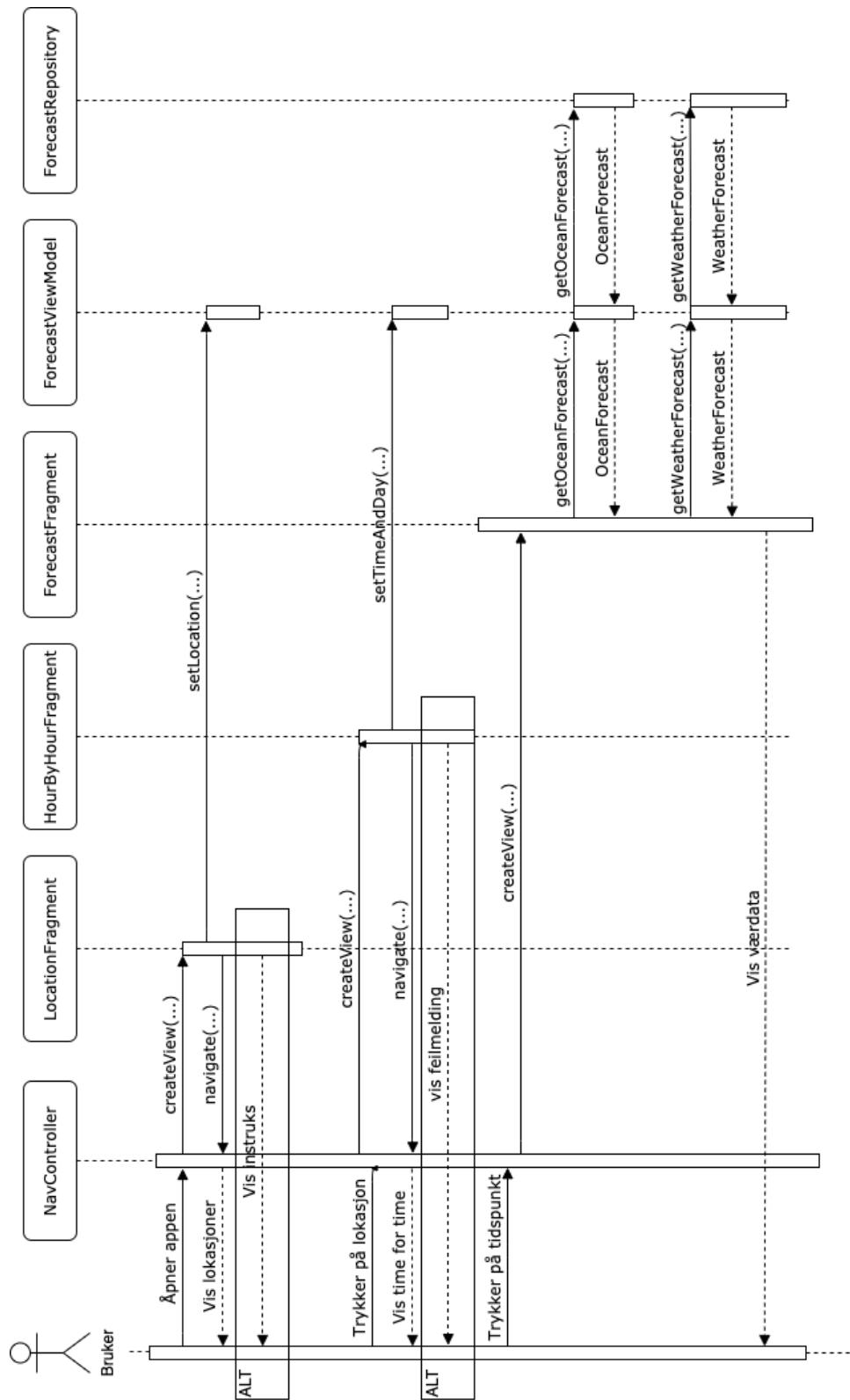
Sekvensdiagrammene viser de tre mest relevante scenarioene. Noen klasser og metodekall er utelatt, men de vi anså som viktigst for å forstå programflyten er inkludert. I første scenario åpner bruker applikasjonen og går inn på "Om appen" for å lese om applikasjonen (figur 12). Neste sekvensdiagram illustrerer hendelsesforløpet når en bruker legger til en lokasjon i "Mine favoritter"(figur 14). Det siste sekvensdiagrammet illustrerer at en bruker åpner appen for å finne værmelding for en posisjon som allerede er lagt inn i "Mine favoritter"(figur 13).



Figur 12: Sekvensdiagram: bruker leser om appen



Figur 13: Sekvensdiagram: bruker legger til posisjon



Figur 14: Sekvensdiagram: bruker sjekker badeforhold på lagret lokasjon  
34

### **5.3 Endringer og implementering av krav**

Etter første runde med brukerundersøkelser, kom vi frem til et sett med funksjonelle- og ikke-funksjonelle krav, som tidligere presentert. Etter mange uker med koding, og en ny brukerundersøkelse på tampen av prosjektet, fikk vi fastslått hvilke krav vi fikk innfridd, og hvilke vi måtte la gå.

Av de funksjonelle kravene, ble alle innfridd med unntak av tre. De første kravene som ikke ble innfridd var å gi bruker mulighet til å se vannstrømninger og vindretning. Til tross for at vannstrømninger i utgangspunktet var ønsket blant flere av brukerne i første runde, ble dette kravet ikke implementert, da vi ingen av brukerne i runde to nevnte at de var interessert i det. Med tanke på vindretning, kom vi frem til at dette ikke er veldig relevant da heller ingen av brukerne i runde to ønsket dette, og at applikasjonen primært er en bade-applikasjon. Vi ville derfor holde værinformasjon til det mest nødvendige. Da første runde av brukerundersøkelser ikke inkluderte en høyoppløselig prototype, men var et semistrukturert intervju med en skisse, anså vi andre runde som mer tungtveiende i begge tilfellene til tross for få deltakere. Det andre kravet som ikke ble implementert var å gi bruker muligheten til å velge posisjon på kartet ved å skrive inn longitude og latitude som tekst. Bakgrunnen for at dette kravet ikke ble implementert, var mangel på tid, samt at vi konkluderte med at longitude og latitude ikke er intuitivt for folk flest å ta i bruk når det kommer til å finne en lokasjon.

Med unntak av kravene nevnt over, ble funksjonelle krav om at appen skal gi brukeren mulighet til å se vanntemperatur, lufttemperatur, vindhastighet, bølgehøyde ved en gitt posisjon implementert. Det gjorde også kravet om at appen skal bruke Google Maps sin kartfunksjon slik at bruker kan velge ønsket posisjon på kartet, samt kravet om å gi brukeren mulighet til å se sannsynlighet for brennmaneter ved en gitt posisjon.

Blant de ikke-funksjonelle kravene var det tre krav som ikke ble implementert, og ett krav som kan diskuteres hvorvidt det ble implementert eller ikke. De to første kravene som ikke ble implementert av de ikke-funksjonelle kravene var å hente vindhastighetsdata og vannstrømningsdata og visualisere dette innen 3 sekunder fra brukeren klikker på kartet i 90% av tilfellene. Disse falt naturligvis bort da vi som nevnt konkluderte med at variablene ikke var nødvendige for brukerne. Det tredje kravet som ikke ble implementert er at appen skal kunne

zoome inn på nåværende lokasjon ved hjelp av Google Maps med en radius på 2km(...). Kodemessig var det vanskelig å finne ut hvordan å implementere en radius fra gitt lokasjon på nøyaktig 2km. Likevel endte vi opp med en radius som visuelt så fornuftig ut med tanke på avstandskravet på 2km.

Noe som kan diskuteres hvorvidt er implementert eller ikke, er kravet om at appen ikke skal lagre og selge brukers posisjon til en tredjepart. Ettersom vi tar i bruk Google-maps sitt API for kartfunksjonen i appen, kan vi ikke være sikre på hvorvidt Google bruker informasjonen fra brukernes posisjon eller ikke, og om de selger denne informasjonen videre. Med tanke på krav som inkluderer at en gitt aktivitet skjer innen en viss tid, er dette også diskuterbart, da vi opplever at emulatoren kjører vesentlig raskere dersom du har en ny maskin enn om man har en eldre maskin som nevnt i seksjonen om testing av ikke-funksjonelle krav. Likevel velger vi å konkludere med at de tidsspesifikke kravene er innfridd, da tiden en aktivitet tar ikke ser ut til å være påvirket av arkitekturen i koden, men heller type hardware man kjører appen på.

