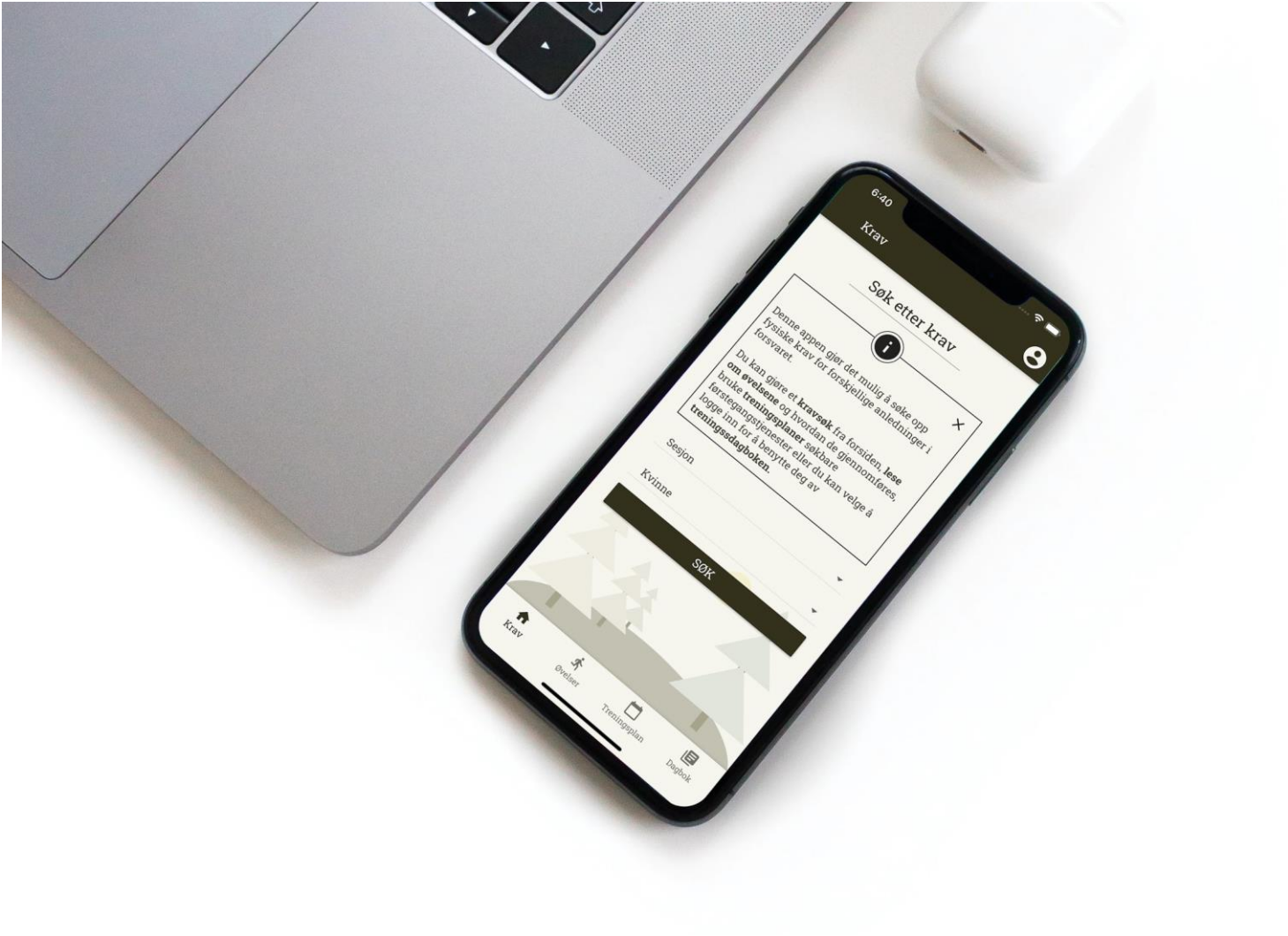


Hovedprosjekt ved Fakultet for teknologi, kunst og design, OsloMet – Storbyuniversitetet, våren 2020

Thora Marie West Mothes
Mathias Lund Ahrn
Martina Rebekka Førre
Tara Jørgensen

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

PROSJEKT NR.
2020-05
TILGJENGELIGHET
Offentlig

BACHELORPROSJEKT

HOVEDPROSJEKTETS TITTEL	DATO
Forsvarets Opptakskrav	24.05.2020
	ANTALL SIDER / BILAG
	145
PROSJEKTDELTAKERE	INTERN VEILEDER
Thora Marie West Mothes Martina Rebekka Førre Mathias Lund Ahrn Tara Jørgensen	Torunn Gjester

OPPDRAGSGIVER	KONTAKTPERSON
Forsvarets Høgskole	Bjørnar Dullum

SAMMENDRAG

Det er utviklet en mobil-applikasjon som omhandler søking på opptakskrav, informasjon relatert til testing og øvelser i det norske Forsvaret. Appen inneholder også funksjonalitet for loggføring av trening.

3 STIKKORD
App-utvikling
Flutter
Firebase

Forord

Dette dokumentet har til hensikt å presentere appen vi har utviklet som hovedprosjekt ved Fakultet for teknologi, kunst og design, OsloMet – Storbyuniversitetet, våren 2020, og utviklingsprosessen knyttet til dette produktet.

Rapporten består av fem deler: introduksjon, prosessdokumentasjon, produktdokumentasjon, konklusjon og kildeliste. Rapporten har også en rekke ulike vedlegg. Det stilles ingen krav om forkunnskap til appen og kan leses av personer uten større teknisk kompetanse. Enkelte faglige ord og uttrykk er skrevet på engelsk, og disse vil bli nærmere forklart i ordlisten (Vedlegg 1), sammen med andre tekniske begreper. Første gang disse begrepene blir introdusert vil de stå i kursiv.

Grunnet situasjonen som oppstod rundt Covid-19 har store deler av prosessen foregått digitalt, og vi så oss blant annet nødt til å skalere ned brukertestingen. Vi tror imidlertid ikke at dette har hatt for stor påvirkning på produktet eller det endelige resultatet.

Vi er takknemlige for all hjelp og støtte vi har fått underveis i prosessen. Vi ønsker spesielt å takke:

- Torunn Gjester for hennes veiledning og støtte under hele prosjektet
- Kirsi Helkala, Bjørnar Dullum, Håvard Engelien og andre representanter fra Forsvaret for deres ekspertise og veiledning
- Sarah Margrethe Thorstensen for de flotte ikonene hun tegnet for oss, som bidro til å sette et personlig preg på appen
- OsloMet – Storbyuniversitetet for tre fine studieår, og for å ha skapt en arena der vi har utviklet vennskap og kunnskap for årene fremover.

Innholdsfortegnelse

1. Introduksjon

Aktører

Utviklingsteam

Teamet består av fire medlemmer fra to ulike bachelorprogram ved OsloMet – Storbyuniversitetet: Dataingeniør og Anvendt Datateknologi.

- Thora Marie West Mothes, Dataingeniør
- Martina Rebekka Førre, Dataingeniør
- Mathias Lund Ahrn, Dataingeniør
- Tara Jørgensen, Anvendt Datateknologi

Figur 1.1: Bilder av utviklingsteamet. Fra venstre til høyre: Thora Marie West Mothes, Martina Rebekka Førre, Mathias Lund Ahrn og Tara Jørgensen.

Intern veileder

Torunn Gjester,
Universitetslektor ved Fakultet for teknologi, kunst og design, Institutt for informasjonsteknologi, OsloMet – Storbyuniversitetet

Produkteier

Prosjektets produkteier er Forsvarets høgskole, som er en akademisk uavhengig institusjon som har som hovedoppgave å utdanne alt befal og alle offiserer i Norge (Forsvaret, 2019, 2. avsn.).

Per April 2020, består Forsvarets høgskole (heretter kalt produkteier) av fem utdanningsavdelinger, to institutter, en fag- og en driftsstab (Forsvaret, 2019, 1. avsn.). Denne bacheloroppgaven er skrevet i samarbeid med Cyberingeniørskolen, lokalisert på Jørstadmoen, og Stabsskolen, som ligger på Akerhus festning.

Eksterne veiledere

Kirsi Helkala,
Professor ved Cyberingeniørskolen, Forsvarets høgskole

Bjørnar Dullum,
Fagansvar/hovedlærer i militær idrett og trening ved Stabsskolen, Forsvarets høgskole

Prosjektbeskrivelse

Bakgrunn for prosjektet

For å finne informasjon om hva som kreves under forskjellige fysiske tester i Forsvaret, må man i dag besøke nettsidene deres. Ifølge representantene vi har snakket med i Forsvaret, er sidene som inneholder informasjon om trening og minstekrav de mest besøkte av Forsvarets nettsider. Flesteparten av de vi snakket med gjennom dette prosjektet, fortalte at de brukte relativt mye tid på å svare på spørsmål om ulike krav til fysiske tester. Dette er sannsynligvis fordi informasjonen på nettsidene ligger spredt på forskjellige sider, noe som gjør at det ikke er enkelt å få god oversikt. I tillegg er dokumentet "Reglementet om fysisk testing", hvor informasjonen originalt kommer fra, langt og noe vanskelig å finne fram i.

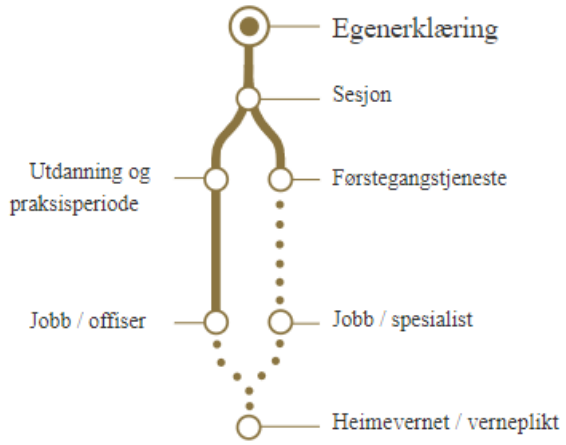
Produkteier observerer at ansatte og nye rekrutter har problemer med å holde oversikt over hva som kreves av dem fysisk for for å kunne tiltre i de ulike stillingene i Forsvaret. Det kan også være vanskelig å holde oversikt over de obligatoriske testene som kreves i løpet av ansettelsesperioden. Produkteier ønsker en app som vil vise relevant informasjon til den individuelle brukeren, uten at man trenger å bla

gjennom mye ikke-relevant data. Hovedmålet med appen er på lang sikt å bidra til kvalitetssikring av fysisk form i Forsvaret.

Oppbyggingen av den norske militærtjenesten

I Norge har vi verneplikt hvor både kvinner og menn kan bli kalt inn til tjeneste, dersom de blir vurdert til å være tjenestedyktige på sesjon. Den første delen av militærtjenesten kalles førstegangstjeneste og består av rekruttskole i 6-8 uker, før man tjenestegjør i en tildelt avdeling (Forsvaret, u.å, Rekruttskolen: Slik blir den første tiden i Forsvaret).

All ungdom som potensielt er vernepliktig må svare på sesjon del 1, som i praksis er en egenerklæring av fysisk form (Forsvaret, u.å, Egenerklæringen). Deretter blir de som er aktuelle for militærtjeneste og verneplikt kalt inn til sesjon del 2. Sesjon del 2 består av et sett med fysiske tester, en teoretisk test, helsekontroll og en motivasjonssamtale. Etter sesjon kan man bli innkalt til førstegangstjeneste, eller få muligheten til å søke utdanning (Forsvaret, u.å, Hva er sesjon?).



Figur 1.2. Flytdiagram av prosessen fra egenerklæring til aktiv tjeneste i Forsvaret (Forsvaret, u.å, Egenerklæringen).

I førstegangstjenesten har man også muligheten til å søke seg til spesielle tjenester innad i Forsvaret, som er mer krevende enn vanlig tjeneste (Forsvaret, u.å, Søkbare førstegangstjenester). Disse spesielle tjenestene har egne søknadsprosesser og egne opptakskrav. Eksempler på slike tjenester er Fallskjermjeger, Marinejeger og Jegertroppen.

Fysisk testing i Forsvaret

Sesjon del 2 er første gang potensielle vernepliktige blir testet fysisk. I tillegg til sesjon finnes det flere andre anledninger hvor militært personell blir testet. Alle disse anledningene har egne sett med øvelser og minstekrav (Forsvaret, 2019, 5.3 opptak til militær utdanning og tjeneste).

Anledninger	Hvem	Hyppighet og utførelse
Sesjon	Potensielle vernepliktige personer	Utført på sesjon del 2
Førstegangstjeneste	Vernepliktige kalt inn til militærtjeneste	Utført innen de første tre ukene av rekruttskolen, og innen de siste tre månedene før dimisjon
Spesielle førstegangstjenester	Søkere til den aktuelle tjenesten	Utført på opptak til den aktuelle tjenesten
Utdanning	Studenter som tar utdanning i Forsvaret	Utføres ved felles opptak og seleksjon (FOS), og ved andre opptak til vervet tjeneste. Studenter under utdanning vil avlegge prøve hvert år
Årlig test	Ansatte i Forsvaret	Utført årlig

Generell testing i Forsvaret reguleres av “Reglement om fysisk testing”, som definerer hvem som skal bli testet ved en gitt anledning, hvilke øvelser som er del av testen og hva som skal til for å oppnå hvert minstekrav (Forsvaret, 2019, 1 Innledning). Testing til spesielle førstegangstjenester reguleres ikke av dette dokumentet, men blir fastsatt av hver avdeling individuelt.

Hva man må oppnå for hvert minstekrav for generelle tester er i hovedsak bestemt av kjønn og anledning. I tillegg er det en aldersfaktor når det kommer til årlig testing for militært ansatte.

"Reglementet om fysisk testing" er et langt dokument som består av detaljert informasjon om hver anledning og hvordan forskjellige øvelser skal utføres. Det inneholder også informasjon som *kun* er relevant for den som administrerer testen. Tabellene vist i figur 1.3 og 1.4 brukes til å vise prestasjonen som skal til for å oppnå de forskjellige kravene for en anledning.

			Minimumskrav								
Egenskap	**Testøvelse**	**Enhet**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Utholdenhet	3000 m løp	min:sek	18:00	16:30	15:00	14:30	14:00	13:30	13:00	12:30	12:00
	Bip-test	level:shuttle	6:1	7:4	8:8	9:3	9:8	10:2	10:7	11:1	11:6
Styrke	Medisinballstøt	meter	3,7	3,9	4,1	4,2	4,4	4,5	4,6	4,8	5,0
	Stille lengde	meter	1,85	1,95	2,05	2,15	2,20	2,25	2,30	2,35	2,45
	Pull-ups	repetisjoner	3-B	6-B	1-A	3-A	4-A	5-A	7-A	8-A	10-A

Figur 1.3: Tabell fra "Reglement om fysisk testing" som viser minstekrav, øvelser og prestasjoner relevant for menn som gjennomfører tester på sesjon. (Forsvaret, 2019, 5.1 Sesjon)

			Minimumskrav								
Egenskap	**Testøvelse**	**Enhet**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Utholdenhet	3000 m løp	min:sek	20:30	18:00	16:30	15:45	15:00	14:30	14:00	13:30	12:00
	Bip-test	level:shuttle	4:4	6:1	7:4	7:10	8:8	9:3	9:8	10:2	11:6
Styrke	Medisinballstøt	meter	2,5	2,7	2,9	3,1	3,2	3,3	3,5	3,7	5,0
	Stille lengde	meter	1,45	1,60	1,70	1,80	1,85	1,95	2,05	2,15	2,45
	Pull-ups	repetisjoner	1-B	2-B	4-B	6-B	8-B	10-B	1-A	3-A	10-A

Figur 1.4: Tabell fra "Reglement om fysisk testing" som viser minstekrav, øvelser og prestasjoner relevant for kvinner som gjennomfører tester på sesjon. (Forsvaret, 2019, 5.1 Sesjon)

Minstekravene strekker seg fra en til ni, hvor ni er beste prestasjon. Forskjellige posisjoner, studier og avdelinger har sine egne minstekrav som søkere er nødt til å oppfylle for å ha muligheten til å bli tatt opp (Forsvaret, 2019, 3.6 beregning av oppnådd minimumskrav). Dette er noe man må undersøke med den spesifikke avdelingen man ønsker å søke seg til.

Målgrupper

- Ungdom som skal inn til testing på sesjon og førstegangstjeneste
- Ansatte i Forsvaret som gjennomgår årlig testing
- Personer som søker opptak til studier i Forsvaret, og elever ved Forsvarets høgskole

Tall på potensielle brukere

Basert på målgruppene kan vi gjøre en grov estimering av hvor mange brukere som kan være interesserte i å bruke appen.

I følge "Reglementet om fysisk testing" er alt militært personell testpliktig. Militært personell defineres der som:

- Alle potensielt vernepliktige personer
- Alle soldater eller lærlinger i førstegangstjeneste
- Alle søkere til militær utdanning og tjeneste, i tillegg til nåværende studenter
- Alle militært ansatte. Dvs. personer med militær bakgrunn som er ansatt i Forsvaret (Utdanning.no, 2020, Du kan være ansatt både sivilt og militært i Forsvaret).

Sivilt ansatte i Forsvaret er som regel ikke pålagt å gjennomføre årlige tester, men har mulighet til å delta om de ønsker (Forsvaret, 2019, 1.3 Testpliktige personellgrupper).

Studenter og rekrutter

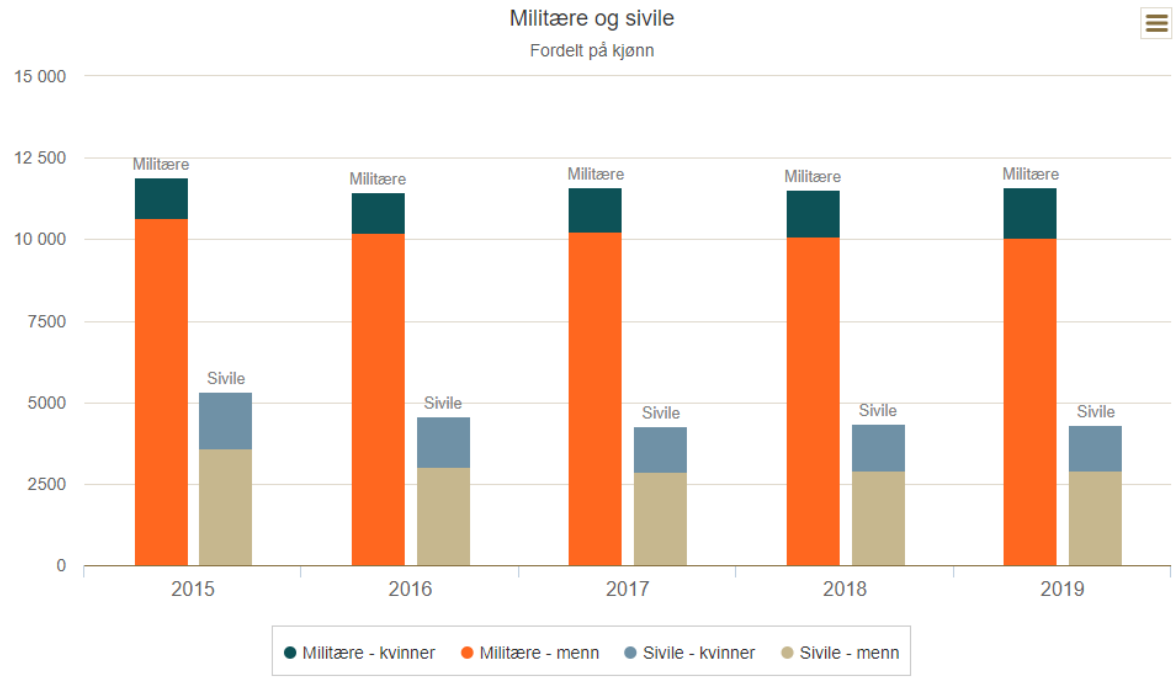
Sesjon del 1 er obligatorisk for alle født i det aktuelle kullet som kan kalles inn til militærtjeneste det året (Forsvaret, (u.å), Hvorfor må jeg dette?). Av dem som svarer på sesjon del 1 kalles ca en fjerdedel inn til sesjon del 2 (Forsvaret, (u.å), Sesjon), og her gjennomføres det en fysisk test. Eksempelvis ble det født ca 58 000 barn i år 2000 (SSB, 2020, Fødte), som vil si at om lag 14000-15000 personer gjennomgår de fysiske testene på sesjon del 2 årlig. Etter sesjon kalles ca 50% av deltakerne inn til førstegangstjeneste eller får mulighet til å søke utdanning. Ut fra disse tallene kan vi estimere at ca 8000 blir erklært tjenestedyktige på sesjon del 2 (Forsvaret, (u.å),

Hvilke muligheter har jeg i førstegangstjenesten) og får mulighet til å gå inn i tjeneste eller utdanning. Omtrent 200 personer blir tatt opp årlig til utdanning (Hultgreen, 2018), hvor man blir testet årlig gjennom en bachelorgrad på 3 år.

Militært ansatte og sivile

Alle militært ansatte gjennomgår obligatoriske, årlige fysiske tester. Noen sivile kan også være pålagt å utføre disse testene.

I følge Forsvarets årsrapport er tallene på ansatte i Forsvaret de siste fem årene som følger:



Figur 1.5: En graf hentet fra Forsvarets årsrapport 2019. Grafen viser antall militære og sivile, fordelt på kvinner og menn, for årene 2015 - 2019 (Forsvaret, 2020, Personell)

Basert på grafen i figur 1.5 kan vi estimere at antallet ansatte som potensielt gjennomførte fysiske tester i 2019 ligger på rundt 16 000 ansatte, inkludert sivile.

Andre

I tillegg til målgruppene har vi gjennom samtaler med Forsvaret kommet frem til at også ansatte som jobber med testing, jobber tett med rekruttering eller med studenter kan ha nytte av en slik app. Flere av dem vi snakket med ga uttrykk for at

en slik app kunne kortet ned tiden de bruker på å slå opp krav og informasjon om ulike tester, da dette er noe de ofte får spørsmål om.

Estimert antall brukere

Dermed sitter vi igjen med potensielt 14000-15000 nye brukere hvert år, i tillegg til ca 12000 ansatte i Forsvaret som testes årlig. Dette inkluderer imidlertid ikke de sivilt ansatte, og heller ikke de under utdanning.

Mål for prosjektet

Effektmål

Produktets effektmål

- Enklere og raskere tilgang på informasjon om fysisk testing
- Kvalitetssikre fysisk form i Forsvaret

Utviklingsteamets effektmål

- Forbedre samarbeidsferdigheter
- Erfaring med arbeidslivet, jobbe med produkteier

Appen har som hovedmål å kvalitetssikre fysisk form i Forsvaret ved å effektivisere informasjonsflyten rundt fysiske krav og tester. Ved å gjøre det enklere å finne ut hva som kreves til hvilken anledning vil kunnskapsnivået om fysiske krav øke, og på den måten bidra til et jevnere og bedre fokus på fysisk form i forbindelse med testene. Ved bruk av treningsdagboken og treningsplanene vil det bli enklere å trene mot målene man ønsker å oppnå.

Ved å effektivisere informasjonsflyten vil både de som deltar i testene, de som avholder testene og de som jobber tett med rekruttering få lettet arbeidsmengden og tiden det tar å finne svar på spørsmål om konkrete krav til en anledning eller test.

Utviklingsteamet har hatt et mål om å lære mer om å jobbe tett med en produkteier og utvikle våre samarbeidsevner både mot kunde og innad i teamet. Dette er viktige egenskaper å ha i våre fremtidige karrierer.

Resultatmål

Produktets resultatmål

- Appens funksjonalitet, i henhold til tabellen under
- Appen tilfredsstiller Forsvarets sikkerhetskrav
- Appen er tilgjengelig for alle i målgruppen

Utviklingsteamets resultatmål

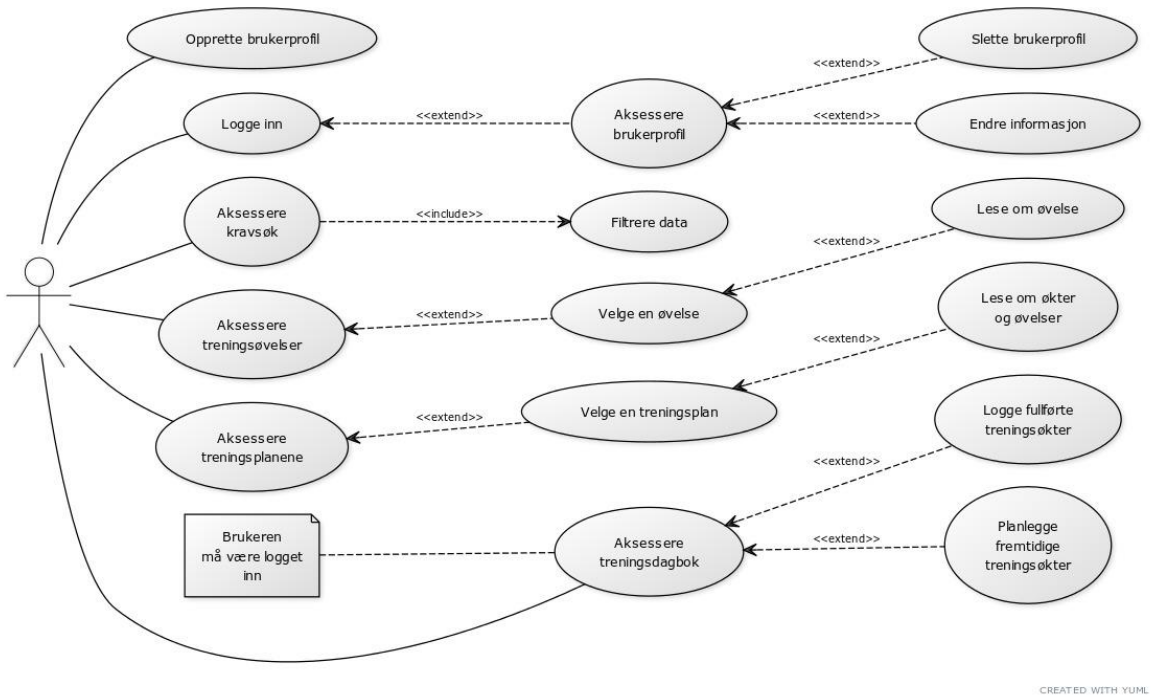
- Lære ny teknologi

Prosjektperioden skal resultere i en funksjonell app som kan bli lastet ned og tatt i bruk av målgruppene våre, og andre parter som måtte ønske det. Appen skal være et oppslagsverk for Forsvarets minstekrav, og en helhetlig og fungerende treningsapp.

Funksjonsmessig er appen delt inn i seks hovedområder, som alle har hvert sitt spesifikke resultatmål. Tabellen under viser disse målene, mens Figur 1.6 viser en mer detaljert oppbygging av en brukers bruksmønster relatert til hovedområdene.

Funksjonalitet / område	Tilgjengelig med eller uten å logge inn?	Mål
Opprette brukerprofil	Ikke relevant	En bruker skal kunne opprette en profil for å kunne aksessere bestemte funksjonaliteter i appen.
Logge inn	Ikke relevant	En registrert bruker skal kunne logge inn ved hjelp av e-mail og passord.
Kravsøk	Uten å logge inn	En bruker skal kunne aksessere informasjon om Forsvarets minstekrav for bestemte øvelser.
Treningsøvelser	Uten å logge inn	En bruker skal kunne aksessere

		informasjon om ulike øvelser og hvordan de utføres for å kunne forbedre måten de trener på.
Treningsplaner	Uten å logge inn	En bruker skal kunne aksessere ferdiglagde treningsplaner de kan følge for å bedre imøtekomme fysiske krav.
Treningsdagbok	Med å logge inn	En bruker skal kunne planlegge fremtidige treningsøkter og logge fullførte treningsøkter for å bedre strukturere treningen og holde oversikt over progresjon.



Figur 1.6: Et use case-diagram av middels detaljnivå som viser de seks bruksmønstrene en appbruker kan ha.

Fordi appen utvikles i samarbeid med Forsvaret skal det være stort fokus på sikkerhet, og lagring av data skal følge *GDPR* / personvernforordningen. Som utviklingsteam har vi også hatt som mål å lære oss nye effektive og moderne teknologier som vi kan dra nytte av videre i vår karriere.

Kort produktbeskrivelse

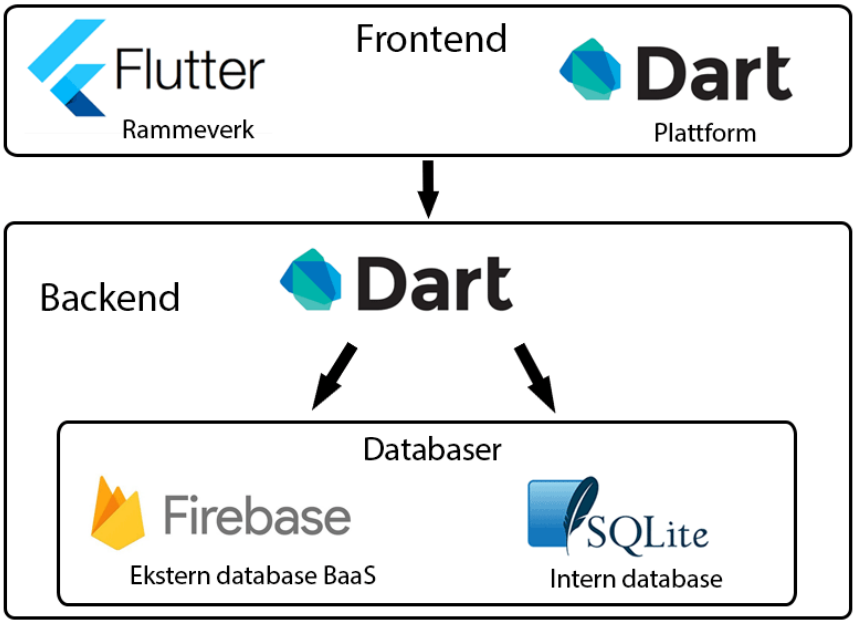
Vi har utviklet en app som vil tilby nye rekrutter, studenter og ansatte i Forsvaret en enklere og mer effektiv måte å holde seg oppdatert på hva som kreves av dem fysisk. Appen vil gjøre det enklere og mer oversiktlig for dem når de skal jobbe mot disse kravene.

Produkteiers minstekrav til appen er at man enkelt skal kunne søke opp krav til gitte anledninger basert på sine personlige forutsetninger, slik som kjønn og anledning. Ut ifra dette, arbeidet vi sammen med produkteier om å utforme tilleggsfunksjonalitet som kan være relevant for brukerne av appen. Vi grupperte den ønskede funksjonaliteten inn i fire hovedkategorier, vist i figur 1.7. Tabellen under figuren viser funksjonaliteten kategorisert under funksjonelle og ikke-funksjonelle krav. Den offisielle kravspesifikasjon er vedlagt i vedlegg 2.

Figur 1.7: Hovedkategoriene for appfunksjonalitet.

Funksjonelle krav	Ikke-funksjonelle krav
Se informasjon om øvelser og hvordan disse utføres	Raskere å slå opp på krav enn gjennom nettsiden
Treningsprogram	Fungerer på både iPhone og Android
Loggføre resultater og progresjon	Raskt å laste inn data
Opprette brukerprofil med egendefinert data	Følger personvernforordningen/GDPR
	Har høy sikkerhet knyttet til lagring av data

Appen er utviklet i programmeringsspråket Dart med *frontend*-rammeverket Flutter. Dart fungerer også som *backend* i produktet, og knytter sammen dataflyten mellom lagringsmedium og grensesnitt. Gjennom Dart kommuniserer appen med en intern SQLite-database og en ekstern Firebase-database. Figur 1.8 viser hvordan disse teknologiene er koblet sammen.



Figur 1.8: Teknologiene appen er utviklet med, og hvordan de er koblet sammen.

2. Prosessdokumentasjon

Prosjektperioden har vært en omfattende prosess fra idé til produkt. Vi presenterte selv ideen om en app ovenfor produkteier, og sammen kom vi frem til et *Minimum Viable Product* vi baserte hele utviklingsprosessen på.

Mesteparten av informasjon som direkte kan relateres til prosessen, og ikke til produktet i seg selv, ligger i vedlegg 3. Her ligger blant annet en oppsummering av planleggingen og utførelsen av prosjektet, i tillegg til detaljer om prosjektets workflow og utviklingsmetodikk. Vedlegget inneholder også notater vi tok underveis i prosessen, og detaljert informasjon om prototypingen av produktet.

Metode og prosessverktøy

Scrum

Scrum er en metode for å utvikle informasjonssystemer. Man bryter ned prosjektet i sprinter som går over kortere perioder med klare progresjonsmål. Metoden innebærer korte daglige møter hvor man går igjennom hva man har gjort, hva man jobber med nå og hva man eventuelt har problemer med.

Hvorfor vi har brukt Scrum:

- Korte sprinter på to uker til maksimalt en måned passet godt inn med antatt tid for utvikling av ulike komponenter
- Det oppfordrer til tett samarbeid og økt kommunikasjon mellom teammedlemmer
- Det fungerer godt for planlegging og for å se tilbake på det som er gjort, for å forbedre prosessen videre.

En oversikt over de gjennomførte sprintene ligger i vedlegg 3.9. Selv om vi ikke møttes ansikt til ansikt på en daglig basis, kommuniserte vi gjennom et eller flere online kommunikasjonsverktøy omtrent 3-5 dager i uken. Vi byttet på hvem som var scrummaster, og som dermed var ansvarlig for backloggen og oppfølging av

utviklingsteamet fra sprint til sprint. Vi hadde sprint-planlegging ved begynnelsen av hver sprint for å planlegge hva som skulle gjøres i den kommende sprinten. Vi hadde også sprint-gjennomgang ved slutten av hver sprint for å oppsummere hva som var blitt gjort, og hva vi ikke hadde rukket. Vi gjennomførte også sprint-retrospektiv der vi reflekterte over hva som hadde fungert bra og dårlig i den foregående sprinten, og hvilke endringer vi kunne implementere for å forbedre oss. For å holde oversikt over hver sprint brukte vi et scrum-board i notion der oppgavene ble delt inn i "backlog", "next up", "in progress", "for review" og "complete" (Vedlegg 3.8).

Teknologier og verktøy

Program	Hva det er	Hva vi brukte det til og hvorfor
Slack 	En kommunikasjons-plattform med støtte for tekst- stemmechat	Til å kommunisere, planlegge, dele linker og idéer osv. Vi kunne koble ulike "bots" til programmet, for eksempel en Github-bot, som ga oss notifikasjoner
Discord 	En kommunikasjons-plattform med støtte for tekst- stemmechat	Til å holde møter innad i gruppen. Vi var alle kjent med hvordan programmet fungerer, så det var raskere å benytte seg av dette enn noe annet.
Notion 	Et online arbeidsområde med støtte for samarbeid	Et felles arbeidsområde med kalender, prosjektdagbok, møtenotater, scrub-brettet vårt, og en rekke seksjoner for notater og ideer. Det hjalp oss med å være organiserte og i kontroll over hva som måtte gjøres.
Google Drive 	En online skybasert lagringstjeneste med støtte for fildeling og samarbeid	Et felles lagringssted for bilder og ulike dokumenter. Det var fint å ha et sted der alle hadde tilgang til de samme filene.

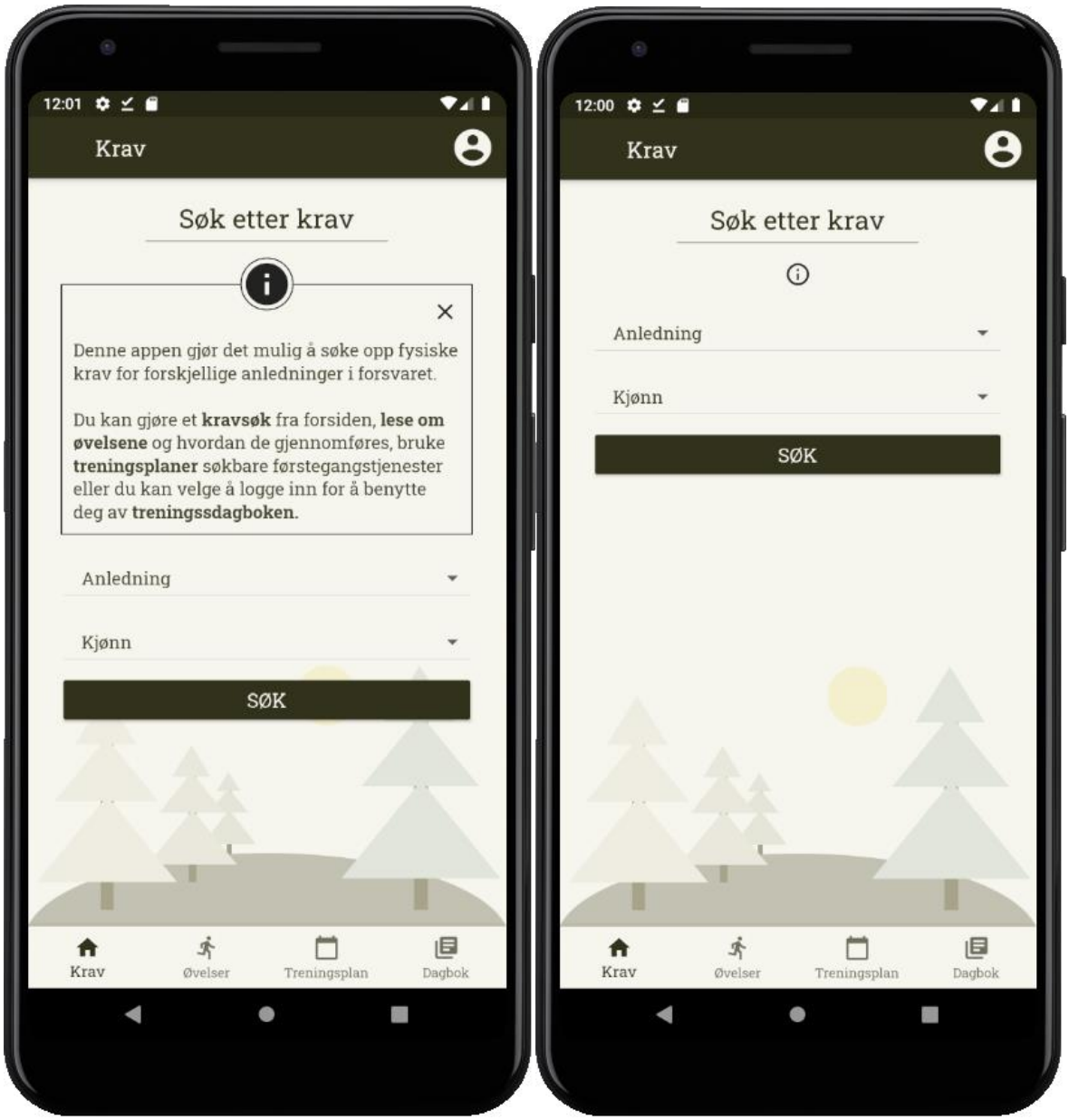
Google Docs 	En online tekstbehandler med støtte for samarbeid	Til å skrive notater og ideer, men hovedsakelig til å samarbeide om å skrive selve hovedoppgaven.
Figma 	Et online designverktøy med støtte for samarbeid der man kan lage digitale prototyper, designskisser og mockups.	Til å prototype og gjennomføre brukertesting og testing av appens informasjonsarkitektur ved å la brukerne leke seg med det vi hadde forberedt. Det var et svært nyttig verktøy når det kom til å bestemme layout og design, ettersom man kunne se for seg det ferdige produktet på en mye bedre måte enn man kunne ved bruk av papir-prototyper.
Visual Studio Code / Android Studio 	En kode-editor og et integrert utviklingsmiljø	Til å kode appen og emulere en telefon der vi testet hvordan koden fungerte. Vi valgte hvilket program vi ville bruke basert på personlig preferanse.
Git / Github 	Et versjonskontroll-system og en tjeneste som tilbyr vertstjenester for kode med bruk av Git.	Til å holde styr på ulike versjoner av koden, og til å jobbe sammen med den samme kildekoden på tvers av maskiner. Github gjorde det dessuten mulig for oss å vurdere hverandres arbeid, og kommentere med forslag til endringer.
CI/Github action 	*Continuous integration* verktøy for Github-*repositories*	Verktøy som tester og bygger all kode som blir pushet til *repoet*. Er med på å sikre at appen til enhver tid fungerer.

