NTNU
Norwegian University of
Science and Technology

EcoHeat - Et styringssystem for å gjøre huset ditt smartere

Forfattere
Henrik Solum
Jostein Enes
Tarjei Holtskog

Bachelor i Programvareutvikling og Ingeniørfag - Data
20 ECTS
Institutt for Datateknikk og Informatikk
Norges teknisk-naturvitenskapelige universitet,

16.05.2017

Veileder Frode Haug

Sammendrag av Bacheloroppgaven

Tittel:	**EcoHeat - Et styringssystem for å gjøre huset ditt smartere**
Dato:	16.05.2017
Deltakere:	Henrik Solum Jostein Enes Tarjei Holtskog
Veiledere:	Frode Haug
Oppdragsgiver:	Hark Technologies
Kontaktperson:	Joar Gunnarsjaa Harkestad
Nøkkelord:	Bachelor, IIK, EcoHeat, React, React Native, smarthusteknologi, styringssystem, automatisering
Antall sider:	154
Antall vedlegg:	16
Tilgjengelighet:	Åpen

Sammendrag:

EcoHeat er et styringssystem for å kontrollere de tilhørende smartpluggene. Denne bacheloroppgaven fokuserer på utviklingen av mobil- og webapplikasjonen for dette systemet. Hensikten med disse applikasjonene er å tilby et intuitivt grafisk brukergrensesnitt til brukerne, slik at de enkelt kan administrere elektriske enheter i boligene sine.

Oppgaven beskriver prosessene rundt utviklingen av applikasjonene, fra planlegging til implementasjon.

Summary of Graduate Project

Title:	**EcoHeat - Et styringssystem for å gjøre huset ditt smartere**
Date:	16.05.2017
Authors:	Henrik Solum Jostein Enes Tarjei Holtskog
Supervisor:	Frode Haug
Employer:	Hark Technologies
Contact Person:	Joar Gunnarsjaa Harkestad
Keywords:	Bachelor, IIK, EcoHeat, React, React Native, smart home technology, control system, automation
Pages:	154
Attachments:	16
Availability:	Open

Abstract:

EcoHeat is a system to control the associated smart plugs, where this thesis focuses on the development the mobile and web applications for the system. The purpose of these applications are to provide a intuitive graphical user interface, so that the users of the system can easily manage electrical devices in their houses.

This thesis describes the processes around developing said applications, from planning to implementation.

Forord

Vi ønsker å takke vår veileder Frode Haug for all veiledning under hele prosjektperioden. Vi vil også takke faglærerne Eivind Johansen for hans grundige gjennomgang av våre Hi-Fi prototyper og alle råd han gav oss til forbedring av design og Tom Røise for hans veiledning rundt kravspesifikasjon.

Vi vil også rette en stor takk til venner og familie som har bidratt med tips og innspill til utførelsen av denne rapporten.

Til slutt ønsker vi å takke vår oppdragsgiver Hark Technologies og kontaktperson Joar Harkestad. Hark Technologies for muligheten til å jobbe med denne oppgaven og all tilretteleggelsen som ble gjort og Joar for å alltid være tilgjengelig gjennom hele prosjektperioden.

Innhold

Forord . . . . . **iii**
Innhold . . . . . **iv**
Figurer . . . . . **vii**
Kodeeksempler . . . . . **viii**
1 Innledning **1**
1.1 Problemområde 1
1.2 Avgrensing 1
1.3 Oppgavedefinisjon 2
1.4 Målgruppe 3
1.4.1 Rapporten 3
1.4.2 Applikasjonene 3
1.5 Formål 3
1.6 Læringsutbytte 4
1.7 Akademisk bakgrunn 4
1.7.1 Faglig bakgrunn 4
1.7.2 Manglede bakgrunnskunnskap 5
1.8 Rammer 5
1.8.1 Gjennomføring 5
1.8.2 Arbeidsmetoder 5
1.8.3 Fremdriftsplan 5
1.8.4 Prosjektorganisering 5
1.8.5 Øvrige roller 6
1.9 Rapporten 7
1.9.1 Terminologi 7
1.9.2 Praktisk 7
1.9.3 Struktur 7
2 Kravspesifikasjon **9**
2.1 Endring av oppgaven 9
2.2 Funksjonelle krav 9
2.2.1 Use case modell 10
2.2.2 Overordnet use case beskrivelse 11
2.2.3 Detaljerte use case beskrivelse 13
2.3 Supplementære krav 14
2.3.1 Systemkrav 14
2.3.2 Ytelse 15

2.3.3 Internasjonalisering . . . 16
2.3.4 Brukervennlighet . . . 16
2.3.5 Sikkerhet . . . 17
3 Analyse . . . 18
3.1 Valg av teknologier . . . 18
3.1.1 Utvikling av applikasjonene . . . 18
3.1.2 Brukerhåndtering . . . 19
3.1.3 Datalagring . . . 21
4 Design og arkitektur . . . 23
4.1 Design . . . 23
4.1.1 Inspirasjon til brukergrensesnitt . . . 23
4.1.2 Plassering av funksjonalitet . . . 25
4.1.3 Mobil . . . 26
4.1.4 Web . . . 29
4.2 Arkitektur . . . 31
4.2.1 Artefakter . . . 31
4.2.2 Filstruktur . . . 33
5 Realisering og implementasjon . . . 35
5.1 Utviklingsmiljø . . . 35
5.2 Versjonskontroll . . . 35
5.3 React . . . 35
5.4 Redux . . . 37
5.5 Redux Thunk . . . 39
5.6 Firebase . . . 39
5.7 Implementasjon av mobilapplikasjonen . . . 41
5.7.1 Navigasjon . . . 43
5.7.2 Internasjonalisering . . . 44
5.7.3 Gjenbruk av komponenter . . . 46
5.8 Implementasjon av webapplikasjonen . . . 47
5.8.1 Internasjonalisering . . . 47
5.8.2 Navigasjon . . . 49
5.8.3 Sikkerhet . . . 50
5.8.4 Implementasjon av webdesign . . . 51
6 Testing og kvalitetssikring . . . 59
6.1 Testing . . . 59
6.2 Kvalitetssikring . . . 59
7 Avslutning . . . 62
7.1 Diskusjon . . . 62
7.1.1 Resultater . . . 62
7.1.2 Alternativer . . . 63

7.2 Kritikk av oppgaven . . . 64
7.3 Videre arbeid . . . 65
7.4 Evaluering av gruppen . . . 66
7.4.1 Introduksjon . . . 66
7.4.2 Organisering . . . 67
7.4.3 Arbeidsfordeling . . . 67
7.4.4 Prosjektet som arbeidsform . . . 67
7.5 Konklusjon . . . 68
Bibliografi . . . **69**
A Terminologi . . . **72**
B Lo-Fi Prototype for GUI . . . **74**
C Hi-Fi Prototype for GUI . . . **79**
D Opprinnelig databasestruktur . . . **88**
E Databasestruktur . . . **90**
E.1 Forklaring til figur 1 . . . 92
E.1.1 Locations . . . 92
E.1.2 Users . . . 93
F Aktivitetsdiagram . . . **94**
G auth.js . . . **105**
H webpack.config.js . . . **108**
I Node.js moduler . . . **110**
J Oppgavetekst . . . **112**
K Prosjektplan . . . **115**
L Første statusrapport . . . **135**
M Andre statusrapport . . . **139**
N Tredje statusrapport . . . **143**
O Møtereferater oppsummert . . . **146**
O.1 Veileder . . . 147
O.2 Oppdragsgiver . . . 147
O.3 Internt . . . 148
P Arbeidslogg . . . **149**

Figurer

1 EcoHeat smartplugg . . . 2

2 Use Case diagram for EcoHeat . . . 10
3 Markedsandeler smarttelefoner 2015Q4-2016Q3 . . . 14
4 Global distribusjon av Android versjoner April 2017 . . . 15

5 Sosiale innloggingsmuligheter. . . . 20
6 NoSQL vs SQL oppsummert . . . 21

7 Slik holder folk mobiltelefonen sin . . . 25
8 Forside (Hjem) - Mobil . . . 26
9 Mine enheter - Mobil . . . 26
10 Detaljert enhetsside - Mobil . . . 27
11 Kalender - Mobil . . . 27
12 Innstillinger - Mobil . . . 28
13 Hjem - Web . . . 29
14 Mine enheter - Web . . . 30
15 Kalender - Web . . . 30
16 Innstillinger - Web . . . 31
17 Deployment diagram . . . 31
18 Overordnet filstruktur for mobil . . . 33
19 Overordnet filstruktur for web . . . 34

20 Sammenligning av prototype og implementasjonen av 'Hjem' på mobil . . . 42
21 Forbedring av boligvalg i implementasjon. . . . 43
22 Gjenbruk av komponenter . . . 46
23 Eksempel på URL med React Router . . . 49
24 Link for innlogging og utlogging . . . 51
25 Navigeringslinjen for mobil . . . 52
26 Nedfallsmeny for valg av hus . . . 53
27 Hjem - Webapplikasjonen . . . 54
28 Filter for graf . . . 55
29 Representasjon av enheter - Web . . . 57
30 Detaljert enhetssiden - Web . . . 57
31 Registrering av ny enhet - Web . . . 58

32 Eksempel på advarsel fra ESLinter, basert på regelsettet fra Airbnb . . . 60

Kodeeksempler

1 Eksempel på bruk av React med både klassebaserte og funksjonelle komponenter og props. . . . 36
2 Eksempel på initiering av Redux i en React-applikasjon. . . . 37
3 Eksempel på en action creator. . . . 38
4 Eksempel på en reducer. . . . 38
5 Eksempel på bruk av connect() og Redux store. . . . 39
6 Eksempel på initiering av Firebase. . . . 40
7 Eksempel på innlogging med Firebase. . . . 40
8 Eksempel på å hente data fra Firebase. . . . 41
9 Oppsett av router med forskjellige navigasjonsflyter 44
10 Konfigurasjon av react-native-i18n modulen. . . . 45
11 Oppsett av ressursfilene. . . . 45
12 Uthenting av verdier fra ressursfilene. . . . 46
13 Ressursfiler for I18N - Web 47
14 Identifisering av brukerens språk 47
15 Oppsett av IntlProvider 48
16 Bruk av React Intl 48
17 Oppsett av React Router 49
18 Rendermetoden til App 50
19 Bruk av Link for navigering i webapplikasjonen 50
20 Betinget rendering 52
21 Oppsett av navigasjonslinjen 52
22 Oppsett av nedfallsmenyen for hus 53
23 Deklarativt oppsett av Recharts 54
24 Oppsett av React Skylight 55
25 Representasjon av smartplugger 56
26 Eksempel på kommentar skrevet i JavaDoc stil. . . . 61
27 Eksempel på bruk av PropTypes 61

1 Innledning

1.1 Problemområde

De siste årene har 'Internet of Things' - IoT (Tingenes internett) gradvis blitt et mer aktuelt tema. I løpet av dette århundret har antallet elektroniske enheter i vår hverdag økt og vi får stadig flere ting som utveksler informasjon over internett. Eksempler på dette er smarttelefoner, smartklokker, armbånd som måler den fysiske formen din og automatisk registrerer dette i en applikasjon eller termostater som varsler deg om at det er for varmt i boligen din, slik at du kan skru ned varmeovnene ved hjelp av mobiltelefonen din.

Selve begrepet 'Internet of Things' kan dateres tilbake til 1999 fra en presentasjon av Kevin Ashton [1], men selve prinsippet kan spores enda lengre tilbake i historien. Allerede i 1982 ble det utforsket da noen studenter ved Carnegie Mellon University installerte sensorer i en Cola-maskin, slik at den kunne rapportere hvorvidt den var tom via en server. [2] Videre i 1990 utviklet John Romkey og Simon Hackett det som av mange regnes som verdens første IoT-enhet, en brødrister som kunne bli skrudd av og på over internett. [3]

Opp gjennom tiden har IoT utviklet seg til å beskrive elektroniske enheter, hus og annet som er utstyrt med elektroniske måleinstrumenter, programvare, sensorer og internettilgang. Disse enhetene kan så samle og utveksle informasjon og data over internett. IoT åpner også opp for fjernstyring via internett, noe som stadig blir en større del av vår hverdag. [4]

Et av de største prosjektene innenfor IoT og fjernstyring er smarthus. Smarthus er et samlebegrep for automatiserte og intelligente boliger med mulighet for styring av lys, varme, overvåkning, sikkerhet og annen velferdsteknologi. Målet med smarthus er hovedsakelig å senke energiforbruk og kostnader, men det handler også om at boligen skal ha en viss intelligens. Dermed skal boligen for eksempel kunne beskytte beboerne ved å si ifra om en komfyr fortsatt står på eller at man har glemt å låse døra når man forlater huset. En ønsker også å utvikle intelligens for praksis og komfort. For eksempel ved å fjernstyre dagligdagse funksjoner som ellers tar tid manuelt.

1.2 Avgrensing

Vår oppdragsgiver, Hark Technologies, er et oppstartsselskap lokalisert i Trondheim. De er blant annet med på å skape løsninger for at smarthusteknologi skal bli tilgjengelig for ethvert hjem, uten større investeringer, ved å produsere en smartplugg. Med smartpluggen skal man få fordelene som eksisterer med smarthus, uten større investeringer eller større endringer i boligen. De eneste kravene som stilles til den som ønsker denne teknologien er å ha en datamaskin eller smarttelefon og tilgang til internett.

Smartpluggen er en enhet som skal gjøre dum teknologi smart. Den fungerer som et mellomledd mellom elektriske artikler og stikkontakten. I stedet for å koble en varmeovn eller lampe direkte i stikkontakten setter du heller pluggen i kontakten og bruker den som stikkontakt. Etter å ha gjort dette kan du kommunisere med pluggen. Du skal kunne slå den av og på, justere temperaturen hvis det som er koblet til den er en varmekilde, se status for pluggen, endre innstillinger og annet.

Figur 1: EcoHeat smartplugg

Kommunikasjonen vil foregå gjennom applikasjonene som skal utvikles i dette prosjektet.

1.3 Oppgavedefinisjon

Oppgaven går ut på å utvikle et visuelt styringssystem som gir oversikt over smartplugger man har i en bolig. Med dette systemet skal du kunne ha oversikt og styre enhetene hvor enn du er. Du vil kunne justere temperaturen i boligen fra sengen, skru av en kaffetrakter du glemte mens du er på jobb eller få beskjed om uforutsette økninger i strømforbruk mens du er på ferie. Styringssystemet skal lanseres som mobilplattform for Android og iOS, samt en løsning for web. En stor del av oppgaven er å gjøre brukergrensesnittet så intuitivt og brukervennlig at folk i alle aldre skal kunne ta i bruk applikasjonene. I tillegg ønskes det av oppdragsgiver å benytte en modulbasert oppbygging, slik at det skal være enkelt å kunne videreutvikle systemet.

Oppdragsgiver har under hele prosjektperioden oppmuntret til at gruppen skal komme med forslag til funksjonalitet for appen. Under er det en liste over mulig funksjonalitet for EcoHeat. Listen består av forslag som har kommet frem under diskusjon mellom oppdragsgiver og bachelorgruppen, hvor mange av disse punktene er en utvidelse av de opprinnelige forslagene til funksjonalitet i oppgaveteksten (se vedlegg J).

- Visning av både ute- og innetemperatur.
- Visning av strømforbruk og kostnader knyttet til dette, samt vise hvor mye du har spart ved å ta i bruk EcoHeat.
- Bruker skal kunne sette en ønsket innetemperatur som systemet automatisk justerer seg etter.
- Gi mulighet til å gruppere flere brukere logisk slik at flere får tilgang til styringssystemet i en bolig.
- Sette timere og/eller ha en kalenderfunksjon for å sette temperaturen på bestemte dager.
- Mulighet for å sette forskjellige moduser for en gitt bolig, hvor brukeren har mulig-

het til å konfigurere alle enhetene etter sitt behov. For eksempel feriemodus hvor bruker konfigurerer alle enhetene i huset, utenom varmeovner, til å skru seg helt av og at den gjennomsnittlige temperaturen senkes.
- Integrering av vekkerklokke på mobilapplikasjonen. Her vil du for eksempel kunne bestemme at hvis alarmen er satt til å vekke deg klokka syv vil systemet begynne å varme opp boligen en halvtime før. Også med mulighet for at systemet lærer hvor lang tid det tar å oppnå optimal temperatur, slik at strømforbruket er minimalt.
- GPS-lokasjon for mobil. Med dette kan for eksempel mobilapplikasjonen automatisk sette på varmen når den merker at du nærmer deg hjemmet ditt.
- Integrering med Google Calendar slik at systemet automatisk kan se når du er borte og justere boligene deretter.
- Systemet skal kunne omtrentlig vite når boligen er forlatt eller ikke (enten basert på GPS-lokasjon eller de automatiske timerene). På denne måten kan man forhindre ulykker med elektriske apparater ved at bruker får melding om at ting potensielt er forlatt påskrudd.

Det skal utvikles en databaseløsning som skal inneholde brukerinformasjon, informasjon om smartplugger i bruk og målingsdata som pluggene sender til serveren og den logiske sammenkoblingen mellom brukerne, husholdene og enhetene.

1.4 Målgruppe

1.4.1 Rapporten

Denne rapporten er skrevet for de som har interesser innenfor temaet smarthusteknologi, medstudenter (spesielt informatikkstudenter) og sensorene. I tillegg er rapporten et dokument som beskriver grundig omfanget av dette prosjektet fra start til slutt. Rapporten tar forbehold om at den som leser har noe teknologisk bakgrunn (helst på nivå med en tredjeårsstudent innenfor informatikk), men vil også i stor grad kunne leses av hvem som helst med hjelp av referanser, terminologiliste og forklaringer rundt omkring i rapporten.

1.4.2 Applikasjonene

Både web- og mobilapplikasjon er laget med tanke på at de skal kunne brukes av mennesker i alle aldre, men vil i hovedsak appellere til boligeiere i privat og offentlig sektor.

1.5 Formål

Formålet med dette prosjektet er først og fremst å tilby et styringssystem for elektroniske artikler for private hushold, med automatisering i fokus.

Bakgrunnen for oppgaven kommer fra studietiden til Joar Harkestad. Studentboligen hans hadde et høyt strømforbruk og for å redusere dette ble varmeovner og tilsvarende skrudd av om kvelden, eller når beboerene ikke var hjemme. Problemet var at det fort kunne ta en halvtime før boligen ble varmet opp til en behagelig temperatur om morningen eller når noen kom tilbake til boligen.

Lignende løsninger på markedet var ikke tilfredsstillende og han endte derfor opp med å lage sin egen. Dette resulterte i EcoHeat smartpluggen, men han trengte programvare for å styre pluggene og for at løsningen skulle bli tatt i bruk av andre. Dette ble presentert som denne bacheloroppgaven.

Vi valgte denne oppgaven fordi den var godt presentert og lagt opp som en potensiell tverrfaglig oppgave, med mulighet for å jobbe med fremtidsrettet teknologi. Oppdragsgiver opplevdes hyggelig og imøtekommende etter første telefonsamtale, som forsterket vår interesse for oppgaven. Vi så på god kommunikasjon med oppdragsgiver som viktig, basert på dårlige erfaringer i tidligere prosjekter.

1.6 Læringsutbytte

Gjennom dette prosjektet ønsket vi å enten oppnå eller øke vår kunnskap innenfor følgende temaer, men ikke begrenset til:

- Prosjektplanlegging.
- Benytte prinsippene om Scrum i et større prosjekt.
- Tilnærming til utvikling på en mer profesjonell måte.
- Arbeide som en gruppe over en lengre periode.
- React og React Native.
- Utvikling av applikasjoner med hensyn på brukervennlighet.

1.7 Akademisk bakgrunn

Bachelorgruppen består av tre informatikkstudenter: Henrik Solum, Jostein Enes og Tarjei Holtskog. Henrik og Jostein har fulgt Programvareutvikling (14HBPUA) og Tarjei (13HBIDATA) Dataingeniør ved NTNU i Gjøvik. Henrik byttet til Programvareutvikling høsten 2015 fra studieprogrammet Drift av nettverk og datasystemer ved NTNU i Gjøvik, mens Jostein og Tarjei har fulgt sine studieløp fra starten av.

1.7.1 Faglig bakgrunn

Gruppemedlemmene har gjennom studieløpene arbeidet noe med C++, men stort sett arbeidet i Java, Javascript og PHP. I tillegg har studiene vært borti flere forskjellige programmerings- og scriptespråk, som også har vært med på å øke kompetansen til gruppen videre. Faget Systemutvikling (IMT2243) har blant annet tatt opp viktig tematikk rundt 'profesjonell arbeidsmetodikk for utvikling av datasystemer og forståelse for grunnleggende administrative og teknologiske aspekter ved spesifisering, utvikling, innføring og vedlikehold av programvare.' [5].

Valgfag

I tillegg har gruppemedlemmene hatt valgfag, der noen av fagene har vært relevante for denne bacheloroppgaven. Jostein og Henrik har hatt faget Mobile development project (IMT3672) der hovedprosjektet omhandlet det å lage en Android applikasjon i Java. Tarjei har hatt Vitenskaplig programmering (IMT3881) der maskinglæring skulle være relevant for denne oppgaven, WWW-tekonologi med oppgaver som innebar utvikling i HTML, CSS, Javascript og PHP, og Applikasjonsutvikling som gav han erfaringer innenfor utvikling i Java. De to sistnevne fagene er obligatoriske fag for Programvareutvikling.

1.7.2 Manglede bakgrunnskunnskap

Fra den opprinnelige oppgaveteksten (se vedlegg J) var det et ønske fra oppdragsgiver at rammeverkene React og React Native skulle brukes for utviklingen av applikasjonene, som ikke hadde vært en del av studiene.

1.8 Rammer

1.8.1 Gjennomføring

Gruppen forpliktet seg til å holde de rammene som var gitt av universitetet med tanke på tidsfrister. Dette inkluderer tidsfristene for de tre statusrapportene underveis i prosjektet, og hovedinnleveringen 16. mai 2017. Når planlegging, prototyper, utvikling og testing skulle være fullført var opp til gruppemedlemmene, da oppdragsgiver ikke hadde noen øvrige krav til dette.

1.8.2 Arbeidsmetoder

I dette prosjektet har den agile systemutviklingsmodellen Scrum blitt brukt, som gjør at prosjektperioden har blitt delt opp i sprinter. De tre første sprintene var to uker lange, men etter dette ble det bestemt at lengden på de neste sprintene skulle reduseres til en uke. Scrum ble valgt fordi oppdragsgiver ønsket seg et modulært system, og denne utviklingsmodellen baserer seg på at hver sprint skal utvikle en eller flere moduler av systemet. I tillegg gir Scrum muligheten til å finne ut hva som skal utvikles underveis, fordi det er enkelt å legge til eller endre oppgaver under utviklingen.

Innhentingen av manglende kunnskap består i stor grad av undersøkelser på internett, hvor gruppen har blant annet gått igjennom kurs for React [6] og React Native [7] (kjøpt inn av oppdragsgiver), gjennomgang av dokumentasjon for forskjellige biblioteker og andre forum på internett.

1.8.3 Fremdriftsplan

Det ble tidlig åpenbart at fremdriftplanen (se vedlegg K - Kapittel 6.1 GANTT) for prosjektet ville være vanskelig å følge, med tanke på at gruppen fulgte prinsippene om sprinter fra Scrum. GANTT diagrammet ble heller en overordnet oversikt over de forskjellige fasene av prosjektet, der den opprinnelige planen ble i stor grad endret underveis. Noe av grunnen til dette er at det er vanskelig å estimere hva hver enkelt sprint kommer til å bestå av, siden disse sprintene blir laget underveis. I tillegg kommer det stadig opp nye uforutsette arbeidsoppgaver.

1.8.4 Prosjektorganisering

Tidlig i prosjektet bestemte gruppen seg for å møte på skolen mandag til fredag, så langt det lot seg gjøre. En slik tilnærming fungerte bra da alle medlemmene alltid hadde innsikt i hva de andre arbeidet med for tiden, og om det var uklarheter kunne man alltid diskutere problemet med en gang. Alle ble enig om å arbeide fra kl 08:15 til 15:00 Mandag til Torsdag, og fra 09:15-14:00 på fredager. Fredagen ble kortere grunnet en forelesning som startet 14:15. Det var også rom for å jobbe utenfor disse tidsrommene, men dette skjedde stort sett på frivillig basis, eller når det var høyst nødvendig. Alle notater, referater, diagrammer, timeføring og lignende ble lagret i Google Drive.

Det ble i tillegg utdelt roller innad i bachelorgruppen, slik at det alltid var noen som hadde hovedansvaret for de forskjellige fasene av prosjektet. Alle gruppemedlemmene ble tildelt rollen 'utvikler', ettersom det var enighet om at det ikke fantes nok administrativt arbeid for gruppelederen til å bare opptre som en gruppeleder. Under presenteres de spesifikke rollene hvert medlem hadde gjennom prosjektet. Rapportskriving og prosjektplanlegging var fellesoppgaver og blir ikke nevnt på de individuelle fremstillingene under. Det må også informeres om at alle var med på å hjelpe hverandre til tross for inndeling av hovedområder og ansvar.

Jostein ble valgt til gruppeleder/Scrum master for hele prosjektperioden, på bakgrunn av hans kunnskap fra valgfaget Organisasjon og ledelse (SMF1071). Dette innebar ansvaret for innkalling til møter og opprettelsen av de forskjellige sprintene under prosjektperioden. For å arbeide på skolen ble det bestemt at grupperom måtte bestilles til hver eneste dag, noe Henrik fikk ansvaret for.

Under planleggingen ble det også relevant å dele ut flere roller. Henrik hadde hovedansvaret for prototypen laget for mobilapplikasjonen, og Jostein den tilsvarende rollen for webapplikasjonen. Tarjei fikk derimot et større ansvar for uttredelsen av de forskjellige aspektene under kravspesifiseringen.

I utviklingsfasen ble det bestemt å ha et hovedfokus på mobilapplikasjonen, der Henrik og Jostein fikk ansvaret for implementeringen. Tarjei ble eneste utvikler med ansvaret for webapplikasjonen.

1.8.5 Øvrige roller

Oppdragsgiver

Oppdragsgiveren for dette prosjektet har vært Joar Gunnersjaa Harkestad og hans oppstartfirma Hark Technologies, lokalisert i Trondheim. Oppdragsgiver er i denne sammenhengen synonymt med Product Owner fra Scrum, og han bestemte i samarbeid med bachelorgruppen hvilke oppgaver som burde prioriteres underveis i de forskjellige sprintene. Joar har også bidratt med hyppige tilbakemeldinger, på møter, telefonsamtaler, tekstmeldinger og epost under hele prosjektet når det har vært nødvendig. Joar har som sagt vært gruppens kontaktperson i dette prosjektet, men mot slutten av mars fikk han flere inn på teamet sitt i Trondheim, der blant annet Stian Michael Årsnes har hjulpet med datastrukturer og Kieu-Van June Bui ble ansvarlig for utviklingen av server.

Veileder

Frode Haug har opptrådt som veileder for dette prosjektet. Han er universitetslektor ved NTNU i Gjøvik og underviser i mange av de obligatoriske fagene for alle studieretningene innenfor data. Han har hjulpet ved å gi tilbakemeldinger på det gruppen har skrevet av prosjektplaner, statusrapporter, diverse spørsmål fra gruppen og selve bachelorrapporten.

Testere av GUI prototypen

Eivind Arnstein Johansen har hjulpet til med å gi tilbakemelding på prototypene for mobil og web. Han er en universitetslektor ved bachelor i mediedesign NTNU i Gjøvik. Med hans hjelp har prototypene blitt forbedret med tanke på universell utforming, som er

viktig for dette prosjektet. Venner og familie har også bidratt med å prøve ut prototypene for brukergrensesnitt. Basert på tilbakemeldingene har prototypene blitt forbedret ytterligere.

Utviklingsmiljøer

Utviklingsmiljøer som Stackoverflow, GitHub og andre har også hatt en viktig rolle i dette prosjektet. Her er det mange som deler sine kodeeksempler og erfaringer og har vært til stor hjelp under utviklingen. I tillegg er det mange moduler og komponenter fritt tilgjengelig for andre å bruke.

1.9 Rapporten

1.9.1 Terminologi

Denne rapporten tar forbehold om at den som leser har en lignende bakgrunn innenfor IT som bachelorgruppen selv. Det vil si at begreper og uttrykk vil bli brukt rundt omkring i rapporten, hvor det nødvendigvis ikke alltid vil være forklart hva det betyr. Om det er kunnskap som har blitt opparbeidet i prosjektperioden, vil det bli forklart der det nevnes eller i vedlegget med listen over terminologi (se vedlegg A).

1.9.2 Praktisk

Denne bachelorrapporten er skrevet i LaTeX, et dokumentforberedelsessystem. LaTex er et system som fokuserer på dokumentets utseende og vil derfor spare brukeren for tid knyttet til estetisk arbeid med rapporten. I tillegg har en LaTeX-mal for bachelor- og masteroppgaver ved NTNU blitt benyttet, påbegynt av Erik Hjelmaas og Ivar Farup og videreutviklet av Simon McCallum slik at malen er nyttig for NTNU. Malen bestemmer alt av fonter, skriftstørrleser og egentlig alt som har med utseende av rapporten å gjøre.

1.9.3 Struktur

Kapittel 1 Innledning

I dette kapitlet kan du lese om bakgrunnen og formålet med prosjektet. Det er i tillegg skrevet om hvordan vi tilnærmet oss prosjektet, med tanke på planlegging, inndeling av roller og hva vi ønsket å lære av dette prosjektet.

Kapittel 2 Kravspesifikasjon

Dette kapitlet tar for seg de funksjonelle og supplementære kravene til den endelige løsningen.

Kapittel 3 Analyse

Her står det om hvilke vurderinger som ble tatt under valg av forskjellige teknologier, hvordan brukerhåndtering skal gjennomføres og hvilken databaseform som man endte opp med.

Kapittel 4 Design og arkitektur

Dette kapitlet går igjennom hvordan utviklingen av de forskjellige brukergrensesnittene har foregått, med en god del figurer og forklaringer bak disse. Arkitekturen rundt forskjellige aspekter av systemet er også beskrevet her.

Kapittel 5 Realisering og implementasjon

Dette kapitlet tar for seg hvordan vi gikk frem for å realisere og implementere applikasjonene og teknologi som ble benyttet for utviklingen.

Kapittel 6 Testing og kvalitetssikring

Testing og kvalitetssikring handler om de tiltakene som har blitt tatt for å produsere en best mulig løsning.

Kapittel 7 Avslutning

Til slutt evaluerer vi prosjektet i sin helhet med drøftinger rundt valg som har blitt tatt underveis, refleksjon, hvilket arbeid som står igjen og til slutt vår konklusjon.

Vedlegg

I vedleggene finnes alt av terminologi, møtereferater, arbeidslogger, alt av dokumenter skapt for prosjektplanleggingen, designdokumenter og oppgaveteksten som introduserte bacheloroppgaven.

2 Kravspesifikasjon

For at EcoHeat skal realiseres stilles det både funksjonelle og supplementære krav. De funksjonelle kravene er et resultat av prosjektets Product Backlog, opprettet tidlig i prosjektet, og har blitt utvidet ved behov. Naturligvis har Product Backlog endret seg underveis, med tanke på at Scrum har blitt brukt som utviklingsmodell for dette prosjektet. De supplementære kravene har kommet frem under diskusjon rundt hvilken funksjonalitet det endelige systemet skal tilby og hvilke teknologier som kreves for å støtte opp under dette.

2.1 Endring av oppgaven

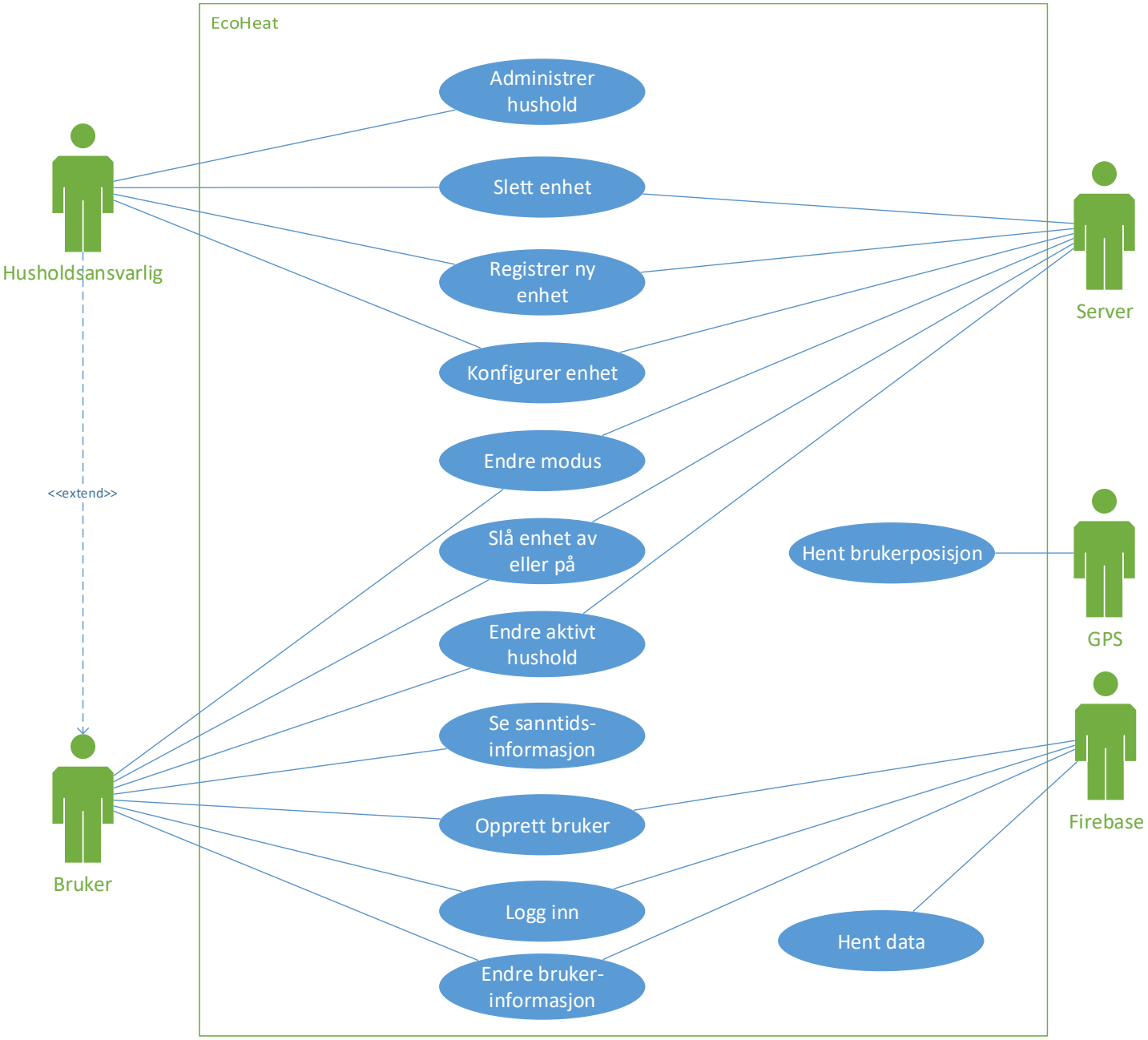
Opprinnelig skulle oppgaven omhandle utvikling av server og klient (web, iOS og Android). I et møte med oppdragsgiver mot slutten av mars (se vedlegg O) kom det frem at oppdragsgiver hadde fått inn folk til å ta seg av utviklingen av server for prosjektet. Dermed ble omfanget av det som skulle utvikles noe redusert, selv om utviklingen for server opprinnelig var tatt i betraktning under planleggingen for prosjektet.

2.2 Funksjonelle krav

I bunn og grunn er EcoHeat en applikasjon for å redusere strømforbruk ved å kunne kontrollere elektroniske artikler via internett (mobil og webgrensesnitt), manuelt eller automatisert ut i fra brukerens konfigurasjon. Systemet må ha mulighet for å registrere brukere av sikkerhetsmessige og administrative grunner.

Applikasjonen må gi brukerne muligheten til å registrere og styre enhetene sine, administrere hushold og få tilgang til sanntidsstatistikk for sine hus/boliger. For at EcoHeat skal redusere strømforbruk må applikasjonen la brukeren sette opp tidsrom for når de forskjellige enhetene i et hushold skal være på- eller avskrudd. Et eksempel på dette kan være at brukeren setter opp en modus som sier at gjennomsnittstemperaturen i huset kan senkes og all belysning skal være skrudd av, når huset er satt i 'borte' modus. Da kan brukeren fortelle EcoHeat at dette moduset skal være gjeldende hver ukedag fra når brukeren forlater huset og drar på jobb til brukeren kommer hjem.

2.2.1 Use case modell



Figur 2: Use Case diagram for EcoHeat

2.2.2 Overordnet use case beskrivelse

Tabell 1: Høynivå use cases

Navn	Opprett bruker
Aktør(er)	Husholdsansvarlig, Bruker, Firebase
Formål	Gi nye brukere tilgang til EcoHeat.
Beskrivelse	For å ta i bruk EcoHeat påkreves det at man har en bruker. Dette gjøres i en av applikasjonene hvor bruker registrerer detaljene sine og dette sendes til Firebase som validerer og deretter sender en bekreftelse/feilmelding tilbake. Hvis ingen feil oppstår så vil brukeren bli logget inn.

Navn	Logg inn
Aktør(er)	Husholdsansvarlig, Bruker, Firebase
Formål	La brukeren logge inn.
Beskrivelse	Har man allerede en bruker kan man logge inn i applikasjonen. Da vil det sendes en forespørsel til Firebase med brukernavn og passord og på samme måte som med "Opprett bruker"vil man få en respons fra Firebase med beskjed om at man enten ble logget inn eller en feilmelding om noe gikk galt.

Navn	Endre brukerinformasjon
Aktør(er)	Husholdsansvarlig, Bruker, Firebase
Formål	La brukeren endre brukerinformasjonen sin.
Beskrivelse	Eksisterende brukere kan sende en forespørel til Firebase om å endre informasjon om seg selv, som for eksempel email, passord og navn.

Navn	Se sanntidsinformasjon
Aktør(er)	Husholdsansvarlig, Bruker
Formål	Kunne kjapt se status for husholdet.
Beskrivelse	Brukeren skal ha mulighet til å kunne se oppsummert informasjon for å få et generelt overblikk over husholdet, men det skal også være mulig å få mer detaljert informasjon også. Slik kan brukeren kjapt se hvordan det står til med husholdet i sanntid. Fremvisningen vil skje i form av tekst, grafer og ikoner.

Navn	Endre aktivt hushold
Aktør(er)	Husholdsansvarlig, Bruker
Formål	Endre hvilket hushold som er aktivt for øyeblikket.
Beskrivelse	Brukeren skal kunne velge mellom alle de forskjellige husholdene som brukeren tilhører. Når husholdet endres vil applikasjonen hente ned dataene fra det nye husholdet, se use case 'Hent data', og vise frem disse til brukeren.

Navn	Endre modus
Aktør(er)	Husholdsansvarlig, Bruker, Server
Formål	Sett husholdet i et annet modus.
Beskrivelse	Brukeren skal kunne sette husholdet til å samsvare med en av de predefinerte modusene. Det vil si at alle enhetene gås igjennom og skrus av eller på, avhengig av hvordan brukeren har konfigurert de forskjellige enhetene i moduset som ble satt. På denne måten kan man kjapt få husholdet sitt i 'feriemodus' slik som man selv har konfigurert det, istedet for å måtte endre enkeltenheter hver eneste gang.

Navn	Registrer ny enhet
Aktør(er)	Husholdsansvarlig, Server
Formål	Legg til en ny enhet i et hushold.
Beskrivelse	En bruker med de riktige tillatelsene skal ha muligheten til å registrere en ny enhet. Serveren vil registrere enheten i EcoHeat-systemet, innenfor det husholdet som ble spesifisert og alle brukerne innenfor dette husholdet vil kunne se og kontrollere enheten via applikasjonen.

Navn	Konfigurer enhet
Aktør(er)	Husholdsansvarlig, Server
Formål	Konfigurer en enkelt enhet.
Beskrivelse	Konfigurasjon av en enhet vil innebære å bestemme om enheten skal være på eller avskrudd i de forskjellige modusene. Om enheten også kan regulere temperatur, vist den for eksempel er en varmeovn, vil man også kunne sette ønsket makstemperatur omkring enheten, for de forskjellige modusene. Endringene sendes til server som justerer enheten deretter.

Navn	Slett enhet
Aktør(er)	Husholdsansvarlig, Server
Formål	Slette en enkelt enhet fra systemet
Beskrivelse	En bruker med de riktige tillatelsene skal ha mulighet til å fjerne enheter fra et gitt hushold.

Navn	Slå enhet av eller på
Aktør(er)	Husholdsansvarlig, Bruker, Server
Formål	Slå en enkelt enhet av eller på.
Beskrivelse	Alle brukere kan skru av og på de enkelte enhetene i boligen og dermed overstyre moduset som er blitt satt. Slik vil det holde seg helt til en bruker eventuelt endrer modus igjen.

Navn	Administrer hushold
Aktør(er)	Husholdsansvarlig
Formål	Endre innstillingene for et hushold.
Beskrivelse	De som er registrert som ansvarlige for et hushold har mulighet til å endre navn, bilde, adresse og tilsvarende for husholdet.

Navn	Hent data
Aktør(er)	Firebase
Formål	Hente husholdsdata fra Firebase.
Beskrivelse	Applikasjonen vil inneha data om det angitte husholdet for oppdatering av statistikken til husholdet. Applikasjonen vil be om å få oppdatering av dataene når det skjer endringer i applikasjonen.

Navn	Hent brukerposisjon
Aktør(er)	GPS
Formål	Hente brukerposisjonen for automatisering.
Beskrivelse	Mobilapplikasjonen vil prøve å hente brukerposisjonen slik at den kan se hvorvidt brukeren er nær hjemmet sitt eller ikke og kan dermed automatisk skru enheter av eller på ut i fra brukerens konfigurasjoner.

2.2.3 Detaljerte use case beskrivelse

Navn	Opprett bruker
Aktør(er)	Husholdsansvarlig, Bruker, Firebase
Forhånds-betingelser	Bruker må ha en av applikasjonene tilgjengelig.
Detaljert hendelsesforløp	1. Bruker navigerer til registreringssiden. 2. Bruker registrerer brukerinformasjon. 3. Applikasjonen sender en forespørsel til Firebase med brukernavn og passord for å registrere brukeren. 4. Firebase sender en bekreftelse tilbake. 5. Brukeren logges inn i applikasjonen.
Variasjoner	Firebase kunne ikke registrere brukeren, sender en feilmelding tilbake til bruker.

