Team 11

Åpent case: «Utepils»

Medlemmer:

Johannes Sognnæs: johansog

Mathilde Tedesco Siem: mathits

Nikita Fedotovs: nikitafe

Ole Kallerud: olekall

Patrick Bruun: patribru

Trym Olsen: trymwo

Veiledere:

Frida Hope Carpenter

Stian Dolmseth

Presentasjon **4**
Kort introduksjon/sammendrag 4
Presentasjon av prosjektet/caset 4
Kort beskrivelse av løsningen 4
Beskrivelse av teamet 5

Brukerdokumentasjon **5**
Applikasjonens funksjonalitet 5
Strukturen i applikasjonen 7
Målgruppe 8
Plattformer 8

Kravspesifikasjon og modellering **8**
Funksjonelle krav 8

Produktdokumentasjon **12**
Jetpack Compose 12
Fuel 12
Gson 12
MVVM (Model View ViewModel) 12
Beskrivelse av løsningen 13
Struktur 14
Funksjonalitet 15
Brukskvalitet 15
Pålitelighet 15
Dark Mode / Light Mode 16
API-nivå 16
Algoritmen 16
ViewModel og DataSource 17
Sunrise 17
LocationForecast og Nowcast 18
Vinmonopolet API 19
Hosting av egen database 19
Implementering av brukerens lokasjon 19
Kalender 20
Anbefalinger 20
Loading View 20

Testdokumentasjon **20**
Arbeidsprosess for enhetstesting 20
Arbeidsprosessen for enhetstesting - mer teknisk 21
Arbeidsprosessen for integrasjonstesting 23

Prosessdokumentasjonen **24**
Verktøy 29
Rapportprosessen 32
Arbeidsprosessen for utvikling av appen 33

Søkefelt fra gammel versjon 39
Hvordan appen kunne ha sett ut 42
Universell utforming 42
Arbeidsprosess for datainnsamling og analyse 43
Resultater fra datainnsamling og analyse 45
Brukerundersøkelse: 46

Refleksjon **49**

Referanser og kildeføring **51**

Vedlegg **51**
Vedlegg 1: Intervjuguiden 51
Vedlegg 2: Aktivitetsdiagram av funksjonalitetene i applikasjonen 53
Vedlegg 3: Filer i prosjektet 53
Vedlegg 4: Klassediagram over værinformasjon 54
Vedlegg 5: Sekvensdiagram av versjonen med søkefelt 56
Vedlegg 6: Bilde av hvordan appen kunne ha sett ut 56

Presentasjon

Kort introduksjon/sammendrag

I denne rapporten skriver vi om hvordan det har vært å utvikle vår app, Utepils, og hvordan vi har gått fram for å få det til. Dette innebærer alt fra stand-up-møter, til testing av appen. Rapporten har noen deler som er beregnet på lesere med programmerings-tekniske forkunnskaper. Dette gjelder produktdokumentasjonen og testdokumentasjonen.

Presentasjon av prosjektet/caset

Har du noen gang stilt deg det viktige spørsmålet: “Hvordan er utepilsværet nå?” Eller: “Jeg lurer på om det kommer til å være utepilsvær på hytta på lørdag?” Vel, det har vi. Derfor har vi valgt et åpent case som vi har kalt Utepils. Appen appellerer til ølelskere som stiller seg selv disse spørsmålene, ved å tilby en enkel måte å få svar på disse problemstillingene. Grunnen til at vi valgte dette åpne caset er at vi synes idéen virker morsom og interessant å jobbe med. I tillegg tenkte vi at det ikke er noen grense på hvor mye tilleggsfunksjonalitet det er mulig å ha med, da vi ofte kom med nye idéer og forslag som vi synes er bra. Vi ønsket også å ha et case som det ikke er så mange andre som har.

Kort beskrivelse av løsningen

Hovedfunksjonaliteten i appen er at når du går inn i appen, får du opp et vindu som sier om det er utepilsvær eller ikke, på din posisjon i dette øyeblikk. Vi har brukt API-ene Nowcast, LocationForecast og Sunrise fra Meteorologisk institutt sammen med enhetens lokasjon til å beregne om det er utepilsvær eller ikke. Basert på disse dataene regner appen ut hvor bra utepilsværet er i en prosent. Vi kan bruke denne prosenten til å fortelle brukeren om det er fint nok vær til å ta en pils ute eller ikke. I Utepilsappen er det også mulig å sjekke utepilsværet i opptil ni dager fremover i tid og man får opp drikkeanbefalinger basert på årstid og hvordan været er.

Beskrivelse av teamet

Gruppen vår består av Johannes, Ole, Patrick og Mathilde som går programmering og systemarkitektur og Trym og Nikita som går design. Vi var to grupper på tre som kjente hverandre fra før som ble satt sammen. Første gang vi møttes var alle med på å snakke, og vi

var flinke på å passe på at alle fikk komme til ordet og si sine tanker om prosjektet og case. Vi fikk til en hyggelig sammenkomst i starten av prosjektet der alle kunne bli mer kjent med hverandre og spise pizza.

Brukerdokumentasjon

Applikasjonens funksjonalitet

Formålet med appen er å fortelle brukeren om det er utepilsvær eller ikke, altså om det er godt nok vær til å sette seg ut og nyte en øl.

Etter at vi ble ferdig med hovedfunksjonaliteten til appen kan man gå inn på appen og få tilbakemelding om det er utepilsvær eller ikke der man er. Appen henter først informasjon om lokasjonen til brukeren. Deretter hentes informasjon om været på denne lokasjonen ved bruk av dataene til Meteorologisk institutt. Her hentes informasjon om temperatur, skydekke, sol, vind og nedbør. Appen inneholder en algoritme som ved hjelp av denne informasjonen beregner om det er utepilsvær eller ikke. Deretter hentes også data fra Vinmonopolet, og fra en egen øl-database som vi har lagt for å kunne vise øl som kun kan kjøpes på vanlige dagligvarebutikker. Denne egne databasen er til fordi øl som kan kjøpes på butikken ikke er tilgjengelige hos Vinmonopolet. Designet på hovedskjermen oppdateres hele tiden basert på de siste dataene som er hentet.

Dersom værdata ikke er tilgjengelig, får brukeren tilbakemelding om dette. Dette kan for eksempel være tilfelle dersom brukeren ikke har gitt tilgang til appen om å få benytte seg av brukerens lokasjon, eller at enheten ikke har nettverkstilgang.

Når værdata og anbefalinger av drikke er hentet og vist til brukeren, kan brukeren endre datoen på dagen de ønsker å se om det er utepilsvær på. Dette skjer ved at brukeren trykker på kalenderen som ligger rett under teksten som viser hvilken dato det er. Denne kalenderen viser ni dager fremover i tid, siden der er så mange dager vi har data tilgjengelig fremover. Når brukeren trykker på en ny dato, hentes værdata fra en cache i enheten og man får beskjed om det kommer til å bli utepilsvær på denne dagen eller ikke.

Algoritmen som beregner om det er utepilsvær eller ikke tar bare hensyn til temperatur om det er veldig kaldt, i og med at det kan være situasjoner, for eksempel hvis man er på hytta om vinteren, der man ønsker at det er utepilsvær hvis det er fint vær selv om det ikke er veldig varmt. Hvis det er nedbør vil det ikke være utepilsvær, ettersom vi har fått tilbakemelding fra brukerne at det ikke er utepilsvær hvis det er noe som helst nedbør. Vi bruker Sunrise-API-et til Meteorologisk institutt for å få informasjon om soloppgang og solnedgang, slik at det bare er utepilsvær når solen er oppe. Hvis det ikke er værdata tilgjengelig på lokasjonen til brukeren skal det komme en tilbakemelding om dette. Når algoritmen har beregnet om det er utepilsvær eller ikke, presenteres det om det er utepilsvær eller ikke. Vi har laget to «banker» med forskjellige tilbakemeldinger, en for hvis det er utepilsvær og en for hvis det ikke er utepilsvær. Tilbakemeldingen som dukker opp på skjermen er tilfeldig valgt fra den riktige «banken» ut ifra utepilsværet.

Drikkeanbefalingene er basert på temperaturen siden utepilsværet gjerne er det samme; lite vind, skyfri himmel og ingen regn. Vi har sett på en bok ved navn *Øl hele året*(Kvig & Thorbjørnsen, 2020) som gir ølanbefaling basert på blant annet vær og årstid. En av forfatterne av denne boken, Jørn Idar Almås Kvig, er en "ølkjenner" og vi har brukt anbefalingene i boken som hjelp til å lage anbefalingene i applikasjonen vår.

Strukturen i applikasjonen

Bildet viser hvordan appen ble til slutt.

Øverst ligger teksten som først beskriver dag og datoen idag, og oppdateres dersom brukeren endrer dato.

Under dato-teksten ligger kalenderen der brukeren kan klikke på den datoen de ønsker å se om det er utepilsvær på. Her er det 26 mai som er valgt. Datoen som er valgt vil være gul, i motsetning til de andre som er sorte. Dette gjør det enkelt å se hvilken dato som er valgt.

På midten av skjermen vises været i form av en emoji, med en liten beskrivelse under på om det er utepilsvær eller ikke.

Nederst ligger ølanbefalingene. Drikkene ligger i sortert rekkefølge med den mest anbefalte drikken på toppen. Det er mulig å scrolle nedover for å se de andre anbefalingene i listen.

Målgruppe

Målgruppen for appen er alle som liker å drikke øl i fint vær, og ønsker informasjon for å vite når det egner seg best å gjøre dette. Vi har en ganske stor målgruppe. Ettersom det kommer opp anbefalinger av alkoholholdige drikker er aldersgrensen 18 år. Vi har ikke implementert en sjekk for dette, men når man legger ut en app på Google Play Store så er det mulig å legge inn hvilken alder appen er ment for. Dette er noe vi nok hadde gjort om vi hadde publisert appen.

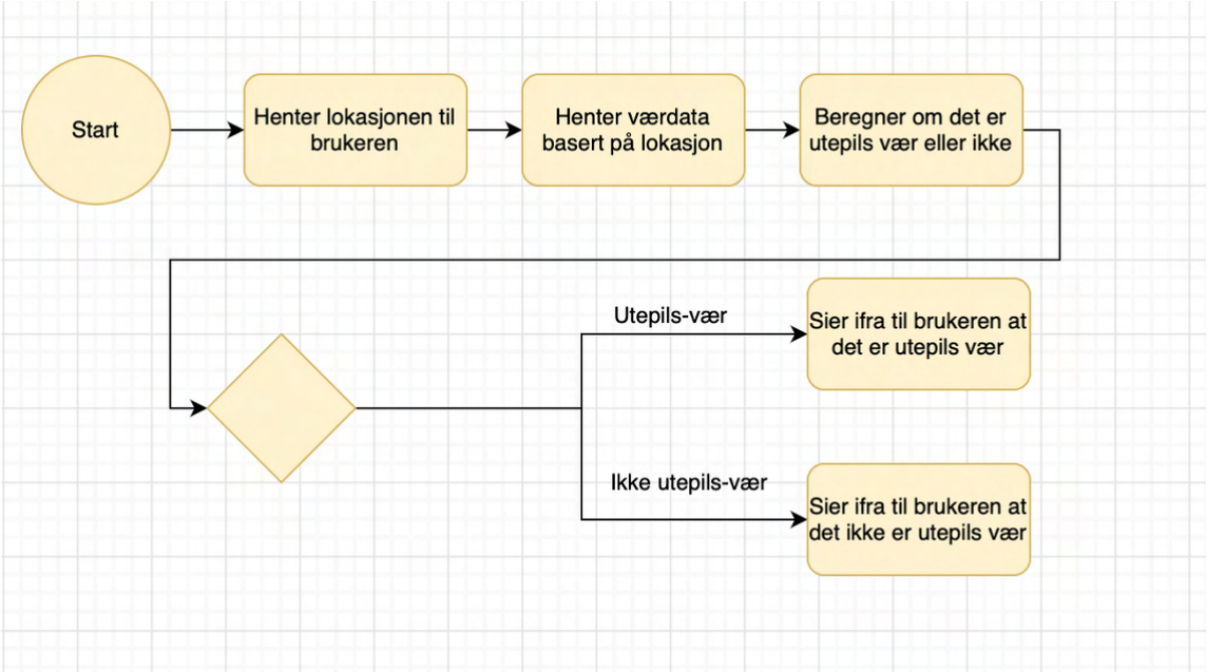
Plattformer

Det er mulig å laste ned applikasjonen fra git eller å laste ned en zip-fil. Appen kan kjøre på alle Android-enheter som har API nivå 24 eller høyere. Dette tilsvarer Android 7.0 Nougat. Den er dermed kjørbar for mer enn 80% av alle android brukere ifølge (AppBrain, 2022).

Kravspesifikasjon og modellering

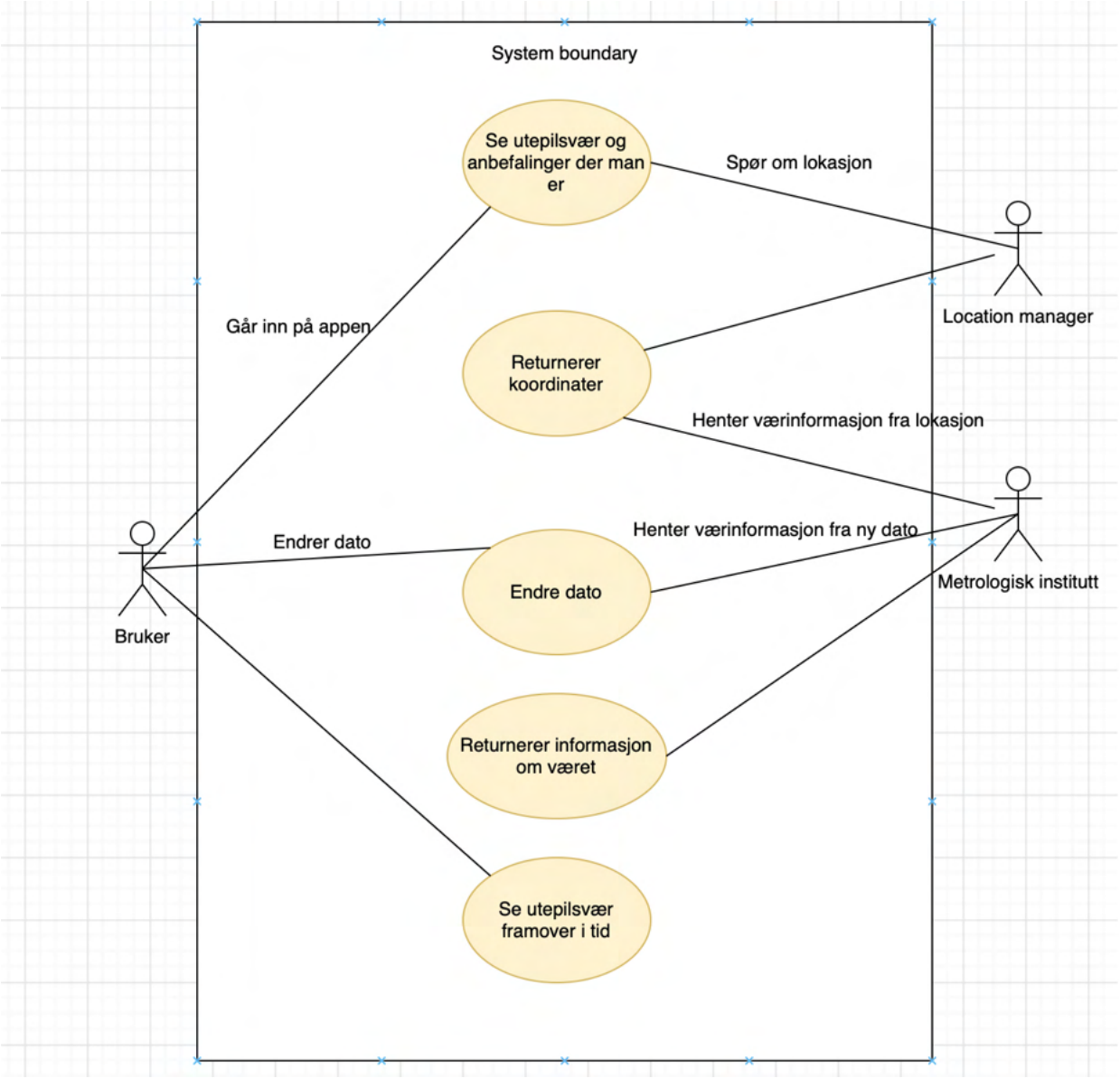
Funksjonelle krav

Vi definerte minimal viable product (MVP) som at brukeren kan gå inn på appen og få beskjed om det er utepilsvær eller ikke på den lokasjonen brukeren befinner seg på. Under kan du se et aktivitetsdiagram over vårt prosjekts minimal viable product. Se på vedlegg 2 for aktivitetsdiagram av hele applikasjonen.

