



Institutt for Informasjonsteknologi

Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo

Besøksadresse: Holbergs plass, Oslo

PROSJEKT NR.

2022 - 15

TILGJENGELIGHET

Åpen

Telefon: 22 45 32 00

## BACHELORPROSJEKT

HOVEDPROSJEKTETS TITTEL	DATO
JUVET	25.05.2022
	ANTALL SIDER / BILAG
	137 sider / 2 bilag
PROSJEKTDELTAKERE	INTERN VEILEDER
MATS NOME SOMMERVOLD, s341829 ADRIAN TOKLE STORSET, s341859 ERIK STORÅS SOMMER, s341870 MATHIAS RUNDGREEN, s341885	NORUN C. SANDERSON

OPPDRAGSGIVER	KONTAKTPERSON
FREMTIND FORSIKRING AS	GLENN A. BROWNLEE

SAMMENDRAG
Fremtind har et behov for å samle ulike kanaler for kompetanseheving i form av tilgjengelige kurs, fagmennesker, arrangement, fagforum, etc. i en intern webløsning. En webbasert kompetanseplattform-applikasjon er spesielt etterspurt av utviklere og designere, men dette er også ønskelig å på sikt kunne implementeres på andre områder hvor det er behov i Fremtind Forsikring.
Vi utviklet en MVP (minimum viable product; enkleste brukbare produkt) etter gjennomføring av én ukes designsprint i starten av januar. Designsprinten ble ledet og fasilitert av interaksjonsdesigner i Fremtind. Vi videreutviklet deretter denne løsningen gjennom den perioden vi hadde til rådighet. Målet var at løsningen skulle kunne settes i produksjon og benyttes i det daglige i bedriften. Teamet som veiledet oss var bevisst på begrensningene vi hadde i tid og ressurser, og ønsket å legge til rette for at vi kunne levere et produkt/en oppgave som passet innen disse rammene. Løsningen fikk navnet «Juvet».

3 STIKKORD	Webapplikasjon
	Ressursplattform
	Kompetanseheving

## Forord

Rapporten er en beskrivelse og forklaring av bacheloroppgaven vi har skrevet for Fremtind Forsikring. Kontakten med Fremtind og avtalen om å utføre bachelorprosjektet med de ble inngått gjennom et gruppemedlem, som jobber i Fremtind Forsikring i tillegg til studiene. Hensikten med prosjektet var å utarbeide en løsning som tok i bruk mange deler av det pensumet vi har gjennomgått i løpet av snart tre studieår. Dette i tillegg til å utvikle noe som gir verdi til bedriften, basert på behov og krav som er gitt av oppdragsgiver.

Vi vil takke Fremtind Forsikring for å ha gitt oss denne muligheten og for at vi har blitt tatt imot og ivaretatt på en eksemplarisk måte. Spesielt Glenn A. Brownlee (leder for Fremtind Digital), som har vært vår kontaktperson. Samtidig vil vi takke William Killerud (utvikler i Fremtind), og Øyvind Nordbø (interaksjonsdesigner i Fremtind) for læring og innsikt tidlig i prosjektet og Tore Haugland (teknisk arkitekt i Fremtind) for hjelpen med oppsett og tilkobling mot Fremtinds systemer. Disse fire, i tillegg til flere i organisasjonen har stilt opp og gitt oss støtte underveis i prosjektet.

Vi vil også takke vår veileder ved OsloMet, Norun C. Sanderson, for konstruktive tilbakemeldinger og støtte i forbindelse med utformingen av denne prosjektrapporten. Dette har vært til stor hjelp ettersom vi ikke har betydelig erfaring med å skrive denne typen dokumentasjon fra tidligere i studiet.

Denne rapporten er optimalisert for digital lesning i form av PDF, med klikkbarer lenker for henvisninger til kilder, figurer, vedlegg og eksterne ressurser.

Ettersom vi var 4 personer og hadde fått gjort et stort arbeid i prosjektgruppa følte vi det var naturlig at rapporten gjenspeiler antallet personer og arbeidsmengden i prosjektet.

# Innholdsfortegnelse

FIGURLISTE.....	4
TABELLISTE.....	6
1. INNLEDNING .....	7
1.1 Prosjektgruppa.....	8
1.2 Veiledere .....	8
1.3 Oppdragsgiver.....	9
2. PROSESSDOKUMENTASJON .....	10
2.1 Planlegging .....	10
2.1.1 Ved oppstart.....	10
2.1.2 Under utviklingen.....	11
2.2 Verktøy brukt i prosessen.....	11
2.3 Rammeverk/bibliotek brukt .....	15
2.4 Arbeidsmetode .....	17
2.4.1 Designsprint.....	17
2.4.2 Smidig utvikling .....	18
2.4.3 Tilbakemeldinger.....	18
3. UTVIKLINGSPROSESSEN .....	19
3.1 Oppstartsfasen.....	19
3.1.1 Gjennomføring av designsprint.....	19
3.1.2 Møter og kommunikasjon .....	22
3.1.3 Utfordringer og diskusjoner rundt grunnleggende oppsett .....	23
3.1.4 Demonstrasjon av prosjektet til Fremtind-ansatte .....	23
3.1.5 Starten av utviklingen .....	23
3.2 Utviklingsfasen.....	24
3.2.1 Prototyping .....	24
3.2.2 Utvikling MVP .....	26
3.2.2.1 Docker-container-hosting .....	26
3.2.2.2 AWS Fargate-hosting .....	27
3.2.2.3 AWS MySQL database .....	28
3.2.2.4 Single Sign On med AzureAD .....	28
3.2.2.5 Utvikling av frontend med React og Jøkul .....	28
3.2.3 Iterasjon 1, 2 og 3.....	29
3.2.4 Ferdigstillelse og finpass av løsningen .....	29
3.2.5 Leveranse til oppdragsgiver .....	29
3.2.6 Dokumentering .....	30
3.2.7 Samsvar mellom prosjektplan og reell gjennomførelse .....	30
3.2.8 Kvalitetssikring .....	31
3.3 Store utfordringer og valg .....	31
4. TESTING.....	32
4.1 Testing under designsprint .....	33
4.2 Testing underveis i utviklingen .....	34
4.3 Akseptansetest.....	36
4.4 Tilgjengelighetstest .....	38
4.5 Samsvar mellom reell testdekning og ideell testdekning .....	38
5. PRODUKTDOKUMENTASJON .....	38
5.1 Kravspesifikasjon.....	38
5.2 Beskrivelse av løsning .....	41
5.2.1 Arkitektur .....	41
5.2.2 Frontend .....	42
5.2.2.1 Arkitektur .....	42
5.2.2.2 Applikasjonstilstand.....	45
5.2.3 Backend.....	46
5.2.3.1 Domain .....	48

5.2.3.2 Controllers.....	50
5.2.3.4 Services.....	51
5.2.3.5 Repositories.....	51
5.2.4 Enkeltpålogging .....	52
5.2.5 Design og tilgjengelighet .....	53
<b>5.3 Samsvar mellom kravspesifikasjon og produkt .....</b>	<b>56</b>
<b>5.4 Sentrale datastrukturer i løsningen .....</b>	<b>57</b>
5.4.1 Prinsipper for utvikling av kode .....	57
5.4.2 Frontend-modeller .....	57
5.4.3 Backend-modeller .....	59
5.4.4 Database – Entitet Relasjons diagram.....	60
<b>5.5 AWS-hosting og -deployering.....</b>	<b>61</b>
<b>5.6 Sikkerhet.....</b>	<b>64</b>
<b>5.7 Forhold til maskiner/database/OS .....</b>	<b>65</b>
5.7.1 API dokumentasjon .....	65
5.7.2 Databaseintegrasjon .....	67
<b>5.8 Hoveddeler av programmet .....</b>	<b>67</b>
<b>6. BRUKERVEILEDNING .....</b>	<b>69</b>
6.1 Landingssiden.....	70
6.2 Søkeresultat .....	73
6.3 Aktivitet detaljert .....	74
6.4 Min side.....	79
6.5 Feilmeldinger.....	84
6.5.1 Funksjoner som ikke er implementert .....	84
6.5.2 Nettverksproblemer .....	86
<b>7. KONKLUSJON OG DRØFTING .....</b>	<b>88</b>
7.1 Utbytte .....	88
7.2 Hva er produktets bruks- og nytteverdi? .....	88
7.3 Hva ville vi ha gjort annerledes? .....	89
7.4 Tilbakemelding fra oppdragsgiver .....	89
7.5 Status for videre utvikling og produksjonssetting av produktet.....	89
7.6 Oppsummering og konklusjon.....	90
<b>REFERANSELISTE.....</b>	<b>91</b>
<b>VEDLEGG .....</b>	<b>98</b>
<b>Vedlegg 1 - Designsprint .....</b>	<b>98</b>
Dag 1.....	98
Dag 2.....	102
Dag 3.....	109
Dag 4.....	114
<b>Vedlegg 2 – Tabell med spørsmål for brukertest under designsprint .....</b>	<b>117</b>
<b>Vedlegg 3 – Svarark brukertest av prototype under designsprint.....</b>	<b>118</b>
<b>Vedlegg 4 – GitHub projects prosjektbrett.....</b>	<b>119</b>
<b>Vedlegg 5 – Resultat WCAG-test via Siteimprove.com .....</b>	<b>127</b>
<b>Vedlegg 6 – Hovedklasser og modeller i backend .....</b>	<b>128</b>
<b>Vedlegg 7 – Mappestruktur backend .....</b>	<b>132</b>
<b>Vedlegg 8 – Mappestruktur frontend .....</b>	<b>133</b>
<b>Vedlegg 9 – Postman-testing og -respons eksempel .....</b>	<b>134</b>
<b>Vedlegg 10 – Attest fra oppdragsgiver .....</b>	<b>137</b>

## Figurliste

FIGUR 1 - GANTT-DIAGRAM FOR PLANLAGT PROGRESJON I PROSJEKTET .....	10
FIGUR 2 - GITHUB PROJECTS PROSJEKTBRETT, "KANBAN", FOR JUDET .....	12
FIGUR 3 - OPPSUMMERING AV DESIGNSPRINT, "HVA HAR DERE GJORT?" .....	22
FIGUR 4 - OPPSUMMERING AV DESIGNSPRINT, "INNSIKTSYNTSE" .....	22
FIGUR 5 - PROTOTYPEN AV LANDINGSSIDEN FOR JUDET FRA DESIGNSPRINT .....	25
FIGUR 6 - DOCKER-OPPSETT I ROTMAPPE (FRA ./JUDET/DOCKER-COMPOSE.YML) .....	27
FIGUR 7 - SVARTABELL FOR SPØRSMÅL I BRUKERTEST AV PROTOTYPE .....	33
FIGUR 8 - EKSEMPLER PÅ TILBAKEMELDINGER UNDER BRUKERTEST. HENTET FRA GOOGLE SHEETS-DOKUMENTET HVOR ALL INNSIKT FRA BRUKERTESTER ER LAGRET .....	34
FIGUR 9 - KODEEKSEMPLER PÅ EN JUNIT5 TEST FRA JUETS PROSJEKTKODE .....	35
FIGUR 10 - MOCKMVC-TESTING I JUETS PROSJEKTKODE .....	35
FIGUR 11 - EKSEMPLER PÅ TESTING AV ET ENDEPUNKT I JUDET MED POSTMAN .....	36
FIGUR 12 - AKSEPTANSETEST SVARPROSENT PÅ TESTSPØRSMÅLENE .....	37
FIGUR 13 - TESTDEKNING AV BACKEND I JUETS PROSJEKTKODE. TOTALVERDIER ER RAMMET INN AV GUL INNRAMMING I FIGUREN ..	38
FIGUR 14 - RESSURSKJEDEN I ET HEADLESS BACKEND OPPSETT - SLIK JUDET ER UTVIKLET .....	41
FIGUR 15 - RESSURSKJEDEN I ET TRADISJONELT BACKEND OPPSETT (IKKE HEADLESS) .....	42
FIGUR 16 - MAPPESTRUKTUR I KILDEKODEN FOR JUDET .....	42
FIGUR 17 - MAPPESTRUKTUR I ASSETS LAGET FOR KILDEKODEN I JUETS FRONTEND .....	43
FIGUR 18 - MAPPESTRUKTUR I BIBLIOTEK LAGET FOR JUETS FRONTEND .....	43
FIGUR 19 - MAPPESTRUKTUR I ROUTES-LAGET FRA JUETS FRONTEND .....	44
FIGUR 20 - DIAGRAM OVER FRONTEND APPLIKASJONENS AVHENGIGHETER INTERNT OG EKSTERNT I JUDET .....	44
FIGUR 21 - DIAGRAM SOM FORKLARER TILSTANDSKJEDEN I APPLIKASJONEN .....	46
FIGUR 22 - LAGINDELINGEN AV JUETS BACKEND .....	47
FIGUR 23 - STRUKTUR PÅ BACKEND FOR MODELLER I «DOMAIN» — FRA JUETS PROSJEKTKODE .....	48
FIGUR 24 - ENTITETSMODELLEN FOR Klassen TAG PÅ BACKEND SOM VISER RELASJONENE DEN HAR MED ENTITETENE USER OG ACTIVITY — FRA JUETS PROSJEKTKODE .....	48
FIGUR 25 — METODE PÅ BACKEND SOM KONVERTERER (CAST) USER-ENTITET TIL USERCARD RESPONS VED BRUK AV «BUILDER PATTERN» — FRA JUETS PROSJEKTKODE .....	49
FIGUR 26 - MODELLKLASSEN SOM DEFINERER HVILKE DATA SOM SKAL SENDES MED AKTIVITETLISTEN PÅ BACKEND — FRA JUETS PROSJEKTKODE .....	49
FIGUR 27 - MAPPESTRUKTUR SOM VISER KONTROLLERKLASSENE SOM DEFINERER API'ET — FRA JUETS PROSJEKTKODE .....	50
FIGUR 28 - ENDEPUNKT FOR Å SØKE PÅ AKTIVITETER — FRA JUETS PROSJEKTKODE .....	50
FIGUR 29 - OVERSIKT OVER SERVICE MAPPER — FRA JUETS PROSJEKTKODE .....	51
FIGUR 30 - DIAGRAM SOM VISER HVORDAN EN SERVICE ER AVHENGIG AV FLERE SERVICES FOR Å SEPARERE ARBEIDSOPPGAVER I JUDET BACKEND .....	51
FIGUR 31 - REPOSITORY FOR Å HENTE UT AKTIVITETER HVOR AVANSERTE SPØRRLINGER ER FORENKLET MED METODENAVN GITT AV JPA — FRA JUETS PROSJEKTKODE .....	52
FIGUR 32 - VISER NETTVERKSFORESPØRSLENE UNDER EN INNLLOGGING PÅ JUDET .....	53
FIGUR 33 - SKJERMbilder FRA JUDET: EKSEMPLER PÅ RESPONSIV NETTSIDE .....	54
FIGUR 34 - "ACTIVITY" TYPESCRIPT-INTERFACE DEFINERT — FRA JUETS PROSJEKTKODE .....	58
FIGUR 35 - UTKLIPP FRA FREMTINDS DESIGNSYSTEM (JOKUL.FREMTIND.NO/KOMPONENTER/BUTTONS). EKSEMPLER PÅ FERDIG KOMPONENT FOR PRIMÆRKNAPP .....	59
FIGUR 36 - MODELLENE ILLUSTRERT I FILSTRUKTUREN FOR BACKEND .....	60
FIGUR 37 - ATTRIBUTTENE I AKTIVITETSKLASSSEN .....	60
FIGUR 38 — ER-DIAGRAM AV VÅR DATABASE SOM VISER RELASJONER MELLOM TABELLER .....	61
FIGUR 39 - ILLUSTRASJON AV AWS-OPPSETT FOR JUDET .....	62
FIGUR 40 - AWS LOAD BALANCER MED BETINGET RUTING, SLIK DET BLE SATT OPP FOR JUDET .....	64
FIGUR 41 - OVERSIKT OVER JUDET S REST API .....	66
FIGUR 42 - AWS MySQL DATABASE KONFIGURASJONSFIL FRA PROSJEKTKODEN I JUDET .....	67
FIGUR 43 — SKJERMbilde FRA JUDET PÅLOGGING: PÅLOGGINGSPORTAL FRA MICROSOFT AZUREAD FOR Å KOMME INN PÅ "JUDET.FREMTIND.NO" .....	69

FIGUR 44 – SKJERMBILDE FRA JUVENT: LANDINGSSIDEN.....	70
FIGUR 45 - SKJERMBILDE FRA JUVENT: KONTEKSTBASERT PRESENTASJON AV AKTIVITETER .....	71
FIGUR 46 - SKJERMBILDE FRA JUVENT: SØKE-LINJEN.....	72
FIGUR 47 – SKJERMBILDE FRA JUVENT: SØKERESULTAT-SIDEN .....	73
FIGUR 48 – SKJERMBILDE FRA JUVENT: AKTIVITET DETALJERT-SIDEN.....	74
FIGUR 49 - SKJERMBILDE FRA JUVENT: AKTIVITET DETALJERT-SIDEN VED REGISTRERT PÅMELDING.....	75
FIGUR 50 - SKJERMBILDE FRA JUVENT: AKTIVITET DETALJERT-SIDEN VED REGISTRERT AVMELDING.....	76
FIGUR 51 - SKJERMBILDE FRA JUVENT: AKTIVITET DETALJERT-SIDEN VED VELLYKKET LAGRING TIL SENERE .....	77
FIGUR 52 - SKJERMBILDE FRA JUVENT: AKTIVITETSDETALJER FOR AKTIVITET.....	78
FIGUR 53 – SKJERMBILDE FRA JUVENT: MIN SIDE .....	79
FIGUR 54 – SKJERMBILDE FRA JUVENT: OPPRETT AKTIVITET-SIDE, VALG AV AKTIVITETSFOMAT .....	80
FIGUR 55 - SKJERMBILDE FRA JUVENT: OPPRETT AKTIVITET-SIDE, VALG MELLOM FYSISK, DIGITAL ELLER HYBRID AKTIVITET .....	81
FIGUR 56 - SKJERMBILDE FRA JUVENT: EKSEMPLER PÅ UTFYLT OPPRETT-AKTIVITET-SKJEMA.....	82
FIGUR 57 - SKJERMBILDE FRA JUVENT: UTFYLNING AV DETALJER FOR HYBRID AKTIVITET.....	82
FIGUR 58 - SKJERMBILDE FRA JUVENT: BEKREFTELSE PÅ SUKSESSFULL OPPRETTELSE AV NY AKTIVITET .....	83
FIGUR 59 - SKJERMBILDE FRA JUVENT: FEILMELDING VED Å SE FLERE PERSONER MED KOMPETANSE .....	84
FIGUR 60 - SKJERMBILDE FRA JUVENT: FEILMELDING VED Å TRYKKE PÅ EMNE-TAG.....	85
FIGUR 61 - SKJERMBILDE FRA JUVENT: LANDINGSSIDEN VED NETTVERKSPROBLEMER.....	86
FIGUR 62 - SKJERMBILDE FRA JUVENT: FEILMELDING PÅ SØKERESULTAT-SIDEN DERSOM BACKEND-TJENER IKKE SVARER .....	86
FIGUR 63 - SKJERMBILDE FRA JUVENT: OPPRETT AKTIVITET VED NETTVERKSOPPLØSNINGER .....	87
FIGUR 64 - DESIGNSPRINT EKSPERTINTERVJU - "HVORDAN KAN VI"-SPØRMÅL MED VARMEKART .....	98
FIGUR 65 - DESIGNSPRINT - DEFINERTING AV MÅL OG LEDESTJERNE FOR DESIGNSPRINT-PROSESSEN.....	99
FIGUR 66 - DESIGNSPRINT: SPRINTSPØRSMÅL OPPSUMMERING .....	100
FIGUR 67 - DESIGNSPRINT: TJENESTEKART .....	100
FIGUR 68 - DESIGNSPRINT: OPPSUMMERT VISJONÆRT MÅL, SPRINTSPØRSMÅL OG VIKTIGSTE DEL AV BRUKERREISE .....	101
FIGUR 69 - KONSEPT 1 AV KONSEPTSKISSER OG VARMEKART, "GOOD GAME, WELL PLAYED!" .....	102
FIGUR 70 - KONSEPT 1 AV KONSEPTSKISSER OG VARMEKART, "FINN EN VENN!" .....	103
FIGUR 71 - KONSEPT 1 AV KONSEPTSKISSER OG VARMEKART, "PAC RATS". .....	104
FIGUR 72 - KONSEPT 1 AV KONSEPTSKISSER OG VARMEKART, "INTO THE LIGHT" .....	105
FIGUR 73 - KONSEPT 1 AV KONSEPTSKISSER OG VARMEKART, "JUVENT REPLAY" .....	106
FIGUR 74 - KONSEPT 1 AV KONSEPTSKISSER OG VARMEKART, "NAN". .....	107
FIGUR 75 - DESIGNSPRINT: STORYBOARD .....	108
FIGUR 76 - PROTOTYPE AV LANDINGSSIDE UNDER DESIGNSPRINT .....	109
FIGUR 77 - PROTOTYPE AV LANDINGSSIDE MED OPPDATERT SØKEFELT UNDER DESIGNSPRINT .....	110
FIGUR 78 - PROTOTYPE AV SØKERESULTATSSIDE UNDER DESIGNSPRINT .....	111
FIGUR 79 - PROTOTYPE AV LANDINGSSIDE, AKTIVITET LAGT TIL I "LAGRET TIL SENERE" UNDER DESIGNSPRINT .....	112
FIGUR 80 - PROTOTYPE AV "INNBOKS" UNDER DESIGNSPRINT .....	113
FIGUR 81 - DESIGNSPRINT OPPSUMMERING - HVA HAR VI GJORT? .....	114
FIGUR 82 - DESIGNSPRINT OPPSUMMERING – INNSIKTSNTES .....	114
FIGUR 83 - DESIGNSPRINT OPPSUMMERT - SPRINTSPØRSMÅLENE .....	114
FIGUR 84 - DESIGNSPRINT OPPSUMMERT - VALIDERINGSGRAD OG INNSIKTER.....	115
FIGUR 85 - DESIGNSPRINT OPPSUMMERT - SITATER FRA BRUKERTESTER .....	115
FIGUR 86 - DESIGNSPRINTEN OPPSUMMERT - FOKUSPUNKTER FOR VIDERE UTVIKLING.....	115
FIGUR 87 - DESIGNSPRINT OPPSUMMERT - TIPS TIL VIDERE STEG I UTVIKLINGEN.....	116
FIGUR 88 - DESIGNSPRINT BRUKERTEST SVARARK .....	118
FIGUR 89 - GitHub PROSJEKTBRETT, "KANBAN" .....	119
FIGUR 90 - GitHub PROSJEKTBRETT, "SPRINT". .....	120
FIGUR 91 - GitHub PROSJEKTBRETT, "MINE OPPGAVER". .....	121
FIGUR 92 - GitHub PROSJEKTBRETT, "ADMINISTRATIVT" .....	122
FIGUR 93 - GitHub PROSJEKTBRETT, "ITERASJONER" .....	123
FIGUR 94 - GitHub PROSJEKTBRETT, PERSONLIGE OPPGAVER .....	124
FIGUR 95 - GitHub PROSJEKTBRETT, "OPPGAYER" .....	125
FIGUR 96 - GitHub PROSJEKTBRETT, "IKKE FERDIG FORRIGE SPRINT" .....	126

FIGUR 97 - RESULTAT AV WCAG-TEST AV JUVETS LANDINGSSIDE .....	127
FIGUR 98 - ACTIVITY-KLASSEN I JUVETS BACKEND .....	128
FIGUR 99 - THUMBNAIL-KLASSEN I JUVETS BACKEND .....	128
FIGUR 100 - LOCATION-KLASSEN I BACKEND .....	128
FIGUR 101 - ADDRESS-KLASSEN I JUVETS BACKEND .....	129
FIGUR 102 - ATTACHEMENT-KLASSEN I JUVETS BACKEND .....	129
FIGUR 103 - TAG-KLASSEN I JUVETS BACKEND .....	129
FIGUR 104 - USER-KLASSEN I JUVETS BACKEND .....	130
FIGUR 105 – «ACTIVITY REPOSITORY»-GRENSESNITT I JUVETS BACKEND .....	131
FIGUR 106 - MAPPESTRUKTUREN I BACKEND-DELEN AV PROSJEKTET.....	132
FIGUR 107 - MAPPESTRUKTUREN I FRONTEND-DELEN AV PROSJEKTET.....	133
FIGUR 108 - EKSEMPEL PÅ TESTING AV ET ENDEPUNKT I JUDET MED POSTMAN.....	134
FIGUR 109 - RESPOND MED FEILMELDING OG 500-STATUS ETTER SPØRRING MOT ET API-ENDEPUNKT .....	136
FIGUR 110 - ATTEST FRA OPPDRAGSGIVER FREMTIND.....	137

## Tabelliste

TABELL 1 - FUNKSJONELLE KRAV TIL MVP .....	39
TABELL 2 - IKKE-FUNKSJONELLE KRAV TIL MVP .....	39
TABELL 3 - BESKRIVELSE AV LAGINNDELINGEN FOR FRONTEND I JUDET .....	43
TABELL 4 - SAMSVAR MELLOM KRAVSPESIFIKASJON OG ENDELIG PRODUKT, FUNKSJONELLE KRAV .....	56
TABELL 5 - SAMSVAR MELLOM KRAVSPESIFIKASJON OG ENDELIG PRODUKT, IKKE-FUNKSJONELLE KRAV .....	56
TABELL 6 - OVERISK OG KRAV FOR DE ULIKE TYPE AKTIVITETENE I JUDET .....	83

## 1. Innledning

Eksisterende løsning for kompetansehevende aktiviteter i Fremtind Forsikring er kompleks og lite effektiv. I dag ligger kurs, fagforum, påmelding til kurs, info om arrangementer og oversikt over fagpersoner spredt i ulike kanaler som Confluence, Teams, Yammer, epost, osv. Ansatte opplever det vanskelig å holde oversikt, og utfordrende å søke opp info eller fagstoff som er gitt. Aktiviteter for kompetanseheving foregår både internt og eksternt i organisasjonen, og det er ikke noen samlet plass disse blir annonsert.

Med vår løsning kan også ledere i Fremtind gi en tydeligere forventning for påmeldingsprosessen til de ansatte, som kan bidra til å senke terskelen for å delta på både interne og eksterne kompetansehevende aktiviteter.

Samtidig er det en policy internt i organisasjonen at det ikke er lov å sende invitasjoner via epost til hele bedriften, slik at de ansatte er nødt til å følge med på aktuelle hendelser i relevante forum og kanaler for å holde seg oppdatert på hva som skjer.

Løsningen skal først og fremst være en løsning for Fremtind Digital, men med mulighet til å kunne løse tilsvarende problem hos andre deler av organisasjonen.

«Juvet» ble navnet på løsningen. Det var oppdragsgiver som kom med dette navneforslaget, og vi valgte å beholde det. Et *juv* betyr en bratt kløftlignende dal og var tenkt at ansatte kan gjøre et dypdykk i tilgjengelig relevant kunnskap (Bolstad, 2020). Navnet passer også godt inn blant Fremtinds andre interne systemer.

## 1.1 Prosjektgruppa

Vi er fire dataingeniør-studenter ved OsloMet. Alle på gruppa har erfaring med å arbeide sammen i tidligere prosjektoppgaver under studiene og vi er derfor godt kjent med hverandres styrker og svakheter. Gruppa utfyller hverandre godt og har til sammen et bredt spekter av kompetanse og erfaring. På bakgrunn av dette var valget enkelt om å gå sammen og skrive denne bacheloroppgaven.

**Adrian Tokle Storset**

*Bachelorprogram:*  
Dataingeniør



Hovedansvar for prosjektdagbok, prototyping og prosjektrapport

**Erik Storås Sommer**

*Bachelorprogram:*  
Dataingeniør



Hovedansvar for backend, database, sikkerhet og testing

**Mats Nome Sommervold**

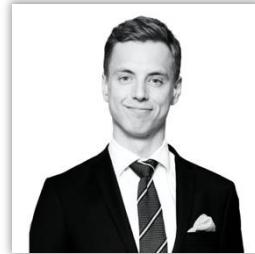
*Bachelorprogram:*  
Dataingeniør



Hovedansvar for arkitektur, deployering, kanbanbrett og frontend

**Mathias Rundgreen**

*Bachelorprogram:*  
Dataingeniør



Hovedansvar for koordinering, prototyping og prosjektrapport

## 1.2 Veiledere

Fremtind:

**Glenn A. Brownlee**

*Utviklingsleder,  
Distribuerte løsninger,  
Fremtind Digital*

Vi kom i kontakt med Glenn gjennom at Mathias (gruppemedlem) jobber hos oppdragsgiver og undersøkte mulighetene for å skrive bachelor hos Fremtind. Glenn har vært en stor bidragsyter til Fremtinds prisvinnende designsystem og har erfaring som frontend arkitekt og -utvikler. Glenn og hans team i Fremtind Digital har vært til stor hjelp under hele prosjektet.

OsloMet:

**Norun Christine Sanderson**

*Førsteamanuensis,  
Institutt for informasjonsteknologi,  
OsloMet*

Etter at instituttet gjorde vurdering av prosjektskissen ble Norun C. Sanderson tildelt som vår interne veileder ved OsloMet.

Vi kontaktet veileder hos OsloMet tidlig i prosjektet og avtalte å holde ukentlige møter.

## 1.3 Oppdragsgiver

**Fremtind**

**Fremtind Forsikring AS**

*Hammersborggata 2,  
PB 778 Sentrum, 0106 Oslo  
Org. nummer: 915 651 232*

Fremtind Forsikring er et relativt ungt forsikringsselskap, opprettet i 2019. Det er eid av Sparebank1 Gruppen og DnB. Fremtind ønsker å være «morgendagens forsikringsselskap» og satser deriblant mye på digital kompetanse. Dette for å gi kunder og ansatte brukervennlige digitale tjenester og løsninger.

## 2. Prosessdokumentasjon

I prosessdokumentasjonen vil vi presentere hvordan vi planla prosjektet og prosessen for hvordan vi kom frem til det endelige produktet.

### 2.1 Planlegging

#### 2.1.1 Ved oppstart

Ved oppstart og videre gjennom den første uken var en designsprint planlagt av Fremtinds interaksjonsdesigner.

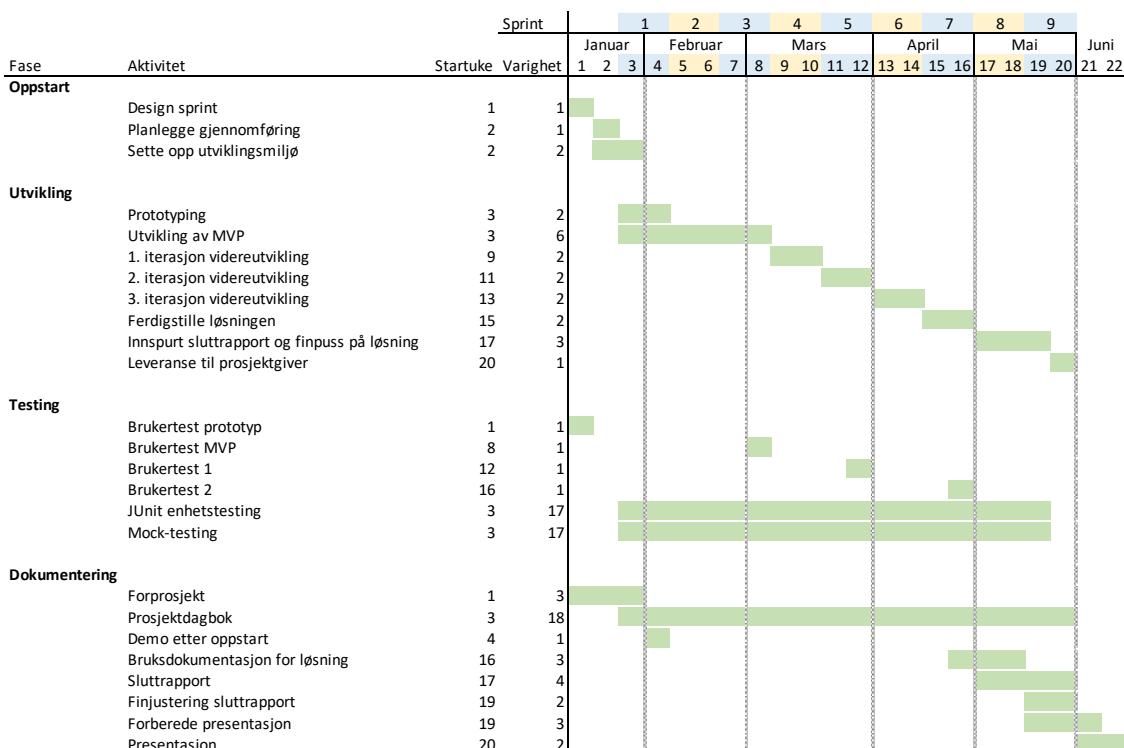
For å få en felles forståelse for hva som var «godt nok» innen funksjonalitet og utseende for leveransen av denne oppgaven, var det viktig å definere hvilke krav oppdragsgiver hadde.

Kravspesifikasjonen sees i [kapittel 5.1](#).

På grunn av pandemi-situasjonen var det i starten av prosjektet pålegg om hjemmekontor, uten forutsigbarhet for når vi hadde mulighet til å møtes fysisk. Dette hadde konsekvenser for hvor tett på vi kunne være med Fremtind, men innad i gruppa utgjorde ikke dette noe hinder for å møtes fysisk.

Utover det å skulle samarbeide om dette prosjektet hadde alle gruppemedlemmene andre forpliktelser i løpet av en vanlig uke. Dette begrenset tidsrommet gjennom uken hvor alle kunne være tilgjengelige samtidig. Følgende var det derfor tre dager per uke som ble definert som primær arbeidstid til prosjektet. Det var tirsdager, onsdager og fredager.

Følgende Gantt-diagram viser de ulike fasene, hva som inngår av aktiviteter og periode for gjennomførelse. Utarbeidelsen av dette diagrammet ble gjort uten særlig erfaring innen prosjektplanlegging hvor vi også var avhengige av en oppdragsgiver for progresjon.



Figur 1 - Gantt-diagram for planlagt progresjon i prosjektet.

### *2.1.2 Under utviklingen*

Vi forsøkte så godt som mulig å holde oss til sprintene, vist ovenfor i figur 1 til å bestå av perioder på 2 uker per sprint, og fokusområdene for disse. Vi innså derfor raskt at det ville bli vanskelig å holde planen slik den var oppsatt i Gantt-diagrammet. Derfor ble det avgjort at de laveste prioriterte kravene i kravspesifikasjonen ville bli prioritert bort – se [kapittel 3.2.7](#). Konsekvensene denne avgjørelsen resulterte i var ikke store fordi oppdragsgiver også var klar over at dette var funksjonalitet som tar tid å utvikle. Særlig når det kom som et tillegg til å utvikle det grunnleggende konseptet, samt utviklingen av systemet og testingen av begge disse.

Planleggingen underveis gikk i stor del fra sprint til sprint. Dette betyddet at vi kun planla i detaljer hva som måtte gjøres for den neste perioden på to uker fremover i tid. Oppgavene ble lagt inn i et kanban-prosjektbrett med de tingene vi tidlig så måtte utføres. De ble lagt under de ulike sprintene i prosjektlopet som virket hensiktsmessig. Oppgaver vi mente vi kunne ta for oss der og da ble fordelt blant gruppemedlemmene, og satt til å bli fullført i løpet av den inneværende sprinten. Kanban-prosjektbrett blir nærmere forklart i [delkapittel 2.2](#).

Det ble et par ganger foreslått at vi burde sette opp en ny overordnet prosjektplan, muligens i form av et nytt Gantt-diagram. Dette ble for hver gang nedprioritert ettersom vi underveis innså at store deler av denne eventuelle nye planen ville kreve å bli revidert gjentatte ganger. Vi innså også at disse revisjonene av det nye Gantt-diagrammet mest sannsynlig ville bestå av en speilet versjon av Kanban-prosjektbrettet på det gitte tidspunktet. Derfor ble det bestemt at Kanban-brettet ville være godt nok for å gjøre jobben prosjektplanleggingen innebar på egen hånd.

## *2.2 Verktøy brukt i prosessen*

Vi vil i dette delkapittelet beskrive de verktøyene som ble benyttet under prosjektet. Det blir her i hovedsak beskrevet generelt, og de delene vi anser som viktige vil bli beskrevet mer detaljert senere i rapporten.

### **GitHub projects**

GitHub er en plattform som legger opp til deling og samskriving av kode basert på Git versjonskontrollen (GitHub, u.d.). Vårt prosjektbrett i GitHub (Kanban board) var tilknyttet repository'et hvor all koden ligger. Et prosjektbrett er en oversikt over alle små og store oppgaver som prosjektet består av. Oppgavene er representert av et kort på dette brettet, som kan tillegges prioritet, frist, hvem som er ansvarlig for oppgavens utførelse og hvilken periode (iterasjon) oppgaven skal utføres i. Et repository er en definert plass hvor all kode for prosjektet er samlet (Beal, 2022). I repository ligger også all metadata og referanser til eksterne avhengigheter for prosjektkoden. Valget av GitHub var enkelt da dette er noe som oppdragsgiver bruker og ville gi de tilgang til vårt arbeid underveis. På denne måten ble tilbakemeldinger fra Fremtind enklere. I tillegg er dette et verktøy vi i prosjektgruppa var godt kjent med og derfor trengte lite opplæring i.

Prosjektbrettet i GitHub ble satt opp med ulike typer brett:

- «Kanban»-brett som viste alle aktiviteter/oppgaver uansett hvilken sprint/iterasjon som ble vist.
- «Sprint»-brett som viste alle oppgaver som var tilknyttet inneværende sprint.

- «Mine oppgaver»-brett som viste alle aktiviteter/oppgaver for hvert enkelt gruppemedlem, hvilken sprint denne oppgaven var tilknyttet, hva dens status var, dens karakter, evt. frist for gjennomførelse og kompleksitet.
- «Administrativt»-brett som viste kun de administrative oppgavene, som ikke direkte hadde noe med utvikling i kode å gjøre. Eksempelvis ble alle oppgaver for dokumentasjon vist her.
- «Iterasjoner»-brett som viste de aktuelle oppgavene som var tiltenkt, under arbeid og fullført innen den aktuelle iterasjonen.
- «Utvikler»-brett per person i gruppa – viste det samme som «Mine oppgaver»-brett. Det ga oss mulighet til å se på hverandres oppgaver.
- «Oppgaver»-brett som viste alle oppgaver.
- «Ikke ferdig»-brett som viste alle oppgaver som ikke ble fullført i den tiltenkte iterasjonen, og som i så fall burde oppdateres og settes til å jobbes med/fullføres i en senere sprint.

Alle prosjektbrettene kan sees i [vedlegg 5](#), men et eksempel på hvordan «kanban»-brettet så ut, kan sees nedenfor, i figur 2. Dette var brettet som vi benyttet mest for å, i fellesskap, skape et bilde over hva som var gjort, hva som ble gjort og hva som måtte gjøres videre. Dette er ment for illustrasjon, slik at selv om skriften i skjermbildet er liten, er det konseptet vi ønsker å illustrere her.

Figur 2 - GitHub projects prosjektbrett, "kanban", for Juvet.

## **GitHub Actions**

GitHub actions er et automasjonsverktøy for å automatisk bygge, teste og deployere prosjektkoden (GitHub, u.d.). Vi etablerte et sett med GitHub Actions i prosjektkoden der alle testene i prosjektet ble gjennomført etter å ha publisert en ny «pull request». En pull request er en måte å kvalitetssikre den koden som andre i prosjektet har skrevet, sikre seg mot potensielle konflikter i koden, samtidig som det er en måte å holde seg oppdatert på siste (foreslår) endringer (GitHub, u.d.).

## **Visual Studio Code (utvikling frontend)**

Visual Studio Code er en kode-editor (redigeringsprogram) (Visual Studio Code, u.d.). Valget av denne som editor på frontend-kode var naturlig da gruppemedlemmene var kjente med denne fra før og krevde dermed ingen opplæring.

## **IntelliJ IDEA (utvikling backend)**

IntelliJ IDEA er en editor for å skrive programvare (JetBrains, u.d.). Det er spesielt egnet for blant annet Java-basert kode. Valget av denne som editor på backend-kode var naturlig da gruppemedlemmene var kjente med denne fra før og krevde dermed ingen opplæring.

## **Draw.io**

Draw.io er et nettbasert tegneverktøy for utforming av diverse diagrammer (UML, ER etc..) (draw.io, u.d.). Det ble brukt for å utforme noen av diagrammene i rapporten.

## **Facebook Messenger**

Facebook Messenger er en nettbasert meldingstjeneste (Meta, u.d.). Det ble brukt for hurtig og uformelle beskjeder og info mellom gruppemedlemmene.

## **Zoom.us**

Zoom.us er et videomøte-verktøy (Zoom.us, u.d.). Ble brukt til veiledningsmøter med intern veileder ved OsloMet. Vi brukte Zoom isteden for Microsoft Teams på grunn av at Zoom er bedre integrert hos OsloMet enn andre slike typer tjenester.

## **Microsoft Teams**

Microsoft Teams er et videomøte-verktøy (Microsoft, u.d.). Det ble brukt i alle møter med gruppa og i alle møter med Fremtind. Fremtind inkluderte gruppemedlemmene i et «Teams-team» under sin organisasjon. På denne måten kunne vi henvende oss til ansatte i Fremtind som var relevante for vårt prosjekt.

## **Microsoft OneDrive**

Microsoft OneDrive er en nettbasert lagringstjeneste (Microsoft, u.d.). Alle rapporter og støttelitteratur i forbindelse med prosjektet ble lagret og delt mellom gruppemedlemmene via OneDrive. Dette gjorde også samarbeid og samskriving på disse dokumentene mulig.

### **Microsoft Word**

Microsoft Word er et digitalt skriveverktøy (Microsoft, u.d.). Alle rapporter og all støttelitteratur i forbindelse med prosjektet ble skrevet i Word. Dette gjorde også samarbeid og samskriving på disse dokumentene mulig.

### **Google Sheets**

Google Sheets er et regneark-verktøy fra Google (Google, u.d.). Dette ble brukt på initiativ fra Fremtind for å kunne registrere tilbakemeldinger under intervju med brukere. Dette ble også brukt til å analysere og utarbeide statistikk basert på tilbakemeldingene vi fikk fra brukerne.

### **FigJam**

FigJam er et skybasert digitalt «whiteboard-verktøy» som ble mest brukt under designsprinten (Figma, u.d.). Der kunne vi enkelt samarbeide og komme med visuelle tilbakemeldinger underveis. En fasilitator hos Fremtind holdte designsprint i FigJam da det var det foretrukne verktøyet for slike prosesser i organisasjonen.

### **Figma**

Figma er et digitalt verktøy for å enkelt sette opp visuell prototyping av nettsider og andre digitale visuelle flater (Figma, u.d.). Dette ble brukt for å bygge og teste vår prototype, samt videreutvikle denne. Resultatet av prototypen ble utgangspunktet for utviklingen og den visuelle representasjonen av vår tjeneste. Dette verktøyet er todelt; én del er Figma, hvor prototypingen foregår, og FigJam, hvor «white board»-delen foregår. Dette er et verktøy Fremtind bruker mye i sitt designarbeid for workshops og prototyping. Vi fikk derfor tilgang til deres organisasjons-område og fikk en rask opplæring i dette fra start (Figma, u.d.).

### **Docker**

Docker er et «plattform-som-en-tjeneste»-program for å opprette, kjøre og håndtere containere, og er en måte å kunne kjøre et program uavhengig av operativsystem/programvareplattform den kjøres på. Docker baseres på virtuelle datamaskiner som kan opprettes, bygges og samhandle i et system (Docker, u.d.).

### **Postman**

Postman er et verktøy for å teste API'er (Postman, u.d.). Det lar oss sende ulike spørrslinger til endepunktene i API'et med ønskede parametere i «header» og «body» som HTTP-protokollen definerer.

## **Amazon Web Services**

Amazon Web Services, forkortet AWS, er en sky-basert plattform-som-en-tjeneste-løsning som blant annet kan «hoste» (publisere) webapplikasjoner. AWS har mange flere valgmuligheter for oppsettet av en sky-basert løsning (Amazon Web Services, u.d.). AWS blir brukt av Fremtind i en økende mengde, og vil bli måten Juvet skal «hostes» på.

## **Yarn/Npm**

Yarn og Npm er pakkehåndteringsverktøy for JavaScript (npm, u.d.) (yarn, u.d.). Bruken av disse har blitt standarden for pakkehåndtering av JavaScript moduler i løpet av de siste 10 årene (Weber, 2022). Npm ble benyttet helt i starten av utviklingen, men ble endret til Yarn da dette var raskere i praksis.

## **Maven**

I likhet med Npm og Yarn er Maven et verktøy som holder orden på avhengighetene innen kodebiblioteker og -rammeverk som blir brukt i prosjektets kodebase (Apache Maven Project, u.d.). Til forskjell fra Yarn og Npm benyttes Maven i hovedsak under utvikling av Java-prosjekter.

### [2.3 Rammeverk/bibliotek brukt](#)

Vi vil her kort og generelt beskrive ulike rammeverk og bibliotek som er brukt i prosjektet. Det er ment som en introduksjon før det senere i rapporten vil beskrives på hvilken måte vi brukte disse rammeverkene/bibliotekene og sammenhengen mellom dem.

## **Java – Spring Boot**

Java er et typestrenge programmeringsspråk, som vil si at alle attributter må defineres med enten primitive datatyper eller klasser (Hartman, What is Java? Definition, Meaning & Features of Java Platforms, 2022). Java brukes til å utvikle i hovedsak backend-applikasjoner som manipulerer og lagrer data. Spring Boot er et Java-rammeverk som gjør det enklere og raskere å sette opp web-applikasjoner (Spring, u.d.).

## **React**

React er et komponentbasert JavaScript-bibliotek for å enkelt og raskt kunne bygge logikk på klient-siden til en nettside (React, u.d.). Fordelen med dette er at vi hurtig kunne få en klient-side opp og kjøre.

Vi bruker flere støtterammeverk for React som blant annet React Query, React Markdown og React Router, som kommer i tillegg til funksjonaliteten som allerede er i React.

## **TypeScript**

TypeScript er et «superset» av JavaScript, som utfører all logikk på frontend (Spínola, 2017). Et superset er en utvidelse av et eksisterende kodespråk. TypeScript kompilerer til JavaScript og gir en mer oversiktlig kode med enklere gjenbrukbare komponenter sammen med React (Prasanjith, 2020).

## **Sass**

Sass er et stilark for HTML-filer som har flere fordeler for større prosjekt enn ved «vanlig» CSS (Sass, u.d.). For eksempel har sass støtte for arv av stiler og gruppering («nesting» på engelsk) som gir en bedre oversikt og lesbarhet ved større prosjekt.

### **OpenID Connect**

OpenID Connect er en autentiseringsmetode som bygger på OAuth 2.0-protokollen (OpenID, n.d.). Denne brukes av Fremtind for autentisering mot deres tilgangsregister i Microsoft Azure AD.

### **Microsoft Azure AD**

Azure Active Directory er et skybasert identitets – og tilgangsstyringssystem (Microsoft, 2022). Dette er brukt av Fremtind og er nødvendig for oss å bruke da vi ville bruke Single Sign On (SSO) for pålogging til Juvet og for å hente info om brukerne. Dette ligger lagret i Azure AD.

### **Mockito og MockMVC**

Mockito og MockMVC er verktøy brukt til mock-testing. Mockito utfører enhetstester (mockito, u.d.). MockMVC er en del av Spring test-rammeverket og tester spring-kontrollerne (Rieckpil.de, 2020). Dette er testing med «mock»-innhold, det vil si oppdiktet data som er relevant for flyten i vår løsning.

### **JUnit5**

JUnit5-testing er enhetstesting av Java-koden og er et testrammeverk hvor det er mulig å teste på klasse- og metode-nivå (JUnit, u.d.). Fremtind bruker dette for enhetstesting av sine løsninger, tillegg legger Spring Boot opp til bruk av dette, og er derfor et naturlig valg. Dette ble brukt til testing av vår backend gjennom hele utviklingsfasen.

### **Object-Relational Mapping (ORM)**

For å opprette databasen med tabeller basert på Java-klassene og for å opprette spørrsager til denne databasen, benyttet vi en ORM som for relasjonsdatabaser i Spring Boot er Java Persistence API (JPA) (TutorialsPoint, u.d.). En ORM håndterer all interaksjon mot databasen, og er definert med egne metoder for å hente den dataen man ønsker fra databasen. På denne måten unngår vi å måtte skrive eksplisitte SQL-setninger i koden selv, som erfaringsmessig er en svakhet i koden, altså ikke beste praksis.

### **Jøkul designsystem**

Fremtind Forsikring har utviklet sitt eget designsystem, Jøkul, som er åpent og tilgjengelig for alle (Fremtind Forsikring, u.d.). Jøkul består av et bredt spekter av ulike komponenter og scss/sass-stilark med dokumentasjon for implementasjon som er ferdig laget og som vi har stått fritt til å bruke. Et designsystem er et sett med standarder og prinsipper for utforming av bedriftens visuelle flater.