Navn	Slå enhet av eller på
Aktør(er)	Husholdsansvarlig, Bruker, Server
Forhånds-betingelser	Bruker er innlogget.
Detaljert hendelsesforløp	1. Bruker navigerer til 'Mine enheter' 2. Bruker trykker på knappen for å skru av eller på en gitt enhet. 3. Applikasjonen sender en forespørsel til serveren med den unike identifikatoren for denne enheten. 4. Enhetens status oppdateres i applikasjonen.
Variasjoner	Enheten har blitt fjernet og bruker vil få en feilmelding om at enheten ikke ble funnet i systemet.

2.3 Supplementære krav

2.3.1 Systemkrav

Mobilplattformer

Skal man utvikle en mobilapplikasjon idag står valget mest sannsynligvis mellom Android og iOS da det er disse operativsystemene som dominerer markedet for smarttelefoner. Ifølge International Data Corporation var Android ledende med 86.8% mens iOS sto for 12.5% av markedsandelen for smarttelefoner i tredje kvartal 2016 [8]. Dette tilsvarer da 99.3% av det totale smarttelefon markedet. Med denne enorme andelen av

Period	Android	iOS	Windows Phone	Others
2015Q4	79.6%	18.7%	1.2%	0.5%
2016Q1	83.5%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%
2016Q3	86.8%	12.5%	0.3%	0.4%

Figur 3: Markedsandeler smarttelefoner 2015Q4-2016Q3 [8]

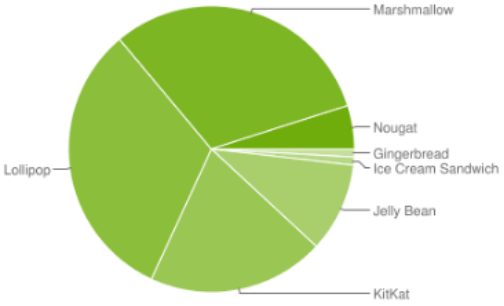
smarttelefonmarkedet er det forsvarlig å utvikle applikasjonen for enten Android eller iOS. Utvikling for iOS er nok enklere i og med at det finnes mange flere variasjoner av smarttelefoner for Android enn iOS. På den andre siden er Android sin plattform åpen, hvor applikasjoner kan lastes opp uten godkjenning ifra Google.

I samarbeid med oppdragsgiver ble det bestemt at applikasjonen skulle utvikles i React Native, som skal fungere på både Android og iOS, med hovedfokus på Android. I dette prosjektet ble det prioritert å få alt til å fungere på Android først og fremst.Ett problem med å bruke React Native er at de ikke støtter Android versjoner lavere enn 4.1 [9], men som figur 3 viser vil dette tilsvare 98.2% av Android enhetene i verden og ansees knapt som et problem.

Webplattformer

Utvikling for web blir i stor grad forskjellig fra det å utvikle for en mobilplattform. Dette er fordi web i seg selv er i større grad uavhengig av operativsystemet under, så lenge det finnes en nettleser. Man kan påstå at web i seg selv er en plattform og det er dermed ikke mange muligheter for hva man skal velge mellom. For å utvikle webapplikasjonen skal det blitt tatt i bruk React, da dette var naturlig ettersom det var enighet om at React Native skulles brukes for mobilløsningen. React er et rammeverk for Javascript og ettersom Javascript støttes i stor grad av alle moderne nettlesere nå til dags er det også få brukere som er ekskludert fra å bruke applikasjonen basert på valg av plattform eller operativsystem.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.9%
4.1.x	Jelly Bean	16	3.5%
4.2.x		17	5.1%
4.3		18	1.5%
4.4	KitKat	19	20.0%
5.0	Lollipop	21	9.0%
5.1		22	23.0%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	4.5%
7.1		25	0.4%



Figur 4: Global distribusjon av Android versjoner April 2017 [10]

2.3.2 Ytelse

For både mobil og web er det flere tiltak som må til for innlasting skal skje raskt og at minimalt med ressurser blir brukt til dette. Til og begynne med må det implementeres systemer for å komprimere og redusere bildestørrelser. Ettersom applikasjonen vil håndtere bilder og eventuelle andre ressurser, må det implementeres metoder for å redusere disse til en mer passende filstørrelse. Om brukerene tar bilder med sine mobiltelefoner er det tvilsomt at applikasjonen trenger hele bildet i den originale størrelsen. Dermed kan bildene heller reduseres til å være et miniatyrbilde eller avskjære deler av bildet som ikke er relevant i forhold til innholdet. Her vil det også være relevant å implementere dette slik at brukeren selv kan velge hvilke deler av bildet som skal beholdes.

Et annet tiltak for begge applikasjonene i forbindelse med ytelse vil være caching. Dette kreves for at applikasjonene ikke skal se tomme ut, om nettverkstilkoblingen kuttes eller er utilgjengelig. Det må også gis tilbakemelding til brukeren om at innholdet er cachet, slik at det ikke virker negativt på brukeropplevelsen om enkelt funksjonalitet ikke er tilgjengelig før nettverkstilkoblingen er gjenopprettet. Caching gjør også at selv om applikasjonen er koblet til internett, kan det fortsatt vises frem noe utdatert informasjon, mens applikasjonene sjekker om det finnes ny informasjon å vise frem i stedet.

Til sist er det viktig å gi brukerne tilbakemelding hver gang applikasjonen venter på en eller annen innlasting av informasjon. Selv om det ikke er direkte ytelse, vil det virke som om applikasjonen har hengt seg opp, dersom brukeren ikke får noe respons. Dessuten kan det argumenteres for at brukere er mer villige til å vente dersom det vises frem en indikasjon om at det pågår arbeid i bakgrunnen, i stedet for at applikasjonen er uendret under innlasting.

Mobil

Mer spesifikt for mobil er det begrenset med prosesseringskraft og batterilevetid, som gjør at alle ressursene må utnyttes på best mulig måte. Målet for applikasjonen er å skape en tynn klient, det vil si at mobilen skal gjøre minst mulig, mens serveren tar seg av det meste som trengs av applikasjonslogikk. Klienten tar i stor grad å ber om JSON formatert data som den kan fylle sine skjermer med. Dette gjør at applikasjonen er i stor grad avhengig av internett tilgang.

2.3.3 Internasjonalisering

Selv om det ikke var et krav fra oppdragsgiver, skal applikasjonen for både mobil og web bli satt opp for å bli fullstendig internasjonalisert tidlig i prosjektet. Dette må gjøres med tanke på at det er mye enklere å sette opp dette på forhånd og kontinuerlig oppdatere de forskjellige oversettelsesfilene, enn det er å implementere et slikt system i etterkant. Applikasjonene må settes opp til å finne ut hvilket språk telefonen/nettleseren er satt til å bruke og automatisk ta i bruk dette språket. Om språket ikke er støttet må applikasjonen ta i bruk engelsk. Til å begynne vil det være fokus på å ha støtte for både engelsk og norsk i de forskjellige applikasjonene, men det skal være mulig med minimale endringer i kildekoden for å få inkludert nye oversettelsesfiler.

2.3.4 Brukervennlighet

Helt fra starten av prosjektet har det vært klart at brukervennlighet må være en av hovedprioriteringene ved utviklingen av en mobilapplikasjon. I den opprinnelige oppgaveteksten (se vedlegg J) for denne bacheloroppgaven sto det som følger:

> Målet er å utvikle appen samt et brukergrensesnitt som er så intuitivt og enkelt at det blir en god opplevelse for brukeren, samtidig som det er lett forståelig både for eldre og unge. (...)

Et forslag fra oppdragsgiver var å gå igjennom eksisterende løsninger, for å se hva som er bra og hva som kunne ha blitt gjort bedre. På denne måten kan alt det som ansees som bra tas med videre og det kan føres diskusjoner på det som er mindre bra og hvorfor det ikke er en god løsning. Et av hovedmålene var at applikasjonsdesignet skal være gjenkjennbar, slik at det skal være enkelt for nye brukere å ta i bruk applikasjonen. EcoHeat skal også være en applikasjon mennesker i alle aldre skal kunne ta i bruk, der kommunikasjon via tekst og ikoner må være tydelig. Et annet mål er at enhver destinasjon skal nås med minst mulig interaksjon fra brukeren, som vil si at funksjonalitet som brukes hyppig aldri skal være langt unna samme hvor brukeren befinner seg i applikasjonen. Det er også viktig at følgende krav rundt universell utforming følges:

- Bruk av farger i henhold til fargeblindhet og svaksynte, i tillegg til miljøet rundt brukeren (sollys, mørke, osv.) Dette inkluderer også sterke kontraster.
- Store trykkbare flater for de med redusert motoriske evner.
- Alt-tekst til bilder slik at svaksynte og blinde kan fortsatt navigere og bruke applikasjonen ved at denne teksten kan leses opp av hjelpeverktøy.

Mobil

Størrelsen på en mobil stiller ekstra krav til brukervennlighet. Du skal kunne navigere rundt i applikasjonen med én hånd så ting kan ikke være for langt unna hverandre. Samtidig skal applikasjonen være oversiktlig så man kan heller ikke presse funksjonalitet for tett sammen. Alle flater skal også være lett klikkbare og tilpasset for en touch-skjerm.

Web

Webapplikasjonen skal i stor grad likne på mobilapplikasjonen. På den måten skal brukerne kunne kjenne seg igjen i det forskjellige plattformene når det kommer til design og oppsett. Samtidig har man på web, fordelen av en langt større skjerm enn for mobil og dermed har man også flere muligheter. Det ble derfor foreslått at webapplikasjonen skulle fungere som en utvidet versjon av mobilapplikasjonen med blant annet grafisk fremvisning av måledata.

2.3.5 Sikkerhet

Ettersom applikasjonen omhandler styring av en hel rekke elektroniske artikler hos privatpersoner, er det viktig at oppkoblingen mot disse er sikker og at uønskede ikke får tilgang til en annen brukers enheter. For å tilfredsstille dette kravet om sikkerhet, påkreves det autentisering med brukernavn og passord opp mot prosjektets Firebase server, et Google produkt som tilbyr autentiseringsmuligheter, database, datalagring, og annet. [11]. Her er det satt opp slik at brukeren må ha tillatelse til å endre tilstanden for en enhet eller for å endre husmodus, for at endringen skal tre i kraft. Det må kreves også en viss lengde på passordene til brukerne.

3 Analyse

Ut i fra den originale oppgaveteksten og møter med oppdragsgiver har man fått god innsikt i det ønskelige resultatet for dette prosjektet. Oppdragsgiver har blant annet fortalt om hvor ideen for dette prosjektet kom fra og delt sin kunnskaper om lignende løsninger som er på markedet idag. I tillegg har oppdragsgiver sendt maskinvare fra disse løsningene, slik at gruppen også kunne diskutere sine erfaringer med oppdragsgiver. Ettersom systemet skulle utvikles fra bunnen av, var det viktig å sette seg godt inn i hvordan de lignende løsningene fungerte.

3.1 Valg av teknologier

For å realisere prosjektet kreves det en del forskjellige teknologier. Som nevnt i 2 Kravspesifikasjonen må systemet blant annet inkludere:

- Applikasjon for både Android, iOS og nettlesere generelt.
- Opprettelse av brukere og sikker brukerhåndtering.
- Datalagring hvor data settes inn hyppig og oppdateres med jevne mellomrom, der dataene består av både ren tekst og bilder. Det vil også være målingsdata som vil bli tatt i bruk i grafer.
- Sikker kommunikasjon mellom den forskjellige maskinvaren som utgjør systemet.

3.1.1 Utvikling av applikasjonene

For å utvikle en applikasjon for mobil og web, måtte det tas en vurdering av hvilke språk og rammeverk som dekker behovene til applikasjonene. En mulighet for å utvikle applikasjonene var å utvikle i Java for Android, i Objective-C/Swift for iOS og webapplikasjonen ved hjelp av HTML, Javascript og CSS. Basert på erfaringene til utviklerne ville dette betydd at alle måtte læres opp i Objective-C og alle utviklerverktøyene for iOS. I tillegg måtte en av utviklerne også bli lært opp i utvikling for Android, ettersom to av tre allerede har erfaring med dette. En annen ting som er verdt å nevne er at om denne fremgangsmåten hadde blitt valgt måtte det utvikles tre individuelle applikasjoner hvor alle applikasjonene i bunn og grunn gjør akkurat det samme, bare på tre forskjellige plattformer, språk og eventuelle rammeverk. Dette øker kompleksiteten til systemet ved at man må holde de forskjellige applikasjonene konsistente med hverandre. Samtidig ville samarbeid på tvers av de forskjellige plattformene blitt vanskeligere, da man må stadig vekk må hoppe mellom de forskjellige utviklingsmiljøene. Det ville også vært vanskelig for utviklerne å fordype seg i alle de forskjellige plattformene.

En annen fremgangsmåte hadde vært å utviklet en fullstendig responsiv webapplikasjon. Det vil si at applikasjonene kunne aksesseres fra enhver nettleser, enten om det er på telefon, tablet, laptop eller stasjonær datamaskin. Fordelen med denne fremgangsmåten er at utviklerne bare trenger å forholde seg til utviklingen av én løsning og brukerne kan nå applikasjonen på et fast sted. Utfordringen med denne fremgangsmåten er å få løsningen til å virke som om den hører til på de forskjellige plattformene, da samhandlingen

med applikasjonene vil ha forskjeller og et annerledes brukergrensesnitt. I tillegg kreves det tilgang til internett for at en slik applikasjon skal være nyttig der en native løsning fortsatt kunne fungert, bare med redusert funksjonalitet.

En tredje tilnærmingsmåte for dette prosjektet var å ta i bruk React og React Native, som oppdragsgiver hadde et ønske om. React er et rammeverk for å bygge brukergrensesnitt for webapplikasjoner og React Native er det tilsvarende for native mobilapplikasjoner. Ettersom det var et ønske fra oppdragsgiver, syntes gruppen at det måtte tas en grundig vurdering av denne tilnærmingen. Dette ble gjort via forskjellige kurs for React [6] og React Native [7] på nettsiden udemy.com og det opplevdes som veldig lovende. Her får man ikke fordelen med at det bare er et miljø å forholde seg til som med responsive nettsider, men likheten i tankegang bak React og React Native er alikevel stor. Utvikling i React Native gjør også at applikasjonen vil fungere for både iOS og Android, slik at man slipper å bekymre seg for å lage en mobilapplikasjon for begge operativsystemene. I tillegg syntes gruppen at dette var en spennende og utfordrende tilnærming til oppgaven og endte dermed opp på denne fremgangsmåten.

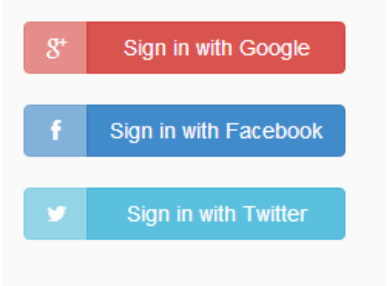
3.1.2 Brukerhåndtering

Når det kommer til brukerhåndtering finnes det også flere muligheter. Først og fremst kan man utvikle et skreddersydd system, der man selv har alt ansvar for private data og at passord blir lagret på en sikker måte. Fordelen med denne tilnærmingen er at man ikke blir avhengig av en tredjepart og om problemer skulle oppstå har man muligheten til å rette opp i det. Nedetid vil for brukeren oppleves negativt selv om det ikke er en feil med applikasjonen, for det er ikke brukerens ansvar å sette seg inn i hvilke systemer applikasjonen er avhengig av. På den andre siden krever brukerhåndtering en god forståelse angående krav som settes rundt datalagring. For det er vanskelig å vinne tilbake tilliten fra brukerne om dataene lekkes på grunn av sikkerhetshull eller andre årsaker.

Dessuten finnes det muligheter for å kunne koble applikasjoner opp mot store selskaper som Google, Facebook eller Twitter og heller la disse selskapene ta seg av alt som har å gjøre med brukerhåndtering. Hvorfor prøve å finne opp hjulet på nytt? Det er en selvfølge at disse selskapene har mange flere ansatte dedikerte for å holde denne informasjonen sikker, enn det EcoHeat vil ha i nærmeste fremtid med tanke på at disse selskapene håndterer millioner, kanskje til og med millarder av brukere. Et ofte brukt uttrykk er at det finnes 'Safety in numbers', som i denne sammenhengen vil si at vi kan stole på at disse selskapene vil gjøre sitt ytterste for å ikke lekke slik informasjon, da dette ville utvilsomt være ødeleggende for omdømmet deres. I følge statista.com har Facebook omtrentlig 2 milliarder brukere, Google 1 millard bare på YouTube, og Twitter 319 millioner brukere i April. 2017. [12]. Med milliarder av brukere kan man påstå at det er mindre sannsynlighet for nedetid for disse selskapene enn det er for EcoHeat, av samme grunn som at det er fordelaktig å la en slik tredjepart ta seg av passordlagring. Det er rett og slett et mye større selskap, med et ansvar for kolossale brukermasser, som naturligvis vil ha mange flere eksperter på både drift og sikkerhet. Et annet poeng er også at brukerhåndtering ikke ansees som kjernefunksjonaliteten til applikasjonen, noe som gjør at utviklingen av et slikt system ikke er forsvarlig med tanke på tiden det ville tatt. Spørsmålet da er hvilke alternativer det finnes der ute og hvilke fordeler og ulemper de

forskjellige alternativene har.

Til å begynne med ble Google, Facebook og Twitter vurdert som innloggingsalternativer til EcoHeat, fordi at det er ofte disse selskapene man ser blir brukt som alternativer for andre løsninger. Alle disse selskapene har sine API for å kunne logge inn via deres systemer, se referansene for Facebook [14], Twitter [15] og Google [16], men det finnes et hav av forskjellige nettsider som tilbyr det samme. Det er også store likheter i hva de forskjellige selskapene tilbyr. I et blogginnlegg av Jennifer Nash på nettsiden crescentinteractive.com [17] tar hun opp en del fordeler og ulemper med å ta i bruk såkalte 'sosiale logins', kort forklart er det innlogging til en egen løsning via en tredjepart. Fordelene hun nevner er først og fremst at brukeren slipper å lage en egen bruker bare for denne løsningen, som både gjør det lettere for brukeren og for løsningen fordi det blir færre forespørsler om å tilbakestille passord. Sosiale logins gjør også at man kan slippe å skrive inn brukernavn og passord på mobil, der brukeren muligens har applikasjonen for disse sosiale mediene nedlastet. Slik kan man heller bli omdirigert til den relevante applikasjonen, hvor man bare må trykke på en knapp for å godkjenne innloggingen. I tillegg kan applikasjonen få tilgang til en god del brukerdata fra det sosiale mediet, som kan brukes til å personliggjøre applikasjonen. På den andre siden nevner hun ulemper som mindre kontroll og sikkerhet, oppetid og unøyaktig brukerinformasjon, men som nevnt over burde disse systemene være mer sikre enn det EcoHeat i seg selv er, når mengden brukere har slike ekstreme forskjeller. Hun nevner i tillegg at det kan bli distraherende om det er for mange muligheter for innlogging og at det ikke er sikkert at en EcoHeat bruker har en konto hos noen av de valgte sosiale innloggingsmulighetene.



Figur 5: Sosiale innloggingsmuligheter. [13]

En bedre løsning ville da vært å tilby innlogging via flere av de vanligste sosiale mediene og i tillegg tilby standard registrering med brukernavn og passord som bare er spesifikk for EcoHeat. Med en slik tilnærming blir ikke EcoHeat sterkt knyttet til ett sosialt medie og i tillegg får hver enkelt bruker muligheten til å ta i bruk akkurat den løsningen som passer dem best. For å få til dette falt det endelige valget for brukerhåndtering på Firebase.

Firebase gir muligheten til å opprette en bruker som er spesifikk for løsningen, i tillegg kan man logge inn via Google, Facebook, Twitter eller Githu. Dette dekker mer enn nok av hva applikasjonen kommer til å trenge av innloggingsmuligheter. Firebase har også innebygd funksjonalitet for email verifikasjon, tilbakestilling av passord og endring av email for en gitt konto, ved å sende ut mail til den relevante brukeren. Når man bruker Firebase trenger ikke EcoHeat å bekymre seg med hvordan brukerinformasjonen lagres. Firebase tilbyr også datalagring i form av JSON databaser, som bringer oss videre til det

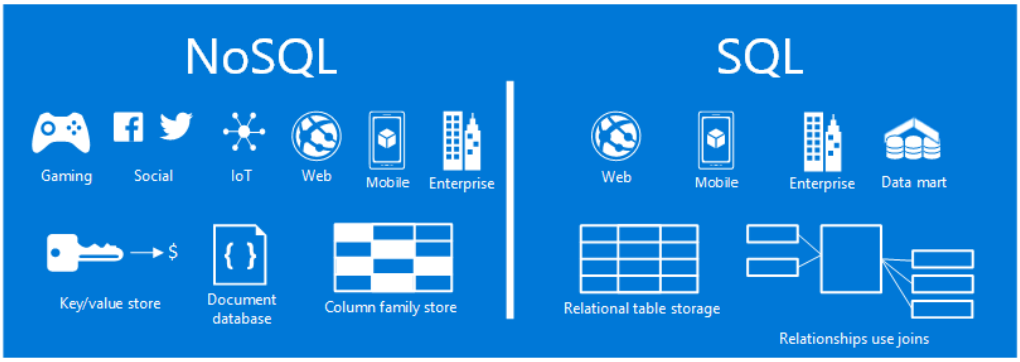
neste delkapitlet.

3.1.3 Datalagring

De fleste applikasjoner trenger på ett eller annet tidspunkt å lagre unna data for senere bruk og EcoHeat er intet unntak. I innledningen til dette kapitlet ble det nevnt at datatypene ville bestå av både tekst og bilder. I tillegg må løsningen som brukes kunne håndtere at det ofte kommer ny data inn i systemet. Grunnen til dette er at smartpluggene som er en del av EcoHeat, kommer til å rapportere målingsdata for temperatur med jevne mellomrom og det betyr at det er stadig nye data som må settes inn i databasen. EcoHeat vil også ha bruk for lagring av bilder, ettersom det endelige produktet skal gi brukeren mulighet til å tilpasse de forskjellige enhetene og husholdene sine med bilder de har tatt selv. Den første vurderingen som må tas er om man skal ta i bruk NoSQL eller SQL (RDBMS).

NoSQL vs. SQL

I den opprinnelige databasestrukturen for EcoHeat, se vedlegg D, ble SQL og relasjonsdatabaser tatt i bruk som utgangspunkt da det var denne tilnærmingen som var mest kjent for hele gruppen. Problemet var at det fort ble klart det måtte kjøres flere spørringer for å få ut enkel informasjon fra strukturen. For å finne ut hvilke enheter en enkelt bruker har, måtte man først ha funnet ut hvilke hushold brukeren er et medlem i, deretter spørre om alle rommene i dette husholdet, for så å spørre om hvilke enheter disse rommene har. Vedlegget viser strukturen på et tidlig punkt i prosjektet og i ettertid ville det nok ha inkludert enda en tabell for å vite hvilke 'Area' en 'Device' tilhører. Poenget er at en slik spørring ville ha krevd sammenslåing av fire eller flere tabeller, bare for å få brukerens enheter listet ut. Derfor kom diskusjonen angående NoSQL opp mot SQL frem. I Microsoft Azure sine dokumentasjonssider er det skrevet en lettleselig oppsummering og laget en figur (figur 6) av de forskjellige databasetilnærmingene satt opp mot hverandre. [18]



Figur 6: NoSQL vs SQL oppsummert[19]

Slik oppsummeringen forklarer er det naturligvis likheter og forskjeller mellom de forskjellige tilnærmingene. Fordelene med NoSQL over SQL er at all data er lagret i JSON objekter, noe som naturligvis er ypperlig for en applikasjon skrevet i Javascript. NoSQL gir også muligheten til å kunne legge til nye egenskaper til objekter kjapt og enkelt og det er ikke noe krav i NoSQL at objekter må ha de samme egenskapene til å begynne med. Dataene trenger heller ikke å være nøyaktig strukturert. Dette bruker EcoHeat i og

med at enheter kan være både elektroniske artikler som regulerer temperatur med ekstra konfigurasjonsmuligheter og vanlige enheter uten denne muligheten. Når det kommer til ytelse i form av oppdatering av en rad, vil et typisk objekt fra NoSQL ha tilgang på all informasjon om seg selv, mens i en SQL database ville en slik oppdatering måtte gå gjennom alle tabeller som referer til denne raden for å være konsistent. Til slutt nevnes det at det er enklere å skalere et NoSQL databasesystem over flere servere, da restriksjonene er løsere. NoSQL traff kravene til prosjektet i større grad enn det SQL gjorde og i tillegg har Firebase, som allerede brukes for brukerhåndteringen i applikasjonen, en innebygd databasestruktur som tar i bruk NoSQL. Derfor var dette det naturlige valget og den endelige datastrukturen kan du finne i vedlegg E.

4 Design og arkitektur

Basert på analysen kan designet og arkitekturen for de forskjellige applikasjonene begynne å ta form. Dette kapitlet tar for seg utformingen av brukergrensesnittene og beskrivelsene av arkitekturen for de forskjellige fysiske og logiske komponentene av løsningen.

4.1 Design

En viktig del av oppgaven har gått ut på å designe et brukergrensesnitt som er enkelt og forståelig for brukere i alle aldre. For å nå dette målet har det blitt satt av mye ressurser i starten av prosjektet. Disse ressursene er blitt brukt til å lage prototyper av brukergrensesnittet til de forskjellige versjonene av applikasjonen som har blitt utviklet. En av tilnærmingene for å lage en god løsning har vært å studere lignende løsninger, slik at det som fungerte i disse løsningene kunne være til inspirasjon for å skape et bedre produkt. Twitter, Facebook, Youtube og Twitch sine mobilapplikasjoner har også blitt sett på for å få en oversikt over hva brukermassene er vant med av forskjellige designløsninger. Ved å ta inspirasjon fra disse løsningene blir det lettere å designe en løsning brukerne kan kjenne igjen.

En annen viktig del av designet er at det skal være lett å navigere i løsningen med minst mulig interaksjon fra brukeren. Det har derfor blitt satt fokus på at applikasjonene skal være lagt opp slik at de funksjonene som er mest i bruk vil være tilgjengelig med lite interaksjon, mens de andre funksjonene som ikke man trenger så ofte ligger i underkategorier som for eksempel min konto/innstillinger og tillates at det kreves mer interaksjon for å navigere til disse.

4.1.1 Inspirasjon til brukergrensesnitt

Lignende løsninger

For å få et godt innblikk i hva slags applikasjoner som skulle lages, foreslo oppdragsgiver tidlig i prosjektet flere lignende løsninger gruppen kunne lære av. De lignende løsningene som ble foreslått heter som følger: "Home control", 'Fibaro', 'Wifi Smart Socket', 'Wemo' og 'Future Home'. Oppdragsgiver var behjelpelig med å sende ned maskinvare for alle disse løsningene, slik at man fikk erfart hva de hadde til felles og hvordan brukergrensesnittene for deres applikasjoner så ut og fungerte i samsvar med maskinvaren. På denne måten fikk man i samarbeid med oppdragsgiver diskutert hva som ble ansett som gode og dårlige aspekter ved de forskjellige løsningene. Naturlig nok ble de gode aspektene inspirasjon til planleggingen av funksjonalitet og design, men enda viktigere var det nok å erfare de dårlige aspektene, slik at man kunne finne bedre løsninger på disse problemområdene, eller unngå de helt og holdent.

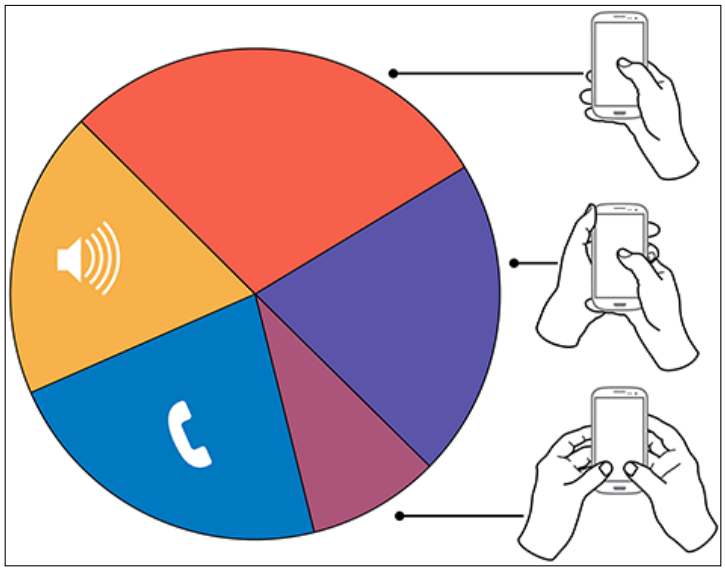
Etter et par runder med diskusjon internt i gruppen og sammen med arbeidsgiver, begynte arbeidet med å finne ut hvordan det grafiske brukergrensesnittet for de forskjellige applikasjonene skulle se ut. Et par av løsningene som ble gjennomgått hadde et lite brukervennlig design. De fremsto som rotete, inneholdt kontraster som ikke opprettholdt WCAG standarder [20], tok i bruk dårlig symbolikk, hadde forvirrende/manglende tilbakemeldinger, samt manglende internasjonalisering og i et tilfelle blanding mellom flere språk samtidig. Takket være disse erfaringene har gruppen vært fokusert på at informasjon må organiseres på en god måte. Det har også blitt tatt i bruk kontrastsjekking på de fargene som endte opp i prototypene. Internasjonalisering ble ikke med i prototypene, men ble satt opp tidlig for begge applikasjonene i utviklingsperioden.

Populære applikasjoner

Det ble deretter bestemt at sosiale medier og populære nettsteder skulle undersøkes som inspirasjon for å lage et brukergrensesnitt som brukerne kunne gjenkjenne. Det første som måtte planlegges var hvordan brukerne skulle navigere rundt i applikasjonen på en god måte. Ettersom det var et ønske at applikasjonene skulle være intuitive, ble det undersøkt hvordan sosiale mediene håndterte navigering i sine løsninger, både for mobil og web. Dette kunne dermed tas med videre i utviklingen av de forskjellige applikasjonene til EcoHeat.

I likhet med de lignende styringsløsningene som ble gjennomgått brukte også mange av applikasjonene til sosiale medier en tabbar for hovednavigasjonen. Hvor mange tabs denne tabbaren hadde varierte i de forskjellige løsningene. Formålet med denne formen for navigasjon er at alle de viktigste navigasjonsmulighetene nesten alltid vil være tilgjengelig, samme hvor man befinner seg i applikasjonen. På denne måten får man også gruppert lignende funksjonalitet i de forskjellige tabbene. Det er også et poeng at man ikke skal ha for mange tabs i en slik tilnærming til navigasjon fordi enhver tab skal representere noe av hovedfunksjonaliteten til applikasjonen. Ender man opp med mange tabs virker det rotete og man burde nok ta en revurdering på hva kjernen i applikasjonen består av, ettersom det ikke er all funksjonalitet som behøver å befinne seg i en egen tab. Spesielt for mobil blir det fort trangt om plassen.

Basert på dette ble det opprettet fire forskjellige tabs i applikasjonene for EcoHeat: Hjemme, Mine enheter, Kalender og Innstillinger. Det ble også diskutert litt ut i prosjektet om tabbaren skulle være øverst eller nederst på mobilapplikasjonen, ettersom plasseringen av denne varierer mellom Android og iOS, men det endelige resultatet ble at den skulle være plassert nederst på skjermen. Dette fordi hovednavigasjonen da er nærmere fingrene til brukeren, som også bygges opp av en artikkel på uxmatters.com [21].



Figur 7: Slik holder folk mobiltelefonene sin når de bruker den. Kilde: uxmatters [22]

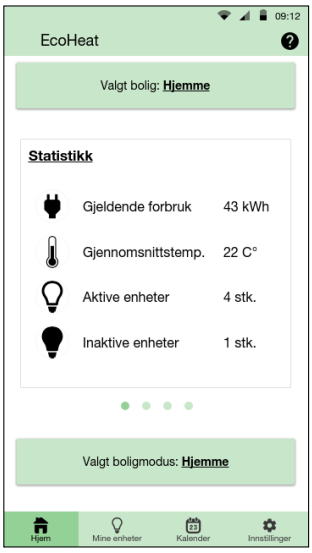
Her skriver Steven Hoober at undersøkelsene hans viste at 49 % bruker mobiltelefonen sin med en hånd, 36 % holder telefonen i en hånd og bruker den med den andre og 15 % bruker begge hendene aktivt. Dette er også illustrert med hans diagram, se figur 7, der man i tillegg ser at en stor del av brukermassen i denne undersøkelsen var i en telefonsamtale eller brukte telefonen passivt (hører på musikk, ser på video, osv). Basert på dette ser man at flertallet bare bruker tomlene sine for å samhandle med skjermen og dermed er det lurt å plassere hovednavigasjonen så nærme tomlene som mulig. Dette gjelder mobilapplikasjonen i større grad enn webapplikasjonen. Siden webapplikasjonen har en større skjerm og brukeren vil enten bruke en musepeker eller ha hendene fritt tilgjengelige ved bruk av en tablet må man ikke ta disse hensynene for navigasjon.

4.1.2 Plassering av funksjonalitet

For å lage brukergrensesnittene måtte det først bestemmes hvordan funksjonaliteten skulle grupperes. Det ble tatt en vurdering av hva som ble sett på som kjernefunksjonaliteten for EcoHeat. Det mest sentrale som ble bragt opp var boligvalg, modusvalg, enhetslisten og kalenderen for applikasjonene. I tillegg forslo gruppen at det man måtte ha en eller annen side for administrasjon. Denne siden ville gi brukeren tilgang til å redigere informasjon om sin egen konto og sine hushold (administrasjon). Det ble også hentet inspirasjon fra de populære applikasjonene til å lage seksjoner for både generelle innstillinger, varsler, tilbakemeldinger og selvhjelp.

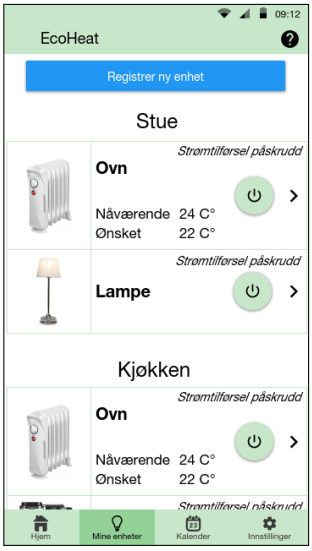
Mye av kjernefunksjonaliteten var enkel å dele opp, ettersom det var tydelige logiske sammenhenger. For eksempel enhetsliste og muligheten for å registrere nye enheter, eller bolig og modusvalg på samme side. Derfor kunne det lages tabs for disse forskjellige inndelingene. Under er en gjennomgang av hvor forskjellig funksjonalitet ble plassert i de forskjellige brukergrensesnittene og figurer av hvordan dette ble seendes ut i prototypene.

4.1.3 Mobil



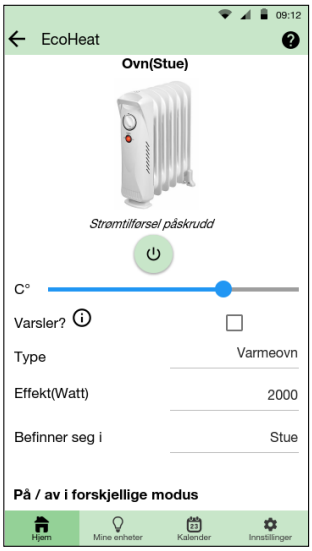
Figur 8: Forside (Hjem) - Mobil

På forsiden ble det bestemt at man skulle vise informasjon som kunne være interessant for brukeren å følge med på, for eksempel antall aktive enheter og gjennomsnittstemperatur samt forbruket i boligen. Dette er implementert ved bruk av en karusell, hvor den første siden gir all informasjonen oppsummert, mens de andre sidene går mer i detalj om hver enkelt del av informasjon. I tillegg til å vise denne informasjonen gir forsiden muligheten til å endre bolig samt modus for dette huset. Ved å trykke på de forskjellige knappene får brukeren opp en liste med radioknapper for alle de forskjellige boligene/modusene. Naturlig nok vil statistikken reflektere den boligen som er valgt for øyeblikket.



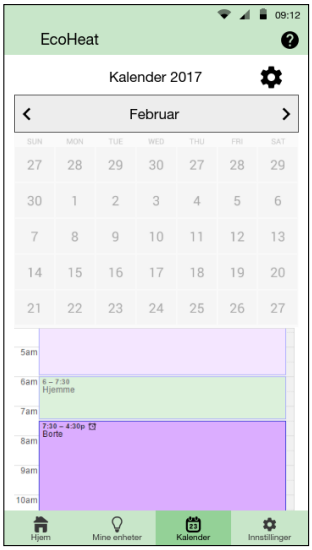
Figur 9: Mine enheter - Mobil

Skjermen for 'Mine enheter' gir brukeren en liste over alle enhetene som er registret i det valgte husholdet, i tillegg til muligheten for å registrere flere enheter via 'Registrer ny enhet' knappen. Enhetene i listen er gruppert på hvilket rom brukeren registrerte den på. Alle enhetene i denne listen har en knapp med mulighet for å kunne manuelt skru enheten av eller på, uten å måtte navigere videre til den mer detaljerte siden for en gitt enhet. Til og begynne med var denne knappen grønn når enheten var skrudd på og rød når den var skrudd av, men dette ble endret til grønn og hvit slik at det ikke skulle være forvirrende for fargeblinde. I tillegg ble det lagt til en informativ tekst over knappen, slik at brukeren får tilbakemelding om at noe faktisk skjedde.



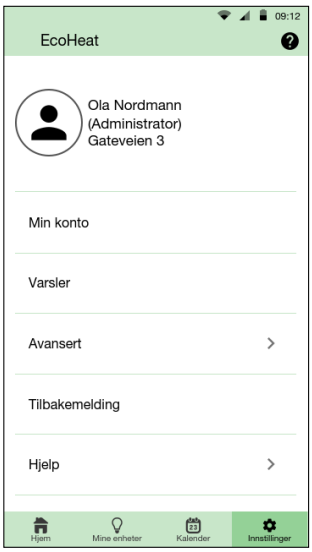
Figur 10: Detaljert enhetsside - Mobil

Om man trykker på en av enhetene fra Mine enheter, blir man tatt videre til en skjerm som viser mer detaljert informasjon om den valgte enheten. I likhet med enhetslisten får man også muligheten til å skru av eller på enheten manuelt, men i tillegg for man også manuelt sette en ønsket temperatur om enheten har mulighet for å regulere temperatur. Dette gir brukeren muligheten til å kunne regulere temperaturen der og da frem til neste automatiske modusendring, uten å redigere moduskonfigurasjonene for enheten. I tillegg skal man kunne redigere all informasjonen om enheten og bla ned for å konfigurere den (om de skal være av eller på og hvilken temperatur enheten skal prøve å oppnå i de forskjellige modusene, se figur 8 i vedlegg C).



Figur 11: Kalender - Mobil

Kalenderen skulle gi brukeren oversikt over hvilke dager og tidspunkter de forskjellige modusene er aktive og gi deg muligheten til å endre og/eller lage nye moduser. Tannhjulet vil ta brukeren videre til hvor han kan legge inn tidspunktene for nye moduser.



Figur 12: Innstillinger - Mobil

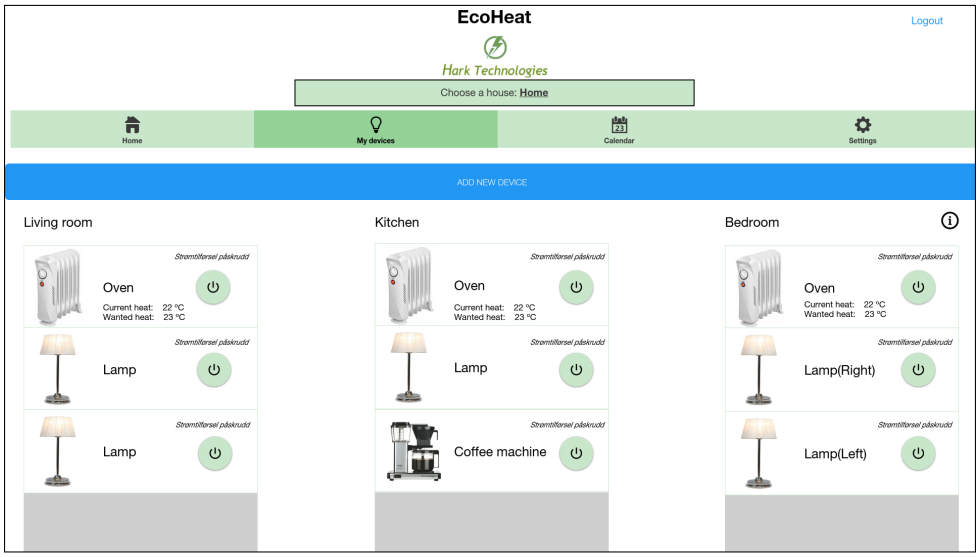
I Innstillinger får man muligheten til å redigere kontoinformasjon, endre hvilke varsler som applikasjonen skal sende ut, mer administrative innstillinger (Avansert), mulighet for å gi tilbakemelding og en selvhjelpsseksjon. Denne siden ble i stor grad brukt til å samle funksjonalitet som ikke ansees som kjernen i applikasjonen og vil dermed ikke benyttes like ofte. I prototypen ble denne tabben kalt 'Innstillinger', men under utviklingen ble det bestemt at 'Min konto' var et mer beskrivende navn, da denne siden inneholder mer enn bare innstillinger. Hvordan hver av de forskjellige underkategoriene så ut i prototypen kan du se i vedlegg C.

4.1.4 Web

Webapplikasjonen skulle egentlig inneholde funksjonalitet som ikke passet inn i en mobilløsning, men det var egentlig ingen funksjonalitet som ikke også kunne implementeres i mobilapplikasjonen. Derfor ble det heller bestemt at webapplikasjonen skulle vise mer informasjon på hver skjerm, da det naturligvis er generelt mer plass i en nettleser. Det var også et mål at web- og mobilapplikasjonene skulle ha noe grad av likhet, slik at brukerne kunne benytte seg av de forskjellige applikasjonene om hverandre.