Utover disse, ble følgende ikke-funksjonelle krav implementert (forutsatt at man velger en lokasjon i havet):

- Appen skal hente vanntemperaturdata, lufttemperaturdata, vindhastighetsdata, bølgehøydedata, og visualisere denne innen 3 sekunder fra brukeren klikker på at han/hun ønsker dette i 90% av tilfellene.
- Appen skal hente data om sannsynlighet for brennmaneter og visualisere dette innen 3 sekunder fra brukeren klikker på at han/hun ønsker dette i 90% av tilfellene.
- Appen skal fungere i både landskap- og portrettmodus.
- Appen skal ikke krasje på API-level 23.

API-level 23 ble valgt fordi det ifølge Android Studio IDEA dekker 84,9% av Androidenheter som er i bruk i dag. I tillegg vil et såpass nytt API gi oss tilgang til flere tilleggsfunksjoner og bedre ytelse.

## 5.4 Universell utforming

Med tanke på universell utforming, er det visse kriterier applikasjonen dekker fra WCAG 2.1 standarden, mens andre kriterier har ikke blitt tatt hensyn til

slik applikasjonen er nå. Da retningslinjene fra WCAG 2.1 er mange, velger vi å kommentere noen vi anser som mest relevante for vår applikasjon.

Kriterier som burde tas hensyn til ved videre utvikling av applikasjonen er fargebruk med tanke på kontraster. I retningslinje 1.4.3 anbefales det at kontrastforholdet mellom tekst og bakgrunn er minimin 4,5:1 (*Kontrastforhold mellom tekst og bakgrunn*, 2020). Dette kontrastforholdet er ikke opprettholdt i applikasjonen der overskrifter har hvit tekst på lyseblå bakgrunn (*Contrast Ratio*, 2020a), samt i grensesnittet hvor havdata blir presentert med sort tekst på en gradert mørkere blå bakgrunn (*Contrast Ratio*, 2020b). I retningslinje 1.4.11 anbefales det videre at kontrast for ikke-tekstlig innhold skal ha et kontrastforhold på minst 3:1 mot farger som ligger ved siden av (*Kontrast for ikke-tekstlig innhold*, 2020). Her er det også forbedringspotensiale da applikasjonen ved flere tilfeller ikke dekker dette kravet, derav hvor pluss-knappen for å legge til en lokasjon er for lys i forhold til blåfargen på kartet. Ettersom karter varierer i farge, var det vanskelig å finne en farge som hadde bra nok kontrast både i vann og på land, men som samtidig var innenfor fargepaletten vi ønsket. Overgangen fra tool-bar til hvit bakgrunn er heller ikke innenfor kontrastkravene.

Retningslinjer vi derimot følger fra WCAG 2.1 er punkt 1.3.4 som anbefaler at bruker skal kunne velge mellom landskap og portrettmodus (*Visningsretning*, 2020). Likevel må det nevnes at designet i landskapsmodus ikke optimalt tilpasset, da bruker må scrollle nedover for å se all presentert informasjon i stedet for at informasjonen tilpasser seg skjermen på en smidig måte. Videre har vi tatt hensyn til punkt 2.5.2; uheldige eller feilaktige inputs via mus eller berøringsskjerm skal lettere kunne forhindres (*Pekeravbrytelser*, 2020). Dette har vi implementert ved å la bruker få opp en melding om han eller hun er sikker på at lokasjonen skal slettes dersom krysset i hjørnet av lokasjonen er trykket på.

## 6 Produktdokumentasjon

### 6.1 Arkitektur

Under utviklingen av Badenymfe applikasjonen har vi prøvd å følge Android Developer sin guide til best practice og anbefalt arkitektur (*Guide to app architecture*, 2019). Dette har vi gjort for å få en mer robust applikasjon med

høy kohesjon og lav kobling.

### 6.1.1 Høy kohesjon & lav kobling

Med høy kohesjon menes det at alle deler av en komponent har en naturlig sammenheng. Eksempelvis kan vi si at en del av en komponent kun bør gjøre én bestemt oppgave, og at denne oppgaven bør være sterkt relatert til denne komponenten.

Lav kobling går ut på å unngå mest mulig avhengigheter mellom ulike komponenter. Avhengigheter fører ofte til en dårligere struktur og et program krever totalt sett mer logikk for å kunne kjøre på en sikker måte. En annen ulempe med avhengigheter viser seg når en skal skrive enhetstester til programvaren. Slike tester kan bli svært kompliserte. Med mange koblinger må alle avhengighetene simuleres for å kunne kjøre testen.

Høy kohesjon fører ofte til lav kobling, og vice versa. Dette er noe som kan gjøre disse to begrepene litt forvirrende i blant. Årsaken til denne sammenhengen kan for eksempel være at en programvarekode med høy kohesjon har alle objekter sterkt relaterte egenskaper knyttet til en og samme komponent. Det vil ikke da være nødvendig å skape en kobling til noen andre komponenter for å oppnå formålet sitt.

### 6.1.2 Databinding

I Badenymfe applikasjonen benytter vi oss i stor grad av en nyere teknologi som heter databinding. Databinding hjelper oss med å øke ytelsen av programkjøringen og lar oss få til bindingen mellom logikk og UI på en mer oversiktlig måte. Dette i kontrast til den orginale `findViewById`-metoden som befinner seg i Android Studio.

Ved bruk av `findViewById`-metoden i Android Studio for å hente en referanse til et view, må appen traversere igjennom hele viewhierarkiet for og finne viewet som korresponderer med ID'en. Denne traverseringen skjer altså da under kjøring. For et større eller dypere viewhierarki, kan denne traverseringen ta nokså lang tid. Noe som kan få appen til å virke treg for brukeren. (Fujiwara, Galpin, Haecky, McQuillan & Samark, 2019)

Databinding lar oss derimot koble en layout til en activity eller et fragment under kompilering. Kompilatoren genererer da en hjelpeklasse kalt en bindingclass. På denne måten kan vi da aksessere viewene igjennom denne genererte hjelpeklassen under kjøring uten noe ekstra overhead. Denne teknologien vil derimot gå ut over kompileringstiden, men dette er ikke viktig fra en brukers perspektiv og det er absolutt noe vi som utviklere er villige til å ofre. (Fujiwara et al., 2019)

### 6.1.3 Model-ViewViewModel

Applikasjonen Badenymfe er utviklet i henhold til arkitekturen Model-View-ViewModel i den grad det var gjennomførbart. Dette er på bakgrunn av anbefalninger fra Android Developer.

Arkitekturen setter opp applikasjonen i ulike logiske komponenter der hver komponent har én enkel — men sterkt definert — oppgave. Figur 15 illustrerer hvordan MVVM er implementert i Badenymfe.

- `fragment_forecast` er en layout og har som oppgave å definere det grafiske brukergrensesnittet i applikasjonen. Definisjonene er gitt ved bruk av XML.
- `ForecastFragment` er et fragment og har som oppgave å binde data til de forskjellige viewene i UI. Her brukes databinding som er beskrevet i avsnitt 6.1.2. Ingen lagring av data bør skje her, da fragmentet kan ødelegges og bygges på nytt — av operativsystemet — flere ganger i løpet av livssyklusen til programmet.
- `ForecastViewModel` er en view model og har som oppgave å levere data til `ForecastFragment`. Her må view model'en sørge for at data er persistent. Det vil si at dataen skal være tilgjengelig uavhengig av om `ForecastFragment` er dekonstruert. All formatering og manipulering av data bør skje her, slik at fragmenter eller aktiviteter bare trenger å binde dataen til view'ene.
- `ForecastRepository` er et repository og har som oppgave å skaffe data til `ForecastViewModel`. Her ligger det logikk som evaluerer om data finnes i en lokal database eller om dataen må skaffes fra en ekstern nettverkskilde.

Repository'et har alst  to bindinger. En binding mot den lokale databasen og en binding mot et eksternt API.

- ForecastDatabase er en abstrakt definisjon av databasen i applikasjonen. Selve implementasjonen overlater vi til Room. Her spesifiserer vi hvordan vi  nsker   ha databaseskjemaet v rt og hvordan vi  nsker   kommunisere med databasen.
- Met ApiService definerer hvordan vi  nsker   kommunisere med det eksterne API'et. Her velger vi hvilke GET'requests som skal kunne kj res opp mot API'et og hvordan dataen skal parses. Dataen som hentes fra API'et kommer i JSON format og parses til Kotlin dataklasser.

