Strai Kjøkken montør-app

IS-304-1 21V Bacheloroppgave i informasjonssystemer

Forord

Vi vil gjerne takke Strai-kjøkken for samarbeidet på bachelor-prosjektet vårt. Takk til Espen Cederberg, IT-sjef hos strai, for den tette oppfølgingene, veiledningen, gode tilbakemeldingene og de hyggelige møtene. Takk til de andre ansatte hos Strai; prosjektavdelingen, salgsavdelingen, montører og andre som har satt av tid til møter og tilbakemelding. Uten dere hadde vi ikke klart å produsere det vi har klart.

I løpet av dette semesteret har Norge vært påvirket av Covid-19 og konsekvensene det medfølger. Mange studenter har måttet endre arbeidsmetoder grunnet nedstengning av arbeidsplasser og universiteter. Vår gruppe har likevel vært så heldige å ha tilgang til to kontorer hos Strai, samt hatt muligheten til å kommunisere ansikt til ansikt med arbeidsgiver hver eneste dag. Vi vil derfor spesielt takke administrasjonen til Strai, som har tilrettelagt for at vi på best mulig måte har kunnet gjennomføre prosjektet vårt .

Vi vil også takke Janis Gailis for veiledningen hans gjennom prosjektet, samt emneansvarlig Hallgeir Nilsen for hans innspill og bidrag. I tillegg vil vi også takke de andre medelevene våre, det å få innspill i hvordan andre har arbeidet samt tilbakemeldingene på vår arbeidsmåte har vært ekstremt lærerikt.

Emnekode	IS-304
Emnenavn	Bacheloroppgave i informasjonssystemer
Emneansvarlig:	Hallgeir Nilsen
Veileder	Janis Gailis
Oppdragsgiver:	Strai Kjøkken

Studenter:

Etternavn	**Fornavn**
Grønås	Runar
Stenberg	Martin
Sæther	Sindre Edvard
Tran	Marcus

Jeg/vi bekrefter at vi ikke siterer eller på annen måte bruker andres arbeider uten at dette er oppgitt, og at alle referanser er oppgitt i litteraturlisten.	JA X	NEI __
Kan besvarelsen brukes til undervisningsformål?	JA X	NEI __
Vi bekrefter at alle i gruppa har bidratt til besvarelsen	JA X	NEI __

Abstrakt

Dette prosjektet dekker oppgaven som gruppen vår ble gitt av Strai Kjøkken. Gjennom deres digitaliseringsprosess, ønsket de å forbedre kommunikasjonen mellom selger, administrasjon og montør. Gruppen ble derfor gitt i oppgave å utvikle en applikasjon, som gjør denne kommunikasjonen enklere, samt minsker mengden “papirarbeid” montørene må utføre i sammenheng med montasje. I applikasjonen skal Strais montører i hovedsak kunne logge inn, finne oversikt over prosjekter og ordre, hente ut plantegninger for montasje, laste opp bilder og skrive kommentarer.

For å løse oppgaven, arbeidet gruppen i henhold til Scrum-rammeverket. Planleggingsfasen av prosjektet omhandlet å sette til grunne hvilke teknologier og funksjoner systemet måtte ha, samt opprette en viss plan over hvordan utviklingen skulle gjennomføres. Dette ble gjort ved hjelp av teoretisk research, og intervjuer av sluttbrukere, som inkluderer ansatte hos Strai og montører. Deretter har gruppen utført 5 sprinter á 1 måned, med utvikling av database, backend og frontend. I løpet av hver sprint har det i tillegg blitt gjennomført minst ett møte med sluttbrukere, for å presentere det gruppen har utviklet så langt, samt gi brukerne mulighet for å komme med forslag og kritikk.

Gjennom dette prosjektet har gruppen måttet tilegne seg mye ny kunnskap. Gruppen har måttet sette seg inn nye rammeverk som Jersey for RESTful APIer i Java, og React for frontend utvikling. I tillegg har gruppen også måttet lære seg et helt nytt programmeringsspråk i form av TypeScript. Gruppen har hatt mye frihet i valg av teknologier og design, noe som har gjort at gruppen har tilegnet seg god erfaring i alle deler av et systemutviklingsprosjekt, både innen analyse, design og programmering. Denne friheten har også resultert i at gruppen har måttet jobbe på en agil måte, hvor en ikke alltid kan følge en rigid plan.

Resultatmessig er produktet nesten ferdigstilt. Gruppen har utviklet en database i Mssql, en REST-API i Java, og en Progressive Web App (PWA) i React. De viktigste funksjonene er implementert og testet opp mot testdata. Applikasjonen er dog ikke testet opp mot serveren til Strai. Dette er et resultat av at implementasjonen av Strais nye ERP-system (Odoo) er blitt forsinket. Strai er uansett godt fornøyd med det gruppen har levert, og er klare for å ta over utviklingen av prosjektet. Strai håper å kunne lansere produktet denne høsten.

Innholdsfortegnelse

1.0 Introduksjon 5
1.1 Informasjon om Strai Kjøkken 5
1.1.1 Systemet i dag 5
1.2 Vårt prosjekt 5
2.0 Sentrale valg 5
2.1 Prosjektstyringsverktøy 6
2.2 Scrum 6
2.2.1 Andre agile rammeverk 6
2.2.2 Scrum team 6
2.2.3 Sprint 7
2.2.4 Sprint-møte 7
2.2.5 Backlog 7
2.2.6 Gruppens anvendelse av scrum 7
2.3 Kvalitetssikring 8
2.4.0 Risikoanalyse 8
2.4.1 Interne risikoer 9
2.4.2 Eksterne risikoer 9
2.5 Teknologi 9
3.0 Gjennomføring av prosjektet 11
3.1 Prosjektstyring 11
3.1.1 Roller 11
3.1.2 Gruppekontrakt 12
3.2 Planlegging og Analyse 12
3.2.1 Prosjektets scope - Brukerhistorie. 12
3.2.2 Planlegging 13
3.3 Backlog 13
3.4 Gjennomføring av sprints 14
3.4.1 Pre sprint 15
3.4.2 Sprint 1-2 15
3.4.2.1 Programmeringsspråk i backend 15
3.4.2.2 Databasestruktur 15
3.4.2.3 Sjekkliste 16
3.4.3 Sprint 3-5 16
3.4.3.1 Design 16

3.4.3.2 Fra Native app til PWA skrevet i React 17
4.0 Produktet 18
5.0 Refleksjon 18
5.1 Utfordringer 18
5.1.2 For mye frihet 19
5.2 Sentrale valg 19
5.2.1 Scrum 19
5.2.2 Kvalitetssikring 19
5.3 Endringer i prosjektet 20
5.3.1 Endringer av scope 20
5.3.2 Endringer av teknologi 20
5.3.3 Testing 21
5.3.3.1 Testing av API 21
5.3.3.2 Testing av frontend 21
5.3.3.3 Refleksjon 21
5.4 Wireframes 22
6.0 Sitat fra oppdragsgiver 22
6.1 Prosjektbeskrivelse fra oppdragsgiver 22
6.2 Gjennomføring 22
7.0 Egenevaluering 23
7.1 Martin Vottestad Stenberg 23
7.2 Marcus Tuan Tran 23
7.3 Runar Schjønning Grønås 23
7.4 Sindre Edvard Sæther 24
8.0 Kilder 24
9.0 Figurliste 26
10.0 Appendix 26
10.1 Bilder av original montør-app 26
10.2 Bilder av statisk iOS app (Før PWA) 27
10.3 Bilder av sluttproduktet. 28
10.4 Kodeeksempler 29
10.5 Gruppekontrakt 31
10.6 Kobling mellom merge request og backlog 33

1.0 Introduksjon

I bachelor-faget IS-304 skal gruppen gjennomføre et IT/IS-relatert prosjekt, samt definere kvalitets- og styringsmetoder i prosjektet. (UiA, 2021)

Gjennom denne rapporten skal vi dokumentere hvordan vår gruppe gikk fram for å produsere en applikasjon for Strai Kjøkken. I de neste avsnittene vil vi introdusere samarbeidspartneren vår Strai Kjøkken, vårt prosjekt og litt kontekst rundt prosjektet.

1.1 Informasjon om Strai Kjøkken

Strai Kjøkken er en norsk familiebedrift som produserer kjøkkeninnredninger. Bedriften har røtter helt tilbake til 1930-tallet, og har siden den gang blitt styrt av fire generasjoner med Tobiassen familiemedlemmer. Strai har i nyere tid investert tungt i moderne produksjonsteknologi, og er inn i en ny digitaliseringsprosess. I 2021 skal de også implementere et nytt ERP-system hvor alle avdelinger nå skal samles under én løsning.

1.1.1 Systemet i dag

I skrivende stund bruker Strai Kjøkken mange små, forskjellige og selvutviklede datasystemer. Odoo er et Open Source, modulært ERP-system som skal erstatte systemene Strai bruker i dag. Strai bytter ut sitt tidligere ERP-system, Ordreboken, for å ha et ERP-system som tilrettelegger mer for deres unike arbeidsprosesser. I tillegg til dette ønsker Strai å forbedre kommunikasjonen mellom selger, administrasjon og montør, og det er her vårt prosjekt kommer inn.

1.2 Vårt prosjekt

Strai har en montør-app som en av deres montør-bedrifter bruker til loggføring og henting av plantegninger for montasje. Denne appen brukes dog kun av dette ene montør-firmaet, i Kristiansand-avdelingen. Vårt prosjekt baserer seg på å ta utgangspunkt i dette konseptet, og lage en mer moderne og funksjonell kode-stack, samt gjøre den tilgjengelig for alle Strais kontorer og montører. Bruker-basen for denne applikasjonen vil trolig være på rundt 200 personer.

Prosjektet vil i hovedsak gå ut på å utvikle en REST-API som inneholder et login-system, en database for lagring av kommentarer og bilder, samt en tilkobling til Strais nye ERP-system for henting av informasjon om prosjekter og ordre. Det skal også utvikles en frontend-app som skal snakke med backend-serveren, og som skal fungere på alle enheter (web, Android, iOS).

2.0 Sentrale valg

Prosjektet hadde ganske åpne retningslinjer fra arbeidsgivers side, både når det kom til teknologi, struktur og scope generelt. For oss som gruppe var det derfor viktig å ta noen sentrale valg, for å vite hvordan vi skulle strukturere arbeidsprosessen vår gjennom prosjektet.

I dette kapittelet går vi over de sentrale valgene vi har gjort.

2.1 Prosjektstyringsverktøy

I dette prosjektet valgte vi i utgangspunktet å bruke GitLab som vårt hoved-prosjektstyringsverktøy. GitLab lar oss samle alt av versjonskontroll, kode og Scrum-informasjon på ett sted. Vi har valgt å bruke GitLab sine “Issue boards” for å samle og sortere backlogen vår. På denne måten får vi en sentral kilde for all informasjon.

I senere tid gikk vi over på GitHub for versjonskontroll med mål om å integrere en automatisk kvalitetskontroll av koden og for å gjøre overtakelsen av prosjektet simplere for prosjekteier.

2.2 Scrum

I dette avsnittet vil vi gå over hvorfor vi har valgt Scrum, noen av hovedelementene til dette rammeverket og hvordan vi har implementert de.