Figur 13: Hjem - Web

Fremsiden på webapplikasjonen viser mye av den samme informasjonen som mobilapplikasjonen og mulighet til endring av bolig og modus. I stedet for en karusell var det heller mulig å vise frem all denne informasjonen direkte, i tillegg til en graf som gir innsikt i hvordan den valgte boligen opptrer i sanntid. I tillegg har grafen ulike filtre for hvilken informasjon man ønsker å vise frem, som man kan aksessere ved å trykke på knappen i toppen av venstre hjørnet av grafen.



Figur 14: Mine enheter - Web

Enhetssiden er så og si den samme som på mobilen, men for å benytte all plassen som er tilgjengelig ble det bestemt å ha så mange rom som mulig i bredden på skjermen og liste opp alle enhetene tilhørende rommene nedover. Når man klikker på en enhet vil også all informasjonen om denne enheten bli vist direkte på den samme siden, uten videre navigasjon.

Figur 15: Kalender - Web

Kalenderen for webprototypen er veldig naken da den ikke fikk gjennomgått flere utviklingsfaser. Tanken bak den var at brukeren skulle kunne se visuelt når de forskjellige modusene var aktive og i tillegg kunne endre dette direkte i kalendere, med mulighet for stor grad av automatisering slik at brukeren ikke behøver å sette opp hver eneste dag individuelt.

Figur 16: Innstillinger - Web

Når det kommer til innstillinger er det en del ting som ser annerledes ut sammenliknet med mobilapplikasjonen. Her prøves det å utnytte plassen slik at brukeren ikke blir navigert frem og tilbake i like stor grad. Til venstre finnes de samme menyvalgene som på mobilapplikasjonen, men all plass til høyre brukes til å vise den samme informasjonen som mobilapplikasjonen ville vist frem ved å navigere brukeren til en ny side.

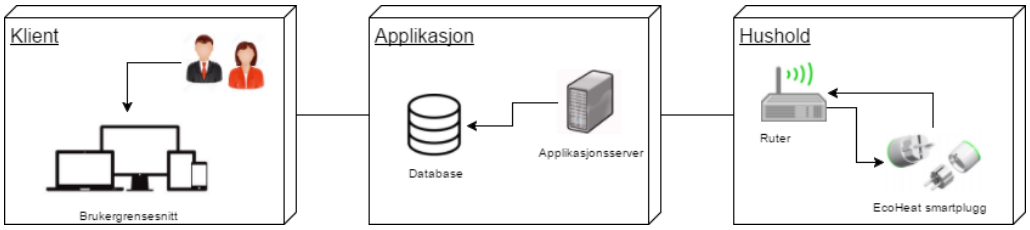
4.2 Arkitektur

Hele systemet baserer seg på klient/tjener-arkitektur, der artefakter fra '4+1 architectural view model' [23] har blitt brukt for å beskrive arkitekturen med hensyn til forskjellige parter (brukere, utviklere og prosjektstyrere).

4.2.1 Artefakter

Topologi av fysiske komponenter

Figur 17 illustrerer topologien over de fysiske komponentene ved systemet. Her er det tre hovedkategorier systemet består av: klient, applikasjon og hushold. Dette artefaktet er hentet fra 4+1 - Physical view.



Figur 17: Deployment diagram

Klient er brukerens måte å kommunisere med applikasjonen på via forskjellige brukergrensesnitt. Ved hjelp av applikasjonsserveren kan klienten autentiseres og innhente eller oppdatere informasjon som tilhører brukeren. Applikasjonsserveren sørger dermed for at

databasen og all informasjon om tilkoblede hushold holdes ved like. I tillegg er applikasjonsserveren ansvarlig for mye av automatikken innenfor systemet ut ifra brukernes konfigurasjoner, uten at det kreves interaksjon fra en klient. Hushold er plassert for seg selv og ikke sammen med klienten fordi en klient aldri vil manipulere enhetene i et hushold direkte, for dette vil alltid skje via applikasjonsserveren. Innad i et hushold sørger ruteren for at smartpluggene mottar de kommandoene som applikasjonsserveren sender ut og mottar målingsdata (om temperatur og lignende) fra smartpluggene som sendes tilbake applikasjonsserveren.

Applikasjonsflyt under bruk

Vedlegg F - Aktivitetsdiagram viser hvordan applikasjonen skal reagere når brukerne samhandler med de forskjellige brukergrensesnittene. Dette vedlegget forteller om hvilke muligheter for navigasjon som finnes samme hvor brukeren befinner seg i applikasjonen og hvilke tilbakemeldinger brukeren kan forventer når forskjellige operasjoner utføres. Diagrammet bryr seg ikke med hvilke teknologier og fysiske komponenter som gjennomfører de forskjellige operasjonene, men heller hva brukeren ser til enhver tid.

Database

Systemet forventes å behandle to kategorier med data: hushold og brukere. Et hushold trenger å holde styr på en hel del forskjellig informasjon, både til fremvisning av informasjon til brukerne og for å ha logiske underkategorier slik at innhenting og oppdatering av data skal være så smertefritt som overhodet mulig. Til å begynne med vil et hushold ha behov for å vite hvilke medlemmer som hører til seg selv. Dette er fordi andre brukere også skal kunne se hvilke medlemmer som er en del av det samme husholdet og administratoren for et hushold må ha mulighet for å legge til eller fjerne medlemmer fra sitt eget hushold. I tillegg skal administratoren kunne gi andre medlemmer flere tillatelser, slik at andre brukere kan få tilgang til operasjoner som ansees å være av en mer alvorlig grad. Eksempler på slike operasjoner kan være å legge til nye medlemmer i husholdet, nye enheter eller sletting av forskjellig data lagret angående husholdet.

Et hushold vil også ha kunnskap om hvilke rom husholdet består av, som er relevant i forhold til systemet. Det vil si at dersom et rom har en tilkoblet enhet i seg, vil alle enhetene innenfor samme rommet kunne grupperes sammen. Slik kan man enkelt kategorisere enhetene i forskjellige rom. Dermed kan det forskjellige brukergrensesnittene enkelt innhente data om alle enheter i enkelte rom eller få en helhetlig oversikt over alle de relevante rommene innenfor husholdet.

Det må også lagres informasjon om enhetene, slik at det kan vises frem i de forskjellige brukergrensesnittene. I tillegg vil hver enhet også ha konfigurasjon tilknyttet forskjellige modus lagret, om hvilken status og eventuell ønsket temperatur enheten skal prøve å oppnå når et gitt modus er aktivt. Slik kan applikasjonsserveren, som nevnt ovenfor, enkelt få iterert over alle enhetene i husholdet ved en modusendring, og endre statusen til hver enhet slik at den samsvarer med moduset husholdet er satt til å være i.

På den andre siden vil systemet ha informasjon om brukerne. For å personliggjøre applikasjonen lagres det unna litt basisinformasjon om brukeren og i tillegg hvilke hushold

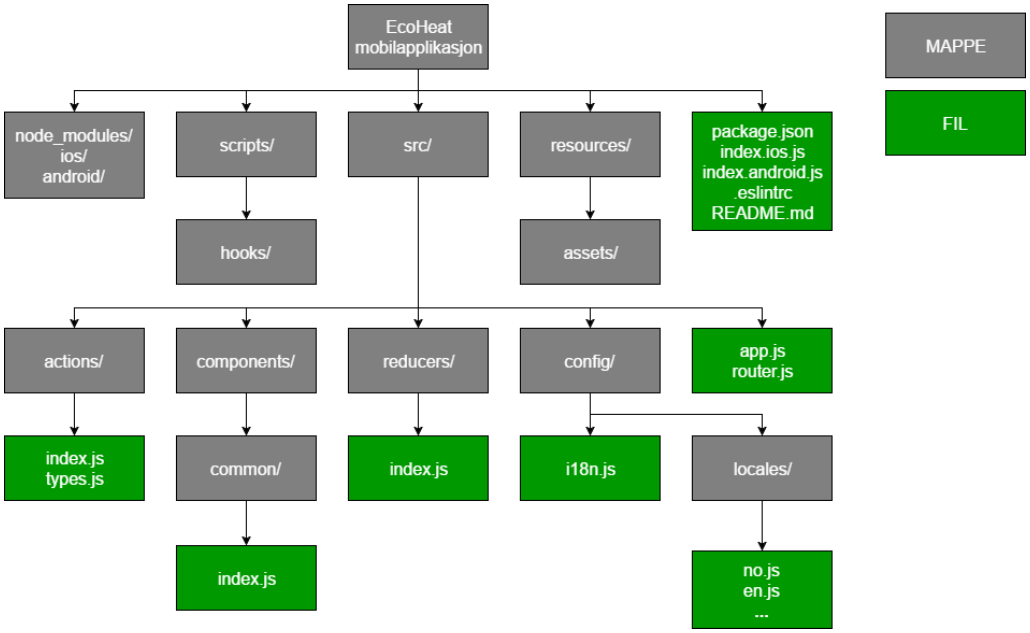
brukeren er en del av. Slik blir det enkelt å innhente all informasjon om hvert av husholdene og fremvise dette i brukergrensesnittene for applikasjonene.

4.2.2 Filstruktur

Filstrukturerene for de forskjellige applikasjonene har mye til felles, men det er også noen forskjeller der blant annet mobilapplikasjonen tar for seg to forskjellige plattformer. I figur 18 og 19 er den overordnede filstrukturen for applikasjonene illustrert. Begge applikasjonene inneholder mappene *node_modules*, *scripts* og *src* i roten av prosjektene, i tillegg er også filene *package.json*, *.eslintrc* og *README.md* felles. Mappen *node_modules* innerholder alt av NPM moduler som applikasjonene avhenger av. Dette kan være alt fra moduler for en ferdiglaget dialogboks til verktøy som gjør det lettere å iterere over data. Dette er en mappe man typisk ser i JavaScript applikasjoner av denne grunn.

I mappen *scripts* lagres alle scripts som ikke har noe direkte med kildekoden å gjøre. For eksempel git hooks som brukes for å automatisere deler av versjonskontrollen.

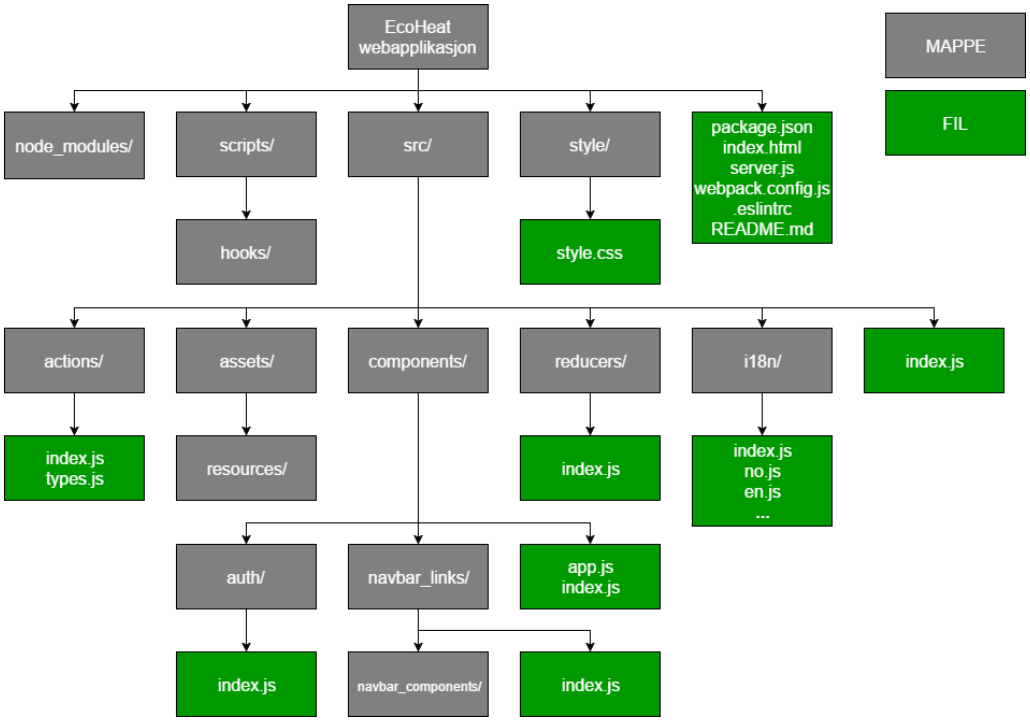
Til slutt finnes mappen *src* som inneholder alt av kildekode innenfor hver av prosjektene. For begge applikasjonene har det blitt lagt opp slik at *src* inneholder undermappene *actions*, *components* og *reducers* som deler opp kildekoden slik at logikk og brukergrensesnitt blir i stor grad liggende hver for seg. I disse prosjektene tar *actions* og *reducers* seg for applikasjonslogikken og *components* de forskjellige komponentene brukergrensesnittene er bygget opp av.



Figur 18: Overordnet filstruktur for mobil

Figur 18 viser mer spesifikt strukturen for filene under mobilapplikasjonen. Det som skiller mobil fra web are blant annet mappene *android* og *ios* som inneholder prosjektfiler og native kildekode for de forskjellige plattformene og blir automatisk generert ved starten av et nytt React Native prosjekt. I tillegg får man filene *index.ios.js* og *index.android.js*

som representerer inngangen til de forskjellige versjonene av applikasjonen. Slik kan man differensiere mellom Android og iOS applikasjonene helt fullstendig, men i dette prosjektet peker disse filene til *src/app.js* ettersom applikasjonen skal i teorien være lik hos begge plattformene. Den siste filen i rotkatalogen som er forskjellig fra web, *router.js*, definerer alle de forskjellige navigasjonsmulighetene i applikasjonen.



Figur 19: Overordnet filstruktur for web

Figur 19 viser derimot filstrukturen for webapplikasjonen. Her har man mappen style som inneholder CSS for applikasjonen, i18n som inneholder språkfilene for internasjonalisering og de underliggende mappene i components. De underliggende mappene var et resultat av omstrukturering gjennom utviklingsprosessen for å gjøre filstrukturen mer oversiktlig.

5 Realisering og implementasjon

For å realisere applikasjonene kreves det en del verktøy, teknologier og innsikt i de forskjellige rammeverkene. I dette kapitlet vil det forklares hvordan forskjellige moduler ble implementert i applikasjonene, både de importerte og selvskrevne, og hvilke verktøy som ble benyttet under utviklingen.

5.1 Utviklingsmiljø

Ettersom Javascript ble brukt som utviklingsspråk, i form av React og React Native, var det en hel del forskjellige tekstbehandlere å velge mellom. Valget endte opp på Atom [24] da denne tekstbehandleren var anbefalt av kursene gruppen hadde gjennomført og det var innebygd funksjonalitet for bruk av Linter. I tillegg hadde flere av gruppemedlemmene erfaring med Atom fra før og og var positive til å ta i bruk denne tekstbehandleren.

For å kjøre mobilapplikasjonen under utviklingen ble Android Studios virtuelle enheter benyttet. Her ble både Nexus 5X og Nexus 6 med API 23 benyttet. I tillegg ble applikasjonen også kjørt på en Honor 7. Mobilapplikasjonen kunne ha blitt testet på enda flere forskjellige modeller, men dette antallet ble sett på som akseptabelt.

For webapplikasjonen ble webpack development server [25] benyttet. Dette er en Node.js server som lager en bundle av Javascript koden og kjører den lokalt på maskinen. Konfigurasjonen for webpack som ble brukt i dette prosjektet ligger i vedlegg H. Deretter er det bare å aksessere applikasjonen med en nettleser, for dette prosjektet ble Google Chrome og Microsoft Edge benyttet.

5.2 Versjonskontroll

For versjonkontroll ble Git og Bitbucket brukt fordi dette er en av de mest populære versjonkontrollsystemene idag og alle gruppemedlemmene hadde erfaring med disse verktøyene gjennom andre prosjekter ved skolen. Atom hadde i tillegg innebygd funksjonalitet for å samarbeide med git, som gjorde det enklere å vite hvilke branches man arbeidet med for til en hver tid.

5.3 React

React er et rammeverk for Javascript utviklet av Facebook. Det er laget for å utvikle brukergrensesnitt og gjør det med sin komponentstruktur enkelt å utvikle en modulbasert applikasjon. React er deklarativt, noe som gjør det enkelt å se logikken i koden og bedrive feilsøking. Det betyr også at du som utvikler bare må bry deg om hva som skal skje og ikke hvordan. [26] React støtter ES6-syntax som gjør Javascript mer intuitivt og forståelig. Man får blant annet scoping for variable og funksjoner, klasser og bedre måter å importere/eksportere verdier. [27] React benytter seg også av npm, pakkesystemet til Node.js. [28] Her har du tilgang til funksjoner andre har utviklet og gjort tilgjengelig.

Disse pakkene er langs hele spekteret når det kommer til kompleksitet, fra en pakke for å vise frem grafer til hele systemet for Redux.

Komponentene er det som står sentralt i React og du har to forskjellige variasjoner, klassebaserte og funksjonelle komponenter. Funksjonelle komponenter er vanlige funksjoner og brukes normalt når du bare skal vise frem informasjon og ikke gjøre noe logikk i komponenten. Klassebaserte komponenter er mer komplette og har tilgang til livssyklusfunksjoner og render, og render står for noe av magien i React. Det er en funksjon som renderer HTML inn i React sin DOM og blir kjørt når en komponent er lastet inn i DOMen. Deretter vil komponenten kunne re-rendre seg hver gang tilstanden dens blir endret og dermed vil applikasjonen holde seg oppdatert når data endres.

Når man først har laget en komponent kan den benyttes hvor som helst i applikasjonen, noe som gjør at du som utvikler er oppfordret til å lage gjenbrukbare komponenter og sende med data og funksjoner som props. Props er parametre man kan gi en komponent og kan være enkeltverdier, objekter, arrayer eller funksjoner som vist i eksempelet under. I komponenten vil de så ligge under 'this.props' (klassebaserte komponenter) eller 'props' (funksjonelle komponenter) og kan aksesseres derifra.

```
import React, { Component } from 'react';

class ClassBasedComponent extends React {
    render() {
        return (
            <div>
                /**
                 * Here we declare two instances of a
                 * component and send with a value as a prop.
                 */
                <FunctionalComponent specifiedValue={5} />
                <FunctionalComponent specifiedValue={'Fish'} />
            </div>
        )
    }
}

const FunctionalComponent = (props) => {
    return (
        <div>
            You sent {props.specifiedValue} as a prop.
        </div>
    );
}
```

Kodeeksempel 1: Eksempel på bruk av React med både klassebaserte og funksjonelle komponenter og props.

5.4 Redux

Redux er en container for å gi Javascript-applikasjoner en tilstand. Denne tilstanden gjør at man enklere kan lage en applikasjon som oppfører seg konsistent. [29] Tilstanden inneholder applikasjonsrelevant informasjon representert som objekter og arrayer som kan aksesseres når som helst. Tilstanden ligger i noe som kalles en 'Store' og det er denne man må kommunisere med for å endre tilstanden. Den blir initiert i roten av applikasjonen og er da en del av applikasjonen.

```
1       import React from 'react';
2       import ReactDOM from 'react-dom';
3       import { Provider } from 'react-redux';
4       import { createStore, applyMiddleware } from 'redux';
5       import reduxThunk from 'redux-thunk';
6
7       import reducers from './reducers';
8       import App from './components/app';
9
10      /**
11       * Creating the store.
12       */
13      const createStoreWithMiddleware = applyMiddleware(reduxThunk)(createStore);
14      const store = createStoreWithMiddleware(reducers);
15
16      ReactDOM.render(
17        <Provider store={store}>
18          <App />
19        </Provider>
20        , document.querySelector('.container'));
```

Kodeeksempel 2: Eksempel på initiering av Redux i en React-applikasjon.

For å benytte Redux i React trengs en pakke som kalles React Redux. [30] Den binder Redux opp til React gjennom komponenten Provider og gjør at Store blir tilgjengelig for alle komponenter. For å få tilgang til den har React Redux funksjonen *connect()*. Den tar komponenten som skal tilkobles applikasjonens Store og returnerer en kopi av komponenten med tilkobling til den. Når man bruker *connect()* må man også spesifisere hvilke verdier man ønsker å hente fra Redux Store. Dette gjør man gjennom *mapStateToProps* (verdier) og *mapDispatchToProps* (action creators). Over er et eksempel på hvordan man inkluderer Redux i React. Her defineres en Store med middleware og applikasjonen blir plassert inne i en Provider slik at den får tilgang til vår Store (sendt med som en prop til Provider).

For å manipulere tilstanden er to ting nødvendig, action creators og reducere. Action creators er funksjoner koblet opp mot Redux og har som oppgave å returnerer objekter kalt actions. Hvordan actions blir definert er opp til brukeren, men en standard er å kalle identifikatornøkkelen state og datanøkkelen payload. Disse actionene blir sendt inn i Redux Store med en metode kalt dispatch, derav *mapDispatchToProps* i *connect*. Du gir creators tilgang til dispatch slik at de kan dispatche actions til Redux Store.

```
import { UPDATE_NUMBER } from '../actions/types';

 export function updateNumber(number) {
     return {
         type: UPDATE_NUMBER,
         payload: number,
     }
 }
```

Kodeeksempel 3: Eksempel på en action creator.

Reducere plukker kontinuerlig opp actions som blir sendt ut i Redux Store og sorterer på type, da gjerne i en switch. Hvis reduceren får en match vil den ta tilstanden og returnere et nytt objekt med den gamle tilstanden og ny/oppdatert data fra actionen. Hvis den ikke får en match returneres bare den gamle tilstanden.

```
import { UPDATE_NUMBER } from '../actions/types';

export default function (state = { number: 0 }, action) {
  switch (action.type) {
    case UPDATE_NUMBER:
      return { ...state, number: action.payload };
    default:
      return state;
  }
}
```

Kodeeksempel 4: Eksempel på en reducer.

Under er et eksempel på hele omløpet. Her kobles en komponent opp mot Redux Store, henter en verdi fra applikasjonstilstanden, importerer en action creator, oppdaterer verdien i tilstanden med creatoren når komponenten lastes inn i DOMen og returnerer en div med verdien fra applikasjonstilstanden.

```
import React, { Component } from 'react';
import { connect } from 'react-redux';
import * as actions from '../actions/number_actions';

class Foo extends Component {
     componentWillMount() {
         this.props.updateNumber(5)
     }

     render() {
         return (
               <div>{this.props.number}</div
         )
     }
}

function mapStateToProps(state) {
     return {
         number: state.data.number,
     }
 }

export default connect(mapStateToProps, actions)(Foo);
```

Kodeeksempel 5: Eksempel på bruk av connect() og Redux store.

5.5 Redux Thunk

Det ble bestemt at man skulle benytte Redux Thunk som middleware for dette prosjektet. [31] En Thunk [32] er en funksjon som omslutter et utrykk for å utsette utførelsen og det er nettopp dette Redux Thunk lar deg gjøre. Med Redux Thunk kan du returnere funksjoner fra action creators i stedet for å returnere actions direkte. I dette prosjektet har disse funksjonene blitt brukt til å utføre asynkrone hendelser som innlogging og innhenting av data. Her vil man ikke få umiddelbar respons og derfor er det heller ikke ønskelig å direkte oppdatere tilstanden og ende opp med utdatert informasjon. Da er det en langt bedre løsning å benytte en thunk for å utsette actions.

Kodeeksempel 7 er et eksempel på bruk av Redux Thunk.

5.6 Firebase

Som nevnt i 3.1.3 Datalagring kom man frem til at å bruke Firebase var den beste løsningen da man kunne gjøre både datalagring og brukerhåndtering der.

For å benytte Firebase i applikasjonene må man gjennomgå et par steg. Først må man laste ned Firebase-modulen fra Node.js slik at man kan aksessere pakken i applikasjonen. Deretter må Firebase initieres i applikasjonen, som vist i kodeeksempel 6.

```
import firebase from 'firebase';

/**
 * Initializing firebase for the application.
 */
const config = {
  apiKey: 'XXXX',
  authDomain: 'name-of-server.firebaseapp.com',
  databaseURL: 'https://name-of-server.firebaseio.com',
  projectId: 'name-of-server',
  storageBucket: 'name-of-server.appspot.com',
  messagingSenderId: '123456789',
};

firebase.initializeApp(config);
```

Kodeeksempel 6: Eksempel på initiering av Firebase.

Denne initieringen skal skje én gang og hvor dette skjer er opp til utvikleren. Normen for dette er å gjøre det i roten av applikasjonen, noe som betydde index.js for webapplikasjonen og App.js for mobilapplikasjonen.

Etter å ha gjennomført initieringen kan man begynne å jobbe opp imot APIet til Firebase. Da kan man autentisere brukeren, registrere en ny bruker, hente data fra databasen og annet Da applikasjonene benytter Redux valgte man å gjøre all interaksjon opp mot Firebase gjennom actions. Dette blir så koblet opp til reducerene som deretter behandler staten til applikasjonen. Et eksempel på dette er det å la en bruker logge inn, se kodeeksempel 7.

```
export function signInUser({ email, password, rememberCheckBox }) {
  /**
   * Dispatching with redux thunk since a login action is asynchronous
   */
  return (dispatch) => {
    firebase.auth().signInWithEmailAndPassword(email, password)
      .then(() => {
        dispatch({ type: AUTH_USER });
        browserHistory.push('/home');

        getInitialLocations(dispatch);
        getInitialUnits(dispatch);
      })
      .catch(error => (
        dispatch(authError(error.message))
      ));
  };
}
```

Kodeeksempel 7: Eksempel på innlogging med Firebase.

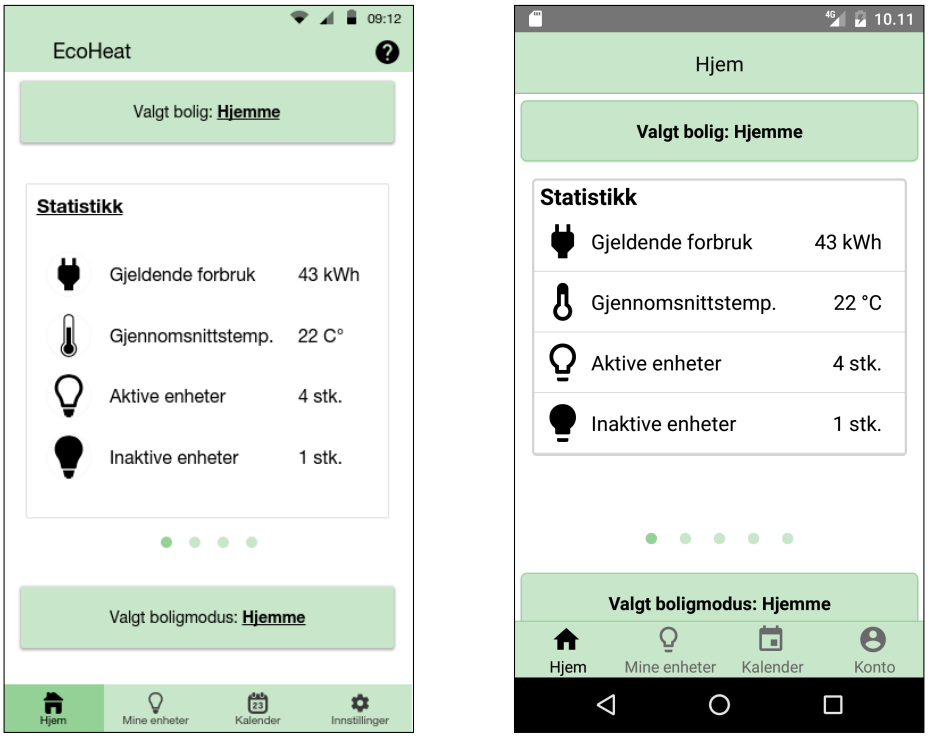
Når en bruker er autentisert kan man i applikasjonen hente data fra databasen. Dette gjøres ved å ta en referanse til databasen på Firebase-serveren og be om et snapshot av den. Etter å ha laget dette snapshotet vil du også lytte til endringer som skjer på path'en og få et nytt snapshot hver gang det skjer en endring. [33] Etter vurderinger innad i gruppen kom man frem til at dataene som ble hentet fra Firebase skulle ligge i applikasjonstilstanden, da det gjorde det enklest mulig å aksessere dataene der man trenger dem i applikasjonen. Dette gjorde også at de kontinuerlige oppdateringene endret tilstanden slik at den alltid inneholdt sanntidsinformasjon.

```
const getInitialLocations = (dispatch) => {
  firebase.database().ref('/locations')
    .on('value', (snapshot) => {
      dispatch({
        type: GET_LOCATIONS,
        payload: snapshot.val(),
      });
    });
};

const getInitialUnits = (dispatch) => {
  firebase.database().ref('/units')
    .on('value', (snapshot) => {
      dispatch({
        type: GET_UNITS,
        payload: snapshot.val(),
      });
    });
};
```

Kodeeksempel 8: Eksempel på å hente data fra Firebase.

5.7 Implementasjon av mobilapplikasjonen

Målet med implementasjonen var å få den så lik som prototypen som mulig. Dette viste seg å være mer tidskrevende enn først antatt og mesteparten av utviklingsperioden ble brukt til akkurat dette. I figur 20 er det lagt inn en sammenligning av prototypen vs. implementasjonen av hjem siden for mobilapplikasjonen, og som du sikkert ser er det ikke store forskjeller. Selv om det finnes moduler for å implementere mange av de elementene som eksisterer på de forskjellige skjermene, ble det fortsatt brukt mye tid på å få det til å se ut slik som prototypen gjorde.

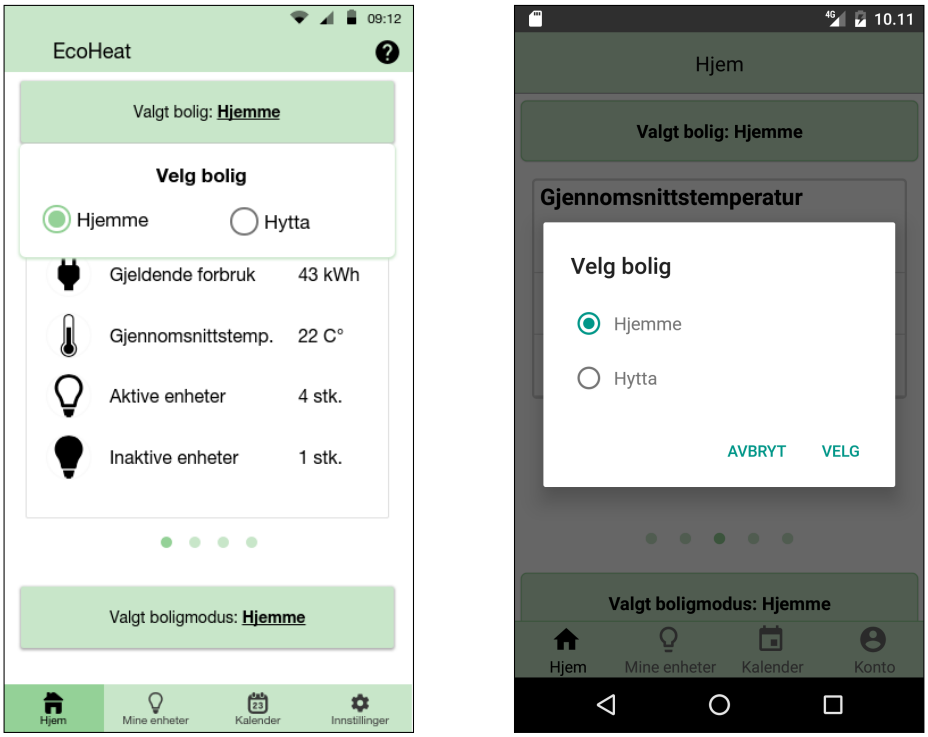


(a) Hjem - Prototype

(b) Hjem - Implementert

Figur 20: Sammenligning av prototype og implementasjonen av 'Hjem' på mobil

Som figuren viser er det ikke stor forskjell mellom prototype og implementasjon, men det også noen forskjeller her og der da det var deler av prototypen som ikke lot seg gjennomføre eller det ble oppdaget bedre løsninger under utvikling. Et eksempel på en bedre løsning som ble funnet i implementasjonen er figuren 21 der bolig og modusvalg„ heller bruker en dialogboks enn å ha en dropdown under knappen. Ikke bare er den tydeligere, men det krevdes veldig lite arbeid for å implementere den, da det allerede finnes React Native moduler for en slik dialog. Ettersom ingen av gruppemedlemmene er designere har det blitt prøvd å bli tatt i bruk eksisterende moduler, for fremvisning av informasjon og dialoger med brukeren i den grad det har latt seg gjøre, i stedet for å implementere dette selv. Det ble tatt en vurdering om utseendet til disse modulene var tilfredsstillende før implementasjon, men det har spart mye tid ettersom vi oppdaget akkurat hvor tidsskrevende brukergrensesnitt kan være å lage.



(a) Boligvalg - Prototype

(b) Boligvalg - Implementasjon

Figur 21: Forbedring av boligvalg i implementasjon.

5.7.1 Navigasjon

Til navigasjon har modulen react-native-router-flux blitt brukt. Denne modulen kommer med mye innebygd funksjonalitet rundt forskjellige navigasjonsmåter, i tillegg kan alt av utseende tilpasses applikasjonens behov. Fra denne modulen er det i hovedsak to av de inkluderte komponentene tatt i bruk, *Router* og *Scene*.

Router komponenten inneholder alle de forskjellige *Scene* komponentene. En *Scene* komponent beskriver hva som vises på skjermen til brukeren når det navigeres til en gitt scene. Dette bestemmes ved at man gir en scene en unik id, en komponent å vise frem, en tittel og eventuell styling man ønsker rundt komponenten som ble sendt med. I tillegg kan man bestemme om en scene skal vise frem en tabbar til navigasjon eller ikke.

I mobilapplikasjonen har det blitt implementert slik at det finnes to inndelinger for navigasjonflyt, autentisering og hovedflyt. Se kodeeksempel 9 for et grovt eksempel på dette oppsettet.

Autentiseringen inneholder naturlig nok alle scenes som har med autentisering å gjøre. Dette inkluderer for øyeblikket en scene for innlogging og registrering. Autentisering er også den første flyten brukeren vil møte på, som bestemmes ved å sende inn en prop som heter *initial* til scenen. Dersom *initial* ikke benyttes, vil modulen velge å vise frem den første scenen i routeren.

```
<Router>
    <Scene key="auth" inital>
        <Scene key="login"      ... />
        ...
    </Scene>

    <Scene key="main">
        <Scene key="home"       ... />
        <Scene key="myDevices"  ... />
        <Scene key="calendar"   ... />
        <Scene key="account"    ... />
    </Scene>

    <Scene key="editAccount"    ... />
    <Scene key="feedback"       ... />
    ...
</Router>
```

Kodeeksempel 9: Oppsett av router med forskjellige navigasjonsflyter

Hovedflyten, det vil si alle scenene som har direkte med applikasjonen å gjøre, inkluderer navigasjonen mellom de fire hovedinndelingene og enkelt scener, som det er mulighet for å navigere til flere steder i applikasjonen.

For å navigere mellom forskjellige scener kan man da benytte seg av *Actions* fra modulen. Actions håndterer navigasjonsstacken, slik at man programmatisk kan fortelle modulen hvor det skal navigeres til. For å navigere til kalender siden, vil man da kalle *Actions.calendar()*. Merk at her bruker man den unike nøkkelen for en scene.

5.7.2 Internasjonalisering

For mobil har internasjonalisering blitt implementert ved hjelp av modulen react-native-i18n. For alle referanser angående filstrukturen for kildekoden i mobilapplikasjonen, se figur 18.

Kodeeksempel 10 viser filen *i18n.js* som i applikasjonen er ansvarlig for konfigurasjonen av internasjonalisering i mobilapplikasjonen. Til og begynne med settes standardspråket for applikasjonen til engelsk (linje 3). Dette gjøres av den grunn at dersom et språk ikke er støttet i applikasjonen, er det større sannsynlighet for at brukeren kan engelsk, enn norsk. På neste linje leter modulen etter hvilket språk mobilen er satt til, ved at *I18N.currentLocale()* blir kalt. Dersom språket ikke støttes vil standardspråket bli tatt i bruk. Deretter blir 'fallbacks' som gjør at dersom ressursfilen for amerikansk englesk ikke eksisterer (en-US), vil heller filen for 'normal' engelsk (en) bli brukt istedet. Til slutt samles alle de forskjellige objektene fra språkfilene i translations, som er der modulen vil lete for å finne oversettelsene basert på nøkkelen. Om man skulle legge til støtte for flere språk, må det opprettes en språkfil som importeres i denne filen, og legge den importerte filen til *translations* objektet.

```
import I18N from 'react-native-i18n';
import en from './locales/en';
import no from './locales/no';

I18N.defaultLocale = 'en';
I18N.currentLocale();

/**
 * Fallbacks are enabled so that'en-US' would go to 'en' if no translation
 * existed in 'en-US' for a given property.
 */
I18N.fallbacks = true;

/**
 * Translations contains all the supported languages.
 */
I18N.translations = {
  en,     // English
  no,     // Norwegian
  ...     // More languages
};

export default I18N;
```

Kodeeksempel 10: Konfigurasjon av react-native-i18n modulen.

Alle de forskjellige språkfilene befinner seg i *src/config/locales/* i filstrukturen. Kodeeksempel 11 viser et utdrag fra disse ressursfilene, eksempelvis engelsk og norsk. Språkfilene består av et enkelt JavaScript objekt med alle nøklene for forskjellige tekststrenger. Det er også mulighet for å gruppere tekststrenger ved å plassere de innenfor enda ett objekt. Det er viktig at språkfilene er konsistente med hverandre med at de inneholder de samme nøkkelene, for det er denne som blir brukt for å finne oversettelsene.

```
const en = {
    loginForm: {
        welcome: 'Welcome',
        email: 'Email',
    },
};
```

```
const no = {
    loginForm: {
        welcome: 'Velkommen'
        email: 'Epost',
    },
};
```

Kodeeksempel 11: Oppsett av ressursfilene.

For å ta i bruk oversettelsene kalles metoden *I18N.t()* med nøkkelen for den tekststrengen man ønsker å få tak i som parameter. Kodeeksempel 12 ville eksempelvis skrevet ut 'Velkommen' og 'Epost' til konsollen, dersom det oppdages at mobilen til brukeren er satt til norsk.

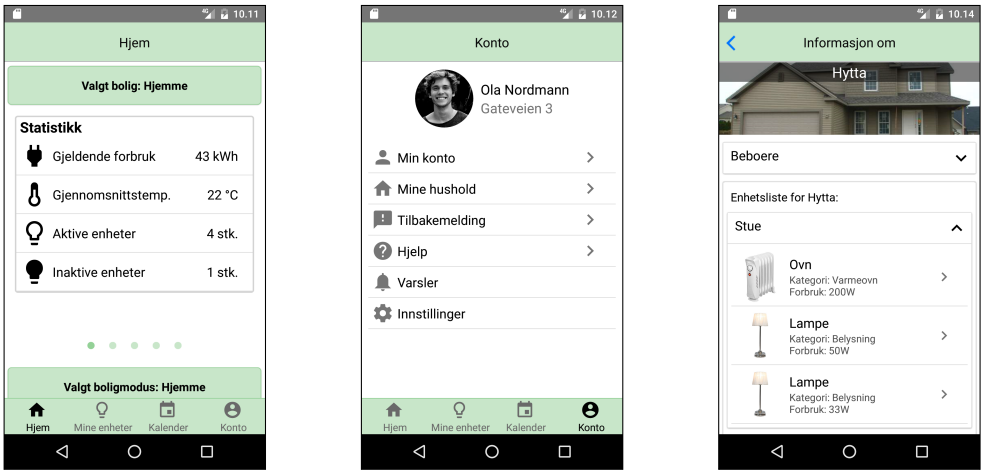
```
import I18N from './config/I18N';
...
console.log(I18N.t('loginForm.email'));
console.log(I18N.t('loginForm.emailPlaceholder'));
...
```

Kodeeksempel 12: Uthenting av verdier fra ressursfilene.

5.7.3 Gjenbruk av komponenter

React er som nevnt et rammeverk for å bygge grafiske brukergrensesnitt og da er det viktig at mange av komponentene man lager også er gjenbrukbare. Dermed må man lage komponenter slik at de ikke er avhengige av en hel del forskjellige ting, for at de skal fungere.

Den mest brukte selvlagde komponenten for mobilapplikasjonen er *ListItem.js* og brukes i samarbeid med et ListView (grafisk komponent for å vise frem en liste fra React Native). ListItem komponenten beskriver hvordan enhver rad i en liste skal se ut. For å unngå at hver eneste liste skal ha sitt eget ListItem, ble denne heller skrevet slik at den kunne takle fremvisning informasjon på forskjellige måter.



Figur 22: Gjenbruk av komponenter

Figur 22 viser flere steder i applikasjonen der ListItem brukes. På hjem skjermen (til venstre) viser den en liste over litt oppsummert statistikk, hvor det var ønskelig med et ikon, en tittel og selve verdien. På konto skjermen (i midten) ble den brukt i noe grad på samme måte, bare med mulighet for videre navigasjon i applikasjonen. Det siste bildet (til høyre) viser ListItem der det ønskes en liste over alle enheter med et bilde, en tittel, litt beskrivende informasjon og mulighet for videre navigasjon indikert med et ikon.

Ellers inneholder også *common/* mappen en god del av de gjenbrukbare komponentene, som stadig blir brukt overalt i applikasjonen. Knapper, tekstfelt og omrisser er blant disse komponentene.

5.8 Implementasjon av webapplikasjonen

Målet med implementasjonen av webapplikasjonen var at den skulle bli lik prototypen. Implementasjonen ble på noen steder forskjellig fra prototypen da aspekter ved denne var upraktisk å implementere, eller det ble funnet bedre løsninger underveis i utviklingen.

