Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Abdulhadi Anwar Mahdi Al-Sayed
Elliot Sveum Torp
Rihards Daniels Ustinovics
Sindre Emil Vikre

Automatisering av ladebrikkeregistrering og samlet strømforbruk

Bacheloroppgave i Programmering
Veileder: Frode Haug
Mai 2022

NTNU
Kunnskap for en bedre verden

Abdulhadi Anwar Mahdi Al-Sayed
Elliot Sveum Torp
Rihards Daniels Ustinovics
Sindre Emil Vikre

Automatisering av ladebrikkeregistrering og samlet strømforbruk

Bacheloroppgave i Programmering
Veileder: Frode Haug
Mai 2022

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

SAMMENDRAG

Tittel :	Automatisering av ladebrikkeregistrering og samlet strømforbruk	Dato : 20.05.2022
Deltaker(e) :	Abdulhadi Anwar Mahdi Al-Sayed, Elliot Sveum Torp, Rihards Daniels Ustinovics, Sindre Emil Vikre	
Veileder(e) :	Frode Haug	
Oppdragsgiver :	ETC Electric Time Car AS	
Stikkord:	Ladebrikke, ETC, CarAdmin, API, Automatisering, Web, EV	

Antall sider : 81	Antall vedlegg : 11	Publiseringsavtale inngått : Åpen

Kort beskrivelse av bacheloroppgaven :

Sammen med elbilmarkedet øker etterspørselen etter systemer som kontrollerer elbilflåter. ETC ønsket at vi skulle undersøke og utvikle et program som løser en del av kompleksiteten knyttet til registrering av ladebrikker til en ladestasjon. Programmet skulle utvikles som en modul i deres CarAdmin-løsning. CarAdmin har moduler for å kontrollere kjøretøyflåter basert på behov fra brukerne. Målet med modulen vi utviklet er å automatisere registreringen av ladebrikkeregistrering. En privat elbil-eier trenger kanskje ikke dette verktøyet, men selskaper som har en stadig voksende bilflåte trenger en automatisert prosess. En del av kompleksiteten i denne oppgaven er hvordan vi måtte automatisere registreringsprosessen for flere ulike ladeselskap til en enkel ladebrikke. Ved å lagre ladebrikker i CarAdmin og autentisere hver bruker til et ladeselskap, utviklet vi en modul der en bruker enkelt kan legge inn flere ladebrikker og registrere/avregistrere dem som en enkelt handling til det foretrukne ladeselskapet. En sekundær oppgave var innsamling av strømforbruk fra hvert ladeselskap. Vi ønsket å hente strømforbruk per elbil, men mangelen på teknologi som støtter dette betydde at vi heller måtte hente strømforbruk per hjemmeladestasjon. Resultatet ble en modul for automatisk registrering av ladebrikker og en modul for innsamling av strømforbruk til ladestasjoner.

ABSTRACT

Title :	Automation of charging tag registration and total power consumption	Date : 20.05.2022
Participants :	Abdulhadi Anwar Mahdi Al-Sayed, Elliot Sveum Torp, Rihards Daniels Ustinovics, Sindre Emil Vikre	
Supervisor(s) :	Frode Haug	
Employer :	ETC Electric Time Car AS	
Keywords :	Charging Tag, ETC, CarAdmin, API, Automation, Web, EV	
Number of pages/words : 81	Number of appendix : 11	Availability : Open

Short description of the bachelor thesis :

Alongside the EV market, the demand for systems that control vehicle fleets is growing. ETC proposed that we research and develop a method in which we solve a part of the complexity pertaining the registration of EV charging tags to a charging station, as a module in their CarAdmin solution. CarAdmin itself includes modules for controlling vehicle fleets based on the needs of its users. The goal of the module we develop is to automate the registration of charging tags. A private EV owner might not need this tool, however companies that need an ever-expanding fleet of vehicles do need an automated process. Part of the complexity in this task is how we had to automate this process for multiple charging stations to a single charging tag. By saving the charging tag to CarAdmin, and authenticating each user to a charging station provider, we developed a module where a user could simply input multiple charging tags, and register/unregister all of them as a singular action to the preferred charging station provider. A secondary task was the collection of power usage from each charging station provider. We wanted to narrow the charging details down to a per vehicle basis, however the lack of technology supporting this endavour meant we had to expand the detail up to a per home charging station basis. The result was a module for the automatic registration of charging tags and a module for the collection of power usage for charging stations.

Forord

Oppdraget for denne bacheloroppgaven er levert av Electric Time Car AS. "ETC er et innovativt IT-selskap med nyskapende total-løsning for helhetlig kjøretøy oppfølging kalt CarAdmin, som benyttes i små og store kjøretøyparker"[1].

Først ønsker vi å takke produkteier Dag Solhaug for en spennende og lærerik oppgave. Takk til Øyvind Flatval og Sebastian Balsam for god hjelp og oppfølging gjennom hele prosjektet. Vi ønsker også å takke Frode Haug for veiledningen gjennom bacheloroppgaven.

Til slutt ønsker vi å takke medlemmene i gruppen for god innsats og arbeidsmoral.

[1] ETC. *Om oss*. URL: https://caradmin.no/om-oss/ (sjekket 12. jan. 2022).

Innhold

Figurer

Tabeller

1 Introduksjon

I dette kapittelet skal vi definere omfanget, målene og målgruppen for prosjektet. vi skal og ta for oss gruppens bakgrunn og kunnskap, rammer, ansvarsforhold, roller og rapportens struktur.

1.1 Fagområde, avgrensning og oppgavebeskrivelse

1.1.1 Fagområde

Tradisjonelt sett har biler vært drevet av fossilt drivstoff. Derimot har elbiler tatt den største delen av det norske markedet de siste årene, og vil trolig vokse enda mer framover[2]. Dette medfører naturligvis mange ulike problemstillinger og utfordringer som må løses. Hvem skal eie elbil laderne? Hvilke betalingsmiddel skal laderne akseptere? Hvem skal få lov til å bruke laderne? Hvilke sikkerhetsmekanismer må laderne ha for å hindre misbruk? Hvordan foregår registrering hos de ulike ladeselskapene, og hvordan skal man få oversikt over strømforbruket til en elbil? Alle problemstillingene er relevante for oppgaven, men det er registrering hos ladeselskaper og samling av strømforbruk vi skal fokusere på å gjøre mer effektivt.

1.1.2 Avgrensning

Vi skal jobbe med å utvide den allerede eksisterende løsningen til ETC, dette setter naturligvis noen avgrensninger til hva vi skal fokusere på innen fagområdet. ETC sin tjeneste, CarAdmin, har som mål i å hjelpe bedrifter og kommuner med å få bedre oversikt og mer effektiv drift av kjøretøyene deres. Vi skal utforske problematikken rundt hva som skjer når det er mange elbiler som skal lades, registreres, og holdes oversikt over. Er det mulig å registrere flere elbiler hos ulike ladeselskaper med bare ett tastetrykk? Hvordan kan en bilansvarlig se hvor mye strøm som blir brukt når en bil lader? Hvilken avdeling i bedriften skal betale? Hvordan kan vi gjøre alt dette enklere og mer effektivt?

1.1.3 Oppgavebeskrivelse

For en privatperson med en til to biler kan registrering av ladebrikke hos en eller flere ladeselskaper være en enkel oppgave. Når et firma som eier en elbil-flåte må utføre samme prosess kan det være en tidkrevende oppgave, ettersom hver brikke må manuelt registreres hos alle de ulike ladeselskapene som benyttes. Når biler lader hos flere ulike ladestasjoner blir er det komplisert å holde oversikt over hvor mye strøm som blir brukt. For en total oversikt må en inn i løsningen til hvert av ladeselskapene og legge sammen forbruk. Dette er igjen veldig tidkrevende for et firma med en stor elbil-flåte (mer om dette i seksjon 2.1).

[2]Norsk elbilforening. *Norwegian EV market*. URL: https://elbil.no/english/norwegian-ev-market/ (sjekket 10. mai 2022).

Oppgaven er todelt. Den første delen er å automatisere registreringsprosessen for el-bil ladebrikker (RFID-brikker) når man skal registrere samme brikke hos flere ulike ladestasjonselskaper. Del to av oppgaven er å hente ut strømforbruk og fakturainformasjon fra ulike ladeselskapers løsning, og samle det i CarAdmin. Løsningen skal ha følgende funksjonalitet:

Generelt

- Vi skal hovedsakelig fokusere på ladestasjonene som NRK benytter siden de er testkunde.
- Systemet vi utvikler må være kompatibel og kunne integreres i CarAdmin.
- Vi skal benytte eksisterende APIs for å kommunisere med ladeselskap sine løsninger.
- Vi skal knytte RFID-brikker til kjøretøy i CarAdmin
- Ladeselskapene vi skal fokusere på er de største interne ladere, i tillegg til de eksterne laderene Eviny og E.ON.
  - Vi har valgt disse ladestasjonene på grunn av tilgang til API, og etter ønske fra NRK.

Ladebrikke behandling

- Registrering og sletting av ladebrikker skal skje gjennom CarAdmin sitt brukergrensesnitt.
  - Bruker må legge til registreringsnummer til bilen, og RFID-nummer kun en gang.
- Automatisk knytte én eller flere brikker hos flere forskjellige ladeselskap.
  - Etter brukeren har skrevet inn informasjonen skal brikken bli registrert hos flere ladeselskaper.

Samlet strømforbruk

- Automatisk få inn strømforbruk etter en bil har ladet.
  - Strømforbruk hentes fra en API som kommuniserer med laderene.
  - Strømforbruket skal vises i CarAdmin sitt grensesnitt.

Vi skal utvikle følgende deler:

- Automatisering av ladebrikke registreringen.
- Samling av strømforbruk og fakturainformasjon.
- Nye grensesnitt i CarAdmin for å registrere RFID-brikker.
- Bygge på ETC sin eksisterende database.

1.2 Prosjektmål

Vi skiller mellom effektmål og resultatmål for prosjektet. De forskjellige måltypene viser det som er viktig for gruppen å oppnå gjennom oppdragsperioden.

1.2.1 Effektmål

Effektmål beskriver hva oppdragsgiver tjener/oppnår/vinner på systemet vi skal utvikle. ETC ønsker at utviklingen av registrerings- og strømforbruk modulene skal bidra til:

- Mer fornøyde kunder/brukere av CarAdmin ved å spare tid og ressurser.
- Kunde-binding, hjelpe kunder med enda et problem slik at de velger CarAdmin overfor andre konkurrenter.
- Å gi kunden bedre oversikt over kostnader, som igjen gir mer fornøyde kunder.
- Et produkt som trekker til seg flere kunder ved å være attraktiv for elbilmarkedet.

1.2.2 Resultatmål

Resultatmål er hva oppdragsgiver får overlevert (programvare/teknisk/fysisk) som resultat av prosjektet. Systemet vi lager skal resultere i:

- Et system for automatisk registrering av ladebrikker hos ulike ladeselskaper gjennom CarAdmin sitt brukergrensesnitt.
- En samlet oversikt over strømforbruk i CarAdmin sitt brukergrensesnitt.
- En CarAdmin modul som kan ta i bruk ladestasjonselskapenes framtidige APIs.

I kapittel 10 diskuterer vi til hvilken grad målene ble oppnådd.

1.3 Målgruppe

Vi vil her skille mellom målgruppen for produktet og rapporten.

1.3.1 Målgruppe for produkt

Målgruppen for produktet er brukerne av CarAdmin. Dette gjelder ikke bare NRK som vi har hatt kontakt med gjennom prosjektet, men også alle de nåværende og framtidige kundene til ETC.

1.3.2 Målgruppe for rapport

Målgruppen for rapporten er hovedsakelig sensor som skal vurdere prosjektet, samt andre som er interessert i vår tematikk. Rapporten er skrevet med utgangspunkt i at leseren har en viss (data)teknisk bakgrunn, men ikke nødvendigvis kjenner til teknologier som blir brukt i dette prosjektet, eller har innsikt i fagområdet vi jobber med.

1.4 Bakgrunn og læring

I denne seksjonen skal vi skrive om forhåndskunnskapen gruppemedlemmene har, og læringsmålene for prosjektet.

1.4.1 Akademisk bakgrunn

Alle gruppemedlemmene går Bachelor i Programmering ved NTNU i Gjøvik, hvor vi har dekket et bredt fagfelt med mange ulike teknologier. Studieløpet dekker flere fagområder som er relevante for denne oppgaven. Ferdigheter som har vært viktige for prosjektet kommer blant annet fra emnene Cloud Technologies, Software Developement, Databasesystemer og databasemodulering, og Integrasjonsprosjekt. Gruppen har erfaring med Java og andre objektorienterte programmeringsspråk.

Gruppen manglet erfaring med enkelte teknologier i Java som f.eks Restlett, og Java-basert web utvikling. Vi hadde heller ikke noe erfaring med å programmere mot elbil-ladere, eller å utvikle i CarAdmin sin kodebase.

1.4.2 Læringsmål

Læringsmål er hva vi vil lære i denne prosessen. I tillegg til læringsmålene i emnebeskrivelsen til PROG2900[3] har vi definert følgende læringsmål:

- Lære om nye teknologier.
  - Java-basert web utvikling.
  - Databaser med MariaDB.
- Lære å utvikle en mikrotjeneste for et eksisterende system.
- Få mer kunnskap om teknologi rundt lading av elbil.
- Lære å utvikle et system som kommuniserer med ulike ladestasjoner.
- Lære mer om sikkerhet rundt behandling av sensitiv informasjon.
- Lære mer om utvikling av brukervennlige grensesnitt.
- Lære om samarbeid i prosjektgruppe.
- Lære å kontakte relevante personer i næringslivet for å undersøke problemer og muligheter.

1.5 Rammer

1.5.1 Praktiske rammer

- NTNU har satt en tidsfrist fra 10. Januar 2022 til 20. Mai 2022 for å ferdigstille arbeidet.
- COVID-19 tiltak fører til mindre mulighet for fysisk oppmøte første måneden av prosjektet.
- Må jobbe sammen med flere ladestasjon-selskaper for å få tilgang til APIs.
- Krav på tre statusrapporter til veileder gjennom arbeidet den 20. Februar, 15. Mars og 15. April.
- Vi jobber med NRK som testkunde gjennom oppgaven.
- Vi sitter på ETC sitt kontor onsdag hver uke.

[3]NTNU. *PROG2900 - Bacheloroppgave*. URL: https://www.ntnu.no/studier/emner/PROG2900#tab=omEmnet (sjekket 26. jan. 2022).

1.5.2 Tekniske rammer

- Oppdragsgivers rammer.
  - Java som programmeringsspråk.
  - Restlet som RESTful web API rammeverk.
  - GitLab som versjonskontrollsystem.
- Må integrere med ladestasjon bedrifter sine APIs eller andre tredjepartsløsninger.
- Løsningen skal fungere i en nettleser.
- Løsningen skal støtte alle ladebrikker.
- Løsningen skal utvikles som en modul av ETCs CarAdmin tjeneste.
- Produktet skal utvikles i den eksisterende kodebasen til CarAdmin.

1.6 Prosjektorganisering, ansvarsforhold og roller

Ansvarsforhold og roller er fordelt i henhold til ordinære scrum roller. Det vil si at roller er fordelt mellom produkteier, scrum-master og utviklere, i tillegg til rollen for veileder.

- **Produkteier:** Dag Solhaug
  - Beskrive hva som er viktig å levere innenfor utvikling av systemet.
  - Skal ha oversikt over produkt backlog.
  - Gi støtte ved tekniske behov for gruppen.
  - Gi tilbakemelding om gruppens arbeid underveis i planlegging og utviklingsfasen av prosjektet.
- **Scrum master:** Abdulhadi Al-Sayed
  - Skal ha oversikt hvordan gruppen ligger an i forhold til prosjektplanen.
  - Innkaller til møter.
  - Være bindeledd mellom utviklingsteam og produkteier.
  - I tillegg til alle punkt under "Utviklere".
- **Utviklere:** Elliot Sveum Torp, Rihards Daniels Ustinovics, Sindre Emil Vikre
  - Håndtere produkt backlog innenfor hver sprint.
  - Daglig scrum møte for å undersøke og tilpasse arbeid.
  - Bidra til å nå sprint mål.
- **Veileder:** Frode Haug
  - Hjelpe med å få gruppen i gang med rapport prosessen.
  - Bidra med konstruktiv kritikk til rapporten.

1.7 Om rapporten

1.7.1 Rapportens struktur

I vedleggene er det en oversikt over ord og akronymer vi har definert. Under er en oversikt over de ulike kapitlene i rapporten.

- Introduksjon gir en introduksjon av fagområdet, oppgaven, og gruppen.
- Teori går mer i dybden om problemstillingen og fagfeltet.
- Kravspesifikasjon beskriver kravene for programvaren.
- Brukergrensesnitt (GUI) omhandler det visuelle designet til programvaren.
- Teknologier forteller om teknologier vi har brukt for å utvikle systemet.
- Arkitektur gir overblikk over systemarkitekturen.
- Utviklingsprosess gjør rede for hvordan gruppen har arbeidet i prosjektperioden.
- Implementasjon beskriver hvordan vi har realisert programvaren.
- Testing og Kvalitetssikring gjør rede for hvordan vi har testet systemet.
- Avslutning oppsumerer, kritiserer og evaluerer arbeidet vi har gjort.

2 Teori

I dette kapitelet skal vi gå enda mer i dybden om fagfeltet, hvorfor ETC ønsker å gjennomføre dette prosjektet, motivasjonen vår for å velge denne oppgaven og kunnskap vi har tilegnet oss. Noe som vil gi leseren et grunnlag for å forstå de kommende kappitlene.

2.1 Fagfelt

2.1.1 Ladetyper

Vi skiller mellom ekstern lader og intern lader. Eksterne ladere er plassert ute i offentligheten, og alle som har en elbil med et ladepunkt som passer har muligheten til benytte seg av dem. For eksterne ladere er det mest vanlig å bruke en ladebrikke som betalingsmiddel. Eksempler på eksterne ladere er Eviny, Mer og Fortum sine ladere.

Interne ladere er private ladere som ofte blir hengt opp i private hjem, eller i garasjen til en virksomhet. Det er vanlig å kunne lade med en intern lader uten å bruke en ladebrikke, og bare plugge laderen rett i bilen uten ekstra steg. Eksemepler på interne ladere er Easee og Zaptec.

2.1.2 Mer om problemstillingen

Som nevnt i seksjon 1.1.1 om fagområde er elbilmarkedet i stor vekst. Det har ført til at mange nye aktører strømmer til, og flere etablerte selskaper tilpasser tjenestene sine for å gjøre krav på sin plass i markedet. Når flere bedrifter kommer med sin egen løsning til et marked som vokser raskt er det lett for å ende opp med mange ulike løsninger som ikke følger en felles standard. Det er utviklingen man ser i elbilmarkedet. Et konkret eksempel er bruken av en ladebrikke som betalingsmiddel for en ekstern lader.

For å lade hos et gitt ladeselskap må man opprette en bruker i den tilhørende applikasjonen eller nettsiden, for å så legge til betalingsinformasjon og RFID-nummeret til ladebrikken. Den samme prosessen må gjentas hos alle de ulike selskapene man ønsker å lade hos. I tilleg er det komplisert å holde oversikt over strømforbruket når man lader med en ekstern lader eller intern lader fra flere ulike leverandører, ettersom man selv må hente ut og legge sammen det totale strømforbruket.

Problemene beskrevet over er ikke en stor sak for enkeltmennesker som eier en elbil, men for bedrifter med opptil flere hundre biler blir det en stor jobb. Vi har hatt møter med Pål Brynhildsen som er bilansvarlig i NRK. Han tok oss gjennom prosessen han må følge for hver ladebrikke han registrerer. Det er uten tvil en repetitiv og tungvint måte å gjøre det på.

ETC merker nå stor etterspørsel fra NRK og andre kunder om en oppfølging for

elbiler, tilsvarende det CarAdmin tilbyr for bensin og diesel biler. Det er naturligvis fordi kundene til ETC er i ferd med å bytte ut fossildrevne biler med elbiler.

2.2 Hvorfor dette emnet?

2.2.1 Motivasjon

Hva var motivasjonen vår for å velge denne oppgaven? Vi fikk et godt inntrykk av ETC og de ansatte der da de presenterte oppgaven for oss på campus. I tillegg virket oppgaven både spennende, og utfordrende. Alle på gruppen hadde interesse i å utvikle programvare som kommuniserer med fysisk hardware som blir brukt av andre. Da passet elbilladere og RFID-brikker veldig bra. Etter en samtale med oppdragsgiver forstod vi raskt at de er ledende innen kjøretøyoversikt, og med dette prosjektet vil de være blant de første som tilbyr en slik oppfølging for elbiler. Å være med på å skape en framtidsrettet løsning som ikke finnes fra før skapte ekstra motivasjon for å velge denne oppgaven.

2.3 Undersøkelse av mulige teknologier

Da vi valgte oppgaven var det ikke bestemt hvordan den skulle løses, og om det var mulig å gjøre det oppdragsgiver ønsket. En stor del av arbeidet har derfor gått til å undersøke hvilke muligheter som finnes, og til å kontakte ladeselskaper for å høre hvilke teknologier de har tilgjengelig som kunne hjulpet oss med å løse oppgaven. Gjennom kartleggingen vår har vi gjort mange interessante funn om hvilke teknologier som finnes, og hva som er mulig å gjøre innen fagfeltet. I tillegg lært mye nytt om elbilladere og ulike standarder ladere følger.

Gruppen begynte tidlig med å undersøke mulige teknologier og metoder. Vi så for oss at markedet for fagområdet er i veldig sterk vekst ettersom populariteten for elektriske kjøretøy vokser internasjonalt, og Norges økonomiske satsing på markedet har tilbrakt en synlig endring i samfunnet[4].

For å hente strømforbruk og registrere ladebrikker trengte vi en metode for å kommunisere med de ulike ladeselskapene sine løsninger. Det var to alternativer vi så for oss: HTML-parsing, eller APIer. Det sistnevnte alternativet var mest ønskelig for oss, og vi begynte å undersøke hvilke APIer som er offentlig tilgjengelige. Vi fant ut at interne ladere pleier å ha en API, men eksterne ladere er mye mer lukket. Ladeselskapene vi var interessert i var Eviny, Mer, og Fortum. Ingen av de selskapene hadde noe informasjon om en API som var tilgjengelig for oss, men vi tenkte at det er stor sjanse for at de har en API de bruker til utvikling innad i bedriften. Derfor tok vi kontakt med ladeselskapene før å undersøke hvilke teknologier de har tilgjengelig for utviklere.

[4] elbilforening, *Norwegian EV market*.

2.3.1 Kontakt med ladestasjonselskap

Vi tok kontakt med ulike ledeselskaper som leverer eksterne ladere fordi vi ønsket å få tilgang til en API som kan kommunisere med laderen. Vi så for oss at dersom selskapene hadde en slik API ville det være stor sjanse for at vi ville få tilgang ettersom vi samarbeider med ETC og NRK. I tillegg til at det potensielt vil føre til flere kunder for selskapet med de eksterne laderne.

For de relevante bedriftene prøvde vi å komme i kontakt med fagpersoner eller ledere gjennom å sende mailer og ringe til bedriften. I et enkelt tilfelle tok vi og kontakt gjennom LinkedIn. Å kontakte bedriftene var en lang prosess med mange avslag og mye motgang. Vi begynte med å sende mailer, men etterhvert skjønte vi at det tar lang tid å få tilbake svar, og vi måtte forsøke noe annet. Vi startet å ringe bedriftene sin kundeservis og håpte at noen der kunne sette oss over til enten en utvikler eller en leder i bedriften.

Etter en lang prosess fikk vi avslag fra alle bedriftene utenom Eviny. Hos Eviny fikk vi telefonnummeret til en som er seksjonssjef for hurtiglading. Han var villig til å hjelpe, og tok saken videre. Når det hadde gått noen uker uten mer svar etter flere mailer og telefonsamtaler måtte vi konkludere med at vi ikke får tilgang til en API til noen eksterne ladere.

Gjennom denne undersøkelsen av teknologier har vi fått god og verdifull innsikt i hvordan det er å jobbe med andre aktører i industrien.

2.3.2 API til interne ladere

Selvom vi ikke fikk tilgang til APIer hos eksterne ladere fant vi en API vi skal bruke for å hente strømforbruk fra interne ladere. Vi skriver mer om denne APIen i seksjon 5.3.

2.3.3 Eksisterende ladebrikketeknologier

Vi skal gå inn i de eksisterende ladebrikketeknologiene som i dag er mest utbredt i markedet, og forholdet mellom selskapene som bruker de. Elbilforeningen er en av de største leverandørene av ladebrikker, og brikkene virker på nesten alle norske ladere. Det er og mulig å kjøpe ladebrikker direkte fra andra ladeselskaper, men registreringsmulighetene kan være begrenset til ladeselskapet man kjøpte brikken hos.

Noen selskaper som Circle K eller E.ON tar imot alle formater av ladebrikker, mer spesifikt serie-nummer formatet disse brikkene deler med ladestasjonen. Andre ladestasjoner som Eviny og Fortum lager sine egne formater og aksepterer derfor ikke alle typer ladebrikker. Dette er grunnet sikkerhetstiltak som ligger i grunnen for NFC teknologien som brukes i ladebrikker, ettersom det er lett å duplisere ladebrikke signatur eller overskrive innholdet til brikken.[5]

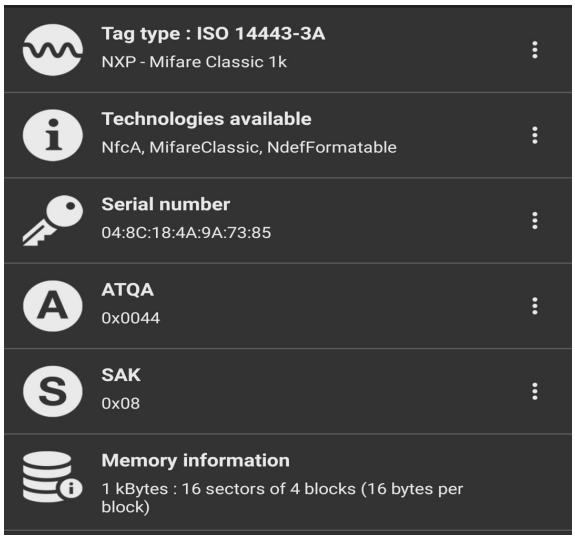
[5]Pierluigi Paganini. *Near field communication (NFC) technology, vulnerabilities and principal*

2.3.4 Standarder

ISO-14443
RFID ladebrikker er den mest utbredte standarden for brukerautorisering ved ladestasjoner. Mer spesifikt er ISO 14443 MiFare standarden mest utbredt i Norge for ladebrikker etter vår observasjon.[6]

MiFare standarden var viktig å få innsikt i for å ha oversikt av hvilke data som deles gjennom ladebrikker, ettersom et system skal kunne bygges rundt det for å registrere ladebrikkene og samle inn strømforbruk.

I implementasjonen vår ender vi opp med å innhente serienummeret lagret i en ladebrikke, enten ved å lese og skrive den inn selv, eller ved å bruke en ekstern NFC-leser. Et eksempel på en ekstern NFC-leser er mobiltelefoner som har denne teknologien inkludert, eller dedikerte maskiner som blir oppkoblet mot datamaskinen for å lese og skrive inn serienummer i markerte skrivefelt.



Figur 1: Data lagret i en Circle K ladebrikke innhentet ved å bruke NFC scanner på en mobiltelefon

ISO-15118
Denne standarden inkluderes ettersom den bygger et fundament for fremtidige teknologier. Der intuitive ideer innen databehandling og energisparing av elkjøretøy, kan innlemmes i standarden for å koble kjøretøy i det elektriske nettet.

ISO-15118 standarden skisserer teknologien for å gi et kjøretøy tilgang til det smarte strømnettet.[7] Dokumentet viser til den digitale protokollen som et elektrisk kjøretøy bør ta i bruk for å identifisere seg og dele informasjon med ladenettverket.[8]

attack schema. URL: https://resources.infosecinstitute.com/topic/near-field-communication-nfc-technology-vulnerabilities-and-principal-attack-schema/ (sjekket 27. apr. 2022).

[6]ISO (the International Organization for Standardization). *Cards and security devices for personal identification*. URL: https://www.iso.org/standard/76566.html (sjekket 27. apr. 2022).

[7]U.S. Department of Energy. *The Smart Grid*. URL: https://www.smartgrid.gov/the_smart_grid/smart_grid.html (sjekket 27. apr. 2022).

[8]Marc Mültin. *What is ISO 15118?* URL: https://www.switch-ev.com/knowledgebase/what-is-

2.3.5 Fremtidige teknologier

Våre observasjoner er at nyere standarder som ISO-15118 har en høy adopsjonstrend, der stadig flere bilprodusenter tar i bruk teknologi som følger denne standarden til deres elektriske og hybride modeller. Standarden har større potensiale en den nåværende NFC standarden som tar i bruk ladebrikker. Dette er grunnet økt sikkerhet, i motsetning til ladebrikker som kan ødelegges eller kopieres. Identifikasjon av bil for autentisering vil kunne lett bli en global standard ettersom biler allerede har en global identifikasjonsnummer, nemlig VIN. Standarden kan gi muligheten for å samle sanntidsdata for å tilby bedre sikkerhet og funksjoner, for flåtestyring hos bedriftskunder og kommuner. Dette blir mulig siden sanntidsdata vil svare på hvem, hva, hvor og når spørsmål når kjøretøy lader.

Fremtidige teknologier innenfor infrastruktur for lading, er ettertraktet spesielt av bedrifter som NRK. Der innsats for tilsyn av bilflåteadministrasjon kan eksponensielt øke. Dette kan også være verdifult for kommuner, ettersom ryddig og analyserbar data er ettertraktet, og mer så når administrasjonsverktøy for kjøretøy som CarAdmin integrerer sanntidsdata. Det er en spennende bestrebelse for fremtidig forskning, men dessverre så er nåværende adopsjon av ISO-15118 standarden i markedet for lite, og spesielt i kjøretøy. Vi så for oss at det å utføre oppgaven med MiFare standarden som fokus ville gi oss et helhetlig bilde av et produkt som i fremtiden kan adoptere nyere teknologier slik som ISO-15118 standarden.

2.4 Opplæring i CarAdmin kodebasen

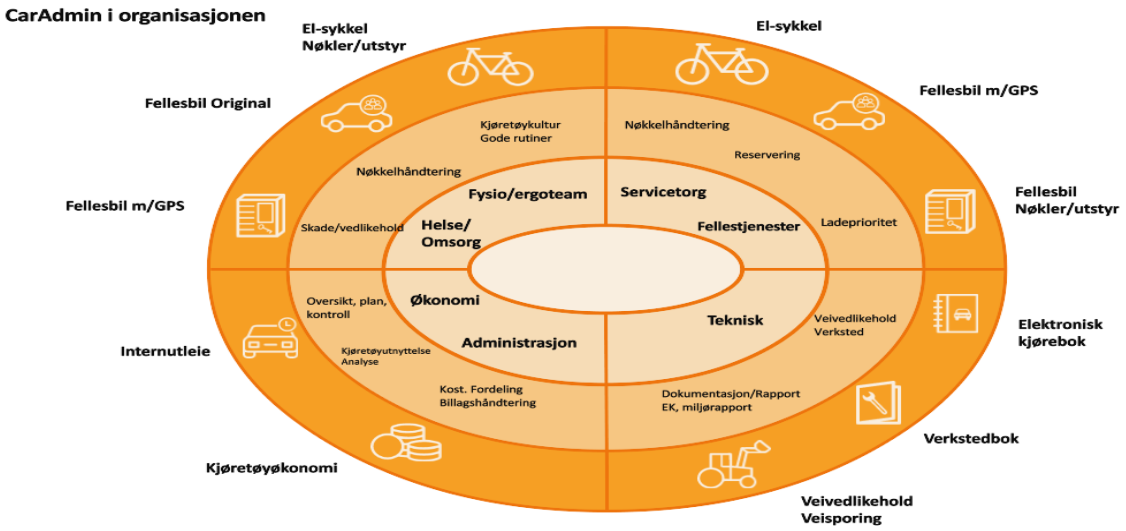
For å bli kjent med ETC og CarAdmin fikk gruppen en presentasjon av produktet og veileding av utviklerne hos ETC. Målet var å fordype forståelsen av selve kodebasen og forstå hvorfor CarAdmin er en attraktivt tjeneste. Gruppen var ferdig med prosjektplanen på dette tidspunktet, og hadde hatt tilgang til kodebasen en stund. Vi brukte tid på å finne ut hvordan systemet settes opp, og på å forstå konseptet bak Restlet. Presentasjonen og opplæringsseminaren fra ETC styrket kunnskapen vår om det vi brukte tid på å lære oss selv.

2.4.1 Presentasjon av ETC

Innholdet i presentasjonen var ment for å gjennomgå CarAdmin sin rolle innenfor kjøretøyoppfølging og administrasjon. Den var holdt av Randi Braastad (Ettermarkedsleder) og Tom Atle Moe (Support og ettermarkedsansvarlig).

Vi gikk gjennom utfordringen som sprang opp behovet for en løsning som CarAdmin. Dette inkluderte utfordringen kommuner hadde med å låne ut biler til ansatte, fordeling og organisering av nøkler, og det mest relevante for oss som var hvordan ladebrikker burde håndteres.

iso-15118 (sjekket 27. apr. 2022).



Figur 2: Oversikt over moduler i CarAdmin

Poenget med presentasjonen var å få en generell oversikt over omfanget til CarAdmin sine moduler og andre tjenester. Etter presentasjonen hadde gruppen en grei oversikt over kodebasens innhold, men fortsatt for lite kunnskap til å kunne sette i gang utviklingen.

2.4.2 Veiledning

Under seminaret, som ble holdt av Sebastian Balsam (Utvikler), ble vi opplært i prosessen for å sette opp CarAdmin sitt utviklingsmiljøet med Docker. Vi ble også gitt veiledning i metodikken for å sette opp en egen side i CarAdmin og dataflyten fra sideoppsett til databasekommunikasjon gjennom MVC modellen og kommunikasjonsmetoden AJAX. Etterhvert så ble gruppen enige med oppdragsgiver om å møte Onsdag hver uke for å få raskere veiledning i tillegg til å bli kjent bedre med utviklingsmiljøet.

2.5 HTML-Parsing

Siden vi ikke fikk tilgang på noen APIer måtte vi se på andre muligheter. Det første vi vurderte var å bruke HTML-parsing til å hente ut, registrere og slette ladebrikker. Det som gjør HTML-parsing til en attraktiv mulighet er at man kan hente ut hele nettsider med all HTML og informasjon, for deretter å plukke ut den informasjonen man vil ha. Det er flere muligheter med HTML-parsing, og du er ikke avhengig av tilgjengelige API-endepunkt. Ulempen er at det kreves mye manuelt arbeid, hvor man må lete gjennom nettsider for å finne ut hvor informasjonen man vil uthente ligger, og hvordan man skal hente den ut. En annen ulempe er at koden som benytter HTML-parsing krever mye vedlikehold, og kan i verste fall bli utdatert og verdiløs dersom nettsiden endrer seg.

På grunn av alle ulempene med HTML-parsing ønsket vi ikke bruke det med mindre

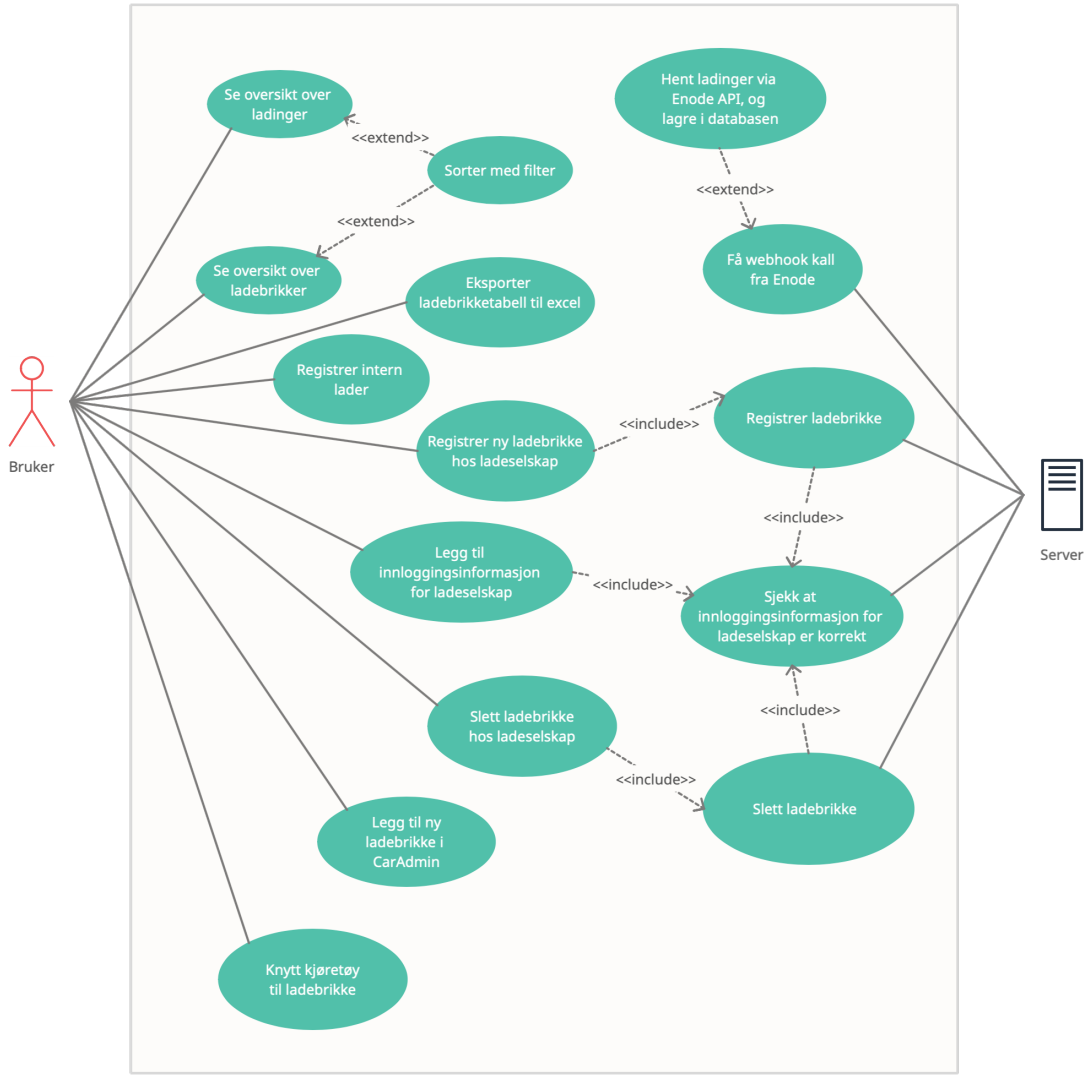
det ble absolutt nødvendig. Vi begynte derfor å grave og undersøke i nettsidene til Eviny og EON etter APIer som blir brukt i bakgrunnen. Ved å inspisere siden og bruke nettverk tabellen i Google Chrome, fant vi hvilke endepunkt vi kunne bruke til å løse ladebrikkeregisteringsdelen av oppgaven. Det er skrevet mer om denne prosessen i 8.4.1.

3 Kravspesifikasjon

Gjennom møter med oppdragsgiver og undersøking av mulighether har vi utformet kravene til programvaren vi skal utvikle. Kravspesifikasjonen har endret seg i løpet av utviklingsprosessen ettersom vi har gjort nye oppdagelser eller fått ønsker fra oppdragsgiver. I dette kapittelet presenterer vi de endelige kravene vi kom fram til.

3.1 Use case diagram

Vi bruker use cases for å beskrive de funksjonelle kravene til systemet.



Figur 3: Use case diagram

3.1.1 Høyniva Use case

Navn: Lading oversikt **Aktør:** Bilansvarlig **Mål:** Se en oversikt over registrerte ladinger. **Beskrivelse:** Bruker går inn på siden for ladinger og ser en oversikt over alle ladingene i en tabell.

Tabell 1: Use case: Lading oversikt

Navn: Ladebrikke oversikt **Aktør:** Bilansvarlig **Mål:** Se en oversikt over ladinger. **Beskrivelse:** Bruker går inn på siden for ladebrikker og ser en oversikt over alle ladebrikkene i en tabell.

Tabell 2: Use case: Ladebrikke oversikt

Navn: Eksporter ladebrikke-tabell **Aktør:** Bilansvarlig **Mål:** Eksportere tabellen med ladebrikker til excel. **Beskrivelse:** Bruker skal kunne trykke på en knapp og få lastet ned et excel dokument med informasjonen om registrerte ladebrikker.

Tabell 3: Use case: Eksporter ladebrikke-tabell

Navn: Registrer intern lader **Aktør:** Bilansvarlig **Mål:** Registrere ny intern lader **Beskrivelse:** Får å hente strømforbruket til en intern lader må brukeren registrere laderen hos Enode som er APIet vi får ladinger fra.

Tabell 4: Use case: Registrer intern lader

Navn: Innloggingsinformasjon til ladeselskap **Aktør:** Bilansvarlig **Mål:** Legge til innloggingsinformasjon for ladeselskap **Beskrivelse:** For at brukeren skal kunne registrere en ladebrikke hos et ladeselskap gjennom CarAdmin trenger man innloggingsinformasjonen til ladeselskapet.

Tabell 5: Use case: Innloggingsinformasjon til ladeselskap

Navn: Ny ladebrikke **Aktør:** Bilansvarlig **Mål:** Legg til ny ladebrikke. **Beskrivelse:** Brukeren skal kunne legge til en ny ladebrikke som blir lagret i CarAdmin. Brikken er ikke registrert hos et ladeselskap ennå.

