Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Sabine Seljeseth
Sivert Utne

Wasteflow

Mobilapplikasjon for avfallshenting

Bacheloroppgave i Dataingeniør
Veileder: Majid Rouhani

Mai 2021

NTNU
Kunnskap for en bedre verden

Sabine Seljeseth
Sivert Utne

Wasteflow

Mobilapplikasjon for avfallshenting

Bacheloroppgave i Dataingeniør
Veileder: Majid Rouhani
Mai 2021

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

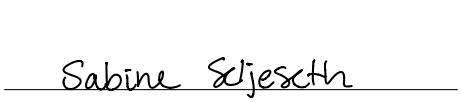
Forord

Denne rapporten er en del av bacheloroppgaven for studieprogrammet Dataingeniør, ved Institutt for Datateknologi og Informatikk, ved NTNU. I tillegg til rapporten er det utviklet en mobilapplikasjon, Wasteflow, som er en bestillingstjeneste for avfallshenting. Denne rapporten tar for seg hvordan utviklingen av applikasjonen ble gjennomført, og hvordan applikasjonen ble etter endt utvikling. Oppgaven er utformet av Favn Software AS. Vi valgte denne oppgaven fordi vi hadde et ønske om å tilegne oss mer erfaring innenfor applikasjonsutvikling, og spesielt innenfor utvikling av applikasjoner for mobilenheter.

I løpet av bacheloroppgaven har vi i teamet tilegnet oss nye kunnskaper og erfaringer vi ikke hadde fått fra andre oppgaver. Mye av teknologien som er brukt var ukjent for oss før vi begynte, dette innebar en bratt læringskurve, men ved avslutning er dette teknologier vi føler oss komfortable med. Vi har også erfart hvordan det er å jobbe for en oppdragsgiver, noe som har vært både morsomt og lærerikt. Dette er erfaringer vi kommer til å ta med oss videre.

Prosjektet har vært preget av koronapandemien. Hele oppgaven ble gjennomført ved hjemmekontor, noe som gjorde at teamet dessverre ikke hadde muligheten til å møte oppdragsgiver og veileder personlig. Kommunikasjonen ble gjennomført med digitale møter gjennom Microsoft Teams.

Teamet ønsker å takke de ansatte fra Favn Software AS som var involvert i prosjektet, Sveinung Øverland og Anders Hallem Iversen, for å ha vært tilgjenglig for spørsmål til tross for omstendighetene. Vi ønsker også å takke vår veileder, Majid Rouhani, for hjelpen og tilbakemeldingene gjennom hele prosjektet. Vi vil også rekke en takk til de som tok seg tid til å teste applikasjonen, både under brukertester og underveis, dette har hjulpet oss med å oppdage feil og finne muligheter og problemer vi selv ikke hadde funnet. Helt til slutt vil vi takke venner og familie for innspill og støtte gjennom prosessen.



Sabine Seljeseth

Sivert Utne

Trondheim

20. mai 2021

Oppgavetekst

Hensikten med oppgaven

Hensikten med oppgaven er å lage en mobilapplikasjon, Wasteflow, der brukere skal kunne bestille henting av avfall. Mobilapplikasjonen skal brukes i et pilotprosjekt med avfallsselskapet Remiks og Studentskipnaden i Tromsø.

Oppgavebeskrivelse

Den opprinnelige beskrivelsen av oppgaven var å utvikle en prototype av en mobilapplikasjon, der brukere skulle bestille og betale for henting av avfall. Applikasjonen skulle fungere på både iOS og Android ved bruk av rammeverket React Native. Brukeren skulle legge inn informasjon om, og bilder av, avfallet som var lagt inn i bestillingen. Videre skulle applikasjonen integreres med Stripe, slik at bestillingene kunne betales med mobilen. Oppgaven inkluderte også utvikling av en tjener og database, der bestillingene skulle bli behandlet og lagret.

Underveis endret rammene for oppgaven seg. En prototype ble utvidet til en mobilapplikasjon, siden det vi skulle utvikle ble endret til en fungerende mobilapplikasjon. Muligheten til å legge til bilder av avfallet som skulle hentes ble nedprioritert. Oppdragsgiver ønsket at Google sin utviklingsplatform Firebase skulle anvendes. Dette betydde bruk av Cloud Firestore som database, Firebase Cloud Functions og Firebase Authentication. Rammeverket ble endret fra React Native til Flutter. For autentisering og oppretting av brukere skulle mobilnummeret til brukeren brukes istedenfor den mer tradisjonelle løsningen med e-mail. Hensikten med applikasjonen er å danne grunnlaget for en del av et sammensatt system for henting av avfall.

Den nye beskrivelsen av oppgaven blir dermed å utvikle en mobilapplikasjon som inngår i et større pilotprosjekt i samarbeid med avfallsselskapet Remiks i Tromsø. Målet med pilotprosjektet er å gjøre det mulig å bestille henting av avfall (som for eksempel møbler og hvitevarer) med applikasjonen. Applikasjonen blir et alternativ til å kjøre avfallet selv. Applikasjonen skal utvikles ved hjelp av Flutter og Firebase. Brukeren skal kunne registrere seg og logge inn med mobilnnummer og opprette bestillinger knyttet til den innloggende brukeren.

Sammendrag

I Norge er det en økende trend i mengden husholdningsavfall som produseres, og en synkende trend i avfallsmengden som blir gjenvunnet. Kommunene i landet har ansvaret for innsamling og behandling av husholdningsavfall. Mange kommuner krever at innbyggerne selv må frakte avfallet til gjenvinningsstasjonen. Dette fører til lange køer og venting for de som ønsker å gjenvinne.

I denne bacheloroppgaven ser vi på en løsning for å forenkle leveringsprosessen av avfall. Dette gjøres ved å gi innbyggerne et nytt og digitalt alternativ, som gjør det mulig å gjenvinne avfall uten behovet for å kjøre det selv til gjenvinningsstasjonen. Denne løsningen utforskes i følgende problemstilling:

Kan en digitalisert løsning for avfallslevering, i form av en mobilapplikasjon, gjøre gjenvinning mer tilgjengelig for befolkningen.

For å svare på problemstillingen er det utviklet en mobilapplikasjon som kan anvendes på iOS og Android mobiltelefoner. Med denne mobilapplikasjonen kan brukerne legge inn bestillinger av vanlige typer husholdningsavfall som for eksempel møbler, hvitevarer, elektronikk mm. Bestillingen vil så bli plukket opp av en hentebil som frakter avfallet til gjenvinningsstasjonen.

Brukertestene gjort på applikasjonen viser at dette er en løsning som er aktuell for brukere uten mulighet til og selv levere avfall til gjenvinningstasjonen. Testbrukerne ser behovet for applikasjonen, og sier de hadde anvendt løsningen fremfor å bruke tid på å levere avfallet selv. Applikasjonen er oversiktlig og lett å navigere, vi mener derfor tjenesten har potensial til å gjøre gjenvinning mer tilgjengelig.

Abstract

In Norway there is an increasing trend in the amount of household waste that is produced, and a decreasing trend in the amount of waste that is recycled. The municipalities are responsible for the handling and recycling of this waste. Some municipalities require its inhabitants to transport the waste to the recycling station themselves. This results in long waiting times for those who wish to recycle.

In this bachelor thesis we will look at a solution to this issue by simplifying the process of transporting the waste. This is achieved by giving the inhabitants a new digital alternative that makes it possible to recycle without the need to deliver the waste themselves. This solution is explored in the following issue:

Can a digital solution for waste delivery, in the form of a mobile application, make recycling more available to the public.

To answer this issue we have developed a mobile application that is available on both iOS and Android phones. With this application, the users are able to order pickup of household waste, such as furniture, appliances, electronics etc. The order will then be collected by a truck that delivers the waste to the recycling station.

User tests that were performed during this bachelor, shows that this is a viable solution for users without the ability to deliver the waste to the recycling station themselves. Most of tthe test users said they would utelize the application to prevent spending their time delivering the waste themselves. The application was described as easy to navigate with clear functionalities. It is therfore our opinion that this application has the potential to make recycling more available to the public.

Forkortelser og definisjoner

Forkortelse	Betydning	Beskrivelse
MVP	Minimum Viable Product	Produkt som kun har nøkkelfunksjonaliteten
OTP	One Time Password	Engangspassord brukt under autentiseringsprosessen
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart	En type omvendt turingtest for å finne ut om brukeren er en datamaskin eller menneske.
VSCode	Visual Studio Code	Integrert utviklingsmiljø for kode
FCF	Firebase Cloud Functions	Verktøy for kjøring av skyfunksjoner
MVC	Model View Controller	Designmønster innenfor systemutvikling
GDPR	General Data Protection Regulation	Forordning som skal styrke og harmonisere personvernet ved behandling av personopplysninger i Den europeiske union (EU)
SDK	Software development kit	Et programvareutviklingssett er en samling programvareutviklingsverktøy i en installerbar pakke

Begrep	Betydning
Veileder	Majid Rouhani
Teamet/vi	Sabine Seljeseth og Sivert Utne
Oppdragsgiver	Favn Software As
Brukere	Alle som bruker tjenesten Wasteflow

Innhold

Figurer

Tabeller

1 Introduksjon og relevans

1.1 Bakgrunn

Norge er et av landene i Europa der innbyggerne kaster mest avfall. Den gjennomsnittlige nordmannen kaster 776 kilo husholdningsavfall. Av de 12,2 millionene tonn med avfall som nordmenn produserte i 2019, ble 71% av de gjenvunnet. Dette er en nedgang på 16% siden 2011 da tallet var oppe på 87% (Miljødirektoratet, 2021).

I Norge er det kommunene som har ansvaret for å samle inn og behandle husholdningsavfall (FHI, 2018). Hvordan dette er løst varierer fra kommune til kommune, men en stor del av dem har en gjenvinningsstasjon. I de kommunene det mangler en henteløsning for avfall som skal til gjenvinningsstasjoner, er det innbyggernes eget ansvar å få fraktet avfallet til stasjonen. I disse kommunene kan det oppstå stor pågang hos gjenvinningsstasjonene, særlig i helgene. Dette har ført til så mye trafikk og kø hos noen gjenvinningsstasjoner, at politiet har måttet blitt involvert for å dirigere køen (Reppen Kvikstad, 2020). En økning i kø og ventetid kan føre til et frafall av folk som gjenvinner.

Av de 71% som ble gjenvunnet i Norge i 2019 var det bare 41% av avfallet som ble materialgjenvunnet. Materialgjenvinning er da ressursene i avfallet blir brukt til råvarer i produksjon av nye produkter eller materialer (Miljødirektoratet, 2021). Norge har som mål å øke andelen husholdningsavfall som blir materialgjenvunnet til 55% innen 2025, det må altså være en økning på 14% de neste fire årene for å oppnå målet (Miljødirektoratet, 2019). Til tross for dette ser vi en nedgang i andelen avfall som går til gjenvinning (Slåen Sæther, 2020).

Bacheloren er en del av et større pilotprosjekt med avfallsselskapet Remiks og Studentskipnaden i Tromsø. Prosjektets hensikt er å minke køene Remiks opplever i helgene, ved hjelp av et system der privatpersoner kan bestille avfallshenting hjemmefra, deretter blir avfallet hentet og fraktet til Remiks. Denne bacheloren vil omhandle utviklingen av mobilapplikasjonen for å bestille avfallshentingen. Andre bacheloroppgaver som inngår i samme pilotprosjekt tar for seg *Administrasjon- og analyseverktøy for innhenting av avfall* og *Strategi- og koordineringsverktøy.*

I tillegg til å minke køer har prosjektet som mål å gjøre terskelen for gjenvinning lavere. En lavere terskel kan føre til en økning i andelen husholdningsavfall som blir gjenvunnet. Dette vil hjelpe Norge på veien i å nå målet sitt om å øke andelen husholdningsavfall som blir materialgjenvunnet.

1.2 Problemstilling

1.2.1 Problemet

Som nevnt i problemsammendraget i visjonsdokumentet (vedlegg A) har høy pågang hos gjenvinningstasjonen preget brukeropplevelsen. Denne høye pågangen har skapt lange køer, som har blitt så voldsomme at politet har vært nødt til bli involvert. Dette kan skape en uønsket høy terskel for gjenvinning i kommunen. For å prøve å unngå frafall i gjenvinning skal vi i denne oppgaven se på en alternativ løsning. Løsningen går ut på å gi innbyggerne muligheten til å bestille avfallshenting, istedenfor å kjøre avfallet til gjenvinningstasjonen selv. Håpet er at dette vil lette på trykket gjenvinningstasjonene opplever, senke terskelen for gjenvinning og gjøre prosessen med gjenvinning av avfall så enkel og behagelig som mulig.

For å få til dette har teamet utviklet en mobilapplikasjon som kan lastes ned på både iOS og Android mobiltelefoner. I denne applikasjonen kan brukerne bestille henting av avfall. Dette avfallet vil deretter bli hentet og fraktet til gjenvinningsanlegget.

1.2.2 Formulering

Problemstillingen som skal tas stilling til i denne rapporten er:

> **Kan en digitalisert løsning for avfallslevering, i form av en mobilapplikasjon, gjøre gjenvinning mer tilgjengelig for befolkningen.**

1.3 Avgrensninger

Wasteflow er et sammensatt system med dedikert mobilapplikasjon til iOS og Android. Systemet består også av en nettside, og bruker et analyse- og administrasjonsverktøy med optimalisert ruteplanlegging for å effektivisere avfallshåndteringen. Denne oppgaven tar for kun seg mobilapplikasjonen. I prosjektet har teamet utviklet mobilapplikasjonen for bestilling av avfallet. De resterende elementene i systemet er fordelt på to andre bacheloroppgaver.

1.4 Relevant arbeid

Noen kommuner har allerede en digital plattform i form av en nettside som tilbyr avfallshenting. Disse er beskrevet under.

1.4.1 hentavfall.no

hentavfall.no er et kommunalt samarbeid mellom kommunene Stavanger, Sandnes, Sola, Randaberg, Time og Gjesdal og Renovasjonen IKS. De tilbyr henting av grovavfall, hageavfall, farlig avfall, glassemballasje og tøy, men dette varierer avhengig av hvilken kommune brukeren befinner seg i. De tilbyr også samlede bestillinger for borettslag (hentavfall.no, u.d.).

1.4.2 RagnSells

Dette er en avfallshentingstjeneste for beboere i Nedre Romerike og Oslo. De tilbyr henting av elektronisk avfall (hvitevarer o.l), blandet avfall, dekk, møbler, gjenstander av metall, papp, plast o.l og hageavfall. Bestillingen gjøres på nettsiden via "Grønn bil", der brukeren bestiller en hel- eller en halvfull varebil (RagnSells, u.d.) som kommer og henter avfallet.

1.4.3 SøppelGutta

I likhet med Regnsells.no er dette en tjeneste som er tilgjengelig for beboere i Oslo, men de operere også i Viken. De tilbyr henting og kjøring til gjenvinningsstasjon der du bestiller en bil av forskjellig størrelse. De tilbyr frakting av avfall som inventar, hageavfall, tekstil og annet søppel (SøppelGutta, u.d.).

1.4.4 soppelsekk.no

"soppelsekk.no" tilbyr avfallshenting i og rundt Trondheim. De henter alt avfall utenom elektrisk avfall, matavfall og miljøfarlig avfall. Dette foregår ved å kjøpe spesielle sekker hos diverse forhandlere rundt omkring i Trondheim, for deretter å fylle sekken med avfallet og bestille henting (soppelsekk.no, u.d.).

1.5 Struktur

Under er det gitt et kort sammendrag om av hva de neste kapittelene vil omhandle.

Kapittel 2 - Teori

Inneholder relevante temaer som omhandler problemstillingen, problemdomene og løsningen som skal utvikles.

Kapittel 3 - Valg av teknologi og metode

Her beskrives de spesifikke verktøyene som er tatt i bruk for å realisere noen av temaene som er nevnt i teorikapittelet.

Kapittel 4 - Resultater

Applikasjonen og oppnåelse av de målene som ble satt under planlegginsfasen av prosjektet, altså resultatene av utviklingsprosessen som er gjort, brukertestene som er utført og måloppnåelse av det administrative, er alt beskrevet i dette kapittelet.

Kapittel 5 - Diskusjon

I denne delen av rapporten legger teamet frem årsakene til at resultatene ble slik de ble. Dette inkludere beslutninger som ble tatt og hvorfor elementer ikke ble fullført. Delen inneholder også de etiske konsekvensene som kan følge av denne oppgaven.

Kapittel 6 - Konklusjon og videre arbeid

I dette kapittelet vil problemstillingen som ble stilt i starten av rapporten undersøkes nærmere. Teamet vil se hvilke konklusjoner som kan trekkes i forhold til problemstillingen ut ifra de observasjonene og resultatene som er oppnådd i løpet av prosjektet. Alt som ikke ble gjennomført, og teamet sine forslag til fremtidige utvidelser av applikasjonen beskrives i delkapittelet "videre arbeid".

2 Teori

I dette kapittelet vil vi gjennomgå relevant teori og metoder som er brukt under utviklingen av applikasjonen. Hensikten med kapittelet er å gi et bedre innblikk i problemsstillingen som skal utforskes, og danne en felles forståelse av løsning og metode. Først vil vi se på betydningen av digitalisering og avfall siden dette er sentrale deler av problemstillingen. Deretter vil vi gå gjennom hvordan Norge tar stilling til den økende avfallsmengden. Videre ser vi på autentisering og hvorfor dette er viktig under behandling av personopplysninger. Til slutt vil vi gå gjennom temaer innenfor systemutvikling, i tillegg til spesifikke temaer innenfor mobilapplikasjoner.

2.1 Digitalisering

Når begrepet "digitalisering" dukker opp, er det viktig å skille dette fra "digitisering". Digitalisering og digitisering er begreper som har ulike definisjoner, men som ofte brukes om hverandre.

2.1.1 Hva er Digitisering?

Digitisering er den tekniske prosessen der noe blir konvertert fra et analogt format til et digitalt format (Brennen & Kreiss, 2016). Et eksempel på dette er om noen tar et fysiskt polaroidbilde og skanner dette slik at man får det på en digital platform, som for eksempel en datamaskin. Da har man gjort det analoge bildet til et digitalt bilde, bildet har da blitt digitisert.

2.1.2 Hva er Digitalisering?

Digitalisering er et videre begrep uten en entydig definisjon. Dette begrepet er blitt popularisert i takt med internettet. Den økte bruken av internett har endret hvordan hverdagen ser ut, og hvilke behov vi har. Store deler av samfunnet bruker digitale verktøy for å kunne være en del av utviklingen som skjer i verden. Vi sier at de digitaliserer virksomheten sin, men hva betyr egentlig dette?

Noen ser på digitalisering i takt med det sosiale livet vi lever, og hvordan det går mer over til en digital plattform. Scott Brennen og Daniel Kreiss, som begge er ansatt hos universitetet av North Carolina, beskriver digitalisering på følgende måte: "Vi refererer til digitalisering som måten mange domener i det sosiale livet er omstrukturert rundt digital kommunikasjon og medieinfrastrukturer"(Brennen & Kreiss, 2016).

Om vi ser på regjeringen sin definisjon, er det mer fokus på forbedring av tjenester og effektivisering. Definisjonen lyder som følger: "Digitalisering handler om å bruke teknologi til å fornye, forenkle og forbedre. Det handler om å tilby nye og bedre tjenester, som er enkle å bruke, effektive og pålitelig" (Regjeringen.no, 2014).

I denne rapporten ser vi på digitalisering i sammenheng med definisjonen fra regjeringen.

2.1.3 Fordeler med digitalisering

Ved å gjøre områder av bedrifter digitale, får bedrifter flere muligheter for kundeinteraksjon. Når interaksjoner er digitalisert kan man hente og analysere data fra kunden, disse analysene kan så brukes til forbedre kundens opplevelse av bedriften og tjenester den tilbyr (Rosin, Proksch, Stubner & Pinkwart, 2020).

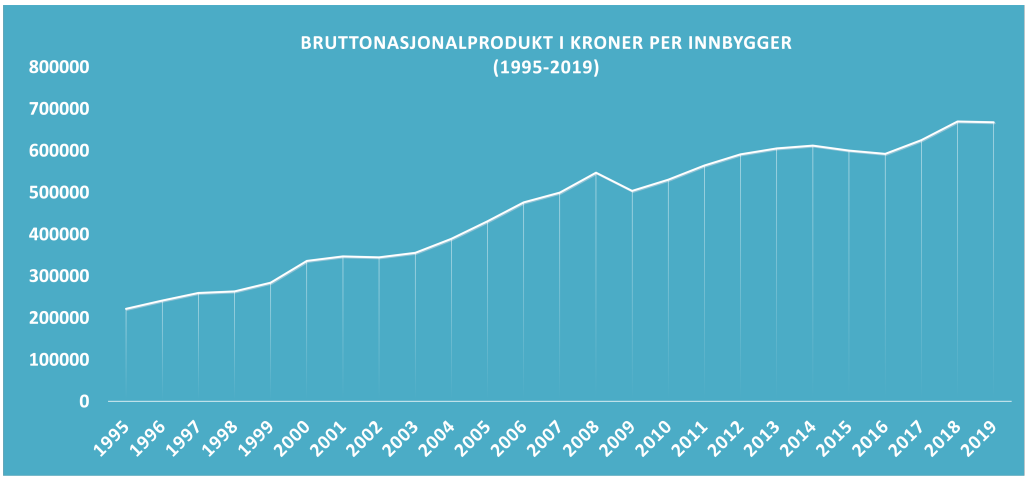
En digitalisering vil også være med på å hjelpe bedrifter med å identifisere risikofaktorer, siden en digital løsning gir bedriften enkel tilgang til dataene i sanntid, noe som gir bedriftene en mulighet til å reagere før problemer oppstår. Dette kan igjen gjøre opplevelsen for kundene bedre, og kan forhindre større utgifter for bedriften (Rosin et al., 2020).

I denne oppgaven ser vi på hvordan digitalisering av avfallshenting kan forhindre problemer og forenkle prosessen for brukerne.

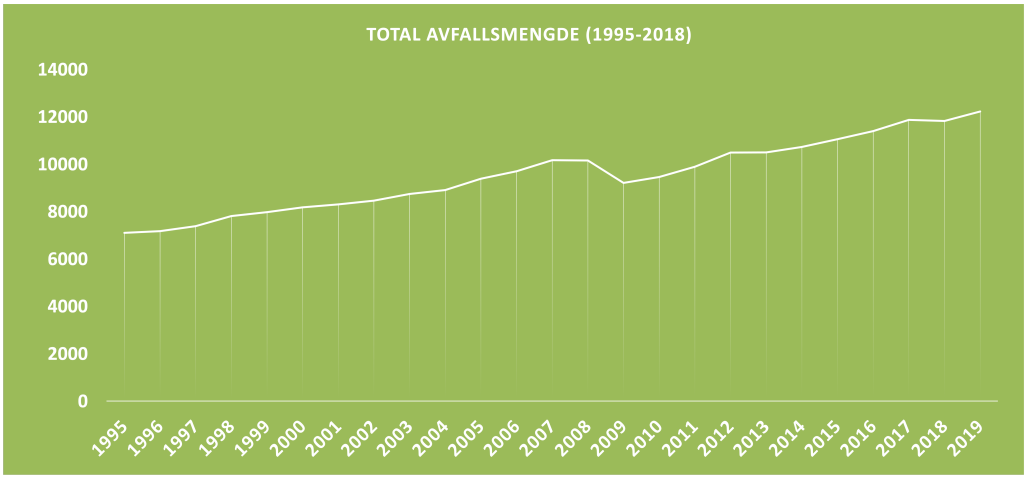
2.2 Avfall

Avfall er en sentral del av problemdomenet til denne oppgaven, derfor er det viktig å vite de sentrale argumentene for avfallsgjenvinning. I dette delkapittelet vil vi gjennomgå hva avfall er, hvorfor og hvordan Norge kan snu trenden av økende mengde avfall og hvordan avfallsbehandling foregår i Norge.

Det store norske leksikon definerer avfall som en samlebetegnelse på kasserte gjenstander og restprodukter som ikke lenger har samme verdi, men fortsatt representerer viktige ressurser ved gjenvinning (for Kildesortering og Gjenvinning., 2018a). Avfall produseres av husholdninger og bedrifter. Økonomien til Norge kobles tett sammen med den økende mengden avfall.

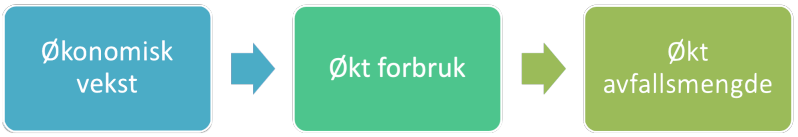


Figur 1: Bruttonasjonalprodukt i Norge fra 1995-2019 (SSB, 2021b).



Figur 2: Total avfallsmengde i Norge fra 1995-2019 (SSB, 2021a)

I figur 1, ser vi at bruttonasjonalproduktet i Norge har hatt en stigende trend. Fra figur 2, ser vi at i den samme perioden har også den totale avfallsmengden økt. Økonomisk vekst bidrar til en større mengde avfall (Sortere.no, 2021). Dette skjer på grunn av et økt forbruk (se figur 3) og gjelder spesielt elektronisk avfall.



Figur 3: Sammenhengen mellom økt økonomi og økt avfall

For å prøve å senke den økende mengden avfall har Norge satt seg flere mål. Dette er å produsere mindre avfall, samtidig som utnyttelsen av ressursene i avfallet må øke. Avfall kan være en nyttig ressurs om det kan gjenvinnes uten store kostnader. For at målene skal kunne nås, er Norge nødt til å omstille økonomien til en sirkulær økonomi.

2.2.1 Sirkulær Økonomi

En del av Norge sitt mål vil være at produkter må vare så lenge som mulig. I en sirkulær økonomi setter man fokus på at produkter skal ha så lang levetid som mulig, altså at vi reparerer, oppgraderer og gjenbruker. Først når dette ikke er mulig skal avfallet materialgjenvinnes og brukes som råvarer i ny produksjon. Dette er i motsetning til den linære økonomien i et bruk og kast samfunn hvor det utvinnes, produseres, konsumeres og kastes. For å oppnå en sirkulær økonomi er man avhengig av at forbrukerne kan velge miljøriktig. Digitalisering har en sentral rolle i å informere, hjelpe og gjøre det mulig for forbrukerne å gjøre dette (Miljødirektoratet, 2020).

En overgang til sirkulær økonomi vil føre til mindre klimautslipp, en oppbremsing i tapt naturmangfold, redusere forurensning og skape flere grønne arbeidsplasser. En overgang vil være en nødvendig del av omstillingen til et lavutslippssamfunn og for å kunne nå FN sitt bærekraftsmål. Om dette ikke skjer vil vi ha en ressursknapp fremtid i takt med befolkningsveksten og det økende forbruket. (Sortere.no, 2021)

2.2.2 Konsekvenser av avfall

Hvor store miljøkonsekvenser avfall fører til, avhenger av ulike faktorer. Disse faktorene er blant annet:

- Hvor mye avfall som blir produsert
- Sammensetningen til avfallet, altså hva det består av og inneholder
- Hvor mye avfall som går til sluttbehandling.

Miljøforurensning forårsaket av avfall skyldes spredning av miljøgifter og klimagasser fra deponi, dårlig avfallshåndtering eller gjennom forsøpling. Forsøpling er et problem siden lite nedbrytbare materialer kan bli værende i miljøet i flere hundre år, noe som vil skade levende organismer. I tillegg kan de ulike delene i avfallet være helseskadelig ved at de kan ha uheldige kjemiske egenskaper, avgi stråling eller føre til spredning av bakterier og virus (FHI, 2018). Helseproblemer og effekter forårsaket av avfall varierer fra de ulike typene avfall. Eksempler på dette er at behandling av husholdningsavfall kan føre til at befolkningen kan bli plaget med lukt og forbrenning av avfall produserer røyk og kan forurense store områder.

Som tidligere nevnt kan avfall utnyttes som en ressurs. Å sørge for at avfallet blir håndtert korrekt er en viktig del av å redusere skadevirkningene. Om man legger til rette for det, kan avfallet materialgjenvinnes eller brukes som energikilde. En sentral rolle i dette er at det er gode systemer for kildesortering av husholdningsavfall.

2.2.3 Avfallshierarkiet

For å enkelt illustrere Norges prioriteringer i avfallspolitikken og EUs rammedirektiv for avfall bruker vi avfallspyramiden. Denne er delt opp i flere lag. Målet er at avfallet skal behandles så nært toppen av pyramiden som mulig.

Figur 4: Avfallspyramiden (Sortere.no, 2021).

Avfallsreduksjon

Dette dreier seg om å redusere avfallsmengdene ved hjelp av å kjøpe og kaste mindre. Målet er å sikre at veksten i avfallsmengden er lavere enn den økonomiske veksten. Under dette punktet havner også forebygging, som innebærer forebyggende tiltak man iverksetter før et produkt er blitt til avfall. Et eksempel på dette er at mengden emballasje minkes der det er mulig, noe som fører til en redusert mengde emballasje som ender opp som avfall.

Gjenbruk

Handler om å bruke ting på nytt fremfor å kaste det bort. Noen av gjenvinningstasjonene i landet som tar imot avfall har f.eks. bruktbutikker som selger gjenstander som blir levert, men som fortsatt er i god stand.

Materialgjenvinning

Materialgjenvinning, ofte fortkortet til gjenvinning eller resirkulering, er å bruke de ulike matrialene fra avfallet som råvarer i nye produkter. Mye av avfallet som kastes er viktige ressurser som trengs i produksjonen av nye produkter. Dette gjelder spesielt elektrisk og elektronisk avfall som inneholder miljøfarlige og dyrebare materialer som f.eks. gull. Ved kildesortering sørger vi for at ressursene inngår i et kretsløp, som gjør at vi reduserer behovet for å hente ut nye naturressurser.

Energiutnyttelse

Om avfallet ikke kan gjenbrukes eller gjenvinnes kan ofte deler eller hele gjenstanden brennes og konverteres til energi. Dette erstatter bruk av olje og gass til oppvarming, og gjør at helse- og miljøskadelige stoffer tas ut av kretsløpet. Til tross for at avfall har et generelt lavt energiinnhold, og at brenning av avfall slipper ut miljøgasser, er allikevell energiutnyttelse mer miljøvennlig enn deponering.

Deponering

Deponering av avfall er når avfallet blir kjørt til et deponi, tidligere kjent som søppelfylling. I forskrift om deponering av avfall fastslås det at det ikke lenger er lov å deponere nedbrytbart avfall i Norge, dette inkluderer avfall som restavfall, hageavfall, papir, kartong og trevirke. Deponering fører også til luktplager, smittefare, skadedyr, utlekking av miljøgifter, forsøpling og støy. Avfall som kjøres til deponi og råtner står idag for omtrent 2,5% av norske klimagassutslipp (for

Kildesortering og Gjenvinning., 2018b).

2.3 Autentisering

Når en applikasjon skal lagre data som er knyttet til spesifikke brukere, er det viktig at kun brukerne som er ment til å se dataen har tilgang. For å forsikre oss om dette bruker vi autentisering. I dette delkapittelet ser vi på hva som ligger i begrepet autentisering og hvorfor er det ønskelig å autentisere brukere. Vi ser også på sammenhengen mellom autentisering og autorisering, og autentiseringsmetoden som brukes i prosjektet hvor ett av kravene til applikasjonen var at brukerne måtte autentisere seg før de kunne bruke tjenesten. Etter at brukeren er autentisert vil den ha autorisasjonen til å sende bestillinger med den innloggede brukeren.

2.3.1 Definisjon

Autentisering er prosessen der en bruker bekrefter identiteten sin ved å oppgi riktig legitimasjon eller oppfylle krav og andre kriterier som kreves av systemet for å få tilgang. Det er slik systemet er sikker på at du er den du utgir deg for å være. Autentisering kan deles inn i tre ulike faktorer (Gibson, 2011). Autentiseringssystemer må bruke minst én av disse faktorene.

Noe du vet

Dette er den vanligste metoden å autentisere på. Dette gjøres ved å utnytte kunnskap kun brukeren har, altså et passord eller en pin-kode. Dette innebærer at brukeren må huske passordet, noe som åpner for sikkerhetsutfordringer.

Noe du har

Dette refererer til noe brukeren har med seg. Et eksempel en husnøkkel eller NTNU kortene som brukes for adgang i NTNU sine ulike bygg.

Noe du er

Når du bruker biometri til å identifisere brukeren, bruker du noe brukeren er. Biometri er måling av biologiske mønstre. Eksempler på bruk av dette er bruk av fingeravtrykk eller ansiktsgjenkjenning.

2.3.2 Motivasjon

Som tidligere nevnt brukes autentisering som regel når personlig data skal lagres. Det brukes altså når man har behov for å kontrollere tilgangen og beskytte ressurser. I tillegg får man en garanti på at brukeren holdes ansvarlig for handlinger dem gjør. I en verden der mer og mer blir gjort over nettet, er dette en viktig prosess for å stoppe identitetstyveri, slik at ikke dataen havner i feile hender.