Under prosjektet valgte gruppen å følge den smidige utviklingsmetodikken Scrum. Scrum baserer seg på å bryte ned kompliserte oppgaver i mindre deler, for å håndtere hver del i en angitt tidsperiode kalt sprint. Metoden har forskjellige teknikker og eventer for å kontinuerlig gjennom prosjektet bearbeide og videreutvikle arbeidsprosesser, gruppen og produktet. (Sutherland, Schwaber, 2020)

2.2.1 Andre agile rammeverk

Gruppen vurderte også å bruke metoden Kanban, men ettersom ingen i gruppen hadde noe erfaring med denne metoden, bestemte vi oss for å ikke ta den i bruk. Kanban er en styringsmetode for arbeidsflyt som i motsetning til Scrum ikke nødvendigvis trenger å bli bygd opp av iterasjoner. Hovedtanken bak Kanban er å visualisere prosesser slik at gruppen får oversikt og flyt i prosjektet. Da kan produktet utvikles i en stor utviklingssyklus. (Skaug, 2013) Gruppen hadde derimot ganske god kjennskap til Scrum-rammeverket, og visste hvor fleksibelt det var. Dette gjorde det enklere for gruppen å anvende Scrum, i motsetning til andre agile rammeverk.

2.2.2 Scrum team

Et scrum team består av få medlemmer med forskjellige roller. Disse rollene kan bli delt i tre: utviklere, prosjekteier og Scrum master. (Scrum, 2019)

- Utviklere står ansvarlig for å lage en plan for hver sprint, opprettholde kvalitetskravene, tilpasse arbeidsplan og holde resten av gruppen ansvarlig.
- Prosjekteiers mål er å effektivisere og maksimere resultatet til gruppen. Oppgaven omhandler det å opprettholde sortering og rangeringer av backlogen. Samt kommunisere et klart produkt mål til resten av gruppen.
- Scrum master har ansvaret for at Scrum-metodikken blir fulgt. Denne personen er også ansvarlig for å passe på at teorien og praksisen er godt forstått av både gruppen og organisasjonen.

2.2.3 Sprint

Scrum er basert på sprinter. En sprint pleier å være tidsbegrenset og varer som regel en måned eller mindre. På den tiden skal utviklerne legge fram deler som er "ferdig", brukbare og potensielt klar for lansering. (Scrum, 2019)

2.2.4 Sprint-møte

I starten av hver sprint vil det foregå et sprint-møte, hvor produkteier og gruppen samles for å diskutere og planlegge neste sprint. Målet med dette møtet er å komme til enighet om hvilke funksjoner fra backloggen som burde bli gjort i løpet av sprinten. Produkteier skal da delegere arbeidet og gi instruksjoner til gruppemedlemmene. Gruppen skal også informere produkteier om hvor mye de regner med å få til på den tiden de har. (Schwaber, 2004, s. 11)

2.2.5 Backlog

Backloggen er en liste med funksjonalitet som ikke er lagt til i prosjektet enda. Det er produkteier som har ansvar for å oppdatere og prioritere hvilke funksjoner som er viktig i backloggen. Backloggen er veldig dynamisk ettersom en kan legge til, prioritere eller fjerne funksjonalitet etter hver sprint. (Cohn, 2010, s. 235)

2.2.6 Gruppens anvendelse av scrum

Gruppen valgte å ta i bruk Scrum på bakgrunn av vår forkunnskap, samt at gruppen selv mente at denne metoden passet prosjektet godt. Gjennom bruken av Scrum forsikret gruppen seg om at kommunikasjonen mellom gruppemedlemmer og arbeidsgiver ble opprettholdt. Det passet også prosjektet godt ettersom Scrum er et veldig fleksibelt rammeverk, som ga oss muligheten til å gjøre endringer basert på vår egen arbeidsprosess.

Vi har valgt å gjennomføre daily standups på begynnelsen av hver dag. På dette møtet deltar gruppens medlemmer og diskuterer hva som skal gjøres denne dagen, samt eventuelle problemer eller utfordringer vi har hatt dagen før. Vi delte også erfaring og lærdom med andre medlemmer, noe som var helt essensielt for at dette prosjektet skulle klare å bli gjennomført.

Det ble også tatt i bruk weekly standup som er et møte vi gjennomfører med produkteier, hvor det blir gjennomgått og diskutert hva gruppen har produsert den uken. Eventuelle problemer gruppen har hatt med oppgaven blir diskutert, og hvis nødvendig blir store endringer bestemt.

Når en sprint ble gjennomført, ble et større ukentlig møte arrangert med arbeidsgiver hvor gruppen viste fram hva som hadde blitt gjort i løpet av sprinten, samt hvorvidt målet for sprinten var nådd. Under dette møte ble også backlogen oppdatert, samt at målet for neste sprint ble definert.

I starten av hver sprint ble det også arrangert et "statusmøte" hvor relevante ansatte utenom IT-avdelingen ble invitert. På dette møte viste gruppen fram designvalg og hvordan appen så

ut på tidspunktet. Gruppen fikk også tilbakemelding og ønsker om funksjoner og designendringer.

Gruppen valgte å bruke Gitlab issues for å systematisere backlogen, på denne måten ble det også lett å koble backlogen opp mot koden.

2.3 Kvalitetssikring

Gruppen vår består av medlemmer med varierende mengde programmeringserfaring. På grunn av dette har gruppen tidlig opprettet en kodestandard for prosjektet. Kodestandarden innebærer at vi følger camelcase-strukturen og at alle gruppens medlemmer bruker pluginen ESLint når mulig. ESlint er et program som analyserer kode og ser etter problemer. ESLint kan konfigureres til å håndheve et hvert team sin kodestandard ved hjelp av deres mange forskjellige regler. (OpenJS, u.å) I vårt prosjekt har vi definert kodestandard ved å bygge på ESLint sin anbefalte konfigurasjon med noen ekstra regler lagt til. (OpenJS, u.å.) Gruppen har også laget en manual/dokumentasjon av endepunktene. Dette er ikke bare for å sikre kvaliteten av koden men også for å gjøre det lettere for oppdragsgiver å videreutvikle programmet i senere tid.

Grunnet den korte tidsperioden prosjektet går over, valgte vi å følge en “mobile first”-metode under utvikling av applikasjonen. Denne fremgangsmåten baserer seg på å ta utgangspunkt i den minste skjermstørrelsen først. Målet er å gjøre brukeropplevelsen så sømløst som mulig mellom alle plattformer. (Tran, 2019)

For vår API har vi brukt Jersey, et rammeverk for Java for å lage RESTful APIer. Jersey er en implementasjon av JAX-RS spesifikasjonen, som definerer hvordan en lager REST-APIer i Java. Jersey gjør det enkelt for oss å definere de forskjellige ressursene vi bruker i vårt system, og håndterer store deler av komplikasjonene ved å motta HTTP-kall. Dette har gjort at vi i store deler ikke har behøvd å tenke på hvordan en skriver en HTTP-server og heller bruke vår tid på å skrive funksjonaliteten vi skulle implementere. (Oracle Corporation, 2021)

På web-appen vår har vi brukt rammeverket React og TypeScript. React lar oss lage mindre, gjenbrukbare komponenter, slik at vi kan bygge vår web app på en modulær måte (Facebook, 2021). Vi har brukt TypeScript framfor vanlig JavaScript, for å gi oss statiske typer i koden. Bruken av TypeScript har gjort at vi må klart definere hvilke typer data som blir gitt til de forskjellige komponentene, og skrive kode for å håndtere hva som skal skje dersom data mangler, for eksempel dersom en bruker spør etter en ressurs som ikke eksisterer. (Microsoft, 2021)

2.4.0 Risikoanalyse

I starten av prosjektet valgte vi å diskutere mulige risikoer vi ser på som relevante for dette prosjektet. Disse risikoene var med på å forme planleggingen og utføringen av prosjektet.

Vi har valgt å dele risikoene våre i to deler; eksterne og interne risikoer.

2.4.1 Interne risikoer

Interne risikoer ser vi på som mulig problemer som kan oppstå direkte koblet til arbeidet vi gjennomfører. Dette er risikoer som uventede bugs, problemer med utstyr, mangel på kunnskap, endringer av prosjektbeskrivelser osv.

Som nevnt tidligere er det varierende mengde programmeringserfaring hos gruppens medlemmer, som gjør at prosjektet krever store mengder research for å finne ut hvordan man skal løse diverse oppgaver. Dette gjør at mye tid kan bli brukt på å finne, forstå og bruke teknologier vi ikke kjenner til. Vi har valgt å strukturere gruppen vår med en teknologi-ansvarlig, dette er fordi det nevnte medlemmet har god erfaring som utvikler og kjenner til flere verktøy vi kan bruke. Dette sammen med daglige diskusjoner både innad i gruppen og med arbeidsgiver gjør at vi raskt har klart å opparbeide oss nødvendig kunnskap for å utvikle applikasjonene.

En annen stor risiko gruppen har måttet behandle er hvordan systemets arkitektur skal se ut. Appen vi utvikler er i stor grad avhengig av hvordan arbeidsgivers nye ERP-system behandler informasjonen applikasjonene trenger. Dette ERP-systemet var i utviklingsfasen i løpet av prosjektet vårt. Selv om dette ga oss flere fordeler i form av at vi kunne påvirke hvordan systemet fungerte, gjorde det også at det var noe usikkerhet rundt hva slags informasjon vi får fra det nye ERP-systemet. Det var også en risiko at ting kunne endre seg underveis. Får å unngå at dette skulle bli et problem, har vi hatt tett kontakt med arbeidsgiver, samt at arbeidsgiver har latt oss komme med innspill om hva vi tror er best å gjøre på deres del av systemet.

2.4.2 Eksterne risikoer

En ekstern risiko ser vi på som mulige problemer som kan oppstå uten at gruppen vår kan forhindre det eller påvirke det. I vårt tilfelle er det spesielt én risiko vi ser på som utfordrende.

Gruppen vår har fått tilgang til 2 kontorer hos arbeidsgiver. Dette har bidratt til å effektivisere arbeidet til gruppen. Det har gjort det lettere for oss å arbeide effektiv, kommunisere mellom gruppens medlemmer og arbeidsgiver, samt behandle prosjektet mer profesjonelt.

På grunn av dagens situasjon med tanke på Coronaviruset, måtte vi tidlig i prosjektet ta høyde for muligheten for at det kunne bli krav om hjemmekontor i Kristiansand. Etter en diskusjon med arbeidsgiver ble det satt opp en VPN, slik at vi fremdeles kunne ha tilgang til prosjektet. Heldigvis ble dette aldri nødvendig.

2.5 Teknologi

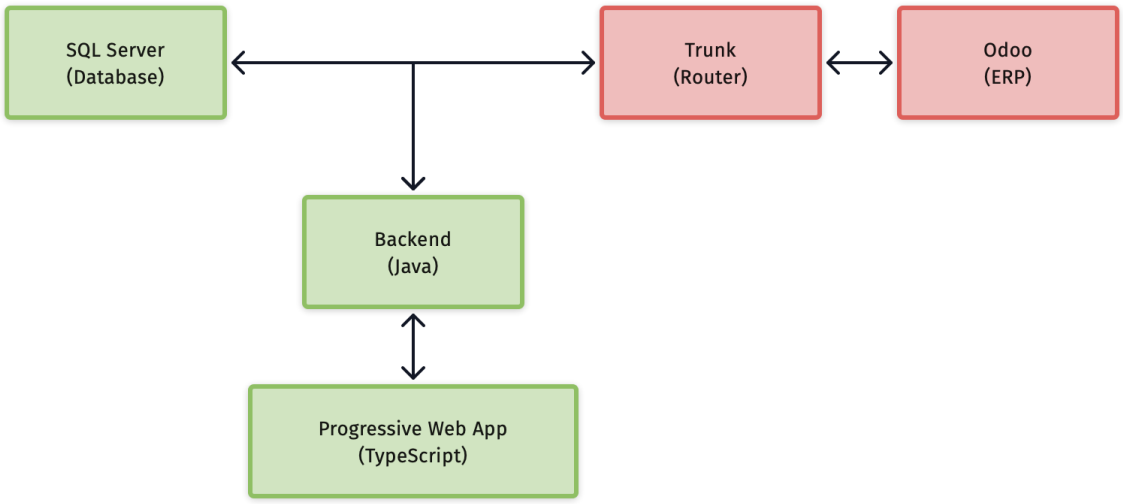
I dette prosjektet brukte vi de følgende teknologiene.