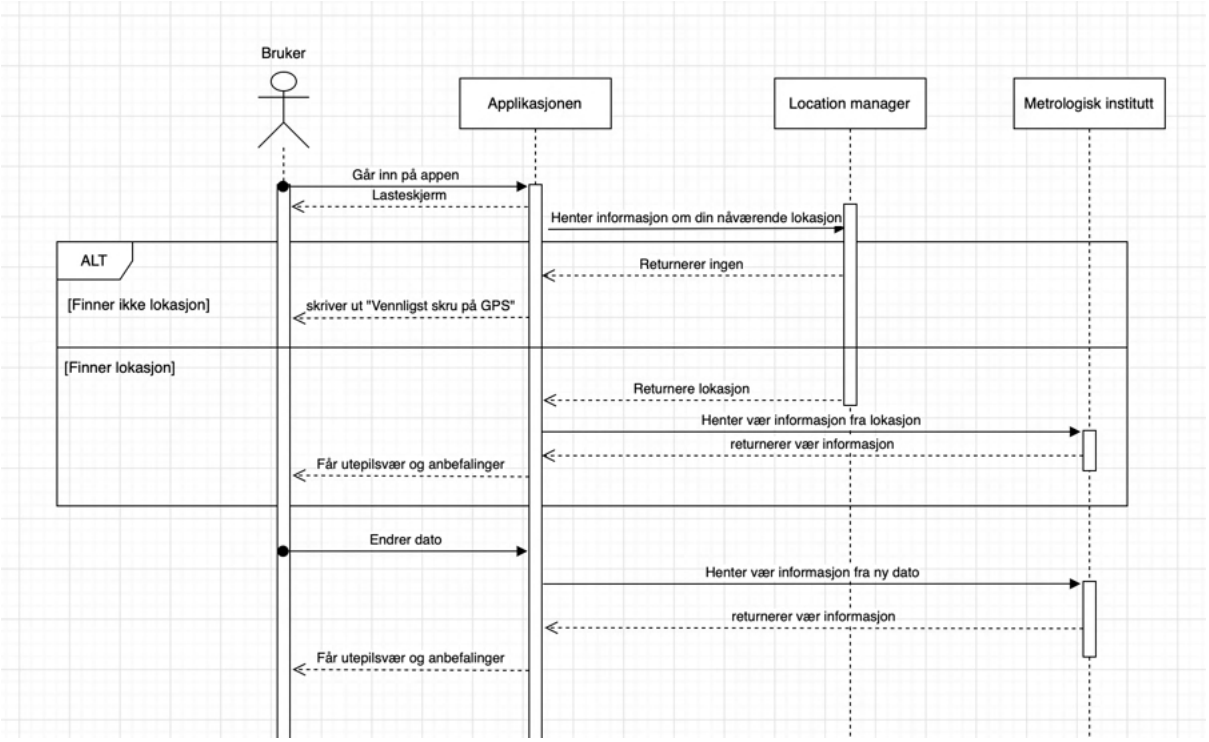


Aktivitetsdiagram over MVP

De viktigste funksjonalitetene i appen er at man på den lokasjonen man er akkurat nå og på lokasjonen man søker opp skal kunne få informasjon om det er fint nok vær til å ta en pils ute eller ikke. Det kommer også opp drikkeanbefalinger basert på været. Det er mulig for brukeren å endre dato slik at man kan se om det er utepilsvær framover i tid. Når man går inn på applikasjonen skal den automatisk hente lokasjonen du er på og vise om det er utepilsvær eller ikke og det skal komme opp drikkeanbefalinger som passer til været. Da vil det komme opp et ikon som viser hvordan været er med en tekstlig beskrivelse under som sier om det er utepilsvær eller ikke. Under ser du et Use Case Diagram av disse funksjonene og et sekvensdiagram for når en bruker logger inn på applikasjonen og endrer dato.

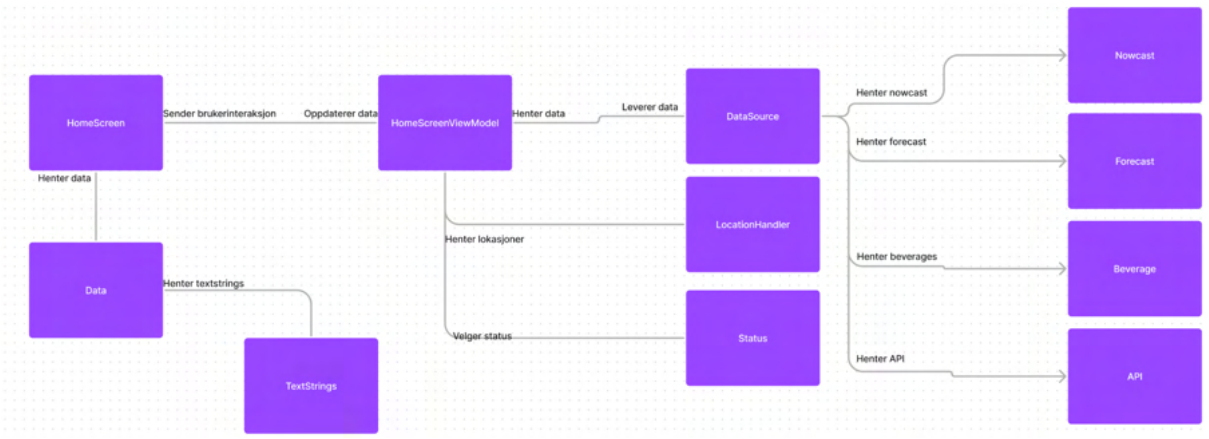


Use Case Diagram over funksjonalitetene i applikasjonen.



Sekvensdiagram for når man går inn på appen og endrer til ny dato

Vi har puttet algoritmen, dataene fra Meteorologisk institutt sine API-er og informasjonen fra Google Maps i egne klasser og funksjoner slik at det er enkelt å implementere nye ting uten at det ødelegger for det som allerede er implementert.
Vi har brukt Model View ViewModel struktur i applikasjonen vår. Vi skriver mer om dette i produktdokumentasjonen. Under kan du se et klassediagram over de ulike klassene i applikasjonen vår. Se vedlegg 4 for klassediagram over alle klassene for informasjonen vi får fra Meteorologisk institutt.



Klassediagram

Produktdokumentasjon

Jetpack Compose

Gruppen var helt fra begynnelsen av prosjektet klare på at det var Jetpack Compose som skulle brukes for byggingen av UI i prosjektet. Dette var fordi vi synes det var kronglete å jobbe med XML i oblig 1 og oblig 2. I tillegg, ønsket gruppen å benytte seg av Jetpack Compose fordi vi ønsket å bruke det som er fremtiden til UI-utvikling for Android, og få erfaring med å jobbe med det når vi hadde mulighet til å velge. Med Jetpack Compose, fikk vi bruke et UI-verktøy som var intuitivt og som vi synes er lettere og mer effektivt enn å jobbe med XML. Siden koden vi skriver i Jetpack Compose kun er Kotlin, slapp vi å bytte frem og tilbake mellom Kotlin og XML. I tillegg, erfarte vi at det kreves vesentlig mindre kode når vi programmerte i Jetpack Compose kontra XML.

Fuel

Som HTTP-bibliotek brukte vi Fuel, som vi fikk gode erfaringer med fra oblig 2. Vi følte Fuel gjorde det enkelt å hente data fra en bestemt URL. Siden Fuel tilbød det vi trengte til prosjektet, var det ikke nødvendig å finne et annet HTTP-bibliotek. Fuel ble benyttet både til å hente værdata fra LocationForecast-, Nowcast- og Sunrise-API-ene, samt henting av data fra Vinmonopolet sitt presse-API, som vi fikk tilgang til etter å ha forespurt via mail. Siden Vinmonopolets API krever en nøkkel måtte dette legges inn i headeren i Fuel, som heldigvis var veldig enkelt å implementere.

Gson

Til konverteringen fra JSON til Kotlin-objekter fra API-responsene brukte vi Gson, som utviklerne, i likhet med Fuel, også benyttet i oblig 2. Gson ga oss en enkel måte å gjøre dette på.

MVVM (Model View ViewModel)

Selve arkitekturen i appen er MVVM. Vi har delt opp implementasjonen av prosjektet i tre deler som inngår i MVVM, nemlig model, view og viewmodel. Her har vi en egen del som er ansvarlig for hentingen av data og selve logikken i prosjektet (model), model er DataSource i vårt tilfelle. I HomeScreen (hovedskjermen i applikasjonen) er ansvarlig for UI, både for det

som vises til brukeren når viewmodel oppdateres, og det som skjer når brukeren interagerer med appen. HomeScreenViewModel, som er vår viewmodel, er ansvarlig for å håndtere data som skal presenteres i HomeScreen. HomeScreenViewModel fungerer også som en link mellom HomeScreen og resten av logikken i applikasjonen, i tillegg til DataSource som er ansvarlig for hentingen av data via API-er i appen vår.

Beskrivelse av løsningen

Når appen åpnes av brukeren for første gang, vil brukeren bli møtt med et pop-up vindu som ber brukeren gi appen tilgang til posisjonen. Neste gang brukeren åpner appen, kommer ikke dette vinduet opp dersom appen allerede har fått lov til å benytte seg av brukeren sin posisjon. Det vil si, at dersom brukeren ikke har gitt tilgang til å bruke posisjonen til appen fra før, ber appen om tillatelse. Dette skjer siden den første funksjonen som kalles når appen åpnes, er HomeScreenViewModel sin fetchLocation. fetchLocation i HomeScreenViewModel sjekker om brukeren har gitt tilgang til å bruke posisjonen. Dersom brukeren ikke har gitt tilgang, kaller fetchLocation på LocationHandler sin metode requestLocationAccess. LocationHandler er klassen i prosjektet som håndterer alt som gjelder lokasjonen til brukeren. Dersom brukeren gir tilgang til posisjonen, kalles HomeScreenViewModel sin fetchData-metode, som henter værdata fra LocationForecast-, Nowcast- og Sunrise-API-ene. Dersom brukeren allerede har åpnet appen tidligere, gitt tillatelse, for så å åpne opp appen igjen, kaller funksjonen requestLocationAcces på fetchLocation i HomeScreenViewModel, som bekrefter at brukeren har gitt tilgang til posisjonen. fetchLocation kaller så på LocationManager sin followLocation som bruker en LocationManager og funksjonen requestLocationUpdates for å finne brukerens posisjon. Funksjonen fetchData i HomeScreenViewModel er en av to funksjoner som faktisk launcher coroutines som kaller på DataSource for å hente data. Denne funksjonen blir kalt fra MainView i HomeScreen.kt. Den andre metoden som launcher coroutines, er fetchReccomendations. Denne metoden henter data fra Vinmonopolet sitt API, og vårt egenlagde, med hjelp fra DataSource. Når data er hentet fra disse metodene, oppdateres variabler i viewmodel seg med data som er relevant for HomeScreen. Siden MainView i HomeScreen hele tiden observerer og oppdateres basert på verdiene i ViewModel, blir UI oppdatert når viewmodel henter nye data. Når data er hentet og vist for brukeren etter åpning av appen, er det neste som skjer, opp til brukeren.

Dersom brukeren trykker på en dato i kalenderen, hentes værdata for den datoen, og vises for brukeren. Dette skjer også i fetchData i HomeScreenViewModel. Anbefalinger for den valgte datoen kommer opp

Drikkeanbefalingene blir hentet via metoden fetchRecommendations som blir kalt i fetchDataWithLocation. Når dataene er hentet, oppdateres viewmodel sine variabler “recommendations”, “dataStatus”, “forecasts”, og “selectedForecast”. Dataene vises for brukeren siden mainview observerer disse variablene.

MainView i HomeScreen observerer variablene forecasts, selectedForecast, location, dataStatus og beverages. Når verdiene i disse variablene oppdateres, oppdateres også UI.

Struktur

Vi har delt opp de forskjellige funksjonalitetene i ulike klasser, som gjør det lett å fortsette å videreutvikle appen. Samtidig har de objektene vi har veldig spesifikke oppgaver, og er i liten grad avhengig av andre objekter. Dette er fordi vi synes det er bedre å eventuelt legge til nye objekter med andre ansvarsområder, slik at det er klart hvilke oppgaver som løses av de ulike objektene.

Funksjonalitet

Vi har definert MVP (minimum viable product) til aktivitetsdiagrammet på side 8. Det appen skulle gjøre, var å be bruker om tilgang til posisjonen, hente værdata fra denne posisjonen, gjøre utregninger basert på disse dataene, og vise frem om det er utepilsvær eller ikke. I tillegg til funksjonalitetene i MVP, er det mulig å velge en annen dato enn den nåværende (opptil 9 dager fremover i tid). Det kommer også opp drikkeanbefalinger på skjermen basert på værdataene som vi henter inn basert på brukerens posisjonen og valgt tidspunkt.

Siden appen gjør disse funksjonalitetene, i tillegg til at vi har implementert drikkeanbefalinger og kalender, gjør systemet det det skal.

Brukskvalitet

Appen er veldig intuitiv og grei å bruke, da det ikke er noe som er uklart for en ny bruker av appen.

Kalenderen er selvforklarende for brukeren, der bruker trykker på ønsket dato.

På skjermen der hvor selve utepilsdataen vises, ser brukeren et vær-ikon, i tillegg til en tekst der det står om det er utepilsvær eller ikke med en kort kommentar under.

På anbefalings-delen er det en tekst der det står "Anbefalinger", med en liste med anbefalte drikker under.

Pålitelighet

Funksjonalitetene som vi har implementert i appen funker som de var tiltenkt, med noen muligheter for forbedring. Blant annet er det ikke mulig å trykke på enter-tasten på tastaturet, men brukeren må i stedet trykke på søk-knappen på det innebygde tastaturet. Dette er et problem som vi har valgt å overse, siden dette kun er noe man trenger dersom man kjører programmet i en emulator på en datamaskin, eller har tilkoblet et tastatur. Dersom appen kjøres på en Android-telefon, vil man vanligvis ikke ha dette.

Når det gjelder Sunrise-API-et, har vi et problem der vi ikke alltid får riktig dato på enten soloppgang eller solnedgang, dette gjør at appen noen ganger tror det er dag når det er natt, og omvendt.

Appen er hovedsaklig ment for å brukes i Norden, og spesielt Norge, men den fungerer også i andre deler av verden. Noen steder, spesielt om brukeren befinner deg langt mot sør eller nord, krasjer appen på grunn av en feil med Sunrise-API-et. Dersom dataene værdataene blir hentet av LocationForecast, og ikke Nowcast, kommer det ikke opp "nå" i den første datoen, men det står heller datoen til den neste dagen. Derimot, hvis det er Nowcast som henter dataene, vil det stå "nå" i den første sirkelen.

Dark Mode / Light Mode

Slik den siste versjonen av appen er, er den i dark mode eller light mode basert på systemets dark-mode innstilling. For videreutvikling er det mulig å bytte mellom dark mode og light mode ved å legge til "(darkTheme = true)" som parameter til metoden UtepilsTheme i MainView (se bildet under)

API-nivå

Det var originalt tenkt at appen skulle ha API nivå 23 for å gi flest mulig antall brukere tilgang til appen. Men siden API nivå 24 kreves for å kunne hente posisjonen til enheten,og derfor byttet vi til API nivå 24. Men selv om vi valgt API nivå 24 så er den tilgjengelig til over 80% av Android brukere.

Algoritmen

Algoritmen er implementert som et companion-objekt til DataSource. Den returnerer true (at det er utepilsvær), dersom visse krav i utregningen av en mengde data blir oppfylt. Dataene som er med i algoritmen er highcloudrate, mediumcloudrate, lowcloudrate, fograte, max_wind_speed, tempRate, percipitationRate og windspeed. Algoritmen er strukturert på en slik måte at det skal være lett å endre på hvor viktig de forskjellige dataene er på utfallet av totalRate (tallet/resultatet av utregningene, som må være over 0.55 for at det skal være godkjent utepilsvær). Grunnen til at vi ønsket en algoritme som var lett å endre på, er at vi har gjort mange observasjoner underveis i prosjektet på hvordan været ser og føles ut, og hvilket resultat algoritmen faktisk gir fra seg. Dersom algoritmen ikke har vært enig med våre observasjoner av været, har vi på denne måten hatt mulighet til å gå inn å endre på algoritmen.