2.3.3 OTP - One Time Password

OTP står for "One Time Password", eller engangspassord på norsk, og brukes i autentiseringsprosessen. Et OTP er en kode eller passord som genereres når en bruker skal autentiseres. Denne koden kan enten være numerisk eller alfanumerisk. De vanligste måtene å levere OTP-er til brukeren på er SMS-basert eller applikasjonsbasert. Når det er SMS-basert blir koden tilsendt til brukeren på deres oppgitte mobilnummer, mens applikasjonsbasert krever at brukeren må ha tilgang til en spesifikk applikasjon som genererer de. Bruken av OTP-er gjør det vanskeligere for angripere å få tak i passordet, siden det kun er gyldig en gang (Bhatia, 2018). Dette baserer seg på faktoren "noe du har", nemlig telefonen eller applikasjonen. Dette har fordelen at brukerne ikke trenger å huske passord til tjenesten.

Wasteflow benytter seg av denne formen for autentisering. Andre applikasjoner som tar i bruk denne autentiseringsmetoden er for eksempel handleapplikasjonen til Rema 1000, "Æ".

2.3.4 Autentisering og autorisering

Et begrep som brukes i sammenheng med autentisering, er autorisering. Autorisering er prosessen der brukeren får innvilget tilgang til spesifikke funksjoner eller ressurser (Okta, u.d.).

I en sikker løsning kommer alltid autorisering etter autentisering. Brukerne må først kunne identifisere seg og får deretter innvilget tillatelse til spesifikke tjenester.

For eksempel vil brukere i Wasteflow måtte logge inn (autentiseres) før de blir autorisert og gitt tilgang til informasjon og muligheten til å legge inn en bestilling.

2.3.5 To-faktor-autentisering

Noe som brukes oftere for digitale løsninger idag er to-faktor-autentisering. To-faktor-autentisering utnytter to av de tre autentiseringsfaktorene. Dette er vanligvis noe brukeren vet (et passord), og noe brukeren har (en mobiltelefon). Hensikten med to-faktor-autentisering er at det skal være vanskeligere for uautoriserte å få tilgang. Denne metoden brukes for eksempel i betalingskort, hvor "noe du har" er selve kortet, og "noe du vet" er pin-koden. I moderne digitale løsninger er det vanlig å kombinere et passord med en engangskode fra en applikasjone eller SMS på en mobiltelefon.

2.4 Systemutvikling

2.4.1 Smidig utvikling

Ifølge en artikkel gitt ut av IEEE Software skiller smidig utvikling seg fra den tradisjonelle utviklingen ved at smidig utvikling er iterativt, utforskende og fremvoksende (Dyba & Dingsoyr, 2009).

The Agile Manifesto

Prinsippene rundt smidig utvikling er oppsummert i "The Agile Manifesto" (Beck et al., 2001). Manifestet sine verdier lyder som følgende:

- **Individer og interaksjoner** over prosess og verktøy
- **Fungerende programvare** over omfattende dokumentasjon
- **Kundesamarbeid** over kontrakt forhandlinger
- **Tilpassning til endring** over å følge en plan

Smidig utvikling vs. tradisjonell utvikling

Smidig utvikling kom som en motreaksjon på tradisjonell utviklingen. Tradisjonell utvikling utnytter ofte "vannfalls-metoden" hvor arbeidet er som et samlebånd. Dette samlebåndet tar lite hensyn til endringer som kan oppstå underveis. Smidig utvikling derimot, er mer tilpasset utfordringene i en uforutsigbar verden. Smidig utvikling er ikke like bundet av regler og rekkefølge, ting skjer iterativt. I artikkelen snakkes det også om at smidig utvikling har vist seg å øke tilfredsheten med arbeidet, effektiviteten og kundetilfredsheten. De viser til forskning som viser at prosjekter som bruker smidig utvikling kan endres enklere og har muligheten til å demonstrere virksomhetsverdi mer effektivt enn tradisjonelle prosjekter (Highsmith & Cockburn, 2001).

Fordelen med smidig utvikling er at det tar hensyn til at utviklere og kunder kan ha ulikt syn på et produkt. Ved å involvere kunden mer, åpner det opp for dialog mellom partene. Dette gjør at partene kan justere prioriteringene sine, og om problemer skulle oppstå, finne en ny løsning. Dette samarbeidet har også den innvirkningen at kunden får et tydelig innblikk i hva som forårsaker problemer og utsettelser, og vil dermed justere forventinger og prioriteringer underveis.

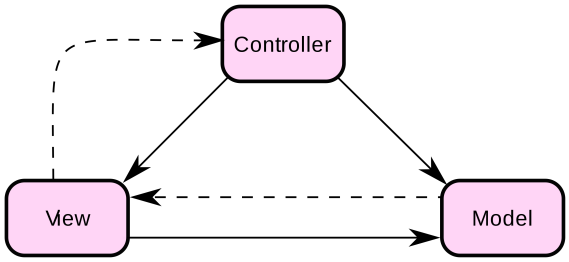
I en smidig prosess har man gjerne små iterasjoner som varer alt fra to til seks uker. I en iterasjon jobber utviklerne med produktet i henhold til det kunden ønsker. Etter en iterasjon møtes partene igjen for å gå over utviklingen, og blir enige om eventuelle endringer og veien videre. For å oppsummere så handler smidig utvikling om å lage og respondere til endringer.

2.4.2 Designmønster

Designmønster er noe som gir systematisk navn, motiverer og forklarer et generelt design som adresserer et gjentagende designproblem. Det beskriver problemet, løsningen og hvordan løsningen skal brukes og dens konsekvenser. Fordelen med et designmønster er at det gir oss kunnskapen vi trenger for å snakke om kodedesign på en bedre måte (Gamma, Helm, Johnson & Vlissides, u.d.).

Model-View-Controller

Et kjent designmønster er MVC, som er kort for model-view-controller. Trygve Reenskaug, mannen bak designmønsteret, beskiver MVC som en åpenbar løsning på det generelle problemet av å gi brukeren kontroll over informasjonen deres, som sett fra flere perspektiv (dataforening Høstakademiet, 1995). En utdypende beskrivelse er at ved hjelp av MVC vil dataen og brukergrensesnittet skilles, der dataen representeres av model og brukergrensesnittet av view. For at disse skal kunne kommunisere brukes controllere. Altså "model" holder på dataen, "view" viser dataen og "controller" flytter dataen. Designmønsteret er illustrert i figur 5. Fordelen ved dette designmønsteret er at man unngår duplisering av kode, og koden får god "Seperation Of Concerns", hvor forskjellige funksjonaliteter er separertt i forskjellige filer. Dette er designmønsteret teamet tok utgangspunkt i under utvikling av applikasjonen og er beskrevet i vedlegg C.



Figur 5: Oversikt over kommunikasjonen i MVC (Wikipedia, 2019).

2.4.3 Versjonskontroll

Versjonskontroll er et system for å håndtere forskjellige versjoner av et prosjekt. For hver endring gjort til filer i et prosjekt, må endringen spores. Et versjonskontrollsystem loggfører hver endring som skjer til en fil/filer, og tilbyr en måte å angre hver endring. (Tsitoara, 2020)

Når mer enn én person jobber på et prosjekt er det vanskelig å holde styr på endringer som skjer. Med versjonskontrollsystem kan flere personer jobbe med hver sin versjon av prosjektet (brancher), og bare flette inn endringene til hovedgrenen når de er fornøyd med jobben som er gjort. Gjennom versjonskontrollsystemet legges også meldinger om hva som blir endret, dette gjør at andre i teamet enkelt kan gå igjennom historikken for å se hva som har blitt gjort.

2.5 Mobilapplikasjon

2.5.1 Native, Hybrid og Cross-plattform applikasjoner

Vi deler gjerne mobilapplikasjoner inn i tre forksjellige kategorier: native applikasjoner, hybrid-applikasjoner og cross-plattform-applikasjoner. Det kan være litt uklart hva som skiller disse ulike typene, og for å utdype hva vi mener med cross-plattform er det dedikert et underkapittel til å skille de ulike typene.

Native applikasjoner

En applikasjon som er skrevet i et programmeringsspråk tilpasset et spesifikk operativsystem kalles en native applikasjon. Sammenlignet med andre type applikasjoner tilbyr native applikasjoner en konsekvent ytelse og er ofte mer optimalisert enn andre typer applikasjoner. Avhengig av hvilken plattform som skal brukes, bruker man ulike programmeringsspråk. For iOS brukes Objective-C og Swift, og for Android brukes Java eller Kotlin.

En native applikasjon får tilgang til alle funksjonalitetene til mobilen. Appen kan få direkte tilgang til mobilens hardware som f.eks. GPS, kamera og mikrofon. Et resultat av dette er at applikasjonene har høyere ytelse, og man får en bedre brukeropplevelse. En av ulempene med native applikasjoner er at du ikke kan bruke samme kodebase på tvers av operativsystemer. Dette krever ofte to forskjellige team spesialisert for hver plattform, og er svært ressurskrevende. De to kodebasene kan også føre til problemer dersom applikasjonen ikke har lik funksjonalitet på de ulike plattformene.

Hybrid-applikasjoner

I motsetning til native applikasjoner kan hybrid-applikasjoner lastes ned på mobiler uavhengig av opreativsystem, uten at man trenger å lage flere kodebaser. Den kan altså gjenbrukes på tvers av operativsystemer. Hybrid-applikasjoner kjøres inne i en native applikasjon og egne innebygde nettlesere ved å bruke Webkits som brukes av de ulike mobilene og skrives ofte i HTML og/eller JavaScript. Ulempen med hybrid applikasjoner er at det er vanskelig å få tilgang til mobiltelefonens funksjonaliteter, og applikasjonene får ikke den samme optimaliseringen som native applikasjoner, noe som kan gjøre dem mindre brukervennlige.

Cross-Plattform applikasjoner

Cross-plattform applikasjoner og hybrid applikasjoner kan virke som to sider av samme sak, men den eneste likheten mellom de er at begge kan gjenbruke samme kodebase. En cross-plattform applikasjon skiller seg fra hybrid applikasjoner ved at den bruker rammeverk og språk som kompilerer koden den har over til kildekode for en native applikasjon. Dette betyr at applikasjonen ikke er like begrenset som en hybrid applikasjon, siden applikasjonen har tilgang til funksjonaliteter og optimaliseres til hver plattform.

Cross-plattform applikasjoner har blitt ekstremt populære på grunn av fordelene ved dem. Rammeverkene React-Native og Flutter er to av rammeverkene som brukes mest under utvikling av cross-plattform applikasjoner.

Figur 6 viser de forskjellige typene applikasjoner med tilhørende språk/rammeverk.

Figur 6: Forskjellen mellom native-, hybrid- og corss-plattform-applikasjoner

2.6 Universell utforming

Universell utforming går ut på å sikre at alle, uavhengig av nedsatt funksjonsevne, kan delta. Om forutsetningene til en person ikke møter samfunnets forventninger oppstår det et funksjonsgap. Når man utvikler en digital løsning er det derfor viktig at alle brukerne kan ta i bruk løsningen uavhengig av forutsetningene. Dette gjør at løsningen kan få flere potensielle brukere og at brukerne får en følelse av at de er tatt hensyn til.

WCAG 2.0 - standarden

Web Content Accessibility Guidelines (WCAG) 2.0 er en internasjonal standard for universell utforming av nettsider. Etter 2013 ble universell utforming av IKT-løsninger lovpålagt, der 35 av suksesskriteriene i WCAG 2.0 gjelder nettløsninger. Applikasjoner faller også under dette kravet (Digitaliseringsdirektoratet, 2021).

De ulike kravene er delt inn i temaer. Temaene er alternativt format, tastaturbetjening, kontrast, navigasjon, presentasjon, skjema, struktur og styring av lyd. Noen av de kravene som er relevante for applikasjonen teamet har laget er:

- **1.4.3 Kontrast**: Kontrasten mellom teksten og bakgrunnen skal minst være 4,5:1
- **1.4.4 Endring av tekststørrelse**: Teksten kan endres til 200% størrelse uten tap av innhold eller funksjon
- **2.1.4 Navn, rolle, verdi**: Alle komponenter har navn og rolle bestemt i koden
- **2.4.3 Fokusrekkefølge**: innholdet i en logisk rekkefølge
- **2.4.4 Formål med lenke**: Alle lenkers mål og funksjoner fremgår tydelig av lenketeksten
- **2.4.5 Flere måter**: Tilbyr brukeren flere måter å navigere på
- **3.3.1 Identifikasjon av feil**: For feil som oppdages automatisk må du vise hvor feilen har oppstått og gi en tekstbeskrivelse av feilen.
- **3.3.2 Ledetekster eller instrukser**: Det vises ledetekster eller instruksjoner når du har skjemaelementer som må fylles ut.
- **3.3.3 Forslag til feil**: Dersom feil blir oppdaget automatisk, gi brukeren et forslag til hvordan feilen kan rettes.

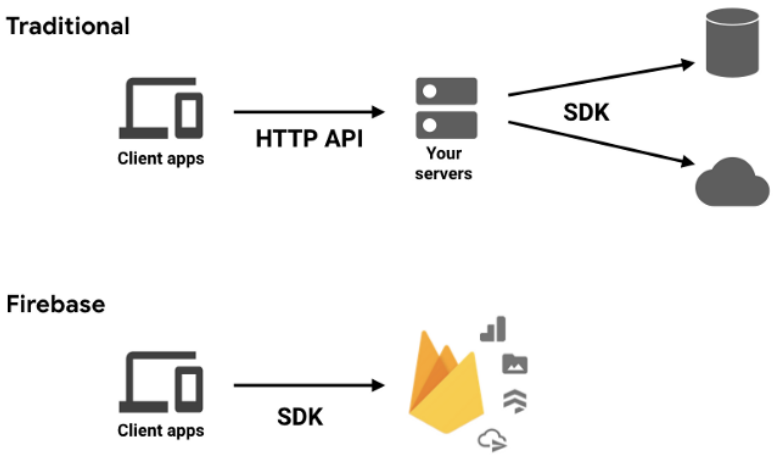
3 Valg av teknologi og metode

Dette kapittele utdyper og forklarer de de ulike teknologiene og verktøyene som er valgt i prosjektet. Kapittelet forteller også om hvilken metoder teamet anvender under arbeidsprosessen, og organiseringen av arbeidet i prosjektet.

3.1 Teknologi

3.1.1 Firebase

Firebase er en applikasjonsutviklingsplattform utviklet av Google som gjør det enklere å lage, skalere og publisere web- og mobilapplikasjoner. Det som gjør Firebase attraktivt er at det i de fleste tilfeller kan erstatte den tradisjonelle klient/tjener strukturen ved å ta over for tjeneren og tilby alt en applikasjon kan trenge, dette er illustrert i figur 7. Firebase forenkler altså arbeidet ved at teamet må bruke hovedfokuset på klienten. For å bruke de forskjellige tjenestene brukes forskjellige SDK-er (Software develpoment kit) som er godt dokumentert og enkle å bruke.



Figur 7: Firebase sammenlignet med en tradisjonell tjener

Firebase Authentication

Firebase authentication er Firebase sin ferdige autentiseringstjener. Denne tilbyr ferdig integrerte løsninger for blant annet autentisering via Apple, Google, GitHub, Mobil mm. Denne tjenesten håndterer alt som har med autentiseringen å gjøre, både når det gjelder verifisering, sikkerhet og håndtering av brukere. Firebase autentisering brukes derfor i applikasjonen til autentisering med mobil, samt håndtering av brukere.

Cloud Firestore

Cloud Firestore, forkortet til Firestore, er Firebase sin NoSQL-database. Denne databasen er dokumentbasert, som betyr at all data lagres i forskjellige dokumenter. Via SDK-en er det mulig å hente ut ett eller flere dokumenter fra databasen. Siden Firestore er en del av Firebase kan vi bruke autorisasjonsinfoen fra Firebase Authentication når en bruker forsøker å hente ut informasjon. Firestore blir brukt til lagring av all informasjon og alle bestillinger i applikasjonen.

Firebase Cloud Functions

Firebase Cloud Functions er skyfunksjoner som Firebase tilbbyr på sine tjenere for deg. Disse funksjonen kan, som i likhet med alle andre tjenester innenfor Firebase, kobles sammen med de andre tjenestene. I applikasjonen er Firebase Cloud Functions derfor brukt for å sende push-notifikasjoner til brukere ved endringer i bestillinger i Firestore.

3.1.2 Flutter

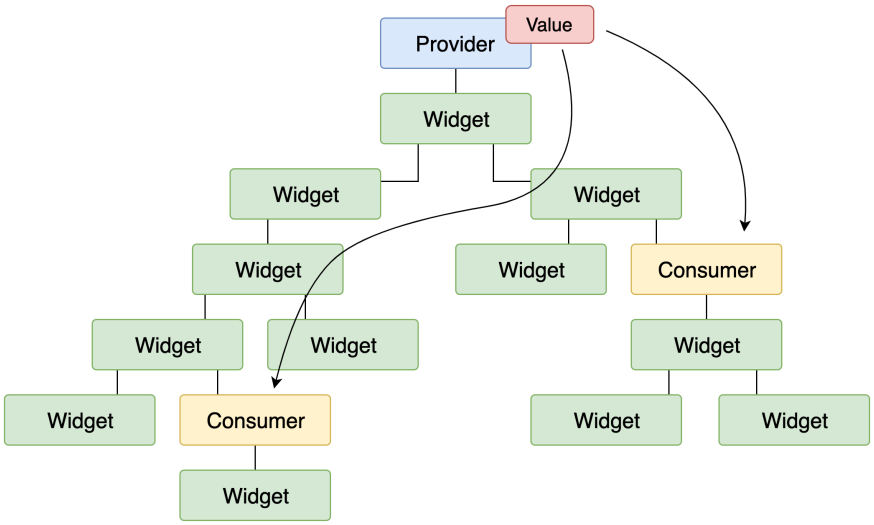
Flutter er et cross-plattform rammeverk. Målet til Flutter er at utviklere skal kunne levere applikasjoner med høy ytelse som føles som en native applikasjon på de ulike plattformene. Dette gjøres ved at Flutter bruker sin egen "rendering engine" som tegner opp widgets istedenfor den gitt av plattformen. Flutter skrives i spårket "dart", og baserer seg på at alt i applikasjonen er en "Widget". En Widget kan være alt fra tekst til bilder, knapper, menyer og animasjoner.

Oppbygging av applikasjonen

Måten Flutter bygger opp en applikasjon på er et Widget-tre. Hele applikasjonen starter som en enkelt widget som man bygger videre på med flere widgets. De som er kjent med React-rammeverket i JavaScript vil kjenne seg igjen i Flutter. Flutter Widgets kan sammenlignes med komponentene i React, og begge rammeverkene bruker tilstander på objekter for å optimalisere applikasjonen slik at kun Widgets som endrer seg blir bygget på nytt.

Provider

I applikasjonen brukes et vanlig designmønster innenfor Flutter, nemlig "Provider"-mønsteret. Her brukes Widgets som heter "Providers" eller forsørger på norsk. Providere inneholder ofte en verdi eller instans av en klasse. Denne Provideren plasseres så i Widget-treet. Hensikten med, og magien bak Providermønsteret, er at nå kan alle Widgets under denne Provideren i treet bruke denne samme verdien, disse Widgetsene kalles "Consumers", eller forbrukere. Mønsteret er svært nyttig for informasjon som vises/kreves forskjellige steder i applikasjonen. Providermønsteret sin deling av verdi er illustrert i figur 8.



Figur 8: Provider sin deling av verdier i Flutter sitt Widget-tre

3.1.3 GitHub

GitHub er en tjener som tilbyr versjonskontrollsystemet Git og lagring av kildekode. GitHub er et verktøy teamet har brukt en del tidligere og er en av verdens største tjenere av kildekode. GitHub tilbyr ikke bare et versjonskontrollsystem, men også et sosialt delingssystem som gjør det mulig å dele sin egen kode, eller lese andre sin kode. Dette gjorde det også mulig for oppdragsgiver å direkte følge fremgangen i prosjektet ved å følge med på prosjektet i GitHub.

Feature branching

For å organisere arbeidsflyten valgte vi å bruke "feature branching". Feature branching går ut på å separere utviklingen av ulike "features" eller funksjonaliteter i ulike "branches" eller grener. Ettersom hver sprint hadde fastsatte funksjonaliteter som skulle implementeres, gjorde dette det enkelt å holde styr på hvem som jobbet med hva. Det ble også lett å følge fremgangen for de ulike

funksjonalitetene ved å se på de forskjellige grenene. Etter en funksjonalitet var ferdig, ble denne slått sammen med hovedgrenen (kalt "main"), på denne måten var det kun aktive grener for de funksjonalitene som ble arbeidet med.

3.2 Metode

3.2.1 Arbeids- og rollefordeling

Smidig utvikling inspirert av SCRUM ble brukt, for nærmere detaljer se 4.3.3. Ettersom sprintene ble delt inn etter de funkjonelle egenskapene som var ønsket falt rollefordelingen ganske naturlig fra starten av. Hvert teammedlem jobbet på hver sin funksjonelle egenskap, og ved hjelp av feature branching var det lett å jobbe med ulike funksjonaliteter samtidig.

Senere i prosjektet ble arbeidsoppgavene mer separert. Sabine gikk tidlig over til å jobbe hovedsakelig med dokumentasjon, mens Sivert fortsatte å jobbe på applikasjonen. Underveis i sprint 3, da oppdragsgiver følte seg fornøyd og de viktigste bugs var fjernet, gikk begge over til å skrive dokumentasjon og jobbe med rapporten.

Under er en kort oppsummering av de ulike oppgaver teammedlemmene hadde gjennom prosjektet. Oppgavene er organisert etter sprintene i prosjektet.

Sprint 1

I denne sprinten var fokuset å få en MVP (Minimum viable product) som skulle være sammenkoblet med databasen. Dette innebar funksjonalitetene som omhandlet å legge inn en bestilling, autentisere og opprette brukere med mobil.

Sabine Seljeseth: var ansvarlig for oppsetting av prosjektet og utviklingen av funksjonaliteten som var nødvendig for å legge inn en bestilling og sammenkobling med Firebase og databasen. I tillegg jobbet hun med et utkast for hvordan en bestilling skulle vises i applikasjonen.

Sivert Utne: hadde ansvar for autentisering og innlogging med mobil, dette innebar også oppsett av navigering i applikasjonen, slik at en bruker kun har tilgang til applikasjonen dersom brukeren er logget inn.

Sprint 2

Hovedfokuset for denne sprinten var å få implementert de funksjonelle egenskapene. Brukeren skulle kunne gi tilbakemeldinger, kunne endre og kansellere bestillingen, se status for bestilling og få notifikasjoner når statusen endrer seg. I tillegg ble det gjort endringer etter tilbakemeldinger fra både oppdragsgiver og veileder.

Sabine Seljeseth: endret på hvordan bestillingene ble sendt etter tilbakemeldingene gitt fra oppdragsgiver og veileder. Hun implementerte også slik at applikasjonen selv finner ut antall seksjoner og pris ut fra dimensjonene til avfallet. Etter dette var ferdig ble dokumentasjonsarbeid hovedfokuset.

Sivert Utne: begynte med et par endringer som var ønsket fra oppdragsgiver og veileder, blant annet lagre all informasjon brukt i applikasjonen i databasen. Etter dette hadde han ansvaret for å implementere de resterende funksjonalitetene som skulle være ferdig i sprint 2.

Sprint 3

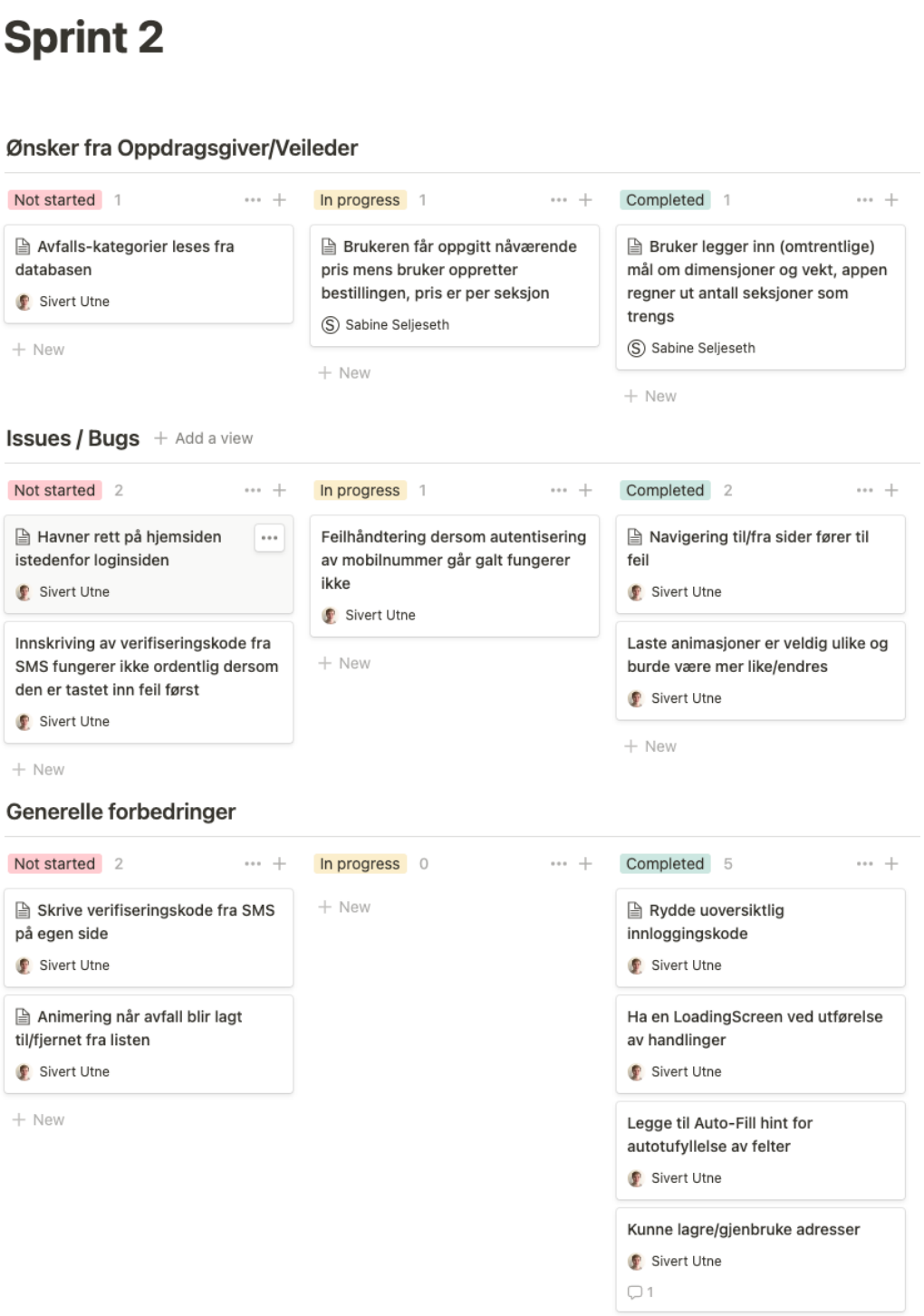
Denne sprinten gikk til bugfixing/ferdigstilling av applikasjonen. Når dette var fullført gikk arbeidet over til dokumentasjon.

Sabine Seljeseth: jobbet videre på hovedrapporten, og gjorde endringer til denne basert på tilbakemeldingene gitt fra veileder.

Sivert Utne: gjorde justeringene i applikasjonen og implementering av noen mindre funksjonaliteter og rettinger etter ønske fra oppdragsgiver. Deretter gikk jobbingen over til dokumentasjon.

3.2.2 Bruk av adminstrative verktøy

For å organisere hva som måtte gjøres ble Notion brukt. Notion er en plattform der man kan dele arbeidsplasser som kan inneholde "to-do"-lister, tidslinjer og kanban-tavler. Teamet brukte hovedsakelig kanban-tavlene for å ha en tydelig oversikt over hva som måtte gjøres, hvem som jobbet med hva, og hva som var fullført. Hvert sprintmål hadde sin egen tavle for å enkelt se hvilke funksjonaliteter som var ferdig. Disse tavlene ble også brukt for dokumentasjonen og ønsker/endringer fra veileder og oppdragsgiver. Hver tavle var delt inn i tre deler: 'Not started', 'In progress' og 'Completed'. Eksempel på en slik tavle vises i figur 9.



Figur 9: Kanban-tavle i Notion

Grunnen til at teamet valgte Notion var muligheten for å skreddersy arbeidsplassen underveis i prosjektets livsløp. Vi hadde muligheten til å bruke flere kanban-tavler på en oversiktlig måte, og organisere disse etter sprint og funksjonalitet. I Notion er det er også mulig å legge igjen kommentarer, samskrive, få varslinger om endringer og legge inn frister.

Fremdriftsplanen og timelisten til teamet ble laget i Microsfot Excel.

4 Resultater

Dette kapittelet tar for seg resultatene teamet oppnådde i prosjektet. De ulike resultatene er delt opp i tre underkategorier. Vitenskapelige resultater, der vi ser på den endelige applikasjonen. Her går vi gjennom både designet og hvordan de ulike funksjonalitetene fungerer. Videre ser vi på de ingeniørfagelige resultatene vi har oppnådd, hvor vi ser på grad av oppnåelse av de ikke-funksjonelle og funksjonelle egenskapene oppdragsgiveren ønsket. I tillegg ser vi også på resultatene av brukertestene som ble gjennomført etter applikasjonen var ferdig. Den siste underkategorien er administrative resultater. Her blir arbeidstimene som er brukt på prosjektet lagt frem, samt hvordan teamet forholdt seg til de planlagte aktivitetenes varighet. Utviklingsprosessen vil også diskuteres i denne delen av rapporten. Kapitellet legger kun frem resultatene. Hvorfor resultatene har blitt som de er diskuteres i kapittel 5.

4.1 Vitenskapelige resultater

I dette delkapittelet vil vi ta for oss design og funkasjonalitet i applikasjonen. Designet er basert på wireframes fra oppdragsgiver, disse finnes i kravsdokumentet (vedlegg B). Underkapitlene er strukturert etter den naturlige flyten i applikasjonen. Først går vi gjennom innloggingsprosessen, deretter bestillingsprosessen, etterfulgt av endringer av bestillinger og status, profilsiden, informasjonssiden og feilhåndteringer.

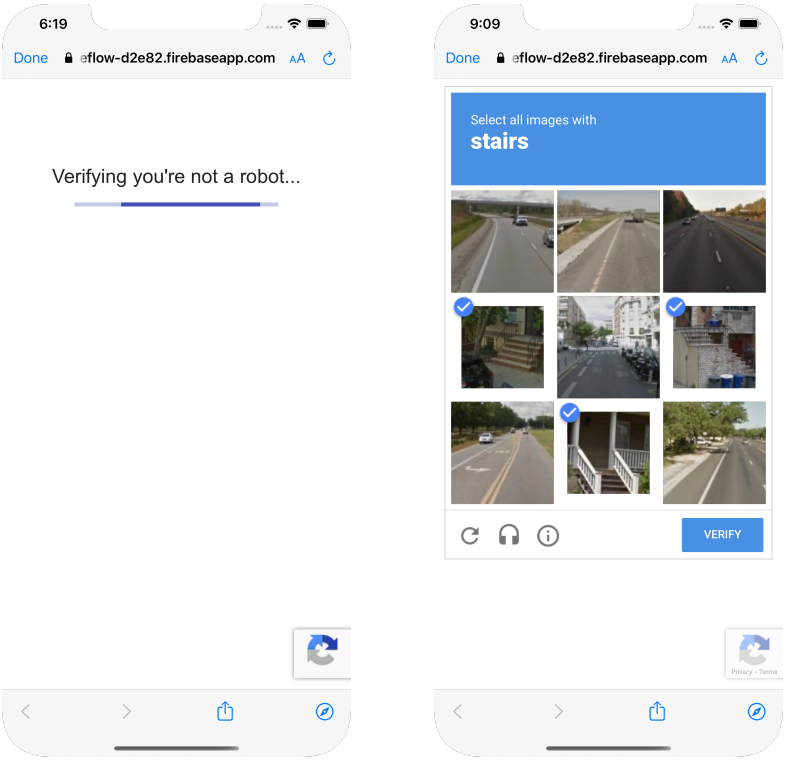
4.1.1 Innloggingsprosessen

Det første som møter brukeren er innloggingssiden til applikasjonen. Her må brukeren fylle inn mobilnummer, for å deretter bekrefte nummeret og gi Wasteflow tillatelse til å sende en verifiseringskode (OTP) til det oppgitte nummeret. Denne prosessen er illustrert i figur 10.

Figur 10: Innloggingsprosessen i Wasteflow

Autentisering

Når brukeren har godtatt at engangskoden kan sendes, sjekker appen først at det er en faktisk bruker som håndterer applikasjonen med CAPTCHA. Samtidig som brukeren utføre CAPTHCA testen, vil mobilnummeret autentiseres. Dersom brukeren har oppgitt ett ugyldig mobilnummer, eller nummeret av andre grunner ikke kan autentiseres, vil brukeren få feilmelding om dette, og må skrive inn mobilnummeret på nytt. Etter CAPTHCA-testen blir den seks sifrede koden sendt til mobilnummeret, og brukeren blir sendt til siden for verifisering av koden. CAPTHCA-testen er vist i figur 11.



