Prosjektrapport i IN2000:

Vindkraftproduksjon i Norge

Team 24:

Andreas Baardseth (andrebaa), Bendik Werp (bewerp), Carina Thanh-Tam Truong (cttruong), Mathias Demeer Strøm (mathiads), Ole-Kristian Ellingsen (okelling), Patrick P. Haukaa (patricph)

Veileder: Emilie Bauck Dynestøl (emiliedy)

Institutt for Informatikk

Universitetet i Oslo

Vår 2020

Innholdsfortegnelse

1. Presentasjon

Andreas Baardseth: 23 år og studerer programmering og systemarkitektur. Han er fra Oslo og har gått på ifi i snart to år. Har tidligere tjenestegjort som grensejeger og har en forkjærlighet for kontroll og orden.

Carina Truong: 24 år og går andre året på studieretningen programmering og systemarkitektur. (Bilde: Simen Kjellin, UiO.)

Mathias Strøm: 23 år og går andre året på Informatikk: Design, bruk og interaksjon. Han er personen som sier ifra når vi bør ta et steg tilbake og liker å jobbe med brukerundersøkelser.

Ole-Kristian Ellingsen: 30 år på 4. semester på programmering og systemarkitektur. Han er utdannet sivilingeniør fra NTNU.

Patrick P. Haukaa: 22 år og går andre året på Informatikk: programmering og systemarkitektur. Er en typisk progger som liker å ha et prosjekt å drive med.

Bendik Werp: 22 år gammel og går andre året på design, bruk og interaksjon. Han er gruppas gledesspreder, og sørger alltid for god stemning.

Team 24 består av seks medlemmer: to studenter fra design, bruk og interaksjon, og fire studenter fra programmering og systemarkitektur. Vi kom raskt igang med planlegging og startet allerede under kick-off arrangementet å avtale faste møtetidspunkter og argumenterte for og mot de ulike casene. Allerede påfølgende møte satt vi oss ned og begynte å utarbeide en teamavtale (se vedlegg 9.4) og bestemte hvilket case vi ville jobbe videre med.

Vi valgte å gå for case 6, en egendefinert oppgave. Vi ønsket å lage en app som brukte Met sitt vær-API til å beregne vindproduksjon i Norge i sanntid. Appen skal kunne beregne hvor mye strøm som produseres i hver enkelt vindpark, hvor mye som produseres til sammen, inntekter for hver vindpark og for alle sammen, og den skal kunne beregne fremtidig produksjon og inntekter. Vi tenkte at for en app som baserer seg på norske brukere vil det være naturlig at appen er på norsk.

Vi ønsker å jobbe med dette caset fordi vi mener at vindkraft er en nøkkel til å løse klimaproblemene. I det siste har vindturbiner vært mye i mediene, men ofte på grunn av negative årsaker. Vi tror at et verktøy som kan illustrere effekten vil bidra til å gjøre diskusjonen mer saklig og faktabasert.

1.1 Om *Windr*

Windr er en applikasjon som er ment for å formidle informasjon om norske vindparker. Appen inneholder data om norske parker, inkludert størrelse, lokasjon og sanntidsproduksjon. Målsetningen har vært at applikasjonen kan bidra med innsikt og føre til at en fremtidig debatt om vindkraftutbygging som baserer seg på fakta snarere enn følelser og gjetning.

Norsk vindkraft har økt kraftig de siste årene, med en dobling i årlig produksjon siden 2017 (NVE, 2020). Det har ført til at vindkraft nå står for rundt 4% av norsk kraftproduksjon (2019-tall), men også at det har dukket opp en del konflikter forbundet med planlagt og faktisk utbygging (Dahl, 2018). Etter hvert som vindkraft har økt i omfang og kontrovers, har behovet for informasjon om egenskaper og påvirkning på miljø også økt. Det skal *Windr* levere.

I appen kan bruker velge mellom en kartløsning, levert av Google, og en listevisning. Ved å trykke på ønsket park, enten i kart eller i listen, kan en bruker se mer detaljerte data om den spesifikke vindparken. Dette inkluderer lokasjon, maksproduksjon til parken (også kalt installert effekt), meldt vindstyrke i parken, beregnet sanntidsproduksjon, informasjon om antall husstander parken forsyner og reduksjon i utslipp av klimagasser i antall kg CO_2.

Informasjon om parkene er hentet fra Norges vassdrag- og energidirektorat (NVE) sitt API om Elsertifikater, bedre kjent som grønne sertifikater (NVE, 2015). Dette gir en oversikt over de parkene som er en del av denne ordningen, samt nøkkelinformasjon om disse. Ettersom at ordningen kun omfatter nye anlegg betyr det at ikke alle vindkraftanlegg er med i appen, men parkene som er i appen representerer 73% av total installert effekt til alle parker i Norge.[1]

For å hente værdata til parkene, benytter vi Meteorologisk Institutt (Met) sitt API. Dette gir en oversikt over meldt vind for gitte koordinater på timebasis. For enkelhets skyld har vi antatt at vindstyrken er konstant gjennom hele timen.

Ved hjelp av vindstyrken blir produksjon for den aktuelle timen beregnet. Her er det antatt at en vindpark ikke produserer noe hverken når det blåser mindre enn 4 m/s (på grunn av for svak vind) eller når det blåser mer enn 25 m/s (på grunn av fare for turbinen), at turbinen produserer maksimalt i hele intervallet mellom 18 m/s og 25 m/s, og at den i det resterende intervallet øker produksjonen proporsjonalt med økende vindstyrke (se figur 1.1.1). Det er også antatt at alle vindparker har identisk produksjonsprofil.

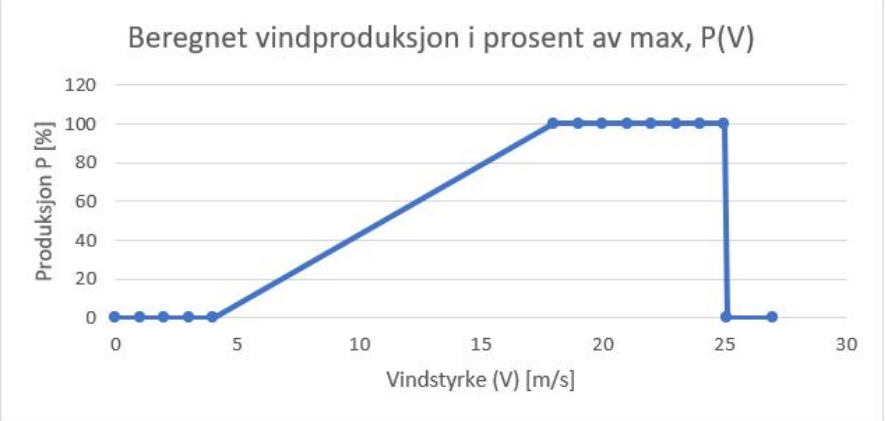
$$E = P(V) * K$$

E: produsert energi

P: andel produsert av installert effekt

V: vindstyrke

K: installert effekt



Figur 1.1.1. Graf som viser vindproduksjon.

Ved hjelp av Google sin kartløsning (Google Maps) kan *Windr* vise plassering av alle vindparkene i en kartvisning. Dette gir brukeren mulighet til å lete etter parker basert på geografisk plassering. Det viser også den geografiske konsentrasjonen til parkene, og dermed også, grovt, fordelingen av vindressursene. Bruker kan velge mellom tradisjonelt kart og satellittkart. Det tradisjonelle kartet

[1] Andelen er beregnet ved å summere total kapasitet i *Windr* og delt det på total installert effekt av vindkraft i Norge (NVE, 2020).

viser topografien til området rundt parken, mens satellittkartet viser bebyggelse i området (inkludert adkomstvei og selve vindturbinene).

Til slutt har *Windr* en søkbar liste over parkene, der man kan søke på både parknavn og kommune. Listen viser grunnleggende informasjon om parken, navn, kommune og installert effekt, og har også et ikon som illustrer hvor mye det blåser i parken på nåværende tidspunkt.

2. Brukerdokumentasjon

2.1 Designprinsipper og brukbarhet

Når man utvikler et produkt er det viktig å ha fokus på brukbarhet. Brukbarhet beskriver om appens design støtter brukeren i å utføre sine oppgaver. Designprinsippene er retningslinjer som kan være til hjelp for å designe gode og effektive brukergrensesnitt (Preece, Rogers & Sharpl, 2016, s. 25). Ved å være bevisst på disse prinsippene kan man lettere oppnå god brukbarhet, og det var derfor noe vi fokuserte på. De seks hovedprinsippene er constraints, visibility, mapping, feedback, affordance og consistency.

Consistency går ut på at brukeren kjenner igjen elementer fra andre løsninger de er kjent med fra før. Dette gjør det enklere for dem å navigere i appen og forstå hva som skal gjøres uten ekstra opplæring. Ettersom alle på gruppa var kjent med mobiløkosystemet var det naturlig for oss å hente ut elementer vi kjente fra før og bruke disse i vår app. Dette inkluderer blant annet å ha en standard navigasjonsmeny nederst i appen. I tillegg består appen av ikoner fra android biblioteket som er kjent for androidbrukere, og den oppfyller android design principles. Alle disse elementene bidrar til økt consistency. Brukerne presenteres for funksjoner og et grensesnitt som ligner på noe de har sett før, og dette kan øke appens brukbarhet.

Feedback er designprinsippet som går ut på å gi brukeren bekreftelse på om en handling er suksessfull eller ikke. Hensikten er å gjøre brukeren sikker på at en handling er gjennomført, og ikke være i tvil. Dette kan oppnås ved tilbakemeldinger til brukeren gjennom visuelle hint. I appen vår har vi derfor valgt å gjøre at knappene i menylinjen indikerer hvilken side brukeren er på, ved å lyse opp denne knappen. Videre får brukeren en tilbakemelding dersom data ikke lastes inn riktig i appen. Dette gjør at brukeren aldri er i tvil om det er noe problemer med kartet.

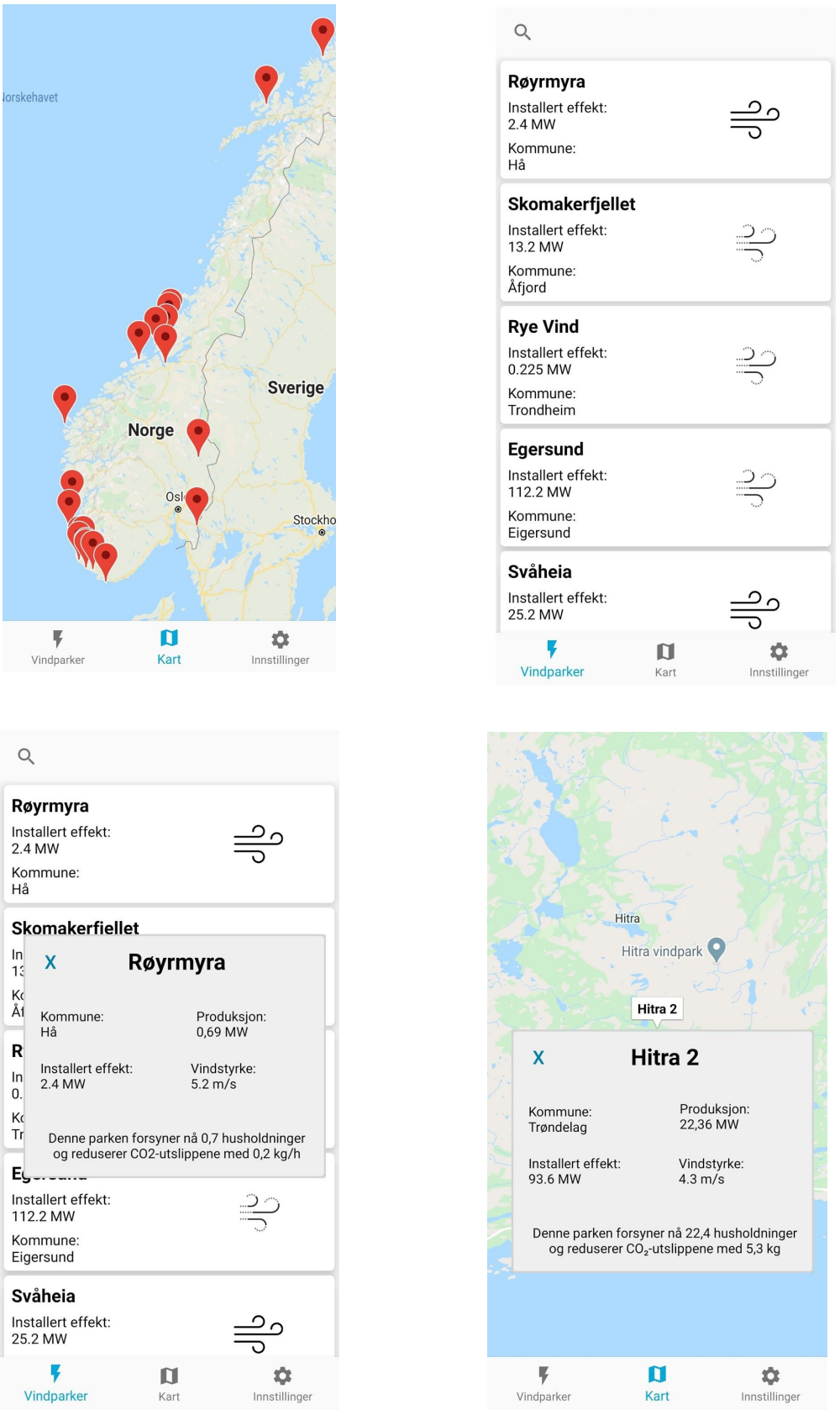
Affordance er designprinsippet som handler om å gi brukeren et inntrykk av hvordan en skal interagere med systemet og hvordan det skal oppfattes. Bruk av et kart er et eksempel på dette i vår app. Man kan intuitivt forstå at det er mulig å navigere seg rundt på et kart. Markerte steder på kartet i form av pins gir inntrykk av at det finnes noe her, og at de kan trykkes på.

2.2 Applikasjonens funksjonalitet

Hovedfunksjoner er viktige funksjoner som appen er avhengig av for å kunne utføre de oppgavene vi ønsker at den skal gjøre. *Windr* består av tre hovedfunksjoner. En av hovedfunksjonene i appen er å vise frem alle vindmølleparkene i Norge på et kart. Dette har vi valgt som en hovedfunksjon da det gir et grafisk inntrykk av hvor i landet det befinner seg vindmølleparker. Vi mener at det gir brukeren en god oversikt over parkenes posisjon, og at det er en veletablert og intuitiv måte å vise beliggenhet.

En annen hovedfunksjon i *Windr* er å vise vindmølleparkene i en liste. Her kan brukeren kjapt få oversikt over alle de ulike parkene i landet, samt kort informasjon om disse. Fordelen med å ha det på denne måten er at brukere enkelt kan finne frem til ønsket vindpark uten å vite hvor i landet den ligger. I tillegg vil brukeren her ha muligheten til å trykke på hver av de ulike parkene og se utdypende informasjon. Ved siden av hver vindpark i listen vil det også være ett av tre vindikoner som representerer vindstyrken i parken.

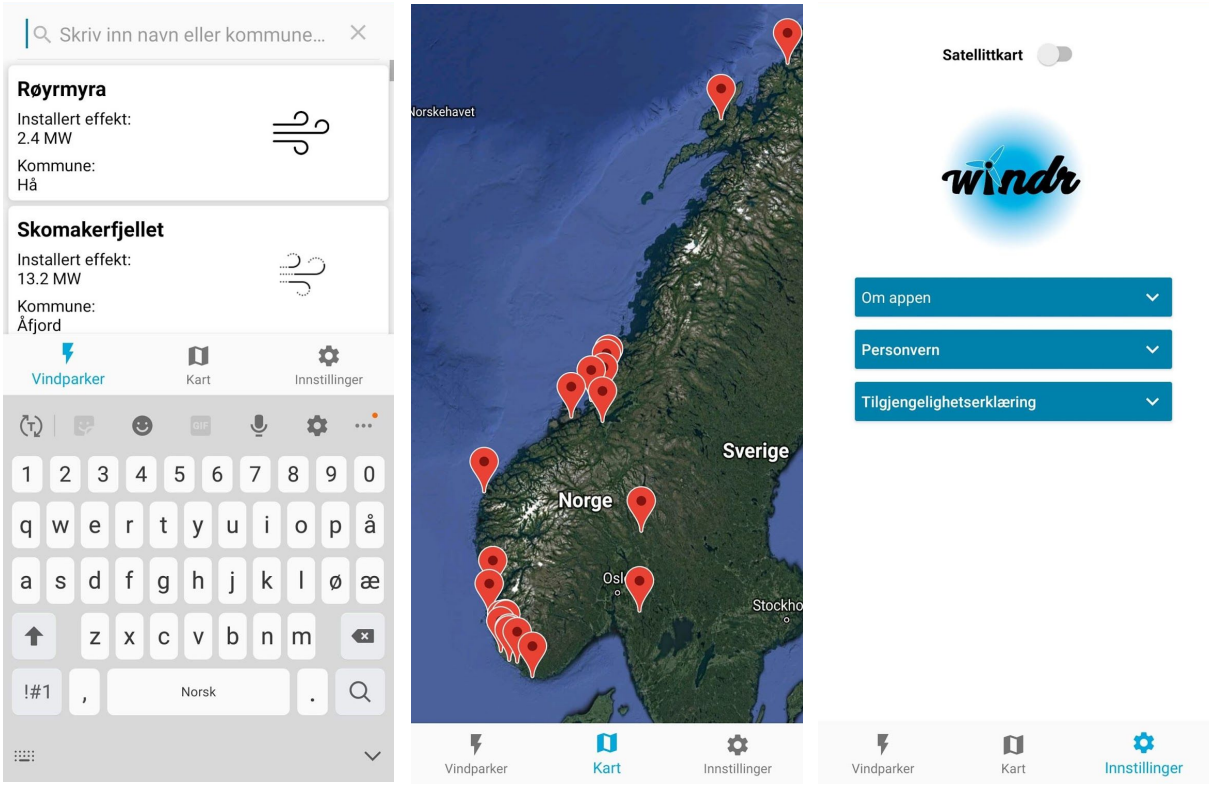
Den tredje og siste hovedfunksjonen i appen gir brukeren oversikt over de ulike dataene vi har samlet om de forskjellige parkene. Dette blir fremstilt på en oversiktlig måte i et kort. Informasjonen vises når brukeren trykker på en vindpark i listen over parkene. Under utvikling av appen har denne funksjonen vært hovedfokuset, og *Windr* er derfor bygget opp rundt dette. Informasjonen brukeren har muligheten til å se omfatter navn, beliggenhet, vindstyrke, strømproduksjon og miljøbesparelser.