Etter at vi fikk inn flere svar på brukerundersøkelsen, viser det seg at vi har undervurdert betydningen av temperatur, da de fleste har svart at de krever en temperatur på rundt 15 grader på det minste for at de ser på været som utepilsvær. Tanken vår i starten av prosjektet var at temperatur ikke var så viktig, da man fint kan ta seg en utepils på hytta i påsken selv om det bare er 5 grader, dersom det er sol. På bakgrunn av dette, er temperatur noe som vi har tatt mer hensyn til i algoritmen.

ViewModel og DataSource

Vår viewmodel, som vi har kalt for mainViewModel, har to metoder som kaller på DataSource, som igjen henter data fra de ulike MET-API-ene og et API fra Vinmonopolet. Disse metoden har vi kalt henholdsvis for “fetchRecommendations” og “fetchData”. Hentingen av data som skjer i disse metodene skjer asynkront. API-ene fra Meteorologisk institutt innebærer Sunrise, LocationForecast og Nowcast.

Sunrise

Vi har implementert henting av Sunrise-API for å bedømme om det er dag, eller natt. Hentingen skjer i DataSource.kt. Grunnen til at vi bruker Sunrise-API-et er fordi vi har som krav til utepilsvær at solen må være oppe for at det skal være mulig at det er utepilsvær. Tidssonene har vi fått til å funke fint, så klokkeslettene stemmer i forhold til hvilke tidssoner det er vi henter data fra.

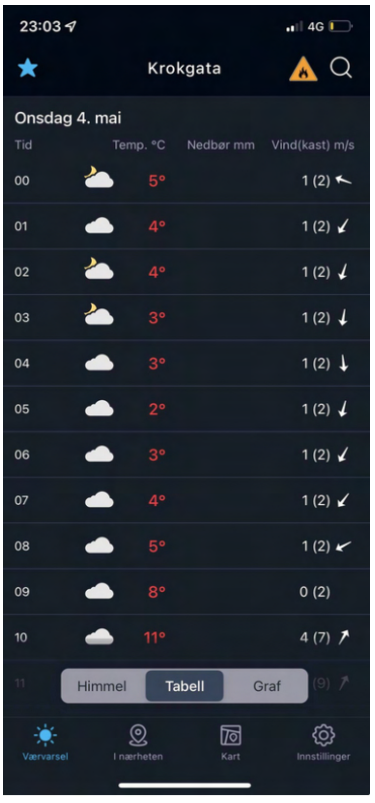
Eksempel:

Her viser Sunrise at solen står opp 20 april kl 15:15:42, og går ned 20 april kl 04:27:22. Problemet her er at solnedgang var 21 april kl 04:27:22. Dermed, når vi sammenligner datoene med nåtiden, får vi ut at det er natt, siden current time ikke er mellom soloppgang og solnedgang. Hadde solnedgang vært 21 april kl 04:27:22, ville vi fått som resultat at det var dag, og det ville ha funket fint.

```
--------- DAYTIME ----------
Sunrise: Wed Apr 20 15:15:42 GMT+02:00 2022
Current: Wed Apr 20 18:44:00 GMT+02:00 2022
Sunset: Wed Apr 20 04:27:22 GMT+02:00 2022
Det er natt!
Total utepilsrate: 0%
Nei, det blir ikke utepils!
```

Utskrift som viser sunrise, current time og sunset

En mulig løsning som vi har sett på for å fikse dette problemet er å bruke symbolene vi henter fra LocationForecast til å bedømme om det er natt eller dag. Tanken bak var at dersom symboltittelen inneholdt "moon", så måtte det være natt. Det viste seg at det heller ikke funket, fordi det kan være overskyet på natta, slik som på dagen, og da vil symbolet kun vise en sky, og ikke noen måne, som på bildet.



Kilde: *yr(2022)*

LocationForecast og Nowcast

I likhet med Sunrise-API-et, skjer hentingen av LocationForecast og Nowcast i DataSource. Highcloudrate, mediumcloudrate, lowcloudrate, fograte, max_wind_speed, tempRate, percipitationRate og windspeed, er dataene fra disse API-ene som vi bruker i appen. Med tanke på videreutvikling, er det mulig å benytte seg av andre dataklasser som appen henter fra API-ene, men som vi ikke har brukt i appen. Data fra LocationForecast-API-et, brukes for å få værdata, dersom valgt posisjon ikke er i Norden, eller dersom man har valgt et annet tidspunkt enn nåtid. Dette har fungert veldig bra så langt, bortsett fra at det var noen komplikasjoner på hvordan vi kan vite om vi kan bruke data fra Nowcast eller om vi må bruke LocationForecast. Nå antar vi at personen befinner seg i Norden.

Vinmonopolet API

I likhet med resten av API-ene, skjer hentingen av Vinmonopolet sitt API i DataSource. Basic, logistics, origins, properties, classification, ingredients, description, assortment, prices, attributes, lastChanged, price, volume, category, kind, alcoholContent, isVinmonopolet og imageURL, er dataen fra dette API-et som vi bruker i appen. Med tanke på videreutvikling, er det mulig å benytte seg av andre dataer som appen henter fra API-et, men som ikke brukes.

Hosting av egen database

JSON filen som inneholder alle Meny sine øldrikker er hostet på Discord, via Discord sin chat logg lenke. Objektet som håndterer etterspørselen heter DataSource.kt i funksjonen fetchBeverages. Dette har blitt gjort fordi Discord ble sett på en mer pålitelig hosting metode enn å hoste vår egne på en privat server, da Discord antakeligvis har mye bedre sikkerhetsmekanismer enn oss. I akkurat dette tilfellet er det en grei løsning siden vi ikke skal gi ut appen. En optimal løsning hadde vært å fått tilgang til Meny sine data direkte, eller opprette et eget domene og hoste det på en sikker server.

Implementering av brukerens lokasjon

Brukerens posisjon er helt sentral for utepilsappen vår, siden det er er funksjonalitet som kreves for at hovedfunksjonaliteten i appen skal fungere. Appen vil ikke kunne vise været ved posisjonen til brukeren dersom appen ikke vet hvor det er brukeren befinner seg. Derfor

var henting av brukerens posisjon og det å be om tilgang til å bruke brukerens posisjon noe av det første vi begynte med i utviklingsfasen.

Appen henter brukerens posisjon ved hjelp av en LocationListener i HomeScreenViewModel. Måten LocationListener fungerer på, er at den har en override-funksjon onLocationChanged. Dersom parameterne til LocationManager sin funksjon requestLocationUpdates ble oppfylt (dersom et minimum antall sekunder har gått og brukeren har forflyttet seg en viss avstand), blir onLocationChanged kalt. Den tar inn et Location-objekt som vi kunne bruke til å sette som brukerens posisjon. Vi kan da bruke det for å finne de værdataene som vi trenger for appen.

Kalender

Kalenderen, som ligger nesten på toppen av skjermen, blir opprettet i funksjonen DateSelector, som blir kalt på i MainView, og tar inn en liste med Forecast-objekter som parameter. Disse forecast-objektene, plasseres inne i en Row med de riktige datoene. Hvert av dato-ikonene i kalenderen, står for hvilken dato den gjelder for.

Anbefalinger

Anbefalingene som kommer opp når appen har hentet inn enten brukerens posisjonen eller søkt på et sted, baseres på en algoritme i Beverage.kt. Listen blir oppdatert basert på hvilke drikker det er som bli hentet i mainViewModel basert på været. Den optimale temperaturen til ølene baseres på alkoholinnhold, friskhet, fylde, bitterhet, tilfeldighet og ulike ord i navnet til ølen. Alkoholinnholdet påvirker den optimale temperaturen litt. Friskhet, fylde og bitterhet påvirker den optimale temperaturen litt mer. Navnet til ølen påvirker den optimale temperaturen mest. Et eksempel på dette er at øl som har ordet “jul” eller “snø” i seg, får en lavere optimal temperatur enn øl som har ord som “sommer” i seg. Dermed har lette, friske øl med lite alkohol en høyere optimal temperatur. Mørke, bitre og tunge øl egner seg mer for kaldere temperaturer.

Loading View

LoadingView er en funksjon i HomeScreen som vi bruker mens data hentes i applikasjonen. Dette skjer når appen åpnes, eller dersom brukeren ønsker å se utepilsværet for en annen

dato. Da kommer det opp et LoadingView der hvor drikkeanbefalingene skal vises. LoadingView kalles fra MainView.

Testdokumentasjon

Arbeidsprosess for enhetstesting

Arbeidsprosess for enhetstesting gikk ganske greit. Da vi jobbet med enhetstesting jobbet en med programmeringen, mens en annen loggførte. Begge diskuterte og snakket mellom seg om hva som burde bli testet og hva som ikke trengtes å bli testet. Denne måten å jobbe på gjorde at det ble veldig godt dokumentert. Men det kan ha gjort at programmeringsdelen har blitt litt svakere enn dokumentasjonen. Det vi kom fram til var at algoritmen og UI var det viktigste og det vi burde teste.

Da vi jobbet med enhetstesting fikk vi en bedre forståelse over hvordan applikasjonen fungerer. Vi klarte også å lage et oppsett av alle dataklassene som vi henter fra MET-API. Vi startet med å hente koden fra main branchen, men det viste seg at main branchen ikke hadde den nyeste versjonen av algoritme objektet som skulle testes.

Arbeidsprosessen for enhetstesting - mer teknisk

Som sagt så var det første problemet vi støttet på at API-kallet ble gjort sammen med kalkulasjonen av utepilsværet. Dette førte til et problem der vi ikke visste om API-kallet som objektet får er riktig og om den er riktig beregnet. Dette kommer av at hvis API-kallet får feil data også setter det i en feil algoritme så kan det gi feil svar enn det som var forventer med den reale dataen. Så da må testet algoritmen og API-kallet være separate. Men vi kan ikke testet API-kallet på noen annen måte enn at vi ikke får null. Det er fordi appen har ingen informasjon om hva været egentlig er utenom dataen den får fra API-kallene. Dette gjorde at det ikke blir mulig å teste API-kallet fordi hvis vi tester API-kallet med et annet API-kall så kan det hende at vi får feil fordi vi tester to feil svar med hverandre. Det er selvfølgelig mulig å teste et nytt API-kall mot et eldre API-kall også ha en begrensning på hvor mye som kan endres over et gitt tidsrom. Problemet med dette er at været kan endres veldig mye over en gitt periode. Samt at da må vi ha en database på det siste kalle som appen har gjort, eller bruke de tidligere dataen vi får fra API-kallet. Men da kan det hende at vi igjen tester et feil API-kall mot et annet feil API-kall. En annen grunn for at vi ikke testet API-kallet er at hvis

vi får data som ikke stemmer med virkeligheten så kommer feilen nok fra MET-API og hvis vi skal sjekke om MET-API stemmer så må vi ha vår egen måte å måle været på. Og det hadde gjort MET-API meningsløst for oss. Samt tenkte vi at MET har nok testet om dataen er riktig før den kommer til oss. Derfor spurte enhetstesting teamet til å splitte disse to funksjonene slik at de kunne bli testet, hver for seg.

Dagen etter så splittet utviklerne API og resultat funksjonene hver for seg i objektet som gjorde det mulig å teste de ulike funksjonene for enhetstesting teamet.

```
@Test
fun isUtepilsTrue() {
    val dataSource = DataSource

    assertEquals(true, dataSource.isUtepils(metObj, metObj))
}

@Test
fun isUtepilsFalse() {
    val dataSource = DataSource

    assertEquals(false, dataSource.isUtepils(metObj, metObjLowTemp))
}
```

Bilde av enhetstesting

Hovedfremværket som ble brukt for enhetstesting var JUnit og det viste seg til å funke ganske bra. Noen problemer som vi møtte på var at frameworket ga en warning for å bruke companion objects som senere enhetstesting teamet sa ifra til utvikler gruppa og hva som skjedde videre ble ikke sagt ifra til enhetstesting gruppa. Enhetstesting gruppen sa også ifra at det ikke spiller noe rolle for å ha med eller ikke companion objektet fordi testene kjører fint med den warningen. Det tok litt tid for enhetstesting gruppen å forstå hvordan man skulle bruke disse @test metodene, men til slutt så klarte gruppen seg veldig fint i en sesjon.

Funksjonene isUtepilsTrue() og isUtepilsFalse() fungere ganske likt. Forskjellen mellom dem er at isUtepilsTrue() putter inn data som vi vet er utepilsvær og dermed skal gi true som svar, og isUtepilsFalse() gjøre det motsatte ved at den tar inn data som vi vet ikke er utepilsvær og

skal bli false. Funksjonene gjøre dette ved å assert at det algoritmen (isUtepils) er det samme som er forventet. metObj er et objekt som vi antar at algoritmen vi gir true. Mens metObjLowTemp er det samme som metObj, men har en lavere temperatur enn det som kreves for å oppnå utepilsvær. Ut ifra dette kan vi sjekke om funksjonen isUtepils vil gi oss det ønskede svaret eller gi oss en feilmelding hvis den ikke er det. En ting som enhetstesting gruppen fant ut da var at temperaturen kunne å være under -50 grader og fortsatt være utepilsvær. Dette ble så diskutert i gruppen dagen etter og gruppen ble enige om at hvordan algoritmen fungerer ikke så viktig det er andre ting av prosjektet som er viktigere. Samt så måtte det være -50 og perfekt vær på alle andre måter enn det for at det skulle bli evaluert til utepilsvær så det hadde bare effekt på en liten andre av tilfeller.

Det neste settet med tester som ble laget etter testene på algoritmen var UI testing. Det først som ble testet var at vær ikonet som vises var det samme som kom fra MET-API-et. Dette ble gjort ved å sette inn et ønsket ikon i UI-en og for å så sjekke om det er det samme som var ønsket. Siden vi vil at testene skal være selvstendige så vil vi ikke at det faktisk skal være avhenger av et faktisk API-kall fordi da vet man ikke hvor feilen er hvis det er en. Fordi da kan det både være en feil i måten vi kaller på og/eller en feil ved å sette inn ikonet.

Så da satt vi inn en klasse hvor vi vet hvilket ikon som skal vises for å så å sjekke UI med en assert funksjon om det er det samme. Her var det et problem som dukket opp fordi enhetstesting gruppen ikke helt visste hva companion object var. Dette er fordi det endte opp med å komme en rekke med feil ved å bruke av companion object. Men etter en dag ved å forstå companion object så endte enhetstesting gruppen med å lage funksjoner som fungerte. Gruppen lagde så to tester, en test som sjekket om ikonet var riktig og en som testen at det ikke viste noe hvis den ikke hadde noe data. Begge disse funksjonen ble sanne. Det eneste problemet var at det ble noen error meldinger som kom av companion object, men alt virker som at det funker så gruppen var fornøyd.

Arbeidsprosessen for integrasjonstesting

Vi har gjennomført integrasjonstester med både Nowcast-API-et til MET og Vinmonopolet sitt API. Dette har vi gjort både ved hjelp av Postman, og mens vi implementerte Nowcast- og LocationForecast-API-ene i appen.

Data som er hentet fra Vinmonopolet sitt API og LocationForecast

På bildet ser til venstre ser man resultatet av hentingen fra Vinmonopolet sitt offentlige API. Bildet til høyre er fra testingen av LocationForecast, og viser resultatet som vi fikk fra det API-kallet.

Prosessdokumentasjonen

Vi begynte med å diskutere hvilket case vi ønsket å jobbe med. Vi hadde mange forskjellige idéer, blant annet sopplukkings app, fjelloverganger og planlegging av reise basert på vær. Idéen som folk syntes virket mest interessant var å lage en app som forteller brukeren om det er utepilsvær eller ikke. Vi var litt usikre på om det var nok funksjoner å implementere slik at denne appen kunne bli nyttig. Vi hadde mange idéer og forslag til ting som kan implementeres, som for eksempel, at med kartet, kan vi ha med mulighet for å velge en annen posisjon enn der du befinner deg, og på den måten sjekke om det er utepilsvær på en bestemt posisjon i for eksempel Spania. Hvis vi også har med mulighet for å velge dato, kan man også sjekke hvordan utepilsværet er på for eksempel neste lørdag på den valgte posisjonen. Andre idéer er drikkeanbefalinger, filtrering av hva slags type drikke man er interessert i (alkoholfritt, øl, vin, cocktails osv.), historiske data for hvilke dager som har vært "utepilsdager", kalender der man kan se hvilke dager fremover i tid det er som kommer til å være "utepilsdager", mulighet til å legge til favoritt-drikker, utepilslogg, som er en

funksjonalitet der man kan registrere når man har tatt seg en “utepils”, bestselgere basert på dato (her kan vi bruke API fra Vinmonopolet) og anbefalinger av drikke basert på været. Da vi hadde snakket om alle disse utvidelsene av appen og fler, kom vi fram til at vi hadde veldig mange gode idéer og at dette dermed kunne bli en interessant app. Ettersom vi bestemte oss for å ha et åpent case som vi var litt usikre på om kom til å bli ansett som en seriøs idé, lagde vi raskt en beskrivelse av caset og sendte den inn for godkjenning.