Figur 11: CAPTHCA test av brukeren

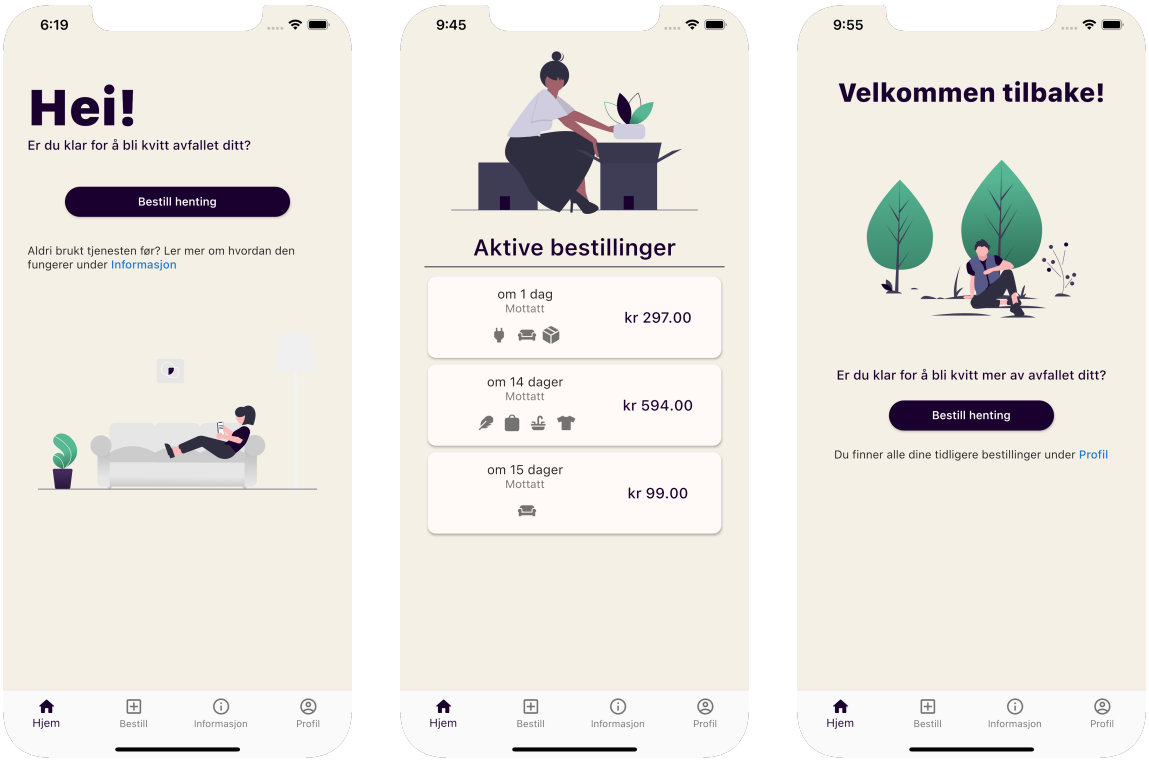
Innskriving av verifiseringskoden

Innskrivingen av koden skjer i seks felt for at det skal være tydelig hva brukeren skal skrive inn. Når alle seks feltene er fylt ut blir koden automatisk verifisert. Om brukeren har fylt inn riktig verifiseringskode, har brukeren blitt autentisert og sendes til hovedsiden. Hvis koden er ugyldig får brukeren beskjed om dette og må skrive in koden på nytt, dette er vist i figur 27a. Telefonen vvil også vibrere dersom koden er ugyldig. Ved andre feil som f.eks. at koden har utløpt blir brukeren sendt tilbake med beskjed om hva som gikk galt, og må sende en ny melding. Dersom det er første gang brukeren logger inn vil applikasjonen be om tillatelse til å sende push-notifikasjoner. Innskrivingen er demonstrert i figur 12

Figur 12: Innskriving av verifiseringskode

4.1.2 Hovedsiden

Etter innloggingsprosessen er gjennomført, havner brukeren på hovedsiden i applikasjonen. Denne vil se ulik ut avhengig av om brukeren har benyttet seg av tjenesten før. Om brukeren ikke har gjort dette, vil de lande på siden vi ser på figur 13a. Om brukeren har bestillinger som enda er aktive vil siden være lik som på figur 13b. Aktive bestillinger, er bestillinger som ennå ikke er blitt hentet og fraktet til gjenvinningsstasjonen. Om brukeren har lagt inn tidligere bestillinger, men ikke har noen aktive, vil de få opp hovedsiden som man kan se i figur 13c. Fra denne siden har brukeren muligheten til å navigere til de ulike sidene man ser nederst på siden i figur 13. Dette kan gjøres på flere måter. Brukeren kan velge om de vil bruke navigasjonsbaren nederst på siden, sveipe til høyre/venstre eller bruke de blå linkene vist i figur 13a. Brukeren vil dermed ha alternative måter å navigere seg igjennom innholdet i applikasjonen, slik som nevnt i WCAG 2.0 kravene (se kapittel 2.6).



(a) Før noen bestillinger er lagt inn

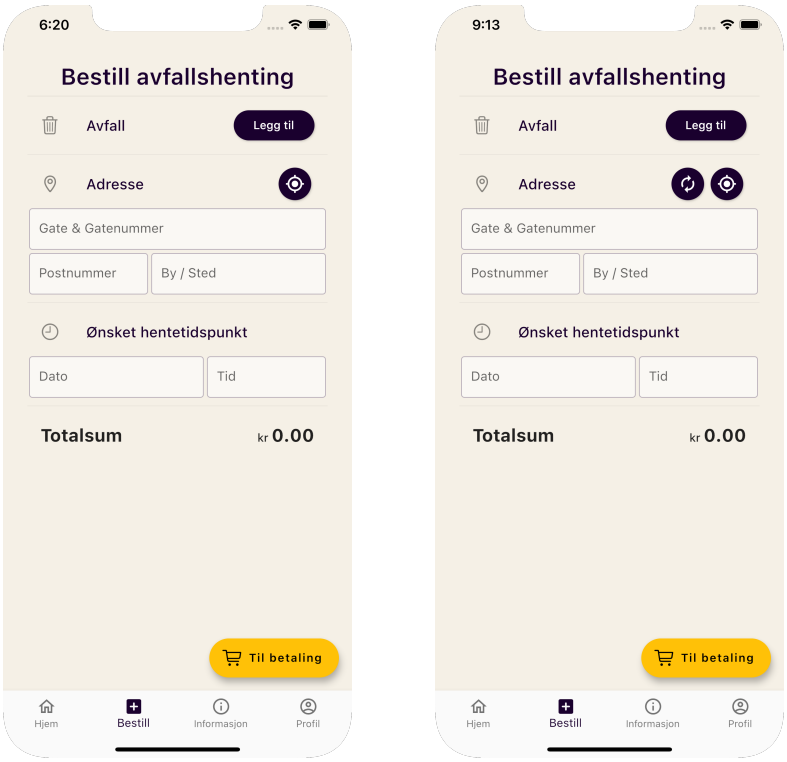
(b) Om brukren har aktive bestillinger

(c) Om man ikke har noen aktive bestillinger

Figur 13: De ulike hovedsidene i Wasteflow

4.1.3 Opprette en bestilling

Etter at brukeren har navigert til bestillingssiden, vil det også her være en ulik side avhengig av om brukeren har benyttet tjenesten før. Om det er brukerens første bestilling vil siden se ut som figur 14a. Om brukeren har lagt inn bestillinger før vil de også ha valget om å gjenbruke tidligere registrerte adresser, se figur 14b. Ulike metoder for å velge adresse beskrives senere.



(a) Om man ikke har bestilt før (b) Om man har bestilt før

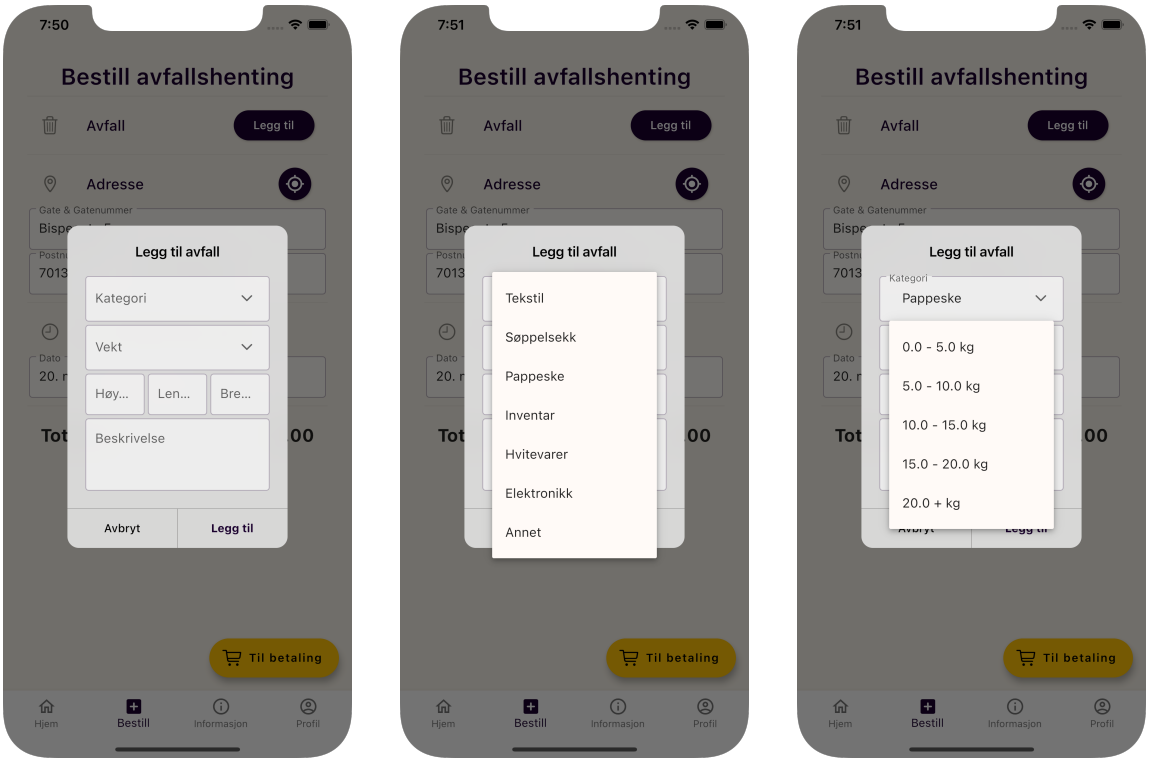
Figur 14: Bestillingssiden i Wasteflow

Avfall

Øverst på bestillingsiden vil brukeren legge til avfallet som ønskes hentet. Når brukeren trykker på "Legg til" knappen vil et popup-vindu dukke opp, slik man ser på figur 15. Her velger brukeren hvilken kategori avfallet hører til. En hjelpetekst, som utfyller kategorien, vil dukke opp i beskrivelsesfeltet under når en kategori er valgt. En bruker som er usikker på hvilken kategori avfallet faller under, vil dermed få hjelp.

Videre fyller brukeren ut vekten av gjenstanden. Her er vekten delt inn i intervaller, der tanken er at brukeren må betale ekstra for tyngre gjenstander. Dette er enda ikke implementert, men valget er lagt til for å gjøre det lettere for videreutvikling. Dropdown menyen for dette ser vi i figur 15c. Legge merke til at alle elementer som dropdown menyer, knapper osv. alle har lik utforming, dette er for å gjøre det tydelig hva elementene gjør ved å opprettholde en konsistent design gjennom applikasjonen.

Videre fyller brukeren inn målene til gjenstanden. Her tillates kun tallinnfylling, målene må være mellom 0.1 meter og 5 meter. Brukeren har også valget om å fylle inn en beskrivelse av gjenstanden, men dette er valgfritt. Om en bruker legger inn en bestilling for inventar, kan de for eksempel bruke beskrivelsesfeltet til å spesifisere at det er en sofa og et bord som skal hentes.



(a) Før brukeren har fylt ut (b) Valg av kategori (c) Valg av vekt

Figur 15: Popup-vinduet for å legge til avfall i en bestilling

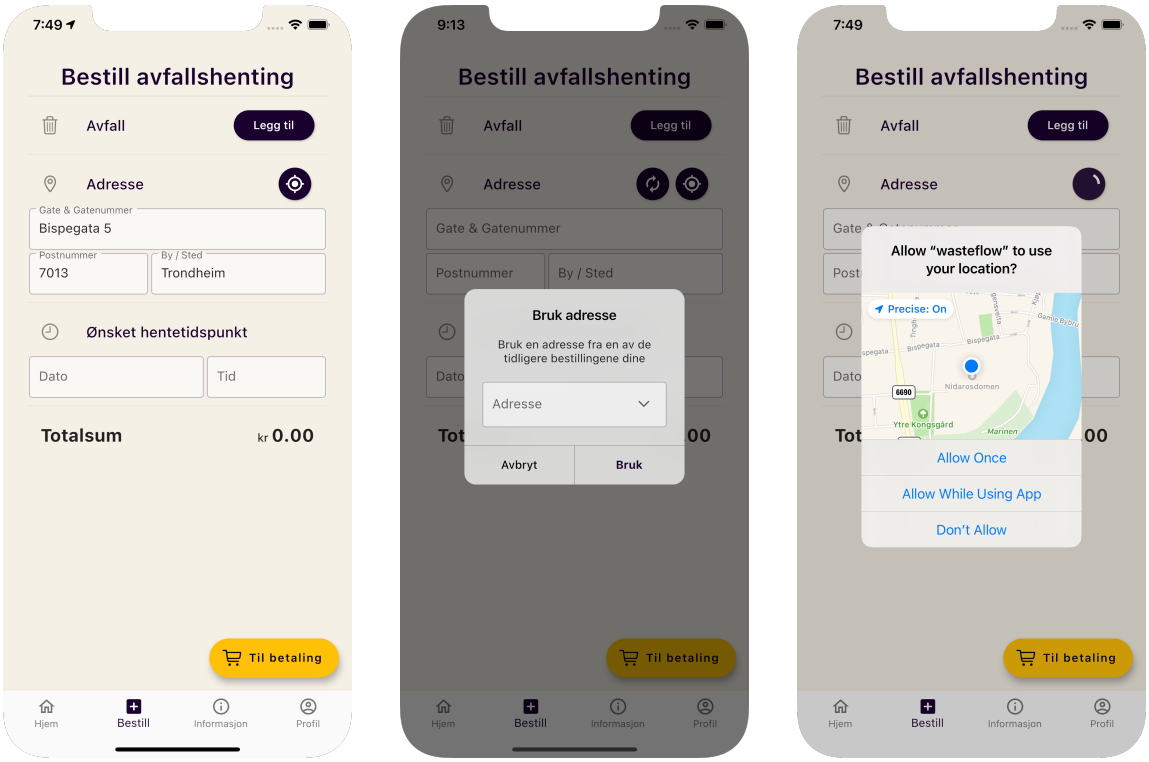
Adresse

Etter alt avfall som skal hentes er lagt til, går brukeren videre til adresseutfylling. Dette kan skje på tre forskjellige måter.

Brukeren kan manuelt eller med autoutfylling skrive inn addressen.

Om brukeren har lagt inn bestillinger tidligere, har brukeren valget om å gjenbruke en tidligere adresse, som vist i figur 16b. I figur 16b får brukeren opp en liste med alle tidligere adresser de har brukt i bestillinger. Dersom brukeren kun har brukt én adresse før, vil de ikke få opp denne dialogboksen, applikasjonen vil heller automatisk velge og fylle inn denne éne adressen.

Det sist valget brukeren har, er å bruke nåværende posisjon til å fylle ut adressen, som vist i figur 16c. Dette gjøres ved å trykke på knappen på høyresiden av adresse. Brukeren vil få opp et popup-vindu slik som vist på figur 16c, der de må godta at applikasjoen bruker posisjonen til mobiltelefonen.

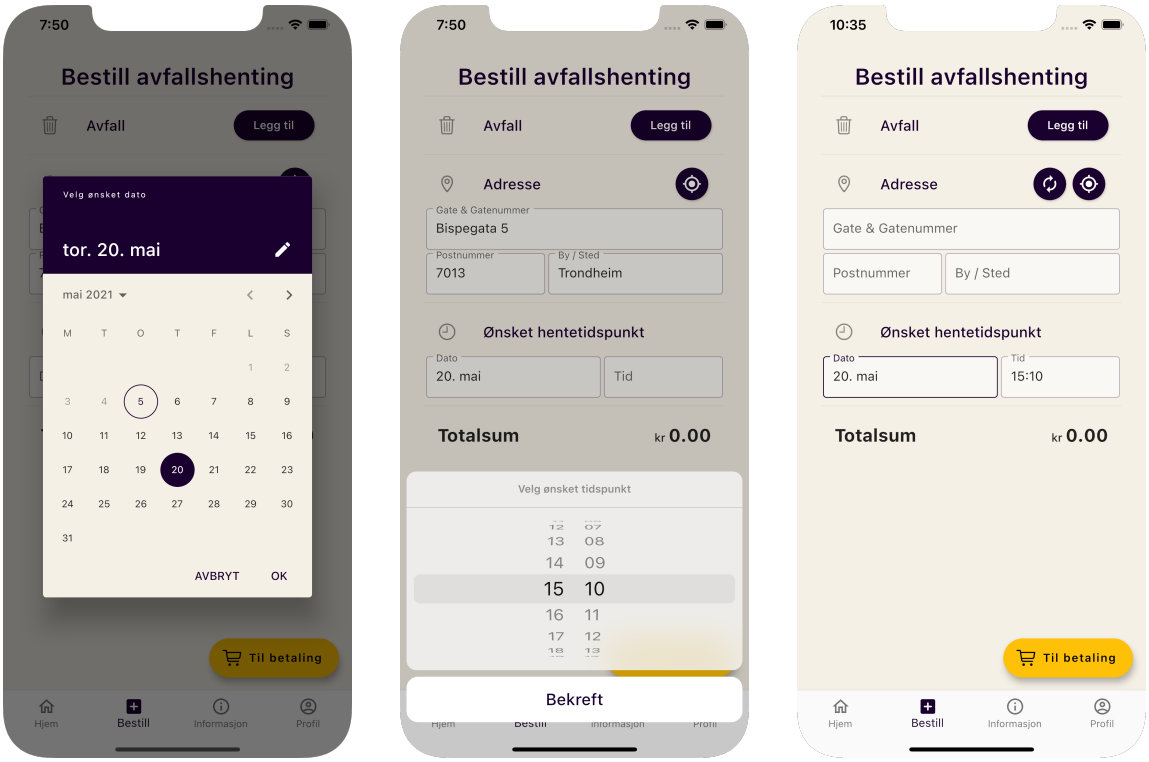


(a) Manuell registrering (b) Bruk av tidligere adresser (c) Bruk av brukerens posisjon

Figur 16: De ulike valgene for adresseregistrering

Dato og tid

Dato og hentetidspunkt følger prosessen vist i figur 17, der brukeren velger ønsket dato og tidspunkt. Tidspunkt er foreløpig ikke begrenset, og dette burde endres slik at henting kun kan bestilles til aktuelle tidspunkter. Datoene er begrenset til å måtte være mellom dagens dato, og 60 dager frem i tid.



(a) Valg av dato (b) Valg av tidspunkt (c) Dato og tid er valgt

Figur 17: Valg av dato og tidspunkt

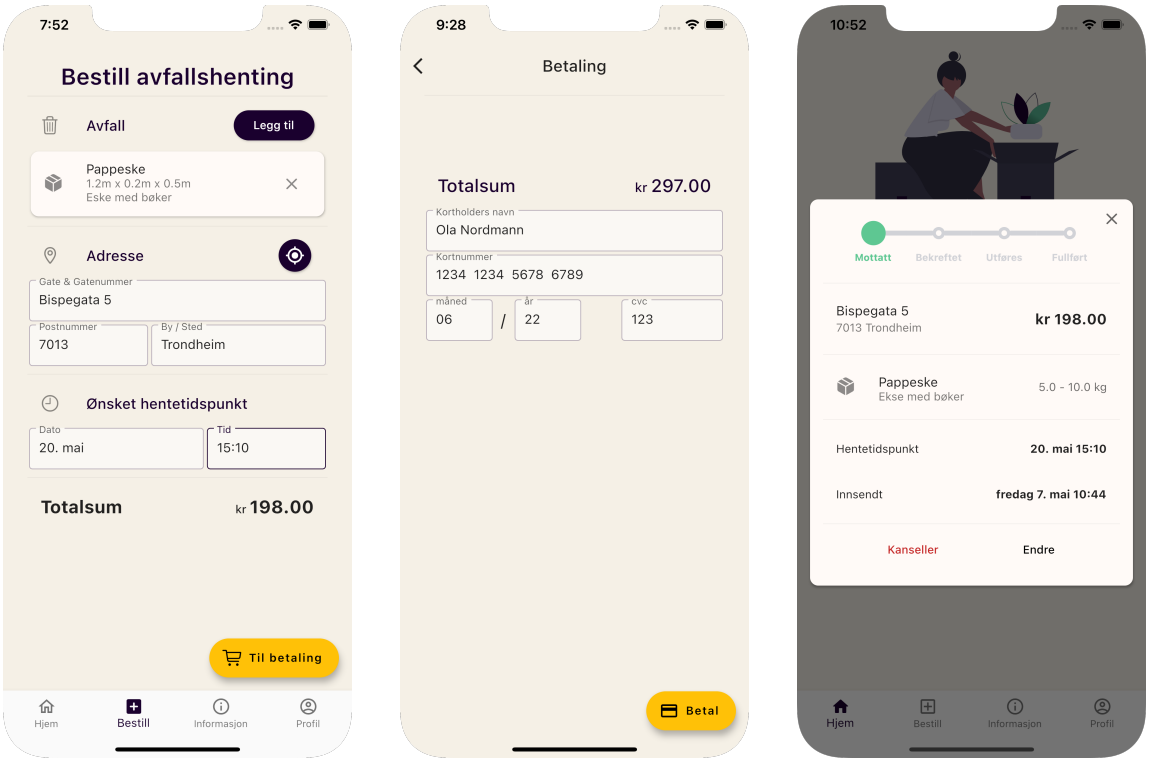
Sending og betaling

På figur 18a ser vi hvordan en ferdig utfylt bestilling ser ut. Prisen på bestillingen er avhengig av størrelsen. En bestilling vil minimum trenge én seksjon i hentebilen. En seksjon er et bestemt volum av hentebilen. Hentebilen er altså delt inn i flere seksjoner. Om det totale volumet av avfallet får plass i en seksjon, vil brukeren kun være nødt til å betale for den seksjonen. Om brukeren har mer avfall enn det er plass til i en seksjonen vil de trenge flere seksjoner, avhengig av hvor mye mer avfall de har i bestillingen. Prisen vil øke i takt med seksjonene som trengs for å frakte avfallet.

Når alle feltene er utfylt, altså det er registrert minst én avfallsgjenstand og adresse, dato og tid for henting er valgt, er bestillingen gyldig, for å sende bestillingen må brukeren først betale for den.

Betalingssiden er kun en stedfortreder til en faktisk betalingsside. Dette er diskutert nærmere i kapittel 4.2.

Når brukeren bekrefter bestillingen med "Betal" knappen, blir de sendt til en oversikt over bestillingen. Statusen på bestillingen er illustrert på toppen av oversiktsiden, se figur 18c.



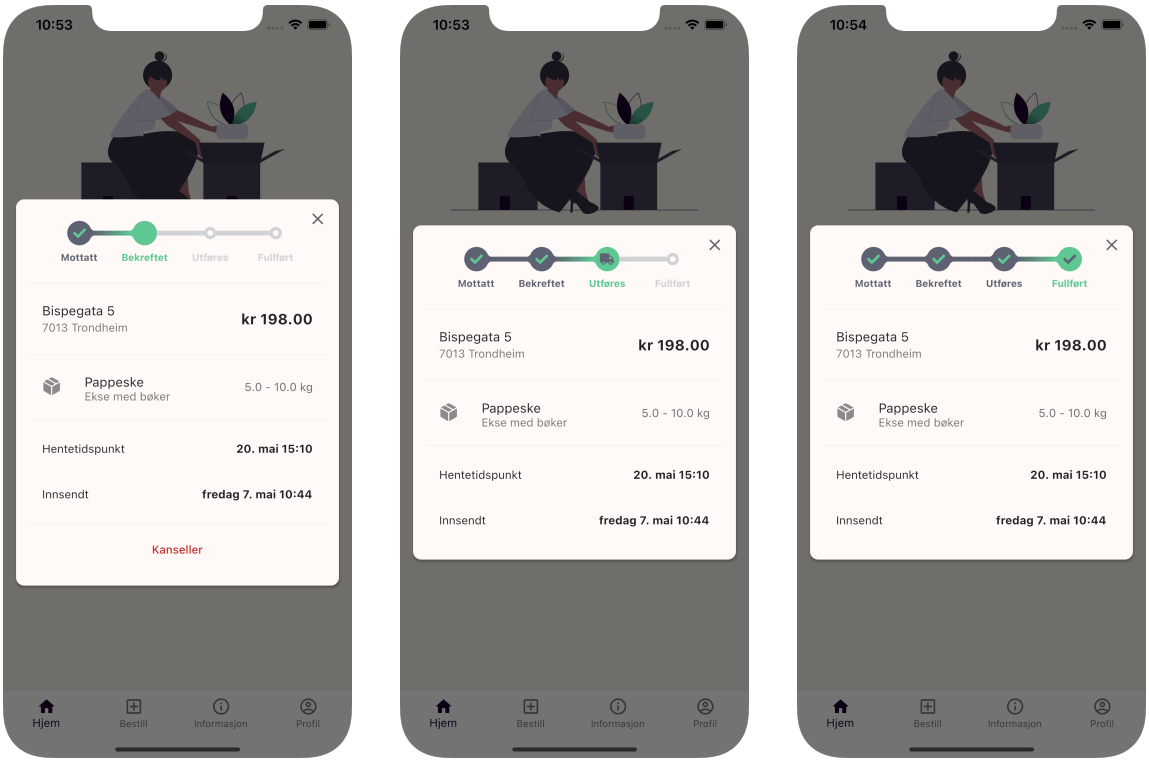
(a) Bestilling klar til sending (b) Betalingssiden (c) Oversikt over bestillingen

Figur 18: Sluttprosessen av en bestilling i Wasteflow

4.1.4 Endringer i status og bestilling

Endring i status

Etterhvert som statusen oppdateres vil oversiktssiden gjøre det samme. Brukeren kan endre på bestillingen helt til den er blitt bekreftet av gjenvinningsstasjonen, se figur 18c. Når bestillingen er blitt bekreftet, vil muligheten for å endre bestillingen falle bort, se figur 19a. Brukeren kan fortsatt kansellere bestillingen, denne muligheten vil falle bort når hentebilen er på vei, og brukeren vil kun se informasjonen over bestillingen, se figur 19b. Etter en bestilling er fullført vil den arkiveres inne på brukerens oversikt over bestillinger, se 22c.

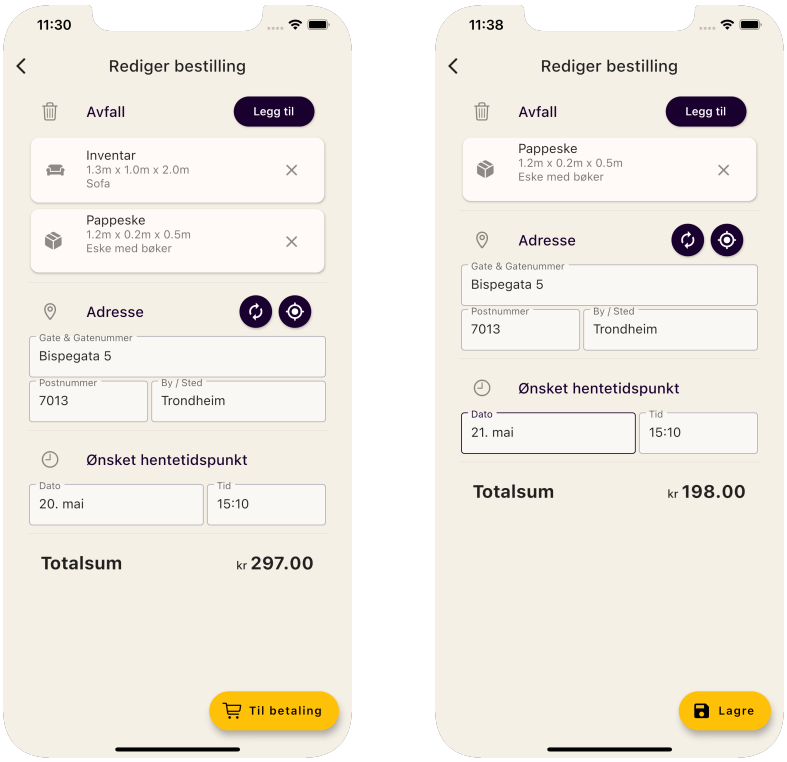


(a) Bestillingen er bekreftet (b) Hentebilen er på vei (c) Bestillingen er hentet

Figur 19: Forskjellige statuser for en bestilling

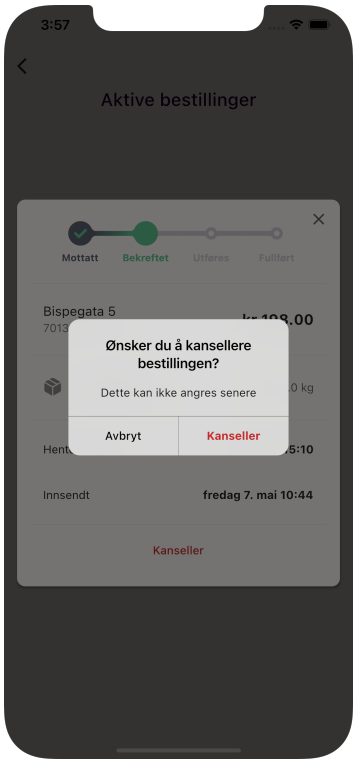
Endring i bestilling

Dersom brukeren ønsker å redigere bestillingen sin før den er blitt bekreftet kan brukeren gjøre det. Dersom brukeren gjør endringer som fører til at prisen endres, er brukeren nødt til å gjennomføre betalingen igjen. Redigeringssiden vil da se ut slik som den gjør på 20a. Dersom brukeren kun gjør endringer som ikke har innvirkning på prisen vil redigeringssiden se ut som figur 20b. Da vil brukeren slippe å gå igjennom betalingsprosessen igjen, og de vil kun være nødt til å trykke på "Lagre"-knappen nederst på siden.



(a) Gjenstander er lagt til (b) Endring av dato

Figur 20: De ulike sidene for å redigere en bestilling i Wasteflow



Figur 21: Dialogboks for bekreftelse på at brukeren vil kansellere bestillingen

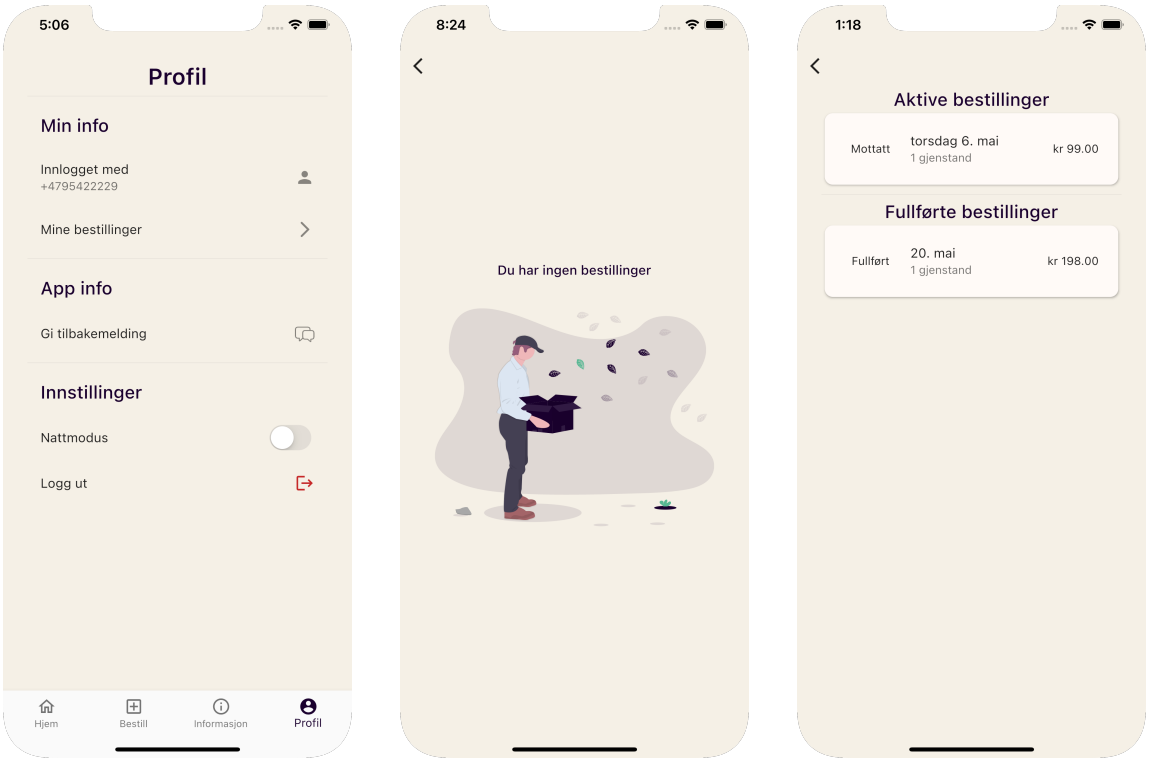
Kansellering

Dersom brukeren ønsker å kansellere en bestilling, gjøres dette ved å trykke på kanselleringsknappen på oversiktssiden. Dette kan gjøres helt til statusen har blitt endret til "utføres". For å forsikre oss om at brukeren vil kansellere bestillingen må brukeren bekrefte dette i en egen dialog. Her får brukeren beskjed om at handlingen ikke kan angres senere, dialogen er vist i figur 21. Etter kanselleringen vil bestillingen slettes fra databasen og dermed også fjernes fra oversikten til brukeren.

