Bacheloroppgave

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

Aleksander Aaboen, Martin Iversen og Tormod Mork Müller

Sporing av stillas

Bacheloroppgave i Bachelor i ingeniørfag, data
Veileder: Frode Haug
Mai 2022

NTNU
Kunnskap for en bedre verden

MB STILLAS

Aleksander Aaboen, Martin Iversen og Tormod Mork Müller

Sporing av stillas

Bacheloroppgave i Bachelor i ingeniørfag, data
Veileder: Frode Haug
Mai 2022

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

NTNU
Kunnskap for en bedre verden

Sammendrag

Tittel : **Sporing av stillas**

Dato: 19.05.2022

Deltakere : Tormod Mork Müller
Aleksander Aaboen
Martin Iversen

Veileder: Frode Haug

Oppdragsgiver : Mylift & Borud Stillas, Knut Rindal

Stikkord/nøkkelord: API, IoT, Stillas, Sporing, iOS

Antall sider/ord : 135 sider
Antall vedlegg : 13
Publiseringsavtale inngått : Åpen

Ettersom logistikk utgjør en stor del av en produserende og transporterende virksomhets lønnsomhet, implementeres det nå IT-tekniske løsninger for logistikk og oversikt i stadig flere bransjer. MB-Stillas kom til oss med et ønske om å finne en sporingsteknologi, samt utvikle et IT-system for å fikse opp i et logistisk kaos ved utkjøring og behandling av stillasdeler. Løsningen er å plassere BLE-tagger på individuelle stillasdeler og å samle dataen fra disse med Gateway-er som kommuniserer dataen til et API ved hjelp av LPWAN. Et API og en dokumentdatabase er utviklet for back-end løsningen. To front-end løsninger er utviklet: En webløsning for administrative oppgaver som registrering av prosjekter og stillasdeler, og en iOS applikasjon for montører ute i felt for å forenkle og loggføre overføringer av stillasdeler. API-et er deployert på OpenStack og prototypen er fungerende som et konseptbevis. Den agile utviklingsmetodologien SCRUM har blitt benyttet gjennom utviklingsfasen for å kunne tilpasse oss oppdragsgivers ønsker, samt en åpen oppgavebeskrivelse.

Summary

Title : **Sporing av stillas**

Date: 19.05.2022

Authors: Tormod Mork Müller
Aleksander Aaboen
Martin Iversen

Supervisors: Frode Haug

Employer : Mylift & Borud Stillas, Knut Rindal

Keywords: API, IoT, Stillas, Sporing, iOS

Number of pages : 135 sider
Attachments: 13
Availability : Open

As logistics plays a large part of a manufacturing and transporting company's profitability, IT technical solutions for logistics and overview are now being implemented in an increasing amount of industries. MB-Stillas approached us with a desire to find a tracking technology, as well as develop an IT system to fix their logistical chaos that occurs when transporting and renting scaffolding parts. The proposed solution is to place BLE tags on individual scaffolding parts to collect data from these with Gateways that communicate the data to an API using LPWAN. An API as well as a document database have been developed as back-end solutions. The front-end solutions that have been developed is: A web solution for administrative tasks such as registration of projects and scaffolding parts, and an iOS application for scaffolding fitters in the field to simplify and log transfers of scaffolding parts. The API is distributed on OpenStack and the prototype serves as a proof of consept. The agile development methodology SCRUM has been used throughout the development phase to be able to adapt to the client's wishes and needs, as well as an open task description.

Forord

Oppgaven er skrevet av *Tormod Mork Müller, Aleksander Aaboen og Martin Iversen* ved *Institutt for datateknologi og informatikk* ved NTNU i Gjøvik. Gruppen ønsker å takke alle som har vært involvert i oppgaven. Vi vil takke gruppens veileder *Frode Haug* for god veiledning. Han har gjennom ukentlige møter gitt gruppen verdifull veiledning og tilbakemelding angående oppgaven. Gruppen ønsker også å takke oppdragsgiver *MB-Stillas*, da særlig *Knut Rindal*. Han har fra oppstart av prosjektet vist stort engasjement og interesse for gruppens arbeid og har vært en kilde til inspirasjon for oss. Vi vil også takke faglærer *Rune Hjelsvold* for rådgivning angående valg av database. Avslutningsvis vil vi takke *Telia* og *ABAX* for engasjement i oppgaven samt rådgivning angående løsning av sporing.
Dette har vært en spennende og lærerik oppgave som har gitt oss et uvurderlig læringsutbytte.

Innhold

Figur-liste

Kode-utdrag

Tabell-liste

1. Innledning

Stillas er en arbeidsplattform som er satt opp midlertidig for å forenkle arbeid på en byggeplass. Delene blir transportert mellom lager, byggeplass og mellom byggeplasser. Utstyret kontrolleres sjeldent før det sendes ut til byggeplassen, ei eller når det returneres til lager. Tap av utstyr er tidkrevende og en uønsket kostnad for bedrifter i bygge-bransjen. Ved å bruke tilgjengelig teknologi ønsker oppdragsgiver å minimere tap av utstyr, få kontroll på hvor utstyret befinner seg, samt frigjøre ressurser til andre arbeidsoppgaver.

1.1.1 Problemområde

Det moderne samfunnet utvikler seg i stor fart og teknologi-bølgen blir større og større. Det produseres og kjøpes nye produkter, og logistikken i hva man eier og hvor det befinner seg blir mer uklart. Derfor har behovet for å holde orden på og ha muligheten til å kunne spore gjenstander økt. Ettersom logistikk utgjør en stor del av en produserende og transporterende virksomhets lønnsomhet, implementeres det nå IT-tekniske løsninger for logistikk og oversikt i stadig flere bransjer. Datateknologiske løsninger for sporing og logistikk er raskere, mer oversiktlige, enklere i bruk og ikke minst langt mer lønnsomme enn manuelle, menneskelige løsninger[1]. Disse teknologiene har fått paraplybetegnelsen «IoT (Internet of Things)».

1.1.2 Avgrensning

Oppdragsgiver *Mylift & Borud stillas AS (MB-Stillas)* er en totalentreprenør som leverer montering, frakt og leie av stillas innenfor bygg- og anleggs-bransjen (1). De har kommet til NTNU med et problem som ligger innenfor logistikk, nærmere bestemt sporing og telling av stillas på byggeplass og lager. Oppdragsgivers problem resulterer i svinn, bortkastet tid og et logistisk kaos ved utkjøring og behandling av stillasdeler. De ønsker derfor en løsning som gjør sporing og logistikk angående utleie av stillas mulig. Gruppen skal gå inn på områder innenfor logistikk, sporing og IoT teknologier. Gruppen ønsker å besvare hvordan *MB-Stillas* kan spore stillasdeler med beste tilgjengelige teknologi.

Oppgaven tar for seg hvordan man kan integrere moderne teknologi inn i en teknologisk understimulert bransje gjennom sporing av stillas på lager og ute på byggeplassen. Gruppens fokus vil ligge i en digital løsning. Utover dette vil løsningen også inkludere en rapport på marked og muligheter innenfor sporing av billige verdier, samt en anbefaling om hvilken sporingsteknologi som er best egnet oppdragsgivers ønsker i skrivende stund.

[1] Et estimat gjort av *World Economic Forum* anslår at i løpet av perioden 2016-2025 vil logistikk i industrien oppleve økt lønnsomhet ved hjelp av digitalisering (58)

1.1.3 Oppgavedefinisjon

Oppgaven tar for seg frigjøring av mest mulig ressurser ved å automatisere stillaslogistikkprosessen til oppdragsgiver. For å oppnå dette skal gruppen:

- Finne en kostnadseffektiv måte å spore enkelte stillasdeler.
  - Finne den mest egnede sporingsteknologien.
  - Finne en sporingsløsning som passer problemstillingen.
- Spore stillasdeler på tvers av byggeprosjekter.
- Sporingsteknologien som benyttes skal være:
  - Bærekraftig
  - Fremtidsrettet
  - Robust
  - Av lav kostnad
- Telle stillas som transporteres inn og ut av lager.
- Loggføre forflytning av stillas.
- Registrere og fjerne stillas fra prosjekter.

Gruppen skal undersøke diverse sporings-muligheter som kan tas i bruk for å løse problemet og diskutere med stillas produsenter og oppdragsgiver for å kunne danne et grunnlag for valget som vil bli tatt angående sporings-teknologi og enhet.
Det skal utvikles en løsning som gjør følgende:

- Ansatte skal ha tilgang til informasjon om byggeprosjekter og stillasdeler uavhengig av arbeidsplass.
- Funksjonaliteten skal bli implementert ved hjelp av et robust API.
- Løsningen skal kunne videreutvikles og/eller integreres i oppdragsgivers andre IT-løsninger.
- Vise alle selskapets stillas i et kart på en enkel måte.

1.2 Mål

Gruppen har for oppgaven definert forskjellige mål og vil først diskutere rapporten og oppgavens målgruppe, før effektmål, resultatmål og læringsmål vil bli definert. Gruppen ønsket gjennom oppgaven å oppfylle så mange av disse målene som mulig. Resultatet av dette vil bli diskutert i kap. 10.1.

1.2.1 Målgruppe

Bacheloroppgaven skrives for *MB-Stillas*. Produktet vil være rettet mot oppdragsgiver og deres problemstilling innenfor logistikk rundt stillassporing. Rapporten retter seg derimot mot en person med tilsvarende kunnskap av en student på femte semester dataingeniør som kjenner til grunnleggende begreper og fagfelt innenfor Telematikk og Informatikk.

1.2.2 Effektmål

Om bacheloroppgaven løses på en god måte vil oppgaven gi oppdragsgiver en løsning som sparer selskapet tid og penger. *MB-Stillas* vil kunne få en oversikt over hvor mange stillasdeler som befinner seg på byggeprosjekter. Dette vil forbedre selskapets logistikk betraktelig gitt at

selskapet i dag ikke har en løsning eller et verktøy som tilbyr dette. I tillegg vil oppdragsgiver få en dypere forståelse for dagens IoT marked, de tilgjengelige teknologiene og deres fordeler/ulemper. Videre vil de få innsikt i hva en oppskalering av teknologien som blir brukt i bacheloroppgaven vil koste. På bakgrunn av dette setter gruppen opp følgende effektmål:

- Frigjøre ressurser hos oppdragsgiver både i form av tid, ansatte og penger ved å redusere tap av stillasdeler.
- Gi oppdragsgiver en prototype de kan arbeide videre med om resultatet tilsier at det vil være lønnsomt.
- Gi oppdragsgiver et bilde av dagens IoT marked og de tilgjengelige teknologiene som kan brukes til å løse problemstillingen.

1.2.3 Resultatmål

Det skal utvikles programvare med følgende funksjonalitet:

- Gi ansatte tilgang til informasjon om stillasdeler uavhengig av arbeidsplass.
- Gi informasjon om stillasdeler og byggeprosjekter ved hjelp av et API
- Løsningen skal kunne videreutvikles og/eller integreres i oppdragsgivers andre IT-løsninger.
- Det skal leveres en mobil og web applikasjon.
- Programvaren skal vise selskapets stillasdeler i et kart slik at oppdragsgiver ser hvor mange stillasdeler som er på forskjellige prosjekter.

Oppdragsgiver vil få:

- En sporingsrapport som presenterer tilgjengelige sporingsteknologier satt opp imot problemstillingen.
- Et API som oppfyller funksjonaliteten i listen over.
- En web-applikasjon som tar i bruk API-et.
- En mobil-applikasjon som tar i bruk API-et.

Viktige momenter ved oppgaven:

- Sporingen må fungere uavhengig av klima.
- Se hvilke byggeprosjekter som har hvilket stillas.
- Frigjøring av ressurser, sporingen må inneholde minimum av humankapital.
- Lang levetid på enheten.
- Bruk med mobil, stillasdeler vises i et kart.
- API slik at neste fase og oppkobling av flere tjenester er mulig.

1.2.4 Læringsmål

Under oppstart av bacheloroppgaven satt gruppen seg følgende læringsmål:

- Få bedre innsikt innenfor teknologi i bygge-bransjen og deres arbeidsmetoder.
- Tilegne seg kunnskap innenfor relevant maskinvare og teknologi.
- Anvende relevant kunnskap for å løse teoretiske og praktiske problemstillinger.
- Planlegge, gjennomføre og dokumentere et ingeniørfaglig arbeid.
- Formidle faglig kunnskap til ulike målgrupper skiftelig og muntlig.
- Arbeide og kommunisere med en reel oppdragsgiver.

1.3 Egen bakgrunn og kompetanse

Gjennom fem semestre på dataingeniør har gruppen tilegnet seg kunnskap om programmering og utvikling. Gruppen har et godt fundament for objektorientert programmering og programvare- og systemutvikling. Gruppen har derimot ingen spisskompetanse da studiets oppbygning har fungert som en introduksjon til flere av de sentrale fagfeltene innenfor informatikk. Medlemmene har noe splittet kompetanse grunnet diverse valgfag, noe som styrker gruppens mangfold med hensyn på kompetanse. De mest sentrale punktene vi har kompetanse innenfor er server-vedlikehold, database, API utvikling, brukerdesign og mobil- og web-utvikling.

1.4 Tilegnet kunnskap

Gjennom bacheloroppgaven har gruppen måttet tilegne seg kunnskap rundt diverse nye temaer. Tidlig i prosessen ble store deler av tiden forvaltet på å undersøke de forskjellige sporingsteknologiene som er tilgjengelige på markedet i dag. Dette ble gjort for å kunne informere og diskutere med oppdragsgiver rundt hvilke teknologier som var praktisk gjennomførbare. Videre dannet dette også grunnlaget for hvordan design av database og API skulle gjennomføres ettersom ulike teknologier førte til ulike design behov.

Gruppen har måttet sette seg inn i database-design med hensyn på en dokument-orientert database og hvordan dette bør implementeres. Kompetansen innenfor mobilprogrammering var begrenset til teori og utvikling innenfor Android. Ettersom oppdragsgiver ønsket en iOS applikasjon måtte denne kunnskapen tilegnes underveis. Gruppemedlemmer som ikke var kjent med rammeverkene som brukes under utvikling av en mobil og web applikasjon måtte også tilegne seg kunnskap i disse fagfeltene. Avslutningsvis har gruppen måttet sette seg inn i formatering og behandling av data fra en BLE Gateway, hvordan BLE tags kommuniserer med Gateway-en, samt hvordan denne dataen kommuniseres til et API og databasen.

1.5 Rammer

Gruppen har et tidsrom fra den 11. januar 2022 til 20. mai 2022 på å gjennomføre oppgaven. Gitt dette tidsrommet så gruppen det som essensielt å tydelig definere rammer som avgrenser oppgavespesifikasjonen gitt av oppdragsgiver. Gruppen satt 31. april som tidspunkt for utviklingsstopp og etter denne datoen lå fokuset på rapporten. Gruppen har ingen satte teknologirammer gitt av oppdragsgiver og sto derfor fritt til å sette disse selv ved oppgavestart. Oppdragsgiver var tydelig på at han ønsket «nye øyne» på problemstillingen og ga derfor gruppen frihet til å sette egne rammer.

Etter diskusjon med oppdragsgiver ble følgende rammer satt:

- Det er essensielt at både mobil-applikasjonen og web-løsning har et enkelt brukergrensesnitt slik at montører på byggeplass kan ta appen i bruk uten problemer.
- Løsningen skal spore følgende stillasdeler: Bunnskrue, Diagonalstang, Enrørsbjelke, Lengdebjelke, Plank, Rekkverksramme, Spir, Gelender, Stillaslem og Trapp.
- Fokuset skal ligge på å forbedre selskapets logistikk, aktiv sporing er ikke målet.

- Den fysiske sporingen skal bruke teknologien gruppen mener er best egnet med tanke på fremtiden og bruksområde, ikke pris.

Gruppen valgte å dele oppgaven i to. Under oppstart var oppgavens fokus å finne en egnet sporingsteknologi som kunne passe oppdragsgivers kravspesifikasjon, denne perioden skulle vare fra oppstart (11. januar) til leveranse av prosjektplanen (31. januar).

Etter denne delen tok gruppen i bruk «SCRUM» metodikk og brukte denne for å jobbe seg iterativt gjennom kravspesifikasjonen (kap. 3.1): Gruppen lagde under oppstart en liste med alle krav de selv hadde til oppgaven og brukte denne for å sette ukentlige SCRUM mål som gruppen skulle fullføre. Disse sprintene gikk ukentlig før gruppen den 30. april avsluttet utviklingsfasen og satt søkelys på hovedrapporten.

Oppdragsgiver ga ikke gruppen tilgang til noen form for maskin- eller programvare da alt av «IT-Kompetanse» i selskapet har vært innleid ved tidligere prosjekter. På grunn av dette hadde gruppen mer eller mindre frie tøyler angående hvordan vi ønsket å avgrense/spesifisere oppgaven. Gruppen bestemte tidlig at programvare skulle være oppgavens fokus gitt at dette er gruppens fagfelt, samt at dette ville samsvare med læringsmålene til Bacheloroppgaven.

1.6 Øvrige roller

Gruppens veileder er Frode Haug. Veileder tilbudte gruppen rådgivning i form av ukentlige møter i kombinasjon med skriftlige tilbakemeldinger på hovedrapporten og andre arbeidsprosesser.

Gruppens oppdragsgiver fungerte som en sparringspartner tidlig i prosessen for å kunne gi gruppen en tydelig kravspesifikasjon med ønsket funksjonalitet. Oppdragsgiver tilbudte seg å komme med tilbakemeldinger på de aspektene ved prosjektet de hadde kompetanse til å vurdere. Der oppdragsgiver ikke har kompetanse har gruppen blitt tilbudt konsultasjon fra *Armada Dynamics AS* et IT-selskap oppdragsgiver har arbeidet med tidligere.

Igjennom bacheloroppgaven var gruppen i kontakt med diverse selskaper, bedrifter og faglærere for å tilegne seg kunnskap, samt få rådgivning (se Appendix for møtereferater). Gruppen var hovedsakelig i kontakt med selskaper som leverer IoT løsninger, da spesielt bredbåndsleverandøren *Telia.* Selskapet ble involvert i oppgaven tidlig og skulle i utgangspunktet tilby gruppen SIM-kort, samt en sporingsenhet med LTE-M funksjonalitet. Gruppen har også vært i kontakt med *HAKI* som er oppdragsgivers hovedleverandør av stillas. Gruppen har brukt selskapet som kilde til informasjon om stillas, de forskjellige delene, produksjon, materialer, monteringsprosess, håndtering og annet.

1.6.1 Prosjektroller

For å distribuere ansvarsområder og sørge for at arbeidsoppgaver følges opp ble gruppen enig om roller og teamoppbygning. Innledningsvis ble det inndelt roller (se Appendix for prosjektplan), men etter hvert ble noen av rollene omrokert for å tilrettelegge for nye, uforutsette rollebehov, samt imøtekomme ulike ønsker internt for kunnskapstilegning.

- **Gruppeleder** – Martin Iversen
  - Overordnet ansvar for oppgavens gjennomføring.
  - Oversikt over prosjektets helhet for å sørge for at prosjektets prosesser går som planlagt.
  - Avgjørende stemme ved uenigheter i avgjørelser.
  - Ta hånd om konflikter internt i gruppen.

- **Dokumentasjonsansvarlig** – Martin Iversen
  - Sørge for at nødvendig dokumentasjon er til stede.
  - Oversikt over dokumentasjonsstruktur.

- **Kommunikasjonsansvarlig** – Varierende avhengig av tematikk
  - Kommunisere med eksterne parter.
  - Formidle informasjon mellom eksterne parter og gruppa, samt internt i gruppa.

- **Sporingsenhet ansvarlig** – Tormod Mork Müller
  - Anskaffe sporingsenhet for prototyping.

- **Mobilutvikling ansvarlig** – Tormod Mork Müller
  - Utvikling av mobilappen.
  - Ta avgjørende avgjørelser angående mobilappens funksjonalitet og utseende.

- **Webutvikling ansvarlig** – Aleksander Aaboen
  - Utvikling av web-løsningen.
  - Ta avgjørende avgjørelser angående web-løsningens funksjonalitet og utseende.

- **API og database utvikling** – Martin Iversen og Aleksander Aaboen
  - Hovedansvar for oppsett og konfigurering av databasen.
  - Ta avgjørende avgjørelser angående databasens design og implementasjon.

1.7 Organisering

Ved oppstart ble det skrevet en prosjektplan som inneholdt informasjon om organisering og planlegging av oppgaven. Denne planen inneholder ansvarsforhold, roller, samt en fremdriftsplan. Den fulle planen kan finnes i Appendix.

1.7.1 Oppbygning av rapporten

Nedenfor er en liste over inndelingen av kapitelene i bacheloroppgaven med en kort beskrivelse av hva de ulike seksjonene inneholder.

Kapittel 1 – Innledning
Dette kapittelet introduserer oppgaven og opplyser om overordnet informasjon rundt hvordan gruppen har valgt å gå frem for å løse problemstillingen. Kapittelet definerer også hvilke mål gruppen satte under planlegging av oppgaven.

Kapittel 2 – Teori
Dette kapittelet tar for seg teknologier og fagstoff som blir diskutert i rapporten. Kapittelet dekker de forskjellige sporingsteknologiene gruppen har undersøkt sammen med IoT teknologi generelt. Det fysiske stillaset blir også diskutert, i tillegg dekker kapittelet alle biblioteker brukt under utvikling og diskuterer «state of the art» innenfor diverse metoder og teknologier brukt.

Kapittel 3 – Kravspesifikasjon
Dette kapittelet vil definere alle kravene gruppen og oppdragsgiver har satt til oppgaven. Kapittelet vil bli brukt gjennom utviklingsprosessen for å kunne lage SCRUM sprinter. Kapittelet vil inneholde Use cases, user stories og annen dokumentasjon som ble brukt for å definere kravspesifikasjonen.

Kapittel 4 – Utviklingsprosess
Dette kapittelet vil ta for seg gruppens fremgangsmåte under utvikling, hvordan utviklingsprosessen ble planlagt. Kapittelet vil også ta for seg en oppsummering av hver SCRUM sprint og dokumentere hele utviklingsprosessen.

Kapittel 5 – Sporing
Kapittelet dokumenterer hvordan gruppen bestemte seg for sporingsteknologi og hvordan gruppen skaffet en aktuell sporingsenhet. Det diskuteres også hvilke resultater enheten ga.

Kapittel 6 – Back-End
Kapittelet diskuterer design og implementasjon av database og API i respektive kapitler. Det drøftes i begge kapitlene rundt resultatet, hva som fungerer, feil i systemet, samt hvordan resultatet måler opp mot målene gruppen satt i kapittel 3.

Kapittel 7 – Front-End
Kapittelet er strukturert på samme måte som kapittel 6, men tar for seg mobil og web applikasjonen gruppen har utviklet.

Kapittel 8 – Testing og sikkerhet
Kapittelet tar for seg all testing av systemet og vil deles opp i testing av API, Web og mobil applikasjon. Fysisk testing av sporingsenheten blir også dokumentert i dette kapittelet. Sikkerhetsaspekter systemet har og/eller burde hatt blir også drøftet her.

Kapittel 9 – Installasjon og realisering
Kapittelet diskuterer hvordan løsningen vil deployeres både programvare og hvordan fysisk sporingsenhet blir plassert på byggeplass, samt hvordan programvaren tåler skalering av data.

Kapittel 10 – Avslutning
Kapittelet vil diskutere hva gruppen kunne/burde gjort annerledes både i forhold til resultatet og arbeidsprosessen. Det diskuteres også hva som skal gjøres ved en eventuell oppfølgende bacheloroppgave/videre utvikling. Kapittelet diskuterer også bachelorprosessen spesifikt, hvordan gruppen selv mener at bachelorprosjektet ble løst. Følgende aspekter vil bli diskutert:

Innledningen av arbeidet, organisering av oppgaven, fordeling av arbeid, samt prosjektarbeid som helhet.

1.7.2 Terminologi

Gruppen vil bruke norsk språk i rapporten da oppdragsgiver, veileder og gruppemedlemmer alle er norske. Engelske begreper som ikke har vanlige norske definisjoner vil ikke bli oversatt (Back-End, Front-End, API er eksempler på ord som ikke vil bli oversatt). Akronymer vil bli forklart i Appendix.

1.7.3 Format

Rapporten vil skrives i skriftstørrelse 12 med skrifttype *Times New Roman* og linjeavstand 1,15. *Word*'s overskriftssystem brukes for å dele inn rapporten.

Gruppen tar i bruk *Word*'s integrerte referansesystem av typen *Dokument på webområde*, samt figur-liste, kodeliste og en liste over tabeller.

2. Teori

Dette kapittelet introduserer diverse teknologier gruppen har vurdert og/eller tatt i bruk under oppgaven. Det vil gås i dybden på hva teknologien er, og funksjonalitet vil forklares der gruppen ser det som hensiktsmessig. Kapittelet utdyper hovedsakelig temaene stillas, IoT, sporingsteknologier og diverse programmeringsbiblioteker brukt under utvikling.

2.1 Stillas

Siden sporingsteknologien skal tas i bruk på stillasdeler så gruppen det som hensiktsmessig å gi leseren en kort introduksjon til produktet stillas, hva det brukes til, samt hvilke deler gruppen har forholdt seg til:

Stillas består hovedsakelig av stål, men finnes også i aluminium. Stillas blir brukt som en midlertidig støttestruktur for både byggematerialer og personell. Oppdragsgiver frakter stillaset i «bareller» som er spesiallagde kasser laget for transport av stillas. Stillaset blir behandlet røft på byggeplass og kan stå uten ly i opptil flere måneder. Per i dag er det ikke noe system rundt sortering av stillasdeler. *MB-Stillas* sin produsent av stillas (*HAKI*) har gitt gruppen følgende deler:

- **Spir:** Stillasdelen brukes som støtte og er hovedkomponent for stillas med mer enn en etasje, brukes i all montering av stillas.
- **Lengdebjelke:** Stillasdelen brukes aktivt som støtte og bæring for plank og lemmer. Delen blir brukt på all stillasmontering uavhengig av antall etasjer og kvadratmeter.
- **Enrørsbjelke:** Delen brukes i alt oppsett uavhengig av etasje og størrelse.
- **Rekkverk:** Delen brukes aktivt på alt stillas og er meget sentral.
- **Plank:** Delen brukes under noen stillasmonteringer, men har i nyere tid blitt erstattet med lemmer, delen blir montert langs stillasets struktur
- **Lemmer:** Delen brukes ved de fleste monteringsjobber og har i nyere tid erstattet plank, i motsetning til plank monteres denne delen på tvers av stillaset.
- **Trapp:** Denne delen brukes ved montering av stillas med flere etasjer.
- **Gelender:** Denne delen brukes i kombinasjon med trapp, og er derfor ikke med på alle monteringsprosjekter.
- **Bunnskrue:** Denne delen brukes under all montering av stillas, den brukes som en grunnstøtte på hele stillasstrukturen.
- **Diagonalstenger:** Denne delen brukes for å støtte opp stillas, den blir brukt i de fleste monteringsjobber.

Figur 1 Stillasdeler, fra første rad venstre: 1.Diagonalstang, 2. Bunnskrue, 3.Enrørsbjelke, 4. Gelender, 5. Lengdebjelke, 6. Plank, 7. Rekkverk, 8. Spir, 9 Stillaslem, 10 Trapp

2.2 Internet of Things (IoT)

Gruppens implementasjon og løsning omhandler i stor grad temaet Internet of Things. På grunn av dette velger gruppen å introdusere både temaet og hvilke teknologier som faller innunder denne paraplybetegnelsen slik at leseren forstår de forskjellige teknologiene som blir diskutert under kap. 5.

IoT er en samlet betegnelse for gjenstander som kommuniserer med hverandre gjennom internettet. IoT åpner opp muligheter for å kunne fjernstyre og samle inn data fra enheter, som vi ikke hadde mulighet til tidligere. Enheter som kan gjøres smarte ved hjelp av IoT er eksempelvis kjøleskap, termostater, kameraer og lyspærer. IoT er skapt for å gjøre livene til brukerne enklere.

Konseptet ble introdusert på slutten av 90-tallet, der teknologien ble muliggjort av datamaskinene og det trådløse telenettverket som var tilgjengelig. Siden da har konseptet blitt videreutviklet, noe som har gjort mulighetene for IoT mer modent, og flere teknologier muliggjør nye løsninger. Disse teknologiene har spesielt blitt brukt innenfor sporing av gjenstander. Gruppen har gjennom bacheloroppgaven undersøkt følgende sporingsteknologier i sammenheng med IoT:

2.2.1 Low-Power Wide-Area Network (LPWAN)

IoT omhandler kommunikasjon med små enheter via internett, men i nyere tid har kommunikasjon via mobilnettet blitt mulig. Dette lar små IoT sensorer og Gateways kommunisere med programvare uten Wi-Fi. Dette ble derimot ikke mye brukt for små enheter da kostnadene ble for høye. IoT hadde behov for en lettvekts kommunikasjonsmetode som tok i bruk mobilnettet. Det er her LPWAN kommer inn. LPWAN gjør kommunikasjon med mange små IoT enheter over lange avstander mulig, grunnet en skreddersydd telekommunikasjonsprotokoll. LPWAN kan deles inn i to grupper: ulisensierte- og lisensierte tjenester. NB-IoT og LTE-M er lisensierte tjenester, NB-IoT leverer nettverkforbindelser med lav båndbredde og lavt batteri forbruk.

LTE-M er teknologi som er optimalisert for høyere båndbredde og mobile forbindelser. Teknologien er designet med fokus på enheter i bevegelse. LTE-M og NB-IoT er gunstige når det gjelder batteriforbruk, men NB-IoT er mer fordelaktig om batteriforbruk er første prioritet for løsning (2).

De ulisensierte tjenestene er Sigfox og LoRa. De to tjenestene leverer samme tjenester som LTE-M og NB-IoT, der det blir utnyttet lav båndbredde. Forskjellene mellom ulisensierte- og lisensierte tjenester er bruken av SIM-kort. Fordelene med ulisensierte tjenester er at de har lengre rekkevidde og enhetene bruker mindre strøm enn de lisensierte enhetene (3). (Se Appendix) Det lave batteriforbruket forklares ved at enhetene selv oppretter kommunikasjonsforbindelsen. Dersom enheten vil kommunisere 100 ganger per dag eller en gang per dag, vil den gjøre det, serveren etterspør ikke signal fra enhetene. Dette resulterer i at enheten ikke trenger å kontinuerlig vente på et signal, noe som øker batteritiden. LPWAN er derfor svært effektiv når det kommer til å sende mindre mengder data innenfor et gitt intervall (4).

Den lange rekkevidden skyldes av den lave båndbredden som teknologien tar i bruk. Bølgelengden på signalene ligger på 400MHz – 900MHz (5). Signaler som dette penetrerer diverse materialer. Desto lavere frekvens bølgene har jo bedre klarer bølgene å trenge gjennom materialer. Til sammenlikning ligger 5G signaler på mellom 28GHz – 39GHz (6).

2.2.2 Globalt Posisjon System (GPS)

Dette er den mest sentrale sporingsteknologien på markedet for øyeblikket. Den ble vurdert av gruppen, men skrinlagt tidlig under utvikling på grunn av batteriforbruk og kostnad. Gruppen mener derimot den er verdt å diskutere.

GPS gir tilgang til koordinater og tidspunkter i hele verden og er utviklet og drevet av USAs forsvarsdepartement. Tidlig i teknologiens levetid ble GPS kun brukt i militære sammenhenger, men i 1993 ble systemet offentligjort og tatt i dag i bruk av diverse næringer (7). Teknologien opererer med satellitter som går i bane rundt jorden. Hver av disse satellittene sender et unikt signal og baneelement som gjør at GPS enheten kan dekode og finne ut den presise lokasjonen til enheten (8).

GPS sender et nøyaktig signal som har presisjon på rundt 10m. Denne teknologien blir ofte brukt i biler og båter. I nyere tid har GPS blitt mer attraktivt i kombinasjon med IoT og LPWAN. Dette er en teknologi vi ser i mobiltelefoner, og andre små sporings enheter.

2.2.3 Radio Frequency IDentification (RFID)

RFID er en av de sentrale teknologiene gruppen vurderte å basere løsningen på. Den er meget sentral innenfor IoT og blir brukt i mange næringer for sporing av billige verdier. RFID er en type teknologi som kommuniserer via radiosignaler. Et RFID system består av to komponenter: en RFID-tag og en RFID-mottaker/leser. En RFID-tag fungerer ved å sende og motta informasjon via en antenne og en mikrochip. Det eksisterer to forskjellige typer RFID-tags.

2.2.3.1 *Passive RFID-tags*

En passiv RFID tag er en enhet/tag som ikke har en intern strømkilde, men er istedenfor drevet av elektromagnetisk energi puls som overføres fra en RFID leser. Når en passiv RFID tag blir skannet av leseren, vil den overføre nok energi til taggen slik at chipen og antennen kan sende informasjon tilbake til leseren. Leseren vil dermed overføre informasjonen tilbake til en server eller et program (9).

2.2.3.2 *Aktive RFID-tags*

I motsetning til passive tags, har aktive RFID tags en intern strømkilde. Aktive tagger sender kontinuerlig ut den lagrede dataen fra tagen. En RFID-leser kan dermed plukke opp signalet fra en avstand på 150 meter. Det finnes to forskjellige underkategorier av aktive RFID-tags, Transpondere og Beacons:

Beacons sender ut informasjon på et satt tidsintervall, og har en rekkevidde på 100 meter. Ettersom signalene blir sendt ut kontinuerlig, vil levetiden bli vesentlig mindre enn på passive tags. For å minske strømbruken, kan rekkevidden og tidsintervallet reduseres (10).

Transpondere krever bruken av en leser for å hente ut informasjon. Når leseren og transponderen er innen rekkevidde vil leseren sende ut et signal til transponderen, som deretter sender tilbake den relevante informasjonen. Ettersom transpondere kun er aktivert når leseren er nærme, er denne typen RFID-tags mer batterivennlig enn Beacons (10)

2.2.4 Bluetooth Low Energy (BLE)

Dette er teknologien gruppen valgte å basere sporingsløsningen på (i kombinasjon med LTE-M). Gruppen ser det derfor som viktig å introdusere teknologien tidlig i dokumentet.

Bluetooth Low Energy, også omtalt som BLE er en Bluetooth-teknologi som benyttes i stadig flere IoT sammenhenger og er designet for å kommunisere små mengder data og konservere batteritid. BLE enheter opererer i 40 forskjellige kanaler på 2.4GHz spektrumet, med et intervall på 2MHz. Av alle disse kanalene blir 37 kanaler brukt for data overføring. Kanal-designet på BLE gjør at frekvenshopping skjer sjeldnere og problemer med forstyrrelser inntreffer derfor

sjeldent sammenliknet med ordinær Bluetooth (11). BLE kommunikasjon er kryptert med AES-128 ende-til-ende kryptering, noe som forhindrer at data kan leses av eksterne aktører om signalet fanges opp.

BLE enheter kan bruke opptil 100 ganger mindre strøm enn en standard Bluetooth enhet. Dette oppnås blant annet ved reduksjon av mengden data som sendes, samt at BLE enheter «sover» konstant til en tilkobling oppstår. Denne tilkoblingen varer i noen millisekunder, kontra standard Bluetooth tilkobling som varer i omtrent 100 millisekunder.

Teknologien er et alternativ til standard Bluetooth for bruksområder som ønsker å sende små, spesifikke datamengder, samtidig som minimalt med strøm brukes, noe som passer ypperlig for IoT bruksområder. Standard Bluetooth tilbyr konstant kommunikasjon mellom to enheter, der BLE kommuniserer kun i korte pulser (12). Noen eksempler på hvor BLE kan brukes er å lokalisere posisjoner inne i en bygning og koble sammen smartklokker med mobiler. BLE er basert på en master/slave konfigurasjon. Dersom du ønsker å koble opp musikk på en høyttaler fra mobilen, vil mobilen være en master og høyttaleren være en slave.

2.2.5 Near-Field Communicaiton (NFC)

Denne teknologien ble vurdert, men aldri implementert. Den blir derimot diskutert i kap. 5, derfor velger gruppen å introdusere den kort her.

Near-Field Communication, også kjent som NFC er en trådløs overføringsmetode som kan overføre små datamengder over svært små avstander. De fleste smarttelefoner kommer med støtte for NFC fra fabrikken, noe som gjør en løsning med NFC tilgjengelig uten behov for mye ekstra utstyr. NFC taggene kan programmeres til å inneholde en liten mengde data som gjøres tilgjengelig for en scanner-enhet, eksempelvis en smarttelefon. NFC-brikkene er passive, som betyr at de ikke har behov for batteri. De benytter energien utsendt fra NFC-antennene i telefonen til å gjøre sine oppgaver. Dermed er NFC-teknologien avhengig av en ekstern enhet med en NFC-antenne for både strøm og kommunikasjon. Det er programvaren på NFC-leseren/telefonen som avgjør hva dataen fra NFC brikken skal brukes til. Bruksområder der NFC blir brukt er betaling fra mobil og bankkort.

2.2.6 Ultra-wideband (UWB)

Gruppen vurderte tidlig i oppgaven å gå for denne teknologien, på grunn av dens egenskaper med nøye lokasjonspresisjon. Gruppen velger derfor å introdusere denne teknologien her.

UWB er en type radioteknologi som kan kommunisere informasjon over korte avstander. Denne type teknologi sender høye frekvenser med korte pulser på to nanosekunder, noe som gjør denne teknologien nøyaktig innenfor romlig og retningsbestemt data. Denne teknologien kan brukes til å vite om du nærmer deg en dør eller om du er inne eller utenfor døren. Dette kan enhetene vite ved å regne ut hvor lang tid det tar for en radiopuls å reise mellom to enheter, noe som kan sammenliknes med hvordan en flaggermus orienterer seg (13). Signalet som UWB sender ut

kan få noen små forstyrringer, noe som gjør at den har en presisjon på opp til 10 cm. Denne teknologien brukes av Apple Air-Tags.

2.3 Sporingsenhet

For å kunne teste systemet hadde gruppen behov for et sporingssystem. Gruppen diskuterer hvordan de kom frem til både sporingsteknologien og et system i kap. 5. Gruppen testet løsningen med en Gateway som kommuniserer med mindre Beacons/tags som sender informasjon. Gruppen endte opp med følgende komponenter:

2.3.1 INGICS IGS03M

For testing brukte gruppen IGS03M en BLE Gateway. Gateway-en er designet for å plukke opp alt av data som sendes av BLE enheter. Enheten ble brukt som en sentral på byggeplass for å plukke opp stillasdeler med en fastmontert IBS05 tag. Enheten fungerer som bindeleddet mellom API-et og taggene festet på stillasdelen. Den støtter LPWAN i form av LTE-M, og Wi-Fi. Dataen sendes via HTTP POST eller MQTT. Enheten krever 5V med strøm for å fungere. Den støtter diverse filtreringsmetoder og kan konfigureres med et filter slik at enheten kun plukker opp tags av typen IBS05.

Figur 2 Ingics IGS03M Gateway

2.3.2 INGICS IBS05

Dette er BLE tag/beacon modellen gruppen har brukt for testing. Taggen er utviklet av *INGICS Technologies* og fungerer som en BLE Beacon. Enheten sender ut informasjon i form av BLE datapakker. Gruppen har valgt å teste løsningen med disse taggene, de ble under testing festet på stillasdeler og flyttet imellom flere IGS03M Gateways. Enheten er IP67 sertifisert, har en batteritid på 2-3 år og sender en BLE datapakke med tag-id, batteritid og diverse andre sensor parametere inkludert et akselerometer og en temperatursensor.

Figur 3 IBS05 Bluetooth low energy Beacon/tag

2.4 Application Programming Interface (API)

API brukes for å sende data frem og tilbake mellom programvareapplikasjoner. Dette skjer over internett med HTTP protokollen. Brukere av et API har ikke behov for å vite hva som skjer i bakgrunnen av API-et for å bruke det. Den mest kjente type API er RESTful API. Denne type API er basert på «representational state transfer», som er en arkitektonisk stil. For å bruke ressursene har et RESTful API kommandoer som GET, PUT, POST og DELETE. Denne type API er stateless, det vil si at mottakeren ikke er pålagt til å beholde økt-status fra tidligere forespørsler. Senderen sender informasjon på en måte slik at forespørslene kan tolkes uavhengig av en referanse til tidligere forespørsler. I dette delkapittelet vil det bli lagt frem de nødvendige bibliotekene som ble brukt under utvikling av API-et.

2.4.1 Golang

Golang er programmeringsspråket gruppen valgte for utvikling av API-et, det er et «open-source»-programmeringsspråk utviklet av *Google* i 2007, språket er basert på programmeringsspråket C og har derfor en lignende syntaks. Språket blir i dag brukt både innenfor og utenfor *Google*. Golang brukes til alt fra Web til spillutvikling og spesialiserer seg på Cloud- og Nettverksutvikling.

2.4.2 Gorilla Mux

Gorilla Mux er et tredjeparts Golang bibliotek som tilbyr en ruter som sender REST-API forespørsler til et passende endepunkt. Hoved-funksjonaliteten til biblioteket er funksjonen *mux.Router* som definerer ruteren og *mux.Vars*: et rammeverk for implementering av spørringer i endepunkter. Pakken ble sist oppdatert 22.August 2020 og er standard for API-utvikling i Golang.

2.4.3 Encoding/json

Biblioteket er et standard Golang bibliotek[2], men utdypes likevel gitt at det er meget sentralt for API-utviklingen. Biblioteket er i all hovedsak laget for behandling og formatering av data på formatet JSON. Biblioteket brukes i API-et for å behandle forespørsel bodyen. API-et tar i bruk funksjonene *json.Marshal* og *json.Unmarshal* for å formatere JSON-data, og *json.Encode* for å sende en respons.

2.4.4 INGICS Parser

Gruppen måtte under utvikling oversette datapakken sendt fra BLE Gateway-en som ble brukt for testing av systemet. For oversetting ble INGICS Parser biblioteket tatt i bruk.
Dette er et enkelt tredjepartsbibliotek som lar programmet oversette datapakker fra *INGICS Technologies* Gateway-ene og Beacons. For å kunne oversette datapakken Gateway-en sender måtte biblioteket bli tatt i bruk. Biblioteket tilbyr to funksjoner *igs.Parse* for IGS Gateway modeller og *ibs.Parse* for IBS Tag modeller[3].

2.4.5 MQ Telemetry Transport Protokoll (MQTT)

MQTT er en protokoll utviklet for kommunikasjon med IoT enheter. Protokollen tar i bruk et Publish/Subscribe system som lar klienter abonnere på en «Topic» som omhandler en enhet eller en type ønsket data. Med protokollen kan en enhet «Publishe» data til en MQTT Broker, denne brokeren lagrer informasjonen og sender den ut til klienter som abonnerer på et tema. Implementasjon av en slik protokoll hadde vært fordelaktig da protokollen gjør det enkelt å skalere opp IoT arkitekturen til systemet. Protokollen hadde også gjort systemet sikrere gitt at data enkelt kan krypteres med TLS.

[2] Mer om encoding/json biblioteket: https://pkg.go.dev/encoding/json
[3] Link til biblioteket på Github: https://github.com/ingics/ingics-parser-go

2.5 Database

Databaser er essensielle back-end tilgjengeligheter for de fleste datasystemer. Her lagres og organiseres all data. På denne måten kan dataaksessering gjøres enkelt og pålitelig. Gruppen tar i bruk et databasesystem for å lagre informasjonen som brukeren samhandler med. Forskjellige databasesystemer har blitt vurdert. Disse blir beskrevet under.

2.5.1 MySQL

MySQL er en «open-source» relasjonsdatabase (RDBMS) som ble utgitt på midten av 1990-tallet av *MySQL AB*, men eies i dag av *Oracle* og er basert på et strukturert spørrespråk. Relasjonsdatabaser består av tabeller organisert i rader og kolonner som knyttes sammen av relasjoner gjennom nøkler. Disse behøver nøye design og modellering for å oppnå normalisering, men i gjengjeld oppnår man at databasens integritet styrkes.

Fordeler	**Ulemper**
Kan lett benyttes av personer med kun grunnleggende databasekunnskap.	Effektiviteten synker hvis databasen blir stor.
Kostnadsfri – gratis i bruk.	Ikke designet for å skalere ut (horisontalt)
En av de sikreste databasesystemene som finnes – tilbyr blant annet Brannmur, Kryptering ved SSL og Bruker autorisering.	Må konverteres til JSON hvis JSON benyttes av API e.l.
Høy ytelse.	
24/7 tilgjengelighet.	
Kan lett skaleres opp (vertikalt).	
Full data integritet støtte.	

Tabell 1 Fordeler og ulemper mySQL

2.5.2 NoSQL

NoSQL databaser, også kjent som «Not only SQL» eller «ikke bare SQL» har en annen type struktur enn MySQL databaser. Disse varierer etter datamodellen deres. Dataen i disse databasesystemene lagres på en annen form enn dataen ville i relasjonsdatabaser. De mest kjente typene er dokument databaser, nøkkel-verdi databaser, graf databaser og wide column store. Alle disse tilbyr fleksible skjemaer og kan enkelt skaleres med store mengder data og/eller forbruk.

En av de sentrale fordelene med NoSQL databaser er at de tillater lagring av store mengder ustrukturert data, noe som gir mye fleksibilitet. Populariteten i bruken av NoSQL databasestrukturen økte også etter hvert som skytjenester ble mer og mer populært, ettersom horisontal skalering ble enklere og mer tilgjengelig (14).

Nøkkel-verdi databaser er databaser som inneholder en nøkkel med en tilhørende verdi. Denne primitive strukturen fører til svært raske lesinger i databasen og behandler store mengder data raskt, men begrenser mengden data man får ut i én og samme forespørsel. Eksempler på nøkkel-verdi databaser er *Redis, Dynamo* og *Riak.*

Dokumentdatabaser er designet for å motta og behandle semi-strukturert data.
Dokumentdatabaser er en sub-klasse av nøkkel-verdi databaser, ettersom dokumentene er strukturert slik at de inneholder nøkkel-verdi relasjoner i feltene, men her kan også dataen indekses. Dermed kan eksempelvis data for en hel webside hentes i én og samme forespørsel.
Eksempler på dokumentdatabaser er *Amazon SimpleDB, CouchDB, Riak, Cloud Firestore* og *MongoDB.*

Wide column store er databaser som lagrer data i tabeller, rader og dynamiske kolonner. Denne kan ansees å være en to-dimensjonal nøkkel-verdi database.
Eksempler på wide column store er *Apache Cassandra, ScyllaDB, Apache HBase, Google BigTable*, og *Microsoft Azure Cosmos DB.*

Graf databaser er dokument databaser som også linker mellom dokumenter for å effektivisere søking og aksessering.
Eksempler på graf databaser er *Neo4J, Infinite Graph, OrientDB* og *FlockDB.*

Fordeler	Ulemper
Fleksible skjemaer.	Back-up funksjonalitet er ikke sikker.
Kan lett skaleres ut (horisontalt).	Mangel på standardisering.
Raske spørringer.	Må ofte skreddersys til applikasjonens funksjonalitet for best ytelse.
Lett å ta i bruk for utviklere.	Eventuell konsistens
Lagrer dataen i dokumenter som ligner på JSON objekter – felt og verdier.	
Støtter ACID transaksjoner og egner seg også for relasjonsdata.	
Høy ytelse.	

Tabell 2 Fordeler og ulemper NoSQL

NoSQL databaser brukes ofte i settinger hvor behovet for å lagre store mengder ustrukturert og urelatert data er sentralt. Selskaper som behandler stordata, er populære forbrukere av NoSQL databaser. *Facebook, Twitter, Amazon* og *Google* er eksempler på firmaer som samler inn og prosesserer store mengder data hver dag.

2.5.3 Google Firestore

Google firestore er *Googles* egne semi-strukturerte NoSQL dokumentdatabase designet med hensyn på web og mobil utvikling. *Firestore* har en hierarkisk kolleksjon-dokument struktur (se kap. 2.5.2). Databasen oppdateres i sanntid og er enkel å skalere.

Firestore tilbyr Cloud functions: Funksjonalitet som tillater databasen å sende varslinger basert på diverse operasjoner i databasen, i tillegg tillater Cloud functions sammenkobling av dokumenter slik at endring i et dokument oppdaterer sammenkoblede dokumenter. *Firestore* tilbyr også «batched-writes», en funksjon som gjør flere database-operasjoner atomiske.

2.5.4 Skalering og datalagring

To sentrale ulikheter mellom relasjonsdatabaser og dokument databaser er måten skalering foregår på og hvordan dataen lagres. Relasjonsdatabaser benytter seg i de fleste tilfeller av vertikal skalering imens dokument databaser stort sett benytter seg av horisontal skalering. Relasjonsdatabaser samler like data i samme tabell, imens dokument databasen kan strukturere dataen på mange måter. Ofte struktureres denne dataen slik at alt som er av relevans til et objekt samles i samme dokument.

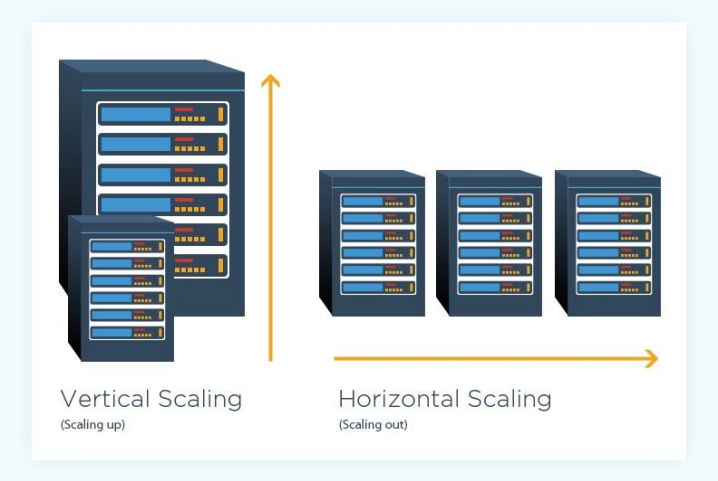
2.5.4.1 *Horisontal- og vertikal skalering*

Skalering endrer størrelsen på systemet. Vertikal skalering har begrensninger i form av hardware imens horisontal skalering kan foregå nærmest ubegrenset.

Vertikal skalering er prosessen i å legge til ressurser til et allerede eksisterende system i form av mer prosesserings kraft (CPU, RAM, etc.). Denne typen skalering er ansett å være det billige alternativet ettersom man ikke må betale for et helt nytt serversystem. Vertikal skalering har begrensninger i hvor mye man kan skalere opp basert på systemet som benyttes.

Horisontal skalering er prosessen i å legge til nye servere til et allerede eksisterende system. Denne typen skalering er vanskeligere og mer kostbar enn vertikal skalering, men den har sine fordeler.

Ved vertikal skalering må serveren spinnes ned og opp igjen noe som gjør oppgraderingen vanskeligere og gir økt nede-tid. Her har horisontal skalering en fordel ettersom den legger til en ny server eller fjerner en server og rører da ikke de andre serverne i systemet. Dermed forenkles oppgraderingsmulighetene og nede-tiden påvirkes ikke i særlig grad. Baksiden er at horisontal skalering er mer tidkrevende å gjennomføre og vanskeligere å implementere enn vertikal skalering (15). I Figur 4 er de to skaleringsmetodene illustrert.

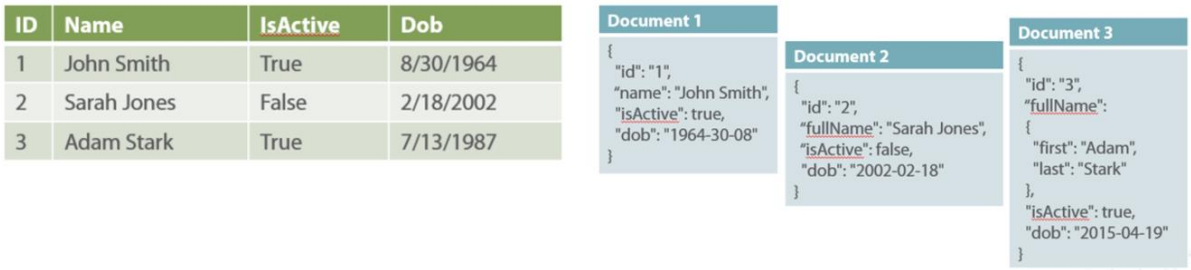


Figur 4 Visuelt bilde på horisontal og vertikal skalering. Bilde hentet fra https://www.cloudzero.com/blog/horizontal-vs-vertical-scaling

2.5.4.2 *Datalagring*

Relasjonsdatabaser lagrer dataen strukturert i tabeller som er bygd opp av rader og kolonner. Her ansees hver rad som et objekt som består av data kategorisert i de ulike kolonnene. Hvert objekt har en primær nøkkel som brukes til å identifisere dataobjektene og objekter kan hentes ut basert på kolonnenavn eller rad indeks. Til venstre i Figur 5 er et eksempel på en relasjonsdatabasetabell for lagring av bruker data. Her lagres data om brukere som har et identifikasjonsnummer *ID*, et navn *Name*, en aktivitetsstatus *isActive* og en fødselsdato *Dob*. Her vil data om alle brukerne i systemet legges inn med tilsvarende informasjon.

Dokumentdatabaser lagrer data i dokumenter, og dokumenter kan lagres i kolleksjoner. Kolleksjonene kan sees på som dokumentdatabasens tabell-alternativ, imens dokumentene er radene/objektene i tabellene. Oppgaven til kolleksjonene er å gruppere dokumenter som faller under samme kategori, til tross for at innholdet i hvert dokument kan variere. Dokumentene lagrer objektene i form av XML, YAML eller JSON, noe som gjør at dokumentobjektene kan variere ettersom feltene for de ulike dokumentene defineres i hvert dokument og ikke av den overordnede kolleksjonen. Til høyre i Figur 5 ser man hvordan dokumenter er delt inn i en dokumentdatabase. Her er JSON strukturens inndeling av felter lik som i relasjonsdatabasen. I dokument 3 blir det vist hvordan dokumenter som inngår i samme kolleksjon kan variere i innhold. Her er *fullName* inndelt i *first* og *last*. En slik struktur ville ikke gå inn i samme tabell som de andre dokumentene i en relasjonsdatabase.



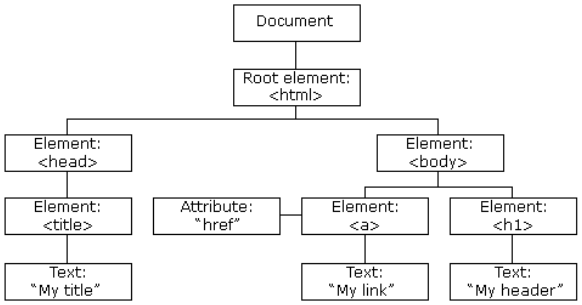
Figur 5 Forskjell på hvordan data er lagret i en SQL database mot en NoSQL database. Hentet fra https://lennilobel.wordpress.com/2015/06/01/relational-databases-vs-nosql-document-databases

2.6 Web-utvikling

Gruppen har valgt å utvikle web-applikasjonen i React da dette var et rammeverk to av gruppemedlemmene var kjent med fra før. React ble laget i 2013 og vedlikeholdes av *Meta* (tidligere *Facebook*). I all hovedsak er React et JavaScript-bibliotek for å bygge brukergrensesnitt. Biblioteket har store likheter med HTML, CSS og JavaScript språkene. Biblioteket er optimalisert for å hente ut skiftende data som registreres. React har som mål å minimere feil som kan skje når det skal utvikles brukergrensesnitt. Dette skjer ved å bruke selvstendige komponenter, med logiske deler av kode. Sammen blir alle disse komponentene satt sammen til et komplekst grensesnitt.

2.6.1 ReactDOM og render

En DOM er en dokument-objekt-modell, som inneholder alle elementene og dens egenskaper. Det er normalt å se på en DOM med en tre-struktur. Når en nettside blir lastet inn er det nettleseren som lager DOM-en til siden.



Figur 6 Representasjon av DOM. Bildet hentet fra https://www.w3schools.com/js/pic_htmltree.gif

En ReactDOM er en pakke som leverer DOM-spesifikke metoder, som gjør det lettere å håndtere DOM elementene i nettsiden. ReactDOM kommer som et API med metoder utviklerne kan bruke. Den viktigste funksjonen API-et leverer er funksjonen *render*.

Render blir brukt til å gjengi en enkelt eller flere komponenter. Det er denne funksjonen som gjør det mulig for nettsiden å bli oppdatert uten å måtte lastes inn på nytt. Dermed er det *render* sin hensikt å vise frem den spesifikke HTML koden innad i dokumentet. ReactDOM sammenlikner de gamle elementene med de nye elementene, og legger kun til de nødvendige oppdateringene for å få DOM til den ønskede tilstanden (16).

2.6.2 Komponent

React komponent er selvstendige biter kode som utvikleren kan gjenbruke i programmet. En komponent fungerer på samme måte som JavaScript funksjoner, men returnerer HTML. Komponenter i React kan enten være klasser eller funksjoner. Når en komponent er definert som en klasse, vil den være kraftigere enn en funksjon komponent. En klasse komponent har muligheten til å lagre lokale variabler som vi kan utføre logikk på når DOM noden er laget og ødelagt.

En funksjon komponent er mindre kraftig enn en klasse, men lettere å jobbe med. Funksjonen oppfører seg som en klasse med en render metode. Det er anbefalt å bruke funksjons komponenter fremfor en klasse i React (17).

2.6.3 Hooks

Hook-er er funksjoner som lar deg feste en variabel sin tilstand fra funksjon komponenter. React tilbyr noen hook funksjoner, slik som *useState* og *useEffect*. Det er også mulig å lage egne hook funksjoner (18). Hook funksjoner kommer med et sett av regler som må tilfredsstilles for å

kunne benyttes. Hook-er må alltid bli bruk på et topp-nivå i koden, og det er ikke tillatt å bruke dem i en løkke, samt at de kun kan brukes i funksjonskomponenter. Disse reglene er satt slik at Hook funksjonene blir kalt på i den samme ordren hver gang en komponent *renderer* (19). Ulempen med en hook funksjon er at den er asynkron, noe som vil si at den ikke oppdateres umiddelbart.

2.6.4 Node.js og NPM

Node.js er en «open-source» og kryssplattform JavaScript «runtime-environment». En Node.js applikasjon kjører i en enkel prosess, uten å lage en ny tråd for hver forespørsel. Ved å levere asynkrone I/O forespørsler vil Node.js hindre JavaScript kode i å bli blokkert. Det vil si når Node.js kjører en I/O operasjon som å hente data fra nettet eller en database, vil ikke denne operasjonen blokkere en tråd. Resultatet av dette vil være høyere ytelse ettersom det ikke blir reservert en prosess i CPU-en. Node.js vil derfor gjøre det mulig at webapplikasjonen kan håndtere opptil tusen koblinger samtidig på en server uten å tenke på å introdusere tråder i programmet (20). *NPM*, Node Package Manager, er en standard pakke-driver for JavaScript i Node.js. NPM tillater utviklere å installere og bruke andre utviklere sine pakker i eget prosjekt. Når en skal bruke NPM pakker i et prosjekt, må prosjektet inneholde en fil som heter package.json. Filen inneholder en liste over alle pakkene som prosjektet bruker, samt hvilken versjon av pakken som er i bruk (21).

2.6.5 React Biblioteker

Bootstrap

Bootstrap er et bibliotek som gjør det lettere for utviklere å lage et godt brukergrensesnitt, uten mye arbeid med design. Bootstrap leverer gjenbrukbare komponenter som er lette å bruke. Dette gjør at utvikleren kan fokusere mer på back-end, kontra front-end utviklingen (22).

Material UI

Material UI er, som Bootstrap, er et brukergrensesnitt-bibliotek. Material UI kom ut i 2014 og har etter dette blitt et av de mest populære bibliotekene for React. Material UI baserer seg på *Googles* design. Søkelyset ligger på å fremme et design som fokuserer på hvordan komponentene reflekterer lys og kaster skygger (23).

Router

React Router DOM er et bibliotek som gjør det mulig å legge inn dynamiske ruter i webapplikasjonen. Det er denne pakken som gjør det mulig å navigere seg gjennom et nettsted. Denne blir brukt på single-page-applikasjoner som har flere «sider» eller komponenter. Istedenfor at nettsiden skal bli lastet inn på nytt vil innholdet bli dynamisk hentet basert på URL-en. Denne type ruting er raskere enn vanlig tradisjonell navigasjon, og brukeren vil få en bedre opplevelse samtidig som applikasjonen vil ha en bedre ytelse (24).

React-Query

React Query er et bibliotek som behandler data. Biblioteket administrerer henting av data, caching, synkronisering og oppdatering av server innhold. React Query løser mye av utfordringene ved å holde data ferskt i minnet, og oppdatere data når det trengs. Dette gjør at flere komponenter kan bruke samme data uten at applikasjonen trenger å kjøre flere hentinger fra et API (25).

Mapbox

Mapbox GL JS er et bibliotek som gjør det mulig å legge til kart og andre kartegenskaper til nettsiden. Dette er et kraftig bibliotek som lar deg legge til forskjellige kart med satellitt-, hybrid- og gatekart, 3D data og animasjoner på nettsiden. I tillegg tilbyr Mapbox muligheten av andre verktøy slik som geokoding, annoteringer og polygoner. (26).

2.6.6 Asyncronous

Asynkron programmering er en måte å si til programmet at det skal starte en oppgave som tar lang tid å kjøre. Istedenfor at programmet venter på at oppgaven gjør seg ferdig, vil den heller fortsette eksekveringen av resten koden. Når oppgaven er ferdig kjørt, vil applikasjonen representere resultatet. Eksemplet på slike oppgaver er å hente data fra API og aksessere I/O fra maskinen (27).

En Promise er en stedfortreder for en verdi som ikke er kjent når promisen er laget. Dermed blir promises en innpakking rundt verdier som blir eller ikke blir kjent før objektet er initialisert. Istedenfor å returnere den endelige verdien, vil metoden returnere en promise, der den endelige verdien kan legges inn. En promise er alltid i en av disse stadiene, «pending», «fulfilled» eller «rejected». Når en promise er «pending» er den i en ventende tilstand. Dersom den blir «fulfilled» betyr dette at operasjonen ble gjennomført uten problemer. I motsetning betyr «rejected» at prosessen feilet. Når tilstanden har gått fra pending til enten «fulfilled» eller «rejected», fortsettes metoden med en «then», «catch» og «finally» metode.

Dette kalles «Chained Promises» dette brukes for å utføre videre handlinger med verdien som erstatter promisen. «Then» metoden tar inn to funksjoner, der den første er for promisen dersom den blir «fulfilled», og den andre dersom den blir «rejected». Dersom man kommer i situasjonen der error-handling må skje med en gang, kan en «catch» metode benyttes. «Catch» metoden er lik «then» uten at det er mulighet for å håndtere promisen dersom den blir «fulfilled» (28).

Dersom du bruker nøkkelordet «async» i en funksjon vil det bli lettere å jobbe med asynkron promise-basert kode. Innad i en asynkron funksjon kan nøkkelordet «await» brukes for å returnere en promise. Dette gjør at koden venter på det spesifikke punktet frem til promisen enten har blitt «fulfilled» eller blitt «rejected». Dette gjør at utvikleren kan skrive asynkrone funksjoner, samtidig som det ser ut som en synkron kode (29). Dersom du bruker nøkkelordet async i en funksjon vil det bli lettere å jobbe med asynkron promise-basert kode.

2.6.7 Lagring

React støtter lagring av data og tokens i websidene. Dette er hensiktsmessig for å lagre innstillinger, og å gi en bedre brukeropplevelse.

LocalStorage

Lokal lagring er en måte å lagre data på brukerens datamaskin i form av en map. Dataen blir lagret uten en utløpsdato, dermed vil den bli værende på brukerens maskin frem til den manuelt blir slettet. Lokal lagring kan lagre opptil 10MB med data, og blir aldri overført til serveren. Ulempen med denne type lagring er at dataen blir lagret i form av ren tekst. Det er dermed ikke lagt inn noe sikkerhet for denne dataen (30).

SessionStorage

Økt lagring er likt lokal lagring. Forskjellen på de to typene lagring er at økt lagring lagrer dataen for den spesifikke økten. Dermed vil dataen bli slettet fra maskinen når brukeren lukker fanen. Økt lagring blir lagret på samme måte som i lokal lagring, og kan lagre like mye data.

Cookies

Cookies blir brukt for å lagre personlig data om brukeren, for å kunne gjøre opplevelsen på nettsiden mer personlig. Cookies blir ofte brukt til å lagre innloggingstilstanden til brukeren, eller innholdet i en personlig handlekurv. Cookies er lagret på datamaskinen til brukeren, og har mulighet til å lagre opptil 4KB data. Cookies har en utløpsdato, men dette avhenger av innstillinger for hver fane og nettleser (31).

2.7 Mobil-utvikling

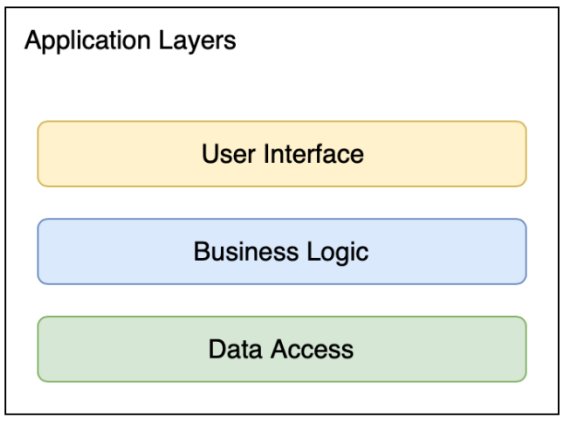
En mobilapplikasjon er et program som installeres på mobile enheter som smarttelefoner og nettbrett. Mobilapplikasjoner kan være utviklet i HTML5 som webapplikasjoner eller hybrid applikasjoner. Dette er «native» konteinere som benytter et mobil WebView-objekt eller er programmert og kompilert for bestemte operativsystemer, også kjent som «native apper» (32). Gruppen har for apputvikling valgt å basere løsningen på Swift da dette er et moderne og hyppig brukt programmeringsspråk.

2.7.1 Swift

Swift er et gratis og «open-source» programmeringsspråk utviklet av *Apple Inc*. og «open-source» fellesskapet, spesiallaget for utvikling av iOS, iPadOS, macOS, tvOS og watchOS applikasjoner. Swift ble først utgitt i 2014 og er etterfølgeren til Objective-C. Swift er designet slik at det problemfritt skal kunne implementeres i allerede eksisterende Objective-C kode og gjennom bruken av LLVM kompileringsteknologi oppnås svært høy ytelse fra «native app»-er (33).

2.7.1.1 *Lag-arkitektur*

En lag-arkitektur benyttes i iOS applikasjoner for å oppnå bedre struktur. Et lag er en logisk struktureringsmekanisme for elementene som er benyttet til å bygge opp en applikasjon. Denne lag-arkitekturen bygges som regel opp av (fra topp til bunn) Brukergrensesnitt, Business Logic og Data Access. Lagene følger avhengighets-regelen som tilsier at et lag kan aksessere sine underliggende lag, men ikke omvendt (34).



Figur 7 Oppbyggingen av lag-arkitekturen i swift. Bilde hentet fra https://www.vadimbulavin.com/layered-architecture-ios/

Brukergrensesnitt

User interface er ansvarlig for samhandlingen og kommunikasjonen mellom brukeren og applikasjonen. Dette laget viser informasjonen til brukeren i form av et grensesnitt bestående av Views. Kommunikasjonen fra brukeren sendes videre til de underliggende lagene hvor de prosesseres og returneres til brukergrensesnittet. De vanligste elementene i brukergrensesnitt laget er Views og Storyboards og disse organiseres vanligvis med MV*-mønstrene (MVC, MVP, MVVM).

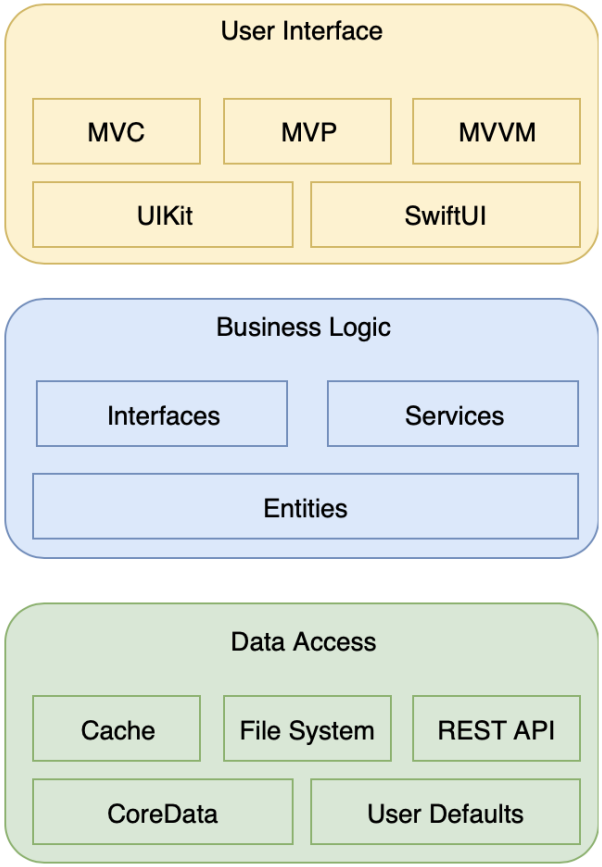
Business logic

Buisness logic layer er ansvarlig for funksjonaliteten til applikasjonen. Det er i dette laget at logikken i applikasjonens oppbygning defineres og gjennomføres gjennom blant annet CRUD operasjoner. Funksjonaliteten tatt i bruk her er etterspurt basert på Use Cases.

Data access

Data access laget er ansvarlig for kommunikasjonen med de eksterne systemene applikasjonen kommuniserer med. Dette vil eksempelvis være databasesystemer, API-er eller annen nettverk kommunikasjon. Det er også i dette laget at cachingen foregår i *CoreData* og *UserDefaults* (34).

I Figur 8 ser man hvordan hierarkistrukturen i en iOS applikasjon ofte vil se ut:

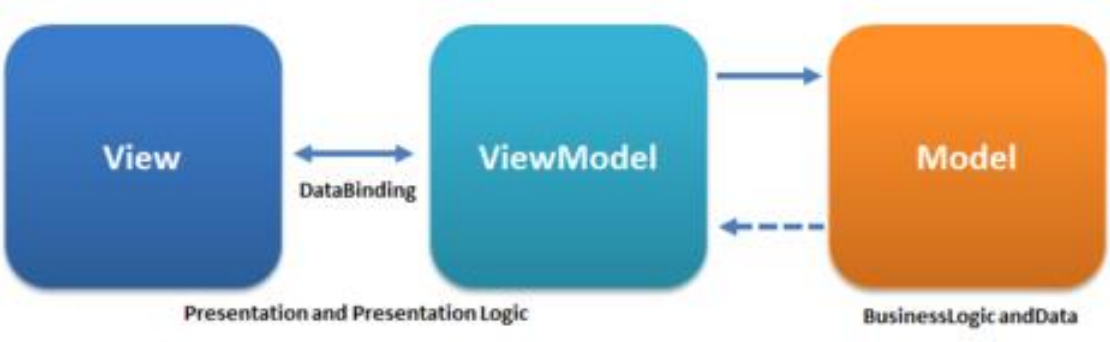


Figur 8 Visuelt Hierarkistrukturen i IOS. Bilde hentet fra https://www.vadimbulavin.com/layered-architecture-ios/

2.7.1.2 *Model-view-viewmodel (MVVM)*

MVVM er et arkitekturdesign mønster som benyttes i utvikling sammenheng. Mønsteret er en forbedring av MVC-mønsteret og har blitt stadig mer populært innenfor iOS apputvikling industrien. Dette mønsteret er anbefalt å følge for å abstrahere koden ut i oversiktlige seksjoner som forbedrer lesbarhet, vedlike-holdbarhet og videreutvikling av koden til applikasjonen. Et problem som fort oppstår i MVC-mønsteret er at kontrollerne fort kan bli veldig store. MVVM benytter seg fremdeles av disse kontrollerne, men legger opp til at ViewModel og Model skal være to forskjellige objekter. Som navnet tilsier deles MVVM inn i tre underkategorier: Models, Views og ViewModels. Når det ses på SwiftUI og MVVM i kombinasjon, trekkes ofte Bindings

inn. Dette er fordi Bindings er sentralt i MVVM. Bindings går ut på å koble sammen dataen og Viewen som viser eller redigerer den, noe som resulterer i at Models og Views kobles sammen og fremdeles forblir «loosely coupled».



Figur 9 Visuelt bilde av Model-View-Viewmodel arkitekturen. Bildet hentet fra https://en.wikipedia.org/wiki/Model–view–viewmodel

Moduler

På lik linje med de fleste andre programmeringsspråk benyttes importering av moduler, pakker og rammeverk.

SwiftUI

SwiftUI er et rammeverk som benyttes i bygging og vedlikehold av brukergrensesnitt. SwiftUI tilbyr mye av de samme funksjonalitetene som UIKit tilbyr, men den tilbyr også verktøy for å lage skreddersydde Views, legge til animasjoner og handlinger basert på bevegelser. I motsetning til UIKits imperative brukergrensesnitt har SwiftUI et deklarativt brukergrensesnitt. Med andre ord har *Apple* abstrahert den komplekse funksjonaliteten vekk fra utvikleren (35).

UIKit

UIKit er et grunnleggende rammeverk for bygging og vedlikehold av brukergrensesnitt. Her defineres blant annet hvordan data skal vises til brukeren og hvordan systemet skal respondere på brukerinput og systemhendelser (36).

MapKit

MapKit er et rammeverk for å vise kart eller satellitt bilder, vise «points of interest» og hente ut informasjon om steder definert av koordinater i en app. I kartet kan man navigere seg, legge til annoteringer for å fremheve «points of interest», samt legge til mulighet for søking i kartet. MapKit muliggjør også definering av ruter og uthentig av ruteinformasjon (37).

Foundation

Foundation er et rammeverk som tilbyr app funksjonalitet for datalagring, dato og tid kalkulasjoner, sortering og filtrering, samt nettverkskommunikasjon med API-er som ikke er en del av Swifts standard biblioteker (38).

CoreLocation

CoreLocation er et rammeverk for å få tak i den geografiske plasseringen og orienteringen til en mobil enhet. Rammeverket bruker komponent enheter som blant annet Wi-Fi og Bluetooth

for å orientere seg. Lokasjonsdata tilbys kun fra enheten til applikasjonen dersom brukeren har godkjent Apples forespørsel om å bruke lokasjonsdata (39).

Combine

Combine er et rammeverk som tilbyr et Swift API for prosessering av verdier over tid. Dette benyttes ofte i sammenheng med Foundation rammeverket og er ansvarlig for å mappe dataen fra API responsen når denne blir publisert (av en «publisher») til en abonnent («subscriber») (40).

2.7.1.3 *Package manager og bibliotek*

Package manager

Kodemoduler som gjenbrukes på tvers av filer internt i programmet eller som stammer fra eksterne utviklere som behøves i et program må importeres. Swift Package Manager er et verktøy som er ansvarlig for å distribuere disse importerte modulene og koden. Swift Package Manager er integrert i byggesystemet i Swift for å forenkle og automatisere installasjons- og kompileringsprosessen, samt linkingen av «dependencies» (41).

Bibliotek

Firebase Authentication er en package tilbudt av *Googles* Firebase for å tillate brukere å registrere seg og logge inn i en applikasjon enten ved bruk av email og passord, eller gjennom eksempelvis *Google* eller *Facebook* (42).

2.8 Sikkerhet

Dette underkapittelet tar for seg de ulike autentifikasjon/autorisasjon teknologiene som gruppen vurderte å implementere i systemet. Autentifikasjonen skulle implementeres på en måte som kunne beskytte API, Mobil- og Webapplikasjonene.

2.8.1 OAuth

OAuth er en standard designet for å gi en nettside eller applikasjon tilgang til ressurser fra andre web applikasjoner. OAuth er en autorisasjonsprotokoll, som autoriserer brukeren ved bruk av tokens. Denne protokollen kan kategorisere 4 aktører;
En eier, som eier dataen i ressurs serveren. Klienten er applikasjonen som ønsker å aksessere dataen. Autorisasjons-server som er hoved driveren i OAuth og ressurs-server som er API-et som lagrer dataen applikasjonen ønsker å aksessere. (43).

OAuth kan benytte seg av flere typer tokens, som kan autorisere brukeren. «Access token» er token som klienten bruker for å aksessere ressurs serveren. Denne type token har en kort levetid på rundt noen minutter til timer. «Refresh token» er en token som har en levetid på noen dager til år. Det er denne token som blir brukt til å skaffe nye tokens. OAuth definerer ikke hvilket format en token skal være, men det er vanlig å bruke en JSON Web Tokens (44).

2.8.2 JSON Web Token (JWT)

JSON web Token er en standard som definerer en måte å sende informasjon sikkert mellom to parter som et JSON-objekt. JWT blir hashet med HMAC, eller kryptert med public/private nøkkel (45). JWT kan brukes på forskjellige måter, inkludert autentifikasjon og autorisasjon. Autentifikasjon blir benyttet ved å sende en ID-token når en bruker har logget inn. Når en bruker er autentifisert kan man filtrere mulighetene brukeren har ved hjelp av ID-token-en (46).

