

PROSJEKTRAPPORT TEAM 39

IN2000 - VÅR 2022

Case 4 - Badesteder



Team

Daniel Corin Vaagland



Morten Åkre

Abdukerim Hasan

Eric Arthur Midford

Eskil August Ødegård



Veiledere

Ingeborg Westborg Steel



Khoi Le



Innholdsfortegnelse

Introduksjon	3
Brukerdokumentasjon	5
Produktdokumentasjon	10
Testdokumentasjon	18
Kravspesifikasjon og modellering	19
Prosessdokumentasjon	26
Refleksjon	41
Kilder	44

Introduksjon

Ole-Johan og Marie våkner opp til en nydelig dag, og solen skinner. De kommer på “Oj, kanskje vi skulle dratt ned til sjøen for å bade?”. Så slår det dem, “Hvor varmt er det i vannet nå? Det er jo litt kjapt å dra hele veien til Ingierstrand og så finner de ut at det er altfor kaldt i vannet?”

Dette scenarioet er det vi har tatt for oss. Vi ønsker å lage en app som til enhver tid kan levere badetemperaturer ved innsjøer og ved sjøen. Med over 400 punkter har vi et bredt spekter av badesteder langs norges kyst. Dette gjør det mulig å lage en app på nasjonal skala.

Vi har fra start ønsket at prosjektet skal gjennomføres med metoder som er på bransjestandard. Gjennom prosjektoppgaven har vi gjort nyttige og relevante erfaringer som vi kan ta med oss i arbeidslivet. Ved enhvers personlige eierskap til prosjektet har vi lært å ta ansvar, bidra, snakke om det en ikke forstår og kommunisere med andre. Ved denne innstilling har vi lært å jobbe som et team, fått et godt mellommenneskelig forhold preget av respekt for hverandre og hverandres kunnskap. Ingen er dårlige, alle har kvaliteter. Fra start har gruppen jobbet hardt, men vi fikk naturligvis utfordringer etterhvert med at vi havna midt i “gryta”. Mye var gjort, men mye gjenstod. Det viste seg at vi kanskje hadde vært litt kortsigting i fremgangsmåten, noe som vi etterhvert tok tak i.

Løsningen er en android-applikasjon som leverer vanntemperaturer på steder hvor en kan bade i Norge. Kledd i et sommerlig, lyst grafisk design gir den mulighet til å søke på badested, lagre sitt favoritt badested, legge inn standard startslokasjon og scroll gjennom mange badeplasser etter valgfri filtrering. Det er også en side med kart, hvor du kan få en visuell visning av hvor badestedene er og se avstanden fra din lokasjon til badestedet. Her kan du også få opp rask informasjon av ditt angitte badested. På tredje side kan du gjøre personlige endringer på hvordan du vil at appen skal fungere.

Beskrivelse av teamet

Teamet vårt består av 5 gutter. Fire av oss går studieprogrammet Programmering og systemarkitektur og én går Design, bruk og interaksjon. Vi er i grunn en fargerik gruppe, på tross at vi kun er gutter. Ingen kommer fra samme sted, vi har ulike aldre, ulik kunnskap og ulike interesser. Til tross for det er vi en gjeng som kommer godt overens, og har fått til et godt samarbeid sammen.

Daniel Corin Vågslid Vaagland

- Går Programmering og systemarkitektur. Er 23 år gammel. Født og oppvokst i Oslo. Har tatt 5 semestre så har litt mer erfaring innenfor linjen som de andre ikke har enda. Prøvde å ta initiativ med å sette opp møter, og lede samtalen på en grei måte.

Abdukerim Hasan

- Går Programmering og systemarkitektur. Er 42 år gammel. Uigur. Kommer opprinnelig fra Nordvest i Kina. Kan 5 språk. Har 10 års erfaring i helsesektoren.

Eric Arthur Midford

- Går Programmering og systemarkitektur. 21 år gammel. Kommer fra Japan. Lager spill, produserer musikk, og tegner pixel art i fritida. Har en maine coon som heter Frosty som elsker å gå på tastaturet på verste tid.

Morten Åkre

- Går Programmering og systemarkitektur. Er 20 år gammel. Kommer fra Arendal. Gikk rett på informatikk-studiet etter videregående. Treffer som regel det nærmeste treet på hvert kast i frisbeegolf.

Eskil August Ødegård

- Går Design, bruk og interaksjon. Er 22 år gammel. Kommer opprinnelig fra Molde. Bor nå i en bygd utenfor Ski, kalt Kråkstad. Tok fagbrev i mediegrafikerfaget før studiene. Har en bakgrunn innen film, klipping, og grafisk design. Kan spille piano, gitar og bass og synger mer enn gjennomsnittet.

Brukerdokumentasjon

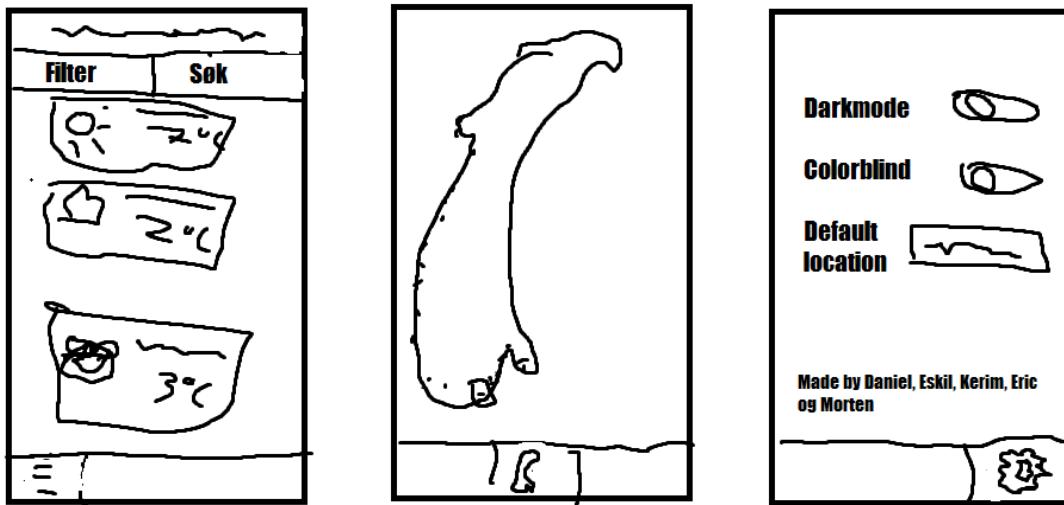
Når man åpner appen kommer man til hjemmesiden. Der kan man søke etter badesteder. Ved søk kan man filtrere badestedene etter hvor nærmeste er, eller hvor høy temperatur de har. Man kan sortere både på vanntemperatur og lufttemperatur. Trykker man på et badested utvides bolken slik at man får mer detaljert værdata på stedet. Man kan også se sist gang den er oppdatert over listen med badesteder. Hjemmesiden er en av tre faner. Navigeringen mellom fanene skjer ved en navigasjonsmeny i bunn av appen.

Velger en den midterste fanen i navigasjonsmenyen får man opp et kart der man kan se alle badestedene som har oppdatert informasjon. En kan se hvor man selv befinner seg ved en blå dot. Denne fanen er ganske simpel, den består av et kart hvor alle badestedene som har blitt oppdatert de siste tre timene står oppført med en rød markør. Du kan navigere deg rundt i kartet ved å zoome inn og ut, senterer til din lokasjon og navigere med touch. Hvis du trykker inn på en markør, får du informasjon om badestedet. Dette illustreres ved at en fane kommer opp og viser lufttemperatur, vindstyrke, solforhold og til slutt da vanntemperatur.

Velger man fanen til høyre i navigasjonsmenyen, kommer man til innstillinger. Der kan man velge om man vil bruke



mørkt tema, legge inn navnet sitt og legge inn lokasjonen sin. Denne delen ble ikke ferdig implementert.



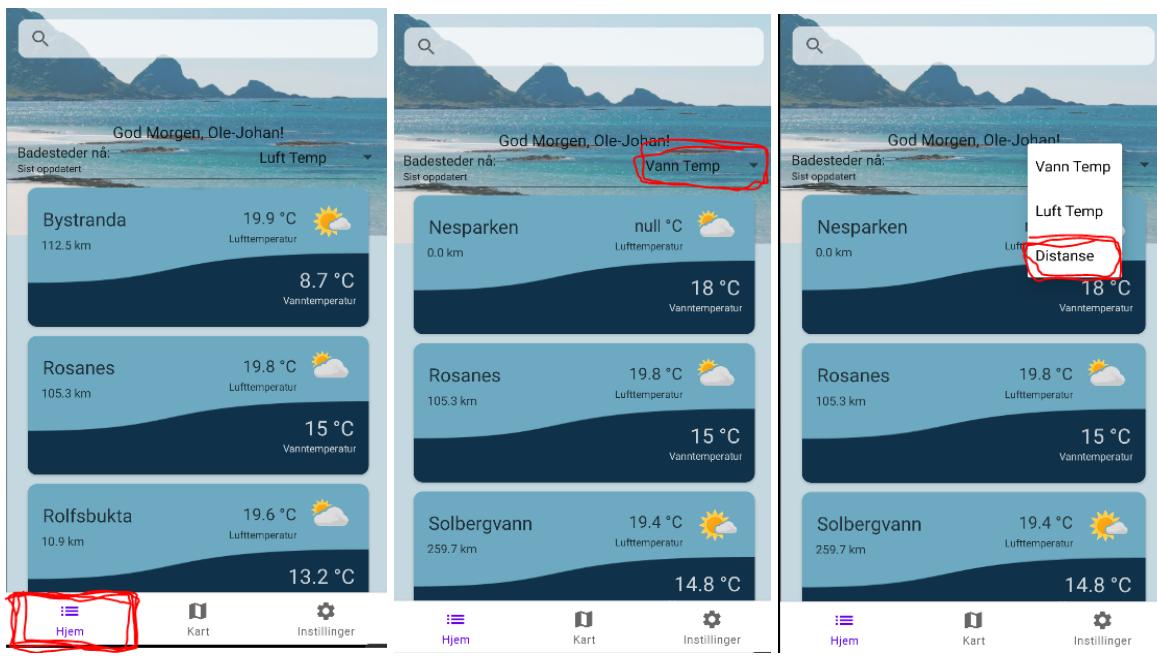
En skisse som ble tegnet tidlig på semesteret, som viser planen til hvordan hovedfunksjonaliteten til appen skulle være. En fane med liste over badesteder, med filter og søkefunksjon, en fane med kart og en fane med innstillinger.

Vi har spesifikt tatt med info om været i tillegg, og ikke bare badetemperaturen. Dette har vi gjort fordi vi har sett i brukerundersøkelsen vi gjorde at det er flere faktorer som er viktige for badeopplevelsen. Derfor har vi lagt vekt på å gi et bilde av totalopplevelsen. Lufttemperatur og værforhold spiller også en viktig rolle. Mer om dette kommer i prosessdokumentasjonen.

De mest nyttige funksjonene til applikasjonen med illustrasjoner:

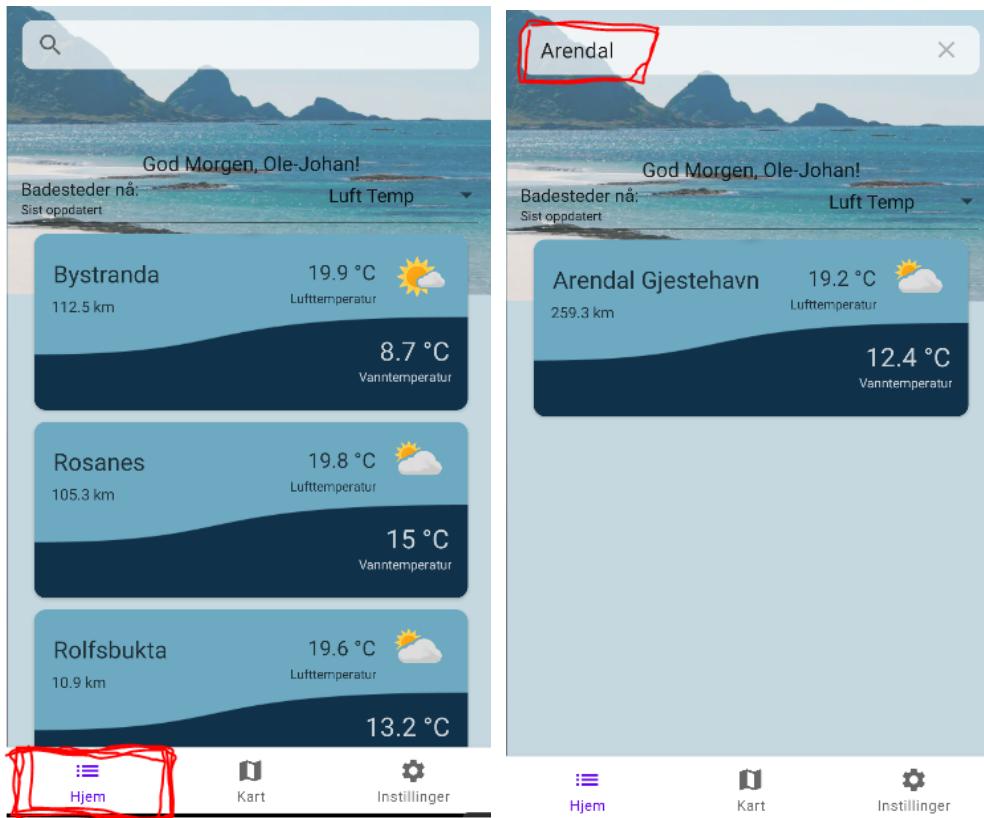
Finne siste badetemperaturer for de nærmeste badestedene:

1. Naviger til menyen nede i venstre hjørne.
2. Trykk på menyen opp i høyre hjørne
3. Sorter etter "Distanse".
4. De nærmeste badestedene med resultater vil vises.