Tabell 6: Use case: Ny ladebrikke

Navn: Registrer hos ladeselskap **Aktør:** Bilansvarlig **Mål:** Registrere en ny ladebrikke hos et ladeselskap. **Beskrivelse:** Etter brukeren har registrert innloggingsinformasjon til et ladeselskap kan brukeren registrere en ladebrikke hos det eller de selskapene.

Tabell 7: Use case:

Navn: Knytt kjøretøy **Aktør:** Bilansvarlig **Mål:** Skape en kobling mellom en ladebrikke og en bil. **Beskrivelse:** Brukeren kan koble en bil mot en ladebrikke.

Tabell 8: Use case: Knytt kjøretøy

Navn: Slett ladebrikke **Aktør:** Bilansvarlig **Mål:** Slette en ladebrikke. **Beskrivelse:** Brukeren skal kunne velge en ladebrikke å slette. Da blir den slettet fra CarAdmin og ladeselskapene den er registrert hos.

Tabell 9: Use case: Slett ladebrikke

3.1.2 Lavnivå Use case

Navn: Hent ladinger
Aktør: Server
Mål: Hente ladinger gjennom Enode API
Beskrivelse: Når serveren får en webhook notifikasjon fra Enode henter den nye ladinger hvis det er aktuelt.
Forutsetning: En intern lader er registrert hos Enode.
Resultat: En ny lading vises i CarAdmin.
Normal flyt:

1. Server får en webhook notifikasjon fra Enode.
2. Server verifiserer at kallet er fra Enode.
3. Server sjekker om notifikasjonen inneholder en hendelse som sier at en ny lading er registrert.
4. Server henter den nye ladingen via Enode API.
5. Server lagrer ladingen i kunden sin database.

Alternativ flyt:
Normal flyt til og med punkt 1:

2. Server kan ikke verifisere kallet, og prosessen avsluttes.

Normal flyt til og med punkt 2:

3. Ingen relevant hendelse funnet, og prosessen avsluttes.

Feilsituasjoner:

- Klarer ikke å hente ladinger med API fra Enode.
- Får ikke kontakt med kunden sin database.

Tabell 10: Lavnivå Use case: Hent ladinger

Navn: Registrer ladebrikke
Aktør: Bilansvarlig
Mål: Registrere en ny ladebrikke hos ladeselskap.
Beskrivelse: Brukeren registrerer en eller flere ladebrikker hos de ladeselskapene brukeren ønsker
Forutsetning: Innloggingsinformasjonen til ladeselskapet er lagt inn hos CarAdmin.
Resultat: Ladebrikker er registrert hos ladeselskap.
Normal flyt:

1. Bruker velger en eller flere ladebrikker som skal registreres.
2. Et dialogvindu med oversikt over ladeselskapene brukeren kan velge mellom blir vist.
3. Bruker venter mens systemet sjekker at minst ett av ladeselskapene har korrekt innloggingsinformasjon lagt inn.
4. Bruker velger hvilke ladeselskap brikkene skal registreres hos.
5. Bruker trykker på registrer knappen for å starte prosessen.
6. Systemet viser en fremdriftsindikator for hvor mange ladebrikker som er registrert.
7. Bruker får beskjed om at alle ladebrikkene har blitt registrert.

Alternativ flyt:
Normal flyt til og med punkt 2:

3. Innloggingsinformasjonen til alle de innlagte ladeselskapene er feil eller mangler.
4. Bruker får ikke registrere ladebrikker før korrekt innloggingsinformasjon er lagt til.

Normal flyt til og med punkt 5:

6. Noen ladebrikker ble ikke registrert, og bruker får en rapport som forteller hvilke brikker som ikke ble registrert.

Feilsituasjoner:

- Systemet klarer ikke registrere ladebrikker hos de ulike ladeselskapene.

Tabell 11: Lavnivå Use case: Registrer ladebrikker

Navn: Legg til innloggingsinformasjon
Aktør: Bilansvarlig
Mål: Legge til innloggingsinformasjon for ladeselskap.
Beskrivelse: Bilansvarlig legger til innloggingsinformasjon for ladeselskap.
Forutsetning: Bilansvarlig har en konto hos ladeselskapene.
Resultat: Innloggingsinformasjon til ladeselskap er registrert hos CarAdmin.
Normal flyt:

1. Bruker velger å legge til et nytt ladeselskap, og får opp et dialog vindu.
2. Bruker velger et ladeselskap fra en nedtrekksmeny i dialogvinduet.
3. Bruker fyller inn brukernavn og passord for ladeselskapet, og trykker lagre. Dialogvinduet lukkes.
4. Systemet sjekker at innloggingsinformasjonen som ble oppgitt er korrekt.
5. Ladeselskapet vises i en liste med lagrede ladeselskaper, sammen med brukernavn som ble skrevet inn, og en grønn tommel opp som forteller at informasjonen er korrekt.

Alternativ flyt:
Normal flyt til og med punkt 2:

3. Brukeren trykker lagre uten å skrive inn innloggingsinformasjon.
4. Ladeselskapet vises i listen sammen med en oransje varseltrekant som forteller at innloggingsinformasjon mangler.

Normal flyt til og med punkt 4:

5. Systemet finner ut at innloggingsinformasjonen er ugyldig.
6. Ladeselskapet vises i en liste med lagrede ladeselskaper, sammen med brukernavn som ble skrevet inn, og et rødt kryss som forteller at informasjonen er ugyldig.

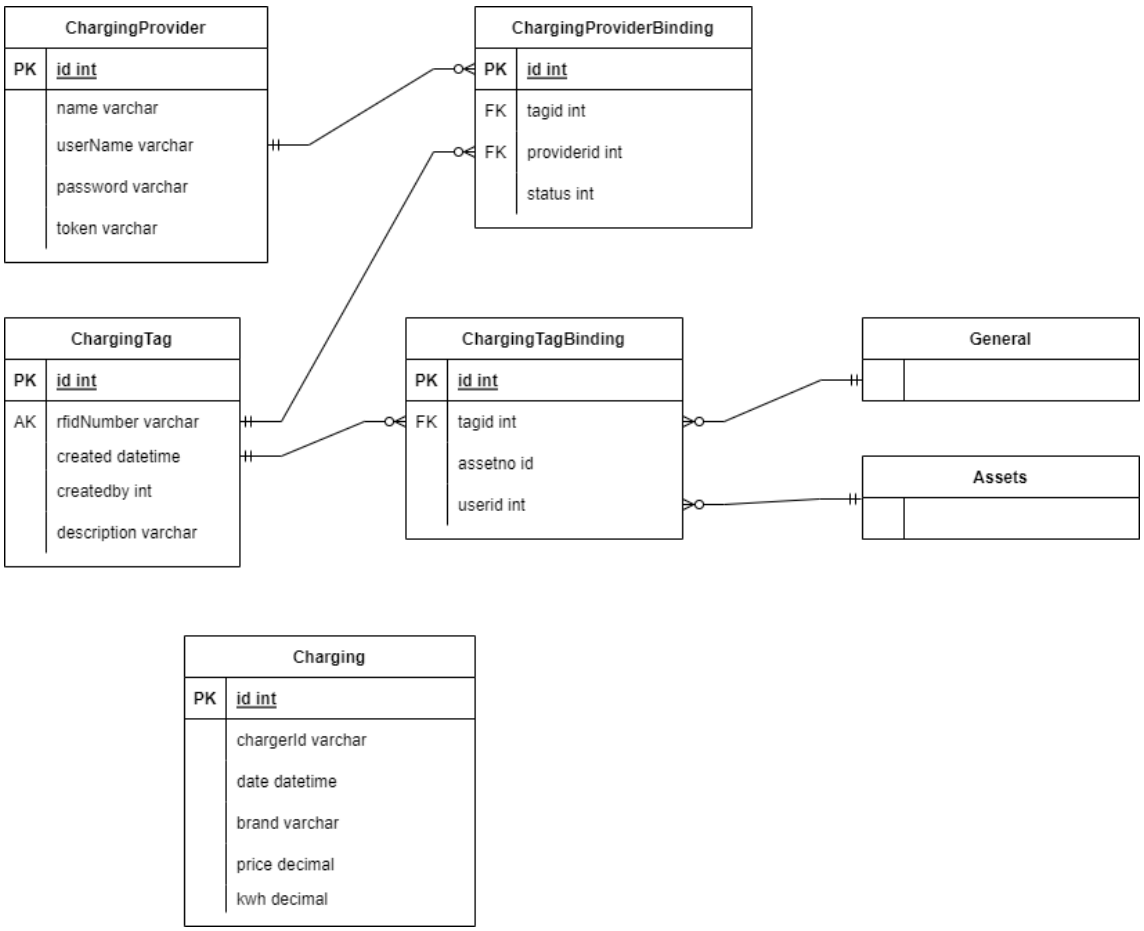
Feilsituasjoner:

- Systemet klarer ikke å vurdere om innloggingsinformasjonen er korrekt eller ikke.

Tabell 12: Lavnivå Use case: Legg til innloggingsinformasjon

3.2 Domenemodell

Domenemodellen beskriver strukturen på databasetabellene vi har utviklet. Oppdragsgiver ønsket at vi ikke viste strukturen og innholdet på deres database, og vi får dermed ikke se tabellene våres i konteksten av hele databasesystemet.



Figur 4: Domenemodell av databasetabeller

3.3 Ikke-funksjonelle krav

Her beskrives spesifikke kriterier vi setter overfor modulens implementasjon. Kravene tilegnes kvaliteter vi ønsker å ha under operasjon av ladebrikkeregistrering og strøminnsamlings modulene.

3.3.1 Brukergrensesnitt

- Vi skal følge CarAdmin sin designguide når vi utvikler brukergrensesnittet. Av ønske fra ETC har vi ikke lagt guiden med som vedlegg.

3.3.2 Operasjonelle krav

- Systemet skal tåle å registrere opp til 100 ladebrikker per ladeselskap.
- Systemet skal få inn en ny lading senest 10 minutter etter ladingen ble fullført.

3.3.3 Sikkerhet

- Alle passord skal bli hashet før det blir lagret i databasen.
- Innkommende API kall skal bli verifisert.

3.3.4 Videre utvikling

- Systemet skal bli utviklet slik at man lett kan legge til flere APIer for ladeselskap i senere tid.

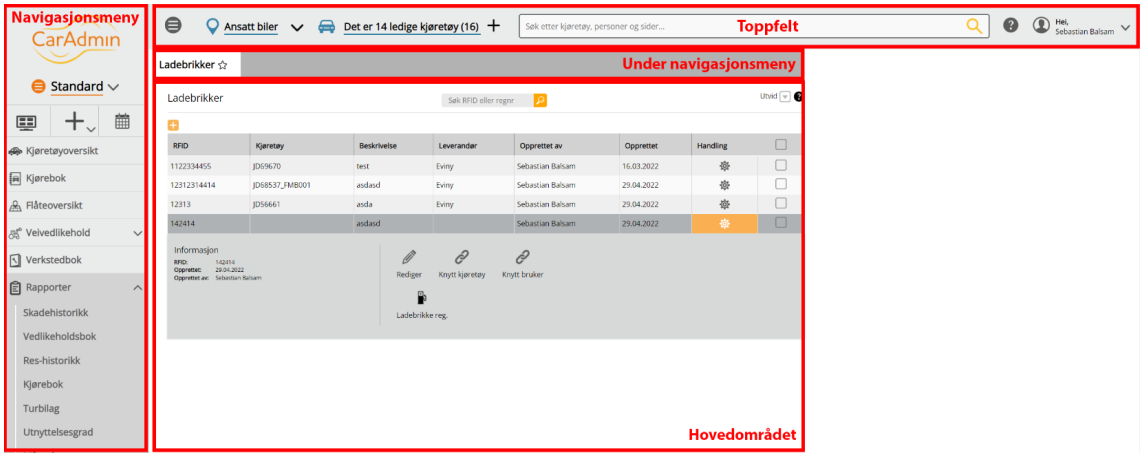
4 Brukergrensesnitt (GUI)

I dette kapitlet skal vi se nærmere på UI designet for modulene våre. Siden modulene våres er utviklet i CarAdmin kodebasen skal vi ikke utforme vårt egen design, derimot er vi nødt til å følge CarAdmin sin designguide. Oppdragsgiver har vært veldig tydelig på hvordan de ønsker at designet skal se ut, slik at modulene våre integreres fint med CarAdmin

4.1 Oppsett

Helt grunnlegene består sidens oppsett av fire hovedelementer. Disse er toppfelt, navigasjonsmeny, undernavigasjonsmeny og hovedområdet. Figur 5 viser et eksempel på en side i CarAdmin, og en markering av de ulike områdene på nettsiden. Videre skal vi se litt nærmere på hva hver av disse områdene inneholder.

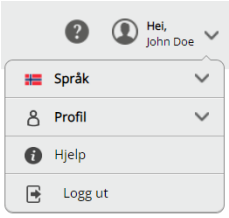
Det vi har utviklet på bildet er det man ser i hovedområdet og undernavigasjonsmenyen. Resten av det på bildet er utviklet av ETC.



Figur 5: Sideoppsett

4.1.1 Toppfelt

På venstre siden av toppfeltet finner vi en knapp for visning eller gjemming av navigasjonsmenyen. Videre har vi en nedtrekksmeny for valg av forskjellige grupper av kjøretøy, transporter og oppdrag, samt en liste med bilder som tilhører den valgte gruppen og deres status (ledig, låst, reservert eller opptatt). Videre finner vi også et søkefelt for enklere navigasjon rundt i systemet, hjelpeknapp som tar brukeren til CarAdmin sitt hjelpesenter og en til nedtrekksmeny vist i figur 6 som viser navnet til brukeren og inneholder forskjellige bruker innstilinger og handlinger.



Figur 6: Bruker meny

4.1.2 Navigasjonsmeny

Navigasjonsmenyen består av CarAdmin logo og en nedtrekksmeny for valg av navigasjonsmeny. Her kan brukeren bytte mellom standard eller egendefinerte navigasjonslister. I tilleg er det en meny som brukes til å navigere mellom sidene i CarAdmin.

4.1.3 Undernavigasjonsmeny

Dette feltet inneholder en liste underseksjoner for en side i CarAdmin. Brukeren har muligheten til å legge underseksjoner til i favoritter, markert med stjerne ikon på høyre siden av navnet, for enklere tillgang. I figur 5 har vi bare ett valg i undernavigasjonsmenyen, men har kan man ha flere sider.

4.1.4 Hovedområdet

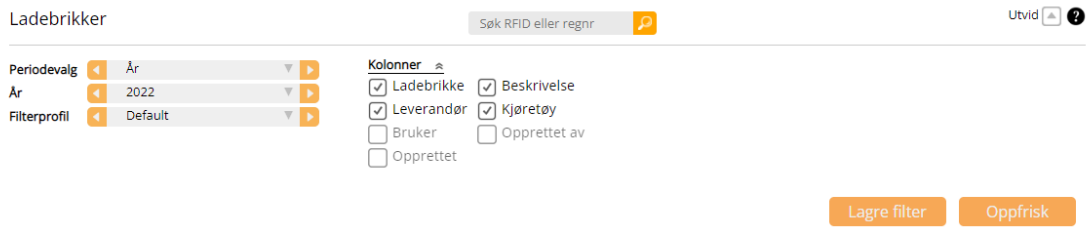
Her ligger selve inneholdet på siden.

4.2 Ladebrikker

I denne seksjonen skal vi se nærmere på ladebrikke siden. Vi skal se nærmere på de forskjellige delene og elementene for å gi oss en bedre forståelse rundt hvordan siden ser ut, er bygget opp og henger sammen.

4.2.1 Filtrering

Øverst i hovedområdet i figur 5 er det et søkefelt hvor brukeren kan skrive RFID-nummer for å søke etter en ladebrikke. Brukeren har muligheten for å utvide filtreringsfanen med å trykke på 'Utvid' knappen til høyre for søkefeltet. Under denne fanen får brukeren tilgang til et par nye funksjoner for visning og organisering av tabellen. I filtreringsfanen er det et datofilter hvor brukeren kan velge hvilke ladebrikker som skal vises, basert på når de ble registrert. Det går an å velge hvilke kolonner som skal vises i tabellen ved å markere avkrysningsboksene. Vi ser denne filtreringsfanen illustrert i figur 7. For at endringene til filteret skal vises i tabellen må brukeren trykke på oppfrisk knappen. Brukeren har også mulighet til å lagre valgte filter instilinger med 'Lagre filter' knappen.



Figur 7: Ladebrikke filter

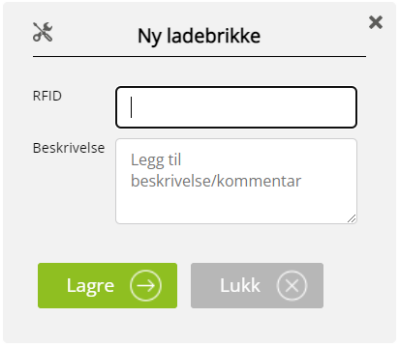
4.2.2 Detalj vindu

Ved å trykke på tannhjulet under handlinger i tabellen får vi opp detaljer for ladebrikker. På figur 8 ser vi hvordan detaljvinduet er fordelt i fire segmenter. Denne fordelingen blir generert automatisk avhengig av hvor mye og hva slags informasjon vi velger å vise. I de to midterste segmentene som viser tilknytninger har vi en liten søppelbøtte ikon ved siden av tilknytningene som gir oss muligheten til å fjerne tilknytningen. Siste segmentet er reservert for funksjonelle knapper som i dette tilfellet er rediger, knytt bruker, knytt kjøretøy og ladebrikke registrering. Siden ladebrikken i figur 8 allerede har en tilknytning til en bil, er knappen for å lage denne tilknytningen skjult.

Figur 8: Ladebrikke detaljer

4.2.3 Ny ladebrikke dialog

Dialogvinduet på figur 9 kommer opp når man trykker på pluss tegnet på venstre side over tabellen. Denne dialogen brukes til å legge til en ny ladebrikke og er et enkelt eksempel på hvordan dialogvinduene er designet i CarAdmin. Vi kan velger utsende og innholdet i dialogvinduet ved å sende inn parametere til taggen FieldFromFlex (mer om tagger i 8.2). Dialogvinduet består av et toppfelt som inneholder et ikon, navnet til dialogen og et kryss for å lukke dialogvinduet. Videre har vi det valgte inneholdet som i dette tilfellet er to tekstinnskrivningsfelt og to knapper. Her er det også viktig å legge merke til at lagre knappen er plassert på venstre side og følger standarden til CarAdmin.

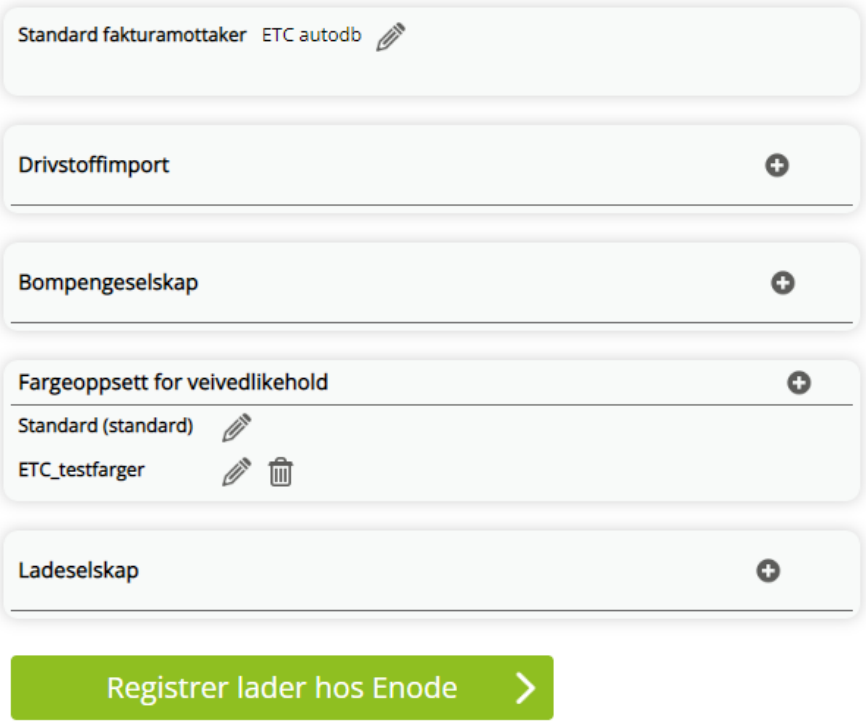


Figur 9: Ny ladebrikke dialog

4.3 Standardopsett

Standardoppsett siden inneholder diverse menyer for både utsende og administrasjon. På denne siden skal man blant annet kunne legge til forbindelser med drivstoff og ladeselskaper. Vår implementasjon i standardoppsett gir brukeren mulighet til å legge til innloggingsinformasjon til støttede ladeselskaper, samt registrere nye interne ladere hos Enode.

I figur 10 kan vi se de to nederste elementene som er en del av vår implementasjon. Ladeselskapelementet bruker samme taggen brukt i elementene overfor i samme figur. Hvis brukeren trykker på pluss tegnet til høyre, får de opp et dialogvindu hvor de da kan velge ønsket ladeselskap fra en nedtrekksmeny, som inneholder ladeselskaper støttet i implementasjonen vår. I tillegg inneholder dette dialogvinduet to inntastingsfelt, et for brukernavn og et for passord. Vi ser også på figuren at pluss tegnene ikke er på samme linje hos alle vinduene. Dette var noe vi diskuterte på et møte med ETC. Vi fant ut at det er en feil som ligger i ETC sin kode for plassering av knappene og ligger utenfor vår ansvarsområde. Av denne grunnen er dette noe vi ikke har fikset.



Figur 10: Standardopsett

Den grønne knappen nederst på figur 10 er også en del av vår implementasjon. Vi valgte å plassere knappen på bunnen av standardoppsett, siden den skiller seg fra de andre ellementene. I tillegg skiller denne knappen seg ut med at den ikke åpner et dialogvindu, men i stedet omdirigerer brukeren til Enode sin side hvor brukeren kan legge til flere interne ladere. Opprinnelig fantes knappen på siden med oversikt over ladinger. Vi snakker mer om dette og lignende endringer i kapittel 9.3.4.

4.4 Standardisering

Mye av tiden under utvikling ble brukt på standardisering av design og implementasjon. Målet med dette var at programvaren skal kunne sømløst integreres mot CarAdmin. Dette var en stor utfordring, men vi hadde tilgang til gode ressurser. Hovedsaklig brukte vi CarAdmin sin løsning som referanse for utviklingen i tillegg til en designguide levert til oss av ETC. Etter ETC sine ønsker kunne designguiden ikke bli lagt til som vedlegg for denne raporten. Vi har i tillegg fått gode innspill og tilbakemelding fra møter med ETC underveis i prosjektet. Vi snakker mer om noen av forandringene gjort mot slutten av utviklingsfasen i kapittelet som heter Demonstrasjon for ETC.

4.4.1 Ikoner

CarAdmin inneholder en stor samling av ikoner som brukes. Dette ga oss en god basis for standardisert utvikling. Siden lading og ladebrikke håndtering er nytt i CarAdmin var det mangel på ikoner som kunne brukes til disse funksjonene. Vi var nødt til å finne ikoner som vi brukte midlertidig under utviklingen. I et møte med ETC hvor vi presenterte framgangen i arbeidet fikk vi vite at det var mulig å legge inn en forespørsel til designeren hos ETC for å lage nye ikoner. Ikonene har foreløpig ikke blitt ferdig før innleveringen av oppgaven, og av den grunn må de bli lagt til på et senere tidspunkt. Under har vi lagtet en liste med ikoner som skal erstattes.

Figur 11: Ladebrikke tilknytning massefunksjon ikon

Figur 12: Ladebrikke tilknytning ikon

Figur 13: Feil ikon

Figur 14: Suksess ikon

Figur 15: Advarsel ikon

Figur 16: Statusbar for tilknytning av ladebrikker

5 Teknologier

Ettersom programvaren utvikles i den godt etablerte kodebasen til CarAdmin er en del teknologier som programmeringsspråk og rammeverk bestemt på forhånd. Vi vil derfor ikke argumentere for hvorfor disse teknologiene er valgt, men heller skrive om fordelene de medbringer.

Teknologiene bestemt av CarAdmin avgjør bare utviklingsmiljøet. En stor del av arbeidet gikk til å undersøke hva vi skulle bruke for å løse oppgaven.

5.1 Programmeringspråk

5.1.1 Java-basert webutvikling

CarAdmin er utviklet i Java, og dermed bruker vi det også. Alle medlemmene på gruppen har erfaring med Java fra tidligere emner, som er fordelaktig for oss siden vi slipper å lære et nytt programmeringspråk fra bunnen av.

Java er et populært språk å bruke for webutvikling, fordi det finnes mange verktøy man kan bruke for å utvide egenskapene til en webserver[9]. I dette prosjektet skal vi bruke Servlets sammen med Java Server Pages, med en Model View Controller arkitektur.

Servlet er en Java klasse som blir brukt for å lage dynamiske nettsider. Når klienten sender en forespørsel til webserveren, sender webserveren forespørselen videre til Servlet. Servlet tar imot forespørsler fra webserveren, prosesserer forespørselen, lager en respons, og sender responsen tilbake til webserveren. Webserveren sender så responsen til klienten, og det nye innholdet blir vist på skjermen.

JSP blir og brukt for å lage dynamiske nettsider. Det er en videreutvikling av Servlet. Med JSP kan en legge til biter med Java kode inn i HTML. Bitene med Java kode er det som gjør nettsiden dynamisk. Når en JSP side åpnes, blir den kompilert og gjøres om til Servlet kode. Videre er prosessen den samme som om man hadde brukt vanlig Servlet. Det går fint å ha litt Java kode i HTML, men for større systemer kan det fort bli komplekst og uoversiktlig. En beste praksis er å kombinere Servlets og JSP med Model View Controller arkitektur. I konteksten av vårt system er det Servlets som styrer logikken for hva som skal vises på skjermen, og JSP som bestemmer hvordan det skal se ut. Samtidig så defineres det AJAX funksjoner i JSP filen, der formålet er å kunne sende kall til webserveren uten å nødvendigvis endre elementer i nettsiden. Om noe skal endres i nettsiden så bestemmes det basert på resultatet fra AJAX kallet. Dette kan for eksempel være en knapp på siden som oppfrisker resultatene i en tabell.

[9] IBM. *JSP and Java servlet programming*. URL: https://www.ibm.com/docs/en/i/7.2?topic=java-jsp-servlet-programming (sjekket 6. mai 2022).

5.1.2 JavaScript, HTML, CSS

I nettapplikasjonen CarAdmin blir det brukt HTML, JavaScript og CSS. HTML brukes til å strukturere siden, CSS står bak designet, og JavaScript håndterer funksjonaliteten.

5.2 Database

CarAdmin bruker MariaDB[10] som er en relasjonsdatabase. Alle gruppemedlemene har god erfaring med SQL, som igjen betyr at gruppen ikke må brukte mye tid på å lære ett nytt språk.

5.3 Enode

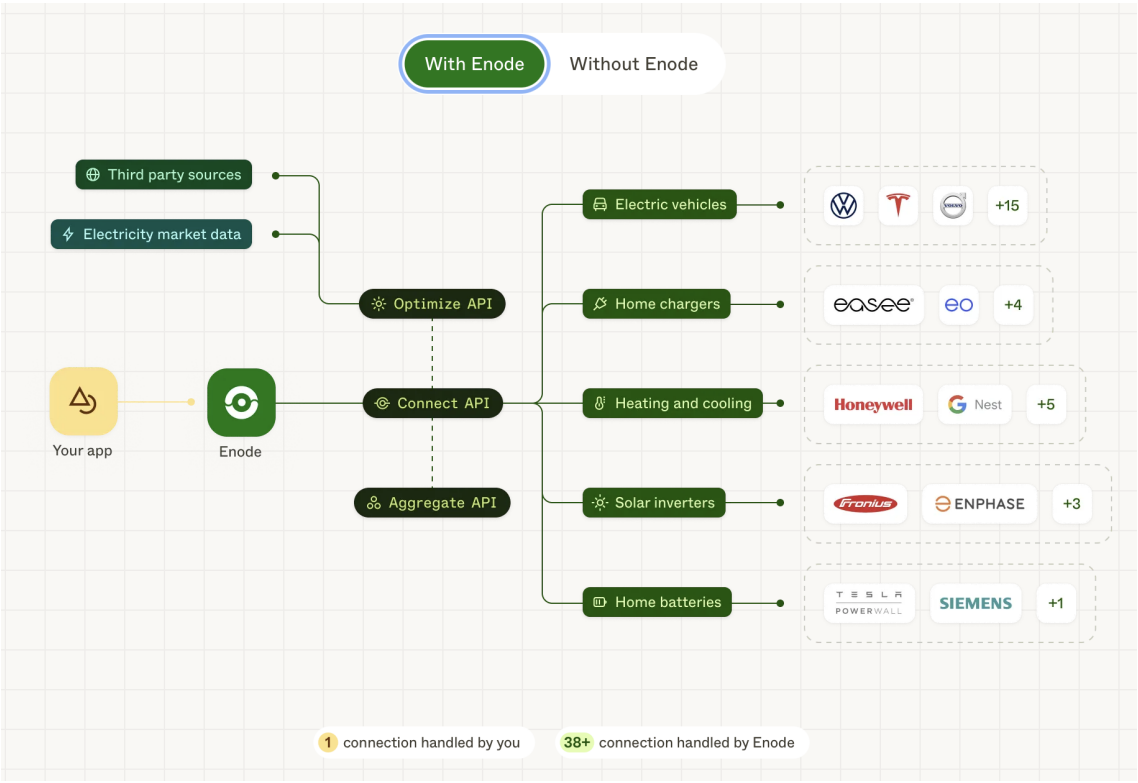
For å hente strømforbruket til en intern lader skal vi bruke Enode sin API[11]. Enode APIen samler APIene til flere ulike ladeselskap på en plass. Det gjør at vi som utviklere kun trenger å fokusere på en API-tilkobling, i stede for en tilkobling for hvert ladeselskap vi henter strømforbruk fra. Figur 17 og 18 er hentet fra nettsiden til Enode, og viser et eksempel på hvordan strukturen på applikasjonen kan se ut med og uten APIen.

Enode har tilkobling til mange ulike maskinvarer som elbiler, klimaanlegg, batterier til hjemmet, og solceller. Vi skal derimot kun benytte oss av endepunktene for hjemmeladere. Gjennom APIet er det mulig å registrere nye ladere, få informasjon om når en bil lader, starte og stoppe ladinger, få informasjon om pris på ladingen, og hvor mange kWh ladingen brukte.

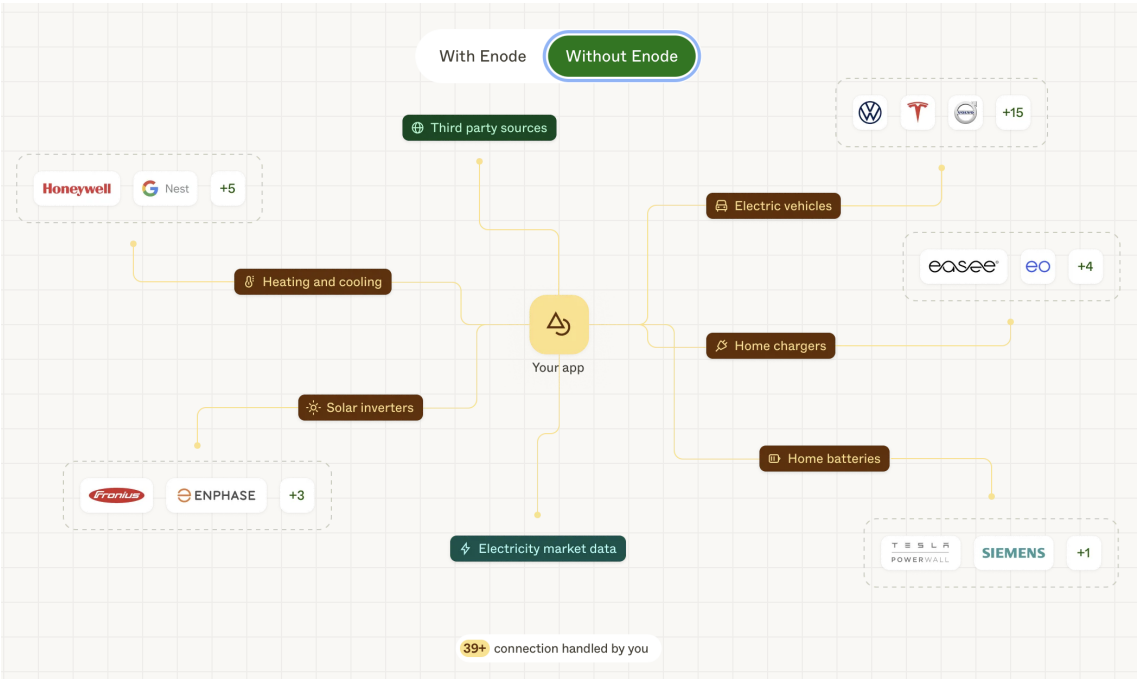
Det er flere fordeler med å bruke Enode. Som nevnt før gjør det at vi kun trenger å fokusere på én tilkobling, i stede for mange. Da får vi mer tid til å jobbe på andre deler av prosjektet. Vi får og muligheten til å hente strømforbruk fra flere hjemmeladere enn vi ellers ville, ettersom det vil være tidkrevende og i enkelte tilfeller ikke mulig å få tilgang til APIene deres. Drift av programvaren vil og bli enklere med Enode, siden man ikke trenger å overvåke mange ulike APIer og respondere dersom noe endrer seg på en av de. Ulempen med å bruke Enode er at vi blir avhengige av en tredjepartsløsning for at programvaren skal fungere. Derimot er fordelene så mange at vi har valgt å gå for den teknologien.

[10] MariaDB. *MariaDB Server: The open source relational database*. URL: https://mariadb.org (sjekket 7. mai 2022).

[11] Enode. *Enode Developer Hub*. URL: https://docs.enode.io (sjekket 15. feb. 2022).



Figur 17: Med Enode API



Figur 18: Uten Enode API

5.4 OkHttp og HttpClient

Vi bruker OkHttp og HttpClient for å gjøre API-kall. OkHttp er et effektiv http bibliotek for Java. God dokumentasjon gjør dem lett å bruke. HttpClient er inkludert i Java, som gjør at den er enkelt å ta i bruk. I starten av prosjektet brukte vi HttpClient, men etter vi oppdaget at Postman har en innebygd funksjon for å oversette curl http-kall til OkHttp byttet vi for å benytte oss av den funksjonaliteten.

5.5 Restlet

CarAdmin benytter seg av Restlet[12]. Restlet er et rammeverk for Java som hjelper med å bygge en RESTful API. Restlet har åpen kildekode, og er gratis å laste ned. Kraftige egenskaper for ruting og autentisering er noe som gjør rammeverket bra. I CarAdmin benytter vi de egenskapene ved å lage ulike routere for offentlige endepunkter, og lukkede endepunkter man må autentisere seg for å gjøre kall til.

[12]Restlet. *Restlet Framework*. URL: https://restlet.talend.com (sjekket 16. mai 2022).

6 Arkitektur

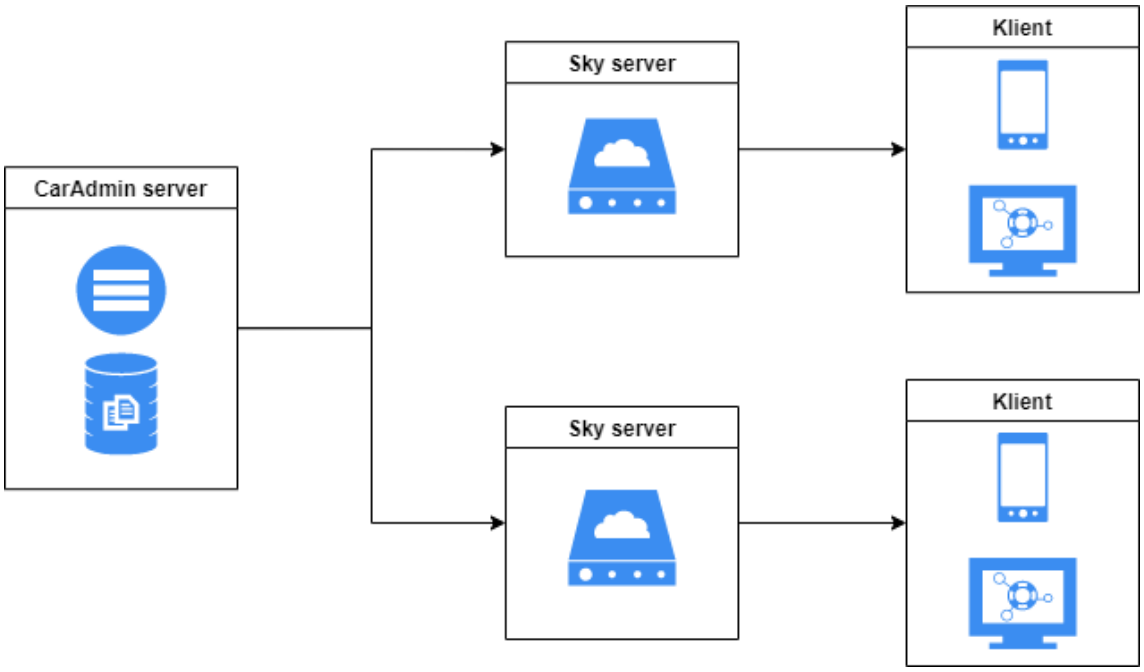
6.1 Arkitekturmodell

Kravspesifikasjonen for dette prosjektet krever ikke at vi tar beslutninger innenfor valg av arkitektur siden vi benytter oss av oppsettet til CarAdmin. Vi har valgt å skrive om det allikevel ettersom innholdet for en arkitekturmodell kommer til nytte for både utviklere av prosjektet og eksterne aktører som vil ha oversikt hvordan arkitekturer innen programvaren ser ut.

6.1.1 Server-nivå

CarAdmin benytter seg av klient-tjener modellen som vi ser i figur 19. I CarAdmin har hver kunde en egen skybasert server og database. Dette skaper ekstra sikkerhet ettersom hver kunde sin løsning er separert fra hverandre. Denne arkitekturen har flere fordeler som vi lister opp under.

- Denne modellen gjør reparasjon og vedlikehold enklere for utviklerene i ETC.
- ETC får flere potensielle kunder, ettersom kunder som ikke er interessert i å vedlikeholde egne servere.
- Analyse av ressursforbruk i skyen over lengre tid gir muligheten for å optimalisere fordeling av datakraft og sparing av serverkostnader basert på kundens egen forbruk.
- Det er enkelt for ETC å skalere opp når de får nye kunder.
- ETC øker sikkerheten i systemet ved å minske antall punkt som kan bli angrepet.