- Java, Typescript, programmeringsspråk.
- Jersey, back-end rammeverk.
- React, front-end rammeverk.
- React-Bootstrap, design.
- Microsoft SQL Server, database.

Det var ingen krav satt til teknologier fra arbeidsgiver, noe som gjorde det mulig for gruppen å velge nøyaktig det vi selv ønsket. Gruppen valgte derfor å bruke Java til vår REST API, og React med typescript til frontend. Dette gjorde at gruppen kunne arbeide på en backend i et programmeringsspråk som vi er kjent med, samtidig som vi kun trengte å utvikle én frontend-applikasjon kompatibel for alle plattformer.

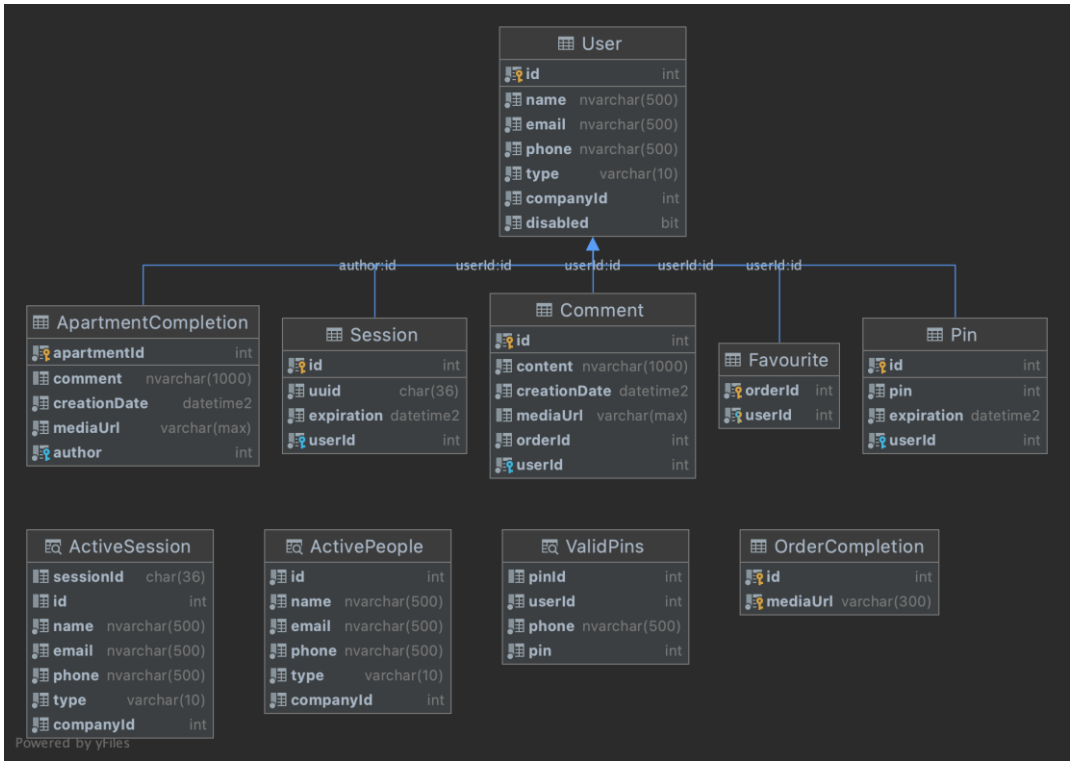
Grunnet gruppens lite erfaring med frontend design valgte også gruppen å ta i bruk React-Bootstrap for det meste av styling. React-Bootstrap er en pakke somme inneholder React-komponenter for UI-design.(React Bootstrap, 2021) I tillegg valgte vi også å bruke React-Icons som er en pakke med grafiske ikoner. (React Icons, 2021)

Samtidig som vår backend skal håndtere data fra vår database, skal den også håndtere data fra Strai sit intern system. På bakgrunn av dette valgte vi tidlig i utviklingen å bruke samme type database som Strai for å unngå mulige problemer med vår databasestruktur.



Figur 1 - Systemarkitektur

Figuren over er en modell av prosjektets systemarkitektur. Fargen grønn tilsvarer elementer vi utvikler selv, og de i rød tilsvarer Strai sitt interne system. Trunk fungerer som et mellomledd mellom vårt system og Strai sitt nye ERP-system. På bildet under, ser en databasestrukturen som gruppen har utviklet i SQL Server. Databasen har som hovedoppgave å håndtere brukere, bilder og kommentarer i applikasjonen.



Figur 2 - Databasestruktur

3.0 Gjennomføring av prosjektet

I sin helhet består dette prosjektet av en pre sprint, og 5 sprints. Når dette prosjektet blir levert er vi fremdeles i sprint 5. Denne sprinten innebærer for det meste å overføre prosjektet til arbeidsgiver, samt avslutte eventuelt arbeid som gjenstår.

Gjennom dette kapitlet skal vi gå gjennom hvordan prosjektet utformet seg, samt hvordan gruppen valgte å gjennomføre prosjektet.

3.1 Prosjektstyring

Mye av motivasjonen til hvorfor gruppen ønsket å delta på akkurat dette prosjektet, var at gruppen ønsket å arbeide på et “reelt” prosjekt som kom til å bli tatt i bruk, og ikke bare et “proof of concept”. På bakgrunn av dette valgte gruppen å følge en streng, strukturert prosjektstyrings-form. I dette kapitlet skal vi dokumentere noen av valgene som ble tatt, samt hvordan prosjektet utformet seg i forhold til prosjektstyring.

3.1.1 Roller

I den første pre-sprinten valgte gruppen å definere en Scrum master / team leder. Denne personer tok hovedansvar for gruppens kommunikasjon med arbeidsgiver, arrangering av møter og eventuelt andre utfordringer som dukket opp. Denne personen tok også ansvaret for å føre dagbok, samt dokumentere møter og oppdatere backlog.

Vi har valgt at vår kontaktperson hos Strai er definert som prosjekteieren. I forhold til Scrum bidrar prosjekteieren på oppretting av backlog og utarbeiding av plan, men hovedansvaret

for oppretting, oppdatering og vedlikehold av disse er i vårt tilfelle gitt til Scrum master. Dette er gjort fordi prosjekteieren ikke har så mye erfaring med Scrum og har begrenset tid til å nøye følge opp prosjektet.

Det ble også valgt en git master / teknologi ansvarlig. Denne personens oppgaver er å opprettholde git, gjennomgå “merge requests”, opprettholde sentrale teknologier, samt passe på at gruppen fulgte kodestandarder.

Selv om noen av gruppens medlemmer fikk tildelt roller, valgte gruppen å ikke følge noe hierarkisk system, men heller gå ut ifra at alle medlemmers meninger og ideer skulle bli vurdert på lik linje. Alle gruppens medlemmer skulle delta i alle deler av arbeidsprosessen, fra undersøkelse, til utvikling, til rapportskriving.

3.1.2 Gruppekontrakt

Før prosjektet startet skrev alle gruppens medlemmer under på en gruppekontrakt. Kort oppsummert består gruppekontrakten av regler og retningslinjer for hvordan gruppen skulle arbeide på prosjektet og hvordan en skulle behandle eventuelle problemer. Gruppekontrakten ligger i appendix.

3.2 Planlegging og Analyse

Som nevnt startet prosjektet med en pre-sprint. I denne perioden fokuserte gruppen på å analysere det eksisterende systemet og arbeidsprosessen til brukerne. Gruppen definerte noen brukerhistorier og lagde en prosjektplan.

3.2.1 Prosjektets scope - Brukerhistorie.

På grunn av det noe udefinerte omfanget av prosjektet ble scope vanskelig å definere i starten av prosjektet. Brukerne hadde mange ønsker om hvilke funksjoner applikasjonen skulle inneha, og hvordan den skulle se ut designmessig. Gruppen tok høyde for disse og la fram en plan. Etter hvert som vi kom lengre ut i prosjektet, fant vi ut at det var en del funksjoner vi måtte nedprioritere. Dette for å holde mengden arbeid på et nivå som vår gruppe på bare 4 medlemmer kunne håndtere. I utgangspunktet var derfor prosjektets scope veldig stort, men ble gjennom prosjektet snevert inn.

For å lettere definere prosjektets scope, valgte gruppen å lage noen brukerhistorier for å demonstrere hva brukerne ønsket fra systemet. En brukerhistorie er en kort setning som beskriver et mål en bruker har når han eller hun bruker systemet. I utgangspunktet ble et antall brukerhistorier laget, men gjennom prosjektet falt flere og flere av disse bort.

Bruker historiene var basert på intervjuer og møter med ansatte hos Strai og montør-firmaer. Målet med disse møtene var å kartlegge dagens arbeidsprosess, samt få innspill til mulige nye funksjoner som appen kunne inkludere.

Det ble veldig klart gjennom dette prosjektet at brukerne hadde mange ønsker og tanker om hvordan applikasjonen skulle fungere. I mange tilfeller ønsket de funksjoner som det i våre øyne ikke ga mening at vårt system skulle håndtere. Gjennom prosjektet ble det derfor klart at de brukerhistoriene vi først hadde laget, ofte ikke passet med hva vi faktisk produserte. På bakgrunn av dette valgte vi heller å bruke de til inspirasjon, framfor systemkrav.

3.2.2 Planlegging

På grunn av den store friheten gruppen hadde i alle deler av prosjektet, ble det svært vanskelig å utarbeide en prosjektplan vi kunne følge. Tidlig i prosjektet ble det derfor klart at vi måtte følge en agil fremgangsmåte, hvor vi satte klare rammer for arbeidet og dokumentering, men hvor innholdet ville kunne endre seg fra dag til dag. På bakgrunn av dette, utarbeidet vi en plan for hvordan vi kom til å arbeide. Denne planen er grovt beskrevet i gruppe kontrakten som ligger i appendix.

