Bjørn Inge Kulsveen
Andreas Maurud Prøven
Marius Dahlen Østli

Lagermodul

Bacheloroppgave i ingeniørfag, data
Veileder: Frode Haug
Mai 2019

Bacheloroppgave

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

NTNU
Kunnskap for en bedre verden

Bjørn Inge Kulsveen
Andreas Maurud Prøven
Marius Dahlen Østli

Lagermodul

Bacheloroppgave i ingeniørfag, data
Veileder: Frode Haug
Mai 2019

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

NTNU
Kunnskap for en bedre verden

Sammendrag av Bacheloroppgaven

Tittel:	**Lagermodul**
Dato:	20.05.2019
Deltakere:	Bjørn Inge Kulsveen Andreas Maurud Prøven Marius Dahlen Østli
Veiledere:	Frode Haug
Oppdragsgiver:	Electric Time Car AS
Kontaktperson:	Dag L. Solhaug
Nøkkelord:	AJAX, Java, JavaScript, JSP, CarAdmin
Antall sider:	85
Antall vedlegg:	11
Tilgjengelighet:	Åpen

Sammendrag: I store bedrifter er det vanlig å ha firmabiler eller andre kjøretøy som benyttes av ansatte i forbindelse med arbeidsoppdrag. Electric Time Car AS har et eksisterende system med navn CarAdmin som brukes til å administrere disse kjøretøyene. Mange av bedriftene som benytter seg av dette har interne verksteder, delelagre eller dekkhotell. Derfor var det ønskelig å utvide dette systemet med en lagermodul. Lagermodulen ble utviklet ved hjelp av teknologiene AJAX, HTML, Java, JavaScript og JSP. Denne utvidelsen inneholder en oversikt over lagerbeholdning, register over alle lagre og bestilling av varer fra MECA. Bestilling av varer skulle utføres via API, men siden ingen leverandører har implementert dette ble det utviklet en skreddersydd løsning som kobler MECA sin nettbutikk sammen med CarAdmin. Dette var et lærerikt prosjekt som endte med en fungerende utvidelse av programvaren.

Summary of Graduate Project

Title:	**Lagermodul**
Date:	20.05.2019
Authors:	Bjørn Inge Kulsveen Andreas Maurud Prøven Marius Dahlen Østli
Supervisor:	Frode Haug
Employer:	Electric Time Car AS
Contact Person:	Dag L. Solhaug
Keywords:	AJAX, Java, JavaScript, JSP, CarAdmin
Pages:	85
Attachments:	11
Availability:	Open

Abstract: In large companies it's common to use cars or other vehicles owned by the company in conjuction with work assignments. Electric Time Car AS owns and develops a product named CarAdmin, which is used to administrate vehicles. The companies who uses this service often have internal workshops and storages for parts and tires. To accomodate for this Electric Time Car AS wished to extend their product to support a storage unit. This was developed using AJAX, HTML, Java, JavaScript and JSP. It contains an inventory for parts and tires, a register of storages and the possibility to order parts from MECA which is a third party provider. It was intended to implement ordering through the usage of API, however none of the third party providers contacted had developed an API for ordering. Due to this it was implemented as a customized- and unique solution which connects MECA's webshop to CarAdmin. This was an educational project which ended in a functioning extension of the software.

Forord

Bacheloroppgaven *Lagermodul* ble gjennomført av Bjørn Inge Kulsveen, Andreas Maurud Prøven og Marius Dahlen Østli på dataingeniørstudiet ved NTNU i Gjøvik. Det har vært en lærerik opplevelse og tilegnede kunnskaper fra tidligere i studiet ble satt på prøve. Oppgaven har vært utfordrende og spennende.

Vi ønsker å rette en stor takk til vår oppdragsgiver *Electric Time Car AS* ved Dag L. Solhaug og Øyvind Flatval. Vi er takknemlige for tilliten dere har gitt oss. Dag har vært en engasjert oppdragsgiver og han har bidratt med gode tilbakemeldinger og vært tilgjengelig for spørsmål. Øyvind var tilgjengelig for tekniske spørsmål gjennom hele prosjektet og var til god hjelp for oss.

Vi vil takke vår veileder Frode Haug for strålende veiledning. Frode bidro med gode råd og tilbakemeldinger gjennom hele bacheloroppgaven.

Vi vil også takke vår kontaktperson i *MECA*, Andreas Grimstad for samarbeidet. Han hjalp oss med opprettelse av en demo-bruker til nettbutikken deres, slik at vi kunne teste gjennomføring av bestillinger.

Til slutt vil vi takke venner og familie som har hjulpet oss med korrekturlesing.

Prosjektet har vært en spennende avslutning på studiet og en fin introduksjon til arbeidslivet. Vi har lært mye nytt som vil være nyttig å ta med seg videre.

Innhold

Figurer

Tabeller

Definisjoner

AJAX er en forkortelse for *Asynchronous JavaScript and XML* og brukes for å oppdatere web-sider dynamisk.

CarAdmin[1] er det eksisterende systemet for bilparkadministrasjon utviklet av ETC. Det er dette systemet som utvides med en lagermodul.

ETC[2] er forkortelse for *Electric Time Car AS*, som er oppdragsiver.

Eksternt lager (fjernlager) er et lager på en annen lokasjon enn bedriften.

Factory[3] er et designmønster som handler om hvordan kreasjonen av objekter kan generaliseres. Dette gjøres ved å samle kreasjonen av lignende objekter bak et grensesnitt, og det resulterer i en ryddig og effektivisert kodestruktur.

Internt lager er et lager på samme lokasjon som bedriften.

JavaDoc er standarden for kodedokumentasjon i Java.

JSP[4] er en forkortelse for *Java Server Page*. Disse filene kan inneholde JavaScript, HTML og Java-kode.

Kjøretøypark er et ord som brukes for å omtale alle kjøretøy en bedrift eier.

MECA Scandinavia AB[5] er en tredjepartsleverandør av bildeler. I lagermodulen kan disse bestilles ifra MECA ved behov for påfyll av lagerbeholdningen.

Menyelement er en side/fane i CarAdmin, som blant annet inneholder rapporter.

Produkt er det ETC tilbyr kundene i form av en tjeneste/funksjonalitet. Disse kan skreddersys ut ifra kundens behov eller ønske.

Rapporter er en måte for å fremvise data. Innholdet vises i form av tabeller.

Trello[6] er et verktøy for konfigurasjonsstyring. Dette brukes for å holde oversikt og dokumentere arbeidsprosessen i prosjektet. Oppgaver kan kategoriseres og tildeles gruppemedlemmer. På denne måten vil det til enhver tid være mulig å holde oversikt hva som gjøres, og hvem som gjør hva. Prosjektet er delt inn i fasene *Gjøremål*, *Under arbeid*, *Testing - Alfa*, *Testing - Beta* og *Ferdig*. Oppgavene er delt inn i kategoriene *Rapport*, *Kode* og *Bugs*. Oppgavene kan flyttes i mellom fasene ettersom de blir gjort eller omprioritert.

1 Innledning

I store bedrifter er det vanlig å ha firmabiler eller andre kjøretøy, disse benyttes av ansatte i forbindelse med arbeidsoppdrag. Alle kjøretøy i en bedrift omtales heretter som en kjøretøypark. En god oversikt over kjøretøyparken er viktig slik at kjøretøyene får nødvendig vedlikehold og service. Den som er ansvarlig for en kjøretøypark må kunne holde oversikt over hvor alle kjøretøy befinner seg til enhver tid, og status på disse. Hvilke personer som har benyttet seg av de er også viktig hvis det skulle oppstå ulykker eller situasjoner der noen må påta seg ansvar for en hendelse.

Electric Time Car AS[2] heretter kalt ETC, er en leverandør av software-løsninger for kjøretøyparker, ressursstyring og skreddersydde utviklingsprosjekter for kunder. De holder til i Gjøvik. ETC sitt hovedprodukt er CarAdmin[1], som er et system for administrasjon av kjøretøyparker. Utviklingen av dette produktet startet i 2004 og har ca. 40.000 registrerte brukere.

1.1 Omfang

1.1.1 Fagområde

CarAdmin har støtte for mange kjøretøytyper. Her er det mulig å se oversikt over alle kjøretøy i en kjøretøypark og hvem som bruker de. Elektronisk kjørebok er en oversikt over registrerte turer og hva formålet med turene var. Flåteoversikten viser et kart over hvor alle kjøretøyene befinner seg, samt at et rodekart viser hvor disse har beveget seg. Verkstedboka viser all oversikt over vedlikehold og skadehistorikk. Videre finnes det ordreoversikt, dokumentarkiv og rapporter for utnyttelsesgrad og månedsoversikt.

Denne funksjonaliteten er delt inn i fire hovedprodukter:

- **Kjøretøyregister** gir oversikt over alle kjøretøy på en lokasjon. Her finnes informasjon angående kjøretøyet, som for eksempel kilometerstand og tid for neste service.
- **Elektronisk Kjørebok** er der kjøreturer loggføres ved hjelp av GPS. Dette gir god oversikt over hvor kjøretøyene befinner seg til enhver tid.
- **Fellesbil** er utviklet for situasjoner der flere sjåfører deler et kjøretøy. Dette gir mulighet til å låse nøkler i et nøkkelskap i forbindelse med uttak og innlevering av kjøretøyene.
- **Flåte** gir oversikt over hvilke kjøretøy som befinner seg i nærheten, eller nærme et målpunkt. Det gis mulighet for å se kjørerute i kart. Dette brukes i forbindelse med ulike arbeidsoppdrag, og når en oppdragsgiver trenger å se hvem som kan ta på seg oppdraget.

1.1.2 Avgrensning

Mange av bedriftene som benytter seg av CarAdmin har interne verksteder, delelagre eller dekkhotell. Derfor ønsket ETC å utvide dette systemet med en lagermodul. Dette for å gi oversikt over hva som er inne på lager og loggføre hva det blir brukt til. Lagermodulen skal gi oversikt over interne lagre, og bestillinger som er på vei inn. Det skal også være mulig å sjekke beholdningen til tredjeparts lagre. All denne informasjonen er tilgjengelig via diverse rapporter der kunden kan sortere og filtrere etter eget ønske.

1.1.3 Oppgavebeskrivelse

Lagermodulen skal inneholde følgende funksjonaliteter:

- **Lagerregister** skal inneholde informasjon om de lagrene kunden har (interne/eksterne), dette kan være fra en til mange for en kunde.

- **Lagerbeholdning** med oversikt over alle deler som finnes på alle kundenes lagre. Denne skal inneholde antall av en del og lagerplassering (reol, hylle og nummer). Informasjonen skal være tilgjenglig via en rapport og kan skrives ut til bruk ved varetelling som en telleliste. Hver del har sin egen QR kode som kan skrives ut og plasseres sammen med delen på lager.

- **Lagerimport** Når nye deler skal legges inn på lager skal det enten gjøres manuelt ved å oppgi all data for produktet, lese data fra fil eller hente data fra en tredjepartsleverandørs API.

- **Rettigheter** CarAdmin har allerede tilgangstyring som gir kunder tilgang til tjenester de har betalt for, men sørger også for at kunden kan fordele rettigheter til sine ansatte. Eksempler på rettighetsnivåer er *Bruker*, *Bestiller* og *Administrator*. Dette skal også inngå i lagermodulen og eventuelle nye rettighetsgrupper må opprettes.

- **Ordre** Ved uttak av deler fra lager skal det opprettes en ordre slik at bedriften kan utføre intern fakturering og holde lagerbeholdningen oppdatert. Dersom delen ikke finnes på lager skal det opprettes en bestilling som blir knyttet opp mot en ordre. Ordren skal knyttes opp mot kjøretøyet den skal brukes på.

  Det ble etterhvert bestemt av ETC at dette punktet var for omfattende. Skulle dette blitt realisert måtte ordre i forbindelse med lagermodulen integreres i et eksisterende ordresystem. Dette mente ETC ville bli for tidkrevende å gjennomføre.

- **Bestilling** Det skal være mulig å bestille deler inn til lager fra tredjeparts leverandører, enten via en intern ordre eller enkeltvis fra lager for å fylle på lagerbeholdningen. Her skal det opprettes en rekvisisjon ut til leverandør via mail eller API. Alle bestillinger listes opp i en rapport som sier status på bestillingen (opprettet, godkjent, bestilt, mottatt). Disse må godkjennes av

en bruker med høye nok rettigheter, men dette kan skje automatisk dersom brukeren som bestiller allerede har det.

- **Retur** Det skal være mulig å returnere en ordre/bestilling.
- **Dekkhotell** skal holde oversikt over dekk som er på lager og hvor de befinner seg. Vinterdekk må defineres som enten piggdekk eller piggfritt. Ved innlegging av dekk eller ved dekkskifte skal det registreres dekkslitasje. Det skal noteres dimensjoner på dekkene. Det skal være mulig å koble opp dekkene til et kjøretøy.

Det er enkelte deler av oppgavebeskrivelsen som har endret seg underveis. Dette skyldes hovedsaklig at ingen tredjepartsleverandører hadde tilgjengelig API-er. Mer om dette kan leses i kapittel 8.1.1.

1.2 Formål

CarAdmin har i dag som nevnt tidligere, fire hovedprodukter som er kjøretøyregister, kjørebok, fellesbil og flåte. For å gjøre CarAdmin mer komplett ønsket ETC å implementere en lagermodul. Målgruppen til ETC er som tidligere nevnt store og offentlige bedrifter som har egne kjøretøyparker. Noen av disse har egne verksteder der de utfører vedlikehold på sine kjøretøy. Hensikten med lagermodulen er at CarAdmin skal bli mer attraktivt for disse. Lagermodulen skal inneholde en hovedside der det kan søkes på deler, dekk og lagre. Det skal også være en side for lagerbeholdning, dekkbeholdning, lagerregister og lagerbevegelse. Disse funksjonalitetene vil gjøre det enkelt for brukere av systemet å holde oversikt over hva de har på lager. Dette vil være med på å bygge videre på ETC sin visjon: ≪Det Gode Bilhold!≫.

1.3 Prosjektmål

I starten av prosjektperioden ble prosjektmål definert. Enkelte av disse ble ikke oppnådd ettersom ingen tredjepartsleverandør hadde API for bestilling av varer.

Dette gjorde oppgaven annerledes enn tenkt. Ordreoversikten lot seg heller ikke gjøre da ETC hadde et eget system for dette som ikke var tilgjengelig i utviklingsmiljøet. ETC mente også at integrasjon med dette ordresystemet ville bli for mye for gruppen å utvikle. Dermed ble ordreoversikten kuttet ut. Mer om dette kan leses i kapittel 8.1.2.
Følgende resultatmål, effektmål og læringsmål skulle i utgangspunktet oppnås:

1.3.1 Resultatmål

Lagermodulen skal tilføye CarAdmin funksjonaliteter som gir oversikt over lagerbeholdingen for en bedrifts interne og eksterne lagre. Lagerbeholdningen skal oppdateres ettersom deler blir brukt på kjøretøy eller når nye deler kommer inn på lager. Dette skal være tilgjengelig i form av en rapport som også brukes som telleliste ved varetelling. Oversikten over alle bestillinger skal være oppdatert og tilgjengelig via en rapport (ordreoversikt). Dersom det interne lageret mangler en del kan denne bestilles fra tredjeparts leverandører, som for eksempel MECA[5]. Her skal det også hentes ut data og egenskaper for de forskjellige delene som er på lager gjennom et API.

1.3.2 Effektmål

Ved gjennomføring av dette prosjektet forventer ETC følgende effekt:

- Komplettere dagens løsning for kjøretøyhold for offentlige- og store bedrifter.
- Tilgang til nye kunder.

Med mer funksjonalitet vil CarAdmin tilfredsstille flere ønsker og bli aktuell for flere kunder.

1.3.3 Læringsmål

Ved gjennomføring av bacheloroppgaven ønsker gruppen å få erfaring med planlegging og utvikling av en større applikasjon. CarAdmin er en webløsning, og gjennom arbeidet med denne er målet at det skal opparbeides kunnskap om hvordan en webløsning fungerer. Her tilegnes erfaring med Apache Tomcat[7] og AJAX (≪Asynchronous JavaScript And XML≫). CarAdmin bruker MariaDB[8] som databaseløsning og det vil gjennom jobbing med dette, opparbeides mer erfaring innenfor databaser. Serverkoden skal skrives i Java og det medfører at det tilegnes mer erfaring med språket. I tillegg ønskes det et tett samarbeid mellom gruppemedlemmene for å sikre deling av kunnskap, og flyt i arbeidet. Med alt dette vil ferdighetene innen programmering generelt forbedres.

1.4 Målgruppe

1.4.1 Målgruppe for lagermodulen

Målgruppen for lagermodulen er brukere av CarAdmin som jobber i bedrifter med egne lagre, enten interne eller eksterne. Disse kan være både delelagre og dekkhotell.

1.4.2 Målgruppe for rapporten

Rapporten er ment for personer med kunnskap innen IT og systemutvikling. Den er også relevant for personer med erfaring innen administrering av kjøretøyparker. Av forkunnskaper er det en forutsetning at leseren har kompetanse tilsvarende 2,5 år på dataingeniørstudiet ved NTNU i Gjøvik.

1.5 Gruppens bakgrunn og kompetanse

Alle på gruppen har samme faglige bakgrunn gjennom dataingeniørstudiet til NTNU i Gjøvik. Medlemmene har opparbeidet seg relevant kunnskap innen fagene *Objekt-Orientert Programmering*, *Datamodellering og Databasesystemer* og

Systemutvikling. Høstsemesteret 2018 besto av valgfag og alle var gjennom *Programvaredesign* og *Applikasjonsutvikling*. Andreas hadde som tredje fag *Cloud Technologies*, mens Bjørn Inge og Marius hadde *Matematikk 3*. Alle har forskjellige spisskompetanser og det dannet et bra grunnlag for bacheloroppgaven.

1.5.1 Hva måtte læres?

For å kunne gjennomføre bacheloroppgaven måtte gruppen tilegne seg nye kunnskaper. Grunnlaget fra dataingeniørstudiet var ikke nok. Nye teknologier måtte læres. Ingen hadde erfaring med bruk av AJAX, JavaScript eller JSP. Prosjektet er hovedsaklig kodet i Java, noe gruppa har erfaring med, men siden det har blitt utviklet siden 2004 har Java-språket blitt utnyttet mer enn hva gruppen lærte i Applikasjonsutvikling. Ingen gruppemedlemmer hadde erfaring med utviklingsplatformen IntelliJ IDEA[9], som medførte at gruppen måtte lære seg å bruke denne.

1.6 Rammer

1.6.1 Tekniske rammer

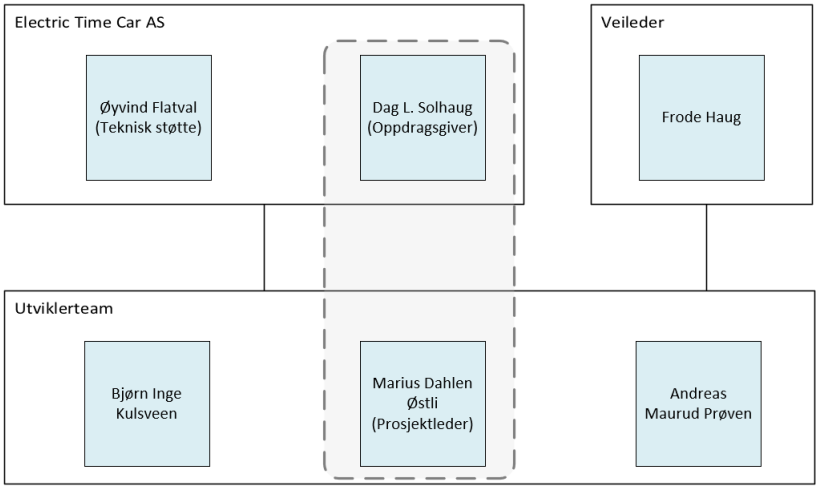
CarAdmin er skrevet i Java og kjører på Apache Tomcat hos ETC. Under utvikling av lagermodulen ble Tomcat kjørt lokalt på egne maskiner slik at endringene som ble utført, ikke påvirket den eksisterende løsningen. ETC benytter IntelliJ IDEA som utviklingsmiljø i CarAdmin-prosjektet. Gradle[10] er verktøyet som brukes for å bygge løsningen. IntelliJ IDEA har støtte for en plugin som heter Lombok[11]. Med denne trengs ikke egne «get- » og «set-metoder» å bli laget for klasser, Lombok gjør det automatisk. Lombok har blitt delvis benyttet i CarAdmin, men i koden for lagermodulen ble den benyttet i alle klasser som holder på data. I prosjektet er det også benyttet et *Factory-pattern*, som er omtalt i kapittel 5.1.6, for å effektivt kommunisere med databasen.

I tillegg til Java-kode ble det brukt JSP, som står for ≪Java Server Page≫. Det er en teknologi for server-side programmering, og brukes for å designe- og bygge platform-uavhengige-, web-baserte applikasjoner[4]. JSP-filer kan inneholde Java-kode for logiske operasjoner, men også JavaScript og HTML for design av web-sider. Kommunikasjonen mellom klientens interaksjoner og server skjer via AJAX-kall.

Ved utviklingen av lagermodulen ble disse teknologiene benyttet på samme måte som ellers i prosjektet. Grunnen til dette var å gjøre det enklere for ETC å jobbe videre med den.
I koden ble kommentarer og *JavaDoc* (dokumentasjon) skrevet på norsk, mens selve koden er logiske variabelnavn skrevet på engelsk. Variabler og funksjoner er skrevet som ≪Camel-case≫, dvs. at et variabelnavn bestående av flere ord, starter med en liten bokstav, mens neste ord starter med en stor bokstav.

1.7 Prosjektorganisering

Prosjektet var et samarbeid mellom ETC og utviklerteamet. Veileder bisto utviklerteamet med råd og tilbakemelding på rapporten, men ikke noe teknisk i forbindelse med utviklingen av lagermodulen. Figur 1 illustrerer dette.



Figur 1: Organisasjonskart

Prosjektleder: Marius Dahlen Østli ble utnevnt til prosjektleder av gruppen. Han hadde hovedansvaret for å kommunisere med ETC, avtale møter, koordinere arbeidstider og skrive møtereferater.

Veileder: Frode Haug var veileder gjennom prosjektet og kom med råd og tilbakemeldinger underveis. Disse tilbakemeldingene var hovedsaklig på rapporten, men også rundt arbeidsprosessen.

Oppdragsgiver: Dag L. Solhaug er daglig leder i ETC og var oppdragsgiver. Når det oppsto spørsmål angående oppgavens eksakte innhold, ble disse tatt opp med han.

Teknisk støtte: Øyvind Flatval er ansatt i ETC og hadde ansvaret for å gjennomføre opplæringen i CarAdmin-prosjektet, slik at gruppa kunne komme i gang med utviklingen. Tekniske spørsmål ble rettet til han. Øyvind var også tilstede under statusmøter for bistand ved tekniske problemstillinger.

Utviklerteam: Utviklerteamet som realiserte lagermodulen besto av Bjørn Inge Kulsveen, Andreas Maurud Prøven og Marius Dahlen Østli.

1.8 Plan for gjennomføring

Prosjektet ble delt inn i tre hoveddeler med planlegging, utvikling og rapportskriving. Gruppen ble enige om å anvende en plandreven systemutviklingsmodell i prosjektet ettersom oppgaven appellerte til det. Den valgte modellen ble *Inkrementell sekvensiell*[12]. Grunnlaget for valget kan leses i kapittel 2.1.1.

Planen for gjennomføring av prosjektet er illustrert med et Gantt-diagram i vedlegg F.
Planleggingsperioden varte fra 10. Januar til 1. Mars. Det ble satt av mye tid til planlegging siden det er viktig når det jobbes etter en plandreven modell. Et forprosjekt ble levert før 1. Februar. Deretter ble det satt av en uke for å studere koden i CarAdmin-prosjektet. Tiden etter dette og fram til 1. Mars skulle brukes til å skrive starten på rapporten med innledning, kravspesifikasjon, arbeidsmetoder og arkitektur. Parallellt med dette kontaktet gruppen tredjepartsleverandører av bildeler med spørsmål om tilgang til API-er for bestilling.

Utviklingsperioden ble delt inn i tre inkrementer. Hvert inkrement besto av tre hovedoppgaver slik at alle på gruppen kunne jobbe selvstendig. Dette sikret god fremdrift. Det første inkrementet besto av å sette opp grunnstrukturen til lagermodulen. Denne skulle inneholde et lagerregister som viser oversikt over alle lagre en bedrift har, en lagerbeholdning som sier hvilke deler som er på lager, og funksjonalitet for å legge til deler. Det andre inkrementet besto av oppkobling mot

tredjeparts API, håndtering av bestilling og ordre. På slutten av dette inkrementet satte gruppen en frist for å få på plass oppkobling til API, da dette var et viktig punkt i oppgaven. I løpet av det tredje og siste inkrementet skulle det gjøres mulig å reservere deler, opprette en side som holder oversikt over lagerbevegelser og lagermodulen skulle internasjonaliseres.

Det ble tatt forbehold om at endringer kunne oppstå slik at noen arbeidsoppgaver kunne flyttes. Rapportskrivingen kunne foregå parallelt med utviklingen for å sørge for dokumentasjon av arbeidet underveis. Etter leveringsfristen for rapporten ble det satt av tid på å forberede seg til en muntlig presentasjon.

1.9 Rapportstruktur

1.9.1 Terminologi

Rapporten er skrevet i LaTeX og Overleaf[13] er teksteditoren som ble brukt. Overleaf er et verktøy som støtter LaTeX og det tillater at flere kan redigere et dokument samtidig. Ord og uttrykk som krever forklaring, er listet opp i **Definisjoner**. Noen faguttrykk er skrevet på engelsk i rapporten. Dette ble gjort fordi ordene brukes slik innenfor fagområdet. Ved referering til kilder benyttes *Vancouver*-metoden. Dette vil kunne sees som et tall i mellom to klammeparenteser. Nederst i rapporten finnes en liste over benyttede kilder. Ved referering til figurer og kapitler, vil det også bli benyttet tall, men disse er ikke innkapslet i noe. Ved referering til vedlegg vil en stor bokstav benyttes. Referanser vil plasseres i teksten hvor det er naturlig. Ved klikk hoppes det til visning av den valgte.

1.9.2 Figurer og tabeller

Etter innholdsfortegnelsen kommer to sider der den ene inneholder en liste over figurer, og den andre en liste over tabeller brukt i rapporten.

1.9.3 Inndeling av kapitler

Innledning er det første kapittelet som beskriver hensikten med oppgaven og omfanget av den. Her beskrives også hvordan prosjektet skal gjennomføres med tanke på organisering av arbeidet.

Arbeidsmetodikk er det andre kapittelet. Her finnes informasjon om den valgte systemutviklingsmodellen og hvordan denne anvendes.

Kravspesifikasjon er det tredje kapittelet. Her står detaljert informasjon om funksjonaliteten lagermodulen skal inneholde. Fra oppdragsgivers krav er det utarbeidet diagrammer som illustrerer ulike aspekter i prosjektet. I tillegg til dette finnes informasjon om sikkerhets- og operasjonelle krav her.

Design er det fjerde kapittelet og utseende til lagermodulen blir beskrevet og illustrert her.

Arkitektur er det femte kapittelet. I dette kapittelet finnes informasjon om eksisterende systemarkitektur og hvor lagermodulen skal integreres.

Implementasjon er det sjette kapittelet som inneholder informasjon om hvordan lagermodulen er implementert. Her beskrives også tekniske aspekter under gjennomføring av utviklingen.

Testing og kvalitetssikring er det syvende kapittelet der det tas opp hvordan funksjonaliteter i lagermodulen ble testet og kvalitetssikret.

Avslutning er det siste kapittelet der gruppen evaluerer hele prosjektperioden med tanke på mulighet for å sette lagermodulen ut i drift, kritikk av oppgaven og potensiale for videre utvikling.

2 Arbeidsmetodikk

Dette kapitlet handler om arbeisprosessen gjennom bacheloroppgaven. Valg av systemutviklingsmodell og anvendelsen av denne inngår her. I tillegg nevnes andre faktorer som påvirket arbeidsprosessen.

2.1 Hovedinndeling av prosjektet

2.1.1 Valg av systemutviklingsmodell

Lagermodulen skulle utvikles i et eksisterende system. Det medførte at det var en del rammer å forholde seg til. Lagermodulen måtte lages slik at den kunne passe inn i CarAdmin. Dette ble gjort ved å bruke samme kodespråk og kodestandarder som ellers i programvaren. Oppgaven var klar på hva slags funksjonalitet som skulle lages, men ikke så klar at det kunne forutses nøyaktig hvor lang tid som trengtes for å utvikle de ulike delene. Som følge av dette ble utviklingsmodellen inkrementell-sekvensiell[12] valgt. Det ble slik fordi det var enkelt å definere inkrementene ut ifra klare krav. Under utviklingsprosessen ble et inkrement ferdigstilt før det neste ble påbegynt. Fordelen med dette var at gruppen hadde noe å levere til slutt hvis uventede problemer med utviklingen skulle oppstå underveis.

Streng følging av fossefall ble vurdert, men det viste seg ikke å være det beste valget. Dette fordi fossefall ikke er åpen for endringer i kravspesifikasjonen underveis. Et usikkert punkt i oppgaven var importeringen av tredjeparts delekataloger, som krevde tilgang til API-er, og det var ikke garantert at det lot seg gjøre. Det måtte derfor finnes en mulighet til å kunne gjøre små justeringer hvis nødvendig, og det var mest kostnadseffektivt i den valgte modellen. Hvis oppgaven hadde vært mer åpen med en uforutsigbar kravspesifikasjon, ville det vært fornuftig med smidig modell i stedet. Det var den ikke i dette tilfelle, derfor endte valget på en plandreven modell.