4.1.5 Profilsiden

Etter at brukeren har logget inn kan de navigere til sin profilside. Dette kan gjøres ved å enten bruke navigasjonsbaren nederst på siden eller sveipe bortover til profilsiden. Inne på profilsiden har brukeren oversikt over hvilket mobilnummer som er registrert til kontoen, samt muligheten til å gå inn på oversikten over alle sine tidligere bestillinger, se figur 22. Inne på oversikten for brukerens bestillinger ligger både aktive og fullførte bestillinger, som vist i figur 22c. Om brukeren ikke har noen bestillinger, vil oversiktet være den samme som på figur 22b.

Innstillingene for applikasjonen befinner seg også inne på profilsiden, der har de muligheten til å benytte applikasjonen i nattmodus. Mer informasjon om dette kommer i delkapittel 4.1.9. Profilsiden er også der brukeren logger seg av applikasjonen.

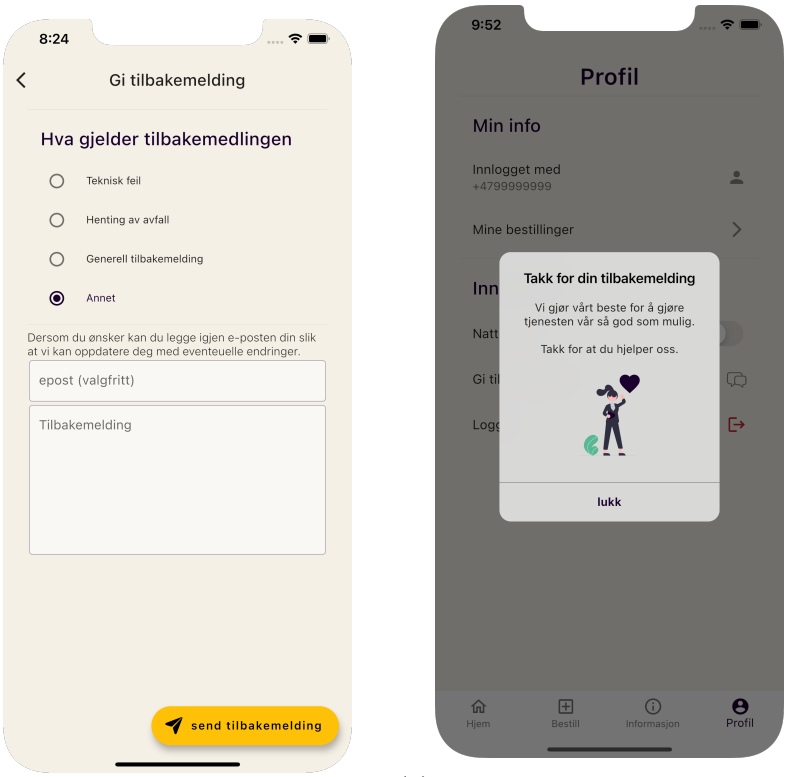


(a) Profilsiden (b) Ingen tidligere bestillinger (c) Aktive/fullførte bestillinger

Figur 22: Brukerens profilside og oversikt over bestillinger

Opprette tilbakemelding

Den siste funksjonaliteten inne på profilsiden er tilbakemeldingsfunksjonen. Denne finner man under ”App info”. Brukeren kan gi tilbakemelding om det skulle oppstå problemer, eller om de har generelle tilbakemeldinger. Inne på tilbakemeldingssiden kan brukeren velge hva tilbakemeldingen dreier seg om, samt et felt for e-postadresse utfylling og et felt for utfylling av selve tilbakemeldingen, se figur 23a. Epost utfylling er frivelig, og gir Wasteflow teamet muligheten til å følge opp tilbakemeldingen. Etter tilbakemeldigen er sendt vil brukeren få en bekreftelse på dette, som vist i figur 23b.



(a) Tilbakemeldingssiden

(b) Bekreftelse av innsendt tilbakemelding

Figur 23: Tilbakemeldingssiden i Wasteflow

4.1.6 Informasjonssiden

Den siste siden brukeren kan navigere til er informasjonssiden. På denne siden vil informasjon over hvordan tjenesten fungerer ligge. Siden er foreløpig en stedforetreder, men det er tenkt at brukeren her kan finne all informasjon om pris, henting osv. Informasjonsiden er vist i figur 24.

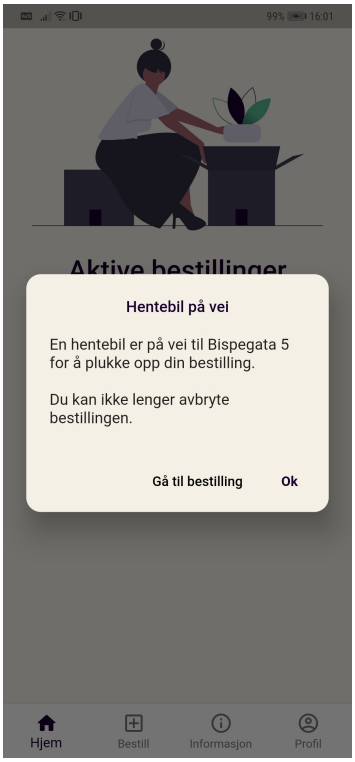
Figur 24: Informasjonssiden i Wasteflow

4.1.7 Notifikasjoner

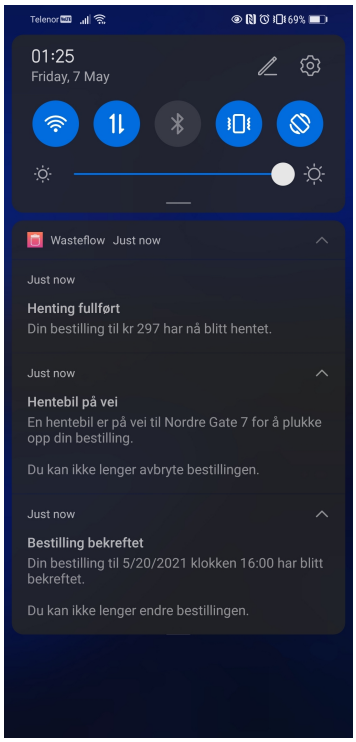
Figur 25: Applikasjonsikon med varsler

Brukeren vil få notifikasjoner hver gang statusen på en av deres bestillinger oppdateres. Dette vil dem få uavhengig om dem har applikasjonen åpen eller lukket. Om brukeren har en eller flere varsler som ikke er åpnet, vil disse vises på applikasjonsikonet til Wasteflow, som vist i figur 25.

Om brukeren har applikasjonen åpen vil notifikasjonen vises som i figur 26a. Hvis ikke applikasjone er åpen vil de ligge i brukerens varslingssenter, se figur 26b.



(a) Inne i applikasjonen



(b) Notifikasjoner i brukerens varslingssenter

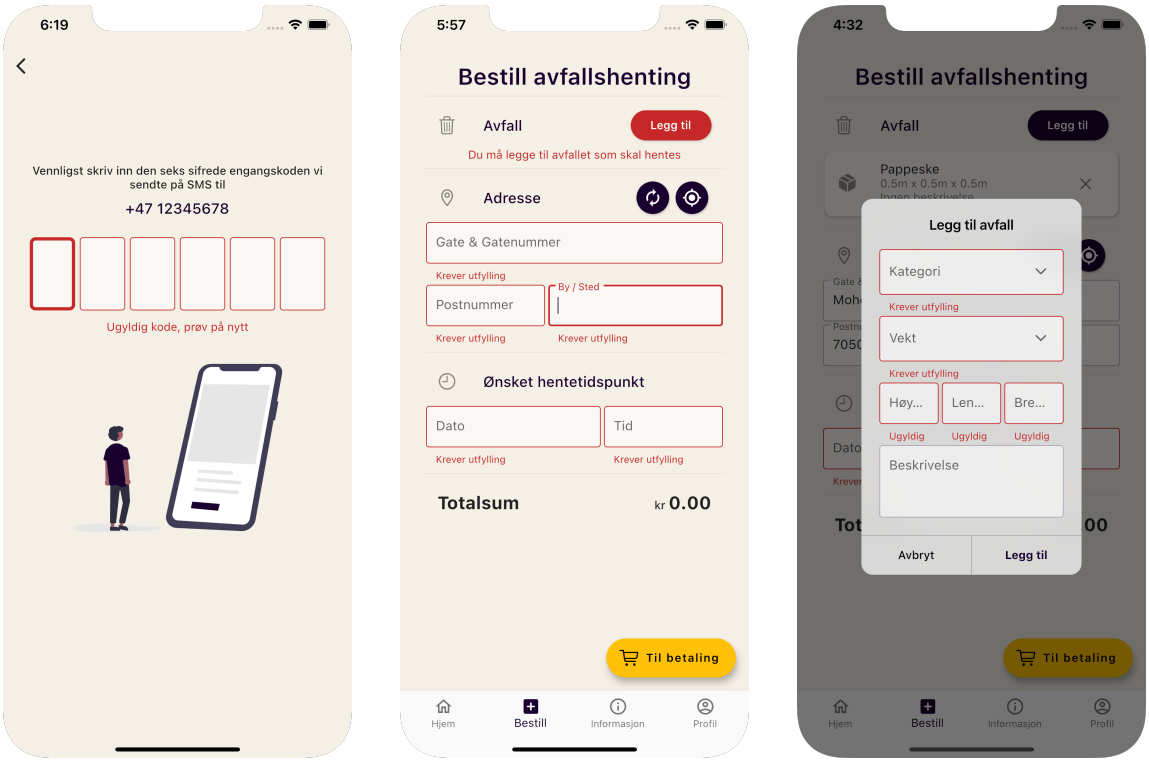
Figur 26: Visning av notifikasjoner

4.1.8 Feilhåndtering

For å sørge for at brukeren vet hva som er lov og ikke, og for å sørge for at ingen ufullstendige bestillinger havner i systemet, brukes validering av input. Dette inngår i kravene til WCAG 2.0, som nevnt i kapittel 2.6. Feil skal automatisk oppdages og brukeren skal bli informert om hvor feilen oppsto og hvordan den kan rettes.

Alle innfyllingsfelter har egne valideringssjekker som er skreddersydd de ulike inputene brukeren blir bedt om. I figur 27 ser man eksempler på de ulike feilmeldingene brukeren kan få.

På bestillingssiden må alle felt være utfylt for at brukeren kan legge inn bestillingen. Postnummeret må oppfylle kravene om å være fire siffer. Her sjekkes det også om bestillingen har noe avfall før den kan legges inn, der hver avfallsgjenstand må oppfylle dimensjonskravene, som er beskrevet i delkapittel 4.1.3. Om brukeren prøver å sende inn en tom bestilling vil de få opp feilmeldingene slik som vist i figur 27b. Om kun noen av feltene mangler vil kun de manglende feltene gi feilmelding. Det er slike tilbakemeldinger for alle obligatoriske felt i applikasjonen. Det er også lagt til vibrering i mobiltelefonen dersom brukeren prøver å lagre endringer som ikke er gyldige.



(a) Ugyldig verifiseringskode (b) Obligatoriske felt mangler (c) Ugyldige verdier for avfall

Figur 27: Eksempler på feilmeldinger

Figur 28: Siden som vises når applikasjonen mister internett-tilkoblingen

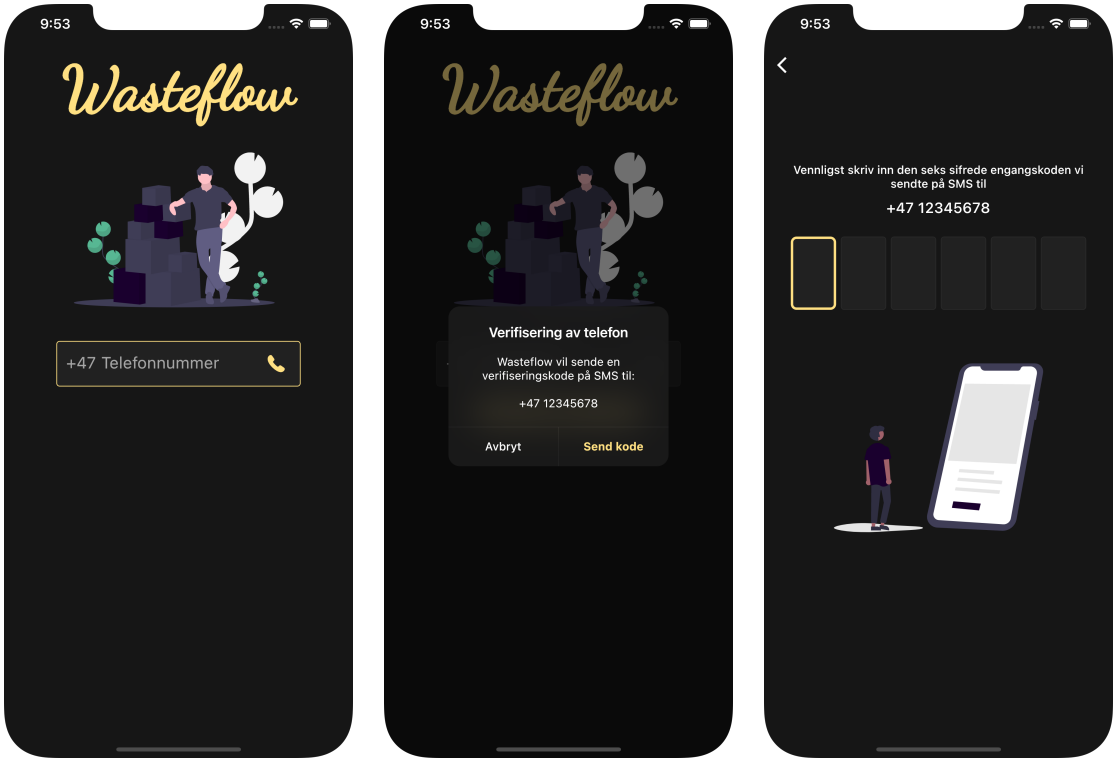
Internett-tilkobling

I situasjoner hvor brukeren mister internett-tilkoblingen, vil brukeren bli sendt til en egen side som informerer om dette. Denne siden ser vi i figur 28. Etter brukeren har fått internett-tilkobingen tilbake må brukeren trykke på knappen "Prøv på nytt". Om forbindelsen er tilbake vil brukeren sendes tilbake til applikasjonen.

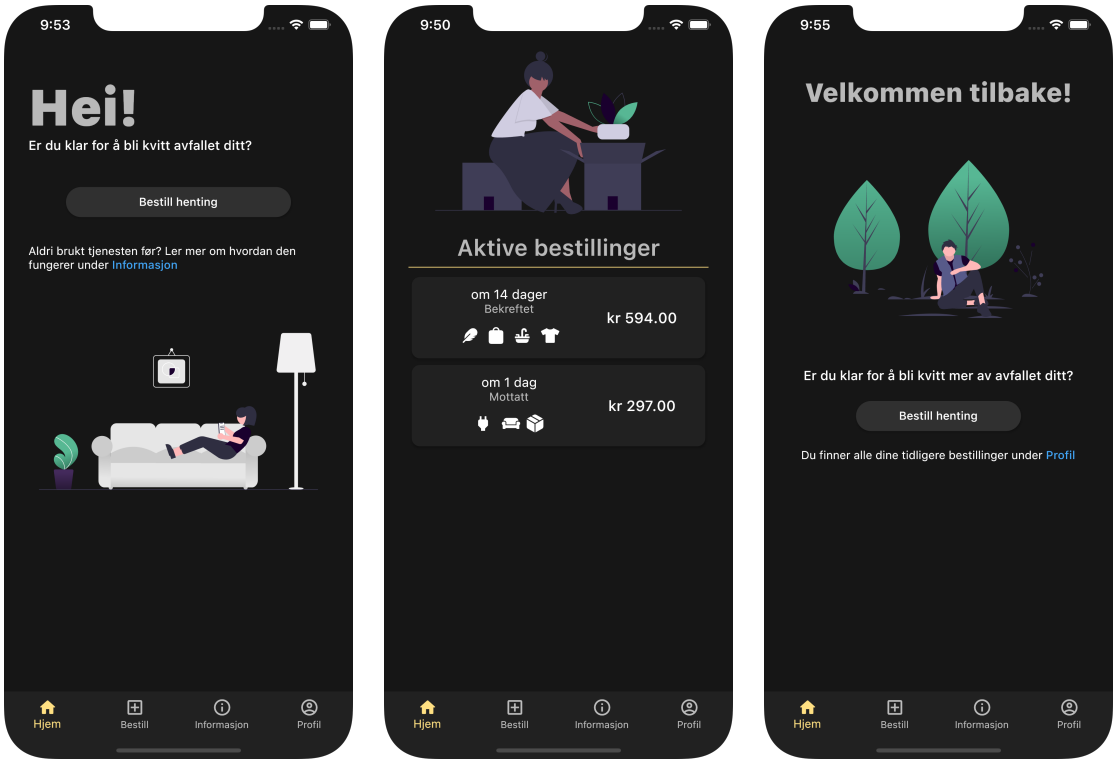
4.1.9 Nattmodus

En funksjonalitet som er lagt til, og som blir mer og mer vanlig i andre applikasjoner, er nattmodus, eller "dark mode". Dette endrer fargetemaet for hver side av applikasjonen til en mørk versjon. Mange brukere foretrekker denne typen applikasjoner da de er mindre anstrengende for øynene. Nattmodus har også blitt vist til å ha en positiv effekt på mobiler med OLED skjerm sin batteribruk. En åpen applikasjon i vanlig modus ble vist til å bruke opptil 43% mer batteri enn i nattmodus (Plikk, 2019). Brukeren kan velge å bytte til nattmodus under "Innstillinger" på profilsiden (se figur 22a). Applikasjonen vil, frem til brukeren endrer det, følge systeminnstillingene til mobiltelefonen. Altså vil applikasjonen automatisk være i nattmodus dersom dette er valgt for mobiltelefonen. Innstillingen lar brukeren skru av eller på nattmodus etter eget ønske. Dersom brukeren vil gå tilbake til å følge systeminnstillingene til mobiltelefonen, kan brukeren holde inne knappen "Nattmodus", og bruker vil få beskjed om at applikasjonen igjen følger systeminnstillingne.

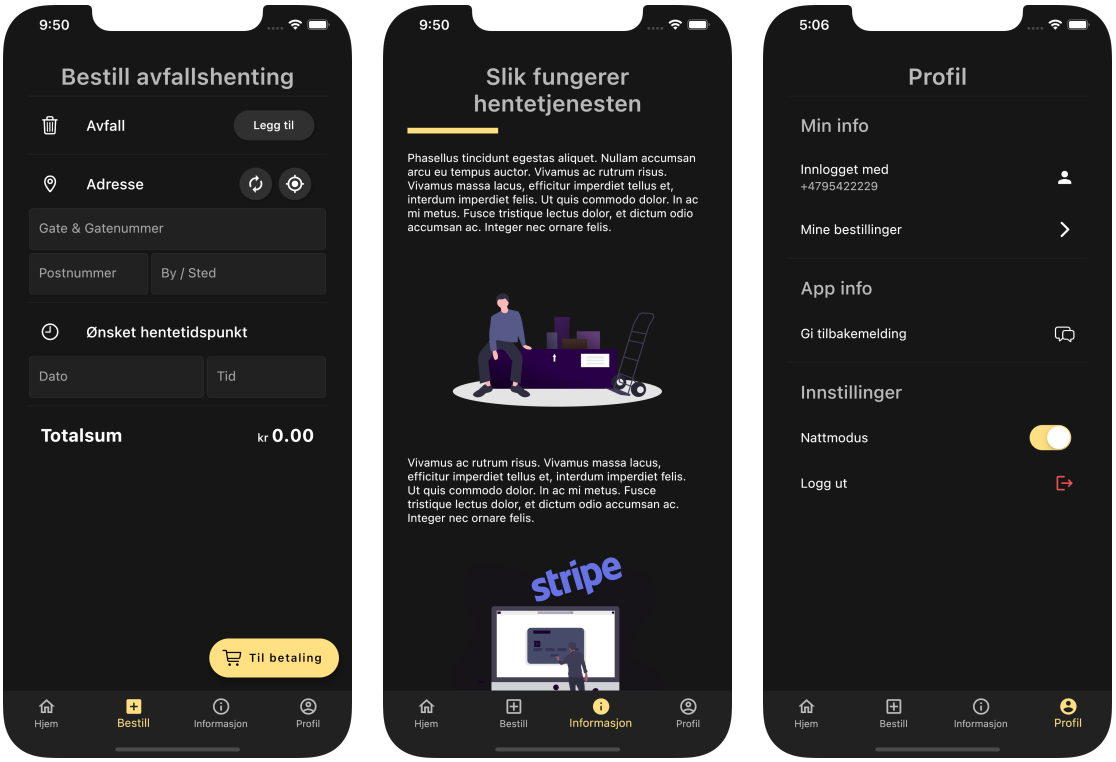
I figurene 29, 30 og 31 ser vi eksempler av hvordan applikasjonen ser ut i nattmodus. Nattmodus vil gjelde for alle sidene og visninger i applikasjonen, dette er bare noen få utdrag.



Figur 29: Innloggingsprosessen i nattmodus



Figur 30: Hovedside i nattmodus



Figur 31: Resterende hovedsider i nattmodus

4.2 Ingeniørfaglige resultater

Kravene for applikasjonen var delt inn i funksjonelle og ikke-funksjonelle krav, se Visjonsdokumentet i vedlegg A. I starten av prosjektet ble noen av kravene eller deler av kravene bortprioritert, dette blir beskrevet nærmere i kapittel 4.2.1 og 4.2.2.

4.2.1 Funksjonelle krav

I tabell 1 er de funksjonelle kravene listet opp og sortert etter prioriteringen de fikk av oppdragsgiver. Hvert krav er bygget opp av mindre delkrav, for en mer detaljert oversikt over hvilke kriterier som måtte møtes for å oppfylle kravene.

Hvert delkrav er markert med "●" dersom det var planlagt å gjennomføre, og "✓" dersom det ble fullført. Kravene som ikke ble fullført er markert med "✗". "∼" betyr at kravet har blitt sløyfet, enten under planleggingsfasen eller underveis. At kravene er sløyfet betyr ikke at de ikke lenger er ønsket, men at de ikke lenger var prioritet til å implementeres i løpet av bacheloroppgaven.

Nr.	Funksjonellt krav	Delkrav			
1	Legge inn bestilling av avfall	1.1	Valg av type avfall som skal hentes	●	✓
		1.2	Adresse for henting	●	✓
		1.3	Tidsintervall for henting	●	✓
		1.5	Prisen for bestillingen skal vises	●	✓
2	Oppretting av bruker	2.1	Autentisering med mobilnummer	●	✓
		2.2	Bestillinger koblet til brukeren	●	✓
		2.3	Brukeren kan legge inn navn	●	∼
3	Kunne se bestillingsinfo	3.1	Oversikt over avfallsgjenstander	●	✓
		3.2	Adressen oppgitt	●	✓
		3.3	Tidspunkt for henting	●	✓
4	Se status for bestilling	4.1	Få opp at bestillingen er mottatt	●	✓
		4.2	Få opp når hentebilen er på vei	●	✓
		4.3	Se om bestillingen er fullført	●	✓
		4.4	Prisen for bestillingen skal vises	●	✓
		4.5	Kunne se hvor hentebilen er på kartet	∼	
		4.6	Se "estimated time of arrival"	∼	
5	Endre bestilling	5.1	Endre avfallsgjenstandene	●	✓
		5.2	Kunne endre adresse for henting	●	✓
		5.3	Kunne endre tidspunkt for henting	●	✓
6	Kansellere bestilling	6.1	Få bekreftelse når den er kansellert	●	✓
7	Få notifikasjoner	7.1	Om status endrer seg	●	✓
8	Muligheten for å gi tilbakemelding	8.1	Om systemet	●	✓
		8.2	Etter gjennomført bestilling	●	✗
9	Kunne se prisoversikt	9.1	Kunne se pris pr seksjon i hentebil	●	✗
10	Kunne endre profilinformasjon	10.1	Kunne endre navn på brukeren	●	∼
11	Betale for bestilling	11.1	Kunne gjennomføre betaling i applikasjonen	∼	

Tabell 1: Status over implementasjonen av de funksjonelle kravene

Underveis i prosjektet kom oppdragsgiver med nye ønsker om funksjonaliteter og endringer. Dette er altså ikke funksjonelle krav teamet hadde planlagt å gjennomføre, men funksjonelle ønsker teamet har utført dersom oppdragsgiver ønsket det. Disse er listet opp på samme vis som de funksjonelle kravene i tabell 2.

Funksjonelle ønsker	
En bestilling består av avfall som tilhører ulike kategorier	✓
Status for om bestilling er avbrutt/kansellert/ikke kan gjennomføres	✗
Beregne antall seksjoner utifra avfallet	✓
En tidslinje over statusen til en bestilling	✓
Hver ordre har en historikk med tidspunkt	✗

Tabell 2: Status over implementasjonen av de funksjonelle ønskene

4.2.2 Ikke-funksjonelle krav

I tillegg til de funksjonelle kravene var det ikke-funksjonelle krav som skulle oppfylles. Disse er beskrevet i tabell 3, der statusen for implementasjonen av disse kravene er oppgitt. Krav markert med "✓" er gjennomført og krav markert med "✗" er ikke gjennomført.

Ikke-funksjonelle krav	
Applikasjonen skal lages med Flutter	✓
Bruk av Googles Firebase Cloud Firestore for database	✓
Bruk av Googles Firebase Cloud Functions som server	✗
Kompatibel med både iOS og Android	✓
Bruk av Cloud Firestores sikkerhetsregler for å sikre sensetive opplysninger	✓
Bruk av stripe som betalingstjeneste	✗

Tabell 3: Status over implementasjonen av de ikke-funksjonelle kravene

4.2.3 User-stories

I kravsdokumentet (vedlegg B), er en rekke user-stories listet opp. Under er en oversikt over hvor mange av disse som ble fullført. For fullførte user-stories er alle akseptansekriteriene oppfylt. Delvis fullførte user-stories er user-stories der de fleste akseptansekriteriene er oppfylt, men ikke alle. Ikke fullført er user-stories der ingen av akseptansekriteriene er oppfylt.

User-Stories	**Fullført**	**Delvis Fullført**	**Ikke Fullført**
13	8 (61.54%)	2 (15.38%)	3 (23.08%)

Tabell 4: Dekningsgrad av user-stories

De user-storiesene som var delvis eller ikke fullført er listet under. De delvis fullførte er listet med de uferdige akseptansekriteriene.

- **Delvis fullført:** Opprette bruker
  - Felt for navn
  - Lagre informasjon om adresser og epost for kvitteringer

  - Godkjenne vilkår
- **Delvis fullført:** Status for bestilling
  - Se forventet hentetidspunkt
  - Se hentebilen på kartet
- **Ikke fullført:** Endre bruker
- **Ikke fullført:** Vite hva tjenesten koster
- **Ikke fullført:** Betale for bestilling

4.2.4 Brukertester

For å kartlegge brukervennlighet til applikasjonen ble det gjennomført brukertester. Hovedfokuset for disse testene var å finne ut hvor intuitivt designet av applikasjonen virket.

Det ble gjennomført brukertester av fem ulike personer ved sluttfasen av utviklingen. Alle ble spurt et par bakgrunnsspørsmål, deretter ble det gitt en rekke oppgaver som skulle gjøres inne i applikasjonen. Til slutt ble det spurt et par spørsmål om hva de nettopp testet, for mer detaljer se vedlegg F. Alle signerte en samtykkekontrakt, slik at observasjonene og intervjudataen kunne brukes i denne rapporten. Malen for denne er vedlagt, se vedlegg E

De mest sentrale spørsmålene og svarene er beskrevet i tabell 5 og 6. Ingen av de fem brukerne som gjennomførte testen hadde vært borte i en lignende tjeneste tidligere, til tross for at de aller fleste hadde vært å levert avfall hos gjenvinningsstasjoner tidligere. De av dem som hadde vært hos en gjenvinningsstasjon tidligere hadde kjørt med henger, der noen av de nevnte at de måtte stå i kø.

De fleste brukerne klarte å navigere rundt i appen og gjennomføre oppgavene som ble gitt uten hjelp, med noen unntak. Et par av brukerne trengte hjelp når de skulle ferdigstille bestillingen. Disse observasjonene er tatt opp nærmere, se kapittel 5.1.1 og kapittel 5.2.3.

Spørsmål	**Antall brukere**	
	Ja	**Nei**
Har du brukt en lignende tjeneste tidligere?	0	5
Har du noen gang trengt å levere avfall til en gjenvinningsstasjon før?	3	2

Tabell 5: Spørsmål før brukertesten

Spørsmål	**Antall brukere**	
	Ja	**Nei**
Hadde du brukt tjenesten du nettopp testet om den var tilgjengelig?	4	1

Tabell 6: Spørsmål etter brukertesten

4.3 Administrative resultater

Dette delkapittelet vil gi et overordnet bilde over resultatene fra prosjekthåndboka som er vedlagt. I prosjekthåndboka finner man fremdriftsplanen i sin fullstendighet med timeliste og statusrapporter, samt møtereferater og innkallinger fra alle møter iløpet av prosjektet.

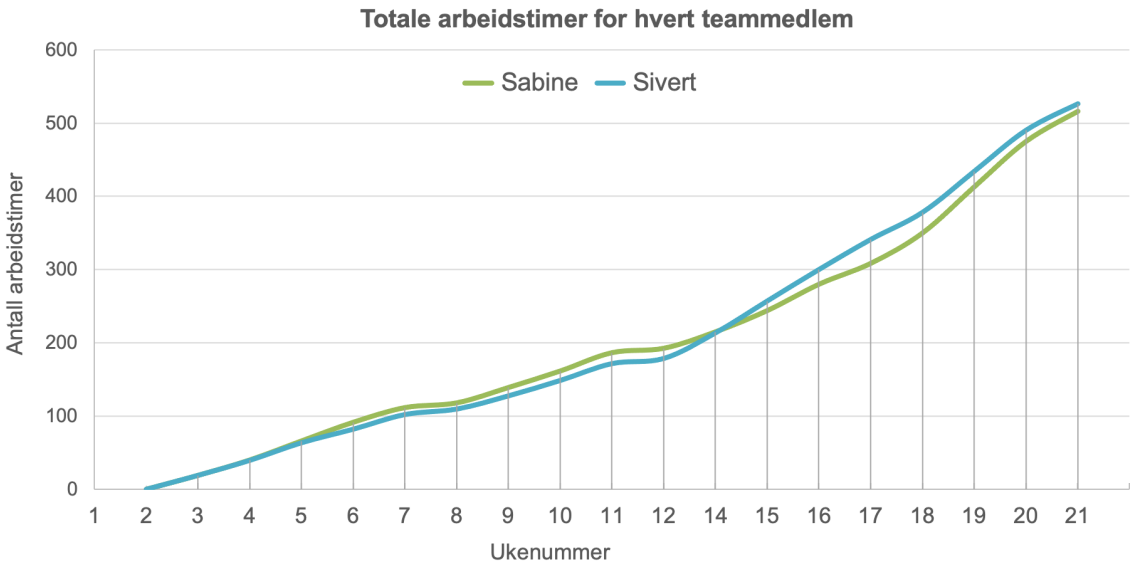
4.3.1 Timeregnskap

Tabell 7 viser det totale antallet arbeidstimer for hvert teammedlem gjennom prosjektet. En fullstending timeliste med ukentlige rapporter finnes i prosjekthåndboka. Bacheloroppgaven hadde en forventet arbeidsmengde på totalt 501,5 arbeidstimer.

Teammedlem	Totalt antall arbeidstimer
Sabine Seljeseth	516
Sivert Utne	526

Tabell 7: Totale arbeidstimer i prosjektet

Figur 32 viser hvordan den totale mengden arbeidstimer har økt fra starten av prosjektet til slutten.



Figur 32: Utvikling av totale arbeidstimer for prosjektet

4.3.2 Fremdriftsplan

Tabell 8 inneholder en oppsummert fremdriftsplan. Fremdriftsplanen ble opprettet i starten av prosjektet med forventet antall arbeidsdager på forskjellige faser og oppgaver. I tabellen sammenlignes disse estimatene med de faktiske arbeidsdagene for å se hvor flinke teamet har vært til å følge planen. Tabellen viser kun varigheten av periodene, og ikke om periodene ble gjennomført innenfor de planlagte datoene. Årsaker og drøfting rundt resultatene blir gjort i kapittel 5.3. Verdien i tabell 8 er i antall arbeidsdager. For full oversikt se fremdriftsplanen i prosjekthåndboka.

Aktivitet	Planlagt varighet	Faktisk varighet
Forarbeid	4	6
Oppstartsfasen	11	12
Sprint 1 - MVP	14	14
Sprint 2 - Utvide funksjonalitet	14	14
Sprint 3 - Finpussing og testing	14	8
Avlustningsfasen - Rapportskriving	12	18
Hovedrapport	34	35

Tabell 8: Sammenligning av planlagt og faktisk varighet av faser fra fremdriftsplanen

4.3.3 Utviklingsprosessen

Systemutviklingsteknikken som ble brukt er inspirert av SCRUM. Grunnet teamets størrelse var det overflødig å følge SCRUM slavisk, spesielt med hensyn til SCRUM-master og standup møter. Teamet plukket istedenfor ut de elementene som var nyttig.