Rive 	Animasjonverktøy for å animere vektorgrafikk	Lage errorapp-animasjonen (svingende lyspære)

3. Produktdokumentasjon

Produkt

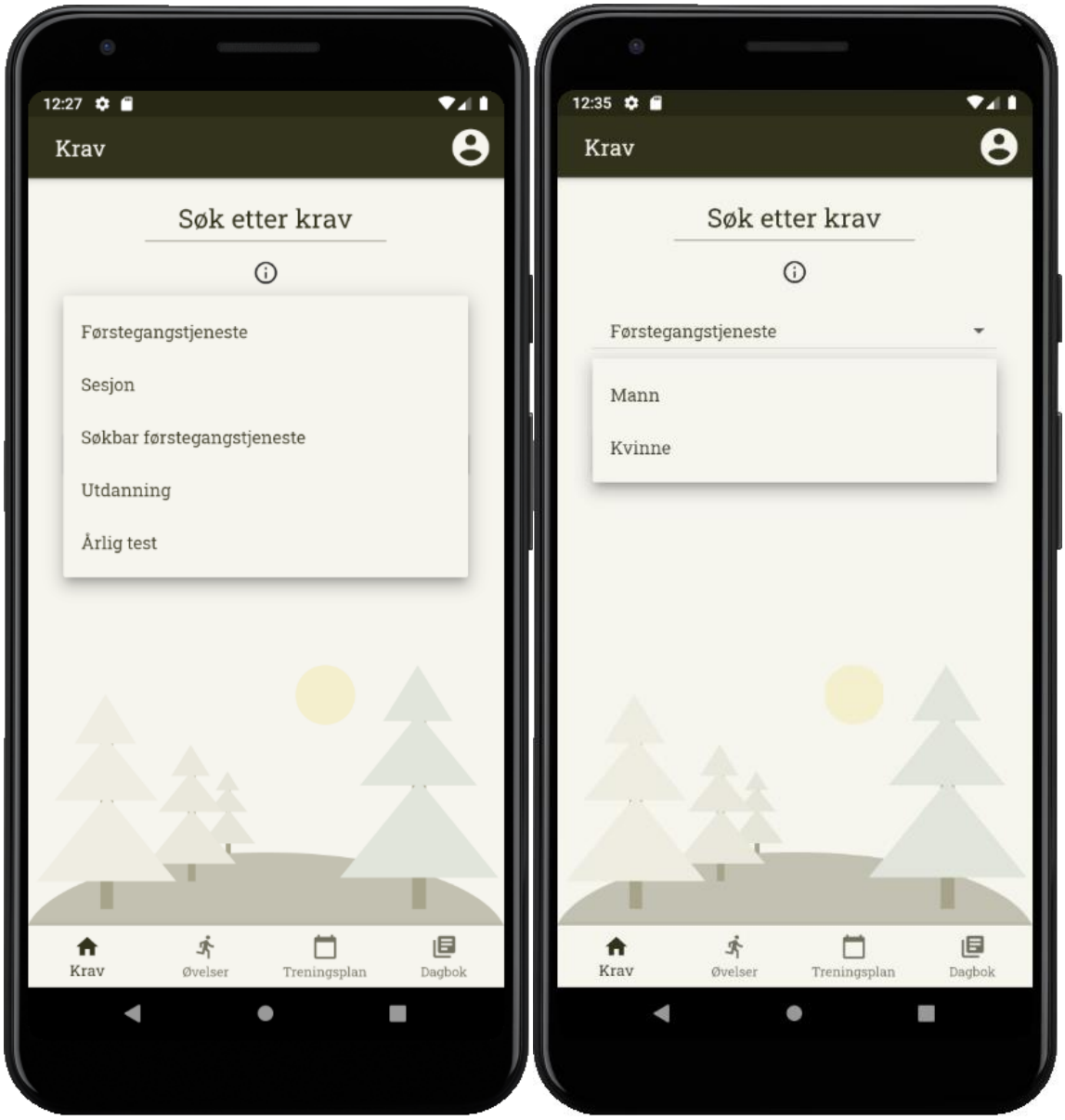
Forside / Kravsøk



Figur 3.1: Åpen og lukket infoboks.

Det første brukeren ser når appen åpnes er siden for kravsøk. Her vil man bli møtt med en infoboks som gir grunnleggende informasjon om hva appen kan brukes til.

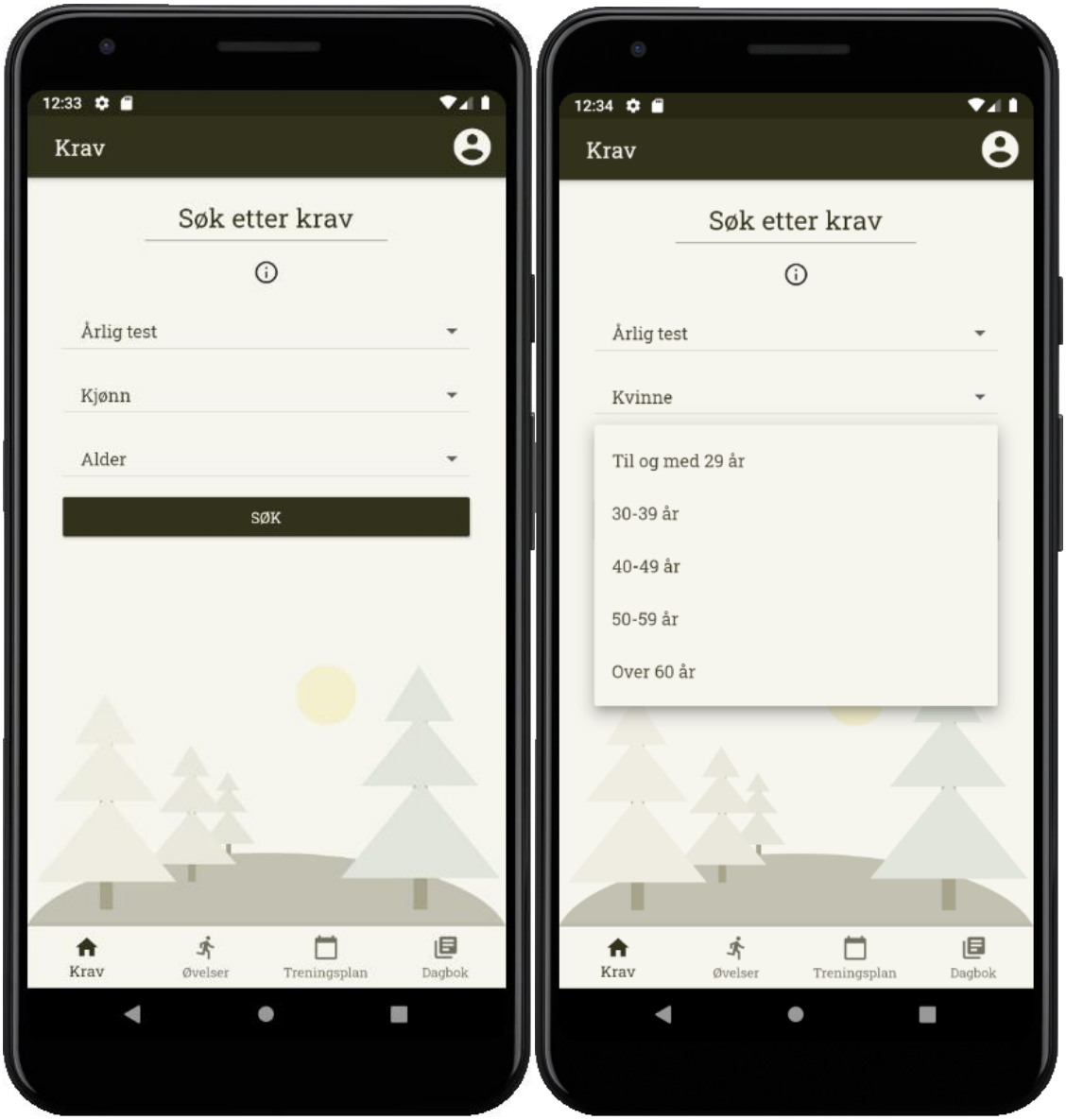
Denne infoboksen kan brukeren selv lukke og åpne etter eget ønske. Hvorvidt infoboksen er åpen eller lukket registreres og lagres i *delte preferanser* på telefonen. Dette betyr at om man har lukket infoboksen vil man ikke se denne neste gang man åpner appen, og åpner man den igjen før man lukker appen vil den være åpen ved neste oppstart.



Figur 3.2: To nedtrekkslister i kravsøk-filteret.

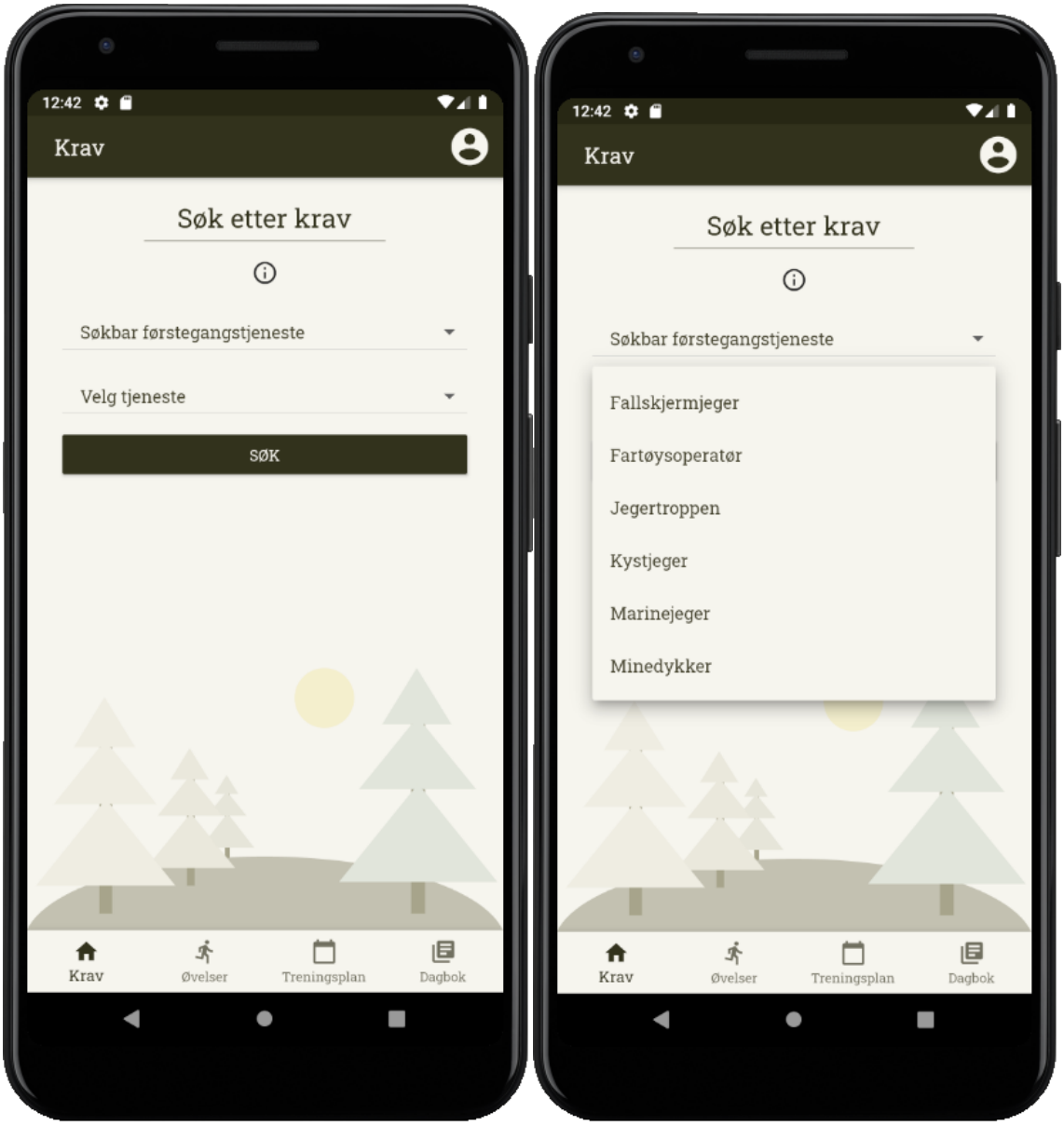
Under infoboksen ligger filteret for kravsøk. Filteret består av et sett med nedtrekkslister hvor brukeren kan velge søkeparametere. Basert på hvilken

anledning som blir valgt i den øverste nedtrekkslisten, vises det andre nedtrekkslister som må fylles ut for å søke på krav til den valgte anledningen.



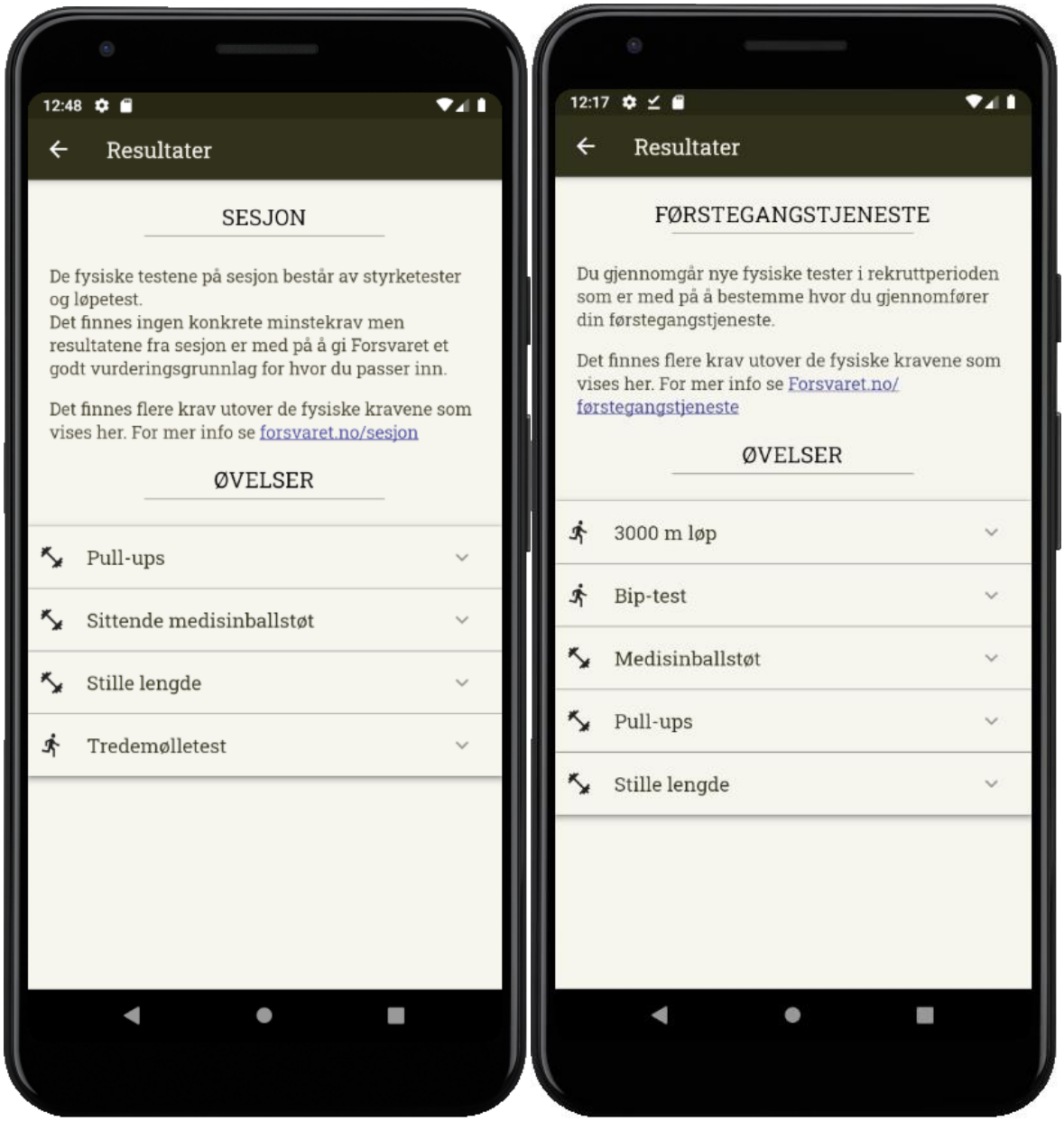
Figur 3.3: Tre nedtrekkslister i kravsøk-filteret. Her må alder velges.

For førstegangstjeneste, sesjon og utdanning må kun kjønn velges, men for årlig test må brukeren i tillegg velge en aldersgruppe (Figur 3.3).



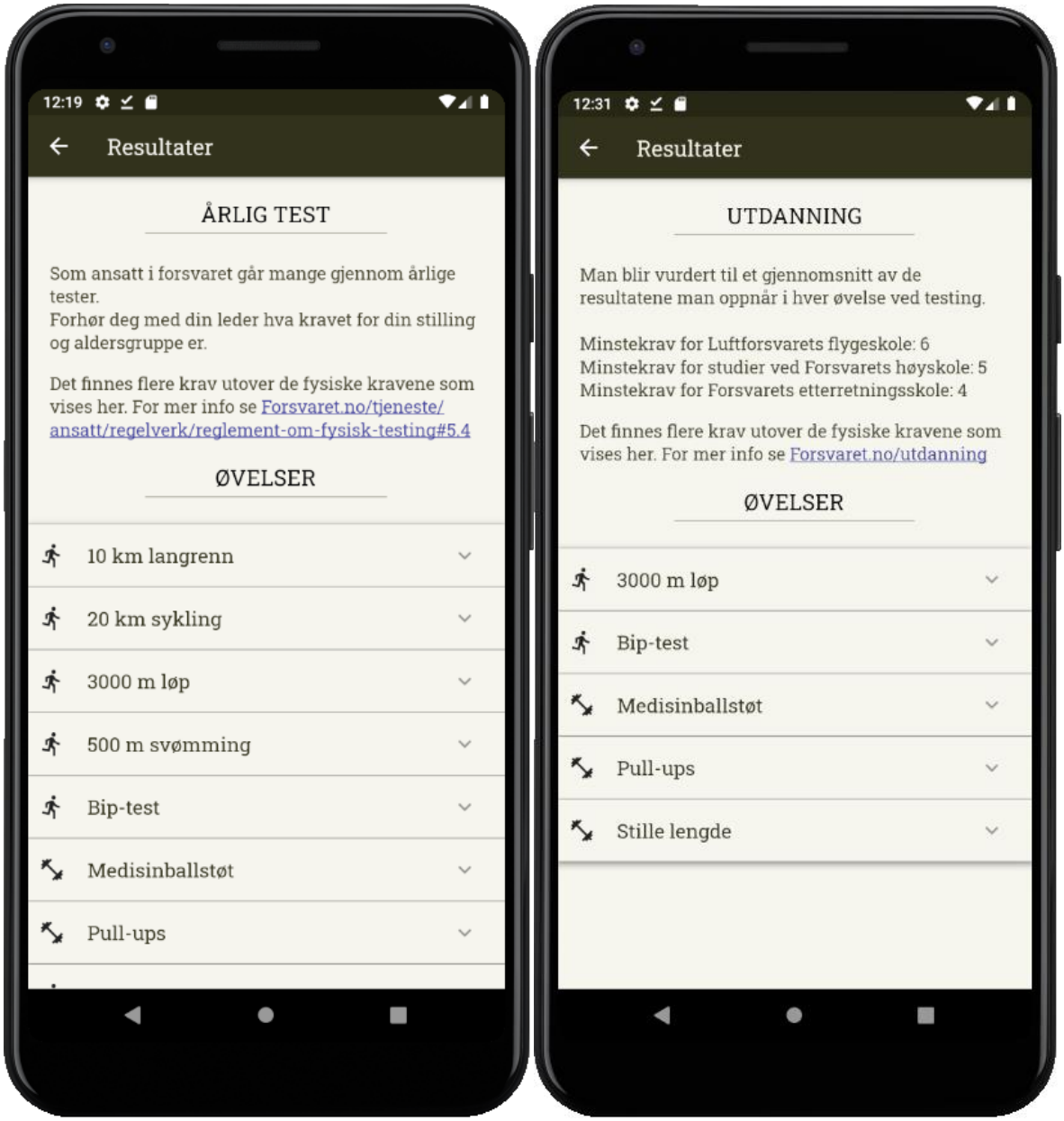
Figur 3.4: To nedtrekkslister i kravsøk-filteret. Her må spesifikk tjeneste velges.

Ved søkbare førstegangstjenester, som Fallskjermjeger, Marinejeger osv, må bare tjeneste velges ettersom kravene er kjønnsnøytrale og ikke aldersbetinget (Figur 3.4).



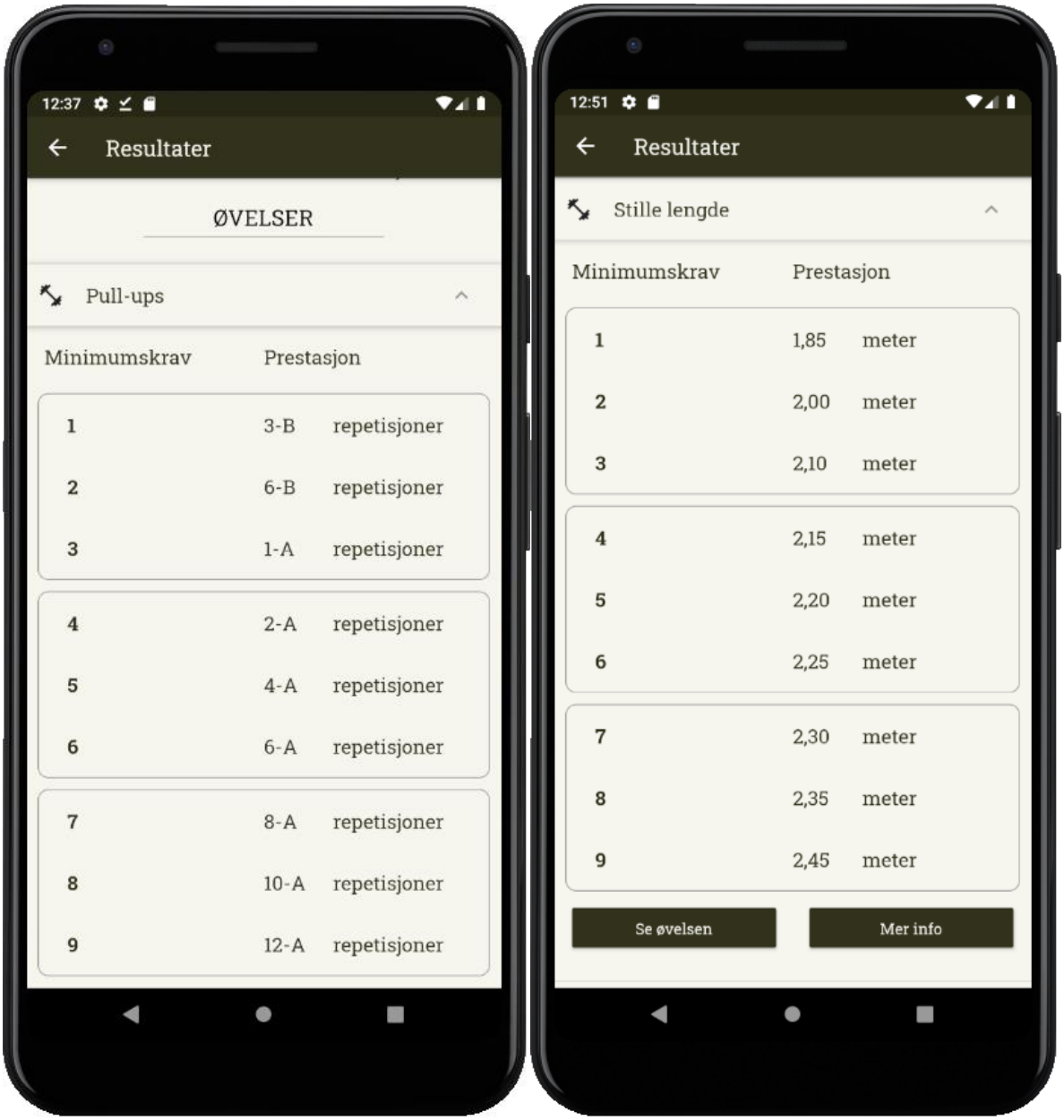
Figur 3.5: Søkeresultater etter et søk i filteret, her for sesjon og førstegangstjenete.

Ved et gjennomført søk vil brukeren få opp en ny side hvor resultatet vises (Figur 3.5). Resultatet for et søk er delt opp i to hovedseksjoner. Øverst finner man navnet på anledningen man har søkt på, og en beskrivelse av testen som utføres ved anledningen. Her finner man også en link til forsvarets sider, dersom man skulle ønske ytterligere informasjon utover det som presenteres i appen.



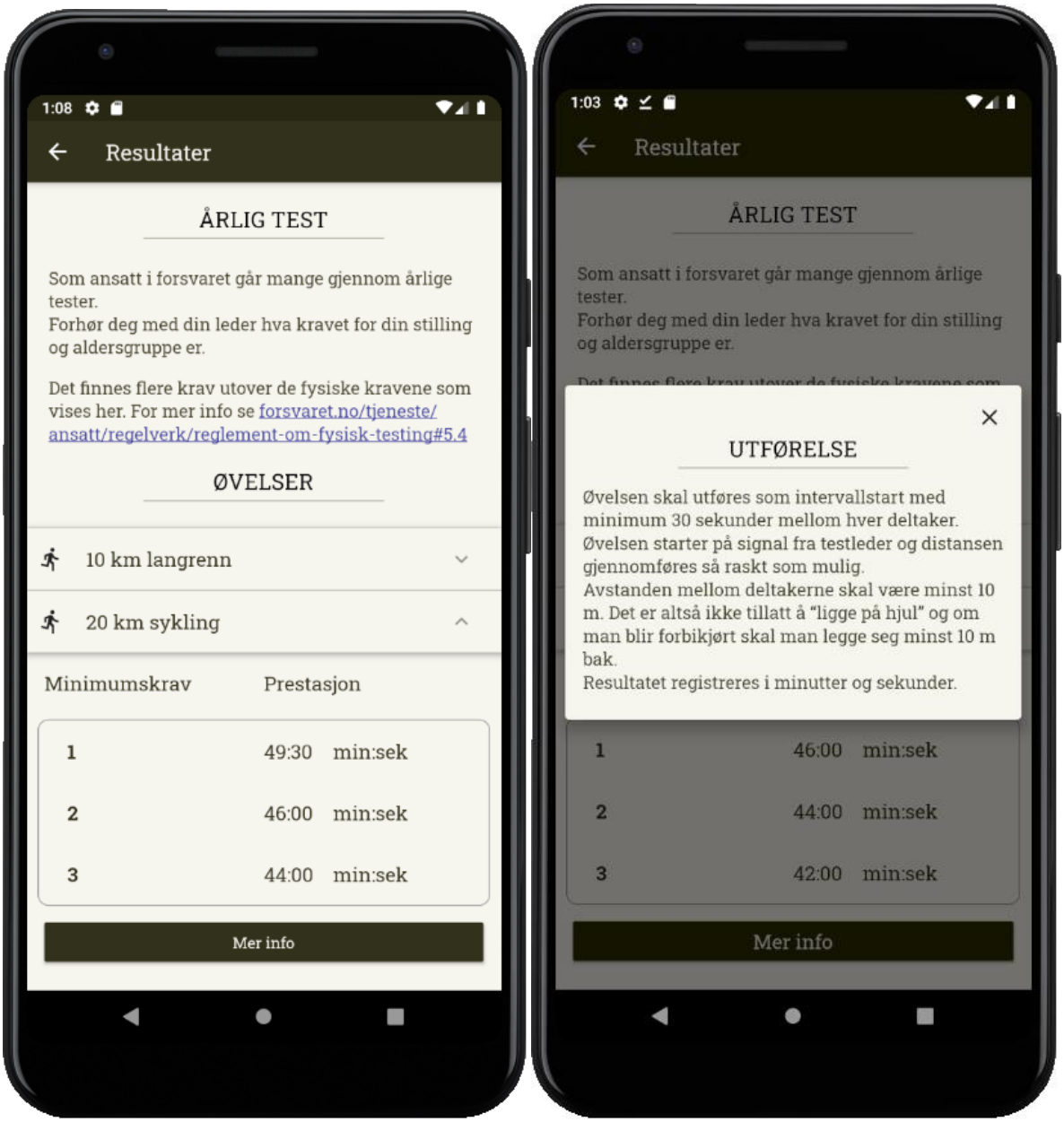
Figur 3.6: Flere søkeresultater fra søk i filteret, her for årlig test og utdanning.

Den nederste delen består av en oversikt over øvelsene som tilhører anledningen. Øvelsene er delt inn i utholdenhetsøvelser og styrkeøvelser hvor hver øvelse har et eget ikon, for å vise hvilken type den tilhører. Øvelsene for utdanning og førstegangstjeneste er like, men krever forskjellig prestasjon for å oppnå de ulike minstekravene. De årlige testene har flere valg når det kommer til utholdenhets testene enn det andre anledninger har (Figur 3.6, venstre bilde).



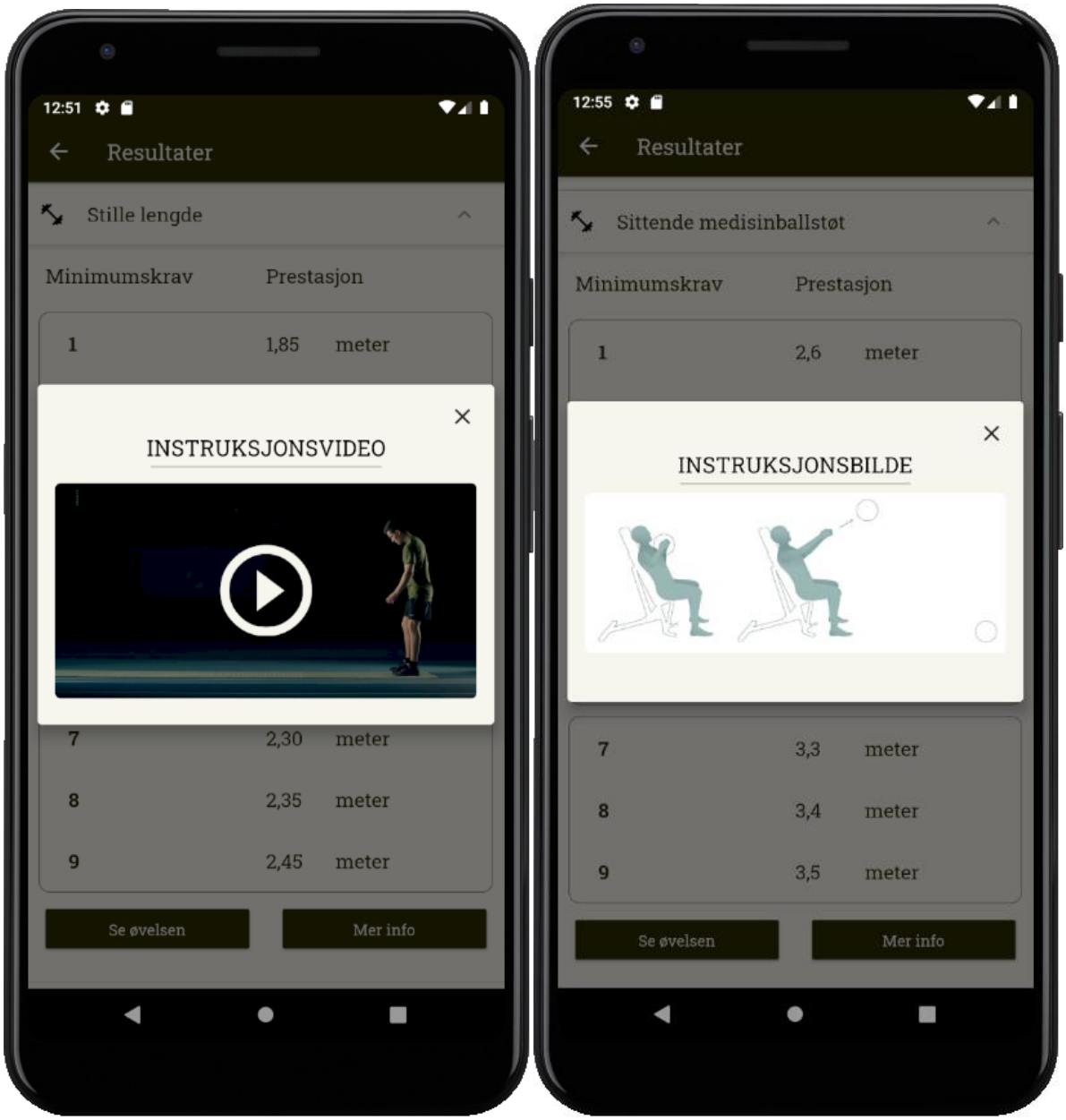
Figur 3.7: Åpne ekspanderbare panel som viser oversikt over minstekrav for forskjellige øvelser

Når man trykker på en øvelse vil den ekspandere og vise mer informasjon om kravene (Figur 3.7). Informasjonen er gruppert for økt lesbarhet. Nederst, under minstekravene, finner man knapper for å lese mer om øvelsen eller for å se video/illustrasjonsbilde av hvordan øvelsen utføres. Man kan for eksempel få mer informasjon om pullups A og B og hva denne benevningen betyr.



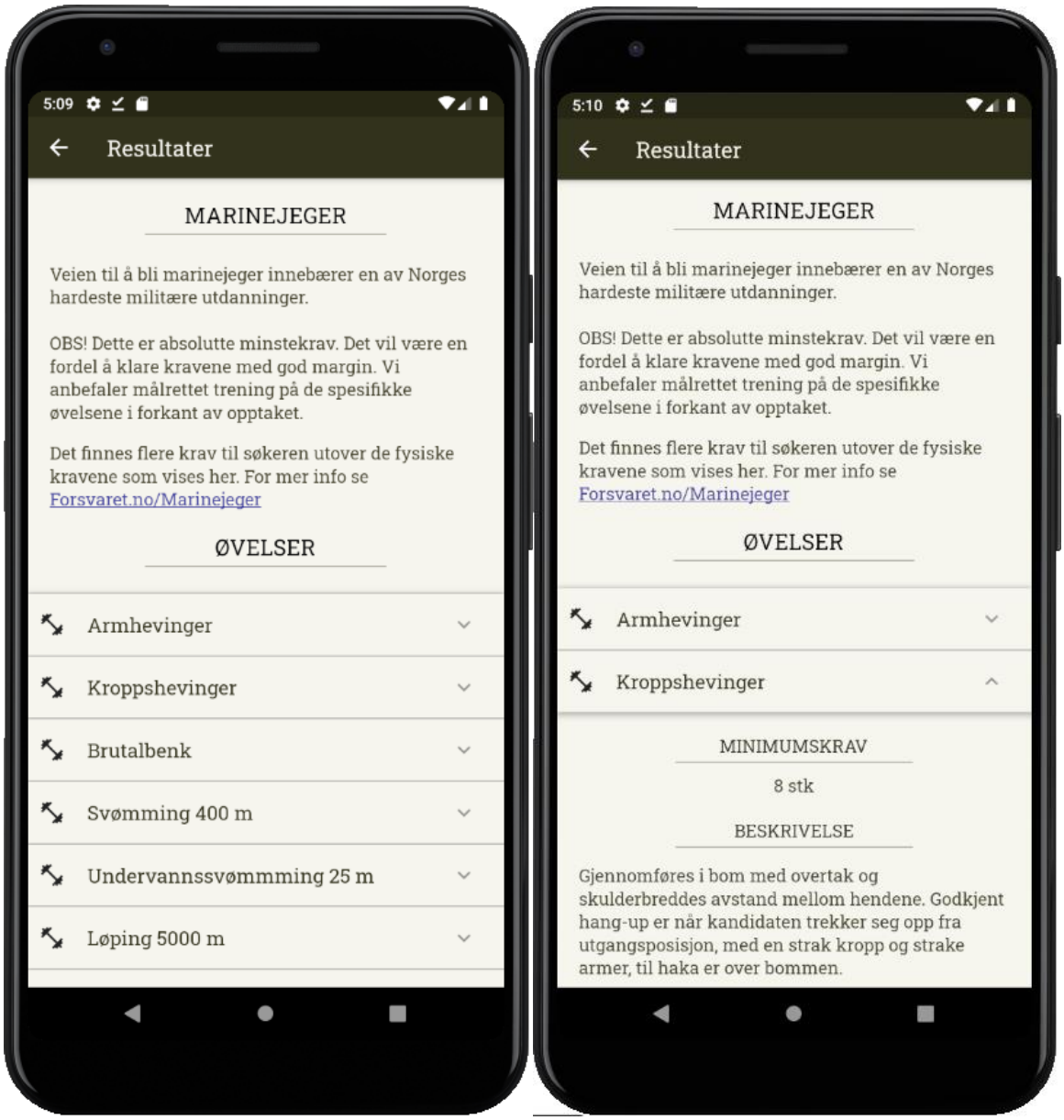
Figur 3.8: Popup-vindu som viser mer informasjon om en gitt øvelse

De valgbare utholdenhetsøvelsene for årlige tester har kun minstekrav opp til 3. Dette vil si at alle som trenger høyere minstekrav i sin stilling enn 4, ikke har mulighet til å velge disse. Det finnes heller ikke video eller bilderessurser for disse øvelsene og dermed får man kun mulighet til å se mer info om øvelsen i dette tilfellet. Når man trykker på knappen "Mer info" får man se beskrivelsen av øvelsen (Figur 3.8, høyre bilde).



Figur 3.9: Popup-vindu med videoer eller bilder som beskriver den gitte øvelsen

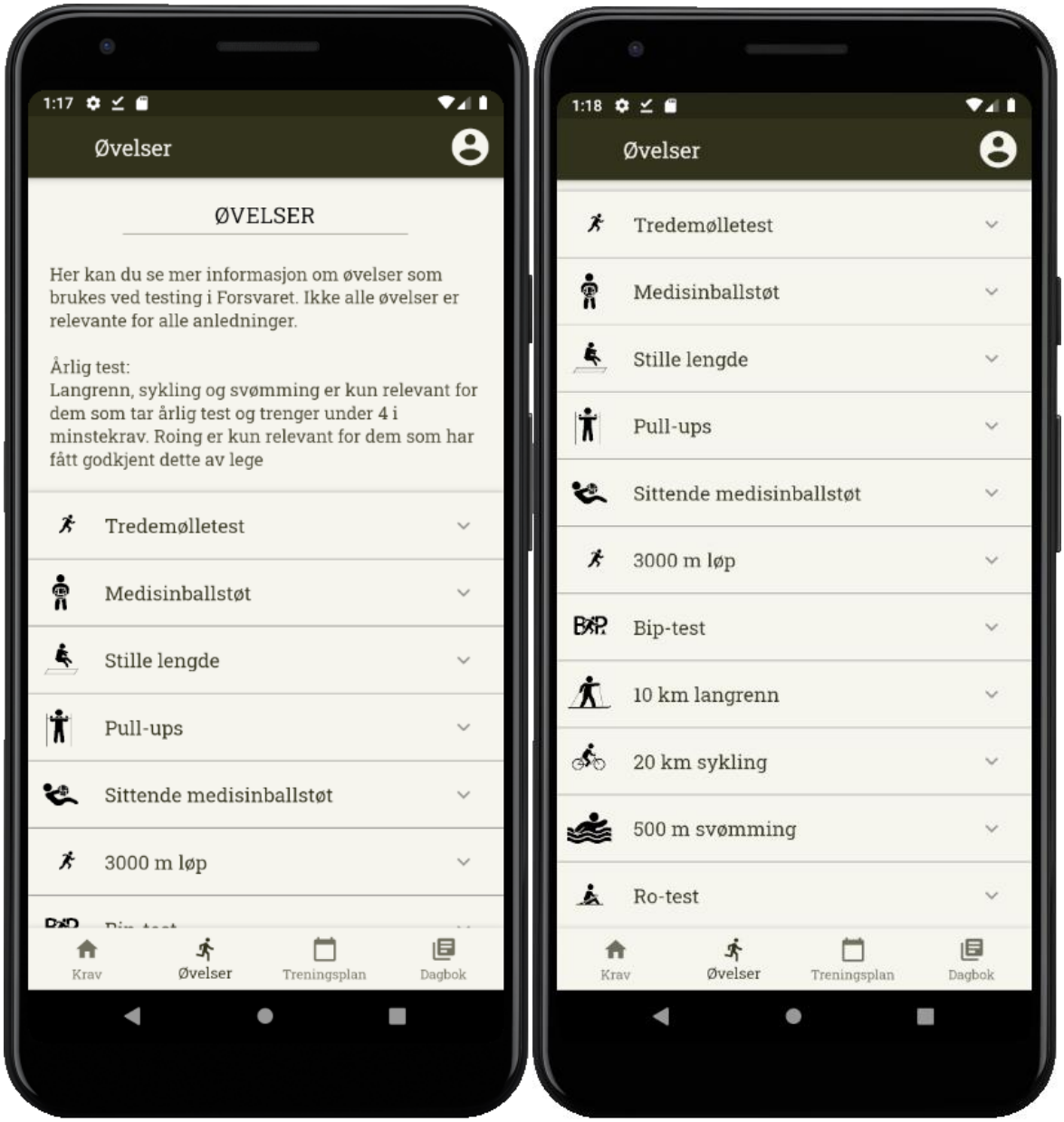
Trykker man på “Se øvelsen” får man et popup-vindu med et videoklipp eller et bilde av hvordan øvelsen skal utføres (Figur 3.9).



Figur 3.10: Søkeresultat for søkbare førstegangstjenester, med lukket og åpnet ekspanderbart panel.

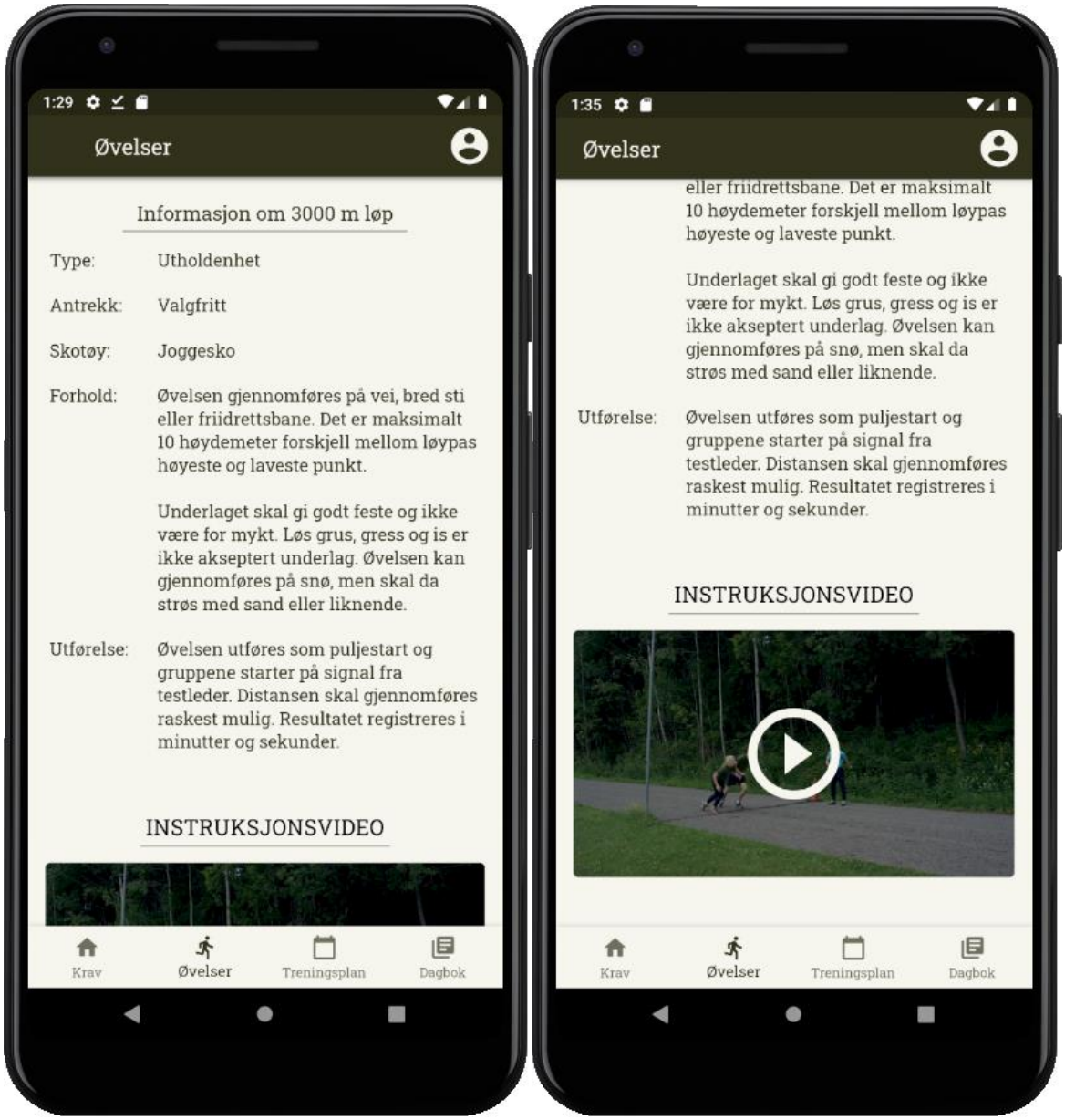
Et søk for en søkbar førstegangstjeneste ser litt annerledes ut enn andre søk (Figur 3.10). Den øverste delen består fortsatt av en beskrivelse og link til forsvarets sider for å lese mer om anledningen, men øvelsene for disse anledningene har ikke flere minstekrav. I stedet vises hver øvelses enkelte minstekrav, og en kort beskrivelse av hvordan den skal utføres.