2.1.2 Anvendelse av systemutviklingsmodell

Utviklingsmodellen Inkrementell-Sekvenisell kan følges både plandrevet og smidig. Forskjellen er hvordan inkrementene defineres. I en plandreven versjon blir alle inkrementene definert på forhånd, mens i en smidig versjon blir kun enkelte inkrementer definert i starten av et prosjekt. I den smidige versjonen vil inkrementer som kommer senere i prosjektperioden bli definert ut ifra progresjonen i utviklingen og hva kunden prioriterer. I dette prosjektet ble den plandrevne modellen fulgt da alt som skulle implementeres sto oppført i oppgaveteksten.
Fordeler med modellen[12]:

- Endringer i kravspesifikasjonen er enklere å håndtere enn i fossefall.
- Oppdragsgiver kan gi tilbakemeldinger underveis ettersom inkrementene utvikles. Utviklerteamet kan demonstrere funksjonalitet i stedet for å bare vise skisser som ofte gjøres i fossefall. Det gjør tilbakemeldingene bedre og utviklerteamet kan tilpasse planen videre ut ifra kundens ønsker.
- Funksjonalitet kan leveres tidlig i prosessen. Skulle ikke utviklerteamet bli ferdig med alt, vil det fortsatt være mulig å levere det som fungerer ettersom inkrementene i prinsippet er uavhengig av hverandre.

Modellen[12] har også ulemper:

- Prosessen er ikke synlig og det er vanskelig å dokumentere denne. Fremgangen kan kun måles ved jevnlige leveranser av funksjonaliteter.
- Bruk av denne modellen kan føre til ustrukturert kode. Hvis kunden kommer med nye ønsker underveis vil utviklingsteamet ofte gjøre det de kan for at kunden skal bli fornøyd. Det kan føre til at funksjonalitet blir implementert på en forhastet og dårlig måte. Dette kan på lang sikt gjøre det vanskelig å videreutvikle systemet.

Til tross for ulempene ble gruppen enige om at denne modellen er den beste å følge for dette prosjektet. Statusmøtene på fredager ble brukt til å demonstrere

fremgangen i utviklingen for oppdragsgiver. Her ga de tilbakemeldinger på hva de syntes og hva de kunne tenke seg av justeringer.

2.1.3 Inkrementene

Ut ifra kravene på hvilken funksjonalitet ETC ville ha, ble det bestemt at utviklingen skulle deles inn i tre inkrementer. For å gjøre utviklingsprosessen effektiv, ble inkrementene definert slik at alle på gruppen kunne jobbe med forskjellige oppgaver. Et inkrement besto av tre oppgaver. Det var fortsatt fokus på samarbeid, men det ga raskere progresjon i utviklingen. De to første inkrementene skulle jobbes med over to perioder på to og ei halv uke. Det ble satt av tre uker til det siste. Dette ville gi tid til å finpusse programvaren og eventuelt legge til ekstra funksjoner.

I det første inkrementet skulle hovedfunksjonaliteten på plass. Dette innebar lagerregister som inneholder alle kundens lager, lagerbeholdning på disse lagrene og import av delekatalog for å kunne legge deler inn på lager.
Det andre inkrementet skulle brukes til å ta for seg endringer i lagerbeholdningen. Dette gjaldt ordre, bestillinger og oppkobling til tredjeparts-leverandørers API slik at bestillinger skal kunne utføres. Her ble det også lagt inn en selvbestemt frist for oppkobling mot API siden dette var kristisk for videre utvikling.
I det tredje og siste inkrementet skulle det bli mulig å reservere deler etter innkjøp. Det vil si at et verkstedet som trengte en del og den ikke fantes på lager, hadde førsterett når delen ankom lageret. Det skulle også gjøres mulig å følge lagerbevegelser. Til slutt skulle hele modulen internasjonaliseres slik at den kunne støtte både norsk og engelsk.

2.2 Dokumentasjon av arbeidsprosess

For å kontrollere at utviklingsmodellen ble fulgt, ble konfigurasjonsstyrings-verktøyet *Trello*[6] benyttet. Det var enkelt å holde god oversikt over hva som skulle gjøres, hva som var under arbeid, hva som var klart til testing og hva som var ferdigstilt.

Prosjektet ble delt inn i tre fargekodede kategorier i Trello. Disse kategoriene var *rapport*, *kode* og *bugs*. Alle oppgaver som var relatert til rapporten ble markert med rapportkategorien. Alt som skulle programmeres ble markert med kodekategorien og bugs-kategorien var uforutsette problemer som oppsto.

I tillegg til bruken av Trello, mottok veileder Frode Haug tre statusrapporter underveis i prosjektperioden. Disse rapportene inneholdt hvor langt gruppen hadde kommet i forhold til prosjektplanen, hva som var under arbeid, hva som gjensto og hva som var ferdigstilt. Statusrapportene kan leses i vedlegg E.

2.3 Statusmøter

2.3.1 Møter med ETC

Gruppen prioriterte å ha jevnlige statusmøter med oppdragsgiver for å få tilbakemelding på arbeidet. Møtene ble avholdt hver fredag kl. 08:00 i ETC sine lokaler. ETC bisto på disse møtene ved å hjelpe til med tekniske utfordringer, samt idéer til videre framdrift. Ønsket om disse møtene var noe av grunnen til at fossefallmodellen ikke ble valgt. Gruppen ønsket fortløpende tilbakemeldinger for å kunne forbedre eller endre på funksjonalitet underveis. Dette bidro til en bedre lagermodul. Møtereferater kan leses i vedlegg H.

2.3.2 Møter med veileder

Statusmøter med veileder Frode Haug ble avholdt hver mandag kl. 09:00 på NTNU i Gjøvik. Frode kom med gode råd når det gjaldt arbeidsprosess og rapportskriving. Det ble kun notert møtereferater da gruppen ønsket tilbakemeldinger rundt dette. Ved møter under utviklingsperioden (Mars/April) ble det ikke notert så mye. Disse møtene ble stort sett brukt til å holde Frode oppdatert på progresjonen i prosjektet og vise frem det vi hadde utviklet. Møtereferater kan leses i vedlegg I. Når det gjaldt spørsmål om utviklingen var det ETC som var kontaktpunktet.

3 Kravspesifikasjon

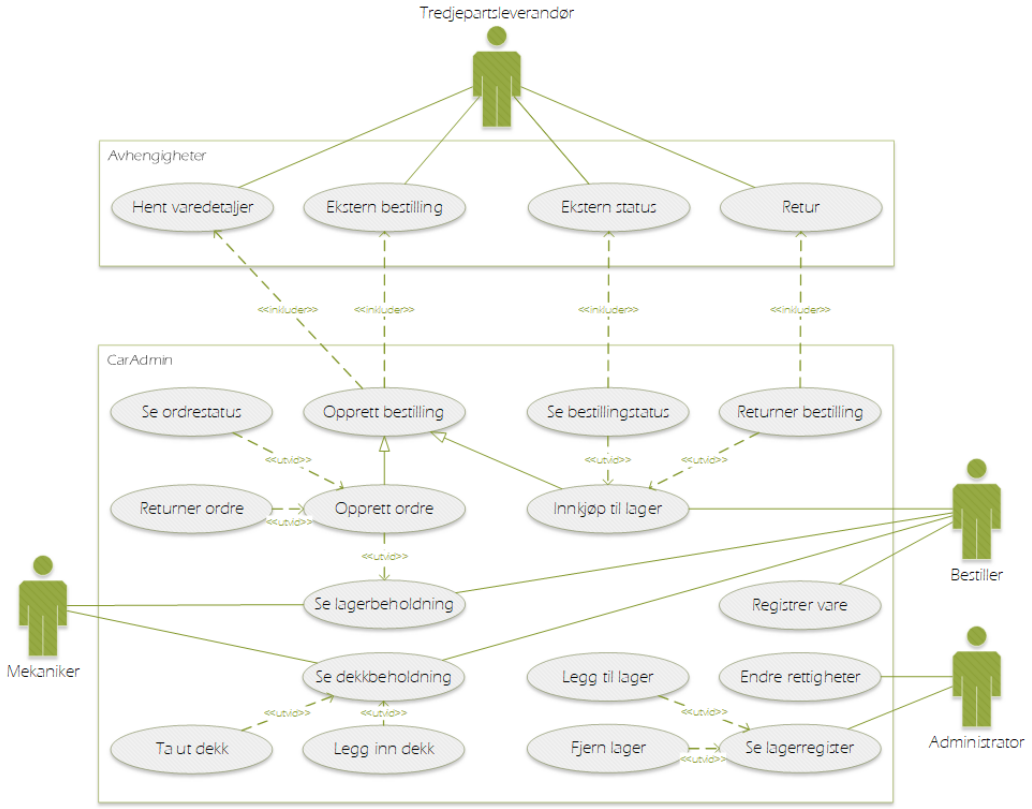
Dette kapitlet handler om hva som skulle utvikles og hvilke hensyn som måtte tas. Et Use Case diagram beskriver ulike funksjonaliteter og hvilke brukere/aktører som skulle ha tilgang til de. En detaljert Use Case beskrivelse ble laget for den mest kompliserte funksjonaliteten. I tillegg til hva som skulle utvikles er det beskrevet hvilke operasjonelle krav og sikkerhetskrav lagermodulen måtte oppfylle, samt en utført risikoanalyse.

3.1 Brukergrupper

Mekaniker: Jobber på verkstedet og kan ta deler ut fra lager.
Bestiller: Kan utføre og godkjenne bestillinger til tredjepartsleverandører. Bestiller kan også gjøre alt en mekaniker kan gjøre.
Administrator: Kundens systemansvarlig som kan legge til lager og kjøretøy.
Løsningsadministrator: ETC med alle rettigheter.

3.2 Use Case

Figur 2 viser et komplett Use Case diagram for hvilke funksjonaliteter som skulle implementeres.



Figur 2: Use Case for lagermodul

3.2.1 Høynivå Use Case

Navn: Se dekkbeholdning
Primær aktør: Mekaniker, Bestiller
Hensikt: Ha oversikt over hvilke dekk som er på lager
Beskrivelse: Kunne klikke seg inn på dekkbeholdning hvor man får oversikt over alle dekk som er på lager.

Navn: Legg inn dekk
Primær aktør: Mekaniker, Bestiller
Hensikt: Legge inn nye dekk.
Beskrivelse: Brukeren klikker på «Legg til nytt dekk», og blir bedt om å fylle inn data om dekket.

Navn: Ta ut dekk
Primær aktør: Mekaniker, Bestiller
Hensikt: Ta ut et dekk.
Beskrivelse: Brukeren klikker på ett dekk og velger «Ta ut dekk». Dekket fjernes fra lagerbeholdning (saldo trekkes fra med 1).

Navn: Se lagerbeholdning
Primær aktør: Mekaniker, Bestiller
Hensikt: Oversikt over lagerbeholdning på alle lagre til kunden.
Beskrivelse: Gir en oversikt over alle varer som er på kundens lagre. Her skal det kunne søkes etter varer og filtreres ut ifra diverse parametre. Hver vare ligger inne i systemet med data som sier hvilke egenskaper den har, og antall som ligger på lager.

Navn: Opprett ordre
Primær aktør: Mekaniker
Hensikt: Uttak fra lagerbeholdning.
Beskrivelse: Når en vare tas ut fra lager skal det opprettes en ordre internt i systemet. Denne skal inneholde informasjon om hvilken vare det gjelder, hva den brukes til og hvem som tok den ut. Dette vil gi en oversikt over hva varene blir brukt til, som også kan brukes for å utføre intern fakturering ut ifra hvem som registrerte ordren.

Navn: Se ordrestatus
Primær aktør: Mekaniker
Hensikt: Se status på en ordre
Beskrivelse: Kunne gå inn på en ordre og se status på denne. Status vil være mottatt/ikke mottatt.

Navn: Returner ordre
Primær aktør: Mekaniker
Hensikt: Gjøre det enkelt å returnere varer tilbake til internt lager.
Beskrivelse: Dersom det trengs å foreta en retur på en vare til eget lager, skal det være mulig. Dette tilfellet er når en vare er tatt ut ifra lager, men det er ønskelig å returnere den. Det skal være mulig å klikke på retur inne på en ordre.

Navn: Returner bestilling
Primær aktør: Bestiller
Hensikt: Gjøre det enkelt å returnere varer som er sendt fra en tredjepartsleverandør.
Beskrivelse: Dette tilfellet er når en vare er blitt bestilt, men det er ønskelig å returnere varen. Det skal være mulig å klikke på retur inne på en bestilling. Det skal da fylles ut et skjema og varen sendes tilbake. Denne er avhengig av at det eksterne systemet har støtte for retur.

Navn: Innkjøp til lager
Primær aktør: Bestiller
Hensikt: Det skal være mulig å bestille varer inn til lager fra eksterne leverandører.
Beskrivelse: En bruker skal kunne bestille varer fra en ekstern aktør når det er tomt på lager, eller hvis det trengs påfyll.

Navn: Opprett bestilling
Primær aktør: Mekaniker, Bestiller
Hensikt: Bestille nye deler fra tredjepartsleverandør.
Beskrivelse: Dersom bestiller har utført et ≪innkjøp til lager≫ eller mekaniker har opprettet en ordre og delen ikke finnes på lager, så skal det opprettes en bestilling i systemet. Dersom bestillingen ble opprettet av en mekaniker må denne godkjennes av en bestiller. Videre skal det opprettes en rekvisisjon som sendes til tredjepartsleverandør.

Navn: Hent bestillingsstatus
Primær aktør: Bestiller
Hensikt: Status på bestillingen hentes fra eksterne leverandører.
Beskrivelse: Det skal være mulig å sjekke status på bestillinger som ligger inne i systemet. Dette skal gi en statusmelding (opprettet, godkjent, bestilt, mottatt). Denne er avhengig av informasjon fra tredjepartsleverandør (Ekstern status).

Navn: Registrer vare
Primær aktør: Bestiller
Hensikt: Registrere nye varer i systemet.
Beskrivelse: Bestiller skal kunne legge til nye varer i systemet. Dette gjøres manuelt. Bestilleren trykker på ≪Registrer vare≫, og det kommer opp et nytt vindu hvor det fylles inn detaljer. Det skal også være mulig å fjerne varen.

Navn: Se lagerregister
Primær aktør: Administrator
Hensikt: Oversikt over de forskjellige lagrene kunden eier.
Beskrivelse: Skal gi en oversikt over alle lagre kunden har. Disse kan være plassert på forskjellige lokasjoner enten internt i kundens lokaler eller på et eksternt lager. Her kan kunden legge til nye lagre eller fjerne eksisterende ut fra hva situasjonen måtte være.

Navn: Legg til lager
Primær aktør: Administrator
Hensikt: Det skal være mulig å legge til nye lagre
Beskrivelse: Dersom kunden utvider med et nytt lager enten internt eller eksternt, skal det være mulig å legge dette til i systemet.

Navn: Fjern lager
Primær aktør: Administrator
Hensikt: Lagre skal kunne fjernes hvis de ikke skal brukes mer.
Beskrivelse: Lagermodulen skal ha funksjonalitet for å fjerne eksisterende lagre. Dette gjelder både interne og eksterne.

Navn: Endre rettigheter
Primær aktør: Administrator
Hensikt: Det skal være mulig for kunden å fordele rettigheter til sine ansatte.
Beskrivelse: Administrator huker av på ulike bokser som bestemmer hvilke rettigheter en ansatt har.

3.2.2 Eksterne avhengigheter

Navn: Hent varedetaljer
Primær aktør: Tredjepartsleverandør
Hensikt: Beskrivelse om deler fra eksterne leverandører hentes fra deres system.
Beskrivelse: Kunne se informasjon om alle varer. Noen av varene vil komme fra eksterne leverandører. Siden informasjon om disse allerede eksisterer skal disse hentes eksternt.

Navn: Ekstern bestilling
Primær aktør: Tredjepartsleverandør
Hensikt: For å kunne bestille er systemet avhengig av at det tas imot en bestilling hos de eksterne aktørene.
Beskrivelse: Bestillingen håndteres av tredjepartsleverandørers egne systemer.

Navn: Ekstern status
Primær aktør: Tredjepartsleverandør
Hensikt: Motta status på en bestilling fra tredjepartsleverandør.
Beskrivelse: Det skal mottas en status fra en tredjepartsleverandør for å kunne se bestillingsstatus inne i systemet.

Navn: Retur
Primær aktør: Tredjepartsleverandør
Hensikt: For å returnere en bestilt vare må det kommuniseres med et eksternt system.
Beskrivelse: Systemet samhandler med et eksternt system som mottar en retur.

3.3 Detaljert Use Case beskrivelse

Dersom alle Use Case-er skulle beskrives detaljert ville det blitt omfattende og langt. Det er derfor blitt valgt ut én som beskrives detaljert. Gruppen valgte å beskrive ≪Opprett bestilling≫ i detalj, siden det er en veldig essensiell funksjonalitet i lagermodulen. Denne er en generalisering av ≪Opprett ordre≫ og ≪Innkjøp til lager≫. Grunnen er at det skal opprettes en bestilling hvis det enten er tomt for deler eller ved generell påfylling av lager. Bestillingen skjer via en tredjepartsleverandør. Use Casen inkluderer ≪Hent varedetaljer≫ som viser detaljer om en

del hos tredjepartsleverandør. ≪Ekstern bestilling≫ inkluderes også, men dette er tredjepartsleverandør sin egen bestillingsprosess.

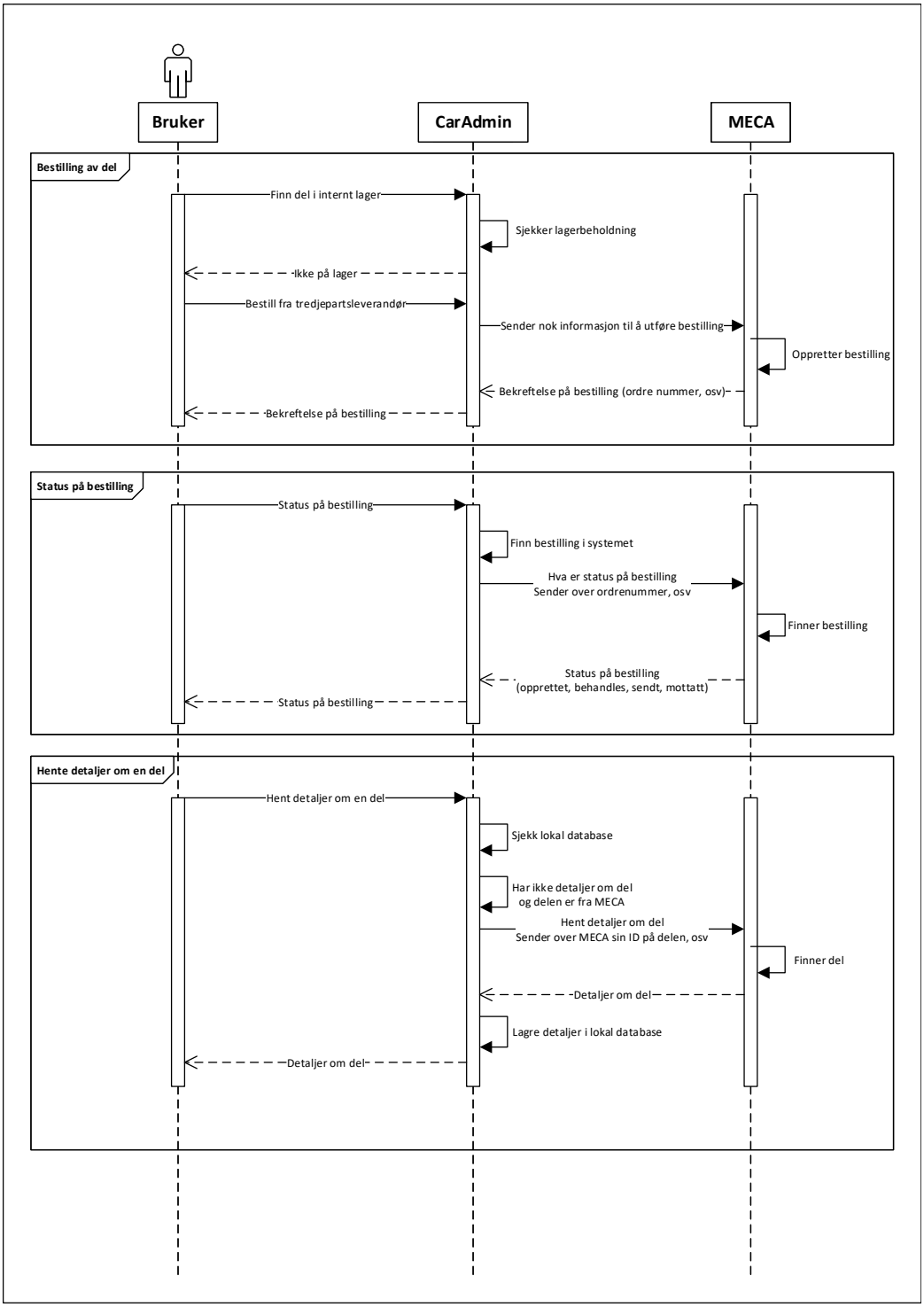
Opprett bestilling

- **Aktører:** Bestiller, mekaniker og tredjepartsleverandør
- **Hensikt:** Det skal være mulig å bestille varer inn til lager fra tredjepartsleverandør ved behov for påfyll av lagerbeholdningen.
- **Handlingsforløp:**
  1. Bestiller ønsker å fylle på med mer av en vare eller et lager mangler en vare.
  2. Det opprettes en bestilling i systemet som spesifiserer leverandør og type vare som skal bestilles.
  3. Rekvisisjon sendes til tredjepartsleverandør.
  4. Tredjepartslevereandør håndterer bestilling og sender tilbake et bestillingsnummer.
  5. Statusen for bestillingen oppdateres til ≪bestilt≫.
  6. Ved leveranse oppdateres lagerbeholdningen.
- **Alternativt handlingsforløp:**
  1. Mekaniker ønsker en vare og oppretter en ordre i systemet.
  2. Ønsket vare finnes ikke på lager.
  3. Bestilling opprettes.
  4. Bestillingen godkjennes av autorisert bruker (Bestiller).
  5. En rekvisisjon sendes til tredjepartsleverandør.
  6. Tredjepartsleverandør håndterer bestilling og sender ønsket vare.

7. Bestillingsstatus oppdateres til ≪bestilt≫.
8. Når varen ankommer lageret oppdateres statusen til ≪mottatt≫.

3.4 Sekvensdiagram for bestilling fra MECA

Det ble laget et forslag i form av et sekvensdiagram (se figur 3 på neste side) for hvordan bestilling via API-et til MECA kunne løses. Sekvensdiagrammet ble sendt til MECA. Gruppen fikk vite at MECA ikke hadde utviklet API for bestilling, men at de hadde planer om å gjøre dette i nær fremtid.



Figur 3: Sekvensdiagram for bestilling fra MECA

3.5 Domenemodell

For å illustrere sammenhengen i systemet, ble en domenemodell laget. Se vedlegg J. Denne viser sammenhengen mellom klasser som er involvert i forbindelse med *ServiceArtOverview*. Domenemodellen ble satt opp for å få en oversikt over hvordan løsningen er bygget opp, og den inneholder alle klasser som er involvert i forbindelse med produktregisteret, under menyelementet *Verkstedbok*. Kodestrukturen til lagermodulen skal bygge på prinsippene fra produktregisteret.

3.6 Operasjonelle krav

CarAdmin er i dag et system med mange brukere. Det var derfor viktig at lagermodulen ble integrert uten å endre eller fjerne eksisterende funksjonalitet. Dette ble sikret ved å bruke «branching» i Git[14] for å skille gruppens utviklingsmiljø fra ETC sitt, inntil lagermodulen ble ferdigstilt. Videre ble gruppens utviklingsmiljø delt inn i tre nye «branch-er». Hvert gruppemedlem brukte hver sin, og etterhvert som utviklingen foregikk ble fungerende kode lagt til i «hoved-branchen». Ved hjelp av dette ble det også lettere for gruppen å fordele oppgaver, og det oppsto færre problemer med «merging».
Funksjonaliteten til lagermodulen skulle ha minst like god responstid og oppetid som resten av systemet. For å oppnå dette ble det benyttet samme kodestandarder og vektlagt en god og smart utnyttelse av den eksisterende koden.
Lagermodulen måtte være brukervennlig. Det betydde et oversiktlig og intuitivt grensesnitt, slik at brukeren slipper å bruke mye tid på å sette seg inn i virkemåten. Dette ble gjort ved å designe utseende til lagermodulen på samme måte som andre deler av systemet.

3.7 Sikkerhetskrav

Sikkerhet var et viktig å punkt. Det viktigste er at rettigheter en bruker innehar, styrer hva brukeren skal ha tilgang til. For eksempel skal det ikke være mulig for

en mekaniker å legge til lagre eller fjerne et lager. Det måtte sikres at funksjonalitet for å gjøre dette, kun er tilgjengelig for riktige brukergrupper. Utvidelsen måtte ivareta sikkerheten til det eksisterende systemet. Input skal sikres/saneres, dette for å unngå for eksempel ≪buffer overflow≫. Her brukes det ulike saneringsfunksjoner. SQL-spørringer parametriseres ved å bruke ≪prepared statements≫ slik at ≪SQL-injections≫ unngås. Trafikken mellom klient og server går over HTTPS, som krypteres ved hjelp av ≪TLS≫, som er en ende til ende kryptering. Tilgang til systemet sikres ved hjelp av brukernavn og passord. Det er gjort en vurdering på flere sikkerhetstiltak rundt dette, som for eksempel 2-trinns autentisering. Dette benyttes ikke da det ikke eksisterer i systemet fra før av, og det ville vært lite tidseffektivt for gruppen å implementere dette. Mange av kundene til ETC ønsker en enkel prosess rundt det å logge inn, noe 2-trinns autentisering motvirker.

3.8 Risikoanalyse

Det ble utarbeidet en risikoanalyse, se tabell 1 som tar for seg de risikoene som måtte tas hensyn til i oppgaven. Den beskriver sannsynlighet, konsekvens og risiko. Sannsynlighet og konsekvens rangeres fra svært lav til svært høy. Risikoen er et produkt av sannsynligheten og konsekvensen.
Det er viktig at risikoen tilknyttet de mest kritiske scenariene minimeres, da de kan ha betydelig negativ effekt på det endelige produktet. I henhold til dette ble det sett nærmere på de to risikoene som var mest kritisk. Tiltak ble beskrevet for disse og kan leses i tabell 2.

Tabell 1: Risikovurdering

#	Scenario	Sannsynlighet	Konsekvens	Risiko
1	En eller flere av medlemmene i gruppen blir syke, eller kan ikke møte opp av en annen grunn	Middels	Middels	Middels
2	Ved et uhell lekkes sensitiv informasjon tilhørende ETC	Svært lav	Svært høy	Middels
3	Ved utvikling av lagermodulen påvirkes negativt det allerede eksisterende system	Lav	Høy	Middels
4	API eksisterer ikke hos tredjeparts aktører	Middels	Høy	Høy
5	Prosjektet rekker ikke å ferdigstilles	Middels	Høy	Middels
6	At det skjer endringer underveis i arbeidet som krever at det trengs endringer i kravsspesifikasjonen	Lav	Middels	Lav
7	Tap av dokumentasjon og/eller arbeid	Svært lav	Svært høy	Middels

Tabell 2: Risikotiltak

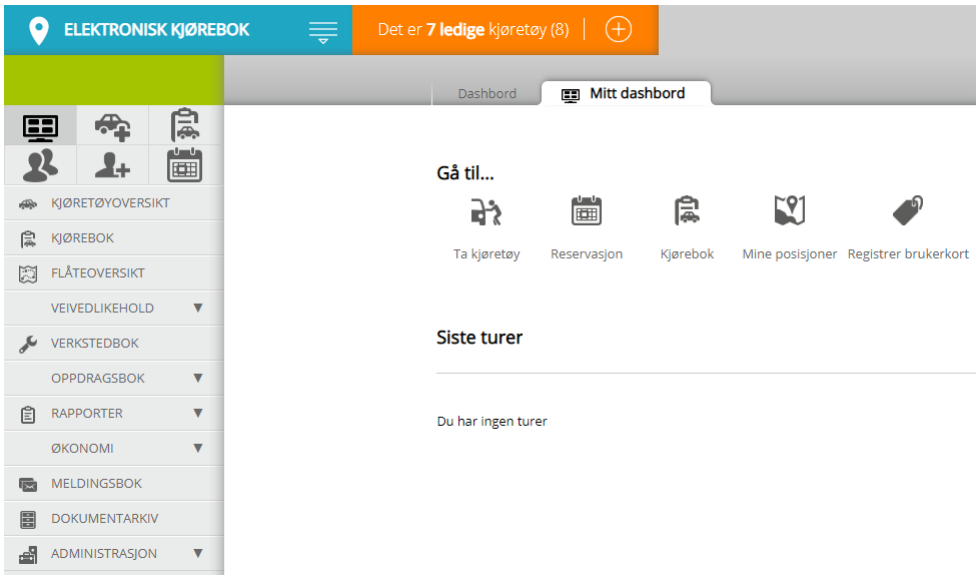
Scenario	**Tiltak**
API eksisterer ikke hos tredjeparts aktører	Være tidlig ute med å sjekke hvilke aktører som har egen API og holde ETC oppdatert på denne fremgangen.
Prosjektet rekker ikke å ferdigstilles	Ut ifra tidligere bacheloroppgaver ser vi at mange oppgdrag ikke ferdigstilles i tide. For å unngå dette setter vi opp en god plan som tar høyde for uforutsette hindringer i utviklingen.

4 Design

I dette kapittelet er det visuelle designet beskrevet. Her illustreres hvordan de ulike menyelementene, dialogene og rapportene er bygget opp i lagermodulen. Beskrivelse av lagermodulens virkemåte er beskrevet i kapittel 6.

4.1 Beskrivelse av eksisterende system

CarAdmin er bygget opp av mange forskjellige seksjoner og det er enkelt å navigere seg gjennom systemet. Her tilbys, som tidligere nevnt, tjenestene kjøretøyregister, kjørebok, fellesbil og flåteoversikt. Seksjonene er listet opp i menyen til venstre, som vist i figur 4.



Figur 4: CarAdmin startside

Denne figuren er hentet fra en annen versjon av CarAdmin for å illustrere alle seksjonene som eksisterer, og ikke bare de i utviklingsmiljøet. Systemet er bygget opp slik at det navigeres gjennom menyer og undermenyer. Ved å klikke seg inn på en av seksjonene åpnes en eller flere faner, heretter omtalt som menyelemen-

ter. Det er mulig å lese ulike rapporter som gir informasjon om skadehistorikk, vedlikehold osv. Det finnes allerede en verkstedbok som har en liknende oppbygning som lagermodulen. Det finnes også en seksjon under «økonomi» som vist i figur 4 med navnet ordreoversikt. ETC har utviklet denne, som lagermodulen må knyttes sammen med i ettertid av levert bacheloroppgave. Dette kan leses mer om i kapittel 8.1.1.