Figur 2.2.1. Fra øverst til venstre og mot høyre: a. Vindmølleparkene vist på kart. b. vindmølleparkene vist som en liste. c og d. Informasjon når man trykker på en park.

I tillegg til hovedfunksjonene har vi også valgt å legge til en del tilleggsfunksjoner. Dette er funksjoner vi har implementert for at brukeren lettere skal kunne bruke og forstå appen. En av tilleggsfunksjonene vi har valgt å ha med er et søkefelt i listen over vindmølleparker. Her kan en bruker søke på navn eller kommune, og listen vil filtreres deretter. Dette er noe vi ønsker å ha fordi det gir brukeren muligheten til å raskere navigere i listen og se data om ønskede parker.

Vi har også valgt å ha med et interaktivt kart som en tilleggsfunksjonalitet. Her vil brukeren ha muligheten til å trykke på markeringen av de ulike parkene, og kartet vil justeres slik at denne parken kommer i fokus. Om brukeren igjen trykker på markeringen vil man bli vist informasjon om parken. Denne funksjonen gjør at brukeren kan finne frem informasjon om parker direkte ved å lete seg frem på kartet. Bruker må altså ikke bytte til listevisning for å finne informasjonen til ønsket park, men kan enkelt aksessere infosiden til parken direkte gjennom kartet. Ved å trykke på kartet utenom parkene vil brukeren bli sendt tilbake til den opprinnelige visningen av kartet der man ser oversikt over hele landet og alle parkene.



Figur 2.2.2. Fra venstre: søkefunksjonalitet i liste, tilleggsfunksjonalitet for å bytte kart, informasjon og innstillinger side.

Som en tredje tilleggsfunksjonalitet har vi valgt å ta med en egen innstillinger-side. På denne siden vil brukeren ha mulighet til å endre utseendet til kartet (hvilken karttype man ønsker å se). Denne siden vil også inneholde informasjon om appen, personopplysningsvern, tilgjengelighetserklæring, samt kontaktinfo for muligheten til å gi oss direkte tilbakemelding på eventuelle feil eller tilbakemeldinger om appen.

Hovedfunksjonaliteten til appen er i stor grad sentrert rundt å vise utdypende informasjon om de ulike vindmølleparkene i landet, og vise deres geografiske plassering. Tilleggsfunksjonene er implementert for å gi økt brukervennlighet.

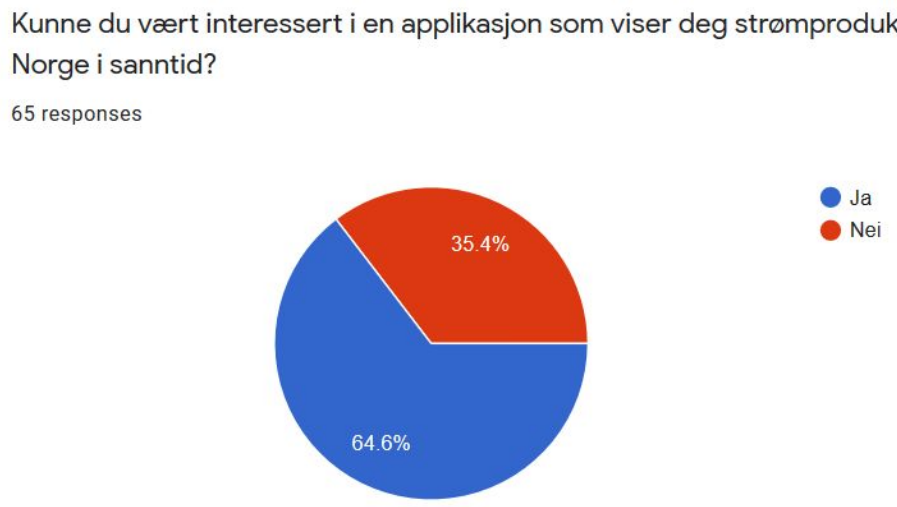
2.3 Målgruppen

I idéfasen av prosjektet bestemte vi oss for å rette applikasjonen vår mot næringslivet, og da spesielt mot bedrifter som jobber med vindkraft. Dette viste seg å by på noen problemer ved konkretisering av krav og behov, og vi endte derfor opp med en ny målgruppe rettet mot individer.

Grunnen til at vi først valgte å gå for næringslivet var fordi vi så for oss at bruksområdet til applikasjon skulle være å vise produksjon og inntekter til alle landets vindkraftanlegg i sanntid. Vi tenkte derfor at applikasjon ville ha større nytteverdi for de som jobber med, eller har spesiell interesse for, vindkraftproduksjon enn for privatpersoner.

Da vi begynte å konkretisere krav og behov til applikasjonen så vi at det kunne være vanskelig å tilby denne målgruppen noe nytt. Vi regnet med at bedriftene allerede hadde gode verktøy som kunne løse oppgavene bedre enn det vi kunne tilby med vår applikasjon, og det ville derfor ikke vært så interessant for målgruppen å være med på brukerundersøkelser. I tillegg var vår opprinnelige plan å hente strømprisen fra Nord Pool sitt API, men den koster penger og er derfor ikke aktuell å bruke i dette prosjektet.

Vi tenkte fortsatt at vindkraft var et interessant tema, og at vi kunne benytte relevante APIer fra både Met og NVE. Før vi startet på prosjektet valgte vi derfor å forhøre oss med bekjente om deres forhold til vindkraftproduksjon gjennom en spørreundersøkelse (første datainnsamling). Resultatet viste at det var en interesse i denne målgruppen.



Figur 2.3.1. Andel som er interessert i en app om vindkraftproduksjon.

Data fra undersøkelsen (figur 2.3.1) viste at det var en interesse blant deltakerne for å kunne følge med på strømproduksjon fra vindparker, og som gjorde at vi kunne rette målgruppen vekk fra næringslivet og mot privatpersoner. På bakgrunn av denne undersøkelsen bestemte vi oss derfor for å forandre på målgruppen til personer med interesse for klima og fornybar kraftproduksjon.

Formålet med en spesifikk målgruppe er å kunne målrette markedsføring og produktutvikling til de som produktet er tiltenkt. Det gjør det interessant å se på den demografiske fordelingen i målgruppen vår. En tydelig definert målgruppe kan også gi fremtidige brukerundersøkelser økt relevans ved å plukke de riktige deltakerne.

Ser vi på statistikken fra de siste valgene i Norge (Kleven, 2020) er det to grupper som skiller seg ut når de kommer til å velge miljøpartier. Disse er personer i aldersgruppen 18-29, og personer over 25 år med lengre utdanning fra universitet og høyskole. Vi ønsker derfor å fokusere på personer mellom 18 og 29 år som tar eller har tatt høyere utdanning på høyskole eller universitet. Aldersgruppen er også den som ifølge SSB (Fjørtoft, 2017) har størst kunnskap innen IKT, og dermed vil være mest tilbøyelige til å ta i bruk teknologi for å tilegne seg kunnskap. Dette fokuset er valgt for at vi lettere skal kunne finne deltakere da yngre er en stor del av vår målgruppe, men vi begrenser oss ikke kun til dem.

2.4 Brukskontekst

For å bedre forstå behovet til målgruppen er det nyttig å vite hva som er sannsynlige omstendigheter for brukeren, nærmere bestemt brukskonteksten. Basert på formålet med applikasjonen er det rimelig å anta at hovedbruk vil være som oppslagsverk, der man starter applikasjonen for å undersøke hva produksjon for sanntid er i en eller flere parker, og deretter går ut av applikasjonen.

Det er ikke lagt opp til mye interaksjon mellom applikasjon og bruker, og dataene i applikasjonen vil kun oppdatere seg når applikasjonen starter og vil være statiske under bruk. Dette forsterker antagelsen om at applikasjonen vil brukes i korte perioder av gangen.

Basert på målgruppen kan vi anta at hovedtyngden av bruk vil skje i byer, som betyr at det som regel vil være god dekning under bruk. Samtidig vil det være naturlig å anta at mange brukere vil sjekke applikasjonen når de er i eller ved en vindpark. Det er steder der det kan være dårlig dekning, og av naturlige årsaker, værhardt. En applikasjon som trenger lang tid på å starte og å hente og vise data kan gi en veldig dårlig brukeropplevelse ved slike omstendigheter.

2.5 Plattformer for *Windr*

Vi har valgt at appen skal fungere på alle enheter med Android versjon 6.0, Marshmallow (API level 23) og høyere (se kapittel 4.4 for dypere begrunnelse). Vi har også valgt å tilrettelegge slik at appen tilpasser seg og fungerer på ulike skjermstørrelser, slik at den kan brukes på forskjellige telefoner og nettbrett. For å få til dette har vi valgt å designe mye ved bruk av constraints, slik at applikasjonen selv tilpasser seg størrelsen på enheten på best mulig måte.

2.6 Aksessere *Windr*

Vi tar utgangspunkt i at applikasjonen blir godkjent gjennom Google Play Store, som er distribusjonsplattformen til Android. For at en bruker skal aksessere applikasjonen må de ha en android enhet, en bruker med tilgang til Play Store og kontinuerlig tilgang til internett. Bakgrunnen for å distribuere produktet vårt på Play Store handler i stor grad om tilgjengelighet. Vi har best mulighet for å treffe et stort antall brukere, hovedsakelig i Norge, men også i utlandet gjennom denne plattformen. Nye brukere kan bruke søkefunksjonen for å finne applikasjonen i Play Store, men det er også mulig å dele en lenke gjennom direktemeldinger, sosiale nettverk og hjemmesider.

En begrensning vi møter på med valget om Play Store som distribusjonsplattform er brukere med iOS enheter. Det er vanskelig å gjøre noe med dette umiddelbart etter at applikasjonen er utgitt med den koden som er skrevet i Kotlin, men i fremtiden vil det være mulig å bruke eksisterende kildekode som utgangspunkt for en applikasjon som støtter begge operativsystemene.

3. Kravspesifikasjon og modellering

3.1 Funksjonelle krav og kravspesifikasjon

En kravspesifikasjon gjøres for å få konkrete holdepunkter for hva appen skal inneholde slik at man best mulig kan dekke brukernes behov. Disse behovene har kommet fram både gjennom datainnsamlinger, men også gjennom det som var formulert i casebeskrivelsen. På bakgrunn av disse behovene satt vi opp brukerhistorier, og formulerte kravene for applikasjonen vår. Kravene deles opp i funksjonelle og ikke-funksjonelle krav. For å stille de riktige kravene er det er viktig å danne en god forståelse av brukergruppen, samt de andre aktørene som inngår i systemet.

De funksjonelle kravene som stilles til applikasjonen skisserer hva *Windr* må gjøre fra et funksjonelt ståsted. Kravene deles i tre forskjellige deler, henholdsvis forretnings-, bruker- og systemkrav. (Lindsjørn, 2019) Vi ser på instituttets krav og retningslinjer til appen og koden som forretningskravene i dette prosjektet. Brukerkravene er utviklet over tid fra begynnelsen av prosjektet gjennom hjelp fra brukerundersøkelser for å sikre at brukerens interaksjon med appen dekker brukerens behov. Disse kravene gjenspeiler de funksjonene som flest potensielle brukere har etterlyst i applikasjonen, og er resultat av brukerundersøkelser og analyse. Systemkravene bestemmer hva appen skal kunne fra et teknisk ståsted og er mer tekniske krav som illustrerer teknologien bak i større grad enn de andre kravene. Alle disse kravene utgjør fokuspunktene vi som utviklere må ha når vi implementerer funksjonalitet og skriver kode, og er en del av kontrakten med produkteier (Lindsjørn, 2019).

3.2 De viktigste funksjonelle kravene

Systemet skal vise alle vindmølleparker på kart

Dette kravet er et brukerkrav og beskriver hvordan appen skal kunne vise et grafisk kart hvor alle vindparkene er plassert. Dette er viktig for å gi brukeren en rask og enkel oversikt over hvor vindparker befinner seg i Norge. Kartet skal fungere som en slags hjemskjerm og være den første skjermen brukeren kommer til når de åpner appen. Tanken om å vise vindmølleparkene på et kart var noe vi så for oss helt fra starten ettersom dette er en intuitiv måte å fremstille geografisk lokasjon, og som vi tror brukerne våre er vant med fra lignende applikasjoner. Vi fikk også bekreftet at kartvisning var en ønsket funksjonalitet gjennom den første datainnsamlingen.

Systemet skal vise spesifikk informasjon om hver enkelt vindmøllepark

Dette kravet er et brukerkrav og beskriver hvordan appen skal kunne vise mer spesifikk informasjon om hver enkelt vindpark til brukeren. Informasjonen som vises vil ikke være veldig teknisk, da vi fikk vite at dette ikke var ønskelig gjennom første datainnsamling. Informasjon som skal vises vil være info om vindparkens navn, beliggenhet og størrelse.

Systemet skal vise vindstyrke og produksjon i sanntid

Dette kravet er et brukerkrav og beskriver hvordan appen skal vise informasjon om vindstyrke og produksjon i sanntid. Dette var et logisk krav å ha med for oss fordi vindstyrke er den viktigste parameteren for vindkraftverk og er det som avgjør parkenes produksjon. Det er også viktig å vise energiproduksjonen til brukeren fordi produksjonen er selve formålet med parken, dersom det ikke blir produsert noe energi vil det heller ikke være noen hensikt ved å bygge vindparker. Vi tenkte at denne informasjonen burde vises i sanntid for å gjøre appen mer dynamisk og gi brukeren mulighet til å se hvordan parken gjør det akkurat nå.

Systemet skal vise energiproduksjon på en forståelig og håndfast form

Dette er et brukerkrav og beskriver hvordan appen skal vise energiproduksjonen på en forståelig og håndfast form. Det betyr at appen ikke bare skal vise strømproduksjon i Watt, men også på et mer relaterbart format, som for eksempel antall lyspærer eller husholdninger som kan drives med denne produksjonen. Dette kravet ble lagt til basert på funn fra første datainnsamling.

Systemet skal hente informasjon om vindparkene fra API

Dette kravet er et systemkrav og beskriver hvordan appen skal hente informasjonen om vindparkene som vises i appen fra et API (Elsertifikater fra NVE). Dette kravet ble laget med tanke på vedlikehold og videreutvikling av appen. Ved å hente informasjonen fra et API vil det være enkelt å fjerne og legge til vindparker eller informasjon ved et senere tidspunkt.

3.3 Use-case

Figur 3.3.1. Et use case diagram over funksjonalitetene i Windr.

Figur 3.3.1 viser alle interaksjonene brukeren kan gjøre i applikasjonen, relasjonen mellom brukeren og bruksområdene, og hvilke tredjeparter som er involvert i de forskjellige områdene. Under er det en rekke skriftlige beskrivelser av de forskjellige områdene i diagrammet.

Vise informasjon om vindpark

Prebetingelse:

Oppdatert informasjon om alle vindparker er hentet

Oppdatert værdata for alle koordinater er hentet

Parkene vises på kartet

Postbetingelse:

Applikasjon viser kort med detaljert informasjon om park

Applikasjon viser kart

Aktør:

Bruker

Hovedflyt:

1. Bruker trykker på markør til ønsket vindpark
2. Systemet henter relevante data fra parkobjekt
3. Systemet viser informasjon i pop-up kort
4. Bruker lukker kort

Alternative flyt 1, steg 4 - Bruker ønsker å se detaljert informasjon

A.1.1 Brukers trykker for å se detaljert informasjon om parken

A.1.2 System henter tilgjengelige data fra parkobjekt

A.1.3 Bruker sendes til recycler view der informasjonen vises

Alternative flyt 2, steg 2 - Data mangler for aktuell park

A.2.1 Parkobjekt mangler innhold

A.2.2 Pop-up kort viser feilmelding der dataene mangler

A.2.3 Bruker lukker kort

Sjekke vindstyrke og produksjon

Prebetingelse:

Applikasjon har hentet parkdata

Applikasjon har hentet vinddata

Postbetingelse:

Card view og kart har oppdaterte vind- og produksjonsdata

Ugyldige verdier gir feilmelding

Aktør:

Bruker

Hovedflyt:

1. Systemet henter informasjon om vindstyrke og installert effekt til park
2. Systemet sjekker om vind er under minimum eller over maksimum for produksjon
3. Systemet regner ut produksjon
4. Systemet lagrer informasjonen i vindpark-objekt
5. Systemet sender signal om at kartvisning og card view oppdateres
6. Systemer viser vindstyrke og produksjon til brukeren

Alternativ flyt 1, steg 3 - Ugyldige verdier i beregningen (installert effekt er 0 MW)

A.1.1 Systemet returnerer at produksjon er 0 MW

Se vedlegg 9.1 for resten av use-casene.

3.4 Aktører

De forskjellige gruppene som har en rolle i applikasjonen deles opp i aktører og interessenter. Disse identifiseres med hensikt om å bedre forstå brukerkonteksten og hvem som blir påvirket av prosjektet. Vi bruker disse i plandrevet utvikling for å konkretisere den enkeltes rolle i applikasjonen og prosjektet. Dette hjelper oss å identifisere ansvarsområdene og hensikten deres, som vi som utviklere må ta hensyn til.