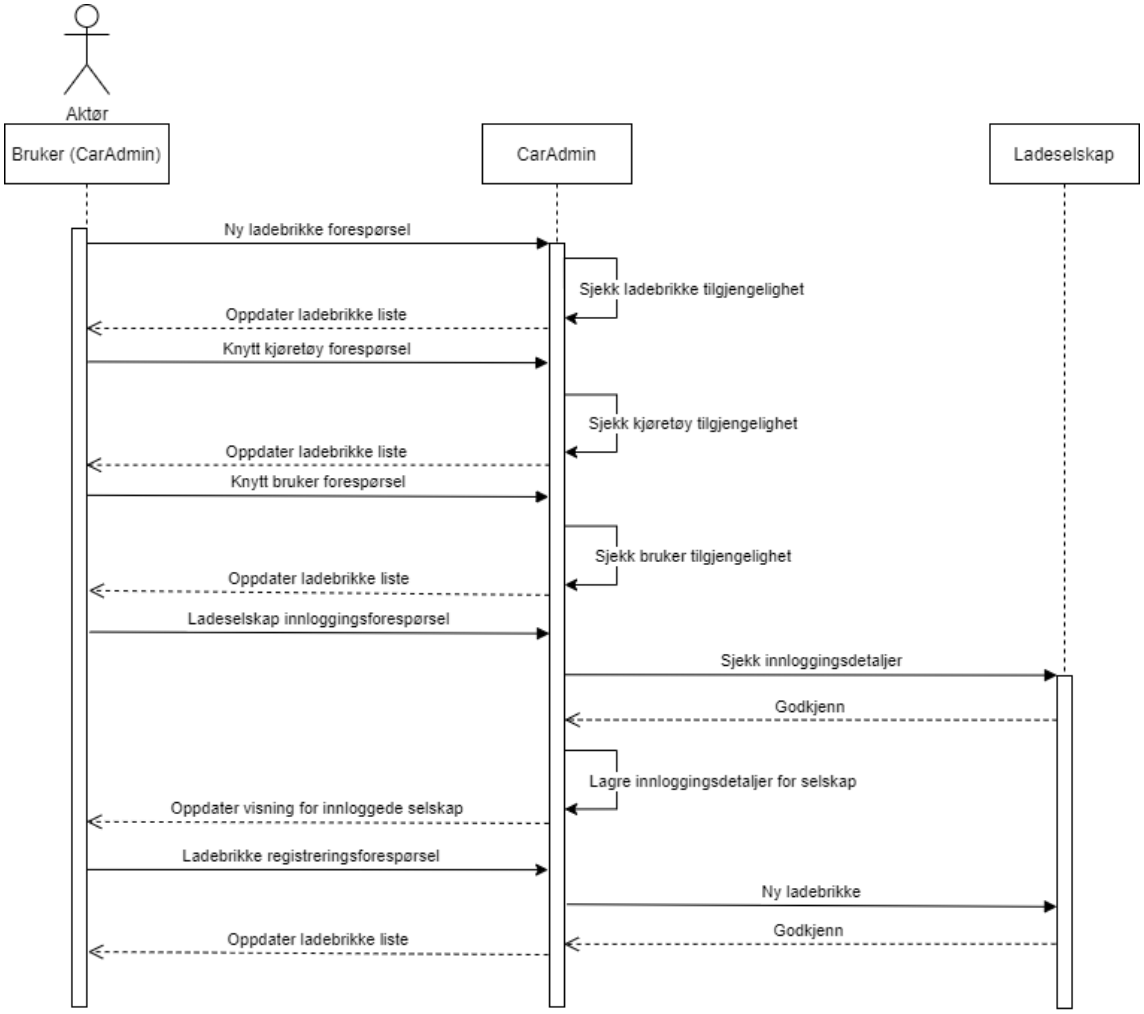
Figur 19: Eksempel av klient-tjener modell for CarAdmin

6.1.2 Programvare-nivå

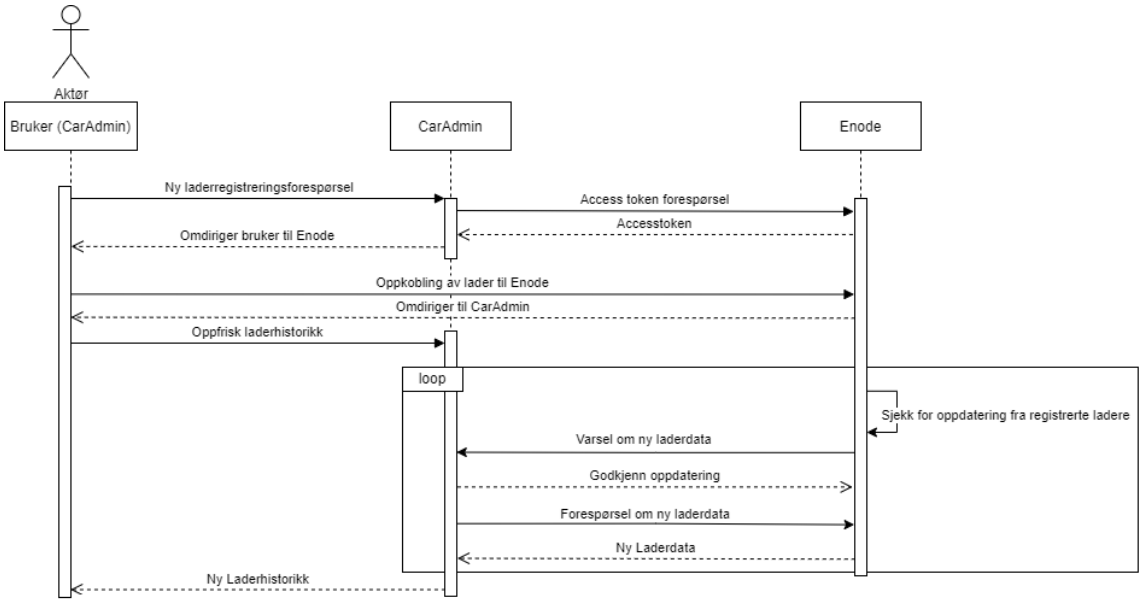
MVC er brukt i CarAdmin kodebasen. Modellen deler opp tjenesten i komponenter som behandler forretningslogikk, datalagring, sikkerhet, ruting og grafisk grensesnitt. Slike komponenter kan videreutvikles hver for seg og egner seg godt for kommunikasjon av lite data i store mengder i tillegg til isolert testing.

En viktig karakteristikk ved MVC modellen er at modulene under hver komponent blir laget med tanke på at de blir gjenbrukt. Dette kommer mye til spill ettersom våre moduler for ladebrikkeregistrering og strøminnhenting skal utnytte de samme konvensjonene som er laget for MVC modellen. Dette gjør de ved å enten gjenbruke eksisterende moduler eller tilegne sine egne submoduler til gjenbruk i andre komponenter.

6.2 Sekvensdiagram



Figur 20: Sekvens over registreringsprosess av en eller flere ladebrikker etter prototype evaluering



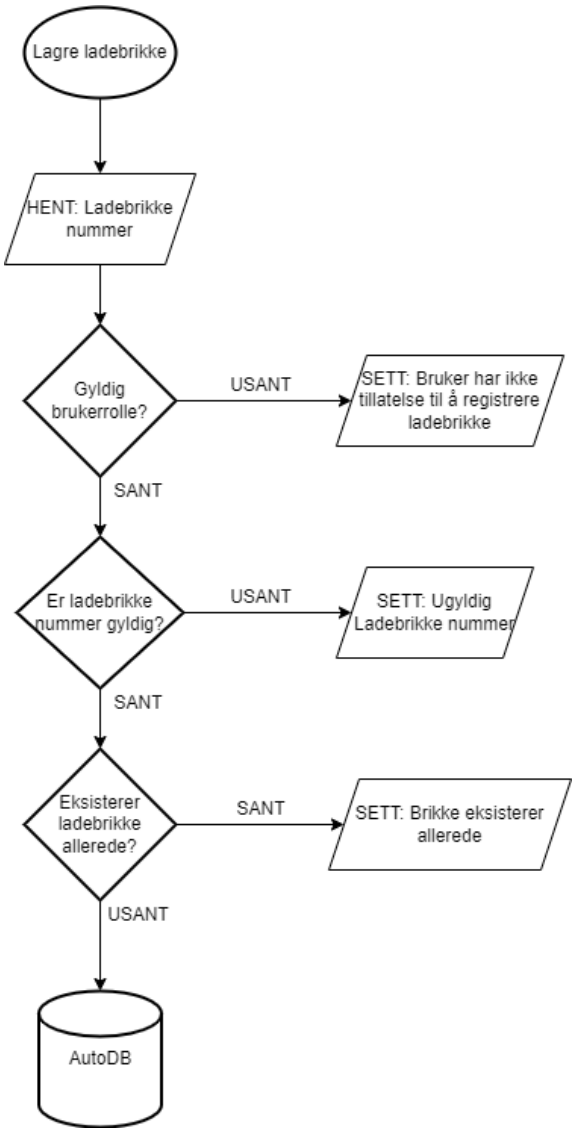
Figur 21: Sekvens over registreringsprosess av en lader mot Enode etter prototype evaluering

6.3 Flytskjema

Flytskjema vil gi oss innsikt i sekvensen av avgjørelser som må tas hensyn for, slik at en oppgave i hvert skjema kan utføres. Vi laget flytskjema ettersom de er en støttespiller for diverse prosesser for en programmerer. For en gruppe programmerere er det viktig å finne verktøyet som får jobben gjort. Fordelene for flytskjema er:

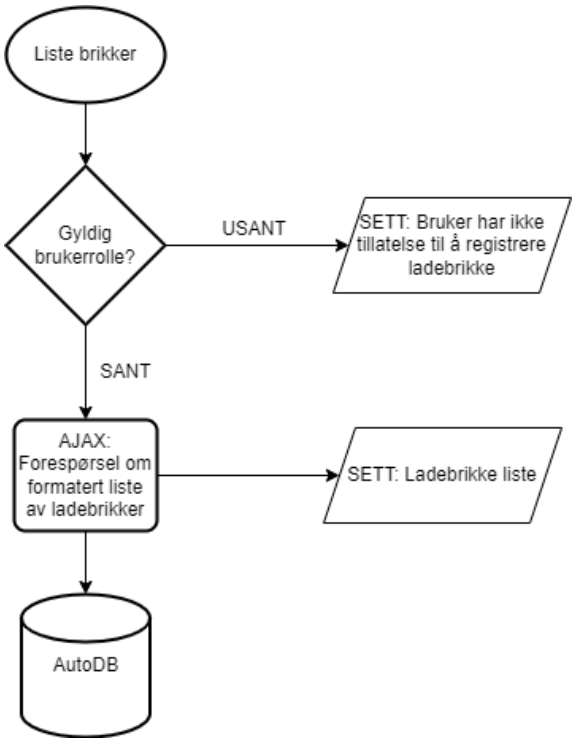
- Flytskjema kan sette lys på mulige problemer i arbeidsflyten for en prosess, som forårsaker en økning i effektivitet, kvalitet, sikkerhet eller ytelse.
- Flytskjema gir et perspektiv på hvordan høye datamengder, eller små datamengder i store antall ser ut på en simpel tegning.
- Flytskjema viser sammenhengen mellom andre flytskjema for å lett forstå konteksten til en større prosess.
- Flytskjema organiserer dets elementer kronologisk og gir de forskjellige roller basert på deres effekt. Type roller kan være valg, data, mottak av data, visning av data, osv.

Prosessen for å registrere en ladebrikke krever at den både registreres hos en ladeoperatør og lagres i CarAdmin databasen. Dette gir muligheten for å kontrollere og verifisere validiteten til en ladebrikke. Dette kan for eksempel være en prosess for å forebygge duplikate ladebrikkenummer i databasen. Derfor trengs en egen større prosess for lagring av ladebrikker hos CarAdmin som vist i følgende skjema.



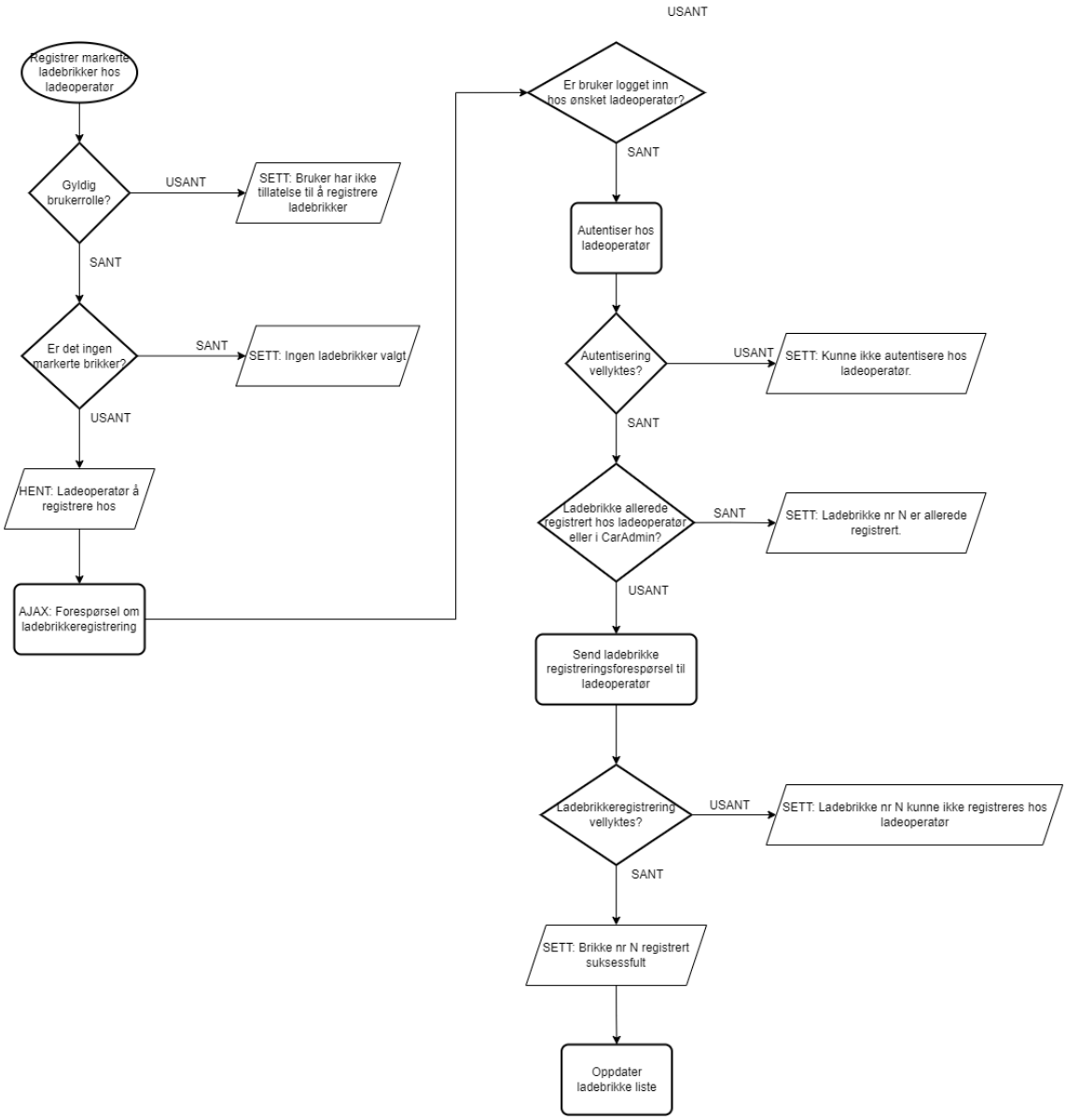
Figur 22: Flytskjema over lagring av en ladebrikke i CarAdmin databasen

Listing av ladebrikker trenger egne mindre prosesser som knyttes til andre flytskjema mot slutten.



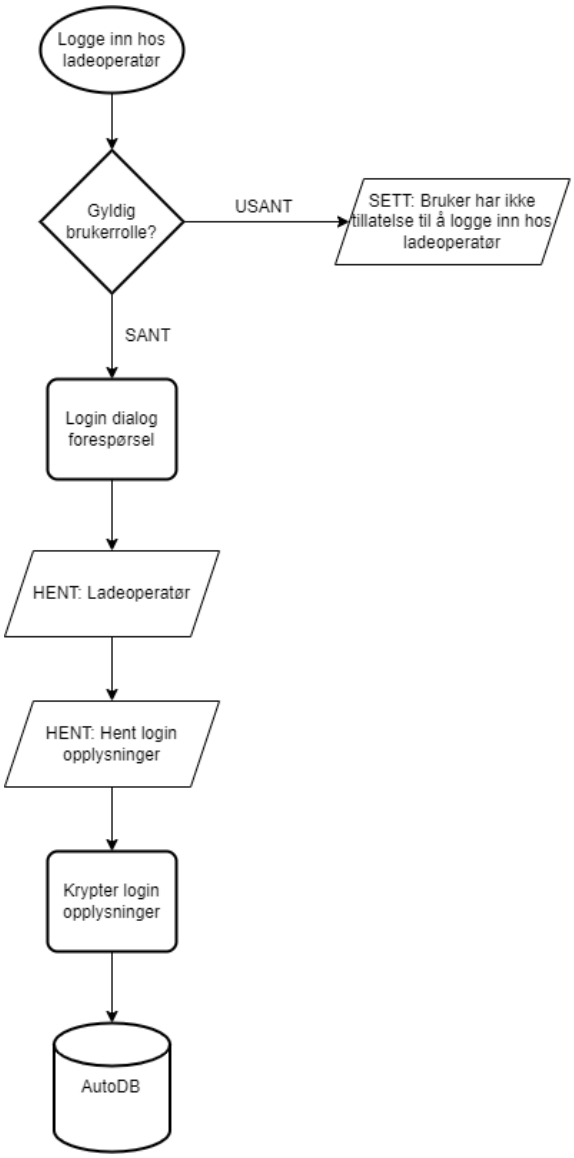
Figur 23: Flytskjema over listing av ladebrikker i CarAdmin databasen

Registrering av en ladebrikke krever at prosessen jobber i tandem med innlogging til ladeoperatør og lagring av ladebrikker.



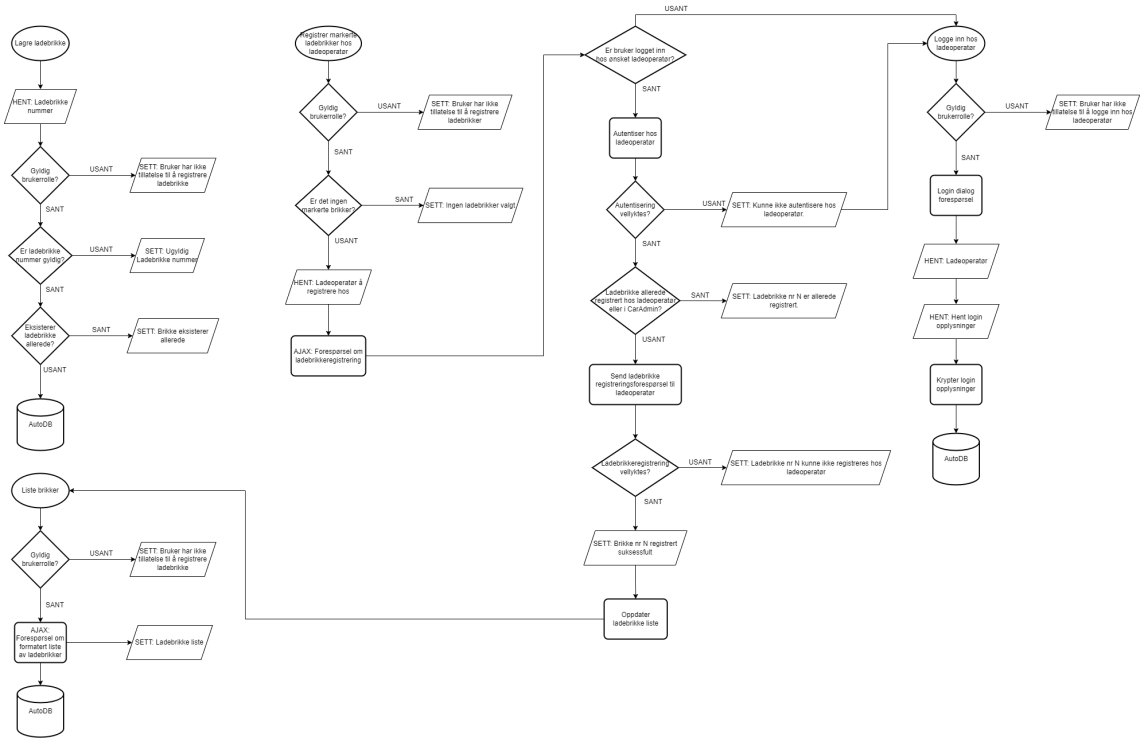
Figur 24: Flytskjema over registrering av en eller flere ladebrikker hos ladeoperatør

Ladebrikkeinnlogging er veldig direkte når det gjelder dataflyt og det er viktig at den holder en Single Sign On ordning[13]. Dette er grunnet den kan bli mer komplisert ettersom flere prosesser begynner å bli avhengig av den som en autoriseringskilde for kontakt med relevante ladeoperatører.



Figur 25: Flytskjema over innlogging hos ladeoperatørselskap

[13]Auth0. *Single Sign-On*. URL: https://auth0.com/docs/authenticate/single-sign-on (sjekket 6. mai 2022).



Figur 26: Flytskjema over steg og beslutninger som må utføres under ladebrikkeregistrering

6.4 Filstruktur

I denne seksjonen skal vi se på filstrukturen, hvordan modulene er bygget opp og hvordan koden henger sammen. For å få et bedre forståelse av hvordan disse modulene er implementert er det viktig å forstå hvordan de forskjellige delene henger sammen.

Vi kan dele opp modulene våre i tre hoveddeler:

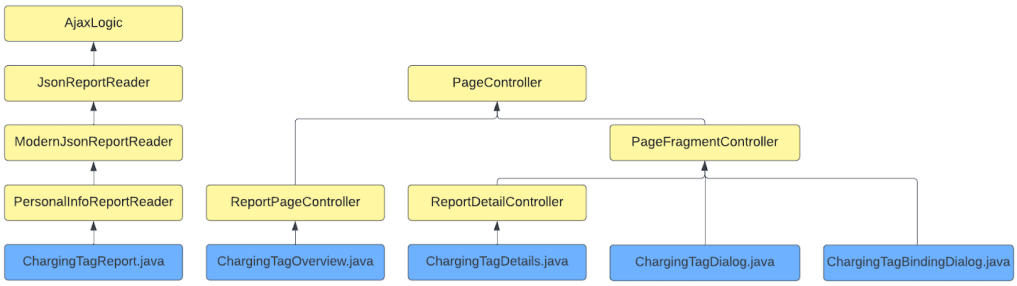
- Side kontroller filer
- JSP
- IO/AJAX

6.4.1 Side kontroller filer

Side kontroller filer inneholder subklasser som arver fra PageController superklassen. Denne superklassen fungerer som kontroller for en JSP-side. Den inneholder hovedsakelig kode for standard forespørsel håndtering og omdirigering i tillegg til å hente ut URL til JSP visningssiden.

Klassene er fordelt enda videre for forskjellige bruksområder. For eksempel kan vi se nærmere på figur 27 som viser kontroller klassehierarkiet for ladebrikke tabell

siden. Vi ser at ChargingTagDetails.java som er ansvarlig for detalj vinduet for hver ladebrikke, inkluderer ReportDetailController klassen. Det er denne klassen som tar for seg oppgaven av å hente ut og returnere en liste med alle rapport detaljene vi skal presentere. Videre blir PageFragmentController klassen brukt til å gjøre siden om til et fragment og enten vise den som et dialog vindu eller som i dette tilfellet integrere rett inni siden. Denne fordelingen oppnår høy gjenbruk av kode og gjør det enklere å implementere nye komponenter og hele sider.



Figur 27: Kontroller klassehierarkiet for ladebrikke tabell siden

6.4.2 JSP

JSP er server side programmer som kjører inne i en HTTP server. Man kan si at JSP er Java inne i HTML, som gir godt grunnlag for design. JSP gir oss også separasjon av dynamiske og statiske komponenter som vil si at vi kan lage kode logikk og legge det inne i statiske mal. I implementasjonen vår er disse filene ansvarlig for oppsett og selve visningen av siden. Det er også her vi utnytter og legger sammen tagger for å generere ønsket utseende. Hver JSP har en tilsvarende java klasse som sender informasjonen og kaller den fram. Vi ser nærmere på sammenhengen av disse filene i kapittel Klassestruktur.

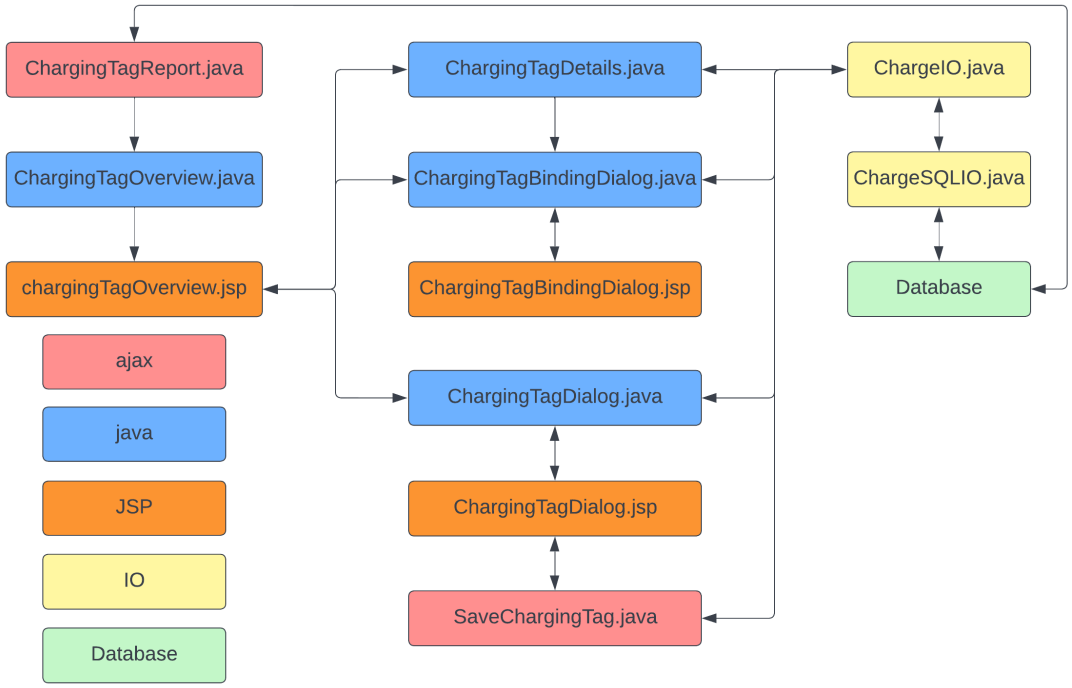
6.4.3 IO/Ajax

Disse filene håndterer kommunikasjonen mellom programmet og databasen. Begge inneholder SQL kode som henter, skriver eller oppdaterer informasjonen i databasen, men har forskjellige bruksområder. Som regel når vi definerer en ny funksjon for å hente informasjon fra databasen, som for eksempel alle ladebrikker når vi laster inn ladebrikke tabellen, samler vi disse funksjonene i IO filene. AJAX filene brukes når vi trenger å oppdatere siden uten å måtte laste den inn på nytt.

Som regel når vi definerer en ny funksjon for å hente informasjon fra databasen, for eksempel uthenting av ladebrikker til ladebrikke tabellen, samler vi disse funksjonene i IO filene.

6.5 Klassestruktur

På figur 28 kan vi se en forenklet modell av klassestrukturen for ladebrikke tabell siden. For enkelhets og oversiktlighetens skyld har vi valgt å ikke ta med alle komponentene fra tabell siden. Modellen viser hvordan de forskjellige delene inkluderer hverandre og kommuniserer med hverandre. Når tabellen blir lastet inn begynner vi i ChargingTagReport.java som er en AJAX klasse og henter inn data fra databasen gjennom SQL kall. Videre går denne dataen til ChargingTagOverview.java som organiserer dataen for tabellen. Her setter vi også opp innholdet til kolonene, rekkefølgen, antall elementer og linker knapper med funksjoner. Denne dataen blir da videresendt til chargingTagOverview.jsp som har ansvaret for visningen av siden som blir lastet inn i hovedområdet referert i figur 5 om sideoppsett.



Figur 28: Forenklet modell av ladebrikke tabell filstrukturen

6.5.1 Funksjonalitet

Siden inneholder flere funksjoner som for eksempel muligheten til å se detaljer for hver ladebrikke håndtert av ChargingTagDetails.java. Filen bruker funksjoner definert i ChargeIO.java filene til å hente detaljer om ladebrikken fra databasen, organisere informasjonen og sende den tilbake for visning til chargingTagOverview.jsp. Videre inneholder detalj vinduet flere funksjoner som gir muligheten for å legge til og fjerne forbindinger mellom ladebrikke og bil. Funksjonene ligger i ChargingTagBindingDialog.java som også har sin egen chargingTagBindingDialog.jsp fil som håndterer visningen av dialogvinduet for å skape en forbinding. Disse funksjonene er bundet til knapper som er definert i chargingTagOverview.jsp.

Et annet eksempel på funksjonene er for å legge til nye ladebrikker håndtert av ChargingTagDialog.java. Dette er dialogvinduet som håndterer lagring av nye ladebrikker og ladebrikke redigereing. I redigeringstilfellet bruker vi funksjoner definert i ChargeIO.java til å hente inn informasjonen for ladebrikken som skal redigeres. Videre blir denne informasjonen sendt til chargingTagDialog.jsp som tegner dialogvinduet. Her har vi et tilfelle hvor vi bruker en AJAX klasse til å lagre eller oppdatere ladebrikke informasjonen. Dette er fordi vi vil at tabellen skal oppdateres med den nye informasjonen uten at vi trenger å laste inn siden på nytt. Dette tilfellet er også litt spesielt fordi isteden for å ha SQL kallene direkte i Ajax funksjonen for å hente data, bruker den funksjoner som er definert i ChargeIO.java. Dette er fordi vi gjenbruker noen av funksjonene fra IO klassen til error håndtering og da var det naturlig å samle alle SQL funksjonene for ladebrikker på samme sted.

7 Utviklingsprosess

7.1 Utviklingsmetodikk

Formålet med å følge en smidig utviklingsmetodikk er å gi utviklerene et effektivt rammeverk med metoder og aktiviteter som skal hjelpe med å gå fra ide til ferdig produkt. Her skal vi presentere metodikkvalget, inkludert foretatte endringer gjennom prosjektperioden og motivasjonen bak de.

7.2 Valg av metodikk

Vi har valgt å følge Water-Scrum-Fall[14] modellen for utvikling. Videre i kapitelet skal vi argumentere for hvorfor vi har gått for denne modellen.

7.2.1 Karakteristikk ved prosjektet som var styrende for valg av utviklingsmodell

- Ettersom det må både læres opp i kodebasen for CarAdmin og innen teknologier rundt styring av ladebrikker, vil det forekomme endringer i løsningen, inkludert å beregne tid for jevn opplæring av program-/maskinvare teknologier.
- Vår gruppe er avhengig av tett samarbeid med oppdragsgiver gjennom hele prosjektet.
- Prosjektets implementasjon er en modul i kodebasen til CarAdmin og trenger ikke sitt eget oppsett for programvaredistribusjon eller vedlikehold, ettersom dette er allerede implementert.
- Prosjektet krever en tidlig prototype for å forsterke design og implementasjonsvalg.
- Det kreves initiativ til å holde jevn kontakt med forskjellige ladeselskaper eller tredjeparts-teknologier gjennom prosjektet.

7.2.2 Argumentasjon for valg av modell

Det er viktig for oss at fordeler og ulemper mellom modellene ble veiet, for å praktisere metodikken best tilegnet oppgaven vår. Oppgaven ble hovedsakelig delt opp i fire hovedfaser; planlegging, opplæring, implementasjon, og rapport, hvor implementasjon innebærer alt av design, kode og testing.

En betydelig del av opplæringsfasen var undersøkelse og kartlegging av ladestasjoners løsninger. Startfasen av prosjektet krever grundig undersøkelse og planlegging av

[14] ReQtest. *How to Make Agile and Waterfall Methodologies Work Together*. URL: https://reqtest.com/agile-blog/agile-waterfall-hybrid-methodology-2/ (sjekket 17. jan. 2022).

mulighetene rundt bruk av ladestasjoner, ettersom gruppen hadde veldig lite erfaring med fagområdets teknologier. Siden vi jobber med tredjeparts løsninger som vi ikke har kontroll over er det viktig at vi planlegger hva som skal utvikles, og tenke på hva vi trenger fra ladestasjonene for å gjennomføre oppgaven. Hvis ikke risikerer vi å møte på problemer senere i utviklingen, som vi ikke kan løse på grunn av manglende funksjonalitet fra ladestasjonene sine tjenester. Denne arbeidsformen passer inn med fossefallsmetoden, som legger vekt på nøye planlegging i startfasen av prosjektet.

Ulempen med fossefallsmodellen er at det er en lineær modell hvor man i prinsippet ikke kan gå tilbake faser. Dette er spesielt problematisk i implementasjonsfasen, siden vi vil gi oppdragsgiver rom til å komme med innspill underveis. Derfor mener vi SCRUM modellen er bedre egnet enn fossefallsmodellen i implementasjonsfasen. Ved å kombinere fossefallsmodellen med SCRUM modellen vil vi benytte fordelene fra begge modellene og fjerne ulempene. Systemutviklingsmodellen vi velger er altså Water-Scrum-Fall.

Det var ikke nødvendigvis dårligere kvaliteter som andre smidige modeller hadde over denne modellen, men heller at gruppen og oppdragsgiveren har mest erfaring med metodikken bak SCRUM, i tillegg til at sprint karakteristikken bak SCRUM kan simpelt inkludere andre konsepter. Derfor valgte vi å innlemme elementer som kjennetegner andre smidige modeller i SCRUM modellen. Dette inkluderer backlog som kjennetegner Kanban metodikken.

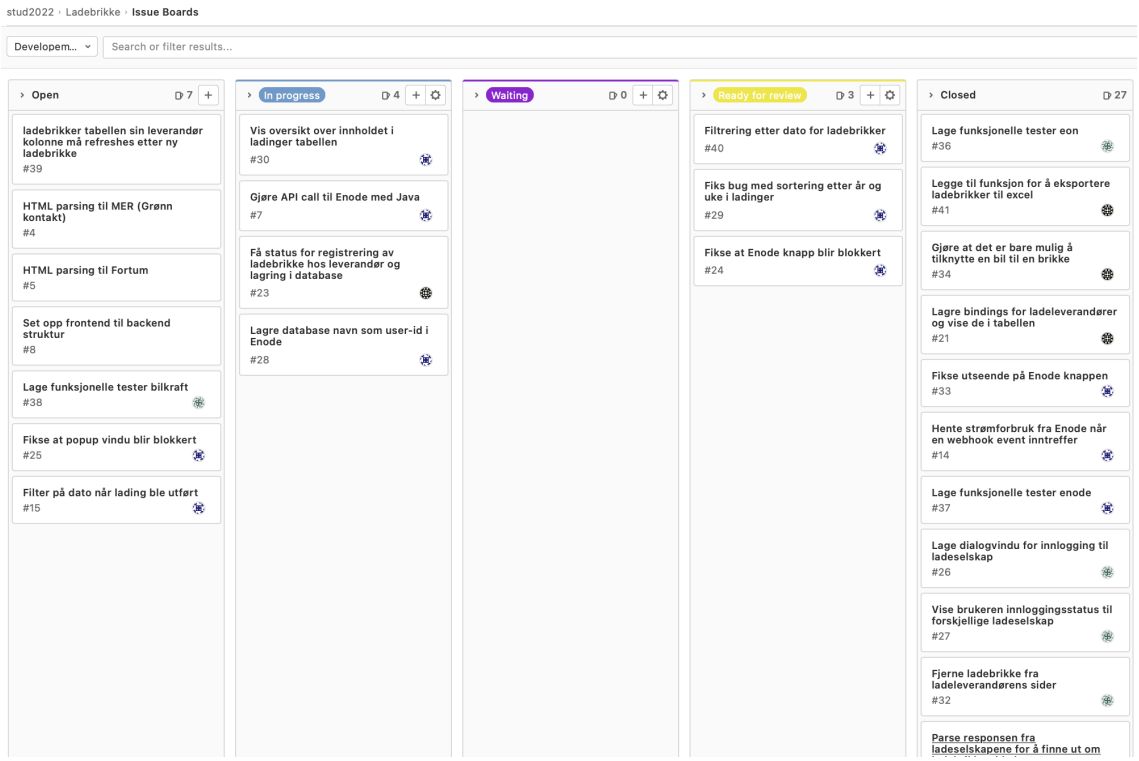
7.2.3 Anvending av utviklingsmodellen

Det finnes ingen fasit på hvordan man skal sette opp en utviklingsprosess siden den må tilpasses hvert prosjekt. Vi har valgt å bruke Water-Scrum-Fall. Derfor legger vi mye vekt på grundig planlegging i startfasen av prosjektet. Når vi har nok informasjon til å starte utvikling går vi over til å følge Scrum. Vi kombinerer Scrum med Kanban for å holde oversikt over statusen til de ulike oppgavene.

Vi har valgt å ha sprinter som varer i en uke. Dette er på grunn av at oppgaven vår gjennomgikk mange endringer innen metoden for henting av ressurser som API, og vi trenger kortere sprint for å raskere ta endringer i utviklingsfasen. Dermed har vi et sprint møte hver Mandag (etterhvert flyttet til Onsdag) hvor vi blir enige om arbeidet som skal gjøres den kommende sprinten, og fordele oppgaver innad i gruppen. I samme møtet skal vi reflektere over arbeidet som ble gjort forrige sprint. Dette er for å passe på at vi er i rute, og identifisere hva som kan bli gjort bedre eller annerledes. Vi har møte med oppdragsgiver etter forespørsel og behov fra begge parter hver uke. I hvert møte skal vi demonstrere hva vi gjorde i forrige sprint og stille spørsmål som vi noterer ned på forhånd.

I startfasen av prosjektet lager vi en produkt backlog med oppgavene som må bli gjort. Underveis i utviklingen kan vi legge til nye oppgaver som kommer fram. Vi skal hente oppgaver fra backloggen og legge de til på Scrumban-tavlen vår i GitLab(figur 29) slik at det er enkelt å holde oversikt over statusen til de forskjellige oppgavene. Scrumban-tavlen vår skal ha følgende tabeller:

- To do
  - Oppgaver i backloggen som ikke har blitt påbegynt.
- In progress
  - Oppgaver som er under arbeid.
- Waiting
  - Oppgaver som venter på eksterne faktorer for å fortsette med.
- Review
  - Oppgaver som er klar for gjennomgang av et annet gruppemedlem.
- Done
  - Ferdige oppgaver.



Figur 29: Scrumban i gitlab

7.3 Verktøy

- Overleaf
  - Skybasert LaTeX-redaktør vi har brukt til å skrive rapporten og prosjektplan.
- Postman

  - Brukt til å utforske API-endepunkt.

- Docker
  - Brukt til å sette opp servermiljøet programmet kjører på.
- IntelliJ
  - Vi brukte utviklingsmiljøet IntelliJ gjennom hele prosjektet.
- HeidiSQL
  - HeidiSQL kan brukes til å åpne og redigere databaser. Vi brukte den gjennom prosjektet om vi ville manuelt forandre noe med databasen.
- Google Drive
  - Brukt til å ta notater og skrive timelogg.
- Gitlab
  - Versjonskontrollverktøy.
- GanttProject
  - Design av Ganttmodell
- Draw.io
  - Tegneprogram for diverse modeller
- Meld
  - Versjonskontrollverktøy for inspisering av filforskjell
- Creately
  - Tegneprogram brukt til å lage Use case diagram.

8 Implementasjon

Prosjektet vårt ble utviklet i ETC sitt utviklingsmiljø. Vi fikk altså tilgang til et godt etablert rammeverk for utvikling og fungerende programvare vi kunne tilknytte implementasjonen vår til. Dette så vi først på som en fordel, men det viste seg også å ha noen ulemper.

Ulempen med å kode i en etablert kodebase, var at vi måtte bruke lang tid på å lære rammeverket før vi kunne begynne på løsningen. Kodebasen er stor og komplisert, og viste seg å være en større utfordring å lære enn vi først så for oss.

Selv om det var tidkrevende, var det også fordeler med å jobbe i kodebasen til ETC. Etter vi hadde lært om miljøet deres hadde vi tilgang til mange ferdiglagde klasser og funksjoner. Vi kunne også følge standardene deres innen design, struktur og filoppsett.

8.1 Moduler

Implementasjonen vår kan deles opp i tre moduler:

- Ladebrikke tabell
- Strømforbruk
- Registrering av ladeselskap

8.1.1 Ladebrikke tabell

Denne modulen inneholder informasjon om ladebrikker og deres status. Her har brukeren også muligheten å utføre forskjellige funksjoner med disse ladebrikkene:

- Legge til eller fjerne ladebrikke fra CarAdmin.
- Registrere eller fjerne ladebrikke hos et ladeselskap.
- Massefunksjon for registrering av mange ladebrikker.
- Redigere ladebrikkens informasjon.
- Tilknytte en bruker mot en ladebrikke.
- Tilknytte et kjøretøy mot en ladebrikke.

I tillegg har brukeren muligheten til å filtrere og søke etter ladebrikker, samt bestemme hva slags informasjon som skal vises for hver av ladebrikkene i tabellen.

8.1.2 Strømforbruk

I modulen som omhandler strømforbruk har brukeren muligheten til å registrere en ny lader, og automatisk få inn strømforbruk fra de registrerte laderne etterhvert som man lader en elbil.

8.1.3 Registrering av ladeselskap

Her har brukeren muligheten til å tilknytte en registrert bruker fra et av de mulige ladeselskapene. For å gjøre dette blir brukeren bedt om å oppgi brukernavn og passord som brukes ved innlogging hos disse ladeselskapene. Dette er nødvendig for å automatisk registrere ladebrikker hos selskapene. Innloggingsinformasjon blir testet når brukeren legger den til, og statusen vises som et lite ikon ved siden av brukernavnet.

8.2 Tagger

JSP gir oss muligheten til å lage egne gjenbrukbare tagger som kan brukes som byggeklosser for våre sider. Det som gjør taggen gjenbrukbar er muligheten til å bruke taggen og sende inn variabler fra en annen JSP fil. ETC utnytter taggene for hjelpe med å definere et standardisert designoppsett for de aller fleste sidene i applikasjonen deres. I tillegg til å hjelpe med å standardisere design og forenkle vedlikehold og eventuelle endringer i fremtiden, fremmer det også gjenbruk av kode. Taggene vi har brukt i dette prosjektet er laget av ETC.

Et eksempel på en tag vi har brukt er OverviewReport.tag. Taggen brukes for å tegne opp hele innholdet i hovedområdet for ladebrikke siden. Denne taggen er bygget opp av to komponenter, ReportFilter.tag som tar for seg filtreringen og søking for ladebrikker og DataGrid.tag som har ansvar for tabellen. Listing 1 viser hvordan man kan bruke OverviewReport taggen og hva som sendes med.

```
<etc:OverviewReport
    params="${chargingTagParameters}"
    dataGridName="ChargingTagOverview"
    ajaxUrl="ChargingTagReport"
    noDataMsg="${m['ChargingTag.NoData']}"
    tableFields="${mainLayout}"
    detailsFunction="loadChargeTagDetails"
    refreshFunction="reloadChargingTagOverview"
    exportReportUrl="ChargingTagExport.ajax"
    gridConditions=""
    hasLocChooser="${hasLocChooser}"
    dataParameters="${dataGridParameters}"
    headers="true"
/>
```

Listing 1: Eksempel på bruk av OverviewReport tag

I kodesnutten over er noen attributter enkapsulert i tegnene '${}', som betyr at dette er en variabel som har blitt sendt fra en Java fil til JSP. Et eksempel på dette er '${mainLayout}', som inneholder hvilke kolonner ladebrikketabellen skal

ha. 'mainLayout' blir lagd og sendt fra ChargingTagOverview.java filen, som vist i listing 2.

```
JSONArray mainLo = new JSONArray();

mainLo.put(DataGridUtil.getColumnDescriptor("rfidNumber", m.t("ChargingTag.
rfidNumber"), false));

mainLo.put(DataGridUtil.getColumnDescriptor("assets", m.t("ChargingTag.
bindAsset"), false));

mainLo.put(DataGridUtil.getColumnDescriptor("description", m.t("ChargingTag.
description"), false));

mainLo.put(DataGridUtil.getColumnDescriptor("providers", m.t("ChargingTag.
chargeCompany"), false));

....

mainLo.put(DataGridUtil.getColumnDescriptor("id","", false, "reportCheckbox").
put("headerFormatter", "reportHeaderCheckbox"));

request.setAttribute("mainLayout",mainLo);
```

Listing 2: Kodesnutt fra ChargingTagOverview.java

Vi velger spesifikasjonene til tabellen i java filen og sender informasjonen videre til JSP ved hjelp av 'request.setAttribute'. Informasjonen blir lagt inni taggen ved forespørsel for nettsiden som da automatisk genererer tabellen vår med spesifikasjonene vi har gitt den. Tabellen vil i utgangspunktet være tom og vi trenger å lage en funksjon som skal kunne hente data fra databasen og sette den på riktig sted i tabellen. Denne delen av arbeidet gjøres gjennom AJAX eller IO filer som vi snakker mer om i kapittel 6.4.3. Dette vil da generere en ny rad for hver gjenstand (i dette tilfellet ladebrikke) i tabellen med tilhørende informasjon.