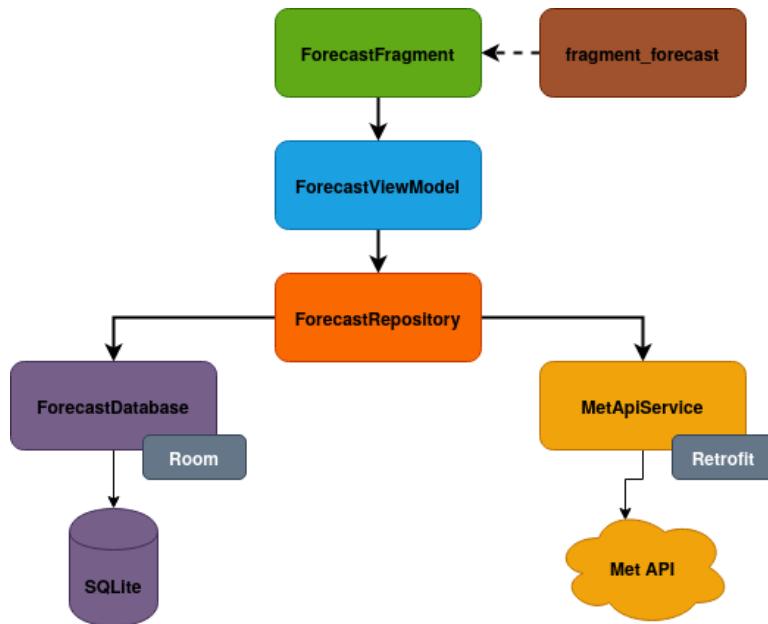
MVVM-arkitekturen l ser en rekke velkjente problemer som man ofte st ter p  under apputvikling. Her er noen av disse l sningene beskrevet.

- **Lav kobling** er tydelig illustrert i figur 15. Her ser vi at det finnes f  koblinger mellom komponentene at koblingene former en rettet asyklisk graf.
- **H y kohesjon** er tydelig beskrevet ovenfor. Her har hver komponent tydelige og konkrete oppgaver. Fragmenter eller aktiviter kan enkelt hente sin data hos en view model, uten   tenke p  om dataen er riktig formatert eller oppdatert. View model'er kan igjen hente sin data fra repository uten   tenke p  om dataen b r hentes fra databasen eller fra API'et.
- **Persistens** kommer tydelig fram i view model'ens beskrivelse, da hovedoppgaven nettopp er det   holde data persistent igjennom hele livssyklusen til applikasjonen.

## 6.2 Periodisk arbeider

All v r- og havdata endres med tid og m  derfor oppdateres. Dataen som hentes fra Meterologisk Institutt har en utl psdato. Denne dato  tilsier at det er gjort nye m linger som gir en bedre prediksjon for hvordan v ret blir i framtiden. Badenymfe-applikasjonen har derfor funksjonalitet som tar seg av slike oppdateringer.

En periodisk arbeider settes i gang hver gang appen starter og stopper n r appen



Figur 15: MVVM

avsluttes. Arbeideren jobber i bakgrunnen slik at brukeren ikke skal forstyrres og sørger for at all data oppdateres — hvis nødvendig — en gang i timen.

Arbeideren fungerer på en slik måte at den sjekker utløpsdatoen på all værdata-en i databasen. Og den oppdaterer kun dataen hvis utløpsdatoen er nådd.

### 6.3 Live data

Når man studerer figur 15, ser man kun bindinger som går nedover i hierarkiet. Man kan da begynne å lure på hvordan for eksempel et fragment eller en aktivitet skal kunne fange opp når data endres eller oppdateres, når det ikke finnes noen direkte eller indirekte bindinger fra view model'en som har dette i oppgave. Her kommer live data inn i bildet. Når dataen i applikasjonen lagres som live data, kan det festes en observatør til dataen som trigges hver gang dataen endres. På denne måten vil fragmentet eller aktiviteten få beskjed uten at view model'en trenger å gi denne beskjeden.

## 6.4 Teknologier

### 6.4.1 Android Plattformen

Badenymfe applikasjonen er utviklet for Android-plattformen. Her er noen punkter som forklarer hva som gjør Android-plattformen unik.

Tradisjonelle desktop-applikasjoner har ofte ett enkelt startpunkt og kjører ofte med en enkel monolitisk kjerne. Androidapplikasjoner er litt annerledes da de ofte lages som flere enkeltkjørende komponenter som kan kjøres uavhengig av hverandre. Hver komponent har sitt eget startpunkt og man kan traversere mellom komponenter med såkalte intents. Samtidig holder operativsystemet Android en oversikt over traverseringen i en såkalt backstack. Backstacken holder på traverseringens historie og lar brukeren traversere tilbake. (*Guide to app architecture*, 2019)

En intent er en form for forespørsel, der en komponent ønsker at en annen komponent skal utføre en oppgave. Eksempelvis kan en mailapplikasjon gi en forespørsel til en nettleserapplikasjon, om å åpne en nettside dersom brukeren trykker på en link. I dette tilfellet er mailklienten og nettleseren helt uavhengige av hverandre. Ulike komponenter innenfor samme applikasjon fungerer også på denne måten. (*Guide to app architecture*, 2019)

### 6.4.2 Android Studio

Android Studio er en IDEA — basert på IntelliJ — skreddersydd for Android utvikling. IDEA'en tilbyr en hel rekke verktøy som skal gjøre utviklingen enklere og mer effektiv. Av disse verktøyene finner man blant annet visuelle editorer for grafisk brukergrensesnitt, emulatorer for testing og live høynivå debugging. Android Studio gir også muligheten til å installere sammfunnsutviklede plugins, slik at editoren skal kunne tilpasses enhver utvikler.

### 6.4.3 Kotlin

Kotlin er et programmeringsspråk til generelle formål utviklet av JetBrains. Språket er utviklet til å inkorporere programmeringsspråket Java og Java Virtuel Maskin. Kotlin er kompatibelt med Java og språkene kan derfor brukes om hverand-

re. I 2017 annonserte Google I/O sin støtte for Kotlin som programmeringsspråk i Android. Det er defor sannsynlig at Kotlin vil ta en større del i Android-utvikling fremover. (*Kotlin (programming language)*, 2020)

#### 6.4.4 API meteorologisk institutt

Vi har hentet data fra to forskjellige API'er via Metrologisk Institutt (MET). API'ene heter "LocationForecast" (*Weather Forecast API*, 2020) og "OceanForecast" (*Ocean Forecast API*, 2020). Ved å bruke disse API'ene har vi hentet data om havforhold og værdata. For å lage strukturen for klassene fra API'ene har vi brukt en plugin fra Android Studio, som heter "JSONtoKotlin" og Moshi for å parse til JSON. Dette resulterer i at vi mottar dataene i form av strings der dataen blir strukturert i relevante klasser som ligger under tilhørende mapper (OceanForecast og WeatherForecast). I behandlingen av dataen som blir hentet, gjør vi den om til forskjellige modeller (database- og domenemodeller). Når vi legger dataene inn i databasen, gjør vi først objektene om til database-entiteter og konverterer typer. Dette fordi databasen kun håndterer et spesifikt format, som f.eks. tidspunkt må være av datatypen long. Når dataene skal vises på skjerm, f.eks. ved at bruker åpner informasjonen om en gitt lokasjon, blir dataene hentet fra databasen og gjort om til domenemodeller før det vises på skjerm. Domenemodeller er en forenklet representasjon — i form av en klasse — av dataen vi ønsker å aksessere.

#### 6.4.5 Room

Værdata hos Meterologisk Institutt oppdateres gjevnlig. Tid for neste oppdatering legges med i responsen fra Meteorologisk Institutt sitt API. Det ville vært uforsvarlig å hente data fra API'et hver gang brukeren ønsker dette, dersom dataen er den samme. All værdata caches derfor i en database for å unngå unødvidig stress på Meteorologisk Institutt sine servere. I tillegg gir dette brukeren muligheten til å sjekke værdata som allerede er cachet selv om brukeren ikke har noe nettverksforbindelse.

