

Applikasjon for innmelding av arkeologiske funn

Hovedprosjekt ved OsloMet - Storbyuniversitetet
Fakultetet for teknologi, kunst og design
Våren 2021

Gruppe 54

Adrian Szabo Aabech
Benjamin Nils Øvergaard
Helge Helland

Laréb Fatima Ahmad
Tor Ryan Andersen



**Institutt for Informasjonsteknologi**

Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo
Besøksadresse: Holbergs plass, Oslo

PROSJEKT NR.
54

TILGJENGELIGHET
Åpen

Telefon: 22 45 32 00

BACHELORPROSJEKT

HOVEDPROSJEKTETS TITTEL	DATO
Applikasjon for innmelding av arkeologiske funn	26.05.2021
PROSJEKTDELTAKERE	ANTALL SIDER / BILAG
Adrian Szabo Aabech Laréb Fatima Ahmad Tor Ryan Andersen Helge Helland Benjamin Nils Øvergaard	88
OPPDRAKGIVER	INTERN VEILEDER
Riksantikvaren	Lothar Fritsch
KONTAKTPERSON	Maria Sølversen

SAMMENDRAG
Riksantikvaren ønsker en digital løsning for å forenkle og redusere feil i utfylling av funnskjema og funnmeldinger. Bachelorprosjektet vil hovedsakelig være en "Proof of concept"-applikasjon for å vise nytten av et større mer omfattende prosjekt senere.
For å løse problemet utviklet gruppen en applikasjon for å forenkle utfylling og innsending av både funnskjema og funnmelding. Applikasjonen er en android mobil applikasjon. Applikasjonen er bygd i java med en backend lagd med ASP .NET.
Applikasjonen hjelper brukeren i informasjonsinnhenting, utfylling og sending av funnmeldig og funnskjema. Den bidrar til å redusere feilkilder og forenkler prosessen.

3 STIKKORD
Android
Azure
Arkeologi

Forord

I dette dokumentet presenteres applikasjonen gruppe 54 har utviklet som bachelorprosjekt ved Fakultet for teknologi, kunst og design ved OsloMet - Storbyuniversitetet. Dokumentet er forfattet våren 2021 og omhandler utviklingsprosessen av applikasjonen.

Oppgaven er skrevet mellom to informasjonteknologistudenter og tre dataingeniørstudenter. Studentene heter Adrian, Benjamin, Fatima, Helge og Tor.

Bidrag

Tor og Helge utviklet frontend-siden. Tor var også ansvarlig for kommunikasjonen mellom oss og Riksantikvaren.

Benjamin, Fatima og Adrian utviklet design og innhold for produktet. Fatima og Adrian hadde hovedansvaret for backend-siden. Benjamin bidro der det var behov.

Leserveileddning:

Vi antok at leseren har visse kunnskaper om programmering og app-utvikling da vi skrev oppgaveteksten. Vi har likevel valgt å forklart store deler av begrepene vi bruker. Dette gjør at det til en viss grad er mulig for en lekemann å kunne forstå teksten. I oppgaveteksten er begrepene tilføyd opphøyde tall. Tallene referer til en ordliste for tekniske begreper. Ordlisten ligger blant oppgavens vedlegg.

Anerkjennelser

- Takk til vår eksterne veileder Maria Sølversen
- Takk til vår interne veileder Lothar Fritsch
- Takk til OsloMet for tre fine år

Sammendrag

Riksantikvaren ønsker en digital løsning for å forenkle og redusere feil i utfylling av funnskjema og funnmeldinger. Bachelorprosjektet vil hovedsakelig være en “Proof of concept”¹ applikasjon for å vise nytten av et større mer omfattende prosjekt senere.

For å løse problemet utviklet gruppen en applikasjon for å forenkle utfylling og innsending av både funnskjema og funnmelding. Applikasjonen er en android mobil applikasjon. Applikasjonen er bygd i java med en backend lagd med ASP .NET.

Applikasjonen hjelper brukeren i informasjonsinnhenting, utfylling og sending av funnmeldig og funnskjema. Den bidrar til å redusere feilkilder og forenkler prosessen.

Forord	3
Bidrag	3
Leserveiledning:	3
Anerkjennelser	3
Sammendrag	4
1. Innledning	9
1.1 Bachelorgruppen	9
1.2 OsloMet – storbyuniversitet	9
1.3 Oppdragsgiver- Riksantikvaren	10
1.4 Oppdrag	10
1.4.1 Bakgrunn for oppdraget	10
1.4.2 Oppdraget	11
1.4.3 Mål og forventet gevinst	11
1.4.4 Begrensende faktorer	11
2. Produktdokumentasjon (Resultatet)	12
2.1 Frontend-dokumentasjon	12
2.1.1 Frontend Struktur	12
2.1.2 Gjennomgang	14
2.1.2.1 Velkommen til applikasjonen	14
2.1.2.2 Logg inn	15
2.1.2.3 Registrer Bruker	16
2.1.2.4 Nytt funn - Registrering funn	17
2.1.2.5 Mine funn	19
2.1.2.6 Enkeltfunn	20
2.1.2.7 Innstillinger	22
2.1.2.8 Info-siden	23
2.1.2.9 Profil	24
2.1.3 Generell funksjonalitet	26
2.1.3.1 Inputvalidering	26
2.1.3.2 Tilrettelegging for oversettelse	26
2.1.4 Design og brukeropplevelse	27
2.2 Backend-dokumentasjon	28
2.2.1 API - bindeleddet mellom database og frontend	28
2.2.2 Backend	29
2.2.3 Sikring av data	29
2.2.4 Databasen	30
2.2.5 Tabellene i databasen	31
2.2.6 Bruker-metodene	33
2.2.6.1 CreateUser	33
2.2.6.2 Postadresse	33
2.2.6.3 EditUser	33

2.2.6.4 DeleteUser	34
2.2.6.5 GetUser	34
2.2.6.6 LogIn	34
2.2.6.7 LogOut	34
2.2.7 Funn-, grunneier- og gbnr-metodene	34
2.2.7.1 RegistrerFunn	35
2.2.7.2 EditFunn	35
2.2.7.3 DeleteFunn	35
2.2.7.4 GetAllUserFunn	35
2.2.7.5 GetFunn	36
2.2.8 Injeksjonsvern	36
3. Planlegging og kravspesifikasjon	38
3.1 Informasjonsinnhenting og forprosjekt	38
3.1.1 Møte: Gary Brun	38
3.1.2 Møte: Riksantikvaren, arkeologer og detektorister	38
3.1.3 Møter: Maria Sølversen	38
3.1.4 Markedsanalyse, litteratur og interesseorganisasjoner	39
3.2 Kravspesifikasjon	39
3.2.1 Brukerhistorier	40
3.2.2 Funksjonelle krav: Minimumskrav	41
3.2.3 Krav utover minimumskravene	42
3.2.4 Logiske datamodeller	43
3.3 Strukturering av tiden	44
4. Prosessdokumentasjon og metodikk	45
4.1 Covid-19	45
4.2 Teknologier og verktøy	45
4.3 Utviklingsmetode	47
4.3.1 Sprinter	47
4.4 Frontend-utvikling	47
4.4.1 Kort om utviklingsprosess	47
4.4.2 Versjonskontroll	48
4.4.3 Utfordringer	48
4.4.3.1 Utfordring med funnskjema-PDF	48
4.4.3.2 Treg nedlasting av data	48
4.4.4 Viktige beslutninger	49
4.4.4.1 Design	49
4.4.4.2 Info-siden	49
4.5 Backend-utvikling	50
4.5.1 Database initialisering	50
4.5.2 Utfordringer	51
4.5.2.1 Database på serveren	51
4.5.2.2 Testing	52
4.5.2.3 Innsending av et funnbilde fra frontend til backend	53

4.5.3 Viktige beslutninger	54
4.5.3.1 Utelatte metoder	54
4.5.3.1.1 Innlogget-sjekk	54
4.5.3.1.2 Glemt passord	54
4.6 Hovedrapport	55
4.7 Ressursbruk	56
4.7.1 Refleksjon rundt roller	57
4.8 Testdokumentasjon	57
4.8.1 Enhetstesting	57
4.8.2 Integrasjonstesting	58
4.8.3 Systemtesting	58
4.8.4 Testskript	58
4.8.5 Gjennomførelse og intern brukertesting	59
4.8.6 Resultat	59
4.8.7 Forbedringer	60
4.10 Kort teknisk evaluering	60
5. Konklusjon	62
6. Vitenskapelig litteraturliste	63
7. Vedlegg	64
Vedlegg 1 Ordliste	64
Vedlegg 2 Kravspesifikasjon	66
Presentasjon	66
Om bakgrunnen	66
Forord	66
Leserveiledning	68
Kort systembeskrivelse	68
Brukerhistorier	68
Funksjonelle krav: Minimumskrav	70
Krav utover minimumskravene	71
Utfordringer	72
Forventede gevinster	72
Rammekrav i systemet	73
Sikring mot tap, ødeleggelse, tyveri og misbruk av data	73
Kapasitet	73
Fremtidig utvidelse av systemet	74
Brukervennlighet og universelt design	74
Logiske datamodeller	75
Overordnet prosessdiagram	75
Flyt i programmet	78
Wireframe:	79
Interaksjoner/funksjonskall	80
Markedsanalyse	81
Tabell som utforsker organisasjoner, apper og nettplattformer:	81

Krav til dokumentasjon	83
Arbeidsplan	84
Vedlegg 3 Rolletabellen	85
Vedlegg 4 PDF Biblioteker	85
Vedlegg 5 Sprint-tabell	86
Vedlegg 6 Funnskjema	88

1. Innledning

Første del av dette kapittelet omhandler viktige aktører. Siste del av kapittelet handler om oppdraget /oppgaven.

1.1 Bachelorgruppen

Bacheloroppgaven er forfattet av fem studenter:

- Adrian Szabo Aabech - aceasa08@hotmail.com
- Laréb Fatima Ahmad - Ifahmad98@gmail.com
- Tor Ryan Andersen - tor.ryan.andersen@gmail.com
- Helge Helland - helghelland@gmail.com
- Benjamin Nils Øvergaard - ben.overgaard@gmail.com

Studentene studerer ved OsloMet - storbyuniversitet i Oslo. Alle går siste semester på en bachelorgrad ved Fakultetet for teknologi, kunst og design (TKD).

Ved TKD studerer Adrian, Helge og Tor Dataingeniør. Benjamin og Fatima studerer Informasjonsteknologi.

Gruppemedlemmene er godt kjent med hverandre fra tidligere arbeidsoppgaver og prosjekter gjennom studietiden.

1.2 OsloMet – storbyuniversitet



Oppgaven er skrevet med veiledning fra Lothar Fritsch.
Lothar Fritsch er professor med tilhørighet til TKD ved
OsloMet.

Oslomet om seg selv:

OsloMet skal være et urbant og mangfoldig universitet med tydelig internasjonalt preg, samtidig som vi er tett på samfunns- og arbeidslivets behov. Arbeidet vårt skal gjøres i en profesjonell organisasjon som er ledende å ta i bruk ny teknologi og nye løsninger.



1.3 Oppdragsgiver- Riksantikvaren

Det ble sett på flere alternative oppdrag. Et av de mer spennende var et oppdrag for Riksantikvaren. Gruppen fikk kjennskap om oppdraget gjennom gruppemedlemmet Adrian. Adrian er ansatt hos Riksantikvaren. Gruppen aksepterte oppdraget fra Riksantikvaren tidlig i Januar. Maria Sølversen var gruppens kontaktperson hos

Riksantikvaren og produkteier. Hun fungerte som gruppens eksterne veileder under arbeidet med bacheloroppgaven.

Riksantikvarens beskrivelse av seg selv, hentet fra deres egne nettsider:

"Riksantikvaren er et direktorat underlagt Klima- og miljødepartementet, og vi er departementets rådgiver i alle saker som gjelder kulturminner og kulturmiljø. Riksantikvaren er den overordnede kulturminnemyndigheten og ansvarlig for å sette i verk den nasjonale kulturminnepolitikken. Vi har et faglig ansvar overfor kommunene, fylkeskommunene, Sametinget, Sysselmannen på Svalbard og forvaltningsmuseene på kulturminnefeltet."

1.4 Oppdrag

1.4.1 Bakgrunn for oppdraget

Bachelorgruppens oppdragsgiver, Riksantikvaren, er i ferd med å digitalisere prosessene sine. En av prosessene de vurderer å digitalisere er innsending av funnmelding og funnskjema ved oppdagelse av et arkeologisk funn.

Fylkesarkeologene og arkeologer knyttet til Riksantikvaren behandler meldingene og skjemaene. De har problemer med ustandardisert innsending av funnmeldinger og feil utfylte funnskjemaer.

Innsenderne av skjemaene er ofte metalldetektorister (heretter detektorister). Disse ble derfor tidlig pekt ut som en viktig interessent. Deres motivasjon for å melde funn og sende inn funnskjema, er å få finnerlønn. For å få finnerlønn må funnskjemaet fylles ut korrekt, noe som har vist seg å være en utfordring. Oppdraget er en del av større prosess hvor Riksantikvaren kartlegger behovet for utvikling og forbedring av tjenester rettet mot detektoristene.

1.4.2 Oppdraget

Oppdraget fra Riksantikvaren gikk ut på å bygge og presentere en prototype-android-applikasjon. Prototypens formål er å vise verdien av å digitalisere innsending av funnmelding og funnskjema. Verdien er hentet fra prototypens evne til å effektivisere innsending av skjema og å eliminere gjentagende feilkilder under utfylling av skjemaene.

Riksantikvaren vil bruke applikasjonen som et *Proof of concept*¹ i videre arbeid for å vise nytten av å digitalisere funnregistreringen og å utvikle en endelig løsning.

1.4.3 Mål og forventet gevinst

Effektmål knyttet til produktet, tidlig påtenkt fra Riksantikvarens side:

- Innhenting av funnskjema-informasjonen på en god og effektiv måte (blant annet GPS koordinater, bildet, beskrivelser av funn og funnområdet)
- Innsending av funnmelding og funnskjema på en ryddig og standardisert måte

Forventet gevinst for Riksantikvaren og fylkesarkeologen:

- Reduksjon av feil i skjemautfyllingen
- Flere innmeldte funn (som følge av lavere terskel for innmelding)
- Reduksjon i ulovlig søking
- Bedre oversikt over nye funn

Forventet gevinst for detektoristen (sluttbruker):

- Reduksjon av tidsbruk ved funnregistrering
- Forenklet utfylling av funnmelding og funnskjema
- Forenklet innmelding av funn og innsending av funnskjema
- Mindre usikkerhet rundt lover og regler knyttet til metallsøking
- Raskere utbetaling av finnerlønn

1.4.4 Begrensende faktorer

Bachelorgruppen og Riksantikvaren ble enige om at oppdraget skulle utføres i henhold til følgende begrensninger:

- Tidsfristen for innleveringen av bachelorrapporten
- Gruppens kompetanse
- Gruppens mangel på mulighet til å følge opp og oppdatere applikasjonen etter endt prosjektperiode
- Riksantikvaren har ikke et drifts- og vedlikeholdsbudsjett for dette prosjektet
- Riksantikvaren har ingen teknisk veileder

2. Produktdokumentasjon (Resultatet)

Dette kapittelet har to deler.

1. *Frontend-Dokumentasjon: I første halvdel av kapittelet beskriver vi applikasjonen. Det blir gjort en gjennomgang av alle sidene i applikasjonen og det blir sett på funksjonalitet side for side.*
2. *Backend-Dokumentasjon: I andre halvdel av kapittelet beskriver vi serveren, databasen og kommunikasjonen mellom backend og frontend. Det blir gjort en gjennomgang av backend-metodene.*

2.1 Frontend-dokumentasjon

Frontend er delen av programmet brukeren er i kontakt med. Frontend omhandler den koden som former det brukeren ser på skjermen og bestemmer hva som skjer når brukeren anvender elementer i applikasjonen.

2.1.1 Frontend Struktur

Applikasjonen er laget i Android Studios. Den består hovedsakelig av Java- og XML-filer.

Java-filene deles inn i fragmentklasser, dataklasser og logiske klasser:

- Fragmentklasser er klasser som utvider android klassen “Fragment”. Et fragment er en side i en android applikasjon. Fragmentklassene beskriver hvordan sidene oppfører seg og ser ut. De inneholder logikken som utføres på siden.
- Dataklassene er klasser som holder data. Klassene inneholder viktige datafelter. Klassene er fylt med Get¹⁰ og Set¹² metoder. Et eksempel på en dataklasse er funn-klassen. Denne klassen inneholder felter med data for et funn.
- Logiske klasser er klasser som inneholder logikk. Logikken kan brukes flere steder i applikasjonen og er ikke bundet til for eksempel et fragment.
Et eksempel på en logisk klasse er “GetJSON”-klassen. GetJSON inneholder kode for å sende en HTTP GET-request¹⁴ til serveren. Requesten sendes for å motta og håndtere data.

I tillegg til de tre hovedtypene er det en klasse som heter “MainActivity”. Det er MainActivity som kobler sammen XML-knappene med java-metodene. MainActivity initialiserer og håndterer fragmentene som skal vises på skjermen.

XML-filene inneholder XML-kode. Det er XML-filene som definerer hvordan sidene ser ut i applikasjonen. XML står for “Extensible Markup Language”. XML bruker emneknagger og attributter for å definere hvordan en side skal se ut.