8.3 Databasekall

Vi bruker klassen SQLFactory som ETC har utviklet for å gjøre database kall. SQLFactory klassen brukes for håndtering av SQL kallene, håndtering av data og logging av utførte kall. Under har vi lagt til et eksempel fra ChargeSQLIO.java som er et godt eksempel på et vanlig SQL kall. Funksjonen brukes både til å lagre nye ladebrikker eller redigere eksisterende ladebrikker i databasen.

```
/**
 * Hvis ladebrikken har en id, oppdater den, ellers leg den til i databasen.
 *
 * @param ct ladebrikke objektet som blir lagret
 */
@Override
public void saveChargingTag(ChargingTag ct) throws CarAdminException {
    // Setter sql queryen til å lagre ladebrikke i databasen
    String sql = "INSERT INTO chargingtag (rfidNumber,description,created,
        createdby) VALUES(?,?,NOW(),?)";
    // Hvis ladebrikken har en id blir queryen byttet ut med å oppdatere
    //  den eksisterende ladebrikken
    if(ct.getId() > 0) {
        sql = "UPDATE chargingtag SET rfidNumber=?,description=? WHERE id = ?";
    }

    int idx = 1;
    // Forsøker å koble til databasen
    try(Connection c = sqlFactory.getConnection() ) {
```

```
20              try(PreparedStatement p = c.prepareStatement(sql,
21              Statement.RETURN_GENERATED_KEYS)) {
22                  // Legger til dataen fra ladebrikke objektet for pushing
23                  p.setString(idx++,ct.getRfidNumber());
24                  p.setString(idx++,ct.getDescription());
25                  if(ct.getId()>0){
26                      p.setInt(idx,ct.getId());
27                  }else{
28                      p.setInt(idx,ct.getCreatedBy());
29                  }
30                  // Utfører oppdateringen
31                  p.executeUpdate();
32              }
33          }
34          // Hånterer error ved lagring og skriver ut en feilmelding i konsolen
35          catch(SQLException sqle) {
36              throw new CarAdminException("Feil ved lagring av ladebrikke: "+
37                  sql,sqle);
38          }
39      }
```

Listing 3: Eksempel på databasekall

Funksjonen over tar imot et ChargingTag dataklasse objekt som inneholder informasjon til en ladebrikke. Det som skiller om funksjonen skal legge til en ny ladebrikke eller oppdatere en eksisterende er ladebrikkens ID. Hvis ladebrikken er ny så inneholder den medsendte dataklassen ikke en ID, og blir heller generert automatisk i databasen. Hvis ladebrikke objektet inneholder en ID, så vet vi akkurat hvilken ladebrikke som skal oppdateres i databasen. Vi kan se denne testen skje på linje 13 i koden.

På linje 19 og 20 ser vi bruken av SQLFactory klassen. På disse linjene forsøker vi å koble til databasen og gjøre klar SQL kallet. Videre organiserer vi og legger til dataen fra ladebrikke dataklassen. SQL-kallet blir utført på linje 31. Under der igjen finner vi litt kode for error håndtering og produsering av utskrift til konsoll.

Når vi legger til eller oppdaterer bindinger for objekter i databasen blir dette logget i tillegg. Under har vi lagt til kode for denne loggingen. Koden er ganske enkel og inneholder et kall på funksjonen som legger til selve bindingen på linje 13, en sjekk for userID og kall for en funksjon som utfører loggingen.

```
1       /**
2        * Legger til en knytting for ladebrikke, og logger tilknytningen dersom en
        bruker blir knyttet.
3        *
4        * @param tagId  id på ladebrikken
5        * @param assetNo id på kjøretøy
6        * @param userId  id på bruker
7        * @param si      sesjonsobjekt
8        * @param context Kontekst for uthenting
9        * @throws CarAdminException hvis io feiler
10       */
11      @Override
12      public void addChargingTagBinding(int tagId, int assetNo, int userId,
        SolutionSessionInfo si, GdprPrivacyContext context) throws CarAdminException {
13          chargeIO.addChargingTagBinding(tagId, assetNo, userId);
14
15          if (userId > 0) {
16              // Logg knytning av bruker mot ladebrikke
17              doLogChangeEntry(si, null, String.valueOf(userId), userId,
18                               QueryAction.INSERT, GDPRChangeContext.CHARGING_USER,
        context, tagId);
19          }
```

```
20      }
```

Listing 4: Eksempel på logge funksjon

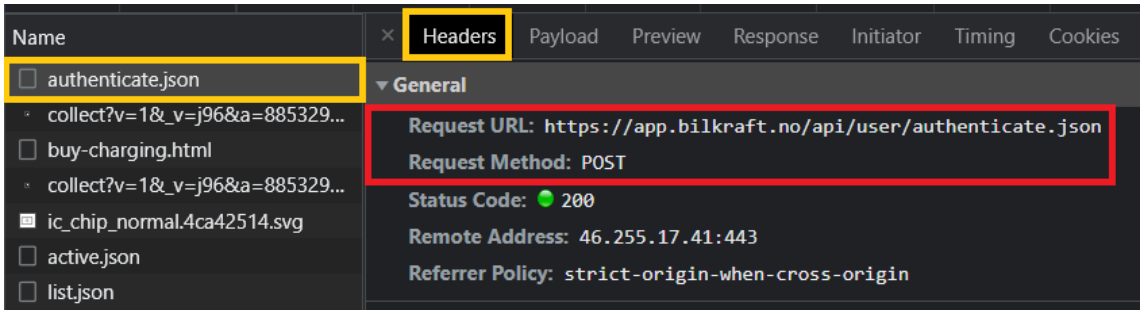
8.4 Implementasjon for ladebrikker

I denne seksjonen skal vi se på måten vi implementerte modulen for registrering av ladebrikker. Eviny(Bilkraft) og E.ON er begge ladeselskap, og har relativ lik implementasjon. Vi valgte derfor å slå dem sammen i denne seksjonen, men påpeker når implementasjonene skiller seg.

8.4.1 Utforsking av API endepunkt

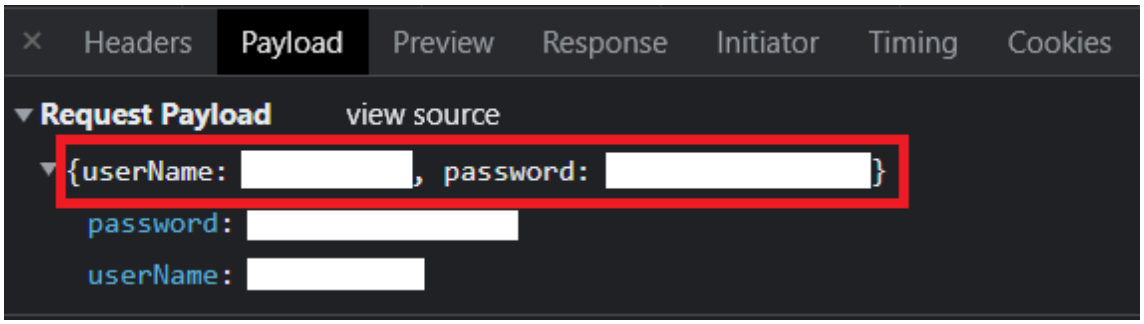
For å bruke API endepunktene til Bilkraft og E.ON, begynte vi med å undersøke hvilke endepunkt som var tilgjengelige. Vi måtte og finne ut av hva vi måtte sende inn til endepunktene, hvordan vi autentiserte forespørselene våre, og hva vi fikk tilbake. Mangel på dokumentasjon gjorde dette til en omstendelig jobb, hvor vi ble nødt til å inspisere nettsiden og se gjennom nettverkstabellen for å finne tilgjengelige endepunkter. Vi skal vise både Bilkraft og E.ON sine endepunkt for innlogging, da de er ganske forskjellige.

Bilkraft
Endepunktet fra Bilkraft vi skal se på som eksempel, håndterer autentisering av innloggingsinformasjon. Figur 30 viser generell info om endepunktet, som addresse og forespørselsmetode.



Figur 30: Bilde av generell info om Bilkraft sitt login endepunkt

Man må i tillegg til endepunkt URLen vite hva man skal sende inn, og det fant vi under 'Payload'. Figur 31 viser at vi må sende inn brukernavn og passord som json.



Figur 31: Bilde av payload til Bilkraft sitt login endepunkt

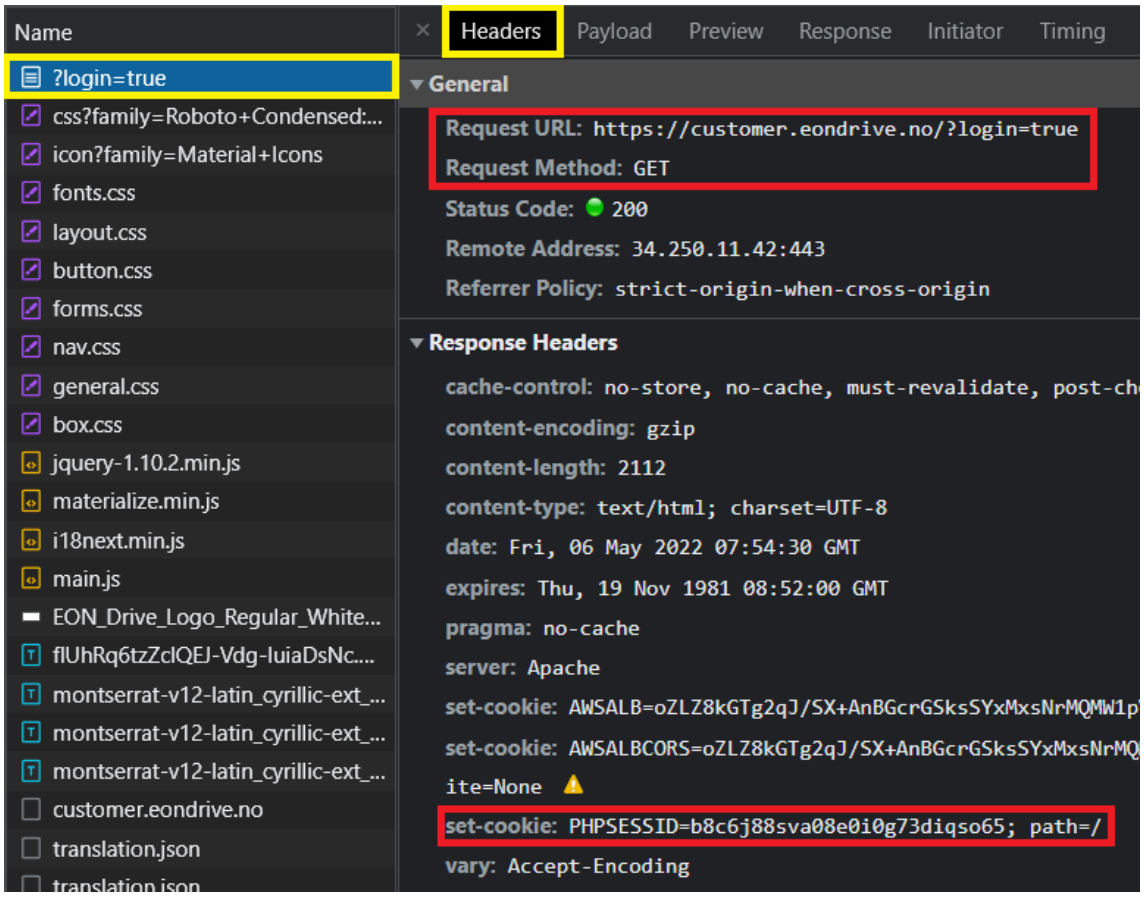
Til slutt må man vite hva APIet returnerer, og det fant vi under 'Response'. Figur 32 viser dette.

Figur 32: Bilde av response til Bilkraft sitt login endepunkt

Det vi bruker av responsen fra autentiseringsendepunktet er 'token' som inneholder en validert token så lenge man har sendt inn riktig innloggingsinformasjon. Denne token må vi bruke i resten av API-kallene for å bevise at forespørselene våre er autentiserte.

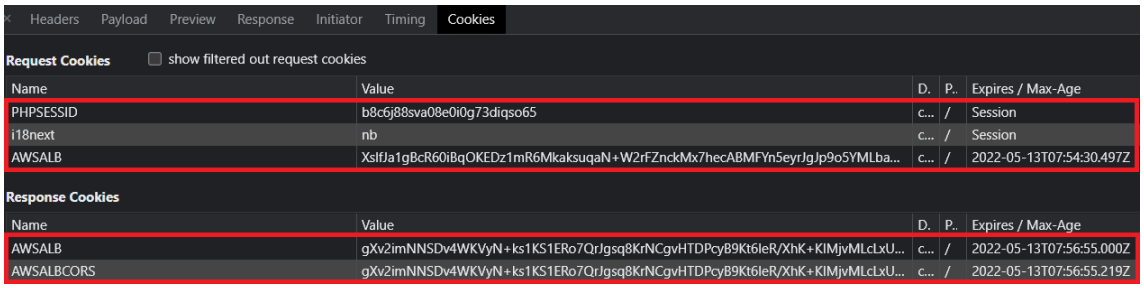
E.ON

Endepunktet vi skal se på fra E.ON håndterer også autentisering, men har en annen implementasjon enn Bilkraft sitt endepunkt. E.ON gir oss en sesjons id, som vi senere må autentisere med APIet deres. Vi starter derfor med å sende et GET kall til 'customer.eondrive.no/?login=true', og henter ut responsen 'set-cookie' som inneholder 'PHPSESSID'. I figur 33 kan vi se at sesjons id-en er 'PHPSESSID b8c6j88sva08e0i0g73diqso65'.



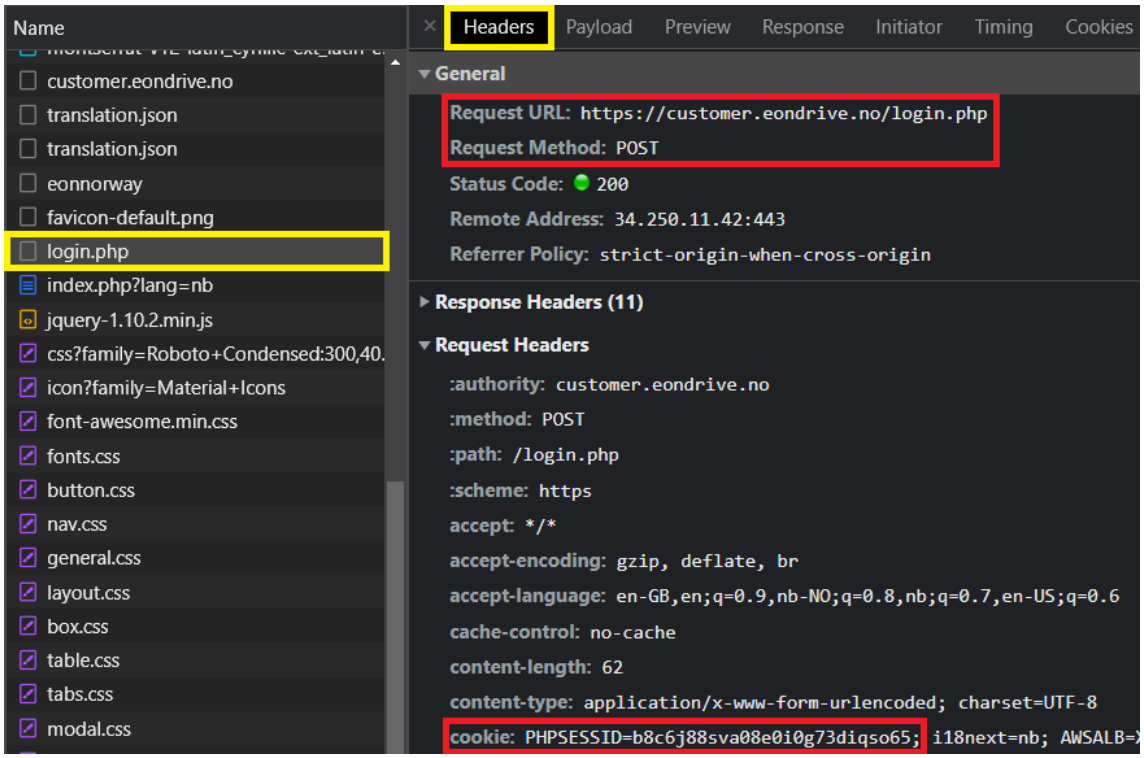
Figur 33: Bilde av generell info fra E.ON sin login side

Når man bruker en nettleser, som f.eks. Chrome, vil alle cookies bli automatisk lagret (figur 34) og brukeren trenger ikke tenke over hva som skjer i bakgrunnen. Vi må derimot lagre sesjon id-en, og sende den med API-kallet som en cookie (se kodesnutt 6).



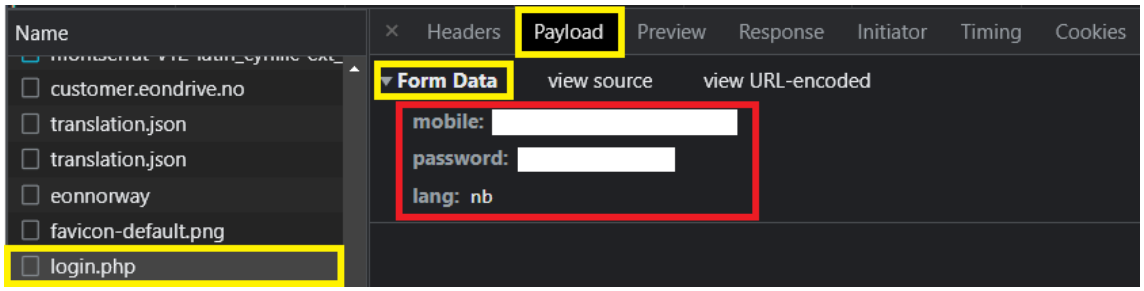
Figur 34: Bilde av cookies lagret i nettleseren Chrome, fra E.ON

Figur 35 viser endepunktet vi sender en POST request til, og hva som blir sendt med i forespørsel sin header. Cookie feltet i headeren inneholder sesjon id-en vi hentet tidligere.



Figur 35: Bilde av generell info til E.ON sitt login endepunkt

I 'payload' feltet vises hva som sendes med for å autentisere sesjons id-en, og i figur 36 kan man se at det er 'mobile', som er email, passord, og 'lang' som er språk. Legg også merke til at Bilkraft sitt endepunkt tar i mot json, mens E.ON sitt endepunkt tar i mot form data. Om email og passord er riktig blir sesjons id-en validert, og kan legges til i forespørsel header som cookie for å autentisere API-kall.



Figur 36: Bilde av payload til E.ON sitt login endepunkt

8.4.2 Innlogging

For å slette, hente og legge til nye ladebrikker krever både Bilkraft og EON at man sender med en validert token med hvert API-kall. For å få tak i en validert token, trenger vi at CarAdmin sine brukere oppgir brukernavn og passord til alle ladeselskap de vil knytte til CarAdmin. Vi lagrer dette trygt i CarAdmin sine databaser ved å kryptere det med CarAdmins krypteringsalgoritme. Deretter sender programmet en forespørsel for å logge inn hos ladeselskapet, og vi får returnert en validert token om

innloggingsinformasjonen var korrekt.

Som sagt er det viktig å hele tiden passe på at token man får tilbake er validert, eller så vil ikke de andre API-kallene til ladeselskapene fungere. Under skal vi se på et eksempel hvor vi behandler innlogging for Eon, og henter ut en validert token.

```
1     /**
2      * Oppdaterer validert token for forskjellige ladeselskap.
3      * @param cio - ChargeIO
4      * @param ladeselskap - ladeselskapet som får oppdatert token
5      * @param username - brukernavn brukt til å logge inn hos ladeselskap
6      * @param password - passord brukt til å logge inn hos ladeselskap
7      * @return String "success" om brukeren fikk logget inn og fikk en ny validert
     token, "fail" om det failet.
8      * @throws IOException
9      */
10    String dbTokenUpdate(ChargeIO cio, String ladeselskap, String username, String
     password) throws IOException {
11        ChargingTagProviderParser loginParser = new ChargingTagProviderParser();
12
13        String token = "";
14
15        //Sjekker hvilket ladeselskap vi skal logge inn på
16        if(Objects.equals(ladeselskap, "Eviny")) {
17            token = loginParser.bilkraftLogin(username, password);
18        }
19
20        if(Objects.equals(ladeselskap, "EON")) {
21            token = loginParser.eonLogin(username, password);
22        }
23
24        //Sjekker om vi fikk en validert token
25        if(token!=null && !token.isEmpty()){
26            try {
27                cio.updateChargingProviderToken(ladeselskap, token);
28                return "success";
29            } catch(CarAdminException e) {
30                e.printStackTrace();
31            }
32        }
33
34        return "fail";
35    }
```

Listing 5: Funksjon for å oppdatere token etter suksessfull innlogging

Det første vi gjør er å lage 'loginParser', som er et objekt fra klassen ChargingTagProviderParser. Denne klassen inneholder alle funksjoner vi trenger for å kunne logge inn, hente, slette og legge til nye ladebrikker. Om vi later som ladeselskapet 'E.ON' har blitt valgt, kan man se på linje 21 i kodesnutten over at vi kaller på funksjonen 'loginParser.eonLogin' og sender med brukernavn og passord. Denne funksjonen prøver å logge inn og hente et validert token. Under er det enda en kodesnutt, som viser 'eonLogin' funksjonen.

```
1     /**
2      * Logger inn til e.on, og returnerer token for innsending av data til e.on
3      *
4      * @return validert token
5      * @throws IOException
6      */
7     public String eonLogin(String username, String password) throws IOException {
8
9         //Henter SessionID før man logger inn
10        HttpClient httpClient = HttpClientBuilder.create().build();
11
12        HttpGet firstCookieRequest = new HttpGet("https://customer.eondrive.no/?
     login=true");
13
```

```
        HttpResponse firstResponse = httpClient.execute(firstCookieRequest);

        //Henter cookies under header 'Set-Cookie', og fjerner data vi ikke trenger
        Header[] firstCookies = firstResponse.getHeaders("Set-Cookie");
        String cookies = "";
        for(Header cookie : firstCookies) {
            cookies += cookie.getValue().split(";")[0];
            cookies += ";";
        }
        cookies = cookies.split(";")[2];

        //Logger inn og aktiverer dermed SessionID.
        OkHttpClient clientLogin = new OkHttpClient().newBuilder()
                .build();
        MediaType mediaType = MediaType.parse("text/plain");
        RequestBody bodylogin = new MultipartBody.Builder().setType(MultipartBody.
    FORM)
                .addFormDataPart("mobile",username)
                .addFormDataPart("password",password)
                .addFormDataPart("lang","nb")
                .build();
        Request request = new Request.Builder()
                .url("https://customer.eondrive.no/login.php")
                .method("POST", bodylogin)
                .addHeader("Cookie", cookies)
                .build();
        Response loginRes = clientLogin.newCall(request).execute();

        //Sjekker om vi klarte å logge inn, og returnerer enten cookies eller tom
    string.
        try {

            JSONObject resBody = new JSONObject(loginRes.body().string());

            if(resBody.has("result")){
                if(Objects.equals(resBody.getString("result"), "success")) {
                    return cookies;
                }
            }
        } catch(JSONException e) {
            return "";
        }
        return "";
    }
```

Listing 6: Funksjon for innlogging til E.ON

eonLogin er funksjonen vi bruker til å logge inn hos eon. Om brukernavn og passord brukeren sender med er korrekt, returnerer funksjonen en validert token. Hvis brukernavn eller passord er feil, returnerer den i stedet en tom string.

8.4.3 Registrering av ladebrikke

For å registrere en ny ladebrikke hos Bilkraft bruker vi API-endepunktet 'https://app.bilkraft.no/api/user/rfid/add.json'. Endepunktet tar 'name' og 'hexValue' som input. For 'name' sender vi inn bilnummer, og for 'hexValue' sender vi inn RFID-nummer. Som vanlig trenger vi å sende med en validert token for at endepunktet skal vite at vi er autentisert. Token sendes med som 'x-token'. Til slutt henter vi resultatet, og ser om kallet lykkes. Listing 7 viser koden for dette.

```
        /**
     * Bruker token til å sende rfidnummer og bilnummer til bilkraft, for å
    registrere en ny ladebrikke.
     *
     * @param rfidnummer RFID nummeret som skal bli lagt inn i bilkraft
```

```
     * @param bilnummer Bilnummeret som skal bli lagt inn i bilkraft
     * @param token Token man får fra login funksjonen
     * @throws IOException
     * @return Resultatet fra API kallet
     */
    public String bilkraftPost(String rfidnummer, String bilnummer, String token)
    throws IOException {
        // Legger inn bilnummer og rfidnummer i payload
        String payloadJava = "{\"rfid\":{\"name\":\""+ bilnummer +"\",\"hexValue\":
     \"" + rfidnummer + "\"}}";
        StringEntity entityJava = new StringEntity(payloadJava,
                                                   ContentType.APPLICATION_JSON);

        // Lager en forespørsel til bilkraft
        HttpClient httpClientJava = HttpClientBuilder.create().build();
        HttpPost httpRequestJava = new HttpPost("https://app.bilkraft.no/api/user/
    rfid/add.json");
        httpRequestJava.setEntity(entityJava);

        // Legger inn validert token i header
        httpRequestJava.setHeader("x-token", token);

        // Kjører forespørselen og henter ut resultat
        HttpResponse responseJava = httpClientJava.execute(httpRequestJava);
        String responsestring = new BasicResponseHandler().handleResponse(
    responseJava);

        // Setter resultat lik 'FAIL'. Blir satt til 'SUCCESS' om vi fikk noe
    nyttig som resultat fra forespørselen over
        String result = "FAIL";

        try {
            // Legger resultatet inn i et JSON Objekt
            JSONObject a = new JSONObject(responsestring);

            // Leser resultatkode og returnerer resultat
            if(Objects.equals(a.getString("resultCode"), "SUCCESS")) {
                result = "SUCCESS";
                return result;
            }
            if(Objects.equals(a.getString("errorCode"), "RFID_IS_IN_USE")) {
                result = "RFID_IS_IN_USE";
                return result;
            }
        } catch(JSONException e) {
            e.printStackTrace();
        }
        return result;
    }
```

Listing 7: Funksjon for innsending av ladebrikke til E.ON

8.4.4 Sletting av ladebrikke

Endepunktene for sletting av ladebrikker hos E.ON og Bilkraft er ganske like, men Bilkraft trenger å hente ladebrikker fra nettsiden før man kan slette dem. Derfor skal vi se på Bilkraft som eksempel for sletting.

Funksjonen som tar seg av sletting av ladebrikke heter 'bilkreaftDelete'. Den tar inn 'rfidnummer' og 'token' som parameter, og returnerer 'success' eller 'fail' avhengig av respons fra bilkraft endepunkt for sletting. Det som gjør sletting av ladebrikker litt vrient, er at endepunktet trenger id i tillegg til navn (bilnummer) og hexValue (RFID-nummer). I dette tilfellet er id knyttet opp til hver ladebrikke registrert hos Bilkraft, og vi har derfor ingen kontroll over id-en. Derfor må vi starte med å hente

ut alle registrerte ladebrikker fra Bilkraft, hente id-en deres, og kryssreferere med RFID-nummeret vi har lyst til å slette. Vi bruker altså to endepunkt for å slette ladebrikker fra Bilkraft. Dette er ikke et problem hos E.ON, da ladebrikkenummer er alt man trenger. Listing 8 viser kode for 'bilkraftDelete'.

```
1
2   /**
3      * Sletter en rfidbrikke fra bilkraft
4      *
5      * @param rfidnummer RFID-nummeret til brikken som skal slettes
6      * @param token Token fra login funksjon
7      * @return "success" eller "fail" avhengig av respons fra bilkraft endepunkt
8      */
9     public String bilkraftDelete(String rfidnummer, String token) throws
      IOException {
10        bilkraftTags ladebrikker = bilkraftGetLadebrikker(token);
11        String id = "";
12        String name = "";
13        for(int i = 0; i < ladebrikker.rfids.size(); i++) {
14            if(ladebrikker.rfids.get(i).hexValue.equals(rfidnummer)){
15                id = ladebrikker.rfids.get(i).id;
16                name = ladebrikker.rfids.get(i).name;
17                break;
18            }
19        }
20
21        if(id.equals("") || name.equals("")){
22            return "fail";
23        }
24
25        String bodyString = "{\n" +
26                "  \"rfid\": {\n" +
27                "    \"id\": \""+ id +"\",\n" +
28                "    \"name\": \""+name+"\",\n" +
29                "    \"hexValue\": \""+rfidnummer+"\",\n" +
30                "    \"externalReference\": null,\n" +
31                "    \"isActive\": true\n" +
32                "  }\n" +
33                "}";
34
35        OkHttpClient client = new OkHttpClient().newBuilder()
36                .build();
37        MediaType mediaType = MediaType.parse("application/json");
38        RequestBody body = RequestBody.create(mediaType, bodyString);
39        Request request = new Request.Builder()
40                .url("https://app.bilkraft.no/api/user/rfid/remove.json")
41                .method("DELETE", body)
42                .addHeader("x-token", token)
43                .addHeader("Content-Type", "application/json")
44                .build();
45        Response response = client.newCall(request).execute();
46
47        ObjectMapper mapper = new ObjectMapper();
48        mapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, false);
49        String test = response.body().string();
50        bilkraftDeleteRes resObject = mapper.readValue(test, bilkraftDeleteRes.
      class);
51
52        if(resObject.resultCode.equals("SUCCESS")) {
53            return "success";
54        }
55
56        return "fail";
57    }
```

Listing 8: Funksjon for sletting av ladebrikke til Bilkraft

8.5 Rapporter

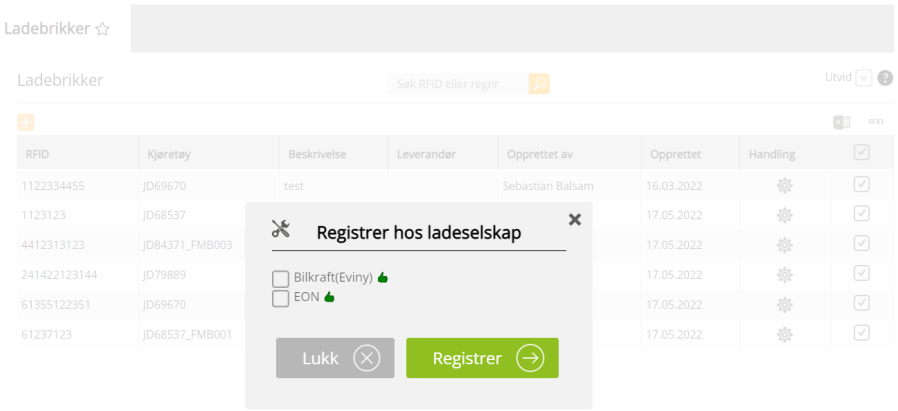
Som eksempel på en rapport skal vi se på tabellen for ladebrikker. Måten vi genererer en rapport har vi beskrevet i seksjonen Tagger.

I listing 2 kan man se måten vi legger inn felt i rapportene. Vi bruker deretter taggen vist i listing 1 til å vise feltene vi la inn i 'mainLo'.

8.6 Dialogvindu

Som eksempel på dialogvindu skal vi se på dialogvinduet som tar seg av ladebrikkeregistrering.

Når brukeren bruker massefunksjonen for ladebrikkeregistrering, blir funksjonen 'registerRFID' kalt. Denne funksjonen henter ut alle ladebrikker brukeren trykket på, og åpner dialogvinduet for registrering. Koden for dette er i listing 9. Figur 37 viser et eksempel på hvordan ladebrikke siden ser ut når brukeren åpner et dialogvindu.



Figur 37: Eksempel på dialogvindu

```
// Åpner dialogvind og sender med kjøretøy og rfidbrikker brukeren har valgt
function registerRFID() {
    // Henter rader valgt av brukeren
    var result = getChosenRows()
    var data = {}
    data.kjoretoy = result.kjoretoy
    data.rfidbrikker = result.rfidbrikker

    // Åpner dialogvindu ChargingProviderRegisterDialog og sender med data
    dialogLinkItemShow("${solutionUrl}/x/ChargingProviderRegisterDialog",{
        kjoretoy: data.kjoretoy,
        rfidbrikker: data.rfidbrikker,
    },);
}
```

Listing 9: kodesnutt for åpning av dialogvindu for ladebrikkeregistrering

I kodesnutten laster vi inn 'ChargingProviderRegisterDialog.java', som videre laster inn 'ChargingProviderRegisterDialog.jsp'. Innholdet i JSP filen blir vist som et dialogvindu i stedet for å fylle hele siden, fordi Java filen ChargingProviderRegisterDialog arver egenskaper fra 'PageFragmentController', i stedet for 'PageController'

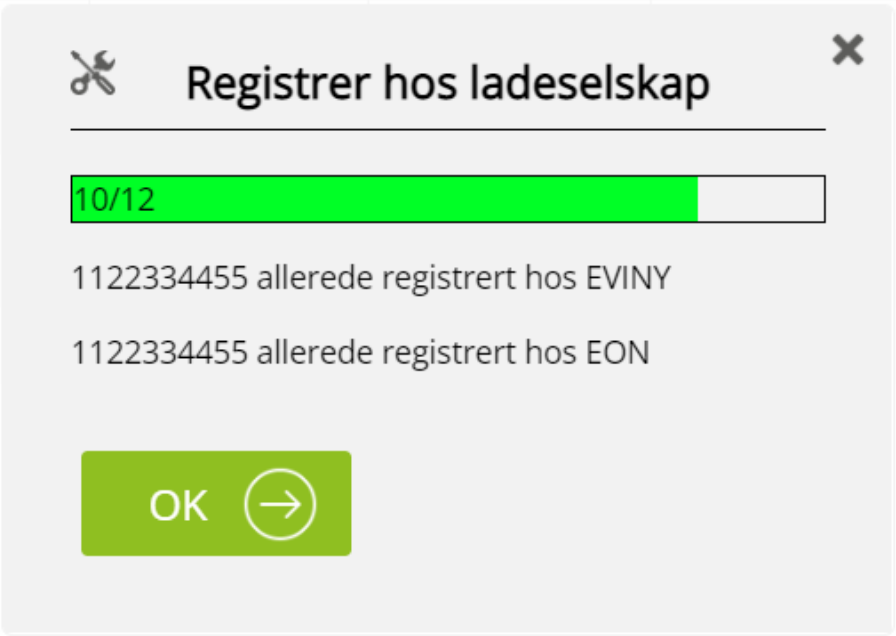
(se PageFragmentController eksempel). JSP filen er altså bare et lite fragment av hele siden.

```
1    public class ChargingProviderRegisterDialog extends PageFragmentController
```

Listing 10: PageFragmentController eksempel

8.7 Statusbar

Når brukeren registrer ladebrikker vises en statusbar som viser hvor mange ladebrikker som blir registrert. Om noen brikker allerede er registrert, får man opp hvilke brikker som allerede er registrert, og hvilket ladeselskap de er registrert hos. Se Dialogvindu med statusbar for eksempel.



Figur 38: Dialogvindu med statusbar

8.8 Strømforbruk

Den andre delen av oppgaven går ut på å hente strømforbruket når en elbil har ladet. Som nevnt i seksjon 5.3 bruker vi Enode for å hente strømforbrukte fra interne ladere. I den neste delen av rapporten skal vi gjøre rede for hvordan vi får tilgang til Enode sin API, hvilke endepunkter vi skal bruke, hvordan bilansvarlig registrerer en ny lader og hvordan vi henter strømforbruk når en bil er ferdig å lade.

8.8.1 Tilgang til Enode

Enode er ikke en åpen API som alle kan bruke. For å få tilgang måtte vi sende en forespørsel på e-post til utviklerene om å få tilsendt en klient id og et passord. En

må betale får å få tilgang til APIet, men de har og et testmiljø som kan brukes uten betaling. Det skal vi benytte oss av under utviklingen av programvaren. Testmiljøet er likt som produksjonsmiljøet, bortsett fra at dataene man får fra APIet er simulert.

Vi har fått tilgang til APIet, men får å kunne gjøre API-kall trenger vi et tilgangstoken. Det får vi med å bruke endepunktet som heter 'ClientAccessToken' i API dokumentasjonen.[15]. For a verifisere seg selv må man inkludere klient id og passordet i forespørselen. Vi bruker OkHttp og HttpClient til å få tilgangstoken fra Enode. For å unngå duplisering av kode har vi laget en funksjon som returnerer et nytt tilgangstoken når man kaller på funksjonen. Kodesnutten under viser API-kallet vi bruker for å få tilgangstoken.

```
1
2       OkHttpClient client = new OkHttpClient().newBuilder()
3               .build();
4           MediaType mediaType = MediaType.parse("application/x-www-form-urlencoded");
5           RequestBody body = RequestBody.create(mediaType, "grant_type=
    client_credentials");
6
7           Request request = new Request.Builder()
8                   .url("https://oauth." + enodeClass.getEnvironment() + ".enode.io/
    oauth2/token")
9                   .method("POST", body)
10                  .addHeader("Authorization", "Basic
    Y2I4MzM4MDEtNjdkMy00MzRkLThhNjktYWQyMzVlNDJjMWQ0OlZzLmpPUGJfLnRUUnVfdzF6cUlHe-
11                  Glwa35z")
12                  .addHeader("Content-Type", "application/x-www-form-urlencoded")
13                  .build();
14
15          Response response = null;
16
17          try {
18              response = client.newCall(request).execute();
19          } catch(IOException e) {
20              e.printStackTrace();
21              System.out.println("Enode http call to get acces token failed");
22          }
```

Listing 11: API-kall for tilgangstoken

Under ser vi JSON objektet man får som respons dersom kallet er vellykket. Det viktigste i responsen er "access_token", som er tokenet man bruker for å gjøre API-kall til andre endepunkter i APIet.

```
1   {
2       "access_token": "{YOUR_ACCESS_TOKEN}",
3       "expires_in": 3599,
4       "scope": "",
5       "token_type": "bearer"
6   }
```

8.8.2 API endepunkt i Enode

Enode sin API har mange forskjellige endepunkter ettersom de kommuniserer med flere ulike energienheter, og tilbyr mye forskjellig støtte til hver av dem. Her er en

[15]Enode. *Enode API (1.12.0)*. URL: https://docs.enode.io/api-reference (sjekket 15. feb. 2022).

oversikt over endepunktene vi bruker i implementasjonen vår.

Link user (POST)

Responsen fra 'Link user' endepunktet er url til en Enode nettside som blir brukt for å registrere en ny lader. Curl kommandoen under viser hvordan API-kallet til 'Link user' ser ut.

```
$ curl -X POST -d "vendorType=charger" -d "redirectUri=https://www.google.com" -H
   "Authorization: Bearer 1ooJa0IHs4ElMbJoT30AQSYSTZ65XTDYR_K_qnwh3Qc.
  T_MjBsJqk8VdTO90eDPdGnyejW0VihqB4z_DZr7BND0" https://enode-api.sandbox.enode.io
  /users/{user-id}/link
```

Listing 12: Link user

Som man ser bestemmer vi variablene 'vendorType' og 'redirectUri'. VendorType er hvilken type energienhet brukeren skal registrere. Siden vi skal få informasjon fra ladere, legger vi til parameteret 'charger'. RedirectUri er nettsiden brukeren skal bli omdirigert til etter en er ferdig med å registrere nye ladere. I lenken til endepunktet legger vi og til id på brukeren som laderen skal tilhøre.

Update Firehose Webhook (PUT)

'Update Firehose Webhook' blir brukt for å registrere en url Enode kan sende webhook notifikasjoner til. I forespørselskroppen sender man med url-en og en nøkkel som blir brukt til å verifisere at webhook notifikasjonen kommer fra Enode. Eksempelet under viser hvordan innholdet i kallet man sender ser ut. Vi bruker webhooks til å få beskjed om at en ny lading er registrert.

```
{
"secret": "OKvs1tAUQ69FOMBiWlt5XJSrruXMhWDiVbyrWaNm",
"url": "https://brainpower.co/enode-webhooks/firehose"
}
```

Get User Charging Statistics (GET)

'Get User Charging Statistics' blir brukt for å hente informasjon om strømforbruk etter at systemet får en webhook notifikasjon om at en ny lading er registrert.

```
$ curl -X GET -H "Authorization: Bearer Ax0mxNb2oIdXaCFlj8nyvm-
  kNoApCagPt69Mm2B8f38.fl3SmwGUE1jumogShS_AWCbE1MOPcRtl0OqlPrl6uLw" -H "Enode-
  User-Id: 32" https://enode-api.sandbox.enode.io/statistics/charging?startDate
  =2022-02-25&endDate=2022-02-26&type=charger&id=26c89fa7-2c6f-4842-b079-
  e43ba9e735b9
```

Listing 13: Get User Charging Statistics

I curl kommandoen over ser vi at etter 'Bearer' trenger man å legge til et tilgangstoken for å bli autentisert til å gjøre API-kall. Enode-User-Id er hvilken bruker vi skal hente ladinger for. Vi legger til fire variabler i url-en. 'startDate' og 'endDate' er intervallet vi skal hente nye ladinger fra, 'type' er hvilken type energienhet vi skal hente informasjon fra, og 'id' er id-en på laderen vi skal hente strømforbruk fra.