Fikk presentert dette ved det første innledende møte med oppdragsgiver, som ga oss mulighet til å gjøre oss kjent med dette før oppstart av prosjektet. Dette var verdifullt da ingen av oss på gruppa hadde erfaring med Jøkul tidligere.

### **Swagger**

Swagger er et rammeverk for automatisk dokumentasjon av et REST API (Swagger, u.d.). Ved å annotere vårt API med Swagger-deklarasjoner, vil Swagger automatisk opprette et eget web-område med en visuell oversikt for alle endepunktene og beskrive spesifikasjonene for hver av disse. Dette kan videre brukes til å teste kommunikasjonen mot API’et direkte på web-området, i tillegg til å automatisk opprette dokumentasjon.

For vedlikehold og videreutvikling av løsningen er det en fordel å ha implementert Swagger fordi det gir en tydelig og strukturert oversikt over måten man kommuniserer med API’et på, uten å måtte sette seg inn i koden/strukturen som ligger til grunn i API’et.

## Lombok

Lombok er et støttebibliotek for Java som genererer «boilerplate»-kode (generell og ofte gjentatt kode) for java-klassene (Project Lombok, u.d.). Fordelen med dette er at vi slipper å skrive og holde disse oppdatert selv, da de automatisk generes og oppdateres ved oppstart av Spring Boot-prosjektet.

## 2.4 Arbeidsmetode

Valg av arbeidsmetode var innledningsvis bestemt av oppdragsgiver, som arrangerte en designsprint. Dette er, som mer utfyllende beskrevet i neste delkapittel, en metode for å raskt utvikle en testbar prototype på en løsning som kan bekrefte og avkrefte antakelser om denne løsningen. Dette inngår i smidig metodikk og det var derfor naturlig å fortsette utviklingen innen en smidig retning. Vi vil videre beskrive metodikkene og årsakene til at disse ble benyttet.

### 2.4.1 Designsprint

Ved oppstart ble det gjennomført en *designsprint* i regi av Fremtind Forsikring, hvor de stilte med én fasilitator, én utvikler og lederen for digital-avdelingen. På denne måten sikret Fremtind og bachelorgruppa at konseptet og løsningen ville være relevant for problemet.

I tradisjonell Lean-metodikk går en slik utviklingssyklus gjennom fire punkter (idé, produsere, lansere, lære), men i en designsprint kutter man ut to av disse punktene; produsere og lansere - og holder fokus på idé og læring. Dette er for å minimere fallhøyde i utviklingen av løsningen.

Designsprinten er en metode for å utvikle en brukerfokuseret prototype og teste denne i løpet av fire til fem dager (The Sprint Book, u.d.). Denne metoden ble utviklet av Google-ansatte og lansert i 2012, og korter ned tiden fra idéskaping til læring. Designsprint er basert på konseptene *Lean Startup* og *Design Thinking*, og er en serie øvelser som er veldefinerte og tidsavgrensede. Lean Startup og Design Thinking er i hovedsak metodikker som brukes i innovasjonsprosesser (Løvlie, 2021). Gruppen som gjennomfører denne prosessen, er gjerne tett knyttet til det som skal utvikles og har ofte en tverrfaglig bakgrunn. Antall personer i gruppen bør ikke overstige 7 stk. Under vår gjennomføring var vi 6 i sprintgruppen i tillegg til én fasilitator. Blant de 6 i gruppa må det være minst én «bestemmer» (engelsk: «decider»), som tar alle de viktige avgjørelsene - særlig dersom det er uenighet i gruppa om valgene som skal tas. Da har bestemmeren myndighet til å skjære igjennom og velge det de mener er det riktige valget.

Gjennom hele sprinten foregikk samarbeidet slik at man «jobber alene sammen» - altså hver gang en ny sekvens i sprinten startet, jobbet alle individuelt med oppgaven før alle samles og presenterte sitt arbeid. På denne måten vil gruppemedlemmene ikke påvirkes av hverandres tanker og arbeid, og når gruppa da samles vil sannsynligvis spekteret av løsninger være bredere enn ved samarbeid hele

veien. Etter å ha presentert det individuelle arbeidet, ble de(n) beste delen(e) av alles arbeid valgt og videreført til neste steg av sekvensen.

Designsprinten gikk over fire hele dager; hver dag med hver sine mål.

Designsprinten ble gjennomført på hjemmekontor og vi var derfor avhengig av et digitalt «whiteboard» som inneholdt relevante verktøy for gjennomføringen. Fremtind bruker Figma og FigJam, som vi fikk tilgang til, og opplæring i. Vår fasilitator hadde forberedt sider i FigJam som konkretiserte og samlet alt arbeid underveis i designsprinten. Dette ble videre viktig dokumentasjon til fortsettelsen av utviklingen av den tekniske løsningen, og til utarbeidelse av prosjektrapporten.

Selv designsprint-prosessen beskrives nærmere, gjennom oppgavene dag-for-dag, under prosessdokumentasjonen i [kapittel 3.1.1](#).

#### [2.4.2 Smidig utvikling](#)

Etter designsprinten var det vår intensjon å jobbe etter en smidig tilnærming til prosjektet med iterasjoner basert på smidig prosjektmetodikk. Dette viste seg å bli utfordrende. Mer om disse utfordringene og samsvar mellom plan og utførelse blir diskutert i [delkapittel 3.2.7](#). Som beskrevet i *Prosjektveilederen*, legger en smidig prosjektgjennomføring opp til en mer lærende prosess (Vaagaasar & Skyttermoen, 2021, ss. 41-44), og vurderinger i valget av prosjektmetodikk baseres på ulike momenter. Disse momentene var graden av usikkerhet i prosjektet, muligheten for endringer underveis, graden av brukerinvolvering, hvor utfordrende aktivitetsplanleggingen var, hvor nødvendig et tillitsbasert samarbeid var og viktigheten av tempoet i prosjektet. Jo høyere grad prosjektet scorer på disse momentene, jo bedre egnet er en smidig gjennomføringsmetode (Vaagaasar & Skyttermoen, 2021, s. 40).

Tiden fra designsprint til frist for levering av rapporten ble delt opp i sprinter på 2 uker per sprint. Hovedfokus og mål for de ulike sprintene og iterasjonene ble satt opp og definert i et Gantt-diagram for bedre oversikt.

#### [2.4.3 Tilbakemeldinger](#)

Tilbakemeldinger fra oppdragsgiver ble under designsprint gitt fortløpende da vi hadde daglige møter. Deretter fikk vi tilbakemeldinger gjennom en ukentlig Stand-up med Glenn, som var vår kontaktperson hos Fremtind. Mer om disse Stand-up-møtene og hva de innebar blir beskrevet i [delkapittel 3.1.2](#). Det ble også holdt demo'er for ansatte i Fremtind Digital – én tidlig i prosjektet, hvor prototypen og innsiktene fra brukertestene ble presentert. Her fikk vi generell og uformelt positive tilbakemeldinger på løsningen.

En ny demo ble holdt for ansatte i Fremtind i slutten av sprint 6. På dette tidspunktet var mye av backend ferdig utviklet og på frontend var forsida og søkeresultat-side i stor grad ferdig. Vi viste også en halvferdig aktivitetsside, som manglet mye informasjon. Flere ansatte i Fremtind Digital var til stede, særlig godt representert av ledergruppa i Fremtind Digital. Mer om tilbakemeldinger fra oppdragsgiver kan leses i [delkapittel 7.4](#).

Gjennom testing av løsningen fikk vi også verdifulle tilbakemeldinger. Mer om dette kan leses om i [kapittel 4](#).

### 3. Utviklingsprosessen

Prosjektet har gått gjennom flere faser med tydelige skiller, men noen prosesser har pågått gjennom hele prosjektets varighet. I alt har prosjektet vart i 21 uker, som vi valgte å dele opp i 9 sprinter med 2 uker per sprint. Vi vil her beskrive de ulike sprintene og fasene for prosjektet.

#### 3.1 Oppstartsfasen

Vi vil her presentere hvordan prosjektet tok form fra start og beskrive de ulike prosessene i denne fasen.

##### 3.1.1 Gjennomføring av designsprint

Som presentert i [delkapittel 2.4.1](#), ble det fra start av prosjektet gjennomført en designsprint fasilitert av Fremtind. Her vil vi presentere denne prosessen dag for dag:

Dokumentasjonen av designsprinten kan sees i sin helhet i [vedlegg 1](#).

##### Dag 1: Ekspertintervju, definerte mål og skisserte prototype

Gjennom intervju av eksperter på området fikk vi innsikt i hvordan situasjonen var i dag og hva de ønsket seg for en løsning av problemet. Disse ekspertene er personer som jobber «tett på» problemet og vil ha stor bruk for løsningen. De kan derfor si mye om hva som savnes med dagens løsning og hvordan de ser for seg morgendagens løsning burde være.

Med grunnlag i disse intervjuene ble det definert en «ledestjerne» som hovedmål for resten av designsprinten. Disse målene var utformet som en setning som startet med «Om 2 år ...». Vår ledestjerne ble «*Om 2 år er alle ansatte i Fremtind med på min fagutvikling gjennom å bruke og oppdatere Juvet hver dag*».

For å kunne holde et avgrenset fokus mot målet med sprinten, kom vi opp med ulike «sprintspørsmål». Disse spørsmålene inneholdte de viktigste antagelsene om hvorfor konseptet skulle feile eller bli en suksess, og var utformet som spørsmål startende med «Kan vi ....». Se [vedlegg 1, dag 1](#) for sprintspørsmålene som ble utvalgt til å være i hovedfokus under denne sprinten. Det viktigste spørsmålet vi burde fokusere mest på, ble bestemt av «bestemmeren» og lød slik: «*Kan vi få til en søkefunksjon som går på tvers av personer/emner/kurs?*».

Siste aktivitet dag 1 var å finne og vise eksempler på allerede eksisterende løsninger som svarte på de sprintspørsmålene vi hadde valgt ut. Disse eksemplene ble hentet som skjermbilder fra internett og presentert i presentasjonsbrettet. Videre ble dette bruk som utgangspunkt til å skissere en løsning som passet vårt problem. Skissesekvensen var todelt; først en sekvens hvor man tok tak i ett sprintspørsmål og skisserte en løsning som kunne svare til dette spørsmålet. Andre del av skissesekvensen gitt ut på en såkalt «crazy 8s»; hvor man tok tak i én spesifikk del av løsningen og skisserte 8 ulike utforminger av denne delen på maks 8 minutter. Dette tvang frem kreativiteten og gjorde samtidig at de skissene som ble laget var veldig enkle og lite detaljrike. Alle satte til slutt sammen de beste skissene til ett eget konsept hver for seg, som de kunne presentere til gruppa. Hele skisse-prosessen ble gjort individuelt for å unngå å bli påvirket av ideene eller arbeidet til de andre på gruppa. På denne måten endte vi sannsynligvis opp med et bredere spekter av ideer enn om man skulle ha samarbeidet gjennom hele skisseprosessen. De individuelle konseptene ble lagret til å presenteres neste dag.

## Dag 2: Bestemte hvilke deler av skissene som skulle prioriteres og bestemte en brukerreise

De individuelle skissene fra dag 1 ble presentert anonymt og uten forklaring for de andre i gruppen. Så fikk alle 20 minutter på å gjennomgå hverandres konsepter og «stempel» de delene eller områdene som man likte ekstra godt med et hjerte-stempel, i tillegg til å komme med oppklarende spørsmål til de delene av konseptene som var uklare. Videre ble alle de oppklarende spørsmålene lest opp av personen som hadde skissert konseptet, som deretter også måtte besvare spørsmålet og eventuelt forklare hva som var tiltenkt ved denne delen.

I neste steg ble testflyten for testen definert. Her skulle alle trinn fra start til slutt bestemmes.

Testflyten startet med at en Fremtind-ansatt hadde fått tilsendt en lenke til Juvet fra en kollega på sin Fremtind-epost. Deretter klikker den ansatte seg inn på lenken og kommer til landingssiden og ser søkerfeltet, kategorier og «populært innhold». I søkerfeltet står prompt-teksten «Hva vil du lære om i dag?» og testperson trykker «Søk» for å søke på valgt tema. På søker treff-siden får testpersonen opp innhold som passer til det søkerordet de søker på. Aktivitetene i søkerresultatene er presentert som klikkbar «kort» på siden, med to knapper, hvor den ene knappen er «Lagre til senere».

Testperson trykker på denne knappen for å lagre aktiviteten til senere. Rett etter å ha trykket på «Lagre til senere» kommer et meldings vindu opp nederst i vinduet som bekrefter at den valgte aktiviteten er lagt til på «Mine sider» og inneholder en lenke til å gå til denne siden. Dette er den ideelle slutten på testflyten.

Under siste sekvens for dagen ble det utarbeidet et storyboard som speilet stegene i testflyten. Se [vedlegg1, dag 2 – Designsprint: Storyboard](#). Der kom de viktigste ideene om utseende og funksjonalitet innen hvert av de 7 stegene frem. I denne delen ble det brukt 20 minutter til å individuelt gjennomgå storyboardet og å stille spørsmål ved de tingene som virket uklare med skissen. Dette ble grunnlaget for utarbeidelse av testbar digital prototype.

## Dag 3: Laget en testbar prototype av brukerreisen

Målet for dag 3 var å utarbeide en testbar prototype med utgangspunkt i storyboardet med skissene dag 2 hadde som resultat. Prototypen ble laget i Figma, et prototype-verktøy som Fremtind bruker. Vår fasilitator jobber som interaksjonsdesigner hos Fremtind, og hadde god kjennskap til dette verktøyet. Han ga en lyn-demo i den viktigste funksjonaliteten i Figma og kom med tips til hvordan vi kunne bygge komponenter for gjenbruk i prototypen. Måten prototypen ble bygd var ustrukturert og lite styrt av fasilitator, i motsetning til de tidligere sekvensene i designsprinten, men vi organiserte arbeidet mellom oss i gruppa og opplevde god flyt. De involverte fra Fremtind var med under byggingen av prototypen, og hadde på den måten påvirkningskraft gjennom å kunne komme med innspill på hvordan utseendet av prototypen og dens tenkte funksjoner burde være. Dette så vi som positivt fordi de har erfaring med slik type utvikling fra tidligere, og vil selv være potensielle brukere av denne løsningen.

Første prototype inneholdt 5 skjermbilder og kan sees i [vedlegg 1, dag 3](#).

## Dag 4: Testing av prototypen gjennom fem brukertester

Den fjerde og siste dagen i designsprinten ble alle brukertestene gjennomført. Fem brukertester ble holdt med fem ulike personer i Fremtind Digital, som alle vil være reelle brukere av løsningen.

Testpersonene hadde ulike stillinger i Fremtind, og ville av den grunn se på- og bruke Juvet på ulike måter. Testpersonenes stillinger strekte seg fra tech-lead, løsningsarkitekt, fagansvarlig design, frontend-arkitekt og utviklingsleder. Brukertestene ble gjennomført via Teams som en samtale mellom fasilitator og testpersonen. Testpersonen ble tilsendt en lenke til en side som viste vår

prototype som om det var en ekte nettside. Testpersonen ble så bedt om å dele sin skjem, slik at alle i samtalene kunne se hvordan de brukte ‘nettsiden’. Sprintgruppa hadde «tilskuer»-rolle i denne Teams-samtalen, slik at testpersonen ikke var klar over at vi observerte testen underveis.

Fasilitatoren vår brukte de 20 spørsmålene rundt våre antakelser som et slags manus for å kunne være konsekvent gjennom de 5 ulike testene. Disse 20 spørsmålene sees i tabellen i avsnittet under. Det var lagt opp til at fasilitator i liten grad ga instrukser, men at han stilte spørsmål som inneholdt hint til hva neste steg i testflyten var. På denne måten kunne vi se hva testpersonen enkelt fant frem til og hva som ikke var så enkelt å gjette seg fram til.

Under alle brukertestene satt sprintgruppen og noterte observasjoner og innsikter fra testene i et eget Google Sheets-regneark. I dette regnearket hadde fasilitator forberedt 20 ja-nei-spørsmål til brukertestene som ble brukt til å danne en statistikk på hvor godt vi hadde svart på de antagelsene til løsningen som ble gjort helt i starten av hele designsprint-prosessen. Til slutt ble alle disse svarene, innsiktene og observasjonene samlet i ett slik at man kunne få en oversikt over all infoen gruppa hadde samlet gjennom dagen. Mer om brukertesten i designsprinten beskrives i [delkapittelet 4.1](#).

Spørsmålene vi ønsket svar på under brukertestene skulle i hovedsak kunne gi svar på om de antakelsene vi gjorde under designsprinten stemte. Spørsmålene var utformet på denne måten: «Virker brukeren interessert i faglig utvikling?», «Skjønner brukeren at løsningen handler om kurs og aktiviteter?» og «Virker det som brukeren ønsker å bruke Juvet for egen faglig utvikling?». Dette var tre av de 20 spørsmålene som vi med brukertesten ønsket svar på. Alle spørsmålene kan sees i [vedlegg 3](#).

Hele arbeidsflyten i designsprinten kan sees i vedlegg, men i figur 3 nedenfor, vises de to viktigste bildene i et sammendrag av stegene vi gjennomførte, beslutningene og innsiktene vi satt igjen med etter å ha gjennomført prosessen. Dette sammendraget fikk vi av designsprint-fasilitator i etterkant av designsprinten, og kan sees i sin helhet i [vedlegg 1, dag 4](#).

## Hva har dere gjort?

Recap av uken som var.

- 🎯 Dere gikk fra bred problemstilling til fokusert utfordring.
- 🚀 Dere utforsket og definerte visjonære mål.
- 💡 Dere valget ut en rekke spørsmål dere ville ha svar på.
- 💡 Dere kom opp med **seks konsepter**, og hentet de beste ideéne fra hver av de.
- 🛠️ Dere har designet en testbar, interaktiv prototype i løpet av åtte timer.
- 📝 Dere validerte konseptet deres på ekte brukere, i løpet av fire dager.
- 📝 Dere samlet hele **898 innsikter!**
- 📊 Dere fikk svar på spørsmålene dere lurt på.

Figur 3 - Oppsummering av designsprint, "Hva har dere gjort?".

## Innsiktssyntese

Et par kjappe stats.

- 👉 Dere har fått bekreftet behovet for tjenesten.
- ✅ **Åtte av ti** sprintspørsmål ble bekreftet
- 💡 Kun ett sprintspørsmål må behandles med varsomhet
- 🚫 Kun ett sprintspørsmål ble underkjent
- 📝 Dere samlet 233 viktige innsikter (**filtrert ned fra 898!**)
- 📊 121 positive, 51 negative, og 31 sitater\*, 30 generelle

\* Mathias sine notater var alle i bold, så det var vanskelig å filtrere ut hva som var sitater, mulig jeg har missa noen. Sjekk over en gang sammen.

Figur 4 - Oppsummering av designsprint, "Innsiktssyntese".

### 3.1.2 Møter og kommunikasjon

Stand-up møter er kalt nettopp "Stand Up" fordi de konseptuelt skal utføres stående for å holde møtene korte. I disse møtene går man gjennom hva man har gjort siden sist møte i en kort-og-godt forklart stil. Ettersom vi hadde fått tilgang til gjestebrukere i Microsoft Teams i Fremtind sine systemer, hadde vi også enkel tilgang til å sette opp møter innad i organisasjonen. Det ble satt som et mål å ha stand-up møter med vår kontaktperson fra Fremtind hver uke, samt hver tirsdag, onsdag og fredag innad i gruppen om timeplanene våre tilbydde muligheten.

De første møtene innad i gruppen bestod for det meste av å diskutere hovedansvar for hver av oss, i tillegg til valg av oppsett for applikasjonen. Ettersom vi allerede var delvis kjent med hverandres styrker og svakheter, fikk vi enkelt fordelt oppgaver og ansvar i samsvar med gruppens styrker. Når vi så tok opp hvordan vi skulle sette opp prosjektet, ble vi møtt med ett av våre første store valg. Fremtind hadde ingen mal på hvordan de satte opp webapplikasjonene deres så vi ble derfor nødt til

å planlegge dette uten en struktur fra organisasjonen å referere til. Det ble hovedsakelig dette temaet de første møtene med vår kontaktperson i Fremtind handlet om.

### *3.1.3 Utfordringer og diskusjoner rundt grunnleggende oppsett*

Vi bestemte oss i utgangspunktet for at det grunnleggende oppsettet skulle bestå av ett repository for frontend og et separat repository for backend. Dette endret vi, etter en del omtanke og konsultasjon med Fremtind, til å være ett repository som inneholdte både frontend og backend. Etter at dette grunnleggende oppsettet var klart, var vi veldig interesserte i å i det minste få se et eksempel på hvordan en webapplikasjon utviklet av Fremtind var satt opp bak kulissene. Dette viste seg å være problematisk, ettersom det å gi oss tilgang til kildekoden bak Fremtind sine webapplikasjoner skapte grunnlag for sikkerhetsrisiko for konfidensialiteten til organisasjonens data. Derfor ble vi heller enige om at vi skulle få møte med en utvikler fra Fremtind, med erfaring innen oppsettet av deres applikasjoner, for å diskutere muligheter for oppsett.

### *3.1.4 Demonstrasjon av prosjektet til Fremtind-ansatte*

Med hovedansvarene fordelt og det grunnleggende oppsettet ikke langt fra å være klargjort, ville Fremtind at vi skulle fremføre prosjektet for ansatte i Fremtind Digital. Dette var en ide som vår kontaktperson Glenn mente hadde potensialet til å høyne interesse for vår løsning blant de ansatte i Fremtind. Demonstrasjonen av vårt prosjekt kunne muligens også ha bieffekten av å gjøre det enklere for oss å få kontakt med sluttbrukere for senere brukertester av løsningen. Vi så på dette som meget positivt, ikke bare for å få litt øvelse til den avsluttende prosjektpresentasjonen etter innlevering, men også fordi det ga oss følelsen av at dette prosjektet var etterspurt og interessant for de ansatte.

Vi valgte at demoen vår skulle være en presentasjon som ville inneholde en introduksjon av hvem vi var, hvor vi studerte og hvilke problemer vi hadde lyst til å hjelpe Fremtind med å løse. Vi hadde også en gjennomgang av prototypen vi produserte i designsprinten for å gi sluttbrukerne en god visuell representasjon av hva vi ønsket å oppnå med prosjektet. Tilbakemeldingene vi fikk var meget positive, med flere ansatte som sa de var imponerte over fremgangen vi hadde så langt og ville bli holdt oppdatert på videreutviklingen av løsningen. Det eneste som nå gjenstod før vi kunne sette i gang utviklingen for fullt var å bestemme oss for hvilket utviklingsmiljø vi skulle bruke. Altså hvilke kodespråk og hvilke kodebiblioteker vi ville bruke som grunnlag for utviklingen.

### *3.1.5 Starten av utviklingen*

Etter designsprinten, som tok plass i den første uken av prosjektet, hadde vi allerede fått ett meget godt utgangspunkt for videre planlegging og målsetting. Vi hadde nå klargjort, i samarbeid med representanter fra Fremtind, hva de grunnleggende målene for prosjektet skulle være. Dette samt hvilke krav, både funksjonelle og ikke-funksjonelle, vi burde fokusere på å oppfylle med vår løsning. Etter så å ha idémyldret oss fram til hvordan vår applikasjon skulle fungere, kom vi i plenum fram til et konsept for vår applikasjon som vi deretter fikk konstruert en prototype basert på. Denne prototypen var interaktiv med funksjoner som var lagt opp til å følge en brukerreise som ville være typisk for de fleste brukerne av løsningen. Prototypen var også designet med tanke på reglene i Fremtinds designsystem Jøkul så presist som mulig. Til slutt hadde vi også testet prototypen på flere sluttbrukere fra Fremtind, og fått samlet inn hundrevis av innsikter/observasjoner og mulige endringer vi kunne ta for oss i utviklingen. Med en testet prototype av webapplikasjonen, hadde vi

innad i gruppa nå et likt syn på hvilke hovedfunksjoner vi skulle legge mest vekt på i utviklingen av løsningen.

For å komme i gang med den konkrete løsningen, var første steg å fordele oppgaver og ansvar gjennom opprettelsen av et kanban-brett i GitHub repository'et for applikasjonen. Her fikk vi satt opp en plattform for deling av fremgang i utviklingen av spesifikke funksjoner, fullførelse av administrative oppgaver og endringer gjort i rapporten. Dette oversiktlig brettet med fordelte oppgaver og ansvar er også perfekt for gjennomgang av fremgang i løpet av Stand-up møter.

Ettersom vi i løpet av designsprinten hadde planlagt å bruke Jøkul som designsystem tenkte vi Fremtind mest sannsynlig hadde en standard for utviklingsmiljø også. Det var ikke tilfellet ettersom de ikke hadde en global standard, men det var uansett tydelig at de hovedsakelig tar i bruk Java Spring Boot i backend og TypeScript React i frontend.

Dette var midt i blinken for oss ettersom to av oss hadde kunnskap til Spring Boot, mens de andre to hadde et fag dette semesteret som ville gjennomgå bruken av nettopp Spring Boot. Én av oss hadde også kunnskap til React og kunne veilede resten med konsultasjon eller henvisning til opplæringskilder på nettet. Både Spring Boot og React er mye brukte rammeverk, og er dermed godt dokumentert og testet (Kapoor, 2021) (Grabski, 2020). Ettersom det også ville være enklere å få direkte hjelp fra Fremtind med koden vår om vi valgte Spring Boot og React som språk, var valget enkelt og krevde ikke ytterligere omtanke. Med det grunnleggende utviklingsmiljøet klargjort, og flere gode tilbakemeldinger fra organisasjonen, startet vi utviklingsfasen med god moral og høy motivasjon.

### 3.2 Utviklingsfase

Overgangspunktet fra oppstartsfase til utviklingsfase ble definert til det tidspunktet det grunnleggende utviklingsmiljøet ble etablert.

Utviklingsfasen ble delt opp i 9 sprinter. Disse sprintene hadde hver sine fokusområder, hvor den første innen produksjonsdelen (startet i sprint 1) hadde fokus på å utvikle en prototype og samtidig starte utviklingen av MVP. En MVP står for «minimum viable product». Dette kan ifølge Thomas Svensen i BEKK, oversettes til «*minste, akseptable leveranse*» og betyr «... den versjonen av et nytt produkt som gjør at et team kan innhente mest mulig validert læring om kunder med minst mulig anstrengelse» (Svensen, 2016). Det skulle vise seg at vår planlegging var i overkant optimistisk med tanke på tidsrammen og omfanget av kravspesifikasjonene.

Det var gjennom hele utviklingen en god flyt av kommunikasjon innad i prosjektgruppen, noe som førte til en meget god gruppodynamikk. I situasjoner med dårlig motivasjon eller usikkerhet rundt egen innsats for hver enkelte, ble dynamikken oppholdt ved å være åpne med hverandre om personlige utfordringer i henhold til prosjektet. Vi tok hensyn til- og støttet hverandre fram til å nå en høyere motivasjon for prosjektet samt en dypere interesse i å utføre det på en god måte.

Vi vil i dette delkapittelet gjennomgå fasene slik de ble definert i prosjektplanene, for så å drøfte samsvaret mellom planlegging og faktisk gjennomføring i slutten av delkapittelet.

#### 3.2.1 Prototyping

Siste dag av designsprinten resulterte i en prototype av de sidene som demonstrerte kjernefunksjonaliteten i vår løsning. Denne ble laget på én dag, og dannet et godt utgangspunkt med tanke på både utseende og hvordan sideoppsettet kunne se ut. Vi justerte prototypen basert på de

tilbakemeldingene vi fikk under brukertestene. Første utkast av prototypens landingsside, som ble utarbeidet under designsprinten, så slik ut:

The screenshot shows the Juvet prototype landing page. At the top, there's a header with a logo (a stylized 'J'), a search bar, and a menu icon. Below the header, a main title reads "Velkommen til Juvet, kompetanseplattformen av og for Fremtindere". A search bar with placeholder text "Hva vil du lære om i dag?" and a "Søk" button follows. Below the search bar are five category icons: Koding (code), Design, Aktiv (active), Pedagogikk (pedagogy), and Ledelse (leadership). A banner below these categories says "16 aktiviteter begynner allerede i morgen". The main content area is divided into two sections: "Populært faginnhold siste måneden" and "Faginnhold for deg basert på dine interesser". Each section contains three cards, each with a title, a brief description, a user profile, and two buttons: "Meld på" and "Lagre til senere".

Figur 5 - Prototypen av landingssiden for Juvet fra designsprint.

Brukertesten tok for seg kun de tre sidene av prototypen som ble laget under designsprinten (blant annet landingssiden i figuren over). Vi var derfor nødt til å lage de resterende sidene for å lage en komplett visuell representasjon av hele løsningen vår. De resterende sidene var: En side hvor man ville kunne opprette ny aktivitet, en side som viste full og detaljert oversikt over én bestemt aktivitet, og en profilsida hvor brukere kunne registrere sine interesse- og kunnskaps-knagger («tags») og se lagrede, påmeldte og egenlagde aktiviteter.

Andre utkast av prototypen, med de manglende sidene lagt til, ble i stor grad ferdigstilt i første uke av sprint 2. Fullstendig oversikt over endelig utkast av prototypen sees i et selvstendig dokument med navn «*Prototype Juvet.pdf*» som vedlegg til rapporten. Den fullstendige prototypen ble ferdig én uke bak opprinnelig plan. Hvordan vi håndterte forsinkelser kan leses om i [delkapittel 3.2.7](#).

### *3.2.2 Utvikling MVP*

Etter at utviklingsmiljøet var etablert, begynte vi på de sentrale modellene for løsningen i backend og de grunnleggende komponentene i frontend. Dette var primært aktivitet-modellen og bruker-modellen for backend, og aktivitetskort (for å presentere aktivitetene) og navigasjonsbar for nettsiden for frontend. De sentrale modellene for frontend beskrives nærmere i [delkapittel 5.4.2](#), mens de sentrale modellene i backend beskrives i [delkapittel 5.4.3](#).

#### *3.2.2.1 Docker-container-hosting*

Fra start av kodeutviklingen ble det satt opp et GitHub-repository på Fremtind sin eksterne GitHub-side. Der ga vi repository'et navnet «Juvet». Vi delte opp prosjektet i en backend-del og en frontend-del. Begge disse delene lå under samme versjonskontroll, men de var i praksis to adskilte prosjekter som ble utviklet parallelt. Det ble bestemt at disse skulle åpnes adskilt; backend-prosjektet ble kun åpnet i IntelliJ IDEA og frontend-prosjektet kun åpnet i Visual Studio Code. Ved å gjøre det slik kunne vi tydelig differensiere mellom de to delene. Med samme delemåte ble et docker-container-oppsett satt opp for å inneholde hver sin del av backend og frontend i hver sin container. Dette ble definert i tre docker compose-filer (docker konfigurasjonsfiler); én i juvet-mappen, én i frontend-mappen og én i backend-mappen. Frontend-containeren ble definert med en port for kommunikasjon på adresse 3000. Backend-containeren ble definert med en port for kommunikasjon på adresse 8080. I docker-compose-filen ble også lenker til frontend og backend gitt som client-server (frontend) og resource-server (backend), slik at nginx-serveren visste hvor den skulle finne de ulike ressursene.

Backend-severen var basert på Java Spring Boot, som kjøres på en Apache Tomcat-server. En Apache Tomcat-server er en Java-basert webserver (CBR Staff Writer, 2020).

```
1  services:
2    resource-server:
3      build:
4        context: backend
5        dockerfile: Dockerfile
6      ports:
7        - "8080:8080"
8      restart: always
9      # environment:
10     client-server:
11       build:
12         context: frontend
13         dockerfile: Dockerfile
14       # volumes:
15       #   - './app'
16       #   - '/app/node_modules'
17       ports:
18         - "3000:3000"
19       # environment:
20       #   - CHOKIDAR_USEPOLLING=true
21     depends_on:
22       - resource-server
23   nginx:
24     restart: always
25     build:
26       context: ./nginx
27     ports:
28       # The public available ports on the host machine
29       - "80:80"
30       - "443:443"
31     links:
32       - client-server:frontend
33       - resource-server:backend
34     # logging:
35     #   # Disable nginx logging
36     #   driver: none
37     depends_on:
38       - resource-server
39       - client-server
```

Figur 6 - Docker-oppsett i rotmappe (fra ./juvet/docker-compose.yml).

Under utvikling av løsningen valgte vi å bruke en in-memory H2-database. «In-memory» betyr at det er en database som kun lever lokalt på den aktuelle datamaskinen backend-tjenesten blir kjørt på. H2-databasen er en rask relasjonsdatabase som har et enkelt oppsett for å hurtig koble opp til en Java-basert backend-tjeneste (javaTpoint, u.d.). Den er ikke anbefalt som database ved produksjonssetting av løsningen og ble derfor kun brukt under utviklingsfasen (Metabase, u.d.). Siden utviklingen ikke gikk til en produksjonsfase, men systemet fortsatt var i en utviklingsfase ved prosjektslutt, ble H2-databasen fortsatt benyttet. Vi gjorde et forsøk med å migrere databasen til en AWS RDS MySQL-database (forklares i [delkapittel 3.2.2.3](#)), men på grunn av ytelsesproblemer ble migreringen aldri fullt gjennomført. Se mer om dette i [delkapittel 3.3](#).

Vi valgte å bruke en relasjonsdatabase fremfor en ikke-relasjonell database fordi det var en tydelig struktur og sammenheng mellom våre klasser i backend og det som skulle lagres.

### 3.2.2.2 AWS Fargate-hosting

I starten av sprint 6 valgte vi å sette i gang med deployment til AWS Fargate hosting. Deployment betyr å flytte eller «publisere» en kjørende versjon av kodebasen fra å kjøres lokalt på en datamaskin til å kjøre eksternt på en server eller «i skyen» slik at den er tilgjengelig for flere (Oracle,

u.d.). AWS Fargate er en tjeneste fra Amazon Web Services hvor kodebasen kjøres i containere på virtuelle datamaskiner. Vi fikk tilgang til Fremtinds område på AWS og i tillegg til et eget «sandbox»-prosjektområde der. På denne måten fikk vi de tilgangene vi trengte for å sette opp en løsning som passet vårt prosjekt samtidig som vi ikke sto i risiko for å påvirke noen av de andre tjenestene som Fremtind allerede hadde kjørende på sitt AWS-område. Vi valgte å sette opp en tidlig versjon av vår løsning på AWS for å unngå å potensielt møte på større utfordringer rundt dette senere i prosjektet. Dette ble gjort med hjelp av en veiledning på video via YouTube (Be a Better Dev, 2021) i tillegg til å bruke AWS sin dokumentasjon for Fargate (Amazon Web Services, u.d.).

En viktig endring med løsningen ved deployment til AWS, var at docker-container-oppsettet vi brukte i startfasen ble erstattet av AWS sin versjon av container-basert kjøring av programvaren. Det betydde at konfigurasjonsdetajlene som ble definert i docker-compose-filen i rotmappen ikke lengre var relevant. Disse detaljene ble isteden konfigurert gjennom ulike parametere under oppsettet av AWS-hostingen. En mer detaljert beskrivelse av AWS-Fargate-oppsettet kan sees i [delkapittel 5.5](#).

#### 3.2.2.3 AWS MySQL database

Amazon Web Services tilbyr også en database-løsning som ble utforsket mot slutten av utviklingsfasen, gjennom sin «RDS»-tjeneste. RDS står for «relational database service» (Amazon Web Services, u.d.). Gjennom denne tjenesten ble det etablert en MySQL-database «i skyen». Dette var for å teste om AWS MySQL-databasen kunne erstatte H2-databasen som vi hadde arbeidet med under hele utviklingsfasen så langt. AWS's MySQL-database var tregere å aksessere enn H2-databasen. Dette var på grunn av at H2-databasen er en in-memory-database, mens MySQL-databasen måtte hente databaseinnholdet fra en ekstern database via en nettadresse. Den første MySQL-databasen som ble etablert for å i første omgang teste at tilkoblingen, initialiseringen av databasen og spørninger mot databasen fungerte som de skulle. Men, som beskrevet i tidligere i dette kapittelet og i [kapittel 3.3](#), var det utfordringer med hastigheten på henting av dataene fra denne databasen i vår applikasjon.

#### 3.2.2.4 Single Sign On med AzureAD

Siste hoveddel av utviklingen dreide seg i stor grad om å få implementert innloggingen til løsningen. Autentiseringen av Fremtind-brukere via AzureAD-plattformen, hvor alle ansatte var registrert med egen profil, for pålogging på Juvet var en viktig del av løsningen. Ettersom alt innhold i Juvet er personalisert, var det viktig at denne delen fungerte slik vi ønsket. Vi erfarte store utfordringer knyttet til implementering av denne delen, som var en av grunnene til at vi ikke fikk tid til å utvikle annen ønsket funksjonalitet. Mer om utfordringen og valgene kan ses i [delkapittel 3.3](#), mens hvordan vi faktisk fikk løst det til slutt er beskrevet i [delkapittel 5.2.4](#).

#### 3.2.2.5 Utvikling av frontend med React og Jøkul

Fremtinds designsystem ble i stor grad brukt til å utvikle frontend-delen av applikasjonene. Dette tok noe tid å sette seg inn i, men var til stor nytte for prosjektet sammen med egenlagde React-komponenter. Mer om hvordan vi bygget frontenden kan sees under [delkapittel 5.2.2](#).

Vi måtte bli enige om en del regler for hvordan vi skulle anvende React i applikasjonen vår for å opprettholde en konsekvent kodebase.

#### Klasse eller funksjonelle komponenter

Vi valgte å bruke funksjonelle komponenter til fordel for klasse-komponenter i applikasjonen vår. En

komponent er en gjenbrukbar kodebit av logikk, struktur og styling som presenterer noe data. Funksjonelle komponenter er helt vanlige JavaScript/TypeScript funksjoner som tar inn noe data og spytter ut en presentasjon av den dataen. Funksjonelle komponenter har av erfaring til fordel å resultere i mindre bugs og færre linjer kode.

### Logikk i tilpassede hooks

React har en rekke funksjoner man kan hooke inn i komponentene sine for å håndtere blant annet tilstand av komponenter. Vi valgte å bruke tilpassede React hooks for å abstrahere ut så mye logikk som mulig fra komponentene. Det finnes mange innebygde React hooks, men vi har også valgt å lage våre egne. Ved å bruke egne hooks kan vi si i en komponent at vi ønsker å få tak i noe data, men så er det opp til hook'en å bestemme hvordan den skal få tak i den dataen. Vi kan gjenbruke hooks i flere komponenter. Det gjør at logikk ikke trenger å dupliseres og testing av komponentene vil være mye enklere. For mer om testing av komponentene, se «Testing på frontend» under [delkapittel 4.2](#).

### Testing i utviklingsfasen

Flere ulike tester ble gjennomført underveis i utviklingen. Dette forklares mer i kapittelet om testing, [kapittel 4](#).

#### 3.2.3 Iterasjon 1, 2 og 3

På grunn av at utvikling av MVP tok lengre tid enn det vi hadde forutsett, ble det ikke tid til å gjennomføre utviklingen gjennom flere iterasjoner, på den måten vi hadde tenkt. Ambisjonen var å få tid etter å utvikle en MVP til å legge til tilleggsfunksjonalitet som i kravspesifikasjon ble nevnt som «kan»-krav. For eksempel «Som oppdragsgiver ønsker Fremtind at løsningen *kan* «gamify» («spillifiseres») deltakelse på aktiviteter».

Vi jobbet iterativt med utvikling av MVP, hvor modeller og struktur på både frontend og backend først ble etablert med de nødvendige attributtene, for så å bli utvidet ytterligere.

Mer om forsinkelse kan sees i [delkapittel 3.2.7](#).

#### 3.2.4 Ferdigstillelse og finpuss av løsningen

Ferdigstillelse av løsningen gikk i stor grad ut på å teste integrasjonen OpenID Connect og AzureAD for pålogging av brukere. Dette var en utfordring som krevde mye tid gjennom hele utviklingen, men som var en sentral brikke å få på plass i løsningen. I tillegg ble det gjennomført en akseptansestest for å kunne si noe om måloppnåelse for kravspesifikasjonen. Videre ble praktiske ting som hvordan overlevering av prosjektkoden skulle foregå og hvor backlog for videre forbedringer og videreutviklingen skulle ligge, slik at Fremtind enklast mulig kunne overta dette.

#### 3.2.5 Leveranse til oppdragsgiver

Ettersom alle kodebasen og systemoppsett er gjort innenfor Fremtind sine system (AWS, AzureAD og GitHub), var det ikke nødvendig å gjøre noe spesifikt for å overlevere løsningen til oppdragsgiver annet enn å samle den koden vi ønsket å levere. Det eneste som i utgangspunktet ikke var tilgjengelig for oppdragsgiver i deres systemer, var brukerveiledning av løsningen. Dette ble overlevert til kontaktpersonen hos oppdragsgiver som PDF.

### *3.2.6 Dokumentering*

I dette delkapittelet vil de relevante dokumentasjonsmåtene bli presentert.

#### **Prosjektdagbok**

Prosjektdagboken ble et godt verktøy og en slags logg for hva som ble gjort når, og hvilke stadier, valg og utfordringer som oppsto i prosjektløpet. Spesielt var dette nyttig for notater av detaljer som var viktige i utviklingen og for utarbeidelse av prosessdokumentasjonen i denne rapporten.

#### **Brukerveiledning for løsning**

For å gi et tydelig bilde av hvordan løsningen er ment å brukes, og hvilke muligheter som finnes, ble brukerveiledning med skjermbilder laget. Den største utfordringen med denne var at den ikke kunne skrives presist eller i sin helhet før hele løsningen var ferdig utviklet på både frontend og backend.

Denne dokumentasjonen sees i sin helhet i [kapittel 6](#).

#### **Sluttrapport**

Sluttrapporten (dette dokumentet) ble påstartet sørka midtveis i prosjektperioden. På dette tidspunktet hadde bildet av hvordan løsningen kom til å være utkristallisert seg, slik at arbeidet med rapporten også kunne gjøres med en større presisjon.

### *3.2.7 Samsvar mellom prosjektplan og reell gjennomførelse*

Som nevnt i starten av dette kapittelet, var antakelsene for tidsbruk på de ulike delene av prosjektet i overkant optimistisk. Til tross for at tidsplanen for produksjonsfasen ble kraftig forskjøvet og iterasjonene på MVP'en kuttet, ble mye av det andre som var planlagt gjennomført i henhold til planen. Oppstartsfasen ble gjennomført som planlagt og utvikling av prototype overholdte i stor grad tidsplanen. Generelt var alt vi gjorde i løpet av de første åtte ukene etter designsprinten også på linje med den planlagte fremgangen, men vi ble senere møtt med flere forsinkelser på veien. Vi opplevde i løpet av disse første ukene at vi hadde en god arbeidsflyt, noe som muligens gjorde at det ble vanskeligere å innse mulige rom for forsinkelser gjennom valgene vi tok.

Først hadde undervurdert tidsbruk for oppsettet av utviklingsmiljøet. I tillegg ble det mer og mer klart at prosjektets omfang sannsynligvis var i overkant av det vi opplevde som forventet arbeidsmengde av en bacheloroppgave innen den tiden som var til rådighet. Ikke bare utviklet vi et konsept for å løse en stor problemstilling og testet det gjennom en prototype, men vi utviklet deretter også en nærmest fullstendig applikasjon basert på prototypen med alle grunnfunksjoner til stede. For hver iterasjon hadde vi mer planlagt enn det vi fikk gjennomført. Vi endte derfor opp med å motvillig legge fra oss funksjoner vi gjerne ville hatt med i det fullførte produktet. Dette skyldes i stor grad vår optimisme.

Vi håndterte nærmest alle forsinkelsene på lik måte; med å se på prosjektet så langt i sin helhet og innse verdien av det vi allerede hadde fått til. Det ble på denne måten enklere å være optimistisk for at vi, tross forsinkelsene og de mulige nye prioriteringene vi måtte ta for oss, ville ende opp med et produkt vi kom til å være stolte av. Selv om vi ikke ville få muligheten til å legge til all funksjonaliteten vi hadde håpet på, beholdte vi tanken og troen på at vårt beste produkt innen prosjektets tidsrammer fortsatt ville være til stor verdi for Fremtind. Denne tankegangen ble også styrket av de positive tilbakemeldingene og stolthet fra våre representanter i Fremtind.

Under dokumenteringsdelen av prosjektplanen var det fra start satt opp kun én demo av løsningen, men ved endt prosjekt hadde vi gjennomført til sammen 3 demo'er for Fremtind. Én ved oppstarten, én ca. halvveis i prosjektet og én ved prosjektets slutt.

### 3.2.8 Kvalitetssikring

For å kvalitetssikre arbeidet under utviklingen ble blant annet GitHub projects brukt. Fordelen med GitHub projects var at vi kunne knytte oppgaver innen kodeutviklingen direkte til en «issue», eller «utfordring» på norsk, i prosjektbrettet. Når disse utfordringene trengte en felles gjennomgang eller ble løst, ble den aktuelle utfordringen gjennomgått av den eller de i gruppa som hadde ansvar for å løse den. I tillegg var det mulig å kommentere disse utfordringene i prosjektbrettet hvis noen i gruppa hadde tanker, tips eller spørsmål til utfordringen. Alle utfordringer som var i fremgang, trengte gjennomgang eller hadde status som ferdig i prosjektbrettet, ble gjennomgått i plenum under standup-møtene i gruppa.

## Rutiner for Git

Testene gjennom GitHub Actions ble utført ved en ny pull request, som nevnt i [delkapittel 2.2](#). Det ble angående disse bestemt en betingelse for at minst én i gruppa måtte godkjenne en pull request før den kunne sammenslås («merge») med hovedgreinen i prosjektet. Hovedgreinen og konseptet med greiner er at utviklingen av kode kan isoleres for å ikke påvirke annen kode i prosjektet, for så å kunne slås sammen igjen (GitHub, u.d.) På denne måten kunne vi enklere holde oversikt over utviklingen som ble gjort av andre på gruppa og oppdage mulige feil og misforståelser underveis. Det ble også bestemt at disse gjennomgangene og godkjennelsene av pull requests skulle prioriteres fremfor annet arbeid i prosjektet. På denne måten unngikk vi at vedkomne som hadde laget pull requesten ikke ble unødig hindret i å arbeide videre.

Produksionsgreinen i prosjektet var på «main»-greinen. For å starte et arbeid med en ny funksjonalitet ble det opprettet en ny grein med et relevant navn for den funksjonaliteten som skulle utvikles. Dersom det også var en utfordring i prosjektbrettet tilknyttet denne funksjonaliteten, ble gjerne id-nummeret på utfordringen en del av navnet på greinen. På denne måten ble det enklere å holde oversikt over hvor den nye funksjonaliteten ble utviklet.

## 3.3 Store utfordringer og valg

Underveis i prosjektet har vi blitt stilt ovenfor utfordringer som måtte løses, og valg som måtte tas, gjerne i forbindelse med de store utfordringene. Disse utfordringene har vært tidkrevende og var også grunner til forsinkelse i prosjektet.

### Tilganger

Fysisk tilgang til Fremtind-kontoret var i starten av prosjektet ikke relevant på grunn av pandemien, men utover prosjektet, da restriksjonene avtok, var dette ønskelig fra alle i prosjektgruppa. Det ga en større nærhet og tilgjengelighet til oppdragsgiver som opplevdes som verdifull.

Andre tekniske tilganger som krevdes for å utvikle og deployere systemet på den måten vi ønsket, tok også lengre tid enn forventet å få på plass. Selv med en undertegnet taushetserklæring fra alle i prosjektgruppa, kunne ikke Fremtind gi tilgang for alle til deres systemer. Gruppemedlemmet som var ansatt i Fremtind hadde en jobb-PC som ble inngangen til å kunne etablere og kontrollere

instanser innenfor blant annet AWS og for å enklere komme i kontakt med ressurser internt i Fremtind.

### Sikkerhet/autentisering

I all hovedsak dreide den største utfordringen seg om autentisering mot oppdragsgivers AzureAD-identitetssystem. Det viste seg at AzureAD og Spring sitt sikkerhetsbibliotek, Spring Security, ikke snakket så godt sammen som først antatt. Autentiseringsflyten vi ønsket var det få veiledninger og eksempler på. Oppdragsgivers it-arkitekt var også godt involvert i å løse denne utfordringen og kunne bekrefte at det ikke virket til å være en enkel løsning på dette. Etter mye arbeid på denne delen, fikk vi til slutt en fungerende autentisering slik vi ønsket på backend. Vi vurderte en periode å gjøre dette med en annen flyt, ettersom den ønskede flyten var mer krevende å implementere.

### Databasemodellering

Under utviklingen av databasestrukturen hadde vi for dårlig forståelse av hvilke konsekvenser de ulike implementeringene og valgene vi tok hadde. Det var valget om hvilken type måte vi hentet ut data fra relasjoner i en spørring, som viste seg å være kostbar i tid ved emigrering fra H2-database til AWS MySQL database. Her hadde vi valget mellom «eager» og «lazy» lasting av data, hvor vi valgte «eager» ettersom det fungerte med algoritmene vi hadde utviklet og gikk raskt med H2-database. Ved å bruke «eager» lasting hentet vi ut all data i alle relasjoner den gitte tabellen/entiteten hadde (Mahesh, 2019). Det vil kort sagt si at vi hentet ut unødvendig mye data som tok lengre tid. På den andre siden, med «lazy» lasting, ville ikke relasjonene bli hentet ut (Mahesh, 2019). I vår applikasjon vil det si at listen over forfattere til en aktivitet ikke vil bli hentet ut ved en spørring mot aktivitet-tabellen. En løsning på dette vil være å endre litt på databasestrukturen og endre noen relasjoner fra «eager» til «lazy» slik at det ikke blir en ond sirkel av datahenting.