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/tittel"  
    android:textColor="#000000"  
    android:textSize="18sp" />
```

Figur 2.1: Slik er koden for å vise tekst på skjermen med XML.

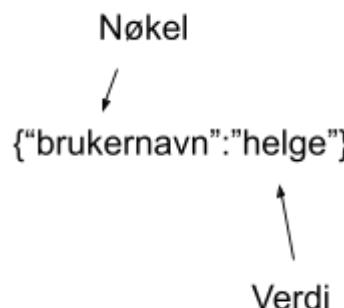
Android applikasjoner inneholder også en spesiell XML-fil kalt “AndroidManifest”. I denne filen defineres applikasjonens rettigheter, navn, ikoner med mer.

Applikasjonen bruker i tillegg gradle-filer. I gradle-filene er det definert versjoner og biblioteker som brukes i applikasjonen. Et eksternt bibliotek er lagt til i våre gradle-filer. Det eksterne biblioteket heter Volley. Ser man bort fra dette eksterne biblioteket, er gradle-filene våre autogenererte.

Volley er et bibliotek for å sende og motta JSON⁹. Applikasjonen bruker Volley hovedsakelig for å sende funn-objekter som JSON til serveren. Volley lager et JSON-objekt fra et Java map¹⁰.

- Java map-er inneholder et nøkkel- og verdi-par for hvert felt i funn-objektet
- Nøkkelen er navnet på JSON-feltet og verdien er JSON-feltets verdi (Se [figur 2.1](#))

JSON ble brukt for å sende funn-objektet. JSON måtte brukes fordi vi skulle sende bilder. Når bildene skal sendes konverteres de til en Base64-streng⁵. Lengden på Base64-strenge vil ofte overstige den maksimale lengden en URL¹¹ kan ha. Derfor kunne vi ikke sende strengen via URL¹¹.



Figur 2.1: Figuren viser et JSON objekt⁴

2.1.2 Gjennomgang

I denne delen dokumenteres frontend-funksjonaliteten. Vi har valgt å gå gjennom applikasjonen fragment for fragment. For hvert fragment beskrives funksjonalitet knyttet til fragmentet.

2.1.2.1 Velkommen til applikasjonen

Ved første oppstart av applikasjonen vil velkomstsiden komme opp.

Etter første oppstart vil ikke velkomstsiden dukke opp igjen. Dette oppnås ved å ta vare på en boolean verdi i "Shared Preferences"²⁰. Shared Preferences brukes til å ta vare på verdier man vil lagre lokalt i applikasjonen.

Boolean-verdien endres til "false" når knappen "Gå videre" blir trykket på. Når verdien er "false" vil applikasjonen åpne "Logg inn"-siden ved oppstart, i stedet for åpne velkomstsiden.



Figur 2.2: Bilde av velkommen siden

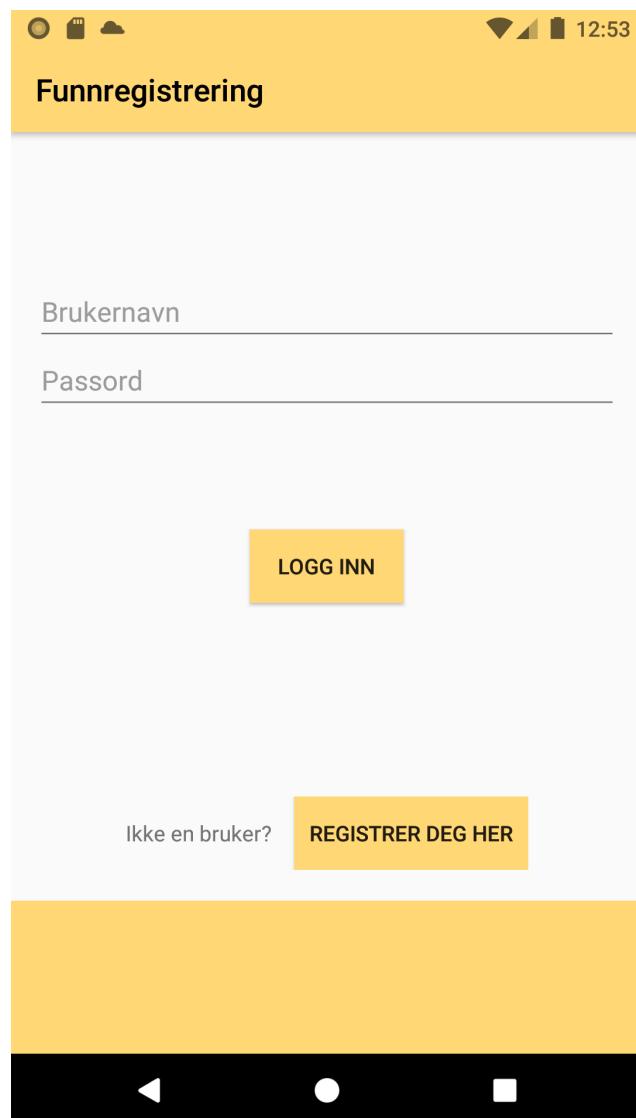
2.1.2.2 Logg inn

“Logg inn”-siden er den første siden man kommer til når applikasjonen åpnes. Her kan brukeren enten logge inn eller lage en ny bruker.

Ved innlogging må brukernavn og passord tastes inn. Deretter kan brukeren trykker på logg inn-knappen. Når knappen trykkes, sendes en HTTP POST-request¹⁶ til serveren. Serveren sjekker om kombinasjonen av brukernavn og passord korresponderer med et brukernavn og passord i databasen. Om brukeren finnes og passordet er riktig blir bruker logget inn.

Om brukernavn eller passord er feil får brukeren en feilmelding. Da kan brukeren prøve igjen. Feilmeldingen er en enkel beskjed som kommer opp på skjermen. Toast²¹ blir brukt for å vise feilmeldingen på skjermen. En toast er en Android klasse som brukes til å vise en melding på skjermen.

Ved suksessfull innlogging blir brukeren sendt til “registrer nytt funn”-siden. Brukernavn og passord lagres i applikasjonen. Dette gjør at brukeren blir logget inn automatisk neste gang applikasjonen åpnes.



Figur 2.3: Viser hvordan logginn siden ser ut og fungerer.

2.1.2.3 Registrer Bruker

Når bruker trykker på "Registrer deg her"-knappen åpnes "registrer bruker"-siden. Her må brukeren legge inn nødvendig informasjon for å opprette en profil. Profilen benyttes til innlogging.

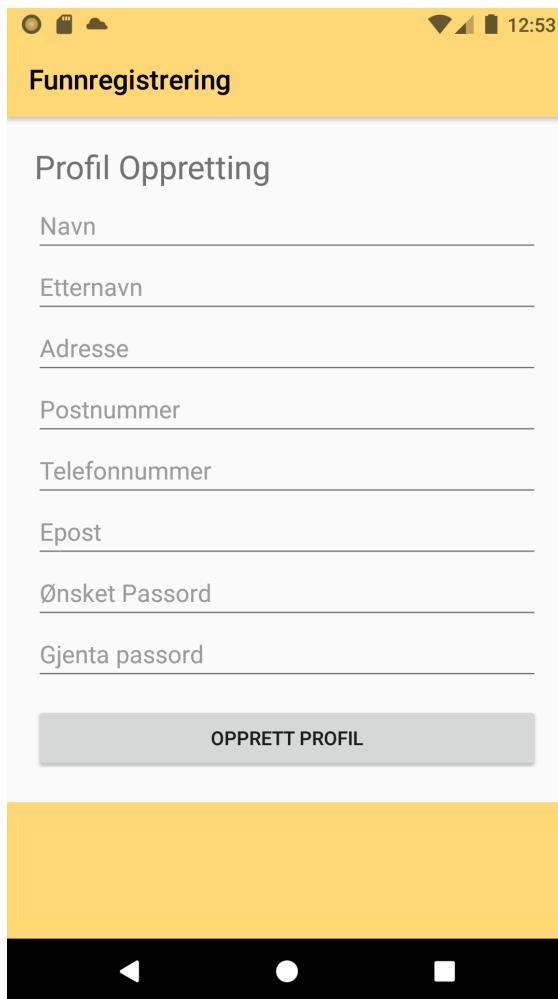
Nye brukere registreres med: navn, adresse, postnummer, poststed, telefonnummer, e-postadresse og passord.

Når "Opprett bruker"-knappen trykkes sendes det en HTTP POST-request¹⁶ til serveren.

Requesten inneholder informasjonen brukeren oppga ved registrering.

Ved feil under registrering vil brukeren få en feilmelding, og muligheten til å prøve på nytt.

Når en ny bruker er registrert, vil brukerens data bli lagret i databasen. Deretter vil brukeren bli sendt tilbake til innloggingssiden.



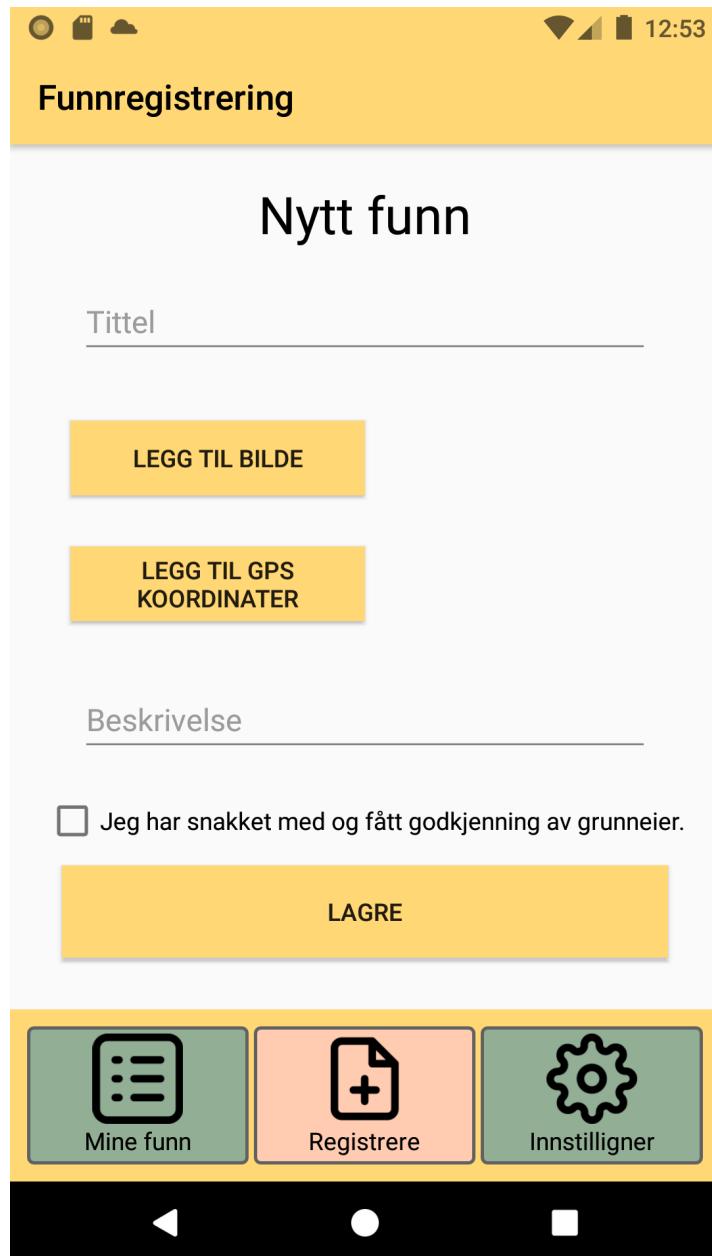
Figur 2.4: Viser hvordan bruker-opprettings-siden funker.

2.1.2.4 Nytt funn - Registrering funn

“Registrer funn” er siden hvor brukeren kan registrere et nytt funn. Da blir funnet lagt til i listen over egne funn. Når det registreres et funn kan man legge til: tittel, bilde, GPS-koordinater og beskrivelse. Tittel *må* legges til for at man skal kunne opprette et funn. I tillegg må man bekrefte at grunneier har gitt tillatelse til søking. Bekreftelse av godkjenning gjøres ved å sjekke av en boks. Bilde, GPS-koordinater og beskrivelse er frivillige felter. De må likevel fylles inn senere om de ikke utfyller her. Fra GPS-koordinatene hentes det ut fylke- og kommunenavn.

Når et nytt funn blir lagret sendes et JSON-objekt⁹ med funnets felter til serveren. Serveren lagrer dette i applikasjonens databasen. Ved feil under registrering av nytt funn, vil en feilmelding vises i applikasjonen.

Om funnet er registrert vil applikasjonen motta en godkjent-respons. Ved mottat godkjent-respons oppdateres funnlisten til brukeren på “mine funn”-siden. Funnet vil deretter åpnes slik at brukeren kan legge til ytterligere informasjon. Etter vellykket oppretting av nytt funn vil “registrer funn”-siden tilbakestilles. Tilbakestillingen gjør at siden er tom neste gang brukeren ønsker å opprette et funn.



Figur 2.5: Viser hvordan registrere nyttfunn-siden ser ut og fungerer.

2.1.2.5 Mine funn

“Mine funn”-siden inneholder en liste fylt med funn knyttet til brukeren. Funnene blir hentet fra serveren ved hjelp av en HTTP GET-request¹⁴:

`https://funnapi.azurewebsites.net/Funn/GetAllUserFunn?brukernavn=helge&passord=helge123`

Nedbryting av URL-en¹¹:

- Domenenavnet: “funnapi.azurewebsites.net”
- Server-klasse og -metoden: “/Funn/GetAllUserFunn”
- Metodens parametre og verdien de inneholder:
“?brukernavn=helge&passord=helge123”

I requesten som blir gjort i eksempelet vil metoden hente ut alle funn som tilhører brukeren “helge”.

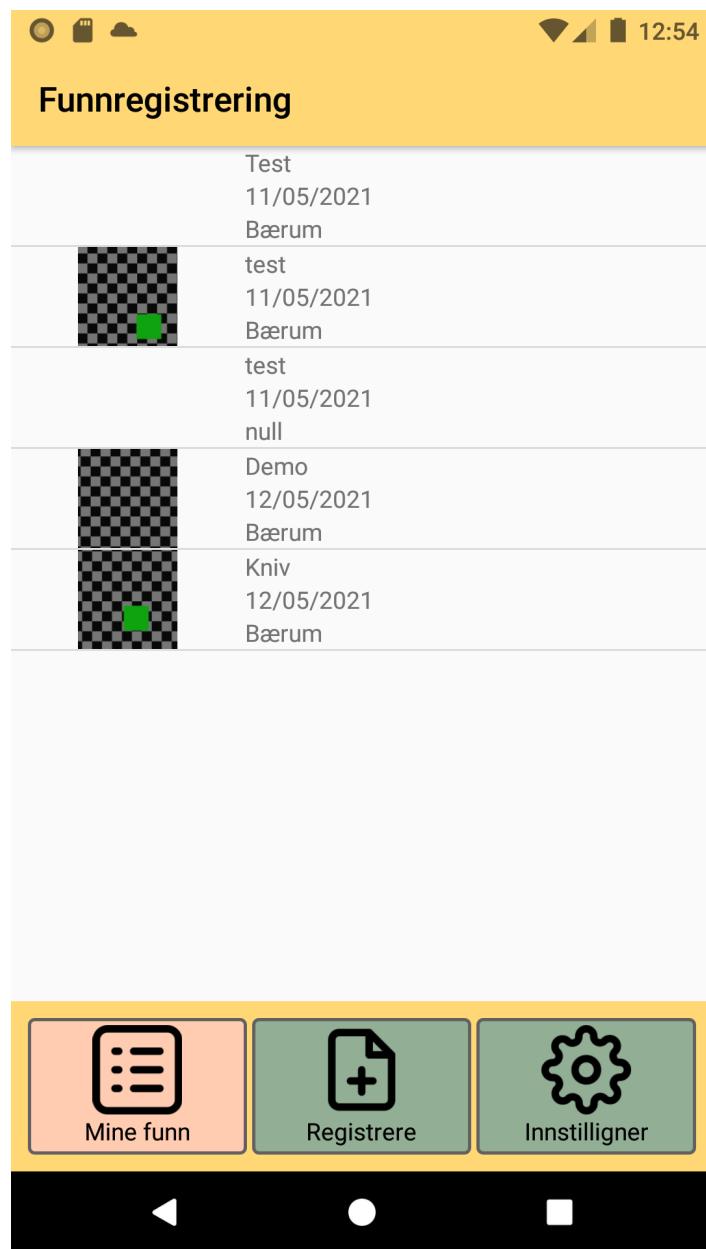
I applikasjonen bygges og sendes URL-en i en separat tråd. Dette gjør at applikasjonen ikke fryser mens informasjonen hentes fra serveren. Når innhenting av listen pågår vises en roterende sirkel på skjermen([se bilde 4.1](#)). Det viser brukeren at applikasjonen jobber.

Om listen ikke kan hentes fra serveren vil applikasjonen vise en feilmelding i applikasjonen. Listen vil vises som tom.

Om listen kan hentes fra serveren vil applikasjonen vise den på skjermen. I listen vises hvert funn med bilde til venstre. Tittel, dato og kommune vises over hverandre, til høyre for bilde.

Trykkes det og holdes det inne på et av funnene i listen åpnes det opp en dialogboks. Boksen spør brukeren om de ønsker å slette funnet fra sin liste.

Ved å trykke på et funn åpnes “enkeltfunn” siden. Den inneholder flere detaljer om funnet.



Figur 2.6: Viser hvordan mine funn siden ser ut og fungerer.

2.1.2.6 Enkelfunn

På enkelfunn-siden kan brukeren se og endre informasjonen knyttet til et funn.

Informasjonen som ble fylt ut under registreringen ligger allerede inne.

Om GPS-koordinater og bilde er fylt ut vil knappen "Send funnmelding" være gul. Dette betyr at tilstrekkelig informasjon for å sende funnmelding er fylt ut.

Ved å trykke på knappen "Send funnmelding" kommer brukeren til et vindu hvor epost-applikasjon kan velges. Ved valgt epost-applikasjon åpnes en epost ferdig utfyldt med funnmeldingsinformasjon. Dette oppnås ved å opprette en ny email-intent²².

Knappen “Send inn funnskjema” vil være rød første gang funnet åpnes. Rød knapp tilslører at det ikke er fylt ut tilstrekkelig informasjon for å sende funnskjema.

Alle feltene med informasjon må fylles ut og redigeres.

Når alle feltene er fylt ut vil “Send inn funnskjema” endre farge til gul. Gul betyr at skjemaet er klar til innsending. Da vil det være mulig å trykke på “send inn funnskjema”-knappen.

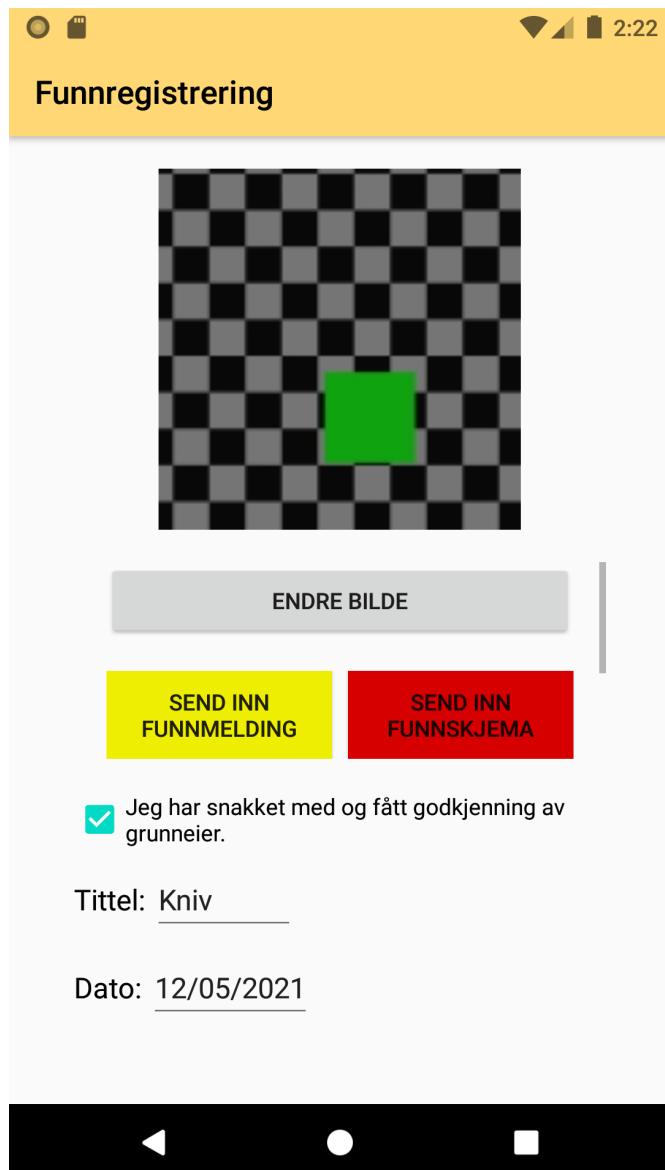
Ved trykk på “send funnskjema” åpnes et vindu hvor brukeren kan velge epost-applikasjon.

Når epost-applikasjon er valgt vil applikasjonen åpne seg med en ferdig utfylt epost. Eposten har funnskjemaet ferdig utfylt som vedlegg.

Dette oppnås ved å opprette en ny email-intent²². En PDF-fil blir lagt til i intenten.

For å konstruere en PDF-fil av funnskjema kjøres “pdfGenerator”-metoden. Metoden tar i bruk flere “androids graphics”-klasser²³ for å tegne inn funninformasjonen i et funnskjema og danne en PDF-fil (Se [vedlegg 6](#)).

For å registrere endringene brukes Volley for innsending av et JSON-objekt⁹ til serveren.



Figur 2.7: Viser hvordan enkelt funn siden ser ut og fungerer.

2.1.2.7 Innstillinger

På siden for innstillinger utføres applikasjon-konfigurasjoner. Innstillinger inneholder to knapper. Den ene knappen fører til "info og regler"-siden. Den andre knappen fører til "profilsiden".



Figur 2.8: Viser hvordan innstillingen siden ser ut og fungerer.

2.1.2.8 Info-siden

Brukeren kommer inn på infosiden via innstillinger-siden. På denne siden kan brukeren lese om relevante lover og regler, samt få tips rettet mot uerfarne detektorister. Her finnes det også en lenke til riksantikvarens nettsider.

Teksten er delt opp i kategorier og ligger bak en knapp. Ved å trykke på knappen vil man se teksten. Trykker man på nytt gjemmes teksten.



Figur 2.9: Bilde viser info og regler siden.

2.1.2.9 Profil

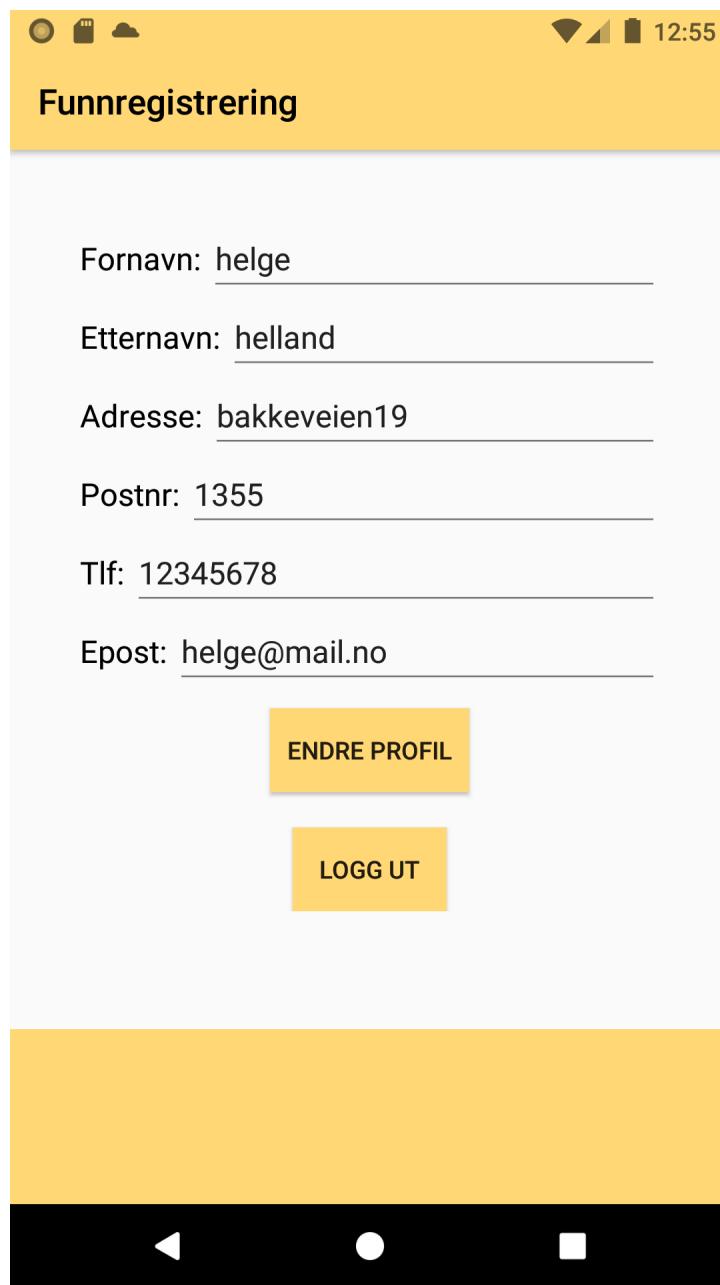
Når bruker trykker på "Profil" knappen på innstillinger siden, åpnes profilsiden. Her vises informasjon om det innloggede brukeren. Brukeren kan endre informasjon i feltene og lagre endringene ved å trykke på "Endre profil".

Ved trykk på "Endre profil"-knappen sendes en HTTP POST-request¹⁶ til serveren for å endre brukerens informasjon i databasen.

Ved feilmelding fra serveren vil brukeren bli informert og bedt om å prøve igjen senere.

Ved suksessfull endring vil brukeren oppdateres i databasen og profilsiden lukkes.

Siden har også en "logg ut"-knapp. Den logger brukeren ut. Når brukeren logges ut åpnes logginn-siden. Automatisk login blir skrudd av til neste suksessfulle innlogging.



Figur 2.10: Bildet viser profilsiden.

2.1.3 Generell funksjonalitet

I denne delen dokumenteres frontend-funksjoner som ikke er knyttet til et spesifikt fragment. Vi har valgt å trekke frem funksjonalitet knyttet til kravspesifikasjonen.

2.1.3.1 Inputvalidering

I applikasjonen er det satt inputvalidering²⁴ på alle feltene. For valideringen er det laget en klasse som utvider android-klassen “TextWatcher”. Når et tekstfelt blir endret vil metoder arvet fra TextWatcher-klassen kjøre. Teksten i feltet sjekkes. En feilmelding vises om teksten ikke stemmer med ønsket input.

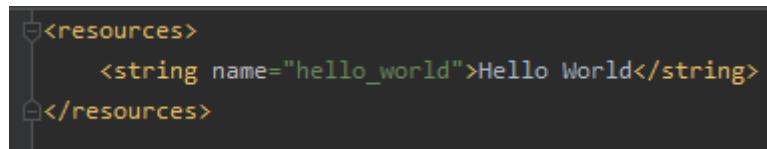
Klassen for validering er dynamisk og brukes til all validering i applikasjonen. For å bestemme hva som skal valideres brukers input validering-klassens konstruktør.

Konstruktøren tar inn en boolean for: normale bokstaver, tall og spesialtegn, i tillegg tar den inn minimum og maksimum lengde for teksten i feltet.

2.1.3.2 Tilrettelegging for oversettelse

I android er det en XML-fil som inneholder alle strengene programmet bruker. Denne filen heter “strings.xml” og finnes i mappen “values”.

Hvis applikasjonen skal oversettes til norsk gjøres dette ved å lage en ny mappe kalt “values-no”. Inne i denne mappen opprettes det strings.xml-filer. Mappen legger til strengene fra “values”-mappen på norsk. Her er det viktig at navnet på strengene er det samme i alle strings.xml-filene.