2.8.3 Firebase Authentication

Firebase Authenticaton leverer en back-end tjeneste som lar deg autentifisere en bruker i en applikasjon, med passord og email. *Firebase Authentication* lar utviklere integrere autentifikasjon lettere i systemer. Her blir email og passord lagret, og brukeren kan håndtere sin egen bruker ved å lage en ny bruker, endre passord, og få en forespørsel dersom brukeren har glemt passordet (47).

3. Kravspesifikasjon

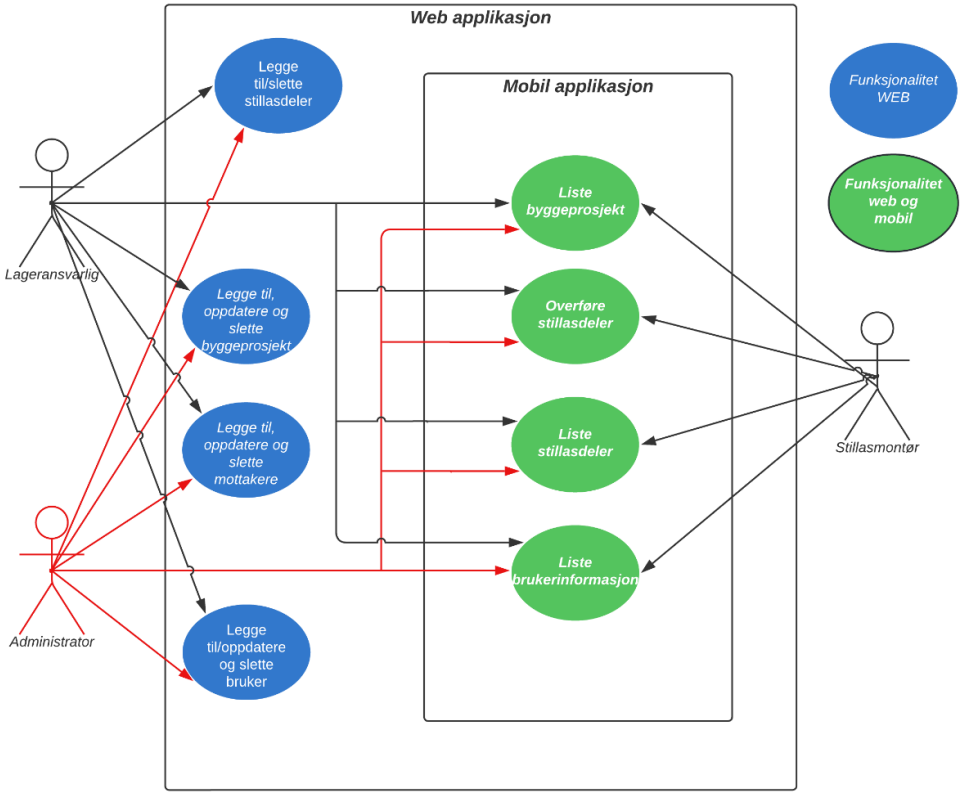
Dette kapittelet vil definere hvilke krav oppdragsgiver og gruppen har satt til løsningen. Det ble definert mål som inneholdt både satte krav fra oppdragsgiver, samt krav satt av gruppen etter møter og dialog med oppdragsgiver. Kravene ble presentert i form av User-stories gruppen utviklet ved hjelp av oppgavebeskrivelsen. Gruppen valgte denne fremgangsmåten tidlig i prosessen da oppdragsgiver ville at gruppen skulle ha full frihet til å løse oppgaven slik vi ønsket.

3.1 Kravspesifikasjon

For å definere en god kravspesifikasjon gruppen kunne ta i bruk ble det begynt med å konkretisere krav oppdragsgiver satte fra oppgavebeskrivelsen. Gruppen valgte å dele opp kravspesifikasjonen i to, der del en tar for seg krav til den fysiske sporingsenheten/sporingsteknologien og del to tar for seg krav til den digitale løsningen som skulle utvikles.

3.1.1 Brukermønster

Gruppen valgte å ta i bruk et enkelt use-case diagram (Figur 10) med all front-end funksjonalitet for både mobil og web-løsningen. Use case diagrammet inneholder tre aktører som skal ta i bruk systemet. Gruppen har valgt å rangere tilgang på funksjonalitet hierarkisk, der administratoren har tilgang til all funksjonalitet spesielt håndtering av brukere. Lageransvarlig har tilgang til all funksjonalitet utenom brukerhåndtering, og stillasmontører har kun tilgang til funksjonalitet tilgjengelig i mobil applikasjonen.



Figur 10 Use case diagram med systemets tre aktører

3.1.2 User stories

Use case	Aktør	Hensikt	Beskrivelse
Lage byggeprosjekter	Lager-Ansvarlig	Opprette byggeprosjekter	Legger inn et nytt byggeprosjekt i databasen
Liste byggeprosjekter med stillasdeler	Lageransvarlig Stillasmontør	Hente ut informasjon om byggeprosjektet	Henter ut byggeprosjektet med antall stillasdeler
Oppdatere byggeprosjekter	Lageransvarlig	Oppdatere byggeprosjekt	Brukeren ønsker å oppdatere informasjon om et gitt byggeprosjekt
Slette byggeprosjekter	Lageransvarlig	Fjerne byggeprosjekter	Brukeren setter et prosjekt til inaktivt/sletter prosjektet
Overføre stillasdeler	Lageransvarlig Stillasmontør	«Sende» stillasdeler til byggeprosjekter	Brukeren ønsker å flytte stillasdeler til et nytt byggeprosjekt
Lage mottakere	Lageransvarlig	Legge inn nye mottakere	Lageransvarlig har behov for en ekstra mottaker på et byggeprosjekt/ønsker å legge inn en mottaker på et nytt prosjekt
Oppdatere mottaker	Lageransvarlig	Oppdatere prosjekt på mottaker	Lageransvarlig ønsker å overføre en mottaker til et nytt prosjekt
Slette mottaker	Lageransvarlig	Fjerne mottaker fra prosjekt	Lageransvarlig ønsker å fjerne en mottaker fra et gitt byggeprosjekt
Sjekke stillasdeler	Lageransvarlig	Hente ut informasjon om stillasdeler	Lageransvarlig er ute etter informasjon om en spesifikk stillasdel i systemet
Lage stillasdeler	Lageransvarlig	Legge inn stillasdeler i systemet	Lageransvarlig ønsker å legge inn nye stillasdeler i systemet
Slette stillasdeler	Lageransvarlig	Fjerne stillasdeler fra systemet	Lageransvarlig ønsker å fjerne ødelagte stillasdeler fra systemet
Lage Bruker	Administrator	Legge inn en ny bruker	System-Administrator ønsker å legge inn en ny bruker i systemet
Oppdatere Bruker	Administrator	Oppdatere brukerinfo	System-Administrator må oppdatere informasjon om en bruker i systemet
Slette bruker	Administrator	Slette brukerinfo	System-Administrator ønsker å fjerne en bruker fra systemet

Tabell 3 User-Story tabell konstruert fra krav

3.1.3 Funksjonelle krav

Følgende funksjonelle krav ble satt etter granskning av oppgaven, konstruksjon av User-stories, samt dialog med oppdragsgiver:

Sporingsenhet:

- Sporingsenheten må fungere uavhengig av norske temperaturer og klimaforhold:
  - Sporingsløsningen må tåle regn og snø.
  - Sporingsløsningen må kunne operere i -20 til 50 varmegrader.
  - Sporingen må være uavhengig av sikt.

- Sporingsløsningen må kunne vise antall enheter på en gitt byggeplass til enhver tid.
- Løsningen må være meget skalerbar.
    - Pris bør ideelt sett ligge i området en femtedel av prisen på stillasdelen.
    - Løsningen må kunne registrere et stort volum av enheter uten problem.
- Enheter må ha en levetid tilsvarende stillasets levetid (10-15 år).

Digital løsning:

- Løsningen skal gi oppdragsgiver informasjon om hvilket utstyr som går fra lager til byggeprosjekt på et gitt tidspunkt.
- Løsningen skal vise stillasdeler i kart.
- En bruker skal kunne se byggeprosjekter og registrerte stillasdeler.
- En bruker skal kunne legge til/slette og oppdatere byggeprosjekter.
- Det skal kunne legge til/slettes og oppdateres stillasdeler i systemet.

3.1.4 Ikke-funksjonelle krav

Sporingsteknologi

- Enheter skal ta i bruk beste tilgjengelige sporingsteknologi.
- Den fysiske delen av sporingsløsningen må kreve minimalt med oppsett og menneskelig interaksjon.

Digital løsning:

- Løsningen må ha et enkelt brukergrensesnitt.
- Det må utvikles et API slik at systemet kan kobles på eksisterende løsninger, samt videreutvikles.
- Digital løsning skal utvikles uavhengig av sporingsenhet der det er mulig slik at denne kan byttes ut når et bedre alternativ er tilgjengelig på markedet.
- Løsningen skal kunne tas i bruk på iOS og Windows 10.

3.2 Product backlog

Under ligger den totale backlogen gruppen brukte igjennom prosessen, for full oversikt over backlogen igjennom prosessen se Appendix:

Mål	Start dato	Ansvarlig	Kommentar	Status
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Ferdig vi bruker Telia
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Ferdig
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Ferdig
Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Ferdig
Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Ferdig
Lage hovedrapport dokument	11.03.2022	Martin	Sette opp dokument	Ferdig
Revidere API design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må API revideres	Ferdig
Revidere DB design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må DB revideres	Ferdig
Begynne implementasjon Web	11.03.2022	Aleksander		Ferdig
Begynne implementasjon Mobil	11.03.2022	Tormod	Satt opp kart og grunnstruktur	Ferdig
Finne BLE tag til enhet	11.03.2022	Tormod og Martin	Gruppen trenger en BLE enhet til sentralen fra Telia	Scrappet
Hente Prosjekt endpoint	11.03.2022	Aleksander	Gruppen begynner utvikling	Ferdig
Hente stillasdel endpoint	11.03.2022	Martin	Gruppen begynner utvikling	Ferdig

Slette prosjekt endpoint	12.03.2022	Aleks		Ferdig
Finne ny sporingsenhet	15.03.2022	Tormod	Gruppen kan ikke bruke enhet fra Telia og må finne ny løsning	Ferdig
Begynne skriving på avsluttende rapport	18.03.2022	Martin	Gruppen ønsker å begynne på rapportskriving tidlig	Ferdig
Standardisere error-handling for API	16.03.2022	Aleksander	Aleksander ønsker å standardisere error handling for API-et	Ferdig
Implementere slette stillasdel funksjon	13.03	Martin		Ferdig
Implementere legge til prosjekt	14.03.2022	Aleksander		Ferdig
Implementere legge til, hent ut og slett bruker	19.03.2022	Aleksander		Ferdig
Sette opp API-Tester	24.03.2022	Martin	Martin ønsker å teste systemet	Ferdig
Legge inn søkefunksjonalitet i Mobil-Applikasjon	23.03.2022	Tormod		Ferdig
Legge inn funksjonalitet for overføring av stillasdeler	24.03.2022	Aleksander		Ferdig
Legge inn kartfunksjonalitet i mobil	26.03.2022	Tormod	Ønsker å oppgradere kartfunksjonalitet	Ferdig
Hente ut prosjektinfo og stillasdel info i mobil appen	27.03.2022	Tormod	Ønsker å hente data fra API	Ferdig
Implementere «Annotations» mobil-aplikasjon	28.03.2022	Tormod		Ferdig
Implementere Docker for deployering av løsning	01.04.2022	Martin	For å kunne spinne opp alle systemets komponenter trengs det en konteiner løsning som Docker	Ferdig
Oppdatere kart funskjonalitet i Web	01.04.2022	Aleksander		Ferdig
Starte med overføring av stillasdeler Mobil applikasjon	01.04.2022	Tormod		Ferdig
Implementere Docker Compose fil	06.04.2022	Martin		Må gjøres

Få ut data fra sporingsenhet	04.04.2022	Martin	Vi har nå fått sporingsenhet og Martin jobber med å få ut data	Ferdig
Legge inn egen fil for API info henting, mobil	02.04.2022	Tormod		Ferdig
Legge inn Modaler for stillasdel og overføring	08.04.2022	Aleksander		Ferdig
Implementere MQTT broker	08.04.2022	Martin	Ser på implementasjon av mqtt broker	Scrappet
Implementere filtrering i mobil	13.04.2022	Tormod		Ferdig
Legge til rette for geofence	14.04.2022	Aleksander		Ferdig
Få sporingsdata inn i API	19.04.2022	Martin		Ferdig
Oppdatere byggeprosjekter og stillasdeler med data	21.04.2022	Martin		Ferdig
Implementere caching i mobil	19.04.2022	Tormod		Ferdig
Oppdatere data i web applikasjon	18.04.2022	Aleksander		Ferdig
Ferdigstilling av prosjekt i mobil applikasjon	19.04.2022	Tormod		Ferdig
Implementere gateway endpoint i API	19.04.2022	Martin	Api-et trenger et gateway endpoint da enhetene må inn i databasen	Ferdig
Ferdigstille oppdatering av posisjon på stillasdeler	25.04.2022	Martin		Ferdig
Implementere login funksjonalitet	28.04.2022	Aleksander	Systemet bør ha login funksjonalitet	Ferdig
Gjenoppta arbeid med filtrering mobil applikasjon	27.04.2022	Tormod	Gjenopptar arbeid på filtrering	Ferdig
Ferdigstille rapport	02.05.2022	Alle gruppemedlemmer		Jobbes med
Kommentere kode	05.05.2022	Alle gruppemedlemmer		Jobbes med

Tabell 4 Komplett Backlog, brukt igjennom SCRUM sprintene

4. Utviklingsprosess

Kapittelet vil beskrive utviklingsprosessen gruppen valgte for å gjennomføre oppgaven. Utviklingsmetoden og eventuelle endringer vil bli forklart. Det vil også bli forklart hvordan møtene ble planlagt, både internt, med oppdragsgiver, eksterne bedrifter, samt veileder. Til slutt vil kapittelet ta for seg en kort oppsummering av de forskjellige SCRUM sprintene gruppen gjennomførte.

4.1 Møter

Gitt at gruppen planla å gjennomføre hele bacheloroppgaven på skolen og ikke hjemmefra/digitalt ville det holdes følgende ukentlige møter:

Møte med veileder: Møtene holdes ukentlig og fysisk (der det er mulig), men holdes digitalt via zoom om veileder ikke er tilgjengelig ved satt tidspunkt. Under møtet vil det bli diskutert spørsmål gruppen har, samt informasjon/tilbakemeldinger fra veileder. Spørsmål vil i all hovedsak være rettet mot gjennomførelse av oppgaven. Ved tekniske spørsmål står gruppen selv ansvarlig for å avtale et møte med andre faglærere eller andre. Grunnen til dette er at veileders kompetanse i all hovedsak ligger innenfor grunnleggende objektorientert programmering, samt gjennomføring av bacheloroppgaven. Det vil bli skrevet møtereferater om nødvendig.

Møte med oppdragsgiver: Møte med oppdragsgiver vil forekomme ukentlig og vil brukes som et statusmøte hvor oppdragsgiver får informasjon om oppgavens status. Tidlig i prosessen vil møtene bli brukt for å spesifisere kravspesifikasjonen da oppgaveteksten er noe vag, samt at oppdragsgiver ønsker at gruppen selv er med på å forme spesifikasjonene. Tidlige møter ble også brukt til å diskutere diverse sporingsteknologier og enheter da dette var en sentral del av oppgaven hvor oppdragsgiver måtte involveres. Senere møter ble brukt til å oppdatere oppdragsgiver på generell fremgang og eventuelle spørsmål/spesifikasjoner.

Interne møter: Gruppen har strukturert utviklingsprosessen med to møter per SCRUM sprint: Et oppstartsmøte der SCRUM master setter opp sprint mål og diskuterer prosjektets Backlog. Under møtet settes målene, hvem som er ansvarlig for oppgavene, samt over hvilket tidsrom sprinten skal foregå. Det vil deretter bli avholdt et oppsummeringsmøte på slutten av den planlagte SCRUM sprinten, her diskuteres resultatene som er oppnådd satt opp imot målene. Eventuelle andre oppnåelser vil også bli diskutert. Backlogen vil bli diskutert og oppdatert. Utenom disse to møteformene vil det bli holdt interne møter ved behov.

4.2 Metodikk

Ettersom oppdragsgiver ikke krevde en spesifikk utviklingsmetode, sto gruppen fritt til å velge utviklingsmetoden som var best egnet til oppgaven. Da utviklingsmetoden skulle velges måtte gruppen sette de tilgjengelige metodene opp imot følgende momenter:

- Oppgaven er en utviklingsoppgave med en relativt åpen oppgavetekst.
- Oppdragsgiver er ikke erfaren innenfor programvareutvikling noe som satt krav til gruppen angående besluttsomhet og definisjon av kravspesifikasjon.

- Kommunikasjon med oppdragsgiver var essensielt og krevende da oppdragsgivers fagfelt ikke er programvareutvikling/informatikk.
- Systemet ble utviklet for en oppdragsgiver med et ukonvensjonelt arbeidsmiljø for IT-bransjen.

Følgende krav til utviklingsmetoden ble satt ved hjelp av momentene over: Metoden måtte gi gruppen frihet til å tilpasse seg oppdragsgiver og deres ønsker. Den må åpne for aktiv dialog imellom både gruppemedlemmer og oppdragsgiver, og metodikken måtte gi gruppen mulighet til å jobbe med flere deler av prosjektet samtidig.

På bakgrunn av dette valgte gruppen å ta i bruk SCRUM metodikk. Metoden ga frihet til å tilpasse utviklingsprosessen i forhold til arbeidsgivers ideer. SCRUM påla også ukentlig planlegging i form av SCRUM møter, noe som holdt prosessen oversiktlig. Metoden inkluderte også oppdragsgiver i stor grad ved hjelp av ukentlige SCRUM møter. Gruppen vurderte en mer rigid utviklingsmetode (fossefall metodikk), men gikk bort ifra dette av frykt for at metodikken ville begrense kreativitet og funksjonalitet på bakgrunn av at oppdragsgiver var meget fleksibel rundt tilnærming av oppgaven.

4.3 Scrum oppsummering

Kapittelet vil gå gjennom alle SCRUM sprintene gruppen har hatt. Sprintene vil bli oppsummert her. For en komplett oversikt over alle sprintene, mål og resultater, samt back-log, se Appendix.

4.3.1 Sprint 1 - 11.01-14.01

Prosjektplan og oppstartsmøter:

Gruppen har påbegynt bachelor-oppgaven og brukte Sprint en på å spesifisere oppgaveteksten samt diskutere hvilken funksjonalitet som var viktig for oppdragsgiver, i tillegg ble prosjektavtalen skrevet og signert av gruppemedlemmer. Gruppen begynte også å skrive på prosjektplanen i tillegg til å ha et møte med både oppdragsgiver og veileder.

4.3.2 Sprint 2 - 17.01-28.01

Prosjektplan og sporingsteknologier:

Under Sprint to ble tiden brukt på å skrive ferdig et førsteutkast til veileder (sendt inn den 25 januar). I tillegg har gruppen også påbegynt en rapport som skal inneholde informasjon om aktuelle sporingsteknologier og et forslag på hvordan gruppen skal løse den fysiske delen av oppgaven (sporing på byggeplass). Det er også brukt tid på å undersøke hvordan lignende problemstillinger (sporing av billige verdier) har blitt gjort tidligere. Gruppen har til slutt hatt møter med både veileder og oppdragsgiver. Gruppen håper å ha valgt en sporingsteknologi innen den fjerde februar.

4.3.3 Sprint 3 - 31.01-11.02

Sporingsteknologier, aktuelle selskaper og besøk hos HAKI

Sprint 3 ble hovedsakelig brukt til undersøkelse av tidligere løsninger som inneholdt sporing av billige verdier. Gruppen undersøkte tidligere forskningsoppgaver i tillegg til å gå i kontakt med diverse selskaper som tilbød sporingsenheter. Gruppen var i kontakt med diverse selskaper

angående sporingsenheter, men gruppen snakket hovedsakelig med følgende: *ABAX*[4]om deres Mini2 enhet, Telia for diskusjon rundt deres mobilnettverk og integrasjon av *LPWAN*[5]. Gruppen var også på besøk hos *HAKI*[6] for å se nærmere på det fysiske stillaset og montering og lagringsprosessen deres.

4.3.4 Sprint 4 - 14.02-25.02

Sporingsteknologier, design og fremlegg for oppdragsgiver

Gruppen har i perioden jobbet med sporingsteknologi dokumentet, dette nærmer seg ferdig og design av både Database, API og Front-end løsningene har begynt. Gruppen var på visitt hos Oppdragsgiver (*MyLift og Borud stillas*) i Hamar den 25.02.2022, her presenterte gruppen hvordan vi ønsket å gå frem for å løse oppgaven. Under besøket bestemte oppdragsgiver og gruppen at løsningen skulle ta i bruk *LPWAN* og *BLE*.

4.3.5 Sprint 5 - 26.02-03.03

Programmeringsspråk og sporingsenhet

Sprinten har gått ut på å komme til en beslutning angående de forskjellige programmeringsspråkene gruppen skal ta i bruk. API skal utvikles i *Golang*, mobil-applikasjonen skal utvikles i *Swift*, Databasen skal ta i bruk *NoSql* og deployeres i *Google Firestore, til slutt skal Web-Løsningen utvikles med React.* Gruppen har også vært i kontakt med Telia angående deres sporingsenhet, gruppen har forhåpninger om at denne vil egne seg for problemstillingen til oppdragsgiver.

4.3.6 Sprint 6 - 04.03-11.03

UI Design og API

Gruppen har i perioden jobbet for å ferdigstille design for brukergrensesnittet til mobil og web applikasjonen, i tillegg har en plan på nødvendige «endpoints» til API-et blitt laget. Gruppen har også planlagt hvordan systemet skal deployeres på byggeplass: Det skal plasseres en gateway med *GPS* og *LPWAN* funksjonalitet på byggeplass som kommuniserer med små *BLE* tags som er festet til stillaset.

4.3.7 Sprint 7 - 12.03-18.03

API programmering og forarbeid Front-End

Gruppen har i Sprint syv begynt med API utvikling, Aleksander og Martin har begynt på stillasdel og prosjekt endpointene, Tormod setter seg inn i Swift rammeverket slik at han kan begynne utvikling i neste sprint. Gruppen har også hatt et avsluttende møte med Telia hvor de forklarte at deres enhet ikke passet vårt use case, dette traff gruppen meget hardt i og med at dette betyr at gruppen må finne en alternativ data, eller teste systemet med «mock data».

4.3.8 Sprint 8 - 19.03-25.03

API utvikling og mobilprogrammering

[4] Mer om Abax: https://www.abax.com/no

[6] Mer om Haki: https://www.haki.no/

Gruppen har i denne sprinten jobbet videre med API utvikling, stillasdel og prosjekt endepunkt nærmer seg ferdige. Aleksander har også laget ett bruker endepunkt. Tormod jobber videre med mobil-programmering. Bacheloren går som planlagt, men gruppen er noe stresset angående alternativ sporingsenhet.

4.3.9 Sprint 9 - 26.03-01.04

Webutvikling og mobilprogrammering

I Sprint 9 har gruppen jobbet gruppen for å bli ferdig med all API-funksjonalitet som ikke var avhengig av data fra sporingsenheten, Aleksander går nå i gang med Web-utvikling. Martin bruker tid på API-tester slik at gruppen kan forsikre seg om at API fungerer som forventet. Tormod jobber videre med mobil programmering. Tormod har også gått i kontakt med INGICS[7] technologies som produserer en enhet som gruppen tok i bruk som et alternativ til Telias sporingsenhet.

4.3.10 Sprint 10 - 02.04-08.04

Sporingsenhet, web og mobilapplikasjon

I denne sprinten har Martin fokusert på å ferdigstille API funksjonalitet, fokus legges derimot på sporingsenheten slik at gruppen får testet systemet fysisk. Tormod og Aleksander jobber med front-end løsningene. Begge jobber i denne sprinten hovedsakelig med behandling av data fra API og databasen.

4.3.11 Sprint 11 - 09.04-15.04

Sporingsdata, web, mobil og påske

Gruppen har i påskeferien jobbet med å få ut data fra sporingsenheten samt fortsette utvikling av mobil og web applikasjonen. Det ble ikke holdt et offisielt møte etter denne sprinten på grunn av påskeferien. Gruppemedlemmer sto fritt til å jobbe om de hadde lyst.

4.3.12 Sprint 12 - 18.04-22.04

Ferdigstille API, behandle sporingsdata og Front-End

Gruppen jobber iherdig for å ferdigstille API og Front-end løsningene. Vi innser at det vil bli knapt med tid til rapportskriving så dette må påbegynnes i løpet av neste Sprint. I tillegg jobber Martin med å oppdatere stillasdeler i databasen med data fra sporingsenheten. Gruppen håper å bli ferdig med all utvikling den første mai.

4.3.13 Sprint 13 - 23.04-29.04

Avslutning av utvikling og oppstart av rapport

Sprint 13 ble brukt til å planlegge hovedrapporten, gruppen satt opp en struktur med kapitler og diskuterte denne med veileder. I tillegg jobbet gruppen for å bli ferdig med utvikling da fristen for dette er satt til første mai. Gruppen har også i perioden fysisk testet rekkevidden til sporingsenhetene slik at vi kan konkludere om hvorvidt teknologien egner seg for problemstillingen.

[7] Mer om INGICS: https://www.ingics.com/

4.3.14 Sprint 14 – 30.04-06.05

Rapportskriving og kodekommentering

Gruppen brukte nest siste SCRUM sprint på å for alvor begynne rapportskriving. Det ble under sprinten holdt et møte med veileder angående rapportens oppbygning og innhold, det ble finjustert, og kommentert kode da gruppen hadde tid.

4.3.15 Sprint 15 – 07.05-13.05

Rapportskriving og finpuss

Gruppen har brukt siste SCRUM sprint på å få ferdig mye av rapporten, det ble levert et førsteutkast den 08.05 og et andreutkast den 13.05. Gruppen ligger an til å bli ferdig med rapporten innen den 18.05. Gruppen har også finpusset kode når vi hadde muligheten, men hovedfokus har vært rapportskriving.

5. Sporing

Kapittelet vil dokumenterer prosessen rundt valg av sporingsteknologi og anskaffelse av sporingsenheter. Under introduksjonen vil formålet med sporingen defineres, argumentasjon og diskusjon rundt hvilke teknologier som ble vurdert, samt fordeler og ulemper rundt dem. Til slutt vil den valgte sporingsteknologien og enhetene brukt til prototyping bli diskutert.

5.1 Introduksjon

Bacheloroppgaven ble i kap. 3 delt opp i to separate deler. Følgende oppgavebeskrivelse ble gitt av oppdragsgiver:

«Hvordan kan vi spore og telle utstyret vårt med beste tilgjengelige teknologi, slik at vi kan vite akkurat hvilket utstyr som går ut fra et av våre lager til hvilket prosjekt på hvilket tidspunkt, hvilket utstyr som returneres fra hvilket prosjekt til hvilket tidspunkt, i tillegg hvilket utstyr som beveger seg fra prosjekt til prosjekt og dette vil igjen da kunne danne grunnlag for hva som går tapt av utstyret på leveransene»

For å kunne løse dette problemet ønsket oppdragsgiver å finne en billig måte å spore stillasdeler på. Dette var hovedsakelig for å oppnå en oversikt over hvor stillasdeler befinner seg, både for å kunne minke svinn, spare tid og sørge for at fakturering foregår riktig. Gruppen gikk frem ved å se på diverse sporingsteknologier (BLE, RFID, LPWAN, etc.) og hvordan disse kunne anvendes for å oppfylle oppdragsgivers ønsker, samt hvordan en eventuell enhet med denne teknologien ville kommunisere med vårt system.

5.2 Sporingsteknologi og enhet

Kapittelet dokumenterer hvordan gruppen valgte sporingsteknologi og prosessen rundt hvordan gruppen gikk til anskaffelse av enheter.

5.2.1 Oppstart og konkretisering

I de første ukene av bacheloroppgaven ble det kollektivt sett på forskjellige sporingsteknologier og mulige måter å ta disse i bruk på. Gruppen ønsket ikke å lage en skreddersydd sporingsenhet for oppgaven, da dette ville kreve for mye tid, samt at det faller for langt utenfor gruppens fagområde. Istedenfor ble det valgt å sanke informasjon om tilgjengelige teknologier og produkter gjennom tidligere oppgaver, fagartikler og web søk, for så å velge en sporingsteknologi og eventuell enhet som passet problemstillingen.

De første møtene med oppdragsgiver ble brukt på å konkretisere hvilke aspekter av sporingen som var av høyest prioritet gitt at en sporingsløsning som oppfyller alle krav fra kravspesifikasjonen ikke var realistisk. Gjennom dialog og møter med oppdragsgiver ble det bestemt at en fremtidsrettet teknologi med god rekkevidde og enkel oppkobling var ønskelig. Pris var en viktig faktor, men oppdragsgiver mente at dette ikke skulle begrense valget av teknologi.

Det ble tidlig i prosessen avklart at pris var problematisk da oppdragsgiver ønsket en pris på en femtedel av stillasdel prisen eller mindre per enhet. På grunn av dette ble det diskutert muligheter for å kun spore «Bareller», kassene som stillasdelene ble fraktet i (se Figur 11). Dermed ville muligheten for aktiv og kontinuerlig sporing med for eksempel GPS være aktuell. Oppdragsgiver var enig i at «barellene» kunne inkluderes og eventuelt brukes for å oppbevare komponenter eller annen «middleware» som kreves for løsningen. Det var derimot ønskelig å spore stillasdelene individuelt da «bareller» sjeldent forsvinner og fraktes ofte vekk fra byggeplassen etter montering.

Figur 11 Barelle for transport av stillasdeler

Med et klarere fokusområde kunne gruppen ta for seg de forskjellige tilgjengelige sporingsteknologiene. Følgende teknologier ble vurdert, men skrinlagt da de ikke egnet seg for bruksområdet.

Strekkoder

- Meget billige, men krever at en fysisk skanner er veldig nær koden og kun én kode kan leses samtidig, noe som gjør teknologien uaktuell for alt annet enn varetelling.

NFC

- Kan brukes for å identifisere individuelle deler på byggeplass, men krever derimot en avstand på maksimalt 20 cm.

QR-koder

- Kan brukes for å identifisere individuelle deler på byggeplass, men krever at koden er på lesbart hold og kan sees av et kamera.

UWB

- Var en av teknologiene som var meget aktuell for problemstillingen, men etter mer undersøkelse ble det avdekket at teknologiens rekkevidde ikke var stor nok og høy pris gjorde teknologien uaktuell for løsningen (se kap. 2.2.6).

GPS

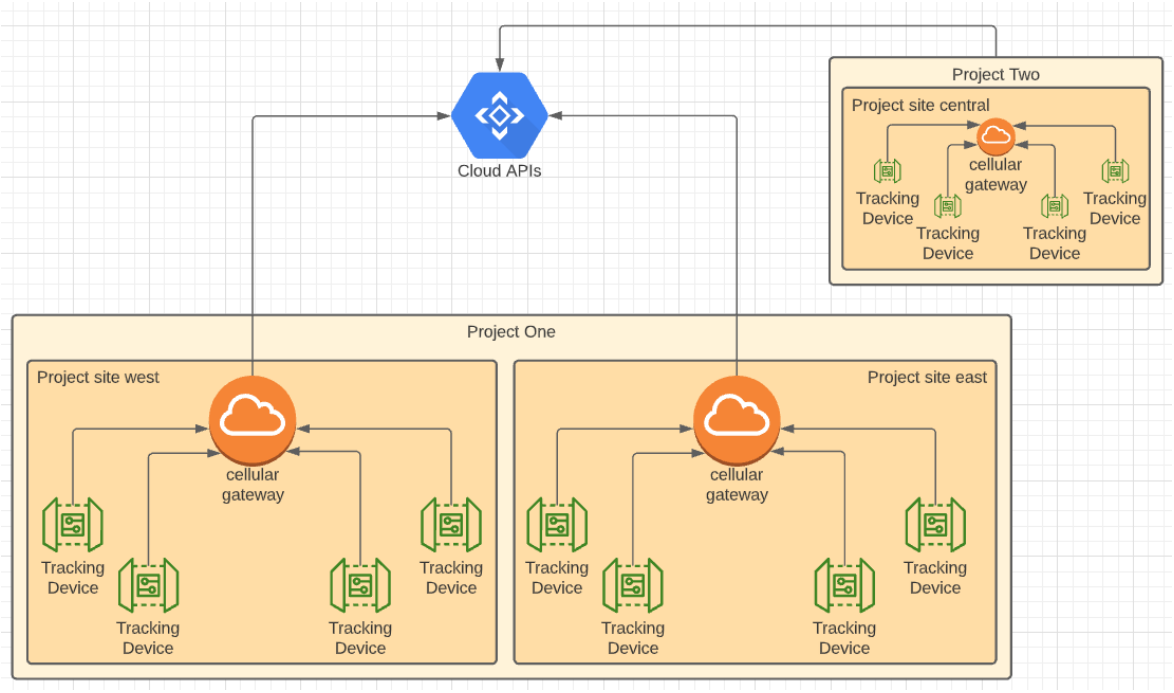
- Nøyaktig posisjon, men dette gikk på bekostning av batteritid og kostnader
- Tilstrekkelige enheter ville bli for store til å montere på stillasdelene gruppen jobber med.

5.2.2 Valg av teknologi

Etter tre uker med undersøkelser rundt de tilgjengelige teknologiene, sto gruppen igjen med tre aktuelle sporingsteknologier:

- Radio Frequency Identification (RFID)
- Low Powered Wide Area Network (LPWAN)
- Bluetooth Low Energy (BLE)

På dette stadiet var det lagt en plan for den fysiske arkitekturen på løsningen. Denne gikk originalt ut på å ha tags som kommuniserte direkte med API-et for å slippe bruk av «middleware». Etter nøyere undersøkelser på gjennomførbarhet innså gruppen at det var behov for et mellomledd som kunne distribuere informasjon fra taggene til skyen, noe som resulterte i arkitekturen vist i Figur 12.



Figur 12 Planlagt system-arkitektur sporing

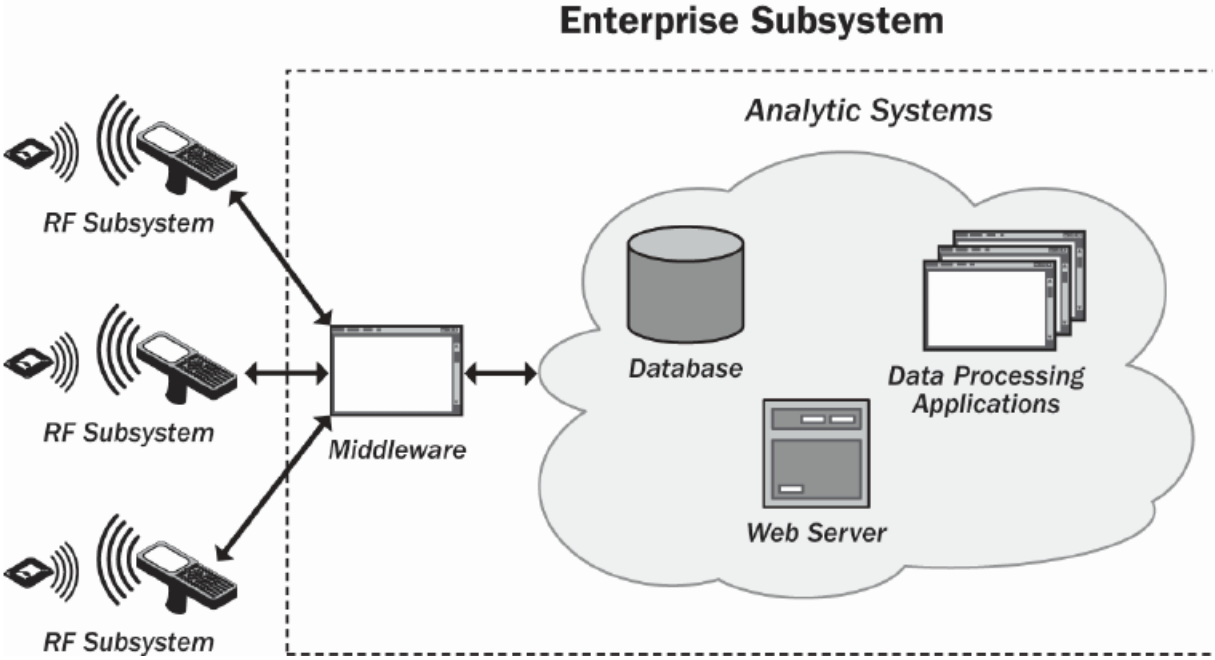
5.2.2.1 *LPWAN*

LPWAN ble undersøkt tidlig, da teknologien i motsetning til de andre sporingsteknologiene tok i bruk 4/5G og telekommunikasjon via mobilnettet. Gruppen så dette som meget fordelaktig da oppdragsgiver ikke ville ha tilgang til internett på alle byggeplasser. LPWAN er også spesiallaget for IoT noe som gjør det veldig aktuelt for oppgavens problemområde. *Telia* var en av få aktører i Norden som tilbyr denne teknologien og gruppen gikk derfor i dialog med selskapet for å få mer informasjon om hvorvidt teknologien kunne anvendes for gruppens problemstilling. Etter møter og diskusjon med *Telia* ble det bestemt å ta i bruk LPWAN som et mellomledd mellom taggene/Beacons og API-et. Prisestimater og diskusjon med *Telia* tilsa at løsningen måtte ha en Gateway-enhet (sporingsenhet på byggeplass som samler og videresender informasjon fra mindre Beacons via LPWAN). *Telia* ønsket å tilby denne LPWAN Gateway-en som et mellomledd i løsningen.

5.2.2.2 *RFID*

Tidlig i prosessen ble det vurdert å basere sporingen på RFID. Teknologien er gammel, men uten tvil den mest anvendte teknologien innenfor sporing av billige verdier. Dette gjorde teknologien meget aktuell for løsning av problemet. Gruppen fant derimot ut at en

implementasjon med RFID hadde begrensninger. RFID tagger krever i de fleste tilfeller manuell skanning for å registreres, noe som vil føre til mye manuelt arbeid ved oppskalering av systemet. Gruppen så på muligheter for oppsett av RFID-porter[8] hvor taggene blir registrert ved avreise og ankomst til lager. Dette løser derimot kun logistikkproblemet på lager hos oppdragsgiver.



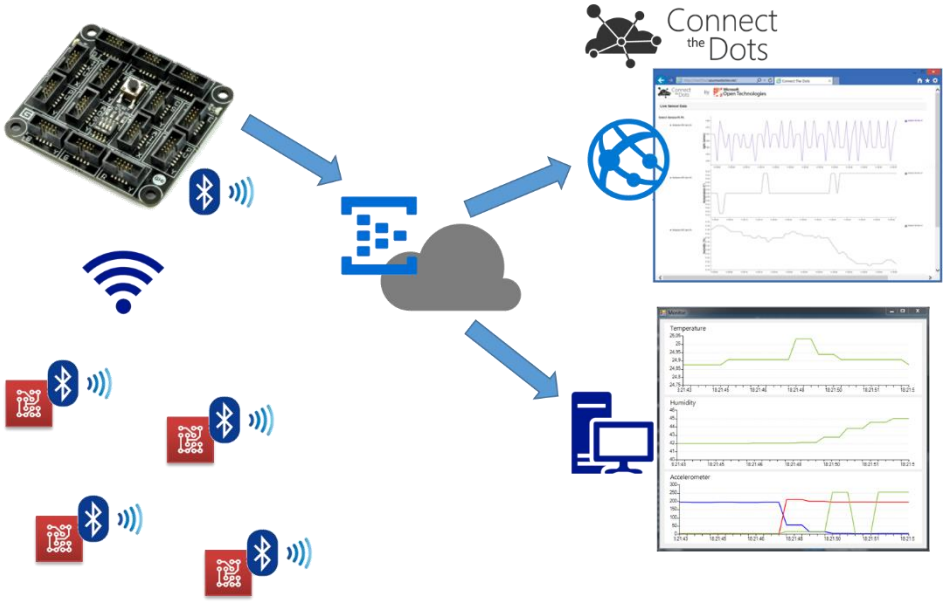
Figur 13 Eksempel på RFID Arkitektur tatt fra https://www.researchgate.net/profile/Juraj-Vaculik-2/publication/276431870/figure/fig1/AS:294532718579713@1447233480208/RFID-system-architecture.png

Gruppen så også på muligheter for bruk av aktive RFID tagger (se kap. 2.2.3 for mer info), men batteritiden på disse var begrenset, samt at pris øker betraktelig ved bruk av aktive tagger kontra passive tagger. Gruppen så også på muligheten for å lage et lese-system på «barellene» som stillaset fraktes på. Dette ville muliggjøre skanning av stillasdeler på byggeplass, men systemet ville hatt for mange komponenter, samt at det ville gjøre overføring av stillasdeler internt mellom prosjekter vanskelig.

5.2.2.3 *BLE*

Gitt at gruppen nå hadde tilrettelagt oppgaven for å spore stillaset ved hjelp av en Gateway og tags på hver del, trengte vi kun å finne en aktuell enhet vi kunne plassere på de ulike stillasdelene. BLE var den første teknologien som ble sett på som en aktuell løsning av problemet. Den tilbød aktiv sporing på spesifiserte intervaller, i tillegg har BLE en stor rekkevidde avhengig av terreng og antenne. Teknologien har også blitt brukt i mange selskaper som et verktøy for sporing av små verktøydeler og verdier. Dette i kombinasjon med at Bluetooth er en teknologi som utvikler seg hurtig gjorde den meget aktuell for gruppen.

[8] Se mer om RFID Porter: https://www.aucxis.com/en/rfid/rfid-products/rfid-gates



Figur 14 BLE Arkitektur https://github.com/ppatierno/ble2azure/blob/master/images/overall.png

Gruppen gikk i dialog med *ABAX*[9], en av Nordens ledende bedrifter innenfor sporing av verdier for å få mer informasjon om deres ABAX Mini2, en BLE Beacon som lignet det gruppen var ute etter. Gruppen fikk informasjon angående hvordan deres sporingsenhet fungerte[10]. Dette møtet ga mer innsikt rundt rekkevidde, innvirkning av interferens, samt muligheter og begrensninger rundt teknologien. Teknologien krevde ingen bindingsprosess (master/slave tilkobling) mellom Gateway-enhet og tagger noe som ville gjøre skalering av systemet enkelt i forhold til RFID. På grunn av disse momentene ble det valgt å ta i bruk BLE for taggene på stillaset, de ville gjøre overføring av deler automatisk, samt at oppkobling av mange enheter ville være raskt og effektivt.

5.2.3 Anskaffelse av enhet

Etter å ha bestemt hvilke teknologier som skulle tas i bruk mailet gruppen *Telia* angående anskaffelse og pris på LPWAN Gateway enheten de tilbudte. De følgende to ukene undersøkte gruppen de forskjellige fordelene og ulempene rundt LPWAN og BLE. Det ble holdt to oppfølgingsmøter som ble brukt til å spesifisere hvilken funksjonalitet oppdragsgiver og gruppen ønsket fra *Telia*. Salgspersonen gruppen var i kontakt med kommuniserte at enheten støttet både LPWAN (i form av LTE-M) og BLE. Gruppen ble enige om å bestille denne enheten og det ble satt opp et møte angående bestilling av enhet. I dagene før bestilling skulle finne sted kalte *Telia* inn til et nytt møte 15.03.2022 hvor det ble diskutert funksjonalitet med en konsulent hos selskapet. *Telia* forklarte at enheten de kunne tilby dessverre ikke hadde funksjonaliteten som hadde blitt lovet av salgsansvarlig. Enheten hadde i spesifikasjonsdokumentet støtte for GPS og LPWAN, samt BLE, men dette var ifølge *Telia* en feil. Enheten hadde kun LPWAN og GPS støtte og kunne ikke brukes som en Gateway for BLE tagger, ettersom dette ikke ble støttet i *Telias* digitale infrastruktur.

[9] Mer om Abax her: https://www.abax.com/no/om-abax
[10] Se info om ABAX Mini2: https://www.abax.com/no/hw/mini2https://www.abax.com/no/hw/mini2

Dette ledet til at gruppen måtte finne en ny Gateway som kunne benyttes for testing. Gruppen fant ved undersøkelse selskapet *INGICS Technologies* som tilbød en pakke med både en BLE Gateway med LTE-M (se kap. 2.2.1) funksjonalitet, samt BLE-tagger som kommuniserte med Gateway-en. Grunnet begrenset tid til å finne flere enheter, samt at *INGICS*-enhetene oppfylte kravene for prototyping, ble disse enhetene tatt i bruk for prototyping.

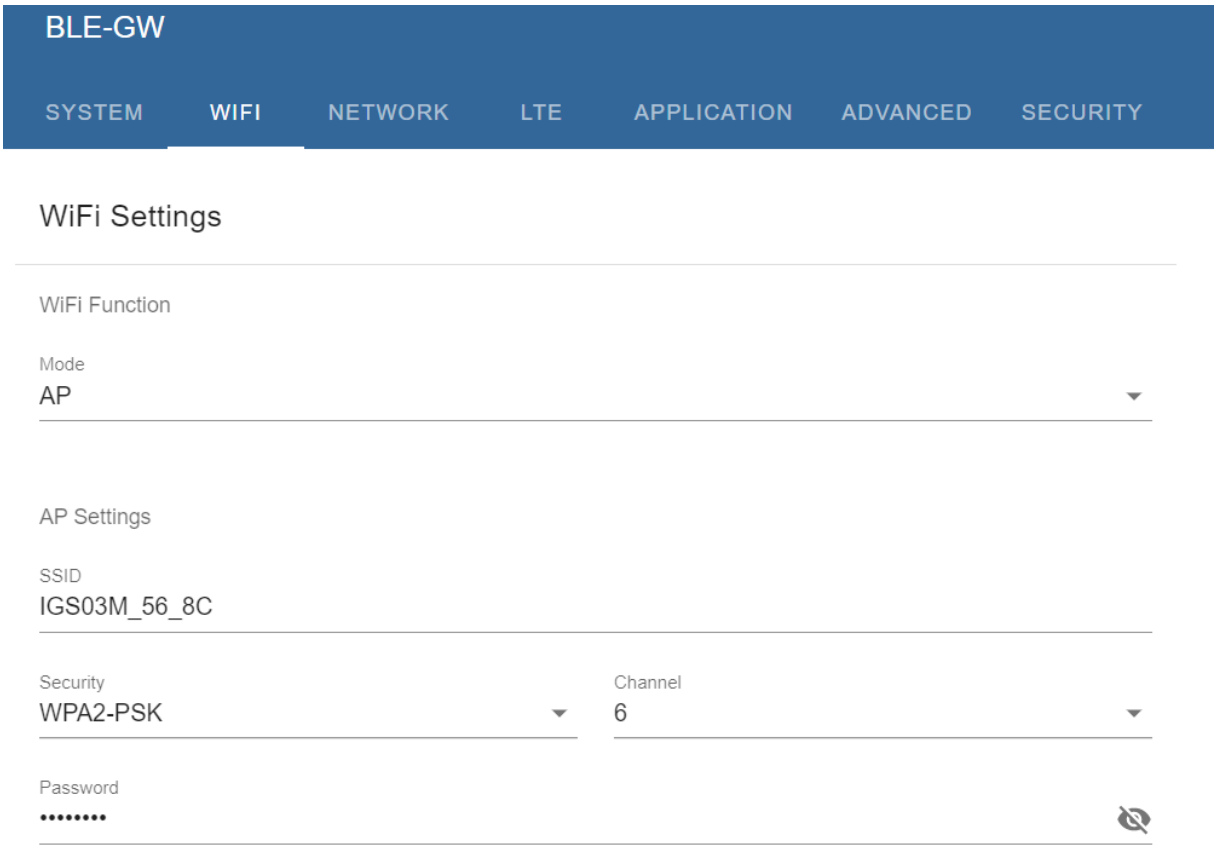
5.3 Resultat

Gruppen valgte å basere løsningen på en kombinasjon av BLE og LPWAN. Protorypeløsningen tar i bruk IBS05 Beacons/tagger, dette er små BLE tagger som automatisk sender ut datapakker (se Kode-utdrag 1) som inneholder diverse data (se kap. 2.3.2 for detaljert informasjon om datapakken). Disse taggene vil bli utplassert på stillasdeler slik at API-et kan registrere hvor mange typer av hver stillasdel som befinner seg innenfor rekkevidden til en gitt Gateway.

```
$GPRP,D85CEAE974B6,34AB954B568C,-59,02010612FF2C0883BC2D0100AAAAFFFF000030030000
$GPRP,FA2AE822D7D9,34AB954B568C,-60,02010612FF2C0883BC2D0100AAAAFFFF000030030000
$GPRP,FFE5AF978A71,34AB954B568C,-66,02010612FF2C0883BC2D0100AAAAFFFF000030030000
$GPRP,EB1FA39B701A,34AB954B568C,-59,02010612FF2C0883BC2D0100AAAAFFFF000030030000
$GPRP,E4867AB2D998,34AB954B568C,-66,02010612FF2C0883BC2E0100AAAAFFFF000030030000
```

Kode-utdrag 1 IBS05 datapakke

INGICS IGS03M, en BLE Gateway blir utplassert på byggeplass, denne enheten lytter etter BLE Beacons og sender informasjon om taggene den plukker opp. Enheten har et enkelt brukergrensesnitt som blir tatt i bruk ved oppsett av systemet (se Figur 15 IGS03M BrukergrensesnittFigur 15).



Figur 15 IGS03M Brukergrensesnitt

Denne Gateway-en kommuniserer med API-et via HTTP-POST og sender alle registrerte tagger på et satt intervall. Vi konfigurerte også Gateway-en slik at den kun plukket opp IBS05 Beacons. På denne måten vil Gateway-ene kun plukke opp denne typen tags. Dette var ikke nødvendig for funksjonalitet i API-et, men gitt at Gateway-en sender informasjon utenfor intervallet om enhetens lokale minne blir fult. Det er derfor fordelaktig at det kun er tagger av typen IBS05 som blir plukket opp, slik at enheten kun sender informasjon i det spesifiserte intervallet.

Med dette oppsettet kan stillasmontører hos oppdragsgiver enkelt flytte på stillasdeler både på byggeplass, mellom byggeprosjekter og stillasdelen vil bli plukket opp automatisk om det er satt ut en Gateway på byggeplassen (for informasjon om hvordan API-et håndterer. Gateway datapakken se kap. 6.2.2).

Gruppen fikk følgende resultater på testing og rekkevidde (for detaljert testing se kap. 8.4):

Område	**Plassering av Gateway**	**Plassering av tag**	**Obstruksjoner**	**Resultat**
NTNU Gjøvik klasserom	Bord på klasserom	Gang utenfor klasserom	Betongvegger og annen interferens	Rekkevidde 30 meter
Parkeringsplass	Enden av parkeringsplassen	Andre side av parkeringsplass	Biler	Rekkevidde 120 meter
Lagerplass MB-Stillas på Hamar	Ene siden av lagerplassen	Festet på stillasdel med	Stål og stillas	70 meter

		antenne vekk fra Gateway		
Lagerplass MB-Stillas på Hamar	Ene siden av lagerplassen	Festet på stillasdel med antenne vekk fra Gateway	Aluminium	15 meter

Tabell 5 Oppsummering av testing

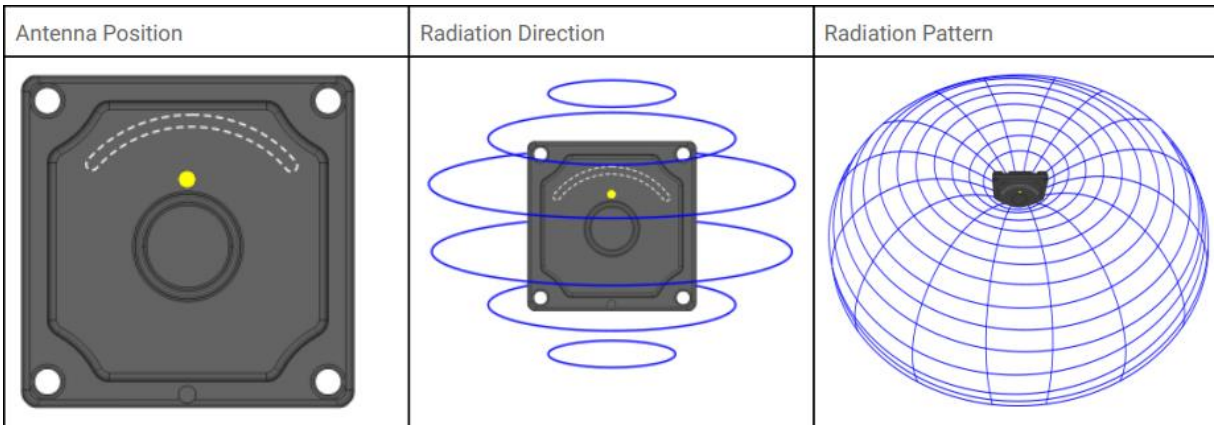
5.4 Diskusjon

Ser gruppen tilbake på resultatene satt opp imot målene mener vi at sporingsteknologien løser problemet på en god måte. BLE fungerer ypperlig da tagger ikke krever manuell oppkobling mot Gateway-ene. Dette gjør at forflytning av stillasdeler mellom Gateway-er fungerer uten problemer. En stillasmontør kan ta med seg et spir med en montert tag fra byggeplass A, frakte den til byggeplass B og så lenge en Gateway er plassert på byggeplass B vil byggeprosjektets mengde stillasdeler bli oppdatert. Oppdragsgiver vil da til enhver tid kunne få informasjon angående hvor mange stillasdeler som befinner seg på de forskjellige byggeprosjektene.

Figur 16 IGS03M Antenne arkitektur, bilde funnet i dokument: https://www.ingics.com/doc/Gateway/GW0038_Beacon_Gateway_iGS03W_E_M_User_Manual.pdf

Gruppen mener derimot for at løsningen skal kunne deployeres på byggeplass bør en annen Gateway tas i bruk. Gateway-en som er brukt for prototyping fungerer for testing, «proof of concept» og for å teste at IT-systemet fungerer som tiltenkt, men hverken Gateway-en eller taggene er robuste nok til å tåle håndteringen stillaset blir utsatt for. Videre er den også avhengig av strømtilførsel og sender ikke ut egne koordinater. Dette er derimot noe vi vet tilbys av andre Gateway-er. Gruppen mener at for å oppnå en optimal løsning må oppdragsgiver gå i dialog med en produsent av BLE tagger for å spesial-lage en mer robust enhet som kun inneholder funksjonaliteten oppdragsgiver har behov for.

Rekkevidden oppnådd med enhetene er noe upålitelig ettersom testingen ikke har stort nok datasett. Om oppdragsgiver skal ta i bruk Gateway-ene brukt for prototyping vil de ha behov for opp til flere Gateway enheter per byggeplass gitt at nåværende resultater stemmer, men valg av andre Gateway-er kan gi bedre resultater. Gruppen finner det sannsynlig å tro at den noe begrensede rekkevidden er en konsekvens av størrelsen på antennen både på IBS05 taggene og IGS03M Gateway-en (se Figur 16 og Figur 18).



Figur 17 IBS05 Antenne posisjon og bølgemønster

Hver Gateway kan maksimalt dekke et areal på 45 238 kvadratmeter, 120 meter (oppnådd under testing i åpent lende) i radius gir 3,14*120*120. Et mer realistisk estimat på dekningsareal er 11 310 kvadratmeter, 3,14*70*70 (70 meter oppnådd under testing med obstruksjoner og stål) da det for dette arealet regnes med rekkevidden registrert da taggene var festet på stillas. Signalet ble også svekket betraktelig når taggene ble festet på stillasdeler laget av aluminium. En mulig løsning på dette vil være montering av en forsterker som fungerer med taggene slik at en akseptabel rekkevidde kan oppnås til tross for interferens med aluminiumen. Dette er derimot ikke testet og er foreslått på bakgrunn av undersøkelser rundt signalforsterkning av BLE tagger.

Gruppen ønsket også at Gateway-enheten skulle sende ut koordinatene Gateway-en befant seg på. Dette ville både gjøre sporingen mer detaljert og automatisk (i vår leverte løsning - med prototype enhetene - må brukeren selv legge inn Gateway-en med koordinater), samt at det også ville være mulig å estimere posisjonen til taggene ved hjelp av signalstyrke og flere Gateway-er. Dette ville gjort det lettere å finne igjen en eventuell bortkommen stillasdel da løsningen per nå kun rapporterer om stillasdelen er innenfor rekkevidden til en Gateway.

6. Back-End

Back-end kapittelet vil ta for seg utviklingsprosessen av både gruppens database og API. De blir delt opp i to separate kapitler og diskutert separat. Databasedesign og implementasjon vil bli diskutert først, da det er viktig at brukeren har en overordnet forståelse for hvordan databasen er strukturert og implementert. Deretter vil API-ets design og implementasjon diskuteres. Kapitlene vil ta for seg all relevant implementasjon, men testing, sikkerhetsaspekter og feilhåndtering vil bli diskutert i kap. 8 sammen med testing av sporingsenhetene. Lengden på kapittelet skyldes innhold av figurer og kodesnutter, noe som vil gjøre det lettere å sette seg inn i.

6.1 Database

Bruksområde, optimalisering og krav fra oppdragsgiver, har blitt tatt i betrakting når det kommer til valg av database. I kapittelet under vil design og implementasjon drøftes. For ytterligere innsikt i det endelige databasedesignet, se Appendix L.

6.1.1 Design

Starten av design fasen ble brukt til å kartlegge hvilke data som skulle lagres. Gruppen så fort at det var fire kategorier som måtte lagres: lokasjoner, stillasdeler, brukere, og Gateway-er. Dataen som skulle lagres i de ulike kategoriene var ikke gitt av oppdragsgiver, disse var avhengig av systemet som skulle lages.

Prosjekt	Ansatt	Stillasdel	Gateway
- Kunde - Adresse - Lokasjon - Periode - ProsjektID - Prosjekt navn - Størrelse - Status - Geofence	- Navn - Telefonnummer - Email - AnsattID - Fødselsdag - Rolle - Admin rettigheter	- ID - Type - Prosjekt - Batteri nivå	- ID - Lokasjon - ProsjektID - Prosjekt navn - Status

Tabell 6 Felt-eksempler databasedesign

Tidlig i designfasen var ikke Gatway-er en del av løsningen. Gruppen så på muligheten for å innlemme stillasdelene som en map i prosjektdokumentet, for å knytte stillasdeler til prosjekt. Med dette designet måtte stillasdeler fjernes og flyttes mellom prosjektdokumentene (se Kode-utdrag 2 Eksempel på hvordan stillas kunne blitt innlemmet i prosjektdokumentet.).

```
{
    "projectID": 1,

        ...

    "stillas":{
    [
        {
            "stillasID": "4A8C62"
        },
        {
```

```
                "stillasID": "22D7D9"
            },
            {
                "stillasID": "978A71"
            }
        ]
    }
}
```

Kode-utdrag 2 Eksempel på hvordan stillas kunne blitt innlemmet i prosjektdokumentet.

Hvordan dette skulle løses i praksis måtte tas en vurdering på når valget av database ble tatt. Gruppen ønsket å designe databasen lik API-et, dataen kunne dermed hentes med kun én spørring. Det ble derfor valgt å holde seg til fire kategorier. Dette gjorde også systemet mer oversiktlig.

6.1.2 Implementasjon

Gruppen valgte å gå for en NoSQL database (se kap. 6.1.4). *Google Firestore* databasen ble brukt til lagring av data, da gruppemedlemmene allerede er kjent med strukturen og implementasjonen av databasen i programmeringsspråket Golang.

Under implementasjonen måtte strukturelle valg tas for inndelingen av data. Gruppen la vekt på hastighet ved henting av data og integritet under implementasjon. Måten dette kunne optimaliseres på gjennom strukturen, var å legge inn informasjon i en kolleksjon innad i dokumentet, fremfor å ha det som datafelt i dokumentet.

6.1.3 Resultat

Dersom vi ser tilbake på kravspesifikasjonen (se kap. 3.1) har det blitt lagt inn nødvendig data for å oppnå kravene satt av ved starten av oppgaven. Lokalisering av stillasdeler og lagerlogistikk blir håndtert med dataen som er lagret. Databasen har derimot mulighet til optimalisering ved bruk av referansesystemer. (se kap. 10)

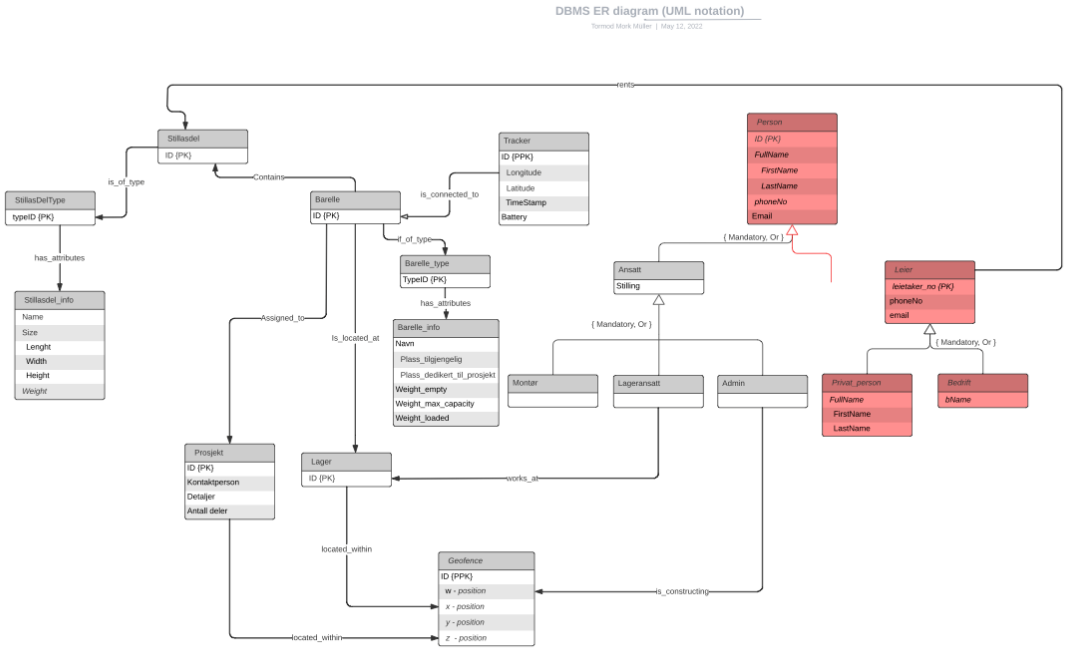
6.1.4 Diskusjon

Valget av databasen var påvirket av kravene som gruppen satt opp sammen med oppdragsgiver, samt andre tekniske krav gruppen mente var vesentlige i et slikt system. Disse kravene var:

- Mulighet for vertikal og horisontal skalering
- Atomiske operasjoner
- Fremtidsrettet teknologi og design
- Infrastruktur som kan håndtere mange hentinger
- Enkel integrasjon med et API

Valget stod mellom en relasjons- eller dokumentorientert database (se kap. 2.5). Gruppen har tidligere erfaring med relasjonsdatabase, gjennom faget *Datamodellering og databasesystemer*[11]. Gruppen tok en beslutning om at en relasjonsdatabase ville vært best egnet for å sikre integriteten på systemet. Tidlig i utviklingen ble det designet et førsteutkast av en konseptuellmodell for relasjonsdatabasen. (se Figur 18)

[11] Mer om faget her https://www.ntnu.no/studier/emner/IDATG2204#tab=omEmnet



Figur 18 Tidlig utkast av konseptuell modell, en større versjon av denne figuren finnes i appendix L

Etter diskusjon med oppdragsgiver og professorer ble det mer hensiktsmessig å benytte seg av en dokumentorientert database ved utvikling av et API. En dokumentorientert database vil gjøre det enklere for oppdragsgiver å skalere systemet vertikal og horisontal, siden dette vil være en database i stor vekst.

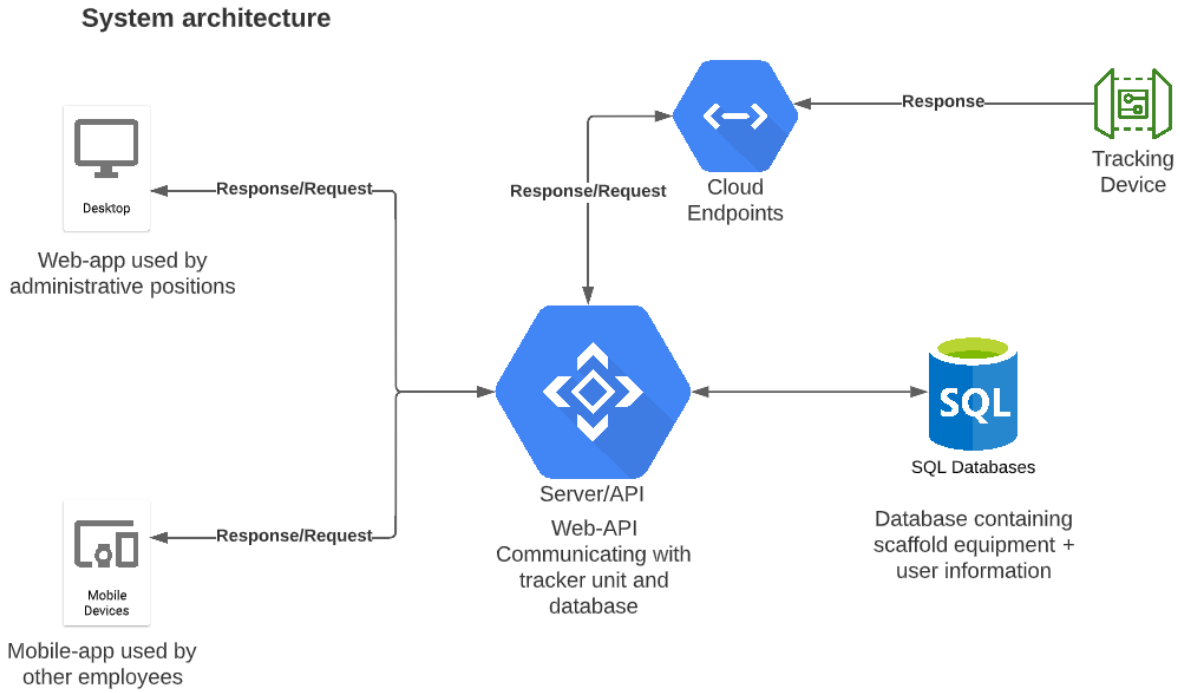
Valget av en dokumentorientert database, vil gå på bekostning av dataintegritet. For å minske skaden dette kunne forårsake, måtte gruppen designe databasen med minst mulig duplisering av data, da det ofte er her dataintegriteten feiler. I tillegg ble det brukt cloud functions (se kap. 2.5.3). Bruken av cloud functions gjorde det mulig å gjøre transaksjoner mellom flere dokumenter atomiske.

Antall lesinger som kreves i en dokumentorientert database vil variere basert på databasedesignet, men i de fleste tilfeller kreves det flere lesinger enn i en relasjonsdatabase. Ettersom dataen i en dokumentorientert database er semi-strukturert, vil ikke spørringer resultere i en tabell med resultater. For å hente frem de ønskede dokumentene, må store deler av dataen itereres gjennom. Gruppen ønsket å minimere lesingen mest mulig ved å legge inn kolleksjoner for å hente ut data av lik kategori. Dette minimerer antall lesinger i databasen, ved å hindre over henting av data, samt at hentingen blir begrenset til en gitt kategori.

6.2 API

Løsningens hovedkomponent er API-et, dette er en del av oppgaven oppdragsgiver var svært opptatt av da funksjonalitet i API-et skal brukes til videre utvikling/integrering i selskapets andre systemer. Kapittelet vil ta for seg API-ets design, hvordan strukturen er lagt opp samt hvordan API-et kommuniserer med både database, sporingsenhet og front-end. Deretter vil implementasjonen av API-et bli forklart før gruppen til slutt diskuterer diverse valg tatt under utvikling.

Figur 19 Overordnet system arkitekturFigur 19 viser hele systemets arkitektur, diagrammet illustrerer hvor essensielt API-et er for gruppens løsning da det er kommunikasjonsleddet imellom alle andre komponenter i systemet.



Figur 19 Overordnet system arkitektur laget i lucidchart

6.2.1 Design

Gruppen valgte å følge *Microsofts* REST API standarder[12] for design av API-et. Retningslinjene ble brukt for å definere både endepunktene til applikasjonen, body-strukturen for POST, PUT og DELETE forespørsler og API-ets responsstruktur.

6.2.1.1 Endepunkt design:

Før utviklingen satte gruppen opp en dokumentasjons side i *Postman*[13] hvor de forskjellige endepunktene til API-et ble lagt inn. Dette ble gjort slik at en fremtidig utvikler enkelt skal kunne gå inn i API dokumentasjonen for å se de tilgjengelige endepunktene i API-et. I tillegg ville en klar definisjon på de forskjellige endepunktene til API-et gjøre Front-end utvikling lettere siden utvikleren enkelt kan gå inn og se de ulike endepunktene i dokumentasjonen.

[12] Link til microsofts REST api design: https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design
[13] Mer om postman verktøyet her: https://www.postman.com/

For definisjon av endepunkter ble user-stories (se kap. 3.1.2) og kravspesifikasjonen tatt i bruk. Dermed kunne gruppen være sikre på at all avtalt funksjonalitet kom med under design fasen av API-et. Hovedmålet med API-et er å hente ut, oppdatere og lagre informasjon i løsningens database (se kap. 6.1) Endepunktene er av fire typer forespørsler:

- **GET**: Henter ut informasjon fra databasen
- **POST**: Sender inn og oppdaterer informasjon i databasen
- **PUT**: Oppdaterer informasjon i databasen
- **DELETE**: Fjerner informasjon i databasen

Eksempelvis satte gruppen opp et endepunkt for å kunne hente ut et byggeprosjekt ved hjelp av Use-caset: Sjekke stillasdeler. Endepunktet ble definert som følger:

Endepunkt: stillastracker/v1/api/unit?type=Bunnskrue&id=4A8C62
Body: Gitt at dette er en GET forespørsel vil det ikke være en body assosiert med denne forespørselen.
Response:

```
{
    "type": "Bunnskrue",
    "project": "CCHamar",
    "batteryLevel": 3.03,
    "tagID": "4A8C62"
}
```

Kode-utdrag 3 Eksemepel respons body for stillasdel endepunkt

Alle API-ets GET forespørsler er av samme struktur som eksempelet over, for endepunktene der informasjon må oppdateres blir endepunktene sendt inn med en body på følgende måte:

Endepunkt: stillastracker/v1/api/project
Eksempel body:

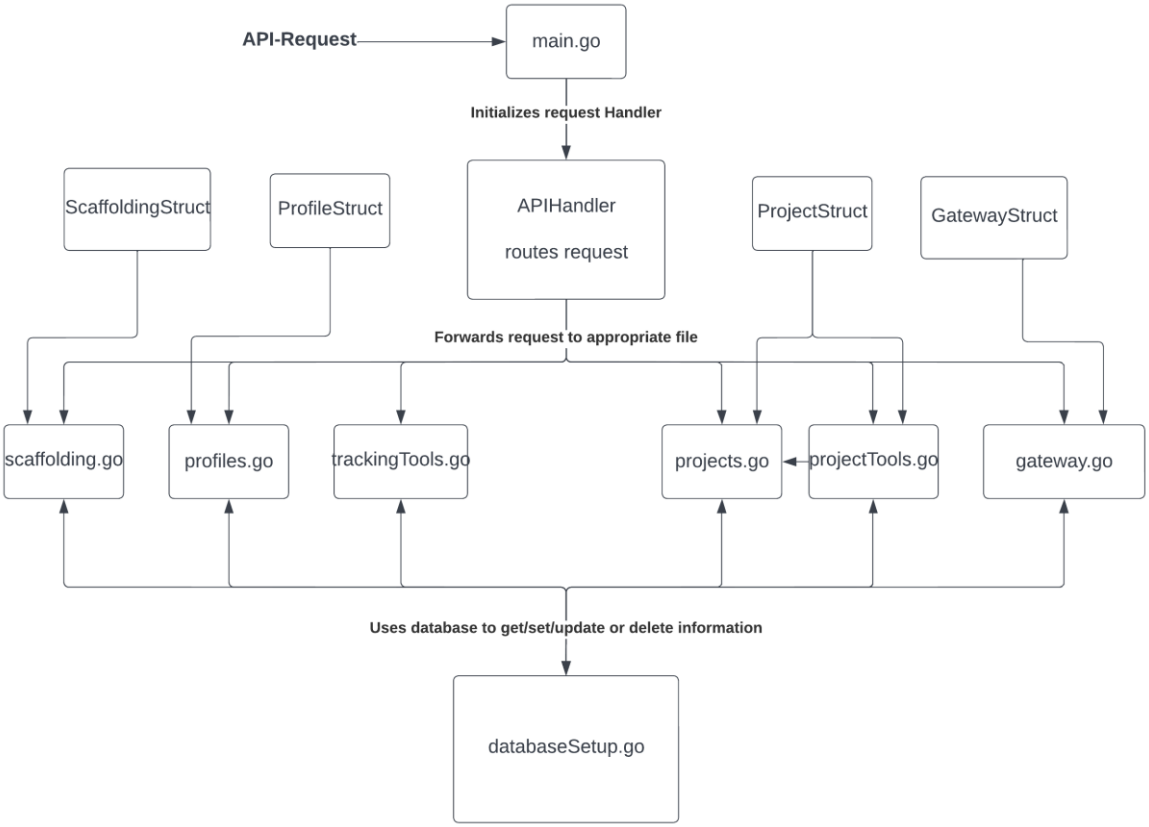
```
{
    "projectID":2321112,
    "projectName":"MBStillas",
    "latitude":60.79077759591496,
    "longitude":10.683249543160402,
    "state":"Active",
    "size":322,
    "period":{
        "startDate":"25-04-2022",
        "endDate":"30-04-2022"
        },
        "customer":{"name":"Martin Iversen",
        "number":98435621,
        "email":"martin@mail.no"
        },
    "address":{
        "street":"Halsetsvea 40",
        "zipcode":"2323",
        "municipality":"Ingeberg",
        "county":"Innlandet"
```

```
        },
    "geofence":{
        "w-position":{
            "latitude":-73.98326396942211,
            "longitude":40.69287790858968
            },
            "x-position":{
                "latitude":-73.98387551307742,
                "longitude":40.6922433936175
                },
            "y-position":{
                "latitude":-73.98255586624245,
                "longitude":40.691999347788055
                },
            "z-position":{
                "latitude":-73.98124694824298,
                "longitude":40.69267453906423
                }
        }
}
```

Kode-utdrag 4 Eksempel forspørsel Body for prosjekt endepunktet

Response: “Project was successfully added”.

6.2.2 Implementasjon



Figur 20 Dataflyt i API-et

6.2.2.1 *Grunnleggende forespørselshåndtering:*

Gitt at all funksjonaliteten til API-et baserer seg på den samme forespørselshåndteringsmetodikken vil gruppen forklare arbeidsflyten til denne prosessen i detalj. Alle API-forespørsler blir behandlet i *APIHandler.go* og sendt videre til korresponderende endepunkts-klasse (se Figur 20 for overordnet dataflyt i API-et), for å implementere dette, krever systemet en ruter og definerte endepunkter som ruteren kan ta i bruk:

```
func Handle() {
    InitLog()

    router := mux.NewRouter()

    //Scaffolding endpoint
    router.Path(baseURL+"/unit").HandlerFunc(ScaffoldingRequest).Queries("type",
"{type}").Queries("id", "{id}") //GET POST PUT DELETE
    router.Path(baseURL+"/unit").HandlerFunc(ScaffoldingRequest).Queries("type",
"{type}")                       //GET POST PUT DELETE
    router.Path(baseURL + "/unit").HandlerFunc(ScaffoldingRequest)
//GET POST PUT DELETE
```

Kode-utdrag 5 Initialisering av Handleren og definisjon av endepunkter ved hjelp av Gorilla Mux

Derfor har gruppen valgt å ta i bruk *Gorilla Mux* (se kap. 2.4.2), et tredjeparts ruter bibliotek både for å initialisere en ruter, samt for å kunne definere endepunkter med spørringer uten

problemer. Funksjonen *Handle* initialiserer en ruter med *mux.NewRouter()* og definerer endepunkter ved hjelp av *router.Path()*, ruteren har dermed en liste med gyldige endepunkter en utvikler kan ta i bruk (se Kode-utdrag 5).