Arbeidsprosessen ble delt inn i tre ulike sprinter, der hver sprint varte 14 dager. Før hver sprint var det møte med både oppdragsgiver og veileder der målene for sprinten ble gjennomgått. Dette sørget for at alle partene var informert om hva som skulle gjøres i prosjektet. Sprintmålene var knyttet til produktets funksjonelle egenskaper som nevnt i visjonsdokumentet (se vedlegg A). Der de funksjonelle egenskapene med høyest prioritet ble prioritert i sprintene.

Etter hver sprint ble det holdt et nytt møte, der gruppa oppsummerte hvilke av sprintmålene som ble gjennomført, og hva som eventuelt måtte videreføres til neste sprint. Her ble også eventuelle ønsker fra oppdragsgiver gjennomgått. Grunnet koronapandemien, og dens innvirkning på hverdagen, ble alle møtene med oppdragsgiver og veileder gjennomført over Microsoft Teams.

5 Diskusjon

I denne delen av rapporten vil vi drøfte resultatene vi presenterte i forrige kapittel. Vi ser på hvordan sluttproduktet ble i forhold til oppdragsgiverens forventninger, og hvorfor noen av kravene falt bort. I tillegg vil vi ta for oss hva som fungerte og hva som ikke fungerte av metodene og teknologiene vi benyttet.

5.1 Drøfting av vitenskapelige resultater

I dette delkapittelt blir designet av applikasjonen gjennomgått. Vi vil gjennomgå hvorfor det oppsto endring i designet. Vi vil diskutere designaspekter som ikke funket etter sin hensikt og årsaker til dette. Vi legger også frem et par tekniske aspekter teamet la til eller ønsker å legge til applikasjonen.

5.1.1 Designet

Som tidligere nevnt, er applikasjonen basert på wireframes utformet av oppdragsgiver, se vedlegg B. Dette gjorde det enkelt for teamet å følge dette mønsteret gjennom applikasjonen og gjorde sluttresultatet svært likt det som var ønsket fra oppdragsgiver. Til tross for dette var det noen avvik fra det originale designet. Et av disse var hvordan bestillingssiden er oppbygget. Oppdragsgiver hadde originalt sett for seg at brukeren selv skulle fylle ut hvor mange seksjoner bestillingen trengte. Etter at funksjonaliteten var implementert konkluderte teamet i samarbeid med oppdragsgiver at det var lite forståelig for brukeren. Det ble dermed endret slik at applikasjonen automatisk regner ut antall seksjoner en bestilling trengte.

Designet tar hensyn til svaksynte brukere som krever større tekst siden endringen av tekststørrelsen ikke vil gi tap av innhold. I tillegg har alle inputfeltene i applikasjonen navn på hva som skal fylles inn og valideringssjekker. Kontrasten mellom tekst og bakgrunn opprettholder kravene stilt av WCAG 2.0 og navigasjonen kan også utføres på ulike måter. Dette gjør at oppdragsgiver slipper å tenke på disse kravene når de skal videreutvikle applikasjonen.

De fleste brukerne som testet applikasjonen navigerte seg rundt uten hjelp. Til tross for dette var det noen deler av applikasjonen flere av brukerne stoppet opp ved, og måtte ha hjelp for å komme videre. Da de skulle legge inn en bestilling ble noen av dem usikre på om bestillingen var sendt inn. Dette så ut til å være en konsekvens av at betalingssiden ikke var funksjonell, noe som så ut til å forvirre noen av brukerne. Da de kom til betalingssiden og ikke trengte å legge inn sin egen info, stoppet det litt opp og noen gikk tilbake til bestillingssiden siden de trodde bestillingen var fullført. Observasjonene tilsier at dette problemet mest sannsynlig vil falle bort når betalingsfunksjonaliteten blir implementert.

Applikasjonen kan brukes både på iOS og Android mobiler. Applikasjonen vil ha et identisk design og funkasjonalitet, uavhengig av plattformen brukeren benytter. Det eneste som foreløpig skiller opplevelsen er notifikasjoner. Teamet fikk ikke testet notifikasjonene på iOS siden dette krevde en "Apple Developer Account" for å aktivere. Funksjonaliteten for dette skal fungere, men må aktiveres med kontoen i Xcode for å kjøre på simulatoren teamet brukte for å teste applikasjonen. Dette er dermed flyttet til videre arbeid, se kapittel 6.

En funksjonalitet teamet la til på eget initiativ, var muligheten til å fylle ut adressen basert på lokasjonen til brukeren. Dette ble gjort siden teamet så verdien av å kunne slippe å fylle ut adressen selv om du aldri hadde brukt tjenesten før. I tillegg er det en funksjonalitet flere applikasjoner tar i bruk for å gjøre ting lettere for brukeren. Selv om intensjonen var å gjøre utfylling av adresse lettere for nye brukere, viste brukertestene at knappen ikke var tydelig nok. Ingen av brukerne benyttet seg av knappen, og etter de ble spurt om de så knappen svarte samtlige nei. Teamet benyttet seg av tegnet Google maps bruker for nåværende posisjon, men dette så ikke ut til å vekke assosiasjoner hos brukerne som testet applikasjonen.

I oversikten over bestillingene til brukeren får brukeren ikke opp de kansellerte bestillingene. Dette er noe teamet ønsket vi løste på en annen måte. Slik det er nå vil bestillinger som kanselleres

fjernes fra databasen, og brukeren har ikke muligheten til å få en oversikt over dem. Om vi skulle ha løst dette på nytt ville teamet tatt vare på bestillingene som ble kansellerte og lagt dem inn i en egen liste, slik at brukeren hadde en oversikt over de også. I tillegg ville dette vært til nytte om gjenvinningsstasjonen kansellerte en bestilling av ulike grunner, slik at brukeren kunne sett grunnen til dette.

Nattmodus ble implementert da dette er noe som ofte blir etterspurt av brukere. Teamet implementerte derfor dette i starten av prosjektet ettersom dette er en endring som blir vanskeligere jo lenger ut i prosjektet man kommer. Ved å implementere dette tidlig er det enkelt å sørge for at nye funksjonaliteter ser bra ut og oppfyller WCAG også i nattmodus.

5.1.2 Tekniske aspekter

Noen tekniskere aspekter som teamet skulle ha gjort annerledes var måten vi testet funksjonaliteter. Mye tid ble brukt på integrasjonstesting som kunne vært spart ved implementering av enhets- og Widget-tester. Dette ble derimot ikke prioritert av teamet ettersom vi følte det var viktigere å være sikker på at alle elementene fungerer korrekt sammen, enn at alle de små delene hver for seg fungerte korrekt. Dette ble basert på at oppdragsgiver har vært tydelig underveis på at en del av verdiene og informasjonen som brukes kun var midlertidig. Teamet vurderte derfor at implementering av tester ville skape mer arbeid enn det sparte tid og gjorde derfor integrasjonstesting underveis.

5.2 Drøfting av Ingeniørfaglige resultater

5.2.1 De funksjonelle kravene

Teamet har oppfylt nesten alle de funksjonelle kravene som ble stilt av oppdragsgiver. Noen krav ble bortprioritert, og andre ble teamet, etter samtale med oppdragsgiver, enig om å ikke gjennomføre. Denne utvelgelsen ble basert på et estimat fra teamet der antall arbeidsdager for de forskjellige delkravene ble vurdert. Valgene over hva som skulle prioriteres ble da valgt ut ifra disse estimatene og tiden teamet hadde til rådighet. Dette er også estimatene som er utgangspunktet for fremdriftsplanen i prosjekthåndboka. Ut i fra estimatene var det tydelig at noen av de ønskede funksjonalitetene ble for tidskrevende, selv de funksjonalitetene som ble valgt viste seg å kreve mer tid enn forventet. De funksjonaliteter som ikke ble en del av denne bacheloroppgaven blir ført til videre arbeid, dette er beskrevet i kapittel 6.2.

Prioritering av funksjonelle krav

I tabell 1 ser vi hvilke funksjonelle krav og delkrav som teamet etter enighet med oppdragsgiver valgt å ikke gjennomføre. Delkrav 4.5 og 4.6 ble valgt bort ettersom disse delkravene ville krevd flere komplekse funksjoner. Disse kravene hadde blant annet krevd et kart, kunne få eller simulere posisjonen til hentebilen og vise denne posisjonen på kartet. Dette forventet teamet kom til å kreve flere arbeidsdager å få til, samtidig som ruteplanleggingen, som er nødvendig for å finne estimert hentetidspunkt, er en del av administrasjonsverktøyet, som ikke er en del av denne bacheloroppgaven, se kapittel 1.3.

Det siste kravet som tidlig ble bortprioritert var å kunne gjennomføre betaling i applikasjonen. Dette kravet ble valgt bort da teamet ikke har jobbet med behandling av transaksjoner, eller Stripe, tidligere. Grunnet dette mente teamet at delkravet ville kreve mye tid for å forsikre oss om at alt ble håndtert på et profesjonelt og sikkert vis. Etter samtale med oppdragsgiver ble teamet derfor enige om å la denne delen gå over til videre arbeid, og heller implementere en "placeholder" betalingsside slik at det skal være så enkelt som mulig for oppdragsgiver å legge til denne funksjonlaiteten i ettertid.

Avvik fra funksjonelle krav

Blant kravene som var planlagt å gjennomføre i løpet av prosjektet, men som ikke ble gjort, er to av dem tett koblet sammen. Delkrav 2.3 "Brukeren kan legge inn navn" og delkrav 10.1 "kunne endre navn på brukeren". Etter en samtale med oppdragsgiver kom vi frem til at det ikke hadde

noen gevinst for brukeren og skrive inn navnet sitt, og heller ikke for systemet da navnet ikke blir brukt. Dette gjorde at disse kravene ble priotert bort, og ikke ble gjennomført. Oppdragsgiver var bekymret for at det ville være problemer med å få identifisert brukere ved behov. Her ble teamet og oppdragsgiver enige om siden at bruker var blitt autentisert via mobilnummeret, ville det være mulig å identifisere brukere ved hjelp av dette.

Andre krav som ikke ble gjennomført er delkrav 8.2, her var det var tenkt at brukeren ville bli spurt om å gi tilbakemelding om applikasjonen etter en bestilling var lagt inn. Dette konkluderte teamet med at kunne bli slitsomt i lengden, og valgte derfor å legge inn en egen siden for tilbakemeldinger. Der kan brukeren selv velge når de vil gi tilbakemeldinger, og dette kan skje uavhengig om en bruker legger inn en bestilling.

Prioritering av funksjonelle ønsker

Når det gjelder de funksjonelle ønskene som kom underveis fra oppdragsgiver, se tabell 2, ble noen av dem gjennomført mens andre falt bort. En av ønskene som ble gjennomført var at avfallgjenstander skulle tilhøre ulike kategorier. Dette ble prioritert ettersom dette vil gjøre det lettere å håndtere og resirkulere avfallet korrekt, samtidig som det hjelper brukeren å finne ut hva tjenesten kan hente og ikke. Ved å gi brukeren alternativer kan det hende brukeren finner ut at de har mer avfall enn det de trodde de kunne legge ved i bestillingen, som vil gagne alle. Disse kategoriene vises i figur 15b.

Videre ble også "Beregne antall seksjoner utifra avfallet" prioritert da den første løsningen, hvor brukeren selv måtte finne ut av hva en seksjon er og hvor mange de trenger, ikke var lett for brukeren å forstå, slik som nevnt i delkapittel 5.1.1.

Det siste ønsket teamet prioriterte var å implementer en tidslinje for progresjonen av statusen til bestillingen. Dette ble prioritert fordi det gir brukeren viktig informasjon om hvilke steg som står igjen før bestillingen er fullført. Denne tidslinjen er vist i figur 19.

De funksjonelle ønskene temaet ikke hadde muligheten til å implementere er blant annet å ha en historikk for hver bestilling. Her var det tenkt at hver bestilling tar vare på og tidfester alle endringer for bestillingen som f.eks. ved endring av bestillingen, tidspunkt når hentebil er på vei, tidspunkt hentebilen plukket opp avfallet osv. Dette var for omfattende til at teamet hadde tid til å gjennomføre.

Det andre ønske teamet ikke fikk gjennomført, var å ha statuser for om bestillingen blir kansellert av bruker eller avbrutt av hentebilen osv. Dette følte teamet var så tett koblet opp mot administrasjonsverktøyet, at dette sannsynligivs kom til å bli endret etter prosjektets slutt. Teamet valgte derfor også å ikke gjennomføre dette ønske heller, men har plassert det under videre arbeid i kapittel 6.2.

5.2.2 Avvik fra de ikke-funksjonelle kravene

De ikke funksjonelle kravene omhandler selve oppbyggingen av løsningen, disse kravene styrer altså i stor grad hvordan den endelige løsningen ble. Teamet holdt seg tro til nesten alle kravene, de to kravene som ikke ble gjennomført var "Bruk av Googles Firebase Cloud Functions som server", og "Bruk av stripe som betalingsmåte". Som nevnt tidligere ble implementeringen av Stripe flyttet til videre arbeid. Bruken av Stripe vil ikke påvirke funksjonalitet til applikasjonen direkte, da den eneste konsekvensen av dette er at det må implementeres senere.

Da det kom til bruk av FCF som server, ble dette valgt bort av to grunner. Den første og kanskje viktigste grunnen er hvordan FCF ikke har samme mulighet til å ha en kontinuerlig tilkobling til databasen slik som løsningen har nå. Dette er fordi FCF har en begrensing på at hver funksjon maksimalt kan kjøre i 540 sekunder. Ved å bruke FCF som server hadde altså applikasjonen mistet muligheten til å oppdateres samtidig som endringer skjer i Firestore. Da ville brukeren ikke lenger fått oppdateringene som skjer i databasen momentant, som f.eks. oppdatering av status. Dette er noe som kommer til å bli viktig senere dersom applikasjonen skal motta og bruke posissjoen til hentebilen av samme grunn. Derfor har applikasjonen en direkte tilkobling til Firestore istedenfor. FCF blir derimot brukt for å sende push-notifikasjoner. Ved å ha en direkte kobling til Firestore i

applikasjonen er det derimot en større fare for at brukere kan finne svakheter og potensielt hente ut/endre informasjon de ikke har tilgang til. Dette burde forhindres av Firestore sine sikkerhetsregler (se vedlegg C for nærmere forklaring), men sannsynligheten er noe større enn hvis denne koden lå på en server, eller via FCF som oppdragsgiver ønsket.

Dersom det er ønskelig er det mulig å bruke FCF som en server ved å bruke den samme funksjonaliteten som push-notifikasjoner, hvor ved hver endring eller opprettelse av et dokument blir det sendt til brukeren med dataene. Dette blir en mer kompleks løsning da lettvintheten ved å bruke koblinger direkte fra Firestore gjør håndteringen av autentisering sømløst. Denne økte bruken av funksjoner vil også øke kostnaden til prosjektet, da det er betydelig dyrere å bruke FCF i tillegg til Firestore istedenfor kun Firestore. Dette er altså et alternativ som kan vurderes i videre arbeid.

5.2.3 Brukertestene

Under diskuteres resultatene av brukertestene som er gjennomført, og teorier om brukerdemografen til applikasjonen. Siden teamet ikke får rullet ut applikasjonen på en større skala, vil vi heller ikke få testet disse teoriene på en større del av brukerdemografen. Disse teoriene er altså basert på observasjonene gjort under brukertestene.

Fra resultatdelen av rapporten ser vi at av de fem brukerne som testet applikasjonen ville fire benyttet seg av den om de hadde muligheten. Den brukeren som svarte nei, nevnte at de heller ville ha kjørt avfallet personlig om de hadde midlene. Om midlene ikke hadde vært tilgjengelig mener derimot brukeren at de ville benyttet seg av tjenesten, om det ikke var for dyrt. Brukeren ser ikke ut til å opplevd noe problemer med køer ved sin lokale gjenvinningstasjon og har tidligere hatt tilgang på midlene som trengs for å frakte avfall. Brukeren ser derfor ikke ut til å ha et stort behov for tjenesten.

En annen bruker, som stiller seg positiv til å ta i bruk applikasjonen, nevner at de tidligere har opplevd lange køer ved sitt lokale gjenvinningsanlegg. Brukeren sier at de mye heller vil benytte seg av applikasjonen, enn å bruke en stor del av dagen sin i kø. De nevner at det vil gi dem mer fritid til å gjøre andre ting enn å levere avfall.

5.3 Drøfting av Administrative resultater

I dette delkapittelet vil vi ta for oss grunnene til at det oppsto eventuelle avvik fra de originale planene som ble fastsatt i starten av prosjektet.

5.3.1 Fremdriftsplanen

I starten av prosjektet utarbeidet teamet en overordnet fremdriftsplan basert på forventningen av arbeidsmengden til de forskjelle delene av prosjektet. Underveis i prosjektet viste det seg at enkelte oppgaver krevde mer tid enn forventet, mens andre oppgaver tok kortere tid enn forventet. Vi tar for oss de ulike fasene og deres varighet, som listet opp i tabell 8, og grunnen til avvik i disse fasene.

Forarbeid

Det var tre aktiviteter som inngikk i forarbeidet til prosjektet. De ulike aktivitetene var følgende:

- Lesing av relevant teori og metode
- Innsetting i maler
- Innsetting i oppgaven

Selve forarbeidsfasen hadde en planlagt varighet på fire arbeidsdager, dette endte med å bli seks arbeidsdager. Dette grunnet at oppgaven og malene var mer omfattenende enn først antatt.

Oppstartsfasen

Oppstartsfasen tok for seg utarbeiding av dokumentene som definerte rammene til prosjektet. Disse dokumentene var visjonsdokumentet, kravsdokumentet og oppsett av prosjekthåndboka. I tillegg var det andre aktiviteter som oppstartsmøte og lesing av teori på relevante temaer. Her tok samtlige dokumenter mer tid enn først planlagt. Dette kom av at teamet ikke hadde tatt i betrakning at veileder skulle komme med tilbakemeldinger, og at endringene skulle gjøres etter gitte tilbakemeldinger.

Sprint 1

Sprintvarigheten gikk som planlagt over 14 arbeidsdager. Sprintmålene som skulle nås i løpet av sprinten hadde varierende varighet, der noen tok lengre tid enn forventet. Funksjonaliteten for å legge inn bestilling, autentisering og oppretting av bruker tok lengre tid. Dette siden teamet ikke hadde noe tidligere erfaring med Firebase og de funksjonalitetene som skulle tas i bruk fra Firebase.

Sprint 2

I likhet med sprint 1 gikk varigheten av sprint 2 etter planen. Det er sprintmålene i perioden som hadde noe avvik fra planlagt varighet. Der noen funksjonaliteter utgikk, og beregning av pris i ordren gikk på overtid. Den ene funksjonaliteten som ble kuttet var muligheten til å endre profilinformasjon. Dette var ikke lenger nødvendig basert på hvordan teamet hadde implementerte brukere. Det ble dermed flyttet til videre arbeid, om oppdragsgiver ombestemmer seg i fremtiden. Å implementere prisen i orderen tok lenger tid enn forventet, fordi teamet ikke hadde tatt hensyn til hvor mye av en bestillingsoppsettet som måtte endres.

Sprint 3

Sprint 3 ble kortere enn først antatt. Varigheten gikk fra 14 dager til 8 dager. Dette skylles at vi innså at dokumentasjonen måtte prioriteres siden prosjektet nærmet seg slutten. Det som ble fokusert på var bugfiksing av de mest alvorlige bugsene teamet fant. Etter at dette var gjort begynte begge for fullt å jobbe med hovedrapporten.

Avslutningsfasen

Avslutningsfasen var det kun ferdigstilling av hovedrapporten som var målet. Siden sprint 3 ble kortere enn først antatt ble avslutningsfasen lengre. Teamet forventet at mer av hovedrapporten skulle utarbeides underveis i prosjektet, men dette viste seg å være vanskeligere enn først antatt.

Hovedrapporten

I denne fasen ble det mest avvik fra planen som var satt i begynnelsen av prosjektet. Teamet slet med å løsrive seg fra utviklingen av applikasjonen til fordel for å skrive dokumentasjon under sprintene. Dette gjorde at store deler av rapporten ble utarbeidet i sluttfasen av prosjektet, til tross for at temaet hadde satt opp datoer for jobbing jevnt ut under prosjektet.

5.3.2 Timeregnskapet

I starten av prosjektet var det kun tre dager i uken satt av til jobbing med bacheloren. Dette skyltes at vi hadde et annet fag parallelt med oppgaven helt frem til mars. Dette kan vi se ut fra arbeidstimene i figur 32. Vi ser at teamet ikke jobbet like mye med bacheloren i uke 7 og 11. Uke 7 ble det gjennomført en obligatorisk innovasjonscamp som tok store deler av denne uken. I uke 11 var det eksamen i det andre faget som gjorde at prosjektet ble nedprioritert. Sivert hadde også en eksamen i uke 10. Under noen perioder av prosjektet var teamet upresise under føring av arbeidstimer og oppgaver. I ettertid var det derfor noen problemer med å få loggført alt av oppgaver nøyaktig. I retrospekt burde teamet ha notert oftere i løpet av arbeidsdagen enn hva de gjorde, slik at det hadde vært lettere å fylle inn timeregnskapet. Når det er sagt har teamet jobbet godt underveis i hele prosjektet og har fulgt de planlagte arbeidsdagene.

5.3.3 Utviklingsprosessen

Om vi skulle ha valgt utviklingsprosses på nytt ville vi kjørt samme taktikk. Å organisere utviklingen i sprinter gjorde det enkelt for teamet å fokusere på de prioriterte funksjonalitetene i den gitte perioden. Det ble en naturlig overgang, der oppdragsgiver kunne involveres uten noe hinder for utviklingen. I sprintene var oppsett av sprintmål basert på de funksjonelle kravene en lettvidt måte å sørge for at kravene ble møtt.

En ting som viste seg å ikke funke fullt så bra, var å legge dokumentasjonsarbeid parallelt med sprintene. Teamet fant det vanskelig å legge fra seg appliksjonen under en sprint for å jobbe med dokumentasjon, noe som gjorde at store deler av dokumentasjonen ble gjort på slutten av prosjektet. Neste gang vil vi legge inn dager utenom sprintene for å jobbe med dokumentasjon, slik at det er lettere å fokusere kun på dette.

Bruk av digital kanban-tavle i Nortion, slik nevnt i kapittel 3.2, var en praktisk måte å organisere arbeidsflyten på. Det var lett å ha oversikt over hva den andre parten gjorde til en hver tid, og hvor mye som gjensto for at et sprintmål var ferdig. Dette er noe teamet vil anbefale fremtidige teams som samarbeider på et prosjekt der kanban-tavler er nødvendig. Kort oppsummert var oppdeling av arbeidet i sprinter noe teamet ville gjort igjen, men med lengre mellomrom mellom sprintene for å ha mer dedikert tid til dokumentasjon.

5.4 Gruppearbeidet

I oppstartsfasen av prosjektet var det tett samarbeid for å utarbeide de ulike dokumentene for kravene og funksjonaliteten oppdragsgiver ønsket i applikasjonen. Etter selve utviklingen startet, ble oppgavene til hvert teammedlem delt basert på de ulike sprintmålene, der jobbingen foregikk selvstendig. Senere i prosjektet, mot slutten av sprint 2, ble ansvarsområdene mer delt. Sabine gikk over til å jobbe mer med dokumentasjonen og hovedrapporten, mens Sivert fortsatte utviklingen av applikasjonen. Dette førte til at begge parter ble mer separert fra hva den andre jobbet med, noe som gjorde det tungvint å sette oss inn i hva den andre hadde jobbet med.

Delingen av ansvarsområde skjedde naturlig, da mye av funksjonaliteten Sivert jobbet med skulle flettes inn i applikasjonen og det var lettere om han var den som gjorde det. I motsetning til Sabine som måtte først ha satt seg inn i endringene for å deretter flette det sammen. Det var altså en beslutning tatt basert på hva som ville tatt kortest tid. Grunnet statusen på hovedrapporten i samme periode, var det naturlig at Sabine gikk over på utarbeiding av denne.

5.4.1 Samarbeidet

Innad i teamet har det vært et godt samarbeid hele veien. Om problemer oppsto var det alltid lett å få hjelp. Teamet møttes fysisk hver arbeidsdag for å samarbeide, noe som gjorde kommunikasjonen lettere enn via nettet, og gjorde det enkelt å be om hjelp eller tilbakemeldinger.

Kommunikasjonen mellom teamet og oppdragsgiver var også tett da teamet utarbeidet applikasjonen. Før hver av sprintene ble det holdt møter for å informere om hva teamet ble å fokusere på i de ulike sprintene, og hva de kunne forvente etter sprintens ende. Det ble også holdt en kort fremvisning etter sprintene for å demonstrere hva som var blitt gjort, slik som nevnt i kapittel 3.2. Den tette kommunikasjonen gjorde at teamet fikk tilbakemeldinger underveis, og eventuelle endringer i planer ble gjennomgått. Oppdragsgiveren stilte seg tilgjengelig for spørsmålene teamet hadde, og brukte ikke lang tid på å svare. De ga oss også albuerom til å komme med forslag til endringer i applikasjonen, noe vi satte stor pris på.

Under prosjektet hadde teamet møte med veileder ca. hver andre uke. Her ble oppdragsgiver oppdatert på hva teamet hadde gjort siden sist møte, og teamet fikk svar på eventuelle spørsmål. I tillegg til møtene stilte veileder seg tilgjengelig for tilbakemeldinger av dokumentasjon som ble gjort. Dette var til stor hjelp for teamet da hovedrapporten begynte å ta form. Til tross for at all kommunikasjonen ble gjort over Microsoft Teams, er teamet fornøyd med kommunikasjonen

mellom både oppdragsgiver og veileder.

5.5 Etiske konsekvenser

I denne delen av rapporten ser vi på hvordan etiske konsekvenser applikasjonen kan ha. Først vil vi se på miljøkonsekveser deretter personvern og tilslutt samfunnsmessige konsekvenser.

5.5.1 Miljøkonsekvenser

De konkrete miljøkonsekvensene applikasjonen vil ha er vanskelig å si, siden vi ikke får gitt ut applikasjonen til markedet og observert virkningene den vil ha. Dermed blir mye av det vi diskuterer spekulasjoner, der brukertestsvarene er mye av det vi kan støtte oss opp med.

Applikasjonen vil gi et nytt alternativ for hvordan innbyggere i en kommune leverer avfall. Jo flere benytter seg av tjenesten, jo færre vil ha et behov for å kjøre selv til gjenvinningsstasjonen. Dette vil bety færre biler og mindre kø. I tillegg vil denne trafikken bli erstattet med gjenvinningsstasjonens egne biler som rommer mer enn en vanlig bil. Dette kan tyde på at det vil være færre biler i omløp i løpet av en dag, dette betyr også mindre utslipp enn tidligere.

Et annet aspekt som kan endre seg, er at innbyggere som tidligere ikke har hatt muligheten til å kjøre avfall til gjenvinning har nå et alternativ de kan benytte. De kan dermed sende mer til gjenvinning, noe Norge ønsker, slik at de kan nå sine gjenvinningsmål.

En digitalisering av tjenester kan komme med negative miljøaspekter om energien som brukes for å drive den digitale løsningen ikke er fornybar. Datasentre som kreves for å lagre og drive tjenester vil dermed ha en innvirkning på miljøet. I 2018 bidro datasentre til 2% av de totale globale utslippene av klimagasser (Supermicro, 2018). En økning i mengden data som produseres vil gi et større behov for dataprosessering og -lagring. Ifølge en analyse gjort av NVE, kan Norge forvente bebyggelse av flere datasentre i landet. En økt etterspørsel om data vil derfor påvirke Norge (Hole & Horne, 2019).

5.5.2 Personvern

En stor mangel i applikasjonen er fraværelsen av muligheten til å slette en bruker. Dette er noe som må integreres før applikasjonen lanseres for å opprettholde retten til personvern og GDPR. Applikasjonen lagrer kun brukerens mobilnummer, men dette faller under kategorien personopplysninger og burde derfor være mulig å få brukeren fjernet om ønskelig. Brukere har ikke muligheten til å lete opp andre brukere, dermed er opplysningene som er lagret om brukeren, bare noe brukeren selv og de som administrer applikasjonens database som har tilgang til.

5.5.3 Samfunnsmessige konsekvenser

Løsningen er tilpasset privatpersoner og vil øke tilbudet for innbyggere i kommuner som tar i bruk løsningen. Dette vil gjøre at innbyggere har flere valgmuligheter enn tidligere, og kan selv velge om de vil benytte tjenesten. Om en stor mengde av de som leverer avfallet selv benytter seg av tjenesten vil dette bety færre biler i kø. Det vil altså være en mulighet for å få kuttet ned på køene som innbyggerne opplever, noe som igjen kan gangne de som velger å fortsette å levere avfallet selv. Tjenesten åpner også for nye arbeidsplasser for de som må kjøre hentebilene og administrere tjenesten. Dersom tjenesten lykkes og køene faktisk blir kortere vil dette også fjerne behovet for politi til å dirigere trafikken, som frigjør politiet til å fokusere på nye problemområder i samfunnet.

6 Konklusjon og videre arbeid

6.1 Konklusjon

Etter utviklingen av mobilapplikasjonen står vi igjen med et produkt som i stor grad oppfyller de ønskene oppdragsgiver kom med. Løsningene i applikasjonen er lagt opp på en måte som gjør tillegg og eventuelle endringer lettest mulig å implementere. Applikasjonen kan benyttes av både iOS og Android brukere, og danner en alternativ løsning for innbyggere for å levere avfall. Siden applikasjonen er utviklet for både iOS og Android enheter vil den kunne benyttes av majoriteten av innbyggere.

Observasjoner og intervjudata fra brukertestene indikerer at mobilapplikasjonen er et alternativ som innbyggere ville ha benyttet seg av. Om innbyggere ikke har tilgang på midlene for å frakte avfallet selv vil de kunne benytte seg av applikasjonen. Tjenesten er også aktuell for innbyggere som heller vil betale for tjenesten enn å bruke tid på å levere avfallet selv.

De som ikke benytter seg av tjenesten vil kunne merke effekten av den i form av kortere køer og mindre venting ved gjenvinningsstasjonen. Dette gjør at tjenesten også har en positiv effekt for de som ikke bruker den. Tjenesten vil dermed gjøre prosessen med å levere avfall mer behagelig for alle, og kan dermed motivere innbyggerne til å gjenvinne oftere. Dette kan igjen hjelpe Norge med sitt mål om å øke andelen husholdningsavfall som blir materialgjenvunnet.

Problemstillingen som ble presentert i delkapittelet 1.2 var **"Kan en digitalisert løsning for avfallslevering, i form av en mobilapplikasjon, gjøre gjenvinning mer tilgjengelig for befolkningen"**. Etter utviklingen og observasjonene gjort i forbindelse med denne bacheloroppgaven vil teamet konkludere med at mobilapplikasjonen vil gjøre gjenvinning mer tilgjengelig for befolkningen. Applikasjonen vil utvide tilbudet i kommunene som tar det i bruk, og gjøre at flere av innbyggerne har muligheten til å gjenvinne.

6.2 Videre arbeid

Gjennom hele prosjektet har oppdragssgiver vært tydelig på at applikasjonen kommer til å videreutvikles etter prosjektets slutt. Dette har gjort at teamet ofte har lagt til rette for enkel utviding og endring i applikasjonen. Tabell 9 lister opp alle punkter som ikke har blitt gjennomført i prosjektet, forslag og anbefalinger fra teamet om hva som kan/burde være en del av den ferdige applikasjonen, og forslag til endringer.

Hva	**Beskrivelse**
Kunne slette brukere	En bruker bør kunne slette alle personopplysningene sine i tråd med GDPR.
Push notifikasjoner på iOS	Gi tilgang til å motta push-notifikasjoner på iOS.
Informasjonssiden	Informasjonen på informasjonssiden må fylles ut for å hjelpe brukerne og gi svar på vanlige spørsmål en bruker kan ha.
Betaling med Stripe	Gjennomføring av betaling med Stripe.
Godkjenning av vilkår	Før brukeren logger inn må brukeren godta vilkårene for applikasjonen.
Legge til flere språk	La brukeren velge mellom forskjellige språk.
Estimert hentetidspunkt	Gi brukeren et estimert hentetidspunkt for en bestilling.
Vise hentebil på kart	Vise hvor hentebilen befinner seg på et kart når bilen er på vei for å hente avfallet.
Profilinformasjon	La brukeren skrive inn mer profilinformasjon som f.eks navn og e-post for å sende kvitteringer.
Hentebil kan avbryte en bestilling	Dersom en bestilling er for stor eller noe annet er galt med bestillingen bør hentebilen kunne avbryte bestillingen på en slik måte at brukeren får melding.
Historikk for hver bestilling	En historikk over alle endringer i bestillingen, f.eks. tidspunkt bestilingen ble bekreftet, tidspunkt og endring dersom bruker endrer en bestilling, tidspunkt hentebilen plukket opp avfallet osv.
Dato- og tidsvelger kan kun velge aktuelle hentetidspunkt	Ikke la brukerne velge tidspunkt som hentebilen ikke har mulighet til å hente, f.eks. kun hentetidspunkt på hverdager mellom 09:00 og 17:00.
Bruke Firebase Cloud Functions som server	Som nevnt tidligere er det mulig å bruke FCF som en server istendefor å koble appen direkte til Firestore.
Kansellering endrer status	Ved kansellering slettes ikke bestillingen fra databasen, men den endrer status slik at administrasjonsverktøyet fortsatt har oversikt.