```
<resources>
    <string name="hello_world">Hello World</string>
</resources>
```

Figur 2.11: strings.xml på engelsk. Dette er standardverdien (ligger i values)



```
<resources>
    <string name="hello_world">Hallo verden</string>
</resources>
```

Figur 2.12: strings.xml med norsk oversettelse (ligger i values-no).

Applikasjonen vil automatisk velge “values-no”-mappen om norsk er satt som valgt språk på telefonen. Eksister ikke “values-no”-mappen blir “values”-mappen valgt.

Løsningen er det tilrettelagt for oversettelser fordi applikasjonens strenger er i strings.xml-filer. Hvilket som helst språk kan derfor enkelt legges til.

2.1.4 Design og brukeropplevelse

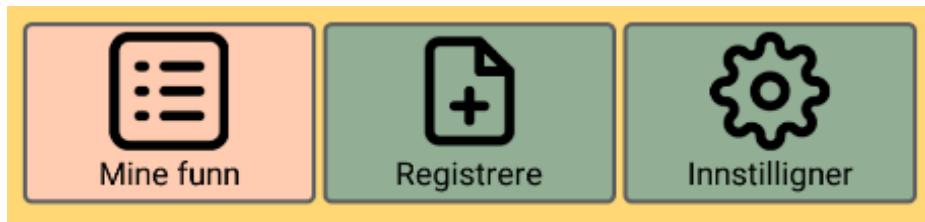
Riksantikvaren ga oss ingen retningslinjer for hvordan applikasjonen skulle se ut. De kommuniserte at fokuset for utviklingen skulle ligge i prototypens funksjonaliteten. Implementering av design ble derfor ikke høyt prioritert. Våre tanker og planlegging av designet vil likevel ha verdi i utviklingsprosessen av et endelig sluttprodukt. I designprosessen har vi hatt fokus på brukervennlighet og prinsipper for universell utforming. Riksantikvarens nettprofil²⁵ ble benyttet til inspirasjon, men ble ikke strengt fulgt.

Vi har hatt et fokus på at systemet skal være intuitiv og lett å bruke. De mest brukte funksjonene skal bare være et tastetrykk eller to unna. Funksjonen "Nytt funn" ble satt som applikasjonens forside. På den måten kan brukeren åpne applikasjonen og starte funnregistreringen uten å først måtte navigere til "Nytt funn"-siden.

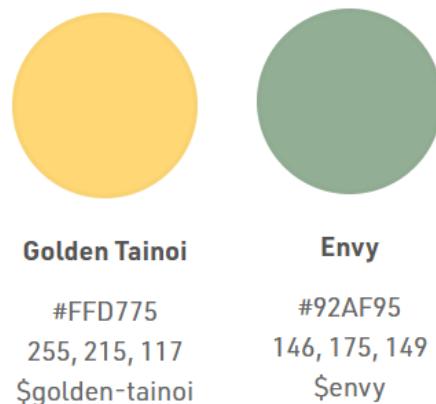
Systemet skal ta hensyn til personer med nedsatt syn. Det er forventet at applikasjonen vil brukes mye utendørs på dagtid. Lysforhold må derfor også taes hensyn til. Tilrettelegging for disse forholdene og personer med nedsatt syn skal gjøres gjennom sterke kontraster, gode fargevalg, stor skrift og ledbare fonter.

Ikoner, tekst og farger skal veilede brukeren (se [figur 1](#)). De hjelper til med å unngå feiltasting, og forsikrer at brukeren forstår hvilke funksjonaliteter som tilbys. Ikoner og bilder skal være relevante til den informasjonen de hinter til. Ikonene, teksten og fargene skal også helst være relevante i den konteksten de skal brukes i. Farger ble valgt ut fra Riksantikvarens fargepalett (se [figur 2](#)).

Informasjonen i applikasjonen skal være lett forståelig. Den skal tas hensyn til at personer har ulike mengder erfaring med metallsøking og bruk av mobilapplikasjoner. Feilmeldinger skal være korte og enkle å forstå. Det har derfor blitt tatt i bruk et enkelt språk i applikasjonen.



Figur 2.13: Enkel navigasjon og meningsfylte ikoner. Den siden brukeren er på har annen farge(rosa).



Figur 2.14: Fargene vi valgte ut av RA sin fargepalett.

2.2 Backend-dokumentasjon

Delen av systemet som tar hånd om data er kalt “backend”. I vårt prosjekt er backend-en laget med ASP .NET Core Web API og Microsoft Entity Framework rammeverk ([se teknologier og verktøy](#)). Løsningen kjører på skyplattformen Microsoft Azure. Det brukes også en SQL-database på Azure-serveren for lagring av data. Azure har et integrert sikkerhetssystem som beskytter dataene i databasen.

2.2.1 API - bindeleddet mellom database og frontend

API er en del av backend. API-ets rolle er å sørge for kommunikasjon mellom front- og backend. Det fungerer ved å sende informasjon gjennom en HTTP-request. Dette kan gjøres ved å sende JSON⁹ i HTTP-en¹³. Informasjonen kan også sendes direkte i URL-en¹¹. HTTP-metoden som anvendes avgjør hvordan informasjonen sendes. HTTP-metoder vi har brukt er:

- GET-metoder¹⁰ som skrives direkte i URL-en. Brukes til å hente data

- POST-metoder som sender data gjennom “request body”¹⁷-en. Brukes til å lagre data
- PUT-metoder¹² som sender data gjennom “request body”-en. Brukes til å oppdatere data
- DELETE-metoder som skrives direkte i URL-en. Brukes til å slette data

2.2.2 Backend

Backend består i hovedsak av “CRUD”-oppgaver (Create, Read, Update, Delete). Dette er oppgaver relatert til opprettelse, henting, oppdatering og sletting av elementer i databasen.

Oppgavene er definert i form av metoder. Metodene er koblet til klasser i koden.

Klassene i vår kode er kalt:

- Bruker
- Funn
- Grunneier
- Postadresse
- GBNr

Klassene tilsvarer tabeller i databasen som vist i [figur 2.17](#).

Det er to versjoner av hver klasse. En klasse for innsendt data og en klasse for lagring i databasen. Vi vil referer til klassene for innsendt data som “inn-klasse”. Klassene i databasen vil bli referert til som klasser.

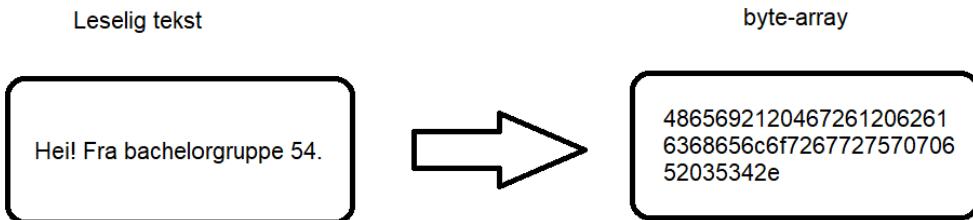
Inn-klassene inneholder ingen ID. Når et inn-klasse-objekt sendes til backend lages det et klasse-objekt med inn-klassens data. Klasseobjektet har en ID. Det blir i tillegg gjort noen endringer for å sikre data.

2.2.3 Sikring av data

Passordet som tidligere var i en streng blir lagret som et byte-array og kan ikke lenger leses direkte. En streng er en ren tekst som kan leses. Et byte-array deler opp alle tegnene i en tekst. Hvert tegn blir oversatt til sin bit-verdi. Bit-verdier defineres i heksadesimaler.

For eksempel:

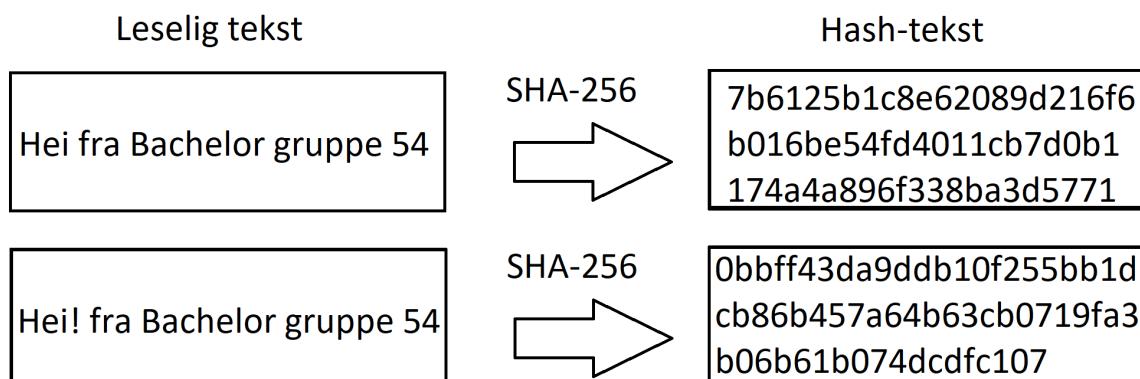
Mellomrom oversettes til “20”, “Å” oversettes til “c5”. Da ser man ikke ser hva passordet er med en gang. Dette styrker sikkerheten til applikasjonen.



Figur 2.15: Fra leselig tekst til byte-array.

For å videre forbedre sikkerheten blir passordet hashet sammen med et “salt”⁸. Ett salt er en tilfeldig tekst som tilføyes et passord før det lagres i databasen. Saltet er også hashet. Salte lagres også i databasen. Hashing brukes i kryptering ved hjelp av en hash-algoritme. Det gjør en leselig tekst om til en meningsløs tekst med tall og bokstaver. Hash-verdien blir like lang samme hva den originale teksten er. Likevel skal en liten endring i teksten gi en drastisk annerledes hash. Se figuren nedenfor.

Dette gjør at man ikke kan lese en brukers passord i databasen. Saltet hindrer at man kan finne et mønster i hash-teksten og gjette passordet.



Figur 2.16: Nesten like tekster produserer svært ulike hash-er. SHA-256 er en hash-algoritme.

2.2.4 Databasen

Løsningen bruker Entity Framework rammeverket, og utnytter rammeverkets støtte for “Code-first”⁶ database-design. Code-first database-design fungerer ved at man lager klasser i backend først. Deretter gjør Entity Framework dette om til en reell database. Opprinnelig brukte backend-en en SQLite database. Databasen ble senere oppgradert til en SQL database på en Azure server.

I hovedsak dreier håndteringen seg rundt to hovedentiteter, bruker og funn. Bruker og funn er koblet sammen ved at hver bruker har en liste med funn knyttet til seg. Andre entiteter databasen tar hånd om er postnummer, grunneiere og gårds- og bruksnummer. Vi har valgt å bryte entitetene ned i disse gruppene slik at databasen oppnår høyere grad av normalform².

2.2.5 Tabellene i databasen

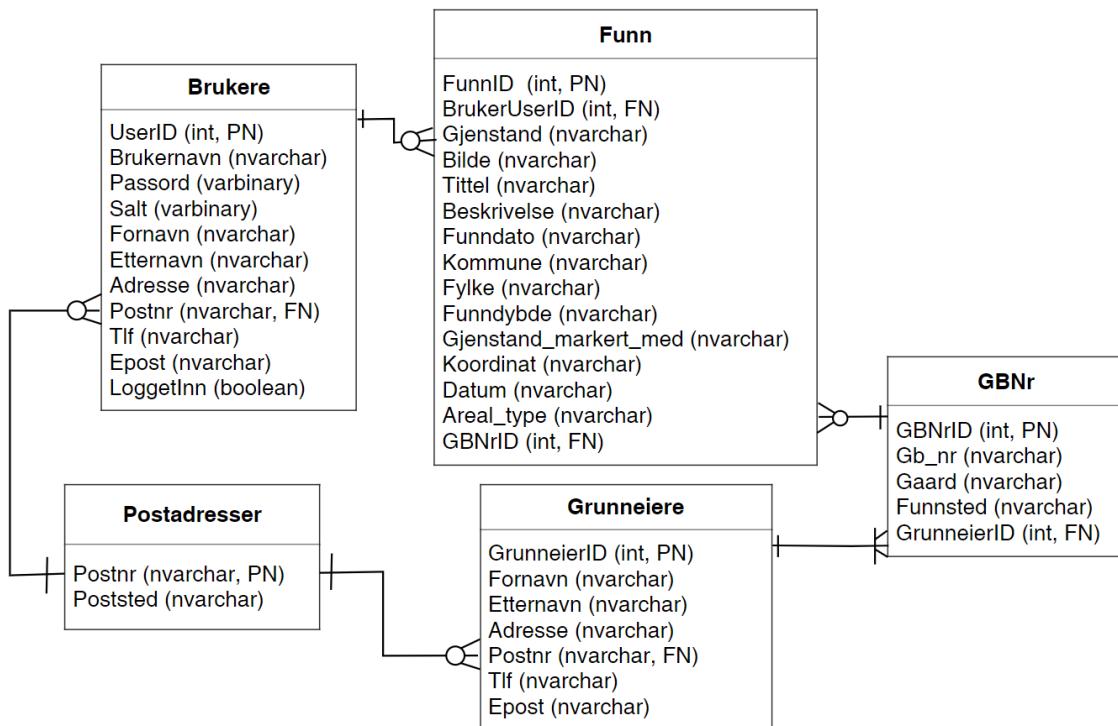
I databasen lagres det funn-informasjon og brukerinformasjon i henholdsvis tabellene "Funn" og "Brukere". Disse er koblet sammen slik at én bruker kan ha ingen eller flere funn. Ett funn må derimot tilhøre én bruker. Dette vises som en sirkel med tre streker i databasediagrammet ([figur 2.17](#)). Det kalles et en-til-mange forhold. Alle forholdene som beskrives videre er også en-til-mange forhold.

Et funn har et felt "GBNrID". Feltet tilsvarer en ID som referer til gårds- og bruksnummeret (gbnr). ID-en kobler "Funn"-tabellen til "GBNr"-tabellen.

Et funn knyttes til et gbnr, flere funn kan knyttes til samme gbnr.

"GrunneierID" kobler "GBNr"-tabellen til "Grunneiere"-tabellen. Grunneiere-tabellen inneholder personinformasjon om grunneieren. En grunneier knyttes til et eller flere gbnr. Dette vises som en loddrett strek med tre streker i figuren. Et gbnr kan derimot bare knyttes til én grunneier.

Både "Grunneiere"- og "Brukere"-tabellen har et felt kalt "Postnr". Feltet referer til "Postadresser"-tabellen. Her kan man finne poststeder som tilhører et postnummer. En grunneier eller en bruker har bare ett postnr, men ved ett postnr kan det bo flere grunneiere eller en brukere.



Figur 2.17: Diagram som viser oversikt over databasen

Leserveileitung:

Diagrammet i figur 2.17 viser en oversikt over klassene i databasen. Klassene som tas inn har samme struktur og koblinger. Klassenavnene vil være innFunn, innBruker, innGBNr og innGrunneier. Når vi prater om innKlasse-objekter senere i teksten vil vi derfor referere til samme figur.

2.2.6 Bruker-metodene

```
2 references
public Task<string> CreateUser(IInnBruker bruker);
2 references
public Task<string> EditUser(IInnBruker bruker);
2 references
public Task<string> DeleteUser(string brukernavn, string passord);
2 references
public Task<Bruker> GetUser(String brukernavn);
2 references
public Task<string> LogIn(string brukernavn, string passord);
2 references
public Task<string> LogOut(string brukernavn);
```

Figur 2.18: Metodene for behandling av brukere. Metoder som returnerer strenger (<string>) gjør det for å bistå i feilsøking for utviklerne.

Det er seks metoder som behandler brukerinformasjonen. Disse er CreateUser, EditUser, DeleteUser, GetUser, LogIn og LogOut (figur 2.18). Alle brukermetodene bruker brukernavn til å sjekke om brukeren eksisterer i databasen når de kjører. Feilmelding blir vist i applikasjonen om en metode ikke kjører. Dette gjelder alle metodene i løsningen.

2.2.6.1 CreateUser

Metoden for å lage en bruker (“CreateUser”) tar et “InnBruker”-objekt som parameter. Av dataene i InnBruker-objektet opprettes det en ny bruker.

CreateUser sjekker om brukernavnet eksisterer. Eksisterer brukernavnet i databasen vil det ikke være mulig å opprette brukeren. Eksisterer ikke brukernavnet vil det opprettes en ny bruker.

2.2.6.2 Postadresse

Ved registreringen og redigering av en bruker, sjekkes det om det oppgitte postnummeret til InnBruker-en finnes i databasen. Eksisterer postnummeret i databasen settes brukerens postnummer lik dette. Eksisterer ikke postnummeret, opprettes det i databasen. Dette gjøres fordi “Postadresser” er en egen tabell som må oppdateres individuelt. Tabellen inneholder feltene “Postnr” og “Postadresse”. Tabellen er koblet til en bruker gjennom “Postnr”-feltet.

2.2.6.3 EditUser

“EditUser”-metoden endrer informasjonen til eksisterende brukere. Metoden tar et InnBruker-objekt som parameter. Om brukeren ikke finnes i databasen, kan den heller ikke redigeres. Eksisterer brukeren, vil brukerinformasjonen oppdateres.

2.2.6.4 DeleteUser

Metoden for å slette en bruker er “DeleteUser”. Den tar inn en brukernavn-streng og en passord-streng som parameter. Først slettes alle funn knyttet til brukeren. Deretter slettes brukerinformasjonen fra databasen. I tillegg til brukernavnsjekken, kjøres en passordsjekk. Begge sjekkene må bestås for at slettingen skal gjennomføres.

2.2.6.5 GetUser

Metoden for å hente data heter “ GetUser ”. Den tar inn en brukernavn-streng som parameter. Om brukeren finnes i databasen, returneres brukerinformasjonen så den kan vises på frontend-siden.

2.2.6.6 LogIn

Metoden “LogIn” logger brukeren inn. Den tar inn en brukernavn-streng og en passord-streng som parameter.

En bruker har et salt⁸ lagret i databasen. Parameter-passordet er oppgitt i leselig tekst.

Passordet blir hash-et sammen med bruker-saltet. Det må produsere en hash-verdi lik passordet til brukeren i databasen. Da er oppgitt passord riktig.

Brukeren kan da logges inn. I “Bruker”-tabellen er det et boolsk felt kalt “LoggetInn”. Feltet kan enten ha verdien “true” eller “false”. Ved innlogging blir feltet endret til “true”.

2.2.6.7 LogOut

Metoden “LogOut” logger brukeren ut. Den tar inn en brukernavn-streng som parameter. Metoden sjekker om bruker er logget inn. Da endres den boolske verdien “LoggetInn” til “false” i databasen.

2.2.7 Funn-, grunneier- og gbnr-metodene