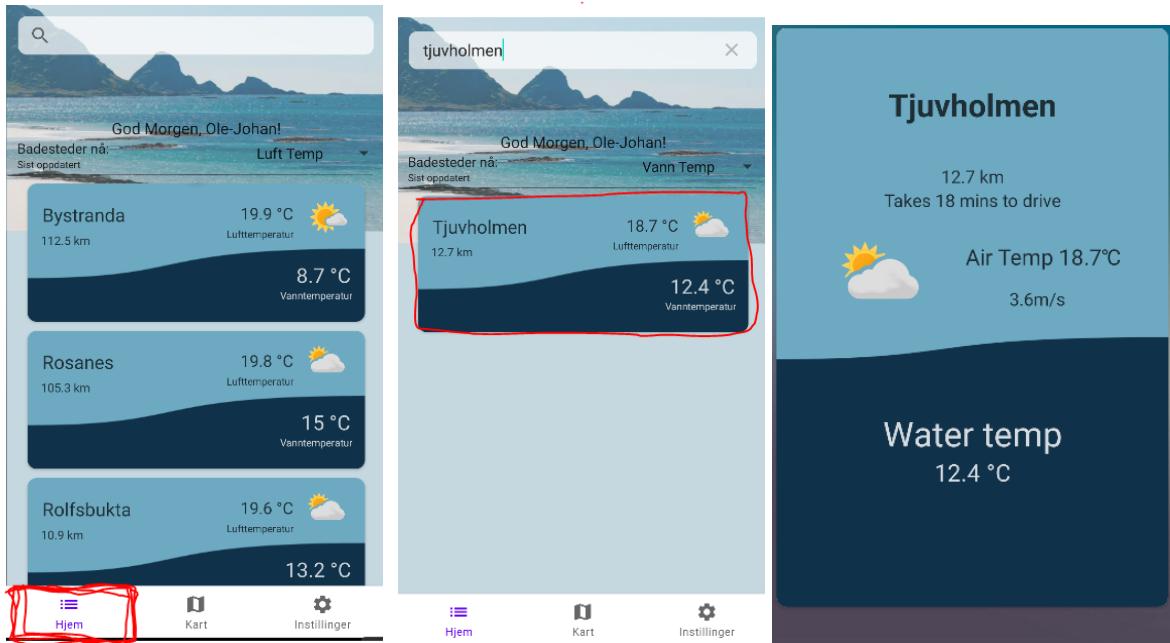
Finne siste badetemperatur for et spesifikt badested:

1. Naviger til menyen nede i venstre hjørne.
2. Skriv inn badestedet du vil finne temperaturen på i søkefeltet på toppen av skjermen.
3. Resultatet vil komme opp på skjermen.



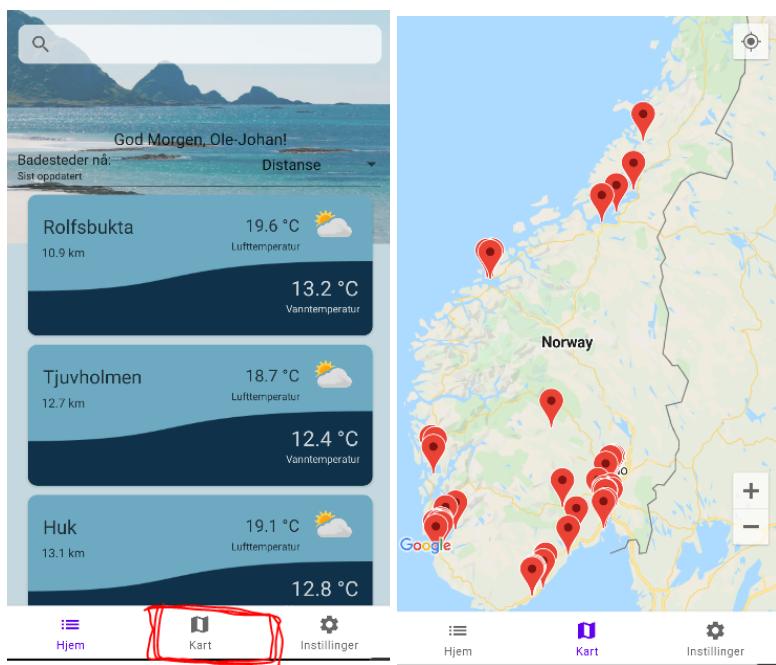
Finne tilleggsinformasjon om spesifikt badested:

1. Naviger til menyen nede i venstre hjørne.
2. Skriv inn badestedet du vil finne temperaturen for i søkefeltet på toppen av skjermen.
3. Videre trykker du boksen til badestedet.
4. Værdata for stedet vil bli presentert.



Åpne kart:

1. Trykk på den midterste knappen på bunnen av skjermen.
2. Tillatt at applikasjonen bruker posisjonen din, hvis applikasjon spør om det.
3. Du kan nå navigere deg rundt på kartet, og trykke deg inn på en markør du er interessert i.



Målgruppen

Målgruppen for applikasjonen er først og fremst for de som ønsker å bade i Norge. Det er de som er hovedbrukerne våre. Det kan være alt fra vanlige folk som vil ut for å bade en varm sommerdag, turister som vil planlegge hvilket badested de vil dra på eller bare en nysgjerrig person som liker å sjekke ut vanntemperaturen i nærheten av der han/hun bor. Appen skal være såpass simpel at enhver person skal kunne ta i bruk appen, uavhengig av hvor mye teknisk kunnskap personen har. Men appens funksjonalitet gir muligheter for et ganske bredt spekter av individuelle interesser. Den kan brukes for de som ønsker å drive vannsport, de som vil sole seg, spille beachvolley, ro i jolle eller hva det enn kan være. Derfor så kan appen brukes til mange tilfeller, men vi har spesifisert oss mot de som skal bade.

Praktisk sett har vi ikke noen planer om å legge ut appen. Det betyr at det er kun oss utviklere av applikasjonen som kan kjøre den. Men ideelt sett så hadde applikasjonen vært i nedlastbar på Play Store. Applikasjonen er geografisk begrenset til Norge, men alle skal i teorien få tak i appen.

Produktdokumentasjon

Kort fortalt:

Hele prosjektet skrives i Android Studio. Vi bruker Kotlin som programmeringsspråk. Fuel for å parse json til data classes. XML for layout. For design av UI har vi brukt Figma. Våre eksterne APIer var Havvarsel-Frost, Nowcast og Google Maps.

Konstruksjon av appen

Vi har valgt API-nivå 23 fordi det var det som ble anbefalt å bruke i de obligatoriske oppgavene. Dette API nivået kan brukes av 85% av alle android enheter. Dette gjør det mulig at flest mulig enheter kan laste ned appen. Hvis en bruker må oppdatere telefonen sin først vil det kanskje resultere i at brukeren ikke laster ned i det hele tatt. Hvis vi skulle ha lagt ut appen på google play så måtte vi brukt API-nivå 30 eller høyere, men dette er ikke noe vi har planlagt å gjøre.

Appen fungerer ved å først lese inn data fra Havvarsel-Frost for å sette dataen i en liste, og så går den igjennom hver buoi som er i listen og sjekker om Nowcast har data for koordinatene som buoyen er på. Før listen av data er ferdig bruker vi Google Maps for å få distanse til badestedet.

Vi begynner så å vise frem dataen grafisk der hver buoi får sin egen Card View. Card Viewene består av informasjon som for eksempel temperatur i vannet, distanse og navn på badested. Vi bruker Havvarsel-Frost sine ikoner for å representere dataene. Ikonene ble lastet ned fra Oslo MET, og er lagret lokalt i appen.

I tilfelle at Havvarsel-Frost eller Nowcast blir for mangefullt eller slutter å fungere og du må bytte endepunkt så trenger du å bytte ut et par ting.

- Endre data klassene sånn at de kan håndtere det nye endepunktet. Hvis du bruker samme navn på data klassene og variablene vil du trenge å endre mindre på programmet.
- Endre hvordan path variabelen ser ut, sånn at du får informasjonen du trenger fra den det nye endepunktet.

- Potensielt sette opp en proxy hvis det nye endepunktet krever det.



Hvordan den endelige card view listen vises grafisk

Listen av Card Views blir sortert etter spinneren som er nær listen. For å legge til en ny sorteringsalgoritme trenger man bare å legge til en ny comparator i GetComparator metoden som er i DataSource filen. GetComparator metoden ble laget for å være veldig lett å utvide. GetComparator tar inn en string og returnerer en comparator, ved å bruke when() for å se om strengen er lik ett av casene, som vil returnere riktig comparator. Det er viktig å legge til en identisk string inn i spinneren for å kunne velge den nye sorteringsalgoritmen.

```

companion object {
    fun getComparator(type: String): Comparator<Tsery> {

        when(type) {
            "Vann Temp" -> return compareByDescending { it: Tsery
                it.observations[0].body.value.toDouble()
            }
            "Distanse" -> return compareBy { it: Tsery
                it.header.extra.pos.distance
            }
            "Luft Temp" -> return compareByDescending { it: Tsery
                it.header.extra.pos.airTemp
            }
        }

        Log.d( tag: "COMPARE", msg: "Bad parameter: $type" )
        return compareBy { // default comparator if we get a bad parameter
            it.observations[0].body.value.toDouble()
        }
    }
}

```

Funksjon som returnerer et comparator objekt av type Tsery

```

<string-array name="sorting_types">
    <item>Vann Temp</item>
    <item>Luft Temp</item>
    <item>Distanse</item>
</string-array>

```

Hvor du må legge til nye spinner valg

Du kan også søke i listen av buoyer. Listen vil bli oppdatert ved å finne alle badeplasser som har strengen i søkerfeltet i navnet sitt. Dette er ikke case sensitive, og man trenger ikke å søke fra starten av stringen, altså så lenge det er en substring av navnet vil det vises.

For å vise de sorterte og filtrerte listene har vi en statisk og en midlertidig kopi av listen av badeplasser. Den statiske vil bare endres hvis dataen fra endepunktet skal oppdateres, mens den midlertidige kopien oppdateres hver gang noe søkes, eller skal sorteres. Søkene vil altså alltid gjøres på den statiske listen sånn at ikke noe data blir tapt.

Innstillinger fragmenten har kun fått et interface, og ikke fått funksjonalitet enda. Men dette kan utvides i fremtiden. Vi hadde tenkt til å legge til forskjellige tilgjengelighetsalternativer som for eksempel dark mode og alternative for fargeblinde, og personalisering ved at en kunne legge inn lokasjonen og navnet sitt. Dette er det naturlige stedet å fortsette å jobbe.

Fuel

Vi brukte fuel for å sende forespørselen til json filen som vi fikk fra Havvarsel-Frost og Nowcast. Fuel kan byttes ut med hvilken som helst modul med lik funksjonalitet.

Gson

Vi brukte Gson for å gjøre parse json objektene til data klassene som vi jobbet med. Gson kan byttes ut med andre json parsers.

Havvarsel-Frost

Havvarsel-Frost er hoved endepunktet som vi bruker i denne oppgaven. Vi får en liste med badestedene som inneholder info som for eksempel temperatur. Vi får også koordinater som vi bruker når nowcast skal finne info om badestedet. Havvarsel-Frost har også gitt oss et par problemer som for eksempel treige responstider og få resultater. Havvarsel-Frost krever ingen proxy for å svare på forespørsler. Noen av medlemmene våre har fått problemer med å få resultater fra Havvarsel-Frost der noen av gangene man sender en forespørsel så får vi en feilmelding som sier:

Read error: ssl=0xec6e5498: Failure in SSL library, usually a protocol error

Denne oppstår oftere på visse tidspunkter av døgnet og vi fant ikke ut av hvorfor bare noen av oss fikk dette problemet.