Kort oppsummert gikk denne planen ut på:

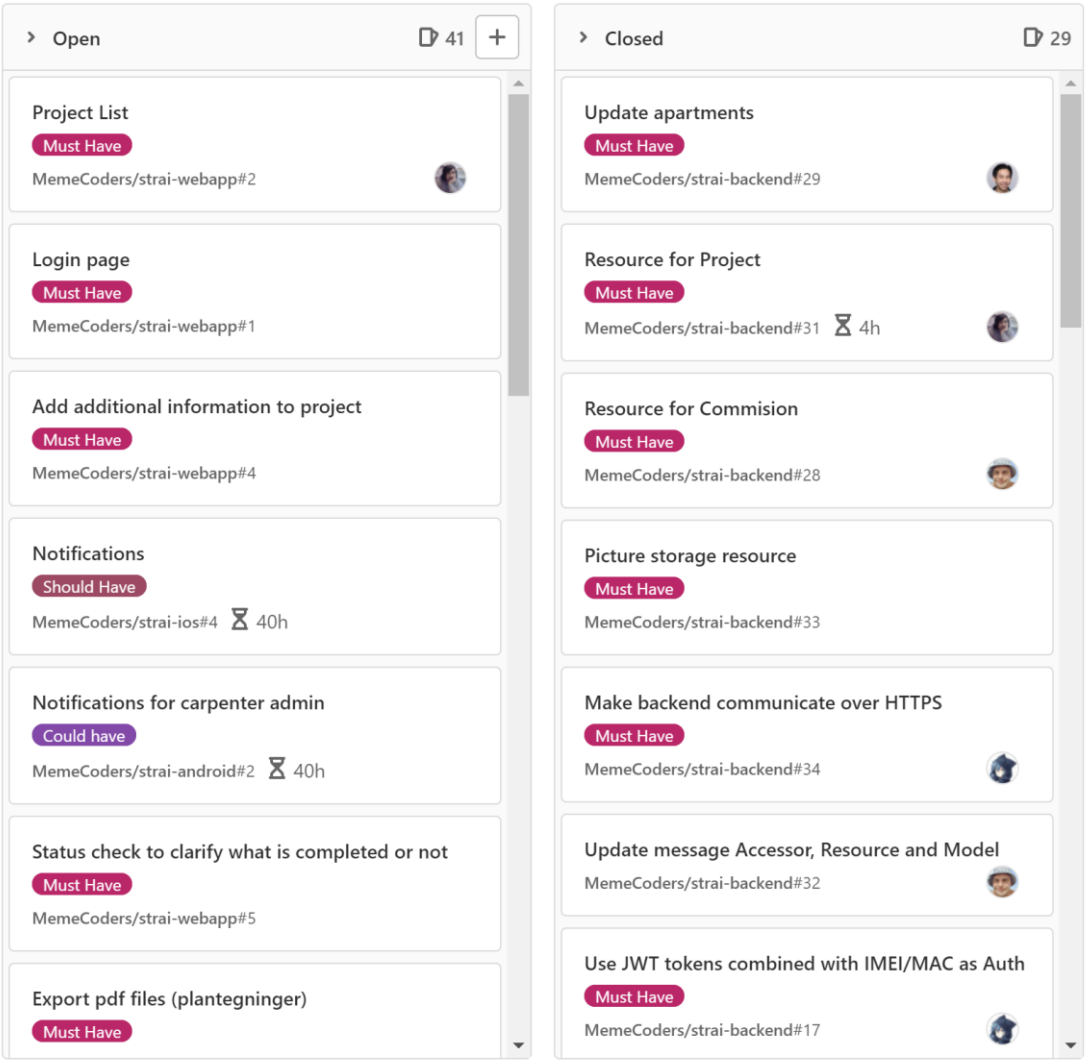
- Gruppen skal være hos Strai mandager, tirsdager og fredager fra kl 9:00 til 16:00
- Alle gruppens medlemmer skal bidra på alle deler av prosjektet.
- Gruppen skal oppføre seg så profesjonelt som mulig, spesielt i samtaler med prosjekteier og andre ansatte hos Strai.

I forhold til tidsestimering, valgte vi ikke å koble timer direkte til backlogens gjenstander. Dette kommer av at vi som gruppe ikke følte vi hadde nok erfaring til å estimere hvor mye tid utviklingen av de spesifikke funksjoner ville ta. Når vi prøvde å estimere tid, estimerte vi ofte altfor mye eller altfor lite tid. Tidsestimatene ble da vanskelig å bruke til planlegging. Samtidig måtte gruppen bruke mer tid på å estimere, enn å faktisk undersøke og utvikle programmet.

Gruppen valgte heller å spesifikt definere en mengde tid vi skulle bruke på utvikling hver uke. Denne tiden tilsvarer de timene hele gruppen bruker til sammen på kontoret, som tilsvarer 84 timer til sammen pr uke.

3.3 Backlog

I utgangspunktet bestod backloggen av rundt 20 punkter sortert etter prioritering. For å lettere håndtere prosjektet valgte vi å koble hvert backlog-objekt til koden den omhandler. På denne måten var det lettere for gruppens medlemmer å direkte finne ut hvor en spesifikk oppgave hadde blitt håndtert. Dette resulterte også i at mengden elementer økte drastisk. For å gjøre dette mer oversiktlig gjorde vi det slik at noen elementer var sub-elementer av større objekter.



Figur 3 - Backlog før overgang til PWA

Figuren over viser backloggen for prosjektet før vi gikk fra bruk av “native applications” til PWA. Alle frontend-objekter har derfor tre forskjellige “versjoner” basert på hvilken plattform de tilhører. Grunnet overgang til Github mistet vi muligheten til å bruke det samme Issue-bordet for backlogen.

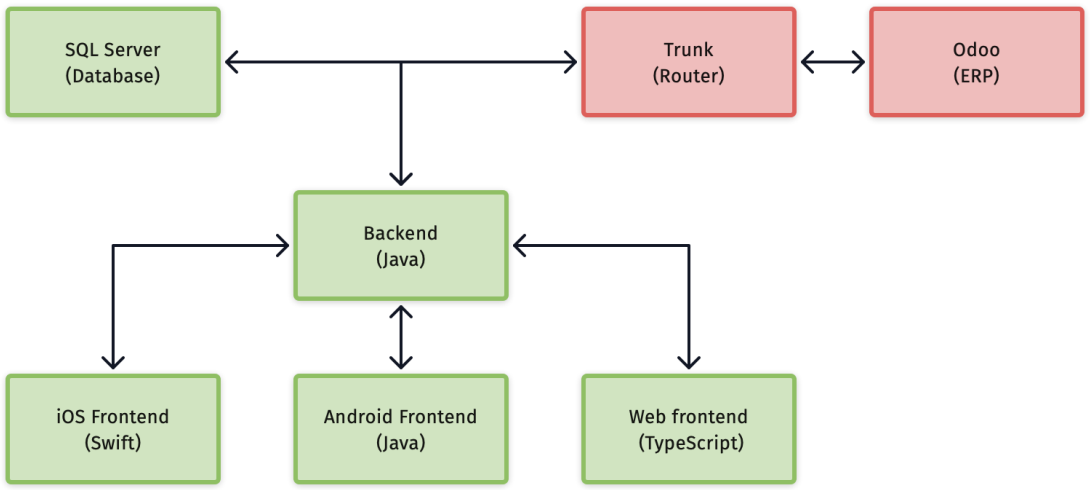
3.4 Gjennomføring av sprints

En stor del av denne oppgaven handlet om å analysere og avgjøre hva produktet skulle inneholde, samt finne den smarteste måten å utvikle det på. Gjennom prosjektet ble derfor både scope, teknologi og arbeidsmåte endret drastisk flere ganger. I dette kapittelet skal vi dokumentere noen av de store hendelsene fra hver sprint.

Som nevnt ble pre-sprint brukt til planlegging og analyse av det eksisterende systemet. Sprint 1-5 ble brukt til utvikling av prosjektet.

3.4.1 Pre sprint

I pre sprinten avgjorde gruppen sammen med arbeidsgiver at det å lene seg på gruppens tidligere erfaringer ville være det mest gunstige. Vi håpet å bruke kunnskapen gruppen allerede hadde rundt Java og Swift for å lage én native applikasjon til hver plattform. Siden mesteparten av funksjonaliteten kom til å ligge i REST api-en, valgt vi også å fokusere bare på den i starten av prosjektet.



Figur 4 - Systemarkitektur Sprint 0-2

Figuren over er en demonstrasjon av den originale system-arkitekturen til prosjektet.

3.4.2 Sprint 1-2

Denne perioden varte fra slutten av januar til slutten av mars. I denne perioden fokuserte gruppen utelukkende på backend utvikling. Gjennom denne perioden var valg av teknologi, databasestruktur og hvordan vi skulle håndtere data hentet fra Strai sine systemer, en stor utfordring.

3.4.2.1 Programmeringsspråk i backend

Til å starte med brukte vi en uke på å teste forskjellige programmeringsspråk. Gruppen var innom Rust, TypeScript/JavaScript og Java. Bakgrunnen for testingen var at vi ønsket å finne den beste løsning for kommunikasjon med backend. Testingen gikk ut på å teste hvorvidt forskjellige funksjoner ble støttet i de forskjellige språkene. Mer om hvordan vi utførte testingen blir reflektert over i 5.3.2. Til slutt bestemte vi oss for å bruke Java i vår backend. Grunnen til dette er at Java er det programmeringsspråket gruppen har mest erfaring med, og at det fylte de kravene vi hadde så langt, i forhold til database-tilkobling.

3.4.2.2 Databasestruktur

I utgangspunktet var ideen at vårt system skulle ta ordre-strukturen til Odoo og legge til kolonnene som omhandlet den ekstra informasjonen vi ønsket å lagre. Ved å gjøre det på denne måten ville vi da kunne ha relasjoner fra ordre til både ansvarlig bruker og

kommentarer. Men i praksis ville dette gi en oss en stor synkroniserings-utfordring siden vi da ville sitte med to kilder for ordre, både fra vår database og fra Odoo sin.

I diskusjon med prosjekteier ble det klart at mange av funksjonene som først var tenkt skulle ligge i vårt system, egnet seg bedre i deres ERP-system. Et eksempel på dette er hvordan appen skal vite hvilke brukere som skulle se hvilke prosjekter. Vi mente at det ikke ga logisk mening at vårt system skulle legge til informasjon som omhandlet ordene, annet en kommentarer og bilder. Dette mente vi ville innebære mer jobb for selger og ansatt. I et slikt tilfelle måtte selger, etter å ha opprettet en ordre i Odoo, også lagt til ansvarlig montør i vår app. Samtidig måtte vårt system ha tatt høyde for potensielle synkroniseringsproblemer.

Etter diskusjon med oppdragsgiver flyttet vi derfor en del funksjoner over til Odoo. Når en ordre blir opprettet i Odoo vil selgeren da sette en ansvarlig kontaktperson fra Strai og et ansvarlig montør-firma. Vi valgte derfor at når en bruker blir opprettet i vårt system, får personen en referanse-ID som er lik IDen til tilhørende firma i Odoo. Dette gjør at backenden vår kan bruke brukerens referanse-ID til å spørre Odoo om alle ordre som tilhører brukerens firma. Disse referanse-IDene gjør det også er mye lettere for oss å sende varsel til brukeren over mail når det blir lagt til nye kommentarer. Den samme logikken er brukt i forhold til ordre. I stedet for at vår database lagrer en liste av ordre, har alle kommentarer og bilder en referanse-ID som tilsvarer ordre-ID i Odoo. Dette gjør at så lenge IDen er riktig fra Odoo, vil systemet vårt hente kommentarer og bilder til riktig ordre.