### Prosjektomfang

Omfanget av prosjektet og tidsbruk var vanskelig å forutse ved prosjektets oppstartsfase. Vi utviklet både et konsept, testet konseptet og utviklet hovedfunksjonaliteten av konseptet på en så solid og gjennomtenkt måte som mulig. Dette gjaldt både kodemessig og konseptmessig. Kravspesifikasjonen ble utarbeidet i tro om at vi kom til å klare å svare til alle kravene. Etter relativt kort tid ble det klart at det å blant annet utvikle en spill-basert integrasjon i systemet (gamifye løsningen), kom til å bli i overkant tidkrevende. Dette gjorde at vi valgte å prioritere den grunnleggende funksjonaliteten, som i seg selv var et omfang som kunne bli krevende å utvikle innen den gitte tidsrammen.

## 4. Testing

Vi vil her beskrive den testingen som ble gjennomført i de ulike stadiene av prosjektet. I et utviklingsprosjekt er det ulike former for testing, delt i to hovedkategorier; funksjonell testing og ikke-funksjonell testing (JavaTPoint, u.d.). Den funksjonelle testingen går ut på å teste det rent tekniske ved systemet, mens den ikke-funksjonelle testingen avdekker prestasjon, brukbarhet og kompatibiliteten til systemet.

Den funksjonelle testingen foregår etter et hierarki; enkelkomponenter av koden testes først for å se at de har den funksjonaliteten som ønskes, for så å teste samspillet av ulike komponenter. Til slutt gjennomføres ende-til-ende-testing hvor hele systemet testes og forsikres om at alt fungerer sammen (Globalluxsoft, 2018).

Den ikke-funksjonelle testingen følger ikke det samme hierarkiet som funksjonell testing, men består av eksempelvis brukertesting, tilgjengelighetstesting og akseptansetesting.

#### 4.1 Testing under designsprint

##### Brukertester

I starten av prosjektet, i løpet av designsprinten, utførte vi fem sluttbrukertester på første utkast av prototypen til applikasjonen. Ved å gå ut ifra målene vi hadde satt for applikasjonen så langt, fikk vi stilt viktige spørsmål om funksjonalitet, brukerflyt og løsningen som konsept. Svarene vi fikk på disse spørsmålene hadde stor effekt på hvordan vi utviklet funksjonene i utviklingsfasen. Hovedmålet for disse testene var å få et klart bilde av hvilke funksjoner som ville vise seg å være vitale og hvilke som ville bli vanskelige eller unødvendige å implementere. Gjennom brukertester i designsprint fikk vi verdifulle tilbakemeldinger fra noen av de som ville bli brukere av Juvet. Tilbakemeldingene var i form av svar på forhåndsdefinerte spørsmål, definert av fasilitator, og basert på de antakelsene vi hadde utarbeidet tidligere i designsprinten. Alle designsprint-deltakerne utenom fasilitator hadde hver sitt ark, slik vi ser i figur 7 og 8, og hadde under brukertestene som oppgave å notere nyttige innspill eller sitater fra den aktuelle testen. Spørsmålene skulle svares på med «Yes» dersom de var tydelig at personen bekreftet spørsmålet gjennom dialog eller bruk av prototypen, «?» dersom det var uklart om testbrukeren svarte på spørsmålet eller «No» hvis testbrukeren var tydelig negativ til spørsmålet. Dette ble så brukt til å avslutningsvis danne en statistikk for alle testbrukerne ved å ta i bruk alle svarene fra designsprint-deltakerne.

Spørsmål	Skriv inn ditt navn til høyre:					Mats
	10:00	11:00	12:00	14:00	15:00	
	Testbruker 1	Testbruker 2	Testbruker 3	Testbruker 4	Testbruker 5	
Spørsmål vi vil ha svar på						
Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Virker brukeren interessert i faglig utvikling?
Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Vet brukeren at de kan reise på konferanse på jobbens regning, eller kjøpe seg faglige tjenster via jobb?
Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Skjønner brukeren at løsningen handler om kurs og aktiviteter?
?	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Vekker kurs-temaene interesse?
Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Finner brukeren søker?
No	✗ Yes	✗ ?	✗ ?	✗ Yes	✗ Yes	✓ Forstår brukeren hva tagsene i et kurs betyr?
No	✗ No	✗ ?	✗ ?	✗ Yes	✗ Yes	✓ Virker brukeren interessert i å samle "badges" ved å ta kurs?
Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Forstår brukeren hvordan søker fungerer?
Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✗ No	✓ Finner brukeren innboksen sin?
?	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✗ ?	✓ Oppleves tekst og bilder forståelig?
Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Skjønner brukeren at de kan kontakte fagpersoner med kompetanse på det de har søkt etter?
Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Virker det som brukeren ønsker å bruke Juvet for egen faglig utvikling?
No	✗ Yes	✓ Yes	✓ Yes	✗ ?	✗ ?	✓ Virker det som brukeren ønsker/vil/kommer ta kontakt med eksperter?
No	✗ Yes	✓ Yes	✓ Yes	✗ ?	✗ Yes	✓ Er de villige til å bidra med innhold (kurs, aktiviteter, etc) til i tjenesten?
Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Kan vi oppfordre brukere om å ta i bruk Juvet uten å virke masele?
No	✗ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Kan Juvet kan gjøre rekruttering enklere for Fremtid?
Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Kan vi få bedre oversikt over hvem som sitter på ulike kunskaper?
No	✗ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Kan vi spre relevant informasjon kontinuerlig uten å overvelde eller komplisere?
Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✓ Klarer vi å hjelpe Fremtidindire til å finne fagpersoner, kurs eller emner via Juvet?
No	✗ No	✗ No	✗ No	✗ Yes	✗ Yes	✓ Kan vi "gamifye" systemet slik at det blir mer attraktivt å delta regelmessig?

Figur 7 - Svartabell for spørsmål i brukertest av prototype

Et eksempel på tilbakemeldinger i form av innsikter ser vi i tilbakemeldingene fra testbruker 1 i figur 8 nedenfor. Her fikk vi blant annet innsikt om at testbrukeren forstår vår innboks-funksjon og hva som ligger bak denne. Det denne brukeren derimot ikke likte så godt var at de måtte gå videre med søker på ønsket tema (ønsket heller å trykke seg frem via flere kategori-grupperinger). Vi ser samtidig at denne brukeren ikke er helt sikker på hva som ligger i knappen «Lagre til senere». All denne innsikten ble vektet med hvor betydningsfull vi mente dette var, fra 1 til 3, hvor 1 er minst viktig og 3 er mest viktig.

Under kolonnen «Hvem har lagt inn» står navnet til den aktuelle designsprint-deltakeren som noterte innsikten underveis i brukertesten av testbruker 1. Tekst i kolonnen "Innsikt" markert i grønt vil representer en positiv tilbakemelding, mens markert i rødt vil være negativt, og fet skrift er et

sitat. I kolonnen «Skjerm nr» ble den aktuelle siden som testbrukeren var på under brukertesten notert.

Hjem har lagt inn	Brukere	Skjerm nr	Innsikt	Hvor betydningsful var innsikten på en skala fra 1-3
Adrian	Testbruker 1	Landingsside	Mye info på landing	2
Adrian	Testbruker 1	Landingsside	Liker populære	1
Adrian	Testbruker 1	Landingsside	Mulighet til å melde interesse mangler	3
Adrian	Testbruker 1	Landingsside	Koding kategori første klikk, forventet diverse kurs	2
Adrian	Testbruker 1	Landingsside	Trykke på kurs for mer informasjon er ikke tilstede	3
Adrian	Testbruker 1	Landingsside	Ville foretrykket å ikke gå videre via søker	2
Adrian	Testbruker 1	Resultatside	Forventet endring etter filter	1
Adrian	Testbruker 1	Resultatside	Mulighet for juks relatert til faktisk kompetanse	3
Adrian	Testbruker 1	Resultatside	Savner å kunne trykke på et kurs kort for mer info	3
Adrian	Testbruker 1	Resultatside	Er ikke helt sikker på lagre til senere	2
Adrian	Testbruker 1	Innboksen	Forstår innboksen	3
Adrian	Testbruker 1	Landingsside	Overskrift er litt lang (småplukk)	1
Adrian	Testbruker 1	Landingsside	Ser det er mer gruppert etter å ha sett resten	2

Figur 8 - Eksempel på tilbakemeldinger under brukertest. Hentet fra Google Sheets-dokumentet hvor all innsikt fra brukertester er lagret.

Fullstendig oversikt over dette brukertest-svararket kan sees i [vedlegg 3](#).

Denne brukertesten tok for seg kun de tre sidene av prototypen som ble laget under designsprinten, og vi fikk dermed ikke testet de andre sidene som vi laget i etterkant. Vi fikk derfor heller ikke testet de reviderte versjonene av de sidene vi presenterte i denne runden med brukertester.

Konsekvensen av dette var at vi ikke var trygge på om brukerne av løsningen oppfattet og brukte de endelige sidene med den intensjonen for bruken av dem på den måten vi tenkte. Selv om vi ikke fikk gjennomført flere brukertester under utviklingen, ble det gjennomført en akseptansetest ved prosjektperiodens slutt. Svarene fra akseptansetesten ble brukt til å avgjøre om løsningen svarte til kravspesifikasjonen.

Testing av prototypen, både som konsept og som designforslag resulterte i verdfull innsikt som vi brukte til å justere målene våre for den endelige applikasjonens utseende og funksjonaliteter.

#### 4.2 Testing underveis i utviklingen

##### JUnit5

Det ble gjennom store deler av utviklingen av backend tatt i bruk JUnit5 for å teste samsvar mellom klasser og metoder. Hovedsakelig ble det gjennomført JUnit5-tester for å finne ut i hvilken grad metodene vi skrev returnerte forventede resultater. En del av disse testene bestod også av å se om visse uriktige input- eller bruk av metoder ga riktige samsvarende feilmeldinger.

Et eksempel på vår bruk av JUnit5-testing er testen vi brukte for å finne ut om metoden 'getActivity', som skal hente et aktivitetsobjekt fra databasen, returnerte det vi forventet. I figur 9 nedenfor er koden for denne testen. Her blir testen først arrangert med 'mock' data, som er falsk data, og en forventet retourverdi før den tester med de valgte dataene. Til slutt verifiserer metoden at den riktige informasjonen returneres, og sender en feilmelding hvis dette ikke er tilfellet. I dette testeksempelet brukes 3 tilfeldige tall som mock-data. Med disse mock-verdiene unngikk vi å bruke ekte data fra databasen i testen, noe som ville ha gjort det mye mer tidkrevende å sette opp og utføre.

```

    @Test
    void getActivities_should_return_http_200_given_valid_activity_id() throws Exception {
        // arrange - train your mock
        given(service.getActivitiesPageCount(anyInt(), anyInt(), anyLong())).willReturn(generateResponse());

        // act & assert
        mockMvc.perform(get("/api/v1/activities/?page=1&count=2"))
            .andExpect(status().isOk());

        mockMvc.perform(get("/api/v1/activities/?page=2&count=2"))
            .andExpect(status().isOk());

        // verify that dependency is invoked
        verify(service, times(2)).getActivitiesPageCount(anyInt(), anyInt(), anyLong());
    }

```

Figur 9 - Kodeeksempel på en JUnit5 test fra Juvets prosjektkode

## MockMVC og Mockito

Flere av testene vi ville utføre på metoder, både i backend og frontend, viste seg å være utfordrende å fullføre uten samsvarende kodebiblioteker eller funksjonaliteter som enda ikke var implementert. For å unngå at dette skulle skape store forsinkelser i utviklingen av nye funksjoner tok vi i bruk Mockito og MockMVC. Med disse samsvarende verktøyene kunne vi teste våre metoder med Mockito sine etterligninger av funksjonaliteter som eventuelt manglet i vår kode. Noe som gjorde det enklere å fullføre testene med informerende resultater.

Et eksempel på hvordan vi brukte MockMVC og Mockito på aktivitetskontrolleren sees under. Her testet vi etter prinsippet «arrange, act and assert» også kalt AAA (Automation Panda, 2020). Det kan oversettes til «gitt – når – så», og kan leses som «gitt at vi har denne funksjonen, når dataene er slik og sånn, så skal testen returneres som feilet». En slik test kan ha to utfall; suksess eller feilet.

```

74 ►  public void createActivity_should_return_http_201_given_a_activity() throws Exception {
75
76     // arrange
77     ActivityRequest activity = generateTestActivityRequest();
78     given(service.createActivityByUser(any(), anyLong())).willReturn(activity.cast());
79
80     // act and assert
81
82     mockMvc.perform(post( urlTemplate: "/api/v1/activities")
83                     .contentType(MediaType.APPLICATION_JSON)
84                     .accept(MediaType.APPLICATION_JSON)
85                     .content(asJsonString(activity)))
86         .andExpect(status().isCreated());
87
88     // verify
89     verify(service, times( wantedNumberOfInvocations: 1)).createActivityByUser(any(), anyLong());
90
91 }

```

Figur 10 - MockMVC-testing i Juvets prosjektkode.

## Postman

Postman er et verktøy som lar oss teste API'er med å sende spørninger til endepunkter i API'et (Postman, u.d.). Det ble i hovedsak brukt med en lite strukturert ad-hoc-tilnærming for enhetstesting av API og under integrasjonstest mellom API'et og databasen. En integrasjonstest er en test som avdekker om to eller flere komponenter i et system kommuniserer med hverandre på ønsket måte (Hamilton, 2022). I figuren under kan et eksempel på en spørring mot et endepunkt i Juvets API sees:

The screenshot shows the Postman interface with a GET request to 'localhost:8080/api/v1/activities/14/card'. The 'Headers' tab is selected, showing a single header 'Content-Type' with the value 'application/json'. Other tabs like 'Params', 'Authorization', 'Body', 'Pre-request Script', 'Tests', and 'Settings' are visible at the top. A 'Send' button is at the top right.

Figur 11 - Eksempel på testing av et endepunkt i Juvet med Postman

En spørring mot et endepunkt vil resultere i en respons. Med mindre man har angitt en feil URL. Responsene inneholder ulike statuskoder ut ifra hvordan ressursen (API'et) klarer å tolke spørringen. Eksempelvis er statuskode 200 i en respons en beskjed om at API'et, ut ifra sitt perspektiv, forstod henvendelsen og responderte med det som endepunktet er definert til å utføre eller returnere (MDN Web Docs, u.d.).

Et fullt eksempel på testing i Postman med ulike typer responser og feilmeldinger kan sees i [vedlegg 9](#).

## Testing på frontend

I hovedsak er frontend-delen av løsningen bygget slik at all logikk er skrevet i React Hooks og det er disse og andre funksjoner som burde ha blitt nøyde enhetstestet. Disse enhetstestene ble nedprioritert i konkurransen med andre oppgaver for utviklingen. React-komponentene i seg selv er ganske "dumme" og trengte derfor kun enkel nettleser-testing, utover det å se at de oppførte seg på tenkt måte i en nettleser i henhold til responsivitet og struktur.

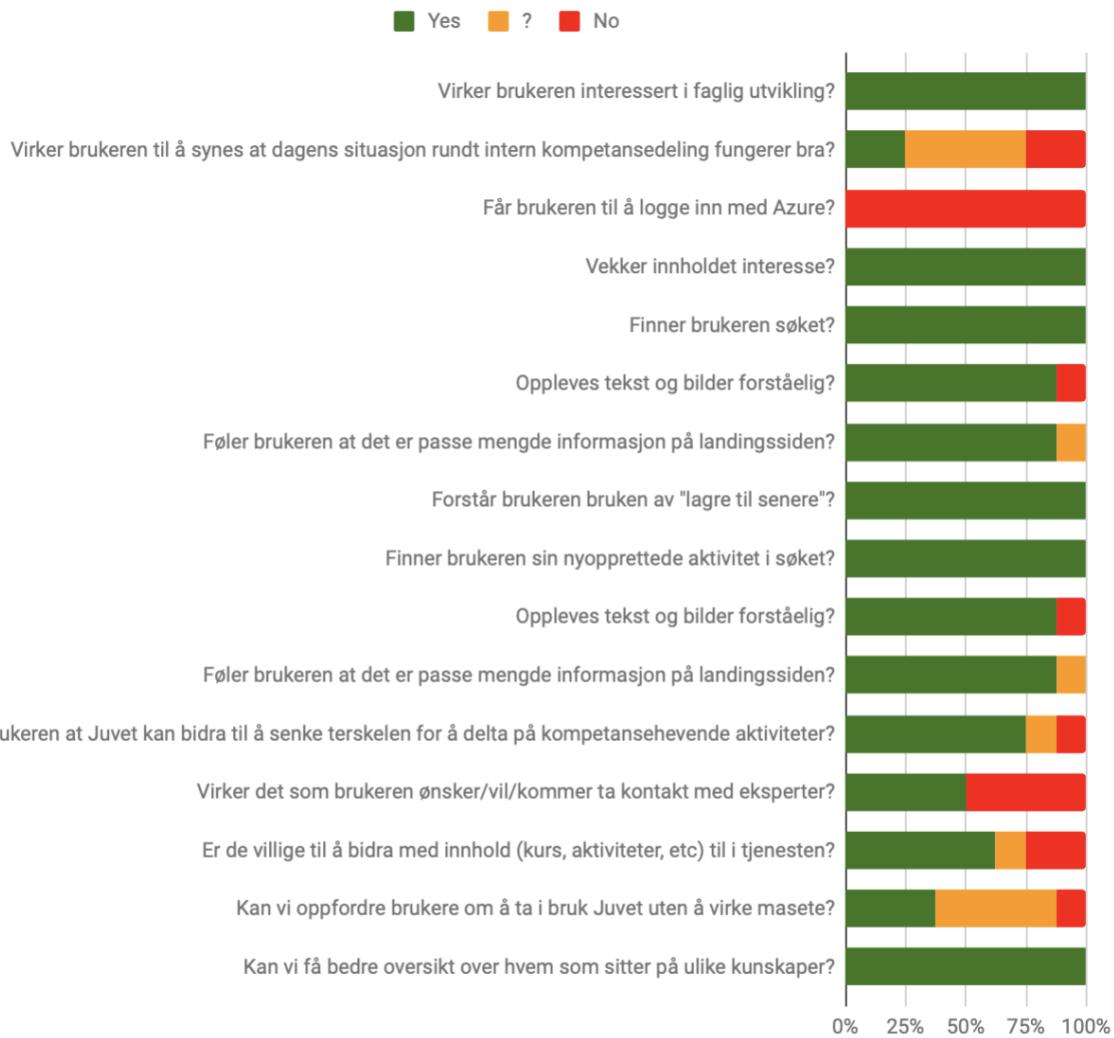
### 4.3 Akseptansetest

Ved slutten av sprint 9 ble en akseptansetest av løsningen gjennomført. På grunn av at ikke hele løsningen ble ferdig implementert, ble hovedflyten og opprettelse av en aktivitet testet. På samme måte som under testing av prototypen, besto hovedflyten av å gå inn på landingssiden, for så å trykke seg videre inn på en aktivitet og se detalj-siden til aktiviteten. På den detaljerte aktivitetssiden fikk testbrukeren mulighet til å melde seg på en aktivitet (trykke på «meld på»-knapp) og å lagre aktiviteten til senere ved å trykke på «Lagre til senere»-knappen. Deretter søkte testbrukeren på valgt emne innen IT-utvikling, som «java», og fikk opp søkeresultatsiden. Etter å ha gjennomgått søkeresultatet ble testbrukeren spurta om å kunne opprette en ny aktivitet i Juvet. Opprettelsen av aktivitet var en del av løsningen som ikke hadde blitt brukertestet tidligere, men som vi testet under akseptansetesten.

Akseptansetesten ble gjennomført på 4 ulike personer med tittel utvikler hos digital-avdelingen i Fremtind. Testene ble gjennomført med fysisk møte på et møterom i Fremtinds lokaler. I motsetning

til testen under designsprint, ble ikke akseptanseteten gjennomført av fasilitator fra Fremtind, men vi måtte planlegge og gjennomføre hele testen selv. Testen ble gjennomført med utgangspunkt i det samme oppsettet som test av prototypen. Spørsmålene i testskjema var rettet mot kravspesifikasjonen som var utarbeidet i starten av prosjektet. Innlogging på løsningen med AzureAD var tenkt å testes i akseptansetesten, men på grunn av problemer i forkant av testene ble dette tatt ut av testflyten. Likevel ble det stående som et punkt blant testspørsmålene.

Svarprosent på testspørsmålene vises i figuren under.



Figur 12 - Akseptansetest svarprosent på testspørsmålene

Oppsummert viser svarprosentene et signal om at konseptet og løsningen vil gi verdi for bedriften dersom den blir satt i produksjon for fullt. Under gjennomgang av swarene med prosjektgiver, var også vår kontaktperson enig i dette. De to mest interessante funnene i akseptansetesten var svarprosenten for spørsmålet «Føler brukeren at Juvet kan bidra til å senke terskelen for å delta på kompetansehevende aktiviteter?», hvor 75% mente at løsningen kan senke terskelen for å delta på kompetansehevende aktiviteter. Den siste 25% var enten ikke tydelig på om han/hun mente dette eller mente at det ikke ville bidra til senket terskel. Og det andre mest interessante funnet var at over halvparten (62,5%) var villige til å bidra med innhold til denne løsningen. Til sammen utgjør dette et godt utgangspunkt for levedyktigheten for Juvet som konsept og løsning i Fremtind.

Punktet med innlogging i AzureAD er det eneste punktet som feiler helt, og årsaken til dette var at det ikke ble inkludert i testflyten. Innloggingsfunksjonaliteten ble fikset i etterkant av akseptansetesten og deretter testet av gruppa.

#### 4.4 Tilgjengelighetstest

Som også nevnt i [delkapittel 5.2.5](#) om utvikling av design og tilgjengelighet, ble tilgjengelighetstester gjennomført. For å få en score på hvor godt løsningen er utviklet med tanke på universell utforming, ble den testet i en tilgjengelighetstest, Siteimprove.com (Siteimprove, u.d.). Resultatet av tilgjengelighetstesten av landingssiden finnes i [vedlegg 5](#). Den tar for seg «WCAG-standarden» og undersøker de kravene den stiller til universell utforming av nettsider.

#### 4.5 Samsvar mellom reell testdekning og ideell testdekning

Ideelt sett skulle testdekningen ha vært betydelig høyere enn det vi har implementert i løsningen. Testdekning viser prosentvis andel av koden i prosjektet som blir testet. Ved 100% testdekning blir alle mulige utfall i systemet testet. Fremtind har et krav om for sine systemer å ha en testdekning på minst 80%, men det ble tydelig bemerket av oppdragsgiver fra start at dette ikke var et krav for oss.

Reell testdekning i løsningen var for controllerene på backend er 23%, som er lavt. Dette er kun det prosentvise antallet linjer kode som er blitt omfattet i den strukturerte enhetstesting. I figur 13 kan en oversikt over testdekningen sees. Den mindre strukturerte tesingen, som f.eks. integrasjonstesting med Postman, ble hyppig gjennomført i mange ulike former under hele utviklingen, uten at det er mulig å tallfeste dekningen av denne testingen.

Element ▲	Class, %	Method, %	Line, %
▼  com.fremtind.juvet.application.controller	100% (5/5)	33% (15/45)	23% (85/363)
ActivityController	100% (1/1)	53% (8/15)	43% (54/125)
TagController	100% (1/1)	25% (1/4)	11% (3/27)
UserController	100% (1/1)	18% (2/11)	10% (10/94)
UsersController	100% (1/1)	42% (3/7)	29% (15/51)
ValidationController	100% (1/1)	12% (1/8)	4% (3/66)

Figur 13 - Testdekning av backend i Juvets prosjektkode. Totalverdier er rammet inn av gul innramming i figuren.

### 5. Produktdokumentasjon

I produktdokumentasjonen vil programmets oppbygging og virkemåte beskrives i detalj.

#### 5.1 Kravspesifikasjon

En kravspesifikasjon er et sett med oppgaver som oppdragsgiver ønsker at et system skal løse (Rolstadås & Liseter, 2018). Innen smidig utvikling angis disse gjerne som brukerhistorier, som beskriver hva en bruker av systemet ønsker å oppnå ved bruk av løsningen. Andre typer krav kan være forventinger til for eksempel brukervennlighet og/eller ytelse.

Videre deles kravene opp i funksjonelle og ikke-funksjonelle krav. De funksjonelle kravene definerer hvilke tekniske problemer systemet skal løse, mens de ikke-funksjonelle kravene definerer andre aspekter ved systemet, som blant annet design, ytelse, brukervennlighet og dokumentasjon.

### Utgangspunkt

Den overordnede kravspesifikasjonen har fra start vært å utvikle en MVP. Kravspesifikasjonen for MVP'en ble definert i form av spørsmål i startfasen av designsprinten og inneholdt disse momentene:

- Gi en søkefunksjon for å søke i tidligere aktiviteter og på tvers av personer/emner/kurs
- Sørge for at personer som ikke kunne delta på aktivitet også får et læringsutbytte
- Skape en lav terskel for å delta på aktiviteter
- Nå de riktige personene for ulike aktiviteter/faggrupper
- Kunne spre relevant informasjon kontinuerlig uten å overvelde eller komplisere
- Få en bedre oversikt over hvem som sitter på ulike kunnskaper
- «Gamifye» systemet slik at det blir mer attraktivt å delta regelmessig (minst viktig)

Sprintspørsmålene ble omformulert til kravlister, fordelt på funksjonelle krav og ikke-funksjonelle krav. Viktigheten av kravene har vi bestemt i måten de blir formulert på; *skal/bør/kan*. «Skal» er et krav som fikk høy prioritet og er ansett som en del av kjernefunksjonaliteten i løsningen. «Bør» er et krav som fikk middels prioritet og «kan» fikk lav prioritet. De kravene med lavest prioritet var de kravene som vi valgte å prioritere bort da vi så at vi ikke fikk tid til å implementere alle kravene i løsningen.

Tabell 1 - Funksjonelle krav til MVP

### Funksjonelle krav

Som bruker...	Viktighet
skal løsningen være tilgjengelig for internt ansatte i Fremtind	Middels
skal jeg kunne søke på innhold på tvers av personer/emner/aktiviteter	Høy
skal jeg kunne opprette nye aktiviteter i løsningen	Høy
skal jeg kunne kontakte andre i organisasjonen basert på mitt søker på kompetanse	Høy
skal jeg kunne logge inn med «single sign on»	Middels

Som oppdragsgiver ønsker Fremtind at løsningen...	Viktighet
bør være «hostet» på AWS	Middels

Tabell 2 - Ikke-funksjonelle krav til MVP

### Ikke-funksjonelle krav

Som bruker...	Viktighet
bør jeg få relevant informasjon uten å bli overveldet av informasjon	Høy

skal det være enkelt å legge aktiviteter i «lagre til senere»	Høy
skal det være intuitivt å finne hvor mine aktiviteter er lagret	Høy
skal føle en lav terskel for å delta på aktiviteter	Middels

Som oppdragsgiver ønsker Fremtind at løsningen...	Viktighet
kan «gamifye» deltagelse på aktiviteter	Lav
skal hente brukerinfo fra AzureAD for å opprette bruker i Juvet	Høy
kan sørge for at ansatte som ikke fikk tid til å være med på aktiviteten også kan få læringsutbytte	Lav

Kravspesifikasjonen var tydelig på hva vi skulle legge mest vekt på i utviklingen av design og implementering. Særlig var dette med en søkefunksjon som kunne søke på tvers av alle typer aktiviteter og personer viktig. Siste punkt som omhandlet å «gamifye» løsningen, som vil si å benytte spillteori og belønning etter deltagelse på aktiviteter for å gjøre det mer attraktivt å være en aktiv bruker av Juvet. Dette fikk vi tidlig varierende og tydelige tilbakemeldinger på via brukertester og konkluderte med at det måtte utvikles «på riktig måte» for at det skulle gjøre sin hensikt. Med «på riktig måte» menes at det er varierende hvor mottakelig potensielle brukere av Juvet er for denne funksjonaliteten. Én tilbakemelding handlet om at brukeren ikke kunne vite om andre deltok eller fullførte aktiviteter kun for å sanke poeng/premier. Dette var begrunnet med at løsningen ikke la opp til noe slags kontroll på om brukeren faktisk hadde deltatt på aktiviteten eller bare hadde jukset. Andre brukere var veldig positive til denne funksjonaliteten, da de så muligheten til å sammenligne seg med andre og se sin prosgresjon. Samtidig var det ulikt hvor mye dette potensielt kunne ha påvirket brukerens motivasjon for å delta basert på innsikt fra brukertestene. Derfor ble dette «kravet» minst prioritert.

Et moment som ikke ble nevnt i utarbeidelsen av kravspesifikasjonen, men som likevel ble oppfattet som viktig for oppdragsgiver, var å designe løsningen basert på deres eget designsystem. Ved å ta i bruk designsystemet, ville løsningen fremstå i Fremtind-drakt og lettere kunne identifiseres med organisasjonen. Designsystemet ble presentert av kontaktpersonen i Fremtind allerede under første møte, hvor han gjennomgikk mulighetene for bacheloroppgaver i bedriften. Dette la en tydelig føring for at vi gjerne kunne benytte designsystemet i vår løsning.

Krav til dokumentasjon fra oppdragsgiver var fra start ikke noe tema, men ble senere i prosjektet bestemt til å være en oversikt over API’et og en enkel brukerveiledning for løsningen. Siden løsningen skal være en nettside med kun én type brukertilgang, vil fremgangsmåter være intuitive og ligne på tilsvarende løsninger for oppslagsverk (eks. YouTube ol.), som mange er kjent med fra før. Samtidig har oppdragsgiver bidratt til utforming av løsningen og har derfor erfaring med hvordan bruksflyten i løsningen er lagt opp. Kodeflyten i løsningen, både for frontend og backend, er oversiktlig da vi har etterstrebet å strukturere prosjektkoden etter «bestepraksis».

### Endringer underveis

Det ble ikke gjort store endringer i kravspesifikasjonen underveis, men den minst viktige funksjonaliteten ble bortprioritert på grunn av at vi innledningsvis så at den ville kreve for mye ressurser gjennom utvikling og testing. Funksjonaliteten som ble bortprioritert var det å «gamifye» løsningen. *Gamifying* av løsningen var blant annet et av få punkter i brukertesten under

designsprinten som fikk noen negative tilbakemeldinger, og på den måten var det ikke vanskelig å bortprioriteres.

## 5.2 Beskrivelse av løsning

Vår løsning er splittet opp i to separate applikasjoner. I dette kapitlet skal vi gjøre rede for implementasjonen av de to applikasjonene. Det har i løpet av utviklingen blitt brukt støttebiblioteker for både frontend og backend som henvises til i kildekoden. I frontend er disse definert i en 'package.json'-fil, mens de i backend defineres i en 'pom.xml'-fil.

### 5.2.1 Arkitektur

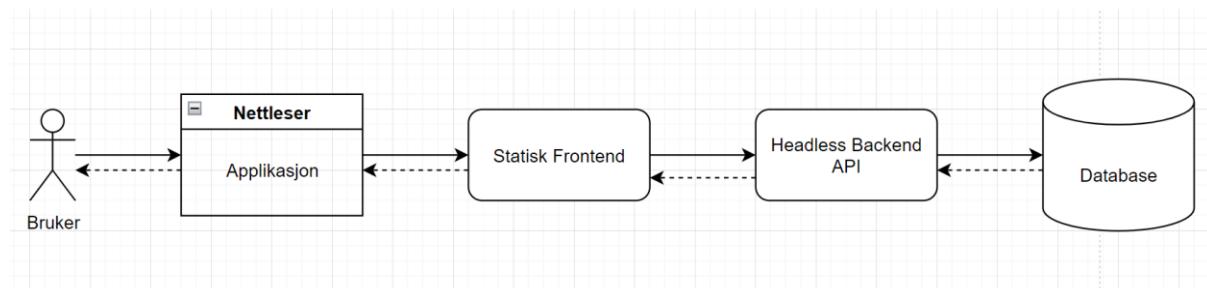
Løsningen er satt sammen av 2 selvstendige applikasjoner og en database. En statisk frontend og en backend-API der frontenden er avhengig av backenden og ikke motsatt. Vi ender da opp med en *Headless* backend (et API som ikke har noen mening om hvordan dataen skal presenteres). Dette er til forskjell fra den mest tradisjonelle, eldre måten en webapplikasjon er satt opp på hvor backenden har ansvar for å servere frontenden til brukeren (Heslop, 2020). I stedet lever frontenden på en annen server. Når frontenden kjøres i nettleseren vil den dynamisk hente data fra backenden ettersom den trenger.

Fordelen med dette er at vi øker separasjon av bekymringer, som er bokstaven «S» beskrevet i akronymet SOLID, som vi har brukt som retningslinje for utviklingen (Oloruntoba, 2020). Det gjør både at frontend og backend kan jobbes på selvstendig uten omfattende kommunikasjon, foruten en definert API.

Av erfaring kan en headless backend kreve mer konfigurasjon. For eksempel om backenden og frontenden serveres på forskjellige domener vil nettlesere forhindre frontenden fra å snakke med backenden uten konfigurasjon av CORS (Cross Origin Resource Sharing). Nettlesere blokker API-responser som går på tvers av domener om ikke API'et spesifikt tillater det domene (Polo, 2020). I vår situasjon må vi derfor tillate frontend domenet vårt på backenden. Dette er av sikkerhetsmessige årsaker. Erfaringsmessig vil en headless backend også ofte bety mer kompleksitet ved deployering da man ender opp med en ekstra applikasjon å ta hensyn til. Selv om det kan være noe mer konfigurasjon i starten utveier fordelene ulempene i veldig stor grad når det kommer til ryddighet og muligheter for videre utvikling.

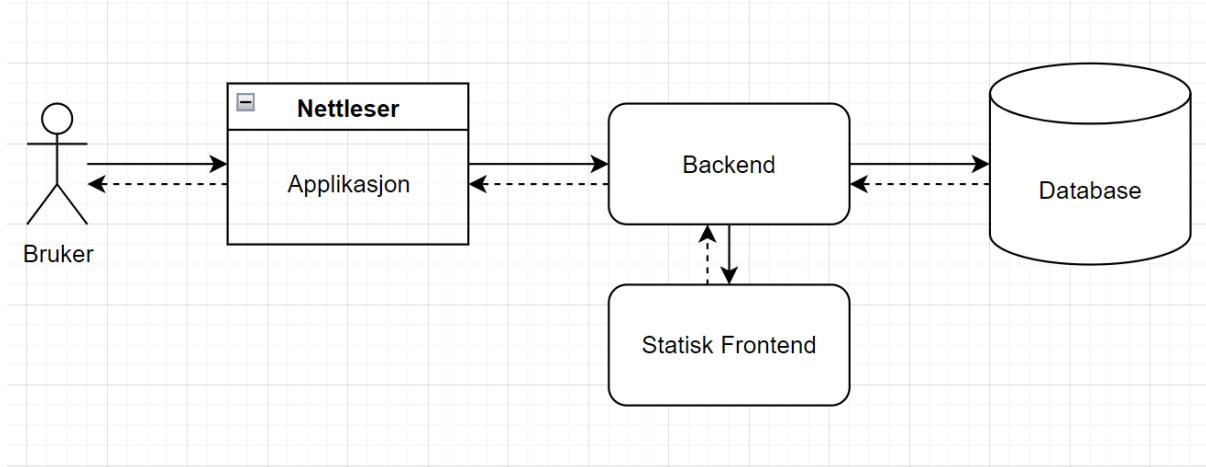
Se de to figurene under for illustrasjon av en headless og en tradisjonell backend:

#### Eksempel med headless backend:



Figur 14 - Ressurskjeden i et headless backend oppsett - slik Juvet er utviklet.

## **Eksempel med tradisjonell backend:**



Figur 15 - Ressurskjeden i et tradisjonelt backend oppsett (ikke headless)

## 5.2.2 Frontend

Frontend applikasjonen er en «SPA» («Single Page Application»)/Enkeltside applikasjon). Dette er det React (som vi bruker som hovedbibliotek) er bygd til å lage ut av boksen (React, u.d.). Det betyr at hele applikasjonen lever på én og samme html-side.

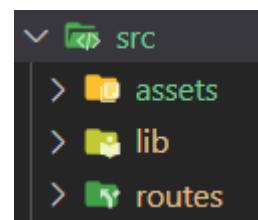
Fordelene med en SPA er at applikasjonen føles helhetlig ut og rask når man navigerer ettersom nettleseren ikke trenger å laste inn data på nytt. Dette gjelder selv om du navigerer frem og tilbake i applikasjonen. Hvis du navigerer på siden vil React kun oppdatere det som er nødvendig. En navigasjonsbar som er lik på begge sidene du navigerer mellom vil for eksempel ikke bli rekonstruert som ved tradisjonell html.

Ulempene med SPA-sider er at de ofte tar lenger tid å laste inn når brukeren først ankommer siden, i hvert fall om «code splitting» (inkrementell innlasting av kildekode) ikke gjøres (Baranchuk, 2022). Dette skjer fordi nettleseren laster inn all koden for å vise alle sidene på applikasjonen med en gang, selv om bare forsiden vises. En annen ulempe med SPA-sider er at de er avhengige av JavaScript for å fungere. Dette påvirker anslagsvis 1% av alle nettsidens brukere, som ikke kan kjøre JavaScript av ulike grunner (Silver & Adam, 2019).

Disse ulempene kan overkommes ved bruk av rammeverk som for eksempel Next.js (Vercel, u.d.). Dette var opp til diskusjon i gruppa, men ble nedstemt grunnet at det ville ta tid å sette opp. Tid som vi ikke hadde til rådighet under en allerede stram tidslinje.

### 5.2.2.1 Arkitektur

Kildekoden i frontenden er separert inn i 3 lag. De 3 lagene er «assets», «lib» og «routes». «Assets» er gjenbruksbare kodebiter og grafiske elementer som ikke nødvendigvis er spesifikke til applikasjonen. «Lib» er gjenbruksbare kodebiter og komponenter som er spesifikt til domenet Juvet. Og «routes»-laget har ansvar for applikasjonsspesifikk ruting og oppbygning av delene av applikasjonen.



Figur 16 - Mappestruktur i kildekoden for Juvet

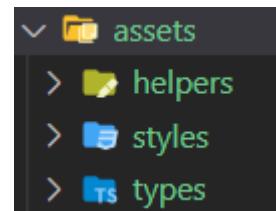
## Assets

«Assets»-laget består av 3 deler. Her ligger «helpers» (hjelbere), «styles» (stilark) og «types» (typer).

Hjelbere er små gjenbruksbare kodebiter som ikke nødvendigvis er spesifikke for applikasjonen, men som heller ikke er eksterne avhengigheter. Vi har veldig få hjelbere, da det av erfaring generelt er bedre å gå for eksterne verktøy som er jevnlig opprettholdt og nøye testet.

Stilarkene i «styles» er generelle stilark som like gjerne kunne være brukt i andre applikasjoner. Som for eksempel standard fontstørrelse eller bakgrunnsfarge.

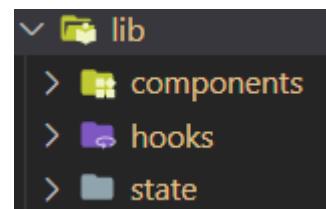
Typer er datamodeller for TypeScript. I types lever globale gjenbruksbare typer som ikke er spesifikke til løsningen, som for eksempel en Adresse-type.



Figur 17 - Mappestruktur i assets-laget for kildekoden i Juvets frontend

## Biblioteket (lib)

Biblioteket består igjen av 3 forskjellige lag som illustrert i figur 18. Vi har brukt samlokaliseringssprinsippet («colocation principle») i utviklingen av frontenden (filer som har med hverandre å gjøre skal leve nærmere hverandre i filsystemet) (Barral, 2019). Det betyr at det for eksempel kan ligge noen «hooks» sammen med en komponent i «components»-laget om den hooken er spesifikk til kun den komponenten.



Figur 18 - Mappestruktur i bibliotek-laget for Juvets frontend

Tabell 3 - Beskrivelse av laginndelingen for frontend i Juvet

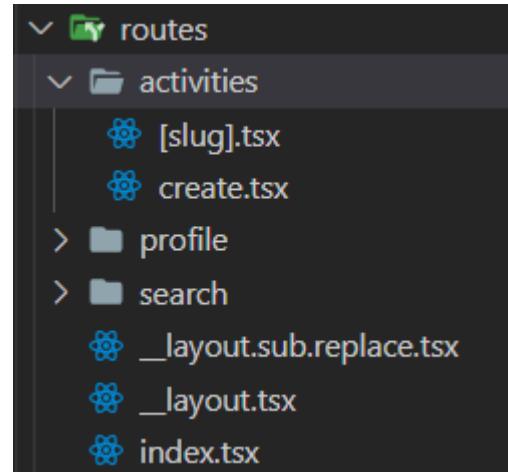
Lag	Beskrivelse
<b>State</b>	Grensesnitt for lokal og ekstern asynkron lagring ligger her. «State»-laget er avhengig av «types» og «helpers», men typene som er viktig for datamodellene i applikasjonene finnes direkte i state-mappen basert på samlokaliseringssprinsippet.
<b>Hooks</b>	De fleste React hooks, som tidligere beskrevet under «Logikk i tilpassede hooks» i <a href="#">delkapittel 3.2.2.5</a> lever her. Hooks blir ofte anvendt til å hente og formtere data fra state til komponentene, men det er også andre funksjonaliteter hooks hjelper med. Hooks-laget er avhengig av state.
<b>Components</b>	Komponent-laget har er avhengig av hooks, assets og types. Laget består av gjenbruksbare React komponenter spesifikke til løsningen. Vi bruker også komponenter fra Fremtids designsystem «Jøkul». De komponentene er bufret under «node_modules» som ligger i rot mappen av prosjektet. Verktøyet Yarn, som nevnt tidligere om verktøy brukt i prosessen under <a href="#">delkapittel 2.2</a> , holder styr på de. Components-laget er avhengig av hooks.

## Routes

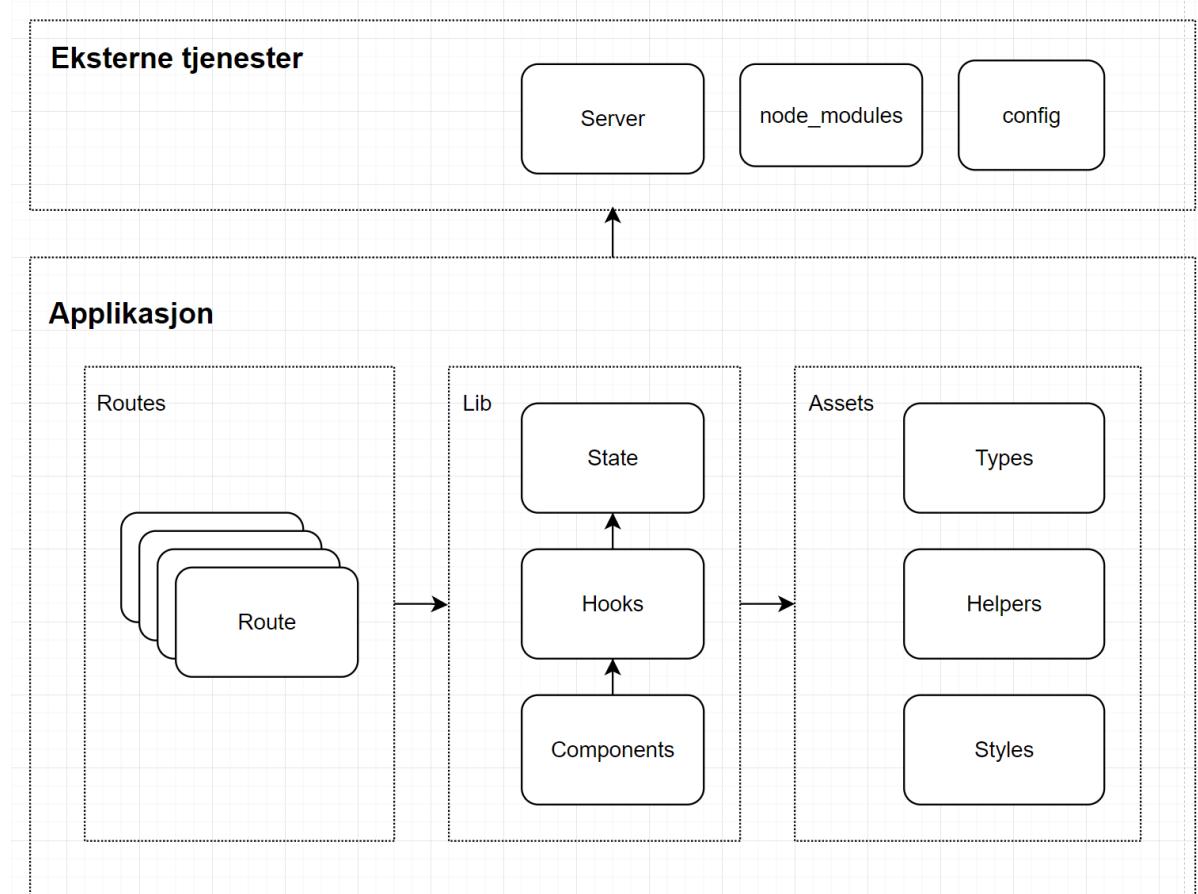
«Routes» er den andre delen av frontenden og er kun avhengig av «lib». Routes har ansvar for å

håndtere hva som skal vises på skjermen gitt hvilken rute som er skrevet i adressefeltet i nettleseren, og hvordan hver av de sidene er bygd opp. Routes-mappen er strukturert for å emulere filesystem-basert ruting. Det er når et rammeverk automatisk setter opp rutene på nettsiden basert på hvordan filesystemet er organisert (Farooq, 2021). Vi kunne implementert filesystem-basert ruting om vi hadde valgt å bruke Next.js som omtalt i [delkapittel 5.2.2](#) om frontend, men vi valgte uansett å emulere strukturen fordi den blir mer oversiktlig.

I figuren ved siden av ser du et eksempel på hvordan vi har strukturert mappene. Nederst ser vi «*index.tsx*». Den er inngangspunktet for nettsiden ettersom den ligger i roten av routes-mappa. Inne i activities-mappa kan du se en fil som heter «*create.tsx*». Hvis vi oversetter den filen til en filbane i filesystemet på maskinen vil den bli «*/activities/create.tsx*» og i nettleseren vår vil du kunne finne denne siden hvis du går til «*/activities/create*». Med unntak av filendingen vil alltid siden leve på samme sted i kildekoden som på nettsiden. Dette er vanlig for tradisjonelle sider hvor det lastes inn én html-fil på en bestemt plassering, men er kun en visuell fordel når det kommer til SPA'er, som forklares i starten av [delkapittel 5.2.2](#).



Figur 19 - Mappestruktur i routes-laget fra Juvets frontend



Figur 20 - Diagram over frontend applikasjonens avhengigheter internt og eksternt i Juvet.

### 5.2.2.2 Applikasjonstilstand

Hovedoppgaven til enhver frontend er å behandle data og presentere det til en bruker (Berkeley Extension, u.d.) - men data kan endre, og da er det opp til applikasjonen å oppdatere presentasjonen til å reflektere nye data. Dette kalles applikasjonens tilstand. Siden tilstanden er avhengig av data og tilstanden er applikasjonens viktigste avhengighet er det viktig å ha et godt system rundt det.

Data kan leve mange plasser; i minnet i nettleseren, lagret på brukerens maskin, på serveren eller en ekstern API. Siden frontend ikke kan få tak i all data umiddelbart, som ved å hente data fra en API over internett, har vi valgt å behandle all data som asynkront. Det vil si at selv lokale data blir behandlet som om det kan ta lang tid før de kommer frem. Det lar oss behandle all data likt over hele applikasjonen, som betyr at frontend spør om *noe* (data) og så må reagere (behandle dataen) når dataen ankommer en gang i fremtiden (Jofche, 2020).

### React Query

Vi har valgt å bruke React Query som et hjelpeverktøy for å holde på og oppdatere applikasjonens tilstand. React Query er et tilstandsbibliotek samt et buffer (React Query, u.d.). Det vil si at vi kan bruke React Query til å holde på tilstanden til applikasjonen, samt bufre data slik at frontend ikke trenger å hente fra kilden hver gang den spør.

La oss si vi har to komponenter på siden som begge viser noe data om brukeren som er logget inn. Den ene viser navnet, den andre viser epostadressen. Begge disse to tingene kan frontend få fra serveren hvis den spør, men de kommer alltid i et samlet objekt med både navn og epost. Hadde vi ikke tenkt noe over det, ville begge komponentene gått til serveren og hentet brukerinformasjonen, og deretter vist frem navn eller epost avhengig av komponenten. Med React Query kan frontend fra begge komponentene spørre om den samme dataen, men React Query vil kun spørre serveren én gang. Den andre gangen ligger det bufret rett i minne uten at vi trengte å tenke over det i noen av komponentene.