Etter det diskuterte vi hvordan vi skulle jobbe sammen og hvor mange møter i uka vi skulle ha. Da vi satt alle sammen sammen og jobbet med oblig 3 fikk vi diskutert godt hvordan vi planlegger å gå fremover i prosjektet. Her diskutert vi hvilke verktøy vi ønsker å utnytte, hvordan vi ønsker å jobbe fremover og hvilke forventninger vi har til hverandre i teamet. Vi hadde veldig mange idéer til funksjoner vi kunne implementere, og derfor rangerte vi de ulike funksjonene etter viktighet/enkelthet/nyttighet/humor. I tillegg definerte vi MVP. Dette ga oss en fin oversikt over hva vi måtte begynne å jobbe med og hva som var viktigst å bli ferdig med først. Det er veldig mye informasjon man kan få fra API-ene til Meteorologisk institutt, så vi så på det, og fant ut av hvilke som vi trengte til å kunne beregne utepilsvær. Da vi bestemte oss for hva som var viktigst for om det er utepilsvær eller ikke kom vi fram til at sol, regn, vind og skydekke er de viktigste faktorene. Vi sjekket været som var ute flere ganger og holdt en liten vær «kalender» og sammenlignet det med ulikt vær og hva slags informasjon vi fikk fra Meteorologisk institutt. Da kom vi blant annet fram til at høye skyer ikke påvirket været særlig mye, men at lave skyer har mye å si for om det er utepilsvær eller ikke.

Da vi diskutere hvor mange møter vi skulle ha i uka bestemte vi oss først for kun ha ett møte i uka. Dette gjorde vi en uke før vi fant ut at det var vanskelig å vite hvor andre var i prosjektet og hva som hadde blitt gjort. Samt at det var lite oversikt over hvem som gjorde hva. Derfor bestemte vi oss å ha et hovedmøte på mandager og at fra tirsdag til fredag skulle vi ha korte stand-up-møter. De første få dagene fungerte dette fint. Gjennom ukene så merket vi at mer og mer folk startet å ikke møte opp hver dag på stand-up-møtene. Samt merket vi at det bare var de samme som møtte opp. Dette ble så tatt opp på et av de større mandags møtene, men fordi de som møtte opp var syntes at det var nyttig og at det var en bra ting fortsatte dette en uke til. Men etter en uke så fant vi ut at det ikke fungerte. Dette førte til at vi skiftet fra å ha stand-up-møtene hver dag, til kun å ha de på torsdager. Allikevel ble det fortsatt diskusjon om de møtene var viktige. Dette er fordi mye av det samme fortsatt skjedde. Hvor det var lite

oppmøte og at det bare var de samme. Som en gruppe har vi kommet fram til at stand-up er bra, men de funker ikke hvis ikke alle eller nesten alle møter opp regelmessig.
Vi hadde intervju med en fra Meteorologisk institutt for å få litt mer oversikt og innsikt i hvordan API-ene fungerer og hvordan informasjonen påvirker været. Mer informasjon om dette er i arbeidsprosess for intervju. Vi starter med å implementere Google Maps, hente informasjon fra Meteorologisk institutt og sette dette inn i en algoritme.
Vi opprettet en GitHub og lagde hver vår branch før vi startet arbeidet, sånn at det skal være lett å merge sammen alle delene senere. Å jobbe med branches ble litt kronglete fordi enhetstesting gruppen og utvikler gruppen hadde ulik idé av hvor 'main' branch skulle være. Vi passet derfor også på å bruke samme package name; no.uio.utepils.

På starten av uka til prosjektarbeidet, så startet vi uten stand-up-møter hver dag. Neste uke så tenkte vi å implementere stand-up-møter hver dag, på Discord (kommunikasjonsplattform), der vi gir beskjed på hva vi har gjort, møter opp og snakker sammen om utviklingen av appen og hvilke valg vi bør ta.

Den første uka så tenkte vi å fordele gruppeoppgavene etter vær person, en skulle ta seg av UI, feilmelding ved ingen data, lage MET-klassene, lage et DataSource objekt og implementere utepilsalgoritmen. Vi merket at noen på gruppa ikke helt forsto hva de skulle gjøre og hadde litt annerledes synspunkter på hvordan systemet skulle se ut til slutt. Så en person som hadde bedre kompetanse kom inn i oppgavene og gjorde dem alle selv. Vi hadde en diskusjon om dette den andre uken. I den andre uken ble vi enige om fremtidige oppgaver og prøve å få til daglige stand-up-møter, slik at vi kan rapportere vår progresjon i oppgaven, denne gangen så tenkte vi å sette sammen utviklere til å jobbe sammen med hverandre og ha ansvar for samme oppgave, slik at da kan man lettere få til parprogrammering og ikke føle at oppgaven blir for mye for å håndtere.

Den andre uken gikk veldig greit, alle fikk gjort sine oppgaver og nå lagde vi planer for neste uke, vi ble enige om å fokusere på rapportskriving og tok påskeferien uken etter. Etter påskeferien på vårt fysiske hovedmøte, så diskuterte vi hva -oppgavene videre er. Vi ble litt usikre på hva vi burde prioritere fordi vi manglet en person som hadde ansvar for hvordan appen så ut, så derfor følte vi at det kanskje var noe som manglet før vi begynte på neste oppgave. Uansett så ville vi bli enige om å gjøre noe denne uken og planen var å skrive arbeidsprosessen til denne uka.

Vi tenker det har vært ganske greit med stand-up-møter, som er digitale på Discord, og et hovedmøte hver mandag fysisk. Vi startet hovedsakelig å fordele oppgaver med at hver person tar en oppgave. Men merket at på starten så ble noen av oppgavene vanskelige for alle gruppemedlemmene og derfor endte det opp med at bare en kar med best kompetanse klarte å løse alle av dem gjennom uken. Så til neste uke så tenkte vi å la noen samarbeide på en oppgave slik at man kan enkelt hjelpe hverandre uten at det påvirker den andre oppgaven til den ene personen negativt.

Vi føler at å samarbeidet sammen på en oppgaven førte til bedre forståelse for systemet og at par programmering førte til mer effektivitet.

Hoved-møtene som gikk på mandager ble de første gangene brukt til å diskutere å planlegge hvordan vi ville gå frem med hele oppgaven og hva det var vi tenkte vi skulle fokusere på. Et eksempel på det er fra et av de første møtene hvor vi skulle bestemme oss hvor viktig de ulike funksjonene vi hadde kommet opp med var for appen vår. Vi endte med å lage en liste med alle funksjonene som vi skulle rangere etter hvordan vi skulle prioritere dem. Når vi rangerte brukte vi ulike faktorer som nyttig den er, hvor enkel den er å implementere, hvor viktig den er for at appen skal fungere godt og hvor morsom den er for appen vår. Når vi rangerte så leste vi opp en funksjon og alle skulle deretter notere seg ned et tall fra 1 til 10 uten å vise det til noen andre. Når alle hadde skrevet ned et tall så delte vi tallene med hverandre og begynte å diskutere hvorfor vi valgte det tallet som vi hadde. Etter diskusjonen kunne vi gjøre endringer på tallene våres for så å slå dem sammen til en felles poengsum som ble lagt inn i skjemaet. Dette gjorde vi med alle funksjonene til vi hadde fylt ut hele listen og satt dem i en rekkefølge som viste hvilke funksjoner vi tenkte vi burde jobbe med først. Det var i dette møtet at vi fikk idéene om brukersøk (utepilsvær andre steder), utepilsvær fremover i tid og drikkeanbefalinger, som er implementert i den siste versjonen av appen.

Funksjon	Rating
Utepils-været nå	50/50
Feilmelding ved lite/ingen data	49/50
Appen viser utepils-vær på andre steder	47/50
Utepils-dager framover i tid	43/50
Utepils-algoritmen gir fra seg en prosent istedenfor true/false	46/50
Drikkeanbefaling (øl)	45/50
Representasjon av hvordan utepils-været utvikler seg utover dagen. Kan legge til graf	41/50
Bestselgere (kun polet)	38/50
Faste-helligdager/feriedager	38/50
Ølsalgs-tider	37/50
Drikkeanbefaling (øl, vin, sprit, alkoholfri) (Kun alkoholfri hvis man er under 18)	35/50
Velge drikketype preferanse (øl, vin, sprit, alkoholfri)	27/50
Utepils-logg (krysse av for dagene man har tatt seg en utepils), med streak	29/50
Bruker kan svarteliste drikker/legge til favoritter	24/50

Utklipp fra rangeringen av funksjoner

Av disse funksjonene ovenfor så fikk vi til alt fra "drikkeanbefaling" og oppover. Alle som har en lavere rating enn 42 fikk vi dessverre ikke tid til å implementere. Vi syntes vi fikk til kravspesifikasjonen ganske greit med at vi klarte å implementere de viktigste kravene. Men allikevel så tenkte vi at vi kunne gjort det annerledes ved å inkludere våre brukere.

Under utviklingen så møtte vi på utfordringer, som for eksempel ved at vi lagde kravene uten brukere, dette førte til at vi lurte på om de kravene vi forestilte oss at appen skulle inneholde var faktisk det brukerne av appen vår ville ha. Senere i prosjektet så lagde vi en spørreundersøkelse for å finne ut hva brukerne synes. Appen var allerede fokusert på de hovedkravene vi forestilte oss og derfor uansett hvilke nye innsikt brukerne våre ga oss, som for eksempel å legge til en chat funksjon eller vise ulike steder som man kan ha utepils. Alt dette var møtt med at "det tar for langt tid". Vi senere ville skrape vekk kartet og spørre brukerne om de virkelige trengte det, som forhåpentligvis av oss de kommer til å svar nei til.

Vi startet først med å tenkte at brukeren skulle klikke på en knapp som skulle da vise om det var utepilsvær eller ikke. Senere ble det tatt en beslutning om å vise utepilsvær med en gang appen åpnes av brukeren. Vi tenkte også om vi bør legge til en melding som spør brukeren om de er over 18 år gammel. Det ble diskusjon om det var viktig og om Google Play store hadde en innebygd funksjon derfor trengs det ikke.

Når vi skulle implementere drikkeanbefalinger, så benyttet vi Discord chatten til å skrive at vi trenger noen til å lage en stor JSON fil fra Meny sin liste med øldrikker. Vi hadde benyttet Vinmonopolet sitt API og det som sto igjen var sette sammen disse to til å funke sammen med hverandre i appen vår.

Som sagt tidligere så hadde vi fått tak i Vinmonopolets API, men fordi Vinmonopolet ikke har alle øl som man kan finne i matvarebutikker så bestemte vi oss for å legge dette til også. Det er fordi mange tenker ikke alt på hva som er på Vinmonopolet, men også hva som er på diverse matvarebutikker. Siden vi fant ut av dette relativt sent i utviklingen så bestemte vi oss å bare for å kopiere fra Meny sine sider og lage en egen "API". Dette ble da gjort ved å gå på Meny sin nettside også søke opp øl. Deretter ble det manuelt kopiert dataen som Meny har inn til en JSON som var på samme format som det vi får fra Vinmonopolet. Dette er fordi da kunne vi bruke samme funksjon for å lese dette API som vi brukte på å leste Vinmonopolet sitt API. Hvis dette hadde kommet tidligere så kunne det ha vært mulig å spørre Meny eller en annen butikkjede om å få tilgang til deres API for å bare hente detaljene direkte fra dem.

I lagingen av algoritmen så startet vi med at algoritmen og API-kallet fra Meteorologisk institutt var sammenslått, dette førte til at vi ikke kunne teste om algoritmen gjorde jobben sin riktig. Vi splittet kalkuleringen og API-kallet som algoritmen hadde innebygd og da kunne teste algoritmen onkelig. Vi testet hva som skjedde om algoritmen ikke hadde fått noe data og om en feilmelding hadde skjedd. Vi har fikset at det kommer feilmelding. Andre ting vi kunne ha testet var mer hvordan algoritmen sammenslår seg med begge to API-kallene. Fordi algoritmen bruker Nowcast og LocationForecast API så burde vi også ha sett på hvilken API som dominerer over den andre og om den klarer å skifte verdidata til det andre API-et om det ene ikke får noe som helst respons.

For å gå i detalj av hvem som arbeider med hva, så har vi alle prøvd å fordele oppgaver likt over alle. Mathilde og Trym tok ansvaret for brukerundersøkelsen og intervjuprosessen med

det meteorologiske instituttet, Ole og Johannes tok ansvar for implementasjonen og UI, Nikita og Patrick hadde ansvar for enhetstesting.

Verktøy

Underveis i prosjektet benyttet vi oss av flere ulike verktøy og nettsteder. Det vi brukte aller mest var en nettside kalt Notion. Det er en programvare for prosjektledelse og notering. Den har mange funksjoner som har hjulpet oss gjennom prosjektet. Blant annet fikk vi laget en forside med informasjon om gruppemedlemmer, veilederne, kalender med planer vi har satt, andre verktøy vi har brukt og en liste over hvordan vi går fram i prosjektet. I tillegg har vi brukt Notion sitt kanban brett for å lage en utviklingsplan som viser fordelingen av oppgavene vi skal gjøre, hvordan vi ligger an i oppgavene og hvem som skal gjøre den, samtidig som vi kan notere på oppgaven hva vi gjør. Du kan se et utklipp av hvordan vi brukte Notion sitt Kanban brett under. Vi laget også en egen side hvor vi hadde alle møteloggene våres på ett sted. Vi opprettet også flere sider for ulike notater som for eksempel værlogg hvor vi noterer værdata.

Når det kommer til kommunikasjon, brukte vi Discord. Vi opprettet en gruppe i Discord hvor vi hadde ulike kanaler for å snakke om forskjellige ting, som for eksempel en kanal til UI, en til datainnsamling og en til generell snakk. Vi følte at kommunikasjon gjennom Discord funket helt strålende og at mange klarte å følge med på hva som skjer i chatterommet. Vi brukte også Discord til digitale stand-up-møter og merket at noen ikke møtte opp på disse møtene, men grunnen var mindre på grunn av hvordan Discord er designet men mer på grunn av andre faktorer i hverdagen.

GitHub, vi brukte GitHub for kodeskikk og bevaring av koden. GitHub er et kode oppbevarings verktøy for å passe på at utviklere som jobber på koden samtidig ikke overskriver hverandres data. Måten vi brukte GitHub var å dele opp våre oppgaver i branches, (lage UI, algoritme & enhetstesting), slik at vi enkelt kan merge sammen branches til vår main branch når vi er ferdige med koden.

Vi utnyttet et designverktøy kalt Figma til å designe ulike utseender for appen vår før vi hadde laget selve appen, slik at vi kunne vise frem hvordan vi så for oss at vi ville appen

skulle se ut. Vi brukte også et verktøy kalt FigJam til å lage ulike diagrammer for å vise hvordan ulike aspekter ved appen vår fungerer eller hvordan vi ser for oss at den skal fungere, som for eksempel et diagram som vi lagde i starten som viser hvordan vi kunne bruke API-ene vi har funnet til ulike funksjoner vi har tenkt på. Her har vi da API-ene som gule bokser, med piler til blå bokser som er funksjoner, også har vi grønne bokser til funksjoner som ikke henger sammen med et API.

