

OSLOMET UNIVERSITET

NABONETT

av

Mohamud Abdi Muhumed

Duale Mohamed

Mehbob Singh

Berket Tekeste

Bachelorprosjekt av

Gruppe 46

Faculty of Technology, Art and Design (TKD)

Institutt for Informasjonsteknologi

**Institutt for Informasjonsteknologi**

Postadresse: Postboks 4 St. Olavs plass, 0130

Oslo

Besøksadresse: Holbergs plass, Oslo

Telefon: 22 45 32 00

PROSJEKT NR. 46 / 0046

TILGJENGELIGHET

Nabonett - Boligstyring

Boligstyring gjort enklere.	DATO 2020-2021
	ANTALL SIDER / BILAG 112 sider
PROSJEKTDELTAKERE Duale Mohamed Mohamed / S333972 Mohamud Abdi Muhumed / S333774 Mehboob Singh / S333722 Bereket Tekeste / S331401	INTERN VEILEDER Raju Shresta

OPPDRAKGIVER Styret.com	KONTAKTPERSON Kristoffer G. Skappel Daglig Leder
-----------------------------------	---

SAMMENDRAG

3 STIKKORD
Webapplikasjon, Boligstyring, Boligselskap

Innholdsfortegnelse

Tittelside: Nabonett - Boligstyring	2
Sammendrag	8
Forord	9
1. Introduksjon	10
1.1 Bakgrunn	10
1.2 Beskrivelse av prosjektet	10
1.3 Problemstilling	11
1.3.1 Oversikt over økonomi	11
1.3.2 Dokumentasjon om egen bolig	11
1.3.3 Stemmegivning og meningsmåling	12
1.3.4 Kontaktinformasjon og beboer	12
1.4 Gruppepresentasjon	13
1.5 Introduksjon til Styret.com	14
1.6 Rådgiver OsloMet	15
2. Prosessdokumentasjon	16
2.1 Planleggingsprosess	16
2.2 Arbeidsmåte og prosessmodell	18
2.3 Kravspesifikasjoner og utviklingsprosess	20
2.4 Risikoanalyse	23
2.5 Verktøy og teknologier	25
2.5.1 Front-end teknologi	25
2.5.3 Back-end teknologi	27
2.6 Design og prototype	32
2.4.1 Første iterasjon av design og prototype	32
2.4.2 Kunde undersøkelse	33
2.4.3 Database og Backend design	34
2.4.3 Andre iterasjon av prototype og design	36
2.4.4 Presentere dette til arbeidsgiver	37

2.7 System spesifikasjoner	38
2.7.1 Flytskjema (Flow chart)	38
2.7.2 Use-case diagram	40
2.7.3 Entity Relasjon Diagram	41
3. Produktdokumentasjon	42
3.1 Introduksjon til Nabonett - Webapplikasjon	42
3.3 Grafisk grensesnitt og funksjoner	43
3.3.1 Logg-inn side	43
3.3.2 Hjemmeside	46
3.3.3 Min bolig	47
3.3.4 Skjema	52
3.3.5 Min side	53
3.3.6 Design og heuristikk	53
4. Testing Dokumentasjon	55
4.1 Introduksjon	55
4.1.1 Testing Plan	55
4.2 Enhetstesting (Unit Testing)	57
4.2.1 Test objekter	57
4.2.2 Enhetstesting verktøy	62
4.2.3 Konklusjon og refleksjoner til enhetstesting	64
4.3 Integrajonstesting(Integration Testing)	65
4.3.1 Test objekter	65
4.3.2 Integrasjonstesting verktøy (Integration testing)	66
4.3.3 Manuell integrasjonstesting av Postman	68
4.3.3 Konklusjon og refleksjoner til integrasjonstesting	70
4.4 Akseptansetesting	71
4.4.1 Test objekter	71
4.4.2 Resultat av akseptansetest	71
4.4.3 Konklusjon og møte med klient	75

4.5 Utplassering på Azure	75
5. Teknisk dokumentasjon	76
5.1 Front-end struktur	76
5.1.1 Kilde mappe (src)	77
3.5.1 Klient (Front-end) Validering	80
5.2 Back-end struktur	82
5.2.7 Tjener (Back-end) Validering	92
6. Diskusjon og konklusjon	93
6.1 Diskusjon	93
6.2 Konklusjon	94
6.3 Tilbakemelding fra arbeidsgiver	95
7. Forslag til videre utvikling	96
Ordliste:	96
Bruksanvisning	99
Kilder:	113

Sammendrag

Dette prosjektet er blitt gjort i under henvisningen til OsloMet storbyuniversitet og institutt for informatikk. Denne bacheloroppgaven er skrevet under semester perioden høst 2020 og vår 2021. Ferdig produkt består av webapplikasjonen Nabonett.

Arbeidet gjort under dette prosjekt løpet har bestått av mange deler, men er fordelt i rapportens åtte kapitler. Prosjektet har hatt som mål å utvikle en løsning på et problemstilling stilt av arbeidsgiver Styret.com. Problemstillingen er: Mangel på enkel styring og oversikt over bolig for eiere. Selve problemstillingen besto av flere problemer i seg selv.

Ved hjelp av tidlig prototyping av front-end og back-end arkitektur har produktet gått gjennom ulike faser av produksjon. Disse fasene startet med enkle wireframes og bilder, og utviklet seg senere til fullspekkede løsninger på både front-end og back-end. Arbeidsmetode brukt er fossefallsmodellen. Testing har også blitt gjennomført og har forsikret både arbeidsgiver og gruppen om at produktet er trygg i bruk. Sluttresultatet av en webapplikasjon med funksjoner som beober komponent, stemmegivnings komponent og annet funksjonalitet diskutert herunder og videre rapporten.

Forord

Struktur av rapport følger en kapittelvis rekkefølge og består av syv deler. Disse delene og deres innhold er beskrevet i større detalj under hvert kapittel og innholdslisten. Tradisjonell innhold av forord er skrevet i [Kapittel.1 Introduksjon](#).

[Kapittel. 2 Prosessdokumentasjon](#). Innholdet består av informasjon omkring viktige prosesser av prosjektet, som kravspesifikasjoner, planleggingsprosess, design og prototype og mer.

[Kapittel. 3 Produktdokumentasjon](#). Dette kapittelet beskriver samsvaret mellom kravspesifikasjoner og produkt. Innholdet består av grafisk funksjonalitet av webapplikasjon.

[Kapittel. 4 Testing Dokumentasjon](#). Dette kapittelet beskriver testing utført i prosjektet. Flernivå testing er blitt gjort på produkt og er beskrevet i utført rekkefølge.

[Kapittel. 5 Teknisk dokumentasjon](#). Dette kapittelet beskriver den tekniske funksjonalitet til produktet. Innholdet består av produkt struktur i forhold til front-end og back-end.

[Kapittel. 6 Diskusjon og konklusjon](#). Dette kapittelet beskriver gruppens tanker og refleksjoner rundt problemer og spørsmål som har oppstått før og under prosjektløp, i tillegg til konklusjon av prosjekt.

[Kapittel. 7 Forslag til videreutvikling](#). Dette kapittelet beskriver ulike forslag til videreutvikling av webapplikasjonen. Disse forslagene består av brukbare og relevante funksjoner.

Nederst i rapport ligger [ordliste](#) og beskrivelse av begreper som leser kan ha behov for å se gjennom. I tillegg til [bruksanvisning](#) for webapplikasjonen.

1. Introduksjon

Dette kapitlet er ment for å gi lesere en introduksjon til alle parter som tar del i prosjektet. I tillegg til en generell beskrivelse av prosjektet. Denne beskrivelsen er basert på problemstillingen og delproblemer som følger med, i tillegg til foreslalte løsninger.

1.1 Bakgrunn

Styret.com er et Norsk selskap som leverer tjenester og programvare til boligselskaper og leverandører til disse. Eksisterende plattform yter tjenester til tre brukertyper: vaktmestere, styrer og forretningsfører.

Styret.com har tidligere kjøpt Nabonett.no, et selskap og nettportal som forsøkte å lage en sosial plattform for naboer og samhandling mellom disse. Nabonett ble aldri en kommersiell suksess og ble kjøpt med en intensjon om å utvikle dette til en del av Styret.com, der eiere av bolig kunne finne informasjon om sin bolig og samhandle med sitt boligselskap.

Styret.com hadde i første omgang tenkt at bachelor-gruppen vår skulle lage en enklere webapplikasjon, men vi ønsket oss et større prosjekt for mer utfordring og læring.

1.2 Beskrivelse av prosjektet

I dette bachelorprosjektet har gruppen i samarbeid med Styret.com laget en web applikasjon som skal gi eiere av boliger i boligselskaper tilknyttet Styret.com oversikt over deres boliger. Webapplikasjonen skal gi brukerne en enkel og oversiktlig portal med oversikt over sin bolig eller sine boliger. Eierne kan logge inn og finne sin "reskontro", en oversikt over fakturaer fra boligselskapet og betalingsstatus for disse. I tillegg har eierne mulighet til å registrere beboere av boligen for blant annet å få informasjon fra styret vedrørende strømbrudd og andre relevante hendelser fra boligselskapet. I forbindelse med årsmøte eller generalforsamling vil eier kunne finne sin stemmeseddel her. Detaljerte beskrivelser er presentert i seksjon [3.3](#).

1.3 Problemstilling

Webapplikasjonen skal brukes av eiere i boligselskaper. Disse personene har veldig ulik kompetanse, erfaring og forutsetninger for bruk av en online tjeneste. Derfor må webapplikasjonen være brukervennlig og intuitiv. Nabonett skal gi brukerne muligheten til selv å finne informasjonen de ønsker seg, fremfor å kontakte forretningsfører eller styret sitt for å status på økonomi eller få tilsendt dokumentasjon om sin bolig.

Denne seksjonen tar for seg å forklare de ulike problemene som problemstillingen inneholder og løsningen på disse.

1.3.1 Oversikt over økonomi

Brukere skal ikke trenge å kontakte forretningsfører eller logge inn i nettbanken for å få vite om de skylder penger til boligselskapet. I de tilfellene hvor en bruker eier flere boliger kan det ofte være vanskelig å holde oversikt eller finne ut hva som er feil i de tilfellene noe ikke stemmer.

Dette ble løst gjennom å lage en kobling til forretningsførers system og finne økonomisk informasjon om den enkelte bolig. Vi henter alle fakturaer og innbetalinger som gjelder for valgt bolig for å presentere disse for brukeren på en enkel måte. Brukeren får enkel kontroll og oversikt over egen økonomi.

1.3.2 Dokumentasjon om egen bolig

Mange eiere mangler all dokumentasjon om egen bolig og andre har laget sine egne systemer eller tatt i bruk forskjellig programvare. Dette kan være dokumentasjon fra håndverkere, søknader til styret, klager til styret, generalforsamlingsvedtak om boligen eller rettigheter, eller dokumentasjon på utstyr, maling eller materialer i boligen. Ofte vil det også være relevant å dele dette med styret i boligselskapet.

Problemet er løst ved hjelp av en boligmappe i nabonett hvor bruker kan laste opp relevante dokumenter for boligen. Dokumentene følger boligen og er trygt lagret. Dette er blant annet veldig relevant når man skal selge boligen.

1.3.3 Stemmegivning og meningsmåling

Boligselskaper avholder både uformelle meningsmålinger og formell stemmegivning ved årsmøte/generalforsamling. Styret.com har funksjonalitet for å sende ut dette, men det er ikke alle som ønsker å gi sin kontaktinformasjon til styret, ofte av personvernghensyn. Disse vil i dag ikke kunne avgi sin stemme uten å levere fysisk stemmeseddel og det vil være problematisk å avgı sin stemme ved hemmelig valg.

Vi har koblet vår løsning mot Styret.com slik at stemmesedler er tilgjengelig for eiere i nabonett, selv om eieren ikke har registrert epost eller telefonnummer som er tilgjengelig for styret i boligselskapet.

1.3.4 Kontaktinformasjon og beboer

Boligselskaper i dag har oversikt over de som eier en bolig, men det er ikke alltid kontaktinformasjonen er oppdatert eller finnes i det hele tatt. I mange tilfeller er det kun en av beboerne i en bolig som er eier, men de andre beboerne, som ektefelle, samboer, barn og leietaker, har også et informasjonsbehov.

I nabonett kan eier registrere sin egen og beboere sin kontaktinfo, slik at også disse får relevant informasjon fra boligselskapet. Beboere er knyttet til eieren, slik at når eierskapet opphører i forbindelse med salg av bolig, blir beboere anonymisert for forretningsfører og styret. Dette i tråd med GDPR.

1.4 Gruppepresentasjon

Gruppen består av fire elever fra ulike studier ved OsloMet. Vi har jobbet sammen de tre siste årene og har gjennom dette samarbeidet blitt godt kjent med våre styrker og svakheter. Dette har gjort det mulig for oss å deleger arbeid og stole på at vi alle utfører en god jobb. Det at vi har forskjellige studier fører til at vi kan bringe et utbredt mengde med kunnskap og erfaring. Enkelte av oss er bedre på programmering enn andre, men da dekker dem andre viktige egenskaper som også behøves for prosjektet.

Navn:	Studentnummer:	Studie:
Duale Mohamed	s333972	Anvendt dataeknologi
Mohamud Abdi	s333774	Informasjonsteknologi
Bereket Tekeste	s331401	Dataingeniør
Mehboob Singh	s333722	Informasjonsteknologi

Prosessen for å finne aktuell arbeidsgiver startet tidlig for prosjektet og innebærte flere runder med epost og intervju. Vi kontaktet flere bedrifter om mulig samarbeid og endte til slutt med to potensielle arbeidsgivere. Resultatet ble at vi bestemte oss å gå videre med Styret.com. Grunnlaget bak denne beslutningen var at vi så en mulighet for å kunne utforske og forbedre våre programmering og utviklings evner gjennom et stort prosjekt, i tillegg til å bidra med å produsere et brukbart produkt som løser et manglende problem hos dem.

1.5 Introduksjon til Styret.com

Bedriften er et av Norges største bolig forvaltingsselskap og har per i dag over 1700 boligselskaper og sameier som dem styrer med i Norge. Selskapet har vært i drift i 4 år og kjører sterkere på for hvert år. Styret har tre kundetyper på samme plattform med hver deres applikasjon for styring. Tabellen under gir en generell beskrivelse av dette.

Vaktmesterselskap	Styrer	Forretningsfører-selskap
<ul style="list-style-type: none">- Profesjonell.- Drifter ofte over 300 eiendommer.- Håndterer både rutiner og andre hendelser.	<ul style="list-style-type: none">- Ikke-profesjonell.- Ansvar kun for en bolig.- Har ofte vervet fordi "ingen andre ville".	<ul style="list-style-type: none">- Profesjonell.- Administrerer ofte over 300 sameier.- Jobber med regnskap, rådgivning og administrasjon.

Styret.com har som arbeidsgiver bidratt med tjenester og verktøy brukt under utviklingen av web applikasjonen, i tillegg til kontinuerlig kommunikasjon for å avklare og assistere med problemer som har oppstått underveis i prosjektet. Som arbeidsgiver har Styret.com vært en respektable og hjelpsom kunde som har gitt oss lærerikt erfaring som vi vil verdsette og ta med oss videre.

Kristoffer G. Skappel | Daglig Leder | Styret.com

Kris@styret.com | Nedre Vollgate 3, 0158 Oslo



Kristoffer er tidligere uteksaminert elev fra HiOA (OsloMet) og per i dag Daglig leder for Styret.com. Som produkteier har han utarbeidet både funksjonelle og ikke-funksjonelle krav for system. I tillegg til å hjelpe oss med å jobbe målrettet. Som gruppens primære kontaktperson og arbeidsgiver så har vi hatt kontinuerlig kommunikasjon og samarbeid gjennom hele prosjekt.

Varun Patial | Programutvikler | Styret.com



Varun har vært gruppens tekniske kontaktperson hos Styret.com. Med hans bakgrunn som programutvikler har han bidratt til bedre forståelse av prosjektets teknologiske stack. Varun har hjulpet gruppen med forslag til hvordan bearbeide teknologiske problemer.

1.6 Rådgiver OsloMet

Raju Shrestha | Førsteamanuensis i Institutt for Informatikk |

Raju.Shrestha@oslomet.no | Pilestredet 35, 0166 Oslo



Raju er en erfaren professor og ingeniør som ledet oss ved å tilby gode forslag og løsninger til problemer som vi kom borti under prosjektarbeidet, gjennom regelmessig tilsyn.

2. Prosessdokumentasjon

I denne delen av prosjektet så tar vi for oss de ulike prosessene som blitt brukt for å effektivisere og redegjøre for god planlegging og metode. Dette blir gjort i samsvar med dokumenter som prosjektdagbok og forprosjektrapport som inneholder informasjon om prosjektet.

2.1 Planleggingsprosess

Denne prosessen startet tidlig i November 2020 etter å ha blitt enig om samarbeid med arbeidsgiver. I første møte med arbeidsgiveren så introduserte han oss for ulike prosjektforslag som han trengte hjelp med. Vi valgte å ta for oss en mer omfattende oppgave enn det han hadde planlagt, og gikk for å løse flere av forslagene gjennom en felles prosjekt/applikasjon. Deretter ble vi enige om offisielt oppstart 01.01.2021, men i mellomtiden så skulle vår gruppe gjøre så mye forarbeid som mulig for å stille oss sterkest klare til start. Dette krevde blant annet å bli kjent med verktøy som skal brukes i prosjektet. Noe som enkelte av oss ikke hadde noe erfaring i fra før av.

Arbeidsplan og fremdriftsplan

For at prosjektet skal opprettholde leveringsdato er det viktig å drøfte en plan. Denne planen er laget i forhold til ulike faser i prosjektet og tiden det tar for å ferdigstille dem. Denne planen fungerer som en rettesnor som både gruppe og arbeidsgiver kan forholde seg til.

Fase	Tid og dato:	Beskrivelse:
Forarbeid og planlegging	01.01.2021 - 15.01.2021	Tid brukt på å bli kjent med relevant verktøy som skal brukes til å utføre prosjekt.
Dokumentering og rapportering	01.10.2020 -	Dokumentering av all informasjon som bearbeides gjennom hele prosjektløp.
Skissere grensesnitt	15.01.2021 - 01.02.2021	Utvikle skisser av grensesnitt for applikasjon.
Utvikling av front-end	01.02.2021- 01.05.2021	Frontend og backend skal utvikles parallelt.
Utvikling av back-end	01.02.2021- 01.05.2021	Backend og frontend skal utvikles parallelt.
Testing av applikasjon	01.02.2021 - 01.05.2021	Testing av applikasjon for å opprettholde god funksjonalitet og standardisering.
Skriving av rapport	10.04.2021 - 25.05.2021	Selve finale rapport skrives sist, men har blitt bearbeidet siden prosjektstart.

2.2 Arbeidsmåte og prosessmodell

I dette prosjektet så har vi brukt "fossefallsmetoden" som er en plandrevet prosessmodell. Denne modellen er basert på å planlegge alle prosess aktivitetene før arbeidet er startet. Hver aktivitet er representert som en separat fase og er arrangert i en rekkefølge. (Sommerville, 2015, s. 30-31)

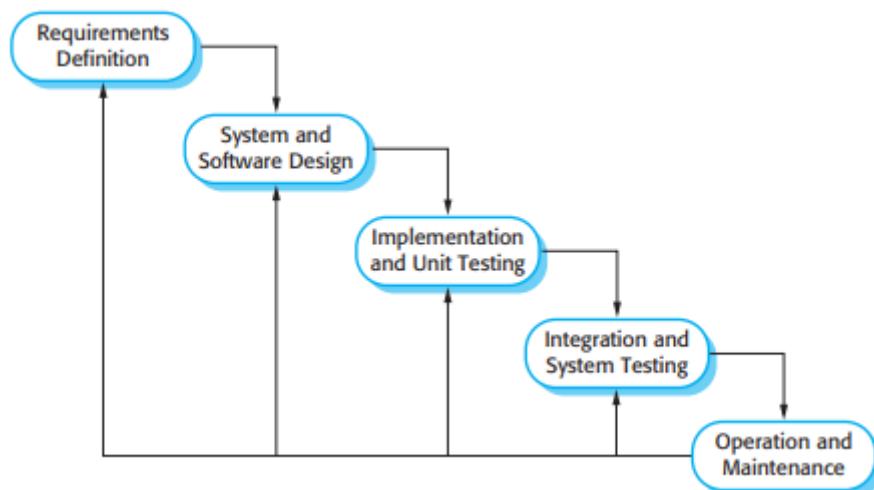
Hvorfor velge fossefallsmetoden?

Grunnene til at vår gruppe tar i bruk denne prosessmodellen er for opprettholde en formell arbeidsmetode som arbeidsgiver og rådgiver kan forholde seg til. Dette gjør prosjektet mer oversiktlig og muliggjør tidlig sporing av problemer.

Fossefallsmodellen inneholder en klar struktur og gjør det enkelt for gruppen å forholde seg til hver fase i prosjektet og sørge for fullført levering. På grunn av de allerede definerte målene og forventningene av sluttproduktet er denne metoden en god løsning for vårt prosjekt.

2.2.1 Faser i fossefallsmodeten

Fossefallsmodellen inneholder fem faser som satt opp i en lineær rekkefølge. En fase bør ikke starte før den andre er blitt fullført, men i praksis så kan enkelte av fasene overlappe.



Figur 2.1: Fossefallsmodellen

Kravspesifikasjon analyse og definisjon

Denne fasen involverer forståelse og definering av systemets krav, mål og begrensninger. Kravspesifikasjonene for et system er definert i denne fasen.

System og Software Design

Definerte kravspesifikasjoner fra forrige fase er brukt for å studere og produsere system design. Denne designen hjelper med å spesifisere programvare og systemkrav, og hjelper med å definere sammenlagt systemarkitektur.

Implementasjon og enhetstesting

Ved hjelp av tilførsel fra de to forrige fasene er systemet produsert i enheter som blir testet hver for seg under enhetstesting for funksjonalitet og feil. Hver enhet er produsert i henhold til å fungere hver for seg og skal inneholde en viss funksjonalitet beskrevet i kravspesifikasjonene som også kan testes.

Integrasjon og systemtesting

De individuelle enhetene er integrert sammen for å danne et helhetlig system som testes for å sjekke om kravspesifikasjoner og system design er blitt møtt. Etter fullført testing er systemet klar for levering til kunde.

Drift og vedlikehold

Dette er siste fase av fossefallsmodellen og definert som produktets lengste fase i livssyklusen. Etter fullført leveranse er systemet tatt i bruk og vedlikehold er istandgjort for å opprettholde funksjonalitet, i tillegg til opprettning av mulige problemer. Vedlikehold tar også utgangspunkt i forbedring av funksjoner som er avhengig kundens ønsker.

2.3 Kravspesifikasjoner og utviklingsprosess

Utviklingsprosessen startet med å bli bedre kjent med arbeidsgiver og bedriftens utvikler og ledelsesteam. Sammen så diskuterte vi og gjorde en behovsanalyse for å samle inn den informasjonen som er nødvendig for å designe et brukergrensesnitt. (Sandnes, 2018, s. 218)

Senere i prosessen definerte vi sammen med arbeidsgiver kravspesifikasjoner og deres prioritet, i tillegg til å produsere en prototype av GUI (graphical user interface). Kravspesifikasjonene er blitt definert ut fra arbeidsgiver ønsker og ved hjelp av brukerhistorier. Metodene beskrevet i kapittel 4 Requirements Engineering har sørget for at kravspesifikasjonene opprettholder riktig teoretisk bakgrunn. (Sommerville, 2015, s. 82)¹

Kravspesifikasjonene er inndelt etter Funksjonelle og Ikke-funksjonelle krav.

- **Funksjonelle krav** er beskrevet som hva systemet skal gjøre. Disse kravene er basert på systemet som blir utviklet og brukere av systemet. Funksjonelle krav spesifiserer de funksjonelle aspektene ved en programvare i form av dens funksjoner, input, output, unntak osv. (Sommerville, 2015, s. 85)²
- De **ikke-funksjonelle** kravene er derimot relatert til systemegenskaper som pålitelighet, ytelse, sikkerhet eller tilgjengelighet. Som kan spesifisere eller begrense hvordan systemet skal fungere i sin helhet. (Sommerville, 2015, s. 86)³

¹ Behovsanalyse

² Funksjonelle krav

³ Ikke-funksjonelle krav

Ikke-funksjonelle krav:	Prioritet:
Webapplikasjon skal være enkel og intuitivt.	Høy
Webapplikasjon skal være kjapp og enkel å navigere.	Høy
W3C prinsipper skal overholdes og ikke brytes.	Høy
GDPR og personopplysningsloven skal opprettholdes.	Høy
Design skal etterligne Styret.com sitt webportal for enklere crossover.	Middels
Webapplikasjonen skal brukes både på PC og mobil-enheter.	Høy
Webapplikasjonen skal være godt tilrettelagt for vedlikehold og skalerbarhet.	Middels

Figur 2.2: Ikke-funksjonelle krav tabell

Funksjonelle krav:	Prioritet:
Logg Inn	Systemet bør la brukere logge inn i applikasjonen med deres konto.
	Systemet bør la brukere få tilbake tilgang ved «Glemt passord».
Hjemmeside	Systemet tillater brukere å sjekke deres bolig i hjemmesiden.
Min Bolig	Systemet tillater brukere å legge til nye beboere.
	Systemet tillater brukere å endre på beboere.
	Systemet tillater brukere å slette beboer.
	Systemet tillater brukere å legge til dokumenter i gjeldende bolig.
	Systemet tillater brukere å sjekke økonomi for gjeldende bolig.
Skjema	Systemet tillater brukere å gi deres stemme i en stemning.