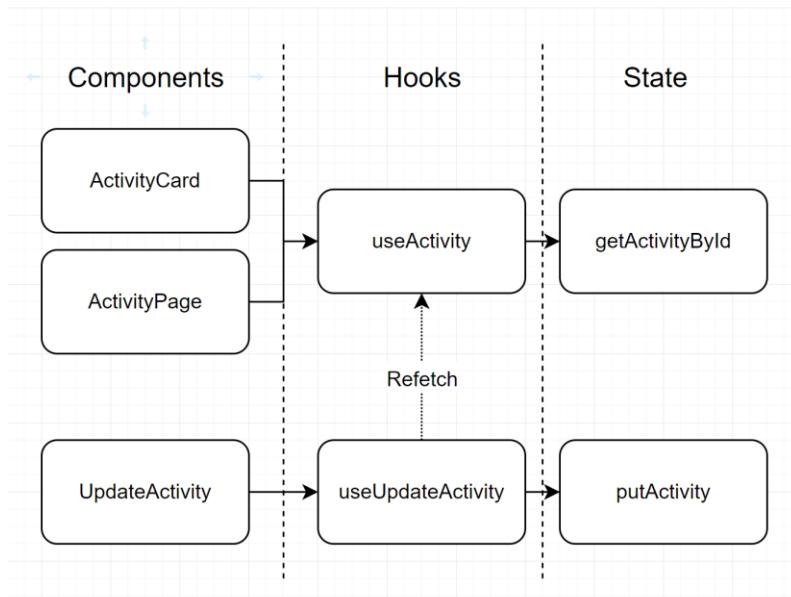
Utfordringen med dette, er at vi med bufret data kan fort ende opp med utdatert data. Siden vi ikke lenger henter dataen rett fra kilden vet vi ikke om den har endret. Vi kan forhindre dette med å fortelle React Query at den må laste inn noe på nytt hvis vi oppdaterer noe. Dette kalles mutasjoner i React Query (Tanstack.com, u.d.). Hvis vi bruker React Query til å oppdatere epost-adressen til brukeren, kan vi samtidig fortelle at brukerdataen som er bufret er utdatert og burde oppdateres. Da vil React Query oppdatere dataene i bakgrunnen og komponentene vil få et nytt varsel om at det har kommet noe data som de må vise frem. Slik opprettholder vi en fersk tilstand i applikasjonen.

### «Tilstandskjeden»

Vi har en kjede fra komponent til data i applikasjonen. Vi kaller denne for «*tilstandskjeden*». Den begynner med at en komponent trenger en type data. Vi lager derfor en React hook i hooks-laget som returnerer ønsket data etter påkalling fra en komponent. I React-hook'en har vi business-spesifikk logikk som formaterer og presenterer dataen den får tilbake fra React Query sin useQuery-hook. useQuery påkaller så datagrensesnittet i state-laget for å hente data fra hvor enn det befinner seg.

I figuren under ser vi et eksempel fra applikasjonen hvor vi har et aktivitetskort og en aktivitetsside som begge ønsker å vise en aktivitet. De implementerer derfor «useActivity»-hook'en som påkaller datagrensesnittet i state-laget. Vi ser også en komponent som oppdaterer en aktivitet. Når brukere

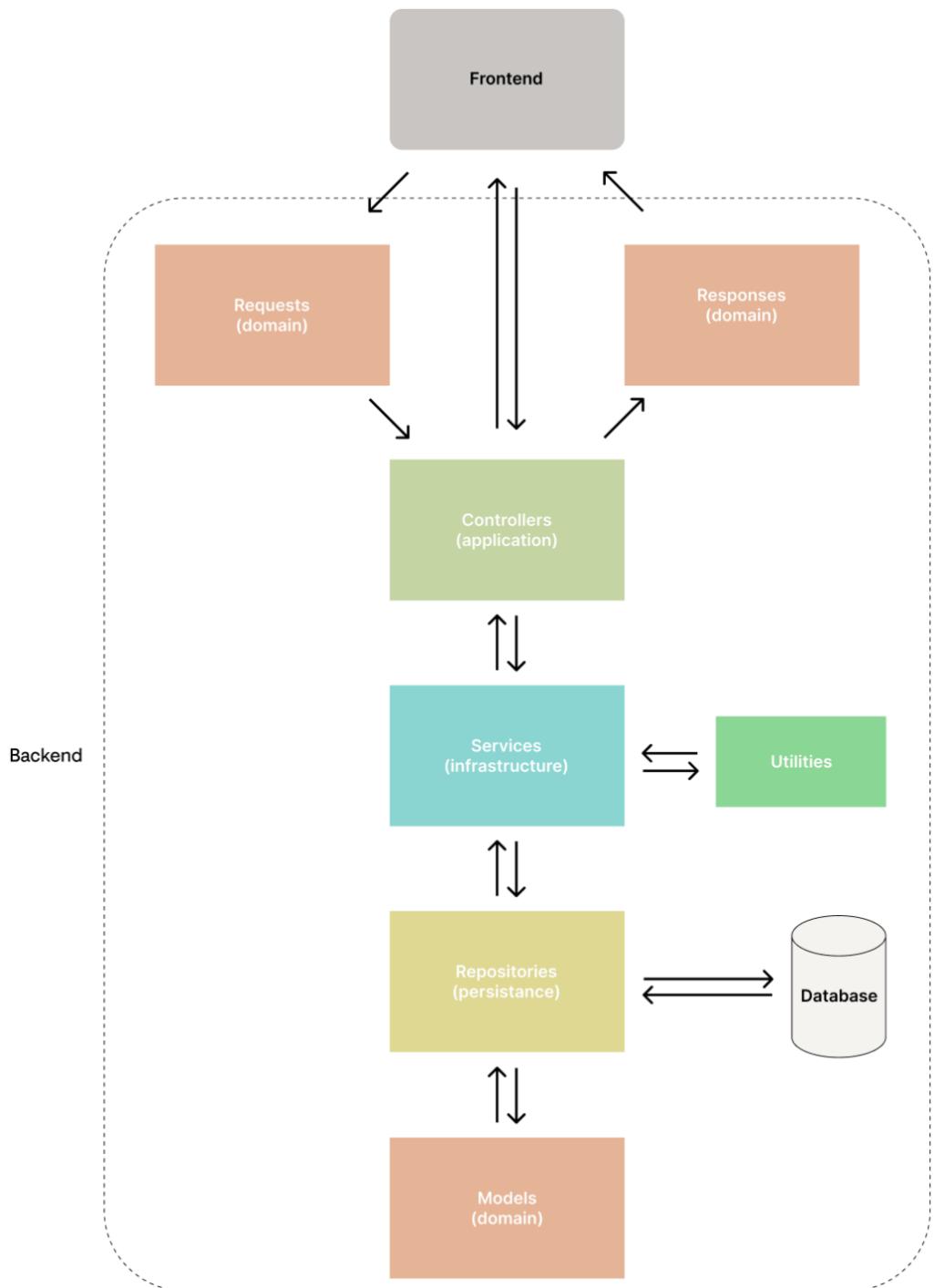
endrer på en aktivitet vil «useUpdateActivity»-hook'en fortelle useActivity i bakgrunnen at den burde oppdatere for å holde tilstanden fersk.



Figur 21 - Diagram som forklarer tilstandskjeden i applikasjonen

### 5.2.3 Backend

Backend-applikasjonen, som er en server/tjener, har i oppgave å motta, sende, bearbeide, analysere og lagre data. I tillegg til dette definerer den API'et, som er grensesnittet for å kunne kommunisere med den. Vår backend er delt opp i flere lag som har egne ansvarsoppgaver. Disse lagene består av contollere, services, repositories og models i domain. Illustrasjonen nedenfor viser denne inndelingen:



Figur 22 - Laginndelingen av Juvets backend

### 5.2.3.1 Domain

Domain-laget består av modeller som applikasjonen bruker til å lagre data. Disse modellene er representert ved vanlige POJO klasser. POJO står for «Plain Old Java Objects», som er helt standard javaklasser med attributter, konstruktør og metoder for å endre og få tak i attributtene verdi (JavaTPoint, 2021). Vi har videre strukturert de ulike modellene inn i pakker/mapper som beskriver hvilken nytte de har. Videre blir de ulike modellene og hvilke bruksområder de har forklart.

Modellene som representerer dataene som er lagret i databasen kalles entiteter (Oracle, u.d.). Vi har brukt Java Persistence API'et, som er Java sin ORM-standard for å lagre, hente og administrere Java-objekter i en relasjonsdatabase (Tyson, 2019).

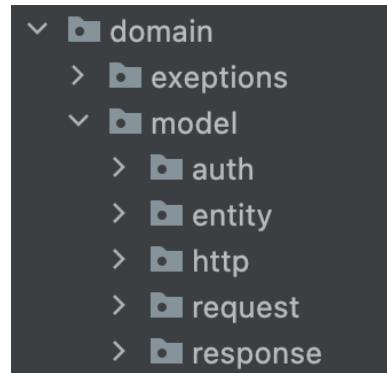
Ved å definere de modellene som representerer de ulike entitetene vi ønsker å lagre i databasen, sørger JPA for å frembringe databasetabellene med relasjoner mellom dem og gir mulighet til å utføre SQL spørninger mot disse dataene.

For å lagre den dataen vi behøvde i vår applikasjon trengte vi syv entitet-modeller. De 7 modellene er: activity, address, attachement, location, thumbnail, tag og user. Activity-entiteten lagrer data om aktiviteter som tittel, beskrivelser og tidspunkt. For å lagre informasjon om en bruker, så har vi en user-entitet. Ettersom vi har implementert innloggingen med Microsoft Azure AD så unngår vi å lagre sensitiv data som passord. Entiteten tag har to attributter; type og value. Denne entiteten brukes til å lagre interesser og kunnskap på en bruker og for å lagre kort hva en aktivitet omhandler. Ved å bruke tag-entiteten til å lagre interesse og kunnskap på en bruker, kan vi enkelt sammenligne og søke på aktiviteter basert på interesser som brukeren har lagret om seg selv. Relasjonene mellom de ulike entitetene blir illustrert og beskrevet i [delkapittel 5.4.4](#). Under er det et bilde som viser klassen for entiteten tag:

```

14     @Entity
15     @Getter
16     @Setter
17     public class Tag {
18         1 usage
19         @Id
20         @Column(nullable = false, name = "tag_id")
21         @GeneratedValue(strategy = GenerationType.AUTO)
22         private Long id;
23
24         2 usages
25         @NotNull(message = "Type kan ikke være tom")
26         private String type;
27
28         2 usages
29         @NotNull(message = "Verdi kan ikke være tom")
30         private String value;
31
32         @JsonIgnore
33         @ManyToMany(mappedBy = "tags", cascade = CascadeType.PERSIST, fetch = FetchType.EAGER)
34         private Set<Activity> activities;
35
36         @JsonIgnore
37         @ManyToMany(mappedBy = "competences", cascade = CascadeType.PERSIST, fetch = FetchType.EAGER)
38         private Set<User> userCompetences;
39
40         @JsonIgnore
41         @ManyToMany(mappedBy = "intrests", cascade = CascadeType.PERSIST, fetch = FetchType.EAGER)
42         private Set<User> userIntrests;

```



Figur 23 – Struktur på backend for modeller i «domain» – fra Juvets prosjektkode

Figur 24 - Entitetsmodellen for klassen Tag på backend som viser relasjonene den har med entitetene user og activity – fra Juvets prosjektkode

Vi har definert egne modellklasser for hvordan data blir sendt og returnert til API'et. Modellene som definerer hvordan data blir send til API'et har ansvar for å validere og opprette et entitet-objekt

basert på hvilke attributter som objektet har definert. Et eksempel på dette er hvis aktiviteten ikke har en tid definert, så skal det opprettes et entitet-objekt som har både tid og varighet definert, mens hvis tid ikke er gitt så skal bare aktivitetens varighet være definert. Hver slik klasse implementerer derfor en konverteringsmetode, som er definert i et grensesnitt, som har ansvar for å opprette en riktig entitet. På denne måten kan vi opprette ulike aktivitet-entiteter basert på hvilken data klienten har definert ved opprettelse av en aktivitet.

For å unngå å sende unødvendig mengde med data, har vi differensiert mellom to ulike format som vi har kalt «kort» og «side». Kort-formatet brukes når en bruker eller aktivitet skal vises som et kort på frontend, mens side-formatet brukes for å returnere mer data om en gitt bruker eller aktivitet for en detaljert side på frontend. Et eksempel er at en aktivitet-side har en lang beskrivelse, mens et aktivitet-kort ikke har det. Hver modell har derfor også her en konverteringsmetode som gjør om en entitet til et gitt format. Hver modell bruker designmønsteret «builder pattern» for å enkelt bygge ulike objekter ut fra hvilket format som skal bli returnert. I Figur 25 er metoden for å konvertere en user-entitet til et user-kort-format med «builder pattern» vist.

Modeller som definerer hvordan data skal bli pakket og returnert fra API’et er definert i http-mappen. Disse modellene definerer altså hvordan json-objektet skal bygges opp. Et eksempel på dette er når vi returnerer en aktivitet-side, så ønsker vi å sette dette objektet på en activity-attributt. Dette vil gjøre det lettere for klienten å skjønne hva den får tilbake fra API’et. En annen fordel med denne modellen er at vi kan sende mer enn selve aktivitet-objektet. Når en liste med aktiviteter blir returnert, så får også klienten tilgang til hvilken side den er på, hvor mange sider det er totalt og et tall på hvor mange aktiviteter det er totalt. Hvordan disse lenkene og verdiene blir generert og kalkulert blir nærmere beskrevet i [delkapittel 5.2.3.4](#) om servises. Figuren under viser modellklassen for en liste med aktiviteter som blir returnert til klienten:

```

28     @Override
29     public UserCard cast(User user) {
30         return UserCard.builder()
31             .id(user.getId())
32             .avatar(user.getAvatar())
33             .name(Name.builder().firstName(user.getFirstName())
34                 .lastName(user.getLastName())
35                 .shortForm(user.getShortForm())
36                 .build())
37                 .email(user.getEmail())
38                 .slug(user.getSlug())
39                 .title(user.getTitle())
40                 .phoneNumber(user.getPhoneNumber())
41                 .preferredContact(user.getPreferredContact().getValue())
42                 .build();
43     }

```

Figur 25 – Metode på backend som konverterer (cast) user-entitet til usercard respons ved bruk av «builder pattern» – fra Juvets prosjektkode.

```

11     7 usages  ▲ Erik Storås Sommer
12
13     @Getter
14     @Setter
15     @ApiModel
16     @AllArgsConstructor
17     public class HttpActivityCardResponse<T extends ActivityCast<T>> extends HttpActivityResponse<T> {
18         private List<T> activities;
19         private String next;
20         private String previous;
21         private int page;
22         private int pages;
23         private int total;
24     }

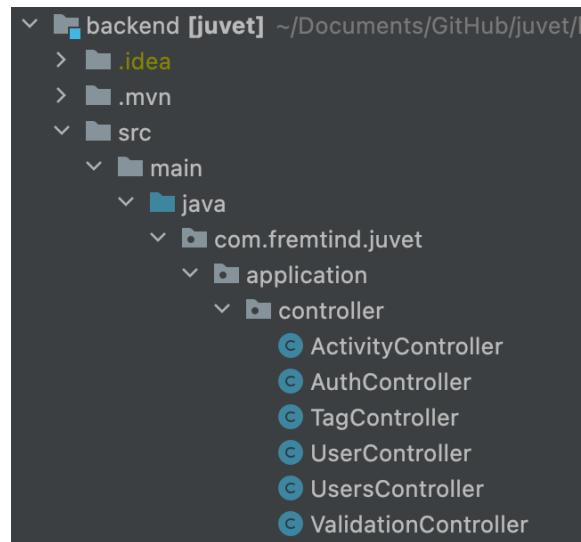
```

Figur 26 - Modellklassen som definerer hvilke data som skal sendes med aktivitetlisten på backend – fra Juvets prosjektkode

### 5.2.3.2 Controllers

Kontrollerne ligger i applications-laget og består av metoder som definerer endepunktene i API'et. Kontrollerne er altså den delen av applikasjonen som tar imot forespørsler over http fra klienten og returnerer tilhørende data. Hvert endepunkt som krever autentisering, er beskyttet med hvilke rettigheter den gitte brukeren må ha. Kontrollerne skal ikke gjøre mye logikk, men heller sende arbeidsoppgaven videre til forskjellige servises. Hver kontroller håndterer én entitet, som er lagret i databasen. I figur 27 er en oversikt over de seks kontrollerklassene og hvor de ligger i prosjektstrukturen.

Hver kontrollerklasse inneholder metoder som kalles et endepunkt. For å skille mellom ulike operasjoner, benyttes CRUD. CRUD er en forkortelse for Create, Read, Update, Delete, og er hovedoperasjonene som blir gjort mot dataene som er lagret i databasen (Stowe, 2014). Endepunktet for å søke etter aktiviteter er vist i figuren nedenfor, som er en get-metode med tilhørende http-url. En get-metode er en «Read»-operasjon som spesifiserer at endepunktet returnerer data (Lokesh, 2022).



Figur 27 - Mappestruktur som viser kontrollerklassene som definerer API'et – fra Juvets prosjektkode

```

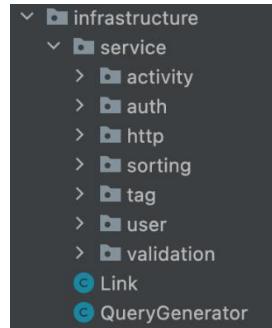
112     @PreAuthorize("hasAuthority('APPROLE_users')")
113     @GetMapping(value = "@{/}")
114     public ResponseEntity<?> getActivitiesSearchCard(@OAuth2AuthenticationToken auth, HttpServletRequest request,
115                                                       @RequestParam(required = false) String search,
116                                                       @RequestParam(required = false) String tags,
117                                                       @RequestParam(required = false) Integer count,
118                                                       @RequestParam(required = false) Integer page,
119                                                       @RequestParam(required = false) String date_from,
120                                                       @RequestParam(required = false) String date_to,
121                                                       @RequestParam(required = false) String sort) {
122
123         try {
124             Long userId = authService.getUserIdFromToken(auth);
125             if (userId == null) {
126                 log.error("User does not have an account in Juvet");
127                 return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(new HttpErrorResponse("Bruker er ikke registrert i juvet"));
128             }
129
130             HttpActivityResponse<ActivityCard> res = activityService.getActivitiesSearch(search, tags, count, page, date_from, date_to, sort, request, userId);
131             if (!((HttpActivityCardResponse<ActivityCard>) res).getActivities().isEmpty()) {
132                 log.info("Retrieved activities from search");
133             } else {
134                 log.error("Did not find any activities");
135             }
136             return ResponseEntity.status(HttpStatus.OK).body(res);
137         } catch (Exception e) {
138             log.error(e.getMessage());
139             return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(new HttpErrorResponse(e.getMessage()));
140         }
141     }
  
```

Figur 28 - Endepunkt for å søke på aktiviteter – fra Juvets prosjektkode

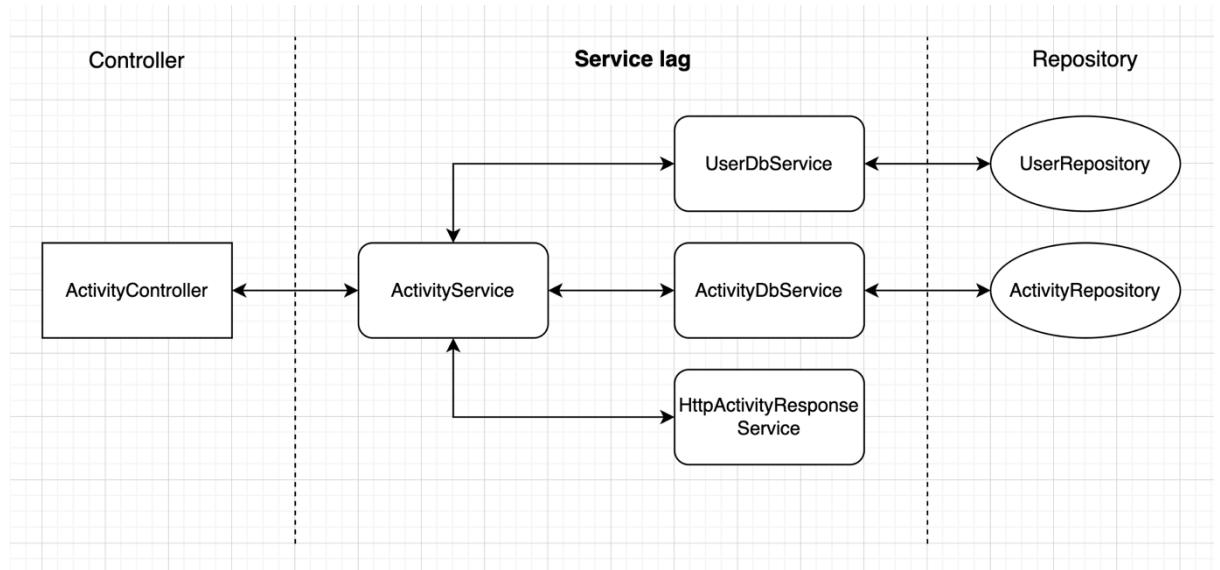
#### 5.2.3.4 Services

Service-klassene befinner seg i «*infrastructure*»-mappen og inneholder mesteparten av logikken applikasjonen trenger for å behandle data. Hver service-klasse implementerer et grensesnitt som definerer alle metodene service-klassen må inneholde, som til fordel gir løst koblet kode en god oversikt (GeeksforGeeks, 2020). Dette gjør det videre mulig å bruke Spring-rammeverket sin funksjonalitet for å injisere avhengigheter, på engelsk «dependency injection», i de klassene som behøver å bruke et service (Chand, 2019). Figur 29 viser en oversikt over service-inndelingen i prosjektstrukturen.

Service-klassene er delt inn i flere lag, hvor noen service er avhengig av flere services. Vi har gjort det slik for å fordele arbeidsoppgaver, gjenbruke kode, og minimere antall avhengigheter en kontroller har. Diagrammet i figur 30 beskriver hvordan kontrolleren for aktiviteter kun er avhengig av ActivityService-klassen for å få den dataen den trenger, og hvor ActivityService-klassen videre bruker tre services for å separere arbeidsoppgaver.



Figur 29 - Oversikt over service mapper – fra Juvets prosjektkode



Figur 30 - Diagram som viser hvordan en service er avhengig av flere services for å separere arbeidsoppgaver i Juvet backend

Service-laget utgjør en stor del av backend, og inneholder algoritmene som gjør arbeid på data. Her ligger algoritmene for å filtrere og sortere aktiviteter basert på flere parametere samt algoritmen for å generere http-linker for neste og forrige API-kall. Andre oppgaver service-laget har er å validere input og håndtere autentiseringen av brukere fra Azure AD opp mot vår database.

#### 5.2.3.5 Repositories

Hovedoppgaven til et repository er å utføre spørninger mot en entitet i databasen og returnere resultatet (Paraschiv, 2022). Vi behøvde å gjøre spørninger mot entitetene tag, user og activity, som krevde tre repository-klasser. Ved å definere et grensesnitt som arver fra klassen JpaRepository, fikk vi ferdig implementerte metoder som utførte CRUD-spøringer rettet mot de entitetene vi hadde definert. Ettersom vår applikasjon har en søkefunksjon, trengte vi mer spesifikke og avanserte spørninger. Jpa gjør det da mulig å definere sine egne metoder i grensesnittet som ut ifra

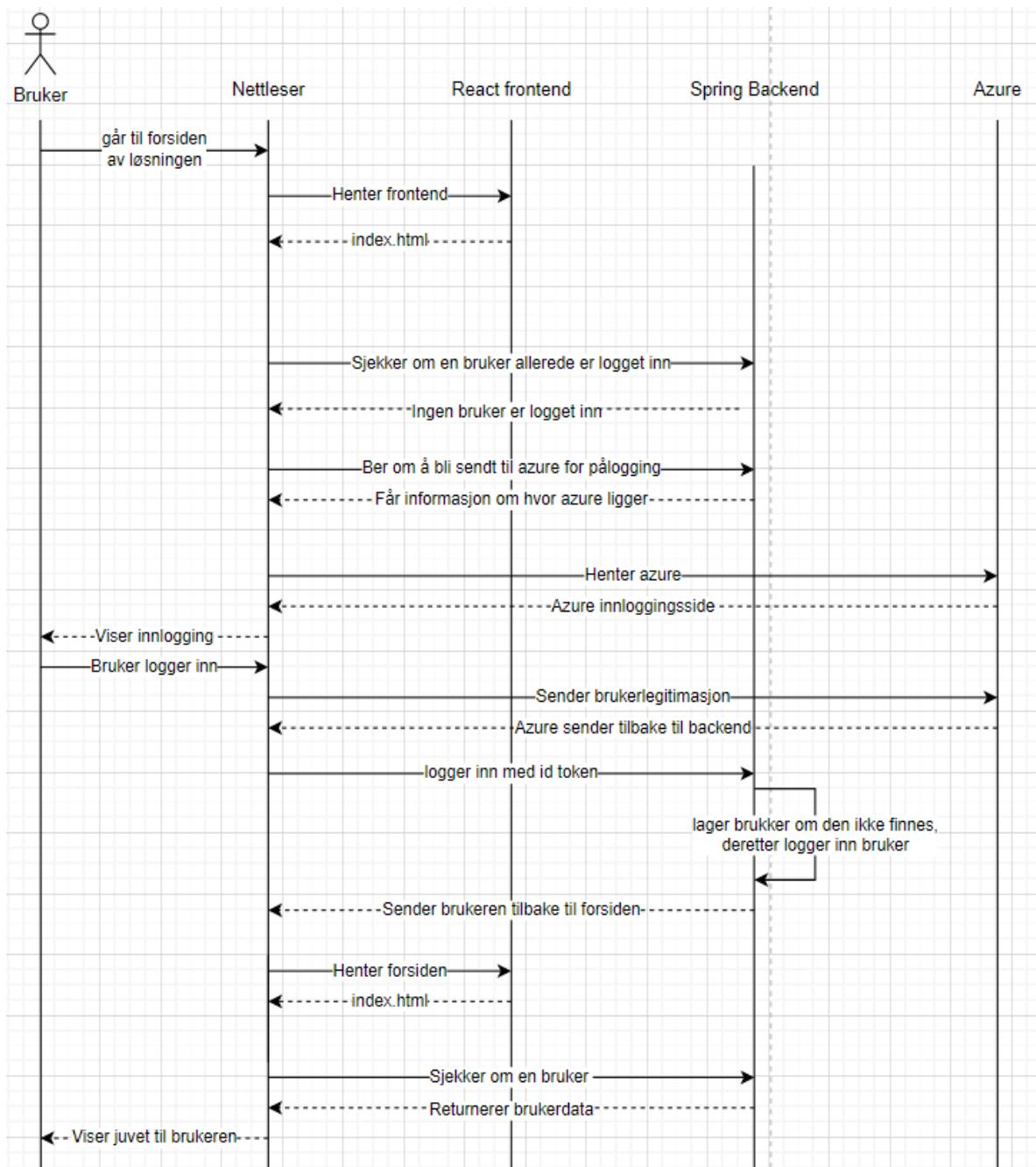
metodenavnet og parametere utfører en spørring (Tyson, What is JPA? Introduction to the Jakarta Persistence API, 2022). Figuren under viser noen av metodene vi definerte for å effektivisere søket:

```
11 @Repository
12 public interface ActivityRepository extends JpaRepository<Activity, Long> {
13     2 usages ▲ Erik Storås Sommer
14         List<Activity> findByTagsValueIgnoreCase(String type);
15
16     2 usages ▲ Erik Storås Sommer
17         List<Activity> findAllByDateAfterOrDateIsNull(Date date);
18
19     1 usage ▲ Erik Storås Sommer
20         List<Activity> findAllByDateBefore(Date date);
21
22     1 usage ▲ Erik Storås Sommer
23         List<Activity> findAllByDateBetween(Date dateStart, Date dateEnd);
24
25     1 usage ▲ Erik Storås Sommer
26         List<Activity> findAllByDateBetweenAndTitleContainingIgnoreCaseAndTagsValueIgnoreCase(Date dateStart,
27                                         Date dateEnd,
28                                         String title,
29                                         String type);
```

Figur 31 - Repository for å hente ut aktiviteter hvor avanserte spørringer er forenklet med metodenavn gitt av jpa – fra Juvets prosjektkode

#### 5.2.4 Enkeltpålogging

Hele API’et er sikret bak en autentiseringsvegg. For at det skulle være enkelt og sømløst for ansatte i Fremtind å ta i bruk Juvet, valgte vi å implementere deres enkeltpålogging, «single sign on» på engelsk. Det gjør at Fremtinds ansatte logger inn på Juvet med den samme Microsoft-kontoen de bruker til alt annet i bedriften. I figuren under vil du se strømmen i nettverksforespørslene under en innlogging på juvet.



Figur 32 - Viser nettverksforespørslene under en innlogging på Juvet

### 5.2.5 Design og tilgjengelighet

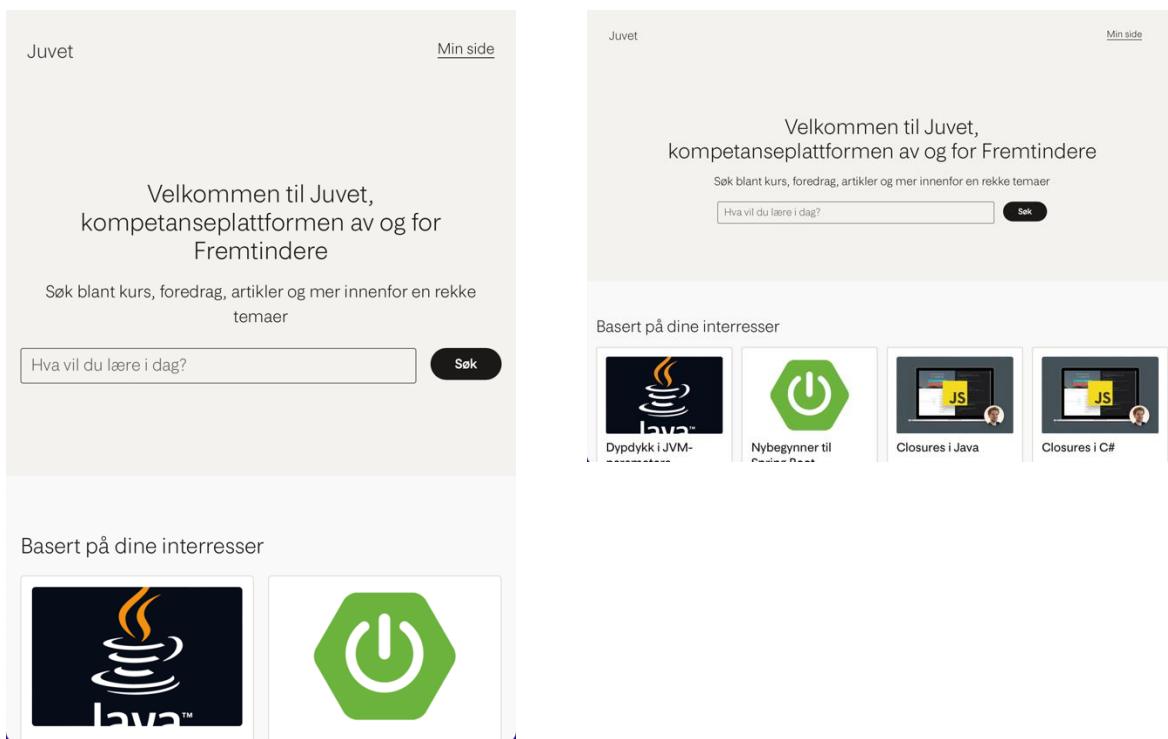
#### Design

Fra tidlig i prosjektet fikk vi tilbakemelding om at vi tolket designspråket og -prinsippene til Fremtind på en god måte. Mye av dette ble etablert fra designsprint, hvor vi også fikk inngående kjennskap til designsystemet Jøkul. I tillegg ble det i slutten av sprint 4 gjennomført et arbeidsmøte med en interaksjonsdesigner fra Fremtind. Dette var i sluttfasen av utformingen av endelig prototype. De faglige vurderingene rundt design ble gjort i samråd med interaksjonsdesigner. Samtidig som vi fulgte retningslinjene i designsystemet Jøkul; som hadde veldefinerte rutiner for bruk av deres komponenter. Siden ingen i prosjektgruppa hadde noe betydelig erfaring innen designarbeid, var det naturlig å følge alle tips og råd fra interaksjonsdesigneren og rutinene innen designsystemet. Blant

tilbakemeldingene var en av de mest konkrete at vi måtte passe på å vekte informasjonen på siden etter hvilken informasjon som var viktigst å få frem, og å bruke de ulike tekst- og tittelnivåene for å differensiere på dette.

### Responsivt design

Juvet ble utviklet med et responsivt design. En responsiv nettside betyr at innholdet tilpasses hvilken type skjerm/nettleservindu nettsiden vises på (Nettmaker.no, u.d.). Dette er viktig med tanke på brukervennligheten til nettsiden, da den sannsynligvis vil brukes på mange ulike typer enheter som smarttelefoner, nettbrett og PC'er. I figuren under vises landingssiden som eksempel, med to ulike vindusbredder, hvor innholdet stables mer i høyden jo smalere vinduet blir.



Figur 33 - Skjermbilder fra Juvet: Eksempel på responsiv nettside.

### Brukervennlighet

Brukervennlighet består av hvor enkelt, effektivt og/eller optimalisert (innen tid og kvalitet - engelsk: «effectiveness») en tjeneste eller applikasjon oppleves å bruke (Nigel, James, & Susan, 2015). Brukervennligheten kan ha ulike prioriteringsområder avhengig av hvem som er brukere av løsningen og hvilket mål som skal oppnås med den (Heintz, 2020). To ikke-funksjonelle krav til brukervennlighet ble stilt til vår løsning: «som bruker skal det være intuitivt å finne hvor mine aktiviteter er lagret» og «som bruker skal det være enkelt å legge aktiviteter i 'lagre til senere'». Dermed ble vår prioritering innen brukervennlighet enkelhet og effektivitet. Måten dette ble hensyntatt var i første omgang via utforming av prototypen og videre innen utformingen av den endelige prototypen samt i utviklingen av frontend.

For å forsøke å møte disse kravene ble «Min side»-knapp plassert øverst til høyre på navigasjonsbaren. Det at den ble plassert på navigasjonsbaren betydd også at knappen var tilgjengelig uansett hvilken side i løsningen brukeren var inne på. Det var på «min side»-siden det var tenkt at alle aktiviteter som hadde blitt «lagret til senere» skulle listes opp. Det ble dessverre ikke tid

til å implementere dette fullt ut, men gjennom spørsmålene i brukertestene ble kravet om å intuitivt skulle finne lagrede aktiviteter bekreftet. Inne på aktiviteten ble «Lagre til senere»-knappen plassert godt synlig til høyre i bildet ovenfor detaljene om aktiviteten. Gjennom brukertestene fikk vi bekreftet at kravene ble innfridd.

I designsprinten innledningsvis i prosjektet, ble noe av brukervennligheten testet. Ettersom komponentene som var brukt til å bygge prototypen var fra Jøkul, var mye av innholdet lett gjenkjennelig for de Fremtind-ansatte som var testbrukere. Dette på grunn av at løsningen vår hadde en tydelig «Fremtind-drakt». Mer om tilbakemeldingene fra brukertesten kan sees i [delkapittel 4.1](#) som omhandler denne testen, eller i vedlegget om [brukertest under designsprint](#).

## Universell utforming

I designsystemet Jøkul inngår også veiledning i hvordan deres egne utviklede nettsider skal være for å tilfredsstille kravene til universell utforming (Fremtind Forsikring, u.d.). I utviklingen ble den universelle utformingen hensyntatt ved å følge guiden fra Jøkul til en viss grad. På grunn av begrenset tid måtte implementering av full universell utforming nedprioriteres.

For å få en score på hvor godt løsningen er utviklet med tanke på universell utforming, ble den testet i en tilgjengelighetstest, Siteimprove.com (Siteimprove, u.d.). Mer om tilgjengelighetstesting kan sees i [kapittel 4.4](#). Den tar for seg «WCAG-standarden» og undersøker de kravene den stiller til universell utforming av nettsider. WCAG står for «Web Content Accessibility Guidelines» og er en rekke retningslinjer for å tilgjengeliggjøre nettsider. Ved å følge disse retningslinjene vil personer med nedsatt funksjonsevne, enten det er syn, hørsel eller andre begrensninger som skaper større utfordring i konsumering av innhold på en nettside, også kunne få den samme informasjonen fra nettsiden som en fullt funksjonsfrisk person.

Siden 1. januar 2021 ble det et krav om at allment tilgjengelige nettløsninger må følge kravene til universell utforming (Tilsynet for universell utforming av IKT, u.d.). Vår løsning er ikke omfattet av dette kravet fordi den ikke er allment tilgjengelig. I tillegg skal alle intranett i offentlig sektor fra 1. februar 2023 følge krav om universell utforming, etter innføring av EUs webdirektiv (WAD) i norsk lov (Tilsynet for universell utforming av IKT, u.d.). I tilfellet for Juvet, vil ikke dette være gjeldende da organisasjonen ikke er en del av offentlig sektor. Likevel er det ønskelig at løsningen skal tilfredsstille så mange av retningslinjene som mulig og dette er etterstrebet. Samtidig krever implementering i koden for å hensynta alle kravene for universell utforming mye tid, og vi var derfor nødt til å nedprioritere dette til fordel for viktigere funksjonalitet i løsningen.

## Brukeropplevelse

Ved å utvikle denne typen løsning til å være en webapplikasjon, gir det en tilgjengelighet som styrker mulighetene for å oppnå en god brukeropplevelse. En god brukeropplevelse består av både begrepet brukervennlighet og tilgjengelighet, i tillegg til andre aspekter som f.eks. merkevarebygging (Sander, 2021). Det vil si at brukeropplevelsen utgjør den totale oppfattelsen av tjenesten eller løsningen. For vår løsning, Juvet, som er en webapplikasjon betyr det at alle Fremtinds ansatte med en PC med nettilgang og tilkobling til Fremtinds nettverk via VPN (virtuelt privat nettverk), vil få tilgang til Juvet fra sin PC. Alle Fremtind-ansatte jobber med denne tilkoblingen som standard, slik at det ikke krever noe ekstra innsats for den enkelte brukeren. Alternativer til former for å utvikle løsningen, som å utvikle Juvet som en egen applikasjon på en PC ville blant annet ha begrenset tilgjengeligheten og dermed sannsynligvis også påvirket brukeropplevelsen i negativ grad.

### 5.3 Samsvar mellom kravspesifikasjon og produkt

Samsvaret mellom kravspesifikasjon og endelig produkt var, etter en akseptansetest hos oppdragsgiver, oppsummert som godt. En akseptansetest er en form for test som avdekker om produktet oppfyller oppdragsgivers krav til funksjonalitet, brukervennlighet og forretningsmessige krav (Software Testing Help, 2022). Kontaktpersonen var godt fornøyd med produktet, særlig det tydelige konseptet som vi hadde utviklet for organisasjonen og testet på potensielle brukere i bedriften. Basert på akseptansetesten ble graden av suksess vurdert for de ulike kravene som ble definert i kravspesifikasjon.

I tabellene under betyr grønn bakgrunnsfarge at kravet er oppfylt, gul bakgrunnsfarge betyr at kravet verken er oppfylt eller underkjent, og rød bakgrunnsfarge betyr at kravet er underkjent.

*Tabell 4 - Samsvar mellom kravspesifikasjon og endelig produkt, funksjonelle krav.*

#### Funksjonelle krav

Som bruker...	Prioritet	Krav oppfylt
må løsningen skal være tilgjengelig for internt ansatte i Fremtind	Høy	Ja
må jeg kunne søke på innhold på tvers av personer/emner/aktiviteter	Høy	Ja
må jeg kunne opprette nye aktiviteter i løsningen	Høy	Ja
må jeg kunne kontakte andre i organisasjonen basert på mitt søk på kompetanse	Høy	Ja
må jeg kunne logge inn med «single sign on»	Høy	Ja

Som oppdragsgiver ønsker Fremtind at den...		Krav oppfylt
bør den være «hostet» på AWS	Middels	Ja

*Tabell 5 - Samsvar mellom kravspesifikasjon og endelig produkt, ikke-funksjonelle krav.*

#### Ikke-funksjonelle krav

Som bruker...	Prioritet	Krav oppfylt
bør jeg få relevant informasjon uten å bli overveldet av informasjon	Middels	Ja
må det være enkelt å legge aktiviteter i «lagre til senere»	Høy	Ja
må det være intuitivt å finne hvor mine aktiviteter er lagret	Høy	Ja
bør føle en lav terskel for å delta på aktiviteter	Middels	Ja

Som oppdragsgiver ønsker Fremtind at løsningen...		Krav oppfylt
kan «gamifye» deltagelse på aktiviteter	Lav	Nei
Må hente brukerinfo fra AzureAD for å opprette bruker i Juvet	Høy	Ja
kan sørge for at ansatte som ikke fikk tid til å være med på aktiviteten også kan få læringsutbytte	Lav	Vet ikke

Av de 13 kravene som er definert i kravspesifikasjonen er 11 av disse oppfylt, ett krav «vet ikke» som betyr at det er vanskelig å vurdere uten å gjennomføre en brukertest/spørreundersøkelse etter at løsningen har vært brukertestet en lengre periode. Til slutt er ett krav underkjent. Det betyr at vi nådde en prosentvis oppnåelse på ca. 85%. Vi går nærmere inn på drøfting og konklusjon rundt dette i [kapittel 7](#).

#### 5.4 Sentrale datastrukturer i løsningen

Vi vil her presentere de mest sentrale datastrukturene for både frontend- og backend-siden av løsningen.

##### 5.4.1 Prinsipper for utvikling av kode

###### Objekt-orientert programmering

Objekt-orientert programmering er en måte å programmere på for å definere og strukturere objekter og klasser slik at det best mulig representerer det vi ønsker å bruke programmet til å bearbeide (Vihovde, 2021). Dette gjør at vi kan benytte arv, polymorfisme og gjenbruke klassene for å få en tydelig struktur og effektiv kode.

Vi har utviklet etter dette prinsippet i backend hvor objektene som prosesseres er definert som modeller.

Programmeringsspråket Java, som backenden er skrevet i, er et objekt-orientert programmeringsspråk som utnytter fordelene det innebærer, som blant annet er gjenbruk av kode og lavere produksjonskostnader (Hartman, What is Java? Definition, Meaning & Features of Java Platforms, 2022).

##### 5.4.2 Frontend-modeller

De to mest sentrale modellene for TypeScript-typer i frontend, speiler de sentrale klassene i backend; aktivitet («Activity») og bruker («User»). I figuren under er Activity definert og er eksempel på en modell i Juvets frontend.

```
export interface Activity extends Entity {
  slug: string;
  title: string;
  shortDescription: string;
  thumbnail: {
    url: string;
    alt: string;
  };
  duration: Duration;
  authors: Author[];
  location?: Location;
  format: ActivityFormat;
  registration?: Registration;
  saved: boolean;
}
```

Figur 34 - "Activity" TypeScript-interface definert - fra Juvets prosjektkode.

Fremtinds designsystem, Jøkul, hjalp oss mye i byggingen av siden ved at de hadde noen ferdigbygde komponenter. Jøkul ligger åpent tilgjengelig på en egen nettside hvor hvem som helst kan gå inn og lære systemet og se både eksempler på bruk av komponentene og koden som skal til for å bruke den. Bildet i figur 35 nedenfor er hentet fra Jøkul sine nettsider, «[jokul.fremtind.no](http://jokul.fremtind.no)», og beskriver hvordan denne knappen ser ut, hvordan den oppfører seg, og hvordan den er ment å brukes. I tillegg er det mulig å prøve funksjonaliteten til komponenten. For at vi skulle kunne bruke den spesifikke komponenten måtte kodepakken for komponenten lastes inn i vårt prosjekt ved å bruke «Npm» eller «Yarn». Eksempel på hvordan en komponent i Jøkuls komponentbibliotek så ut sees i figuren under.

# Buttons

[React ↗](#) [Sass ↗](#)

Knapper starter en handling. Teksten på knappen forteller hva som vil skje når brukeren klikker på den.

**Primærknapp:** brukes til den viktigste handlingen på en side - hovedhandlingen. En side kan noen ganger ha mer enn en hovedhandling, men ikke mange.

The screenshot shows the 'PrimaryButton' component configuration interface. On the left, there's a preview window showing a dark button with white text that says 'Lagre og send inn'. To the right of the preview are several configuration sections with checkboxes and radio buttons:

- Egenskaper**:
  - Compact
  - withLoader
  - isLoading
- Pil**:
  - uten
  - left
  - right
- Visning**:
  - Dark mode

Skjul kode ↑

TSX

```
<PrimaryButton
    forceCompact={false}
    loader={false}
    onClick={simulateLoading}
    className="jkl-spacing-1--right"

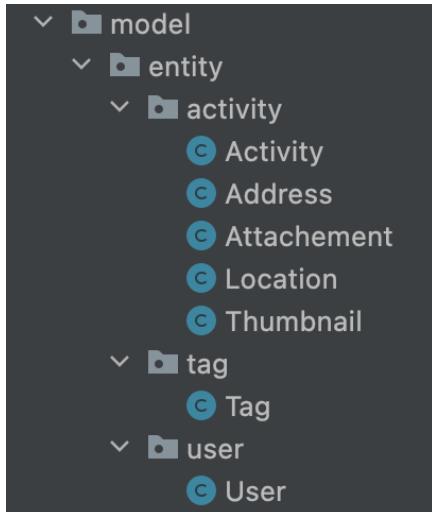
    >
    Lagre og send inn
</PrimaryButton>
```

Figur 35 - Utklipp fra Fremtinds designsysteem ([jokul.fremtind.no/komponenter/buttons](http://jokul.fremtind.no/komponenter/buttons)). Eksempel på ferdig komponent for primærknapp.

### 5.4.3 Backend-modeller

I likhet med frontend-siden, er aktivitetsklassen og brukerklassen de to mest sentrale datastrukturene. Det var viktig at aktivitets- og brukerklassen var veldefinerte ettersom løsningen i sin helhet omhandler aktiviteter og brukere. Disse aktivitetene har minst én forfatter (som også er

en bruker), og aktivitetene kan ha flere påmeldte (brukere). En bruker kan lagre denne aktiviteten i sin «lagret til senere»-liste eller være påmeldt aktiviteten. I figuren under sees filstruktur for modellene i vår løsning, samt attributtene i «Activity»-klassen. Se mer om relasjonene mellom klassene i diagrammet i neste delkapittel, [5.4.4](#).