Aktører	**Ansvarsområde**	**Interesse**
Sluttbruker	Bruke applikasjonen og informasjonen.	Ha en effektiv, intuitiv og nyttig applikasjon for å hente ønsket informasjon.
Kunde (IFI)	Overse og sette krav til utviklingen.	Presentere en pålitelig og troverdig produkt til brukerne.
Utvikler	Utvikle applikasjonen på en slik måte at alle interessenter/aktører har best mulige forutsetninger for å dekke sine interesser.	Ha et ferdigstilt produkt som kan selges.
API-eier	Levere APIer med relevant informasjon.	Formidle korrekt informasjon på korrekt format til bruker og programvare.

Interessenter		
Vedlikeholder	Vedlikeholde kildekoden og utformingen.	At appen er robust og kjører som forventet over tid.
Politikere	Drive Norge AS hensiktsmessig, effektivt og i henhold til internasjonale forpliktelser.	Markedsføre at Norges miljøvennlige forpliktelser oppfylles.
Vindkraftverk	Produsere miljøvennlig energi til Norge og utland.	Selge energi til strømnett og kunder, formidle produksjon i sanntid.

Figur 3.4.1. Tabell over aktører og interessenter.

3.5 Ikke-funksjonelle krav

De ikke-funksjonelle kravene er krav som beskriver de ønskede kvaliteter ved produktet. Disse deles inn i hovedkategoriene produktkrav, organisatoriske krav og eksterne krav (Sommerville, 2011). De mest relevante kravene for vårt prosjekt omfatter områder som utviklingskrav, brukerkrav og sikkerhetskrav. Vi så på kvaliteter vi ønsket at applikasjonen vår skulle ha og satt opp en liste med de viktigste ikke-funksjonelle kravene.

Universelt utformet

Dette kravet er et brukerkrav og beskriver hvordan vi ønsker at applikasjonen skal være utformet for et bredt spekter med brukere med ulike forutsetninger. Dette vil vi oppnå ved å følge retningslinjene i WCAG 2.1 for universell utforming. Disse prinsippene er følgende; mulig å oppfatte, mulig å betjene, forståelig og robust (DIFI, 2020). Dette sikrer at alle brukere føler seg komfortabel i samhandling med grensesnittet i *Windr.* Appen skal derfor oppfylle alle suksesskriterier med nivå A.

Høy brukbarhet

Dette kravet er et brukerkrav og beskriver hvordan vi ønsker at nye brukere skal hente ut informasjon av applikasjonen på en enkel og intuitiv måte. For å oppnå dette bruker vi Googles material design principles som er grunnmuren for android økosystemet. Vi vil også prøve å unngå overflødig funksjonalitet i grensesnittet, samt være bevisst på å gi kontinuerlig feedback til brukeren om hva som skjer i applikasjonen. Dette sørger for rask læringsgrad hos nye brukere og høy

brukbarhet. Vi ønsker derfor at 90% av nye brukere skal kunne finne informasjon om en tilfeldig vindpark på under 30 sekunder.

Høy tilgjengelighet

Dette kravet er et utviklingskrav og beskriver hvordan vi ønsker at appen skal være utviklet på en måte som støtter flest mulig brukere. For å oppnå dette skal appen kunne kjøre på ulike enheter med ulik skjermstørrelse, og være tilgjengelig på de mest brukte Android versjonene i Norge. *Windr* skal være tilgjengelig på minst 85% av alle Android telefoner i Norge.

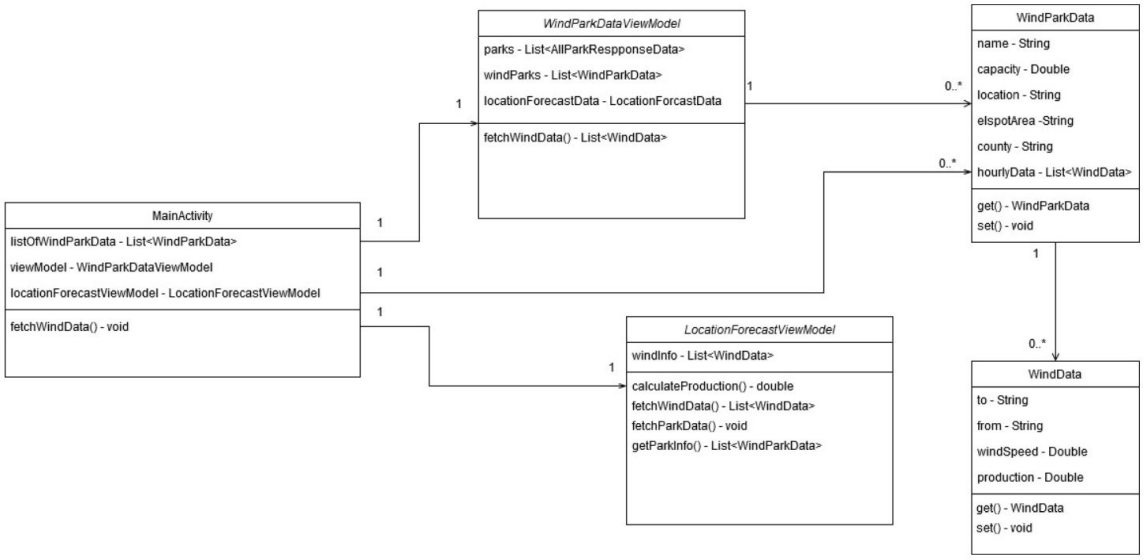
Rask respons

Dette kravet er et utviklingskrav og beskriver hvordan vi ønsker at innlasting av data skal skje relativt raskt for å støtte varierende nettverkshastighet og hardware hos brukeren. Derfor skal appen med all informasjon være lastet inn på under 2 sekunder ved 80% av tilfeller.

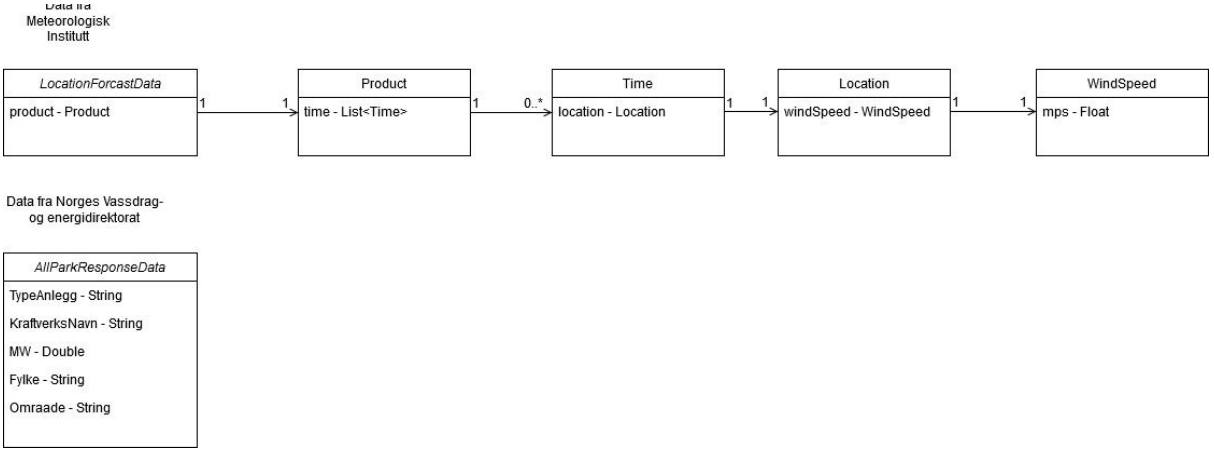
Høy trygget

Dette kravet er et sikkerhetskrav og beskriver hvordan vi samler og behandler brukerdata. Vi ønsker at alle brukere skal føle seg trygge i interaksjonen med *Windr*. Vi velger derfor å ikke skjule informasjon bak en innloggingsmur eller lagre personopplysninger. Dette er heller ikke nødvendig ettersom data vi henter er fra et åpent API.

3.6 Klassediagrammer



Figur 3.6.1. Klassediagram av viewmodel.



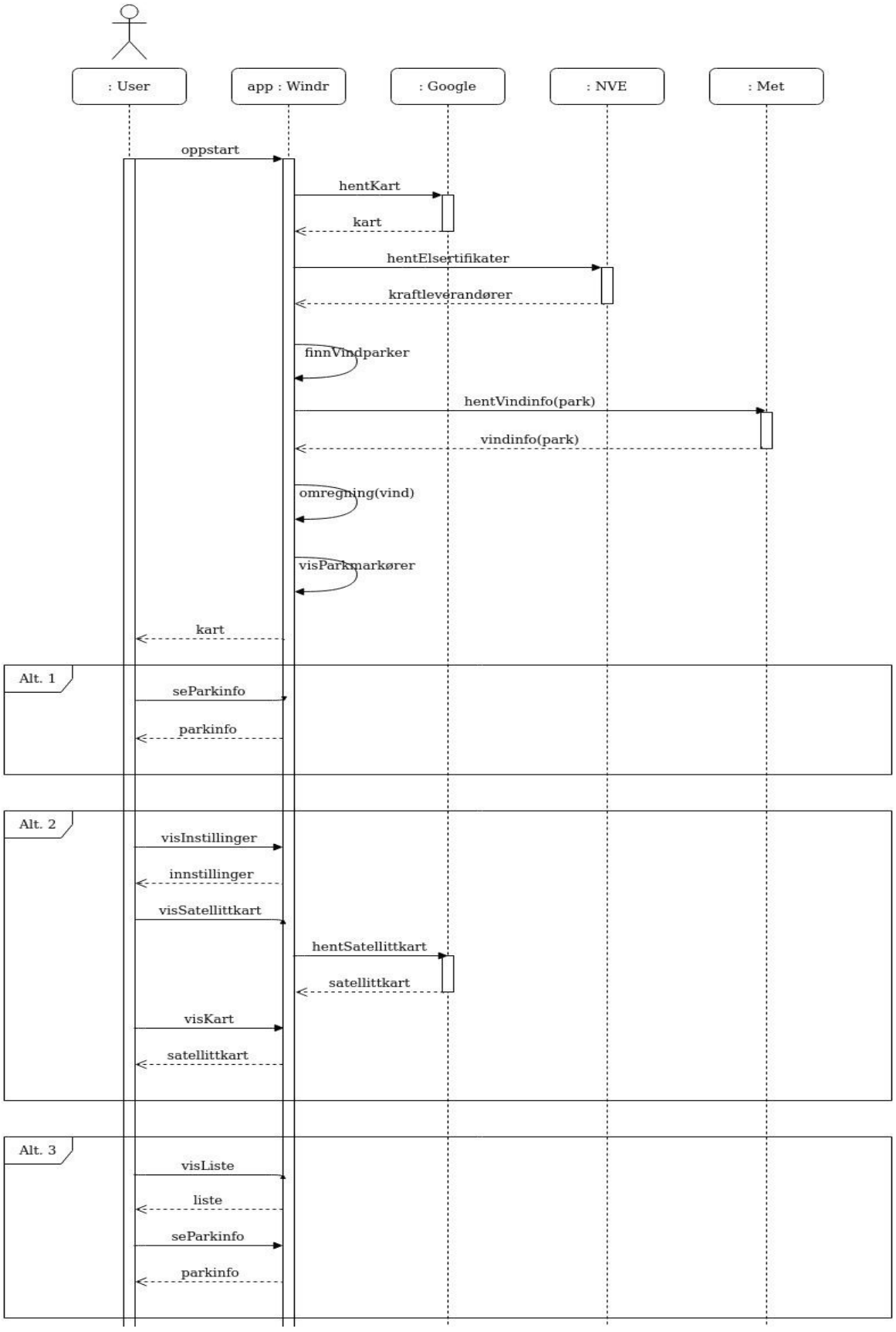
Figur 3.6.2. Klassediagram av APIer.

3.7 Sekvensdiagrammer

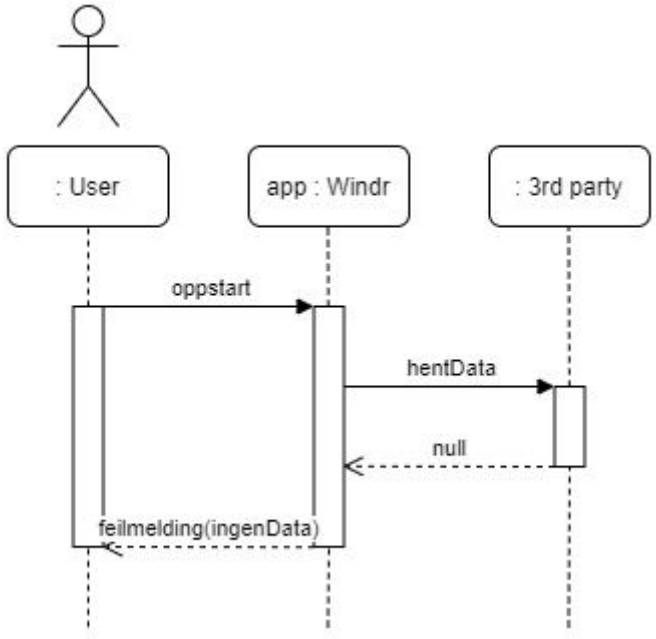
Figur 3.7.1 viser et enkelt sekvensdiagram om hvordan vi ønsker at appen skal fungere. Ved oppstart av appen vil programmet hente inn all informasjon om vindkraftverk og vindstyrke først før brukeren kan interagere med systemet. Dette er gjort med hensyn til både API-leverandørene og brukeropplevelsen.

For at systemet skal belaste serverne til API-leverandørene minst mulig er det best å hente inn all nødvendig informasjon først. Sidene dataene heller ikke endrer seg så ofte er det også mulig for oss å gjøre alle API-kallene øyeblikkelig ved oppstart. Dette fører også igjen til at brukerne får en mer sømløs opplevelse siden de slipper å vente på data.

Figur 3.7.1 tar derimot ikke for seg hva som kan skje ved eventuelle feil. Den viktigste feilen som kan oppstå i applikasjonen og som vi må ta høyde for er at kallene til Google, NVE, eller Met ikke gir noe resultat, eller at den ikke finner APIet. Siden dette er samme feilen som kan skje alle tre steder, vil det også føre til samme oppførsel av programmet og tilbakemelding til brukeren. Figur 3.7.2 illustrerer denne sekvensen når et kall til en tredjepart ikke går som forventet.



Figur 3.7.1. En enkel sekvensdiagram som viser samhandlingen mellom objekter når alt går som forventet. Alternativer med feilhåndtering er ikke tatt med.



Figur 3.7.2. En enkel sekvensdiagram som viser tilbakemeldingen til bruker hvis noe feil oppstår ved kall til tredjeparter.

3.8 Kobling og kohesjon

Lav kobling og høy kohesjon er prinsipper for objektorientert design som har blitt fulgt fra starten av prosjektet. Lav kobling beskrives som et mål på hvor sterkt et objekt er knyttet til andre objekter (Runde, 2017). Det betyr at alle objekter skal ha et begrenset antall avhengigheter, og heller ikke avhenge av mange objekter. Dette vil bidra til bedre lesbarhet, vedlikeholdbarhet og gjenbrukbarhet. Det vil også føre til at applikasjonen får bedre effektivitet, samt høyere robusthet og pålitelighet. Få avhengigheter gjør at deler av applikasjonen kan fungere, selv om enkelte moduler ikke fungerer. Dette vil være viktig for både brukere og utviklere. For brukeren vil det være en fordel at applikasjonen fortsatt opprettholder enkelte funksjonaliteter. For utviklere vil det bli lettere å feilsøke og rette feil fordi feilen kan isoleres til en mindre del av appen.

Vi ønsker også høy kohesjon i applikasjonen vår. Høy kohesjon er beskrevet som objekter som har moderat ansvar og utfører et begrenset antall oppgaver innenfor ett funksjonelt område (Runde, 2017). Dette har vi prøvd å ivareta i appen vår ved at ingen objekter skal ha for mye ansvar, og skal ikke alene kunne gjøre at applikasjonen feiler ved svikt i en enkelt modul. Vi har vektlagt dette for å få en elegant app med veldefinerte ansvarsområder til de ulike objektene. I tillegg vil det gjøre appen

utvidbar og enkel å forstå. Prinsippet om høy kohesjon vil også kunne bidra til høyere robusthet og pålitelighet.

Begge disse prinsippene ble lagt stor vekt på under utviklingen av applikasjonen vår. Før vi startet programmeringen laget vi et dokument om hvordan vi ønsket at kodestilen vår skulle være, der disse punktene var noe av hovedfokuset for hvordan vi skulle utforme applikasjonen. Dette gjorde vi fordi det er punkter som er viktige å definere og ha i tankene allerede før utviklingen begynner, slik at applikasjonen fra starten av opprettholder prinsippene og det er mindre sannsynlig at dette er noe som vil kunne føre til problemer senere.

3.9 MVVM/MVP

For å få en god og oversiktlig arkitektur i applikasjonen er det vanlig å skille mellom brukergrensesnittet og modell. Brukergrensesnittet, viewet, viser data til brukeren og tar imot input. Viewet er gjerne ressurskrevende, og på håndholdte enheter, med potensielt mange applikasjoner som kjører i parallell, er det nødvendig å frigjøre disse ressursene når det er mulig.

Modellen representerer den persistente delen av applikasjonen. Den inneholder data som applikasjonen bruker, samt logikken og verktøyene applikasjonen trenger for å bearbeide dataene. I modellen ligger all data som ikke bør gå tapt når applikasjonens tilstand endres. Dataklasser, API-kall og databaseinteraksjoner er eksempler på komponenter som bør ligge i modellen. Dette er deler som kan være tidkrevende å hente eller konstruere hvis applikasjonen av forskjellige grunner har mistet ressurser, og ved uheldig arkitektur kan bruker oppleve at applikasjonen ikke flyter godt.