I Badenymfe applikasjonen benytter vi oss av en SQLite database for denne cachingen. Videre benytter vi oss av Room Persistence Library som gir oss et abstraksjonslag over databasen, slik at det skal bli lettere å kommunisere med

databasen via Kotlin. Room lar oss blant annet implementere spørninger som funksjonskall og entiteter som dataklasser. (*Room Persistence Library*, 2020)

#### 6.4.6 Retrofit

Meteorologisk Institutt sitt API gir værdata basert på HTTP-forespørsler. Badenymfe applikasjonen trenger derfor en HTTP-klient som kan koble til API'et og sende disse forespørslene. Vi valgte å bruke Retrofit sin HTTP-klient til nettopp dette da det regnes som industri standard. (Lim, 2019)

Retrofit er en typesikker HTTP-klient for Android utviklet av Square. Retrofit gjør det enkelt å hente JSON eller XML data som kan parses til Kotlin-objekter. (Pun, 2017)

#### 6.4.7 Google Maps

For å legge til kartfunksjonalitet i appen har vi brukt Google Maps Software Development Kit (SDK). Denne lar oss bruke Google Maps API'et til å håndtere brukerinteraksjon med kartet, hente adresser med mer. Google anbefaler å bruke nettopp Android Studio dersom en skal lage en app med Google Maps SDK. For å få tilgang til Google Maps sine servere, trengs en API-nøkkel. Denne kan genereres via en lenke som automatisk dukker opp i `google_maps_api.xml`, en fil som genereres av Android Studio dersom man oppretter en Google Maps Activity. Biblioteket som brukes i denne sammenheng er tilbuddt via Google Play Services, og deres SDK må dermed settes opp i prosjektet for at kartfunksjonaliteten skal fungere. (*Add maps*, 2019; *Get an API Key*, 2020; *Maps SDK for Android Get Started*, 2020; *Maps SDK for Android Overview*, 2020)

Klassen `GoogleMap` håndterer automatisk tilkobling til Google Maps' servere, laster ned og viser kartet på skjermen, samt at det viser og responderer på diverse brukerinteraksjon. I tillegg kan metoder med ekstra funksjonalitet implementeres. Dette gjelder blant annet å legge til markør på kartet ved klikk på en posisjon, zoom mot brukerens posisjon ved oppstart, og å hente adressen til et punkt markert av bruker. Det kan også spesifiseres hvilken type kart som skal vises, for eksempel som satellittbilde, vanlig typisk veikart, hybrid eller terreng med topografisk data. Vi har valgt å ta i bruk det "vanlige" kartet, dvs. det Google

kaller “typisk veikart”. (*Maps SDK for Android Overview*, 2020; *Map Objects*, 2020; *Add maps*, 2019)

## 7 Testdokumentasjon

Målet vårt når det gjelder testing har vært å begynne testingen tidlig og kontinuerlig teste underveis i utviklingsløpet for å avdekke feil og mangler både ved statisk og dynamisk testing. Statisk testing har blitt gjennomført ved parprogrammering og hjelp fra Android Studio som f. eks gir melding om ubrukte variabler. Dynamisk testing har gitt oss verdifull tilbakemelding når vi har kjørt koden, som f. eks at applikasjonen ikke har oppført seg slik vi ønsket da vi skrev koden. Ved å implementere testaktiviteter kontinuerlig i prosessen og tidlig har vi derfor både validert og verifisert at koden gjør det vi ønsker etter å ha laget funksjonalitet og delfunksjonalitet basert på brukerundersøkelser og krav. Ved å teste tidlig slipper vi også overraskelser helt til slutt, i tillegg til at vi ser på testing som en essensiell del gjennom hele utviklingsløpet. Siden det er umulig å teste alt har vi hatt en proaktiv og analytisk tilnærming til testing for å finne ut av hva vi ønsker å teste. For å hjelpe oss med å finne ut av dette har vi tatt i bruk trusselpoker som verktøy.

### 7.1 Trusselpoker

Vi gjennomførte trusselpoker over Zoom (se figur 16) for å kartlegge de forskjellige aspektene og funksjonene vi anså som mest kritiske ved appen, for så å teste disse. Gjennomidémyldring og analyse av “brukerreisen” brukerne gjennomgår ved å bruke Badenymfe kom vi opp med forskjellige scenarioer til trusselpokeren.



Figur 16: Trusselpoker på Zoom

Vi definerte en tallskala med tall fra 0 til 10, hvor 0 var lavest og 10 høyest som vi skulle bruke til å stemme på scenariene. For hvert scenario talte vi til 3 og hver og en av oss viste et tall i form av et kort fra kortstokk, antall fingre, eller ark slik at alle kunne se tallverdien gjennom kameraet. Vi stemte på hvor enkelt vi anså at hvert scenario var å løse, og hvor kritisk vi anså scenariet for å være. Det var ikke alltid lett å definere dette fordi vi ikke kunne vite hvor omfattende noe ville være å løse, eller hvor kritisk noe er, som f. eks fra et brukerperspektiv, at en repeterende feil kan være veldig irriterende for noen brukere og kanskje uviktig for andre. For de scenarioene hvor risikovurderingen vår var variert tok vi en ny diskusjon og stemte på nytt. Som vist i tabellene under ser vi de forskjellige scenariene og resultatet fra avstemning. For noen av scenariene var det avstemning basert på om scenariet var sannsynlig at forekom istedenfor hvor enkelt scenariet er å løse og hvor kritisk det er.

Basert på resultatene fra trusselpoker, ser vi at de scenarioene som viste seg å være mest kritiske er: *at applikasjonen viser data fra feil posisjon i forhold til den posisjonen bruker ønsker, eller at dataene som vises er utdatert. I tillegg anser vi det som kritisk om applikasjonen krasjer. Det tredje og siste scenariet vi anser som kritisk er om værdata ikke dukker opp i det hele tatt når brukeren forventer det.*

Tabell 1 viser en oversikt over alle scenarioene, hva vi stemte, og evt. omstemming.

Tabell 2 viser en oversikt over de scenariene hvor vi stemte over hvor sannsynlig det er at de forekommer.

Ettersom vi fant de mest kritiske aspektene ved applikasjonen var planen

Tabell 1: Avstemning rundt scenarioene

Scenario	Avstemning: hvor enkelt er scenariet å løse?	Avstemning: hvor kritisk er dette scenariet?	Omstemming: hvor enkelt er scenariet å løse?	Omstemming: hvor kritisk er dette scenariet?
Data fra feil posisjon/ utdatert data	2, 5, 2, 4, 5, 5	5, 5, 9, 5, 4, 7	2, 2, 2, 2, 2, 1	8, 8, 9, 4, 8, 8
Appen krasjer	5, 5, 5, 7, 6, 6	10, 10, 10, 8, 6, 7	Ingen om- stemming	8, 9, 9, 10, 9, 9
Værinfo dukker ikke opp	5, 4, 4, 6, 5, 6	7, 7, 9, 6, 8, 7	5, 4, 4, 4, 6, 6	7, 7, 7, 7, 7, 7
Data forsvinner fra brukerens enhet	3, 7, 5, 6, 3, 5	2, 2, 5, 5, 6, 7	5, 5, 6, 4, 3, 5	4, 4, 4, 4, 3, 4

Tabell 2: Avstemning rundt sannsynlighet

Scenario	Avstemning: Hvor sannsynlig er det at scenarioet skjer?	Omstemming: Hvor sannsynlig er det at scenarioet skjer?
Kan posisjonsdata misbrukes av eksterne aktører	1, 5, 2, 5, 1, 3	4, 3, 2, 4, 1, 2
Stemmer sannsynlig- hetsvurderingen av brennmaneter med virkeligheten	7, 5, 7, 7, 7, 7	6, 5, 7, 5, 7, 7

videre å teste disse scenarioene for å minimere risikoen for at de forekommer.  
Fremgangsmåten vår for å teste de forskjellige scenarioene vises i tabell 3.