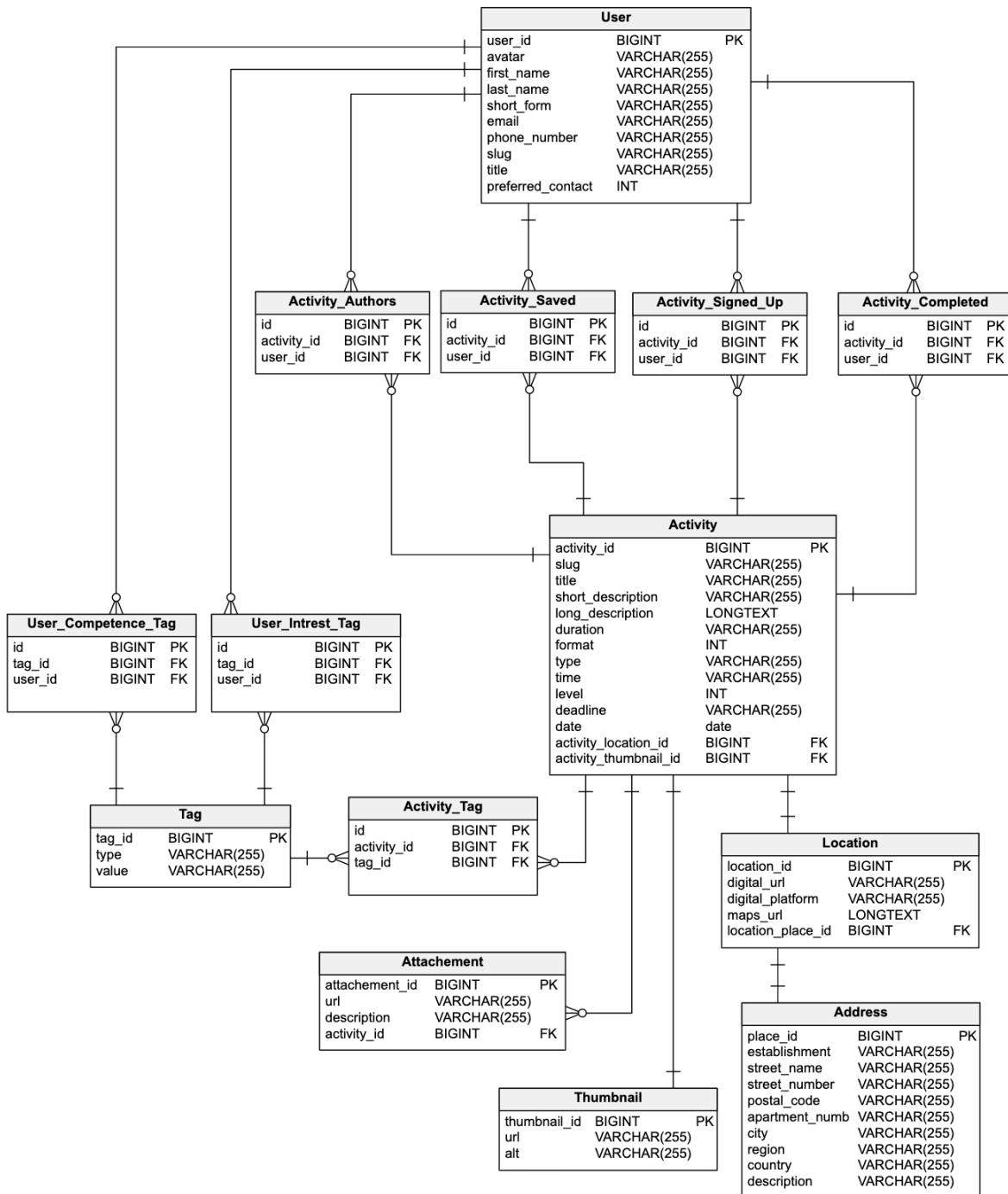
Figur 36 - Modellene illustrert i filstrukturen for backend

Activity		
f	time	String
f	location	Location
f	slug	String
f	level	Level
f	usersSavedForLater	Set<User>
f	usersCompleted	Set<User>
f	shortDescription	String
f	longDescription	String
f	title	String
f	usersSignedUp	Set<User>
f	thumbnail	Thumbnail
f	deadline	String
f	id	Long
f	type	String
f	format	Format
f	tags	Set<Tag>
f	duration	String
f	attachements	Set<Attachement >
f	authors	Set<User>
f	date	Date

Figur 37 - Attributtene i aktivitetsklassen

#### 5.4.4 Database – Entitet Relasjons diagram

I figur 38 er implementeringen av relasjonsdatabasen vår illustrert i et ER-diagram. I figuren ser vi entitetene som utgjør tabellene i databasen med tilhørende attributter og relasjoner til andre tabeller. Ettersom en bruker kan melde seg opp, lagre, fullføre og være forfatter til en aktivitet, og motsatt, opprettet vi en samletabell for alle slike relasjoner. Dette ble gjort for å splitte opp mange-til-mange-relasjoner. Totalt, med de syv entitetene vi har definert og samletabellene mellom dem, består vår relasjonsdatabasen av 14 tabeller.

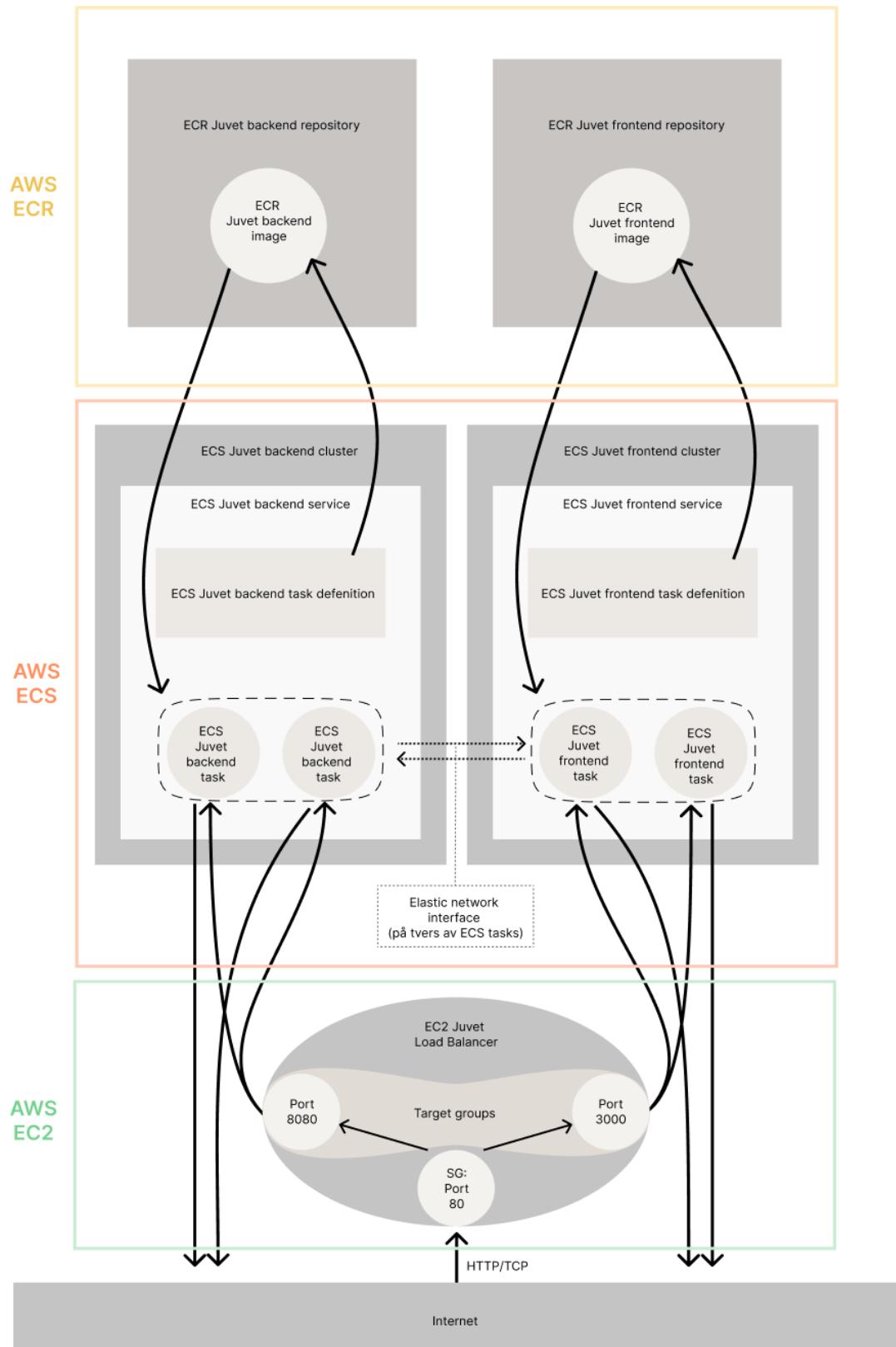


Figur 38 – ER-diagram av vår database som viser relasjoner mellom tabeller

## 5.5 AWS-hosting og -deployering

For beskrivelse av oppsett og deployering i Amazon Web Services, beskrives flyten i oppsettet i figuren under, som inneholder hovedkomponentene i oppsettet. Detaljene er beskrevet i teksten.

### AWS EC2 Fargate oppsett



Figur 39 - Illustrasjon av AWS-oppsett for Juvet.

For å etablere et fungerende oppsett med AWS Fargate må flere parametere defineres på AWS-plattformen (Amazon Web Services, u.d.). Først opprettet vi to repositories i AWSs «Elastic Container Registry» (heretter kalt ECR); én for backend og én for frontend. Begge disse ble satt opp

med privat tilgjengelighetsmodus, som betyr at vi måtte definere adganger til repository'ene med AWSs «IAM» bruker- og rolleregister (Amazon Web Services, u.d.).

I AWS IAM ble det registrert én Juvet-admin-bruker med alle rettigheter og én Juvet-rolle som også hadde alle rettigheter. Så lenge vi var i utviklingsfase hadde vi ikke behov for flere nivå med tilganger, og heller ikke at flere andre personer skulle ha tilgang til å endre på våre ECR-repository'er.

Deretter opprettet vi to «clusters» i «Elastic Container Service» (heretter kalt ECS). Et cluster er en gruppe av tjenester (services) som logisk hører sammen (Amazon Web Services, u.d.). Vi hadde kun én tjeneste pr. cluster, da det passet vårt oppsett best. Inne i hver av clustrene ble en ECS-service opprettet. Det er denne servicen som kjører nye instanser (tasks) av koden og holder orden på hvor mange instanser som skal kjøres (Amazon Web Services, u.d.). ECS-servicen har en «scheduler» som passer på å opprette nye instanser, følge med på helsen til de kjørende instansene, og avslutter instansene dersom noe i instansen går galt. Denne scheduleren kan ha to planleggingsstrategier; replica og daemon. Med «daemon» valgt kjører kun én instans av servicen til enhver tid, til forskjell fra «replica» hvor man selv kan definere hvor mange instanser som skal kjøres. I vårt tilfelle valgte vi minimum 2 replica-instanser av hver service; dvs. 2 backend-instanser og 2 frontend-instanser. Dette valgte vi for å demonstrere at kommunikasjon mellom flere frontend-instanser og backend-instanser fungerte, og at det senere dermed er mulig skalere antall instanser etter hvilken belastning tjenesten opplever.

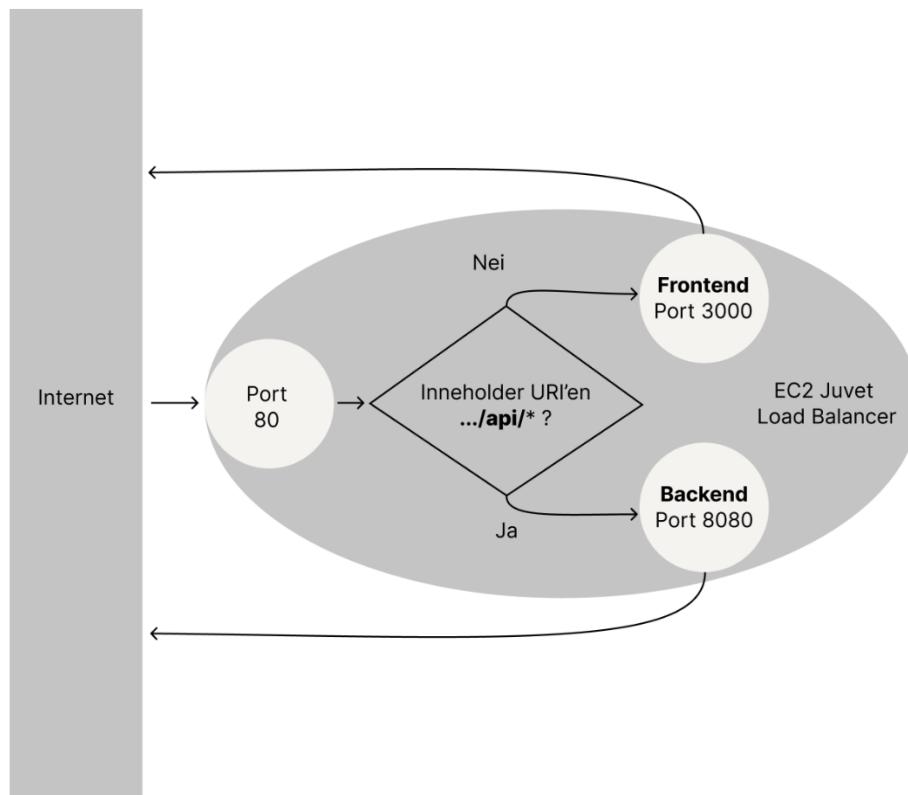
Hvilket image som kjøres i ECS-servicen bestemmes av ECS «task definition» (Amazon Web Services, u.d.), som henter det imaget som ønskes fra ECR og oppretter en node av denne som instans i ECS-service. Ved opprettelse av en ny «task definition» bestemmer vi hvilken styrke i form av CPU-kraft og mengde RAM som våre instanser skal få tildelt på den virtuelle serveren. Vi valgte minimum av det som var anbefalt av AWS som var beskrevet via hjelpetekst ved opprettelse av ny task definition. Dette fordi vår løsning ikke setter så høye krav til hverken CPU-regnekraft eller plass på RAM. De ble derfor satt til 2GB RAM og 1024 CPU-enheter.

En viktig parameter vi bestemte i task defenition var nettverksmodus. Som ble satt til «awsvpc» som står for «AWS viritual private cloud» og sørger for «elastic network interface» til sine instanser, som er kravpålagt i et AWS Fargate-oppsett (Amazon Web Services, Inc., u.d.). Et «elastic network interface» gir hver instans sin egen nettverksadresse som gjør at hver instans kan kommunisere med både andre instanser utenfor sin cluster og ellers på nettverket eller andre deler av internett. Hvilke adganger og begrensninger for kommunikasjon til og fra disse instansene blir bestemt av «EC2 security groups». En EC2 security group består av regler for inngående og utgående trafikk. Security group for våre instanser på både backend og frontend er åpen for all TCP-trafikk på alle porter fra IPv4-adresser, både for inngående og utgående trafikk. I tillegg måtte vi definere en security group for vår «load balancer». Hva en load balancer er forklares i avsnittet under, men dens security group var definert til å godta all trafikk på port 80, som er standard HTTP-trafikk.

Det ytterste «laget» mot det åpne internett i vår løsning, var en «load balancer». En load balancer er en omvendt proxy, som betyr at denne har en virtuell nettadresse, men dirigerer trafikken videre til servere som står i bakkant (Amazon Web Services, u.d.). Disse serverne behandler- og eventuelt responderer på henvendelsene. Vår load balancer ble satt opp til å være en «application load balancer», som støttet AWS Fargate-tjenester, og var mest korrekt for vår løsning ettersom den var en komplett web-applikasjon. Load balanceren omdirigerte all trafikk som hadde «.../api/\*» (f.eks. «juvet.fremitind.no/api/v1/activities/all») direkte til backend-tjenesten. Ellers ble all annen trafikk dirigert til frontend-tjenesten. En load balancer kan også konfigureres til å skalere de ulike tjenestene som applikasjonen består av ut ifra hvor stor trafikk den opplever. I vårt tilfelle; backend-

tjenesten og frontend-tjenesten. Dette valgte vi å ikke implementere under utviklingen fordi det ikke var nødvendig ettersom antall henvendelser til våre services var relativt få.

En siste viktig del som inngår i load balancer er «target groups». Target groups definerer hvor trafikken skal omdirigeres av load balancer. Det betyr at der vi definerte at alle henvendelser med «.../api/\*» skulle gå til backend, ble dirigert til port 8080, mens all annen trafikk gikk til frontend-tjenesten på port 3000. Se figur under.



Figur 40 - AWS load balancer med betinget ruting, slik det ble satt opp for Juvet

## 5.6 Sikkerhet

Fremtind Forsikring er en bedrift som er avhengig av å ha integritet og verdsetter sikkerhet høyt.

For å få tilganger til deres systemer, måtte vi benytte en Fremtind-PC som den ene i gruppen hadde. Med denne kunne vi få tilgang til AWS-plattformen og samtidig via Outlook og Teams fikk en bedre oversikt over ansatte i Fremtind som kunne bistå oss i utviklingen.

De av gruppemedlemmene som ikke hadde et ansettelsesforhold hos Fremtind måtte levere en taushetserklæring for å få innsyn i kodebasen og for å kunne motta et personlig adgangskort til kontorlokalene.

Ved AWS-deployering av koden satte Fremtind tydelige føringer for hva som ansees som sikkert og hva som ikke burde tilknyttes deres navn eller domene. Under testing av AWS-deployering ble vi bedt om å ikke knytte dette til en url/web-adresse som kunne forbides med Fremtind. Dette gjaldt så lenge det ikke lå et sikkerhetslag mellom løsningen og det åpne internett. Derfor ble AWS-testen gjennomført med å deployere løsningen på en tilfeldig url som ble generert av AWS; «<http://juvet-dev-lb-1329289883.eu-west-1.elb.amazonaws.com>» (adressen er ikke lengre i bruk). Selv om noen skulle klare å finne frem til denne webadressen, ville vedkomne uansett ikke få noe ut av det da det

ikke ble brukt ekte data på dette stadiet - kun mock-data som ikke ga noen verdi annet enn for vår testings skyld.

Et teknisk sikkerhetsaspekt som dreide seg om selve koden i prosjektet, var eventuelle svakheter som lå i de ulike avhengighetene som er benyttet. Disse avhengighetene (engelsk: «dependencies») var bibliotek som ble importert fra en ekstern ressurs ved bygging av prosjektkoden. Det gjaldt både på frontend og backend. Alle disse avhengighetene ble deklarert med pakkenavn og versjonsnummer i en «pom.xml»-fil på backend, mens de på frontend lå i en «package.json»-fil. Avhengighetene ble importert av avhengighetsverktøyene Yarn og Maven. Versjonsoppdatering av disse avhengighetene, derimot, utføres ikke av Yarn og Maven, og utgjorde en potensiell sikkerhetsrisiko dersom det var et sikkerhetshull i en av avhengighetene. Derfor er det ønskelig at versjonen av avhengighetene ble oppdatert etter hvert som nye versjoner ble tilgjengelige. Denne oppdateringen var det «dependabot» som var ansvarlig for i kodebasen til Juvet (dependabot, u.d.). Det var en fordel for sikkerheten, men også for vedlikeholdet av prosjektet.

## 5.7 Forhold til maskiner/database/OS

Vi vil her presentere de funksjonelle grensesnittene i systemet. Spesielt viktig er API'et, som er bindeleddet mellom frontend og backend. Ettersom vår applikasjon kjører på containere og er for en bruker av applikasjonen publisert på intranett, vil ikke typen operativsystem være en avgjørende faktor for kjøringen. Juvet støtter ikke eldre nettlesere, da Fremtind kun bruker moderne nettlesere.

### 5.7.1 API dokumentasjon

API dokumentasjonen gir en oversikt over hvilke endepunkter som er tilgjengelig på backend-serveren og hvordan du sender og mottar data fra den (Swagger.io, u.d.). Dette er til fordel for klienten, som ønsker å benytte seg av serveren, og for utvikleren, som skal kunne videreutvikle backend. I figur 41 er det en oversikt over hvilke endepunkter REST API'et består av. Oversikten er hentet fra Swagger UI som er implementert og konfigurert på backend-serveren.

<b>activity-controller</b> Activity Controller	<b>users-controller</b> Users Controller
<a href="#">GET /api/v1/activities getActivitiesSearchCard</a>	<a href="#">GET /api/v1/users getUsers</a>
<a href="#">POST /api/v1/activities postActivity</a>	<a href="#">POST /api/v1/users/ createUser</a>
<a href="#">PUT /api/v1/activities/{id} putActivity</a>	<a href="#">GET /api/v1/users/{id} getUser</a>
<a href="#">DELETE /api/v1/activities/{id} deleteActivity</a>	<a href="#">GET /api/v1/users/{id}/card getUserCard</a>
<a href="#">GET /api/v1/activities/{id}/card getActivityCardByld</a>	<a href="#">GET /api/v1/users/all getUsers</a>
<a href="#">POST /api/v1/activities/{id}/completed userCompletedActivity</a>	<a href="#">GET /api/v1/users/name getUsersName</a>
<a href="#">POST /api/v1/activities/{id}/notcompleted userNotCompletedActivity</a>	
<a href="#">POST /api/v1/activities/{id}/quit userRemoveActivitySignedUp</a>	
<a href="#">POST /api/v1/activities/{id}/remove userRemoveActivitySaved</a>	
<a href="#">POST /api/v1/activities/{id}/save userSaveActivity</a>	
<a href="#">POST /api/v1/activities/{id}/signup userSignUp</a>	
<a href="#">GET /api/v1/activities/{param} getActivity</a>	
<a href="#">GET /api/v1/activities/all getActivities</a>	
<a href="#">GET /api/v1/activities/author getActivitiesAutor</a>	
<a href="#">GET /api/v1/activities/completed getActivitiesCompleted</a>	
<a href="#">GET /api/v1/activities/saved getActivitiesSaved</a>	
<a href="#">GET /api/v1/activities/signup getActivitiesSignedUp</a>	
<b>tag-controller</b> Tag Controller	<b>auth-controller</b> Auth Controller
	<a href="#">GET /api/v1/auth/signin signin</a>
	<a href="#">GET /api/v1/auth/whoami whoAmI</a>
<b>validation-controller</b> Validation Controller	<b>tag-controller</b> Tag Controller
<a href="#">POST /api/v1/validate/activities/longDescription validateActivityLongDescription</a>	<a href="#">GET /api/v1/tags getTagsSearch</a>
<a href="#">POST /api/v1/validate/activities/shortDescription validateActivityShortDescription</a>	<a href="#">POST /api/v1/tags createTag</a>
<a href="#">POST /api/v1/validate/activities/title validateActivityTitle</a>	<a href="#">DELETE /api/v1/tags/{id} deleteTag</a>
<a href="#">POST /api/v1/validate/activities/type validateActivityType</a>	
<a href="#">POST /api/v1/validate/date validateDate</a>	
<a href="#">POST /api/v1/validate/duration validateDuration</a>	
<a href="#">POST /api/v1/validate/tags/value validateTagsValue</a>	
<a href="#">POST /api/v1/validate/users/name validateUserName</a>	
<a href="#">POST /api/v1/validate/users/title validateUserTitle</a>	
	<b>user-controller</b> User Controller
	<a href="#">PUT /api/v1/user updateUser</a>
	<a href="#">GET /api/v1/user/competence getCompetences</a>
	<a href="#">POST /api/v1/user/competence createCompetence</a>
	<a href="#">DELETE /api/v1/user/competence/{id} deleteCompetence</a>
	<a href="#">GET /api/v1/user/interests getInterests</a>
	<a href="#">POST /api/v1/user/interests createInterest</a>
	<a href="#">DELETE /api/v1/user/interests/{id} deleteInterest</a>
	<a href="#">POST /api/v1/user/preferred-contact/email updatePreferredContactEmail</a>
	<a href="#">POST /api/v1/user/preferred-contact/phone updatePreferredContactPhone</a>
	<a href="#">POST /api/v1/user/preferred-contact/teams updatePreferredContactTeams</a>

Figur 41 - Oversikt over Juvets REST API

### 5.7.2 Databaseintegrasjon

Grensesnittet for databasen og backend kommuniserte via en URL med pålogging som innebar brukernavn og passord. Databaseinstansen ble satt opp som en offentlig og tilgjengelig ressurs hos AWS RDS. Se database-konfigurasjonsfilen i figuren under. Her er brukernavn og passord til databasen sensurert av sikkerhetshensyn for oppdragsgiver. Denne ble veldig implementert, men ble på grunn av ytelsesproblemer fjernet til fordel for H2 in-memory-database.

```
## AWS hosted MySQL test-database
spring.jpa.open-in-view=false
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto= update
spring.jpa.hibernate.use-new-id-generator-mappings=true
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://juvetdbtest.cn92wj9yxr.eu-west-1.rds.amazonaws.com:3306/juvetdbtest
spring.datasource.username=[REDACTED]
spring.datasource.password=[REDACTED]
spring.data.jpa.repositories.enabled=true
spring.datasource.hikari.maxLifeTime = 300000
spring.jpa.show-sql=false
spring.jpa.database=mysql
```

Figur 42 - AWS MySQL database konfigurasjonsfil fra prosjektkoden i Juvet

## 5.8 Hoveddeler av programmet

Vi vil her kort presentere løsningens hoveddeler med formål og oversikt over rutinene i løsningen.

### SSO med AzureAD

Alt innhold på Juvet er presentert i kontekst av den personlige brukeren som er pålogget. Derfor var SSO med AzureAD en viktig del av løsningen. Siden Fremtinds PC'er og systemer baserer seg på tilgang til organisasjonens VPN og SSO med AzureAD, var det naturlig å utvikle denne løsningen på lik linje med de andre systemene. På denne måten slipper en bruker å opprette en profil og passord på Juvet for å kunne bruke løsningen. For å kunne gi en profil med brukers epost, var det ønskelig at dette skulle hentes fra AzureAD. Bruker-modellen på både frontend og backend inneholder attributter for dette.

### Bruker

En bruker i Juvet vil opprettes ved første innlogging og krever ikke et større oppsett av brukerprofil, passord da denne infoen ligger i SSO-innloggingen med AzureAD. En bruker har mulighet til å opprette nye aktiviteter, melde seg på eksisterende aktiviteter som ligger i Juvet og å lagre interessante aktiviteter i en «Lagret til senere»-liste på sin profil.

Noen funksjoner i konseptet ble utviklet på backend-delen av applikasjonen, men ble dessverre ikke utviklet på frontend på grunn av tidsbegrensning. Dette dreide seg om:

- Funksjoner hvor brukeren kan definere en liste med ekspert-emner på sin profil. Det er emner som brukeren sier seg villig til å kunne være søkbar på i Juvet og dermed være tilgjengelig for andre kollegaer for å bli kontaktet ved spørsmål rundt emnet.
- Funksjon hvor brukeren kunne definere sine interesse-emner på sin profilside, og basert på disse emnene kunne få presentert det mest relevante innholdet øverst i Juvet.

## Aktivitet

Aktivitetene er hovedelementene i Juvet. Siden aktiviteter kan være fysiske, digitale eller hybride, er aktivitetene delt inn i disse tre hovedkategoriene. I tillegg til å tydelig kunne skille på fysiske, digitale og hybride aktiviteter, og deres relevante attributter, gir dette også mulighetene for å skalere løsningen til å omfatte flere områder enn det den i utgangspunktet var tiltenkt.

Det er ulike betingelser/begrensninger som er satt for de ulike aktivitetene:

- Dato/tid: En ressurs-basert aktivitet har ikke en dato eller tid, men en anslagsvis varighet for å si noe om forventet tidsbruk på aktiviteten. Dette kan for eksempel et innspilt webinar. Den har kun en adresse, enten fysisk eller digital. Alle andre aktiviteter har en dato, starttidspunkt og en varighet.
- Adresse: En fysisk aktivitet har kun en fysisk adresse. En digital aktivitet har kun en digital adresse, mens en hybrid aktivitet skal ha både en fysisk adresse og en digital adresse for oppmøte.
- Tekstinput: En tittel kan maks innehölde 50 antall tegn. En kort beskrivelse kan maks innehölde 100 antall tegn. En lang beskrivelse må minimum innehölde 100 antall tegn.

Aktiviteten er søkbar på tittel, dato og emner, og kan sorteres på relevans på brukers interesser.

I likhet med utviklingen for brukere, ble ikke all funksjonalitet for aktiviteter fullt utviklet på frontend. Dette omhandlet aktivitetenes emner, eller «tags», på hvilken type kompetanse/kunnskap som en bruker kan erverve ved å delta på aktiviteten. Ved at en bruker fullfører aktiviteten, vil kompetansen/kunnskapen som er registrert på aktiviteten, kopieres til brukerprofilens kompetanseliste.

## Eksperter

Som nevnt i avsnittet om bruker, er det at en bruker definerer seg som ekspert innen et eller flere fagområder viktig for løsningen. Dette betyr at løsningen ikke bare kan sette kollegaer på tvers av team/avdelinger i kontakt med hverandre, men det vil også være en presentasjon av kompetansen i organisasjonen for alle brukere av Juvet.

## Lagre til senere

Funksjonen «lagre til senere» er en viktig del av det å kunne legge en aktivitet til på sin profilside for å kunne ta en kikk på den senere. For eksempel for et fremtidig kurs som virker interessant, men som brukeren på det aktuelle tidspunktet ikke ønsker å ta stilling til om vedkomne skal delta på eller ikke. Funksjonen kan også lagre for eksempel en ressurs til fremtidig bruk ved å trykke «lagre til senere» på detalj-siden for ressursen.

De lagrede aktivitetene kan sees i en liste på brukers «min side».

## Påmeldinger

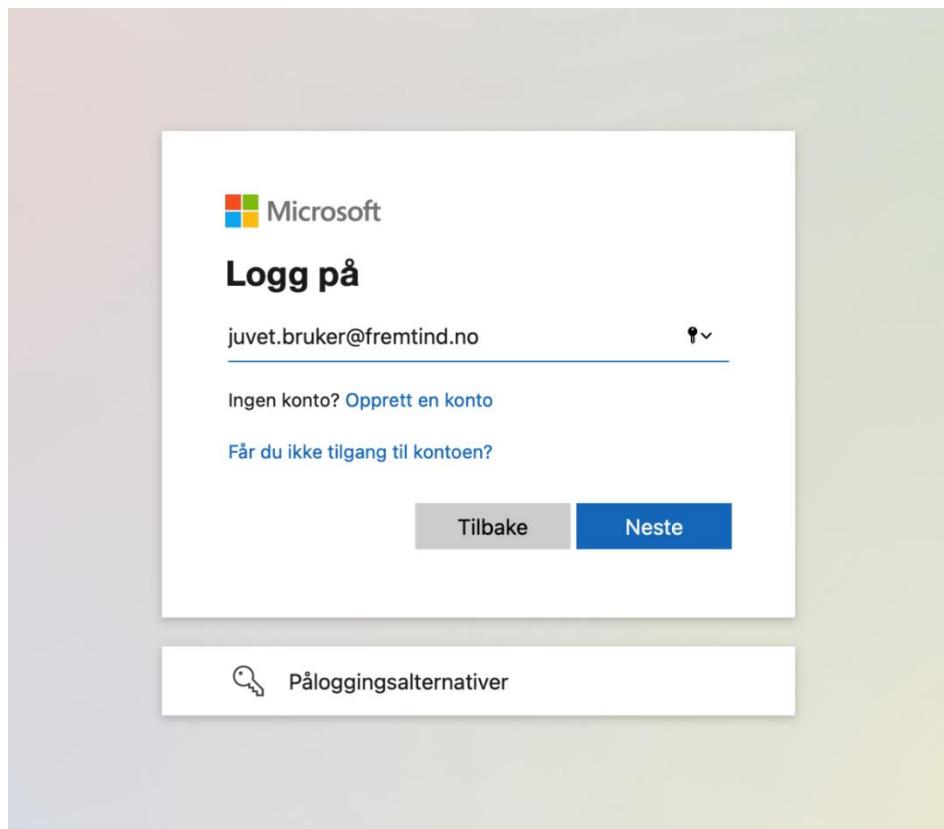
Mange av aktivitetene i Juvet vil være hendelsesbaserte aktiviteter, som møter, kurs, konferanser og lignende, som krever en påmelding. Det er implementert i Juvet slik at alle brukere kan se hvor mange som har meldt seg på den aktuelle aktiviteten.

## 6. Brukerveiledning

Løsningen er laget for å være så selvfklarende og intuitiv som mulig, men for å gjennomgå alle visninger og muligheter følger det en brukerdokumentasjon med skjermbilder og forklaringer. Juvet består av kun én brukerrolle, men alle brukere får opp innhold basert på sin egen brukerprofil med registrerte interesser. Det betyr at ved innlasting av landingssiden, og ved søk på aktiviteter, vil aktiviteter som treffer best på brukerens interesser komme lengst opp i resultatet.

### Autentisering og adgangskontroll med AzureAD/SSO

- Innlogging på løsningen foregår via AzureAD sin autentiseringsløsning. Dersom du som bruker går inn på Juvets nettadresse «juvet.fremtind.no», vil du automatisk bli videresendt til Microsoft Azure sin innloggingsportal.
- Alle Fremtind-ansatte har allerede en konto i AzureAD og vil dermed kunne logge inn på Juvet.
- Du kan bli spurta om å autentisere din identitet med to-faktur-autentisering via Microsofts Authenticator-app under innloggingssekvensen.
- Etter vellykket pålogging vil du bli sendt tilbake til «juvet.fremtind.no» og komme inn på landingssiden.
- Alle endepunkter og sider i systemet ligger bak innlogging, slik at du som bruker uten gyldig pålogging eller ved at påloggingssesjonen har utgått, vil bli bedt om å logge inn på nytt.



Figur 43 – Skjermbilde fra Juvets pålogging: Påloggingsportal fra Microsoft AzureAD for å komme inn på "juvet.fremtind.no"

## 6.1 Landingssiden

- Dersom du har en gyldig påloggingssesjon, vil du komme rett inn på landingssiden til Juvet.
- På langinssiden ser du et inputfelt med veiledingsteksten «*Hva vil du lære i dag?*» hvor du har mulighet til å søke på den kompetansen du ønsker å oppsøke blant aktiviteter og kolleger.
- Aktivitetskortene på landingssiden viser hvilken type aktivitet det dreier seg om, med bilde, tittel, kort beskrivelse og dato for når aktiviteten foregår. Det er tre ulike visninger av aktiviteter på landingssiden (se figur 44); «*Basert på dine interesser*», «*Populært i Fremtind nå*» og «*Alt innhold frem i tid*». «*Basert på dine interesser*» viser de aktivitetene i Juvet som stemmer best overens med de interesse-emnene som du vil kunne definere på din profilside. «*Populært i Fremtind nå*» viser de aktivitetene frem i tid som har flest antall påmeldte. Og «*Alt innhold frem i tid*» viser alle aktiviteter som har en definert dato og ligger i fremtiden. Under alle tre aktivitetvisningene er en knapp som i konseptet vil ta deg videre til en utfyllende liste med de kriteriene som visningen har, men disse sidene er ikke implementert.

The screenshot shows the Juvet landing page. At the top, a large header reads "Velkommen til Juvet, kompetanseplattformen av og for Fremtindere". Below it is a search bar with the placeholder "Hva vil du lære i dag?" and a "Søk" button. The main content area is titled "Basert på dine interesser" and displays four activity cards:

- En grundigere titt på Java**: An icon of a laptop with code and coffee. Description: "Hvis du er fan av Java eller interessert i hvordan...". Type: Presentasjon, Date: 8. jan. 24 - 12.
- Nybegynner til Spring Boot**: An icon of a green hexagon with a white power symbol. Description: "Lorem ipsum dolor sit amet consectetur adipisicing elit.". Type: Kurs, Date: 12. aug. - 09.
- Dypdykk i JVM-parametere**: An icon of a Java logo with a coffee cup. Description: "Lorem ipsum dolor sit amet consectetur adipisicing elit.". Type: Artikkel, Date: 5t 30 min.
- Closures i Java**: An icon of a laptop with JS and Java code. Description: "Lorem ipsum dolor sit amet consectetur adipisicing elit.". Type: Kurs, Date: 19. aug. - 09.

At the bottom left, there is a link "Se alle aktiviteter →".

Figur 44 – Skjermbilde fra Juvet: Landingssiden

## Landingssiden aktivitetskontekst

Basert på dine interesser



### Dypdykk i JVM-parametere

Lorem ipsum dolor sit amet consectetur...

Aktivitet

5t 30 min



### Nybegynner til Spring Boot

Lorem ipsum dolor sit amet consectetur...

Aktivitet

12. aug. - 09



### Closures i Java

Lorem ipsum dolor sit amet consectetur...

Aktivitet

19. aug. - 09



### Closures i C#

Lorem ipsum dolor sit amet consectetur...

Aktivitet

13. okt. - 09

[Se alle aktiviteter →](#)

Populært i Fremtind nå



### Nybegynner til Spring Boot

Lorem ipsum dolor sit amet consectetur...

Aktivitet

12. aug. - 09



### Dypdykk i JVM-parametere

Lorem ipsum dolor sit amet consectetur...

Aktivitet

5t 30 min



### Closures i Java

Lorem ipsum dolor sit amet consectetur...

Aktivitet

19. aug. - 09



### Closures i C#

Lorem ipsum dolor sit amet consectetur...

Aktivitet

13. okt. - 09

[Se alle aktiviteter →](#)

Alt faginnhold frem i tid



### Nybegynner til Spring Boot

Lorem ipsum dolor sit amet consectetur...

Aktivitet

12. aug. - 09



### Dypdykk i JVM-parametere

Lorem ipsum dolor sit amet consectetur...

Aktivitet

5t 30 min



### Closures i Java

Lorem ipsum dolor sit amet consectetur...

Aktivitet

19. aug. - 09



### Closures i C#

Lorem ipsum dolor sit amet consectetur...

Aktivitet

13. okt. - 09

Figur 45 - Skjermbilde fra Juvet: Kontekstbasert presentasjon av aktiviteter

## Søkesiden

- Du vil kunne komme inn på søkesiden ved å skrive inn et tema i søkefeltet på landingssiden og trykke *linjeskift* eller ved å trykke knappen «Søk» ved inputfeltet.

A screenshot of a search bar interface. On the left, the word "Juvet" is displayed. To its right is a rectangular input field containing the placeholder text "Hva vil du lære i dag?". To the right of the input field is a dark blue button with the white text "Søk" (Search). Further to the right is a link labeled "Min side" (My page) underlined in blue.

Figur 46 - Skjermbilde fra Juvet: Søke-linjen

## 6.2 Søkeresultat

- I søkeresultatet ser du de treffene som er gjort på ditt søkeord blant aktiviteter og eksperter i Juvet.
- Øverst ser du de ekspertene som har best treff på søkeordet. Du kan ta direkte kontakt med en ekspert ved å trykke «Kontakt [Navn Navnesen] på Teams», og du vil da bli videresendt til din Teams-klient for å starte en chat/samtale med eksperten. Ekspertene kan selv velge hvilken kanal de ønsker å være tilgjengelig på; teams, telefon eller epost.
- Videre ser du en liste med de aktivitetene i Juvet som treffer med søkeordet.
- Søket baserer seg på din input i tekstfeltet, og du kan søke på tittel eller emner.

The screenshot shows the Juvet search results page. At the top, there is a search bar with the word "java" and a "Søk" button. Below the search bar, the text "Vi fant 5 treff på java i Juvet" is displayed. Under this, there is a section titled "Personer med kompetanse på java" showing two profiles: Tom S. (Leader) and Mathias S. (Utvikler), each with a photo and a "Kontakt" button. Below this, a link "Se alle personer med denne kompetansen →" is shown. The main content area is titled "Søketreff for java" and contains two cards. The first card is for "En grundigere titt på Java" by Nina, featuring a laptop icon and a presentation thumbnail. The second card is for "Closures i JavaScript" by Nina and Tom, featuring a laptop icon and a course thumbnail. Both cards have "Se detaljer" and "Lagre til senere" buttons.

java

Søk

## Vi fant 5 treff på **java** i Juvet

### Personer med kompetanse på **java**

Tom S.  
Leader  
[Kontakt Tom på Teams](#)

Mathias S.  
Utvikler  
[Kontakt Mathias på Teams](#)

Se alle personer med denne kompetansen →

### Søketreff for **java**

**En grundigere titt på Java**  
Hvis du er fan av Java eller interessert i hvordan avanserte Java applikasjoner fungerer; bli med!  
av [Nina](#)

[Se detaljer](#) [Lagre til senere](#)



**Closures i JavaScript**  
Lorem ipsum dolor sit amet consectetur adipisicing elit.  
av [Nina](#) og [Tom](#)

[Se detaljer](#) [Lagre til senere](#)



Figur 47 – Skjermbilde fra Juvet: Søkeresultat-siden

### 6.3 Aktivitet detaljert

- Du kommer inn på den detaljerte oversikten for en aktivitet ved å trykke «Se detaljer» eller trykke på selve aktivetskortet.
- Du ser forklaring på inndelingen av innholdet på siden i figuren under. Aktivitetsdetaljer og knappenes funksjoner kan du se senere i dette delkapittelet.

Tittel	<p>Juvet</p> <p>Hva vil du lære i dag?</p> <p>Søk</p> <p>Min side</p>	Aktivitetsbilde
Kort beskrivelse	<p>← Tilbake</p> <h2>En grundigere titt på Java</h2> <p>Hvis du er fan av Java eller interessert i hvordan avanserte Java applikasjoner fungerer; bli med!</p> <h3>Vi studerer de mer avanserte oppsettene for objekt-orientert programmering i Java!</h3> <p>Her er alle Java-interesserte velkomne til en oversiktlig og informativ presentasjon om hvordan vi kan lære nye ting fra et gammelt språk. Det kommer til å bli servert boller til de første 20 som møter opp, her blir både med- og uten rosiner tilstede.</p> <p>Serveringene stopper ikke med boller, ettersom det blir servert mange gode tips til hvordan du kan bli en mester i programmering med Java.</p> <p>Vi kommer til å snakke om Servlet, JSP, JDBC og avansert Socket programering.</p>	 <p>Meld på</p> <p>Lagre til senere</p> <p><b>Presentasjon</b> av <u>Tom Tom</u></p> <p><b>Tid</b> 3. juni • 14 - 15 Ingen deltakere enda Bli den første til å melde deg på!</p> <p><b>Sted</b> <i>Fremtind</i> Rom 7F i 7. etasje</p> <p><b>Emner</b></p>
Lang beskrivelse (markdown)	<h2>Fremtind</h2>	

Figur 48 – Skjermbilde fra Juvet: Aktivitet detaljert-siden

## Aktivitet detaljert påmelding registrert

- Når du melder deg på vil du ved vellykket påmelding få denne beskjeden.

The screenshot shows a web page from Juvet. At the top, there is a navigation bar with 'Juvet' on the left, a search bar containing 'Hva vil du lære i dag?', a 'Søk' button, and 'Min side' on the right. Below the navigation is a backlink '← Tilbake'. The main content area features a large heading 'En grundigere titt på Java'. Below it is a subtext: 'Hvis du er fan av Java eller interessert i hvordan avanserte Java applikasjoner fungerer; bli med!'. To the right of the text is an illustration of a laptop displaying code and a coffee cup. Below the illustration are two buttons: 'Meld av' (highlighted with a green border) and 'Lagre til senere'. Further down, there is information about the presentation: 'Presentasjon' by 'Tom Tom', 'Tid' (3. juni 14 - 15), and '1 deltagere'. A prominent blue box at the bottom contains the message 'Påmelding registrert!' with an info icon and a close 'X' button.

En grundigere titt på Java

Hvis du er fan av Java eller interessert i hvordan avanserte Java applikasjoner fungerer; bli med!

Vi studerer de mer avanserte oppsettene for objekt-orientert programmering i Java!

Her er alle Java-interesserte velkomne til en oversiktig og informativ presentasjon om hvordan vi kan lære nye ting fra et gammelt språk. Det kommer til å bli servert boller til de første 20 som møter opp, her blir både med- og uten rosiner tilstede.

Serveringene stopper ikke med boller, ettersom det blir servert mange andre ting til hvertandre du kan bli interessert i programvareingen med. Du

Meld av Lagre til senere

Presentasjon  
av Tom Tom

Tid  
3. juni 14 - 15

1 deltagere

Påmelding registrert!

Figur 49 - Skjermbilde fra Juvet: Aktivitet detaljert-siden ved registrert påmelding

## Aktivitet detaljert avmelding registrert

- Når du melder deg av vil du ved vellykket avmelding få denne beskjeden.

The screenshot shows a web page from Juvet. At the top, there is a navigation bar with 'Juvet' on the left, a search bar containing 'Hva vil du lære i dag?', a 'Søk' button, and 'Min side' on the right. Below the navigation is a backlink '← Tilbake'. The main content features a large heading 'En grundigere titt på Java'. Below it is a subtext: 'Hvis du er fan av Java eller interessert i hvordan avanserte Java applikasjoner fungerer; bli med!'. To the right of the text is a graphic of a laptop displaying code and a coffee cup. Below the graphic are two buttons: 'Meld på' (highlighted with a green border) and 'Lagre til senere'. Further down, there is information about the presentation: 'Presentasjon av Tom Tom', 'Tid 3. juni • 14 - 15', and 'Ingen deltakere enda'. A blue box at the bottom contains the message 'Avmelding registrert!' with an info icon and a close 'X' button.

En grundigere titt på Java

Hvis du er fan av Java eller interessert i hvordan avanserte Java applikasjoner fungerer; bli med!

Vi studerer de mer avanserte oppsettene for objekt-orientert programmering i Java!

Her er alle Java-interesserte velkomne til en oversiktlig og informativ presentasjon om hvordan vi kan lære nye ting fra et gammelt språk. Det kommer til å bli servert boller til de første 20 som møter opp, her blir både med- og uten rosiner tilstede.

Serveringene stopper ikke med boller, ettersom det blir servert mange gode tips til hvordan du kan bli en mester i programmering med Java.

Meld på Lagre til senere

Presentasjon av Tom Tom

Tid 3. juni • 14 - 15

Ingen deltakere enda

Avmelding registrert!

Figur 50 - Skjermbilde fra Juvet: Aktivitet detaljert-siden ved registrert avmelding

## Aktivitet detaljert lagret til senere

- Når du lagrer aktiviteten til senere vil du ved vellykket lagring få denne beskjeden.

The screenshot shows a user interface for saving an activity. At the top, there's a navigation bar with 'Juvet', a search bar containing 'Hva vil du lære i dag?', a 'Søk' button, and a 'Min side' link. Below the navigation is a back-link '← Tilbake'. The main content area features a large title 'En grundigere titt på Java'. Below the title is a descriptive text: 'Hvis du er fan av Java eller interessert i hvordan avanserte Java applikasjoner fungerer; bli med!'. To the right of this text is a blue laptop icon with code and coffee on its screen. Below the laptop are two buttons: 'Meld på' (black background) and 'Fjern fra senere' (green border). Further down, there's a section for a presentation: 'Presentasjon' by 'Tom Tom' (with a link), 'Tid' (3. juni • 14 - 15), and a note 'La til aktiviteten i ditt innhold →' with a green 'Sted' button. A small info icon is also present.

Juvet

Hva vil du lære i dag?

Søk

Min side

← Tilbake

# En grundigere titt på Java

Hvis du er fan av Java eller interessert i hvordan avanserte Java applikasjoner fungerer; bli med!

Vi studerer de mer avanserte oppsettene for objekt-orientert programmering i Java!

Her er alle Java-interesserte velkomne til en oversiktlig og informativ presentasjon om hvordan vi kan lære nye ting fra et gammelt språk. Det kommer til å bli servert boller til de første 20 som møter opp, her blir både med- og uten rosiner tilstede.

Serveringene stopper ikke med boller, ettersom det blir servert mange gode tips til hvordan du kan bli en mester i programmering med Java.

Meld på

Fjern fra senere

Presentasjon  
av Tom Tom

Tid  
3. juni • 14 - 15

La til aktiviteten i ditt innhold →

Sted

Figur 51 - Skjermbilde fra Juvet: Aktivitet detaljert-siden ved vellykket lagring til senere

## Aktivitet detaljert-info

The screenshot shows a summary of an activity. At the top right are two buttons: 'Meld på' (dark background) and 'Lagre til senere' (light background). Below these are sections for 'Aktivitet' (Activity), 'Tid' (Time), 'Sted' (Place), and 'Emner' (Topics). The 'Aktivitet' section lists 'av Mathias Mathias og Nina Nina'. The 'Tid' section shows '12. august • 09 - 13:40'. The 'Sted' section shows 'Ingen deltakere ennå' (No participants yet) and 'Bli den første til å melde deg på!' (Be the first to register!). The 'Emner' section shows 'spring boot' and 'java' as selected topics.

Figur 52 - Skjermbilde fra Juvet: Aktivitetsdetaljer for aktivitet

**«Meld på» og «Lagre til senere»:** Knapper for å melde deg på aktiviteten, eller å lagre den til å se senere.

**«Aktivitet»:** Viser hvilken type aktivitet det handler om. Kan også være «fagtimen», kurs, etc.

**Av:** Personene som er nevnt her er forfattere av aktiviteten.

**Tid:** Viser tidspunktet aktiviteten foregår. I dette eksempelet vil aktiviteten skje 12.august kl. 0900 – 1340.

**Deltakere:** Viser antall påmeldte til den aktuelle aktiviteten. Hvis ingen er påmeldt vises teksten «Ingen deltakere ennå», slik som i dette eksempelet.

**Sted:** Viser hvor aktiviteten foregår. Kan være fysisk, digital eller hybrid. En fysisk aktivitet har kun en fysisk adresse, en digital aktivitet har kun en digital adresse (eks. Teams-lenke, mens en hybrid aktivitet har både fysisk og digital adresse).

**Emner:** Viser hva forfatteren(e) har merket aktiviteten til å omhandle.

#### 6.4 Min side

- Du kommer inn på din side, ved å trykke på «Min side» i navigasjonsbaren øverst på nettsiden.
- Aktivitetene du har lagret til senere vil listes opp her.
- Her vises brukeren du er innlogget som og knapp for å gå videre til å opprette en ny aktivitet i Juvet.

# Velkommen til min side!

Du er logget inn som mats.sommervold@gmail.com

Velkommen til din side. Den er ikke helt ferdig enda, men du kan prøve å opprette en aktivitet!

[Opprett Aktivitet](#)

## Dine lagrede aktiviteter

### En grundigere titt på Java

Hvis du er fan av Java eller interessert i hvordan avanserte Java applikasjoner fungerer; bli med!

av [Nina Nina](#) og [Tom Tom](#)

[Se detaljer](#)

[Fjern fra senere](#)



Presentasjon - Fremtind  
8. jan. 24 - 12

Figur 53 – Skjermbilde fra Juvet: Min side

### **Opprett aktivitet – valg av aktivitetsformat**

- Du kommer inn på «Opprett aktivitet»-siden ved å trykke deg inn på «Min side» og så «Opprett aktivitet».
- Her må du velge om aktiviteten din har en tid og sted for hendelsen, eller om det er en ressurs, som ikke trenger å bli opprettet med tid og sted.

## Opprett en ny aktivitet

Aktivitetene på juvet er av og for Fremtindere. Her skal det være lav terskel for å dele med hverandre!

### Detaljer

Aktiviteten min har  
en tid og dato→

Aktiviteten min er kun  
en digital ressurs→

Figur 54 – Skjermbilde fra Juvet: Opprett aktivitet-side, valg av aktivitetsformat

**Opprett aktivitet – valg mellom fysisk, digital eller hybrid aktivitet**

- Her må du velge om aktiviteten din skal være en fysisk, digital eller hybrid hendelse.