3.4.2.3 Sjekkliste

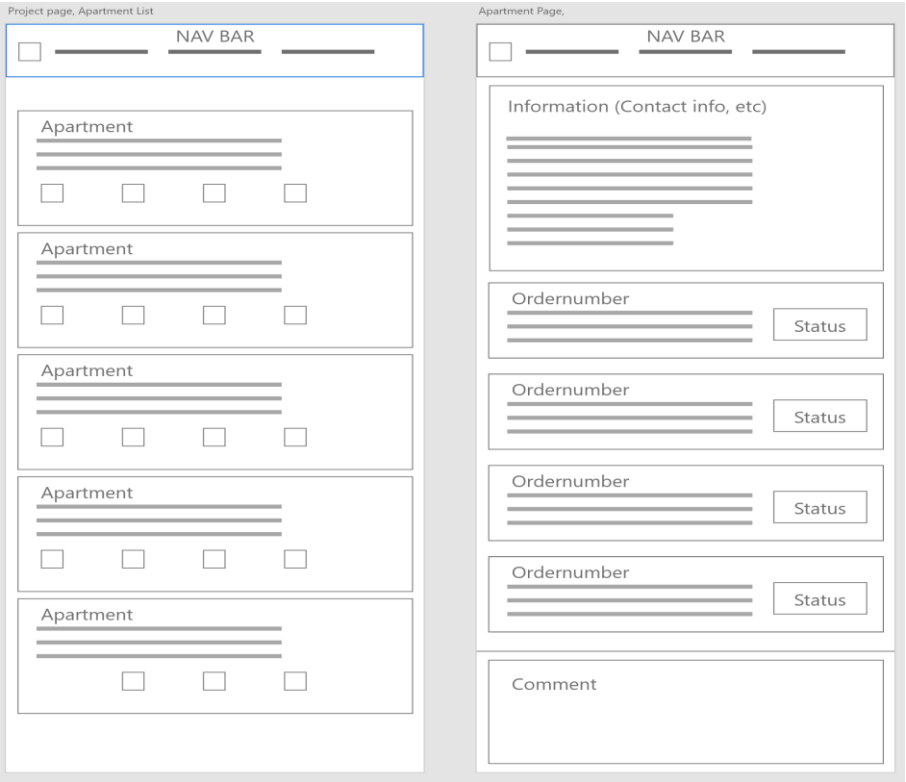
En annen funksjon som ble mye diskutert i denne perioden, var sjekklisten. I dag skal montørene krysse av på en sjekkliste når de har ferdigstilt en leilighet. Sjekklisten er et A4-ark med avkrysnings-punkter hvor montøren skal bekrefte at han har utført arbeid i henhold til det oppgavene krever. Deretter skal han ta bilde av sjekklisten og laste den opp i appen. Dette gjør de fordi Strai ønsker dokumentasjon på at montøren har montert etter henvisningen, og at byggherren eller andre snekkere kan gå inn i leiligheten og få bekreftet at oppgaver er gjort ferdig. I utgangspunktet ønsket Strai at denne funksjonens skulle digitaliseres og at det heller burde bli laget en automatisk sjekkliste som montøren fyller ut i appen. Etter mer undersøkelse ble det klart at denne funksjonen ikke ville fjerne papir-sjekklisten, siden byggherren fremdeles måtte ha tilgang til sjekklisten. Det vil da si at montøren både måtte ha skrevet under på papirversjonen, og ført inn i den digitale sjekklisten. Resultatet av dette ville vært dobbelt arbeid for montørene. For å unngå dette bestemte vi oss for å heller bare legge til rette for at en montør kan laste opp et bilde av sjekklisten, ettersom den digitale versjonen av sjekklista ikke kom til å gi noen ekstra verdi utover den originale.

3.4.3 Sprint 3-5

I sprint 3-5 var fokuset på front-end utvikling og småjusteringer på back-end.

3.4.3.1 Design

I starten av sprint 3 la vi fram et designforslag, dette i form av en low fidelity prototype laget i Adobe XD.



Figur 5 - Et utdrag fra Wireframes versjon 1.

Denne prototypen tok utgangspunkt i den allerede eksisterende applikasjonen. Brukerne var positive til det originale designet, og vi valgte derfor å bruke den som inspirasjon til vår prototype. Bilder av den originale applikasjonen ligger i appendix

Prototypen ble godkjent av oppdragsgiver og ble positivt mottatt av brukerne. Deretter begynte vi for fullt å arbeide på frontend utvikling.

3.4.3.2 Fra Native app til PWA skrevet i React

Gruppen splittet seg opp på de forskjellige plattformene, men det ble etter to uker veldig klart at dette var en stor ulempe. Utfordringer som async kode, CORS og native funksjoner som bildeopplasting eller filbehandling måtte håndteres på helt forskjellige måter avhengig av språk og plattform. Dette gjorde at framgangen en person gjorde på et system, ikke kunne bli brukt på de andre. Gruppen kunne ikke lengre samarbeide på samme måte, og fungerte mer som tre ulike teams i stedet for et. Det ble veldig klart for oss at vi ikke kom til å nå målet, og ville endt med å levere tre så og si uhåndterlige produkter hvis vi fortsatte med denne strukturen.

I en diskusjon med arbeidsgiver la vi fram et par ideer til hvordan vi kunne løse dette problemet:

- Droppe web-applikasjon for å ha flere personer på de mobile plattformene
- Bytte ut alle påbegynte applikasjoner, og gå over til et nytt rammeverk som React Native eller PWA skrevet i React.

Strai anbefalte oss å se på React, og etter et par dager med analyse og planlegging bestemte vi oss for å endre frontend fra tre native applikasjoner til én PWA (Progressive Web App) skrevet i React. Dette gjorde at vi kun trengte å lage én applikasjon som kunne bli brukt på alle plattformer. Endringen gjorde også at gruppen kunne samles og samarbeide på én og samme kodebase. Effektiviteten økte og vi klarte å produsere mye mer på de neste 2 ukene, enn det vi klarte originalt. I tillegg fikk vi flere andre fordeler. PWA har automatisk støtte for offline mode, noe som gjør at vi sparer oss noe tid i utviklingen. I tillegg trenger ikke Strai å forholde seg til tre forskjellige kildekoder, men heller én.

Samtidig som gruppen valgte å ta i bruk React, valgte vi også å bruke React-Bootstrap som vårt design-rammeverk. Gruppen har relativt lite erfaring i design, og ønsket heller å bruke etablerte pakker. Etter å ha teste Material UI og Bootstrap bestemte vi oss for å bruke React-Bootstrap. Som nevnt tidligere har denne pakken ferdiglagde React komponenter, som sparte oss mye tid i designprosessen.

Resten av perioden ble designet bearbeidet, styling ble lagt på, og databasen tweaket til å passe endringene i frontend. Til slutt kommer tiden etter innleveringsfristen til å bli brukt på eventuelle andre oppgaver som må ferdigstilles, før IT-sjefen hos Strai skal ta over og gjøre systemet klart for lansering.

4.0 Produktet

Vi har valgt å legge ved en kort video som går gjennom produktet.

Video-gjennomgang av frontend applikasjon

5.0 Refleksjon

I dette kapittelet skal vi reflektere over de sentrale valgene vi tok, og utfordringene vi møtte på i løpet av prosjektet.

5.1 Utfordringer

I dette avsnittet skal vi gå over og reflektere over noen av utfordringene som dukket opp i løpet av prosjektet.

Prosjekteier hadde gitt oss frie tøyler, i form av hva slags teknologi vi kunne bruke, hvordan appen skulle se ut, osv. Det eneste som var definert var at applikasjonen skulle inneholde grunn-funksjonaliteten som allerede finnes i den originale montør-appen. De frie tøylene gjorde som at prosjektet ble ekstremt lærerikt ettersom vi måtte være kritiske og tenke godt over valgene som presenterte seg for oss.

Vi støtte også på utfordringer da vi bestemte oss for hva slags teknologi vi skulle ta i bruk. Det startet med at vi ville implementere tre native applikasjoner. Grunnen til dette, var at vi allerede hadde kjennskap til Java og Swift, som kunne bli brukt til å utvikle Android og iOS-applikasjoner. Om vi ønsket å utvikle en app som støtter alle plattformer, måtte vi brukt tid på å lære oss et nytt programmeringsspråk. Vi lærte senere at dette var en feilvurdering fra

vår side, ettersom det ble altfor mye arbeid å skulle utvikle tre forskjellige applikasjoner i en så liten gruppe. Dette resulterte i at vi kastet bort rundt to uker med programmerings-tid.

5.1.2 For mye frihet

Arbeidsgiveren hadde i utgangspunktet ikke definert scope, men heller gitt oss prosjektbeskrivelse med en funksjonsliste. Dette gjorde at vi måtte diskutere med arbeidsgiver for å bli enige om valg av teknologi, deadlines, og andre tilnærmingsmåter vi måtte ta hensyn til for å klare å utføre prosjektet.

Det er utrolig mange forskjellige måter å løse oppgaven på. Derfor ble de første ukene brukt på å diskutere og teste flere forskjellige programmeringsspråk. Det var mye fram og tilbake, angående hvilke teknologier og språk vi mente egnet seg best til prosjektet. Dette er noe vi utdyper, og reflekterer mer over i 5.3.2.

Vi fikk også frihet til å velge hvordan vi skulle jobbe. Dermed var det viktig at vi satte deadlines for oss selv i form av sprints. Dette ga oss et unikt innblikk i hvordan det var å planlegge og utføre et prosjekt. Vi måtte sette opp ukentlige møter med Strai, der deltakerne var forskjellige etter behov. Møtet kunne for eksempel være med salgsavdelingen den ene uka, og med montører uken etter. Det var også tilfeller der begge partene var tilstede. På disse møtene fikk vi presentert hva vi hadde gjort så langt og vi fikk konstruktive tilbakemeldinger. Vi fikk også innspill på hva slags funksjoner de ville ha med i applikasjonen, og funksjonslisten kunne endres etter behov.

5.2 Sentrale valg

I dette avsnittet skal vi reflektere over de sentrale valgene vi tok, og hvilke utfall de ga.

5.2.1 Scrum

Som nevnt valgte gruppen å følge Scrum gjennom prosjektet. Selv om vi ikke har fulgt absolutt alle deler eller eventer som Scrum består av, har vi fulgt det vi mener er det viktigste.

Sprints og backlog ble et viktig verktøy for å kunne videreutvikle produkt og arbeidsprosess. I etterkant av hver sprint brukte gruppen mye tid på gjennomgang av kode, slik at gruppens kollektive forståelse ble bedre.

5.2.2 Kvalitetssikring

Når det kommer til kvalitetssikring føler vi at vi har gjort et godt arbeid. Siden Strai skal ta over prosjektet var det viktig for oss å skrive logisk og strukturert kode. Samtidig fulgte vi etablerte regler og brukte rammeverk for å gjøre det lettere for gruppen å produsere kode med høy kvalitet.

Vi prøvde også å ikke bruke en større mengde unødvendige pakker for å løse oppgaver, dette gjør vi igjen for å hindre systemet fra å bli “bloated” og for at Strai ikke skal måtte forholde seg til mye forskjellige dokumentasjon i senere tid.

5.3 Endringer i prosjektet

Gjennom prosjektet var det mye endringer i både scope og teknologier. Det kom fram nye ideer, og nye optimaliseringer som gjorde at funksjonene til applikasjonen endret seg. I dette kapittelet skal vi reflektere over endringene som ble gjort i både scope og teknologi.

5.3.1 Endringer av scope

Under hele prosjektperioden var scope under store endringer. De første ideene om hvordan systemet og arbeidsprosessen skulle se ut, var ofte ineffektive og trengte å bli bearbeidet. Samtidig gjennom prosessen kom det fram mange nye ideer som prosjekteier aldri hadde hørt om før.

Gjennom diskusjoner og undersøkelse ble scope endret flere ganger. Nye funksjoner viste seg å være lette å implementere, eller helt umulige. Gruppen måtte bruke mye tid på å undersøke mulige løsninger og samtidig argumentere disse til Strai. I etterkant av prosjektet har vi fått tilbakemelding på at dette var noe Strai følte var bra gjennomført av oss, og at det har bidratt til et bedre produkt.

Et godt eksempel på en prosess som ble totalt endret, var sjekklisten. Som nevnt i sprint 1-2 endte vi med å endre prosessen som Strai foreslo. Det er mange andre måter man kunne ha løst denne situasjonen. Eksempelvis, hatt en mulighet for å printe ut den digitale sjekklisten. Vi mener at dette fremdeles hadde resultert i unødvendig arbeid på montøren, og i forhold til systemet så ville det fremdeles ikke gi noen annen verdi. Basert på erfaringen vi opplevde i henhold til denne situasjonen, ser vi at det er viktig å fokusere ekstra på å ikke overkomplisere løsningen hvis det ikke skaper noe ekstra verdi.