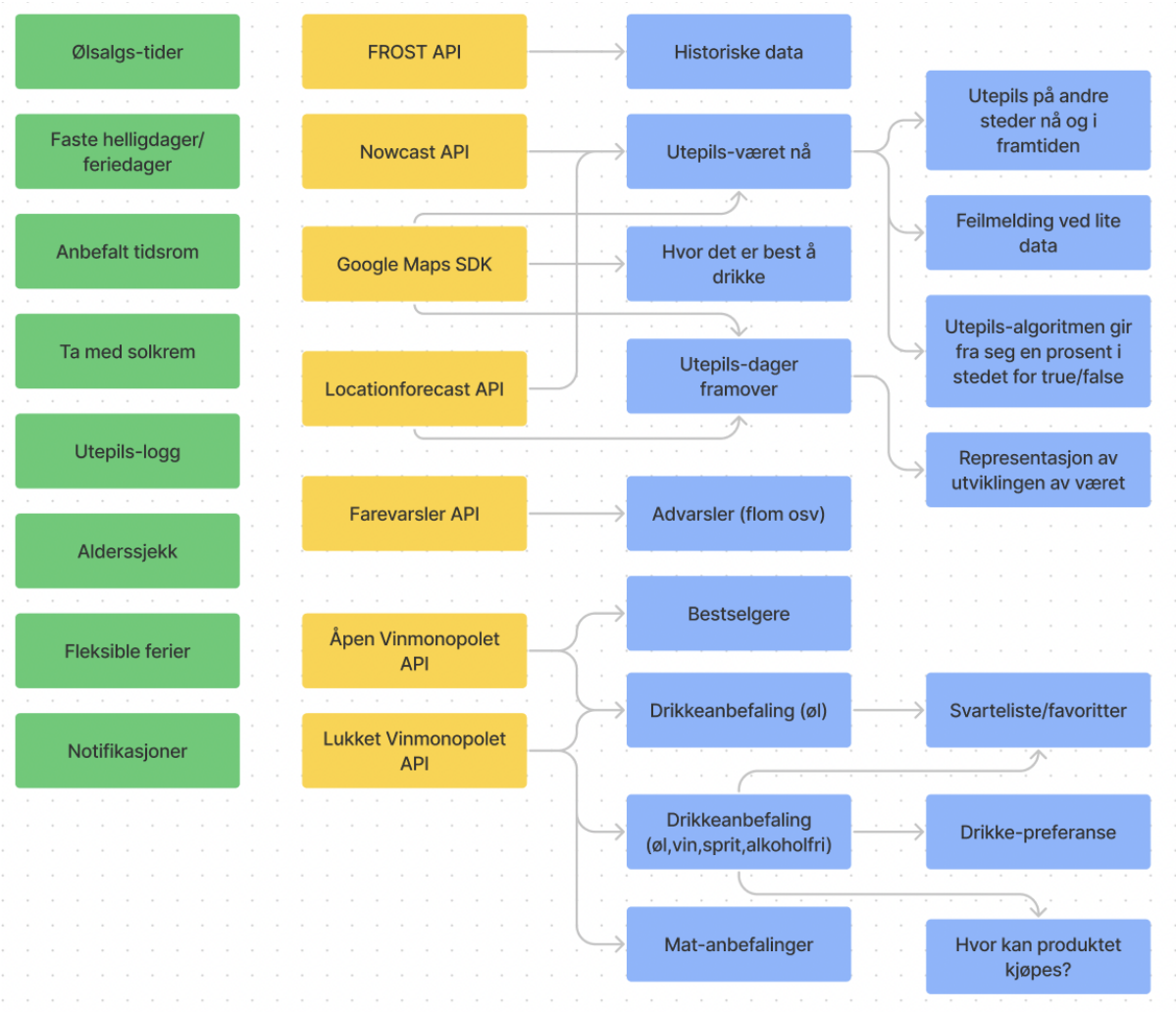
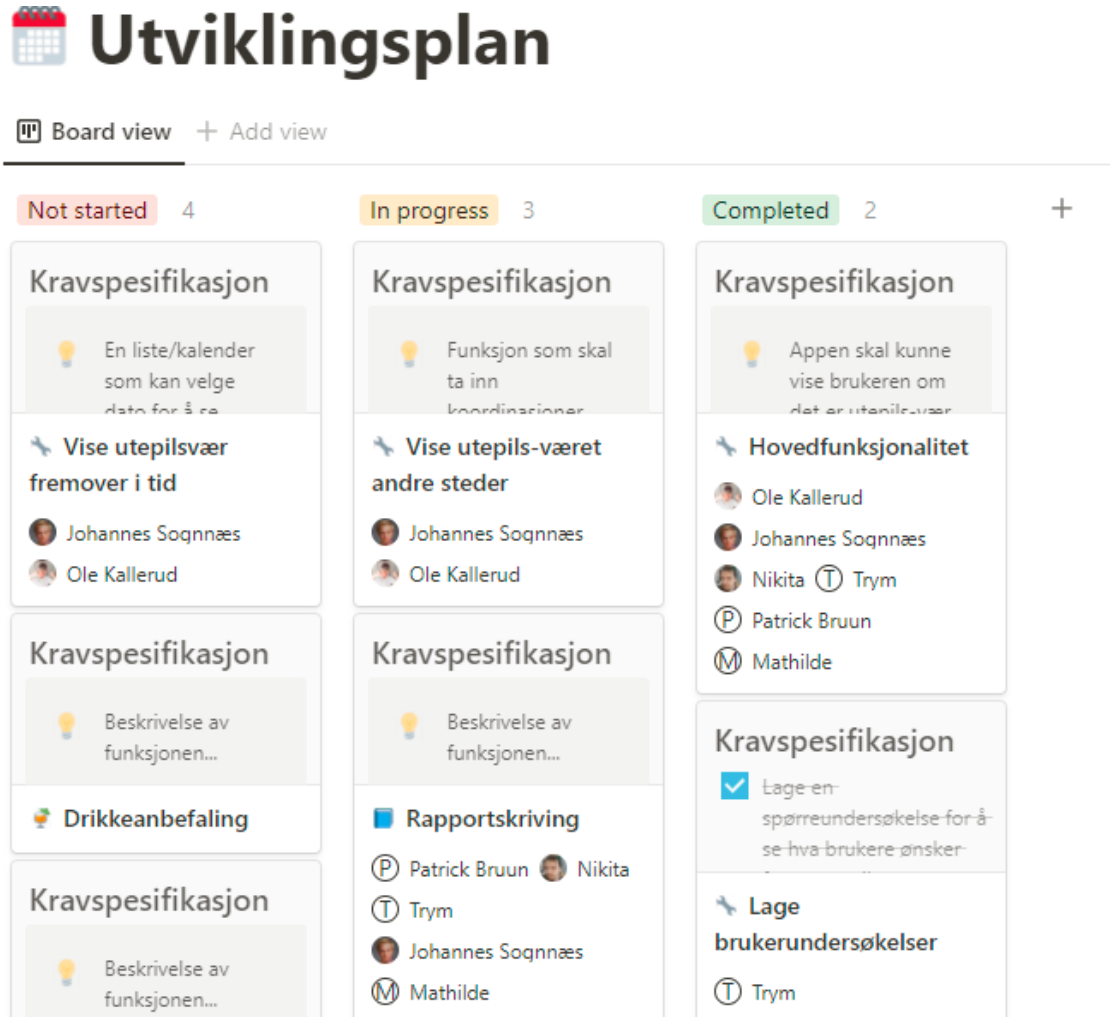


Diagram over funksjoner og API opprettet i FigJam.



Utklipp fra kanban-brettet i notion underveis i prosjektet

Rapportprosessen

Vi startet ganske sent på selve rapportskrivingen, så selve rapporten tok litt tid å få i gang, men vi begynte helt fra starten å loggføre alt vi gjorde i oppgavene våres. Både hvordan vi gikk frem og hva resultatene ble. Dette gjorde at vi hadde en god start når vi først begynte på rapporten vår fordi vi hadde allerede skrevet kort, men informativt om prosessen vår. Da var det ikke noe problem å få skrevet mer utfyllende i rapporten, selv om det var lenge siden vi hadde gjort det vi skrev om. Noe kunne ha vært viktigere til en lignende oppgave i fremtiden kunne ha lagt mer fokus på rapporten tidligere for å få mindre å skrive på slutten og mere spredt ut over arbeidsperioden.

Vi prøvde å loggføre alle våre møter, både fysiske møtene og digitale. Samtidig så loggførte vi vår prosess i de oppgavene vi skulle gjøre. Dette tenkte vi ville hjelpe når vi skulle skrive rapporten. Gjennom å ha lest alt loggføringen så fikk vi brukt veldig mye av den til rapporten,

men vi merket også at noen ting gjentar seg i loggene. Vi merket også at noen var bedre på å loggføre enn andre som gjorde at det var lettere å skrive om noen ting enn andre. Hvis vi alle hadde hatt loggført like godt så hadde det nok vært lettere å skrive om de tingene som ikke var så godt loggført.

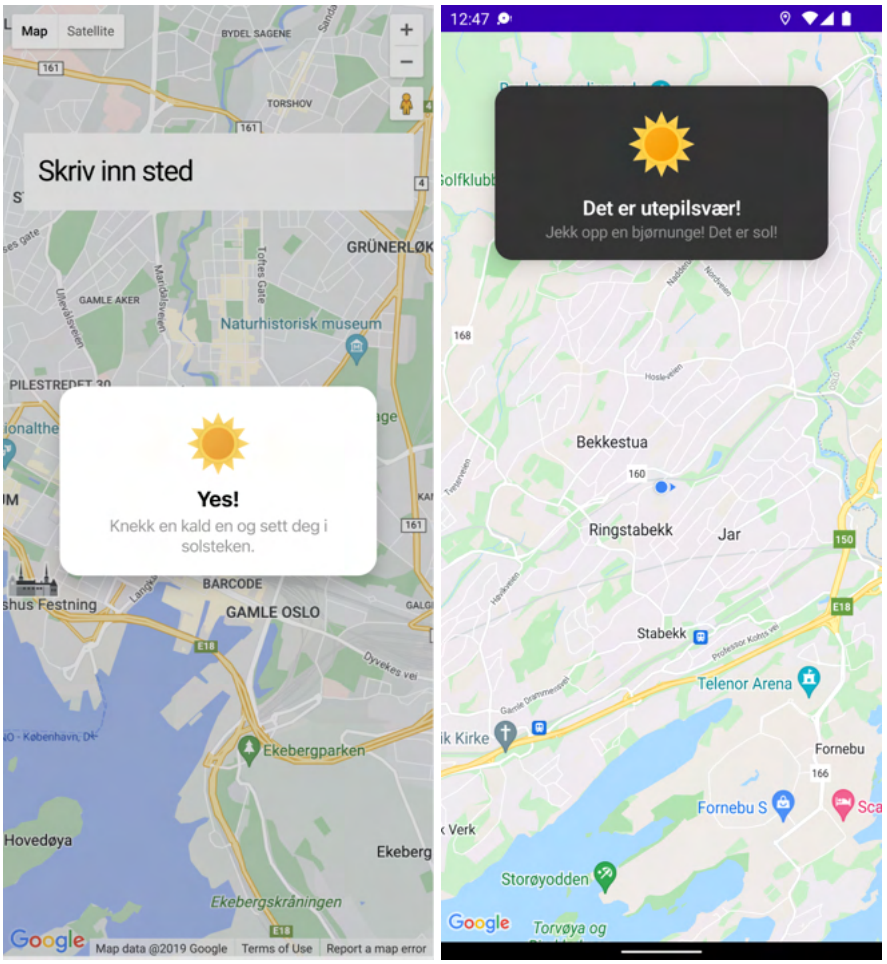
Arbeidsprosessen for utvikling av appen

Da vi testa Vinmonopolet sitt API i postman i starten av prosjektet etter at vi hadde fått idéen om å hente drikker derfra, fant vi ut at det ikke var nok informasjon på det offentlige API-et. Derfor bestemte vi oss for å se på de andre API-ene til Vinmonopolet og finne ut av hvilket API som inneholdt den informasjonen vi trengte. Vi kom da fram til at vi trengte tilgang til Presse-API-et for å kunne hente nok informasjon til å vise fram drikkevarer derfra i applikasjonen vår. Vi sendte dem en mail der vi forklarte at vi hadde et prosjekt og ønsket tilgang til deres Presse-API. Vi fikk svar om at dette var greit så lenge vi ikke publiserte appen offentlig, og dermed fikk vi tilgang til Presse-API-et deres.

I starten av utviklingsprosessen lagde vi et par forskjellige branches i GitHub med ulike formål/deloppgaver. Vi hadde da branchene maps, develop og main. Der var tanken at maps SDK skulle implementeres i maps-branchen, mens henting av API, algoritme og UI ble gjort i develop-branchen. På den måten kunne vi merge disse branchene sammen når hver enkelt branch var klar. Dette funket fint. Når vi begynte med enhetstesting, opprettet vi branchene enhetstest og enhetstest-fra-develop som var for nettopp enhetstesting. Deretter opprettet vi branchen Compose-Maps, siden utviklingen i maps-branchen var gjort i XML for å få en forståelse av maps SDK og ulike funksjonaliteter. Etter det, opprettet vi Compose-Maps, der maps skulle skrives om til Jetpack Compose. Dette viste seg å bli veldig vanskelig, da Compose Maps er svært nytt, og funksjonaliteten viste seg å ikke være helt tilstrekkelig enda.

I starten av utviklingen brukte vi FusedLocationProviderClient for å hente den siste kjente lokasjonen til brukeren etter at vi hadde bedt om å få bruke brukeren sin posisjon. Dette funket ikke helt optimalt, da appen ikke var i stand til å hente den siste posisjonen til brukeren den første gang appen ble åpnet på en ny device. For at dette da skulle funke, måtte vi nemlig starte appen en gang til etter den første gangen for at appen fikk tak i den siste posisjonen til brukeren. Siden vi ikke så på det som en superviktig bug å fikse i starten av utviklingsfasen, valgte vi å ignorere den i starten, og heller fokusere på å få opp data på

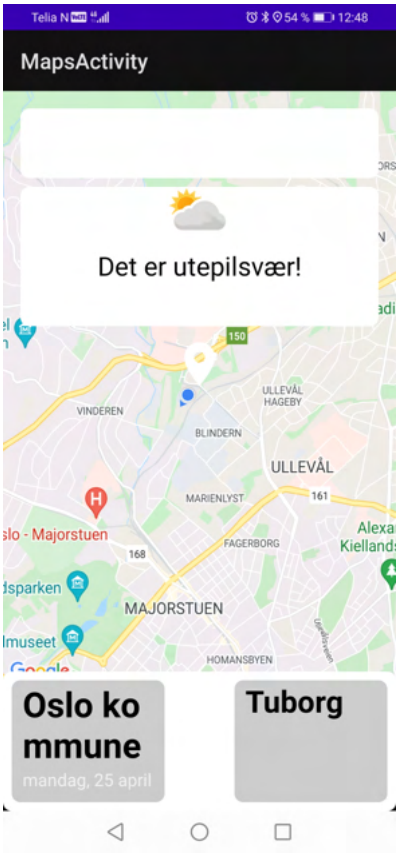
skjermen. Da vi kom tilbake til denne problemstillingen, fant vi ut at en mulig løsning på problemet var å bruke interfacet LocationListener, som bruker devicen sin GPS til å finne posisjonen.



Bilder fra appens utseende i starten av prosjektet, med kart

Når vi var kommet godt i gang med prosjektet og hadde fått med mye av av de viktigste funksjonalitetene, så hadde vi en godt fungerende app, men vi var ikke helt fornøyde. Foreløpig i appen kan brukeren søke opp den lokasjonen man ønsker og se om det er utepilsvær eller ikke der. Nederst til venstre på appen er det en boks der man kan se posisjonen man har valgt og dato. Klikker man på denne boksen kan man endre datoen til den datoen man ønsker å se utepilsværet på. (Det er begrenset hvor lang tid fremover man kan se, fordi det bare er så så mye informasjon fremover i tid tilgjengelig fra Meteorologisk institutt.) Nederst til høyre ligger det en boks som man kan klikke på for å få opp øl anbefalinger basert på været. Her vil man få opp et bilde av ølen som er anbefalt, samt en kort beskrivelse.

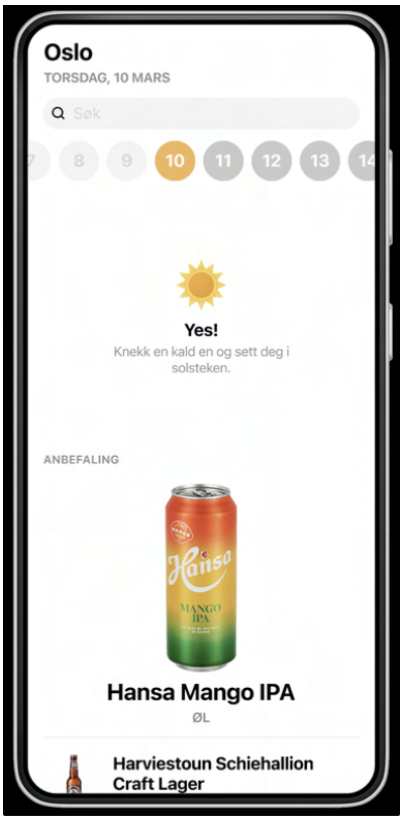
Dette prøvde vi så å får på en vi så på en fysisk enhet. Dette ble gjort ved å koble en ledning til en PC som hadde den mest oppdaterte versjonen av appen da. For å så teste hvordan den fungerte på en faktisk enhet kontra en emulator. Det vi fant ut var at posisjonen oppdaterte se mens enheten ble beveget. Det ble så testen om den kunne oppdater været hvis enheten ble flyttet til et sted med andre værforhold. Dette viste seg at appen taklet begge disse testen. Noe som var forventet var at design så annerledes ut på appen. Dette var noe som ble med på å endre designet til noe annet. Under kan du se hvordan det så ut på den fysiske enheten.