Funksjonen initialiserer også *Initlog()*. Denne funksjonen initialiserer en log fil og definerer tre loggtyper, *InfoLogger* som blir brukt når endringer skjer i API-et, *DatabaseLogger* som tas i bruk når det skjer en endring i databasen og *ErrorLogger* som tas i bruk når det skjer en feil i API-et. Funksjonen lar gruppen kategorisere informasjonen de logger på en ryddig måte (se Kode-utdrag 6).

```
func InitLog() {
   file, err := os.OpenFile("logs.txt",
os.O_APPEND|os.O_CREATE|os.O_WRONLY, 0666)
   if err != nil {
      log.Fatal(err)
   }
   log.SetOutput(file)

   InfoLogger = log.New(file, "INFO:",
log.Ldate|log.Ltime|log.Lshortfile)
   DatabaseLogger = log.New(file, "DATABASE:",
log.Ldate|log.Ltime|log.Lshortfile)
   ErrorLogger = log.New(file, "ERROR:",
log.Ldate|log.Ltime|log.Lshortfile)
}
```

Kode-utdrag 6 Funksjon InitLog() forklart over

Det er disse filene som er hovedkomponentene i API-et, filene tar imot en forespørsel og basert på hva som blir passert inn, vil filen hente, oppdatere eller slette informasjonen filen er ansvarlig for. Alle disse filene starter med en generell funksjon som videresender forespørselen til passende funksjon baser på typen forespørsel (GET, PUT, POST eller DELETE).
Koden under illustrerer hvordan dette fungerer, funksjonen setter passende *Headere* slik at mobil- og webapplikasjonene får tilgang til funksjonene. Deretter leser funksjonen av forespørselens metode med *r.Method* og basert på dette blir forespørselen sendt til korrekt funksjon (se Kode-utdrag 7).

```
func ProjectRequest(w http.ResponseWriter, r *http.Request) {
   //Sets headers to communicate with mobile and web application
   w.Header().Set("Content-Type", "application/json")
   w.Header().Set("Access-Control-Allow-Origin", "*")
   w.Header().Set("Access-Control-Allow-Methods", "POST, GET, OPTIONS,
PUT, DELETE")
   w.Header().Set("Access-Control-Allow-Headers", "Access-Control-Allow-
Headers, Origin,Accept, X-Requested-With, Content-Type, Access-Control-
Request-Method, Access-Control-Request-Headers")

   //Defines the path to the project Documents in the firestore database
   ProjectCollection = database.Client.Doc(constants.P_LocationCollection
+ "/" + constants.P_ProjectDocument)
   requestType := r.Method

   //Redirects the request to the appropriate function based on the type
of request
   switch requestType {
   case http.MethodGet:
      getProject(w, r)
   case http.MethodPost:
      createProject(w, r)
   case http.MethodPut:
      putRequest(w, r)
   case http.MethodDelete:
      deleteProject(w, r)
   }
}
```

Kode-utdrag 7 Delegeringsfunksjon ProjectRequestVidere blir informasjonen behandlet og sendt til passende funksjon.

Etter forespørselen har blitt sendt til riktig metode blir spørringene lest av og sendt videre til korresponderende funksjon, URL-en vil tas inn og spørringen vil bli lest. Dermed vil switch casen fange opp id og sende forespørselen til *getProjectWithID*. Forespørselen er nå sendt til funksjonen som er ansvarlig for å behandle forespørselen, kontakte databasen og sende en respons (se Kode-utdrag 8).

```
func getProject(w http.ResponseWriter, r *http.Request) {
   w.Header().Set("Content-Type", "application/json")
   w.Header().Set("Access-Control-Allow-Origin", "*")
   queries := mux.Vars(r)

   switch true {
   case queries[constants.P_idURL] == "" && queries[constants.P_nameURL]
== "" && queries[constants.P_State] == "":
      getProjectCollection(w, r) //If the query has keywords specific to
the state of the project it ends up here
      break
   case queries[constants.P_idURL] != "" || queries[constants.P_nameURL]
!= "" || queries[constants.P_State] != "":
      getProjectWithID(w, r) //If the query has keywords specific to the
state of the project it ends up here
      break
   default:
      tool.InvalidRequest(w, r) //If the query has neither the request is
invalid
      break
   }
}
```

Kode-utdrag 8 Funksjonen getProject, leser av spøøringene i forespørselen og sender informasjonen til passende funksjon

6.2.2.2 *Forespørselshåndtering GET:*

Endepunkter av denne typen inneholder kun spørringer og brukes for å hente ut informasjon fra databasen, med følgende prosessflyt: Eventuelle spørringer blir lest og sjekket (om en utvikler vil ha et dokument fra databasen av en spesiell type eller ID blir dette sendt inn som spørringer). Spørringene blir brukt til å definere dokumentbanen som leder til det ønskede dokumentet i databasen(se Kode-utdrag 9). Dokumentet blir hentet og det blir deretter sendt en respons ved hjelp av *json.NewEncoder(w).Encode().* Responsen blir sendt som JSON og brukes av front-end delene av løsningen.

```
func getIndividualScaffoldingPart(w http.ResponseWriter, scaffoldType
string, scaffoldId string) {
   w.Header().Set("Content-Type", "application/json")
   w.Header().Set("Access-Control-Allow-Origin", "*")

   objectPath :=
database.Client.Collection(constants.S_TrackingUnitCollection).Doc(consta
nts.S_ScaffoldingParts).Collection(scaffoldType).Doc(scaffoldId)

   part, err := database.GetDocumentData(objectPath)
   if err != nil {
      tool.HandleError(tool.DATABASEREADERROR, w)
      return
   }

   err = json.NewEncoder(w).Encode(part)
   if err != nil {
      tool.HandleError(tool.ENCODINGERROR, w)
      return
   }
}
```

Kode-utdrag 9 Eksempel på henting av informasjon fra databasen ved hjelp av spørringer

I de tilfeller hvor forespørselen etterspør flere dokumenter eller en kolleksjon av dokumenter blir en *iterator* brukt. Denne går igjennom alle dokumentene i en kolleksjon og returnerer en liste med objekter (se Kode-utdrag 10).

```
var scaffoldList []_struct.ScaffoldingType
partPath :=
database.Client.Collection(constants.S_TrackingUnitCollection).Doc(consta
nts.S_ScaffoldingParts).Collections(database.Ctx)
for {
   scaffoldingType, err := partPath.Next()
   if err == iterator.Done {
      break
   }
   if err != nil {
      tool.HandleError(tool.COLLECTIONITERATORERROR, w)
      return
   }
   document :=
database.Client.Collection(constants.S_TrackingUnitCollection).Doc(consta
nts.S_ScaffoldingParts).Collection(scaffoldingType.ID).Documents(database
.Ctx)
```

Kode-utdrag 10 Utdrag fra funksjon getAllScaffoldingParts: Itererer igjennom kolleksjonen med stillasdeler

Etter dokumentet er hentet ut blir dataen formatert ved hjelp av *json.Marshall* og *json.Unmarshall*. Responsen blir deretter returnert og inneholder den ønskede informasjonen (se Kode-utdrag 11).

```
partRef, err := document.Next()
      if err == iterator.Done {
         break
      }

      part, err := database.GetDocumentData(partRef.Ref)
      if err != nil {
         tool.HandleError(tool.DATABASEREADERROR, w)
         return
      }

      var scaffoldPart _struct.ScaffoldingType
      partByte, err := json.Marshal(part)
      if err != nil {
         tool.HandleError(tool.MARSHALLERROR, w)
         return
      }

      err = json.Unmarshal(partByte, &scaffoldPart)
      if err != nil {
         tool.HandleError(tool.UNMARSHALLERROR, w)
         return
      }
      scaffoldList = append(scaffoldList, scaffoldPart)
   }
}

err := json.NewEncoder(w).Encode(scaffoldList)
if err != nil {
   tool.HandleError(tool.ENCODINGERROR, w)
   return
}
```

Kode-utdrag 11 Del av funksjon getAllScaffoldingParts, funksjonen henter ut database dokumentet og formaterer dataen ved hjelp av en "struct" og json

6.2.2.3 *Forespørselshåndtering POST/PUT:*

POST forespørsler blir håndtert med samme metodikk som GET forespørsler, men krever derimot håndtering av forespørselens body da det er her informasjonen som skal oppdateres/legges inn befinner seg. Body-en til forespørselen blir lest av *io.ReadAll* før dataen blir formatert av *json.Unmarshall* (se Kode-utdrag 12).

```
requestBody, err := io.ReadAll(r.Body) //Reads body
if !checkStruct(requestBody) {
   tool.HandleError(tool.INVALIDBODY, w)
   return
}

var employee _struct.Employee                //Defines the appropriate
struct
err = json.Unmarshal(requestBody, &employee) //Unmarshall the requestBody
into the struct
if err != nil {
   tool.HandleError(tool.UNMARSHALLERROR, w)
   return
}
```

Kode-utdrag 12 Del av funksjon createProfiles: Body blir lest og formatert

Deretter blir dataen sendt og reformatert til datatypen som kreves av databasen før informasjonen blir lagt inn og en passende respons blir returnert (se Kode-utdrag 13).

```
err = json.Unmarshal(requestBody, &firebaseInput) //Formats the
requestBody
if err != nil {
   tool.HandleError(tool.UNMARSHALLERROR, w)
   return
}

err = database.AddDocument(documentPath, firebaseInput) //Adds the
profile to the database
if err != nil {
   tool.HandleError(tool.COULDNOTADDDOCUMENT, w)
   return
} else {
   tool.HandleError(tool.ADDED, w)
}
```

Kode-utdrag 13 Del av funksjonen createProfile, bodyen blir formatert og sendt til databasen

6.2.2.4 *Forespørselshåndtering DELETE:*

For sletting i databasen følger systemet en lignende metodikk som POST forespørsler, brukeren av API-et sender inn en body, denne inneholder en liste med id-ene brukeren ønsker å slette (se Kode-utdrag 14). Body-en blir som for POST forespørselen lest inn og formatert før listen med id-er blir sendt til databasen, som sletter dokumentene og sender en passelig respons (se Kode-utdrag 15).

```
[
    {
        "id": 430
    },
    {
        "id": 420
    },
    {
        "id": 4
    }
]
```

Kode-utdrag 14 Eksempel body, bruker ønsker å slette prosjekt 430,420 og 4

```
for i := range deleteList {
   objectPath :=
database.Client.Collection(constants.S_TrackingUnitCollection).Doc(consta
nts.S_ScaffoldingParts).Collection(deleteList[i].Type).Doc(deleteList[i].
Id)
   err := database.DeleteDocument(objectPath)
   if err != nil {
      tool.HandleError(tool.COULDNOTFINDDATA, w)
      return
   }
}

err = json.NewEncoder(w).Encode("All parts deleted successfully, number
of parts deleted:")
if err != nil {
   tool.HandleError(tool.ENCODINGERROR, w)
   return
}
```

Kode-utdrag 15 Utdrag fra funksjon deletePart, listen med id-er blir slettet i databasen

6.2.2.5 *Kommunikasjon med database:*

All database kommunikasjon går gjennom *database.go* filen. Filen inneholder funksjoner hentet fra *Googles firestore* dokumentasjonsside og lar API-et oppdatere, legge til, hente og slette data fra databasen. I tillegg har filen en initialiserings-funksjon som etablerer kommunikasjon mellom API-et og databasen (se Kode-utdrag 16). Databasen krever en autentiseringsnøkkel i form av en *.json* fil, denne ble ekskludert fra git-repository og manuelt lagt på testserveren.

```
//Code taken from
https://firebase.google.com/docs/firestore/quickstart#go
func DatabaseConnection() {
   file, err := filepath.Abs("database/stillas-16563-firebase-adminsdk-
wd82v-a9fe8919b7.json")
   if err != nil {
      log.Fatal(err)
   }
   // Creates instance of firebase
   Ctx = context.Background()
   sa := option.WithCredentialsFile(file) //Initializes database
   app, err := firebase.NewApp(Ctx, nil, sa)
   if err != nil {
      log.Println("error occured when initializing database" +
err.Error())
      _ = fmt.Errorf("error initializing app: %v", err)
   }

   Client, err = app.Firestore(Ctx) //Connects to the database
   if err != nil {
      log.Fatalln(err)
   }
}
```

Kode-utdrag 16 Funksjon DatabaseConnection, etablerer kommunkasjonslinje imellom API og database

API-et bruker hovedsakelig tre funksjoner fra databasen: *AddDocument*, *DeleteDocument* og *GetDocumentData*. Disse funksjonene tar i bruk grunnleggende database-operasjoner og tar kun imot datatypen **firestore.DocumentRef* (se Kode-utdrag 17).

```
func GetDocumentData(document *firestore.DocumentRef)
(map[string]interface{}, error) {
   dsnap, err := document.Get(Ctx)
   if err != nil {
      return nil, err
   }
   m := dsnap.Data()
   return m, nil
}
```

Kode-utdrag 17 Funksjon GetDocumentData hentet fra Googles dokumentasjon
https://firebase.google.com/docs/firestore/query-data/get-data

For database-transaksjoner som gikk over flere dokumenter ble det implementert en *Google Firestore* metode kalt *batched writes*. Her initialiserte gruppen et «batch» før transaksjonen, gjorde alle nødvendige database-transaksjoner før programmet til slutt «commit-er» programmet «batchen», endringen vil da kun skje om alle transaksjonene gikk igjennom (se Kode-utdrag 18)

```
batch := database.Client.Batch() //Defines the database operation

requestBody, err := io.ReadAll(r.Body) //Read body of the request
if err != nil {
   tool.HandleError(tool.READALLERROR, w)
   return
}

ok := checkDeleteBody(requestBody) //Checks format
if !ok {
   tool.HandleError(tool.INVALIDBODY, w)
   return
}

var deleteID _struct.ProfileDelete           //Defines the structure of
the request
err = json.Unmarshal(requestBody, &deleteID) //Unmarshall the body into
the defined struct
if err != nil {
   tool.HandleError(tool.UNMARSHALLERROR, w)
   return
}

for _, num := range deleteID { //Iterates through the profiles
   document, err := iterateProfiles(num.Id, "")
   if err != nil {
      tool.HandleError(tool.CouldNotDelete, w)
      return
   }

   batch.Delete(document[0]) //Deletes the profile
}
_, err = batch.Commit(database.Ctx) //Commits the change if all deletes
pass
if err != nil {
   tool.HandleError(tool.CouldNotDelete, w)
}
```

Kode-utdrag 18 Implementasjon av batched writes

6.2.2.6 *Kommunikasjon med sporingsenhet:*

Kommunikasjonen mellom sporingsenheten og API-et skjer i filen *trackingTools.go*. Denne klassen håndterer Gateway-ens datapakke. Gateway enheten sender en POST forespørsel til et definert endepunkt i API-et med en body som inneholder datapakken til Gateway-en (Kode-utdrag 19).

```
$GPRP,C745C54A8C62,34AB954B54E4,-18,02010612FF2C0883BC2F0100AAAAFFFF000030030000
$GPRP,D8E01B848E6E,34AB954B54E4,-19,02010612FF2C0883BC2D0100AAAAFFFF000030030000
```

Kode-utdrag 19 INGICS IGS03M payload

Deretter blir denne Datapakken oversatt ved hjelp av et bibliotek laget av produsenten av enheten (*INGICS Technologies*). Denne Datapakken inneholder en liste med hvilke sporingsbrikker som er registrert, hvor mye batteri de har, samt hvilken Gateway som plukket de opp. Det blir deretter laget to lister, en med tag informasjon og Gateway informasjon. (se Kode-utdrag 20) Deretter blir disse listene sendt til *getTagLists* som returnerer to nye lister, en inneholder en oppdatert tag-liste som kun inneholder de seks siste sifrene i MAC-adressen på taggen. Det blir også laget en batteriliste (se Kode-utdrag 21).

```
payload, _ := ioutil.ReadAll(r.Body)
convertedPayload := string(payload)
payloadList := strings.Split(convertedPayload, "\n")

for i := 0; i < len(payloadList)-1; i++ {
   if m := igs.Parse(payloadList[i]); m != nil {
      gatewayList = append(gatewayList, m)
      if bytes, err := hex.DecodeString(m.Payload()); err == nil {
         p := ibs.Parse(bytes)
         beaconList = append(beaconList, p)
      }

   } else {
      InfoLogger.Printf("Invalid input message: %v", os.Args[1])
   }
}
```

Kode-utdrag 20 Lesing av gateway payload ved hjelp av biblioteker fra INGICS Technologies

```
var tagIDList []string
batteryList := make(map[string]float32)

for i := 0; i < len(tagList); i++ {
   tagInfo := gatewayList[i].Beacon()
   runePayload := []rune(tagInfo)
   tagID := string(runePayload[6:12])
   battery, _ := tagList[i].BatteryVoltage()

   tagIDList = append(tagIDList, tagID)
   batteryList[tagID] = battery
}
return tagIDList, batteryList
```

Kode-utdrag 21 Funksjon getTagLists henter ut en tagliste og en batteriliste

Deretter begynner hovedprosessen i klassen. BeaconID, id-listen og batterilisten blir sendt til *updateAmountProjects*, denne funksjonens formål er å finne Gateway-ens tilhørende prosjekt samt lage en liste med mengden registrerte stillasdeler gruppert på type. Prosessen begynner med henting av det utdaterte byggeprosjektet fra databasen, dette hentes ut ved hjelp av Gateway dokumentet.

```
for j := range scaffoldingArray {
   scaffoldingType = scaffoldingArray[j].Type
   documentPath :=
database.Client.Collection(constants.S_TrackingUnitCollection).Doc(consta
nts.S_ScaffoldingParts).Collection(scaffoldingType).Doc(idList[i])
   _, err := documentPath.Update(database.Ctx, []firestore.Update{
      {
         Path:  "project",
         Value: projectName,
      }, {
         Path:  "batteryLevel",
         Value: batteryList[idList[i]],
      },
   })
   if err != nil {
      ErrorLogger.Printf("Document with id: %v is not in scaffoldingType
collection %n", idList[i], scaffoldingType)
   } else if err == nil {
      DatabaseLogger.Printf("Succsessfully updated scaffolding part: %v",
idList[i])
      resultList[scaffoldingType] = resultList[scaffoldingType] + 1
   }
}
```

Kode-utdrag 22 Udrag fra getTagTypes, for loop iterer igjennom stillasdelene, oppdaterer assosiert prosjekt og batterinivå samt legger stillastype i et map med tag-id

Deretter blir listen med registrerte stillasdeler sendt til en funksjon som lager en liste med tag-id-en og hvilken stillastype taggen er festet til (Kode-utdrag 22). Deretter blir denne listen med tag-id-er og assosiert stillastype lagt inn i byggeprosjektet og oppdatert i databasen (Kode-utdrag 23).

```
var update bool
for i := range updatedProject.ScaffoldingArray {
   doc, err :=
database.Client.Collection(constants.P_LocationCollection).Doc(constants.
P_ProjectDocument).Collection(updatedProject.State).Doc(strconv.Itoa(upda
tedProject.ProjectID)).Collection(constants.P_StillasType).Doc(updatedPro
ject.ScaffoldingArray[i].Type).Set(database.Ctx,
      newMap["scaffolding"].([]interface{})[i],
      firestore.MergeAll)

   if err != nil {
      tool.HandleError(tool.DATABASEADDERROR, w)
      ErrorLogger.Printf("Database ADD error on object %v", doc)
      update = false
   } else {
      update = true
   }
}
if update == true {
   DatabaseLogger.Printf("Successfully updated project with gateway id
%v\n", beaconID)
} else {
   DatabaseLogger.Printf("Unsuccessfully updated project with gateway id
%v\n", beaconID)
}
```

Kode-utdrag 23 Utdrag fra updateAmountProject, byggeprosjektet med oppdatert mengde stillasdeler blir lagt inn i databasen

6.2.3 Resultat

API-et håndterer både byggeprosjekter, stillasdeler, profiler, Gateway-er og sporingsdata. Gruppen ser derimot at API-et har behov for optimalisering og at skalering av løsningen vil øke responstid betraktelig (se kap. 9.2). Dette skyldes forespørsel-håndtering og mangel på lokal lagring. Gruppen har ikke implementert all ønskelig funksjonalitet. Vi ønsket under definisjon av Use-caset å implementere en logg for hver stillasdel slik at en bruker hadde mulighet til å sjekke hvor en stillasdel hadde vært tidligere.

Gruppen mener at API-et burde optimaliseres, samt at informasjonen som kommer inn ikke valideres godt nok på enkelte endepunkter, særlig *gateway.go* og *scaffolding.go* mangler noe input validering i forhold til *projects.go*. Gruppen ønsker også mer loggefunksjonalitet, dette er påbegynt, men gjøres ikke for alle klassene i API-et.

API-et er også tilgjengelig for alle som har tilgang til API-dokumentasjonen, gruppen ønsket å implementere en API-nøkkel som kun gir utviklere tilgang til API-et, men dette ble ikke implementert på grunn av manglende tid.

6.2.4 Diskusjon

Kapittelet vil diskutere diverse valg gruppen tok under utvikling av API-et. Dette innebærer blant annet valg av programmeringsspråk og biblioteker, samt optimalisering av API-et, dets filstruktur og hvordan gruppen har implementert forespørselshåndteringen i systemet.

6.2.4.1 *Programmeringsspråk*

Gruppen valgte tidlig i prosessen å ta i bruk programmeringsspråket Golang, dette var i stor grad på grunn av tidligere kjennskap til språket igjennom faget *Cloud Technologies*[14] hvor alle gruppemedlemmer hadde jobbet med Golang og API-utvikling.

Gruppen vurderte derimot andre programmeringsspråk, blant annet Python, på grunn av den store brukerbasen språket har i tillegg til mangfoldet av tredjeparts-biblioteker språket tilbyr. PHP ble også vurdert tidlig i prosessen, hovedsakelig fordi en relasjonsdatabase ble vurdert, samt tidligere erfaringer ved språket. Etter at valget falt på en dokumentorientert database, ble det enighet om at Golang egnet seg bedre for denne typen database da integrasjon av *Google firestore* i Golang ble dekket i *Cloud Technologies*. Java og C++ ble også vurdert da gruppen har erfaring i begge språkene. I tillegg har både Java og C++ et rikt tredjeparts bibliotek og en aktiv brukerbase og tilbyr derfor langt mer funksjonalitet enn Golang.

Gruppen endte derimot opp med å velge Golang siden vi hadde et ønske om å dykke dypere ned i språket. Etter planlegging av funksjonalitet så gruppen også at API-et hadde lite bruk for eventuelle tredjepartsbiblioteker.

[14] Se mer om Cloud Technologies https://www.ntnu.edu/studies/courses/PROG2005/#tab=omEmnet

6.2.4.2 *Filstruktur*

API-et ble igjennom utvikling jobbet på av to gruppemedlemmer. På grunn av dette var det viktig med en robust og definert filstruktur. Valget falt derfor på å splitte opp prosjektet på følgende måte se Kode-utdrag 24.

```
api/
├── Dockerfile
├── apiTools
│   ├── basicTools.go
│   └── errorHandling.go
├── cmd
│   ├── cmd
│   └── main.go
├── constants
│   └── constants.go
├── database
│   ├── databaseSetup.go
│   └── stillas-16563-firebase-adminsdk-wd82v-a9fe8919b7.json
├── docker-compose.yml
├── endpoints
│   ├── APIHandler.go
│   ├── gateway.go
│   ├── homePage.go
│   ├── profiles.go
│   ├── projectTools.go
│   ├── projects.go
│   ├── scaffolding.go
│   └── trackingTools.go
├── go.mod
├── go.sum
├── logs.txt
├── mqtt
│   ├── mosquitto.conf
│   ├── mosquitto.passwd
│   └── mqttBroker.go
├── struct
│   ├── GatewayStruct.go
│   ├── ProfileStruct.go
│   ├── ProjectStruct.go
│   └── ScaffoldingStruct.go
└── tests
    ├── diagnostics_test.go
    ├── gateway_test.go
    ├── profiles_test.go
    ├── projects_test.go
    ├── scaffolding_test.go
    ├── stillastestdatabase-firebase-adminsdk-tvp5e-1eb0fe0a3b.json
    └── testDatabase.go
```

Kode-utdrag 24 Filstruktur API

Hoved-funksjonalitet ligger i filen *endpoints.go*. Denne filen inneholder en fil per endepunkt gruppen har utviklet. Gruppen valgte å splitte endepunktene på denne måten slik at gruppemedlemmer kunne jobbe med API-et samtidig uten at det oppsto konflikt i kodebasen. Gruppen valgte også tidlig i prosessen å opprette en *constants* mappe. Denne inneholder filer med konstante verdier som blir brukt igjennom API-et. Vi så dette som nødvendig da API-et ofte definerer nøkkelord ved kommunikasjon med databasen og aksessering av felter i objekter.

Ved å ta i bruk denne *constants* mappen har gruppen én plass hvor alle konstanter er definert. Dette gjør det lettere å oppdatere eventuelle konstanter om systemet må endres (databasetype endres/restruktureres, felter oppdateres i structer etc). Gruppen må kun oppdatere konstanten endringen gjelder, istedenfor å gå igjennom hele kodebasen og oppdatere hard-kodede verdier i enkeltfiler.

```
U_idURL    = "id"
U_nameURL  = "name"
U_User     = "user"

//Gateway url
G_idURL              = "id"
G_gidURL             = "gatewayID"
G_GatewayURL         = "gateway"
G_Gateway            = "Gateway"
G_GatewayCollection  = "Gateways"
G_ProjectName        = "projectName"
G_ProjectID          = "projectID"
G_GatewayID          = "gatewayID”
```

Kode-utdrag 25 Utdrag fra constants dokument med definerte konstanter

Gruppen valgte også å gruppere database, verktøy, «structer» og tester i egne mapper slik at kodebasen skulle være lett å navigere samt enklere å vedlikeholde om flere jobbet på API-et samtidig.

6.2.4.3 *Forespørselshåndtering*

Forespørselshåndtering er svært sentralt i API-utvikling, all informasjon og funksjonalitet omhandler forespørsler, gruppen hadde diverse valg da vi skulle bestemme hvordan dette skulle implementeres. Tidlig under utvikling ble standard http biblioteket i Golang brukt[15], både for definering av endepunkter, ruter initialisering og håndtering av forespørsel-body og spørringer.

Gjennom prosessen ble det holdt et fokus på at applikasjonen skulle være fremtidsrettet og ta i bruk biblioteker som ikke risikerer å bli utdaterte. Valgte falt derfor først på å ta i bruk standardbiblioteket. Gruppen måtte derimot gå bort ifra dette under utvikling av front-end løsningene da bruk av *net.http* biblioteket ikke tillot innsending av spørringer i endepunktene for web-rammeverket. Dette i tillegg til at API-testing er vesentlig lettere med valgt alternativ.

[15] Mer om net.http i Golang: https://pkg.go.dev/net/http

Net.http biblioteket ble erstattet av *Gorilla mux*[16]. Gruppen var fra *Cloud Technologies*[17] kjent med biblioteket, dette i kombinasjon med at biblioteket er både stabilt, samt at det er *de facto* standard for API-utvikling i Golang. Gruppen innser at dette biblioteket kan bli utdatert i fremtiden da siste oppdatering ble gjort den 22 august 2020, samt at biblioteket ser etter en ny vedlikeholder. Da gruppen kun tar i bruk grunnleggende funksjonalitet, ser vi ikke for oss at dette vil føre til problemer for API-et.

For forespørselshåndtering har gruppen valgt å følge en lik struktur der det lar seg gjøre: API-et tar imot og behandler forespørsler ved hjelp av tre biblioteker, mest sentralt er bruken av *encoding/json* bilbioteket[18]. Biblioteket blir brukt under alle former for forespørselshåndtering. Biblioteket er essensielt da all kommunikasjon mellom endepunkt, database og API foregår i data formatet JSON[19]. Biblioteket lar programmet oversette, behandle og videresende data på et universelt format.

Prosessering av data fra endepunktene innebærer oversetting fra JSON input data til egendefinerte *struct*-er. For denne prosessen valgte vi å ta i bruk egne *struct*-klasser som inneholder data formater for de forskjellige formene for input og output data API-et prosesserer (se Kode-utdrag 26).

```
type NewProject struct {
   ProjectID   int      `json:"projectID"`
   ProjectName string   `json:"projectName"`
   Size        int      `json:"size"`
   State       string   `json:"state"`
   Latitude    float64  `json:"latitude"`
   Longitude   float64  `json:"longitude"`
   Period      `json:"period"`
   Address     `json:"address"`
   Customer    `json:"customer"`
   Geofence    `json:"geofence"`
}

type GetProject struct {
   NewProject
   ScaffoldingArray `json:"scaffolding"`
}
```

Kode-utdrag 26 Struct objekter tatt fra klassen ProjectStruct

Ettersom mange av forespørslene i API-et er strukturert på samme måte ledet dette til en større mengde duplisert kode enn det gruppen ønsket. Eksempelvis er henting av et dokument i databasen, formatering til passende *struct* nærmest identisk for opptil flere funksjoner (se Kode-utdrag 27). Gruppen innser at kode som dette burde blitt abstrahert og gjenbrukt for å minimere duplisert kode og redusere muligheter for feil, samt gjøre koden mer lesbar samt enklere og vedlikeholde.

`data, err := database.GetDocumentData(ref)`	`part, err := database.GetDocumentData(partRef.Ref`

[16] Mer om Gorilla mux på github repoet: https://github.com/gorilla/mux
[17] Mer om Cloud Technologies: https://www.ntnu.edu/studies/courses/PROG2005/#tab=omEmnet
[18] Les mer på golangs bilbioteksnettside: https://pkg.go.dev/encoding/json
[19] Les mer om JSON her: https://www.json.org/json-en.htmlhttps://www.json.org/json-en.html

```
if err != nil {

tool.HandleError(tool.NODOCUMENTWITHID, w)
    return
}

jsonStr, err := json.Marshal(data)
if err != nil {

tool.HandleError(tool.MARSHALLERROR, w)
    return
}

var projectNew _struct.NewProject
err = json.Unmarshal(jsonStr, &projectNew)
if err != nil {
tool.HandleError(tool.UNMARSHALLERROR, w)
    return
}
```

```
)
if err != nil {

tool.HandleError(tool.DATABASEREADERROR, w)
    return
}

var scaffoldPart _struct.ScaffoldingType
partByte, err := json.Marshal(part)
if err != nil {

tool.HandleError(tool.MARSHALLERROR, w)
    return
}

err = json.Unmarshal(partByte, &scaffoldPart)
if err != nil {
tool.HandleError(tool.UNMARSHALLERROR, w)
    return
}
```

Kode-utdrag 27 Duplisering av kode i klassene scaffolding.go go projects.go

6.2.4.4 *Optimalisering*

Gitt at hovedfunksjonen til API-et er å kommunisere med en database ble det lagt mye vekt på å kun hente ut nødvendig informasjon fra databasen der det trengs, slik at de sentrale funksjonene som brukes ofte, bruker så få «reads» og «writes» som mulig (se kap. 6.1). Dette er spesielt viktig for funksjoner som henter ut dokumenter eller kolleksjoner basert på en parameter (Kode-utdrag 28). Disse funksjonene ble skrevet slik at de kun itererer igjennom nødvendige dokumenter og kolleksjoner istedenfor å hente ut alle dokumentene/kolleksjonene i databasen.

Dette reduserer mengden «reads» og «writes» til en viss grad, men gruppen innser at implementering av et caching system i API-et hadde redusert mengden «reads», da spesielt for stillasdeler siden skalering av systemet vil kunne kreve lesing av opp til 2 000 000 stillasdeler. Som diskutert i kap. 6.1 er stillasdeler gruppert på type, dette tilsier at når en Gateway sender inn informasjon om 100 stillasdeler, må API-et iterere gjennom alle typene stillasdeler som ligger inne på et prosjekt, for å finne de enkelte stillasdelene med en spesifikk tag-id. Implementering av en stillasdel-cache med tidligere registrerte stillasdeler vil potensielt spare databasen for ekstremt mange «reads».

Det ble også undersøkt muligheter for å ta i bruk en *MQTT Broker* for å gjøre kommunikasjon med sporingsenhetene mer skalerbar. Gateway-ene tar i bruk HTTP POST for å sende informasjon til API-et. Dette fungerer godt for testing å småskala sporing, men ved skalering av systemet vil ytelse i stor grad være avhengig av serveren API-et kjører på. Implementasjon av en *MQTT* protokoll hadde avlastet API-et da kommunikasjon mellom sporingsenhetene og API-et hadde gått gjennom en *MQTT Broker* istedenfor å gå direkte i API-et. Dette hadde også gjort løsningen lettere å skalere fordi all data hadde blitt tatt imot, filtrert og videresendt via

Brokeren. Antall sporingsenheter hadde da hatt mindre innvirkning på/redusert trafikk i API-et.

Det ble satt opp og testet en *MQTT Broker* i kodebasen ved hjelp av «Open-source» brokeren *MOSQUITTO*, men dette ble ikke implementert da gruppen ikke hadde tid, samt at Gateway-ene kunne sende informasjon via HTTP POST.

```
for i := range idList {
   for j := range scaffoldingArray {
      scaffoldingType = scaffoldingArray[j].Type
      documentPath :=
database.Client.Collection(constants.S_TrackingUnitCollection).Doc(consta
nts.S_ScaffoldingParts).Collection(scaffoldingType).Doc(idList[i])
      _, err := documentPath.Update(database.Ctx, []firestore.Update{
         {
            Path:  "project",
            Value: projectName,
         }, {
            Path:  "batteryLevel",
            Value: batteryList[idList[i]],
         },
      })
      if err != nil {
         ErrorLogger.Printf("Document with id: %v is not in
scaffoldingType collection %n", idList[i], scaffoldingType)
      } else if err == nil {
         DatabaseLogger.Printf("Succsessfully updated scaffolding part:
%v", idList[i])
         resultList[scaffoldingType] = resultList[scaffoldingType] + 1
      }
   }
}
```

Kode-utdrag 28 Funksjonen itererer igjennom alle typen stillasdeler på et byggeprosjekt og finner tag-id-er

7. Front-End

Kapittelet er strukturert på samme vis som kapittel 6, begge front-end løsningene gruppen har laget vil bli diskutert her. Gitt at både web- og mobilapplikasjonen har blitt utviklet med hensyn på prinsipper tatt fra faget *Introduksjon til Brukersentrert design*[20] vil gruppen introdusere brukte prinsipper i starten av kapittelet. Deretter vil design, implementasjon og diskusjon av webløsning forekomme før gruppen avslutningsvis repeterer prosessen for mobilapplikasjonen. Lengden på kapittelet skyldes innhold av figurer og kodesnutter, noe som vil gjøre det lettere å sette seg inn i.

7.1 Brukergrensesnitt

Innenfor web- og interaksjonsdesign finnes det mange designprinsipper. *Don Norman* er en sentral skikkelse innenfor menneske-maskin interaksjon (HCI) og brukersentrert design. Prinsippene hans er blant de mest anerkjente i bransjen og er sentrale i det meste av utvikling i dag. Tanken bak *Don Normans* prinsipp om universelt design er at alle gjenstander, grensesnitt og datamaskiner skal være lett å bruke, funksjonell og ikke-diskriminerende. De seks viktigste prinsippene er *Visibility, Feedback, Affordance, Mapping, Constraints* og *Consistency* (48).

7.1.1 Interaksjonsdesign (IxD)

Som navnet impliserer, omhandler interaksjonsdesign designet rundt hvordan et system eller en gjenstand skal respondere og fungere når et menneske samhandler med det. Tankegangen er at systemet skal tilpasses brukeren, ikke motsatt. Nedenfor er en elaborering over ulike design prinsipper.

7.1.1.1 *Discoverability*

Discoverability, er prinsippet som tar for seg graden av hvor lett det er for en bruker av et nytt system å finne fram til all funksjonalitet og informasjon i systemet, samt navigere seg gjennom dette systemet.

7.1.1.2 *Affordance / Signifiers*

Affordance definerer hvilke handlinger som kan gjennomføres. Et eksempel på dette er at knapper som regel kan trykkes på eller at en smarttelefon tilbyr interaksjon som sveiping og tapping.

Signifiers spesifiserer og forteller til brukeren hvordan «affordanc-en» kan benyttes. Har man en knapp med teksten «Bruk» impliserer dette at man er i ferd med å velge noe til å bli benyttet om man trykker på denne knappen.

7.1.1.3 *Feedback*

Feedback (norsk: respons), er tilbakemeldingen fra systemet på en handling brukeren gjør. Eksempelvis om en bruker skal kunne velge flere knapper i en flervalgs seksjon bør det markeres til brukeren hvilke knapper brukeren har valgt (se Figur 21).

[20] Introduksjon til brukersentrert design https://www.ntnu.no/studier/emner/IDG1362#tab=omEmnet

Figur 21 Eksempel på en tilbakemelding. Bilde hentet fra https://www.questionpro.com/features/multiple-choice-multiple-answer.html

7.1.1.4 *Mappings*

Mappings er relasjonen og plasseringen mellom kontrollene og effekten de har på et system. Dette vil for eksempel være at elementer for registrering av en ny bruker kommer sekvensielt nedover med en «Registrer» knapp nederst, fremfor øverst. På denne måten er det enklere for brukeren å forstå at all dataen brukeren har fylt inn ned til «Registrer-knappen» kommer til å bli registrert.

7.1.1.5 *Constraints*

Constraints (norsk: begrensninger) omhandler å begrense brukerens interaksjonsmuligheter. Dette gjøres av og til for å begrense muligheten for at brukeren samhandler «feil» med systemet, men som regel gjøres det for å veilede brukeren gjennom flere etterfølgende handlinger eller operasjoner som må gjennomføres i en bestemt rekkefølge.

7.1.1.6 *Consistency*

Consistency (norsk: konsistens) handler om å sette standarder for brukerhandlinger slik at systemet gir lik tilbakemelding på like handlinger. Eksempelvis skal responsen være lik hver gang brukeren trykker på en bestemt knapp eller legger til nye brukere.

7.1.1.7 *Patterns and learnability*

Patterns and learnability omhandler mønstre i handlinger og hvor enkelt det er å lære seg å bruke systemet. Budskapet her er at ved konsekvent systemet design skal hver interaksjon med eksempelvis en kalender se lik ut, i tillegg til at tiden brukeren må investere i å lære seg systemet er så liten som mulig.

7.1.1.8 *Visual hierarchy*

Et godt strukturert visual hierarchy (norsk: visuelt hierarki) veileder blikket til brukeren til de sentrale og viktige delene av en side. Dette kan skje gjennom variasjon i farge og kontrast, størrelsesforhold og gruppering (49). Ellers vil også det å følge et typisk oppsett av en applikasjon og nettside være nyttig for å oppnå et bedre visual hierarchy, ettersom øynene av vane vet hvor de skal fokusere for ulike funksjonaliteter.

7.1.1.9 *Avoid error*

Avoid error (norsk: unngå error) går ut på å være tilgivende og tillate brukeren å rette opp i en feil ved å reversere handlinger. Informative handlingsalternativer, sanntid advarsler som at brukeren må fylle ut et felt for å få gå videre, forhåndsvisning av resultat, deaktivering av knapper til en handling er utført etc., er med på å unngå error.

7.1.2 Brukergrensesnitt og visuelt design

I denne seksjonen vil ulike temaer rundt hvordan brukergrensesnitt bør struktureres og hensyn som bør tas, introduseres.

7.1.2.1 *Språk*

I en mobilapplikasjon eller webside er det viktig å ordlegge seg basert på målgruppen og brukerne. Dette innebærer alt fra ordlegging til tone når man kommuniserer tilbake til brukeren fra systemet. Kommunikasjonen bør være kort og konkret, tydelig for bruken, samt ærlig.

7.1.2.2 *Former*

Etter hvert som mobilapplikasjoner og webløsninger har blitt populære, har bruken av figurer og former blitt stadig mer populært både for knapper og for å separere informasjon vist til brukeren. Det er derfor viktig å være varsom med hvilke former og figurer man velger å bruke til hvilke oppgaver.

7.1.2.3 *Farge*

Fargevalg er i mange tilfeller avgjørende for front-end-løsningens suksess og brukernes tilfredsstillelse. Når det gjelder fargevalg i en applikasjon er det følgende anbefalte retningslinjer.

Omtrent 60% av designet skal bestå av nøytrale farger. Omtrent 30% sekundær merkevare farge og de siste 10% kan være den primære merkevare fargen. Avslutningsvis anbefales det også å ta hensyn til farge teori («Color Theory») som omhandler brukerens følelser koblet opp mot fargen de ser og hvordan farger samhandler med hverandre.

7.1.2.4 *Imagery*

Imagery innenfor grafisk design er å gi en visuell representasjon av en gjenstand eller en handling gjennom bilder, illustrasjoner eller grafiske representasjoner. Dette benyttes ofte for å gi brukeren en sterkere tilknytning til det de ser og for å tydeliggjøre eventuelle usikkerheter.

7.1.2.5 *Typografi og Fonter*

Typografier og fonter er sentrale i applikasjonsdesign for å fremme informasjon og få brukerens oppmerksomhet. Eksempelvis vil utheving av viktig informasjon i form av titler eller fet tekst være et virkemiddel for å veilede brukerens oppmerksomhet dit.

7.1.2.6 *Ikoner*

Ikoner er nyttige hjelpemidler for å formidle en funksjonalitets funksjoner eller et handlingsformål i de tilfeller hvor de benyttes riktig. Disse bør benyttes sammen med en

tilhørende tekst, særlig om det er symboler som ikke er universelle i mange bruksområder. Ikoner bør være deskriptive for bruken, og ikke bare benyttet som et tilfeldig symbol.

7.1.3 Standarder og retningslinjer

I dette underkapittelet introduseres stadarder og retningslinjer benyttet i de to frond-end løsningene.

7.1.3.1 Web Content Accessibility Guidelines (WCAG) 2.1

WCAG 2.1 dekker store deler av anbefalingene for å gjøre web innhold mer tilgjengelig for brukere både med og uten funksjonsnedsettelser (50). Disse retningslinjene er til stede for å veilede utviklere til hvordan man kan og skal tilrettelegge en applikasjon for brukere med blant annet nedsatt funksjonsevne inklusive synsnedsettelser, motoriske lammelser, hørselsnedsettelse og kognitive begrensninger. Å følge disse retningslinjene vil gi en økt brukertilfredshet.

De mest kjente tilgjengelighets barrierene for brukere er:

- Syn (e.g., fargeblindhet)
- Motorikk (e.g., Parkinson)
- Hørsel (Hørsels nedsettelse)
- Anfall (Fotosensitiv epilepsi)
- Kognitivt (e.g., dysleksi)

(51)

7.1.3.2 Human Interface Guidelines

For å imøtekomme høye forventninger for både kvalitet og funksjonalitet i en iOS applikasjon benyttes *Apple Developers Human Interface Guidelines*. Klarhet, respekt og dybde er tre sentrale begreper benyttet i denne sammenhengen, i tillegg til design prinsippene estetisk integritet, konsistens, tilbakemelding, brukerkontroll, metaforer og direkte manipulasjon (52).

7.2 Web-Applikasjon

Dette kapitelet tar for seg alle fasene innenfor utviklingen av webapplikasjonen. Her skal vi blant annet gå gjennom designfasen av brukergrensesnittet, implementasjonen og diskutere valgene som har blitt tatt, og hvilke utfordringer som ble møtt under utviklingen.

7.2.1 Krav

I utgangspunktet spurte oppgaven kun etter en mobilapplikasjon. Gruppen mente at en webapplikasjon ville ha vært fordelaktig for å kunne administrere databasen, ettersom det kan virke mer oversiktlig. MOSCOW konseptet ble brukt for å definere hvilke funksjonaliteter som virket realistisk for gruppen å gjennomføre på gitt tidsrom. Funksjonaliteten ble hentet fra kap. 3.1

MOSCOW-modellen:

Must have	- Liste byggeprosjekter - Overføre stillasdeler mellom prosjekter - Legge til stillasdeler - Legge til prosjekter - Registrere ny bruker - Autorisering av brukere
Should have	- Ha brukere - Vise prosjekter i kart - Oppdatere informasjon om prosjekt
Could have	- Vise historie/log for mengden stillas av hver del på et prosjekt
Won't have	- Vise stillasdeler i kart.

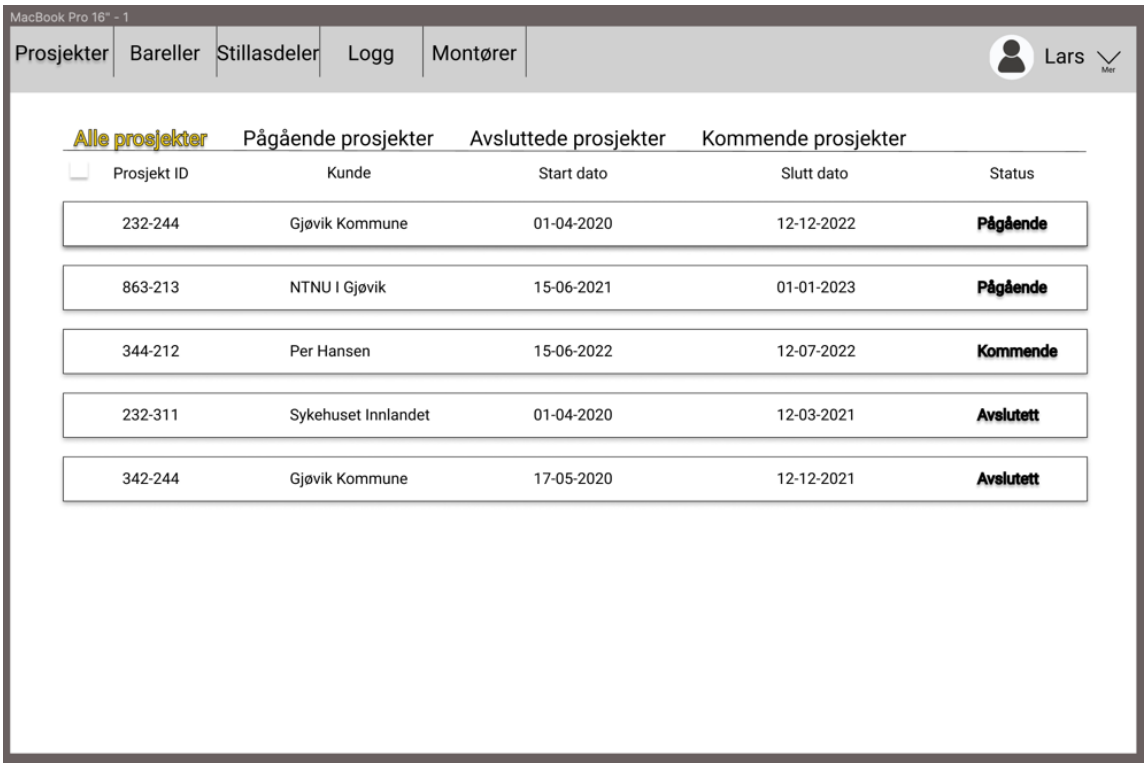
Figur 22 MOSCOW for webapplikasjonen

7.2.2 Design

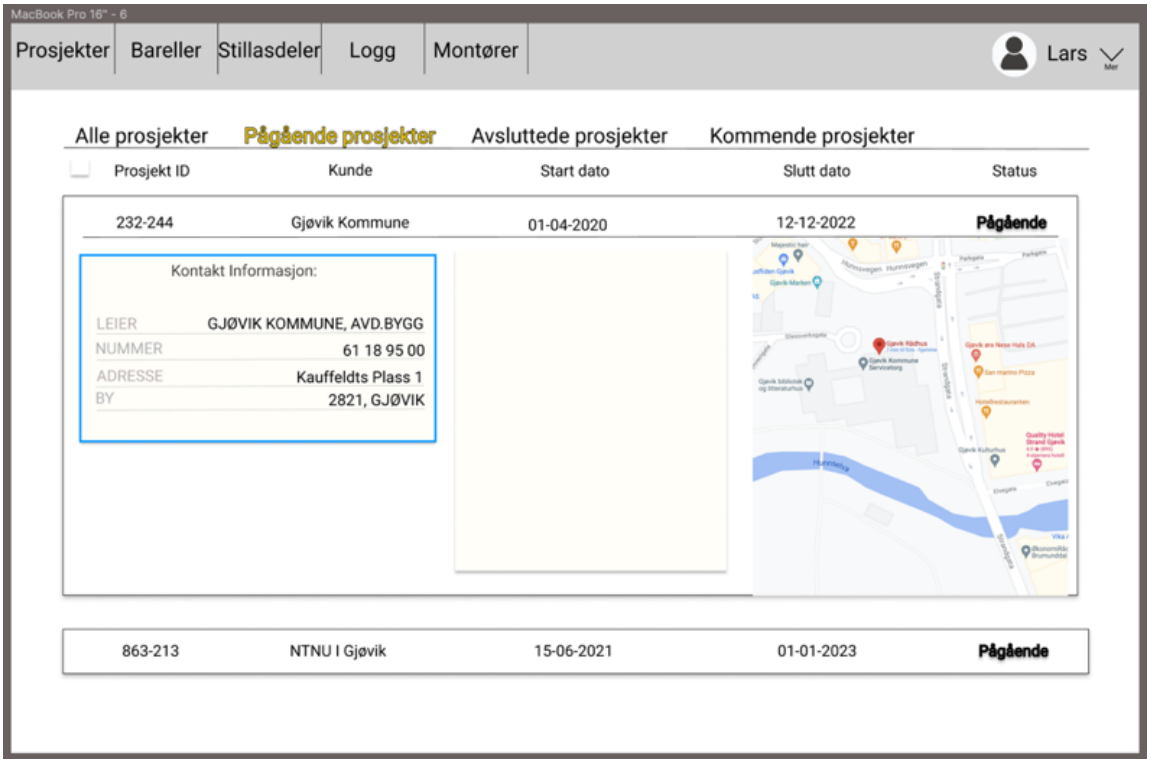
Design er et sentralt tema i front-end løsningen. Dette kapittelet skal ta for seg de ulike designene som har blitt vurdert i løpet av designfasen. Deretter vil filstrukturen i kildekoden drøftes.

7.2.2.1 Brukergrensesnitt

I oppstartsfasen til utviklingen av web-applikasjonen, ble det laget utkast av hvordan systemet kunne designes for å få frem den viktigste informasjonen for brukeren. Utkastene ble laget det nettbaserte prototypeverktøyet Figma (se Appendix K). Under førsteutkastet så gruppen for seg et design der prosjektene blir listet opp, som på bildene under.



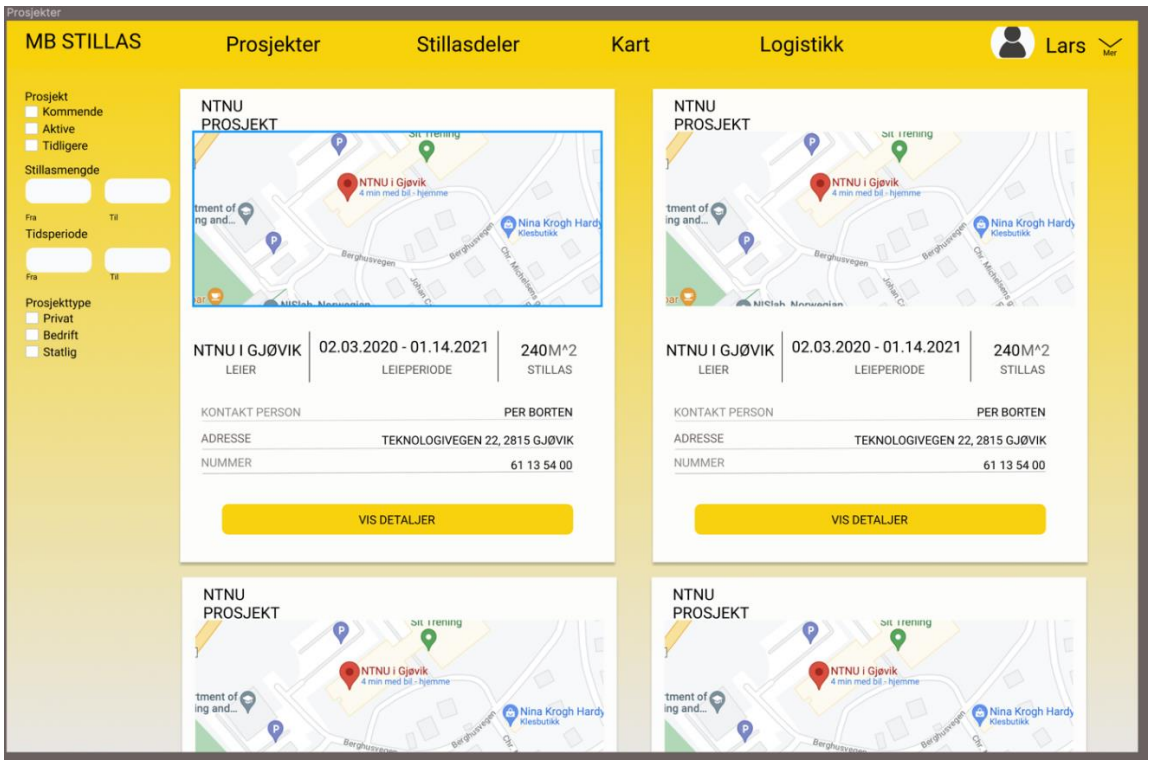
Figur 23 Første utkast av hovedsiden



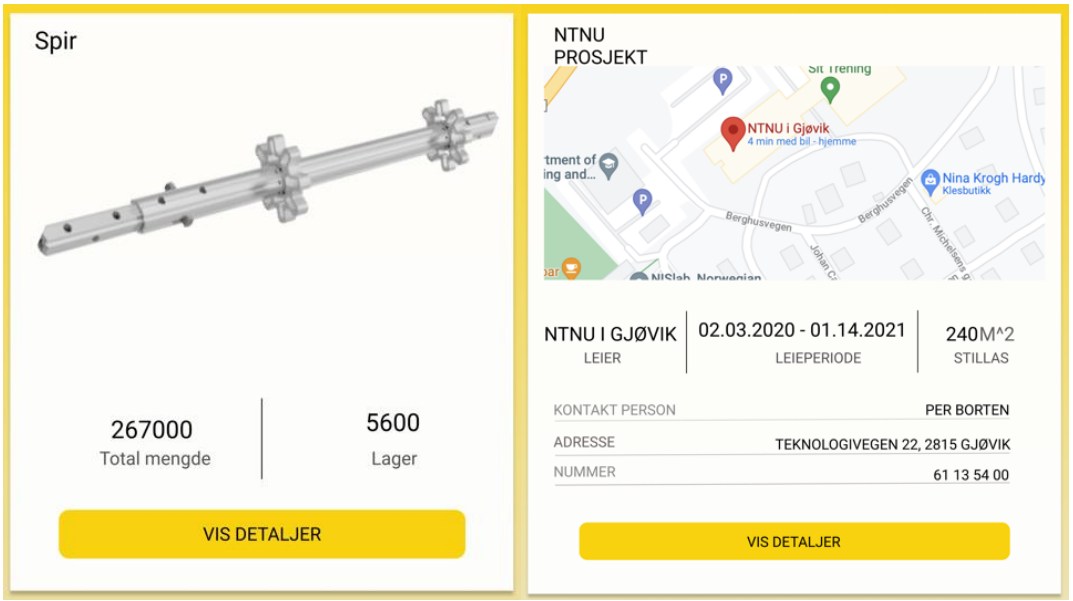
Figur 24 Første utkast av prosjekt informasjon siden

Under vurdering av førsteutkastet, mente gruppen at dette ga lite informasjon til brukeren, noe som ville resultere i at brukeren måtte trykke seg gjennom mange prosjekter for å finne frem til ønsket prosjekt, samt lite plass til å vise ønskede detaljer om hvert prosjekt.

Gruppen startet på andreutkastet med et ønske om å vise nok informasjon til brukeren, slik at det var lett å finne fram til riktig prosjekt. Ved introduksjon av informasjonskort og en filtreringsmekanisme, ble dette ønsket realisert. Prosjektene kan filtreres på diverse faktorer, som status, navn, dato og størrelse. Brukeren kan lett navigere seg frem til ønsket prosjekt, basert på diverse betingelsene (se Figur 25).



Figur 25 Utkast for prosjekt siden, med informasjonskort og filtreringsmekanisme



Figur 26 Informasjonskort for stilladel og prosjekt

Webapplikasjonen tok inspirasjon fra nettsider som *Apple* og *Tesla*[21], for å gjøre den oversiktlig og enkel å bruke. *Don Normans* prinsipp om universelt design, er anvendt gjennom design perioden av webapplikasjonen. Gitt at systemet skulle utvikles for ansatte i bygge bransjen, ønsket gruppen å holde designet minimalistisk, intuitiv og responsivt.

Brukeren navigerer seg gjennom hovedsidene av applikasjonen med navigasjonsbaren på toppen av nettsiden (se Figur 27).

Figur 27 Navigasjonsbaren til webapplikasjonen

For bilder av ferdig design for webløsningen, se Appendix M.

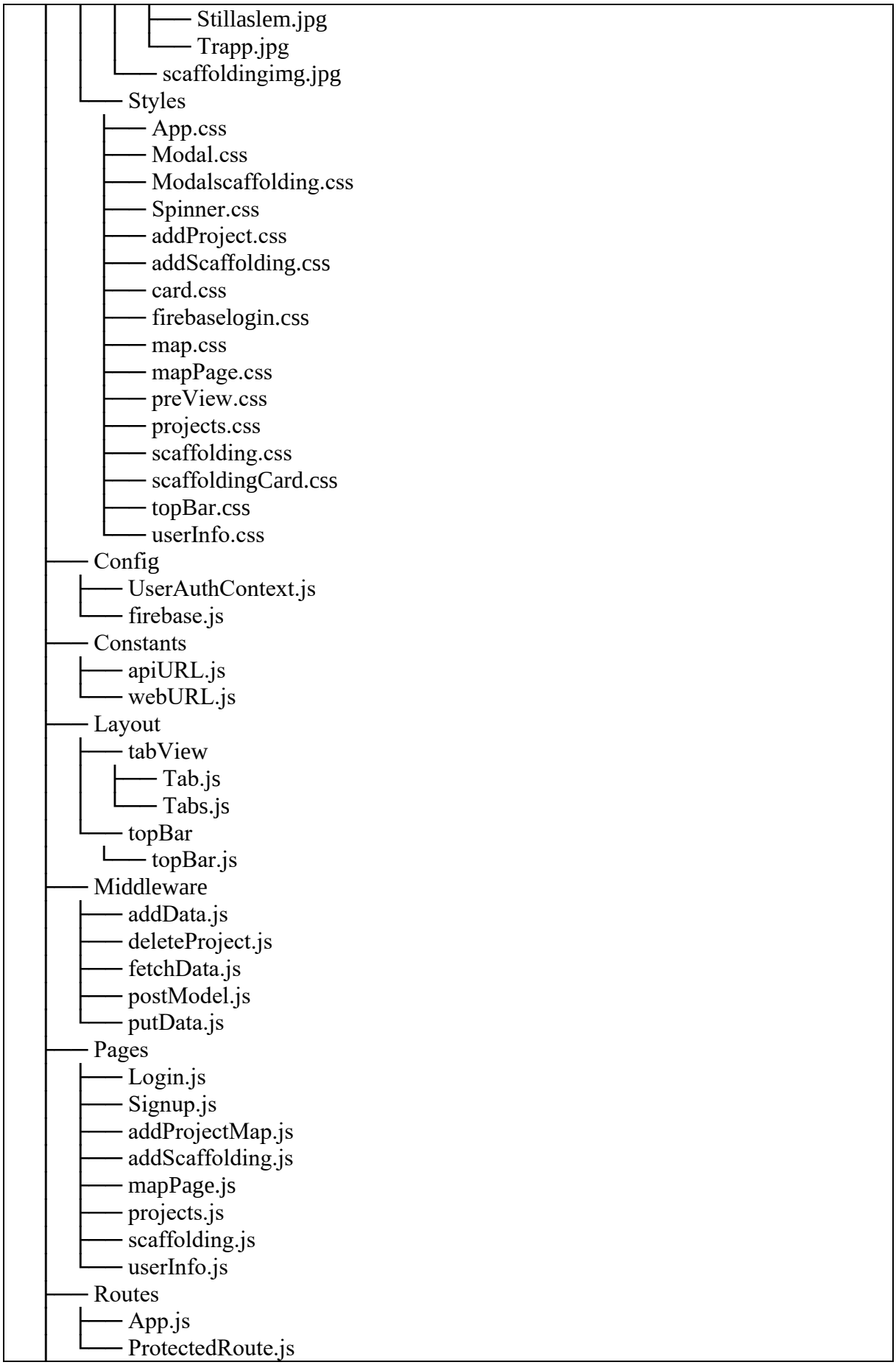
Endepunktene er designet basert på kategori, og på en måte som beskriver hva nettsiden viser. Prioriteringene gikk i korte og beskrivende endepunkt, gjør det lett for brukeren å navigere seg gjennom de forskjellige nettsidene. Konvensjoner som å separere ord med bindestrek og unnlate bruken av tegn som ikke støttes av UTF-8, da dette kunne bli problematisk for nettleseren å lese inn, har blitt brukt under designfasen av endepunktene. Et eksempel på endepunktet for å legge inn stillasdeler er: *http://localhost:3000/add-scaffolding*.

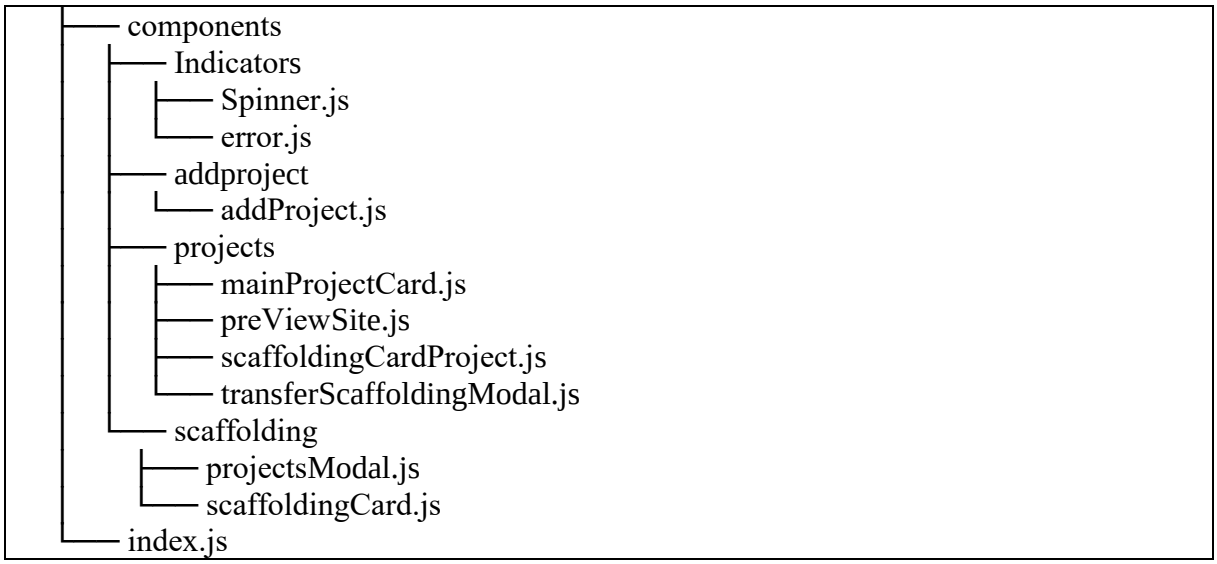
7.2.2.2 *Filstruktur*

Filene i webapplikasjonen ble i utgangspunktet lagt i en *Components*-mappe, med en mappeinndeling for hver nettside, sammen med filens CSS-fil. Filstrukturen ble konstruert slik, for å enkelt utvikle og designe nettsiden samtidig. Etter hvert som prosjektet utviklet seg med bilder og abstrahering av komponenter, ble filstrukturen uoversiktlig.

```
webstillas/
├── README.md
├── package.json
├── public
│   └── index.html
└── src
    ├── Assets
    │   ├── Images
    │   │   ├── marker.png
    │   │   ├── profile-png-icon-2.png
    │   │   └── scaffoldingImages
    │   │       ├── Bunnskrue.jpg
    │   │       ├── Diagonalstang.jpg
    │   │       ├── Enrørsbjelke.jpg
    │   │       ├── Gelender.jpg
    │   │       ├── Lengdebjelke.jpg
    │   │       ├── Plank.jpg
    │   │       ├── Rekkverksramme.jpg
    │   │       └── Spir.jpg
```

[21] https://www.apple.com og https://www.tesla.com/no_NO/inventory/used/ms?arrangeby=plh&zip=&range=0


Stillaslem.jpg
Trapp.jpg
scaffoldingimg.jpg
Styles
App.css
Modal.css
Modalscaffolding.css
Spinner.css
addProject.css
addScaffolding.css
card.css
firebaselogin.css
map.css
mapPage.css
preView.css
projects.css
scaffolding.css
scaffoldingCard.css
topBar.css
userInfo.css
Config
UserAuthContext.js
firebase.js
Constants
apiURL.js
webURL.js
Layout
tabView
Tab.js
Tabs.js
topBar
topBar.js
Middleware
addData.js
deleteProject.js
fetchData.js
postModel.js
putData.js
Pages
Login.js
Signup.js
addProjectMap.js
addScaffolding.js
mapPage.js
projects.js
scaffolding.js
userInfo.js
Routes
App.js
ProtectedRoute.js



Figur 28 Filstrukturen til webapplikasjon

Mot slutten av arbeidet ble filstrukturen endret, slik at den skulle være mer oversiktlig for andre utviklere. Den nye filstrukturen ble delt inn i hoved-mappene: *Assets*, *Components*, *Config*, *Constants*, *Layout*, *Middleware*, *Pages* og *Routes*.

Assets inneholder alt av bilder som blir brukt i prosjektet, samt alle CSS-filene. Alt av komponenter, slik som informasjonskort, inputfelt-grupper og modaler, ligge i *Components*-mappen. Innad i *Components*-mappen er de videre fordelt inn i mapper basert på i hvilken nettside de er brukt. *Constants* inneholder alle filene med definerte konstanter. *Config* inneholder filene som setter opp databasen og initialiserer innloggingsfunksjonaliteten. *Layout* inneholder filer som hjelper sideoppsettet i nettsiden, slik som topbaren og tab-viewen. Alt av databehandling slik som datahenting, sending og caching ligger i mappen *Middleware*. Hovedsidene ligger i en egen mappe, *Pages*. Hovedsidene er filene brukeren blir sendt til når en URL kalles på. Alle filene som arbeider med videresending til nettsider, ligger i mappen *Routes*.

7.2.3 Implementasjon

Websiden inneholder flere nettsider. Det er ruteren som gjør det mulig å navigere seg gjennom de forskjellige sidene i webapplikasjonen. Her måtte det defineres hvilket endepunkt som skulle brukes for å vise de enkelte React komponentene på siden (se Kode-utdrag 29 Utdrag fra Rute funksjonen, oppsett av funksjonen.

```
function App() {
    return (
        //Authorisation of user
        <UserAuthContextProvider>
            {/*Caching provider client*/}
            <QueryClientProvider client={queryClient}>
                <TopBar/> {/*Topbar for the user to navigate throughout
the webpage}*/}
                <Routes> {/*Router that creates the routes the user is
able to navigate*/}
                    <Route path={PROJECT_URL} exact={true}
element={<ProtectedRoute> <Project/></ProtectedRoute>}/>
```

```
                    <Route path={MAP_URL} exact={true}
element={<ProtectedRoute> <MapPage/></ProtectedRoute>}/>
                    <Route path={SCAFFOLDING_URL} exact={true}
element={<ProtectedRoute> <Scaffolding/></ProtectedRoute>}/>
                    <Route path={PROJECT_URL_ID} exact={true}
element={<ProtectedRoute> <PreView/></ProtectedRoute>}/>
                    <Route path={LOGIN} exact={true} element={<Login/>}/>
                    <Route path={SIGNUP} exact={true}
element={<Signup/>}/>
                    <Route path={ADD_PROJECT_URL} exact={true}
                           element={<ProtectedRoute>
<AddProjectFunc/></ProtectedRoute>}/>
                    <Route path={ADD_SCAFFOLDING_URL} exact={true}
                           element={<ProtectedRoute>
<AddScaffolding/></ProtectedRoute>}/>
                    <Route path={USERINFO_URL} exact={true}
element={<ProtectedRoute> <UserInfo/></ProtectedRoute>}/>
                    <Route path={NOTFOUND} element={<NotFound/>}/>
                </Routes>
                <ReactQueryDevtools initialIsOpen={true}/>
            </QueryClientProvider>
        </UserAuthContextProvider>

    );
}
```

Kode-utdrag 29 Utdrag fra Rute funksjonen, oppsett av funksjonen

Komponentene er designet som React funksjoner. Nettsidene er designet med enkle innebygde React komponenter, som *Input* og *Button*. Et fellestrekk som går igjen i applikasjonen er bruken av Hook funksjonen, *useState* (se kap. 2.6.3). Innebygde funksjoner har blitt brukt konsekvent gjennom applikasjonen for å endre på verdier i inputfelter, der brukerinput er mulig.

```
//Defining the json body to add a new unit
const [scaffolding, setScaffolding] = useState({
    id: "",
    type: "",
    batteryLevel: 100,
    location: {
        longitude: 0,
        latitude: 0,
        address: ""
    }
})
//Verification of a buttonPress
const [buttonPress, setButtonPress] = useState(false)
```

Kode-utdrag 30 Definering av variabler ved bruk av useState. Utdrag fra addProject.js.

```
<input type={"text"} className={"form-control scaffolding-input"}
onChange={event => {
    //Setting the id
    setScaffolding({...scaffolding, id: event.target.value})
}}/>
```

Kode-utdrag 31 Endringer av variabler ved bruken av set. Utdrag fra funksjonen scaffoldingInformation i filen addProject.js

Slik som vist i Kode-utdrag 30 har *useState* blitt brukt for å definere variabler. Her defineres selve variabelen som holder på en verdi, og en set-funksjon som endrer verdien. Det blir også vist eksempel på bruken av en set-funksjon i Kode-utdrag 31.
Mapbox (se kap. 2.6.5) sitt API har blitt brukt for å vise kart i applikasjonen. Dette API-et har gjort det mulig å legge inn funksjonalitet som annotasjoner og andre verktøy som gjør det mulig for brukeren å legge inn «geofence». Ved bruken av kartet ble det muligheter for å legge inn verktøy, bla. zooming og tegneverktøy.

```
return (
    <div className="App">
        <div className={"map"}>
            <Map
                style={MAP_STYLE_V11}
                containerStyle={{
                    height: "80vh",
                    width: "50vw"
                }}
                zoom={[17]}
                center={[Number(project.longitude),
Number(project.latitude)]}
            >
                <DrawControl
                    position="top-left"
                    displayControlsDefault={"polygon"}
                    onDrawCreate={onDrawCreate}
                    controls={{
                        point: false,
                        line_string: false,
                        combine_features: false,
                        uncombine_features: false
                    }}
                    default_mode={"polygon"}
                    onDrawDelete={() => setOk(false)}
                />
                <ZoomControl
                    position="bottom-right"
                />
            </Map>
        </div>
        <button className={"confirm-btn"} disabled={!ok || !props.valid}
onClick={AddProjectRequest}>Add Project</button>
    </div>
);
```

Kode-utdrag 32 Eksempel på hvordan kartet blir lagt inn. Utdrag fra addProject.js filen

Gruppen legger inn kartet som en komponent, sammen med ekstra funksjoner. Kode-utdrag 32, er et eksempel på hvordan har gruppen lagt inn ekstra funksjonalitet i karet, slikt som tegne funksjoner og zoom kontroll.

Informasjonen som blir vist i applikasjonen er hentet fra det utviklede API-et (se kap. 6.2). Gruppen har valgt å bruke «promises» (se kap. 2.6.6) for å hente ut dataen. Funksjonen *fetchModel*, er hentet fra et prosjekt introdusert i faget *WWW-teknologier*[22]. Funksjonen ble

[22] Les mer om faget her https://www.ntnu.no/studier/emner/PROG2053#tab=omEmnet

hentet fra tidligere kode, ettersom den utførte det gruppen ønsket oppnå. Gruppen bruker også aktivt async/await, for funksjonene som henter data ned fra et API.

```
export default function fetchModel(url) {
    return new Promise(function (resolve, reject) {
        const xhr = new XMLHttpRequest();

        xhr.addEventListener('readystatechange', () => {
            if (xhr.readyState !== 4) {
                return;
            }
            if (xhr.status !== 200) {
                reject({
                    status: xhr.status,
                    statusText: xhr.statusText,
                    text: xhr.responseText
                });
            } else {
                resolve({
                        statusCode: xhr.status,
                        text: xhr.responseText
                    }
                );
            }
        });
        xhr.open('GET',  BASE_URL + url);
        xhr.send();
    });
}
```

Kode-utdrag 33 Promise funksjon som gruppen bruker for å hente ut dataen av API-et. Utdrag fra fetchData.js

Promise funksjonen som gruppen bruker henter data gjennom en *XMLHttpRequest*. Dersom hentingen av dataen gir en statuskode på 200 (OK) vil promisen bli resolved. Dersom statuskoden ikke er lik 200, blir promisen rejected.

```
const AddScaffold = async () => {
    setButtonPressed(true)
    try {
        await putModel(TRANSFER_SCAFFOLDING, JSON.stringify(move))
        await queryClient.resetQueries(["project", props.id])
    } catch (e) {
        if (e.text === "invalid body"){
            window.alert("500 Internal Server Error\nNoe gikk galt! Prøv
igjen senere")
        }else {
            window.alert("Advarsel: Kan ikke overføre antall
stillasdeler")
        }
    }
}
```

Kode-utdrag 34 Eksempel på bruken av en promise. Utdrag fra Modal.js

Kode-utdrag 34 viser eksempel på hvordan promise funksjonen blir anvendt i koden. Dette er et eksempel på hvordan koden er synkront strukturert, fremfor en «chained promise».

Bruken av *react-query* (se kap. 2.6.5) har gjort det mulig å cache dataen hentet fra API-et. Dette ble implementert ved å bruke funksjonen *useQuery*. Denne funksjonen krevde en nøkkel som ble bundet til dataen hentet fra API-et. Denne dataen ble holdt lagret så lenge brukeren trykket seg gjennom applikasjoner, ved bruk av navigasjonsbaren.

```
export const GetCachingData = (dataName, url) => {
    const { isLoading, data, isError, isLoadingError} =
useQuery(dataName, ()=>{
        return fetchModel(url)
    }, {
        refetchOnMount: false,
        refetchOnWindowFocus: false,
        refetchOnReconnect: false

    })
    return {isLoading, data, isError, isLoadingError}
}
```

Kode-utdrag 35 Viser funksjonen som gruppen bruker for å hente data og cache data hentet fra API-et. Utdrag hentet fra addData.js

Det ble laget en egen funksjon til henting av data med *react-query*, noe som gjorde det mulig å cache dataen på lik måte. Bruken av funksjonen *useQuery*, tillot gruppen å sette egenskaper til den lagrede dataen. Dermed ble ikke dataen hentet på nytt når brukeren gikk tilbake til den aktive fanen, når nettsiden ble lastet på nytt og når dataen ble utdatert.

Dataen og funksjonaliteten som webapplikasjonen tilbyr er det kun arbeiderene i *MBStillas* som skal ha tilgang til. Innloggingsfunksjonalitet ble implementert for å holde denne dataen sikker fra eventuelle trussel aktører. Sikkerheten rundt programmet omtales i mer detalj i kap. 8.1. For å implementere denne funksjonaliteten ble *Firebase Authentication* brukt. Dette ble implementert ved bruk av integrerte *Firebase* funksjoner. Implementeringen av Authentication, åpner muligheten for autorisering av den enkelte brukerens rettigheter innad i applikasjonen. Dette er ikke implementert, men vil bli snakket om i kapittelet videre arbeid (se kap. 10.2).

```
const handleSubmit = async (e) => {
  e.preventDefault();
  setError("");
  try {
    await logIn(email, password);
    navigate(PROJECT_URL);
  } catch (err) {
    setError("Feil brukernavn eller passord. \nVennligst prøv igjen");
  }
};
```

Kode-utdrag 36 Eksempel på innloggingen av en bruker. Utdrag hentet fra Login.js

Kode-utdrag 36 viser hvordan en bruker logger seg inn i systemet med en funksjon fra *Firebase*. Brukeren skiver inn email og passord i et inputfelt, når innloggings-knappen aktiveres vil brukeren enten bli navigert til prosjekt siden, eller få en error om at brukernavn eller passord er feil.

7.2.4 Resultat

Funksjonaliteten som ble definert i MOSCOW tabellen, har blitt implementert i webapplikasjonen. Applikasjonen støtter kravene som ble satt av oppdragsgiver ved å vise prosjekter i kart og kunne holde logistikken. Funksjonalitet som kan optimalisere og sikre applikasjonen, har ikke blitt implementert. Dette gjelder oppdatering av prosjekt, samt legge inn autorisering for brukere. Dette blir videre diskutert i kap. 10.2.2.1.

Under fremlegg hos oppdragsgiver 09.mai fikk webapplikasjonen positiv tilbakemelding av design. Designet ble beskrevet som oversiktlig og brukervennlig sammenliknet med andre systemer oppdragsgiver har tatt i bruk.

7.2.5 Diskusjon

Dette underkapittelet vil ta for seg tanker og vurderinger gjort under implementeringen av webapplikasjonen.