Øvelsesside



Figur 3.11: Utsnitt av oversikten over øvelser fra øvelsessiden.

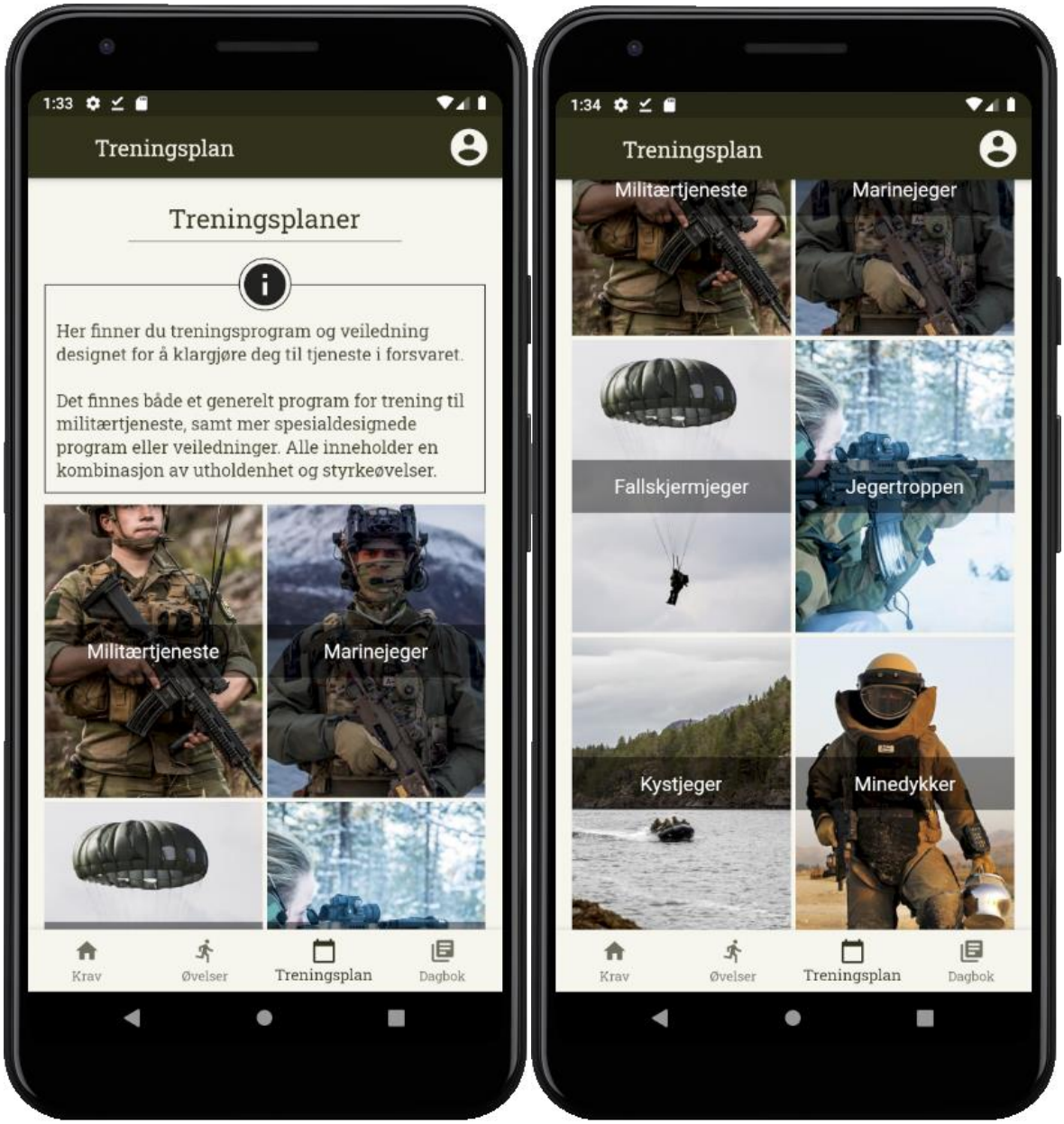
Øvelsessiden er en oversikt over alle øvelser som er å finne i “Reglement for fysisk testing” (Figur 3.11). Den øverste seksjonen gir tilleggsopplysninger om noen av øvelsene, mens den nederste seksjonen er, i likhet med resultatene til et kravsøk, en liste med ekspanderbare panel.



Figur 3.12: Detaljert informasjon om en gitt øvelse.

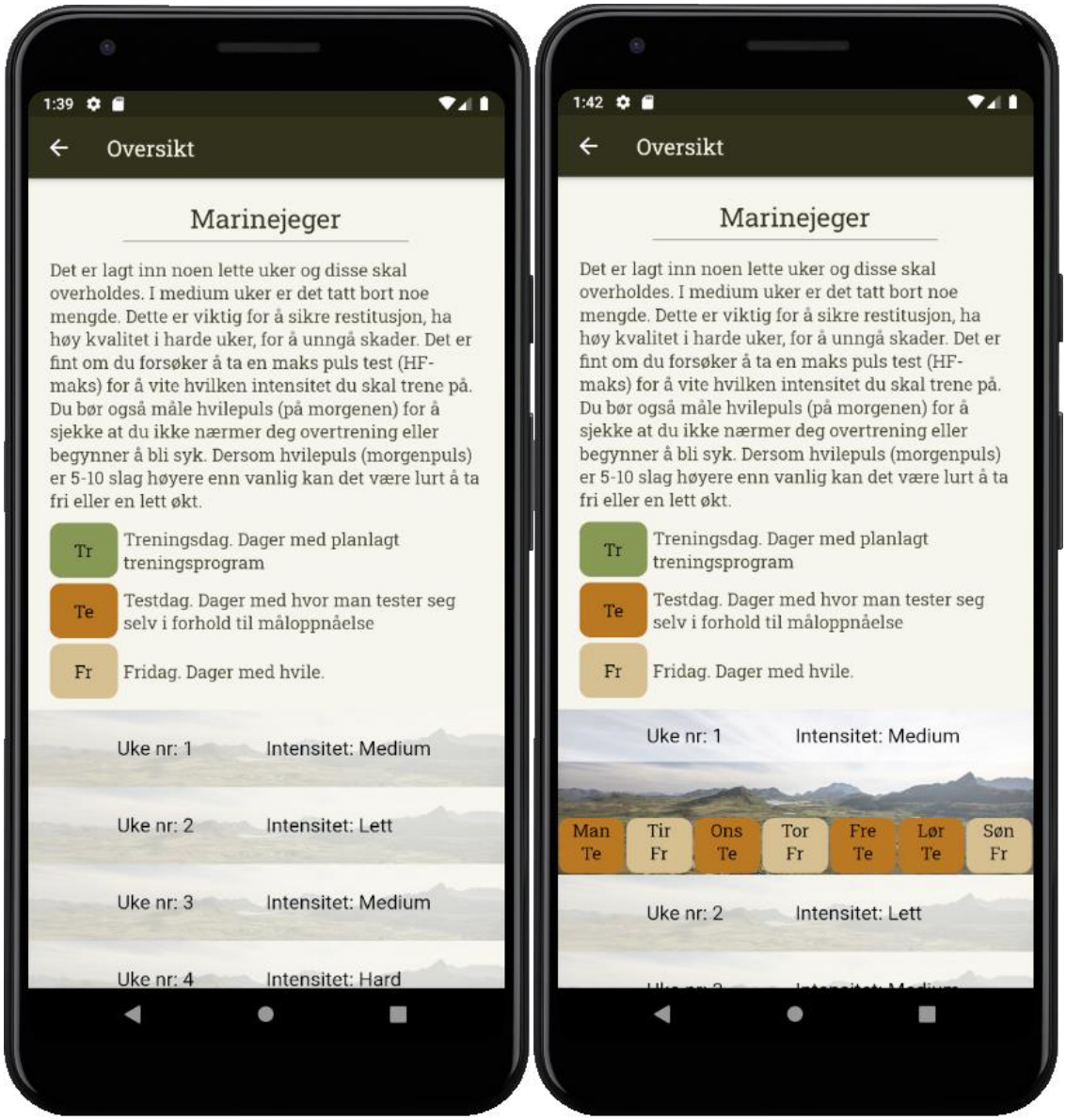
Når en øvelse ekspanderes vises det en detaljert beskrivelse av øvelsen, i tillegg til praktiske og tekniske retningslinjer for utførelse (Figur 3.12). For de øvelsene der det finnes tilgjengelige ressurser, vises det også et bilde eller en video.

Treningsplanside



Figur 3.13: Treningsplansiden med oversikt over ulike treningsprogram.

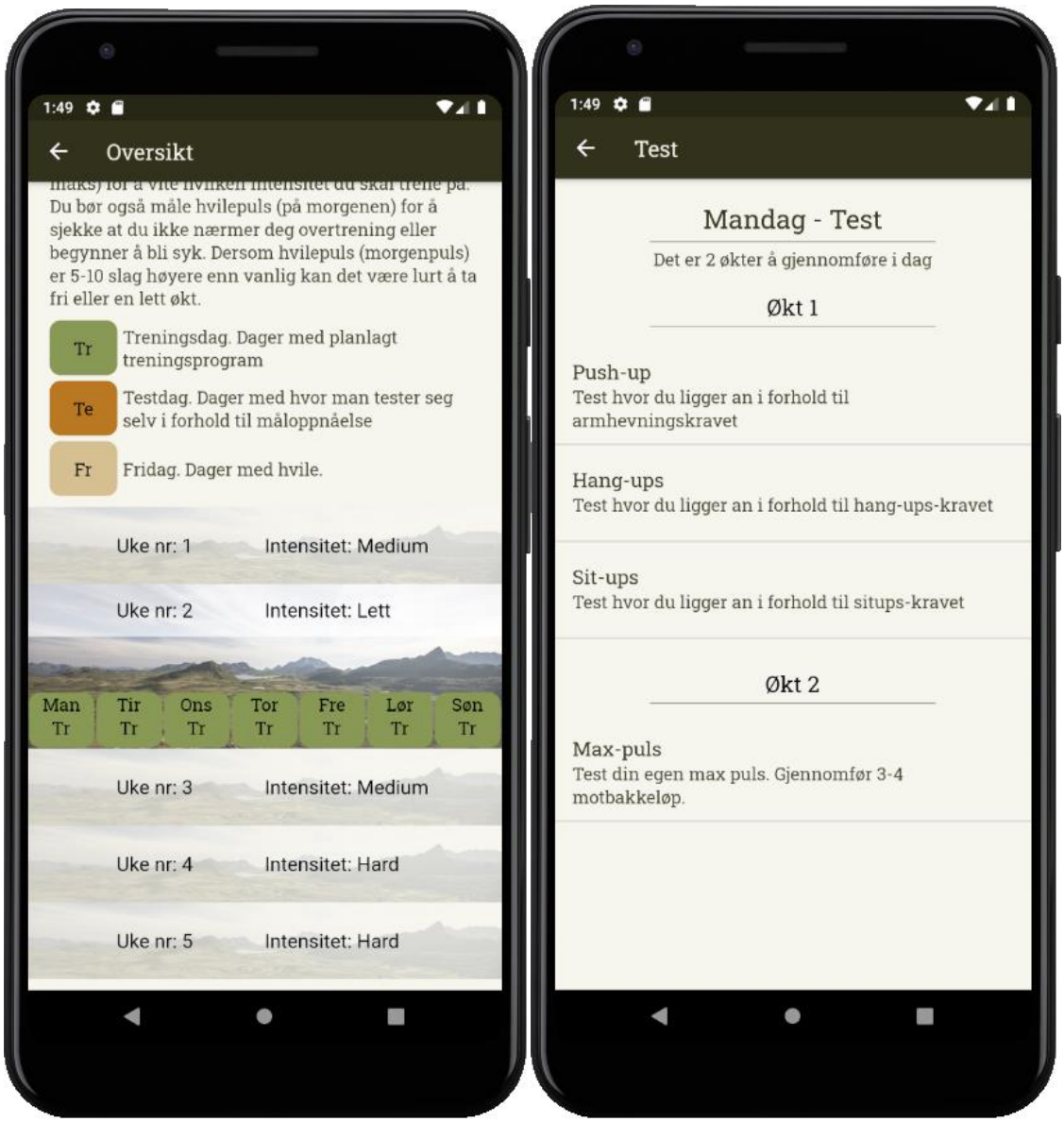
Forsvarets nettsider har flere treningsprogram tilgjengelige for å forberede rekrutter til generell militærtjeneste eller de søkbare førstegangstjenestene. Disse programmene er direkte implementert i appen slik at det skal være enkelt å slå opp hvilken tjeneste man vil forberede seg til, og følge et treningsprogram utarbeidet for den relevante tjenesten (Figur 3.13).



Figur 3.14: Mer detaljert informasjon om et spesifikt treningsprogram

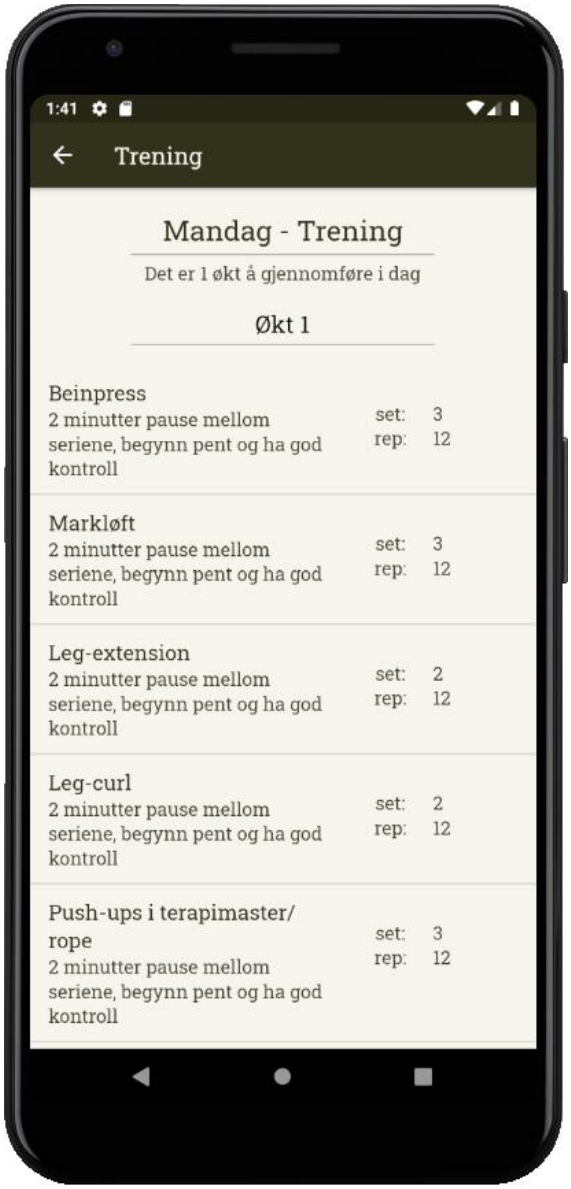
Når et treningsprogram åpnes (Figur 3.14) vises en beskrivelse av programmet øverst, og under ligger en beskrivelse av strukturen programmet følger. Alle treningsprogrammene er delt opp i treningsdager, testdager og fridager. Disse har fått hver sin farge for å lettere kunne se hvordan en treningsuke ser ut, men er også markert med bokstaver for å hjelpe de som ikke klarer å skille mellom farger.

Under beskrivelsen ligger ukene listet opp med ukenummer og intensitetsgrad. Disse ukene er ekspanderbare, og når de klikkes på får man opp syv fargekodede dager. Både testdager og treningsdager er klikkbare.



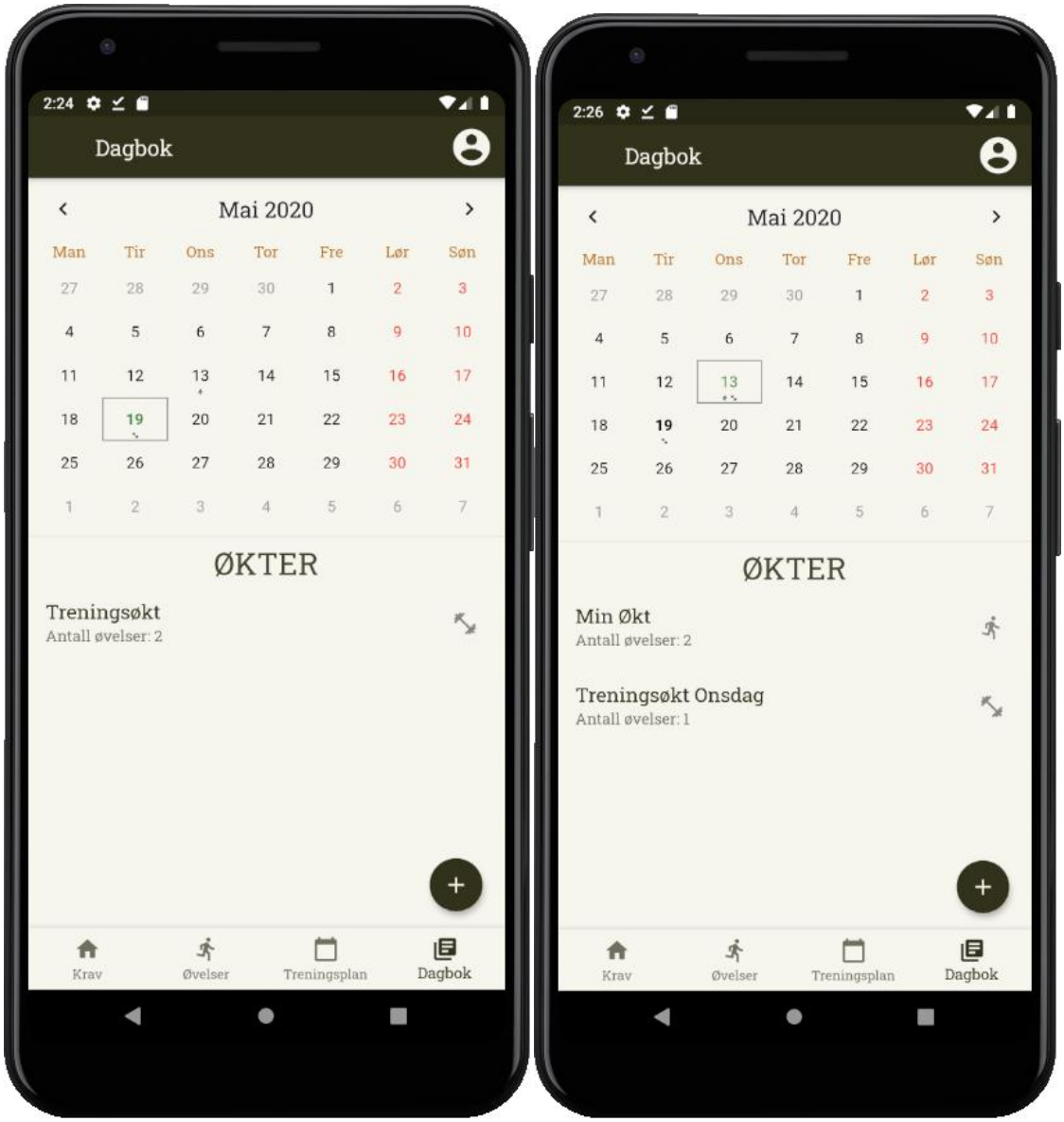
Figur 3.15: Detaljert informasjon om den planlagte treningen i et gitt treningsprogram. I dette tilfellet har brukeren valgt en testdag.

Ved å klikke på en dag vil en ny side åpnes (Figur 3.15). Her vises hvilken dag det er, om det er trening- eller testdag, og hvor mange økter det er den dagen. Deretter vises hvilke økter som skal gjennomføres, og øvelsene for hver økt. Både utholdenhetsøvelsene og styrkeøvelsene har tydelige mål når det kommer til tid, distanse, vekt, repetisjoner og sett (Figur 3.16).



Figur 3.16: Detaljert informasjon om den planlagte treningen for et gitt treningsprogram. I dette tilfellet har brukeren valgt en treningsdag.

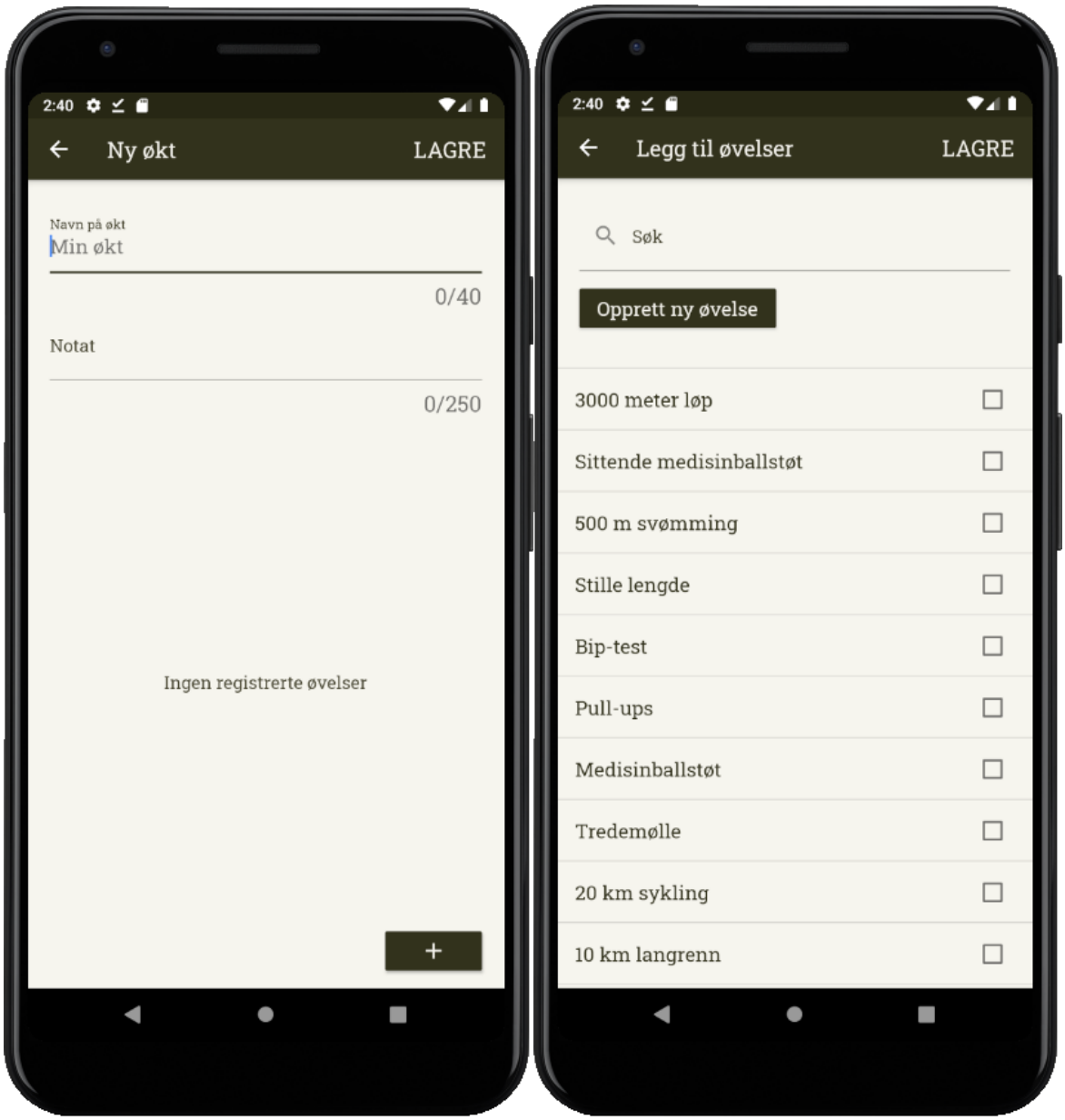
Treningsdagbok



Figur 3.17: Kalenderoversikt i treningsdagboken der antall økter for en gitt dag vises.

Tilgang til treningsdagboken krever at brukeren har opprettet en profil og er logget inn. Dette er nødvendig for at dataene skal kunne lagres til en spesifikk bruker i den eksterne databasen, og kunne hentes ned igjen til hvilken som helst enhet hvor appen er installert og brukeren er logget inn. På forsiden av dagboken får man se en kalender med ikoner på de dagene hvor økter er registrert (Figur 3.17). Ikonene viser en løpende mann om det er en utholdenhetsøkt og en vekt om det er en styrkeøkt. Om økten er en styrke eller utholdenhetsøkt kommer an på hvilken type øvelse det

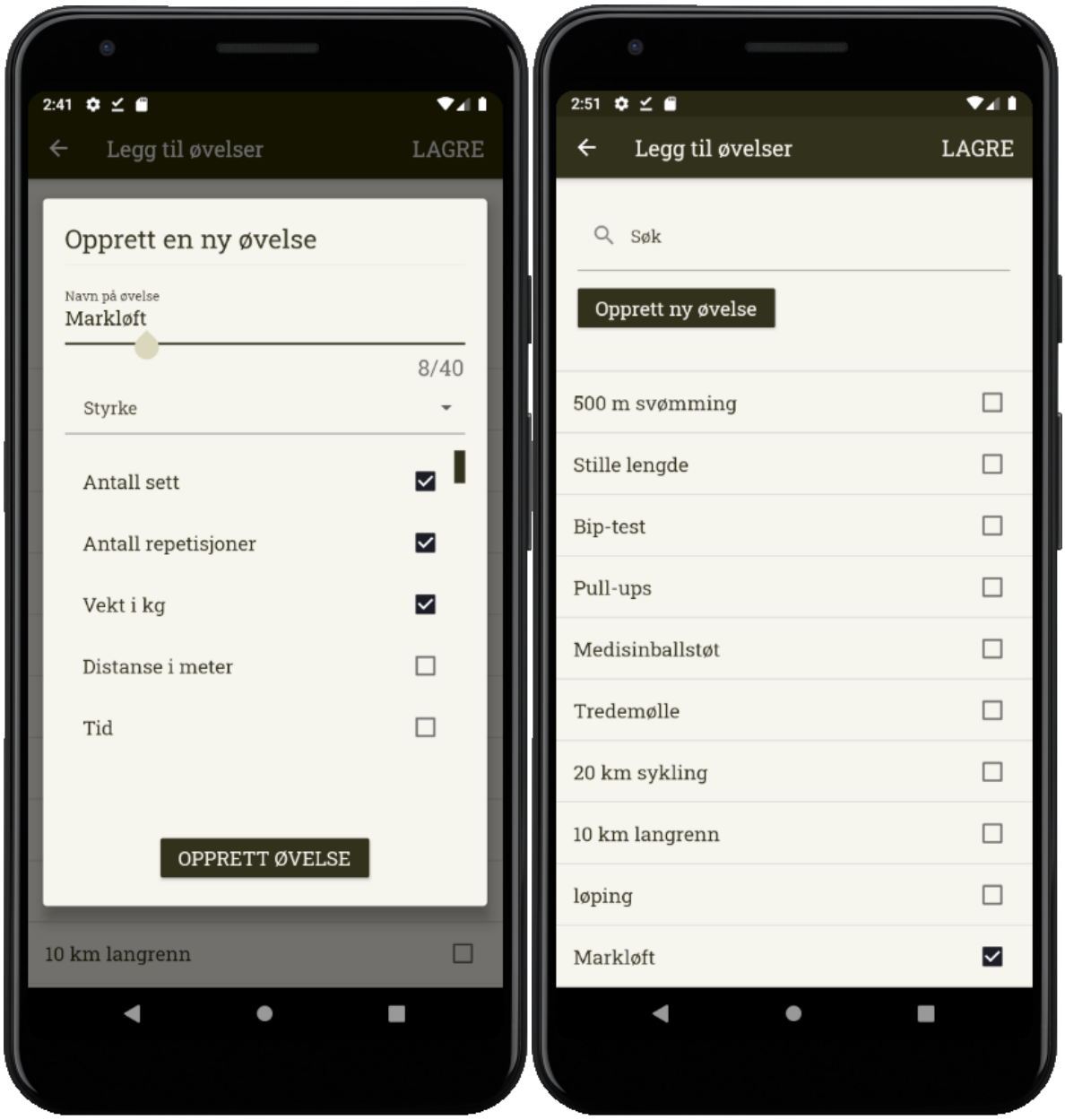
er flest av i økten. Om økten markeres som fullført vil ikonet i kalenderen vises i grønn farge.



Figur 3.18: Skjerm for å opprette en ny økt, og skjerm for å legge øvelser til en økt.

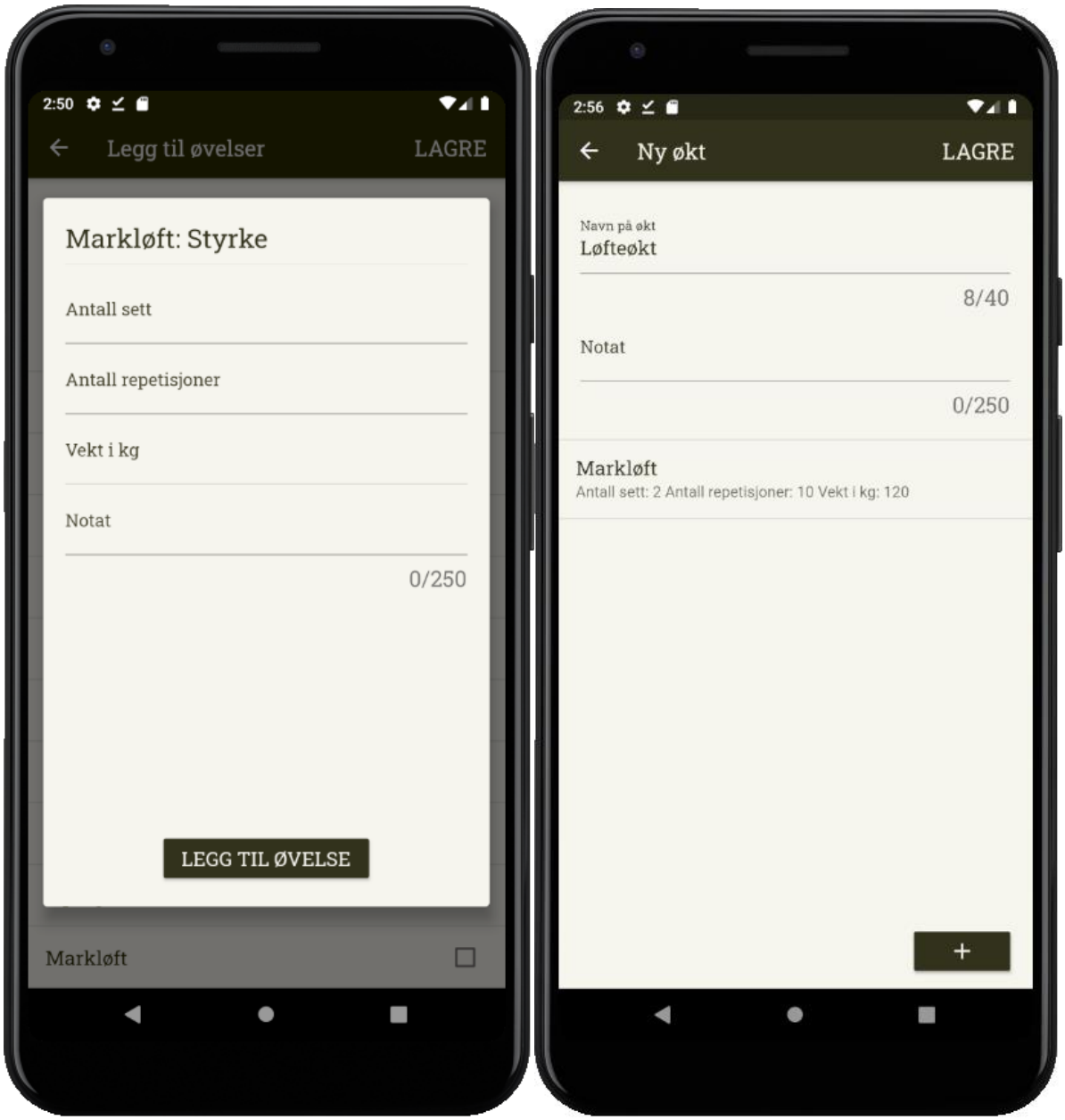
Ved å trykke på pluss-knappen i nedre høyre hjørne (Figur 3.17) åpnes en ny side der brukeren kan planlegge en ny treningsøkt. For hver økt kan brukeren legge til øvelser ved å klikke på pluss-knappen i nedre høyre hjørne (Figur 3.18, venstre bilde). Også her åpnes en ny side, hvor man kan velge øvelser fra en liste over tilgjengelige øvelser (Figur 3.18, høyre bilde). Listen består av de predefinerte øvelsene som ligger felles for alle brukere i databasen, og de øvelsene brukeren

selv har opprettet slik at de skal være enkle å gjenbruke. De predefinerte øvelsene er forsvarsøvelser som kommer fra “Reglementet for fysisk trening”.



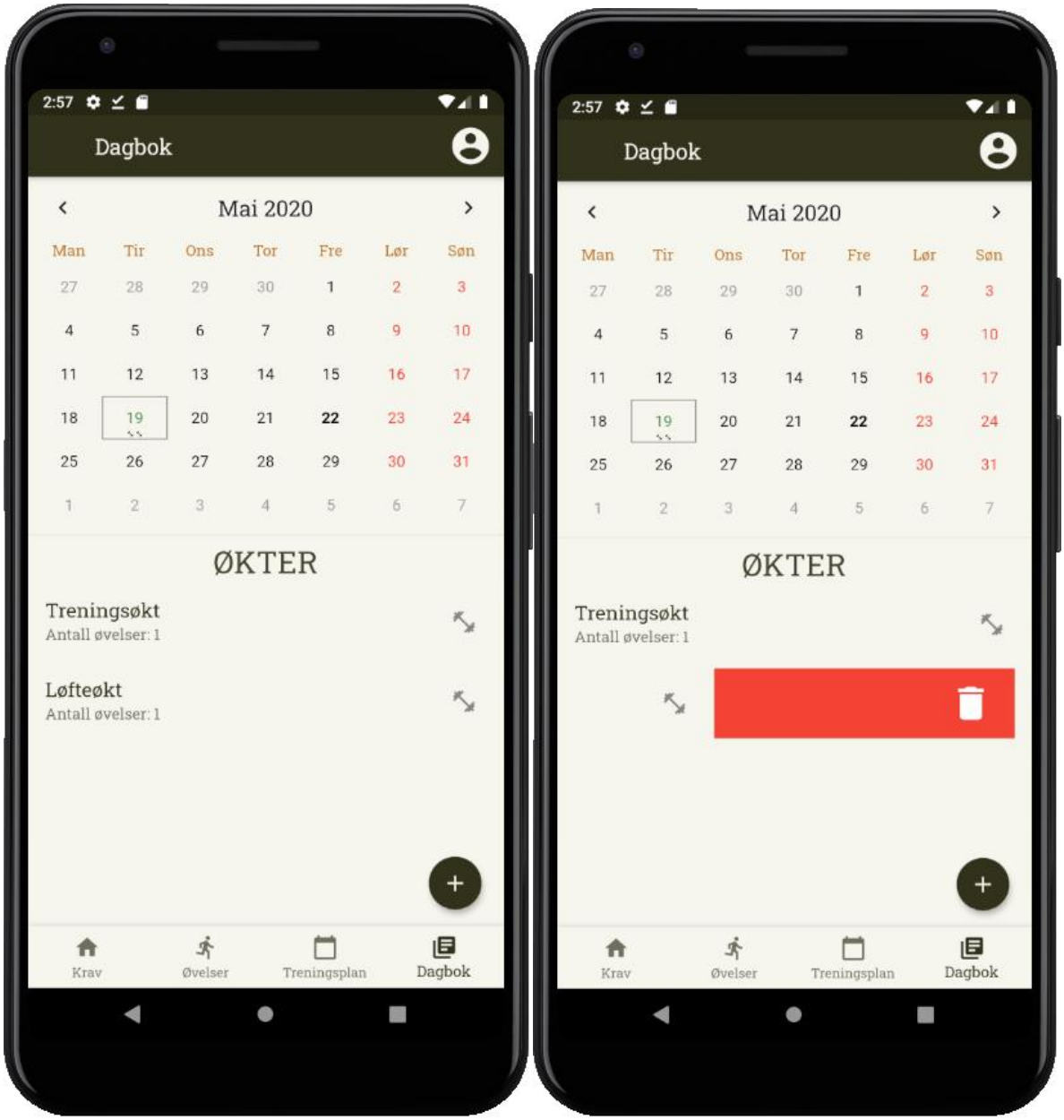
Figur 3.19: Skjema for å opprette en ny, egendefinert øvelse

Når man oppretter en ny øvelse må man fylle inn navn, type øvelse og velge hvilke parametre som skal høre til den nye øvelsen (Figur 3.19). Dette kan for eksempel være sett, repetisjoner eller tid. Når en ny øvelse er opprettet finnes den i listen over øvelser som man kan legge til en økt.



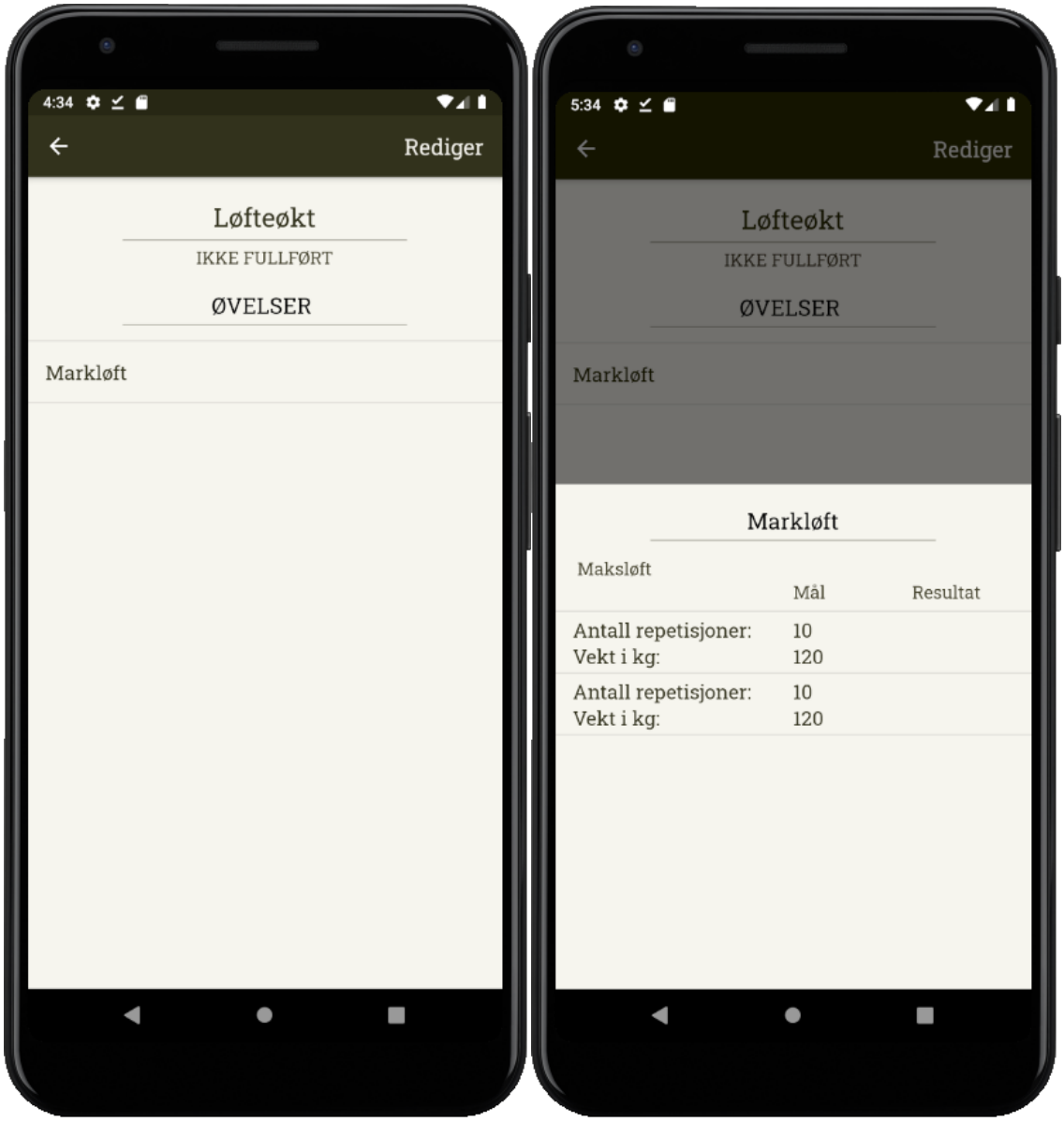
Figur 3.20: Skjema for å legge en øvelse til en økt, og skjemaet man fyller ut for å navngi økten.