Bilde fra appen omtrent midt i prosjektet. Rett før vi valgte bort kart.

Vi fant til slutt ut at vi kanskje ikke trenger kart i appen våres i det hele tatt, både fordi selve implementasjonen var komplisert, og at kartet gjør utseende på appen litt uryddig. Vi følte også at kartet tok bort fokuset fra det vi synes er viktigst i appen, nemlig beskjeden om det er utepilsvær på din lokasjon i dette øyeblikket, eller ikke. Dette var også noe vi fikk tilbakemelding om i brukerundersøkelsen.

Dermed gikk vi for et design uten kart:



Med dette designet får vi med de viktigste funksjonalitetene som vi trenger i appen våres, og som er høyt oppe på lista over funksjonaliteter som både vi og tilbakemeldinger fra brukerundersøkelsen ønsker:

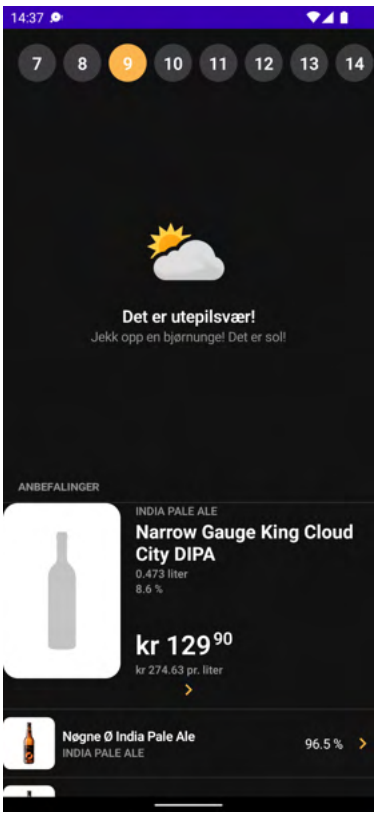
- Ryddig design uten kart som tar for mye oppmerksomhet
- Viser hvor du befinner deg, og dato med tid
- Viser utepilsværet
- Viser anbefalte drikker
- Det eneste som blir borte med dette designet er at brukeren kanskje ikke får en like enkel måte til å se område som de befinner seg i, og at man mister muligheten til å markere lokasjon i kartet. Men siden vi synes at det er viktigere å kunne søke etter ønsket lokasjon i stedet for å plassere en markør på kartet, synes vi fordelene var større en ulempene når det gjaldt å ha med kart eller ikke.

Denne avgjørelsen om å gå med et design uten kart kom ganske sent i prosessen, men siden implementering av algoritme og henting av værdata allerede er implementert og funker som det skal, tror vi at det skal være relativt greit å utvikle det nye designet.

Det betyr, at når vi bestemte oss for å gå over til et design uten Google Maps, var det kun hovedfunksjonaliteten som var fullstendig på plass. Det vil si at vi fikk opp på hovedskjermen om det var utepilsvær eller ikke på brukerens lokale posisjon. Dette krevde derfor at brukeren hadde gitt tillatelse til å benytte posisjonen sin.

Dermed var det fremdeles en del ting som manglet før appen var på plass:

I tillegg, måtte vi implementere kalenderen, som gir brukeren mulighet til å velge dato. Dette skulle gi brukeren mulighet til å velge en dato på opptil ni dager fremover i tid, for å kunne se hvordan utepilsværet kom til å være. Tanken var at kalenderen og søkefeltet jobbet sammen, slik at resultatet (utepilsvinduet) hele tiden var oppdatert basert på hvilken dato som var den siste som ble valgt av brukeren, og det siste stedet som brukeren hadde søkt på. Dersom brukeren ikke hadde søkt etter et sted i søkefeltet, skulle brukerens posisjon benyttes for hentingen av utepilsværet. Siden søkefeltet ikke ble med i den siste versjonen av appen, er det kun brukerens nåværende posisjon og den valgte datoen som utgjør værdataen som blir vist.

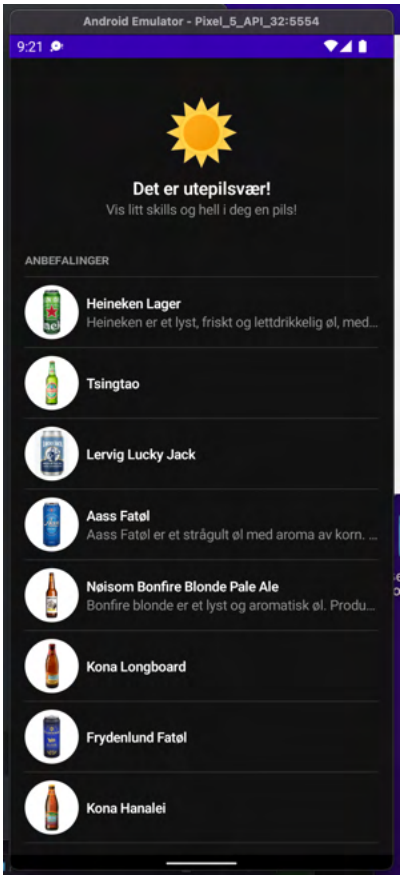


Bilde fra appen nærme slutten av prosjektet, de fleste funksjoner er på plass

Til sist, manglet vi drikkeanbefalingene, som skulle gi brukeren anbefalinger på hvilke drikker som var anbefalt basert på en algoritme som regnet ut de mest egnede drikkene basert

på værdataene som vi hentet inn. Vi fikk til å vise drikkeanbefalinger og har implementert i en versjon at det er mulig å klikke på anbefalingene. En ide vi hadde i forhold til det, var at brukeren ville bli tatt til nettsiden som selger drikken, når brukeren trykker på en anbefaling. Nettsiden vil være enten fra Vinmonopolet eller meny. Dessverre så ble det vanskelig å implementere sammen med andre funksjoner så derfor skrapet vi vekk muligheten på å klikke på anbefalingene.

Vi valgte at JSON filen som skulle inneholde alle Meny sine øldrikker skulle bli hostet på Discord, fordi den andre muligheten var å hoste den på serveren til en av våre gruppemedlemmer. Siden dette kunne ha vært ustabilt, brukte vi heller Discord.



Bilde av øl-anbefalingen til appen, sent i prosjektet

Vi var ikke helt ferdig med appen siste dag. Dette gjorde at noen deler av rapport måtte endres litt ettersom det ferdige produkt endret seg utover dagen. Funksjonalitetene vi prøvde å implementere fungerte hver for seg, men vi fikk problemer da vi skulle merge de sammen.

Vi har en fungerende versjon der man kunne søke opp andre steder, men hvor det ikke var noen kalender, altså der man ikke kunne se utepilsværet fremover i tid. Vi har en annen versjon der kalenderen fungerer og man kan se utepilsværet fremover i tid, men hvor det ikke er mulig å søke opp andre steder for å se utepilsvær der. Da vi prøvde å merge disse to versjonene fikk vi flere problemer. Derfor endte vi opp med en versjon uten søkefelt.

Søkefelt fra gammel versjon

Et av problemene som vi ikke fikk tid til å fikse, var å merge søkefeltet sammen med resten av appen. Teksten i kursiv under er delen av produktdokumentasjonen som handlet om søkefeltet fra da vi hadde en versjon av appen der søkefeltet fungerte.

Implementeringen av søkefeltet som skal gi brukeren mulighet til å søke etter en ønsket lokasjon var ganske rett frem. Det er et TextField som sender teksten som er skrevet inn når brukeren trykker på søk-knappen på tastaturet. Denne tekststrengen blir så sendt til LocationHandler sine funksjoner getSearchedLocation og getLocationLatLng. getSearchedLocation bruker Geocoder til å finne lokasjoner basert på teksten som er sendt inn fra søkefeltet. Stedet som returneres fra getSearchedLocation blir sendt til HomeScreenViewModel sin variabel place. Tekstfeltet øverst på skjermen oppdateres hele tiden til verdien til HomeScreenViewModel sin variabel place. Dette var sånn det fungerte da vi hadde denne funksjonaliteten i en egen branch, men det ble komplikasjoner når vi prøvde å merge funksjonaliteten inn i resten av appen. Derfor er ikke søkefeltet en del av den endelige appen.

getLocationLatLng bruker også Geocoder til å finne koordinatene til tekststrengen som er skrevet inn i søkefeltet. Deretter kalles HomeScreenViewModel sin fetchDataWithLocation som henter værdata med koordinatene som et av argumentene. Denne funksjonen henter værdata basert på koordinatene som blir tatt inn som parameter. Siden værdataene på hovedsiden hele tiden oppdateres basert på HomeScreenViewModel sine variabler, oppdateres skjermen til å inneholde data fra den lokasjonen som brukeren har søkt på i søkefeltet.

HomeScreenViewModel sin updateDate blir også kalt for at den andre teksten på skjermen skal bli oppdatert til å inneholde datoen til den oppdaterte posisjonen.

Dersom brukeren skriver inn et sted i søkefeltet og trykker på søke-knappen på tastaturet, kalles viewmodel sine funksjoner updatePlace, updateDate og fetchDataWithLocation. Funksjonene getSearchedLocation og getLocationLatLng i LocationHandler kalles også. Funksjonen getSearchedLocation bruker Geocoder til å finne en lokasjon basert på det brukeren har skrevet inn i søkefeltet. Funksjonen omgjør strengen som er skrevet inn i søkefeltet til en liste med adresser ved hjelp av metoden getFromLocationName. Deretter, sjekker metoden getSearchedLocation først om adressen ligger i Norge. Dersom den gjør det, returnerer funksjonen byen som adressen tilhører. Dersom adressen ikke ligger i Norge, sjekker metoden om adressen inneholder en "locality", og returnerer denne. Hvis adressen ikke har en locality, sjekker metoden om adressen har en "subAdminArea". Dersom adressen har en subAdminArea, returneres denne. Dette var den rekkefølgen av deler av address-objektet som ga mest resultat. Hvis ikke, returneres "no data". Når mainview får tilbake stringen som inneholder stedet som er funnet, kaller den på viewmodel sin updatePlace, som oppdaterer variabelen place. Siden mainview hele tiden observerer denne variabelen, vises stedet i teksten øverst på skjermen for brukeren. Drikkeanbefalingene blir hentet via metoden fetchRecommendations som blir kalt i fetchDataWithLocation.

Vi manglet teksten på toppen som beskriver hvor brukeren befinner seg når brukeren går inn i appen, og som skulle endres til det stedet brukeren hadde søkt på i søkefeltet. Teksten under som viser datoen manglet også. Vi endte opp med å ikke ha med teksten som beskriver hvor enheten befinner seg, og i stedet kun ha med datoen.

Da søkefeltet var implementert, var det ikke mulig å trykke på enter-tasten på tastaturet, men brukeren må i stedet trykke på søk-knappen på det innebygde tastaturet. Dette var et problem som vi valgte å overse, siden dette kun er noe man trenger dersom man kjører programmet i en emulator på en datamaskin, eller har tilkoblet et tastatur. Dersom appen kjøres på en Android-telefon, vil man vanligvis ikke ha dette. I tillegg, var det en tekst helt på toppen av skjermen som beskrev hvor det var brukeren befant seg. Denne teksten er ikke med i den siste versjonen av appen. I denne gamle versjonen med søkefelt, var det også andre metoder i LocationHandler, som ikke lenger finnes, som for eksempel getLocationLatLng som returnerte koordinater ut ifra det brukeren hadde søkt på. fetchDataWithLocation er den funksjonen i viewmodel som i den siste versjonen kun heter fetchData. Metoden getSearchedLocation tok inn strengen som brukeren hadde søkt på, og brukte en geocoder til å finne en faktisk adresse basert på søket til brukeren.

Bildet under er versjonen der søkefeltet var implementert med Jetpack Compose. Grunnen til at søkefeltet er plassert såpass høyt oppe på skjermen, er at det ikke vil komme i veien for tastaturet. Det var planlagt at versjonen skulle merges sammen med resten av appen, men dette skjedde såpass sent i prosessen, ble vi nødt til å ta til takke med en app uten søkefelt.

Vedlegg 5 viser sekvensdiagrammet som viser appen med implementasjon av søkefelt.

Versjonen med søkefelt

Hvordan appen kunne ha sett ut

Vedlegg 6 viser hvordan appen ville ha sett ut dersom vi hadde klart å merge versjonen som hadde et fungerende søkefelt sammen med den siste versjonen av appen, som vi leverte. Grunnen til at vi ikke fikk til å koble disse versjonene sammen, var at implementeringen av søkefelt kom såpass sent i utviklingsfasen at vi ikke fikk nok tid.

Universell utforming

Vi har prøvd å ta hensyn til universell utforming under hele utviklingen av applikasjonen. Selv om vi ikke har laget en app som er brukbar for absolutt alle, så har vi hatt flere idéer og tanker underveis i prosjektet som skulle hjelpe med å gjøre appen mer universelt utformet. Noe vi fikk implementert var gode fargevalg. Ifølge WCAG sine retningslinjer er det et krav

om å ha store nok kontraster til at ting er godt synlig. Det har vi tenkt på når vi utviklet appen og vi har da brukt svart tekst på hvit bakgrunn og omvendt. Vi har sørget for å ha store kontraster slik at det har blitt god leservennlighet.

En annen fargerelatert funksjon vi tenkte å implementere var muligheten til å velge mellom light og darkmode slik at man kan velge lyse farger eller mørke. Dette var noe vi fikk et ønske om fra brukerundersøkelsen. På grunn av tid fikk vi ikke implementert dette, men applikasjonen følger enhetens sine dark mode innstillinger. Dette gjør at folk kan endre innstillingene sine på enheten sin, og dette kan være nyttig, blant annet for personer som bruker dark mode for å se teksten bedre, eller for å forhindre hodepine. Vi har også valgt å bruke ikoner fra yr som viser hva slags vær det er, ettersom disse er synlige, enkle og informative for været.

Når det kommer til øl-anbefalingene har vi begynt med å tenke på universell utforming ved at vi har lagt til både glutenfritt og alkoholfri øl, som ble ønsket i spørreundersøkelsen. Vi har også hatt flere idéer som vi ikke har lagt til som for eksempel muligheten til å sortere enkelt mellom disse typene med filtre. Vi har også vurdert å legge til andre drikkevarer, både med og uten alkohol.

Arbeidsprosess for datainnsamling og analyse

Den første form for datainnsamling vi bestemte oss for å utnytte var et intervju. Vi ønsket å intervjue noen innenfor Meteorologisk institutt for å kunne bedre forstå hvilke av dataene i API-ene som var relevante for å kunne lage en utepilsalgoritme som sier noe om det er bra nok vær til å ta seg en utepils.

Det ble først laget en intervjuguide med spørsmål om de ulike dataene fra hovedsakelig API-ene Nowcast og LocationForecast, men også åpen for bruk av andre API-er. Det ble også laget et samtykkeskjema som ble tilsendt intervjuobjektet. Vi fikk kontakt på mail med noen fra Meteorologisk institutt som ville være med på intervjuet, så vi satte opp et intervju 22. mars.

Ut i fra intervjuet fikk vi store mengder med nyttig informasjon for vårt prosjekt. Selv om intervjuobjektet ikke hadde veldig mye erfaring med tolkning av værdata spesielt, så fikk vi

noen innspill som hjalp oss med å få en bedre forståelse av værdataene. Vi fikk også flere gode forslag til andre API-er vi kunne bruke, funksjoner som kunne vært nyttige for appen vår, hvem som kanskje har mest nytte av appen og noen kontakter eller kilder som vi kunne utnytte om vi trengte mer data.

En annen form for datainnsamling vi brukte var loggføring av værdata kombinert med faktiske bilder av været. Dette ble loggført nesten hver dag på samme sted i løpet av store deler av prosjekttiden. Det at vi kunne se på dataene som blir skrevet ut samtidig som vi kan se bilde av været, gjorde at vi fikk en bedre forståelse av hvordan været ser ut ved ulike data og hva som faktisk er godkjent utepilsvær. Grunnet denne loggføringen fikk vi forbedret vår algoritme som bestemmer godkjent utepilsvær.