```
2 references
public Task<string> RegistrerFunn(InnFunn nyttfunn, String brukernavn);
1 reference
public Task<List<Funn>> GetAllUserFunn(String brukernavn);
2 references
public Task<string> DeleteFunn(int funnID);
2 references
public Task<string> EditFunn(Funn f);
2 references
public Task<Funn> GetFunn(String brukernavn, int funnID);
```

Figur 2.19: Metodene for behandling av funn, grunneier og gbnr.

Det er fem metoder som behandler funn. Disse er RegistrerFunn, GetAllUserFunn, DeleteFunn, GetFunn, EditFunn (figur 2.19). Feilmelding blir vist i applikasjonen om en metode ikke kjører. Dette gjelder alle metodene i løsningen.

2.2.7.1 RegistrerFunn

Metoden “RegistrerFunn” oppretter et funn. Metoden tar inn et “InnFunn”-objekt og en brukernavn-streng som parametre.

Ved registrering av et nytt funn sjekkes det om feltene innGBNr, innGrunneier eller postnummer er tomme. Dette er felt i innFunn-objektet.

Hvis et av feltene er tomme starter konstruksjonen av et nytt funn med tomme felt. Dette er fordi en bruker skal kunne lagre et funn med manglende data. Det må være støtte for dette ettersom registrering av funn (illustrert i [2.1.2.4](#)) registrerer kun noen spesifikke felt med data. Alle feltene må derimot være utfyldt for å sende et funnskjema.

Hvis ingen av feltene er tomme vil et nytt funn konstrueres med dataene fra innFunnet. Først må det sjekkes om dataene i feltene finnes i databasen. Hvis dataene ikke finnes i databasen, må de registreres. Hvis dataene finnes, hentes de ut av databasen.

Til slutt vil et nytt funn konstrueres med innGBNr, innGrunneier og postnummer fra innFunnet.

2.2.7.2 EditFunn

“EditFunn”-metoden redigerer et eksisterende funn. Metoden tar et “Funn”-objekt som parameter. Her var det nødvendig å bruke et “Funn”-objekt framfor et InnFunn. Dette må gjøres fordi innFunn-objektet ikke har en en ID som brukes til å identifisere objektet⁴.

EditFunn bruker de samme sjekkene som RegistrerFunn:

Er den redigerte informasjonen lik et objekt som ikke finnes i databasen, skal det opprettes et nytt objekt. Hvis ikke, skal kun de andre feltene oppdateres uten å lage et nytt objekt.

2.2.7.3 DeleteFunn

“DeleteFunn”-metoden brukes for å slette eksisterende funn. Metoden tar et tall tilsvarende en funn-ID(int) som parameter. Det gjøres en sjekk om funnet finnes i databasen. Funnet blir slettet om det finnes. Hvis funnet ikke finnes får brukeren beskjed om dette.

2.2.7.4 GetAllUserFunn

Metoden for å hente alle funn som tilhører én bruker heter “GetAllUserFunn”. Metoden tar et brukernavn(streng) som parametre.

En bruker og et funn er koblet sammen gjennom et felt kalt “BrukerUserID” i Funn-tabellen (Se figur 2.5).

GetAllUserFunn-metoden vil finne alle funn med BrukerUserId lik brukerens BrukerUserID. Det lages så en liste av alle funnene. Denne listen returneres så den kan hentes ut på frontend.

2.2.7.5 GetFunn

“GetFunn”-metoden brukes for å hente ut et enkelt funn. Metoden tar et brukernavn (streng) og en funn-ID (int) som parametre. Metoden finner først alle funn som tilhører brukeren og legger dem i en kortere liste. Det letes da gjennom en kortere liste når programmet deretter leter etter funnet med riktig ID. Om funnet finnes, returneres det.

Her er det ikke CRUD-metoder for hver av klassene Funn, Grunneier og GBNr fordi det ikke var nødvendig.

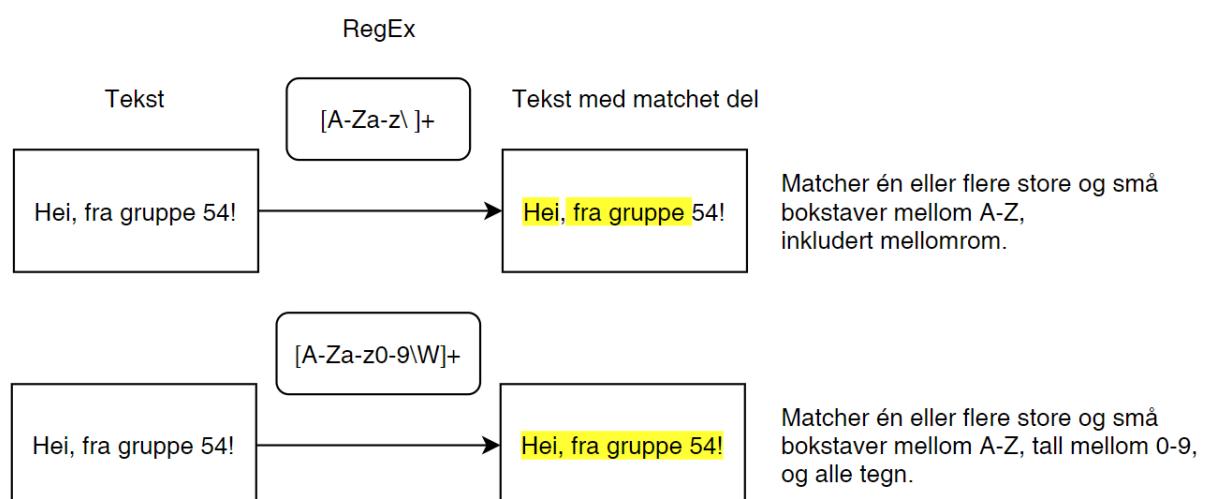
2.2.8 Injeksjonsvern

RegEx⁷ er blitt satt på feltet “Epost” for å hindre et SQL-injeksjonsangrep²⁶.

RegEx-en spesifiserer at innskrivningen må være en gyldig e-postadresse med riktig format. Den hindrer dermed muligheten for å skrive inn SQL-spørninger i feltet. Å ha muligheten åpen for å legge inn SQL-spørninger i feltet kan potensielt la en ondsinnet bruker aksessere backend-koden eller slette databasen.

RegEx-mønstrene for passord og brukernavn brukes for å sjekke metodene før de kjøres.

Om mønsteret er gyldige, er en bruker logget inn.



Figur 2.20: RegEx-mønstre. Tekst som ikke passer mønstret blir ugyldig.

3. Planlegging og kravspesifikasjon

Dette kapittelet handler om innhenting av informasjon, planlegging av gjennomføringen og kartlegging oppgaven. Kravspesifikasjonen er lagt som et vedlegg. Noen viktige punkter fra kravspesifikasjonen er trukket ut i kapittelet.

3.1 Informasjonsinnhenting og forprosjekt

Detektoristene og fylkesarkeologene ble tidlig identifisert som viktige aktører. Detektoristene er sluttbrukerne. Fylkesarkeologene vil behandle informasjon sendt fra applikasjonen. Det var derfor viktig å gjøre seg kjent med aktørene. Å hente inn informasjon om deres prosesser, behov og ønsker var essensielt for å utvikle en god applikasjon.

Formålet med informasjonsinnhenting var å danne oss et grunnlaget for utviklingen av kravspesifikasjonen.

3.1.1 Møte: Gary Brun

I midten av Januar møtte vi den erfarne detektoristen Gary Brun. Han har gjort store funn i flere land i Europa. Gary er kjent fra fjernsynsprogrammet "Hoard Hunters" som vises på History Channel. Han drifter et metallsøkerforum og en britisk nettside. På nettsiden kan brukerne få hjelp med identifikasjon av arkeologiske funn.

Gary er fra Storbritannia og har derfor mest erfaring med søking der. Han har mindre erfaring med søking i Norge. Vi lærte mye av Gary og han ga oss god innsikt i mange aspekter ved metallsøking.

3.1.2 Møte: Riksantikvaren, arkeologer og detektorister

I starten av Februar deltok vi i et møte i regi av Riksantikvaren. Her var det møtedeltakere med høy ekspertise innenfor metallsøking, arkeologi og kulturminneforvaltning til stedet. Henholdsvis arkeologer, detektorister og representanter fra Riksantikvaren. Alle var interessenter i vår prosess. De lærte oss mye om metallsøking og finnerlønnprosessen. De uttrykte også ønsker og behov til applikasjonen.

3.1.3 Møter: Maria Sølversen

Vi hadde flere møter med Maria Sølversen. Det ble også jevnlig holdt kontakt via e-post. I kommunikasjon med Maria uttrykte hun RA sine behov og ønsker til applikasjon. Hun hjalp til med å sette rammer og avgrensninger for prosjektet. Maria bidro direkte i kravspesifikasjonen ved å lage et prosessdiagram som beskrev finnerlønnprosessen.

3.1.4 Markedsanalyse, litteratur og interesseorganisasjoner

For å få innsikt i hva det vil si å være en detektorist og hvilke verktøy de bruker, utforsket vi en rekke forskjellige detektoristgrupper. Vi lærte mye fra å besøke sidene til Norges Metallsøkerforening (NMF) (<https://nmf.nu/>), National Council for Metal Detecting (NMCD) (<https://www.ncmd.co.uk/>) og The Ring Finders (<https://theringfinders.com/>).

Det er ingen digital løsning for funnregistrering i Norge. Det var likevel mulig å hente inspirasjon fra utenlandske applikasjoner. Detektoristene vi pratet med hadde selv tatt i bruk webapplikasjonen DIME (<https://www.metaldetektorfund.dk/>) og en android applikasjon kalt iSmartDetect. I tillegg til å gjøre oss kjent med disse applikasjonene så vi på applikasjonene Minelab Treasure Tracking, Tect O Trak og onX Maps.

I vedlegget “Kravspesifikasjon” har [tabell 5](#) en grundigere gjennomgang av organisasjoner og applikasjoner.

3.2 Kravspesifikasjon

En kravspesifikasjon omhandler oppgavene som avgjør krav og omfang i applikasjonen. Prosjekteier ga oss ikke en konkret kravspesifikasjon. Informasjon innhentet i [punkt 3.1.1](#) til [3.1.4](#) dannet grunnlaget for utviklingen av kravspesifikasjonen.
I punktene under er noen viktige utdrag fra kravspesifikasjonen. Full kravspesifikasjon er lagt til som [vedlegg 2](#).

3.2.1 Brukerhistorier

En brukerhistorie er uformell forklaring av en funksjon. De er skrevet fra brukerens perspektiv.

Vi valgte å skrive brukerhistorier for å få oversikt over hvordan planlagt funksjonaliteten vil gi brukeren verdi.

“Som en detektorist,

ønsker jeg å registrere og lagre løse funn i applikasjonen,

slik at jeg har den informasjonen jeg trenger når jeg skal melde funn eller sende funnsekjema.”

“Som en detektorist,

ønsker jeg å melde inn funn i applikasjonen,

slik at jeg letttere kan gi beskjed til riktig person om at jeg har gjort et funn.”

“Som en detektorist,

ønsker jeg at applikasjonen husker informasjon om meg,

slik at det blir mindre å fylle ut i funnsekjemaet senere.”

“Som en detektorist,

ønsker jeg å kunne se oversikt over mine tidligere funn,

slik at jeg enkelt kan se hva jeg har funnet før og ha en liste over min samling.”

“Som en detektorist,

ønsker jeg å kunne sende inn funnsekjema gjennom applikasjonen,

slik at jeg sparer tid, informasjonen er korrekt og at applikasjonen kan fylle ut store deler av feltene automatisk.”

“Som en detektorist, ønsker jeg å fort kunne registrere et oppdaget funn, slik at ingen

opplysninger går fortapt i korttidsminnet og jeg kan fort få tilbakemelding fra

fylkesarkeologen.”

“Som en fylkesarkeolog,

ønsker jeg å ha henvendelser om funn på ett sted,

slik at det blir letttere for meg å behandle henvendelsene.”

“Som en fylkesarkeolog,

ønsker jeg en app som hjelper brukerne med å fylle ut funnsekjemaet riktig,

slik at jeg ikke trenger å spørre om tilleggsinformasjon senere.”

Tabell 3.1: Brukerhistorier

3.2.2 Funksjonelle krav: Minimumskrav

Det ble utarbeidet en liste over funksjonelle krav. I programutvikling er funksjonelle krav definisjonen av en funksjon eller et komponent. Tabellen under inneholder de funksjonelle kravene vi har valgt å prioritere. Dette er krav som må implementeres. Verdien fra implementeringene må realiseres for ønsket sluttprodukt. Vi har valgt å definere tabellen som minimumskrav.

Lang	2-8 uker
Medium	1 uke
Kort	1-3 dager

Tabell 3.2: Fargekoding av prosesstid som brukes i tabellene nedenfor

Funksjon	Beskrivelse	Verdi	Prioritet	Tid
Registrere løse funn	Utfylling av viktig info om funnet. Tar bilde og GPS-koordinater med tastetrykk.	Utfylling av innmelding og funnsekjema blir korrekt. <i>F.eks GPS koordinatene kan ikke tastes feil inn.</i>	Høy	Kort
Innmelding av funn (Epost)	Innsending av funnmelding ved et tastetrykk. E-post fyller seg ut med nødvendig info og sendes riktig fylkesarkeolog.	Fylkesarkeologen får informasjonen riktig utfylt og i et godt format.	Høy	Kort
Logg inn	Registrering og innlogging. Funn knyttes til bruker. Lagres i skyen. Brukerinfo brukes i automatisk skjemautfylling.	- Tidsbesparende og kvalitetssikrende (ved skjemautfylling). - Tilgang på funnene i funnliste. - Tilgang fra andre enheter.	Høy	Medium
Oversikt over egne funn	Oversikt i form av en liste over brukerens tidligere funn. Hvert enkelt funn kan redigeres(Fylle ut mer info og endre info.)	- Brukeren får en oversikt over tidligere funn. - Brukeren slipper å legge inn all info ved funnregistrering. (Brukeren er ofte ute i felten. Det kan det være utfordrende å fylle ut mye informasjon.)	Høy	Kort
Sende funnsekjema	Automatisk utfylling av funnsekjemaPDF. Sender PDF via epost til fylkesarkeologen.	- Fylkesarkeologen får tilsendt skjema digitalt. - Minimerer feil ved skjemautfyllingen.	Høy	Medium

Tabell 3.3: Oversikt over minimumskravene

3.2.3 Krav utover minimumskravene

Tabellen under inneholder funksjonelle krav utover kravene nødvendige for ønsket sluttprodukt. Implementasjon av kravene avhenger av at minimumskravene allerede er implementert og av hvor mye tid som gjenstår til leveranse. For Riksantikvaren vil funksjonene definert i tabellen kunne brukes som inspirasjon utover funksjonalitet som blir presentert i prototypen.

Funksjon	Beskrivelse	Prioritet	Tid
Kvalitetssikre informasjonen på funnsekjema	Videre automatisering av utfylling. Brukeren fyller kun ut felter som er strengt nødvendig å fylle ut manuelt. Funnsekjema blir kun sendt om det er riktig utfylt.	Middels	Lang
Brukerveiledning i applikasjonen.	Går i gjennom applikasjonen side for side og beskriver funksjonaliteten.	Middels	Medium - Kort
Manuel GPS innsetting	Innskrivning av GPS-koordinater eller valg av lokasjon på kart i applikasjonen.	Middels	Kort - Medium
Info lover og regler-side	Informasjonsside med opplysninger om lover, regler og linker til ressurser.	Middels	Kort - Medium
Automatisk oppdagelse av Gårds/Bruks nr.	Automatisk innhenting og utfylling av gårds- og bruksnr. (Hentet ved bruk av GPS-koordinatene knyttet til funnet.)	Lav	Ukjent - Lang
Bytte språk	Muligheter for å velge andre språk i applikasjonen (engelsk, polsk, etc)	Lav	Kort
Panoramabilde av terrenget	Ta bildeserie og opprette panoramabilde av funnsted.	Lav	Kort - Medium
Kommunikasjon i app (svar på app)	Svar på funnmelding og funnsekjema i applikasjonen.	Lav	Lang
Feilmelding ved søk på fredet område	Sjekke om området er fredet. Returnerer melding til bruker om området er fredet.	Lav	Lang
Presentere funn VIA webservice	Enkel webside som kan vise oversikt over egne funn.	Lav	Medium
Generering av Funnummer	Genererer enkel kode og knytter den til funnet. <i>Hensikt: Koden kan skrives på funnposen for å lettere koble funnsekjema med det innsendte funnet.</i>	Lav	Kort

Tabell 3.4: Oversikt over andre ønskede egenskaper

3.2.4 Logiske datamodeller

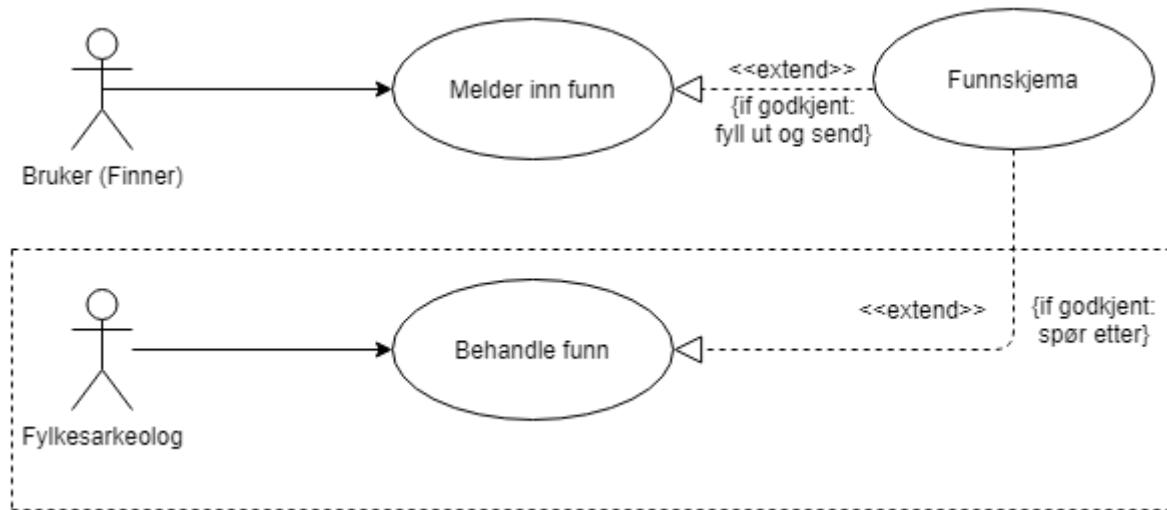
Under arbeidet med kravspesifikasjonen ble det tegnet:

- Prosessdiagram for å beskrive finnerlønnprosessen slik den fungerer i dag
- Flytdiagram for å vise hovedflyten i programmet
- Wireframe som viser en grov skisse av funksjonaliteten
- Seksjonsdiagram som viser funksjonskall (i koden)

Se [vedlegget ‘Kravspesifikasjon’ under ‘Logiske modeller’](#) for å ta en nærmere titt på disse diagrammene.

Vi har valgt å trekke frem use-case diagrammet.

Use-case diagrammet utforsker hvordan bruker og fylkesarkeologen medvirker i innmelding av funn. Brukerens handlinger er hovedfunksjonene i applikasjonen.



Figur 3.1: Brukerinteraksjon (Hoved use-case diagram)

3.3 Strukturering av tiden

Det ble lagt en grov plan for tidsbruk for de forskjellige delene av prosessen.

Ferie	Lav	Middels	Høy
-------	-----	---------	-----

Måned	Januar			Februar				Mars				April				Mai				Juni	
Uke	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
Behovsanalyse																					
Videre planlegging																					
Design																					
Utvikling																					
Fase slutt testing																					
Testing, Validering og brukertesting																					
Skrive hovedrapport																					
Forberede muntlig presentasjon																					
Dokumentasjon																					

Tabell 3.5: Gruppens planlagte tidsbruk i løpet av semesteret.

4. Prosessdokumentasjon og metodikk

4.1 Covid-19

På grunn av koronasituasjonen har alt prosjektarbeid i løpet av prosjektperioden foregått hjemmefra. Det ble brukt ulike programmer, verktøy og Internett-plattformer for å bistå gruppearbeidet. Slik kunne gruppen enkelt dele arbeidet med hverandre, og forsikre god og jevnlig kommunikasjon.

4.2 Teknologier og verktøy

Program	Hva det er	Hva vi brukte det til og hvorfor
Discord 	En kommunikasjonsplattform med støtte for tekst- og stemme chat.	Hovedverktøyet for kommunikasjon i løpet av bachelorprosjektet. Vi brukte programmet til å dele lenker, ideer og ha digitale møter.
Git / Github  	Git er et versjonskontrollsyste. Git lar utviklere jobbe sammen, mens de har hele prosjektet på sin lokale datamaskin. Github gir muligheten til enkelt å migrere endringer fra utviklerne sine lokale prosjekter.	Utover versjonskontroll ble GitHub aktivt brukt for kode gjennomgang. Ingen ny kode ble slått sammen med hoved applikasjonen uten at en annen utvikler gikk gjennom koden først.
Azure 	En tjeneste som lar brukere bygge, teste og kjøre applikasjoner i skyen.	Løsningen kjøres på Azure-skyplattformen. Databasen i programmet er også på Azure.
Trello 	Et organiseringsverktøy. Lager kort som viser prosesjonen av forskjellige oppgaver.	Ga en oversikt over statusen på de forskjellige oppgavene; hva som skal gjøres, hva som jobbes med, hva som er fullført. Fikk også oversikt over hvem som har utført de ulike oppgavene.
Google docs 	En nettbasert plattform der man kan skrive i dokumenter sammen med andre personer.	Gruppen har brukt google docs til å skrive styringsdokumenter og hovedrapporten.

Visual studio 	En “Integrated Development Environment” (IDE) eller integrert utviklingsmiljø som brukes til å skrive og kjøre kode i.	Var det primære utviklingsmiljøet som ble brukt ved utvikling av backend-siden.
Postman 	En plattform der man kan teste nett API’et.	Bruk til å teste om API metodene fungerte som forventet. Nyttig for API gruppen så de slapp å teste gjennom applikasjonen.
SQL 	Et dataspråk egnet til å administrere data i administrasjonssystemer for relasjonsdatabaser.	Ble brukt i sammenheng med Azure databasen, som opererer på SQL.
Google calendar 	Et kalender program hvor flere personer kan se og endre samme kalender.	Bruk til å planlegge når gruppemedlemmer var ledige og kunne arbeide med prosjektet.
Android Studios 	En utviklingsplattform for android applikasjoner, hvor man kan kjøre programmet i en emulator.	Bruk for utvikling og testing av android applikasjonen.
ASP .NET Core 	Et “open-source” .NET rammeverk. Den tilbyr mange biblioteker og rammeverk som kan hjelpe i utvikling av flere typer prosjekter	Bruk som hovedrammeverk i backenden.
Microsoft Entity Framework 	Et rammeverk brukt i .NET applikasjoner. Entity Framework bidrar til enkel databasebehandling.	Bruk for å lage en database fra klasser i koden automatisk.

Tabell 4.1: Utviklingsverktøy.

4.3 Utviklingsmetode

Scrum er en systemutviklingsmetode. Scrum handler om å bryte ned prosjektet i mindre deloppgaver med korte tidsfrister og klare progresjonsmål. Tidsfristene er gjerne på én uke og blir kalt sprinter. Det er vanlig med korte "standup"-møter hver dag, med større møter på slutten av hver sprint. Gruppen hadde tre ukentlige standup-møter på Discord. På stand-up møtene ble det diskutert hvordan gruppen lå ann i forhold til sprinten.

Hvorfor vi har brukt Scrum:

- Korte sprinter på en til to uker gjør det lettere å holde produkteier oppdatert på endringer
- Gjør det lettere å gjøre endringer i krav underveis i utviklingen
- Standup-møtene oppfordrer til jevn kommunikasjon og arbeid mellom gruppemedlemmene

4.3.1 Sprinter

Sprintene varte to uker hver. I tillegg til standup-møtene, hadde gruppen større møter ved start og slutt av sprintene. Gruppen hadde mål om når faser av prosjektet skulle være ferdig ([Tabell 3.5](#)). Mindre og mer konkrete mål ble satt for hver sprint. Sprint-målene var "løse" og dersom gruppen ikke klarte å ferdigstille en funksjon kunne man jobbe med den uken etter. Hver sprint ble det notert ned løse beskrivelser av hva gruppen ønsket å oppnå i en tabell ([Tabell 7.9](#)). Det ble også lagd trello-kort med konkrete oppgaver som måtte løses for sprinten.

4.4 Frontend-utvikling

4.4.1 Kort om utviklingsprosess

De første sidene som ble opprettet var de viktigste for hovedfunksjonaliteten i applikasjonen. Disse sidene var: registrer funn, mine funn og enkelt funn. Det ble implementert et system for lokal lagring. Slik var det mulig å utvikle og teste applikasjonen parallelt med databaseutviklingen. Etterhvert som grunnfunktionaliteten i applikasjonen kom på plass, ble det utviklet sider for: innstillinger, innlogging, registrering av bruker, og informasjon. Da mesteparten av applikasjonens funksjonalitet var på plass, ble den koblet sammen med serveren. Dataene ble ikke lenger lagret lokalt, men i databasen. Til slutt gjenstod det å fikse feil og gjøre små justeringer.

4.4.2 Versjonskontroll

Under hele utviklingen av løsningen ble GitHub aktivt brukt. Dette forenklet distribusjon av nye versjoner av koden til alle utviklerne. Under vårt arbeid jobbet utviklerne på hver sin git branch. En git branch er en versjon av koden som kan endres uten å påvirke hovedversjonen av koden. Slik ble endringer mellom hovedversjonen (master-branchen) og nye versjoner (nye brancher) sammenlignet med hverandre. Git branches gjorde dette enkelt og oversiktlig. Dette gjorde at utviklerne kunne se over hverandres kode før den ble lagt til master-branchen. Det hjalp med å unngå feil i koden.

4.4.3 Utfordringer

I denne delen trekkes det frem et par utfordringer som vi møtte på under frontendutviklingen.

4.4.3.1 Utfordring med funnskjema-PDF

Å redusere feil i og effektivisere utfyllingen av funnskjemaet var et viktig mål. Dette var en hovedfunksjonalitet. Mange arbeidstimer gikk med på å utforske den beste måten å implementere denne funksjonaliteten på.

Flere eksterne biblioteker kunne blitt brukt til å gjøre opprettelse og skriving til PDF letttere enn løsningen vi kom frem til.

Blant bibliotekene vi så på var [iText7](#), [Android-XML-to-PDF-Generator](#) og [Foxit](#). (Se [vedlegg 4](#))

En utfordring ved å bruke disse eksterne bibliotekene var tilgjengeligheten. Bibliotekene måtte ofte kjøpes tilgang til. Vi fant muligheter for å få kostnadsfri tilgang til noen av bibliotekene. Men det viste seg å være en tidkrevende prosess å avklare bruk av bibliotekene og få tilgang på en gratis-lisens.

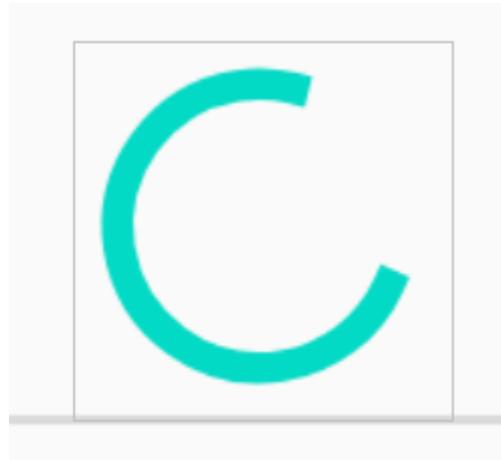
Det ble tatt et valg på å bygge en egen PDF-generende metode. Metoden oppretter og fyller ut en funnskjema-PDF. Denne PDFen vil se identisk ut med det originale funnskjemaet Riksantikvaren og fylkesarkeologen opererer med.

Metoden benyttet seg av “androids graphics”-klasser²³ som Paint og Canvas til PDF-genereringen. Ingen eksterne biblioteker blir brukt.

4.4.3.2 Treg nedlasting av data

I applikasjonen hadde vi problemer med varierende tid på nedlasting av data fra serveren. Noen ganger kan det ta lengre tid enn forventet å hente data fra serveren. Dette kan særlig være tilfellet når applikasjonen brukes ute i felten, hvor dekning og internett hastighet kan variere. Løsningen vår på dette problemet var å implementere en sirkel som snurrer rundt for

å indikere at applikasjonen holder på å laste ned data fra serveren. På denne måten ser brukeren at applikasjonen faktisk gjør noe og at den ikke har stoppet.



Figur 4.1: Bilde av sirkel som indikerer innlasting av data.

4.4.4 Viktige beslutninger

4.4.4.1 Design

Design er ofte en prosess som tar lang tid, og mange iterasjoner før man kommer fram til ønsket produkt. Siden applikasjonen skulle leveres som et “proof of concept”¹, var det viktigst å fokusere på applikasjonens funksjonalitet. Implementering av design ble derfor nedprioritert.

4.4.4.2 Info-siden

Da minimumskravene var på plass begynte vi å legge til funksjonalitet fra listen over krav utover kravspesifikasjonen.

Siden som omhandlet informasjon om lover og regler var ikke øverst på listen, men ble prioritert ut ifra tidsestimatet det ville ta å implementere denne siden.

Grunnen til at tidsestimatet var så lavt var fordi mye av tiden brukt i startfasen av prosjektet gikk til å sette seg inn i lover og regler. Under møtene med Riksantikvaren, detektoristene og arkeologene ble det klart hvilke opplysninger det er viktigst å formidle til de som anvender applikasjonen.

4.5 Backend-utvikling

Arbeidet på backend-siden startet med å lage klasser til brukere og funn. Det ble utviklet metoder for å lagre, endre, slette og hente ut funn- og brukerinformasjon. Senere ble det også lagt til klasser og metoder for grunneiere og gbnr. I tillegg ble det lagt til metoder for håndtering av inn- og utlogging av brukere.

Til å begynne med ble det brukt en lokal database for å lagre informasjon om de ulike objektene⁴. Prosjektet ble senere flyttet til en server. Først da kunne metodene i backend kjøres fra frontend. Databasen ble også flyttet til en server-database.

4.5.1 Database initialisering

Som tidligere nevnt bruker backend-en Entity Framework. Entity Framework ser på klassene i programmet og lager en SQLite-fil ut av det. Dette er den lokale databasen. Dersom databasen ikke eksisterer, produseres den når programmet kjøres.

For å teste backend-metodene trengtes det gyldige data. Derfor var det nyttig å ha et skript som fyller databasen med objekter satt opp slik som ønsket. Dette skriptet heter “DBInit”, som står for database initialisering. DBInit definerer hva databasen skal inneholde. Om databasen er tom eller slettet, fylles den med dataene fra DBInit.

Dette var nyttig for testing lokalt. Dataene kunne lett endres på. I tillegg kunne databasen enkelt slettes og opprettes med oppdatert data.

DBInit bidro til at utviklingen kom fort i gang. Alternativet var å fylle databasen ved hjelp av SQL-setninger. Et SQL-skript krever mer manuelt arbeid for å oppdatere en database.

Den lokale databasen ble senere erstattet av en SQL-database på Azure-serveren. I Azure-databasen ble det dannet tabeller basert på klassene i koden automatisk. Tabellene måtte manuelt finjusteres og kobles sammen.

Med Entity Framework skrives databasebehandlingen i LINQ¹⁹. Azure oversetter LINQ-setninger til SQL-setninger av seg selv. Dette sparte utviklerne fra å måtte manuelt skrive om LINQ-setningene til SQL.

Etter implementering av en Azure-database måtte endringer i databasen gjøres gjennom Azure sine portaler. Endringene på Azure gjøres med SQL-setninger.

Tidligere testet og arbeidet backendutviklerne på lokale databaser. Når Azure-Databasen ble koblet opp ble arbeidet flyttet til databasen og applikasjonen ble koblet til.

4.5.2 Utfordringer

4.5.2.1 Database på serveren

Da prosjektet ble flyttet over til en Azure server, ble det opprettet en database på serveren som ikke kunne aksesseres. Innholdet var dermed ikke mulig å se. Oppkoblingen mot en server var derimot nødvendig for å binde sammen frontend-applikasjonen med backend-funksjonene.

The screenshot shows the Microsoft Azure portal interface for a SQL database named 'FunnAPIDb'. The left sidebar contains navigation links for Overview, Activity log, Tags, Diagnose and solve problems, Quick start, and Query editor (preview). The main area displays the database structure with tables like 'dbo.brukere', 'dbo.funn', 'dbo.GBNr', 'dbo.grunneiere', and 'dbo.postadresser'. A message box indicates 'Showing limited object explorer here. For full capability please open SSDT.' Below the table list, there are sections for Views and Stored Procedures. On the right, the 'Query editor (preview)' tab is active, showing three tabs: 'Query 1', 'Query 2', and 'Query 3'. The 'Query 3' tab contains the following SQL code:

```
1 SELECT TOP (1000) * FROM [dbo].[brukere]
```

The results section shows a table with columns: UserID, Brukernavn, Passord, Salt, Fornavn, and Etternavn. The data includes several rows of user information. At the bottom of the results table, a green bar indicates 'Query succeeded | 0s'.

Figur 4.2: SQL-databasen på Azure.

Dette ble løst ved å ta i bruk en Azure SQL database. Å koble prosjektet opp mot en Azure database var utfordrende. Det var i tillegg en mangel på god dokumentasjon for dette brukstilfellet. Det ble gjort flere forsøk på å migrere databasen. På første forsøk gikk den uaksesserbare serverdatabasen tapt.

I neste forsøk ble server-databasen korrekt integrert i prosjektet. I den nå aksesserbar databasen var det nå mulig å se hva som ble lagret. Databasen hjalp også med å holde styr på koblingene mellom de ulike tabellene.

4.5.2.2 Testing

Hvor det tidligere bare var behov for å teste backend-metodene lokalt, var det nå behov for å teste metodene både lokalt og på serveren. Testingen foregikk ved at programmet ble kjørt. Deretter ble metodenavnet og de tilhørende parameterne skrevet rett inn i URL-en¹¹.

Prosjektet måtte “publiseres” til serveren for at programmet skulle oppdateres i henhold til koden. Når koden ble oppdatert, måtte den publiseres på nytt. Etter publisering, ble samme testmetode brukt på server-siden. Dette viste seg å være både tidkrevende og tungvint siden alt måtte manuelt skrives inn.

Ved hjelp av Postman ble det utført raskere testing av metoder med ulike parameterverdier. Postman hjalp i tillegg med å holde en oversikt over alle metodene.

På frontend-siden ble det brukt JSON⁹ for å teste metodene. JSON brukes til lagring og sending av data. Et JSON-objekt har en struktur som består av attribut- og verdipar. Se figur 4.4. På Postman kan man fylle inn felt som automatisk omgjøres til et JSON-objekt.

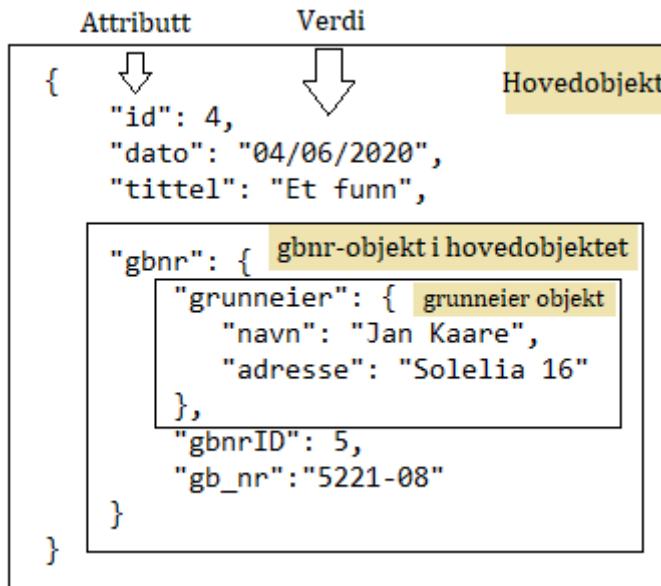
Gjennom Postman var derfor testing både lokalt og på serversiden mulig på ett sted.

The screenshot shows the Postman interface with the following details:

- Request Method:** POST
- Request URL:** https://funnapi.azurewebsites.net/funn/RegisterFunn
- Body Type:** form-data
- Body Fields:**

KEY	VALUE
image	/9j/IVBORw0KGgoAAAANSUhEUgAAA9Q...
funndato	04/08/2020
kommune	Sollia
fylke	Innlandet
funnbynde	0.321m
- Response Status:** 200 OK
- Response Time:** 29.49 s
- Response Size:** 556 B
- Response Content:** Funn is registered!

Figur 4.3: Postman dashboard. Her er “RegistrertFunn” kjørt på server. Nederst står tilbakemeldingen.



Figur 4.4: Et eksempel på et JSON-objekt som inneholder flere objekter.

4.5.2.3 Innsending av et funnbilde fra frontend til backend

En viktig del av løsningens funksjoner er å ta bilde av et funn. Tidlig i utviklingen ble det forstått at bilder måtte konverteres til Base64-strenger⁶. Prosessen startet med utvikling av metoder som behandlet dette. Metodene virket ved de første testene med strenger av diverse lengder. Det som ikke ble tatt i betrakning var hvor store Base64-strenger kunne bli. URL-er¹¹ kan maksimalt være omtrent 2000 tegn lange. Base64-strenger overstiger ofte denne grensen. Dette gjorde det *umulig* å sende data gjennom URL.

For å løse dette måtte både backend-koden (i tillegg til JSON-objektet⁹ som sendes fra frontend) omformateres til å kun å lese fra "body"¹⁷ i JSON-requesten. Body-en kan inneholde uendelig mange tegn, eller en egendefinert mengde for økt sikkerhet.

Når databasen tar i mot et JSON-objekt med en Base64-streng, kan det behandles på to ulike måter: en plassoptimal måte eller en enkel måte. Den plassoptimale måten gjør Base64-strenget tilbake om til et bilde. Bildet lagres deretter i en folder på API-et. I databasen blir det da lagret en tekst som inneholder lenken til bildet. Dette er mer plasseffektivt ettersom Base64-symboler representerer 6 bit med data, på tross av at symbolet tar opp 8 bits med data. Derfor bruker Base64 33% mer plass enn den opprinnelige filen.

Det andre alternativet er å konvertere Base64-strenget til et byte-array.

 BASE64.txt	Date modified: 25/05/2021 22:16 Size: 4.09 KB
 bizza.png	Size: 3.07 KB

Figur 4.5: En bildefil versus en tekstfil som inneholder Base64 av bildet. Base64 tar mer plass.

Først valgte vi å gå med den plasseeffektive løsningen. Det oppstod da problemer med overføringen av den nye strengen. Vi bestemte oss da for å gå for den enkle løsningen på tross av at den ikke er optimal.

4.5.3 Viktige beslutninger

4.5.3.1 Utelatte metoder

I løpet av backend-arbeidet ble det bestemt å utelukke noen metoder. Disse metodene ble ferdig utviklet og tatt i bruk. Metodene ble utelukket ettersom de ikke lenger virket som ønsket.

4.5.3.1.1 Innlogget-sjekk

Originalt var det en metode for å sjekke om en bruker er logget inn eller ikke. Metoden ble brukt som en sjekk i de andre metodene. Da var det enkelte oppgaver som ikke kunne utføres med mindre en bruker er logget inn. Det viste seg at dette ble upraktisk når en bruker ikke kunne logge inn med mindre de alt var logget ut, og omvendt. Det er likevel lurt å ha oversikt over om en bruker er logget inn eller ikke. Dette er for å unngå at metoder i backend kan kjøres manuelt og ikke gjennom applikasjonen.

Dette ble løst ved hjelp av RegEx⁷.

4.5.3.1.2 Glemt passord

Det var også planlagt en “glemt passord”-funksjonalitet der bruker kunne endre sitt passord ved hjelp av en lenke mottatt på e-post. Denne metoden ble ferdigutviklet og fungerte ved testing lokalt. Etter migrasjonen til Azure oppstod det noen problemer med e-post-sendingen. Denne funksjonen var dessuten utenfor den definerte minimums-kravspesifikasjonen. Av disse grunnene ble den utelukket.

4.6 Hovedrapport

Rapportskrivingen ble jobbet med fortløpende gjennom prosjektperioden. Det ble gjennom hele arbeidsperioden ført ned en prosjektdagbok der all arbeidet ble loggført. I tillegg ble det ført ned annen dokumentasjon som hjalp til med en rask oppstart av skrivingen.

Dokumentasjon som ble skrevet:

- kravspesifikasjon
- forprosjektrapport
- møtereferater
- liste over kilder til ulike nettsteder der vi samlet informasjon
- et dokument med spørsmål til ulike aktører av prosjektet
- oversikt og forklaring av roller i gruppen
- notater fra design- og appinnhold-idemyldring

Dette var dokumentasjon som bistod med å holde styr på de ulike delene av prosjektet gjennom perioden.

4.7 Ressursbruk

For å gjøre gruppen mest mulig effektiv ble det holdt et møte hvor gruppen diskuterte individuelle styrker og svakheter. Utdanningsløp, erfaring og ønsker ble tatt i betrakning. I kartleggingen ble Belbins rammeverk (Voldsund et al., 2020) tatt i bruk. I [vedlegg 3](#) finnes det en oversikt over rollediensjonene i Belbins rammeverk.

Gruppens medlemmer hadde varierende erfaring med de ulike arbeidsoppgavene. Helge og Tor hadde allerede erfaring fra android-applikasjonsutvikling. Deres fokus ble derfor oppbygningen av android applikasjonens utseende og funksjonalitet. Fatima, Benjamin og Adrian hadde fra før erfaring med webutvikling og bruk av programmeringsgrensesnitt (API). Fatima og Adrian fikk derfor ledende roller innen backend-utviklingen: database og API. Benjamin tok en rolle hvor han bidro der behovet var størst.

Tabellen ([Tabell 4.2](#)) viser en oversikt over gruppens spesialiseringer og roller. Tabellen gir en god oversikt over hvor hvert enkelt medlem har sin faglige styrke. I tillegg gir rolleinndelingen en god pekepinne for hvilke styrker og svakheter de andre medlemmene kan forvente seg fra hvert individuelle medlem.

Person	Dev-rolle	Spesialiseringer	Rolle
Fatima	UI/UX ¹⁵ designer, QA, Backend	Design, API, Web	Ressursinnhenteren
Benjamin	Techlead, Frontend	Web, API	Analytikeren, Lagspilleren
Tor	Fullstack	Android	Koordinatoren
Helge	Fullstack	Android	Spesialisten
Adrian	Backend, testing	DB, API, Web	Iverksetteren, Ideskaperen

Tabell 4.2: Viser gruppens spesialiseringer og roller

4.7.1 Refleksjon rundt roller

Det ble identifisert at det var mangel på gruppemedlemmer som fyller rollene i den "handlende" rolledimensjonen. Det ble derfor et fokuspunkt at alle medlemmene bidro til å holde god fremdrift og slutføre prosjektet på en god måte, selv om dette kanskje ikke kom naturlig i rollen det enkelte medlemmet var identifisert i.

4.8 Testdokumentasjon

Både før, underveis og etter leveranse av programvare så er det viktig å teste. Det finnes mange forskjellige måter å dele opp tester på. Vi valgte å dele testene opp i enhets-, integrasjons- og systemtesting.

Vi kommer til å bruke begrepene "black-box", "white-box" og "grey-box". Black-box testing undersøker funksjonaliteten til et program uten å se på programmets interne funksjoner og strukturer. White-box testing tester derimot de interne funksjonene og strukturene. Grey-box testing tar hensyn til både funksjonaliteten og de interne strukturene til et program.

4.8.1 Enhetstesting

Enhetstesting tar for seg de minste og mest hyppige testene. De består av å teste individuelle komponentene i koden. Enhetstesting er en form for "white-box" testing. Det testes for eksempel enkelte metoder eller et sett med metoder.

Enhetstesting brukes for å teste at metodene kjører som forventet. Det gjør det også lettere å finne feil tidlig i systemet og fastsette hvilke komponenter feil oppstår i.

For en oversiktlig struktur er det best å lage enhetstester i egne filer. Da kan de kjøres på egenhånd. Testene følger [AAA-mønsteret](#) (Microsoft, 2019) og deles inn i tre ulike deler: "arrange", "act" og "assert". I "arrange" gjøres alt forarbeidet, som å opprette de nødvendige objektene⁴. I "act" kjøres metoden som skal testes. I "assert" sjekkes det om metoden oppførte seg slik som forventet.

C# og Visual Studio har automatisk integrert støtte for enhetstesting gjennom biblioteket "TestTools". I android og java utvikling må man bruke eksterne bibliotek som JUnit. Android Studios har automatisk integrert støtte for JUnit. På grunn av den integrerte støtten i utviklingsmiljøene ble det lett å utvikle og kjøre enhetstester. En annen måte det er gjort enhetstester på er ved å sende inn JSON-objekter⁹ til backenden gjennom Postman.

4.8.2 Integrasjonstesting

Etter enhetstesting er fullført, slås de individuelle komponentene sammen.

Integrasjonstestene sjekker om komponentene fungerer etter de er sammensatt. Selv om komponentene fungerer hver for seg, betyr det ikke nødvendigvis at de vil fungere kombinert.

Dersom kun backend eller frontend skal testes kan det tas i bruk lignende biblioteker som ved enhetstesting. Hvis integrasjonen mellom de to sidene skal testes, forberedes et skript som kommuniserer mellom frontend og backend.

4.8.3 Systemtesting

Systemtesting er typisk en black-box test. Den tester hele systemet for å se om det oppfyller funksjonelle krav. Systemet blir testet opp mot forventet oppførsel. Gruppen utførte systemtesting når ny funksjonalitet ble introdusert i programmet. Etter at programmet var ferdigstilt ble det utført en siste intern systemtest. Systemtesten var med på å avgjøre om kravspesifikasjonen var oppfylt.

Etter dette må det vanligvis utføres en akseptansetest. En akseptansetest gjennomføres av produkteieren og testere fra organisasjonen deres, for å se om produktet aksepteres. Dette kunne ikke gjennomføres på grunn av COVID-19 og mangel på ressurser. Det ble holdt en demonstrasjon av applikasjonen for produkteieren.

4.8.4 Testskript

Et testskript er en rekke instruksjoner for testere. Testskriptet blir brukt som en "instruksjonsmanual" for test-cases. Et test-case er et spesifikt sett med forhåndskrav, handlinger og resultater. Det ble utviklet et kort testskript for intern og fremtidig ekstern brukertesting. Testskriptet er laget for et test-case basert på brukerhistoriene nevnt i [punkt 3.2.1](#).

Test-skriptet er skrevet med et enkelt språk. Dette forsikrer at testpersonen kan lett forstå hvilke handlinger som må gjøres. For økt UX¹⁵ og potensielt bedre funksjonalitet bes det i dette tilfellet om en "score" for funksjonaliteten (Se [figur 4.6](#)).

I tillegg spørres det om potensielle forbedringer for de enkelte funksjonene. For eksempel: om noe ikke er intuitivt, noe kjører sakte eller at det mangler noe. Dette øker kompleksiteten på responsene, men gir til gjengjeld mer verdifulle tilbakemeldinger.

4.8.5 Gjennomførelse og intern brukertesting

På grunn av Covid-19 var det ikke mulig å utføre større brukertester for innsamling av tilbakemeldinger. Alternativet ble en intern test og en demonstrasjon av applikasjonen.

Den interne brukertesting ble gjennomført av gruppemedlemmene som testbrukere.

Testene ble utført på brukernes telefoner. Brukerne fikk utdelt et testskript som de fulgte. De skulle bruke 10 minutter på testskriptet, etterfulgt av 5 minutter med fri-aktivitet. I fri-aktivitet delen kunne brukerne gjøre det de ønsket på applikasjonen. Målet var å prøve å finne feil i applikasjonen.

Presentasjon av applikasjonen ble gjort over videosamtale på Microsoft Teams.

Hovedfunksjonene i applikasjonen ble demonstrert for produkteier og andre representanter fra RA. Blant dem var den ansvarlige for "finnerlønns"-ordningen.

"Test-script" for funnregistrerings applikasjonen			Score	Antall	Prosent
Denne er lagd for å kvantifisere og få en oversikt over bra funksjonaliteten til applikasjonen er. Scoren er målt fra 1-5 hvor 1 er best og representerer at funksjonen opererer nøyaktig slik man ønsker det			1	0	#DIV/0!
			2	0	#DIV/0!
			3	0	#DIV/0!
			4	0	#DIV/0!
			5	0	#DIV/0!