## 7.2 Testing av ikke-funksjonelle krav

Som vi nevner i *Kravspesifikasjon og modellering* har vi implementert en rekke ikke-funksjonelle krav. Vi har ønsket å teste alle disse kravene, men det er variert spenn når det gjelder hvor mange sekunder det tar å vise resultater på

Tabell 3: Testplan

Scenario	Testplan
Data for feil posisjon/utdatert data vises	Sammenligne API i nettleser for gitt posisjon
Applikasjonen krasjer	Kjøre appen på forskjellige OS på forskjellig og samme tid
Ingen værdata vises når det forventes	Teste forskjellige lokasjoner, lukke appen og prøve igjen, lagre som favoritt og vis på ny

en emulator til en annen. Derfor har dette kravet ikke latt seg teste med et spesifikt antall sekunder. Vi har gjennomgått en rekke tester og enhetstester for andre krav manuelt og ved å definere en rekke testmetoder i testmappen “.com.example.badenymfe.com (test)”. Dette er tester vi har fokusert på i tråd med resonnementet over: i kontekst av grunnfunksjonalitet som er kritisk for applikasjonen og brukeropplevelsen, og som utarbeidet av teamet gjennom trusselpoker. Vi har flere ganger testet om applikasjonen krasjer ved å observere applikasjonens oppførsel etter forskjellige valg og navigasjonsmønstre. Et annet eksempel på testing av ikke-funksjonelle krav er at vi har sjekket forskjellige ganger at både portrett- og landskapsmodus fungerer til forskjellige tider og etter varierte navigasjonsvalg. Et eksempel på enhetstesting er ved å bruke assert funksjon kan vi ved å manuelt lese fra API’et før vi kjører applikasjonen, skrive inn forventet resultat og se om testen passerer uten feil eller ikke. Dette er noe vi har gjort underveis i kodingen for å teste at programmet gjør som forventet, og evt. feilsøke og endre om det ikke gjør som forventet. Under følger mer detaljert informasjon rundt testingen vi har gjennomført i Android Studio.

### 7.3 Instrumentelle tester

I mappen “.com.example.badenymfe.userinterface(androidTest)” har vi implementert en rekke instrumentelle tester som kjører på emulatoren i Android studio. Testene er en form for intergrasjons- og brukergrensesnitt-tester. Disse har som formål å sjekke at brukergrensesnitt og det funksjonelle i appen fungerer som det skal. Testene simulerer interaksjon med brukeren, og sjekker at resultatet etter visse handlinger i appen er som forventet.

I filen “AddLocationsTest.kt” simuleres en rekke handlinger relatert til å lagre

en posisjon i favoritter og sjekke værdata på disse stedene. Testene kjøres med AndroidJUnit4, med versjonen “`androidx.text.ext.junit.runners.AndroidJUnit4`”. Dette fordi `AndroidJUnit4` er utdatert, og Android da anbefaler å kjøre med den nevnte versjonen (*AndroidJUnit4*, 2019). Testene bruker også `androidx.test.rule.ActivityTestRule`, med `ActivityTestRule` satt til `MapsActivity`-klassen. Dette fordi testene skal starte i kartet.

I noen tilfeller ønsket vi å teste at appen oppfører seg som den skal dersom bruker trykker på en gyldig/ikke gyldig posisjon, det vil si hvorvidt data eller feilmelding om manglende data dukker opp. For å kunne spesifisere hvilke koordinater det skulle trykkes på, måtte vi ha tilgang til den private metoden “`placeMarkerOnMap`” i `MapsActivity`, og spesifisere input-koordinatene til metoden. For å få tilgang til denne, brukte vi metoden “`getDeclaredMethod()`”, satt “`isAccessible`” til true og kalte “`method.invoke()`” for å kalle på metoden. For å få dette til å fungere, måtte vi også bruke “`androidx.core.app.ActivityScenario`” i metoden.

Utover dette ble “`androidx.test.espresso.*`” brukt til å simulere klikk fra bruker, samt til å sjekke at resultatet i testen ble som forventet med `.check(matches(...))`.

Testene som utføres i “`AddLocationsTest.kt`” sjekker at:

- Plussknappen i kartet dukker opp dersom bruker markerer et vilkårlig punkt på kartet
- Appen navigerer til “Mine favoritter” dersom bruker legger til en lokasjon
- Bruker får opp data i “Time for time” dersom posisjonen som er lagt til er gyldig (i vannet, et sted API’et har data)
- Bruker får opp melding om at data ikke finnes i viewet “Time for time” dersom stedet lagt til er ugyldig
- Melding om manglende data ikke dukker opp dersom stedet er gyldig
- Dersom bruker legger til et gyldig sted og navigerer til “Værmelding” så vises værmeldingen
- Korrekte koordinater vises i “Mine favoritter” dersom bruker legger til en gitt posisjon
- Dersom bruker sletter en lokasjon så forsvinner den fra “Mine favoritter”

- Dersom bruker trykker på slett for så å trykke på “HELL NO!” så forblir lokasjonen i “Mine favoritter”.

For at testene skal fungere som forventet må vi også kunne kontrollere hvilke lokasjoner som er lagret i appen. Løsningen på dette var å bruke “androidx.test.core.app.ApplicationProvider.getApplicationContext” for å få tilgang til å slette databasen før og etter kjøring. Dette er ikke en optimal løsning, og hvis vi hadde hatt mer tid ville vi endret det slik at det lages en testdatabase istedenfor. Dette rakk vi dessverre ikke å gjennomføre.

En annen suboptimal løsning har med treghet i appen å gjøre. For å sikre at klick fra bruker ikke ble utført før viewet var klart, var vi nødt til å forsinke interaksjonen med appen. Måten vi løste dette på var å bruke Thread.sleep(). Det finnes helt sikkert bedre løsninger på utfordringen, men også dette ble nedprioritert grunnet tidsmangel.

I filen “FragmentsTest” er det skrevet en rekke tester som åpner et fragment i appen og sjekker at bestanddelene vi forventer å ha i viewet er på plass. Denne klassen kjører med AndroidJUnit4, og bruker “androidx.fragment.app.testing.launchFragmentInContainer” for å åpne fragmentene, og “androidx.test.espresso.\*” for å sjekke at viewet er som forventet.

#### 7.4 Test av database

I filen “ForecastDatabaseTest” tester vi at databasen fungerer som den skal. Testklassen kjører med AndroidJUnit4, og bruker “org.junit.\*” for å definere hva som skal skje før, under og etter testen, samt for å sjekke hvorvidt resultatet ble som forventet. Før testen skal kjøre oppretter vi en database, og til dette kreves kontekst. Til dette brukes “androidx.test.platform.app.InstrumentationRegistry” sin metode getInstrumentation().targetContext.

Databasen lagres kun midlertidig i minnet, og forsvinner etter testen. For å oppnå dette bruker vi “androidx.room.Room” sin metode “inMemoryDatabaseBuilder(...)”. I tillegg spesifiserer vi at databasen skal kunne kjøre på Main Thread, da Android ellers ikke ville latt spørninger kjøre der. Siden vi skal legge data inn i databasen under testen, og dette skjer i en coroutine, måtte vi også ta i bruk “kotlinx.coroutines”.

## 7.5 Enhetstester

I mappen “.com.example.badenymfe.userinterface(test).domain” ligger enhets-tester av algoritmen for sannsynlighet for brennmaneter i filen OceanForecastTest. Her oppretter vi objekter av klassen OceanForecast med ulike input-verdier, og sjekker at beregningen av jellyfishProb blir som forventet. Testverktøyene vi bruker er “org.junit.Test” og “junit.framework.Assert.assertEquals”.

## 8 Refleksjon

Vi kan alle være enige om at vårsemesteret 2020 ikke ble som forventet. Allerede i startgropen av prosjektet kom SARS-CoV-2, populært kalt Korona-viruset (eller Corona med C om du vil), og stengte ned Norge. Tolvte mars møtte vi opp for gruppemøte på IFI, da Blindern som oppholdssted ikke enda var stengt. Dette skulle imidlertid vise seg å bli vårt siste fysiske møte, da vi halvveis ble bedt om å forlate bygget og dra hjem. Smånervøse med en surrealistisk følelse i kroppen dro vi hvert til vårt, og måtte finne nye måter å utføre gruppearbeid på.

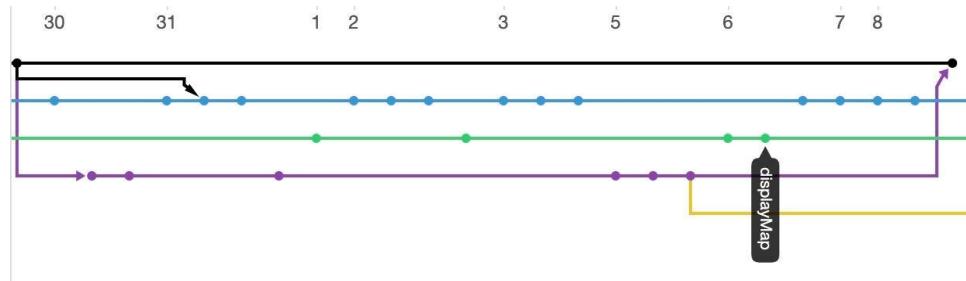
Tilfeldigvis hadde vi i arbeidsavtalen (se vedlegg A) uttrykt at ved fysisk fravær kunne deltagelse skje via Skype. Når vi alle så, etter kommando fra Erna, ble nødt til å ha fysisk fravær, var det nettopp videomøter som skulle sikre prosjektets gang. Riktig nok ikke via Skype, men heller via Zoom. Møtene fortsatte med samme struktur, og med delt skjerm som erstatning for skjermene på IFI sine grupperom.