5.8.1 Internasjonalisering

For internasjonalisering ble pakken React Intl [34] benyttet. For å sette opp dette internasjonaliseringssystemet må man først identifisere hvilket språk som blir benyttet i nettleseren. Dette gjøres ved å undersøke språkkoden i navigator-objektet til nettleseren. Denne koden defineres forskjellig fra nettleser til nettleser, for eksempel kan norsk ha koden no-NO, no-NB, no-NN eller nb og dette må det tas høyde for når man skal velge riktig språk fra ressursfilen.

```
1  const en = {
2    'header.signin': 'Sign In',
3    'header.signout': 'Sign Out',
4  };
```

```
1  const no = {
2    'header.signin': 'Logg Inn',
3    'header.signout': 'Logg Ut',
4  };
```

Kodeeksempel 13: Ressursfiler for I18N - Web

```
1  import { IntlProvider } from 'react-intl';
2  import languagePackage from './i18n';
3
4  let language = (navigator.languages && navigator.languages[0]) ||
5                  navigator.language ||
6                  navigator.userLanguage;
7
8  let languageWithoutRegionCode = language.toLowerCase().split(/[_-]+/)[0];
9
10 /**
11  * Choosing the messages for the users language, fallback to 'en'.
12  */
13 const messages = languagePackage[languageWithoutRegionCode] ||
14                  languagePackage[language] ||
15                  languagePackage.en;
```

Kodeeksempel 14: Identifisering av brukerens språk

Etter å ha definert språket og valgt riktig språkpakke fra objektet 'languagePackage' kan man sende den inn i applikasjonen. Dette gjøres, som med Redux, ved hjelp av en provider kalt IntlProvider. Denne tar i mot språkpakken som et prop og gir deg mulighet til å aksessere strengene fra pakken i applikasjonen.

```
ReactDOM.render(
  <Provider store={store}>
    <IntlProvider locale={language} messages={messages}>
      ...
    </IntlProvider>
  </Provider>
  , document.querySelector('.container'));
```

Kodeeksempel 15: Oppsett av IntlProvider

Når IntlProvider er satt opp kan språkstrengene benyttes i applikasjonen. Dette kan gjøres på to måter, enten ved å bruke komponenten FormattedMessage eller hente ut strengene direkte fra context. Den førstnevnte metoden er den anbefalte da context er en eksperimentell API, men i enkelte tilfeller er det ikke mulig å representere en streng med en tag.

```
import React, { Component, PropTypes } from 'react';
import { FormattedMessage } from 'react-intl';

class Foo extends Component {
  static contextTypes = {
    intl: PropTypes.object.isRequired,
  }

  constructor(props, context) {
    super(props);

    /**
     * Extracting string with context.
     */
    this.namePlaceholder = context.intl.formatMessage(
      { id: 'form.placeholder.name' },
    );
  }

  render() {
    return (
      <div>
        <label>
          <RactIntl id="form.name" />
        </label>
        <input placeholder={this.namePlaceholder} />
      </div>
    );
  }
}
```

Kodeeksempel 16: Bruk av React Intl

5.8.2 Navigasjon

For å håndtere navigasjon ble React Router [35] benyttet. Dette er et system som lar deg deklarativt koble en URL til de enkelte komponentene i applikasjonen. Dette gir deg en single page application hvor du samtidig kan bokmerke de forskjellige sidene. I denne applikasjonen går man ikke lengre ned enn ett nivå, men det er ikke noe problem å plassere ruter inni hverandre slik at man får en URL med en nøstet path.

Figur 23: Eksempel på URL med React Router

I tillegg ble React Router Redux [36] benyttet for å synkronisere historikken med applikasjonens Store. Hittil i utviklingen har dette bare blitt brukt til å registrere URL-endringer for å tilbakestille endringer, men det er ikke noe i veien for å senere utvide bruken.

```
1  import { Router, Route, IndexRoute, browserHistory } from 'react-router';
2  import { syncHistoryWithStore } from 'react-router-redux';
3
4  /**
5   * Create an enhanced history that syncs navigation events with the store.
6   */
7  const history = syncHistoryWithStore(browserHistory, store);
8
9  <Provider store={store}>
10   <IntlProvider locale={language} messages={messages}>
11     <Router history={history}>
12       <Route path="/" component={App}>
13         <IndexRoute component={Welcome} />
14         <Route path="home" component={RequireAuth(Home, true)} />
15         <Route path="devices" component={RequireAuth(MyDevices, true)} />
16         <Route path="calendar" component={RequireAuth(Calendar, true)} />
17         <Route path="settings" component={RequireAuth(Settings, true)} />
18         <Route path="signup" component={RequireAuth(SignUp, false)} />
19         <Route path="signout" component={RequireAuth(SignOut, true)} />
20         <Route path="forgotpassword" component={RequireAuth(ForgotPassword, false)} />
21       </Route>
22     </Router>
23   </IntlProvider>
24 </Provider>
```

Kodeeksempel 17: Oppsett av React Router

React Router gir tilgang til flere navigasjonskomponenter, i dette prosjektet har Router, Routes og IndexRoute blitt benyttet. Router er roten i React Router og den komponenten som håndterer navigasjonen og hvordan React Router skal lytte til endringer i adresselinjen. History forteller React Router hvordan den skal lytte til disse endringene slik at systemet kan vise frem riktig navigeringskomponent. I dette prosjektet ble browserHistory benyttet, som bruker History APIet til nettlesere [37] for å lage en vanlig URL (se

figur 23).

Routes er navigasjonskomponentene i React Router og defineres med en path og en enkelt komponent knyttet til seg. Applikasjonen trenger en hovedkomponent, her kalt App, som kan brukes til å vise frem den aktive navigasjonskomponenten. Denne komponenten blir lagt under App og representeres som en underliggende komponent. For å kunne vise den frem vil App returnere 'this.props.children' til React DOM (se kodeeksempel 18). IndexRoute er en navigasjonskomponent som definerer en index for applikasjonen og vil tilsvare URLen *our-application.com*.

```
render() {
  return (
    <div>
      <Header />
      {/*
        * If the user is authenticated we can show him the
        * dropdown and navbar
        */}
      {this.renderAuthenticatedComponents()}
      {/*
        * Rendering the underlying components from react router
        */}
      {this.props.children}
    </div>
  );
}
```

Kodeeksempel 18: Rendermetoden til App

For å kunne navigere i applikasjonen brukes komponenten Link. Den tar i mot path'en du ønsker å navigere brukeren og representeres som en <a />. Her er det mulig å benytte Bootstrap slik at du kan endre utseendet til linkene, i dette prosjektet ble de stylet som knapper. Du kan også direkte be browserHistory om å navigere til en ny komponent, se kodeeksempel 7 for bruk av dette.

```
<Link
  to="/home"
  className="btn btn-custom-navbar"
>
  <FormattedMessage id="navbar.home" />
</Link>
```

Kodeeksempel 19: Bruk av Link for navigering i webapplikasjonen

5.8.3 Sikkerhet

Applikasjonen benytter funksjonen *RequireAuth()* (se vedlegg G) for å sikre navigasjonskomponentene fra brukere som ikke skal kunne aksessere de. Eksempelvis skal ikke en

allerede innlogget bruker få registrere en ny bruker og uvedkommende skal ikke kunne komme seg inn i applikasjonen. Hvordan funksjonen blir brukt vises i kodeeksempel 17.

5.8.4 Implementasjon av webdesign

App består av to hoveddeler, statiske elementer og navigasjonskomponenter. De statiske elementene er headeren, navigasjonsbaren og en nedfallsmeny for å la brukeren bytte mellom sine registrerte boliger.

Headeren inneholder to elementer, en logo og en link for innlogging og utlogging. Som med navigeringslinjen og nedfallsmenyen (se kodeeksempel 20) ble det her brukt betinget rendering for å endre funksjonen til linken. Innloggingsmenyen består av en Redux Form [38] og to linker, en til en side for å tilbakestille passordet og en for å registrere en ny bruker. Når brukeren prøver å logge seg inn kalles action creatoren signinUser (se kodeeksempel 7) med informasjonen fra formen og systemet prøver å få brukeren autentifisert av Firebase. Utloggingslinken kaller direkte på en annen action creator som fjerner autentifiseringen til brukeren og logger han ut.

(a) Innlogging (b) Utlogging

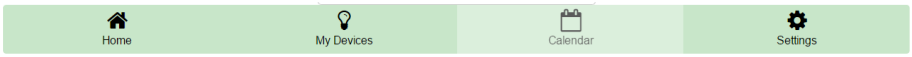
Figur 24: Link for innlogging og utlogging

Navigeringslinjen og nedfallsmenyen klassifiseres som autentiserte komponenter og vil bare bli vist til brukeren hvis han eller hun er logget inn (se kodeeksempel 18). Metoden for å gjennomføre dette er vist under.

```
/**
 * Method for doing conditional rendering. A user should only see the
 * dropdown menu and navigation bar if he's authenticated.
 */
renderAuthenticatedComponents() {
  if (this.props.authenticated === true) {
    return (
      <div>
        <div className="col-xs-4 col-xs-offset-4">
          <HouseDropdown />
        </div>
        <Navbar />
      </div>
    );
  }

  return null;
}
```

Kodeeksempel 20: Betinget rendering

Navigeringslinjen ble representert som en button group fra Bootstrap der hver knapp er en Link representert som en Bootstrap knapp. Ikonene ble hentet fra Font Awesome [39] og er de samme ikonene som ble brukt i mobilapplikasjonen.

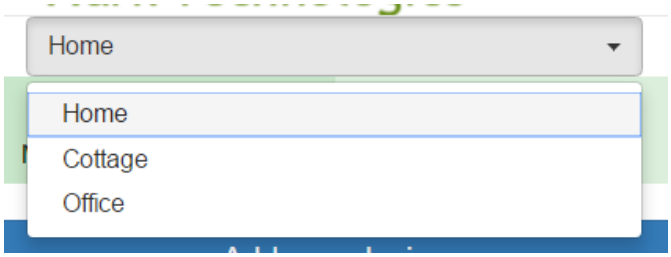


Figur 25: Navigeringslinjen for mobil

```
const Navbar = () => (
  <div className="btn-group btn-group-justified">
    <Link
      to="/home"
      className="btn btn-custom-navbar"
    >
      <FontAwesome name="home" size="2x" />
      <div><FormattedMessage id="navbar.home" /></div>
    </Link>

    ...
  </div>
);
```

Kodeeksempel 21: Oppsett av navigasjonslinjen

Nedfallsmenyene for webapplikasjonen er laget med pakken React-Select [40]



Figur 26: Nedfallsmeny for valg av hus

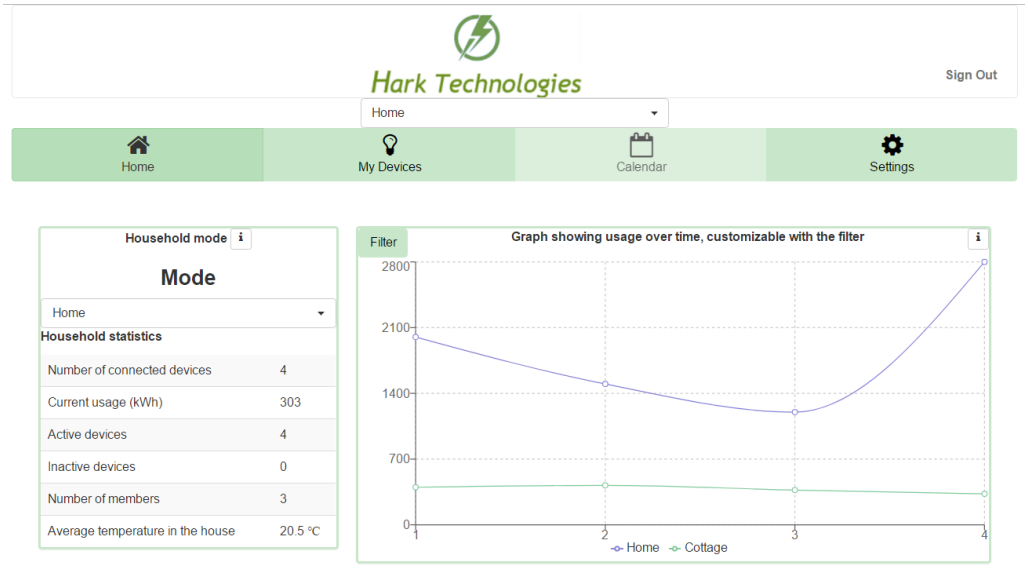
Da antall hus en bruker er knyttet til er forskjellig med forskjellige navn må menyalternativene settes opp på bakgrunn av dette. Her blir dette gjort ved å hente ut samlingen av hus i applikasjonstilstanden, gå gjennom samlingen og returnere et alternativ for hvert hus til SelectPicker. Hver gang brukeren endrer hus i menyen kaller SelectPicker på action creatoren changeHouse() som oppdaterer applikasjonstilstanden.

```
1  import SelectPicker from 'react-select-picker';
2
3  onHouseChange(houseName) {
4    this.props.changeHouse(houseName);
5  }
6
7  renderHouses() {
8    return this.props.houses.map(house => (
9      <option key={house.name}>
10       {house.name}
11     </option>
12   ));
13 }
14
15 <SelectPicker
16   className="form-control"
17   title={this.props.activeHouse}
18   defaultValue={this.props.activeHouse}
19   onChange={this.onHouseChange.bind(this)}
20 >
21   {this.renderHouses()}
22 </SelectPicker>
```

Kodeeksempel 22: Oppsett av nedfallsmenyen for hus

For dette prosjektet ble to av de planlagte hovedkomponentene utviklet, Hjem og Mine enheter.

Hjem består av fire deler: En tabell med statistikk for det aktive husholdet, en graf for å presentere datamålinger grafisk, et filter for å konfigurere grafen og en nedfallsmeny for å endre moduset til den aktive boligen. Tabellen er en 'striped table' fra Bootstrap og innholdet justerer seg etter hvilket hus og modus som er satt. Implementasjonen av nedfallsmenyen følger samme fremgangsmåte som i kodeeksempel 22.



Figur 27: Hjem - Webapplikasjonen

Grafen kommer fra Recharts [41], en pakke som lar deg sette opp en graf deklarativt hvor du som bruker selv kan designe den slik du ønsker. Den gir deg forskjellige komponenter du kan benytte for å oppnå et ønskelig resultat.

```
const data = [
  { name: 1, Home: 2000, Cottage: 400 },
  { name: 2, Home: 1500, Cottage: 420 },
  { name: 3, Home: 1200, Cottage: 370 },
  { name: 4, Home: 2800, Cottage: 330 },
];

<ResponsiveContainer height="80%" width="100%" aspect={2}>
  <LineChart
    width={700}
    height={300}
    data={data}
  >
    <Line type="monotone" dataKey={'Home'} stroke="#8884d8" />
    <Line type="monotone" dataKey={'Cottage'} stroke="#82ca9d" />
    <CartesianGrid stroke="#ccc" strokeDasharray="3 3" />
    <XAxis dataKey={'name'} />
    <YAxis />
    <Tooltip />
    <Legend />
  </LineChart>
</ResponsiveContainer>
```

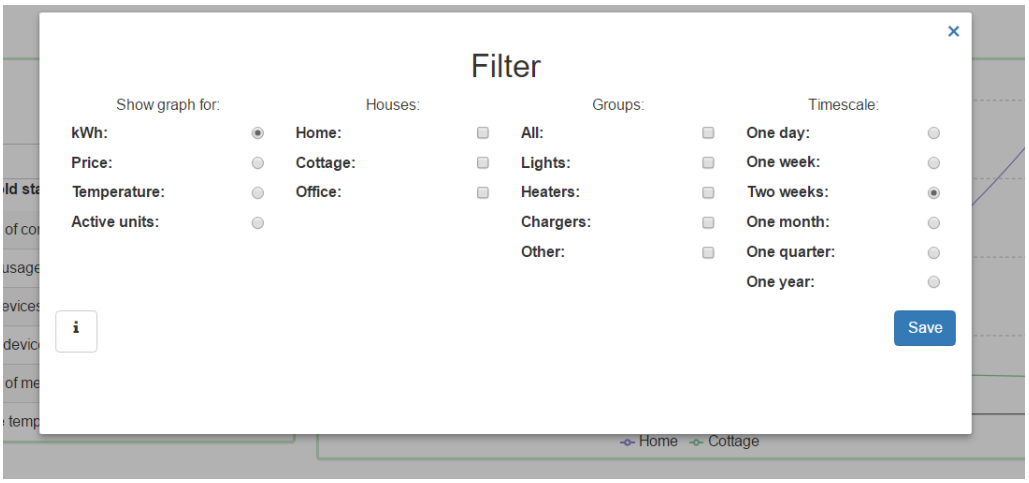
Kodeeksempel 23: Deklarativt oppsett av Recharts

Filteret er en modal hentet fra pakken React Skylight [42]. Inne i modalen ligger en

Redux Form med alternativer for fremvising av data og brukeren kan justere disse som han ønsker. Modalen blir tilegnet en referanse (ref) som gjør at den kan benyttes andre steder i komponenten og vises ved å kalle på 'this.referanse.show'.

```
<button
  className="btn btn-custom-navbar pull-left"
  onClick={() => this.filter.show()}
>
  <FormattedMessage id="home.filter.title" />
</button>
<SkyLight
  ref={ref => (this.filter = ref)}
>
  ...
</SkyLight>
```

Kodeeksempel 24: Oppsett av React Skylight

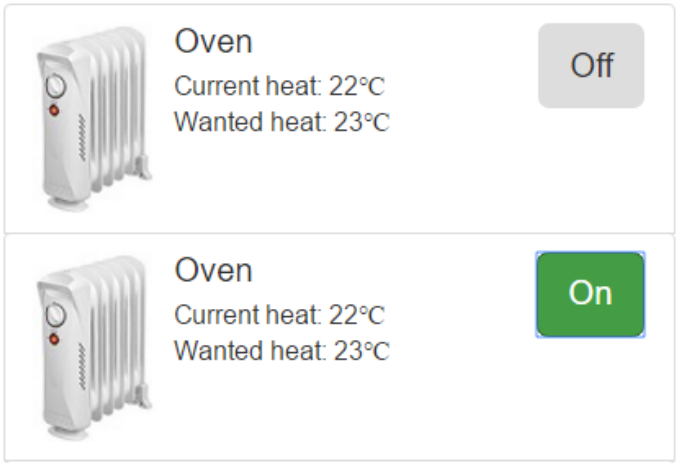


Figur 28: Filter for graf

I Mine enheter blir hver registrerte smartplugg i husholdet representert hver for seg. De blir listet opp for hvert rom og representert som list-group lister fra Bootstrap. Dette vil være gjenbrukbare komponenter som vil motta dataene til enheten den skal representere som props.

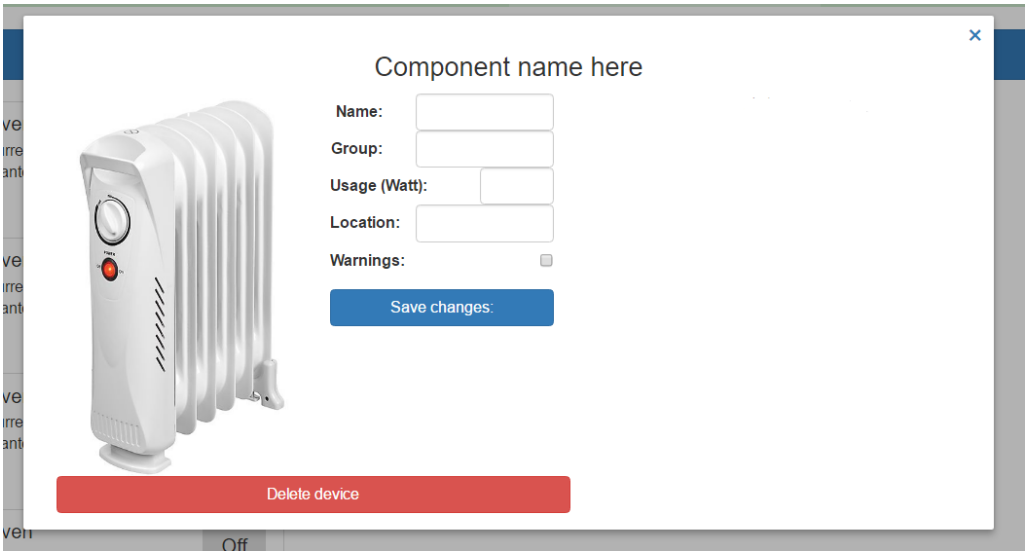
```
handleListItemClick(event) {
  if (event.target.name === 'button') {
    this.setState({ isActive: !this.state.isActive });
    {/* Call on action creator for changing state of component */}
  } else {
    this.componentEditor.show();
  }
}

<li
  onClick={this.handleListItemClick}
  className="list-group-item"
  id="listItem"
>
  <div className="media-left">
    <img
      className="media-object"
      src={props.img}
      alt=""
    />
  </div>

  <div className="media-body">
    <h4 className="media-heading">{props.componentName}</h4>
    <div>
      Current heat: {props.currentHeat}&#x2103;
    </div>
    <div>
      Wanted heat:  {props.wantedHeat}&#x2103;
    </div>
  </div>

  <div className="media-right">
    {this.renderToggleButton()}
  </div>
</li>
```

Kodeeksempel 25: Representasjon av smartplugger



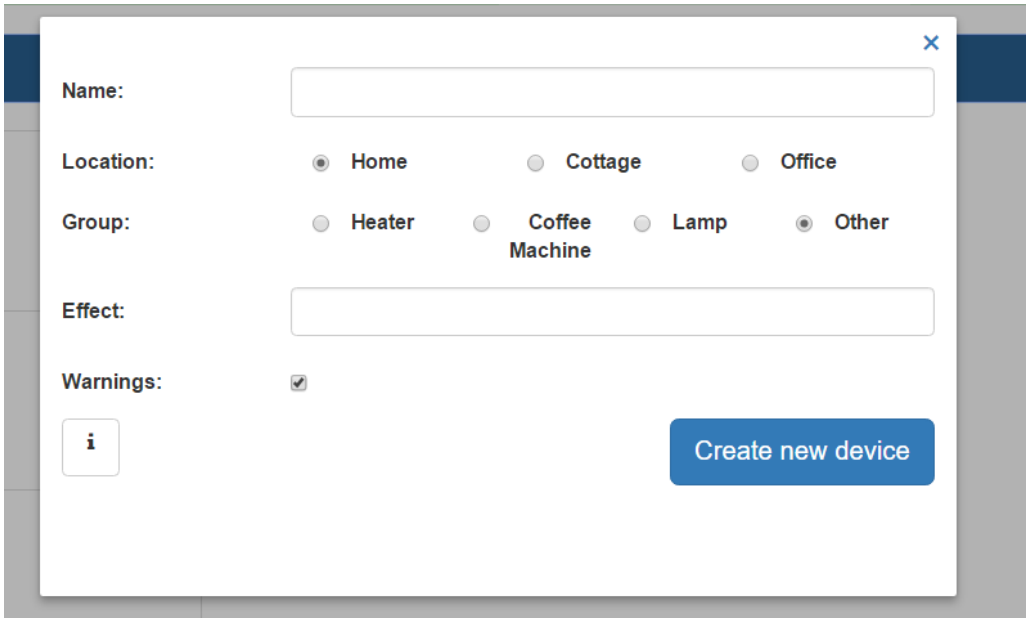
Figur 29: Representasjon av enheter - Web

I representasjonen av smartpluggen har du også metoden *handleListItemClick()* som håndterer klikk på komponenten. Metoden lar brukeren skru smartpluggen av og på og åpne en modal, se kodeeksempel 28 for implementasjonsmetoden, hvor man kunne redigere innstillingene til smartpluggen.



Figur 30: Detaljert enhetssiden - Web

Til sist har brukeren også mulighet for å legge til en ny enhet i husholdet, noe som også blir gjort med en modal som inneholder en Redux Form (se kodeeksempel 28 for implementasjonsmetoden).



Figur 31: Registrering av ny enhet - Web

6 Testing og kvalitetssikring

For prosjektet har det blitt gjort tester av både prototypene og applikasjonene, både før og underveis i utviklingen. Det har også blitt gjort tiltak for å kvalitetssikre planlegging og implementering.

6.1 Testing

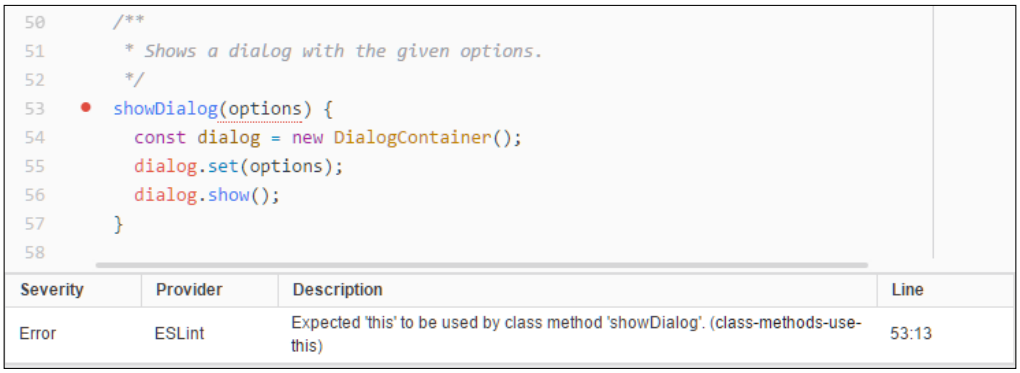
Testing i dette prosjektet har primært skjedd gjennom brukertesting av prototypene. Disse ble testet ved hjelp av venner og familie, og i tillegg viste oppdragsgiver frem prototypene til representanter fra Trønderenergi og andre bekjente. Oppdragsgiver var også aktivt med i prosessen, ved at prototypene ble tilsendt etter hvert som det ble gjort endringer av betydelig størrelse. Basert på tilbakemeldingene fra alle de sistnevnte instansene og sammenligninger med andre applikasjoner (se kapittel 4.1.1 Inspirasjon), ble protypene stadig forbedret. Ettersom det ble gjort så mye testing av brukergrensesnittene gjennom prototypene ble det ikke lagt veldig stor vekt på å gjøre tester av de implementerte brukergrensesnittene.

Under utvikling har kodetesting i stor grad vært manuell. Det vil si at når nyutviklet kode har oppført seg på en annen måte enn det som var forventet, ville utvikleren skrive ut forskjellige meldinger til konsollvinduet for debugging. For eksempel om data man får i respons stemmer overens med det som var forventet eller at riktig reducer blir trigget av en action.

6.2 Kvalitetssikring

Gjennom utviklingsprosessen har kvalitetssikring vært viktig, både når det kommer til planleggingen og utviklingen. For planleggingen har dette i stor grad forgått muntlig, ved at det har blitt holdt diskusjoner rundt de forskjellige figurene, diagrammene og prototypene som har blitt produsert underveis.

For å kvalitetssikre kildekode har det blitt gjennomført flere tiltak, blant annet bruk av ESLint, code reviews og branching. ESLint er en linter for Javascript og JSX. [43] Den analyserer koden fortløpende ut ifra et sett med regler og korrigerer feil i syntax, kodefeil, logiske feil og annet. Reglene er enkeltstående og som utvikler kan man selv velge hvilke regler man ønsker å følge, eller man kan velge å benytte seg av regelsett andre utviklere allerede har satt sammen. For dette prosjektet falt valget på å Airbnb sitt regelsett [44] da de har et av de mest kjente og største prosjektene som benytter seg av React Native. Slik slapp gruppen å bruke tid på å sette opp alle reglene selv, men kunne heller legge til eller fjerne regler fra det importerte regelsettet.



Figur 32: Eksempel på advarsel fra ESLinter, basert på regelsettet fra Airbnb

Branching betyr at man i praksis lager sin egen kopi av prosjektet, hvor man kan utvikle funksjonalitet før den inkluderes i den originale koden. Dette har blitt brukt for å sikre at uferdig kode ikke ligger er ute i produksjon, ved at enkeltfunksjonalitet får sin egen branch man kan arbeide uforstyrret i. Dette er et verktøy for at man kan enklere samarbeide på den samme kodebasen, uten at en utviklers endringer har innvirkning på det en annen utvikler arbeider med.

Code reviews derimot er en prosess hvor man får andre utviklere til å gå gjennom sin egen kildekode. Dette for å forsikre at den samsvarer med selskapets (i dette tilfellet bachelorgruppens) regler for struktur og tilnærminger til kildekode. Et problem med å samarbeide på utviklingsprosjekter er at det er mange løsninger som kan produsere et godt resultat, men det er stor forskjell på hvor lesbar kildekoden til de forskjellige tilnærmingene er. En stor del av utvikling handler om å videreutvikle eksisterende løsninger, og derfor må man også utvikle slik at koden man skriver er mulig å vedlikeholde. I dette prosjektet har det derfor blitt kjørt code reviews før ny funksjonalitet har blitt lagt til i utviklingsgrenen. I tillegg til at det øker kvaliteten på koden som produseres, ved at flere kan finne feil, lar det de andre gruppemedlemmene få en forståelse av funksjonaliteten som utvikles. På denne måten lærer man av hverandre og man blir mer bevisst på å produsere lesbar kode mens man utvikler. I code reviews for dette prosjektet har det vært like viktig at dokumentasjon er på et tilfredsstillende nivå.

I stor grad har det blitt forsøkt å skrive alt av dokumentasjon før kode produseres, slik at man er sikker på at oppgaven som skal løses er forstått i forkant. Selvfølgelig er det ikke alltid man treffer når det kommer til denne tilnærmingen for dokumentasjon, men derfor er det også helt åpent for å endre dokumentasjonen i ettertid. Det har blitt satt et krav til dokumentasjon før ny funksjonalitet tillates i produksjonsgrenen, dermed blir det lettere å gjøre endringer i ettertid. Alle kommentarene i kildekoden er også skrevet i JavaDoc stil, fordi alle gruppemedlemmene allerede var godt inneforstått med denne type dokumentering. Denne måten å skrive kommentarer på sørger for at den som skriver koden forklarer i detalj hvordan de forskjellige delene av kildekoden fungerer. Mer spesifikt til React og React Native så har det blitt brukt noe som kalles for PropTypes. PropTypes passer på at props som brukes i komponenter er av korrekt type. Dette er satt opp slik at når man lager komponenter vil man fortløpene fortelle komponenten hvilke

```
/**
 * Adds to numbers together.
 * @param  a The first number to be added.
 * @param  b The second number to be added.
 * @return sum The result of the addition.
 * @see https://en.wikipedia.org/wiki/Addition
 */
function sum(a, b) {
  return a + b;
}
```

Kodeeksempel 26: Eksempel på kommentar skrevet i JavaDoc stil.

typer de forskjellige propsene skal ha. Om man da prøver å ta i bruk en komponent og sender inn en prop av feil type, vil man få en advarsel om dette. På denne måten er man sikker på at komponentene har props av korrekt type, og man blir selv mer bevisst på hva komponentene skal inneholde av forskjellige props.

```
1  MyComponent.propTypes = {
2    optionalBool: React.PropTypes.bool,
3    optionalFunc: React.PropTypes.func,
4    optionalNumber: React.PropTypes.number,
5    optionalString: React.PropTypes.string,
6
7    date: React.PropTypes.oneOfType([
8      React.PropTypes.string,
9      React.PropTypes.number,
10   ]),
11
12   dataObject: React.PropTypes.shape({
13     color: React.PropTypes.string,
14     fontSize: React.PropTypes.number
15   }),
16
17   callback: React.PropTypes.func.isRequired,
18 }
```

Kodeeksempel 27: Eksempel på bruk av PropTypes, modifisert fra den offisielle dokumentasjonen til React. [45]

I kodeeksempel 27 er det lagt ved et eksempel på hvordan man bruker PropTypes. Her vises blant annet muligheten for å fortelle komponenten at den skal få inn props av typene bool, func, number og string. Det er også mulighet for at en prop kan være av flere typer om nødvendig, som her er vist på linje 7 med metoden *oneOfType()*. Denne metoden forteller at en dato kan være enten en tekststreng eller et tall. På linje 12 vises også hvordan man kan fortelle hvilke verdier som forventes innenfor et objekt med *shape()* metoden. Til slutt på linje 17 vises det at man kan kreve at en prop er sendt med til komponenten, ved å legge til *isRequired*, for at komponenten skal fungere som forventet.

7 Avslutning

I dette kapitlet vil aspekter i og rundt gjennomførelsen av prosjektet tas opp. Dette vil da bestå av hvilke resultater som ble oppnådd og hvilket læringsutbytte dette prosjektet har hatt for gruppen. Det vil også bli tatt for seg kritikk av oppgaven og en evaluering av gruppens arbeidsmetoder før man helt til slutt kommer med en konklusjonen.

7.1 Diskusjon

7.1.1 Resultater

Til å begynne med bestod oppgaven av å skape en kryssplattform applikasjon for både iOS, Android og Web, som nevnt i oppgavebeskrivelsen. Applikasjonen skulle ta for seg kommunikasjonen med smartpluggene og ha et intuitivt og enkelt brukergrensesnitt. Ved endt prosjektperiode har mobilapplikasjonen mye av brukergrensesnittet ferdig implementert, i tillegg er oppkobling mot Firebase for brukerhåndtering og datalagring klart. Applikasjonene er også satt opp slik at det skal være enkelt å koble seg opp mot APIet til server når dette er ferdig utviklet. Webapplikasjonen er litt lengre unna en ferdigstillelse av brukergrensesnittet enn mobilapplikasjonen.

I henhold til læringsmålene for IMT3912 har vi integrert flere aspekter fra studieprogrammenes innhold. Blant annet planlagt gjennomføringen av prosjektet på en systematisk og strukturert måte og fulgt denne planen så langt det lot seg gjøre. I tillegg benyttet vi oss av opparbeidet kunnskap innenfor programvareutvikling generelt, med tanke på struktur og dokumentering av kildekode. Det å utvikle programvare på en profesjonell måte er en prosess ingen av oss har fått føle på skikkelig før denne bacheloroppgaven, men ved endt prosjektperiode kan vi med sikkerhet si at vi opptrer langt mer profesjonelt i våre arbeidsmetoder. Fra å benytte Jira som prosjektsstryingsverktøy og branching til gjennomføring av code reviews for ny funksjonalitet.

Av erfaringer har vi fått opplevd det å arbeide sammen på større prosjekter. Det å ha flere instanser å forholde seg til over lengre perioder med tanke på hverandre, veiledere, oppdragsgiver og andre. Dette har vist seg å være utfordrende, men en veldig god erfaring å ta med videre.

Hvis man ser på fremdriftsplanen (se vedlegg K side 15) ser man at planen for prosjektet var å bruke første sprinten på å designe applikasjonen for både web og mobil. Dette var noe vi ikke klarte å gjøre på to uker, men brukte i stedet nærmere to sprinter på dette. I den første sprinten ble litt over halve designet ferdigstilt mens det ble forbedret basert på tilbakemeldinger fra familie, arbeidsgiver og andre i den neste sprinten. Når vi startet med utviklingen viste det seg at estimeringen på hvor lang tid vi trengte på å lage brukergrensesnittet ble feil.

Det var også planlagt at vi skulle lage serveren og koble pluggene opp mot den, men dette var noe som aldri ble gjennomført. Både fordi pluggene ble forsinket og det viste seg at arbeidet med implementeringen av brukergrensesnittene var mer omfattende enn først antatt. Dermed fokuserte vi på fullførelsen av brukergrensesnittene.

Etter å ha programmert i React og React Native føler vi at dette var det riktige valget for oppgaven. Det tok tid å sette seg inn i temaet, men det var en lettere måte å samkjøre et todelt prosjekt. Ettersom React Native bygger på React er det naturlig nok store likheter i den grunnleggende logikken. Dette gjorde at prosjektene kunne struktureres på lignende måte.

Fra prosjektplanen ble tre resultatmål definert:

- Den ferdige applikasjonen skal være skalerbar og ha versjoner for både nettleser og mobile operativsystemer (Android og iOS).
- Brukerne skal kunne ha tilgang til systemet sitt fra hvor som helst.
- Applikasjonen skal være intuitiv og enkel å bruke for alle, uavhengig av bakgrunn og alder.

Selv om applikasjonene ikke er ferdigstilte vil vi påstå at de dekker kravet om å være skalerbare. Det vil etter vår mening være enkelt å utvide applikasjonene med ny funksjonalitet i senere tid, uten at dette skal skape problemer. Utviklingen av brukergrensesnittet har også fulgt prinsippene for universal utforming og skal være mulig å bruke for folk, uavhengig av alder.

Effektmålene (se vedlegg K, side 1) er vanskelige å måle siden de tar utgangspunkt for at applikasjonen er ferdig utviklet og lansert.

7.1.2 Alternativer

Under planleggingen av prosjektet lagde vi personas som user stories ble basert på. Problemet var at vi glemte det nødvendige forarbeidet ved å intervjue potensielle brukere for applikasjonene. Dermed ble personas oppdiktet, og heller ikke representativt for en eventuell brukergruppe. Dette ble påpekt da vi kontaktet Eivind Johansen for gjennomgang av disse og prototypene som ble laget. Personas ble derfor forkastet et stykke ut i planleggingsperioden da det allerede hadde blitt investert mye tid i prototypene. Vi hadde nå kommet såpass langt ute i prosjektet at det ikke var forsvarlig å bruke mer tid på dette. Spesielt med tanke på at oppdragsgiver var fornøyd med det vi hadde presentert. Oppdragsgiver hadde også vist frem prototypene til potensielle kunder og fått positive tilbakemeldinger.

Bakgrunnen for valget av Scrum som utviklingsmodell var basert på et ønske fra oppdragsgiver om å lage modulbaserte applikasjoner, som nevnt i 1.8.2 Arbeidsmetoder. Vi så at Scrum passet til prosjektet fordi det ga oss muligheten til å ta for oss mindre arbeidsmengder av gangen, der vi fikk praktisert og forbedret det å estimere hvor lang tid forskjellige arbeidsoppgaver ville ta. Scrum virket også motiverende da vi kunne se tydelig fremgang ved slutten av hver sprint. Artefaktene sprint planning meeting og sprint review meeting ga oss mulighet til å diskutere oppgavene som skulle påbegynnes eller

gå over det som hadde blitt gjort. I tillegg har Sprint retrospective fått oss til å reflektere over hva vi ønsket å ta med videre til neste sprint og hva som må forbedres. Andre utviklingsmodeller ble vurdert, men vi syntes ikke at de passet for denne oppgaven. Enten var de for omfattende eller hadde for strenge rammer (RUP, Fossefall) for et lite team eller så baserte de seg på at oppdragsgiver alltid måtte være tilgjengelig (Extreme).

7.2 Kritikk av oppgaven

Ved endt prosjektperiode er det en del ting vi ser kunne ha gjort annerledes for å oppnå et mer tilfredsstillende resultat. Her kommer vi til å ta for oss problemer med vår tilnærming til gjennomføring av oppgaven, og kritikk til oppgaven i seg selv.

Bedre arbeidsfordeling

Til å begynne med ville vi ha begynt prosjektet med å splitte opp arbeidsoppgaver i større grad enn det vi endte opp med å gjøre. Slik vi tilnærmet oss oppgaven var at først skulle planlegging og prototyping gjennomføres, deretter skulle utvikling begynne og til slutt skulle vi begynne på rapportskrivingen. Det dette førte til var at alle samarbeidet om å fullføre de forskjellige fasene av prosjektet, som i seg selv er positivt, men endte opp med at tidsfordelingen ble ujevn. Det at vi angrep oppgaven med en tanke om at vi skulle bli helt ferdig med en og en fase, gjorde at når planleggingsfasen ble lengre enn først antatt. Dermed ble det også mindre tid til de andre fasene. Selv om vi syntes det fungerte å samarbeide om alt underveis, var det heller ingen som fikk noe følelse av hvor lang tid utvikling og rapportskriving ville ta fordi ingen av oss arbeidet med dette før planleggingen var ferdig. Hadde vi heller satt det opp slik at noen kunne planlegge mens andre utviklet, ville den som utviklet kunne gitt en bedre indikasjon på hvor lang tid det tar å gjennomføre de forskjellige oppgavene.

Reduksjon av oppgaveomfang

Vi merket også utover i utviklingsfasen at vi heller skulle valgt å fokusere på å ferdigstille applikasjonen for en av plattformene. Det at vi prøvde å få ferdig både mobil og webapplikasjonen gjorde at vi heller endte opp med to halvferdige applikasjoner, når vi heller kunne ha kommet nærmere målene våre for en av applikasjonene. Det at vi prøvde å utvikle for begge plattformene gjorde at det ble brukt ekstra mye tid på planlegging og prototyping i starten av prosjektet, og dermed ble det også lite tid til å implementere alt vi hadde sett for oss. Hadde vi redusert oppgaveomfanget tidlig i oppgaven, kunne vi kjapt ha kommet igang med utvikling mye tidligere og hatt en mer fullverdig applikasjon å vise frem. Nå må det sies at dette er det største prosjektet vi har vært borte i, så vi var naturlig nok litt ambisiøse i være estimeringer. Mye av dette kommer rett og slett fra manglende erfaring med slike prosjekter.

Ettersom dette prosjektet har vært et av de viktigste prosjektene vi har vært borti, så ønsket vi å gjøre alt ordentlig. Dessverre gjorde denne tankegangen at vi brukte overdrevent mye tid på enkelte oppgaver, som kunne ha blitt unnagjort mye kjappere. For eksempel prototypene for brukergrensensnitt endte opp med å ta 2-3 arbeidsuker, og dette er tid som kunne ha blitt brukt på en annen måte. Problemet er egentlig ikke at prototypene i seg selv tok tid, for både vi og oppdragsgiver er veldig fornøyde med slik ting ble seendes ut. Når grensesnittene var av en slik grad av detalj, følte vi at disse

måtte følges til punkt og prikke når vi skulle implementere brukergrensesnittene. Dette var spesielt utfordrende med tanke på at utviklingen foregikk i rammeverk ingen av oss hadde noe erfaring med i forkant av prosjektet.

Tunnelsyn

En annen ting vi skulle innsett tidligere i prosjektet var det med at smartpluggene ikke var essensielle for at vi kunne begynne på utviklingen. Vi tenkte at vi kunne fortsette med utviklingen av brukergrensesnittene mens vi ventet på at smartpluggene fra oppdragsgiver skulle bli ferdige. Da det nærmet seg slutten av mars ble det klart at smartpluggene mest sannsynlig ikke ville bli tilgjengelig i løpet av prosjektperioden. Alternativet da var å simulere protokollene smartpluggene benyttet seg av, slik at vi kunne få på plass mye logikk rundt kommunikasjon med disse. Problemet var at vi oppfattet det slik at en server (som i senere tid ble bestemt skulle utvikles av oppdragsgiver) skulle ta seg av dette, og at vi heller ville kommunisere med smartpluggene via serveren. Ettersom server ikke var vårt ansvar lengre fikk vi heller opprettet en databasestruktur i Firebase i samarbeid med oppdragsgiver. Da hadde vi allerede kommer ut i begynnelsen av april og det begynte å bli lite tid igjen til utvikling, fordi vi måtte bruke den resterende tiden frem til innlevering på rapportskriving.