Test	Forventet resultat	Beskriv resultat	Score	Potensiell forbedring	
Åpne applikasjonen	Applikasjonen åpner fort og viser en introduksjonstekst				
Apne registreringssiden - tast inn tilfeldige "feil" opplysninger	Applikasjonen stopper deg fra å registrere				
Fyll inn "riktige" opplysninger denne gangen.	Applikasjonen registrerer brukeren din				
Trykk "Logg inn" og fyll inn riktig brukernavn og passord	Applikasjonen slipper deg inn på brukeren din				
Trykk på "Registrer funn"	Registrer funn siden åpner seg fort				
Legg til bilde og ta et bilde	Det fungerer som ønsket og bildet blir tatt				
Legg til GPS koordinat, dobbelt sjekk på google maps om det er nøyaktig	Koordinatene er riktige intil nærmeste meter				
Fyll ut resten og trykk på "Lagre"	Det initiale funnet blir lagret				
Send deretter inn funnmeldingen	Funnmeldingen kan bli sendt på epost.				
Oppdater funnet med der det mangler data.	Funnet blir oppdatert og man kan sende inn funnskjemaet.				
Send funnskjema	PDF blir generert og mail appen åpnes.				
Generelle inntrykk	Hva synes du om applikasjonen? Hvor har vi store mangler?				

Figur 4.6: Testskriptet.

4.8.6 Resultat

Resultatet fra den praktiske interne testen var at hovedfunksjonene utføres som de skal. Det var likevel noen mangler når det kom til design og brukervennlighet. Blant disse var lite intuitive ikoner og for mange menyvalg.

Noen forslag for funksjonalitet som var ønsket var som definert i [tabell 3.4](#). Disse var det ikke tid til å implementere.

Score	Antall	Prosent
1	50	83%
2	7	12%
3	3	5%
4	0	0%
5	0	0%

Figur 4.7: Resultat fra intern testing. 1 er høyest uttelling

Under demonstrasjonen ble arbeidet av applikasjonen presentert. I tillegg uttrykte gruppen sin vurdering om hva som eventuelt kunne forbedres av RA i fremtiden. Produkteieren og de andre på møtet var positive til produktet. De pekte ut at applikasjonen ble utviklet i løpet av en kort periode, men at den likevel løste problemstillingene på en god måte. Et nytt møte ble avtalt for en ny demonstrasjon av produktet 1. juni. Denne gangen foran Seksjon for Digitale Tjenester og organisasjonsavdelingen hos Riksantikvaren.

4.8.7 Forbedringer

I resultatene av testingen ble det foreslått ulike ting som kunne endres på applikasjonen. Etter tilbakemeldinger om meny-ikonenes manglende mening ble de byttet ut. Erstatningene er intuitive universalt. Et “+”-tegn er for eksempel globalt forstått som et symbol for tilføyelse.

Antall elementer i menyen ble også kritisert. Valgene var mange, noe som kan desorientere en bruker. Antallet ble redusert til tre menyknapper.

Valgene var kritiske for å betydelig øke brukervennligheten i applikasjonen.

4.10 Kort teknisk evaluering

Applikasjonen løser hovedproblemet til Riksantikvaren. Den gjør det lettere å sende inn funnskjema, samtidig som den reduserer antall feilkilder. Vår tekniske evaluering er derfor positiv.

Applikasjonen har blant annet automatisk innhenting av GPS-koordinatene. Innhentingens reduserer feil som kan oppstå ved manuell inntasting av brukeren. I tillegg vil

inputvalidering²⁴ redusere feilkilder ved at brukeren ikke kan skrive inn ugyldige tegn. Dette styrker også sikkerheten til applikasjonen. At applikasjonen automatisk genererer en PDF bidrar til et lettleselig digitalt funnskjema, framfor håndskrevne skjema. Lagring av brukernes data gjør at en bruker slipper å fylle inn sin informasjon hver gang et funnskjema skal genereres.

5. Konklusjon

Det ble i løpet av prosjektperioden utviklet en prototype-applikasjon som oppfyller kravene kunden ønsket. Under presentasjon for produkteier og interessenter, har applikasjonen allerede vist at den fungerer godt som en demo på hvordan funnregistreringsprosessen kan digitaliseres.

Vi i bachelorgruppen mener selv vi har nådd hovedmålet i oppgaven, knyttet til innhenting av funnskjema informasjon og sending av funnskjema og funnmelding. Vi har implementert minimumskravene definert i kravspesifikasjonen. I tillegg ble det implementert krav utover minimumskravene. Gruppen er veldig fornøyd med dette sett i tidsrammen arbeidet foregikk.

Det har gjennom hele prosjektet fokuseret på å ta i bruk gode utviklingsmetoder og verktøy. Effekten av dette er at vi har jobbet metodisk og målrettet, noe som kjennetegner en god systemutviklingsprosess. Veileder hos Riksantikvaren har i tilbakemeldinger gitt uttrykk for at hun er svært fornøyd, og et av gruppemedlemmene har fått jobbtilbud hos Riksantikvaren.

Vi møtte på enkelte tekniske utfordringer under utviklingen. Å løse disse var en tidkrevende, men lærerik prosess. Om noe skulle vært gjort annerledes er tilgang på en teknisk veileder noe som ville spart oss mye tid.

I denne rapporten har vi fokuseret på å gi et innblikk i vår utviklingsprosess. Vi mener rapporten beskriver systemet godt, og at beskrivelsen kan brukes til videreutvikling. Vi mener også at rapporten dokumenterer en god planleggingsfase og gode arbeidsmetoder.

Kort oppsummert vil vi si at prosjektet har vært en spennende og lærerik prosess. En prosess hvor vi har tilegnet oss mye god erfaring som vil komme godt med videre i yrkeslivet.

6. Vitenskapelig litteraturliste

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D. (2001). *Manifesto for Agile Software Development*.
<https://agilemanifesto.org/principles.html>

Entity Framework (No date) *EF Code First*. Entity Framework 6. Hentet 21. mai 2021.
<https://entityframework.net/ef-code-first>

Gundersen, J., Lie, R. O, Rasmussen, J. M. Private Metal Detecting and Archaeology in Norway. (2016) *Open Archaeology*, 2(1), s. 160-170. <https://doi.org/10.1515/opar-2016-0012>

Kristoffersen, B. (2016). *Databasesystemer* (4. utg.). Universitetsforlaget.

Microsoft (2021, 19.mai). *Beskrivelse av grunnleggende om databasenormalisering*. Microsoft Docs. Hentet 23. mai 2021.
<https://docs.microsoft.com/nb-no/office/troubleshoot/access/database-normalization-description>

Microsoft. (2021, 7. april). *Byte Struct*. Microsoft Docs. Hentet 7. mai 2021.
<https://docs.microsoft.com/en-us/dotnet/api/system.byte>

Microsoft. (2019, 8. juli). *Unit test basics*. Microsoft Docs. Hentet 20. mai 2021.
<https://docs.microsoft.com/en-us/visualstudio/test/unit-test-basics?view=vs-2019>

Olsen, T. (2021, 5. mai). *Hva er Microsoft Azure?* Basefarm. Hentet 10. mai, 2021 fra
<https://basefarm.no/blogg/hva-er-microsoft-azure/>

Oracle. (2020, 20. august). *What is a Relational Database?* Hentet 10. mai, 2021 fra
<https://www.oracle.com/database/what-is-a-relational-database/>

Voldsgård, K. H., Skjølsvik, T., Braglien, J. J. (2020). *Forretningsforståelse* (2. utg.). Cappelen Damm Akademisk.

Wessman, A., Thomas, S., Rohiola, V., Kuitunen, J., Ikkala, E., Tuominen, J., Koho, M., & Hyvönen, E. (2019). A Citizen Science Approach to Archaeology: Finnish Archaeological Finds Recording Linked Open Database (SuALT). *Digital Humanities in the Nordic Countries*.
<https://www.researchgate.net/publication/335928893>

7. Vedlegg

Vedlegg 1 Ordliste

	Ord	Betydning		Ord	Betydning
1	Proof of Concept	En demonstrasjon av en ide for å vise at det har praktisk potensiale.	2	Normalform	Viser til en serie med teknikker som er lagd for å minimere duplikasjoner og sikre god "dataintegritet".
3	Pseudokode	Kode omskrevet på en lettleselig måte, så det kan forstås av personer som ikke er utviklere.	4	Objekt	En instans av en klasse, med klassens tilhørende data.
5	Base64-streng	En variabel som oversetter binære data til tekstformat, som for eksempel et bilde.	6	Code-first database design	Code-first handler om å designe en database ved å definere den i kode.
7	RegEx	Står for "Regular Expression". RegEx er et mønster som beskriver regler til en syntaks.	8	Salt	Et salt er tilfeldig tekst som tilføyes et passord når det lagres i databasen.
9	JSON	JavaScript Object Notation - En måte å formtere data på. Brukes som regel til overføring av data.	10	GET-metoder	Metode for å hente verdien til et felt i en klasse
11	URL	Uniform Resource Locator - En URL er en web-lenke.	12	SET-metoder	Metode for å sette verdien til et felt i en klasse
13	HTTP	Hypertext Transfer Protocol. Protokoll for å	14	HTTP GET request	HTTP GET brukes til å forespørre data

		overføre data mellom applikasjonen og serveren.			fra serveren.
15	UX	User Experience eller Brukeropplevelse. Et begrep som handler opplevelsen av å bruke et produkt. God UX betyr isåfall en god opplevelse.	16	HTTP POST request	HTTP POST brukes til å sende data til serveren.
17	Request-body / JSON-body	Er data-en som sendes i en forespørsel. Dette skiller seg fra parametrene til forespørselen, f.eks. ved at de ikke skrives i URL-en.	18	Java Map	En liste som inneholder par av nøkler og verdier.
19	LINQ	Et "database-query" språk som er integrert inn i et programmeringsspråk. I dette tilfellet C#.	20	Shared Preferences	Klasse for å lagre variabler lokalt på telefonen.
21	Toast	En toast er en Android klasse for å vise en melding på skjermen.	22	Email-intent	En intent lar en Android applikasjon sende en oppgave til en annen applikasjon. I tilfelle med email-intent vil dette si, bruke en email app til å sende en email.
23	Androids graphics	Tilbyr grafiske lav nivås verktøy, som lar deg håndtere tegning direkte på skjermen.	24	Inputvalidering	Å sjekke felter for ugyldige verdier.

25	Riksantikvarens Nettprofil	Riksantikvarens Nettprofilen er et design system for Riksantikvarens digitale flater.	26	SQL-injeksjonsan grep	SQL-injeksjonansa ngrep består av innsetting av SQL-spørring via inngangsdataene fra klienten til applikasjonen.
----	-------------------------------	---	----	--------------------------	--

Tabell 7.1: Ordliste.

Vedlegg 2 Kravspesifikasjon

Presentasjon

Gruppe 54 består av Fatima, Tor, Helge, Benjamin og Adrian og har akseptert en oppgave fra Riksantikvaren for å utvikle en tjeneste for å registrere løse funn oppdaget av "detektorister". Vi har blitt tildelt veilederen **Lothar Fritsch** på OsloMet som kontaktes gjennom mailen hans: Lothar.Fritsch@oslomet.no.

Kontaktpersonen vår i Riksantikvaren er **Maria Sølversen** og kontaktes gjennom mailen hennes på: maria.solversen@ra.no.

Om bakgrunnen

Vi skal lage en prototype av en Android app for å registrere oppdagede løse funn, hvor primærinteressenten er metallsøkere, eller "detektorister".

Ideen for oppgaven har vi fått fra Riksantikvaren med Maria Sølversen som kontaktperson. Applikasjonen lages for å effektivisere innmelding og registrering av løse funn. Lovverket sier at funnet må meldes inn innen 48 timer. Dersom fylkesarkeologen ønsker at funnet skal sendes inn vil det være nødvendig å sende inn funnskjema. Applikasjonen skal lagre informasjon som trengs for å registrere funnet, bl.a. GPS koordinater, bilde og beskrivelse av funn og funnområdet. Applikasjonen skal fungere som et Proof of Concept¹ for Riksantikvaren.

Forord

Hensikten med applikasjonen er å bidra til å løse utfordringer for detektorister og fylkesarkeologer knyttet til innmelding og innsending av funn. For detektoristene vil applikasjonen gjøre det lettere å melde inn funn og for arkæologene vil den gjøre det lettere å behandle funn. Dette skal gjøres i henhold til begrensninger som: en tidsfrist for innlevering

av oppgaven, kompetansen til de ulike medlemmene i gruppen og mangel på muligheten til å opprettholde/oppdatere applikasjonen etter endt prosjektperiode.

I hovedsak skal applikasjonen være et verktøy for detektoristene. Når de er ute i felten og finner noe av interesse, er det optimalt at prosessen med å registrere funnet starter med en gang. Å gjøre det så enkelt og lavterskel som mulig å ta bildet og registrere lokasjonen på funnet vil være avgjørende for å sikre mer nøyaktig funnadata, og dette er noe applikasjonen kan løse.

Utfylling av funnskjema er også en utfordring for mange detektorister. Funnskjema må fylles ut riktig for å i det hele tatt ha mulighet til å få finnerlønn. Finnerlønnen deles mellom finner og grunneier. Nøyaktigheten av koordinater har vist seg å være en utfordring, i tillegg har funnskjemaet flere 'bokser' som *ikke* skal fylles ut av detektoristen. I applikasjonen vil vi løse disse utfordringene ved at store deler av funnskjemaet blir fylt ut automatisk, og detektoristen vil kun få mulighet til å fylle ut det de skal fylle ut (ikke noe mer).

Om man er en fersk detektorist har det vist seg at det kan være utfordrende å vite hvem man skal kontakte om funn. Dette er noe applikasjonen løser ved å sende melding om funn til riktig fylkesarkeolog.

En annen utfordring for ferske detektorister er å vite hvor det er lov å lete, vanlig fremgangsmåte, skikk og bruk i miljøet og hvilke lover og regler som gjelder.

Mye av dette kan applikasjonen hjelpe til med. Små tekstbokser, med tips og linker til forum og lovdata hvor man kan lese seg opp mer, vil hjelpe de ferske detektoristene til å se hvilke ansvar de har og også hvordan de går frem for å sikre seg finnerlønn.

For fylkesarkeologen (og Riksantikvaren) vil applikasjonen bidra til bedre og mer nøyaktig informasjonsinnsamling. For dem er det veldig viktig at all informasjon er fylt ut riktig, spesielt informasjon knyttet til lokasjonen på funnstedet. Dette løses som nevnt i applikasjonen ved at brukeren i applikasjonen får veldig mye hjelp til utfyllingen av informasjonen som trengs på funnskjemaet.

På sikt kan det være ønskelig å lage egen funksjonalitet for arkeologene, slik at behandling av søknadene kan skje gjennom applikasjonen. På denne måten kan all kommunikasjon mellom detektorist og arkeolog finne sted der.