For å skille modellen fra viewet anbefaler Android at man lar en del av programmet håndtere interaksjonen mellom view og modell. Dette omtales gjerne som en presenter eller viewmodel. Ideen her er at viewet styrer viewmodellen, og viewmodellen styrer modellen. Samtidig varsler modellen viewmodellen når det skjer endringer i dataene, viewmodellen varsler viewet om det samme, og viewet sørger for at informasjonen brukeren ser til enhver tid representerer modellen.

Android har implementert viewmodellen med en livssyklus for å gjøre det enklere for utviklere å behandle lagring av data og tilstand. Livssyklusen lar applikasjonen håndtere syv forskjellige tilstander som er oppstart, start, fortsett, pause, stopp, restart og avslutt i modellen. Vi som utviklere kan selv velge hvilke av disse tilstandene som skal implementeres, men ved å ta høyde for flere tilfeller under utvikling kan man få en bedre brukeropplevelse.

Under utvikling av appen har det blitt lagt vekt på at innlasting av data skal gå hurtig under oppstart og at brukeropplevelsen skal være best mulig. Testing av applikasjonen viste at applikasjonen startet hurtig og at dataene ble lastet inn uten merkbare forsinkelser. Da dataene appen bruker i stor grad er statiske, og av lite omfang, ble det vurdert at det var lite behov for omfattende implementering av livssyklus.

I *Windr* har prioriteten vært å skille viewet fra dataen og modellen. Det er blitt implementert ved at det er laget en ViewModel-klasse for hvert av API'ene vi har hentet data fra. Lagring av data er lagt til egne dataklasser, der formålet til enhver tid har vært å minimere lagring av ubrukt informasjon. Siden disse dataene er såpass grunnleggende for appens funksjonalitet har det vært viktig at klassene er oversiktlige og enkle å bruke, og at de også vil kunne benyttes dersom kravspesifikasjonen endrer seg på et senere tidspunkt. For å holde dataklassene så enkle som mulig ble det også besluttet å beholde noe av app-logikken og databehandlingen i ViewModel-klassene. Vi vurderte det som et akseptabelt kompromiss ettersom det er lite omfattende og utføres i parallell med datainnhenting.

4. Produktdokumentasjon:

4.1 *Windr*

Windr er skrevet i programmeringsspråket Kotlin, med unntak av enkelte biblioteker tatt i bruk som originalt er skrevet i Java. XML er blitt brukt ved utforming av layout. Vi har brukt utviklerverktøyet Android Studio versjon 3.x som IDE ettersom dette er den offisielle IDEen til Android-utvikling. Som en naturlig følge av dette har vi brukt Gradle til å bygge prosjektet vårt.

4.2 Bruk av API og beregning av produksjonsdata

4.2.1 Værdata

For å kunne si noe om produksjonen i vindparkene er det avgjørende å ha vinddata for de aktuelle parkene. Ved hjelp av parken sine GPS-koordinater kan appen hente info i i Met sitt Locationforecast-API. Dette APIet gir værdata for spesifiserte tidsintervaller fremover i tid, fra og med neste hele time og ti dager frem i tid.

For å hente vinddata bruker *Windr* Retrofit-biblioteket, mens Moshi brukes for å konvertere data til Kotlin-objekter (LocationForecastData.kt). På grunn av oppbyggingen til JSON-objektet blir vinddataen lagret på en lite hensiktsmessig måte, med data-klasser i dataklasser. Derfor konverteres informasjonen til en ny dataklasse som er skreddersydd appen sitt formål (WindData.kt).

WindData-klassen inneholder start- og sluttid for vindprognosen og vindstyrke. I tillegg brukes vindstyrken til å beregne teoretisk produksjon ved opprettelsen av objektet, og lagres som en instansvariabel. Hele værmeldingen, for en park, lagres som en ArrayList<WindData> i parkobjektene, der hvert element i listen representerer vinddata for et unikt tidsintervall.

4.2.2 Parkdata

For å hente data om vindparkene benytter appen NVE sitt API Elsertifikater. Dette APIet tilbyr data om alle norske kraftverk som omfattes av Elsertifikatordningen. Appen bruker data om parknavn, hjemkommune til parken, kraftverktype (e.g vann eller vind) og installert størrelse. APIet lar ikke bruker søke på kraftverk etter type, som betyr at appen må hente data til alle kraftverk, for å så filtrere bort de som ikke er vindparker.

For å hente dataene benytter appen Retrofit-biblioteket og Moshi for å konvertere JSON-objektet til et dataobjekt (AllParkResponsData.kt). Disse dataobjektene behandles for å fjerne de anleggene som ikke er vindparker, og til slutt lagres dataen i objekter av en vind-dataklasse (WindParkData.kt) som inneholder de dataene appen trenger.

Ved videre utvikling av appen bør det vurderes å endre metode for å hente data om vindparkene da NVE sitt API har flere begrensninger. Mangel på søkemuligheter i GET-request gjør at appen må utføre et ekstra steg for å filtrere bort uønskede data. Det er også inkonsekvente data, blant annet i formattering av kommune og fylkesnavn.

Videre mangler APIet data om GPS-koordinater, som er lagt inn i en egen dataklasse, ParkCoordinates.kt, og fullstendig oversikt over alle vindparkene i Norge. Det siste er spesielt akutt siden Elsertifikatordningen kun gjelder anlegg bygget til og med 2021 (NVE, 2015) og det vil dermed ikke legges inn nye kraftverk bygget etter det tidspunktet.

Siden det kan være utfordrende å finne eksterne API som oppfyller alle behovene appen har kan det være nødvendig å håndtere dette selv. Det anbefales derfor å opprette en database over vindparker. Det vil gjøre det mulig å holde appen oppdatert med en komplett liste over vindparker, samt legge til data som applikasjonen trenger, men som ikke er tilgjengelig i APIet til NVE i dag.

4.3 Frontend - bruk av Google Maps og layout

Applikasjonens frontend blir presentert til brukeren ved bruk av ulike views fra Android sitt com.android.support bibliotek i forskjellige activities. Når vi først startet utviklingen ble det diskutert om det ville være hensiktsmessig å presentere frontenden ved hjelp av activities eller fragments. Det vi kom frem til var at forskjellen mellom disse to i en applikasjon som vår, der vi laster inn og bruker veldig lite forskjellig data, ville være minimal. Fordelen med å bruke fragments over activities er hovedsakelig for å spare ressurser, da man kan vise flere views i én activity, og derfor kunne gjenbruke mye av den samme informasjonen i flere views.

I applikasjonen vår valgte vi å bruke activities, mest fordi dette var noe vi var kjent med fra før, og ville gjøre prosessen enklere med tanke på tidsbruk. Det å bruke samme informasjonen gjennom hele applikasjonen løste vi ved å ha et globalt array med vindparker, der all informasjonen vi trengte ble liggende. Dette gjorde at vi ikke fikk noen ulemper ved å bruke activities over fragments.

Applikasjonen starter først med et view der kun applikasjonens logo vises, for så å bli sendt videre til hovedsiden der vi viser kartet. Applikasjonen har tre sider en kan velge mellom som blir representert på en menylinje nederst på skjermen.

På siden som viser kartet blir kartet vist som et fragment med et Google Maps kart, og vi har brukt Google sitt Maps API for å vise kartet. På kartet vises det markører for hver vindmøllepark vi har hentet fra NVE sitt API. Hver markør kan her trykkes på, og det vil da vises et cardview på skjermen som viser resten av informasjonen vi har hentet om den gjeldende vindmølleparken. De to andre sidene vi har i applikasjonen er en side for å vise en liste over parkene, og siden for innstillinger og generell informasjon om applikasjonen.

Listen over vindmølleparker er representert som et recyclerview i en constraintlayout. Dette har vi gjort for å kunne vise en større liste over parker, og samtidig hele tiden ha et søkefelt over listen, og menyen nederst. Søkefeltet her er et searchview, og blir brukt til å filtrere listen. I recyclerviewet har vi valgt å ha et cardview for å presentere hver park. Her vises enkel informasjon om parken, og man har også mulighet til å trykke på hvert enkelt kort for å få presentert mer informasjon, akkurat som på kartsiden.

Den siste siden, innstillinger, blir presentert som et scrollview. Dette viewet inneholder en switch der man kan velge om man ønsker å ha satellittkart eller vanlig, logoen til applikasjonen, og tre knapper. Disse tre knappene er for å vise ulik informasjon, og når en trykker på hver av de vises den aktuelle informasjonen. Grunnen til at vi har valgt å gå for et scrollview er for å ha muligheten til å vise mye informasjon under hver knapp, samtidig som siden tilpasser seg dette.

4.4 Manglende implementasjon

Underveis i prosessen har det vært mange ting vi har ønsket å implementere, men vi har dessverre ikke fått til alt vi har ønsket. Dette gjelder enkelte funksjoner i applikasjonen, men også noen metoder som gjør applikasjonen lettere å bruke, og hindrer den fra å avslutte under noen omstendigheter.

Den viktigste metoden her, som vi naturligvis ville ha implementert dersom vi hadde hatt mer tid til utvikling, er en sikring mot at applikasjonen avsluttes dersom den ikke har internett. Appen vår er som tidligere nevnt avhengig av internett til både henting av kart og data fra API, og vil derfor heller ikke ha noen funksjon for en bruker dersom en ikke har internett. Av denne grunn har vi valgt å utvikle med utgangspunkt i at brukeren ved bruk av appen har tilgang til internett. Dette har igjen

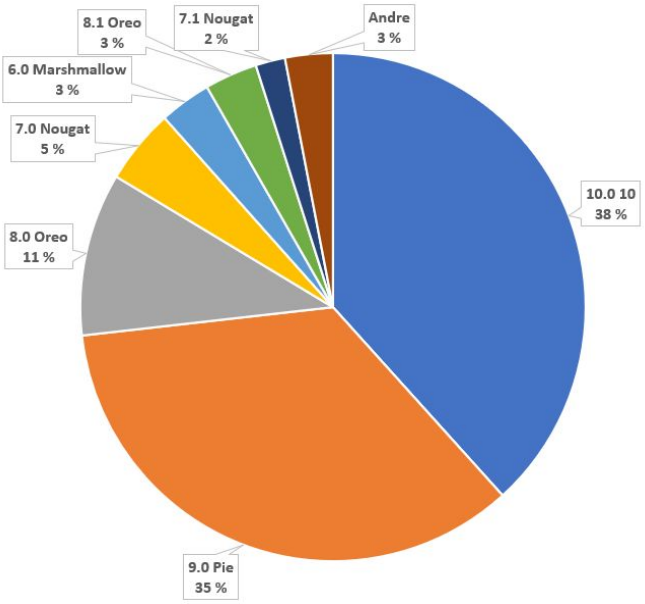
ført til dårlig håndtering av unntakstilstander som vi har avdekket gjennom runder med testing. Ved videre utvikling av applikasjonen er dette noe vi ville lagt stor vekt på å implementere.

Når det gjelder funksjoner vi gjerne skulle ha implementert er språk en av de vi setter høyt. Det at en applikasjon har mulighet til å endre språk etter språket brukeren har på telefonen fører til økt brukervennlighet og en bredere brukergruppe. Men ettersom det å implementere flere språk er en tidkrevende prosess, og applikasjonen vår hovedsakelig omhandler Norge og norske brukere, er dette en funksjon vi har valgt å nedprioritere slik appen er nå.

Vi har også valgt å nedprioritere en oppdateringsknapp. Dette er er en funksjon som ville gitt en bruker mulighet til å oppdatere data manuelt, eller etter å ha hatt applikasjonen i bakgrunnen. Dette er kun en mindre funksjonalitet vi tror ikke hadde blitt brukt så veldig mye, ettersom applikasjonen kun er ment for enkle oppslag og mindre bruk over lengre tidsperioder.

4.5 API-level

Etter å ha undersøkt fordelingen av ulike androidversjoner har vi valgt API-nivå 23. Det vil si at appen fungerer på alle enheter med android versjon 6.0 Marshmallow og nyere. Som sett på fordelingen av versjonene blant brukerne ser vi at ved å velge dette API-nivået kan appen kjøre på 96.9% av alle enheter med android (tall hentet fra Statcounter). Grunnen til at vi ikke valgte et høyere nivå var at vi ikke var avhengig av noen spesifikk funksjonalitet fra disse nyere versjonene av Android, og derfor ønsket å nå så stort publikum som mulig. For å kunne implementere nyere funksjoner senere valgte vi å ikke ha et API-nivå som var lavere.



Figur 4.5.1. Fordeling av brukere på forskjellige android versjoner.

5. Testdokumentasjon

Testing er et viktig aspekt ved en vellykket utviklingsprosess. Ved å ha fokus på testing helt fra starten av prosessen kan man lettere sørge for at koden er robust og minimere risikoen for at noe ikke fungerer som forventet. Det kan gjerne være krevende å teste kode på slutten av prosessen på grunn av omfanget til koden. Det gjør også at det kan være vanskeligere å feilsøke koden. Vi har derfor prøvd å være bevisste på å teste funksjonalitet hver for seg, samt sørge for at koden fungerer sammen. Dette ble gjort gjennom enhetstesting, i tillegg ble koden testet som helhet i en egen development branch før den ble godkjent til master (se kapittel 6.4.1 for mer om Git-rutiner).

5.1 Enhetstesting

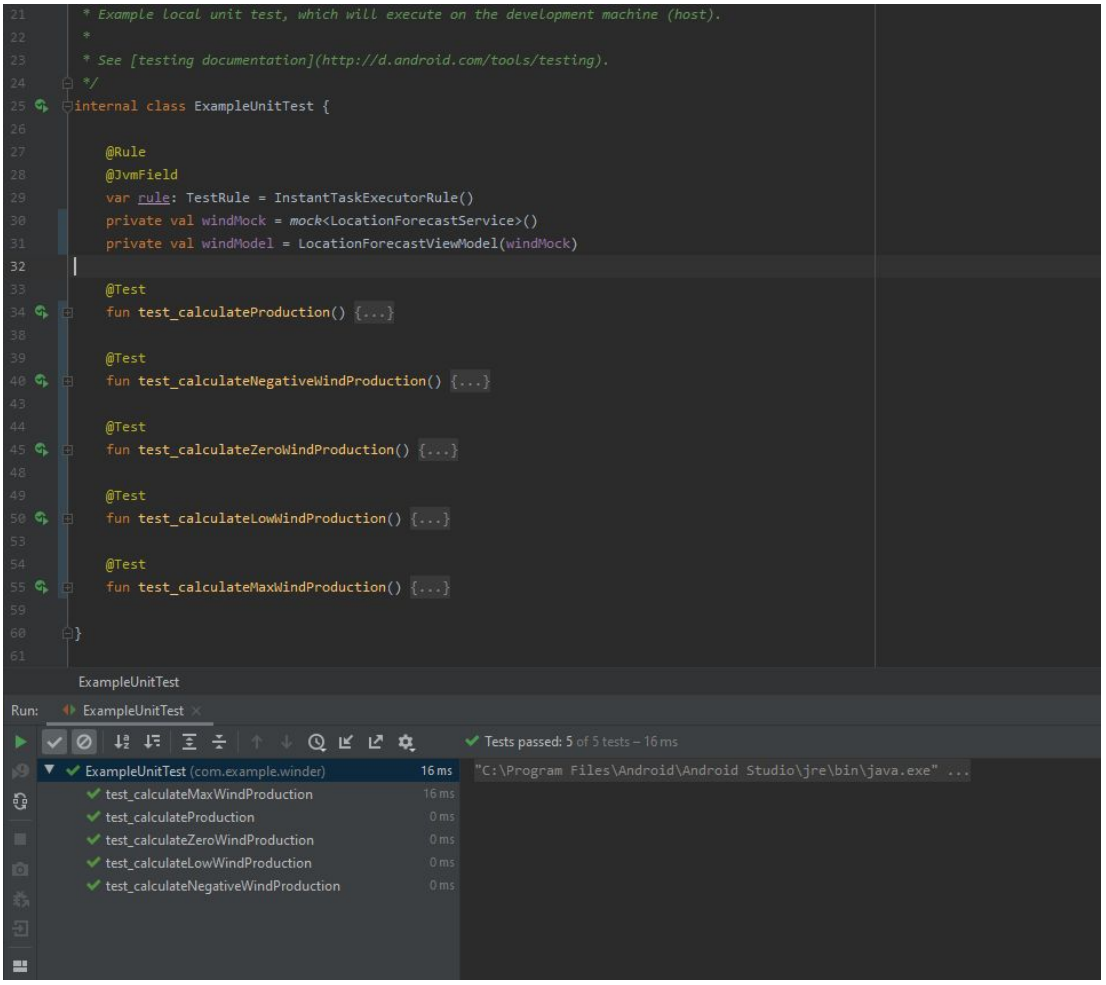
Gjennomføring av enhetstester på de minste komponentene i kildekoden til prosjektet har vært essensielt for å gjennomføre oppgaven. Som et sammensatt team vil naturligvis koden som prosjektet bygger på stamme fra en rekke forskjellige personer som ikke nødvendigvis tenker eller utvikler på samme måte. For å avdekke feil som kan være potensielt ødeleggende for sluttresultatet har vi under hele utviklingsprosessen gjennomført disse mindre testene i den hensikt å sikre at ny funksjonalitet fungerer i henhold til produktkravene (Sjoli, 2019).

For å teste hvordan kart og koordinater virker i applikasjonen har vi i større grad måttet benytte manuell testing. Målet med denne testen var å avdekke om posisjonene til parkene ble vist korrekt. Koordinatene har ikke vært tilgjengelige for oss gjennom bruk av offentlige, ikke-betalte API, og har dermed blitt lagt til som ressurser i kildekoden. Resultatet av dette er at potensielt nye vindkraftparker ikke automatisk vil bli hentet inn i applikasjon, og derfor ikke vil vises feilaktig eller med mangler. Dette er naturligvis et kompromiss som vi ikke har ønsket å gjøre, men som har vært uunngåelig.