Min Side	Systemet tillater brukere å endre på informasjon i «Min side»	Høy
-----------------	---	-----

Figur 2.3: Funksjonelle krav tabell

2.4 Risikoanalyse

Risikostyring og analyse er en viktig del av prosjektarbeidet og går ut på forutse potensielle risiko som kan påvirke prosjekt timeplanen, kvaliteten av produkt som produseres, i tillegg til å muliggjøre tidlig anerkjennelse for å unngå problemet. (Sommerville, 2015, s. 595) Under prosjektarbeidet så kan det oppstå mange ulike risikoer som kan føre til forskjellige utfall. Utførelse av en risikoanalyse som forutser potensielle risikoer, og som muliggjør håndtering av disse er svært viktig.

Sammen med arbeidsgiver ble risikoanalyseprosessen utført for å bestemme sannsynligheten for at de ulike risikoene kan hende og samtidig måle hvor stor innvirkning det kan ha på prosjektet. (Sommerville, 2015, s. 598)

Hendelse:	Risikotype:	Sannsynlighet:	Grunnen bak?	Innvirkning:	Hvordan unngå?
Corona og Sykdom	Menneske	Veldig høy	Ikke overholdt gode nok smittevernregler. Forårsaket smitte blant gruppe.	Moderat	Unngå å samles for ofte, og jobbe så mye som mulig gjennom Zoom og meets.
GDPR og Personopplysningsloven	Teknologi Organisasjon	Lav	Ikke sikret data. Privat data blir aksesserbar av fremmede.	Veldig Høy	Sikre data på en trygg måte og kryptere det i database.
Miste data eller gi tilgang til feil person	Teknologi Menneske	Lav	Skade på database forårsaket av f.eks virus eller hacking.	Veldig Høy	Beskyttelse i API layer og backup for å unngå slike problemer.
Lite tid for å implementere nye ting	Organisasjon	Moderat	Vanskeligheter rundt planlegging med arbeidsgiver og estimering av tiden ting tar.	Moderat	Gjøre ferdig det viktigste og prioritere det som er beskrevet i MVP.
Ikke levere ferdig produkt	Teknologi Menneske	Lav	Vanskeligheter i kommunikasjon og estimering kan føre til at ting ikke blir ferdiggjort.	Veldig høy	Sette av tid og prioritere viktige funksjoner og ha det ferdig i forkant.

Figur 2.4: Risiko Analyse tabell

2.5 Verktøy og teknologier

Denne delen av prosessdokumentasjon tar for seg de ulike aspektene rundt hvilke teknologier som er blitt brukt til å produsere applikasjonen. Strukturen til prosjektet blir beskrevet i detalj under denne seksjonen, i tillegg til fordelingen av “klient” (front-end) og “tjener” (back-end). Produktet er produsert i henhold til et todelt prosjekt. Den ene inneholder front-end, mens den andre inneholder back-end. Seksjonen under er også beskrevet ut fra en slik inndeling og gir leseren forståelse overfor hvilke verktøy og teknologier som er blitt brukt i prosjektet.

2.5.1 Front-end teknologi

Valg av teknologisk stack er blitt bestemt på forhånd av prosjektstart. Dette ble bestemt av arbeidsgiver ettersom produktet skal tas i bruk og videreført av dem. Seksjonene under beskriver teknologiene og viser til deres bruk i prosjektet.

2.5.1.1 Angular

Angular er en plattform og rammeverk for bygging av “klient” applikasjoner som er enkeltsidet (single page application). Teknologien bruker HTML og Typescript. Angular har mange innebygde funksjonaliteter som muliggjør enkel håndtering av store applikasjoner på en side.

Arkitekturen til angular-applikasjon er basert på komponenter organisert i Ng-Modules mappen. Komponenter gjør det mulig for relatert kode å fungere som et funksjonelt sett. Front-end prosjektmappe har en “app” undermappe som inneholder “rot” modul og tillater bootstrapping av hele prosjektet. Prosjektet tar i bruk angular for å lage komponenter og modulbaserte elementer som kan vedlikeholdes og videreført.



Figur 2.5: Angular

2.5.1.2 Bootstrap

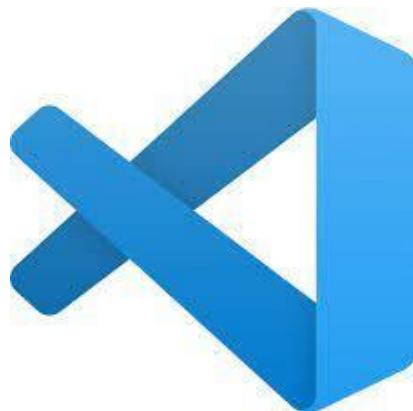
Bootstrap er en populær rammeverk som brukes til å bygge responsiv, mobil-først nettsider. Bootstrap er en kraftig verktøy som er basert på HTML, CSS og Javascript. Bootstrap er brukt i front-end for å gjøre den mer responsive. (Bootstrap, u.å.)



Figur 2.6: Bootstrap

2.5.1.3 Visual studio code

Visual studio code er en populær og fri kode-editor laget av Microsoft. Den kommer med innebygd støtte for flere teknologier som Javascript, Typescript og Node.js. Den har også store muligheter for utvidelse og kan lett importere utvidelser for andre språk som C++, C#, .NET core, osv. Visual studio code ble brukt for vår “klient side” applikasjon som var basert på Angular. Den har gitt oss mange muligheter for feilsøking, syntaks fremheving, utdrag og innebygd git.



Figur 2.7: Visual studio code

2.5.1.4 Ng-Bootstrap

Ng-bootstrap er Bootstrap baserte widgets som er tilpasset for angular prosjekter. Den gjør bruk av Bootstrap enkelt i Angular applikasjoner. Og trenger kun Bootstrap 4 CSS, men er ikke avhengig av Javascript. Ng-bootstrap er brukt blant annet for å lage interaktive komponenter som “Modal” og “Forms”.



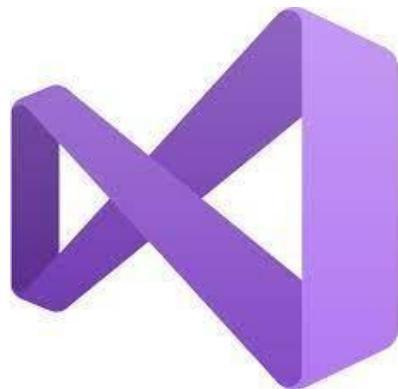
Figur 2.8: Ng-Bootstrap

2.5.3 Back-end teknologi

Denne seksjonen beskriver valg av back-end teknologi brukt i prosjektet. Som front-end ble også back-end teknologi bestemt på forhånd av arbeidsgiver.

2.5.3.1 Visual studio 19

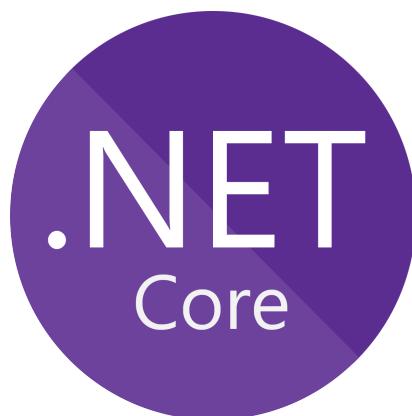
Visual studio er en integrert utviklingsmiljø(IDE) utviklet av microsoft. Den brukes for utvikling av forskjellige typer applikasjoner. Visual studio har en editor som gjør feilsøking, syntaks fremheving og andre funksjonaliteter enkelt. Visual Studio er brukt i tjener (back-end) som er basert på ASP .Net Core C#.



Figur 2.9: Visual studio 19

2.5.3.2 .Net Core

.Net core har erstattet .Net framework som er bygd av Microsoft. Den støtter alle aktuelle plattformer som Linux, windows og macOs. I tillegg er .Net core åpen kildekode. Ved bruk av Nuget package manager, er det enkelt å legge til avhengigheter man trenger i applikasjonen. Ettersom .Net framework blir utfaset og ikke lenger støttet av Microsoft, er det uaktuelt å ta i bruk den. Dermed brukes .Net core til å utvikle en multi-platform webapplikasjon som er sikret videreutvikling av Microsoft.



Figur 2.10: .Net Core

2.5.3.3 Azure data studio

Azure data studio er databaseverktøy som støtter aller platformer. Prosjektet bruker ADS for å håndtere tabller i database som simulerer den aktuelle databasen som skal brukes i applikasjonen.



Figur 2.11: Azure data studio

2.5.3.4 Docker

Docker er en åpen plattform for utvikling og kjøring av applikasjoner. Docker er perfekt for å kjøre database i et utviklingsmiljø. Et Docker bilde (Docker-image) inneholder et sett med instruksjoner for å lage en container som kan kjøre på Docker-plattformen. Dette gir en praktisk måte å pakke sammen applikasjoner og server miljøer, som brukes videre med andre Docker brukere. Docker er brukt for å lage docker-bilde for database (msSQL) og er satt opp på Azure Data Studio ved å bruke SQL logg-inn med brukernavn og passord opprettet via docker-bilde.



Figur 2.12: Docker

2.5.3.5 Swagger UI

Swagger er åpen kildekode og rammeverk for å beskrive API ved bruk av et format som er lett å lese og forstå av utviklere og testere. Verktøyet brukes for å teste verdier. Swagger UI tilbyr en HTML visning av API i både JSON og YAML. I utviklingen av dette prosjektet, ble Swagger brukt for å teste tjener (back end) og hvilken type data API kunne motta, sende, endre og slette. Data er formatert til JSON format og er enkel og lesbar på nettleseren.



Figur 2.13: Swagger UI

2.5.3.6 Postman

Postman er en populær API-klient som gjør det enkelt for utviklere å lage, dele, teste og dokumentere API. Dette blir utført ved oppretting og lagring av enkle og komplekse HTTP requests og lesing av deres responses. Ved bruk av Postman i dette prosjektet testet utviklerne om verdier kan endres, slettes og hentes. I utviklingsfasen er Postman også brukt for manuell integrasjonstesting når et spesifikt testscenario var nødvendig.



Figur 2.14: Postman

2.5.3.7 Bogus

Bogus er en enkel falsk data generator for C# og hjelper med å lage data verdier. Bogus forenkler prosessen med randomisering av data. Ved bygging av Webapplikasjonens database er det nødvendig med store mengder falske dataverdier. Ekte dataverdier er strengt tatt forbudt og ikke noe som kan brukes på grunn av GDPR og personopplysningsloven. (Personopplysningsloven, 2018, § 12-13) Dermed brukte utviklere Bogus for å generere falske dataverdier som er basert på et Norsk format.



Figur 2.15: Bogus

2.6 Design og prototype

Denne seksjonen i prosess dokumentasjonen skal vise fram prosjektets ulike design og prototyper. Gjennom prosjekt løpet har utviklere produsert flere prototyper av vårt design og har brukertestet det med relevant kundekrets. Hver iterasjon av design og prototype har vært gjennom arbeidsgiver for godkjennelse.

2.4.1 Første iterasjon av design og prototype

Dette ble utført i samarbeid med arbeidsgiver og utgjorde første utkast av hvordan webapplikasjonen skal se ut. Målet med første design er en simpel utkast som enkelt viser fram funksjonalitet bak systemkravene. Figurene under viser fram dette.

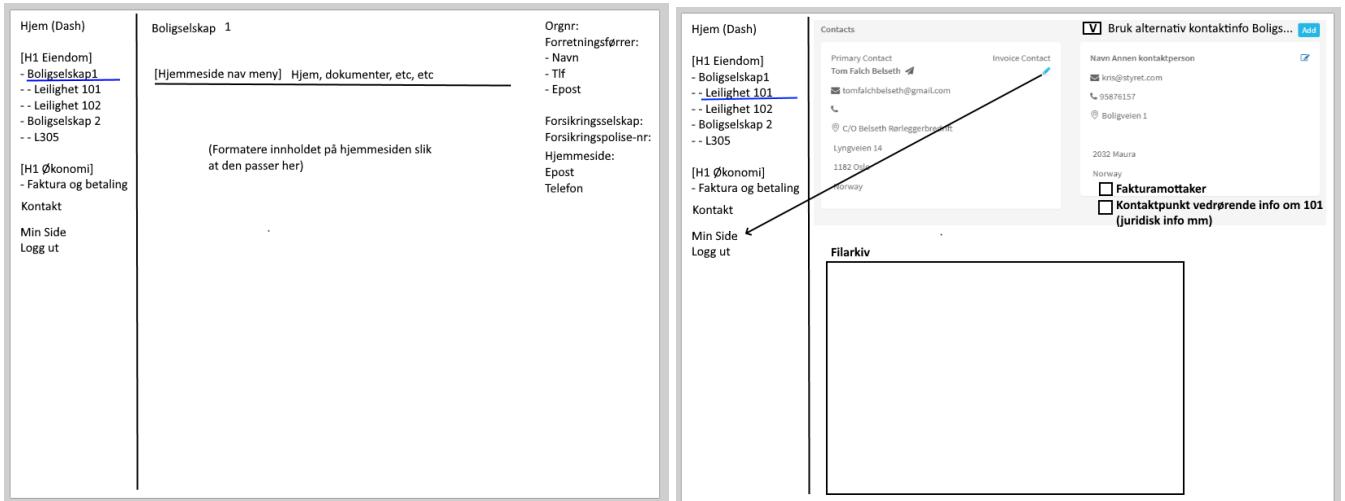
The image displays two wireframe prototypes of a web application interface, arranged vertically. Both prototypes are divided into columns for navigation, main content, and footer.

Top Prototype:

- Left Column (Hjem (Dash)):** Contains links for [H1 Eiendom] (with sub-links for Boligselskap1, Leilighet 101, Leilighet 102, Boligselskap 2, L305), [H1 Økonomi] (with sub-links for Faktura og betaling), Kontakt, Min Side, and Logg ut.
- Middle Column (Dash):** A large empty area representing the main content space.
- Right Column (Økonomi):** Contains a header with fnr, a search bar with '1022 Boligselskap1' and 'Leilighet 101', and a table with columns for Avsender, Tittel, Seksjon/andel/Aksje, and Status. The table lists 14 rows of voting data.

Bottom Prototype:

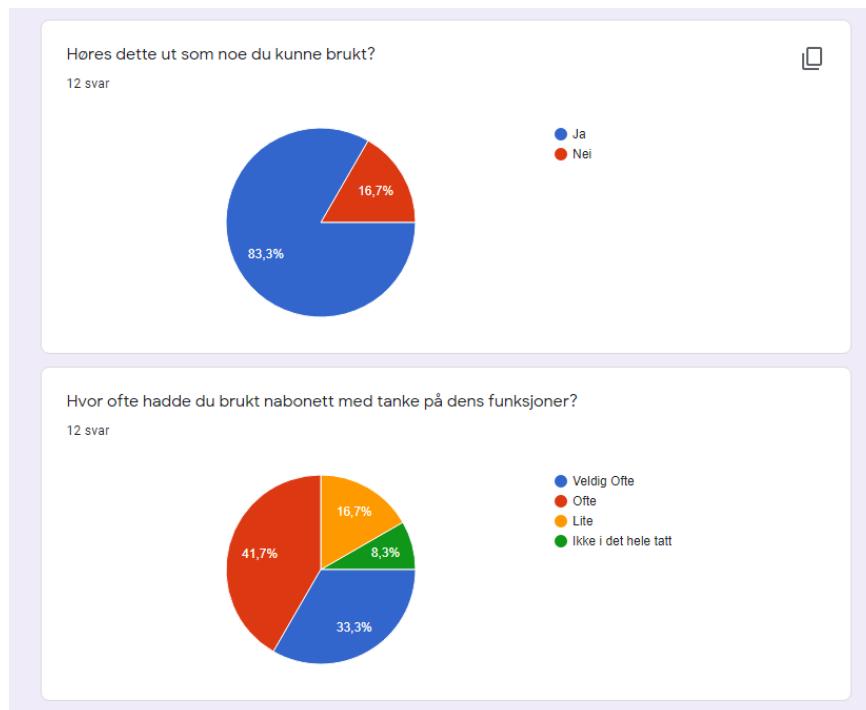
- Left Column (Hjem (Dash)):** Contains links for [H1 Eiendom] (with sub-links for Boligselskap1, Leilighet 101, Leilighet 102, Boligselskap 2, L305), [H1 Økonomi] (with sub-links for Faktura og betaling), Kontakt, Min Side, and Logg ut.
- Middle Column (Min side):** Contains fields for Navn, Epost, Telefon, Personnr, Postadresse, Etc, and several buttons: Bytt passord, Annen faktura-informasjon, bekreft ID med bankID, Bekrefte Epost, Bekrefte telefon, and Endre kontaktinfo.
- Right Column (Hjem (Dash)): Meldinger og spørreundersøkelser og avstemming**
- Table:** A detailed table showing a list of 14 voting entries (Avstemming 1 to 14) with columns for Datotid, Tittel, Avsender, Seksjon/andel/Aksje, and Status. The table shows alternating rows for Boligselskap1 and Boligselskap2, with sections 101, 230, and 102.



Figur 2.16: Første utkast

2.4.2 Kunde undersøkelse

Sammen med første utkast av prototype så utførte vi en brukerundersøkelse sendt til beboere av boligselskap. Selv om spørreundersøkelse ikke bør nødvendigvis være det første metode for informasjonsinnhenting etter en behovsanalyse. Gruppen ble enige med arbeidsgiver at dette er beste valget med tanke på vår begrensende tidsramme og tilgjengelighet til beboere. Spørreundersøkelse besto av to enkle spørsmål av kvantitativ metode som kan besvares innen fem minutter. Testpersonene leste innledningen av prosjekt før besvaring. Resultatet av undersøkelsen er presentert i figur 2.17 under.

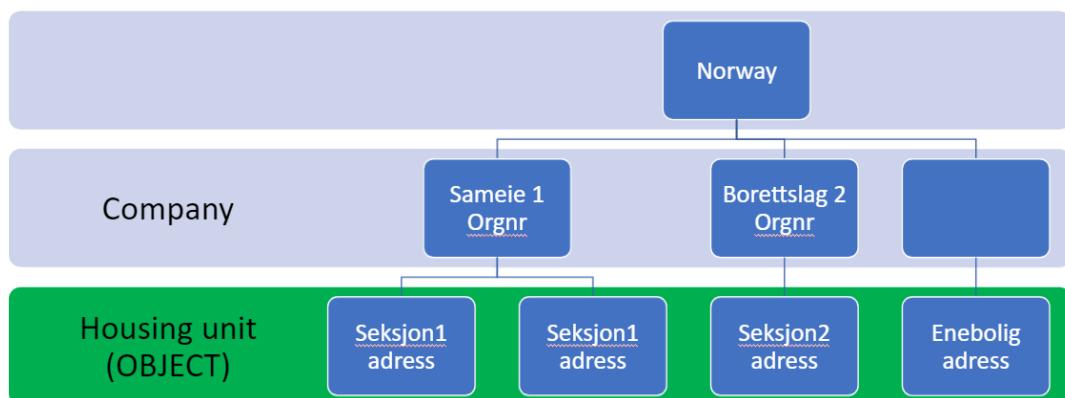


Figur 2.17: Resultat

2.4.3 Database og Backend design

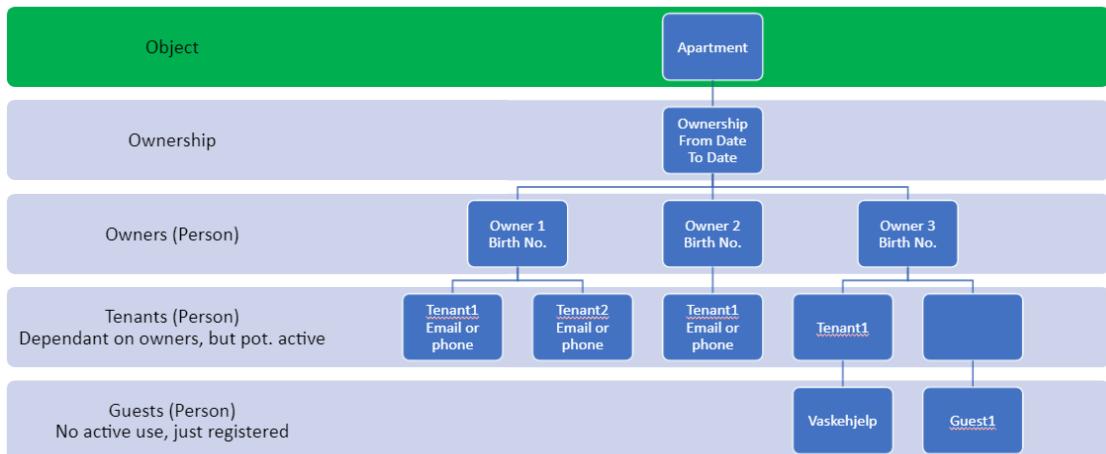
Sammen med arbeidsgiver laget vi et utkast av hvordan database oppsettet skal foretas. Dette er blitt gjort med fokus på en relasjonsbasert databasemodell. Selve utkastet består av enkle bilder og er ikke reflekterende overfor ferdig produkt som vises til i seksjon [2.7](#).

Real Estate



Figur 2.18: Eksempel av housing tabeller

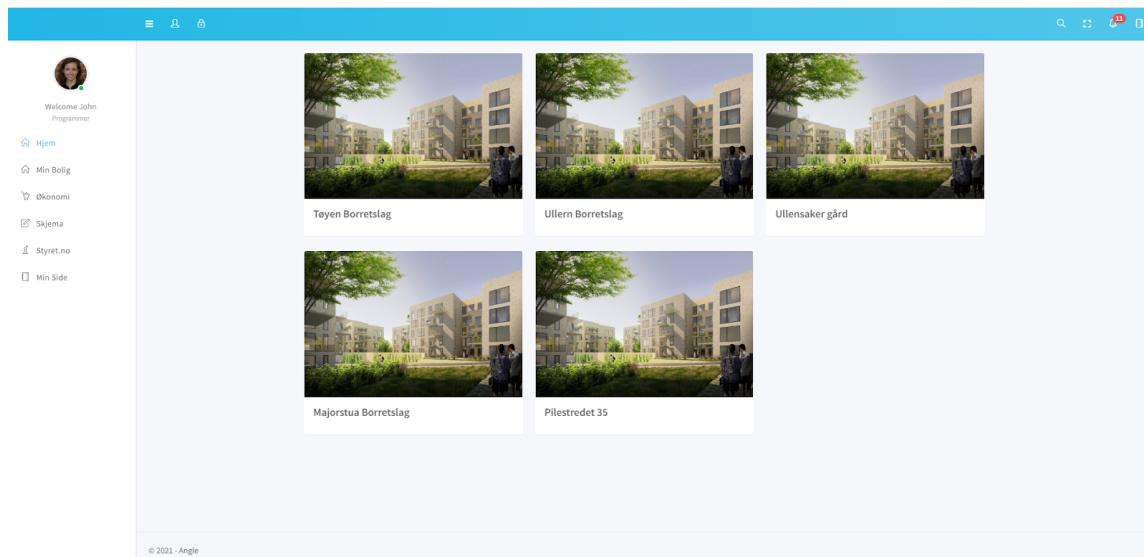
User Entities



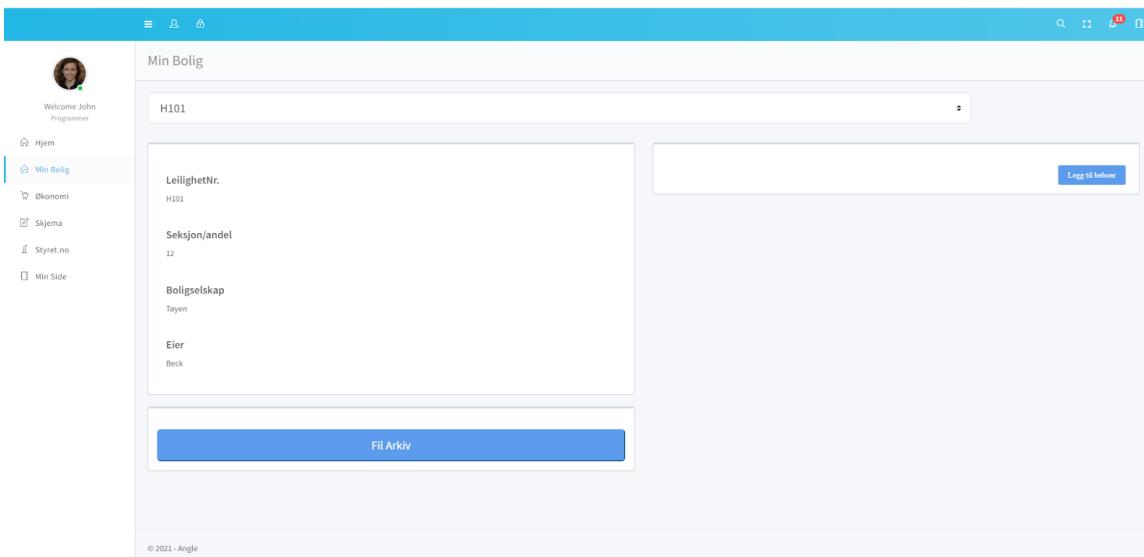
Figur 2.19: Eksempel av “User”, “Owner” tabeller

2.4.3 Andre iterasjon av prototype og design

Neste prototype er basert på realisering av funksjonaliteten beskrevet i Minimum viable product. Dette utføres gjennom Angular template i Front-end. Prototype besto av kun front-end kode og illustrerer hvordan funksjonalitet skal se ut. Arbeidsgiver er fornøyd med denne prototypen og kom til enighet om å gå videre med dette designet. Ettersom vi tar i bruk fossefallsmodellen som krever større kostnader og tid ved hver iterasjon, er det særdeles viktig å komme til enighet om design tidlig.