Tabell 9: Videre arbeid for applikasjonen

Referanser

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... others (2001). *Manifesto for agile software development.* Hentet fra http://agilemanifesto.org/ (Hentet: 24.04.21)

Bhatia, A. (2018). *One time password (otp) algorithm in cryptography.* Hentet fra https://www.geeksforgeeks.org/one-time-password-otp-algorithm-in-cryptography/ (Hentet: 05.05.21)

Brennen, J.S. & Kreiss, D. (2016). Digitalization. *The international encyclopedia of communication theory and philosophy*, 1–11.

dataforening Høstakademiet, D.N. (1995). *Seminar om virksomhetsmodellering : torsdag 7. desember 1995 håndverkeren møtesenter, oslo.* Oslo: [Den norske dataforening].

devs, F. (u.d.). *Flutter architectural overview.* Hentet fra https://flutter.dev/docs/resources/architectural-overview (Hentet: 29.04.21)

Digitaliseringsdirektoratet. (2021). *Wcag 2.0-standarden.* Hentet fra www.uutilsynet.no/wcag-standarden/wcag-20-standarden/86 (Hentet: 09.04.21)

Dyba, T. & Dingsoyr, T. (2009). What do we know about agile software development? *IEEE software*, *26*(5), 6–9.

FHI. (2018). *Generelt om avfall.* Hentet fra www.fhi.no/ml/avfall-og-soppel/handtering-helseeffekter/generelt-om-avfall/ (Hentet: 24.04.21)

FHI. (2018). *Hvem gjør hva?* Hentet fra www.fhi.no/ml/avfall-og-soppel/info-kommune-og-naring/ansvarsfordeling-mellom-kommuner-og-naringsliv/ (Hentet: 09.02.21)

for Kildesortering og Gjenvinning., L.S. (2018a). *Avfall.* Hentet fra snl.no/avfall (Hentet: 24.04.21)

for Kildesortering og Gjenvinning., L.S. (2018b). *Avfallshierarki.* Hentet fra snl.no/avfallshierarki (Hentet: 24.04.21)

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (u.d.). *Design patterns : elements of reusable object-oriented software.* Reading, Mass: Addison-Wesley.

Gibson, D. (2011). Understanding the three factors of authentication. *Online] http://www. pearsonitcertification. com/articles/article. aspx*.

hentavfall.no. (u.d.). *hentavfall.no.* Hentet fra https://www.hentavfall.no/ (Hentet: 06.05.21)

Highsmith, J. & Cockburn, A. (2001). Agile software development: the business of innovation. *Computer*, *34*(9), 120-127. doi: 10.1109/2.947100

Hole, J. & Horne, H. (2019). *Energibruk fra datasentre i norge.* Hentet fra https://publikasjoner.nve.no/faktaark/2019/faktaark2019_13.pdf (Hentet 14.05.21)

Kidecha, S. (2020). *Native vs. hybrid vs. cross-platform: How and what to choose?* Hentet fra dzone.com/articles/native-vs-hybrid-vs-cross-platform-how-and-what-to (Hentet: 27.04.21)

Miljødirektoratet. (2019). *Avfallsplan 2020-2025.* Hentet fra www.regjeringen.no/contentassets/c6a9a384d90c4af18bfd8458f3167708/avfallsplan-2020-2025.pdf (Hentet: 09.02.21)

Miljødirektoratet. (2020). *Sirkulær økonomi.* Hentet fra www.miljodirektoratet.no/ansvarsomrader/avfall/sirkular-okonomi/ (Hentet: 09.02.21)

Miljødirektoratet. (2021). *Avfall.* Hentet fra miljodirektoratet.no/ansvarsomrader/avfall/ (Hentet: 09.02.21)

Okta. (u.d.). *Authentication vs. authorization.* Hentet fra https://www.okta.com/identity-101/authentication-vs-authorization/ (Hentet: 05.05.21)

Pcmag. (u.d.). *Hybrid mobile app.* Hentet fra www.pcmag.com/encyclopedia/term/hybrid-mobile-app (Hentet: 27.04.21)

Plikk, N. (2019). *Nattmodus eller ikke? batteritest viste enorm forskjell.* Hentet fra https://www.tek.no/nyheter/nyhet/i/dOzXVO/nattmodus-eller-ikke-batteritest-viste-enorm-forskjell (Hentet: 11.05.21)

RagnSells. (u.d.). *Grønn bil fra ragnsells.* Hentet fra https://butikk.ragnsells.no/gronn-bil (Hentet: 06.05.21)

Regjeringen.no. (2014). *Digitalisering i offentlig sektor.* Hentet fra https://www.regjeringen.no/no/tema/statlig-forvaltning/ikt-politikk/digitaliseringen-i-offentlig-sektor/id2340245/ (Hentet: 09.04.21)

Reppen Kvikstad, H. (2020). *Tirsdag morgen er det igjen stor kø ved flere av landets gjenvinningstasjoner.* Hentet fra www.nettavisen.no/nyheter/politiet-anmoder-de-som-ikke-ma-levere-avfall-om-a-vente-til-over-paske/s/12-95-3423950447 (Hentet: 09.02.21)

Rosin, A.F., Proksch, D., Stubner, S. & Pinkwart, A. (2020). Digital new ventures: Assessing the benefits of digitalization in entrepreneurship. *Journal of Small Business Strategy*, *30*(2), 59–71.

Sharma, S. (2020). *Native vs hybrid vs cross-platform — what to choose?* Hentet fra medium.com/flutterdevs/native-vs-hybrid-vs-cross-platform-what-to-choose-3221130f7cc5 (Hentet: 27.04.21)

Slåen Sæther, M.. (2020). *Veksten i avfallsmengdene flater ut.* Hentet fra www.ssb.no/natur-og-miljo/artikler-og-publikasjoner/veksten-i-avfallsmengdene-flater-ut (Hentet: 09.02.21)

soppelsekk.no. (u.d.). *soppelsekk.no - få unna søppelet!* Hentet fra https://www.soppelsekk.no/ (Hentet: 17.05.21)

Sortere.no. (2021). *Avfalls- og gjenvinningsbransjen i norge.* Hentet fra www.sortere.no/avfallsbransjen (Hentet: 24.04.21)

SSB. (2021a). *Avfallsregnskap.* Hentet fra ssb.no/avfregno (Hentet: 24.04.21)

SSB. (2021b). *Bruttonasjonalprodukt.* Hentet fra www.ssb.no/nasjonalregnskap-og-konjunkturer/faktaside (24.04.21)

Supermicro. (2018). *Data centers and the environment.* Hentet fra https://www.supermicro.com/wekeepitgreen/Data_Centers_and_the_Environment_Dec2018_Final.pdf (Hentet 14.05.21)

SøppelGutta. (u.d.). *Søppelgutta kaster søpla di.* Hentet fra https://www.soppel-gutta.no/ (Hentet: 06.05.21)

Tsitoara, M. (2020). *Beginning git and github.* Springer.

Wikipedia. (2019). *Model–view–controller.* Hentet fra no.wikipedia.org/wiki/Model-view-controller (Hentet: 27.04.21)

Vedlegg A Visjonsdokument

,

Prosjekt nr 078

Wasteflow Visjonsdokument

Versjon 1.1

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
20.01.2021	0.1	Første utkast	Sabine Seljeseth
25.01.2021	0.2	Utfylling og omformulering	Sabine Seljeseth og Sivert Utne
26.01.2021	0.3	Endringer i Kapittel 4	Sabine Seljeseth og Sivert Utne
27.01.2021	0.4	Endring av kapittel 3.4 og 5. Feilrettinger og omformuleringer	Sabine Seljeseth og Sivert Utne
28.01.2021	0.5	Ferdigstilling av første utkast	Sabine Seljeseth og Sivert Utne
01.02.2021	0.6	Endring av prioriteringene på brukerens behov	Sabine Seljeseth og Sivert Utne
03.02.2021	1.0	Endringer i hendhold til tilbakemeldinger gitt i møte med veileder	Sabine Seljeseth og Sivert Utne
17.05.2021	1.1	Korrigering og formatering	Sabine Seljeseth og Sivert Utne

Innhold

1 Innledning

Hensikten med dette dokumentet er å tydeliggjøre visjonen for applikasjonen som er utviklet i forbindelse med bacheloroppgaven i faget TDAT3001 Bacheloroppgave. Dokumentet tar for seg problemet applikasjonen vil være en del av å løse, og hvordan en slik applikasjon vil være med på å forbedre tilbudet om gjenvinning til forbrukerne. Vi tar for oss gevinsten Wasteflow vil ha for brukere og interessenter, og hvilke krav som må oppfylles for at applikasjonen skal kunne anses som ferdig. Vedlagt er et oppstartsdokument fra oppdragsgiver med ønsker og beskrivelser, se vedlegg A. Dokumentet er brukt som et utgangspunkt for kravene til applikasjonen.

2 Sammendrag problem og produkt

2.1 Problemsammendrag

Problem med	Svært høy pågang hos gjenvinningsstasjoner i helgene i Tromsø, og det er kun de som har tilgang på transport som har muligheten til å levere avfall til gjenvinningsstasjonene.
berører	innbyggere som ønsker å gjenvinne.
som resultatet av dette	skapes det store køer i helgene som må dirigeres av politiet og øker terskelen for gjenvinning blant innbyggerne i kommunen.
en vellykket løsning vil	gi et alternativ til innbyggerne slik at flere har muligheten til å levere avfall, og samtidig gjøre køene kortere.

2.2 Produktsammendrag

For	innbyggere i kommuner
som	ønsker å gjenvinne
produktet Wasteflow	er en mobilapplikasjon for avfallshenting
som	tilbyr en ny henteløsning for innbyggere
I motsetning til	der innbyggerne selv må kjøre avfallet til gjenvinningsstasjonen og vente i kø
Har vårt produkt	et tilbud til innbyggerne der avfallet hentes hos dem og fraktes til gjenvinningsstasjoner mot betaling

3 Overordnet beskrivelse av interessenter og brukere

3.1 Oppsummering interessenter

Navn	Utdypende beskrivelse	Rolle under utviklingen
Sluttbruker	Bruker av produktet	Vil gjennomføre brukertester og bistå med innspill
Prosjektgruppe/teamet	studentene som skriver bacheloroppgaven	Skal utvikle produktet og bli vurdert for arbeidet som er utført
Veileder	foreleser ved NTNU	Bistå prosjektgruppen når det er nødvendig, komme med innspill og sensur av produktet med rapport.
Oppdragsgiver	Favn Software AS	Vil overta sluttproduktet for videreutvikling og komme med tilbakemeldinger og ønsker underveis i prosjektet.
Produktadministrator	Remiks Tromsø	Vil få overlevert å drifte det endelige systemet produktet er en del av

3.2 Oppsummering brukere

Navn	Utdypende beskrivelse	Rolle under utviklingen	Representert av
Privatpersoner i kommuner med gjenvinningsstasjon	Vil benytte seg av sluttproduktet	Brukertesting	Brukere
Favn Software AS	Tar over produktet etter bacheloroppgaven er levert	produkteier og bistår med svar på evt. spørsmål	Sveinung Øverland og Anders Hallem Iversen
Gjenvinningsstasjon	bekrefte bestillinger og oppdatere status av henting	Vil bruke produktet etter det er ferdigstilt	Remiks Tromsø

3.3 Brukermiljø

Brukermiljøet for Wasteflow er mobiltelefoner, der tanken er at brukere laster ned applikasjonen før bruk. Omfanget til denne bacheloroppgaven tar ikke for seg brukermiljøet for de som skal administrere systemet, bare sluttbrukeren sitt.

Sluttbrukeren vil kunne benytte applikasjonen i kommuner Wasteflow opererer i. De vil kunne legge inn bestillinger av husholdningsavfall for henting innenfor de gitte tidsrammene til gjenvinningsstasjonen. Avfallet må klargjøres før henting ved å sette det tilgjengelig for hentebilen. Brukeren må være tilkoblet internett for å benytte tjenesten.

3.4 Sammendrag av brukerens behov

Behov	**Prioritet**	**Dagens løsning**	**Foreslått løsning**
Kunne fylle inn og lagre kontakt-informasjon	Høy	Ingen	Kunne opprette personlig bruker første gang appen åpnes, bruker mobilnummer for autentisering.
Legge inn bestilling av avfall	Høy	Ingen	Kunne legge inn bestilling med: • Type avfall • Antall seksjoner i bilen • Adresse for henting • Valg av tidsintervall avfallet kan hentes i
Se status for bestilling	Høy	Ingen	Se informasjon om: • Bestillingen er mottatt • Informasjon om bestillingen • Om hentebilen er på vei • Estimatert tid for henting • Om bestillingen er fullført
Endre bestilling	Middels	Ingen	Bruker kan endre bestilling fra statusoversikten frem til en tid før hentetidspunkt.
Avbestille henting	Høy	Ingen	Bruker kan avbestille bestilling fra statusoversikten frem til en tid før hentetidspunkt.
Få notifikasjoner når bestilling endrer seg	Middels/Høy	Ingen	Få varsel på mobiltelefonen dersom bestillingen blir endret.
Få informasjon om pris	Middels	Info om avfallshenting finnes på Remiks sine nettsider	Bruker kan lese om hvordan prising blir satt. Under en bestilling ser bruker hvor mye bestillingen vil koste.
Redigere profil	Middels	Ingen	Kan endre eller slette informasjon fra profilen.

Kunne gi tilbakemeldinger	Høy	Sende mail til Remiks Tromsø med evt. tilbakemeldinger	Bruker får mulighet til å gi tilbakemelding etter en fullført bestilling.
Betale for bestilling	Lav	Betaling gjennomføres på gjenvinningsstasjon etter levert avfall.	utføre betaling via *Stripe i applikasjon.*

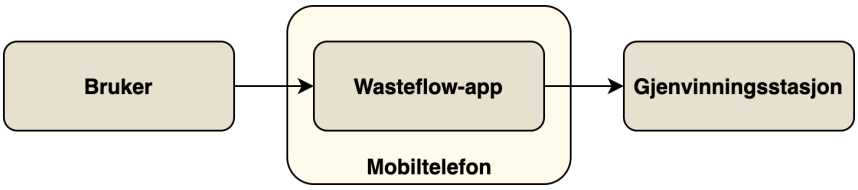
3.5 Alternativer til vårt produkt

Kommunen som er involvert i pilotprosjektet til Favn Software AS, Tromsø, har per dags ingen henteordning for avfall. I noen deler av landet tilbys det avfallshenting, der brukeren legger inn bestillinger via de aktuelle nettsidene. De nettstedene teamet fant var:

- **hentavfall.no**
  - Opererer i Stavanger, Sandnes, Sola, Randaberg, Time og Gjesdal
  - https://www.hentavfall.no/
- **RagnSells**
  - Opererer i nedre Romerike og Oslo
  - https://butikk.ragnsells.no/gronn-bil
- **SøppelGutta**
  - Opererer i Viken og Oslo
  - https://www.soppel-gutta.no/
- **soppelsekk.no**
  - Opererer i Trondheim
  - https://www.soppelsekk.no/

4 Produktoversikt

4.1 Produktets rolle i brukermiljø



Figur 1: Brukeren benytter sin mobil for å legge inn bestillinger via applikasjon. Disse sendes til gjenvinningsstasjonen

4.2 Forutsetninger og avhengigheter

Applikasjonen forutsetter at brukeren har en Android eller iOS mobiltelefon for å laste ned tjenesten. Brukeren må befinne seg i en kommune Wasteflow opererer i. Forutsetter at brukeren har et gyldig mobilnummer som kan benyttes.

5 Produktets funksjonelle egenskaper

Nr.	Funksjonellt krav	Delkrav	
1	Legge inn bestilling av avfall	1.1	Valg av type avfall som skal hentes
		1.2	Adresse for henting
		1.3	Tidsintervall for henting
		1.4	Hvor mange seksjoner av bilen som trengs
		1.5	Prisen for bestillingen skal vises
2	Oppretting av bruker	2.1	Autentisering med mobilnummer
		2.2	Bestillinger koblet til brukeren
		2.3	Brukeren kan legge inn navn
3	Kunne se bestillingsinfo	3.1	Oversikt over avfallsgjenstander
		3.2	Adressen oppgitt
		3.3	Tidspunkt for henting
4	Se status for bestilling	4.1	Få opp at bestillingen er mottatt
		4.2	Få opp når hentebilen er på vei
		4.3	Se om bestillingen er fullført
		4.4	Prisen for bestillingen skal vises
		4.5	Kunne se hvor hentebilen er på kartet
		4.6	Se ”estimated time of arrival”
5	Endre bestilling	5.1	Endre avfallsgjenstandene
		5.2	Kunne endre adresse for henting
		5.3	Kunne endre tidspunkt for henting
6	Kansellere bestilling	6.1	Få bekreftele når den er kansellert
7	Få notifikasjoner	7.1	Om status endre seg
8	Muligheten for å gi tilbakemelding	8.1	Om systemet
		8.2	Etter gjennomført bestilling
9	Kunne se prisoversikt	9.1	Kunne se pris pr seksjon i hentebil
10	Kunne endre profilinformasjon	10.1	Kunne endre navn på brukeren
11	Betale for bestilling	11.1	Kunne gjennomføre betaling i applikasjonen

6 Ikke-funksjonelle egenskaper og andre krav

Ikke-funksjonelle krav
Applikasjonen skal lages med Flutter
Bruk av Googles Firebase Cloud Firestore for database
Bruk av Googles Firebase Cloud Functions som server
Kompatibel med både iOS og Android
Bruk av Cloud Firestores sikkerhetsregler for å sikre sensetive opplysninger
Bruk av stripe som betalingstjeneste

A Oppstartsdokument fra oppdragsgiver

Wasteflow App - Oppstartsdokument

Innledning

Bakgrunn

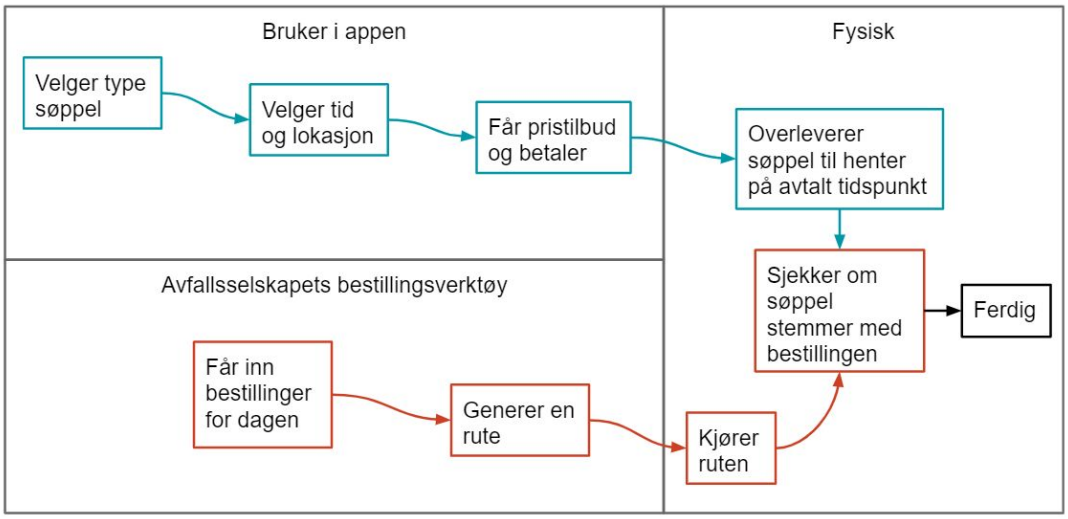
“Norge er blant et av de landene i verden som produserer mest søppel per innbygger. Vi produserte i 2018 omtrent 49% mer søppel enn gjennomsnittet blant de europeiske OECD-landene (OECD, 2020), og med en vekst på 12,7% mellom 2012 og 2018 har vi et voksende behov for effektiv og miljøvennlig håndtering av avfall (SSB, 2019).

I vår dialog med det Tromsø-baserte avfallsselskapet Remiks har vi avdekket at det er svært høy pågang på gjenvinningsstasjoner i helgene, som resulterer i flere timer lange køer som om sommeren til og med krever trafikkdirigering av politiet. Dette øker terskelen for å resirkulere spesialavfall som ikke kan kastes i vanlige søppeldunker. Et annet problem med dagens system er at de forutsetter at man har tilgang til bil for å kunne levere spesialavfallet. Dette bidrar til å øke terskelen for resirkulering ytterligere for studenter, pensjonister og samtlige som ikke har tilgang til bil.

For å kunne tilby bærekraftige og effektive løsninger som legger grunnlaget for framtidens byer er effektiv avfallshåndtering et kritisk punkt. I FNs bærekraftsmål for å stoppe klimaendringer innen 2030, er god avfallshåndtering et av undermålere for bærekraftige byer og lokalsamfunn. Å redusere avfallsmengden gjennom blant annet materialgjenvinning er et underpunkt under målet ansvarlig forbruk og produksjon. Disse målene er ikke tilfredsstilt i dag. Det må bli lettere for brukeren å håndtere avfall på en bærekraftig måte. (FN, 2020)”

Løsning

“Vår løsning for å effektivisere henting og levering av avfall som faller utenfor eksisterende henteordninger er en todelt digitalisering av prosessene. På forbrukersiden ønsker vi å utvikle en mobilapplikasjon der brukeren kan kjapt og enkelt bestille og betale for henting av avfall som de ellers måtte kjørt selv. Dette krever en modernisering av avfallsselskapenes systemer for å registrere innsamlet søppel og en utvidelse av disse systemene til å automatisk kunne motta og håndtere bestillinger for henting. I vår dialog med Remiks har vi foreløpig kommet frem til at hentingen vil gjennomføres av avfallsselskapet kjører en egen tilpasset lastebil for en rute som plukker opp søppel fra flere bestillinger.



Fordelen med denne løsningen er at en samling av flere mindre kjøreturer med personbiler til en kjøretur med en lastebil vil redusere utslippene knyttet til levering av avfall til gjenvinningsstasjoner. Ved at disse turene gjennomføres på hverdager i stedet for i helgene vil det bidra til mindre kø og dermed lavere terskel for gjenvinning og vil spre trykket på gjenvinningsanlegget i helgene utover hele uken.

På lengre sikt legger denne løsningen til rette for å digitalisere tjenester rettet mot bedriftssegmentet innen avfallshåndtering, som for eksempel bestilling av containere og håndtering av avfall fra byggeplasser og industriområder. På sikt kan det også være verdifullt å samle inn anonymisert data om avfallshentingen. Dette muliggjør omfattende dataanalyse som kan gi kommuner og offentlige aktører verdifull innsikt ved planlegging av fremtidig avfallshåndtering, byplanlegging og mer. Digitalisering av avfallshenting og datainnsamling rundt dette er også et nødvendig første skritt for å muliggjøre autonom søppelhenting i fremtiden ved hjelp av selvkjørende søppelbiler.”

Omfanget for bacheloroppgaven

- Skal lage appen for bestilling av søppelhenting
    - Er for forbrukerne
    - Autentisering (Mobil)
        - Skriver inn mobilnummer og får kode på melding
    - Registreringsflow
        - Første gangen man åpner appen må man fylle ut profil-info
            - Fornavn, etternavn, osv.
    - Bestille søppelhenting
        - Legg til gjenstand
            - Velge type søppel
            - Oppgi størrelse (?)
        - Oppgi adresse for henting
        - Velge tidspunkt for henting
            - Dynamisk valg av tidsrom
            - Tilpasset hver enkelt søppelaktør
        - Vise pris
            - Kalkuleres gjennom et API som studentene lager (Firebase Function)
    - Status av bestilling
        - Mottatt, Registrert i rute, på vei (ETA, evt. vise søppelbilen i kart), underkjent
        - Hentes fra databasen
        - Avbestille, endre
        - Vise bestillingsinfo
    - Notifikasjoner
        - Oppdatert status
    - Vise info på hvordan tjenesten fungerer (“i”-ikonet)
        - Pris-info
            - Pris per seksjon i bilen
            - Pris på vekt
    - Profilinnstillinger
        - Navn
    - Feedback
        - Kunne gi feedback om systemet
        - Spørre om å gi feedback etter gjennomført bestilling
    - Betaling **(lav prio)**
        - Bruk av for eksempel Stripe
    - Google analytics **(lav prio)**
        - Antall aktive brukere
        - Hvor mange fullfører bestillingsskjemaet?

Krav

Ikke-funksjonelle krav

- Valg av teknologi
    - Flutter (for appen)
    - Firebase
        - Autentisering
        - Notifikasjoner
        - Database (Firestore)
        - APIs (Firebase Functions)
- Basic sikkerhet bør være på plass
- Følge wireframes.

Vedlegg B Kravsdokument

,

Prosjekt nr 078

Wasteflow Kravdokumentasjon

Versjon 1.1

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
28.01.2021	0.1	Start på første utkast	Sabine Seljeseth og Sivert Utne
03.02.2021	0.2	Tilføying av User-Stories	Sabine Seljeseth
18.02.2021	0.3	Oppdatering av domenemodell	Sabine Seljeseth og Sivert Utne
22.02.2021	0.4	Utfylling av bildebeskrivelser og tekster samt korrigeringer	Sivert Utne
23.02.2021	1.0	Korrigering av skrivefeil og formuleringer	Sabine Seljeseth og Sivert Utne
17.05.2021	1.1	Justeringer og formattering	Sabine Seljeseth og Sivert Utne

Innhold

1 Introduksjon

Hensikten med dette dokumentet er å definere hvilke krav som skal oppfylles av applikasjonen. Dokumentet inngår i dokumentasjonen gjort i forbindelse med bacheloroppgaven ved NTNU. Kravene er fremstilt ved bruk av User-Stories. I dokumentet er wireframesene som er gitt fra oppdragsgiver inkludert. I tillegg er domenet til applikasjonen illustrert i domenemodellen.

2 User-stories

Som **system administrator**
Ønsker jeg at **personer autentiserer seg**
Slik at jeg **vet at bestillinger blir gjort av ekte personer**
og at **personer ikke kan utgi seg for å være noen andre**

Akseptansekriterier:

- Brukeren fyller inn gyldig mobilnummer
- Brukeren får et engangspassord på SMS
- Brukeren utfyller passordet som stemmer overens med engangspassordet fra SMS

Som **en person**
Ønsker jeg **å opprette en bruker**
Slik at jeg **får tilgang til systemet**

Akseptansekriterier:

- Jeg har gyldig mobilnummer
- Jeg har fylt ut feltene mobilnummer og navn
- Jeg autentiserer meg via mobilen
- Jeg kan lagre informasjon om:
  - Adresser
  - Epost for kvitteringer
- Jeg har trykket godkjenn vilkår
- Jeg har trykket lagre

Som **bruker**
Ønsker jeg **å lagre én eller flere adresser**
Slik at jeg **slipper å fylle inn på nytt hver gang**

Akseptansekriterier:

- Jeg kan bruke adresser jeg har brukt i tidligere bestillinger

Som **bruker**
Ønsker jeg å **legge inn en bestilling for avfallshenting**
Slik at jeg **blir kvitt avfallet**

Akseptansekriterier:

- Jeg er logget inn
- Jeg befinner meg i en by Wasteflow opererer i
- Jeg har fylt ut:
  - Gyldig adresse
  - Avfallet som skal hentes og dimensjonen på det
  - Dato- og tidsintervall for hentingen
- Systemet forteller meg hvor mye det vil koste
- Jeg trykker send bestilling

Som **bruker**
Ønsker jeg å **se status for bestilling**
Slik at jeg **vet at bestillingen min blir behandlet**
og at jeg **vet om hentebilen er på vei**

Akseptansekriterier:

- Jeg har en bestilling som ikke er fullført
- Jeg har fått bekreftelse på at bestilling er mottatt
- Kan se forventet hentetidspunkt
- Kan se om hentebilen er på vei
- (Mulig utvidelse) se hvor hentebilen befinner seg på et kart

Som **bruker**
Ønsker jeg å **kunne endre en bestilling**
Slik at **bestillingen blir oppdatert med ny info**

Akseptansekriterier:

- Jeg har en aktiv bestilling
- Bestillingen skal ikke gjennomføres samme dag
- Alle feltene er korrekt utfylt
- Jeg får bekreftelse på at bestillingen er endret

Som **bruker**
Ønsker jeg å **kunne avbestille en bestilling**
Slik at jeg **får bekreftet at bestillingen min er avbestilt**

Akseptansekriterier:

- Jeg har en aktiv bestilling
- Bestillingen skal ikke gjennomføres samme dag
- Bruker får bekreftelse på at hentingen er avbestilt

Som **bruker**
Ønsker jeg å **se bestillingsinfo**
Slik at jeg **kan se hva jeg har bestilt**

Akseptansekriterier:

- Jeg har en aktiv eller fullført bestilling
- Jeg kan for hver bestilling se en oversikt over:
  - Avfallet lagt inn i bestillingen
  - Henteadresse
  - Tidspunkt bestilling ble sendt
  - Tidspunkt bestilling blir/ble fullført
  - Prisen

Som **bruker**
Ønsker jeg å **få notifikasjoner**
Slik at jeg **blir informert om statusen på min bestillingen endrer seg**

Akseptansekriterier:

- Jeg har en aktiv bestilling
- Jeg får opp en notifikasjon på mobilen når statusen er endret

Som **bruker**
Ønsker jeg å **kunne endre profilen min**
Slik at **den blir oppdatert med den nye informasjonen**

Akseptansekriterier:

- Har fylt inn informasjon som skal endres
- Trykket på lagre

Som **bruker**
Ønsker jeg å **vite hva tjenesten koster**
Slik at jeg **kan vurdere om jeg ønsker å bestille**

Akseptansekriterier:

- Jeg kan velge hva jeg vil ha hentet og se en beregnet pris for dette
- Se pris for henting av vanlige gjenstander (sofa, vaskemaskin osv.)

Som **bruker**
Ønsker jeg å **kunne gi tilbakemeldinger om tjenesten**
Slik at jeg **kan varsle om feil eller komme med forslag til forbedringer**

Akseptansekriterier:

- Jeg kan sende en egenskrevet melding til noen ansvarlige for tjenesten
- Jeg får respons på at tilbakemeldingen min er mottatt

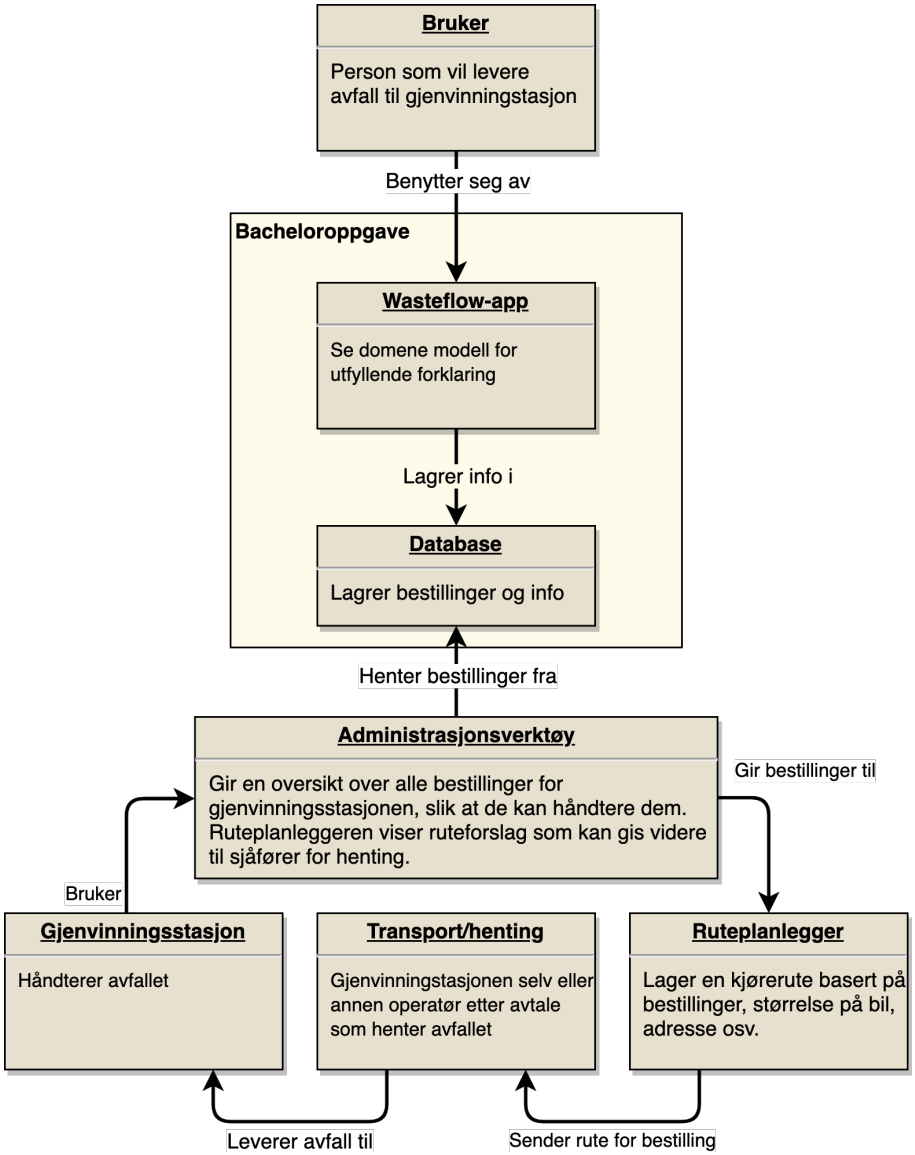
Som **bruker**
Ønsker jeg å **betale for min bestilling**
Slik at **bestillingen min er gjort opp for**

Akseptansekriterier:

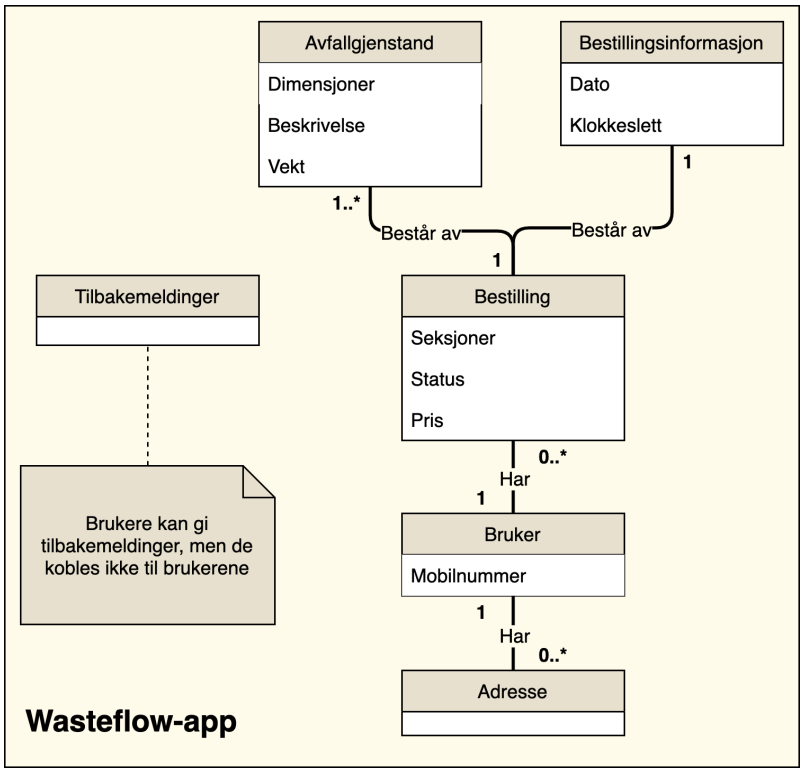
- Fylt inn gyldig betalingsinfo
- Betalingen blir godkjent

3 Domenemodell

Wasteflow er et større pilotprosjekt. Systemet er illustrert i figur 1. Teamet sitt ansvarsområdet er markert i figuren. Teamet står for utviklingen av Wasteflow mobilapplikasjonen og oppretting av databasen som bestillingene skal lagres i. Videre vil systemet bygges opp av et administrasjonsverktøy som vil brukes av gjenvinningstasjonen og ruteplanlegging som vil brukes av hentebilen. Hentebilen vil deretter levere avfallet etter plan til gjenvinningstasjonen. Hvordan mobilapplikasjonen er bygget opp er illustrert i figur 2.