## Format

Hvilke format skal aktiviteten ha?

Fysisk

Hybrid

Digitalt

← Tilbake

Neste →

Figur 55 - Skjermbilde fra Juvet: Opprett aktivitet-side, valg mellom fysisk, digital eller hybrid aktivitet

## Opprett aktivitet – utfylt beskrivelse

- For å opprette en aktivitet, må du fylle ut alle feltene i henhold til tittelen på feltet.

The screenshot shows a form for creating an activity. On the left, there are three input fields: 'Tittel' (Title) containing 'Java for Nybegynnere', 'Kort beskrivelse' (Short description) containing 'Dette kurset er for nybegynnere i java.', and 'Bildeblink' (Image link) containing 'https://fremtind.no/bilde'. On the right, there is a larger field labeled 'Lang beskrivelse (Markdown)' which contains two paragraphs of text: 'Vi vil gjennomgå java steg for steg fra begynnelsen.' and 'Dette kurset vil gi grunnlag for videre utvikling av dine'.

Figur 56 - Skjermbilde fra Juvet: Eksempel på utfylt opprett-aktivitet-skjema

## Opprett aktivitet – utfylt detaljer

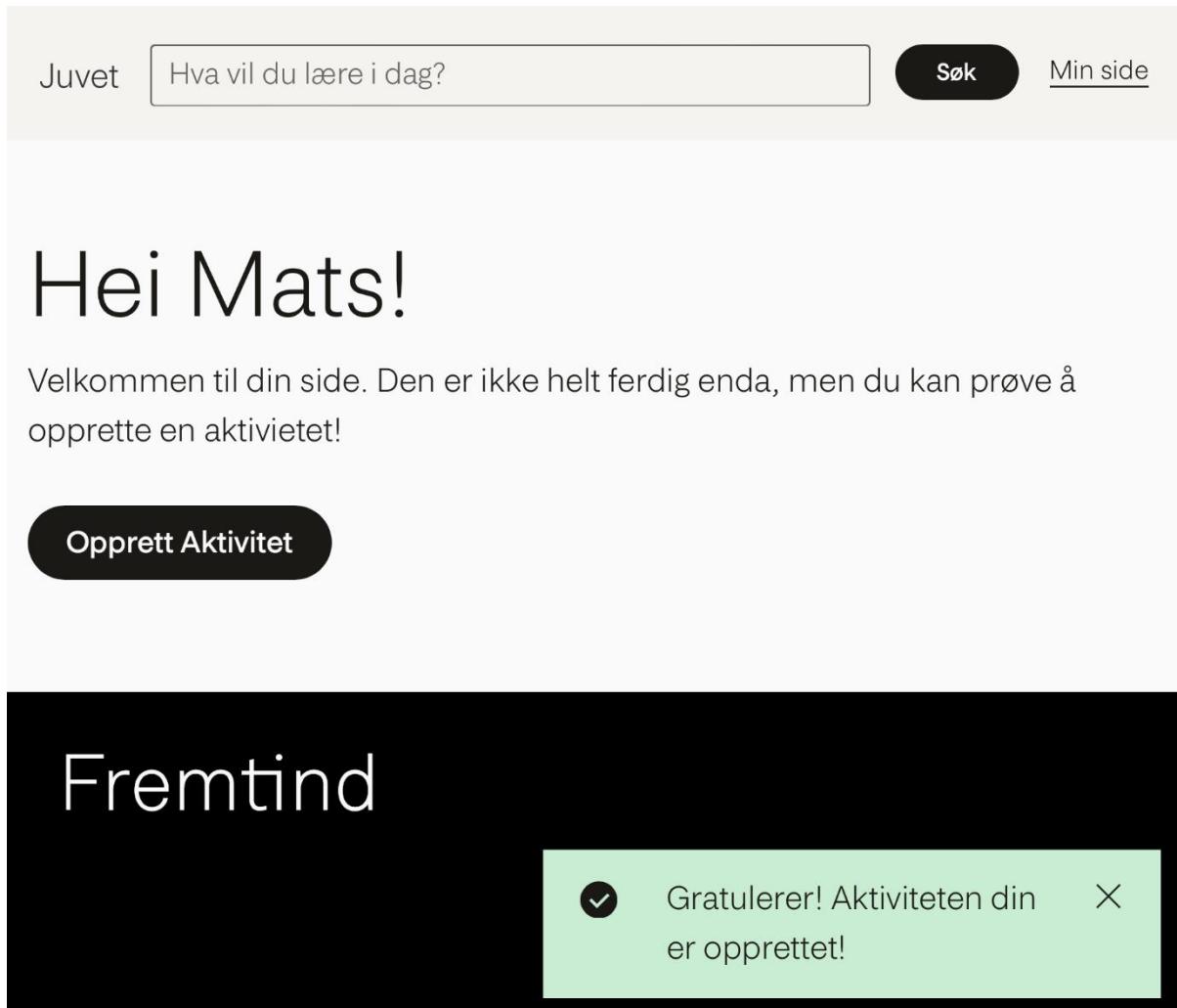
- For å opprette en aktivitet, må du fylle ut alle feltene i henhold til tittelen på feltet.
- Siden dette er en hybrid aktivitet, må alle feltene fylles ut.

The screenshot shows a form for creating an activity with detailed information. It includes sections for 'Tid og sted' (Time and place), 'Digital adresse' (Digital address), and 'Fysisk adresse' (Physical address). In the 'Digital adresse' section, there is a dropdown menu 'Digital platform' with 'Velg' selected and a 'Digital link' input field. In the 'Fysisk adresse' section, there are fields for 'Organisasjon (valgfritt)', 'Adresse' (Address), 'Veinummer' (House number), 'Postnummer' (Postal code), and 'By' (City). At the bottom, there are buttons for '← Tilbake' (Back) and 'Opprett Aktivitet' (Create Activity).

Figur 57 - Skjermbilde fra Juvet: Ufylling av detaljer for hybrid aktivitet

## Opprett aktivitet - suksess

- Når du har suksessfullt opprettet en ny aktivitet blir du videresendt til detalj-siden for den nye aktiviteten din
- Du får samtidig en meldingsboks med bekrefteelse om at aktiviteten er opprettet.



Figur 58 - Skjermbilde fra Juvet: Bekrefteelse på suksessfull opprettelse av ny aktivitet

Komplett oversikt og krav for opprettelsen av ny aktivitet sees her:

Tabell 6 - Overisk og krav for de ulike type aktivitetene i Juvet

	Har aktiviteten en tid og dato?	Fysisk adresse	Digital adresse	Dato og tid	Format
Ressurs	Nei	Nei	Ja	Nei	Nei
Fysisk aktivitet	Ja	Ja	Nei	Ja	Fysisk
Digital aktivitet	Ja	Nei	Ja	Ja	Digital
Hybrid aktivitet	Ja	Ja	Ja	Ja	Hybrid

## 6.5 Feilmeldinger

Ved kritiske feil, som at tilkoblingen mellom frontend-tjeneren og backend-tjeneren ikke fungerer som den skal, eller ved forsøk på å gå videre på funksjoner som ikke er implementert, vil en feilmelding vises. Feilmeldinger ved validering av tekst-input er beskrevet under «Min side» hvor opprettelse av aktivitet skjer. Alle feilmeldinger i eksemplene under er markert med en grønn ramme for å tydeliggjøre meldingen.

### 6.5.1 Funksjoner som ikke er implementert

#### På søkeresultat-siden

- Ved å trykke på «Se alle personer med denne kompetansen» får du denne feilmeldingen.

Personer med kompetanse på **java**

**Tom S.**  
Leader



[Kontakt Tom på Teams](#)

**Mathias S.**  
Utvikler



[Kontakt Mathias på Teams](#)

Se alle personer med denne kompetansen →

Søketreff for **java**

**Closures i JavaScript**

**!** Denne funksjonaliteten er desverre ikke implementert ennå X

Figur 59 - Skjermbilde fra Juvet: Feilmelding ved å se flere personer med kompetanse

## På aktivitet detaljert-siden

- Ved å trykke på emnet «jvm» får du denne feilmeldingen.

The screenshot shows a Java activity detail page. At the top is the Java logo. Below it is a button labeled "Lagre til senere". The activity details are as follows:

- Aktivitet**: av Mathias Mathias og Nina Nina
- Lengde**: 5 timer og 30 minutter
- Ingen deltakere enda**
- Emner**: **jvm**

A yellow warning box with a green border is overlaid on the page, containing the text: "Du kan desverre ikke søke på tags ennå" (You can unfortunately not search for tags yet) with an exclamation mark icon. There is also a close button (X) in the top right corner of the box.

Figur 60 - Skjermbilde fra Juvet: Feilmelding ved å trykke på emne-tag

### 6.5.2 Nettverksproblemer

#### Nettverksproblemer på landingssiden

- Hvis du av en eller annen grunn ikke har kontakt med backend-tjenesten, vil disse feilmeldingene vises på landingssiden

Basert på dine interesser

Network Error

Se alle aktiviteter →

Populært i Fremtind nå

Network Error

Se alle aktiviteter →

Alt faginnhold frem i tid

Network Error

Se alle aktiviteter →

Figur 61 - Skjermbilde fra Juvet: Landingssiden ved nettverksproblemer

#### Nettverksproblemer på søkeresiden

- Hvis du av en eller annen grunn ikke har kontakt med backend-tjenesten, vil denne feilmeldingene vises på søkeresiden

Juvet

java

X

Søk

Min side

Obs! Her skjedde det en noe galt

Er du koblet til internet?

Fremtind

Figur 62 - Skjermbilde fra Juvet: Feilmelding på søkeresultat-siden dersom backend-tjener ikke svarer

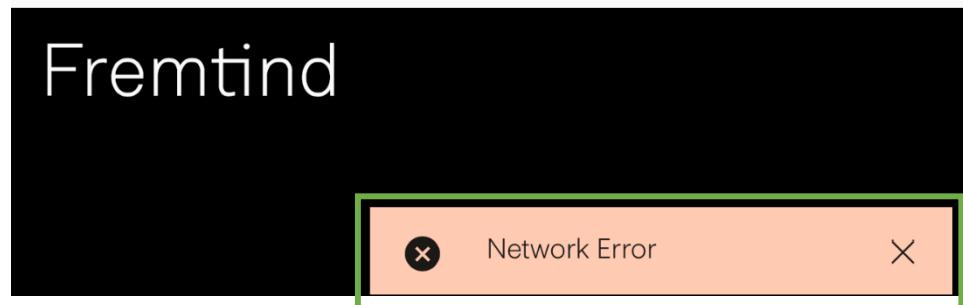
### **Nettverksproblemer ved opprettelse av aktivitet**

- Hvis du av en eller annen grunn ikke har kontakt med backend-tjenesten ved opprettelse av en aktivitet, vil denne feilmeldingen vises.

Hei Mats!

Velkommen til din side. Den er ikke helt ferdig ennå, men du kan prøve å opprette en aktivitet!

**Opprett Aktivitet**



Figur 63 - Skjermbilde fra Juvet: Opprett aktivitet ved nettverksproblemer

## 7. Konklusjon og drøfting

Etter avsluttet utvikling og leveranse til oppdragsgiver vil vi oppsummere prosjektet. Vi vil nevne hva utbyttet av prosjektet har vært og hvilken verdi det har ført til, både for oppdragsgiver og oss som prosjektgruppe. Samtidig vil produktets status og dets fremtid beskrives.

### 7.1 Utbytte

I tillegg til å være en god inngang til arbeidslivet som utviklere, har dette prosjektet gitt hele gruppen et mye tydeligere bilde av prosessene innen programvareutvikling i praksis. Ved å jobbe tett på med ansatte i Fremtind, og i tillegg ta i bruk deres designsystemer, har vi følt oss- og fungert som et utviklingsteam. Vi har, ikke minst, fått lære om systemer som tidligere har vært ukjent for flere av oss. For eksempel oppsøkte vi kunnskap og prøvde oss frem innen i AWS, AzureAD, OpenID Connect og Jøkul.

Ved å ta hensyn til hverandres styrker og svakheter, gi rom for feil, og ved å bygge tillit til hverandres ferdigheter har vi fått en ny forståelse for hvordan man jobber mer effektivt som et team. Det kan allikevel bli uforutsigbart hvordan arbeidsmengden i et gruppeprosjekt av denne størrelsen blir balansert, noe vi fikk føle på. Det ble i løpet av prosjektet bevist litt etter litt at den tidlige ansvar- og oppgavefordelingen, i samsvar med regelmessige standup-møter, ga hver av oss tillit til hverandres kunnskap, motivasjon, og interesse i oppgaven. Store deler av motivasjonen og interessen i oppgaven var basert i tanken om at resultatet av prosjektet vil bli tatt aktivt i bruk av Fremtind.

Vi som prosjektgruppe er meget fornøyde med vår innsats. Selv om vi kunne og ville ha gjort mer for å utfylle applikasjonens potensiale om vi hadde mer tid til gode, kan vi si oss stolte med hvor mye vi har gjort. Vi oppnådde en måloppnåelse på rundt 85% av våre kravspesifikasjoner og endte dermed opp med en løsning av høy verdi for både oss og Fremtind.

Startskuddet designsprinten ga oss for utforming av konseptet til oppgaven, ble vi også gjort oppmerksom på hvor betydningsfull en slik intern kompetanseplattform potensielt vil være for Fremtind. Med dagens ikke-prioriterte systemer for intern faglig utvikling, vil vår løsning kunne åpne dørene for bruker-etterspurte endringer som vil gjøre et slikt internt system attraktivt, effektivt og resultatrikt.

### 7.2 Hva er produktets bruks- og nytteverdi?

Det er, basert på tilbakemeldinger fra ansatte i Fremtind gjennom brukertestene våre, muligheter for at Juvet kan spille en rolle i den faglige utviklingen til Fremtind som helhet.

Ikke bare vil all intern lærdom ha ett sentralt møtepunkt i vår applikasjon, men muligheten til å ta direkte kontakt med kolleger vil også kunne oppmuntre til større fellesskap og engasjement for hverandres faglige utvikling. Dette er ifølge vår kontaktperson i Fremtind en høyt etterspurtt endring som kan ha positiv effekt på de ansattes kompetanse.

Med så store muligheter for å gi Fremtind et nyttig verktøy, har vi jobbet så godt vi kunne for å gi bedriften den letteste mulige veien til applikasjonens lansering. Innad i gruppen, med bekreftelse fra kontaktperson i Fremtind, har vi innsett at dette potensielt kan påvirke bedriften i positiv grad, som har bidratt til å gi oss motivasjon til å levere et så godt prosjekt som mulig. Mer om videreutviklingen av Juvet kan leses om i [delkapittel 7.5](#).

### 7.3 Hva ville vi ha gjort annerledes?

Dersom vi skulle ha startet på dette prosjektet på nytt med den erfaringen som vi sitter igjen med i etterkant, hadde vi gjort noen endringer i måten vi løste dette på. Først og fremst gjelder dette de utfordringene som vi har brukt mest tid på å løse. Dette gjelder blant annet for autentiseringen, som ble en solid nøtt underveis i prosjektet. Utviklingen av autentiseringsdelen av løsningen ble startet på tidlig i utviklingsfasen, men satt på vent i påvente av tilganger fra Fremtind. Underveis fikk vi også mange timers hjelp fra en av oppdragsgivers it-arkitekter, og det løste seg til slutt. I ettertid ser vi at dette burde ha blitt prioritert tidligere, særlig siden det var en såpass viktig del av løsningen.

Innlasting av databasen ble en uforutsett utfordring som vi innså for sent i prosjektperioden til å rekke å endre noe på. Med den erfaringen som vi i etterkant sitter med, ville vi ha modellert lasting av data fra databasen på en annen måte enn det som ble gjort for denne løsningen.

Oppgavefordelingen innad blant gruppemedlemmene var innledningsvis tiltenkt å kunne være mer dynamisk, hvor alle kunne bidra mer på flere områder. Dette viste seg å etter hvert bli mer «låst» da den enkelte hadde opparbeidet seg såpass med erfaring og innsikt i hvordan «sin del» av prosjektet hadde utviklet seg. Årsaken til dette var at personene i gruppa satt på ulik kunnskap, som gjorde at det ikke var naturlig eller hensiktsmessig å dele mer på oppgavene enn det som ble gjort. Dette antar vi er vanskelig å gjøre så mye med for å unngå, men forventningene våre til neste prosjekt av lignende type vil være ulikt.

### 7.4 Tilbakemelding fra oppdragsgiver

Tilbakemeldingene fra oppdragsgiver har gjennom hele prosjektet vært utelukkende positive. Under designsprint ble oppdragsgiver positivt overrasket over hvor effektive og fokuserte vi var underveis i prosessen. Vi fikk høre at vi raskt forsto konseptet med designsprint, brukte verktøyene som ble gitt på en god måte og tok raske valg med gjennomtenkte begrunnelser. Dette skapte tidlig et positivt og givende klima mellom oppdragsgivers kontaktpersoner i prosjektet og oss i bachelorgruppa.

Videre i prosjektet var oppdragsgiver sterkt engasjert i oppgaven i de ukentlige møtene og vi opplevde stor støtte, uansett om utfordringene i utviklingen var positive eller om de var begrensende. Ved utfordringer som var til hinder ble vi tilbuddt drahjelp fra Fremtind sine egne utviklere, it-arkitekter eller interaksjonsdesignere.

Prosjektgiver har gitt en attest for arbeidet vi gjennomførte i prosjektperioden. Denne kan sees i sin helhet i [vedlegg 10](#).

### 7.5 Status for videre utvikling og produksjonssetting av produktet

Juvet er fortsatt i en tidlig fase av sin levetid og vil derfor bli ytterligere testet og forbedret av Fremtind sine utviklere etter at det er levert til oppdragsgiver. Oppdragsgiver ser for seg å tilby et begrenset antall ansatte tilgang til Juvet for å la de bruke- og utforske løsningen i løpet av dens første aktive periode. Dersom dette går som ønsket vil brukergruppen som har tilgang til Juvet skaleres opp slik at flere kan bruke det, og det etterhvert komme i full produksjon, som er målet. Et av hovedmålene med dette konseptet var at en ansatt i Fremtind skulle føle at alle sine kolleger i organisasjonen kunne være med å bidra til den ansattes kompetanseheving. Noe vi har fått bekreftet at Juvet innfrir via svar i våre brukertester.

Juvet har sine forbedringspotensialer og kommer til å videreutvikles av Fremtind for å kunne settes i full produksjon. Vi samlet alle oppgavene som omhandlet forbedring og videreutvikling i vårt kanban-prosjektbrett som oppdragsgiver hadde tilgang til.

For videreutvikling av produktet er det noen aspekter som ikke ble vektlagt i like stor grad som det helst burde på grunn av tidsrammene som var gitt, og gruppas kapasitet til å håndtere disse. I hovedsak dreier dette seg om testing av systemet, som ideelt sett skulle ha vært gjort på en mer grundig og helhetlig måte. Videre gjelder også å sørge for at løsningen er så godt universelt utformet som mulig. Selv om løsningen scorer godt innen WCAG-tester, er det noen feil som burde ha vært løst, som vi ikke kunne prioritere.

Mer konseptuelt ser vi for oss en løsning på kravet om å «gamifye» løsningen ved å bruke spillteori og mulighet for å samle belønninger til en privat samling på sin Juvet-profil. Dette var det noe blandede, men likevel stort sett positive tilbakemeldinger på under brukertest av prototypen. Samtidig anså vi dette som en for stor oppgave å skulle løse innen den tiden som var til rådighet. Blant tilbakemeldingene på dette var det mange som så for seg å bli motivert av en slik funksjon og derfor sannsynligvis ville ha tilført løsningen ytterligere verdi for både oppdragsgiver og mange av løsningens brukere.

## 7.6 Oppsummering og konklusjon

Oppsummert vil vi si at prosjektet har vært gøy, utfordrende, omfattende og lærerikt. Spesielt det at vi var med å utvikle et helt nytt konsept som var en skreddersydd løsning for et behov som oppdragsgiver hadde. Å gjennomføre en designsprint var høydepunktet i problemløsningen hva gjaldt utvikling av konseptet.

Tre momenter som vi ønsker å trekke frem som gjorde dette prosjektet spesielt interessant og utfordrende; den store bredden i oppgaven - som har vært en utfordring i utviklingen av systemet, å komme opp med et konsept med en løsning for et åpent problem hos oppdragsgiver, og til slutt det å jobbe med en oppdragsgiver; hvor det å forholde seg til fullpakket kalendre hos kontaktpersoner og ressurser hos oppdragsgiver. Disse tre momentene hadde hver for seg både positive og negative sider.

Vi konkluderer derfor med at vi har lært og erfart massevis underveis, vi er stolte over konseptet og det vi fikk utviklet, men at vi skulle ønske at vi hadde kommet lengre med den fulle utviklingen av løsningen. Likevel var oppdragsgiver strålende fornøyd med oss hele veien – en støtte og bekrefteelse vi satte stor pris på.

Som nevnt i forordet, ble denne rapporten stor. Ettersom vi var 4 personer og hadde fått gjort et stort arbeid i prosjektgruppa følte vi det var naturlig at den gjenspeiler antallet personer og arbeidsmengden i prosjektet.

## Referanseliste

- Amazon Web Services. (n.d.). *Amazon ECS clusters*. Retrieved from amazon.com, sist besøkt 23.05.22: <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/clusters.html>
- Amazon Web Services. (n.d.). *Amazon Elastic Container Service*. Retrieved from amazon.com, sist besøkt 29.04.22: [https://docs.aws.amazon.com/AmazonECS/latest/developerguide/AWS\\_Fargate.html](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/AWS_Fargate.html)
- Amazon Web Services. (n.d.). *Cloud computing with AWS*. Retrieved from Aws.amazon.com, sist besøkt 02.05.22: [https://aws.amazon.com/what-is-aws/?nc1=f\\_cc](https://aws.amazon.com/what-is-aws/?nc1=f_cc)
- Amazon Web Services. (n.d.). *Create an Application Load Balancer*. Retrieved from amazon.com, sist besøkt 22.05.22: <https://docs.aws.amazon.com/elasticloadbalancing/latest/application/create-application-load-balancer.html>
- Amazon Web Services. (n.d.). *Getting started with Amazon RDS*. Retrieved from docs.aws.amazon.com, Sist besøkt 05.05.22: [https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP\\_GettingStarted.html](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.html)
- Amazon Web Services. (n.d.). *Getting started with IAM*. Retrieved from amazon.com, sist besøkt 23.05.22: <https://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started.html>
- Amazon Web Services. (n.d.). *Getting started with the classic console using Linux containers on AWS Fargate*. Retrieved from amazon.com, sist besøkt 23.05.22: <https://docs.aws.amazon.com/AmazonECS/latest/userguide/getting-started-fargate.html>
- Amazon Web Services. (n.d.). *Services*. Retrieved from amazon.com, sist besøkt 22.05.22: <https://docs.aws.amazon.com/AmazonECS/latest/userguide/what-is-fargate.html>
- Amazon Web Services. (n.d.). *Tasks*. Retrieved from amazon.com, sist besøkt 22.05.22: <https://docs.aws.amazon.com/AmazonECS/latest/userguide/what-is-fargate.html>
- Amazon Web Services, Inc. (n.d.). *Task networking with the awsvpc network mode*. Retrieved from amazon.com, sist besøkt 17.04.22: <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task-networking-awsvpc.html>
- Apache Maven Project. (n.d.). *What is Maven?* Retrieved from maven.apache.org, sist besøkt 08.05.22: <https://maven.apache.org/what-is-maven.html>
- Automation Panda. (2020, juli 7). *ARRANGE-ACT-ASSERT: A PATTERN FOR WRITING GOOD TESTS*. Retrieved from automationpanda.com, sist besøkt 21.04.22: <https://automationpanda.com/2020/07/07/arrange-act-assert-a-pattern-for-writing-good-tests/>
- Baranchuk, Y. (2022, mai 10). *The pros and cons of single page applications (SPAs)*. Retrieved from iTechArt.com, sist besøkt 20.05.22: <https://www.itechart.com/blog/pros-cons-of-single-page-applications/>
- Barral, D. (2019, mai 27). *Colocating React component files: the tools you need*. Retrieved from Medium.com, sist besøkt 24.05.22: <https://medium.com/trabe/colocating-react-component-files-the-tools-you-need-c377a61382d3>

- Beal, V. (2022, januar 28). *Repository*. Retrieved from webopedia.com, sist besøkt 24.05.22: <https://www.webopedia.com/definitions/repository/>
- Be a Better Dev. (2021, Mai 17). *How to Setup AWS ECS Fargate with a Load Balancer | Step by Step*. Retrieved from youtube.com, sist besøkt 02.04.22: <https://www.youtube.com/watch?v=07seigrMAI>
- Berkeley Extension. (n.d.). *What Does a Front End Web Developer Do?* Retrieved from Berkeley Extension sist besøkt 24.05.2022: <https://bootcamp.berkeley.edu/resources/coding/learn-web-development/what-does-a-front-end-web-developer-do/>
- Bolstad, E. (2020, oktober 10). *Juv*. Retrieved from Snl.no, sist besøkt 20.05.22: <https://snl.no/juv>
- CBR Staff Writer. (2020, Januar 20). *What is Apache Tomcat? Introducing the Widely Used Java Servlet and JSP Container*. Retrieved from techmonitor.ai, sist besøkt 17.04.22: <https://techmonitor.ai/technology/hardware/what-is-apache-tomcat>
- Chand, S. (2019, Juli 2). *Dependency Injection Using Spring Boot*, sist besøkt 22.05.22. Retrieved from https://medium.com: <https://medium.com/edureka/what-is-dependency-injection-5006b53af782>
- dependabot. (n.d.). *README.md*. Retrieved from github.com, sist besøkt 09.05.22: <https://github.com/dependabot/dependabot-core>
- Docker. (n.d.). *Docker: Empowering App Development for Developers*. Retrieved from docker.com, sist besøkt 02.05.22: <https://www.docker.com/>
- DP Staff Writer. (2021, 10 12). *HTTP and HTTPS protocols*. Retrieved from developpaper.com, sist besøkt 20.05.22: <https://developpaper.com/http-and-https-protocols/>
- draw.io. (n.d.). *draw.io*. Retrieved from draw.io, sist besøkt 23.05.22: [draw.io](https://draw.io)
- Farooq, I. (2021, september 16). *File Based Routing in NextJS*. Retrieved from enlear.academy, sist besøkt 23.05.22: <https://enlear.academy/file-based-routing-in-nextjs-74278063ebda>
- Figma. (n.d.). *Figma, the collaborative interface design tool*. Retrieved from figma.com, sist besøkt 26.04.22: <https://www.figma.com/>
- Fremtind Forsikring. (n.d.). *Bygg med Jøkul*. Retrieved from jokul.fremtind.no, sist besøkt 23.04.22: <https://jokul.fremtind.no/komigang/bygg>
- Fremtind Forsikring. (n.d.). *Universell utforming*. Retrieved from jokul.fremtind.no, sist besøkt 03.05.22: <https://jokul.fremtind.no/universell-utforming/guide>
- GeeksforGeeks. (2020, Februar 5). *GeeksforGeeks*. Retrieved from Coupling in Java, sist besøkt 20.05.22: <https://www.geeksforgeeks.org/coupling-in-java/>
- GitHub. (n.d.). *About branches*. Retrieved from docs.github.com, sist besøkt 24.05.22: <https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-branches>
- GitHub. (n.d.). *About pull requests*. Retrieved from github.com, sist besøkt 23.05.22: <https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests>
- GitHub. (n.d.). *GitHub Actions - Automate your workflow from idea to production*. Retrieved from github.com, sist besøkt 02.05.22: <https://github.com/features/actions>

- GitHub. (n.d.). *GitHub: Where the World Builds Software*. Retrieved from github.com, sist besøkt 01.05.22: <https://github.com/>
- Globalluxsoft. (2018, juli 13). *Unit testing: leave no chance for bugs in your app!* Retrieved from medium.com, sist besøkt 10.05.22: <https://medium.com/globalluxsoft/unit-testing-leave-no-chance-for-bugs-in-your-app-9eb00dd47f0d>
- Google. (n.d.). *Google Sheets: Free Online Spreadsheet Editor*. Retrieved from google.com, Sist besøkt 19.04.22: <https://www.google.com/sheets/about/>
- Grabski, T. (2020, august 10). *REACT JS PROS AND CONS IN 2020*. Retrieved from pagepro.co, sist besøkt 23.05.22: <https://pagepro.co/blog/react-js-pros-and-cons-in-2020/>
- Hamilton, T. (2022, april 16). *Integration Testing: What is, Types, Top Down & Bottom Up Example*. Retrieved from guru99.com, sist besøkt 05.05.22: <https://www.guru99.com/integration-testing.html>
- Hartman, J. (2022, april 16). *What is Java? Definition, Meaning & Features of Java Platforms*. Retrieved from guru99.com, sist besøkt 23.05.22: <https://www.guru99.com/java-platform.html>
- Hartman, J. (2022, april 2022). *What is Java? Definition, Meaning & Features of Java Platforms*. Retrieved from guru99.com, sist besøkt 04.05.22: <https://www.guru99.com/java-platform.html>
- Heintz, O. (2020, Januar 15). *Hva er brukervennlighet*. Retrieved from convert.no, sist besøkt 01.05.22: <https://www.convert.no/blogg/hva-er-brukervennlighet>
- Heslop, B. (2020, Februar 19). *How to Spot a Headless CMS “Imposter”*. Retrieved from contentstack.com, sist besøkt 24.05.22: <https://www.contentstack.com/blog/all-about-headless/api-doesnt-make-headless-cms/>
- JavaTPoint. (2021). *POJO*. Retrieved from www.javatpoint.com, sist besøkt 23.05.22: <https://www.javatpoint.com/pojo-in-java>
- JavaTPoint. (n.d.). *Spring Boot H2 Database*. Retrieved from javatpoint.com, sist besøkt 24.05.2022: <https://www.javatpoint.com/spring-boot-h2-database>
- JavaTPoint. (n.d.). *Types of Software Testing*. Retrieved from javatpoint.com, sist besøkt 07.05.22: <https://www.javatpoint.com/types-of-software-testing>
- JetBrains. (n.d.). *IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains*. Retrieved from jetbrains.com, sist besøkt 22.04.22: <https://www.jetbrains.com/idea/>
- Jofche, N. (2020, Februar 27). *Synchronous vs. Asynchronous data sources*. Retrieved from Medium: <https://levelup.gitconnected.com/synchronous-vs-asynchronous-data-sources-f78e40f998c1>
- JUnit. (n.d.). *The 5th major version of the programmer-friendly testing framework for Java and the JVM*. Retrieved from junit.org, sist besøkt 27.04.22: <https://junit.org/junit5/>
- Kapoor, A. (2021, mai 21). *2. A Mature Framework*. Retrieved from medium.com, sist besøkt 22.05.22: <https://medium.com/spring-boot/top-pros-cons-of-using-spring-boot-for-business-enterprises-64d777db6fa4>
- Kenton, W. (2021, Oktober 17). *Investorpedias*. Retrieved from Lean Startup, sist besøkt 05.04.22: <https://www.investopedia.com/terms/l/lean-startup.asp>

- Khan, N. (2021, November 30). *How to Create REST APIs with Java and Spring Boot*. Retrieved from www.twilio.com, sist besøkt 12.05.22: <https://www.twilio.com/blog/create-rest-apis-java-spring-boot>
- Løvlie, L. (2021, Mai 11). *Lean startup, design thinking, lean. Hva er det egentlig?* Retrieved from pwc.no, sist besøkt 05.04.22: <https://www.pwc.no/no/pwc-aktuelt/lean-startup-design-thinking-hva-er-det.html>
- Lokesh, G. (2022, Februar 28). *Spring – @GetMapping and @PostMapping*. Retrieved from HowToDoInJava, sist besøkt 23.05.22: <https://howtodoinjava.com/spring5/webmvc/controller-getmapping-postmapping/>
- Mahesh, P. (2019, November 18). *Difference between lazy and eager loading in Hibernate*. Retrieved from tutorialspoint.com, sist besøkt 22.05.22: <https://www.tutorialspoint.com/difference-between-lazy-and-eager-loading-in-hibernate>
- MDN Web Docs. (n.d.). *HTTP response status codes*. Retrieved from mozilla.org, sist besøkt 05.05.22: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>
- Meta. (n.d.). *Messenger*. Retrieved from messenger.com, sist besøkt 07.04.22: <https://www.messenger.com/>
- Metabase. (n.d.). *Configuring the Metabase Application Database*. Retrieved from metabase.com, sist besøkt 01.05.22: <https://www.metabase.com/docs/latest/operations-guide/configuring-application-database.html>
- Microsoft. (2022, Februar 12). *What is Azure Active Directory?* Retrieved from microsoft.com, sist besøkt 30.04.22: <https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis>
- Microsoft. (n.d.). *Microsoft Word – programvare for tekstbehandling | Microsoft 365*. Retrieved from microsoft.com, sist besøkt 10.04.22: <https://www.microsoft.com/nb-no/microsoft-365/word>
- Microsoft. (n.d.). *Personlig skylagring – Microsoft OneDrive*. Retrieved from microsoft.com, sist besøkt 10.04.22: <https://www.microsoft.com/nb-no/microsoft-365/onedrive/online-cloud-storage>
- Microsoft. (n.d.). *Videokonferanser, møter og anrop | Microsoft Teams*. Retrieved from microsoft.com, sist besøkt 10.04.22: <https://www.microsoft.com/nb-no/microsoft-teams/group-chat-software>
- mockito. (n.d.). *Why drink it?* Retrieved from site.mockito.org/, sist besøkt 14.04.22: <https://site.mockito.org/>
- Mozilla. (n.d.). *Cross-Origin Resource Sharing (CORS)*. Retrieved from mozilla.org, sist besøkt 18.04.22: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>
- Nettmaker.no. (n.d.). *Hva er responsivt design og hvorfor er det viktig?* Retrieved from nettmaker.no, sist besøkt 16.05.22: <https://nettmaker.no/kunnskap/responsivt-design>
- Nigel, B., James, C., & Susan, H. (2015, Juli 21). ISO 9241-11 Revised: What Have We Learnt About Usability Since 1998? *Human-Computer Interaction: Design and Evaluation*, pp. 143-151.
- npm. (n.d.). *About packages and modules*. Retrieved from docs.npmjs.com, sist besøkt 04.05.22: <https://docs.npmjs.com/about-packages-and-modules>

- Oloruntoba, S. (2020, september 21). *SOLID: The First 5 Principles of Object Oriented Design*. Retrieved from digitalocean.com, sist besøkt 24.05.22:  
[https://www.digitalocean.com/community/conceptual\\_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design](https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design)
- OpenID. (n.d.). *What is OpenID Connect? How does it work?* Retrieved from openid.net, sist besøkt 10.04.22: <https://openid.net/connect/faq/>
- Oracle. (n.d.). *Deploying Web Applications*. Retrieved from Oracle, sist besøkt 24.05.2022:  
[https://docs.oracle.com/cd/E13222\\_01/wls/docs70/webapp/deployment.html](https://docs.oracle.com/cd/E13222_01/wls/docs70/webapp/deployment.html)
- Oracle. (n.d.). *The Java EE 6 Tutorial*. Retrieved from docs.oracle.com, sist besøkt 23.05.22:  
<https://docs.oracle.com/javaee/6/tutorial/doc/bnbqa.html>
- Paraschiv, E. (2022, Mars 29). *Introduction to Spring Data JPA*. Retrieved from baeldung.com, sist besøkt 23.05.22: <https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa>
- Peterson, R. (2022, April 16). *ER Diagram: Entity Relationship Diagram Model | DBMS Example*. Retrieved from uru99.com, sist besøkt 12.05.22: <https://www.guru99.com/er-diagram-tutorial-dbms.html>
- Polo, L. (2020, juni 8). *Stop Cursing CORS*. Retrieved from medium.com, sist besøkt 20.05.22:  
<https://medium.com/tribalscale/stop-cursing-cors-c2cbb4997057>
- Postman. (n.d.). *About Postman*. Retrieved from postman.com, sist besøkt 05.05.22:  
<https://www.postman.com/company/about-postman/>
- Prasanjith, D. (2020, November 2020). *5 Reasons to Use TypeScript with React*. Retrieved from blog.bitsrc.io, sist besøkt 18.04.2022: <https://blog.bitsrc.io/5-strong-reasons-to-use-typescript-with-react-bc987da5d907>
- Project Lombok. (n.d.). *Project Lombok*. Retrieved from projectlombok.org, sist besøkt 18.04.22:  
<https://projectlombok.org>
- React Query. (n.d.). *Overview*. Retrieved from react-query.tanstack.com, sist besøkt 13.04.22:  
<https://react-query.tanstack.com/overview>
- React. (n.d.). *React – A JavaScript library for building user interfaces*. Retrieved from reactjs.org, sist besøkt 08.04.22: <https://reactjs.org/>
- Rieckpil.de, P. (2020, September 28). *Guide to Testing Spring Boot Applications With MockMvc*. Retrieved from rieckpil.de, sist besøkt 21.04.22: <https://rieckpil.de/guide-to-testing-spring-boot-applications-with-mockmvc/>
- Rolstadås, A., & Liseter, I. M. (2018, april 24). *kravspesifikasjon i Store norske leksikon på snl.no*. Retrieved from snl.no, sist besøkt 08.04.22: <https://snl.no/kravspesifikasjon>
- Sander, K. (2021, Oktober 28). *Brukeropplevelse (UX – Design)*. Retrieved from estudie.no, sist besøkt 01.05.22: <https://estudie.no/ux-design/>
- Sass. (n.d.). *Sass Basics*. Retrieved from sass-lang.com, sist besøkt 18.04.22: <https://sass-lang.com/guide>
- Silver, & Adam. (2019, November 4). *JavaScript isn't always available and it's not the user's fault*. Retrieved from adamsilver.io, sist besøkt 23.05.22: <https://adamsilver.io/blog/javascript-isnt-always-available-and-its-not-the-users-fault/>

- Siteimprove. (n.d.). *WCAG compliance checker*. Retrieved from siteimprove.com, sist besøkt 03.05.22: [https://www.siteimprove.com/glossary/wcag-compliance-checker/?utm\\_campaign=no\\_ppc\\_accessibility&utm\\_source=google&utm\\_medium=ppc&utm\\_content=wcag&utm\\_term=wcag%20checker&campaign\\_id=11197313445&ad\\_group\\_id=126703848656&ad\\_id=541097697987&match\\_type=b&targ](https://www.siteimprove.com/glossary/wcag-compliance-checker/?utm_campaign=no_ppc_accessibility&utm_source=google&utm_medium=ppc&utm_content=wcag&utm_term=wcag%20checker&campaign_id=11197313445&ad_group_id=126703848656&ad_id=541097697987&match_type=b&targ)
- Software Testing Help. (2022, april 3). *What Is User Acceptance Testing (UAT): A Complete Guide*. Retrieved from softwaretestinghelp.com, sist besøkt 22.04.22: <https://www.softwaretestinghelp.com/what-is-user-acceptance-testing-uat/>
- Spínola, D. (2017, juli 29). *TypeScript is a superset of JavaScript*. Retrieved from medium.com, sist besøkt 23.05.22: <https://medium.com/@daspinola/typescript-is-a-superset-of-javascript-meaning-its-a-layer-around-js-with-more-methods-and-that-46bbc9368be1>
- Spring. (n.d.). *Spring Boot - Overview*. Retrieved from spring.io, sist besøkt 29.03.22: <https://spring.io/projects/spring-boot>
- Stowe, M. (2014, Desember 4). *API Best Practices: Nouns, CRUD, and More*. Retrieved from blogs.mulesoft.com, sist besøkt 23.05.22: <https://blogs.mulesoft.com/dev-guides/api-design/api-best-practices-nouns-crud-etc/>
- Svensen, T. (2016, Januar 26). *MVP — viktig verktøy, vanskelig forkortelse*. Retrieved from blogg.bekk.no, sist besøkt 08.04.22: <https://blogg.bekk.no/mvp-viktig-verktøy-vansklig-forkortelse-79aecbc2e246>
- Swagger. (n.d.). *What is Swagger*. Retrieved from swagger.io, sist besøkt 29.03.22: <https://swagger.io/docs/specification/2-0/what-is-swagger/>
- Swagger.io. (n.d.). *Swagger.io*. Retrieved from swagger.io, sist besøkt 21.05.22: <https://swagger.io/solutions/api-documentation/>
- Tanstack.com. (n.d.). *Mutations*. Retrieved from react-query.tanstack.com, sist besøkt 23.05.22: <https://react-query.tanstack.com/guides/mutations>
- The Sprint Book. (n.d.). *A brief history of the Design Sprint*. Retrieved from thesprintbook.com, sist besøkt 25.03.22: <https://www.thesprintbook.com/the-design-sprint>
- Tilsynet for universell utforming av IKT. (n.d.). *EUs webdirektiv (WAD) og WCAG 2.1*. Retrieved from utilsynet.no, sist besøkt 24.05.22: <https://www.uutilsynet.no/webdirektivet-wad/eus-webdirektiv-wad-og-wcag-21/827>
- Tilsynet for universell utforming av IKT. (n.d.). *Kva seier forskrifter?* Retrieved from uutilsynet.no, sist besøkt 24.05.22: [https://www.uutilsynet.no/regelverk/kva-seier-forskrifter/153#privat\\_sektor](https://www.uutilsynet.no/regelverk/kva-seier-forskrifter/153#privat_sektor)
- TutorialsPoint. (n.d.). *JPA - ORM Components*. Retrieved from tutorialspoint.com sist besøkt 18.04.22: [https://www.tutorialspoint.com/jpa/jpa\\_orm\\_components.htm](https://www.tutorialspoint.com/jpa/jpa_orm_components.htm)
- TypeScript. (n.d.). *Classes*. Retrieved from typescriptlang.org, sist besøkt 04.05.22: <https://www.typescriptlang.org/docs/handbook/classes.html>
- Tyson, M. (2019, April 2). *What is JPA? Introduction to the Java Persistence API*. Retrieved from www.infoworld.com, sist besøkt 15.05.22: <https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html>

- Tyson, M. (2022, mai 20). *What is JPA? Introduction to the Jakarta Persistence API*. Retrieved from infoworld.com, sist besøkt 20.05.2022: <https://www.infoworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html>
- utilsynet. (n.d.). *Oppdatert regelverk som følge av webdirektivet (WAD) er no klart*. Retrieved from utilsynet.no, sist besøkt 03.05.22: <https://www.utilsynet.no/webdirektivet-wad/oppdatert-regelverk-som-folge-av-webdirektivet-wad-er-no-klart/1124>
- Vaagaasar, A. L., & Skyttermoen, T. (2021). *Prosjektveilederen*. Oslo: Cappelen Damm.
- Vercel. (n.d.). *What is Next.js?* Retrieved from nextjs.org, sist besøkt 23.05.22: <https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs>
- Vihovde, E. H. (2021, desember 16). *objektorientert programmering*. Retrieved from snl.no, sist sett 25. april 22: [https://snl.no/objektorientert\\_programmering](https://snl.no/objektorientert_programmering)
- Visual Studio Code. (n.d.). *Visual Studio Code - Code Editing. Redefined*. Retrieved from code.visualstudio.com, sist besøkt 05.04.22: <https://code.visualstudio.com/>
- Weber, S. (2022, februar 16). *JavaScript package managers compared: npm, Yarn, or pnpm?* Retrieved from blog.logrocket.com, sist besøkt 23.05.22: <https://blog.logrocket.com/javascript-package-managers-compared/>
- yarn. (n.d.). *Introduction*. Retrieved from yarnpkg.com, sist besøkt 04.05.22: <https://yarnpkg.com/getting-started>
- Zoom.us. (n.d.). *Video conferencing, Cloud Phone, Webinars, Chat, Virtual Events | Zoom*. Retrieved from zoom.us, sist besøkt 05.04.22: <https://zoom.us/>

## Vedlegg

I vedlegg ligger alle skjermbilder, tabeller, figurer og annet som har en relevant sammenheng til rapporten, men som tar for stor plass til å ligge i selve rapporten.

### Vedlegg 1 - Designsprint

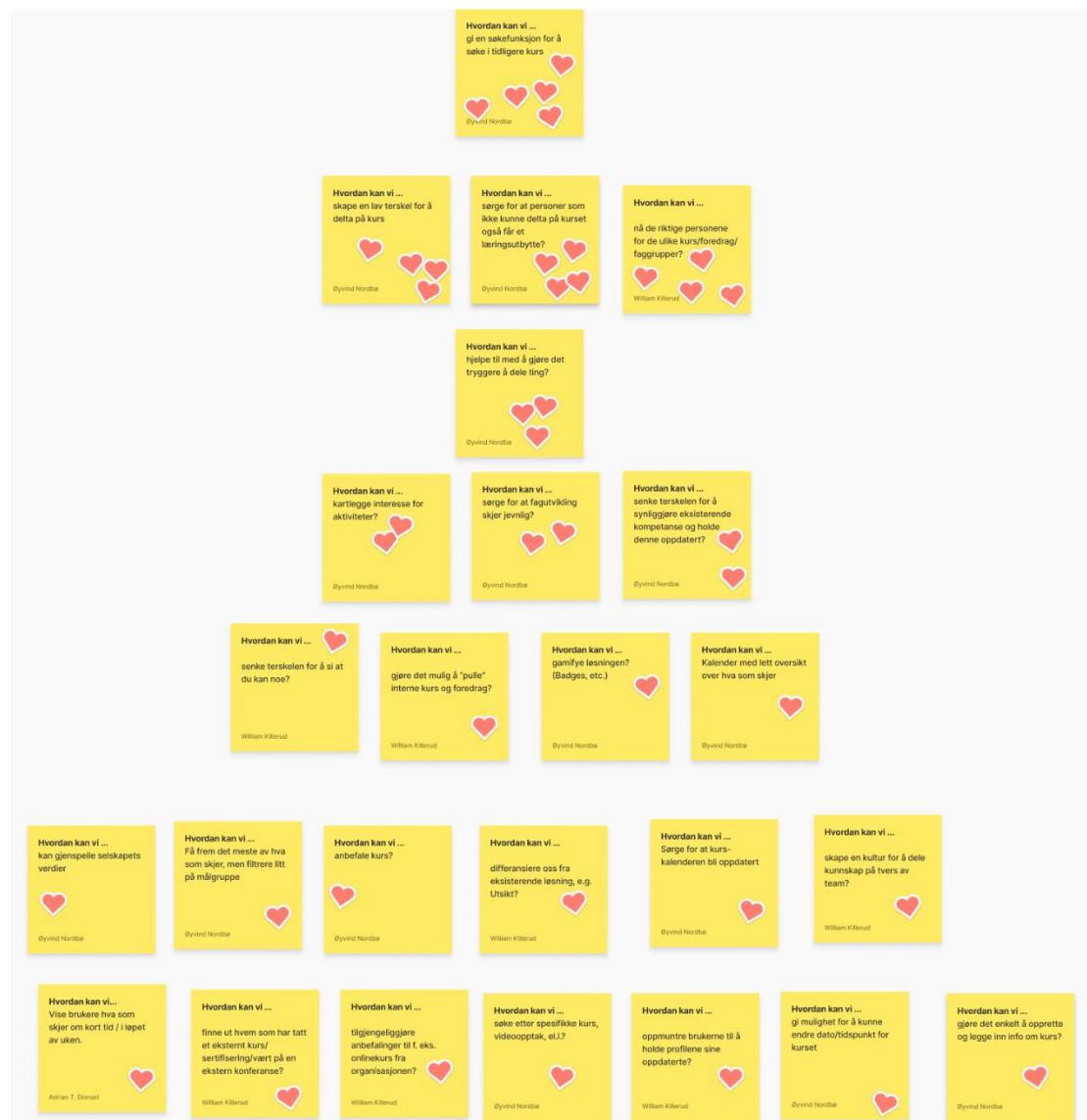
Alle hjerter og «tommel-opp»-symbol er tegn på at dette var noe designsprint-deltakerne likte.

Dette ble brukt i avstemninger om hva som skulle tas med videre under en valgprosess i designsprinten.