Figur 2.20: Hjemmeside



Figur 2.21: Min bolig

The screenshot shows a user interface for managing invoices. On the left, there is a sidebar with a profile picture and the text "Welcome John Programmer". Below this are links: "Hjem", "Min Bolig", "Økonomi" (which is highlighted with a blue border), "Skjema", "Styret.no", and "Min Side". The main content area has a header "Min økonomi". Below it is a table with columns: "Faktura Nummer", "Konti", "Forfall", "For periode", "Beløp", and "Status". The table contains three rows of data:

Faktura Nummer	Konti	Forfall	For periode	Beløp	Status
01	faktura.2141	01.01.2021	Januar	1099	Lukket
02	faktura.2141	01.01.2021	Januar	1099	Åpen
03	faktura.2141	01.01.2021	Januar	1099	Lukket
01	faktura.2141	01.01.2021	Januar	1099	Lukket
6 total					

At the bottom of the table, there are navigation arrows and a page number indicator "1 2 3". At the very bottom of the screen, there is a copyright notice: "© 2021 - Angle".

Figur 2.22: Økonomi

The screenshot shows a contact information form. On the left, there is a sidebar with a profile picture and the text "Welcome John Programmer". Below this are links: "Hjem", "Min Bolig", "Økonomi" (which is highlighted with a blue border), "Skjema", "Styret.no", and "Min Side". The main content area has a header "Kontakt Informasjon". Below it is a form with fields for "Navn" (Petter Solberg), "Epost" (mail@example.com), "Telefon +47" (92 32 22 46), "Gate Adresse" (Osloveien 21), "Postnr" (0363), and "Poststed" (Oslo). There is also a checkbox labeled "Favorite contact?" which is unchecked. At the bottom of the form is a blue button labeled "Oppdater". At the bottom of the screen, there is a copyright notice: "© 2021 - Angle".

Figur 2.23: Min side

2.4.4 Presentere dette til arbeidsgiver

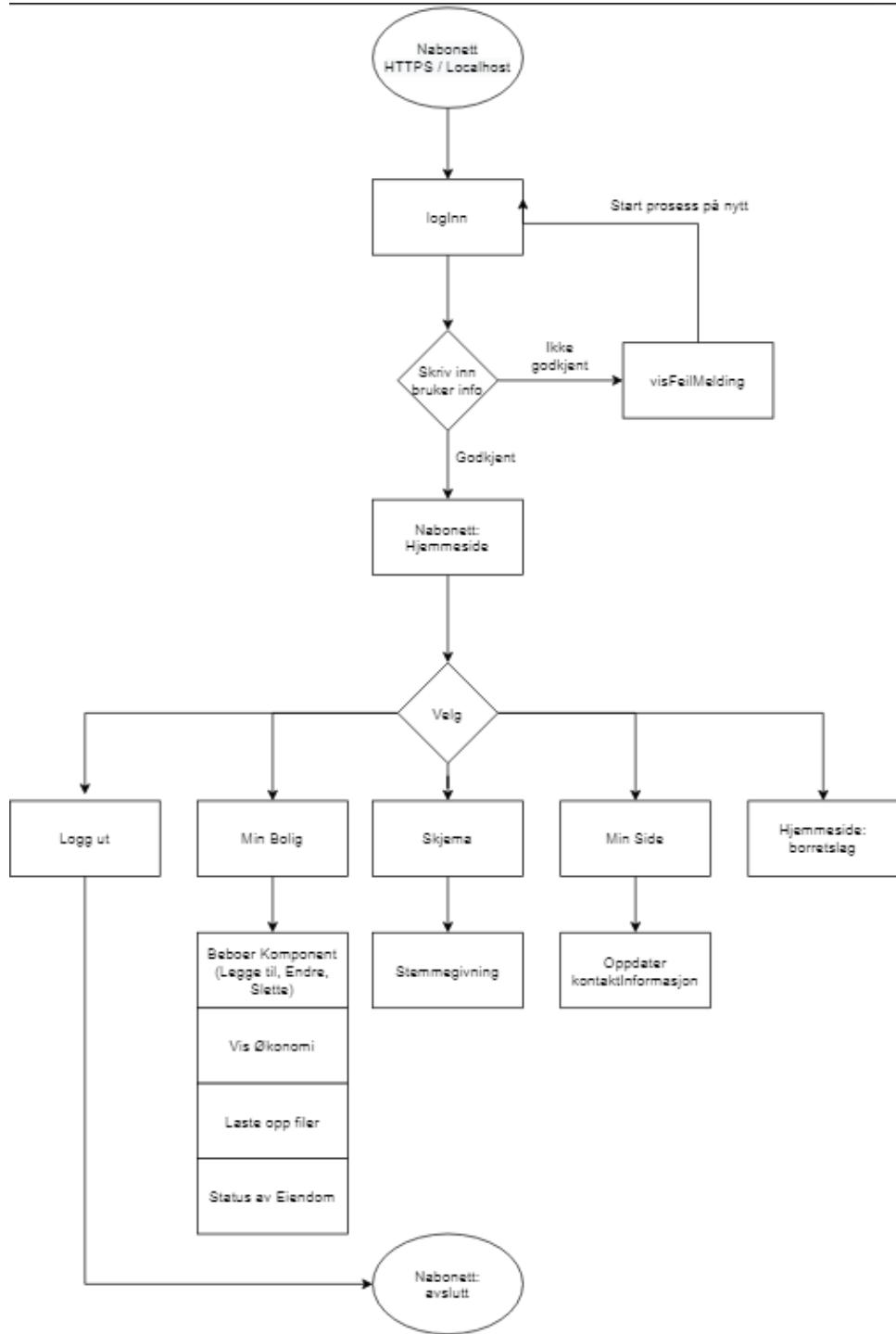
Arbeidsgiver er presentert for andre prototype produsert ved bruk av Angular template. Arbeidsgiver uttrykte god fornøyelse overfor prototypen og godkjente dermed videreføring av design basert på denne prototypen. Endringer i design gjort i forhold til prototype er dokumentert videre i seksjon [3.3](#).

2.7 System spesifikasjoner

Denne seksjonen tar seg å beskrive ulike illustrasjoner brukt for å illustrere web applikasjonens strukturelle oppbygning. Seksjonen inneholder flere diagrammer som viser til en oversiktig oppbygging av innholdet. Dette er blitt gjort i samarbeid med produkteier.

2.7.1 Flytskjema (Flow chart)

Dette er en visuell bilde av valgene og prosessene bak Nabonett webapplikasjon. Flytskjemaet (flow chart) viser gjennom en grafisk bilde hvilke steg som tas i et rekkefølge for å presentere flyten i systemet. Dette diagrammet viser fram de ulike prosessene som Nabonett inneholder, i tillegg til valgene som tas for å flyte gjennom webapplikasjonen.

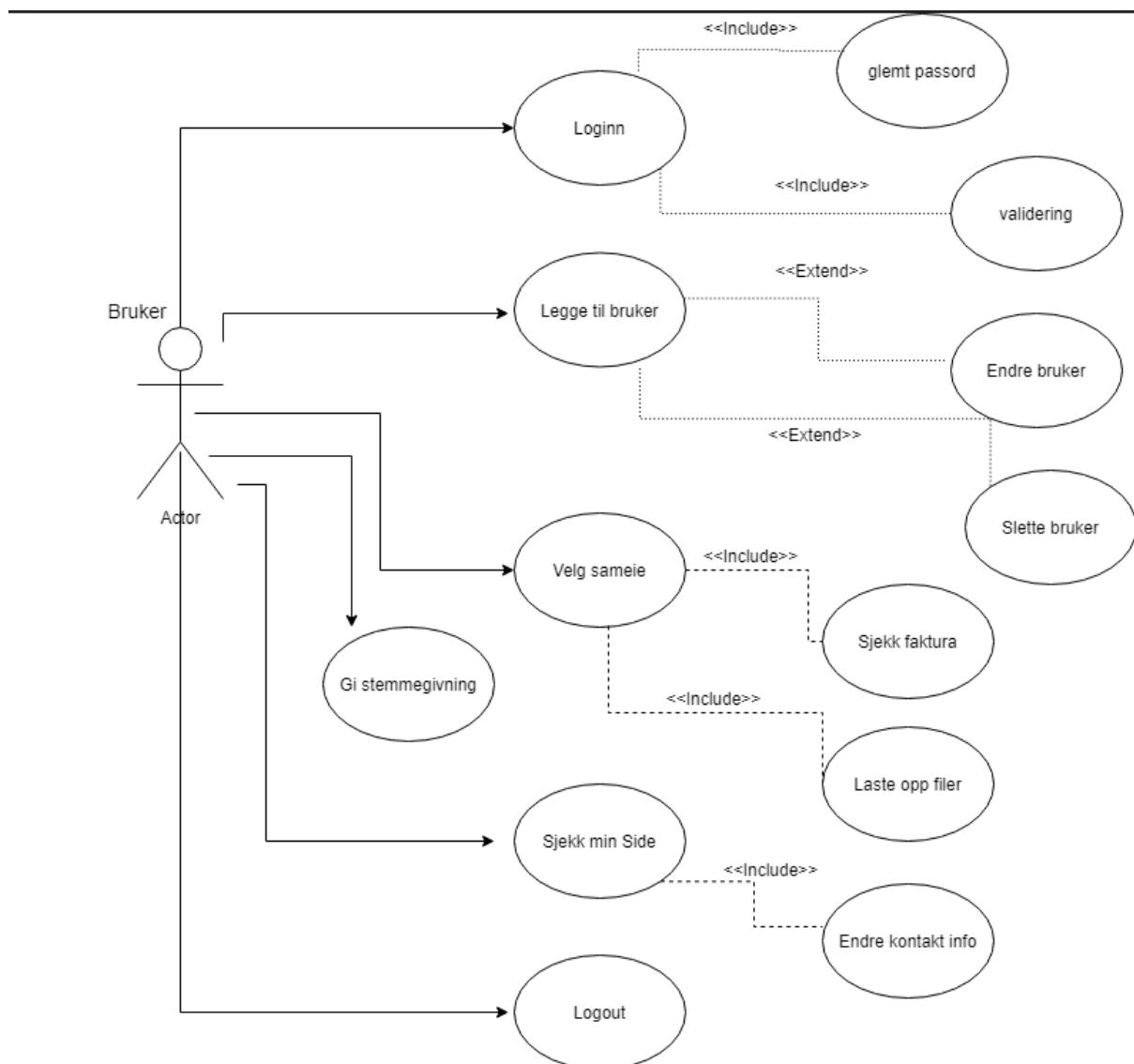


Figur 2.24: Flytskjema diagram

2.7.2 Use-case diagram

Denne seksjonen er interaksjonsmodellen use-case diagram. Dette diagrammet illustrerer enkelt hvordan en bruker skal samhandle med systemet. Use-case modeller blir ofte brukt for å modellere interaksjonen mellom system og aktør, det kan anses som hva en bruker forventer fra et system. (Sommerville, 2015, s. 124)⁴

Figur 2.25 viser til use-case diagram brukt i dette prosjektet.



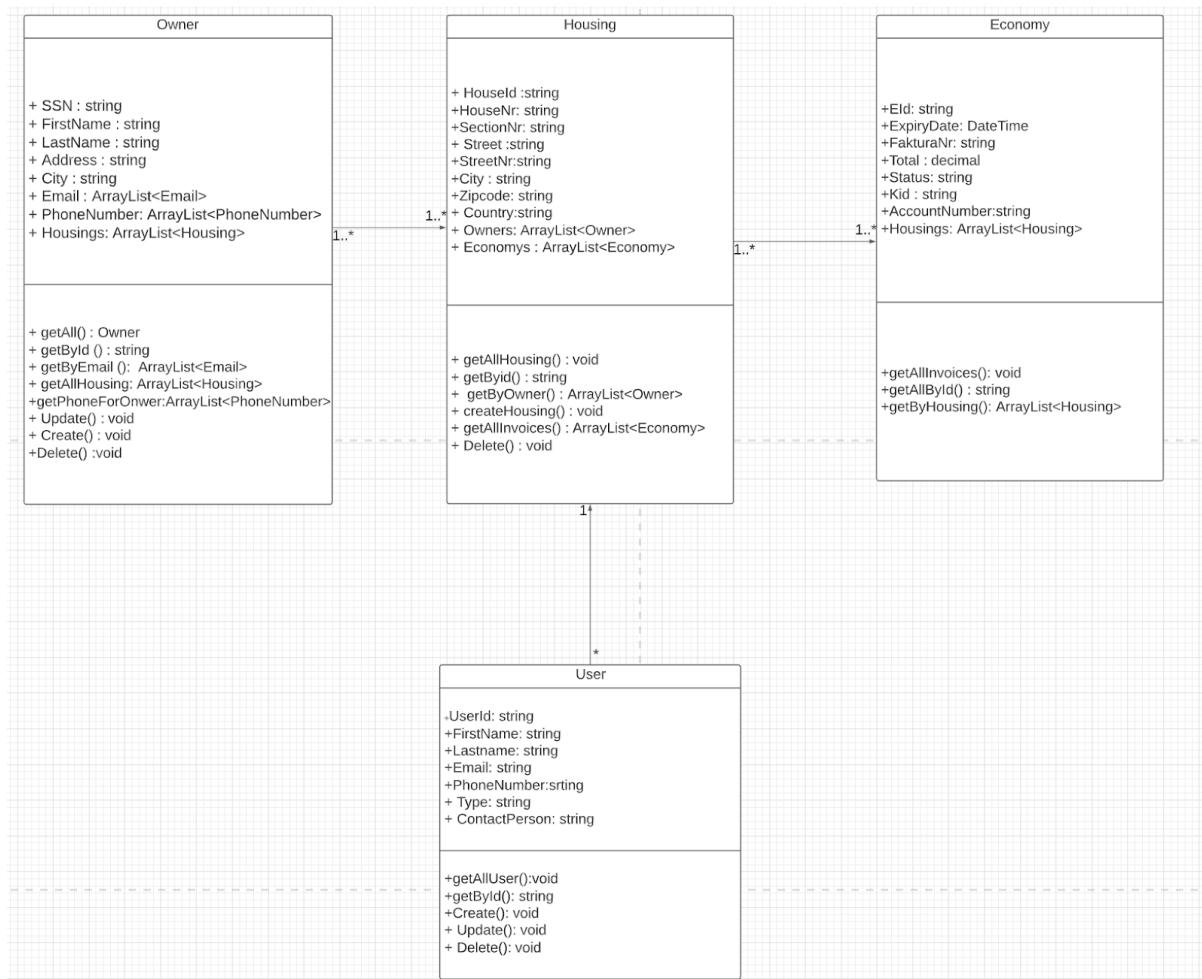
Figur 2.25: Use-case diagram

⁴ Use-case modellering

2.7.3 Entity Relasjon Diagram

Planen med arbeidsgiver var å ta i bruk deres allerede eksisterende database for å hente relevant bruker data. En slik endring i kravspesifikasjon er gjort for å minske faren for datatap. Løsningen ble deretter å produsere egen database med Dummy verdier. Database design er gjort gjennom en Entity relationship diagram (ER Diagram) som viser data enheter, deres assosierte attributter og relasjonen mellom disse enhetene. (Sommerville, 2018, s. 130).⁵

Database strukturen er produsert med tanke på denne modellen og har i tillegg blitt strukturert for å gi arbeidsgiver enkel overgang til deres egne database med ekte dataverdier i produksjon miljøet.



Figur 2.26: ER diagram

⁵ Semantiske modeller: ER Diagram

3. Produktdokumentasjon

3.1 Introduksjon til Nabonett - Webapplikasjon

Nabonett er en web portal som skal aksesseres av brukere som består av beboere innen Styret.com sitt boligforvaltnings krets. Disse beboere kan bestå av personer av ulike alder og kunnskapsnivåer som eier boliger. Dermed er det særdeles viktig at applikasjonen er enkel å bruke og navigere. Brukere kan styre deres boliger gjennom applikasjonen og kan sjekke faktura som gjelder hver enkelte bolig. Brukere kan i tillegg legge til dokumenter inn i en boligmappe og ha det tilgjengelig.

Brukere som eier bolig har muligheten til å legge til nye beboere bestående av enten gjester eller leietakere. Disse kan registreres som fakturamottaker og/eller kontaktperson hvis det trengs. Dette gir brukere enkel styring av deres boliger og oversikt over hvilke personer som er tilknyttet hver bolig. (*nye brukere lagt inn har ikke tilgang til system*). Brukeren kan også endre på eller slette disse beboerne i de tilfellene det trengs.

Innimellan så blir det avholdt generalforsamling som brukere av Nabonett kan stemme på gjennom en digital stemmeseddel. Webapplikasjonen gir brukere evnen til å endre på kontaktinformasjonen deres inne i ”Min side”. Funksjonaliteten er beskrevet i større detalj i seksjon [3.3](#).

3.3 Grafisk grensesnitt og funksjoner

Funksjonalitet beskrevet i denne seksjonen fordeles inn etter hver side i webapplikasjonen. Dette gir enkel forståelse for lesere og gir en kronologisk følgeseddel. Denne seksjonen beskriver funksjonalitet i større detalj enn det tidligere er blitt gjort i prosjektet. Lesere skal kunne oppnå høyere forståelse for applikasjonen og hvordan den fungerer etter å ha lest gjennom denne seksjonen.

3.3.1 Logg-inn side

Nabonett sitt første side som brukere blir introdusert til er logg-inn siden. Her skal man skrive inn epost og passord for å få tilgang til web applikasjonen.

Autentisering av brukere blir gjort ved hjelp av Azure AD B2C som støtter OAuth 2.0 og OpenID Connect. Dette blir fullført gjennom JSON web tokens (JWT) som autentiserer og gir sikker tilgang til ressursene. (Martin, Horvitz, Krish, Gomez & Macy, 2020, s.1)⁶

A screenshot of a web-based login form. At the top right is the logo 'styret.com' with a small icon of a key inside a lock. Below the logo is a field labeled 'Email Address' containing the text '§333972@oslomet.no'. To the right of this field is a link 'Forgot your password?'. Below the email field is a password input field containing several dots ('.....'). To the right of the password field is another link 'Forgot your password?'. At the bottom right of the form is a blue button labeled 'Sign in'.

Figur 3.2: LoggInn side

Sikkerhet

En av de viktigste aspektene ved applikasjonen er sikkerhet. Dette er blitt løst gjennom flere metoder for å sikre webapplikasjonen på en trygg måte og sikre informasjonen til brukere. Registrering er ikke mulig som minsker muligheten for angrep fra ukjente aktører og lignende sårbarheter. Brukere blir tildelt deres konto gjennom Styret.com og sikrer kun tilgang til personer som har deres bolig forvaltet gjennom firmaet.

⁶ JWT Azure AD B2C

Single Sign-on

For å sikre applikasjonen enda grundigere er Single-Sign-On ikke tatt i bruk (SSO). Grunnen bak dette er for å minske antall sårbarheter, og feil som kan oppstå ved usikre tredjeparts konto. Ved å tillate brukere tilgang til Nabonett gjennom Styret.com konto er webapplikasjonen utsatt for en sårbarhet som kan bli utnyttet av ukjente aktører.

Spesielt i de tilfellene bruker ikke har sikker nok konto brukt i SSO innlogging. OWASP sine “Security-by-design” prinsipper som “Minimize attack surface area”, “Don't trust services” og “Keep security simple” kan bli brutt hvis SSO ikke blir brukt på en riktig og trygg metode. (Van der Stock & Frenchie, 2014, s. 1)⁷. Derfor tildeles brukere kontoinnloggingene som kun fungerer på Nabonett.

Autentisering

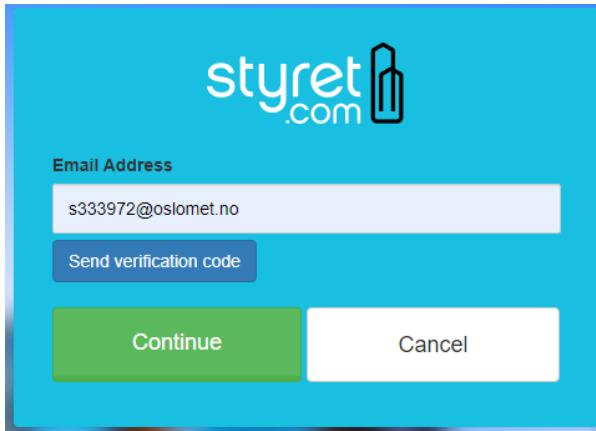
Logg-inn prosessen tar i bruk autentiserings faktoren “Authentication by knowledge” som går ut på at brukeren skal bevise sin identitet gjennom informasjon som dem vet om. I dette tilfellet er informasjonen “passord”. (Harris & Maymi, 2018, s. 878)⁸

3.3.1.1 Tilbakestilling av passord

I de tilfellene en bruker glemmer deres passord kan dem gå gjennom en tilbakestillings prosess for å danne en ny passord. Dette blir gjort ved hjelp av “Forgot your password?” som aksesseres gjennom Logg-inn siden. Ved å trykke på lenken og skrive inn mailadresse blir man tilsendt en verifikasjonskode på mail. Denne koden skal tastes inn og blir den verifisert før bruker mulighet til å lage en ny passord for konto. Figurene under viser fram denne prosessen.

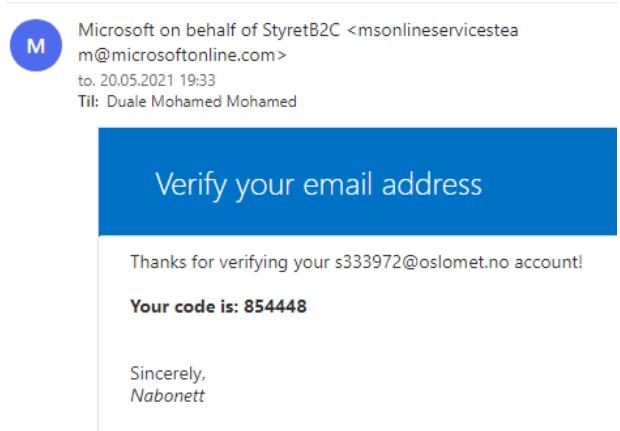
⁷ OWASP Secure Coding principles

⁸ CISSP autentisering

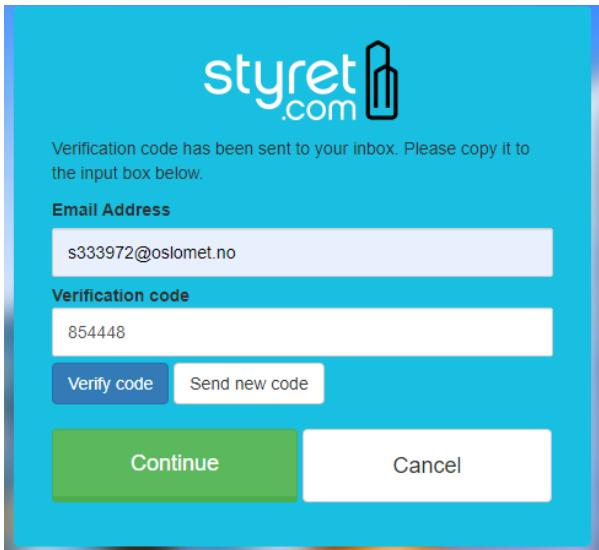


The screenshot shows a blue-themed web page for Styret.com. At the top is the Styret logo. Below it is a form field labeled "Email Address" containing "s333972@oslomet.no". To the right of the field is a blue button labeled "Send verification code". Below these are two large buttons: a green one labeled "Continue" and a white one labeled "Cancel".

Figur 3.3: Send verifikasjon

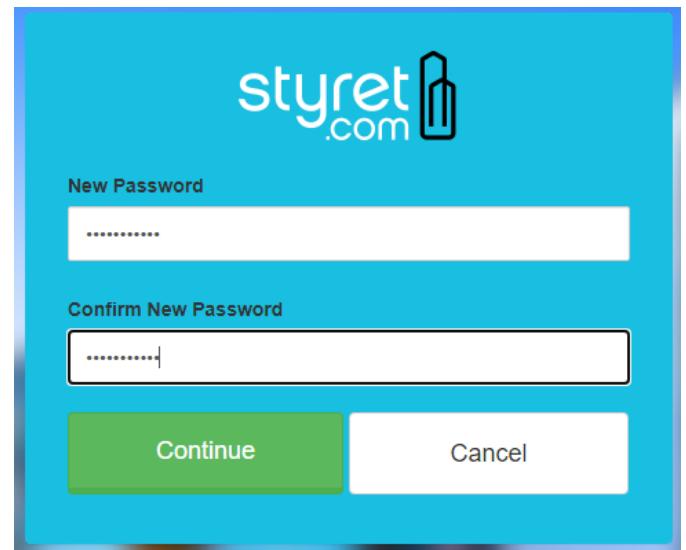


Figur 3.4: Tilbakestillings mail



The screenshot shows a blue-themed web page for Styret.com. It displays a message: "Verification code has been sent to your inbox. Please copy it to the input box below." Below this is a form field labeled "Email Address" with "s333972@oslomet.no". Underneath is another field labeled "Verification code" containing "854448". To the right of the code field are two buttons: "Verify code" (blue) and "Send new code" (white). At the bottom are two large buttons: a green one labeled "Continue" and a white one labeled "Cancel".