Når man ønsker å legge øvelser til en økt velger man en øvelse fra listen og fyller ut målene sine (Figur 3.20). Øvelsen vil så bli lagt til i økten.



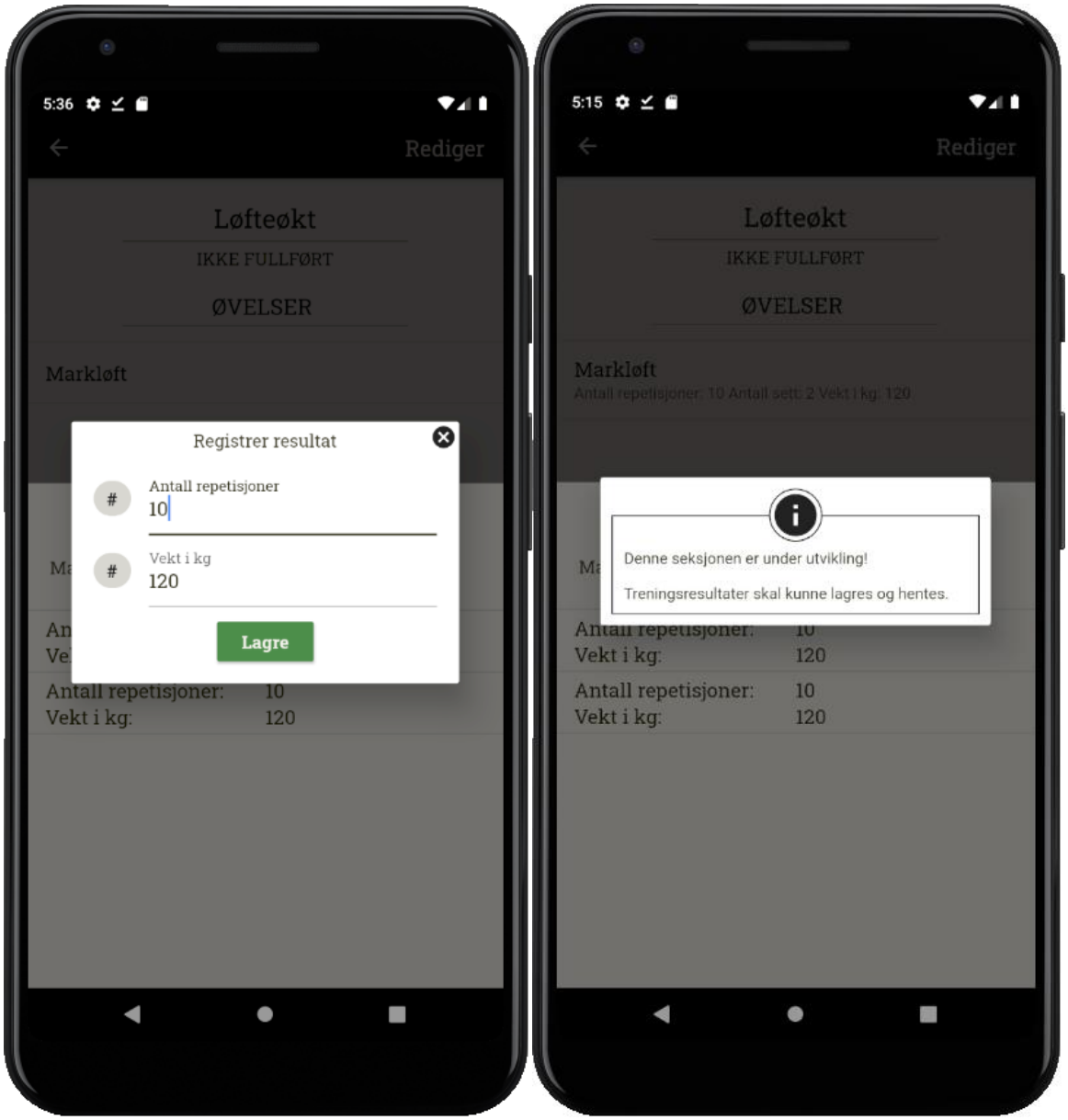
Figur 3.21: Kalenderoversikt i treningsdagboken der antall øvelser for en gitt dag vises. Ved å sveipe til venstre kan øvelsen slettes.

Når økten lagres, vil den dukke opp på øktoversikten på forsiden av dagboken under den dagen den ble opprettet på. Fra denne oversikten kan man slette økter man ikke lenger har behov for ved å sveipe til venstre (Figur 3.21). Før øvelsen faktisk slettes vil man få en popup-meny der man må bekrefte handlingen, for å forhindre at man sletter en økt ved en feil.



Figur 3.22:Oversikt over en planlagt økt med spesifiserte mål og resultater for ulike øvelser.

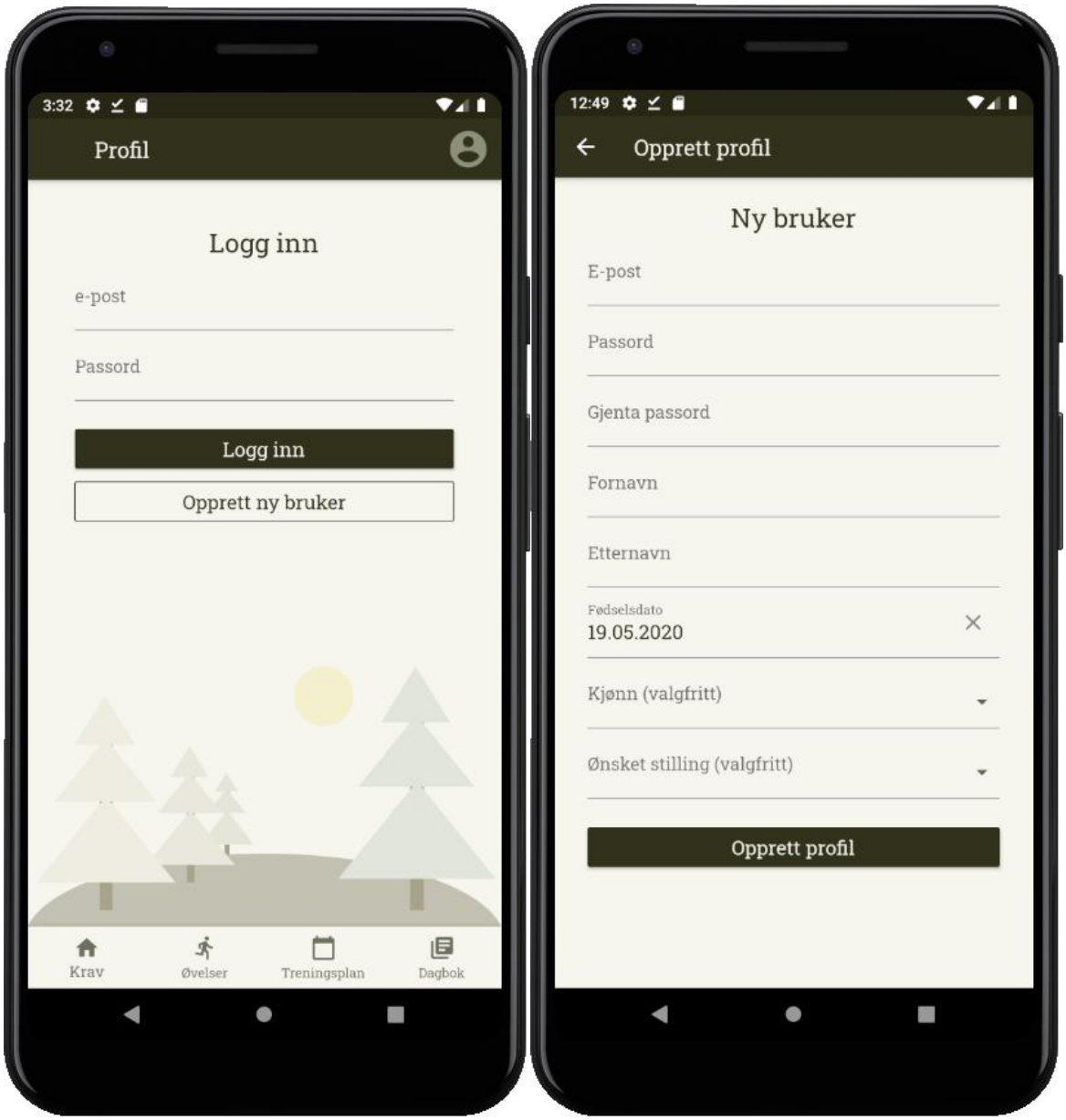
Når en bruker ønsker å registrere et resultat for en planlagt økt klikker man på økten i øktoversikten på forsiden. Man vil få opp en side som viser treningsøkten og hvorvidt den er fullført eller ikke (Figur 3.22). Notatet for økten og hvilke øvelser som er registrert vises. Klikker man på en øvelse vil man få en oversikt over målene som er satt. Verdien brukeren har fylt inn i “Sett”-parameteren bestemmer hvor mange runder brukeren ønsket å gjennomføre øvelsen.



Figur 3.23: Popup-vindu med et skjema for å fylle inn resultater for en øvelse, og popup-vinduet som informerer om at lagrings-funksjonaliteten ikke er implementert.

Her kan man igjen trykke på hvert mål for å fylle ut sine resultater og lagre dem til øvelsen (Figur 3.23). Inputfeltene er fylt ut med målene man har satt seg og kan endres etter hvor mange man klarte å gjennomføre. Lagring av resultater er ennå ikke implementert. Inntil videre er det laget en infoboks som vises når man trykker på “Lagre”, som informerer om at denne delen av appen fortsatt er under utvikling.

Autentisering og profil



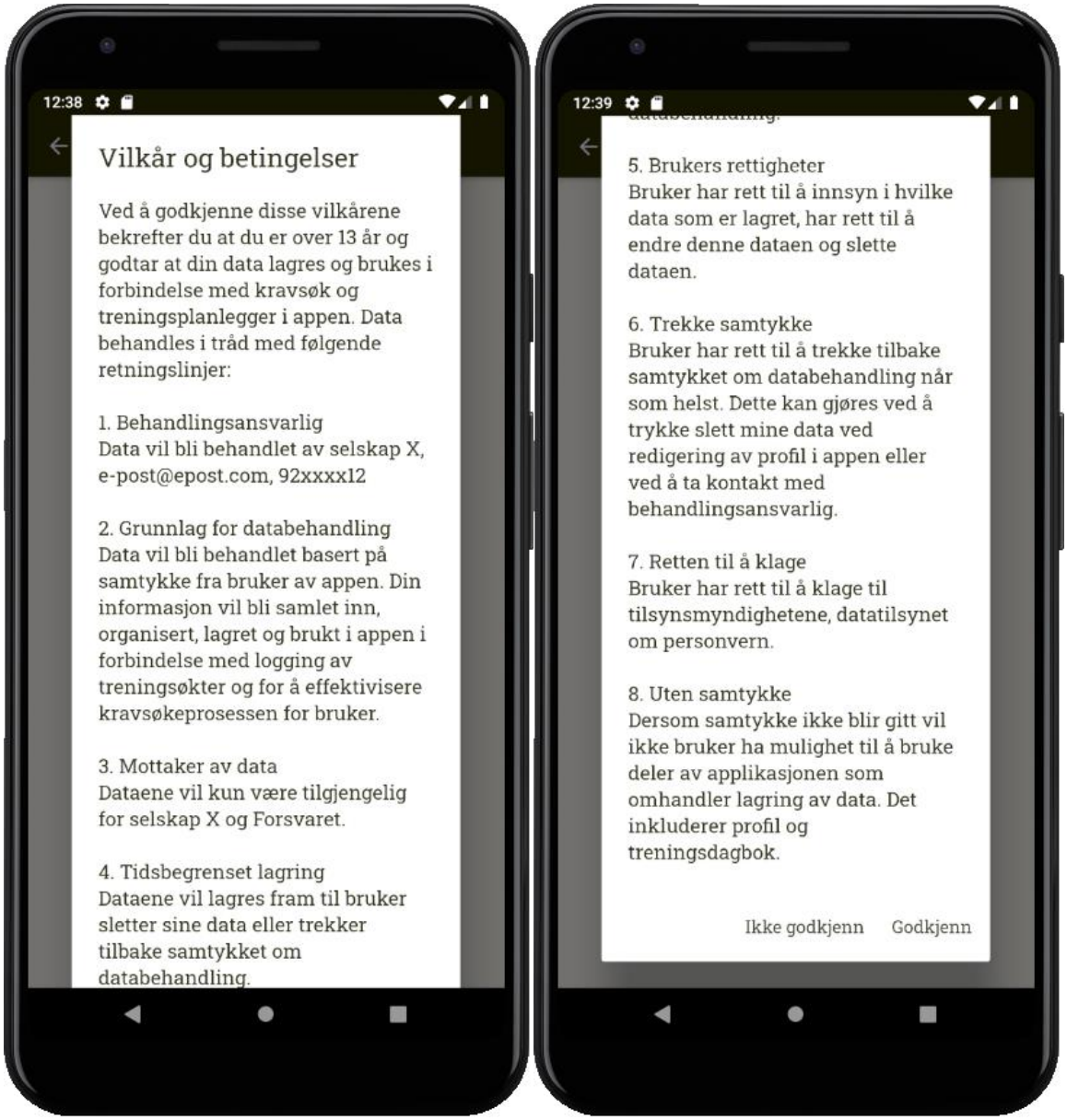
Figur 3.24: Logg inn-skjema, og skjemaet som skal fylles ut når man ønsker å opprette ny profil.

Når man trykker på profil-ikonet øverst til høyre, eller prøver å gå til dagboken uten å være logget inn, blir man møtt med et logg inn-skjema (Figur 3.24, venstre bilde). Fra logg inn-skjemaet kan man fylle ut brukernavn og passord, som går gjennom en autentiseringsprosess når man prøver å logge inn. Alternativt kan man opprette en ny bruker ved klikke på “Opprett ny bruker”. Feltene i skjemaet blir validert ved hjelp av regulære uttrykk. Blant annet kan et fornavn kun inneholde bokstaver,

mellomrom, bindestrek og punktum. Valgt fødselsdato krever at man er minst 13 år gammel.

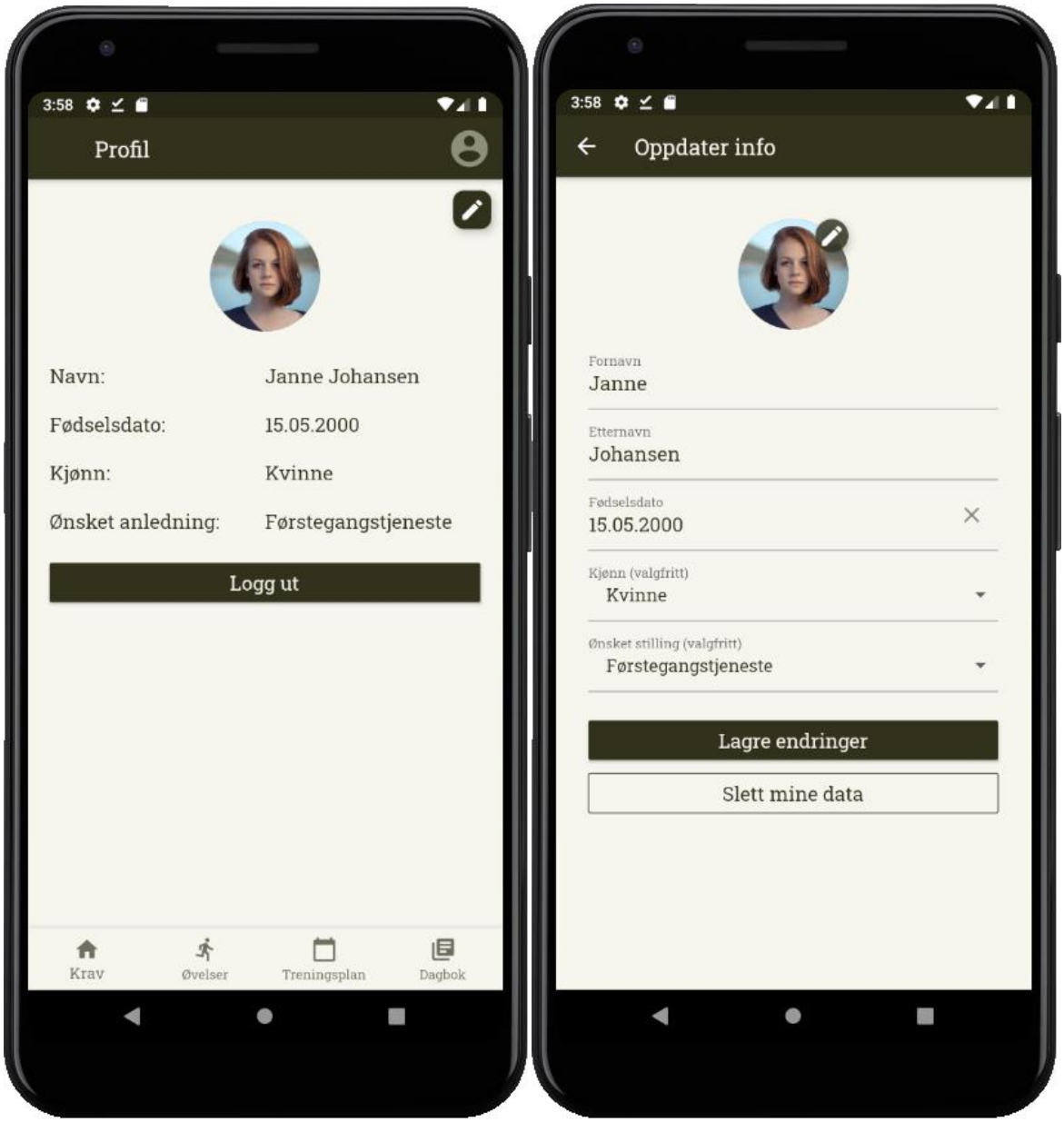
Figur 3.25: Kalender-*widgeten* som dukker opp når man velger fødselsdato.

Fødselsdato velges i en kalenderwidget (Figur 3.25), mens kjønn og ønsket anledning velges ved hjelp av nedtrekkslister. Det er valgfritt om man vil lagre kjønn og anledning til profilen sin. Om man gjør det, vil søkeprosessen for krav bli effektivisert ved at forsiden vil ha prepopulerte felter i filteret basert på hva man har registrert i sin profil.



Figur 3.26: Popup-vindu med vilkår som må godkjennes for at en profil skal bli opprettet.

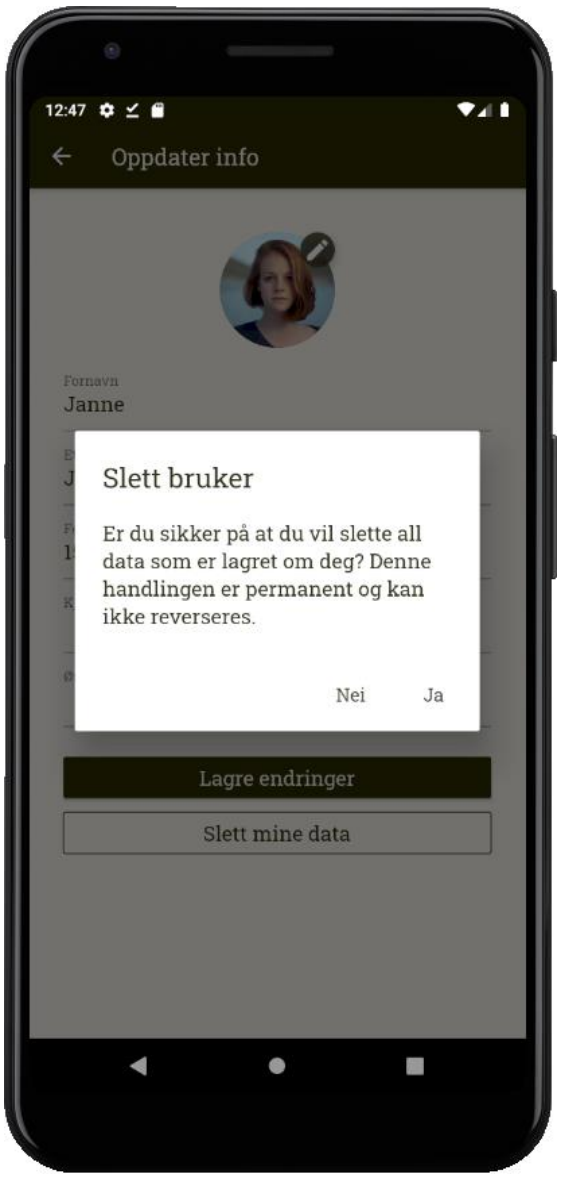
Når registreringsskjemaet er ferdig utfylt og godkjent kan man klikke på “Opprett profil”. Da vil et popup-vindu dukke opp med vilkår brukeren må godkjenne som ber om tillatelse til å behandle brukerens informasjon, i samsvar med GDPR (Figur 3.26). På bunnen av vilkårene får man valget om man ønsker å godkjenne betingelsene eller ikke. Dersom brukeren ikke godkjenner vil ingen profil bli opprettet, og man blir tatt tilbake til skjemaet for å ikke miste verdiene i inputfeltene. Om man godkjenner betingelsene, blir profilen opprettet i databasen og man tatt videre til profilsiden.



Figur 3.27: Profilinformasjon for en innlogget bruker, og skjemaet der man kan endre profilinformasjon.

På profilsiden kan man se informasjonen som er registrert, med unntak av autentiseringsinformasjonen. Fra denne siden kan man logge ut, noe som sender en tilbake til logg inn-skjermen, eller man kan redigere profilinformasjonen ved å trykke på ikonet av en penn i øvre høyre hjørne (Figur 3.27, venstre bilde). Skjemaet der man redigerer profilinformasjonen er gjenbrukt fra opprettingen av en profil, med noen distinkte endringer. Når man endrer en profil har man mulighet til å legge til / endre profilbilde, og man får ikke opp autentiseringsinformasjonen. Man har også

mulighet til å slette all brukerinformasjon. Dette vil slette all autentiseringsinfo, persondata, treningsdata og bilde. Før alt blir slettet får man opp et popup-vindu som krever bekreftelse på at man virkelig vil slette alt(Figur 3.28).



Figur 3.28: Popup-vindu med spørsmål om brukeren ønsker å slette profilen sin for godt.

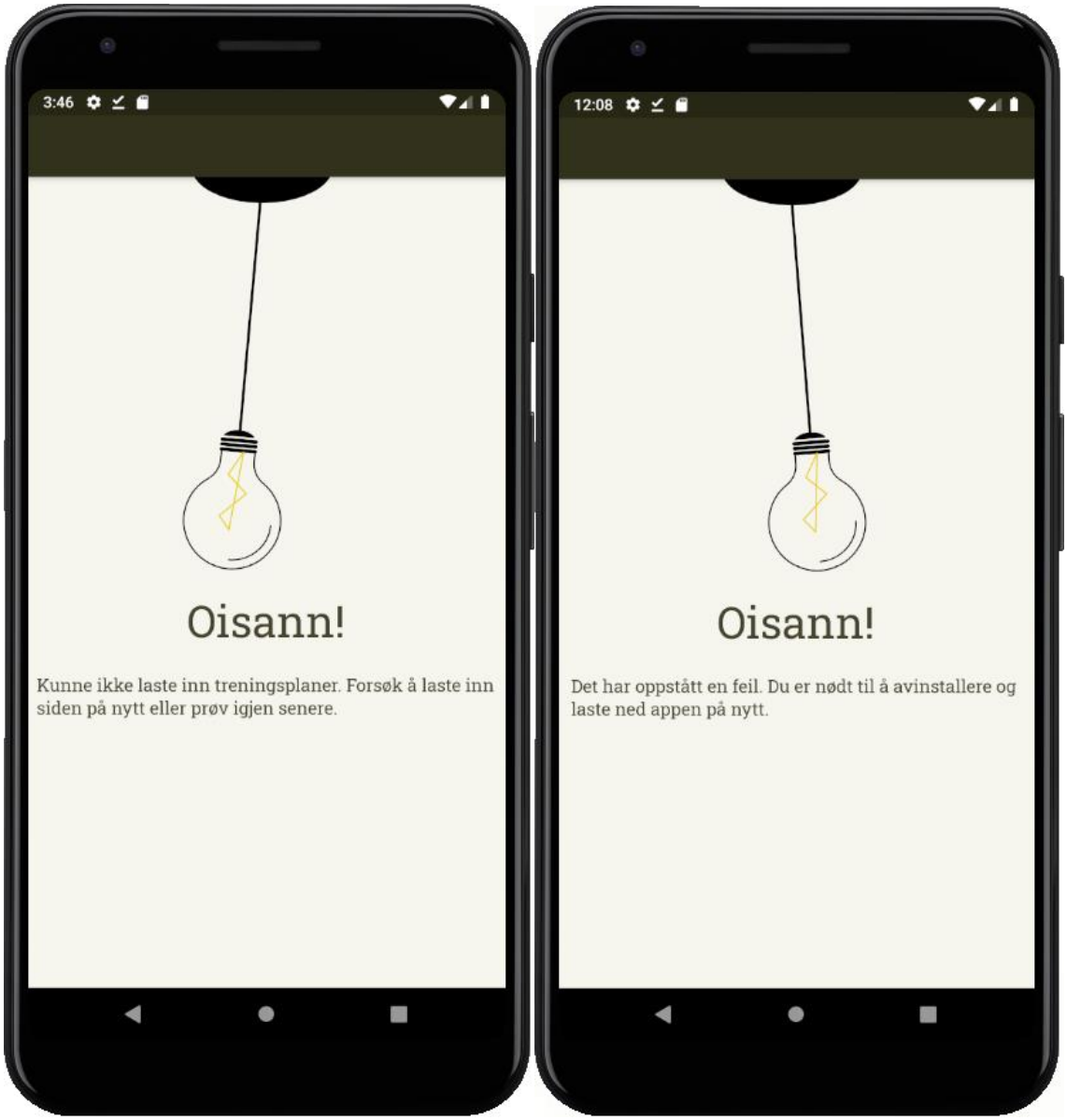
Feilhåndtering

Det er tre kategorier av feilhåndtering i appen:

1. For feil som oppstår på bakgrunn av at brukeren ikke skriver inn korrekt informasjon i et input-felt, vil det vises en rød tekst ved det eller de gjeldende feltene som ikke er godkjent.

2. For feil ved nettverksforespørlser, vil det vises feil og feilmelding i et eget popup-vindu som forteller brukeren hva som er galt.
3. For feil ved lokal databasetilknytning eller andre app-avhengige feil fra for eksempel kode-*bugs*, vil det vises en egen skjerm med forklarende beskjed til bruker og hva som er galt (Figur 3.29).

Ved oppstart av appen kjøres det en rekke initialiserings-prosedyrer, blant annet versjonssjekk av den lokale databasen som avgjør om den skal oppdateres eller ikke, og en sjekk av brukerdefinerte innstillinger. Hvis noen av disse prosedyrene krasjer vil ikke hoved-appen startes opp, men en error-app vil kjøres i stedet. Denne vil be brukeren om å laste ned appen på nytt. Ved slike tilfeller vil ingen brukerdata gå tapt, siden all slik data ligger lagret i Firebase og hentes ned ved pålogging. All forsvarsdata følger med appen ved nedlastning, så dette vil slettes og så lastes inn på nytt ved reinstallasjon.



Figur 3.29: Skjermbilde som viser en feilmeliding i appen, og en feilmelding i error-appen. Feilmeldingen indikerer at noe har gått virkelig galt. I appen vil lyspæren være animert og svinge fra side til side.

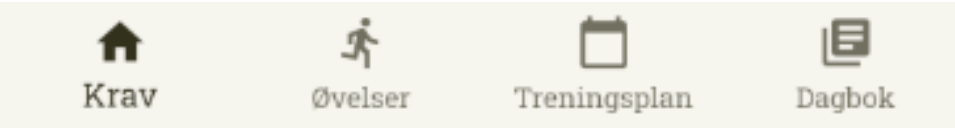
Denne skjermen brukes også til andre runtime errors i hoved-appen, for eksempel dersom treningsplaner ikke er mulig å hente fra databasen. Layouten består av en animasjon som er er laget med animasjonverktøyet Rive og inneholder feilmeldingstittel, forklarende tekst og eventuelle knapper for å rapportere feil.

Brukeropplevelse

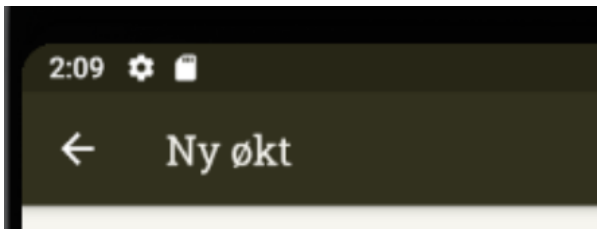
Brukeropplevelse omhandler opplevelsen en bruker har av et gitt produkt, spesielt med tanke på hvor enkelt produktet er å bruke.

Vi har gått inn for å øke brukeropplevelsen ved å ha en sterk informasjonsarkitektur, hovedsakelig i forhold til den logiske strukturen i appen og hvordan vi har organisert informasjonen. Det har vært viktig for oss at brukerne av appen ikke føler seg overveldet av for mange valg. De fire hovedfunksjonalitetene i appen representerer hver sin side i navigasjonbaren (Figur 3.30), og fra hver av disse sidene kan man bevege seg dypere ned i app-hierarkiet.

Den siden brukeren er på vil alltid være markert ved at det gjeldende elementet er uthevet i navigasjonsbaren (Figur 3.30). Dette, i likhet med tilbakepilene øverst i venstre hjørne (Figur 3.31), sier noe om hvor brukeren befinner seg, og gir en pekepinn på hvor man kan gå videre. Funksjonalitet som dette gir brukerne noe de kan støtte seg på, slik at de ikke går seg vill i hierarkiet



Figur 3.30: Skjermbilde av navigasjonsbaren i appen. I dette skjermbildet er Krav utvidet (mørkere farge og større tekst) for å indikere at brukeren er på denne siden, eller en underside av siden.



Figur 3.31: Skjermbilde av øverste venstre hjørne i appen. Denne tilbakeknappen dukker opp på visse steder, og indikerer at man kan gå tilbake til den foregående siden ved å klikke på den.

Det er viktig at tekst på knapper og merkelapper er forståelig og at teksten som sier noe om den bakenforliggende funksjonaliteten (for eksempel “logg ut”). Dette kan forhindre at brukeren foretar en handling de egentlig ikke ønsket å ta. I visse tilfeller,

som i popup-vinduene der det kreves en bekreftelse, blir man kun presentert med valgene “ja” og “nei” eller “godkjenn” og “ikke godkjenn”. Disse valgene sier ikke så mye i seg selv, men konteksten av spørsmålet vil hjelpe brukeren til å ta det rette valget.

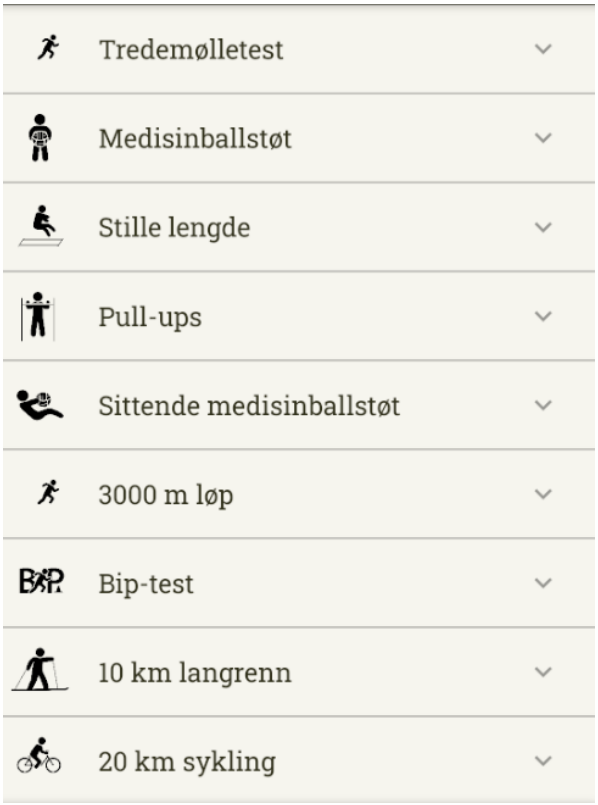
Et annet viktig aspekt for å øke brukeropplevelsen, er å tilrettelegge produktet for en størst mulig brukergruppe. Forsvarets minstekrav er knyttet til kjønn. Ettersom det kun finnes to alternativer i tabellene, avgrenset vi nedtrekksmenyene til å kun inneholde “kvinne” og “mann”. Man kan også velge å ikke ha kjønn lagret i sin profil, om man ikke ønsker det. Kvinne står forøvrig over mann i nedtrekksmenyen på bakgrunn av alfabetisk sortering.

Design

Produkteier ga oss tilnærmet full kunstnerisk frihet når det kom til hvordan appen skulle se ut. Vi benyttet oss av denne muligheten, men baserte likevel valgene våre på forskningslitteratur og etablerte konvensjoner. Vi tok også utgangspunkt i Forsvarets mediesenter sine retningslinjer for utforming av digitale medier.

Bilder og ikoner

Gjennomgående i appen har vi benyttet oss av ikoner, bilder og illustrasjoner for å skape et spennende visuelt uttrykk, og bryte med de store mengdene tekst appen inneholder. Ikonene våre er en blanding av ferdiglagde ikoner som følger med Flutter gjennom Material Library, og ikoner en medstudent var generøs nok til å tegne for oss (Figur 3.32). Bildene vi har brukt er i hovedsak tatt fra Forsvarets nettsider og mediearkiv, med unntak av bakgrunnsbildet (Figur 3.33) som vi laget selv i Adobe XD.

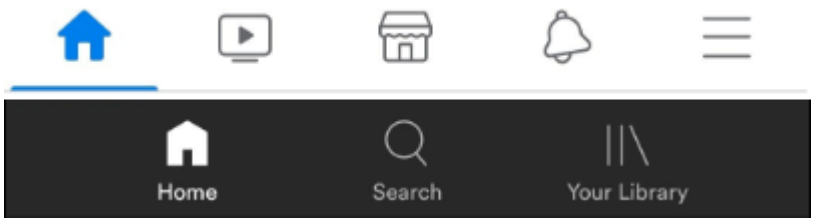


Figur 3.32: Et utsnitt av ikonene vi fikk av vår medstudent, presentert i en liste over øvelser i appen.

Figur 3.33: Bildet vi har benyttet som bakgrunnsbilde flere steder i appen. Det fremstiller en skog, både for å få et naturlig uttrykk, og fordi en skog er noe som kan assosieres med Forsvaret.

Ikonbruk er blitt en veletablert konvensjon i alle typer applikasjoner, både på web og håndholdte enheter. Et ikon med en forutbestemt betydning kan raskt bli kjent igjen og forstått, og vil ta mindre plass enn en tekstlig forklaring. Bruk av ikoner kan også

være til hjelp dersom en bruker har problemer med språket applikasjonen er presentert på. I noen tilfeller blir ikoner supplert med forklarende tekst, mens andre ganger står de alene og snakker for seg selv (Figur 3.34).

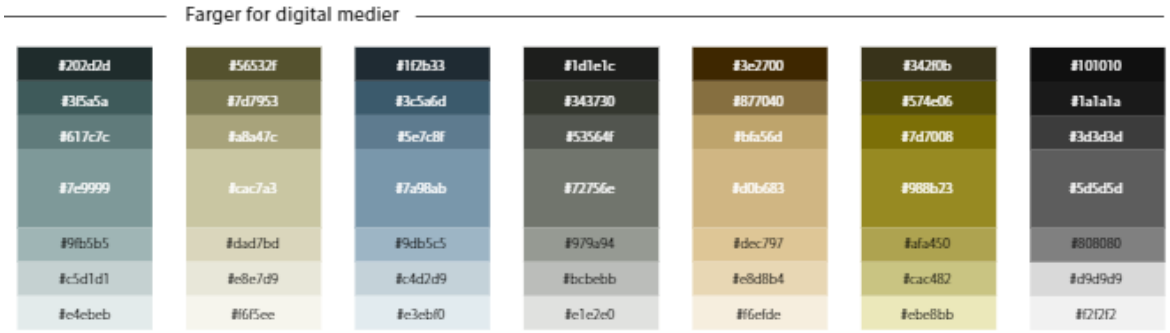


Figur 3.34: To navigasjonsmenyer, fra to forskjellige mobil-apper. Øverste meny tilhører Facebook, mens nederste meny tilhører Spotify.

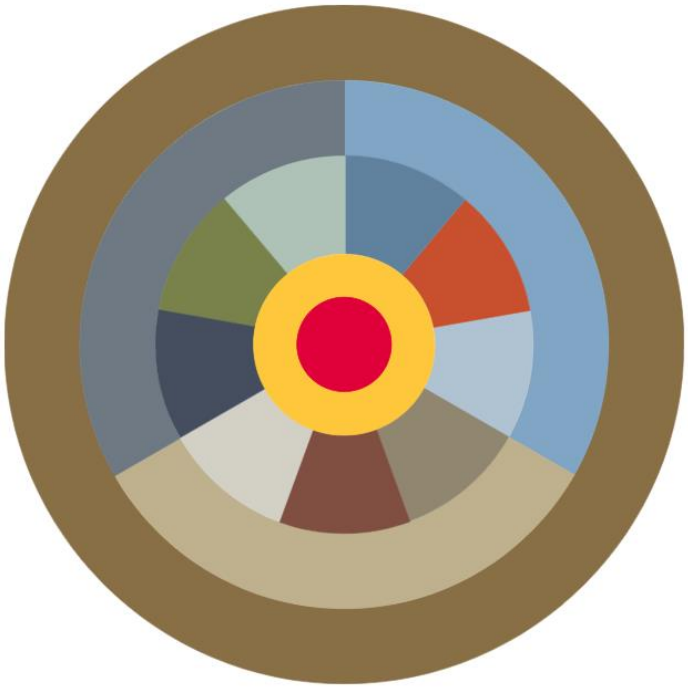
I vår app gikk vi for begge disse løsningene. Ikoner som hører til navigasjonsbaren har fått supplerende tekst som forklarer hvor du vil bli tatt hen (Figur 3.30), siden vi har benyttet oss av utradisjonelle ikoner uten etablerte betydninger. Vi benyttet oss av hus-ikonet, som inneforstått betyr “Hjem” / forside, nettopp fordi vi ønsket å formidle denne betydningen, selv om det ikke har noe med kravsøket å gjøre. Dette ble gjort på bakgrunn av brukertest 1, som blir presentert lenger ned i rapporten. Ikoner som i hovedsak er brukt som dekorasjon har derimot ingen tilhørende tekst, fordi de i seg selv ikke formidler viktig informasjon (eksempelvis løpe- og vektstang-ikonene foran øvelser).

Bruk av farger

Produkteier ønsket en “militær” fargepalett på appen, noe de spesifiserte som grønn eller mørk blå. Vi tok inspirasjon fra de eksisterende fargepalettene som brukes i Forsvarets digitale medier (Figur 3.35) og fargepaletten som brukes ved visuell kommunikasjon (Figur 3.36), og lagde våre egne paletter (Figur 3.37-3.39)

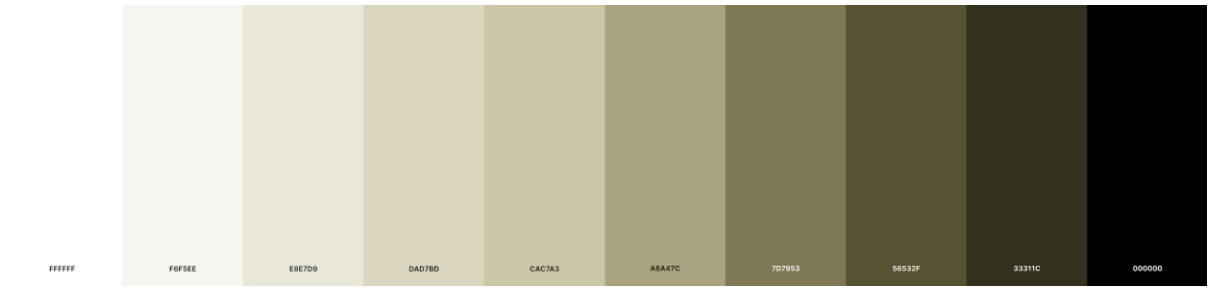


Figur 3.35: Fargepalettene Forsvaret bruker i deres digitale medier (Forsvaret, 2011, Forsvarets visuelle profil, side 18).

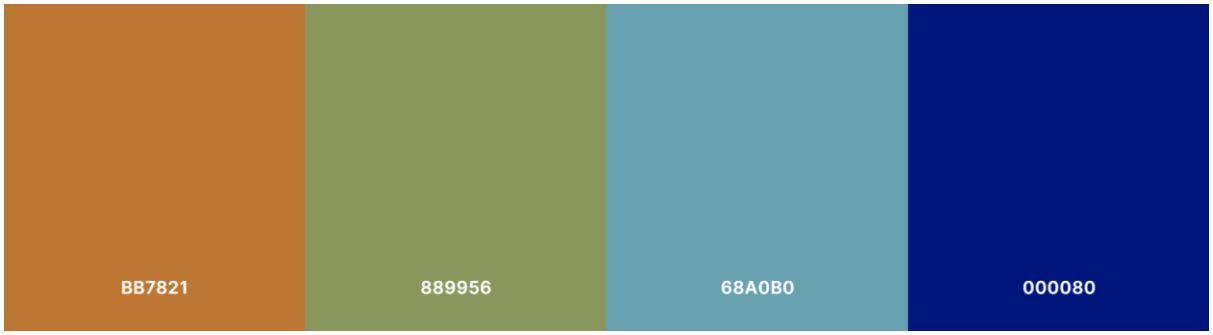


Figur 3.36: Fargepaletten som brukes ved visuell kommunikasjon i Forsvaret. Sirkelen er delt inn i en kjerne, tre sektorer og en ytre ring. De tre delene har hver sin symbolske betydning og bruksområde (Forsvaret, 2015, Farger)

Figur 3.37: Fargene brukt på varslinger i appen. Fra venstre til høyre: Suksess, Advarsel, Fare.