Vi vil også nevne at Strai var ekstremt villig til å tilpasse Odoo etter våre meninger, noe som har vært en ekstremt gøy og lærerik situasjon for oss.

5.3.2 Endringer av teknologi

Det skjedde flere endringer i teknologi i løpet av prosjektet. Det var blant annet fordi gruppen var usikre på hvilke språk som var best egnet for prosjektet. Som nevnt tidligere, testet vi forskjellige programmeringsspråk for å komme fram til en beslutning. Måten vi testet dette på var å sette opp en liste med funksjoner og krav, og deretter se igjennom dokumentasjonen til de forskjellige språkene, og se hvilke språk som hadde best støtte for de funksjonene vi trengte. Deretter prøvde vi ut disse funksjonene i små omfang, og sjekket om de ga oss et forventet resultat. Et typisk eksempel er hvordan de forskjellige språkene tar for seg HTTP-kall. Dette var drastisk enklere i noen språk kontra andre. Et annet eksempel er hvorvidt språket hadde støtte for integrasjon av databasen vi brukte, dette var for eksempel mye enklere i Java, enn i Rust.

Det ble vurdert å skrive backenden i Rust eller JavaScript. Vi måtte ta hensyn til programmerings-nivået til gruppen og det ville tatt mye tid å lære et helt nytt språk, derfor bestemte vi oss at det var lurt å heller bruke et språk gruppemedlemmene hadde erfaring med.

En av de store endringene vi gjorde under prosjektet var å bytte fra å lage én frontend-applikasjon for hver plattform (iOS, Android, web), til å lage én PWA som fungerer på alle

plattformer. Bakgrunnen for endringen var at det var forskjellige måter man løste problemer på avhengig av plattform. Dette resulterte i at når én person løste et problem, kunne ikke nødvendigvis den samme løsningen bli brukt av noen andre. På denne måten kunne vi ikke lære av hverandres erfaringer.

Vi måtte også ta hensyn til videreutvikling av applikasjonen, ettersom IT-sjefen hos Strai skal ta over utvikling av applikasjonen når vi som gruppe er ferdige. PWA gir da den fordelen at han ikke trenger forholde seg til tre forskjellige språk og kodebaser, men heller kun 1.

Fra starten av prosjektet var det klart at gruppen måtte tilnærme seg ny kunnskap for å kunne produsere det produktet Strai ønsket. Det var derfor mye usikkerhet rundt valg av teknologi, struktur og design. Gjennom prosessen ble det klart at selv om man gjør et valg basert på informasjonen man har, kan ting ofte endre seg.

Ofte møtte vi på problemer, og mottok ønsker om endring selv om vi allerede var kommet langt i utviklingsfasen. Dette gjorde at vi måtte gå tilbake og gjennomføre en ny analyse. Vi har hatt daglig kontakt, oftest ansikt til ansikt med arbeidsgiver og prosjekteier, for å diskutere funksjonalitet og generelle tema. Disse arbeidsmetodene, samt flere andre punkter, finner man som grunnprinsipper i det agile manifesto. På denne måten kan vi si at vi har brukt en agil metode for å gjennomføre prosjektet (Agile manifesto, 2001)

Gruppen var klar for å ta tak i nye konsepter og teknologier, og lære seg dem så godt man klarte på egen hånd. Så selv om ikke alt vi produserte kom med i det ferdig produktet, tror vi at vi har fått erfaring som vil være ekstremt viktig i arbeidslivet.

5.3.3 Testing

I løpet av prosjektet har det gjentatte ganger vært et tema hvordan vi skal utføre testing. Alle deler av prosjektet har blitt manuelt testet i løpet av utviklingen, men automatisert testing har ikke blitt implementert på noe vis. Grunnen til mangel på automatiserte tester har vært at det har vært problematisk å faktisk utføre slike tester uten å erstatte andre systemer vi snakker til med såkalte “mock” objekter.

5.3.3.1 Testing av API

Testing av backend har foregått manuelt via enten curl eller Postman-programmene. Her har vi da selv gjort kall til de forskjellige endepunktene og verifisert at resultatene er korrekte eller at en oppdatering i databasen har skjedd i henhold til forventet handling.

5.3.3.2 Testing av frontend

Appen har blitt manuelt testet og har lite funksjonalitet utenom å kalle endepunkt fra vår API, samt vise resultater. All annen funksjonalitet som å vise bokser hvis knapper blir trykket på, har vært gjort via React Bootstrap-rammeverket. Dermed har vi ikke sett behovet for å skrive automatiserte UI-tester.

5.3.3.3 Refleksjon

I ettertid ser vi at det hadde vært gunstig og hatt automatiserte tester av SQL kode. Dersom vi må endre databasestruktur nå, må vi passe på at kode i backend også endres for å

reflektere databasen. Denne koblingen har ført til noe problemer når gruppemedlemmene har hatt forskjellige versjoner av databasestrukturen på sin lokale database.

5.4 Wireframes

Som nevnt tidligere brukte vi wireframes for å vise fram prototypen. Dette gjorde vi ettersom vi tok utgangspunkt i designet til den allerede eksisterende applikasjonen. I senere tid ser vi muligheten for at vi mistet noe av den potensielle innovasjonen ved å gjøre det slik. En wireframe kan ofte føles komplett, noe som kan forhindre test-brukeren i å gi god tilbakemelding. Sketcher hadde potensielt egnet seg bedre til første prototype.

6.0 Sitat fra oppdragsgiver

6.1 Prosjektbeskrivelse fra oppdragsgiver

Studentene skal lage en applikasjon for kjøkkenmontører som skal kunne brukes på PC, mobil og nettbrett. Appen skal digitalisere og effektivisere hverdagen for kjøkkenmontører som tar oppdrag fra Strai Kjøkken AS. Vi har en mobilapp i dag hvor vi ivaretar prosjekt kunder i butikken vår i Kristiansand som benytter et bestemt montasje-selskap, men har behov for å utvide til alle våre butikker og alle montører. Appen skal integreres med et ERP-system og hente ut informasjon til montørene derfra, samt noe informasjon skal også tilbake til ERP-systemet.

Kravspesifikasjonen til appen inneholder funksjoner som:

- Innlogging
- Liste opp prosjekter med tilhørende leiligheter og ordrer
- Lese sjekklister
- Ta bilder
- Ferdigmelding
- Kommentarer med avvik med varsel til våre selgere

Studentene har fått frie rammer til å velge programmeringsspråk/rammeverk for å løse oppgaven.

6.2 Gjennomføring

Studentene har lånt to kontorer på Strai Kjøkken AS som er blitt brukt flittig. De samarbeider godt og har definerte roller i gruppen. Vi har valgt å holde kontorene åpne under korona, men i tilfeller hvor studentene har vært nødt til å jobbe eksternt har de selv funnet og brukt teknologi for samarbeid seg imellom.

Oppfølgingen av bachelorgruppen skjer på et fast oppsatt tidspunkt en gang pr. uke, og et større møte med flere deltakere en gang pr. mnd. Samarbeidet oppleves som bra.

Studentene valgte innledningsvis å benytte native teknologi på Android og iOS, Java for backend og React på web. Halvveis i prosjektet fant studentene ut at dette ble for mye

arbeid på for få personer, og valgte å begynne på nytt med kun en React app, og gjøre denne om til en PWA for å kunne brukes offline / installeres på mobile enheter.

Arbeidsgiver står midt i en implementeringsprosess med nytt ERP-system som fått en forsinket oppstart. Dette har medført at studentene ikke har fått mulighet til å koble seg til dette systemet tidsnok, og dermed ikke fått testet applikasjonen sin ut mot sluttbrukere. Studentene har likevel gjort ferdig applikasjonen utenom integrasjonen og den tilfredsstiller de fleste av våre kravspesifikasjoner.

Vi som arbeidsgiver er godt fornøyd med innsatsen til studentene.

— Espen Cederberg It-Sjef Strai Kjøkken

7.0 Egenevaluering

7.1 Martin Vottestad Stenberg

Gjennom dette prosjektet har jeg fungert som en gruppe leder. Mye av ansvaret for å holde styr på hva vi skal gjøre og hvordan vi skal gjøre ting har falt på meg. Jeg føler at dette er en rolle jeg har gjennomført godt og har bidratt godt til at vi har kommet fram til slutt produktet. Men selv om jeg er fornøyd med den delen av arbeidet mitt ser jeg at jeg potensielt ikke har skrevet like mye kode som jeg selv skulle ønske. Mens restens av gruppens medlemmer fokuserte på å fordype seg i deler av systemet tok jeg en mer overordnet utvikler rolle, og prøvde mer å diskutere og bearbeide designet og arkitekturen til systemet i stedet for direkte kode. Annet en dette har jeg fått ekstremt positiv tilbakemelding for min business forståelse og føler at den har bidratt ekstremt godt i kommunikasjonen med Strai samt forståelsen av deres arbeidsprosess.

7.2 Marcus Tuan Tran

Gjennom dette prosjektet har jeg hatt rollen som utvikler. I starten hadde jeg ansvar for database design og har videre jobbet på back-end design og innloggingssiden i appen. Bachelorprosjektet har gitt meg mange gode erfaringer ettersom jeg satte meg inn i nye rammeverk og lærte et nytt programmeringsspråk. Jeg synes det var utrolig spennende å få jobbe i et prosjekt som gir relevant erfaring til arbeidslivet. Jeg har også fått et innblikk av business aspektet med en utvikler ettersom det ikke bare var ren koding, men å prøve å forstå hva de på strai ville at vi skulle lage.

7.3 Runar Schjønning Grønås

I løpet av prosjektet har jeg vært versjonskontroll administrator og hoved code-reviewer for hele gruppen. Det har vært min oppgave å opprettholde kode standard, som jeg føler jeg har fått til ganske bra.

I starten av prosjektet var jeg også hovedansvarlig for utvikling av iOS appen i Swift. Swift er et språk jeg har lang erfaring med samt stor interesse for. Selv om iOS appen senere ble lagt til side, føler jeg selv at jeg kom veldig langt på tiden vi hadde.

Jeg har vært teknisk bistand for hele gruppen, introdusert forskjellige løsninger til infrastruktur av systemet og hjulpet dem med å få satt opp forskjellige verktøy vi har brukt i løpet av prosjektet.

7.4 Sindre Edvard Sæther

I dette prosjektet har jeg hatt rolle som utvikler. Jeg hadde i starten hovedansvar for Android-applikasjonen, før vi gikk over til PWA. Jeg har også hatt en sentral rolle i utviklingen av backend. I løpet av prosjektet har jeg tatt til meg mye ny kunnskap utover det jeg allerede hadde innen Java. Jeg har også lært meg programmeringsspråket TypeScript, og rammeverket React, samt utviklet større moduler i PWAen vår. I den teoretiske delen av prosjektet har jeg vært hovedansvarlig for språk, grammatikk og rettskrivning.

I løpet av prosjektet har jeg fått ekstremt god erfaring innen systemutvikling, og hvordan det er å jobbe i en gruppe i et prosjekt av denne størrelsen. Det har vært mye usikkerhet, men vi har sammen klart å håndtere alt av utfordringer på en god måte. Det har vært veldig gøy å være en del av et team som leverer et produkt av denne størrelsen, hvor produktet faktisk skal brukes i den virkelige verden.

8.0 Kilder

Agile Manifesto. (2001). Manifestet for smidig programvareutvikling. Hentet fra https://agilemanifesto.org/iso/no/manifesto.html

Facebook. (2021). React – A JavaScript library for building user interfaces. Hentet fra https://reactjs.org

Ken Schwaber. (2014). *Agile Project Management with Scrum*. u.s: Microsoft Press.