8.8.3 Registrere en lader

For å registrere en lader bruker vi endepunktet 'Link User' beskrevet i seksjon 8.8.2. Når brukeren trykker på knappen i standardoppsett som heter 'Registrer bruker hos Enode' gjør vi API-kallet for å hente lenken til Enode-siden hvor man registrerer ladere. Når bilansvarlig har fullført prosessen med å registrere en lader hos Enode blir nettsiden omdirigert tilbake til CarAdmin.

For å lage knappen bruker vi en ferdiglaget tag fra CarAdmin som vi ser i eksempelet under. Når brukeren trykker på knappen blir funksjonen 'openEnodeLink' kjørt.

```
<etc:FilledButton
        id="Button1"
        text="Registrer lader hos Enode"
        cssClass="btn-green-next"
        action="openEnodeLink();" />
```

I listing 14 ser vi JavaScript funksjonen 'openEnodeLink'. Dens jobb er å åpne Enode-nettsiden man bruker for å registrere ladere. For å gjøre det, utfører funksjonen et AJAX-kall til klassen 'ChargingEnodeLink' som vi skriver mer om senere i denne seksjonen. I funksjonen henter vi først navnet på kunden sin database som vi bruker som bruker-id for laderen, siden hver kunde i CarAdmin har sin egen databaseløsning med et unikt navn. Så åpner vi en blank nettside i brukeren sin nettleser. Grunnen for at vi åpner en blank side og fyller den med rett innhold senere er for å unngå at nettleseren skal tro at det er et pop-up vindu og blokkere siden. Fordi 'openEnodeLink' gjør et ajax-kall for å hente lenken, tror nettleseren at det er kode som utløser at en ny side skal åpnes, og ikke en brukerhandling. Dermed må siden åpnes før ajax-kallet. Responsen fra ajax-kallet er en statusmelding som enten er 'success' eller 'error'. Hvis statusen er 'success' får man og tilsendt lenken som skal åpnes. Hvis statusen er 'error' får man tilsendt en melding om at programmet ikke får tilgang til Enode. Denne meldingen blir vist på skjermen i det språket brukeren har valgt at nettsiden skal vises på.

```
async function openEnodeLink() {


        document.getElementById("message").innerHTML = "";

        var data = {};
        data.customer_id = '${solutionDb}'; //Henter navnet på databasen

        var win = window.open('', "_blank"); //Åpner et blankt vindu

        //Gjør et post kall til ajax fil
        $.ajax({
            url: "/ChargingEnodeLink.ajax", //Url til ajax fil
            type: "POST",
            dataType: "json",
            data: data,
            success: function(data) {
                if(!data || data.error) {

                    //Beskjed hvis noe går galt
                    document.getElementById("message").innerHTML = "Kunne ikke
    koble til enode. Ta kontakt hvis " +"problemet vedvarer.";

                    win.close(); //Lukker vinduet

                } else {


                    const status = data.status; //Henter status på responsen
                    if(status !== "error"){     //Hvis det ikke er en error
                        const link = data.urlEnode;  //Hent lenken

                        //Fyll det blanke vinudet med lenken
                        win.location = link.toString();

                    }else{ //Hvis det er en error
                        win.close(); //Lukker vinduet

                        //Viser en melding om at noe gikk galt
                        const message = data.message;
                        document.getElementById("message").innerHTML = message.
    toString();

                    }
                }
            }
        });
    }
```

Listing 14: Funksjon som åpner nettside for å registrer en lader

'ChargingEnodeLink' er en ajax-klasse med funksjoner som henter lenken til nettsiden hvor man registrerer ladere hos Enode. Vi bruker endepunktet 'Link User' som nevnt før. Etter vi har gjort API-kallet med OkHttp validerer vi om vi fikk tilbake en lenke, eller om noe gikk galt. Før vi sender json objektet med status og innhold tilbake til funksjonen 'openEnodeLink' bygger vi responsen som vist under.

```
//Bestemmer nøkkel og innhold basert på om man fikk en error eller en link fra
    funksjonen.
        obj = Objects.equals(enodePair.a, "success") ? obj.put("urlEnode",
    enodePair.b) : obj.put("message", enodePair.b);
        obj.put("status", enodePair.a);
        obj.write(response.getWriter()); //Sender objektet med info til jsp filen.
        response.getWriter().flush();
```

Listing 15: Funksjon som åpner nettside for å registrer en lader

8.8.4 Hente strømforbruk

Det eneste bilansvarlig må gjøre for å få inn strømforbruk er å registrere en lader. Derfor skjer hele prosessen med å hente strømforbruk uten interaksjon fra et menneske. Først skal vi skrive om hvordan vi får beskjed om at en ny lading er registrert, deretter skrive om hvordan vi henter den nye ladingen.

Verifisere webhook

Systemet hører etter en webhook notifikasjon fra Enode gjennom Restlet funksjoner. Det første programmet gjør når den får inn en webhook er å verifisere at webhooken er sendt fra Enode. I headeren til den innkommende webhooken er det en signatur som er en HMAC[16]. Signaturen er generert ved å hashe innholdet i webhooken med hemmeligheten vi selv sendte inn som nøkkel i hashen ved hjelp av en SHA-1 algoritme. Det vil si at for å verifisere at webhooken er fra Enode, så genererer vi en hash av webhookens innhold ved å bruke samme algoritme, og samme nøkkel. Hvis hashene er like kan vi verifisere at webhooken er fra Enode. I eksempel 16 ser man hvordan vi verifiserer webhooks. Funksjonen vi bruker for å generere hashen har vi hentet fra et eksempel i GitHub[17].

```
1  /**
2      *
3      * @param payload
4      * @return - bool som forteller om kallet ble verifisert eller ikke.
5      */
6     private boolean verifyEnodeSignature(JSONArray payload) {
7
8         //Henter signatur fra header
9         var enodeSignature = getRequest().getHeaders().getFirstValue("x-enode-
      signature");
10
11        String hmac = null;
12        try {
13
14        //Hasher bodyen
15        hmac = calculateRFC2104HMAC(payload.toString(), key);
16
17         } catch(SignatureException | NoSuchAlgorithmException |
      InvalidKeyException e) {
18            e.printStackTrace();
19            System.out.println("Could not create hmac");
20        }
21
22        hmac = "sha1=" + hmac;
23
24        return hmac.equals(enodeSignature); //returnerer true eller false
25    }
```

Listing 16: Verifisering av webhook fra Enode

Filtrere webhook innhold

I tiden når denne oppgaven ble skrevet har Enode bare en prekonfigurert webhook som rapporterer om alle hendelsene i systemet. Det betyr at når vi får en notifikasjon må vi selv filtrere gjennom innholdet for å vite om det er en ny lading som har blitt registrert. Under ser vi eksempel på hvordan et objekt i JSON-arrayen fra en webhook ser ut. Det første vi gjør er å se på 'event' feltet. Hvis den inneholder

[16] GeeksForGeeks. *hmac – Keyed-Hashing for Message Authentication*. URL: https://www.geeksforgeeks.org/hmac-keyed-hashing-message-authentication/ (sjekket 16. mai 2022).

[17] Takanori Ishikawa. *Java Sample Code for Calculating HMAC-SHA1 Signatures*. URL: https://gist.github.com/ishikawa/88599?permalink_comment_id=2029198 (sjekket 16. mai 2022).

'user:charger:updated' betyr det at noe med en lader er oppdatert, og det kan hende en lading er registrert. Hvis 'event' sier noe annet stopper vi letingen. 'updatedFields' er en liste som forteller hvilket felt som er oppdatert, og hvorfor denne notifikasjonen ble sendt. Dersom listen inneholder 'chargeStateIsPluggedIn' og 'isPluggedIn' er false, betyr det at en lader har blitt plugget ut av en bil, og en ny lading har blitt registrert. Vi lager en liste med alle objektene som forteller at en ny lading er registrert, og sender listen til en funksjon som henter ladinger fra Enode. Hvert objekt i listen vil inneholde ulike ladere.

```
[
{
        "createdAt": "2022-03-30T10:56:04.401Z",
        "event": "user:charger:updated",
        "user": {
            "id": "autodb"
        },
        "charger": {
            "id": "447146d9-1d4b-4abd-92d4-458b7a4e4298",
            "lastSeen": "2022-04-01T10:56:04.351Z",
            "isReachable": true,
            "chargeState": {
                "isPluggedIn": false,
                "isCharging": false,
                "chargeRate": 0,
                "lastUpdated": "2022-04-25T11:43:36.067Z"
            },
            "information": {
                "id": "447146d9-1d4b-4abd-92d4-458b7a4e4298",
                "brand": "Easee",
                "model": "ZCH041944",
                "year": 2022
            }
        },
        "updatedFields": [
            "lastSeen",
            "updatedAt",
            "chargeStateIsPluggedIn"
        ]
}
]
```

Hente ladinger

Etter vi har filtrert innholdet i webhooken er det på tide å hente ladinger fra Enode. For det bruker vi 'Get User Charging Statistics' endepunktet. Funksjonen som tar imot listen med JSON-objekter vi nevnte over heter 'findChargingData'. Funksjonen itererer gjennom hvert objekt og henter ut dataen vi trenger for å hente strømforbruk, og som vi ønsker å lagre i databasen til CarAdmin. Vi finner lader-id, merke til laderen, og bruker-id som laderen tilhører. Når vi har dataen vi trenger gjør vi et kall til funksjonen 'getChargingData' som henter informasjon om ladingen fra Enode, og returnerern json med informasjonen som en string. API-kallet i 'getChargingData' blir vist i listing 17. Her ser man at vi sender med to datoer med

tidsintervallet vi skal hente ladinger fra, og hvilke ladere vi skal hente strømforbruket til. Datoene er fra tiden ladingen ble registrert, til API-kallet blir gjort.

```
1
2  OkHttpClient client = new OkHttpClient().newBuilder()
3                  .build();
4          Request request = new Request.Builder()
5                  .url("https://enode-api." + enodeClass.getEnvironment() + ".enode.
       io/statistics/charging?startDate=" + date + "&endDate=" + now + "&type=charger&
       id=" + chargerId)
6                  .method("GET", null)
7                  .addHeader("Authorization", "Bearer " + accessToken)
8                  .addHeader("Enode-User-Id", customerSolution)
9                  .build();
10
11         Response response = null;
12         try {
13             response = client.newCall(request).execute();
14         } catch(IOException e) {
15             e.printStackTrace();
16         }
```

Listing 17: Henter ladinger fra Enode

Når får tilbake tilbake resultatet fra 'getChargingData' går vi gjennom json-stringen og lager et SQL-query for å lagre dataen i databasen.

9 Testing og Kvalitetssikring

Testingen vår består av funksjonelle tester og brukertester. Vi bruker funksjonelle tester for å sjekke at alle funksjoner returnerer det vi forventer at de skal returnere. Denne grundige testingen av funksjonene våre fører til at programmet alltid oppfører seg slik vi forventer.

Desverre har CarAdmin sitt system en ulempe når det gjelder testing av våre moduler. Vi bruker Docker til å kjøre systemet lokalt. Dette er ikke måten ETC vanligvis kjører programmet sitt, men det er slik de valgte å sette det opp til oss. Dette betyr at noen deler av programmet, inkludert testing, kjører litt annerledes for ETC enn for oss. Etter mye undersøkelse, og samtale med utviklere hos ETC, fant vi ut at det ville være umulig å kjøre tester på deler av programmet som krever tilgang til en database, med mindre vi setter opp en egen sql server lokalt. ETC rådet oss derfor til å bruke 'hard kodet' data i testene våre, som ble løsningen vår.

9.1 API Kommunikasjon

For testing av APIer har vi prøvd å lage tester som replikerer måten APIene blir brukt i programmet. Å ha tester som tester alle API endepunkt forsikrer at dersom et endepunkt forandrer seg vil vi lett kunne oppdage det ved å kjøre testene. Vi valgte å dele opp seksjonen inn i én del for hver API.

9.1.1 Bilkraft

Vi bruker og tester 4 endepunkter fra Bilkraft sin API:

- Innlogging
- Hente ut ladebrikker
- Legge til ladebrikker
- Slette ladebrikker

Under er et eksempel på hvordan vi tester det å legge til, og slette en ladebrikke hos Bilkraft.

```
1     @Test
2     void testBilkraftPostAndDeleteLadebrikker() {
3         try {
4             // Henter token, som blir brukt til å legge til, og slette ladebrikke.
5             String token = parser.bilkraftLogin("46919299", "bachelor12345!");
6             // Kaller på funksjonen som legger til ladebrikke.
7             testBilkraftPostLadebrikke(token);
8             // Kaller på funksjonen som sletter ladebrikke.
9             // Sletter samme ladebrikke som nettop ble lagt inn.
10            testBilkraftDeleteLadebrikker(token);
11        } catch(IOException e) {
12            e.printStackTrace();
13        }
```

```
14
15      }
16
17      void testBilkraftPostLadebrikke(String token) {
18          try {
19              // Sender inn rfidnummer og bilnummer til Bilkraft, for å
20              // registrere en ny ladebrikke knyttet til bilen.
21              String testRes = parser.bilkraftPost("123444441", "FT 16778", token);
22              // Sjekker at registrering av ladebrikke ikke feilet.
23              assertNotEquals("FAIL", testRes);
24          } catch(IOException e) {
25              e.printStackTrace();
26          }
27      }
28
29      void testBilkraftDeleteLadebrikker(String token) {
30          try {
31              // Sender inn rfidnummer til brikken vi ønsker å slette.
32              String testRes = parser.bilkraftDelete("123444441", token);
33              // Sjekker om sletting var suksessfull.
34              assertEquals("success", testRes);
35          } catch(IOException e) {
36              e.printStackTrace();
37          }
38      }
```

Listing 18: Test hvor vi legger inn og sletter en ladebrikke hos Bilkraft.

Vi tester at endepunktene fungerer, ved å kalle på funksjonene vi bruker i programmet, hente resultatet, og bruke 'assertEquals' eller 'assertNotEquals' for å sjekke at vi fikk riktig resultat.

9.1.2 EON

Vi bruker og tester 3 endepunkter fra EON sin API:

- Innlogging
- Legge til ladebrikker
- Slette ladebrikker

EON sine endepunkter blir testet på samme måte som Bilkraft, og mange av testene er nesten identiske. Den største forskjellen er at vi ikke henter ut ladebrikker, og derfor ikke trenger å teste det.

9.1.3 Strømforbruk

For funksjonalitet som handler om å hente strømforbruk testet vi både API endepunkt, og andre funksjoner. Det første vi tester er at det er mulig å hente strømforbruk fra Enode vist i listing 19. Vi bruker funksjonen 'getChargingData' som utfører kallet til Enode.

```
1       @Test
2       void testGetChargingData(){
3           GetChargingDataEnode charging = new GetChargingDataEnode();
4
```

```
5         try{
6
7             String token = charging.getAccesToken();
8             String json = charging.getChargingData("autodb", "447146d9-1d4b-4abd-92
      d4-458b7a4e4298","2022-04-04", token);
9             assertNotNull(json);
10
11        }catch(Exception e){
12            e.printStackTrace();
13        }
14    }
```

Listing 19: Test for å hente strømforbruk

Vi tester og at systemet klarer å hente et tilgangstoken fra Enode. Måten vi gjør det på er veldig likt eksempelet for å hente strømforbruk, bortsett fra at vi kaller på en annen funksjon. Disse testene viser at kommunikasjonen mellom CarAdmin og Enode virker.

De andre testene vi utfører for strømforbruket er funksjonstester som ikke utfører API-kall. Målet med de testene er å sjekke at funksjonene returnerer det vi forventer etter at de er ferdig, og dermed vet vi at logikken i funksjonene er korrekt. Vi får og testet at funksjonene returnerer det samme hver gang de får lik parameter.

9.2 API begrensninger

Det er viktig å finne ut hvor mange kall i minuttet man kan gjøre hos APIene man bruker. I vårt tilfelle fantes det ikke dokumentasjon på nettet, så vi valgte å teste det på egenhånd. Vi Lagde en tester for både EON og Bilkraft, som legger inn 100 nye ladebrikker og deretter sletter de samme ladebrikkene. Denne testen vil simulere en bruker som har fått inn mange nye ladebrikker og biler som skal bli registrert. Vi forventer ikke at mange kunder skal registrere 100 ladebrikker på en gang, men å kjøre testen med et høyt antall ladebrikker gir oss en god forståelse for begrensninger til APIen. Under kan man se koden for dette.

```
1     @Test
2     void testBilkraftPostAndDelete100Ladebrikker() {
3         try {
4             // Henter token, som vi bruker til å legge til ny ladebrikke.
5             String token = parser.bilkraftLogin("46919299","bachelor12345!");
6             String rfidnummer = "";
7
8             for(int i = 0; i < 100; i++) {
9                 rfidnummer = "123444441";
10                rfidnummer += i;
11                // Kaller på funksjonen som legger til ladebrikke.
12                testBilkraftPostLadebrikke(token, rfidnummer);
13            }
14
15            for(int i = 0; i < 100; i++) {
16                rfidnummer = "123444441";
17                rfidnummer += i;
18                // Kaller på funksjonen som sletter ladebrikke.
19                // Sletter samme ladebrikker som nettop ble lagt inn.
20                testBilkraftDeleteLadebrikker(token, rfidnummer);
21            }
22
23        } catch(IOException e) {
24            e.printStackTrace();
25        }
```

```
26     }
```

Listing 20: Tester å registrere og slette 100 ladebrikker

For denne testen logger vi inn én gang, og henter ut en validert token. Vi bruker deretter denne validerte token til å legge til 100 nye ladebrikker i en for-loop. Deretter bruker vi en annen for-loop til å slette alle ladebrikker etter de har blitt lagt inn.

Resultatet fra testen var at både EON og Bilkraft lar brukere registrere og slette 100 ladebrikker på kort tid uten problemer. Vi fant også ut at EON sin API er tregere, og brukte rundt 3 minutter og 59 sekunder på testen, mens Bilkraft brukte bare 56 sekunder. Dette vil i praksis bety at brukeren vil merke en forsinkelse mens programmet kommuniserer med EON sin API, sammenliknet med Bilkraft sin API.

9.3 Brukertester

9.3.1 Test brukere

Vi valgte å bare gjennomføre tester med NRK og ETC, da vår løsning ikke vil bli brukt av noen andre enn ansatte hos ETC eller kundene deres. ETC gjennomfører et kurs og demonstrerer bruk av deres programvare til nye kunder, og vi har derfor konkludert at vår løsning vil hovedsaklig bli brukt av brukere som er godt kjent med CarAdmin fra før. Vi gjennomførte ikke tester på venner eller familie. Antell tester vi gjennomførte var tre.

Selv om vi gjennomførte et lavt antall brukertester, mener vi at vi fikk mye ut av dem. Vi som har utviklet løsningen kan ofte bli blinde til forvirrende design, da vi er så vandt til å bruke løsningen. Vi trenger ikke lete gjennom grensesnittet for å finne en funksjon eller et ikon, siden vi vet hvor alt ligger. Derfor var det nyttig å få innspill fra nye brukere.

9.3.2 Gjennomføring

I brukerundersøkelsene gav vi test brukerene 7 oppgaver:

1. Registrer et ladeselskap.
2. Registrer en ny ladebrikke i CarAdmin.
3. Knytt en bil til rfid brikken
4. Registrer en ladebrikke hos et eller flere ladeselskap.
5. Registrer flere ladebrikker hos et eller flere ladeselskap.
6. Slett en ladebrikke fra et ladeselskap.
7. Registrer en lader hos Enode.

Basert på hvor raskt brukeren løste oppgaven og hvor store problemer de hadde, gav vi dem poeng på hver oppgave. Alle brukerenes poengsummer blir lagt sammen per kategori og vi kan dermed se hvilke oppgaver som generelt var vanskelige. Jo lavere prosent oppgaven fikk, jo vanskeligere var den.

	Oppg. 1	**Oppg. 2**	**Oppg. 3**	**Oppg. 4**	**Oppg. 5**	**Oppg. 6**	**Oppg. 7**
Bruker 1	1.0	1.0	0.4	0.4	0.6	1.0	1.0
Bruker 2	1.0	1.0	1.0	0.7	0.8	0.9	1.0
Bruker 3	1.0	1.0	1.0	1.0	0.7	0.8	1.0
TSR(%)	100%	100%	80%	70%	70%	90%	100%

Tabell 13: Brukertest TSR tabell

Som vi kan se i tabellen fikk alle brukerene full poengsum på oppgave 1 og 2. Dette betyr at måten man registrerer et nytt ladeselskap og en ny ladebrikke i CarAdmin er lett, selv om brukeren ikke har sett løsningen vår før.
Oppgave 3 var det én deltager som slet med å finne knappen for å knytte en bil til en RFID-brikke, siden den lå inne i et detalj vindu man må først åpne. Dette designvalget er i tråd med resten av CarAdmin, og deltageren sa selv at de hadde funnet knappen lett på egenhånd. De følte på litt press fra brukerundersøkelsen, som kan ha påvirket resultatet til denne deltageren.
Av alle oppgavene var det nummer 4 og 5 deltagerene hadde mest problemer med. To av deltagerene slet med å finne den riktige knappen, og de samme deltagerne slet med å finne massefunksjonsknappen.
Oppgave 6 hadde ikke like lav TSR som oppgave 4 og 5, men allikevel var det noen problemer vi oppdaget under brukertesten. Noen av deltagerene slet med å finne søppelbøtteikonet. Dette viste seg å være fordi størrelsen på ikonet er mindre enn ikonene for de andre funksjonene, og plasseringen er annerledes. Oppgave 7 var det ingen som slet med. Alle deltagerene klarte å registrere en ny lader hos Enode uten problemer.

Etter brukeren hadde gjennomført alle oppgavene, stilte vi spørsmål for å få tilbakemeldinger på løsningen. Vi stilte til sammen 5 spørsmål, med oppfølgingspørsmål der vi følte det var nødvendig. Spørsmålene vi stilte var:

- **Hva synes du er bra med nettsiden?**
  Vi fikk mange positive tilbakemeldinger om løsningen vår. To av deltagerene sa at løsningen passer veldig godt inn i CarAdmin, og at den hadde fulgt designprinsippene så godt at den var enkel å sette seg inn i. Én deltager mente at ladebrikketabellen vi hadde laget var bedre enn drivstofftabellen, som er tabellen vi brukte som utgangspunkt.

- **Hva kunne vært bedre?**
  Tilbakemeldingen vi fikk passet godt med TSR vi regnet ut i tabell 13. Vi fikk blandt annet tilbakemelding om at søppelbøtteikonet var for lite, og plasseringen forvirrende. Det var også ønske om en måte å slette flere ladebrikker på en gang, så man slipper å slette én om gangen. To av deltagerene foreslo en massefunksjon til sletting. Flere av deltagerene mente også at funksjonsnavnet

'Ladebrikke reg.' var forvirrende, da det kunne forveksles med knappen 'Ny ladebrikke'. Forslag til navnforandring var 'knytt ladebrikke'.

- **Hva synes du om måten man registrerer innloggingsinfo for ladeselskap?**
  Alle deltagere syntes måten man registrerer innloggingsinfo var enkel og bra.
- **Hva synes du om måten man registrerer en ladebrikke hos ladeselskap?**
  Alle deltagere syntes måten man registrerer en ladebrikke hos ladeselskap var enkel og bra.
- **Hva synes du om oversikten over ladinger?**
  For oversikten over ladinger fikk vi ganske mange forskjellige tilbakemeldinger. Én av deltagerene ønsket å kunne sortere i tabellen basert på lokasjoner, og hvilke ladere som er i bruk og ikke. Deltager 2 ville ha et felt i tabellen med et navn til hver lader bestemt av brukeren, så det er enklere å vite hvilken lader det er snakk om. Den siste deltageren ville kunne sortere etter kjøretøy, og at man skal kunne få strøm brukt per bil i stedet for per lader.

Under brukerundersøkelsen fant vi også en feil i programmet, hvor valget om å registrere en ladebrikke hos et ladeselskap kom opp, selv om innloggingsinformasjon var feil.

9.3.3 Demonstrasjon for ETC

Da vi mente at løsningen vår var i nærheten av ferdig holdt vi en demonstrasjon for ETC hvor vi viste frem hele løsningen. Formålet med demonstrasjonen var å få tips til forbedringer og se om designet holdt høy nok standard. Vi fikk mange gode tilbakemeldinger og små forandringer vi kunne implementere for å øke brukervennligheten og den generelle kvaliteten på løsningen. Noen forandringer ETC foreslo var:

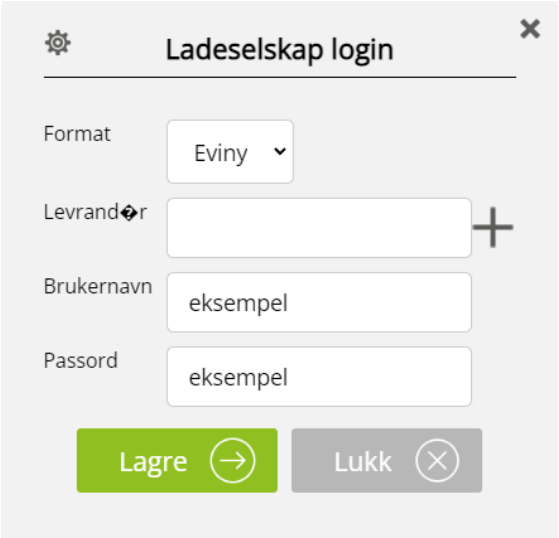
- Legge til passord sensurering for passord feltet i ladeselskap innlogging.
- Endre navn på knapp fra 'lagre' til 'registrer', og bytte om på knappene så 'lagre' knappen er på høyre side.
- Oppdater ladebrikketabellens ladeselskap felt etter dialogvindu lukkes.
- Flytte knapp for laderregistrering til standardoppsett.

Resten av forslagene er i vedlegg F Referat fra møter med ETC.

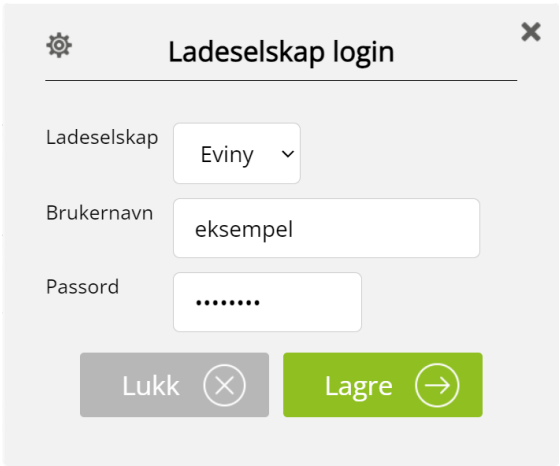
9.3.4 Resultat og endringer etter brukertester

Vi fikk mange gode tilbakemeldinger fra brukertestene og demonstrasjonen vi holdt for ETC. Vi implementerte mange av design forandringene og bugs brukertestene avdekket. Her er forandringene vi gjorde:

I dialogvinduet for innlogging til ladeselskap fikset vi på navnene til feltene, fjernet feltet for leverandører, sensurerte passord, og byttet posisjon på knappene for lagring og kansellering. I figurene 39 og 40 vises før og etter bilde.

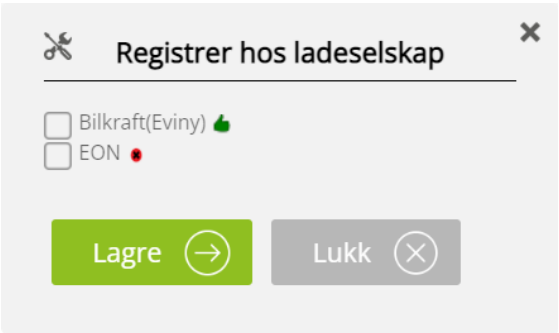


Figur 39: Dialogvindu for innlogging til ladeselskap før forandringer

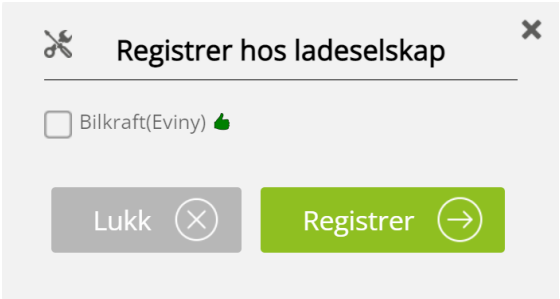


Figur 40: Dialogvindu for innlogging til ladeselskap etter forandringer

Brukertestene avdekket en svakhet i systemet, hvor brukeren fikk valget om å registrere ladebrikker opp mot selskap hvor innlogginsinformasjon var feil. Vi løste dette ved å unngå å legge til ladeselskap i dialogvinduet om det ikke hadde riktig innloggingsinformasjon. Dette kan også gjøre dialogvinduet mindre forvirrende for brukere, da bare gyldige valg vises. Figur 41 og 42 viser før og etter bilde.

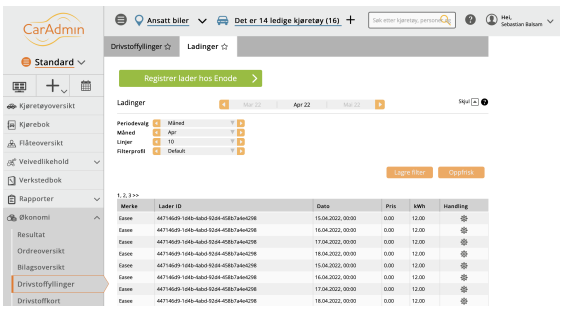


Figur 41: Dialogvindu for registrering av ladebrikker hos ladeselskap før forandringer

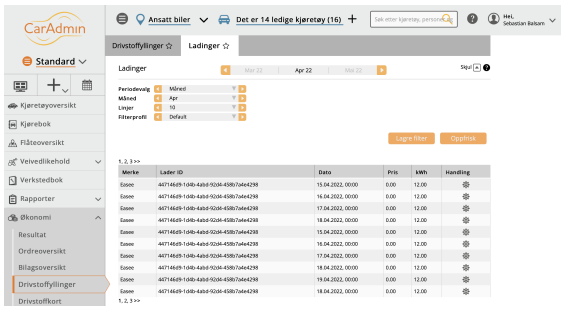


Figur 42: Dialogvindu for registrering av ladebrikker hos ladeselskap etter forandringer

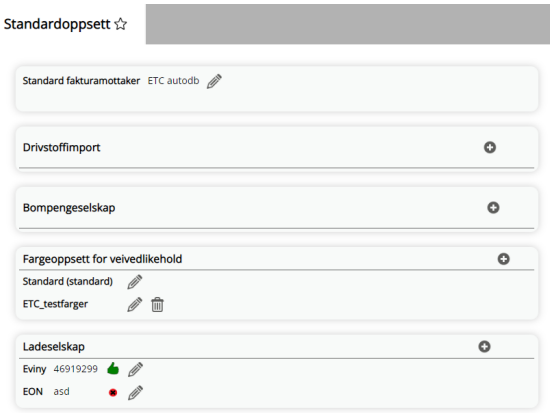
I tabellen for ladinger flyttet vi knappen for registrering av lader hos enode. Knappens posisjon stod ikke i stil med resten av CarAdmin sitt design, og vi fikk tips fra ETC om å legge den i Standardoppsett. Figur 43 og 44 viser før/etter i siden for ladinger, og figur 45 46 viser før og etter i standardoppsett.



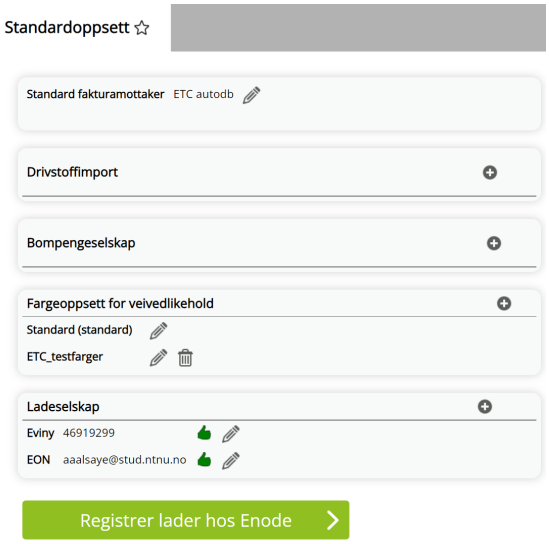
Figur 43: Tabell over ladinger før forandringer



Figur 44: Tabell over ladinger etter forandringer



Figur 45: Standardoppsett utseende før forandringer



Figur 46: Standardoppsett utseende etter forandringer

10 Avslutning

I dette avsluttende kapittelet skal vi reflektere rundt hva vi har oppnådd i prosjektet, kritikk til oppgaven og videre arbeid. Vi skal og oppsummere og konkludere arbeidet vi har gjennomført.

10.1 Resultater

Her oppsummerer vi resultatene vi har oppnådd i prosjektet og ser tilbake på læringsmål, effektmål og resultatmål. Vi er fornøyde med målene vi har oppnådd til tross for mindre tilgang på APIer enn vi hadde forventet.

10.1.1 Læringsmål

Vi skal se tilbake til læringsmål vi satt i starten av prosjektet og diskutere hvorvidt vi har oppnådd dem.

- **Lære om nye teknologier.**
  - Java-basert web utvikling.
  - Databaser med MariaDB.

  Gjennom prosjektet har vi fått god erfaring med java-basert web utvikling, og MariaDB. Begge deler ble brukt mye under utviklingen.
- **Lære å utvikle en mikrotjeneste for et eksisterende system.**
  Mikrotjenestene vi utviklet var uthenting av ladinger og registrering av ladebrikker. Å implementere mikrotjenestene som sider i CarAdmin viste seg å være en utfordring, da vi måtte sette oss inn i hvordan store deler av rammeverket funket. Det var tidskrevende og utfordrene, men vi mener løsningene integreres godt inn i CarAdmin. Brukerundersøkelsene og demonstrasjonene støtter også opp under dette.
- **Få mer kunnskap om teknologi rundt lading av elbil.**
  Gjennom utforsking av ladere, ladebrikker, og ISO standarder har vi lært mye rundt lading av elbil.
- **Lære å utvikle et system som kommuniserer med ulike ladestasjoner.**
  En stor del av arbeidet i starten gikk til undersøking av mulige løsninger og teknologier innenfor kommunisering med ladestasjoner. Vi endte opp med å bruke Enode som samler mange interne ladeselskaper og gir oss tilgang til denne informasjonen gjennom en API. Dermed ble læringsmålet oppfylt.
- **Lære mer om sikkerhet rundt behandling av sensitiv informasjon.**
  Løsningen vår behandler sensitiv data som brukernavn og passord til ladeselskap. Derfor er det viktig at vi krypterer dataen, og lagrer den sikkert. I vårt

tilfelle hadde ETC allerede laget en klasse vi kunne bruke til å kryptere data, og vi slapp derfor undersøke forskjellige krypteringsalgoritmer. Vi lærte også om sikkerhetsprinsipper, som å sensurere passordfelt for økt konfidensialitet.

- **Lære mer om utvikling av brukervennlige grensesnitt.**
  Et av kravene til ETC var at løsningen vår skulle følge designprinsippene og strukturen til resten av CarAdmin. Gjennom opplæring, testing, og demonstrasjoner for ETC mener vi at vi har skapt et brukervennlig grensesnitt som følger den røde tråden i CarAdmin.

- **Lære om samarbeid i prosjektgruppe.**
  Gjennom prosjektet har vi jobbet godt sammen. Vi har fulgt gruppeavtalen, som har hjulpet oss til å arbeide jevnt og holde god gruppemoral. Vi har også jobbet sammen på tidligere prosjekter, og vet at vi fungerer godt som gruppe.

- **Lære å kontakte relevante personer i næringslivet for å undersøke problemer og muligheter.**
  Før vi kontaktet ulike ladeselskaper begynte vi med å undersøke relevante løsninger i markedet. Denne delen var viktig for å ha en forståelse rundt stoffet før vi kontaktet ladeselskapene. Vi konstruerte manus, snakkepunkter og spørsmål for samtalene vi hadde. Dette hjalp med å få tilgang til folk som ligger høyere i selskapet som kunne en del om stoffet. Vi har derfor fått mye erfaring med å kontakte personer i næringslivet.

10.1.2 Effektmål

I denne seksjonen skal vi se på effektmålene vi hadde satt for prosjektet, og hvorvidt vi nådde dem.

- **Mer fornøyde kunder/brukere av CarAdmin ved å spare tid og ressurser.**
  Vi ser for oss at brukere vil være fornøyde ettersom løsningen vår sparer på tid og ressurser. Tilbakemeldinger fra brukertester indikerte dette for oss.

- **Kunde-binding, hjelpe kunder med enda et problem slik at de velger CarAdmin overfor andre konkurrenter.**
  Systemet vi har utviklet hjelper brukeren med utfordriger rundt å registrere ladebrikker, og ha oversikt over strømforbruk. I tilleg følger vi designprinsippene til CarAdmin slik at det blir lett for brukeren å lære. Dette fører til sterk kunde-binding ettersom et ettertraktet system blir utviklet i et enda mer helhetlig bilstyringssystem.

- **Å gi kunden bedre oversikt over kostnader, som igjen gir mer fornøyde kunder.**
  Ettersom vi måtte utvikle innhenting av strømforbruk for ladestasjoner, så vil kunden ha en generell oversikt over strømforbruken for sine private ladestasjoner. Det er også mulig å filtrere resultatene for å ha oversikt over ladesesjoner over en spesifikk periode.

- **Et produkt som trekker til seg flere kunder ved å være attraktiv for elbilmarkedet.**
  Ideen fra starten av er veldig attraktiv for elbil markedet og at vi har en implementasjon for å vise potensiale for ideen er oppsiktsvekkende. Dette kan tiltrekke seg flere kunder.

10.1.3 Resultatmål

I denne seksjonen skal vi se på resultatmålene vi hadde satt for prosjektet, og hvorvidt vi nådde dem.

- **Et system for automatisk registrering av ladebrikker hos ulike ladeselskaper gjennom CarAdmin sitt brukergrensesnitt.**
  Vi har utviklet et system som kan registrere en mengde ladebrikker samtidig hos ulike ladeselskaper gjennom grensesnittet til CarAdmin.
- **En samlet oversikt over strømforbruk i CarAdmin sitt brukergrensesnitt.**
  Vi har utviklet et system som tilbyr en samlet oversikt over strømforbruken til private ladere med en maks 10 minutters forsinkelse på innhenting av ladesesjonsdata.
- **En CarAdmin modul som kan ta i bruk ladestasjonselskapenes framtidige APIs.**
  Systemet kan enkelt adoptere nye APIs dersom ladeselskaper kommer med de i framtiden.

10.2 Alternative løsninger og valg

I denne seksjonen skal vi diskutere alternative løsninger og valg vi kunne ha gjort i prosjektet.

10.2.1 Simulering av API

Simulering av APIer var en alternativ løsning vi vurderte sterkt. Vi hadde store problemer med å få tak i API-er, og brukte mye tid på å kontakte forskjellige ladeselskap. Samtidig ville ikke simulering gi ETC et fungerende produkt slik som de ønsket. Derfor var simulering av kun en aktuell løsning dersom vi ikke hadde noen andre alternativer.

10.2.2 HTML Parsing

På grunn av mangel av API-er og ønsket om et fungerende produkt fra ETC, virket det som vi ble nødt til å bruke HTML parsing. Løsningen ville da krevet mye

vedlikehold, da ladeselskapenes nettsider kan forandres uten forvarsel, og vi måtte ha laget egen kode til å parse gjennom hver enkelt nettside. Dette ville vært en stor jobb, og da vi fant API-endepunkt i ladeselskapenes nettsider, valgte vi å ikke bruke HTML-parsing ettersom det ikke var nødvendig lengre.

10.2.3 Valg av bruker-id hos Enode

Som nevnt har CarAdmin en egen database for alle kundene deres. Det spilte inn når vi valgte å bruke navnet til kunden sin database som bruker-id når vi registrerer en lader i Enode. Det var ikke noe grunn for å bruke en bruker sin id, ettersom ladere ikke tilhører en spesifikk bruker, men til bedriften. Fordelen med å bruke databasenavnet som id er at det gjorde det lett å vite hvilken database man skal lagre en ny lading i.

10.2.4 Velge når vi henter ladinger

Vi diskuterte når vi skal hente ladinger fra Enode sin API. De tre alternativene var at brukeren trykker på en knapp for å hente ladinger, at ladinger ble hentet på et bestemt tidspunkt eller at vi henter ladinger når vi får en webhook notifikasjon. Etter møte med ETC ble vi enige om at å bruke webhooks var den beste løsningen.

10.3 Kritikk av oppgaven

- **Minimal forskning på forhånd om gjennomførbarheten av oppgaven.**
  I løpet av første måneden fant vi ut at noen av teknologiene som var essensielle for deler av oppgaven var foreløpig ikke tilgjengelige i markedet. Dette ledet til at vi var nødt til å vurdere alternative løsninger for deler av oppgaven.
- **Avansert kodebase**
  ETC sin kodebase er stor og avansert å sette seg inn i. Vi synes det hadde vært fordelaktig med ende bedre opplæring i startfasen av prosjektet.
- **API nøkler som klartekst i koden**
  I koden lagrer vi API nøkler i klartekst. Dette er måten ETC lagrer nøkler, og vi ble anbefalt å også gjøre det slik. En bedre løsning hadde vært å lagre nøklene i miljøvariabler.

10.4 Videre arbeid

- Utvide modulen for registrering av ladebrikker med flere ladeselskap.
- Hente strømforbruk per bil i stedet for per lader.
- Hente strømforbruk for eksterne ladere.
- Få tak i status om biler lader eller ikke og vise det i CarAdmin.