For å oppsummere vil applikasjonen være et verktøy for detektorister som hjelper dem å fylle ut funnskjema korrekt og opprette kontakt med arkeologene. I tillegg vil applikasjonen gi hint til ferske detektorister om fremgangsmåte, lover og regler. Applikasjonen vil bidra til å

gjøre fylkesarkeologens jobb enklere ved at det blir en standardisert digital løsning hvor mange mulige feil i utfylling er eliminert.

Leserveiledning

Ettersom prosjektet fungerer som et Proof of Concept¹ samtidig som det blir et reelt produkt, har vi bestemt oss for å strukturere kravspesifikasjonen i minimumskrav og andre ønskede egenskaper. Minimumskravet er de funksjonene vi mener er essensielle for et brukbart sluttprodukt, mens andre ønskede egenskaper er funksjoner vi ønsker å legge til om vi har tid, men som ikke er kritiske for et brukbart sluttprodukt.

Kort systembeskrivelse

Systembeskrivelsen inneholder de gevinster vi venter å oppnå, systemets funksjonelle egenskaper beskrevet med diagrammer og prosess-spesifikasjoner. Vi inkluderer også en rekke egenskaper som vi hadde sett for oss kunne gitt stort utbytte, men som mulig må ekskluderes på grunn av omstendighetene, som den begrensed utviklingstiden og applikasjonens mangel på vedlikehold.

Brukerhistorier

En brukerhistorie er uformell forklaring av en funksjon. De er skrevet fra brukerens perspektiv.

Vi valgte å skrive brukerhistorier for å få oversikt over hvordan planlagt funksjonaliteten vil gi brukeren verdi.

*“Som en detektorist,
ønsker jeg å registrere og lagre løse funn i applikasjonen,
slik at jeg har den informasjonen jeg trenger når jeg skal melde funn eller sende
funnsekjema.”*

*“Som en detektorist,
ønsker jeg å melde inn funn i applikasjonen,
slik at jeg lettere kan gi beskjed til riktig person om at jeg har gjort et funn.”*

*“Som en detektorist,
ønsker jeg at applikasjonen husker informasjon om meg,*

slik at det blir mindre å fylle ut i funnsekjemaet senere.”

*“Som en detektorist,
ønsker jeg å kunne se oversikt over mine tidligere funn,
slik at jeg enkelt kan se hva jeg har funnet før og ha en liste over min samling.”*

*“Som en detektorist,
ønsker jeg å kunne sende inn funnsekjema gjennom applikasjonen,
slik at jeg sparer tid, informasjonen er korrekt og at applikasjonen kan fylle ut store deler av
feltene automatisk.”*

*“Som en detektorist, ønsker jeg å fort kunne registrere et oppdaget funn, slik at ingen
opplysninger går fortapt i korttidsminnet og jeg kan fort få tilbakemelding fra
fylkesarkeologen.”*

*“Som en fylkesarkeolog,
ønsker jeg å ha henvendelser om funn på ett sted,
slik at det blir lettare for meg å behandle henvendelsene.”*

*“Som en fylkesarkeolog,
ønsker jeg en app som hjelper brukerne med å fylle ut funnsekjemaet riktig, slik at jeg ikke
trenger å spørre om tilleggsinformasjon senere.”*

Tabell 7.2: Brukerhistorier

Brukerhistorier går ofte hånd-i-hånd med personas. Personas er abstrakte karakterer som representerer en sluttbruker eller en gruppe man ønsker å nå ut til. Man bruker disse karakterene for å legge til rette for disse gruppene. Personas brukes både i utvikling og i testing for å sørge for at man har gjort en god jobb i å utvikle en løsning for de ulike gruppene. Personasene vi har brukt for dette utviklingsprosjektet er:

Rolle	Beskrivelse
Detektorist	En fersk detektorist som startet sammen med en av kompisene hans. Er i 40-årene og jobber som mekaniker, men har alltid hatt en interesse for historie. Drar aldri ut med metalldetektoren uten vennene hans som er erfarte detektorister som kan lovene. Relativt kompetent med IT.
Detektorist	En erfaren detektorist som er styreleder i en av

	metalldetektor-lagene. Er i tidlig-50 årene og jobber i oljesektoren. Drar gjerne ut alene, i små grupper eller med en stor del av det lokale metalldetektor-laget. Ikke spesielt teknisk, men kan lover og forskrifter godt. Synshemmelse, fargeblindhet og svaksynt.
Detektorist	En tidligere detektorist som har startet nå nylig igjen. Er ikke spesielt kjent med nye forskrifter og lover, men er veldig ivrig. Har ikke blitt med lokallaget ennå og drar ut gjerne med en liten gruppe venner. Greit teknisk, klarer å bruke en mobil og vet hvordan man bruker applikasjoner. Dårlig dekning der han bor.
Arkeolog	En erfaren arkeolog som har jobbet i fylket i 15 år. Har ikke veldig tett kontakt med de ulike detektorist lagene, men har jobbet med behandling av funnsvkjema i mange år.
Arkeolog	En ny arkeolog som har nettopp startet med å behandle funnsvkjemaer for funn i fylket hans. Startet arbeidet sitt med å ta kontakt med detektorist lagene for å opprette et samarbeid. Tar gjerne å leser innmeldinger som blir sendt til han gjennom epost i helgene.

Tabell 7.3: Personas

Funksjonelle krav: Minimumskrav

Det ble utarbeidet en liste over funksjonelle krav. I programutvikling er funksjonelle krav definisjonen av en funksjon eller komponent. Tabellen under inneholder de funksjonelle kravene vi har valgt å prioritere. Dette er krav som må implementeres. Verdien fra implementeringene må relaiseres for ønsket sluttprodukt. Vi har valgt å definere tabellen som minimumskrav.

Lang	2-8 uker
Medium	1 uke
Kort	1-3 dager

Tabell 7.3: Fargekoding av prosesstid som brukes i tabellene nedenfor

Funksjon	Beskrivelse	Verdi	Prioritet	Tid
Registrere løse funn	Utfylling av viktig info om funnet. Tar bilde og GPS-koordinater med tastetrykk.	Utfylling av innmelding og funnsvkjema blir korrekt. <i>F.eks GPS koordinatene kan ikke tastes feil inn.</i>	Høy	Kort

Innmelding av funn (Epost)	Innsending av funnmelding ved et tastetrykk. E-post fyller seg ut med nødvendig info og sendes riktig fylkesarkeolog.	Fylkesarkeologen får informasjonen riktig utfylt og i et godt format.	Høy	Kort
Logg inn	Registrering og innlogging. Funn knyttes til bruker. Lagres i skyen. Brukerinfo brukes i automatisk skjemautfylling.	- Tidsbesparende og kvalitetssikrende (ved skjemautfylling). - Tilgang på funnene i funnliste. - Tilgang fra andre enheter.	Høy	Medium
Oversikt over egne funn	Oversikt i form av en liste over brukerens tidligere funn. Hvert enkelt funn kan redigeres(Fylle ut mer info og endre info.)	- Brukeren får en oversikt over tidligere funn. - Brukeren slipper å legge inn all info ved funnregistrering. (Brukeren er ofte ute i felten. Det kan det være utfordrende å fylle ut mye informasjon.)	Høy	Kort
Sende funnskjema	Automatisk utfylling av funnSkjemaPDF. Sender PDF via epost til fylkesarkeologen.	- Fylkesarkeologen får tilsendt skjema digitalt. - Minimerer feil ved skjemautfyllingen.	Høy	Medium

Tabell 7.4: Oversikt over minimumskravene

Krav utover minimumskravene

Tabellen under inneholder funksjonelle krav utover kravene nødvendige for ønsket sluttprodukt. Implementasjon av kravene avhenger av at minimumskravene allerede er implementert og av hvor mye tid som gjenstår til leveranse. For Riksantikvaren vil funksjonene fra tabellen kunne brukes som inspirasjon utover funksjonalitet som blir presentert i prototypen.

Funksjon	Beskrivelse	Prioritet	Tid
Kvalitetssikre informasjonen på funnSkjema	Videre automatisering av utfylling. Brukeren fyller kun ut felter som er strengt nødvendig å fylle ut manuelt. FunnSkjema blir kun sendt om det er riktig utfylt.	Middels	Lang
Brukerveiledning i applikasjonen.	Går i gjennom applikasjonen side for side og beskriver funksjonaliteten.	Middels	Medium - Kort
Manuel GPS insetting	Innskrivning av GPS-koordinater eller valg av lokasjon på kart i applikasjonen.	Middels	Kort - Medium

Info lover og regler-side	Informasjonsside med opplysninger om lover, regler og linker til ressurser.	Middels	Kort - Medium
Automatisk oppdagelse av Gårds/Bruks nr.	Automatisk innhenting og utfylling av gårds- og bruksnr. (Hentet ved bruk av GPS-koordinatene knyttet til funnet.)	Lav	Ukjent - Lang
Bytte språk	Muligheter for å velge andre språk i applikasjonen (engelsk, polsk, etc)	Lav	Kort
Panoramabilde av terrenget	Ta bildeserie og opprette panoramabilde av funnsted.	Lav	Kort - Medium
Kommunikasjon i app (svar på app)	Svar på funnmelding og funnskjema i applikasjonen.	Lav	Lang
Feilmelding ved søk på fredet område	Sjekke om området er fredet. Returnerer melding til bruker om området er fredet.	Lav	Lang
Presentere funn VIA webservice	Enkel webside som kan vise oversikt over egne funn.	Lav	Medium
Generering av Funnummer	Genererer enkel kode og knytter den til funnet. <i>Hensikt: Koden kan skrives på funnposen for å lettere koble funnskjema med det innsendte funnet.</i>	Lav	Kort

Tabell 7.5: Oversikt over andre ønskede egenskaper

Utfordringer

- Datavern - Det er potensielt personopplysninger lagret i denne tjenesten, må være kryptert.
- Sikker overføring/kommunikasjon - dersom man har en form for kommunikasjon så må opplysningene sendes sikkert. Kan en generisk epostserver være sikker nok?

Forventede gevinstar

Løsningen forventes å minske tiden det tar å registrere et funn, fra det går fra finnen til fylkesarkologen. Det er ventet av applikasjonen skal gjøre innmelding av et funn lettere ved hjelp automatisering der det er mulig når det kommer til utfylling av skjema.

Automatiseringen vil også føre til mindre feil ved skjemautfylling, som igjen vil gjøre hele prosessen raskere ved at fylkesarkologen ikke har feil eller manglende data. Det vil også spare den tiden det tar å fylle ut skjema manuelt.

En brukervennlig og intuitiv applikasjon vil gjøre terskelen til å melde inn funn lavere og dermed motivasjonen til å melde dem inn høyere. Hvis prosessen med å utbetale finnerlønn også blir raskere er det forventet at antall funn som ikke meldes inn vil bli mindre.

Ved å implementere en opplæring eller “tutorial” i systemer, er det forventet at det skal bidra til å minske tapte funn ved å informere en nybegynner om lover og regler som en metallsøker er forventet å følge, både angående innmeldingsplikt innen en tidsfrist og hvordan man skal behandle et funn for å ikke skade det. Ved å informere flere om reglene rundt metallsøking vil man i tillegg kunne minimere ulovlig metallsøking. En god opplæringsprosess i applikasjonen vil også føre til at flere funn blir meldt inn med korrekt informasjon.

En lavterskel applikasjon som inkluderer opplæring for nye brukere og gjør innmelding enkelt, forventes å gjøre innmelding av funn mer nøyaktig og korrekt i henhold til lovverket, samtidig som det skal minske antall tapte funn.

Rammekrav i systemet

Sikring mot tap, ødeleggelse, tyveri og misbruk av data

Dataene skal i første omgang lagres lokalt på brukerens telefon, da er det ikke mye vi kan gjøre for å sikre mot tap eller ødeleggelser av dataen. Den lokale dataen kan eventuelt krypteres for å hindre at dataene ikke kan stjeles eller misbrukes.

Vi ønsker i utvidet spec å legge til muligheter for lagring av data på en ekstern database. Da kan vi ha backup på flere databaser og brukere kan få tilbake tapt data med å synkronisere mot skytjenestene. På databasene er det fokus på sikkerhet og dataene skal lagres i kryptert form, og kun være tilgjengelige for autentiserte brukere.

Kapasitet

Metallsøking er i hovedsak en fritidsaktivitet, og basert på det vi fant ut av i et intervju med detektorister og arkeologer, sammen med Sølversen, så er de fleste aktive i helger og ferier. Systemet må ha kapasitet til å kunne betjene alle brukerne som er aktive da.

En foreslått løsning på dette kan være å ta i bruk en skytjeneste, ettersom de har muligheten til å skru opp og ned på kapasitet ved behov. Andre fordeler med skytjenester er at da slipper man å tenke på å hoste noe selv, man slipper innkjøp av hardware, og de har allerede masse sikkerhet implementert.

Fremtidig utvidelse av systemet

Systemet skal fungere som en prototype for en app som Riksantikvaren selv skal utvikle senere. Det er derfor viktig å legge til rette for at Riksantikvaren kan fortsette arbeidet med applikasjonen på et senere tidspunkt.

Vi har mange funksjoner som vi vil utvikle i applikasjonen, men vi får ikke tid til å legge til alle i dette prosjektet det er derfor viktig at vi fokuserer på “Clean code” og god dokumentasjon så Riksantikvaren senere kan legge til disse funksjonene om de ønsker.

Brukervennlighet og universelt design

Systemet må være intuitiv og lett å bruke. Det skal også ta få tastetrykk for å komme fram til noe. De mest brukte funksjonene skal være lettest tilgjengelig (helst via ett tastetrykk). Mindre brukte funksjoner (for eksempel slette konto) skal være vanskeligere å komme fram til for å unngå feiltasting.

Som enhver god løsning bør systemet ta hensyn til personer med nedsatt syn. Dette kan gjøres ved hjelp av designvalg som: kontraster, gode fargevalg, stor skrift og et lestbar font. Disse valgene vil gjøres brukergrensesnittet mer brukervennlig for alle. Det er også viktig at vi velger farger med god kontrast i applikasjonen, blant annet fordi applikasjonen forventes å bli brukt utendørs i sterkt sollys.

Bruk av ikoner, tekst og farger for å veilede brukeren hjelper med å unngå feiltasting og forsikre at brukeren forstår hvilke funksjonaliteter som tilbys i applikasjonen. Ikoner og bilder skal være relevante til den informasjonen de hinter til (for eksempel en konvolutt for e-post) og helst relevante i den konteksten de skal brukes i (for metallsøking i norske miljøer).

Informasjonen i applikasjonen skal være lett forståelig og ta hensyn til at personer har ulike mengder erfaring med metallsøking. Dette kan gjøres ved at det for eksempel i

opplæringsdelen gis informasjon som forventer at en person har ingen kunnskap, men at en bruker likevel har mulighet til å hoppe over opplæringen om de har mer kunnskap.