4.2 Design av ny funksjonalitet

Som en egen meny ligger blant annet verkstedbok, ved å klikke seg inn på den vil brukeren få tilgang til et produktregister som vist i figur 5.

Produktnr.	Navn	Beskrivelse	Produkttype	Enhet	Lagerbeholdning	Bilagstype	Produktgruppe	Stykkpris eks mva	Handling
1	Eksternt bilag		Verksted	Stykk	-	Diverse	Diverse	0,00	
100	Faktura	Service i hht fktura	Verksted	Stykk	-	Vedlikehold	Diverse	0,00	
1500	Bremseklosser	Forran begge sider	Verksted	Stykk	7.00	Vedlikehold	Karosseri	500,00	
1601	Kniver	Kniver til klipperagrigat	Verksted	Stykk	-	Vedlikehold	Diverse	285,00	

Figur 5: Produktregister

Produktregisteret har vært et utgangspunkt for å utvikle funksjonalitet til lagermodulen. All funksjonalitet i lagermodulen er uavhengig av produktregisteret, men oppbyggingen har en del likheter. I produktregisteret kan finnes en rapport med all data, det er mulig å legge til nye produkter, utvide for å filtrere og det kan utføres diverse handlinger som redigering og sletting. Dette har vært utgangspunktet for lagermodulen. Derfor finnes det likhetstrekk med produktregisteret sin oppbygning.

Designet av lagermodulen er slik at det samsvarer med resten av CarAdmin. Nesten alt i forbindelse med lagermodulen er samlet under menyen *lageroversikt*, som vist i figur 6.

Figur 6: Lageroversikt

Felles for menyelementene *lagerbeholdning*, *dekkbeholdning* og *lagerregister* er at de har en rapport med data. Det er mulig å søke i, legge til, filtrere, slette og redigere deres data. Ved å klikke på handling vises utvidet informasjon.

Lageroversikt skulle i utgangspunktet inneholde et avansert søkefelt, der det kunne søkes i hele lagermodulen. Dette inkluderer deler, dekk og lagre. Søket har et grunnleggende design og en del kode er skrevet, men funksjonaliteten ble ikke ferdigstilt. Grunnen til at det ikke ble ferdigstilt kan leses i kapittel 8.1.2. Figur 7 viser oppsettet til søkefeltet.

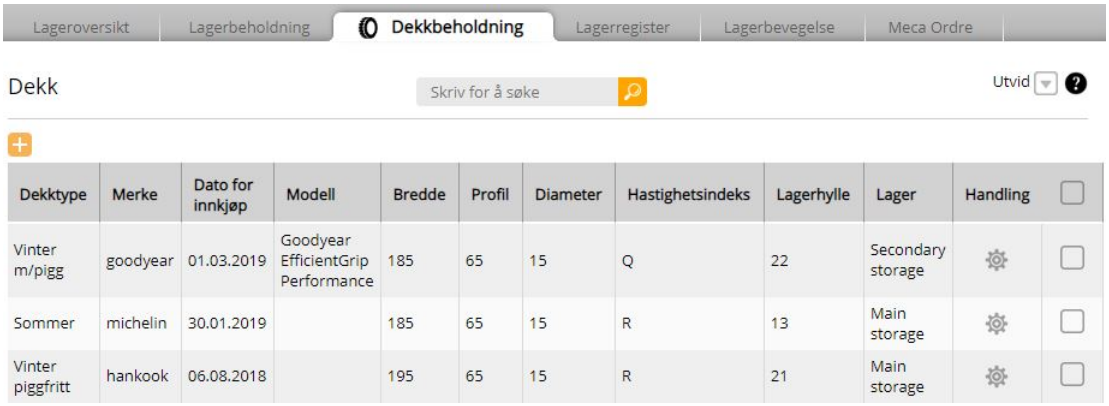
Figur 7: Søkefelt i lageroversikt

Lagerbeholdning som vist i figur 8 gir oversikt over hvilke varer som ligger inne på lager og prisen på disse. Disse er knyttet opp til hvilket lager de befinner seg på. Varene er delt inn i 13 kategorier som det kan filtreres på.

Navn	Beskrivelse	Pris	Beholdning	Enhet	Kategori	Lager	Handling
hjullager	bredde:102, diameter:29, årsmodell:200702	1090	2	stk	Hjullager	Main storage	
bremseskive	diameter:280,høyde:40	558	4	stk	Brems	Main storage	
bremsecaliper	diameter:57, tykkelse:25	1704	1	stk	Brems	Secondary storage	
sonnak 12v	volt:12, amp:600, ampH:61	1860	8	stk	Batteri	Main storage	

Figur 8: Lagerbeholdning

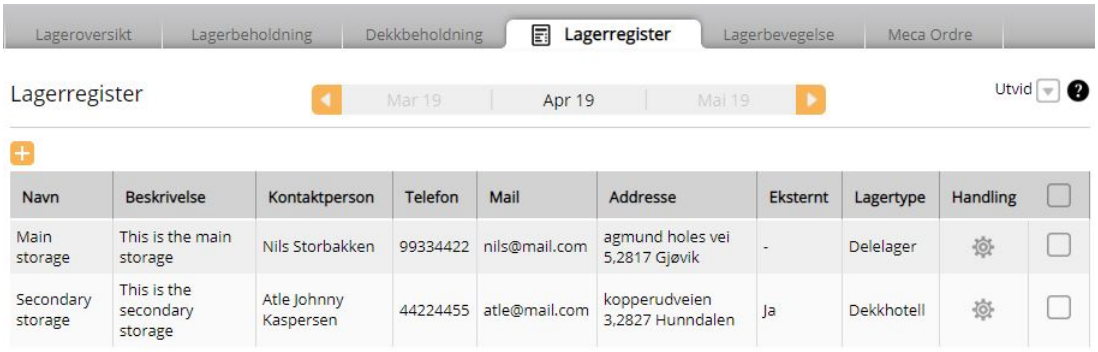
Dekkbeholdning som vist i figur 9 har mange av de samme egenskapene som *lagerbeholdning*, men her holdes det oversikt over alle dekk som finnes på et lager. Det registreres om et dekk er av typen sommer- eller vinterdekk og om det har pigger. Dekkene har en hastighetsindeks som sier hvor fort man kan kjøre med de. Dekkene er også knyttet opp til hvilket lager de befinner seg på.



Dekktype	Merke	Dato for innkjøp	Modell	Bredde	Profil	Diameter	Hastighetsindeks	Lagerhylle	Lager	Handling
Vinter m/pigg	goodyear	01.03.2019	Goodyear EfficientGrip Performance	185	65	15	Q	22	Secondary storage	
Sommer	michelin	30.01.2019		185	65	15	R	13	Main storage	
Vinter piggfritt	hankook	06.08.2018		195	65	15	R	21	Main storage	

Figur 9: Dekkbeholdning

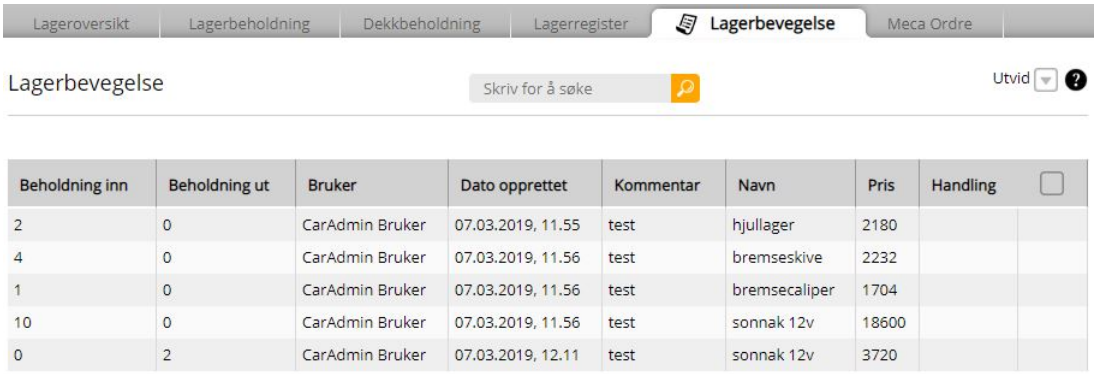
Lagerregister som vist i figur 10 gir oversikt over alle kundens lagre. Disse registreres som eksterne eller interne, og videre om de er et dekkhotell eller et delelagre.



Navn	Beskrivelse	Kontaktperson	Telefon	Mail	Addresse	Eksternt	Lagertype	Handling
Main storage	This is the main storage	Nils Storbakken	99334422	nils@mail.com	agmund holes vei 5,2817 Gjøvik	-	Delelager	
Secondary storage	This is the secondary storage	Atle Johnny Kaspersen	44224455	atle@mail.com	kopperudveien 3,2827 Hunndalen	Ja	Dekkhotell	

Figur 10: Lagerregister

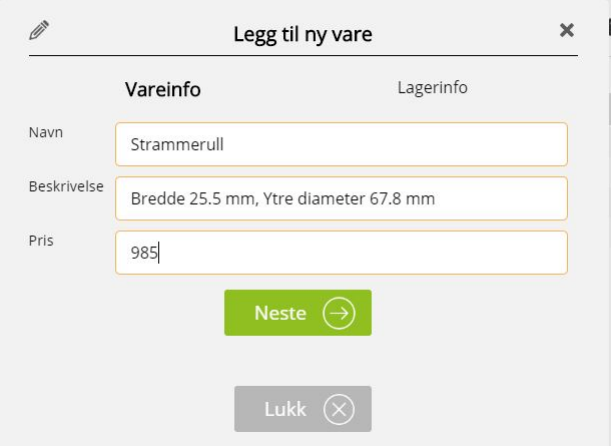
Lagerbevegelse som vist i figur 11 inneholder en rapport som gir informasjon om når en vare er lagt inn eller tatt ut av lager. Her vises navnet på varen, og summen av varen multiplisert med antall, utgjør prisen. Det gis også informasjon om hvilken bruker som har endret beholdningen.



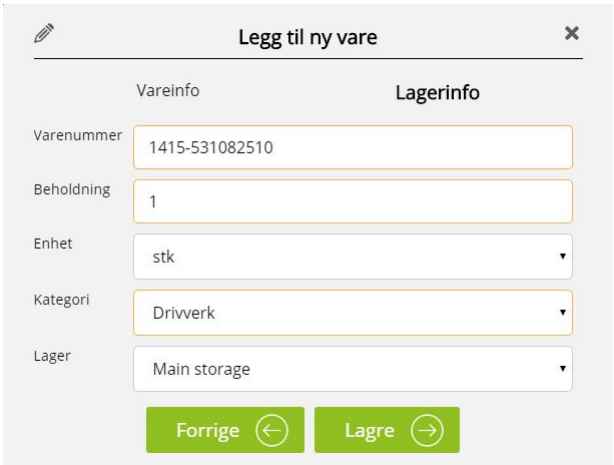
Beholdning inn	Beholdning ut	Bruker	Dato opprettet	Kommentar	Navn	Pris	Handling
2	0	CarAdmin Bruker	07.03.2019, 11.55	test	hjullager	2180	
4	0	CarAdmin Bruker	07.03.2019, 11.56	test	bremseskive	2232	
1	0	CarAdmin Bruker	07.03.2019, 11.56	test	bremsecaliper	1704	
10	0	CarAdmin Bruker	07.03.2019, 11.56	test	sonnak 12v	18600	
0	2	CarAdmin Bruker	07.03.2019, 12.11	test	sonnak 12v	3720	

Figur 11: Lagerbevegelse

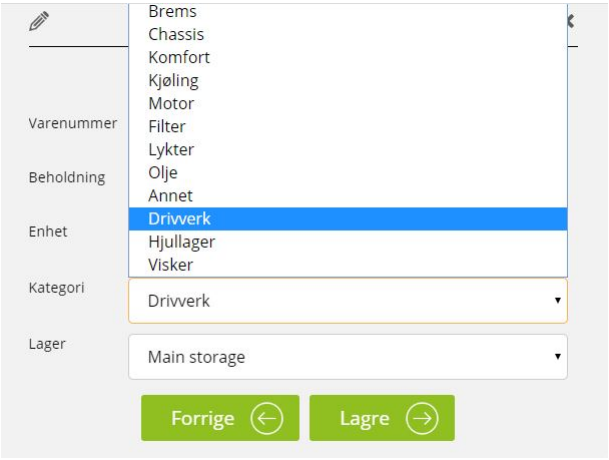
Figurene 12, 13 og 14 viser de ulike stegene i dialogen for å legge til en vare i *lagerbeholdning*. Varen registreres med navn, beskrivelse, pris, varenummer, beholdning, enhet, kategori og hvilket lager varen legges inn på.



Figur 12: Dialog lagerbeholdning steg 1



Figur 13: Dialog lagerbeholdning steg 2

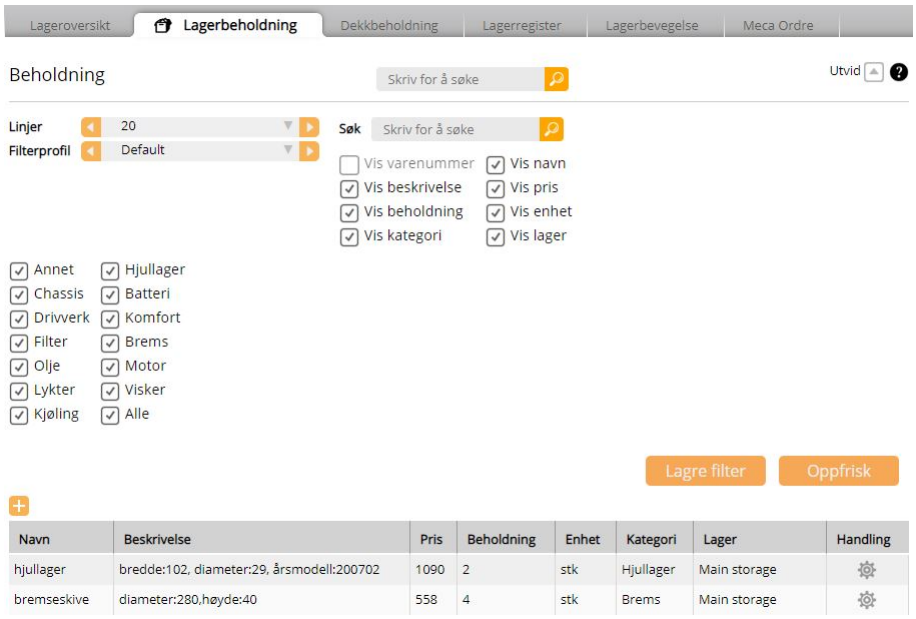


Figur 14: Dialog lagerbeholdning kategori

Det kan søkes i all data som ligger i rapporten ved å skrive i søkefeltet som vist i figur 15. Søket finner alt som samsvarer med teksten skrevet i søkefeltet.

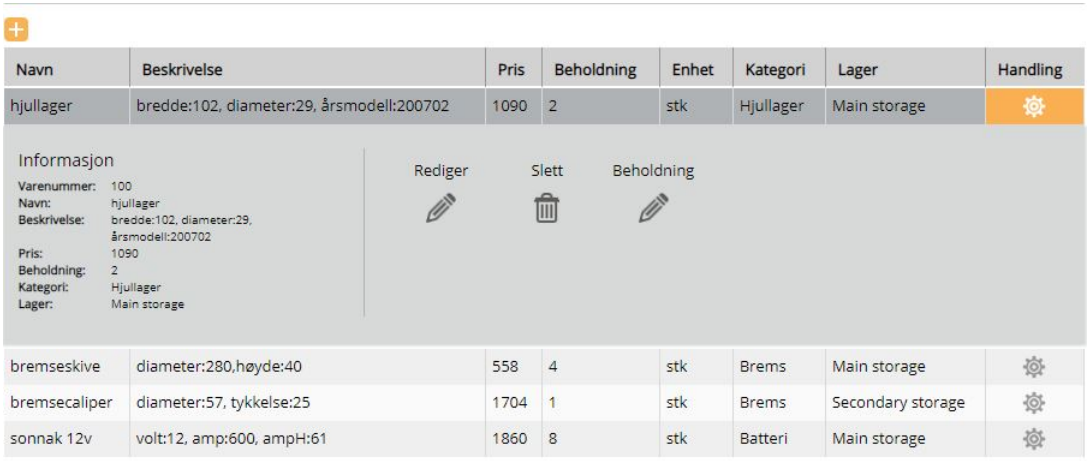
Figur 15: Søkefelt i lagerbeholdning

Ved å trykke på knappen utvid som vist i figur 16 får man mulighet til å filtrere dataen. Rapporten oppdateres med hensyn på filtreringen.



Figur 16: Filtrering av lagerbeholdning

Ved å klikke seg inn på handling som vist i figur 17 vil det vises utvidet informasjon. Man får mulighet til å slette, redigere og endre beholdning av varen ved hjelp av ulike dialoger. Dialogene for disse er vist i figurene 18, 19 og 20.



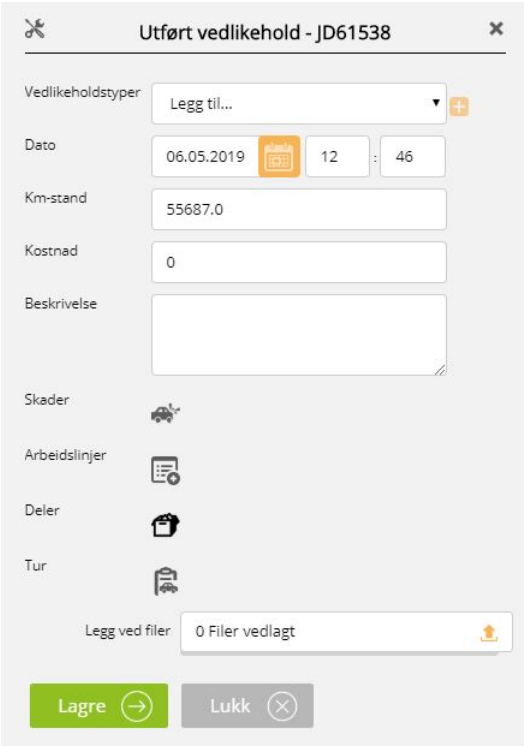
Figur 17: Detaljer i lagerbeholdning

Figur 18: Redigering i lagerbeholdning

Figur 19: Sletting i lagerbeholdning

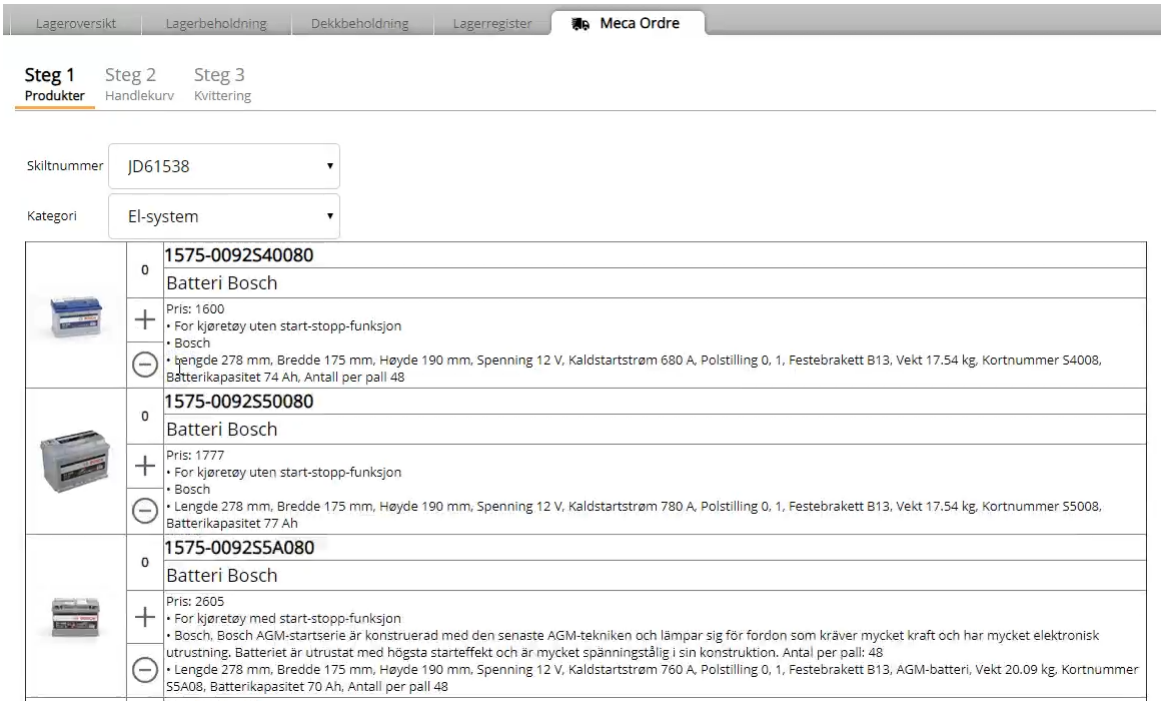
Figur 20: Oppdatere lagerbeholdning

I kjøretøyoversikten er det mulig å registrere et utført vedlikehold av en bil. Her skulle det registreres hvilke deler som ble brukt, og som videre skulle inngå i en ordre. Figur 21 viser den nye funksjonaliteten med navn på deler som er lagt til, som en egen seksjon i dialogen. Denne skulle inneholde et søk på *lagerbeholdning* og varen ville bli lagt til i vedlikeholdet ved hjelp av en egen dialog. Ved uttak av varen skulle rapporten i *lagerbevegelse* oppdateres. Grunnen til at dette ikke ble fullført kan leses om i kapittel 8.1.2.



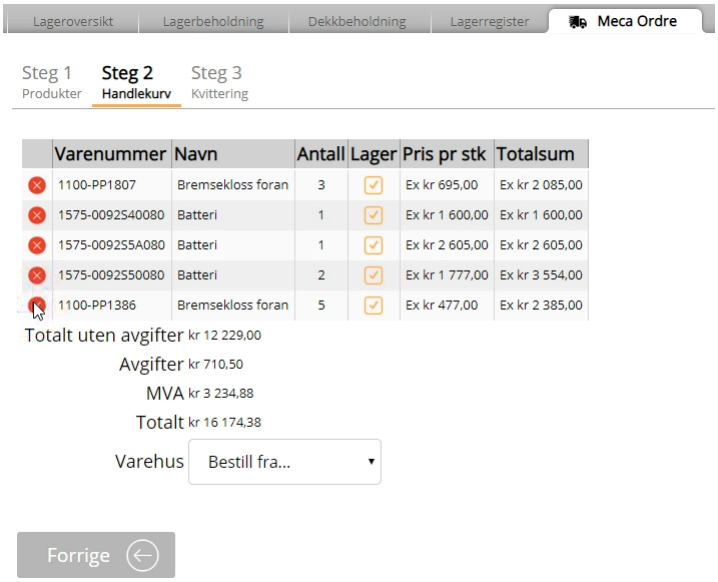
Figur 21: Legge til deler i utført vedlikehold

Det er mulig å bestille fra tredjepartsleverandøren MECA internt i CarAdmin. Som vist i figur 22 kan det velges deler ut ifra skiltnummer og kategori på del. Skiltnummer hentes fra eksisterende biler i CarAdmin. Løsningen er dynamisk, det vil si at siden oppdateres når skiltnummer eller kategori endres.

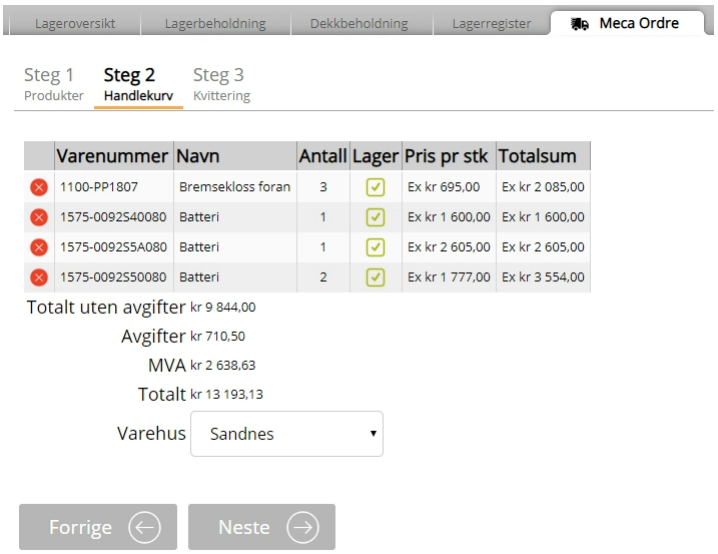


Figur 22: Bestille deler

Etter alle varer er lagt til i handlekurven kan disse fjernes eller kjøpet kan utføres. I handlekurven velges hvilket lager varene skal bestilles fra, som vist i figur 23. Fargen under kolonnen «Lager» oppdateres fra rød til grønn dersom varen er på det valgte lageret, som vist figur 24.



Figur 23: Handlekurv



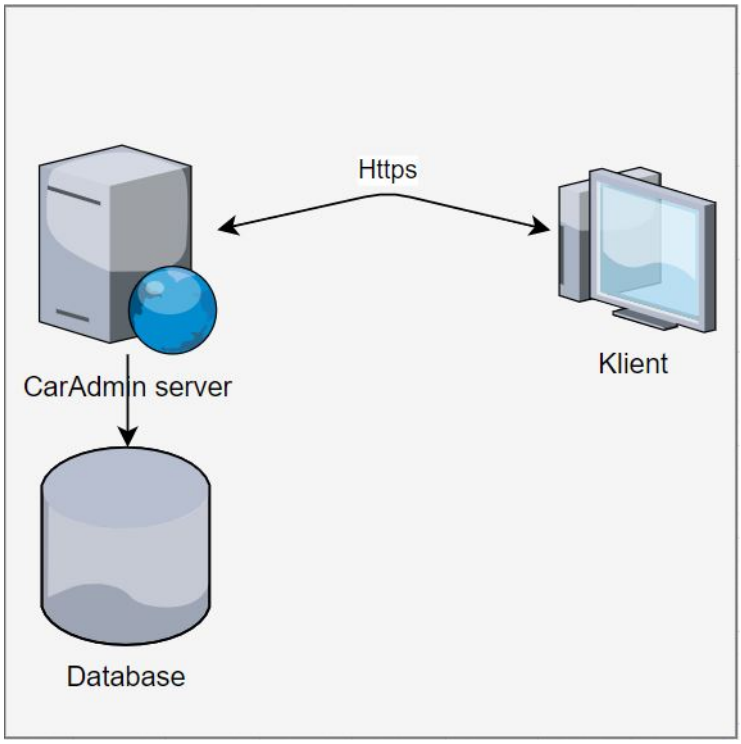
Figur 24: Lagerstatus

5 Arkitektur

Dette kapittelet inneholder informasjon om hvordan CarAdmin er bygget opp. Etter å ha lest dette kapittelet vil det være klart hva utgangspunktet var da utviklingen av lagermodulen startet. Gruppen måtte følge systemets eksisterende arkitektur som er beskrevet her.

5.1 Systemarkitektur

Figur 25 viser en overordnet oversikt over den eksisterende løsningen.



Figur 25: Overordnet oversikt over systemet

CarAdmin kjøres på Tomcat[7] på en webserver hos ETC. Klientene er tynne ettersom løsningen aksesseres via en nettleser. Informasjon lagres på og hentes fra

en MariaDB-database[8] ved hjelp av SQL-spørringer. Kommunikasjonen mellom server og klient foregår med AJAX (Asynchronus JavaScript And XML), som gjør det mulig å oppdatere innhold på en nettside, uten at den må lastes inn på nytt.

5.1.1 Site-Router

For å navigere brukeren gjennom systemet er det satt opp en *Site-Router*. Denne henter ut underliggende kontrollerklasser ettersom brukeren navigerer seg gjennom systemet.

5.1.2 Page-Controller

PageController er en av disse kontrollerklassene som brukes når en bruker navigerer seg gjennom systemet. Denne har i oppgave å klargjøre data som trengs for den aktuelle siden som skal lastes. Alle *PageControllere* har en tilhørene .jsp-fil som brukes for å opprette et HTML-dokument, som brukeren mottar idet siden er lastet. Dette dokumentet er i utgangspunktet statisk, men ved hjelp av JavaScript-funksjoner og AJAX kan dette oppdateres dynamisk.

5.1.3 Menyelement

For å strukturere og definere når disse kontrollerene blir tilkalt, brukes det *menyelementer*. Disse knytter en kontroller sammen med en av menyene brukeren har tilgang til.

5.1.4 AJAX

AJAX er et rammeverk som brukes når en bruker utfører en handling i CarAdmin. Enten ved lagring av data, uthenting av data eller andre endringer som skal utføres. Denne prosessen starter med en JavaScript-funksjon som tilkalles. Her hentes det ut data som sendes over til en AJAX-klasse. Disse klassene er skrevet i Java og satt opp til å utføre en av endringene tidligere nevnt. Resultatet av dette ender med

et svar som sendes tilbake til JavaScript-funksjonen. Dette tillater at det statiske HTML-dokumentet til brukeren kan oppdateres dynamisk ettersom endringer blir utført.

5.1.5 Dialog

Et annet eksempel på en kontrollerklasse er *Dialog*. Denne klassen brukes for å opprette et nytt vindu inne i systemet, slik at brukeren kan fylle inn ekstra informasjon. Et eksempel på dette er når brukeren skal legge til en ny vare på lager. Da brukes en *Dialog* for å hente inne informasjonen om den nye varen.

5.1.6 Factory

Factory er et designpattern omtalt i boken *Elements of Reusable Object-Oriented Software*[3]. I CarAdmin er dette et grensesnitt som brukes for å hente ut riktig versjon av en klasse. Eksempelvis brukes dette for å hente ut klasser som skal ha begrenset tilgang til den aktuelle kundens system.