Nowcast

Nowcast er vårt sekundær endepunkt som vi bruker for å finne lufttemperatur og vind hastighet. Vi bruker koordinater for å finne stedene som vi har lyst til å hente data fra. Har ikke funnet noen mangler fra nowcast.

Directions API fra Google Platform

Med Google Platforms Direction API, kan vi få ganske mye informasjon mellom to lokasjoner. Vi kan hente ut avstand, reisetid med forskjellige transportmidler som for eksempel bil, kollektivtransport, sykling og gåing.

Får å kunne bruke Directions API, må man ha Google Platform konto og man må aktivere Directions API service. Du kan få tilgang til Directions API via et HTTP-request, med forespørsler konstruert som en URL-streng. Strengen inneholder en startposisjon og en destinasjon. Man må også oppgi API-nøkkelen sin med hver forespørsel. (Google MAPs Platform, Directions overview)

Typisk API request:

https://maps.googleapis.com/maps/api/directions/json?origin=xxx&destination=xxx&key=YOUR_API_KEY

Man kan bruke steds navn eller koordinator posisjon til origin og destination. I vår tilfelle, brukte vi koordinatoren til brukerens posisjon som origin og koordinatorene til badestedene som destinasjon, i denne måten har vi fått avstanden til de forskjellige badestedene.

Vi har også tatt med oss reisetiden til badestedene. Man kan få opp forskjellige tider avhengig av transport metode med directions API, men vi brukte bare tiden som tar til å kjøre destinasjonen.

Google Maps

For å ha kart funksjon i appen, brukte vi Google MAPs Software Development Kit (SDK). Med MAPs SDK, kan man ikke bare inkludere kart i appen, også tilpasse og modifisere funksjonaliteten etter eget ønske.

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 eller MAPs fragment.

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.

Klassen Google MAP håndterer automatisk tilkobling til Google MAPs' servere, laste ned og vise kartet på skjermen, samt at det viser og responderer på diverse brukerinteraksjon. I tillegg kan metoder med ekstra funksjonalitet implementeres. (Add MAPs, 2019)

Når brukeren åpner kartet for første gang, spør appen om brukerens lokasjon. Når brukeren trykker på «aksept», viser appen kartet over Norge med markering av brukerens posisjon. Hvis appen hadde fått data fra endepunktene, så lages markører til ethvert badested. Brukeren også lett zoome inn og zoome ut ved hjelp av zoom-Controller.

Når brukeren klikker på markørene, fører den til details_fragment som man kan gå også gjennom list_fragment item klick.

Fragments

Navigasjonsknappene som ligger nederst på appen er en del av MainActivity. Alle andre UI-elementer ligger i en Fragment som blir lagt inn i en LinearLayout med id container i MainActivity. Når man trykker på en av navigasjonsknappene blir korresponderende fragment klasse lagd og byttet ut med fragmentet som ligger på skjermen. Funksjonskallet tar inn fire animasjoner som bestemmer hvordan det ser ut når den gamle fragmentet går fra skjermen og når den nye kommer inn. Animasjonene kan egendefineres, og skal ligge i res/anim.

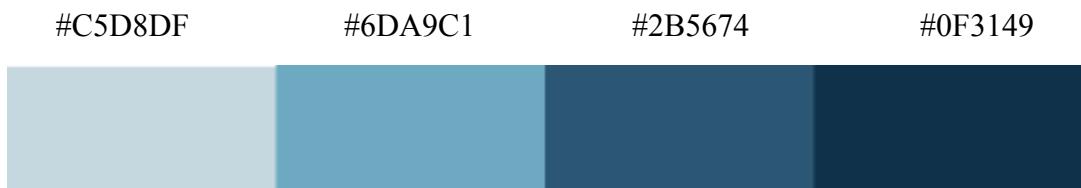
Navigasjonsknappene er laget med en BottomNavigationView, som tar informasjon fra res/menu/nav_items.xml til å lage knappene. Her kan du spesifisere tekst og ikoner som skal vises på dem. Du må gi knappene en android:id slik at du kan binde kode til onItemSelected i MainActivity.

Om du skal lage en ny Fragment, må du huske å lage i klassen en companion object med en funksjon newInstance() som returnerer en instans av fragmentet. Dette gjør det enklere å instansiere en ny Fragment når du bytter dem i MainActivity.

Fragments kan gjøre det først litt avansert, men den gir flere muligheter enn om man skulle ha en activity for hver side. Appen bruker `MainActivity` som host. Inne i main finnes det flere fragments. Hjem, kart, bade og innstillinger er fragments. Grunnen til at vi bruker fragments er at det gir mulighet for gjenbruk. Hvis vi ikke hadde brukt fragments ville livssyklusen til hver side blitt avsluttet når du hadde begynt en annen. Ved fragments kan en navigere frem og tilbake. Eksempelvis vil du ikke måtte sortere på nytt på hjemmesiden etter å ha vært i settings.

User Interface

For interfacet har vi brukt xml til utforming. Vi har hentet designelementer fra Figma, og lagt de inn som drawables. Filene er png-format som er det prefererte filformatet for grafikk. Egenskaper som komprimering uten datatap, lav fargetap og gjennomsiktighet gjør det til det beste alternativet. Disse har blitt lagt inn som background i de ulike elementene. Når man skal gjøre endringer er det viktig at man sørger for at det er innenfor det grafiske designet. Derfor må en bruke de rette fargene som du kan se her:



Fargepalett

Dette kan endres ved å redigere fargeattributtet som ligger i hvert element.

Det er mye tekst og informasjon som skal illustreres, og det kan bli noe utfordrende for bruker å vite hva som er viktig og hvor personen skal se. Dette har vi løst på den måten at vi bruker farger til å skille på elementer og samtidig ulik tekststørrelser. Teksten kan endres med `TextSize`. Sørg derfor for at det viktige har noe større skrift og/eller at det er luft rundt. Elementer som sitter for tett er ikke estetisk.

Tilbake til det tekniske så har vi som nevnt id. Alle elementer har en id som du kan binde til koden. Dette er bindeleddet mellom hva du ser og hva som er i koden, altså mellom xml og kotlin. Det er viktig at alle navn på `IDer` er logiske slik at en lett kan finne den frem.

Kvalitetsegenskaper

Positive:

- Siden vi bruker google maps for å vise distanse til badeplassene blir avstanden målt med veien man faktisk trenger å reise hvis man skal til badeplassen. For badeplasser som er nærmere vil dette være en mye mer realistisk representasjon av tiden det vil ta å ankomme der.
- All data som vises vil være oppdatert innen de 3 siste timene.. Temperaturen i vannet vil ikke endres alt for mye på 3 timer, derfor vil informasjonen være pålitelig.
- User interface er laget med tanke på brukervennlighet, knappene er tilgjengelig ved tommelen sånn at du kan lett bytte side. Siden man også kan oppdatere listen ved å dra listen opp er dette også veldig lett å gjøre. Teksten er stor for å være lesbar for flest mulig mennesker. Dette er for å gjøre appen leselig av mest mulig folk, altså dette fremmer universell utforming.
- Brukere er allerede vant til interfacet som google maps tilbyr, derfor vil brukeren mest sannsynlig være tilvent hvordan maps fragmenten fungerer.
- Selv om appen ikke får tilgang til informasjon fra ett av endepunktene, vil dette ikke lede til noe problemer i appen, bare refresh listen og prøv igjen. Ingenting i appen skal lede til at appen kræsjer.
- Appen krever ikke mye plass, veldig lite informasjon er lastet ned lokalt, men hentes fra internettet isteden. Appen bruker ikke lang tid på å laste alt unntatt dataen fra API kallene.
- Appen er sikker med tanke på at ikke noe persondata lagres eller vises i appen. Det er ingen åpenbare sikkerhetsfeil.
- Appen bruker fragments som lar oss lett bytte hva som vises på appen uten å bytte ut andre viktige deler av grensesnittet.
- Appen ser moderne og er visuelt appellerende.
- Informasjonen en bruker kan hente ut fra appen er all den viktigste informasjonen som man kan få ut fra endepunktene Havvarsel-Frost og Nowcast på en veldig tilgjengelig måte.

Negative:

- Havvarsel-Frost kan bruke lang tid for å svare på forespørselen vår. Dette kan være litt irriterende for brukeren, men det tar ikke så lang tid at det påvirker tilgjengeligheten på dataen.

- Siden appen bruker Havvarsel-Frost har den en mangel av data å representer. Dette er fordi det ikke alltid er oppdatert data som programmet kan vise, siden den bare viser data som er opp til 3 timer gammelt.
- Havvarsel-Frost kan gi en timeout, eller lignende feil når appen spør om informasjon om badeplasser. Dette er ikke noe vi kan gjøre noe med.
- Det er ikke implementert noen form for innstillinger som kan endres av brukeren. UIet er designet, men den vil ikke gjøre noe. Derfor svekker dette hvor tilgjengelig appen er. Kunne vært flere funksjoner som fremmer universell utforming.
- Ikke personalisert funksjonalitet. Det er ingen måte å legge inn eget navn, eller endre hvilken posisjon appen bruker som standard posisjon. Distansen blir generert fra ifi sine koordinater som et default, noe som ikke er relevant for de fleste.

Testdokumentasjon

Siden vi har brukt Android Studio til å jobbe har vi også brukt deres innebygde emulator for å teste appen under utvikling. Når det kommer til statiske tester har vi gjort to forskjellige metoder. Under kjøring har vi logget mye til logcat for å se hvordan programmet fungerer. Veldig fint for å se når APIet ikke svarer blant andre problemer som kan oppstå. Vi har også sett på koden som hverandre har skrevet og sjekket den for feil, noe som har resultert i en mer robust kode. Det er mange feil som man ikke ser selv.

Integrasjonstest

Vi har en API-test som sjekker om vi får et resultat fra Havvarsel-Frost. Dette er for å kunne se om vi faktisk får noe form for data fra endepunktet. Vi har brukt Android Studio for å lage

en instrumented test. Den sitter i androidTest mappen generert av Android Studio. Denne testen bruker også JUnit.

UI-testing

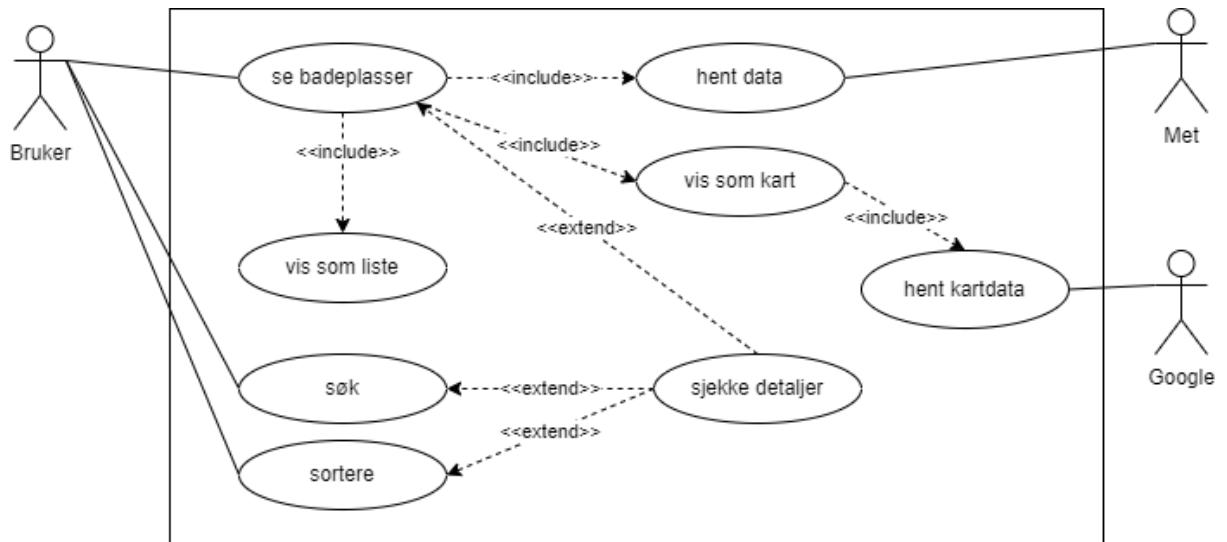
Vi har også implementert en rekke intergrasjons- og brukergrensesnitt-tester. Disse har som formål å sjekke at brukergrensesnitt og det funksjonelle i appen fungerer som det skal. Testene simulerer interaksjon med brukeren, og sjekker at resultatene blir som forventet.

For å utføre disse testene brukte vi Espresso biblioteket. Med fragment-test dependency sin fragment launcher funksjon, testet vi de forskjellige view-ene vises riktig på skjermen.