Annet

Det har også blitt en del ekstra arbeid for at vi skulle prøve å opptre som profesjonelle utviklere, i og med at vi gjennomførte kurset Professional Programming (IMT3602) ved siden av bacheloroppgaven. Dette faget tok opp mange nyttige temaer som vi prøvde å anvende i prosjektet, noe som førte til en del ekstra hodebry. Ting som branching, pull requests og code reviews var ikke anvendt av noen av oss før bacheloroppgaven. Vi ser på alle disse temaene som veldig nyttig, men det er noe vi skulle ønske var satt et sterkere fokus på tidligere i studiet.

7.3 Videre arbeid

Som rapporten allerede har antydet er det fortsatt en del arbeidsoppgaver som ikke er fullført. Til å begynne med er det mye av kommunikasjon til server (som ikke er klar ved prosjektets slutt) som må implementeres i begge applikasjonene. Applikasjonene er satt opp slik at man kan kjapt legge inn nye *action creators* og *reducere* for å ta seg av dette, men det er vanskelig å få implementert det uten at det finnes et API man kan kommunisere med. For webapplikasjonen er det også igjen en del arbeid med brukergrensesnitt på forskjellige områder, mens på mobil mangler det en måte for å vise frem grafer og kalenderen.

Det kreves også en del arbeid for at applikasjonene skal ta i bruk dataene fra Firebase og få applikasjonen til å oppdatere disse dataene. Dette ble det påbegynt utvikling for, men er høyt eksperimentell og uferdig kode. Det applikasjonene viser av informasjon i dag er bare såkalte 'dummy' data fra ressursfiler som ligger lokalt i applikasjonene. Det må også implementeres metoder for å sette inn data til Firebase, ved å bruke brukergrensesnittene som vi har utviklet. Applikasjonene har allerede mange av de forskjellige skjemaene klare visuelt sett, men enn så lenge sender de ikke informasjonen noe sted.

Kalenderen og hvilken funksjonalitet den skal ha er bestemt i noen grad, men det er en del som må gjøres for å implementere denne funksjonaliteten. For det første må serveren som tar for seg kommunikasjonen med smartpluggene ferdigstilles. Denne serveren må for kalenderen sin del inneholde mye av logikken rundt selve automatiseringen av modus. I tillegg må serveren ha et API for at applikasjonene skal kunne konfigurere når forskjellige modus skal være aktive og for å hente ut informasjon om eksisterende konfigurasjoner, slik at det kan vises frem i brukergrensesnittene for de forskjellige applikasjonene.

7.4 Evaluering av gruppen

7.4.1 Introduksjon

Prosjektet startet naturlig nok med opprettelse av grupperegler og prosjektplanleggingen. Skrivingen av prosjektplanen (vedlegg K) gjorde at mye av det administrative blant gruppen ble bragt opp og diskutert. Blant annet grupperegler, valg av utviklingsmodell, vårt formål meg oppgaven og grove estimeringer av når de forskjellige fasene av prosjektet skulle være fullført. Slik fikk vi innsikt i hva oppgaven omhandlet, og hvordan vi tenkte å angripe den.

Det ble også bestemt at vi skulle møte på skolen hver eneste arbeidsdag slik at alle til enhver tid skulle være inkludert i det de andre arbeidet med. En slik tilnærming synes vi har fungert for det har vært enkelt å ta kontakt med de andre gruppemedlemmene, enten om det noe man ikke helt vet svaret på eller om man ikke kommer seg videre med en oppgave.

Arbeidsoppgavene for dette prosjektet har, som nevnt flere steder i rapporten, blitt lagt i en product backlog. Slik vi tilnærmet oss oppgaven har vi lagt til nye oppgaver i denne køen underveis i prosjektet, hovedsakelig under planleggingsmøtene før hver sprint. Det vi oppdaget med å arbeide på en slik måte var at våre estimeringer ikke var realistiske. Dermed valgte vi å redusere lengden på sprintene slik at det ble lettere å estimere hvor mye vi kunne ta for oss i forkant av hver sprint. Med redusert lengde på sprintene og med erfaring, har vi kommet nærmere målene våre ved de forskjellige sprintenes slutt.

Gjennom perioden har vi holdt god kontakt med både oppdragsgiver og veileder, slik at de også har fått et innblikk i hva vi har arbeidet med til forskjellige tider. Oppdragsgiver har vært veldig imøtekommende, som har gjort det enkelt å planlegge møter eller kommunisere via epost selv om det til tider ble på kort varsel.

Prosjektet har stort sett gått bra, selv om vi helst skulle hatt mer tid til utvikling. Motivasjonen har definitivt vært høyest under utviklingen, men det har ikke vært noen særlige konflikter internt. Vi har stort sett kommer til enighet om forskjellige spørsmål og tilnærminger, og der vi ikke har blitt enige synes vi det har vært rettferdig å la flertallet bestemme.

7.4.2 Organisering

Alt som har blitt gjort av arbeid har blitt lagt inn i en felles mappe på Google Drive (utenom kildekode), og i tillegg har Google Sheets blitt brukt til å loggføre hva hver av oss har arbeidet med til enhver tid. Dette synes vi har fungert bra for da har alle tilgang til alt av dokumenter til enhver tid. Det å ta i bruk skylagring har gjort det slik at dokumentene alltid er tilgjengelig for alle, og vi behøver ikke å bekymre oss for at dokumenter forsvinner som et resultat av feil ved maskinvare, eller at forskjellige papirer forsvinner. Vi endte også opp med å bruke ShareLaTeX [46] for rapportskrivingen, for i likhet med Google sine løsninger får vi sett hva de andre gruppemedlemmene skriver i sanntid. Timeføringen har også fungert bra i Google Sheets etter at vi lagde et oppsett for å føre inn alt vi trengte av informasjon, med noen regneformler for lett statistikk.

Til kommunikasjon internt i gruppen utenom arbeidstidene har stort sett Messenger (samtale via Facebook) blitt brukt, i det tilfelle at noen ikke kunne møte, om problemer oppsto, eller for samtaler generelt mellom gruppemedlemmene. Dette har fungert bra for det har gjort at alle er tilgjengelige til enhver tid om det skulle være nødvendig. Kommunikasjon med arbeidsgiver forgikk stort sett via epost og nettsiden appear.in. Denne nettsiden lar oss strømme video og lyd, slik at møtene føltes mer personlig selv om oppdragsgiver befinner seg i Trondheim. Vi synes slike møter har gjort det enklere å ta kontakten med oppdragsgiver, og har gjort at vi har et avslappet forhold til hverandre.

7.4.3 Arbeidsfordeling

For å fordele arbeidet har vi stort sett respektert ønskene til hverandre, slik at alle kan få arbeide med det de selv vil fra tid til annen. Ettersom alle har vært inkludert i hverandres oppgaver gjennom hele prosjektet, er det egentlig ingen som har vært alene på en arbeidsoppgave. Vi la det også opp slik at hovedansvaret for de større oppgavene var hos enkeltpersoner, slik at personen kunne virkelig fordype seg uten å bli forstyrret av andre oppgaver. Erfaringen til gruppemedlemmene ble også tatt i betrakting for å dele ut oppgaver, men ettersom oppgaven omhandlet områder hvor vi stort sett sto likt var det ikke klart å si hvem som kunne utføre en oppgave bedre eller kjappere enn de andre gruppemedlemmene. Det eneste stedet vi faktisk så at erfaringsbasert inndeling var relevant, var i forkant av utviklingsfasen. Der to av oss allerede hadde noe erfaring med utvikling av applikasjoner for mobil takket være studiene og valgfag. Derfor ble det naturlig at de med erfaring for mobil fikk ansvaret for å utvikle mobilapplikasjonen.

7.4.4 Prosjektet som arbeidsform

Prosjektet som arbeidsform har i stor grad vært en positiv erfaring. Det å jobbe i grupper gjør at oppgaven har hatt et stort nok omfang, slik at vi har måttet angripe oppgaven på en strukturert måte fra start til slutt. Vi tror heller ikke at mindre grupper eller individuelt arbeid ville gitt en like realitisk erfaring med utviklingsprosjekter, fordi oppgaveomfanget må naturligvis skalere med antall personer på gruppene. Det å arbeide i grupper gjør det slik at oppgavene kan i den grad av detalj, der oppgaven gir oss en realistisk innføring i hvordan utviklingsprosjekter faktisk foregår.

7.5 Konklusjon

For å oppsummere dette prosjektet så sitter vi igjen med med to applikasjoner som fremdeles trenger videreutvikling, men hvor det har blitt lagt et godt utgangspunkt for å gjennomføre den resterende utviklingen. Vi har kommet frem til et brukergrensesnitt og databasestruktur som vi er fornøyde med.

Prosjektet har latt oss bruke mye av kunnskapene opparbeidet gjennom studiene, men også bidratt til at vi har lært mye nytt. Vi har tilegnet oss nye kunnskaper, utvidet eksisterende kunnskapsområder og fått mange erfaringer rundt det å arbeide med større prosjekter. Den største utfordringen med prosjektet har vært estimeringen og vi ser nå hvorfor prosjekter sliter med å opprettholde tidsfrister.

Vi håper at oppdragsgiver setter pris på arbeidet som har blitt gjort, selv om vi skulle ønske vi hadde kommet lengre med oppgaven.

Bibliografi

[1] Ashton, K. June 2009. That 'Internet of Things' Thing. http://www.rfidjournal.com/articles/view?4986. (Sist besøkt Mai 2017).

[2] The First Coke Machine on the Internet. https://www.cs.cmu.edu/~coke/history_long.txt. (Sist besøkt Mai 2017).

[3] The Internet Toaster. http://www.livinginternet.com/i/ia_myths_toast.htm. (Sist besøkt Mai 2017).

[4] Internet of things. https://en.wikipedia.org/w/index.php?title=Internet_of_things&oldid=777092697. (Sist besøkt Mai 2017).

[5] IMT2243 - Systemutvikling. https://www.ntnu.no/studier/emner/IMT2243#tab=omEmnet. Emneansvarlig Tom Røise v/NTNU i Gjøvik.

[6] Grider, S. Modern react with redux. https://www.udemy.com/react-redux/. (Sist besøkt Mai 2017).

[7] Grider, S. The complete react native and redux course. https://www.udemy.com/the-complete-react-native-and-redux-course/. (Sist besøkt Mai 2017).

[8] International Data Corporation. 2016. Smartphone OS Market Share, 2016 Q3. http://www.idc.com/promo/smartphone-market-share/os. (Sist besøkt Mai 2017).

[9] Facebook. facebook/react-native. https://github.com/facebook/react-native. (Sist besøkt Mai 2017).

[10] Android developers. Dashboard. https://developer.android.com/about/dashboards/index.html. (Sist besøkt Mai 2017).

[11] App success made simple. https://firebase.google.com/. (Sist besøkt Mai 2017).

[12] Leading social networks worldwide as of april 2017, ranked by number of active users (in millions). https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/. (Sist besøkt Mai 2017).

[13] 2013. http://tech.yeesiang.com/wp-content/uploads/2013/11/bootstrap-login.png. (Sist besøkt Mai 2017).

[14] Facebook. https://developers.facebook.com/docs/facebook-login. (Sist besøkt Mai 2017).

[15] Twitter. https://dev.twitter.com/web/sign-in. (Sist besøkt Mai 2017).

[16] Google. https://developers.google.com/identity/protocols/OpenIDConnect. (Sist besøkt Mai 2017).

[17] Nash, J. Thinking about using social logins on your website? http://www.crescentinteractive.com/using-social-logins/. (Sist besøkt Mai 2017).

[18] Gentz, M. & Rabeler, C. Nosql vs sql. https://docs.microsoft.com/en-us/azure/documentdb/documentdb-nosql-vs-sql. (Sist besøkt Mai 2017).

[19] Nosql vs sql overview figure. https://docs.microsoft.com/en-us/azure/documentdb/media/documentdb-nosql-vs-sql/nosql-vs-sql-overview.png. (Sist besøkt Mai 2017).

[20] WCAG. Wcag 2.0 - standarden. https://uu.difi.no/krav-og-regelverk/wcag-20-standarden. (Sist besøkt Mai 2017).

[21] Hoober, S. How do users really hold mobile devices? http://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php. (Sist besøkt Mai 2017).

[22] Hoober, S. Summary of how people hold and interact with mobile phones. http://www.uxmatters.com/mt/archives/2013/02/images/HoldPhones_Figure-1.png. (Sist besøkt Mai 2017).

[23] 4+1 architectural view model. https://en.wikipedia.org/wiki/4%2B1_architectural_view_model. (Sist besøkt Mai 2017).

[24] Atom.io. https://www.atom.io/. (Sist besøkt Mai 2017).

[25] Webpack Dev Server. https://webpack.github.io/docs/webpack-dev-server.html. (Sist besøkt Mai 2017).

[26] Facebook. React. https://facebook.github.io/react/. (Sist besøkt Mai 2017).

[27] Es6. http://es6-features.org/. (Sist besøkt Mai 2017).

[28] NPM. https://www.npmjs.com/. (Sist besøkt Mai 2017).

[29] Redux. http://redux.js.org/. (Sist besøkt Mai 2017).

[30] React Redux. https://github.com/reactjs/react-redux. (Sist besøkt Mai 2017).

[31] Redux Thunk. https://github.com/gaearon/redux-thunk. (Sist besøkt Mai 2017).

[32] Thunk. https://en.wikipedia.org/w/index.php?title=Thunk&oldid=776106662. (Sist besøkt Mai, 2017).

[33] Firebase. Read and Write Data on the Web. https://firebase.google.com/docs/database/web/read-and-write. (Sist besøkt Mai 2017).

[34] React Intl. https://github.com/yahoo/react-intl. (Sist besøkt Mai, 2017).

[35] React Router. https://github.com/reacttraining/react-router. (Sist besøkt Mai, 2017).

[36] React Router Redux. https://github.com/reacttraining/react-router. (Sist besøkt Mai, 2017).

[37] Mozilla. History API. https://developer.mozilla.org/en-US/docs/Web/API/History. (Sist besøkt Mai, 2017).

[38] Redux Form. https://github.com/erikras/redux-form. (Sist besøkt Mai, 2017).

[39] Font Awesome. http://fontawesome.io/. (Sist besøkt Mai, 2017).

[40] React-Select. https://github.com/lovedota/react-bootstrap-select. (Sist besøkt Mai, 2017).

[41] Recharts. https://github.com/recharts/recharts. (Sist besøkt Mai, 2017).

[42] React Skylight. https://github.com/marcio/react-skylight. (Sist besøkt Mai, 2017).

[43] ESLint. ESLint - About. http://eslint.org/about/. (Sist besøkt Mai 2017).

[44] Airbnb. airbnb/javascript. https://github.com/airbnb/javascript/tree/master/packages/eslint-config-airbnb. (Sist besøkt Mai, 2017).

[45] Typechecking with proptypes. https://reactjs.vn/react/docs/typechecking-with-proptypes.html. (Sist besøkt Mai 2017).

[46] ShareLaTeX. https://www.sharelatex.com/. (Sist besøkt Mai 2017).

A Terminologi

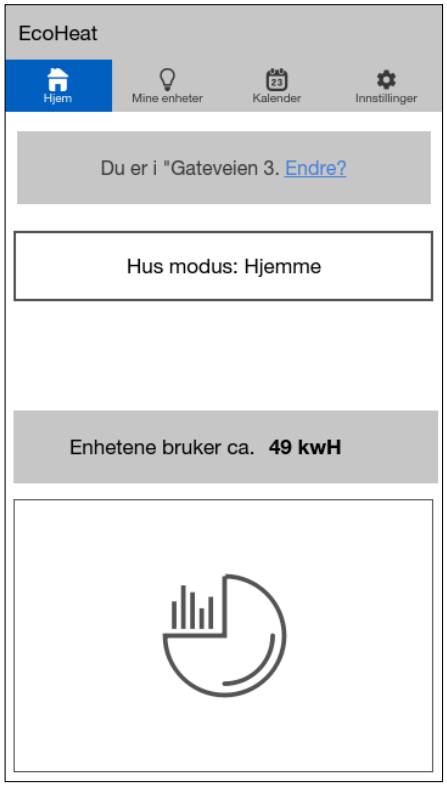
I dette vedlegget finnes det en liste med forklaringer til ord og uttrykk brukt i denne rapporten, med tanke på at leseren har omtrentlig lik bakgrunn innenfor IT som bachelorgruppen.

Tabell 1: Terminologi

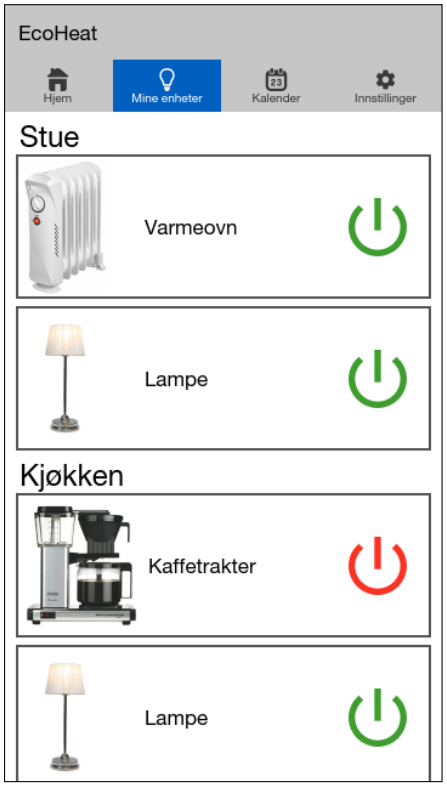
Begrep	Forklaring
Action	Javascript objekt med data som sendes fra applikasjonen til Redux store.
Action creator	Funksjoner som skaper Actions..
Code review	Det å la andre evaluere din kode. I dette prosjektet: Hele gruppen vurderer koden som en av medlemmene av laget for å ferdigstille en funksjon.
Firebase	Utviklerplattform fra Google som tilbyr autentisering, datalagring m.m.
Hi-Fi	High Fidelity
IDC	International Data Corporation
IoT	Internet of Things (norsk: Tingenes internett). At alt av forskjellige enheter blir koblet opp mot internett med sin egen unike IP-adresse, hvor de så kan kommunisere og utveklse informasjon.
JSON	Javascript Object Notation.
Lo-Fi	Low fidelity
MQTT	Message Queue Telemetry Transport. Maskin til masking og IoE kommunikasjonsprotokoll
NoSQL	Not only SQL.
NPM	Pakkehåndteringssystem for Node.js
Product backlog	En liste over ting som trenger å bli gjort. For eksempel ny funksjonalitet, eller bugfix.
Props	Parametere man kan sende til React komponenter, for å tilpasse komponeneten eller overføre data mellom komponentene.
QoS	Quality of Service
Reducer	Kontinuerlig lytter etter Actions fra Redux Store. Modifiserer applikasjonstilstanden når en action ankommer.
RDBMS	Relational database management system.
React	Deklarativt Javascript bibliotek for å bygge brukergrensesnitt. Utviklet av Facebook.
React Native	Javascript rammeverk for å bygge native applikasjoner for Android og iOS ved hjelp av React.
Redux	Container for å håndtere tilstanden til Javascript applikasjoner.
Redux Store	Objekt som inneholder hele tilstandstreet for Redux applikasjoner.
Smartplugg	Plugg som sitter mellom stikkontakter og elektroniske artikler. Den fullførte versjonen vil ha både Wi-Fi tilkobling og termostat slik at den kan styres over internett. Produsert av Hark Technologies.
SQL	Structured Query Language.
Tabbar	Navigasjonsbar brukt til hovednavigasjon i applikasjoner, som kan bestå av ikoner og tekst. Benyttet i mobilapplikasjonen.
VCS	Version Control System.
WCAG	Web Content Accessibility Guidelines. Retningslinjer for universell utforming med tanke på at det som skapes av løsninger skal være tilgjengelig for **alle** potensielle brukere.

B Lo-Fi Prototype for GUI

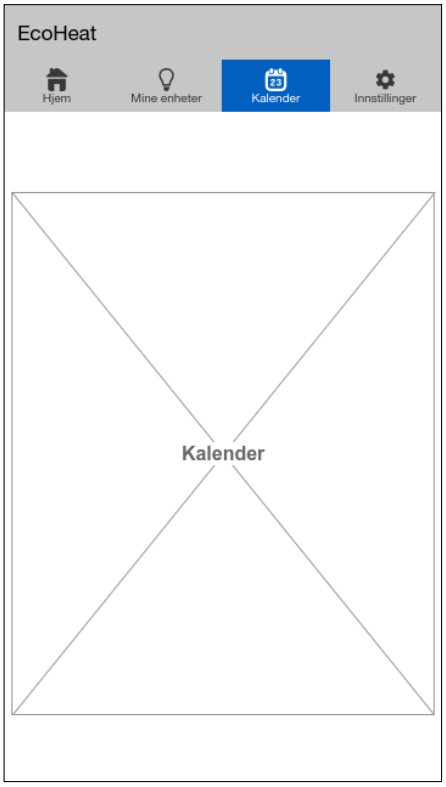
Dette vedlegget består av Lo-Fi prototypen laget for brukergrensesnittet, og ble brukt for å kjapt vise oppdragsgiver forslag til utseende for mobilapplikasjonen (Lo-Fi for web ble dessverre ikke lagret). Lo-Fi prototypen var interaktiv til en viss grad, demonstrert her https://pr.to/ARB8UH/.



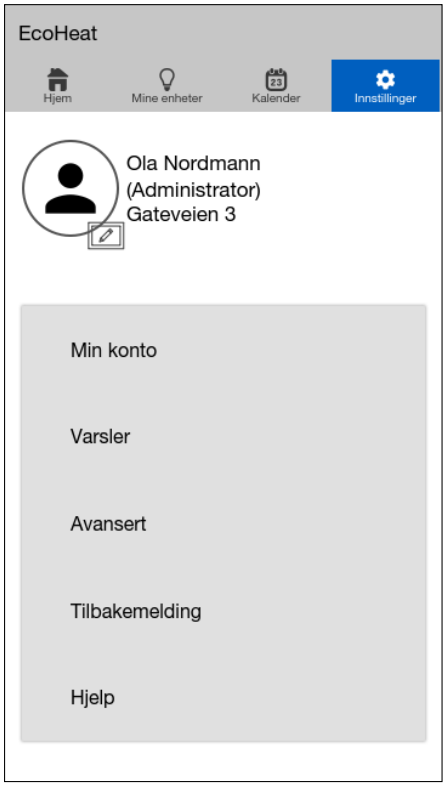
Figur 1: Denne figuren viser Hjem skjermen for mobilapplikasjonen, med rom for litt generell statistikk om det valgte husholdet



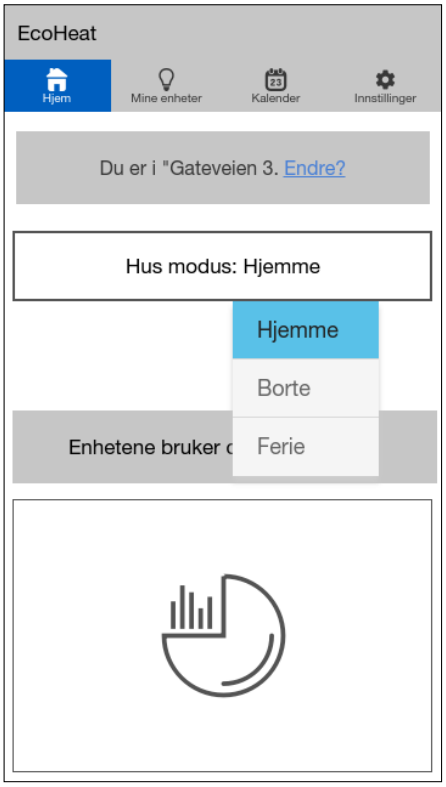
Figur 2: Trykker man på 'Mine enheter' for man opp listen over enhetene i det valgte husholdet. Øverst er det mulighet for å legge til en ny enhet og i midten er selve enhetslisten.



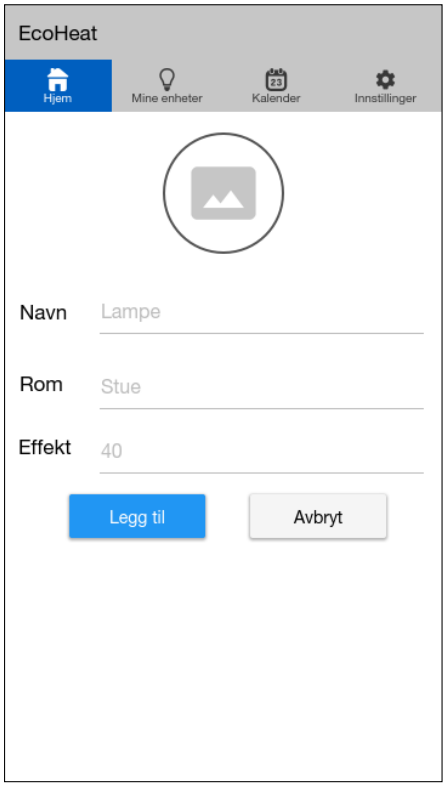
Figur 3: Kalender skjermen ble ikke fullført i Lo-Fi prototypen, pga. uklarhet om hvilken funksjonalitet kalenderen skulle ha.



Figur 4: Trykker man på 'Innstillinger' får man opp en rekke menyvalg for å navigere videre, navigasjonen ble ikke laget for Lo-Fi prototypen.



Figur 5: Hjem skjermen hvor brukeren har trykket på knappen for å endre hvilket hushold som er valgt for øyeblikket. En dropdown meny dukker opp, med mulighet for å velge mellom alle husholdene en bruker er del av.

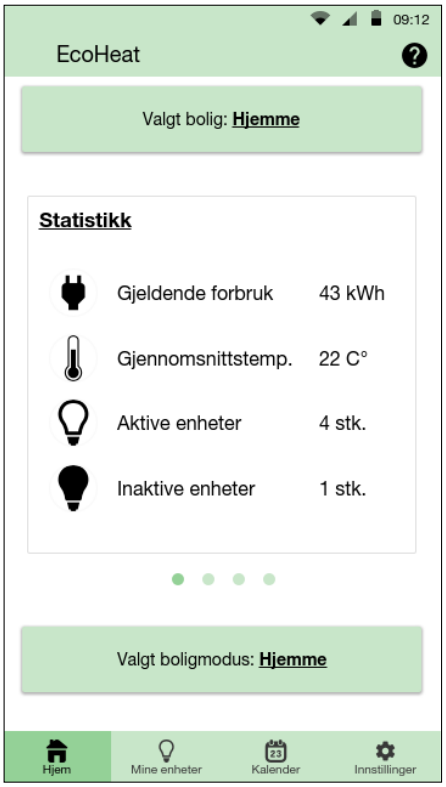


Figur 6: Trykker man på 'Registrer en ny enhet' fra 'Mine enheter' blir man navigert til en skjerm der man kan skrive inn litt informasjon om enheten.

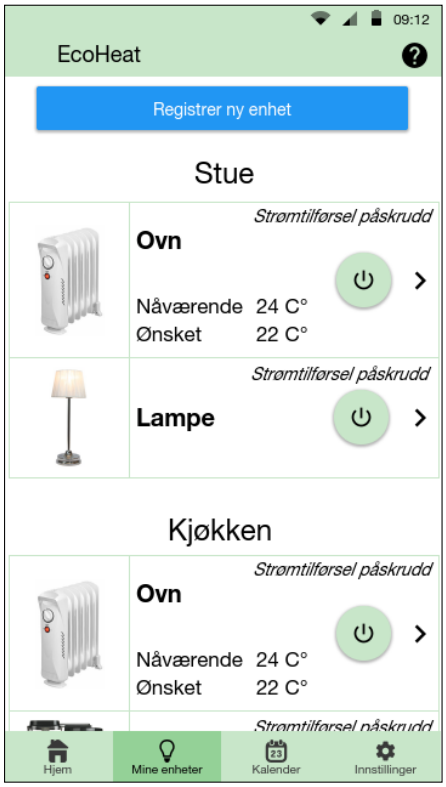
Figur 7: Ved å trykke på en enhet fra 'Mine Enheter', blir man navigert til en skjerm med mulighet for å se detaljer for den valgte enheten.

C Hi-Fi Prototype for GUI

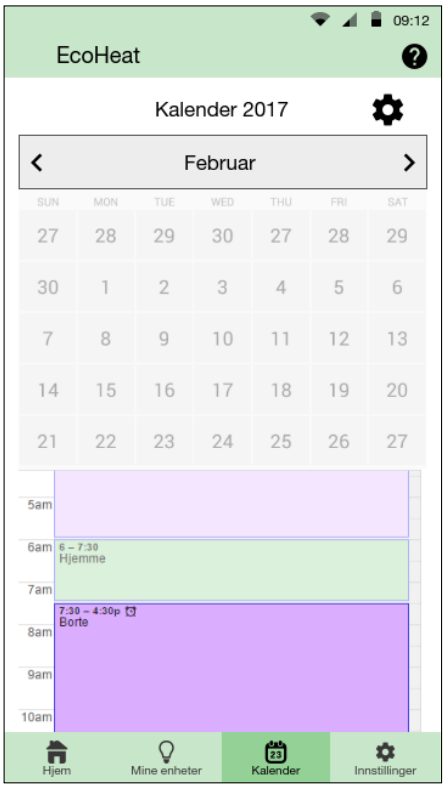
Dette vedlegget består av skjermbilder av alle mulige skjermer for både mobil (de fleste skjermbildene for web er allerede innlagt i 4.1.4 Web). Dette inkluderer hvordan hver hovedskjerm så ut, og alle skjermene man kunne navigere til fra hovedskjermene. Prototypene ble laget ved hjelp av den betalte versjonen av verktøyet http://www.Proto.io, som gjorde at den var interaktiv og virket som en ekte applikasjon. Designet var en viktig del av oppgaven, og derfor er det lagt mye tid i å lage de forskjellige prototypene. Bildene er egentlig ikke godt nok til å vise frem prototypen, så det anbefales å prøve den ut i en nettleser her: https://pr.to/M6H6SZ/ (mobil) og https://pr.to/86JX7P/(web)



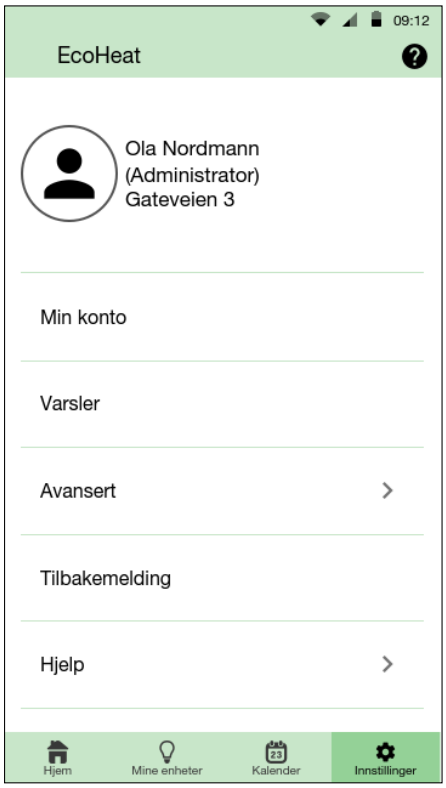
Figur 1: Denne figuren viser Hjem skjermen for mobilapplikasjonen, som også er den første skjermen en innlogget bruker vil se. På denne skjermen kan man velge mellom husholdene man er et medlem av, ved å trykke på knappen der det står 'Valgt bolig: Hjemme'. Det er også mulighet for å endre moduset for den valgte boligen ved å trykke på knappen der det står 'Valgt boligmodus: Hjemme'. I midten av skjermen ser man litt sanntidsstatistikk for husholdet. Dette er en karusell og man kan bla igjennom den for å få mer detaljert informasjon om husholdet.



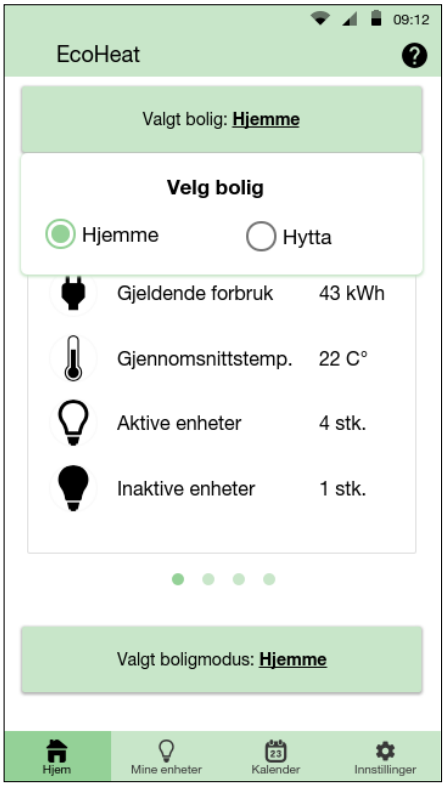
Figur 2: Trykker man på 'Mine enheter' for man opp denne listen over enhetene i det valgte husholdet. Øverst er det mulighet for å legge til en ny enhet og i midten er selve enhetslisten. Fra enhetslisten kan enhetene skrus av eller på manuelt og se nåværende og ønsket temperatur i rommet om enheten kan regulere temperatur. Trykker man på en av enhetene vil man bli navigert til en skjerm som gir mer detaljert informasjon om enheten.



Figur 3: Trykker man på 'Kalender' får man en oversikt over når de forskjellige modusene er aktive, og lar brukeren konfigurere disse tidsrammene selv.

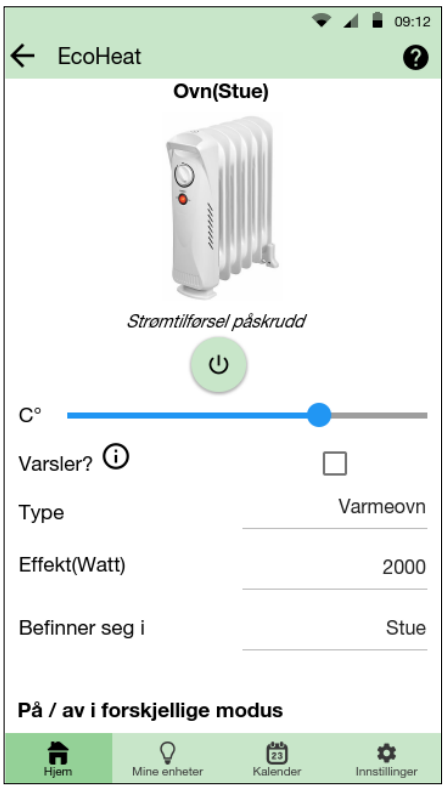


Figur 4: Trykker man på 'Innstillinger' får man opp en rekke menyvalg for å navigere videre til en skjerm for å endre informasjon om enten sin egen konto, endre informasjon om husholdene man administrerer, endre applikasjonsinnstillinger, mulighet for å sende en tilbakemelding og en selvhjelp/kontakt support skjerm.

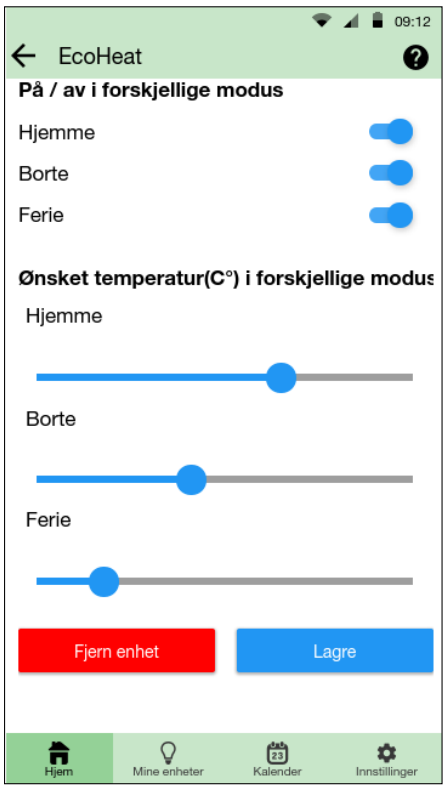


Figur 5: Hjem skjermen hvor brukeren har trykket på knappen for å endre hvilket hushold som er valgt for øyeblikket. En liten popup dukker opp under, med mulighet for å velge mellom alle husholdene en bruker er del av.

Figur 6: I toppen av venstre hjørnet finnes det en knapp for å få tilgang til instrukser og forklaringer for siden brukeren er på for øyeblikket. Eksempelvis her med 'Hjem' sin hjelp side.



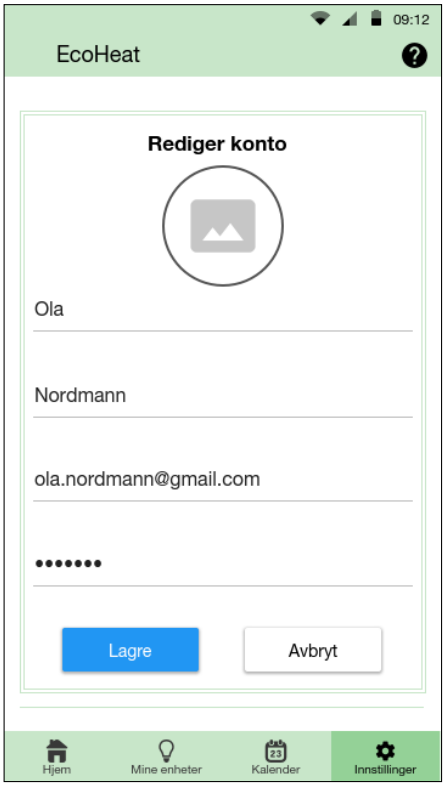
Figur 7: Ved å trykke på en enhet fra 'Mine Enheter', blir man navigert til en skjerm med mulighet for å se og redigere detaljer ved den valgte enheten. Her kan man også manuelt skru av eller på enheten eller sette en ny ønsket temperatur.



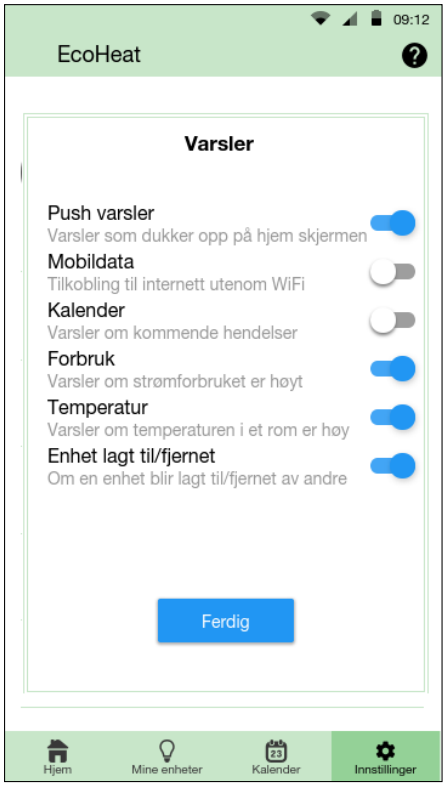
Figur 8: Blar man nedover på skjermen for detaljert informasjon om en enhet, kommer man til flere konfigurasjonsinnstillinger(enda flere om enheten kan regulere temperatur)

Figur 9: Trykker man på 'Registrer ny enhet' fra 'Mine Enheter' blir man ført videre til en skjerm for å registrere en ny enhet. I prototypen var det satt opp slik at det virket som om den ventet på å oppdage enheter på nettverket.

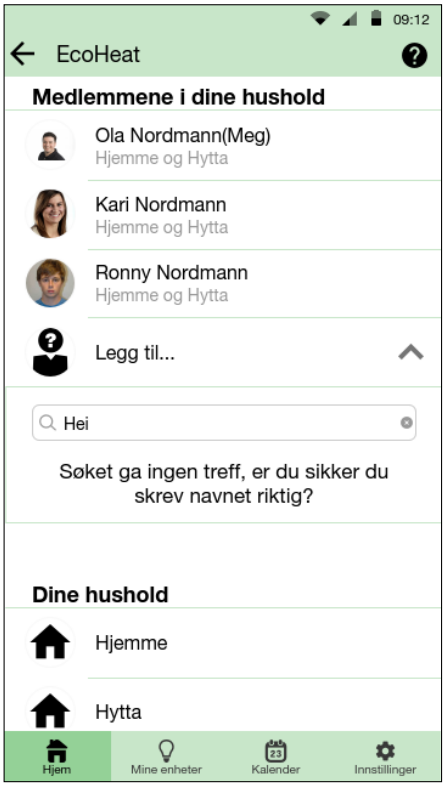
Figur 10: Når 'scan' er fullført får brukeren muligheten til å skrive inn informasjon om enheten.



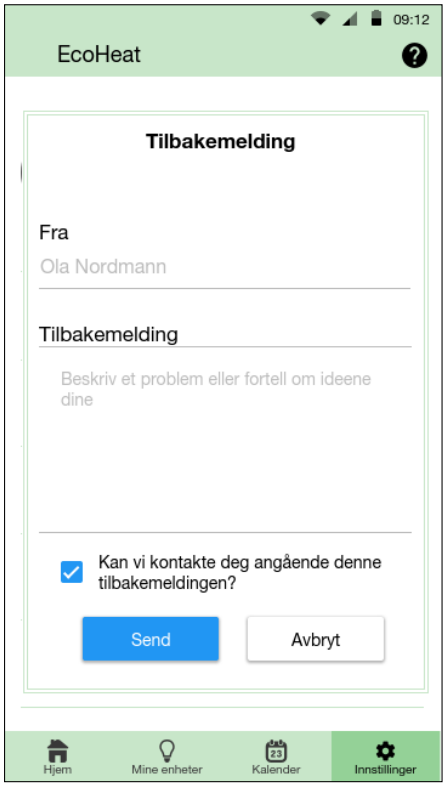
Figur 11: Fra 'Innstillinger' kan man navigere til en skjerm for å redigere kontoinformasjon.



Figur 12: Fra 'Innstillinger' kan man komme til 'Varsler' som lar brukeren konfigurere hvilke varsler som skal være på.



Figur 13: Fra 'Innstillinger' kan man navigere videre til mer 'Avanserte innstillinger' for å redigere husholdene og medlemmene under disse.