5.1.7 IO grensesnitt

I CarAdmin benyttes et *IO*-grensesnitt, som brukes til å hente ut data. Dette implementeres for eksempel i *SQLIO*-klasser som henter denne dataen fra databasen.

5.1.8 Messages

Internasjonaliseringen i CarAdmin bygger på Java sitt bibliotek, ≪Resourcebundle≫[15], som er implementert i en *messages*-klasse. Dette gjør at programvaren støtter engelsk og norsk.

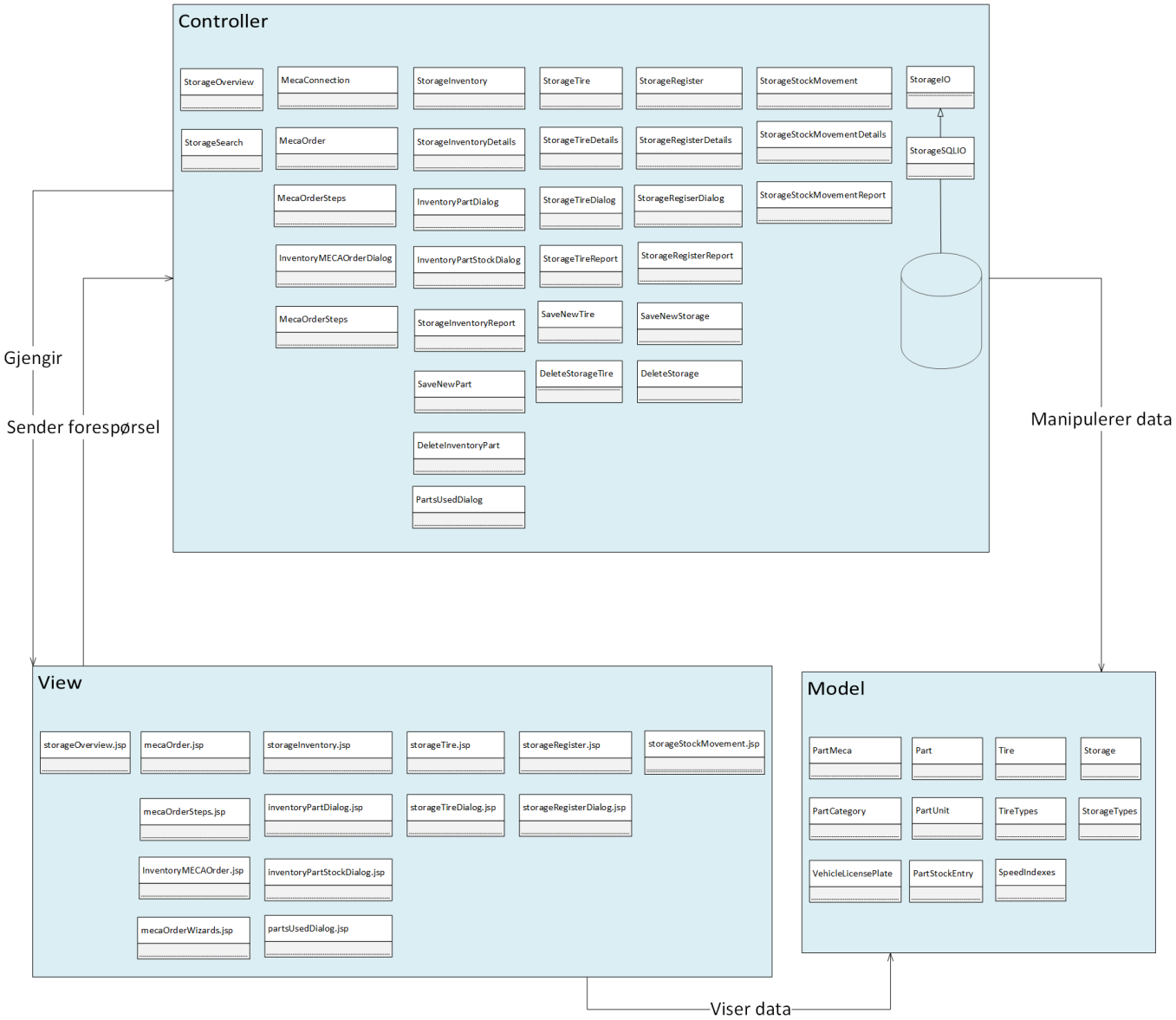
6 Implementasjon

Dette kapittelet handler om hvordan funksjonaliteten til lagermodulen fungerer og hvordan den ble implementert. Her beskrives også databaseoppsett og kommunikasjonen med den. Prosjektet har resultert i 7565 kodelinjer inkludert kommentarer fordelt på 62 filer, hvorav 58 av disse er blitt opprettet i løpet av prosjektperioden. Følgende liste viser fordelingen:

- AJAX: 1617 linjer
- IO: 775 linjer
- Controller: 3063 linjer
- JSP: 1740 linjer
- MenuElements: 6 linjer
- Messages: 364 linjer

6.1 Model-View-Controller

Store deler av CarAdmin er strukturert etter *Model-View-Controller*, heretter omtalt som MVC. Lagermodulen følger den eksisterende oppbygningen. Figur 26 viser inndelingen.



Figur 26: Model-View-Controller

I *controller* ligger alle klassene som sørger for å manipulere data og oppdatere *view*. *Model* tar vare på den ulike dataen og viewet sørger for å vise det. Alle klasser som henger sammen er plassert loddrett. Disse er plassert fra høyre til venstre samsvarende mellom de ulike delene. Eksempelvis ligger *StorageOverview* helt til venstre i *controller* og den tilknyttede .jsp-filen *storageOverview.jsp* ligger helt til venstre i *View*.

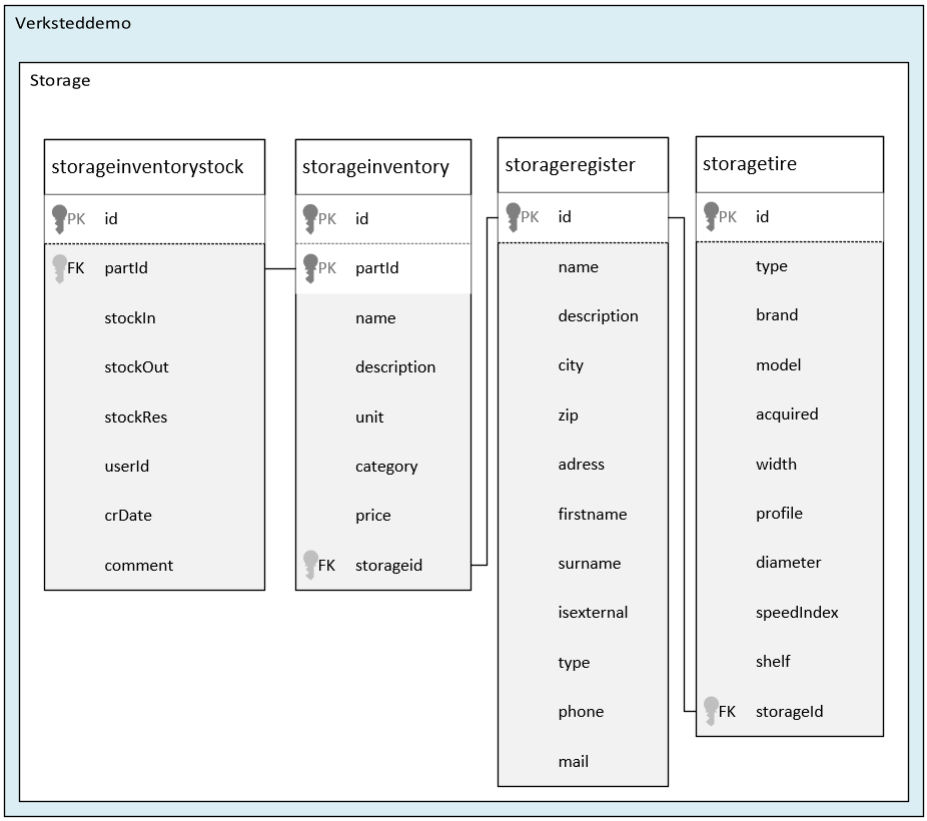
For å illustrere hvordan MVC fungerer i lagermodulen vises det her et overordnet eksempel på hvordan lagerbeholdningen er satt sammen og hva som skjer ved noen ulike handlinger ifb. med *controller*, *model* og *view*. Ved å klikke seg inn på menyelementet *Lagerbeholdning* tilkalles *storageInventory.jsp* som ligger i *view* fra *StorageInventory* i *controller*. Rapporten til lagerbeholdningen er satt opp i *StorageInventoryReport*. Denne henter data ved hjelp av *StorageIO* og *StorageSQLIO* som inneholder SQL-spørringer til databasen fra *Part* og *PartUnit* som ligger i *model*.

Ved å klikke på tannhjulet kalles *StorageInventoryDetails* i *controller* og *view* oppdateres igjen. *StorageInventoryDetails* henter også data fra *model*. Her er det også mulighet til å redigere, endre og slette beholdningen. Ved sletting tilkalles *DeleteInventoryPart* som sender en SQL-spørring til *StorageIO* grensesnittet.

Ved å klikke på pluss-knappen for å legge til en vare, tilkalles *InventoryPartDialog* i *controller* og *inventoryPartDialog.jsp* i *view*. Ved å klikke «lagre» i dialogen vil den nye varen lagres ved hjelp av *SaveNewPart* som tilkaller *StorageIO*. Det blir da lagt til ny informasjon i *model* og *view* oppdateres.

6.2 Oppsett av database

Databasen ble satt opp i MariaDB[8] og administrert ved hjelp av HeidiSQL[16]. Figur 27 viser hvilke tabeller som ble laget for lagermodulen.



Figur 27: Databaseoppsett

I databasen har ikke ETC brukt «foreign keys» og derfor måtte tabellene til lagermodulen følge samme oppsett. All databaselogikk løses i koden. Selv om figuren viser «foreign keys», er dette kun for å illustrere hvordan tabellene henger sammen. Det ble lagt til data i forbindelse med lagermodulen i flere eksisterende tabeller i databasen. Disse dataene sørget for å gi korrekte rettigheter til *løsningsadministrator*. CarAdmin har et komplisert produktstyring- og tilgangskontrollsystem. ETC ønsket ikke at denne arkitekturen skulle vises, så derfor er

dette ikke illustrert i figuren. De nye tabellene holder på data for de ulike meny-elementene i lagermodulen. ≪id≫ er primærnøkkel i alle tabellene. Denne er ikke synlig i lagermodulen. ≪id≫ telles opp med én for hver rad som legges til.

Tabellene *storageinventory* og *storageinventorystock* inneholder data om lager-beholdningen. Sistnevnte ble laget for å holde oversikt over lagerbevegelse ved innkjøp, reservasjoner og bruk av deler. Tabellene er koblet sammen ved hjelp av ≪partId≫, som er produktnummeret til en del. *Storageregister* er tabellen for lagerregisteret. Dette er tabellen med mest data. Grunnen til det er informasjon om et lager og i tillegg opplysninger om lageransvarlig. *Storagetire* er tabellen for dekkbeholdning, som inneholder data om dekk.

Lagerbeholdning og dekkbeholdning er begge koblet sammen med lagerregisteret slik at det står oppført hvilket lager en del eller et dekk befinner seg på.

6.3 Kommunikasjon med database

For å lagre ny data eller hente data fra databasen ble egne IO-klasser brukt til SQL-spørringer. *Prepared statements* ble brukt for å unngå «SQL-injection». *Storage-SQLIO* er klassen som inneholder funksjoner for å aksessere dataene i tabellene for lagermodulen. Figur 28 viser et kodeeksempel på hvordan dekk lagres i databasen:

```
137     /**
138      * Lagrer nye dekk i DB
139      * @param tire Dekk som skal legges inn
140      * @throws CarAdminException
141      */
142     @Override
143     public void saveTire(Tire tire, SolutionSessionInfo si) throws CarAdminException {
144         try {
145             if(tire.getId() > 0){
146                 String sql = "UPDATE storagetire SET type=?, brand=?, model=?, acquired=?, width=?, profile=?, diameter=?, speedIndex=?, shelf=?, storageId=? WHERE id =?";
147                 int idx = 1;
148                 try(Connection c = sqlFactory.getConnection()) {
149                     try (PreparedStatement s = c.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {
150                         s.setString(idx++, tire.getType());
151                         s.setString(idx++, tire.getBrand());
152                         s.setString(idx++, tire.getModel());
153                         s.setString(idx++, tire.getAcquired().getFormatedDate(SQLBuilder.getDateFormat()));
154                         s.setInt(idx++, tire.getWidth());
155                         s.setInt(idx++, tire.getProfile());
156                         s.setInt(idx++, tire.getDiameter());
157                         s.setString(idx++, tire.getSpeedIndex());
158                         s.setInt(idx++, tire.getShelf());
159                         s.setInt(idx++, tire.getStorage());
160                         s.setInt(idx++, tire.getId());
161                         s.executeUpdate();
162                     }
163                 }
164             }else{
165                 String sql = "INSERT INTO storagetire (type, brand, model, acquired, width, profile, diameter, speedIndex, shelf, storageId) VALUES (?,?,?,?,?,?,?,?,?,?)";
166 
167                 try(Connection c = sqlFactory.getConnection()) {
168                     try (PreparedStatement s = c.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {
169                         s.setString(1,tire.getType());
170                         s.setString(2, tire.getBrand());
171                         s.setString(3, tire.getModel());
172                         s.setString(4, tire.getAcquired().getFormatedDate(SQLBuilder.getDateFormat()));
173                         s.setInt(5, tire.getWidth());
174                         s.setInt(6, tire.getProfile());
175                         s.setInt(7, tire.getDiameter());
176                         s.setString(8, tire.getSpeedIndex());
177                         s.setInt(9, tire.getShelf());
178                         s.setInt(10, tire.getStorage());
179                         s.executeUpdate();
180 
181                         try (ResultSet rs = s.getGeneratedKeys()) {
182                             if (rs.first()) {
183                                 int tireid = rs.getInt(1);        //henter ny id
184                                 tire.setId(tireid);
185                             }
186                         }
187                     }
188                 }
189             }
190         } catch (SQLException e) {
191             throw new CarAdminException("Feil ved lagring av dekk",e);
192         }
193     }
```

Figur 28: Funksjon for lagring av dekk i database

Samme oppsett brukes for andre lagrefunksjoner i lagermodulen, bare med andre data. Denne funksjonen har en «if-else» setning som sjekker om et dekk som skal lagres, finnes i databasen fra før. Det ble satt opp på denne måten så funksjonen kan brukes ved både registrering av nye og redigering av eksisterende dekk. Om dekket finnes fra før kjøres SQL-spørringen «UPDATE...», hvis ikke kjøres SQL-spørringen «INSERT INTO...». I figuren er «s» en *prepared statement* som brukes til å sette dataene på riktig plass i tabellen. «tire» er et såkalt *Bean*-objekt som holder på all data om et dekk. Rekkefølgen på kolonnene i dekkbeholdning er *dekktype*, *merke*, *dato for innkjøp*, *modell*, *bredde*, *profil*, *diameter*, *hastighetsindeks*, *lagerhylle* og *lager*. Dette samsvarer med rekkefølgen dataene legges til i funksjonen.

Det er også forskjell på hvordan ID-en til et dekk legges til ut ifra om dekket finnes i databasen fra før eller ikke. ID-en er skjult fra visning da denne kun brukes til å telle opp rader i databasen. Et annet eksempel er det å slette et dekk fra databasen. Figur 29 viser denne funksjonen:

```
195     /**
196      * Fjerner dekk fra DB
197      * @param id
198      * @throws CarAdminException
199      */
200     @Override
201     public void deleteTire(int id) throws CarAdminException {
202         String sql = "DELETE FROM storagetire WHERE id = ?";
203         try {
204             sqlFactory.executeUpdate(sql, id);
205         } catch (SQLException e) {
206             throw new CarAdminException("Feil ved sletting av dekk",e);
207         }
208     }
```

Figur 29: Funksjon for sletting av dekk fra database

Disse funksjonene inneholder et *sqlFactory*-objekt. Dette brukes som kobling til databasen.

6.3.1 Vurdering av databaseoppsett

Tabellene *storageinventory*, *storageinventorystock*, *storageregister* og *storagetire* er i hovedsak satt opp utifra deres tilhørende rapporter. For det meste brukes datatypene *INT* og *VARCHAR*. *INT* brukes med en lengde på 11 som passer for de aller fleste verdier. *VARCHAR* brukes til «strings» i programvaren med varierende lengde. Ved «booleans» brukes datatypen *TINYINT*. Denne har en lengde på 4 og dette gjør at verdien tar så lite plass som mulig i databasen. For datoer brukes datatypen *DATE* og for dato med tidspunkt brukes *DATETIME*. Disse datatypene gjør det enkelt å håndtere datoer og tid. Alle tabellene har en id som er «primary key». Id'en settes alltid ved bruk av *AUTO-INCREMENT*. I tabellen *storageinventorystock* ligger *partId* som referer til *id* i tabellen *storageinventory*. Denne er i praksis en «foreign key» gjennom kode, men er ikke satt opp i databasen som dette. Årsaken til dette er som tidligere nevnt i kapittel 6.2.

6.3.2 Vurdering rundt «foreign keys»

Som nevnt er CarAdmin i dagens løsning implementert uten bruk av «foreign keys» i databasen. Årsaken til dette er at CarAdmin er utviklet over lang tid og det ikke var naturlig å implementere «foreign keys» da utviklingen startet opp. I praksis løses dette nå i koden og dette er tungvint. Det ville vært hensiktsmessig å implementere et mer moderne design på databasen med «foreign keys». Ved sletting av data håndteres dette automatisk i forhold til databasen når «foreign keys» tas i bruk. Fordeler ved å implementere dette ville derfor vært enklere koding på sikt da mye automatiseres. En stor fordel ved å gjøre dette er at det blir enklere å se sammenhengen mellom de ulike tabellene og det gir mer oversikt. Slik løsningen er bygd opp nå kan det være vanskelig å sette seg inn i hvordan databasen henger sammen. Kortsiktig vil det være en del jobb å skrive om systemet slik at «foreign keys» vil fungere, men på lang sikt vil dette gi store fordeler, spesielt for nyansatte som skal sette seg inn i systemet.

Som tidligere nevnt fulgte gruppen utviklingsmodellen inkrementell-sekvensiell. Her er hva som ble gjort i de ulike inkrementene.

6.4 Inkrement 1

Dette inkrementet ble påbegynt den 25.februar som var 7 dager før planen.

6.4.1 Gjøremål

Følgende funksjonalitet skulle i utgangspunktet utvikles i det første inkrementet:

- Lagerregister
- Lagerbeholdning
- Import av delekatalog

6.4.2 Resultat

Følgende funksjonalitet ble utviklet i det første inkrementet:

- Menyelementer
- Komplett lagerbeholdning
- Dekkbeholdning (Uten dekktype, hastighetsindeks og lagerhylle)
- Lagerregister nesten komplett

6.4.3 Oppsummering

Et skjelett til lagermodulen var det første som ble satt opp. Under menyelementet *Lageroversikt*, ble menyelementene *lageroversikt*, *lagerbeholdning*, *dekkbeholdning* og *lagerregister* lagt til. Det ble tidlig i denne perioden bestemt at dekkbeholdning skulle skilles fra lagerbeholdning på grunn av veldig forskjellige egenskaper. Dette ble løst ved å innføre dekkbeholdning som et eget menyelement.

Lagerbeholdningen ble ferdigstilt. Samtidig ble et lagerregister og dekkbeholdning påbegynt og nesten fullført. Dekkbeholdningen ble egentlig ferdigstilt, men det dukket opp et ønske om flere kolonner med andre data mot slutten av inkrement 2. «Import av delekatalog» ble satt opp i dette inkrementet fordi gruppen tenkte det ville bli en enkel oppgave å liste ut alle tilgjengelige deler hvis en oppkobling mot et API lot seg gjøre. Siden ingen tredjepartsleverandører hadde API, kunne ikke dette gjennomføres på dette stadiet.
På slutten av det første inkrementet ble dette presentert for ETC. De var fornøyde, men de mente også at det var mer å gå på. Dette tok gruppen med seg videre inn i inkrement 2.

6.5 Inkrement 2

Dette inkrementet ble påbegynt som planlagt den 18.mars.

6.5.1 Gjøremål

Følgende funksjonalitet skulle i utgangspunktet utvikles i det andre inkrementet:

- Oppkobling mot API
- Bestilling
- Ordre

6.5.2 Resultat

Følgende funksjonalitet ble utviklet i det andre inkrementet:

- Komplett dekkbeholdning (gikk to dager ut i inkrement 3)
- Søkefelt under lageroversikt påbegynt
- Bestilling via ≪Web-crawling≫ påbegynt
- Ordreoversikt påbegynt

6.5.3 Oppsummering

Dette inkrementet måtte endres totalt grunnet situasjonen med API. Det var på dette stadiet at utviklingsmodellen måtte gå over til å bli smidig i stedet for plandrevet. Grunnet ingen tilgang til API, måtte gruppen tenke nytt når det gjaldt bestilling. Dekkbeholdningen ble fullført. Den var egentlig ferdig, men etter ønske om flere kolonner med data, ble dette lagt til. En ordreoversikt og et avansert søkefelt ble påbegynt. Det ble bestemt på et møte med ETC at det var ønskelig med en alternativ løsning for bestilling av deler. Dette førte til starten på ≪Web-crawl≫ løsningen. Mer om dette kan leses i kapittel 8.1.1.

6.6 Inkrement 3

Dette inkrementet ble påbegynt den 10.april som var 2 dager etter planen.

6.6.1 Gjøremål

Følgende funksjonalitet skulle i utgangspunktet utvikles i det tredje inkrementet:

- Innkjøpsreservasjon av deler
- Lagerbevegelse
- Internasjonalisering
- Søkefelt under lageroversikt
- Knapp for uttak av deler ved registrering av utført vedlikehold

6.6.2 Resultat

Følgende funksjonalitet ble utviklet i det tredje inkrementet:

- Bestilling fra tredjepartsleverandør (GUI)
- Lagerbevegelse
- Internasjonalisering (foregått fortløpende gjennom hele utviklingsperioden)
- Uttak av deler ved registrering av utført vedlikehold påbegynt (Skal vises i lagerbevegelse)
- Knapp for registrering av utført vedlikehold ble ikke helt ferdig
- Søkefelt under lageroversikt avbrutt

6.6.3 Oppsummering

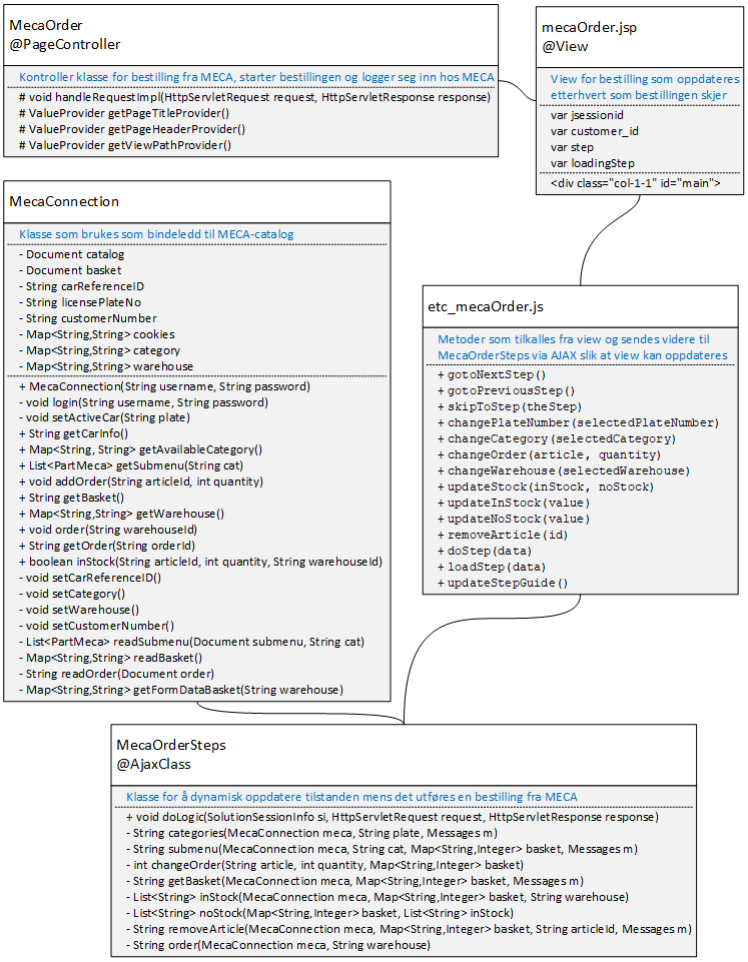
Mye av tiden i dette inkrementet ble brukt til å fullføre bestilling fra tredjeparts-leverandør. Samtidig ble det utarbeidet et menyelement for lagerbevegelse. Her skulle det listes opp uttak av deler med informasjon om *Beholdning inn*, *Beholdning ut*, *Bruker*, *Dato opprettet*, *Kommentar*, *Navn* og *Pris*. I dialogen for registrering av utført vedlikehold (se figur 21 fra kapittel 4) ble det lagt til en ny knapp som heter **Deler**. Her var tanken at deler som ble brukt kunne registreres etter utført vedlikehold og at lagerbevegelsen skulle oppdateres deretter. For å lage denne funksjonaliteten måtte det søkes gjennom lagerbeholdningen for å velge deler, og dette ba på utfordringer. Gruppen kom ikke i mål med dette. Internasjonalisering ble satt opp som et eget punkt i det siste inkrementet. Gantt-skjemaet fra januar (se vedlegg F) ble laget før gruppen visste hvordan CarAdmin-prosjektet var bygd opp. Det fantes gode rutiner for internasjonalisering som gjorde det mulig å gjøre dette fortløpende i utviklingsperioden. Søkefeltet som ble påbegynt under *Lageroversikt*, var utfordrende. Det ble til slutt for tidkrevende, og dermed lagt til side.

6.7 Bestilling fra tredjepartsleverandør

Som tidligere nevnt fantes det ingen leverandører som hadde et API for bestilling av varer. Her ble det tatt et valg i samarbeid med ETC om å bruke MECA sin nettbutikk for å utføre bestillingen. Den enkleste løsningen hadde vært å legge ved en link i applikasjonen som åpner nettsiden til MECA. Ulempen med dette er at det ikke ville blitt en kobling mellom systemet til MECA og CarAdmin. Derfor ble valget å gå for en ≪web-crawl≫ løsning, slik at ved bestilling hos MECA legges varene inn i Lagermodulen. Dette ble den nærmeste løsningen til bestilling via API.

6.7.1 UML klassediagram

For å kunne sette opp en slik løsning trengtes en del klasser og i figur 30 vises klassene som er involvert.



Figur 30: Bestilling domenemodell

6.7.2 MecaOrder.java

Denne klassen arver fra ETC sin *PageController*-klasse og er den første som blir tilkalt. Denne tilkalles idet brukeren velger å gå til menyelementet *MecaOrder*.

Her sjekkes det om brukeren skal ha mulighet til å gjøre en bestilling fra MECA. Deretter opprettes objektet *MecaConnection* der brukernavn og passord for denne brukeren logges inn hos MECA. Her er det lagt opp til at hver bruker har sitt eget brukernavn i MECA sitt system, slik at hvis flere brukere er inne samtidig, så legges varene i to separate handlekurver. *MecaConnection*-objektet puttes deretter inn som en parameter. Dette tillater at det senere kan hentes ut og tilstanden kan ivaretas gjennom hele prossessen. Det første steget i bestillingen er valg av bil slik at MECA kan søke etter deler som passer til denne. Disse bilene ligger allerede inne i CarAdmin, derfor hentes de ut og plasseres inn i første steg av bestillingen. Alle kontrollere som er av typen *PageController* er koblet sammen med en .jsp-fil som setter opp et HTML-dokument for viewet til siden.

6.7.3 MecaOrder.jsp

Denne tilkalles etter *MecaOrder-controlleren* er ferdig og setter opp HTML-koden som utgjør viewet til bestillingen. Denne filen er HTML-dokumentet som sendes ut til brukere, men siden dette er en statisk side så trengs det JavaScript-funksjoner som tilkalles i det brukeren gjør et valg. Disse funksjonene ligger i *etc_mecaOrder.jsp* og blir koblet sammen med HTML-elementer som brukeren kan gjøre endringer på.

6.7.4 etc_mecaOrder.js

Alle elementer der brukeren kan gjøre et valg under bestillingen er koblet opp mot en funksjon som ligger i *etc_mecaOrder.jsp*. Disse funksjonene henter ut valget brukeren gjorde og sender dette videre til *MecaOrderSteps* via AJAX. Etter endringen har blitt utført sendes det et svar tilbake til JavaScript-funksjonen i form av et JSON-objekt. Dette er endringen som ble utført, og dette svaret blir lagt til som en endring av HTML-dokumentet. Dette er logikken som tillater at HTML-dokumentet kan oppdateres dynamisk.

6.7.5 MecaOrderSteps.ajax

Mottar et JSON-objekt fra en JavaScript-funksjon som ble tilkalt av brukeren. Denne inneholder data som sier hvilke steg brukeren er på, og et valg som ble tatt. Oppgaven til denne AJAX-klassen er å igangsette en endring i MECA sitt system, og sette opp et svar som sendes tilbake til JavaScript-funksjonen. Dette svaret kan inneholde et eller flere HTML-elementer som settes opp inne i denne Java-klassen. Selv om det ville være naturlig å fordele oppsettet av HTML-elementer utover forskjellige .jsp-filer, ble ikke dette gjort siden det er enklere å håndtere dataen som ligger i Java-objekter inne i denne Java-klassen. Ut fra valget som ble tatt tilkalles en av funksjonene som ligger i *MecaConnection*.