#### Dag 1

##### Ekspertintervju:

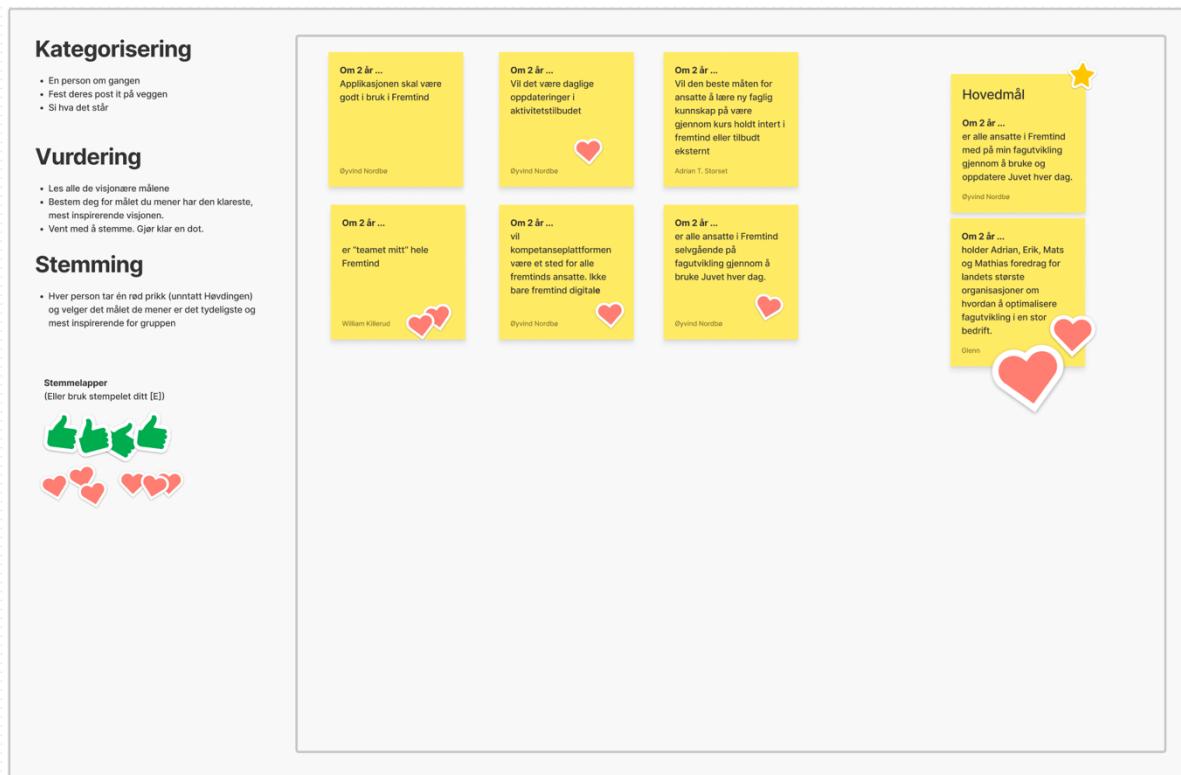
Innspill fra ekspertene omformulert som «hvordan kan vi»-spørsmål.



Figur 64 - Designsprint ekspertintervju - "hvordan kan vi"-spørsmål med varmekart

## Definering av mål:

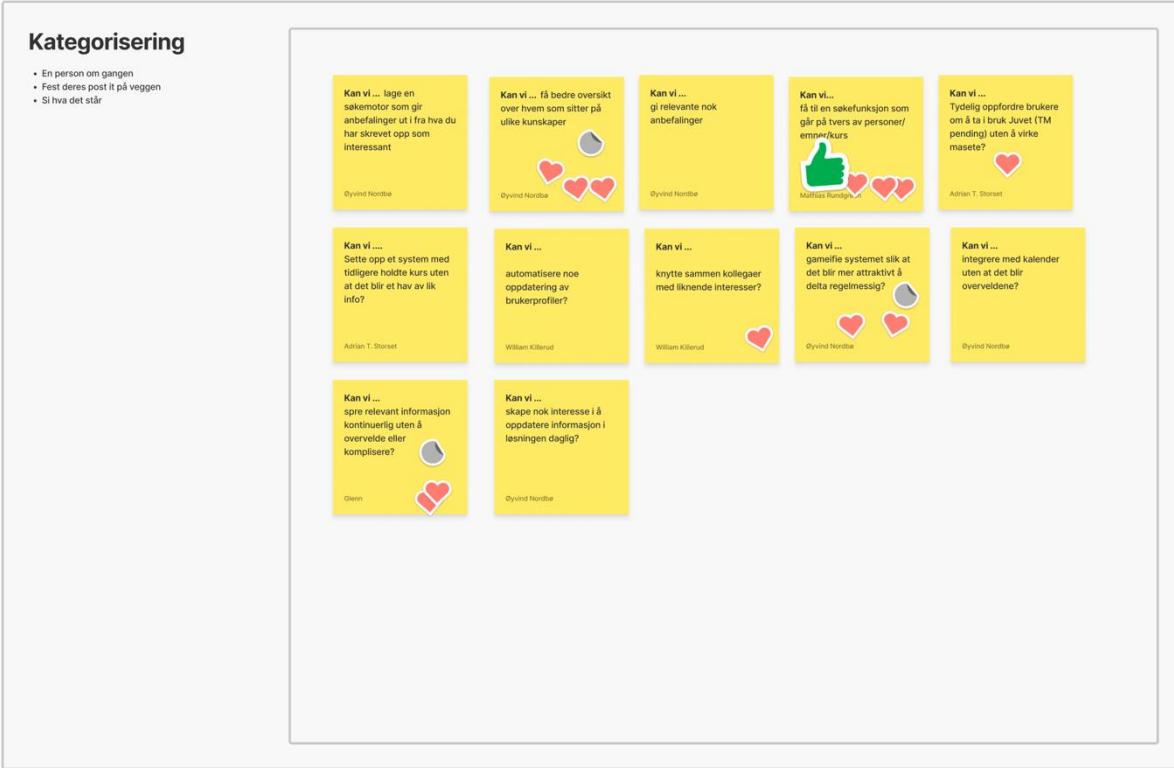
Hensikten er å gi en ledestjerne for en felles retning i resten av sprinten.



Figur 65 - Designsprint - definerting av mål og ledestjerne for designsprint-prosessen

## Sprintspørsmål:

Avgrenser hva man ønsker å løse, teste og prototype i løpet av designsprinten.



Figur 66 - Designsprint: sprintspørsmål oppsummering

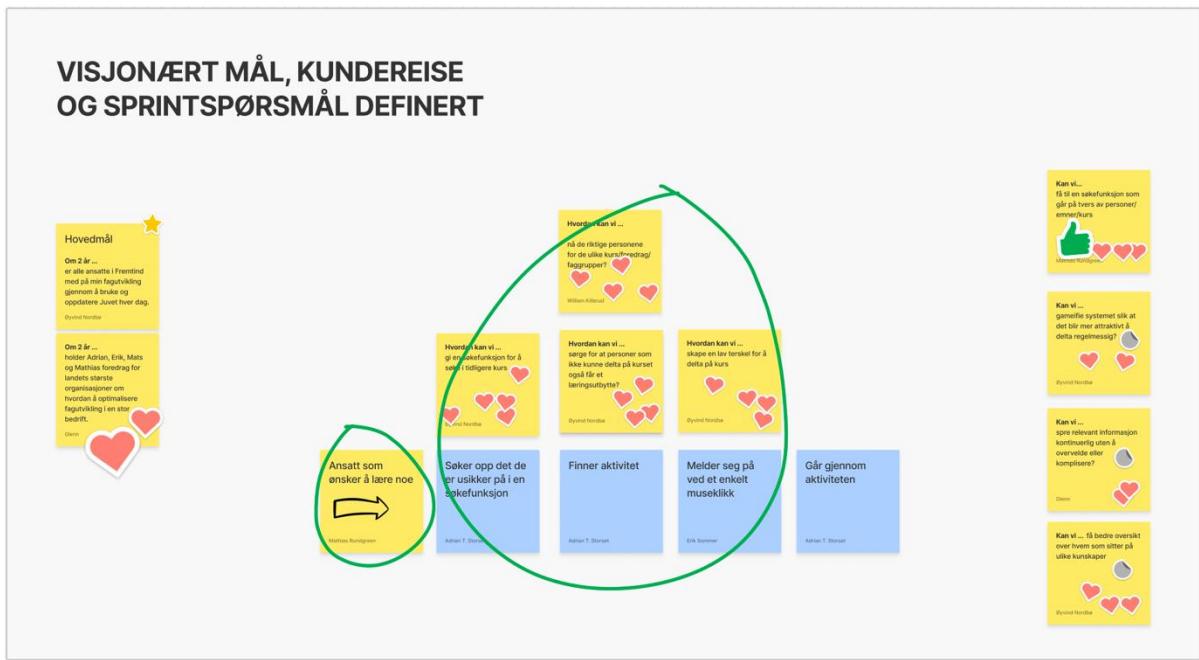
### Tjenestekart:

Hensikten er å få oversikt over stegene i brukerreisen og hvilke «hvordan kan vi»-spørsmål kan løses under denne reisen.



Figur 67 - Designsprint: tjenestekart

Oppsummert visjonært mål, sprintspørsmål og viktigste del av brukerreise.



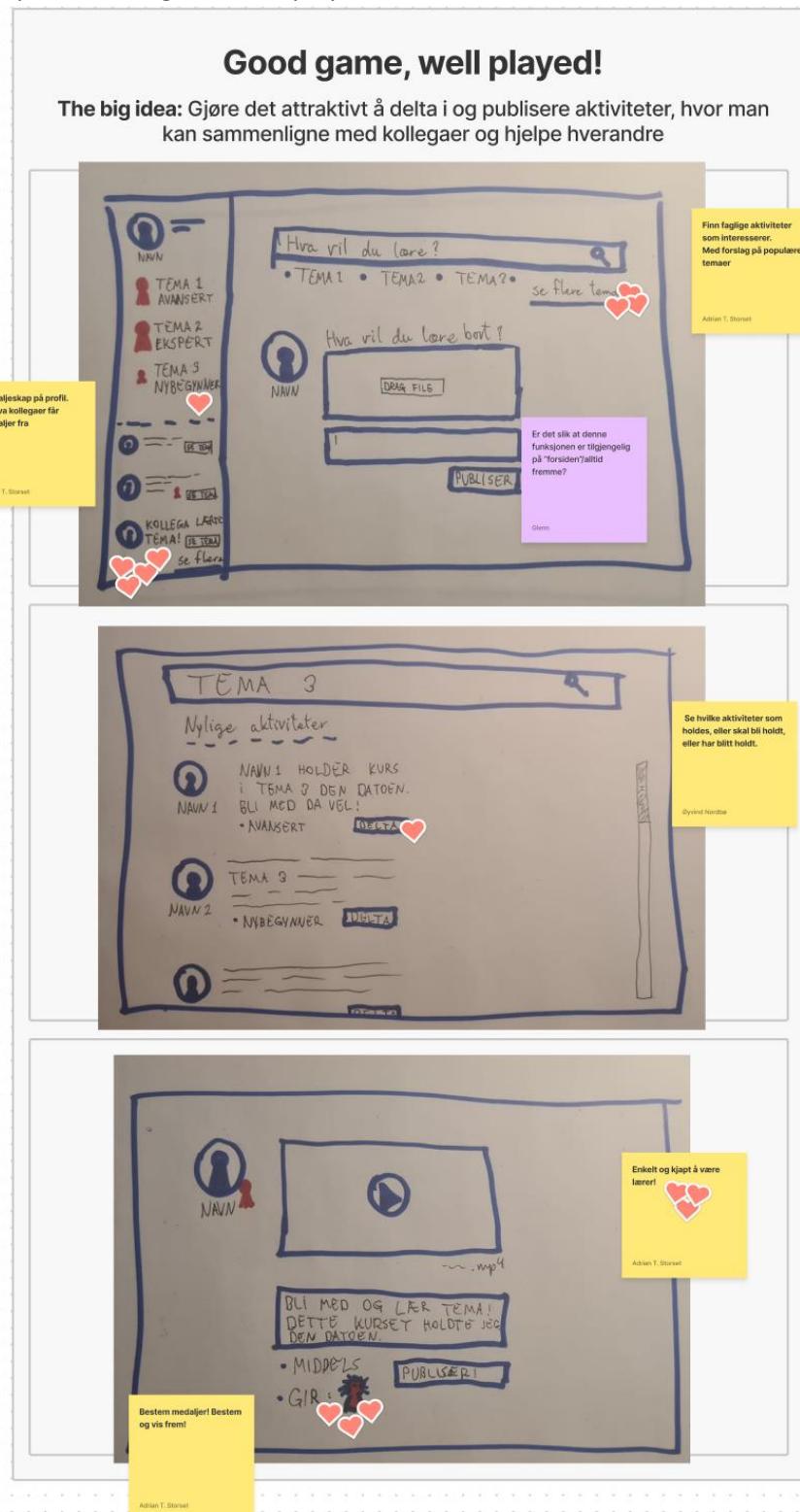
Figur 68 - Designsprint: Oppsummert visjonært mål, sprintspørsmål og viktigste del av brukerreise

## Dag 2

Konseptskisser og «varmekart»:

Hensikten er å få frem ulike konsepter for en løsning, og se hvilke deler av de ulike konseptene som får mest «varme». Dette brukes videre til å dele av konseptene som bør prioriteres.

Konsept 1: «Good game, well played!»



Figur 69 - Konsept 1 av konseptskisser og varmekart, "Good game, well played!".

## Konsept 2: «Finn en venn!»

# Finn en venn!

**The big idea:** Gjennom søk, kan man også finne personer som har kompetanse du leter etter og som har meldt seg til å kunne lære bort denne kompetansen.

Hva vil du lære i dag? ❤️

Hooks Q

Søk er først og fremst. Her starter du hver gang du skal finne noe.

Glen

P2P-sesjoner, Kurs <hr/> Konferanser Personer ...	<b>React Conf</b> <a href="http://www.react-conf.no">www.react-conf.no</a> 24.10.2020 Sydney, Australia  <span style="color: red;">❤️</span>
---	---

Man kan hoppe mellom kategoriene uten å miste resultatene.

Glen

Sekeresultatene er kategorisert i få kategorier som man kan filtrere resultatene med.

Glen

P2P-sesjoner, Kurs <hr/> Konferanser Personer ...	<b>Fronteers</b> <a href="http://www.fronteers.com">www.fronteers.com</a> 12.12.2022 Oslo, Norge  <span style="color: red;">❤️</span>
---	--

Under personer, finner man kun de personene som har sagt de kan hjelpe andre med et gitt tema.

Glen

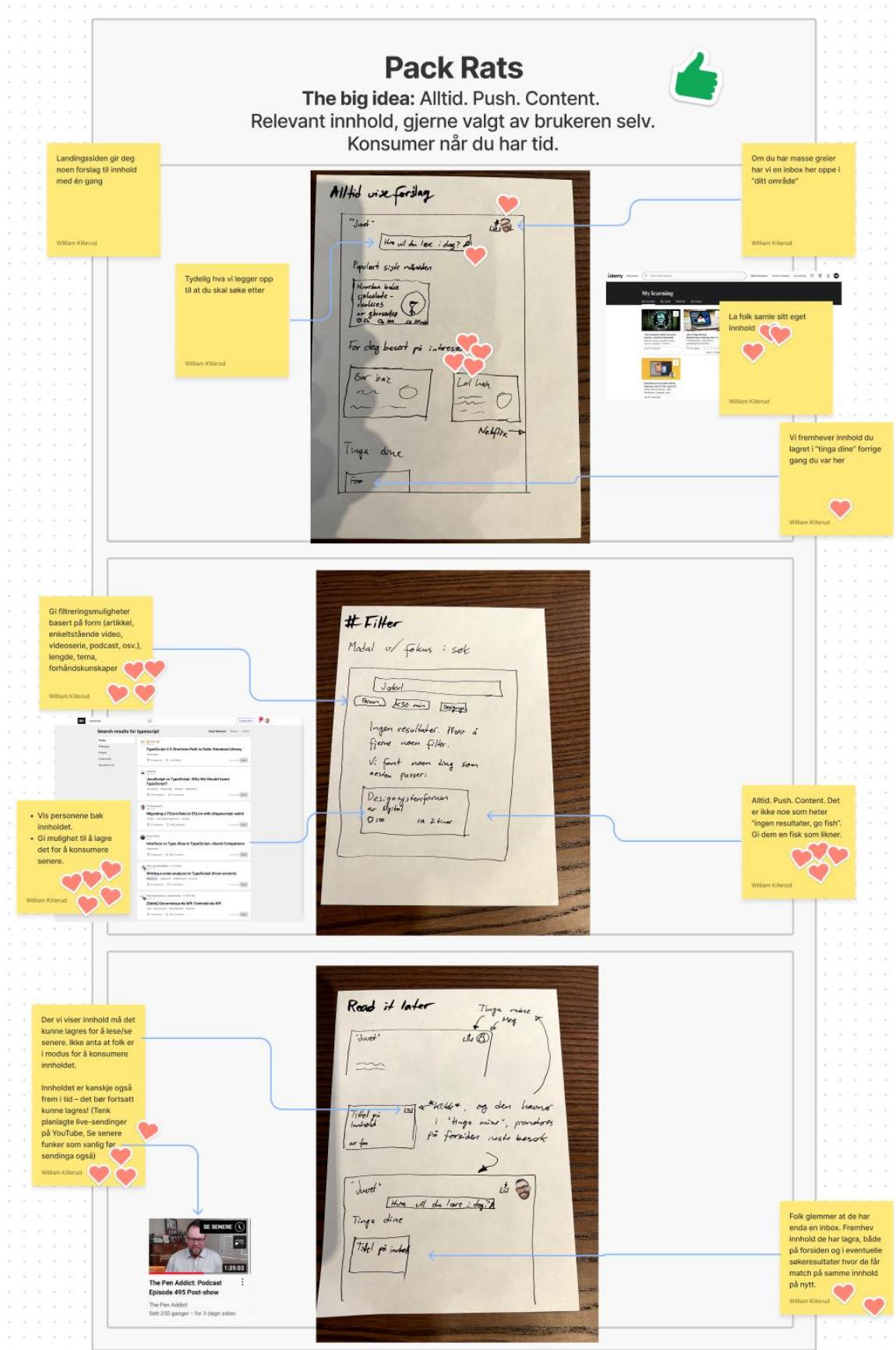
P2P-sesjoner  Kurs  Konferanser  Personer  ...	 <b>William</b> Utvikler, Designsystemteamet • React, Hooks, designsystem   <b>Tom</b> Utvikler, BM-teamet • Html, Css, React, Hooks
--	--

Det er også mulig å finne kontaktinfo og annen informasjon i profilen til en person.

Glen

Figur 70 - Konsept 1 av konseptskisser og varmekart, "Finn en venn!".

### Konsept 3: «Pack Rats»



Figur 71 - Konsept 1 av konseptskisser og varmekart, "Pac rats".

## Konsept 4: «Into the light»

# Into the light

**The big idea:** Søk deg enkelt frem til den kunnskapen du ønsker å få ved et enkelt søk og museklikk.

Kategorier og matchende ikoner  
Erik Sommer

How does the "Join"-functionality work? Does it send an email? Does it integrate with learning management systems? Does it have a calendar invite? Does it have variable mail invites? (configurable)

Join - tenk Meetup.com  
Lever i "Juvet", men enkelt å få påminnelser i "sine" systemer  
Kaleender-invite  
Mail-varslere (konfigurerbare)

Velg enkelt ferdighetsnivå  
Erik Sommer

JUVENT DESIGN MØNSTRE  
17.01.22 Intro til design mønstre i Java Av: Nina Olsen JOIN  
23.02.22 Design mønstre i python Pro level Av: Olav Nilsen JOIN

JUVENT LEVE  
ALLE  
BEGYNNER  
MID  
PRO

OLA N. UTVIDER JS

MEST KURS  
Nina 304  
Per 293  
Petter 282  
dora 253  
Olene 199  
TOP 200  
203. OEG 23  
204. ola 21

PLIST UTMEKKELSER  
dora 80  
Per 78  
Morten 76  
Petter 74  
Nina 73  
TOP 50  
53. OEG 52  
54. Nora 50

Hold følge med telen og ta de igjen  
Erik Sommer

LEADERBOARDS!!  
OLA N. UTVIDER JS  
Fortsæt slit!  
Bore 21 utmøkkelse fra topp 5!!

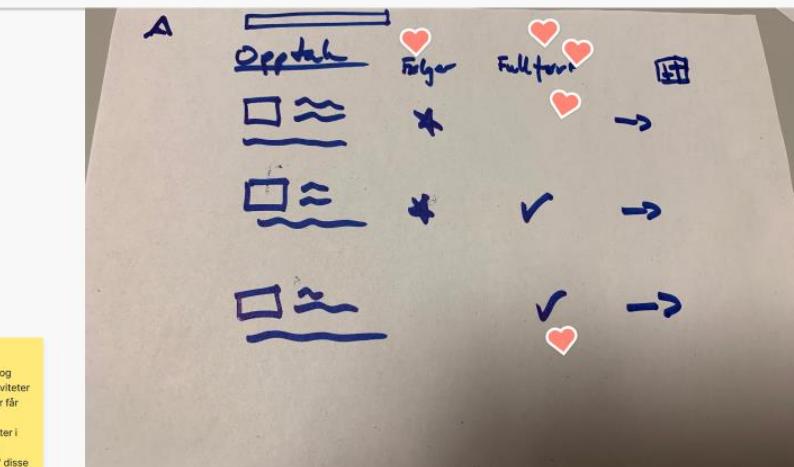
Få en god motivasjon  
Erik Sommer

Figur 72 - Konsept 1 av konseptskisser og varmekart, "Into the light".

## Konsept 4: «Juvet Replay»

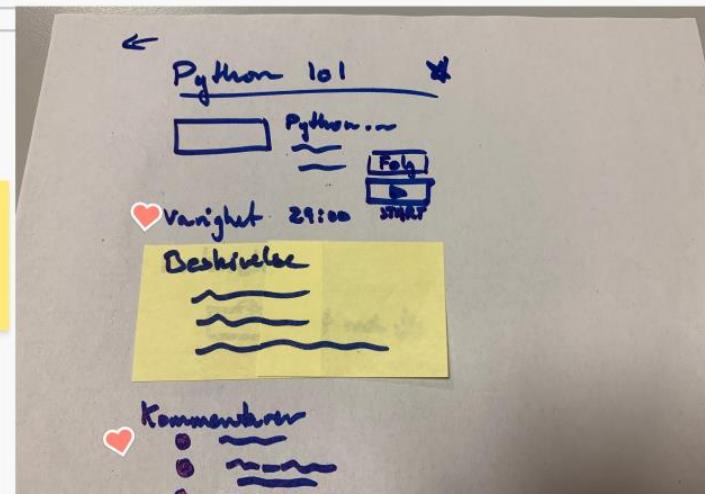
### Juvet Replay

**The big idea:** Gjøre det enkelt for brukere å holde oversikt og få tilgang til aktiviteter som har blitt gjennomført



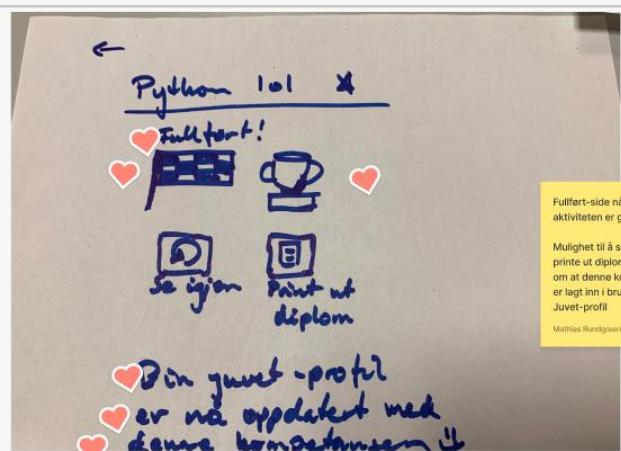
Bruker kan klikke seg videre inn på valgte aktiviteter

Matthias Rundgreen



Bruker trykker på Start og gjennomfører aktiviteten

Gynd Nordba



Fullført-side når aktiviteten er gjennomført.

Mulighet til å se om igjen, printe ut diplom og info om at denne kompetansen er lagt inn i brukerens Juvet-profil

Matthias Rundgreen

Figur 73 - Konsept 1 av konseptskisser og varmekart, "Juvet replay".

## Konsept 5: «NaN»

**NaN**

**The big idea:** En webløsning som fokuserer på å knytte personer som har en viss kunskap med personer som ønsker den kunskapen. Enten det er gjennom kurs eller 1 til 1 møter ❤️

På startsidén finner du en aktivitetskalender for deg selv som skal være minimalistisk og oversiktlig.

Mats Sommervold

Aktivitetene denne uka (Baseret på dine interesser) ❤️

Mandag Tirsdag Onsdag Torsdag Fredag

Finn en resurs! ❤️ Del noe! 🌟

Personer som kan noe om... det du vil lære:

- Olaar React
- Tanmay Python
- Ida Python
- Adrian Git

Se alle ❤️

Baser på hva du har fylt ut i profilen din trekker den frem personer som er villig til å dele sin kunskap om emnet

Mulig å se hvem

Hvordan holder denne oppdatert? Folk larer stadig nye ting og har mer kunnskap å dele. Evt glemmer ting :)

Mathias Rundgreen

Når du søker får du personlig autocomplete suggestions på hva du søker etter.

React

Janice er god på React

Kurs i nærmeste fremtid:

I profilen din kan du sette opp hva du ønsker å lære, og hva du er villig til å lære bort

Mulig å fjerne gammelt kunnskap også

Jeg er interessert i: ❤️ ❤️

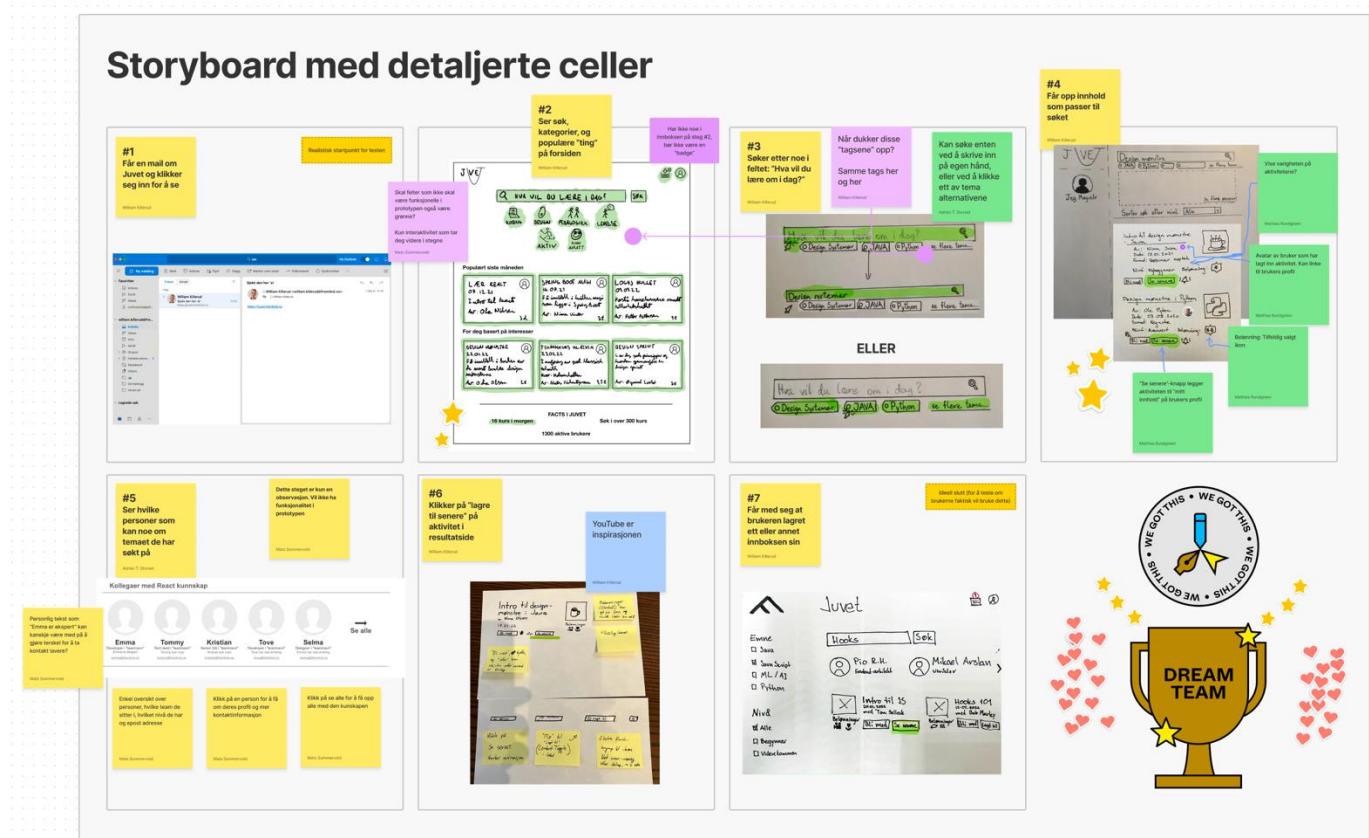
Legg til +

Jeg kan noe om: ❤️ ❤️ Legg til +

Figur 74 - Konsept 1 av konseptskisser og varmekart, "NaN".

## Storyboard oppsummering:

Hensikten er å få på plass detaljene til prototypen slik at det skal gå an å lage prototypen uten å ta flere avgjørelser. Dette viser også flyten i brukerreisen. Dette ble gjort i felleskap og er en sammenstilling av de individuelle story boardene.



Figur 75 - Designsprint: Storyboard

## Dag 3

### Prototyping

Hensikten med lage en prototype er å få testet løsningen tidlig i prosessen slik at det er «mindre kostbart» å gjøre endringer. Prototypen ble laget i Figma og har siden denne ble laget vært igjennom flere forbedringer og utvidelser for å passe behovet.

Landingsside

Velkommen til Juvet,  
kompetanseplattformen av og for Fremtindere

Søk blant over 300 kurs, foredrag, artikler og mer innenfor en rekke temaer

Hva vil du lære om i dag?  🔍 Søk

Koding Design Aktiv Pedagogikk Ledelse

16 aktiviteter begynner allerede i morgen

Populært faginnhold siste måneden

**Dypdykk i JVM-parameterase**  
1,5t Avansert 11.04.22  
Tom Sellebek  
Tom tar oss igjennom avansert konfigurasjon av JVM'en for å få maksimal ytelse aksjon  
Meld på Lagre til senere  
[Se mer →](#)

**Introkurs i Python**  
1t Nybegynner 11.04.22  
Linda Johnsen  
Linda bruker Python aktivt hver dag og vil gjerne hjelpe alle som er usikre på hvor man skal starte.  
Meld på Lagre til senere  
[Se mer →](#)

**Alle Log4j-hullene 🤔**  
30m Nybegynner 04.01.22  
Petter Thoresen  
Petter har lest alle artiklene om Log4j-sikkerhetshullene, slik at du ikke trenger  
Lagre til senere  
[Se mer →](#)

Faginnhold for deg basert på dine interesser

**Designmønstre i Java**  
2t Viderekommen 11.04.22  
Tommy Jensen  
Tommy gir oss innsikt i de mest brukte designmønstrene i Java-verden  
Meld på Lagre til senere  
[Se mer →](#)

**Teknikkurs Klassisk**  
1d Avansert 21.02.22  
Kristian Knudsen  
Kristian har upåklagelig stil i klassisk. I dette kurset forteller han deg hemmelighetene sine.  
Meld på Lagre til senere  
[Se mer →](#)

**Hvordan facilitere designsprint**  
5d Avansert 11.04.22  
Tirill Junge  
Tirill tar oss igjennom en designsprint fra start til slutt med stødig hånd  
Meld på Lagre til senere  
[Se mer →](#)

Figur 76 - Prototype av landingsside under designsprint.

Søkefelt oppdateres

The screenshot shows a landing page for 'Juvet', a competence platform. At the top, there's a search bar with 'Spring Boot' typed in, a clear button, and a 'Søk' (Search) button. Below the search bar are five category icons: 'Koding' (Coding), 'Design', 'Aktiv' (Active), 'Pedagogikk' (Pedagogy), and 'Ledelse' (Leadership). A banner below these categories says '16 aktiviteter begynner allerede i morgen' (16 activities start tomorrow). The main content area has two sections: 'Populært faginnhold siste måneden' (Popular academic content last month) and 'Faginnhold for deg basert på dine interesser' (Academic content for you based on your interests). Each section contains three cards, each with a title, a brief description, a timestamp, the author's name, and two buttons: 'Meld på' (Sign up) and 'Lagre til senere' (Save for later). In the 'Populært faginnhold' section, the first card is 'Dypdykk i JVM-parameterase' by Tom Sellebek, the second is 'Introkurs i Python' by Linda Johnsen, and the third is 'Alle Log4j-hullene' by Petter Thoresen. In the 'Faginnhold for deg' section, the first card is 'Designmønstre i Java' by Tommy Jensen, the second is 'Teknikkurs Klassisk' by Kristian Knudsen, and the third is 'Hvordan fasilitere designsprint' by Tirill Junge.

Velkommen til Juvet,  
kompetanseplattformen av og for Fremtindere

Søk blant over 300 kurs, foredrag, artikler og mer innenfor en rekke temaer

Spring Boot x

**Søk**

Koding Design Aktiv Pedagogikk Ledelse

16 aktiviteter begynner allerede i morgen

Populært faginnhold siste måneden

**Dypdykk i JVM-parameterase**  
1,5t Avansert 11.04.22  
 Tom Sellebek  
Tom tar oss igjennom avansert konfigurasjon av JVM'en for å få maksimal ytelse akselj  
**Meld på** **Lagre til senere**  
Se mer →

**Introkurs i Python**  
1t Nybegynner 11.04.22  
 Linda Johnsen  
Linda bruker Python aktivt hver dag og vil gjerne hjelpe alle som er usikre på hvor man skal starte.  
**Meld på** **Lagre til senere**  
Se mer →

**Alle Log4j-hullene**   
30m Nybegynner 04.01.22  
 Petter Thoresen  
Petter har lest alle artiklene om Log4j-sikkerhetshullene, slik at du ikke trenger  
**Lagre til senere**  
Se mer →

Faginnhold for deg basert på dine interesser

**Designmønstre i Java**  
2t Viderekommen 11.04.22  
 Tommy Jensen  
Tommy gir oss innsikt i de mest brukte designmønsterne i Java-verden  
**Meld på** **Lagre til senere**  
Se mer →

**Teknikkurs Klassisk**  
1d Avansert 21.02.22  
 Kristian Knudsen  
Kristian har upåklagelig stil i klassisk. I dette kurset forteller han deg hemmelighetene sine.  
**Meld på** **Lagre til senere**  
Se mer →

**Hvordan fasilitere designsprint**  
5d Avansert 11.04.22  
 Tirill Junge  
Tirill tar oss igjennom en designsprint fra start til slutt med stødig hånd  
**Meld på** **Lagre til senere**  
Se mer →

Figur 77 - Prototype av landingsside med oppdatert søkerfelt under designsprint

**Søkeresultater**

Juvet

## Vi fant 12 treff på **Spring Boot**

Eksperter på **Spring Boot** i Fremtind

Tiril Junge  
Utvikler

Glenn B.  
Leder for Utvikling

Tiril er ekspert  
[Book en økt med Tiril](#)

Glenn er ekspert  
[Book en økt med Glenn](#)

Se alle som kjenner til temaet →

**Relevante kurs i Spring Boot**

**Neste steg med Spring Boot**

1t Nybegynner 11.04.22

 Nina Vårvævel

Nina viser oss hvorom vi kan ta steget videre med noen ofte brukte features av Spring Boot

[Meld på](#) [Lagre til senere](#)

Se mer →

**Dypdykk i JVM-parametere**

1,5t Avansert 11.04.22

 Tom Sellebek

Tom tar oss igjennom avansert konfigurasjon av JVM-en for å få maksimal ytelse akkurat

[Meld på](#) [Lagre til senere](#)

Se mer →

**Kotlin er best, ingen protest**

0,5t Nybegynner 27.02.22

 Tom Sellebek

Tom forteller oss hvorfor alle som liker Java tar feil

[Meld på](#) [Lagre til senere](#)

Se mer →

**Emne**

Kurs

Webinar

Seminar

Sosialt

**Nivå**

Alle

Nybegynner

Viderekommende

Avansert

Lagt til i innboks

Juvet Spring Boot

## Vi fant 12 treff på **Spring Boot**

Eksperter på **Spring Boot** i Fremtid



Tiril Jungle  
Utvikler



Glenn B.  
Leder for Utvikling

Tiril er ekspert  
[Book en økt med Tiril](#)

Glenn er ekspert  
[Book en økt med Glenn](#)

**Emne**

- Kurs
- Webinar
- Seminar
- Sosialt

**Nivå**

- Alle
- Nybegynner
- Viderekommen
- Avansert

Se alle som kjenner til temaet →

Relevante kurs i **Spring Boot**

**Neste steg med Spring Boot**

1t Nybegynnervennlig 11.04.22

 Nina Vårstøvel

Nina viser oss hvorvidt vi kan ta steget videre med noen ofte brukte features av Spring Boot

Du samler: 

Se mer →

**Dypdykki i JVM-parametere**

1,5t Avansert 11.04.22

 Tom Sellebek

Tom tar oss igjennom avansert konfigurasjon av JVM-en for å få maksimal ytelse akkurat

Du samler: 

Se mer →

**Kotlin er best, ingen protest**

0,5t Nybegynnervennlig 27.02.22

 Tom Sellebek

Tom forteller oss hvorvidt alle som liker Java tar feil

Du samler: 

Se mer →

**Log4j? Prøv logging til stdout!**

1t Litt erfaring 21.03.22

 Tom Sellebek

Kan ikke få remote code execution om du bare logger til konsollen 😊

Du samler: 

Se mer →

**ObjectFactoryBuilder**

2t Avansert 09.08.22

 Tom Sellebek

Builders er den eneste riktige måten å lage instanser av objekter på. Tom forteller deg hvordan

Du samler: 

● "Neste steg med Spring Boot" ble lagt til i ditt innhold

Figur 79 - Prototype av landingsside, aktivitet lagt til i "lagret til senere" under designsprint

Figur 80 - Prototype av "Innboks" under designsprint

Kilder i prototypen: [Profilbilde av «Glenn B.» - fra Medium.com](#), [alle andre profilbilder – fra ThisPersonDoesNotExist.com](#), [Spring Boot logo – fra Spring Boots offisielle Twitter-konto](#), [Java-logo – fra Java Icon Images # 201804](#), [Kotlin-logo – fra JetBrains offisielle nettside](#), [Sort kode-symbol med hvite streker - Code icons skapt av Freepik – Flaticon](#), [medalje-symbol – Ikon av Iconpacks](#). Utøver dette er all design, skrift og symboler hentet fra Fremtinds designsystem, [Jøkul](#).

## Dag 4

Oppsummering av designsprinten, skrevet av fasilitator i Fremtind:

## Hva har dere gjort?

Recap av uken som var.

- 🎯 Dere gikk fra bred problemstilling til fokusert utfordring.
- 🚀 Dere utforsket og definerte visjonære mål.
- 🤔 Dere valget ut en rekke spørsmål dere ville ha svar på.
- 💡 Dere kom opp med **seks konsepter**, og hentet de beste ideéne fra hver av de.
- 🛠️ Dere har designet en testbar, interaktiv prototype i løpet av åtte timer.
- 💻 Dere validerte konseptet deres på ekte brukere, i løpet av fire dager.
- 📝 Dere samlet hele **898 innsikter!**
- 📊 Dere fikk svar på spørsmålene dere lurt på.

Figur 81 - Designsprint oppsummering - hva har vi gjort?

## Innsiktssyntese

Et par kjappe stats.

- 🎯 Dere har fått bekreftet behovet for tjenesten.
- ✓ **Åtte av ti** sprintspørsmål ble bekreftet
- ⚠️ Kun ett sprintspørsmål må behandles med varsomhet
- 🚫 Kun ett sprintspørsmål ble underkjent
- 📝 Dere samlet **233 viktige innsikter (filtrert ned fra 898!)**
- 📊 121 positive, 51 negative, og 31 sitater\*, 30 generelle

\* Mathias sine notater var alle i bold, så det var vanskelig å filtrere ut hva som var sitater, mulig jeg har missa noen. Sjekk over en gang sammen.

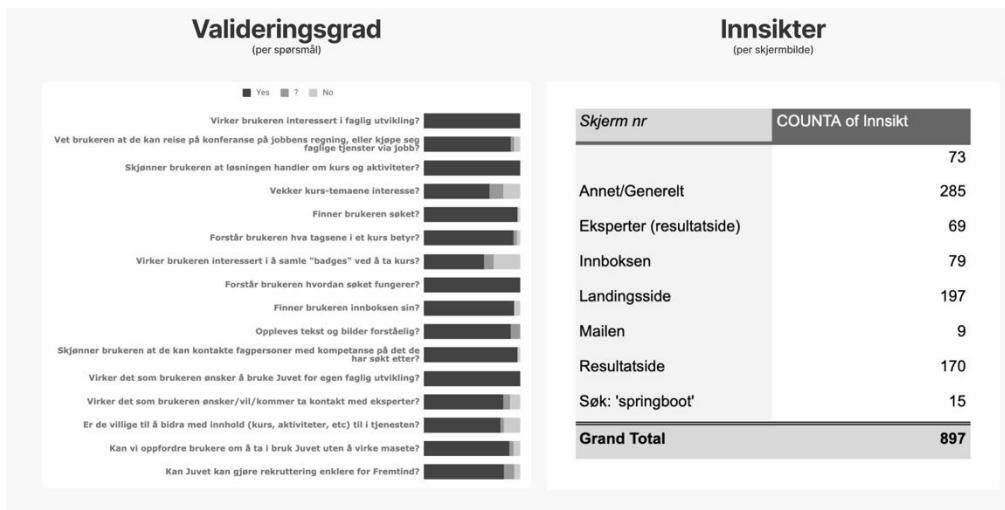
Figur 82 - Designsprint oppsummering – Innsiktssyntese

## Sprintspørsmålene

✓ 80% bekreftet ✅ 10% trå varsomt 🚫 10% underkjent

- ✓ Virker det som brukeren ønsker å bruke Juvet for egen faglig utvikling?
- ✓ Virker det som brukeren ønsker/vil/kommer ta kontakt med eksperter?
- ✓ Finnes det villighet til å bidra med innhold i tjenesten?
- ✓ Kan vi oppfordre brukere om å ta i bruk Juvet uten å virke masete?
- ✓ Kan Juvet kan gjøre rekruttering enklere for Fremtind?
- ✓ Kan vi få bedre oversikt over hvem som sitter på ulike kunskaper?
- ✓ Kan vi spre relevant informasjon kontinuerlig uten å overvelde eller komplisere?
- ✓ Klarer vi å hjelpe Fremtindere til å finne fagpersoner, kurs eller emner via Juvet?
- ⚠️ Kan vi "gamifye" systemet slik at det blir mer attraktivt å delta regelmessig?
- 🚫 Virker brukeren interessert i å samle "badges" ved å ta kurs?

Figur 83 - Designsprint oppsummert - Sprintspørsmålene



Figur 84 - Designsprint oppsummert - Valideringsgrad og innsikter

## Sitater

Noen uttrekk, sjekk scorecardet for totalen.

- Hvordan kan vi skaffe innholdet?
- Bryr meg ikke om medaljer.
- Lærer meg ikke noe for å få medalje, det sier ikke noe om hva jeg kan.
- Er Tiril og Glenn bedre eller dårligere enn *meg*?
- Kan man komme med ukens foredrag og kurs?
- Hva skal til for å være en ekspert?
- Hva er mine interesser? Hvordan vet systemet det?

Figur 85 - Designsprint oppsummert - Sitater fra brukertester



Figur 86 - Designsprinten oppsummert - Fokuspunkter for videre utvikling



Figur 87 - Designsprint oppsummert - Tips til videre steg i utviklingen

## Vedlegg 2 – Tabell med spørsmål for brukertest under designsprint

Tabell 6 - Spørsmål ved brukertest designsprinten.

1	Virker brukeren interessert i faglig utvikling?
2	Vet brukeren at de kan reise på konferanse på jobbens regning, eller kjøpe seg faglige tjenester via jobb?
3	Skjønner brukeren at løsningen handler om kurs og aktiviteter?
4	Vekker kurs-temaene interesse?
5	Finner brukeren søkeret?
6	Forstår brukeren hva tagsene i et kurs betyr?
7	Virker brukeren interessert i å samle "badges" ved å ta kurs?
8	Forstår brukeren hvordan søkeret fungerer?
9	Finner brukeren innboksen sin?
10	Oppleves tekst og bilder forståelig?
11	Skjønner brukeren at de kan kontakte fagpersoner med kompetanse på det de har søkeret etter?
12	Virker det som brukeren ønsker å bruke Juvet for egen faglig utvikling?
13	Virker det som brukeren ønsker/vil/kommer ta kontakt med eksperter?
14	Er de villige til å bidra med innhold (kurs, aktiviteter, etc) til i tjenesten?
15	Kan vi oppfordre brukere om å ta i bruk Juvet uten å virke maserte?
16	Kan Juvet kan gjøre rekruttering enklere for Fremtind?
17	Kan vi få bedre oversikt over hvem som sitter på ulike kunnskaper?
18	Kan vi spre relevant informasjon kontinuerlig uten å overvelde eller komplisere?
19	Klarer vi å hjelpe Fremt indirekt til å finne fagpersoner, kurs eller emner via Juvet?
20	Kan vi "gamifye" systemet slik at det blir mer attraktivt å delta regelmessig?

## Vedlegg 3 – Svarark brukertest av prototype under designsprint

### Spørsmål

Skriv inn ditt navn til høyre:					Erik
10:00	11:00	12:00	14:00	15:00	
Testperson 1	Testperson 2	Testperson 3	Testperson 4	Testperson 5	<b>Spørsmål vi vil ha svar på</b>
Yes	Yes	Yes	Yes	Yes	Virker brukeren interessert i faglig utvikling?
Yes	Yes	Yes	Yes	Yes	Vet brukeren at de kan reise på konferanse på jobbens regning, eller kjøpe seg faglige tjenster via jobb?
Yes	Yes	Yes	Yes	Yes	Skjønner brukeren at løsningen handler om kurs og aktiviteter?
No	No	Yes	Yes	Yes	Vekker kurs-temaene interesse?
Yes	Yes	Yes	Yes	Yes	Finner brukeren søker?
Yes	Yes	Yes	Yes	Yes	Forstår brukeren hva taggene i et kurs betyr?
No	Yes	Yes	Yes	Yes	Virker brukeren interessert i å samle "badges" ved å ta kurs?
Yes	Yes	Yes	Yes	Yes	Forstår brukeren hvordan søker fungerer?
Yes	Yes	Yes	Yes	Yes	Finner brukeren innboksen sin?
Yes	Yes	Yes	Yes	Yes	Opplevdes tekst og bilder forståelig?
Yes	Yes	Yes	Yes	Yes	Skjønner brukeren at de kan kontakta fagpersoner med kompetanse på det de har søkt etter?
Yes	Yes	Yes	Yes	Yes	Virker det som brukeren ønsker å bruke Juvet for egen faglig utvikling?
Yes	Yes	Yes	Yes	Yes	Virker det som brukeren ønsker/vil/kommer ta kontakt med eksperter?
No	Yes	Yes	Yes	Yes	Er de villige til å bidra med innhold (kurs, aktiviteter, etc) til i tjenesten?
Yes	Yes	Yes	Yes	Yes	Kan vi oppfordre brukere om å ta i bruk Juvet uten å virke masete?
No	Yes	Yes	?	Yes	Kan Juvet kan gjøre rekrytering enklere for Fremtind?
Yes	Yes	Yes	Yes	Yes	Kan vi få bedre oversikt over hvem som sitter på ulike kunskaper?
Yes	Yes	Yes	Yes	Yes	Kan vi spre relevant informasjon kontinuerlig uten å overveide eller komplisere?
Yes	Yes	Yes	Yes	Yes	Klarer vi å hjelpe Fremtindere til å finne fagpersoner, kurs eller emner via Juvet?
No	?	Yes	Yes	Yes	Kan vi "gamify" systemet slik at det blir mer attraktivt å delta regelmessig?