I testøyemed ble det derimot noe lettere å gjennomføre testing av komponenten. Google Maps tillegget som viser markører for hver park på kartet bruker et WGS 84 koordinatsystem, dette er kritisk for å vise frem korrekt plassering av parkene. Manuell testing av koordinatene ble gjennomført ved at vi sammenliknet koordinatene fra NVE sine nettsider med koordinatene i Google Maps. Det viste seg at posisjonene vi fant på NVE sine sider måtte korrigeres til WGS 84 formatet for å få riktig plassering på kartet.

Søkefunksjonene er det eneste stedet i *Windr* hvor brukeren kan gi input til systemet. Uforventet brukerinput er ofte en årsak til at et system feiler. Vi gjennomførte derfor enkle tester av søkefunksjonene som gikk ut på å skrive inn mange forskjellige input som ikke skal gi noe resultat. Dette inkluderte emojis, spesialtegn og tall. Testingen viste ingen åpenbare svakheter ved søkefunksjonen.

For å teste beregningene av produksjon brukte vi det innebygde testmiljøet i Android Studio. Målet med denne testingen var å bekrefte at funksjonen for å gjøre beregningene ga det forventede resultatet. Dette ble gjort med assertEquals-metoden i Kotlin. Det innebygde testmiljøet er et godt verktøy for denne typen tester.



Figur 5.1.1. Skjermbilde av testmiljøet i Android Studio.

5.2 Ikke-funksjonelle krav

Et av hovedpoengene med ikke-funksjonelle krav er at de skal være formulert slik at de er mulige å teste eller etterprøve. Ved å ha tydelige suksesskriterier kan man forsikre seg om at appen holder

standarden kunden forventer. Kravene kan ettergås på forskjellige måter. Der enkelte kan sjekkes forholdsvis enkelt, for eksempel ved spesifisering av programmeringsspråk, kan andre kreve store brukertester for å kontrollere at kravene er oppnådd. Det siste gjelder blant annet krav om at en app skal være enkel å bruke.

Til dette prosjektet er det spesifisert en rekke ikke-funksjonelle krav. Enkelte er satt av kursledelsen, andre er satt av teamet selv. I dette delkapittelet blir bare krav som kan testes eller kontrolleres gjennomgått. På grunn av corona-situasjonen har det ikke latt seg gjøre å teste alle kravene slik som vi ønsket.

Et av kravene teamet har satt er at appen skal oppfylle WCAG-standarden til nivå A. For å etterleve dette kravet har teamet jobbet seg gjennom kravene (Digitaliseringsdirektoratet, 2020) og gjort de endringene som har vært nødvendige. Dette inkluderer å øke kontraster, tekstlig beskrivelse av ikke-tekstlig innhold, hint til søkefunksjonen og listevisning som et alternativ til kartvisning. Ikke alle nivå A kravene ble oppfylt, teamet rakk blant annet ikke å implementere dynamisk skriftstørrelse.

Et annet ikke-funksjonelt krav var høy trygghet. Dette er overholdt ved å ikke lagre informasjon om brukeren. Tidlig i utviklingsfasen ble det vurdert å bruke både brukers plassering og la brukeren velge en eller flere favorittparker. Teamet konkluderte med at utbytte av disse funksjonene ikke ville veie opp for ulempene det kunne medføre med tanke på brukerne sitt personvern. Som en konsekvens inneholder ikke *Windr* noe identifiserbar informasjon om brukeren.

De resterende ikke-funksjonelle kravene går i stor grad på brukeropplevelse og responstid, som krever større brukbarhetstester (se kapittel 6.7), eller er organisatoriske. Kursledelsen har bestemt både programmeringsspråk og arbeidsmetode. Appen er kodet i Kotlin, og oppfyller dermed det kravet. Det ble også stilt krav om at appen skulle være tilgjengelig på majoriteten av Android-enheter. Dette er oppfylt (se kapittel 4.5). Når det gjelder arbeidsmetode ble det bestemt at alle grupper skulle arbeide smidig, altså inkrementelt og iterativt. Dette vil bli beskrevet i detalj i neste kapittel.

6. Prosessdokumentasjon

6.1 Smidig utvikling

Formålet med dette prosjektet har vært å benytte smidig utvikling i et selvstendig team for å produsere en full funksjonell app. Omfanget av prosessen har vært fra idé-stadie til testing og dokumentasjon av det ferdige produktet. I tillegg har kontinuerlig dokumentering og refleksjon rundt prosess og progresjon vært viktig. Ettersom smidig utvikling har vært en forutsetning for prosjektet har dette hatt et sterkt fokus. Dette innebærer at teamet har ønsket å benytte iterativ utvikling og smidige praksiser.

Iterativ utvikling er et motstykke til den mer tradisjonelle, plandrevne utviklingen. Produktet utvikles gjennom en serie utviklingsblokker, gjerne kalt sprinter, der programvaren alltid er fungerende etter hver sprint. Målet er at en sprint skal legge til ny funksjonalitet, og i løpet av utviklingsløpet går programmet fra å kun ha helt grunnleggende funksjonalitet til det endelige produktet.

Tanken bak smidig utvikling er at det skal være lettere å gjøre endringer underveis dersom nye behov dukker opp, eller antatte behov faller bort. Det smidige manifestet (Kent et al, 2001) fremhever endringsvilje og fokus på kunden sitt behov fremfor detaljert planlegging. Målet er at det endelige produktet skal dekke hele, og bare, kunden sitt behov på kortest mulig tid.

6.1.1 Metodikk

To av de vanligste smidige metodene er Kanban og Scrum. I Scrum planlegger man sprinter der en gitt mengde funksjonalitet skal utvikles. Sprinten har en forhåndsbestemt lengde og starter og slutter uavhengig av om oppgavene er fullført. Scrum kalles derfor gjerne en tidsboksdrevet prosess.

Sentralt i Scrum er sprintmøtene, med sprintplanlegging (sprint planning), sprintavsluting (sprint review), retrospektiv og daglig standup. I planleggings- og avsluttingsmøtene henholdsvis planlegger man og oppsummerer sprinten. I et retrospektiv jobber man med å forbedre teamarbeidet og i en standup kan man holde teammedlemmer oppdatert og koble sammen utviklere som kan bidra til hverandres arbeid.

Kanban har mye til felles med Scrum, men er i motsetning det vi kaller oppgavedrevet. Det betyr at en sprint sin lengde defineres av oppgaven, gjerne kalt WIP for engelske Work in Progress, snarere

enn tiden den oppgaven skal ta. Sprinten er ferdig når de valgte oppgavene er ferdige. Sentralt i Kanban er prinsippet om å ha få WIPs med rask leveranse.

6.1.2 Smidig i praksis

I vårt prosjekt ønsket vi å benytte de elementene som passet oss, og prosjektet, snarere enn å låse oss til én rigid modell. Derfor kombinerte vi de praksisene vi ville bruke fra modellene til vår egen metode. Det resulterte i en modell som var tungt basert på Scrum. Vi benyttet oss av tidsboksdrevet utvikling, da det gjorde det enklere å planlegge fremdrift i prosjektet. De fire smidige møtene ble sentrale, spesielt etter at coronasituasjonen gjorde kommunikasjon ansikt-til-ansikt umulig.

Fra Kanban var vårt viktigste element prinsippet om å holde antall aktive oppgaver, WIP, så lavt som mulig. Spesifikt innebar det at de fleste sprinter hadde tre til fire uavhengige WIP som ble fordelt på gruppen. For å organisere og ha oversikt over oppgavene ble det laget en prosjekttavle, et sentralt element i både Scrum og Kanban, som ble oppdatert kontinuerlig. Mellom sprintene ble nye WIP valgt ut. Underveis i sprintene ble tavlen brukt til å vise progresjon for den enkelte oppgaven.

Hele poenget med utviklingsprosessen i dette emnet var å jobbe med smidige praksiser og å følge smidige metoder. For å få en oversikt over om måten vi har jobbet på tilfredsstiller kravene som stilles til en smidig utviklingsprosess, har vi satt opp en tabell. I denne tabellen tar vi for oss de ulike prinsippene fra det smidige manifestet (Kent et al, 2001) og ser på hvordan vi har jobbet i forhold til disse.

Prinsipp	**Tiltak**	**Suksess**	**Kommentar**
Tidlige og verdifulle leveranser	Tidlig og detaljert tidsplan for leveranse av appens funksjonalitet	nei	Teamet evnet ikke å følge tidsplanen - manglende funksjonalitet langt ut i prosjektet.
Omfavn endringer gjennom hele prosessen	Passe på at alle får med seg alle endringer (møtetidspunkter, krav, fokuseringspunkt osv.)	ja	Få endringer gjennom prosessen, men de endringene har vært godt forankret.
Hyppige leveranser av	Dele opp utviklingsløpet i	ja	Hadde tydelig mål for

programvare	sprinter med tydelige mål for hver sprint.		hver sprint, men noe bredt fokus.
La utviklere samarbeide med forretningsdelen	N/A	N/A	
Bygg teamet rundt motiverte utviklere med støtten de trenger	Tydelig avklaring av ambisjoner tidlig i utviklingen.	nei	Synkende motivasjon grunnet situasjon, og begrenset tilgang på veiledning.
Ansikt til ansikt er beste form for kommunikasjon	Møte 4 ganger i uken, enten fysisk eller med web-kamera.	ja	Corona gjorde fysiske møter umulige, men bruk av web-kamera har kompensert.
Fungerende programvare er primært mål på fremdrift	Ha koding som en del av oppgavene i hver sprint, og sjekke koden ved slutten av hver sprint.	delvis	Enkelte sprinter har det kun vært fokus på rapport.
Smidige metoder fremmer god utvikling. Etterstreb jevnt utviklingstempo	Fordele oppgaver likt mellom teamet, og sørge for at alle får gjort sin del ved hjelp av standup.	delvis	Har vært avbrytende med endringer i studiesituasjon, eksamen etc.
Kontinuerlig fokus på høy kvalitet på design og produkt	La andre se gjennom det man selv har gjort.	delvis	Mye i rapporten, ikke like mye i koden.
Maksimer mengden arbeid som ikke må gjøres	Unngå å legge til funksjonalitet som ikke er etterspurt.	ja	
Best produkt kommer fra selvstyrt team	Fordele oppgaver til teammedlemmer med tillit til at de blir gjort.	ja	

Vurder prosessen med jevne mellomrom	Retrospektive møter etter hver sprint.	ja	Fullført retrospektive etter hver sprint, og utført planlagte tiltak.

Figur 6.1.2.1. Smidige prinsipper i praksis.

6.2 Iterasjonene

I planleggingsprosessen bestemte vi oss for å legge opp prosjektet i 2-ukers sprinter. Vi mente at dette ville være tilstrekkelig tid til å få gjort mye arbeid både med rapport og kode. Før hver sprint gjennomførte vi en sprint planning. Her satt vi oss mål og definerte oppgaver som skulle være fullført ved sprintslutt. Vi lagde en prosjekttavle for både oppgaver knyttet til rapport og en med oppgaver knyttet til applikasjonen. I hver sprint tildelte vi gruppens medlemmer ulike oppgaver, og for å følge opp progresjonen til gruppen gjennomførte vi standupmøter flere ganger i uken.

På slutten av hver sprint gjennomførte vi både et sprint review og en sprint retrospektiv. Formålet med sprint review er å vise produkteier produktet og hva som er blitt gjort i sprinten. Deretter er det opp til produkteier om resultatet kan brukes eller om det må jobbes videre med. Vi gjennomførte en slags versjon av dette ved at vi så på alt arbeidet som var blitt gjort i sprinten, og tok deretter en vurdering på om dette tilfredsstilte de målene vi hadde satt oss for iterasjonen.

Sprint retrospektiv ble også gjennomført ved slutten av hver iterasjon. Her fikk hvert medlem i gruppen fortelle litt om hva vedkommende syntes fungerte bra i sprinten, og hva som kunne trengt forbedring. Vi noterte opp både positive og negative aspekter slik at vi kunne passe på å fortsette med det positive. Samtidig innførte vi tiltak for å forbedre punktene vi manglet litt på. En utfordring vi hadde i den første sprinten gikk ut på flytende arbeidsoppgaver, og medlemmer uten spesifikke oppgaver.

Etter den første retrospektiven innførte vi tiltak for å forbedre progresjonen. Vi tildelte spesifikke oppgaver til alle på gruppen, og opprettet en egen kanban tavle for rapport. Vi merket allerede i den påfølgende iterasjonen at moralen økte. Grunnen til dette var at det var mye enklere å sette seg ned å jobbe med noe spesifikt. Samtidig var det enklere å kartlegge hva hvert enkelt gruppemedlem hadde gjort fra dag til dag.

Alle møtene ble dokumentert i en egen prosjektlogg. Dette var slik at vi kunne holde oversikt over agenda og tidspunkt for kommende møte, samt ha notater fra tidligere møter. I starten rullerte vi på

å være referent fra møte til møte, men etter hvert fant vi ut at det fungerte bedre at vi hadde en fast referent. Det var også en av teammedlemmene som var møteleder for hvert møte. Denne personens rolle var å forberede agenda, og sørge for at vi fikk gjort det vi skulle på møtet. Først byttet denne rollen fra møte til møte, men vi ble ganske raskt enige om at det var mer effektivt om rollen rullerte fra uke til uke.

6.2.1 Endring av sprintvarighet

Med omtrent en måned igjen til leveringsfrist så vi oss nødt til å gjøre en endring i iterasjonene våre. Etter en lengre periode med sprinter på to uker innså vi at det hadde noen fordeler, men også sine ulemper. Under sprintplanleggingen var det vanskelig å tildele en riktig arbeidsmengde som tilsvarte to ukers arbeid. En konsekvens av dette var at samtlige på gruppen endte opp med mindre arbeidsoppgaver enn det man kan forvente i løpet av to uker. Vi innså dermed at det ville være mer hensiktsmessig å gå over til sprinter på en uke. På denne måten var det enklere å tildele en rimelig arbeidsmengde til gruppemedlemmene. Samtidig var det enklere å kontrollere at oppgavene ble gjennomført i løpet av uken. Oppgaver vi tidligere hadde brukt to uker på å gjennomføre ble nå ferdig på halvparten av tiden. Overgangen fra to til en ukes sprinter var derfor et veldig lønnsomt tiltak, og det hjalp oss i den smidige arbeidsprosessen.

6.3 Fasene i prosjektet

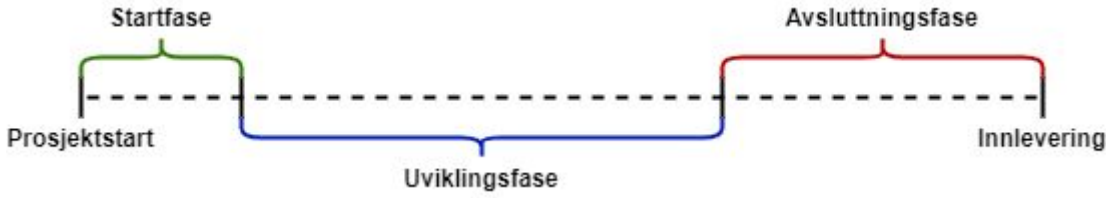
Prosjektet kan deles inn forskjellige faser basert på hva som var fokus på ulike tidspunkter i prosjektet. Vi regner de første to ukene etter at vi fikk prosjektoppgaven som oppstartsfasen. I denne fasen var fokuset å planlegge hvordan vi skulle gjennomføre prosjektet på en best mulig måte. Vi hadde møter hvor vi diskuterte hvordan arbeidsflyten skulle være. På dette tidspunktet i prosjektet utformet vi en teamavtale som alle signerte (se vedlegg 9.4).

Vi regner også valg av case som en del av denne fasen. Vi diskuterte ulike muligheter ved casene og kom med forslag. Etter at vi hadde valgt case ble mye tid brukt på idémyldring. Her var fokuset å komme med forslag til funksjonalitet til appen, hvordan vi ville at appen skulle se ut, og hvilken målgruppe vi skulle rette oss mot. I denne fasen satte vi oss inn i Github, og opprettet brancher. Vi bestemte oss også for hvilke verktøy vi ønsket å bruke. Etter omtrent to uker følte vi at vi hadde nok grunnlag for appen til å gå videre til utviklingsfasen.

Utviklingsfasen var den lengste fasen i prosjektet. Den startet idet vi begynte å skrive kode og varte i nærmere seks uker. Først begynte teamet å kode den grunnleggende funksjonaliteten. Målet var å

ha en fungerende app så fort som mulig. Dette ville gi oss et utgangspunkt å bygge på, noe vi kunne videreutvikle og forbedre. Videre i utviklingsfasen ble det lagt til ny funksjonalitet for å dekke de ulike kravene. I tillegg ble det gjort visuelle endringer og mindre justeringer av den funksjonaliteten som allerede var laget. Det var også i denne fasen av prosjektet vi begynte å skrive på rapporten. Vi så det som svært viktig å dokumentere alt som ble gjort fortløpende slik at det var ferskt i minnet.

I den siste avsluttende fasen hadde vi fokus på testing og ferdigstilling av appen og rapporten. Dette var omtrent de fire siste ukene av prosjektet. Vi var hele tiden bevisst på at testing ville bli et viktig aspekt. Vi satt derfor av tid i denne fasen for å jobbe med testing av applikasjonen. Vi skrev enhetstester og diskuterte hvordan det var hensiktsmessig å teste de ulike delene av koden. I den avsluttende fasen brukte vi også tid på å gjennomføre brukertester. Dette gjorde vi for å validere om det vi hadde laget faktisk dekket brukerens behov på ønsket måte, samt om de ikke-funksjonelle brukerkravene ble tilfredsstilt. I denne fasen gikk det også mye tid til å ferdigstille rapporten.



Figur 6.3.1. Fasene i prosjektet.