Mandag 14. Mars - veldig godkjent utepilsvær

Utepilsvær


```
"air_pressure_at_sea_level": 1027.7,
"air_temperature": 8.3,
"air_temperature_percentile_10": 7.4,
"air_temperature_percentile_90": 8.8,
"cloud_area_fraction": 0.0,
"cloud_area_fraction_high": 0.0,
"cloud_area_fraction_low": 0.0,
"cloud_area_fraction_medium": 0.0,
"dew_point_temperature": -4.7,
"fog_area_fraction": 0.0,
"relative_humidity": 44.6,
"ultraviolet_index_clear_sky": 1.9,
"wind_from_direction": 93.6,
"wind_speed": 2.5,
"wind_speed_of_gust": 5.5,
"wind_speed_percentile_10": 2.2,
"wind_speed_percentile_90": 2.6
```


Utklipp fra en av dagene vi loggførte med bilde og værdata

Med anbefaling fra gruppe-veilederne, bestemte vi oss også for å lage en brukerundersøkelse. Målet med brukerundersøkelsen var å få en bedre forståelse av brukerne og hvordan de ønsket at en slik app skulle blitt laget. Dermed lagde vi spørsmål som spurte etter brukernes preferanser på ulike vær som kan være utepils, hva slags funksjoner som de ville hatt med eller ikke hatt med i appen, hva som måtte til for at de skulle bruke en slik app og hvordan de helst ønsker at appen skal se ut. Det ble brukt en kombinasjon av likert-skala, åpne spørsmål, litt kortere og direkte spørsmål, rangeringer og bilde av appens foreløpige utseende.

Det ble først gjennomført en pilot-test av spørreskjemaet. Dette gjorde at vi fikk oppdaget skrivefeil, misforståelser ved spørsmål, gjentakelser eller lignende spørsmål som ga samme svar, og andre småfeil for å forbedre undersøkelsen. Deretter sendte vi den ut til flere av våre venner og bekjente som var interessert i å svare på denne undersøkelsen om utepils. Litt senere bestemte vi oss på et møte den 2. mai, for å legge til et spørsmål til hvor vi har to ulike bilder av forestilte utseende som ble modellert i Figma. Vi fikk deretter lagt ut undersøkelsen på Ifi sin facebook-gruppe som gjorde at vi fikk mange flere svar enn tidligere.

Vi fikk også tak i en bok som heter *Øl hele året* som er skrevet av Stian "Staysman" Thorbjørnsen og Jørn Idar Almås Kvig. Denne boken hadde mye informasjon om øl, blant annet øl-anbefalinger til ulike årstider. Dette tenkte vi at ville være til god nytte hvis vi fikk tid til å implementere en funksjon som anbefaler øl. Den hadde også anbefalinger til noen matretter og hvilke øl som passet til disse. Dette var også noe vi kunne utnyttet hvis vi ville ha med det, men fikk ikke tid til dette og vi hadde flere andre funksjoner som var viktigere en denne.

Vi hadde også flere idéer til datainnsamling og analyse som vi ikke fikk gjennomført eller som vi valgte bort. Et eksempel på det er en analyse. Vi tenkte at vi kunne ha en form for dagbok som en eller flere brukere svarer på flere ganger i uken. De kunne skrive om hvor bra de synes appen fungerer. Slik som om de synes været som faktisk er ute hos brukerne passer med det appen sier til om det er utepilsvær. Eller om de synes interaksjonen i appen fungerer godt eller om det er noen funksjoner de synes mangler når de bruker appen. Dette hadde gitt oss en god idé på hvor godt appen faktisk fungerer for brukerne våres. Vi valgte å droppe denne analysen ettersom vi fikk idéen ganske sent i prosjektet og vi hadde dermed veldig liten tid til å få tak i både brukere og få dem til å gjennomføre dagboken i flere dager. Vi planla isteden å ha en brukertest ettersom det tar kortere tid, krever mindre av brukeren og vi trenger ikke mange for dette, det hadde holdt med en bruker. Det ble til slutt for lite tid fordi appen vår var ikke ferdigstilt før siste dagen før fristen og vi hadde fullt fokus på rapporten. Et annet eksempel på datainnsamling vi ikke fikk gjennomført var å intervjue en øl-ekspert som vi ble anbefalt fra vårt første intervju. Et annet eksempel på datainnsamling vi ikke fikk gjennomført var å intervjue en øl-ekspert som vi ble anbefalt fra vårt første intervju. Etter å ha blitt anbefalt en øl-ekspert, tenkte vi at det ville være en god idé til øl-anbefaling funksjonen, om vi hadde hatt et intervju med en ekspert som kunne si masse om øl og hjelpe oss med å kunne lage en god anbefaler av øl. Siden vi fikk tak i en bok som var skrevet av en

ekspert som ga oss akkurat det vi var ute etter, så droppet vi denne formen for datainnsamling. I tillegg tenkte vi at det ville være mer krevende og usikkert, siden det ikke var sikkert denne eksperten hadde lyst eller tid til å være med på intervju.

Resultater fra datainnsamling og analyse

Intervju:

På møtet den 28. mars diskuterte vi noen av resultatene fra intervjuet. Blant annet ble vi anbefalt å bruke en "slider" som kunne la brukeren selv velge hva han/hun ønsker som laveste tillatte temperatur. Dette er ettersom folk har forskjellige ønsker og meninger om hva som er godkjent temperatur for en utepils og det kan være ulike situasjoner som tillater ulike temperaturer, for eksempel hyttetur på fjellet er kaldere enn midt i byen. Vi har ikke tid til å ta hensyn til alle disse, så dermed vurderte vi denne funksjonen som ganske nyttig for appen vår.

Personen vi intervjuet hadde mye fokus på nedbør som en viktig faktor i å bestemme utepilsvær. En ting som ble sagt var at om sommeren kan det noen steder oppstå lokale byger hvis det er vannkilder i nærheten. Dette er fordi vannet fordamper av den høye varmen og stiger oppover for så å kjøles ned i høyden og slippes ned igjen. Vi tenkte at dette var noe som kunne være nyttig å beregne sannsynligheten for, når man skal se på vær fremover i tid. Vi bestemte oss for å droppe denne funksjonen ettersom det vil være veldig krevende å finne ut hvor nærme vann man er og hvor høy temperatur det er, samtidig som det beregnes sannsynligheten for nedbør.

Vi fikk også en interessant potensiell brukergruppe av intervjuet. Nemlig at appen kan bli brukt av bransjer som sjekker været for bemanning. F.eks restauranter eller barer som har god nytte av å se om det er utepilsvær, fordi da vil det mest sannsynligvis komme flere kunder som krever flere på jobb.

Noe annet vi ble anbefalt var en podcast fra Meteorologisk institutt, hvor værmeldere snakket om litt forskjellige ting angående vær, men etter å ha hørt flesteparten av episodene, fant vi ut at det var lite relevant for vårt prosjekt.

Til slutt ble vi også anbefalt en person som heter Knut Albert som har gitt ut flere bøker om øl og som har et eget forum som vi eventuelt kunne utnytte for å stille flere øl-relaterte spørsmål senere i prosjektet vårt.

Brukerundersøkelse:

På stand-up-møte 22. april diskuterte vi første gang noen av funnene fra undersøkelsen som da bare hadde rundt 5 besvarelser. De fleste besvarelsene var ganske enig i svarene sine. De så for seg 15 grader som ønskelig temperatur, men kunne tillate lavere og helst lite skyer og ingen nedbør eller vind. Alle så for seg at appen skulle brukes til å finne når og hvor det er best med utepils. Ut ifra hvordan appen vår var ved "Minimal Viable Product", så var det liten interesse for å bruke appen for brukerne ettersom det blir litt for likt yr sine nettsider. Funksjoner som måtte til for å øke nyttigheten av appen var alle enig om at måtte være anbefalinger av øl eller hvor man kan ta seg en øl. De var også enige om at dette er en app som helst blir brukt om sommeren, men kan også bli brukt om våren og høsten, helst ikke om vinteren. Utseende til appen ønskes helst ryddig, profesjonelt og enkelt, men en bruker skrev spesifikt et ønske om et kart med ikoner for utesteder. En interessant observasjon vi gjorde var at en av brukerne absolutt ikke ønsket øl-anbefaling i motsetning til alle de andre som sterkt ønsket denne funksjonen. Vi tenkte først at vi kanskje hadde formulert spørsmålet dårlig og at brukeren misforstod, men vi skjønte senere at denne brukeren var en som var interessert i idéen/appen, men som ikke var glad i øl og heller ønsket andre drikkevarer.

Etter møtet 2. mai hadde vi langt flere svar på grunn av innlegget på IFI sin facebook-gruppe. Vi gjorde oss flere observasjoner av de nye svarene som overraskende nok reflekterte de første fem sine svar. Selv om fem svar var nå blitt til nærmere 50, så var det mye enighet blant svarene. Det var fortsatt foretrukket 15 grader i gjennomsnitt av svarene med lite til ingen vind og skyer uten nedbør, men det var nå mye større variasjon med alt fra -10 grader til opp i mot 25 grader. Den ideelle temperaturen var nå mellom 20-30 grader uten vind, skyer eller nedbør. De fleste ser for seg at en utepilsapp skal vise når, hvor og hva slags utepils man skal ta. Ved "Minimal Viable Product" så er det stor variasjon i hvor nyttig brukerne synes det er, men de er fortsatt enig i at det er for likt yr og trenger flere funksjoner for å være nyttig nok. Det var mange ulike funksjoner som ble anbefalt for å øke nytten av appen. Noen eksempler som vi ikke selv hadde tenkt på var for eksempel lykkehjul til å velge øl, noen form for chattefunksjon slik at man kan kontakte venner og dele øl-opplevelser med dem, wedge som kan vise det enkleste fra appen til forsiden av mobilen, pris på øl i nærheten eller

legge til spesielle øl for folk med behov/allergener som cøliaki eller lignende. Nå var det ikke lenger bare øl-anbefaler og utested-anbefaler, men mange andre ting også. Ut ifra rangeringen av funksjoner er det øl-anbefaling, utepilsvær fremover i tid og anbefaling av utesteder som er viktigst for brukerne fortsatt, men det er flere andre funksjoner som blir sterkt ønsket. Blant annet utepilsværet vises i prosent, mat og øl kombinasjoner og notifikasjoner, men det blir mye større variasjon enn tidligere. Brukerne ser for seg at de kommer til å bruke denne appen rundt sommertider og i helger til å planlegge noe sosialt og morsomt med øl og venner. Omtrent alle som svarte på undersøkelsen skriver at de ønsker at appens utseende skal være enkel ryddig og fin, så det er likt som de første fem personene ønsket. I det nye spørsmålet vi la til hvor det kan velges mellom et utseende uten kart eller et med kart, så velger de aller fleste uten kart. Dette skriver de er fordi de syntes kartet gjør utseende dårligere ved at det ikke lenger ser enkelt, ryddig og fint ut. Flere mener også at et kart er unyttig og fokuserer mer på funksjoner som ikke nødvendigvis har behov for et kart. På den andre siden, de som valgte med kart mener at det er en kjernefunksjon som er helt nødvendig for appens funksjoner som for eksempel å vise utesteder å dra til.

Vi endte med å velge uten kart ettersom flertallet fra undersøkelsen ønsket dette, men vi diskuterte også mulige fremtidige løsninger som vi ikke hadde tid til, men som vi kunne ha gjort om vi hadde mer tid. Det vi kom frem til var at vi kunne hatt en egen bar nederst på siden med ikoner som fører til egne sider. For eksempel et som fører til et kart om man vil bruke noen kart-relaterte funksjoner. Eller et tannhjul for å komme til en egen side hvor man kan gjøre endringer på hvordan man ønsker appen skal være, for eksempel dark/light mode eller laveste ønsket temperatur. Vi har mange idéer til hva som kan være med på denne baren om vi hadde hatt mer tid.

Annet:

Vi utnyttet også flere artikler fra yr.no sine nettsider som ga forklaringer på ulike værdata som var relevant for vårt prosjekt. Blant annet en skala som forklarte kjennetegn på ulike vind-nivåer og hvor sterk vinden er da. Denne brukte vi til å bestemme grensen på hvor mye vind som er tillatt for utepils. En annen ting fra yr som vi diskuterte var forklaring på hvordan de beregner “føles som” temperaturen. Vi fant også en formel som kunne hjelpe oss å regne ut dette. Det vil være en mer nøyaktig beskrivelse av temperaturen som faktisk er ute, så vi tenkte at dette kunne være en nyttig funksjon, men det er ikke veldig store forskjeller fra

faktiske og føles som, når vi allerede har satt lave grenser i vind osv. Dermed bestemte vi oss for at den ikke er like viktig som andre funksjoner.

Vindens virkning på land

Navn	Symbol	m/s	Knop	Kjennetegn
Stille	vindstille	0,0-0,2	0-1	Røyken stiger rett opp
Flau vind		0,3-1,5	1-3	En kan se vindretningen av røykens drift
Svak vind		1,6-3,3	4-6	En kan føle vinden. Bladene på trærne rører seg, vinden kan løfte små vimpler.
Lett bris		3,4-5,4	7-10	Løv og småkvister rører seg. Vinden strekker lette flagg og vimpler
Laber bris		5,5-7,9	11-16	Vinden løfter støv og løse papirer, rører på kvister og smågreine, strekker større flagg og vimpler
Frisk bris		8,0-10,7	17-21	Småtrær med løv begynner å svaie. På vann begynner småbølgene å toppe seg

Kilde: *(yr, 2022)*

Refleksjon

Vi fant ut gjennom prosjektet at kommunikasjon er viktig for å få til et godt teamarbeid mellom hverandre. Vi fant også ut at standup-møter er veldig fine å ha, men at det ikke var så lurt å ha dem hver dag, da dette førte til at ikke alle klarte å møte opp hver gang. Det var noen ganger en del personer som ikke møtte opp på møtene og da var ikke møtene veldig informative. Under møtene var vi flinke på å holde møtene korte og konkrete. Vi synes generelt sett at møtene er bra å ha med da det gjorde at alle var mer sikre på hvor vi lå ann under prosjektet, men at vi muligens kunne ha vært bedre på å møte opp alle sammen.

Under utviklingen så merket vi at det var noen vanskeligheter med å finne ut hva som var hoved-branchen, den branchen som inneholder den nyeste versjonen av appen. Fordi det var litt surr i hvilken branch den ene gruppen jobbet med og hva den andre gruppen tenkte var hoved-branchen. Hvis vi ser bort fra dette problemet så har vi hatt et veldig positivt forhold

med å jobbe i github. Vi tenkte at det ga oss ganske bra oversikt over hva som har skjedd med koden og hvilke implementasjoner det ene teamet gjorde før det andre.

Vi merket at oppgaven av å få tak i lokasjon og Google API kartet var veldig vanskelig å håndtere. Så vi tenker det hadde vært en god idé å prøve å fordele oppgaven litt mer slik at mer folk kunne hjelpe til og jobbe med det. Problemet var at oppgaven føltes veldig avansert ut for de som ikke drev på med koden fra starten av. Vi kunne også ha prøvd på at utviklerne hadde undervist litt om hvordan Google API fungerer, slik at man slipper mye lesing gjennom dokumentasjonen.

Vi lærte at det kunne ha vært lurt å ha en mer konkret plan på hvordan appen skal se ut til slutt, selv om dette er vanskelig. Dette er spesielt med tanke på at det tok såpass lang tid før vi fant ut at vi skulle droppe kartet. Hvis vi hadde skjønt at vi kunne droppe kartet tidligere, kunne vi heller brukt tiden på å få på plass søkefeltet i appen.

Vi tenkte ikke at vi ville jobbe med appen helt til siste dag, men på grunn av hvor vanskelig det var å implementere hovedfunksjonalitet så krevde det mer tid. Problemet var hvordan det er å jobbe sammen med andre programmører på et team. At hvis man er alene så blir man ferdig med en funksjon og fikser bugs før man går til neste, men hvis man jobber sammen så kan det bli mer kronglete ved at begge jobber samtidig på en kode eller at begge jobber på kode som skal være veldig tett sammen med hverandre.

Vi tenkte også at det hadde vært en god idé å forme funksjonskravene ved å spørre brukerne våre av hvilke funksjoner de vil ha, tidlig i prosjektet slik at vi kan da jobbe videre med disse kravene, istedenfor å bruke langt tid på å lage et krav og så til slutt finne ut at våre brukere synes dette kravet ikke er relevant for appen. På en måte så var det som skjedde med kartet der vi både fant veldig mange feil som tok langt tid å fikse og samtidig så spurte vi brukerne og fant ut at de ikke ville inkludere kart i det hele tatt.

Vi synes også at brukbarhetstesting funket veldig greit når vi fikk sett appen fysisk på en mobiltelefon og ikke simulert i android studio. Dette ga oss muligheten til å se hvordan en bruker hadde interagert med vår applikasjon og samtidig så fant vi ut nye bugs/features. Vi tenkte at det var en viktig prosess for å se hva vi burde gjort videre.

Alt i alt synes vi dette prosjektet har vært bra. Vi er fornøyd med app og rapport. I retrospekt så synes vi at vi kanskje kunne ha begynt med rapporten tidligere for å unngå skippertak på slutten. Men alt i alt så har det gått bra.