### 8.1 Samarbeid, kommunikasjon og Korona

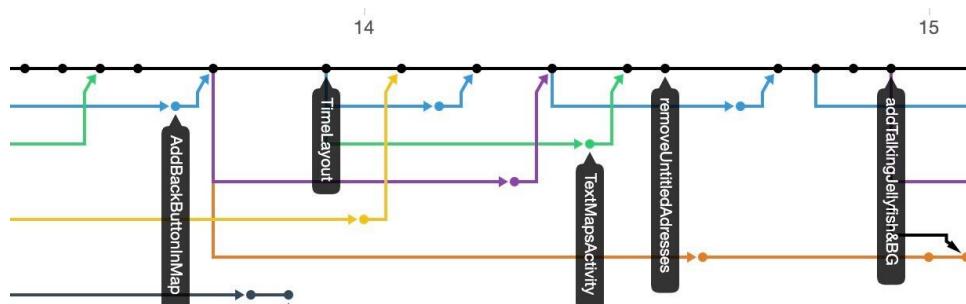
Koronasituasjonen har påvirket samarbeidet og gruppodynamikken gjennom semesteret. Manglende tilfeldige møter fjernet muligheten til å spontant ta opp problemer eller stille spørsmål til andres kode. Vanligvis ville vi kunnet løst problemer raskt og mens de fortsatt var små, selv uten å planlegge det. Når all kommunikasjon nå måtte være planlagt og initiert via en Zoom-avtale, tenderte vi å la problemene vokse seg større før det ble tatt initiativ til å løse dem. Dette kan ha gjort utviklingsprosessen tregere, og forståelsen av andres kode lavere.

Mot slutten av prosjektet viste GitHub seg å være viktig for gruppens samarbeid,

og kommunikasjon på Slack ble viktig for unngå merge-conflicts. I startfasen av prosjektet var de fleste av medlemmene lite komfortable med GitHub, og merging av brancher ble utsatt til vi ikke lenger hadde noe valg (figur 17). Heldige som vi var hadde et av teammedlemmene en samboer som viste seg å være et GitHub-orakel, og dermed var det ikke lenger noen unnskyldning for å ikke lære seg merging og alt som hørte med. For å hjelpe alle i gang med Git, arrangerte vi felles Zoom-møter hvor alle fikk merget brancher sammen. Etterhvert som vi fikk prøvd oss på mer merging, var det til slutt ingen sak å ta i bruk GitHub, og det viste seg å bli et verktøy med stor nytte (figur 18). Når masterbranchen endelig bestod av en prototype og alle hadde blitt komfortable med å merge, gikk både kommunikasjon og utvikling mye bedre. En erfaring vi tar med oss videre til fremtidige prosjekter er dermed viktigheten av å avklare forventninger til kommunikasjon og å legge en strategi for hvordan jobbe sammen via GitHub. Når alle i gruppen var komfortable med å lage nye brancher og merge, ble det lettere å finne mindre oppgaver og merge disse raskt. Det ga også økt motivasjon i gruppen, da alle kunne se hvilke stadige forbedringer som skjedde i appen.



Figur 17: GitHub i starten av prosjektet



Figur 18: GitHub mot slutten av prosjektet

En annen utfordring var balansen mellom å spille på hverandres styrke og å gi

alle eierskap til produktet. Noen var naturligvis mer motivert for programmering enn andre, og tok mer ansvar for både arkitektur og implementering. På samme måte var andre mer engasjert i rapport-skrivering og tok mer styring der. Dette måtte kanskje til for å bli ferdig med prosjektet i tide, men med tanke på programmeringen gjorde det at det ble vanskelig for andre å forstå arkitekturen og koden bak produktet. Som følge av dette ble det vanskeligere for andre å skjonne hvordan man kunne få de ulike klassene til å kommunisere på en gitt måte når man skulle legge til funksjonalitet i innspurten av apputviklingen. Kanskje hadde dette også vært lettere dersom vi kunne møttes mer fysisk, og dermed avklare ting raskere og i person. Med Zoom som kommunikasjonsmiddel var det alltid en høyere terskel for å spørre.

## **8.2 Digital arbeidshverdag, motivasjon og .... Korona**

En uventet utfordring ved digitale gruppemøter ble ordstyring. Når vi ikke satt i samme rom og dermed ikke var like oppmerksomme på hverandres kroppsspråk, var det fort gjort å snakke i munnen på hverandre. Mye subtil kommunikasjon foregår uten ord, noe som ble vanskeligere å oppfatte på en videosamtale. Dette gikk seg imidlertid til med litt øving, og vi praktiserte håndsoprekning om nødvendig.

I starten føltes videomøter som en god erstatning for sosialt samvær. Det lettet ensomheten ved å være innestengt i egen leilighet. Etter rundt en uke endret dette seg. Det var tydelig at fysisk samvær tilfører noe mer enn digitale møter. Dette påvirket motivasjonen til flere i gruppen, da vi ble fratatt muligheten til å dekke et grunnleggende menneskelig behov.

En fordel med å ta IN2000 var at det ga struktur i en ellers strukturløs hverdag. Tirsdager og torsdager hadde gruppen en avtale de måtte holde, i tillegg til digital parprogrammering. Dette ga en viss følelse av normalitet, og hjalp oss med å opprettholde noen rutiner i hverdagen. Videomøter og avtaler om parprogrammering sikret dermed prosjektets fremgang.

En stor andel av programmeringen foregikk i par. Kanskje hadde vi ikke tatt dette i bruk i like stor grad dersom vi hadde vært i en normalsituasjon. Men situasjonen påførte oss, som tidligere nevnt, generelt lav motivasjon til å gjennomføre arbeid. Avtaler med andre førte til at vi klarte å sette oss ned med prosjektet, og gjennomføre arbeidet. Selv om starten av møtene ofte bar preg av

lav lyst til å programmere, fikk vi gjort mye bare vi kom i gang. Parprogrammering har rett og slett vært helt nødvendig for flere i gruppen. Måten vi utførte arbeidet på var at en hadde Android Studio åpent og programmerte med delt skjerm, mens den andre Googlet underveis.

En ulempe med parprogrammering, var av teknisk art. Med både Android Studio og Zoom kjørende samtidig, ble datamaskinen noe overbelastet. Alt gikk tregt. Fra vi satte i gang kjøringen av programmet og til applikasjonen var klar i emulatoren, tok det gjerne 4-5 minutter. I tillegg ble maskinen til den som hadde Android Studio åpent ubruklig til det meste av annet arbeid. For hver gang vi hadde endret noe i koden, gjerne bare et par linjer, måtte vi vente 4-5 minutter før vi fikk testet det. En kan spekulere i hvor mye kortere tid arbeidet hadde tatt dersom vi hadde sluppet all denne ventingen.

Teambuilding ble også en utfordring. Like vel forsøkte vi å gjøre det beste ut av situasjonen og arrangere alternative teambuilding-aktiviteter. En kveld spilte vi Cards Against Humanity og Hang-man på Zoom. En annen kveld så vi film sammen via NetFlix party. Selv om disse løsningene ikke var det samme som å møtes i person, var det tross alt det beste vi kunne få til for en sosialkveld ala Korona.

### **8.3 Store ambisjoner, begrenset tid og enda mer begrenset kompetanse**

En åpenbar utfordring ved dette prosjektet, var å skulle utvikle en velfungerende app i løpet av få måneder i et språk og med en teknologi ingen hadde særlig kjennskap til fra før. Ambisjonene var høye og planene mange, men implementeringen viste seg å være tidkrevende. Å sette seg ordentlig inn i kommunikasjonen mellom front-end og back-end, MVVM-arkitekturen og konseptene i android-utvikling var tidkrevende i seg selv. Vi ønsket alle at vi hadde hatt mer kunnskap om dette før apputviklingen startet, slik at ikke all læring måtte foregå parallelt med programmeringen.