Testene som utføres i “UITest.kt” sjekker at:

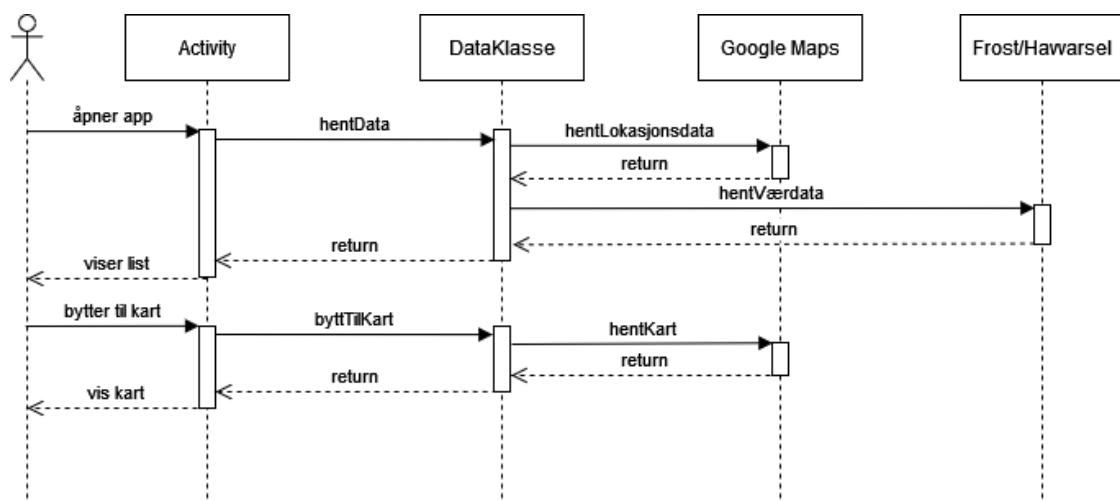
- ListFragment vises som det skal, og sjekker at textView – «sist_oppdatert» vises på skjermen og faktisk inneholder teksten «Sist Oppdatert»
- Navnet til applikasjons context faktisk heter «gurppe39.android. in2000.prosjekt»
- Menu bar nederste i appen faktisk navigerer til riktig fragment. Med klikke på knapper, brukeren navigert til tilsvarende fragment.

Kravspesifikasjon og modellering



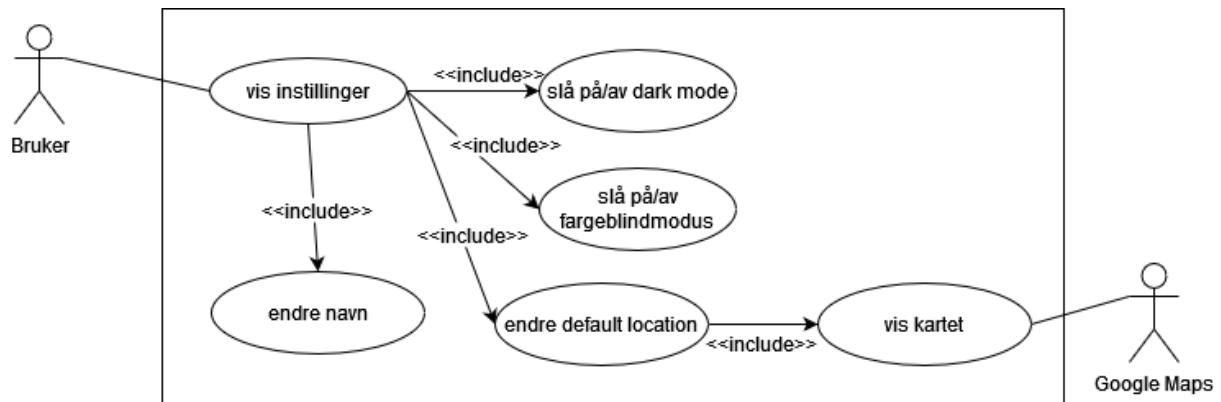
Use case diagram til MVP

Det viktigste funksjonelle kravet er å vise frem badesteder og relevant informasjon. Appen skal vise brukeren hvor badestedet er på kartet, distanse til badestedet, vanntemperatur, lufttemperatur, og værforhold. Dette var basisen som vi bygde på med MVP-en. Vi bestemte oss mens vi lagde MVP-en å legge krav om at bruker må ha muligheten til å sortere resultatene og søke for badesteder slik at brukeren kan enkelt finne relevante badesteder, da dette er en vanlig og veldig nyttig funksjon.



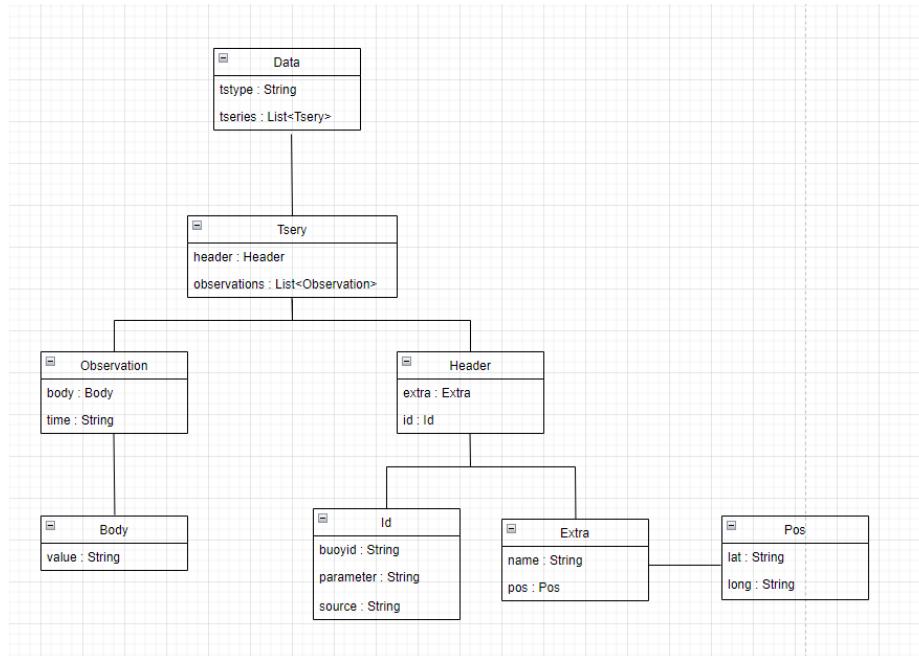
Sekvensdiagram til MVP

Innstillinger ble aldri fullt implementert. Men vi har fortsatt modellert hvordan det ville ha vært. SettingsView skulle hatt fire innstillinger: en som slår dark mode av og på, en som slår fargeblindmodus av og på, en som lar deg endre navnet som appen bruker for å hilse deg, og en som setter din default location. Innstillingene ville ha vært lagret lokalt på enheten og lastet opp hver gang prosjektet ble kjørt.



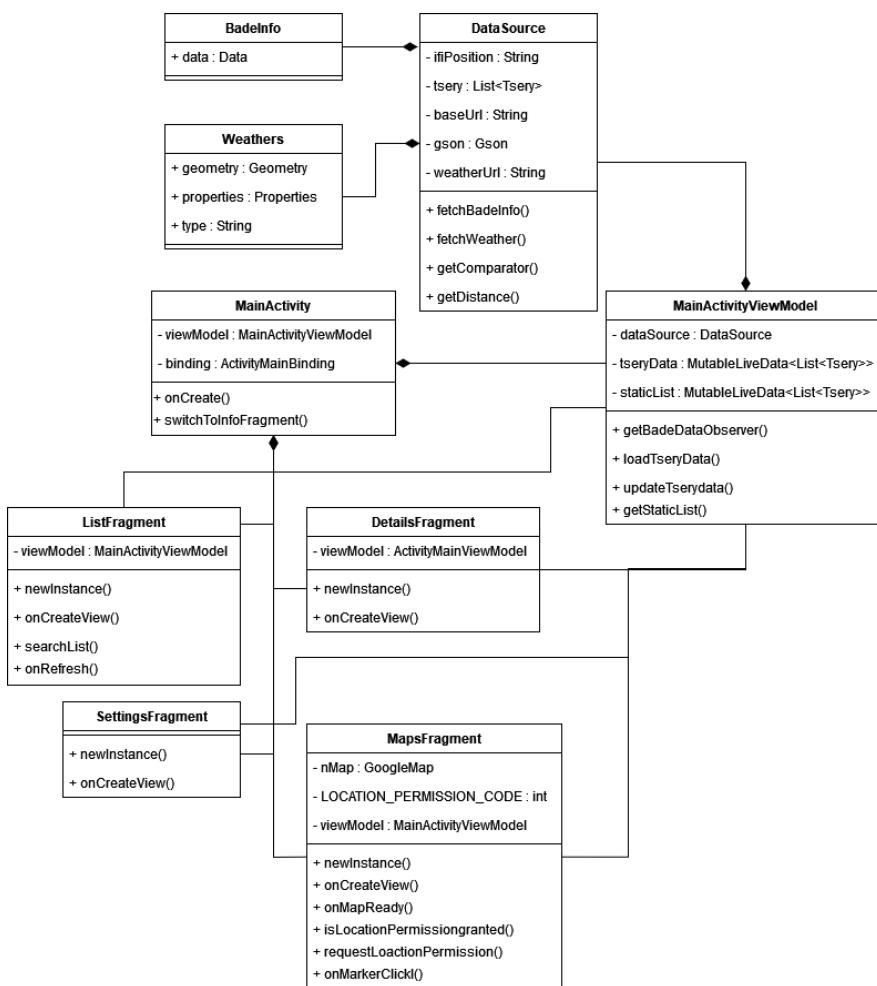
Use case diagram til innstillinger

Modellering



Diagrammet over illustrerer klassene vi får når vi kaller Havvarel-Frost APIet. Når vi derealiserer med Gson får vi et Data-objekt som inneholder en String, som i alle tilfeller vil

være "badenvann", pluss en liste av objektet Tsery. Det er denne listen vi bruker til å vise dataen visuelt i appen. Objektet inneholder en liste over observasjoner og en Header. Listen over observasjoner vil i alle våre tilfeller bare inneholde akkurat ett objekt. Det er fordi når vi kaller APIet ønsker vi kun å få opp badesteder som har observasjoner de siste tre timene, og denne siste observasjonen. Et Observation-objekt inneholder en Body, som igjen har en value som er antall grader celsius i vannet. Den inneholder også en time som er et tidspunkt for observasjonen. Headeren inneholder to klasser, Id og Extra. Id inneholder en id som er unik for hver badepllass, en parameter som i alle tilfeller vil være "temperature", og en kilde. Kilden kan være for eksempel "badetassen.no" eller "yr.no". Extra inneholder navnet på badestedet og et posisjonsobjekt. Dette objektet inneholder koordinater i form av lengde- og breddegrader. Disse bruker vi videre til å finne værdata som lufttemperatur og vind på de ulike badestedene.



Diagrammet over illustrerer klassestrukturen i prosjektet. Prosjektet er bygd rundt MainActivity, som holder instanser av alle fragmentene og MainActivityViewModel. ListFragment, DetailsFragment og MapsFragment har en referanse til MainActivityViewModel, og bruker dette til å få tak i data som er viktig til kjøringen av prosjektet. MainActivityViewModel holder på alt av data som kommer fra DataSource, som lager API-kall, henter resultatene, og returnerer dem. MainActivityViewModel legger dem inn i en LiveData liste, som melder fragmentene at det har blitt en endring i listen.

MVVM og Objektorientert Prinsipper

I vårt tilfelle har mye av det å gjøre med objektorientert prinsipper og MVVM design patterns ikke blitt diskutert, men takket være Kotlin og Android Studio sitt infrastruktur ivaretar prosjektet disse prinsippene og design patterns.

Vår “Model” i MVVM kommer fra DataSource klassen og Met sitt API. API-et holder alt av data vi trenger, og DataSource tar dataene og gjør det brukbart for appen slik at vi kan vise det til brukeren. Fordi vi bruker ikke noen annen data er dette delen av prosjektet godt innenfor MVVM.

Vi bruker vår egen ViewModel, MainActivityViewModel, til å håndtere kommunikasjon mellom DataSource og vår View. Den oppdaterer informasjonen med å bruke metodene i DataSource, og informerer vår forskjellige fragments om endringene slik at de kan vise frem dataene. Våre fragments, som håndterer alt grafisk, kan spørre ViewModel til å hente ny data, men endrer ikke på dataene selv. Men, i forhold til vår implementasjon av en Model, har vi litt av en overskridning her. For å implementere søkefunksjonen i ListFragment endrer vi på en midlertidig liste som holder relevante resultater og gir den listen tilbake til ViewModel. For å best ivareta MVVM ville det vært bedre å ha en metode i MainActivityViewModel som kunne gjøre dette. Dette er også en av de største overskridelser av objektorientert prinsipper, der en klasse endrer på data i en annen klasse. Her er det kobling mellom ViewModel og ListFragment. Ellers er klassene godt skiltet, med lav kobling og høy kohesjon over hele prosjektet.