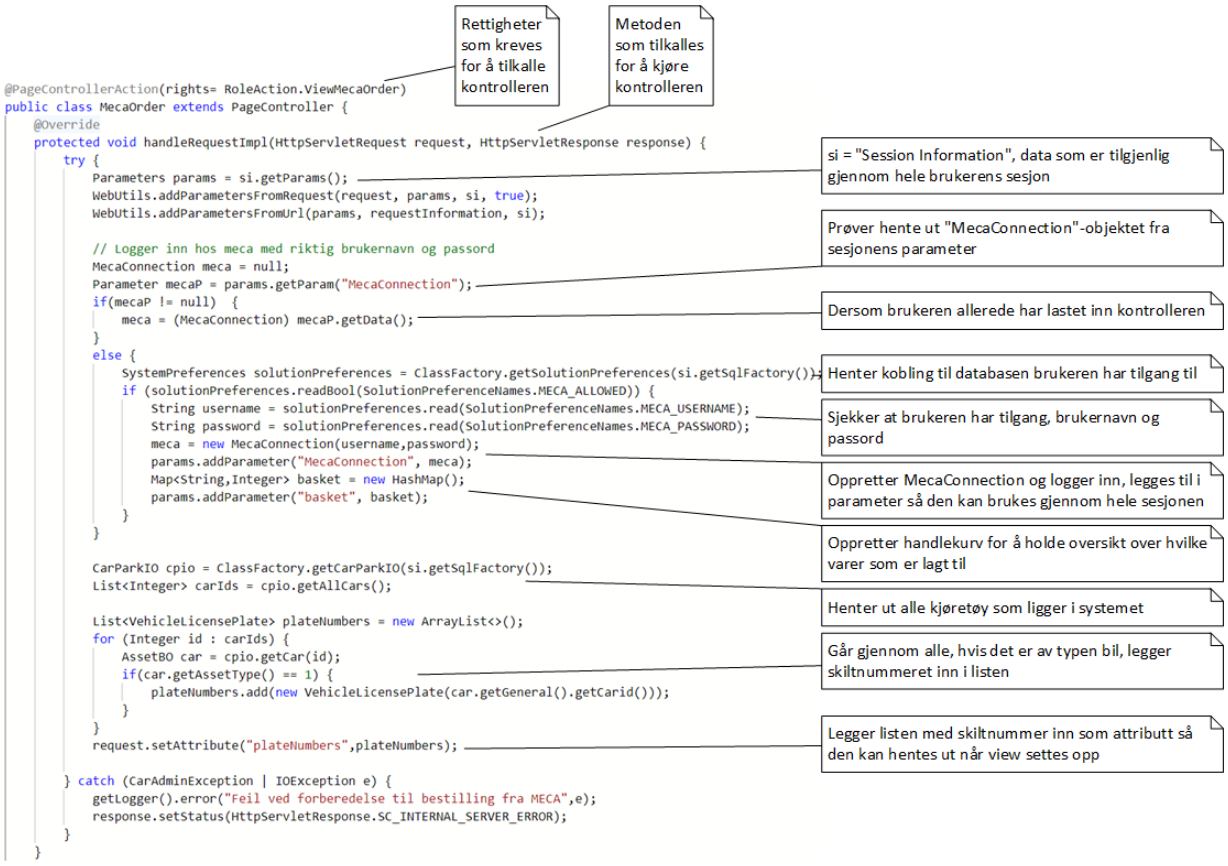
6.7.6 MecaConnection.java

En av funksjonene blir tilkalt utfra et valg brukeren gjorde. Oppgaven til *MecaConnection* er å utføre dette valget i MECA sitt system i form av et HTTP POST/GET kall, dette resulterer i et svar fra MECA i form av et HTML-dokument. Dette dokumentet gås gjennom og dataen plasseres inn i Java-objekter, Java-lister og Java-maps slik at det blir enklere å håndtere dem. Disse dataene sendes tilbake til *MecaOrderSteps* som legger denne dataen inn i HTML-elementer, slik at det kan utføres en oppdatering av viewet.

6.7.7 Detaljert gjennomgang av en funksjonalitet

Her blir det beskrevet i detalj hvilke klasser og funksjoner som er involvert når brukeren laster inn *MecaOrder* og velger seg en bil, som videre skal brukes under bestilling.

I figur 31 beskrives hvordan *PageControlleren* fungerer. Denne lastes inn når brukeren velger å gå til menyelement *MecaOrder*.

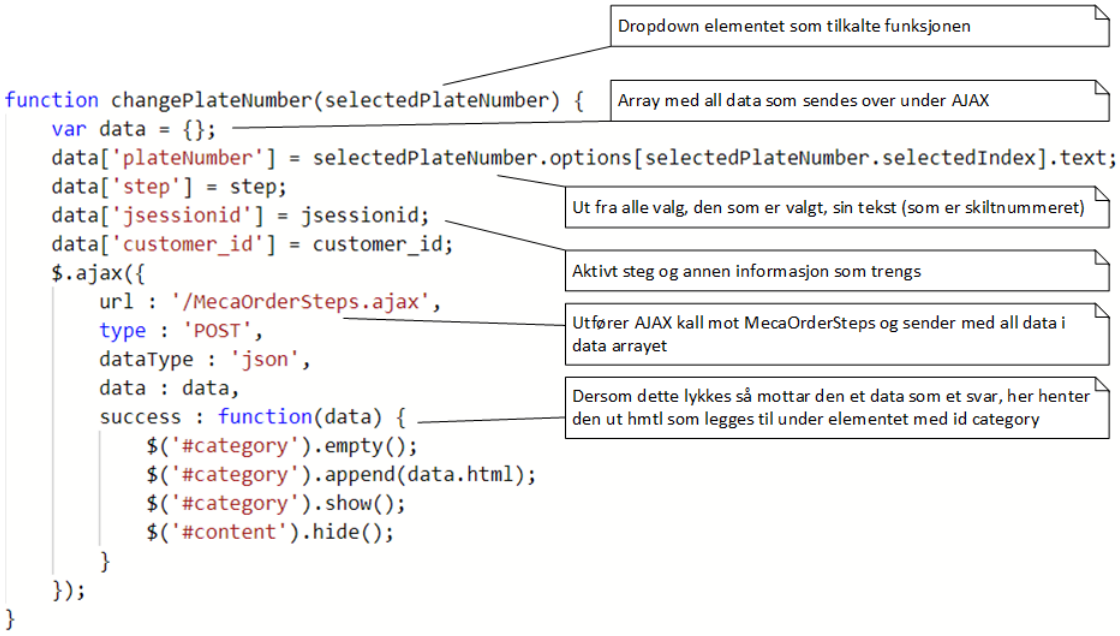


Figur 31: MecaOrder Controller

I figur 32 beskrives hvordan utseende til *MecaOrder* settes opp, dette ligger i filen *mecaOrder.jsp* og tilkalles etter *controller* er ferdig med å kjøre.

Figur 32: MecaOrder View

I figur 33 beskrives JavaScript-funksjonen som tilkalles når brukeren har valgt et av skiltnummerene.



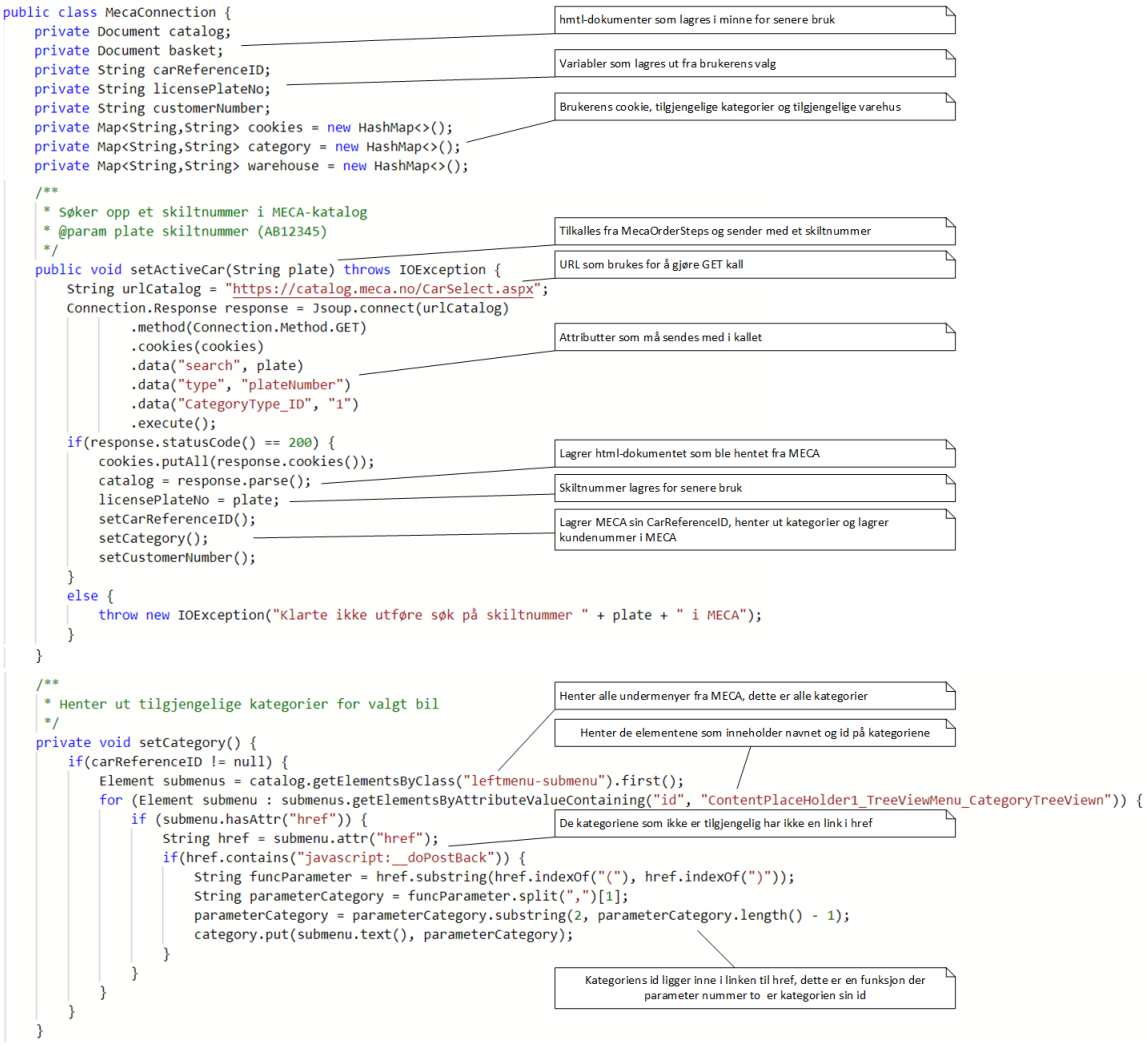
Figur 33: MecaOrder JavaScript

Figur 34 på neste side beskrives hvordan AJAX-klassen tilkaller *MecaConnection* for å utføre en handling mot MECA. Dette resulterer i et svar med data om hvilke kategorier som er tilgjengelig, og med disse settes det opp HTML-elementer som sendes tilbake for å oppdatere utseende ute hos brukeren.


```
@AjaxClass
public class MecaOrderSteps extends AjaxLogic  {

    @Override
    public void doLogic(SolutionSessionInfo si, HttpServletRequest request, HttpServletResponse response) throws IOException {
        try {
            Messages m = si.getMsg();
            JSONObject json = new JSONObject();
            int step = WebUtils.getIntOrDefault(request, null, "step");
            MecaConnection meca = (MecaConnection) si.getParams().getParam("MecaConnection").getData();
            Map<String,Integer> basket = (Map<String, Integer>) si.getParams().getParam("basket").getData();

            if (step == 1) {
                String plateNumber = WebUtils.getStringOrDefault(request, null, "plateNumber");
                String category = WebUtils.getStringOrDefault(request, null, "category");
                String article = WebUtils.getStringOrDefault(request, null, "article");
                if(plateNumber != null) {
                    json.put("html", categories(meca, plateNumber, m));
                }
                else if(category != null) {
                    json.put("html", submenu(meca, category, basket, m));
                }
                else if(article != null) {
                    int quantity = Integer.parseInt(WebUtils.getStringOrDefault(request, null, "quantity"));
                    json.put("quantity", changeOrder(article, quantity, basket));
                    json.put("article", article);
                }
            }
            else if (step == 2) {
                String warehouse = WebUtils.getStringOrDefault(request, null,"warehouse");
                String articleId = WebUtils.getStringOrDefault(request, null, "articleId");
                if (warehouse != null) {
                    List<String> inStock = inStock(meca, basket, warehouse);
                    List<String> noStock = noStock(basket, inStock);
                    json.put("inStock", inStock);
                    json.put("noStock", noStock);
                }
                else if(articleId != null) {
                    json.put("html", removeArticle(meca, basket, articleId, m));
                }
                else {
                    json.put("html", getBasket(meca, basket, m));
                }
            }
            else if (step == 3){
                String warehouse = WebUtils.getStringOrDefault(request, null, "warehouse");
                json.put("html", order(meca,warehouse));
            }
            json.write(response.getWriter());
            response.getWriter().flush();
        }
        catch (JSONException | IOException e) {
            log.error("Noe gikk feil under bytte til neste steg i MecaOrder", e);
        }
    }

    /**
     * Oppsett av kategorier
     * @return html
     */
    private String categories(MecaConnection meca, String plate, Messages m) throws IOException {
        String html;
        meca.setActiveCar(plate);
        Map<String,String> categories = meca.getAvailableCategory();

        html =  "<td> " + m.t("part.category") + " </td>" +
                "<td>" +
                "<select style=\"width: 100%;\" class=\"etc_input etc_input100\" size=\"1\" id=\"categories\" onchange=\"changeCategory(this)\">" +
                "<option SELECTED value=\"\" > " + m.t("Choose") + " </option>";
        for (Map.Entry<String,String> category : categories.entrySet()) {
            html += "<option value=" + category.getValue() + " > " + category.getKey() + " </option>";
        }
        html += "</select>" +
                "</td>";

        return html;
    }
```

Figur 34: MecaOrder Steps

I figur 35 beskrives hvordan data hentes ut fra HTML-elementer som kommer fra MECA og hvordan dette plasseres i Java-objekter.



Figur 35: MecaConnection

6.8 Lagerregister

Hensikten med lagerregisteret er å holde oversikt over de ulike lagrene til en kunde. Lagerregisteret er implementert med dataene: *navn*, *beskrivelse*, *kontaktperson*, *telefon*, *mail*, *adresse* med by og postnummer, om lageret er *eksternt* og hvilket *type* lager det er (delelager eller dekkhotell). Lagerbeholdning og dekkbeholdning er knyttet opp til lagerregister for å vise hvilket lager varene eller dekkene ligger på. I lagerregisteret er det mulig å legge til nye lagre ved hjelp av en dialog. Unikt for denne dialogen er at det spesifiseres om lageret er eksternt eller ikke ved hjelp av en «bool»-løsning. Det spesifiseres også om lageret er et delelager eller dekkhotell. Dette løses ved å bruke en «enum». Det er også mulig å redigere, slette og filtrere data.

6.9 Lagerbevegelse

I utgangspunktet ble det påbegynt en ordreoversikt. Etter møte med ETC fremkom det at de hadde opprettet en egen ordreoversikt og at dette derfor skulle nedprioriteres. Ordreoversikten ble derfor omgjort til en lagerbevegelse. Lagerbevegelsen er laget for å gi en rapport på når ulike deler er blitt tatt ut av eller lagt inn på lager. Hensikten med å ha en slik rapport er å dokumentere de ulike bevegelsene som skjer på et lager. Dette gir en oversikt over hvilke deler det går mye av og visa versa. Det vil også være lettere å spore feil som har oppstått ved at datoen på når en vare er tatt ut eller lagt inn, er dokumentert. Lagerbevegelsen holder på informasjon om en del er tatt ut, lagt inn, datoen bevegelsen fant sted, en kommentar, navn på delen og prisen på delene sammenlagt. Det var planlagt at dekk skulle inngå i lagerbevegelsen, men dette ble ikke implementert grunnet tidsmessige årsaker. Lagerbevegelsen leser data fra tabellen *storageinventorystock* som holder på informasjon om alt som går inn og ut av lager. Kolonnen *partId* fungerer som en «foreign key», som brukes for å hente navn og pris på varen. *Storageinventorystock* oppdateres når varer registreres, og denne skulle også oppdateres når en vare ble tatt ut ved registrering av utført vedlikehold. Sistnevnte funksjonalitet

ble ikke ferdigstilt, grunnet mangel på tid.

6.10 Dekkbeholdning

Dekkbeholdning er implementert med dataene *dekktype*, *merke*, *dato for innkjøp*, *modell*, *bredde*, *profil*, *diameter*, *hastighetsindeks*, *lagerhylle* og hvilket *lager* det ligger på (se figur 9 fra kapittel 4). Dekktype indikerer om det er bildekk, traktorhjul, buss, gravemaskin, gressklipper eller el-sykkel. Dekktyper for bil er sommer, vinter og vinter med pigg. ETC ønsket at dekkbeholdningen skulle ha støtte for alle kjøretøy med hjul som er registrert i CarAdmin. Derfor ble disse dekktypene lagt inn.

Hastighetsindeks er et viktig begrep når det gjelder dekk. Hastighetsindeksen indikerer hvor høy hastighet et dekk maksimalt tåler. Det er forbudt å benytte dekk på et kjøretøy dersom de ikke tåler minst like høy hastighet som kjøretøyets maksimalhastighet.

6.11 Lagerbeholdning

Her finnes det oversikt over alle varer som er inne på lager, med unntak fra dekk. Disse varene består av følgende data: *navn*, *beskrivelse*, *pris*, *beholdning*, *enhet*, *kategori* og hvilket *lager* varen ligger på. Dette gir en god oversikt over hva som er tilgjengelig på lager. Det er mulig å søke utfra navnet til en vare eller filtrere varer som tilhører en av de 13 tilgjengelige kategoriene.

Det finnes to forskjellige måter for å legge til nye varer. Første måte er ved å utføre en bestilling fra menyelementet *MecaOrder*. Her hentes all informasjon direkte fra MECA sin nettside og varene legges inn automatisk i det en bestilling blir utført. Den andre måten er inne i lagerbeholdningen der det er satt opp en dialog. Her fylles all informasjon inn manuelt og dette kan brukes for å legge til varer som ikke er bestilt fra MECA.

7 Testing og Kvalitetssikring

I dette kapittelet beskrives metoder for hvordan lagermodulen ble testet og kvatlietssikret.

7.1 Manuell testing

Det har blitt testet fortløpende gjennom hele utviklingen. Det er blitt testet manuelt at all utviklet kode fungerer som tenkt og ikke inneholder bugs. Det gjenstår to bugs som har vært vanskelige å rette opp i og det er blitt brukt mye tid på å prøve å finne ut av årsaken uten å ha lykkes. Disse var søkefeltet i *lagerregister* og «handling»-knappen i *lagerbevegelse*. De ble derfor nedprioritert for å ferdigstille andre oppgaver.

Det ble testet grundig på slutten av inkrementene for å sikre at funksjonaliteten fungerte, og at den kunne anses som ferdigstilt. Gruppen har jobbet på hver sin «branch». Fordelen med dette i forhold til testing er at det kan testes på egen «branch» før man «merger» sammen med «masterbranchen». Testingen innebar plotting av ulike data, både gyldige og ugyldige data. Det har blitt testet at de forskjellige funksjonalitetene samsvarer med hverandre.

7.2 Testing av lagerbeholdning

Det har blitt testet at ved registrering av en vare kan det kun legges inn gyldige verdier i dialogen. Det er sjekket at databasen samsvarer med programvaren. Dette innebærer at ved registrering har ny data havnet i databasen, ved sletting har den forsvunnet og ved redigering har den endret seg. Det er blitt testet at alle knapper fungerer og resulterer i forventet resultat. Det er testet at alt av språk er riktig i forbindelse med internasjonaliseringen. Det er blitt testet at filtreringen fungerer og at riktig data vises i rapporten utifra filtreringen. Ved søk av data er det sjekket at dette stemmer overens med søkeordet. Ved å klikke på tannhjulet i rapporten,

vises detaljert informasjon om den valgte varen. Det ble sjekket at detaljene om denne stemmer overens med raden for varen i rapporten.

7.3 Unit testing

Gjennom utviklingsperioden ble det diskutert hvorvidt «unit testing» ville vært hensiktsmessig å implementere. Ved møte med ETC fremkom det at de hadde lite «unit testing» i sin egen kode, og at det derfor ville være unødvendig å implementere i lagermodulen. Årsaken er at det er en svært tidskrevende prosess å utvikle testene. I et system som CarAdmin er det ingen stor gevinst å vinne på å utvikle unit tester, da det vil tidsmessig vil være mer effektivt å teste manuelt. En fordel ved å ha implementert «unit testing» er det kunne blitt testet oftere, men denne fordelen anses ikke som stor nok til at det ville vært hensiktsmessig å implementere i dette prosjektet.

7.4 Alfa- og beta testing

Gjennom prosjektet har all kode gått gjennom en alfa og en beta test. Alfa testing har bestått av å teste kode på egen «branch» før koden «merges» sammen med «masterbranchen». Beta testing har bestått av å teste «masterbranchen» etter oppdateringer i koden. Dette kan betraktes som en integrasjonstest. Fordelen med å gjøre det på denne måten er at man får testet at det ikke introduseres nye bugs ved «merging» med «masterbranchen».

7.5 Rapporten

For å sikre god kvalitet på rapporten ble det utført korrekturlesing både av gruppen, men også av venner og familie. Det ble også utført en validerings-test i «NTNU Grafisk senter»[17], som sjekket kvaliteten på figurer, riktig font osv. Gruppen gikk manuelt over for å sjekke at alle figurer var leselig ved utskrift med kvaliteten 600dpi.

8 Avslutning

Dette kapittelet omhandler hvordan arbeidsprosessen utartet seg, hvilke valg som ble tatt, måloppnåelse, muligheter for videre arbeid, kritikk av oppgaven, evaluering av arbeidet og en konklusjon for hele bacheloroppgaven.

8.1 Diskusjon

8.1.1 Endringer og valg underveis

Et viktig punkt i bacheloroppgaven var å kunne bestille deler fra tredjepartsleverandører via API. I starten av prosjektet, som vist i Gantt-diagrammet i vedlegg F, ble det satt av mye tid til å kontakte potensielle leverandører. Dette ble gjort parallelt med andre gjøremål. Det viste seg at det ikke ville bli enkelt å finne en leverandør som hadde API for bestilling. Etterhvert som flere og flere leverandører ble kontaktet, var det en leverandør som sa de hadde API for bestilling. Denne leverandøren var Veng[18]. Selv om de hadde API, viste det seg å være vanskelig å opprettholde en dialog med de. De sa tidlig at de skulle se på muligheten for å opprette en testbruker til systemet deres, slik at det ville være mulig for gruppen å koble seg opp mot API-et. Så gikk tiden og de svarte sjeldent på mail og tok ikke telefonen. Da de først tok telefonen kunne de ikke gi noe svar på hva status var. Ca. 2 uker etter de ga beskjed om at de hadde API for bestilling, ringte de og sa det ikke ville la seg løse allikevel, til gruppens store fortvilelse.

Da det ble klart at bestilling via API var helt utelukket, ble det bestemt å gjøre noe tungvint, og ikke helt optimalt. Allikevel ble det en midlertidig løsning av det selv om arbeidet var tidkrevende. Dette var en beslutning tatt i felleskap med ETC[2]. Løsningen ble såkalt ≪Web-crawling≫ mot MECA sin nettbutikk. Dette handler om å navigere via HTML-koden til en nettside for å hente ut informasjon. Dette fungerer, men ulempen er at det muligens ikke vil virke lenger om nettsiden oppgraderes med nytt design. Det er fordi mye av det som hentes ut er hardkodet utifra klassenavn i HTML-elementer. MECA ble valgt fordi nettbutikken var

oversiktlig og kontaktpersonene i MECA var positivt innstilt på å hjelpe gruppen med å lykkes. Gruppen fikk en testbruker til en demoside slik at bestillinger kunne testes. I tillegg hadde de planer om å lage et API for bestilling. Dette API-et ville ikke bli klart til å brukes i forbindelse med bacheloroppgaven, men det kunne bli aktuelt å gå over til å bruke dette senere.

I oppgavebeskrivelsen ble det spesifisert at det skulle opprettes en oversikt over interne ordre. Det vil si en oversikt over hva en mekaniker trenger for å utføre et verkstedoppdrag. På den måten kunne lageransvarlig håndtere denne og oppdatere lagerbeholdningen deretter. Underveis i prosjektperioden ble denne biten kuttet ut. Det var fordi ETC allerede hadde et eksisterende ordresystem med egne standarder for bilag. Arbeidet som krevdes for å koble dette sammen med lagermodulen ville bli for omfattende. I tillegg var ikke all koden for dette ordresystemet gjort tilgjengelig i utviklingsmiljøet til gruppen. Det ble bestemt å lage et menyelement for lagerbevegelse i stedet. Denne skulle gi en oversikt over hvilke deler som ble brukt etter registrering av utført vedlikehold eller oppdatering av lagerbeholdning.

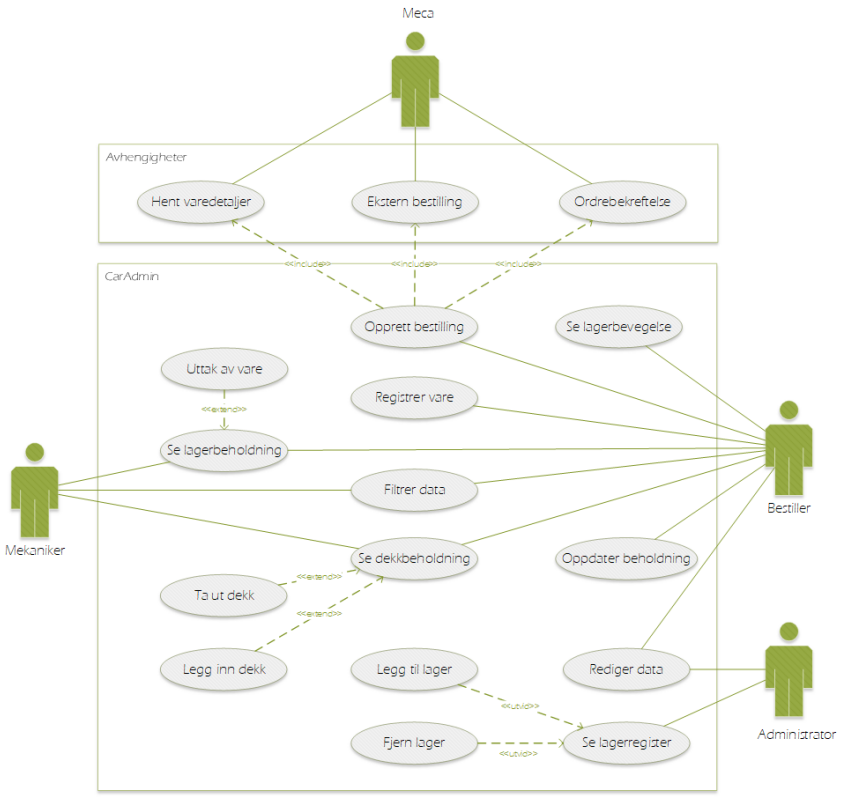
8.1.2 Måloppnåelse

I kapittel 1.3 ble prosjektmålene omtalt. De ble delt inn i resultatmål, effektmål og læringsmål. Alle ble ikke oppnådd, men mange ble det. Det er viktig å påpeke at disse ble definert før gruppen fant ut at ingen tredjepartsleverandører hadde API.

Når det gjelder resultatmålene ble det viktigste oppnådd. Det er en velfungerende lagerbeholdning som viser hva en bedrift innehar av ulike deler, antall og hvilke lagre de befinner seg på. Det kan legges til helt nye deler eller ved påfyll kan antallet oppdateres enkelt. Lagerbeholdningen kan også brukes til varetelling. gruppen rakk ikke å implementere utskrift av telleliste, men en alternativ løsning er å skrive ut hele nettsiden. Det ble ikke implementert QR-koder til delene, da gruppen ikke fikk nok tid til dette. Lagerbeholdningen skulle oppdateres når det skjedde et uttak av deler ved registrering av utført vedlikehold, ved at saldoen trekkes fra når

en del tas ut. Mye ble gjort, men funksjonaliteten ble ikke ferdig utviklet. Derfor er ikke dette på plass i løsningen.
Lagerimport gjøres ved å registrere nye varer/dekk manuelt. Ved bestilling hentes varer ut ifra spesifisert skiltnummer og kategori. Det er ikke utviklet en løsning for importering ved å lese data fra fil. Det sto ikke spesifisert i oppgavebeskrivelsen, men i enighet med ETC ble det laget en egen oversikt over dekkbeholdning. Dekk har helt andre egenskaper enn bildeler og det var da naturlig å skille dette fra lagerbeholdningen. Dette førte til at et eget menyelement for dekkbeholdning ble opprettet. Å utvikle dette tok nesten like lang tid som å utvikle lagerbeholdningen, ettersom en del data måtte presenteres på en litt annerledes måte. Dekkbeholdningen støtter mer enn bare bildekk, den støtter alle kjøretøytypene med hjul registrert i CarAdmin. Dekk er nå koblet opp mot produkt i form av at det registreres hvilket type kjøretøy dekket passer til. Det bør også implementeres nøyaktig hvilke kjøretøy dekkene har vært registrert på. Dette finnes ikke i dagens løsning, men vil være naturlig å utvikle.

Ordreoversikten gikk over til å bli et menyelement for lagerbevegelse. Denne oppdateres ettersom lagerbeholdningen endres. Bestilling fra tredjepartsleverandøren MECA fungerer til tross for en komplisert løsning. Det ble ikke implementert en løsning for retur av bestilling gjennom CarAdmin, årsaken til dette er at dette er mangel på tid, mye grunnet mangel på API.
Menyelementet *Lageroversikt* som skulle inneholde oversikt over lagermodulen var ikke et resultatmål, men etterhvert ble gruppen enige sammen med ETC om at et søkefelt som søker gjennom hele lagermodulen var ønskelig. Dette ble påbegynt, men ikke fullført på grunn av mangel på tid.
Følgende Use Case diagram i figur 36 viser hvilke funksjonaliteter som ble implementert i løpet av utviklingsperioden.



Figur 36: Use Case for hvordan det ble

Om effektmålene ble oppnådd kan det ikke svares på, men gruppen tror de vil oppnås hvis lagermodulen blir tatt i bruk.
Når det gjelder læringsmål, er gruppen fornøyd med å ha oppnådd disse. Etter bacheloroppgaven har alle gruppemedlemmene fått erfaring med planlegging og utvikling av en større programvare, fått erfaring med ny teknologi og ikke minst erfaring med prosjektarbeid.