Apputvikling for Android er en stor verden å navigere seg i, og det å finne frem til de rette kildene var ikke alltid like enkelt. De fleste søkeresultatene var skrevet i Java, biblioteker var utdaterte, eller verktøyene hadde begynt å koste penger. Vi fulgte ved flere tilfeller en oppskrift på noe, for så å oppdage at den ikke fungerte.

Vi byttet så gjerne ut det første med en annen oppskrift, før det viste seg at heller ikke denne fungerte. Vi erfarte etterhvert at det var best å ta utgangspunkt i en oppskrift, forså å bytte ut enkellementer i koden for å få alt til å fungere. Med tiden ble også evnen til å finne de riktige løsningene og kompetansen til å tenke selv forbedret.

Vi rakk aldri å realisere alle våre idéer, men sitter igjen med en applikasjon som oppfyller hovedpunktene av det vi ønsket oss. Vi har lært mye på veien, både om grupperarbeid, apputvikling og samarbeid over GitHub. I tillegg har vi erfart hvordan en digital arbeidshverdag kan gjøres best mulig, og føler at vi har mestret oppgaven godt til tross for omstendighetene.

## Referanser

- Add maps.* (2019). Hentet 26-05-2020 fra <https://developer.android.com/training/maps>
- Androidjunit4.* (2019). Hentet 26-05-2020 fra <https://developer.android.com/reference/androidx/test/runner/AndroidJUnit4>
- Android studio.* (u.d.). Hentet 24-05-2020 fra <https://developer.android.com/studio>
- Black, R., Van Veenendaal, E. & Graham, D. (2015). *Foundations of software testing: Istqb certification.* Hampshire: Cengage Learning.
- Contrast ratio.* (2020a). Hentet 22-03-2020 fra <https://contrast-ratio.com/#white-on-%23A7E2E3>
- Contrast ratio.* (2020b). Hentet 22-03-2020 fra <https://contrast-ratio.com/#black-on-%232b46c7>
- Er badevannet ditt fullt av brennmaneter?* (2019). Hentet 17-03-2020 fra <https://forskning.no/havforskning-havforskningsinstituttet-partner/er-badevatnet-ditt-fullt-av-brennmaneter/1353020>
- Fosette, S., Gleiss, A., Karpytchev, M. & Hays, G. (2015). Current-oriented swimming by jellyfish and its role in bloom maintenance. *Current Biology*, 25, 342-347.
- Fujiwara, L., Galpin, D., Haecky, A., McQuillan, S. & Samark, A. (2019). *Developing android apps with kotlin by google.* Hentet 20-05-2020 fra <https://www.udacity.com/course/developing-android-apps-with-kotlin--ud9012>
- Get an api key.* (2020). Hentet 26-05-2020 fra <https://developers.google.com/maps/documentation/android-sdk/get-api-key>
- Guide to app architecture.* (2019). Hentet 20-05-2020 fra <https://developer.android.com/jetpack/docs/guide>
- Hva er personas og hvordan identifiserer du dem?* (2019). Hentet 14-04-2020 fra <https://www.awidlyagency.com/no/growthhub/hva-er-personas>
- Hva er personas og hvorfor bør man ha det?* (2018). Hentet 14-04-2020 fra <https://blogg.optimalnorge.no/hva-er-personas-og-hva-er-det-godt-for>
- Kontrastforhold mellom tekst og bakgrunn.* (2020). Hentet 22-05-2020 fra [https://uu.difi.no/krav-og-regelverk/wcag-20-standeren/143-kontrast-minimum-niva-aa](https://uu.difi.no/krav-og-regelverk/wcag-20-standarden/143-kontrast-minimum-niva-aa)

- Kontrast for ikke-tekstlig innhold.* (2020). Hentet 22-05-2020 fra <https://uu.difi.no/krav-og-regelverk/webdirektivet-og-wcag-21/wcag-21-standarden/1411-kontrast-ikke-tekstlig-innhold-niva-aa>
- Kotlin (programming language).* (2020). Hentet 24-05-2020 fra [https://en.wikipedia.org/wiki/Kotlin\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Kotlin_(programming_language))
- Lim, I. (2019). *Advanced networking: when retrofit is now enough.* Hentet 26-05-2020 fra <https://proandroiddev.com/advanced-networking-when-retrofit-is-not-enough-316a897ddf40>
- Map objects.* (2020). Hentet 26-05-2020 fra <https://developers.google.com/maps/documentation/android-sdk/map>
- Maps sdk for android get started.* (2020). Hentet 26-05-2020 fra <https://developers.google.com/maps/documentation/android-sdk/start>
- Maps sdk for android overview.* (2020). Hentet 26-05-2020 fra <https://developers.google.com/maps/documentation/android-sdk/intro>
- Nielsen, A., Pedersen, A. & Riisgård, H.U. (1997). Implications of density driven currents for interaction between jellyfish (*aurelia aurita*) and zooplankton in a danish fjord. *Sarsia*, 82, 297-305.
- Ocean forecast api.* (2020). Hentet 26-05-2020 fra <https://in2000-apiproxy.ifi.uio.no/weatherapi/oceanforecast/0.9/documentation>
- Pekeravbrytelser.* (2020). Hentet 22-05-2020 fra <https://uu.difi.no/krav-og-regelverk/webdirektivet-og-wcag-21/wcag-21-standarden/252-pekeravbrytelse-niva>
- Preece, J., Sharp, H. & Yvonne, R. (2015). *Interaction design beyond human-computer interaction.* West Sussex: Wiley.
- Pun, P. (2017). *Retrofit - a simple android tutorial.* Hentet 26-05-2020 fra [https://medium.com/@prakash\\_pun/retrofit-a-simple-android-tutorial-48437e4e5a23](https://medium.com/@prakash_pun/retrofit-a-simple-android-tutorial-48437e4e5a23)
- Room persistence library.* (2020). Hentet 26-05-2020 fra <https://developer.android.com/topic/libraries/architecture/room>
- Sommerville, I. (2020). *Engineering software products, an introduction to modern software engineering.* Hoboken: Pearson.
- Test management.* (2020). Hentet 20-03-2020 fra <https://www.uio.no/studier/emner/matlnat/ifi/IN3240/v20/forelesningsvideoer/chapter-5-part-1-20slides.pdf>
- Visningsretning.* (2020). Hentet 22-05-2020 fra <https://uu.difi.no/krav-og-regelverk/webdirektivet-og-wcag-21/wcag-21-standarden/>

134-visningsretning-niva-aa

*Wcag 2.1-standarden.* (2018). Hentet 22-03-2020 fra <https://uu.difi.no/krav-og-regelverk/webdirektivet-og-wcag-21/wcag-21-standarden>

*Weather forecast api.* (2020). Hentet 26-05-2020 fra <https://in2000-apiproxy.ifi.uio.no/weatherapi/locationforecast/1.9/documentation>

*Web content accessibility guidelines (wcag) 2.1.* (2018). Hentet 22-03-2020 fra <https://www.w3.org/TR/WCAG21/#new-features-in-wcag-2-1>

## **A Teamavtale**

### **A.1 Tilstedeværelse**

Det forventes at alle møter til avtalte tidspunkter. Ved forsinkelser/fravær gis det beskjed i god tid. Ved tre forsentkomminger må vedkommende gjøre noe trivselsrelatert for gruppen. Ved fysisk fravær kan deltakelse være via Skype. Dersom dette ikke lar seg gjøre skal den fraværende lese seg opp på møtereferat i ettertid. Avgjørelser tas av de som er tilstede.

### **A.2 Tidsbruk**

Planlagt to møter á 2t per uke fast. Tidsbruk vurderes fra uke til uke etter behov.

### **A.3 Forventninger**

Det forventes at alle gruppemedlemmer deltar og engasjerer seg i alle prosjekts fasér. Alle gjør tildelte oppgaver etter beste evne, og sier ifra dersom det oppstår problemer, tidsklemme eller andre behov for hjelp. Vi blir enige om hvordan vi utfører arbeid, og uenigheter legges til side dersom de ikke er relevante for gjennomføring/funksjonalitet. Alle skal være snille med hverandre, og vise respekt for ulike metoder.

### **A.4 Konflikthåndtering**

Ved uenigheter skal det tilstrebes å løse konflikten mellom de involverte partene. Om konflikter ikke kan løses mellom disse, skal Scrum master innvolveres.