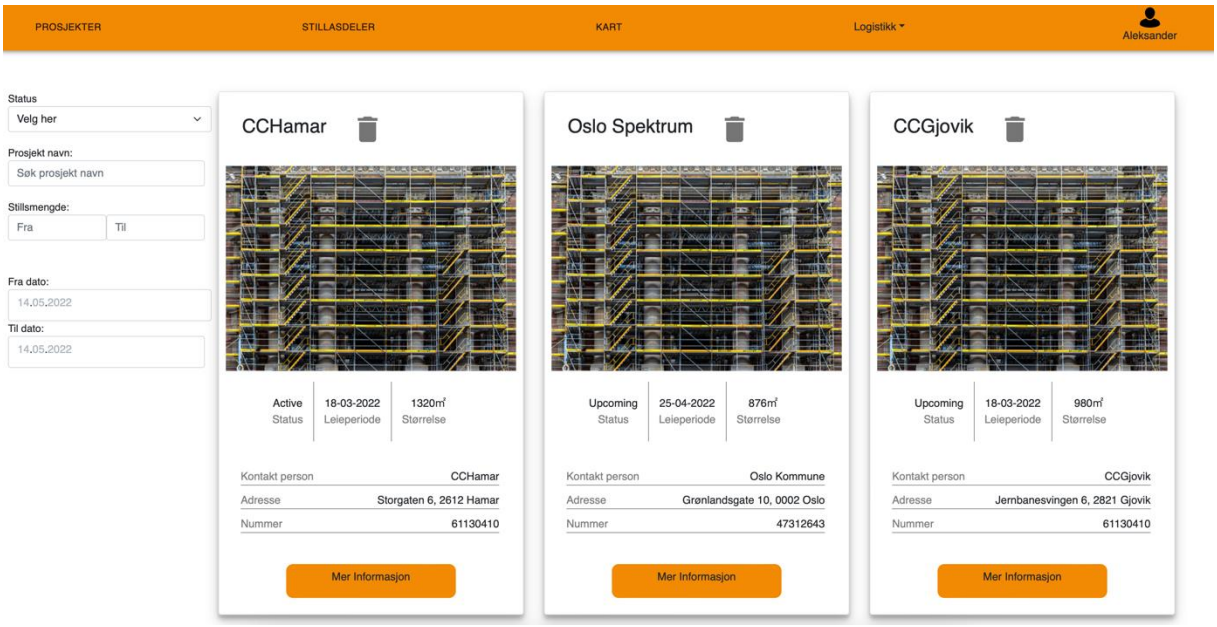
7.2.5.1 Design

Don Normans design prinsipper ble brukt for å skape best mulig opplevelse for brukeren. Webapplikasjonen tar i bruk en navigasjonsbar som gjør det enkelt for brukeren å navigere seg gjennom nettstedet. Valget av en slik navigasjons mekanisme falt naturlig på en navigasjonsbar på toppen av siden. Testobjektene synes navigasjonsbar var en fin måte å navigere seg gjennom nettstedet (se Appendix).

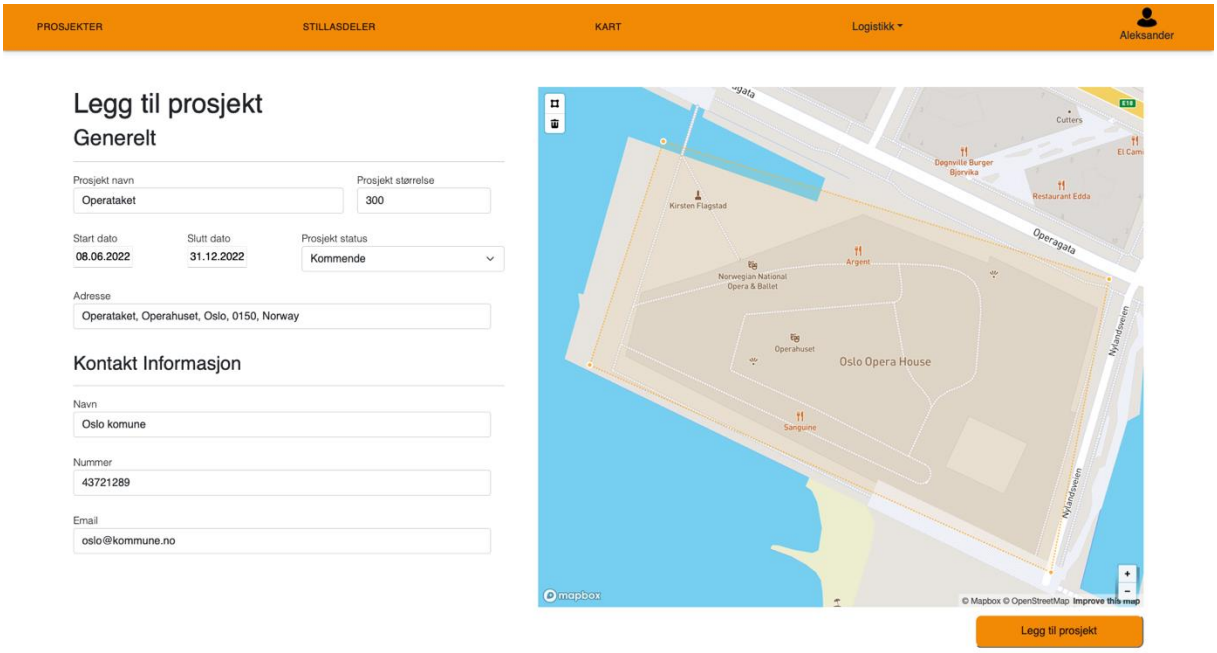
Konsekvent design er brukt på knappene, for å legge inn nye «gjenstander» og navigere seg gjennom nettsiden. Informasjonskort, som viser informasjon om stillasdeler, prosjekter og brukerinformasjon, gjør det lett for brukeren å identifisere informasjon i applikasjonen.

Webapplikasjonen bruker feedback for å gi brukeren en indikasjon på at det skjer noe i bakgrunn av systemet. Brukeren får feedback i webapplikasjonen ved bruk av en «spinner» som indikerer på at dataen hentes fra systemet. En spinner-knapp blir brukt for å indikere til brukeren at forespørselen blir behandlet.

Hovedforskjellene mellom utkastet og endelig design, var at systemet gikk bort i fra «barrelle» designet. En annen stor forskjell var at gruppen bestemte seg for å komprimere prosessen i å legge inn et prosjekt til én side, sammenliknet med 4 sider i designutkastet. Gruppen mente dette ville være belastende for brukeren, samt gjøre prosessen unødvendig lang. Det ble tatt et valg om å nøytralisere bakgrunnsfargen til kun hvit, for å gi et mer minimalistisk og moderne design. Etter tilbakemelding fra testobjektene, ble informasjonskortene komprimert i størrelse så langt det lot seg gjøre.



Figur 29 Endelig design av prosjektsiden



Figur 30 Design på legg til prosjekt

Grunnet bruken av *React Router*, var det ikke mulig å legge til spørringer på sortering eller filter i endepunktene. Dette er en egenskap som kunne ha gjort bruken av webapplikasjonen bedre. Brukeren kunne med denne egenskapen lagret endepunkt med ønsket filter, i motsetning til å legge inn filtrene på nytt hver gang.

7.2.5.1 *Klasser og Funksjoner*

Gruppen startet implementeringen av alle komponenter som klassekomponenter, ettersom dette ble lært i faget *WWW-teknologier*. Dette gjorde det lett å programmere hele nettsiden uten å måtte holde styr på hvor dataen måtte bli sendt videre for å kunne oppnå ønsket funksjonalitet.

Eksempel på dette er å binde verdier til et inputfelt. Ulempen som ble opplevd av å bruke klasse komponenter var at koden til tider ble uoversiktlig og tett sammenkoblet.

På en annen side, var det situasjoner der klasser hadde vært et bedre alternativ. Bruken av hook funksjonen *useState*, erstattet binding i funksjonskomponent. Som nevnt i kap. 2.6.3, er disse funksjonene asynkrone. Under error håndtering ble det opplevd problemer ved bruken av disse funksjonene.

```
const parseReverseGeo = async (lat, long) => {
    let street, postcode, region, place

…...

        let validStreet, validZip, validCounty, validMunicipality
            for (const re of res.features) {
                    switch (re.place_type[0]) {
                       case "address": {
                          street = re.text
                          if ((re.text.length !== undefined)) {
                              validStreet = true
                              }
                          }
                          break;
…...

                        case ("place") : {
                          place = re.text
                            if ((re.text.length !== undefined)) {
                               validMunicipality = true
                               }
                            }
                         break;
                         }
                    }
      if (validStreet && validZip && validCounty && validMunicipality) {
                         setValid({
                             ...valid,
                             countyValid: validCounty,
                             municipalityValid: validMunicipality,
                             zipcodeValid: validZip,
                             streetValid: validStreet
                         })
```

Kode-utdrag 37 Eksempel på hvordan gruppen løste problemet med aynkrone hooks

Ettersom *setValid* ble eksekvert i en sekvens med kort tidsintervall, rakk ikke verdiene å bli satt før den neste skulle settes. Dette resulterte i at kun den siste verdien ble endret. Her kunne gruppen istedenfor en funksjon komponent, brukt en klasse komponent da denne typen er synkron, og kraftigere.

7.2.5.2 *Databehandling*

Da caching skulle implementeres, fant gruppen ut at biblioteket brukte hook funksjoner. Som nevnt i teorien, støtter ikke React klasser hook funksjoner, noe resulterte i omstrukturering av koden fra klasser til funksjons komponenter. I tillegg til at vi fikk utnyttet oss av caching, ble kodekvaliteten forbedret ved oppnå mer loose coupling og high cohesion. Der variabler kunne

direkte bindes i klassen, ved bruken av *this.state*, ble bindingen muliggjort ved hook funksjonen *useState*.
Et behov for caching ble tidlig tydelig. Dette var en kombinasjon av responstiden til dataen, og antall lesinger i databasen. Webapplikasjonen trengte en caching som ble holdt i et par minutter, og ble oppdatert etter et kort tidsintervall.

Noen av alternativene som ble vurdert for caching var *sessionStorage*, *localStorage* og *cookies*. Ulempen med lokal lagring av dataen, var at dataen ikke ble fornyet. Dette hadde ikke passet inn i systemet, ettersom det kunne skje oppdateringer i dataen etter et gitt tidsintervall (se kap. 2.6.7). *Cookies* er ikke egnet for å holde store mengder med data, noe applikasjonen bruker, dermed ble dette skrinlagt. Ut ifra kravene som ble satt, ble *sessionStorage* mest attraktivt. Noe av problemene vi så med denne type lagring var dersom brukeren hadde nettleser fanen aktiv for lenge, kunne noe av dataen i databasen endres av andre brukere. Dataen kunne dermed ikke være oppdatert, noe som kunne forårsake feil ved eksempelvis flytting av stillasdeler. I tillegg var det utfordrende når en bruker la til nye stillasdeler eller prosjekter. Grunnen til dette er at dataen måtte slettes når noe ble lagt til i databasen, deretter hente dataen på nytt (se Kode-utdrag 38).

```
    if (sessionStorage.getItem('allProjects') == null) {
         try {
            const project = await fetchData(PROJECTS_WITH_SCAFFOLDING_URL)
            sessionStorage.setItem('allProjects', JSON.stringify(project))
            this.setState({
                 isLoaded: true,
                 projectData: project,
             });
         } catch (error) {
             console.log(error)
         }
     } else{
         this.setState({
             isLoaded: true,
         });
     }
};
```

Kode-utdrag 38 Hvordan gruppen sjekket og lagret data dersom den ikke hadde blitt hentet før

Gruppen kom over biblioteket *react-query*, som tilfredsstilte alle kravene for caching. Denne gjorde det mulig å lagre dataen i et gitt tidsintervall, og hente dataen på ny dersom brukeren gjorde en endring i systemet.

7.2.5.3 *Autentisering*

Autentiseringen som er implementert i programmet er hentet fra kildekode[23] på *GitHub*. Bruken av denne kildekoden ligger i filene *Login.js* og *Signup.js*. Kildekoden passet bra, da den hadde alt av funksjonalitet som krevdes. Koden som ble hentet fra *GitHub*, bruker *Firebase* sine innebygde funksjoner til å autorisere brukeren, og legge inn en ny bruker. Gruppen har også tatt i bruk React koden til innlogging siden og registrering siden, med noen endringer. Disse

[23] https://github.com/WebDevSimplified/React-Firebase-Auth

endringene var nødvendig med tanke på sikkerhetsaspektet og integrering i systemet. Et eksempel er å legge inn mer detaljer for brukeren, samt generalisere errormeldingene. Dette blir snakket mer om i kap. 8.1.

7.2.5.4 *Generelt*

Gruppen har hentet kilde kode fra *digitalocean* som muliggjør å legge inn forskjellige faner innad i en komponent. Denne koden ligger i *Tab.js* og *Tabs.js*. Gruppen valgte denne koden ettersom den hadde ønsket funksjonalitet, samt at den var enkel å implementere.

Slik webapplikasjonen er designet, er den ikke optimalisert for vertikal skalering. Prosjekt-siden henter alle prosjekter med stillasdeler, noe som har en lang responstid. Denne dataen blir brukt videre i stillasdel-siden. Dataen blir sammen med dataen fra lageret brukt for å regne ut totalt innhold av stillasdelene i systemet, noe som resulterer i lang prosesseringstid.

7.3 Mobilapplikasjon

I dette kapittelet sees det innledningsvis på hvilke krav som ble stilt til applikasjonen. Videre sees det på design av grensesnittet og filsystemet før implementasjonen diskuteres. Avslutningsvis oppsummeres resultatene for hva som ble gjennomført og så diskuteres de ulike valgene gjort rundt applikasjonen.

7.3.1 Krav

Oppdragsgiver ønsket seg en mobilapplikasjon for å kunne ha lett tilgang på informasjonen om hvor stillaset deres befinner seg og enkelte operasjoner som for eksempel overføring av stillas for montører ute «i felten». Innledningsvis oppsto spørsmålet rundt hvilken plattform applikasjonen skulle utvikles på. Oppdragsgiver ønsket seg originalt en applikasjon som fungerte på både Android og iOS plattformen. Gruppen og oppdragsgiver så seg nødt til å avgrense oppgaven til kun å gjelde ett operativsystem. Valget falt dermed på en iOS basert native applikasjon, ettersom iOS er det dominerende operativsystemet hos oppdragsgiver.

For å definere hvilken funksjonalitet applikasjonen skulle inneholde ble kravspesifikasjonen benyttet sammen med MOSCOW konsept evalueringsmodellen. Gruppen gikk gjennom kravspesifikasjonen og hva oppdragsgiver ønsket og satte opp modeller av hva som virket oppnåelig og gjennomførbart. I fellesskap med oppdragsgiver kom gruppen fram til følgende modell (se Tabell 7):

MOSCOW-modellen:

Must have	- Liste byggeprosjekter - Overføre stillasdeler mellom prosjekter
Should have	- Liste alle stillasdeler med tilhørende mengde - Ha brukere
Could have	- Oppdatere informasjon om prosjekt - Vise historie/log for mengden stillas av hver del på et prosjekt - Vise stillasdeler i kart

	- Registrere ny bruker
Won't have	- Registrere ny stillasdel - Registrere ny og slette prosjekt

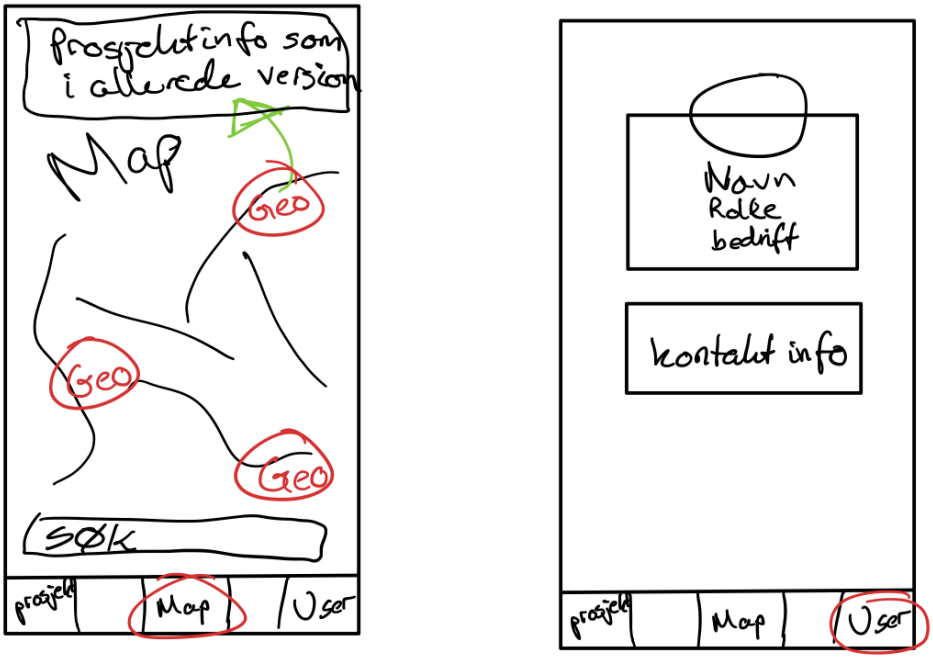
Tabell 7 MOSCOW modellen for mobilapplikasjonen

7.3.2 Design

I de kommende seksjonene vil designvalg gjort med hensyn på brukergrensesnittets- og fildesignets implementasjon presenteres.

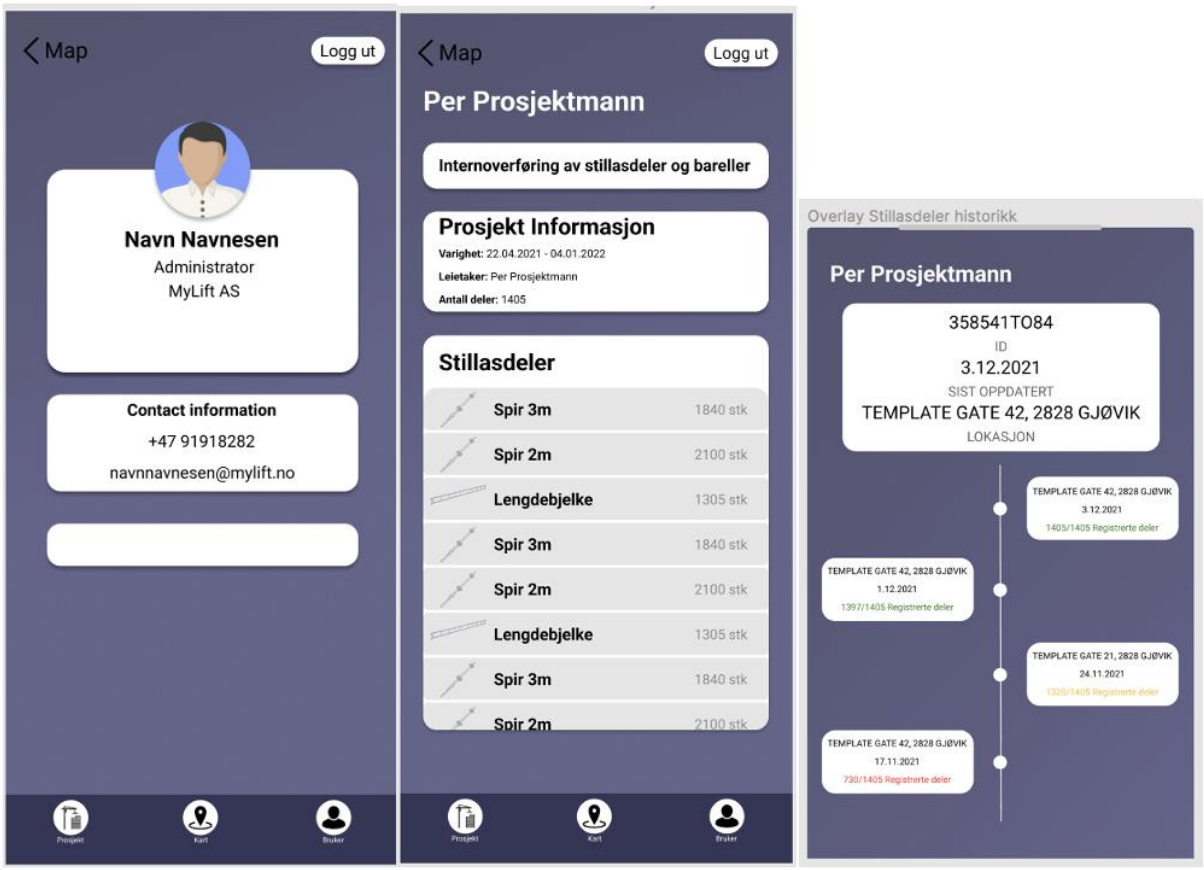
7.3.2.1 *Grensesnitt*

Innledningsvis i applikasjonsutviklingen ble det laget sketsjer basert på kravspesifikasjonen og MOSCOW modellen. Disse sketsjene var røffe utkast av hvordan applikasjonen kunne se ut med piler for å vise hvilke handlinger som skulle føre til hvilke responser fra systemet. Eksempel på hvordan sketsjene ble seende ut kan sees i Figur 31.

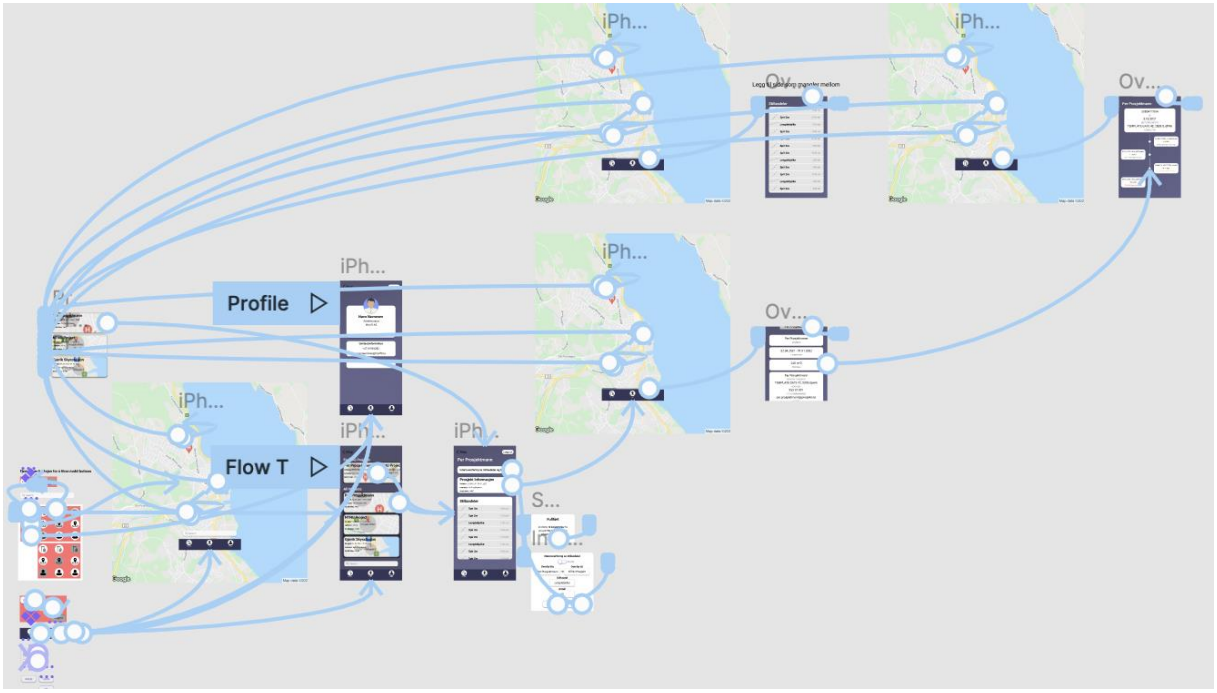


Figur 31 Sketsj av mobilapplikasjon design

For høy-funksjonell prototyping ble *Figma* benyttet. Her ble sketsjene utformet mer detaljert som en prototype brukeren kan samhandle med og navigere seg gjennom. Her startet fokuset på Interaksjonsdesignet (se kap. 7.1.1). I Figur 31 er eksempler på designet av profil-siden, prosjektinformasjon-siden og historie for stillasdeler registrert i applikasjonen. Figur 33 viser hvordan de ulike sidene er koblet sammen for prototyping.



Figur 32 Utdrag fra Figma prototypen av mobilapplikasjonen

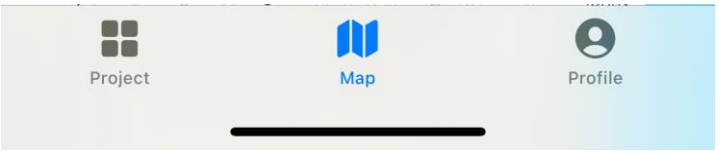


Figur 33 Eksempelutdrag av interaksjonskart i Figma

For utviklingen av selve applikasjonen ble Human Interface Guidelines (se kap. 7.1.3.2) benyttet gjennomgående sammen med WCAG 2.1 retningslinjene (se kap. 7.1.3.1). Alt fra navigerings strategi, hvordan tilbakemeldinger gis til brukeren, generelt design rundt knapper og hvordan fokus skal oppdateres når tekstfelter aktiveres ble vurdert her. Et prinsipp vi har

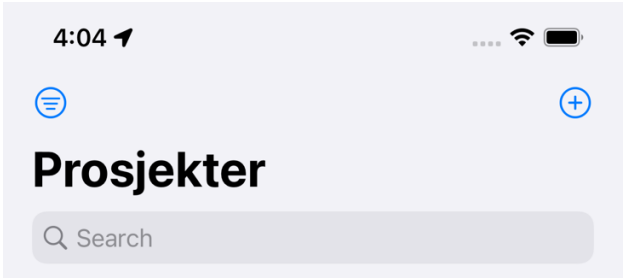
fulgt er konsekvent design hvor elementer, tekststiler og terminologi, samt velkjente ikoner tilbudt av systemet benyttes der det er hensiktsmessig. På denne måten integreres handlinger og elementer som brukeren forventer og kjenner til. Dette fører til en bedre brukeropplevelse (se kap. 7.1).

Applikasjonen er designet med en fanelinje nederst på skjermen som tillater brukeren å navigere seg mellom de tre sentrale sidene (se Figur 34).



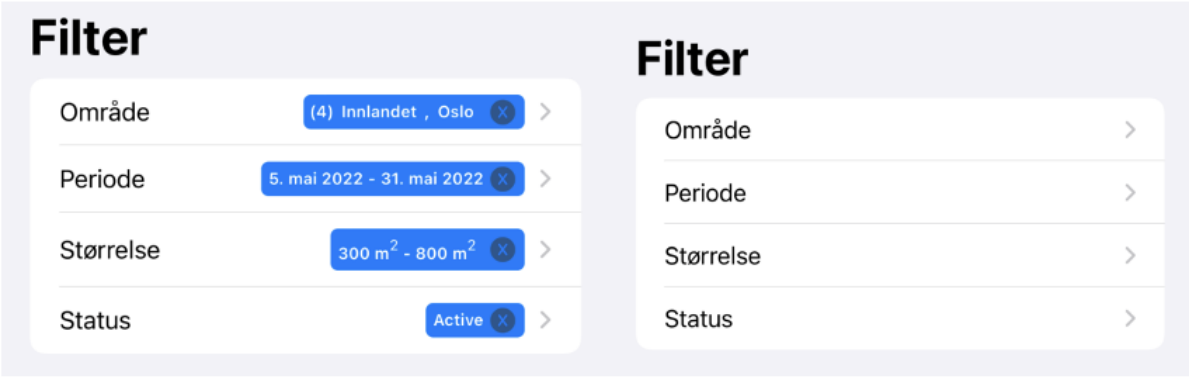
Figur 34 Navigasjonsbar for de tre hovedsidene i mobilapplikasjonen

Under prosjekt er tilretteleggingen for filtrering og å legge til nye prosjekter lagt til i navigasjonsbaren som ikoner tilbudt av systemet (se Figur 35). Denne bruken av systemikoner er benyttet gjennomgående og konsekvent hvor like ikoner tilbyr lik funksjonalitet.



Figur 35 Navigasjonsbar i prosjekter med ikoner for filtrering og opprettelse av prosjekter i mobilapplikasjonen

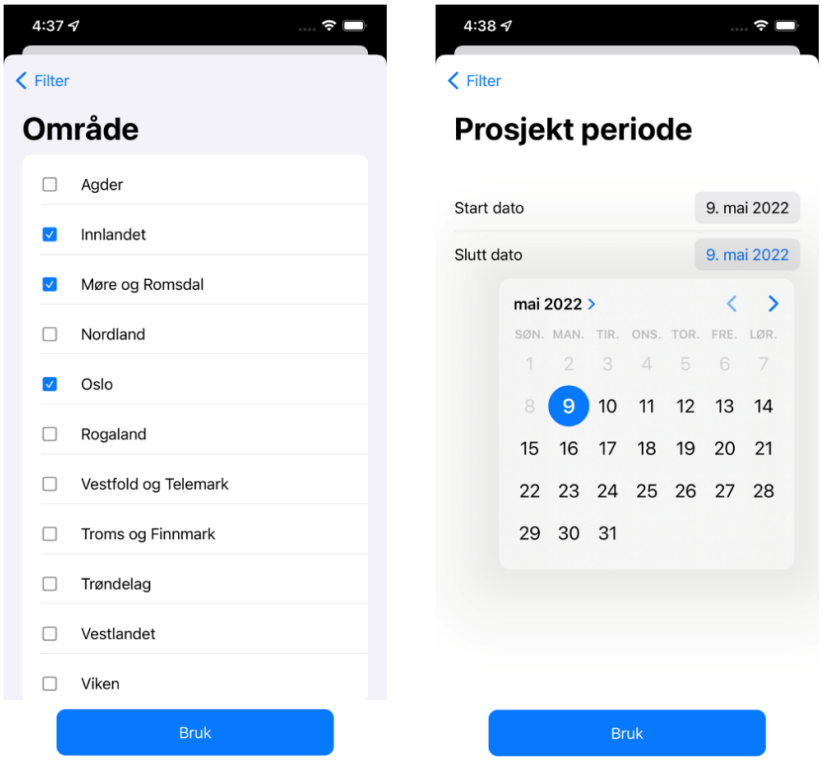
Filtrering er innført som et *sheet* (en skuffe), som åpner seg over prosjekt siden. Denne er delt inn i fire underkategorier *Område*, *Periode*, *Størrelse* og *Status* (se Figur 36). Hver av disse aktiveres med en blå markering som viser brukerens filtreringsvalg. Disse kan fjernes ved å krysses ut.



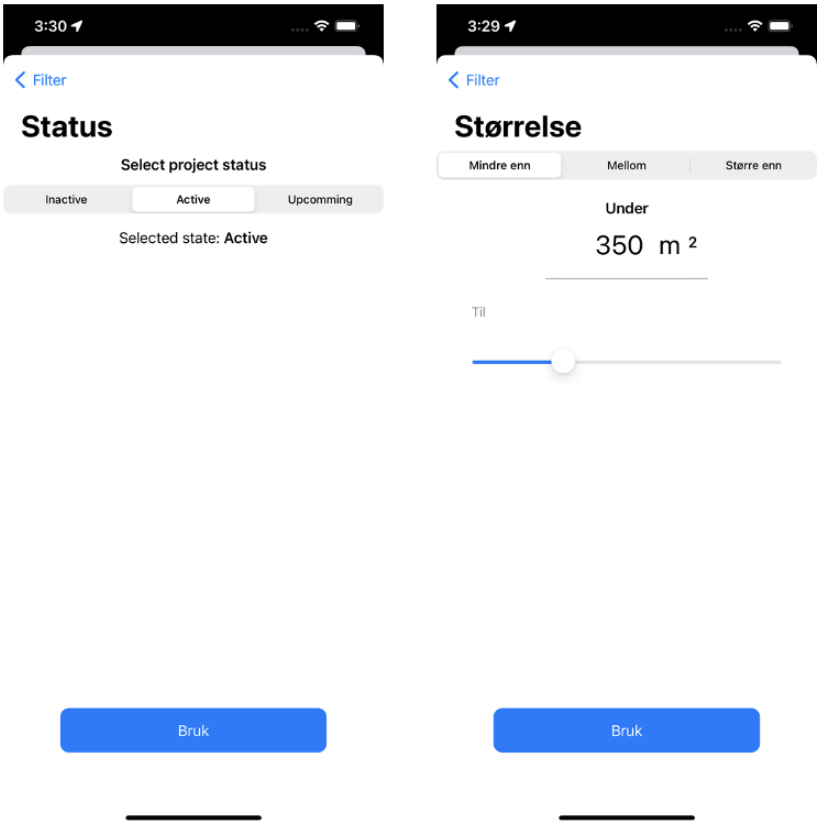
Figur 36 Filter med valgte filter (venstre) og uten filter (høyre) i mobilapplikasjonen

De ulike filtreringssidene er designet minimalistisk med kun den nødvendige og relevante dataen. *Område* er designet med firkantede avkrysningsfelt som tillater og impliserer at

brukeren kan gjøre flere valg (se Figur 37). *Periode* er designet med en start- og sluttdato som kan samhandles med av brukeren. Denne er designet i henhold til Human Interface Guidelines (se kap. 7.1.3.2) hvor brukeren får opp kalenderen for dato valg (se Figur 37). *Størrelse* er designet som tre ulike sider underlagt en *tab-bar*. En for prosjekter mindre enn-, en for prosjekter større enn-, og en for prosjekter mellom to oppgitte størrelser. Her får brukeren et tekstfelt de kan redigere manuelt eller de kan benytte en *slider* som oppdaterer dette feltet. Disse to samhandler og oppdaterer hverandre. Her også er minimalistisk design forsøkt ivaretatt (se Figur 38). Avslutningsvis er *Status* underlagt en *tab-bar* som velger om prosjekter er Aktive, Inaktive eller Kommende (se Figur 38).

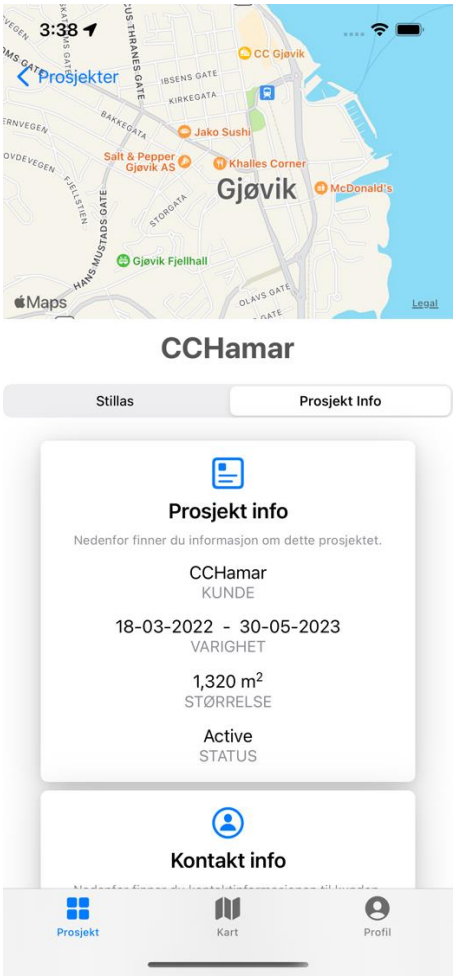


Figur 37 Filter for prosjekt Område og Periode i mobilapplikasjonen



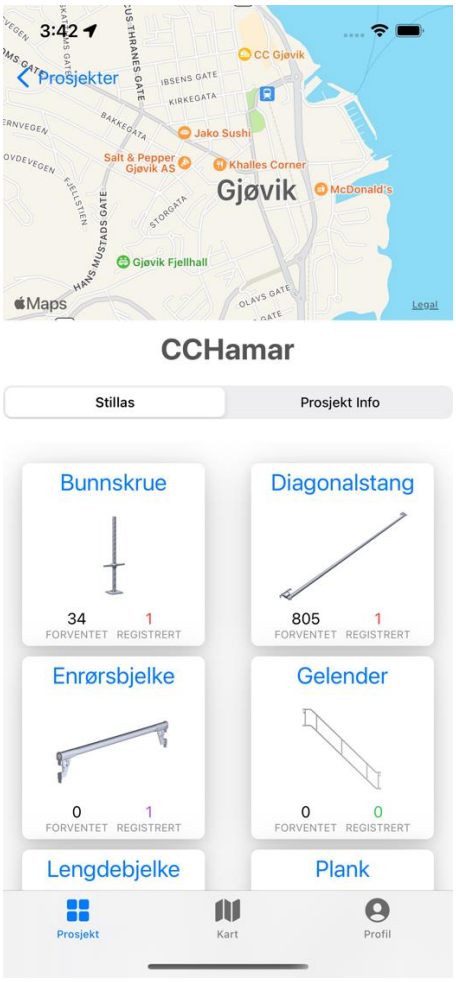
Figur 38 Filter for prosjekt Status og Størrelse i mobilapplikasjonen

Prosjektsiden viser informasjon om prosjektet og hvilke stillasdeler som befinner seg der. Prosjekt infosiden og stillasdelsiden er underlagt prosjektsiden som to under sider. På denne måten separeres to separate under-funksjonaliteter for å ikke forvirre brukeren med for mye informasjon på samme side. Prosjekt info er inndelt i praktisk informasjon om prosjektet og kundens kontaktinfo i to separate informasjonskort. På denne måten ledes brukerens syn til de ulike overkategoriene (se Figur 39).



Figur 39 Prosjekt informasjon i mobilapplikasjonen

Stillasdeler siden er lagt opp som et rutenett av knapper. Hver av knappene tilhører hver enkelt stillasdel og har deskriptiv og konkret forhåndsvisningsinformasjon om delen. Tittel i annen farge for å markere navnet på delen, bilde av delen, samt forventet og registrert mengde. All den informasjonen brukeren behøver vises i en kort oppsummering for å redusere nødvendige interaksjoner fra brukeren for å gjennomføre en oppgave (se Figur 40).

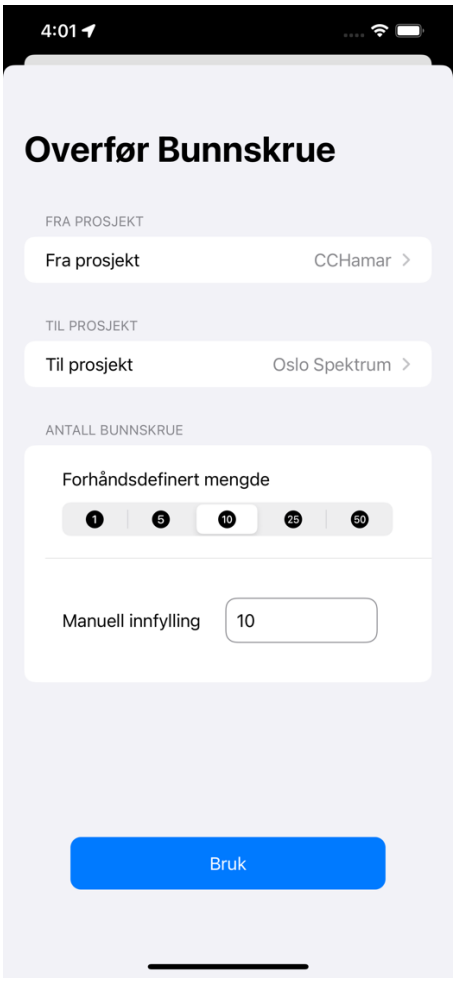


Figur 40 Stillasdeler for et prosjekt i mobilapplikasjonen

Stillasdel-detaljsiden består av en historieside som er designet som en vertikal, rullbar tidslinje med nyeste historie øverst. Denne er også holdt minimalistisk og beholder det konsekvente designet i hvordan forventet og registrert stillas vises. Her er også knappdesignet på bunnen av siden for overføring av stillas holdt konsekvent og likt som «Bruk-knappene» i appen (se Figur 41).

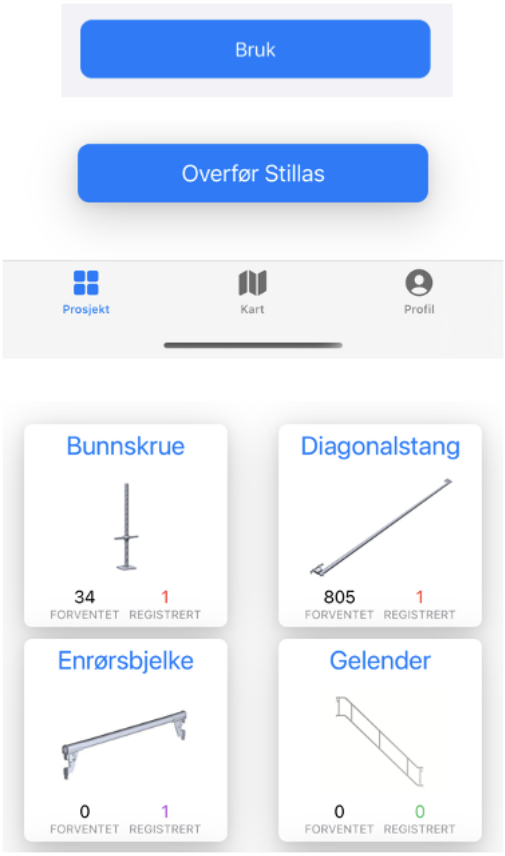
Figur 41 Historie for stillasdel på prosjekt i mobilapplikasjonen

Overføring av stillasdel er strukturert i en egen *sheet-side* som er strukturert kronologisk nedover med informasjon brukeren må fylle inn. Dette veileder brukeren gjennom de nødvendige feltene og valgene, og er designet så minimalistisk og konkret som mulig for å forhindre feil med luft mellom de ulike seksjonene for å tydelig separere oppgaver (se Figur 42).



Figur 42 Overføring av stillasdel i mobilapplikasjonen

Designet har gjennomgående blitt holdt konsekvent hvor like oppgaver på ulike sider ser like ut. Dette er særlig tilfellet for knapper som «Bruk» knappen og navigering. Fargebruken er konsekvent hvor blant annet knapper og handlinger som leder videre til nye sider eller handlinger er blå (Se Figur 43).



Figur 43 Knappbruk (Bruk, Overfør, Navigering og Stillasdel) i mobilapplikasjonen

7.3.2.2 *Teknisk design*

Fildesignet for applikasjonen kan sees i Kode-utdrag 39. Filene er inndelt i henhold til MVVM arkitektur mønsteret (se kap. 2.8.1.2) hvor *Views*, *Models* og *ViewModels* er forsøkt separert. Funksjonalitet er inndelt slik at de ulike filene stort sett kun inneholder én *View*, eventuelt kun funksjonalitet relevant for de andre klassene, funksjonene og *View*-ene i filen.

```
stillasMobileApplication/
├── stillasMobileApplication
│   ├── Assets.xcassets
│   │   ├── AccentColor.colorset
│   │   │   └── Contents.json
│   │   ├── AppIcon.appiconset
│   │   │   └── Contents.json
│   │   ├── Contents.json
│   │   ├── Profile
│   │   │   ├── Contents.json
│   │   │   └── UserProfile.imageset
│   │   │       ├── Contents.json
│   │   │       └── clipart2557794.png
│   │   └── Scaffolding
│   │       ├── Bunnskrue.imageset
│   │       │   ├── Bunnskrue.jpeg
│   │       │   └── Contents.json
```

```
│   │       ├── Contents.json
│   │       ├── Diagonalstang.imageset
│   │       │   ├── Contents.json
│   │       │   └── Diagonalstag.jpeg
│   │       ├── Enrørsbjelke.imageset
│   │       │   ├── Contents.json
│   │       │   └── Enrørsbjelke.jpeg
│   │       ├── Gelender.imageset
│   │       │   ├── Contents.json
│   │       │   └── Gelender.jpeg
│   │       ├── Lengdebjelke.imageset
│   │       │   ├── Contents.json
│   │       │   └── Lengdebjelke.jpeg
│   │       ├── Plank.imageset
│   │       │   ├── Contents.json
│   │       │   └── Plank.jpeg
│   │       ├── Rekkverksramme.imageset
│   │       │   ├── Contents.json
│   │       │   └── Rekkverk.jpeg
│   │       ├── Spir.imageset
│   │       │   ├── Contents.json
│   │       │   └── Spir.png
│   │       ├── Stillaslem.imageset
│   │       │   ├── Contents.json
│   │       │   └── Stillaslem.jpeg
│   │       └── Trapp.imageset
│   │           ├── Contents.json
│   │           └── Trapp.jpeg
│   ├── ContentView.swift
│   ├── GoogleService-Info.plist
│   ├── Login&Signup
│   │   ├── AppViewModel.swift
│   │   ├── Login
│   │   │   └── SignInView.swift
│   │   └── SignUp
│   │       └── SignUpView.swift
│   ├── Preview Content
│   │   └── Preview Assets.xcassets
│   │       └── Contents.json
│   ├── Views
│   │   ├── BlurView
│   │   │   └── BlurView.swift
│   │   ├── Cache
│   │   │   ├── CacheEntry.swift
│   │   │   └── CacheImplementation.swift
│   │   ├── Helpers
│   │   │   ├── CheckBoxRow.swift
```

```
│   │   │   ├── CircleImage.swift
│   │   │   ├── CornerRadiusStyle.swift
│   │   │   └── TextFieldModifiers.swift
│   │   ├── MapView
│   │   │   ├── MapDisplay.swift
│   │   │   ├── MapView.swift
│   │   │   └── MapViewModel.swift
│   │   ├── Model
│   │   │   ├── FetchingApiData
│   │   │   │   ├── ProfileData.swift
│   │   │   │   └── ProjectData.swift
│   │   │   ├── Profile.swift
│   │   │   ├── Project.swift
│   │   │   ├── ScaffoldingTransfere.swift
│   │   │   └── User.swift
│   │   ├── NavigationBarBottom.swift
│   │   ├── ProfileView
│   │   │   └── ProfileView.swift
│   │   └── ProjectView
│   │       ├── FilterData
│   │       │   ├── AreaFilter
│   │       │   │   └── FilterProjectArea.swift
│   │       │   ├── FilterView
│   │       │   │   ├── ActiveFilterViews
│   │       │   │   │   ├── ActiveAreaFilterView.swift
│   │       │   │   │   ├── ActivePeriodFilterView.swift
│   │       │   │   │   ├── ActiveSizeFilterView.swift
│   │       │   │   │   └── ActiveStatusFilterView.swift
│   │       │   │   └── FilterView.swift
│   │       │   ├── PeriodFilter
│   │       │   │   ├── CalendarView.swift
│   │       │   │   └── FilterProjectPeriod.swift
│   │       │   ├── SizeFilter
│   │       │   │   ├── BetweenGreaterLess
│   │       │   │   │   ├── SizeGreaterThanFilter.swift
│   │       │   │   │   └── SizeLessThanFilter.swift
│   │       │   │   └── FilterProjectSize.swift
│   │       │   └── StatusFilter
│   │       │       └── FilterProjectStatus.swift
│   │       └── ProjectView
│   │           ├── ProjectDetailView
│   │           │   ├── HistoryOfScaffolding.swift
│   │           │   ├── ProjectInfoDetailedView.swift
│   │           │   ├── ProjectInfoView.swift
│   │           │   ├── ScaffoldingDetailedView.swift
│   │           │   ├── ScaffoldingItem.swift
│   │           │   ├── ScaffoldingItems.swift
```

```
│   │       │   ├── ScaffoldingView.swift
│   │       │   ├── TransfereScaffolding.swift
│   │       │   ├── TransfereScaffoldingButton.swift
│   │       │   └── TransfereScaffoldingView.swift
│   │       └── ProjectListView.swift
│   └── stillasMobileApplicationApp.swift
└── stillasMobileApplication.xcodeproj
```

Kode-utdrag 39 Filsystem mobil-applikasjon

7.3.3 Implementasjon

I dette underkapitlet fremlegges implementasjonsvalg og hvordan disse ble brukt i praksis.

7.3.3.1 *Model-View-ViewModel*

Applikasjonen er bygd opp på MVVM design mønsteret (se kap. 2.8.1.2). Dette innebærer en struktur som abstraherer koden ut i *View*, *Models* og *ViewModels*. Ettersom applikasjonen henter data fra-, og kommuniserer data til API-et og bruker denne dataen ulikt på forskjellige steder i appen er dette design mønsteret gunstig. På denne måten behøves det ikke at *ViewModel* varsles og må oppdatere *Viewen* når data endres i bakgrunnen for data som ikke er aktiv på siden brukeren benytter. Ellers abstraheres også GUI-relatert kode vekk fra business- og back-end logikk.

Nedenfor er et utdrag av koden som illustrerer hvordan MVVM er benyttet i applikasjonen. Her benyttes en innloggings-*View* for GUI funksjonalitet (se Kode-utdrag 40). I denne instansieres et *EnvironmentObject* (et observerbart objekt) av *AppViewModel* som kobler *View* og *Model* sammen. Brukeren fyller inn email og passord i *TextField* og *SecureField*. Denne dataen sendes så inn i *Model* sin *signIn*-funksjon. *Model* dataen finner man i klassen *AppViewModel* (se Kode-utdrag 41). Her prosesseres innloggingsforespørselen ved at den sjekkes opp mot dataen i *Firebase Authentication* for innlogging. På denne måten har man oppnådd en abstrahering av de ulike lagene i arkitekturen (se kap. 2.8.1.1).

```
struct SignInView: View {
    @State var email = ""
    @State var password = ""

    /// The model responsible for sign in
    @EnvironmentObject var viewModel: AppViewModel
...

Button(action: {
    guard !email.isEmpty, !password.isEmpty else {
      return
    }
     viewModel.signIn(email: email, password: password)
     }) {
        Text("Sign in")
            .frame(width: 150, height: 50, alignment: .center)
```

```
    }
...
}
```

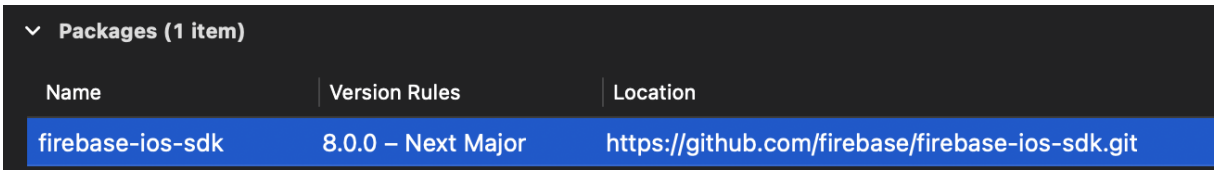
Kode-utdrag 40 SignInView - Innloggingssiden for mobilapplikasjonen

```
class AppViewModel: ObservableObject {

   let auth = Auth.auth()
    /// Is user signed in?
   @Published var signedIn: Bool = false

   ...

   /// Attempts to sign in user to the application by authorizing the user through the Firebase
Authentication
   /// - Parameters:
   ///   - email: The email of the user
   ///   - password: The password of the user
   func signIn(email: String, password: String) {

      /// Attempts to authorize the credentials in Firebase Authentication
      auth.signIn(withEmail: email, password: password) {
                        [weak self ] (result, error) in
         guard result != nil, error == nil else {
            return
         }
         DispatchQueue.main.async {
            // Success
            self?.signedIn = true
         }
      }
   }
   ...
}
```

Kode-utdrag 41 AppViewModel - Model klassen for innlogging gjennom Firebase Authentication

7.3.3.2 *Innlogging og registrering*

Mobilapplikasjonen benytter seg av *Firebase Authentication* biblioteket (se kap. 2.8.1.4). Denne benyttes for å håndtere brukerinnlogging, brukerregistrering og økt-håndtering. Måten dette er gjort på er at brukeren fyller ut innloggingsinformasjonen og sender forespørselen til systemet. Dette omdirigeres til *Firebase Authentication* som godkjenner eller avviser innloggingsforsøket. Autoriseres brukeren får brukeren tilgang til applikasjonen, om ikke avvises brukeren og bes om å forsøke å logge inn på nytt. Et utdrag av koden benyttet for innlogging kan sees i Kode-utdrag 40 og Kode-utdrag 41 eller i filene *AppViewModel.swift* og *SignInView.swift/SignUpView.swift* og prosessen er omtalt i kap. 7.3.2.1. Figur 44 viser *Firebase Authentication* biblioteket i Xcode.



Figur 44 Firebase Authentication i Xcode

7.3.3.3 *Kart*

Når det gjaldt implementasjonen av kart var det mange kart varianter som kunne brukes, deriblant *Apples MapKit, Google Maps* og *Mapbox*. Valget falt på å bruke Apples rammeverk for kart og satellitt bilder *MapKit* (se kap. 2.8.1.3). Kartet er en av de tre hovedsidene i applikasjonen. Her skal prosjektene vises på deres lokasjon med annoteringer, samt brukerens nåværende lokasjon dersom brukeren tillater dette. Prosjektene vises i kart ved innlasting fra fil, men fungerer ikke når dataen hentes fra API-et. Mer om dette i kap. 7.3.4 og kap 7.3.5. *MapKit* ble implementert ved å bruke *MKMapView* kart grensesnittet. Dermed kunne annoteringer enkelt tas i bruk for prosjekter. I Kode-utdrag 42 MapDisplay - Lager og oppdaterer kartet i mobilapplikasjonen lages et kart grensesnitt som dekker hele skjermen.

```
struct MapDisplay: UIViewRepresentable {
    ...
    /// Makes the MKMapView
    /// Allows to show user location, sets tracking mode and region of interest on "open"
    /// - Parameter context: A context structure containing information about the current state of the system
    /// - Returns: A MKMapView
    func makeUIView(context: Context) -> MKMapView {
        let mapView = MKMapView(frame: UIScreen.main.bounds)
        mapView.showsUserLocation = true
        mapView.userTrackingMode = .follow
        mapView.setRegion(viewModel.region, animated: true)
        updateUIView(mapView)
        return mapView
    }

    /// Updates the MKMapView
    /// - Parameter uiView: The MKMapView to be updated
    func updateUIView(_ uiView: MKMapView) {
        let annotations = ProjectListView().projects.map { project -> MKAnnotation in
            let annotation = MKPointAnnotation()
            annotation.title = project.projectName
            annotation.subtitle = "\(project.projectID)"
            annotation.coordinate = CLLocationCoordinate2D(latitude: project.latitude, longitude: project.longitude)
            return annotation
        }
        uiView.addAnnotations(annotations)
    }
    ...
}
```

Kode-utdrag 42 MapDisplay - Lager og oppdaterer kartet i mobilapplikasjonen

7.3.3.4 *Promises og async/await*

Promises og async funksjoner (se kap. 2.7) ble benyttet i situasjoner hvor det var kommunikasjon med API-et. Når data skulle lastes inn var det nødvendig å sende forespørselen til API-et og vente med videre fullføring av funksjonen til en respons var til stede. Promises og async/await er brukt blant annet for henting av prosjekt- og bruker data, samt når overføring av stillasdeler skal finne sted. I Kode-utdrag 43 Promises og async implementasjon for prosjektdata i mobilapplikasjonen benyttes et *ObservableObject* for å laste inn prosjektdata. En *boolean* benyttes for å informere brukeren om at dataen laster gjennom bruken av en *ProgressView*[24]. For hentingen av data benyttes en async funksjon *loadData* som laster inn dataen i *projects arrayen*. Når dette er fullført oppdateres *completion handleren* og returnerer *projects* til await kallet i Kode-utdrag 44.

```
class ProjectData: ObservableObject {
    /// Is data loading?
    @Published private var _isLoading: Bool = false

    /// Getter for the _isLoading
    var isLoading: Bool {
        get { return _isLoading}
    }

    /// Responsible for getting the data about projects from the API
    /// - Parameter completion: completion handler
    func loadData(completion:@escaping ([Project]) -> ()) async {
        _isLoading = true

        guard let url = URL(string:
"http://10.212.138.205:8080/stillastracking/v1/api/project?scaffolding=true") else {
            print("Invalid url...")
            return
        }
        /// Sends the request and gets the data
        URLSession.shared.dataTask(with: url) { [self] data, response, error in
            let projects = try! JSONDecoder().decode([Project].self, from: data!)
            DispatchQueue.main.async {
                completion(projects)
                self._isLoading = false
            }
        }.resume()
    }
}
```

Kode-utdrag 43 Promises og async implementasjon for prosjektdata i mobilapplikasjonen

[24] Mer om *ProgressView* https://developer.apple.com/documentation/swiftui/progressview

```
NavigationView {...}
        .task {
            /// On first time opening --> Load in project data
            if projects.isEmpty {
                await projectData.loadData { (projects) in
                    self.projects = projects
                }
            }
        }
```

Kode-utdrag 44 Kall av async funksjon for prosjektdata i mobilapplikasjonen

7.3.3.5 *Bindings og State*

State og *Bindings* operatorene som er *PropertyWrappers* benyttes for å sende data på tvers av *Views* hvor dataen kan endres i et annet *View*. Dette skjer ved at *SwiftUI* monitorer verdien til variabelen og endre denne og renderer denne på nytt på en effektiv måte. Dette har gjennomgående vært nødvendig for å løse applikasjonen og er særlig nødvendig i filtrering av prosjekt.

I Kode-utdrag 45 og Kode-utdrag 46 er eksempler på hvordan *Bindings* og *State* benyttes i mobilapplikasjonen. Variabler defineres med *@State* i *Parent-Viewen* og initialiseres med en verdi og variabler defineres med *@Binding* i *Child-Viewen* uten å initialiseres, men en datatype defineres. Disse to verdiene knyttes sammen ved at *@Binding* variablene tildeles *@State* variablene når *Child-Viewen* kalles på fra *Parent-Viewen* som i Kode-utdrag 45. På denne måten kan den overvåkede variabelen definert med *@State* passeres videre inn i nye *Views*, og endringer som skjer på variabelen i *Child-Viewen* oppdaterer *Parent-View* verdien.

```
struct FilterView: View {
    ...
    /// Selected filter start date and end date
    @State var selStartDate = Date()
    @State var selEndDate = Date()

    /// Returns whether a filter is active or not
    @State var periodFilterActive: Bool = false


    ...
    var body: some View {
        NavigationView {
            ...
                switch filterItem {
                case "Periode":
                    FilterProjectPeriod(selStartDateBind: $selStartDate, selEndDateBind: $selEndDate,
periodFilterActiveBind: $periodFilterActive)
            ...
        }
    }
    ...
}
```

Kode-utdrag 45 Bruken av @State i mobilapplikasjonen

```
struct FilterProjectPeriod: View {
    /// Selected start date and end date
    @Binding var selStartDateBind: Date
    @Binding var selEndDateBind: Date

    /// Tells if filter is activated or not
    @Binding var periodFilterActiveBind: Bool

    ...
}
```

Kode-utdrag 46 Bruken av @Binding i mobilapplikasjonen

7.3.3.6 *Caching*

Caching er en sentral del av en applikasjons livssyklus. *Caching* ble forsøkt implementert i applikasjonen, men ble ikke ferdigstilt og fungerer dermed ikke som planlagt. Alternativet som ble tatt i bruk for å gi en god brukeropplevelse var å utnytte *@State* variabler som lagrer dataen midlertidig imens appen er aktiv.

7.3.3.7 *Sikkerhet og error håndtering*

Error håndtering er håndtert ved å bruke standardverdier for «optional values» (se Kode-utdrag 47), ved å bruke *guard* for eksempelvis URL konstruksjon (se Kode-utdrag 48) og validere *TextField* input (se Kode-utdrag 49) ved at brukeren kun får tilgang på tall-tastaturet og lignende (se kap. 8.1).

```
scoreFrom = Int(value) ?? Int(sliderSizeMax + sliderSizeMin) / 2
```

Kode-utdrag 47 Eksempel på bruk av "optional values" i mobilapplikasjonen

```
guard let url = URL(string:
"http://10.212.138.205:8080/stillastracking/v1/api/project?scaffolding=true")
else {
    print("Invalid url...")
    return
}
```

Kode-utdrag 48 Eksempel på bruken av "guard" i mobilapplikasjonen

```
/// **NumbersOnly**
/// Validates textfield input to be Int
class NumbersOnly: ObservableObject {
    @Published var value = "" {
        didSet {
            let filtered = value.filter { $0.isNumber }

            if value != filtered {
                value = filtered
            }
        }
    }
}
```

Kode-utdrag 49 Validering av Int input i mobilapplikasjonen

7.3.3.8 *Navigering*

Applikasjonen er inndelt hierarkisk med de tre hovedsidene *Profil, Kart* og *Prosjekt øverst.* Applikasjonen er forsøkt inndelt slik at all funksjonalitet relatert til de ulike hovedkategoriene er å finne under disse. Stillas og stillasoverføring gjøres inne i de respektive prosjektene og historien for hver av stillastypene underlegges de ulike stillastypene.

7.3.4 Resultat

Sett opp mot MOSCOW modellen for mobilapplikasjonen (se Tabell 7) oppnådde gruppen punktene i *Must have* og *Should have*. Av punktene i *Could have* uteble «Oppdatere informasjon om prosjekt». Det er lagt opp til at applikasjonen kan vise historie for stilassdeler på prosjekter, men ettersom dette ikke er støttet av API-et enda er verdiene i applikasjonen hard-kodet for å illustrere konseptet (se kap. 10.2.2.2).

7.3.5 Diskusjon

I denne seksjonen vil de ulike delene nevnt ovenfor bli diskutert. Implementasjonen drøftes, og eventuelle forbedringer som burde og kan finne sted tas opp.

Det aller meste av kode er skrevet selv. Et fåtalls elementer er i sin helhet hentet fra tredjeparter (dette er dokumentert i kildekoden). Ellers har noen seksjoner og elementer tatt en del inspirasjon fra tredjeparter (dette er også dokumentert i kildekoden). Bemerk at det ikke er snakk om hele filer som er hentet eller inspirert fra tredjeparter, men at elementer som har en tilkobling til en tredjepart befinner seg i disse filene.
Filer som inneholder seksjoner og elementer med kode som er inspirert eller hentet fra en tredjepart er:

- *ScaffoldingItem.swift*
- *TransfereScaffoldingView.swift*
- *MapView.swift*
- *MapDisplay.swift*
- *MapViewModel.swift*
- *NavigationBarBottom.swift*
- *TextfieldModifiers.swift*
- *SignUpView.swift*
- *SignInView.swift*
- *AppViewModel.swift*

7.3.5.1 *Design*

Design prosessen startet så fort kravspesifikasjonen (se kap. 3.1) og MOSCOW modellen (se kap. 7.3.1) var på plass. Designprosessen er sentral i utvikling ettersom utvikling er tidkrevende og kostbart. For å få en oversikt over hvordan applikasjonen skulle se ut på de ulike sidene ble sketsjing tatt i bruk (se Figur 31 Sketsj av mobilapplikasjon design). Dermed kunne gruppen raskt få en visuell representasjon av applikasjonen. Videre ble *Figma* benyttet for høyfunksjonell prototyping ettersom det er et verktøy som er gratis, intuitivt å jobbe med, samt at gruppen hadde erfaring med verktøyet fra tidligere fag. En høyfunksjonell prototype er godt

egnet for å oppleve hvordan de ulike delene av systemet samhandler. Verktøyet egner seg derfor også godt til brukertesting.

Designprosessen begynte når løsningen fremdeles baserte seg på barellesystemer noe som gjør at *Figma* prototypen er designet for denne løsningen. Ettersom gruppen kom fram til at barelleløsningen ikke skulle videreføres måtte applikasjonen re-designes. Grunnet tidsbegrensninger ble det ikke prioritert å lage en ny høyfunksjonell prototype, noe som gjorde at kun en detaljert sketsj ble laget for de sidene som endret seg i andre iterasjon. Dette var ikke en gunstig gjennomføring, men gruppen følte at det var det beste alternativet gitt situasjonen.

7.3.5.2 *Model-View-ViewModel*

Når det gjaldt design arkitekturen for applikasjonen kunne gruppen velge mellom en MVC (Model-View-Controller) og MVVM (se kap. 2.8.1.2). I MVC er *View* og *Controller* ansvarlig for kalkulering av verdiene, samt å tildele verdien. I MVVM derimot er *View* og *Controller* kun ansvarlig for å tildele verdien, ikke kalkuleringen av verdien. Dermed abstraheres business logikken fra UI noe som gjør koden lettere å vedlikeholde og gjenbruke, samt tilrettelegger for unit testing for både *ViewModel* og *Model* lagene uten å måtte referere til *Views*.

MVVM er forsøkt innført der det er mulig. Det skal derimot sies at det er et klart forbedringspotensial der hvor blant annet *projects* arrayen kan trekkes ut i *ObservableObjects* som kalles på og funksjonen *transfereScaffoldingUnit* i filen *TransfereScaffoldingView.swift* burde vært abstrahert ut fra *Viewet* i en egen klasse sånn som det er gjort for prosjektdata og brukerdata.

7.3.5.3 *Innlogging og registrering*

Når det gjaldt innlogging og registrering av nye brukere så gruppen på alternativer for å «out-source» innlogging ved hjelp av *Oauth* eller *Firebase*, samt å lage en egen innloggingsfunksjonalitet fra bunnen av. Valget falt på *Firebase* ettersom de tilbyr det vi er ute etter på en sikker og intuitiv måte. Med denne løsningen må brukeren registreres to steder når ny bruker registreres. (se kap. 8.1.2)

Koden vedrørende innloggingen og registreringen av nye brukere ble i stor grad inspirert fra blant annet *Firebase Authentication* dokumentasjonen (henvisninger til brukte kilder er gjort i *AppViewModel.swift, SignInView.swift* og *SignUpView.swift*). Registreringen av en bruker er ikke fullført i mobilløsningen. Per nå registreres kun brukeren i *Firebase Authentication*. Støtten er lagt til rette for, og det er kun front-end som behøver justeringer.

7.3.5.4 *Kart*

Gruppen vurderte å *bruke MapBox, Google Maps* og *Apple MapKit* i applikasjonen for kartløsningen. Her falt valget på *Apples MapKit* ettersom det er god integrasjon for rammeverket i utviklingen av Swift applikasjoner, de tillater annotering og reiseruter og tid beregning, det imøtekommer brukervennlighet med hensyn på «Patterns and learnability» (se kap. 7.1.1.7) ettersom brukeren av en iOS applikasjon trolig er kjent med *Kart* appen tilbudt av *Apple*.

Innledningsvis ble kartet implementert med annoteringer for prosjekter. Dette fungerte problemfritt når prosjektdataen ble lest fra fil. Problemet oppsto når dataen hentes fra API-et. Da opprettes ikke annoteringene med prosjektdataen. Gruppen tror at løsningen er implementert riktig og er derfor usikker på hva som forårsaker problemet, men tror at dette kan ha noe med rekkefølgen på når dataen lastes inn versus når annoteringene produseres. Utover dette planla gruppen også at kartet skulle være søkbart, men dette ble ikke prioritert implementert.

7.3.5.5 *Promises og async/await*

For henting av data fra, og sending av data til API-et ble Promises og async/await funksjonalitet benyttet. Dette ble gjort ettersom det er beste praksis å benytte disse for henting av data fra et API og de er svært intuitive å benytte i Swift.

7.3.5.6 *Bindings og State*

Bruken av *Binding* og *State* variabler var nødvendig for å overvåke variabler som ble passert mellom ulike *Views*. Dette har blitt benyttet gjennomgående gjennom applikasjonen ettersom mye forhåndsvisning av data skjer, og endringer av data bør gjøres så fort en endring oppstår.

I noen tilfeller er *Binding* og *State* variabler brukt noe unødvendig mye. Eksempel på dette er at variabler som ikke behøves å sendes videre, sendes videre til et *Child-View*. Dette kan løses ved at objekter og variabler defineres et annet sted for så å bli kalt på via en funksjon eller som et *ObservableObject*.

Enkelte steder er *Views* brukt kun som en «bro» for en variabel fra ett *View* til et annet uten at variabelen blir samhandlet med av brukeren. Noen av disse «broene» er definert med en ny *State* variabel som tildeles verdien av *Binding* variabelen før denne sendes videre over «broen». Dette kunne vært endret for å forenkle koden betydelig ved å bare sende *Bindings* variabelen videre uten å re-definere den med en ny *State*.

7.3.5.7 *Sikkerhet og error håndtering*

Vi har forsøkt å implementere error-håndtering gjennom flere tiltak. Det første tiltaket var å redusere muligheten for en bruker til å gjøre feil. Dette gjøres blant annet gjennom å kun tillate brukeren å benytte talltastaturet når den forventede verdien er et tall, gjennom bruken av Pickers[25].

Et annet tiltak er å validere data som brukeren passerer inn, samt data fra API-et ved å forsøke å oversette den til en datatype og tilby en standard verdi om oversettelsen feiler. Utover dette valideres det meste av innsendt data også i API-et (se kap. 8.1).

Innlogging og brukerregistrering skjer ved hjelp av *Firebase Authentication*. Dette gjøres fordi *Firebase* tilbyr funksjonaliteten vi behøver og behandler brukerdataen på en trygg måte (se kap. 8.1.2.1).

[25] Se for mer informasjon om pickers https://developer.apple.com/documentation/swiftui/picker

Error håndtering og bruken av try/catch burde vært ytterligere utviklet og benyttet i enda flere tilfeller enn den gjør i nåværende versjon. For eksempel burde *URLSession.shared.dataTask* vært puttet inn i en try/catch i innlastingen av API data selv om den også sjekker en error variabel. På denne måten passer man på å fange alle error-er som oppstår, noe som blant annet forhindrer lekking av systemkritisk informasjon til eventuelle eksterne trussel aktører og er med på å gi brukeren tilstrekkelig informasjon om hva som gikk galt.

7.3.5.8 *Navigering*

Applikasjonen er inndelt i en hierarkisk arkitektur for navigering hvor funksjonalitet er forsøkt plassert logisk som underkategorier. Dette er med på å gi en bedre brukeropplevelse ettersom det som forventes å finnes ligger der man forventer det (se kap. 7.3.1).

8. Testing, sikkerhet og kvalitetssikring

Kapittelet vil gå i dybden på avgjørelser og refleksjoner rundt temaene testing (både kodetesting og faktisk testing av fysisk hardware), sikkerhet (sikkerhetstiltak gruppen har gjort og burde gjort), samt hvilke grep gruppen har tatt for å forsikre seg om at programvare er av en viss standard.

8.1 Sikkerhet

Gruppen ønsket å sette fokus på kjerne- og generelle sikkerhetskrav under utviklingen av systemet. Det ble derfor i oppstartsfasen undersøkt ulike sikkerhetsstandarder og metoder, gruppen kunne ta i bruk for å oppnå dette på best mulig måte. Mange av sikkerhetskravene oppnås innad i API-et ettersom det er her dataen blir behandlet.

8.1.1 Integritet og tilgjengelighet

Gruppen holdt et fokus på input validering for å sikre systemets integritet og tilgjengelighet (se kap. 7.2.3). I motsetning til en SQL-database som krever en struktur, vil en NoSQL-database kunne legge inn felter og verdier som i utgangspunktet ikke var ment å legges inn. Det er derfor viktig med validering av input data og datatyper. Validering av felter og datatyper vil sikre integriteten til systemet. Dersom feltene eller datatypen er forskjellig fra det systemet har definert, vil det oppstå en feil, noe som vil resultere i utilgjengelig data.

Atomiske operasjoner forsterker integriteten av data i systemet. Som nevnt i kap. 6.1.4, er ikke dette en funksjonalitet i NoSQL-databaser. Gruppen løste dette ved bruk av cloud functions. Dette sikrer at datatransaksjoner vil skje atomisk, der dette vil være nødvendig. Et eksempel er overføringer av stillasdeler, dersom en av transaksjonene feiler, vil ingen av de andre transaksjonene fullføres. Dermed sikrer man at det ikke oppstår ukonsekvent data i databasen.

8.1.2 Autorisering og autentisering

Autorisering og autentisering er viktig slik at dataen ikke kommer i feil hender. Teknologiene gruppen vurderte var Oauth, JWT, *Firebase Authentication*, API-nøkkel og lagring av brukernavn og passord i databasen. Gruppen begynte noe sent i utviklingsprosessen med å implementere autorisering og autentisering, noe som førte til at implementering av JWT ble vanskelig ettersom dette ville resultert i fullstendig omstrukturering av API-et.

For å sikre dataen, ville gruppen kombinere en API-nøkkel sammen med en innlogging metode i web- og mobilapplikasjonen. Dette sikrer at kun de med gyldig API-nøkkel får tilgang til API-et, samt at de med gyldig innlogging får tilgang til web- og mobilapplikasjonene.

Gruppen holdt sikkerhets standardene relativt like for både på mobil og web løsningen. Innloggingen på klientene ble gjort ved bruk av *Firebase Authentication*. Gruppen vurderte å implementere Oauth, *Firebase Authentication* eller en hjemmelaget innlogging. Oauth vil ikke gi oss muligheten til å koble opp brukeren med informasjonen som er lagret i databasen. Resultatet av dette ville bli at vi ikke hadde hatt mulighet til å autorisere brukeren med tanke på roller innad i systemet.

En enkel versjon der gruppen lagret passord i databasen, ville gjort det enkelt å knytte brukeren opp mot den andre lagrede dataen i systemet. Dersom gruppen skulle implementere dette, måtte passordet bli hashet, slik at ved en potensiell hack kunne ikke trussel aktøren fått direkte tilgang til passordet. Firebase Authentication ville gi løsningen økt sikkerhet. Dermed gikk gruppen for denne metoden, og passordet til brukeren kunne bli lagt inn og lagret på en sikker måte.

Brukeren registrerer epost og passord som blir registrert i *Firebase Authentication* før en autogenerert userID (UID) må hente ut denne bruker ID-en og registrere denne sammen med øvrig bruker informasjon til vår database. På denne måten kan de ulike brukerne få relevant informasjon og rettigheter for sine brukere og roller. Grunnen til at valget falt på å benytte et allerede eksisterende innloggings- og registreringssystem var at dette var et trygt og fungerende system med stor sikkerhet.

8.1.3 Ansvarlighet

Ansvarlighet blir til en viss grad oppnådd i API-et ved bruk av logging. Loggingen gjør at vedlikeholderen av systemet har mulighet til å finne ut hvilket kall som har blir kjørt i API-et, når det ble kjørt og hvor API-et potensielt har krasjet. Dette kan redusere nedetiden til systemet ved at vedlikeholderen vet hvor programmet må rettes opp. Ettersom API-et ville blitt sikret med en API-nøkkel, vil ikke auditing være hensiktsmessig. Derimot vil dette kunne bli implementert i web- og mobilløsningen.

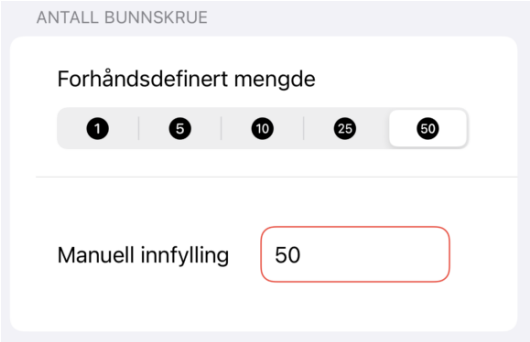
8.1.4 Økt-håndtering

Gruppen benytter seg av Firebase Authentication, som autentiserer brukeren med en token. Denne type token er delt i access token og en refresh token. Refresh token er til for å fornye token når den «går ut på dato». Gruppen har ikke satt noen regler på dette innad i firebase, og som et resultat vil denne token alltid bli fornyet. Dette resulterer i at web- og mobilapplikasjonen vil være sårbare mot en session hijacking.

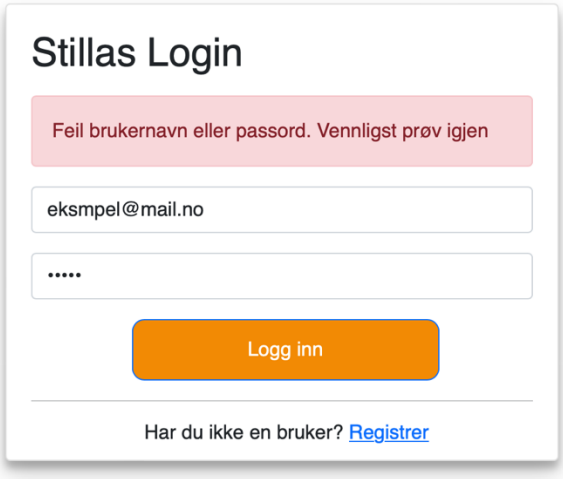
8.1.5 Error håndtering

Dersom det skal oppstå en error i systemet vil brukeren få en feilmelding. Feilmeldinger er en kilde til informasjonsutlevering direkte fra systemet. U-håndterte feil i systemet kan gi ut informasjon om systemets arkitektur design og konfigurasjoner. Gruppen har valgt å benytte seg av kortfattede feilmeldinger, for å unngå at teknisk informasjon blir returnert til brukeren med eventuelle statuskoder hvor hensiktsmessig (se Figur 46).

Ellers forsøkes det å håndtere error-er før de oppstår. Dette gjøres blant annet gjennom å deaktivere «Neste»/ «Bruk» knappen før brukeren har fylt inn all nødvendig informasjon og markere input feltet med rød ramme hvis brukeren forsøker å velge en utilgjengelig mengde stillas for overføring (se Figur 45 Tilbakemelding error stillasmengde i mobilapplikasjonen).



Figur 45 Tilbakemelding error stillasmengde i mobilapplikasjonen



Figur 46 Eksempel på en generisk feilmelding

Gruppen har håndtert feil ved bruk av try/catch blokker, ved eksekvering av asynkrone funksjoner.

```
const AddScaffold = async () => {
    try {
        setButtonPressed(true)
        await putModel(TRANSFER_SCAFFOLDING, JSON.stringify(move))
        await queryClient.resetQueries(["project", props.id])
    } catch (e) {
        setButtonPressed(false)
        if (e.text === "invalid body"){
            window.alert("500 Internal Server Error\nNoe gikk galt! Prøv
igjen senere")
        }else {
          window.alert("Advarsel: Kan ikke overføre antall stillasdeler")
        }
    }
}
```

Figur 47 Eksempel på bruken av try/catch i webapplikasjonen

Dersom eksekveringen av noen asynkrone funksjoner ikke blir fullført vil denne error-en bli håndtert i «catch» blokken, og brukeren vil få en egendefinert tilbakemelding, som ikke gir ut informasjon om systemet.