Microsoft. (2021). TypeScript: Typed JavaScript at Any Scale. Hentet fra https://www.typescriptlang.org

Mike Cohn. (2009). *Succeeding with Agile: Software Development Using Scrum*. u.s: Addison-Wesley Professional.

OpenJS. (u.å). Eslint. Hentet fra https://eslint.org/

OpenJS. (u.å.). Rules. Hentet fra https://eslint.org/docs/rules/

Oracle Corporation. (2021, 29. April). Jersey – RESTful Web Services in Java. Hentet fra https://eclipse-ee4j.github.io/jersey.github.io/

React Bootstrap. (2021). React Bootstrap. Hentet fra https://react-bootstrap.github.io/

React Icons. (2021) React Icons. Hentet fra https://react-icons.github.io/react-icons/

Scrum. (2019). What is Sprint Planning?. Hentet fra https://www.scrum.org/resources/what-is-sprint-planning

Skaug, I. (2013, 12. April) Hva er Kanban?. Hentet fra https://www.bouvet.no/bouvet-deler/utbrudd/hva-er-kanban

Sutherland J., Schwaber K. (2020) The 2020 Scrum Guide™. Hentet fra https://scrumguides.org/

Tran, T. H. (2019, 27. Desember) What does mobile-first design mean for digital designers?. Hentet fra https://www.invisionapp.com/inside-design/mobile-first-design/

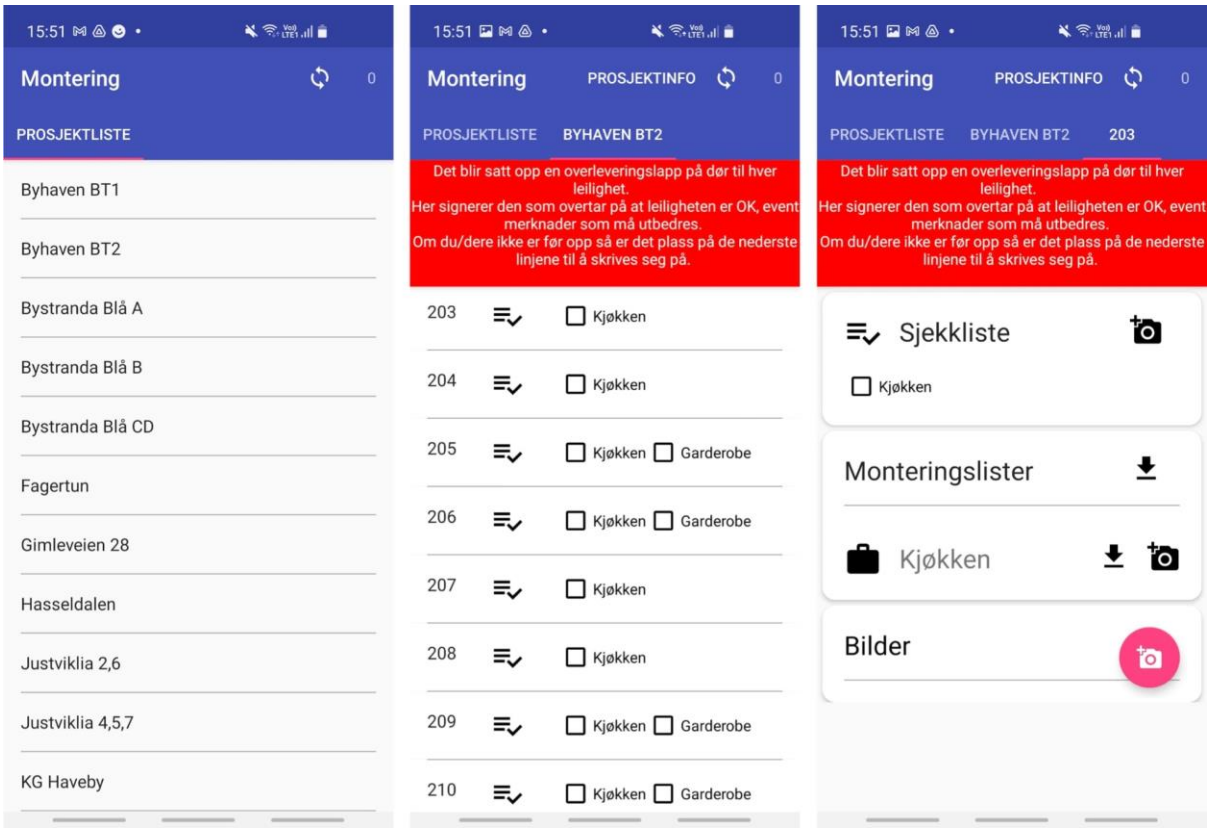
Universitetet i Agder. (2021). Bacheloroppgave i informasjonssystemer. Hentet fra https://www.uia.no/studieplaner/topic/IS-304-1

9.0 Figurliste

Figur 1 - Systemarkitektur 10
Figur 2 - Databasestruktur 11
Figur 3 - Backlog før overgang til PWA 14
Figur 4 - Systemarkitektur Sprint 0-2 15
Figur 5 - Et utdrag fra Wireframes versjon 1 17

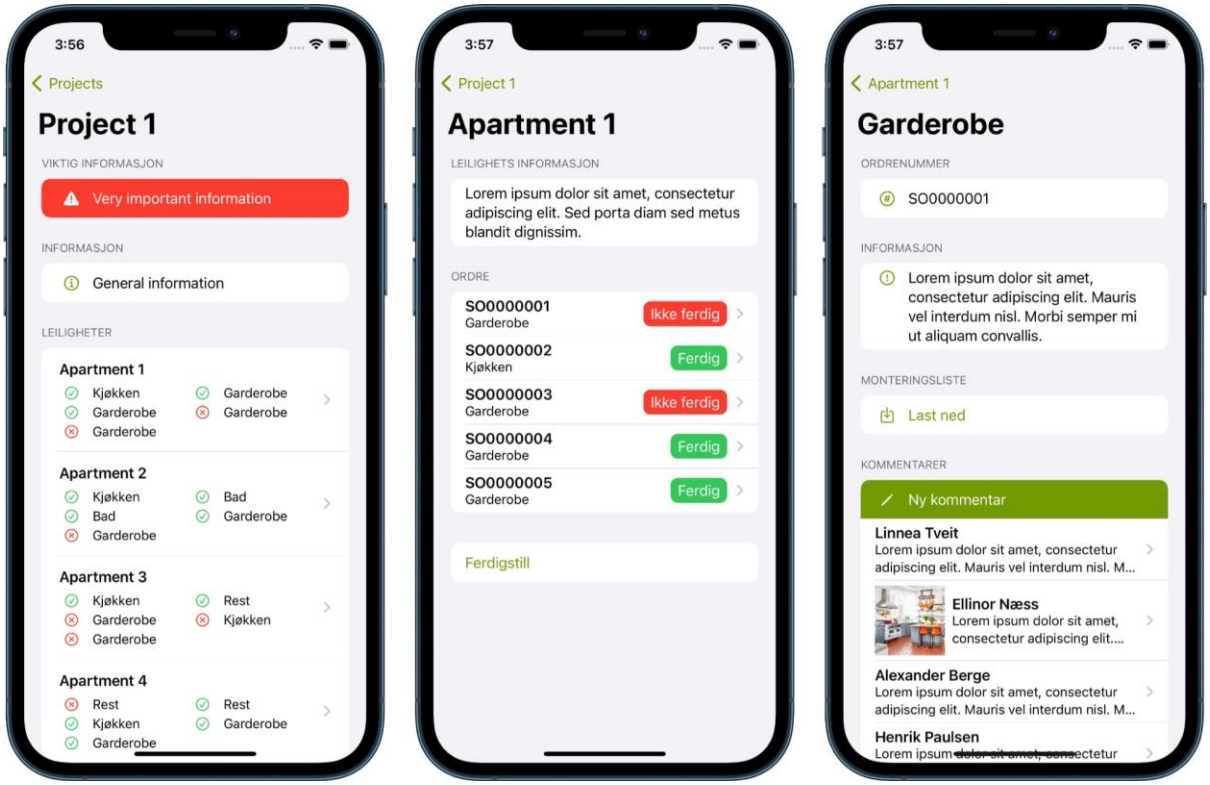
10.0 Appendix

10.1 Bilder av original montør-app



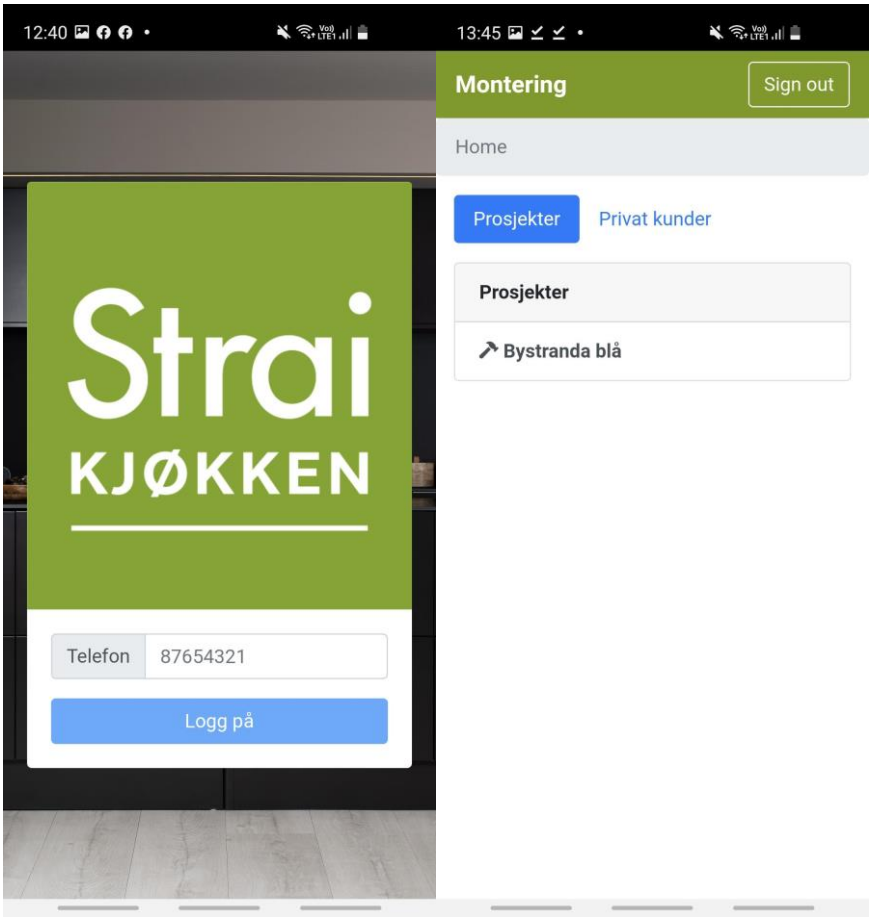
Prosjektliste, ordreliste og ordre side i original app

10.2 Bilder av statisk iOS app (Før PWA)



Prosjektliste, leilighets side og ordreside i iOS appen.