8.1.3 Resultat av risikovurdering

Scenario 4, ≪API eksisterer ikke hos tredjepartsaktører≫, i tabell 1 inntraff. Dette resulterte i mye ekstra arbeid for gruppen. Det var tvil rundt om det ville være mulig å implementere bestilling opp mot tredjepartsleverandører. Men etter mye

arbeid, ble det utviklet en løsning for bestilling fra MECA ved hjelp av «web-crawling». Selv om risikoen var høy ble det iverksatt tiltak som endte i et positivt resultat. Scenario 5, «Prosjektet rekker ikke å ferdigstilles», i tabell 1 inntraff delvis. Store deler av prosjektet ble ferdigstilt, men noe funksjonalitet mangler. Scenario 6, «At det skjer endringer underveis i arbeidet som krever at det trengs endring i kravspesifikasjonen», i tabell 1 inntraff både ved at at API ikke var tilgjengelig og at ordre ikke skulle utvikles av gruppen allikevel. Selv om det har skjedd endringer i kravspesifikasjonen har det blitt tilpasninger underveis.

8.2 Kritikk av Oppgaven

Ut ifra oppgavebeskrivelsen omtalt i kapittel 1.1.3 virket det veldig klart hva som skulle lages. Punktet som omhandlet bestilling fra tredjepartsleverandør via API, viste seg å ikke være mulig. Det gikk mye tid på å finne ut av dette og denne tiden kunne blitt utnyttet bedre hvis det hadde blitt avklart før. Mot slutten av utviklingsperioden da gruppen hadde startet på en mulig ordreoversikt, ble det ikke som ETC hadde sett for seg. De hadde sett for seg at lagermodulen var koblet opp mot produktregisteret i verkstedboka, men det ble diskutert og avklart tidlig i prosjektperioden at gruppen skulle utvikle noe eget i stedet for å jobbe ut ifra produktregisteret. Dette skapte litt forvirring og endring i veien videre.

8.3 Videre Arbeid

Brukergruppene som er omtalt i kapittel 3.1 er et eksempel på hvordan rettigheter kan fordeles mellom *Bestiller*, *Mekaniker* og *Administrator*. Det ble enighet med ETC at disse ikke var nødvendig å implementere. Her ble det fortsatt satt opp forskjellige rettigheter som kreves for hvert av produktene, men det ble ikke lagt noe fokus på å fordele disse rettighetene til forskjellige brukergrupper. Dette fordi brukergrupper skreddersys i samarbeid med kunden som tar i bruk løsningen.

Koblingen opp mot MECA for å utføre en bestilling er satt opp som en egen klasse. Her vil det være naturlig å opprette et grensesnitt som definerer noen fastsatte funksjoner som brukes for å utføre en bestilling. Dette for å videre kunne lage nye klasser som utfører bestillinger ut mot andre leverandører. Grunnen til at dette ikke ble implementert i lagermodulen skyldes mangel på tid mot slutten av prosjektet, men klassen *MecaConnection* er et godt utgangspunkt for et fremtidig grensesnitt.

Den manglende funksjonaliteten som nevnt i kapittel 6.9 bør implementeres ved videre arbeid. I forhold til lagerbeholdningen bør det implementeres en telleliste (skrive ut rapport) og det kan utvikles QR-koder til varene. Det vil være naturlig å implementere en løsning for å importere data ved lesing fra fil. Da det ikke ble tid til å implementere en løsning for retur av bestillinger i prosjektperioden bør dette gjøres ved videre arbeid.

8.4 Evaluering av arbeidet

8.4.1 Samarbeid

Samarbeidet mellom gruppemedlemmene har fungert bra. Gruppen har tidligere samarbeidet ved andre prosjekter i forbindelse med dataingeniørstudiet så alle kjente hverandre fra før. Gjennom hele prosjektperioden har gruppen vært flinke til å fordele oppgaver og samtidig hjelpe hverandre. Gruppen har sittet sammen og jobbet i kjernetiden, med det har også blitt noe jobbing på egenhånd på kveldstid. Som tidligere nevnt i kapittel 1.5 så har gruppemedlemmene forskjellige spisskompetanser. Dette ble utnyttet ved fordeling av arbeidsoppgaver. Bjørn Inge har jobbet med den vanskeligste funksjonaliteten, mens Andreas og Marius har jobbet mer med det grunnleggende i lagermodulen.

Alle gruppemedlemmene respekterte gruppereglene (se vedlegg B) og overholdt de. Kravet om arbeidstid har ikke alltid blitt innfridd, men har ofte vært i nærheten av avtalt tid. Noen av årsakene til dette er jobbsøking og arbeid med em-

net TØL1011-Ingeniørfaglig Systememne. Det må nevnes at noen uker jobbet gruppemedlemmene lengre enn forventet.

Etter påske ble rapporten satt mer i fokus etter en intens utviklingsperiode. I følge planen skulle utviklingen være ferdig 1. mai, men dette gikk ikke helt i mål. For å komme i mål fordelte gruppen arbeidet på den måten at Andreas og Marius jobbet med rapporten, mens Bjørn Inge jobbet med å få bestilling fra tredjepartsleverandør til å fungere. Det må nevnes at denne fordelingen ikke var absolutt. Andreas og Marius utviklet innimellom, og Bjørn Inge skrev rapport av og til. Uansett sørget gruppen for å gjøre hverandre gode for å sikre rask og god progresjon frem mot innleveringsfristen.

8.4.2 Prosjekt som arbeidsform

Gjennom prosjektperioden har gruppen opparbeidet seg nyttig erfaring med prosjektarbeid. Det var noe nytt og lærerikt ved å jobbe etter en systemutviklingsmodell i praksis. Det å kunne ta i bruk teori fra studiet var veldig givende.
Valget av systemutviklingsmodell viste seg å være klokt. Etter prosjektperioden har gruppen produsert noe som kan brukes videre. Det var et mål fra start av. I planleggingsfasen ble det bestemt å dele prosjektet inn i tre inkrementer. Planen fungerte gjennom det første inkrementet, men da situasjonen med bestilling fra tredjepartsleverandør endret seg drastisk, måtte gruppen prioritere annerledes.

Hadde fossefall vært den valgte modellen ville det blitt langt vanskeligere å justere veien videre enn ved inkrementell-sekvensiell. Planen var i utgangspunktet å følge den plandrevne versjonen av utviklingsmodellen, men ut ifra situasjonen måtte gruppen redefinere inkrementene og jobbe mer smidig. Derfor ble den smidige versjonen av utviklingsmodellen fulgt mot slutten av utviklingen. Møtene med ETC var grunnlaget til valgene som ble tatt videre.

Til tross for en mer ustrukturert innspurt enn planlagt, kom gruppen i mål med

det viktigste i lagermodulen. Et Gantt-diagram for hvordan hele prosjektperioden utartet seg kan leses i vedlegg G.

8.5 Konklusjon

Arbeidet med bacheloroppgaven resulterte i stort læringsutbytte for alle gruppemedlemmene. Muligheten til å jobbe med et reelt prosjekt og anvende en systemutviklingsmodell var lærerikt. I tillegg til dette fikk gruppen erfaring med ny teknologi og prosjekt som arbeidsform.
God planlegging og arbeidsmetodikk resulterte i at bacheloroppgaven ble en suksess. Det meste av funksjonalitet ble implementert og dette kan bygges videre på. Den alternative løsningen i stedet for API ordnet seg, og ved generalisering av denne kan den brukes fram til eventuelle API-er blir tilgjengelig.
Kort oppsummert så kan lagermodulen etterhvert bli en del av CarAdmin som bidrar til å bygge på ETC sin visjon: ≪Det Gode Bilhold!≫.

Referanser

[1] E. T. C. AS. (2019). CarAdmin, side: http://www.caradmin.no (sjekket 10.01.2019).

[2] ——, (2007). Electric Time Car, side: http://www.electrictimecar.com (sjekket 10.01.2019).

[3] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns: Elements of Reusable Object-OrientedSoftware*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA ©1995, 1995.

[4] (2019). JSP Tutorial, side: https://www.tutorialspoint.com/jsp/ (sjekket 24.04.2019).

[5] M. N. AS. (2019). MECA, side: https://www.meca.no/ (sjekket 28.01.2019).

[6] (2011). Trello, side: https://trello.com/ (sjekket 26.04.2019).

[7] T. A. S. Foundation, *Apache Tomcat*, versjon 9.0.14. side: http://tomcat.apache.org/.

[8] M. Foundation, *MariaDB*, versjon 10.3. side: https://mariadb.org/.

[9] JetBrains, *IntelliJ IDEA*, versjon 2018.3.3. side: https://www.jetbrains.com/idea/.

[10] G. Inc., *Gradle*, versjon 5.1.1. side: https://gradle.org/.

[11] T. P. L. Authors, *Lombok*, versjon v0.23-2018.3. side: https://projectlombok.org/.

[12] I. Sommerville, *Software Engineering Tenth Edition*. Pearson, 2016, s. 49–51.

[13] (2019). Overleaf, side: https://www.overleaf.com/ (sjekket 25.04.2019).

[14] (2019). Git, side: https://git-scm.com/ (sjekket 15.02.2019).

[15] (). ResourceBundle, side: https://docs.oracle.com/javase/7/docs/api/java/util/ResourceBundle.html (sjekket 19.05.2019).

[16] (2002). HeidiSQL, side: https://www.heidisql.com/ (sjekket 01.05.2019).

[17] (). NTNU Grafisk senter, side: https://www.ntnu.no/grafisksenter/ansatt_student (sjekket 19.05.2019).

[18] (2019). Veng, side: https://www.veng.no/ (sjekket 30.04.2019).

Vedlegg

A Forprosjekt

Forprosjektrapport - Lagermodul

Andreas Maurud Prøven, Bjørn Inge Kulsveen, Marius Dahlen Østli

Januar 2019

Innhold

1 Mål og Rammer

1.1 Bakgrunn

Electric Time Car AS[1] heretter kalt ETC er en leverandør av softwareløsninger for kjøretøypooler, ressursstyring og skreddersydde utviklingsprosjekter for kunder. CarAdmin[2] er ETC sitt hovedprodukt, som benyttes av både store og små aktører. Dette brukes til oppfølging av kjøretøyer et firma har. Ved hjelp av kjøretøyregister, rodekart, verkstedbok og rapporter om skader og vedlikehold, holdes det god oversikt over selskapets bilpark. Mange av disse selskapene har interne verksteder og delelagre. Ved service og reparasjon må deler bestilles hvis de ikke finnes på lager. ETC ønsker at en lagermodul skal utvikles for CarAdmin. Det vil gjøre det enklere for administratorer å holde oversikt over lagerbeholdningen og ordrestatuser.

1.2 Prosjektmål

1.2.1 Effektmål

Ved gjennomføring av dette prosjektet forventer ETC følgende effekt:

- Komplettere dagens løsning for kjøretøyhold for offentlige- og store bedrifter.
- Tilgang til nye kunder.

Med mer funksjonalitet vil CarAdmin tilfredsstille flere ønsker og bli mer aktuell for flere kunder.

1.2.2 Resultatmål

Lagermodulen skal tilføye CarAdmin funksjonaliteter som tillater oversikt over lagerbeholdingen for en bedrifts interne og eksterne lager. Lagerbeholdningen skal oppdateres ettersom deler blir brukt på kjøretøy eller når nye deler kommer inn på lager. Dette skal være tilgjengelig i form av en rapport som også brukes som telleliste ved varetelling. Oversikten over alle bestillinger skal være oppdatert og tilgjengelig via en rapport (ordreoversikt). Dersom det interne lageret mangler en del så kan denne bestilles fra tredjeparts leverandører, som for eksempel MECA[3]. Her skal det også hentes ut data og egenskaper for de forskjellige delene som er på lager gjennom et API.

1.2.3 Læringsmål

Ved gjennomføring av bacheloroppgaven ønsker gruppa å få erfaring med planlegging og utvikling av en større applikasjon. Caradmin er en webløsning, gjennom arbeidet med denne er målet at det skal opparbeides kunnskap om hvordan en webløsning fungerer. Her tilegnes erfaring med Apache Tomcat[4] og AJAX (≪Asynchronous JavaScript And XML≫). CarAdmin bruker MariaDB[5] som databaseløsning og det vil gjennom jobbing med dette, opparbeides mer erfaring innenfor database. Serverkoden skal skrives i Java og det medfører at det tilegnes mer erfaring med språket. I tilegg ønskes det et tett samarbeid mellom gruppemedlemmene for å sikre deling av kunnskap og flyt i arbeidet. Med alt dette vil ferdighetene innen programmering generelt forbedres.

1.3 Rammer

CarAdmin er skrevet i Java og kjører på Apache Tomcat hos ETC. Under utvikling av lagermodulen skal Tomcat kjøres lokalt på egne maskiner slik at endringene som utføres, ikke påvirker den eksisterende løsningen. ETC benytter IntelliJ IDEA[6] som utviklingsmiljø i CarAdmin prosjektet. Gradle[7] og Ant[8] er verktøyene som brukes for å bygge løsningen. Lombok[9] er en plugin som kan installeres i IntelliJ. Med denne trengs ikke egne ≪get-≫ og ≪set-metoder≫ å bli laget for klasser, Lombok gjør det automatisk.

2 Omfang

2.1 Fagområde

[1] Electric Time Car (ETC) er en softwareleverandør som holder til i Gjøvik. De har et bilpark system med navn CarAdmin som hjelper bedrifter, i hovedsak offentlig sektor, med å administrere kjøretøy til sine ansatte. Kommuner er et godt eksempel. De har mange kjøretøy og forskjellige ansatte som benytter dem.
≪CarAdmin er en helhetlig løsning for kjøretøyoppfølging av alle kjøretøy for privat og offentlig sektor. CarAdmin er tilpasset ulik bruk med forskjellige løsninger for alt fra kjøretøypooler til serviceoppfølging av gressklippere. Ønskes det mer automatisert oppfølging kan dette gjøres via GPS som i vår elektroniske kjørebok.≫ ([10])

[1] Fagområde er et samarbeid mellom gruppe 4 og 5 grunnet samme oppdragsgiver og etter råd fra veileder.

Caradmin er delt inn i fire moduler:

- **Kjøretøyregisteret** holder oversikt over alle kjøretøy i Norge. Det inneholder diverse informasjon angående kjøretøyet, f.eks. km-stand, tid for neste service og lokasjon.
- **Kjøreboka** loggfører kjøreturer ved hjelp av GPS. Dette gjør det enkelt å foreta kjøregodtgjørelse.
- **Fellesbil** er utviklet for situasjoner der flere sjåfører deler et kjøretøy. Modulen gir mulighet til å låse nøkler i et nøkkelskap i forbindelse med uttak og innlevering av kjøretøyene. Fellesbil er med på å sikre god ansvarsfordeling mellom sjåførene.
- **Flåte** gir en oversikt over hvilke biler som befinner seg i nærheten, eller nærme et målpunkt. Det gis mulighet for å se kjørerute i kart. Denne modulen brukes i forbindelse med ulike oppdrag, og oppdragsgiver trenger å se hvem som kan ta på seg oppdraget.

2.2 Avgrensning

Mange av kundene har interne verksteder og delelagre, dermed ønsker ETC å utvide dette systemet med en lagermodul. Dette for å gi oversikt over bildeler som er inne på lager og loggføre hva de blir brukt til. Lagermodulen skal holde oversikt over interne lagre og bestillinger som er på vei inn, og gi muligheten til å sjekke beholdningen til eksterne / tredjeparts lagre. All denne informasjonen skal være tilgjengelig for kunden via diverse rapporter der kunden kan sortere og filtrere etter eget ønske.

2.3 Oppgavebeskrivelse

Lagermodulen deles inn i følgende funksjonaliteter:

- **Lagerregister** skal inneholde de lagrene som kunden har (interne/eksterne), dette kan være fra en til mange for en kunde.
- **Lagerbeholdning** med oversikt over alle deler som finnes på alle kundenes lagre. Denne skal inneholde antall av en del og lagerplassering (Reol, hylle og nummer). Informasjonen skal være tilgjenglig via en rapport og kan skrives ut til bruk ved varetelling (telleliste). Hver del har sin egen QR kode som kan skrives ut og plasseres sammen med delen på lager.

- **Lagerimport** Når nye deler skal legges inn på lager så skal det enten gjøres manuelt ved å oppgi all data for produktet, lese data fra fil eller hente data fra tredjeparts leverandørers API.

- **Rettigheter** CarAdmin har allerede tilgangstyring som gir kunder tilgang til tjenester de har betalt for, men sørger også for at kunden kan fordele rettigheter til sine ansatte slik som (bruker, bestiller, administrator). Dette skal også inngå i lagermodulen og eventuelle nye rettighetsgrupper må opprettes.

- **Ordre** Ved uttak av deler fra lager skal det opprettes en ordre slik at bedriften kan utføre intern fakturering og holde lagerbeholdningen oppdatert. Dersom delen ikke finnes på lager så skal det opprettes en bestilling som blir knyttet opp mot en ordre. Ordren skal knyttes opp mot kjøretøyet den skal brukes på.

- **Bestilling** Det skal være mulig å bestille deler inn til lager fra tredjeparts leverandører, enten via en intern ordre eller enkeltvis fra lager for å fylle på lagerbeholdningen. Her skal det opprettes en rekvisisjon ut til leverandør via mail eller API. Alle bestillinger listes opp i en rapport som sier status på bestillingen (opprettet, godkjent, bestilt, mottatt). Alle bestillinger må godkjennes av en bruker med høye nok rettigheter, men dette kan skje automatisk dersom brukeren som bestiller allerede har det.

- **Retur** Det skal være mulig å returnere en ordre/bestilling.

- **Dekkhotell** skal holde oversikt over dekk som er på lager og hvor de befinner seg. Vinterdekk må defineres som enten piggdekk eller piggfritt. Ved innlegging av dekk eller ved dekkskifte skal det registreres dekkslitasje. Det skal noteres maks/min dimensjoner på dekkene. Man skal ha mulighet for å koble opp dekkene til et produkt.

3 Prosjektorganisering

3.1 Ansvarsforhold og roller

Gruppemedlemmer: Gruppa består av tre personer, Andreas Maurud Prøven, Bjørn Inge Kulsveen og Marius Dahlen Østli. Medlemmene studerer på NTNU i Gjøvik sitt dataingeniørstudie. Av de fagene som medlemmene

har hatt i løpet av studiet så vil relevante fag være Systemutvikling, Datamodellering og Databasesystemer, Objekt-Orientert programmering, Programvaredesign og Applikasjonsutvikling. Andreas har i tillegg hatt faget Cloud Technologies som vil være relevant for oppkobling mot API-er.

Prosjektleder: Marius Dahlen Østli er utnevnt til prosjektleder av gruppa. Han har hovedansvaret for å kommunisere med ETC, avtale møter, koordinere arbeidstider og skrive møtereferater.

Veileder: Frode Haug vil være veileder gjennom prosjektet og komme med tilbakemeldinger og statusmøter underveis.

Oppdragsgiver: Dag L Solhaug er daglig leder i ETC og er oppdragsgiver. Hvis det er spørsmål angående oppgaven, skal disse henvendes til han.

Teknisk støtte: Øyvind Flatval er ansatt i ETC og har ansvaret for å gjennomføre en opplæring av systemet deres, slik at gruppa kan komme i gang med utviklingen. Tekniske spørsmål kan rettes til han.

3.2 Rutiner og regler i gruppa

Generelle regler

1. Alle på gruppen forventes å jobbe minst 30 timer i uka ekskludert undervisning, med unntak av de tre ukene i semesteret vi har seminar i emnet TØL1011. Hvert gruppemedlem skal loggføre antall timer med arbeid hver dag i et regneark. Arbeidstid vil hovedsaklig være mellom kl. 08:00 og kl. 16:00. Utover dette kan det ikke forventes at alle er tilgjengelig til enhver tid.

2. Det forventes at alle deltar aktivt.

3. Statusmøte med oppdragsgiver hver fredag kl. 08:00 og med veileder hver mandag kl. 09:00. Alle plikter å møte opp. Det skal skrives møtereferat etter hvert møte og alle på gruppen skal ha tilgang til det.

4. Alle skal gjøre sine oppgaver etter beste evne. Skulle det oppstå problemer tas dette opp i gruppa i god tid.

5. Dersom det oppstår faglig uenighet skal det fattes en flertallsavgjørelse i gruppa.

6. Et gruppemedlem skal informeres i god tid dersom gruppen ikke er fornøyd med arbeidsinnsatsen til vedkommende.

7. Det skal informeres så fort som mulig hvis et gruppemedlem ikke kan møte til avtalt tid grunnet sykdom, eller andre årsaker.

8. Alle gruppemedlemmer skal informere de andre på gruppen om hva som gjøres og holde seg oppdatert på progresjonen i bacheloroppgaven.

9. Det skal noteres hvilke kilder som brukes underveis slik at det ikke blir noe problem med mangler på referanser.

Tiltak ved brudd på regler

1. Dersom problemer oppstår diskuteres dette først internt i gruppa.

2. Vedkommende får en tidsfrist med en skriftlig advarsel til å forbedre seg. I advarselen skal det påpekes regelbruddet, hva som kreves av vedkommende innen tidsfristen og hvilke konsekvenser det vil ha dersom dette ikke overholdes.

3. Hvis gruppa ikke klarer å løse problemet på egenhånd tas det videre til veileder.

4. Hvis det ikke bedrer seg etter samtale med veileder, blir vedkommende skriftlig ekskludert fra gruppa.

4 Planlegging, Oppfølging og Rapportering

4.1 Hovedinndeling av prosjektet

4.1.1 Valg av systemutviklingsmodell

Det skal legges til en ny modul i et eksisterende system. Det medfører at det er en del rammer å forholde seg til. Lagermodulen må lages slik at den passer inn i CarAdmin. Dette gjøres ved å bruke samme kodespråk og kodestandarder. Oppgaven er ganske klar på hva slags funksjonalitet som skal lages, men ikke så klar at man kan forutse nøyaktig hvor lang tid som trengs for å utvikle de ulike delene. Som følger av dette er det valgt inkrementell sekvensiell-modellen[11]. Det ble slik fordi kravene fra ETC er såpass klare, derfor blir det enkelt å definere inkrementene. Under utviklingsprosessen vil et inkrement ferdigstilles før det neste påbegynnes. Fordelen med dette er hvis det skulle oppstå problemer med ferdigstilling av funksjonaliteter, så har gruppa garantert noe å levere til slutt.

Fossefall ble vurdert, men det viste seg å ikke være det beste valget. Grunnen til det er fordi fossefall ikke er åpen for endringer i kravspesifikasjonen underveis. Et usikkert punkt i oppgaven er importeringen av tredjeparts delekataloger, som krever tilgang til API-er og det er ikke garantert at det er mulig. Det må derfor være en mulighet til å kunne gjøre små justeringer hvis det trengs og det er mest kostnadseffektivt i den valgte modellen. Hvis oppgaven hadde vært mer åpen med en uforutsigbar kravspesifikasjon, som ofte er vanlig ved utvikling av ny programvare, ville det vært fornuftig med smidig modell i stedet. Det er den ikke i dette tilfelle, så derfor endte valget på en mer plan-dreven modell.

4.1.2 Inkrementene

For å gjøre utviklingsprosessen effektiv, defineres inkrementene slik at alle på gruppa kan jobbe med forskjellige oppgaver. Det vil si at et inkrement består av tre oppgaver. Det skal fortsatt være fokus på samarbeid, men det åpner for raskere progresjon i utviklingen. Ut ifra kravene på hvilken funksjonalitet ETC vil ha, så er det blitt bestemt at utviklingen skal deles inn i tre inkrementer. De to første inkrementene vil det jobbes med over to perioder på to og ei halv uke. Det siste har det blitt satt av tre uker til. Dette for å kunne finpusse programvaren og eventuelt legge til ekstra funksjoner.

I det første inkrementet skal hovedfunksjonaliteten på plass, slik som lager-

register som inneholder alle kundens lager, lagerbeholdning på disse lagrene og import av delekatalog for å kunne legge deler inn på lager.
Det andre inkrementet tar for seg endringer i lagerbeholdning slik som ordre, bestillinger og oppkobling til tredjeparts leverandørers API slik at bestillinger skal kunne utføres. Her er det lagt inn en selvbestemt frist for oppkobling av API siden dette er kristisk for videre utvikling.
I det tredje og siste inkrementet skal det gjøres mulig å reservere deler etter innkjøp, dvs. Hvis et verksted trenger en del og den ikke finnes på lager, har de førsterett på delen når den kommer inn. Det skal også gjøres mulig å følge lagerbevegelser. Til slutt skal hele modulen internasjonaliseres slik at den har støtte for både norsk og engelsk.

4.2 Plan for statusmøter og beslutningspunkter i perioden

Fredag kl 08:00 Faste møter med oppdragsgiver for å avklare spørsmål og oppdatere status.
Mandag kl 09:00 Faste møter med veileder.
01.02.2019: Frist for innlevering av forprosjektrapport og signering av prosjektavtale.
20.02.2019: Statusrapport 1, tilbakemelding fra veileder på prosjektrapport.
01.03.2019: Dato for ferdigstilling av planlegging.
01.04.2019: Statusrapport 2, statusmøte med veileder ang. progresjon i utvikling.
05.04.2019: Mål for implementering av tredjeparts API.
26.04.2019: Dato for ferdigstilling av programvare.
01.05.2019: Statusrapport 3, tilbakemelding fra veileder på hovedrapport.
20.05.2019: Frist for levering av hovedrapport.
04.06.2019: Muntlig presentasjon av bacheloroppgave.

5 Organisering av kvalitetssikring

5.1 Dokumentasjon, standardbruk og kildekode

- LaTeX brukes som skriveverktøy av rapport og møtereferat.
- BibTeX brukes for å referere til kilder.
- Git-repository settes opp sammen med ETC og brukes for versjonskontroll under hele prosjektet.

- Microsoft Excel[12] brukes for å føre timer.
- GanttProject[13] brukes til å lage Gantt-diagrammer.
- Serverkode skal skrives i Java og kjøres i Apache Tomcat.
- Klientkode kjøres fra en nettleser og bruker blant annet AJAX.

5.2 Konfigurasjonsstyring

5.2.1 Rapport

Rapporten skrives i LaTeX(Overleaf) og jobbes med jevnt gjennom hele prosjektet. Hvert gruppemedlem er selv ansvarlig for å legge til kilder som brukes i BibTeX og sørge for at referanser følger oppsettet til harvard-stil. Kristiske valg og systemarkitekturen dokumenteres i rapporten.

5.2.2 Kildekode

Fredag den 25.01.19 ble det faste møte med ETC gjort om til en gjennomgang av CarAdmin. Her ble det satt opp en fork av Git-repositoriet til ETC for å adskille utvikling fra produksjonsmiljøet. Alle tre gruppemedlemmer skal benytte seg av Bitbucket[14] grunnet gratis lisens fra tidligere prosjekter. Det skal brukes Javadoc i all kode som er programmert i Java. I annen programvare skrives generelle kommentarer.

5.2.3 Versjonskontroll

Siden gruppa under utviklingsfasen har valgt å jobbe med tre moduler samtidig så er det viktig at disse utvikles uavhengig av hverandre. Her vil det opprettes «brancher» i Git som tillater uavhengig utvikling frem til modulen er klar til å «merge» sammen med «hovedbranchen».

5.2.4 Testing

Når utviklingen av en modul er ferdig så skal det utføres to tester, alfa- og betatesting. Alfatesing utføres internt i «branchen» for å sikre at modulen fungerer som den skal, og betatesting som sørger for at modulen fungerer etter den er «merget» sammen med «hovedbranch».

5.3 Risikoanalyse

Det er utarbeidet en risikoanalyse, se tabell 1 som tar for seg de risikoene som må tas hensyn til i oppgaven. Den beskriver sannsynlighet, konsekvens og risiko. Sannsynlighet og konsekvens rangeres fra svært lav til svært høy. Risikoen er et produkt av sannsynligheten og konsekvensen. Det er viktig at risikoen tilknyttet de mest kritiske scenariene minimeres, da de kan ha betydelig negativ effekt på produktet. I henhold til dette er det blitt sett nærmere på de to risikoene som anses som mest kritisk og beskrevet tiltak for disse, se tabell 2.

Tabell 1: Risikovurdering

#	**Scenario**	**Sannsynlighet**	**Konsekvens**	**Risiko**
1	En eller flere av medlemmene i gruppen blir syke, eller kan ikke møte opp av en annen grunn	Middels	Middels	Middels
2	Ved et uhell lekkes sensitiv informasjon tilhørende ETC	Svært lav	Svært høy	Middels
3	Ved utvikling av lagermodulen påvirkes negativt det allerede eksisterende system	Lav	Høy	Middels
4	API eksisterer ikke hos tredjeparts aktører	Middels	Høy	Høy
5	Prosjektet rekker ikke å ferdigstilles	Middels	Høy	Middels
6	At det skjer endringer underveis i arbeidet som krever at det trengs endringer i kravsspesifikasjonen	Lav	Middels	Lav
7	Tap av dokumentasjon og/eller arbeid	Svært lav	Svært høy	Middels

Tabell 2: Risikotiltak

Scenario	**Tiltak**
API eksisterer ikke hos tredjeparts aktører	Være tidlig ute med å sjekke hvilke aktører som har egen API og holde ETC oppdatert på denne fremgangen.
Prosjektet rekker ikke å ferdigstilles	Ut ifra tidligere bachelor oppgaver ser vi at mange oppgdrag ikke ferdigstilles i tide. For å unngå dette setter vi opp en god plan som tar høyde for uforutsette hindringer i utviklingen.