Figur 3.5: Verifisert kode

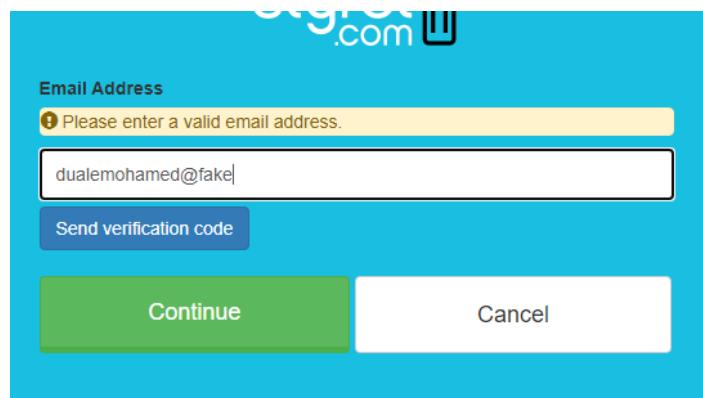


The screenshot shows a blue-themed web page for Styret.com. It has fields for "New Password" and "Confirm New Password", both containing ".....". At the bottom are two large buttons: a green one labeled "Continue" and a white one labeled "Cancel".

Figur 3.6: Tilbakestill passord

Klientside validering

Tilbakestilling av passord funksjon tar i bruk form-validering som sjekker at data er blitt innført i riktig format definert i webapplikasjonen klientside kode. Feil blir oppdaget gjennom validering og sørger for at riktig data, er i riktig format. Figurene under viser til et slikt eksempel.

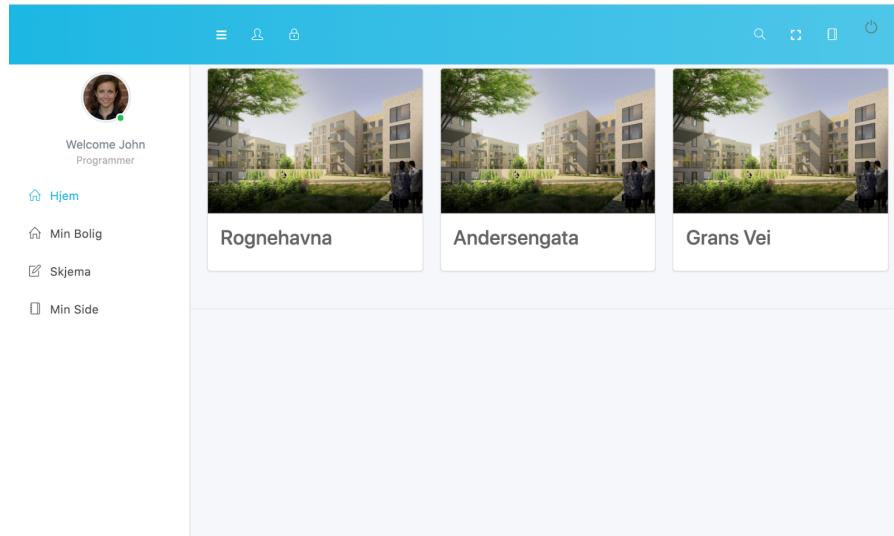


The screenshot shows a web form with a light blue header containing the text 'Email Address'. Below it is a red error message box with the text 'Please enter a valid email address.' A text input field contains the value 'dualemohamed@fake'. Below the input field is a blue button labeled 'Send verification code'. At the bottom of the form are two buttons: a green 'Continue' button on the left and a white 'Cancel' button on the right.

Figur 3.7: Validering av mailadresse

3.3.2 Hjemmeside

Dette er første side alle verifiserte brukere av systemet blir henvist til. Her blir alle boliger tilknyttet bruker vist, i form av boligselskap og/eller adresse. Brukere kan enkelt navigere gjennom ett tastetrykk og få tilgang til deres spesifikke bolig i "Min bolig". Annet funksjonalitet kan også nås gjennom navigasjon menyen på venstre side ved å trykke på tilsvarende ikoner. Figur 3.8 viser fram både hjemmesiden og navigasjonsmeny.



Figur 3.8: Hjemmeside

3.3.3 Min bolig

Det er “Min bolig” siden som inneholder mesteparten av funksjonaliteten til webapplikasjonen. Den inneholder en beboer komponent som tillater legge til, slette og endre beboer funksjonene. I tillegg til filopplasting og økonomi visning

Beboere kan bestå av mange forskjellige personer med hver deres unike forhold til en bolig. Disse er blitt generalisert inn som enten leietaker eller gjest. Dette er for å skille og ha enkel oversikt over deres forhold til boligen. Beboere skal ikke ha tilgang til Nabonett i første omgang.

3.3.3.1 Legge til beboer

Brukere skal kunne ha muligheten til å legge til nye beboere knyttet til deres bolig. Webapplikasjonen har forenklet denne prosessen ved å gjøre det så smertefritt som mulig gjennom få tastetrykk. Brukeren kan trykke på “Legg til beboer” knapp for å få opp “Registrer bruker” modal som fylles opp med informasjonen til beboer som legges til. Deretter blir den nye brukeren bekreftet og registrert gjennom “Lagre” knapp. Figur 3.9 viser “Legg til beboer” knapp og figur 3.10 viser modalen for registrering.

The screenshot shows a user interface for managing residents. On the left, a sidebar menu includes 'Hjem', 'Min Bolig' (selected), 'Skjema', and 'Min Side'. The main area displays a table of residents with columns for address, name, and actions ('Økonomi', 'Last opp'). Below this is a 'Beboere' (Residents) section with a table for Ida Borge and buttons for 'Slett' (Delete) and 'Endre' (Edit). A 'Legg til beboer' (Add resident) button is at the bottom right.

Figur 3.9: Beboer komponent

The modal dialog is titled 'Registrer Beboer'. It has a dropdown 'Leietaker Bruker' set to 'Leietaker'. The form fields are: Fornavn (Duale), Etternavn (Mohamed), Telefon (48502245), Email (dualemohamed1@hotmail.com). There are two checkboxes: 'Fakturamottaker' and 'Kontaktperson'. At the bottom are 'Lagre' (Save) and 'Avbryt' (Cancel) buttons.

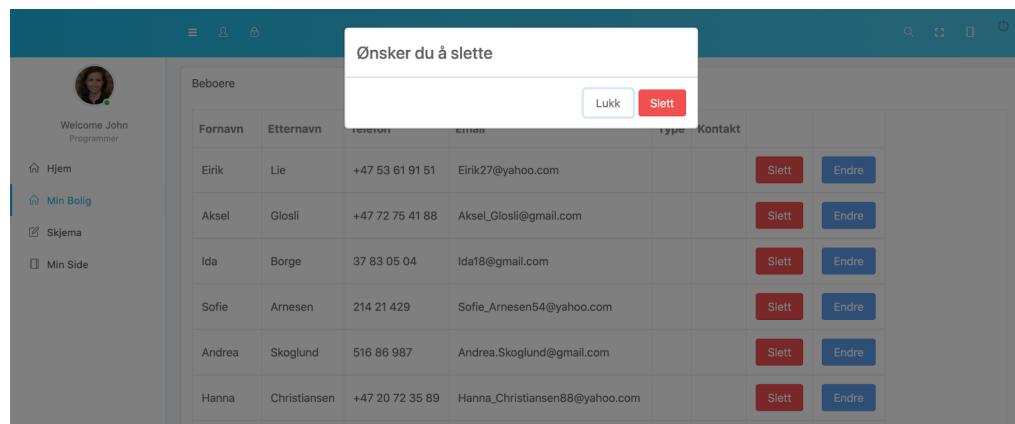
Figur 3.10. Legg til beboer modal

3.3.3.2 Slette beboer

Denne funksjonen er en viktig del av hele beboer komponenten. Brukeren skal kunne ha muligheten til å slette eventuelle beboere fra systemet. Det kan ofte oppstå situasjoner hvor en beboer har flyttet ut og skal dermed ikke lenger være i systemet. Figur 3.11 viser fram “Slett” knapp i Beboer komponenten. For at sletting skal kunne konkluderes så blir bruker presentert for en siste bekreftelses vindu for å bekrefte eller avslutte valg.

Beboere						
Fornavn	Etternavn	Telefon	Email	Type	Kontakt	
Karoline	Amundsen	+47 57 31 92 34	Karoline.Amundsen86@hotmail.com			<button>Slett</button> <button>Endre</button>
William	Vedvik	039 41 502	William.Vedvik@gmail.com			<button>Slett</button> <button>Endre</button>
Marie	Larsen	253 69 453	Marie46@hotmail.com			<button>Slett</button> <button>Endre</button>
Elise	Glosli	314 20 391	Elise_Glosli78@gmail.com			<button>Slett</button> <button>Endre</button>
Sara	Kristensen	17 52 18 68	Sara_Kristensen69@gmail.com			<button>Slett</button> <button>Endre</button>

Figur 3.11: Slette knapp

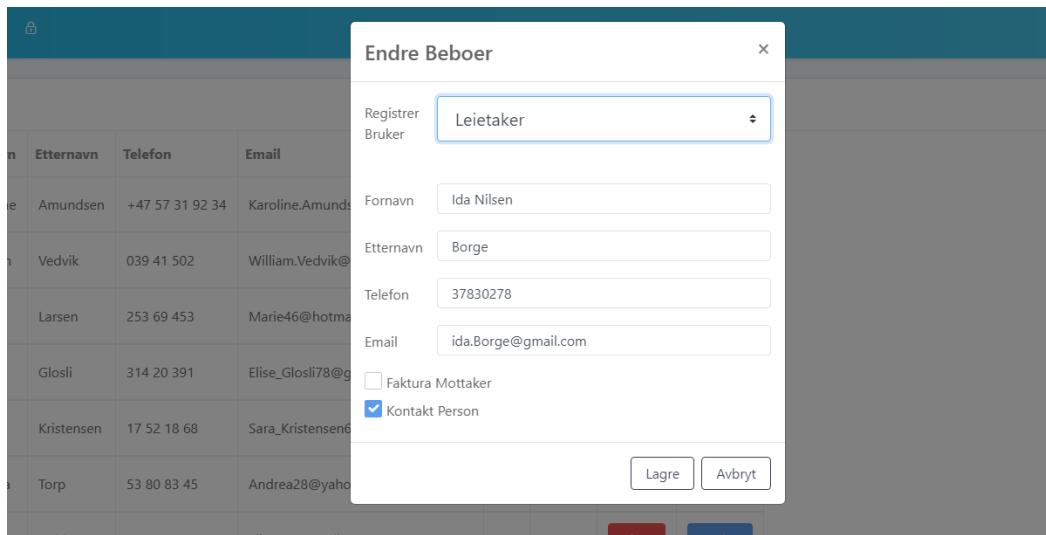


Figur 3.12: Slett Bekreftelse⁹

⁹ Bekrefteles vindu

3.3.3.3 Endre beboer

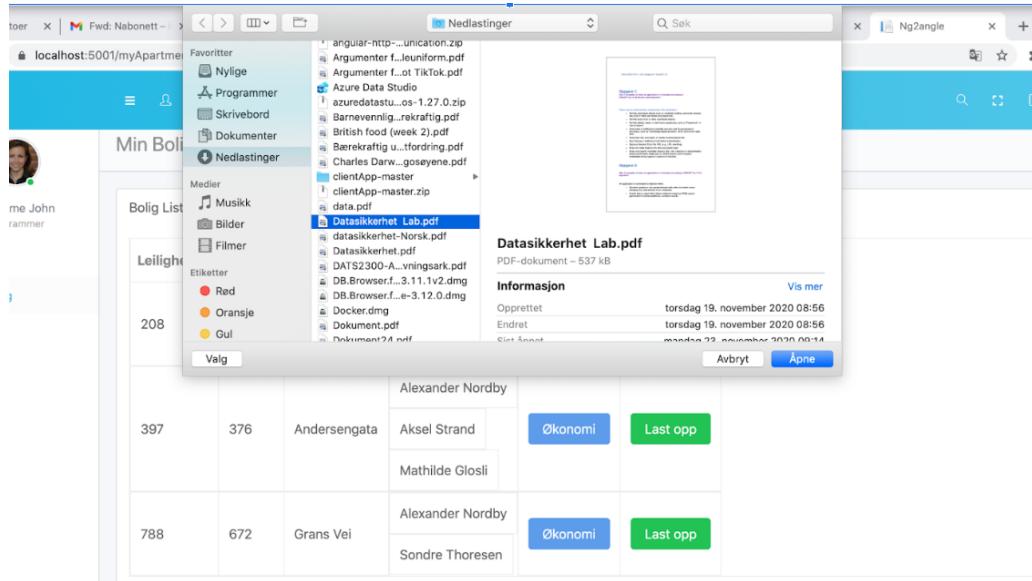
Dette er siste funksjon i beboer komponent. Bruker skal kunne endre på informasjon som gjelder for valgt beboer. I situasjoner hvor det er nødvendig å endre på informasjon til beboer, er det viktig med tilstedeværelsen av en slik funksjon. Bruker trykker på “Endre knapp” som vist i figur 3.11 og endrer deretter ønsket informasjon i “Endre Beboer” modal. Figur 3.13 viser denne funksjonaliteten og “Lagre” knapp for å bekrefte og registrere endringer.



Figur 3.13: Endre Beboer

3.3.3.4 Boligmappe - Laste opp dokumenter

Eiere skal ikke være bundet til en fysisk boligmappe for å holde styr på dokumenter for gjeldende bolig. Denne funksjonen tillater brukere å laste opp alle dokumenter som gjelder for bolig, med enkel og kjapp tilgang. Bruker opplaster gjennom “laste opp” knapp og lagrer i database. Dokumentene vises i en boligmappe tilknyttet bolig. Figur 3.14 viser denne funksjonaliteten.



Figur 3.14: Laste opp funksjon

3.3.3.5 Vis økonomi

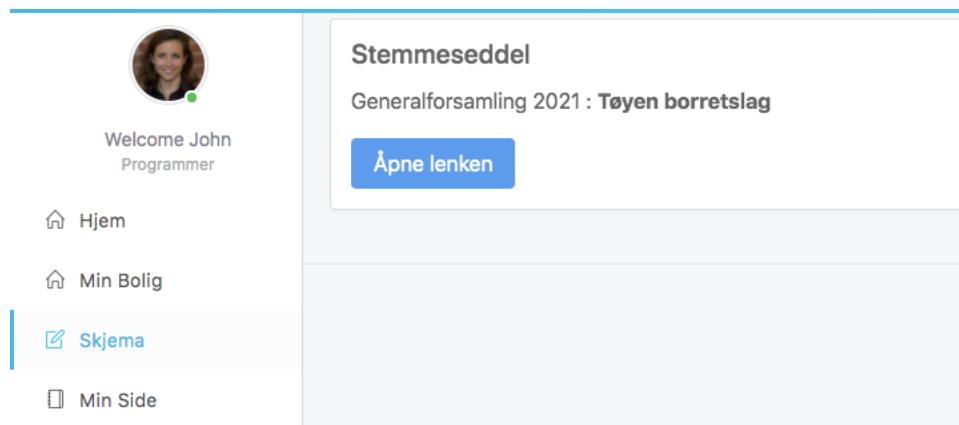
Vis økonomi funksjonen tillater brukere å få tilgang til fakturaer og regnskap for gjeldende bolig. Bruker trykker på “Økonomi” knapp ved valgt bolig, som viser liste av fakturaer for valgt bolig. Figur 3.15 viser fram denne funksjonaliteten.

Beløp	Forfallsdato	Status	Faktura Nr	KID	KontoNummer
993,1	2022-03-18T22:04:13.2181905	Betalt	8407223081564	6728272458933	4144039714166
882,31	2021-08-16T10:35:22.7652007	Avventer	0351664329984	0108202485097	6195552399499
321,44	2021-09-02T02:40:33.2928627	Betalt	2421638338644	2016535356701	2531440196339
357,13	2021-05-20T11:52:47.5504192	Avventer	7774453004177	4095002163811	4399351440624
157,22	2021-11-02T07:43:09.6402508	Betalt	3605384056543	1953637842323	3726784820920
596,15	2022-01-30T14:52:34.6776361	Avventer	2517401795940	7105091590195	0442117970379

Figur 3.15: Vis økonomi funksjon

3.3.4 Skjema

Minimum en gang i året samles beboere av boligselskap for generalforsamling. Dette er møter hvor beboere sammen med styret skal behandle saker omkring blant annet årsregnskapet. Webapplikasjonen har som hensikt å forenkle stemmegivning prosessen under generalforsamlinger. Gjennom en digital stemmegivnings lenke føres brukere til en stemmeseddel hvor dem kan avgi deres stemmer omkring de ulike sakene. Figur 3.16 og 3.17 under viser til denne funksjonaliteten.



Figur 3.16: Skjema lenke

The screenshot shows a survey page. At the top, it lists "Styreleder (2 år)" with two options: "1 Robert Downey Jr." (selected, highlighted in green) and "Colin Farrell". Below this is a section for "Styremedlem (2 år)" with four options: "1 Robert Downey Jr.", "2 David Beckham", "Colin Farrell", and "Christian Bale". At the bottom of the page is a checkbox labeled "Jeg ønsker ikke å stemme på denne saken (blank stemme)". A green button at the very bottom says "Send inn svarene dine".

Figur 3.17: Stemmeseddel

3.3.5 Min side

Denne siden inneholder personlig informasjon om bruker som Navn, adresse, telefon og epost. Informasjonen blir i første omgang hentet fra database og blir ikke mellomlagret for å sikre informasjonen. Brukeren har muligheten til å endre på felter som ikke er grået ut. Endringer blir lagret ved å trykke på “Oppdater” knapp.

Kontakt Informasjon	
Navn	Elise
Epost	Elise.Bjerke19@hotmail.com
Telefon	83 13 75 87
Adresse	Osloveien 23
Poststed	Lodal

Oppdater

Figur 3.18: Min Side

Readonly attribut

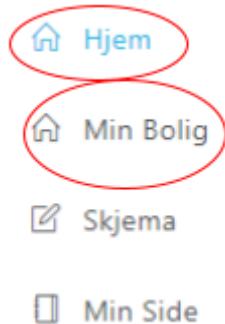
Enkelte av feltene i “Min side” lar seg ikke endre. Dette er fordi database strukturen er avhengig av at attributter som Navn blir hentet fra folkeregisteret og matches med Styret.com sitt originale database. Input feltene dette gjelder for er grået ut og registrerer ikke tastetrykk.

3.3.6 Design og heuristikk

Valg av design og visuelle ikoner har blitt gjort med fokus på å utfylle Nielsens 10 usability heuristikk for bruker design. Målet er at bruker skal ha best mulig brukeropplevelse og ikke måtte overtenke om valg eller feil gjort. (Nielsen, 1994, s.1)

Slette knappen er blitt designet med tradisjonelle farger og mønstre som fokus. De fleste kjenner igjen rødfargen som symboliserer fare eller alvor. Dette hjelper brukere med å forstå og kjenne igjen informasjonen. Valg av ikoner og tekstbeskrivelse er

bestemt med fokus på heuristikk. Brukere skal ha en gjenkjennbar omgivelse ved bruk av universelt gjenkjennelige farger og ikoner. Figur 3.19 viser ikoner som reflekterer betydningen.



Figur 3.19: Ikoner

Heuristikk blir også brukt forebygging av feil og gjenoppretting etter feiltakelse. En bruker som ved uhell trykker på ”Slett” knapp, skal bli presentert for bekreftelse vindu. Utgang skal være tilgjengelig ved slike feil for å opprettholde flere av Nielsens heuristikk. Figur 3.12 viser en slik bekreftelses vindu.

4. Testing Dokumentasjon

4.1 Introduksjon

Testing er en viktig del av utviklingsfasen av et produkt. Denne delen beskriver de forskjellige testing metodene brukt og deres utførelsen. Testing er et av de mest utfordrende delene av utviklingsprosessen. Hovedfokus med testing er å finne feil i applikasjon og fikse dem før leveranse. Fullstendig testing er umulig, målet er derimot testing for å oppnå god nok sikkerhet. Testing bør ikke skape noe problem for utviklingen av applikasjon/systemet. Det er alltid bedre å ha balanse mellom testing og utvikling.

4.1.1 Testing Plan

Som nevnt tidligere blir Nabonett webapplikasjon brukt av kunder i hverdagsliv. Derfor er det viktig å ha en test plan som reflektere dette behovet. Hvis applikasjon ikke fungerer som den skal og ikke utfører de forskjellige funksjonene riktig, vil det trolig føre til stor misnøye hos sluttbruker og ikke minst arbeidsgiver. Denne planen avgjør også hvilke aspekter av systemet som skal testes og hvor stor grad av intensitet hver av dem skal tilføres. Testing planen er basert på ”Fossefallsmodellen” som tilskir at testing er en fase i seg selv etter utvikling av enheter. (Spillner, Linz & Schaefer, 2014, s. 39) Figur 4.1 viser rekkefølgen testing er blitt utført etter.

1. Enhetstesting
2. Integrasjonstesting
3. Systemtesting
4. Akseptansetesting

Figur 4.1: Testing nivåer

Enhetstesting er første nivå av testing utført og tester at hver komponent fullfører sin spesifikasjon individuelt. Den viktigste oppgaven enhetstesting fullfører er å validere om hver enhet utfører sin oppgave som forventet, og hvis ikke fanger opp feilene på riktig måte. Det er viktig for utviklingsprosessen at enhetstesting blir gjort så tidlig som mulig.

Integrasjonstesting utfører tester på individuelle metoder i systemet og tester dem som en kollektiv gruppe. En forutsetning for integrasjonstesting er at enhetstesting er blitt fullført, dette krever også at feil som er blitt funnet er rettet opp. Prosjektet har hatt mer fokus på kontrollere som var brukt under bygging av prosjektet.

Systemtesting er tredje nivå av testing og sjekker at system i helhet utfører krav som forventet. Enhetstesting og integrasjonstesting er ikke nok for å ha en komplett ferdig testet system. De testene blir gjort mot tekniske spesifikasjoner som f.eks en metode som fullfører en spesifikk funksjon. Systemtesting derimot ser på systemet fra sluttbrukerens perspektiv. Mye av funksjonaliteten og resultatet av systemet blir kun oppdaget når systemet i sin helhet blir kjørt. Systemtesting er viktig for å oppdage utfordringer og feil som vanligvis ikke blir oppdaget gjennom andre testing metoder.

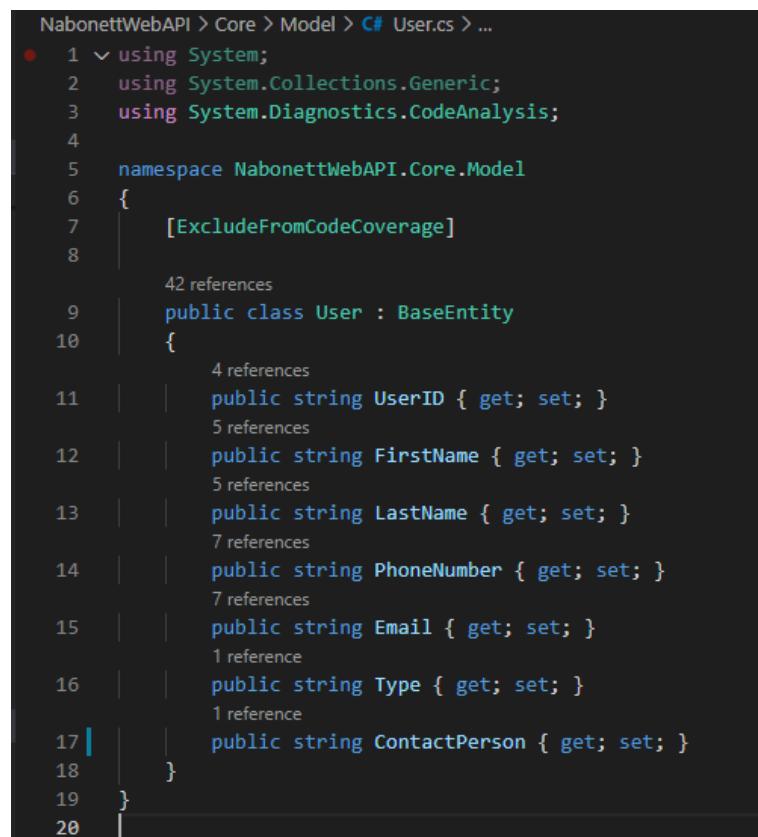
Siste fase av testing er akseptansetesting. Den fasen viser om applikasjon er klar til å bli utgitt for sluttbruker forbruk. I denne fasen vil testere sjekke om applikasjon kan utføre alle spesifiserte funksjoner i form av brukerhistorier. Denne fasen i prosjektet er blitt utført ved hjelp av arbeidsgiver.

4.2 Enhetstesting (Unit Testing)

Enhetstesting er første steg som skal utføres når utviklere arbeider med testingen og er laveste nivå av testing. Enhetstesting gjør sporing av feil, og isolering av feil enkelt som sparar mye tid og penger tidlig i utvikling. Under denne delen av testingen er alle metodene testet linje for linje for å sjekke at metodene er riktig. Det er også viktig å sjekke at metoder kaster riktig unntak når metoder feiler. I dette prosjektet har gruppen testet alle metodene i tjener (backend).