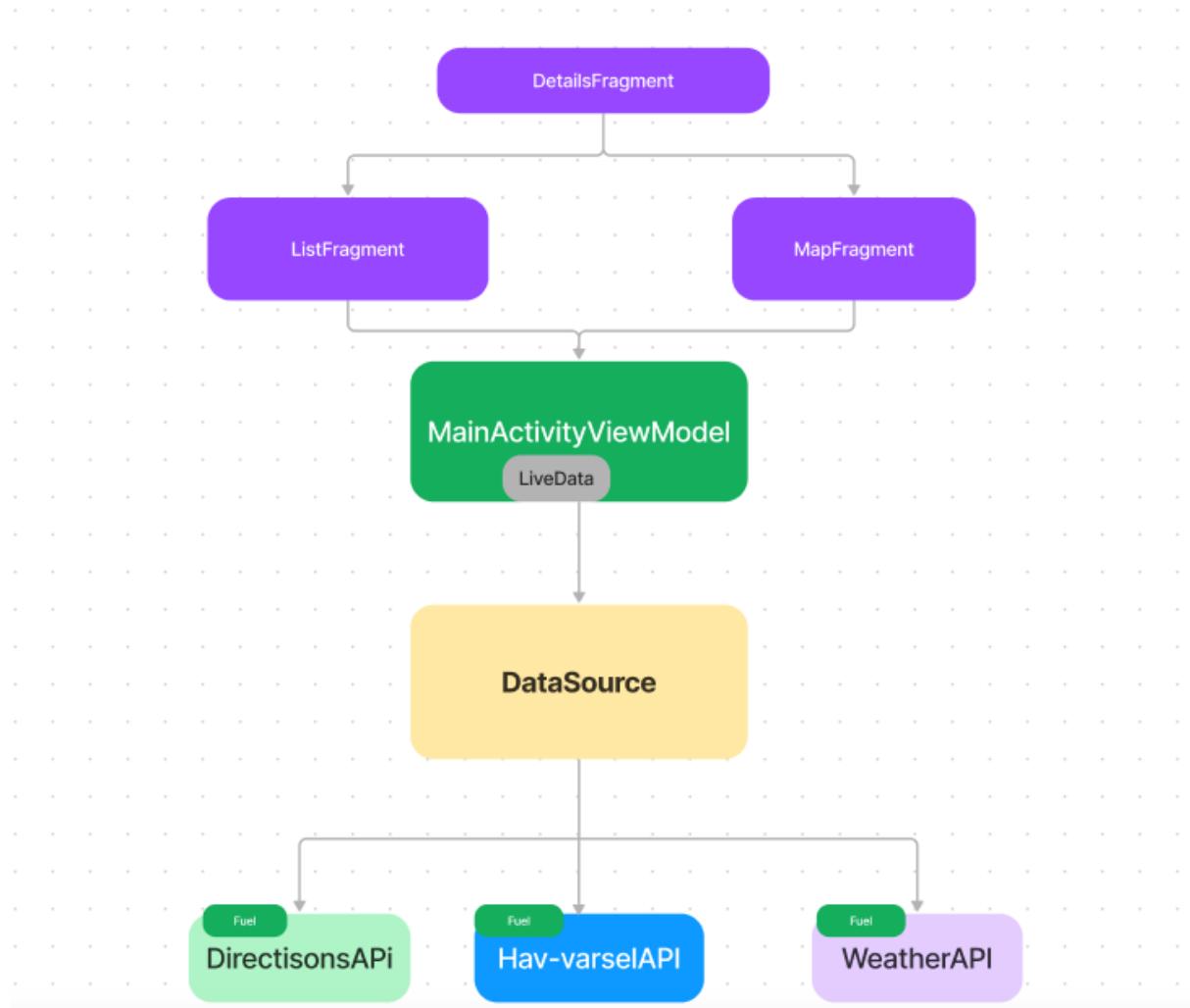
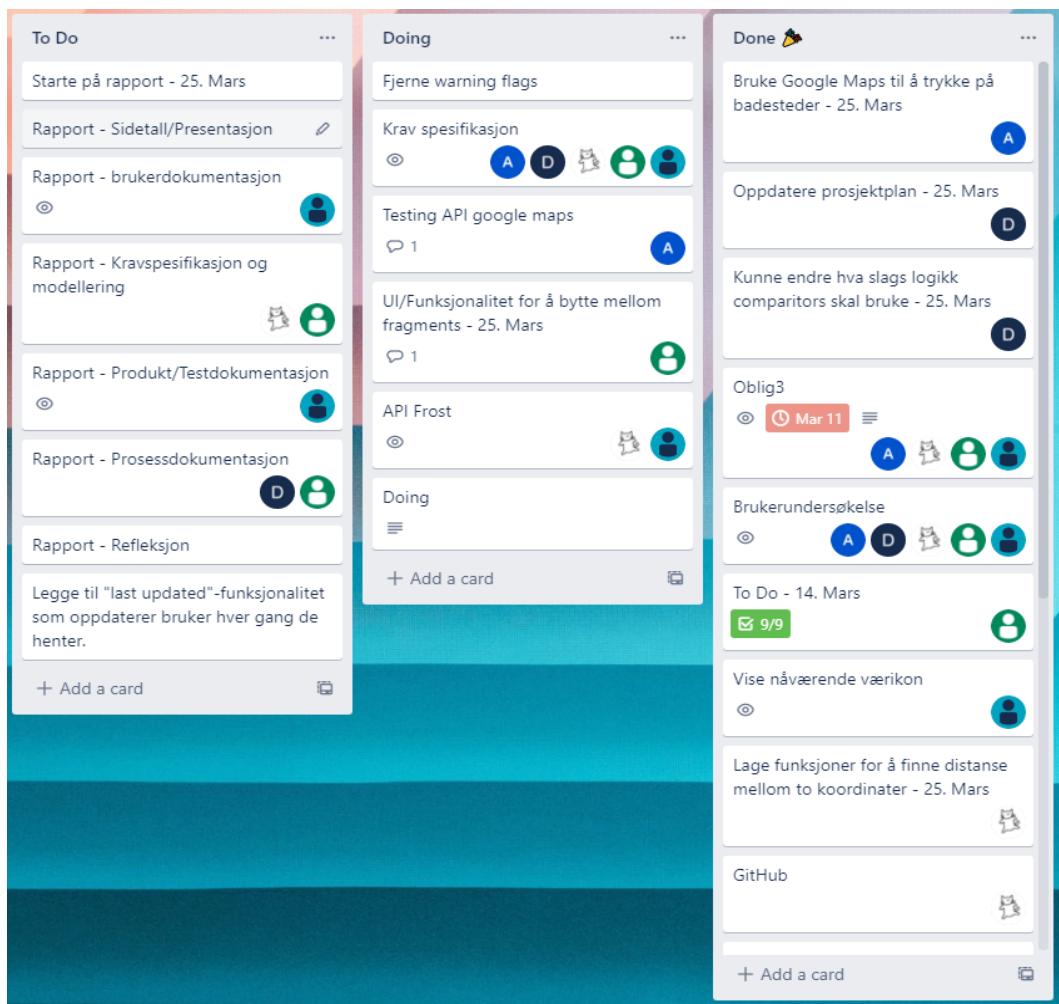


Figure: MVVM

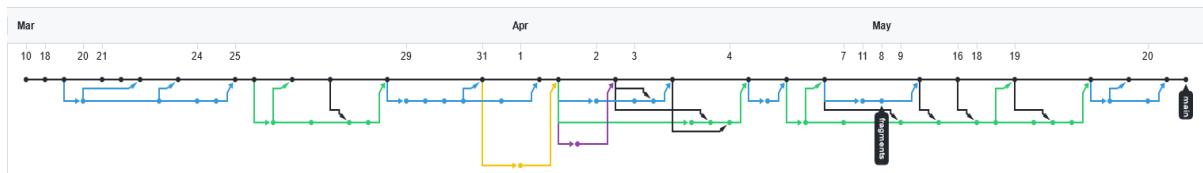
Prosessdokumentasjon

På starten av prosjektet snakket vi om hvordan vi ønsket å bruke en mellomting mellom scrum og kanban som utviklingsmetode. Vi bruker et kanban-board i Trello for å sette opp oppgaver, og setter opp hvem som skal gjøre hvilke oppgaver. Dette fungerte bra i en lengre periode. Vi satte opp nye oppgaver om hva som skulle gjøres fram til neste gang vi skulle møtes, og hvem som skulle gjøre hva. Vi har diskutert hva som må gjøres, men etterhvert ble det til at vi heller tok det muntlig siden vi møttes jevnlig. Grunnen til dette kan være at vi ble trygge på hverandre i løpet av prosjektet. Men likevel skulle vi absolutt fortsatt med det helt frem til levering. Rundt påske ble oppgavene litt vag, og tidsperspektivet druknet litt. Så vi skulle absolutt ha holdt oss til kanban-boardet. Her opplevde vi hvordan tiden gjorde at vi ikke prioriterte app-prosjektet så mye. Men vi fikk en vekker noen uker før slutten når vi innså at vi var nær slutten.



Bildet over viser kanbanboardet vi har brukt til planlegging.

Vi har brukt GitHub for deling av kode på prosjektet. En utfordring vi har hatt i løpet av prosjektet er bruken av dette, særlig tidlig. På gruppen var det noen som hadde god kunnskap og erfaring med bruk av dette, mens noen ikke hadde noe særlig erfaring med det i det hele tatt. Når vi tok en rask gjennomgang om hvordan det skulle brukes synes alle det virket greit ut, noen lærte seg noen nye uttrykk som "push", "commit", "fork", "merge", "fetch" og "repository". Når vi senere skulle ta i bruk dette på maskinen alene hjemme, oppstod det ofte litt problemer i starten av prosjektet. Vi fant til slutt ut at alle måtte ha en egen "fork" av hovedprosjektet på sin egen profil. Det var da enkelt å "fetche" den eldste versjonen av prosjektet til sin egen maskin, gjøre endring og deretter "pushe" endringene. Vi har gjort det slik at en av oss har vært ansvarlig for å "merge" endringene, slik at han kan ta en sjekk over at alt er som det skal før han velger å la endringene bli en del av hovedprosjektet. Et problem med dette var at kommunikasjon i forhold til ny pull requests var ikke særlig god. På grunn av dette var det flere pull requests som ble ikke merget inn for en god stund. Alt i alt har det fungert bra å bruke GitHub, det var bare litt nytt for enkelte i starten.



Grafisk representasjon av historikken av prosjektet på GitHub

Prosjektets gang

Prosjektet hadde en litt treg start på grunn av at det tok litt tid før vi alle fikk møtes samtidig. Vi møttes på ifi, motiverte og spente på å komme i gang med prosjektet. Vi så at vi hadde en god spredning av evner. Vi planla og møtes minst en, helst to ganger i uken fra starten av, og heller øke etter behov etter hvert. Dette har fungert bra, på møtene har vi snakket om hva vi

har gjort, eventuelle utfordringer vi har støttet på, og hva som må gjøres videre. Når vi først kom i gang med prosjektet tok det ikke lang tid før vi hadde kommet ganske langt på appen.

Vi har diskutert hvordan vi ser for oss at appen skal se ut, hva som er viktig funksjonalitet for en bruker og hvordan ting skal implementeres. Dette har vært viktig slik at vi alle hadde den samme forståelsen av målet. Ofte har noen kommet på en god ide som har ledet til gode diskusjoner. Eksempler på dette kan være noe som følger: "Har vi tenkt noe på universell utforming? Vi kan jo for eksempel lage en modus for fargeblinde?" eller "Hadde det ikke vært kult om brukeren kan ha sin egen personlige liste over favoritt-badesteder?".

Når vi drøftet hvilket case vi skulle velge tok det ikke så lang tid før vi fant en favoritt. Vi hadde ingen på gruppen som brant for noe spesielt eller hadde noen fancy ideer, så da tok det ikke lang tid før vi kunne legge bort det åpne caset. En hangglider-app virket kanskje litt spennende på starten, men vi fant ut at en slik app ville blitt litt for spesifikk, spesielt med tanke på at ingen på gruppen har noe spesielt forhold til hanggliding. Teamet vårt består av flest prosa-studenter, noe som gjorde at vi tenkte at case nr. 2 ikke passet så godt for oss. Vi så for oss at dette caset ville kreve en god del mer design enn de andre casene. Vi tenkte litt på det samme når det gjelder case nr. 5. Det var ingen på gruppen som synes det var noe spesielt spennende med lakseoppdrett. Etter vi hadde eliminert fire av casene, stod vi igjen med case nr. 1 og nr. 4. Her var vi litt usikre, men det endte til slutt med case 4. Med denne oppgaven så vi fort for oss noen ideer om hvordan vi kunne få appen til å se ut. Vi snakket blant annet om at vi kunne ha topplister over de badestedene med høyest temperatur og lister over de badestedene som er nærmest din egen posisjon.