8.1.6 Konfigurasjon parametere

Gruppen har beskyttet deler av koden ved å hente ut konfigurasjons filer og nøkler, som er vesentlig for at programmet skal kunne kjøre. Dette er essensielle deler for at programmet skal fungere, det er derfor viktig at disse filene blir holdt unna trussel aktører.

8.2 API

Gruppen har for utvikling av API-et skrevet egne API-tester. Disse ble delt opp i forskjellige klasser som testet de forskjellige endepunktene og deres funksjonalitet. Dette ble gjort slik at gruppen enkelt kunne forsikre seg om at endepunktene fungerte som forventet uten å måtte manuelt teste hvert endepunkt.

```
dataBaseTestConnection()
handler := http.HandlerFunc(endpoints.ScaffoldingRequest)

//Add list of Scaffoldingparts which sends a post request and creates 12
scaffolding parts
t.Run("Add list of Scaffoldingparts", func(t *testing.T) {
   apitest.New().
      HandlerFunc(handler).
      Post("/stillastracking/v1/api/unit/").
      Body(`ENDPOINT BODY`).
      Expect(t).
      Body("\"12 new scaffolding units added to the system the following
units were added:ECPECTED BODY HERE").
      Status(http.StatusOK).
      End()
})
```

Kode-utdrag 50 Eksempel på en av API-Testene for scaffolding endepunktet

Kode-utdrag 50 illustrerer hvordan gruppen har valgt å strukturere API testene: Vi tar i bruk en egen test-database slik at produksjonsdatabasen ikke blir modifisert av testene. Deretter initialiseres det en ruter og en handler som håndterer testens forespørsel. Testen sender inn informasjon til endepunktet, før testen sjekker respons body og status kode. Med dette oppsettet kan gruppen enkelt sjekke om endepunktet oppfører seg som forventet.

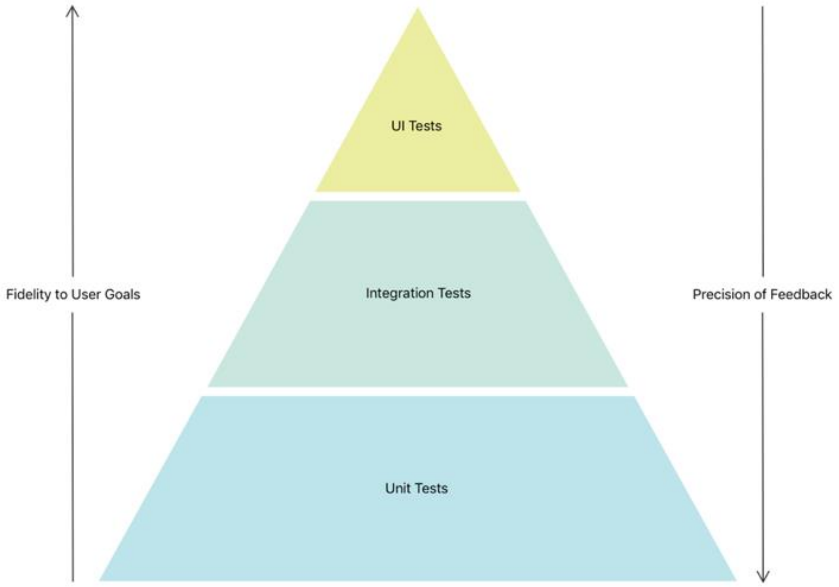
Alle tester som behandler dokumenter er strukturert på følgende måte: Testene legger inn dokumentene i databasen, deretter kjører en ny test som henter ut dokumentene basert på diverse filter, deretter oppdateres dokumentene i databasen før de til slutt slettes. Gruppen har implementert disse testene der de hadde tid, men det er derimot ikke implementert tester for alle endepunktene ved levering.

8.3 Front-end

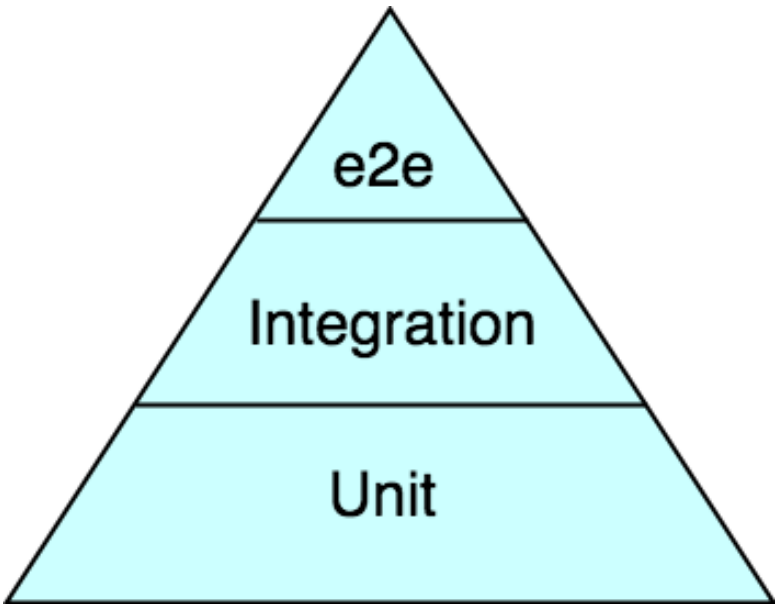
Mobilapplikasjonen og webapplikasjonen er kun testet for brukerinteraksjon (se Appendix H). I den videre utviklingen av systemene vil testing ha en sentral rolle for kvalitetssikring, ettersom at gruppen ikke fulgte test drevet utvikling. Det er mange tester som kan og bør gjennomføres.

Mobilapplikasjonen kan bli testet ved «Manuell testing – Bruk av enheten» (se Appendix H), «Manuell testing – Bruk av emulator» og «Automatisk testing». På samme måte vil

webapplikasjonen kunne bli testet ved «Unit testing», «End-To-End testing», «Komponent testing»



Figur 48 Testing pyramide for Swift. Bilde tatt fra: https://developer.apple.com/documentation/xcode/testing-your-apps-in-xcode



Figur 49 Test pyramide for webapplikasjon testing. Bilde hentet fra https://medium.com/expedia-group-tech/integration-testing-in-react-21f92a55a894

8.3.1 Unit Test

Gruppen anser det som viktig og nyttig å gjennomføre Unit tester av systemfunksjonalitet. Dette er særlig viktig, men også enkelt å gjennomføre når applikasjonen er bygd opp på MVVM design mønsteret (se kap. 7.3.2.1). Her bør da det logiske laget testes (53).

I Kode-utdrag 51 kan man se eksempel på hvordan Unit testing utføres i Swift og eksempelet viser hvordan man kan teste at en stillastype er av en bestemt type.

```
class HelloWorldTests: XCTestCase {
   func testScaffoldingTypeReturnsScaffoldingType() {
      XCTAssertEqual(scaffolding(scaffoldingType: "Bunnskrue"), "Type is Bunnskrue!")
   }
}

func scaffolding(scaffoldingType: String?) -> String {
   "Type is \(scaffoldingType.map { $0 } ?? "none")!"
}
```

Kode-utdrag 51 Unit testing i mobilapplikasjonen

I React ønsker vi å teste funksjonaliteten på hver enkelt komponent. Disse testene er til for å identifisere error-er i koden. Eksempel på test i React (se Kode-utdrag 52):

```
test('Valid user Email', () => {
    render(<contactInformation />);

    const emailInput = screen.getByTestId("email");
    userEvent.type(emailInput, "mbstillas@gmail.com");

    expect(emailInput.toHaveValue("mbstillas@gmail.com"));
    expect(screen.queryByTestId("Alert")).not.toBeInTheDocument();
});
```

Kode-utdrag 52 Unit testing i webapplikasjonen

I koden over blir email inputfeltet testet, i addScaffolding.js filen. Her blir det sjekket om riktig format blir plassert inne i feltet. Dersom dette stemmer, skal det ikke vises en Alert. Ender dette i suksess, vet vi at sjekken for email fungerer som den skal.

8.3.2 Integrasjonstest

Implementasjonen av integrasjonstesting anser vi som viktig for å sjekke at alle de ulike delene av systemet fungerer som forventet når satt i et system og de skal samhandle med hverandre. I motsetning til Unit testing, testes nå også systemets interaksjon med de andre delene utenfor den spesifikke «Unit-en». Dette brukes når objekter og data og ikke bare er «mock-ede» instanser, noe som muliggjør trigging av error-er og bugs som først oppstår så fort en faktisk interaksjon foregår (54).

```
describe('App', () => {
    it('Search for project', () => {
        render(<Project />);

        const searchInput = screen.getByTestId("searchProjectFilter");
        userEvent.type(searchInput, "CCGjøvik");

expect(screen.getByPlaceholderText('projectName').value).toEqual('CCGjøvi
k');
    });
});
```

Kode-utdrag 53 Integrasjonstest i webapplikasjonen

I koden over blir det test hvordan Prosjekt siden, snakker med API-et, samt filtreringsfunksjonen blir testet. Dette blir gjort ved å hente ut enkelt komponenter fra siden. Disse komponentene vil bli påført en brukerinteraksjon. I eksempelet over er det søkefeltet i prosjekter, som foretar et søk. Det vil da bli sjekket om informasjonskortet som blir værende på prosjekt siden inneholder prosjekt navnet CCGjøvik. Dersom denne er ender i suksess, blir det bekreftet at søkefeltet fungerer, samt at dataen fra API-et blir behandlet på riktig måte.

8.3.3 End-To-End

End-to-end tester er en test metode som tester applikasjonen sin arbeidsflyt fra start til slutt. Slike tester er designet for å gjenskape bruker senarioer slik at systemet kan bli validert for integritet. Testen vil gå igjennom de meste operasjonene applikasjonen har å tilby, for å se hvordan den samarbeider med maskinvare, nettverk, databaser, API-er og mer.

For webapplikasjonen ville en end-to-end testing se slik ut:

a. Skrive inn url til innlogging siden
b. Logge inn med gyldig bruker
c. Åpne et prosjekt
d. Overføre stillasdeler fra lager til åpnet prosjekt
e. Logge ut av applikasjonen

For å foreta en slik test må man utnytte seg av tredjeparts biblioteker som Cypress[26]. En slik test vil bekrefte a bruker grensesnittet fungerer, samt at alt back-end virker som det skal. På denne måten kan vi sikre kvaliteten på webapplikasjonen (55).

8.3.4 UI-Testing

UI-tester kjører ikke appens kode, men bruker appens brukergrensesnitt kontroller som en vanlig bruker ville benyttet dem for å se om en handling (i likhet med oppgaver/aktiviterer i brukertester) er gjennomførbar eller ikke (53).

I Kode-utdrag 54 er to ulike UI-tester demonstrert. Den første testen *testLoginButton* sjekker om knappen «Login» eksisterer og den siste testen *testTransfereScaffoldingAppearance* sjekker hvordan systemet reagerer på et trykk på «TransfereScaffolding» knappen.

```
func testLoginButton() throws {
    let app = XCUIApplication()
    app.launch()

    let login = app.buttons["Login"]

    XCTAssert(login.exists)
}

func testTransfereScaffoldingAppearance() throws {
    app.buttons["TransfereScaffolding"].tap()
}
```

[26] https://www.cypress.io

Kode-utdrag 54 UI-tester i mobilapplikasjonen (Inspirert fra: https://www.appcoda.com/ui-testing-swiftui-xctest/)

8.4 Sporing

Sporingsenhetene har blitt testet på tre forskjellige lokasjoner, et klasserom inne på NTNU Gjøvik for å se hvordan BLE signalet ble påvirket av forskjellige typer interferens samtidig (betong, stål og elektronikk). Parkeringsplassen på NTNU Gjøvik for å sjekke maksimalrekkevidde med fri sikt og til slutt lagerplassen hos oppdragsgiver på Hamar for å kunne teste enhetens rekkevidde i en realistisk setting på stillas.

8.4.1 A155.4 Klasserom på NTNU Gjøvik

Første rekkeviddetest ble gjort på et alminnelig klasserom på NTNU Gjøvik, dette var hovedsakelig for å sjekke rekkevidden på enhetene i forhold til mange forskjellige typer interferens. Syv IBS05 enheter ble plassert på rundt campus (se Figur 50). Kun de to nærmeste enhetene ble plukket opp av Gateway-en, som var plassert i et klasserom. Den lengste registrerte rekkevidden i dette miljøet var 30 meter. Betongvegger, samt elektronikk som befinner seg på NTNU, kan være faktorer som gjør det vanskelig for Gateway-en å finne enhetene.

8.4.2 Parkeringsplass på NTNU Gjøvik

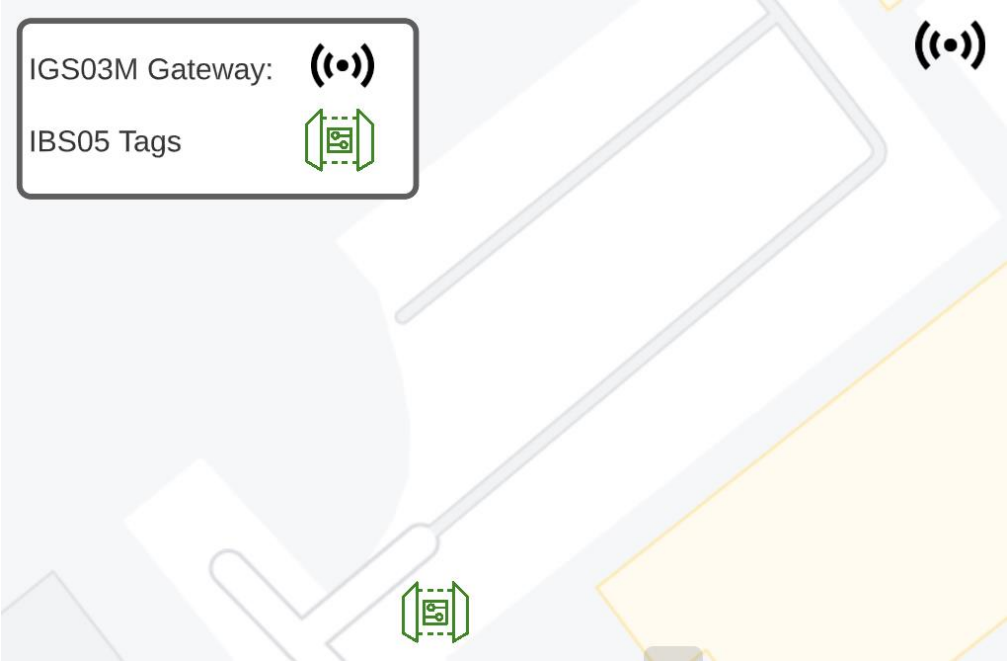
Figur 50 Plassering av tags NTNU Gjøvik

Figur 51 Test oppsett for rekkeviddetesting på parkeringsplass

Rekkevidden ble testet på parkeringsplassen på NTNU Gjøvik, for å få et bilde av enhetens rekkevidde ved optimale forhold. Denne lokasjonen hadde lite interferens, da eneste forstyrringer var elektronikken i bilene.

Gateway enheten som vist på Figur 51 og Figur 52 ble plassert over parkeringsplassen for å oppnå maksimal rekkevidde. Maksimal rekkevidde før enheten mistet forbindelsen med

Gateway ble målt til 120 meter. For å få data på hvordan rekkevidden bli påvirket av obstruksjoner plasserte gruppen IBS05 enheten på innsiden av et skilt laget av stål og plasserte skiltet bak en aluminiums-vegg. Med dette oppsettet ble det registrert en maksimal rekkevidde på 15 meter.



Figur 52 Oppsett rekkeviddetesting parkeringsplass

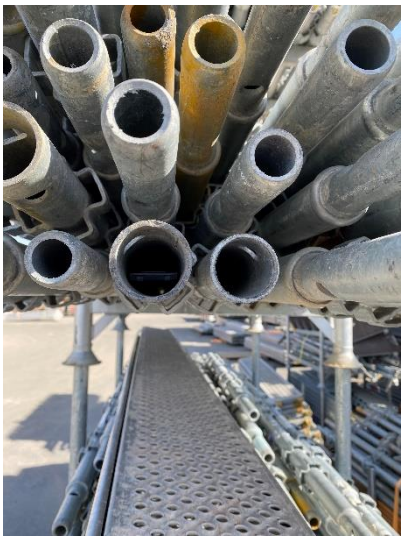
8.4.3 Lagerplass hos MB-Stillas

Gruppen dro mot slutten av bacheloroppgaven på et nytt besøk til Oppdragsgiver (*MB-Stillas*) for å ha et fremlegg angående status på prosjektet, samt for å teste rekkevidden på løsningen i en mer realistisk setting.

Figur 53 Plassering av enheter under testing hos MB-Stillas

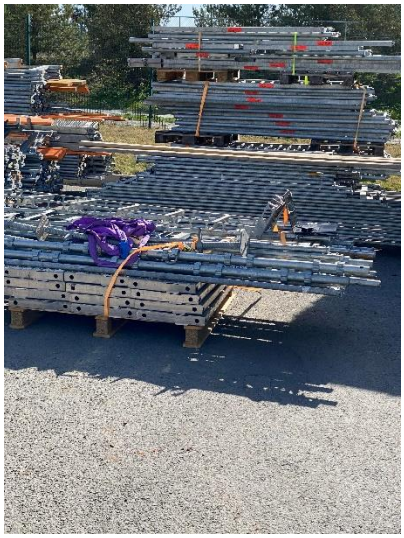
På oppdragsgivers lagerplass ble enhetene plassert rundt/inne i diverse stillasdeler: Inne i et stålspir (se Figur 54), inne i et aluminiumspir (se Figur 5655) og inne i en pall med stillaslemmer (se Figur 55). Dette var i all hovedsak for å sjekke rekkevidden til enhetene da de var omringet av ekstremt mye stillas.

Det ble registrert en maksimalrekkevidde på 70 meter for både enheten inne i stålspiret og enheten under stillaslemmene (se Figur 53). Aluminium hadde derimot en stor effekt på rekkevidden til IBS05 enheten, 15 meter ble registrert på det meste, men signalet var meget ustabilt. Gruppen mener derimot at dette er noe ekstreme forhold da et oppsatt stillas vil skape mindre interferens enn lagret stillas.



Figur 56 Enhet plassert inne i aluminiumspir

Figur 54 Enhet plassert inne i stålspir



Figur 55 Enhet plassert inne i stillaslem

9. Installasjon og realisering

Dette kapitelet vil ta for seg prosessen videre av det utviklede systemet. Her snakker vi om hva som må forbedres på de forskjellige systemene for at de skal bli optimalisert for større datamengder og bruk. Hvordan oppdragsgiver kan deployere systemene vil også bli diskutert i dette kapittelet.

9.1 Database

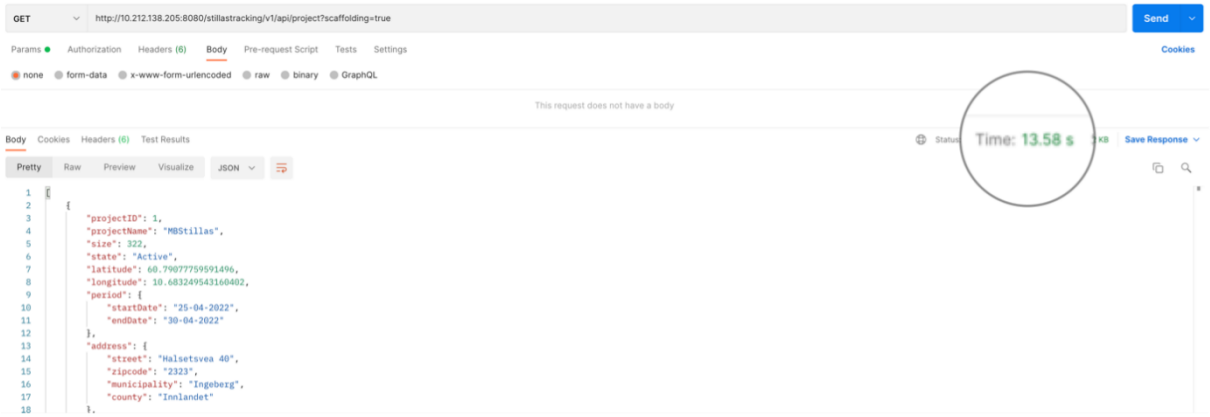
En realisering av oppgaven vil bety en enorm vertikal skalering av data. Firebase har ikke en kvote på antall dokumenter eller kolleksjoner databasen tillater. Derimot er det en grense på størrelsen av data i et dokument, som er satt til 1 MiB. Kvoten på antall lesinger, oppdateringer og slettinger i databasen kan oppdragsgiver justere selv, med påkostninger.

Dersom vi ser for oss et tidsrom der *MBStillas* har 500 prosjekter aktive, med 20 aktive brukere, og alle disse brukerne skulle ha aksessert web- og mobilløsningen 2 ganger om dagen, ville vi ha brukt 20 000 lesinger fra systemet, kun for å hente ut prosjektene. I tillegg må de også hente ut alle stillasdelene som er tilknyttet hvert prosjekt som tilsvarer 10 lesinger per prosjekt, noe som får totale lesinger opp i 30 000 lesinger. Web-løsningen oppdaterer alt av data hvert 5. minutt, noe som vil si at dersom alle 20 brukerne er på web-løsningen i litt over 10 minutter, vil alle lesingene dobles til 60 000. Systemet har dermed oversteget antall gratis lesinger, som ligger på 50 000.

Bruksmønsteret til brukeren er vanskelig å forutsi, ettersom faktorer som antall aktive enheter, brukere, og nødvendighet til å aksessere systemet er vanskelig å forutse. Noe av grunnen til at vi valgte å benytte oss av en NoSQL database var den lave kostnaden. Dette blir derfor ikke et stort problem for oppdragsgiver. Videre arbeid på API-et vil være å redusere antall lesinger, noe som resulterer i lavere bruksprис.

9.2 API

Gjennom bruken av API-et, har det blitt observert lang lastetid av data, spesielt prosjekter med stillasdeler. Dette kan skyldes kvaliteten og logikken til koden. En test ved henting av 13 prosjekter med stillasdeler, ble lastetiden målt oppimot 13 sekunder (se Figur 57 Bilde som viser responstiden fra Postman).



Figur 57 Bilde som viser responstiden fra Postman

Ved en realisering av oppgaven, er nødvendige optimaliseringer ved koden og databasen nødvendig for å få ned hentetiden av dataen. Dette for å unngå tidsavbrudd ved henting, samt å tilfredsstille brukeren. Deployering av API-et må skje på en server som er tilgjengelig for en eventuell utvikler. Spesifikasjonene til serveren vil avgjøre hastigheten på API-et, men til en liten grad. Gruppen har deployert API-et på skolens servertjeneste «OpenStack» og har brukt dette for å enkelt kunne integrere API-et med front-end tjenestene. Ved deployering for oppdragsgiver ville gruppen tatt i bruk tjenester som *Google Cloud Platform* eller *AWS*.

9.3 Front-End

Delkapittelet tar for seg eventuell deployering og installasjon av mobil og web-applikasjonene utviklet under oppgaven.

9.3.1 Web

Web applikasjonen er avhengig av API og database. En horisontal skalering av systemet vil ikke påvirke applikasjonen til en stor grad. Ventetiden vil gå betydelig opp ved henting av data fra API. En mulig løsning vil være å sette en grense på antall hentinger fra API-et, men dette gjør at søkefunksjonen ikke vil kunne søke gjennom alle prosjekter i databasen. En omstrukturering av hvordan dataen blir anvendt i web-løsningen kan også reduser denne tiden, ved at applikasjonen eksempelvis ikke må hente alle prosjekter med alle stillasdeler.

Installasjon av webapplikasjonen krever først og fremst en DNS, «domain name system», dette vil gjøre det lettere for brukeren å ta i bruk nettsiden. Uten et domenenavn ville brukeren måtte ha tastet inn en IP-adressen, slik som nå blir gjort i API-et. Oppdragsgiver eier et domene navn allerede, «mbstillas.no», som kan brukes for en realisering av tjenesten (56).

Etter domenenavn er satt opp må applikasjonen bli lastet opp på en server. Dette kan skje fra forskjellige IaaS, slik som *OpenStack, Firebase, AWS* og *Azure*. Dersom nettsiden er satt opp på hoved serveren eller domenet, blir subdomenet brukt for internutvikling. Subdomenet kan brukes til testing av nettsiden før hoved lanseringen av nettsiden. Etter disse stegene har blitt gjennomført, kan websiden deployeres. Når nettsiden er oppe er det viktig å overvåke nettsiden for å forsikre at alt fungerer som det skal (57).

9.3.2 Mobil

Den leverte versjonen av mobil-applikasjonen er ikke deployert. Ettersom gruppen har valgt å utvikle en native-mobil-applikasjon kan ikke løsningen kjøres på en server. Appen må sendes til *Apple* for godkjenning før den kan lanseres på *App Store*. Dette er ikke noe gruppen fikk tid til å realisere. Et alternativ hadde vært å utvikle en web basert mobil applikasjon slik at vi kunne deployert og testet mobil applikasjonen ute i felten.

Siden applikasjonen er laget for å kjøre native på brukerens iPhone avhenger ytelse av hvilken modell brukeren har. Vi har programmert med hensyn på Apples iPhone 11/12/13 Pro Max modeller og mener at vi også burde ha tatt utgangspunkt i en eldre modell da vi er usikre på hvordan applikasjonen kjører på disse eldre enhetene. Applikasjonen mangler også en

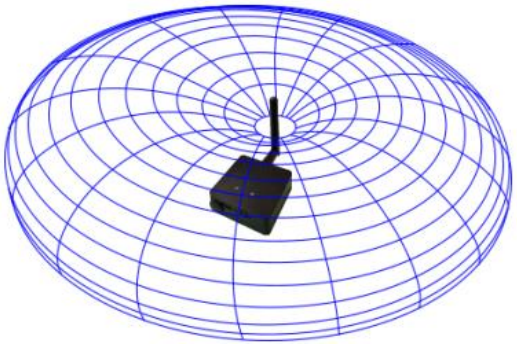
brukeravtale, samt et ikon. Dette er ting som bør implementeres om oppdragsgiver velger å ta i bruk applikasjonen.

9.4 Sporing

Kapittelet vil dekke en plan og anbefaling angående utplassering av Gateway-enhet og BLE-tags på byggeplass. Det vil ikke bli gått i dybden på fysikken rundt antennesignaler og radiobølger da gruppen ikke har et godt nok grunnlag til å argumentere for dette.

9.4.1 Gateway plassering

Gitt resultatene i kap. 5.3 og kap. 8.5 kan det være hensiktsmessig å benytte seg av flere Gateways på byggeplass. Testresultatene under kap. 8.5.3 vil bli brukt for vurdering av rekkevidde i dette kapittelet, da omgivelsene er mest likt en reel byggeplass. Disse testene ga en rekkevidde på 70 meter når enhetene var festet på stillas.



Figur 58 Signalmønster IGS03M BLE Gateway

Dette tilsier at hver Gateway vil dekke et areal på 15 393 kvadratmeter ($\pi * 70^2$) noe som er et stort areal i teorien, men gitt at signalet kan forstyrres av andre ting på byggeplass vil en dekning på liknende areal kun kunne forekomme under optimale forhold («tom» byggeplass uten maskineri og store bygg). Gruppen konkluderte også med at Gateway-enheten hadde en større rekkevidde om enheten ble plassert i høyden/over dekningsområdet. BLE antennen på enheten er omni-direksjonell (se Figur 58), noe som gjør plassering ganske enkelt og enheten skal fungere under de fleste omstendigheter, det er ønskelig unngå å plassere enheten i nærheten av aluminiums vegger da materialet hindrer radiosignalene.

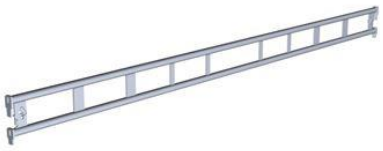
9.4.2 Beacon plassering

Plassering av IBS05 Taggene vil være den mer utfordrende delen av oppgaven. Enhetene må monteres på stillasdelene noe som er av varierende utfordringsgrad. Som vist i Figur 17 er antennen på enhetene meget liten og omni-direksjonell. Dermed må det bli tatt hensyn til interferens ved plassering av enhetene. Følgende forslag er tenkt ut for plassering av enhet på de forskjellige delene (for bilde av delene se Figur 1):

- **Spir Figur** 59**:**
  Rørdiameter: 48 millimeter
  Pris 618,00 kroner
  Plassering av enhet er noe vanskelig da delen er rørformet, plassering av enhet inne i spiret er mulig, men dette vil gå på bekostning av rekkevidde. Gruppen mener at montering på utsiden av spiret vil gi best resultat om mann ser bort ifra robusthet.

Figur 59 Spir

- **Lengdebjelke Figur** 60**:**
  Rørdiameter 48 millimeter
  Pris 564 kroner
  Den eneste reelle plasseringsmulighet vil være på en av støtteflatene på bjelken.

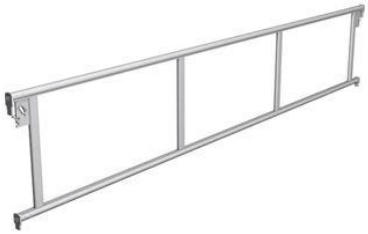


Figur 60 Lengdebjelke

- **Enrørsbjelke Figur** 61**:**
  Rørdiameter 48 millimeter
  Pris 285,00 kroner
  Delen er meget billig, dette gjør det mindre lønnsomt å montere en sporingsenhet på delen med test-enhetens pris. Om det derimot skulle plasseres en enhet på delen vil den måtte monteres på utsiden av røret slik som på et spir.

Figur 61 Enrørsbjelke

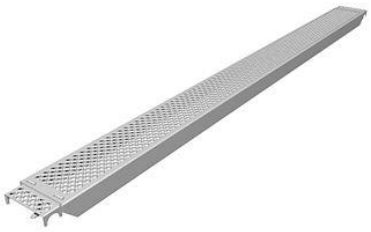
- **Rekkverk Figur** 62**:**
  Rørdiameter 28 millimeter
  Vekt 10,5 kilogram
  Pris 570,00 kroner
  Plassering av enhet må skje på støtteflate imellom de to hovedrørene på delen.



Figur 62 Rekkverk

- **Plank Figur** 63**:**
  Pris 813,00 kroner
  Monteringsplassen på planken vil være i hulrommet på undersiden av planken.

- **Lemmer** Feil! Fant ikke referansekilden.**:**
  Pris 510,00 kroner
  Delen ligner plank og vil derfor ha samme forslag angående montering



Figur 63 Plank

- **Trapp Figur** 65**:**
  Pris 4 670,00 kroner
  Delen er stor og har mange monteringsmuligheter. Montering under en av trappetrinnene vil være den beste løsningen for denne delen.

- **Gelender Figur** 67**:**
  Pris 1 639,00 kroner
  Plassering er problematisk på denne delen da støtterør og andre rør er ekstremt tynne. Montering må derimot skje på utsiden av en av disse rørene.

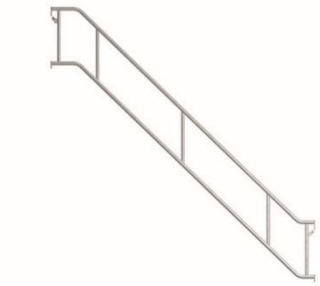
Figur 64 Lem

- **Bunnskrue Figur** 68**:**
  Pris 207,00 kroner
  Dette er den billigste delen å forholde seg til, den koster 50% av prisen for en BLE tag. Derfor kan det diskuteres om at det ikke vil være forsvarlig å plassere en tag på denne enheten på grunn av prisen samt at delen sjeldent fraktes fra byggeplass løst (ofte montert i spir eller andre deler da den ankommer lager)

Figur 65 Trapp

- **Diagonalstenger** Feil! Fant ikke referansekilden.**:**
  Rørdiameter 48 millimeter
  Pris 637,00 kroner
  Delen er noe lik som gelender og har derfor samme problem, montering må skje på utsiden av enheten til tross for at dette vil gå på bekostning av robusthet.

Figur 68 Bunnskrue



Figur 67 Gelender

Figur 66 Diagonalstang

10. Avslutning

Det avsluttende kapittelet i rapporten diskuterer løsningen gruppen har utviklet, resultatet og alternative måter gruppen vurderte å løse problemstillingen på. Oppgaven vil videre kritiseres og vurderes fra gruppens perspektiv før videre arbeid drøftes. Kapittelet tar for seg funksjonalitet gruppen ønsket å implementere. Avslutningsvis drøftes oppgaven som et teamarbeid og hvordan fremgangsmåten ble strukturert for å løse problemstillingen, før bacheloroppgaven oppsummeres og konkluderes.

10.1 Drøftinger

Kapittelet vil diskutere resultatene oppnådd opp imot de forskjellige målene definert i kap. 1.2. Det vil drøftes rundt hvorvidt de satte målene ble oppnådd. Det vil bli diskutert hvilke alternativer som ble vurdert, samt drøfting rundt valg som ble tatt vedrørende både design og utviklingsprosessen. Siden resultater rundt de forskjellige delene av oppgaven har blitt diskutert i de tidligere kapitlene vil vi ta for oss den helhetlige løsningen gruppen har utviklet.

10.1.1 Resultater

Kapittelet vil diskutere i hvilken grad resultatet oppnår de satte målene definert kap. 1.2.

10.1.1.1 *Resultater mot mål*

Målet oppdragsgiver hadde med oppgaven var å frigjøre ressurser hos bedriften. Gruppen har valgt å løse problemet ved å utvikle et system som sporer stillasdeler på byggeplass og lager. En ansatt hos oppdragsgiver kan ta i bruk gruppens mobil eller web-applikasjon, for å sjekke antallet registrerte stillasdeler på byggeprosjekter. Den ansatte får dermed raskt og enkelt oversikt over antall stillasdeler gruppert på type som befinner seg på de forskjellige byggeprosjektene. Om en stillasmontør har behov for ekstra deler kan vedkommende overføre stillasdeler fra nærmeste byggeprosjekt til sitt prosjekt uten at dette går på bekostning av tiden til andre ansatte.

En del av målet som ble satt var å utvikle en prototype som viser lønnsomheten og muligheten for sporing av individuelle stillasdeler per dags dato. Resultatet tilsier at oppdragsgiver ikke bør ta i bruk og skalere ut systemet gitt dagens marked og muligheter, dette er hovedsakelig på grunn av 4 faktorer:

- **Batteritid**: IBS05 Taggene som ble brukt til testing har en batteritid på 3-4 år. Det betyr at hvert fjerde år må oppdragsgiver bytte batterier på alle taggene noe som vil ta enormt med tid om systemet blir tatt i bruk på skalaen oppdragsgiver ønsker (oppdragsgiver har estimert at de har 2 millioner stillasdeler i sirkulasjon). For at teknologien skal kunne tas i bruk bør batteritiden ligge på minimum 10 år, gitt at en stillasdel har en levetid på cirka 10-15 år.
- **Pris**: Hver IBS05 tag koster 10 USD og hver Gateway koster 100 USD. Det estimeres at pris vil reduseres om oppdragsgiver kjøper tags i større volum, men det vil ikke være forsvarlig å kjøpe ønsket volum av tags for denne prisen.
- **Rekkevidde**: Taggene ble under testing plukket opp med en maksimal rekkevidde på 120 meter med fri sikt uten obstruksjoner. Med taggen festet på stillaset ble det registrert

en maksimalrekkevidde på 60 meter med noen obstruksjoner, dette er det relevante bruksområdet og resultatene ansees som gode. Det nevnes her, ettersom datasettene er noe vage og bør testes ytterligere før man med sikkerhet kan stille seg bak resultatene.

- **Skalering av programvare**: Den leverte versjonen av API-et og databasen er designet for å kunne håndtere store mengder data og en utskalering. Systemet er derimot ikke testet for dataprosessering i den skala oppdragsgiver skal benytte og hvorvidt system-designet tåler en slik belastning er usikkert. Responstiden i mobil- og webapplikasjonen er tidvis treg ettersom de nå henter prosjekt informasjon fra alle prosjekter med deres tilhørende stillasmengde. Dette burde optimaliseres ved å hente all prosjekt-informasjon og hente stillas-informasjonen først når et prosjekt aksesseres. På denne måten vil de ulike front-end løsningens responstid reduseres betydelig og løsningen burde først da tas i bruk.

Det er hovedsakelig disse fire faktorene som hindrer løsningen i å være lønnsom. Gruppen mener at ved optimalisering av systemet for store mengder data og en skreddersydd BLE tag med bedre levetid vil systemet være realiserbart for bedriften.

Oppdragsgiver har fått et innblikk i hvordan IoT markedet så ut på tidspunktet oppgaven ble skrevet. Dette har blitt gjort gjennom møter og dialog med oppdragsgiver, fysisk testing og diverse fremlegg hos bedriften. Ses det på de funksjonelle målene, har gruppen oppnådd følgende funksjonalitet fra User-Stories (se kap. 3.1). En bruker av systemet kan hente ut informasjon om, og behandle både stillasdeler, byggeprosjekter, Gateway-er og brukere. Front-end løsningen viser byggeprosjekter i kart og viser også en oversikt over forventede og registrerte mengder stillasdeler på byggeprosjekt.

Opprinnelig var ønsket å levere en løsning hvor oppdragsgiver kunne se spesifikk posisjon på hver enkelt stillasdel i kartet, men gitt at sporingsenhetene ikke tilbudte denne funksjonaliteten var ikke dette mulig. Implementasjon av dette ville også gjort håndtering av datapakker ressurskrevende og kostbart, gitt mengden stillasdeler oppdragsgiver har. Vi har utviklet API-et slik at det enkelt kan kobles på andre systemer oppdragsgiver tar i bruk. Dette illustreres ved at gruppen selv har koblet API-et opp mot to front-end-løsninger.

Ved oppstart av prosjektet ønsket gruppen å få bedre innsikt i bygge-bransjen, deres arbeidsmetoder, samt oppleve hvordan det er å arbeide med oppdragsgivere i denne bransjen. Dette mener gruppen selv de har fått til i stor grad. Læringsutbyttet har økt betydelig gjennom å ha møter med oppdragsgiver og *HAKI*. Vi syntes det har vært utfordrende og lærerikt å kommunisere, samt diskutere med disse rundt muligheter, begrensninger og kravspesifisering for løsningen.

En sentral del av læringsutbyttet for oppgaven har vært IoT og de forskjellige teknologiene tilgjengelig på markedet. Gruppen har lært mye av å undersøke disse teknologiene, men mener at en enda dypere undersøkelse av teknologiene ville ledet til en bedre løsning av sporingsdelen av oppgaven. Om valget hadde falt på å kun lage én front-end løsning og hatt et økt fokus på

sporingsenheten tror vi at læringsutbyttet rundt dette målet hadde vært bedre, men på bekostning av systemutviklings-læringsutbyttet.

Gruppen har også lært mer om API utvikling, databasemodellering, utvikling av dokumentdatabaser og særlig mobilprogrammering. Behovet for tilpasningsdyktighet og hurtig omstilling har vært avgjørende i prosessen. Dette gjelder både ved tilpasning til oppdragsgivers ønsker, men da særlig for å kunne forholde seg til selskapene gruppen var i kontakt med angående sporingsenhet.

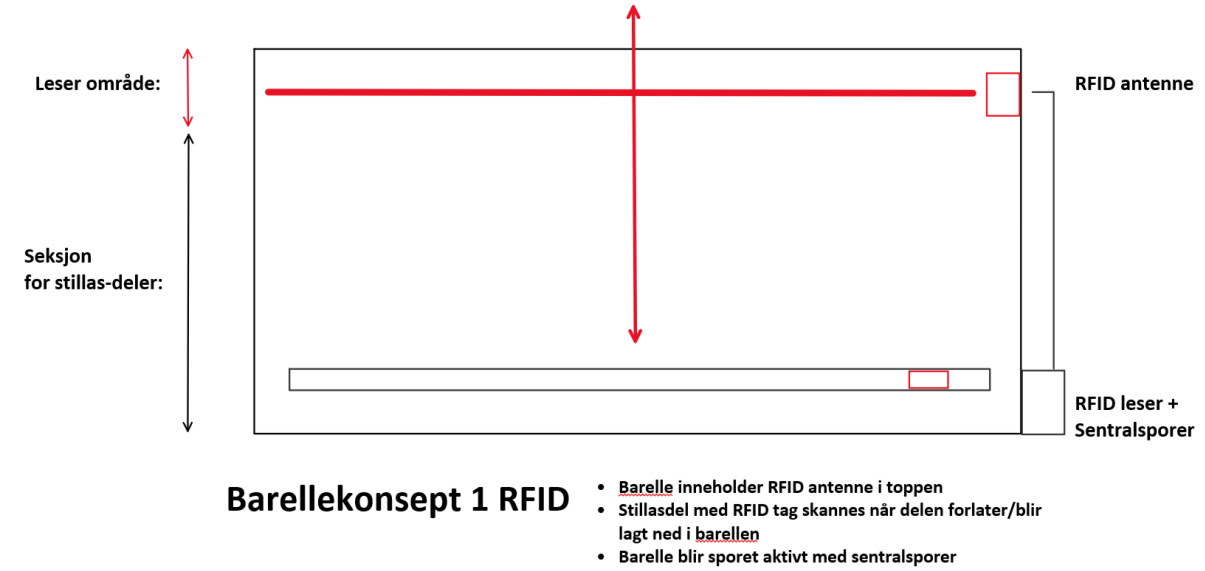
Det ble gjennomført ukentlige møter med oppdragsgiver, IoT-selskaper og en stillasprodusent. Disse møtene og mailkorrespondansene har belyst viktigheten av kommunikasjon for å gjennomføre et godt ingeniørfaglig arbeid. Gruppen mener at ved tydeligere kommunikasjon med både oppdragsgiver og eksterne selskaper (da spesielt med *Telia*) hadde det endelige produktet vært av bedre kvalitet ettersom tid og ressurser kunne vært distribuert annerledes. Viktigheten av en kostnadseffektiv løsning har kommet tydelig fram underveis i prosessen ettersom kostnadene øker hyppig for sporingsenheter, serverleie og databasekall i slike systemer. Dette er noe gruppen skulle ønske de hadde hatt et søkelys på tidligere i prosessen da den leverte versjonen av løsningen ikke er optimalisert for skalering av den grad oppdragsgiver har behov for.

10.1.1.2 *Valg gjort underveis*

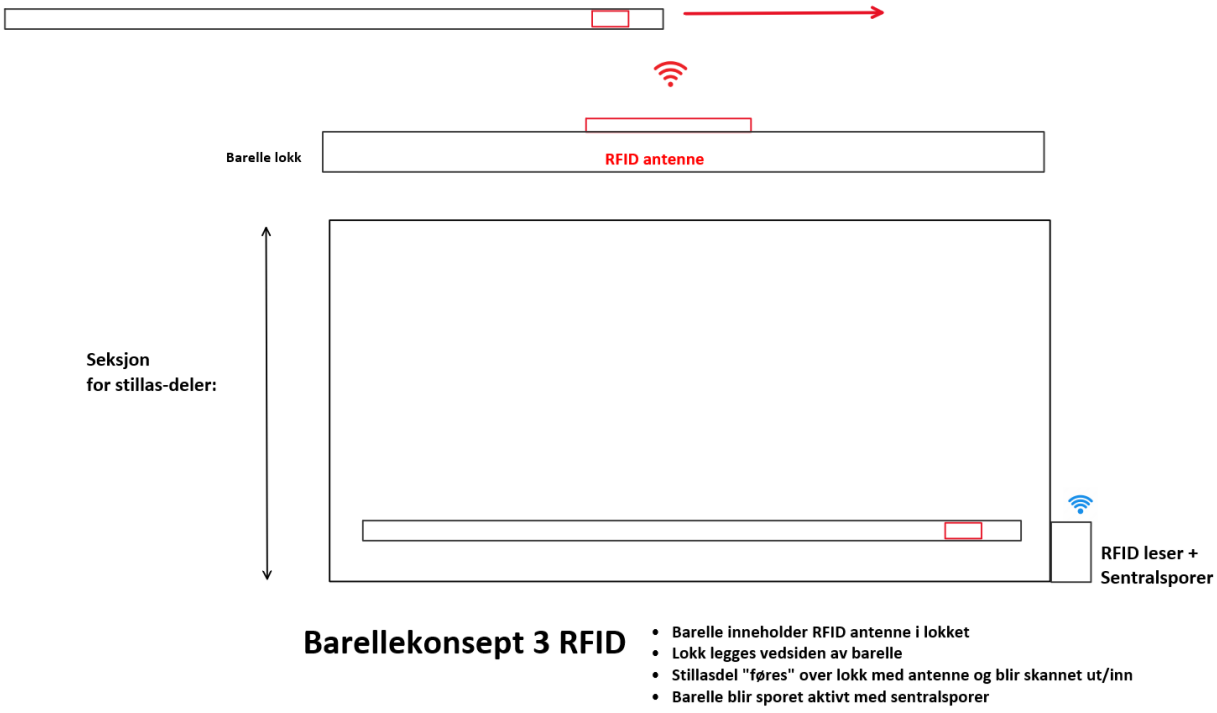
Oppgaven hadde krav om utvikling av en mobil applikasjon, samt et åpent API, men utover dette satte oppgaven ingen krav til hvordan oppgaven skulle løses. På grunn av dette ble det vurdert mange alternativer underveis i prosessen, da særlig ved valg av sporingsteknologi og den fysiske løsningen av sporingen (se kap. 5.2). Det ble tidlig i prosessen vurdert å ta i bruk Computer Vision. Vi ønsket å telle stillasdeler ved hjelp av et mobilkamera og bildeprosessering. Dette ville la oppdragsgiver telle stillasdeler på lager og på byggeplass. Ideen ble derimot skrinlagt da det ikke løste oppdragsgivers logistikkproblem. Det ville også vært en tidkrevende prosess å lære opp systemet, samt avgjørende å ha et stort datasett for stor nøyaktighet.

Oppdragsgiver ønsket mulighet for å identifisere stillasdeler på byggeplass. For å løse dette ønsket gruppen å ta i bruk QR-koder eller NFC teknologi. Dette ville la en stillasmontør identifisere sine stillasdeler på byggeplass ved å skanne en NFC-tag eller en QR-kode festet på stillasdelen.

Under diskusjon rundt det fysiske oppsettet av sporingsløsningen på byggeplass ble det som nevnt i kap. 5.2.2.2 diskutert om «bareller» skulle inkluderes i løsningen. Om løsningen skulle baseres på RFID trengte gruppen en måte å skanne stillasdeler på byggeplass på. Det ble derfor skissert et par ideer på hvordan dette kunne gjøres (se Figur 69 og Figur 70). Originalt var planen å gå videre med dette konseptet og designet på både API, mobil og web applikasjon ble designet med hensyn på «bareller». Ideen ble skrinlagt ettersom gruppen valgte å basere løsningen på BLE istedenfor RFID og vi måtte derfor re-designe begge front-end-løsningene og API endepunktene.



Figur 69 Skisse for planlagt løsning av barelle med rfid



Figur 70 Barelleskisse for RFID med leser i lokk

Under design av mobilapplikasjonen ble det diskutert rundt hvilket operativsystem applikasjonen skulle utvikles for. Under oppstart var ønsket å utvikle for både iOS og Android. Etter diskusjon internt og med oppdragsgiver ble det derimot avgjort at appen skulle utvikles for iOS (se kap. 7.3).

10.2 Videre arbeid

Kapittelet vil ta for seg de ulike delene av løsningen, hva som bør/må utbedres.

10.2.1 Back-end

Både database og API har behov for optimalisering. Som diskutert i kap. 6 og kap. 10.1.1 er ikke disse systemene klare for skalering. For at dette skal kunne skje må API-ets forespørsel håndtering forbedres og cloud-functions (referanser og linking av objekter) bør tas i bruk i databasen, da særlig for stillasdeler. Det er også som diskutert i kap. 6.2.4 mye duplikat kode i API-et som bør bli abstrahert. Dette vil gjøre koden enklere å vedlikeholde, samt å lese. API-et og databasen mangler funksjonalitet for å hente ut og lagre informasjon om historien til stillasdeler på prosjekt.

Videre bør også implementering av en docker-compose fil skje. Dette er påbegynt, men ikke ferdig. Det ble også påbegynt arbeid på en MQTT Broker som burde implementeres i et eventuelt videre arbeid. Dette vil forenkle prosessen rundt utskalering av systemet med flere Gateway-er på grunn av MQTT Protokollen og dets publish/subscribe system.

Fremtidige arbeidsoppgaver kan være å integrere oppdragsgivers ordrebehandlingssystem for å kunne automatisk legge inn byggeprosjekter og knytte disse til innkommende bestillinger. Sammenslåing av ulike systemer og API-er for eksempelvis ulikt utstyr til ett og samme system for en enkel brukeropplevelse. Behandling av en plukkliste for å knytte stillasdeler til et byggeprosjekt for å fjerne mye manuelt arbeid i systemet.

10.2.2 Front-end

Sikkerheten til systemet må oppjusteres da dataintegriteten er en hovedfaktor. Sikkerhet med tanke på innlogging, har blitt implementert. På en annen side, vil denne funksjonaliteten trenge litt modifikasjon ettersom token-en til brukeren blir kontinuerlig oppdatert. Dermed er systemet eksponert for økt-kapring.

Alt som skjer i systemet bør bli logget, da dette er vesentlig for å finne fatale feil i systemet. Per nå, har gruppen ikke implementert dette, men ser det som hensiktsmessig da systemet er omfattende med mange aktive brukere. Auditing, vil også være viktig, for å se hvem som har foretatt endringer i systemet. Her vil også autorisasjon komme inn i bildet, da oppdragsgiver kan velge hvem som skal ha de forskjellige rettighetene.

Funksjonaliteten i webapplikasjonen er så godt som fullført i henhold til MOSCOW-tabellen som gruppen satt opp i kap. 7.2.1. Funksjonaliteten for å redigere et prosjekt og autorisering av brukere er gjenstående. I tillegg kan det bli behov for endringer, slik at webapplikasjonen blir optimalisert for bruksflyten til oppdragsgiver. Systemet er per nå avhengig av store mengder data fra databasen. Responstiden er derfor lang, og må optimaliseres for videre arbeid. Webapplikasjonen er kun optimalisert for fullskjermsvisning. Applikasjonen er ikke tilrettelagt for bruk i et vindu som er eksempelvis 50% størrelse. Datahentingen har nå ikke et tidsavbrudd. Resultatet av dette vil være et endeløst forsøk på henting av data. For en realisering burde et tidsavbrudd blitt implementert.

Ettersom tap av utstyr er en viktig del av oppgaven, var det ønskelig å kunne legge inn historie på hver stillasdel også i web løsningen. Mobil løsningen har tilrettelagt for dette så fort API-et legger til støtte for det. Dette vil gjøre det lettere for oppdragsgiver å se når og hvor stillasdelene sist ble registrert, noe som vil være en vesentlig del av logistikkarbeidet.

Mobilapplikasjonen har integrert det meste som blir etterspurt i MOSCOW modellen. Optimalisering er derimot nødvendig, og enkelte aspekter mangler. Støtte for å legge til, oppdatere og slette prosjekter kan og bør bli vurdert utviklet. Å vise prosjekter i kart når dataen hentes fra API har feil som ikke gjør dette mulig. Registrering av nye brukere i vårt databasesystem må ferdigstilles. Caching må implementeres. Dette kan gjøres ved å ta i bruk *@AppStorage* og *@SceneStorage* «property wrapper-ne»[27]. Implementering av Unit-, Integrasjons- og UI-testing må finne sted. Utbytting av piksler til prosentvise størrelser for knapper etc., er nødvendig for optimal støtte på andre enheter enn iPhone Max modellene.

10.2.3 Sporing

Her mener gruppen det er potensiale for forbedring. Vi har basert løsningen på riktig teknologi, men vi mener at det finnes en bedre egnet Gateway og tag for løsningen. En bacheloroppgave sentrert rundt kun sporingsenhetene der en gruppe får i oppgave å designe en robust BLE tag med god batteritid ville vært meget verdifull for bedriften og hadde forbedret løsningen betraktelig. Det kan også undersøkes om enheten kunne festes på stillaset under produksjon.

10.3 Evaluering av gruppas arbeid

Kapittelet vil omhandle arbeidet gjort gjennom bacheloroppgaven, det vil diskuteres rundt oppsettet og planlegging av oppgaven, om arbeidsmengden har vært god og om den valgte metodikken var korrekt for oppgaven. Avslutningsvis vil det drøftes rundt arbeidsfordeling og hvordan gruppen syntes det har vært å jobbe med et prosjekt av så stor skala og tidsrom.

10.3.1 Organisering

Under utviklingsprosessen ble SCRUM metodikken benyttet (se kap. 1.5). Denne prosessen har fungert bra og vi mener at en agil utviklingsmetode var et godt valg for oppgaven. SCRUM gjorde at tilpasning til uforutsette komplikasjoner, samt innspill fra oppdragsgiver var enkelt og krevde ingen store omveltninger i prosessen.

SCRUM metodikken burde derimot blitt fulgt mer slavisk, da SCRUM oppsummering og planleggingsmøter som ble avholdt utover i utviklingsprosessen ble mer ustrukturert og uformelle. Dette var hovedsakelig ettersom alle gruppemedlemmer hadde ting de jobbet med og ønsket å få møtet unnagjort. Om gruppen hadde fulgt møtestrukturen vi satte opp innledningsvis i prosessen mener vi at resultatet og dokumentasjonen av bacheloroppgaven hadde vært mer innholdsrik og nyttig.

[27] Mer om implementering av @AppStorage og @SceneStorage her: https://www.answertopia.com/swiftui/swiftui-data-persistence-using-appstorage-and-scenestorage/

Vi mener at oppdragsgiver kunne blitt inkludert mer under utviklingsprosessen. Oppdragsgiver ble oppdatert ofte tidlig i oppgaven fordi vi hadde behov for tilbakemelding angående kravspesifikasjon og valg vi tok før utvikling satt i gang. Utover i utviklingsprosessen mener vi derimot at mer tid kunne blitt satt av til de ukentlige møtene med oppdragsgiver for å inkludere og utdype mer det vi holdt på med. Gruppen syntes kommunikasjon var utfordrende da oppdragsgiver ikke hadde mye IT kompetanse og erfaring. Dette gjorde det vanskelig for gruppen å oppdatere oppdragsgiver på mål som ble oppnådd og hvilken funksjonalitet vi hadde implementert – særlig i back-end løsningen.

Gruppen møtte opp på faste tidspunkt, og hadde spesifikke mål for hva som skulle bli gjort i løpet av dagen. Noen av sprint målene var derimot litt for ambisiøse, og det kunne til tider skje at prosessen ble forlenget av uventede hendelser, som feil i kode, feil tankegang, og vanskelig implementasjon. Resultatet av dette var at målene som ble satt ikke ble nådd, og dermed måtte arbeidsperioden forlenges.

Resultatet av en agil utviklingsmetode var at man kunne gå tilbake for å forbedre funksjonalitet der det var nødvendig. Dette var særlig viktig under utviklingen av front-end da API-et ble brukt for fullt og gruppen kunne gå tilbake for å rette opp og forbedre funksjonalitet når feil oppsto. Ved hyppigere bruk av kvalitetssikring og vedlikehold kunne små feil vært unngått, som til gjengjeld ville spart tid.

10.3.2 Fordeling av arbeid

Fordelingen av arbeid fungerte godt under oppstart av prosjektet. Hvert gruppemedlem fikk hovedansvar for en del av oppgaven. Dette fungerte godt under utviklingsfasen og lot hvert gruppemedlem ta sjefsavgjørelser for sine respektive deler av oppgaven. Dette gjorde at alle gruppemedlemmer måtte lære seg å ta avgjørelser angående implementasjon og utvikling. Det ble derimot åpenbart under utvikling at programmeringsarbeidet var noe skjevt fordelt: gruppen mener at vi burde delt opp programmeringsarbeidet likt slik at alle fikk samme læringsutbytte av dette. Det går derimot ikke på bekostning av resultatet, men her hadde gruppen mulighet til å lære mer. På en annen side, mener vi at det å fordele ressursene på de ulike utviklingsoppgavene gjorde at gruppen fikk til mer arbeid, i motsetning til at alle medlemmene hadde hatt en sentral rolle i hver av de enkelte utviklingsoppgavene.

10.3.3 Prosjekt som arbeidsform

Gjennom bacheloroppgaven har viktigheten av å dokumentere arbeid, skrive møtereferater og holde ukentlige møter for et prosjekt som dette kommet tydelig fram. Særlig viktigheten av kommunikasjon med oppdragsgiver har blitt innsett. Mye fordi oppdragsgiver ikke var kjent med muligheter og begrensninger innenfor IT systemer, telematikk og programmering. Dialog med oppdragsgiver ga mye lærdom om kommunikasjon og hva man skal og ikke skal gjøre.

Struktur og selvdisiplin har vært viktig for en oppgave som dette. Oppgaven setter et stort ansvar på gruppen og dette har vi hatt mye læringsutbytte av. Det å jobbe konsekvent igjennom bacheloroppgaven har vært avgjørende. Samarbeid og tillit har også vært en essensiell del av oppgaven, da medlemmene kunne fordele arbeid mellom hverandre, og slå seg til ro med at

arbeidet blir gjort. Sammenligner vi den planlagte fremtidsplanen med den faktiske fremdriftsplanen innser vi at utvikling har tatt mer tid enn forventet. Innledningsvis ble det planlagt å dele opp alle utviklingsfaser (designe og utvikle API først, deretter front-end etc.) slik at front-end utvikling skulle gå problemfritt. API-utvikling og sporingsteknologi tok derimot mer tid enn forventet noe som gikk på bekostning av rapportskriving. Gruppen hadde også problemer med å legge fra seg deler av prosjektet og slå seg til ro med funksjonalitet når frister ankom.

10.4 Konklusjon

Ser vi tilbake på oppgavebeskrivelsen gitt av oppdragsgiver mener vi at oppgaven har blitt besvart på en god måte. Sporingsteknologien som har blitt utnyttet i oppgaven fungerer godt til problemstillingen, men er i dag for dyr til å kunne tas i bruk i stor-skala. Den økende trenden i teknologiene gir en indikator på at skalering skal være gjennomførbart om noen år, med tanke på kravene gitt av oppdragsgiver. Teknologien leverer god rekkevidde og fungerer godt gjennom stål. Bruken av LPWAN og BLE gir oppdragsgiver muligheten til å spore stillasdelene på prosjekt. Gruppen har ikke funnet brukbare enheter til dette bruksområdet, enhetene bør bli spesiallaget for å imøtekomme robusthet, batteritid og pris.

Den digitale løsningen vi har utviklet oppfyller oppdragsgivers ønsker, API-et har muligheter for optimalisering og forbedring, men leverer funksjonaliteten som er etterspurt. API-et kan integreres med allerede eksisterende systemer, eller brukes som en back-end for nye systemer. Kravene satt for databasen har i stor grad blitt oppfylt, men hastighet og optimalisering kan ved bruken av cloud functions forbedres. Databasen kan skaleres ut om det er ønskelig slik at oppdragsgiver kan utvide sporingen til annet utstyr. Web- og mobilapplikasjonen har funksjonaliteten som oppdragsgiver etterspurte, men gruppen mener at applikasjonene har rom for forbedring, da særlig innenfor optimalisering for skalering av data.

Til tross for forbedringspotensialet i alle deler av oppgaven mener vi at vi har levert et godt arbeid vi er stolt av. Vi har oppfylt kravene vi satte innledningsvis og leverte det oppgaven etterspurte. Vi har for noen av delene i oppgaven også levert mer enn det som ble etterspurt. Avslutningsvis vil vi gi en takk til oppdragsgiver *MB-Stillas,* da spesielt for deres engasjement og iver i oppgaven, men også for læringsutbyttet vi har fått fra oppgaven.

Bibliografi

1. **MyLift & Borud Stillas.** Webområde for MyLift & Borud Stillas. *OM MB STILLAS.* [Internett] 2021. [Sitert: 31 03 2022.] https://mbstillas.no/hvem-er-vi.html.
2. **DIGI.** digi.com. *What are the Differences Between LTE-M and NB-IoT Cellular Protocols?* [Internett] 06 03 2017. [Sitert: 01 02 2022.] https://www.digi.com/videos/what-are-the-differences-between-lte-m-and-nb-iot.
3. **Semtech.** lora-developers.semtech.com. *Wireless RF: The Ins and Outs of LPWAN Technologies.* [Internett] 10 2020. [Sitert: 20 01 2022.] https://lora-developers.semtech.com/uploads/documents/files/Wireless_RF_Matchbox_Approved_Final.pdf.
4. **Beta Solutions.** Beta Solutions Blog. *LPWAN Benefits for IoT Connectivity.* [Internett] Beta Solutions, 30 11 2017. [Sitert: 20 01 2022.] https://www.betasolutions.co.nz/blog/lpwan-benefits-for-iot-connectivity.
5. **Medium.** Medium. *LoRa LPWAN — Long Range Low Power Wide Area Network.* [Internett] Medium, 4 06 2020. [Sitert: 04 03 2022.] https://medium.com/jungletronics/lora-lpwan-long-range-low-power-wide-area-network-65cfa264d7c6.
6. **Verizon.** Verizon News Center. *What frequency is 5G?* [Internett] 18 11 2019. [Sitert: 04 05 2022.] https://www.verizon.com/about/our-company/5g/what-frequency-5g.
7. **Børje Forssell, Norvald Kjerstad.** Store Norske Leksikon. *GPS.* [Internett] 30 11 2021. [Sitert: 04 05 2022.] https://snl.no/GPS.
8. **Garmin.** Garmin. *What is GPS?* [Internett] [Sitert: 04 05 2022.] https://www.garmin.com/en-US/aboutgps/.
9. **Smiley, Suzanne.** Atlasrfidstore. *Active RFID vs. Passive RFID: What's the Difference?* [Internett] 10 12 2019. [Sitert: 04 05 2022.] https://www.atlasrfidstore.com/rfid-insider/active-rfid-vs-passive-rfid.
10. **NextPoints.** NextPoints. *What are active RFID tags and how do they work?* [Internett] [Sitert: 04 05 2022.] https://nextpoints.com/en/rfid-blog/active-rfid-tags/.
11. **Litum.** Litum. *What is Bluetooth Low Energy (BLE)? How does BLE work?* [Internett] 11 08 2021. [Sitert: 04 03 2022.] https://litum.com/blog/what-is-ble-how-does-ble-work.
12. **NESBO, ELLIOT.** Makeuseof. *What Is BLE (Bluetooth Low Energy) and How Does It Work?* [Internett] 27 12 2021. [Sitert: 04 05 2020.] https://www.makeuseof.com/what-is-ble-bluetooth-low-energy/.
13. **Stone, Mark.** Insights Samsung. *What is ultra-wideband, and how does it work?* [Internett] 25 08 2021. [Sitert: 04 05 2022.] https://insights.samsung.com/2021/08/25/what-is-ultra-wideband-and-how-does-it-work-3/.
14. **Schaefer, Lauren.** MongoDB. *What is NoSQL?* [Internett] [Sitert: 05 05 2022.] https://www.mongodb.com/nosql-explained.
15. **GeeksforGeeks.** Geeks For Geeks. *Horizontal and Vertical Scaling In Databases .* [Internett] 05 05 2022. https://www.geeksforgeeks.org/horizontal-and-vertical-scaling-in-databases/.
16. **Geeks for Geeks.** geeksforgeeks. *ReactJS | ReactDOM.* [Internett] 07 01 2021. [Sitert: 05 05 2022.] https://www.geeksforgeeks.org/reactjs-reactdom/.
17. **Abramov, Dan.** ReactJS. *React Components, Elements, and Instances.* [Internett] React, 18 12 2015. [Sitert: 05 05 2022.] https://reactjs.org/blog/2015/12/18/react-components-elements-and-instances.html.
18. **ReactJS.** ReactJS. *Hooks at a Glance.* [Internett] 26 04 2022. [Sitert: 05 05 2022.] https://reactjs.org/docs/hooks-overview.html.
19. —. ReactJS. *Rules of Hooks.* [Internett] React, 04 26 2022. [Sitert: 05 05 2022.] https://reactjs.org/docs/hooks-rules.html.

20. **Node.js.** Node.js. *Introduction to Node.js.* [Internett] [Sitert: 05 05 2022.] https://nodejs.dev/learn/introduction-to-nodejs.
21. **Gray, Tanya.** Medium. *What the heck is npm?* [Internett] 02 07 2018. [Sitert: 05 05 2022.] https://medium.com/@tanya/what-the-heck-is-npm-b8168f61e3b5.
22. **Bacinger, Tomislav.** Toptal. *What is Bootstrap? A Short Bootstrap Tutorial on the What, Why, and How.* [Internett] 2015. [Sitert: 06 05 2022.] https://www.toptal.com/front-end/what-is-bootstrap-a-short-tutorial-on-the-what-why-and-how.
23. **Khan, Shahpar.** Educative. *What is Material UI in React?* [Internett] [Sitert: 06 05 2022.] https://www.educative.io/edpresso/what-is-material-ui-in-react.
24. **React Router.** React Router. *Turtorial.* [Internett] [Sitert: 06 05 2022.] https://reactrouter.com/docs/en/v6/getting-started/tutorial#tutorial.
25. **Kodemaker.** kodemaker. *Overview.* [Internett] 2020. [Sitert: 06 05 2022.] https://react-query.tanstack.com/overview.
26. **Mapbox.** Mapbox Documentation. *Mapbox GL JS.* [Internett] 21 04 2022. [Sitert: 06 05 2022.] https://docs.mapbox.com/mapbox-gl-js/guides/.
27. **Mozilla.** Mozilla Developer. *Introducing asynchronous JavaScript.* [Internett] 15 03 2022. [Sitert: 07 05 2022.] https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing.
28. —. Mozilla Developer. *Promise.* [Internett] 05 04 2022. [Sitert: 07 05 2022.] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise.
29. —. Mozilla Developer. *How to use promises.* [Internett] 15 03 2022. [Sitert: 07 05 2022.] https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Promises#async_and_await.
30. **Gaur, Chandan.** xenonstack. *Local Storage vs Session Storage vs Cookie.* [Internett] 21 01 2022. [Sitert: 07 05 2022.] https://www.xenonstack.com/insights/local-vs-session-storage-vs-cookie.
31. **Geeks for Gekks.** geeksforgeeks. *Difference Between Local Storage, Session Storage And Cookies.* [Internett] 19 01 2022. [Sitert: 07 05 2022.] https://www.geeksforgeeks.org/difference-between-local-storage-session-storage-and-cookies/.
32. **Valdellon, Lionel.** CleverTap. *What Are the Different Types of Mobile Apps? And How Do You Choose?* [Internett] 01 11 2020. [Sitert: 07 05 2022.] https://clevertap.com/blog/types-of-mobile-apps/.
33. **Apple.** Apple. *Swift.* [Internett] [Sitert: 07 05 2022.] https://developer.apple.com/swift/.
34. **Bulavin, Vadim.** Vadimbulavin. *Layered Architecture to Design iOS Apps.* [Internett] 29 07 2019. [Sitert: 07 05 2022.] https://www.vadimbulavin.com/layered-architecture-ios/.
35. **Apple.** Apple Developer. *SwiftUI.* [Internett] [Sitert: 07 05 2022.] https://developer.apple.com/xcode/swiftui/.
36. —. Apple Developer. *UIKit.* [Internett] [Sitert: 08 05 2022.] https://developer.apple.com/documentation/uikit.
37. —. Apple Developer. *MapKit.* [Internett] [Sitert: 08 05 2022.] https://developer.apple.com/documentation/mapkit/.
38. —. Apple Developer. *Foundation.* [Internett] [Sitert: 08 05 2022.] https://developer.apple.com/documentation/foundation.
39. —. Apple Developer. *Core Location.* [Internett] [Sitert: 08 05 2022.] https://developer.apple.com/documentation/corelocation.
40. —. Apple Developer. *Combine.* [Internett] [Sitert: 08 05 2022.] https://developer.apple.com/documentation/combine.
41. —. Swift. *Package Manager.* [Internett] [Sitert: 08 05 2022.] https://www.swift.org/package-manager/.

42. **Google.** Firebase. *Get Started with Firebase Authentication on Apple Platforms.* [Internett] 18 05 2022. [Sitert: 08 05 2022.] https://firebase.google.com/docs/auth/ios/start.
43. **auth0.** auth0. *What is OAuth 2.0?* [Internett] [Sitert: 12 05 2022.] https://auth0.com/intro-to-iam/what-is-oauth-2/.
44. **Raible, Matt.** Okta Developer. *What the Heck is OAuth?* [Internett] 21 05 2017. [Sitert: 12 05 2022.] https://developer.okta.com/blog/2017/06/21/what-the-heck-is-oauth.
45. **auth0.** jwt. *Introduction to JSON Web Tokens.* [Internett] [Sitert: 12 05 2022.] https://jwt.io/introduction.
46. —. auth0. *JSON Web Tokens.* [Internett] [Sitert: 12 05 2022.] https://auth0.com/docs/secure/tokens/json-web-tokens.
47. **Google.** Firebase. *Firebase Authentication.* [Internett] 18 05 2022. [Sitert: 12 05 2022.] https://firebase.google.com/docs/auth.
48. **Enginess.** Enginess. *The 6 Principles Of Design, a la Donald Norman.* [Internett] 03 11 2014. [Sitert: 10 05 2022.] https://www.enginess.io/insights/6-principles-design-la-donald-norman.
49. **Gordon, Kelley.** nngroup. *Visual Hierarchy in UX: Definition.* [Internett] 17 01 2021. [Sitert: 09 05 2022.] https://www.nngroup.com/articles/visual-hierarchy-ux-definition/.
50. **Andrew Kirkpatrick, Joshue O Connor, Alastair Campbell, Michael Cooper.** w3. *Web Content Accessibility Guidelines (WCAG) 2.1.* [Internett] 05 06 2018. [Sitert: 10 05 2022.] https://www.w3.org/TR/WCAG21/.
51. **interaction-design.** interaction-design. *Accessibility.* [Internett] [Sitert: 10 05 2022.] https://www.interaction-design.org/literature/topics/accessibility.
52. **Apple.** iOS Design Themes. [Internett] Apple. [Sitert: 16 05 2022.] https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/.
53. —. Apple Developer. *Testing Your Apps in Xcode.* [Internett] [Sitert: 14 05 2022.] https://developer.apple.com/documentation/xcode/testing-your-apps-in-xcode.
54. **Sundell, John.** SwiftBySundell. [Internett] swiftbysundell, 10 03 2019. [Sitert: 16 05 2020.] https://www.swiftbysundell.com/articles/integration-tests-in-swift/.
55. **Bose, Shreya.** Browserstack. *End To End Testing: A Detailed Guide.* [Internett] 19 05 2021. [Sitert: 14 05 2022.] https://www.browserstack.com/guide/end-to-end-testing.
56. **Seodesignchicago.** Seodesignchicago. *How to Deploy a Website* . [Internett] 14 05 2022. https://seodesignchicago.com/web-design-blog/how-to-deploy-a-website/.
57. **umbraco.** Umbraco. *What is Deployment?* [Internett] 14 05 2022. https://umbraco.com/knowledge-base/deployment/.
58. **World Economic Forum.** Webområde for World Economic Forum. [Internett] Uvisst. [Sitert: 2022 01 26.] https://reports.weforum.org/digital-transformation/understanding-the-impact-of-digitalization-on-society/.
59. **Telia.** telia.no. [Internett] [Sitert: 26 04 2022.] https://www.telia.no/bedrift/digitalisering/iot/skreddersydde-iot-losninger-for-din-bedrift/low-power-wide-area-iot/.
60. **Armada Dynamics .** armadadynamics.no. [Internett] [Sitert: 2022 02 22.] https://armadadynamics.no/en/ .
61. **HAKI.** haki.no. [Internett] [Sitert: 20 02 2022.] https://www.haki.no/om/.
62. **Nikolaieva, Aliona.** Up Tech. [Internett] 17 01 2022. [Sitert: 21 01 2022.] https://www.uptech.team/blog/software-development-methodologies.
63. **Phuong.** DevOps. *Relational vs non-relational databases.* [Internett] 16 07 2018. [Sitert: 06 05 2022.] https://devops.com.vn/relational-vs-non-relational-databases/.
64. **ABET, Mathieu.** Elainnovation. *Bluetooth vs Bluetooth Low Energy, what's the difference?* [Internett] 03 12 2021. [Sitert: 04 05 2022.] https://elainnovation.com/en/bluetooth-vs-bluetooth-low-energy-whats-the-difference/.

Appendix

Appendix A Akronymer

ACID
atomicity, consistency, isolation, durability 17
BLE
Bluetooth low energy 4; 12; 13; 14; 37; 42; 43; 44; 45; 47; 48
CRUD
Create, Read, Update, Delete 25
DOM
Document Object Model 20; 21
GPS
Global Positioning System 11; 12; 37; 41; 44
HTTP
Hypertext Transfer Protocol 14; 46; 70; 71
IoT
Internet of things 1; 3; 5; 7; 10; 11; 12; 13; 15; 42
JSON
JavaScript Object Notation 15; 16; 17; 19; 69
JWT
JSON WEB TOKEN 112
LPWAN
Low power wide area nework 11; 12; 14; 37; 42; 44
MiB
Mebibyte 123
MOSCOW
must-have, should-have, could-have, won't-have, will not have right now 76; 90; 91
MQTT
Message Queuing Telemetry Transport 14; 15; 70; 71
MVC
Model–view–controller 24; 25; 109
MVVM
Model-view-viewmodel 24; 25; 26; 100; 103; 109
NFC
Near Field Communication 13; 41
QR
Quick response 41
RDBMS
relational database management system 16
RFID
Radio Frequency IDentification 12; 40; 42; 43; 44
TLS
Transport Layer Security 15
UWB
Ultra-wideband 14; 41
WCAG
Web Content Accessibility Guidelines 75; 92

Appendix B Definisjoner

AES-128 er en kryptering algoritme som blir brukt for å sikre sensitiv data. Denne algoritmen bruken er 128-bit symmetrisk nøkkel som krypterer og dekrypterer informasjonen.

Annoteringer blir brukt for å fremheve et punkt eller geografisk område på kart.

Beacons er en enhet som sender pulser med energi, data eller andre signaler. Disse signalene kan bli plukket opp av andre enheter, som leser signalet.

Bindings er en variabel som kan lese og skrive en verdi på tvers av views.

Broker er en programvare som oversetter og sorterer informasjon fra IoT-enheter.

CSS, Cascading Style Sheets, er et språk som brukes til å definere utseende på filer skrevet i XML eller HTML.

DNS, domain name system, er en Internett-tjeneste som kobler sammen domenenavne og IP-adressen. DNS oversetter domenenavnet *mbstillas.no* til IP-adressen til serveren som inneholder nettsiden.

Forespørsel body er data sendt fra brukeren til et API.

Gateway er en maskinvarekomponent som brukes for sammenkobling mellom ulike nettverk for telekommunikasjon.

Geokoding er prosessen å forandre geografiske data, som eksempelvis adresser til geografiske koordinater.

HCI, Human Computer Interaction, er et fagfelt om omhandler interaksjonen mellom mennesker og datamaskiner.

Header er tilleggsdata som er plassert i begynnelsen av en blokk med data som blir overført vi nettet.

HMAC, Hash-Based Massage Authentication Codes, er en spesifikk type meldingsautentiseringskode som utnytter en kryptografisk hash-funksjon og en hemmelig kryptografisk nøkkel.

Informasjonskort er kort som gruppen anvender i den digitale løsningen. Kortene fremlegger informasjon fra API-et.

iOS er et operativsystem utviklet av Apple. Dette operativsystemet blir kjørt på Apples iPhone-modeller.

IP-sertifisering er et system for å oppgi beskyttelsesgraden mot inntrengning av faste gjenstander og vann.

JSON, JavaScript Object Notation, er et tekstformat som presenterer strukturert data basert på JavaScript objekt syntaks. JSON blir vanligvis brukt for overføring av data i webapplikasjoner.

Kompileringsteknologi er en teknologi som oversetter kode skrevet i høynivåspråk til objekt kode, slik at den kan bli kjørt av datamaskinen.

LTE-M er en standard for lavt strømnettverksradioteknologi. Denne teknologien er en underkategori av LPWAN, som er optimalisert for høye brebånds forbindelser.

Middleware er en betegnelse på et mellomledd mellom systemer. Dette mellomleddet vil gjøre det lettere å få to systemer til å snakke sammen, på tvers av formater, protokoller og teknologier.

NB-IoT er en standard for lavt strømnettverksradioteknologi. Denne teknologien er en underkategori av LPWAN, som leverer nettverk med lave brebånds forbindelser.

Open-Source er programvare med åpen kildekode. Opphavsrettsinnehaveren gir brukerne rettigheter til å utnytte seg av kildekoden til ethvert formål.

Polygoner er en geometrisk figur med flere kanter. (I den digitale løsningen blir dette brukt for å markere arealet til et geografisk område.)

Ruter er et verktøy som brukes i diverse programmer, som lar deg definere en URL. Når URL-en blir forespurt vil det kunne bli programmert en handling som skal skje.