- Vise oversikt over registrerte ladere.
- Kunne slette ladere fra systemet.
- Massefunksjon for sletting av ladebrikker.
- Binde ladere til en lokasjon.
- Utnytte fremtidig adopsjon av datautveksling fra biler som nevnt i Fremtidige teknologier.

10.5 Evaluering av gruppens arbeid

Gruppen har gjennom hele prosjektet jobbet godt sammen. Jevne møter over Teams og på skolen, sammen med valget vi tok om å bruke Water-Scrum-Fall modellen med ukentlige sprinter, gav oss muligheten til å ha god innsikt i hva alle jobber med. Det å sittet mye sammen har hjulpet med å skape et bedre arbeidsmiljø og muligheten til å stille spørsmål til hverandre. Vi har hjulpet hverandre om noen sitter fast, og på den måten hatt en jevn progresjon. Dette har ført til at vi har truffet alle milepæler rundt den tiden vi forutsa i gantskjema (figur 2 og 3 i appendiks C Prosjektplan), med unntak av at opplæringsprosessen i CarAdmins kodebase fortsatte lengre ut i utviklingsfasen. Oversikt over disponering av tid kan man se i K Timelogg.

10.6 Konklusjon

Gjennom prosjektet har vi lært mye og fått mange gode erfaringer. Vi har fått mye erfaring med å arbeide i et større prosjekt i en allerede eksisterende løsning. I tillegg fikk vi oppleve hvordan det er å kontakte relevante aktører i industrien. Det var spennende å utvikle i CarAdmin og jobbe sammen med flinke utviklere fra ETC. Oppgaven vi fikk var krevende, samtidig som vi opplevde mestring. På grunn av godt samarbeid innen gruppen og med ETC har vi utviklet en programvare vi er stolte av, og som vi er sikker på vil være til nytte. Vi vil takke Electric Time Car AS for muligheten de har gitt oss.

Referanser

Auth0. *Single Sign-On*. URL: https://auth0.com/docs/authenticate/single-sign-on (sjekket 6. mai 2022).

elbilforening, Norsk. *Norwegian EV market*. URL: https://elbil.no/english/norwegian-ev-market/ (sjekket 10. mai 2022).

Energy, U.S. Department of. *The Smart Grid*. URL: https://www.smartgrid.gov/the_smart_grid/smart_grid.html (sjekket 27. apr. 2022).

Enode. *Enode API (1.12.0)*. URL: https://docs.enode.io/api-reference (sjekket 15. feb. 2022).

— *Enode Developer Hub*. URL: https://docs.enode.io (sjekket 15. feb. 2022).

ETC. *Om oss*. URL: https://caradmin.no/om-oss/ (sjekket 12. jan. 2022).

GeeksForGeeks. *hmac – Keyed-Hashing for Message Authentication*. URL: https://www.geeksforgeeks.org/hmac-keyed-hashing-message-authentication/ (sjekket 16. mai 2022).

IBM. *JSP and Java servlet programming*. URL: https://www.ibm.com/docs/en/i/7.2?topic=java-jsp-servlet-programming (sjekket 6. mai 2022).

Ishikawa, Takanori. *Java Sample Code for Calculating HMAC-SHA1 Signatures*. URL: https://gist.github.com/ishikawa/88599?permalink_comment_id=2029198 (sjekket 16. mai 2022).

MariaDB. *MariaDB Server: The open source relational database*. URL: https://mariadb.org (sjekket 7. mai 2022).

Mültin, Marc. *What is ISO 15118?* URL: https://www.switch-ev.com/knowledgebase/what-is-iso-15118 (sjekket 27. apr. 2022).

NTNU. *PROG2900 - Bacheloroppgave*. URL: https://www.ntnu.no/studier/emner/PROG2900#tab=omEmnet (sjekket 26. jan. 2022).

Paganini, Pierluigi. *Near field communication (NFC) technology, vulnerabilities and principal attack schema*. URL: https://resources.infosecinstitute.com/topic/near-field-communication-nfc-technology-vulnerabilities-and-principal-attack-schema/ (sjekket 27. apr. 2022).

ReQtest. *How to Make Agile and Waterfall Methodologies Work Together*. URL: https://reqtest.com/agile-blog/agile-waterfall-hybrid-methodology-2/ (sjekket 17. jan. 2022).

Restlet. *Restlet Framework*. URL: https://restlet.talend.com (sjekket 16. mai 2022).

Standardization), ISO (the International Organization for. *Cards and security devices for personal identification*. URL: https://www.iso.org/standard/76566.html (sjekket 27. apr. 2022).

Appendix

A Ordliste

bilansvarlig er en person som har ansvar for bedriften sine kjøretøy.. 1, 8, 61

ekstern lader er en offentlig lader langs veien som alle kan benytte seg av.. 2, 8–10

HTML-parsing HTML-parsing brukes til å navigere nettsider gjennom kode og hente ut informasjon. 9, 13

intern lader er en privat lader i garasjen en bedrift som de ansatte kan benytte.. 2, 8, 9, 26, 29

ladebrikke er en RFID-brikke man skanner på en elbillader. Det blir brukt som betalingsmiddel, og/eller for å autentisere brukeren som vil lade.. 6, 8, 24, 25, 27, 40, 42, 49, 50

RFID-brikke En RFID-brikke er en brikke som kommuniserer med en mottaker i nærområdet gjennom radiosignaler.. 73, 84

tilgangstoken heter access token på engelsk, og er et token man bruker for å få tilgang til en API.. 62, 63

B Akronymer

AJAX Asynchronous JavaScript And XML. 13, 28, 40–43, 50, 64

API Application Programming Interface. 2, 9, 13, 31, 47, 53, 61, 69, 77, 79

ETC Electric Time Car AS. 1, 8, 10, 48

IO Input/output. 40, 41, 43, 50

JSON JavaScript Object Notation. 62, 66, 67

JSP Java Server Pages. 28, 40, 41, 49, 50, 60

kWh Kilowattime. 29

MVC Model View Controller. 13, 28, 33

NFC Near Field Communication. 10, 11

REST Representational State Transfer. 31

RFID Radiofrekvensidentifikasjon. 2, 8, 9, 11, 24, 57–59, 84

SQL Structured Query Language. 29, 41–43, 68

TSR Task success rate. xii, 73

UI User interface. 23

VIN Vehicle Identification Number. 12

C Prosjektplan

NTNU
Kunnskap for en bedre verden

Prosjektplan

Automatisering av ladebrikkeregistrering og samlet strømforbruk

Abdulhadi Anwar Mahdi Al-Sayed (505984)
Elliot Sveum Torp (526356)
Rihards Daniels Ustinovics (526378)
Sindre Emil Vikre (526335)

I samarbeid med

ETC
ELECTRIC TIME CAR

Vår, 2022

Table of Contents

1 Mål og rammer

1.1 Bakgrunn

"ETC er et innovativt IT-selskap med nyskapende total-løsning for helhetlig kjøretøyoppfølging kalt CarAdmin, som benyttes i små og store kjøretøyparker."[1]

Elbilmarkedet har opplevd enorm vekst de siste årene. Kundene til Electric Time Car (videre benevnt som "ETC" eller "oppdragsgiver") er i en overgang fra å bruke bensin- og dieselbiler, til elbiler. Overgangen har medført to nye problemer; ineffektiv registrering av ladebrikker og uoversiktlig strømforbruk/fakturering.

ETC ønsker å utvide modulen for registrering av ladebrikker slik at man bare trenger å registrere brikken en gang gjennom grensesnittet til CarAdmin. Deretter blir brikken automatisk registrert hos de ulike ladestasjonselskapene. I tillegg ønsker de å samle strømforbruk og fakturainformasjon på deres nettside, slik at det er enklere for kundene å holde oversikt.

1.2 Prosjektmål

1.2.1 Effektmål

Effektmål beskriver hva oppdragsgiver tjener/oppnår/vinner på systemet vi skal utvikle. ETC ønsker at utviklingen av registrerings- og strømforbruk modulene skal bidra til:

- Mer fornøyde kunder/brukere av CarAdmin ved å spare tid og ressurser.
- Kunde-binding, hjelpe kunder med enda et problem slik at de velger CarAdmin overfor andre konkurrenter.
- Å gi kunden bedre oversikt over kostnader, som igjen gir mer fornøyde kunder.
- Et produkt som trekker til seg flere kunder ved å være attraktiv for elbil markedet.

1.2.2 Resultatmål

Resultatmål er hva oppdragsgiver får overlevert (programvare/teknisk/fysisk) som resultat av prosjektet. Systemet vi lager skal resultere i:

- Et system for automatisk registrering av ladebrikker hos ulike ladeselskaper gjennom CarAdmin sitt brukergrensesnitt.
- En samlet oversikt over strømforbruk og fakturainformasjon i CarAdmin sitt brukergrensesnitt.
- En CarAdmin modul som kan adoptere og tilpasse ladestasjon selskap sine APIs i fremtiden.

[1] ETC. *Om oss*. URL: https://caradmin.no/om-oss/ (visited on 12th Jan. 2022).

1.2.3 Læringsmål

Læringsmål er hva vi vil lære i denne prosessen. I tillegg til læringsmålene i emnebeskrivelsen til PROG2900[2] har vi definert følgende læringsmål:

- Lære om nye teknologier.
  - Java-basert web utvikling.
  - Databaser med MariaDB.
- Lære å utvikle en mikrotjenester for et eksisterende system.
- Få mer kunnskap om teknologi rundt lading av elbil.
- Lære å utvikle et system som kommuniserer med ulike ladestasjoner.
- Lære mer om sikkerhet rundt behandling av sensitiv informasjon.
- Lære mer om utvikling av brukervennlige grensesnitt.
- Lære om samarbeid i prosjektgruppe.
- Lære å kontakte relevante personer i næringslivet for å undersøke problemer.

1.3 Rammer

1.3.1 Praktiske rammer

- NTNU har satt en tidsfrist fra 10. Januar 2022 til 20. Mai 2022 for å ferdigstille arbeidet.
- COVID-19 tiltak fører til mindre mulighet for fysisk oppmøte.
- Må jobbe sammen med flere ladestasjon bedrifter for å få tilgang til APIs.
- Krav på tre statusrapporter til veileder gjennom arbeidet den 20. Februar, 1. April og 1. Mai.
- Vi jobber med NRK som testkunde gjennom oppgaven.

1.3.2 Tekniske rammer

- Oppdragsgivers rammer.
  - Java som programmeringsspråk.
  - Restlet som RESTful web API rammeverk.
  - GitLab som versjonskontrollsystem.
- Må integrere med ladestasjon bedrifter sine APIs eller andre tredjepartsløsninger.
- Løsningen skal fungere i en nettleser.
- Løsningen skal støtte Elbilforeningens RFID ladebrikke.
- Løsningen skal utvikles som et modul av ETCs CarAdmin tjeneste.

[2]NTNU. *PROG2900 - Bacheloroppgave*. URL: https://www.ntnu.no/studier/emner/PROG2900#tab=omEmnet (visited on 26th Jan. 2022).

2 Omfang

2.1 Fagområde

Tradisjonelt sett har biler vært drevet av fossilt drivstoff. De siste årene har elbiler tatt den største delen av markedet, og vil trolig vokse enda mer framover. Dette medfører naturligvis mange ulike problemstillinger og utfordringer som må løses, f.eks; Hvem skal eie elbil laderne? Hvilke betalingsmiddel skal laderne akseptere? Hvem skal få lov til å bruke laderne? Hvilke sikkerhetsmekanismer må laderne ha for å hindre misbruk? Hvordan foregår registrering hos de ulike ladeselskapene, og hvordan skal man få oversikt over strømforbruket til en elbil? Alle problemstillingene er relevante for oppgaven, men det er registrering hos ladeselskaper og samling av strømforbruk vi skal fokusere på å gjøre mer effektivt.

2.2 Avgrensning

Vi skal jobbe med å utvide den allerede eksisterende løsningen til ETC, dette setter naturligvis noen avgrensninger til hva vi skal fokusere på innen fagområdet. ETC sin tjeneste, CarAdmin, har som mål i å hjelpe større bedrifter og kommuner med å få bedre oversikt og mer effektiv drift av kjøretøyene deres. Dermed skal vi utforske problematikken rundt hva som skjer når det er mange elbiler som skal lades, registreres, og holdes oversikt over. Er det mulig å registrere flere elbiler hos ulike ladeselskaper med bare ett tastetrykk? Hvordan kan en bilansvarlig se hvor mye strøm som blir brukt når en bil lader? Hvilken avdeling i bedriften skal betale? Hvordan kan vi gjøre alt dette enklere og mer effektivt?

2.3 Oppgavebeskrivelse

For en privatperson med en til to biler kan registrering av ladebrikke hos en eller flere ladeselskaper være en enkel oppgave. Når et firma som eier en elbil-flåte må utføre samme prosess, kan det være en tidkrevende oppgave ettersom hver brikke må manuelt registreres hos alle de ulike ladeselskapene som benyttes. Når en bil lader hos flere ulike ladestasjoner blir er det komplisert å holde oversikt over hvor mye strøm det enkelte kjøretøyet bruker. For en total oversikt må en inn i løsningen til hver av selskapene og legge sammen sitt forbruk. Dette er igjen veldig tidkrevende for et firma med en stor elbil-flåte.
Oppgaven er todelt; Den første delen er å automatisere registreringsprosessen for elbil ladebrikker (RFID-brikker) når man skal registrere samme brikke hos flere ulike ladestasjon selskaper. Del to av oppgaven er å hente ut strømforbruk og fakturainformasjon fra ulike ladeselskapers løsning, og samle det i CarAdmin. Løsningen skal ha følgende funksjonalitet:

Generelt

- Vi skal hovedsakelig fokusere på ladestasjonene som NRK benytter siden de er testkunde.
- Systemet vi utvikler må være kompatibel og kunne integreres i CarAdmin.
- Vi skal kun simulere et ladestasjon selskap på grunn av manglende APIs fra eksisterende bedrifter. (Hvis vi ikke får tilgang til APIs)
- Vi skal bruke RFID-brikker til gjenkjenning av spesifikke kjøretøy.
  - Ladere som bruker Plug and Charge tillater utvidet kommunikasjon mellom kjøretøy og ladestasjon, så gjør dette det mulig å identifisere eieren av kjøretøyet via ladekabelen for å starte lading[3]. Vi skal ikke jobbe mot slike ladestasjoner, og fokuserer derfor på RFID-brikker.
- Ladeselskapene vi skal fokusere på er Zaptec, Easee, og Eviny.
  - Zaptec og Easee er hjemmeladere, mens Eviny er offentlige ladere langs veien. Vi har valgt disse ladestasjonene på grunn av tilgang til API.

Ladebrikke registrering

- Registrering av ladebrikker skal skje gjennom CarAdmin sitt brukergrensesnitt.
  - Bruker må legge til registreringsnummer til bilen, og RFID-nummeret kun en gang.
- Automatisk registrere en brikke hos flere forskjellige ladestasjon selskap.
  - Etter brukeren har skrevet inn informasjonen skal brikken bli registrert hos flere ladeselskaper.

Samlet strømforbruk

- Strømforbruk hentes fra en API tilbudt av ladestasjonene.
  - Strømforbruket skal vises i CarAdmin sitt grensesnitt.

Vi skal utvikle følgende deler:

- Automatisering av ladebrikke registreringen.
- Samling av strømforbruk og fakturainformasjon.
- Nye grensesnitt i CarAdmin for å registrere RFID-brikker.
- Bygge på ETC sin eksisterende database modell.

[3]Zaptec. *ISO 15118*. URL: https://zendesk.zaptec.com/hc/nb/articles/360008819417-ISO-15118 (visited on 27th Jan. 2022).

3 Prosjektorganisering

3.1 Ansvarsforhold og roller

Ansvarsforhold og roller er fordelt i henhold til ordinære scrum roller. Det vil si at roller er fordelt mellom produkteier, scrum-master og utviklere, i tillegg til rollen for veileder.

- **Produkteier:** Dag Solhaug
  - Beskrive hva som er viktig å levere innenfor utvikling av løsning
  - Skal ha oversikt over produkt backlog
  - Gi støtte ved tekniske behov for gruppen
  - Gi tilbakemelding om gruppens arbeid underveis i planlegging og utviklingsfasen av prosjektet
- **Scrum master:** Abdulhadi Al-Sayed
  - Skal ha oversikt hvordan gruppen ligger ann i forhold til prosjektplanen
  - Innkaller til møter
  - Være bindeledd mellom utviklingsteam og produkteier
  - I tillegg til alle punkt under "Utviklere"
- **Utviklere:** Elliot Sveum Torp, Rihards Daniels Ustinovics, Sindre Emil Vikre
  - Håndtere produkt backlog innenfor hvert sprint
  - Daglig scrum møte for å undersøke og tilpasse arbeid
  - Bidra til å nå sprint mål
- **Veileder:** Frode Haug
  - Hjelpe med å få gruppen i gang med rapport prosessen
  - Bidra med konstruktiv kritikk til rapport prosessen

3.2 Rutiner og grupperegler

3.2.1 Arbeidsinnsats

1. Gruppemedlemmer forventes å jobbe innen arbeidstimer på mandag, onsdag, torsdag og fredag fra 08.15-16.00 i vårsemesteret 2022.
2. Man kan få unntak hvis man gir beskjed minst dagen før, og for planlagte feriedager som f.eks påske/vinterferie, dersom dette ikke har dramatisk påvirkning for framgangen.
3. Gruppemedlemmer forventes å legge inn maks innsats i arbeidet, og å strebe etter å gjøre sitt beste.

3.2.2 Deltakelse

1. Gruppemedlemmer forventes å møte til hvert planlagte gruppemøte, møte med veileder, og møte med ETC.
2. Gruppemedlemmer forventes å være tilgjengelig i de planlagte arbeidstimene.
3. Dersom et medlem ikke kan møte opp til et møte skal de gi beskjed to dager før slik at møte kan flyttes.

3.2.3 Dokumentasjon

1. Referat skrives i et felles dokument(f.eks Google Docs) under hvert møte. Alle samarbeider om å skrive referatet.

3.2.4 Arbeidsoppgaver

1. Avtalte oppgaver forventes å være fullført til planlagt tid.
2. Gruppemedlemmer skal gi beskjed med en gang man merker at man henger etter, slik at gruppen kan tilpasse arbeidet.

3.2.5 Faglig uenighet

1. Ved faglig uenighet skal gruppen undersøke problemet sammen for å komme til enighet. Dersom gruppen ikke blir enig skal man stemme, og flertallet vinner. Det kreves lojalitet til avgjørelsen.
   (a) Det oppfordres å kontakte utviklerne i ETC og veileder for å høre deres meninger.

3.2.6 Misnøye i gruppen

1. Gruppen plikter til å snarest informere dersom de er misfornøyd med innsatsen til enkelte gruppemedlemmer. Dette blir tatt opp i samsvar med metoden under "Rutiner".

3.3 Rutiner

1. Samtale mellom alle gruppedeltagerne om problemet.
2. Deretter en skriftlig advarsel fra de andre (evt. bare lederen) der:
   (a) Regelbrudd blir påpekt.
   (b) Tidsfrister, forventet arbeid(innsats) om hva som kreves for å løse problemet.
   (c) Samtale m/veileder, ekskludering fra gruppen dersom kravene ikke blir møtt.
3. Samtale mellom hele gruppen og veileder.
4. Til slutt: skriftlig ekskludering fra gruppen. Veileder skal informeres og godkjenne valget. NB: Ingen ekskludering de 4 siste ukene før innleveringsfristen.

4 Planlegging, oppfølging og rapportering

4.1 Valg av utviklingsmodell

For å velge den riktige systemutviklingsmodellen, er det viktig at vi sammenlikner fordeler og ulemper mellom modellene, og velger den som er best tilegnet oppgaven vår. Vi kom fram til at oppgaven hovedsakelig vil bli delt opp i tre hovedfaser; planlegging/kartlegging, implementasjon, og rapport, hvor implementasjon innebærer alt av design, kode og testing.

En viktig del av oppgaven er kartlegging og undersøking av ladestasjoners løsninger. I startfasen av prosjektet bruker vi mye tid på grundig undersøkelse og planlegging av mulighetene rundt bruk av ladestasjoner. Siden vi jobber med tredjeparts løsninger som vi ikke har kontroll over er det viktig at vi planlegger hva som skal utvikles, og tenke på hva vi trenger fra ladestasjonene for å gjennomføre oppgaven. Hvis ikke risikerer vi å møte på problemer senere i utviklingen som vi ikke kan løse på grunn av manglende funksjonalitet fra ladestasjonene sin tjeneste. Denne arbeidsformen passer inn med fossefallsmodellen, som legger vekt på nøye planlegging i startfasen av prosjektet.

Problemet med fossefallsmodellen er at det er en lineær modell hvor man i prinsippet ikke kan gå tilbake faser. Dette er spesielt problematisk i implementasjonsfasen, siden vi vil gi oppdragsgiver rom til å komme med innspill underveis. Derfor mener vi SCRUM modellen er bedre egnet enn fossefallsmodellen i implementasjonsfasen. Ved å kombinere fossefallsmodellen med SCRUM modellen, vil vi benytte fordelene med begge modellene og fjerne ulempene. Systemutviklingsmodellen vi velger er altså Water-Scrum-Fall[4].

4.2 Bruk av modellen

Det finnes ingen fasit på hvordan man skal sette opp en utviklingsprosess, siden den må tilpasses hvert prosjekt. Vi har valgt å bruke Water-Scrum-Fall, så i startfasen legger vi mye vekt på grundig planlegging som vi skal dokumentere i rapporten. Når vi har nok informasjon til å starte utvikling går vi over til å følge Scrum. Vi kombinerer Scrum med Kanban for å holde oversikt over statusen til de ulike oppgavene.

Vi har valgt å ha sprinter som varer i en uke. Hver mandag har vi et sprint møte hvor vi blir enige om arbeidet som skal gjøres den kommende sprinten, og fordeler oppgaver innad i gruppen. I samme møtet skal vi reflektere over arbeidet som ble gjort forrige sprint for å passe på at vi er i rute, og identifisere hva som kan bli gjort bedre. Klokken 14 hver mandag skal vi ha møte med oppdragsgiver. I hvert møte skal vi vise hva vi gjorde i forrige sprint.

I startfasen av prosjektet lager vi en produkt backlog med oppgavene som må bli gjort. Underveis i utviklingen kan vi legge til nye oppgaver som kommer fram. Vi skal hente oppgaver fra backloggen on legge de til på Scrumban-tavlen vår i GitLab slik at det er enkelt å holde oversikt over statusen til de forskjellige oppgavene. Scrumban-tavlen vår skal ha følgende tabeller:

[4]ReQtest. *How to Make Agile and Waterfall Methodologies Work Together*. URL: https://reqtest.com/agile-blog/agile-waterfall-hybrid-methodology-2/ (visited on 17th Jan. 2022).

- To do
  - Oppgaver i backloggen som ikke har blitt påbegynt.
- In progress
  - Oppgaver som er under arbeid.
- Waiting
  - Oppgaver som venter på eksterne faktorer for å fortsette med.
- Review
  - Oppgaver som er klar for gjennomgang av et annet gruppemedlem.
- Done
  - Ferdige oppgaver.

4.3 Plan for statusmøter og beslutningspunkter i perioden

Vi har laget en fast plan for statusmøter internt i gruppen, med veileder, og med oppdragsgiver. Innad i gruppen kommuniserer vi sammen gjennom hele uken, men hver mandag morgen har vi et Scrum møte der vi går gjennom arbeidet som ble gjort forrige uke, og hva som gjør gjøres i den kommende uken.

Vi har møte med oppdragsgiver hver mandag fra kl. 14.00-14.30. Hvis vi vil ha mer tid på møtet, eller ha flere møter må vi spesifikt gi beskjed om det. Disse møtene blir brukt til å stille spørsmål, og avklare valg vi skal ta angående oppgaven.

Veiledningstimer er hver onsdag kl. 09.00, og varer opp til en time. Hvis vi vil møtes sjeldnere skal vi gi beskjed om det til veileder.

5 Organisering av kvalitetssikring

5.1 Dokumentasjon, standarder, konfigurasjonsstyring, verktøy

5.1.1 Dokumentasjon

For dokumentasjon av kildekode skal vi følge JavaDoc standarden[5] etter ETC sitt ønske. Vi skal bruke Javadoc verktøy i IntelliJ IDEA som automatisk genererer JavaDoc kommentarer[6].

5.1.2 Standarder

Standard for koding holdes konstant ved å bruke SonarLint[7]. SonarLint er en offentlig IDE utvidelse som hjelper programmerere ved å avsløre dårlig kvalitet og sikkerhetshull i koden. Verktøyet støtter utvikleren med kodingen i sanntid til motsetning av standard Lintere som må kjøres etter koden er ferdig skrevet.

For å navngi klasser, funksjoner, og variabler skal vi bruke Oracle sine "naming conventions"[8], sammen med hungarian notation på variabler. Når alle navn i koden følger samme stil blir det mer oversiktlig.

5.1.3 Konfigurasjonsstyring

Vi skal bruke GitLab som versjonskontrollsystem. ETC tilbyr med kopi av den kildekoden vi trenger for å integrere med deres systemer. Gruppen skal fokusere på å ha gode rutiner for bruk av Git når endring i koden skjer ofte. Gruppen planlegger å følge BitBuckets arbeidsflyt[9].

Vi planlegger å bruke tre nivåer for branching; main, development og feature. Hvert gruppemedlem jobber hovedsaklig i hver sin feature branch som blir slått sammen med development etter funksjonaliteten har blitt implementert. Kode skal committes jevnlig med korte og informative meldinger som forklarer jobben som har blitt gjort. All ny kode på development branchen må kommenteres, testes og bekreftes av minst to gruppemedlemmer før det kan bli slått sammen med main branchen.

5.1.4 Verktøy

Verktøy	Beskrivelse
Overleaf	LaTeX editor vi bruker til å skrive prosjektplan og prosjektrapport.
IntelliJ	IDE til koding med Java.
GitLab	Versjonskontroll verktøy til lagring av kildekode. Vi bruker og Git til å lage product backlog til Scrum.
Teams	Kommunikasjon internt i gruppen, og mellom gruppen og oppdragsgiver.
Zoom	Kommunikasjon mellom veileder og gruppen.
Excel	Loggføring av arbeidstimer.
GanttProject	Program for å lage Gantt skjema.
Google Drive	Deling av dokumenter.
Doxygen	Dokumentasjon av kode.
SonarLint	Kontrollere kodekvalitet og sikkerhets feil under kodingen.

[5] Oracle. *How to Write Doc Comments for the Javadoc Tool*. URL: https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html (visited on 28th Jan. 2022).

[6] Jetbrains. *Javadoc*. URL: https://www.jetbrains.com/help/idea/working-with-code-documentation.html (visited on 28th Jan. 2022).

[7] SonarSource. *SonarLint*. URL: https://www.sonarlint.org/intellij (visited on 20th Jan. 2022).

[8] Oracle. *9 - Naming Conventions*. URL: https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html (visited on 20th Jan. 2022).

[9] Atlassian. *Comparing Workflows*. URL: https://www.atlassian.com/git/tutorials/comparing-workflows (visited on 20th Jan. 2022).

5.2 Plan for Inspeksjoner og Testing

Utførelse av testing skal skje ved forskjellige lag i systemet. Vi utfører Unit testing for å sikre at kjernefunksjonalitet i koden virker slik det skal.

Integrasjonstesting utføres på API som produktet tilbyr for å sikre at komponenter følger REST standarder og for å sikre mot typiske angrep som injection attack[10].

Akseptansetest utføres med NRK som sluttbruker for å analysere tilbakemelding om menneskelig bruk av programvare.

Det er viktig at kode fra utviklings-branchen har blitt testet, kommentert og bekreftet av minst to gruppemedlemmer før den blir pushet og slått sammen med produksjon.

5.3 Risikoanalyse på prosjektnivå

Fargekoder for risikonivå

Lav risiko	grønn
Middels risiko	gul
Høy risiko	rød

Fargekoder for sannsynlighet og kritisk grad

Usannsynlig/Lite kritisk	grønn
Lite sannsynlig/Kritisk	gul
Sannsynlig/Veldig kritisk	oransje
Svært sannsynlig/Ekstremt kritisk	rød

[10] Lifars. *Injection Attacks Explained.* URL: https://lifars.com/2020/04/injection-attacks-explained/ (visited on 20th Jan. 2022).

Risiko før tiltak

Nr.	Hendelse	Sannsynlighet	Kritisk	Risiko før tiltak
1.	**Drastiske endringer i API eller nettsider.** Om Api-ene eller nettsidene vi planlegger å bruke får drastiske endringer, vil ikke kunden kunne registrere ladebrikkene sine. Konsekvensene av dette blir at produktet må i så fall oppdateres henholdsvis til den nye API-en.	Lite sannsynlig	Veldig kritisk	Middels
2.	**Mangel på kompetanse.** Mangel på kompetanse hos en eller flere gruppemedlemmer kan føre til tregt og dårlig arbeid.	Sannsynlig	Kritisk	Middels
3.	**Strengere COVID-19 tiltak.** Strenge COVID-19 tiltak kan føre til at det blir vanskelig å møtes fysisk. Konsekvensene av dette kan bli færre møter, dårligere arbeidsmoral, og risiko for generelt dårligere arbeid.	Sannsynlig	Kritisk	Middels
4.	**Gruppen mister medlem.** Gruppen kan miste medlem om det er store uenigheter over lang perioder i løpet av prosjektet, eller om et medlem blir syk i lengre perioder.	Usannsynlig	Veldig kritisk	Middels
5.	**Forespørselsgrenser i API.** APIer har ofte grenser på hvor ofte man kan gjøre forespørsler. Om vi sender for mange forespørsler, vil vi ikke få noe svar fra APIen.	Sannsynlig	Kritisk	Middels
6.	**Får ikke tilgang til API.** For å automatisere registreringsprosessen er vi avhengig av å få tilgang til ladeselskapers APIer. Om selskapene ikke har en offentlig API og ikke vil gi oss tilgang til deres private API, vil det være umulig å få tak i nødvendig data. Vi blir i så fall nødt til å simulere denne delen av oppgaven	Sannsynlig	Ekstremt kritisk	Høy
7.	**Manglende funksjoner hos lade leverandører.** For å kunne møte oppgavens behov er vi avhengig av at en rekke funksjoner skal være tilgjengelig fra elbilene og ladestasjonene. Om disse funksjonene ikke er tilgjengelige eller om vi ikke har muligheten til å kunne teste og utvikle for de, kan vi være nødt til å omformulere deler av oppgaven.	Sannsynlig	Kritisk	Høy

Risiko etter tiltak

Nr.	Hendelse	Tiltak	Risiko etter tiltak
1.	**Drastiske endringer i API eller nettsider.**	Holde oss oppdatert på nye versjoner av API og nettsiden, så man umiddelbart kan oppdatere produktet til å funke på den nye APIen. Skulle det være umulig å registrere ladebrikker automatisk etter en forandring i API eller nettsted, vil dette bety at dette ladeselskapet faller utenfor det vi klarer å inkludere i den automatiske ladebrikke registreringen.	Middels
2.	**Mangel på kompetanse.**	Holde hverandre oppdatert om man føler en arbeidsoppgave er for vanskelig. Jobbe sammen, hjelpe hverandre og utnytte individuelle styrker for å øke den generelle kompetansen i gruppen.	Lav
3.	**Strengere COVID-19 tiltak.**	Ved å møtes på teams og ha på kamera, kan vi forsikre oss om at alle på gruppen jobber.	Lav
4.	**Gruppen mister medlem.**	Ved å holde hverandre oppdatert på hva vi jobber med og dokumentere arbeid, vil resten av gruppen kunne fortsette arbeidet til medlemmet som forlot gruppen.	Lav
5.	**Forespørselsgrenser i API.**	Ved å legge til forsinkelser mellom API kall, kan vi passe på at vi ikke overstiger grensen på APIen.	Lav
6.	**Får ikke tilgang til API.**	Om vi ikke får tilgang til APIer for å registrere ladebrikker, kan vi bruke html parsing for å registrere brikker gjennom nettsidene. Skulle html parsing være umulig, blir det umulig å gjennomføre oppgaven, og vi må i stedet simulere at vi snakker med en API. Vi har allerede snakket med ETC, og simulering av API er et alternativ.	Lav
7.	**Manglende funksjoner hos lade leverandører.**	Om selskapene ikke tilbyr funksjonene vi krever for oppgaven og det ikke er noen andre alternative løsninger, er vi nødt til å omformulere oppgaven eller gjenkjenne at den ikke kan utføres til det som er ønsket.	Lav

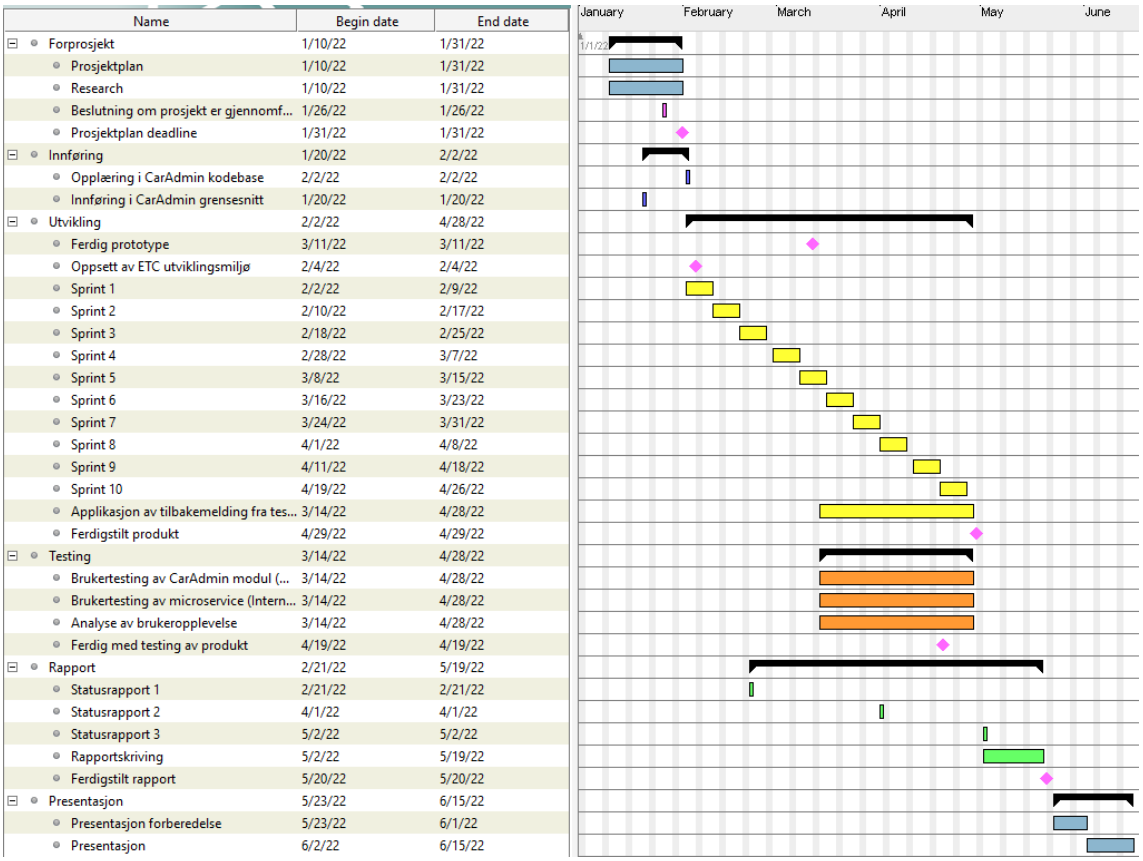
6 Plan for gjennomføring

6.1 Gantt-skjema

6.1.1 Fargekoder i Gantt-skjema

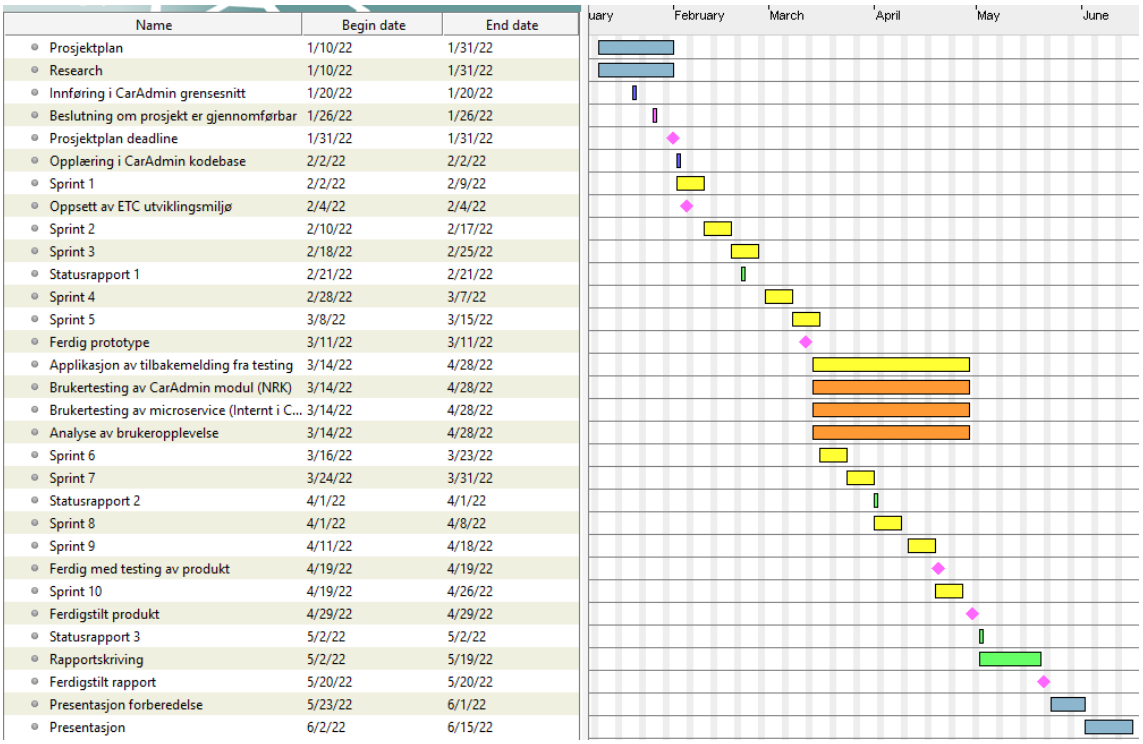
Figure 1: Gantt-skjema fargekart

6.1.2 Gantt-skjema



Name	Begin date	End date
Forprosjekt	1/10/22	1/31/22
Prosjektplan	1/10/22	1/31/22
Research	1/10/22	1/31/22
Beslutning om prosjekt er gjennomf...	1/26/22	1/26/22
Prosjektplan deadline	1/31/22	1/31/22
Innføring	1/20/22	2/2/22
Opplæring i CarAdmin kodebase	2/2/22	2/2/22
Innføring i CarAdmin grensesnitt	1/20/22	1/20/22
Utvikling	2/2/22	4/28/22
Ferdig prototype	3/11/22	3/11/22
Oppsett av ETC utviklingsmiljø	2/4/22	2/4/22
Sprint 1	2/2/22	2/9/22
Sprint 2	2/10/22	2/17/22
Sprint 3	2/18/22	2/25/22
Sprint 4	2/28/22	3/7/22
Sprint 5	3/8/22	3/15/22
Sprint 6	3/16/22	3/23/22
Sprint 7	3/24/22	3/31/22
Sprint 8	4/1/22	4/8/22
Sprint 9	4/11/22	4/18/22
Sprint 10	4/19/22	4/26/22
Applikasjon av tilbakemelding fra tes...	3/14/22	4/28/22
Ferdigstilt produkt	4/29/22	4/29/22
Testing	3/14/22	4/28/22
Brukertesting av CarAdmin modul (...	3/14/22	4/28/22
Brukertesting av microservice (Intern...	3/14/22	4/28/22
Analyse av brukeropplevelse	3/14/22	4/28/22
Ferdig med testing av produkt	4/19/22	4/19/22
Rapport	2/21/22	5/19/22
Statusrapport 1	2/21/22	2/21/22
Statusrapport 2	4/1/22	4/1/22
Statusrapport 3	5/2/22	5/2/22
Rapportskriving	5/2/22	5/19/22
Ferdigstilt rapport	5/20/22	5/20/22
Presentasjon	5/23/22	6/15/22
Presentasjon forberedelse	5/23/22	6/1/22
Presentasjon	6/2/22	6/15/22

Figure 2: Gantt-skjema sortert etter tema



Name	Begin date	End date
Prosjektplan	1/10/22	1/31/22
Research	1/10/22	1/31/22
Innføring i CarAdmin grensesnitt	1/20/22	1/20/22
Beslutning om prosjekt er gjennomførbar	1/26/22	1/26/22
Prosjektplan deadline	1/31/22	1/31/22
Opplæring i CarAdmin kodebase	2/2/22	2/2/22
Sprint 1	2/2/22	2/9/22
Oppsett av ETC utviklingsmiljø	2/4/22	2/4/22
Sprint 2	2/10/22	2/17/22
Sprint 3	2/18/22	2/25/22
Statusrapport 1	2/21/22	2/21/22
Sprint 4	2/28/22	3/7/22
Sprint 5	3/8/22	3/15/22
Ferdig prototype	3/11/22	3/11/22
Applikasjon av tilbakemelding fra testing	3/14/22	4/28/22
Brukertesting av CarAdmin modul (NRK)	3/14/22	4/28/22
Brukertesting av microservice (Internt i C...	3/14/22	4/28/22
Analyse av brukeropplevelse	3/14/22	4/28/22
Sprint 6	3/16/22	3/23/22
Sprint 7	3/24/22	3/31/22
Statusrapport 2	4/1/22	4/1/22
Sprint 8	4/1/22	4/8/22
Sprint 9	4/11/22	4/18/22
Ferdig med testing av produkt	4/19/22	4/19/22
Sprint 10	4/19/22	4/26/22
Ferdigstilt produkt	4/29/22	4/29/22
Statusrapport 3	5/2/22	5/2/22
Rapportskriving	5/2/22	5/19/22
Ferdigstilt rapport	5/20/22	5/20/22
Presentasjon forberedelse	5/23/22	6/1/22
Presentasjon	6/2/22	6/15/22

Figure 3: Gantt-skjema sortert etter tid

6.1.3 Gantt-skjema agenda

Navn	Dato	
	Start	Slutt
Forprosjekt	1/10/22	1/31/22
Prosjektplan	1/10/22	1/31/22
Research	1/10/22	1/31/22
Beslutning om prosjekt er gjennomførbar	1/26/22	1/26/22
Prosjektplan deadline	1/31/22	1/31/22
Innføring	1/20/22	2/2/22
Opplæring i CarAdmin kodebase	2/2/22	2/2/22
Innføring i CarAdmin grensesnitt	1/20/22	1/20/22
Utvikling	2/2/22	4/28/22
Ferdig prototype	3/11/22	3/11/22
Oppsett av ETC utviklingsmiljø	2/4/22	2/4/22
Sprint 1	2/2/22	2/9/22
Sprint 2	2/10/22	2/17/22
Sprint 3	2/18/22	2/25/22
Sprint 4	2/28/22	3/7/22
Sprint 5	3/8/22	3/15/22
Sprint 6	3/16/22	3/23/22
Sprint 7	3/24/22	3/31/22
Sprint 8	4/1/22	4/8/22
Sprint 9	4/11/22	4/18/22
Sprint 10	4/19/22	4/26/22
Applikasjon av tilbakemelding fra testing	3/14/22	4/28/22
Ferdigstilt produkt	4/29/22	4/29/22
Testing	3/14/22	4/28/22
Brukertesting av CarAdmin modul (NRK)	3/14/22	4/28/22
Brukertesting av microservice (Internt i CarAdmin)	3/14/22	4/28/22
Analyse av brukeropplevelse	3/14/22	4/28/22
Ferdig med testing av produkt	4/19/22	4/19/22
Rapport	2/21/22	4/19/22
Statusrapport 1	2/21/22	2/21/22
Statusrapport 2	4/1/22	4/1/22
Statusrapport 3	5/2/22	5/2/22
Rapportskriving	5/2/22	5/2/22
Ferdigstilt rapport	5/20/22	5/20/22
Presentasjon	5/23/22	6/15/22
Presentasjon forberedelse	5/23/22	6/1/22
Presentasjon	6/2/22	6/15/22

Table 1: Gantt-skjema agenda

6.2 Milepæler og beslutningspunkter

Vi har bestemt noen milepæler og beslutningspunkter for å ha interne frister for gruppen, og sørge for at vi har den progresjonen vi trenger. I gantt skjemaet er milepæler er merket med en rosa diamant, og beslutningspunkter er merket med en rosa rektangel.