Figur 3.38: En monokrom grønn-brun gradient med farger som er blitt brukt gjennomløpende i appen. Ved noen anledninger har vi brukt andre varianter av disse fargene for å kunne oppnå riktig fargekontrast og overholde WCAG-kravene.



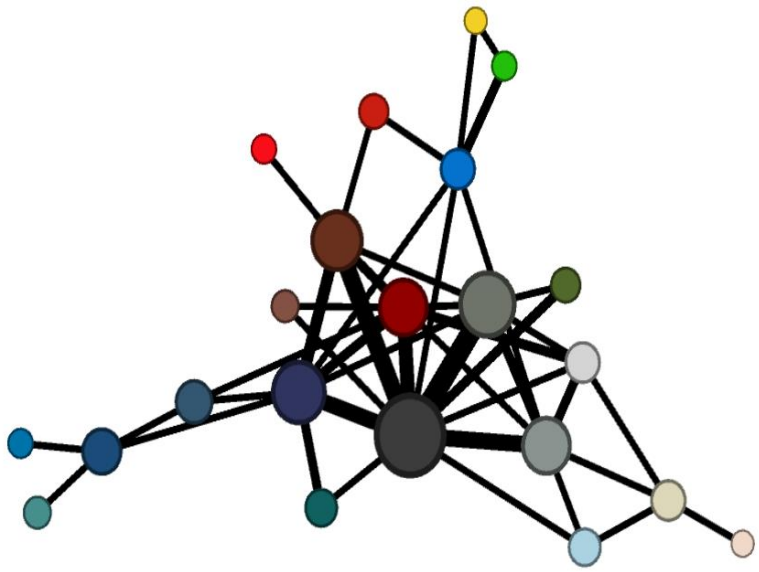
Figur 3.39: Fargene som er blitt brukt som aksentfarger i appen. Disse fargene er mer vibrante enn de vi finner i gradienten, som resulterer i et skift vekk fra et ellers monotont uttrykk. Blå, grønn og orange er også naboer i fargesirkelen, så de fungerer godt sammen.

Alle fargene er relativt dempede, inkludert de vi har benyttet som varsel-farger. Dette resulterer i et uniformt uttrykk der vibrante farger ikke tar for mye oppmerksomhet vekk fra det helhetlige inntrykket. Dersom et varsel skulle dukke opp er appen utformet på en slik måte at varslingen selv vil fange brukerens oppmerksomhet. Dette løser også eventuelle problemer som måtte oppstå dersom en bruker har rød/grønn-fargeblindhet, som er den vanligste formen for fargeblindhet (Norges Blindeforbund, u.å.), ettersom de røde og grønne fargene alene ikke er ansvarlige for å formidle suksess / feil.

Fargene vi har benyttet oss av minner om de man finner i naturen, spesielt de man finner i jorden, skogen og havet. Forsvaret skriver på sine nettsider at fargene de bruker symboliserer jorden, luften og vannet, som er de tre områdene de har sin operative tilstedeværelse i (Forsvaret, 2015, Farger). Vi vil argumentere for at vi klarer å beholde denne symbolikken med våre grønne -, brune - og blå fargetoner.

Bartram, Patra og Stone skriver om hvordan farger påvirker mennesker i sin forskningsrapport “Affective Color in Visualization” (Bartram, Patra, Stone , 2017). I løpet av tre eksperimenter (bildeanalyse, brukerskapte fargepaletter og bruker-rangerte fargepaletter) kommer Bartram et al. opp med åtte paletter de mener skaper konkrete følelser hos tilskuerne: calm, exciting, positive, negative, serious, playful, disturbing og trustworthy.

Fordi Forsvaret er en offentlig institusjon ville vi at bruken av farger skulle gi et profesjonelt inntrykk av appen. Fargene våre ser ut til å ha mye til felles med fargepaletten som uttrykker seriøsitet (Figur 3.40), noe som indikerer at de gir appen et seriøst og profesjonelt uttrykk.



Figur 3.40: Fargepaletten “Serious”, tatt fra “Affective Color in Visualization”. Størrelsen på nodene representerer hvor ofte den gitte fargen ble kategorisert som seriøs, og tykkelsen på kantene representerer hvor ofte to farger ble plassert i samme fargepalett under de tidligere eksperimentene i forskningsrapporten. (Bartram et al., 2017, s. 1370)

Tilgjengelighet

WCAG

WCAG (Web Content Accessibility Guidelines) består av spesifikke, individuelle krav, også kjent som suksesskriterier. Disse må møtes for å overholde WCAG-kravene som er pålagt ved norsk lov om universell utforming. Suksesskriteriene er delt inn i tre nivåer av samsvar: Nivå A, Nivå AA og Nivå AAA. Ingen av kravene fra Nivå AAA er pålagt i Norge, men de vil bidra til å gjøre grensesnitt enda mer tilgjengelig for allmennheten.

"Forskrift om universell utforming av informasjons- og kommunikasjonsteknologiske (IKT)-løsninger" (Lovdata, 2013, Forskrift om universell utforming [...]) sier per i dag at alle nettsider må møte 35 av 61 suksesskriterier fra WCAG 2.0 (Digitaliseringsdirektoratet (u.å.), WCAG 2.0-standarden). En oppdatert utgave av denne forskriften er på vei, der nye suksesskriterier fra WCAG 2.1 vil bli lagt til. Digitaliseringsdirektoratet sier på sine nettsider at Nivå AAA ikke er aktuelt å ta inn i norsk regelverk (Digitaliseringsdirektoratet (u.å.), WCAG 2.1-standarden), så trolig vil kravet i fremtiden være 47 av 78 suksesskriterier.

WCAG har ikke egendefinerte retningslinjer for apper, men det blir derimot spesifisert at tilgjengelighet på mobil dekkes av de eksisterende WCAG-retningslinjene (W3C, 2019, Mobile Accessibility at W3C). Dette betyr at apper er pålagt å følge de samme suksesskriteriene som nettsider. Ved å basere vår app på de gjeldende kriteriene fra WCAG 2.0, i tillegg til de nye Nivå A og Nivå AA-kriteriene fra WCAG 2.1, forsikrer vi oss om at vi overholder WCAG-kravene og norsk regelverk.

Suksesskriterie-analyse

Ved å følge WCAGs suksesskriterier øker man tilgjengeligheten av produktet. Å ville ha økt tilgjengelighet og god brukervennlighet kan begrunnes i økonomiske motiver, ettersom man kan øke markedsandeler og redusere supportkostnader (Sandnes, 2011, s. 19), men det kan også begrunnes i medmenneskelige motiver. Universelt

utformede grensesnitt resulterer i at flere kan bruke tjenestene, også de med funksjonshemninger. Vi mener at universell utforming er viktig for å inkludere nettopp denne brukergruppen, og derfor ønsker vi å følge suksesskriteriene.

Tabellen under er et utsnitt av en lengre tabell som finnes i vedlegg 4, og viser oversikt over ulike suksesskriterier, og hvor godt vi har oppnådd dem. Denne gjennomgangen har vi valgt å kalle en suksesskriterie-analyse. Kolonnen til høyre er fargekodet basert på grad av implementasjon / oppnåelse: Grønt = fullt implementert / oppnådd, orange = delvis implementert / oppnådd, krever videreutvikling, rødt = ikke implementert / oppnådd, krever videreutvikling.

Suksesskriterie	Nivå	Implementering
1.3.2 - Meningsfylt rekkefølge (Presenter innhold i en meningsfull rekkefølge)	Nivå A	Innholdet er presentert enten ved hjelp av alfabetisk sortering, eller basert på hvilken informasjon som er viktigst.
1.3.4 - Visningsretning (Brukeren må få velge om innholdet skal vises i liggende eller stående retning)	Nivå AA	Vi har foreløpig lagt inn en sperre som forhindrer at appen fungerer i landskapsmodus.
1.4.3 - Kontrast (Kontrastforholdet mellom teksten og bakgrunnen er minst 4,5:1)	Nivå AA	All tekst har minst et kontrastforhold på 4,5:1 til bakgrunnen den står på. Dette sjekket vi ved hjelp av programmet Colour Contrast Analyser (CCA)
1.4.4 - Endring av tekststørrelse (Tekst kan bli endret til 200% størrelse uten tap av innhold eller funksjon)	Nivå AA	Vi har benyttet oss av media queries, og endrer blant annet layout og skriftstørrelse basert på bredde i pixler, men vi har ikke gjennomført en test som baserer seg på

		endringer i telefoninnstillingene vil gjøre med appens utseende. Appen inneholder heller ikke innstillinger til å øke skriftstørrelse eller å zoome.
2.4.1 - Hoppe over blokker (Gi brukeren mulighet til å hoppe direkte til hovedinnholdet)	Nivå A	Brukeren kan hoppe til den siden de ønsker ved hjelp av navigasjonsmenyen, og hovedinnholdet vil alltid være plassert øverst. Sidene er ikke lange som på en nettside, så å hoppe over innhold vil ikke være relevant.

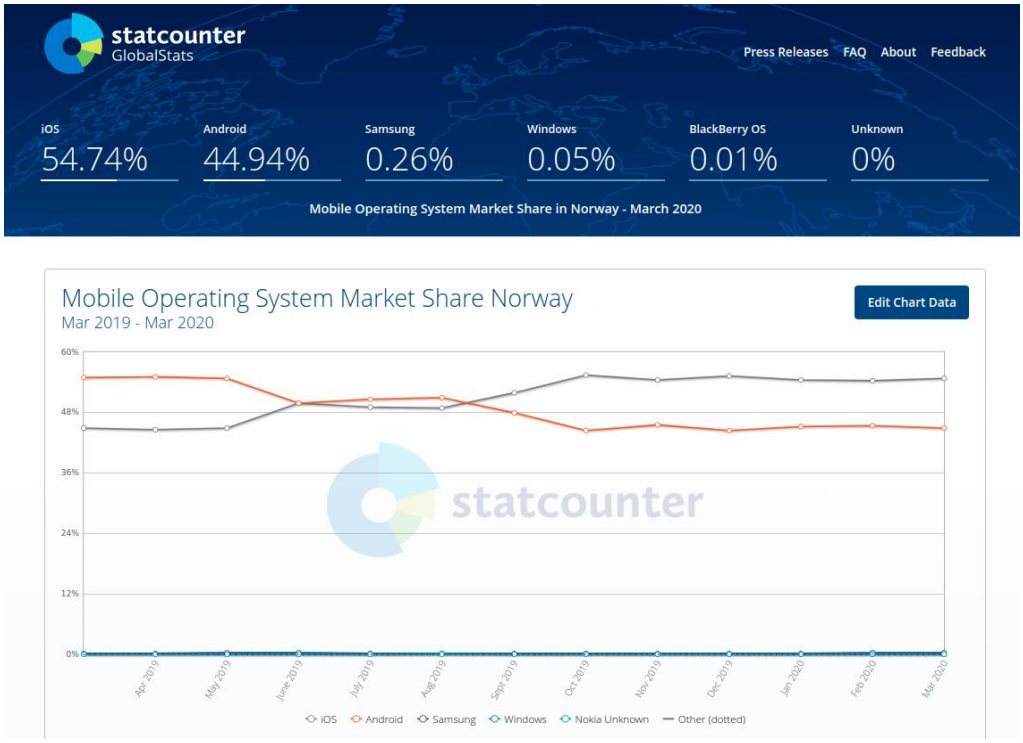
Den fulle suksesskriterie-analysen viser oss at appen vår følger 33 av 47 suksesskriterier. De resterende 14 kriteriene krever videreutvikling av varierende grad for å oppfylles. Dette kommer vi nærmere inn på under “Videre utvikling”, og i vedlegg 5. Som nevnt under WCAG-kapittelet, har ikke apper egne suksesskriterier men er pålagt å følge de samme som nettsider. Det er flere kriterier vi mener ikke helt passer med vår app, men vi har likevel jobbet for å oppfylle dem.

Utviklingsteknologier

Valg av språk / rammeverk

Vi bestemte oss tidlig for at vi ikke nødvendigvis ønsket å holde oss til et programmeringsspråk vi kjente fra før, men at vi heller ønsket å ta utgangspunkt i mobilmarkedet, og deretter se hva som ville være nødvendig å lære seg for å imøtekomme de aller fleste.

Vi startet med å se på trender i markedet for mobil-operativsystemer i Norge. Dette ville ha mye å si for hvem som vil ha mulighet til å bruke appen, og ettersom Forsvaret er en stor nasjonal aktør vil det være ekstra viktig at flest mulig har mulighet til å benytte seg av den.



Figur 3.41: Graf som viser fordelingen mellom Android og iOS i det norske markedet. (Statcounter, u.å)

Grafen i figur 3.41 er hentet fra StatCounter og viser en prosentvis sammenlikning av de mest populære mobil-operativsystemene i Norge per Mars 2020. På topp ligger Android og iOS med tilnærmet 45 og 55 % av markedet. Den neste på listen er Samsung med under 1%. Samsung bruker Android som en base for sin plattform og har noen Samsung-spesifikke endringer. I praksis vil det si at apper laget for Android også kan kjøres på Samsung-operativsystemet, og vi grupperer dem derfor med resten av Android-brukerne. Windows, som er den neste på listen med 0.05% av markedsandelen, annonserte nylig at de vil gå bort fra sin egenproduserte plattform og vil benytte Android i fremtidige produkter.(Dolcourt, 2019) Ut fra dette ser markedet todelt ut, med en relativt jevn fordeling mellom Android og iOS. Vi valgte dermed å fokusere på disse to plattformene.

Videre i prosessen trengte vi et språk som kunne dekke disse to plattformene mest effektivt. Enten kunne vi utvikle én app for Android ved bruk av Java eller Kotlin og én app for iOS ved bruk av Swift, eller vi kunne lære oss et nytt språk som ville dekke begge plattformene.

Alle medlemmene av utviklingsteamet hadde erfaring med Java fra tidligere skoleprosjekter, og det var fristende å skulle utvikle i et språk vi allerede kjente til. Men ettersom ca. 50% av markedet i Norge er iOS-brukere, ville ikke dette være ideelt, når målet til produkteier er å få informasjon ut til så mange som mulig. Å utvikle en app i Swift for iOS ville også ha utelatt nesten halvparten av befolkningen. Disse vurderingene gjorde at vi konkluderte med vårt tredje alternativ: å lære et nytt språk som ville dekke begge plattformer. Dette alternativet ville best dekke produkteiers ønsker, og i tillegg gi oss en mulighet til å lære noe nytt.

Ved å gjøre noen undersøkelser angående kryssplattform-språk fikk vi raskt kortet ned listen til noen få kandidater: React Native, Flutter, Xamarin, and Ionic.

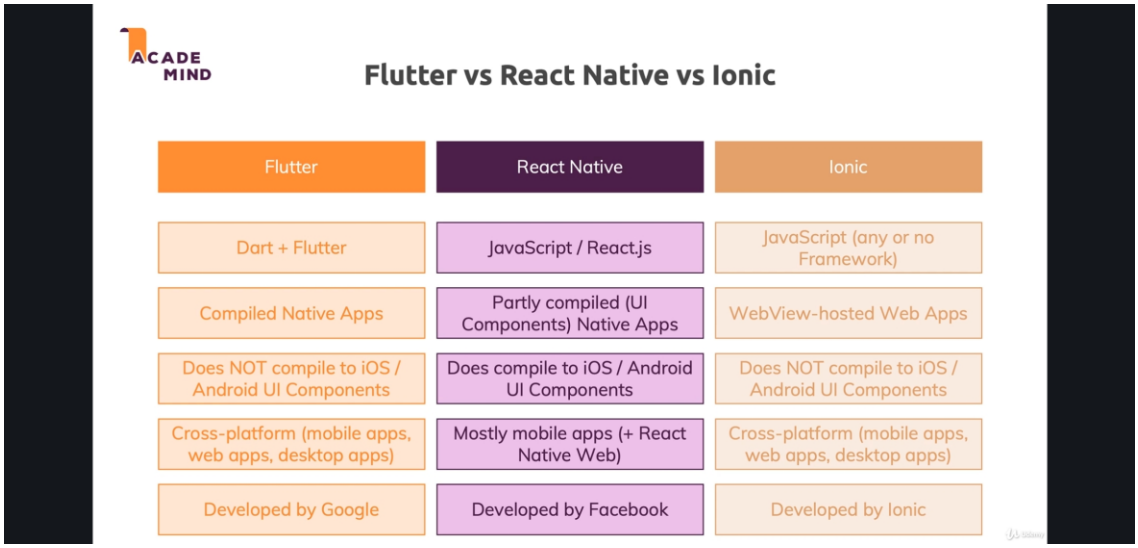
React Native er et rammeverk utviklet og vedlikeholdt av Facebook og har vært et veldig populært valg for utvikling av kryssplattform applikasjoner. Rammeverket gir mange muligheter for gjenbruk av kode og bruker JavaScript som basespråk. Selv om det har vært populært i lengre tid har markedsandelen deres gått noe ned i løpet av 2019 da det mangler regelmessig oppdateringer og har en tendens til å henge litt bak de *native* språkene til Android og iOS da det ikke utvikles i samarbeid med noen av dem.(Manchanda (2019))

Flutter er et rammeverk basert på programmeringsspråket Dart som i struktur og oppbygging er ganske likt Java, med noen alterasjoner. Både Flutter og Dart utvikles og vedlikeholdes av Google og Flutter er relativt nytt på markedet (2017). (Beklemysheva, (u.å), avsnitt 2) Flutter kompileres ned til native kode for både iOS og Android og er strukturert i widgets som er relativt enkle byggestener å bruke. Ulempene med Flutter er stort sett relatert til at rammeverket fortsatt er relativt nytt, og at ikke alle funksjoner som en del andre rammeverk gjør nytte av er tilgjengelige enda. Appene utviklet med Flutter har en tendens til å være litt større enn React Native apper, på bakgrunn av hvordan widgets i Flutter er bygget opp. (Manchanda (2019)

Xamarin er vedlikeholdt av Microsoft siden 2016 og er et *open-source* rammeverk som bruker programmeringsspråket C# og .NET. Det er godt testet og gir tilgang til

en hel del platformspesifikke elementer. Man må derimot benytte seg av flere av Microsofts plattformer når man utvikler i Xamarin og appene er en del større enn de fleste andre apper ettersom de bruker mange biblioteker gjennom .NET nettverket.(Sakovich, 2019)

Ionic er et open-source brukergrensesnitt verktøy som brukes til å utvikle mobilapplikasjoner, pc-applikasjoner og progressive web-applikasjoner. Drifty som eier Ionic ble startet i 2012. (Ionic, (u.å), avsnitt 1) I motsetning til de andre teknologiene allerede nevnt som er hybrid-Native, er Ionic en hybrid-web løsning. Hybrid-Native vil si at man programmerer i et språk og at rammeverket deretter organiserer hvilken komponent som vises i brukergrensesnittet basert på hvilket native språk som brukes av plattformen programmet kjøres på. Hybrid-Web bruker de eksakt samme komponentene for alle plattformer ved å kjøre programmet i en container som har støtte for komponentene som brukes. Begge løsninger har dermed samme kodebase for appene, men det er forskjell i hvordan løsningen bygges basert på platform.(Kremer, (u.å), Hybrid-Native vs. Hybrid-Web)

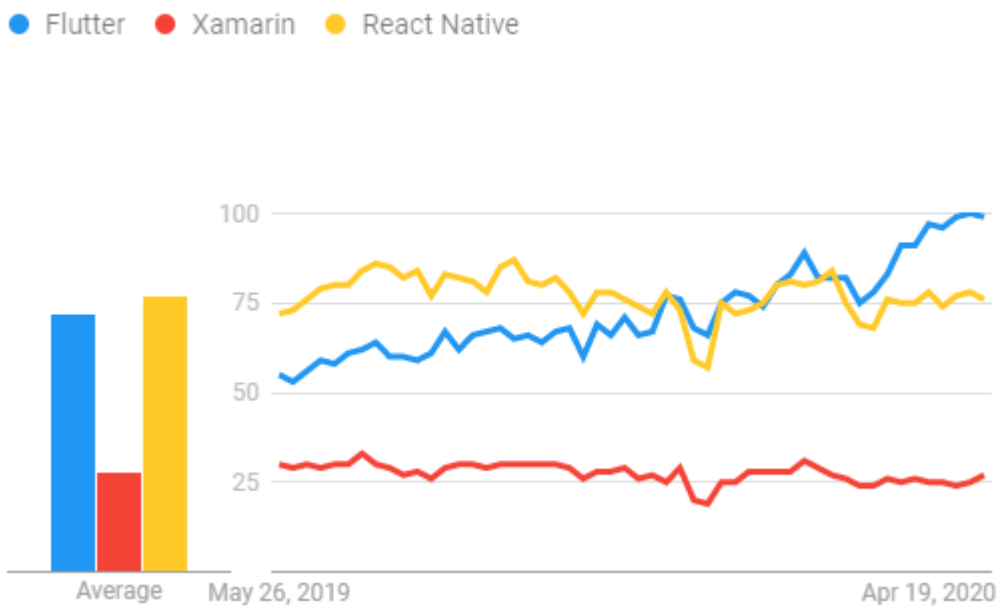


Figur 3.42: Sammenlikning av Flutter, React Native og Ionic. (Schwarzmüller, 2020, 0:00)

Da vi skulle velge språk og rammeverk var det spesielt noen områder det var viktig for oss å ta hensyn til. Vi ønsket at språket vi valgte skulle være utviklet og vedlikeholdt av en solid aktør slik at det fantes nok ressurser og brukerstøtte. Vi

ønsket også å ta hensyn til ytelse. I tillegg så vi på hvilke språk vi kunne fra før, hva vi vurderte som mest nyttig fremover, og hva vi selv vurderte som mest interessant.

Ettersom vårt mål for dette prosjektet var å utvikle en mobilapplikasjon og ikke en webapplikasjon, var det mest naturlig for oss å velge en teknologi hvor man først og fremst fokuserer på mobilplattformene. Ionic, hvor man utvikler en web-app som kan kjøres i *webcontainere* på mobil var dermed ikke naturlig å velge. I tillegg var det viktig for oss at vi endte opp med en native app for hver brukerplattform ettersom dette gir den beste ytelsen og et appformat brukerne er kjent med. Dette er noe Ionic heller ikke tilbyr. (PcMag, (u.å), 2 avsnitt)



Figur 3.43: Google trend graf for søkeordene Flutter (Blå), Xamarin (Rød) og React Native (Gul).(Google trend, (u.å))

Når det kom til Xamarin så vi at populariteten lå betydelig lavere enn både Flutter og React-Native. Ettersom man må benytte mange av Microsofts produkter for å utvikle og appene blir relativt store i størrelse valgte vi ikke å gå for dette.

Et av de største argumentene mot React Native i dag er at det har en tendens til å ligge litt bak de native utviklingsspråkene når det kommer til funksjonalitet. Dette problemet er ikke noe som eksisterer i samme størrelsesorden hos Flutter ettersom

det utvikles tett med Android, fordi Google står bak begge teknologiene. Flutter appellerte til oss ettersom at språket det siste året har overgått React Native i popularitet (Figur 3.43) (Google trend, (u.å)). og at det er et relativt nytt språk. Språkmessig bruker Flutter Dart som basespråk mens React Native bruker JavaScript. Dart er på mange måter veldig likt Java som vi alle hadde erfaring med, mens det var flere i teamet som ikke hadde JavaScript-erfaring fra før.

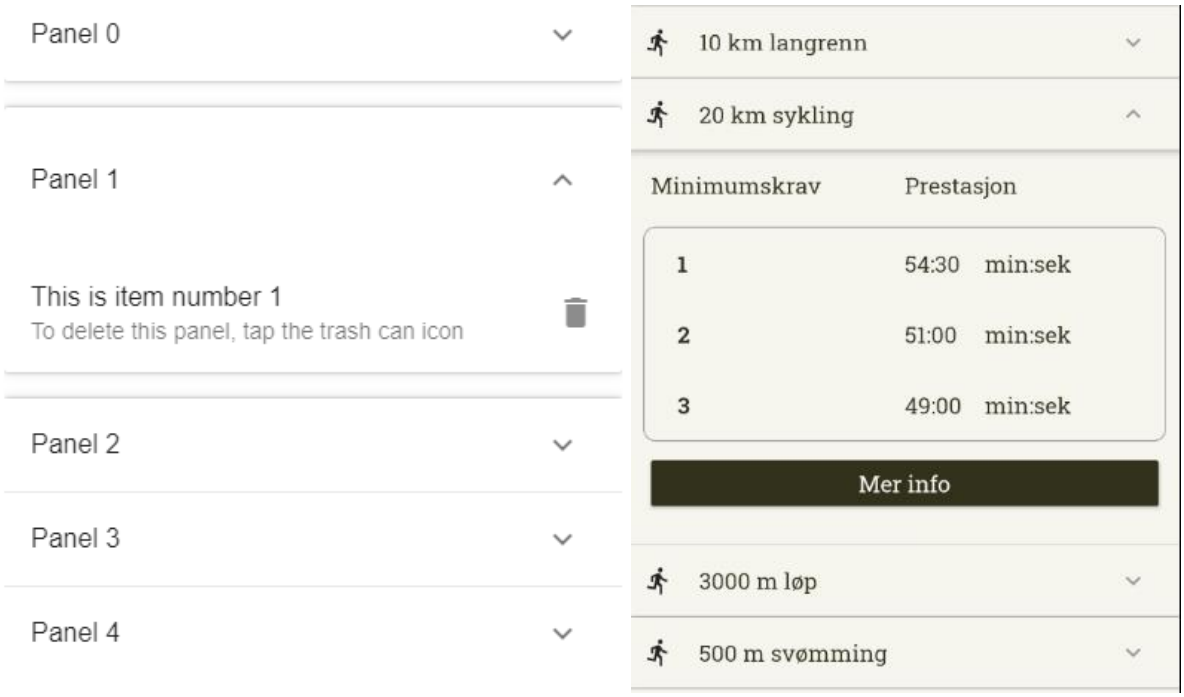
Flutter kompileres ned til fullstendig native apper som kjøres på målplattformen. React Native kompileres på nesten samme måte men er annerledes ved at ikke all koden kompileres ned, men noe JavaScript-kode kjøres inne i native-appen. En annen essensiell forskjell mellom de to rammeverkene er at Flutter ikke kompilerer ned til iOS og Android *grensesnitt*-komponenter slik som React Native gjør. I praksis vil dette bety at man i Flutter kan designe utseende på komponenter akkurat slik man ønsker, mens React Native bruker de forhåndsdefinerte grensesnitt-elementene for iOS og Android. Man vil dermed få mye større frihet når det kommer til design og layout ved å bruke Flutter.

Frontend: Flutter og Dart

Strukturmessig består Flutter av widgets. Widgets fungerer i utviklingssammenheng som en liten komponent av en app. Widgets kan settes sammen med andre widgets, til man til slutt ender opp med et fullverdig produkt. På grunn av at Flutters modulære struktur, er det gunstig å dele opp så mye man kan av programmet i mindre widgets som senere kan gjenbrukes på andre steder ved behov. For eksempel er infoboksen vi har brukt gjennomgående i appen en egen widget som kan gjenbrukes. Spesifikke kodeeksempler ligger i vedlegg 6.1.

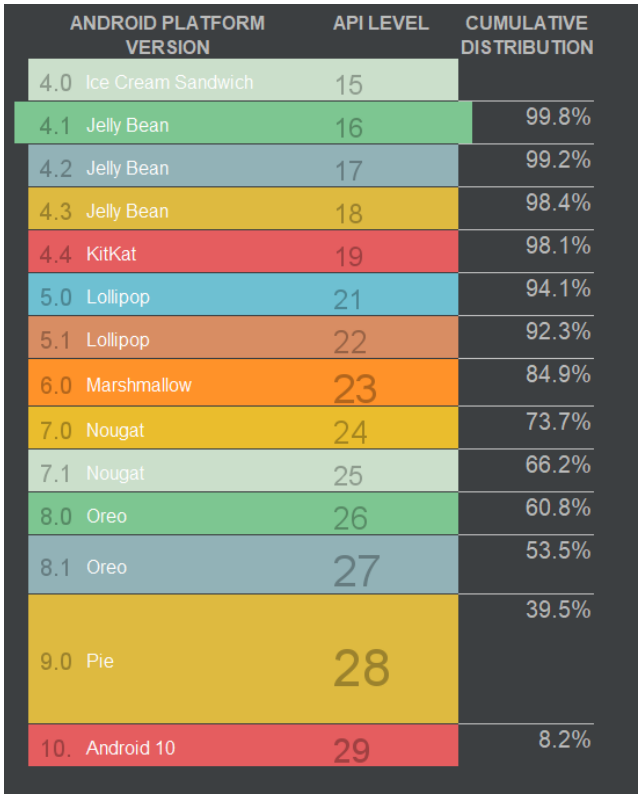
Flutter kommer med sitt eget sett med widgets og med muligheten til å lage sine egne widgets, eller bygge videre på dem som allerede eksisterer. Vi har benyttet oss av de eksisterende widgets der dette har passet, men også bygget videre på eller endret widgets for å skreddersy dem til vår bruk. Et konkret eksempel på dette er vår Expansion Panel List, hvor vi endret den originale Expansion Panel-widgeten fra å

ha mellomrom mellom liste-elementene til å ikke ha det, ettersom vi synes dette gjorde seg bedre (Figur 3.44).



Figur 3.44: Skjermbilder som viser standard Expansion Panel List sammenliknet med vår Custom Expansion Panel List. Standardversjonen har en mellomrom mellom panelene når de er åpne.

Flutter kjører på alle versjoner av iOS siden iOS 8, og Android siden Android Jelly Bean. Siden 99.8 % av Android brukere i dag har telefoner som støtter Jelly Bean, dekker dette stort sett hele Android-brukergruppen. (Flutter, (u.å), What devices and OS versions does Flutter run on?)



Figur 3.45: Oversikt over prosentandelen Androidenheter som støtter de forskjellige versjonene av Android operativsystem.(Android studio, 2020)

Når det kommer til iOS er det kun 7% som bruker en eldre versjon enn iOS 12, (Apple (u.å), iOS and iPadOS Usage) og allerede i mai 2016 var prosentandelen som brukte en eldre versjon enn iOS 8 nede i 5% (Statista, 2019).

Vi konkluderte med at vi ved å bruke Flutter som utviklingsverktøy vil dekke stort sett alle brukere som skulle ønske å benytte seg av appen. I tillegg vil vi ha store muligheter til å gjøre frie designvalg uavhengig av plattform, og dermed ende opp med en app for både Android og iOS med god ytelse.

Backend: Dart

Flutter er skrevet i Dart som er et open-source programmeringsspråk utviklet av Google for apputvikling på flere plattformer. Det er et objektorientert, klassebasert språk som kan kompileres ned til JavaScript eller maskinkode (Dart, u.å.).

Database: Firebase (BaaS)

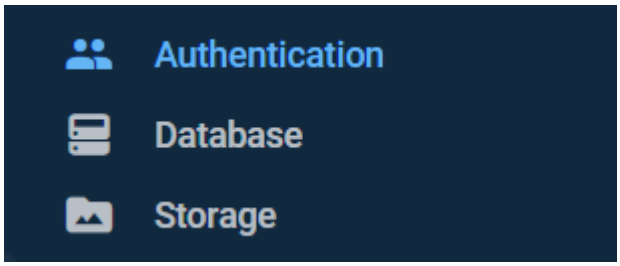
Firebase er en sanntidsdatabase som tilbyr backend as a service (BaaS). BaaS fungerer slik at mye av det som skjer “behind the scenes”, for eksempel brukerautentisering, blir gjort av systemet slik at utviklerne ikke trenger å tenke på dette under utvikling. Flutter bruker plugins for å aksessere Firebase sitt *API*. Gjennom prosessen har vi støtt på både fordeler og ulemper med å bruke Firebase.

Flutter og Firebase er begge utviklet av Google og basert på dette antok vi at disse teknologiene ville fungere bra med hverandre. Dette er nettopp en av grunnene til at vi valgte å bruke Firebase. Samtidig var det å benytte Firebase enda en mulighet for oss som gruppe å lære noe nytt. Tabellen under viser oversikten over fordelene og ulempene Firebase hadde under utviklingen:

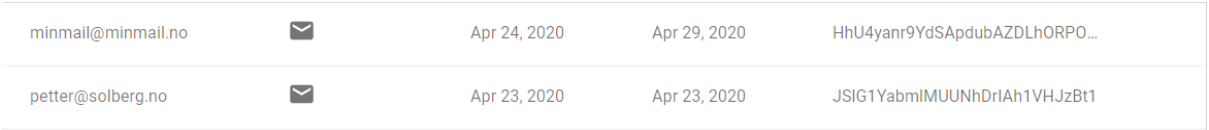
Fordeler	Ulemper
Automatisk validering av e-postformatering og mulighet for e-postverifisering ved utsendelse av bekreftelses e-post	Noen exceptions fra Firebase vises ikke som exceptions i Flutter, som fører til uspesifikk feilhåndtering (github user: kataklysmo, 2018)
Eget system for å håndtere brukerkontoer	Noen issues som er rapportert til utviklerne har enda ikke blitt fikset, selv etter veldig lang tid.
Eget fillagringssystem	
Man kan spesifisere egne regler for lesing og skriving til databasen for hver tabell i databasen og for fillagringssystemet	
Gir mulighet for oppdatering av innloggingsinformasjon og resetting av passord	

Firebase hindrer misbruk av databasen ved å sette en grense på opprettede brukere fra en IP-adresse innenfor et tidsrom	
Hashing av passord skjer automatisk i databasen med scrypt passordbasert "key derivation function"	

Vi har benyttet oss av tre deler av Firebase-databasen: Authentication, Database og Storage (Figur 3.46).



Figur 3.46: Skjermbilde som viser et utsnitt av mappestrukturen i Firebase. Her er Authentication uthevet.

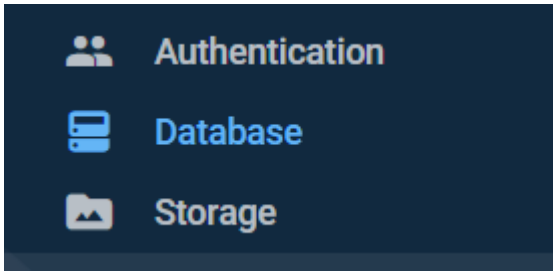


Figur 3.47: Skjermbilde som viser et utsnitt av test-brukerkontoer vi har opprettet under prosjektperioden.

Authentication-fanen er der brukerkontoene blir opprettet med e-post og passord (Figur 3.47). De blir også tildelt en brukerid. Kodeeksempler på bruk av Firebase API ligger i vedlegg 6.2. Firebase tilbyr flere andre innloggingsmetoder, blant annet via Google, Facebook og Twitter. Vi valgte å ikke benytte oss av tredjepartsinnlogging av følgende årsaker:

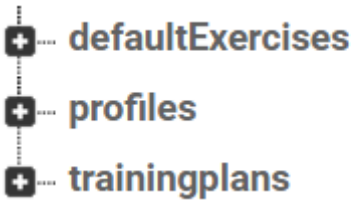
- Det er ikke alle som har kontoer på sosiale medier og vi ønsker ikke å utelukke noen

- Det er ikke sikkert brukerne stoler på applikasjonen i forhold til tilgang til deres sosiale medier-konto
- Tredjepartskontoen kan inneholde falsk informasjon
- Å gi for mange login-valg kan være forvirrende, og brukerne kan glemme hvilken metode de har benyttet tidligere



Figur 3.48: Skjermbilde som viser et utsnitt av mappestrukturen i Firebase. Her er Database uthevet.

Database-fanen er delt inn i tre deler vi har spesifisert: “defaultExercises”, “profiles” og “trainingplans” (Figur 3.49).



Figur 3.49: Skjermbilde som viser det øverste nivået av vår egendefinerte struktur i databasen.

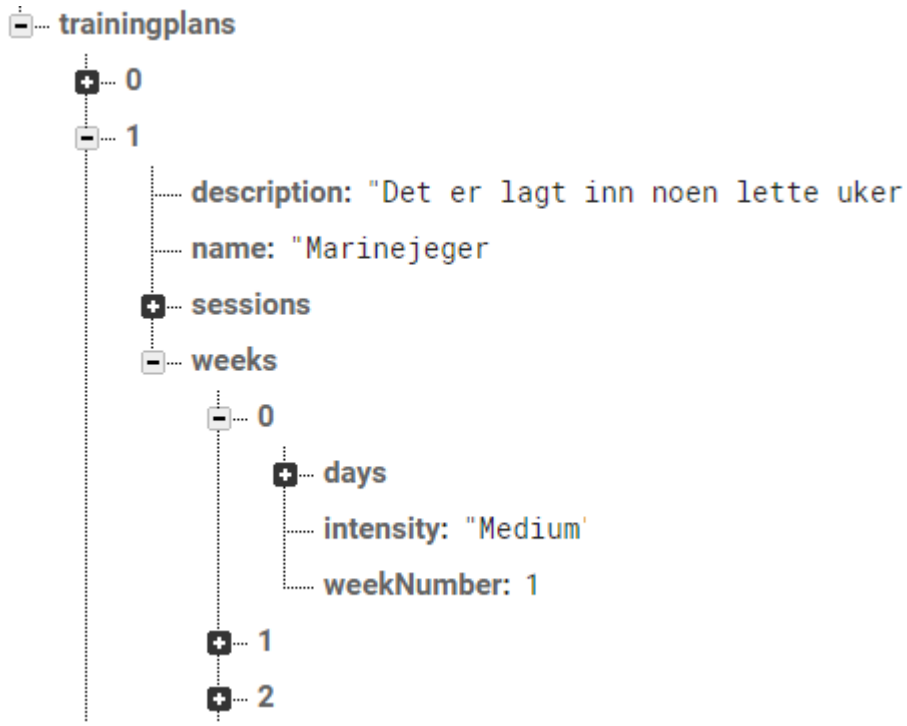
“DefaultExercises” inneholder de øvelsene som brukes i Forsvarets generelle tester i blant annet førstegangstjeneste og sesjon. Alle brukere som oppretter en profil har tilgang til disse øvelsene fra begynnelsen av, slik at de kan sette opp mål for seg selv og jobbe mot å nå minstekravene de ønsker. En gitt øvelse inneholder et navn, type øvelse, og inputfelter der brukeren kan fylle inn informasjon (Figur 3.50).

Figur 3.50: Skjermbilde som viser en mer detaljert oppbygging av “defaultExercises”.

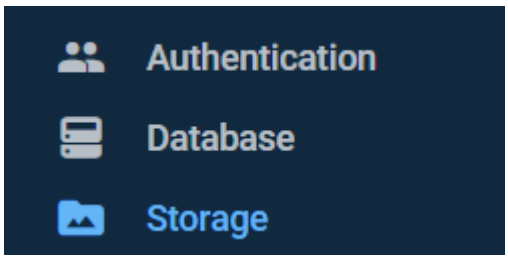
“Profiles” inneholder flere brukerid’er, og er stedet der all profilinformasjon utenom autentiserings-info lagres. Hver brukerid inneholder fornavn, etternavn, fødselsdato, kjønn og ønsket stilling. Data fra treningsdagboken blir også lagret under profil, og er koblet til de spesifikke brukerid’ene. Her lagres resultater fra treningsøktene man logger,i tillegg til de egendefinerte øvelsene man oppretter på sin profil slik at de kan gjenbrukes (Figur 3.51).

Figur 3.51: Skjermbilde som viser en mer detaljert oppbygging av “profiles”.

“Trainingplans” er en samling av treningsplaner fra forskjellige avdelinger i forsvaret. Hver treningsplan inneholder en liste med ukedager, som igjen inneholder treningsøkter med forskjellige øvelser (Figur 3.52).



Figur 3.52: Skjermbilde som viser en mer detaljert oppbygging av “trainingplans”



Figur 3.53: Skjermbilde som viser et utsnitt av mappestrukturen i Firebase. Her er Storage uthevet.

Storage-fanen er der filer blir lagret. Dette er i vårt tilfelle profilbildene til brukerprofilene. Disse bildene blir også lagret til en spesifikk brukerid.

SQLite

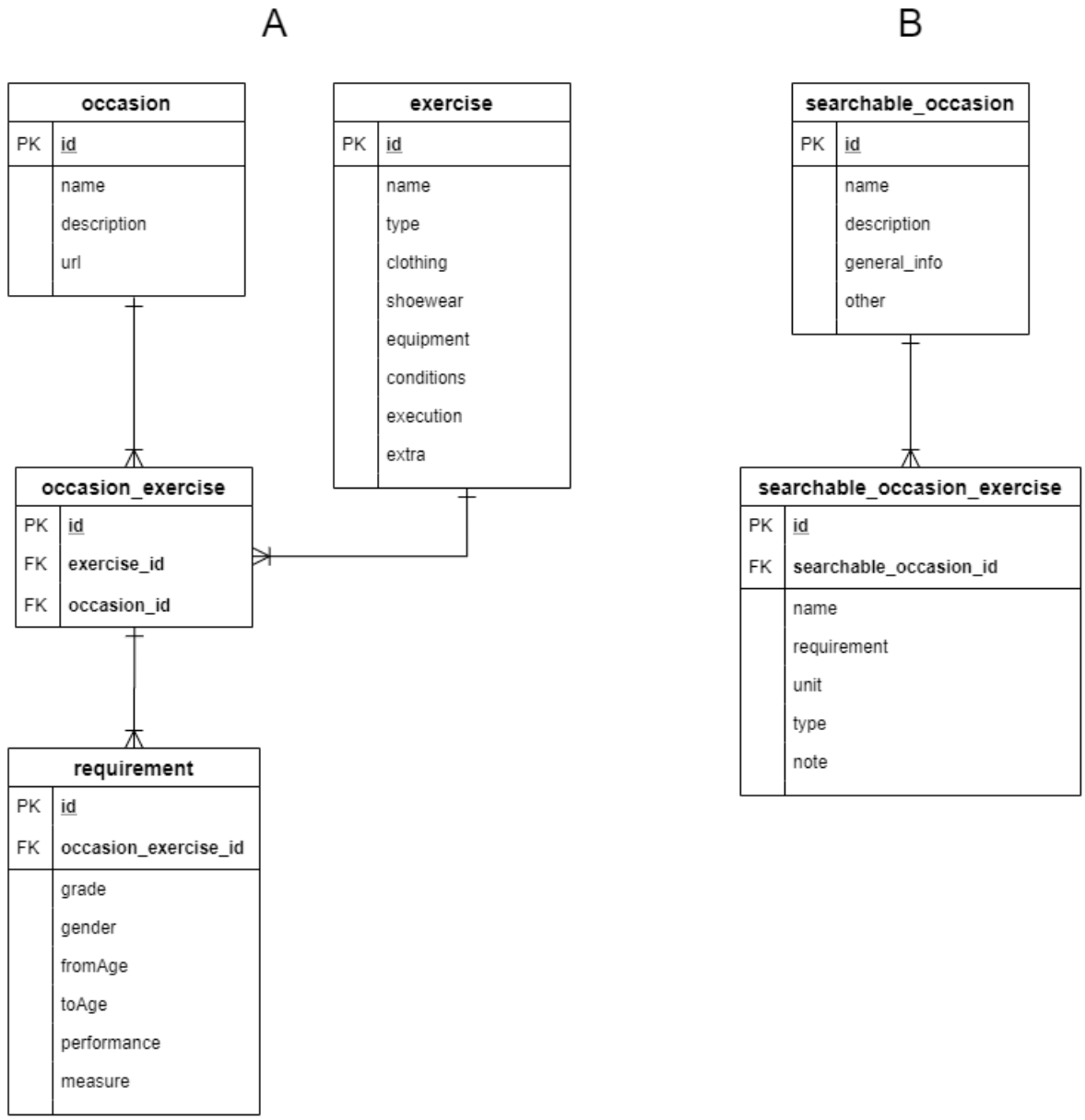
All forsvarsdata som er relatert til opptakskrav, øvelser og krav er implementert i en SQLite database som følger med appen. Dataen er hentet fra siste versjon av “Reglement for fysisk testing” som ble revidert og publisert 1. januar 2020.