6 Plan for gjennomføring

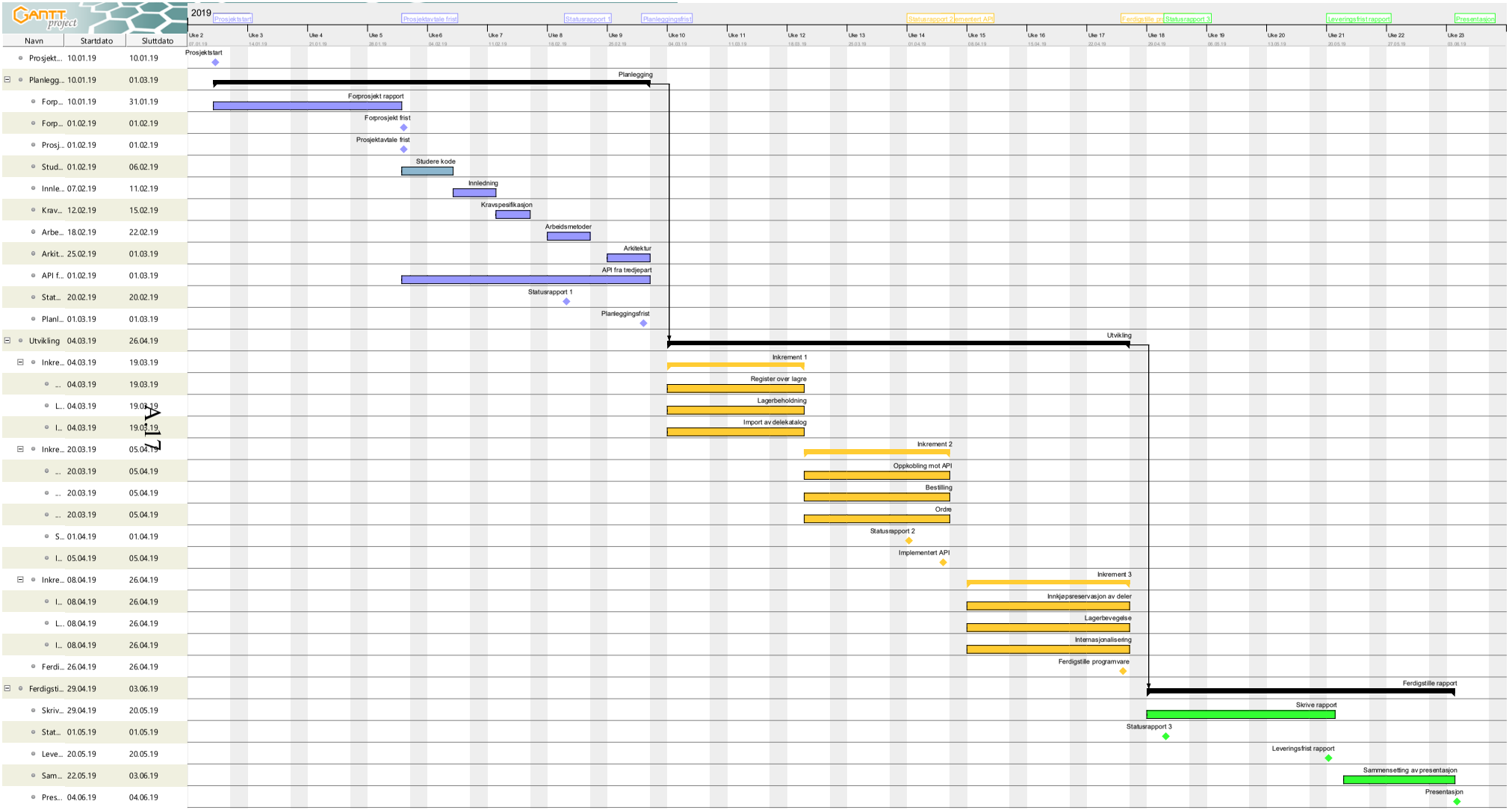
Oppgave

Navn	Startdato	Sluttdato
Prosjektstart	10.01.19	10.01.19
Planlegging	10.01.19	01.03.19
Forprosjekt rapport	10.01.19	31.01.19
Forprosjekt frist	01.02.19	01.02.19
Prosjektavtale frist	01.02.19	01.02.19
Studere kode	01.02.19	06.02.19
Innledning	07.02.19	11.02.19
Kravspesifikasjon	12.02.19	15.02.19
Arbeidsmetoder	18.02.19	22.02.19
Arkitektur	25.02.19	01.03.19
API fra tredjepart	01.02.19	01.03.19
Statusrapport 1	20.02.19	20.02.19
Planleggingsfrist	01.03.19	01.03.19
Utvikling	04.03.19	26.04.19
Inkrement 1	04.03.19	19.03.19
Register over lagre	04.03.19	19.03.19
Lagerbeholdning	04.03.19	19.03.19
Import av delekatalog	04.03.19	19.03.19
Inkrement 2	20.03.19	05.04.19
Oppkobling mot API	20.03.19	05.04.19
Bestilling	20.03.19	05.04.19
Ordre	20.03.19	05.04.19
Statusrapport 2	01.04.19	01.04.19
Implementert API	05.04.19	05.04.19
Inkrement 3	08.04.19	26.04.19
Innkjøpsreservasjon av deler	08.04.19	26.04.19
Lagerbevegelse	08.04.19	26.04.19

Oppgave

Navn	Startdato	Sluttdato
Internasjonalisering	08.04.19	26.04.19
Ferdigstille programvare	26.04.19	26.04.19
Ferdigstille rapport	29.04.19	03.06.19
Skrive rapport	29.04.19	20.05.19
Statusrapport 3	01.05.19	01.05.19
Leveringsfrist rapport	20.05.19	20.05.19
Sammensetting av presentasjon	22.05.19	03.06.19
Presentasjon	04.06.19	04.06.19

Ganttskjema



Gantt-diagrammet over viser planen for utførelse av bacheloroppgaven. Den er delt inn i planlegging, utvikling, rapportskriving og forberedelse til muntlig presentasjon. Den inneholder milepæler som er markert med en diamant. Disse er frister og statusrapporter som skal inn til veileder.
Planleggingsperioden starter med et forprosjekt og etter det er gjort, skal prosjektplanen utarbeides. Antatt arbeidstid for denne er fire uker. Den er delt inn på en slik måte at vi ferdigstiller en og en del, men samtidig vil det fra starten av brukes tid på å kontakte eksterne leverandører angående oppkobling mot deres API. Dette grunnet usikkerhet rundt tilgang til disse.
Utviklingsperioden vil vare i ca. to måneder og brukes til gjennomføring av de tre planlagte inkrementene. Det kan forekomme overlapping mellom utviklingsperioden og hovedrapporten da det er usikkert hvor lang tid utviklingen tar.
Når resultatmålene er oppnådd og utviklingen er ferdig, skal tiden fram til leveringsfristen brukes til rapportskriving. Da skal hovedrapporten ferdigstilles. Den siste tiden fram til den muntlige presentasjonen skal brukes til forberedelse.

Referanser

[1] E. T. C. AS. (2007). Electric Time Car, side: http://www.electrictimecar.com (sjekket 10.01.2019).

[2] ——, (2019). CarAdmin, side: http://www.caradmin.no (sjekket 10.01.2019).

[3] M. N. AS. (2019). MECA, side: https://www.meca.no/ (sjekket 28.01.2019).

[4] T. A. S. Foundation, *Apache Tomcat*, versjon 9.0.14. side: http://tomcat.apache.org/.

[5] M. Foundation, *MariaDB*, versjon 10.3. side: https://mariadb.org/.

[6] JetBrains, *IntelliJ IDEA*, versjon 2018.3.3. side: https://www.jetbrains.com/idea/.

[7] G. Inc., *Gradle*, versjon 5.1.1. side: https://gradle.org/.

[8] T. A. S. Foundation, *Apache Ant*, versjon 1.10.5. side: https://ant.apache.org/.

[9] T. P. L. Authors, *Lombok*, versjon v0.23-2018.3. side: https://projectlombok.org/.

[10] E. T. C. AS. (2019). CarAdmin Produkter, side: http://www.caradmin.no/index.php/en/produkter (sjekket 21.01.2019).

[11] I. Sommerville, *Software Engineering Tenth Edition*. Pearson, 2016.

[12] M. Corporation, *Microsoft Excel*, versjon 1812. side: https://products.office.com/nb-no/excel.

[13] G. Team, *GanttProject*, versjon 2.8.9. side: https://www.ganttproject.biz.

[14] A. C. Plc. (2019). Bitbucket, side: https://bitbucket.org/ (sjekket 22.01.2019).

B Grupperegler

Generelle regler

1. Alle på gruppen forventes å jobbe minst 30 timer i uka ekskludert undervisning, med unntak av de tre ukene i semesteret vi har seminar i emnet TØL1011. Hvert gruppemedlem skal loggføre antall timer med arbeid hver dag i et regneark. Arbeidstid vil hovedsaklig være mellom kl. 08:00 og kl. 16:00. Utover dette kan det ikke forventes at alle er tilgjengelig til enhver tid.

2. Det forventes at alle deltar aktivt.

3. Statusmøte med oppdragsgiver hver fredag kl. 08:00 og med veileder hver mandag kl. 09:00. Alle plikter å møte opp. Det skal skrives møtereferat etter hvert møte og alle på gruppen skal ha tilgang til det.

4. Alle skal gjøre sine oppgaver etter beste evne. Skulle det oppstå problemer tas dette opp i gruppen i god tid.

5. Dersom det oppstår faglig uenighet skal det fattes en flertallsavgjørelse i gruppen.

6. Et gruppemedlem skal informeres i god tid dersom gruppen ikke er fornøyd med arbeidsinnsatsen til vedkommende.

7. Det skal informeres så fort som mulig hvis et gruppemedlem ikke kan møte til avtalt tid grunnet sykdom, eller andre årsaker.

8. Alle gruppemedlemmer skal informere de andre på gruppen om hva som gjøres og holde seg oppdatert på progresjonen i bacheloroppgaven.

9. Det skal noteres hvilke kilder som brukes underveis slik at det ikke blir noe problem med mangler på referanser.

Tiltak ved brudd på regler

1. Dersom problemer oppstår diskuteres dette først internt i gruppen.

2. Vedkommende får en tidsfrist med en skriftlig advarsel til å forbedre seg. I advarselen skal det påpekes regelbruddet, hva som kreves av vedkommende innen tidsfristen og hvilke konsekvenser det vil ha dersom dette ikke overholdes.

3. Hvis gruppen ikke klarer å løse problemet på egenhånd tas det videre til veileder.

4. Hvis det ikke bedrer seg etter samtale med veileder, blir vedkommende skriftlig ekskludert fra gruppen.

Dato: 17.01.2019

Signaturer

Andreas M. Prøven

Andreas Maurud Prøven

Bjørn Inge Kulsveen

Bjørn Inge Kulsveen

Marius Dahlen Østli

Marius Dahlen Østli

C Prosjektavtale

NTNU

Vår dato Vår referanse

Norges teknisk-naturvitenskapelige universitet

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

ETC, Electric Time Car AS

(oppdragsgiver), og

Andreas Naurud Prøven

Bjørn Inge Eriksen

Marius Dahlen Østli (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 10.01.19 til 20.05.19.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

C.3

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Frode Haug

Oppdragsgivers kontaktperson (navn): Dag L. Solhaug

Student(er) (signatur): Bjørn Inge Fabricius dato 25-01-19

Marius Dahlen Østli dato 25-01-19

Andreas Maurud Prøven dato 25-01-19

_____ dato _____

Oppdragsgiver (signatur): Dag L Solhaug dato 25.01.19

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato _____

D Oppgavebeskrivelse

Oppdragsgiver

Oppdragsgiver: ETC, Electric Time Car AS
Kontaktperson: Dag L Solhaug
Adresse: Studieveien 2, 2815 Gjøvik
Telefon: +47 901 01 344
Epost: dag.solhaug@electrictimecar.com

Lagermodul

ETC er en lokal Software leverandør som bla leverer løsninger for Kjøretøypooler og ressursstyring løsninger samt skreddersydde utviklingsprosjekter for kunder.

CarAdmin er en løsning for oppfølging og drift av alle typer kjøretøy for kommuner og bedrifter. Mange av våre kunder har interne verksteder og egne lager av deler. CarAdmin trenger derfor en lagermodul.

Dette er oppgaven for dere som liker å programmere. Ei realistisk oppgave med utfordringer ift å lage ny funksjonalitet til et eksisterende system, lage rapporter, definere og lage API mot eksterne leverandører. Kort sagt ei fullstack oppgave.

Oppgaven

Utvikle en lagermodul som inneholder funksjoner for:

- Lagerbeholdning
- Bestilt / ikke mottatt
- Ordre / ikke levert
- Import av tredjeparts delekatalog
- Innkjøpsreservasjon av deler
- Lagerbevegelse / reservasjoner
- Lagerregister – støtte for flere lagre/fjernlagre internt og eksternt
- API – fjernlager / lagerbeholdningsendring / bestilling fra leverandør…

Programmering

ETC vil bistå prosjektet i gjennomføringen av oppgaven. Jevnlige møter og oppgavefordelinger avtales underveis.

Studentgruppen vil gjennom prosjektet få erfaring med:

- Database - MariaDB
- Server side programmering
- Web/Klientprogrammering
- Java / HTML / Ajax / SQL
- Flerspråklighet

Oppgaven passer best for en gruppe på 2 - 4 personer.

ETC har gjennomført bacheloroppgave ved HiG/NTNU siden 2004 hvor flere av oppgaven har fått topp karakter. Alle bacheloroppgaver har resultert i ansettelser i ETC, eller ansettelser for en eller flere av studentene som følge av referanse fra ETC etter endt bacheloroppgave.

Vi er for tiden på utkikk etter å ansette Java programmerere, lokal tilhørighet vektlegges.

E Statusrapporter

Statusrapport 1

Marius Dahlen Østli, Bjørn Inge Kulsveen, Andreas Maurud Prøven

18. februar 2019

E.2

1 Statuser

1.1 Planlegging

Planleggingen er godt i gang. Oppgaven har i utgangspunktet ganske klare krav og siden vi skal utvide et eksisterende system, er det allerede satt rammer for utviklingen. I forhold til fremdriftsplanen ligger vi på skjema. Verken foran eller etter. På nåværende tidspunkt har vi kommet fram til hvordan lagermodulen skal fungere. Use-Case diagrammet har vi gått igjennom sammen med oppdragsgiver og de var fornøyd med det.
Det som gjenstår i kravspesifikasjonen er domenemodell og sekvensdiagrammer. Før dette kan lages må det studeres mer kode. Vi hadde satt av første uka etter forprosjektet til dette, men vi endte opp med å jobbe mere parallellt med planlegging og kodelesing. Grunnen til dette var å få en bedre oversikt over systemet. Kapittel 1 må utdypes mer grundig med tanke på rapporten på et senere tidspunkt. Vi prioriterer å få på plass en grundig kravspesifikasjon før vi gjør dette.
Utviklingsmodell er valgt og vi har planlagt hvordan vi skal anvende denne under utviklingen. Denne skal beskrives nøyere etterhvert.

1.1.1 Hva er ferdigstilt?

Det er mange punkter vi ikke vil si oss ferdig med enda da mye må skrives nøyere og bedre etterhvert med tanke på rapporten. Vi kan si oss ferdig med Use-Case diagrammet og valg av systemutviklingsmodell.

1.1.2 Hva er under arbeid?

Domenemodell og sekvensdiagrammer er under arbeid. Dette skal inngå i kravspesifikasjonen. For tiden jobber vi fortsatt med å sette oss inn i den eksisterende koden. Det er et stort prosjekt som har blitt utviklet over 10 år. Det vil si at det består av svært mange kodelinjer. Vi føler det er viktig å få en god forståelse på hvordan vi kan legge inn ny funksjonalitet på en best mulig måte. Det finnes allerede massevis av hjelpeklasser/funksjoner/variable.

1.1.3 Tidsfrister

Vi ligger bra an i forhold til Gantt-skjemaet vi laget i januar. Der satte vi en selvbestemt frist for å bli ferdig med planleggingen 01.03.2019 og det ser ut som vi blir ferdig før denne datoen.

1.2 Organisering av gruppens arbeid og ansvarsområder

Vi har kommet godt i gang med arbeidet. Vi bestemte oss tidlig for at vi skal legge inn en skikkelig innsats på bacheloroppgaven og det har ført til at vi er godt igang. Vi sørger for å starte tidlig på morgenen kl 08:15 ved å reservere grupperom. Ved slike reservasjoner må man møte opp for å beholde rommet så det er en fin motivasjon for å komme seg til Campus tidlig. Målet vårt med 30 timer per hode i uka holder vi nesten. Det har hendt i starten at vi har endt opp på litt under 30 timer, så dette må bedres. Det har vært vanskelig å fordele oppgaver i starten siden vi har vært usikre på hva oppgaven innebærer. Derfor har vi brukt mye tid på at alle samarbeider om en ting om gangen, men dette har bedret seg utover i prosjektperioden. Nå fordeler vi oppgaver oftere mellom oss. Ansvarsområdene er uendret. Vi følger fortsatt det vi satte opp i forprosjektet.

1.2.1 Hva er ferdigstilt?

Vi mangler å skrive mer utdypene om rollefordelinger i rapporten.

1.2.2 Hva er under arbeid?

Vi jobber med å kontinuerlig forbedre arbeidsrutinene for å bli mer effektive.

1.2.3 Tidsfrister

Ingen tidsfrister innenfor denne kategorien.

1.3 Klargjøring av problemstilling

Vi tolker at dette punktet appelerer til oppgavebeskrivelsen og kravspesifikajsonen. Oppgaven var ganske klar fra starten av, men vi har brukt mye tid på å komme fram til hvordan det skal lages.

1.3.1 Hva er ferdigstilt?

Vi vet hva vi skal lage. Ikke helt ferdig med hvordan vi skal komme fram til det enda.

1.3.2 Hva er under arbeid?

Det må nevnes at vi er i kontakt med tredjepartsleverandører angående mulighet for bestilling av deler fra de. Det er ønskelig at hele prosessen skal

foregå internt i CarAdmin, men det ser ut til at dette blir vanskelig da ingen har åpne API-er. Det ligger an til å bli en hybridløsning der man automatisk sendes til en tredjepartsleverandør sin webshop via registreringsnummer til et kjøretøy, for så å registrere bestillingen i CarAdmin. Dette er som sagt usikkert per nå.

1.3.3 Tidsfrister

I Gantt-diagrammet fra januar, satte vi en frist for oppkobling mot tredjepartsleverandør sitt API til 05.04.2019. Dette punktet er nå veldig usikkert da det mest sannsynlig ikke er mulig å koble seg opp til et. Når vi får en bekreftelse på hva tredjepartsleverandør kan bidra med, tar vi stilling til dette så fort som mulig. Status nå er at vi venter på svar. Det bør ikke ta lang tid før vi får det.

1.4 Løsningsmetode

Vår oppgave er å programmere lagermodulen. Vi har foreløpig ikke startet med programmeringen da vi følger en plan-dreven systemutviklingsmodell. Vi bruker mye tid på å sette oss inn i eksisterende kode, men vi regner med å komme i gang med programmeringen om ca en uke. Det er som tidligere nevnt at vi må bruke mye tid på kodelesing for å utnytte eksisterende funksjonalitet på en effektiv og god måte.

1.4.1 Hva er ferdigstilt?

Ingenting er programmert enda.

1.4.2 Hva er under arbeid?

Det jobbes med diagrammer og kodelesing.

1.4.3 Tidsfrister

Som tidligere nevnt skal vi være i gang med programmeringen innen 01.03.2019. Det ser ut til at vi klarer å overholde denne fristen med god margin.

1.5 Rapportskriving

Vi har startet på hovedrapporten. Mye av kapittel 1 og 2 er unnagjort. Vi må gå over dette nøyere når det kommer til perioden med rapportskriving.

Vi har per nå mest fokus på det å få på plass hvordan vi skal komme i mål med oppgaven.

1.5.1 Hva er ferdigstilt?

Ingenting i rapporten er ferdigstilt enda. Dette blir først gjort i rapportskrivingsperioden. Nå fyller vi bare inn for å gi oss selv god oversikt over hva som skal gjøres.

1.5.2 Hva er under arbeid?

Se avsnitt over.

1.5.3 Tidsfrister

Rapporten skal leveres 20.05.2019. Vi har ikke satt noen egne frister for å eventuelt bli ferdig før enda.

2 Totalstatus

Alt i alt så føler vi at vi ligger bra an. Vi ligger i det minste på skjema, egentlig litt foran. Gruppa jobber godt sammen, men vi bør sørge for å opprettholde 30 timer i uka per hode. Kravspesifikasjonen ser ut til å komme på plass snart og det nærmer seg utvikling. Rapporten skal forbedres etter utviklingen, men vi ser for oss at dette blir en parallell prosess.

3 Muligheter? Trusler/problemer?

Det kritiske punktet for oss er hvordan vi skal løse det med bestilling fra tredjepartsleverandør. Vi avventer svar og bør få dette innen få dager. Først da kan vi si noe om hvordan det blir videre.
Om utviklingen går bra og vi får tid til overs har vi mulighet til å ta et dypdykk når det kommer til rettigheter til lagermodulen. Dette kan bidra til å styrke oppgaven vår enda mer, men vi vil først og fremst ha fokus på å utvikle det som står i oppgavebeskrivelsen.

4 Motivasjon

Motivasjonen til gruppa er god. Vi ønsker å avslutte studiet med en god karakter på bacheloroppgaven. Det er ekstra motiverende da det er et reelt

oppdrag. Siden oppgaven er veldig rettet mot programmering vil det gi oss verdifull erfaring som vi kan ta med oss videre.

5 Veilederkontakt

Frode er en strålende veileder som gir gode tilbakemeldinger på det vi ønsker å få tilbakemelding på, samt andre nyttige ting å tenke på angående bacheloroppgaven. Vi er svært fornøyde.

Statusrapport 2

Marius Dahlen Østli, Bjørn Inge Kulsveen, Andreas Maurud Prøven

01.04.2019

E.8

1 Statuser

1.1 Planlegging

Ved forrige statusrapport 18.02.2019 lå vi verken foran eller bak fremdriftsplanen. Siden da har det skjedd mye både i positiv og negativ forstand.
Det negative er at leverandøren som sa de hadde en API-løsning for bestilling, ga oss beskjed om at det ikke ville la seg løse allikevel. Det betyr at det ikke finnes tredjepartsleverandører som har API for bestilling av deler. Det medførte at vi måtte skifte taktikk og avtale dette med oppdragsgiver. Vi havnet litt bak skjema på grunn av denne prosessen. Vi holder nå på med såkalt ≪web-crawling≫ for å hente ut informasjon fra MECA sin web-shop. Det er tungvint i forhold til bruk av API og om MECA skulle oppgradere web-shopen vil det mest sannsynlig ikke fungere uten å oppdatere koden. ETC ønsker allikevel at det gjøres på denne måten.

Det er derimot mye positivt som har skjedd i det siste. Vi ligger på skjema i forhold til fremdriftsplanen. Grunnoppsettet er ferdig. Deler, dekk og lagre kan legges til, slettes og redigeres. I tillegg fungerer filtreringen bra. Nå er det den avanserte funksjonaliteten som er i fokus. Bestilling fra tredjepartsleverandør fungerer nå, men det gjenstår å integrere dette i CarAdmin. Lagermodulen sin hovedside (lageroversikt) skal inneholde et avansert søkefelt der det kan søkes gjennom lagerbeholdning, dekkbeholdning og lagerregister. Dette er påbegynt, men ikke ferdigstilt. En egen fane for ordreoversikt er også blitt laget. Oppgaven spesifiserer at det skal gjelde interne og eksterne ordre, men dette har blitt avgrenset til å gjelde kun eksterne (beslutning tatt av oppdragsgiver). Dette fordi ETC allerede har et system for ordrehåndtering og det ville bli veldig mye jobb for oss å koble dette sammen med lagermodulen.

Det gjenstår nå å få integrert bestillingen av deler fra MECA, avansert søkefelt og ordreoversikt. I tillegg skal det legges til støtte for flere dekktyper i dekkbeholdningen.

I Gantt-diagrammet satte vi internasjonalisering som et eget punkt i det siste inkrementet, men dette er noe vi har gjort underveis. Det ble slik fordi det er opprettet en god standard for internasjonalisering i løsningen fra før.

1.2 Organisering av gruppens arbeid og ansvarsområder

Da vi startet med utviklingen avtalte gruppa å jobbe oftere hjemmefra. Med hjemmefra menes at gruppa var samlet hos en av oss. Det innebar fordeler

og ulemper. Ulempene var at vi ikke kom igang like tidlig som vi gjorde de to første månedene, men selv om vi startet senere så jobbet vi som oftest desto lengre utover dagen. Fordelene med å jobbe hjemmefra er tilgang på stasjonær PC med flere skjermer og roligere arbeidsforhold. Muligheten til å programmere på en større skjerm har mye å si da det gir bedre oversikt over koden.

Planen videre er å jobbe mer på campus igjen etter utviklingsperioden. Da rapportskrivingen starter for fullt, mener vi det er gunstig.

1.3 Rapportskriving

Det har blitt skrevet litt på rapporten siden sist, men det er minimalt. Utvikling har blitt og blir høyt prioritert fram til påske.

2 Totalstatus

Totalt sett ligger vi på skjema, men det er viktig at vi jobber hardt for å ikke havne bak.

3 Muligheter? Trusler/problemer?

Tiden er det som kan være en trussel. Den avanserte funksjonaliteten er ikke lett å programmere.

4 Motivasjon

Motivasjonen om å gjøre en god bacheloroppgave har alltid vært der i hele gruppa, men det kan ikke legges skjul på at det har blitt noen tunge dager innimellom. Gruppa er motivert for å gjøre en god sluttspurt slik at resultatet blir best mulig. Gruppa har fortsatt et mål om en god karakter.

5 Veilederkontakt

Vi har fortsatt hatt møte med Frode Haug hver mandag kl. 09:00. I utviklingsperioden har vi ikke hatt så mye utbytte av det, men møtene er en fin måte å starte uka på. Når det gjelder spørsmål om rapporten er vi svært fornøyd med veiledningen vi har fått.

Statusrapport 3

Marius Dahlen Østli, Bjørn Inge Kulsveen, Andreas Maurud Prøven

30.04.2019

E.11

1 Statuser

1.1 Planlegging

Ved forrige statusrapport 01.04.2019 lå vi på skjema med tanke på fremdriftsplanen. I følge denne skulle utviklingen være ferdig innen 1. mai, men det trengs enda litt tid på å bli ferdig med funksjonalitet for bestilling fra tredjepartsleverandør. Vi var en sliten gjeng som tok påskeferie fra 15-22. April. Dette trengte alle for å samle krefter til en skikkelig innspurt. Vi visste at vi kom til å havne litt bakpå på grunn av det, men vi konkluderte med at det var bedre å ha energi til å jobbe effektivt, enn å sitte lenge og jobbe tregere fordi vi var slitne. Rapportskrivingen er i gang for fullt og det har blitt produsert mange sider etter påske. Vi skal klare å komme i mål med rapporten, men det er noe usikkert om det lar seg gjøre å implementere alt som gjenstår. Bestillingen kommer mest sannsynlig på plass veldig snart. Oversikten over bestillingene som er gjort, er det ikke sikkert vi rekker å implementere. i stedet for en ordreoversikt for interne ordre er det meningen å få på plass en fane for lagerbevegelse. Denne skal oppdateres ved uttak av deler ved registrering av utført vedlikehold. Dette er nære på å komme på plass.

1.2 Organisering av gruppens arbeid og ansvarsområder

Som det ble nevnt i forrige statusrapport var planen å jobbe mer på campus når det skulle skrives rapport. Dette har vi fulgt opp og i det siste har det blitt mange dager på campus. Vi er alle enige om at vi må jobbe lengre dager enn vanlig nå fram til innleveringsfristen 20.05.2019. Gruppa er flinke til å fordele oppgaver slik at vi oppnår raskest mulig framgang. Dialogen er fortsatt tett slik at det ikke oppstår uenigheter underveis.

1.3 Rapportskriving

Rapporten begynner å ta form. Kapitell 1 og kapittel 2 er så og si ferdigstilt. I tillegg er mye på plass i kapittelet om kravspesifisering. Videre arbeid nå blir å jobbe mye med rapport og lite utvikling, selv om vi må prøve å få utviklet noe mer funksjonalitet.

2 Totalstatus

Kort oppsummert så ligger vi noe bak skjema, men vi har troen på at en god arbeidsinnsats etter å ha fått slappet av litt i påsken skal være nok til å

levere en bra bacheloroppgave. Gruppa er forberedt på å jobbe hardt i tiden fram mot fristen.

3 Muligheter? Trusler/problemer?

Ettersom det ikke ble noe oppkobling mot API-er som egentlig var utgangspunktet i oppgaven, måtte bestilling fra tredjepartsleverandør gjøres via såkalt ≪Web-crawling≫. Altså navigasjon via HTML-koden. Skulle leverandøren oppdatere webshopen kan det hende at bestillingen plutselig ikke fungerer lengre og det kan by på utfordringer hvis dette skulle skje før innleveringsfristen, og muntlig presentasjon.