4.2.1 Test objekter

Hovedmålet for enhetstesting er å teste alle metodene. Hovedfokus er på alle mappene i tjener (backend) som inneholder metoder. “Controllers”, “Services” og “Repository” er undermappene som inneholder disse metodene. Figur 4.2 viser et eksempel på en vanlig klasse (User.cs) som inneholder forskjellige attributter. De vanlig klassene var ikke målet for testing fordi dem ikke inneholder metoder.



The screenshot shows a code editor with the file 'User.cs' open. The code is annotated with various numbers indicating references to the class and its properties:

```
NabonettWebAPI > Core > Model > C# User.cs > ...
● 1 using System;
  2 using System.Collections.Generic;
  3 using System.Diagnostics.CodeAnalysis;
  4
  5 namespace NabonettWebAPI.Core.Model
  6 {
  7     [ExcludeFromCodeCoverage]
  8
  9     public class User : BaseEntity
 10    {
 11        42 references
 12        public string UserID { get; set; }
 13        5 references
 14        public string FirstName { get; set; }
 15        5 references
 16        public string LastName { get; set; }
 17        7 references
 18        public string PhoneNumber { get; set; }
 19        7 references
 20        public string Email { get; set; }
 21        1 reference
 22        public string Type { get; set; }
 23        1 reference
 24        public string ContactPerson { get; set; }
 25    }
 26 }
```

Figur 4.2: User.cs

“IUserRepository” inneholder signatur metoder for repository klasse. Disse klassene er brukt til å håndtere dataflyt mellom klient og tjener. Alle metoder i “IUserRepository” er testet og resultat er dokumentert. Metodene i klassen inkluderer henting, oppdatering, opprettning og sletting av “User” objekt fra database. Figur 4.3 viser (IUserRepository.cs).

```
NabonettWebAPI > Core > DataInterfaces > C# IUserRepository.cs > {} NabonettWebAPI.Core.DataInterfaces
  1 < using System;
  2   using System.Collections.Generic;
  3   using System.Threading.Tasks;
  4   using NabonettWebAPI.Core.Model;
  5   using NabonettWebAPI.Data;
  6
  7 < namespace NabonettWebAPI.Core.DataInterfaces
  8   [
    4 references
  9     < public interface IUserRepository
10       {
11         [
          1 reference
12           < Task<IReadOnlyCollection<User>> GetAllAsync();
13           [
              1 reference
14             < Task<User> GetByIDAsync(string ID);
15             [
                  1 reference
16               < Task UpdateAsync(User user);
17               [
                  1 reference
18                 < Task CreateAsync(User user);
19                 [
                      1 reference
                    ]
                  ]
                ]
              ]
            ]
          ]
        }
      ]
    ]
  ]
}
```

Figur 4.3: IUserRepository.cs

Kontroller er brukt til å håndtere API kall fra klient. Klassen sender videre informasjon til service klassen som kommuniserer med database. Alle metoder i kontroller er testet for å sjekke om logikk fungerer som det skal. Figur 4.4 viser et eksempel på kontroller (UserController.cs).

```
NabonettWebAPI > Controllers > UserController.cs > {} NabonettWebAPI.Controllers > NabonettWebAPI.Controllers.UserController
  1  using System;
  2  using System.Collections.Generic;
  3  using System.Threading.Tasks;
  4  using Microsoft.AspNetCore.Mvc;
  5  using Microsoft.Extensions.Logging;
  6  using NabonettWebAPI.Core;
  7  using NabonettWebAPI.Core.DataInterfaces;
  8  using NabonettWebAPI.Core.Model;
  9
10  namespace NabonettWebAPI.Controllers
11  {
12
13      [ApiController]
14      [Route("api/[controller]")]
15  public class UserController : Controller
16  {
17      private readonly IUserService userService;
18
19
20      public UserController(ILogger<UserController> logger, IUserService userService)
21      {
22          this.userService = userService;
23      }
24
25
26      [HttpGet]
27      public async Task<IReadOnlyCollection<User>> GetAllAsync()
28      {
29
30          var result = await userService.GetAllAsync();
31
32          return result;
33      }
34
35
36      [HttpGet("{id}")]
37      public async Task<User> GetByIdAsync(string ID)
38      {
39
40          var result = await userService.GetByIdAsync(ID);
41
42          return result;
43      }
44  }
```

```
45
46
47     [HttpDelete("{id}")]
48
49     0 references
50     public async Task delete(string id)
51     {
52
53         if (id == null) { return; }
54
55         await userService.DeleteAsyn(id);
56
57     }
58
59
60     [HttpPost]
61     0 references
62     public async Task CreateAsync([FromBody] User user)
63     {
64
65         await userService.CreateAsync(user);
66     }
67
68
69
70
71     [HttpPut]
72     0 references
73     public async Task UpdateContactAsync([FromBody] User user)
74     {
75
76         //if (contact.SSN == null) { return; }
77
78         await userService.UpdateAsyn(user);
79
80
81
82
83
84     }
85 }
86 }
```

Figur 4.4: UserController.cs

Repository klassene kobler Service klassene til database. Metodene sender kall til database og verdiene er returnert til API kallene. Det er viktig å teste disse klassene for å sikre at riktig verdier er returnert og SQL kommandoene henter de verdiene som er etterlyst avhengig av hvilken API metode kall bruk. Alle metoder har passert testene. Figur 4.5 viser et eksempel på Repository (HousingRepository klassen) som inneholder metoder for “Housing”. Metodene testet i unit testing.

```
public async Task<IReadOnlyCollection<Housing>> GetAllsync()
{
    return await Context.Housings
        .Include(h => h.Owners)
        .ThenInclude(c => c.Email)

        .Include(h => h.Owners)
        .ThenInclude(c => c.PhoneNumber)
        .Include(h => h.Economys)

        .Where(h => !h.IsDeleted)
    .ToListAsync();
}

public async Task<Housing> GetByIdAsync(string id)
{
    return await Context.Housings
    .Where(h => !h.IsDeleted)
    .SingleOrDefaultAsync(c => c.HouseId.Equals(id));
}

public async Task CreateHousingAsync(Housing housing)
{
    await Context.Housings.AddAsync(housing);
    await Context.SaveChangesAsync();
}

public async Task UpdateHousingAsync(Housing updatedHousing)
{
    Context.Housings.Update(updatedHousing);
    await Context.SaveChangesAsync();
}

public async Task DeleteAsync(string id)
{
    var housing = await Context.Housings.
    SingleOrDefaultAsync(c => c.HouseId == id);

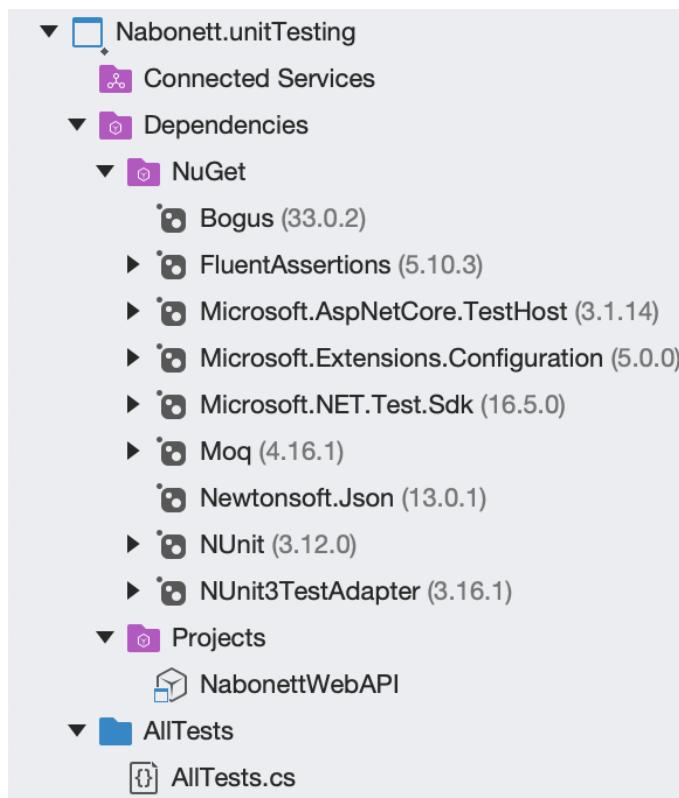
    housing.IsDeleted = true;

    Context.Housings.Update(housing);
    await Context.SaveChangesAsync();
}
```

Figur 4.5: “HousingRepository” metoder

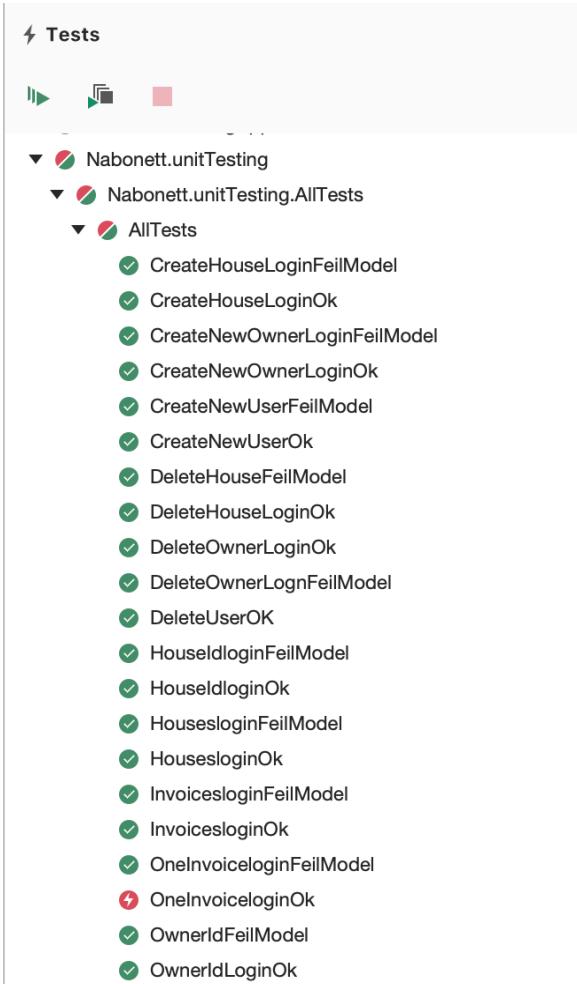
4.2.2 Enhetstesting verktøy

Enhetstestings verktøy brukt i dette prosjektet er NUnit. Dette er et mye brukte rammeverk for enhetstesting for alle .Net-språk. Det er et åpen kildekode verktøy som tillater å skrive manuelle tester. Den støtter tester som kan kjøres parallelt. Enhetstesting i dette prosjektet ble utført og satt opp av samme utvikler som hadde ansvar for Nabonett API kode. Figur 4.6 viser bilde av verktøy inne i prosjektet.



Figur 4.6: Nunit verktøy

Figur 4.7 viser “OneInvoiceLoginOk” feiler når testene var kjørt i denne fasen. Feilen var pga mislykket ID. Figur 4.8 viser vellykket kjøring av alle tester.



Figur 4.7: Kjøring med feil



Figur 4.8: Vellykket kjøring

```

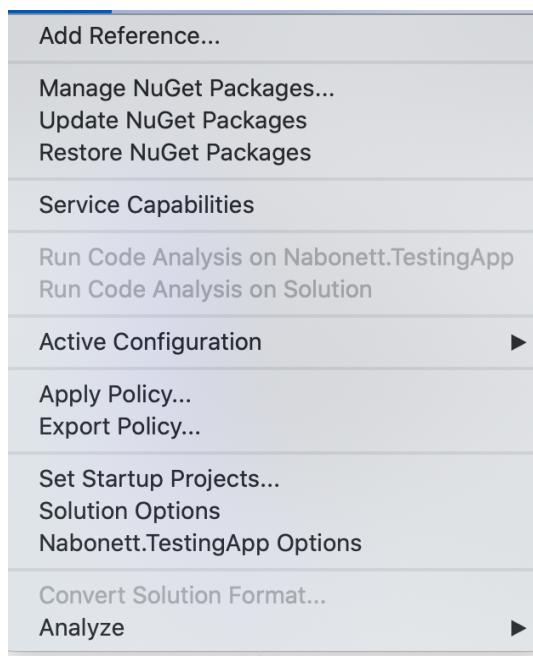
    ✓ NabonettWebAPI.Nabonett.unitTesting.Nabonett.unitTesting.AllTests.AllTests.ShouldDeleteUserAsync
    ✓ NabonettWebAPI.Nabonett.unitTesting.Nabonett.unitTesting.AllTests.AllTests.UserIdFeilModel
    ✓ NabonettWebAPI.Nabonett.unitTesting.Nabonett.unitTesting.AllTests.AllTests.UserIdloginOk
    ✓ NabonettWebAPI.Nabonett.unitTesting.Nabonett.unitTesting.AllTests.AllTests.UsersFeilModel
    ✓ NabonettWebAPI.Nabonett.unitTesting.Nabonett.unitTesting.AllTests.AllTests.UsersloginOk

Passed: 34 Failed: 0 Errors: 0 Inconclusive: 0 Invalid: 0 Ignored: 0 Skipped: 0 Time: 00:00:09.3685258
  
```

Figur 4.9: Tid for kjøring

4.2.3 Konklusjon og refleksjoner til enhetstesting

Feil funnet i enhetstesting er fikset i utviklingsfasen. Alle testene i enhetstesting er utført og gir 100% bestått resultat. Det hadde vært fint å vise Kodedekning (Code coverage) av enhetstesting, men gruppen hadde ikke tilgang til “code analysis” på Visual Studio.



Figur 4.10: Manglende Code coverage i VS 19

Visual Studio Enterprise er nødvendig for visning av kode dekning. Dette er en dyr kostnad for gruppen å betale selv, og en begrensning oppdaget i Visual studio 19 sent i utviklingsprosessen. Alle metoder og funksjoner i prosjekt er testet. Gruppen anser enhetstesting er viktig for å teste metoder tidlig i utviklingsprosessen for å spare tid og forhindre feil i andre deler av testing fasen.

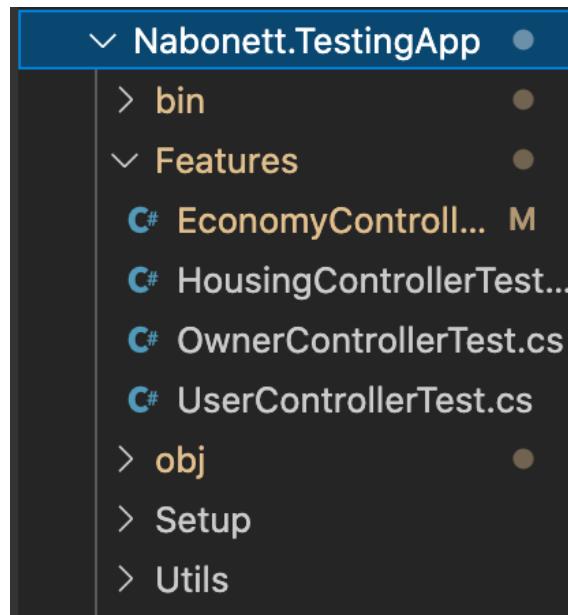
4.3 Integrasjonstesting(Integration Testing)

Integrasjonstesting er definert som en type testing der programvare moduler er integrert logisk og testet som en gruppe. Et typisk programvare prosjekt består av flere programvare moduler, kodet av forskjellige utviklere. Hensikten med integrasjonstesting er å sjekke at komponenter fungerer riktig sammen.

4.3.1 Test objekter

Hovedmålet med integrasjonstesting er å teste “Controller” mappe fordi den inneholder kontroller som styrer interaksjonen mellom komponenter. Målet er også å teste “Controllers” som en gruppe. Alle kontrollene i Nabonett Web Api ver testet.

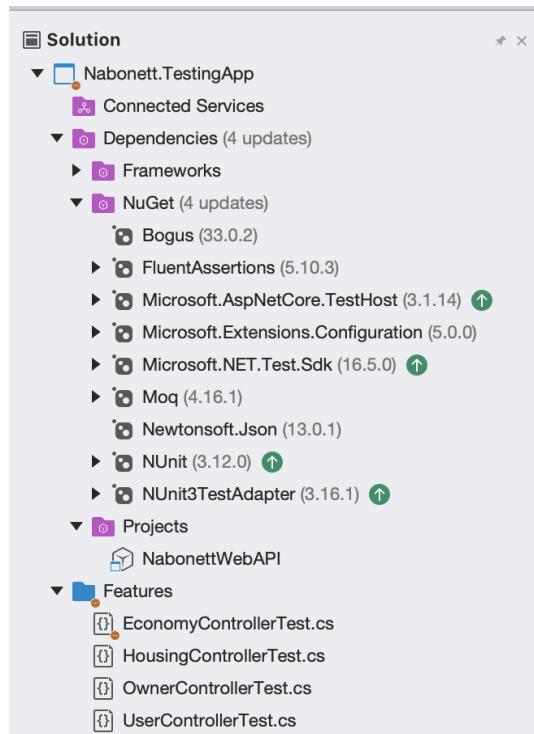
Figur 4.12 viser kontrollere testet i denne fasen.



Figur 4.12: Kontrollere

4.3.2 Integrasjonstesting verktøy (Integration testing)

I dette prosjektet har utviklere brukt NUnit testing tools. Figur 4.13 viser et bilde av Nunit i prosjekt. Som i enhetstesting er det samme person som utfører integrasjonstesting for å ha så enkel overgang og oversikt som mulig.



Figur 4.13: Nunit verktøy (integrasjonstesting)

Fokus på integrasjonstesting er forskjellige metoder og klasser håndterer data sammen. Prosjektet bruker mange tabeller hvor data fra forskjellige tabeller er brukt til å sette opp sammensatt SQL spørninger til database. Integrasjonstesting er brukt til å sikre at data hentet er samme som metode skulle hente for innkommende API kall fra klient. Alle metoder i alle klasser har passert tester. Figur 4.14 viser et eksempel av “HousingControllerTest” satt opp.

```
NabonettWebAPI > Nabonett.TestingApp > Features > C# HousingControllerTest.cs
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using FluentAssertions;
6  using Nabonett.TestingApp.Setup;
7  using Nabonett.TestingApp.Utils.Extensions;
8  using NabonettWebAPI.Core.Model;
9  using NabonettWebAPI.DTOs;
10 using NabonettWebAPI.TestBuilders;
11 using NUnit.Framework;
12
13 namespace Nabonett.TestingApp.Features
14 {
15     public class HousingControllerTest : TestFixture
16     {
17
18         [Test]
19         public async Task GetAllHouses()
20         {
21             var response = await Client.GetAsync("api/housing/");
22             var result = await response.EnsureSuccessAndReturnResponseAsync<List<Email>>();
23             result.Count.Should().Be(100);
24         }
25
26
27         [Test]
28         public void HouseWithOwner()
29         {
30             var housing = HousingBuilder.CreateRandomized().Generate(10);
31             var test = OwnerBuilder.CreateRandomized().Generate(10);
32             housing.Select(c => c.Owners).Should().OnlyHaveUniqueItems();
33         }
34
35         [Test]
36         public async Task GetSpecificHouse()
37         {
38
39             var id = "01015508422"
40
41             var response = await Client.GetAsync($""/id/{id}");
42             var result = await response.EnsureSuccessAndReturnResponseAsync<HousingContactDTO>();
43             result.Should().ToString();
44         }
45     }
46 }
```

Figur 4.14: “HousingControllerTest”

4.3.3 Manuell integrasjonstesting av Postman

I denne fasen av testing er Postman brukt for manuell integrasjonstesting for å sjekke testverdier i tjener (backend). Dette er veldig nyttig for testing av verdier. Postman i denne fasen er brukt for å teste og modifisere verdier manuelt. Postman tillater utviklere å lage enkle og komplekse HTTP/s kall. Postman viser returnmeldinger og returnerte HTTP status kode. Figur 4.15 viser et eksempel av test verdier fra "<https://localhost:4201/api/owner>" i Postman og returnerte statuskode (200 OK).

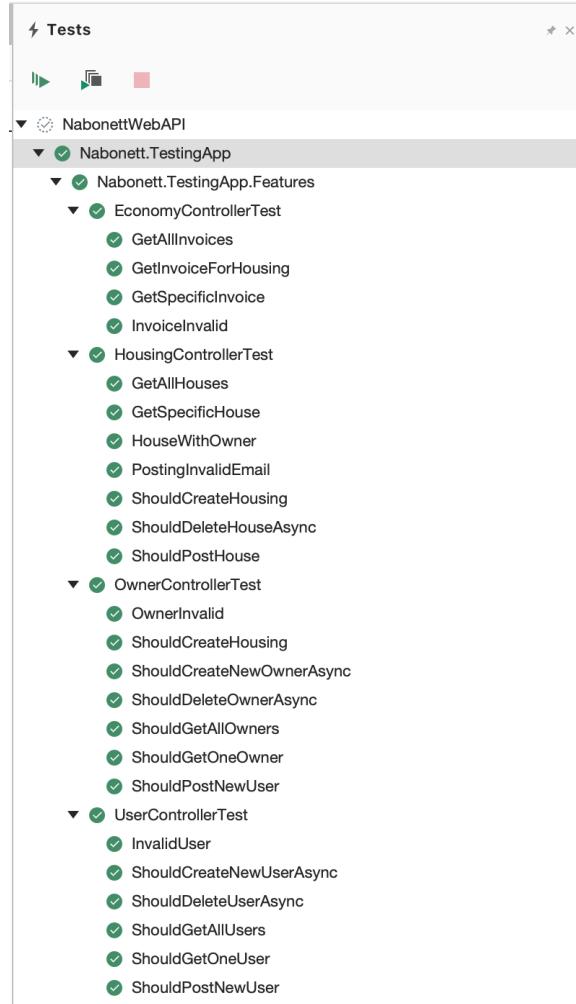
The screenshot shows the Postman interface with the following details:

- Method:** POST
- URL:** https://localhost:4201/api/owner
- Body:** JSON (selected tab)
- Request Body Content:**

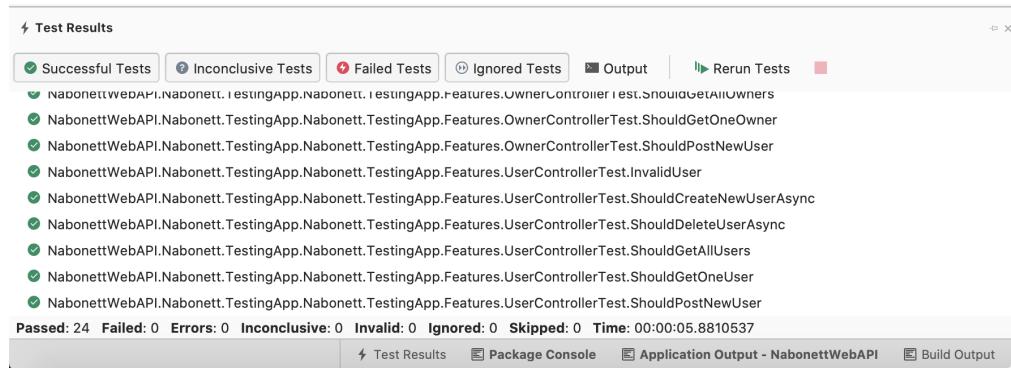
```
1
2   {
3     "ssn": "01015512039",
4     "firstName": "Mehboob",
5     "lastName": "Kalu",
6     "address": "0345",
7     "city": "Oslo",
8     "email": "Tiril_Aalerud55@hotmail.com",
9     "phoneNumber": "752 41 296"
10    }
```
- Response Status:** 200 OK
- Response Time:** 148 ms
- Response Size:** 15.74 KB
- Response Body Content:**

```
1 [
2   {
3     "ssn": "01015512039",
4     "firstName": "Tiril",
5     "lastName": "Aalerud",
6     "address": "0056",
7     "city": "Lomark",
8     "email": "Tiril_Aalerud55@hotmail.com",
9     "phoneNumber": "752 41 296"
10    },
11    {}
```

Figur 4.15: Postman test



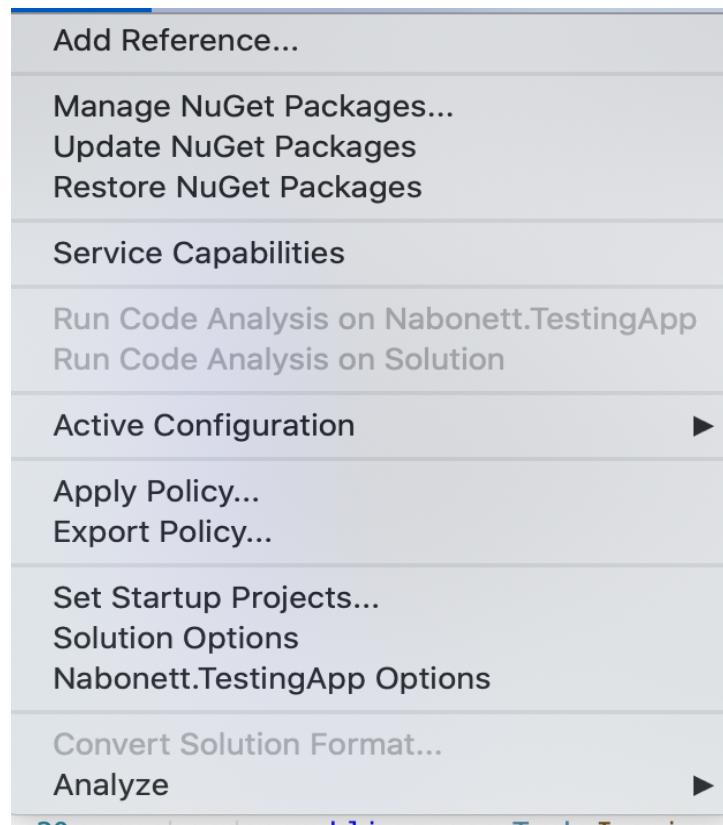
Figur 4.16: Godkjent kjøring (Integrasjonstest.)