- **Beslutningspunkt 1, 26. januar:** Bestem om oppgaven er gjennomførbar med tanke på tilgang til APIs, eller om den krever omformulering.
- **Milepæl 1, 31. januar:** Prosjektplan ferdig.
- **Milepæl 2, 4. februar:** Få på plass utviklingsmiljø.
  - Etter utviklingsmiljøet er på plass begynner vi å utvikle programmet.
- **Milepæl 3, 11. mars:** Prototype ferdig.
  - Prototypen skal være klar med grunnleggende funksjonalitet.
- **Milepæl 4, 19. april:** Grundig testet produkt med all funksjonalitet er klart.
  - All funksjonalitet skal være ferdig og fungerende, vi begynner ikke på utvikling av ny funksjonalitet.
  - Hoveddelen av testing er ferdig. Idealt har vi testet med en bil og en ladestasjon.
- **Milepæl 5, 29. april:** Ferdigstilt produkt.
  - Ferdig med finpuss og testing, fokuset går over til rapport.
- **Milepæl 6, 20. mai:** Ferdigstilt rapport.

References

Atlassian. *Comparing Workflows*. URL: https://www.atlassian.com/git/tutorials/comparing-workflows (visited on 20th Jan. 2022).

ETC. *Om oss*. URL: https://caradmin.no/om-oss/ (visited on 12th Jan. 2022).

Jetbrains. *Javadoc*. URL: https://www.jetbrains.com/help/idea/working-with-code-documentation.html (visited on 28th Jan. 2022).

Lifars. *Injection Attacks Explained*. URL: https://lifars.com/2020/04/injection-attacks-explained/ (visited on 20th Jan. 2022).

NTNU. *PROG2900 - Bacheloroppgave*. URL: https://www.ntnu.no/studier/emner/PROG2900#tab=omEmnet (visited on 26th Jan. 2022).

Oracle. *9 - Naming Conventions*. URL: https://www.oracle.com/java/technologies/javase/codeconventions-namingconventions.html (visited on 20th Jan. 2022).

— *How to Write Doc Comments for the Javadoc Tool*. URL: https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html (visited on 28th Jan. 2022).

ReQtest. *How to Make Agile and Waterfall Methodologies Work Together*. URL: https://reqtest.com/agile-blog/agile-waterfall-hybrid-methodology-2/ (visited on 17th Jan. 2022).

SonarSource. *SonarLint*. URL: https://www.sonarlint.org/intellij (visited on 20th Jan. 2022).

Zaptec. *ISO 15118*. URL: https://zendesk.zaptec.com/hc/nb/articles/360008819417-ISO-15118 (visited on 27th Jan. 2022).

D Grupperegler

Kontrakt med grupperegeler

Arbeidsinnsats

1. Gruppemedlemmer forventes å jobbe innen arbeidstimer på mandag, onsdag, torsdag og fredag fra 08.15-16.00 i vår semesteret 2022.
2. Man kan få unntak hvis man gir beskjed minst dagen før, og for planlagte feriedager som f.eks påske/vinterferie, dersom dette ikke har dramatisk påvirkning for framgangen.
3. Gruppemedlemmer forventes å legge inn maks innsats i arbeidet, og å strebe etter å gjøre sitt beste.

Deltakelse

1. Gruppemedlemmer forventes å møte til hvert planlagte gruppemøte, møte med veileder, og møte med ETC.
2. Gruppemedlemmer forventes å være tilgjengelig i de planlagte arbeidstimene.
3. Dersom et medlem ikke kan møte opp til et møte skal de gi beskjed to dager før slik at møte kan flyttes.

Dokumentasjon

1. Referat skrives i et felles dokument(f.eks Google Docs) under hvert møte. Alle samarbeider om å skrive referatet.

Arbeidsoppgaver

1. Avtalte oppgaver forventes å være fullført til planlagt tid.
2. Gruppemedlemmer skal gi beskjed med en gang man merker at man henger etter, slik at gruppen kan tilpasse arbeidet.

Faglig uenighet

1. Ved faglig uenighet skal gruppen undersøke problemet sammen for å komme til enighet. Dersom gruppen ikke blir enig skal man stemme, og flertallet vinner. Det kreves lojalitet til avgjørelsen.
    a. Det oppfordres å kontakte utviklerne i ETC og veileder for å høre deres meninger.

Missnøye i gruppen

1. Gruppen plikter til å snarest**(hva er tidlig?)** informere dersom de ikke er fornøyd med innsatsen til enkelte gruppemedlemmer. Dette blir tatt opp i samsvar med metoden under “Rutiner”.

Rutiner

1. Samtale mellom alle gruppedeltagerne om problemet.
2. Deretter en skriftlig advarsel fra de andre (evt. bare lederen) der:
    a. regelbrudd blir påpekt.
    b. Tidsfrister, forventet arbeid(innsats) om hva som kreves for å løse problemet.
    c. Samtale m/veileder, ekskludering fra gruppen dersom kravene ikke blir møtt.
3. Samtale mellom hele gruppen og veileder.
4. Til slutt: skriftlig eksludering fra gruppen. Veileder skal informeres og godkjenne valget.
   NB: Ingen ekskludering de 4 siste ukene før innleveringsfristen.

Signaturer:	Dato:
Sindre E. Vibre	31.01.2022
Emil S. Topp	31.01.2022
Abdulhaf. Al-Sayed	31.01.2022
Rihards Dorads Ustinovics	31.01.2022

E Arbeidskontrakt

NTNU

Norges teknisk-naturvitenskapelige universitet

Fastsatt av prorektor for utdanning 10.12.2020

STANDARDAVTALE

om utføring av studentoppgave i samarbeid med ekstern virksomhet

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

Forklaring av begrep

Opphavsrett
Er den rett som den som skaper et åndsverk har til å fremstille eksemplar av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

Eiendomsrett til resultater
Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

Bruksrett til resultater
Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

Prosjektbakgrunn
Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

Utsatt offentliggjøring
Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt:
Veileder ved NTNU: e-post og tlf.
Ekstern virksomhet: **ETC, Electric Time Car AS** Ekstern virksomhet sin kontaktperson, e-post og tlf.: Dag L Solhaug, dag@electrictimecar.com, 90101344
Student: Sindre E. Vihre Fødselsdato: 25.07.2000
Ev. flere studenter[1] Eliot S. Torp 03.06.2000
Abdulhadi Al-Sayed 26.06.1999

Rikards Daniels Ustinovics 15.11.2000

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bacheloroppgave	X
Prosjektoppgave	
Annen oppgave	

Startdato: 10.01.2022
Sluttdato: 20.05.2022

Oppgavens arbeidstittel er: **Felles Ladebrikke registrering – samlet strømforbruk**

[1] Dersom flere studenter skriver oppgave i fellesskap, kan alle føres opp her. Rettigheter ligger da i fellesskap mellom studentene. Dersom ekstern virksomhet i stedet ønsker at det skal inngås egen avtale med hver enkelt student, gjøres dette.

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven: **Avgjøres etter løpende forespørsel fra studentene.**

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

4. Studentens rettigheter

Studenten har opphavsrett til oppgaven[2]. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten

[2] Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

Alternativ a) (sett kryss) Hovedregel

	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

Alternativ b) (sett kryss) Unntak

X	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene: **Kommersielt produkt.**

6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

	Oppgaven skal være offentlig

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Oppgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss		Sett dato
	ett år	
	to år	
X	tre år	

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Behov for varig utsettelse av offentliggjøring gjelder kun de deler som viser vår kode, eller utnytter/arver vår kode eller metoder. Dette kan legge i vedlegg slik at Prosjektrapporten kan offentliggjøres, mens vedlegget ikke offentliggjøres. Dette er begrunnet i kommersielle hensyn og sikkerhetshensyn for drift og persondata lagret i vår løsning for våre kunder.

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

9. Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

Signaturer:

Instituttleder: Dato:
Veileder ved NTNU: Dato:
Ekstern virksomhet: Dato: 28.01.2022 Dag L Solhaug
Student: Sindre E. Vikre Dato: 31.01.2022
Ev. flere studenter Elliot S. Torp 31.01.2022
Abdulhadi Al-Sayed 31.01.2022

Richards Dainuts Ustinovics ~~31.01.2022~~ 31.01.2022

F Referat fra møter med ETC

10.12.2021

Bli kjent møte med oppdragsgiver

06.01.2022

Møte med ETC og NRK representant, hvor vi lærte mer om NRK sine systemer og deres forventninger til bacheloroppgaven. Avtalte ukentlige møter med ETC.

13.01.2022

Kort møte med oppdragsgiver for å stille oppklarende spørsmål om registreringen av en ladebrikke.

17.01.2022

Oppdaterte oppdragsgiver på status som er at vi har funnet APIs til strømforbruket til de interne laderne, men ikke noe til de eksterne. Vi fikk og avklart hvordan NRK bruker laderne sine, at de ikke scanner en RFID brikke til de interne laderne i garasjen. Derfor må vi få info fra “handshake”. Vi fikk høre noe mer om målene deres med prosjektet, og ble enig om å sende oppgavebeskrivelsen til ETC for å få flere detaljer.

20.01.2022 salg team

Vi hadde et møte med Tom og Randi fra ETC som gav en grundig innføring i tjenestene som ETC tilbyr, og gjennomgang av nettsiden til CarAdmin.

24.01.2022

Spurte om det er noen rammeverk og standarder de ønsker vi skal bruke. Hoveddelen av møte gikk til å fortelle om funnene våres, og hva vi tror er mulig. Videre skal vi vente på svar fra Eviny om vi får tilgang til API, og høre om vi skal omformulere oppgaven, eller om vi kan utføre det vi orginalt planlagte.

31.01.2022

Innkalling: Be om innføring i kodebasen. Fortelle om status med Eviny. Status på oppgaven. Deltidsjobb.

Vi avtalte å få innføring i koden torsdag 3. januar kl 09. Vi fortalte og om status på APIs, og fant ut at Dag heller foretrekker html parsing enn simulering av API, men aller helst begge deler. Vi snakket og om at oppgaven ligger litt foran hva elbil-lader markede er klar for, og at Dag foretrekker en løsning som fungerer nå, og som enkelt kan tilpasses når bedre løsninger i markedet kommer fram. Slik som det er nå skal vi fokusere på Zaptec og Easee sine APIs for strømforbruk, og bruke html parsing til å registrere RFID-brikker.

03.02.2022 med Sebastian(seniorutvikler)

Innkalling: Sette opp utviklingsmiljø

I dag hadde vi et møte med Sebastian fra ETC. Han hjalp oss med å sette opp prosjektet, så nå har vi CarAdmin tjenesten som kjører lokalt på våre maskiner. Vi snakket og om videre veiledning med utvikling, så etter Sebastian har snakket med Dag og Øyvind skal de sende forslag til hvordan og når vi skal få mer veiledning.

07.02.2022

Innkalling: Høre om når vi får innføring i koden. Fortelle om Enode.

Vi skal få innføring i koden torsdag kl 12.00. Vi skal møte fysisk på kontoret deres! De var positiv til Enode API.

14.02.2022

Innkalling: Oppdatering på APIs. Fortelle status på prosjektet.

Vi fortalte at vi har fått tilgang til Enode API, og ble enige om at vi skal undersøke hva det vil koste å bruke den APIen. Vi fortalte at vi har hørt fra Odd Olaf, og håper på å få tilgang til en API hos Eviny etterhvert, men ingenting er sikkert. Spurte om å få test brukere, og ble enig om å kontakte Pål fra NRK direkte. Fortalte at vi begynner på simulering.

23.02.2022

Innkalling: Oppdatere på framgang.

Vi fortalte Øyvind hvordan vi ligger an, og fikk positiv respons. De i ETC synes vi ligger bra an, og jobber fort.

02.03.2022

Innkalling: Spørre om det vi planlegger med Enode er noe ETC vil bruke.

Vi viste dem Enode, og Øyvind var positiv til det, og vil at vi skal fortsette med Enode. Vi ble og enige om å bruke webhooks til å hente ny data om ladinger (fra Enode)

16.03.2022

Innkalling: Status, og avtale framvisning av prototype
Vi fortalte hva vi har fått til så langt, og bel enige om å vise fram prototypen fredag 25. mars siden de var litt opptatt hos ETC.

25.03.2022

Innkalling: Vise prototype, snakke om hva som skal endres på, og hva som er bra.

Til stede i møtet var utviklerne Øyvind og Sebastian fra ETC, i tillegg til grunne vår. Vi viste prototypen vår, og de var generelt fornøyd med det vi har utviklet. De hadde noen tilbakemeldinger på hva vi måtte endre på for å som f.eks hvor brukeren skal legge til innloggingsinfo til et nytt ladeselskap, og hva som skal skje etter en bruker registrerer en ladebrikke hos et ladeselvskap. Vi fikk og tips til hva vi måtte gjøre for å legge til data om strømforbruk i rett selskap sin database, siden ETC har helt separate databaser for hver kunde.

30.03.2022

Innkalling: Spørre om vi kan lagre database-navn hos Enode, og noen oppklarende spørsmål rundt design på den nye måten å legge til ladeselskaper. Spurte hvordan ETC tester kode.

Vi fikk vite fra Øyvind at det går helt fint å lagre database navnene hos Enode. Det gjør det mye lettere å vite hvilken bedrift hver enkelt lading hører til. Vi fikk beskjed om å bruke eksisterende design i nettsiden som eksempel på hvordan det skal se ut. Sebastian viste oss hvordan de skriver tester i kodebasen deres.

20.04.2022

Innkalling: Spørre om hjelp med sorteringen etter dato i tabeller. Hvordan vi lagrer API keys.

Sebastian deltok i møte, da viste han oss hvordan vi kunne sortere etter dato. Vi lurte og på hvordan vi skal sortere API keys som blir brukt i koden. Da anbefalte han oss å lagre nøklene som en konstant i koden.

27.04.2022

Innkalling: Vise ferdig produkt, spørre om brukertest, tilgang til gitlab for sensor.

—----------Notater—----------

06.01.2022

Avtalt ukentlig møte hver mandag klokken 14:00

Pål sin hverdag internt og eksternt:

- 80% elektrisk
- 30% elektrisk nå
- Lite skrudd for flåtebrukere
- Dårlig opplevelse å registrere
- Vanskelig å sortere registrerte basert på f.eks reg.nummer
- Samme backend på noen operatører
- Brutt lading varsling mulig å implementere?

Fokusere på info fra lader for økonomi
Identifisere gjennom VIN nummer

Charge and drive bkk har API kanskje
Andre har api?

Circlek, ionity og eon deler samme backend leverandør.
CircleK har eget kort.

13.01.2022

Elbilforening eller møller ladebrikke
Mulighet for lagring av påloggingsinfo og ladebrikker.
Circle K - har løsning allerede.
Brikke id og reg nummer.
To konto nummere for hybrid biler. Drivstoff kort og elkort. Kan teknisk sett knytte så mange kort til en bil som man vil.

17.01.2022

Oppdaterte ETC med statusrapport

20.01.2022 Innføring i ETC og CarAdmin

Var store utfordringer i kommuner rundt å låne biler, som “hvor er nøkkelen….osv”

De er delt opp i moduler. fellesbil/gps, felles orginal, el-sykkel

De må ha et system hvor de kan

ok så de har en app

man kan starte med en modul, også utvide etterhvert som man trenger det sky.

er det reg. nummer de bruker for registrering. ja, reg.nummer blir mest brukt.

noen vil og ha kallenavn, som “sykepleier bil”

registrere brikke i carAdmin. Kjøretøyoversikt -> bruksinfo

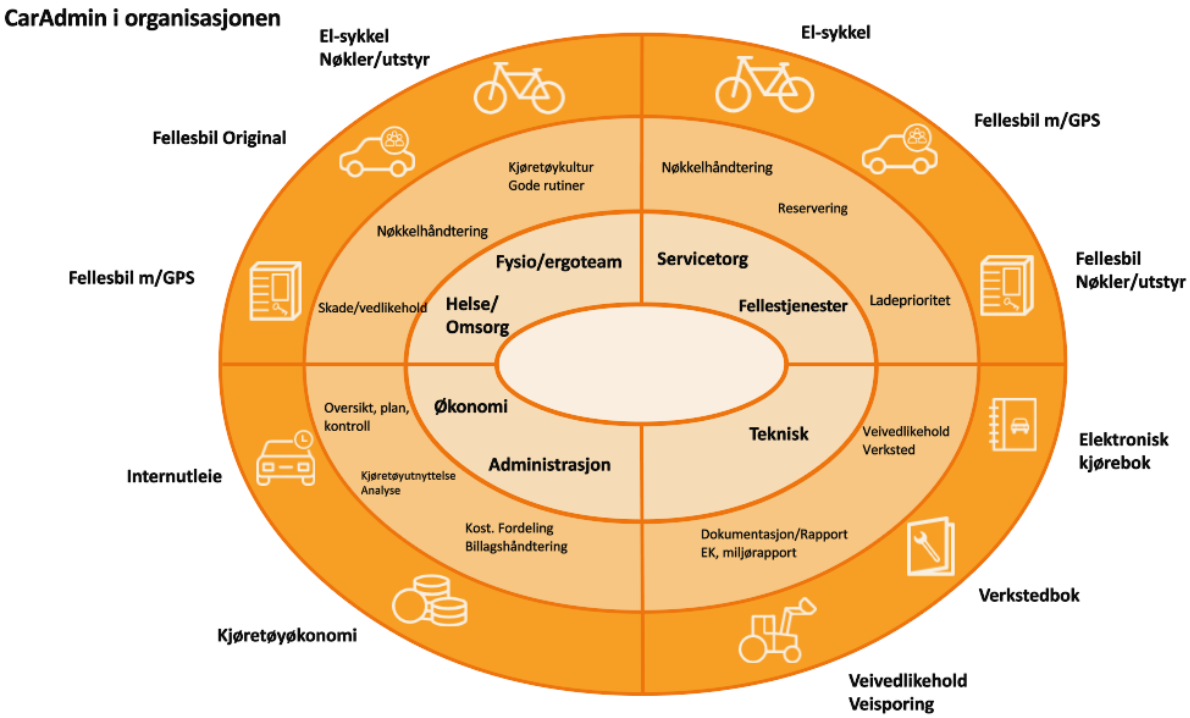
enten registrere brikke n skapet, eller sitte med PC.

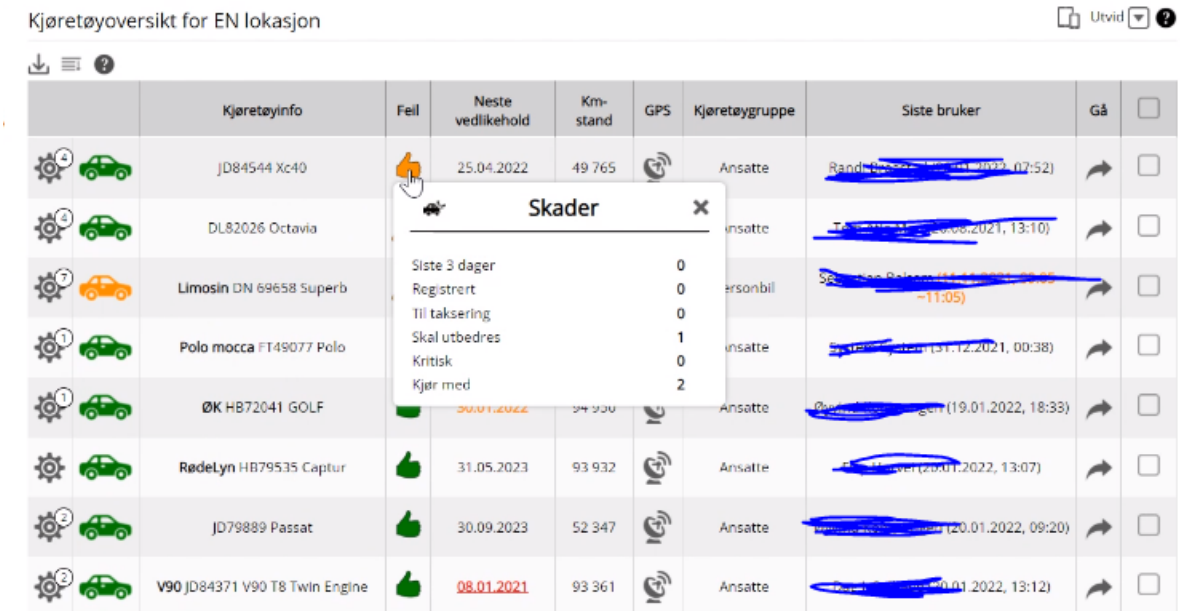
koden på RFID brikken er forskjellig ulike steder? annet format? what?

de fordeler kostnadene også sennder det tilbake til bedrifen som skal betale det gjennom en API.

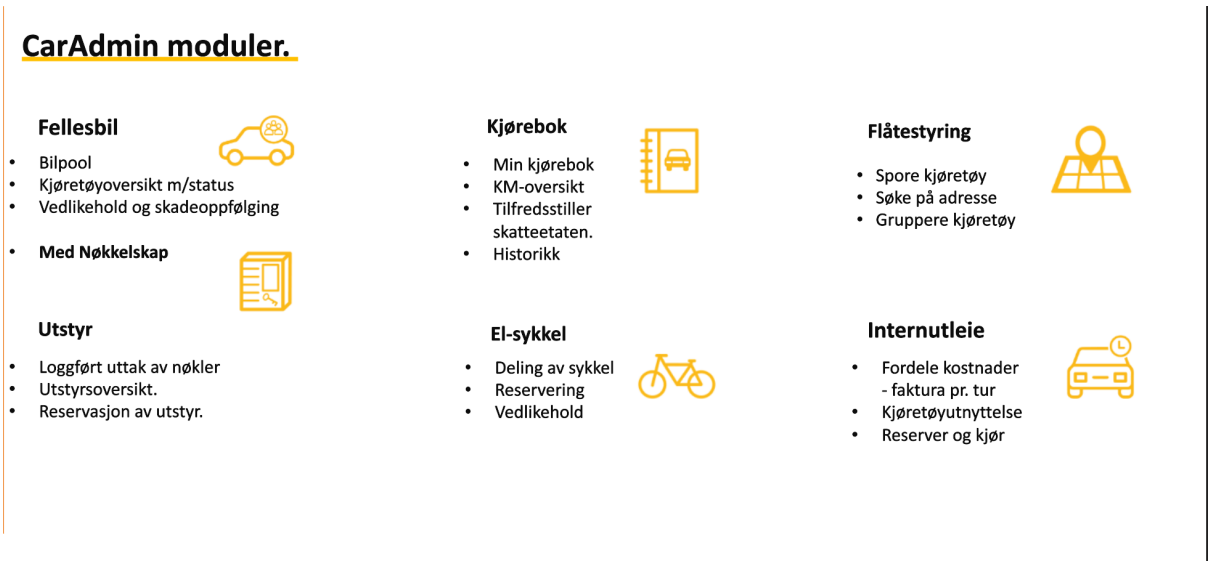
ok det er vår oppgabe basically.

de regner fra tidspunktet de tar ut nøkkelen fra forrige gang den ble levert inn, også tiden den var ute, også bare en ca beregning på hvor mye den har ladet, og hvor mye strøm de har brukt.



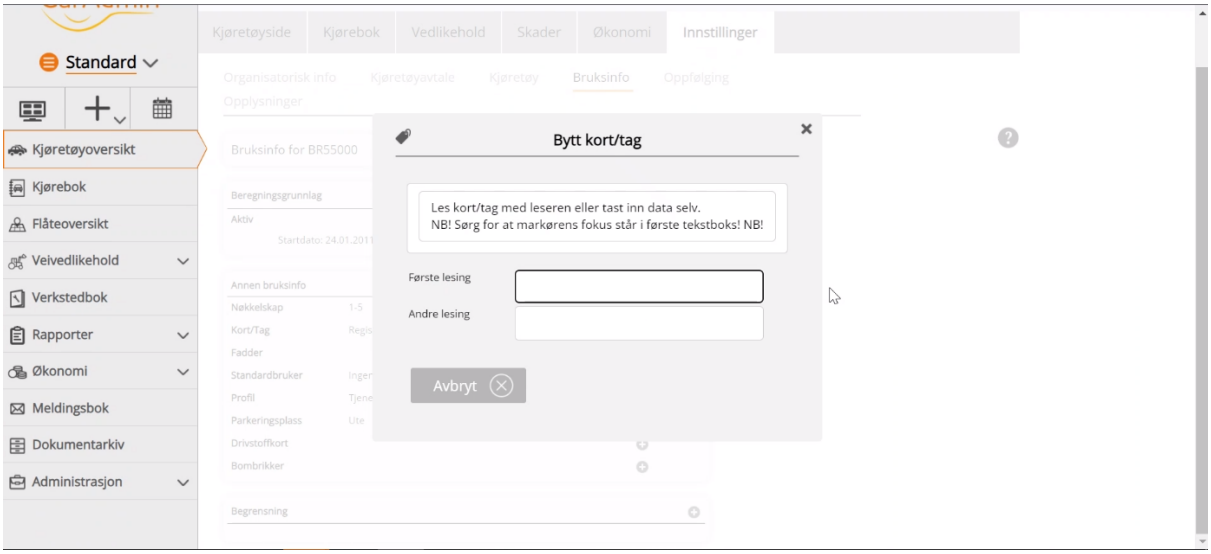


Måten koden fra brikke blir lest kan være forskjellig hos caradmin og ladeselskapene.



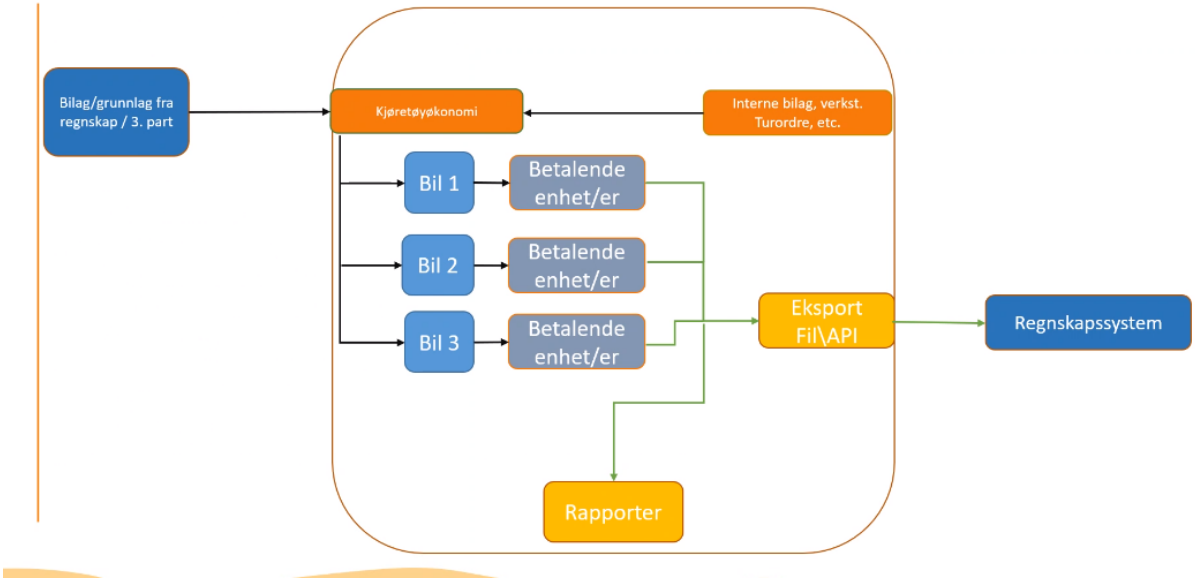
25.000 aktive brukere. Oppi 80 kommuner/bydeler. Nesten alle bydeler i Oslo. <- just fun fact

Kobler opp RFID brikker sammen med bilen:



Virker som de nå tar å beregner stoppe tid hvor nøkkelen har blitt levert, som ladetid, for å sjekke hvem som har blitt ladet lengst, og deretter gi nøkkelen til bilen som har blitt ladet

lengst. Men dette er bare en anntagning, kan jo hende bilen ikke har blitt ladet i det hele tatt, bare stått stille. Derfor er dette en del av oppgaven vår.



03.02.2022

- Gikk gjennom oppsett av ETC server med Sebastian
- Avtalte hjelp for guide av å lage nye sider

25.03.2022

- Demonstrasjon av prototype
    - Registrere ladebrikker
    - Demonstrere login (med riktig/feil passord)
    - Demonstrere login sjekk ved mangel for detaljer

Tips fra ETC:

- Restlet har egen prosess for innlogging, dermed klipping og liming av sessionid er frowned upon nå henting av enode data
- Ved registrering savner antall brikker å registrere,
    - grønn knapp må ligge til høyre og hete registrer
    - Må ha instant statusmeldinger på samme dialog om status for registrering
    - Bruk bompengeselskap registrering og drivstoffimport dialog som templates for å logge inn til ladeoperatør

    - Støtter en ekstra side som har logg av status for hver ladebrikke
    - Muligens instant test for feil innlogging og gi beskjed
    - Bruk restletapplicationrouter, med klasse createrouter
    - publicrouter er alle endpunkter som ikke krever pålogging
    - privaterouter det motsatte

i privaterouter.attach /API/Enodewebhook/… attache flere rutere om trengs
sjekk restlet webhook tutorials

createasset extends roledsolution

- Bruk/lag en ny challengeauthenticator for å sette opp sikker identifikasjon for tilkobling og innlogging
- Bruk gpsunits som template for å lagre hvilke ladestasjoner som knyttes til hvilke strømforbruk fra enode.
- restlet har ikke si så det blir å lage en classfactory og bruke getsqlfactory med db navn som parameter
- sharedc når vi jobber med delt db
- getsolutioncustomerbyid viser hvordan en henter med id for en kunde (ikke bruker)

27.04.2022

Demonstrasjon av ferdig produkt

- Ladebrikkeregistrering
    - Registrering av ladebrikke
    - Demonstrere login (med riktig/feil passord)
    - Demonstrere login sjekk ved mangel for detaljer
    - Vise endringer foreslått av ETC
    - Demonstrere registrering av ladebrikke som allerede er registrert
    - Demonstrere sletting av ladebrikke

- Ladeoversikt
    - Tilkobling til enode
    - Demonstrasjon av tabell funksjoner (filtrering o.l)

Kommentarer fra ETC:
Registrer hos ladeselskap:

- Sett passord sensurering for ladeselskap innlogging
- Undefined i error hos automatisk registrering
- Registreringsknapp mens den logger inn, mer informasjon til bruker(ladebrikker registreres)
- Lagre knapp på høyre siden, og skal endre navn til registrer.
- Mer info i selve dialogen, mengde med ladebrikker som blir registrert osv.
- Ikon i dialogvindu registrer hos ladeselskap, burde være samme som funksjonsikon. Akkurat nå er det en setingsikon.
- Refresh siden når dialogen lukkes for å få vist oppdatert tabell
- Detalj vinduet lukker når tabellen laster inn på nytt
- Filtrering - Ladebrikke burde hete RFID
- Fjerne dato filtrering, skru av “opprettet av” som default.
- Antall linjer per side
- Opprettet-av fjernet fra default visning i filter
- Side for ladere, legge til hos enode og se statusen på om de er plugget inn osv. viktig utvidelse og mye man kan skrive om her.
- Trenger mer informasjon på registrering for ladebrikke (lagreknappen kan ikke eksistere hvis vi fortsatt driver å logge inn)
- Kolonne filter ved ladinger
- Legg til funksjonaliteten for å ta imot info om hvilke bil som lader
- Sette laderregistreringsknapp i standardoppsett for fremtidig integrasjon med andre selskaper enn enode, som for eksempel direkte kontakt med private/offentlig ladere og deres selskaper
- Rydde alt vi kan til presentasjonen, gjøre at det ser penere ut og forbrede alt innskrivning, osv..
- Oversettelse til engelsk på ladebrikker
- Vis fram internasjonalisering
- Ikke risiker i presentasjon at uventede ting skjer, f.eks hver eneste input.
- Test å registrere 10 eller flere ladebrikker hos begge ladeselskap og list ut resultater

Ladinger:
Kolonnefilter -> skal kunne skru av og på alle kolonner man ser her.
Den biten de ha mest lyst på -> Den bilen som ladet

Ha med hva man kan utvide senere. Billadere som blir registrert, legge de inn hos CarAdmin, slette dem når de blir slettet, kanskje en oversikt over alle ladere.

Standardoppsett kunne hatt oversikt over alle ladere. Også knappen for registrering av nye ladere.

Kanskje Enode ikke blir “enerådene” kanskje vi trenger mulighet for å snakke med flere enn bare Enode. Utvidelse? Noe sånt.

G Referat fra møter med veileder

Referat fra møte med Frode Haug

07.01.2022

Vi hadde vårt første møte med Frode i dag. Vi snakket litt generelt om bacheloroppgaven, og hva Frode sin rolle er i prosessen. Han fortalte at at vi skal levere en prosjektplan 1. februar. Vi snakket om at i januar er det viktig å jobbe mye med hva oppgaven skal innebære. Frode fortalte og at vi bør lage en gruppeavtale med regler som alle er enige i og signerer. Frode skal sende et pdf dokument med noen punkter som kan være lur å ha med.

12.01.2022

Vi gikk igjennom malen for prosjektplanen og fikk noen råd om hvordan vi burde skrive den. Vi snakket og om hva som skjer dersom vi finner ut at det ikke er mulig å løse oppgaven grunnet manglende tilgang til APIs eller andre måter for å automatisere registrering av RFID brikker, og få ut infomasjon fra ladeboksene. Vi snakket og om hvilke språk vi skal skrive rapporten på, og fant ut at vi skriver bacheloroppgaven på norsk med mindre ETC har et ønske om at vi skriver på engelsk. Vi fikk og mer info om hvor vi skal levere forskjellige deler av oppgaven, og når vi skal levere f.eks statusrapport til Frode.

19.01.2022

Hørte at vi skal dokumentere prosessen av å undersøke ladestasjoner og APIs, og skrive om det i rapporten slik at vi får vist til det arbeidet vi har gjort. Ble enig om å ta opp med oppdragsgiver på neste møte om det vi har funnet, og si hva vi tror er mulig utifra det. Vi forklarte problemene vi har støtt på til Frode.

26.01.2022

Gav statusrapport til Frode, og gikk gjennom førsteutkast prosjektplan.

02.02.2022

Innkalling: Fortelle status på API og oppgaven. Si at vi får innføring 03.02.22.

I dette møte fortalte vi om status for oppgaven, og at vi har gått for å bruke HTML parsing. Vi fikk noen siste kommentarer på prosjektplanen som vi skal endre på. Etter det er den helt klar. Vi fikk navn på noen fra NTNU som er gode på HTML parsing; Mariuz og Johanna Johannesen. Vi fikk høre mer om hvor mye vi bør fokusere på rapporten i tiden framover.

16.02.2022

Innkalling: Fortelle at vi føler vi ligger litt bak, og spørre om tips.

Frode sa at vi burde være litt direkte med ETC om statusen vår, og si at vi føler det er sånn på grunn av at vi ikke har fått god nok innføring i koden, og vi venter mye på eksterne faktorer.

25.02.2022

Innkalling: Fortelle om status nå. Mer fremgang siden sist!

Vi konkluderte med at gruppen ligger bedre an enn statusrapporten vi leverte ga inntrykk av. Eller hadde ingen parter noe viktig å si.

10.03.2022

Innkalling: Hva bør vi dokumentere av prototypen?

Vi gikk gjennom litt av det Frode hadde fortalt i Rapport lynkurs seminaren.

17.03.2022

Innkalling: Gjennomgang av statusrapport

Ingen parter hadde veldig mye å si eller spørre om. Statusrapporten var grei, ingen alarm der. Frode sa at vi må passe på å ha nok å gjøre. Avtalte å møte fysisk kl 09 neste torsdag.

24.03.2022

Innkalling: Vise prototype til veileder

Vi viste prototypen til veileder. Vi hadde og noen spørsmål om database, da ble vi referert til Rune Hjelsvold som er ansvarlig for database emnene på universitetet. Når man ser på det vi har laget virker det ikke veldig imponerende, men det er mye som skjer i backend. Så vi spurte om hvordan vi kan få fram i rapporten at det er mye undersøking og arbeid som ligger bak det vi har utviklet. Vi fikk svar om at vi må bare skrive om det i rapporten, og for oss er det kanskje naturlig å legge til et kapittel som handler om undersøking osv.

07.04.2022

Innkalling: Fortelle status, spørsmål om rapport.

Vi fortalte status til Frode. Da sa vi at vi begynner å bli ganske ferdig med utvikling, og vil bruke tiden framover på testing, funpussing, og rapport. Vi spurte om hvordan vi skal skrive prosessen med å kontakte bedrifter, og ble enig om at en god måte er å liste opp hvem vi kontaktet, også fortelle litt mer rundt det i en tekst under. Vi fikk vite at det ikke er noe minus at produktet ikke er i produksjon, men bare i et testmiljø, og at det går fint at vi ikke har mer enn to ladeselskaper vi kan registrere ladebrikker hos.

21.04.2022

Innkalling: Veiledning på rapportskriving, testing, og API keys.

Vi spurte en flere spørsmål om rapporten, som om vi skal skrive i fortid eller nåtid. Da sa Frode at vi bør helst velge en, men det blir fort litt blanding noen steder. Vi spurte og hva vi burde gjøre med at ETC anbefalte oss å lagre API keys som en konstant i koden. Da fikk vi bekreftet at siden det er det ETC anbefaler kan vi gjøre det, men gjerne vær litt kritisk til det i rapporten. Vi lurte på hvem vi burde teste produktet vårt på i tillegg til Pål i NRK. Da fikk vi svar at vi bør teste på 2 personer til, som enten kan være ansatt i ETC, eller bare noen andre som ikke nødvendigvis er datafolk, for å se om de og skjønner programmet.

28.04.2022

Innkalling: Tilbakemelding på rapporten så langt.

- Muligens bytte om arkitektur og design kapitler for lettere lesing
- Kapittel 2 skal være teori, 3 kravspek, 4 analyse, 5 design, 6 arkitektur, 7 utviklingsprosess.
- Analyse mulig dårlig navn?

Videre fortalte vi om status. Først at vi hadde vist produktet til Pål og Øyvind i CarAdmin og fått positiv respons fra de, i tillegg til gode forslag til forbedringer og videre utvikling. Så fortalte vi at vi skal ha brukertest med Pål fra NRK fredag 29. april.

10.05.2022

Innkalling: Tilbakemelding på rapporten så langt.

- Legge det vi har i analyse inn i teori, skrive om så det passer.
- Fortelle i implementasjon at i GIT ligger det x filer som vi har laget, men i CarAdmin repo var det y filer vi ikke har lagt til.
-

H Referat fra gruppemøter

10.01.2022

Vi avtalte hvilke dager og tider vi skal jobbe, og lagde en kontrakt med grupperegler.