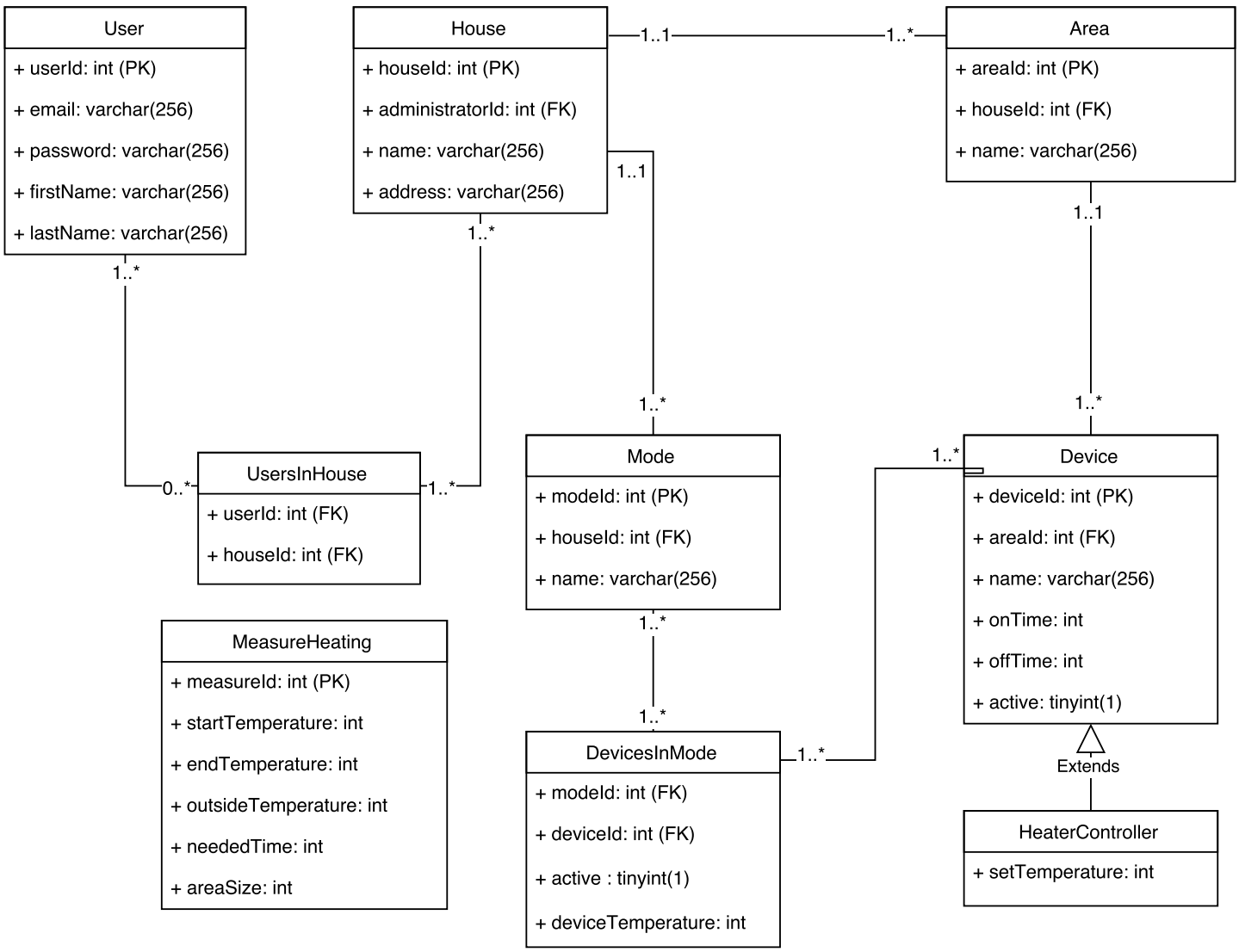


Figur 14: Fra 'Innstillinger' kan man navigere til en skjerm for å skrive en tilbakemelding til systemansvarlige. Dette kan alt fra spørsmål om applikasjonen (funksjonalitet, osv) til innrapportering av bugs.

Figur 15: Fra 'Innstillinger' kan man navigere til denne skjermen for å se en liste over ofte spurte spørsmål, eller kontakte support om listen ikke dekket problemet til brukeren.

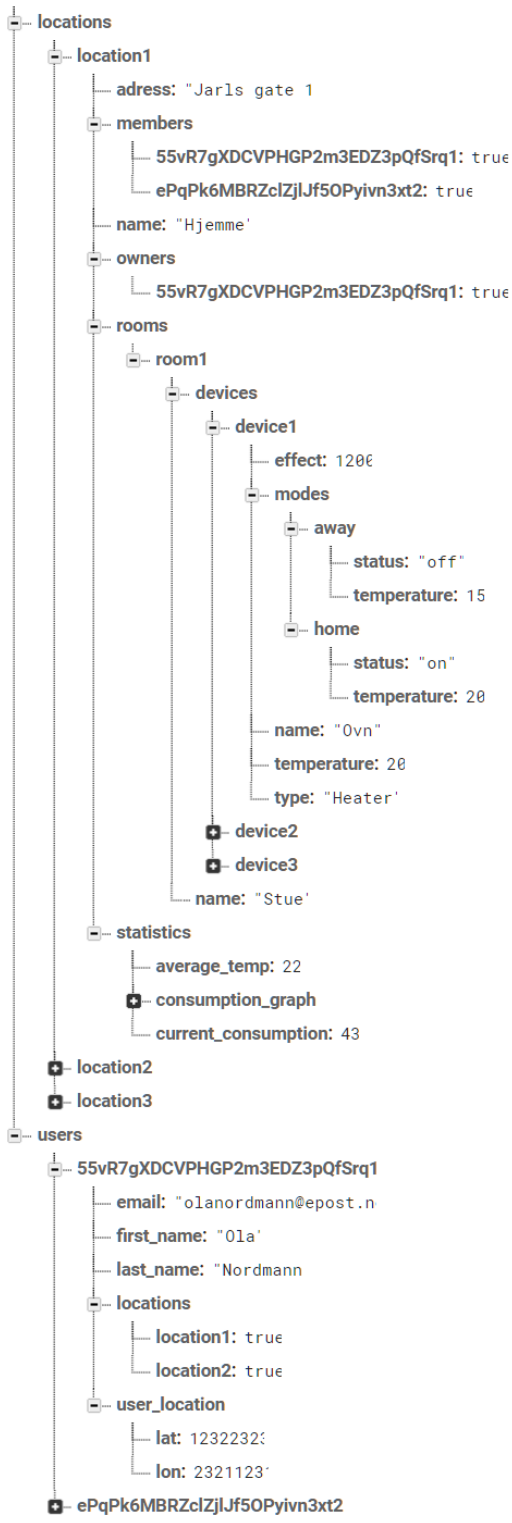
D Opprinnelig databasestruktur

I dette vedlegget finner du den opprinnelige påtenkte databasestrukturen for systemet EcoHeat. Dette ble i ettertid endret til å se ut som databasestrukturen i vedlegg E.


User
+ userId: int (PK)
+ email: varchar(256)
+ password: varchar(256)
+ firstName: varchar(256)
+ lastName: varchar(256)
House
+ houseId: int (PK)
+ administratorId: int (FK)
+ name: varchar(256)
+ address: varchar(256)
Area
+ areaId: int (PK)
+ houseId: int (FK)
+ name: varchar(256)
UsersInHouse
+ userId: int (FK)
+ houseId: int (FK)
Mode
+ modeId: int (PK)
+ houseId: int (FK)
+ name: varchar(256)
Device
+ deviceId: int (PK)
+ areaId: int (FK)
+ name: varchar(256)
+ onTime: int
+ offTime: int
+ active: tinyint(1)
MeasureHeating
+ measureId: int (PK)
+ startTemperature: int
+ endTemperature: int
+ outsideTemperature: int
+ neededTime: int
+ areaSize: int
DevicesInMode
+ modeId: int (FK)
+ deviceId: int (FK)
+ active : tinyint(1)
+ deviceTemperature: int
HeaterController
+ setTemperature: int
1..1
1..*
0..*
Extends

E Databasestruktur

I dette vedlegget finner du slik databasestrukturen endte opp etter at det ble bestemt at Firebase skulle bli brukt. Firebase benytter seg av JSON lagring, ved at all informasjon er lagret i JSON objekter. Her er en figur av hvordan oppsettet ser ut i Firebase, med forklaringer. Denne databasestrukturen ble opprettet i samarbeid med Stian Michael Årsnes.



Figur 1: Databasestruktur i firebase

E.1 Forklaring til figur 1

I figur 1 er et utdrag av slik databasestrukturen i Firebase ser ut. Dataene som ligger i denne figuren er bare eksempel data. Databasen består hovedsakelig av 2 objekter: Locations og Users.

E.1.1 Locations

Locations er objektet i Firebase som inneholder absolutt alle registrerte hushold (Location) i EcoHeat. I den reelle databasen ville husholdene hadde unike identifikatorer, mens i denne figuren benyttes heller 'location1', 'location2', osv. for at det skal være mer lesbart.

Location

Et Location objekt, det vil si et hushold, vil ha en del informasjon knyttet til seg. Under er en liste over hvilken informasjon en Location inneholder.

- En adresse spesifisert av brukeren.
- Et objekt med oversikt over alle medlemmene i husholdet (members).
- Et navn som også er spesifisert av brukeren, og er som vises frem til brukeren i stedet for den unike identifikatoren.
- Et objekt som beskriver eierne i dette husholdet (owners).
- Et objekt som inneholder alle rom objektene (rooms).
- Et statistikk objekt som inneholder litt ferdig beregnet statistikk som brukes til fremvisning av statistikken på forsiden av applikasjonene (statistics).

Oppsummert satt vi det opp slik et en Location holder styr på alle sine medlemmer, eiere og rom. Rommene derretter holder styr på hvilke enheter de har registrert under seg, og enhetene holder styr på hvordan de skal reagere til forskjellige modus.

Members

Members objektet inneholder en rad per 'User', med deres unike identifikator, som tilhører dette husholdet. Dette så vi var nødvendig fordi andre brukere skulle ha muligheten til å se hvilke brukere som tilhørte det samme husholdet som seg selv. I tillegg gir dette oss en enkel måte å kunne referere til de forskjellige brukerne i Users objektet.

Owners

Owners objektet er i stor grad likt som Members objektet, men brukerne som er lagt inn her vil også få rettigheter til å gjøre med administrative oppgaver i løsningene. Dette kan være alt fra å endre informasjon om husholdet i seg selv til å legge til/fjerne enheter. Hver gang en operasjon krever høyrere tillatelser, vil dette objektet sjekkes for å se om brukerens unike identifikator tilhører denne listen.

Rooms

Rooms objektet inneholder alle Room objektene i datastrukturen, som i praksis betyr at den holder listen over alle de forskjellige rommene i et hushold.

Room

Et room objekt inneholder all informasjon om ett enkelt rom. Her har den en liste over alle enhetene i rommet (devices) og navnet på rommet.

Devices

Devices objektet inneholder alle device objektene, som i likhet med rooms, blir en liste over alle registrerte enheter som tilhører dette rommet.

Device

Device objektet inneholder all informasjon om en enkelt enhet. I den ekte versjonen av databasen er disse skilt med en unik identifikator, men for lesbarhetens skyld er de kalt 'device1', 'device2', osv. En enhet har informasjon som effekt, konfigurasjonsobjektet for denne enheten (modes), et navn, sist registrerte temperatur ved enheten (temperature) og hvilken type brukeren har sagt denne enheten er.

Modes

Modes objektet inneholder alle de forskjellige objektene som beskriver forskjellig konfigurasjon av enkelt enheter i de forskjellige modusene. Som et eksempel har vi lagt ved 'away' og 'home' som moduser her, der enheten vil oppføre seg forskjellig når et hushold endres til et av disse modusene. Her er det lagt opp slik at en enhet kan være satt til å være av eller på (status) i modusene og om enheten kan i tillegg regulere temperatur, vil dette også lagres i et modus (temperature). Dette vil bli brukt slik at når brukeren setter husholdet i et visst modus, vil serveren gå igjennom alle enhetene for å finne ut hvordan hver enkelt enhet skal reagere på denne endringen.

Statistics

Statistics objektet inneholder data som skal vises frem til brukeren, slik at de får et oversiktelig bilde over hvordan husholdet opererer for øyeblikket. Det som kommer til å ende opp i dette objektet er ennå ikke helt bestemt.

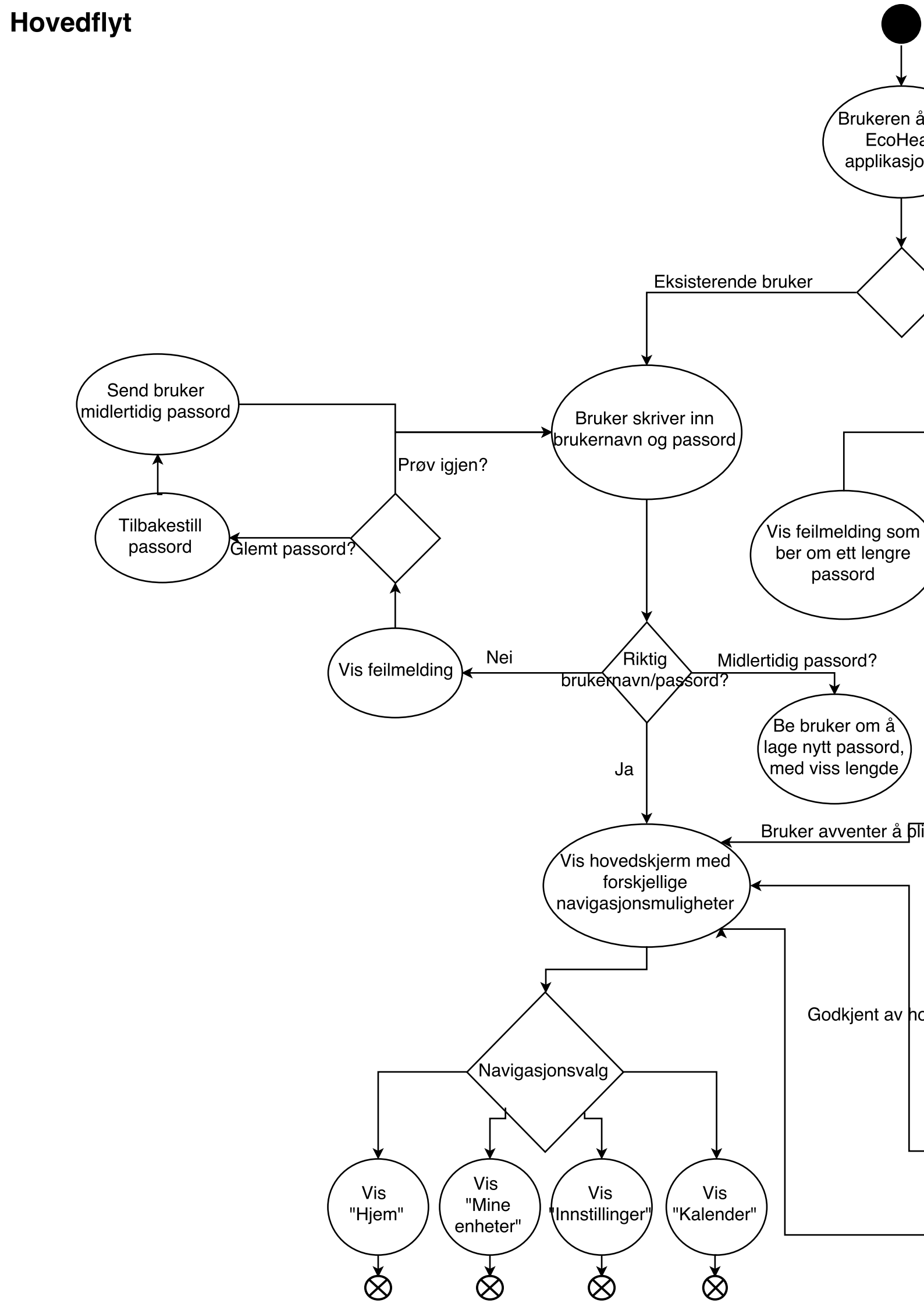
E.1.2 Users

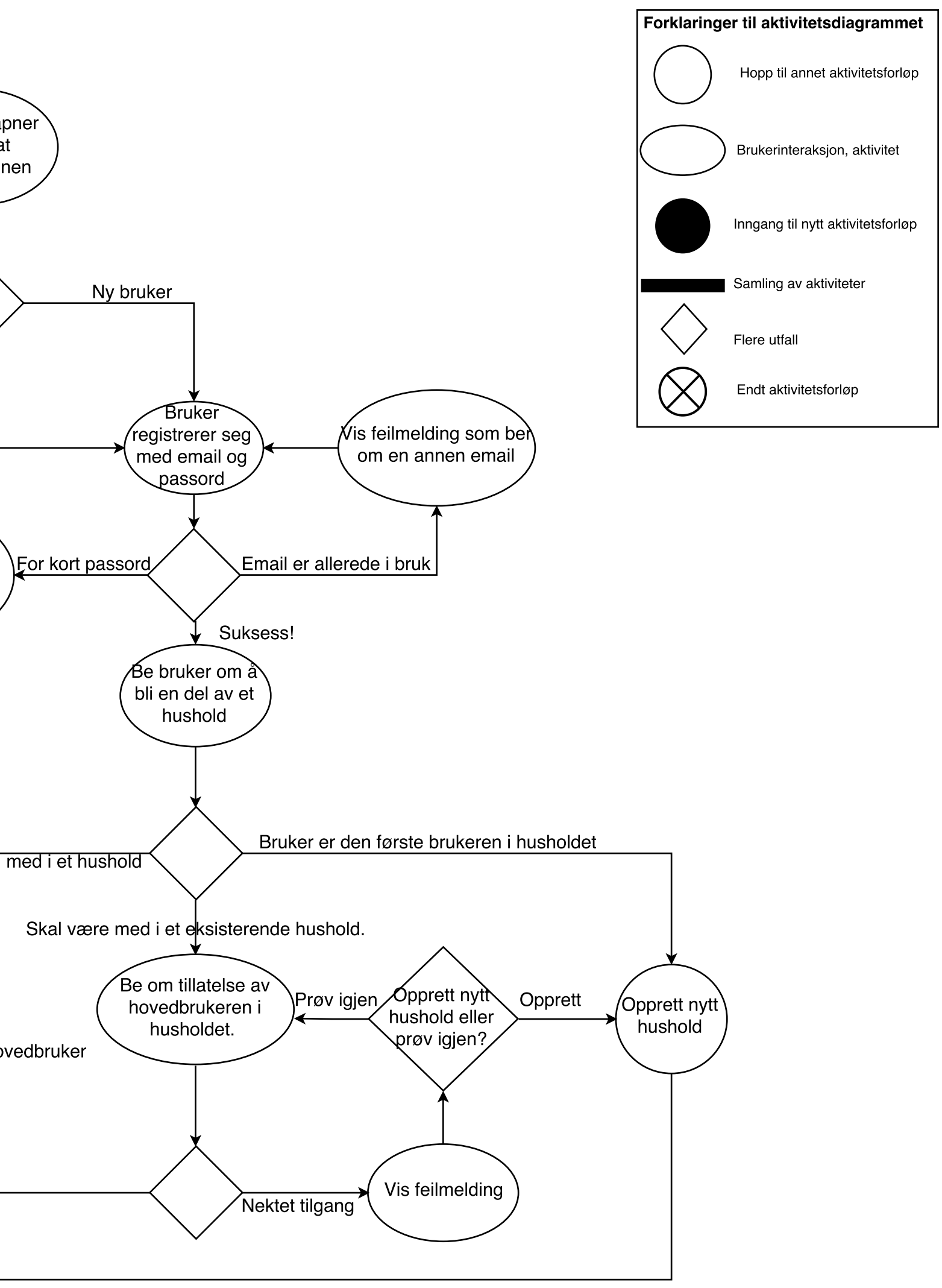
Users objektet inneholder alle brukerne som er registrert i EcoHeat. Her er de adskilt med unike idenfikatorer for hvert enkelt objekt. Disse objektene inneholder informasjon om brukeren som epost, fornavn, etternavn, en liste over alle lokasjonene de er en del av og sist registrerte geolokasjon. Den siste biten med geolokasjon kommer av at oppdragsgiver hadde et ønske om funksjonalitet tilknyttet brukerens lokasjon, men det har ikke blitt implementert noe som bruker lokasjonen i dette prosjektet.

F Aktivitetsdiagram

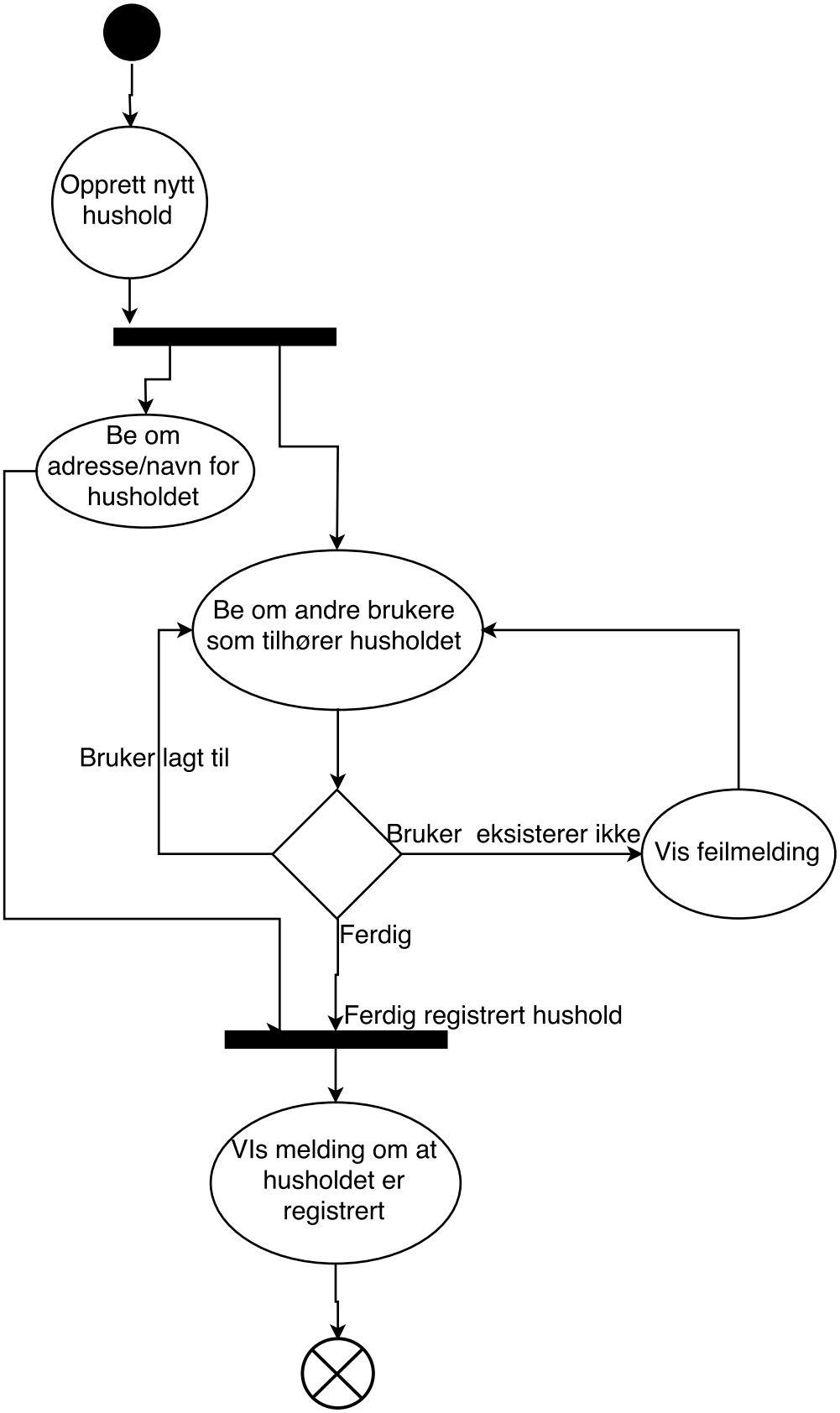
I dette vedlegget finner du aktivitetsdiagrammet som viser hvordan applikasjonene skal reagere når brukerne samhandler med brukergrensesnittet. Dette vedlegget er strukturert slik at stort sett at alle mulige forløp for en gitt skjerm er samlet på en side, med unntak av de to første sidene. De to neste sidene er hovedforløpet i applikasjonene, og er dessverre av den detaljgrad slik at de ikke fikk plass på en side. Side 2 har en forklaring over hva de forskjellige figurene representerer.

Hovedflyt

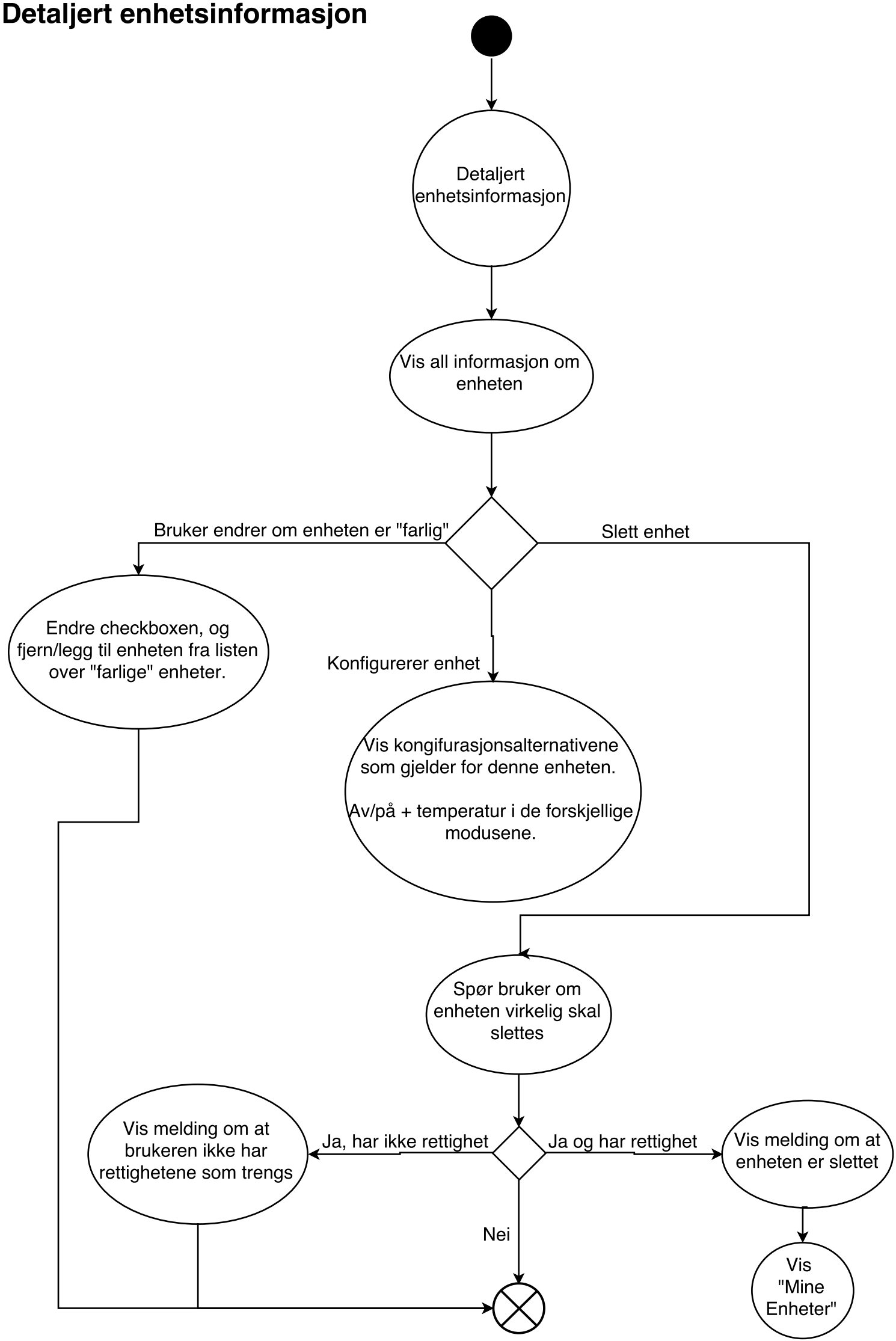

Brukeren å
EcoHea
applikasjo
Eksisterende bruker
Send bruker midlertidig passord
Bruker skriver inn brukernavn og passord
Prøv igjen?
Tilbakestill passord
Glemt passord?
Vis feilmelding som
ber om ett lengre
passord
Vis feilmelding
Nei
Riktig brukernavn/passord?
Midlertidig passord?
Be bruker om å lage nytt passord, med viss lengde
Ja
Bruker avventer å bli
Vis hovedskjerm med forskjellige navigasjonsmuligheter
Godkjent av hc
Navigasjonsvalg
Vis "Hjem"
Vis "Mine enheter"
Vis "Innstillinger"
Vis "Kalender"


Forklaringer til aktivitetsdiagrammet
Hopp til annet aktivitetsforløp
Brukerinteraksjon, aktivitet
Inngang til nytt aktivitetsforløp
Samling av aktiviteter
Flere utfall
Endt aktivitetsforløp
apner
at
nen
Ny bruker
Bruker registrerer seg med email og passord
Vis feilmelding som ber om en annen email
For kort passord
Email er allerede i bruk
Suksess!
Be bruker om å bli en del av et hushold
med i et hushold
Bruker er den første brukeren i husholdet
Skal være med i et eksisterende hushold.
Be om tillatelse av hovedbrukeren i husholdet.
Prøv igjen
Opprett nytt hushold eller prøv igjen?
Opprett
Opprett nytt hushold
ovedbruker
Vis feilmelding
Nektet tilgang

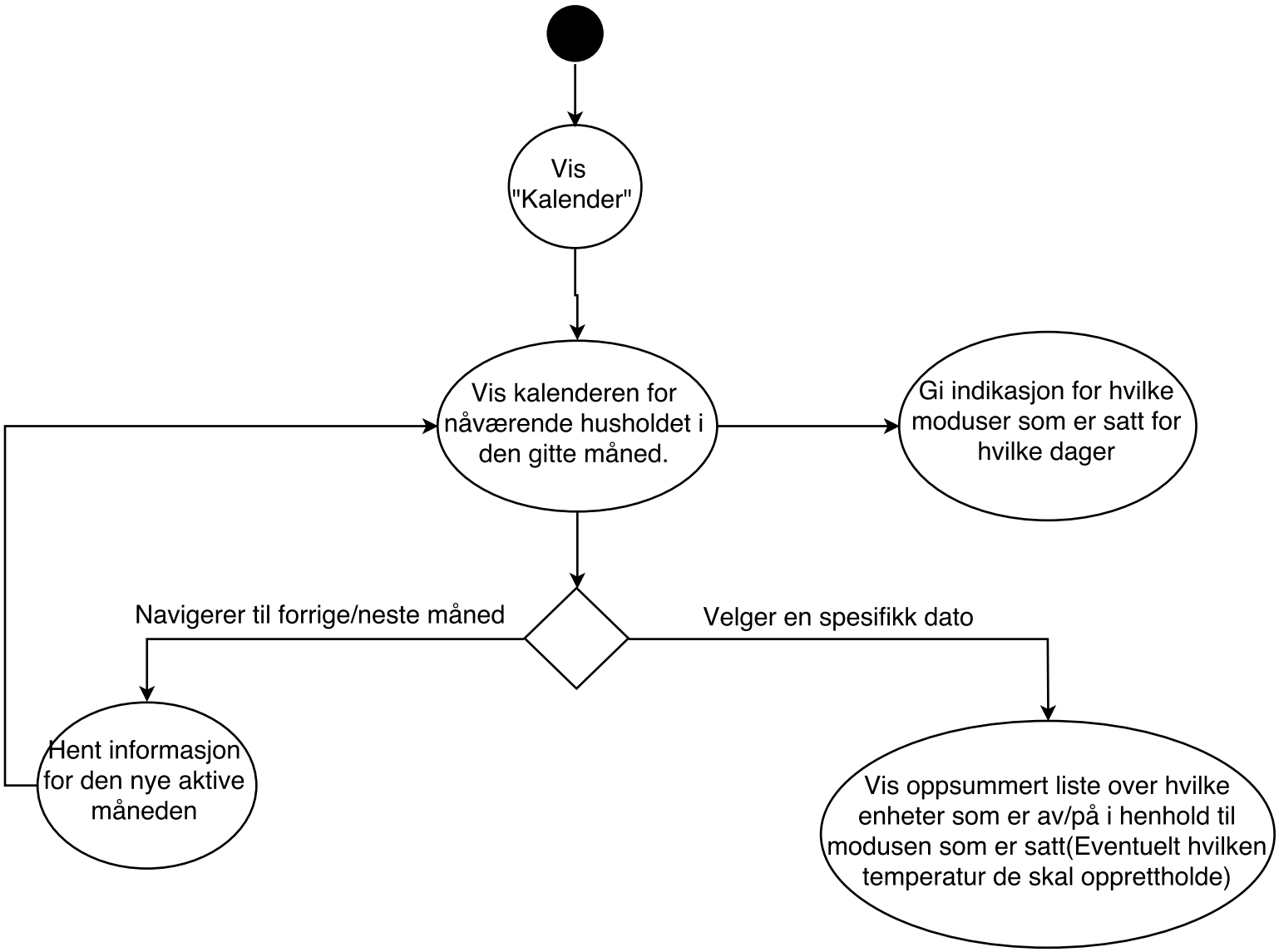
Opprett nytt hushold



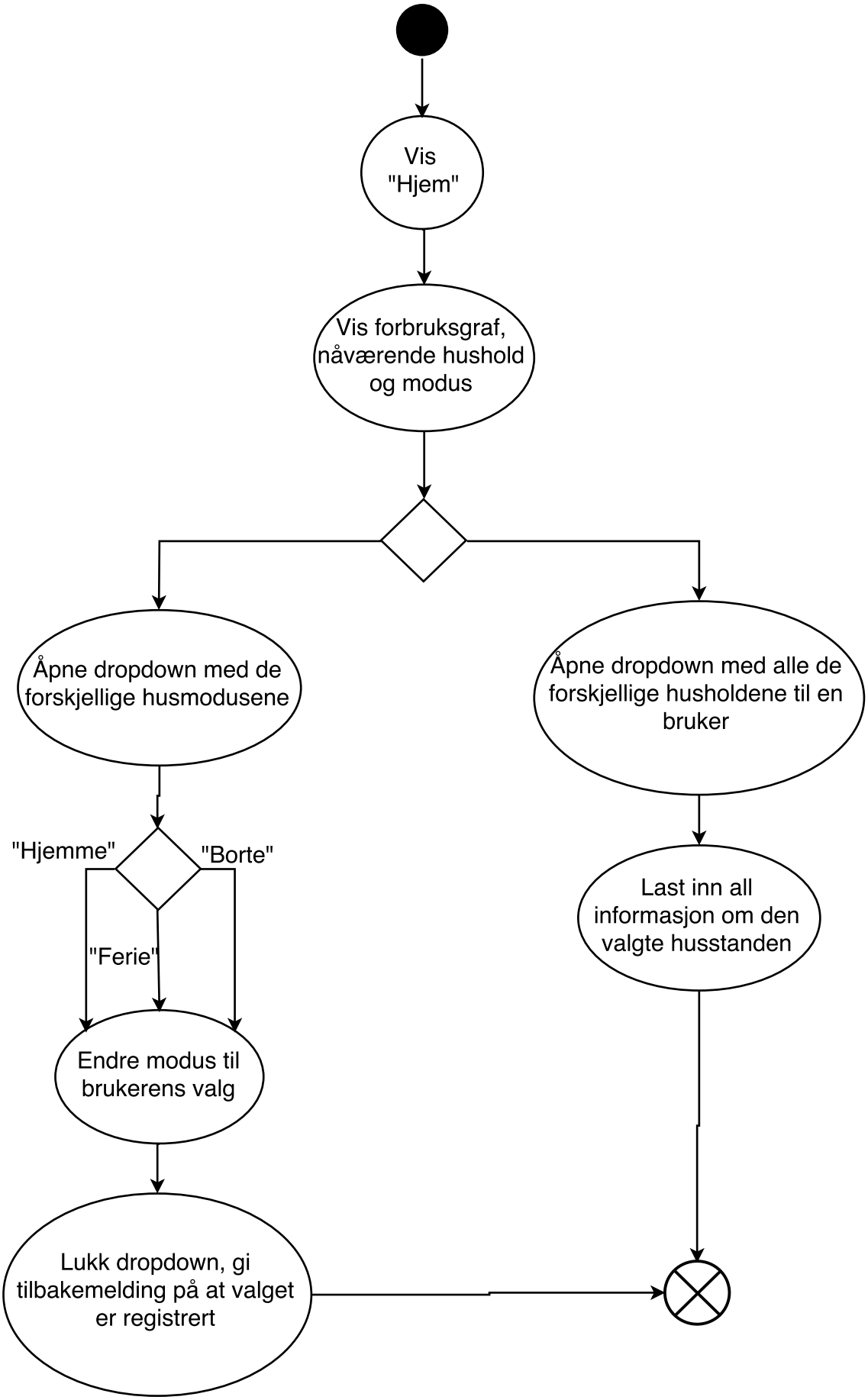
Detaljert enhetsinformasjon



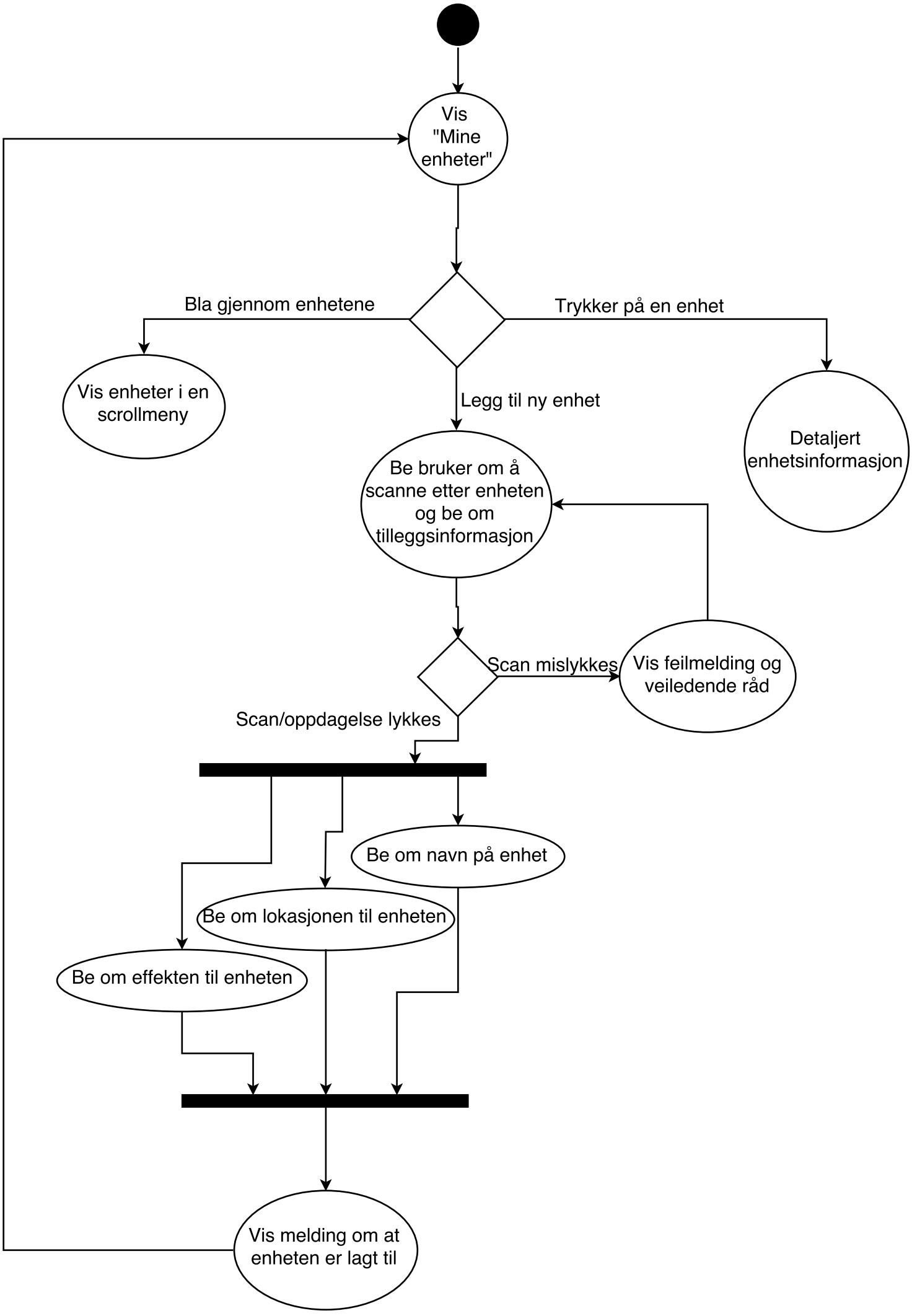
Vis Kalender



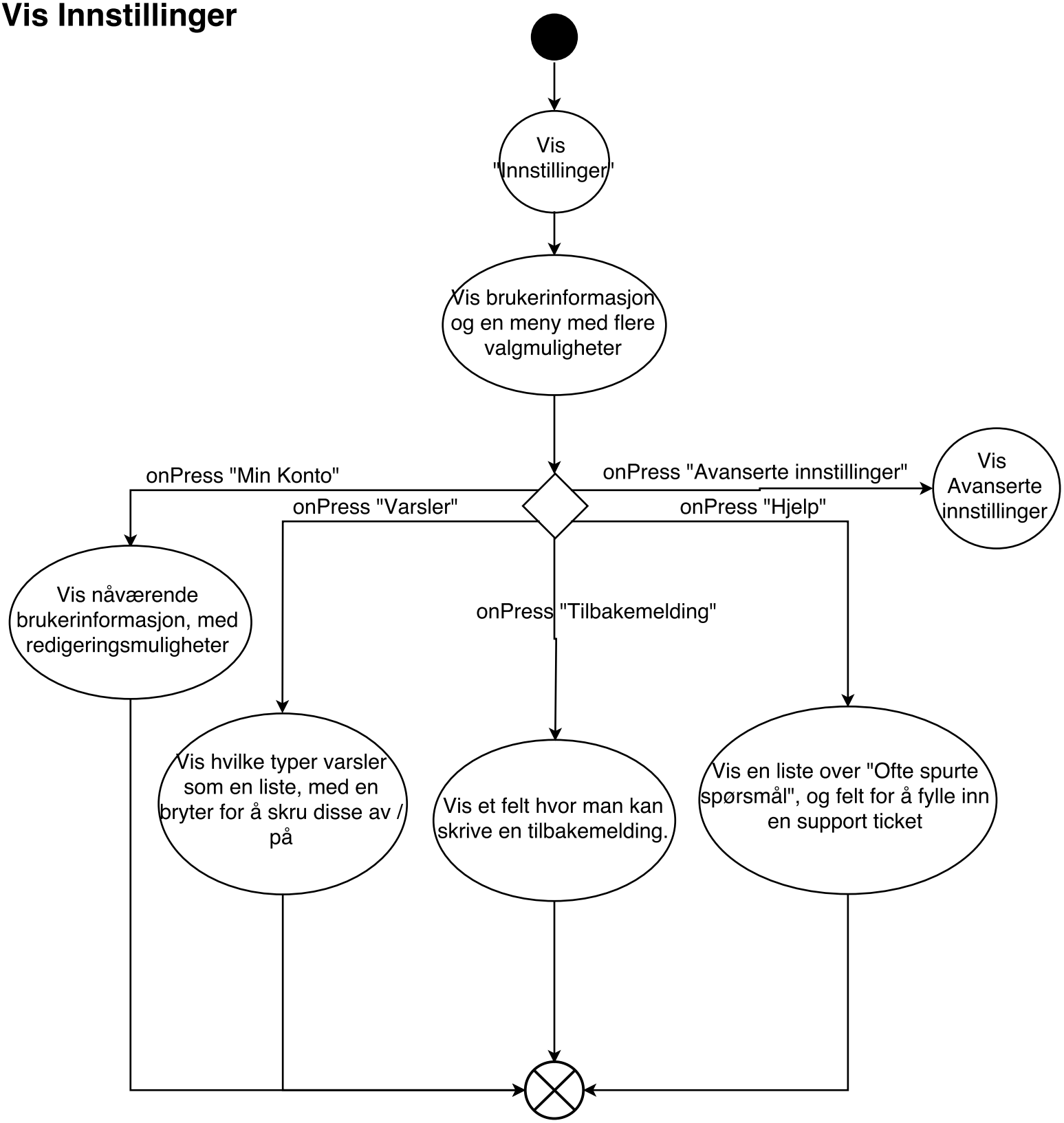
Vis Hjem


Vis
"Hjem"
Vis forbruksgraf, nåværende hushold og modus
Åpne dropdown med de forskjellige husmodusene
Åpne dropdown med alle de forskjellige husholdene til en bruker
"Hjemme"
"Borte"
"Ferie"
Last inn all informasjon om den valgte husstanden
Endre modus til brukerens valg
Lukk dropdown, gi tilbakemelding på at valget er registrert