Databasen er satt opp med egne scripts (Vedlegg 6.3) siden Forsvaret ikke kunne dele en databasedump av dataen på bakgrunn av sikkerhetsmessige årsaker. Begrunnelsen for å ha denne dataen i en lokal SQLite database på telefonen er at denne dataen skal være tilgjengelig i offline-modus. I tillegg så revideres reglementet kun hvert fjerde år, så behovet for kontinuerlige oppdateringer er ikke nødvendig.

ER-diagram

Forsvarsdataen er delt opp i to separate relasjoner, vist i Figur 3.54 som relasjon A og relasjon B. Tabellen under inneholder en mer detaljert forklaring av relasjonene.

Relasjon	Forklaring	Oppsett av tabeller
A	For sesjon, førstegangstjeneste, utdanning og årlige tester som deler flere like øvelser,og har en skala fra en til ni for prestasjonen knyttet til hver øvelse.	Fire tabeller: Anledning (occasion), øvelse (exercise), anledningsøvelser (occasion_exercise) og krav (requirement). Siden det finnes inntil ni krav per øvelse for en anledning, så er dette løst med en koblingstabell mellom anledning og øvelse som holder på et sett med krav.
B	For de søkbare førstegangstjenestene, som blant annet fallskjermjeger, kystjeger og marinejeger. Disse har en egen struktur for opptakskrav og eget sett med øvelser som gjennomføres.	Det er forskjellige øvelser, og bare ett krav per øvelse for de søkbare førstegangstjenestene. Dette løses med to tabeller: søkbar førstegangstjeneste (searchable_occasion) og søkbar førstegangstjeneste-øvelser (searchable_ocassion_exercise).



Figur 3.54: Et ER-diagram som viser de to ulike relasjonene forsvarsdataen er delt inn i, i den lokale SQLite-databasen.

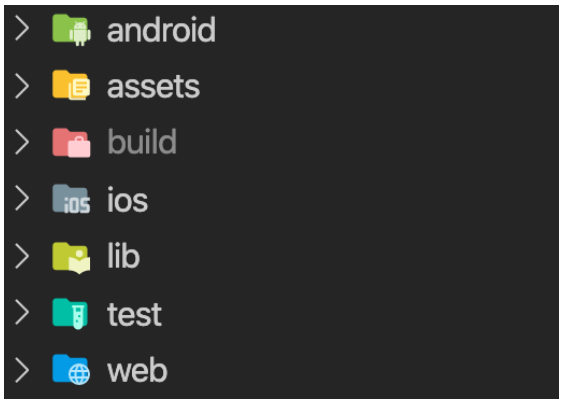
Kodestandarder

Når det kommer til kodestandarder gjorde vi en del valg tidlig i prosessen for at prosjektet skulle være konsistent. Det finnes ikke en riktig eller feil måte å gjøre det på når det kommer til kodestandarder, så lenge hele gruppen er enige om å følge et mønster.

- Filnavn følger snake_case
- Metodenavn og variabelnavn følger camelCase
- PascalCase blir brukt for klassenavn
- lowercase for mappe/package-navn

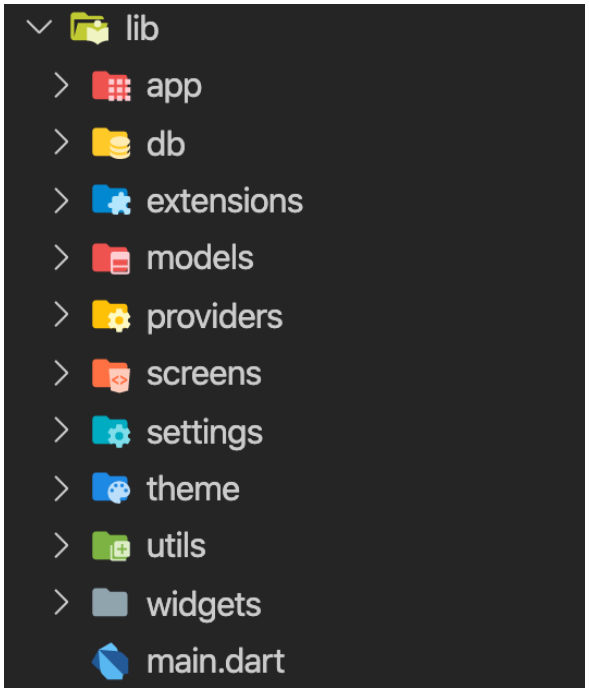
Vi har også benyttet oss av ulike formateringsverktøy som sørger for naturlig linjedeling og innrykk i koden, slik at den skal være oversiktlig å lese.

Det var viktig for gruppen å følge flere av “clean code”-prinsippene (Shekhawat, 2018), blant annet ved å jobbe for å holde widgetene nette og enkle både for lesbarhet og gjenbrukbarhet, skille *business-logikk* fra visnings-logikk og ha selvforklarende kode.



Figur 3.55: Skjermbilde som viser et utsnitt av kodens mappestruktur.

Figur 3.55 viser hvordan de viktigste elementene i vår mappestruktur ser ut. Den viktigste mappen er lib, der hele prosjektet blir laget. Koden som ligger her blir kompilert til iOS- og Android-kode som havner i hver sin mappe. Assets-mappen inneholder bilder, videoer, fonter og den lokale SQLite-databasen. Test-mappen inneholder enhetstester til prosjektet, og web-mappen eksisterer for å gjøre det mulig å bygge prosjektet til en webapplikasjon, dersom man ønsker det.



Figur 3.56: Skjermbilde som viser videre innhold i lib-mappen.

Lib-mappen er videre delt opp i undermapper ut ifra hva de spesifikke filene gjør (Figur 3.56). Main.dart ligger på toppnivå og er filen som kjører hele prosjektet. Mappen app inneholder prosjektets *Material Apper* som setter opp material design og navigator (routing) for appen. Det er satt opp to Material Apper, en hoved-app som wrapper selve produktet og en error-app som brukes hvis oppstart av programmet feiler. Db inneholder en *singleton* databasehjelper-klasse som brukes for versjonshåndtering og kopiering av SQLite databasen over til appen, og sikrer en trygg tilkobling i kjøretid for spørringer. Models inneholder modellklassene som fungerer som blueprints for å opprette objekter. Providers er *state management*-verktøyet som blir brukt i dette prosjektet. Disse filene ivaretar datatilstand og datatilgang på tvers av widget-hierarkiet og brukes for å separere business-logikk vekk fra widgetene. Screens inneholder koden for å vise hvert enkelt skjermbilde, mens theme inneholder alt av designvalg relatert til fonter, farger, størrelser osv. Utils inneholder vår fargepalett og innstillinger relatert til skjermstørrelser. Mappen widgets inneholder alle mindre komponenter som blir brukt i screens for å lage de komplette skjermbildene. Extensions brukes for å legge til funksjonalitet til allerede eksisterende dart-biblioteker (Figur 3.57).

```
extension Age on DateTime {
  int getAge(DateTime dob) {
    var birthday = DateTime(this.year, dob.month, dob.day);
    var age = this.year - dob.year;
    if (this.isBefore(birthday)) {
      age--;
    }
    return age;
  }
}
```

Figur 3.57, eksempel på extension som gjør det smidig å finne alder på et datetime-objekt kombinert med et fødselsdag datetime-objekt.

Sikkerhet

Sikkerhet er viktig i ethvert digitalt system. For akkurat vår app er det enda viktigere, siden den er utviklet i samarbeid med Forsvaret. Vi fikk klare retningslinjer fra Forsvaret at det var viktig å følge GDPR og ha fokus på sikkerhet gjennom utviklingen av appen.

GDPR

General Data Protection Regulation (GDPR) er en EU-lov som omhandler personvern og datasikkerhet. Målet er å gi personer kontroll over deres personlige data i forhold til hvor og hvordan den blir behandlet. Data må lagres trygt, og datasystemer må derfor utvikles med hensyn til personvern.

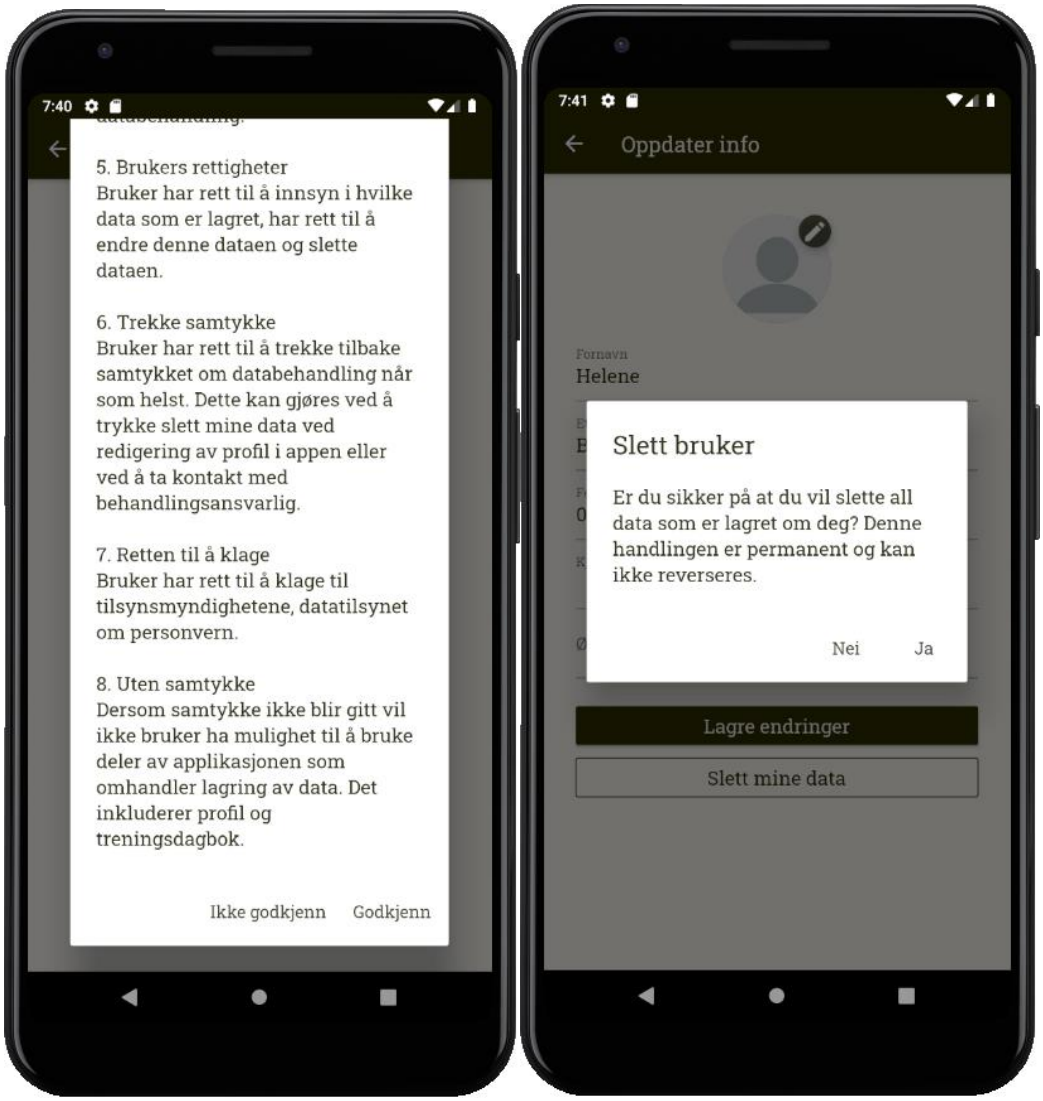
Data kan kun bli behandlet dersom en av følgende forhold gjelder: Brukeren har gitt samtykke til databehandling, det er nødvendig for en kontrakt, kontrolløren har en juridisk plikt til å behandle brukerdataene, helsemessige årsaker som kan være avgjørende for liv og død, eller behandlingen er viktig for allmenn interesse (European union, 2016).

Behandling av personopplysninger innebærer i vårt tilfelle innsamling, organisering, lagring og bruk av personlig data. Personlig data er definert som en samling av data,

som sammen kan lede til identifisering av et levende individ (Directorate-General for Communication, u.å). Når man baserer rettighetene til å behandle data på brukerens samtykke, kan brukeren trekke tilbake samtykket når som helst. I samsvar med GDPR kreves det i personvernlovn at vedkommende må være over 13 år for å kunne samtykke til bruk av personopplysninger, jf. personopplysningsloven kapittel 3 §5.

Når data blir behandlet må brukeren informeres om følgende (Justis- og beredskapsdepartementet, 2018):

1. Identiteten og kontaktinformasjon til behandlingsansvarlig
2. Formålet med at dataene behandles og det lovlige grunnlaget for behandlingen.
3. Mottaker og potensielle mottakere av dataene.
4. Tidsbegrensingen på hvor lenge dataene blir lagret, eller kriteriene som bestemmer denne tidsbegrensingen.
5. Rettighetene til å se, endre eller slette data og rettighetene til å nekte databehandling.
6. Rettighetene til å trekke tilbake samtykket
7. Rettighetene til å klage til tilsynsmyndighetene
8. Hva som skjer dersom samtykke ikke blir gitt



Figur 3.58: Skjermbilder av appen som viser et utsnitt av brukervilkårene, og popup-vinduet som dukker opp om man ønsker å slette dataene som er lagret om seg.

Disse retningslinjene påvirker vårt prosjekt når det gjelder profilinformasjonen som lagres. Dataene blir lagret med bakgrunn i samtykke fra bruker, og det er derfor nødvendig å implementere mulighet for å slette dataene i appen. Dette kan gjøres via “rediger profil”-siden hvor man har en egen knapp for å slette all brukerdata. Samtykket blir gitt når man oppretter en bruker, ved å godkjenner vilkårene og betingelsene som dukker opp på skjermen. Disse brukervilkårene er basert på listen i over. Det er også satt aldersrestriksjoner for opprettelse av brukerprofil, i samsvar med personvernlovens 13 års grense. Dette blir både validert i inputfelt og informert om i vilkårene.

Sikker lagring av data

All personlig data i applikasjonen er lagret i Firebase. Firebase benytter seg av sikkerhetsregler for å sette restriksjoner for lesing og skriving av data. Disse reglene kan settes individuelt for hver tabell basert på om en bruker er logget inn, eller basert på om man skal kunne lese eller skrive til tabellen uavhengig av bruker. Når det gjelder “profiles” vil man kun ha lese og skriverettigheter dersom den gitte brukerid’en er blitt autentisert, for å forhindre at uvedkommende skal få tilgang til personlig data.

```
{
  "rules": {
    "profiles": {
      "$uid": {
        ".read": "auth.uid == $uid",
        ".write": "auth.uid == $uid",
      },
    },
    "trainingplans": {
    ".read": "true",
    ".write": "false",
    },
    "defaultExercises": {
      ".read": "true",
      ".write": "false"
    }
  }
}
```

Figur 3.59: Et skjermbilde som viser de gjeldende sikkerhetsreglene i Firebase-databasen, og hvorvidt de tre tabellene “profiles”, “trainingplans” og “defaultExercises” er lesbare og skrivbare eller ikke.

Firebase Storage er en egen del av Firebase. Mens man i databasen kun kan lagre tekst-strenger, kan man i Storage lagre filer. Storage har egne regler, men de er satt på samme måte som profilinformasjon i figur 3.59. Kun en autentisert bruker har tilgang til data lagret under sin brukerid, da denne blir brukt til å lagre profilbilder. Her kan en bruker skrive, lese, oppdatere og slette til databasen, så lenge brukeren er autentisert (Figur 3.60).