Vi lagde en teamavtale i oblig3, som vi har fulgt hele veien. Den består av noen punkter som vi tenkte var lurt å kunne forholde oss til. Vi synes at vi har vært flinke til å holde det som står i avtalen. Et unntak er i perioden etter påskeferien, der det tok en stund før vi møttes og begynte å jobbe med prosjektet igjen etter ferien.

- Hvis vi har avtalt et møte, skal alle medlemmer møte opp. Hvis man plutselig ikke kan møte til tidspunktet som er avtalt, skal det gis beskjed på teams.
- Hvis man har fått en oppgave som skal være ferdig til et tidspunkt, skal man bli ferdig til tidspunktet som er gitt. Hvis man sliter med oppgaven skal man gi beskjed til de andre på gruppen, slik at han kan få hjelp før møtetidspunktet.
- Hvis vi finner ut at arbeidsmengden er ujevnt fordelt, skal den som har fått mindre

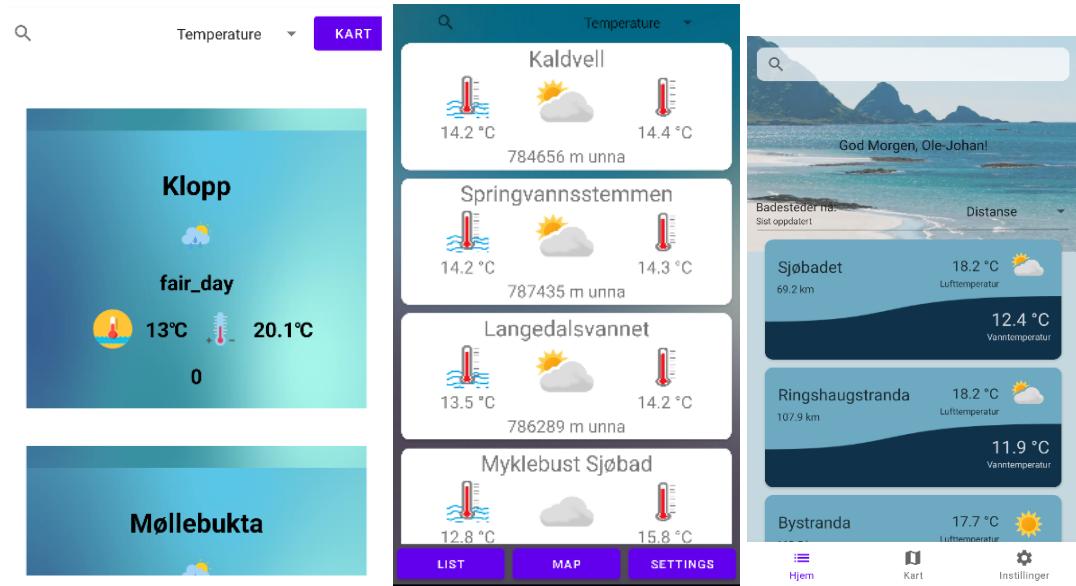
arbeidsmengde hjelpe den som trenger hjelp. Det er viktig at vi starter i god tid, sånn at man har tid til å spørre om hjelp.

- Vi skal møtes minst en gang i uken, enten fysisk eller digitalt. Vi skal møtes oftere ved behov.
- Alle føler personlig ansvar for prosjektet. Alle skal gjøre sin del, men man skal også kunne hjelpe andre i gruppen.
- Ved uenigheter eller avvik skal vi snakke om det og løse det.
- Når prosjektet er levert, skal vi finne på noe gøy sammen.

Selve prosjektet startet med å gi alle medlemmene på teamet et API som de skulle se nærmere på. Vi så på tre forskjellige APIer: Havvarel-Frost, NowCast og Google Maps. Når vi møttes gangen etter dette hadde vi fått et godt overblikk over hvordan disse så ut, og hvordan vi så for oss å bruke dem. Videre lagde vi funksjonalitet for å ta i bruk de ulike API-ene. Når dette var gjort kunne vi bruke dataene vi fikk av APIene til å fremstille infoen om badestedene grafisk. Underveis i prosjektet har vi fått mange forskjellige ideer om hva vi vil ha med i appen. Mange av disse har det ikke blitt gjort noe av, mens noen har blitt diskutert nøyere og til slutt blitt implementert. Det har også blitt gjort mye endringer på UI i løpet av prosjektet. På starten ville vi bare sjekke at alt fungerte ved å fremstille dataene vi fikk fra Frost-APIet med tekst, slik at vi fikk se om alt fungerer. Videre har vi laget flere design som har blitt finere og finere, og til slutt kom vi med et design som vi var fornøyd med som et ferdig produkt.

Når vi diskuterte hvordan vi ville at appen skulle se ut, kom vi på noen ting som var viktige for oss. Vi ville få til et design som gjorde at appen skulle få sitt eget særpreg, og skille seg litt ut fra andre lignende apper. Vi begynte derfor å diskutere om vi skulle lage egne ikoner for temperatur, vanntemperatur, sol, sky og så videre. Vi var enige i at kunne være en gøy ting å gjøre. Fra NowCast APIet får vi en beskrivelse av hvordan været er, som for eksempel: "fair_day", "heavysnow" eller "cloudy". Vi fant ut at APIet har over 50 forskjellige slike beskrivelser, og dermed ville det tatt alt for lang tid å lage egne ikoner for alle disse. Dermed endte vi opp med å kun lage ikoner for vanntemperatur og lufttemperatur, og bruke disse for hvert badested. Til slutt satt vi igjen med et design som ikke bruker disse ikonene, men som

har et design som skal gjøre at brukeren ser hvilken temperatur som er vann og luft på grunn av designet.



På bildene over ser vi hvordan designet til hovedskjermen har endret seg over tid. Vi er enige om at designet har endret seg til å bli mer estetisk enn de tidligere versjonene. Det har vært viktig siden appen skal være innbydende for målgruppen. Opplevelsen av appen skal være noe som gir gode, sommerlige assosiasjoner til å bade. Dette er også begrunnen til de fargene vi har gått for. Det er veldig interessant å se progresjonen fra første versjonen til den siste.

Borrevarnet

fair_day



Water temp: 13.82°C

Air temp: 21.2°C

Humidity: 46.4%

Wind speed: 3.2m/s

Distance: 585273

Takes 7 hours 35 mins to drive

IMAGE

Tjuvholmen

12.7 km

Takes 18 mins to drive



Air Temp 18.7°C

3.6m/s

Water temp

12.4 °C

Det samme gjelder for den detaljerte visningen. Den gikk fra å ha en mer teknisk visning med ramset opp tekst til å få et mer visuelt fint design.

En ting som har vært viktig for oss når det kommer til utvikling av appen er brukervennlighet. Appen skal både være enkel å bruke og det skal være enkelt å navigere seg rundt på appen. En spesifikk ting vi har tenkt på når det kommer til brukervennlighet er plassering av de ulike knappene. Knappene du bruker til å navigere deg fra en side av appen til en annet, for eksempel fra kart til hovedsiden er plassert på bunnen av skjermen fordi det er der det er naturlig for en bruker å ha fingrene sine når man holder en mobiltelefon.

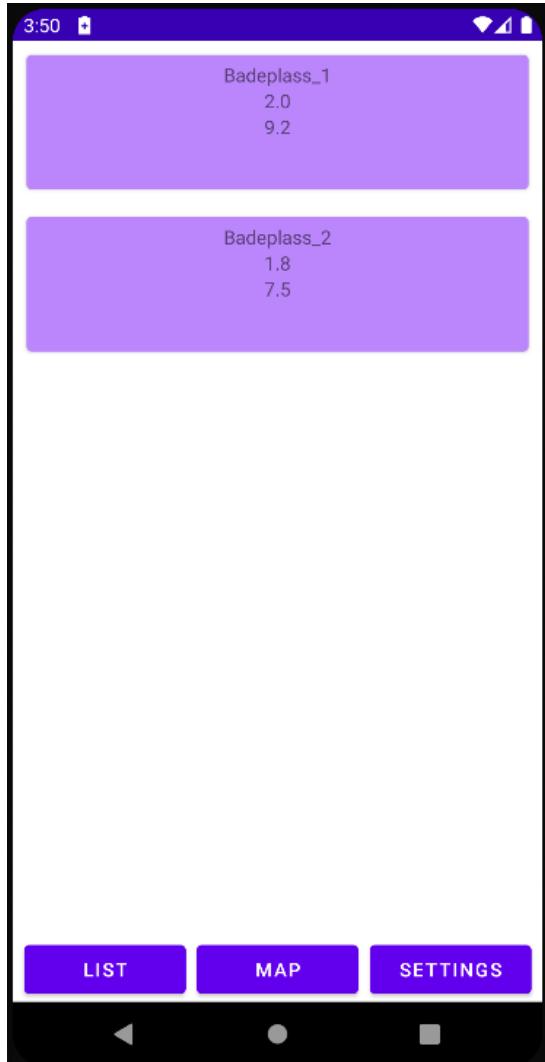
Vi har hatt møter for det meste fysisk på IFI, enten på et grupperom eller i kantina, men også noen ganger digitalt når det passet bedre for enkelte. Møtene har stort sett vart i mellom 1 og 2 timer. Vi har nok hatt litt bedre utbytte av de fysiske møtene, fordi det er lettere å kommunisere og vise fram ting til hverandre fysisk enn over teams, som er det vi har brukt når vi har snakket sammen over nettet. Når det er sagt har de digitale møtene også vært gode. Det er ikke alle som bor nærmere IFI, så det kan noen ganger være lettere å ta digitale møter, fordi da trenger man ikke bruke lang tid på å reise fram og tilbake.

Den største endringen som har blitt gjort i løpet av prosjektet var når vi skulle bytte fra activities til fragments. Vi satt på et møte da vi tegnet litt opp på tavlen og ble enige om at vi burde endre alle activities vi har til fragments. Vi så for oss at denne endringen ville gjøre appen mye bedre, på grunn av måten fragments fungerer på kontra hvordan activities fungerer. Når vi brukte activities og ville bytte mellom for eksempel kart og liste, ville hele koden for liste med API-kall og hele pakken kjøre på nytt. Når vi implementerte fragments førte dette til at appen ble mye mer smooth når man navigerer seg rundt i appen. En fragment er som nevnt ment for å kunne gjenbrukes, så i ettertid ser vi at det hadde vært lurt å starte med fragments fra starten av, på grunn av at denne prosessen tok en del tid.



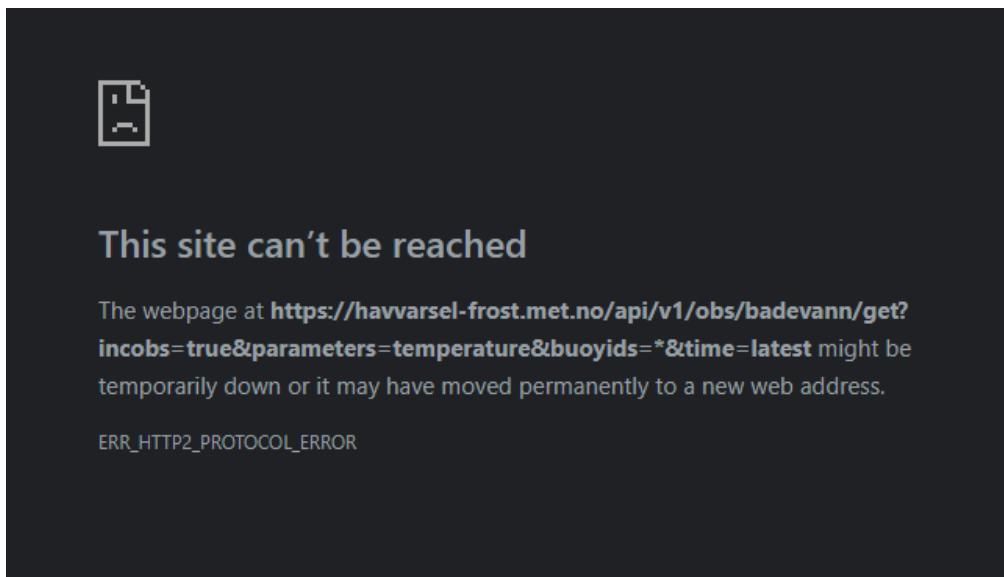
Bildet er tatt fra et møte vi hadde på et grupperom på IFI fredag 25. mars, da vi ble enige om å bytte ut activities med fragments. Fra venstre: Kerim, Eric, Daniel og Eskil. Bak kamera står Morten.