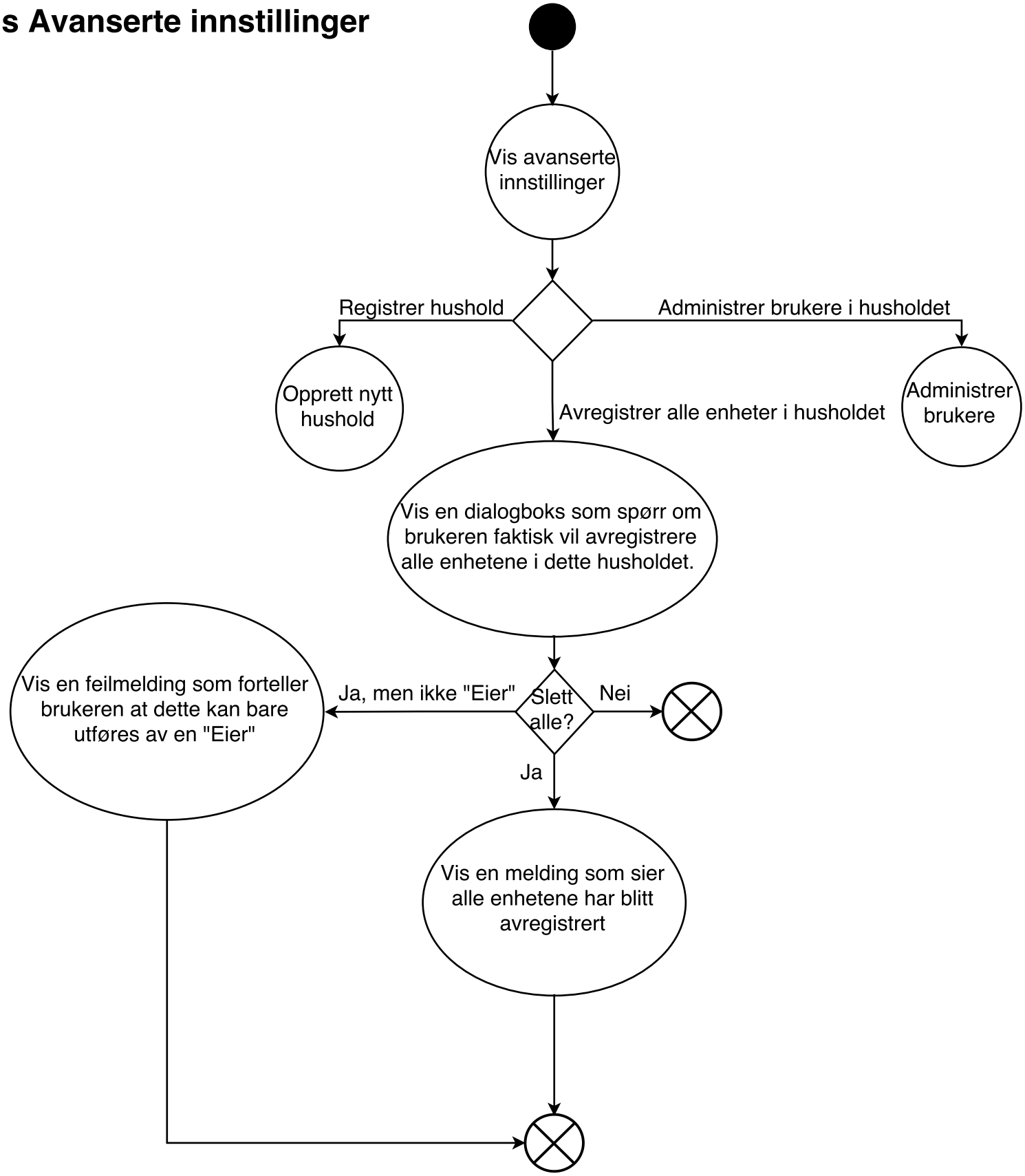
Vis Mine Enheter



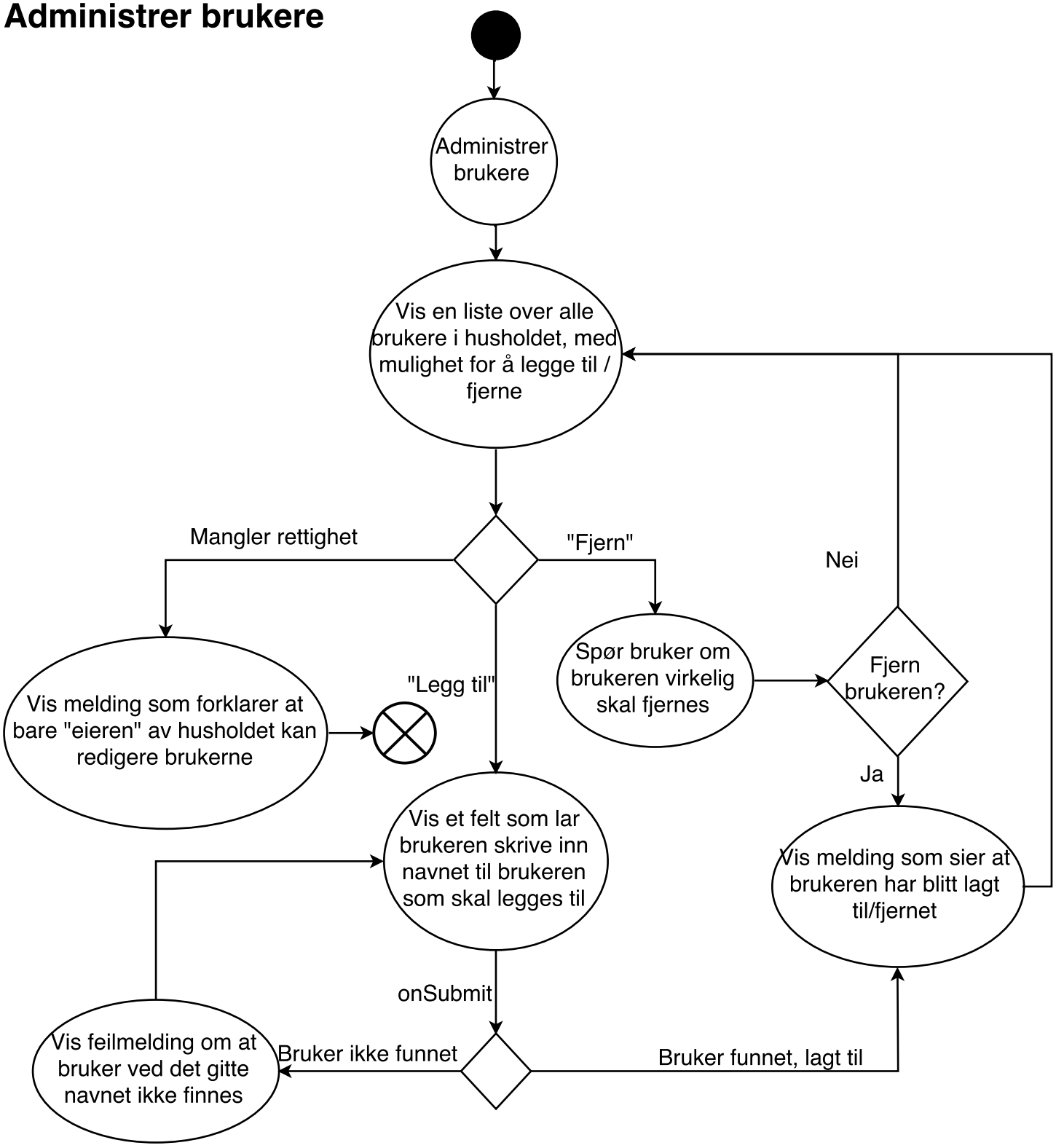
Vis Innstillinger



Vis Avanserte innstillinger



Administrer brukere



G auth.js

I dette vedlegget ligger koden for å sikre deler av applikasjonen fra brukere som ikke har riktig autentifikasjonsstatus. Store deler av applikasjonen skal bare vises hvis brukeren er autentifisert (alle oversiktssider) mens noen andre deler (registrasjon) skal bare vises hvis brukeren ikke er autentifisert.

```
/**
 * Method for protecting components from users. Some components should only be
 * shown when the user is signed in, other should only be shown if the user
 * ISN'T signed in. If the check fails the users is sent back to the index.
 */

import React, { Component, PropTypes } from 'react';
import { connect } from 'react-redux';

export default function (ComposedComponent, shouldBeAuthenticated = true) {
  class Authentication extends Component {
    static contextTypes = {
      router: PropTypes.object,
    };

    static propTypes = {
      authenticated: PropTypes.bool.isRequired,
    };

    /**
     * Doing the whole check before the component is mounted. if the users
     * authentication status doesn't match what was initially expected he's
     * pushed to the index route.
     */
    componentWillMount() {
      const { authenticated } = this.props;

      if ((authenticated !== true && shouldBeAuthenticated === true) ||
          (authenticated === true && shouldBeAuthenticated !== true)) {
        this.context.router.push('/');
      }
    }

    /**
     * We need to do the same check when the component is updated, if we eg.
     * update the application state.
     */
    componentWillUpdate(nextProps) {
      if ((nextProps.authenticated !== true && shouldBeAuthenticated === true) ||
          (nextProps.authenticated === true && shouldBeAuthenticated !== true)) {
        this.context.router.push('/');
      }
    }


```

```
46      render() {
47        return <ComposedComponent {...this.props} />;
48      }
49    }
50
51    function mapStateToProps(state) {
52      return { authenticated: state.auth.authenticated };
53    }
54
55    return connect(mapStateToProps)(Authentication);
56  }
```

H webpack.config.js

I dette vedlegget vises konfigurasjonen som ble benyttet for webpack dev server.

```
module.exports = {
  entry: [
    './src/index.js',
  ],
  output: {
    path: __dirname,
    publicPath: '/',
    filename: 'bundle.js',
  },
  module: {
    loaders: [{
      exclude: /node_modules/,
      loader: 'babel',
      query: {
        presets: ['react', 'es2015', 'stage-1'],
      },
    }],
  },
  resolve: {
    extensions: ['', '.js', '.jsx'],
  },
  devServer: {
    historyApiFallback: true,
    contentBase: './',
  },
};
```

I Node.js moduler

I dette vedlegget ligger en oversikt over de forskjellige Node.js modulene som har blitt brukt i prosjektet.

```
"devDependencies": {
    "babel-cli": "^6.24.0",
    "babel-core": "^6.2.1",
    "babel-loader": "^6.2.0",
    "babel-preset-es2015": "^6.1.18",
    "babel-preset-react": "^6.1.18",
    "chai": "^3.5.0",
    "chai-jquery": "^2.0.0",
    "eslint": "^3.15.0",
    "eslint-config-airbnb": "^14.1.0",
    "eslint-config-rallycoding": "^3.2.0",
    "eslint-plugin-import": "^2.2.0",
    "eslint-plugin-jsx-a11y": "^3.0.2",
    "eslint-plugin-react": "^6.9.0",
    "jquery": "^2.2.1",
    "jsdom": "^8.1.0",
    "mocha": "^2.4.5",
    "react-addons-test-utils": "^0.14.7",
    "webpack-dev-server": "^1.14.0"
},
"dependencies": {
    "babel-preset-stage-1": "^6.1.18",
    "bootstrap-sass": "^3.3.7",
    "bootstrap-select": "^1.12.2",
    "firebase": "^3.7.5",
    "lodash": "^3.10.1",
    "react": "^0.14.3",
    "react-dom": "^0.14.3",
    "react-fontawesome": "^1.5.0",
    "react-intl": "^2.2.3",
    "react-redux": "^4.0.0",
    "react-router": "^2.0.1",
    "react-router-redux": "^4.0.8",
    "react-select-picker": "0.0.7",
    "react-skylight": "^0.4.1",
    "react-sparklines": "^1.6.0",
    "react-tooltip": "^3.2.10",
    "recharts": "^0.21.2",
    "redux": "^3.0.4",
    "redux-form": "5.3.3",
    "redux-thunk": "^2.2.0",
    "webpack": "^1.12.9"
}
```

J Oppgavetekst

Dette vedlegget består av den teksten som introduserte prosjektet på begynnelsen av prosjektperioden, skrevet av Hark Technologies.

Hark Technologies

Oppdragsgiver: Hark Technologies AS
Kontaktperson: Joar Gunnarsjaa Harkestad
Adresse: Smørblomstveien 27, 7050 Trondheim
Telefon: 90533609
Epost: joar.harkestad@gmail.com

EcoHeat

Hark Technologies utvikler den første Smartpluggen med både Wi-Fi og termostat. Vårt produkt EcoHeat gjør varmeovnen din (og air-conditionen i varmere land) trådløs, og automatiserer prosessen med justering av temperatur når du er borte eller sover, slik at du kan spare strøm på en utrulig lett måte. Gjennom den tilhørende appen kan du se ditt strømforbruk i sanntid, og hvor mye du har spart og kommer til å spare med EcoHeat systemet. I tillegg vil du ha et smarthus med alle de fordelene og mulighetene high-end systemer gir til en brøkdel av prisen.

Oppgaven

Utvikle en «EcoHeat» app med online samt Wi-Fi kommunikasjon med Smartpluggen. Målet er å utvikle appen samt et brukergrensesnitt som er så intuitivt og enkelt at det blir en god opplevelse for brukeren, samtidig som det er lett forståelig både for eldre og unge. Interaksjonsdesignet her blir en essensiell del av oppgaven.

Features som foreslås er:

- Visning av ute og innetemperatur
- Visning av strømforbruk samt kostnader knyttet til dette
- Sette temperatur
- Sette timere/kalenderfunksjon
- Definere og koble sammen brukere til samme hus
- Integrering med vekkerklokke
    - Når man setter alarmen om morgningen til kl 07.00 blir det f.eks. også automatisk satt at ovnen i stua slås på kl 06.30 slik at det er varmt når en står opp.
- Integrering med GPS
    - Når man går hjemmefra skrus varmen i huset seg automatisk ned til valgt "borte" temperatur, og når appen merker at du er på veg hjem (v.h.a. GPS lokasjon) skrus varmen opp igjen til valgt hjemme temperatur.
- Integrering med Google Calendar
    - Ser når du er borte slik at temperaturen kan skrus ned i de tidsrommene
- Visning av hvor mye penger som er spart ved å ha smartstyringen på ovnen
- Sammenligning av forbruk med andre brukere/naboer/ditt nabolag
- Oppkobling til nettverksserver for styring av systemet via internett fra hvor som helst
- Mulighet for å bruke Wi-Fi Direct for å koble til systemet

Elles foreslås det også på sikt å lage en oversikt over hvor lang tid det tar å varme opp et rom ved forskjellige ute- og inne-temperaturer slik at ovnen kan slå seg på ved optimal tid. (Her mulig å integrere machine learning om ønskelig.)

Hark Technologies

Funksjonaliteten er ikke begrenset av ovenstående, da det er viktig for fremtiden å kunne se andre muligheter for utvidelser. Systemet vil også på sikt vokse til en større løsning med flere enheter som til sammen vil utgjøre et fullt smarthus. Det er derfor viktig å utvikle programvaren slik at den er enkelt skalerbar.

Programmering

EcoHeat appen bør i tillegg til å være skalerbar utformes slik at den skal kunne kjøres på ulike plattformer og OS for tablet, telefon, samt gjennom en webside. Det foreslås å bruke React Native for utvikling av applikasjonen, språket brukt for å bygge blant annet Facebook, Instagram, Airbnb, samt mange andre.

Studentene vil få mulighet til å være med på å utforme og definere de funksjonene appen skal inneholde, samt mulighet til å teste de opp imot hardware som styres.

Programmeringsoppgaven vil blant annet bestå i:

- Innhenting av informasjon fra hardware
- Utvikling – React Native eller tilsvarende
- Grensesnitt mellom server og app
- Database for lagring av data

Interaksjonsdesign

Mange av dagens smarthus system presenterer alt for mange valg for brukeren, og er dermed forvirrende og vanskelig å bruke. Oppkobling er ofte vanskelig, og det er mye som må settes opp og tas stilling til. Det fokuseres derfor i utviklingen av EcoHeat på at interaksjonsdesignet skal være så enkelt som over hode mulig. Det vil da også være rom for å diskutere hva som virkelig er nødvendig for brukeren å ha kontroll på, og hva som eventuelt kan kjøres i bakgrunnen. Forskjellige brukertester kan potensielt brukes som et verktøy her for å komme fram til optimalt design. Det legges også vekt på at designet skal se stilrent ut for å oppnå en Premium brukeropplevelse.

Oppgaven passer best for en gruppe på 2-3 personer, og kan passe godt både for Programvareutvikling og Webutvikling, eller som et samarbeidsprosjekt mellom de to. Det finnes også utvidelser til prosjektet med tanke på informasjonssikkerhet og lignende dersom dette skulle være ønskelig å se på.

Hark Technologies er en oppstartsbedrift lokalisert i Trondheim kommune som fokuserer på hardware og software utvikling spesielt rettet mot smarthus markedet og Internet of Things. Hark Technologies vil bistå prosjektet i gjennomføringen av oppgaven og ser muligheter for ansettelser på bakgrunn av baceloroppgaven(e), da teamet skal utvides med blant annet flere programmerere og designere i løpet av 2017. Jevnlige møter og oppgavefordelinger avtales underveis. Ta gjerne kontakt ved spørsmål eller bare for å slå av en hyggelig prat.

Hark Technologies

K Prosjektplan

I dette vedlegget finnes prosjektplanen for dette prosjektet. Dette var en del av forprosjektet, for at gruppene skulle sette seg inn i oppgaven, før de begynte med utviklingen og/eller rapportskrivingen.

Innhold

Innhold i
Introduksjon til prosjektplanen iii
1 Mål og Rammer 1
1.1 Bakgrunn 1
1.2 Prosjektmål 1
1.2.1 Resultatmål 1
1.2.2 Effektmål 1
1.3 Rammer 2
2 Omfang 3
2.1 Fagområde 3
2.2 Avgrensing 3
2.3 Oppgavebeskrivelse 4
2.4 Utviklingsverktøy 5
3 Prosjektorganisering 6
3.1 Ansvarsforhold og roller 6
3.2 Rutiner og regler i gruppa 6
3.2.1 Rutiner 6
3.2.2 Regler 7
3.2.3 Konsekvenser 7
4 Planlegging, Oppfølging og Rapportering 8
4.1 Hovedinndeling av prosjektet 8
4.1.1 Innledning 8
4.1.2 Kravspesifikasjon 8
4.1.3 Design og arkitektur 8
4.1.4 Koding, kvalitetskontroll, implementering og testing 9
4.1.5 Avslutning og konklusjon 9
4.2 Systemutviklingsmodell 10
4.3 Plan for statusmøter og beslutningspunkter 10
5 ORGANISERING AV KVALITETSSIKRING 11
5.1 Dokumentasjon, standardbruk og kildekode 11
5.2 Versjonskontroll 11
5.3 Konfigurasjonsstyring 11
5.4 Risikoanalyse 12
5.4.1 Identifisere mulige risikoer 12
5.4.2 Analyse 12

5.4.3 Tiltak 13
5.4.4 Oppfølgning 13
6 PLAN FOR GJENNOMFØRING 14
6.1 Gantt-skjema 14
Bibliografi 16

Introduksjon til prosjektplanen

Forord

Dokumentet presenterer planen for prosjektet. Her vil vi ta for oss hvordan vi har tenkt til å tilnærme oss oppgaven med tanke på:

- Mål og rammer
- Omfang
- Prosjektorganisering
- Planlegging
- Kvalitetssikring
- Gjennomføringsplan

Dette dokumentet vil klargjøre hvilke fremgangsmåter/teknologier vi ønsker å benytte oss av, med argumentasjon for de valgene vi har tatt.

Informasjon om deltagerne

Prosjektgruppen består av tre studenter, der to går programvareutvikling og en går dataingeniør ved NTNU i Gjøvik.

- Henrik Haugom Solum, Programvareutvikling(14HBPUA).
- Jostein Enes, Programvareutvikling(14HBPUA).
- Tarjei Holtskog, Dataingeniør(13HBIDATA).

Veileder: Frode Haug v/NTNU i Gjøvik.
Oppdragsgiver: Hark Technologies v/Joar Gunnarsjaa Harkestad.

1 Mål og Rammer

Dette kapittelet omhandler målene det ønskes å oppnå med dette prosjektet og hvilke rammer som har blitt satt.

1.1 Bakgrunn

Hark Technologies arbeider med å lage den første smartpluggen, EcoHeat, med både Wi-Fi og termostat. EcoHeat er en type smartplugg som kan åpne eller kutte strømtilførselen til forskjellige elektroniske apparater, ved at den kan motta og sende informasjon over internett. Bakgrunnen for prosjektet er at oppdragsgiver opplevde hvor mye strøm man kunne spare ved å slå av varmekilder når man ikke oppholdt seg i hjemmet, men at det senket komforten. Det å manuelt måtte skru ting av og på viste seg å ha den ulempen at det tok tid fra man kom hjem til boligen var varm igjen. Derfor ønsker han seg nå å lage et system som kan automatisere dette slik at man kan få fordelen av redusert forbruk uten å måtte redusere sin egen komfort i hjemmet.
For at dette skal fungere trengs det en applikasjon som kan sørge for at EcoHeat kan kommunisere med omverden, og omvendt. Denne oppgaven skal ta for seg dette ved å utvikle en slik applikasjon for både web og mobil.

1.2 Prosjektmål

Under presenteres de forskjellige målene man ønsker å oppnå med dette prosjektet.

1.2.1 Resultatmål

Resultatmålene er de faktiske resultatene, det vil si hele systemet, som skal leveres til oppdragsgiver ved endt prosjektperiode. Under er det en liste over de resultatene man ønsker å oppnå:

- Den ferdige applikasjonen skal være skalerbar og ha versjoner for både nettleser og mobile operativsystemer (Android og iOS).
- Brukerne skal kunne ha tilgang til systemet sitt fra hvor som helst.
- Applikasjonen skal være intuitiv og enkel å bruke for alle, uavhengig av bakgrunn og alder.

1.2.2 Effektmål

Effektmålene er det arbeidsgiver ønsker å oppnå med det ferdige produktet.

- Senke strømforbruket med opptil 20%* i gjennomsnitt til alle husholdninger som har tatt i bruk EcoHeat.
- Redusere ulykker forårsaket av el-artikler som burde være skrudd av når man forlater hjemmet sitt med 90%.
- Ecoheat gjør smarthusteknologi blir tilgjengelig for alle, til en lavere pris enn løsningene som finnes på markedet i dag (januar 2017).

- EcoHeat er blitt kjent og testet ut av oppdragsgiverens kontakter, som er interessert i lignende produkter.
- I første kvartal etter utslipp av systemet skal mengden brukere være i en størrelsesorden som gjør at kostnadene rundt prosjektet blir dekket (produksjon av smartplugger, serverkostnader m.m.).

**Ifølge Hafslund står oppvarming for 60% av strømforbruket[1] til en gjennomsnittlig husstand. EcoHeat sørger for at oppvarmingen i husstanden skrus ned eller helt av når beboerne er på jobb, eller når de sover. Dette tilsvarer omtrentlig 67% i en gjennomsnittlig hverdag om vi beregner 8 timer til arbeid og 8 timer til søvn. Det vil si at EcoHeat kan skru av oppvarming(og andre el-artikler), når beboerne er på jobb, og dempe temperaturen når de sover. Bare for oppvarming burde EcoHeat kutte forbruket som går til oppvarming med 40-50%, som gjør at det totale strømforbruket blir i teorien redusert med 20-30%. Dette forutsetter da at brukerne ikke bruker mer strøm enn de normalt ville ha gjort uten EcoHeat.*

1.3 Rammer

- Det ble satt en tidsramme fra 10. januar til 16. mai for å ferdigstille prosjektet.
- Det skal skrives statusrapporter gjennom prosjektløpet for å holde kontroll på hvordan gruppen ligger an til sammenliknet med forventningen.
- Om det ender opp med å være forskjeller på plattformene, spesifikt mellom iOS og Android, velger vi å fokusere på Android. Dette er fordi iOS er en såpass lukket plattform i forhold til Android, og ingen av gruppemedlemmene har spesifikke utviklingsverktøy for iOS

2 Omfang

Dette kapittelet skal ta for seg selve oppgaven, hva den består av, hvilke fagområder den ligger innenfor og avgrensninger som har blitt tatt.

2.1 Fagområde

Miljø er et tema som berører oss alle i stor eller liten grad i hverdagen. Man får stadig høre at vi går en skummel fremtid i møte om vi ikke endrer vår livsstil, men man føler seg ofte for liten til å kunne utgjøre en forskjell og det er ofte dyrere å gå for de løsningene som er best for miljøet rundt oss. For at folk skal begynne å tenke miljøbevisst trengs det løsninger som både er intuitive og økonomisk forsvarlige. Avfallssortering er en løsning som har begynt å lede massene inn på riktig spor, men det er fortsatt mye potensiale for at enhver person kan bidra til at våre økologiske fotavtrykk minskes. Både nå og i fremtiden.
Her har teknologi og internett mye å bidra med. Teknologien utvikler seg i en rasende fart og ting som smarttelefoner blir en del av stadig flere sin hverdag. Alt dette er noe man f.eks. kan benytte i smarthus, et tema som denne oppgaven går innunder.

Smarthus

Smarthus er et samlebegrep for automatiserte og intelligente boliger som bidrar til styring av lys, varme, overvåkning, sikkerhet og annen velferdsteknologi. Målet med smarthus er hovedsakelig å senke energiforbruk og kostnader rundt dette, men det handler også om at boligen skal ha en viss intelligens. Dermed skal den f.eks. kunne beskytte beboerne ved å si ifra om en komfyr fortsatt står på eller at man har glemt å låse døra når man forlater huset. I tillegg skal det også være enklere for brukeren å styre enkeltdeler av husets funksjoner, f.eks. sette på kaffetrakteren mens du ligger i sengen.
Vår oppdragsgiver, Hark Technologies, er et selskap som blant annet er med på å skape løsninger for at smarthusteknologi skal bli tilgjengelig for ethvert hjem uten store investeringer. Dette gjøres ved å produsere en smartplugg. Med smartpluggen skal man få fordelene som eksisterer med smarthus, uten å måtte investere store summer og gjøre større endringer av boligen. De eneste kravene som stilles til den som ønsker denne teknologien er å ha en datamaskin eller smarttelefon og tilgang til internett.

2.2 Avgrensing

Smartpluggen er en enhet som skal gjøre dumteknologi smart, på nåværende tidspunktet gjelder dette varmeovner/klimaanlegg. I stedet for at du plugger ovnen direkte i støpselet setter du smartpluggen i stikkontakten og plugger ovnen i pluggens støpsel. Deretter kan du kommunisere med ovnen (f.eks. fjernstyre den) gjennom pluggen, enten i en nettleser eller med mobil via en applikasjon.

2.3 Oppgavebeskrivelse

Oppgaven går ut på å utvikle et visuelt styringssystem som gir oversikt over smartpluggen(e) man har i boligen. Med dette systemet skal du f.eks. kunne styre temperaturen i boligen f.eks. fra senga, på jobb eller på ferie. Styringssystemet skal lanseres som mobilplattform for Android og IOs samt som en nettleserløsning. En stor del av oppgaven er å gjøre brukergrensesnittet så intuitivt og brukervennlig at folk i alle aldre skal kunne bruke det. I tillegg ønskes det også å benytte en modulbasert oppbygging slik at det skal være enkelt å kunne videreutvikle systemet.
Selve oppgaven er åpen hvor gruppen har mulighet til å komme med innspill om hvilke funksjoner styringssystemet bør ha, men noen som er blitt foreslått av arbeidsgiver er:

- Visning av både ute- og innetemperatur.
- Visning av strømforbruk og kostnader knyttet til dette, samt vise hvor mye du har spart på systemet.
- Bruker skal kunne sette en ønsket innetemperatur som systemet automatisk justerer seg etter.
- Man skal kunne registrere flere brukere i samme system slik at flere får tilgang til styringsystemet i en bolig.
- Sette timere og/eller ha en kalenderfunksjon for å sette temperaturen på bestemte dager.
- Mulighet for å sette forskjellige moduser. F.eks. feriemodus hvor alt skrus ned til et absolutt minimum.
- Integrering av vekkerklokke. Dette er noe som primært vil være for mobilappen. Her vil du f.eks. kunne bestemme at hvis alarmen er satt til å av klokka syv vil systemet begynne å varme opp boligen en halvtime før.
- GPS-lokasjon for mobil. Med dette kan mobilapplikasjonen automatisk sette på varmen når den merker at du nærmer deg hjemmet ditt.
- Integrering med Google Calendar slik at systemet automatisk kan se når du er borte og justere varmen deretter.
- Systemet skal kunne koble sammen påskrudde enheter som lamper og kaffetrakter med at alle registrerte brukere har forlatt boligen. Kan på den måten forhindre ulykker med elektriske apparater ved at bruker får melding om at ting potensielt er forlatt påskrudd.
- Implementere algoritmer som automatisk lærer systemet å vurdere tiden som trengs for å varme opp en bolig gitt ute- og innetemperatur. F.eks. ved å basere disse på tidligere målinger lagret i databasen.

I tillegg til dette skal det også utvikles en serverløsning for at brukerne skal kunne kommunisere med smartpluggene sine og en databaseløsning. Databasen skal inneholde brukerinformasjon, informasjon om smartplugger i bruk og målingsdata som pluggene sender til serveren.

2.4 Utviklingsverktøy

Det vil benyttes flere verktøy for dette prosjektet, noe felles og noe spesifikt for nettleserløsningen og mobilløsningen.

Felles

For begge løsningene trengs en teksteditor. Her har valget falt på Atom, en "hackablett-eksteditor. Det som legges i denne beskrivelsen er at det er mulig å legge til pakker laget av både utviklerne og andre brukere. Disse gir deg utvidet funksjonalitet og på den måten kan du tilpasse teksteditoren slik du ønsker.
I tillegg er Node.js en viktig del av utviklingen. Deres pakkesystem npmbenyttes flittig da bibliotekene til React og React-Native eksisterer som pakker i dette systemet.

Mobilapplikasjon

Her trengs det å ha en Android-emulator for å kunne kjøre og debugge applikasjonen gjennom utviklingen. Her falt valget på Android Studio da dette utviklingsmiljøet har innebygget mulighet for å lage egne emulatorer med forskjellige versjoner av Android. I tillegg kan det også være at man må gjøre deler av utviklingen med Java og da er Android Studio et program med full støtte for programmering og debugging av Java.

Nettleserløsning

For utviklingen av nettløsningen er det eneste som trengs tilgang til en nettleser med støtte for JavaScript (f.eks. Google Chrome, Firefox eller Microsoft Edge).

3 Prosjektorganisering

Dette kapitelet skal ta for seg hvordan gruppen har tatt seg av organiseringen. Hva er ansvarsforholdet, hvilke roller har de forskjellige involverte i prosjektet, rutiner som skal benyttes og regler gruppemedlemmene har blitt enige om å følge gjennom prosjektperioden.

3.1 Ansvarsforhold og roller

Prosjektleder: Jostein Enes. Oppgaven til en prosjektleder er å fordele arbeid, passe på at frister blir holdt og at de alltid er minimum to personer på avtalte møter med veileder eller arbeidsgiver. Samtidig har han i oppgave at prosjektet blir gjennomført utifra den utviklingsmodellen som gruppa blir enig om på slutten av prosjektplanen.
Utviklere: Jostein Enes, Tarjei Holtskog og Henrik Solum. Oppgaven til utviklerne er å planlegge og utvikle applikasjonen som arbeidsgiver ønsker seg. Dette er i form av design og funksjonalitet.
Veileder: Frode Haug. Veilederen har i oppgave å komme med råd rundt prossesen om å skrive en bachelor rapport, slik produktet blir best mulig.
Arbeidsgiver: Joar Gunnarsjaa Harkestad. Arbeidsgiver er den som vil ha programvaren som blir lagd. Han sitter på informasjonen om hvordan han mener produktet skal virke til slutt og det er opp til oss, gjennom diskusjon og avklaringer, at produktet som blir laget er det arbeidsgiver ønsket.

3.2 Rutiner og regler i gruppa

3.2.1 Rutiner

- Referater, kvitteringer og kontrakter lagres i felles mappe for bachelorgruppen.
- Under utvikling skal endringer commites regelmessig selv om endringene ikke går igjennom kompilatoren. Når endringene går igjennom kompilatoren og er i en størrelsesorden som tilfører ny eller endret funksjonalitet til programvaren, skal dette pushes opp til felles repo.
- Commit meldinger skal være skrevet på engelsk og skal beskrive arbeidet som har blitt gjort, slik at det er enkelt å forstå hva som har blitt gjort. Om commiten var spesifikk for å låse ett kjent problem, må dette også noteres i commit meldingen.
- Tester skal skrives til moduler som har større deler logikk i seg.
- Ved møter med personer utenfor gruppen, skal det skrives referater som lastes opp til felles mappe.
- Ved innkjøp av maskin- eller programvare skal det tas vare på kvitteringer som lastes opp til felles mappe. Det skal også sendes en kopi av disse kvitteringene til oppdragsgiver.

3.2.2 Regler

- Det forventes 30 timers arbeidsuker i henhold til standard tidsbruk for et fag verdt 20 studiepoeng[2]. Om et gruppemedlem feiler ved gjentatte tilfeller å møte den forventede arbeidsmengden, vil antallet timer som mangler i gjennomsnitt føres til de førstkommende ukene for å bøte på skaden.
- Alle medlemmene i gruppen plikter seg til å gjennomføre de kursene alle på gruppen er enige om at skal gjennomføre. Disse kursene må være fullført innen de tidsrammer som settes før begynnelsen på et kurs. Om kursene ikke er fullført innen avtalt tid, må gruppemedlemmene arbeide med kurset utenfor de timene som er avsatt til å arbeide med bacheloroppgaven.
- Alle medlemmer plikter å møte til avtalte tidspunkter og møter. Om man av en eller annen grunn ikke har mulighet til å møte, må dette meldes fra om på forhånd.
- Alle medlemmer plikter til å føre de timene man bruker til å jobbe med bacheloroppgaven, med en kort beskrivelse av hva som ble gjort, i et eget Google Sheets dokument[3]. Dette er for å passe på at alle gruppemedlemmene yter omtrentlig samme innsats i prosjektet, men også for å kunne vurdere styrkene til hvert enkelt gruppemedlem og dermed delegere oppgaver basert på dette.
- Alle medlemmer plikter seg til å fullføre de oppgaver man har fått tildelt eller frivillig har tatt på seg. Om man oppdager at oppgaven ikke kan bli løst innenfor tidsfristen, må medlemmet selv sørge for å opplyse resten av gruppen om dette slik at flere gruppemedlemmer kan samarbeide om oppgaven eller at oppgaven tas med videre til neste iterasjon (hvis man bestemmer å benytte iterasjoner i utviklingen).
- Gruppen plikter tidlig å informere dersom de ikke er fornøyd med innsatsen til enkelte gruppemedlemmer. Slike beskjeder skal være diskré(bak lukkede dører), og gruppemedlemmene plikter seg til å holde slikt innenfor gruppen, eventuelt med veileder og eller oppdragsgiver.
- Ved faglige uenigheter skal gruppen forsøke å komme til enighet, evt. fattes en flertallsavgjørelse som det kreves lojalitet til. Om gruppen ikke kommer til enighet, vil saken bringes videre til veileder og eventuelt oppdragsgiver.
- Ingen eksludering de siste 20 dagene før innleveringsfrist i henhold til prosjektavtalen.

3.2.3 Konsekvenser

Ved brudd på gruppereglene vil de bli kalt inn til et gruppemøte mellom alle medlemmene av gruppa. Her vil de bli diskutert alvorlighetsgraden på bruddet og måter å løse problemet.

- Hvis gruppemøtet ikke løser problemet vil det bli sendt en skriftlig advarsel til vedkommende med hvilket regelbrudd som gjelder, konkrete frister og arbeidsinnsats som forventes, samt hva som skal til for å bøte på skaden, samt konsekvensene som vil hende om dette ikke blir ordnet opp i (samtale m/veileder, ekskludering o.l).
- Hvis den skriftlige advarselen ikke fungerer, vil de bli et siste møte mellom alle medlemene og veileder. Ved neste brudd på reglene vil dette føre til skriftlig ekskludering fra gruppa via dekanen.

4 Planlegging, Oppfølging og Rapportering

Dette kapitelet omhandler hvordan prosjektet skal deles opp i gjennomførbare deler, slik at det blir lettere å jobbe med prosjektet. Det skal samtidig velges systemutviklingsmodell og begrunne hvorfor denne modellen ble valgte over andre. Til slutt skal er det en plan for statusmøter med veileder og arbeidsgiver, samt tidspunkter for når deler av prosjektet skal være ferdig.

4.1 Hovedinndeling av prosjektet

4.1.1 Innledning

Selve prosjektrapporten blir skrevet på måten vist under. Innledningen vil ta for seg målene med prosjektet og rammene man skal holde seg innenfor, hva som skal gjøres, hvordan man fordeler oppgaver, planlegger og gjennomføre av disse oppgavene, hvordan man skal risikovurdere oppgaven og til slutt gjennomføringsplanen for selve prosjektet.

- Mål og rammer
- Omfang
- Prosjektorganisering
- Planlegging
- Risikovurdering
- Gjennomføringsplan

4.1.2 Kravspesifikasjon

Kravspesifikasjonen vil være kravene som må være oppfylt for at prosjektet skal være ferdig. De som går innenfor dette er sikkerhet, antall bugs og funksjonalitet. For å få denne kravspesifikasjonen må oppgaveteksten konkretiseres gjennom diskusjoner med arbeidsgiver.

4.1.3 Design og arkitektur

Design og arkitektur vil ta opp hva som skal være med i prosjektet og hvordan dette skal implementeres. Måten dette skal beskrives er ved hjelp av underkategoriene nedenfor:

- personas:
- scenarios:
- use case:
- use case diagram
- risikoanalyse
- low-fi prototypes
- high-fi prototypes
- system arkitektur
- UML klassediagram / konseptuell modell
- UML server oppbygging

4.1.4 Koding, kvalitetskontroll, implementering og testing

Denne delen vil inneholde en rapport om koden som blir lagd, samt beskrive kvalitetskontrollen, implementeringen og testingen som blir gjort med produktet.

- Implementering av server og database
- Produkttesting (slutten på hver sprint)
- Problemer og løsninger ved implementasjonene

4.1.5 Avslutning og konklusjon

Til slutt vil man komme med tanker omkring gjennomføringen av bachleoren, vurdering av egen innsats og en konklusjon.

- Diskusjon av oppgaven, gruppen o.l.
- Konklusjon

4.2 Systemutviklingsmodell

En systemutviklingsmodell skal hjelpe utviklere med å finne metoder for å gjennomføre prosjekter på en god og effektiv måte. Dette gjør de på forskjellige måter ut ifra hvilken modell som blir brukt. Vanligvis skiller man modellene i to hovedgruppene smidig og lineær utvikling. De lineære fokuserer på å planlegge alt før man begynner og kan gjøre det vanskelig å endre utviklingen underveis. Smidige modeller planlegger derimot litt om gangen slik at produktet kan forandre seg hvis f.eks. oppdragsgiver kommer på ny funksjonalitet han eller hun ønsker. De smidige gjør også at man raskere kan gi en prototype til arbeidsgiveren slik at man kan få tilbakemeldinger ut ifra han samt brukerne mener.
Under diskusjonen om hvordan vi skulle utvikle prosjektet ble vi sittende igjen med to utviklingsmodeller. Scrum og Inkrementell sekvensielt. Grunnen til vi kom frem til disse to er at prosjektet skal fokusere på å lage funksjonalitet litt etter litt. Dette gjør at vi kan dele oppgaven opp i flere moduler, noe som igjen gjør at de blir lettere å strukturere og planlegge oppgaven med arbeidsgiver.
Etter noe diskusjon bestemte vi oss for at vi skulle bruke Scrum. Grunnen til at den ble valgt er at Scrum har innebygd planmøter som kan varier mellom 1 og 2 uker noe som passer bra for et prosjekt på denne størrelsen. Dette er ikke noe den inkrementell sekvensielle utviklingsmodellen har.