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{userId} {
      allow read, write, update, delete: if request.auth.uid == userId;
    }
  }
}
```

Figur 3.60: Skjermbilde av reglene i Firebase Storage.

Lagring av passord

Passordene til brukerkontoene som lagres i Firebase blir hashet med en modifisert versjon av *scrypt* (Firebase Open Source, 2020). Scrypt er en passordbasert "key derivation function" som er designet slik at det er kostbart (i tid og minne) å dekryptere data (Tarsnap, u.å.). En bruker som logger inn benytter gjerne ett eller veldig få forsøk på å autentisere seg, og dermed blir tiden og minnet brukt på å kjøre funksjonen såpass liten at den blir neglisjerbar. Derimot vil et brute-force attack resultere i at denne funksjonen kjøres milliarder av ganger, som fører til at tidsbruken og minnebruken blir signifikant og forhåpentligvis uoverkommelig. Hvert Firebase-prosjekt får unike passordhash-parametere som kun er tilgjengelig når man er logget inn som eier eller redaktør. Dersom man eksporterer all brukerdata fra Firebase krever det i tillegg at man har en eier- eller redaktør-rolle for å få tilgang til passordhash og *passordsalt* i denne filen.

SQL-injeksjon

SQL-injeksjon er en måte å misbruke en applikasjon på ved å bruke SQL-spørringer i inputfelt, som kan resultere i endring eller sletting av data fra databasen. Firebase er en *noSQL* -database, og er dermed ikke utsatt for dette. Databasen bruker et API som leser og skriver data, og dermed er ikke profildataene utsatt for SQL-injeksjon. SQLite-databasen ville vært utsatt dersom brukerne hadde kunnet lagret noe til denne, noe de ikke har mulighet til. Denne databasen er kun brukt til å hente ut data og benyttes ikke til lagring av data opprettet av brukere.

Apptillatelser

Det er viktig for Forsvaret at lokasjonsopplysninger om brukeres telefoner ikke blir lagret eller tilgjengeliggjort for eksterne parter. Apper som lekker posisjonsinformasjon utgjør en betydelig risiko for deres operasjoner og ansattes sikkerhet og er i dag et høyst aktuelt tema. (NRK, 2020) For at Forsvarets personell trygt skal kunne bruke appen på militære områder, er det svært viktig med et kritisk forhold til hvilke tillatelser appen ber brukeren om. Appen vi har utviklet ber om to tillatelser, og ingen av dem er nødvendige å akseptere for å bevare appens hovedfunksjon.

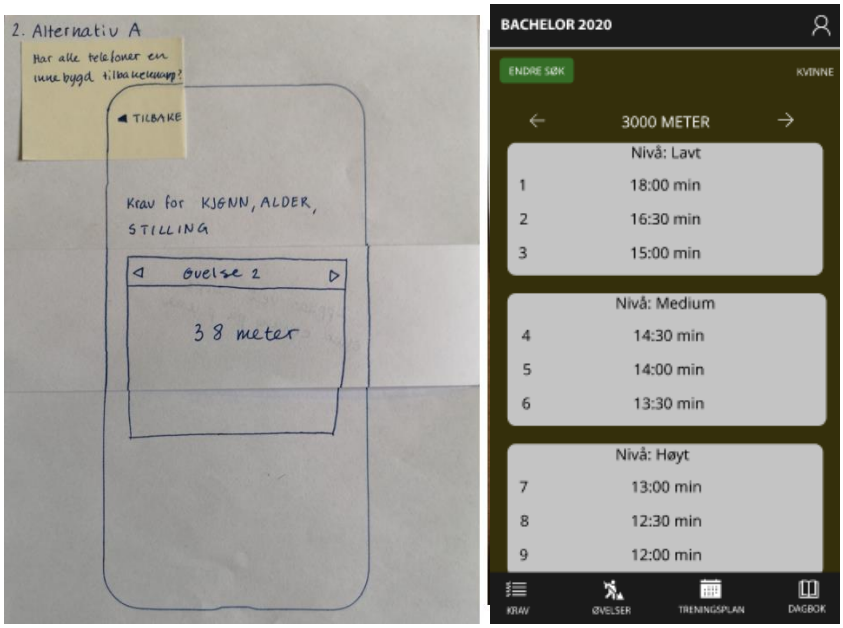
Den første tillatelsen innebærer bruk av internett. Denne tillatelsen kreves for at data skal kunne hentes og skrives til ekstern database, og for å ha muligheten til å åpne linker til Forsvarets nettsider. Er denne tillatelsen ikke gitt har man ikke mulighet til å benytte seg av treningsplaner, treningsdagbok eller profilside. Man kan derimot fortsatt benytte seg av all søkefunksjonalitet i forbindelse med krav og øvelser, ettersom forsvarsdata lagres i en database på telefonen.

Den andre tillatelsen innebærer å skrive og lese fra telefonens filer. Denne tillatelsen er nødvendig for at brukeren skal kunne legge profilbilde til sin bruker og er ikke påkrevd.

Brukertesting

Vi hadde ambisjoner om mange brukertester gjennom prosjektet, ettersom vi ser verdien i å la potensielle brukere ta aktiv del i utviklingsprosessen for å hindre ekstra arbeid og løsninger som ikke er brukervennlige. Dessverre var ikke dette noe vi fikk gjennomført i like stor grad som vi hadde ønsket på grunn av den pågående Covid-19 situasjonen i 2020. Vi har likevel fått utført to brukertester, en før Norge ble stengt ned og en på familie og venner i senere tid med et mer ferdig produkt.
I dette avsnittet er resultatene av brukertestene kort oppsummert. For mer om hvordan testene ble utført se Brukertester i vedlegg 8.

Brukertest 1



Figur 3.61: Eksempel fra papirprototypen til venstre og fra Figma prototypen til høyre.

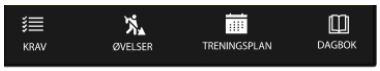
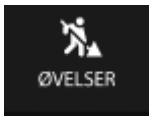
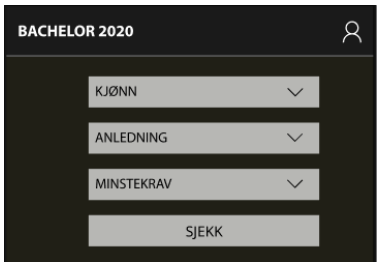
Den første brukertesten var tiltenkt utført med en papirprototype (Vedlegg 3.6.1), men vi fant tidlig ut at det ikke var hensiktsmessig å gjennomføre en brukertest med en slik lavnivå løsning. Papirprototypen bestod av mange ark og mindre papirbiter, og fordi det var mye å holde styr på måtte alle vært til stede under enhver brukertest, noe som ville vært lite effektivt. Vi valgte derfor å gå over til å bruke en digital prototype i Figma isteden. Mer av Figma-prototypen ligger i vedlegg 3.6.2.

Før vi gikk over til Figma prototypen fikk vi en del tilbakemeldinger på papirprototypen om at utformingen i appen kunne forbedres. Vi tok dette med oss da vi gikk over til Figma, noe som resulterte i at utseende på de to prototypene er ganske forskjellige. Gjennomføringen og metode for brukertest 1 kan sees i vedlegg 8.1, her presenteres funn og læring:

Funn:

Vi samlet brukernes tilbakemeldinger i en tabell:

Tilbakemelding	Område	Eksempel

Rart å bare få se en liten del av minstekravene til en øvelse.	Visning av minstekrav	
Leter etter et hjem ikon eller knapp som kan ta en tilbake til starten	Utforming av navbar	
Rart at man må bla til siden for å se flere øvelser og minstekrav, ikke umiddelbart intuitivt	Visning av minstekrav	
Ikonet for øvelser i navigasjonsbaren virket litt rart	Utforming av navbar	
Rart at kjønn kom først i filtermenyen, ville hatt anledning først	Søk etter minstekrav	

Vi tok med oss tilbakemeldingene fra testen og gjorde følgende tiltak i videreutviklingen av appen:

1. Man får se alle minstekravene til en anledning når man gjør et søk, man trenger ikke fylle ut minstekravet man leter etter når man gjennomfører et søk.
2. Krav skjermen fikk et hus-ikon for å illustrere at den fungerer som hjemskjerm for appen.
3. Øvelsene ligger under hverandre i en liste slik at det er enklere å navigere til den øvelsen man vil se og mer intuitivt å finne frem.

4. Vi gikk gjennom alle ikoner og fikk våre egne ikoner for mange av øvelsene slik at det skulle være enkelt for brukeren å finne fram.
5. Kjønn kommer nå etter anledning i filteret slik at det første man blir bedt om å fylle ut er Anledning og ikke kjønn.

Brukertest 2

Brukertest 2 ble utført med familie og venner ettersom det ikke var muligheter for å samles med andre fremmede som kunne teste appen. Alle fikk i oppgave å finne spesifikk informasjon om fysiske krav og øvelser først på Forsvarets nettsider på laptop deretter på nettsider på mobil, før de så skulle gjøre det samme i appen. Vi målte tiden. I denne seksjonen presenteres funn og læring, mer informasjon ligger i vedlegg 8.2.

Funn:

Vi samlet brukernes resultater i en tabell:

Oppgave 1		Oppgave/Søkekriterier: - Mann - Førstegangstjeneste - Finn krav for karakter 4 i medisinballstøt		
Person	Kjennskap til nettsidene?	Nettside	App	Nettside på mobil
Mann 20	Ingen	2.05 min	12 sek	3:20 min
Kvinne 24	Ingen	46 sek	15 sek	2:04 min
Mann 24	Ingen	57 sek	11 sek	2:35 min
Mann 25	Noe	4:50 min	9 sek	3:12 min
Kvinne 55	Noe	1:28 min	9 sek	2:46 min

Mann 56	Ingen	37 sek	13 sek	1:53 min
Gjennomsnitt:		1:47 min	11 sek	2:38 min
Oppgave 2		Oppgave/Søkekriterier: - Kvinne - Sesjon - Finn øvelser som testes på sesjon - Hva er kravet for å få 6 på en styrkeøvelse?		
Mann 20	Ingen	2:34 min	11 sek	1:00 min
Kvinne 24	Ingen	2:46 min	12 sek	1:22 min
Mann 24	Ingen	1:42 min	16 sek	1:35 min
Mann 25	Noe	55 sek	12 sek	59 sek
Kvinne 55	Noe	3:41 min	14 sek	1:46 min
Mann 56	Ingen	1.55 min	15 sek	2:23 min
Gjennomsnitt:		2:15 min	13 sek	1:30 min
Oppgave 3		Oppgave/Søkekriterier: - Marinejeger - Finn de fysiske kravene for marinejeger. - Hvor mange brutalbenk må man ta?		
Mann 20	Ingen	1:15 min	15 sek	1:31 min
Kvinne 24	Ingen	1:17 min	18 sek	1:22 min
Mann 24	Ingen	31 sek	12 sek	1:36 min
Mann 25	Noe	45 sek	11 sek	58 sek
Kvinne 55	Noe	2:00 min	20 sek	2:12 min

Mann 56	Ingen	25 sek	23 sek	40 sek
Gjennomsnitt:		1:02 min	16 sek	1:23 min
Oppgave 4		Oppgave/Søkekriterier: - Mann - Årlig test - 50-59 år - Hvor mange pullups skal til for å få karakter 3?		
Mann 20	Ingen	1:02 min	11 sek	3:25 min
Kvinne 24	Ingen	48 sek	18 sek	2:41 min
Mann 24	Ingen	1:12 min	14 sek	2:05 min
Mann 25	Noe	50 sek	10 sek	1:46 min
Kvinne 55	Noe	1:35 min	19 sek	2:35 min
Mann 56	Ingen	1.40 min	25 sek	3:05 min
Gjennomsnitt:		1:11 min	16 sek	2:36 min

Læring:

Av denne brukertesten kan vi se at man i gjennomsnitt sparer opp mot ett minutt for hvert søk man gjør i appen sammenliknet med nettsiden på desktop, og to minutter sammenliknet med nettsiden på mobil. Ettersom man stort sett ikke har tilgang til laptop i test/treningssammenheng ser vi på differansen mellom søk i appen og søk i mobilnettleser som sterkest indikator på at appen kan bli nyttig for å tilgjengeliggjøre informasjon og lette arbeidet med å finne frem til minstekrav.

Enhetstesting

Enhetstesting er en testmetode som innebærer å teste individuelle komponenter i en applikasjon. Det går under kategorien white-box testing som betyr at det er selve koden som blir testet, og ikke funksjonalitet eller design. Automatiserte enhetstester gjør det mulig å gjøre regresjonstesting mens man utvikler for å sørge for at endringer til en gitt komponent ikke påvirker utfallet. Enhetstesting gjør det lettere å finne bugs tidlig i systemet og fastsette hvilken komponent buggen oppstår i.

Vi har laget flere enhetstester for appens modellklasser, de finnes i vedlegg 9. Disse testene blir kjørt når en *pull request* blir opprettet, og når endringer blir pushet til master, ved continuous integration.

Systemtesting

Systemtesting er black-box testing av hele systemet for å se om det oppfyller kravene som er satt sammen av de funksjonelle kravene, de ikke-funksjonelle kravene og brukerhistoriene (Vedlegg 7). Black-box testing er testing av systemet fra et brukerperspektiv, og er dermed ikke relatert til koden, men til hva brukeren faktisk opplever gjennom å bruke systemet. Man kan teste både systemets design og systemets funksjonelle og ikke-funksjonelle krav. Systemet blir testet opp mot forventet oppførsel og forventninger fra potensielle brukere. Denne testen inneholder dermed en liste med krav som originalt stammer fra brukerhistorier vi skrev tidlig i prosjektet (Vedlegg 7), og produkteiers krav for å sjekke om disse kravene er oppfylt. Resultatene fra systemtesten ligger i vedlegg 10.

Videre utvikling

Det første vi må se på er hva som skal til for at Forsvaret skal kunne ta over appen. En viktig del av dette innebærer at appen har kapasitet til å håndtere mange brukere på en gang. Slik som det er nå, er det kun Firebase-databasen som setter grenser for antall aktive brukere til enhver tid. Under utviklingen har vi brukt en gratisversjon, men her må det opprettes et abonnement om man ønsker større brukergrupper på permanent basis.(Firebase, u.å.).

Videre må det tas hensyn til at Forsvaret stiller høye krav til sikkerhet i appen for at den skal kunne publiseres. Det er svært viktig at appen er trygg å bruke og at informasjon lagres på trygge forsvarlige måter. Det er også svært viktig å følge GDPR. Disse tingene har vi tatt hensyn til under utvikling, men det kreves en gjennomgang av appen fra Forsvarets side for å verifisere at sikkerheten samsvarer med kravene de stiller.

Når godkjenninger er gitt, er neste steg å få appen publisert i App Store og Google Play Store. For å publisere en app i både Google Play Store og App Store må det følges en del retningslinjer (Apple, 2020) (Google, 2020). Appen blir først lastet opp for review der den blir vurdert, og når den godkjennes kan den publiseres.

Når det kommer til tilgjengelighet og universell utforming, kreves det som nevnt i kapittelet om tilgjengelighet videreutvikling relatert til 14 av 47 suksesskriterier. De viktigste endringene er relatert til hvordan folk bruker telefonen, og ikke direkte til hvordan de bruker appen i seg selv. Ulike brukere har ulike behov. Kanskje har de telefonen montert på et spesielt stativ som bare støtter én posisjon, eller kanskje de er svaksynte og har spesifisert i telefonens innstillinger at de ønsker større tekst eller opplesning av innhold ved hjelp av en skjermleser. For at appen skal være universelt utformet, er det viktig at vi tilrettelegger for disse brukerne, og implementerer at appen skal reflektere innstillingen en bruker har satt på telefonen sin.

Vi må tilrettelegge for navigasjon utelukkende ved bruk av tastatur, og sørge for at appen fungerer som planlagt uten bruk av en telefons touch-funksjonalitet. Dette må gjøres fordi noen personer velger å koble tastatur til telefonene sine, og vi ønsker å tilrettelegge for disse, ettersom de kan være potensielle brukere av appen. Videre planer for utvikling tilknyttet tilgjengelighet ligger i vedlegg 5.

Gjennom systemtesting fant vi også forbedringspotensiale relatert til innlastingshastighet. I følge Salz P. A. er ett sekund eller mindre det magiske tallet. Trege applikasjoner fører til tap av brukere(Salz P. A., 2019). For vår applikasjon tar Innlogging ca 2 sekunder. Dette gjelder tilkobling til Firebase databasen, men dette

kan optimaliseres ved å komprimere bildene som blir lastet opp for raskere innhenting av data. Vi har allerede forbedret hastigheten noe ved å bruke Firebase plugin API fremfor REST API, men det er fremdeles forbedringspotensiale. Vi kan gjøre databasestrukturen flatere for å ikke hente inn mer data enn man trenger, for eksempel ved å flytte øvelser opprettet av en bruker ut i en egen tabell, istedenfor at det er en del av profildata. Det tar også ca 2 sekunder å laste inn treningsplanene. Dette kan optimaliseres ved å dele opp innhenting av dataene til å laste inn når man går inn på hver enkelt plan, istedenfor at alle planene lastes når treningsplansiden åpnes.

Når det kommer til videreutvikling av funksjonalitet kom vi nesten helt i mål med all funksjonalitet som var tiltenkt appen. Det eneste som gjenstår på dette området er å implementere lagring av treningsresultater. Dette er noe vi ønsker å arbeide videre med etter endt prosjektperiode.

4. Konklusjon

Avslutningsvis vil vi se igjennom alle målene i oppgaven for å se om vi har oppnådd dem. Resultatmålene ble delt opp etter produktets mål og utviklingsteamets mål. Produktmålene bestod av å oppnå oppgitt app-funksjonalitet, å se til at appen oppfyller Forsvarets sikkerhetskrav og at appen er tilgjengelig for alle i målgruppen. Disse punktene er presentert og nøye vurdert i rapporten.

App-funksjonaliteten er diskutert i produktdokumentasjonen, og vi har nådd alle kravene med unntak av resultatloggføring. Sikkerhet har hatt høyt fokus og er vurdert i kapittelet om sikkerhet, og vi har tatt hensyn til alle Forsvarets krav under utviklingen.

Ut ifra systemtesten i vedlegg 10, ser man at produktet fungerer godt på både Android, og iOS. Som diskutert i delen om valg av språk og Flutter, gjør det at vi har muligheten til å nå bortimot 100% av den potensielle brukergruppen. Dermed konkluderer vi med at vi har oppnådd målet om å være tilgjengelig for så godt som alle i målgruppen, med unntak av en svært liten gruppe som ikke har mobiltelefon eller benytter et svært gammelt mobiloperativsystem.

Utviklingsteamets resultatmål var å lære en ny teknologi. Gjennom dette prosjektet har vi ikke bare lært oss én ny teknologi, men flere. Vi har lært oss Flutter og Dart, Firebase, SQLite, Rive og Continuous Integration.

Effektmålene ble også delt opp på samme måte, etter produktets og utviklingsteamets mål. Produktmålene bestod av å tilby enklere og raskere tilgang til informasjon om fysisk testing, og å kvalitetssikre fysisk form i Forsvaret. Sistnevnte er vanskelig å vurdere på nåværende tidspunkt, men må måles over tid hos Forsvaret når de tar appen i bruk. Produktet er imidlertid spesielt rettet mot å nå akkurat dette målet. Vi har også gjennomført brukertester, som tydelig viser at produktet vårt oppnår målet om at det skal være raskere og enklere å finne informasjon med appen, enn ved å søke på nettsidene.

Utviklingsteamets effektmål var å forbedre samarbeidsferdigheter og få erfaring med arbeidslivet ved å jobbe med en produkteier. Utviklingsprosessen har definitivt vært en test av samarbeidsferdigheter med tanke på Covid-19. På grunn av dette har vi vært nødt til å lære oss helt nye måter å samarbeide på, som peker mot at våre samarbeidsferdigheter har forbedret og utviklet seg.

Gjennom prosjektet har vi som gruppe opptrådt mer som en uavhengig enhet som har tilbudt et produkt, enn som ansatte hos produkteieren. Det har gitt oss en annen type erfaring med arbeidslivet enn vi forventet, men det har medført mye læring. Vi har ikke jobbet like tett med produkteier som vi hadde ønsket på grunn av Covid-19 situasjonen. Vi har hatt stor grad av frihet og har tatt de aller fleste valg og beslutninger på egenhånd, og senere fått det godkjent av produkteier. Dermed har vi fått mer erfaring med arbeidslivet, men på en annen måte enn forventet.

Det ferdige produktet gir et profesjonelt inntrykk gjennom farge og ikonbruk. Brukertest 2 viser at vi har oppnådd ønsket resultat med tanke på hvor mye raskere og enklere det er å finne informasjon gjennom appen enn på nettsidene til Forsvaret. Dette tyder på at utformingen av appen er brukervennlig og at utviklingen har resultert i et godt produkt. Vi har også kontroll på hva som må utvikles i fremtiden for å gjøre produktet *enda* bedre.

Det er vår endelige vurdering at vi har hatt klare mål og at disse målene er nådd på en god måte. Dermed avslutter vi prosjektet med en applikasjon vi er stolte av og som vi håper Forsvaret benytter seg av i fremtiden.

5. Kilder

- Android studio (2020), Help me choose device hentet fra The Android Studio Program.
- Apple (u.å) App Store, hentet fra: https://developer.apple.com/support/app-store/
- Apple (2020, 4. mars), App Store Review Guidlines, hentet fra: https://developer.apple.com/app-store/review/guidelines/
- Bartram, Patra, Stone (2017). *Affective Color in Visualization*. Denver, CO, USA
- Beklemysheva, (u.å) Flutter: Pros and Cons for Seamless Cross Platform Development [Blogginnlegg] hentet fra https://steelkiwi.com/blog/flutter-pros-and-cons-for-seamless-cross-platform-development/
- Dart (u.å.) A tour of the Dart language, hentet fra: https://dart.dev/guides/language/language-tour#important-concepts
- Digitaliseringsdirektoratet (u.å.), WCAG 2.0-standarden, hentet fra: https://uu.difi.no/krav-og-regelverk/wcag-20-standarden, 03.05.20
- Digitaliseringsdirektoratet (u.å.), WCAG 2.1-standarden, hentet fra: https://uu.difi.no/krav-og-regelverk/webdirektivet-og-wcag-21/wcag-21-standarden, 06.05.20
- Directorate-General for Communication (u.å.), What is personal data, hentet fra: https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data_en
- Doulcourt, (2019, 5 oktober), Microsoft isn't making another Windows phone for one simple reason, hentet fra https://www.cnet.com/news/microsoft-isnt-making-another-windows-phone-for-one-simple-reason/
- European union (2016, 27. april), General Data Protection Regulation, hentet fra: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN
- Firebase (u.å.), Pricing plans, hentet fra: https://firebase.google.com/pricing/
- Firebase Open Source (2020,18. mai), Firebase Authentication Password Hashing, hentet fra: https://firebaseopensource.com/projects/firebase/scrypt/

- Flutter, (u.å) FAQ, hentet fra https://flutter.dev/docs/resources/faq#what-devices-and-os-versions-does-flutter-run-on
- Flutter, (u.å) Documentation hentet fra: https://api.flutter.dev/flutter/services/SystemChrome/setPreferredOrientations.html
- Forsvaret (u.å) Egenerklæringen, hentet fra https://forsvaret.no/egenerklæring
- Forsvaret (2015, 10. desember), Farger, hentet fra https://forsvaret.no/brandbook/hvordan/farger#tab1 04.05.20
- Forsvaret (2011, 31. mai), Forsvarets visuelle profil, hentet fra: https://docplayer.me/2544039-Forsvarets-visuelle-profil.html
- Forsvaret (2019, 22 mai) Fysisk testing, hentet fra https://forsvaret.no/tjeneste/ansatt/regelverk/reglement-om-fysisk-testing
- Forsvaret (u.å). Førstegangstjeneste hentet fra https://forsvaret.no/forstegangstjeneste
- Forsvaret. (2019, 22. november). Hva er Forsvarets høgskole? Hentet fra https://forsvaret.no/hogskolene/forsvarets-hogskole
- Forsvaret, (2020, 30 april) Personell, hentet fra https://forsvaret.no/aarsrapport/statistikk/personell
- Forsvaret (u.å) Sesjon, hentet fra https://forsvaret.no/karriere/forstegangstjeneste/sesjon
- Forsvaret (u.å) Søkbare førstegangstjenester, hentet fra https://forsvaret.no/søkbare-førstegangstjenester
- Github user: kataklysmo (2018, 22. november), Firebase Storage catch error when file that does not exist, hentet fra: https://github.com/flutter/flutter/issues/24652
- Google (2020), Prepare your app for review, hentet fra: https://support.google.com/googleplay/android-developer/answer/9815348
- Google trend, (u.å) hentet fra: https://trends.google.com/trends/explore?q=%2Fg%2F11f03_rzbg,Xamarin,React%20Native
- Hultgreen, G. (2018, 19 april), Rekordmange kriger om å få studere i Forsvaret, hentet fra : https://www.dagbladet.no/nyheter/rekordmange-kriger-om-a-fa-studere-i-forsvaret/69717250

- Ionic, (u.å), About us, hentet fra: https://ionicframework.com/about
- Justis og beredskapsdepartementet (2020, 20. juli), Lov om behandling av personopplysninger (personopplysningsloven), hentet fra: https://lovdata.no/dokument/NL/lov/2018-06-15-38/KAPITTEL_gdpr-3#KAPITTEL_gdpr-3
- Kremer, (u.å) Comparing Cross-Platform Frameworks, hentet fra: https://ionicframework.com/resources/articles/ionic-vs-react-native-a-comparison-guide
- Lovdata (2013, 25. juni), Forskrift om universell utforming av informasjons- og kommunikasjonsteknologiske (IKT)-løsninger, hentet fra https://lovdata.no/dokument/SF/forskrift/2013-06-21-732
- Manchanda, A (2019, 6 desember) Where do cross-platorm app frameworks stand in 2020? hentet fra https://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019/
- Norges Blindeforbund (u.å.), Fargeblind / fargesvak, hentet fra https://www.blindeforbundet.no/oyehelse-og-synshemninger/fargeblindhet
- NRK, (2020, 18 Mai) Norske offiserer og soldater avslørt av mobilen, hentet fra: https://www.nrk.no/norge/xl/norske-offiserer-og-soldater-avslort-av-mobilen-1.14890424
- PcMag (u.å) Native application hentet fra: https://www.pcmag.com/encyclopedia/term/native-application
- Sakovich, N. (2019, 16 desember), Pros and Cons of Using Xamarin for iOS and Android Development [blogginnlegg] hentet fra https://www.sam-solutions.com/blog/the-pros-and-cons-of-using-xamarin-for-mobile-app-development/
- Salz P. A. (2019, 22. August), How fast is fast enough?, hentet fra: https://www.forbes.com/sites/peggyannesalz/2019/08/22/how-fast-is-fast-enough-mobile-load-times-drive-customer-experience-and-impact-sales/#35b54105ef31
- Sandnes, F. (2011). *Universell utforming av IKT-systemer*. Oslo: Universitetsforlaget

- Schwarzmüller, M. (2020), Flutter Alternatives [Videoklipp], hentet fra: https://www.udemy.com/course/learn-flutter-dart-to-build-ios-android-apps/learn/lecture/10459804#overview
- Shekhawat, R (2018, 24. august), Clean code principles: Be a better programmer, hentet fra: https://simpleprogrammer.com/clean-code-principles-better-programmer/
- SSB, (2020, 11 mars) Fødte, hentet fra https://www.ssb.no/fodte/
- StatCounter, (u.å), Mobile Operating System Market Share Norway, hentet fra https://gs.statcounter.com/os-market-share/mobile/norway
- Statista, (2019, oktober), Share of Apple devices by iOS version worldwide from 2016 to 2019, hentet fra: https://www.statista.com/statistics/565270/apple-devices-ios-version-share-worldwide/
- Tarsnap (u.å.), Stronger key derivation via sequential memory-hard functions, hentet fra: https://www.tarsnap.com/scrypt/scrypt.pdf
- Utdanning.no (2020, 30 april) Ansatt i Forsvaret, hentet fra https://utdanning.no/yrker/beskrivelse/ansatt_i_forsvaret
- W3C (2019, 1. mars), Mobile Accessibility at W3C, hentet fra: https://www.w3.org/WAI/standards-guidelines/mobile/ 07.05.20

6. Vedlegg

1. Ordliste

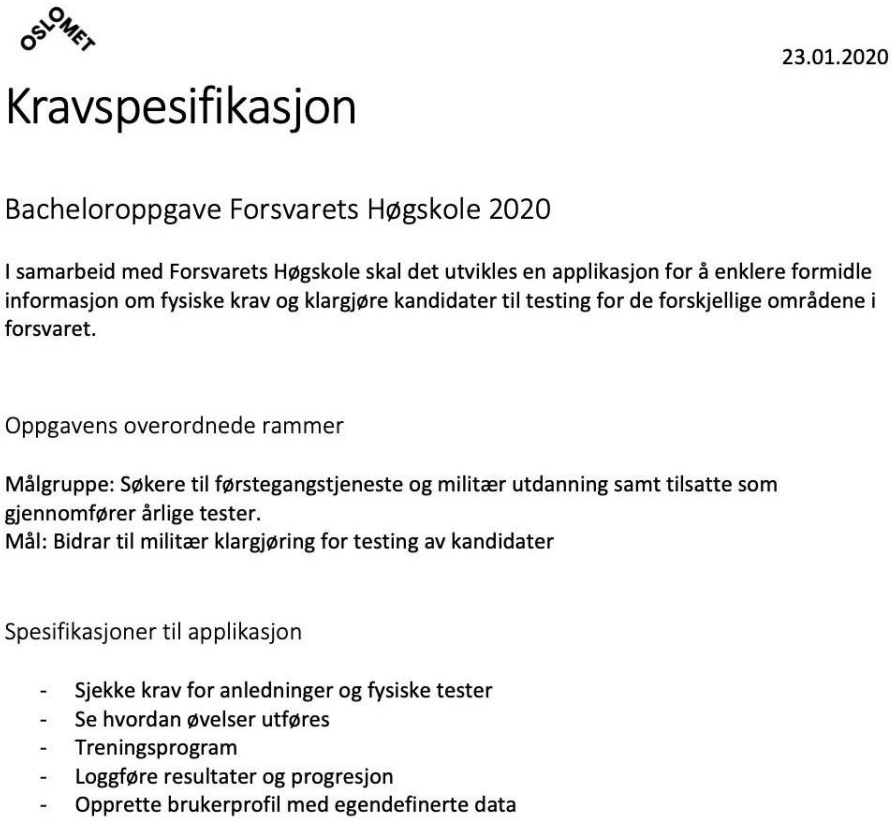
Ord / Begrep	Forklaring
API https://snl.no/API	Programvaregrensesnitt som lar en løsning benytte seg av tjenester tilbudt av en annen løsning.
Backend https://snl.no/backend	Delen av koden som ligger nærmest databasen. Håndterer det meste av tyngre logikk.

Branch https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-branches	En “gren” med kronologisk opprettet kode. En viktig del av versjonshåndtering ettersom man kan lage “avgreninger” av eksisterende branches og jobbe med kode som ikke påvirker de andre grenene. Master-branch er “hovedgrenen”.
Bugs https://en.wikipedia.org/wiki/Software_bug	Feil i kode som resulterer i uønsket oppførsel
Business-logikk https://en.wikipedia.org/wiki/Business_logic	Kode som bestemmer hvordan data skal kunne opprettes, lagres og endres
Continuous integration https://help.github.com/en/actions/building-and-testing-code-with-continuous-integration/about-continuous-integration	Et oppsett som sørger for at kode blir automatisk bygget, testet, utgitt og opprettholdt, men bare dersom testene er suksessfulle og koden er kjørbar
Frontend https://snl.no/frontend	Delen av koden som ligger nærmest brukeren og som former det man visuelt ser på skjermen
GDPR https://arbinn.nho.no/forretningsdrift/personvern/personopplysningsverktoy/personvernforordningen/	General Data Protection Regulation eller Personvernforordningen er en forordning som regulerer behandling av personopplysninger for å opprettholde personvern i EU og EØS
Hash https://en.wikipedia.org/wiki/Cryptographic_hash_function	En tekststreng som har gått gjennom en funksjon og blitt en bit tekststreng med en fast lengde, brukt for å kryptere passord

Issues https://help.github.com/en/enterprise/2.15/user/articles/about-issues	En koderelatert oppgave som skal gjennomføres, her relatert til prosjektet opprettet på Github
Delte preferanser https://pub.dev/packages/shared_preferences	En brukers sett med preferanser som blir lagret lokalt på enheten. Brukes blant annet for innstillinger
Merge https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/merging-a-pull-request	Å flette sammen to *branches* til én branch. Man kan selv velge hvilket innhold man ønsker at skal utgjøre den nye branchen.
Minimum Viable Product https://en.wikipedia.org/wiki/Minimum_viable_product	"Minste brukbare produkt", et produkt som kun inneholder grunnleggende funksjonalitet
Native https://searchsoftwarequality.techtarget.com/definition/native-application-native-app	En Native applikasjon er et program som er utviklet for bruk på en spesiell plattform eller enhet
noSQL https://www.ibm.com/cloud/learn/nosql-databases	Databaser som ikke følger den samme relasjonsoppbyggingen mellom tabeller som en SQL database gjør
Open-source https://opensource.com/resources/what-open-source	Noe alle kan modifisere og dele fordi oppsettet og oppbyggingen er offentlig tilgjengelig
Pull request https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests	En forespørsel om å *merge* kode inn i master branch, som må godkjennes før den kan merges
Repo / Repository	Et sted der kildekoden lagres online, i

https://help.github.com/en/github/getting-started-with-github/create-a-repo	vårt tilfelle hos GitHub. Man kan aksessere koden online, eller laste den ned til en lokal maskin
Salt https://en.wikipedia.org/wiki/Salt_(cryptography)	Tilfeldig data som gis som input til en hash funksjon sammen med hashen for at to like inputs skal gi forskjellig output
Singleton https://no.wikipedia.org/wiki/Singleton	Et designmønster som forhindrer instansiering av mer enn ett objekt til en gitt klasse
State management https://flutter.dev/docs/development/data-and-backend/state-mgmt/simple	Tilstandsbevaring av data i kjøretid
Webcontainer https://en.wikipedia.org/wiki/Web_container	Container som kjører inne i en applikasjon hvor JavaScript eller annet språk brukt for webutvikling kan kjøre på en plattform som vanligvis ikke støtter gitt språk.
Widget https://www.merriam-webster.com/dictionary/widget	En liten del av maskinvare designet for å bidra med spesifikk funksjonalitet.

2. Kravspesifikasjon



OSLOMET

23.01.2020

Kravspesifikasjon

Bacheloroppgave Forsvarets Høgskole 2020

I samarbeid med Forsvarets Høgskole skal det utvikles en applikasjon for å enklere formidle informasjon om fysiske krav og klargjøre kandidater til testing for de forskjellige områdene i forsvaret.

Oppgavens overordnede rammer

Målgruppe: Søkere til førstegangstjeneste og militær utdanning samt tilsatte som gjennomfører årlige tester.
Mål: Bidrar til militær klargjøring for testing av kandidater

Spesifikasjoner til applikasjon

- Sjekke krav for anledninger og fysiske tester
- Se hvordan øvelser utføres
- Treningsprogram
- Loggføre resultater og progresjon
- Opprette brukerprofil med egendefinerte data

Figur 1: Skjermbilde av dokumentet som utgjør appens kravspesifikasjoner.

3. Prosessdokumentasjon

3.1 Anskaffelse av prosjektet

Det var en lang prosess å finne en oppgave å skrive. Tidlig i prosessen sendte vi e-mail til følgende selskaper med spørsmål om vi kunne jobbe for dem med vårt bachelorprosjekt:

- Politiet
- Forsvaret
- Evry
- Accenture
- Bekk

- NetCompany
- Computas
- IBM
- Finn
- Datatilsynet
- Visma
- IKEA
- Bouvet
- Sanity
- FunCom

I de fleste tilfellene var vi relativt passive, og var mer på utkikk etter hvilke muligheter de ulike selskapene hadde for oss. Disse forespørslene ble hovedsakelig møtt med et nei, med unntak av IBM og Sanity som i utgangspunktet var interessert. Når det kom til Forsvaret og FunCom hadde vi en mer aktiv tilnærming, og kom selv med forslag til ting vi kunne jobbe med under prosjektperioden.

Til Forsvaret pitchet vi vår egen idé om en app vi mente de kunne ha bruk for. De viste interesse og kalte inn til et møte der vi gikk gjennom problemene de hadde, og mulige løsninger på dem. Etter en diskusjon innad i gruppen aksepterte vi samarbeidet.

3.2 Oppsummering av prosjektperioden

Fordi vi selv kom opp med ideen, manglet det fastsatte rammer rundt prosjektet vårt. Vi laget en kravspesifikasjon (Vedlegg 2) og presenterte denne for våre kontaktpersoner i Forsvaret. Produkteier hadde heller ingen krav til design og layout, men vi ble henvist til Forsvarets mediesenter som vi hentet inspirasjon fra. Det har med andre ord vært et prosjekt med svært frie tøyler. Dette har vært interessant og lærerikt, fordi vi har vært gjennom hele systemutviklingsprosessen fra idé til produkt, men det har også vært utfordrende da det har vært mange ting å stå ansvarlig for.

Det var også utfordrende og tidkrevende å lære oss et nytt programmeringsspråk. Vi brukte mye tid på å sette oss inn i språket før vi kom i gang med programmeringen,

men til tross for at dette tok lenger tid enn vi først forventet er det ikke et valg vi angrer på.

Milepælsplanen, som vi utarbeidet tidlig i prosjektperioden, ble raskt vanskelig å overholde. Det viste seg å være vanskelig å estimere hvor lang tid ting tok. Spesielt tok utviklingsprosessen langt lengre tid enn planlagt, fordi vi ikke hadde nok forhåndskunnskap om hvordan ting skulle gjøres. Likevel klarte vi å hente oss inn mot slutten, selv om en del utvikling måtte fortsette i perioden vi hadde satt av til rapportskriving.

En annen viktig årsak til at vi ble noe forsinket var at vi ikke fikk tilgang til digital data fra forsvaret. Vi håpet på å få kravdataene gjennom en databasedump, men det var ikke tilgjengelig av sikkerhets- og logistikkmessige årsaker. Derfor la vi inn all data i databasen selv, noe som var svært tidkrevende.

Selve utviklingen av appen gitt relativt greit for seg, ettersom vi hele tiden lærte mer om Flutter og fant bedre løsninger på problemer. Vi baserte oss på et Minimum Viable Product vi hadde utarbeidet sammen med Forsvaret, som skulle presentere opptakskrav til forskjellige anledninger i Forsvaret. Utover dette utviklet vi funksjonalitet for å få informasjon om øvelser og treningsprogram rettet mot spesielle anledninger, og funksjonalitet for å kunne loggføre treningsøkter med tilhørende øvelser i en treningsdagbok. En av utfordringene var å presentere store mengder tekst som samtidig skulle fremstå som ryddig og oversiktlig. Det var en krevende prosess, men vi opplever at vi løste dette på en god måte i det ferdige produktet.

Vi fikk ikke tid til å implementere annen funksjonalitet vi hadde diskutert, som for eksempel en kommunikasjonsdel for brukerne som kunne brukes til å kunne trene sammen, eller til å kunne chatte med andre brukere.

Under prosessen benyttet vi oss av Scrum med tilhørende sprinter. Mer detaljer om dette blir presentert i de siste delkapittelet i dette vedlegget. Både prosjektdagboken, møtenotatene, og prototypingsprosessen har også fått sine egne delkapitler.

3.3 Prosjektdagbok

Tabellen under viser en prosjektdagboken vår i løpet av prosjektperioden. I likhet med møtenotatene sluttet vi i stor grad med å skrive innføringer i prosjektdagboken etter utviklingsperioden begynte, ettersom vi fikk en mer hektisk hverdag. I flere tilfeller er heller ikke prosjektdagboken så detaljert som den kunne vært, men den hjalp oss likevel når vi så tilbake på hvordan prosjektet hadde gått.

Innføring	Dato
Offisiell opprettelse av utviklingsteamet	03.04.19
Kontaktet eventuelle produkteiere	01.09.10 - 31.10.19
Kontaktet eventuelle interne veiledere	01.09.19 - 30.09.19
Møte med Kyrre om potensiell veileder og potensiell oppgave	27.09.19
Møte / diskusjon med utviklingsteamet	10.10.19
Takket nei til Kyrre og hans oppgave	11.10.19
Leverte statusrapport til OsloMet	16.10.19
Fikk nytt medlem til utviklingsteamet	31.10.19
Fikk Torunn som intern veileder	07.11.19
Møte med produkteier	07.11.19
Leverte prosjektskisse til OsloMet	08.11.19
Kontaktet produkteier med oppdateringer og informasjon	03.01.20
Kontaktet intern veileder angående første veiledermøte	03.01.20

Første offisielle gruppemøte	09.01.20
Fullført mockup av prosjekt-nettside	14.01.20
Påbegynt utvikling av prosjekt-nettside	14.01.20
Gruppemøte	15.01.20
Møte med intern veileder	15.01.20
Gruppemøte	22.01.20
Brainstorming av design	22.01.20
Møte med produkteier	23.01.20
Skrev brukerhistorier	24.01.20
Retrospektiv av sprint + påbegynte ny sprint	24.01.20
Kontaktet intern og eksterne veiledere på mail	24.01.20
Møte med intern veileder	29.01.20
Gruppemøte	31.01.20
Fikk innkalling til møte med produkteier	03.02.20
Møte med produkteier	06.02.20
Gruppemøte	07.02.20
Ny sprint + bestemte konvensjoner for kode	07.02.20
Møte med intern veileder	12.02.20
Møte med intern veileder	26.02.20

Gruppemøte + ny sprint	26.02.20
Gruppemøte	04.03.20
Gruppemøte	18.03.20
Gruppemøte	25.03.20

3.4 Møtenotater

Tabellen under viser en kort oppsummering av møtenotatene for de gitte datoene i venstre kolonne. Etter vi begynte med utvikling av appen gjennomførte vi oftere møter som var kortere og mer “hands on”, og gradvis sluttet vi å føre møtenotater.

Dato	Deltakere	Oppsummering
07.11.19	Utviklingsteamet og produkteier	Diskuterte problemstillingen og mulige løsninger, “MVP”, målgrupper, design og hvordan Forsvaret fungerer.
09.01.20	Utviklingsteamet	Tok en bestemmelse angående teknologier og verktøy som skal brukes under prosjektperioden, opprettet et prosjekt på Github repo, startet å organisere Notion, opprettet scrum-brettet, planla fremtidige møter og snakket om regler innad i gruppen og gruppedynamikk.
15.01.20	Utviklingsteamet	Bestemte oss for programmeringsspråk, startet på prosjekt-nettsiden

		og begynte på samarbeidskontrakten som gjelder for oss og Forsvaret.
15.01.20	Utviklingsteamet og intern veileder	Presenterte prosjektet for veileder og planla hovedfunksjonaliteten til produktet.
22.01.20	Utviklingsteamet	Diskuterte ideer og jobbet med prototypene.
23.01.20	Utviklingsteamet og produkteier	Diskuterte de fysiske minstekravene og interne strukturer i Forsvaret. Planla appen i videre detalj og diskuterte mål og krav relatert til utvikling.
29.01.20	Utviklingsteamet og intern veileder	Presenterte kravspesifikasjonen til veileder og snakket videre om programmeringsspråk og valg av database.
31.01.20	Utviklingsteamet	Jobbet med prototypen og planla brukertesting.
06.02.20	Utviklingsteamet og produkteier	Diskuterte potensielle sikkerhetsproblemer, gikk gjennom kontrakten, diskuterte hvilke ressurser som er tilgjengelig for oss og ikke.

6.02.20	Utviklingsteamet	Oppsummerte gårsdagens møte med produkteier, diskuterte brukertesting og hvilket program vi skulle bruke til å skrive selve hovedoppgaven.
07.02.20	Utviklingsteamet	Diskuterte databasestruktur og ER-diagrammer, planla første sprint og fordelte oppgaver innad i gruppen.
12.02.20	Utviklingsteamet og intern veileder	Oppsummerte tidligere møte med produkteier for veileder, diskuterte ER-diagram.
26.02.20	Utviklingsteamet og intern veileder	Presenterte fremgangen vi har hatt og snakket om fremtidige planer.
26.02.20	Utviklingsteamet	Planla fremtidig utvikling og jobbet med utvikling av appen.
04.03.20	Utviklingsteamet	Diskuterte universell utforming, selve hovedoppgaven, design og fremgangen vi har hatt kontra hvor vi skulle ha vært i følge milepælsplanen. Jobbet videre med utvikling av appen.
18.03.20	Utviklingsteamet	Diskuterte fremgang, jobbet med utvikling av appen.

25.03.20	Utviklingsteamet	Diskuterte fremgang i forhold til milepælsplanen, jobbet med utvikling av appen.

3.5 Milepælsplan

Tabellen under viser prosjektets milepælsplan fra starten av utviklingsperioden til appen er ferdig og rapporten er levert. Vi har basert oss på at ting skal være ferdig fredagen i den spesifiserte uken.

I tabellen er hver av de ulike funksjonalitetene delt inn i opp til tre sub-milepæler. At en funksjonalitet er ferdig designet betyr at vi vet hva vi skal lage, at den er ferdig utviklet betyr at vi har implementert den i appen og kodet ferdig layout osv, og at den har fått funksjonalitet betyr at den er koblet til andre funksjoner i appen og at den fungerer som planlagt.

Hendelse / Funksjonalitet	Uke fullført
Profilsiden er ferdig designet	9
Krav-siden er ferdig designet	9
Forsiden er ferdig designet	9
Forsiden er ferdig utviklet	10
Den lokale databasen har fått innlagt informasjon	10
Profil-databasen er opprettet	11
Profilsiden er ferdig utviklet	11
Login siden er ferdig designet	11
Login siden er ferdig utviklet	11
Treningsdagboken er ferdig designet	11

Treningsplan-siden er ferdig designet	11
Funksjonalitet for søking fra forside er implementert	11
Profil databasen har fått endpoint	12
Øvelses-siden er designet	12
Øvelses-siden er utviklet	12
Treningsdagbok-siden er ferdig utviklet	12
Treningsplan-siden er ferdig utviklet	12
Øvelses-siden har fått funksjonalitet	13
Funksjonalitet for endring og sletting av profil er implementert	13
Login side har funksjonalitet for å logge inn	13
Treningsdagbok-siden har fått funksjonalitet	15
Treningsplan-siden har ferdig funksjonalitet	15
Error page er designet	16
Error page er utviklet	16
App ferdig	17
Rapport ferdig	21
Rapport levert	21
Muntlig presentasjon av prosjekt	24

I løpet av prosjektet har vi ved flere anledninger sklidd litt vekk fra milepælsplanen, ettersom ting har tatt lenger tid eller uforutsette hendelser har oppstått. I disse

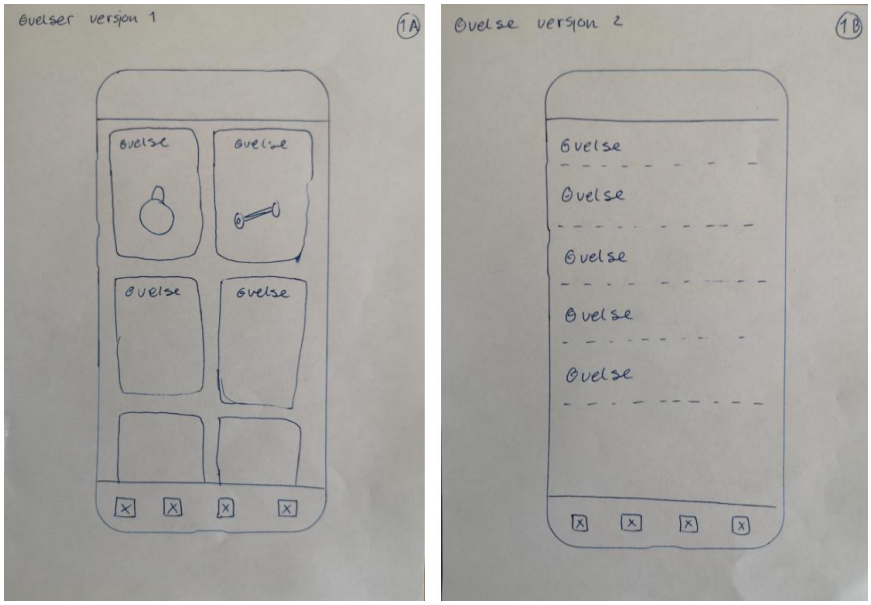
situasjonene har vi likevel hatt godt bruk for planen, ettersom den har hjulpet oss med å få oversikt over hvordan vi burde ligge an.
ed merging inn til master, slik at vi hele tiden vet at master-*branchen* er i orden.

3.6 Prototyping

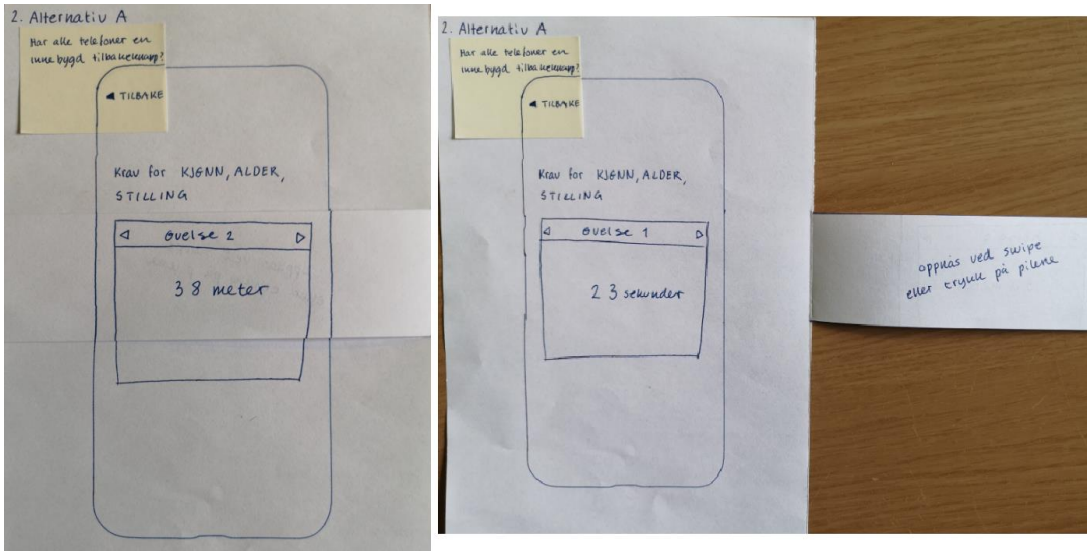
Prototyping har vært en stor del av prosjektet vårt fra start til slutt, og det har hjulpet oss med utformingen av appen. Appen har endres deg veldig i løpet av prosjektperioden, og vi vil argumenter for at den har endret seg til det bedre. Dette er takket være prototypingen i seg selv, og brukertestene vi har gjennomført ved bruk av prototypene.

3.6.1 Papirprototyper

Tidlig i prosessen skisserte vi enkle papirprototyper som omhandlet appens funksjonalitet, og vi lagde ofte flere varianter av samme funksjonalitet for å vurdere hva som fungerte best (Figur 1). Vi lagde også prototyper som hadde en viss grad av interaksjon ved seg (Figur 2).

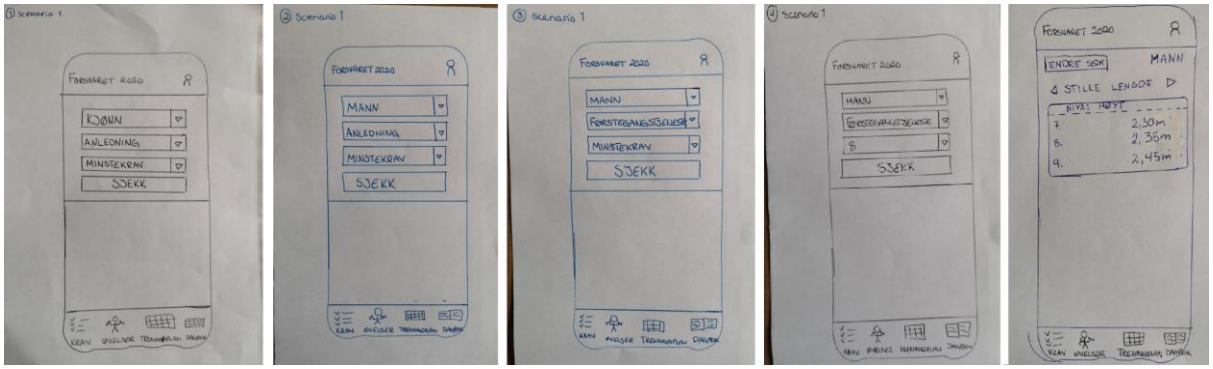


Figur 1: To tidlige lo-fi papirprototyper som viser to mulige alternativer vi kunne fremstille øvelsene på, enten i “kort” som stod ved siden av hverandre i et rutenett, eller som en liste med elementer under hverandre.

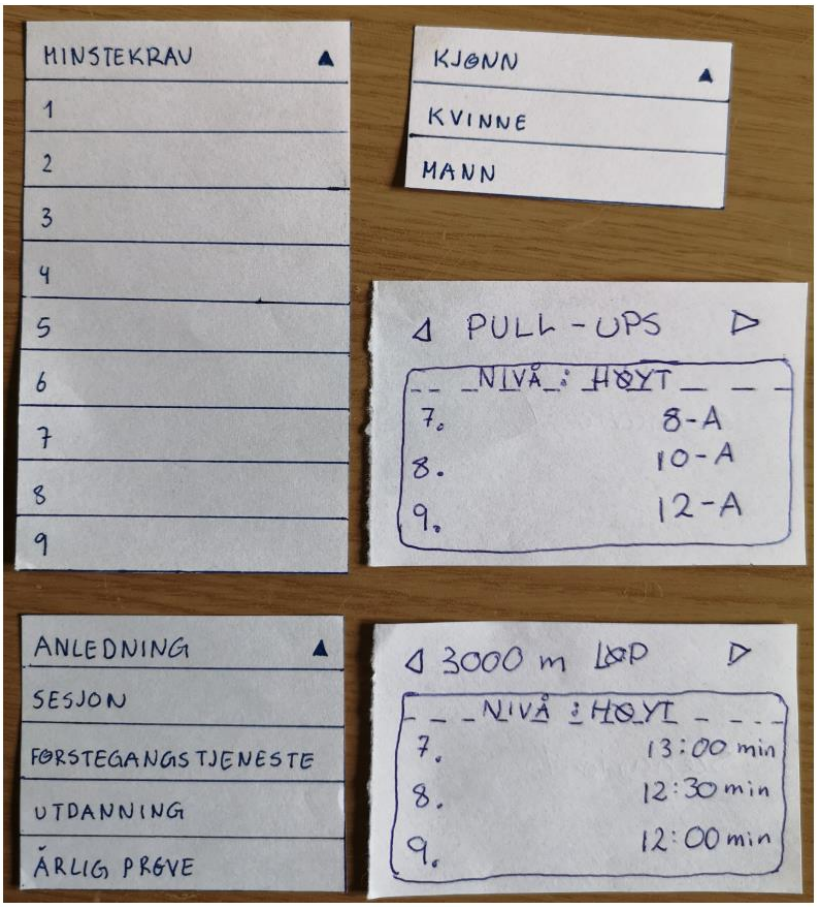


Figur 2: En tidlig lo-fi papirprototype som viser hvordan vi originalt planla å vise de fysiske minstekravene. I denne prototypen skulle hver øvelse få tildelt et "kort" der minstekravet sto, som brukeren kunne sveipe gjennom.

Vi hadde planer om å gjennomføre brukertester av kravsøket ved hjelp av mer kompliserte papirprototyper (Figur 3-4). Denne prototypen var delt inn i tre scenarioer der hvert scenario presenterte et predefinert søk brukeren skulle foreta seg. En fullstendig prototype for tre søk resulterte i svært mange ark og papirbiter, så vi bestemte oss for å migrere over til en digital prototype for å kunne effektivisere gjennomføringen av brukertestene.



Figur 3: Arkene som hører til scenario 3 av protoypen vi originalt skulle benytte i brukertestene. Dette scenarioet krevet at brukeren velger verdiene kjønn = Mann, anledning = Førstegangstjeneste og minstekrav = 8. Det siste bildet i rekken viser hvordan dataene skulle bli presentert.



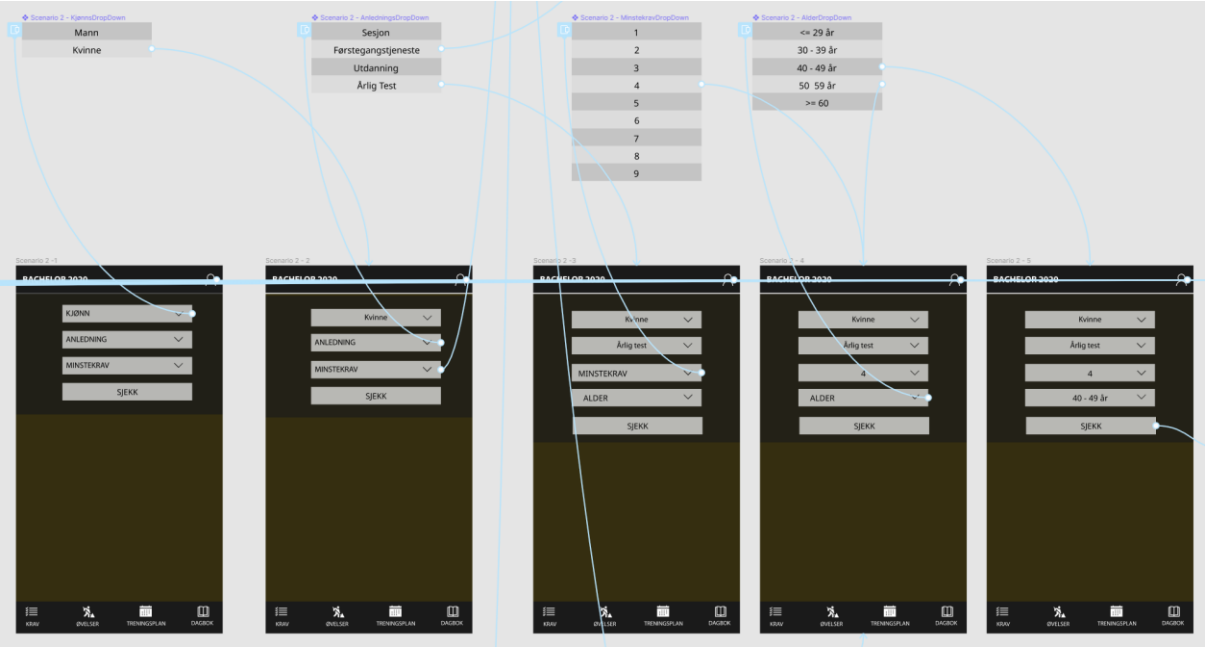
Figur 4: Innholdet i nedtrekksmenyene fra figur 3, i tillegg til to presentasjoner av minstekrav. Disse papirbitene skulle brukes under brukertesten, der en person som handlet som en “datamaskin” skulle legge ut de relevante menyene når brukeren “trykker” på dem.

3.6.2 Digital prototype i Figma

Den digitale prototypen er laget i Figma og er en medium-fidelity prototype. Prototypen er satt sammen av ulike “bokser” som representerer skjermbilder i appen. “Skjermbildene” har blant annet bilder, tekst og knapper på seg, og er koblet sammen ved hjelp av innebygget funksjonalitet i Figma (Figur 1).

Figur 1: Et skjermbilde tatt fra Figma som viser oversikten over den digitale prototypen, og hvordan de ulike “boksene” er koblet sammen.

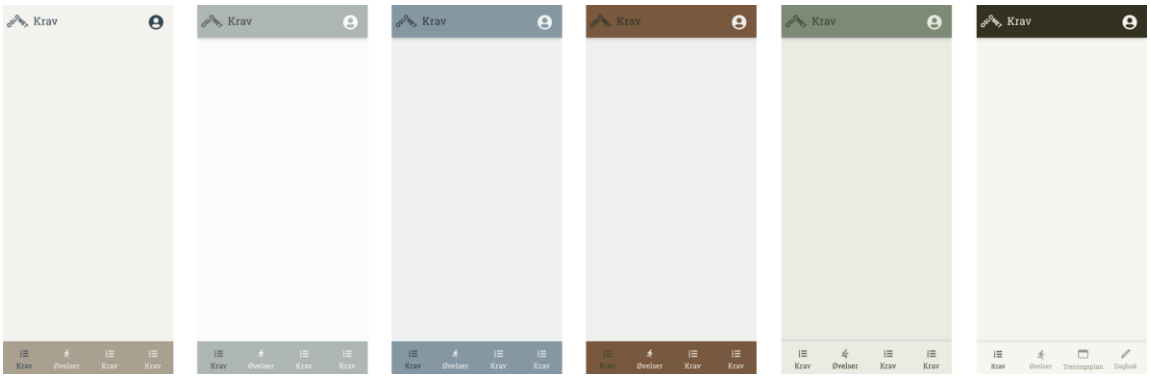
I likhet med papirprototypen er også denne prototypen er en gjennomgang av kravsøket, og den har innebygd funksjonalitet for tre ulike søk som stemmer med spørsmålene som ble spurt under de tidlige brukertestene. Vi definerte hvilke knappetrykk som var “riktige” (Figur 2), og ingen andre knapper har funksjonalitet. Dette skaper en illusjon om at prototypen faktisk fungerer som et ordentlig produkt, men kun om man tar de rette stegene for å oppnå et gitt mål.



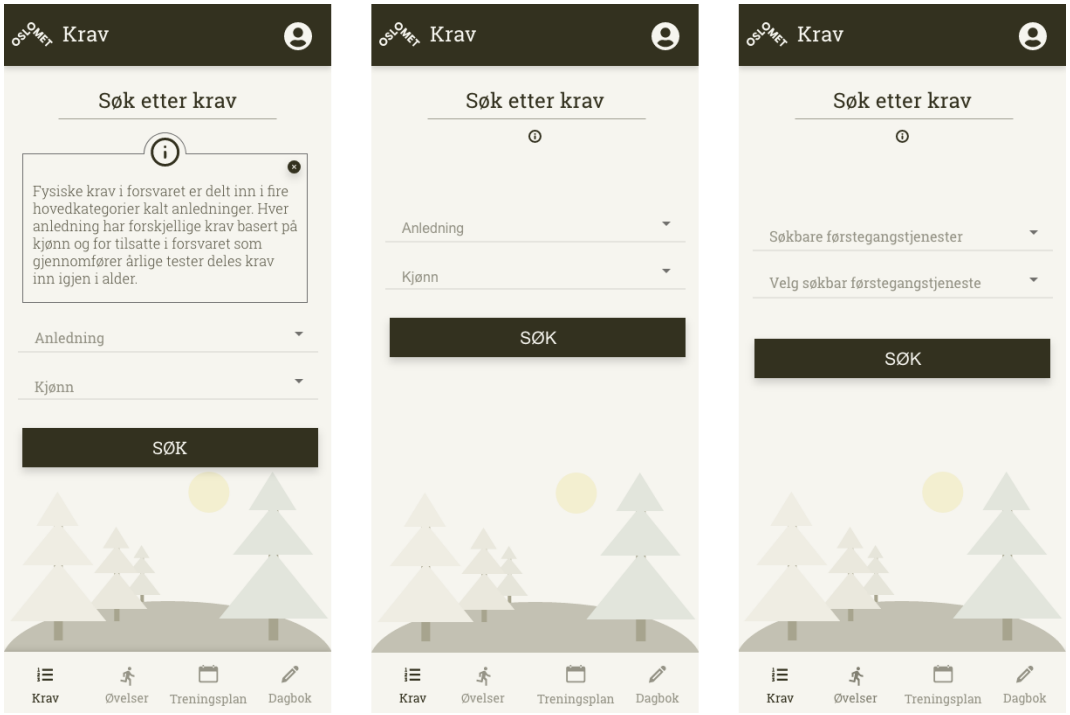
Figur 2: “Skjermbildene” som utgjør filteret / kravsøket til ett av de predefinerte søkene. Den nederste raden viser nedtrekksmenyene før og etter man har valgt et alternativ, og de øverste raden viser innholdet i de ulike menyene. Dette bestemte søket krever at brukeren velger verdiene kjønn = Kvinne, anledning = Årlig test, minstekrav = 4 og alder = 40 - 49 år.

3.6.3 Design-prototype i Adobe XD

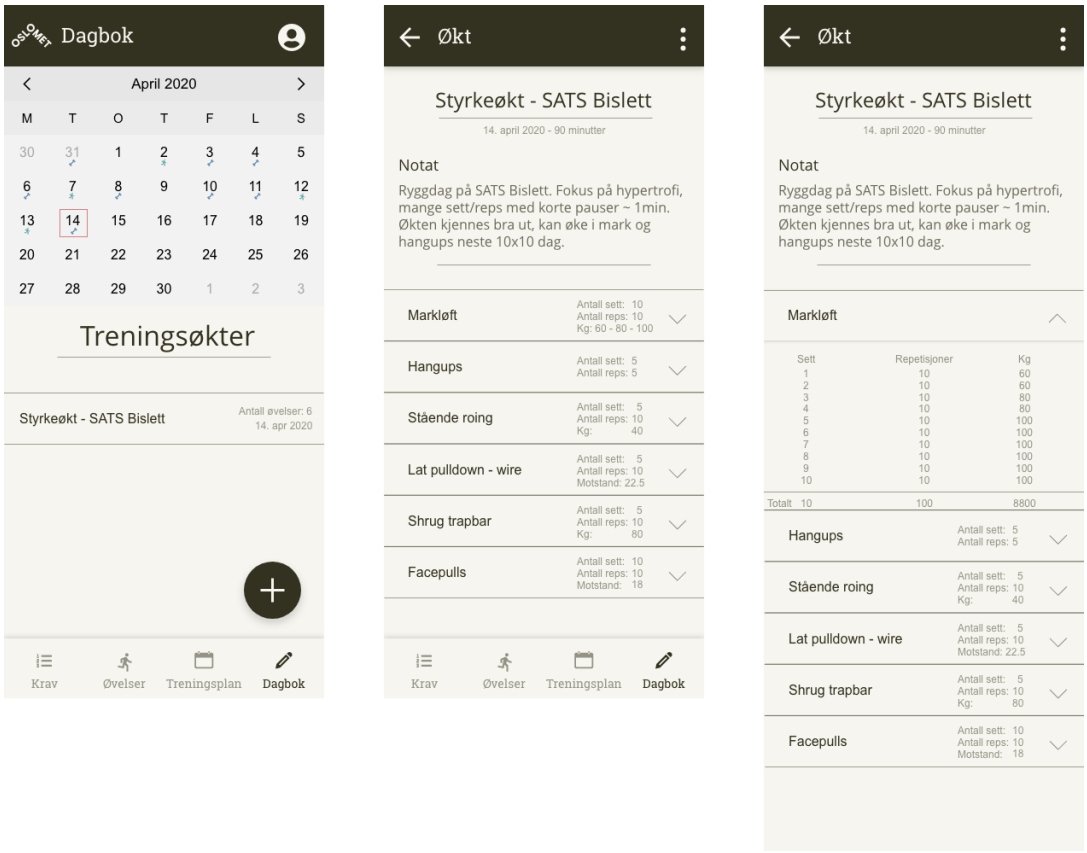
I tillegg til de allerede nevnte prototypene, benyttet vi oss også av Adobe XD for å lage ulike design-prototyper (Figur 1 - 4) , der vi spesifikt så på utforming og fargebruk. Også disse prototypene førte til store endringer i appen, og vi endte opp med et mer “clean” design.



Figur 1: Forskjellige fargealternativer som ble laget for å se hvilke farger som gjorde seg best i appen.



Figur 2: Siden for kravsøk med og uten en ekspandert infoboks.

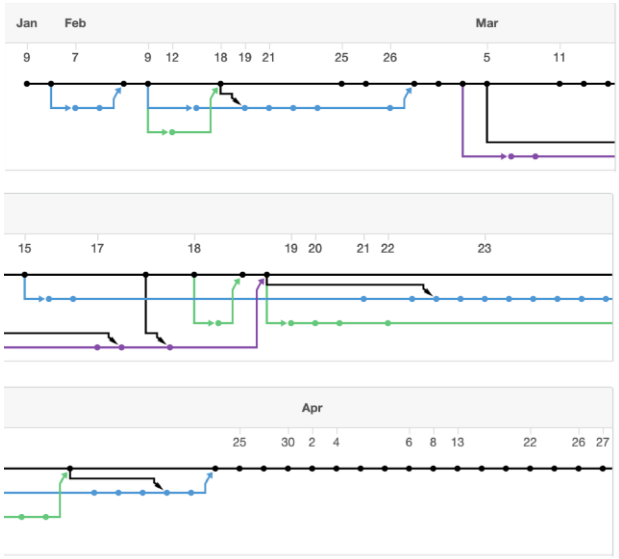


Figur 3: Dagboksiden og kalender med symboler for registrerte kondisjons- og styrketreninger. Treningsøkter åpnes i eget vindu hvor notater og øvelser vises.

Figur 4: Resultat fra søk for kystjeger.

3.7 Workflow

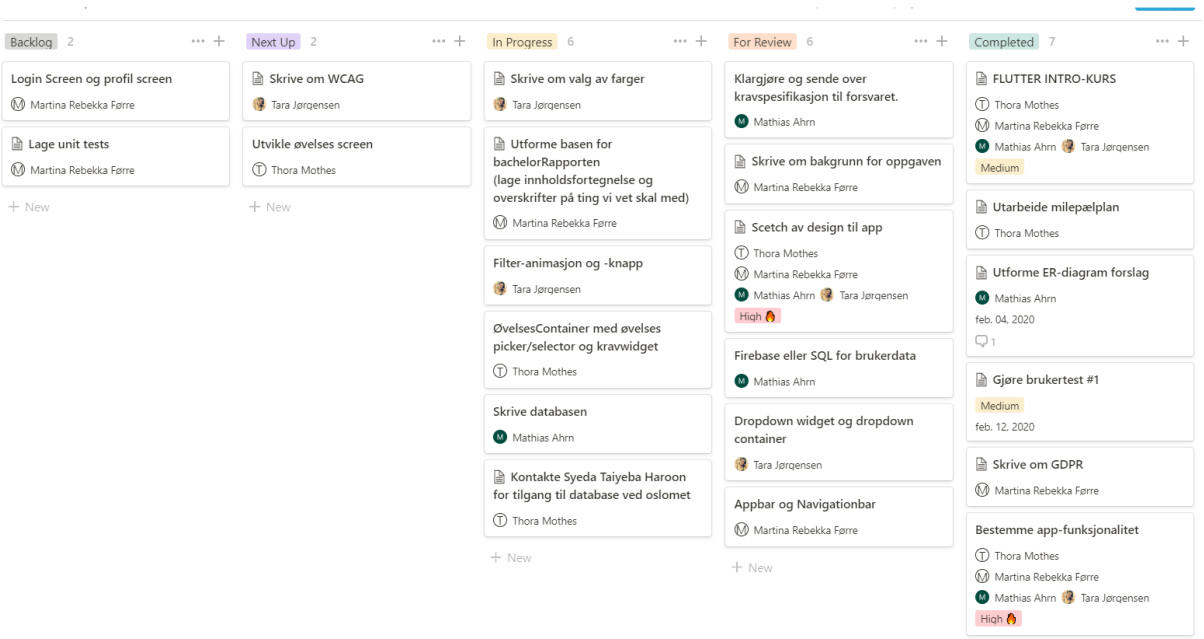
Prosjektet er satt opp med Kanban-project på Github. Alt av kode som har blitt produsert har blitt delt opp i mindre oppgaver/*issues,* og er blitt utviklet i feature-branches. Ved ferdigutviklede oppgaver opprettes pull requests slik at alle på gruppen har mulighet til å se igjennom både kode og funksjonalitet før det *merges* inn i master-branchen. På denne måten har vi en ryddigere historikk og mer oversikt over hva som er gjort og når.



Figur 1: Et utsnitt av historikken til Github-repoet.

Som et ekstra kvalitets- og sikkerhetsaspekt er også continuous integration (CI) implementert via Github Actions. All kode som pushes til en pull request blir testet med enhetstestene i prosjektet og deretter bygges kode-versjonen for å sjekke at den fungerer. Denne prosessen kjøres også igjen ved merging inn til master, slik at vi hele tiden vet at master-branchen er i orden.

3.8 Sprint-planlegging



Figur 1: Skjermbilde fra kanban-brettet vårt i Notion, som vi benyttet for å holde oversikt på oppgaver under hver sprint.

3.9 Sprint-oversikt

Sprint	Tid	Innhold
Planleggingssprint	1 måned	• Møte med forsvaret • Bestemme utviklingsverktøy • Kurs i Flutter • Lage milepælsplan
Utvilingsprint 1	2 uker	• Design av profilside, krav, forside • Forside utviklet • Data inn i lokal database
Utviklingssprint 2	2 uker	• Opprette ekstern database

		• Utviklet profil • Designe øvelsesside, treningsdagbok og treningsplan
Utviklingssprint 3	2 uker	• Utvikle øvelsesside • Profilfunksjonalitet: endring og slett • Utvikle autentisering
Utviklingssprint 4	3 uker	• Utvikle treningsdagbok, treningsplan, errorpage
Rapportsprint	1 måned	• Rapport • App bugfix

4. Full suksesskriterie-analyse

Tabellen under viser alle de gjeldende suksesskriteriene fra WCAG 2.0, i tillegg til de suksesskriteriene fra WCAG 2.1 det er aktuelt å ta med i norsk regelverk. Tabellen viser også hvor godt vi har implementert de gitte suksesskriteriene ved ferdig prosjektperiode (mai 2020). Kolonnen til høyre er fargekodet basert på grad av implementasjon / oppnåelse: Grønt = fullt implementert / oppnådd, orange = delvis implementert / oppnådd, krever videreutvikling, rødt = ikke implementert / oppnådd, krever videreutvikling.

Suksesskriterie	Nivå	Implementering
1.1.1 - Ikke-tekstlig innhold (Gi brukeren et tekstalternativ for innhold som ikke er tekst)	Nivå A	Alle bilder og figurer som har en funksjon utover å bare være pynt har en tekstlig beskrivelse, enten i koden selv, eller linket til ikonet på en eller annen måte (som i

		navigasjonsbaren). Unntaket er "profil"-ikonet, som ikke har en tekst i frontend som indikerer at man blir tatt til profilen.
1.2.1 - Bare lyd og bare video (Gi brukeren et alternativ når innholdet presenteres kun som video eller lyd)	Nivå A	Appen inneholder ingen lydklipp
1.2.2 - Teksting (Tilby teksting for video med lyd)	Nivå A	Appen inneholder ingen lydklipp
1.3.1 - Informasjon og relasjoner (Ting skal være kodet som det det ser ut som)	Nivå A	Vi har benyttet oss av ferdiglagde widgets eller kodet våre egne og navngitt dem med navn som beskriver funksjonaliteten. Knapper ser ut som knapper, nedtrekksmenyer ser ut som nedtrekksmenyer, skjemaer ligger inne i "forms"-widgets, osv.
1.3.2 - Meningsfylt rekkefølge (Presenter innhold i en meningsfull rekkefølge)	Nivå A	Innholdet er presentert enten ved hjelp av alfabetisk sortering, eller basert på hvilken informasjon som er viktigst.
1.3.3 - Sensoriske egenskaper (Instruksjoner må ikke utelukkende være avhengige av form, størrelse, visuell plassering, orientering, eller lyd for å kunne bli forstått)	Nivå A	Innholdet i appen består av en blanding av ulike sensoriske egenskaper, med unntak av profil-ikonet som tidligere nevnt, og vektstang-ikonene som viser hvorvidt en økt er fullført eller ikke (Grønt ikon indikerer at økten er

		fullført, sort ikon illustrerer at økten *ikke* er fullført.). Denne informasjonen er tilgjengelig andre steder, så det formidles likevel.
1.3.4 - Visningsretning (Brukeren må få velge om innholdet skal vises i liggende eller stående retning)	Nivå AA	Vi har foreløpig lagt inn en sperre som forhindrer at appen fungerer i landskapsmodus.
1.3.5 - Identifiser formål med inndata (Skjemaelementer er kodet med inndata)	Nivå AA	Feltene i skjemaene som skal fylles ut har “hint-tekst” som indikerer hva slags informasjon som skal fylles inn (Figur 1)
1.4.1 - Bruk av farge (Ikke bruk presentasjon som bygger utelukkende på farge)	Nivå A	Innholdet i appen blir aldri presentert med bare bruk av farge, med unntak av vektstang-ikonene som viser hvorvidt en økt er fullført eller ikke (Grønt ikon indikerer at økten er fullført, sort ikon illustrerer at økten *ikke* er fullført.). Denne informasjonen er tilgjengelig andre steder, så det formidles likevel.
1.4.2 - Styring av lyd (Gi brukeren mulighet til å stoppe eller pause lyd som starter automatisk)	Nivå A	Appen inneholder ingen lydklipp
1.4.3 - Kontrast (Kontrastforholdet mellom teksten og bakgrunnen er minst 4,5:1)	Nivå AA	All tekst har minst et kontrastforhold på 4,5:1 til bakgrunnen den står på. Dette sjekket vi ved hjelp av programmet Colour Contrast

		Analyser (CCA)
1.4.4 - Endring av tekststørrelse (Tekst kan bli endret til 200% størrelse uten tap av innhold eller funksjon)	Nivå AA	Vi har benyttet oss av media queries, og endrer blant annet layout og skriftstørrelse basert på bredde i pixler, men vi har ikke gjennomført en test som baserer seg på endringer i telefoninnstillingene vil gjøre med appens utseende. Appen inneholder heller ikke innstillinger til å øke skriftstørrelse eller å zoome.
1.4.5 - Bilder av tekst (Bruk tekst i stedet for bilder av tekst)	Nivå AA	All tekst i appen er faktisk tekst, og ikke bilder av tekst
1.4.10 - Dynamisk tilpasning (Innhold skal kunne endres til 400 prosent størrelse uten tap av informasjon eller funksjonalitet)	Nivå AA	Vi har benyttet oss av media queries, og endrer blant annet layout og skriftstørrelse basert på bredde i pixler, men vi har ikke gjennomført en test som baserer seg på endringer i telefoninnstillingene vil gjøre med appens utseende. Appen inneholder heller ikke innstillinger til å øke skriftstørrelse eller å zoome.
1.4.11 - Kontrast for ikke-tekstlig innhold (Ikke-tekstlig innhold skal ha et kontrastforhold på minst 3:1 mot farge(r) som ligger siden av)	Nivå AA	Vi har benyttet oss av illustrerende bilder med lav kontrast for at de ikke skal ta fokus fra innholdet, men bare eksistere i bakgrunnen. Disse har vi ikke sjekket kontrasten på. Alle andre ikoner og bilder har riktig kontrastforhold.
1.4.12 - Tekstavstand	Nivå AA	Vi har ikke gjennomført en test som

(Tekstavstanden skal kunne overstyres for å gjøre teksten lettere å lese)		baserer seg på endringer i telefoninnstillingene vil gjøre med appens utseende, og appen inneholder ikke innstillinger til å øke tekstavstand.
1.4.13 - Pekerfølsomt innhold eller innhold ved tastaturfokus (Brukeren skal ha mer kontroll over innholdet på nettsiden som får fokus med musepeker eller tastatur)	Nivå AA	Vi har ikke gjennomført fulle tester ved bruk av tastatur, og har heller ikke fokusert på utvikling rettet mot tastaturbruk.
2.1.1 - Tastatur (All funksjonalitet skal kunne brukes kun ved hjelp av tastatur)	Nivå A	Vi har ikke gjennomført fulle tester ved bruk av tastatur, og har heller ikke fokusert på utvikling rettet mot tastaturbruk.
2.1.2 - Ingen tastaturfelle (Unngå tastaturfeller)	Nivå A	Vi har ikke gjennomført fulle tester ved bruk av tastatur, og har heller ikke fokusert på utvikling rettet mot tastaturbruk.
2.1.4 - Hurtigtaster som består av ett tegn (Brukeren skal enkelt kunne slå av hurtigtaster som består av ett tegn)	Nivå A	Appen inneholder ikke hurtigtaster
2.2.1 - Justerbar hastighet (Tidsbegrensninger skal kunne justeres av brukeren)	Nivå A	Appen inneholder ikke tidsbegrensninger
2.2.2 - Pause, stopp, skjul (Gi brukeren mulighet til å stoppe, pause eller skjule innhold som	Nivå A	Appen inneholder ikke innhold som automatisk endrer seg. Videoene som viser gjennomføringen av

automatisk endrer seg)		øvelser vil spilles av kun om brukeren trykker på dem, og de kan settes på pause.
2.3.1 - Terskelverdi på maksimalt tre glimt (Innhold skal ikke blinke mer enn tre ganger per sekund)	Nivå A	Appen inneholder ikke innhold som blinker
2.4.1 - Hoppe over blokker (Gi brukeren mulighet til å hoppe direkte til hovedinnholdet)	Nivå A	Brukeren kan hoppe til den siden de ønsker ved hjelp av navigasjonsmenyen, og hovedinnholdet vil alltid være plassert øverst. Sidene er ikke lange som på en nettside, så å hoppe over innhold vil ikke være relevant.
2.4.2 - Sidetitler (Bruk nyttige og tydelige sidetitler)	Nivå A	Alle sidene har korte, konsise og relevante titler som samsvarer med sidens innhold (for eksempel “Krav”, “Ny Økt”, og “Profil”)
2.4.3 - Fokusrekkefølge (Presenter innholdet i en logisk rekkefølge)	Nivå A	Innholdet er presentert enten ved hjelp av alfabetisk sortering, eller basert på hvilken informasjon som er viktigst.
2.4.4 - Formål med lenke (Alle lenkers mål og funksjon fremgår tydelig av lenketeksten)	Nivå A	Tekst på knapper og merkelapper som fungerer som lenker til andre sider er beskrivende, som argumentert for tidligere i rapporten.
2.4.5 - Flere måter (Tilby brukeren flere måter å navigere på)	Nivå AA	Appen støtter navigasjon med trykk, og teoretisk sett navigasjon med tastatur. Vi har ikke gjennomført fulle tester ved bruk av tastatur, og har

		heller ikke fokusert på utvikling rettet mot tastaturbruk.
2.4.6 - Overskrifter og ledetekster (Sørg for at ledetekster og overskrifter er beskrivende)	Nivå AA	Alle sidene har korte, konsise og relevante overskrifter som samsvarer med sidens innhold (for eksempel “Krav”, “Ny Økt”, og “Profil”). Tekst på knapper og merkelapper er også beskrivende, som argumentert for tidligere i rapporten.
2.4.7 - Synlig fokus (Sørg for at alt innhold får synlig fokus når du navigerer med tastatur)	Nivå AA	Vi har ikke gjennomført fulle tester ved bruk av tastatur, og har heller ikke fokusert på utvikling rettet mot tastaturbruk.
2.5.1 - Pekerbevegelser (Innhold på nettsiden skal kunne brukes med enkel pekerinput)	Nivå A	Alt innhold i appen kan aksesseres ved et enkelt-trykk med fingeren, og krever ikke dobbelt-trykk eller at man holder inne
2.5.2 - Pekeravbrytelse (Uheldige og feilaktige input via mus eller berøringsskjerm skal lettere kunne forhindres)	Nivå A	Alle feiltrykk fra brukerens side skal lett kunne fikses opp i ved å trykke igjen (for eksempel på en sjekkboks). Brukeren vil også bli spurt om bekreftelse før man for eksempel avbryter skjemautfylling eller sletter et element (Figur 2).
2.5.3 - Ledetekst i navn (Brukere som bruker visuelle ledetekster skal også kunne bruke kodede ledetekster)	Nivå A	Tekst på knapper og merkelapper er beskrivende og relatert til funksjon, og navnene i koden er også beskrivende, enten ved at vi benytter

		oss av ferdige widgets eller ved at vi har navngitt våre egne widgets med forståelige navn.
2.5.4 - Bevegelsesaktivering (Funksjonalitet som kan betjenes med å bevege enheten eller ved brukerbevegelse, skal også kunne betjenes med brukersnittkomponenter)	Nivå A	Appen inneholder ikke bevegelsesaktiverte aktiviteter
3.1.1 - Språk på siden (Sørg for at språket til innholdet på alle websider er angitt i koden)	Nivå A	Vi har ikke spesifisert norsk språk i koden.
3.1.2 - Språk på deler av innhold (Sørg for at alle deler av innholdet som er på et annet språk enn resten av siden er markert i koden)	Nivå AA	Appen inneholder ikke flere språk
3.2.1 - Fokus (Når en komponent kommer i fokus medfører dette ikke automatisk betydelige endringer i siden)	Nivå A	Ingen betydelige endringer skjer i appen basert på hvilke elementer som står i fokus
3.2.2 - Inndata (Endring av verdien til et skjemafelt medfører ikke automatisk betydelige endringer i siden)	Nivå A	Ingen betydelige endringer skjer i appen basert på inndata, utenom vanlige endringer man forventer at vil skje basert på konvensjoner. Ekspanderbare panel ekspanderes / slås sammen og noen valg i nedtrekksmenyer vil legge til et nytt nedtrekks-felt.

3.2.3 - Konsekvent navigering (Navigasjonslinker som gjentas på flere sider skal ha en konsekvent rekkefølge)	Nivå AA	Alle typer knapper ser like ut, alle typer linker ser like ut og alle typer navigasjonselementer i navigasjonsmenyene bruker ikoner som er utformet i samme stil, takket være appens theme.
3.2.4 - Konsekvent identifikasjon (Elementer som har samme funksjonalitet på tvers av flere sider er utformet likt)	Nivå AA	All tekst av samme funksjon (tittel, brødtekst osv) ser lik ut, alle typer knapper ser like ut, alle typer linker ser like ut osv., takket være appens theme.
3.3.1 - Identifikasjon av feil (For feil som oppdages automatisk må du vise hvor feilen har oppstått og gi en tekstbeskrivelse av feilen)	Nivå A	Dersom en feil oppstår vil det komme en popup-melding som beskriver feilen. Denne meldingen vil være relatert til hvor feilen oppstod og hvilken handling som utløste den (Figur 3).
3.3.2 - Ledetekster eller instruksjoner (Det vises ledetekster eller instruksjoner når du har skjemaelementer som må fylles ut)	Nivå A	Feltene i skjemaene som skal fylles ut har “hint-tekst” som indikerer hva slags informasjon som skal fylles inn
3.3.3 - Forslag ved feil (Dersom feil blir oppdaget automatisk, gi brukeren et forslag til hvordan feilen kan rettes)	Nivå AA	Dersom en feil oppstår vil det komme en popup-melding som beskriver feilen. Denne meldingen vil også komme med forslag til hva brukeren kan gjøre (Figur 3)
3.3.4 - Forhindring av feil (For sider som medfører juridiske	Nivå AA	Brukeren kan velge å godkjenne / ikke godkjenne at dataene lagres, og

forpliktelser må det være mulig å kunne angre, kontrollere eller bekrefte dataene som sendes inn)		kan ved senere anledning gjøre endringer i informasjonen eller slette brukerprofilen helt.
4.1.1 - Parsing (Alle sider skal være uten store kodefeil)	Nivå A	Appen har ingen store kode-feil som gjør at den krasjer eller liknende
4.1.2 - Navn, rolle, verdi (Alle komponenter har navn og rolle bestemt i koden)	Nivå A	Navn på widgets er beskrivende og relatert til funksjon, enten ved at vi benytter oss av ferdige widgets med beskrivende navn eller ved at vi har navngitt våre egne widgets basert på deres funksjon.
4.1.3 - Statusbeskjeder (Brukeren skal få statusbeskjeder om viktige endringer på nettsiden uten at det gir kontekstendring)	Nivå AA	Appen inneholder ikke statusbeskjeder i den vanlige betydningen, og i de tilfellene brukeren får en beskjed på skjermen vil det være en feilmelding som ble utløst av en spesifikk handling, og som har en tilhørende forklaring.

Figur 1: Hvert inputfelt i et skjema har et hint som indikerer hva slags informasjon brukeren skal skrive inn i det gitte feltet.

Figur 2: Brukeren blir presentert med popup-vinduer som krever bekreftelse dersom de trykker at de vil slette et element fra appen.

Figur 3: Brukeren blir presentert med en forklarende feilmelding dersom en feil skulle oppstå. I dette tilfellet er det en feil med brukernavnet og / eller passordet som ble skrevet inn, og brukeren blir oppfordret til å prøve igjen.

5. Tilgjengelighetserklæring

Alle skal ha like muligheter til å bruke appen. Vårt mål er å gjøre innholdet så tilgjengelig som mulig for alle brukere.

Vi skal:

- Sørge for at nåværende innhold oppfyller minst Nivå AA i WCAG 2.0 og 2.1
- Sørge for at nytt innhold i appen oppfyller minst nivå AA i WCAG 2.0 og 2.1
- Følge forskrift om universell utforming av IKT.
- Inkludere universell utforming i valget av eventuelle tredjepartssystemer og oppgraderinger av eksisterende systemer

Dette må vi rette opp i:

- Tilrettelegge for landskapsmodus
- Sørge for at appen kan kan benyttes utelukkende med tastatur, selv om dette hovedsakelig er viktig for nettsider og ikke apper
- Sørge for at ikoner og bilder har alternativ tekst som kan leses opp av en skjermleser
- Sørge for at ingen elementer fremstilles med kun én sensorisk egenskap, men at alle elementer er en blanding av egenskaper som ikke utelukker noen brukere.
- Reflektere brukerens telefoninnstillinger i appen, relatert til zoom og skriftstørrelse
- Sørge for at alle elementer, inkludert ikke-tekstlige, oppnår rett kontrastnivå
- Sørge for at norsk språk blir spesifisert i koden, slik at en skjermleser vil lese teksten på norsk

Tilgjengelighetsprinsippene vi følger:

- Vi følger per mai 2020 33 av 47 pålagte suksesskriterier.

6. Kodeeksempler

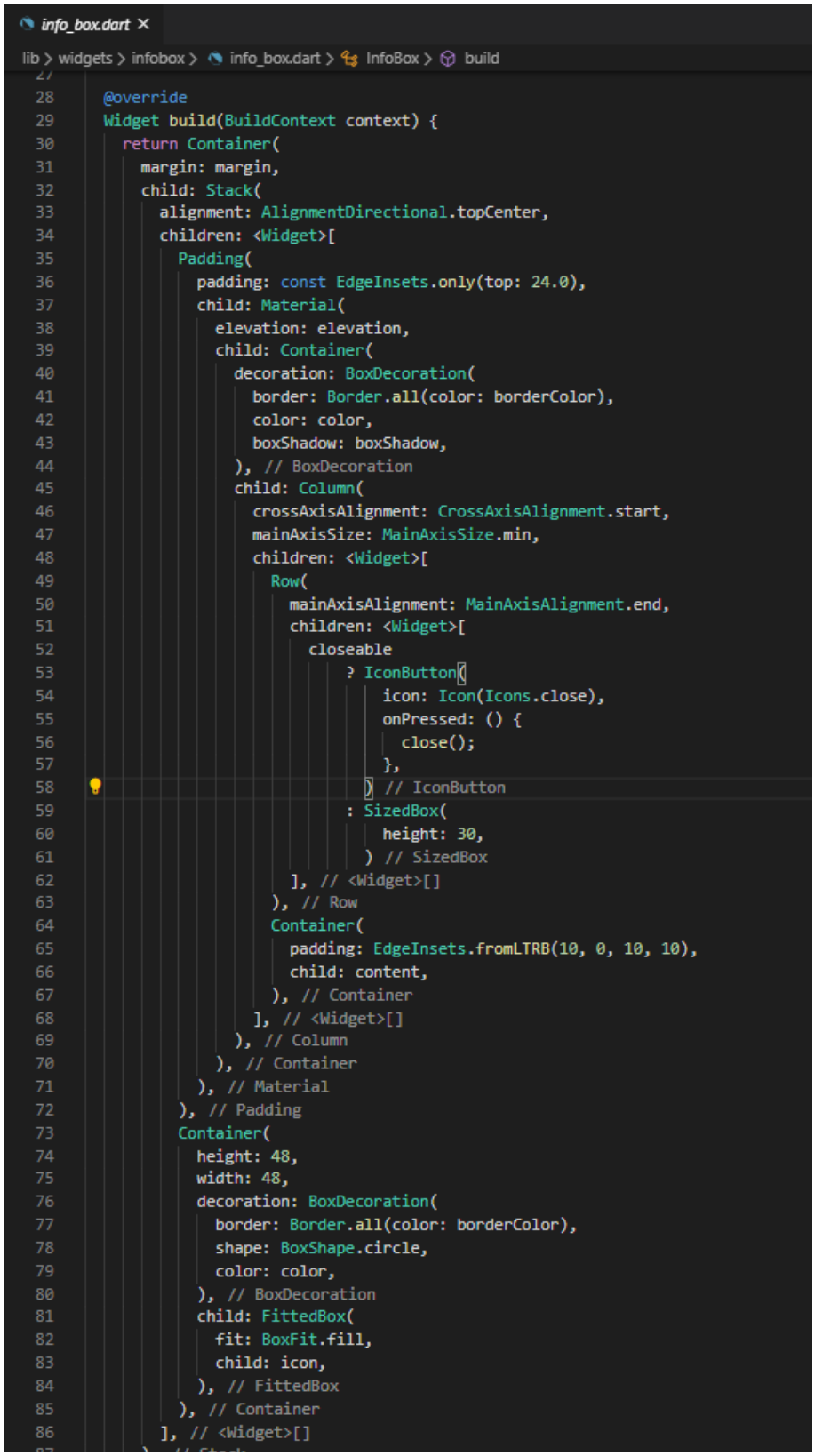
6.1 Infoboks widget

info_box.dart

lib > widgets > infobox > info_box.dart > InfoBox > InfoBox