10.3 Bilder av sluttproduktet.



Login page, og hjem skjerm

10.4 Kodeeksempler

strai-pwa > src > Pages > TS ProjectDetailPage.tsx > ProjectDetailPage

```tsx
import { ReactElement } from 'react'
import { Alert, Container } from 'react-bootstrap'
import { BsExclamationTriangleFill, BsInfoCircleFill } from 'react-icons/bs'
import { useParams } from 'react-router-dom'
import { ApartmentList } from '../Lists/ApartmentList'
import { fetchApartmentsForProject, fetchProject } from '../Store'
import { RouteParams } from '../Types'
import { ErrorPage } from './ErrorPage'

export const ProjectDetailPage = (): ReactElement => {
    const { projectId } = useParams<RouteParams>()

    const project = fetchProject(parseInt(projectId))

    if (project) {
        return (
            <Container className='mb-3'>
                <h1>{ project.name }</h1>

                { project.importantInformation &&
                    <Alert variant='danger'>
                        <BsExclamationTriangleFill className='mb-1' /> { project.importantInformation }
                    </Alert>
                }

                { project.information &&
                    <Alert variant='info'>
                        <BsInfoCircleFill className='mb-1' /> { project.information }
                    </Alert>
                }

                <ApartmentList apartments={ fetchApartmentsForProject(project.id) } />
            </Container>
        )
    } else {
        return <ErrorPage message='Prosjektet du forsøkte å åpne finnes ikke.' />
    }
}

```

React komponent skrevet i etter Eslint standard. Project Detail Page

strai-pwa > .eslintrc.yml

```yaml
env:
  browser: true
  es2021: true
extends:
  - 'eslint:recommended'
  - 'plugin:react/recommended'
  - 'plugin:@typescript-eslint/recommended'
parser: '@typescript-eslint/parser'
parserOptions:
  ecmaFeatures:
    jsx: true
  ecmaVersion: 12
  sourceType: module
plugins:
  - react
  - '@typescript-eslint'
settings:
  react:
    version: detect
rules:
  indent: ["error", 4, { "SwitchCase": 1 }]
  quotes: [error, single]
  semi: [error, never]
  curly: [error, multi-line]
  no-extra-parens: error
  no-template-curly-in-string: error
  no-await-in-loop: error
  no-console: warn
  no-unreachable-loop: error
  no-unsafe-optional-chaining: error
  require-atomic-updates: error
  default-case: error
  default-case-last: error
  dot-location: [error, property]
  dot-notation: error
  eqeqeq: error
  no-multi-spaces: error
  no-return-await: error
  no-self-compare: warn
  no-useless-concat: error
  no-useless-return: warn
  brace-style: error
  camelcase: error
  comma-dangle: error
  eol-last: error
  implicit-arrow-linebreak: error
```

Konfigurasjonsfil for Eslint

10.5 Gruppekontrakt

Gruppenavnet ble endret tilbake til vårt originale navn BowlerBoys

Gruppe-kontrakt

Gruppe navn: Mama Rusi

Medlemmer og roller:

Hvert gruppemedlem skal bidra på alle deler av prosjektet, selv om det ikke dirkete er deres ansvarsområde.

- Martin Stenberg
  - Gruppeleder
  - Delegering av arbeid.
  - Ansvarlig for å arrangere og gjennomføre de daglige og ukentlige møtene
  - Ansvarlig for kommunikasjon med faglærer og evt andre kontaktpersoner
- Marcus Tuan Tran
  - Delansvar for utvikling (Android), forskning og rapport
  - Ansvar for innkjøp av nødvendig resurser (Kaffe, godteri etc)
- Sindre Sæther
  - Delansvar for utvikling (Android), forskning og rapport
  - Ansvar for å løse problemer innad i gruppen.
  - Gruppens ansikt ut til media
- Runar Grønås
  - Git-Master
  - Hovedansvar for programmerings-arkitektur
  - Hovedansvarlig for IOS utvikling

Hvert gruppemedlem har også ansvar for å holde seg oppdatert og bidra på studierelaterte oppgaver og forelesninger.

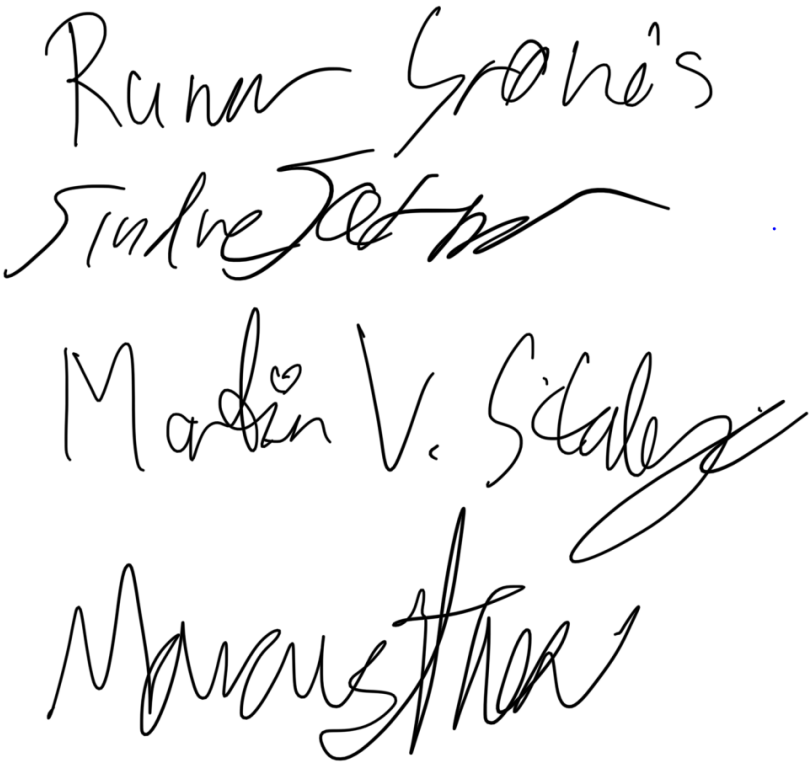
Mål og forventninger

Gruppe har som mål om å produsere en prototype av prosjektet, det trenger ikke inneholde design- eller implementasjonsplaner, men må inneholde funksjonaliteten. Karaktermessig skal gruppen oppnå høy måloppnåelse med minimumskarakter C. Mål som direkte omhandler prosjektet kan endre seg basert på produkteiers mening, samt at gruppen vil gjøre vurderinger gjennom hele prosjektet om målene fremdeles er realistiske.

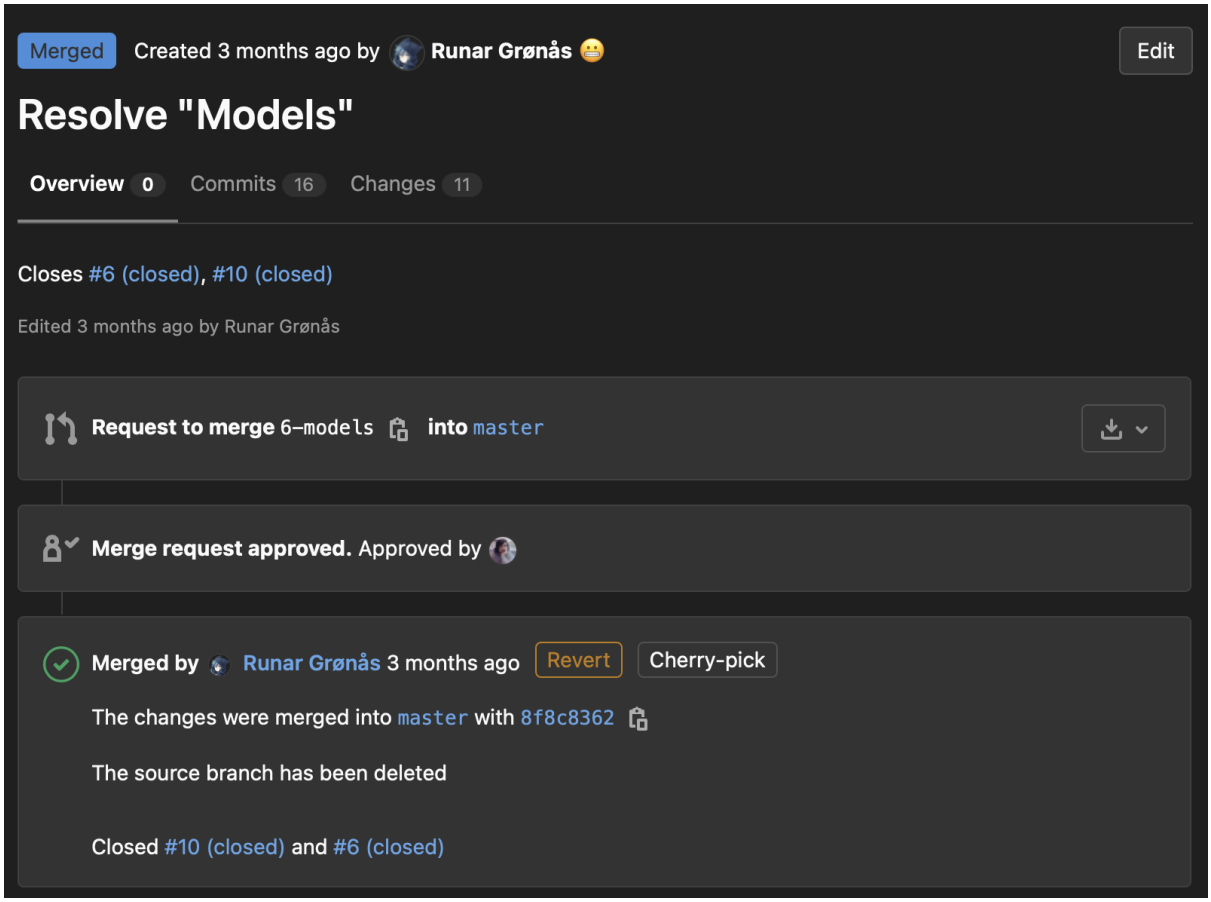
Regler:

1. Alle medlemmer skal være til stede på kontoret til de avtalte dagene.
2. Respekter andre gruppemedlemmers meninger. La andre medlemmer få bidra/tenke på løsninger, slik at ansvaret ikke blir lagt på én enkeltperson og at gruppen føler et eierskap til prosjektet.
3. Hvert medlem skal fokusere på arbeidet i arbeidstiden, så lenge gruppen ikke har en felles pause.

4. Hvis du ikke kan møte på kontoret eller annet avtalt møtested, må dette bli gitt beskjed om til resten av gruppen så fort som mulig.
5. Hvis du ikke kan møte, er det dit ansvar at resten av gruppen har tilgang til nødvendige resursers som du har, for å fortsette deres arbeid.
6. På grunn av størrelsen på prosjektet forventer vi at gruppen også bruker tid hjemme og på fritiden til å arbeide med prosjektet. (Forskning, rapporten, utvikling etc.)
7. Gruppen skal følge de avgjørelsene som blir gjort i gruppen.
8. Hvis det oppstår en konflikt, skal gruppen ta dette opp på en sivilisert og ordentlig måte.
9. Hvis en konflikt ikke blir løst innad i gruppen, er det gruppeleders ansvar å ta kontakt med veileder for råd og evt assistanse.
10. Gruppen er et lag, og skal samarbeide. Gruppemedlemmer skal motivere og hjelpe hverandre når det trengs.
11. Hvis et gruppemedlem bidrar merkbart mindre enn andre skal dette bli tatt opp. slik at man kan løse problemet eller delegere nye oppgaver til medlemmet, for å oppnå at han kan bidra på en annen måte.
12. Hvis gruppemedlemmet ikke forbedrer seg over en lengere periode vil kontaktlærer bli kontaktet og gruppen kan avgjøre om gruppemedlemmet skal bli fjernet fra gruppen.
13. Respekter kontoret og utstyret vi får låne av Strai. Vask pulter, fjern kopper, og behandle utstyret pent.



10.6 Kobling mellom merge request og backlog



En merge request som implementerte to funksjonaliteter fra backlog