Referanser og kildeføring

AppBrain. (2022, 20. mai). *Top Android OS versions*. Hentet fra https://www.appbrain.com/stats/top-android-sdk-versions

Kvig, J. I. A. & Thorbjørnsen, S. (2020). *Øl hele året*. Oslo: Strawberry Publishing AS

Meteorologisk institutt (2022) *IN2000 caser 2022*. Hentet fra https://in2000.met.no/2022/

W3C (2018, 5. Juni) *Web Content Accessibility Guidelines (WCAG) 2.1*. Hentet fra https://www.w3.org/TR/WCAG21/#intro

yr (2022, 20. mai) *Vindpiler og Beaufortskalaen*. Hentet fra https://hjelp.yr.no/hc/no/articles/360002022134-Vindpiler-og-Beaufortskalaen

Vedlegg

Vedlegg 1: Intervjuguiden

Introduksjon:

Hei! Jeg går på Universitetet i Oslo og studerer informatikk. Akkurat nå jobber jeg og fem andre medstudenter med et prosjekt i samarbeid med dere/Meteorologisk institutt hvor vi skal lage en app hvor vi tar i bruk en eller flere av API-ene dere tilbyr. Vi har valgt et åpent case som vi har valgt å kalle utepils. Denne appen som vi ser for oss skal ta i bruk værdata-API-ene LocationForecast og nowcast for å kunne beregne hva som er fint nok vær til å ta seg en pils ute. I dette intervjuet har jeg et ønske om å få dine meninger og erfaringer om de ulike dataene og muligens noen tips om hvordan vi kan henvende disse i vårt prosjekt.

Innledning:

- Hvor lenge har du jobbet i Meteorologisk institutt?
- Hvordan har du jobbet med disse værdataene?

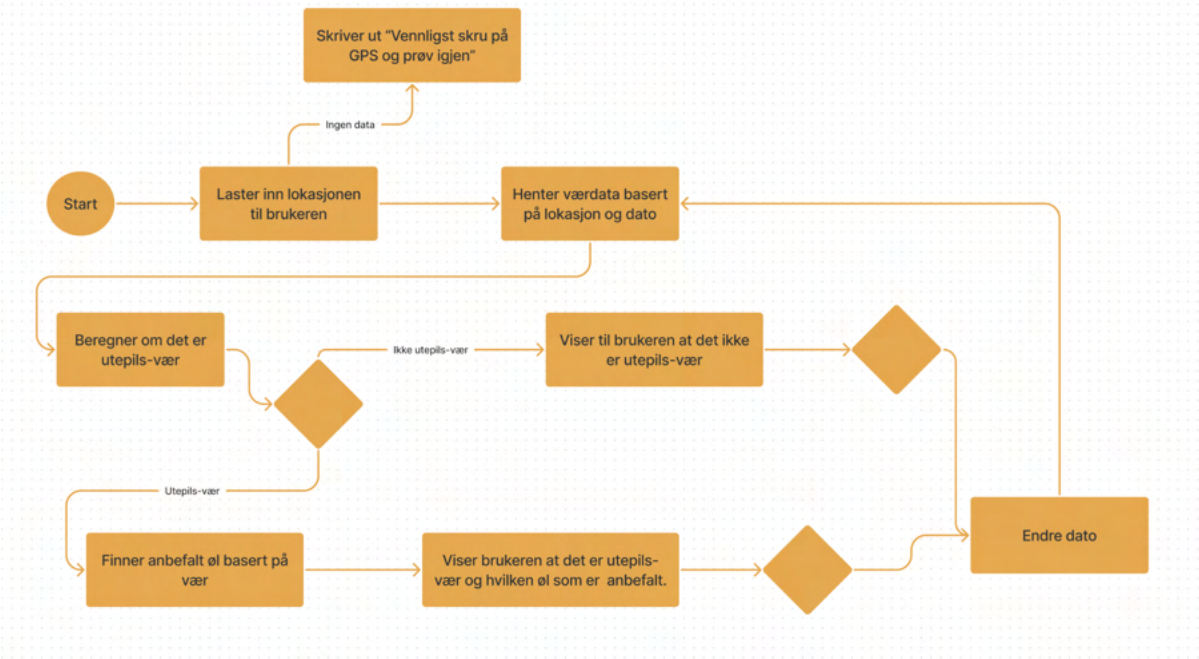
Hoveddel:

- Vi har sett i dataene deres i LocationForecast at det deles inn i 3 forskjellige skynivåer. Sånn vi har forstått det, så er høyere skyer mindre påvirkene for været og hvordan temperaturen føles, og lavere skyer mer påvirkene. Hvordan er forskjellene mellom disse skynivåene?
- Rundt hvilken prosentandel begynner de forskjellige skynivåene å påvirke været generelt og følelsen av temperaturen?
- Hvor mye vind i m/s ca. må det være før man merker det godt på følelsen av temperaturen?
- Hvor mye nedbør ca. i tall tilsvarer bittelitt småregn?
- Hvor varm må lufttemperaturen om sommeren være før bittelitt småregn blir ubetydelig?
- Hva vil du si er en relativt varm eller god temperatur uten at det er sol?
- Hvilken UV index ved klart skydekke vil du si er så varm at man kan kjenne det på huden?

Avslutning:

- Er det noen flere av dataene du tenker kan være nyttige å ta i bruk for å beregne et godt utepilsvær?
- Kommer du på noe innenfor dataene som vi burde tenke på når vi skal bestemme vær for utepils? (Evt. andre API-er?)
- Hva mener du er noen av de dårligste tillatte vær for å ta seg en utepils?
- Noe mer du vil legge til angående prosjektet vårt?

Vedlegg 2: Aktivitetsdiagram av funksjonalitetene i applikasjonen

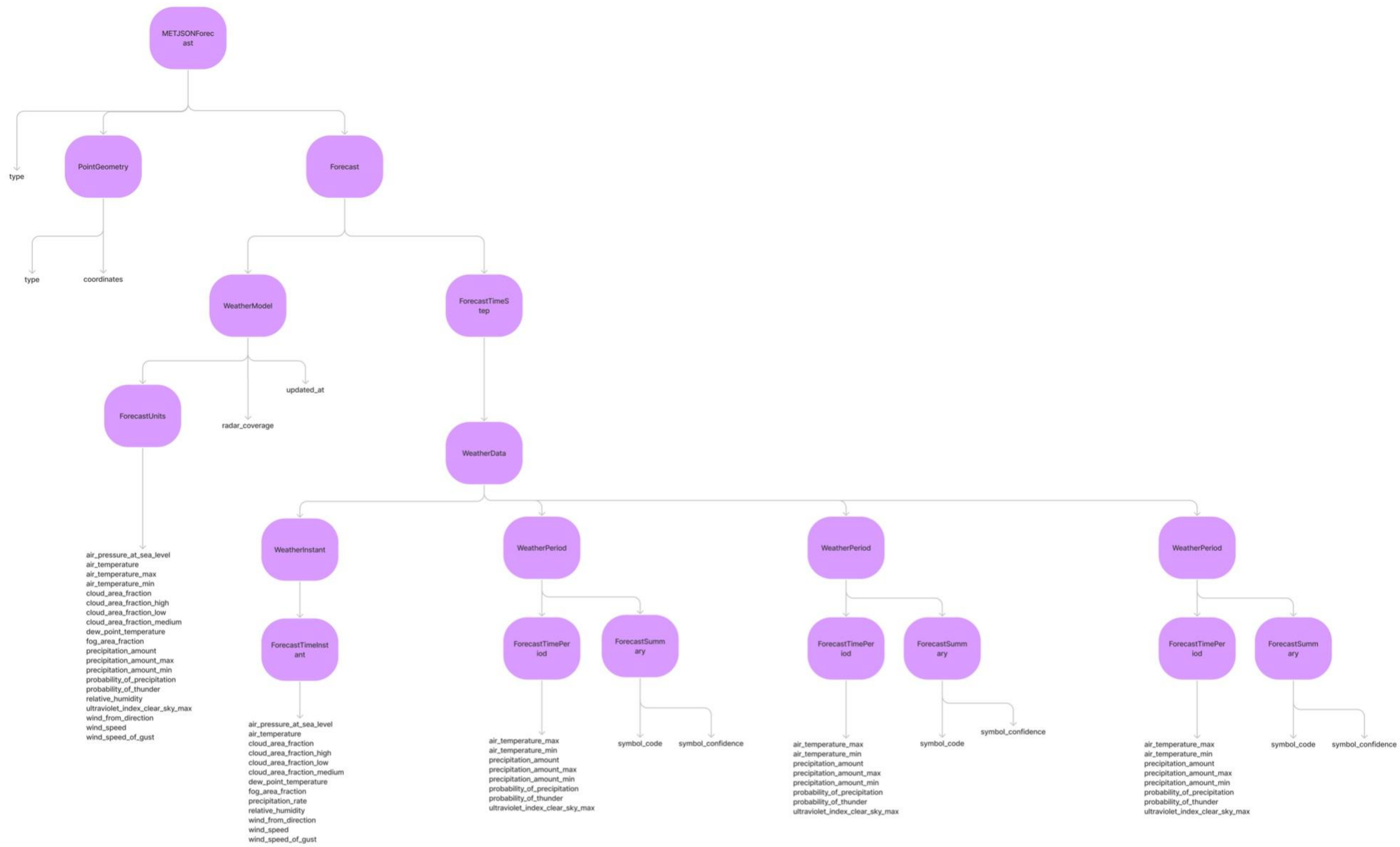


Vedlegg 3: Filer i prosjektet

- Algorithm.kt - Algorithm.kt er et companion-objekt til DataSource, som tar inn LocationForecast, Nowcast og Sunrise-objekter, og som bruker data fra disse klassene til å regne ut utepilsværet i prosent.
- Data.kt - En klasse som inneholder navnene til alle vær-symbolene, og hvilke bilder de tilhører. Disse symbolene (bildene). Disse bildene ligger under drawable
- DataSource.kt - Klassen der selve hentingen av MET-data skjer ved hjelp av metodene fetchForecast og fetchSunrise. Fuel og gson brukes til hentingen av data.
- HomeScreen.kt - ComponentActivity som inneholder UI til hjem-skjermen.
- HomeScreenViewModel.kt - Vår viewmodel som holder styr på de viktigste dataene i prosjektet (variabler som weathersymbol, isUtepils, locationPermitted, location, place og date). Det er i HomeScreenViewModel vi kaller på DataSource sine metoder for henting av API-data.
- LocationHandler.kt - Klasse for å requeste lokasjonstilgang fra brukeren. Har også funksjoner for å geocode fra koordinater (LatLng) til steder (adresser), og fra søkefelt-teksten til koordinater.

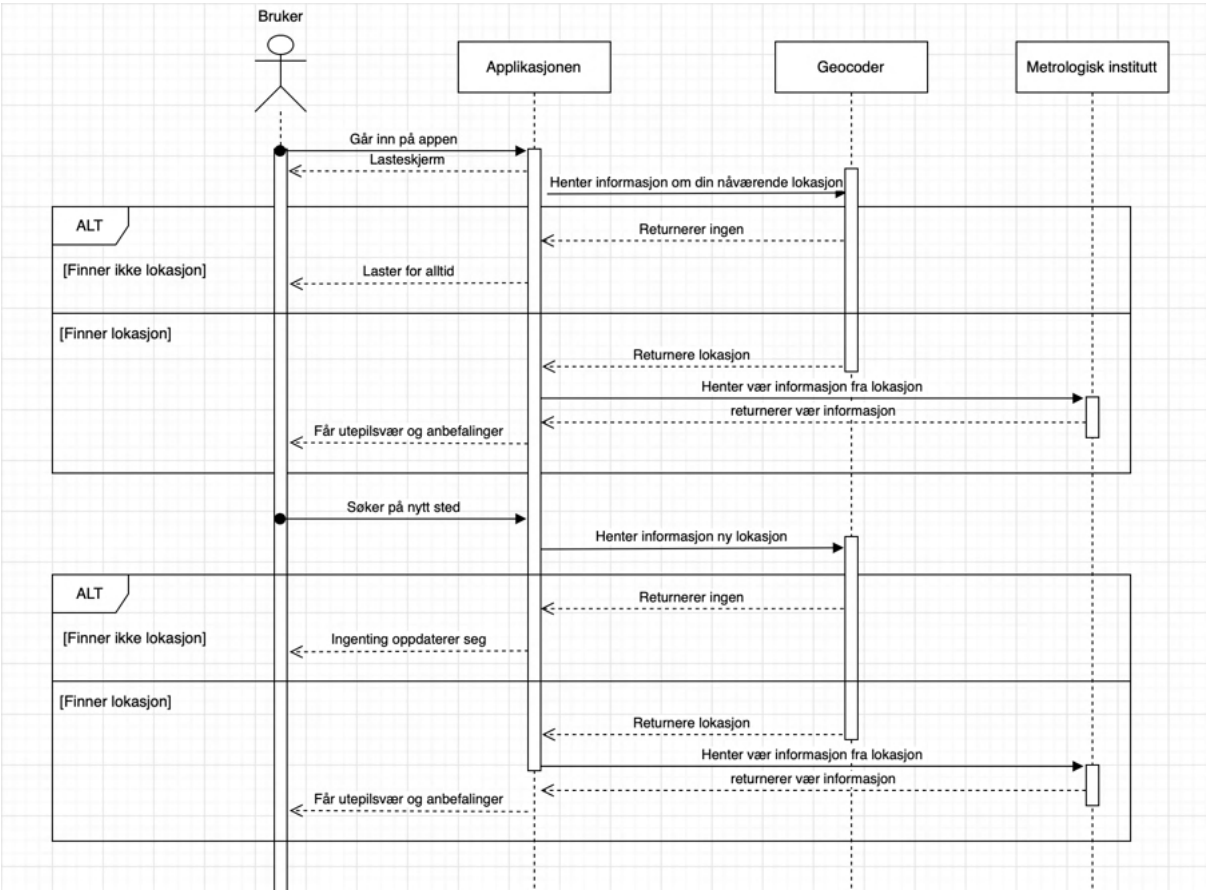
- METJSONForecast.kt - Dataklasser til henting av LocationForecast-API og Nowcast-API, i tillegg til funksjoner som getWeatherSymbol, getWeather, getCurrentPercipitationRate, getPercipitationAmount, og hasCloudData.
- METSunrise.kt - Dataklasser til henting av Sunrise-API, i tillegg til klassen METSunrise med funksjoner som getSunrise, getSunset og isDay
- PNG-filer som inneholder værsymbolene ligger i drawable
- I filen Theme.kt ligger data med “themet” til appen, som vi har lagd.

Vedlegg 4: Klassediagram over værinformasjon



METJSONForecast
type
PointGeometry
type
coordinates
Forecast
WeatherModel
ForecastUnits
air_pressure_at_sea_level
air_temperature
air_temperature_max
air_temperature_min
cloud_area_fraction
cloud_area_fraction_high
cloud_area_fraction_low
cloud_area_fraction_medium
dew_point_temperature
fog_area_fraction
precipitation_amount
precipitation_amount_max
precipitation_amount_min
probability_of_precipitation
probability_of_thunder
relative_humidity
ultraviolet_index_clear_sky_max
wind_from_direction
wind_speed
wind_speed_of_gust
radar_coverage
updated_at
ForecastTimeStep
WeatherData
WeatherInstant
ForecastTimeInstant
air_pressure_at_sea_level
air_temperature
cloud_area_fraction
cloud_area_fraction_high
cloud_area_fraction_low
cloud_area_fraction_medium
dew_point_temperature
fog_area_fraction
precipitation_rate
relative_humidity
wind_from_direction
wind_speed
wind_speed_of_gust
WeatherPeriod
ForecastTimePeriod
air_temperature_max
air_temperature_min
precipitation_amount
precipitation_amount_max
precipitation_amount_min
probability_of_precipitation
probability_of_thunder
ultraviolet_index_clear_sky_max
ForecastSummary
symbol_code
symbol_confidence
WeatherPeriod
ForecastTimePeriod
air_temperature_max
air_temperature_min
precipitation_amount
precipitation_amount_max
precipitation_amount_min
probability_of_precipitation
probability_of_thunder
ultraviolet_index_clear_sky_max
ForecastSummary
symbol_code
symbol_confidence
WeatherPeriod
ForecastTimePeriod
air_temperature_max
air_temperature_min
precipitation_amount
precipitation_amount_max
precipitation_amount_min
probability_of_precipitation
probability_of_thunder
ultraviolet_index_clear_sky_max
ForecastSummary
symbol_code
symbol_confidence

Vedlegg 5: Sekvensdiagram av versjonen med søkefelt



Vedlegg 6: Bilde av hvordan appen kunne ha sett ut om vi hadde fått med søkefelt