6.4 Verktøy

Under planlegging og gjennomføring av prosjektet har vi tatt i bruk en rekke ulike verktøy for å fasilitere blant annet programmering, kommunikasjon, tekstbehandling og smidige praksiser. Som utviklerteam er det viktig å være bevisst på hvilke verktøy som brukes for å velge de som best mulig støtter opp under oppgavene som skal utføres og arbeidsflyten. Det er verdt å nevne at teamet kun består av privatpersoner, og at vi derfor ikke har tilgang til alle verktøy på grunn av lisensbegrensninger og kostnader. Tidligere kjennskap til verktøyene har også vært en viktig faktor for oss. Vi så det som mest hensiktsmessig å velge blant verktøy som deler av teamet hadde erfaring med fra før av for å ikke bruke unødvendig mye tid på å sette seg inn i nye verktøy.

For programmering og utvikling har vi brukt Android Studio som IDE (integrated development environment). Dette var et naturlig valg for oss ettersom det er den anbefalte IDEen for Android utvikling fra Google, og lar oss kjøre appen på emulator direkte. Som programmeringsspråk har vi

brukt Kotlin da dette var et krav til prosjektet. Kotlin regnes også som det nyeste og mest lettvinte språket for Android utvikling, og er standardspråket for offisiell Android dokumentasjon.

Kommunikasjon er en viktig del av teamarbeid. For skriftlig kommunikasjon har vi brukt Slack og Messenger. Planen var å bruke Slack for all kommunikasjon relatert til arbeid med koden og rapporten, for eksempel diskusjoner og spørsmål. Messenger skulle kun bli brukt til mindre viktig ting som ikke var like faglig. Grunnen til dette er at Slack er mer oversiktlig, i tillegg til at det lar oss skille privatliv fra skole. Det viste seg utover i prosjektet at behovet for Slack ikke var like stort som først ventet. Regelmessige møter gjorde at vi ikke hadde mye som måtte diskuteres over chat. Teammedlemmene syntes også det var vanskeligere å holde seg oppdatert på Slack fordi man ikke var vant med å sjekke appen regelmessig. Dette gjorde at kommunikasjonsflyten gikk tregere enn i Messenger. Messenger ble derfor den viktigste kommunikasjonskanalen gjennom prosjektet.

Zoom er et verktøy for videokonferanser. Verktøyet ble først tatt i bruk etter coronasituasjonen oppsto. Da fikk vi et behov for å ha videomøter over nett. Programmet ble anbefalt av UiO og det var mulig å få full tilgang ved innlogging med Feide-bruker. Zoom ble brukt til alle møter fra og med 16. mars.

Google sitt økosystem, Google Drive, har blitt brukt i prosjektet. Google Docs har blitt brukt til tekstbehandling. Dette inkluderer skriving av prosjektlogg, diverse dokumenter og rapport. Google Docs er skybasert og er laget for å enkelt kunne dele dokumenter mellom flere personer. Arbeid i Docs gjør at vi enklere kan samarbeide på dokumenter fordi flere kan jobbe på ett dokument samtidig. Docs har også automatisk lagring som hindrer at informasjon går tapt. Vi tok også i bruk programmets funksjoner for redigering og kommentering som en måte å bearbeide tekst på. Google Forms har blitt brukt for å utforme og gjennomføre spørreundersøkelser.

6.4.1 Git og github

Git har blitt brukt for versjonskontroll, og i samspill med Github er dette verktøyet vi har brukt for deling og samkjøring av koden. Git har god integrasjon med Android Studio og er i tillegg bransjestandard som versjonskontrollsystem.

All koden som har blitt distribuert mellom gruppemedlemmer har blitt sendt gjennom UiO sin lokale Github tjeneste, og tjenesten har vært instrumental for å kunne gjennomføre prosjektet på en profesjonell måte. Vi har benyttet oss av 'branches' i tjenesten for å kunne separere koden i utviklingsfasen, dette har fungert slik at alle medlemmene har fått hver sin dedikerte branch å

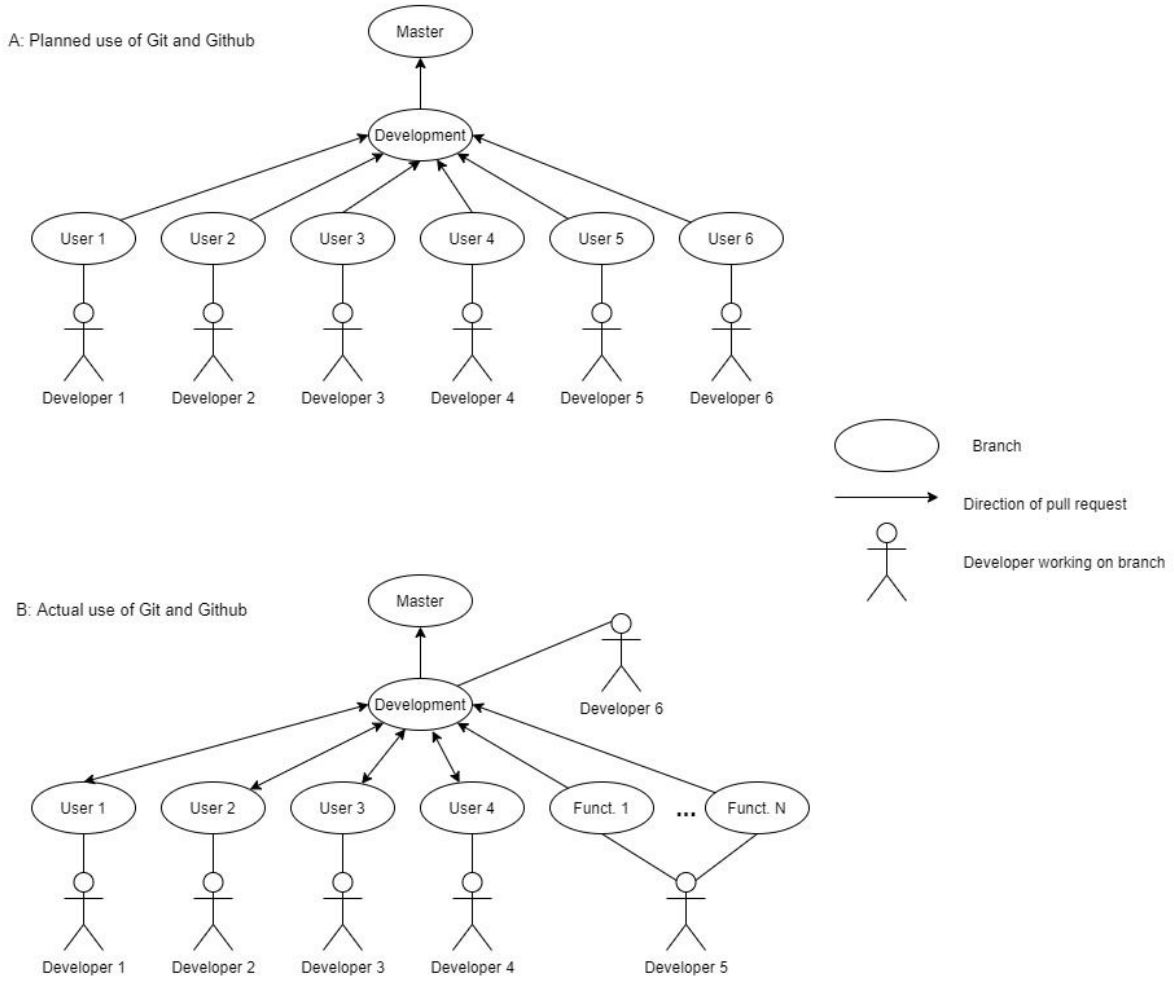
utvikle i. Koden har blitt samlet i en egen development-branch som har fungert som grunnmuren i den fungerende kildekoden. Her har vi samlet funksjonalitet som har vært resultater av sprintene, og videreutviklet hverandres kode. Denne branchen er et resultat av alles kode og har vært en målestokk på om koden vår er modulær og lett å jobbe med. Ved endringer i development-branchen har alltid code-review vært sentralt, der en person som ikke har skrevet koden ser over endringene og godkjenner endringene som er gjort.

Vi har ønsket at Master-branchen til prosjektet skal være det ferdige resultatet av development, i den forstand at funksjonaliteten er testet. For å gjennomføre prosjektet på en mest mulig naturlig og relevant måte har vi fulgt flere av metodene som arbeidslivet praktiserer, der det største unntaket er at funksjonalitet som implementeres ikke dedikeres til sin egen branch.

Github ble også brukt som verktøy for smidige praksiser. Github Projects gir tilgang til prosjekttavle og organisering av oppgaver på denne. Vi brukte denne funksjonen for både kode- og rapportsprint, fordelt i to forskjellige prosjekttavler. Det var naturlig for oss å benytte Github Projects til fordel for lignende programmer som Trello fordi vi allerede hadde kodebiblioteket vårt på samme side.

Teammedlemmene hadde blandet erfaring med bruk av Git. Alle var enig i at det å gjøre seg kjent med bruken av Git var litt vanskelig, men den totale opplevelsen etter å ha blitt bedre kjent varierte. Mens halvparten følte de hadde en bratt læringskurve og fikk god utbytte av Git, følte andre igjen at de fortsatt ikke har like god kontroll på Git som de skulle ønsket.

Selv om den faktiske bruken var forskjellig fra det vi planla, hadde teamet ingen store problemer med Git eller Github. Likevel ser vi at mangelen på forståelse av Git førte til at vi ikke greide å bruke verktøyet slik vi hadde planlagt. I videre prosjekter blir det derfor viktig for oss å passe på at alle forstår hvordan vi har planlagt å bruke et verktøy, og passe på at alle får prøvd seg frem og skjønner bruken av det slik det er planlagt. Dette gjelder ikke bare Git og Github, men alle nye verktøy vi velger å ta i bruk.



Figur 6.4.1. Illustrasjon over planlagt bruk og faktisk bruk av Git og Github.

6.5 Den første datainnsamlingen

Den første datainnsamlingen vi gjorde var et digitalt spørreskjema som ble sendt ut til venner og bekjente via sosiale medier. Målet var å få svar fra flest mulig personer for å få en oversikt over interessen for vindkraft, samt se hvilken funksjonalitet som er mest relevant for en app om vindkraft.

Spørreskjema regnes som en kvantitativ datainnsamlingsmetode. Det gir oss svar fra mange deltakere og disse er lett å kvantifisere i form av talldata eller kategorier. Metoden er rask å gjennomføre, og det er lett å motivere deltakere til å svare, fordi det krever lite av deltakeren. Vi vurderte derfor et spørreskjema som en god metode for å få et utgangspunkt å jobbe videre fra. Dataene ga oss også muligheten til å validere om antakelsene vi hadde tatt i oppstartsfasen stemte.

Den første delen av spørreskjemaet var fokusert på deltakerens interesse for vindkraftproduksjon. Her fikk vi bekreftet at det fantes en interesse for en slik app, spesielt blant de som svarte at de interesserte seg for klima og miljø.

I oppstartsfasen kom vi opp med funksjonalitet vi tenkte ville være relevant i forhold til casebeskrivelsen vår. I den andre delen av undersøkelsen stilte vi spørsmål om hvilken av disse funksjonene som var mest foretrukket. I tillegg lot vi deltakerne fylle inn ønsket funksjonalitet fritt i et åpent tekstfelt. Det var denne delen av spørreskjemaet som ga oss funn som var med på å forme de videre kravene for applikasjonen.

Etter å ha analysert dataen (se vedlegg 9.3) fra undersøkelsen valgte vi å jobbe videre med de funksjonene som viste seg som mest interessante. Disse var 'visning av vindmøller på kart' og 'produksjon i sanntid'. Vi valgte å legge mer vekt på de åpne svarene fra deltakerne, ettersom dette gir høyere grad av brukermedvirkning enn hvis vi vektlegger de svarene vi selv har kommet frem til. Fra dette ble det dannet to nye funksjonaliteter til appen: 'produksjon kan relateres til sammenlignbare enheter' og 'miljøgevinst ved vindkraftproduksjon'.

6.6 Den andre datainnsamlingen

Etter den første datainnsamlingen satt vi igjen med en del kvantitativ data. Ved å analysere dette etablerte vi en oversikt over potensielle krav for appen vi kunne tenke oss å jobbe videre med. For å få et inntrykk av målgruppens meninger om våre foreløpige designvalg og funksjoner, ønsket vi videre å gjennomføre en kvalitativ datainnsamling.

En av brukerundersøkelsene vi hadde planlagt å gjennomføre var en fokusgruppe med fire til seks potensielle brukere. Dette ønsket vi å gjøre for å få bedre oversikt over hvordan potensielle brukere ser for seg en slik applikasjon, og hvilke funksjoner de ville foretrukket. En fokusgruppe er en samling av inviterte personer som deltar i en planlagt diskusjon. Satt opp mot et intervju vil vi i begge tilfeller få kvalitativ data da vi kan stille oppfølgingsspørsmål og be om utdypning fra deltakerne. Fordelen her er at det vil være mer tidseffektivt å gjennomføre en fokusgruppe enn å gjennomføre disse som individuelle intervjuer. Ulempen er naturligvis at vi ikke får gått like mye i dybden, da det er flere deltakere under fokus.

Fokusgruppe er en god start for å etablere krav, forstå målgruppens behov og eventuelle problemer (Lazar, Feng & Hochheiser, 2017, s. 187-228). Dette vil gi oss muligheten til å presentere en prototype av hvordan vi tenker oss appen til flere brukere, i tillegg til at brukerne selv får muligheten

til å vise oss mulige funksjoner og designvalg de kunne tenke seg å ha med. Vi hadde også kunnet få inspirasjon til ting vi selv ikke hadde tenkt på, eller bekreftelse på at det vi fremstilte er av interesse hos målgruppen. Dette vil i motsetning til andre datainnsamlingsmetoder, slik som spørreskjemaer, gi en mye dypere forståelse av hva brukerne vil ha i appen, og hvorfor de vil ha dette. Dette vil være kvalitativ data, som på dette tidspunktet i prosjektet ville gitt oss veldig god tilbakemelding på hvordan vi burde utvikle appen videre med brukerne i tankene.

I praksis ville vi gjennomført fokusgruppen ved å utforme samtykkeskjemaer som samtlige av deltakerne skulle skrive under på. Disse skjemaene forklarer kort hensikten med fokusgruppen, og rettighetene til deltakerne, blant annet muligheten til å trekke både seg og dataen til enhver tid. Videre ville alle deltakerne fått samme forberedende informasjon om undersøkelsen slik at alle skulle stille likt. For å dokumentere prosessen var planen å ha et gruppemedlem med hovedfokus på å lede fokusgruppen, mens to andre gruppemedlemmer tok notater og bilder underveis. Ut i fra deltakeres holdninger ville vi tatt en vurdering på om vi i tillegg skulle filme eller ta lydopptak av hele fokusgruppen for å ha muligheten til å gå gjennom flere ganger og trekke ut data vi kan ha oversett.

Selv om fokusgruppe er en god måte å få oversikt over brukernes behov og krav, er det en vanskelig prosess med mye å tenke på. Selektering av deltakere er krevende, da vi må passe på at vi finner deltakere som representerer målgruppen. For å løse dette planla vi å ta kontakt med brukere i ulike spektre av målgruppen, slik at vårt utvalg ble mest mulig representativt. Dette vil uansett alltid være et usikkerhetsmoment da vi ikke kan være sikre på om vårt utvalg kan tale på vegne av hele målgruppen. Det tar også mye ressurser både å forberede og gjennomføre fokusgruppe, noe vi må ta i betraktning da vi har begrenset med både personer og tid for å gjennomføre.

Videre er det vanskelig å gjennomføre en god fokusgruppe, da en samling av flere personer fort kan føre til digresjoner og dårlig fokus hos deltakerne. Vi må også ta i betraktning at det er ulike personligheter som samles, og sørge for at alle får sagt det de ønsker, da dette ikke alltid kommer naturlig. Her må vi også passe på å være kritiske til hva brukerne svarer, siden det ofte forekommer at brukere svarer det de tror vi ønsker å høre, og på denne måten gir oss bias. Sensitive spørsmål og temaer bør også unngås, da dette også kan føre til feilaktig data.

Dessverre valgte vi å ikke gjennomføre denne brukerundersøkelsen, situasjonen tatt i betraktning, da vi ble enige om at det ville vært verken forsvarlig eller mulig for oss å fysisk samle sammen en gruppe mennesker for å gjennomføre. Dette medførte at vi fikk mindre kvalitativ data i prosjektet enn opprinnelig ønsket, og vi ble nødt til å ta mer avgjørelser selv i forhold til hva vi trodde en bruker

hadde ønsket. Vi ble også i stor grad tvunget til å lene oss mer på den kvantitative brukerundersøkelsen vi gjennomførte.

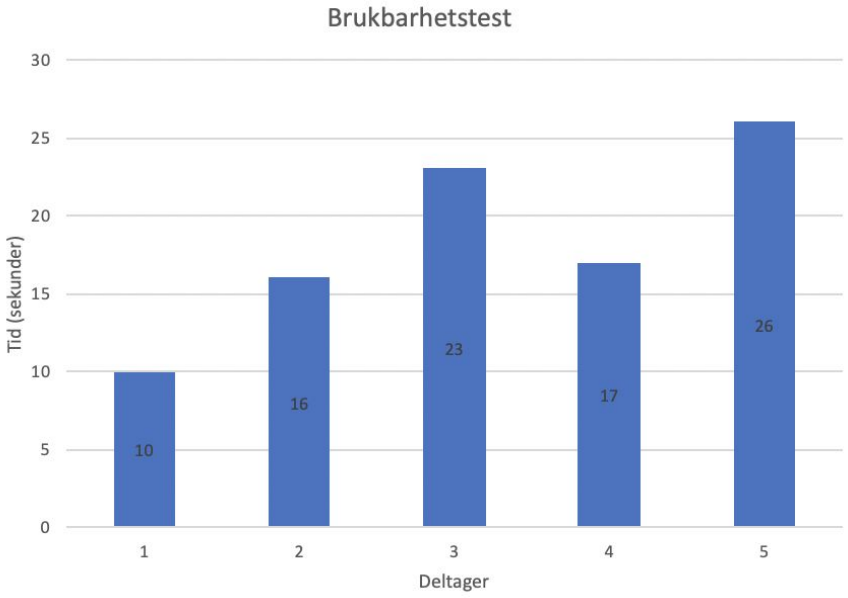
6.7 Summativ evaluering

I slutten av prosessen ønsket vi å gjennomføre en summativ evaluering av applikasjonen. Dette er en evaluering som skjer med brukerne, ofte når mange designvalg er tatt, hvor produktet er tilnærmet ferdig (Preece et al, 2015). Hensikten er å måle påvirkningen systemet har på brukeren, og dermed evaluere om man har klart å oppfylle kravene for å dekke behovene til brukergruppen. For å gjøre dette gjennomførte vi en brukertest for å teste brukbarhet og brukertilfredshet.