Figur 1: Systemets oppbygging

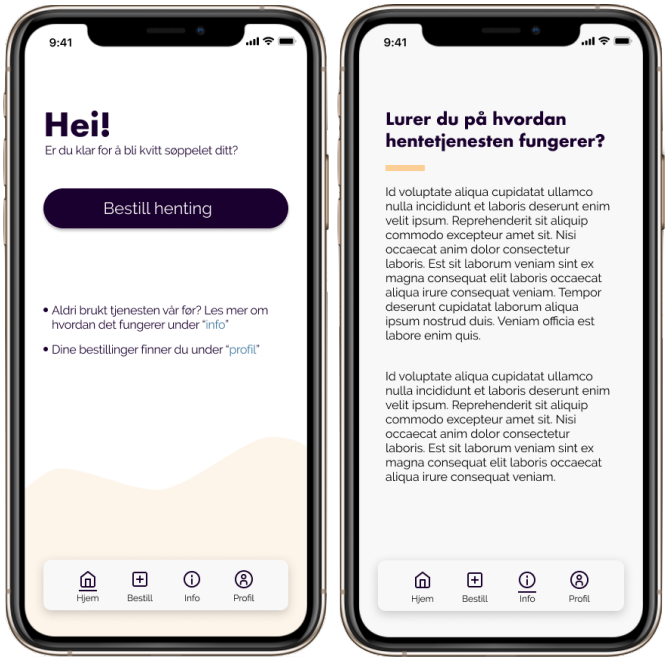


Figur 2: Domenemodell

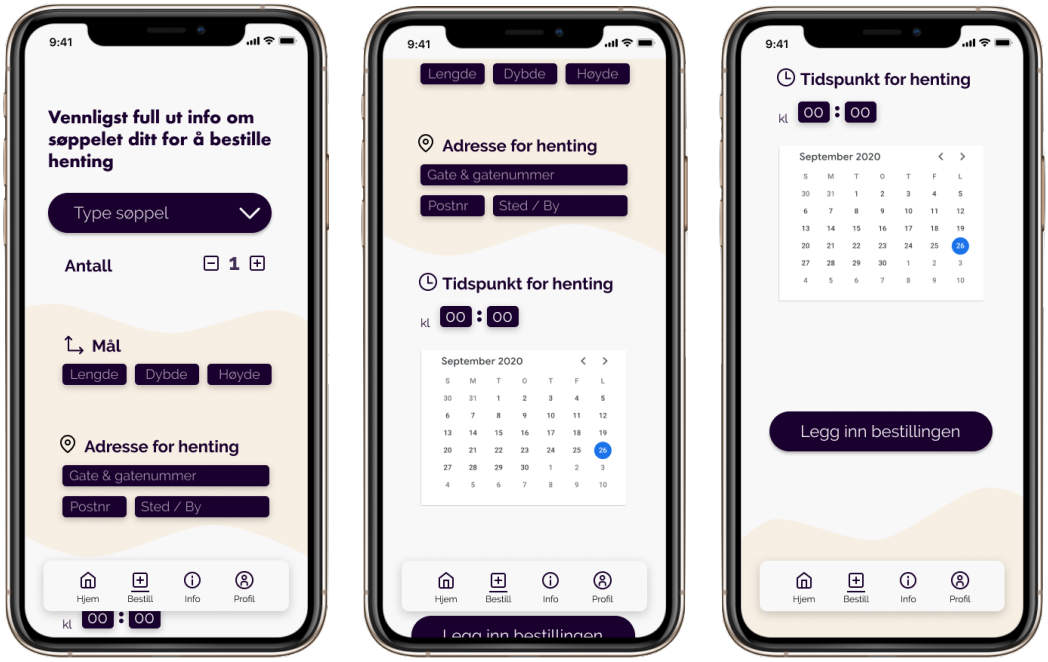
4 Prototyper

4.1 Wireframes

Oppdragsgiver stilte med wireframes for hvordan applikasjonen skulle se ut. Figur 3 viser hvordan hjemsiden og informasjonssiden til applikasjonen skal se ut. I tillegg er det laget wireframes for bestillingssiden til Wasteflow, som vist i figur 4. Tanken er at applikasjonens design og fargevalg skal være så lik som mulig wireframesene.



Figur 3: Hovedmeny og Info



Figur 4: Wireframesene til Wasteflow mobilapplikasjonen

Vedlegg C Systemdokumentasjon

,

Prosjekt nr 078

Wasteflow Systemdokumentasjon

Versjon 1.0

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
07.05.2021	0.1	Opprette mal	Sivert Utne
14.05.2021	0.2	Fullført første utkast	Sivert Utne
19.05.2021	1.0	Feilrettinger og endringer	Sabine Seljeseth og Sivert Utne

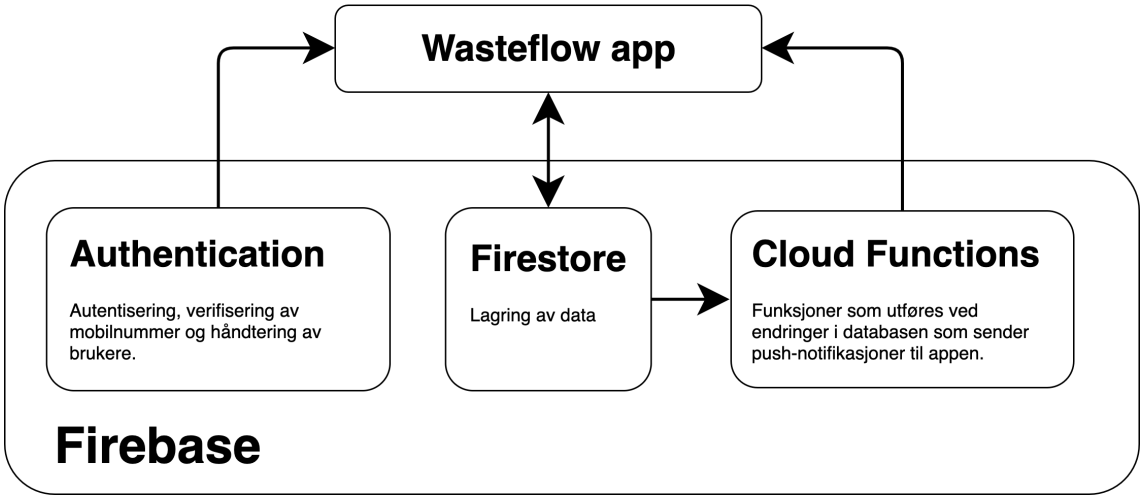
Innhold

1 Introduksjon

Dette dokumentet er skrevet i forbindelse med bacheloroppgaven Wasteflow. Dokumentet gir en oversikt over de sentrale delene av det utviklede systemet. Hensikten med dokumentet er å gi en helhetlig forståelse for hvordan systemet er bygd opp. Dokumentet inneholder informasjon om prosjektstrukturen, dokumentasjon av kildekode, oversikt over klassenes oppbygging, databasens oppbygning, avhenigheter og sikkerhet.

2 Arkitektur

2.1 Prosjektarkitektur



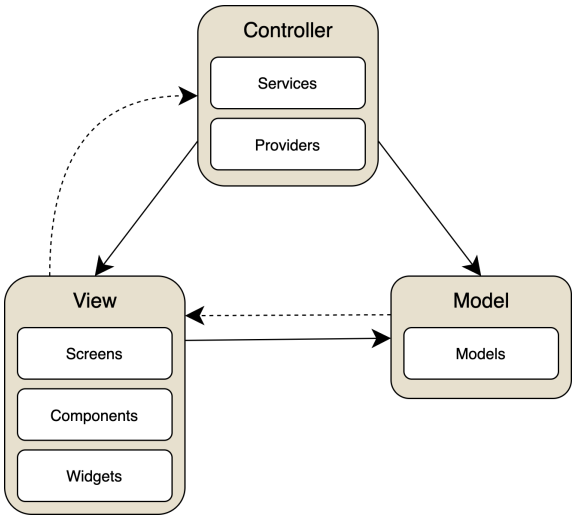
Figur 1: Hovedarkitektur til systemet

Applikasjonen bruker Firebase som backend, det er derfor ingen egen server for applikasjonen.

Pilene i figur 1 viser den primære kommunikasjonsveien til informasjonen. Wasteflow henter autorisasjons og brukerdata fra Firebase Authentication. Applikasjonen både sender data til, og henter data fra Firebase sin NoSQL-database Firestore. Firebase Cloud Functions varsles ved endringer i Firestore og sender melding til applikasjonen.

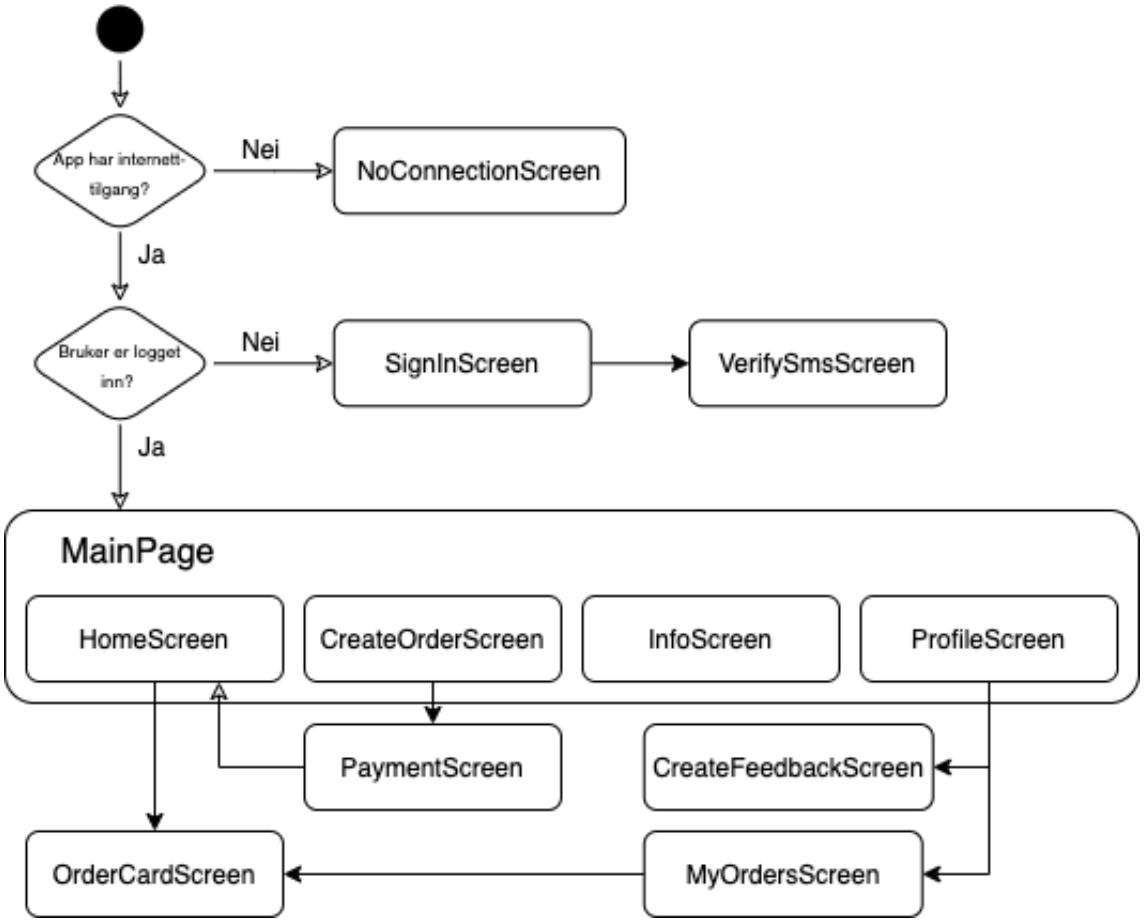
2.2 Applikasjon-arkitektur

Applikasjonen følger det tradisjonelle "Model-View-Controller" designmønsteret. Figur 2 viser de viktigste bibliotekene som hører til de forskjellige delene av MVC. Disse bibliotekene beskrives nærmere i kapittel 3.



Figur 2: Model-View-Controller mønster for applikasjonen

2.2.1 Flytdiagram



Figur 3: Navigasjon i appen

Figur 3 viser flyten i applikasjonen. Den sorte sirkelen er "startpunktet" for applikasjonen. De rombeformete boksene er sjekker som blir gjort kontinuerlig. Dersom en av disse sjekkene endrer seg på noe tidspunkt vil veivalget endres. F.eks. dersom applikasjonen mister internett-tilgang

vil applikasjonen gå til "NoConnectionScreen", uansett hvor brukeren måtte befinne seg ellers i applikasjonen.

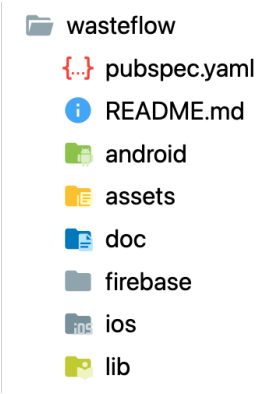
MainPage er hovedsiden i Wasteflow og består av fire sider, disse fire sidene kan brukeren navigere fritt imellom ved å f.eks. bruke navigasjonsbaren på bunnen av siden eller ved å sveipe.

Pilene markerer hvilken retning navigasjonen skjer. En pil med sort spiss betyr at bruker kan returnere fra denne siden til pilens startpunkt. En pil med hvit spiss indikerer at bruker kan navigere til denne sidenen, men ikke gå direkte tilbake til startpunktet til pilen. Dette betyr at den eneste måten brukeren kan gå tilbake til innloggingssiden etter bruker har logget inn er ved å logge ut.

3 Prosjektstruktur

Her vil vi gå gjennom hvordan prosjektstrukturen er bygd opp, hva de forskjellige mappene inneholder, samt de viktigste filene for prosjektet. Vi har altså ikke med alle genererte filer og annen boilerplate.

3.1 Mappestruktur



Figur 4: Overordnet mappestruktur

Figur 4 viser den overordnede strukturen til prosjektet.

pubspec.yaml inneholder alle avhengigheter for prosjektet, i likhet med en **package.json** fil i et javascript prosjekt.

Mappene **android** og **ios** inneholder platform spesifikke filer, disse trengs i hovedsak ikke å endres på, men enkelte tillateleser og innstillinger som håndteres ulikt av platformene må gjøres i filer i disse mappene.

Mappen **doc** inneholder generert dokumentasjon av kildekoden.

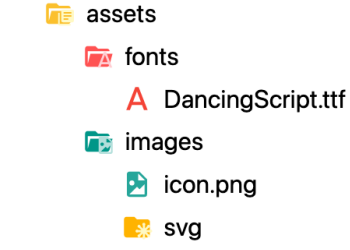
Mappene **firebase** og **lib** er mappene som inneholder koden til prosjektet, vi skal se nærmere på hva disse inneholder.

Figur 5: Mappestruktur

Figur 5 viser den utvidede mappestrukturen til prosjektet.

Mappen **lib** inneholder all kode for applikasjonen og filen **main.dart** er startpunktet for prosjektet. Videre i **lib** har vi mappene **libraries** og **src**.

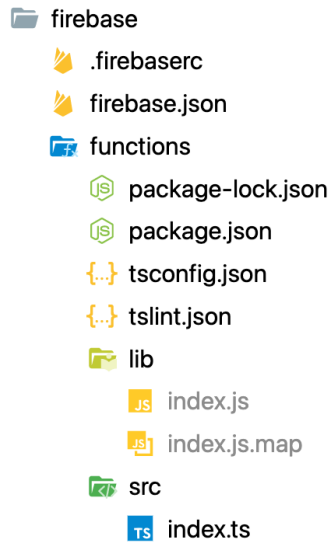
3.1.1 Ressurser



Figur 6: wasteflow/assets

assets inneholder diverse ressurser. Dette er ting som bilder, skrifttyper og **.svg** filer. Viktigste filen her er **icon.png**, som er ikonet som brukes for appen.

3.1.2 Firebase

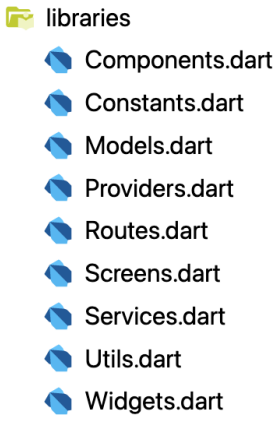


Figur 7: wasteflow/firebase

firebase inneholder alle filer som omhandler Firebase og dets funksjonaliteter. I denne applikasjonen er det foreløpig kun brukt Firebase Cloud Functions, disse ligger i mappen **functions**.

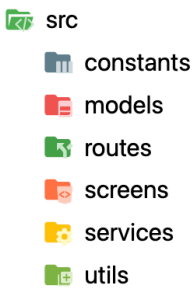
Funksjonene i seg selv ligger i **index.ts** filen. Typescript er brukt for å ha tett kontroll på typene i funksjonene for lettere og tidlig feilhåndtering. Denne filen blir konvertert til javascript filen under deployering til **index.js**. Samtidig opprettes **index.js.map** som kobler linjenummer osv. fra **index.js** filen som kjøres til de korrekte linjenumrene i **index.ts** filen for å løse eventuelle feil. Det er altså kun **index.ts** filen som trengs å endres på i **firebase**.

3.1.3 Kildekode



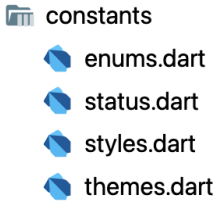
Figur 8: wasteflow/lib/libraries

libraries mappen inneholder filer som kun ekporterer filer fra **src** mappen. Det er altså ikke noe kildekode i disse filene. Grunnen til at dette gjøres er at alle filer som ligger i **src** mappen i utgangspunktet er skjult. Å samle kildekodefilene i slike bibliotek gjør det også lettere å importere de delene av koden man trenger. F.eks. dersom man skal jobbe med layout trenger man kun importere **Widgets.dart** for å ha tilgang til alle egendefinerte widgets, istedenfor å måtte importere dem hver for seg.



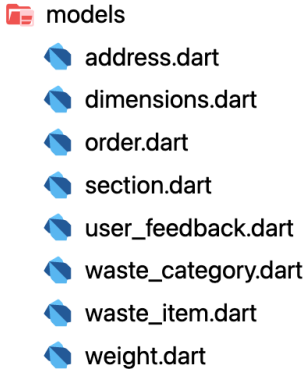
Figur 9: wasteflow/lib/src

src mappen inneholder all kildekoden til applikasjonen. Kildekoden er organisert etter hvilken funksjon de har i applikasjonen, som igjen bestemmer hvilket bibliotek de hører til i. Hver mappe inneholder altså filene til kun ett bibliotek i **libraries**.



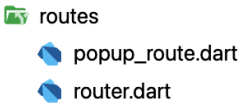
constants inneholder klasser og verdier som er konstante. Disse verdiene trengs sjeldent å endre på og styrer hovedsakelig utseende og layout for widgets.

Figur 10: wasteflow/lib/src/constants



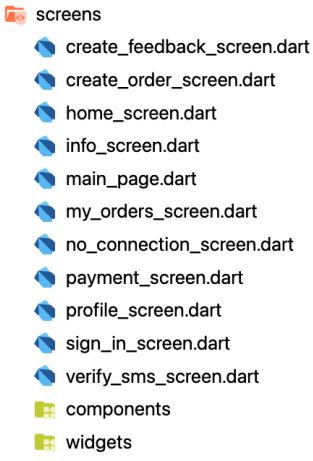
Mappen **models** inneholder objektene som brukes i applikasjonen. Dette er klassene som beskrives i klassediagrammet og er alle entitetene Wasteflow må forholde seg til.

Figur 11: wasteflow/lib/src/models



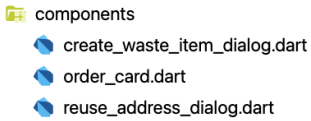
Mappen **routes** inneholder alt som er koblet til navigering mellom forskjellige sider i Wasteflow. Filen **router.dart** inneholder argumentene for hver side, samt navngitte ruter med egendefinerte håndteringer for enkel navigering fra hvor som helst i appen.

Figur 12: wasteflow/lib/src/routes



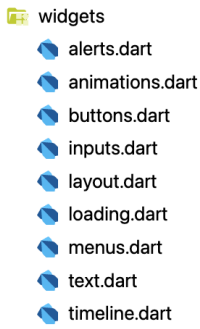
Mappen **screens** inneholder naturlig nok alle de forskjellige sidene. En side vil være en Widget som tar opp hele mobilskjermen. Videre har **screens** to undermapper, **components** og **widgets**. Mappen **screens** inneholder altså all kode som håndterer utseende og layout av applikasjonen.

Figur 13: wasteflow/lib/src/screens



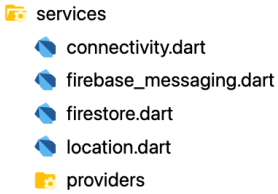
Mappen **components** inneholder større komponenter som er satt sammen av flere widgets, men som ikke tar opp en hel mobilskjerm.

Figur 14: wasteflow/lib/src/screens/componentts



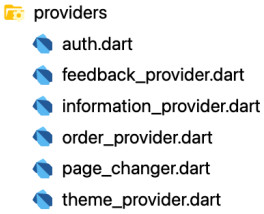
Figur 15: wasteflow/lib/src/screens/widgets

Mappen **widgets** inneholder alle egendefinerte widgets. Dette er widgets som gjerne brukes flere steder i applikasjonen, eller som tar ut enkelt mer kompliserte komponenter for å opprettholde en god Seperation of Concerns (som f.eks. **timeline.dart**).



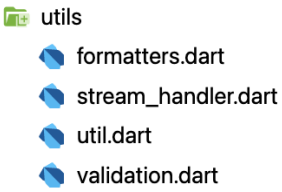
Figur 16: wasteflow/lib/src/services

Mappen **services** er mappen som inneholder logikken i Wasteflow. Dette innebærer kommunikasjon med Firestore, håndtering av meldinger osv.



Figur 17: wasteflow/lib/src/services/-providers

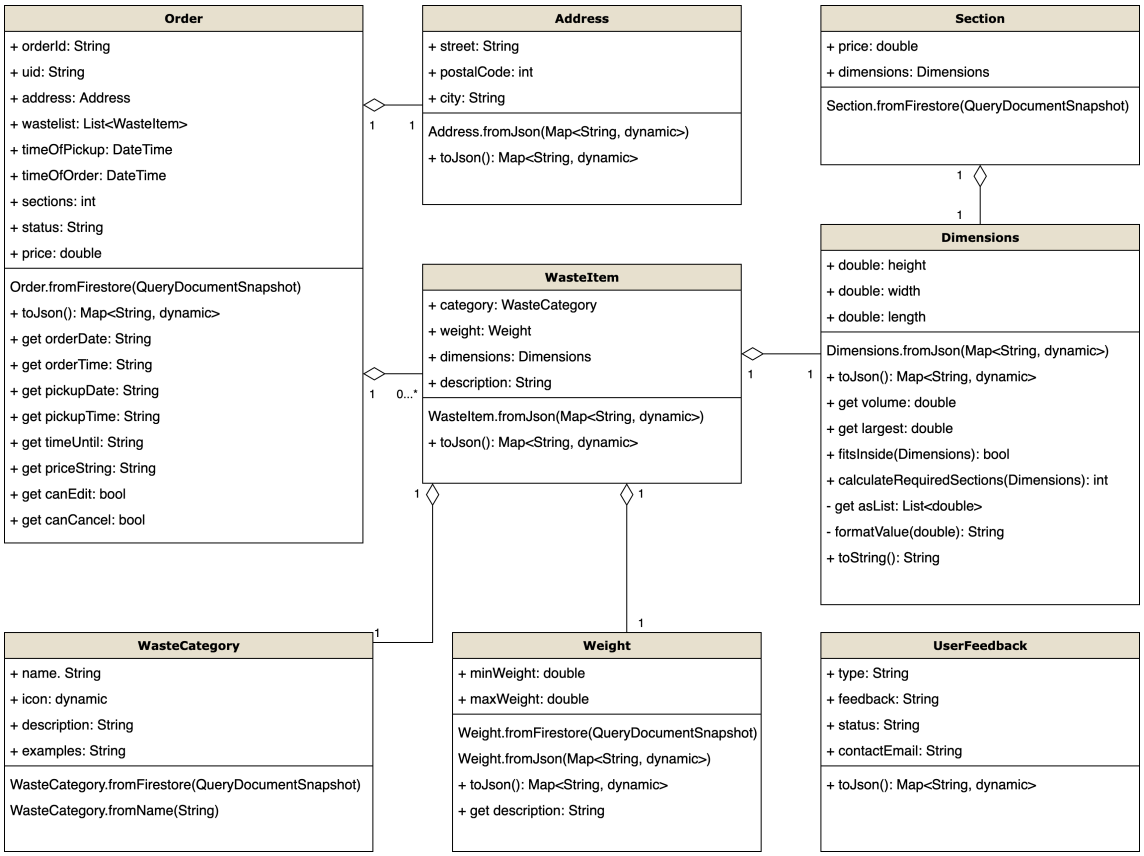
Mappen **providers** inneholder klasser applikasjonen bruker som "Providere". Dette er klasser som opprettes tidlig i widget treet og kan dermed brukes av alle komponenter og sider på en slik måte at de ikke alle drukner i samme klasse. Foreløpig er alle av typen "ChangeNotifier", som betyr at dersom det f.eks. er en endring i Firestore, sier klassen ifra til alle widgets som bruker en av dens verdier at de må bygges på nytt.



Figur 18: wasteflow/lib/src/utils

Mappen **utils** inneholder metoder og funksjoner som brukes flere ganger i Wasteflow, eller som forenkler koden andre steder i applikasjonen. Dersom en funksjon er nyttig, men ikke hører hjemme i en annen klasse plasseres de her.

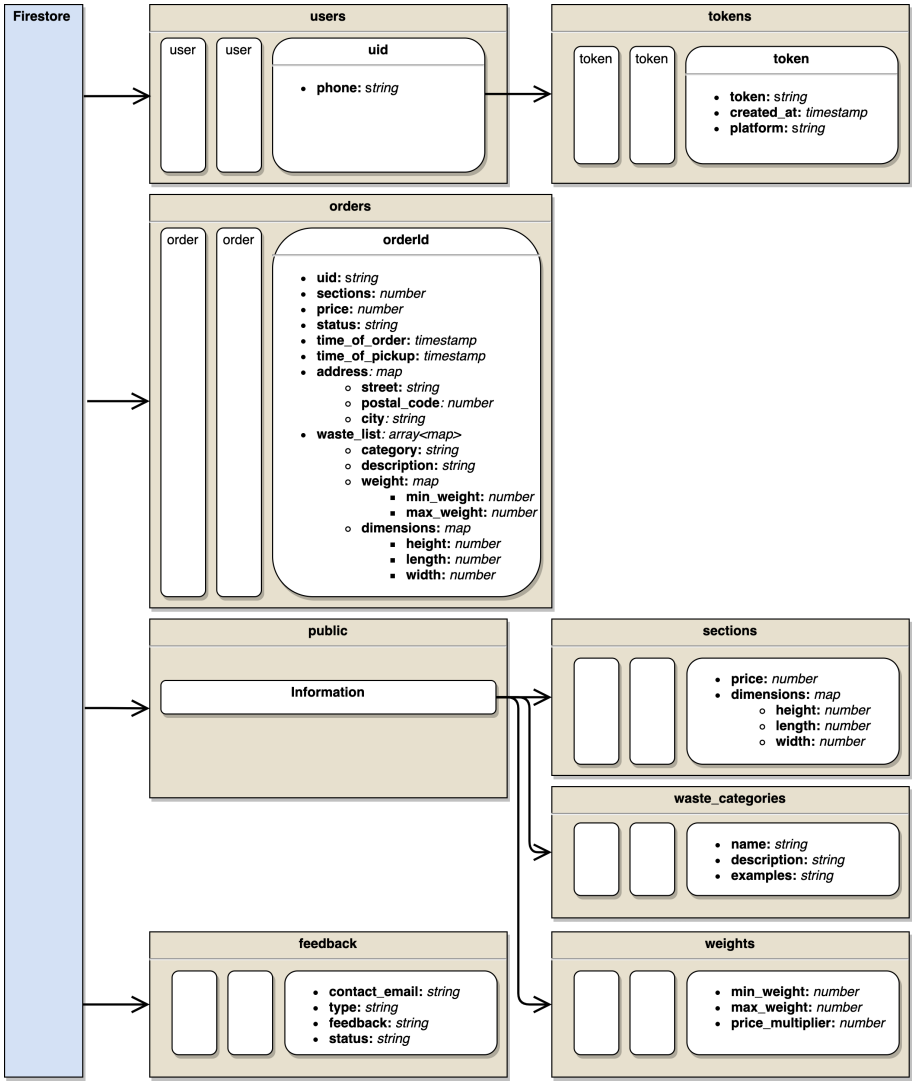
4 Klassediagram



Figur 19: Klassediagram

Figur 19 viser oversikten over objektene som er brukt i applikasjonen. Disse objektene er tett knyttet til databasen og inneholder derfor metoder for å konvertere til og fra datatypen som databasen lagrer data i.

5 Databasemodell

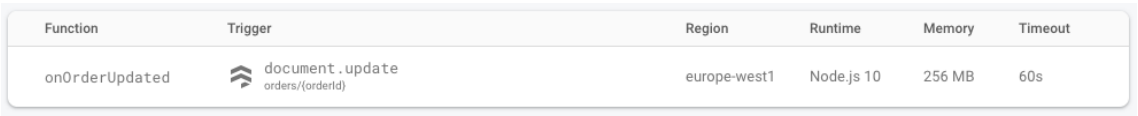


Figur 20: Databasemodell

Figur 20 gir en oversikt over hvordan informasjonen lagres i databasen. Databasen som brukes er en NoSQL-database fra Firebase og heter Firestore. Firestore lagrer informasjonen i form av dokumenter i forskjellige samlinger (”Collections”). Videre kan hvert dokument ha sine egne ”Sub-Collections”.