```dart
import 'package:flutter/material.dart';

class InfoBox extends StatelessWidget {
  final Key key;
  final EdgeInsets margin;
  final double elevation;
  final Color borderColor;
  final Color color;
  final Icon icon;
  final RichText content;
  final bool closeable;
  final Function close;
  final List<BoxShadow> boxShadow;

  InfoBox(
    this.content, {
    this.key,
    this.margin = const EdgeInsets.all(10),
    this.elevation = 0,
    this.borderColor = Colors.black,
    this.color = Colors.white,
    this.boxShadow,
    this.icon = const Icon(Icons.info),
    this.closeable = true,
    this.close,
  });

```

Figur 1: Skjermbilde av koden for infoboks-konstruktøren. Infoboks-konstruktøren tar inn diverse parametere for at widgeten skal kunne brukes i ulike sammenhenger.



Figur 2: Skjermbilde av koden der infoboksen opprettes. Grønn tekst utgjør predefinerte widgets og klasser som kommer med Flutter, og som her settes sammen for å utgjøre infoboksen.

6.2 Kodeeksempler for bruk av API

```
Future<void> signup(String email, String password, Person person) async {
  try {
    final AuthResult result =
        await _firebaseAuth.createUserWithEmailAndPassword(
      email: email,
      password: password,
    );
    FirebaseUser user = result.user;
    _userId = user.uid;
    await _registerUser(person);
    notifyListeners();
  } catch (error) {
    throw error;
  }
}
```

Figur 1: Skjermbilde av koden for signup-metoden i appen

```
Future<void> login(String email, String password) async {
  try {
    AuthResult result = await _firebaseAuth.signInWithEmailAndPassword(
        email: email, password: password);
    FirebaseUser user = result.user;
    _userId = user.uid;
    await _retrieveUser();
  } catch (error) {
    throw error;
  }
}
```

Figur 2: Skjermbilde av koden for logg inn-metoden i appen.

```
void logout() async {
  _user = null;
  _userId = null;
  _firebaseAuth.signOut();
  notifyListeners();
}
```

Figur 3: Skjermbilde av koden for logg ut-metoden i appen.

6.3 SQLite-script

```sql
--- OCCASION ---
DROP TABLE IF EXISTS "occasion";
CREATE TABLE "occasion" (
  "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  "name" TEXT NOT NULL UNIQUE,
  "description" TEXT,
  "url" TEXT
);
--- EXERCISE ---
DROP TABLE IF EXISTS "exercise";
CREATE TABLE "exercise" (
  "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  "name" TEXT NOT NULL UNIQUE,
  "type" TEXT NOT NULL,
  "clothing" TEXT,
  "shoewear" TEXT,
  "equipment" TEXT,
  "conditions" TEXT,
  "execution" TEXT,
  "extra" TEXT
);
--- OCCASION_EXERCISE ---
DROP TABLE IF EXISTS "occasion_exercise";
CREATE TABLE "occasion_exercise" (
  "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  "occasion_id" INTEGER NOT NULL,
  "exercise_id" INTEGER NOT NULL,
  FOREIGN KEY("occasion_id") REFERENCES "occasion"("id") ON DELETE CASCADE,
  FOREIGN KEY("exercise_id") REFERENCES "exercise"("id") ON DELETE CASCADE
);
--- REQUIREMENT ---
DROP TABLE IF EXISTS "requirement";
CREATE TABLE "requirement" (
  "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  "occasion_exercise_id" INTEGER NOT NULL,
  "gender" TEXT NOT NULL,
  "fromAge" INTEGER,
  "toAge" INTEGER,
  "grade" INTEGER NOT NULL,
  "performance" TEXT NOT NULL,
  "measure" TEXT NOT NULL,
  FOREIGN KEY("occasion_exercise_id") REFERENCES "occasion_exercise"("id") ON DELETE CASCADE
);
--- SEARCHABLE_OCCASION ---
DROP TABLE IF EXISTS `searchable_occasion`;
CREATE TABLE `searchable_occasion`(
  "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  "name" TEXT NOT NULL UNIQUE,
  "description" TEXT,
  "general_info" TEXT,
  "other" TEXT
);
--- SEARCHABLE_OCCASION_EXERCISES ---
DROP TABLE IF EXISTS `searchable_occasion_exercises`;
CREATE TABLE `searchable_occasion_exercise`(
  "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  "name" TEXT NOT NULL,
  "requirement" TEXT,
  "unit" TEXT,
  "type" TEXT,
  "note" TEXT,
  "searchable_occasion_id" INTEGER NOT NULL,
  FOREIGN KEY ("searchable_occasion_id") REFERENCES "searchable_occasion"("id") ON DELETE
CASCADE
);
```

Figur 1: Skjermbilde av databasescriptet som brukes for å opprette den lokale SQLite-databasen.

7. Brukerhistorier

Malen alle de påfølgende brukerhistoriene følger: Som "rolle", ønsker jeg "funksjon" for å oppnå "verdi".

- Som 16 år gammel elev ved videregående skole så ønsker jeg å vite hva jeg må ha av fysiske egenskaper for å komme inn i jegertroppen i forsvarets spesialkommando.
- Som 18 år gammel jente ønsker jeg å vite hvordan jeg kan forberede meg fysisk til førstegangstjeneste
- Som tilsatt på 46 år i luftforsvaret er jeg interessert i å vite hva minstekravene er og hvilke øvelser som skal gjennomføres for den årlige testen.
- Som soldat i førstegangstjeneste ønsker jeg å finne minstekravet for å søke meg inn på krigsskole/befalsskole.
- Som videregående elev, ønsker jeg å se kravene for å bli marinejeger, slik at jeg kan forberede meg på testene.
- Som ansatt i forsvaret ønsker jeg å få et treningsprogram, slik at jeg kan stille forberedt til årlig test.
- Som videregående elev, ønsker jeg å vite hvilke muligheter jeg har innen førstegangstjenesten, slik at jeg kan søke meg til noe jeg interesserer meg for.
- Som ny til å trene ønsker jeg å få en oversikt over øvelser slik at jeg vet jeg gjør dem riktig
- Som en som trener ønsker jeg en mulighet til å logge resultater for å se fremgang over tid
- Som bruker av appen ønsker jeg å endre brukernavn for å personalisere opplevelsen min
- Som person i forsvaret ønsker jeg å legge til venner slik at jeg kan kommunisere med andre i forsvaret.

App-tekniske brukerhistorier:

- Som bruker av appen ønsker jeg å CRUD’e(Create, Read, Update, Delete) brukerprofilen min fordi jeg ønsker endringer i profilen min.

8. Brukertesting

8.1 Brukertest 1

Hva skal testes: Brukervennlighet av lo-fi prototype av Krav-fanen

Hvem skal teste: Studenter ved OsloMet, alle aldre. Disse simulerer søkere til førstegangstjeneste / utdanning innen militæret og ansatte i Forsvaret, der ingen av dem kjenner minstekravene som kreves av dem.

Materialer: Prototypen i Figma og dette dokumentet

Roller under brukertesten:

1. Testleder —> Fungerer som både ordstyrer og resepsjonist (den som ønsker velkommen)
2. Observatør —> Skal ikke si noe, kun observere hva som blir gjort og skrive notater
3. "Datamaskin" —> Den som setter opp figma og kontrollerer dette
4. Bruker —> Den som gjennomfører testen

Independent Variables: Prototypen og scenarioene / oppgavene

Dependent Variables: Brukernes opplevelse av produktet

Scenarioer / oppgaver:

1. Du er en mann på 19 som er i førstegangstjenesten. Du vil oppnå minstekrav 8 på 3000-meteren, men er usikker på hvor fort du må løpe. Finn ut hvor fort du må løpe.
2. Du er en ansatt kvinne i Forsvaret på 45 år, og du skal ta din årlige prøve om to uker. Du ønsker å forberede deg, og vil se hvor langt du må kaste medisinballen. Du vet at minstekravet for din stilling er 4. Finn ut hvor langt du må kaste ballen.
3. Du er en 16 år gammel jente på videregående som har gjennomført sesjon del 1 og har fått innkallelse til sesjon del 2. Du ønsker å forberede seg til de

fysiske testene. Du ønsker å finne ut hvilke øvelser som gjennomføres på sesjon og hvilke minstekrav de enkelte øvelsene har.

Spørsmål etter brukertesten:

1. Del dine generelle følelser, var noe lett / vanskelig / irriterende?
2. Hva synes du om teksten på knappene / feltene? Ikke dataen i nedtrekkslistene, men de andre.
3. Har du noen kommentarer til designet, selv om det foreløpig ser veldig røft og skisset ut?
4. Var testen i seg selv ok? Hadde du problemer med å skjønne oppgavene?

8.2 Brukertest 2

Hva testet vi:

Forskjell i tid mellom å søke etter krav på nettsidene versus å søke i appen

Hvordan testet vi:

Oppgavene ble delt ut til en og en etter hvert som de ble ferdige. Testpersonene var adskilt fra hverandre og fikk ikke se hva de andre testpersonene gjorde.

9. Enhetstesting

```
test("get from age", () {
  //stage
  search = new Search(
      selectedOccasion: "",
      selectedAge: "30-39",
      selectedGender: "",
      selectedRequirement: "");
  //do
  var fromAge = search.fromAge;
  //expect
  expect(fromAge, 30);
});
```

Figur 1: Skjermbilde som viser et eksempel på testkode fra appen.

Enhetstesting er delt opp i tre deler: “arrange/stage”, “act/do” og “assert/expect”. Første steg, “stage”, er der alt som er nødvendig for å gjennomføre testen blir satt opp. I figur 1 trenger man en instans av Search for å kunne teste metoden fromAge, og denne blir opprettet i “stage”. Neste steg, “do”, er der metodene som testes blir kjørt. Metoden fromAge skal ta utgangspunkt i variabelen selectedAge som er et intervall i form av en string, og returnere det laveste tallet i intervallet. I “expect” delen sammenlikner man utfallet fra fromAge metoden med det man forventer som resultat, i dette tilfellet skal resultatet bli 30 siden input er 30-39. Vi tester at alle objekter blir opprettet med riktige verdier, verdityper og metodene tilhørende disse klassene. Et utvalg av testene vi gjennomførte følger under.

Testing av Person-modellen

```
void main() {
  test("creating person object", () {
    //stage
    File file = new File("");

    //do
    Person person = new Person(
        id: "id",
        firstName: "firstName",
        lastName: "lastName",
        birthday: DateTime(2017, 9, 7, 17, 30),
        gender: "gender",
        image: file,
        desiredPosition: "desiredPosition"); // Person

    //expect
    expect(person.id, "id");
    expect(person.firstName, "firstName");
    expect(person.lastName, "lastName");
    expect(person.gender, "gender");
    expect(person.birthday, DateTime(2017, 9, 7, 17, 30));
    expect(person.image, file);
    expect(person.desiredPosition, "desiredPosition");
    expect(person.id.runtimeType, String);
    expect(person.birthday.runtimeType, DateTime);
    assert(person.image.runtimeType == new File("").runtimeType);
  });
}
```

Figur 2: Skjermbilde av test-koden for person-modellen

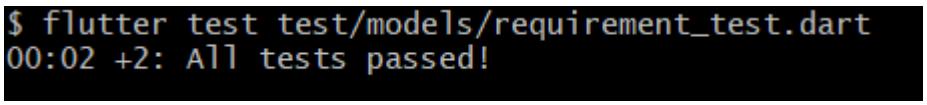
Testing av requirement modellen

```
void main() {
  test("requirements list should increase by one with one value pair in map",
      () {
    //stage
    List<Map<String, dynamic>> requirements = new List<Map<String, dynamic>>();
    Requirement requirement = new Requirement(
        measure: "",
        requirements: requirements,
        exerciseName: "",
        exerciseType: "");

    var map = {
      "key1": "value1",
    };
    //do
    requirement.addRequirement(map);

    //expect
    expect(requirement.requirements.length, 1);
    expect(requirement.requirements[0].keys.elementAt(0), "key1");
    expect(requirement.requirements[0].values.elementAt(0), "value1");
  });

  test("creating requirement object", () {
    //stage
    List<Map<String, dynamic>> requirements = new List<Map<String, dynamic>>();

    //do
    Requirement requirement = new Requirement(
        measure: "measure",
        requirements: requirements,
        exerciseName: "exerciseName",
        exerciseType: "exerciseType");

    //expect
```

Figur 3: Skjermbilde av test-koden for minstekrav-modellen, del 1 / 2

```
    expect(requirement.measure.runtimeType, String);
    expect(requirement.measure, "measure");
    expect(requirement.exerciseName, "exerciseName");
    expect(requirement.exerciseType, "exerciseType");
    assert(requirement.requirements.runtimeType ==
        new List<Map<String, dynamic>>().runtimeType);
  });
}
```

Figur 4: Skjermbilde av test-koden for minstekrav-modellen, del 2 / 2



Figur 5: Skjermbilde av bestått enhetstesting.

10. Systemtesting

Brukerhistorier (Vedlegg 7) / krav (Vedlegg 2)	Implementert	Design	Funksjonalitet	Kommentarer
Opprette profil	Ja	Bestått	Bestått	
Se profil-info	Ja	Bestått	Bestått	
Oppdatere profil	Ja	Bestått	Ikke bestått	Man kan ikke oppdatere brukernavn og passord, men all annen info kan redigeres
Glemt passord	Nei	Ikke bestått	Ikke bestått	
Slette profil	Ja	Bestått	Bestått	
Følger lov om GDPR	Ja	Bestått	Bestått	
Muligheten til å filtrere ut minstekrav for en spesifikk person	Ja	Bestått	Bestått	

Legge til venner for kommunikasjon	Nei	Ikke bestått	Ikke bestått	
Sjekke krav for å bli tatt opp i en anledning	Ja	Bestått	Bestått	
Få et treningsprogram for å nå de fysiske kravene	Ja	Bestått	Bestått	
Sjekke minstekrav for årlig test	Ja	Bestått	Bestått	
Se hvordan øvelser er utført riktig	Ja	Bestått	Bestått	
Se mulige søkbare førstegangstjenester	Ja	Bestått	Bestått	
Mulighet til å forberede seg fysisk til tester	Ja	Bestått	Bestått	
Lagre et treningsresultat for å se fremgang over tid	Nei	Ikke bestått	Ikke bestått	

Lettere å finne informasjon om minstekrav enn på nettsidene	Ja	Ikke relevant	Bestått	Med utgangspunkt i brukertestene i vedlegg 7
Fungerer på forskjellige skjermstørrelser	Ja	Bestått 8 tommer skjerm, bestått 5 tommer skjerm, bestått 3 tommer skjerm	Bestått 8 tommer skjerm, bestått 5 tommer skjerm, bestått 3 tommer skjerm	
Fungerer både på iPhone og Android	Ja	Android bestått iOS bestått	Android bestått iOS, rotering av skjerm er ikke sperret for iPad på grunn av multitasking (Flutter, u.å.)	Har for øvrig kun testet med to enheter for iOS,da det var vanskelig å få tak i flere test-enheter pga Covid-19
Kan legge til egne treningsøvelser	Ja	Må lukke tastaturet for å få tilgang til all informasjon, kan ikke scrolle	Bestått	

Kan registrere nye økter	Ja	Bestått	Bestått	
Rask innhenting av data	Ja	Ikke relevant	Åpne appen: 1 sek Innlogging: 2 sek Forsiden: < 1 sek Øvelser: <1 sek Treningsplan: 2 sek Dagboken: 1 sek	Her er det tilkobling til Firebase som tar noe lengre tid, i tillegg til at det er forbedringspotensiale på innhenting av treningsplanene Testet på en Samsung S10+ og en iPad pro