Brukbarhet beskriver om appens design støtter brukeren i å utføre sine oppgaver. Brukbarhet kan objektivt måles ved hjelp av ulike metrikker eller suksesskriterier. Disse inkluderer blant annet antall feil, tid, fullføringsrate og antall forsøk. En brukbarhetstest er å måle disse metrikkene med brukere i kontrollerte omgivelser. Gjennom denne testen kan du få et inntrykk av om applikasjonen har god brukbarhet.

For å teste brukbarheten til *Windr* ønsket vi å samle noen representanter fra målgruppen og la dem teste appen i kontrollerte omgivelser. På grunn av Covid-19 pandemien ble dette en utfordring, og vi bestemte oss for å teste med venner og bekjente som vi enkelt hadde tilgang til. Testen ble gjennomført på samme måte som planlagt. Vi var bevisste på at deltakerne ikke representerer målgruppen, og at dette derfor kan gi skjevhet i dataen. Metrikken som ble testet var tidsbruk ved gjennomføring av oppgaver i *Windr*. Et av de ikke-funksjonelle kravene var at bruker skulle finne frem til informasjon om en spesifikk vindpark på 30 sekunder. Å måle tidsbruken var dermed et viktig aspekt ved testingen.

Vi gjennomførte testen med fem deltakere. Alle deltakerne signerte samtykkeerklæring (se vedlegg 9.2) og fikk samme informasjon om testen de skulle være med på. Deltakerne fikk vite navnet på en vindpark, og ble deretter bedt om å finne informasjon om den i *Windr*. Deltakerne hadde aldri brukt appen tidligere. Vi tok tiden fra brukeren åpnet appen til de hadde funnet riktig informasjon. Alle de fem tidene var mellom 10 og 26 sekunder og gjennomsnittstiden var 18,4 sekunder. Fordi vi kun hadde fem datapunkter kunne vi ikke gjennomføre en gyldig statistisk analyse. Resultatene ga oss likevel en pekepinn på hvor brukbar appen er. Vi så også at alle tidene var under 30 sekunder, som var et av de ikke-funksjonelle kravene våre. Vi vil dermed si at vi har klart å dekke kravet om høy brukbarhet.



Figur 6.7.1. Tiden deltakerne brukte på å gjennomføre en spesifikk oppgave i Windr.

I tillegg til brukbarhet ønsker vi å vite noe om brukernes generelle tilfredshet ved bruk av appen. Med tilfredshet mener vi hvorvidt brukeren føler at appen gjør det de ønsker på en god måte, og uten vanskeligheter. For å teste brukertilfredsheten lot vi brukerne prøve seg fritt med appen i fem minutter. Deretter stilte vi noen åpne spørsmål for å forstå brukernes subjektive meninger. Spørsmålene vi stilte var:

- Hva synes du om appen?
- Var det noe du likte med appen?
- Var det noe du ikke likte med appen?
- Var det noe du savnet i appen?

Vi følte at disse spørsmålene ville gi oss et godt overblikk hva som var bra med appen, og hva som kan forbedres. Under evalueringen ble svarene fra deltakerne notert fortløpende som stikkord. Vi så det ikke som nødvendig å ha en mer omfattende dokumentasjon eller analyse fordi brukerne ikke var reelle. Tilbakemeldingene om *Windr* var likevel nyttige i den forstand at vi fikk et nytt perspektiv på vår løsning.

Under evalueringen fikk vi mange ulike innspill, men det var også mye som gikk igjen hos flere deltakere. Det som kom frem fra alle deltakerne var ønsket om å kunne søke direkte inne på kartsiden. Flere av deltakerne ønsket også en 'vis på kart ' funksjon på listevisningen, slik at

de enkelt kunne finne ut hvor parken lå. Det kom også frem at det var et ønske om å kunne skru på 'darkmode' da flere var vant med dette fra andre apper. Disse tingene er funksjonalitet vi gjerne skulle implementert dersom vi hadde bedre tid til å utvikle appen.

Det var også noen mindre viktige uklarheter som deltakerne trakk frem. Blant annet var ikke helt tydelig hva 'installert effekt' betydde. Deltakerne ga også uttrykk for at det hadde vært lettere å se at man kunne endre kartet hvis satelittkartknappen var plassert på kartsiden. I tillegg fikk vite at det det hadde vært mer intuitivt om info om en vindpark dukket opp første gang man trykket på markøren, i stedet for å måtte trykke en gang for å zoome inn og en gang til for å få opp parken. Disse tingene er mindre justeringer som vi relativt enkelt kunne endret på for å tilfredsstille brukerne, og er noe av det første vi ville gjort hvis vi hadde bedre tid.

Det var også en del positive tilbakemeldinger. Generelt synes deltakerne at *Windr* var enkel å bruke og at den var raskere enn forventet. Deltakerne likte også at appen hadde et enkelt utseende og ikke for mange visuelle elementer på hver side. En av deltakerne sa blant annet: "Jeg likte at det var lite å se på og at det var et cleant interface". Alle deltakerne var generelt positive til *Windr* og likte ideen. Deltakerne klarte å gjennomføre oppgaver uten problemer og synes appen var enkel å bruke. Flere ga uttrykk for mestringsfølelse ved bruk. Dette tyder på god brukertilfredshet.

Alt i alt var evalueringen veldig nyttig, til tross for begrensningene rundt gjennomføringen. Vi fikk en god oversikt over hva som kunne vært endre eller jobbet videre med, men fikk også bekreftet at vi hadde et godt utgangspunkt. Basert på tilbakemeldingene vi fikk virket det som appen har god brukbarhet og generelt god tilfredshet blant brukerne.

6.8 Kravspesifikasjon

Ideen om en app som kunne beregne vindkraftproduksjon kom tidlig i prosjektet. Den opprinnelige tanken var at det kunne bli en app som ga nyttig markeds- og produksjonsinformasjon til aktører innen vindkraftbransjen spesielt og kraftmarkedet generelt.

I den opprinnelige kravspesifikasjonen skulle appen levere produksjonsdata for alle landets parker i sanntid, prediksjoner for produksjon fremover i tid og, ved hjelp av prisdata, inntekter for alle vindparkene. På dette tidspunktet var det ikke noen konkrete krav til UX.

Den første endringen av kravspesifikasjonen kom som en følge av tilgangen på prisdata. I utgangspunktet var planen at dette kunne hentes gratis fra NVE sine API. Det viste seg at tilgangen

på prisdata lå hos Nord Pool, den største kraftbørsen i Norge, og koster penger. Dessuten har Nord Pool bare prisdata frem til kl. 12 på dagen, som gjorde det vanskelig å beregne inntekter frem i tid på en forutsigbar måte. Uten tilgang på prisdata mistet appen mye av den kommersielle nytten.

Derfor endret appen formål og målgruppe, og dermed også krav. Den nye målgruppen ble klimainteresserte og formålet med appen ble å vise hvor viktig vindkraft er og klimaeffekten av den. Med dette formålet var ikke prisdata interessant lenger, men det ble viktig å presentere informasjonen på en måte som skapte entusiasme. Teamet kom frem til at en kartvisning var en god måte å vise omfanget av vindparker i hele landet, og dette ble tatt inn i kravspesifikasjonen.

For å gjøre appen mer oversiktlig ble det besluttet å legge til en søkbar liste. Denne listen skulle ha en tett kobling med kartet slik at bruker kunne velge et element i listen og se hvor det var i kartet. Det ble besluttet at bruker skulle kunne søke etter vindparker etter kommunenavn, og ikke bare navn på parken, da det var usikkert hvor mange som kjente til navnet på spesifikke vindparker. Det ble også lagt til krav om adaptive ikoner i listen. Ikonene skulle illustrere om parken produserer mye, lite eller ingenting.

Enkelte funksjoner som fremdeles er i kravspesifikasjonen, ble ikke ferdige. Sanntidsdata om vind og produksjon er ferdig, men teamet rakk ikke å finne en god måte å illustrere produksjonsdata frem i tid. Koblingen mellom kart- og listevisning ble heller ikke implementert. Det betyr at bruker er nødt til å lete igjennom markørene i kartet for å finne en spesifikk park der. Visning av produksjonsdata over tid og kobling mellom kart og liste vil være høyt prioritert ved videre utvikling av applikasjon.

7. Refleksjon

I dette prosjektet har vi utviklet en android app som et team gjennom en smidig prosess. Alt gikk ikke som planlagt, men vi har fått mange gode erfaringer og lærdommer fra utfordringene vi har møtt på.
Hele utviklingsprosessen har blitt ganske annerledes enn det vi så for oss da vi begynte å planlegge på starten av semesteret på grunn av Covid-19. Dette har medført at de fysiske møtene frafalt helt etter at IFI stengte 12. mars. Dette var ganske utfordrende til å begynne med da nettbaserte møter og samarbeid over internett var mer krevende enn ved fysisk samarbeid. Med tiden utviklet dette seg likevel nærmest til en fordel. Det var lavere terskel for å gjennomføre hyppige standup møter. Samtidig fant vi løsninger på hvordan gruppen kunne jobbe med de samme oppgavene over nett ved å dele skjermer, vise ting til kamera og jobbe i samme nettbaserte dokument.

En viktig del av prosjektet var å følge smidige utviklingspraksiser. Vi delte arbeidet opp i sprinter og brukte prosjekttavler for å planlegge og følge opp arbeidet. Vi gjennomførte standup møter, retrospektive møter, sprint planning og sprint reviews. Dette er noen av prinsippene som gjorde prosessen vår smidig. Vi ser likevel at det var andre smidige praksiser som vi kunne vært flinkere til å følge for å få en best mulig smidig utviklingsprosess. Skulle vi gjennomført prosjektet igjen ville vi hatt større fokus på tidlige leveranser. Dette kunne vi gjort ved å ha et mer konkret fokus for hver iterasjon. Det var også flere ting vi ikke fikk implementert grunnet dårlig tid, og vi fikk heller ikke lagt til mye ny funksjonalitet. Dette kunne vært unngått om vi hadde vært flinkere til å beregne arbeidsmengde i de tidligere sprintene. Vi ville også prøvd å være mer effektive med å sette oss inn i Git og API'et tidlig i prosessen.

Selve teamarbeidet har gått bra, og vi møtte ikke på noen store utfordringer. Vi er fornøyde med egen innsats og synes vi har kommunisert godt gjennom prosjektet. Motivasjonen var noe synkende mot slutten, men vi har alltid vært enige om ambisjonsnivå og arbeidsmengde. Vi har hatt en jevn arbeidsfordeling, og alle har bidratt på ulike deler av prosjektet innenfor både kode og rapport. Dette var et mål vi satt oss i starten. Fordi vi har forskjellig kompetanse og interesser var det naturlig at gruppemedlemmene bidro forskjellig på ulike deler av prosessen for å effektivisere arbeidet og få et best mulig resultat. Vi synes dette fungerte godt, og alle medlemmene føler de har bidratt tilstrekkelig.

Alt i alt er vi fornøyde med gjennomføringen av prosjektet. Vi har lært mye om hvordan det er å jobbe i et selvstendig team. Vi har også lært mye om scrum og kanban som smidige utviklingsmetoder og føler oss godt rustet til å jobbe med smidig utvikling i fremtiden. Selv om alt vi ønsket i appen ikke ble implementert har vi utviklet en app som vi er fornøyd med. Vi føler at *Windr* oppfyller de viktigste kravene vi satt i starten av prosjektet, og er resultatet av et godt samarbeid.

8. Referanseliste

Dahl, I.D. (2019, 8. oktober). *Tenner varder Norge rundt for å vise motstand mot vindindustri*. Hentet 26. april 2020 fra https://www.vg.no/nyheter/innenriks/i/EWEjJA/tenner-varder-norge-rundt-mot-vindindustri

Digitaliseringsdirektoratet (i. d.). WCAG 2.0-standarden. Hentet 29. mai, 2020 fra https://uu.difi.no/krav-og-regelverk/wcag-20-standarden

Fjørtoft, T.O. (2017, 06. juni). *Unge og høyt utdannede er flinkest foran PC-en*. Hentet 26. mars 2020 fra https://www.ssb.no/teknologi-og-innovasjon/artikler-og-publikasjoner/unge-og-hoyt-utdannede-er-flinkest-foran-pc-en

Kent, B. et al (2001, 13 februar). *Manifestet for smidig programvareutvikling*. Hentet 26. april 2020 fra https://agilemanifesto.org/iso/no/manifesto.html

Kleven, Ø. (2020, 24. mars). *Miljøpartiet gjør det godt blant de yngre*. Hentet 25. mars, 2020 fra https://www.ssb.no/valg/artikler-og-publikasjoner/miljopartier-gjor-det-godt-blant-de-yngre

Lazar, J., Feng, J. H. & Hochheiser, H. (2017). *Research methods in human-computer interaction (2nd edition).* USA: Elsevier Inc.

Lindsjørn, Y. (2019, 4. april). Modellering av krav. Hentet 27 mai, 2020 fra https://www.uio.no/studier/emner/matnat/ifi/IN1030/v19/undervisningsmateriale/in1030_2019.04.04_use-case-modellering.pdf

Norges vassdrag- og energidirektorat (2015, 11. juni). *Elsertifikater*. Hentet 21. mai 2020 fra https://www.nve.no/energiforsyning/elsertifikater/

Norges vassdrag- og energidirektorat (2020, 9. mars) Vindkraft. Hentet 15. mai 2020 fra https://www.nve.no/energiforsyning/kraftproduksjon/vindkraft/

Norges vassdrag- og energidirektorat (i. d.). Vindkraftdata. Hentet 27. mai 2020 fra https://www.nve.no/energiforsyning/kraftproduksjon/vindkraft/vindkraftdata/

Preece, J., Rogers, Y., & Sharp, H., (2015). *Interaction Design: Beyond human-computer interaction (4th edition).* West Sussex: John Wiley & Sons Ltd.

Runde, R. S. (2017, 2. mars). Objektorientert design av kode. Refaktorering. Hentet 27. mai 2020 fra https://www.uio.no/studier/emner/matnat/ifi/INF1010/v17/lysark/oodesign1-2017.pdf

Sjoli, S. (2019, 24. februar). Automatiserte tester - hva og hvorfor. Hentet 24. mai 2020 fra https://www.bouvet.no/bouvet-deler/automatiserte-tester-hva-og-hvorfor

Sommerville, I. (2011). *Software Engineering (9th edition).* USA: Addison-Wesley.

Statcounter (i. d.). Mobile & Tablet Android Version Market Share Norway. Hentet 5. april 2020 fra https://gs.statcounter.com/android-version-market-share/mobile-tablet/norway

9. Vedlegg

9.1 Use-case

Hente informasjon om vindparkene fra API

Prebetingelse:

Applikasjon har tilgang på internett

Postbetingelse:

Applikasjonen har liste over vindpark med komplette data

Applikasjonen har liste over vindparker med inkomplette data

Applikasjonen mangler data og starter use-case på nytt

Applikasjonen avsluttes

Aktør:

NVE og Met

Hovedflyt:

1. Systemet henter informasjon om kraftprodusenter fra NVE
2. Systemet sjekker hvilke som er vindkraft
3. Systemet henter koordinater fra XML-fil basert på parknavn
4. Systemet henter vind-data fra Met API
5. Regn ut produksjon fra vinddata
6. Systemet lagrer informasjonen som en liste over parker

Alternativ flyt 1, steg 1 - API-kallet returnerer med feilkode(4XX)

A1.1. App skriver ut feilmelding til bruker

A1.2. La bruker velge mellom å forsøke på nytt(1) eller eller avslutte(2)

A1.3.1 Start på nytt fra hovedflyt 1

A1.3.2 Avslutt applikasjon

Alternativ flyt 2, steg 4 - Vinddata-use-case gir feilmelding

A.2.1 Systemet skriver feilmelding til bruker

A.2.2 Systemet manglende informasjon med “NA” i UI

Velge vindpark på kart

Prebetingelse:

Applikasjon har tilgang på internett

App har koordinater til alle parkene

Postbetingelse:

Alle parkene er vist i kartet med markør

Parkene vises kun i recycler view

Applikasjon henter nye parkdata

Aktør:

Bruker

Hovedflyt:

1. Hent liste med vindparkdata
2. Hent koordinatene fra hver park
3. Plasser markør på korrekt plass vha. google maps API
4. Opprett kobling mellom markør og korresponderende card view
5. Flytt senter av kart til markør

Alternativ flyt 1, steg 1 - Liste med vindparker er tom

A.1.1 Hent parkdata

A.1.2 Start hovedflyt fra punkt 1

Alternativ flyt 2, steg 2 - Ikke kontakt med google sitt API

A.2.1 Skriv feilmelding til bruker

A.2.2 Bytt activity til recycler view med kraftverk

Systemet skal vise energiproduksjon på en forståelig og håndfast form

Prebetingelse:

Har vindpark data

Postbetingelse:

Informasjon vises i card view

Hovedflyt:

1. Velg sammenligningsgrunnlag
2. Hent tall for produksjon
3. Gjør beregninger
4. Hent visningsikon
5. Vis resultat i cardview

Alternativ flyt 1, steg 1 - Mangler produksjonsdata

A.1.1 Sett inn “NA” plasser i card view

A.1.2 Hent nye produksjonsdata i bakgrunnen

A.1.3 Oppdater card view med korrekte data

Se informasjon om en spesifikk vindmøllepark

Prebetingelse:

- Applikasjon har tilgang til kartet.
- Applikasjon har koordinater på alle vindmølleparker i Norge.