Figur 4.17: Tid for kjøring (Integrasjonstest)

4.3.3 Konklusjon og refleksjoner til integrasjonstesting

Alle tester i integrasjonstesting er utført og gir 100% bestått resultat og viser at hele API ble godt testet. Alle komponentene fungerer godt sammen. Det hadde vært fint å vise Kodedekning (Code coverage) av testingen, men gruppen har ikke tilgang til kodeanalyse i Visual Studio.



Figur 4.18: Manglende code coverage (Integrasjonstesting)

4.4 Akseptansetesting

Denne seksjonen tar for seg akseptansetest av systemet. Denne testen er blitt utført i samarbeid med arbeidsgiver og skal redegjøre for om systemet er klart for bruk i et kunde miljø. Testen blir brukt for å fastslå om webapplikasjonen har fullført de kriteriene som er blitt bearbeidet sammen arbeidsgiver. Kriteriene er basert på kravspesifikasjonene og er skrevet ned som bruker historier for å teste. Resultat av akseptansetest er blitt innført i en tabell.

4.4.1 Test objekter

Denne testen fullføres ved hjelp av brukerhistorier for å dokumentere bruken av systemet gjennom noen andres synspunkt, utenom utviklere. Arbeidsgiver har utført denne testen.

4.4.2 Resultat av akseptansetest

Brukerhisto rie:	Akseptansekri terier:	Test steg:	Result at:	Kommentar:
1.Som bruker vil logge inn i min Nabonett konto.	Hvis bruker har konto, er det mulig å logge inn i logg-inn siden.	1. Gå til https://nabonett.azurewebsites.net/ 2. Skriv inn konto info. 3. Trykk på logg inn knapp.	Bestått	
2. Som bruker vil jeg tilbakestille passord når glemt.	Hvis bruker har konto epost, er det mulig å resette passord.	1. Gå til https://nabonett.azurewebsites.net/ 2. Trykk på «Glemt passord?» 3. Utfør stegene. 4. Sett ny passord.	Bestått	

3. Som bruker vil jeg velge bolig i hjemmesiden.	Hvis bruker er logget inn i Nabonett, og nавигerer til hjemmeside. Vil det være mulig å velge bolig.	1. Gå til https://nabonett.azurewebsites.net/home 2. Velg bolig.	Bestått	
4. Som bruker vil jeg se oversikt over boliger jeg eier.	Hvis bruker er logget inn i Nabonett og nавигerer til "Min bolig". Vil det være mulig å ha oversikt over boliger.	1. Gå til https://nabonett.azurewebsites.net/myApartment	Bestått	
5. Som bruker vil jeg se økonomi til valgt bolig.	Hvis bruker er logget inn i Nabonett og nавигerer til "Min bolig". Vil det være mulig å sjekke økonomi for valgt bolig.	1. Gå til https://nabonett.azurewebsites.net/myApartment 2. Trykk "Økonomi" knapp. 3. Sjekk at økonomi liste er tilstede.	Bestått	Brukere har kun mulighet til å sjekke økonomi for en enkelt bolig om gangen.
6. Som bruker vil jeg laste opp dokumenter til valgt bolig.	Hvis bruker er logget inn i Nabonett og nавигerer til "Min bolig". Vil det være mulig å laste opp dokument.	1. Gå til https://nabonett.azurewebsites.net/myApartment 2. Trykk "Laste Opp" knapp. 3. Sjekk filmappen	Bestått	

7. Som bruker vil jeg legge til beboer i en bolig.	Hvis bruker er logget inn i Nabonett og nавигerer til "Min bolig". Vil det være mulig å legge til en beboer.	<ol style="list-style-type: none"> 1. Gå til https://nabonett.azurewebsites.net/myApartment 2. Trykk på "Legg til beboer" knappen. 3. Skriv inn Beboer info og lagre. 4. Sjekk om beboer er blitt lagt til. 	Bestått	
8. Som bruker vil jeg slette en Beboer fra en bolig.	Hvis bruker er logget inn i Nabonett og nавигerer til "Min bolig". Vil det være mulig å slette en Beboer.	<ol style="list-style-type: none"> 1. Gå til https://nabonett.azurewebsites.net/myApartment 2. Trykk på "Slett" knappen. 3. Sjekk om Beboer er blitt slettet. 	Bestått	
9. Som bruker vil jeg endre på informasjonen til en Beboer.	Hvis bruker er logget inn i Nabonett og nавигerer til "Min bolig". Vil det være mulig å endre på informasjonen til en Beboer.	<ol style="list-style-type: none"> 1. Gå til https://nabonett.azurewebsites.net/myApartment 2. Trykk på "Endre" knappen. 3. Endre på Beboer info og lagre. 4. Sjekk om Beboerinfo er blitt endret. 	Bestått	

10. Som bruker vil jeg ha oversikt over beboere i valgt bolig.	Hvis bruker er logget inn i Nabonett og nавигerer til "Min bolig". Vil det være mulig å se en liste av beboere.	<ol style="list-style-type: none"> 1. Gå til https://nabonett.azurewebsites.net/myApartment 2. Se liste av beboere. 	Bestått	
11. Som bruker vil jeg kunne stemme i avstemning..	Hvis bruker er logget inn i Nabonett og nавигerer til "Min skjema". Vil det være mulig å utføre en avstemning.	<ol style="list-style-type: none"> 1. Gå til https://nabonett.azurewebsites.net/survey 2. Trykk på "Åpne lenken" knapp. 4. Utfør avstemning og lagre. 	Bestått	Dette medfører til at bruker blir sendt til en stemmeseddel som inneholder informasjon omkring avstemningen.
12. Som bruker vil jeg endre på min egen konto informasjon.	Hvis bruker er logget inn i Nabonett og nавигerer til "Min side". Vil det være mulig å endre på konto informasjon.	<ol style="list-style-type: none"> 1. Gå til https://nabonett.azurewebsites.net/contactinformation 2. Endre på feltene som gjelder. 3. Trykk på "Oppdater" for å lagre. 4. Sjekk at endringer er blitt gjort. 	Bestått	Bruker kan kun endre på spesifikke felter.
13. Som bruker vil jeg logge ut av Nabonett.	Hvis bruker er logget inn i Nabonett og trykker på "Logg ut" knapp vil man bli logget ut.	<ol style="list-style-type: none"> 1. Gå til https://nabonett.azurewebsites.net/home 2. Trykk på "Logg ut" knapp. 3. Sjekk at man blir overført til "logg inn" side. 	Bestått	

4.4.3 Konklusjon og møte med klient

Dette møtet med klient gikk ut på å diskutere resultatet av akseptansetesten. Klienten er fornøyd med testen og produktet og anser det som et godt realisert prosjekt. Selve akseptansetesten konkluderte med 100% bestått resultat.

4.5 Utlassering på Azure

Gruppen har ikke hatt direkte tilgang til Azure skytjenester. Styret.com har på vegne av gruppen utplassert en versjon av Nabonett på Azure. Utlassering har ikke vært et stort fokus, men heller stabilisering og sikkerhet av webapplikasjon.

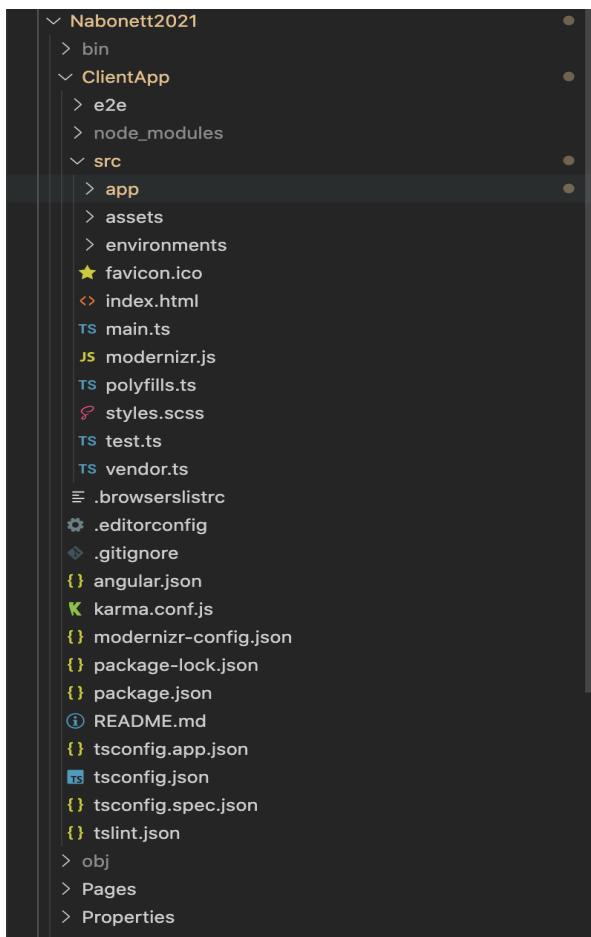
5. Teknisk dokumentasjon

5.1 Front-end struktur

Neste avsnittet inneholder “klient” (front-end) mappe arkitektur av Nabonett webapplikasjon. Angular som rammeverk har et mappestruktur som kommer med prosjekt grunnleggelse. Vi har brukt denne arkitekturen til å bygge videre på vår webapplikasjon. Mesteparten av struktur er plassert i rotmappen “Client-App” og fokuset er på innhold i undermapper “Node Modules” og “src”.

Node Modules

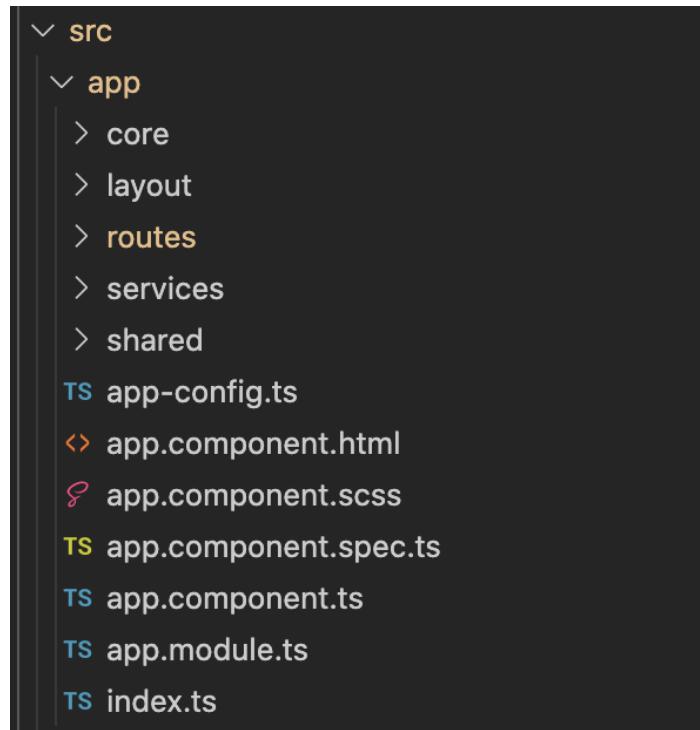
Denne mappen inneholder biblioteker for applikasjonen. Tredjeparts biblioteker som applikasjonen er avhengig av finnes i denne mappen. Node modules er bare med i utviklings versjon av applikasjonen.



Figur 5.1: Front-end struktur

5.1.1 Kilde mappe (src)

src-mappen fungerer som et utgangspunkt i applikasjonen. Det er der kompilatoren ser etter filer for å kompilere applikasjonen. Mappen inneholder ikke kode, men er nødvendig å ha i den opprinnelige strukturen til prosjektet. Som nevnt ovenfor er mappen nødvendig, ellers vil ikke prosjektet kompilere. Mappen fungerer også som en kontainer for hovedprosjektet. Denne mappen inneholder undermapper som "App" og "Assets". Hver mappe har beskrivelse på sin struktur og innhold beskrevet under i detalj.



Figur 5.2: src/app mappestruktur

Assets

“Assets” mappen ble brukt mer i startfasen av prosjektet, før backend var klar. Utviklerne lagret falske dummy verdier og ressurser for å lage den opprinnelige prototypen til applikasjonen i den mappen. I de senere stadiene av prosjektet har utviklerne hentet data verdier fra server, og mye av innholdet i denne mappen er dermed kastet og utfaset. Denne mappen inneholder alle statiske elementer brukt i Nabonett. Bilder, ikoner og annet visuelle elementer brukt i webapplikasjon er baserte i denne mappen.

App

Mappen inneholder "app-config.ts" brukt til å konfigurere påloggingsprosessen ved hjelp av “Azure AD B2C”. Filen "app-module.ts" er øverste modul som kompileres for å konfigurere alle andre moduler i applikasjonen. Filen starter også hele applikasjonen. Den kjører også konfigurasjonen av autorisasjon prosessene. "App-component.ts" har mange metoder for å kontrollere pålogging, avlogging, tilbakestilling av passord osv. Undermapper som former deler av prosjektstruktur, er som følgende “Core”, “Layout”, “Routes”, “Services” detaljert nedenfor.

Core

Mappen inneholder mappene “menu” og “themes” mapper. “Menu” mappen inneholder setup for meny i applikasjonen. Her kan utviklere legge til ny meny i navigasjonen. Themes mappen inneholder kode brukt for å endre på farge i øverste navbar.

Layout

Denne mappen har prosjektets oppsett for GUI i applikasjonen. Layout mappen setter skjelettet til prosjektet når det gjelder brukervennlighet og struktur. Header styrer den øverste navigasjonslinjen som inneholder funksjoner som avlogging, ved å velge mobil- eller skrivebordsversjon. Sidebar styrer hovednavigasjonen til funksjonalitetene som liste av eiendom i “Min Bolig”, “Skjema” og “Minside”. Footer brukes til å vise ønsket bunntekst for det aktuelle vinduet. Dette ble ikke brukt mye.

Routes

Routes modul er grunnleggende mappe i applikasjonens logikk. Den inneholder komponenter, routing-moduler og modell-klasser. Ettersom denne mappen inneholder mesteparten av front-end kode funksjonalitet blir det gjort om til sin egen seksjon for å gi en bedre beskrivelse av innholdet.

Routes.ts

Denne filen styrer navigasjonen i applikasjonsgrensesnittet. Det gjengir komponenter avhengig av hvilken komponent brukeren klikker fra Sidemeny/sidebar). Hvis brukeren klikker hjem, blir hjemme komponenten gjengitt ved hjelp av banen til det klikkede elementet. På denne måten kunne vi interaktivt bruke sidemeny for navigering, noe som gjør applikasjonen veldig brukervennlig.

Menu.ts-filen ble brukt til å angi meny objekter som skal brukes i routes.ts. Banen til hvert element av objektet angis i filen. Dette gjør det enkelt å kontrollere hvordan navigasjonen reagerer på hvert klikk.

Routes mappen har også alle komponentene og modell filene som ble brukt i applikasjonen. Komponentene ble strukturert i et borte som gjør at prosjektoversikten er lett forståelig og kan skaleres med minimal kompleksitet. Hver komponent har en bestemt funksjonalitet som utfører en bestemt bruk i applikasjonen. når en komponent er stor, blir den delt inn i underkomponenter. For eksempel hadde “Min Bolig” mange funksjoner som var nødvendige for å gjengi en gang, men noen av funksjonalitetene som økonomi har sin egen komponent.

3.5.1 Klient (Front-end) Validering

Validering av front-end er utført blant annet i “logg-inn”, “Min side” og flere steder i Nabonett. Data er hentet fra tjener for det meste, men i tilfeller der kunde sender verdier til tjener f.eks “Endre kontaktinformasjon”. Blir verdier testet gjennom form-validering. Angular reactive forms er brukt til å håndtere validering av data, og hjelper med å unngå lagring av ugyldige verdier og tomme verdier. Figurene under viser de ulike punktene nødvendig for form-validering

```

Nabonett2021 > ClientApp > src > app > routes > contactinformation > contactinformation > contactinformation.component.ts > ContactinformationComponent
  1 ~ import { HttpClient } from '@angular/common/http';
  2 ~ import { Component, OnInit } from '@angular/core';
  3 ~ import { HttpService } from 'src/app/services/http.service';
  4 import { Contactinformation } from '../contactinformation';
  5 import { FormBuilder, FormGroup, Validators } from '@angular/forms';
  6 import { ActivatedRoute, Router } from '@angular/router';
  7 import { PhoneNumber } from '../../telephone';
  8
  9
10 ~ @Component({
11   selector: 'app-contactinformation',
12   templateUrl: './contactinformation.component.html',
13   styleUrls: ['./contactinformation.component.scss']
14 })
15 export class ContactinformationComponent implements OnInit {
16   form : FormGroup;
17
18   userDetails: Contactinformation;
19   // contactURL: string = "https://localhost:4201/api/owner/";
20
21
22 ~ validation = {
23   ssn: ['02089200760'],
24   firstName: [
25     null, Validators.compose([Validators.required, Validators.pattern("[a-zA-ZæøåØÅ. \-]{2,30}")])
26   ],
27   address: [
28     null, Validators.compose([Validators.required, Validators.pattern("[a-zA-ZæøåØÅ. \-]{2,30}")])
29   ],
30   city: [
31     null, Validators.compose([Validators.required, Validators.pattern("[a-zA-ZæøåØÅ. \-]{2,30}")])
32   ],
33   number: [
34     null, Validators.compose([Validators.required, Validators.pattern("[0-9]{4,8}")])
35   ],
36   email: [
37     null, Validators.compose([Validators.required, Validators.pattern(`^${[A-ZÆØÅa-zæøåØ-9._%+-]+@[A-Za-z0-9.-]+.[A-Za-z]{2,16}$`)])
38   ]
39 }
40
41
42
43
44 constructor(private http: HttpClient, private fb: FormBuilder,
45             | | | | private route: ActivatedRoute, private router: Router) {
46   this.form = fb.group(this.validation);
47 }

```

Figur 5.3: Form-validering i “contactinformation.component.ts”

```

37 <div class="form-group row">
38   <label class="text-bold col-xl-2 col-md-3 col-4 col-form-label text-right" for="firstName" aria-readonly="">Navn</label>
39   <div class="col-xl-10 col-md-9 col-8">
40     <input class="form-control" formControlName="firstName" type="text" placeholder="" readonly />
41     <p class="alert alert-warning" [hidden]="form.controls.firstName.valid || (form.controls.firstName.pristine )">
42       Navn er obligatorisk.</p>
43   </div>
44 </div>
45 <div class="form-group row">
46   <label class="text-bold col-xl-2 col-md-3 col-4 col-form-label text-right" for="email">Epost</label>
47   <div class="col-xl-10 col-md-9 col-8">
48     <input class="form-control" formControlName="email" type="email" />
49     <p class="alert alert-warning" [hidden]="form.controls.email.valid || (form.controls.email.pristine )">
50       Epost er obligatorisk, og må være slik eksempel@yahoo.com.</p>
51   </div>
52 </div>
53 <div class="form-group row">
54   <label class="text-bold col-xl-2 col-md-3 col-4 col-form-label text-right" for="number">Telefon</label>
55   <div class="col-xl-10 col-md-9 col-8">
56     <input class="form-control" formControlName="number" type="text" />
57     <p class="alert alert-warning" [hidden]="form.controls.number.valid || (form.controls.number.pristine )">
58       Telefon er obligatorisk.</p>
59   </div>
60 </div>
61 <div class="form-group row">
62   <label class="text-bold col-xl-2 col-md-3 col-4 col-form-label text-right" for="address">Adresse</label>
63   <div class="col-xl-10 col-md-9 col-8">
64     <input class="form-control" formControlName="address" type="text" />
65     <p class="alert alert-warning" [hidden]="form.controls.address.valid || (form.controls.address.pristine )">
66       Address er obligatorisk.</p>
67   </div>
68 </div>
69 <div class="form-group row">
70   <label class="text-bold col-xl-2 col-md-3 col-4 col-form-label text-right" for="city">Posted</label>
71   <div class="col-xl-10 col-md-9 col-8">
72     <input class="form-control" formControlName="city" type="text" placeholder="Ingen Poststed" readonly />
73     <p class="alert alert-warning" [hidden]="form.controls.city.valid || (form.controls.city.pristine )">
74       City er obligatorisk.</p>
75   </div>
76 </div>
77 </div>
78 </div>
79 </div>
80 </div>
81 </div>
82 </div>

```

Figur 5.4: Form-validering i HTML

The screenshot shows a web application interface for entering contact information. On the left, there is a vertical sidebar with icons for home, edit, and file operations. The main area has a title "Kontakt Informasjon". The form consists of five fields:

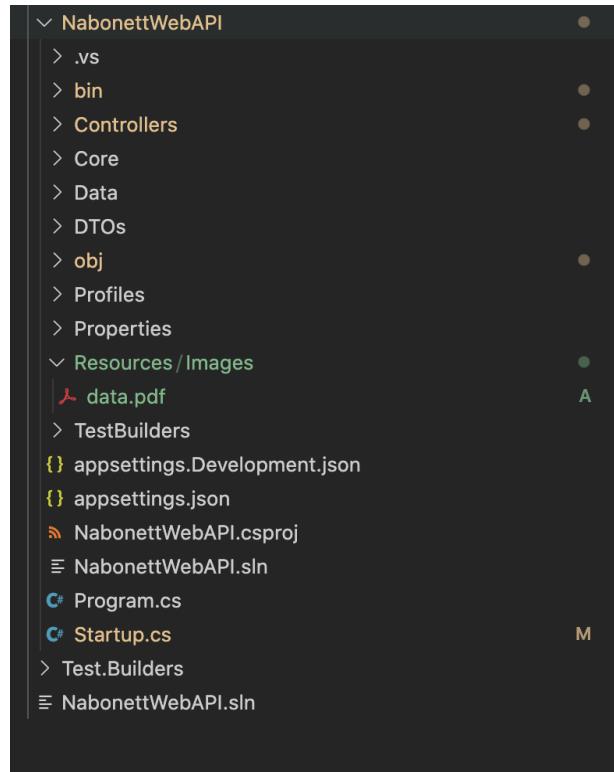
- Navn:** Input field containing "Mina".
- Epost:** Input field containing "@gmail.com". Below it, an orange validation message box says: "Epost er obligatorisk, og må være slik eksample@yahoo.com."
- Telefon:** Input field containing "701 30". Below it, an orange validation message box says: "Telefon er obligatorisk."
- Adresse:** Input field is empty. Below it, an orange validation message box says: "Address er obligatorisk."
- Posted:** Input field containing "Storvåg".

At the bottom left is a blue "Oppdater" button.

Figur 5.5: Visning av validering på nettleser

5.2 Back-end struktur

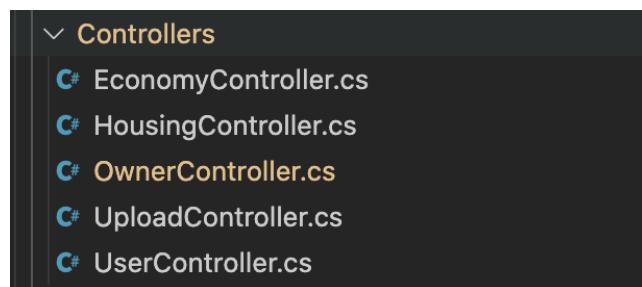
Denne seksjonen beskriver oppbygging av back-end strukturen. Dette består av seks hovedmapper og deres innhold som vil bli videre forklart i seksjonene under.



Figur 5.6:Back-end struktur

5.2.1 Kontroller

En kontroller(controller) gir API til klient (frontend) og styrer dataflyten mellom klient og tjener. Kontrolleren håndterer forespørsler fra klientsiden og returnerer data i JSON format til klienten via HTTP / HTTPS protokoller. Følgende figur 5.7 er bilde av kontroller i “Controllers” mappe, og figur 5.8 beskriver funksjonalitet i detalj.