6 Server-tjenester

Applikasjonen har ikke en egenutviklet server med endepunkter da applikasjonen bruker Firebase sin Software development kit (SDK) for databasekall og andre tjenester. Applikasjonen bruker allikevel Firebase Cloud Functions (FCF), som kan regnes som server-tjenester ettersom dette er funksjoner eller ”endepunkter” som kjøres på en tjener hos Firebase. Disse funksjonene kan knyttes til HTTPS-kall, men i applikasjonen brukes kun FCF til å sende push-notifikasjoner til brukere ved endringer i deres bestilling. Dette skjer direkte ved å koble en funksjon til Firestore.



Function	Trigger	Region	Runtime	Memory	Timeout
onOrderUpdated	document.update orders/{orderId}	europe-west1	Node.js 10	256 MB	60s

Figur 21: server-funksjon

Figur 21 viser endepunktet og triggeren til endepunktet (hvilken endring i Firestore funksjonen utføres etter).

7 Sikkerhet

7.1 Autorisasjon

Ettersom applikasjonen er tett knyttet til Firebase betyr dette at vi kan bruke sikkerhetsfunksjonaliteten til Firebase, Firebase Authentication, som ikke bare er enklere, men også sikrere. Autorisasjon og innlogging via Firebase håndterer alt av oppretting, bekrefting, oppdatering og validering.

7.2 Tilgang til data

For å sikre dataene til brukeren, slik at kun de som har tilgang til dataene får tak i de, bruker vi sikkerhetsreglene i Firestore. Disse reglene bestemmer hvem som kan lese, redigere, opprette og slette dokumenter i de forskjellige samlingene. Det som gjør dette både enkelt og sikkert er at disse reglene kobles direkte til Firebase Authentication slik at vi kan bruke autorisasjonsinfoen direkte i reglene.

```
rules_version = '2'
service cloud.firestore {

  /// Functions ///

  function senderIsSignedIn() { return request.auth != null; }
  function senderId() { return request.auth.uid; }
  function existingData() { return resource.data; }
  function incomingData() { return request.resource.data; }

  /// Rules ///

  match /databases/{database}/documents {
    match /public/{category}/{document=**} {
      allow read: if senderIsSignedIn();
    }
    match /users/{userId}/{document=**} {
      allow read, write: if senderId() == userId;
    }
    match /orders/{orderId} {
      allow create: if senderId() == incomingData().uid;
      allow read, update, delete: if senderId() == existingData().uid;
    }
    match /feedback/{feedbackId}{
      allow create: if senderIsSignedIn();
    }
  }
}
```

Figur 22: Sikkerhetsregler for Firestore

Figur 22 viser reglene vi har opprettet for prosjektet. For lesbarhetenes skyld har vi opprettet funksjoner for vanlig data.

Vi har blant annet regler for at en bruker kun kan opprette en ordre med sin egen uID. Dersom en

bruker (Ole) f.eks. har klart å få tilgang til en annen bruker (Per) sin uID, og Ole har kommet seg inn i koden til applikasjonen for å manuelt opprette en ordre med uID-en til Per , vil ikke Ole få lov til dette ettersom brukeren som er logget inn (Ole) ikke har samme uID som den som blir sendt (incomingData().uid). Det samme skjer dersom en bruker vil lese/endre en bestilling, da krever det at de er innlogget som brukeren med uID i ordren.

Det er også verdt å nevne at tillatelser utenom de som er eksplisitt gitt gjennom reglene er ugyldige. Det betyr at med reglene i figur 22 er det f.eks. ikke mulig for en bruker å gjøre noe annet enn å opprette dokumenter i samlingen "feedback".

7.3 Injection

Firestore vil ikke være sårbar for injection. I motsetninger til SQL databaser er firestore en NoSQL-database der man ikke bygger SQL kommandoer for å utføre spørringer. Istedenfor brukes en API som er gitt av SDK-en. Her blir en hver streng automatisk behandlet av API-et. Derfor vil ikke injections være et problem. For å forsikre oss om for at brukeren ikke kan har tilgang til noe de ikke burde, brukes autentisering og sikkerhetsreglene som beskrevet i kapittel 7.2.

7.4 Cross-Site Scripting

Cross-Sire scripting, som innebærer at en bruker skriver inn for ekesmpel html eller javascript kode som senere vil bli kjørt av enten en server eller klient er også forhindret gjennom Firebase.

8 Installasjon og kjøring

8.1 Anbefalinger

Vi anbefaler å bruke **Visual Studio Code (VSCode)** som kan lastes ned her:
https://code.visualstudio.com/Download

Grunnen til at vi anbefaler dette er at VSCode har tillegg som er til god hjelp under utvikling, samtidig som VSCode enkelt kan bytte mellom å kjøre på iOS eller Android, i motseting til f.eks. Android Studio eller Xcode. Resten av installasjons og kjøringshjelpen kommer til å ta utgangspunkt i VSCode.

Noen tillegg vi anbefaler å installere til VSCode er:

- Awesome Flutter Snippets
  Snarveier til å opprette ofte brukte widgets/klasser
- dart-import
  Automatisk organisering av importerte pakker

8.2 Avhengigheter

Applikasjonen er avhengig av følgende eksterne biblioteker.

Firebase

- **firebase_core** - https://pub.dev/packages/firebase_core
- **firebase_auth** - https://pub.dev/packages/firebase_auth

- **cloud_firestore** - https://pub.dev/packages/cloud_firestore
- **firebase_messaging** - https://pub.dev/packages/firebase_messaging

Tjenester

- **provider** - Enkelt dele objekter og instanser mellom sidene i applikasjonen: https://pub.dev/packages/provider
- **shared_preferences** - Lagring av verdier/innstillinger til lokalt minne på mobiltelefonen: https://pub.dev/packages/shared_preferences
- **connectivity** - Sjekk etter om det er tilgjengelig internett-tilkobling: https://pub.dev/packages/connectivity
- **location** - Hent nåværende posisjon til mobiltelefonene: https://pub.dev/packages/location
- **geocoder** - Konvertere posisjon til nærmeste adresser: https://pub.dev/packages/geocoder

Annet

- **cupertino_icons** - iOS ikoner: https://pub.dev/packages/cupertino_icons
- **font_awesome_flutter** - Enda flere ikoner fra diverse ikon-bibliotek: https://pub.dev/packages/font_awesome_flutter
- **auto_size_text** - Automatisk endring av skriftstørrelse etter tilgjengelig plass: https://pub.dev/packages/auto_size_text
- **flutter_svg** - Vise **.svg** filer som bilder: https://pub.dev/packages/flutter_svg
- **timelines** - Opprette tidlinjer: https://pub.dev/packages/timelines

8.3 Installasjon av prosjektet

I tilfeller hvor kommandolinje kommandoer beskrives gis de på formen;

```
/mappe/kommando/kjørs/i > kommando
```

8.3.1 Nedlastning av prosjektet

Prosjektet blir overtatt av Favn Software AS etter endt prosjekt. Favn Software er eieren av prosjektet i GitHub, dette betyr at de kan klone/laste ned prosjektet herfra. Kildekoden er ellers ikke tilgjengelig for offentligheten da den etter endt prosjekt tilhører Favn Software. Kildekoden blir derimot lastet opp i en **.zip** fil som vedlegg sammen med hovedrapporten dersom sensor ønsker å se på og/eller kjøre den produserte kildekoden.

8.3.2 Installasjon av Dart og Flutter

Installasjon av Dart og Flutter i VSCode gjøres enkelt ved å installere utvidelsen **Flutter** fra utvidelseskatalogen, dette vil også automatisk installere Dart utvidelsen.

8.3.3 Installasjon av avhengigheter

Alle avhengigheten for applikasjonen ligger i filen **pubspec.yaml** og installeres ved å bruke kommandoen

```
/wasteflow > flutter pub get
```

I VSCode er det også mulig via instillinger at nye avhengigheter lastes ned automatisk ved lagring av filen.

Avhenighetene for Firebase Cloud Functions installeres ved å bruke kommandoen:

```
/wasteflow/firebase/functions > npm install
```

For å sjekke at alt er installert korrekt brukes kommandoen:

```
/wasteflow > flutter doctor -v
```

8.4 Kjøring av prosjektet

For å kjøre prosjektet må man åpne filen **main.dart** i mappen **wasteflow/lib**. Her er det en funksjon som heter **main**, over denne vil valgene "run" og "debug" vises. "Hot-reload" osv fungerer i både "run" og "debug", men under "debug" vil applikasjonen stoppe utførelsen ved feil. Etter å ha valgt "run" eller "debug" vil VSCode starte applikasjonen i en tilkoblet/åpen enhet/emulator/simulator, hvis ingen er tilkoblet får man opp et valg om hvilken enhet VSCode skal kjøre applikasjonen i.

8.4.1 Virtuelle enheter

Android emulator
For å bruke en Android emulator trengs **Android Studio**:
https://developer.android.com/studio?gclid=Cj0KCQjw4ImEBhDFARIsAGOTMj_zFBoxwgHyJ8ZMRFiPyua-s0SPvYtQ-4SpzqVjjEn9C4ysBKybaFIaAmkHEALw_wcB&gclsrc=aw.ds

Etter Android Studio er installert kan man opprette en emulator ved å gå til: **Tools > AVD manager**. Her vil det vises en oversikt over alle Android emulatorer, og man kan også opprette nye. Etter en emulator er opprettet sørg for å starte den, da vil man kunne bruke denne direkte fra VSCode.

iOS simulator
Å bruke en iOS simulator krever en datamaskin med MacOS, deretter trengs programmet **Xcode**:
https://developer.apple.com/xcode/.

Etter Xcode er installert kan man åpne simulatorprogrammet på et par forskjellige måter:

- åpne spotlight (**cmd + *space***) og skriv **simulator** for å åpne programmet.
- etter å ha åpnet Xcode, i menylinjen velg **Xcode > åpne utviklerverktøy > Simulator**

Når man har fått åpnet simulatorprogrammet kan man starte eller endre en enhet fra menylinjen **Fil > åpne simulator > iOS > *ønsket enhet***.

8.4.2 Fysiske enheter

For å kjøre applikasjonen på en fysisk enhet må man først koble til enheten til datamaskinen via USB.

Android
Følg følgende steg etter å ha koblet enheten til datamaskinen:

1. Ha utvikleralternativer skrudd på. Dette gjøres ved å trykke på **Build-nummer**et til mobiltelefonen 7 ganger. Du finner dette her:

   - Android 9 (API level 28) og høyere:
     Innstillinger > Om mobiltelefonen > Build-nummer
   - Android 8.0.0 (API level 26) og Android 8.1.0 (API level 26):
     Innstillinger > System > Om mobiltelefonen > Build-nummer
   - Android 7.1 (API level 25) og lavere:
     Innstillinger > Om mobiltelefonen > Build-nummer

2. Gå inn i utvikleralternativer, sørg for at de er skrudd på og bla ned til **Feilsøking**. Her må **USB-debugging** være skrudd på.

iOS
Teamet har kun brukt appen på en fysisk Android enhet, men instruksjoner for å kjøre på en fysisk iOS enhet finnes her:
https://medium.com/front-end-weekly/how-to-test-your-flutter-ios-app-on-your-ios-device-75924bfd75a8

8.5 Gjøre endringer i prosjektet

8.5.1 Applikasjonen

Flutter vil utføre såkalt "hot-reload" ved lagring av endringer i koden. Ved en "hot-reload" vil tilstand og verdier opprettholdes, men alle widgets vil bygges på nytt. Dette betyr at det kun er endringer gjennom "build" funksjoner som kan sees ved "hot-reload", ved endringer utenfor dette kreves en "hot-restart". Dette gjelder f.eks. endring i temafarger og konstanter.

Skru på Push-notifikasjoner for iOS
For å gi tillatelse for push-notifikasjoner på iOS er instruksjoner for dette her:
https://firebase.flutter.dev/docs/messaging/apple-integration.

> Obs: Push-Notifikasjoner fungerer **IKKE** på iOS simulatorer, for å teste om push-notifikasjonen fungerer trengs derfor en fysisk iOS enhet.

Endre applikasjonsikonet
For å bytte ut applikasjonsikonet med ett nytt, erstatt **icon.png** i **wasteflow/assets** med det nye ikonet, sørg for at den nye filen også heter **icon.png**. Deretter bruk kommandoen:

```
/wasteflow > flutter pub run flutter_launcher_icons:main
```

Dette vil opprette alle bildefilene for begge operativsystemene. Når prosessen er fullført må applikasjonen lukkes og kjøre på nytt for å se det nye ikonet.

Oppstartsbilde (Splash Screen)
For å endre oppstartsbilde, bilde som vises mens applikasjonen åpnes, er instruksjonene for dette her:
https://flutter.dev/docs/development/ui/advanced/splash-screen.

8.5.2 Firebase Cloud Functions

For å gjøre endringer i sky-funksjonen gjøres endringene i filen **index.ts** i mappen **wasteflow/-firebase/functions/src**.

For å publisere endringene til skyen brukes kommandoen:

```
/wasteflow/firebase > firebase deploy --only functions
```

For å teste funksjonene lokalt før publisering er instruksjoner for dette her:
https://firebase.google.com/docs/functions/local-emulator.

8.6 Utgivelse

Under debugging kan applikasjonen virke treg og/eller hakkete, dette er fordi den kompilerte koden ikke er optimalisert. Ved å bygge applikasjonen i utgivelsesmodus vil applikasjonen oppleves som raskere, men vil ikke gi hjelp eller feilmeldinger dersom noe går galt.

For å bygge applikasjonen i utgivelses modus, ha en enhet koblet til og bruk kommandoen:

```
/wasteflow > flutter run --release
```

Obs: ikke alle enheter og emulatorer støtter å bygge i utgivelsesmodus.

8.7 Testing

Prosjektet inneholder ingen automatiske tester, dette er en svakhet ved løsningen, men teamet prioriterte integrasjonstesting fremfor enhetstesting. Dersom det er ønskelig å opprette enhetstester, som er anbefalt, opprettes disse i en egen mappe **test** i mappen **wasteflow**. Instruksjoner for dette finnes her:
https://flutter.dev/docs/testing.

9 Dokumentasjon av kildekode

9.1 Se dokumentasjon av kildekode

Dokumentasjon av kildekoden finnes her: https://folk.ntnu.no/sivertut/documentation/wasteflow.
Som nevnt tidligere ligger også alle dokumentasjonsfilene i mappen **wasteflow/doc/api**. Her kan man åpne dokumentasjone lokalt ved å åpne filen **index.html** i en nettleser, men her vil ikke "søk" funksjonen fungere.

9.2 Skrive dokumentasjon av kildekode

For å skrive dokumentasjon i **dart** brukes tredobbel skråtrek: "///". Du kan referere til andre klasser og biblioteker ved å bruke hakeparantes slik: "[KlasseNavn]", dette vil vises i editoren og opprette en kobling i dokumentasjonssidene.

9.3 Opprette dokumentasjon av kildekode

Opprettelse av dokumentasjon kan gjøres automatisk ved å bruke "dartdoc". For å installere dartdoc kan du i terminalen fra mappen **wasteflow** skrive kommandoen:

```
/wasteflow > pub activate dartdoc
```

For å generere ny dokumentasjon bruker du da kun kommandoen:

```
/wasteflow > dartdoc
```

Den nye dokumentasjonen vil da lagres i **wasteflow/doc/api**. Vær oppmerksom på at kommandoen ikke sletter den gamle dokumentasjonen.

Vedlegg D Testplan

For å se hvor brukervennlig applikasjonen er etter utviklingen som er gjort under dette prosjektet, utføres brukertester. Hensikten er å hente inn data om hvorvidt brukerne i målgruppe hadde benyttet seg av tjenesten om den var tilgjengelig på markedet. I tillegg få generell tilbakemeldinger på hvordan applikasjonen er å bruke og om de føler noe mangler. Under er testplanen beskrevet i mer detalj.

D.1 Testplan

1. **Formålet med brukertestene:** Teste brukervennligheten og om brukere kan navigere seg igjennom appen uten komplikasjoner. Finne ut om tjenesten er noe de ville brukt om de hadde muligheten.
2. **Funksjonaliteten som skal testes:**
   - Legge inn bestilling
   - Endre bestilling
   - Avlyse bestilling
   - Feedback funksjonen
   - Opprette bruker
3. **Systemet som skal testes:** Wasteflow, applikasjonen som er utviklet i forbinelse med faget *TDAT3001 Bacheloroppgave Dataingeniør*.
4. **Testbrukerne:** I Visjonsdokumentet(se vedlegg A) er studenter nevnt som målgruppen så de vil også være testbrukerne vi bruker.
5. **Testlokalet:** Grunnet korona vil testingen foregå i kollektivet til teamet.
6. **Testutstyr:** Under testingen trenger brukerne en mobiltelefon, denne vil være en av teamet sine.
7. **Oppgavene som gis til brukerne:** Se avsnitt D.2
8. **Timeplan:** Det vil bli satt av 15 minutter for hver bruker, der spørsmålene og oppgavene skal gjennomføres.
9. **Spørsmål til brukerne før og etter testen:** Se avsnitt D.3
10. **Testteamet:** Teamet som skriver hovedrapporten og utvikler Wasteflow er også testteamet.
11. **Funnene i testen:** Disse vil diskuteres i hovedrapporten
12. **Videre aksjonspunkter for funnene:** Disse vil være opp til oppdragsgiver, og de som skal videreutvikle Wasteflow, å ta tak i siden dette vil være utenfor teamets tidsramme for prosjektet. Videre arbeid med applikasjonen vil også diskuteres i hovedrapporten for oppgaven.

D.2 Oppgaver

- Lag en bruker
- Legg inn en bestilling for henting som består av:
  - En sofa som har målene: bredde 2 m x Dybde 0,9 m x Høyde 0,8 m og vekt 40kg
  - En vaskemaskin som har målene: bredde 0,6m x dybde 0,5m x høyde 0,8m og vekt 67 kg

  - Ei eske med bøker med målene: bredde 0,3m x dybde 0,6m x høyde 0,4m og vekt 30 kg
  - Bestillingen skal hentes 20. mai 2021 klokka 18:00
- Gi en generell tilbakemelding på hva du synes om opplevelsen din i appen.
- Finn frem bestillingen i oversikten over dine bestillinger
- Endre din bestilling.
- Kanseller din bestilling
- Logg ut

D.3 Spørsmål

- **Stilles før testen gjennomføres**
  - Har du brukt en lignende tjeneste tidligere?
  - Har du noen gang trengt å levere avfall til en gjenvinningsstasjon før?
    * Hvis ja, hvordan ble det gjort?
- **Stilles etter testen er gjennomført**
  - Om du skulle levert avfall til gjenvinningsstasjon hadde du brukt tjenesten du nettopp testet?
    * Hvis nei, hvorfor?
    * Hvis ja, hvorfor?
  - Føler du at noe mangler?

Vedlegg E Samtykkeerklæring

Samtykkeerklæring for applikasjonstesting

Hva skal testes?

Brukeren, altså deg, vil være med på å teste en mobilapplikasjon som er under utvikling. Applikasjonen er laget som et alternativ for å få fraktet avfall til gjenvinningsstasjoner. Tanken er at applikasjonen skal gjøre denne prosessen lettere for personer som skal levere avfallet sitt.

Hva innebærer din deltagelse?

Du vil bli observert mens du blir bedt om å utføre en rekke oppgaver inne i applikasjonen. Disse observasjonene vil bli brukt i en rapport som inngår i faget TDAT3001 Bacheloroppgave Dataingeniør ved NTNU. Det vil også bli foretatt et intervju både før og etter oppgavene er utført som også vil inngå i den samme rapporten. Alle observasjonene og intervjudataen som hentes i forbindelsen med prosjektet vil være fullstendig anonymt.

☐ *Jeg samtykker at observasjoner og intervjudata kan brukes i rapporten.*

Sted og dato

Deltakers signatur

Vedlegg F Brukertester

Brukertest 1

,

Intervju før testing

- **Har du brukt en lignende tjeneste tidligere?**
    - Nei har ikke det.
- **Har du noen gang trengt å levere avfall til en gjenvinningsstasjon før?**
    - **Hvis ja, hvordan ble det gjort?**
    - Ikke jeg personlig, men mine foreldre har nok gjort det.

Observasjoner

- **Lag en bruker**
    - Brukeren skjønner at de skal fylle inn telefonnummeret for å logge inn, og at det kommer en kode på mobilen
    - Brukeren kommer seg inn uten noen problemer
- **Legg inn en bestilling for henting som består av:**
    - **En sofa som har målene: bredde 2 m x Dybde 0,9 m x Høyde 0,8 m og vekt 40kg**
    - **En vaskemaskin som har målene: bredde 0,6m x dybde 0,5m x høyde 0,8m og vekt 67 kg**
    - **Ei eske med bøker med målene: bredde 0,3m x dybde 0,6m x høyde 0,4m og vekt 30 kg**
    - **Bestillingen skal hentes 20. mai 2021 klokka 18:00**
        - Brukeren finner frem til bestillingssiden uten hjelp.
        - Brukeren finner frem til hvor man skal legge inn avfall selv og får lagt inn alle verdiene uten spørsmål.
- **Gi en generell tilbakemelding på hva du synes om opplevelsen din i applikasjonen.**
    - Brukeren klarer å navigere til riktig siden, men må lete litt for å komme frem.
    - Brukeren leser ikke helt teksten som sier at å legge igjen en mail adresse er frivillig, men etter oppklaring fra testansvarlig blir det tydelig.
- **Finn frem bestillingen i oversikten over dine bestillinger**
    - Finner frem og er innom begge sidene for bestillingsoversikt før de til slutt velger den inne på profilsiden. Kanskje et litt dårlig formulert spørsmål fra testteamet sin side, siden det er to mulige alternativer.
- **Endre din bestilling.**
    - **Fjern en av tingene som skal hentes, deretter lagre endringen.**
        - Brukeren gjør dette uten hjelp, men stusser litt på at de må gå til betaling igjen.
- **Kanseller din bestilling**
    - Brukeren finner frem og nevner at de la merke til det tidligere på grunn av den røde knappen.
- **Logg ut**

- Brukeren får til dette uten problemer

Intervju etter testing

- **Om du skulle levert avfall til gjenvinningsstasjon hadde du brukt tjenesten du nettopp testet?**
  - **Hvis nei, hvorfor?**
  - **Hvis ja, hvorfor?**
    - Fordi noen situasjoner har man dårlig tid og man bruker jo å få ting på døren mer og mer nå til dags, da syns jeg at dette er en naturlig addisjon til denne hverdagen. Vil heller bruke tid på andre ting enn å levere avfall.
- **Har du noen andre tilbakemeldinger?**
  - Jeg synes den var veldig lett håndterlig og ligner på apper som man har brukt før. Logisk oppbygging av navigasjon, som gjorde det enkelt å finne frem. Det var strukturert og lett å se ting.

Brukertest 2

Intervju før testing

- **Har du brukt en lignende tjeneste tidligere?**
    - Nei
- **Har du noen gang trengt å levere avfall til en gjenvinningsstasjon før?**
    - **Hvis ja, hvordan ble det gjort?**
    - Ikke, meg personlig, men har hjulpet andre å levere deres avfall tidligere.

Observasjoner

- **Lag en bruker**
    - Brukeren ser ikke at de må bekrefte koden for at den skal sendes, og må ha hjelp for å fullføre.
    - Brukeren trodde bekreftelsen var der om koden ikke kom automatisk.
    - Etter oppklaring fra testansvarlig kommer brukeren seg videre uten flere problemer.
- **Legg inn en bestilling for henting som består av:**
    - **En sofa som har målene: bredde 2 m x Dybde 0,9 m x Høyde 0,8 m og vekt 40kg**
    - **En vaskemaskin som har målene: bredde 0,6m x dybde 0,5m x høyde 0,8m og vekt 67 kg**
    - **Ei eske med bøker med målene: bredde 0,3m x dybde 0,6m x høyde 0,4m og vekt 30 kg**
    - **Bestillingen skal hentes 20. mai 2021 klokka 18:00**
        - Brukeren bruker ikke knappen for å bruke nåværende lokasjon som henteadressen, etter testingen ble brukeren spurt om de la merke til knappen, det hadde de ikke gjort.
        - Brukeren har ikke noen problemer med å legge inn gjenstandene, adressen eller tidspunktet.
- **Gi en generell tilbakemelding på hva du synes om opplevelsen din i applikasjonen.**
    - Brukeren kommer seg ikke av bestillingsbekreftelsen med det første og virker usikker på om bestillingen er bekreftet, etter bekreftelse fra testansvarlig at bestillingen ert gått inn kommer brukeren ut av siden.
    - Brukeren navigere seg til siden for å gi tilbakemelding uten noen problemer.
- **Finn frem bestillingen i oversikten over dine bestillinger**
    - Brukeren finner frem oversikten uten noen hinder
- **Endre din bestilling.**
    - **Fjern en av tingene som skal hentes, deretter lagre endringen.**
        - Brukeren stusser over at de må betale på nytt og mener dette ikke var veldig logisk.
        - Etter disse tilbakemeldingene får brukeren bekreftet endringene.

- **Kanseller din bestilling**
    - Brukeren finner raskt frem til kanselleringsknappen og har ingen problemer
- **Logg ut**
    - Brukeren får dette til uten problemer

Intervju etter testing

- **Om du skulle levert avfall til gjenvinningsstasjon hadde du brukt tjenesten du nettopp testet?**
    - **Hvis nei, hvorfor?**
    - **Hvis ja, hvorfor?**
        - Ja, om jeg ikke hadde hatt henger eller mulighet for å frakte det selv og om det var ikke for dyrt. Usikker på om jeg hadde brukt tjenesten om jeg hadde hatt en egen henger.
- **Har du noen andre tilbakemeldinger?**
    - Var litt vanskelig å skjønne om bestillingen var lagt inn etter den var fullført. Hadde kanskje vært lettere om det var en ok knapp.
    - Ellers meget oversiktlig app ellers.

Brukertest 3

Intervju før testing

- **Har du brukt en lignende tjeneste tidligere?**
    - Ikke noe bestillingsgreier nei
- **Har du noen gang trengt å levere avfall til en gjenvinningsstasjon før?**
    - **Hvis ja, hvordan ble det gjort?**
        - Ja, med henger der jeg leverte selv.

Observasjoner

- **Lag en bruker**
    - Brukeren har ikke noen problemer meg å komme seg inn.
- **Legg inn en bestilling for henting som består av:**
    - **En sofa som har målene: bredde 2 m x Dybde 0,9 m x Høyde 0,8 m og vekt 40kg**
    - **En vaskemaskin som har målene: bredde 0,6m x dybde 0,5m x høyde 0,8m og vekt 67 kg**
    - **Ei eske med bøker med målene: bredde 0,3m x dybde 0,6m x høyde 0,4m og vekt 30 kg**
    - **Bestillingen skal hentes 20. mai 2021 klokka 18:00**
        - Brukeren bruker ikke knappen for å hente adressen, i ettertid ble det avklart at de ikke så at det var noe knapp med den funksjonaliteten
        - Brukeren trenger hjelp for å ferdigstille bestillinga, når de kommer til betalingsskjermen blir dem forvirret siden det er ferdig utfylt (men dette kan ha noe med at betalingsfunksjonaliteten ikke er integrert og at det bare har en placeholder de må klikke seg igjennom)
- **Gi en generell tilbakemelding på hva du synes om opplevelsen din i applikasjonen.**
    - Brukeren finner frem til tilbakemeldingene ganske rask og uten hjelp
- **Finn frem bestillingen i oversikten over dine bestillinger**
    - Brukeren trenger ikke hjelp og finner raskt frem
- **Endre din bestilling.**
    - **Fjern en av tingene som skal hentes, deretter lagre endringen.**
        - Brukeren trenger ikke hjelp og finner raskt frem
- **Kanseller din bestilling**
    - Brukeren trenger ikke hjelp og finner raskt frem
- **Logg ut**
    - Brukeren trenger ikke hjelp og finner raskt frem

Intervju etter testing

- **Om du skulle levert avfall til gjenvinningsstasjon hadde du brukt tjenesten du nettopp testet?**
    - **Hvis nei, hvorfor?**
    - **Hvis ja, hvorfor?**
        - Det hadde jeg nok, for det er rimelig komplisert å komme seg til gjenvinningsstasjon om man ikke har bil eller kan kjøre med henger
        - Slipper bryet og det er lettvint for folk som studenter, eldre mennesker eller folk som bor i by generelt
- **Har du noen andre tilbakemeldinger?**
    - Ganske oversiktlig og ligner andre apper så var enkelt å skjønne hva jeg kunne trykke på.

Brukertest 4

Intervju før testing

- **Har du brukt en lignende tjeneste tidligere?**
    - Nei
- **Har du noen gang trengt å levere avfall til en gjenvinningsstasjon før?**
    - **Hvis ja, hvordan ble det gjort?**
        - Jeg har vært med andre, da kjørte vi selv med tilhenger så måtte vi stå i kø. Vi ventet vel i kø i 2 timer.

Observasjoner

- **Lag en bruker**
    - Brukeren kommer seg uten problemer eller hjelp
- **Legg inn en bestilling for henting som består av:**
    - **En sofa som har målene: bredde 2 m x Dybde 0,9 m x Høyde 0,8 m og vekt 40kg**
    - **En vaskemaskin som har målene: bredde 0,6m x dybde 0,5m x høyde 0,8m og vekt 67 kg**
    - **Ei eske med bøker med målene: bredde 0,3m x dybde 0,6m x høyde 0,4m og vekt 30 kg**
    - **Bestillingen skal hentes 20. mai 2021 klokka 18:00**
        - Brukeren har ikke noen problemer når bestillingen skal legges inn, trenger verken hjelp eller nærmere forklaring.
- **Gi en generell tilbakemelding på hva du synes om opplevelsen din i applikasjonen.**
    - Brukeren må lete litt, men finner til slutt frem uten hjelp.
- **Finn frem bestillingen i oversikten over dine bestillinger**
    - Brukeren trenger ikke hjelp og finner frem raskt
- **Endre din bestilling.**
    - **Fjern en av tingene som skal hentes, deretter lagre endringen.**
        - Brukeren har ingen problemer og løser oppgaven raskt.
- **Kanseller din bestilling**
    - Brukeren har ingen problemer og løser oppgaven raskt.
- **Logg ut**
    - Brukeren har ingen problemer og løser oppgaven raskt.

Intervju etter testing

- **Om du skulle levert avfall til gjenvinningsstasjon hadde du brukt tjenesten du nettopp testet?**
    - **Hvis nei, hvorfor?**
    - **Hvis ja, hvorfor?**
        - Ja, det virker veldig praktisk og måtte fikse det selv i en ellers travel hverdag.
- **Har du noen andre tilbakemeldinger?**
    - Synes det var veldig bra.

Brukertest 5

Intervju før testing

- **Har du brukt en lignende tjeneste tidligere?**
    - Ikke brukt lignende tjenester før, har heller ikke hørt om noe lignede tjeneste.
- **Har du noen gang trengt å levere avfall til en gjenvinningsstasjon før?**
    - **Hvis ja, hvordan ble det gjort?**
        - Da leide vi en tilhenger så kjørte vi det selv / eller bare fått plass i bilen og kjørt selv bort

Observasjoner

- **Lag en bruker**
    - Brukeren fyller inn telefonnummer og får logget seg inn
    - Brukeren utrykker at de
- **Legg inn en bestilling for henting som består av:**
    - **En sofa som har målene: bredde 2 m x Dybde 0,9 m x Høyde 0,8 m og vekt 40kg**
    - **En vaskemaskin som har målene: bredde 0,6m x dybde 0,5m x høyde 0,8m og vekt 67 kg**
    - **Ei eske med bøker med målene: bredde 0,3m x dybde 0,6m x høyde 0,4m og vekt 30 kg**
    - **Bestillingen skal hentes 20. mai 2021 klokka 18:00**
        - Brukeren fyller først ut adressen til tross for at utfylling av avfall kommer før på skjermen.
        - Brukeren finner ikke frem til hvor de kan legger beskrivelse uten hjelp.
        - Brukeren får fint til å fylle ut adresse, dato og sende inn bestilling uten hjelp.
- **Gi en generell tilbakemelding på hva du synes om opplevelsen din i applikasjonen.**
    - Brukeren må trykke litt rundt før de finner tilbakemeldingen, men finner til slutt frem og får sendt inn.
    - Har litt problemer med å skjønne hvilken mail som skal skrives inn i feltet som kan fylles inn med brukerens mail.
    - Brukeren får ikke bort tastaturet, noe som er en bug vi kjenner til.
- **Finn frem bestillingen i oversikten over dine bestillinger**
    - Brukeren trenger ikke hjelp til å finne frem
- **Endre din bestilling.**
    - Fjern en av tingene som skal hentes, deretter lagre endringen.
        - Brukeren trenger ikke hjelp for å få til dette
- **Kanseller din bestilling**
    - Brukeren trenger ikke hjelp til å finne frem

- **Logg ut**
    - Brukeren trenger ikke hjelp til å finne frem

Intervju etter testing

- **Om du skulle levert avfall til gjenvinningsstasjon hadde du brukt tjenesten du nettopp testet?**
    - **Hvis nei, hvorfor?**
    - **Hvis ja, hvorfor?**
        - Veldig lettvint måte å bli kvitt søppel på. Enkel tjeneste å bruke. Krevde ikke noe komplisert innlogging, kan være billigere enn å gjøre det selv
- **Har du noen andre tilbakemeldinger?**
    - Har den funksjonaliteten den trenger.
    - Litt vanskelig å se hvor man skulle legge inn avfallet i en bestilling.

Sabine Seljeseth og Sivert Utne

Wasteflow - Mobilapplikasjon for avfallshenting

NTNU
Kunnskap for en bedre verden