Single-page applikasjoner er en webapplikasjon som navigerer brukeren gjennom nettstedet ved å oppdatere innholdet på siden, i motsetning til å laste inn en helt ny side fra en server.

Struct representerer en kolleksjon av elementer med forskjellige datatyper. Det kan bli brukt for en gruppe elementer av mange typer til en enkel type.

TLS, Transport Layer Security, er kryptografiske protokoller som sikrer kommunikasjon på nettet.

Transpondere er et svar anlegg som består av en mottaker og en sender. Når leseren og transponderen er innen rekkevidde vil leseren sende ut et signal til transponderen, som deretter sender tilbake den relevante informasjonen.

Tredjeparts bibliotek er et bibliotek som ikke er laget eller vedlikeholdt av selskapet som har utviklet programmeringsspråket. En utvikler vil ha muligheten til å importere dette biblioteket, og anvende det i koden.
UTF-8 er en binær representasjonsform for tegn. UTF-8 koding omgjøre string til tall. Eks. «hello»: 104 101 108 108 111

Views er visuelle blokker av en applikasjon sitt brukergrensesnitt.

XML er et markeringsspråk brukt for deling av strukturert data mellom informasjonssystemer. Dette språket er likt HTML, men uten predefinere tagger.

YAML er et dataserialiseringsspråk som er menneskelig lesbart. Applikasjoner der data blir lagret og overført, blir YAML ofte brukt.

Appendix C Oppdatert Fremdriftsplan

Mylift Oppgave

Milepæler

Beslutningspunker

tir, 1.11.2022

1

TASK	PROGRESS	START	END
Oppstart			
Oppsett av verktøy	100 %	1.11.22	1.15.22
Sporingteknologi	100 %	1.11.22	2.25.22
Prosjekt Plan	100 %	1.11.22	1.31.22
Løsning av sporer	100 %	2.28.22	4.1.22
Nødvendige modeller (DB, DFD, UC)	100 %	1.31.22	2.5.22
Satusrapport	100 %		
Database			
Design	100 %	2.28.22	3.11.22
Implementasjon	100 %	3.11.22	3.21.22
Test	100 %	3.11.22	3.21.22
Deployment	100 %	3.21.22	3.23.22
API			
Design	100 %	2.28.22	3.11.22
Implementasjon	100 %	3.12.22	4.8.22
Test	100 %	3.12.22	4.8.22
Deployment	100 %	4.8.22	4.9.22
Webapplikasjon			
Design	100 %	2.28.22	3.11.22
Implementasjon	100 %	3.26.22	5.5.22
Test	100 %	5.5.22	5.10.22
Deployment	0 %	5.10.22	5.11.22
Mobilapplikasjon			
Design	100 %	2.28.22	3.11.22
Implementasjon	100 %	3.19.22	5.5.22
Test	100 %	5.5.22	5.10.22
Deployment	0 %	5.10.22	5.11.22
Sluttfase			
Rapport	100 %	5.1.22	5.20.22

Appendix D Prosjektplan

Innholdsfortegnelse

1 Mål og rammer

1.1 Bakgrunn

Stillas er en arbeidsplattform som er satt opp midlertidig for å forenkle arbeid på en byggeplass. Et stillas er bygget opp av flere komponenter i forskjellige størrelser og fasonger. Disse delene blir transportert mellom lager og byggeplass, internt på byggeplasser og mellom byggeplasser uten å returnere til lageret. Utstyret kontrolleres sjeldent før det sendes ut til byggeplassen, ei eller når det returneres tilbake til lageret. Tap av utstyr blir tidkrevende og er en uønsket kostnad for bedriften. Ved å bruke tilgjengelig teknologi ønsker bedriften å minimere tap av utstyr, få mer kontroll på hvor utstyret befinner seg til enhver tid i tillegg til å frigjøre ressurser til andre arbeidsoppgaver.

1.2 Prosjektmål

Effektmål: Oppdragsgiver ønsker å oppnå følgende:

- Minke svinn ved transportering, utkjøring og utleie av stillas.
- Hvilket utstyr går ut fra lager til prosjekt, og fra prosjekt til prosjekt.
- Spore stillasdeler med beste tilgjengelige teknologi.
- Effektivisere telling av stillas.
- Danne et grunnlag for hvilket utstyr som går tapt.
- Undersøke forskjellige sporingsløsninger og sporingsenheter som passer bruksområdet.

Resultatmål: Oppdragsgiver vil få en programvare som kan følgende:

- Spore stillas på en effektiv og billig måte med minst mulig menneskelig interaksjon.
- Visualisere oversikt over stillasets lokasjon ved hjelp av en mobil applikasjon.
- Løsningen skal ha et åpent API slik at en potensiell neste fase kan koble på flere teknologier.
- Kunden skal kunne få beskjed når stillaset kjøres ut av lager.
- Skal kunne brukes uavhengig av teknisk kompetanse.

Oppdragsgiver vil få en sporingsløsning som oppfyller følgende:

- Fungerer godt uavhengig av værforhold og håndtering.
- Lang levetid både med tanke på batteritid og robusthet.
- Har en størrelse som gjør at sporingsenheten kan festes på alle ønskelige stillasdeler der det lar seg gjøre.
- Sporingsenheten skal være billig.

Det vil lages en programvare som oppfyller satte krav der dette lar seg gjøre. Det vil leveres et forslag på flere sporingsmetoder og eventuelle enheter som gruppen mener oppfyller så mange av oppdragsgivers krav som mulig. Gruppen vil legge vekt på at den valgte sporingsteknologien er fremtidsrettet slik at oppdragsgiver kan integrere en passende enhet når teknologien ankommer markedet. Gruppen vil derimot ikke utvikle en egen sporingsenhet, da dette ligger utenfor gruppens fagfelt samt læringsmål.

Læringsmål: Studenten sitt læringsutbytte er følgende:

- Innsikt innenfor teknologi i bygg bransjen og deres arbeidsmetoder.
- Innsikt innenfor relevant maskinvare og teknologi.
- Anvende relevant kunnskap for å løse teoretiske og praktiske problemstillinger.
- Dokumentere, planlegge og gjennomføre et ingeniørfaglig arbeid.
- Innsikt i økonomiske og bærekraftige konsekvenser av produktløsningen.
- Formidle faglig kunnskap til ulike målgrupper skiftelig og muntlig.
- Arbeide med en reel oppdragsgiver.

1.3 Rammer

Gruppen har fått mye tillit fra oppdragsgiver og står relativt fritt til å ta avgjørelser om hva som IT-teknisk er best for oppgaven, men oppdragsgiver ønsker å sette gruppen i dialog med IT-selskaper som har jobbet med selskapet tidligere slik at gruppen kan henvende seg til et selskap med IT kompetanse der det er behov.

Manglende kunnskap og ferdigheter innenfor maskinvare setter en begrensing på funksjonalitet og durabilitet for det endelige produktet, mer spesifikt sporingsenheten. I tillegg er selskapets stillas en begrensning med tanke på byggematerialer som forstyrrer signaler. Ettersom stillaset har ulik oppbygging, utsettes for brutal behandling, og delene har forskjellige fasonger, vil festing av sporingsenheter bli et mulig problem.

Covid-19 setter en begrensning for gruppens mulighet til å besøke oppdragsgiver. Dette kan begrense innsikten i prosessen til bedriften og systemet de benytter seg av, samt gjøre oppstarten av prosjektet, når fokuset ligger på maskinvare valg og/eller utvikling, vanskeligere. Dette gjør at resultatet kan bli mindre integrert i bedriftens systemer og arbeidsmetoder.

Gruppen står fritt i valg av programmeringsspråk, systemarkitektur, utviklingsmiljø og operativsystem.

2 Omfang

2.1 Fagområde

Det moderne samfunnet utvikler seg i stor fart og teknologi-bølgen blir større og større. Det produseres og kjøpes nye produkter, og logistikken i hva man eier og hvor det befinner seg blir mer uklart. Derfor har behovet for å holde orden på og ha muligheten til å kunne spore gjenstander økt. Ettersom logistikk utgjør en stor del av en produserende og transporterende virksomhets lønnsomhet, implementeres nå IT-tekniske løsninger for logistikk og oversikt i stadig flere bransjer. Datateknologiske løsni nger for sporing og logistikk er raskere, mer oversiktlig, enklere i bruk og ikke minst langt mer lønnsomme enn manuelle, menneskelige løsninger[28].

[28] Et estimat gjort av *World Economic Forum* anslår at i løpet av perioden 2016-2025 vil logistikk i industrien oppleve økt lønnsomhet ved hjelp av digitalisering (World Economic Forum u.d.)

2.2 Avgrensing

Stillas er en arbeidsplattform som er satt opp midlertidig for å forenkle arbeid på en byggeplass. Et stillas er bygget opp av flere komponenter i forskjellige størrelser og fasonger. Mylift & Borud stillas AS er en totalentreprenør som leverer montering, frakt og leie av stillas innenfor bygg- og anleggs-bransjen. Oppdragsgivers problem ligger innenfor logistikk angående sporing og telling av stillas på byggeplass og lager, noe som resulterer i svinn og bortkastet tid. De har kommet til NTNU og ønsker en løsning som gjør sporing og logistikk angående utleie av stillas mulig. Gruppen skal på bakgrunn av dette gå inn på områder innenfor logistikk, sporing og stillas.

Denne oppgaven tar for seg hvordan man kan integrere moderne teknologi inn i en teknologisk understimulert bransje gjennom sporing av stillas på lager, under transport og ute på byggeplassen. Gruppens fokus vil ligge i en digital løsning, gitt at gruppens kompetanse hovedsakelig er programvare, ikke maskinvare. Utover dette vil løsningen også inkludere en anbefaling om hvilken sporingsteknologi som er best egnet oppdragsgivers ønsker på skrivende tidspunkt. Forutsatt at en mulig løsning foreligger, skal også en prototype utvikles.

2.3 Oppgavebeskrivelse

Oppgaven tar for seg frigjøring av mest mulig ressurser ved å automatisere stillaslogistikkprosessen til oppdragsgiver. For å oppnå dette skal gruppen:

- Finne en kostnadseffektiv måte å spore enkelte stillasdeler.
    - Finne den mest egnede sporingsteknologien.
    - Finne en sporingsløsning som passer problemstillingen.
- Spore stillasdeler på tvers av byggeprosjekter.
- Sporingsteknologien som benyttes skal være:
    - Bærekraftig ○ Fremtidsrettet ○ Robust ○ Ha lav kostnad for å anses som lønnsom.
- Telle stillas som transporteres inn og ut av lager.
- Loggføre forflytning av stillas.
- Registrere, og fjerne stillas fra prosjekter.

Gruppen skal undersøke diverse mulige sporings-muligheter som kan tas i bruk for å løse problemet, diskutere med stillas produsenter, samt oppdragsgiver for å kunne danne et grunnlag for valget som vil bli tatt angående sporings-teknologi og enhet.

Det skal utvikles en løsning som gjør følgende:

- Ansatte har tilgang til informasjonen uavhengig av arbeidsplass.
- Funksjonaliteten vil bli implementert ved hjelp av en robust API
- Løsningen skal kunne videreutvikles og/eller integreres i oppdragsgivers andre IT-løsninger.
- Vise alle selskapets stillas i et kart på en enkel måte som gjør det lett for arbeidere å bruke applikasjonen.
- Her skal det kunne registreres stillas som er knyttet til databasen, og administrere stillasene knyttet til prosjektene.

3 Prosjektorganisering

3.1 Ansvarsforhold og roller

Gruppeleder	Martin Iversen	Gruppeleder har overordnet ansvar for oppgaven, skal ha en god oversikt over prosjektet i sin helhet slik at prosjekt-prosessen går som planlagt. Om det oppstår konflikter innad i gruppen vil gruppelederen måtte ta en avgjørelse.
Database	Tormod Mork Müller	Vil ha hovedansvar for oppsett og konfigurering av databasen til løsningen, ansvarlig vil kunne ta hoved-avgjørelser angående databasens design og implementasjon.
Mobilutvikling	Aleksander Aaboen	Vil ha hovedansvar for utvikling av mobil-løsning. Vil ta viktige avgjørelser angående appens funksjonalitet
Webutvikling	Tormod Mork Müller	Vil ha hovedansvar for utvikling av web-løsning. Vil ta viktige avgjørelser angående appens funksjonalitet
Infrastruktur og distribuering	Martin Iversen	Vil ha ansvar for infrastrukturen løsningen er bygget på. Vil ta hovedavgjørelser angående hva løsningen skal distribueres på. Vil også kunne ta avgjørelser angående systemets arkitektur.
Bachelor-Veileder	Frode Haug	Vil veilede gjennomførelse av bacheloroppgaven
Oppdragsgiver	Knut Rindal	Oppdragsgiver for oppgaven

3.2 Rutiner og grupperegler

Dette dokumentet inneholder forventninger, regler og andre momenter knyttet til

gruppens arbeidsinnsats og mengde igjennom bacheloroppgaven.

Gruppemedlemmer:

Tormod Mork Müller, Martin Iversen, Aleksander Aaboen

Arbeidsinnsats:

Det forventes av gruppen at alle gruppemedlemmer møtes fysisk fire av fem dager i uken. Om et medlem er syk eller ikke har mulighet til å møte forventes det at de andre gruppemedlemmene varsles og at man ser på alternative arbeidsmetoder. Det forventes at hvert gruppemedlem jobber minimum 30 timer i uken.

Innen 20. mai bør hvert gruppemedlem ha et minimum av 540 timer investert i prosjektet. Om endringer skal foretas må dette tas opp med hele gruppen og dokumenteres. Det forventes at alle gruppemedlemmer er til stede på alle møter der det er gunstig (spesifikke møter relatert til en spesifikk del av prosjektet trenger nødvendigvis ikke å involvere alle gruppemedlemmer).

Avgjørelser:

Alle gruppemedlemmer må være enige før en avgjørelse blir tatt. Dersom gruppen ikke klarer å bli enige i administrative problemer eller problemer som omhandler oppgaven som helhet, vil gruppeleder stå inne for den endelige avgjørelsen. Om problemet omhandler en spesifikk

del av prosjektet vil først og fremst medlemmet som er ansvarlig for denne delen (eksempelvis database-ansvarlig) ta den endelige avgjørelsen. Dette er på bakgrunn av at denne personen vil ha mer ekspertise og kompetanse innenfor feltet i motsetning til andre gruppemedlemmer.

Kommunikasjon i gruppen

Det forventes at alle gruppemedlemmer informeres om alle avgjørelser og endringer i planer angående oppgaven. Om et gruppemedlem ikke har mulighet til å møte på møter/campus skal alle medlemmer informeres. Hvis et gruppemedlem er misfornøyd med noe som angår oppgaven er de pliktige til å informere gruppemedlemmene så fort som mulig, slik at irritasjonsmomenter ikke blir et problem.

Det forventes at hvert gruppemedlem loggfører timer aktivt, uavhengig av om det er individuelt arbeid eller arbeid med gruppen.

Møter

Det vil holdes ukentlige møter internt i gruppen. Her vil det skrives et referat, dette går på rundgang slik at alle i gruppen er involvert. Møtene vil brukes som en statusoppdatering for å opprettholde oversikt over fremdrift og oppgaven generelt. Møtereferat vil inneholde følgende:

- Hva som har blitt diskutert og gjort
- Avgjørelser som er tatt
- Hva som bør jobbes med til neste møte
- Tidspunkt for neste møte

Konsekvenser ved kontraktbrudd

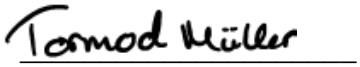
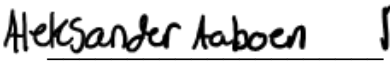
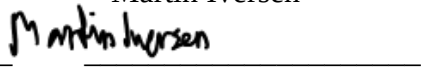
Om deler av kontrakten blir brutt vil følgende konsekvenser følges, a-c basert på alvorligheten på kontraktbruddet:

a) Det vil bli arrangert fellesmøter hvor bruddet diskuteres internt i gruppen.

b) Veileder vil bli involvert i saken og eventuell konsekvens vil bli diskutert.

c) Det sendes ut en skriftlig advarsel fra de andre gruppemedlemmene.

Underskrifter

Tormod Mork Müller	Aleksander Aaboen	Martin Iversen
		

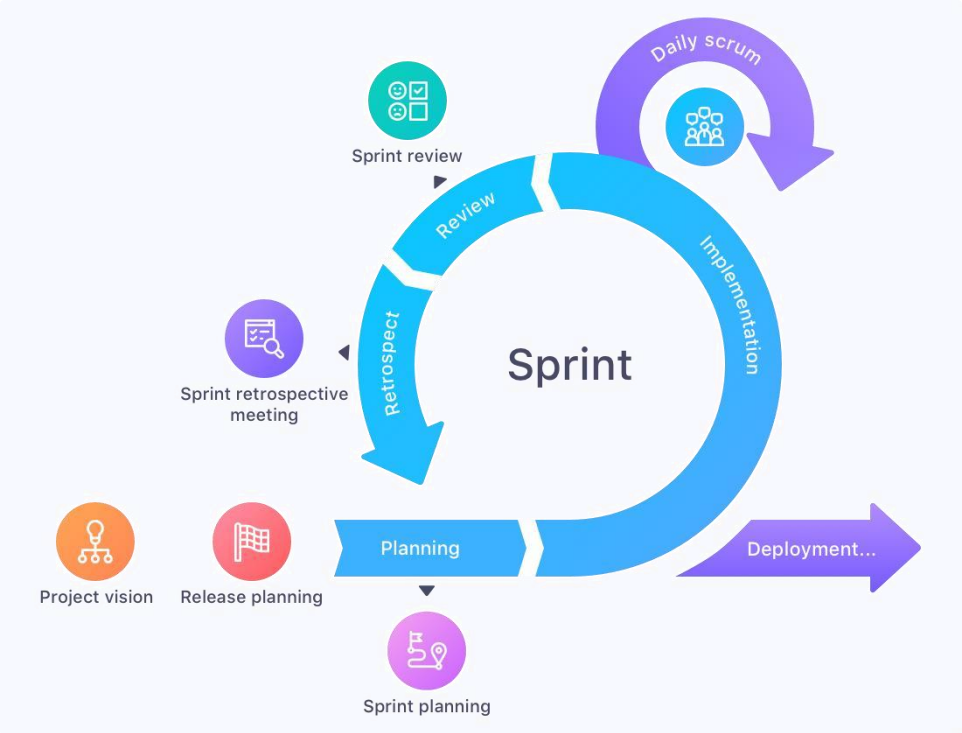
4 Planlegging, oppfølging og rapportering

4.1 Hovedinndeling av prosjektet

Ettersom oppdragsgiver ikke krever en spesifikk utviklingsmetode, står gruppen fritt til å velge det vi mener passer best for vår oppgave. Oppgaven er foreslått av et selskap som har lav IT kompetanse, dermed vil avgrensing og kravsetting tidlig i prosjektprosessen være kritisk for å oppnå et godt resultat.

For prosjektet ønsker gruppen å ha et søkelys på å legge et godt grunnlag for krav og systemarkitektur i løsningen. Dette vil gjøre at utviklingsdelen av prosessen vil flyte bedre samt at det vil brukes minimal tid på usikre krav og diskusjon rundt disse. Ettersom oppdragsgiver er fleksibel i hvordan oppgaven løses/hva som blir vektlagt, vil en arbeidsmetode der gruppen kan tilpasse arbeidet rundt oppdragsgivers ønsker være ønskelig.

Fordi systemet skal lages for en oppdragsgiver med et ukonvensjonelt arbeidsmiljø med tanke på ITbransjen, vil det være ønskelig å designe systemet med fokus på deres arbeidsmetoder og rutiner. På grunn av dette vil en agil utviklingsmetode som gjør det mulig for oss å få tilbakemelding angående bruken av systemet være nyttig, slik at det blir tilpasset arbeidet i bedriften.



Figur 1 Visualisering av SCRUM modell (Engberg 2021)

For prosjektet har gruppen valgt å bruke SCRUM utviklingsmetoden, denne metoden gir frihet til å både tilpasse utviklingsprosessen i forhold til arbeidsgivers ideer, dette i tillegg til gruppens evne til å tilpasse seg uforutsette komplikasjoner. Metoden vil holde prosessen oversiktlig ved hjelp av ukentlige møter og «sprint» forberedelser. Metoden inkluderer også oppdragsgiver i stor grad noe som vil resultere i en mer brukervennlig løsning. Gruppen ønsker i tillegg mer erfaring med en agil utviklingsmetode gjør dette til den foretrukne metodikken innenfor programvare-utvikling (Nikolaieva 2022).

I starten av prosessen vil det bli laget en liste med krav der alt av funksjonalitet til systemet er listet opp, denne funksjonaliteten blir hentet fra oppgavebeskrivelsen og fra dialog med oppdragsgiver. Når kravene er satt og utviklingen går i gang, vil en sprint begynne. I begynnelsen av hver sprint vil gruppen ha et møte angående målene for sprinten, og følge «user-stories» for å forme kravene for den kommende sprinten. Gjennom daglig SCRUM vil det under hver sprint bli holdt møter der det blir diskutert hva som har blitt oppnådd så langt, hva som skal bli oppnådd i løpet av dagen og vanskeligheter som har oppstått under sprinten. Når sprinten er ferdig og kravene fra «user-storiene» er oppnådd, vil gruppen møte med oppdragsgiver for å demonstrere resultatene. Gruppen vil også holde et møte etter hver sprint,

der det blir diskutert hva som var bra med sprinten, og hva som kunne ha blitt bedre, dette er for å øke kvaliteten på sprintene.

4.2 Plan for statusmøter og beslutningspunkter i perioden

Statusrapporter:

Gruppen skal i løpet av oppgaveperioden levere tre statusrapporter til veileder, disse vil inneholde status på prosjektet, hva som er gjort og hva som skal gjøres frem til neste statusrapport (mal blir gitt av veileder). Statusrapporter vil brukes som en tilbakemeldingskilde for gruppen og skal leses og vurderes av veileder.

Rapporter vil leveres på følgende datoer:

- Første statusrapport: 20. februar
- Andre statusrapport: 1. april
- Siste statusrapport: 1. mai

Beslutningspunkter:

Gruppen har laget beslutningspunkter for kritiske momenter ved oppgaven. Dette er punkter som er kritiske for oppgavens gjennomførelse. Om fristene knyttet til punktene blir overskredet må en beslutning tas så fort som mulig slik at prosessen skal kunne fortsette som planlagt.

Valg av sporingsteknologi og mulig enhet:

Valg av sporingsteknologi og enhet vil være kritisk for første del av prosjektet, da løsningen må utvikles rundt en sporingsteknologi som brukes med en sporingsenhet. Om denne delen av prosjektet tar for mye tid vil gruppa måtte starte utviklingsprosessen uten en klar idé om hvilken type sporingsteknologi som vil bli tatt i bruk. Dette vil gå utover kvaliteten på løsningen, samt effektiviteten i prosessen.

Valg av programmeringsspråk:

For å kunne starte utviklingsprosessen må et programmeringsspråk velges, uten dette vil kun databasen kunne settes opp og jobb med selve løsningen vil ikke kunne begynne.

Ferdigstille API/Grunnfunksjonalitet:

Gitt at grunnfunksjonaliteten vil bli sentrert rundt innhenting av data fra sporingen er denne funksjonaliteten essensiell for løsningen. Om designet av denne funksjonaliteten ikke kommer på plass vil prosjektet stagnere, gruppen vil ikke kunne bygge videre på løsningen om dette ikke er gjort skikkelig.

Beslutningspunkt	**Frist**
Valg av sporingsteknologi og mulig enhet	05.02.2022
Valg av programmeringsspråk	11.02.2022
Beslutte API design	23.02.2022

5 Organisering av kvalitetssikring

5.1 Dokumentasjon, standarder, konfigurasjonsstyring og verktøy

5.1.1 Dokumentasjon

All dokumentasjon utenom kode-kommentering vil foregå på norsk da dette er morsmålet til alle gruppemedlemmer, veileder, og oppdragsgiver. Til tross for at IT-bransjen hovedsakelig opererer på engelsk har gruppen valgt å bruke norsk.

Ettersom programvaren er bygget rundt maskinvaren vil ideer og vedtekter som omhandler sporingsenheten bli dokumentert. Gruppen mener at dokumentasjon av denne prosessen er av stor verdi for oppdragsgiver både for å vise at gruppen har investert tid i denne delen av prosjektet, samt for å gi oppdragsgiver en rapport som omhandler forskjellige sporingsløsninger og teknologier. Dette vil gi oppdragsgiver mulighet til å lettere arbeide videre med problemstillingen etter oppgavens avslutning om dette er ønskelig.

Ukentlige møter vil bli avholdt og møtereferater vil bli skrevet både for møte med veileder, oppdragsgiver, eventuelle andre samarbeidspartnere og internt i gruppen. Møtereferater vil inneholde:

- Agenda
- Hovedpunkter ved møtet
- Andre tilbakemeldinger
- Deltakere

Det vil også loggføres timer daglig av alle gruppemedlemmer. Disse timene blir ført og lagt sammen i Excel, og loggføringen vil inneholde mengde timer og stikkord rundt arbeidet. Under utviklingsprosessen vil SCRUM metodikk bli brukt. Her vil hver SCRUM «sprint» bli dokumentert med referater som inneholder hva som har blitt gjort og hva som må gjøres til neste gang.

For kildehenvisning skal Words integrerte referanseverktøy brukes. Her vil kildens lastet opp/endret, sammen med tidspunktet hvor gruppen besøkte/brukte kilden brukt for å dokumentere datoer relevant for kilden. Sitater skal legges inn og referanseliste skal oppdateres for hvert nye sitat. Bilder og figurer skal også legges inn i henholdsvis bildeliste og figur-liste.

5.1.2 Konfigurasjonsstyring

Det skal brukes Git under utviklingsprosessen. Git commits skal brukes ofte for å kunne dokumentere prosessen, samt skape mulighet for å tilbakestille prosjektet til en tidligere fase om nødvendig. Under utviklingsprosessen vil gruppen bruke GitLab og Git kontinuerlig i løpet av utviklingsprosessen. Hvert gruppemedlem vil kode i sin egen *Git* gren og *merge requests* vil bli brukt. Disse må godkjennes av gruppa før kode blir flettet inn i *master* grenen.

Git vil brukes i sammenheng med GitLab, da dette er en DevOps plattform som gjør det enklere for team eller organisasjoner å levere programvare raskere og mer effektivt, samtidig som det blir holdt sikkert og i samsvar med regler. På denne plattformen vil gruppen planlegge, bygge og utplassere programvaren. I tillegg vil *GitLabs* «issue board» bli brukt

under utviklingsfasen av prosjektet for å opprettholde oversikt over utviklingsprosessen og dens gjøremål med hensyn på implementering.

5.1.3 Standarder

Gruppen vil for alle møtereferater og loggføring følge en satt mal laget av gruppen.
Kode vil kommenteres og programmeringsstandarder vil bli brukt med metoder lært fra diverse programmeringsfag hittil i studiet. Disse innebærer blant annet:

- Minske bruk av globale variabler.
- Bruke headere for både klasser og funksjoner.
- Standardisere bruk av navn for variabler.
- Standard bruk av innrykk.
- Feil-håndtering av funksjoner.
- Pålegge bruk av prinsippet «Loose-Coupling and High Cohesion».

5.1.4 Verktøy

Microsoft Word

Microsoft Word er en programvare for tekstbehandling. I dette prosjektet vil Word benyttes i de fleste settinger hvor et tekstbehandlingsverktøy behøves. Noen bruksområder vil være Prosjektplan, Rapport, Logging, Avtaler og Kontrakter, med mer. Dette er fordi prosjektteamet har best kjennskap til programvaren, samt det er et av de ledende tekstbehandlingsvektøyene på markedet.

Microsoft Excel

Microsoft Excel er et program som egner seg godt til loggføring av aktiviteter på en strukturert og oversiktlig måte. For dette prosjektet benyttes Excel til å loggføre timeliste og aktiviteter utført i tillegg til å lage Gantt diagram. Excel benyttes ettersom gruppa har god kjennskap til verktøyet, det er oversiktlig og enkelt å lære, og det er et verktøy som ofte benyttes i loggføring-aktiviteter.

Microsoft Planner

Microsoft Planner er et verktøy som kan brukes på forskjellige måter. For dette prosjektet brukes Planner til å organisere gjøremål utover i prosjektet. Denne plattformen blir brukt på grunn av den enkle måten å organisere gjøre mål i undergrupper, samtidig som vi kan holde styr på hvor i prosessen de forskjellige gjøremålene befinner seg. I prosjektet blir den også brukt til å fordele gjøremålene mellom medlemmene i gruppen.

Discord

Som kommunikasjonsverktøy internt i gruppen brukes Discord, et gratis VoIP-program (Voice over Internet Protocol) og en digital distribusjonsplattform. Fordelen med å bruke Discord er at "rom" og kommunikasjonskanaler kan kategoriseres og struktureres på en slik måte at man kan gjøre kommunikasjon og informasjonsdeling i store prosjekter mer oversiktlig enn om all kommunikasjon skal foregå i én tråd eller ett chatterom.

Figma

Figma er et grafikkredigeringsprogram, samt et prototypeverktøy som vil brukes av gruppen når eventuelt brukergrensesnitt skal designes. *Figma* lar brukere designe en interaktiv prototype som vil gjøre testing og diskutering rundt prototypen lettere.

Programmeringsspråk

Gruppen vil for programmeringsspråk, utviklingsmiljø og dokumentasjonsverktøy bruke nyeste stabile versjon. For utvikling vil ikke språkets versjon oppdateres, og nye versjoner vil ikke tas hensyn til der det er forsvarlig. Dette er for at utviklingsprosessen skal holde seg stabil og forutsigbar.

5.2 Plan for inspeksjoner og testing

Inspeksjoner

Under oppstartsfasen av prosjektet vil all dokumentasjon bli kontrollert av alle gruppemedlemmene, tekster skal leses over og vurderes saklig. Statusrapporter vil også bli skrevet for så å bli vurdert av veileder. Gruppen skal være kritiske til kilder og om en kilde virker upålitelig skal dette undersøkes av gruppen før kilden tas i bruk, som nevnt over vil gruppen bruke *Words* integrerte referanseverktøy.

Testing

Under utviklingsprosessen vil programmeringsspråkets test-rammeverk bli brukt for all funksjonalitet. Dette er både for å lett kunne forsikre oss om at funksjoner fungerer som de skal, samt at det vil brukes minimalt med tid på manuell testing.

Det vil testes progressivt igjennom utviklingsprosessen. Det vil tidlig i utviklingsprosessen bli brukt Enhetstesting (Unit testing). Etter hvert som produktet utvikles til et større, mer komplett og sammensatt system vil det bli laget mer kompliserte og omfattende tester som tester funksjonalitet på tvers av funksjoner. Ved å bruke denne metoden kan vi forsikre oss om at tidligere funksjonalitet vil fungere som planlagt før vi går videre med mer komplekse tester. Hvert aspekt av løsningen skal testes på denne måten: Database, API, samt mobilløsning vil alle ha sine egne testfaser.

Gruppen vil prøve å ha grunnleggende brukertester med søkelys på mobil løsningen og dens brukervennlighet, men gitt den pågående koronasituasjonen kan dette bli vanskelig. Det vil også bli brukt «wireframes» før brukergrensesnittet til mobil applikasjonen lages. Dette er for å kunne få tilbakemelding tidlig på løsningens design og brukervennlighet da dette er sentralt for oppdragsgiv

5.3 Risikoanalyse på prosjektnivå

Svært alvorlig / Svært sannsynlig

Alvorlig / Sannsynlig

Noe alvorlig / Noe sannsynlig

Litt alvorlig / Litt sannsynlig

Ikke alvorlig / Usannsynlig

Nr.	Hva kan gå galt?	Sans	Alvor	Hva er konsekvensen?	Hvordan forhindre?	Hvordan redusere konsekvensene?
1	Det finnes ingen sporingsenheter per dags dato som imøtekommer oppdragsgivers behov og ønsker.			Systemet bygges ikke på en spesifikk enhet. Ingen faktisk enhet å teste ut systemet på.	Gjøre grundig undersøkelse. Være realistisk med hensyn på dagens- og tilgjengelig teknologi.	Se på alternative løsninger til hvordan systemet kan testes (prototype) uten en sporingsenhet som imøtekommer oppdragsgivers ønsker.
2	Programmerings-språket er ikke egnet til formålet.			Dette vil gå på bekostning av systemarkitektur, og det vil gjøre implementasjon vanskelig og tidkrevende.	Bruke god tid på valg av programmeringsspråk samt undersøke tidligere løsninger med lignende funksjonalitet.	Undersøke språkets bibliotek grundig og for å forhøre seg med faglærere som har kunnskaper i språket.

3	Dokumentasjon og kode kan gå tapt ved tap av enheter og utstyr.			Arbeidsmengden øker, kvaliteten på arbeidet vil etter all sannsynlighet svekkes.	Lagre ting på steder som ikke er tilgjengelig på kun én enhet. Eksempelvis benytte cloud services og versjonskontroll i GitLab.	Se tilbake på prosessen fra første gang det ble gjort og unngå å gjøre de samme feilene igjen. Fokusere på nødvendig funksjonalitet og dokumentasjon fremfor å gjøre alt "halvveis".
4	Sikkerhetshull i et bibliotek eller system som benyttes i prosjektet.			Vil dette forekomme må biblioteket byttes ut og gruppen vil måtte implementere funksjonalitet på nytt.	Undersøke biblioteker på forhånd samt holde seg oppdatert på bibliotekets status igjennom prosjektet.	I det tilfellet hvor et bibliotek blir hacket vil det å programmere med sikkerhet som fokus kunne redusere de potensielle skadene biblioteket kan gjøre.
5	Ikke få godkjent appen og system for utgivelse.			Dette vil resultere i at testing og distribusjon blir vanskeligere.	Undersøke dette tidlig for å kunne sende appen til publiseringsselskaper.	Om applikasjonen ikke blir godkjent må gruppen finne en annen metode å deployere appen på.
6	Store uenigheter innad i gruppa eller at gruppa går i oppløsning.			En prosjektkritisk hendelse som vil resultere i ustabilt arbeidsmiljø og dårlige resultater.	Åpen dialog og ta konflikter når de er relativt små.	Prøve å komme til enighet om et kompromiss som imøtekommer alle medlemmers interesse i så stor grad som mulig.
7	Oppdragsgiver mister interesse.			Utviklingsprosessen følges ikke opp av oppdragsgiver. Ettersom vi benytter en SCRUM utviklingsmetode vil ikke utviklingen få den inputen SCRUM	Inkludere oppdragsgiver i utviklingen og prosessen, samt holde	Benytte grundig undersøkelse for å få en bedre forståelse av fagfeltet det utvikles for.

				behøver, som gjør at metodologien vi benytter ikke blir praktisert på ønsket måte.	oppdragsgiver oppdatert ukentlig.	Ellers kan man prøve å få tilbake interessen fra oppdragsgiver gjennom oftere og/eller mer utfyllende møter.
8	Corona situasjon			Ikke mulig å besøke oppdragsgiver. Veiledere og gruppemedlemmer kan ende i karantene.	Følge regjeringens retningslinjer og unngå unødvendig kontakt.	Ta i bruk digitale kommunikasjonsmiddel tidlig for å være forberedt og vedlikeholde kvaliteten til tross for fysisk fravær.
9	Sykdom			Gruppemedlemmer kan ikke jobbe med prosjektet på samme måte.	Holde seg hjemme ved sykdom for å forhindre at andre gruppemedlemmer smittes.	Finne alternative måter å jobbe på til tross for at man ikke kan møte gruppa fysisk. Ta igjen tapt arbeidstid når man blir frisk, eller distribuere tidskritiske arbeidsoppgaver til de andre gruppemedlemmene.
10	Noen andre har utviklet systemet før eller lanserer et slikt produkt/system i løpet av			Oppdragsgiver vil potensielt ikke ha bruk for systemet som produseres.		Ikke ta for mye hensyn til hva andre lanserer, men benytte disse til inspirasjon. Prøve å finne en måte å skille ut vårt produkt fra deres.

	prosjektperioden.					

Plan for gjennomføring

Mylift Oppgave

Milepæler

Beslutningspunker

1.2022

TASK	PROGRESS	START	END
Oppstart			
Oppsett av verktøy	100 %	1.11.22	1.15.22
Sporingteknologi	90 %	1.11.22	1.25.22
Prosjekt Plan	70 %	1.11.22	1.30.22
Løsning av sporer	10 %	1.25.22	2.1.22
Nødvendige modeller (DB, DFD, UC)	0 %	1.30.22	2.4.22
Satusrapport	0 %		
Database			
Design	0 %	2.4.22	2.11.22
Implementasjon	0 %	2.11.22	2.25.22
Test	0 %	2.11.22	2.25.22
Deployment	0 %	2.25.22	2.27.22
API			
Design	0 %	2.21.22	2.24.22
Implementasjon	0 %	2.24.22	3.13.22
Test	0 %	2.24.22	3.13.22
Deployment	0 %	3.13.22	3.14.22
Webapplikasjon			
Design	0 %	3.14.22	3.21.22
Implementasjon	0 %	3.21.22	4.14.22
Test	0 %	4.14.22	4.19.22
Deployment	0 %	4.19.22	4.20.22
Mobilapplikasjon			
Design	0 %	3.14.22	3.21.22
Implementasjon	0 %	3.21.22	4.14.22
Test	0 %	4.14.22	4.19.22
Deployment	0 %	4.19.22	4.20.22
Sluttfase			
Rapport	0 %	2.1.22	5.20.22

Milepæler:
5. Februar: Finne sporingsteknologi og sporer.
27. Februar: Ferdig med database og database design.
14. Mars: Ferdigstille API slik at implementering av Mobil og Web løsning kan starte.
20.April: Ferdig med implementasjon

Beslutningspunkter:
4. Februar: Valg av sporingsteknologi og enhet.
11. Februar: Beslutte programmeringsspråk.
23. Februar: Beslutte API funksjonalitet.

7 Referanser

Engberg, Daniel. *AGDIWO.* 28 09 2021. https://www.hmsmagasinet.no/byggebransjen-digitaliseringmiljo-og-baerekraft/bygg--og-anleggsbransjen-trenger-fokus-pa-framtiden/234474 (funnet 01 18, 2022).

GitLab. *gitlab.com.* 18 01 2022. https://about.gitlab.com/company/ (funnet 01 18, 2022).

HMS Magasinets Redaksjon. *HMS Magasinet.* 21 05 2019. https://www.hmsmagasinet.no/byggebransjen-digitalisering-miljo-og-baerekraft/bygg--oganleggsbransjen-trenger-fokus-pa-framtiden/234474 (funnet 01 18, 2022).

Nikolaieva, Aliona. *Up Tech.* 17 01 2022. https://www.uptech.team/blog/software-developmentmethodologies (funnet 01 21, 2022).

Norges Teknisk Naturvitenskapelige Universitet. *NTNU.no.* 21 01 2022. https://www.ntnu.no/studier/emner/IDATG2900#tab=omEmnet (funnet 01 21, 2022).

World Economic Forum. *World Economic Forum.* u.d. https://reports.weforum.org/digitaltransformation/understanding-the-impact-of-digitalization-on-society/ (funnet 01 26, 2022).

Appendix E Kontrakt

NTNU

Norges teknisk-naturvitenskapelige universitet

Fastsatt av prorektor for utdanning 10.12.2020

STANDARDAVTALE

om utføring av studentoppgave i samarbeid med ekstern virksomhet
Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

Forklaring av begrep

Opphavsrett
Er den rett som den som skaper et åndsverk har til å fremstille eksemplar av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

Eiendomsrett til resultater
Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

Bruksrett til resultater
Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

Prosjektbakgrunn
Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

Utsatt offentliggjøring
Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt: Institutt for datateknologi og informatikk
Veileder ved NTNU: Frode Haug e- post og tlf.

Ekstern virksomhet: Ekstern virksomhet sin kontaktperson, e-post og tlf.:
Student: Tormod Mork Müller Fødselsdato: 31.03.1999
Student: Aleksander Aaboen Fødselsdato: 29.10.2000
Student: Martin Iversen Fødselsdato: 19.12.1999

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bacheloroppgave	x
Prosjektoppgave	
Annen oppgave	
Startdato: 11.01.2022	
Sluttdato: 20.05.2022	

Oppgavens arbeidstittel er: Sporing av stillas

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven:
Nødvendig programvare, maskinvare og komponenter som vil brukes til sporing av stillas og for å imøtekomme oppgavens behov, samt tilgang til nødvendige IT ressurser selskapet har fra tidligere.

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

4. Studentens rettigheter

Studenten har opphavsrett til oppgaven [1]. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

[1] Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 SS I

5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten skal utnytte resultater som inkluderer den eksterne Sjn prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

Alternativ a) (sett kryss) Hovedregel

x	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

Alternativ b) (sett kryss) Unntak

x	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene:

Siden ekstern virksomhet betaler for utgifter knyttet til oppgaven, skal ekstern virksomhet ha overført eiendomsretten til resultatet.

6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven 5 7. Fristbestemmelsene i 5 7 gis tilsvarende anvendelse.

7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

X	Oppgaven skal være offentlig

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Oppgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss		Sett dato
	ett år	
	to år	
	tre år	

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, bJjr enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

9. Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

Signaturer:

Instituttleder:
Dato:

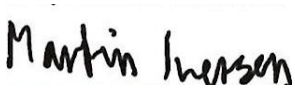
Veileder ved NTNU:
Dato:

Ekstern virksomhet: 21. 01. 2022 Rindal
Dato:

Student: Tormod Mork Müller "fonwoa Müller
Fødselsdato:31.03.1999

Student: Aleksander Aaboen leksander Aaboen
Fødselsdato: 29.10.2000

Student: Martin Iversen
Fødselsdato: 19.12.1999



Appendix N. Møtereferater

Møtereferater

Innhold

Appendix F Møtereferater

Møter med veileder NTNU, Frode Haug

11.01.2022 - spørsmål angående Lynkurs og generelt om dokumentasjon

Sted: Zoom
Klokkeslett: kl. 14.00
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Diskusjon innholdet i Lynkurset, samt generell info om skriving av bacheloroppgaven.
- Innhold i prosjektplanen.

Tilbakemeldinger:

- Ha fokus på software utvikling framfor hardware spesifikasjoner på sporingsenheter. Lurere å fokusere på at det skal være lett å bytte ut brikken med brukergrensesnittet.
- Fokuser på hva vi skal lære og lage, ikke bry oss om hva som potensielt lages av andre under bacheloroppgave perioden.
- 1. mai begynner det å bli kritisk med fokus på praktisk arbeid vs. rapport skriving.
- Sende første utkast av prosjektplanen til Frode ca. 25. Januar.
- 24 timers arbeidstimer varsel med tanke på lesing av tekst før møter eller ved spørsmål før møter etc.
- Behøver ikke benytte LaTeX som skriveverktøy.
- Bruk de første 14 dagene på å finne ut *hva* skal lages, ikke *hvordan*.
- Kontrakt med arbeidsgiver skal lastes inn i BlackBoard.
- Lag grupperegler og gruppe-kontrakt.
- Benytter tre statusrapporter istedenfor statusmøter.
    - Ved behov diskuteres innholdet i statusrapporten i neste møte.
    - Første rapport leveres inn ca. 20. februar.
    - Andre rapport leveres inn ca. 1. april.
    - Tredje rapport leveres inn ca. 1. mai.
- Anbefales å skrive rapporten og dokumentasjon på norsk.
- Kode og kodekommentering kan derimot være på engelsk.
- Møte med Frode er planlagt å avholdes mandager 14.30.
    - Benytter zoom inntil videre, men bytter til fysiske møter så fort det lar seg gjøre.
- Log fra uke til uke eller dag til dag hva som vurderes, velges og skrotes av ideer etc.

17.01.2022 - spørsmål angående prosjektplan

Sted: Zoom
Klokkeslett: kl. 14.30
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Spørsmål angående prosjektplanens innhold.
- Diskusjon rundt møtetider 24. januar og 31. januar.

Tilbakemeldinger:

- Bruken av engelske begreper er greit om det ikke finnes et godt alternativ på norsk. Hovedregelen er at begreper vi kunne før jul kan brukes fritt, men ord, begreper og forkortelser spesifikt for denne oppgaven eller som ikke er allmennkunnskap for en tredje års dataingeniør student bør defineres og beskrives i en ordliste.
- Vær nøye på henvisning til vedlegg at disse er korrekt.
- Møte mandag 24. januar kl. 14.30 flyttes til tirsdag 25. januar kl. 08.30.
- Møte mandag 31. januar kl. 14.30 flyttes til mandag 31. Januar kl. 11.00, fysisk på Frodes kontor.

25.01.2022- Tilbakemelding på prosjektplan

Sted: Zoom
Klokkeslett: kl. 08.30
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Tilbakemelding på prosjektplan

Tilbakemeldinger:

- Bra
- Gjenta ting i omfang da dette skal i rapporten
- Husk å lese over når flere skriver i samme seksjon (mye gjentakelser, ingen rød tråd etc)
- Vær litt selektiv med kilder

31.01.2022 – Ingen spesiell agenda

Sted: Fysisk, Frodes kontor
Klokkeslett: kl. 11.00-11.15
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Tilbakemelding på prosjektplan denne er nå ferdig

Tilbakemeldinger:

- Ingen spesielle momenter
- Vi får tilsendt lynkurs rapport tidlig

15.02.2022 – Ingen spesiell agenda

Sted: Digtalt,zoom
Klokkeslett: kl. 11.00-11.15
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

- Frode liker gruppens plan for løsning
- Ikke legg fokus på pris da det ikke er sikkert at det finnes enheter med planlagt pris (70 kroner)
- Research er bra, kan ikke bli for mye av det, dette skal tross alt med i rapporten
- Må kanskje endre på oppgavebeskrivelsen

28.02.2022 – Ingen spesiell agenda

Sted: Digtalt,zoom
Klokkeslett: kl. 14.00-14.30
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

- Produkt og rapport teller ikke bare rapporten
- Rapporten er viktigst*
- MyLift har ikke så mye å si på oppgaven
- Sammenlign gammelt gant diagram med hvordan det faktisk gikk

02.03.2022 – Lynkurs i rapportskriving

Sted: 2/3 Eureka
Deltakere Tormod og Aleksander

Generelt

- Dersom vi har figurer og tabeller, pass på at de blir nummerert
  - Avklar med opphaver om vi kan bruke tabeller osv, i rapporten
    - Henvis med kilde.
- Vurdere selv om vi skal ha topp- og bunntekst
  - Kapitler og sidenummerering
  - Ikke nødvendig med navn på gruppe medlemmer osv.
- Lengde; antall sider = 40 + (antall studenter * 10), uten vedlegg
- Pass på å ikke skrive muntlig.
  - Les hverandres kapitler, slik at vi skriver på en lik måte med tanke på språk og forkortelser.
- Kan være lurt å lese andre gruppers rapporter, slik at vi kan gi feedback til hverandre osv.
- Kan sende til foreldre og andre, slik at vi kan få feedback
  - Vanligvis dersom de har noe kompetanse innenfor fagfeltet.

Punkter fra Tormod

- Info fra vedlegg, ta inn viktige deler i rapporten ettersom vedlegg ikke nødvendigvis leses
- Lag utkast til rapport og lever til veileder tidlig så du får tilbakemelding
- Vektlegg tyngre det vi er stolt over og mener er viktig framfor den malen Frode ga med sidene. Men bruk antydningen for det dem er verdt
- Presentasjon 7-9 juni, på campus
  - Eget lynkurs for dette.

28.02.2022 – Ingen spesiell agenda

Sted: Digtalt,zoom
Klokkeslett: kl. 14.00-14.30
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

- Produkt og rapport teller ikke bare rapporten

- Rapporten er viktigst*
- MyLift har ikke så mye å si på oppgaven
- Sammenlign gammelt gant diagram med hvordan det faktisk gikk

07.03.2022 – Ingen spesiell agenda

Sted: Digtalt,zoom
Klokkeslett: kl. 14.00-14.30
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Diskusjon rundt valg av sporingsteknologi gruppen må komme til en beslutning

Tilbakemeldinger:

- Gruppen jobber godt men må finne sporingsteknologi

14.03.2022 – Ingen spesiell agenda

Sted: På skolen, Frodes kontor
Klokkeslett: kl. 14.30-15.00
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Gruppen diskuterer hva vi skal gjøre om enheten ikke egner seg (møtet med Telia er i morgen)

Tilbakemeldinger:

- Veileder mener vi bør være forberedt på at enheten ikke egner seg og at vi må ha et par alternativer
- Gruppen må enten finne ny enhet, eller bruker mock data for testing

21.03.2022 – Ingen spesiell agenda

Sted: På skolen Frodes kontor
Klokkeslett: kl. 14.30-15.00
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

- Gruppen jobber med utvikling og har derfor ikke mange spørsmål eller momenter å diskutere med veileder
- Det jobbes med å finne en alternativ sporingsenhet

28.03.2022 – Ingen spesiell agenda

Sted: På skolen Frodes kontor
Klokkeslett: kl. 14.30-15.00
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

- Ingen spesielle tilbakemeldinger

04.04.2022 – Ingen spesiell agenda

Sted: På skolen Frodes kontor
Klokkeslett: kl. 14.30-15.00
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

- Ingen spesielle tilbakemeldinger

25.04.2022 – Ingen spesiell agenda

Sted: På skolen Frodes kontor
Klokkeslett: kl. 14.30-15.00
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

- Ingen spesielle tilbakemeldinger

02.05.2022 – Prosjektrapporten

Sted: På skolen Frodes kontor
Klokkeslett: kl. 14.30-15.00
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Tilbakemelding på rapportstruktur

Tilbakemeldinger:

- Teori: - Briller så mye som mulig her.
- Miljø og rammer og begrensinger i fysisk og stillas osv.
- God struktur - gjerne følge struktur til eureka pris oppgavene.
- Ikke diskuter for mye i teori, ettersom dette kan bli for tørt.
- A, B, C alternativ, valgte dette, kommer tilbake til hvorfor i seksjon 9 feks.
- Ha med alle use casene (kan godt være på 3-5 sider) 20 sider er verre. Kravspek er ok.
- Dele kravspekk i to.
- Men kikk litt på hvordan kravspekk gjøres i systemutvikling. (Evnt rapporter Tom viste til).
- Vurderinger og alternativer og konklusjoner må komme et sted.
- Men man kan konkludere litt hvor man vil.
- Hva man ender på av teknologier osv kan gjerne komme tidligere.
- Konkludere på hele prosjektet og hva man endte opp med er 4-10 linjer med kom vi i mål, er vi fornøyd etc.
- Henvis hvis det blir likt. Kan gjerne være både likt og ulikt. (Mtp prosesser i front end)
- Litt formelt språk, men «gruppen» osv er greit Kritikk: Kritikk av alt vi kommer på. Problemstilling, løsning, arbeidsmåte, hva kunne vi eller oppdragsgiver gjort annerledes.
- Kan være litt kritiske mot Telia pga miskommuinikasjon etc.

- Kap 1: Sånn er rollene Til slutt, henvise til 1, de var ikke sånn i utgangspunktet. (Kap 1 og 2 SOM DET ER NÅR DET ER FERDIG) resten litt som det gikk.
- Runnes ikke, koden sees på. Kjøres på presentasjonen og denne er viktig.

09.05.2022 – Mer prosjektrapport

Sted: Zoom
Klokkeslett: kl. 14.30-15.00
Deltakere: Frode Haug, Aleksander Aaboen, Martin Iversen og Tormod Mork Müller
Agenda:

- Tilbakemelding på prosjektrapporten

Tilbakemeldinger:

- Ordliste, definisjon og forkortelse (Appendix A) VIKTIG. — Samle opp forkortelser og definisjoner etc. Blir fort et par A4 ark for vår del - Til sist fremfor å forklare det i teksten. (Side 12)
- Viktig med korte introer til kapitlene og hva som kommer her.
- BLE er en teknologi som vi bruker til det og det… Så kommer teorien.
- En setning eller to som «intro» for å ikke se ut som klipp og lim
- Spess kap 2 med intro til hvorfor dette kommer nå
- Hyperlink I TEKSTEN er bra. - Resultat vs diskusjon først tar vi det vi mener er mest naturlig.
- Presentasjon med Tom etter levering? Spør Tom - Spør Tom om innlevering 13. mai
- Utviklingsprosessen lenger ut i rapporten? Fått tilbakemelding (Bør komme før backend etc)
- Skriv hvorfor dette kommer her - sporing - utvikling - back end - front end. - Sustainability
- Uenigheter, savnes nok ikke om det ikke er med - Om det skal med må det nevnes der det er hensiktsmessig - Evnt nevne om hardware etc.
- Omfang og dette har vært vanskelig etc ER VIKTIG. Dette ser sensor etter.
- Møte på skolen på neste mandag.

Møter med veileder MyLift (Oppdragsgiver), Knut Rindal

20.12.2021 - Bli-kjent møte med veileder i MyLift Knut Rindal

Sted: Teams
Klokkeslett: 14:00
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver)
Agenda:

- Introduksjon til selskapet
- Bli kjent med arbeidsgiver
- Forventninger ved oppgaven

Tilbakemeldinger:
Arbeidsgiver legger vekt på prosessen, ikke resultatet

14.01.2022 – Første offisielle møte MyLift

Sted: Teams
Klokkeslett: 09:00
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver)
Agenda:

- Fremlegg av hvordan vi tenker oppgaven bør løses
- Ønske om fysisk møte

Tilbakemeldinger:

- Oppdragsgiver vil ha et åpent API han kan bruke og integrere i andre løsninger
- Oppdragsgiver bruker en løsning fra armada, vil sette oss i kontakt med de
- Viktige punkter for oppdragsgiver
    - Hva går inn og ut fra lager, antall ulike deler
    - Hvilket prosjekt er deler knyttet til
    - Soner knyttet til varslinger som fører til forslag om fakturering etc
    - Mindre kommer inn enn ut
    - Bruk av bareller
    - IOS samt Android
- Vi presenterte våre ideer for Knut:
  Web løsning med diverse funksjonalitet
  Mobil app som kan brukes «På gulvet»

Kommentarer fra Knut:
Åpent API er viktig, skal kunne brukes av flere systemer
Vil ha oss i kontakt med Armada som har lagd it løsninger for de før
Kontakt med fremoverlent tysk selskap
Bareller

Aspekter Han mener er viktig:
Inn og ut av lager
Hvilket prosjekt
Sone som gir beskjed til kunder når varer går ut av lager
Sone gir beskjed til de når varer kommer tilbake, fører til forslag om fakturering eller ikke
Automatikk

21.01.2022 Ingen spesiell agenda

Sted: Teams
Klokkeslett: 09:30-10:15
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver)
Agenda:

- Innspill fra Knut
- Statusoppdatering

Tilbakemeldinger:

- Fått kontaktinfo med samarbeidspartnere
- Vi får frie tøyler, ønsker «nye øyne» på problemet
- Igjen det med å kunne bygge på et API
- Haki er hovedleverandør(vi kan få tegninger herfra)
- Vil ha oss med til hamar når mulig
- Mobilapp for ansatte internt i selskapet
- Lage løsning for lageret?
- Hvorfor har mylift oppgaven
    - Hvor er stillaset mitt
    - Feil ved innleid mannskap er stor
    - Fokus på lagertelling
- Overføring internt imellom byggeplass bør være fokus
- Gang i mylifts greier:
    - Tegninger sendes
    - Stillas planlegges
    - Kunde godkjenner evt endringer
    - Stillas plukkes og kjøres ut i paller med karmer
    - Monteres og demonteres
    - Sendes tilbake til lager
- Stillaset behandles røft
- Deler ut og deler inn er viktigst

28.01.2021 – Presentering av sporingsteknologier

Sted: Teams
Klokkeslett: 09:00 - 10:00
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver)
Agenda:

- Presentering av research angående sporingsteknologier
- Statusoppdatering

Tilbakemeldinger:

- Positiv til LPWAN Dette er nytt og han har ikke sett d før
- Fornøyd
- Prøv å utfordre haki og schake litt mer

04.02.2021 – Statusoppdatering

Sted: Teams
Klokkeslett: 09:00 - 09:30
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver)
Agenda:

- Statusoppdatering

Tilbakemeldinger:

- Ingen nevneverdige punkter

11.02.2021 – Statusoppdatering

Sted: Teams
Klokkeslett: 09:00 - 09:15
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver)
Agenda:

- Statusoppdatering

Tilbakemeldinger:
Ingen spesielle tilbakemeldinger

18.02.2022 – Fremlegg for MB-Stillas i Hamar

Sted: MB-Stillas Hamar
Klokkeslett: 11-13
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver) samt daglig leder i MB-Stillas, Lageransvarlig og IT seksjon
Agenda:

- Gruppens fremdrift og planlagt løsning

Tilbakemeldinger:

- Møtenotater MyLift 18. februar:
- Besøke MyLift neste fredag - 10.00
- Logistikk er viktigst - ut og inn.
- Barelleløsning er gunstig.
- Spør når Shake kommer til Norge.
- Mobil, tablet viktigst, Web er også veldig gunstig.
- Hvem skal få beskjed når det er pakket? Og når?
- Kan den beregne hvor lenge det er til den er framme?

25.02.2022 – Fremlegg for MB-Stillas i Hamar

Sted: MB-Stillas Hamar
Dato: 25.02.2022
Klokkeslett: 11-13
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver) samt daglig leder i MB-Stillas, Lageransvarlig og IT seksjon
Agenda:

- Gruppens fremdrift og planlagt løsning

Tilbakemeldinger:
Daglig leder liker ide, momenter:

- Ønsker en sentral LPWAN enhet på byggeplass
- Spore individuelle deler (ikke de minste) med BLE tags
- Viktig å kunne integrere dette med annen programvare (API ER VIKTIG)

- Ønsker å fysisk teste systemet om mulig
- Pris kommer ikke først her, først og fremst et forskningsprosjekt
- Gutta var fornøyde

04.03.2022 – Ingen spesiell agenda

Sted: Teams
Klokkeslett: 11-13
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver)
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

- Ingen spesielle tilbakemeldinger

17.03.2022 – Ingen spesiell agenda

Sted: Teams
Klokkeslett: 11-13
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver)
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

25.03.2022 – Ingen spesiell agenda

Sted: Teams
Klokkeslett: 11-13
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver)
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

01.04.2022 – Ingen spesiell agenda

Sted: Teams
Klokkeslett: 11-13
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver)
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

08.04.2022 – Ingen spesiell agenda

Sted: Teams
Klokkeslett: 11-13
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver)
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

29.04.2022 – Ingen spesiell agenda

Sted: Teams
Klokkeslett: 11-13
Deltakere: Tormod, Aleksander, Martin og Knut Rindal (Oppdragsgiver)
Agenda:

- Ingen spesiell agenda

Tilbakemeldinger:

- Mail angående besøk (haki, mylift)
- Presentasjon - Hvor plassere gatewayen?
- Ta kontakt angående Evnt testing på prosjekter de har

Møter diverse samarbeidspartnere

Møte HAKI 25.01.2022

Sted: Teams
Klokkeslett: 09:30-10:15
Deltakere: Tormod, Aleksander, Martin, Siri Røgeberg, Odd Henning Bratli, Dagfinn Mundal
Agenda:

- Innspill fra Haki

Tilbakemeldinger:

- Har jobbet litt med sporing (Studenter) rapporter skal sendes om de er tilgjengelige
- Forutsetninger:
  - Kostnad (vi ligger på rundt 10-20 kr per enhet)
  - Håndtering (stillas nærmest kastes rundt på byggeplass/lager)
- Inn og ut av lager er fokus
- Helst ettermonteres
- RFID har hatt problemer med signaler tidligere
- Besøk haki
- 6-10 komponenter hovedsakelig

Møte Armada 27.01.2022

Sted: Teams
Klokkeslett: 09:00-09:20
Deltakere: Tormod, Aleksander, Martin, Henrik Eimot
Agenda:

- Introduksjon med armada
- Hva de leverer
- Hva de ønsker av oss

Tilbakemeldinger:

- SQL Relational database
- Armada ønsker all data fra et fellesområde
- Har utviklere vi kan snakke med

Møte HAKI 08.02.2022

Sted: Drammen Haki Lager
Klokkeslett: 10:00-11:30

Deltakere: Tormod, Aleksander, Martin, Siri Røgeberg, Odd Henning Bratli, Dagfinn Mundal
Agenda:

- Se på stillasdeler
- Få evt ekstra informasjon

Tilbakemeldinger:

Møte Telia 07.02.2022

Sted: Teams
Klokkeslett: 09:30-10:15
Deltakere: Tormod, Aleksander, Martin, Telia konsulenter
Agenda:

- Diskusjon rundt lpwan

Tilbakemeldinger:

- Telia ønsker å være involvert
- Kan levere skyløsningen
- Mener at deres gateway kan brukes, men vi må finne små billige tags som snakker med gatewayen

Møte ABAX 17.02.2022

Sted: Teams
Klokkeslett: 13:00-14:00
Deltakere: Tormod, Aleksander, Martin, ABAX salgsansvarlig
Agenda:

- Introduksjon med abax
- Info om mini2 og deres nettverk

Tilbakemeldinger:

- De anbefaler RFID på detaljer
- 30 kroner per mini i måneden
- Deler må komme fra kina om fokus er økonomi

Møte med Rune 09.03.2022

Sted: Runes kontor
Klokkeslett: 11:00-11:30
Deltakere: Tormod, Aleksander, Martin, Kristoffer Tegner
Agenda:

- Diskusjon rundt mySql og noSql

Tilbakemeldinger:
Sql

- Ofte brukes når mange skal endre på data
- Favoriserer ikke en database i forhold til den andre

NOSQL

- Dokument strukturen kan bli ubrukelig til senere bruk
- Referanser
    - Alle operasjoner som fjerner noe, hva refererer til dette?

- Alle som blir fjernet blir renset.
  -Typ on cascade delete

-Må ikke være normalisert.

- Toveis referering med id
  - Sørge for at det blir oppdatert
- Collections er mer abstrahering.
- Ingen enkelt svar på oppbygging, med tanke på det er forskjellige måter å bygge databasen på.
- Hvor finvasket er APIet til Telia?
- Vanlig at Databasen er lik API et.
- Typiske løsninger på utvidelser, er at man begynner å jobbe på en ny database.
- Kan ha flere databaser ettersom det som er nødvendig av behov.
- Designet vi gjør nå, kan hende får konsekvenser til senere (DISKUTERE I RAPPORTEN)
  - Valget vi gjør, er basert på resonnement
  - Hva gjør vi for å minimere utfordringene ved valgene.
- Må sikre at strukturen er riktig
  - Dette går på integritet
    - Sjekke om ting er null
    - Sjekke at strukturen er slik vi forventer.
- Prøve å ikke være så bundet til skjemaet.
  - Dersom det kommer nye element som ikke har vært der tidligere.

- Støtte for atomiske operasjoner innenfor et dokument
  - Men det er ikke støtte for dette mellom forskjellig collections
    - Dette er ikke noe vi må henge oss opp til ettersom det ikke er så mange som bruker systemet.

-Viktig å logge funksjonaliteten
  - Dersom noe blir avbrutt midt i operasjonen.
- Vedlikeholds programmer
  - Dersom noe skjer
  - Her skal vi lage et verktøy som retter opp i feil.
  - Lett å gjøre feil, veldig mye krevende
    - Lettere å lage et program som finner inkonsekvente feil, og presenterer det til brukeren.
      -Brukeren kan da rette opp feilen.
- Struktur
  - Fornuftig å lage en modell struktur
    -Hva er den fornuftige strukturen på databasen.
  - Forlanger ikke en spesiell oppbygging.
  - Lurt å dokumentere til rapporten
    - Også når vi argumenterer / diskutere design valgene.

- Programmeringen
  - Må programmere inn integritet sjekker

Rapport

- Snakke med Frode om:

-Det som er viktig, leseren kan forstå og verdsette det vi har gjort!
- Ikke nyttig for brukeren å lese det vi har skrapet
- Kan ta med SQL research for å argumentere om hvorfor vi byttet.

A og B
- Hvor mye vi kan analysere
- Hvor selvstendig vi jobber.
    - Må bearbeide det av informasjonen vi bruker.

Møte med Telia 09.03.2022

Sted: Teams
Klokkeslett: 11:00-11:30
Deltakere: Tormod, Aleksander, Martin, Kristoffer Tegner
Agenda:

- Introduksjon med abax
- Info om mini2 og deres nettverk

Tilbakemeldinger:

- 1200 kroner per enhet
- Vi kan få tilgang på API
- Starterkit inneholder 5 enheter
- Pris kan diskuteres om volum er stort nok

Møte med Telia 15.03.2022

Sted: Teams
Klokkeslett: 11:00-11:30
Deltakere: Tormod, Aleksander, Martin, Kristoffer Tegner og konsulent Siri
Agenda:

- Info om enhet

Tilbakemeldinger:

- Gateway kan kun kommunisere med omtrentlig 5 tags.
- Med tags behøves trolig en annen gateway.
- Device messaging
- leverandør som matcher så kan vi sende over kort og se hva som skal til for å kommunisere med plattformen.
- IoT Tracking indoor
- Aktuelt ved et Evnt kundeforhold, men
- API støtter ikke å hente ut data fra tags
- Tags og gateways sammensetning må hardkodes.

Appendix G Sporingsteknologier

Innhold

Introduksjon

Dette dokumentet vil inneholde all forskning/undersøkelse rundt forskjellige sporingsteknologier, samt løsninger på oppdragsgivers problemstilling. MB-Stillas sin problemstilling omhandler bedriftens mangel på logistikk rundt utleieprosessen av stillas. Det er per dags dato manglende kontroll på elektronisk varetelling av stillas verken da det blir kjørt ut fra lager, når stillas blir supplert til byggeplasser, når stillasdeler blir flyttet internt imellom byggeplasser eller når stillaset kommer inn på lager igjen.
For å kunne løse dette problemet ønsker oppdragsgiver å finne en billig måte å spore stillasdeler, dette er hovedsakelig for å oppnå en oversikt over hvor stillasdeler befinner seg, både for å kunne minke svinn og spare tid. Oppdragsgiver har gitt gruppen frie tøyler til hvordan vi vil løse dette, men både batteritid og pris er viktige momenter for løsning av oppgaven.

Vi vil i dette dokumentet ta for oss diverse sporingsteknologier som kan brukes for å løse oppgaven. Vi vil se på både sporingsteknologien (Bluetooth, RFID, etc.), hvordan den kan anvendes i vårt bruksområde, samt hvilke sporingsenheter som er tilgjengelige i dag. Dokumentet vil brukes for å dokumentere forskningsprosessen, samt brukes som et grunnlag for hvilken sporingsteknologi gruppen velger å utvikle med hensyn på. Oppdragsgiver har som nevnt over en del aspekter som er viktige for sporingsenheten, disse er:

- Sporings-enheten må fungere uavhengig av klima.
- Oppdragsgiver ønsker minst mulig menneskelig interaksjon med den «fysiske sporingsløsningen».
- Batteritid må være god for at løsningen skal lønne seg (batteri-bytting tar ofte tid og er dyrt).
- Oppdragsgiver ønsker å kunne identifisere stillas på byggeplassen.

Gruppen skal etter beste evne finne en sporingsteknologi og/eller en kombinasjon av teknologier som oppfyller så mange av disse kravene som mulig.

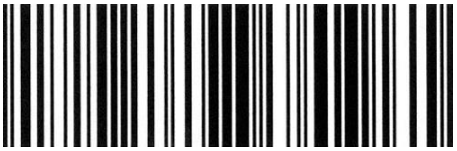
Sporingsteknologi

Denne delen av dokumentet vil gi leseren en oversikt over de forskjellige sporingsteknologiene som er tilgjengelige i dag, hvordan de fungerer, samt fordeler og ulemper ved hver enkelt teknologi. Følgende teknologier vil bli diskutert:

- Strekkode
- GPS sporing
- NFC – Near Field Communication
- UWB – Ultra WideBand
- QR-Koder
- RFID – Radio Frequency Identification
- BLE – Bluetooth Low Energy
- LPWAN – Low Powered Wide Area Network

Disse er alle teknologier gruppen mener at kan brukes til å løse problemet forbeholdt at kompromisser inngås i forhold til oppdragsgivers ønsker, men kun et fåtall vil vise seg å være særlig egnet.

Strekkode



Strekkode er en identifikasjonsteknologi som representerer data i et maskin-leselig format. Teknologien krever en skanner som leser av koden med infrarødt lys for å oppdatere informasjon. Dette er en svært billig form for sporingsteknologi da den kun krever strekkoden, samt en skanner.

Fordeler	**Ulemper**
Ekstremt billig	Ingen kontinuerlig sporing
Rask informasjonshenting	Krever fysisk skanning
Pålitelig	Slitasje på koder krever bytte
Enkel å betjene	Treg mengdeskanning

Ugyldig kilde er angitt.

Denne teknologien kan brukes for å løse problemet, men en løsning vil måtte kreve mye manuelt arbeid for å oppnå oppdragsgivers ønsker. Hvis denne løsningen skulle tas i bruk måtte monteringsansvarlig skanne hver stillasdel da delene ankommer byggeplass, på denne måten vil systemet vite når deler ankommer byggeplassen, men dette vil ta mye tid gitt at det er snakk om ekstremt mange stillasdeler.

Oppdragsgiver vil heller ikke kunne vite akkurat hvor på byggeplass stillaset er med mindre monteringsansvarlig skanner hver del ved forflytning. Strekkodene er heller ikke særlig

holdbare i norske forhold, hver kode hadde krevet en form for forsterkning for å kunne overleve arbeidet.

Denne løsningen må bli kombinert med en annen sporingsteknologi for å kunne være aktuell for oppgaven. Men oppgaven vil ikke kunne løses på en god måte om kun strekkoder vil bli brukt.

QR-kode

QR-kode, også kjent som Quick Response code er en type strekkode som kan lagre store mengder data, men fremdeles fremstille denne dataen umiddelbart. Denne dataen lagres som en serie av piksler i et firkantet rutenett. Måten en QR-kode fungerer på er at den detekterer de tre firkantene i hjørnene på QR-koden og benytter disse til å vite at all informasjonen innenfor denne firkanten er en del av QR-koden. Strekkoder kan kun leses fra topp til bunn og dermed kun lagre mindre mengder data. QR-koder derimot leses fra både topp til bunn og fra høyre til venstre, noe som tillater lagring av betydelige større datamengder. Den store fordelen til QR-koder er at de kan lagre store mengder data, kan leses hurtig og kan leses av en digital enhet som en moderne mobiltelefon.

Figur 71 ***Ugyldig kilde er angitt.***

Fordeler	**Ulemper**
Krever ikke strøm	Fungerer dårlig i mørket og i dårlig vær
Ved riktig implementasjon blir den robust	Kan ikke spores aktivt uten menneskelig interaksjon
Nærmest uendelig med kombinasjoner	Kan ikke loggføre store volumer på kort tid
Rask tilgang på informasjon	
Kan lagre en stor mengde data kontra strekkode	

For sporing av stillas imøtekommer QR-kode behovet for en robust enhet med lang levetid ettersom den ikke har behov for strøm, og kan inngraveres i metall. Problemet oppstår derimot ved behovet for aktiv sporing, samt hurtig loggføring av et stort volum stillas.

GPS-sporing

Denne navigasjonsteknologien bruker antenner og satellitt til å estimere sporingsenhetens posisjon i forhold til oppsatte antenner. Enheten sender koordinater og tidspunkter til systemet og kan brukes for å estimere spesifikk posisjon. Denne teknologien ville fungert bra for vårt bruksområde, ettersom den sender kontinuerlige oppdateringer til systemet. Enheten bruker en intern klokke for å oppdatere posisjon i forhold til fire antenner, dette gir mottakeren et x-koordinat, y-koordinat, z-koordinat og delta-feil i mottakerens interne klokke. Mottakeren bruker denne informasjonen til å estimere sin posisjon. Teknologien kan potensielt gi veldig nøyaktig posisjon overalt til enhver tid avhengig av byggekvalitet og område, men er stor, har høy kostnad og relativt dårlig batteritid om prisen skal holdes nede.

Figur 2 ***Ugyldig kilde er angitt.*** *72*

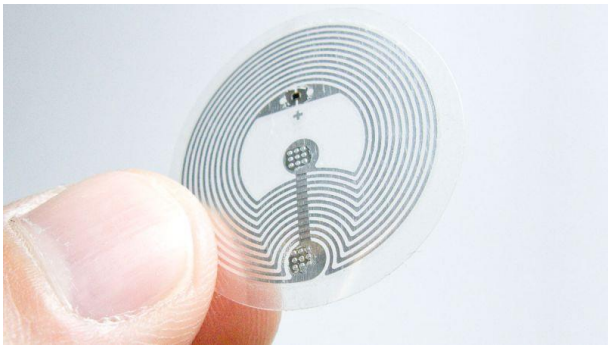
Fordeler	**Ulemper**
Nøyaktig sporing	Dårlig batteritid
Oppdateres kontinuerlig	Stor enhet
Uavhengig av "støtte" I form av forsterkere	Høy pris på sporer

Hovedproblemet med GPS sporing for vår løsning er kostnad og batteritid, GPS systemer har større kostnader enn et system som strekkoder. Gitt at hver GPS-enhet går på batteri må dette skiftes ved behov, noe som krever tid. I tillegg til dette krever mange GPS systemer minnekort som tidvis må skiftes ut. Enhetene er ofte også mye større enn for eksempel en strekkode eller RFID-tag, dette kan skape problemer gitt at enkelt stillasdeler er små, koster lite, samt blir hardt behandlet. Vedlikehold er også en faktor som fungerer dårlig for vår problemstilling, en GPS sporer per stilas-del vil ikke være lønnsomt for oppdragsgiver.

NFC – Near Field Communication

Near Field Communication, også kjent som NFC er en trådløs overføringsmetode som kan overføre små datamengder over svært små avstander. De fleste moderne telefoner kommer med støtte for NFC fra fabrikken, noe som gjør en løsning med NFC tilgjengelig uten behov for mye ekstra utstyr. Brikkene kan programmeres til å inneholde en liten mengde data som gjøres tilgjengelig for en scanner-enhet – eksempelvis en mobiltelefon. NFC-brikkene er passive, som betyr at de er ikke har behov for batteri. De benytter derimot energien utsendt

fra NFC-antennene i telefonen til å gjøre sine oppgaver. Dermed er NFC-teknologien avhengig av en ekstern enhet med en NFC-antenne for både strøm og kommunikasjon.



Figur 73 NFC Tag ***Ugyldig kilde er angitt.***

Fordeler	**Ulemper**
Behøver ikke strøm – dermed lang levetid	Kan kun leses på avstander opptil 10-20 cm
Er veldig små – kan plasseres på nærmest alle enheter	Kan kun scanne én NFC tag om gangen
Rask tilkobling og kommunikasjon	Er avhengig av scanner-enhet for å kommunisere hvor den er
Behøver kun en moderne mobil enhet for å tas i bruk	
Fungerer i mørket og i dårlig vær	

NFC teknologi og tagger imøtekommer levetid, størrelse, pris og robust aspektet ved oppgaven. I motsetning til strekkode-teknologi og QR-kode-teknologi behøve ikke NFC teknologi fysisk berøring eller å være synlig for at en mobilenhet skal greie å oppdage den. Dermed er teknologien gunstig med hensyn på bruk i mørket eller dårlig vær. At NFC kun kan leses på en avstand på opptil 10-20 cm, kombinert med at kun én tag kan leses om gangen gjør den ugunstig som eneste sporingsteknologi for sporing av stillasdeler.

BLE – Bluetooth Low Energy

Bluetooth Low Energy, også omtalt som BLE er en Bluetooth-teknologi som benyttes i stadig flere systemer og er designet for å kommunisere mindre mengder data enn tradisjonell Bluetooth Classic, for å konservere batteritid. Bluetooth Classic enheter bruker omtrent 1 watt strøm, imens en BLE enhet benytter kun 0,01-0,5 watt. Dette betyr at enkelte BLE enheter bruker 100 ganger mindre strøm. Dette oppnås blant annet av reduksjon av mengden data som sendes, samt at BLE enheter «sover» konstant til en tilkobling oppstår. Denne tilkoblingen varer i noen millisekunder, kontra Bluetooth Classics tilkobling som varer i omtrent 100 millisekunder. Teknologien kommer ikke til å erstatte tradisjonell Bluetooth, men er et alternativ for bruksområder som ønsker å sende små, spesifikke datamengder samtidig som minimalt med strøm brukes. Bluetooth tilbyr konstant kommunikasjon mellom to enheter, imens BLE kommuniserer kun i korte pulser. BLE kommuniserer dataen gjentatte

ganger gjennom flere ulike frekvenskanaler for å redusere interferensen på signalet (Fotnote??). En annen fordel som tilbys av BLE kommunikasjon er at den er kryptert med AES-128 (Fotnote??) ende-til-ende kryptering, noe som forhindrer at data kan leses av eksterne aktører om signalet fanges opp. (https://www.makeuseof.com/what-is-ble-bluetooth-low-energy/).

Fordeler	**Ulemper**
Lang levetid – bruker svært lite strøm	Avhengig av tilstedeværelse av en sentralsporer eller mobil med en bestemt applikasjon for å dele posisjon.
Tilbyr veldig små tagger med lang levetid	Tilbyr kun deling av små datamengder
Tilbyr kommunikasjon mellom enheter	Noe kostbar (varierer med krav-spesifikasjoner)
Kosteffektiv og kompatibel med veldig mye	
Lett å integrere og bruk	
Tilkoblingsrekkevidde på ca. 400m (1000+ meter utendørs i åpent landskap) med BLE versjon 5.	