17.01.2022

Snakket om nåværende framgang

- Snakket med oppdragsgiver (Informasjonsmøte med testkunde)
- Hadde lynkurs 1 med Tom og Frode om prosjektplan, grupperegler, arbeidsavtale og leveringstider
- Fordypnings møte om lynkurs 1 med Frode, Jobbet med prosjektplan og kom i kontakt med relevante firma for oppgave.
- Jobbet med prosjektplan (Mål og rammer, og Omfang)
- Ble enige om å bruke overleaf til rapport og prosjektplan.

Plan fremover:

- Fortsette på prosjektplan (Ferdiggjøre Omfang, Prosjektorganisering, Planlegging, Organisering og Gjennomføring).
- Fortsette å prøve å komme i kontakt med bedrifter, og få tak i relevante API og ressurser.
- Presentere prosjektplan til Frode for gjennomgang.
- Ha innførings-møte med ETC, hvor vi lærer mer om det tekniske.
- Skal jobbe med prosjektplan i dag (mandag), men framover skal vi dele dagene inn i halve med rapport og halve lære oss java med spring og restlet.

24.01.2022

Forrige uke:
Forrige uke skrev vi ferdig prosjektplan, så på koding vi må kunne for oppgaven, og fikk litt framgang med å få API fra Eviny. Vi er fornøyd med hvordan vi arbeidet og hva vi har fått gjort. Noe som senker oss ned er at vi må vente på noen ting, som å få svar på ladeselskaper, og venter på innføring i CarAdmin kodebasen.

Plan for neste uke:

- Sjekke deadlines.
- Skrive under grupperegler.
- Skrive under arbeidskontrakt.
- Innlevering av prosjektplan.
- Muligens omformulering av prosjektplan.
- Reservere grupperom.
- Muligens få innføring med ETC i kodebasen.

31.01.2022

Forrige uke:
Vi gjorde ferdig prosjektplan og førte inn i OverLeaf, så mer på Java og andre teknologier, og fikk tilgang til ETC sin kodebase. Siden VI venter mye på svar fra Eviny, og innføring i kodebasen til ETC, føle vi at det ikke var mye mer vi hadde muligheten til å gjøre.

Denne uken:

- Få endelig svar fra Eviny.
- Få innføring i kodebasen til ETC, og sette opp utviklingsmiljø.
- Lage den endelige planen for hva vi skal utvikle, om vi skal bruke APIs, eller bare simulere.
- Vi skal signere og levere kontrakter.

14.02.2022

Forrige uke:
Forrige uke jobbet vi mye med HTML parsing, og å bli kjent med kundebasen til ETC. Det er mye jobb å lære seg kodene til ETC, så det tar mer tid enn planlagt. Vi venter på å få tilgang til Enode API, men vi har fått beskjed om at vi får tilgang. Vi fikk og en ny oppdatering fra Eviny, så det er håp om å få API tilgang der og. Vi har litt mindre framgang enn vi ønsker akkurat nå, men ligger ikke ille ute ennå.

Denne uka:

- Sette opp HTML parsing prosjekt eksempel for de forskjellige ladeselskaper vi skal bruke.
- Statusmøte med ETC.
- Statusmøte med veileder.
- Ha et produkt som kan vises fram i neste mandagsmøte (Til å logge inn i ladestasjon nettsider).
- Begynne med simulering av API

21.02.2022

Innkalling: Status for prosjektet, og veien videre.
Forrige uke: Ble ferdig med å undersøke muligheter, og satt oss inn i CarAdmin kodebasen. Det var mer jobb å lære CarAdmin enn vi hadde sett for oss.

Denne uka:

- Jobbe med første iterasjon av kravspek
- Føre konkret backlog i gitlab
- Sette opp rammeverk for frontend til backend til database kommunikasjon
- Koble til Enode gjennom CarAdmin
- Legge ladebrikker til i Eviny/bilkraft

03.03.2022

Innkalling: Hva vi skal gjøre den kommende uken.
Forrige uke: Vi nådde målene vi hadde for forrige uke, og er fornøyd med framgangen vi har nå. Vi tror vi blir ferdig med en prototype til planlagt dato.

Denne uken:
- Vise strømforbruk fra Enode i en rapport på CarAdmin siden.
- Vise data fra EON i en rapport på CarAdmin siden.
- Implementere egne “massefunksjoner” i CarAdmin
- Tilknytte kjøretøy til ladebrikke i ladebrikke rapporten

09.03.2022

Innkalling: Status nå, og hva som må gjøres før prototypen er klar.
Forrige uke: Fikk til å hente data fra Enode uten webhooks, og satt opp webhooks fre Enode i et test miljø,egen massefunksjon fungerer, klarte å logge inn på E.ON gjennom CarAdmin.

Denne uken
- Registrere ladebrikker på E.ON
- Jobbe med første utkast til rapport
- Få en offentlig url fra CarAdmin som kan ta imot webhooks eventene fra Enode
- Bli klar med prototypen til kommende Fredag
- Slå sammen ferdig kode fra forskjellige test-miljøer inn til master.

17.03.2022

Innkalling: Status, og arbeid den kommende uken.
Forrige uke: Ferdig med prototype.

Denne uken:
- Jobbe med rapport.
- Finpusse deler av prototypen.
- Enode webhooks.
- Presentere prototype for ETC.

23.03.2022

Innkalling: Hvordan vi skal vise prototypen i dag, plan for ukens sprint.
Forrige uke: Finpusset prototypen og funksjonalitet.

Denne uken:

- Lagre Enode data fra et felles sted uavhengig av bedriften.
- Rapport.
- Vise hvor hver enkelt ladebrikke er registrert og ikke.
- Lagre innloggingsinfo til de forskjellige ladeselvskapene.
- Prototype review, hva vi mangler, og hva vi skal gjøre bedre.
- Lage diagram for å framstille systemet vårt.
- Undersøke ladebrikke registrering på fortum.

30.03.2022

Innkalling: Planlegge neste sprint

Denne uken:

- Fikse autorisering av Enode api kall.
- Bruke Restlet til Enode api kall.
- Finpusse og ferdigstille strømforbruk delen.
- Gjøre om på logginsystem til ladeselskap så det møter ETC sine krav.
- Bruke ny tag til loginsystemet.
- Lage en progressbar som viser hvor mange ladebrikker som har blitt registrert hos ladeselskap.

07.04.2022

Innkalling: Planlegge kommende sprint
Påskeferien kommer til å gjøre at det blir litt mindre jobbing enn på en vanlig uke.

- Unit testing av funksjoner
- Rapportskriving
- API response parsing

21.04.2022

Innkalling: Gå gjennom “ready for review” i issue boards, planlegge kommende sprint.

Forrige sprint:

- Ble ferdig med utvikling av produktet.

Denne uken:

- Planlegge kapittelfordeling
- Skrive rapport
- Gjøre ferdig testing

28.04.2022

Innkalling: Sprint review og sprint planning

Forrige sprint:
Vi skrev rapport, lagde flere tester, og fikset ulike bugs.

Denne sprinten:
- Gjennomføre brukertest med NRK og CarAdmin.
- Skrive rapport.

05.05.2022

Innkalling: Diskutere framgang og status for rapporten.

Forrige sprint:
Vi gjennomførte brukertester med NRK, og avtalte brukertester med CarAdmin ansatte neste uke.

Denne sprinten:
- Gjennomføre brukertest CarAdmin ansatte.
- Skrive rapport.

12.05.2022

Innkalling: Diskutere framgang og status for rapporten.

Forrige sprint:
Vi gjennomførte brukertester med ansatte i CarAdmin, og skrev på rapporten.
Denne sprinten:
Ferdigstille rapport, gjøre klar kode og andre filer til innlevering.

I Statusrapport 1

Statusrapport 1

Fullført arbeid

- Undersøkt muligheter rundt APIs.
- Lært oss HTML parsing
- Laget en plan for hvordan vi skal løse problemet med det vi har tilgjengelig av APIs osv.
- Satt oss inn i CarAdmin kodebase.
- Fått klart utviklingsmiljø.

Ikke fullført arbeid

- Vi har ikke begynt å kode selve løsningen, vi har kodet i CarAdmin miljøet for å teste ulike løsninger vi har sett for oss.

Vi har ikke overskredet noen frister ennå, men fristen for å ha første prototype klar til 11. mars kan bli vanskelig å overholde ettersom vi har kommet senere i gang med selve arbeidet enn planlagt. Vi har kommet senere i gang fordi det tok mye lengre tid å undersøke mulige løsninger enn vi trodde det ville ta. Mange av bedriftene vi har vært i kontakt med bruker lang tid på å svare, og vi har derfor måtte vente på de. I tillegg var det mye vanskeligere og mer tidkrevende å sette seg inn i CarAdmin sin kodebase enn vi hadde forventet. Vi har mye igjen å lære om CarAdmin, men vi kan nok om det som er relevant for oppgaven vår til å begynne å kode.

Planlegging

Vi har gjort grundig undersøkelse av hva som er mulig, og er klar for å begynne på utviklingen av selve programmet nå. Som nevnt over har vi kommet senere i gang enn vi planlagte. Derfor er det urealistisk at vi har en prototype klar til 11. mars, men vi tror vi vil ha en del av den grunnleggende funksjonaliteten på plass til da.

Organisering av gruppens ansvarsområder.

Organiseringen av gruppen er fremdeles den samme som er beskrevet i prosjektplanen. Dette synes vi fungerer bra.

Klargjøring av problemstilling.

Vi skal bruke Enode API til å hente ut strømforbruket til hjemmeladere. Foreløpig er det ingen mulighet for å få strømforbruket fra de eksterne laderne, ettersom vi ikke har fått noen APIs fra de. Vi skal bruke HTML parsing til å registrere ladebrikker hos ulike ladeselskaper. Det er viktig for ETC at vi strukturerer koden slik at det er enkelt å bytte fra HTML parsing til å bruke APIs, dersom de blir tilgjengelig i fremtiden.

Rapportskriving
Vi har ikke skrevet noe rapport utenom det som er skrevet i prosjektplanen, men vi dokumenterer arbeidet vi gjør underveis, slik at det ikke blir problemer med å huske hva vi har gjort i senere tid.

2 Oppsummering av punktene

Vi har jobbet mye med å undersøke og teste muligheter. Dette har tatt lengre tid enn vi så for oss, som gjør at vi ligger litt lengre bak enn der vi ønsker å være. Vi har tatt dette opp med oppdragsgiver. De var ikke bekymret, og mente at dette er helt normalt, noe som gir oss håp om at oppgaven vil gå bra til tross for en treg start. Vi har en klar plan for oss nå på hvordan vi skal løse oppgaven.

3 Muligheter, trusler, problemer

Siden vi får bruke Enode sin API er det mulig å spare tid og ressurser på oppgaven, fordi vi trenger bare å forholde oss en en API, i stede for alle de ulike hjemmeladerene sin API. Enode er en API som binder sammen APIene til flere ulike hjemmelader.

Trusler

- Tid, siden vi har kommet litt sent i gang med å utvikle.
- Det tar lang tid å utvikle i CarAdmin kodebasen, siden den er komplisert, og det er mange hjemmelagde biblioteker vi må lære oss.
- Hvis noe på nettsiden til et ladeselskap blir endret som gjør at HTML parsing ikke er mulig. F.eks 2 faktor autentisering, eller noe annet som krever en handling fra et menneske.

4 Motivasjon

Gruppen har fremdeles høy motivasjon, og fungerer bra sammen. Vi har til nå jobbet sammen 4 ganger i uken, med en blanding av fysisk og online. Det er foreløpig ikke noe tegn til forhold som frustrerer medlemmer av gruppen.

5 Kontakt med oppdragsgiver og veileder

Kontakten med oppdragsgiver er bra. Vi har vært på kontoret dere for å jobbe, og framover skal vi sitte hos ETC hver onsdag. Vi synes vi burde fått en litt bedre innføring i kodebasen til CarAdmin, men etter vi tok det opp med dem har vi funnet en løsning vi tror vil hjelpe.

Vi er fornøyd med kontakten vi har med veileder. Det har ikke vært mange spørsmål ennå, ettersom det meste vi har lurt på har vært rundt CarAdmin, APIs, og HTML parsing. Det vil nok bli flere spørsmål når vi begynner å skrive rapporten.

J Statusrapport 2

Statusrapport 2

Fullført arbeid

- Hente strømforbruk fra Enode manuelt med en knapp
- Registrere ladebrikker i CarAdmin
- Registrere ladebrikker hos bilkraft (Eviny)

Ikke fullført arbeid

- Hente strømforbruk fra Enode med webhooks
- Registrere ladebrikker hos E.on
- Begynne på rapportskriving
- Testing av kode

Alt i alt ligger vi godt an på utviklingen av programvaren. Uten APIs har det vært vanskelig å koble programvaren til flere ladestasjoner enn Eviny.

Planlegging
Planen framover blir å hente strømforbruk fra Enode når man får en notifikasjon fra en webhook. Vi må og skrive tester for koden vi har laget.

Organisering av gruppens ansvarsområder.
Organiseringen er den samme som vi begynte med. Akkurat nå jobber alle med utvikling.

En endring fra forrige statusrapport er når vi har møter. Nå har vi gruppemøter og statusmøter med ETC hver onsdag, og møte med veileder hver torsdag, med mindre noe annet blir avtalt.

Klargjøring av problemstilling
Problemstillingen har ikke endret seg siden forrige statusrapport. Vi bruker Enode API til å hente strømforbruk fra hjemmeladere, og HTML parsing til å registrere ladere hos eksterne ladeselskaper.

Rapportskriving
Vi skal begynne å skrive på rapporten denne uken (uke 11), og planlegger å ha en førsteutkast før påskeferien begynner.

2 Oppsummering av punktene

Vi har fullført de fleste oppgavene vi har hatt som mål å gjør til prototypen vår, med unntak av webhooks fra Enode. Planen framover blir å fikse enode webhooks, og lage tester. Problemstillingen har ikke endret seg fra forrige statusrapport. Vi skal

snakke med ETC etter vi har vist prototypen vår for å få innspill på hvordan veien videre ser ut.

3 Muligheter, trusler, problemer

Akkurat nå ser det ut som vi kan bli ferdig mye av hovedfunksjonaliteten innen et bra tidsrom. Da kan vi bruke tid på å finpusse, teste, og høre om ETC har mer funksjonalitet de ønsker at vi skal implementere.

Trusler

- Komme for seint i gang med rapportskriving.

4 Motivasjon

Gruppen har opprettholdt god motivasjon i prosjektet, og alle er klar for å legge inn god innsats mot neste fase av prosjektet.

5 Kontakt med oppdragsgiver og veileder

Vi har god kontakt med oppdragsgiver gjennom at vi sitter på kontoret hver onsdag for å jobbe, i tillegg til at det er lav terskel for å sende mailer dersom vi lurer på noe. Vi har ikke hatt veldig mye kontakt med veileder den siste perioden fordi hovedfokuset har vært på utvikling, og tekniske spørsmål går til utviklerne i ETC. Fremdeles har det vært korte statusmøter nesten hver uke, med unntak av noen uker hvor vi har bedt om å avlyse møtet. Vi har fått all den hjelpen av veileder vi føler har vært nødvendig så langt.

K Timelogg

Sindre		
Dato	Timer	Hva ble gjort
06.01.2022	2	Møte emd gruppen og derreter etc.
07.01.2022	1	Møte med veileder.
10.01.2022	3,5	Skrev mail til ladestasjon firma
11.01.2022	1	Lynkurs i prosjektmal
12.01.2022	8	Møte med veileder. Lest samarbeidsavtale. Funnet APIs. Jobbet med prosjektplan
13.01.2022	8	Kort møte med oppdragsgiver for oppklarende spørsmål. Ringe/mailte bedrifter for API. Prosjektplan
14.01.2022	7,5	Snakket med Easee. Skrev på prosjektplan.
17.01.2022	8	Møte med ETC, prosjektplan, mail til Zaptec, undersøking av API og handshake.
19.01.2022	8	Møte med veileder. Jobbet med prosjektplan. Ringt bedrifter angående API bruk.
20.01.2022	8	Jobbet med prosjektplan. Deltok på presentasjon av ETC/CarAdmin
21.01.2022	8	Finleste prosjektplan. Jobbet med spring, java, og tomcat. Fikk kontakt med Eviny.
24.01.2022	8	Møte med ETC. Jobbet med servlet, restlet.
25.01.2022	1	Undersøkte Enode, tok kontakt med dem.
26.01.2022	7	Møte med Frode. Jobbet med prosjektplan. Sett på kildekode fra ETC.
27.01.2022	7	Finpusset prosjektplan
28.01.2022	7	Ferdigstillte prosjektplan. Førte inn i overleaf.
31.01.2022	8	Signerte kontrakter. Møte med oppdragsgiver. Jobbet med API dokumentasjon.
02.02.2022	8	Møte med veileder. Skrevet om prosjektplan. Øvd på Java.
03.02.2022	7	Fikk hjelp med å sette opp caradmin og utviklingsmiljøet. Mer Java.
04.02.2022	8	Jobbet med HTML parsing og Java.
07.02.2022	8	Jobbet med CarAdmin kodebasen
09.02.2022	7,5	Øvde på html parsing
10.02.2022	8	Jobbet med html parsing og med CarAdmin kodebasen.
11.02.2022	5	Jobbet med ETC kodebase.
14.02.2022	7	Laget API for å simulere registrering av ladebrikker.
16.02.2022	8	Sendt ulike mailer, veiledningsmøte, jobbet med Enode.
17.02.2022	7,5	Enode, CarAdmin kodebase
18.02.2022	8	Enode

21.02.2022	8	Statusrapport, api kall, enode
23.02.2022	8	Implementere enode api tilgang
24.02.2022	8,5	Enode API i CarAdmin
25.02.2022	8	Sende data mellom jsp og ajax filer i CarAdmin.
28.02.2022	7	Enode
02.03.2022	6	Enode
03.03.2022	8	Laget rapport i CarAdmin med charging info
04.03.2022	7	Enode webhooks
05.03.2022	6	Enode webhooks
06.03.2022	5	Legge til strømforbruk i database
07.03.2022	7	Snakket med Enode, hente strømforbruk fra enode
09.03.2022	8	Gruppemøte, filtrering etter dato
10.03.2022	8	filtrering av dato
11.03.2022	7,5	webhook
14.03.2022	7,5	webhook
16.03.2022	8	Impimentering av enode webhook
17.03.2022	8	Fiksing av bugs med henting av ladinger
18.03.2022	7	merge inn i developement
21.03.2022	7,5	utforske løsninger på problem med henting av strøm data
23.03.2022	8	løste problem med åpning av enode link
24.03.2022	8	Rapport
25.03.2022	7	Viste fram prototype til ETC
28.03.2022	7,5	Restlet API for å løse problem med hentign av strøm fra Enode
29.03.2022	5	Restlet API, løste problem med Docker
30.03.2022	8	Fullførte henting av data som fungerer uavhengig av hvilke bedrift ladingen hører til.
31.03.2022	7	Authentisering av enode webhook
01.04.2022	8	Authentisering av enode webhook
04.04.2022	8	Ferdig med Enode Api authorisering
06.04.2022	8	Gjorde Enode knapp fin,
07.04.2022	8	Sortering etter dato i lade tabell feridg, rydding i kode

08.04.2022	8	Action knapp som viser en oppsummering av innholdet i ladinger tabellen /m hjelp fra Elliot
12.04.2022	6	Rapport
13.04.2022	1	Rapport
19.04.2022	4	Testing, gikk gjennom Enode kodene og merger med development
20.04.2022	8	Testing, kommentering, fiksing av bug med sortering etter dato.
21.04.2022	7	Rapport
22.04.2022	7	Rapport
25.04.2022	7	Rapport
27.04.2022	6	Visete produkt til oppdragsgiver, fikse ulike bugs
28.04.2022	7	Møte med veileder, rapport, planlegge brukertest
29.04.2022	5	Brukertest med NRK, rapport
02.05.2022	7	Rapport
03.05.2022	2	Rapport
04.05.2022	8	Brukertest med ETC, rapport
05.05.2022	7	Rapport
06.05.2022	8	Rapport
07.05.2022	4	Rapport
09.05.2022	3	Rapport
10.05.2022	8	Møte med Frode, rapport
11.05.2022	9	Rapport
12.05.2022	8	Rapport
13.05.2022	8	Rapport
14.05.2022	5	Rapport
15.05.2022	5	Rapport
16.05.2022	8	Rapport
17.05.2022	8	Rapport
18.05.2022	12	Rapport, gjøre klar bachelor til innlevering
19.05.2022	12	Rapport, gjøre klar bachelor til innlevering
Sum	593,5	

Rihards		
Dato	Timer	Hva ble gjort
06.01.2022	2	Møte med gruppen og ETC
07.01.2022	1	Møte med Frode
10.01.2022	3,5	Gruppemøte
11.01.2022	1	Lynkurs i prosjektmål
12.01.2022	8	Møte med veileder. Lest samarbeidsavtale. Funnet APIs. Jobbet med prosjektplan
13.01.2022	8	Kort møte med oppdragsgive. Kontakted bedrifter for API. Prosjektplan
14.01.2022	7,5	Jobbet med prosjektplan. Kontaktet Easee
17.01.2022	8	Møte med ETC, prosjektplan, mail til Zaptec, undersøking av API og handshake.
19.01.2022	8	Møte med veileder. Jobbet med prosjektplan. Ringt bedrifter angående API bruk.
20.01.2022	8	Jobbet med prosjektplan. Deltok på presentasjon av ETC/CarAdmin
21.01.2022	8	Finleste prosjektplan. Jobbet med spring, java, og tomcat. Fikk kontakt med Eviny.
24.01.2022	8	Møte med ETC. Jobbet med servlet, restlet.
26.01.2022	7	Møte med Frode. Jobbet med prosjektplan. Sett på kildekode fra ETC.
27.01.2022	7,5	Uforsket mer av kildekoden fra ETC. Lastet ned dependencies. Jobbet med Java.
28.01.2022	7	Ferdigstilte prosjektplan. Overførte i overleaf. Mer øving i Java.
31.01.2022	8	Signerte kontrakter. Møte med oppdragsgiver. Jobbet med API dokumentasjon.
02.02.2022	8	Møte med veileder. Øvd på Java.
03.02.2022	7	Fikk hjelp med å sette opp caradmin og utviklingsmiljøet. Mer Java.
04.02.2022	8	Jobbed i ETC sine systemer.
07.02.2022	8	Jobbet med CarAdmin kodebasen
09.02.2022	7,5	Øvde på html parsing
10.02.2022	8	Jobbet med html parsing og med CarAdmin kodebasen.
11.02.2022	7	Jobbet med CarAdmin sin kodebase
14.02.2022	7	Jobbet med å kode og begynne å sette opp noe eget i CarAdmin.
16.02.2022	6	Jobbet med å kode og begynne å sette opp noe eget i CarAdmin.
17.02.2022	8	Jobbet med å utvikle og utforske koden videre i CarAdmin.
18.02.2022	7	Utvikling
22.02.2022	4	Ladebrikke tabell

23.02.2022	7	Ladebrikke tabell
24.02.2022	10	UI for å legge til ladebrikke
25.02.2022	10,5	UI for å legge til ladebrikke
02.03.2022	6	Tilknytte bil til ladebrikke
03.03.2022	8	Tilknytte bil til ladebrikke
04.03.2022	7	Tilknytte bil og bruker til ladebrikke
08.03.2022	3	Tilknytte bil og bruker til ladebrikke
09.03.2022	8	Filtrering og søke funksjoner for ladebrikker
10.03.2022	7	Bug fixing og merging med Elliot
11.03.2022	9	UI for å legge knytte ladeselskap
15.03.2022	2	Database for lagring og tilknytning av ladeselsaper
16.03.2022	7	Database for lagring og tilknytning av ladeselsaper
17.03.2022	8,5	Vise tillknytede ladeselskaper i ladebrikke tabellen
18.03.2022	6,5	Komentering og rydding av koden, merging inni development
22.03.2022	2	Vsing av status for tillknytning av ladeselskaper
23.03.2022	8,5	Vsing av status for tillknytning av ladeselskaper
24.03.2022	8	Vsing og oppdatering av status for tillknytning av ladeselskaper
25.03.2022	6	Status oppdatering av ladeselskaper
29.03.2022	3	Fikset noe bugs
30.03.2022	7,5	Pusset opp koden
31.03.2022	7	Komentering og rydding av koden
01.04.2022	8	Rapport skriving
04.04.2022	2	Rapport skriving
06.04.2022	6	Rapport skriving
07.04.2022	6	Rapport skriving
08.04.2022	8,5	Rapport skriving
18.04.2022	8	Rapport skriving
20.04.2022	8	Raport skriving og modeller
21.04.2022	7,5	Raporrt skriving, modeller og exportering av ladebrikker til excel
22.04.2022	7,5	Raport skriving og modeller

27.04.2022	6,5	Raport skriving
28.04.2022	7	Rapport
29.04.2022	7	Brukertest med NRK, rapport
04.05.2022	8	Brukertest med ETC, rapport
05.05.2022	8	Rapport
06.05.2022	8	Rapport
10.05.2022	1	Møte med Frode.
11.05.2022	8	Rapport
12.05.2022	8	Rapport
13.05.2022	7	Rapport
14.05.2022	3	Rapport
15.05.2022	3	Rapport
16.05.2022	4	Rapport
17.05.2022	5	Rapport
18.05.2022	12	Rapport, gjøre klar bachelor til innlevering
19.05.2022	12	Rapport, gjøre klar bachelor til innlevering
Sum	496,5	

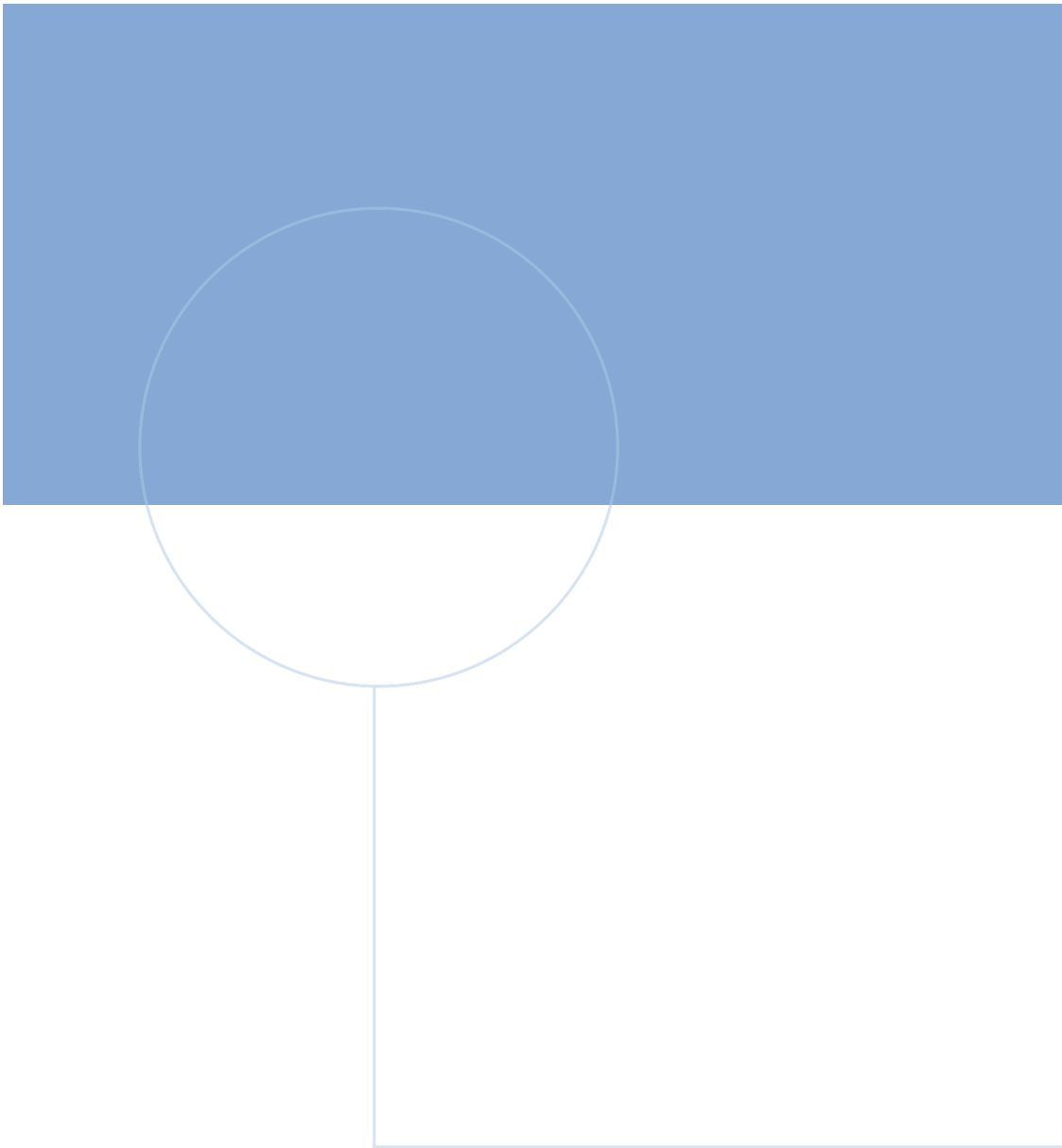
Elliot		
Dato	Timer	Hva ble gjort
06.01.2022	2	Møte emd gruppen og derreter etc.
07.01.2022	1	Møte med Frode Haug
10.01.2022	4	Møte med Gruppen
11.01.2022	1	Lynkurs i prosjektmal
12.01.2022	8	Møte med veileder. Lest samarbeidsavtale. Funnet APIs. Jobbet med prosjektplan
13.01.2022	8	Kort møte med oppdragsgiver for oppklarende spørsmål. Ringe/mailte bedrifter for API. Prosjektplan
14.01.2022	7,5	Snakket med Easee. Skrev på prosjektplan.
17.01.2022	8	Møte med ETC, prosjektplan, mail til Zaptec, undersøking av API og handshake.
19.01.2022	8	Møte med veileder. Jobbet med prosjektplan. Ringt bedrifter angående API bruk.
20.01.2022	8	Jobbet med prosjektplan. Deltok på presentasjon av ETC/CarAdmin
21.01.2022	8	Finleste prosjektplan. Jobbet med spring, java, og tomcat. Fikk kontakt med Eviny.
24.01.2022	8	Møte med ETC. Jobbet med servlet, restlet.
26.01.2022	7	Møte med Frode. Jobbet med prosjektplan. Sett på kildekode fra ETC.
27.01.2022	7,5	Finpusset prosjektplan og lærte mer om Java
28.01.2022	7	Gjorde ferdig prosjektplan, og overførte den inn i Overleaf.
31.01.2022	8	Signerte kontrakter. Møte med oppdragsgiver. Jobbet med API dokumentasjon.
02.02.2022	8	Møte med veileder. Skrevet om prosjektplan. Øvd på Java.
03.02.2022	7	Fikk hjelp med å sette opp caradmin og utviklingsmiljøet. Mer Java.
04.02.2022	8	Jobbed i ETC sine systemer, jobbet med HTML parsing.
07.02.2022	8	Jobbet med CarAdmin kodebasen
09.02.2022	7,5	Øvde på html parsing
10.02.2022	8	Jobbet med html parsing og med CarAdmin kodebasen.
11.02.2022	7	Jobbet med CarAdmin sin kodebase
14.02.2022	8	Laget API for å simulere registrering av ladebrikker.
15.02.2022	8	Forbedret API for å simulere registrering av ladebrikker, og brukte det i CarAdmin.
16.02.2022	4	Fortsatte arbeid ned API, og fikk lagt inn deler av dataen fra API i CarAdmin sin database.
17.02.2022	4	Dro til ETC for å få mer innføring i systemet. Jobbet med systemet deres hele dagen.
18.02.2022	8	HTML parsing øving
21.02.2022	2	Begynte med HTML parsing mot Bilkraft
22.02.2022	5	Logget inn i Bilkraft bed hjelp av HTML parsing
23.02.2022	8	Fikk hentet ut, og lagt til nye ladebrikker til Bilkraft ved hjelp av HTML parsing
24.02.2022	7,5	Begynte å implementere koden fra parsing inn i en funksjon i carAdmin.
25.02.2022	8	Jobbet videre på å lage en funksjon i carAdmin som lar brukeren legge inn ladebrikker til en spesifik bil
01.03.2022	4	Begynte å se på massefunksjon for registrering av biler hos ladeselskap.
02.03.2022	2	Så mer på massefunksjoner
03.03.2022	6	Så mer på massefunksjoner

04.03.2022	7	Fikk hentet ut bilnummer ved hjelp av massefunksjonen, og skal i løpet av de neste dagene sende bilnummeret til Bilkraft gjennom et dialogvindu
08.03.2022	7	Gjorde ferdig massefunksjonen, så den nå sender bilnummer og rfidnummer gjennom et dialogvindu
09.03.2022	4	Fjernet unødvendige innskrivningsfelt i dialogvindu, og gjorde alt klart til å merges med master branch
10.03.2022	8	Merging av branches, og feilsøking etter et docker problem oppstod.
11.03.2022	8,5	Jobbet med EON parsing. Fikk registrert en ladebrikke gjennom postman
14.03.2022	2	Jobbet med Ladebrikke registrering funksjon i lade brikke detaljer vinduet
15.03.2022	9	Gjorde ferdig ladebrikke registrering funksjon i ladebrikke detaljer vindu, hjalp Sindre med CarAdmin cookies i postman, og begynte på filtrering etter måneder i ladere tabell
16.03.2022	6	Fortsatte på filtrering etter måneder i ladere tabell, og så mer på E.ON parsing.
17.03.2022	8	Jobbed med E.ON parsing. Kan nå logge inn og legge inn en ny ladebrikke. Fikset en bug hvor ladebrikke og bilnummer hadde blitt sendt inn i feil rekkefølge til bilkraft.
18.03.2022	7	Jobbet med checkboxer som lar brukeren bestemme ladeselskap de vil registrere ladebrikken til.
22.03.2022	3	Checkboxer er helt ferdig.
23.03.2022	9	Begynte arbeid på en funksjon som skal la brukeren logge inn i Bilkraft og E.ON gjennom CarAdmin.
24.03.2022	12	Jobbet med innloggingsystemet for ladebrikkeselskap, og ble ferdig med hovedfunksjonaliteten.
25.03.2022	5	Begynte på testing på innloggingsystemet, for å prøve å finne bugs, statuskoder fra ladeselskapene, og mer.
29.03.2022	4	Started på forandringer på innloggingsystemet for ladebrikkeselskap, etter vi fikk tilbakemelding fra ETC.
30.03.2022	8	Fortsatte arbeid på innloggingsystemet, kan nå lagre loggininfo men er buggy.
31.03.2022	7	Fikset bugs med innloggingsystemet, bilkraft funker.
01.04.2022	8	Fikset bugs med innloggingsystemet, EON funker.
05.04.2022	2	Innloggingsystemet er komplett og alt funker.
06.04.2022	8	Lagde en progressbar som går opp når ladebrikker blir lagret hos ladeselskap.
07.04.2022	8	Fortsatte arbeid på progressbar
08.04.2022	8,5	Hjalp Sindre. Gjorde ferdig progress bar design. Begynte på bilkraft sletting av ladebrikker gjennom caradmin.
14.04.2022	5	Begynte på EON sletting av ladebrikker gjennom caradmin.
15.04.2022	5	Ferdig med EON sletting av ladebrikker gjennom caradmin.
19.04.2022	8	Jobbet med testing
20.04.2022	3,5	Jobbet på rapport, sendte mailer, lagde zoom bruker for å ha brukerundersøkelse med NRK
21.04.0202	7	Jobbet med testing, gruppemøte, møte med veileder
22.04.2022	8	Jobbet med testing, og rapport
26.04.2022	3	Rapportskriving testing
27.04.2022	6,5	Rapportskriving implementasjon
28.04.2022	7,5	Rapportskriving implementasjon og småforandringer etter feedback fra ETC
29.04.2022	7	Brukertest med NRK, rapport
03.05.2022	3	Fikset bug hvor ladeselskap stod som undefined, fjernet inputfelt som var unødvendig
04.05.2022	8	Brukertest med ETC, rapport, fikset bug vi fant under brukertesten og bug med engelsk oversettelse hvor bruk av anførselstegn i oversettelse krashet siden
05.05.2022	8	Rapport
06.05.2022	8	Rapport
10.05.2022	1	Møte med Frode.
11.05.2022	8	Rapport

12.05.2022	8	Rapport
13.05.2022	8	Rapport
14.05.2022	5	Rapport
15.05.2022	3	Rapport
16.05.2022	8	Rapport
17.05.2022	8,5	Rapport
18.05.2022	12	Finpuss rapport
19.05.2022	12	Finpuss rapport
Sum	539	

Abdulhadi		
Dato	Timer	Hva ble gjort
06.01.2022	2	Møte med med gruppe og ETC
07.01.2022	1	Møte med veileder
10.01.2022	4	Møte med gruppe
11.01.2022	1	Lynkurs i prosjektmål
12.01.2022	8	Møte med veileder. Lest samarbeidsavtale. Funnet APIs. Jobbet med prosjektplan
13.01.2022	8	Kort møte med oppdragsgiver for oppklarende spørsmål. Ringe/mailte bedrifter for API. Prosjektplan
14.01.2022	8	Snakket med Easee. Skrev på prosjektplan.
17.01.2022	8	Møte med ETC, prosjektplan, mail til Zaptec, undersøking av API og handshake.
19.01.2022	8	Møte med veileder. Jobbet med prosjektplan. Ringt bedrifter angående API bruk.
20.01.2022	7	Jobbet med prosjektplan. Deltok på presentasjon av ETC/CarAdmin
21.01.2022	8	Finleste prosjektplan. Jobbet med spring, java, og tomcat. Fikk kontakt med Eviny.
24.01.2022	8	Møte med ETC. Jobbet med servlet, restlet.
26.01.2022	7	Møte med Frode. Jobbet med prosjektplan. Sett på kildekode fra ETC.
27.01.2022	7	Finpusset prosjektplan
28.01.2022	6,5	Ferdigstillte prosjektplan. Førte inn i overleaf.
31.01.2022	8	Signerte kontrakter. Møte med oppdragsgiver. Jobbet med API dokumentasjon.
02.02.2022	8	Møte med veileder. Opplæring i Java konsepter
03.02.2022	7	Fikk hjelp med å sette opp caradmin og utviklingsmiljøet. Mer Java.
04.02.2022	8	Jobbet med CarAdmin kodebasen
07.02.2022	8	Jobbet med Java og CarAdmin kodebasen.
09.02.2022	7,5	Øvde på html parsing
10.02.2022	8	Jobbet med html parsing og med CarAdmin kodebasen.
11.02.2022	6	Jobbet med ETC kodebase.
14.02.2022	8	Jobbet html parsing.
16.02.2022	4	Jobbet html parsing.
17.02.2022	2	Jobbet html parsing.
18.02.2022	8	Jobbet html parsing.
23.02.2022	8	Jobbet med innlogging til EON for å automatisk registrering av ladebrikker
24.02.2022	8	Jobbet med iterasjon av kravspek
25.02.2022	8	Jobbet med iterasjon av kravspek, statusmøte
27.02.2022	4	Kodebase forståelse og implementering av HTML parsing til kodebase for EON innlogging
02.03.2022	3	Lynkurs 2 i rapportskriving, Implementasjon av loginEON HTML parsing
03.03.2022	8	Implementasjon av loginEON HTML parsing, trengte endring av HTML request metode for å tilpasse asynkrone kall
06.03.2022	8	Implementasjon av loginEON HTML parsing, innlogging komplett. Trenger mer arbeid på brikkeregistrering grunnet cookies ikke overføres videre på neste kall

09.03.2022	8	Implementert loginEON parsing i klientside for å teste.
10.03.2022	8	Jobbet med bygging av rapportstruktur i overleaf
11.03.2022	2	Hjalp i teams
15.03.2022	8	Rapport oppsett i LATEX
16.03.2022	8	Rapport oppsett i LATEX
17.03.2022	8	Rapport oppsett i LATEX
18.03.2022	8	Rapport E.ON html parsing
23.03.2022	8	Flowchart design av ladebrikkeregistrerings logikk
25.03.2022	8	Prototype demonstrasjon for ETC, Rapport, flowchartdesign
27.03.2022	5	Innholdsfortegnelse oppsett i LATEX, begynt på sekvensdiagram
30.03.2022	4	Jobbet med sekvensdiagram for ladebrikkeregistrering
31.03.2022	4	Jobbet med sekvensdiagram for energiforbruksrapport
01.04.2022	8	La til pdf inkluderings funksjon i LATEX, slik at Prosjektplan og andre vedlegg kunne simpelt inkluderes som filer
03.04.2022	4	Rapportskriving research
06.04.2022	8	Rapportskriving div
07.04.2022	8	Rapportskriving Utviklingsprosess
13.04.2022	8	Rapportskriving Utviklingsprosess og Arkitektur/Design
14.04.2022	8	Rapportskriving Arkitektur/Design
21.04.2022	8	Rapportskriving Analyse
22.04.2022	8	Rapportskriving Analyse
27.04.2022	6,5	Rapportskriving Analyse
28.04.2022	7	Rapportskriving Analyse, Statusmøte med ETC
29.04.2022	8	Rapportskriving, modelltegning, Brukertest med NRK
04.05.2022	8	Rapportskriving, modelltegning
05.05.2022	8	Div rapportskriving, modelltegning og innholdsplanlegging
06.05.2022	8	Rapport
10.05.2022	1	Møte med Frode.
11.05.2022	8	Rapport
12.05.2022	8	Rapport
13.05.2022	8	Rapport
15.05.2022	1	Rapport
16.05.2022	6	Rapport
18.05.2022	8,5	Finpuss rapport
19.05.2022	9	Finpuss rapport
Sum	455	



NTNU

Kunnskap for en bedre verden