Figur 5.7: Kontroller mappe

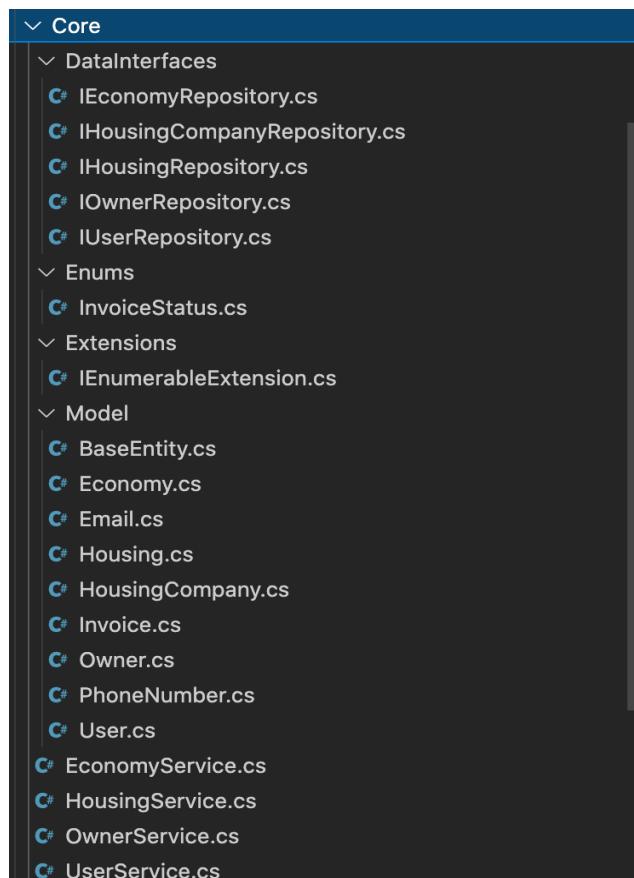
Kontroller	Beskrivelse	Funksjoner
EconomyController	Kontroller som gir API for økonomi delen av Nabonett	<ul style="list-style-type: none"> - Gir informasjon om faktura, utløpsdato og status for regninger. - Gir informasjon om hver faktura knyttet til spesifikk bolig. - En bolig kan ha en eller mange fakturaer og hver av dem er knyttet til eier og bolig gjennom unik ID.
HousingController	Kontroller som gir API for selve bolig.	<ul style="list-style-type: none"> - Gir informasjon om en eller liste av bolig som eies av en eller mange eiere. - Gir informasjon om boligNr, seksjonsNr, beliggenhet av bolig og hvem som eier bolig. - Gir informasjon om en eller flere eiere som er knyttet til en eller flere boliger gjennom unik ID.
OwnerController	Kontroller som gir API for eiere av bolig.	<ul style="list-style-type: none"> - Gir personlig informasjon om eier som Fornavn, Etternavn osv. - Gir informasjon om eier av bolig som kan endres,

		slettes og lages på nytt. F.eks: mailadresse og telefonnummer kan endres på.
UploadController	Kontroller som gir API for boligmappe.	<ul style="list-style-type: none"> - Gir informasjon om dokumenter som kan lastes opp og lagres i systemet. For eksempel PDF dokument.
UserController	Kontroller som gir API for bruker(User)	<ul style="list-style-type: none"> - Gir informasjon om beboer(User) for eksempel Leietaker og Gjest. - Gir informasjon om hvilken bolig denne beboer tilknyttet. - Beboer kan slettes av Eier og sameie. - Bruker kan lagre informasjon om beboer i Nabonett.

Figur 5.8: Kontroller funksjonalitet

5.2.2 Core mappe

Som vist i figur 5.9, inneholder denne mappen fire undermapper som består av “Data Interfaces”, “Enums”, “Extensions”, “Model” og fire service klasser.



Figur 5.9: Core mappe

Data interfaces

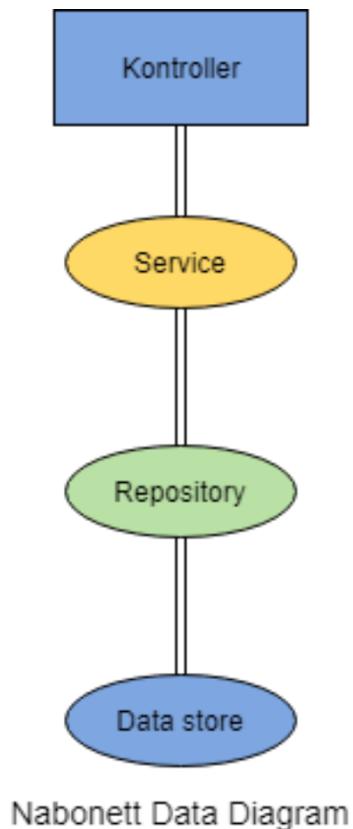
Data Interfaces mappen inneholder interfaces satt opp for “Repositories” som er koblet med kontroller via Service klasser i Core mappen.

Model

Model mappen som vist i figur 5.10 inneholder alle enheter (Klasser) som ble opprettet i tjener (back end). Hver enhet (klasse) responderer på en tabell i database og hver attributt i enheten (klassen) er representert som en kolonne i tabellen.

Extensions

Extensions mappen inneholder en klasse med kode for velging av tilfeldige verdier i data som ble brukt ved oppretting av dette prosjektet.

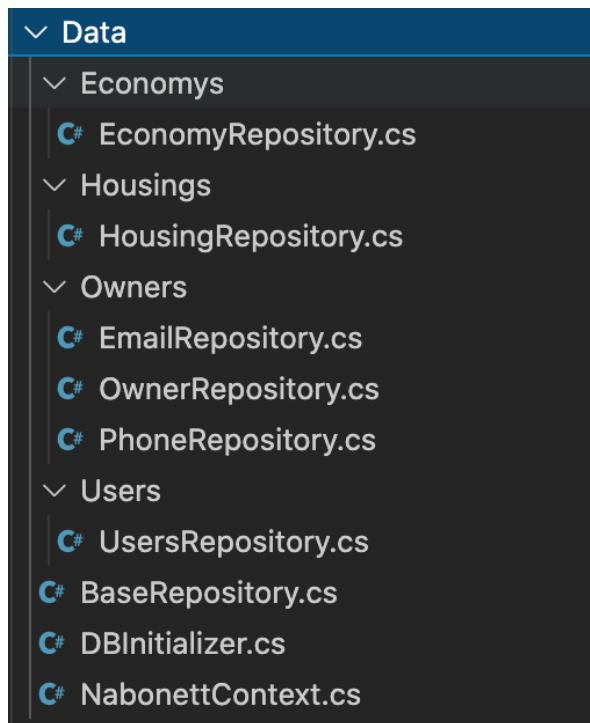


Nabonett Data Diagram

Figur 5.10: Bilde av kobling mellom leddene.

5.2.3 Data mappe

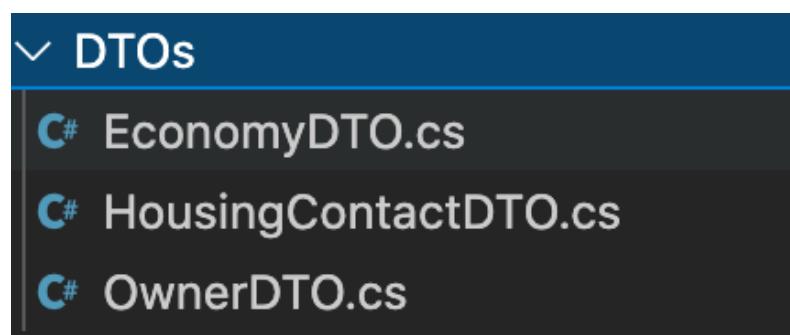
Som vist i figur 5.11 inneholder denne mappen vanligvis database logikken og spørsmål til databasen. Logikken som kreves for tilgang til data, er basert på kode i "Repository" klassene. "DBInitialize" i denne mappen inneholder klasser som tar seg av opprettning av database. Den lager databasen basert på datamodell klassene i Nabonett. I dette prosjektet er "DBInitialize" koblet til "TestBilder" mappen for å kunne generere et stort antall tilfeldige verdier for tabellene i databasen. I Nabonett "Context" klassen opprettet vi tabeller for databasen og viser kobling fra en tabell til en annen gjennom fremmednøkler.



Figur 5.11: Data mappe

5.2.4 DTO mappe

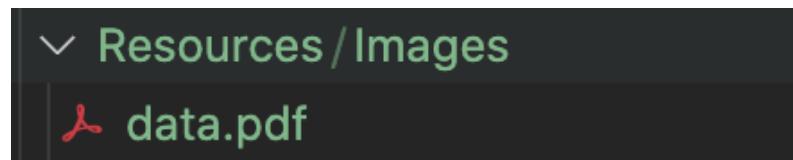
DTO står for “Data Transfer Object”. DTO er viktig for å redusere mengde data som sendes over ledningen mellom tjener og klient. Det blir dyrt å innhente store mengde data som er ikke brukt i klient siden. DTO tillater utviklere en modell som kartlegger store objekter og bare henter de verdiene som er etterlyst. Figur 5.12 viser “DTO” brukt i prosjektet.



Figur 5.12: “DTO” Mappe

5.2.5 Resources

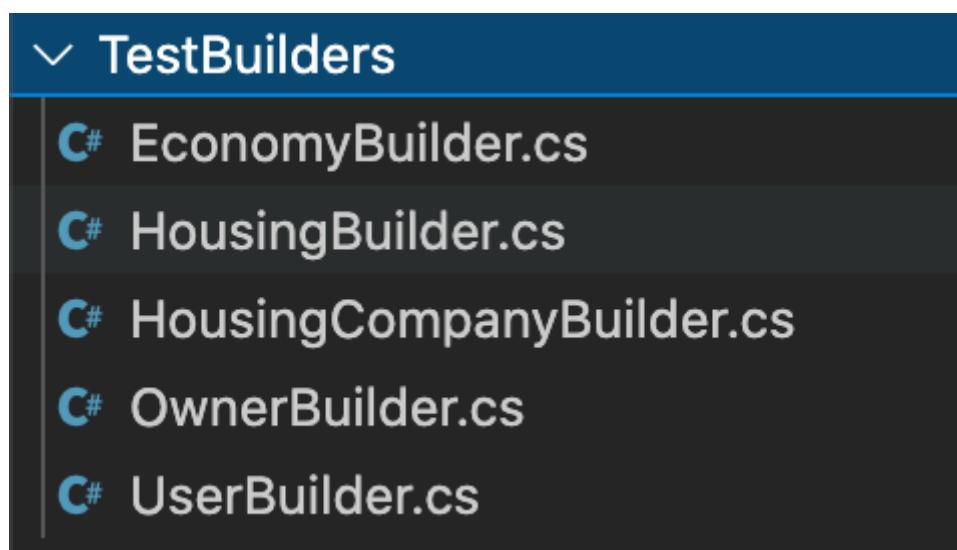
Denne mappen er brukt i lagring av dokumenter som bruker sender fra frontend. Planen var å opprette lenker i relasjonsdatabasen som kobles til hvert objekt i mappen, og at klienten vil kunne laste dem tilbake i applikasjonen. Figur 5.14 viser denne mappen som er direkte koblet med “UploadController” for opplasting av filer i Nabonetts systemet.



Figur 5.14: Resources mappe

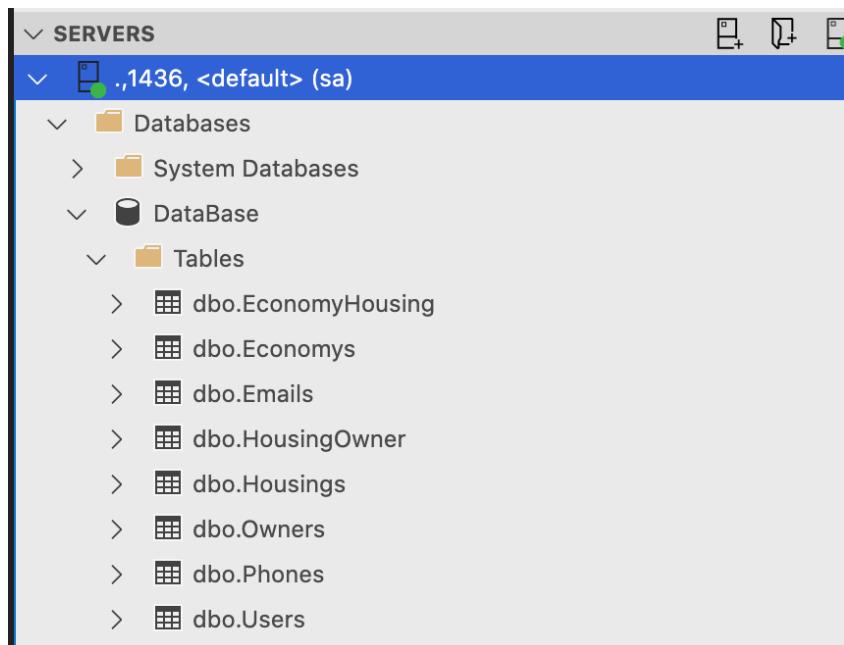
5.2.6 Testbuilder

Som vist i figur 5.15, inneholder denne mappen fem klasser som er knyttet med Model klasser og er brukt for bygging av falske database verdier. Ved bruk av “Bogus” i disse klassene er det laget store mengder data verdier for hver tabell i databasen.



Figur 5.15: “TestBuilders” mappe

Figur 5.16 viser oppretting av tabeller i database i Azure data studio.



Figur 5.16: Tabeller i Azure data studio

Tabellene tilsvarer modellene i applikasjonen. Hver tabell håndterer en kontroller som kobles til klienten. Det er noen tabeller som brukes til å håndtere data kompleksitet som er nødvendig for å lage en fungerende relasjonsdatabase. "Email" og "Phone" tabeller brukes i mange andre tabeller. Dette er viktig for ikke å lagre data i alle tabeller. "HousingOwner" tabell har nøkler fra "Housing" og "Owner" tabeller. Disse tillater søk etter eier av bolig ved hjelp av "ID" fra "Housing" tabell. Figurene herunder består tabeller beskrevet ovenfor.

	HouseId	HouseNr	SectionNr	Street	StreetNr	City	ZipCode	Country
1	01037711485	982	168	Sofiesvei	1	Fagerfjell	891	Norway
2	01068931289	655	260	Noahs Gate	0	Fetstrand	445	Norway
3	01099843159	167	470	Moenveien	1	Lillevik	444	Norway
4	02076907458	640	850	Eliassvei	0	Maleid	163	Norway
5	02105345715	166	350	Nordbys Vei	1	Storborg	948	Norway
6	03019525391	599	238	Olstadsvei	1	Nærødden	76	Norway
7	03047242826	902	346	Nedre Solfaret	0	Høynes	32	Norway
8	03059624317	878	515	Øvre Furuhagen	1	Sandvik	466	Norway
9	03105648854	299	94	Nedre Rognelia	0	Storby	346	Norway
10	03127733448	61	484	Borges Gate	0	Nordhavn	184	Norway
11	04069439425	172	962	Fredriks Vei	1	Malby	887	Norway
12	04097512960	412	862	Skoglundsgate	1	Lillehavn	179	Norway
13	04106249046	826	96	Nedre Granhagen	1	Høyby	847	Norway
14	05066045457	577	203	Madeleines Vei	1	Fetøy	613	Norway
15	06047208139	472	648	Nikolaigata	0	Høyeid	888	Norway

Figur 5.17: “Housings” tabell

Database
Tables
dbo.EconomyHousing
dbo.Economys
dbo.Emails
dbo.HousingOwner
dbo.Housings
dbo.Owners
Columns
Keys
Constraints
Triggers
Indexes
Statistics
dbo.Phones
dbo.Users
Views
Synonyms
Programmability
External Resources
Service Broker
Storage
Security

```

5   ,[City]
6   ,[IsDeleted]
7   FROM [DataBase].[dbo].[Owners]

```

	SSN	FirstName	LastName	Address	City	IsDeleted
3	03085248781	Daniel	Olsen	5368	Tandvik	0
4	03087922463	Julie	Evensen	6626	Maljfjell	0
5	03119104628	Andrea	Henriksen	3232	Loodden	0
6	03129825382	Andreas	Bjørnstad	5811	Lilleås	0
7	04046109177	Herman	Ruud	9538	Sandeid	0
8	04108446906	Elias	Hopland	4892	Fetøy	0
9	05016120687	Synne	Nielsen	8675	Innsand	0
1...	05016132499	Victor...	Dahl	6275	Sandvik	0
1...	05087409580	Ole	Dahl	9967	Malgrunn	0
1...	05096116162	Adrian	Jakobsen	7591	Lilleøy	0
1...	06059449476	Karoli...	Sæther	0504	Fetø	0
1...	06068242760	Kristo...	Svendsen	0062	Vestdal	0
1...	06118200959	Even	Christia...	0422	Storvåg	0
1...	06129602870	Andrea	Tangen	9606	Utsund	0
1	07010009640	Marie	Moe	3186	Ulfss	0

Figur 5.18: “Owners” tabell

	HousingsHouseId	OwnersSSN
1	01037711485	17070086514
2	01037711485	19048426444
3	01037711485	25028242703
4	01037711485	28047226416
5	01068931289	08077533827
6	01099843159	07098634287
7	02076907458	10069346001
8	02076907458	15106843977
9	02076907458	15109203138
10	02076907458	17087124797
11	02076907458	29095832839
12	02076907458	29098531805
13	02076907458	30075442706
14	02105345715	03085248781
15	02105345715	05016132499

Figur 5.19: "HousingOwner" tabell

Figurene herunder viser Swagger UI som er brukt for testing av data på tjener (backend) og eksempler av hvordan Swagger er blitt brukt å hente verdier fra "Housing" tabellen.

Economy	
GET	/api/Economy
GET	/api/Economy/{id}
GET	/housing/{houseId}

Housing	
GET	/api/Housing
POST	/api/Housing
PUT	/api/Housing
GET	/api/Housing/{id}
DELETE	/api/Housing/{id}
GET	/owner/{ssn}

Curl

```
curl -X GET "https://localhost:4201/api/Housing" -H "accept: text/plain"
```

Request URL

```
https://localhost:4201/api/Housing
```

Server response

Code	Details
200	Response body

```
[{"houseId": "01047829735", "houseNr": "323", "sectionNr": "47", "street": "Brustrenda", "streetNr": "1", "city": "Innv\u00f8g", "zipCode": "263", "country": "Norway", "owners": [{"ssn": "06119620048", "firstName": "Martine", "lastName": "Berntsen", "address": "6574", "city": "Lillevik", "email": "Martine_Berntsen@gmail.com", "phoneNumber": "41 68 40 59"}, {"ssn": "08055946974", "firstName": "Erik", "lastName": "Dahl", "address": "7645"}]}
```

Figur 5.20: "Housing" verdier i Swagger UI

5.2.7 Tjener (Back-end) Validering

Validering i back-end er implementert ved hjelp av “NuGet packages” i .NET Core. Validering er implementert i alle DTO. Overføring av data skjer mellom klient og tjener gjennom DTO. Validering er tilstede for å sikre tilførsel av riktig verdier. Figur 5.21 viser validering brukt i “OwnerDTO”. Ved hjelp av .NET Core bibliotek kan alle attributene i DTO klasse valideres.

```
NabonettWebAPI > DTOs > C# OwnerDTO.cs > {} NabonettWebAPI.DTOs
1  using System;
2  using System.Diagnostics.CodeAnalysis;
3
4  namespace NabonettWebAPI.DTOs
5  {
6
7      [ExcludeFromCodeCoverage]
8
9      public class OwnerDTO
10     {
11
12         public string SSN { get; set; }
13
14         public string FirstName { get; set; }
15
16         public string LastName { get; set; }
17
18         public string Address { get; set; }
19
20         public string City { get; set; }
21
22         public string Email { get; set; }
23
24         public string PhoneNumber { get; set; }
25
26     }
27 }
```

Figur 5.21 : Back-end validering i “OwnerDTO.cs”

6. Diskusjon og konklusjon

6.1 Diskusjon

Endrede kravspesifikasjoner underveis

Prosjektarbeid er uforutsigelig i noen tilfeller og det er normalt for kravspesifikasjoner å endre seg underveis i prosessen. Ved bruk av fossefallsmodellen er det nødvendig at kravspesifikasjoner er klare og tydelige. Ved hjelp av kontinuerlig kommunikasjon har vi tilpasset oss og endret der det tillates og løst dette problemet på en akseptabel måte.

Smitteverns utfordringer

En stor utfordring gjennom prosjekt løpet var manglende tilgang til fysiske samarbeid på grunn av Covid-19 situasjonen i Oslo. Opprinnelig plan for prosjektet var tett samarbeid i klientens kontor i Oslo. Dette var ikke mulig å gjøre hvis smitteverns reglene skulle opprettholdes. Planen ble dermed å jobbe mer selvstendig og holde kontinuerlig kommunikasjon med Daglig leder.

Flere medlemmer av gruppen er også småbarnsforeldre som på grunn av stengte skoler og barnehager måtte jobbe hjemmefra med barn og støy. Løsningen ble å jobbe senere på dagen for å kompensere for mistet arbeidstid på dagen. Selv med andre generelle utfordringer rundt Covid-situasjonen i Norge og andre utfordringer omkring prosjektarbeidet ble selve resultatet et godt bearbeidet produkt som både gruppe og klient er fornøyde med. Dermed anser vi prosjektet som vellykket.

Etiske data utfordringer

Database design er noe som endret seg underveis. I planleggingsfasen ble det bestemt at database skal bestå av å få tilgang til Styret.com sitt database. Dette ble endret på grunn av faren oppdaget ved hjelp av risikoanalyse. En database som inneholder personopplysninger skal ikke brukes for produksjon av webapplikasjonen. Det er større fare for tap av data og bryter med GDPR og personopplysningsloven.

Løsningen ble produksjon av en “Dummy” database som inneholder falske dataverdier produsert ved hjelp av “Bogus”.

6.2 Konklusjon

Dette prosjektet har gitt vår bachelor gruppe muligheten til å lære mye nytt. Vi har lært nye teknologier og ressurser som vi tar med oss videre og som styrker våre egne evner. Samtidig så har vi forsterket teori og kunnskap lært gjennom studieløpet og brukt dette i praksis gjennom dette prosjektet.

Nabonett webapplikasjon er en ferdigstilt kundeløsning som Styret.com er fornøyde med. Ferdig produkt utfyller kravspesifikasjonene og klientens ønsker, i tillegg til å bli godt mottatt av kundebasen. Nabonett tillater brukere å ha oversikt og styring over deres boliger. Brukere av Nabonett får deres hverdag effektivisert ved å ha tilgang til forskjellige funksjoner i ett og samme sted. Dem har tilgang til funksjoner som boligstyring, beboer styring, boligmappe og mye mer.

En ting som vi kanskje hadde vurdert annerledes er arbeidsmetode, om en inkrementell utviklingsprosess hadde resultert i et bedre prosjekt. Slik endring hadde kanskje forenklet prosessen bak forandring i kravspesifikasjoner. Men uten erfaring med inkrementelle utviklingsprosesser som scrum, er det vanskelig å adoptere rammeverket og ta det i bruk riktig.

Styret.com har store forventninger for produktet og er svært fornøyde med Nabonett. Selve webapplikasjonen skal bearbeides av interne utviklere hos Styret.com etter leveranse. Og er planlagt for utplassering siste kvartal av 2021.

6.3 Tilbakemelding fra arbeidsgiver

“Det har vært spennende og lærerikt å jobbe med prosjektgruppen. Gruppen hadde ambisjoner langt utover det vi trodde var mulig å få til innen tidsfristen, og har levert et bedre resultat enn vi hadde forventet.

Som det ser ut nå, vil vi videreutvikle nabonett og sette dette i kommersiell drift i løpet av 2021. Dette hadde ikke vært mulig uten prosjektgruppen. I tillegg til programmering, har de også vært med på både konseptutvikling og forretningsutvikling.

Gruppen har utfordret oss som oppdragsgiver på mange områder og har gitt oss løsninger på mulige problemer underveis. Kommunikasjon med gruppen har foregått uten problemer og vi har fått mange innspill vi kan ta med i videreutvikling av applikasjonen.

Tusen takk til alle sammen for en fantastisk innsats og spennende produkt! “ -
(Kristoffer G. Skappel)

7. Forslag til videre utvikling

Dette kapitlet tar for seg forslag til videre utvikling. Disse forslagene er basert på potensielle funksjoner som kan føre til forbedring av Nabonett.

1. Kjøpe tjenester tilknyttet leilighet

Dette er tjenester som brukere vil trenge og gjør det enklere ved å ha det tilgjengelig i Nabonett.

- Be om verdivurdering.
- Forkjøpsrett.
- Varsel om felles utbedring og muligheter.
- Tilvalg ved nybolig.

2. Betale fakturaer gjennom Nabonett

Ved å tillate brukere muligheten til å betale gjennom Nabonett blir det enklere å holde styr på betalinger.

3. GDPR innsyn

Tillate brukere innsyn i bruken av deres informasjon i forhold til GDPR regler.

4. Sende inn klager/krav til styre

Melde av til gjeldende styre i boligselskapet om klager/krav gjennom Nabonett for å starte en oppretting prosess.

5. Admin innsyn

Gjøre det mulig for administratorer å se logg-inn historikk og en loggføring av valgene tatt.

- se logg-inn historikk.
- Se logg (detaljert loggføring).

Ordliste:

Inneholder forklaring på ord og begreper brukt i rapport.

Boligselskap: Borettslag, eierseksjonssameie eller aksjeandelslag.

Fler boligselskap: flere borettslag, eierseksjonssameier eller aksjeandelslag

Forretningsfører: et selskap som på vegne av et boligselskap fører regnskap, rapporterer inn til det offentlige, gir råd til styret og sender ut fakturaer til eierne av boliger i selskapet.

Bolig: eierseksjon i eierseksjonssameie, andel i borettslag, aksje i aksjeandelslag eller enebolig.

Eier: eier av bolig.

Eiendom: Et eller flere bygg eid av et boligselskap. Se "Boligselskap".

Bruker: Eier av bolig er bruker som har tilgang til Nabonett gjennom en konto.

Beboer: En person som blir lagt inni Nabonett av eier, men som har ikke har direkte tilgang.

Teknologisk stack: En liste av alle teknologiske data, systemer, programmer brukt for å bygge en applikasjon eller system.

SSO: Single-sign-on tillater brukere tilgang til flere applikasjoner gjennom en login. Legitimasjon fra en logg-inn blir brukt i andre applikasjoner og hopper over logg-inn der.

Docker-bilde/docker-image: Lager en server miljø for prosjektdatabase.

Klient (Front-end): Referer til alt i webapplikasjonen som er vist på nettleser enhet.

Tjener (Back-end): Referer til den siden av programmet som sluttbruker ikke har innsyn til. Inneholder kode og logikk som er nødvendig for å kjøre innhold til klient siden.

Kontroller (Controller): Kontroller er en klasse som håndterer dataflyt mellom tjener og klient via HTTP kall.

DTO: (Data transfer object) referer til objekter som minsker datamengde som er overført fra server til klient.