### Innsikter

Hjem har lagt inn	Bruk	Skjerm	Innsikt	Hvor betydningsful var innsikten på en skala fra 1 - 3 (3 er viktigst)	
Erik	Testperson 1	Landingsside	Forskjellige farge på de ulike kortene		2
Erik	Testperson 1	Landingsside	Litt mye info		3
Erik	Testperson 1	Landingsside	Forstår ikke helt hva hun skal gjøre		3
Erik	Testperson 1	Landingsside	Vil ha mer informasjon når du trykker på kortene		3
Erik	Testperson 1	Søk: 'springboot'	Vil ha forslag		3
Erik	Testperson 1	Søk: 'springboot'	Gikk litt fort		1
Erik	Testperson 1	Eksperter (resultatside)	Skjønner at vi har eksperter		3
Erik	Testperson 1	Resultatside	Avhuksbokser viser nye filtrerte kurs		3
Erik	Testperson 1	Resultatside	Liker designet på siden		3
Erik	Testperson 1	Resultatside	Skjønner at du samler medaljer, inspirerer ikke		2
Erik	Testperson 1	Resultatside	Medaljene kan gi en falsk trygghet		3
Erik	Testperson 1	Resultatside	Vil ha mer informasjon når du trykker på kortene		3
Erik	Testperson 1	Resultatside	Vet ikke helt hva lager til senere betyr		3
Erik	Testperson 1	Resultatside	Ønsker et stjernemerke		1
Erik	Testperson 1	Eksperter (resultatside)	Ønsker å få et møte i outlook, og info om når		3
Erik	Testperson 1	Landingsside	Litt for lang overskrift		2
Erik	Testperson 1	Resultatside	Bryr seg mest om kursene, og lite om de ansatte		2
Erik	Testperson 1	Resultatside	Fler filtreringsmuligheter		3
Erik	Testperson 1	Mailen	Mailen er forståelig		2
Erik	Testperson 1	Annet/Generelt	Bir ikke skufet om det ikke blir noe av prosjektet		2
Erik	Testperson 1	Annet/Generelt	Viktig å ha en ledesjerne for hva som er målet for året		2
Erik	Testperson 2	Annet/Generelt	Det er et behov for mer faglig aktivitet		3
Erik	Testperson 2	Annet/Generelt	Bygge et godt miljø gjennom forum		3
Erik	Testperson 2	Annet/Generelt	Utfordrende å finne tid og mulighet til personlig kompetansehevnning		3
Erik	Testperson 2	Annet/Generelt	Knytte opp mot prosjekter som er i Fremtind		2
Erik	Testperson 2	Annet/Generelt	Bruker mye eksternt faginnhold		1
Erik	Testperson 2	Eksperter (resultatside)	Ute etter å finne de ansatte som sitter på ulike kunnskaper		3
Erik	Testperson 2	Annet/Generelt	Kun de han har jobbet med som han vet hva kan		3
Erik	Testperson 2	Annet/Generelt	Har ikke peiling på analyse og data for eks.		3
Erik	Testperson 2	Mailen	Skjønner mailen		1
Erik	Testperson 2	Landingsside	Sammensheng med jøkul		3
Erik	Testperson 2	Landingsside	Spennende		3
Erik	Testperson 2	Landingsside	Veldig bra med kategorier		3
Erik	Testperson 2	Landingsside	God oversikt		3
Erik	Testperson 2	Landingsside	Spennede kategorier		3
Erik	Testperson 2	Landingsside	Fint at det står hvor langt, nivå osv		2
Erik	Testperson 2	Landingsside	Bra førsteintrykk		3
Erik	Testperson 2	Søk: 'springboot'	Skjønner sekefell		3
Erik	Testperson 2	Søk: 'springboot'	Veldig bra		2
Erik	Testperson 2	Resultatside	Emene må ha en relevans til søker		3
Erik	Testperson 2	Resultatside	Tør emene er knyttet opp mot søker		2
Erik	Testperson 2	Eksperter (resultatside)	Raske spørsmål til ekspertene		3
Erik	Testperson 2	Eksperter (resultatside)	Liker ekspertene øverst		3
Erik	Testperson 2	Resultatside	Oversiktelig		3
Erik	Testperson 2	Resultatside	Sliter litt med å komme seg videre		2
Erik	Testperson 2	Innboksen	Veldig bra at du kan se hva du har gjort		2
Erik	Testperson 2	Innboksen	Veldig bra med lagring til seinere		3
Erik	Testperson 2	Innboksen	Når han melder seg på så ønsker han seg en bekrefteelse og mer informasjon. Detaljer som hva som er målet med kurset, hva som gjennomgås, hva kan brukes i det daglige, hvem er kursholderen		3
Erik	Testperson 2	Innboksen	La merke til medaljer litt sein		1
Erik	Testperson 2	Annet/Generelt	Se hva andre har tatt av kurs, sier litt om hva de kan, hvordan de har brukt det hvor de jobber		3
Erik	Testperson 2	Annet/Generelt	Lett å navigere seg igjennom		3

Figur 88 - Designsprint brukertest svarark

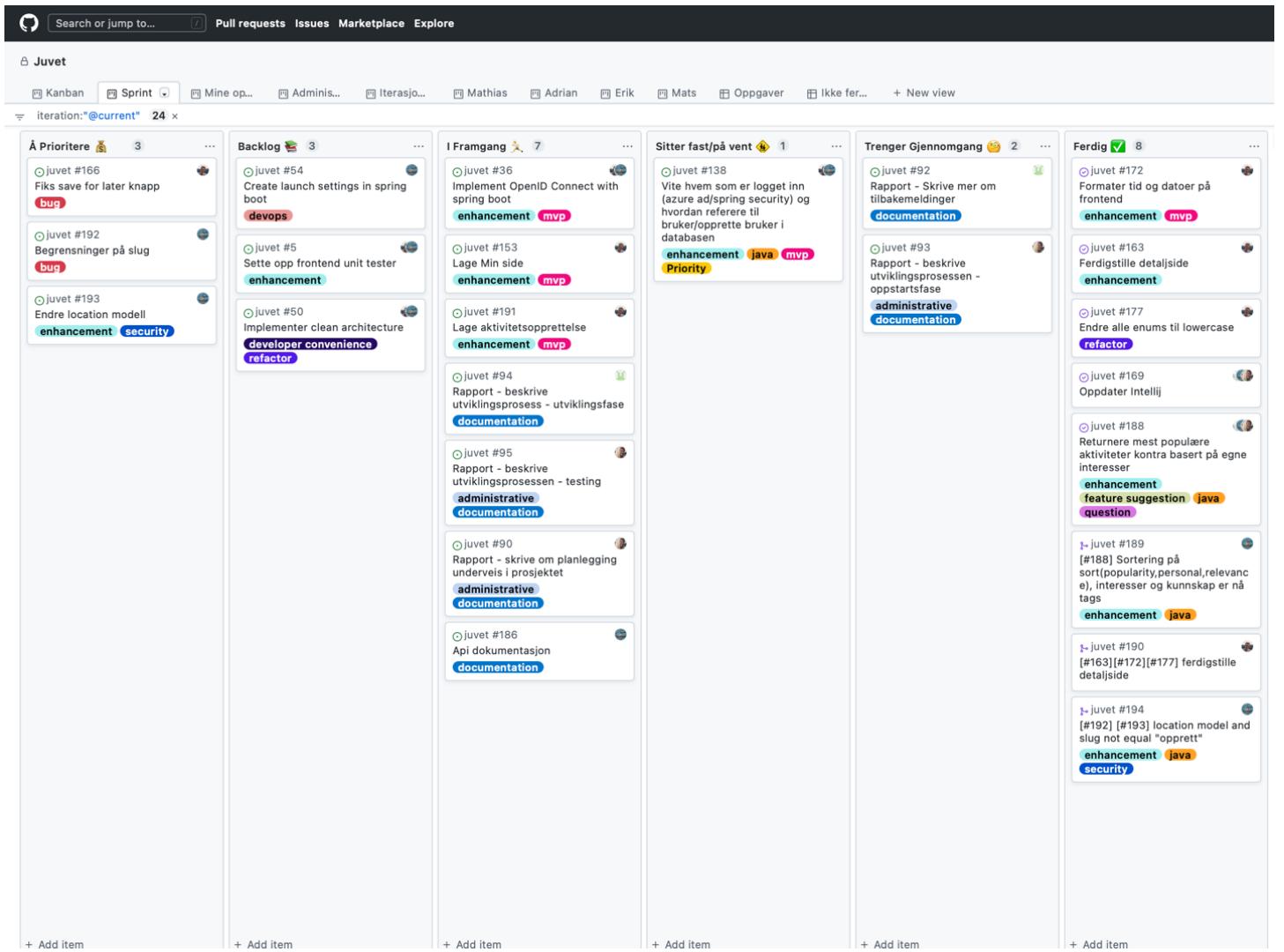
## Vedlegg 4 – GitHub projects prosjektbrett

The image shows a GitHub Kanban board for the project "Juvel". The board is organized into six columns:

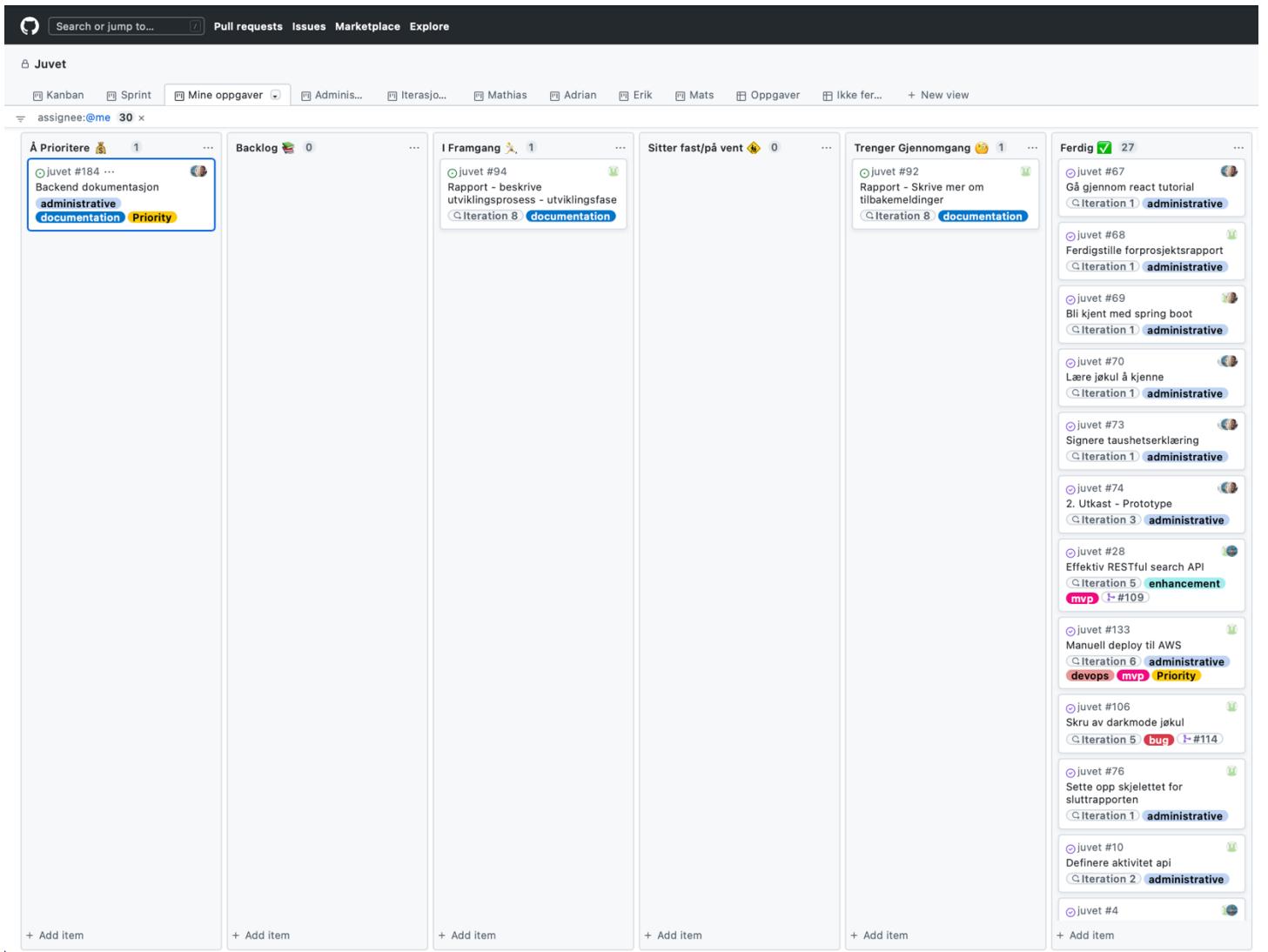
- Å Prioritere**: 6 items. Contains issues like "juvet #184 ... Backend dokumentasjon" and "juvet #9 Draft Implementere funksjonalitet for tags og kategorier". Labels include "administrative", "Priority", and "speed".
- Backlog**: 18 items. Contains issues like "juvet #75 Versoning spring boot application?", "juvet #9 Skrive README.md", and "juvet #78 Sette opp prettier i frontend". Labels include "documentation" and "developer convenience".
- I Framgang**: 2 items. Contains issues like "juvet #102 Rapport - Løsnings oppbygging og virkemåte (gjerne også visuelt)" and "juvet #103 Rapport - Forhold til maskiner/database/OS (funksjonelle grensesnitt) (gjerne visuelt)". Labels include "documentation".
- Sitter fast/på vent**: 1 item. Contains issue "juvet #44 Asynkron backend". Label: "enhancement".
- Trenger Gjennomgang**: 0 items.
- Ferdig**: 110 items. Contains many issues related to documentation and administrative tasks, such as "juvet #33 GitHub Actions frontend tester", "juvet #67 Gå gjennom react tutorial", and "juvet #68 Ferdigstille forprosjektsrapport". Labels include "administrative", "bug", "Priority", "mvp", and "speed".

Each card in the backlog and in progress columns includes a link to the corresponding GitHub issue and a brief description. The "Ferdig" column is notably longer, indicating a backlog of administrative and documentation tasks.

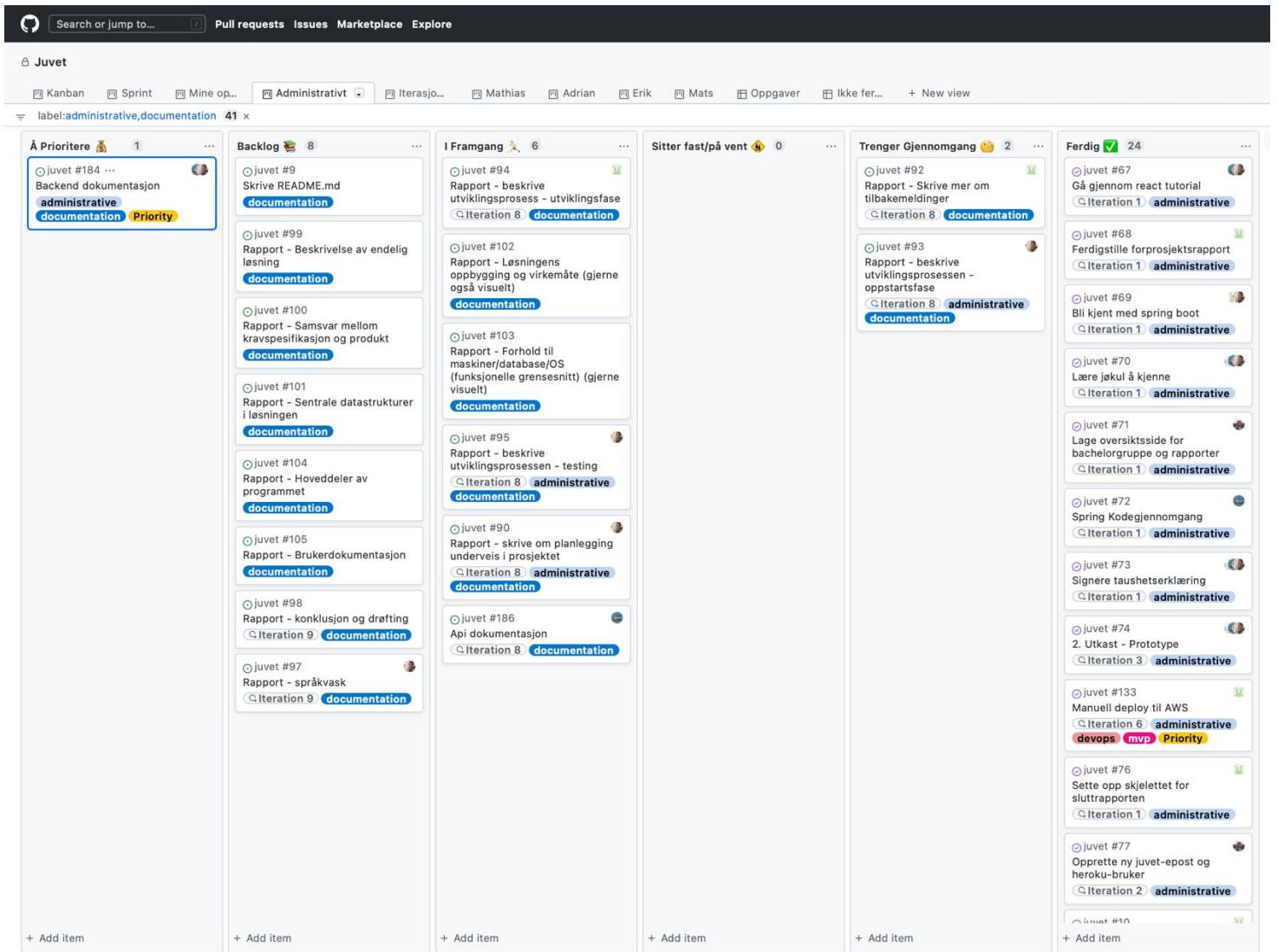
Figur 89 - GitHub prosjektbrett, "Kanban"



Figur 90 - GitHub prosjektbrett, "Sprint".



Figur 91 - GitHub prosjektbrett, "Mine oppgaver".



Figur 92 - GitHub prosjektbrett, "Administrativt".

Search or jump to... Pull requests Issues Marketplace Explore

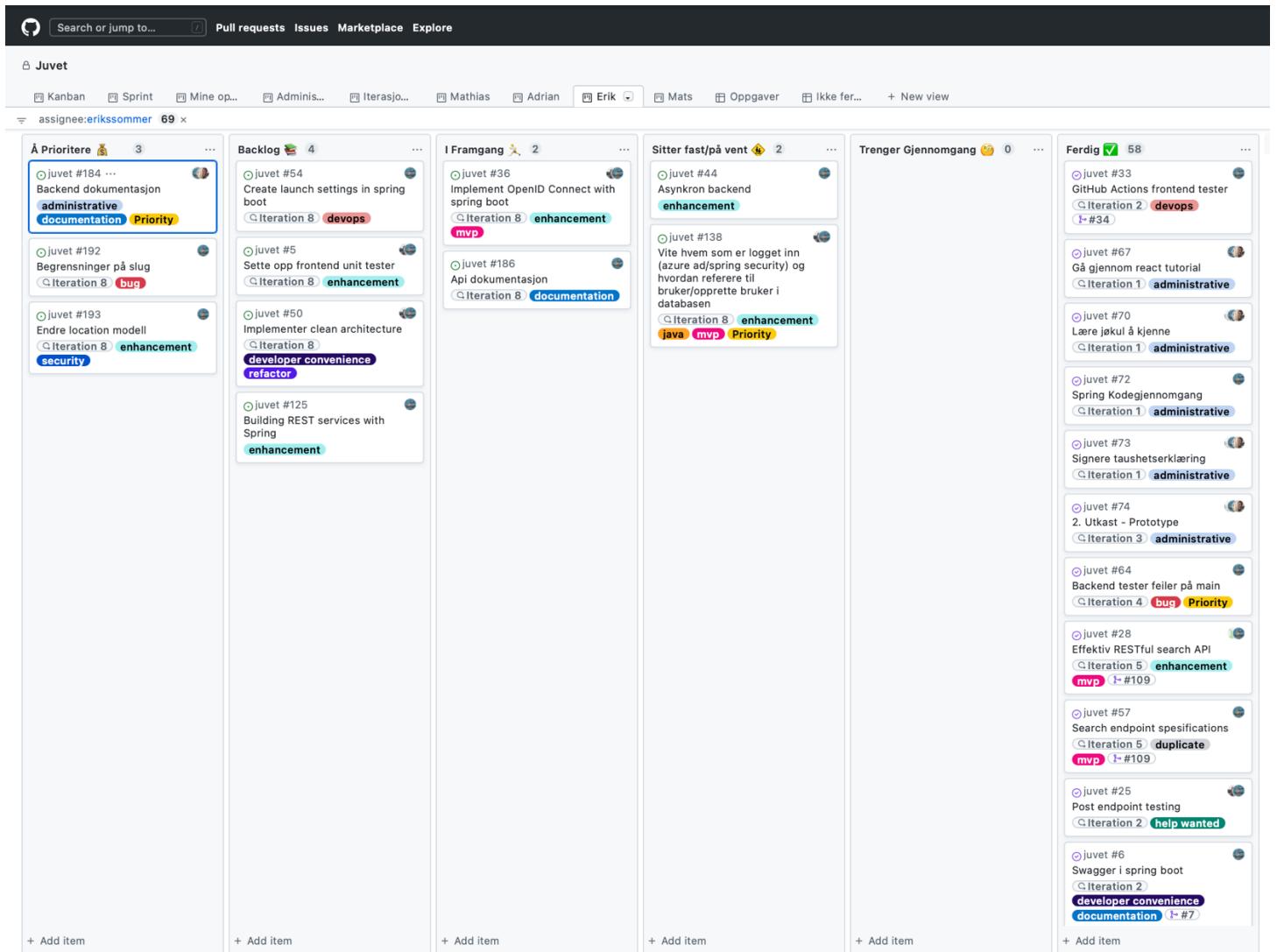
Juvet

Kanban Sprint Mine op... Adminis... Iterasjoner Mathias Adrian Erik Mats Oppgaver Ikke fer... + New view

Filter by keyword or by field

No Iteration 74	Iteration 5 16 Mar 14 - Mar 27	Iteration 6 22 Mar 28 - Apr 10	Iteration 7 23 Apr 11 - Apr 24	Iteration 8 24 Current Apr 25 - May 08	Iteration 9 2 May 09 - May 22
juvet #184 Backend dokumentasjon A Prioritere :moneybag:	juvet #28 Effektiv RESTful search API Ferdig:white_check_mark:	juvet #133 Manuell deploy til AWS Ferdig:white_check_mark:	juvet #110 Rapport - skrive om Design Sprint Ferdig:white_check_mark:	juvet #54 Create launch settings in spring boot Backlog:books:	juvet #98 Rapport - konklusjon og drøfting Backlog:books:
juvet #33 GitHub Actions frontend tester Ferdig:white_check_mark:	juvet #57 Search endpoint spesifikasjoner Ferdig:white_check_mark:	juvet #121 Søk header Ferdig:white_check_mark:	juvet #171 getPreferredContact().getValues() kaster nullpointer Ferdig:white_check_mark:	juvet #5 Sette opp frontend unit tester Backlog:books:	juvet #97 Rapport - språkvask Backlog:books:
juvet #67 Gå gjennom react tutorial Ferdig:white_check_mark:	juvet #41 Lag filter valueObject Ferdig:white_check_mark:	juvet #132 Utforme forsida Ferdig:white_check_mark:	juvet #81 Skriv om docker på rapport Ferdig:white_check_mark:	juvet #50 Implementer clean architecture Backlog:books:	
juvet #68 Ferdigstille prosjektsrapport Ferdig:white_check_mark:	juvet #48 Utforme søkeresultatsiden Ferdig:white_check_mark:	juvet #142 Koble backend og frontend Ferdig:white_check_mark:	juvet #96 Rapport - skrive mer utfyllende om kravsspesifikasjon Ferdig:white_check_mark:	juvet #36 Implement OpenID Connect with spring boot I Framgang:running:	
juvet #69 Bli kjent med spring boot Ferdig:white_check_mark:	juvet #111 Redesign Aktivitetkort Ferdig:white_check_mark:	juvet #118 Stillingstittel på sparrings partner kort Ferdig:white_check_mark:	juvet #91 Rapport - skrive mer utfyllende om smidig metodikk Ferdig:white_check_mark:	juvet #172 Formater tid og datoer på frontend Ferdig:white_check_mark:	
juvet #70 Lære jekul å kjenne Ferdig:white_check_mark:	juvet #106 Skru av darkmode jekul Ferdig:white_check_mark:	juvet #157 Koble til backendserver toggle Ferdig:white_check_mark:	juvet #89 Rapport - skrive mer utfyllende om veiledere Ferdig:white_check_mark:	juvet #153 Lage Min side I Framgang:running:	
juvet #71 Lage oversiktsside for bachelorgruppe og rapporter Ferdig:white_check_mark:	juvet #47 Implement dark mode Ferdig:white_check_mark:	juvet #159 Lage skjerm for søk som feiler eller laster Ferdig:white_check_mark:	juvet #158 Fiks swagger_ui Ferdig:white_check_mark:	juvet #191 Lage aktivitetsopprettelse I Framgang:running:	
juvet #72 Spring Kodegjennomgang Ferdig:white_check_mark:	juvet #66 Gjennomgå feedback fra Øyvind om design Ferdig:white_check_mark:	juvet #65 Deploy til heroku Ferdig:white_check_mark:	juvet #140 User activities completed Ferdig:white_check_mark:	juvet #163 Ferdigstille detaljside Ferdig:white_check_mark:	
juvet #73 Signere taushetsertiklering Ferdig:white_check_mark:	juvet #112 Refactor activities api Ferdig:white_check_mark:	juvet #31 Github actions deploy heroku Ferdig:white_check_mark:	juvet #167 [#152] api changes Ferdig:white_check_mark:	juvet #94 Rapport - beskrive utviklingsprosess - utviklingsfase I Framgang:running:	
juvet #74 2. Utkast - Prototype Ferdig:white_check_mark:	juvet #88 Prototype 3.0 basert på feedback fra Øyvind Ferdig:white_check_mark:	juvet #80 Lage code of conduct Ferdig:white_check_mark:	juvet #152 API Endringer Ferdig:white_check_mark:	juvet #177 Endre alle enums til lowercase Ferdig:white_check_mark:	
juvet #64 Backend tester feiler på main Ferdig:white_check_mark:	juvet #87 String vs Date declaration of date attribute on activity Ferdig:white_check_mark:	juvet #155 Fix frontend tests Ferdig:white_check_mark:	juvet #174 Fjern id fra slutten av sluggen Ferdig:white_check_mark:	juvet #166 Fiks save for later knapp A Prioritere :moneybag:	
Draft Implementere funksjonalitet for + Add item	+ Add item	+ Add item	+ Add item	+ Add item	+ Add item

Figur 93 - GitHub prosjektbrett, "Iterasjoner"



Figur 94 - GitHub prosjektbrett, personlige oppgaver

Search or jump to... Pull requests Issues Marketplace Explore

Juvet

Kanban Sprint Mine op... Adminis... Iterasj... Mathias Adrian Erik Mats Oppgaver ↗ Ikke fer... + New view

Title	Assignees	Status	Iteration	Labels	Må gjøres innen	Kompleksitet
1 Backend dokumentasjon	ATS-Hacker...	Å Prioritere 🐻		administrative docum		
2 Implementere funksjonalitet for tags og kategorier	matssom	Å Prioritere 🐻				
3 Lage filtrering	matssom	Å Prioritere 🐻				
4 Vise til andre kurssider (Udemy etc..) om ingen t	matssom	Å Prioritere 🐻				
5 Vent med entity fetch til liste har lastet på nytt	matssom	Å Prioritere 🐻		refactor speed		
6 Fiks save for later knapp	matssom	Å Prioritere 🐻	Iteration 8	bug		
7 Qna seksjon på aktiviteter	matssom	Å Prioritere 🐻				
8 Begrensninger på slug	erikssommer	Å Prioritere 🐻	Iteration 8	bug		
9 Endre location modell	erikssommer	Å Prioritere 🐻	Iteration 8	enhancement securit	Enkel	
10 Versoning spring boot application?		Backlog 🍔		question		
11 Skrive README.md		Backlog 🍔		documentation		
12 Sette opp prettier i frontend		Backlog 🍔		developer convenience		
13 Next.js vs SPA		Backlog 🍔		question		
14 Create launch settings in spring boot	erikssommer	Backlog 🍔	Iteration 8	devops		
15 Sette opp frontend unit tester	erikssommer...	Backlog 🍔	Iteration 8	enhancement		
16 Implementer clean architecture	erikssommer...	Backlog 🍔	Iteration 8	developer convenience		
17 Lage profilsøk side	matssom	Backlog 🍔				
18 Plassholder bilde og lazy loading på sparringspa	matssom	Backlog 🍔		refactor speed		
19 Lage kalenderoversikt	matssom	Backlog 🍔				
20 jkl-page-navigator håndterer ikke store antall sic	matssom	Backlog 🍔		enhancement	Mellomstor	
21 Rapport - Beskrivelse av endelig løsning		Backlog 🍔		documentation		
22 Rapport - Samsvær mellom kravspesifikasjon og i		Backlog 🍔		documentation		
23 Rapport - Sentrale datastrukturer i løsningen		Backlog 🍔		documentation		
24 Rapport - Hoveddeler av programmet		Backlog 🍔		documentation		
25 Rapport - Brukerdokumentasjon		Backlog 🍔		documentation		
26 Rapport - konklusjon og drøfting		Backlog 🍔	Iteration 9	documentation		
27 Rapport - språkvask	ATS-Hackermai	Backlog 🍔	Iteration 9	documentation		
28 Building REST services with Spring	erikssommer	Backlog 🍔		enhancement		
29 Autocomplete på søk frontend og backend	matssom	Backlog 🍔				
30 Bytte ut a tags med Link	matssom	Backlog 🍔				
31 Implement OpenID Connect with spring boot	erikssommer...	I Framgang 🎉	Iteration 8	enhancement mvp		
32 Lage Min side	matssom	I Framgang 🎉	Iteration 8	enhancement mvp		
33 Lage aktivitetsopprettelse	matssom	I Framgang 🎉	Iteration 8	enhancement mvp	Kompleks	
34 Rapport - beskrive utviklingsprosess - utviklings	matrund	I Framgang 🎉	Iteration 8	documentation		
35 Rapport - Løsningens oppbygging og virkemåte i		I Framgang 🎉		documentation		
36 Rapport - Forhold til maskiner/database/OS (funl		I Framgang 🎉		documentation		
37 Rapport - beskrive utviklingsprosessen - testing	ATS-Hackermai	I Framgang 🎉	Iteration 8	administrative docum		

+ You can use **Ctrl** + **Space** to add an item

Figur 95 - GitHub prosjektbrett, "Oppgaver".

Search or jump to... Pull requests Issues Marketplace Explore

Juvet

Kanban Sprint Mine op... Adminis... Iterasjo... Mathias Adrian Erik Mats Oppgaver Ikke ferdig forrige sprint

iteration:"@previous" is:open 5 ×

Title	Assignees	Status	Linked Pull Requests	Iteration	+
95 Rapport - skrive om Design Sprint	matrund	Ferdig ✓		Iteration 7	
123 Skriv om docker på rapport		Ferdig ✓		Iteration 7	
124 Rapport - skrive mer utfyllende om kravspesifika	matrund	Ferdig ✓		Iteration 7	
125 Rapport - skrive mer utfyllende om smidig metoc	matrund	Ferdig ✓		Iteration 7	
126 Rapport - skrive mer utfyllende om veiledere	matrund	Ferdig ✓		Iteration 7	

You can use Ctrl + Space to add an item

Figur 96 - GitHub prosjektbrett, "Ikke ferdig forrige sprint".

## Vedlegg 5 – Resultat WCAG-test via Siteimprove.com

**Siteimprove**  

---

### High five! You're ahead of the pack.

Your website has only a few accessibility issues. Stay ahead by ensuring accessibility is a part of your design and development process from the start. Maintaining an accessible website is an ongoing journey.



98.2 / 100  
Overall page score



http://juvet-dev-lb-1329289883.eu-west-1.elb.amazonaws.com/

---

### Score breakdown

Your accessibility score is a measure of how well your site performs against the automated Siteimprove accessibility checks. Our checks are based on WCAG success criteria categorized by Level A, AA, or AAA conformance levels. Below you can see how well you scored in each category.



94.7 / 100  
A-level score

You have a few Level A errors to fix, but you're on the right track! Then, you can start to focus more on your AA and AAA level errors.



100 / 100  
AA-level score

Well done! No Level AA errors were found. You can start working on Level AAA errors.



100 / 100  
AAA-level score

Excellent! No Level AAA errors were found. Continue what you are doing to stay ahead.

---

### Overview of automated checks completed

We have checked all the different elements on your page, including forms, images, and links. Here you can see how the checks are categorized by "passed" or "failed" occurrences within each conformance level.

	A	AA	AAA	Total
Passed	55	7	22	<b>84</b>
Failed	2	0	0	<b>2</b>

Figur 97 - Resultat av WCAG-test av Juvets landingsside

## Vedlegg 6 – Hovedklasser og modeller i backend

Activity	
time	String
location	Location
slug	String
level	Level
usersSavedForLater	Set<User>
usersCompleted	Set<User>
shortDescription	String
longDescription	String
title	String
usersSignedUp	Set<User>
thumbnail	Thumbnail
deadline	Date
id	Long
type	String
format	Format
tags	Set<Tag>
duration	String
attachements	Set<Attachement>
authors	Set<User>
date	Date
getSlug()	String
getId()	Long
setTime(String)	void
getLocation()	Location
getThumbnail()	Thumbnail
setLocation(Location)	void
setDuration(String)	void
setDeadline(String)	void
getTitle()	String
getShortDescription()	String
setShortDescription(String)	void
setSlug(String)	void
getLongDescription()	String
setLongDescription(String)	void
setDate(Date)	void
setUsersSavedForLater(Set<User>)	void
setFormat(Format)	void
getAuthors()	Set<User>
getDate()	Date
getAttachements()	Set<Attachement>
setTitle(String)	void
setThumbnail(Thumbnail)	void
setUsersCompleted(Set<User>)	void
getUsersSignedUp()	Set<User>
setId(Long)	void
getDuration()	String
setTags(Set<Tag>)	void
builder()	ActivityBuilder
getUsersSavedForLater()	Set<User>
getFormat()	Format
getType()	String
getDeadline()	String
setLevel(Level)	void
getTags()	Set<Tag>
setAttachements(Set<Attachement>)	void
setAuthors(Set<User>)	void
getUsersCompleted()	Set<User>
getTime()	String
getLevel()	Level
setType(String)	void
setUsersSignedUp(Set<User>)	void
toString()	String
equals(Object)	boolean
hashCode()	int

Figur 98 - Activity-klassen i Juvets backend

Location	
digitalUrl	String
place	Address
digitalPlatform	String
id	Long
activity	Activity
mapsUrl	String
getDigitalUrl()	String
getId()	Long
getDigitalPlatform()	String
getMapsUrl()	String
getPlace()	Address
getActivity()	Activity
setId(Long)	void
setDigitalUrl(String)	void
setDigitalPlatform(String)	void
setMapsUrl(String)	void
setActivity(Activity)	void
builder()	LocationBuilder
setPlace(Address)	void
toString()	String

Figur 100 - Location-klassen i backend

Thumbnail	
activity	Activity
alt	String
id	Long
url	String
getId()	Long
getUrl()	String
getAlt()	String
getActivity()	Activity
setId(Long)	void
setUrl(String)	void
setAlt(String)	void
setActivity(Activity)	void
toString()	String

Figur 99 - Thumbnail-klassen i Juvets backend

Address		
f	city	String
f	streetNumber	String
f	country	String
f	region	String
f	id	Long
f	description	String
f	streetName	String
f	location	Location
f	postalCode	String
f	establishment	String
f	apartmentNumber	String
m	getId()	Long
m	getEstablishment()	String
m	setCity(String)	void
m	setDescription(String)	void
m	getStreetName()	String
m	setRegion(String)	void
m	getStreetNumber()	String
m	getPostalCode()	String
m	setApartmentNumber(String)	void
m	builder()	AddressBuilder
m	setStreetName(String)	void
m	setLocation(Location)	void
m	setCountry(String)	void
m	getApartmentNumber()	String
m	setId(Long)	void
m	getCity()	String
m	getRegion()	String
m	setStreetNumber(String)	void
m	setEstablishment(String)	void
m	getCountry()	String
m	getLocation()	Location
m	getDescription()	String
m	setPostalCode(String)	void

Figur 101 - Address-klassen i Juvets backend

Attachement		
f	url	String
f	id	Long
f	activity	Activity
f	description	String
m	getId()	Long
m	getUrl()	String
m	getDescription()	String
m	getActivity()	Activity
m	setId(Long)	void
m	setUrl(String)	void
m	setDescription(String)	void
m	setActivity(Activity)	void
m	toString()	String

Figur 102 - Attachement-klassen i Juvets backend

Tag		
f	userInterests	Set<User>
f	id	Long
f	type	String
f	userCompetences	Set<User>
f	value	String
f	activities	Set<Activity>
m	getUserInterests()	Set<User>
m	getId()	Long
m	getType()	String
m	getValue()	String
m	getActivities()	Set<Activity>
m	setUserCompetences(Set<User>)	void
m	setUserInterests(Set<User>)	void
m	getUserCompetences()	Set<User>
m	setId(Long)	void
m	setType(String)	void
m	setValue(String)	void
m	setActivities(Set<Activity>)	void
m	toString()	String
m	hashCode()	int
m	equals(Object)	boolean

Figur 103 - Tag-klassen i Juvets backend

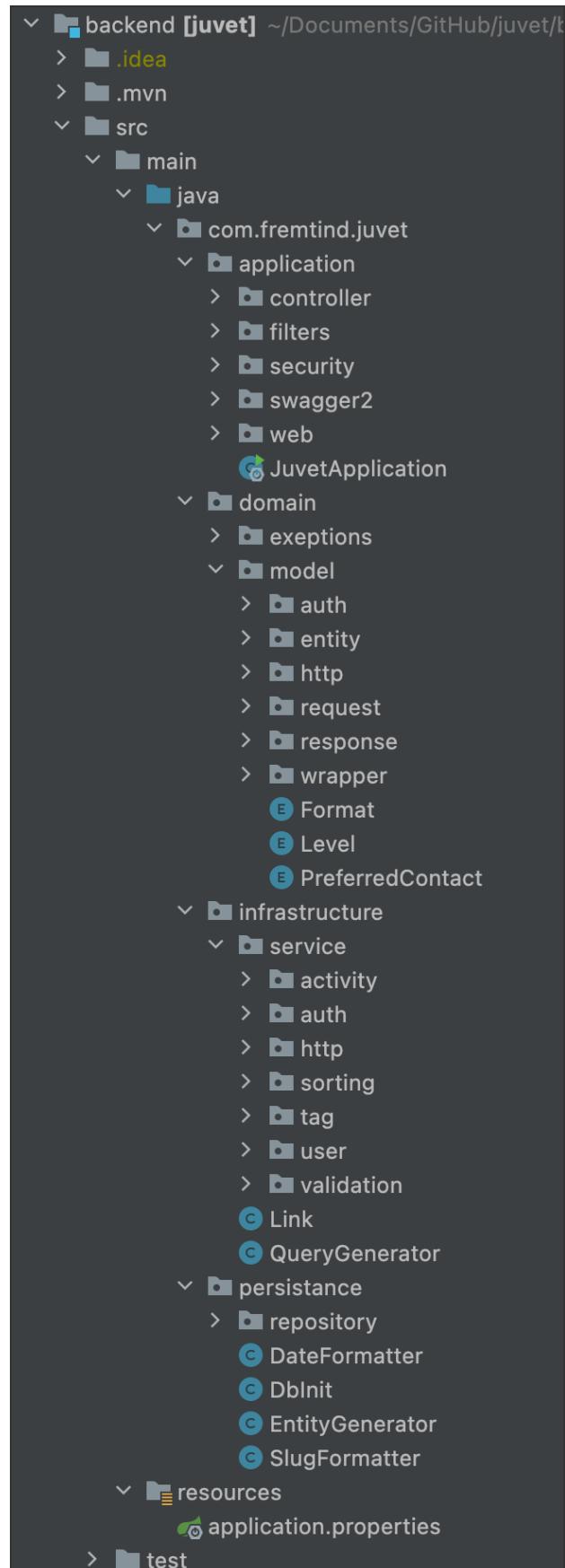
User	
f	id Long
f	avatar String
f	lastName String
f	preferredContact PreferredContact
f	activitiesSignedUp Set<Activity>
f	phoneNumber String
f	email String
f	activitiesSavedForLater Set<Activity>
f	firstName String
f	activitiesCompleted Set<Activity>
f	autherOfActivities Set<Activity>
f	shortForm String
f	slug String
f	competences Set<Tag>
f	title String
f	intrests Set<Tag>
m	getShortForm() String
m	getLastName() String
m	getAvatar() String
m	getId() Long
m	getFirstName() String
m	getEmail() String
m	getActivitiesSignedUp() Set<Activity>
m	setFirstName(String) void
m	setActivitiesSavedForLater(Set<Activity>) void
m	setSlug(String) void
m	getCompetences() Set<Tag>
m	setLastName(String) void
m	setShortForm(String) void
m	setTitle(String) void
m	setAutherOfActivities(Set<Activity>) void
m	getActivitiesSavedForLater() Set<Activity>
m	setIntrests(Set<Tag>) void
m	builder() UserBuilder
m	getActivitiesCompleted() Set<Activity>
m	getAutherOfActivities() Set<Activity>
m	getPhoneNumber() String
m	getPreferredContact() PreferredContact
m	setActivitiesCompleted(Set<Activity>) void
m	getSlug() String
m	setAvatar(String) void
m	setPhoneNumber(String) void
m	setEmail(String) void
m	setPreferredContact(PreferredContact) void
m	getIntrests() Set<Tag>
m	getTitle() String
m	setId(Long) void
m	setCompetences(Set<Tag>) void
m	setActivitiesSignedUp(Set<Activity>) void
m	equals(Object) boolean
m	hashCode() int
m	toString() String

Figur 104 - User-klassen i Juvets backend

ActivityRepository	
(n) <code>findAllByDateBetween(Date, Date)</code>	List<Activity>
(n) <code>findAllByDateBeforeAndTitleContainingIgnoreCaseAndTagsTypeIgnoreCase(Date, String, String)</code>	List<Activity>
(n) <code>findAllByDateBetweenAndTitleContainingIgnoreCaseAndTagsValueIgnoreCase(Date, Date, String, String)</code>	List<Activity>
(n) <code>findAllByDateBetweenAndTagsValueIgnoreCase(Date, Date, String)</code>	Collection<Activity>
(n) <code>findAllByDateAfterOrDateIsNull(Date)</code>	List<Activity>
(n) <code>findBySlug(String)</code>	Activity
(n) <code>findByTitleContainingIgnoreCase(String)</code>	List<Activity>
(n) <code>findAllByDateBefore(Date)</code>	List<Activity>
(n) <code>findAllByDateAfterAndTitleContainingIgnoreCaseAndTagsValueIgnoreCase(Date, String, String)</code>	List<Activity>
(n) <code>findAllByOrderWithTitle()</code>	List<Activity>
(n) <code>findAllByDateBeforeAndTitleContainingIgnoreCase(Date, String)</code>	List<Activity>
(n) <code>findByTagsValueIgnoreCase(String)</code>	List<Activity>
(n) <code>findAllByTitleContainingIgnoreCaseAndTagsValueIgnoreCase(String, String)</code>	List<Activity>
(n) <code>findAllByDateAfterAndTitleContainingIgnoreCase(Date, String)</code>	List<Activity>
(n) <code>findAllByTitleContainingIgnoreCase(String)</code>	List<Activity>
(n) <code>findAllByDateAfterAndTagsValueIgnoreCase(Date, String)</code>	Collection<Activity>
(n) <code>findAllByTagsValueIgnoreCase(String)</code>	Collection<Activity>

Figur 105 – «Activity repository»-grensesnitt i Juvets backend

## Vedlegg 7 – Mappestruktur backend



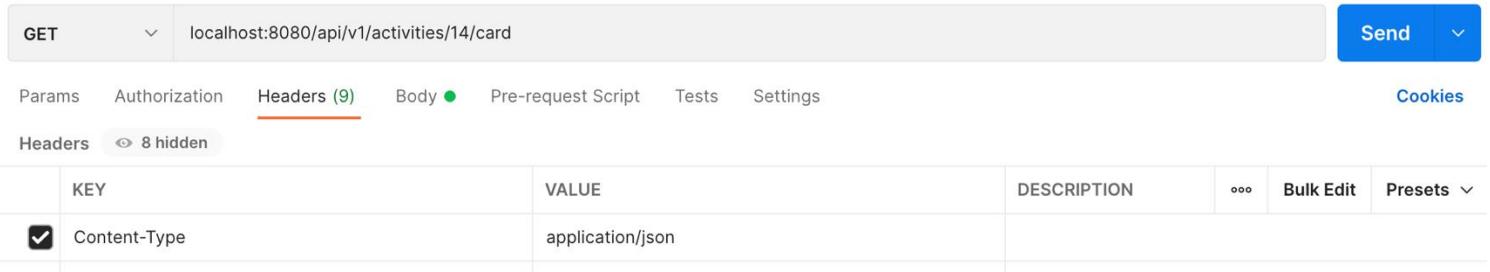
Figur 106 - Mappestrukturen i backend-delen av prosjektet

## Vedlegg 8 – Mappestruktur frontend



*Figur 107 - Mappestrukturen i frontend-delen av prosjektet*

## Vedlegg 9 – Postman-testing og -respons eksempel



The screenshot shows the Postman interface with a GET request to `localhost:8080/api/v1/activities/14/card`. The Headers section is active, containing one entry: `Content-Type: application/json`.

Figur 108 - Eksempel på testing av et endepunkt i Juvet med Postman.

Vellykket respons på spørringen mot endepunktet som er illustrert i bildet over er slik:

```
«  
{  
  "activity": {  
    "id": 14,  
    "slug": "neste-steg-med-spring-boot",  
    "title": "Neste steg med Spring Boot",  
    "shortDescription": "Lorem ipsum dolor sit amet consectetur adipisicing elit.",  
    "thumbnail": {  
      "id": 17,  
      "url":  
        "https://pbs.twimg.com/profile_images/1235868806079057921/fTL08u_H_400x400.png",  
      "alt": "Spring Boot logo"  
    },  
    "duration": "3:00",  
    "authors": [  
      {  
        "name": "Nina",  
        "slug": "nina-vårstøvel",  
        "email": "ninavårstøvel@fremtind.no"  
      }  
    ],  
    "location": {  
      "id": 15,  
      "digitalUrl": null,  
    }  
  }  
}
```

```

    "digitalPlatform": null,
    "mapsUrl":
"https://www.google.com/maps/place/Fremtind+Forsikring+AS/@59.915054,10.7494093,1
7z/data=!4m9!1m2!2m1!1sfremtind!3m5!1s0x46416e63d4745675:0xe7f28d3ac03ac000!8m
2!3d59.9150355!4d10.7515939!15sCghmcmVtdGluZFoKlghmcmVtdGluZJIBEWluc3VyYW5jZ
V9jb21wYW55mgEjQ2haRFNVaE5NRzluUzBWSIEwRm5TVU50Y3pkTU1sbDNFQUU",
    "place": {
        "id": 16,
        "establishment": "Fremtind",
        "streetName": "Hammersborggata",
        "streetNumber": "2",
        "postalCode": "0181",
        "apartmentNumber": null,
        "city": "Oslo",
        "region": null,
        "country": "Norway",
        "description": "8. etg, ved kaffemaskinene"
    }
},
"format": "fysisk",
"type": null,
"level": "nybegynner",
"registration": {
    "attending": false,
    "date": "2022-02-12",
    "time": null,
    "deadline": "2022-02-10",
    "participants": 0
},
"saved": false,
"completed": false
}
}

».

```

En respons på mislykket spørring mot endepunktet kan ha mange ulike statuskoder, men den en av de vanligste er en statuskode 500-error. Det betyr intern server feil, og vil inneholde en feilmelding. Et eksempel på en status 500-error kan se slik ut:



The screenshot shows a REST API response in a browser-like interface. At the top, there are tabs for 'Body' (which is selected), 'Cookies', 'Headers (7)', and 'Test Results'. Below these are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON'. The status bar at the top right indicates 'Status: 500 Internal Server Error', 'Time: 140 ms', and 'Size: 297 B'. The main content area displays a JSON object with three lines of code:

```
1 {  
2   "error": "Klarte ikke å hente aktivitet med id: 140"  
3 }
```

Figur 109 - Respons med feilmelding og 500-status etter spørring mot et API-endepunkt

En annen vanlig feilmeldingsrespons, som har statuskode 404. Den betyr at ressursen som forsøkes å nås med spørringen ikke finnes, eller at den adressen som man forsøker å nå ressursen på er feil. Et eksempel er ved å sende en GET-spørring til «localhost:8080/api/v1/activity», som ikke er definert i vårt API. Derfor får vi denne feilmeldingsresponsen:

«

```
{  
  "timestamp": 1651744850825,  
  "status": 404,  
  "error": "Not Found",  
  "message": "No message available",  
  "path": "/api/v1/activity"  
}
```

»

## Vedlegg 10 – Attest fra oppdragsgiver

Fremtind

OsloMet - storbyuniversitetet  
Pilestredet 46  
0167 Oslo

Oslo, 18.05.2022

Attest – Bacheloroppgaven «Juvet»

Dette bekrefter at *Mathias Rundgreen, Mats Sommervold, Adrian Storset og Erik Sommer* (heretter «studentene») har arbeidet med bacheloroppgave hos *Fremtind Forsikring* i perioden 10.01.2022 til 31.05.2022.

Studentene har levert en teknisk løsning for å koordinere kompetansehevende tiltak hos ansatte i Fremtid Forsikring. Løsningen fikk navnet *Juvet*.

Arbeidet startet med en analyse av problemstillingen gjennom en design sprint. En prototype ble utviklet i løpet av den første uken og denne ble testet på et antall ansatte. Den tekniske løsningen ble deretter utviklet basert på læring fra prototypen og gjennom samarbeid med Fremtids tekniske eksperter. Den ble levert til Fremtind Forsikring ved utviklingsleder Glenn Brownlee den 18.05.2022.

Arbeidet som er gjennomført er av stor verdi for Fremtind og effektiviteten og kompetansen som ble vist har vært imponerende gjennom hele prosessen.

Vi takker studentene for innsatsen og ønsker dem lykke til videre i sine karrierer.

Vennlig hilsen

Glenn A. Brownlee  
Utviklingsleder  
Distribuerte Løsninger / Fremtind Digital  
Fretnind Forsikring AS

**Fremtind Forsikring AS** Hammersborgata 2 Telefon 09020  
Boks 778 Sentrum Telefaks 21 02 50 51  
0106 Oslo [www.fremtind.no](http://www.fremtind.no)

*Figur 110 - Attest fra oppdragsgiver Fremtind*