Postbetingelse:

- Applikasjon har oversikt over alle vindmølleparker i Norge.
- Bruker får informasjon om en spesifikk vindmøllepark.

Aktør:

- Bruker.
- NVE.

Hovedflyt:

1. Bruker åpner applikasjonen.
2. Systemet gjør et API-kall til NVE.
3. NVE sender tilbake informasjon over alle (fornybare) energikilder.
4. Systemet filtrerer på alle vindmølleparker.
5. Systemet viser kart og oversikt over alle vindmølleparker til bruker.
6. Bruker trykker på en spesifikk park.
7. Systemet viser en cardview med informasjon om parken.

Alternativ flyt 1, steg 4: Det finnes ingen oversikt over vindmølleparker

A.1.1. Systemet får ikke tak i oversikten over vindmølleparker.

A.1.2. Systemet viser et tomt kart.

A.1.3. Systemet viser feilmelding om at den ikke fikk tak i noen parker.

A.1.4. Systemet spør bruker om den skal laste inn kartet på nytt.

A.1.5. Bruker trykker på knapp for å laste inn kartet på nytt.

A.1.6. Start hovedflyt, punkt 2.

Alternativ flyt 2, steg 7: Det finnes ingen informasjon om parken

A.2.1. Systemet finner ikke informasjon om parken.

A.2.2. Systemet viser feilmelding om at den ikke greier å hente inn informasjon.

A.2.3. Systemet spør bruker om den skal laste inn kartet på nytt.

A.2.4. Bruker trykker på knapp for å laste inn kartet på nytt.

A.2.5. Start hovedflyt, punkt 2.

Se relaterbart energiproduksjonsnivå

Prebetingelse:

- Applikasjon har tilgang til internett.

- Applikasjon har koordinater på alle vindmølleparker i Norge.

Postbetingelse:

- Applikasjon har oversikt over alle vindmølleparker i Norge.
- Applikasjon viser energiproduksjon på et relaterbart format for bruker.

Aktør:

- Bruker.
- NVE.

Hovedflyt:

1. Bruker åpner applikasjon.
2. Systemet gjør et API-kall til NVE.
3. NVE sender tilbake informasjon over alle (fornybare) energikilder.
4. Systemet filtrerer på alle vindmølleparker.
5. Systemet viser kart med oversikt over alle vindmølleparker til bruker.
6. Bruker trykker på en spesifikk vindmøllepark.
7. Systemet viser et card view med energiproduksjonsnivå på et relaterbart format til bruker.

Alternativ flyt 1, steg 6:

A.1.1. Bruker trykker på listemenyen.

A.1.2. Systemet viser en listeoversikt, sortert alfabetisk, over alle vindmølleparker i Norge.

A.1.3. Bruker trykker på sorteringsmuligheter og velger energiproduksjon.

A.1.4. Systemet viser en listeoversikt, sortert etter synkende energiproduksjon, over alle vindmølleparker i Norge.

A.1.5. Start hovedflyt, steg 6.

9.2 Samtykkeerklæring

Samtykkeerklæring for brukerundersøkelse i IN*2000*-prosjekt

Beskrivelse av prosjektoppgaven
Vi er en studentgruppe i kurset *IN2000: Software Engineering* ved Institutt for informatikk, Universitetet i Oslo. Prosjektgruppen består av Mathias Demeer Strøm, Bendrik Werp, Patrik Haukaa, Andreas Baardseth, Ole-Kristian Ellingset og Carina Truong.

Kontaktperson:
Mathias Demeer Strøm 41353757 mathiads@matnat.uio.no

Prosjektet har som formål å utvikle og ferdigstille en mobilapplikasjon, samt dokumentere denne prosessen i en rapport. Som den del av dette ønsker vi å evaluere appen vår, *Windr,* med hjelp av din deltakelse. Appen handler om klima og vindkraft i Norge. Evalueringen vil gå ut på å svare på noen spørsmål knyttet til appen, samt å være med på noen enkle tester av applikasjonen. Formålet med denne undersøkelsen er å finne ut hvordan *Windr* tilfredsstiller de ulike kravene vi har satt, og evaluere appen som helhet.

Frivillig deltakelse
All deltakelse er frivillig, og du kan trekke deg når som helst. Vi ønsker å notere under og etter undersøkelsen. Disse notatene vil vi kun bruke som informasjon til prosjektet. Du kan når som helst avslutte undersøkelsen eller trekke tilbake informasjon som er gitt under undersøkelsen.

Anonymitet

Alle som deltar på undersøkelsen vil være anonyme i rapporten. Det vil ikke bli notert eller lagret noen form for personopplysninger. Notater fra undersøkelsen vil bare leses av oss i prosjektgruppa og lærerne i kurset. Notatene og rapporten vil bli anonymisert: ingen andre enn prosjektgruppa vil vite hvem som er blitt snakket med, og det du sier vil ikke kunne tilbakeføres til deg. Før undersøkelsen begynner, ber vi deg om å samtykke i deltagelsen ved å undertegne på at du har lest og forstått informasjonen på dette arket, og ønsker å stille opp til undersøkelsen.

Samtykke

Jeg har lest og forstått informasjonen over og gir mitt samtykke til å delta på undersøkelsen.

_____ _____

Sted og dato Signatur

9.3 Analyse av første datainnsamling: Spørreundersøkelse

Den første datainnsamlingen vi gjorde var et digitalt spørreskjema som ble sendt ut til venner og bekjente. Målet var å få svar fra flest mulig personer for å få en oversikt over interessen for vindkraft, samt se hvilken funksjonalitet som er mest relevant for en app om vindkraft.

Deltakere ikke helt riktig målgruppe/brukergruppe som gjør at data veier mindre enn svar fra målgruppen. Ikke representativt.

Relasjon til deltakere: Sendte kun til venner og bekjente som kan ha påvirket svarene. Bias.

Oppbyggingen av spørreundersøkelsen var også noe vi tenkte på under utformingen. Til å begynne med skrev vi ned spørsmål vi mente var relevante, og stilte de i vilkårlig rekkefølge. Det første spørsmålet var da "Har du noen tilknytning til vindkraft utenom det allmenne?" Dette bestemte vi oss fort for å flytte lenger ned da det kan virke litt skremmende for folk som ikke har noe spesielt forhold til vindkraft.

Med de dataene vi sitter igjen med etter denne datainnsamlingen, ser vi at vi kunne fått mye igjen for å ha strukturert fremgangsmåten litt annerledes. Vi hadde ikke spikret målgruppen helt før vi gjennomførte undersøkelsen, og både spørsmålene og svarene har ulik grad av relevans for oss videre. De funnene vi kommer til å jobbe med videre er særlig knyttet til dataen fra spørsmål 5 og 6.

Svar på "Er det noen andre funksjoner du kunne vært interessert i?" (spm 6) Det var 21 av 65 som svarte noe på dette spørsmålet. Vi gikk først gjennom svarene og fjernet feil fra datasettet. Disse feilene bestod av svar som ikke hadde noe med undersøkelsen å gjøre, og svar vi klassifiserte som skjevheter i dataen/ bias. Vi så på noen av svarene at svaret egentlig ikke var av noe relevant betydning, og at det heller virket som deltakerne svarte dette fordi de trodde det ville hjelpe oss i vår undersøkelse. Et eksempel på dette var svaret "En liste over byggeår relatert til vindmølleparker".

Sammenlignbare enheter	**Miljøgevinst**	**Diverse forslag**	**Ikke relevant**

Se hvor stor andel av Norges/ nærområdets strøm som produseres av vindkraft.	Sammenlign opp i mot andre klimavennlige energikilder som feks atomkraft	Mulighet for å se inntekt knyttet til hver vindmøllepark	Nettside istedenfor app
forståelige størrelser på kraften som produseres (i ting som vanlige folk forstår, ikke bare kwt)	Mulighet til å se statistikk over produsert fornybar energi og hvor mye vi har «spart» på det	Prognosert produksjon neste timene	Vet ikke helt hva jeg skulle brukt en slik app til
Kunne vært artig å se produksjon relativt til hvor mye enkelte elektriske enheter i gjennomsnitt trekker. (F.eks nåværende produksjon er nok til å forsyne 50 vaskemaskiner/1000 lyspærer/2 trikker osv osv.) Greit å ha et referansepunkt, da jeg kan tro de aller fleste ikke har noe godt forhold til hvor mye 20 KWh er.	hvor mye CO2 kraften som produseres sparer for jorden,	En liste over byggeår relatert til vindmølleparker.	Se strømprisen per kommune.
Hvor mye strøm den enkelte vindmølleparken produserer og (...)	Sammenliknet med kullkraft	(...) hvor strømmen blir brukt.	Statistikk over hvilke byer som får inn mest inntekt for vindmøllene

Mulighet for å sammenligne vindmølleparkene	Se hvor mye strøm det produseres, hvordan dette brukes og en måte å se hvor mye «miljø» spart vs andre kraftkilder		Veldig mange apper for alt nå, bedre med bare en hjemmeside
Hva mitt hus bruker	(...) hvor mye strøm en vindmøllepark produserer i forhold til andre klimavennlige strømproduksjonen		

Etter å ha undersøkt svarene dukket det opp 4 kategorier: 'sammenlignbare enheter', 'miljøgevinst', 'diverse forslag' og 'ikke relevant'. Vi kodet alle svarene etter disse kategoriene og så at et flertall av deltakerne ønsket å *se strømproduksjon på en mer relaterbar* måte eller kunne *se miljøgevinsten ved vindkraft*. Dette var noe vi ikke hadde tenkt på i den første brainstormingen over funksjonalitet vi ønsket i appen, og noe vi ønsker å ta med videre.

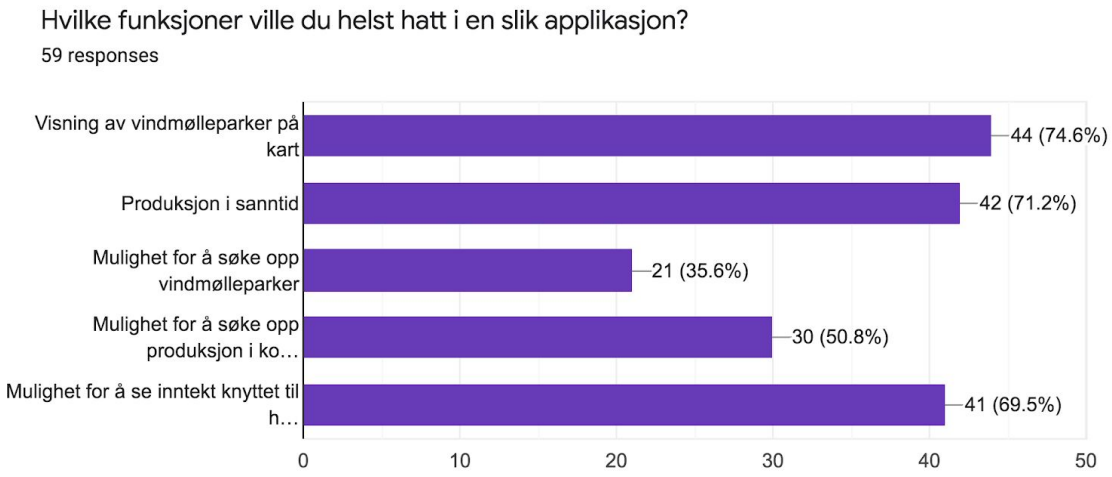
Vi så på svarene fra de 5 som hadde svart de hadde en tilknytning til vindkraft utenom det allmenne, men fant ingen åpenbare forskjeller i svarene deres.

På spørsmål 5 lurte vi på "Hvilke funksjoner ville du helst hatt i en slik applikasjon?". Her var det 6 av de totalt 65 deltakerne som ikke huket av på noen av alternativene. Dette kan både tale for at de ikke ønsket noen av de forhåndsdefinerte funksjonene vi formulerte. Ettersom dette var en undersøkelse som ble sendt ut til mange personer, er det rimelig å anta at flertallet kanskje ikke har noen særlig tilknytning til temaet. Vi må også derfor være åpne for at de rett og slett ikke orket/ville sette seg inn i settingen og se for seg hvilke funksjoner de kunne sett på som relevante.

Som vist i figuren var det tre funksjonaliteter som skilte seg ut og som cirka 70% prosent av alle deltakerne svarte at de ønsket:

- Visning av vindmølleparker på kart (74,6%)
- Produksjon i sanntid (71,2%)

- Mulighet for å se inntekt knyttet til hver vindmøllepark (69,5%)



Ettersom vi kom opp med disse fem forslagene til funksjonalitet uten medvirkning fra brukere vil vi ikke vektlegge svarene i figuren over like mye som svarene på spørsmålet hvor deltakerne selv kunne komme med ønsket funksjonalitet. Dette er fordi vi ønsker å designe appen etter brukerens behov og derfor ha høyest mulig grad av brukermedvirkning. Vi vurderer også spørsmålene med svaralternativer som mer ledende fordi deltakeren kan føle seg tvunget til å velge flere av alternativene, selv om dette kanskje ikke er noe deltakeren hadde ønsket hvis valget av funksjonalitet var helt fritt. Ved å la deltakeren svare i fritekst åpner vi for at brukeren må tenke mer over hva de ønsker.

På bakgrunn av funnene fra datainnsamlingen ser vi at det finnes en interesse for miljøvennlig energiproduksjon og en eventuell applikasjon relatert til dette. Basert på funnene vil vi til å begynne med å ha hovedfokus på følgende funksjonalitet:

- Vindmølleparker på kart
- Produksjon som kan relateres til sammenlignbare enheter
- Miljøgevinst ved vindkraftproduksjon

9.4 Teamavtale

Teamavtale for Team 24

Teammedlemmer

Andreas Baardseth, Ole-Kristian Heskestad Ellingsen, Patrick Persson Haukaa, Mathias Demeer Strøm, Carina Truong, Bendik Elias Eriksen Werp.

Varighet

Teamavtalen trer i kraft f.o.m. datoen teammedlemmet har underskrevet og t.o.m. 29. mai 2020.

Forventning

For at teamet skal fungere godt, og prosjektet skal kunne bli ferdigstilt innen innleveringsfristen 29. mai 2020, har teamet kommet til enighet i noen punkter som det forventes at hvert av teammedlemmene følger:

- Bidra med kildekode.
- Kommentere, og skrive tydelig og lesbar kode.
- Kunne forklare og dokumentere all kode som er skrevet.
- Bidra til å skrive og fullføre rapporten.
- Fylle rollen de har blitt tildelt.
- Komme tidsnok til møter, og melde ifra hvis de ikke kan.
- Jobbe rundt ti timer i uka.

Hva som regnes som tydelig og lesbar kode vil bli fastsatt i eget dokument om hva som kan egne seg som god kodeskikk for dette teamet. Alle forventes å følge denne kodeskikken.

Rollene vil bli rullert på, og det er derfor vurdert som best for teamet at ingen av medlemmene skal ha en fast rolle gjennom hele prosjektet. Likevel vil teammedlemmene få noe hovedansvar, men det forventes ikke at de er alene om å utføre oppgavene tilknyttet dette ansvaret.

Tilstedeværelse

Teamet har avtalt å ha møter tre ganger i uka:

- Mandag, kl. 16:15 - 17:00
- Tirsdag, kl. 12:15 - 14:00
- Torsdag, kl. 12:15 - 14:00

I tillegg til disse faste møtene, har teamet veiledning en gang i uka:

- Tirsdag, kl. 10:15 - 12:00

Det forventes at samtlige av teammedlemmene møter opp til disse tidspunktene. I tillegg vil det forekomme ukentlige variasjoner og avtaler om møter som hvert teammedlem selv er ansvarlig for å holde seg oppdatert på.

Dersom et av medlemmene i teamet ikke har anledning til å komme, har vedkommende plikt til å melde ifra til resten av teamet innen 24 timer før avtalt tidspunkt. Skyldes fraværet sykdom eller annet uforutsett fravær, eller gjelder det forsinkelser, skal teammedlemmet melde ifra så tidlig som mulig og helst innen en time før avtalt tidspunkt.

Konflikthåndtering

Oppstår det konflikter eller andre uoverensstemmelser, skal teamet i utgangspunkt diskutere dette på neste møte. Ved gjentatte brudd på forventningene beskrevet ovenfor skal teamet selv komme til en løsning. Greier ikke teamet selv å fremme en løsning, skal konflikten bli løftet opp til møte med veileder.

For at en beslutning skal være vedtatt må den ha over halvparten av stemmene. I tillegg er det kun teammedlemmene som er til stede på møtet hvor beslutningen blir tatt som har stemmerett, og over halvparten av medlemmene må være til stede for at det skal kunne bli tatt en beslutning. Teammedlemmene forventes å stå bak beslutningene som er vedtatt.

Alle medlemmer i teamet oppfordres til å ta opp hendelser og avvik som kan forbedres.

Ved å signere under, har undertegnede gjort seg kjent med innholdet i avtalen, og er enig i det som står beskrevet.

Dato	Sted	
09/03-20	Oslo	Andreas Baardseth
Dato	Sted	Andreas Baardseth
09/03-20	Oslo	
Dato	Sted	Ole-Kristian Heskestad Ellingsen
9/3-20	Oslo	
Dato	Sted	Patrick Persson Haukaa
9/3 - 20	Oslo	
Dato	Sted	Mathias Demeer Strøm
9/3-20	Oslo	
Dato	Sted	Carina Truong
9/3-20	Oslo	
Dato	Sted	Bendik Elias Eriksen Werp