Klasser: Referer til en objekt som definerer attributter og metoder til en den objektet.

Metode: Referer til instruksjoner som vil utføre bestemt oppgave. Den kan returnere noe eller ingenting.

Kildemappe = Source folder / src : referer til mappe som inneholder kildekode for applikasjonen.

Azure AD B2C : (Azure Active Directory Business to Customer) referer til Microsoft Azure sin løsning for håndtering av autentisering.

Bruksanvisning

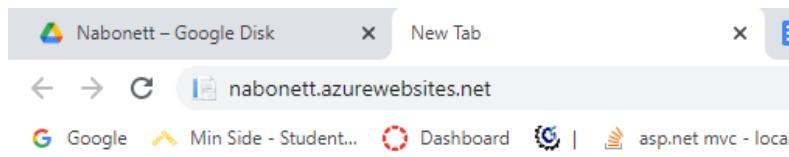
Denne seksjonen tar for seg bruken av web applikasjonen og de ulike funksjonene. Hensikten med denne delen er å gi leseren en forståelse av hvordan man skal navigere seg gjennom og bruke webapplikasjonen. Det kreves ingen forkunnskap utenom generell leseevne. Dermed følger denne seksjonen en stegvis forklaringsmodell og går ut fra samme rekkefølge som forklart i seksjon 3.3.

8.1 Logg-inn side

8.1.1 Logge inn i webapplikasjon

1. Skrive inn nettadressen til Nabonett:

<https://nabonett.azurewebsites.net>

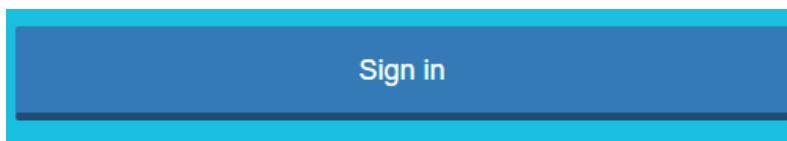


2. Skriv inn mailadresse og passord inn riktige felt.

Email Address
þ333972@oslomet.no

Password Forgot your password?

3. Trykk på "Sign In" knapp.

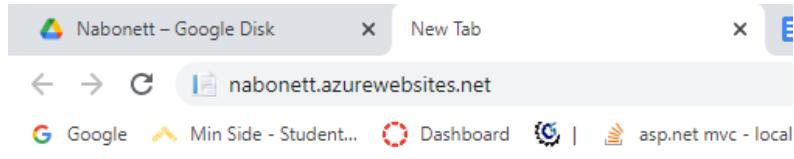


4. Hvis både mailadresse og passord samstemmer med en eksisterende bruker vil man bli overført Nabonett sitt hjemmeside.

8.1.2 Tilbakestilling av passord

1. Skriv inn nettadressen til Nabonett:

<https://nabonett.azurewebsites.net>



2. Trykk på "Forgot your password? knapp.

A screenshot of a password recovery page. It has a teal header with 'Email Address' and a text input field containing 's333972@oslomet.no'. Below it is a 'Password' field with masked input. A 'Forgot your password?' link is visible in the top right.

3. Skriv inn mailadresse og trykk "Send verification code" knapp.

A screenshot of a 'styret.com' verification page. It has a teal header with the logo. Below is an 'Email Address' field with 's333972@oslomet.no'. A blue button labeled 'Send verification code' is highlighted. At the bottom are 'Continue' and 'Cancel' buttons.

4. Motta mail som inneholder verifikasjonskode.

The screenshot shows an email from Microsoft on behalf of StyretB2C. The subject is "Verify your email address". The body of the email says "Thanks for verifying your s333972@oslomet.no account! Your code is: 854448". It also includes a signature "Sincerely, Nabonett". Below the email is a light gray bar with the text "Svar" and "Vidresend".

5. Skriv inn verifikasjonskode og trykk "Verify code" knapp for å bli sendt videre.

The screenshot shows a Styret.com verification page. It displays the text "Verification code has been sent to your inbox. Please copy it to the input box below." Below this are two input fields: "Email Address" containing "s333972@oslomet.no" and "Verification code" containing "854448". There are two buttons: "Verify code" (in blue) and "Send new code". At the bottom are "Continue" (in green) and "Cancel" buttons.

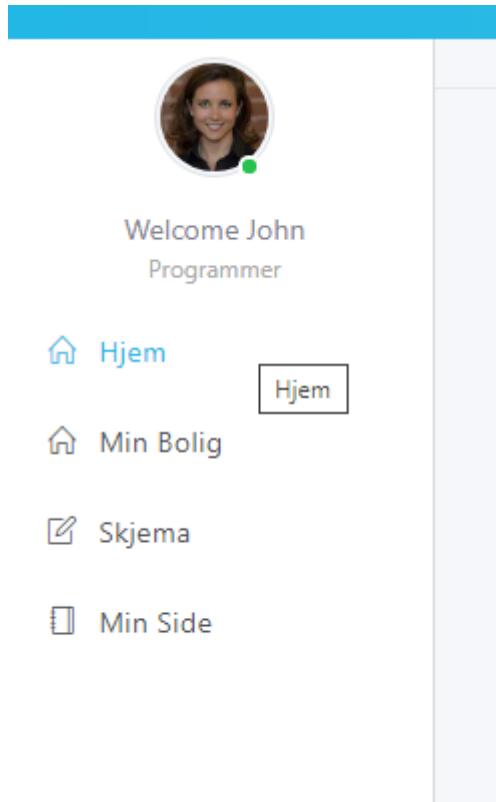
6. Skriv inn nye passord og trykk "Continue" for å lagre.

The screenshot shows a password change form with two input fields: "New Password" and "Confirm New Password", both containing several asterisks. At the bottom are "Continue" (in green) and "Cancel" buttons.

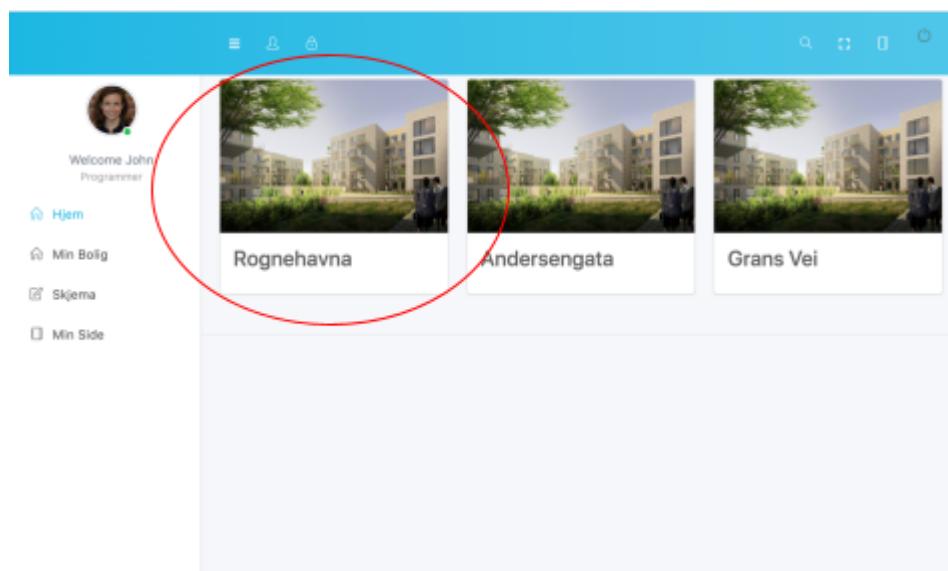
8.2 Hjemmeside

8.2.1 Velg bolig å styre

- Trykk på "Hjem" i sidemenyen for å komme til siden.



- Velg "bolig" kort for å velge bolig.



- Systemet vil overføre bruker til "Min bolig" for valgt bolig.

8.3 Min Bolig

Steg 1. i **8.3.1** er ikke tilstede i **8.3.2**, **8.3.3**, **8.3.4** og **8.3.5** for å minske på unødvendig repetisjon, men er en forutsetning.

8.3.1 Legge til Beboer

- Trykk på “Legg til beboer” knappen for å starte modalen som registrerer nye beboere.

The screenshot shows a user profile "Welcome John" and navigation links for "Hjem", "Min Bolig" (which is selected), "Skjema", and "Min Side". The main area displays a table of residents:

				Madeleine Nilsen		
397	376	Andersengata	Aksel Strand	Økonomi	Last opp	Alexander Nordby
			Mathilde Glosli			
788	672	Grans Vel	Alexander Nordby	Økonomi	Last opp	Sondre Thoresen

Below the table is a modal titled "Beboere" (Residents) with columns: Fornavn, Etternavn, Telefon, Email, Type, Kontakt. It contains one row of data: Ida, Borge, 37830278, ida1B@gmail.com, Økonomi, and two buttons: Slett (Delete) and Endre (Edit). At the bottom right of the modal is a blue button labeled "Legg til nybeboer" (Add new resident), which is circled in red.

- Velg hva slags type bruker beboeren er.

The screenshot shows a modal window titled "Velg bruker" (Select user). On the left, there is a sidebar with the word "takt". The main area has a dropdown menu under "Registrer Bruker" with "Leietaker" selected (highlighted in blue). Below the dropdown is a "Fornavn" (First name) input field.

- Skriv inn informasjon i skjema.

Registrer Beboer

Registrer
Bruker **Leietaker**

Fornavn **Duale**

Etternavn **Mohamed**

Telefon **48502245**

Email **dualemohamed1@hotmail.com**

Fakturamottaker

Kontaktperson

Lagre **Avbryt**

4. Trykk på lagre for å registrere ny beboer med utfylt info.
5. Ny beboer er registrert i system.

8.3.2 Slette beboer

- Trykk på “Slett” knapp for å slette beboer.

The screenshot shows a web application interface. On the left, there is a sidebar with a user profile picture, the name "Welcome John Programmer", and navigation links: "Hjem", "Min Bolig" (which is selected and highlighted in blue), "Skjema", and "Min Side". The main content area has a header "Beboere". Below the header is a table with columns: Fornavn, Etternavn, Telefon, Email, Type, and Kontakt. The table contains several rows of resident data. A modal dialog box is overlaid on the table, centered over the row for "Ida Borge". The modal has the title "Ønsker du å slette". It contains the resident's information: "Ida Borge" and "37830278 Ida18@gmail.com". At the bottom of the modal are two buttons: "Lukk" (Close) and "Slett" (Delete), with "Slett" being highlighted with a red circle. A "Legg til beboer" (Add resident) button is located at the bottom right of the main table area.

- Trykk på “Slett” knappen i pop up vindu for å bekrefte valget.

This screenshot shows the same web application interface as the previous one, but the modal dialog from the previous step is still open. The modal now displays a confirmation message: "Ønsker du å slette". It includes the "Lukk" (Close) and "Slett" (Delete) buttons. The background table of residents is visible behind the dialog.

- Systemet fullfører slettingen

8.3.3 Endre beboer

- Trykk på “Endre” knapp for å åpne “Endre Beboer” skjema.

The screenshot shows a software interface for managing residents. On the left, there's a sidebar with a user profile picture, 'Welcome John Programmer', and navigation links: 'Hjem', 'Min Bolig' (which is selected and highlighted in blue), 'Skjema', and 'Min Side'. The main area has two tables. The top table lists residents by address: Madeleine Nilsen at 397 Andersengata, Alexander Nordby at 376 Andersengata, Aksel Strand at 397 Andersengata, and Mathilde Glosli at 376 Andersengata. The bottom table lists residents by address: Alexander Nordby at 788 Grans Vei and Sondre Thoresen at 672 Grans Vei. Each resident row has a 'Økonomi' (Economy) button and a 'Last opp' (Upload) button. Below these tables is a 'Beboere' (Residents) section with a table showing columns: Fornavn (First Name), Etternavn (Last Name), Telefon (Phone), Email, Type, and Kontakt. A single resident entry for 'Ida Borge' is shown with phone 37830278 and email Ida18@gmail.com. To the right of this table are 'Slett' (Delete) and 'Endre' (Edit) buttons, with the 'Endre' button circled in red. At the bottom right of this section is a 'Legg til beboer' (Add Resident) button.

- Endre informasjon i “Endre Beboer” skjema.

The screenshot shows the 'Endre Beboer' (Edit Resident) dialog box. It includes fields for 'Fornavn' (First Name) set to 'Ida Nilsen', 'Etternavn' (Last Name) set to 'Borge', 'Telefon' (Phone) set to '37830278', and 'Email' set to 'ida.Borge@gmail.com'. There are also checkboxes for 'Faktura Mottaker' (Invoice Receiver) and 'Kontakt Person' (Contact Person), both of which are checked. At the bottom are 'Lagre' (Save) and 'Avbryt' (Cancel) buttons.

- Trykk på “Lagre” for å bekrefte endringer.

8.3.4 Laste opp dokument

- Trykk på “Last opp” knapp for å laste opp dokument til valgt bolig.

The screenshot shows a user interface for managing apartments. On the left, there's a sidebar with navigation links: Hjem, Min Bolig (which is selected and highlighted in blue), Skjema, and Min Side. The main area is titled 'Min Bolig' and contains a table titled 'Bolig Liste'. The table has columns for Leilighet Nr (Apartment Number), Seksjon (Section), Bolig Selskap (Apartment Company), Eiere (Owners), Økonomi (Economy), and Last opp (Upload). The first row shows apartment number 208, section 383, company Rognehavn, owners Alexander Nordby and Madeleine Nilsen, economy button, and an 'Upload' button circled in red. Subsequent rows show apartments 397 and 788 with their respective details and economy/upload buttons.

- Velg dokument som skal lastes opp.

This screenshot shows the same 'Bolig Liste' page as the previous one. A file browser window is overlaid on the page, showing a list of files in a folder named 'Nedlasting'. One file, 'Datsikkerhet Lab.pdf', is selected and highlighted. To the right of the file list, a preview pane displays the selected file's details: name, type ('PDF-dokument'), size ('537 kB'), creation date ('torsdag 19. november 2020 08:56'), modification date ('torsdag 19. november 2020 08:56'), and a 'Vis mer' link. At the bottom of the preview pane are 'Avbryt' and 'Åpne' buttons.

- Dokument er lagret i “Resources” mappe.

8.3.5 Vis økonomi

- Trykk på “Økonomi” knapp for å vise liste av faktura for valgt bolig.

Leilighet Nr	Seksjon	Bolig Selskap	Eiere		
208	383	Rognehavn	Alexander Nordby Madeleine Nilsen	Økonomi	Last opp
397	376	Andersengata	Aksel Strand Mathilde Glosli	Økonomi	Last opp
788	672	Grans Vei	Alexander Nordby Sondre Thoresen	Økonomi	Last opp

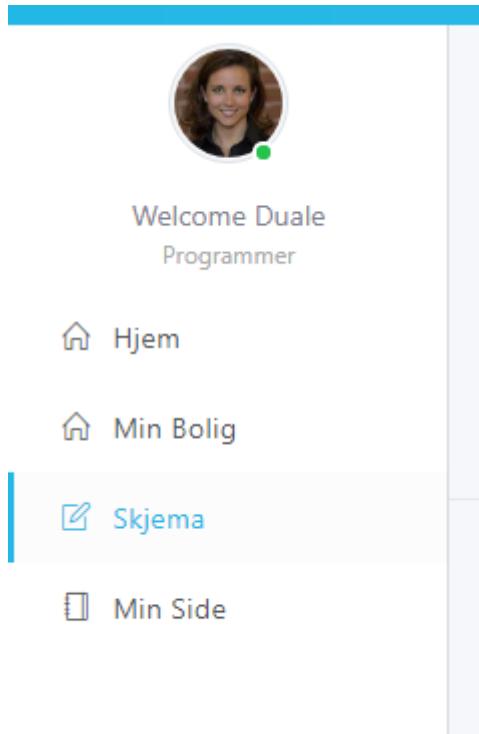
- Systemet vil vise liste av faktura for valgt bolig.

Beløp	Forfallsdato	Status	Faktura Nr	KID	KontoNummer
363.99	2022-04-25T05:01:53.4248834	Avventer	1884225802682	3310097389535	1763700335290
686.03	2021-05-29T16:48:14.5510729	Avventer	3089573372148	5616002808572	8015726367384
795.01	2021-06-30T23:26:38.0716882	Betalt	0838998895571	7865892654334	7071651125193
37.13	2022-03-11T18:51:08.3986178	Betalt	5377205655249	2707799959639	5323236477930
912.71	2021-06-14T16:53:38.6699877	Avventer	5449046676353	8076022904758	3873482874097
319.02	2022-04-01T13:43:11.1750286	Betalt	4280331774757	0254126355807	2566336418882
320.71	2022-03-19T17:41:09.6265093	Avventer	0543673704146	0334449901643	9932000976140
964.46	2022-03-29T18:44:12.281852	Avventer	5645874112036	3186166715082	8831209821453

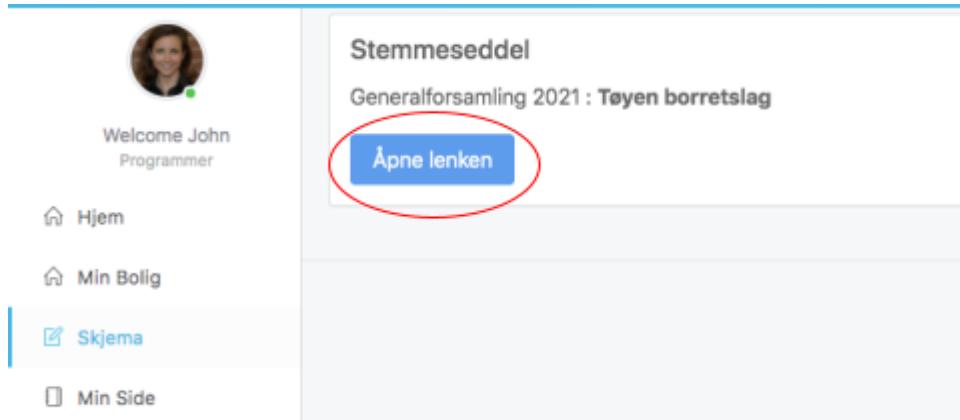
8.4 Skjema

8.4.1 Utfør stemmeseddel

1. Trykk på "Skjema" i sidemenyen for å komme til siden.



2. Trykk på "Åpne lenken" knappen for å bli overført avstemning seddel.



3. Svar på spørsmålene.

Styreleder (2 år)

1

Robert Downey Jr.

2

Colin Farrell

Styremedlem (2 år)

1

Robert Downey Jr.

2

David Beckham

3

Colin Farrell

4

Christian Bale

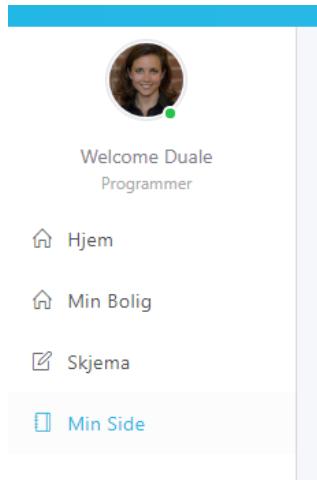
Jeg ønsker ikke å stemme på denne saken (blank stemme)

4. Trykk på “Send inn svarene dine” for å bli ferdig.

8.5 Min Side

8.5.1 Endre kontaktinformasjon

- Trykk på "Min Side" i sidemenyen for å komme til siden.



- Endre adresse felt.

Kontakt Informasjon

Navn	Elise
Epost	Elise.Bjerke19@hotmail.com
Telefon	92235581
Adresse	Bergensveien 13
Posted	Lodal

Oppdater

- Trykk på oppdater for å lagre endringer.

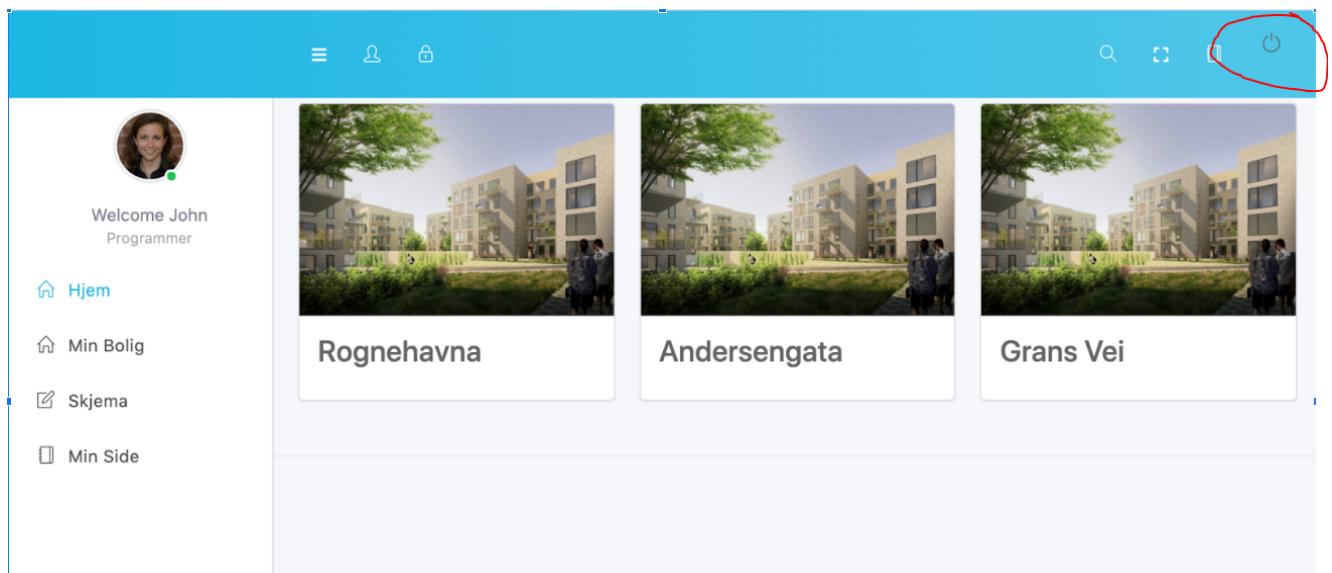
Kontakt Informasjon

Navn	Elise
Epost	Elise.Bjerke19@hotmail.com
Telefon	83 13 75 87
Adresse	Osloveien 23
Posted	Lodal

Oppdater

8.5 Logg-ut

1. Trykk på “logg-ut” ikon i navigasjonsbaren.



2. Systemet fører bruker til innloggingssiden.

Kilder:

- [1] Sommerville, I. (2015). *Software Engineering (Global Edition)*. New York: Pearson Education Limited.
- [2] Sandnes, F.E. (2018). *Universell utforming av IKT-systemer (Brukergrensesnitt for alle)*. Oslo: Universitetsforlaget.
- [3] Sharma, L. (2016). What is the Waterfall method?. Hentet fra
<https://www.toolsqa.com/software-testing/waterfall-model/>
- [4] Martin, M., Horvitz, Y., Krish, G., Gomez, E. & Macy, M (2020). *Overview of tokens in Azure Active Directory B2C*. Azure AD B2C Documentation. Hentet fra:
<https://docs.microsoft.com/en-us/azure/active-directory-b2c/tokens-overview>
- [5] Van der Stock, A & Frenchie (2014). OWASP Developer Guide Reboot. Secure Coding Principles. Hentet fra:
<https://github.com/OWASP/DevGuide/blob/dc5a2977a4797d9b98486417a5527b9f15d8a251/DevGuide2.0.1/Secure%20Coding%20Principles.doc>
- [6] Harris, S. & Maymi, F. (2018). *CISSP All-in-One Exam Guide, (Eighth Edition)*. New York: McGraw Hill Education.
- [7] Nielsen, J. (1994). 10 Usability Heuristics for User Interface Design. Hentet fra
<https://www.nngroup.com/articles/ten-usability-heuristics/>
- [8] Personopplysningsloven. (2018). Lov om behandling av personopplysninger (LOV-2018-06-15-38). Hentet fra <https://lovdata.no/dokument/NL/lov/2018-06-15-38>
- [9] Spillner, A., Linz, T. & Schaefer, H. (2014). *Software Testing Foundations : A Study Guide for the Certified Tester Exam*. San Rafael: Rocky Nook.

Teknologier

[10] Angular (2021). What is Angular?. Hentet fra
<https://angular.io/guide/what-is-angular>

[11] Bootstrap (2021) Build fast, responsive sites with Bootstrap. Hentet fra
<https://getbootstrap.com/>

[12] Microsoft (2021) Getting Started. Hentet fra <https://code.visualstudio.com/docs>

[13] Ng-Bootstrap (2020) Introduction. Hentet fra
<https://ng-bootstrap.github.io/#/getting-started>

[14] Microsoft (2021). Learn to code in Visual studio. Hentet fra:
<https://visualstudio.microsoft.com/vs/getting-started/>

[15] Microsoft (2019). What's new in .Net Core 3.1. Hentet fra
<https://docs.microsoft.com/en-us/dotnet/core/whats-new/dotnet-core-3-1>

[16] Microsoft (2020). What is Azure Data Studio?. Hentet fra
<https://docs.microsoft.com/en-us/sql/azure-data-studio/what-is-azure-data-studio?vie w=sql-server-ver15>

[17] Docker (2021). Docker overview. Hentet fra
<https://docs.docker.com/get-started/overview/>

[18] Swagger (2021). What is OpenAPI?. Hentet fra
<https://swagger.io/docs/specification/about/>

[19] Postman (2021). Introduction. Hentet fra
<https://learning.postman.com/docs/getting-started/introduction/>

[20] Chaves, B. (2021) Bogus for .NET: C#, F# and VB.NET. Hentet fra
<https://github.com/bchavez/Bogus>