## B Samtykkeerklæring

Vi er seks studenter i kurset *IN2000 - Software Engineering med prosjektarbeid* ved Institutt for Informatikk, Universitetet i Oslo. Vi gjennomfører dette semestert et prosjektarbeid hvor vi skal utvikle en app basert på et case med data fra Meteorologisk Institutt. Vi jobber med et case om Bevegelser i havet, spesifikt angående badetemperaturer og forhold.

Vi ønsker gjennom datainnsamlinger (intervjuer, observasjoner osv.) å avdekke behov innen det gitte området, og utvikle en app som kan bidra til å utfylle disse. Det vil også gjennomføres testing av appen underveis i prosjektet. Intervjuer og evalueringer vil transkriberes og analyseres av oss, og rapportering fra dette vil kun være tilgjengelig i sin helhet for gruppemedlemmene. Utdrag fra disse aktivitetene vil kunne brukes i en avsluttende rapport, men vil i tilfelle anonymiseres og ikke kunne tilbakeføres til den enkelte deltaker.

### *Frivillig deltakelse*

Vi vil understreke at deltakelse er frivillig, og at du kan trekke deg fra undersøkelsen når som helst. Vi kan komme til å ta lydopptak, notater, bilder o.l. under datainnsamlingene. Dette informeres om før evt. opptak startes. Bilder vil bli anonymisert i etterkant, og evt. lydopptak slettes senest 30. juni 2020. Du kan når som helst avslutte datainnsamlingen og/eller trekke tilbake informasjon som er gitt. Ved behov eller spørsmål kan du kontakte Johannes Skøien på mail [johasko@uio.no](mailto:johasko@uio.no).

### *Anonymitet*

Bilder, notater og transkribering vil bli anonymisert; ingen andre enn oss vil vite hvem som har deltatt, og det som sies under intervjuer eller observasjoner vil ikke kunne tilbakeføres til dere.

Før prosjektet begynner, ber vi deg om å samtykke i deltagelsen ved å undertegne på at du har lest og forstått informasjonen i dette skrivet, og ønsker å delta i prosjektet.

Med vennlig hilsen,

Lars Erik Wik Maren Zon Rafdal Vilde Paschen Knudsen Ellen Marie Andersen  
Kristin Aurora Sydhagen Johannes Skøien

### *Samtykke*

Jeg har lest og forstått informasjonen over og gir mitt samtykke til deltakelse i prosjektet

## C Innledende brukerundersøkelse

### C.1 Bruker

- Kjønn
- Alder
- Lite/noe/mye erfaring med teknologi?
- Student/i arbeid

*Semi-strukturert intervju vil si noen forberedte spørsmål for å lede intervjuet i "riktig" retning, men deltaker står fritt til å trekke inn andre emner og styre intervjuet til en viss grad. Fokus er å få så utfyllende svar på forberedte spørsmål som mulig.*

*Før oppstart: Informer hvordan intervjuet dokumenteres (notater, bilder, lydopp-tak osv.). Påse at deltaker samtykker til deltagelse.*

### C.2 Introduksjon

Forklare kort:

- Utgangspunkt for oppgave/case vi har valgt
- Allerede bestemte/foreslatt "nødvendige" funksjoner
  - Kart
  - Temperatur
- Grunnleggende data vi har tilgjengelig fra MTI
  - Sjøtemperatur
  - Sjø strømning (m/s)
  - Sjø strømningsretning
  - Bølgehøyde
  - Bølgeretning
  - Sjødybde

- Prosent is i sjøen
- Generell værdata:
  - \* vind (m/s, retning)
  - \* skyer
  - \* trykk
  - \* regn
  - \* temperatur
  - \* luftfuktighet

*Viktig å presisere at ørlige svar er viktige, det finnes ingen riktige eller gale svar. Bedre med for mye informasjon enn for lite. Ikke begrense mulighetene til fantasi i svar og forslag, dette kan heller begrenses i senere analyse.*

### C.3 Hoveddel

Hvilken informasjon er fornuftig å ha tilgjengelig?

- Hva er formålmessig ut over grunnfunksjonene nevnt i innledningen?
- Hvor detaljert bør informasjonen som presenteres være?
- Stedstjenester, finne egen posisjon når appen åpnes?
- Bør det være mulig å lagre favorittsteder? Meny med populære badeområder i et gitt område?
- Hva ønsker bruker å se? (Visuelt design)
- Hvordan skal informasjonen vises? (Tegne forslag/vise på screenshot?)
- Bør kart være zoomet inn på en posisjon ved oppstart?

*Prøv å få så detaljerte svar som mulig, uten å “presse” ut noe av deltaker. Forhold deg nøytral, slik at ikke svarene formes av din adferd og reaksjon. Unngå å respondere med “ja”, “nei”, “riktig” osv, dette kan gi deltaker inntrykk av at det finnes et riktig svar. Vær forsiktig med tonefall, for å unngå at spørsmål og respons virker “dømmende”.*

#### **C.4 Avslutning**

Noe som burde tilføyes? Andre idéer som ikke er nevnt? Stans evt. lydopptak. *Om lyd blir tatt opp, fortell vedkommende som intervjuer når opptaket avsluttet, slik at det ikke er noen tvil om dette. Erfaring har vist at lydopptaker kan "skremme" deltakere, og at det kommer tilleggsinformasjon etter denne er slått av. Hold derfor muligheten til å notere videre åpen, men unngå at det fremdeles fremstår som en intervjuetting.*

## D Brukertest

### D.1 Mål for brukertesten

Brukervennlighet, relevans av data, oppsett, visuell utforming, forbedringspotensialer

### D.2 Deltaker

- Kjønn
- Alder
- Lite/noe/mye erfaring med teknologi?
- Student/i arbeid

### D.3 Tilnærming/metode

Evaluering i kontrollerte omgivelser, for å unngå forstyrrelser som ville kommet ved f.eks. field-testing.

Deltakeren skal gis klare og tydelige instruksjoner og oppgaver, og observeres i gjennomføringen av disse. Deltakeren kan gjerne også bes snakke seg gjennom prosessen, for å få innsyn i tankene rundt de ulike stegene, og tydeliggjøre evt. utfordringer og forvirring.

### D.4 Oppgaver

1. Gå inn på appen, og finn hav-data fra ønsket sted. Er dette intuitivt?
2. Hvilke variabler er nødvendige fra vær/hav-API?
3. Hvor mange timer/dager frem i tid ønsker brukeren data fra?
4. La bruker forklare hvilke muligheter han/hun har til å navigere seg rundt.  
Er navigeringen intuitiv?
5. Noe brukeren savner? Feks. snarveier til å navigere seg rundt osv.

## D.5 Praktiske forhold

Mulig bias? Tekniske forkunnskaper som påvirker gjennomføringen? Dekkes målgruppen? Eksterne virkninger som forstyrrer gjennomføringen?

*Sørg for at deltakeren er i en komfortabel setting, og påpek viktigheten av ørlige tilbakemeldinger og innspill. Gjør gjennomføringen så “ekte” som mulig, og unngå på de konkrete oppgavene å gi mer info enn selve oppgaven, som om deltakeren skulle brukt appen helt alene. Ta gjerne bilder underveis til rapporten (om deltakeren samtykker til dette), men pass i tilfelle på å vise samtykkeskjema, og forsøk så langt det lar seg gjøre å unngå bilder som kan være identifiserende. (Om ansikt blir med på evt. bilder sladdes disse om bildet skal brukes). Pass på at deltaker har god tid og ikke må “stresse” gjennom oppgavene. Det er viktig at gjennomføringen blir gjort grundig, uten tidspress. (Eks. ikke klemt mellom to møter, at vi ikke har tid til å undersøke grundig osv.*

*Forsøk å “grave” etter forbedringspotensialer hele veien, SPESIELT om oppg. 5 gjennomføres. Få deltaker til å snakke seg gjennom alt han/hun gjør og tenker, og noter ned ønsker og irritasjoner underveis. Følg gjerne med på deltakers ansiktsuttrykk og kroppsspråk under gjennomføringen, og noter ned dette. Det kan gi viktig informasjon som ellers ikke nevnes, og vise underbevisste reaksjoner på forskjellige ting.*

*Husk at “Jeg syntes det var bra” eller “Kul app” er vanskelige tilbakemeldinger å jobbe med videre, selv om de er tilfredsstillende å høre der og da. Spør i tilfelle om HVA som var bra, HVA som var kult, hva som kan gjøre det enda bedre osv. ;)*