Ser man på stillassporing er BLE gunstig med hensyn på levetid, størrelse, integrasjons muligheter og rekkevidde. Pris befinner seg per dags dato i en gråsone, hvor man ved et samarbeid kan skreddersy en enhet etter spesifikasjoner i stort volum, for å få prisen ytterligere ned. Basert på tidligere utvikling er det rimelig å tro at prisene vil synke samtidig som funksjonaliteten øker i løpet av de neste årene. Imens RFID kan få problemer med interferens mellom flere signaler, reduseres risikoen for dette med BLE ettersom kommunikasjonen foregår over flere og ulike frekvenser. Tilkoblingsrekkevidden er på ca. 400 meter (dette vil variere noe med ulike tagger og forstyrrelser som vegger etc.), men kan strekke seg helt opp til 1000+ meter i fri sikt med BLE versjon 5. Dermed reduseres behovet for antallet sentralsporere per prosjekt betydelig, som reduserer kostnader per prosjekt og antallet månedlige abonnement. BLE tags i kombinasjon med en gateway-løsning vil tillate kommunikasjon med omtrent 15000 BLE tags per gateway.

LPWAN – Low Power Wide Area Network

*Figur 74**Ugyldig kilde er angitt.***

Low Power Wide Area Network er et trådløst nettverk som samhandler lav båndbredde og enheter med lavt batteri forbruk. LPWAN er ikke kategorisert som én teknologi, men som en gruppe av teknologier som utnytter vidstrakt datanett teknologier som bruker lite strøm. Fellesnevneren for denne type teknologi er at de kan sende en gitt mengde med data.

LPWAN kan deles inn i to grupper. Den ene gruppen er ulisensiert og lisensiert tjeneste. De lisensierte tjenestene er NB-IoT og LTE-M. Som en mulig løsning kunne vi ha sett for oss disse løsningene på problemstillingen. NB-IoT leverer nettverkforbindelser med lav båndbredde og lavt batteri forbruk. LTE-M er teknologi som er optimalisert for høyere båndbredde og mobile forbindelser. LTE-M teknologien er best designet for mobilitet, der enheter kan bevege seg i høye hastigheter. I tillegg er LTE-M optimaliser for dataforbindelser slik at ventetiden er kortere i forhold til NB-IoT. NB-IoT er best egnet for statiske enheter. Begge teknologiene er veldig gunstige når det gjelder batteriforbruk, men NB-IoT har en fordel dersom den er satt opp riktig. (2)

Fordelen med LTE-M og NB-IoT kommer særlig til syne når man ser på delene som er nødvendig for at løsningen skal fungere optimalt. Disse teknologiene er ikke avhengig av eksterne lesere, ettersom alle enhetene er koblet opp til mobilnettet. Dette krever at hver enhet har sitt eget sim-kort. Av den grunn vil en løsning som dette bli kostbar i lengden, men den vil ha mulighet til å se hvor alle enhetene er til enhver tid.

De ulisensierte tjenestene er Sigfox og LoRa. De to tjenestene leverer samme tjenester som LTE-M og NB-IoT, der det blir utnyttet lav båndbredde. Forskjellene mellom ulisensierte og lisensierte er bruk av sim-kort. Dette betyr at en datapakke vil bli mottatt og overført til skyen av en basestasjon som er innenfor rekkevidde. Dette øker sjansene for at signalet blir plukket opp. Fordelene med ulisensierte tjenester er at de har lengre rekkevidde og enhetene bruker mindre strøm enn de lisensierte enhetene (3).

Det lave batteriforbruket kan forklares ved at enhetene kan sette i gang kommunikasjon selv. Dersom enheten vil kommunisere 100 ganger per dag eller en gang per dag, vil den gjøre det, men serveren kan ikke etterspørre et signal fra enhetene. Dette resulterer i at enheten ikke trenger å kontinuerlig vente på et signal, noe som øker batteritiden. LPWAN er svært effektiv når det kommer til å sende mindre mengder data innenfor et gitt intervall, og deretter sette enhetene i dvale (4).

Fordeler	**Ulemper**
Sterke signaler som kan penetrere konstruksjoner	Høy kostnad per sporer
Lite strømforbruk	Månedlig kostnader
Uavhengig av en mottaker	
Lang rekkevidde	
God presisjon per sporer	

Fremtidsrettet

UWB Ultra Wide-Band

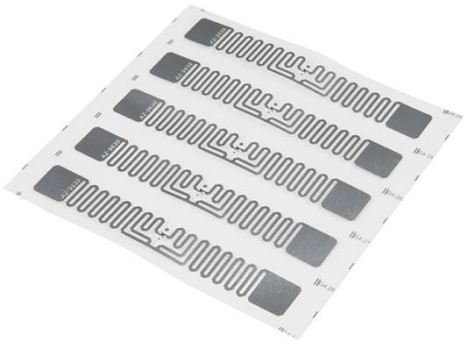
Figur 75 Apple Airtag ***Ugyldig kilde er angitt.***

Ultra Wide-band eller Ultrabredbånd teknologi er en form for radioteknologi som bruker lite energi for å kommunisere mye informasjon over korte avstander ved bruk av radiospekteret. Teknologien er aktuell for vår problemstilling siden teknologien bruker lite strøm i tillegg til at den gir posisjonen til enheten. Teknologien blir mye brukt for innendørs sporing særlig da i fasiliteter hvor andre radiofrekvenser brukes. Teknologien har dessverre for liten rekkevidde for vår problemstilling, med mindre gruppen bestemmer seg for å fokusere på lagerlogistikken alene. Om dette kompromisset inngås vil teknologien kunne tas i bruk, men i dette tilfellet vil billigere teknologier som RFID bli tatt i bruk.

Fordeler	**Ulemper**
God presisjon	Høy kostnad per sporer
Lite strømforbruk	Liten rekkevidde
Går igjennom materialer	Lav data sendings rate

RFID – Radio Frequency IDentification

Radiofrekvensidentifikasjon er en type teknologi som kommuniserer med radiofrekvens. Teknologien brukes ofte i forbindelse med logistikk. For å kunne bruke denne teknologien er du avhengig av en RFID-tag og RFID-mottaker. En RFID-tag funker ved å sende og motta informasjon via en antenne og en mikrochip. Det eksisterer to forskjellen typer RFID-tags, aktive og passive. Aktive tagger er batteridrevne, og passive tagger som ikke går på strøm.



*Figur 76 RFID Tagger****Ugyldig kilde er angitt.***

https://www.indiamart.com/proddetail/passive-uhf-rfid-tags-22301558662.html

Passive RFID-tags:

Når en passiv RFID tag er skannet av RFID-leseren, vil leseren overføre nok energi til taggen slik at chipen og antennen kan sende informasjon tilbake til leseren. Leseren vil dermed overføre informasjonen tilbake til et RFID program.

Aktive RFID-tags:

Aktive RFID-tags er batteridrevne og skal kunne holde seg i live i 3-5 år. Innenfor aktive tags finnes det beacons og transpondere.

Beacons sender ut informasjon kontinuerlig på et intervall innen noen sekunder, og deres signal er lesbart på en rekkevidde rundt 30 meter. Ettersom signaler blir sendt ut kontinuerlig, vil levetiden bli vesentlig mindre.

Transpondere krever bruken av en leser for å hente ut informasjonen. Når leseren og taggen er innen rekkevidde vil leseren sende ut et signal til transponderen, som deretter sender tilbake den relevante informasjonen. Ettersom transpondere kun er aktivert når leseren er nærme, er denne typen RFID-tags batterivennlig

Denne teknologien passer inn i oppgaven vår ved at den er billig, og vi kan få ut informasjon fra hver tag. Taggen er liten, og det vil ikke være et problem å feste den til delene. Et problem som vil dukke opp med RFID er at signalet ikke har mulighet til å penetrere metall. Dette vil bli et problem ettersom stillaset er laget av stål.

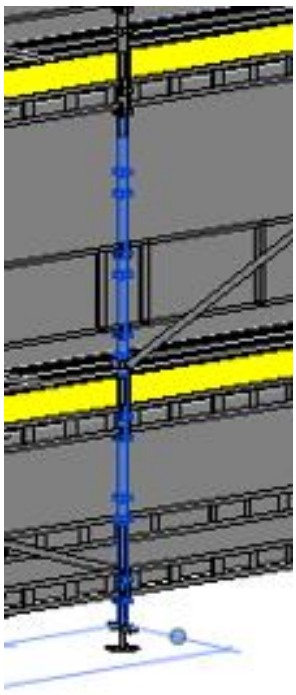
Fordeler	**Ulemper**
Billig	Må ha sender
Lite strømforbruk	Rekkevidden er under 100m
God til logistikk	

Stillas

MB-Stillas har gitt gruppen følgende prisestimat: Hver sporingsenhets totalkostnad bør ikke koste mer enn 10-20% av stillasets kostnad. Stillaset består hovedsakelig av stål, men nyere stillasdeler blir produsert i aluminium, dette begrenser rekkevidden til flere av de aktuelle sporingsteknologiene. Oppdragsgiver frakter stillaset i «bareller» som er spesiallagde kasser laget for transport av stillas. Gruppen har sett på muligheter for å kunne spore disse barellene til fordel for å spore individuelle stillasdeler. Dette vil gi gruppen et mer håndterbart budsjett. MB-Stillas produsent av stillas har gitt gruppen følgende deler:

Spir:

Dimensjoner:
Rørdiameter: 48 millimeter
Vekt 15.2 kilogram
Pris 618,00 kroner

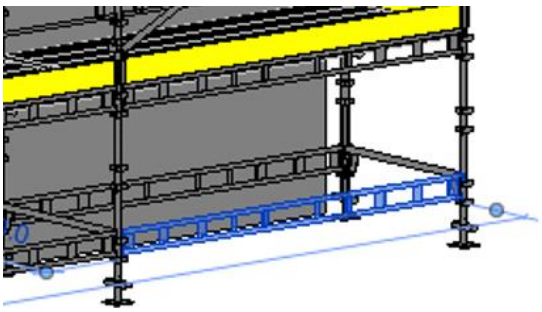


Figur 77 Spir, montering kan skje på innsiden av røret

Stillasdelen brukes som støtte og er hovedkomponent for stillas med mer enn en etasje, brukes i all montering av stillas. Plassering av sporingsenhet på denne delen er noe problematisk for alle andre enheter enn en et RFID klistremerke da dimensjonene er små og delen er rund uten flater. Plassering av eventuell Bluetooth enhet kan gjøres på innsiden av spiret, men dette vil gå på bekostning av rekkevidde. Om BLE enheten skal monteres på utsiden av spiret må et kraftig skall spesial-lages slik at enheten tåler den røffe behandlingen stillaset utsettes for under montering og demonteringsprosessen.

Lengdebjelke

Dimensjoner:
Rørdiameter 34 millimeter
Vekt 12,3 kilogram
Pris 564 kroner



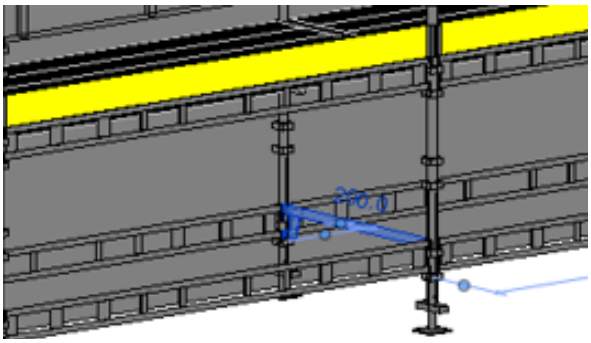
Figur 78 Lengdebjelke, montering må skje på utsiden av del

Stillasdelen brukes aktivt som støtte og bæring for plank og lemmer. Delen blir brukt på all stillasmontering uavhengig av antall etasjer og kvadratmeter. Plassering av sporingsenhet er mindre problematisk her da delen har rektangulære flater hvor både en RFID tag og en BLE tag kan plasseres.

Enrørsbjelke

Dimensjoner
Rørdiameter 48 millimeter
Vekt 5,1 kilogram
Pris 285,00 kroner

Delen brukes i alt oppsett uavhengig av etasje og størrelse. Plassering av enheten vil være noe problematisk for en BLE enhet, men plassering på utsiden av delen er eneste mulighet da røret har for tykt rørdiameter for en innvendig montering av sporingsenhet.

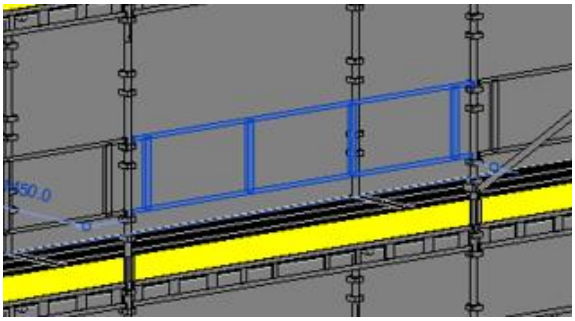


Figur 79 Enrørsbjelke, liten del montering blir problematisk

Rekkverk

Dimensjoner
Rørdiameter 28 millimeter
Vekt 10,5 kilogram
Pris 570,00 kroner

Delen brukes aktivt på alt stillas og er meget sentral, sporingsenhet kan her plasseres på rektangulær flate imellom hoved-rørene.

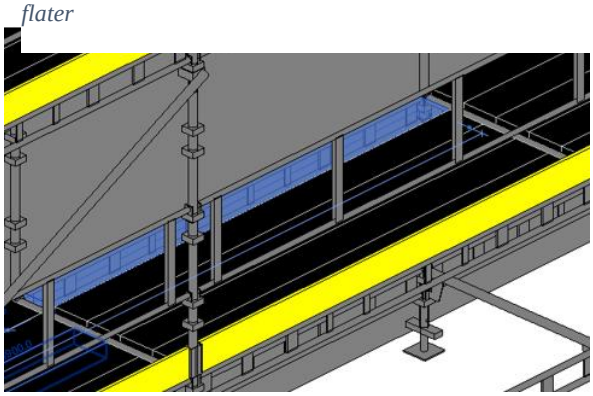


Figur 80 Rekkverk, del kan monteres på rektangulære flater

Plank

Dimensjoner
Rørdiameter Ingen
Vekt 11,2 kilogram
Pris 813,00 kroner

Delen brukes under noen stillasmonteringer, men har i nyere tid blitt erstattet med lemmer. Delen blir montert langs stillasets struktur Sporingsenhet kan monteres på undersiden av delen uten problemer.

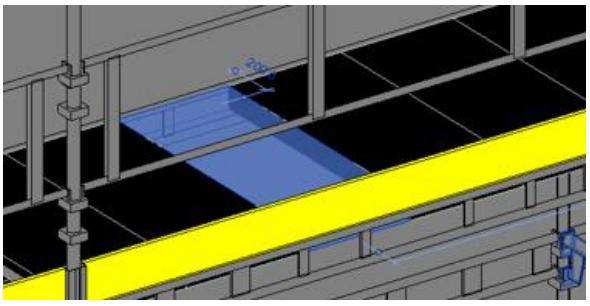


Figur 81 Plank, enhet kan monteres på underside av del

Lemmer

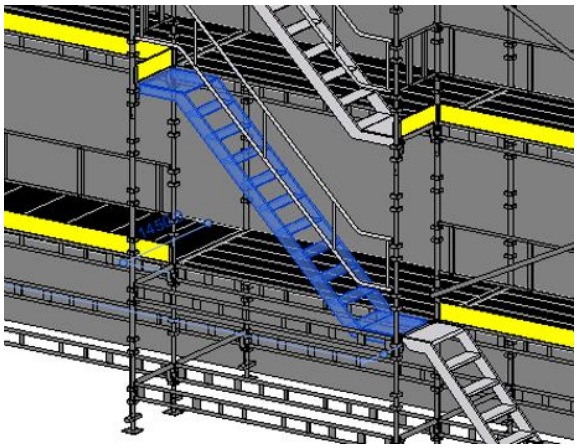
Dimensjoner
Rørdiameter Ingen
Vekt 7,8 kilogram
Pris 510,00 kroner

Delen brukes ved de fleste monteringsjobber og har i nyere tid erstattet plank, i motsetning til plank monteres denne delen på tvers av stillaset. Montering av sporingsenhet kan som på plank skje på undersiden av delen.



Figur 82 Lemmer, sporingsenhet kan monteres på undersiden av delen

UTV Trapp

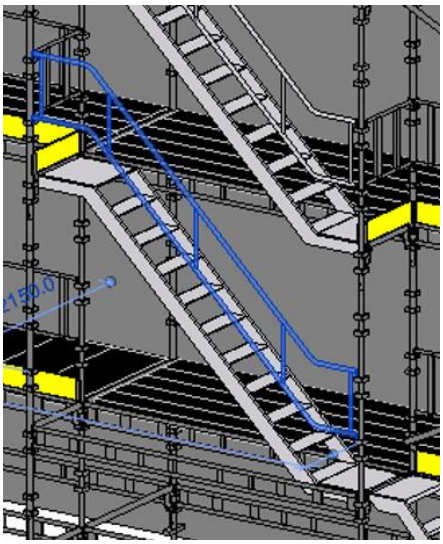


Figur 83 UTV Trapp, sporingsenhet kan monteres på opptil flere steder.

Dimensjoner
Rørdiameter Ingen
Vekt 39,7 kilogram
Pris 4 670,00 kroner

Denne delen brukes ved montering av stillas med flere etasjer. Delen i seg selv er dyr, så eventuell sentralenhet (LPWAN cellular sporer) kan plasseres her). Denne delen har opp til flere plasser hvor en BLE/RFID enhet kan monteres, men delen er ikke med på alle stillas.

Trappegelender

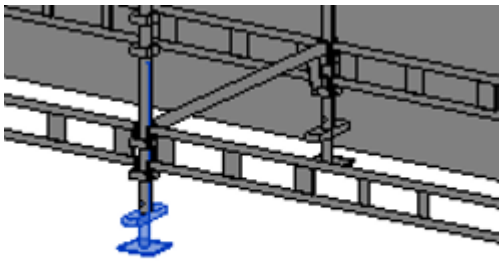


Figur 84 Trappegelender, sporingsenhet kan plasseres på rektangulær flate

Dimensjoner
Rørdiameter Ingen
Vekt 19,3 kilogram
Pris 1 639,00 kroner

Denne delen brukes i kombinasjon med UTV trapp, og er derfor ikke med på alle monteringsprosjekter sporingsenhet kan plasseres på flatt skjøte imellom øvre og nedre rør.

Bunnskrue



Figur 85 Bunnskrue, denne delen er meget problematisk gitt lav pris og liten overflate

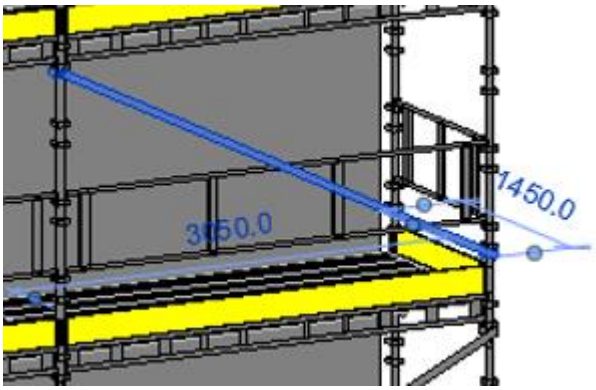
Dimensjoner
Rørdiameter Ingen
Vekt 5,0 kilogram
Pris 207,00 kroner

Denne delen brukes under all montering av stillas, den brukes som en grunnstøtte på hele stillasstrukturen. Dette er den mest problematiske delen gruppen har å jobbe med, den er ekstremt billig samt liten i forhold til andre deler. Dette gjør det vanskelig å montere noen annen form for sporingsenhet enn et RFID klistremerke.

Diagonalstenger

Dimensjoner
Rørdiameter 48 millimeter
Vekt 14,3 kilogram
Pris 637,00 kroner

Denne delen brukes for å støtte opp stillas, den blir brukt i de fleste stillas. Montering av en sporingsenhet må skje på innsiden av røret eller utenpå, men dette kan bli problematisk da delen ikke har noen flater å montere enheten på.



Figur 86 Diagonalstenger, montering av sporingsenhet må skje på rør i form av klistremerke eller på innsiden av røret

Prisestimat og vekting

Vi vil bruke to priser i refleksjonen i denne rapporten. En gjennomsnittlig pris per del (TPPD) som er en gjennomsnitts pris på alle deler vi har fått tildelt (utenom trapp da denne delen er såpass dyr, her vil enhver sporer kunne brukes) og en gjennomsnittlig pris per type del som vil være 10-20% av de forskjellige delene.

Del	**Pris**	**10-20% av pris**
Spir	618 kroner	62-124 kroner
Lengdebjelke	564 kroner	57-113 kroner
Enrørsbjelke	285 kroner	29- 57 kroner
Rekkverk	570 kroner	57-114 kroner
Plank	813 kroner	81-163 kroner
Lemmer	510 kroner	51-102 kroner
UTV Trapp	4 670 kroner	467-934 kroner
Trappegelender	1 639 kroner	164-328 kroner
Bunnskrue	207 kroner	21-42 kroner
Diagonalstenger	637 kroner	64-127 kroner

Ved hjelp av disse tallene kan vi regne ut begge gjennomsnittsprisene vi ønsker å bruke:

$$Gjennomsnittlig\ pris\ per\ del\ (TPPD) = \frac{\sum Pris\ på\ del}{Antall\ deler} = 649{,}22\ kroner$$

$$10\%\ av\ TPPD = 649{,}22 * 0{,}1 = 64{,}9\ \ kroner \approx 65\ kroner$$
$$20\%\ av\ TPPD = 649{,}22 * 0{,}2 = 129{,}84\ kroner \approx 130\ kroner$$

Dette er tallene gruppen har brukt for valg av sporingsteknologi og enheter.

Om vi går ut ifra at hver barelle inneholder 50 stillasdeler vil dette gi oss en gjennomsnittlig pris på følgende:

$$Gjennomsnittlig\ pris\ per\ barelle = 649{,}22 * 50 = 32\ 461\ kroner$$

Dette gir oss en 10-20% pris på 3 246- 6 492 kroner per barelle med stillas.

Gruppen vil vurdere sporingsteknologi i forhold til følgende aspekter:

- Fremtidig utvikling: Om teknologien er ekstremt billig, men utdatert ønsker ikke gruppen å basere løsningen på den, da systemet vil være ubrukelig om noen år
- Integrasjon: Om teknologien er enkel å integrere i en programvare vil dette være meget attraktivt for gruppen.

liv

- Signal: Om teknologien enkelt trenger seg igjennom stål og aluminium vil dette også være til fordel for løsningen.
- Pris: Om pris per enhet overskriver TPPD med en god margin vil ikke gruppen se det som forsvarlig å gå for denne teknologien med mindre forskning og analyse tilsier at prisen vil minke betraktelig i fremtiden.

Gruppen brukte en kombinasjon av disse ved valg av sporingsenhet. Disse ble i utgangspunktet vektlagt likt, men aspekter som hvor fremtidsrettet teknologien er ble av gruppen sett på som mer aktuelt enn pris, da en fremtidsrettet teknologi vil bli mer tilgjengelig og billigere.

Startfase/Ubrukte teknologier

Bacheloroppgaven inneholder både en fysisk sporingsdel og ønske om en digital løsning dette gjorde det hensiktsmessig for gruppen å dele opp oppgaven i to hoveddeler, denne delen av dokumentet vil fokusere på hvordan gruppen har gått frem for å finne en sporingsløsning som kan løse problemet, hvilke problemer som ble støtt på samt om det finnes en teknologi i dag som imøtekommer oppdragsgivers ønsker.

I oppstarts-delen så gruppen kollektivt på diverse sporingsteknologier og mulige måter å ta disse i bruk. Denne delen av prosjektet vil være viktig da den legger grunnlaget for hvilken sporingsteknologi løsningen skal bygges på. Til tross for dette har gruppen valgt å ikke lage en skreddersydd sporingsenhet for oppgaven, gruppen vil undersøke tilgjengelig teknologi samt komme med et forslag på hvordan den valgte teknologien kan implementeres fysisk på byggeplassen. Dette feltet (*Industrial Internet of Things)* er i hurtig utvikling og gruppen mener derfor at å legge mer søkelys på type sporingsteknologi vil være mer hensiktsmessig enn å lete på markedet etter en spesifikk sporingsenhet som kan tas i bruk.

Andre del av prosjektet vil bli sentrert rundt den digitale løsningen og integrasjon av sporingsteknologien i denne. Dette vil være hovedfokuset til gruppen, dette er fordi et bytte av sporingsenhet vil være enklere for oppdragsgiver enn å lage ny programvare. I tillegg vil gruppen få mer læringsutbytte av å legge fokuset på programvaren samt at det er lettere for gruppen å demonstrere ferdigheter tilegnet under bachelorstudiet. På grunn av hvordan gruppen har valgt å angripe oppgaven er det viktig at sporingsteknologien som blir valgt er fleksibel og uavhengig av en spesifikk enhet. Dette vil gi oppdragsgiver mulighet til å finne en sporingsenhet som er egnet for problemet uten å måtte re-designe hele systemet.

Gruppen startet oppgaven ved å kartlegge aktuelle teknologier som kunne bli brukt til å løse problemstillingen, gruppen konkluderte med at følgende teknologier kunne være hensiktsmessige for problemet til MB-Stillas:

- Strekkode
- NFC – Near Field Communication
- UWB – Ultra WideBand
- QR-Koder
- RFID – Radio Frequency Identification
- BLE – Bluetooth Low Energy
- LPWAN – Low Powered Wide Area Network

Disse teknologiene er forklart i detalj under seksjonen sporingsteknologier.

Gruppen fant tidlig ut at flere av disse teknologiene ikke ville egne seg til problemet: Strekkoder er meget billige, men krever at skanneren er veldig nær koden noe som gjør løsningen uaktuell for alt annet enn varetelling, og der vil det kreve en ansatt som må manuelt skanne hver tag.

NFC Teknologi ble også diskutert, denne formen for skanning var noe aktuell, da den enkelt kan brukes for å identifisere individuelle deler på byggeplass. NFC krever derimot en avstand på maksimalt 20 cm noe som gjør teknologien uaktuell for alt annet enn identifisering av deler på byggeplass. NFC ble derfor noe vi hadde lyst til å implementere som en tilleggsfunksjon som lar MB-Stillas identifisere sitt stillas på byggeplassen. QR koder ble også diskutert tidlig i prosessen av samme grunn som NFC, teknologien er lett å bruke samt masseprodusere, men ville hatt samme formål som NFC (identifisering av deler).

UWB var en av teknologiene gruppen mente var meget aktuell for problemstillingen. Siden teknologien tilbøy aktiv sporing med god presisjon samt lite batteribruk virket den som en sterk kandidat for løsning av problemet, men etter nøyere granskning fant gruppen ut at teknologiens rekkevidde ikke var stor nok, dette i kombinasjon med en høy pris gjorde teknologien uaktuell for løsningen til gruppen.

Hoveddel

Etter å ha konkludert med at ingen av sporingsteknologiene diskutert i *Oppstart/Ubrukte teknologier* var nok egnet til å bære hele problemstillingen valgte gruppen å legge hovedfokus på tre sporingsteknologier: *Bluetooth Low Energy (BLE), Radio Frequency Identification (RFID)* og *Low Powered Wide Area Network (LPWAN).* Gruppen mente at disse tre teknologiene kunne brukes for en potensielt god løsning. Gruppen var ute etter en teknologi som tilbød aktiv sporing, men i store intervaller: Dette ville spare batteri i kombinasjon med at lokasjonsoppdateringer i minutt intervaller var unødvendig da problemstillingen er sentrert rundt logistikken i utleie av stillas.

BLE var den første teknologien gruppen så på som en aktuell løsning av problemet. Den tilbyr aktiv sporing på spesifiserte intervaller, i tillegg har BLE en potensielt stor rekkevidde avhengig av terreng samt at den har blitt brukt i mange selskaper som et verktøy for sporing av små verktøydeler og lignende. Dette i kombinasjon med at *Bluetooth* er en teknologi som utvikler seg hurtig gjorde den meget interessant for gruppen. Gruppen gikk i dialog med *ABAX* som er en av Nordens ledende bedrifter innenfor sporing. Her fikk gruppen informasjon angående hvordan deres minste og billigste sporingsenhet fungerte,

Gruppen konkluderte med at BLE var en teknologi som kunne brukes, men for å holde kostnadene rimelige måtte en spesialisert sporingsenhet lages der enhetsintervallet blir satt til 1-3 ganger daglig og kun sender koordinater og en unik ID. Det vil derimot ikke være økonomisk forsvarlig for MB-Stillas å kjøpe inn BLE sporingsenheter til hver enkelt stillasdel.

Forslag til løsning:

Gitt at gruppen har fått full frihet til hvordan den fysiske løsningen blir har gruppen kommet opp med fem potensielle løsninger som tar i bruk en sentral LPWAN enhet:

Foreslåtte løsninger:

Gruppen startet prosessen ved å se på RFID løsninger, dette var mye fordi dette er den mest økonomiske løsningen samt en av løsningene som har blitt prøvd før. Innenfor sporing av små og billige verdier er RFID den mest brukte teknologien.

RFID port på lager:

En RFID port som leser av individuelle stillasdeler når de kommer inn og sendes ut av lager, en slik løsning ville gi MB-Stillas oversikt over hvilke stillasdeler som går ut fra lageret samt når de kommer inn igjen.

Fordeler:

- Økonomisk forsvarlig i dag
- Testet av andre aktører i lignende problemstillinger
- Gir god oversikt over verdier inn og ut av lager

Ulemper:

- Lite innovativt
- Ingen aktiv sporing av stillasdeler
- Løsningen er ikke praktisk på byggeplass (RFID porter kan ikke settes opp på byggeplass så manuell skanning er nødvendig)

Gruppen så på denne løsningen som en mulighet om aktiv sporing ikke var mulig, men ønsket i utgangspunktet å styre unna en slik løsning da den ikke er innovativ og fremtidsrettet. Ved undersøkelse av RFID teknologi fant gruppen ut at radiosignalene som blir sendt ut av porter og andre skannere blir påvirket av stål. På grunn av dette tror gruppen at en port som må lese av mange stillasdeler samtidig vil fungere dårlig. Løsningen ville også legge mye vekt på den digitale løsningen da RFID funksjonalitet er meget begrenset. På grunn av dette ble løsningen lagt bort da gruppen mente den ikke ville gi et godt resultat.

RFID leser på «barelle»

Gitt at en RFID port vil ha problemer med å lese av mengder med stillasdeler ønsket gruppen å finne en potensiell RFID løsning som vil fungere på byggeplass. Siden stillasdeler blir transportert i «bareller» bør disse kassene være involvert i løsningen uavhengig av teknologi. Gruppen mente derfor at en aktuell løsning var å spesial-designe en barelle med en integrert RFID leser i den øvre delen av barellen, dermed vil stillasdeler bli «lest» når de forlater kassen og når de blir lagt tilbake. Løsningen er derimot avhengig av en sentral «gateway-enhet» som lar gruppen kommunisere med barellene.

Fordeler:

- Økonomisk forsvarlig (RFID er den billigste teknologien og en sentralsporer per barelle er mindre kostbart enn sentralenheter på stillasdelene)
- Testet av andre aktører i lignende problemstillinger
- Gir oversikt over stillasdeler på byggeplass

Ulemper:

- Lite innovativt (RFID er en standardisert teknologi som har mindre utvikling enn andre alternativer)
- Kun oppdatering av stillasdelers posisjon da de forlater/blir lagt tilbake i barellene

Gruppen så på denne løsningen som mer aktuell da den tilbyr sporing på byggeplass og lager. Løsningen er derimot avhengig av RFID. Løsningen krever også en spesiallaget barelle noe som faller utenfor gruppens omfang så prototyping vil ikke være mulig.

BLE tags i kombinasjon med sentralenhet
Gruppen så på BLE som et bedre alternativ til RFID da denne teknologien ville være mer fremtidsrettet enn å basere løsningen på RFID. BLE tags krever også mindre komponenter i systemet enn det en RFID løsning gjør, et system som baserer seg på RFID krever tre hovedkomponenter:

- En RFID tag
- En antenne som sender ut signaler
- En leser som tar imot og behandler signalet

I tillegg krever en god løsning også en sentral gateway som snakker med skyen. Dermed vil systemet kreve fire komponenter. BLE derimot har kun to komponenter:

- En BLE tag
- En sentral gateway med Bluetooth kompatibilitet

I tillegg er bluetooth low energy en nyskapende teknologi som forbedres kontinuerlig med oppdateringer fra Bluetooth. Ved undersøkelse av tidligere løsninger som omhandler sporing av små, billige verdier kan gruppen konkludere at Bluetooth Low Energy blir brukt i alle de store selskapene (ABAX, blueiot og Nordic Semiconductur). Gruppen ønsker derfor å basere løsningen rundt BLE i kombinasjon med en sentral sporingsenhet som tar i bruk LPWAN.

Fordeler:

- Nyskapende og fremtidsrettet
- Fremoverlent i forhold til andre sporingsløsninger
- Tilbyr aktiv sporing både på lager og byggeplass

Ulemper:

- Økonomisk uforsvarlig med hensyn på dagens marked

Endelig løsning

Etter undersøkelse, diskusjon og fremlegg for MB-Stillas den 25.02.2022 har gruppen valgt å gå for sistnevnte løsning: En sentralenhet plassert på byggeplass som kommuniserer med Bluetooth Low Energy enheter plassert på stillasdeler. Denne løsningen mener gruppen er fremtidsrettet og nyskapende gitt at den tar i bruk både BLE og LPWAN som begge er store aktører på IOT markedet i tillegg til at de begge vil være dominerende i det fremtidige sporingsmarkedet. Løsningen vil derimot ikke være økonomisk forsvarlig per i dag. På grunn av dette vil gruppens løsning fungere som et «proof of concept» og ikke en hyllevare.

En bluetooth low energy enhet har varierende rekkevidde siden enhetene opererer med signaler i ISM båndet som ligger fra 2.4 til 2.4835 GHz. Dette båndet blir påvirket av byggematerialer som betong og metall. Bluetooth gir selv ut følgende tall på rekkevidde:

Versjon	Bluetooth V2.1	Bluetooth V4.0	Bluetooth V5.0
Rekkevidde	Opp til 100m	Opp til 100m	Opp til 400m
Maksrekkevidde	Opp til 100m	Opp til 100m	Opp til 1000m

lix

Appendix H Brukertester Front-End

Dette dokumentet inneholder alle brukertester gruppen gjorde igjennom bacheloroppgaven. Brukere ble for disse testene bedt om å ta i bruk enten mobil eller web applikasjonen gruppen har utviklet. Det ble gjennomført to forskjellige typer tester:

- En test gjennomført med klare rammer og oppgaver, gitt av gruppen. For eksempel: Kan du finne frem et byggeprosjekt på adressen Jonas Lies gate, disse testene ville gi gruppen tilbakemelding på hvordan det er å ta i bruk brukergrensesnittene uten noen form for erfaring med systemet.
- Under den andre testen fikk testeren muligheten til å bevege seg fritt igjennom brukergrensesnittet i ti minutter før de gjennomførte en spørreundersøkelse angående brukervennligheten til brukergrensesnittene.

Personene som testet systemet var alle jevnaldrende og hadde lite erfaring med IT-systemer. Dette var et bevisst valg gruppen gjorde i og med at systemet utvikles for en bransje med lite IT erfaring. Gruppen er bevisst på at optimale test resultater kun ville oppnås om vi testet på faktiske ansatte i oppdragsgivers bedrift, men gruppen fikk ikke tid til dette. Testere er anonyme og ingen informasjon utenom deres tilbakemeldinger og resultater ble lagret. Testere måtte derimot verbalt bekrefte at de har lest og forstått et Samtykkeskjema som gruppen har laget før testing kunne forekomme.

Innhold

Samtykkeskjema

Formål:
Denne undersøkelsen blir gjort for å teste brukervennligheten til Front-End løsningen som har blitt utviklet i samarbeid med NTNU og MB-Stillas.

Fremgangsmåte:
Det vil bli gjennomført en av to typer tester på testeren:

- Type en gjennomføres med klare rammer og oppgaver, gitt av intervjuer. For eksempel: Kan du finne frem et byggeprosjekt på adressen Jonas Lies gate, disse testene ville gi gruppen tilbakemelding på hvordan det er å ta i bruk brukergrensesnittene uten noen form for erfaring med systemet.

- Under den andre testen får testeren muligheten til å bevege seg fritt igjennom brukergrensesnittet i ti minutter før de gjennomførte en spørreundersøkelse angående brukervennligheten til brukergrensesnittene

Konfidensialitet:
Dette er en anonym undersøkelse hvor kun det skriftlige resultatet fra test to eller notater fra intervjuer vil bli lagret. Testers alder og kjønn vil bli lagret, men ingen annen personlig informasjon vil bli tatt vare på

Test type 1 Oppgaver:

1. Kan du hente finne frem alle byggeprosjektene i appen/nettsiden
2. Kan du finne byggeprosjektet kalt CC Hamar
3. Kan du legge inn et nytt byggeprosjekt?
4. Kan du slette et byggeprosjekt
5. Kan du hente ut alle stillasdeler på CC Hamar prosjektet
6. Kan du legge inn et par nye stillasdeler?
7. Kan du overføre 10 stillasdeler, fra et prosjekt til et annet

Test type 2 Spørsmål:

1. Hvordan syntes du det er å navigere brukergrensesnittet?
2. Hva syntes du er bra med designet?
3. Hva kunne vært bedre/hva syntes du er vanskelig?
4. Hvordan opplever du oppsettet av byggeprosjekter og stillasdeler?
5. Hva tenker du om knappene i systemet?
6. Hvordan er det å navigere brukergrensesnittet i forhold til andre systemer du har brukt tidligere?

Test 1 Web applikasjon type 1

Alder: 38
Kjønn: Mann
Resultat:

1. Tester klarer uten problemer å få oversikt over alle byggeprosjekter i web applikasjonen
2. Tester søker opp CCHamar og får opp prosjektet uten noe problem
3. Tester bruker et par sekunder på å orientere seg før han legger inn et nytt prosjekt med informasjon
4. Brukeren sletter et prosjekt uten problemer
5. Brukeren trykker seg frem til stillasdeler og ser alle stillasdeler på byggeprosjektet
6. Brukeren bruker en del tid på å navigere seg frem til stillasdel siden før han legger inn en stillasdel av type bunnskrue
7. Brukeren tar i bruk overføring uten problemer

Test 2 Web applikasjon type 2

Alder: 45
Kjønn: Mann
Resultat:

1. Jeg syntes det er ganske oversiktlig og enkelt å navigere, litt mye informasjon rett etter innlogging men utenom dette er det uproblematisk å bruke nettsiden
2. Designet fungerer, det er stor tekst med gode forklaringer i toppen av siden.
3. Det er som sagt litt mye informasjon samtidig og dette er litt forstyrrende i starten
4. Dette er bra, det er enkelt å se stillasdelene og byggeprosjektene, det er kanskje litt mye forhånds-vist informasjon jeg hadde kun trengt navn på prosjekt og et bilde ellers ingenting å klage på
5. Knappene er kanskje litt store og teksten er ikke helt midstilt, men de gjør jobben sin.
6. Det er litt mer kronglete enn andre nettsider jeg har brukt, men det fungerer fint og er ikke frustrerende å bruke

Test 3 Mobil applikasjon type 1

Alder: 52
Kjønn: Mann
Resultat:

1. Tester bruker litt tid på å se på første vindu før han finner frem til prosjekter
2. Tester ser seg rundt på skjermen før han omsider søker opp CC Hamar
3. Ikke gjennomført da mobil-applikasjon ikke støtter dette
4. Ikke gjennomført da mobil-applikasjon ikke støtter dette
5. Tester finner CC Hamar prosjektet og trykker seg litt rundt før han scroller ned og finner oversikten over stillasdeler
6. Ikke gjennomført da mobil-applikasjon ikke støtter dette
7. Tester trykker seg frem til overføring og bruker en del tid på å sette seg inn i siden før han overfører stillasdelene

Test 4 Mobil applikasjon type 2

Alder: 28
Kjønn: Kvinne
Resultat:

1. Det er enkelt, med få knapper noe som gjør det lett å navigere
2. Jeg liker utseendet veldig godt, det ligner mange andre apper jeg har brukt før
3. Det er kanskje litt mye tekst på knapper og sånt, noen av symbolene er selvforklarende syntes jeg
4. Det er oversiktlig men jeg må bevege meg mye rundt på appen for å få en god oversikt over prosjektene (mye scrolling)
5. Knappene er kanskje litt store for noen av sidene, men de fungerer helt fint, teksten er kanskje litt tynn?
6. Appen ligner veldig på apples egne apper, noen av sidene deres er litt tomme. For eksempel filter sidene

Test 5 Web applikasjon type 2

Alder: 32
Kjønn: Mann
Resultat:

1. Jeg syntes det er ok, det er ikke så mye å gjøre på appen men jeg finner frem uten mye problemer. Det tar litt tid å sette seg inn i, men etter et minutt eller to føler jeg at jeg kan navigere appen uten problemer
2. Jeg liker at det er en seksjon i toppen hvor jeg kan gå til andre sider av nettsiden, det gjør det enkelt å navigere til riktig del av siden.
3. Jeg syntes at dere skulle hatt en tilbake knapp slik at jeg kom tilbake til forrige side! En hjelp knapp som forklarte appen ville også vært bra!
4. Det er kanskje litte store bokser for prosjekter? Det fyller hele siden og gjør det litt klaustrofobisk. Designet fungerer men det er litt lite pusterom
5. De er ok, ingenting å klage på, det tok litt tid før jeg skjønte at knappene i toppen var knapper kanskje vise dette litt klarere
6. Nettsiden er kanskje litt gammeldags i forhold til nyere sider men den er enkel å navigere

Test 6 Mobil applikasjon type 2

Alder: 60
Kjønn: Mann
Resultat:

1. Jeg brukte ganske lang tid på å forstå hvordan ting fungerte her, men etter hvert ble det enklere å bruke systemet
2. Designet funker, og det er få ting å trykke på som gjør det lett å orientere seg
3. Jeg syntes det er liten tekst så jeg sliter med å lese noe av informasjonen i appen
4. Det fungerer bra og det er lett å se info om prosjektet og deler på prosjektet
5. Det er for liten tekst på knappene i systemet
6. Jeg syntes det fungerer fint å navigere appen, litt små knapper kanskje

Appendix I Utviklingsprosess

Dokumentet har følgende struktur: En total backlog med de forskjellige målene, når de ble påbegynt, hvor mange sprinter gruppen brukte for å gjennomføre målet og når det ble ferdig. Under denne backloggen ligger hver scrum iterasjon gruppen hadde, hver sprint inneholder målet med sprinten, tidsrommet på hver sprint samt hva som ble oppnådd samt backloggen på sprint iterasjonens tidspunkt.

Innhold

Avsluttende Backlog:

Mål	Start dato	Ansvarlig	Kommentar	Status
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Ferdig vi bruker Telia
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Ferdig
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Ferdig
Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Ferdig
Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Ferdig
Lage hovedrapport dokument	11.03.2022	Martin	Sette opp dokument	Ferdig
Revidere API design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må API revideres	Ferdig
Revidere DB design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må DB revideres	Ferdig
Begynne implementasjon Web	11.03.2022	Aleksander		Ferdig
Begynne implementasjon Mobil	11.03.2022	Tormod	Satt opp kart og grunnstruktur	Ferdig
Finne BLE tag til enhet	11.03.2022	Tormod og Martin	Gruppen trenger en BLE enhet til sentralen fra Telia	Scrappet
Hente Prosjekt endpoint	11.03.2022	Aleksander	Gruppen begynner utvikling	Ferdig
Hente stillasdel endpoint	11.03.2022	Martin	Gruppen begynner utvikling	Ferdig
Slette prosjekt endpoint	12.03.2022	Aleks		Ferdig

Finne ny sporingsenhet	15.03.2022	Tormod	Gruppen kan ikke bruke enhet fra Telia og må finne ny løsning	Ferdig
Begynne skriving på avsluttende rapport	18.03.2022	Martin	Gruppen ønsker å begynne på rapportskriving tidlig	Ferdig
Standardisere error-handling for API	16.03.2022	Aleksander	Aleksander ønsker å standardisere error handling for API-et	Ferdig
Implementere slette stillasdel funksjon	13.03	Martin		Ferdig
Implementere legge til prosjekt	14.03.2022	Aleksander		Ferdig
Implementere legge til, hent ut og slett bruker	19.03.2022	Aleksander		Ferdig
Sette opp API-Tester	24.03.2022	Martin	Martin ønsker å teste systemet	Ferdig
Legge inn søkefunksjonalitet i Mobil-Applikasjon	23.03.2022	Tormod		Ferdig
Legge inn funksjonalitet for overføring av stillasdeler	24.03.2022	Aleksander		Ferdig
Legge inn kartfunksjonalitet i mobil	26.03.2022	Tormod	Ønsker å oppgradere kartfunksjonalitet	Ferdig
Hente ut prosjektinfo og stillasdel info i mobil appen	27.03.2022	Tormod	Ønsker å hente data fra API	Ferdig
Implementere «Annotations» mobil-aplikasjon	28.03.2022	Tormod		Ferdig
Implementere Docker for deployering av løsning	01.04.2022	Martin	For å kunne spinne opp alle systemets komponenter trengs det en konteiner løsning som Docker	Ferdig
Oppdatere kart funskjonalitet i Web	01.04.2022	Aleksander		Ferdig
Starte med overføring av stillasdeler Mobil applikasjon	01.04.2022	Tormod		Ferdig
Implementere Docker Compose fil	06.04.2022	Martin		Må gjøres
Få ut data fra sporingsenhet	04.04.2022	Martin	Vi har nå fått sporingsenhet og Martin jobber med å få ut data	Ferdig

Legge inn egen fil for API info henting, mobil	02.04.2022	Tormod		Ferdig
Legge inn Modaler for stillasdel og overføring	08.04.2022	Aleksander		Jobbes med ta
Implementere MQTT broker	08.04.2022	Martin	Ser på implementasjon av mqtt broker	Scrappet
Implementere filtrering i mobil	13.04.2022	Tormod		Ferdig
Legge til rette for geofence	14.04.2022	Aleksander		Ferdig
Få sporingsdata inn i API	19.04.2022	Martin		Ferdig
Oppdatere byggeprosjekter og stillasdeler med data	21.04.2022	Martin		Ferdig
Implementere caching i mobil	19.04.2022	Tormod		Ferdig
Oppdatere data i web applikasjon	18.04.2022	Aleksander		Ferdig
Ferdigstilling av prosjekt i mobil applikasjon	19.04.2022	Tormod		Ferdig
Implementere gateway endpoint i API	19.04.2022	Martin	Api-et trenger et gateway endpoint da enhetene må inn i databasen	Ferdig
Ferdigstille oppdatering av posisjon på stillasdeler	25.04.2022	Martin		Ferdig
Implementere login funksjonalitet	28.04.2022	Aleksander	Systemet bør ha login funksjonalitet	Ferdig
Gjenoppta arbeid med filtrering mobil applikasjon	27.04.2022	Tormod	Gjenopptar arbeid på filtrering	Ferdig
Ferdigstille rapport	02.05.2022	Alle gruppemedlemmer		Jobbes med
Kommentere kode	05.05.2022	Alle gruppemedlemmer		Jobbes med

Scum-Prosess

Sprint 1

Tidsrom: 11-14 Januar

Mål:

- **Oppsett av prosjektplan dokument:** Gruppen ønsker å få skrivingen i gang tidlig
- **Sette opp spørsmål til oppdragsgiver angående oppgaven:** Gitt at gruppen syntes oppgaveteksten er noe vag vil oppgavedefinisjon tidlig i prosessen være ekstremt viktig
- **Sette opp spørsmål til veileder:** Gruppen ønsker å komme godt i gang tidlig så kartlegging av prosessen, da særlig prosjektplanen er viktig.
- **Sette opp timeliste dokument:** Selvforklarende mål
- **Lage en grov plan på hva vi ønsker å utvikle:** Gruppen mener det er fordelaktig å presentere hva vi ønsker å gjøre for arbeidsgiver tidlig

Resultat:

- **Oppsett av prosjektplan dokument:** Dokument ble laget og en grov mal ble laget etter møte med veileder den 11 Januar
- **Sette opp spørsmål til oppdragsgiver angående oppgaven:**
  Følgende spørsmål ble laget:
  1. Har dere en database/it-plattform (Hva bruker dere for å lagre informasjon)
  2. Skal vi ta hensyn til alle typer stillass eller fokusere på ?
  3. Hvordan transporteres stillaset? Bilder?
  4. Vi kan kun software aka vi er ganske avhengige av en "ferdig" tracker vi kan jobbe med
  5. Hvilke av punktene er viktigst for dere?
  6. Hvor involvert vil dere være i Prossessen, vi vil ha dere med på det meste!
  7. Hvordan skal vi kjøpe inn ting?
  8. Skal vi integrere dette i et eksisterende økosystem eller starte "Fra scratch"

- **Sette opp spørsmål til veileder:**
  Følgende spørsmål ble skrevet ned:
  1. Har vi tilgang til GitLab? Er dette noe vi må spørre om?
  2. Hva kan vi gjøre nå denne uka?
  3. Når ser du for deg at vi bør være ferdig med alt d praktiske (Påske)?
  4. Noen vi kan snakke med som har kompetanse innenfor tracking(tracker levetid etc)
  5. Hvor egoistiske skal vi være? (Vi er jo programvare utviklere ikke mikrochip ingeniører)
- **Sette opp timeliste dokument:** Timeoversiktsdokument ble opprettet
- **Lage en grov plan på hva vi ønsker å utvikle:** Gruppen hadde et møte med mylift den 13.01 hvor plan ble lagt og diskutert

Referat:

Gruppen er klare for bachelor oppgave og håper å få skrevet en god del på prosjektplanen til neste Scrum sprint. Møte med veileder gikk bra og ga mye god informasjon angående prosjektplanen

Backlog 14.01.2022

Mål	Start dato	Ansvarlig	Kommentar	Status

lxx

Prosjektplan	11.01.2022	Alle gruppemedlemmer	Påbegynt Tas med til neste sprint

Sprint 2

Tidsrom: 17-28 Januar

Mål:

- **Levere førsteutkast av prosjektplan til veileder:** Frist for levering av planen er 31 Januar så gruppen ønsker å ha et ferdig førsteutkast før avslutningen av Sprinten
- **Sette opp og komme i gang med et sporingsdokument:** Gitt at gruppen selv må finne en sporingsteknologi vi mener egner seg problemstillingen til arbeidsgiver ønsker gruppen å lage en rapport som inneholder all tilegnet kunnskap rundt sporing.

Resultat:

- **Levere førsteutkast av prosjektplan til veileder:**
  Gruppen sendte inn et førsteutkast til veileder den 25.01.2022
- **Sette opp og komme i gang med et sporingsdokument:**
  Gruppen har ikke laget et sporingsdokument, men har derimot brukt mye tid på å undersøke forskjellige måter å spore billige verdier på

Referat:

Vi er fornøyde med innsatsen sålangt, men skulle ønske at vi hadde kommet noe lenger med sporingsteknologien. Gruppen planlegger å ha tatt en endelig beslutning på type sporingsteknologi innen 04.02.2022, målet om sporingsteknologi dokument tas videre til Sprint 3.

Backlog 28.01.2022

Mål	Start dato	Ansvarlig	Kommentar	Status
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Så godt som ferdig	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er ikke laget, men research er påbegynt	Påbegynt, tas videre til neste Sprint

Sprint 3

Tidsrom: 31.Januar-11.Februar

Mål:

- **Jobbe videre med sporingsteknologidokument:** Gruppen fortsetter med sporingsdokumentet gruppen ønsker å bli så godt som ferdig med sporingsdokumentet slik at vi kan begynne på utvikling
- **Finne aktuell sporingsenhet:** Det er essensielt at gruppen finner en passende teknologi og enhet slik at utvikling kan begynne samt at gruppen får bestilt inn en enhet for testing. Fokus på LPWAN, RFID og BLE

Resultat:

- **Jobbe videre med sporingsteknologidokument:**
  Gruppen er godt i gang med dokumentet, foreløpig er det ingen teknologier som passer myLifts use-case til punkt og prikke, men dette er noe gruppen forventet. Gruppen diskuterer de forskjellige alternativene med oppdragsgiver

- **Finne aktuell sporingsenhet:** Gruppen har ikke funnet en passende enhet enda, gruppen har derimot sendt mailer til diverse selskaper angående deres enheter og løsninger.

Referat:

Gruppen har iløpet av disse to ukene fortsatt med sporingsteknologi dokumentet og ser fortsatt etter en aktuell sporingsteknologi og enhet. Spesifikasjoner og fokusområde diskuteres med oppdragsgiver og gruppen håper å lande på en passende enhet innen Sprint 4. Gruppen har valgt å begrense aktuelle teknologier til Bluetooth Low Energy, RFID og LPWAN (Low Powered Wide Area Network).
Gruppen er i dialog med følgende bedrifter angående enhet:

1. Telia (LPWAN)
2. Abax (BLE)
3. Disruptive Technologies (BLE)
4. Schake (RFID)

Gruppen har også vært på besøk hos HAKI for å se nærmere på hvordan de behandler stillas samt det fysiske stillaset. Gruppen følte de lærte mye rundt hvordan stillaset bygges opp og hvilke begrensninger vi jobber med. Gruppen er også i dialog med oppdragsgiver angående et besøk hvor gruppen skal legge frem en plan for hvordan vi skal løse problemstillingen (planlagt visitt er foreløpig 25.02.2022)

Backlog 11.02.2022

Mål	Start dato	Ansvarlig	Kommentar	Status
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Påbegynt, tas videre til neste Sprint
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer ser etter aktuell teknologi	Påbegynt tas videre til neste sprint
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Påbegynt, tas videre til neste sprint
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur må på plass	Påbegynt tas videre til neste sprint

Sprint 4

Tidsrom: 14-25 Februar

Mål:

- **Ferdigstille sporingsdokument og finne en aktuell sporingsteknologi:** Gruppen skal legge frem en plan for oppdragsgiver den 25.02 og må innen da ha et forslag på hvilken sporingsteknologi vi ønsker å basere oppgaven på
- **Lande på system-arkitektur for løsningen:** Gruppen skal legge frem forslag for oppdragsgiver og må derfor ha en system-arkitektur klar til 25.02
- **Finne en potensiell sporingsenhet:** Dette er ønskelig, men gruppen er usikre på om de klarer å finne en aktuell enhet

- **Fullføre presentasjon for oppdragsgiver:** Presentasjon skal holdes og må bli ferdig til frist

Resultat:

- **Ferdigstille sporingsdokument og finne en aktuell sporingsteknologi:** Gruppen er så godt som ferdig med sporingsteknologi dokumentet og har etter møtet med myLift bestemt seg for å velge en kombinasjon av BLE og LPWAN for løsningen.
- **Lande på system-arkitektur for løsningen:** Gruppen har bestemt seg for å lage et API som kommuniserer med en database og sporingsfuksjonaliteten. Dette API-Et skal tas i bruk av en Web-Løsning egnet for kontorbruk og en enklere mobil-løsning som kan tas i bruk på byggeplass for å sjekke info om stillasdeler og byggeplass
- **Finne en potensiell sporingsenhet:** Gruppen har potensielt funnet en enhet etter dialog med telia, denne har både GPS, BLE og LPWAN innebygd ifølge salgsperson hos Telia og kan brukes for å løse gruppens problemstilling
- **Fullføre presentasjon for oppdragsgiver:** Gjort

Referat:

Gruppen har i løpet av sprinten skrevet på sporingsteknologi dokumentet, i tillegg har gruppen bestemt seg, etter dialog og møter med Telia for å ta utgangspunkt i deres sporingsenhet i kombinasjon med små BLE enheter. Dette vil gi gruppen en enhet med LPWAN funksjonalitet i tillegg til GPS koordinater og muligheter for kommunikasjon med små BLE tags. Gruppen var også hos oppdragsgiver i perioden og presenterte ideen for de. Besøket gikk bra og gruppen er fornøyd med fremdriften.

Backlog 25.02.2022

Mål	**Start dato**	**Ansvarlig**	**Kommentar**	**Status**
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer skal gå videre med sporingsenhet fra Telia	Så godt som ferdig
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Påbegynt tas med til neste Sprint

Sprint 5

Tidsrom: 25.02- 03. Mars

Mål:

- **Gruppen ønsker å finne en egnet sporingsenhet:** Gruppen ønsker å ha et møte med telia hvor vi får pris og bestiller enheten de leverer.
- **Planlegge programmeringsspråk:** Gruppen må velge programmeringsspråk og sette opp kodebasen slik at vi kan begynne med utvikling

- **Begynne på database og grunnstruktur i API:** Gitt at hele systemet skal benytte databasen ønsker gruppen å komme i gang med design og implementasjon av denne så i løpet av Sprinten.

Resultat:

- **Gruppen ønsker å finne en egnet sporingsenhet:** Gruppen skal ha et møte med Telia med en konsulent for å kunne få ned alle spesifikasjoner før bestilling av enheten skjer
- **Planlegge programmeringsspråk:** Gruppen har valgt å skrive back-end (API) i Golang, Web løsningen i React og mobil applikasjonen i Swift
- **Begynne på database og grunnstruktur i API:** Gruppen var i dialog med faglærer i databaser og skal i møte med lærer den 09.03 for å diskutere valg av database.

Referat:

Gruppen har i perioden hatt mail korrespondanse med Telia og har avtalt et møte den 15.03 for å kunne spesifisere mer rundt sporingsenhet. I tillegg har gruppen bestemt seg for å skrive programmet i Golang, React og Swift. Gruppen planlegger også å bruke google firestore, en NoSql database. Dette skal diskuteres mer med en faglærer den 09.03.2022.

Backlog 03.03.2022

Mål	Start dato	Ansvarlig	Kommentar	Status
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Så godt som ferdig
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Ferdig
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Påbegynt
Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Påbegynt
Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Påbegynt

Sprint 6

Tidsrom: 03-11 Mars

Mål:

- **UI Design av mobil og web applikasjon:** Gruppen skal prototype design av applikasjonen slik at brukertester kan skje ved neste sprint

- **Planlegging av sporingsløsning på byggeplass:** Gruppen ønsker å planlegge hvordan løsningen skal deployeres på byggeplass.

Resultat:

- **UI Design av mobil og web applikasjon:** Gruppen har designet både web og mobil designet og prototypene er klare for brukertester
- **Planlegging av sporingsløsning på byggeplass:** Gruppen har flere design på hvordan dette skal gjøres

Referat:

Gruppen har i perioden designet UI for front-end, i tillegg har gruppen laget user stories og use cases for applikasjonen. Gruppen må derimot revidere design, da vi ikke er fornøyde med hvordan front-end ser ut for øyeblikket. Oppdragsgiver ønsker å ha en sentral enhet på byggeplass som kommuniserer med BLE tags. Gruppen har med mylift bestemt seg for å bruke Telias enhet

Backlog 11.03.2022

Mål	**Start dato**	**Ansvarlig**	**Kommentar**	**Status**
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Ferdig vi bruker Telia
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Ferdig
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Ferdig
Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Ferdig
Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Ferdig
Lage hovedrapport dokument	11.03.2022	Martin	Sette opp dokument	Ikke påbegynt
Revidere API design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må API revideres	Påbegynt
Revidere DB design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må DB revideres	Påbegynt
Begynne implementasjon Web	11.03.2022	Aleksander		Påbegynt

Begynne implementasjon Mobil	11.03.2022	Tormod		Påbegynt
Finne BLE tag til enhet	11.03.2022	Tormod og Martin	Gruppen trenger en BLE enhet til sentralen fra Telia	Påbegynt
Programmere Prosjekt endpoint	11.03.2022	Aleksander	Gruppen begynner utvikling	Påbegynt
Programmere stillasdel endpoint	11.03.2022	Martin	Gruppen begynner utvikling	Påbegynt

Sprint 7

Tidsrom: 12 Mars-18 Mars

Mål:

- **Sette opp Prosjekt og stillasdel endpoints i API og database:** Gruppen har begynt utvikling og vi setter derfor i gang med api programmeringen
- **Tilegne seg kunnskap rundt mobilprogrammering:** Siden Tormod ikke har programmert tidligere må han sette seg inn i Swift rammeverket
- **Påbegynne web-utvikling:** Aleksander begynner smått med web-utvikling når han ikke jobber med API (ingen spesifikke mål her)

Resultat:

- **Sette opp Prosjekt og stillasdel endpoints i API og database:** Aleksander og Martin jobber med endpoints men er ikke ferdig innen 18.Mars
- **Tilegne seg kunnskap rundt mobilprogrammering:** Tormod har som smått begynt å eksperimentere
- **Påbegynne web-utvikling:** Aleksander har fått inn et kart i web-applikasjonen
- **Database:** Gruppen har også satt opp databasen
- **Finne Sporingsenhet:** Gruppen må finne ny sporingsenhet, dette er ikke etter planen og gruppen vurderer å teste systemet uten en fysisk enhet

Referat:

Gruppen hadde møte med Telia den 15.03, her fikk vi vite at deres enhet ikke kunne brukes. Gruppen må derfor bestemme seg for om vi skal teste med mock data eller finne en annen enhet. Tormod tar ansvar for dette. Gruppen har derimot kommet godt i gang med utvikling og satte mål følges.

Backlog 18.03.2022

Mål	**Start dato**	**Ansvarlig**	**Kommentar**	**Status**
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Ferdig vi bruker Telia

Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Ferdig
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Ferdig
Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Ferdig
Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Ferdig
Lage hovedrapport dokument	11.03.2022	Martin	Sette opp dokument	Ferdig
Revidere API design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må API revideres	Påbegynt
Revidere DB design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må DB revideres	Påbegynt
Begynne implementasjon Web	11.03.2022	Aleksander		Ferdig
Begynne implementasjon Mobil	11.03.2022	Tormod	Satt opp kart og grunnstruktur	Ferdig
Finne BLE tag til enhet	11.03.2022	Tormod og Martin	Gruppen trenger en BLE enhet til sentralen fra Telia	Scrappet
Hente Prosjekt endpoint	11.03.2022	Aleksander	Gruppen begynner utvikling	Ferdig
Hente stillasdel endpoint	11.03.2022	Martin	Gruppen begynner utvikling	Ferdig
Slette prosjekt endpoint	12.03.2022	Aleks		Påbegynt
Finne ny sporingsenhet	15.03.2022	Tormod	Gruppen kan ikke bruke enhet fra Telia og må finne ny løsning	Påbegynt
Begynne skriving på avsluttende rapport	18.03.2022	Martin	Gruppen ønsker å begynne på rapportskriving tidlig	Ikke påbegynt
Standardisere error-handling for API	16.03.2022	Aleksander	Aleksander ønsker å standardisere error handling for API-et	Påbegynt
Implementere slette stillasdel funksjon	13.03	Martin		Påbegynt

Implementere legge til prosjekt	14.03.2022	Aleksander	Påbegynt

Sprint 8

Tidsrom: 19 Mars - 25 Mars

Mål:

- **Begynne på avsluttende rapport:** Gruppen ønsker å begynne på hovedrapporten tidlig og vil derfor sette opp rapportstrukturen
- **Finne alternativ til enhet:** Sporingsenhet er krisesituasjon, Tormod skal sette av litt tid for å prøve å finne en alternativ enhet gruppen kan bruke for å teste systemet
- **Ferdigstille stillasdel og prosjekt endpoint:** Martin og Aleksander jobber med dette og ønsker å få det ferdig til avslutning av sprint

Resultat:

- **Begynne på avsluttende rapport:** Gruppen har opprettet rapport dokument men har ikke begynt på rapporten
- **Finne alternativ til enhet:** Tormod ser etter enhet, men hovedfokus ligger på mobilprogrammering.
- **Ferdigstille stillasdel og prosjekt endpoint:** Martin og Aleksander jobber med dette og ønsker å få det ferdig til avslutning av sprint. Mye av funksjonalitet er feridg
- **Sette opp en standard for error handling i API:** Aleksander satt standard opp for error handling
- **Programmere bruker-endpoint;** Aleksander ble ferdig tidligere enn forventet med prosjekt endpoint og laget derfor bruker endpoint.

Referat:

Gruppen mener selv dette har vært en effektiv sprint hvor det har blitt gjort mye, stillasdel og prosjekt endpointene er så godt som ferdige samt at Aleksander også har implementert et bruker endpoint (hent ut, lag og slett bruker). Tormod jobber fortsatt med mobilprogrammering. Gruppen føler at det er god fremgang men er noe stresset angående sporingsenhet.

Backlog 25.03.2022

Mål	Start dato	Ansvarlig	Kommentar	Status
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Ferdig vi bruker Telia
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande	Ferdig

			på programmeringsspråk	
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Ferdig
Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Ferdig
Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Ferdig
Lage hovedrapport dokument	11.03.2022	Martin	Sette opp dokument	Ferdig
Revidere API design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må API revideres	Ferdig
Revidere DB design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må DB revideres	Ferdig
Begynne implementasjon Web	11.03.2022	Aleksander		Ferdig
Begynne implementasjon Mobil	11.03.2022	Tormod	Satt opp kart og grunnstruktur	Ferdig
Finne BLE tag til enhet	11.03.2022	Tormod og Martin	Gruppen trenger en BLE enhet til sentralen fra Telia	Scrappet
Hente Prosjekt endpoint	11.03.2022	Aleksander	Gruppen begynner utvikling	Ferdig
Hente stillasdel endpoint	11.03.2022	Martin	Gruppen begynner utvikling	Ferdig
Slette prosjekt endpoint	12.03.2022	Aleks		Påbegynt
Finne ny sporingsenhet	15.03.2022	Tormod	Gruppen kan ikke bruke enhet fra Telia og må finne ny løsning	Påbegynt
Begynne skriving på avsluttende rapport	18.03.2022	Martin	Gruppen ønsker å begynne på rapportskriving tidlig	Ikke påbegynt
Standardisere error-handling for API	16.03.2022	Aleksander	Aleksander ønsker å standardisere error handling for API-et	Ferdig
Implementere slette stillasdel funksjon	13.03	Martin		Ferdig
Implementere legge til prosjekt	14.03.2022	Aleksander		Ferdig
Implementere legge til, hent ut og slett bruker	19.03.2022	Aleksander		Ferdig
Sette opp API-Tester	24.03.2022	Martin	Martin ønsker å teste systemet	Påbegynt

Legge inn søkefunksjonalitet i Mobil-Applikasjon	23.03.2022	Tormod	Påbegynt
Legge inn funksjonalitet for overføring av stillasdeler	24.03.2022	Aleksander	Påbegynt

Sprint 9

Tidsrom: 26 Mars - 01 April

Mål:

- **Gruppen ønsker å ferdigstille mye av API funksjonaliteten:** Martin og Aleks ønsker å få ferdig all funksjonalitet som ikke bruker sporingsenheten
- **Implementering av API-Tester:** Martin ønsker å kunne gjøre automatiske API-Tester for å effektivt sjekke om systemet fungere som det skal
- **Få inn grunnfunksjonalitet for utseende i mobil-applikasjonen:** Tormod ønsker å få ferdig grunnfunksjonaliteten i mobil applikasjonen
- **Hente data fra API:** Tormod ønsker å hente JSON data fra api-et og bruke det i mobil-applikasjonen innen sprinten er over

Resultat:

- **Gruppen ønsker å ferdigstille mye av API funksjonaliteten:** Mye av funksjonaliteten er ferdig og Aleks begynner smått å jobbe med web-løsningen. Martin ferdigstiller API/jobber videre med funksjonalitet
- **Implementering av API-Tester:** API tester er implementert for deler av systemet, men gruppen ønsker å teste alle endpoints til deadline
- **Få inn grunnfunksjonalitet for utseende i mobil-applikasjonen:** Søkbar navigation view er implementert. Tormod fortsetter arbeid med mobil-applikasjon. Mobil applikasjon bruker nå også kart og kan bruke byggeprosjekter og stillasdeler
- **Hente data fra API:** Mobil-Applikasjonen kan nå hente ut data fra API-et

Referat:

Backlog 01.04.2022

Mål	**Start dato**	**Ansvarlig**	**Kommentar**	**Status**
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Ferdig vi bruker Telia
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig

Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Ferdig
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Ferdig
Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Ferdig
Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Ferdig
Lage hovedrapport dokument	11.03.2022	Martin	Sette opp dokument	Ferdig
Revidere API design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må API revideres	Ferdig
Revidere DB design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må DB revideres	Ferdig
Begynne implementasjon Web	11.03.2022	Aleksander		Ferdig
Begynne implementasjon Mobil	11.03.2022	Tormod	Satt opp kart og grunnstruktur	Ferdig
Finne BLE tag til enhet	11.03.2022	Tormod og Martin	Gruppen trenger en BLE enhet til sentralen fra Telia	Scrappet
Hente Prosjekt endpoint	11.03.2022	Aleksander	Gruppen begynner utvikling	Ferdig
Hente stillasdel endpoint	11.03.2022	Martin	Gruppen begynner utvikling	Ferdig
Slette prosjekt endpoint	12.03.2022	Aleks		Ferdig
Finne ny sporingsenhet	15.03.2022	Tormod	Gruppen kan ikke bruke enhet fra Telia og må finne ny løsning	Ferdig
Begynne skriving på avsluttende rapport	18.03.2022	Martin	Gruppen ønsker å begynne på rapportskriving tidlig	Ikke påbegynt
Standardisere error-handling for API	16.03.2022	Aleksander	Aleksander ønsker å standardisere error handling for API-et	Ferdig
Implementere slette stillasdel funksjon	13.03	Martin		Ferdig
Implementere legge til prosjekt	14.03.2022	Aleksander		Ferdig

Implementere legge til, hent ut og slett bruker	19.03.2022	Aleksander		Ferdig
Sette opp API-Tester	24.03.2022	Martin	Martin ønsker å teste systemet	Ferdig
Legge inn søkefunksjonalitet i Mobil-Applikasjon	23.03.2022	Tormod		Ferdig
Legge inn funksjonalitet for overføring av stillasdeler	24.03.2022	Aleksander		Ferdig
Legge inn kartfunksjonalitet i mobil	26.03.2022	Tormod	Ønsker å oppgradere kartfunksjonalitet	Ferdig
Hente ut prosjektinfo og stillasdel info i mobil appen	27.03.2022	Tormod	Ønsker å hente data fra API	Ferdig
Implementere «Annotations» mobil-aplikasjon	28.03.2022	Tormod		Påbegynt
Implementere Docker for deployering av løsning	01.04.2022	Martin	For å kunne spinne opp alle systemets komponenter trengs det en konteiner løsning som Docker	Påbegynt
Oppdatere kart funskjonalitet i Web	01.04.2022	Aleksander		Påbegynt
Starte med overføring av stillasdeler Mobil applikasjon	01.04.2022	Tormod		Påbegynt

Sprint 10

Tidsrom: 02 April - 08 April

Mål:

- **Ferdigstille all API funksjonalitet som ikke innebærer sporingsenhet:** Martin og Aleksander ønsker å bli ferdig med så mye av API-et som mulig, etter Sprint 10 vil Aleksanders hovedfokus ligge på web-applikasjonen
- **Ferdigstille kart-funksjonalitet i mobil og web applikasjon:** Aleksander og Tormod ønsker å få på plass all kart-funksjonalitet gitt at dette er sentralt for oppgaven
- **Hente ut API-informasjon for både Mobil og Web applikasjonen:** Aleksander og Tormod klarer nå å hente ut informasjon fra API-et og ønsker i denne sprinten å ta det i bruk for byggeprosjekter og stillasdeler

Resultat:

- **Ferdigstille all API funksjonalitet som ikke innebærer sporingsenhet:** Mye av API er nå ferdig, men gruppen måtte bytte ruter funksjonalitet da standard biblioteket til Golang skapte problemer da vi hentet data til mobil og web applikasjonen

- **Ferdigstille kart-funksjonalitet i mobil og web applikasjon:** Både Web og Mobil viser nå kart over prosjekter
- **Hente ut API-informasjon for både Mobil og Web applikasjonen:** Både Mobil og Web applikasjonen kan nå hente ut informasjon fra API-et

Referat:

Gruppen har i denne sprinten blitt så godt som ferdig med hoved-funksjonalitet i API-et og jobbing med mobil og web-applikasjon går som planlagt. Gruppen har gått til anskaffelse av en sporingsenhet fra INGICS, denne er ikke optimal for vår use-case men gitt tidsrommet vi jobber med er den god nok.

Backlog 08.04.2022

Mål	Start dato	Ansvarlig	Kommentar	Status
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Ferdig vi bruker Telia
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Ferdig
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Ferdig
Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Ferdig
Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Ferdig
Lage hovedrapport dokument	11.03.2022	Martin	Sette opp dokument	Ferdig
Revidere API design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må API revideres	Ferdig
Revidere DB design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må DB revideres	Ferdig
Begynne implementasjon Web	11.03.2022	Aleksander		Ferdig
Begynne implementasjon Mobil	11.03.2022	Tormod	Satt opp kart og grunnstruktur	Ferdig

Finne BLE tag til enhet	11.03.2022	Tormod og Martin	Gruppen trenger en BLE enhet til sentralen fra Telia	Scrappet
Hente Prosjekt endpoint	11.03.2022	Aleksander	Gruppen begynner utvikling	Ferdig
Hente stillasdel endpoint	11.03.2022	Martin	Gruppen begynner utvikling	Ferdig
Slette prosjekt endpoint	12.03.2022	Aleks		Ferdig
Finne ny sporingsenhet	15.03.2022	Tormod	Gruppen kan ikke bruke enhet fra Telia og må finne ny løsning	Ferdig
Begynne skriving på avsluttende rapport	18.03.2022	Martin	Gruppen ønsker å begynne på rapportskriving tidlig	Ikke påbegynt
Standardisere error-handling for API	16.03.2022	Aleksander	Aleksander ønsker å standardisere error handling for API-et	Ferdig
Implementere slette stillasdel funksjon	13.03	Martin		Ferdig
Implementere legge til prosjekt	14.03.2022	Aleksander		Ferdig
Implementere legge til, hent ut og slett bruker	19.03.2022	Aleksander		Ferdig
Sette opp API-Tester	24.03.2022	Martin	Martin ønsker å teste systemet	Ferdig
Legge inn søkefunksjonalitet i Mobil-Applikasjon	23.03.2022	Tormod		Ferdig
Legge inn funksjonalitet for overføring av stillasdeler	24.03.2022	Aleksander		Ferdig
Legge inn kartfunksjonalitet i mobil	26.03.2022	Tormod	Ønsker å oppgradere kartfunksjonalitet	Ferdig
Hente ut prosjektinfo og stillasdel info i mobil appen	27.03.2022	Tormod	Ønsker å hente data fra API	Ferdig
Implementere «Annotations» mobil-aplikasjon	28.03.2022	Tormod		Ferdig
Implementere Docker for deployering av løsning	01.04.2022	Martin	For å kunne spinne opp alle systemets komponenter trengs det en konteiner løsning som Docker	Ferdig
Oppdatere kart funskjonalitet i Web	01.04.2022	Aleksander		Ferdig

Starte med overføring av stillasdeler Mobil applikasjon	01.04.2022	Tormod		Ferdig
Implementere Docker Compose fil	06.04.2022	Martin		Må gjøres
Få ut data fra sporingsenhet	04.04.2022	Martin	Vi har nå fått sporingsenhet og Martin jobber med å få ut data	Påbegynt
Legge inn egen fil for API info henting, mobil	02.04.2022	Tormod		Påbegynt
Legge inn Modaler for stillasdel og overføring	08.04.2022	Aleksander		Påbegynt
Implementere MQTT broker	08.04.2022	Martin	Ser på implementasjon av mqtt broker	Påbegynt

Sprint 11

Tidsrom: 09 April -15 April

Mål:

- **Få ut data fra sporingsenhet:** Martin jobber med sporingsenhet og ønsker å få ut og oversette payload data innen sprint avslutningen
- **Fortsette arbeid med mobil og web applikasjon:** Tormod og Aleks ønsker å bli ferdig med mobil og web applikasjon innen 1 mai så noe jobb skal skje i påsken

Resultat:

- **Få ut data fra sporingsenhet:** Vi får ut rådata fra enhet, Martin ønsket å ta i bruk en mqtt protokoll for å behandle data, men dette ble for tidkrevende
- **Fortsette arbeid med mobil og web applikasjon:** Utvikling går som planlagt

Referat:

Gruppen har i perioden fått sporingsenheten og får ut data, den fungere noe som forventet, men gateway/ruter sender ikke koordinater noe som er noe dumt. Gruppen harkoder koordinater for byggeprosjekter isteden og setter geofence lik ruterens rekkevidde. Gruppen skal jobbe smått i påsken men ingen Sprint blir satt opp

Backlog 15.04.2022

Mål	**Start dato**	**Ansvarlig**	**Kommentar**	**Status**
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Ferdig vi bruker Telia
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig

Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Ferdig
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Ferdig
Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Ferdig
Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Ferdig
Lage hovedrapport dokument	11.03.2022	Martin	Sette opp dokument	Ferdig
Revidere API design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må API revideres	Ferdig
Revidere DB design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må DB revideres	Ferdig
Begynne implementasjon Web	11.03.2022	Aleksander		Ferdig
Begynne implementasjon Mobil	11.03.2022	Tormod	Satt opp kart og grunnstruktur	Ferdig
Finne BLE tag til enhet	11.03.2022	Tormod og Martin	Gruppen trenger en BLE enhet til sentralen fra Telia	Scrappet
Hente Prosjekt endpoint	11.03.2022	Aleksander	Gruppen begynner utvikling	Ferdig
Hente stillasdel endpoint	11.03.2022	Martin	Gruppen begynner utvikling	Ferdig
Slette prosjekt endpoint	12.03.2022	Aleks		Ferdig
Finne ny sporingsenhet	15.03.2022	Tormod	Gruppen kan ikke bruke enhet fra Telia og må finne ny løsning	Ferdig
Begynne skriving på avsluttende rapport	18.03.2022	Martin	Gruppen ønsker å begynne på rapportskriving tidlig	Ikke påbegynt
Standardisere error-handling for API	16.03.2022	Aleksander	Aleksander ønsker å standardisere error handling for API-et	Ferdig
Implementere slette stillasdel funksjon	13.03	Martin		Ferdig
Implementere legge til prosjekt	14.03.2022	Aleksander		Ferdig

Implementere legge til, hent ut og slett bruker	19.03.2022	Aleksander		Ferdig
Sette opp API-Tester	24.03.2022	Martin	Martin ønsker å teste systemet	Ferdig
Legge inn søkefunksjonalitet i Mobil-Applikasjon	23.03.2022	Tormod		Ferdig
Legge inn funksjonalitet for overføring av stillasdeler	24.03.2022	Aleksander		Ferdig
Legge inn kartfunksjonalitet i mobil	26.03.2022	Tormod	Ønsker å oppgradere kartfunksjonalitet	Ferdig
Hente ut prosjektinfo og stillasdel info i mobil appen	27.03.2022	Tormod	Ønsker å hente data fra API	Ferdig
Implementere «Annotations» mobil-aplikasjon	28.03.2022	Tormod		Ferdig
Implementere Docker for deployering av løsning	01.04.2022	Martin	For å kunne spinne opp alle systemets komponenter trengs det en konteiner løsning som Docker	Ferdig
Oppdatere kart funskjonalitet i Web	01.04.2022	Aleksander		Ferdig
Starte med overføring av stillasdeler Mobil applikasjon	01.04.2022	Tormod		Ferdig
Implementere Docker Compose fil	06.04.2022	Martin		Må gjøres
Få ut data fra sporingsenhet	04.04.2022	Martin	Vi har nå fått sporingsenhet og Martin jobber med å få ut data	Ferdig
Legge inn egen fil for API info henting, mobil	02.04.2022	Tormod		Påbegynt
Legge inn Modaler for stillasdel og overføring	08.04.2022	Aleksander		Påbegynt
Implementere MQTT broker	08.04.2022	Martin	Ser på implementasjon av mqtt broker	Scrappet
Implementere filtrering i mobil	13.04.2022	Tormod		Jobbes med, tas videre til neste sprint
Legge til rette for geofence	14.04.2022	Aleksander		Ferdig

Sprint 12

Tidsrom: 18 April -22 April

Mål:

- **Få sporingsdata inn i API-et og oppdatere stillasdeler og byggeprosjekter:** Vi nærmer oss avslutning av utviklingsfasen og vi må derfor få på plass sporingsfunksjonalitet slik at vi kan teste systemet
- **Ferdigstille mobil og web applikasjon:** Vi nærmer oss avslutning av utviklingsfasen og må derfor bli ferdig med utvikling
- **Begynne på hovedrapport:** Dette har blitt utsatt for lenge og gruppen må begynne å skrive, planlegging av rapport oppsett må skje i denne sprinten!