4.3 Plan for statusmøter og beslutningspunkter

Det er planlagt at møter med veileder skal være ukentlige til å begynne med, men dette kan endres til annenhver uke når utviklingen av produktet er i gang. Dette fordi det kan være at det trengs mer tid for å fullføre oppgavene som har blitt satt enn en uke.
Med arbeidsgiver vil det i begynnelsen (januar) være møter annenhver uke, men dette vil endre seg til hver uke når man begynner å designe og ferdigstille kravspesifikasjonen. Som det fremgår i Gantt-skjemaet vil mandagene benyttes for møter. Hvis man er i starten av en sprint (på enten en eller to uker) vil det holdes et planleggingsmøte hvor man går igjennom hva som skal gjennomføres i den kommende sprinten. Hvis man er midt i en sprint på to uker vil dette møtet være et gjennomgangsmøte. Her vil hvert enkelt medlem orientere resten av gruppen om hvor langt man er kommet, hva som gjenstår og hvor mye tid man forventer at det gjenstående arbeidet vil kreve.

5 ORGANISERING AV KVALITETSSIKRING

Utviklingen av EcoHeat antas å bli et av de større prosjektene medlemmene i denne prosjektgruppen har arbeidet med. Oppdragsgiver har stilt noen krav til applikasjonen, men oppfordrer til at gruppen skal komme med forslag til funksjonalitet for den endelige applikasjonen. Det vil si at applikasjonen vil utvides etterhvert som nye tillegg kommer underveis i prosjektperioden, som igjen vil si at det er en del kjente og ukjente risikoer underveis da det endelige målet ikke er klart. Dette kapitlet omhandler risikoer og tanker rundt håndteringen av disse.

5.1 Dokumentasjon, standardbruk og kildekode

En god start for å minimere risikoene ved et utviklingsprosjekt er god dokumentasjon og følge standarder for skrivemåter og kildekode så langt det lar seg gjøre. Dokumentasjon for dette prosjektet vil i hovedsak eksistere i den overordnede bachelor rapporten, wiki seksjonen i bitbucket repositoriet og i denne prosjektplanen. Bachelor rapporten vil inneholde alt av informasjon rundt utviklingen av applikasjonen. Dette kan være alt fra hva vi har tenkt å inkludere i appen, hvilke teknologier vi har tenkt til å bruke, begrunnelser for alle de valgene vi ender opp med å måtte ta, osv. Wikien i bitbucket repositoriet vil ha forklaringer og begrunnelser omkring kildekoden til applikasjonen. Her vil det beskrives hvordan ting henger sammen, med klassediagrammer, beskrivelser med mer.
Som en kvalitetssikring og for å spare utviklerne for elementære feil, vil ESLint[4] bli benyttet til statisk kodetesting. Slik får man hyppige tilbakemeldinger mens produktet utvikles, og man slipper småfeil som skrivefeil i variabelnavn og lignende. Eslint tillater også at man kan definere sine egne regler, dermed kan man tilpasse og bygge på regelsettene slik at Eslint skal støtte oppunder prosjektet på best mulig måte.

5.2 Versjonskontroll

Man har valgt å bruke git og bitbucket til å håndtere versjonskontroll for prosjektet. Dette på bakgrunn av gode erfaringer med systemene i tidligere prosjekter.

5.3 Konfigurasjonsstyring

Til konfigurasjonsstyring vurderes det dithen at det å benytte konfigurasjonsstryingsverktøy vil være en større belastning enn nytteverdien for prosjektet. I stedet vil man avvente oppdateringer av de forskjellige verktøyene som brukes under utvikling/rapportskriving. Om slike skulle komme vil man på slutten av en iterasjon (slutten av en sprint) teste ut oppdateringene i et lukket miljø (for eksempel en virtuell maskin) for å vurdere om de vil ha større betydninger for prosjektet i sin helhet. Om oppdateringen er nyttig og endringene for prosjektets helhet er minimale vil alle gruppemedlemmene installere oppdateringen ved starten av neste iterasjon.

5.4 Risikoanalyse

Naturlig nok er et bachelorprosjekt det mest omfattende prosjektet for alle gruppemedlemmene. Når prosjektet som skal gjennomføres er i denne størrelsesorden er det viktig å identifisere risikoer som kan inntreffe underveis, og ta en vurdering av sannsynlighets- og alvorlighetsgrad.

5.4.1 Identifisere mulige risikoer

Indeks	Risko	Sannsynlighetsgrad	Alvorlighetsgrad
1	Verktøy slipper ut en ny oppdatering	Svært sannsynlig	Krever tiltak
2	Et gruppemedlem er syk eller fraværende over en lengre periode	Lite sannsynlig	Krever umiddelbart tiltak
3	Oppdragsgiver og/eller gruppemedlemmer ønsker ny funksjonalitet under utvikling	Sannsynlig	Krever tiltak
4	Oppstår en situasjon som stopper andre gruppemedlemmer fra å fortsette arbeid med sine oppgaver.	Lite sannsynlig	Krever umiddelbart tiltak
5	Ett eller flere av gruppemedlemmene har ikke fullført sine oppgaver ved slutten av en sprint.	Sannsynlig	Krever tiltak
6	Gruppemedlem møter ikke opp til avtalt møte med andre gruppemedlemmer, veileder eller oppdragsgiver.	Lite sannsynlig	Krever ikke tiltak
7	Ett gruppemedlem kommer ikke videre i oppgaven sin på egen hånd.	Svært sannsynlig	Krever tiltak

Tabell 1: Risikoer

5.4.2 Analyse

Under er en tabell som grafisk forteller hvor alvorlig de forskjellige risikoene er i forhold til hverandre. Naturlig nok representerer rød en kritisk situasjon, gul en alvorlig situasjon, mens grønn er en ikke-kritisk situasjon.

Sannsynlighet	Svært sannsynlig	-	1, 7	-
	Sannsynlig	-	3, 5	-
	Lite sannsynlig	6	-	2, 4
		Krever ikke tiltak	Krever tiltak	Krever umiddelbart tiltak
		Konsekvens		

Tabell 2: Risikoanalyse

5.4.3 Tiltak

Under er det en tilsvarende tabell 5.4.1. Bare her listes de forskjellige tiltakene som brukes for å rette opp i feilsituasjoner.

Indeks	Risko
1	Som nevnt i kapittel 6.3 Konfigurasjonsstyring avventes oppdateringer til starten på en ny sprint. Det tas vurderingen om oppdateringen har store innvirkninger ellers i prosjektet, og/eller oppdateringen er nyttig for prosjektet.
2	Enten må de andre gruppemedlemmene prøve å kompensere for fraværet til det fraværende gruppemedlemmet, ellers må det fraværende gruppemedlemmet ta igjen for tapt tid ved tilbakekomst. Valget må tas ut i fra hvor stort fraværet blir.
3	Gruppen og oppdragsgiver må holde en diskusjon for å finne ut om den nye foreslåtte funksjonaliteten øker nytteverdien av applikasjonen. Det må også tas en vurdering om den nye funksjonaliteten kan implementeres uten å sprenge tidsrammene for prosjektet.
4	Andre gruppemedlemmer kan assistere med oppgaven, ellers rådes det til å oppsøke hjelp fra teknologieksperten(ved NTNU i Gjøvik) innenfor fagfeltet oppgaven faller innenfor.
5	Gruppen tar en vurdering av hva ikke fungerte, som førte til at oppgaven ikke ble løst innen sprintens avslutning. Oppgaven tas dermed videre med til neste sprint.
6	Gruppemedlem møter ikke opp til avtalt møte med andre gruppemedlemmer, veileder eller oppdragsgiver. Reaksjonen vil avhenge av om gruppemedlemmet hadde klarert fraværet på forhånd eller ikke. Om fraværet var meldt fra om på forhånd, med godkjent begrunnelse, vil konsekvensene være minimale. Om fraværet ikke var melt fra om på forhånd vil dette noteres ned og de pårørte gruppemedlemmene vil ta fraværet til etterretning.
7	Se tiltak #4.

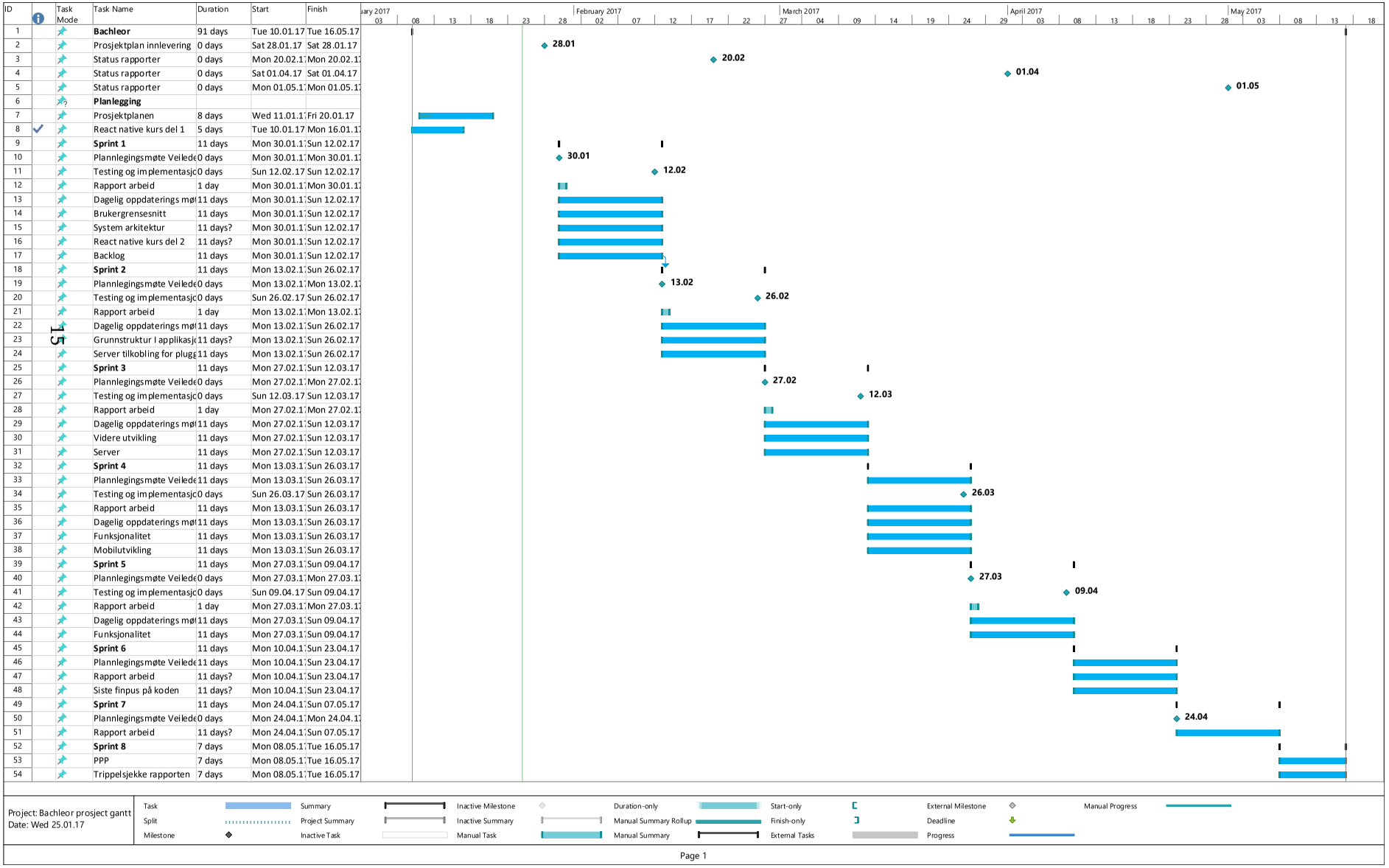
Tabell 3: Tiltak

5.4.4 Oppfølgning

For å være sikre på at de situasjonene som oppstår blir løst, vil alt bli loggført. Ved starten på den neste sprinten vil gruppen gå over forrige ukes logg, og ut i fra den ta en vurdering om situasjonen har blitt løst og på en måte alle på gruppen er tilfredsstilte med.

6 PLAN FOR GJENNOMFØRING

6.1 Gantt-skjema



ID	Task Mode	Task Name	Duration	Start	Finish
1		**Bachleor**	91 days	Tue 10.01.17	Tue 16.05.17
2		Prosjektplan innlevering	0 days	Sat 28.01.17	Sat 28.01.17
3		Status rapporter	0 days	Mon 20.02.17	Mon 20.02.17
4		Status rapporter	0 days	Sat 01.04.17	Sat 01.04.17
5		Status rapporter	0 days	Mon 01.05.17	Mon 01.05.17
6		**Planlegging**			
7		Prosjektplanen	8 days	Wed 11.01.17	Fri 20.01.17
8		React native kurs del 1	5 days	Tue 10.01.17	Mon 16.01.17
9		**Sprint 1**	11 days	Mon 30.01.17	Sun 12.02.17
10		Plannlegingsmøte Veilede	0 days	Mon 30.01.17	Mon 30.01.17
11		Testing og implementasjc	0 days	Sun 12.02.17	Sun 12.02.17
12		Rapport arbeid	1 day	Mon 30.01.17	Mon 30.01.17
13		Dagelig oppdaterings møl	11 days	Mon 30.01.17	Sun 12.02.17
14		Brukergrensesnitt	11 days	Mon 30.01.17	Sun 12.02.17
15		System arkitektur	11 days?	Mon 30.01.17	Sun 12.02.17
16		React native kurs del 2	11 days?	Mon 30.01.17	Sun 12.02.17
17		Backlog	11 days	Mon 30.01.17	Sun 12.02.17
18		**Sprint 2**	11 days	Mon 13.02.17	Sun 26.02.17
19		Plannlegingsmøte Veilede	0 days	Mon 13.02.17	Mon 13.02.17
20		Testing og implementasjc	0 days	Sun 26.02.17	Sun 26.02.17
21		Rapport arbeid	1 day	Mon 13.02.17	Mon 13.02.17
22		Dagelig oppdaterings møl	11 days	Mon 13.02.17	Sun 26.02.17
23		Grunnstruktur I applikasj	11 days?	Mon 13.02.17	Sun 26.02.17
24		Server tilkobling for plugg	11 days	Mon 13.02.17	Sun 26.02.17
25		**Sprint 3**	11 days	Mon 27.02.17	Sun 12.03.17
26		Plannlegingsmøte Veilede	0 days	Mon 27.02.17	Mon 27.02.17
27		Testing og implementasjc	0 days	Sun 12.03.17	Sun 12.03.17
28		Rapport arbeid	1 day	Mon 27.02.17	Mon 27.02.17
29		Dagelig oppdaterings møl	11 days	Mon 27.02.17	Sun 12.03.17
30		Videre utvikling	11 days	Mon 27.02.17	Sun 12.03.17
31		Server	11 days	Mon 27.02.17	Sun 12.03.17
32		**Sprint 4**	11 days	Mon 13.03.17	Sun 26.03.17
33		Plannlegingsmøte Veilede	11 days	Mon 13.03.17	Sun 26.03.17
34		Testing og implementasjc	0 days	Sun 26.03.17	Sun 26.03.17
35		Rapport arbeid	11 days	Mon 13.03.17	Sun 26.03.17
36		Dagelig oppdaterings møl	11 days	Mon 13.03.17	Sun 26.03.17
37		Funksjonalitet	11 days	Mon 13.03.17	Sun 26.03.17
38		Mobilutvikling	11 days	Mon 13.03.17	Sun 26.03.17
39		**Sprint 5**	11 days	Mon 27.03.17	Sun 09.04.17
40		Plannlegingsmøte Veilede	0 days	Mon 27.03.17	Mon 27.03.17
41		Testing og implementasjc	0 days	Sun 09.04.17	Sun 09.04.17
42		Rapport arbeid	1 day	Mon 27.03.17	Mon 27.03.17
43		Dagelig oppdaterings møl	11 days	Mon 27.03.17	Sun 09.04.17
44		Funksjonalitet	11 days	Mon 27.03.17	Sun 09.04.17
45		**Sprint 6**	11 days	Mon 10.04.17	Sun 23.04.17
46		Plannlegingsmøte Veilede	11 days	Mon 10.04.17	Sun 23.04.17
47		Rapport arbeid	11 days?	Mon 10.04.17	Sun 23.04.17
48		Siste finpus på koden	11 days?	Mon 10.04.17	Sun 23.04.17
49		**Sprint 7**	11 days	Mon 24.04.17	Sun 07.05.17
50		Plannlegingsmøte Veilede	0 days	Mon 24.04.17	Mon 24.04.17
51		Rapport arbeid	11 days?	Mon 24.04.17	Sun 07.05.17
52		**Sprint 8**	7 days	Mon 08.05.17	Tue 16.05.17
53		PPP	7 days	Mon 08.05.17	Tue 16.05.17
54		Trippelsjekke rapporten	7 days	Mon 08.05.17	Tue 16.05.17

Bibliografi

[1] Hafslund. Strømforbruk og strømsparing. https://www.hafslund.no/strom/privat/stromforbruk/2025.

[2] Wikipedia. Studiepoeng. https://no.wikipedia.org/wiki/Studiepoeng.

[3] Solum, H., Holtskog, T., & Enes, J. 2017.
Timeføring - google sheets.
https://docs.google.com/spreadsheets/d/1Pn6Qkyn6xqGT148CbNMZWzan6RDstG6yHhitiRoBPlc/edit?usp=

[4] ESLint. Eslint - about. http://eslint.org/about/.

L Første statusrapport

Dette vedlegget består av den første statusrapporten levert inn for denne prosjektperioden. Formålet med rapporten var å fortelle litt om hva som arbeides med på daværende tidspunkt. Rapporten hadde innleveringsfrist til veileder 20. februar 2017.

Første statusrapport - EcoHeat

Planlegging

Vi har laget et GANTT diagram som tar for seg hovedtrekkene av prosjektperioden som vi kommer til å følge, med forbehold om endringer underveis ettersom vi følger SCRUM. I den første sprinten tok vi for oss mesteparten av planleggingen, hvor vi har fulgt RUP 4+1 og arbeidet med mobil og web prototyper. Prototypene ble ikke helt ferdige i sprint 1, dermed har vi valgt å fortsette med dette i sprint 2, der vi også skal begynne med selve utviklingen.

Organisering av gruppens arbeid og ansvarsområder

Jostein Enes er gruppens prosjektleder og har i oppgave å passe på at gruppen holder sine tidsfrister og opptrer som SCRUM master.

Alle gruppemedlemmene møter på skolen fra 8:15 og arbeider til tidligst klokka 14:00 fra Mandag - Fredag. Her arbeider gruppemedlemmene med oppgaver fra sine egne ansvarsområder, men det er alltid åpent for diskusjon internt om det skulle være uklarheter. Det at vi alle sitter på skolen gjør at vi får rask tilbakemelding fra de andre på gruppen.

Jostein Enes har hatt ansvar for:

- Web prototype Lo-Fi
- Web prototype Hi-Fi.
- Process view, class diagram (RUP 4+1)

Henrik Solum har hatt ansvar for:

- Physical view, deployment view diagram (RUP 4+1)
- Process view, activity diagram (RUP 4+1)
- Lo-Fi prototype for mobilapplikasjonen.
- Hi-Fi prototype for mobilapplikasjonen.

Tarjei Holtskog har hatt ansvar for:

- Utvidet risikoanalyse
- Use Case diagram

Felles ansvar:

- Personas
- Scenario
- Prosjektplan
- Testing av lignende eksisterende løsninger
- Ukentlige møter med oppdragsgiver og veileder så langt det har vært nødvendig og mulig
- Sette seg inn i nye programmeringsspråk

Løsningsmetode

Oppgaven skal løses ved å benytte programmeringsspråket React, og utvidelsen React Native for mobil. I tillegg vil det måtte benyttes server og database, men her har vi ikke satt oss inn i hvordan man skal løse dette oppsettet. For mobilapplikasjonen kan det også være at vi må bevege oss inn på Java hvis vi trenger funksjonalitet som ikke React Native innehar, dette gjelder ikke iOS da vi ikke har utstyr for å utvikle i deres miljø.

Rapportskriving.

Rapportskrivingen har ikke blitt påbegynt enda, alt arbeid ligger i egne dokumenter enn så lenge.

Muligheter og utfordringer

Oppdragsgiver har gitt oss tilgang til flere kurs innenfor React og React Native for å gjøre det enklere for oss å komme i gang med arbeidet. Han har også gitt oss mulighet til å teste eksisterende løsninger for å få inntrykk og idéer til vår utvikling.

Hovedutfordringen enn så lenge er å finne ut hvordan vi skal angripe problemet med å opprette kommunikasjon mellom smartpluggen og internett. Oppdragsgiver har sagt han skal sende oss det reelle utstyret i løpet av de kommende ukene, så vi får begynt å testet ut hvordan alt fungerer under panseret.

Gruppens kjemi

Hittil i prosjektet har vi fungert godt som gruppe uten store konflikter, og vi kan ikke se noen grunn til at dette skal endre seg. Det at vi kan sitte og jobbe sammen i et godt miljø er en stor motivasjonsfaktor.

Veileder

Veileder er lett å få kontakt med, god til å få oss inn på riktig spor og oppfordrer oss til å benytte de ressursene som er på huset. Vi føler at kontakten mellom oss fungerer godt.

M Andre statusrapport

Dette vedlegget består av den andre statusrapporten levert inn for denne prosjektperioden. Formålet med rapporten var å fortelle litt om hva som arbeides med på daværende tidspunkt. Rapporten hadde innleveringsfrist til veileder 1. april 2017.

Andre statusrapport - EcoHeat

Planlegging

I den første statusrapporten skrev vi om bruken av RUP 4+1 og SCRUM. Vi har nå skapt mange av artefaktene som går under RUP 4+1 og vi er i 5. sprint under SCRUM. Utenom artefaktene har vi i stor grad fullført prototypene for både mobil og web applikasjon. Når det kommer til utviklingen av selve applikasjonene har vi fokusert på selve brukergrensesnittet enn så lenge, dette fordi vi har ventet på oppdateringer fra oppdragsgiver angående hvordan selve smartpluggen og backend kommer til å fungere. Mobil ligger veldig godt an, med større deler av brukergrensesnittet ferdig, mens webapplikasjonen ligger en del etter. Naturlig nok har det blitt litt forskjell da vi har fokusert på mobil, der to av tre gruppemedlemmer har arbeidet på mobil brukergrensesnittet i denne perioden.

Organisering av gruppens arbeid og ansvarsområder.

Nå som vi har kommet inn i utviklingsfasen av prosjektet har Jostein Enes og Henrik Solum arbeidet med implementasjonen av mobil grensesnittet og Tarjei Holtskog har arbeidet med web grensesnittet. Vi møter fortsatt 8:15 hver eneste dag på skolen, med enkelte unntak, noe som vi fortsatt synes fungerer.

Jostein Enes opptrer fortsatt som SCRUM master, men ansvaret som inngår er i større grad fordelt utover alle gruppemedlemmene, ettersom vi bare er tre personer i gruppen.

Som nevnt i den første statusrapporten, startet vi prosjektet med å holde to ukers lange sprinter, men etter tredje sprinten fant vi ut at dette ikke fungerte. Vi slet med å holde arbeidsmengden i en realistisk størrelse, da ingen av oss har gjennomført et såpass grundig utviklingsprosjekt før. Derfor har vi gått over til 1 ukes sprinter og det føles enklere å estimere og forutse hvordan vi kommer til å bruke tiden vår, over et kortere tidsintervall.

Som nevnt har vi nå fått fullført prototyper for både mobil og web applikasjon, til et punkt hvor oppdragsgiver virker svært fornøyd i sprint nummer 3. Ut i fra den originale oppgavebeskrivelsen[sitat] var det ønsker om interaksjonsdesign. Ettersom som ingen av medlemmene på gruppen har spesialiseringer innenfor design, har vi kontaktet blant annet Eivind Arnstein Johansen, universitetslektor ved NTNU i Gjøvik, som har gitt oss mange gode råd og tips for selve utformingen av applikasjonene.

I sprint 4 og sprint 5 har vi arbeidet med å implementere prototypene, slik at vi har en GUI som samsvarer og kan plugge inn funksjonalitet i sprintene fremover.

Løsningsmetode

Som nevnt har vi begynt utviklingen av de forskjellige brukergrensesnittene. Vi kjører Pull requests hver gang funksjonalitet ansees som ferdig av den gjeldende utvikleren. Da blir all kildekode gjennomgått av de andre gruppemedlemmene, og må godkjennes for pull requesten går igjennom. På denne måten hjelper vi hverandre med å skrive god og lesbar kode, og alle får en god oversikt over alle aspektene ved prosjektet. Ved uenigheter føres det diskusjoner, hvor flertallet ender opp med å bestemme til slutt.

Etter tidligere samtaler med oppdragsgiver hadde vi tenkt til å skrive et script for simulering av kommunikasjonen med mqtt-protokollen, men etter nye samtaler blir dette revurdert. Mer om dette lenger nede.

Rapportskriving.

Vi har ikke begynt å fokusere på selve rapportskrivingen ennå, men vi tar å dokumenterer mye av det vi arbeider med underveis, slik at det blir lettere å det inn i rapporten. Vi skriver blant annet referater etter møter, og all utviklingsarbeid blir loggført i både Jira og Bitbucket.

Planen er i utgangspunktet å starte på selve rapporten rundt en måned før fristen, noe som betyr rundt 15. April.

Ressurser

I dette prosjektet har vi også tatt i bruk flere typer ressurser. Naturlig nok er internett noe man bruker hyppig i et slikt prosjekt, men vi har også som nevnt fått hjelp både av arbeidsgiver og av diverse lektorer ved NTNU i Gjøvik.
Eivind Johansen som nevnt gikk igjennom personas, scenarioene og prototypene våre, som førte til at vi endte opp med å forkaste personasene og scenarioene. Dette gjorde vi fordi vi innså vi hadde kommet så langt på prototypene at det ikke var verdt tiden for å forbedre disse, da vi egentlig manglet en del forarbeid i form av brukerundersøkelser og intervjuer for å skape gode personas. Til gjengjeld har vi fått forbedret prototypen basert på tilbakemeldingene fra Eivind.
Vi har også holdt en god dialog med oppdragsgiver rundt prototypen, der vi ofte har kjørt ukentlige tilbakemeldingsrunder, helt til prototypene ble ansett som ferdigstilt.
Vi har også vært i kontakt med Tom Røise angående oppsett av vårt Use Case diagram. Her kom vi frem til at aktører, systemer og undersystemer ikke nødvendigvis var satt opp i henhold til hvordan oppgaven var skrevet. Disse feilene ble så rettet opp.

Revurdering av oppgaven

Vi har etter nye samtaler med oppdragsgiver kommet frem til at det kan bli vanskelig å få fullført hele oppgaven slik den er definert. Han har derfor fått inn en person i teamet sitt som kan ta på seg oppsettet med server for kommunikasjon mellom smartpluggene, applikasjonene og database. Det kommer da til å bli opprettet et API som vi kan bruke for kommunkasjonen mellom applikasjonen og smartpluggene.

N Tredje statusrapport

Dette vedlegget består av den tredje og siste statusrapporten levert inn for denne prosjektperioden. Formålet med rapporten var å fortelle litt om hva som arbeides med på daværende tidspunkt. Rapporten hadde innleveringsfrist til veileder 1. mai 2017.

Tredje statusrapport - EcoHeat

Planlegging (fremdriftsplan)

All planlegging omkring prosjektet er i stor grad fullført, men det gjenstår fortsatt en god del arbeid for å nå det målet vi satt oss i begynnelsen av prosjektperioden. Brukergrensesnitt for mobil har kommet ganske langt, og det begynner å bli klart for å plugge inn funksjonalitet bak de forskjellige interaksjonsmulighetene. Av funksjonalitet har oppkobling mot Firebase blitt implementert, til brukerhåndtering og datalagring, men det er fortsatt mye funksjonalitet som mangler. Planen fremover for mobilapplikasjonen er å få innhentet data fra Firebase, i stedet for de lokale "dummy" dataene som blir brukt for øyeblikket, I tillegg er det deler av brukergrensesnittet som fortsatt må ferdigstilles, blant annet "Mine enheter" og Kalenderen(ikke påbegynt ennå).

For web har utviklingen naturlig nok ikke hatt den samme fremdriften da man bare har hatt en person for denne utviklingen. Her har man fått implementert det omliggende systemet for autentisering av brukere via firebase, navigering i applikasjonen, endring av bolig og modus samt visning av statistikk i hjemmet. I tillegg er utviklingen for oversikt av enhetene påbegynt og vil forhåpentligvis bli ferdigstilt før mai.

Rapportskriving vil naturligvis bli hovedfokuset fremover, og mer utvikling på dette stadiet vil foregå på frivillig basis eller om det skulle bli tid til overs etter at rapporten er ferdigskrevet.

Tidsfrister

For å komme over til rapportskriving, ble det bestemt at vi måtte avslutte utvikling etter påske. Selv om det fortsatt er igjen mange oppgaver for utviklingen, er det rett og slett for dårlig tid til å fullføre alt det vi planla å gjøre i løpet av prosjektperioden. Ettersom tiden etter påske skal brukes til å skrive bachelor rapporten, vil videre utvikling bare forekomme om rapporten er ferdig i god tid før endelig frist, eller ved at noen tar på seg ekstraarbeid.

Organisering av gruppens arbeid og ansvarsområder

Ingenting har endret seg siden forrige statusrapport når de kommer til organisering av gruppens arbeid og ansvarsområder.

Klargjøring av problemstilling

Ingenting har endret seg siden forrige statusrapport.

Løsningsmetode

Som nevnt tidligere måtte videre utvikling bli kraftig nedprioritert i perioden fremover. Om det blir tid til utvikling, vil det skje på samme måte som nevnt i forrige statusrapport. Den eneste relevante endringen er at det ikke blir tid til de grundige kodegjennomgangene av alle gruppemedlemmene, men heller fokus på å få ting til å fungere.

Rapportskriving

Nå har rapportskrivinga blitt påbegynt, det meste av innledningen er ferdig, analyse er bare påbegynt mens design er godt underveis. Gruppen følger malen som ble gitt under lynkurset og fordeler oppgavene ved at medlemmene fritt velger fra den.

O Møtereferater oppsummert

Dette vedlegget inneholder referater fra møter med forskjellige personer som har vært inkludert i prosjektet, og i bachelorgruppen internt. Dette inkluderer ikke all kommunikasjon på e-post, tekstmeldinger og telefonsamtaler, men heller møter der det har vært avtalt et tid og sted.

O.1 Veileder

24.01.2017

Diskusjon angående forbedringer til prosjektplan. Det tas opp at tre statusrapporter skal skrives, og at diskusjon av utviklingsspråk må tas med oppdragsgiver.

31.01.2017

Diskusjon angående videre forbedringer til prosjektplan som har innlevering rett rundt hjørnet. Veileder sier mer spesifikt at innledningen er for tynn, ettersom den skal mest sannsynlig tas med videre til selve bachelorrapporten.

27.02.2017

Første statusrapport innlevert, veileder går igjennom mulig forbedringspotensiale til den.

27.03.2017

Andre statusrapport førsteutkast blir diskutert, forbedringspotensiale tas opp.

14.04.2017

Spørsmål rundt rapporten tas opp med tanke på at oppgaveomfanget har blitt endret. Diskusjon rundt plasseringen av forskjellige tema i henhold til kapitler i rapporten.

02.05.2017

Diskusjon rundt de delene av rapporten som er skrevet så langt.

05.05.2017

Videre veiledning om rapportens innhold, og de forbedringer som har blitt gjort basert på kommentarer fra forrige møte.

12.05.2017

Veiledning mer spesifikt til de siste kapitlene av rapporten.

O.2 Oppdragsgiver

Møte med oppdragsgiver foregikk i hovedsak via nettsiden http://www.appear.in med videostreaming av begge parter.

18.01.2017

Oppstarts møte for å finne ut hva som allerede eksister/påtenkt av systemet, eventuelle ønsker.

- Smartplugg vil ha WiFi, termostat og bryter.
- Databasevalg er opp til bachelorgruppen.
- Ønskelig med graf med forskjellig data(strømforbruk, penger spart, osv).
- React & React-native som utviklingsverktøy.

27.01.2017

Møte for å ta opp spørsmål rundt teknologien, forventningene, inspirasjonen for oppgaven, ønsket funksjonalitet og sikkerhet til arbeidsgiver.

- Mest sannsynlig MQTT protokoll for kommunikasjon med smartplugg.
- Ingen spesifikke forventinger til eventuell release ved endt oppgave.
- Inspirasjon til oppgaven kom fra egen erfaring fra oppdragsgiver. Måtte skru varmeovn av og på manuelt for å spare strøm, når boligen var forlatt.
- Ønsker en billig og enkel løsning, sammenlignet med markedet idag.
- Ønsket funksjonalitet: Kalender, Koble sammen brukere fra samme bolig, moduser, sammenligning av strømforbruk før/etter installasjon og forskjellige tillatelser for brukerne i et hus. Åpent for mer funksjonalitet.
- Sikkerhet ikke en prioritet i denne oppgaven, løses muligens via MQTT.

07.02.2017

Diskusjon angående eksisterende løsninger, etter at oppdragsgiver sendte ned maskinvare fra disse løsningene. I tillegg diskuteres Lo-Fi prototypene.

- Bachelorgruppen er tydelig mer skeptisk til designet av mange av løsningene.
- Liker Lo-Fi og oppdragsgiver ønsker at bachelorgruppen skal mange av valgene rundt design, heller diskutere i etterkant.
- Mer snakk om bruksområdene til MQTT.

09.03.2017

Mye e-post kommunikasjon angående prototyper siden sist møte. Møtet foretok prototypene, nyhet om smartpluggene og litt om prosjektet ellers utenfor oppgaven.

- Oppdragsgiver synes det er mye bra i prototypene, noe uklart på enkelte områder.
- Smartplugg prototyp har blitt produsert fra Kina, og oppdragsgiver har fått kommunisert med dem via script. Plugger skal sendes.
- Oppdragsgiver har snakket Innovasjon Norge.
- Sikkerhet og utseende er ikke top prioritet, heller få til funksjonalitet.

27.03.2017

Møte for å bestemme prioriteringer, da det begynner å bli stramt med tid. Det er også kommet inn flere på teamet til oppdragsgiver som skal ta for seg server.

- Av / på bryter via applikasjonen er viktigst.
- Server antas å være ferdig i løpet av 1-2 uker.
- Diskusjon angående datastruktur.
- Utviklerarbeidet blir opp mot server API og ikke smartplugg direkte.

04.04.2017

Diskusjon angående datastruktur i Firebase. De to nye på teamet er med på møtet.

O.3 Internt

Internt ble det holdt muntlige møter i starten og slutten av enhver sprint.

P Arbeidslogg

I dette vedlegget er det skrevet en oppsummering av hva som ble gjort i de forskjellige ukene under prosjektperioden. Dette er oppsummert fra gruppens 'Timeføring'(), der alle medlemmene har skrevet ned tidspunktene når de har arbeidet og hva de arbeidet med under hele prosjektperioden.

Uke 2 - 2017	
Oppsummering	Oppstartsmøte med veileder og planlegging av administrativt arbeid. Gruppen begynner å gå gjennom kurs for React-native på http://www.udemy.com
Navn	**Antall timer**
Henrik	18
Jostein	18
Tarjei	21.8

Uke 3 - 2017	
Oppsummering	Alle gruppemedlemmene fullfører kurset for React-native, og begynner å arbeide med prosjektplanen. Prosjektavtale underskrives av gruppen og sendes til oppdragsgiver. Første møte med oppdragsgiver holdes denne uken.
Navn	**Antall timer**
Henrik	29.4
Jostein	29.8
Tarjei	30.1

Uke 4 - 2017	
Oppsummering	Prosjektplanen blir forbedret basert på diskusjon med veileder og utveksling med annen bachelorgruppe. Prosjektplan blir ferdigskrevet og sendt til veileder for gjennomgang. Gruppen begynner gjennomføringen av kurs for React & Redux. Det blir holdt et møte med oppdragsgiver.
Navn	**Antall timer**
Henrik	30
Jostein	29.7
Tarjei	30

Uke 5 - 2017	Sprint 1
Oppsummering	En del sykdom blant gruppen denne uken. Begynner å strukturere arbeidet i to ukers sprinter, med hjelp av Jira. Skriver personas og scenarioer og begynner på Lo-Fi prototyping av begge løsningene. Fortsetter også på React & Redux kurset og tester ut eksisterende løsninger med maskinvare sendt til oss fra oppdragsgiver.
Navn	**Antall timer**
Henrik	24.3
Jostein	26.4
Tarjei	24.7

Uke 6 - 2017	
Oppsummering	Gruppen fortsetter å teste ut eksisterende løsninger i forberedelse til diskusjon med oppdragsgiver senere i uken. Lo-Fi prototyper fullføres og godkjennes av oppdragsgiver, og Hi-Fi protoyper blir påbegynnes. Planlegging og fortsettelse av React & Redux kurs.
Navn	**Antall timer**
Henrik	33.3
Jostein	38.4
Tarjei	26.2

Uke 7 - 2017	**Sprint 2**
Oppsummering	Møte med veileder i samhold med planlegging rundt sprint 2. Hi-Fi prototyping og Use case arbeid er hovedfokuset. Skriver den første av tre statusrapporter for prosjektet. Sender også ut mail til Eivind Johansen for råd rundt prototypene. Fortsetter med React & Redux kurs mot slutten.
Navn	**Antall timer**
Henrik	31.7
Jostein	15.3 (hjemme hos familien denne uken)
Tarjei	27.3

Uke 8 - 2017	
Oppsummering	Gruppen gjennomgår et lynkus om profesjonell git bruk. Hi-Fi prototyper forbedres etter dialog frem og tilbake over ukene med oppdragsgiver.
Navn	**Antall timer**
Henrik	32.6
Jostein	29.5
Tarjei	26.2

Uke 9 - 2017	**Sprint 3**
Oppsummering	Planlegging av sprint og møte med veileder om progressjon. Oppsett av automatisk versjonskontroll og internasjonalisering for web og mobil prosjektene. Prototyper sendes til oppdragsgiver nok en gang, med innfletting av rådene til Eivind Johansen og et par mindre brukertester. Utvikling av GUI begynner basert på prototypene.
Navn	**Antall timer**
Henrik	28.8
Jostein	26.8
Tarjei	25.3

Uke 10 - 2017	
Oppsummering	Utvikling av GUI fortsetter og møte med oppdragsgiver blir holdt for å diskutere veien videre.
Navn	**Antall timer**
Henrik	28.6
Jostein	29.6
Tarjei	25.9

Uke 11 - 2017	Sprint 4
Oppsummering	Det blir bestemt at sprintene bare skal vare i 1 uke, ettersom gruppen sliter med å sette realistiske estimater. Videre utvikling av GUI.
Navn	**Antall timer**
Henrik	29.1
Jostein	29.1
Tarjei	30.2

Uke 12 - 2017	Sprint 5
Oppsummering	Gruppen sliter fortsatt med å lage gode estimater, men ser forbedringer. Videre utvikling av GUI.
Navn	**Antall timer**
Henrik	29.7
Jostein	28.5
Tarjei	29.3

Uke 13 - 2017	Sprint 6
Oppsummering	Møte med veileder og oppdragsgiver. Oppdragsgiver opplyser om at han har fått inn flere på teamet sitt i Trondheim. Videre utvikling av GUI og begynner å fylle det med 'mockup' data.
Navn	**Antall timer**
Henrik	29
Jostein	27
Tarjei	19.8

Uke 14 - 2017	Sprint 7
Oppsummering	Møte med oppdragsgiver blir holdt der gruppen møter de nye teammedlemmene. Stian Årsnes satt opp et forslag til databasestruktur, som ble diskutert på et spontant møte. Det ble bestemt av gruppen at dette var den siste uken for utvikling, da rapporten må prioriteres. Utvikling vil heller fortsette dersom det blir tid til det.
Navn	**Antall timer**
Henrik	24.1
Jostein	23.6
Tarjei	21.6

Uke 15 - 2017	Påskeferie
Oppsummering	Det ble avtalt å ha en ukes påskeferie, med mulighet for frivillig arbeid.
Navn	**Antall timer**
Henrik	0
Jostein	0
Tarjei	3

Uke 16 - 2017	Sprint 8
Oppsummering	Sprint 8 ble påbegynt bare for å kunne kategorisere den tiden som eventuelt blir brukt til utvikling. Bachelorrapporten blir påbegynt i full fart, men bremses litt ned av sykdom og forberedelse/gjennomføring av eksamen i andre fag for noen av gruppemedlemmene.
Navn	**Antall timer**
Henrik	10.3
Jostein	17.5
Tarjei	12.3

Uke 17 - 2017	
Oppsummering	Skriving av bachelorrapport er hovedfokus.
Navn	**Antall timer**
Henrik	27.9
Jostein	22.9
Tarjei	26.5

Uke 18 - 2017	
Oppsummering	Videre arbeid med bachelorrapporten, innledning og hovedkapitlene nærmer begynner å bli ganske utfyllende. Forbedrer rapporten basert på diskusjon/kommentarer fra møtet med veileder på tirsdag & fredag.
Navn	**Antall timer**
Henrik	34.1
Jostein	24.5
Tarjei	30.8

Uke - 2017	
Oppsummering	Avsluttende arbeid med bachelorrapporten, med gjennomgang av tilbakemeldinger fra både veileder og de som har korrekturlest rapporten.
Navn	**Antall timer**
Henrik	55.6
Jostein	49.7
Tarjei	54.6

Uke - 2017	
Oppsummering	Gjennomlesing av rapporten tok lengre tid enn først antatt, og måtte dermed bli gjort natt til mandag. All kildekode samles i master branch og vi forbereder innlevering.
Navn	**Antall timer**
Henrik	19.7
Jostein	20.8
Tarjei	20.3