For å starte denne prosessen lagde Eric en helt ny Android Studio prosjekt slik at han kunne teste ut funksjonalitet uten å ødelegge prosjektet. Flere nye funksjoner ble lagt inn i denne testen: muligheten å trykke på cardviewen, animasjonene når man bytter fragments, og innstillings-fanen.



En skjermbilde fra testprosjektet som først implementerte fragments

Et problem vi har støtt på i løpet av prosjektet er Havvarsels-Frost APIet. Vi har i en lengre periode hatt trøbbel med å få svar på forespørslene våres. Dette har gjort at vi har hatt trøbbel med å få testet applikasjonen vår. På starten av semesteret hadde vi ikke noe trøbbel med å få svar, men cirka midtveis i prosjektet måtte vi begynne å gi API-kallet en timeout på ganske lang tid, for at det skulle være noe håp om å få svar. Vi tror ikke at vi gjort endringer som skal gjøre at problemet ligger på vår side, men vi er ikke 100% sikker på dette. Det er mulig at det er noe med implementasjonen vår som skaper trøbbel. Under er et bilde av en feilmelding vi har fått, denne gangen bare ved å teste APIet i en internett-browser.



Prosjektet har hatt jevn progresjon fra vi startet i slutten av februar fram til rundt påsken. Litt før påsken hadde alle en del oppgaver i andre fag som gjorde at prosjektet ikke hadde så bra fremgang i denne perioden. I selve påskeferien tok vi fri for å få et avbrekk fra studiene og for å lade opp til en ny periode. Etter påskeferien var vi ikke så raske med å møtes igjen, mye på grunn av en 3-ukers hjemmeeksamen i et annet fag rett rundt hjørne for de fleste av oss. I ettertid ser vi at vi burde vært bedre på å planlegge hvordan vi skulle jobbe rundt denne og andre fag. Vi ble ferdige med MVPen litt før påsken, og trodde vi skulle rekke å legge inn en god del mer funksjonalitet etter dette, men undervurderte nok litt hvor mye vi måtte jobbe med andre fag i perioden etter påskeferien. Når vi ser tilbake burde vi kanskje ha møtes litt flere ganger i startperioden der vi hadde mer tid.

På slutten av prosjektet hadde vi en sprint der planen var å ha møter hver kveld for å snakke om hva som hadde blitt gjort og veien videre. Disse møtene så vi for oss at skulle være korte, presise møter, så disse tok vi på teams. Disse var nok noen av de mer fokuserte møtene vi har hatt. Vi snakket raskt om hva som har blitt gjort, om det er noen spørsmål og ting som bør bli tatt opp, og så hva som skal gjøres videre. Det er nok noe med tidspresset (og kanskje litt stress?) som gjorde at vi måtte gire opp noen hakk. Den siste perioden har resultert i stort læreutbytte, og vi er samstemte i at innspurten skjerpet innsatsen en del og vi diskuterte mer åpent om ulike ting i appen. Dette gjorde at vi så store muligheter for å utvide funksjonaliteten.

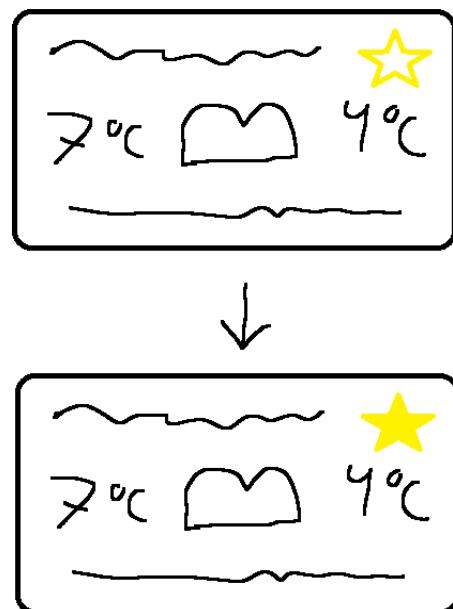
Kravspesifikasjonen og endringer

Dette er listen over ting vi hadde planer om å implementere i starten og i løpet av prosjektet:

- Systemet skal vise alle badetemperaturer i nåtid, fremtid og fortid
- Systemet skal vise stedsnavn
- Systemet skal kunne lagre brukerspesifikke data som favorittbadesteder
- Systemet skal vise reviews fra brukere av badestedene
- Systemet skal kunne opprette brukere
- Systemet skal gi brukerpoeng
- Systemet skal ha et kart med oversikt over badestedene
- Systemet skal kunne vise kart med veibeskrivelse til spesifikt badested.
- Brukeren skal kunne velge elementer som er viktig ved valg av badested, og få anbefalinger på grunnlag av disse valgene.
- Systemet skal gi hilsing etter tidspunktet på dagen
- Systemet skal respondere innen X sekunder(?)
- Systemet skal ha innstillinger
 - Dark mode
 - Colorblind mode
 - Brukeren skal kunne legge inn en standard posisjon
- Systemet skal rangere badesteder
 - Temperatur
 - Avstander
- Systemet skal vise badesteders tilleggsinformasjon.
 - Bilde av badested
 - Hvor lang tid det tar å gå / sykle / kjøre til stedet
 - Luftfuktighet
 - Kilde
 - Når temperaturen ble sist oppdatert

At systemet skal vise badetemperaturer i nåtid, fremtid og fortid er et krav vi så for oss at vi ville implementere i starten av prosjektet. Vi vil at appen vår skal brukes av folk som skal ut å bade, derfor mener vi at temperatur i nåtid er tilstrekkelig informasjon å få, ettersom temperaturen i vannet ikke vil endre seg så mye på kort tid. "Nåtid" i vår app er temperaturer som har blitt oppdatert de siste tre timene. Å få prognosør over hvordan temperatur det vil være i vannet i fremtiden kunne vært noe som brukere kunne haft nytte av dersom de vil planlegge en tur litt i forkant. Dette er noe vi ville implementert om vi hadde hatt bedre tid. Statistikk om fortiden er noe vi ikke så på som så veldig viktig å implementere, så dette punktet gikk vi vekk fra ganske tidlig i prosjektet.

Andre krav vi så for oss var at brukeren skulle kunne legge til favoritter. Dette er også et krav vi synes hadde vært gøy å få til, mye på grunn av filteret vi har på listen over badesteder. Det hadde passet veldig fint med et filter som het "favoritter", for eksempel. Et problemet vi hadde med dette er at vi da måtte få til en måte å cache de favorittene som brukeren hadde lagt til. Dette var ikke så rett-fram som vi trodde det skulle være, og ble derfor droppt. På høyre side ser vi en skisse på hvordan vi prøvd å implementere en slik funksjon. Tanken er at du har en stjerne som ikke er fylt ut, som du kan trykke på for å sette dette stedet som en favoritt, og eventuelt trykke igjen for å fjerne stedet som favoritt. Alle stedene brukeren hadde trykket på sjernen, ville blitt vist dersom brukeren valgte favorittlisten.

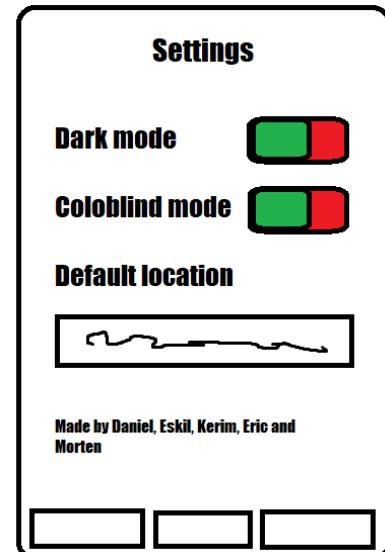


At brukere skal kunne lage sin egen bruker, samt og legge inn anmeldelser av badestedene har vi funnet ut at ikke er noe særlig nødvendig. Dette er fordi det ikke vil være nok brukere av appen til at det skal bli noe nyttig. Det samme gjelder for brukerpoeng. Alle disse er noe som ville krevd for mye tid å implementere i forhold til den lille nytten det hadde hatt. Dette er krav som vi ikke bare hadde for liten tid til å implementere, men vi fant rett og slett ut at disse ikke ville ha noe særlig nytte.

Vi hadde også en tanke om at når brukeren går inn på detaljert visning over et badested, at han/hun skulle kunne trykke på en knapp som gjør at han/hun får opp en veibeskrivelse til badestedet. Det kunne gjøres ved å sende brukeren videre til Google Maps. Dette er noe vi kanskje ville ha implementert om vi hadde hatt ekstra tid. Vi har valgt å implementere hvor lang tid det tar å kjøre til et badested, men ikke hvor lang tid det tar å gå / sykle. Vi valgte denne fordi vi tror denne er mest relevant for flest folk.

Punktet om at brukeren skal kunne legge inn egne preferanser, og få relevante badesteder på grunnlag av dette, har ikke blitt implementert. Tanken her var at en bruker kunne få opp en meny med mange forskjellige faktorer. Faktorene kunne være for eksempel vind, vanntemperatur, lufttemperatur, hvilke type vær det er osv. Alle disse kunne brukeren for eksempel rangere fra 1 til 10 i "viktighet". Da kunne en seiler eller surfer for eksempel skrive i programmet at vind er viktig, og også at sol og god temperatur er pluss. Programmet ville da sammenlignet disse verdiene med verdiene til alle badestedene, og kommet med de passende badestedene rangert.

Innstillinger er den funksjonaliteten som vi aller helst hadde ønsket å implementert dersom vi hadde mer tid. Dette var neste punkt på listen vår over gjøremål. Vi synes det er synd at dette ikke ble implementert, mye fordi dette var ment spesielt med tanke på universell utforming. Vi hadde planlagt en mørk modus med mindre skarpe kontraster som skulle være litt mer behagelig for øynene. Vi hadde også planlagt å lage en fargemodus som skulle være spesielt laget med tanke på folk som er fargeblind. Til høyre ser vi en skisse på hvordan vi så for oss at denne siden ville se ut om vi kom litt lenger på prosjektet. Default location ville være slik at brukeren kunne skrive inn en posisjon som den ville at applikasjonen skulle beregne distansene fra. Hvis du er avgårde og skal på tur med senere på dagen, kunne du for eksempel skrevet inn din hjemadresse og fått opp de nærmeste badestedene fra ditt eget hjem.



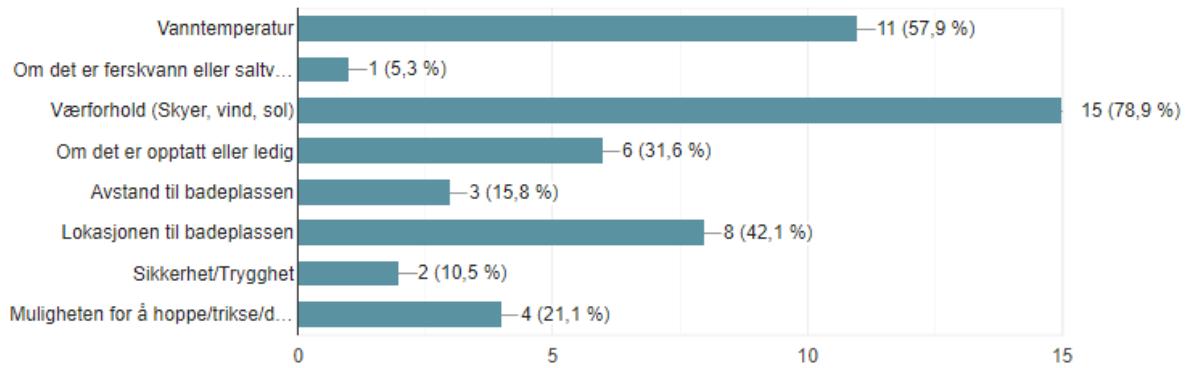
Alt i alt er det en god del vi har pratet om at vi vil implementere som vi ikke har rukket å implementere. Vi tar med oss at vår planlegging ikke har vært den aller beste, spesielt med tanke på perioden rundt og spesielt etter påsken. Hvis vi skulle ha gjentatt et slikt prosjekt burde vi helt klart jobbet enda mer i perioden der vi ikke hadde mye annet skolearbeid og eksamen. Samtidig er det viktig å få med at prosjektet har vært det første ordentlige gruppearbeidet til alle sammen av oss, og vi har lært masse av det.