Resultat:

- **Få sporingsdata inn i API-et og oppdatere stillasdeler og byggeprosjekter:** Vi får nå tag data inn i api-et og mangler kun å oppdatere informasjon i databasen for å ferdigstille sporingsteknologi
- **Ferdigstille mobil og web applikasjon:** Dette jobbes med, Aleksander og Tormod håper å bli ferdig til 01.05
- **Begynne på hovedrapport:** Rapportdokument er laget og gruppen jobber med en grunnstruktur

Referat:

Gruppen jobber godt, men innser at det kan bli knapt med tid og at dager må bli lenger for at gruppen skal komme i mål. Sporingsenhet er løst, men fysisk testing må fortsatt skje. Mobil og Web applikasjon jobbes med og Aleksander og Tormod håper å bli ferdig til fristen (01.05)

Backlog 22.04.2022

Mål	Start dato	Ansvarlig	Kommentar	Status
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Ferdig vi bruker Telia
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Ferdig
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Ferdig
Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Ferdig
Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Ferdig

Lage hovedrapport dokument	11.03.2022	Martin	Sette opp dokument	Ferdig
Revidere API design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må API revideres	Ferdig
Revidere DB design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må DB revideres	Ferdig
Begynne implementasjon Web	11.03.2022	Aleksander		Ferdig
Begynne implementasjon Mobil	11.03.2022	Tormod	Satt opp kart og grunnstruktur	Ferdig
Finne BLE tag til enhet	11.03.2022	Tormod og Martin	Gruppen trenger en BLE enhet til sentralen fra Telia	Scrappet
Hente Prosjekt endpoint	11.03.2022	Aleksander	Gruppen begynner utvikling	Ferdig
Hente stillasdel endpoint	11.03.2022	Martin	Gruppen begynner utvikling	Ferdig
Slette prosjekt endpoint	12.03.2022	Aleks		Ferdig
Finne ny sporingsenhet	15.03.2022	Tormod	Gruppen kan ikke bruke enhet fra Telia og må finne ny løsning	Ferdig
Begynne skriving på avsluttende rapport	18.03.2022	Martin	Gruppen ønsker å begynne på rapportskriving tidlig	Ikke påbegynt
Standardisere error-handling for API	16.03.2022	Aleksander	Aleksander ønsker å standardisere error handling for API-et	Ferdig
Implementere slette stillasdel funksjon	13.03	Martin		Ferdig
Implementere legge til prosjekt	14.03.2022	Aleksander		Ferdig
Implementere legge til, hent ut og slett bruker	19.03.2022	Aleksander		Ferdig
Sette opp API-Tester	24.03.2022	Martin	Martin ønsker å teste systemet	Ferdig
Legge inn søkefunksjonalitet i Mobil-Applikasjon	23.03.2022	Tormod		Ferdig
Legge inn funksjonalitet for overføring av stillasdeler	24.03.2022	Aleksander		Ferdig
Legge inn kartfunksjonalitet i mobil	26.03.2022	Tormod	Ønsker å oppgradere kartfunksjonalitet	Ferdig

Hente ut prosjektinfo og stillasdel info i mobil appen	27.03.2022	Tormod	Ønsker å hente data fra API	Ferdig
Implementere «Annotations» mobil-aplikasjon	28.03.2022	Tormod		Ferdig
Implementere Docker for deployering av løsning	01.04.2022	Martin	For å kunne spinne opp alle systemets komponenter trengs det en konteiner løsning som Docker	Ferdig
Oppdatere kart funskjonalitet i Web	01.04.2022	Aleksander		Ferdig
Starte med overføring av stillasdeler Mobil applikasjon	01.04.2022	Tormod		Ferdig
Implementere Docker Compose fil	06.04.2022	Martin		Må gjøres
Få ut data fra sporingsenhet	04.04.2022	Martin	Vi har nå fått sporingsenhet og Martin jobber med å få ut data	Ferdig
Legge inn egen fil for API info henting, mobil	02.04.2022	Tormod		Ferdig
Legge inn Modaler for stillasdel og overføring	08.04.2022	Aleksander		Påbegynt
Implementere MQTT broker	08.04.2022	Martin	Ser på implementasjon av mqtt broker	Scrappet
Implementere filtrering i mobil	13.04.2022	Tormod		Jobbes med, tas videre til neste sprint
Legge til rette for geofence	14.04.2022	Aleksander		Ferdig
Få sporingsdata inn i API	19.04.2022	Martin		Ferdig
Oppdatere byggeprosjekter og stillasdeler med data	21.04.2022	Martin		Påbegynt
Implementere caching i mobil	19.04.2022	Tormod		Påbegynt
Oppdatere data i web applikasjon	18.04.2022	Aleksander		Påbegynt
Ferdigstilling av prosjekt i mobil applikasjon	19.04.2022	Tormod		Påbegynt
Implementere gateway endpoint i API	19.04.2022	Martin	Api-et trenger et gateway endpoint da enhetene må inn i databasen	Påbegynt

Sprint 13

Tidsrom: 23 April -29 April

Mål:

- **Ferdigstille all utvikling:** Gruppen avslutter utviklingsfasen og må bli ferdig med utvikling.
- **Begynne rapportskriving:** Gruppen ønsker å begynne skriving under denne Sprinten

Resultat:

- **Ferdigstille all utvikling:** Gruppen er nærmest ferdig med all utvikling, gruppen har funksjonalitet de ønsker å implementere spesielt brukerinnlogging.
- **Begynne rapportskriving:** Gruppen har lagt opp rapporten, men har ikke begynt selve skriveprosessen

Referat:

Gruppen er noe stresset da vi innser at de blir knapt med tid og innlevering av oppgaven nærmer seg. Gruppen tror derimot at vi skal komme i mål. Sporingsenheten fungere som forventet og skal testes i neste sprint.

Backlog 29.04.2022

Mål	Start dato	Ansvarlig	Kommentar	Status
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Ferdig vi bruker Telia
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Ferdig
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Ferdig
Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Ferdig
Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Ferdig
Lage hovedrapport dokument	11.03.2022	Martin	Sette opp dokument	Ferdig
Revidere API design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må API revideres	Ferdig

Revidere DB design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må DB revideres	Ferdig
Begynne implementasjon Web	11.03.2022	Aleksander		Ferdig
Begynne implementasjon Mobil	11.03.2022	Tormod	Satt opp kart og grunnstruktur	Ferdig
Finne BLE tag til enhet	11.03.2022	Tormod og Martin	Gruppen trenger en BLE enhet til sentralen fra Telia	Scrappet
Hente Prosjekt endpoint	11.03.2022	Aleksander	Gruppen begynner utvikling	Ferdig
Hente stillasdel endpoint	11.03.2022	Martin	Gruppen begynner utvikling	Ferdig
Slette prosjekt endpoint	12.03.2022	Aleks		Ferdig
Finne ny sporingsenhet	15.03.2022	Tormod	Gruppen kan ikke bruke enhet fra Telia og må finne ny løsning	Ferdig
Begynne skriving på avsluttende rapport	18.03.2022	Martin	Gruppen ønsker å begynne på rapportskriving tidlig	Ferdig
Standardisere error-handling for API	16.03.2022	Aleksander	Aleksander ønsker å standardisere error handling for API-et	Ferdig
Implementere slette stillasdel funksjon	13.03	Martin		Ferdig
Implementere legge til prosjekt	14.03.2022	Aleksander		Ferdig
Implementere legge til, hent ut og slett bruker	19.03.2022	Aleksander		Ferdig
Sette opp API-Tester	24.03.2022	Martin	Martin ønsker å teste systemet	Ferdig
Legge inn søkefunksjonalitet i Mobil-Applikasjon	23.03.2022	Tormod		Ferdig
Legge inn funksjonalitet for overføring av stillasdeler	24.03.2022	Aleksander		Ferdig
Legge inn kartfunksjonalitet i mobil	26.03.2022	Tormod	Ønsker å oppgradere kartfunksjonalitet	Ferdig
Hente ut prosjektinfo og stillasdel info i mobil appen	27.03.2022	Tormod	Ønsker å hente data fra API	Ferdig

Implementere «Annotations» mobil-aplikasjon	28.03.2022	Tormod		Ferdig
Implementere Docker for deployering av løsning	01.04.2022	Martin	For å kunne spinne opp alle systemets komponenter trengs det en konteiner løsning som Docker	Ferdig
Oppdatere kart funskjonalitet i Web	01.04.2022	Aleksander		Ferdig
Starte med overføring av stillasdeler Mobil applikasjon	01.04.2022	Tormod		Ferdig
Implementere Docker Compose fil	06.04.2022	Martin		Må gjøres
Få ut data fra sporingsenhet	04.04.2022	Martin	Vi har nå fått sporingsenhet og Martin jobber med å få ut data	Ferdig
Legge inn egen fil for API info henting, mobil	02.04.2022	Tormod		Ferdig
Legge inn Modaler for stillasdel og overføring	08.04.2022	Aleksander		Jobbes med ta
Implementere MQTT broker	08.04.2022	Martin	Ser på implementasjon av mqtt broker	Scrappet
Implementere filtrering i mobil	13.04.2022	Tormod		Jobbes med, tas videre til neste sprint
Legge til rette for geofence	14.04.2022	Aleksander		Ferdig
Få sporingsdata inn i API	19.04.2022	Martin		Ferdig
Oppdatere byggeprosjekter og stillasdeler med data	21.04.2022	Martin		Ferdig
Implementere caching i mobil	19.04.2022	Tormod		Ferdig
Oppdatere data i web applikasjon	18.04.2022	Aleksander		Ferdig
Ferdigstilling av prosjekt i mobil applikasjon	19.04.2022	Tormod		Ferdig
Implementere gateway endpoint i API	19.04.2022	Martin	Api-et trenger et gateway endpoint da enhetene må inn i databasen	Påbegynt
Ferdigstille oppdatering av	25.04.2022	Martin		Påbegynt

posisjon på stillasdeler				
Implementere login funksjonalitet	28.04.2022	Aleksander	Systemet bør ha login funksjonalitet	Påbegynt
Gjenoppta arbeid med filtrering mobil applikasjon	27.04.2022	Tormod	Gjenopptar arbeid på filtrering	Påbegynt

Sprint 14

Tidsrom: 30 April- 06 Mai

Mål:

- **Skrive rapport:** Gruppen ønsker å fullføre kapittel 1-5 i rapporten innen slutten av sprinten
- **Programmering:** Dette gjøres utenom arbeidstimer på eget initiativ, ingen mål kun finpuss og forbedring av eksisterende funksjonalitet

Resultat:

- **Skrive rapport:**Gruppen har skrevet kapitlene og leverer førsteutkast til veileder den 08.05
- **Programmering:** Kodekommentering og småpirk rettes opp når det er tid til dette

Referat:

Gruppen er i avslutningsfasen av oppgaven og skriver for det meste på bachelorrapporten. Gruppen ønsker å være så godt som ferdig med rapporten den 16.05 slik at vi har 3 dager på å lese gjennom og ferdigstille rapporten. Det må også skrives en readme fil på gitlab.

Backlog 06.05.2022

Mål	**Start dato**	**Ansvarlig**	**Kommentar**	**Status**
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Ferdig vi bruker Telia
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Ferdig
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Ferdig
Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Ferdig

Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Ferdig
Lage hovedrapport dokument	11.03.2022	Martin	Sette opp dokument	Ferdig
Revidere API design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må API revideres	Ferdig
Revidere DB design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må DB revideres	Ferdig
Begynne implementasjon Web	11.03.2022	Aleksander		Ferdig
Begynne implementasjon Mobil	11.03.2022	Tormod	Satt opp kart og grunnstruktur	Ferdig
Finne BLE tag til enhet	11.03.2022	Tormod og Martin	Gruppen trenger en BLE enhet til sentralen fra Telia	Scrappet
Hente Prosjekt endpoint	11.03.2022	Aleksander	Gruppen begynner utvikling	Ferdig
Hente stillasdel endpoint	11.03.2022	Martin	Gruppen begynner utvikling	Ferdig
Slette prosjekt endpoint	12.03.2022	Aleks		Ferdig
Finne ny sporingsenhet	15.03.2022	Tormod	Gruppen kan ikke bruke enhet fra Telia og må finne ny løsning	Ferdig
Begynne skriving på avsluttende rapport	18.03.2022	Martin	Gruppen ønsker å begynne på rapportskriving tidlig	Ferdig
Standardisere error-handling for API	16.03.2022	Aleksander	Aleksander ønsker å standardisere error handling for API-et	Ferdig
Implementere slette stillasdel funksjon	13.03	Martin		Ferdig
Implementere legge til prosjekt	14.03.2022	Aleksander		Ferdig
Implementere legge til, hent ut og slett bruker	19.03.2022	Aleksander		Ferdig
Sette opp API-Tester	24.03.2022	Martin	Martin ønsker å teste systemet	Ferdig
Legge inn søkefunksjonalitet i Mobil-Applikasjon	23.03.2022	Tormod		Ferdig
Legge inn funksjonalitet for overføring av stillasdeler	24.03.2022	Aleksander		Ferdig

Legge inn kartfunksjonalitet i mobil	26.03.2022	Tormod	Ønsker å oppgradere kartfunksjonalitet	Ferdig
Hente ut prosjektinfo og stillasdel info i mobil appen	27.03.2022	Tormod	Ønsker å hente data fra API	Ferdig
Implementere «Annotations» mobil-aplikasjon	28.03.2022	Tormod		Ferdig
Implementere Docker for deployering av løsning	01.04.2022	Martin	For å kunne spinne opp alle systemets komponenter trengs det en konteiner løsning som Docker	Ferdig
Oppdatere kart funskjonalitet i Web	01.04.2022	Aleksander		Ferdig
Starte med overføring av stillasdeler Mobil applikasjon	01.04.2022	Tormod		Ferdig
Implementere Docker Compose fil	06.04.2022	Martin		Må gjøres
Få ut data fra sporingsenhet	04.04.2022	Martin	Vi har nå fått sporingsenhet og Martin jobber med å få ut data	Ferdig
Legge inn egen fil for API info henting, mobil	02.04.2022	Tormod		Ferdig
Legge inn Modaler for stillasdel og overføring	08.04.2022	Aleksander		Jobbes med ta
Implementere MQTT broker	08.04.2022	Martin	Ser på implementasjon av mqtt broker	Scrappet
Implementere filtrering i mobil	13.04.2022	Tormod		Ferdig
Legge til rette for geofence	14.04.2022	Aleksander		Ferdig
Få sporingsdata inn i API	19.04.2022	Martin		Ferdig
Oppdatere byggeprosjekter og stillasdeler med data	21.04.2022	Martin		Ferdig
Implementere caching i mobil	19.04.2022	Tormod		Ferdig
Oppdatere data i web applikasjon	18.04.2022	Aleksander		Ferdig
Ferdigstilling av prosjekt i mobil applikasjon	19.04.2022	Tormod		Ferdig

Implementere gateway endpoint i API	19.04.2022	Martin	Api-et trenger et gateway endpoint da enhetene må inn i databasen	Ferdig
Ferdigstille oppdatering av posisjon på stillasdeler	25.04.2022	Martin		Ferdig
Implementere login funksjonalitet	28.04.2022	Aleksander	Systemet bør ha login funksjonalitet	Ferdig
Gjenoppta arbeid med filtrering mobil applikasjon	27.04.2022	Tormod	Gjenopptar arbeid på filtrering	Ferdig
Ferdigstille rapport	02.05.2022	Alle gruppemedlemmer		Påbegynt
Kommentere kode	05.05.2022	Alle gruppemedlemmer		Påbegynt

Sprint 15

Tidsrom: 07 Mai- 13 Mai

Mål:

- **Ferdigstille rapport:** Gruppen ønsker å ha skrevet alle kapitler innen mandag 16.05

Resultat:

- **Skrive rapport:** Gruppen har levert andreutkast til veileder den 12.05 og planlegger et avsluttende utkast den 18.05

Referat:

Dette er gruppens siste scrumsprint og all fokus ligger på rapportskriving

Backlog 13.05.2022

Mål	**Start dato**	**Ansvarlig**	**Kommentar**	**Status**
Prosjektplan	11.01.2022	Alle gruppemedlemmer	Ferdig og levert	Ferdig
Sporingsteknologi-Dokument	17.01.2022	Alle gruppemedlemmer	Dokument er laget og informasjon samles	Ferdig
Sporingsenhet	17.01.2022	Alle gruppemedlemmer	Gruppemedlemmer i møte med Telia den 15.03.2022	Ferdig vi bruker Telia
Lage fremlegg for MyLift 25.02	14.02.2022	Martin	Gruppen skal legge frem en plan for oppdragsgiver	Ferdig
Fastsette system-arkitektur	14.02.2022	Aleksander og Tormod	Overordnet system-arkitektur er på plass	Ferdig
Planlegge programmeringsspråk	14.02.2022	Alle gruppemedlemmer	Etter definisjon av arkitektur må vi lande på programmeringsspråk	Ferdig
Røft design av web-applikasjon	03.03.2022	Aleksander	Lage design UI til web-applikasjon	Ferdig

Røft design av mobil-applikasjon	03.03.2022	Tormod	Lage design UI til mobil-applikasjon	Ferdig
Design av fysisk løsning	03.03.2022	Martin	Planlegge hvordan sporingen skal fungere på byggeplass	Ferdig
Lage hovedrapport dokument	11.03.2022	Martin	Sette opp dokument	Ferdig
Revidere API design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må API revideres	Ferdig
Revidere DB design	11.03.2022	Martin og Aleksander	Etter dialog med oppdragsgiver må DB revideres	Ferdig
Begynne implementasjon Web	11.03.2022	Aleksander		Ferdig
Begynne implementasjon Mobil	11.03.2022	Tormod	Satt opp kart og grunnstruktur	Ferdig
Finne BLE tag til enhet	11.03.2022	Tormod og Martin	Gruppen trenger en BLE enhet til sentralen fra Telia	Scrappet
Hente Prosjekt endpoint	11.03.2022	Aleksander	Gruppen begynner utvikling	Ferdig
Hente stillasdel endpoint	11.03.2022	Martin	Gruppen begynner utvikling	Ferdig
Slette prosjekt endpoint	12.03.2022	Aleks		Ferdig
Finne ny sporingsenhet	15.03.2022	Tormod	Gruppen kan ikke bruke enhet fra Telia og må finne ny løsning	Ferdig
Begynne skriving på avsluttende rapport	18.03.2022	Martin	Gruppen ønsker å begynne på rapportskriving tidlig	Ferdig
Standardisere error-handling for API	16.03.2022	Aleksander	Aleksander ønsker å standardisere error handling for API-et	Ferdig
Implementere slette stillasdel funksjon	13.03	Martin		Ferdig
Implementere legge til prosjekt	14.03.2022	Aleksander		Ferdig
Implementere legge til, hent ut og slett bruker	19.03.2022	Aleksander		Ferdig
Sette opp API-Tester	24.03.2022	Martin	Martin ønsker å teste systemet	Ferdig
Legge inn søkefunksjonalitet i Mobil-Applikasjon	23.03.2022	Tormod		Ferdig
Legge inn funksjonalitet for	24.03.2022	Aleksander		Ferdig

overføring av stillasdeler				
Legge inn kartfunksjonalitet i mobil	26.03.2022	Tormod	Ønsker å oppgradere kartfunksjonalitet	Ferdig
Hente ut prosjektinfo og stillasdel info i mobil appen	27.03.2022	Tormod	Ønsker å hente data fra API	Ferdig
Implementere «Annotations» mobil-aplikasjon	28.03.2022	Tormod		Ferdig
Implementere Docker for deployering av løsning	01.04.2022	Martin	For å kunne spinne opp alle systemets komponenter trengs det en konteiner løsning som Docker	Ferdig
Oppdatere kart funskjonalitet i Web	01.04.2022	Aleksander		Ferdig
Starte med overføring av stillasdeler Mobil applikasjon	01.04.2022	Tormod		Ferdig
Implementere Docker Compose fil	06.04.2022	Martin		Må gjøres
Få ut data fra sporingsenhet	04.04.2022	Martin	Vi har nå fått sporingsenhet og Martin jobber med å få ut data	Ferdig
Legge inn egen fil for API info henting, mobil	02.04.2022	Tormod		Ferdig
Legge inn Modaler for stillasdel og overføring	08.04.2022	Aleksander		Jobbes med ta
Implementere MQTT broker	08.04.2022	Martin	Ser på implementasjon av mqtt broker	Scrappet
Implementere filtrering i mobil	13.04.2022	Tormod		Ferdig
Legge til rette for geofence	14.04.2022	Aleksander		Ferdig
Få sporingsdata inn i API	19.04.2022	Martin		Ferdig
Oppdatere byggeprosjekter og stillasdeler med data	21.04.2022	Martin		Ferdig
Implementere caching i mobil	19.04.2022	Tormod		Ferdig
Oppdatere data i web applikasjon	18.04.2022	Aleksander		Ferdig
Ferdigstilling av prosjekt i mobil applikasjon	19.04.2022	Tormod		Ferdig

Implementere gateway endpoint i API	19.04.2022	Martin	Api-et trenger et gateway endpoint da enhetene må inn i databasen	Ferdig
Ferdigstille oppdatering av posisjon på stillasdeler	25.04.2022	Martin		Ferdig
Implementere login funksjonalitet	28.04.2022	Aleksander	Systemet bør ha login funksjonalitet	Ferdig
Gjenoppta arbeid med filtrering mobil applikasjon	27.04.2022	Tormod	Gjenopptar arbeid på filtrering	Ferdig
Ferdigstille rapport	02.05.2022	Alle gruppemedlemmer		Jobbes med
Kommentere kode	05.05.2022	Alle gruppemedlemmer		Jobbes med

Appendix J Timeliste

Dato	Felles	Felles opplegg	Felles innhold	Individuelt innhold	Tormod	Aleksander	Martin	Total
20.12.2021	1	Bli-kjent møte med veileder i MyLift						3
10.01.2022	3		Oppsett av Discord, GitLab, enkelte dokumenter. Generell diskusjon.					9
11.01.2022	5	Lynkurs og spørretime bacheloroppgaven. Oppstarts-/veiledningsmøte med veileder på NTNU.	Videre oppsett av nødvendig dokumentasjon og hjelpe-programvare (Microsoft Planner etc.)					15
12.01.2022	6		Ferdigstilt avtale MyLift og NTNU. Ferdigstilt timeliste. Ferdigstilt gruppeavtale(signert). Jobbet med Prosjektplan.					18
13.01.2022	6		Spesifisert førsteutkast til requirements for både software og hardware. Funnet utviklingsmodell (SCRUM). Skrevet på punkt 4.3 i prosjektplan. Laget idé- og beslutnings log/dokument.					18
14.01.2022	5	Møte med veileder i MyLift	Begynt på Gantt diagram. Begynt på Risikoanalyse på prosjektnivå. Skrevet om inspeksjoner og testing. Skrevet hvordan vi planlegger å gjennomføre SCRUM. Planlagt møte med veileder på NTNU.					15
15.01.2022								0
16.01.2022				Aleksander: Sett på relevant syntax til swift.		1,5		1,5
17.01.2022	7	Møte med veileder NTNU.	Jobbing med prosjektplanen. Ferdigstilling gantt diagram.					21
18.01.2022	7		Ferdigstilling av prosjektplanen. Research sporingsteknologi. Lagd møtereferat template. Skrevet møtereferat. Laget plan for beslutninger.					21
19.01.2022	6		Sett på ulike typer teknologi som skal brukes til sporingsenheten (LPWAN, RFID, Bluetooth Low Energy). Konstruert mail til Faglærer om evt hjelp til oppgaven.					18

20.01.2022	6		Vært i kontakt med Telia angående LPWAN teknologi. Lest igjennom prosjektplanen. Laget presentasjon til møte med mylift. Sendt mail til faglærere (angående hjelp til løsning av database oppgaven).					18
21.01.2022	4	Møte med veileder i MyLift	Sett på mulige løsninger av sporingsenheter. Sendt mail til Armada. Diskutert mulig løsningsalternativer(nfc, rfid, lpwan)	Gått gjennom Prosjektplan	1			13
22.01.2022								0
23.01.2022				Sett på swift: MapKit, APIcalls, syntax og programmering på mobil		3		3
24.01.2022	5		Sett på sporingsteknologi. Sendt mail til Haki. Lagde systemarkitektur og use-case diagrammer. Laget et dokument på sporingsteknologer.					15
25.01.2022	6	Møte med veileder NTNU. Møte med Haki.	Arbeidet med prosjektplan (kommentarer fra frode.) Avgrenset problemstillingen mer til et logistikkproblem. Satt oss mer inn på teknologi som kan brukes.					18
26.01.2022	6		Sent mail til tyskland. Forbedret møte med Armada. Fullørt prosjektplan. Fortsatt med sporingsteknologi.					18
27.01.2022	7	Møte med Armada	Rettet på prosjektplanen. Laget presentasjon til mylift. Sett på sporingsteknologi. Skrevet om div sporingsteknologi. Sendt inn prosjektavtalen .					21
28.01.2022	4	Møte med veileder i MyLift	Sett på oppstart til database implementering. Levert prosjektplan. Sett på LPWAN.					12
29.01.2022								0
30.01.2022								0
31.01.2022								0
01.02.2022	0,5	Møte med veileder NTNU.						1,5

02.02.2022	5		API Dokumentasjon. Ferdig med Sporingsteknologi dokument. Snakket med Telia.					15
03.02.2022	5		Startet på Conseptual Model (Database). Forberedet oss til Div møter.					15
04.02.2022	5	Møte med veileder i MyLift	Sett på LPWAN. Sett på swift.					15
05.02.2022								0
06.02.2022								0
07.02.2022	6	Møte med Telia	Sett på BLE.					18
08.02.2022	7	Besøk hos HAKI						21
09.02.2022	0							0
10.02.2022	5		Sett på teknologier. I dialog med Disruptive Technologies. I dialog med Telia.					15
11.02.2022	6	Møte med veileder i MyLift	API Dokumentasjon. Dokumnetasjon av fremgang. Database modellering					18
12.02.2022								0
13.02.2022								0
14.02.2022								0
15.02.2022	0,5	Møte med veileder NTNU.						1,5
16.02.2022								0
17.02.2022	3	Møte med Abax	Sett på forskjellige muligheter med tanke på bareller.					9
18.02.2022	7	Møte med veileder MyLift	Forberedt møte MyLift. Jobbet med databasedesign. Skiftet fokus til sporing av bareller framfor hver individuelle stillasdel. Sett på mulighet for bruk av computer vision. Kontaktet lærer computer vision. Skrevet statusrapport.					21
19.02.2022								0
20.02.2022								0
21.02.2022	6	Møte med Frode	Oppstart med SCRUM sprint 1 Laget persona og user stories Startet med UI design på web og mobil app (kun prototyp ikke kode)					18
22.02.2022	7		Fortsatt med UI design mobil og web Sendt oppfølgingsmail til Telia ang pris Skrevet om stillas samt sporing i sporingsteknologidokument		0,5			21,5

23.02.2022	3,50		Fortsatt med UI design mobil og web. Kommunisert med telia angående LPWAN enhet. Skrevet om sporingsteknologi.	Aleksander: Webdesign Tormod: Mobil applikasjon design / prototype	3	3		16,5
24.02.2022	8		Ferdigstilt design på web. Fortsatt design på mobil. Laget presentasjon til myLift besøk. Sett på React. Mer dialog med telia. Skrevet på sporingsteknologi dokument .	Aleksander: React Tormod: Mobil applikasjon design / prototype Martin Rapportskriving og møtereferatskriving	4,5	2	2	32,5
25.02.2022	7	Tur til oppdragsgiver						21
26.02.2022								0
27.02.2022								0
28.02.2022								0
01.03.2022								0
02.03.2022	1,5	Lynkurs om rapport	Selvstendig arbeid	Aleksander: React, conceptual Model	6	7		17,5
03.03.2022	7,5		Diskusjon rundt valg av programmeringsspråk Scrum møte og planlegging av sprint 2 Research BLE Jobb med sporingsrapport Referater					22,5
04.03.2022	7	Møte med veileder MyLift	Så på database alternativer (noSQL, SQL). Startet endret design på web. Gjorde endringer på API design					21
05.03.2022								0
06.03.2022	3		Sett på noSQL					9
07.03.2022	9		Re-designet API til å imøtekomme ny sporingsmetode. Re-designet DB til å være NoSQL istedenfor MySQL. Vært i kontakt med faglærere angående DB og BLE tags. Begynt å se på GDPR med hensyn på persondata behandling og prosessering.					27
08.03.2022	6		Begynt på API utvikling Begynt på WEB også Begynt på/lære mobil utvikling Mail med diverse aktører					18
09.03.2022	4	Møte Rune angående Database. Møte med Telia angående Gateway.	Sett på mobil app utvikling. Fortsatt med Webutvikling. Laget skisse på database struktur.					12
10.03.2022								0

11.03.2022	1	Møte med veileder MyLift Møte med IoT fyr - Mohammed	Lære mobil programmering	Tormod: Mobil prog. Aleksander: Web programmering	3,5	2,5	1	10
12.03.2022				Tormod: Mobil prog.	4,5			4,5
13.03.2022				Tormod: Mobil prog.	2			2
14.03.2022	0,5	Møte med Frode						1,5
15.03.2022	1	Møte med Telia						3
16.03.2022	3,5		Justere DB design Justere API design					10,5
17.03.2022	6	Møte med Knut	API programmering og BLE lesing					18
18.03.2022								0
19.03.2022								0
20.03.2022								0
21.03.2022								0
22.03.2022								0
23.03.2022								0
24.03.2022	8	x-session	API og Mobilprogrammering					24
25.03.2022	8		API og Mobilprogrammering					24
26.03.2022	5		API og Mobilprogrammering					15
27.03.2022				Martin startet på API testing Aleksander: API	4	5	2	11
28.03.2022	9		API utvikling og testing. Mobilutvikling					27
29.03.2022	8		Api utvikling og Mobilprogging					24
30.03.2022	7		Api utvikling og Mobilprogging					21
31.03.2022	7		Api utvikling, Mobilprogrammering og rapportskriving					21
01.04.2022	7		Api utvikling, Mobilprogrammering og rapportskriving					21
02.04.2022			Api utvikling			4		4
03.04.2022			Api utvikling			4		4
04.04.2022	7		Api utvikling, Mobilprogrammering og Webutvikling					21
05.04.2022			Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering		8	7	7	22
06.04.2022	10		Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering					30
07.04.2022	9		Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering					27
08.04.2022	4							12
09.04.2022								0

10.04.2022							0
11.04.2022				3	4		7
12.04.2022					4		4
13.04.2022					4		4
14.04.2022				4,5			4,5
15.04.2022							0
16.04.2022				3	3		6
17.04.2022							0
18.04.2022	8		Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering				24
19.04.2022	8		Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering	1			25
20.04.2022	8		Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering	1			25
21.04.2022	8		Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering				24
22.04.2022							0
23.04.2022							0
24.04.2022							0
25.04.2022	12		Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering	1			37
26.04.2022	12		Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering				36
27.04.2022	8		Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering	3			27
28.04.2022	10		Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering	5			35
29.04.2022	7		Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering				21
30.04.2022	10		Api utvikling, Mobilprogrammering, Webutvilking og Docker/server deployering				30
01.05.2022	12		Rapportskriving				36
02.05.2022	12		Rapportskriving	2,5			38,5
03.05.2022	7,5		Rapportskriving	5	5		32,5
04.05.2022	12		Rapportskriving				36
05.05.2022	12		Rapportskriving	2,5			38,5
06.05.2022	11		Rapportskriving				33

07.05.2022	10		Rapportskriving					30
08.05.2022	10		Rapportskriving					30
09.05.2022	10		Rapportskriving		2			32
10.05.2022	10		Rapportskriving					30
11.05.2022	10		Rapportskriving		1,5			31,5
12.05.2022	10		Rapportskriving		2			32
13.05.2022	8		Rapportskriving					24
14.05.2022			Rapportskriving			8	8	16
15.05.2022	3		Rapportskriving		7	7		23
16.05.2022	10		Rapportskriving		1			31
17.05.2022	5		Rapportskriving					15
18.05.2022	17		Rapportskriving					51
19.05.2022	13		Rapportskriving					39
20.05.2022								0
							Total tid	1910

Appendix K Kodebase og prototyping

- GitLab Kodebase: https://git.gvk.idi.ntnu.no/aleksaab/stillastracker
- FIGMA Prototyping mobil: https://www.figma.com/file/0c0r0f9gohqgFvavt482qY/Mobil-applikasjon?node-id=0%3A1
- FIGMA prototyping web: https://www.figma.com/file/IgI5xGoYXxjbwiJXP9EmRm/Web-L%C3%B8sning?node-id=0%3A1
- FIGMA Personas: https://www.figma.com/community/file/1109145345962818879

Appendix L Relasjonsdatabase modell og endelig dokumentdatabase design

Spørsmål til Rune

1. Vi ønsker en liste med gps posisjon og tidspunkt for å kunne se tidligere posisjoner for stillas. HVordan blir dette i henhold til normalform
2. Ville du splittet oppp person eller ikke? Vi ser ikke helt poenget med å ha en egen person entity
3. Kan mann ha flere tilknyttninger imellom samme entity? Typ stillasdel knyttes til prosjekt men både prosjekt og stillasdel er knyttet til lokasjon

Informasjon om modellen:

Systemet omhandler logistikk rundt stillasdeler fra bestillinger blir behandlet på lager til det blir kjørt ut til byggeprosjekt og returnert tilbake til lager.

Stillasdeler vil bli delt opp i **X** antall deler, om ønskelig vil de små delene gå i bareller (en entity av disse delene vil tilsvare 1 kasse med delen)

Stillasdeler vil alltid ha en lokasjon som vil bli oppdatert **X** antall ganger om dagen denne lokasjonen vil bli hentet fra sporeren knyttet til stillasdelen,

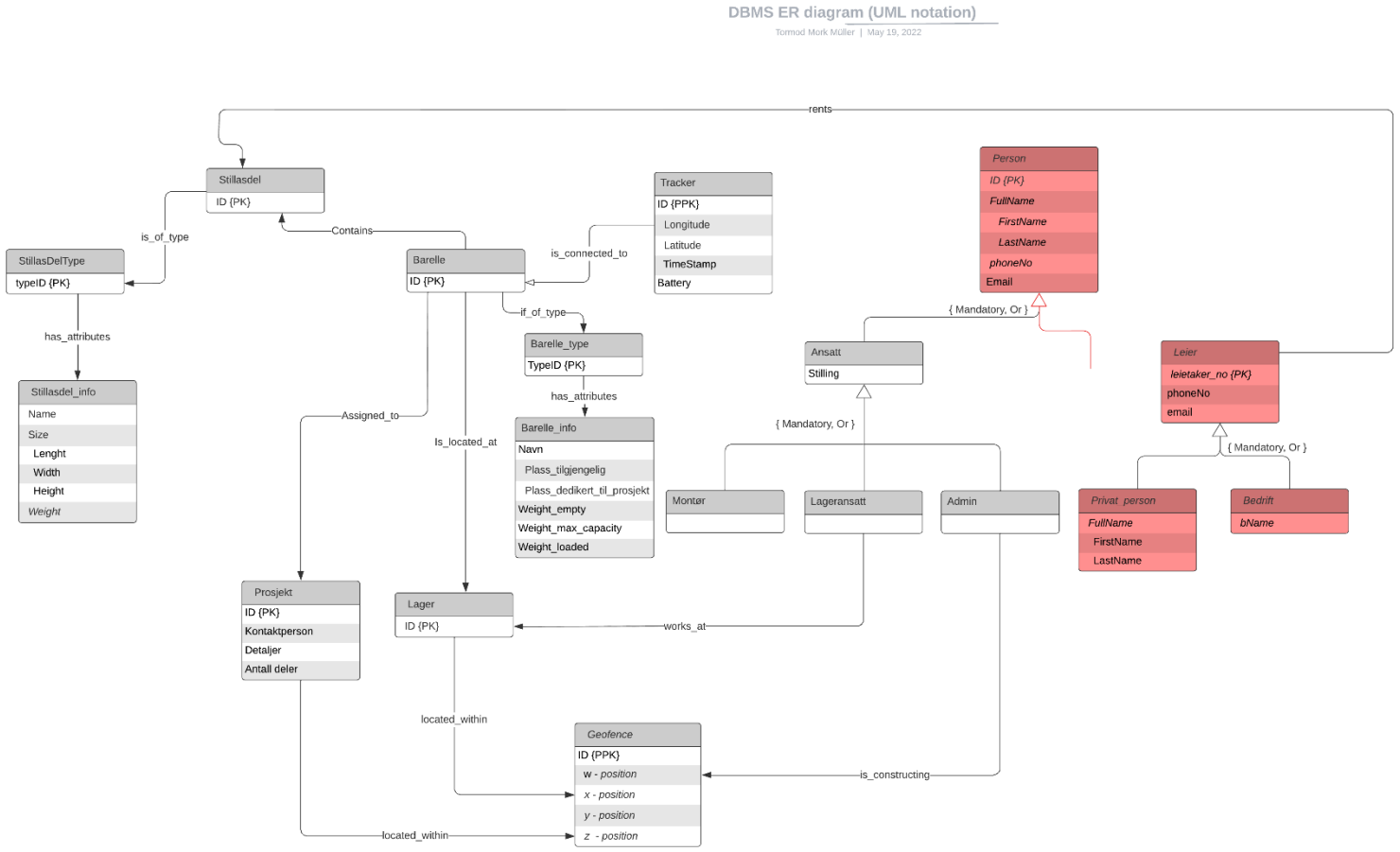
Lokasjonen vil bli hentet fra en Tracker som er knyttet til hver stillasdel. Hver tracker vil ha en lokasjon i tillegg til batteritid.

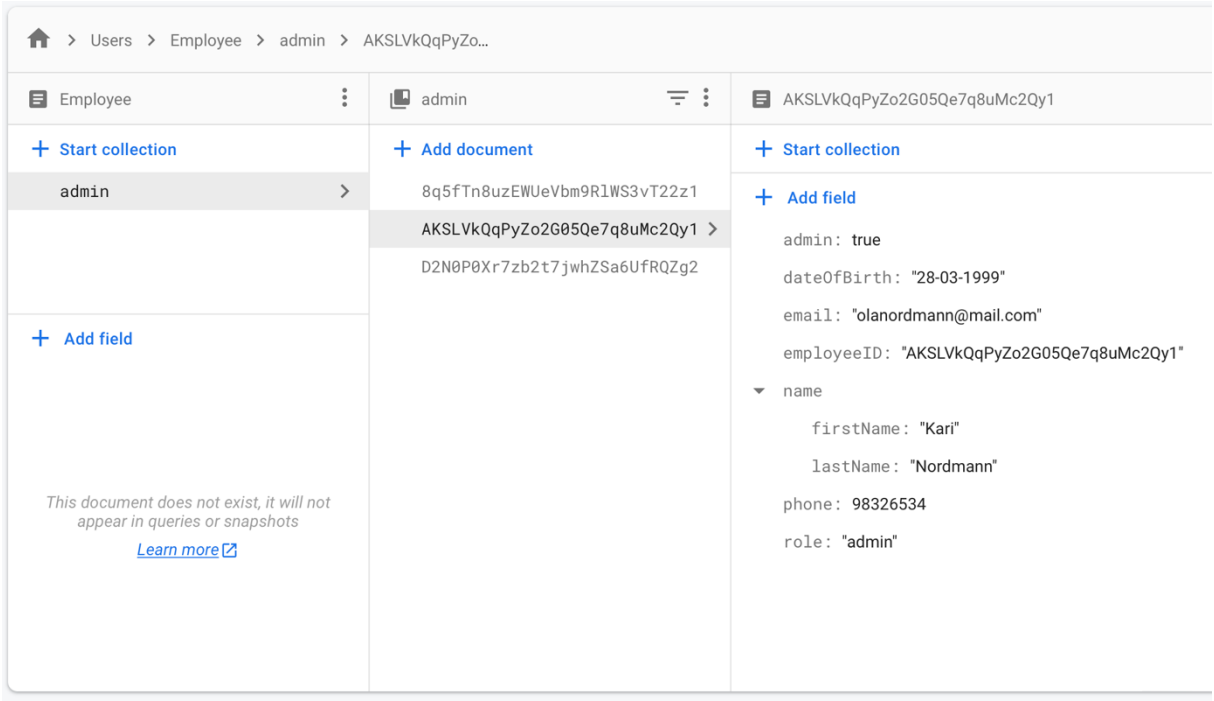
Delers lokasjon vil bli sammenlignet med opsatte geofence (samling av koordinater) som er satt opp rundt lageret samt aktive byggeprosjekter. Delers lokasjon vil bli lagret i en History entity slik at administrator skal kunne sjekke hvor deler har vært tidligere

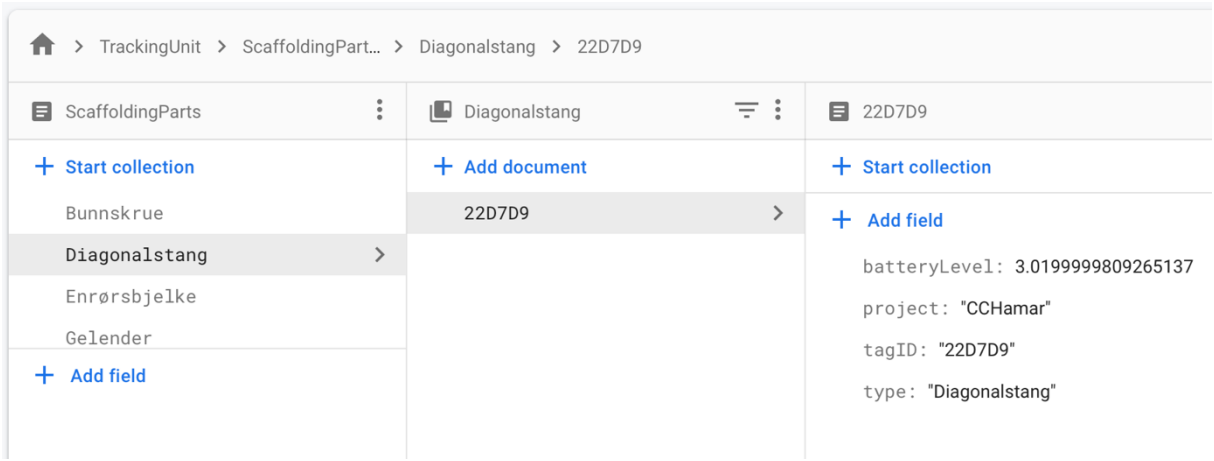
Byggeprosjekter vil inneholde en liste med stillasdeler, en prosjekt id kontaktperson på byggeplass og evt annen informasjon om prosjektet.

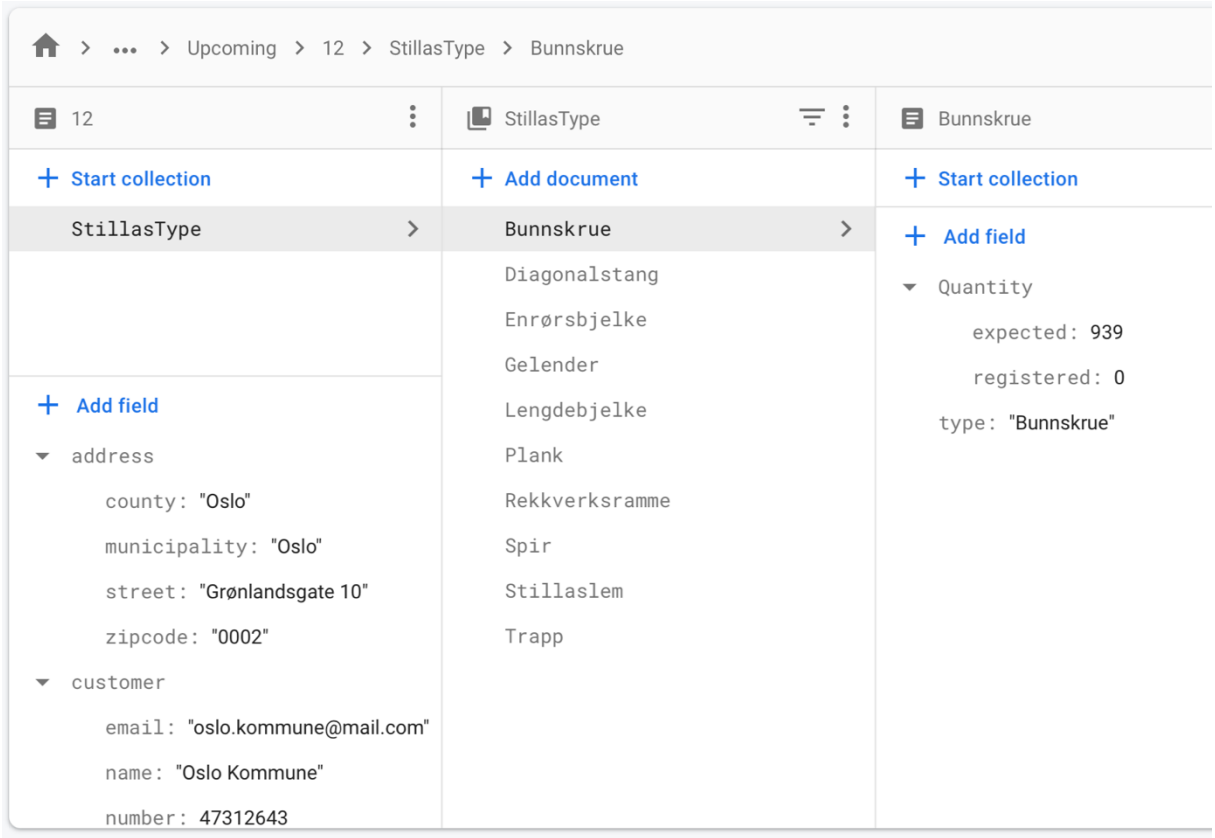
Systemet vil brukes av **X** brukere:
En admin som vil kunne legge inn nye prosjekter, sjekke historien på stillasdeler, slette og legge til deler og se en oversikt over hvilke deler som befinner seg på hvilke prosjekter.
En Lageransatt som skal kunne se oversikten over hvilke stillasdeler som befinner seg hvor.
En monteringsansvarlig som kun skal kunne se hvor stillasdeler befinner seg

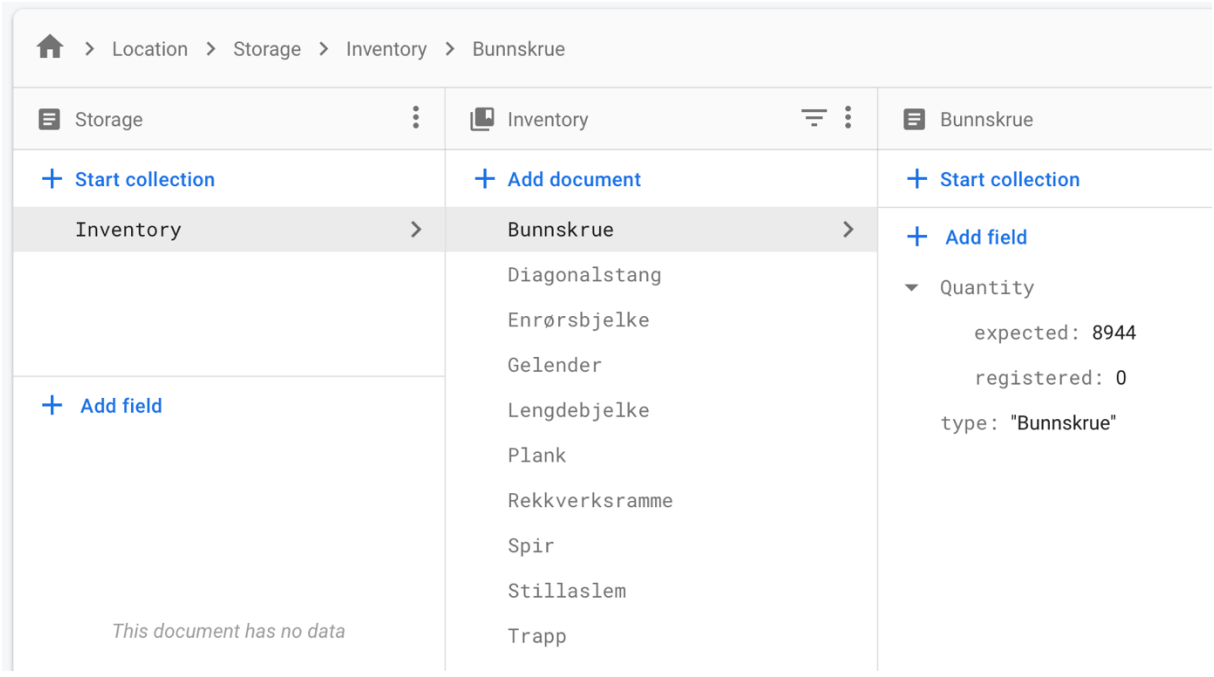
Det skal knyttes en leier til hvert prosjekt slik at admin lett skal kunne se hvilken leier som er ansvarlig for eventuelle bortkomne stillasdeler

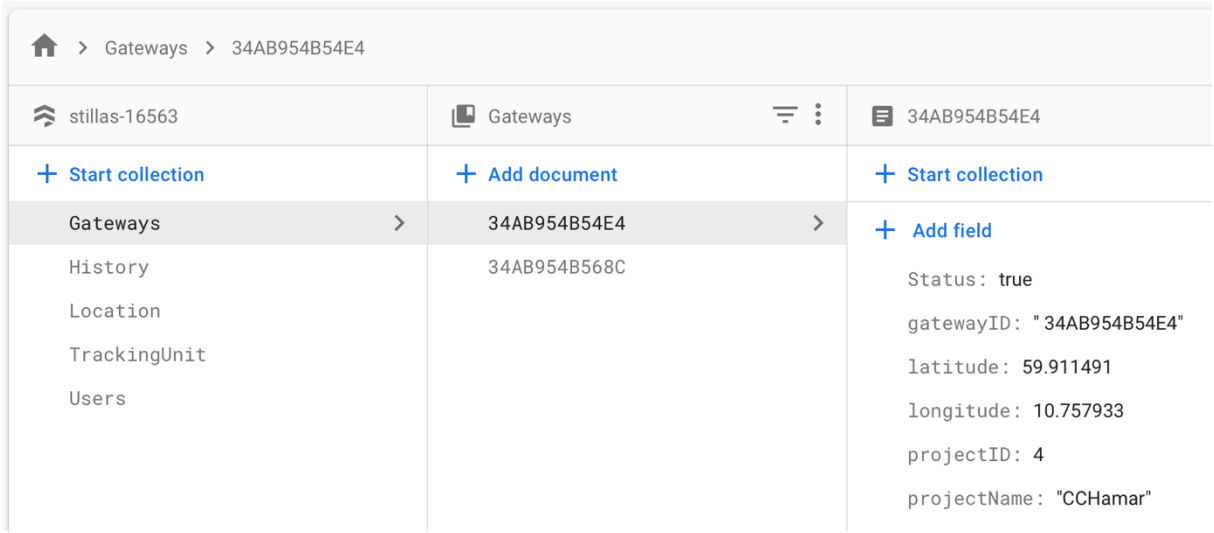



Users > Employee > admin > AKSLVkQqPyZo...
Employee
Start collection
admin
Add field
This document does not exist, it will not appear in queries or snapshots
Learn more
admin
Add document
8q5fTn8uzEWUeVbm9RlWS3vT22z1
AKSLVkQqPyZo2G05Qe7q8uMc2Qy1
D2N0P0Xr7zb2t7jwhZSa6UfRQZg2
AKSLVkQqPyZo2G05Qe7q8uMc2Qy1
Start collection
Add field
admin: true
dateOfBirth: "28-03-1999"
email: "olanordmann@mail.com"
employeeID: "AKSLVkQqPyZo2G05Qe7q8uMc2Qy1"
name
firstName: "Kari"
lastName: "Nordmann"
phone: 98326534
role: "admin"

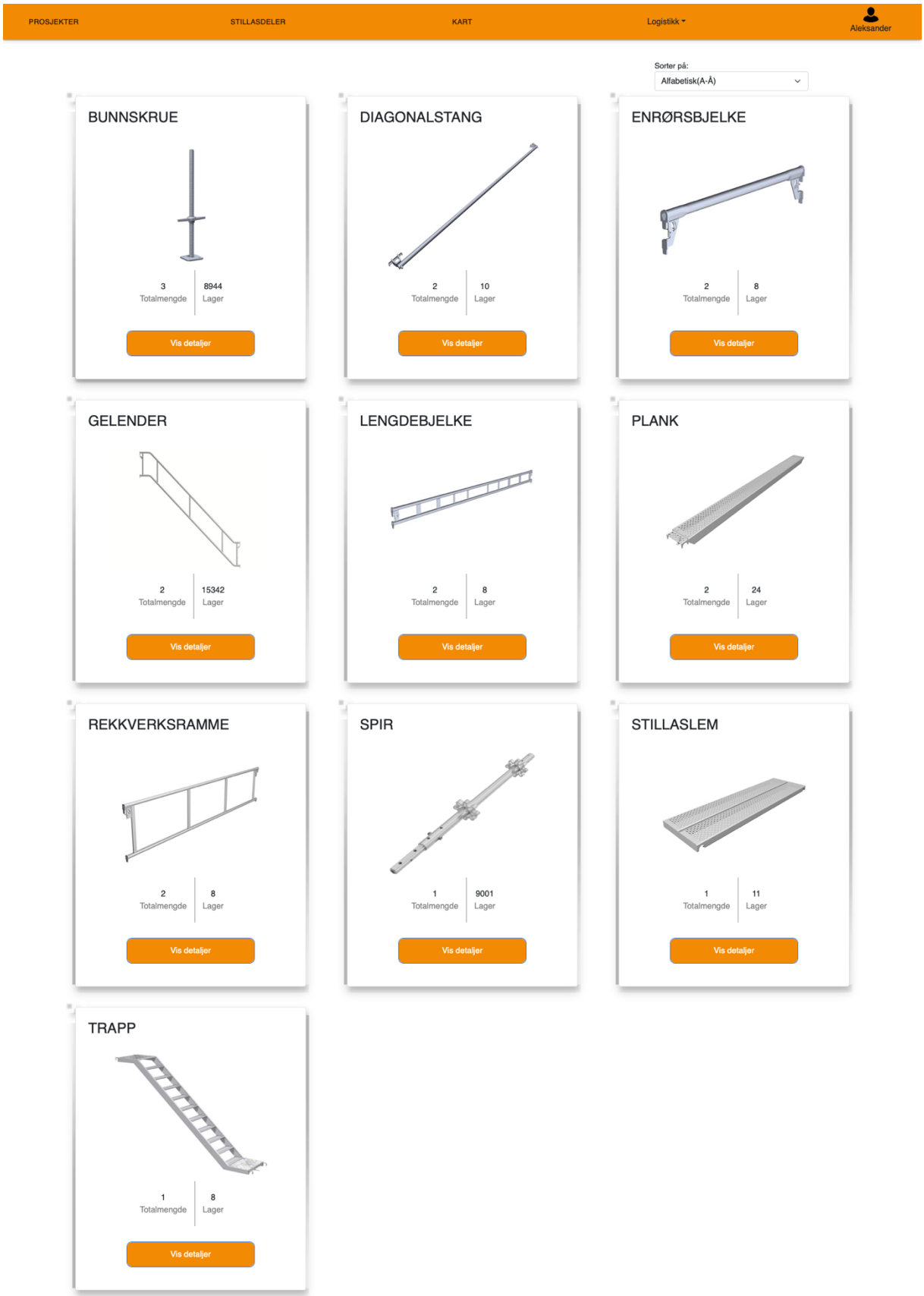

TrackingUnit > ScaffoldingPart... > Diagonalstang > 22D7D9
ScaffoldingParts
Start collection
Bunnskrue
Diagonalstang
Enrørsbjelke
Gelender
Add field
Diagonalstang
Add document
22D7D9
22D7D9
Start collection
Add field
batteryLevel: 3.0199999809265137
project: "CCHamar"
tagID: "22D7D9"
type: "Diagonalstang"

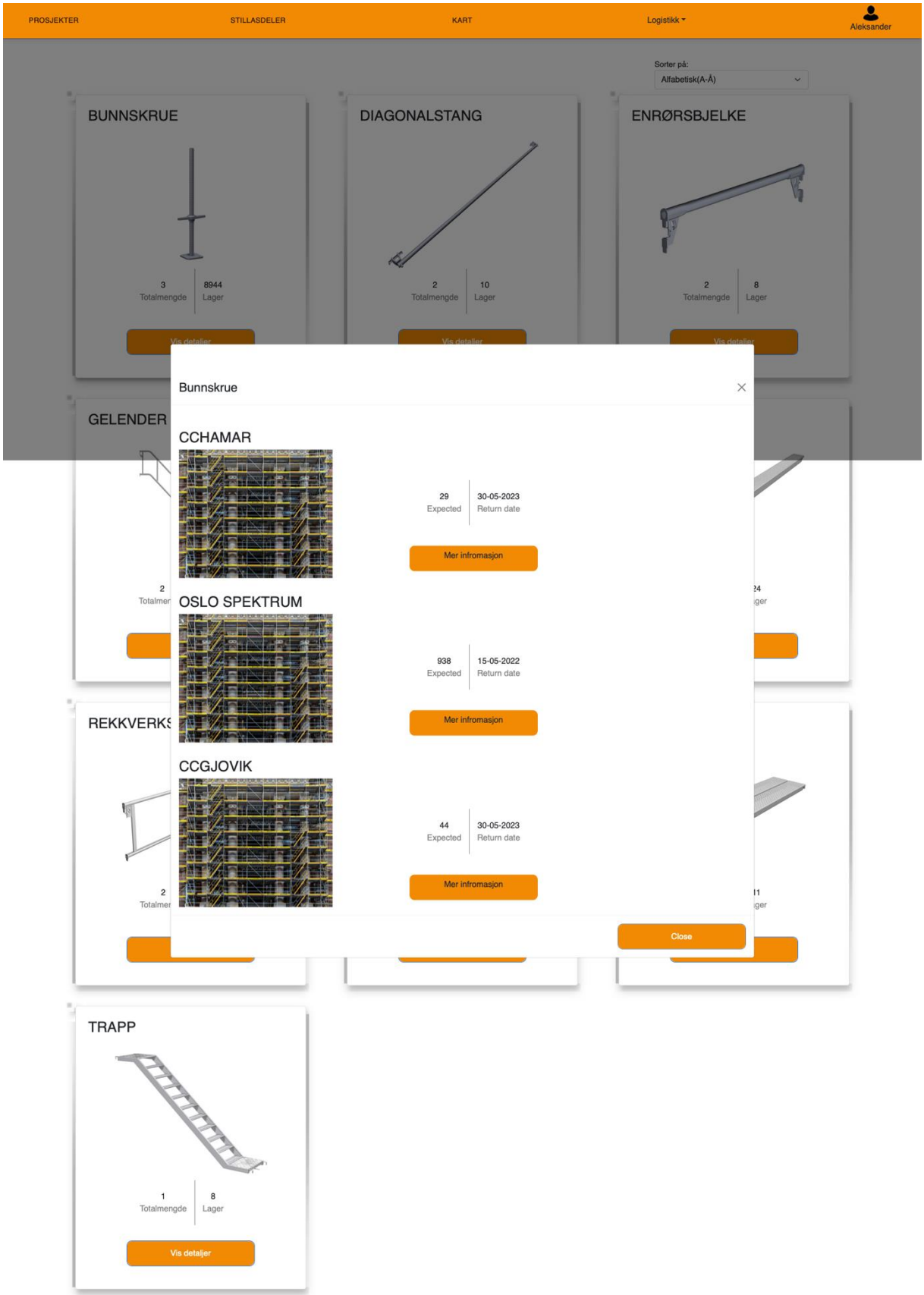

Upcoming > 12 > StillasType > Bunnskrue
12
Start collection
StillasType
Add field
address
county: "Oslo"
municipality: "Oslo"
street: "Grønlandsgate 10"
zipcode: "0002"
customer
email: "oslo.kommune@mail.com"
name: "Oslo Kommune"
number: 47312643
StillasType
Add document
Bunnskrue
Diagonalstang
Enrørsbjelke
Gelender
Lengdebjelke
Plank
Rekkverksramme
Spir
Stillaslem
Trapp
Bunnskrue
Start collection
Add field
Quantity
expected: 939
registered: 0
type: "Bunnskrue"

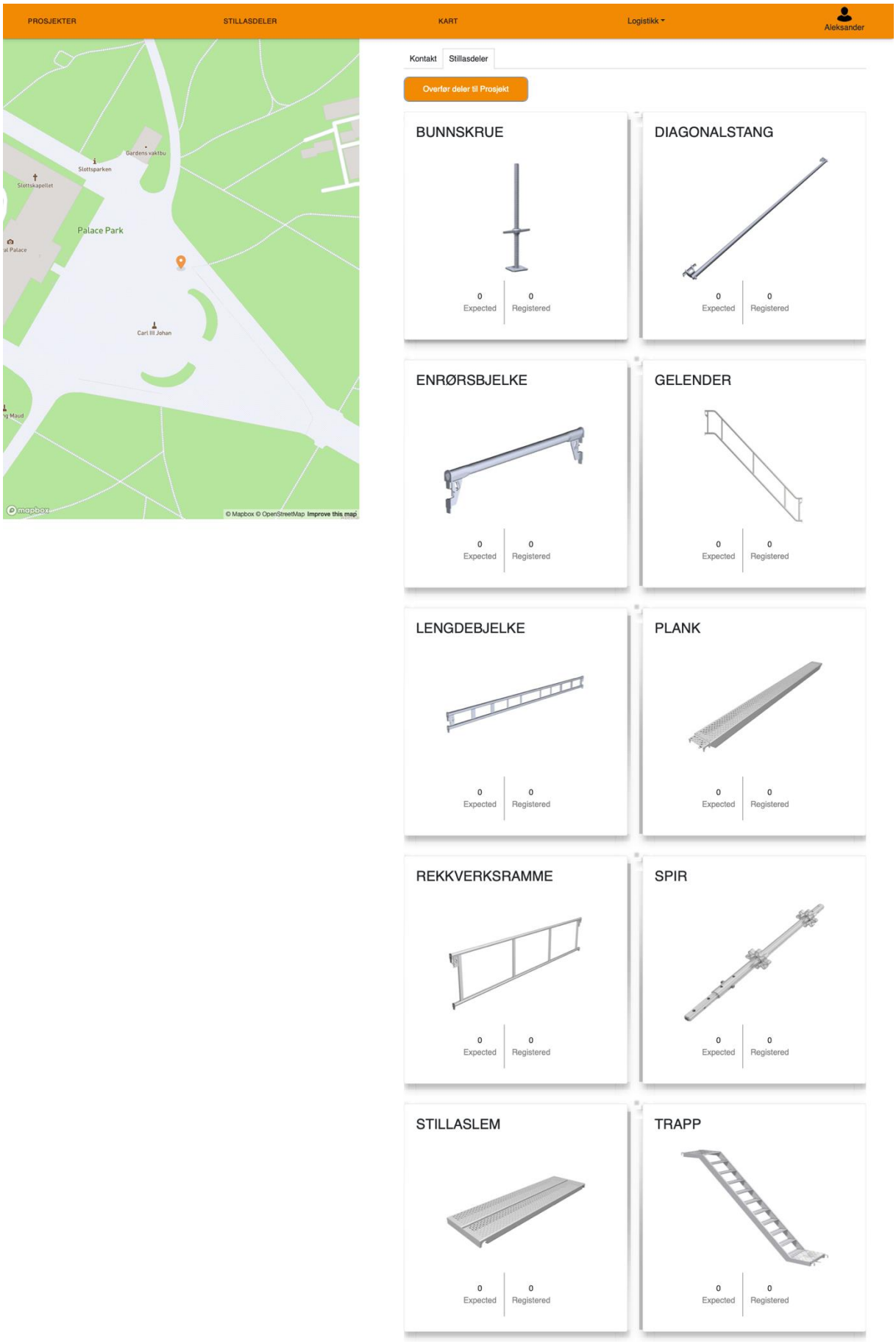

Location > Storage > Inventory > Bunnskrue
Storage
Start collection
Inventory
Add field
This document has no data
Inventory
Add document
Bunnskrue
Diagonalstang
Enrørsbjelke
Gelender
Lengdebjelke
Plank
Rekkverksramme
Spir
Stillaslem
Trapp
Bunnskrue
Start collection
Add field
Quantity
expected: 8944
registered: 0
type: "Bunnskrue"

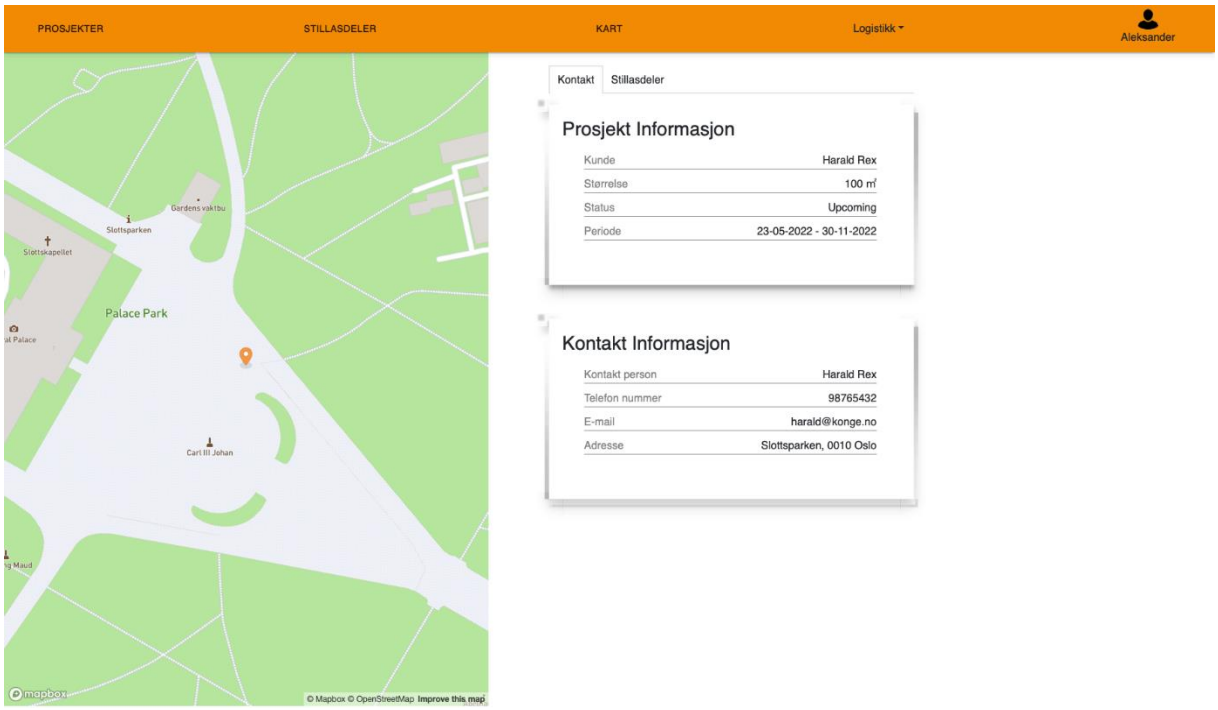

Gateways
34AB954B54E4
stillas-16563
Start collection
Gateways
History
Location
TrackingUnit
Users
Gateways
Add document
34AB954B54E4
34AB954B568C
34AB954B54E4
Start collection
Add field
Status: true
gatewayID: "34AB954B54E4"
latitude: 59.911491
longitude: 10.757933
projectID: 4
projectName: "CCHamar"

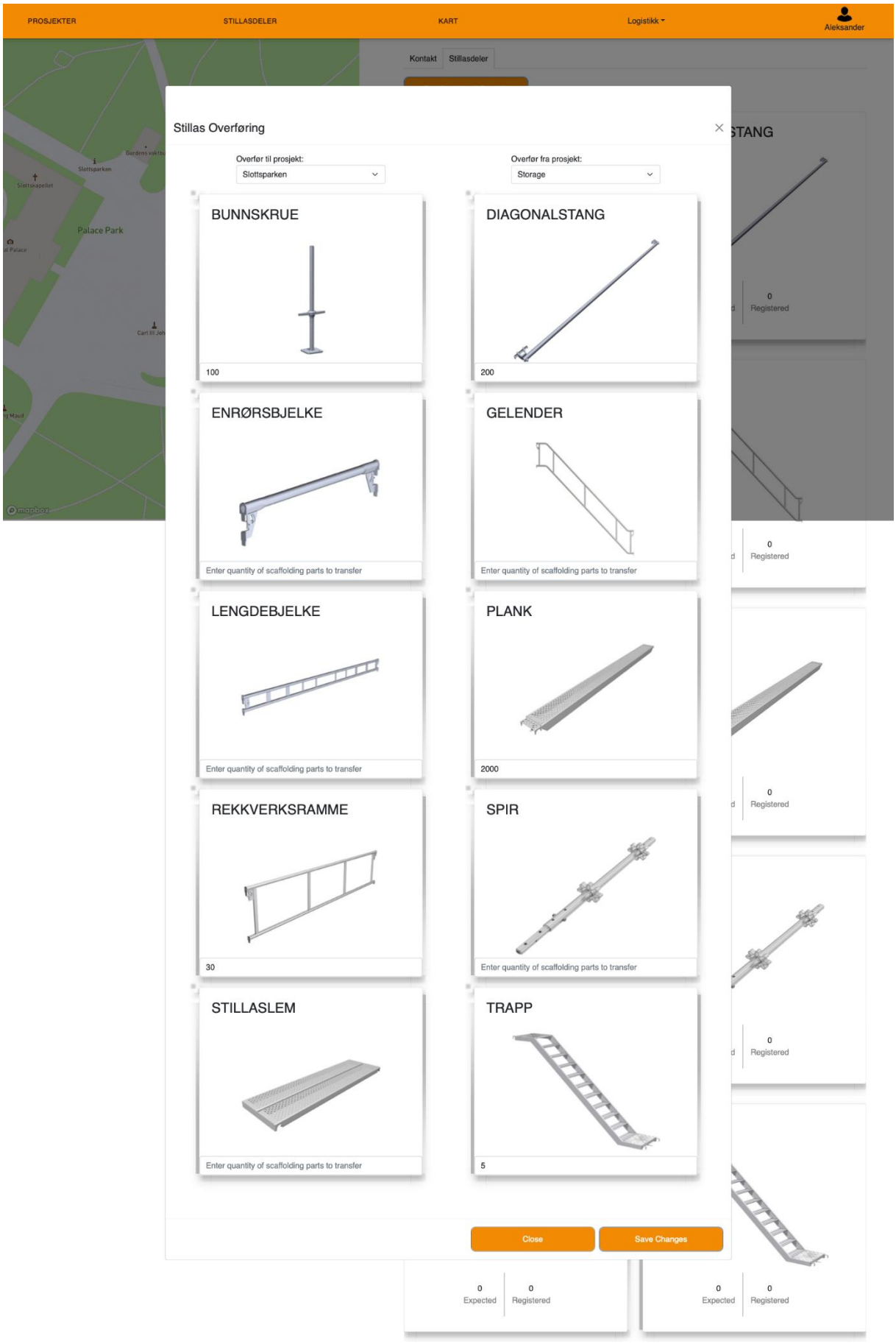
Appendix M Web-Løsning




PROSJEKTER
STILLASDELER
KART
Logistikk
Aleksander
Sorter på:
Alfabetisk(A-Å)
BUNNSKRUE
3
Totalmengde
8944
Lager
DIAGONALSTANG
2
Totalmengde
10
Lager
ENRØRSBJELKE
2
Totalmengde
8
Lager
GELENDER
REKKVERKS
TRAPP
1
Totalmengde
8
Lager
Vis detaljer
Bunnskrue
CCHAMAR
29
Expected
30-05-2023
Return date
Mer infromasjon
OSLO SPEKTRUM
938
Expected
15-05-2022
Return date
Mer infromasjon
CCGJOVIK
44
Expected
30-05-2023
Return date
Mer infromasjon
Close


PROSJEKTER
STILLASDELER
KART
Logistikk
Aleksander
Kontakt
Stillasdeler
Overfør deler til Prosjekt
Palace Park
Gardens vaktbu
Slottsparken
Slottskapellet
Carl III Johan
© Mapbox © OpenStreetMap Improve this map
BUNNSKRUE
DIAGONALSTANG
ENRØRSBJELKE
GELENDER
LENGDEBJELKE
PLANK
REKKVERKSRAMME
SPIR
STILLASLEM
TRAPP
0 Expected
0 Registered


PROSJEKTER
STILLASDELER
KART
Logistikk
Aleksander
Kontakt
Stillasdeler
Prosjekt Informasjon
Kunde
Harald Rex
Størrelse
100 m²
Status
Upcoming
Periode
23-05-2022 - 30-11-2022
Kontakt Informasjon
Kontakt person
Harald Rex
Telefon nummer
98765432
E-mail
harald@konge.no
Adresse
Slottsparken, 0010 Oslo
Palace Park
Slottsparken
Slottskapellet
Gardens vaktbu
Carl III Johan
© Mapbox © OpenStreetMap Improve this map


PROSJEKTER
STILLASDELER
KART
Logistikk
Aleksander
Kontakt
Stillasdeler
Stillas Overføring
Overfør til prosjekt:
Slottsparken
Overfør fra prosjekt:
Storage
BUNNSKRUE
100
DIAGONALSTANG
200
ENRØRSBJELKE
Enter quantity of scaffolding parts to transfer
GELENDER
Enter quantity of scaffolding parts to transfer
LENGDEBJELKE
Enter quantity of scaffolding parts to transfer
PLANK
2000
REKKVERKSRAMME
30
SPIR
Enter quantity of scaffolding parts to transfer
STILLASLEM
Enter quantity of scaffolding parts to transfer
TRAPP
5
Close
Save Changes
Palace Park
Registered
Expected
Registered

Aleksander Aaboen, Martin Iversen og
Tormod Mork Müller

Sporing av stillas

NTNU
Kunnskap for en bedre verden

MB STILLAS