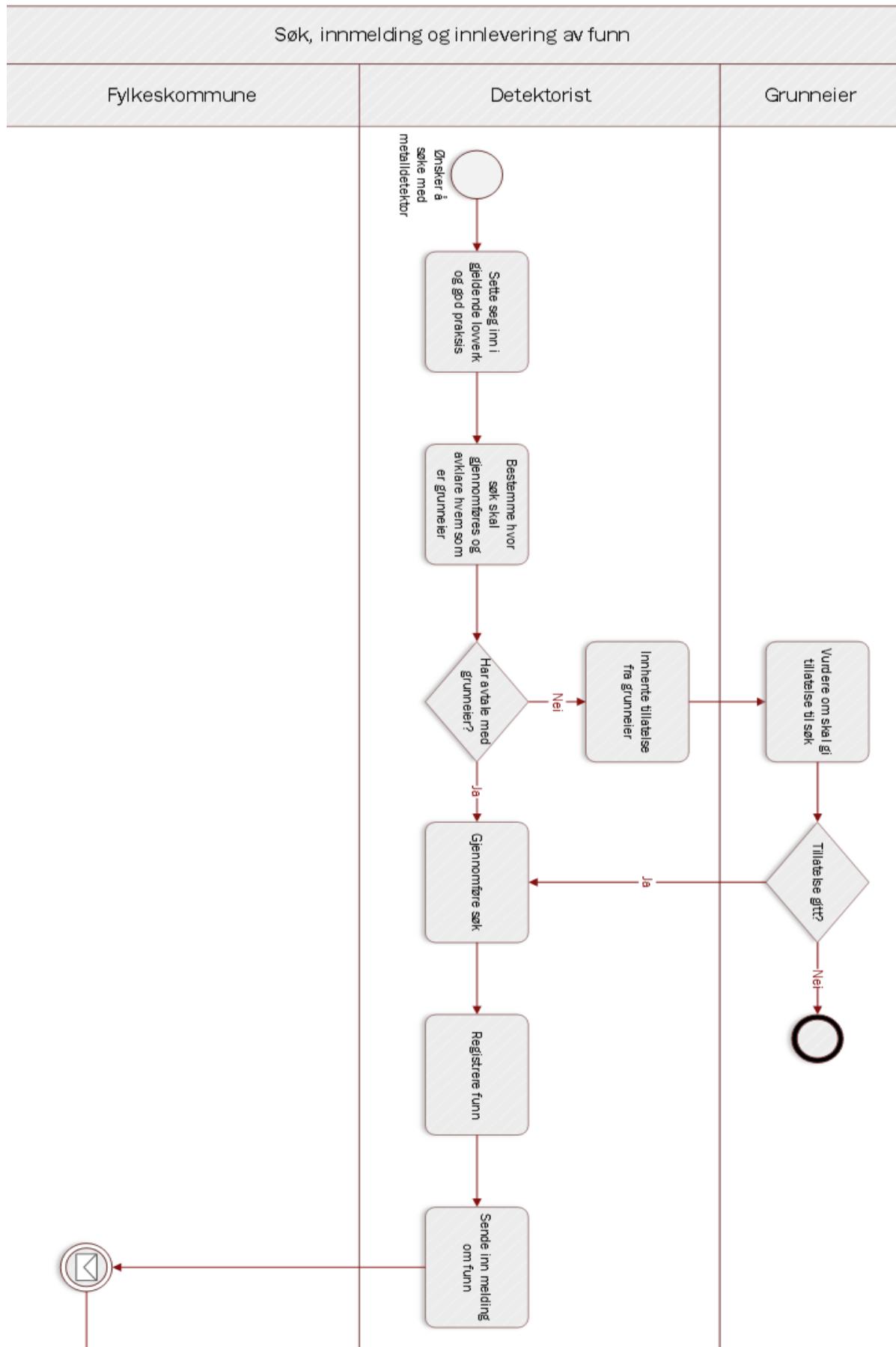
Feilmeldinger skal være korte og enkle å forstå, og lette å komme seg ut av, for eksempel ved hjelp av en "Ok"-knapp. Det er også gunstig å minimere feil i applikasjonen som fører til at den henger seg opp (der løsningen er å restarte applikasjonen). Hvis det blir tilfelle, er det viktig at brukeren ikke mister viktig informasjon og at det er enkelt å begynne på nytt.

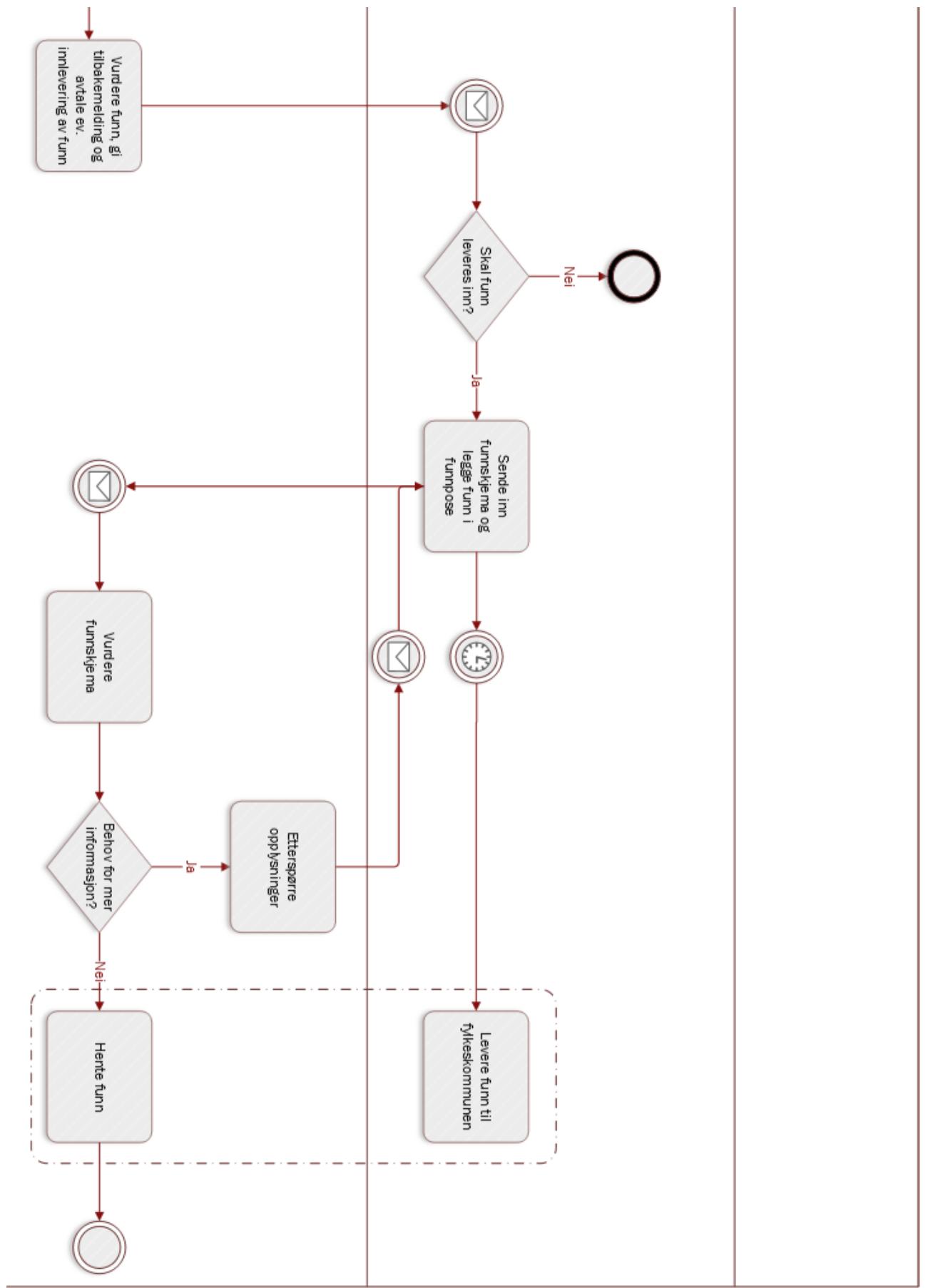
Designet av applikasjonen skal helst være slik at en bruker skal kunne navigere seg rundt og bruke applikasjonen med én hånd.

Logiske datamodeller

Overordnet prosessdiagram

I dette diagrammet beskrives finnerlønn prosessen fram til funnet er levert. Dette er hvordan prosessen fungerer i dag og er hva vi baserer applikasjonen vår på.



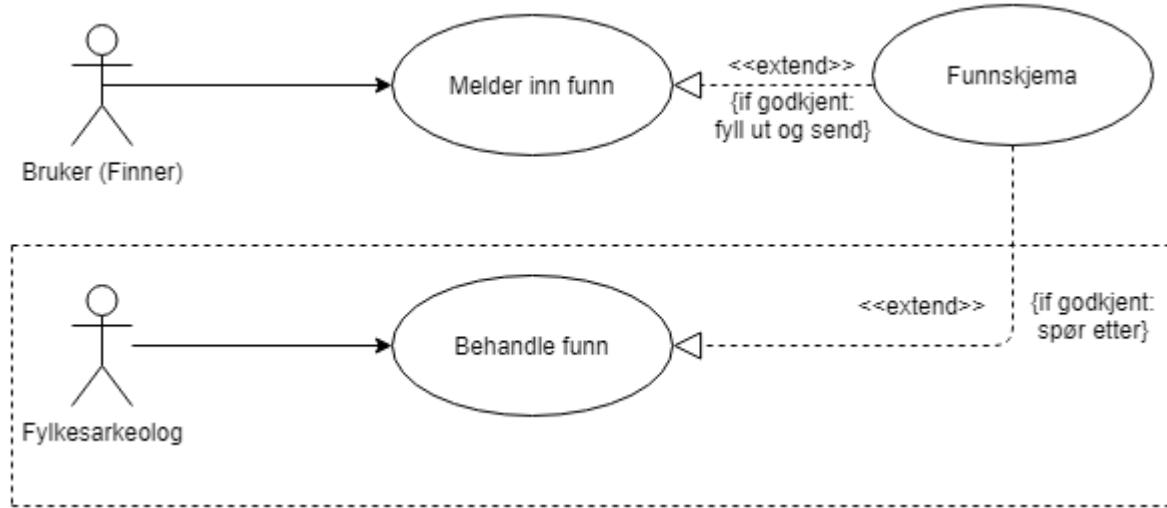


Figur 7.1: Overordnet prosessdiagram av innlevering av funn

Brukerinteraksjon

Her beskrives hvordan to av aktørene, bruker og Fylkesarkeolog, medvirker i innmeldingen av funn. Brukerens handlinger, å melde inn funn og sende inn funnskjema, er hovedmålene i vår applikasjon.

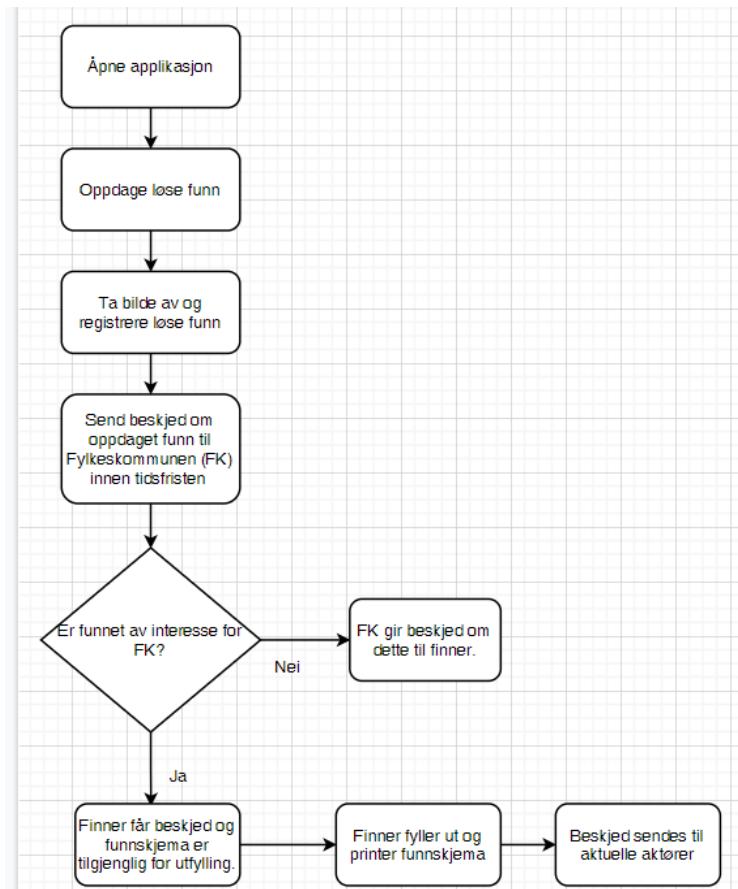
(Hoved use-case diagram)



Figur 7.2: Brukerinteraksjon (Hoved use-case diagram)

Flyt i programmet

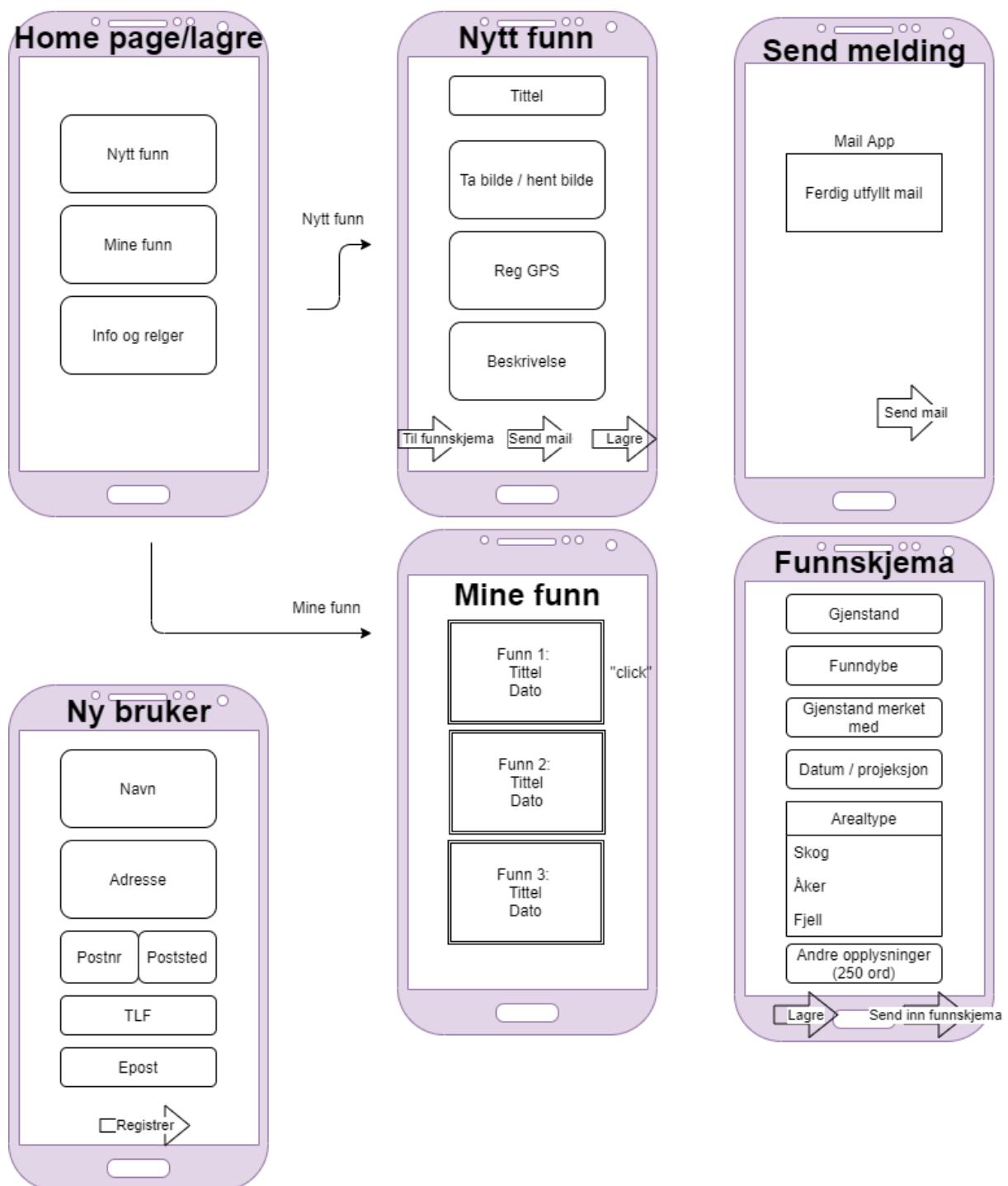
Flytdiagrammet viser oss hovedflyten i programmet. Det viser i grove trekk funksjonene applikasjonen har fra oppstart.



Figur 7.3: Flytdiagram for hovedflyten i programmet.

Wireframe:

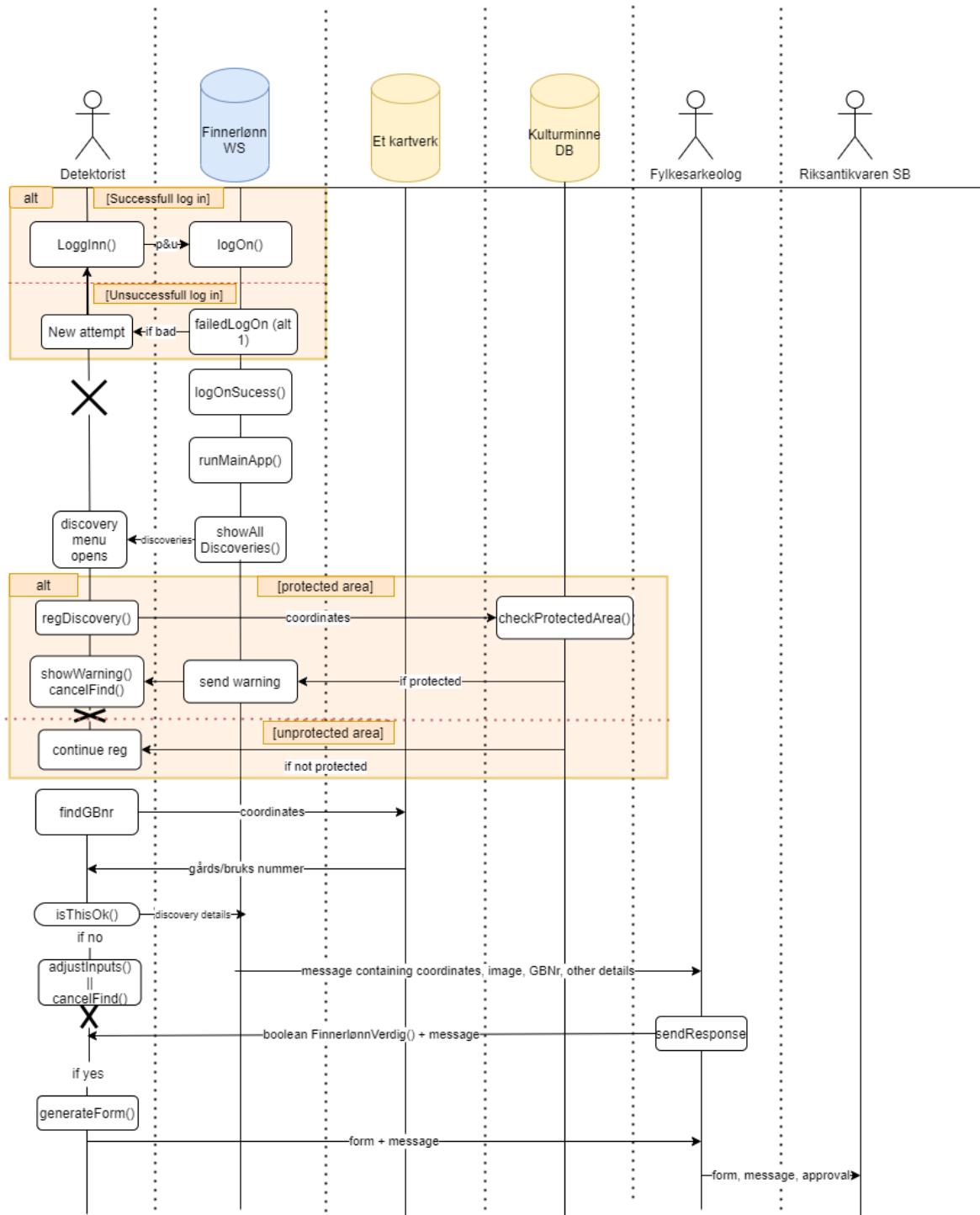
Wireframe-en viser en grov skisse på funksjonalitet.



Figur 7.4: En grov skisse av wireframe i programmet.

Interaksjoner/funksjonskall

Sekvensdiagrammet viser i mer detaljer grad hvordan programmet skal fungere. Her setter man opp hvilke funksjonskall koden skal gjøre. Denne har med funksjoner som er utenfor minimumskravene, men som vi forventer/håper å legge til i prosjektet.



Figur 7.5: Sekvensdiagram som viser funksjonskall.

Markedsanalyse

Vi har funnet inspirasjon og ideer fra lignende prosjekt i utlandet, for eksempel Danmark sin løsning "DIME" og Storbritannia sine løsninger "NCMD" og "UKDFD".

Vi har også lest diverse rapporter på kulturminneforvaltning, detektorister og prosjekter som involverer dette, blant annet Finland sin løsning "SuALT". Andre rapporter inkluderer for eksempel en norsk rapport om arkeologi og privat metallsøking, og flere artikler fra BioSphera, et prosjekt om natur og kulturminneforvaltning fra diverse aktører fra Catalonia.

Vi har også fått høre om apper fra detektoristene selv, som iSmartDetect. Vi kan selvfølgelig ikke kopiere eller herme etter de andre løsningene, men noen av dem har noen gode ideer, som vi vil ta til bruk selv.

Liknende apper

Det er ingen digital løsning for funnregistrering i Norge. Det var likevel mulig å hente inspirasjon fra utenlandske applikasjoner. Detektoristene vi pratet med hadde selv tatt i bruk webapplikasjonen og webforumet DIME((<https://www.metaldetektorfund.dk/>) og en android applikasjon kalt iSmartDetect. I tillegg til å gjøre oss kjent med disse applikasjonene så vi på applikasjonene Minelab Treasure Tracking, Tect O Trak og onX Maps.

I vedlegget "Kravspesifikasjon" har [tabell 5](#) en grundigere gjennomgang av organisasjoner og applikasjoner .

Tabell som utforsker organisasjoner, apper og nettplattformer:

Organisasjon / Applikasjon	Beskrivelse
DIME (dansk platform) https://www.metaldetektorfund.dk/	<ul style="list-style-type: none">• Online platform for registrering av metalldetektorfunn.• Info om funnet:<ul style="list-style-type: none">- Beskrivelse, kategori- GPS-koordinater- Bilde av gjenstand• Andre brukere kan kommentere funn / hjelpe med identifisering av funn.
NCMD https://www.ncmd.co.uk/	<ul style="list-style-type: none">• Interesseorganisasjon for metalldetektorister• Informasjon om blant annet:<ul style="list-style-type: none">- Detektoristklubber

	<ul style="list-style-type: none"> - Treasure Act-loven - Diverse guider knyttet metalldetekting - Et online magasin hvor de publiserer info og nye funn.
UKDFD https://www.ukdfd.co.uk/	<ul style="list-style-type: none"> ● Registrering av funn ● Database med funn ● Forum (Info, kommunikasjon mellom brukere)
iSmartDetect https://play.google.com/store/apps/details?id=com.ngintermarketingaps.ismartdetect.android&hl=en&gl=US	<ul style="list-style-type: none"> ● Bildefunnsted ● Merke funn med: <ul style="list-style-type: none"> - GPS-koordinater (Nord/sør) - Tid /dato - Kategori ● Veipunktsporing ● Sikker arkivering (server side)
onX Maps https://www.onxmaps.com/	<ul style="list-style-type: none"> ● Kart som viser blant annet: <ul style="list-style-type: none"> - Tomtegrenser - Navn på grunneier - Naturreservater ● GPS ● Fotspor
Tect O Trak https://play.google.com/store/apps/details?id=com.trib.app.tectotrak&hl=en_US&gl=US	<ul style="list-style-type: none"> ● Manuell merking av leteområdet ● Logging av fotskritt på kart ● Funnregistrering: <ul style="list-style-type: none"> - Navn, beskrivelse - Bilde - Lokasjon (gps) ● Lokal lagring av funn
Minelab Treasure Tracking https://play.google.com/store/apps/details?id=com.minelabmetal&hl=en_US&gl=US	<ul style="list-style-type: none"> ● Funnregistrering: <ul style="list-style-type: none"> - Tittel, beskrivelse - Bilde - Lokasjon (gps) ● Bibliotek med egne funn

Tabell 7.6: En oversikt over andre lignende organisasjoner og applikasjoner.

Krav til dokumentasjon

Riksantikvaren skal bruke dette prosjektet som grunnlaget for senere utvikling av lignende applikasjoner, derfor er god dokumentasjon av prosjektet viktig. Med god dokumentasjon blir det lettere for Riksantikvaren å forstå hvorfor vi velger de løsningene vi velger, det vil også gjøre det lettere for dem å videreutvikle applikasjon senere om de ønsker det. En annen god grunn er for å gjøre det lettere for oss å utvikle et konsekvent og godt produkt, og samtidig kunne skrive en god rapport.

Vi skal dokumentere alle viktige valg vi gjør underveis i prosjektet, slik at det blir lett for Riksantikvaren å forstå de valgene vi har tatt. Dokumentasjonen er delt inn i to deler - styringsdokumentasjon og slutt-dokumentasjon. Styringsdokumentasjon er dokumenter som vi tar i bruk for å bidra til selve utviklingen, som dette dokumentet. Slutt-dokumentasjon er dokumentasjon vi lager for å oppsummere og slutføre prosjektet. Det kan være blant annet brukerdokumentasjon, testdokumentasjon eller prosessdokumentasjon.

Det blir viktig at vi skriver "Clean code" slik at det blir lett å forstå hvordan koden fungerer, og at koden enkelt kan videreutvikles senere. Vi vil i tillegg skrive utviklerdokumentasjon/teknisk dokumentasjon (f.eks. API-dokumentasjon) som et støttedokument for å bedre forstå koden.

Brukerdokumentasjon må også inkluderes, slik at brukere lett kan finne opplysninger om hvordan applikasjonen skal brukes. Målet er at dette skal integreres inn i applikasjonen i både "tutorial"-form og i dokumentform.

Arbeidsplan

Tabellen nedenfor tar for seg de viktigste delene av utviklingsprosessen og viser ønsket arbeidsplan i løpet av de neste månedene frem til prosjektslutt.

Det er forventet at vi inkluderer potensielle fremtidige brukere av applikasjonen i testingen, som blant annet personer i metallsøkermiljøet.

Ferie	Lav	Middels	Høy
-------	-----	---------	-----

Måned	Januar				Februar				Mars				April				Mai			
Uke	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Behovsanalyse																				
Videre planlegging																				
Design																				
Utvikling																				
Fase slutt testing																				
Testing, validering og brukertesting																				
Skrive hovedrapport																				
Forberede muntlig presentasjon																				
Dokumentasjon																				

Tabell 7.7: Arbeidsplanen til bachelorgruppen.

Vedlegg 3 Rolletabellen

Rolledimensjon	Rolle	Forklaring av Rollen
Tenkende	Ideskaperen	har tendens til å være svært kreativ og flink til å løse problemer på ukonvensjonelle måter
	Analytikeren	gir teamet et logisk blikk og bidrar med upartiske vurderinger
	Spesialisten	har inngående kjennskap til et kjerneområde og tidlig evne til problemløsning
Sosial	Koordinatoren	har oppmerksomheten rettet mot målet og evnen til å fordele oppgaver på en god måte
	Ressursinnhenteren	innhenter informasjon om konkurrerende team og har blikk for hvordan teamet blir sett utenfra
	Lagspilleren	binder teamet sammen og bidrar der det er behov for det
Handlende	Avslutteren	har evne til å slutføre og har en viktig oppgave mot slutten av en oppgave
	Pådriveren	har en utålmodig og pågående stil som bidrar til fremdrift og at farten blir holdt oppe
	Iverksetteren	har en praktisk og gjennomførbar tilnærming

Tabell 7.8: Roller i prosjektet definert.

Vedlegg 4 PDF Biblioteker

Navn	Link
iText7	https://github.com/itext/itext7
Android Xml To PDF Generator	https://github.com/Gkemon/Android-XML-to-PDF-Generator?utm_source=android-arsenal.com&utm_medium=referral&utm_campaign=8165

Foxit	https://developers.foxitsoftware.com/pdf-sdk/android/
-------	---

Vedlegg 5 Sprint-tabell

Sprint	Uker	Oppgaver
1	4-5	Planlegge mer informasjonsinnhenting. Ytterligere planlegging av struktur, hvordan vi arbeider og innhold av applikasjonen. Kommunisere og drøfte ideer med intern veileder og oppdragsgiver. Markedsresearch og literaturresearch.
2	6-7	Stort møte med RA, arkeologer og metalldetektorister. Kartlegge denne informasjonen, konverter til brukerhistorier og funksjoner. Skrive kravspesifikasjon og få tilbakemeldinger fra oppdragsgiver. Lage noen maler for design. Bryte ned funksjonene til oppgaver som vi setter opp på "trello-kort".
3	8-9	Start utvikling med frontend funksjoner vi vet må med, som kamera og registrering. Ferdigstille design og innhold. Gjøre research på å skrive hovedrapport. Starte med å skrive hovedrapport.
4	10-11	Backend utvikling starter. Sett opp struktur for backenden. Backend metoder som burde være ferdig i løpet av sprinten: CreateUser, LoggInn, RegistrerFunn og HentFunn. Sett opp tester for backenden. Frontend fortsetter: Sette opp for mottak av en brukers funn og registrering.
5	12-13	Backend: Sett opp "hosting" for backenden. Slett og endre metoder for både funn og bruker. Første integrasjonstest mellom front og backend i løpet av sprinten. Frontend: Generering av PDF jobbes med. Registrering av bruker og innlogging sider og funksjonalitet ordnes. Teste integrasjon med backend. Hovedrapport: Start med produktdokumentasjon for backend og frontend og prosessdokumentasjon
6	14-15	Backend: Forsøk å finne ut av hva som stopper bildeoversendingen. Skriv om og eksperimenter til vi finner ut. Ikke vær redd for å bytte ulike rammeverk. Konverter databasen fra SQLite til Azure SQL. Frontend: generering av PDF ferdigstilles. Teste mer med backend teamet. Hovedrapport:
7	16-17	Backend: Sørge for at data blir oversendt ordentlig. Tid databasen for å bryte opp potensielle mange til mange forhold. Forandre på metoder slik at de kan håndtere de nye entitetene. Øke sikkerhet med passord eller token checks.

		Frontend: Visuelle endringer, nye introduksjonssider og sørge for at mottak og oversending til backend fungerer slik vi ønsker. Hovedrapport:
8	18-19	Generell utvikling: Siste finpuss før demonstrasjon for Riksantikvaren Hovedrapport: Ferdigstille utkast for produktdokumentasjon og prosessdokumentasjon.
9	20-21	Hovedrapport: Full fokus på å ferdigstille. Høre med veileder og andre som kan hjelpe oss.

Tabell 7.9: Alle mål satt for de ulike sprintene. **NB:** en del av disse sprintene hadde dager som var i ferieuker

Vedlegg 6 Funnskjema

Funnskjema - metalldetektorfunn

Fyll ut ett skjema pr gjenstand.
Skjemaet skal leveres sammen med gjenstanden til fylkeskommunen
eller Sametinget i det fylket funnet er gjort

Finner		Denne delen fylles ut av finner			Grunneier		
Navn	helge helland		Navn	Per Pedersen			
Adresse	bakkeveien19		Adresse	Bakkeveien23			
Postnr.	1355	Sted	poststed	Postnr.	9876	Sted	Kolsås
Tlf	12345678		Tlf	12345678		Grunneier har gitt tillatelse til søkingen <input checked="" type="checkbox"/>	
E-post	helge@mail.no		E-post	per@mail.no			

Funndato	Funnsted, gård, gnr	Kommune	Fylke
12/05/2021	Bakkeveien, 1234, 1234	Bærum	Viken
Gjenstand	Funndybde	Gjenstand merket med (hvis flere funn)	
Stein	28.0	Stjerne	

GPS-koordinat (øst)	GPS-koordinat (nord)	Datum/ projeksjon	Arealtype:
10.450908333333333	59.90836333333334	1923	Skog <input type="checkbox"/> Aker <input type="checkbox"/> Beite <input checked="" type="checkbox"/> Hage <input type="checkbox"/> Fjell <input type="checkbox"/> Strand <input checked="" type="checkbox"/> Vann <input type="checkbox"/>
Målemetode:	Håndholdt GPS <input type="checkbox"/>	Mobiltelefon <input checked="" type="checkbox"/>	Digitalt kart <input type="checkbox"/>
Andre opplysninger og observasjoner (tidligere funn, sagn/tradisjon, observasjon av trekull, bein, kull, steinkonstruksjoner osv., jordsmonn på stedet mm.)			
Rar Stein			

Denne delen fylles ut av fylkeskommunen/ Sametinget		
Askeladden ID	Saksnummer	Melding om funn mottatt (dato)
Kontroll utført av fylkeskommunen/ Sametinget		
Avstand til nærmeste kjente kulturminne/ funn målt i Askeladden	<input type="text"/> meter	Askeladden ID? <input type="text"/>
Grunneier sjekket mot matrikkel <input type="checkbox"/>	Posisjon <input type="checkbox"/>	Funnsted sjekket i felt <input type="checkbox"/>

Denne delen fylles ut av forvaltningsmuseum			
Mottatt dato	Aks. nr.	Saksnummer	Museumsnummer