Funksjonalitet som ikke er implementert:

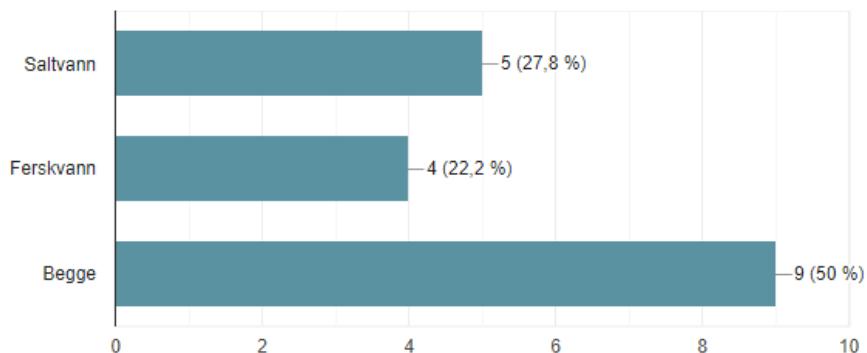
- Endre hilsningen etter tidsrommet. God morgen, god ettermiddag osv..
- Nightmode
- Sist oppdatert
- Noe som kunne illustrert temperaturen litt mer. (om det er kaldt, greit eller varm. mange har ikke forhold til hva tempen betyr.)

Brukerundersøkelse

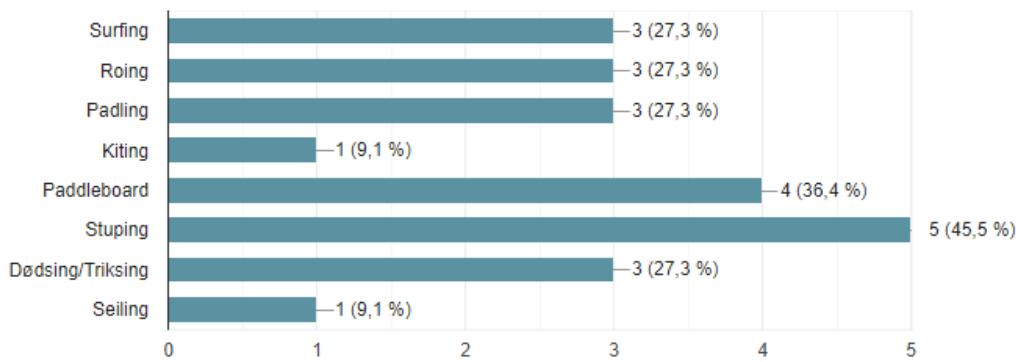
Vi lagde en brukerundersøkelse som vi ønsket å bruke slik at vi kunne få informasjon om hva folk synes er viktig informasjon å få når man skal ut på badetur. Denne undersøkelsen ble laget med ulike spørsmål som ville gi oss noen svar på hvordan vi ville lage appen vår. Det var viktig for oss å ikke bare støle på våre egne tanker om hva som var viktig å få med i appen, men også få et bredere spekter av synspunkter. Derfor tok vi en diskusjon om hvilke spørsmål vi ønsket å stille til "folket". Vi fant så noen ulike spørsmål vi ønsket å ha med i undersøkelsen, som for eksempel: "Hva er viktig for deg ved valg av badepest?" og "Er det noen andre aktiviteter du liker å gjøre på badeplassen?".



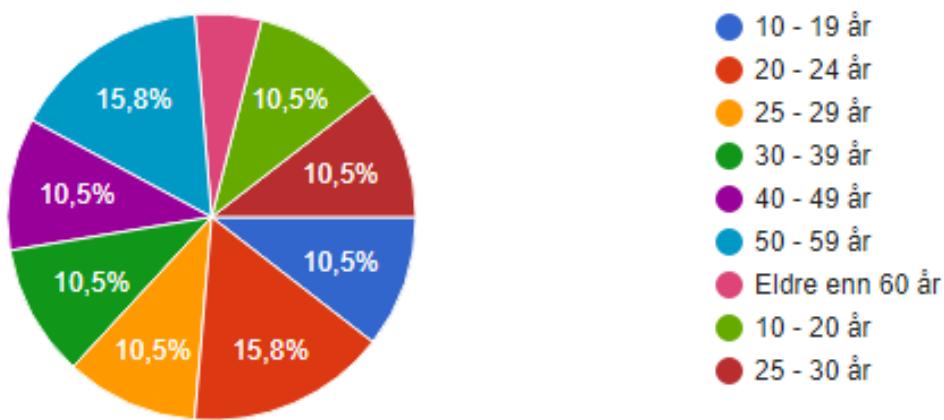
Over ser vi svarene vi fikk på spørsmålet "Hva er viktig for deg ved valg av badeplass?". Svarene vi fikk her var ikke så langt unna slik vi forventet. De aller fleste synes det mest viktige er hvor mange grader det er i vannet og hvordan været er. Med det bekreftet ønsket vi selvfølgelig å la brukerne se både vanntemperatur og hvordan været er i appen vår. Vi ser også at folk synes det er viktig med lokasjon til badeplassen. Med tanke på dette lar vi brukerne av appen vår se lokasjonene på et kart, samt viser dem hvor langt det er til de forskjellige badestedene.



Her ser vi resultatene på spørsmålet om de liker best å bade i saltvann eller ferskvann. Vi ser at det er litt blandede meninger om dette. Flesteparten liker begge, mens noen har en favoritt blant de to vanntypene. Heldigvis gir Havvarsel-Frost APIet data fra både ferskvann og saltvann, slik at de som har en favoritt kan finne et sted på kartet der de ser at det er enten ferskvann eller saltvann.



Vi stilte også spørsmål om det var noen andre vannaktiviteter som de liker å gjøre på et badested. Vi fant ut at det er veldig forskjellig hva folk liker å gjøre her, men det var flest som liker å stupe. Dette er god informasjon å få, vi kunne brukt dette til å gi brukerne et bilde over de ulike badestedene, slik at de kunne se om det var muligheter for den aktiviteten de selv synes er gøy å gjøre. For eksempel om det var et stupebrett på stedet. Dette var en av tingene vi ønsket å implementere, men vi kom aldri så langt at dette ble implementert.



Når vi fant folk til å svare på undersøkelse, hadde vi som mål å inkludere så mange som mulig. Vi ser av svarene på undersøkelsen at vi har folk som har svart i alle forskjellige aldre. Det er viktig fordi vi vil at alle skal kunne bruke appen vår, uansett om du er 12 eller 70 år. Hvis det kun hadde vært unge som hadde svart kunne man kanskje ha forventet litt annet statistikk på for eksempel spørsmålet om hva som er viktig med en badepest. Kanskje ville

triksing og hopping vært enda mer populært om vi bare hadde spurte unge, mens sikkerhet / trygghet ville hatt enda større utslag om vi bare spurte litt eldre folk.

Refleksjon

Fra starten til MVP ble ferdig

Når vi fikk oppgaven møtte vi på et problem med en gang. Dette var at det var veldig vanskelig for oss å få alle til å møte opp samtidig. Det var viktig å få et møte med alle for å bestemme hvilket case vi skulle velge, og generelt få snakket sammen om oppgaven. Dette var på grunn av sykdom hos enkelte deltakere.

Når alle fikk møtt hverandre så fikk vi bestemt case, og delt ut oppgaver. Alle var motiverte og gjorde en solid jobb for begynnelsen av prosjektet. Et av problemene som oppstod flere ganger under oppgaven var når andre ventet på noe som noen andre måtte fullføre før de

kunne jobbe. Vi opplevde flere bottlenecks og dette er et sted der vi burde ha gjort noe annerledes, selv om det ikke er en åpenbar løsning. Et forslag kan være at flere personer jobber på enkelte ting sammen på oppgaver som tar litt lengre tid, da vil samarbeidet kanskje føre til at tidsforbruket reduseres.

Kravspesifikasjon og modellerings delen gikk egentlig veldig greit, vi var alle ganske enige om hvordan det skulle se ut. Vi startet så å jobbe mot vår minimal viable product (MVP). Vi fikk implementert alt innen den tiden vi hadde sett for oss. Det var der vi begynte å snakke om å kjøre en sprint for å bli ferdig med MVP-en. Noe som det ikke ble noe av, men det løste seg greit.

Vi prøvde mange forskjellige former for smidig development, men mange av dem falt ut ganske raskt. Kanban Bordet vårt ble brukt de første ukene, men vi fant ut at det var Microsoft Teams som var mest effektivt for oss å jobbe med. Teams er en utrolig enkel måte og kommunisere sammen med en utrolig brukervennlig user interface. Vi endte opp å bruke Teams helt til slutten. Til tross for dette var kommunikasjon noen ganger sakte, spesielt fordi noen av oss hadde problemer med at Teams på mobil varslet ikke alltid om nye meldinger.

Fra påske til slutten av prosjektet

Når påskeferien sluttet tok det også utrolig lang tid før arbeidet startet igjen. Dette påvirket oss på en utrolig negativ måte med tanke på at 3 medlemmer i gruppen fikk midtveiseksamen i IN2140, og ett medlem hadde en stor IN2100 oblig. Dette var ikke en overraskelse så vi burde ha planlagt mye bedre. Det er mulig at vi ble noe overmodig etter vi viste fram en MVP som vi syntes var veldig god.

Vi prøvde flere ganger å få til en sprint men det ble aldri noe ut av, unntatt for den siste uken før innleveringsfristen. Da begynte varsellampene å lyse, og vi så at vi måtte legge inn et ekstra gir frem mot levering. Det resulterte i kanskje den mest interessante perioden i prosjektet. Ved at mange jobbet samtidig opplevde vi at “toget rullet”, noe som ga utrolig mersmak til å gjøre det så bra som mulig. Som nevnt i prosessdelen om funksjonalitet på slutten, så ble det mange ideer som vi dro frem som noe vi skulle hatt. Her opplevde vi at når

det skjer handling, så begynner kreativiteten og fokuset å bedres. Vi så mange muligheter til videreutvikling, men manglet tid for å realisere ideene.

Det er mange implementasjoner som vi hadde lyst til å legge til som vi ikke fikk tid til. Dette er på måte naturlig med tanke på tidsfristen som ble satt på oppgaven. De fleste apper lages ikke på et par måneder, men det er også på grunn av pausen vi tok. Den viktigste tingen som vi ikke fikk laget er innstillingene, som nå står tom kun med en tekst som viser hvem som har laget appen. Dette var synd, fordi vi hadde lenge planer om å implementere dette, og vi trodde vi skulle rekke det.

Andre refleksjoner

Gruppen vår ble ikke veldig påvirket av covid-19. Vi hadde muligheten til å møtes fysisk, men selv om vi måtte ha møtes digitalt så hadde dette ikke påvirket oss på en veldig negativ måte. Når man jobber fysisk så blir man mer motivert på grunn av at man ser de andre jobbe. Digitalt derimot blir mer effektivt tidsmessig med tanke på at det kutter ut reisetid for alle involvert. Ett av medlemmene våre Daniel, fikk covid 19 på 17 mai. Dette hindret oss i å møtes fysisk men forhindret oss ikke i å jobbe digitalt.

Denne oppgaven har lært oss hvordan det å jobbe i et team er i praksis. Det er mange ting som vi har lært som kommer til å gjøre fremtidige teamoppgaver mer effektive. Mange av problemene som har oppstått for oss, er problemer som absolutt kunne ha blitt unngått med litt mer erfaring. Denne erfaringen var jo en viktig del av dette faget.

Github var utrolig hjelpsomt under prosjektet vårt. Github har alt av funksjonaliteten som trengs hvis flere skal jobbe på samme prosjekt. Litt vanskelig å finne ut hvordan man skal sette opp github, men siden dette har vært en gruppeoppgave har det vært mulig å få hjelp av de andre på gruppen. Erfaringen med github er kanskje en av de største tingene man kan få ut av prosjektet.

En ting som hele gruppen er enige om er det at alle har jobbet hardt, og at ingen har jobbet urettferdig mye. Alle har gjort en innsats og man kan se arbeidet til alle i de forskjellige delene av appen. Vi skrev i team avtalen at hvis arbeidsmengden var urettferdig fordelt så

skulle vi endre den. Dette ble ikke et problem siden det virker som vi er enige på at vi greide dette.

Nå gleder vi oss til sommerferie, og alt i alt kan vi si oss fornøyd med appen, faget, samarbeidet og den nye erfaringen vi har fått. Hvis vi har planer å bade i nærheten, er det bare å åpne appen for å se om det passer best med faktor 15 eller 50, surfebrett eller luftmadrass og t-skjorte eller jakke. God sommer fra gruppe 39 :)

Kilder

- https://www.oslofjorden.com/bilde_badestrand/oslo/soerenga_sjoebad_badeplace_oslo_overikt.JPG
- <https://thumbs.dreamstime.com/z/hand-drawn-watercolor-illustration-palm-tree-tropical-home-plant-drawing-isolated-white-cartoon-style-144745859.jpg>
- https://publicdomainvectors.org/photos/Anonymous_Beachball.png
- <https://i.pinimg.com/originals/2f/aa/2a/2faa2a4bd91a004487cd9481ffd73274.png>
- <https://www.sprell.no/produktbilder/2020/flerfaget-bla-badering-fra-liewood-45-cm.jpg>