4 Motivasjon

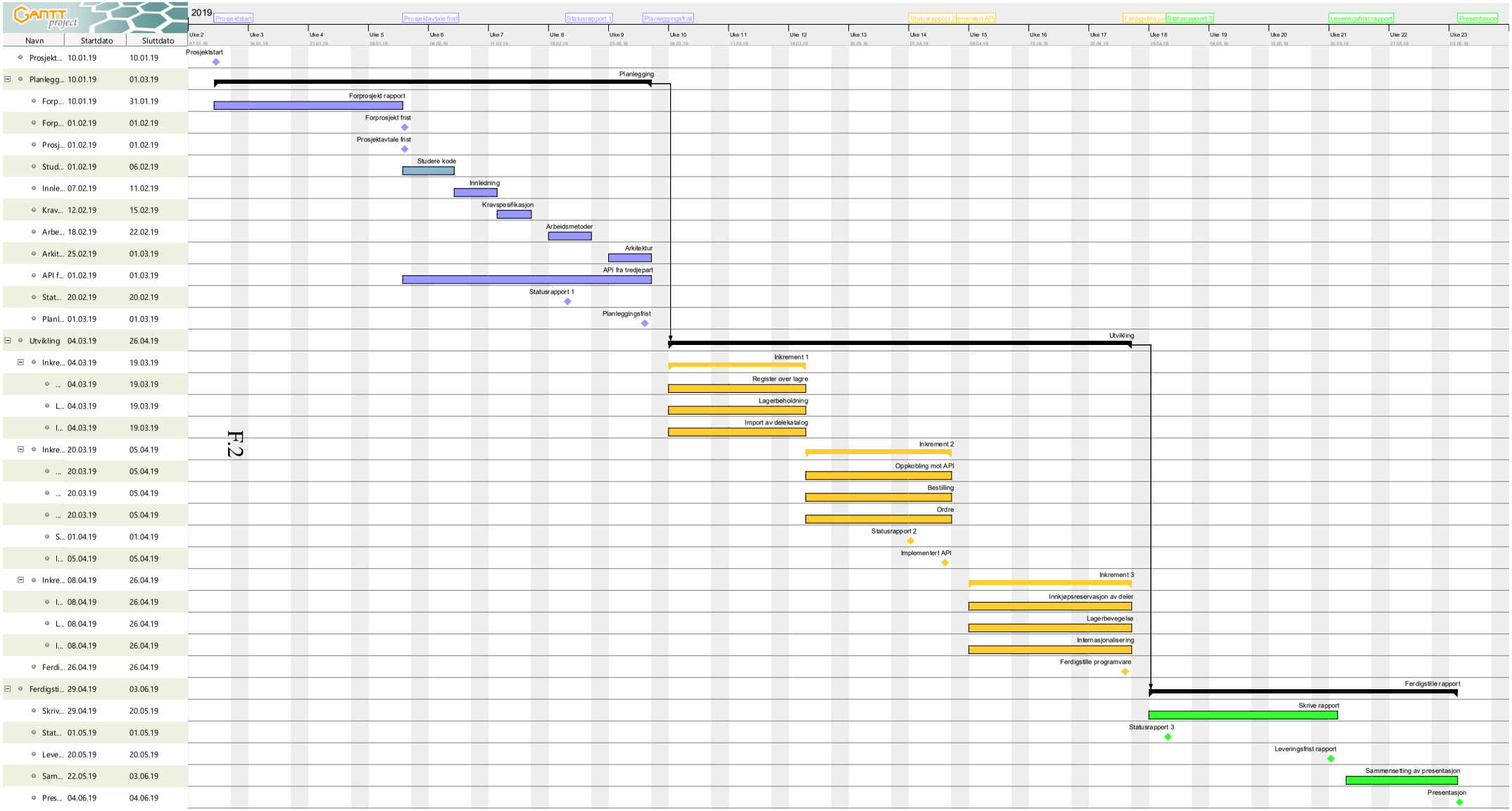
Gruppas motivasjon er meget god og alle er klare for å gjøre en god innspurt.

5 Veilederkontakt

Ettersom rapporten har kommet mer inn i bildet igjen etter påske er vi veldig fornøyd med tilbakemeldingene vi har fått fra vår veileder Frode Haug. Disse hjelper oss med å skrive en god og strukturert rapport. Vi kommer til å stille mange spørsmål om rapporten i de gjenstående møtene med Frode framover.

F Gantt-diagram fra januar

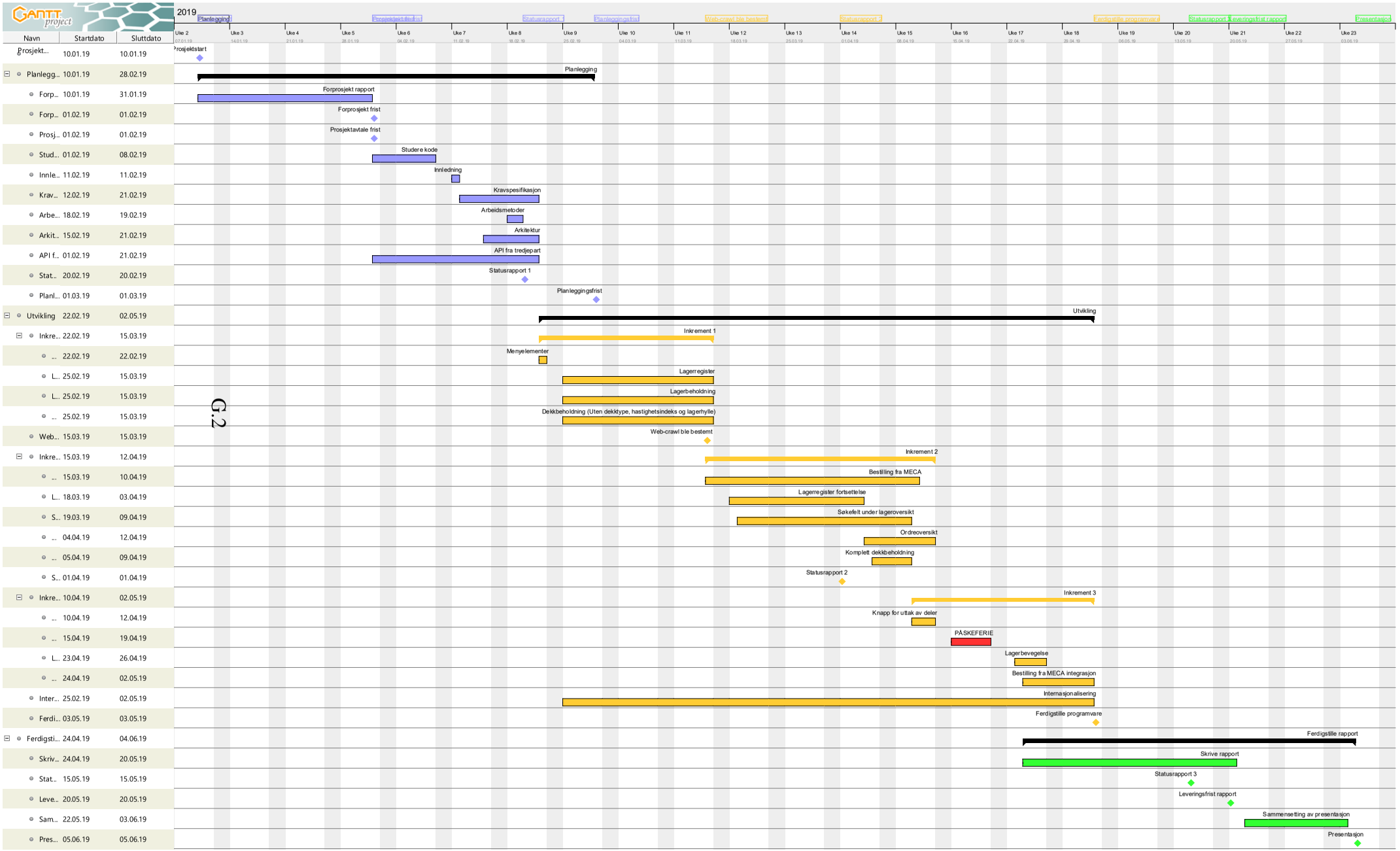
Ganttskjema



F2

G Gantt-diagram fra mai

Ganttskjema



G.2

H Møtereferater med ETC

ETC 1

Dato: 18.01.19
Våre spørsmål:

1. Hvordan skal vi koble det vi skal utvikle opp mot CarAdmin?
2. Resultatmål/effektmål?
3. Hva inngår i å lage rapporter?
4. Hvilke leverandører er det snakk om eller hvor mange? Skal vi ut til de?
5. Hentes deler via leverandørens API?
6. Hva menes med flerspråklighet? internasjonalisering/andre kodespråk?

Svar:

1. Lages i eksisterende løsning. Gjennomgang/kursing av eksisterende system og kode med Øyvind fredag 18.01.2019 (tar hele dagen)
2. CarAdmin skal ha en fungerende lagermodul (kan konkretiseres mer etterhvert).
3. Det skal lages mange ulike rapporter: Lagerbeholdning (ren lagerrapport), Hvilke varer som er på lager, telleliste (ved varetellinger), hvilke varer som er på vei inn/ut.
4. Må kontakte leverandør(er) angående bruk av API for tilgang til delelager. (Skal få liste på mest aktuelle leverandører, men Meca er garantert en av de). En bedrift kan ha interne og/eller eksterne lagre. Ønskes at deler på lager inndeles i grupper.
5. Ja, men dette må tas opp med hver enkelt leverandør som skal benytte seg av systemet.

6. Med flerspråklighet menes internasjonalisering (norsk/engelsk).

Andre ting:

- Avtale møtetidspunkter — FREDAGER KL 08:00
- Programvare vi skal benytte: IntelliJ, TomCat, (venter på liste med resten fra Øyvind i ETC)
- Lagermodulen må kunne inneholde alle lagrene en leverandør har (en til flere).
- En skal kunne søke i lagermodulen, sortere/filtrere dette smart, prioritering av lager basert på geografi.
- Dokumentarkivet til CarAdmin må kunne oppdateres (med rapporter?)
- Har fått opprettet en testbruker for CarAdmin.

ETC 2

Dato: 25.01.19
Spørsmål til ETC:

1. Resultatmål mer spesifikt
2. Mer detaljert oppgavebeskrivelse
3. Rammer
4. Hvordan dokumentere under utvikling?
5. Kontakte 3.parts leverandør, API og datablad, hvem (Liste over leverandører med tanke på tredjeparts delekataloger?) og hvordan skal disse kontaktes?
6. Bestille deler gjennom API/lagermodulen?

7. Kundegruppe?
8. Signere prosjektavtale

Svar:

1. Fått skriftlig en mer detaljert oppgavebeskrivelse som inneholder mer konkrete resultatmål
2. Skriftlig dokument med mere info
3. Rammer har blitt klarere da vi har hatt en gjennomgang av hvordan CarAdmin er bygget opp.
4. Det skal dokumenteres ved bruk av javadoc på alle klasser/variable som er public. Koden skal skrives med engelske ord, mens kommentarene skal være på norsk. Viktig å kommentere mye og detaljert. Private skal også skrives javadoc hvis det er get/set-metoder.
5. Dette kan gjøres via telefon, bør avtale møte med en leverandør siden de har mer kunnskap på hvordan et lager fungerer. Begynn med Meca på Gjøvik. Andre kan være Mekonomen eller Skruvat.
6. Skriftlig dokument med mere info
7. Kundegruppe er alle bedrifter/kommuner som har egne verksteder.
8. Prosjektavtalen ble signert.

Andre ting:

- Forskjell på ordre og bestilling: Hvis en ansatt i en bedrift ønsker en del, er det en intern ordre. Hvis delen ikke finnes på lager, må det opprettes en bestilling/rekvisisjon. Ordren brukes til å fakturere kunden.
- Lombok plugin i IntelliJ: Lombok genererer getters/setters for variable så du slipper å legge det inn manuelt.

- Bruk @Nullable/@NotNull for å flagge når du forventer at en variable skal være/ikke være null
- Se hvordan github anbefaler som arbeidsmetodikk mtp. branching osv.
- Bruker ikke unitTest i dag, må vurdere nødvendigheten av testing selv, integrasjonstest viktigst i så fall.
- Husk at encoding i intellij må settes til Windows1252

ETC 3

Dato: 01.02.19
Våre spørsmål:

1. Hvor kommer Apache Ant inn i bildet?
2. Hvilke kodefiler er relevante for oss?

Svar:

1. For å nullstille databaser, (hvis vi skulle trenge det)
2. Det som er under workshop (SaveWorkshopArt) viser hvordan serveren håndterer AJAX forespørsler. Bruk dette som utgangspunkt. Data fra database puttes inn i request-objekter før de hentes ut igjen og presenteres vha .jsp filer. Eksempel: ServiceArtOverview har funksjoner for å legge til nye produkter og sender dette til SaveWorkshopArt som har funksjonalitet for å skrive dette til siden. Dette vises på siden vha. tilhørende .jsp fil.

Andre ting:

- Lagdeling er brukt i koden for eksempel for å adskille SQL fra andre deler av koden
- Tomcat kan ikke ligge i programfiler siden mappene der er skrivebeskyttet. Generelt ikke ha utviklingsmapper i programfiler.

ETC 4

Dato: 08.02.19
Våre spørsmål:

1. Har snakket med MECA. De har en webshop som mange benytter, men de sa det ville bli vanskelig å integrere dette inn i CarAdmin. Kommentarer rundt dette? MECA skulle vurdere saken over helga.

2. Gå gjennom systemet ved hjelp av klassediagrammet.

Svar:

1. Kontakte MECA

2. ReportParameterUtil brukes som hjelp for å sette opp filter. Serializable lagrer data til disk, sånn at hvis serveren slås av så kan data hentes fra fil, transient definerer objekter dette ikke skal gjelde for. View.jsp setter opp standard oppsett, det som er likt for alle. /view/theme/ETC/contentview.jsp linje 128 laster innhold fra pagecontroller og tar ellers for seg den horisontale menyen.

ETC 5

Dato: 15.02.19
Våre spørsmål:

1. Hvilke tilganger har de forskjellige bruker rollene (superadmin)?

2. Operasjonelle krav (ikke-funksjonelle krav)

3. Gjennomgang av Use-Case diagram.

4. I produktregisteret er det oppført både deler og oppdrag som poster, er dette noe vi må ta hensyn til i lagermodulen?

5. Fakturering ved bestillinger?

Svar:

1. Hos vanlige kunder er administrator den høyeste rettigheten. Superadmin er selgeren sine rettigheter, som har tilgang til mer enn administrator, men ikke like mye som løsningsadmin.

2. Oppetid, Skalering av lagermodul. 55000 unike brukere. 100 aktive klienter som teoretisk kan bruke systemet samtidig. Skjer sjelden at så mange er på samtidig, pleier å kanskje være noen få på morgen eller ettermiddag.

3. Endre fra ta ut dekk og legg inn dekk til sjekk inn/ut dekk".

4. Se bort i fra dette. Arbeidsoppdrag skal være en egen ting. Det skal ETC gjøre selv.

5. Send E-Post til Dag. Usikkert om dette skal være en del av oppgaven eller ei.

Andre ting:

- Sikre/sanere input, bruke parametriserte SQL-spørringer, Sikkerhet ved selvskrevne API'er (Brukernavn/passord over SSL(TLS)). Vurder sikkerhetstiltak vs. tidsbruk.

ETC 6

Dato: 22.02.19
Våre spørsmål:

1. Testbruker i MECA sin nettshop (videre valg?)

2. Hva er en pakkejobb?

3. Vi tenker å legge til en ordreoversikt som en fane under rapporter-¿vedlikeholdsbok

4. Egen fane for dekkhotell under verkstedbok? Eller samle alt under produktregister?

5. Alternativ til innhenting av informasjon om deler fra tredjepartsleverandør.

6. Navngivning til databasemodell? Står vi fritt til å endre på navn? f. eks ServiceArt?

7. Hva betyr category og order under ≪Vouchertype?≫

Svar:

1. Gå for en løsning med web-shopen. (Web-crawl). Gjør om informasjon i html til objekter.

2. Kombinasjon av for eksempel bytte av deler på et kjøretøy + arbeidstid.

3. Dette bør/må i så fall ligge som en fane under et kjøretøy.

4. Start med en egen fane for lagermodulen i menyen til venstre. Under denne kan vi legge til fane for lageroversikt, lagerregister, søkefelt osv.

5. Se svar 1.

6. Ikke noe krav til navn, men prøv å navngi på den måten at tabeller som for eksempel har med verksted å gjøre starter med Service.

7. Category er hva slags bilag-type som er angitt. Order er rekkefølgen.

ETC 7

Dato: 01.03.19
Våre spørsmål:

1. productsetting (ProductID?-¿GlobalID?)

2. adminroleaction (action?)

3. AssetSettings.java

Svar:

1. Kundenummer
2. Brukertilganger i systemet
3. Produkter kunden har tilgang til

Andre ting:

- Dekkbeholdning: modell, når nytt, sistbrukt på kjøretøy, hastighets bokstav
- Lagerregister: internt/ekstern, lagertype, adresse, kontaktperson
- Lagerbeholdning: passer til bil

ETC 8

Dato: 08.03.19
Våre spørsmål:

1. I diverse .jsp filer finnes jsServiceArtReport.getReport(), hvor finner man dette objektet? (eks: serviceArtReport.jsp:45)
2. I ServiceArt.java brukes serialVersionUID, trenger vi å opprette den?
3. Venter på tilgang til testmiljøet for Veng sitt API, skal vi starte å importere delekatalog eller avvente til vi har fått tilgang?
4. Legge til uttak av deler/dekk i registerServiceDialog", trengs det andre steder?

Svar:

1. ETCDataGrid.js, oppretter js filer for kontrollerne.

2. Trengs ikke. Brukte før for å sikkre at serialization gjennoppretter rett objekt.

3. Vent på svar. Ring til uka.

4. Ja.

Andre ting:

- Ikke vis id fra database til brukeren, legg til eget varenummer som fylles ut av bruker.

ETC 9

Dato: 15.03.19
Våre spørsmål:

1. Hvordan fungerer Lombok?

Svar:

1. I stedet for å lage getters og setters til variabler i en klasse, skriv @lombok. Data over klassedeklarasjonen og Lombok ordner det automatisk.

Andre ting:

- For å utføre https kall til MECA, må sertifikatet dems lastes ned og legges inn manuelt på serveren

- For å enklere kunne håndtere html dokumenter så brukes Jsoup som må legges til i ETC sitt .jar bibliotek

ETC 10

Dato: 29.03.19
Våre spørsmål:

1. Hvilke kjøretøyer støtter CarAdmin? Dette med tanke på hvilke dekktyper som skal legges til.

2. Hvor skal oversikt over ordre legges?

3. Har ETC pratet med tredjepartsleverandører?

Svar:

1. Dekktyper til kjøretøyer registrert i CarAdmin. Ønsket å se plassering på lager (reol/hylle/...). Ønskelig også med dekkdimensjon for å vite om det er plass i lagerhylla.

2. Integrer lagerordre inn i eksisterende ordreoversikt. Bilagsmottak finnes ikke i vår løsning, vi må tenke at denne finnes. Det er slik fordi enkelte bilag krever behandling.

3. Ikke enda.

Andre ting:

- Kult om vi hadde fått til at sertifikater legges inn automatisk! Utilityfunksjoner for å gjøre dette dynamisk?

- Få med kostnad i ordrer til verksted/dekkhotell (lagerbeholdning/dekkbeholdning). Et lager har sitt eget økonomisystem.

- Bør legge til et område for lager der man kan sette brukernavn/passord for innlogging hos tredjepartsleverandører. Samme opplegg som under kundeoversikt-¿oppsett, men bruk tekstfelt i stedet for sjekkbokser. Øivind skal sjekke dette nærmere og si oss hvor det bør legges inn i CarAdmin.

- Bør lages en oversikt over hvordan oppkoblingen til MECA fungerer med tanke på videre utvikling/vedlikehold, men også med tanke på rapporten vår.

ETC 11

Dato: 05.04.19
Våre spørsmål:

1. Status på det vi har gjort så langt
2. Meca status

Svar:

1. Status gjennomgang ok
2. Tannhjul -¿ bestill (åpner opp side eller dialog for bestilling)

Andre ting:

- Notis i rapporten hvordan håndtere at det er en handlekurv per konto hvis flere skal bestille.
- Databasetabell Options styrer løsningsoppsettsida. Legg inn 3 linjer (MECA av og på, MECA brukernavn og MECA passord)
- Classfactory getSolutionPreferences (systempreferences reflekterer alt som ligger i Options-tabellen.
- Start med eksterne ordre på ordreoversiktsiden. ETC skal komme tilbake til oss med eventuell håndtering av interne ordre.
- icons8.com for logoer, ETC har lisenser til disse. (stil Windows 8 Metro/IOS 10)

ETC 12

Dato: 12.04.19
Våre spørsmål:

1. Grei plassering av knapp for registrering av deler under registrering av utført vedlikehold?

2. Hva er enklest å kode for å få en dynamisk side? Via dialog eller egen fane? (MECA bestilling)

Svar:

1. Gjør som vi har tenkt. Hadde lagermodulen vært knyttet opp mot produkt-registeret skulle den vært under arbeidslinjer, men det trenger vi ikke gjøre.

2. Finnes ikke noe avansert måte, må gjøres som vanlig". Bør kanskje lage flere dialoger for å innhente nok informasjon til å gjøre en bestilling. Se prosess for å legge til nytt kjøretøy (newAsset). Det er måten ETC løser innhenting av data dynamisk.

Andre ting:

- Bruk ordreoversikten vi har laget. Bruk den som vi tenkte i utgangspunktet. Grunnleggende misforståelse som gjør at det vil ta for lang tid å rette opp i det vi har gjort.

- Ctrl+Shift+F globalt innholdssøk

- Søkefelt: *etc_view.js* studer denne! spesielt funksjonen "catcomplete"

ETC 13

Dato: 26.04.19
Våre spørsmål:

1. AJAX fil for editCarSteps (kode)

Svar:

1. Brukt en gammel metode. Vi fikk oppklart dette.

ETC 14

Dato: 03.05.19
Våre spørsmål:

1. Feilmelding i systemet
2. Hvor mange moduler er egentlig CarAdmin delt inn i? Noe som ikke henger helt på grep ut ifra forprosjektet. De fire nevnte modulene, er det de viktigste modulene? Eller er det kun de 4 modulene som eksisterer?
3. Hva er products-ID 11 og 12? Global ID 2 og 46?
4. Hvor mye informasjon kan stå i rapporten med tanke på diagrammer og figurer?

Svar:

1. Kommer fordi persisten storage prøver laste inn objekter som er endret siden sist kompilering
2. Sidemenyer er sekjsoner"Ingen 1-1 relasjon mot modulene. Modul er et løstbegrep. Dette er for å tilpasse/skreddersydde løsningner. Bruk produkter/tjenester"i stedet for "moduler".
3. Plots, products har globale ID-er.
4. Send tvilsomme tekster/figurer (med tanke på publiseringer av koderelatert informasjon) til ETC.

Andre ting:

- Beskriv valget med HTML-kode i AJAX-klassene
- Beskriv grensesnittet som kan brukes for bestilling i rapporten

ETC 15

Dato: 10.05.19
Våre spørsmål:

1. Antall brukere av CarAdmin?
2. Diskutere skype-demo for Andreas Grimstad, som er vår kontaktperson i MECA. Dette med tanke på innsyn i CarAdmin.

Svar:

1. ca. 40000
2. Kan vise demo til MECA, unngå å vise kode. Spesielt det som ikke har med bestillingen å gjøre.

I Møtereferater med veileder

Veiledermøte 1

Møte med veileder Frode Haug 11.01.2019
Referat:

- Faste møter med Frode mandager kl. 09:00
- Les notat om bacheloroppgave fra 2017 for nyttig informasjon.
- Grupperegler!
- Januar: Prosjektplan, Sette oss inn i ny teknologi. Få på plass HVA SKAL LAGES? (gjelder kravpspek)
- Kapittel 2 i planleggingen bør skrives godt! skriv gjerne dette før kap 1
- Kap 1: Hva tjener bedriften på dette? effektmål/resultatmål, læringsutbytte, tekniske rammer...
- I Gantt-diagrammet marker tydelig milepæler og beslutningspunkter.
- Lag timelogg fra første stund.
- Veileder ønsker 3 statusrapporter: ca 20. februar, ca 1. april og ca 1. mai.
- Forutsett at de som leser rapporten er på samme nivå som oss etter 2,5 år på studiet.
- Skal Frode se på noe må det sendes via mail før fredag kl 09:00 (trenger 24 timer i forveien for å se på det).
- Vurder nøye utviklingsmodell, hvis Scrum så bør Scrummaster og prosjektleder være samme person.
- Lag ett Gantt-diagram i Januar (sånn tror vi det blir)
- Så et nytt i Mai (sånn ble det) og hva lærte vi?

Veiledermøte 2

Dato: 21.01.2019
Våre spørsmål:

1. Hvor detaljert skal Gantt-diagrammet være?

Svar:

1. Kravet er at vi skal ha en plan på hva vi skal drive med. Må ta høyde for at utviklingsperioden kan bli lengre. Det må tas med beslutningspunkter og flere milepæler.

Andre ting:

- Sette oss inn i teknologi
- Trenger ikke starte på hovedrapport før etter forprosjektet.
- Tilbakemelding på rapporten: Fikk utskrift av rapporten. Frode har sett over og skrevet på kommentarer.

Veiledermøte 3

Dato: 28.01.2019
Våre spørsmål:

1. Hva bør vi fokusere på fram til fristen for forprosjektet?

Svar:

1. Finpussing. Frode godt fornøyd med forprosjektrapporten.

Andre ting:

- Gjør Gantt-diagrammet større
- Fjerne signaturer fra gruppereglene
- Generelt veldig bra!

Veiledermøte 4

Dato: 04.02.2019
Våre spørsmål:

1. Godkjente referanser?

Svar:

1. Bruk Vancouver-metoden

Andre ting:

- Fokuser på det praktiske, ikke stress med rapporten enda.

Veiledermøte 5

Dato: 11.02.2019
Våre spørsmål:

1. Hvordan skal vi starte på rapporten

Svar:

1. Kravspesifikasjon(hva), design(hvordan), utvikling(kode)

Veiledermøte 6

Dato: 18.02.2019
Våre spørsmål:

1. Tolket Use-Case riktig med tanke på avhengigheter?

Svar:

1. Hør med Tom Røise angående Use-Case diagram.

Annet:

- Se tilbakemeldinger på ark fra Frode. Fikk også mal for statusrapport som skal inn 20.02.2019.

Veiledermøte 7

Dato: 25.02.2019
Våre spørsmål:

1. Hvor mye skal vi vektlegge beskrivelse av eksisterende system da det er såpass stort fra før av? Dette blir litt irrelevant for vår oppgave.

Svar:

1. Bør bruke et par sider. Ikke nødvendigvis mer enn det. Lurt med figur (utvid den vi allerede har).

Annet:

- Dokumenter hvorfor det blir web-crawling i stedet for API-oppkobling.

Veiledermøte 8

Dato: 11.03.2019
Våre spørsmål:

1. Har vi tilgang til VM'er for å hoste nettsiden, enten under utvikling, men hvertfall til bachelor fremføring.

Svar:

1. Ja, snakk med IT-tjenesten (Lars Erik Pedersen).

Veiledermøte 9

Dato: 08.04.2019 Våre spørsmål:

1. Statusrapport 01.04.2019

Svar:

1. Statusrapporten så grei ut og det var ikke noe å si i forhold til den.

Veiledermøte 10

Dato: 29.04.2019
Våre spørsmål:

1. Plassering av planlegging før og etter. I samme avsnitt i kapittel 1 eller orginal plan i kapittel 1 og endringer som et avsnitt under drøfting/diskusjon?
2. Skrivemåte (objektivt, men skrive som «er» eller «har vært»)?
3. Plassering av endringer i oppgaven (avgrensniger underveis)
4. Risikovurdering under kravspesifikasjon eller arbeidsmetodikk?

Svar:

1. Drøftes i 8.1?
2. Skriv i fortid!
3. Beskriv kort endringer i punktene i oppgavebeskrivelsen som egne avsnitt under punktene det gjelder. Utdyp det mer i kapittel 8.1
4. Risikovurdering etter sikkerhetskrav som et eget underkapittel i kravspesifikasjonen.

Andre ting:

- Neste møte mandag 6. mai kl. 13:00
- Introduser annerledes i kap 1 om fagområde
- Kap 2 skal henvises til i kap 8.1
- Utdyp inkrementell sekvensiell metoden (legg dette til som en fotnote?)
- Innledning på hvert kapittel. Ikke hopp rett på en subsection.

- Kan ta med det vi har kodet i Design-kapittelet så lenge vi sier ifra i tidlig i kapittelet.
- Bakgrunnsinformasjon om ETC kan tas med i forordet

Veiledermøte 11

Dato: 06.05.2019
Våre spørsmål:

1. Prosjektmål, kopierefra forprosjektet og inn i rapporten? (med tanke på drøfting i kap 8)
2. Vedlegg en del av rapporten? eller to separate filer?
3. Beskrive hvert inkrement under implementasjon? eller under arbeidsmetodikk? (tenker på detaljer om hva som ble utviklet i hver av de)
4. Legge et helt avsnitt der vi diskuterer kildekode som vedlegg?
5. Lage nytt Use-Case som beskriver hvordan det ble under kap 8.1?
6. Hvor ofte må det refereres? For eksempel vi nevner ETC mange ganger. Holder det med en gang eller må vi referere hver gang vi nevner det?
7. Plassering av Gantt (etter prosessen) i kap 8?

Svar:

1. Kan ta det med under formål. Kan henvise fra kap 8.
2. Alt skal være en PDF. Untatt kode (zip-fil).
3. Kan ta det med i arbeidsmetodikk da det blir bedre for leseren.
4. Hør med ETC, abslolutt best med kodeutsnitt!

5. Dette kan være lurt. Ta det med i kapittel 8.1
6. Bedre med for mye referering enn for lite, Trenger ikke hver gang vi sier ETC.
7. Ja, passer bra å ta det med under kap 8.1.

Andre ting:

- Vedlegg A,B,C....
- Vedlegg etter referanser.
- Store figurer kan legges som vedlegg i tillegg. (Se større versjon i vedlegg...)
- Lese nøye over rapporten.
- Ekstra på hva vi gjorde på veiledningsmøter i januar/februar.
- Detaljert Use-Case, forklar hvorfor vi valgte de vi gjorde. (Det hadde blitt mange sider om vi skulle beskrevet alle).
- Tekst før figurer i Design-kapittelet.
- Kravspek er (hva?hva?hva?), Design og Implementasjon er (hvordan?,hvordan?,hvordan?)
- Pass på ugunstige linjeskift.
- Må ha et 8.1.2 som henviser til alternativer. Valg underveis, andre muligheter?
- Driftsetting under måloppnåelse?
- Ikke sidetall på vedleggene!
- Møte kl 13:00 neste mandag, lever forrige tilbakemelding før møtet. Holder å levere denne fredag morgen. Kan vente til søndagskvelden med å sende ny versjon.

Veiledermøte 12

Dato: 13.05.2019
Våre spørsmål:

1. Skrive i fortid på prosjektmål, oppgavebeskrivelse og kravspek?
2. Regler for font-størrelse? Times New Roman 12 + 1,5 linjeavstand? LaTeX default greit med tanke på marger osv?
3. Hvor finner vi standard mal for forside og abstracts?
4. Møtereferater med dato?

Svar:

1. Trenger ikke det så lenge vi skriver i fortid i starten av kapitlene.
2. Sjekk ut dette på NTNU sine side.
3. Se PDF-mail. Sjekk ut grafisk senter på NTNU-sidene.
4. Ta med det.

Andre ting:

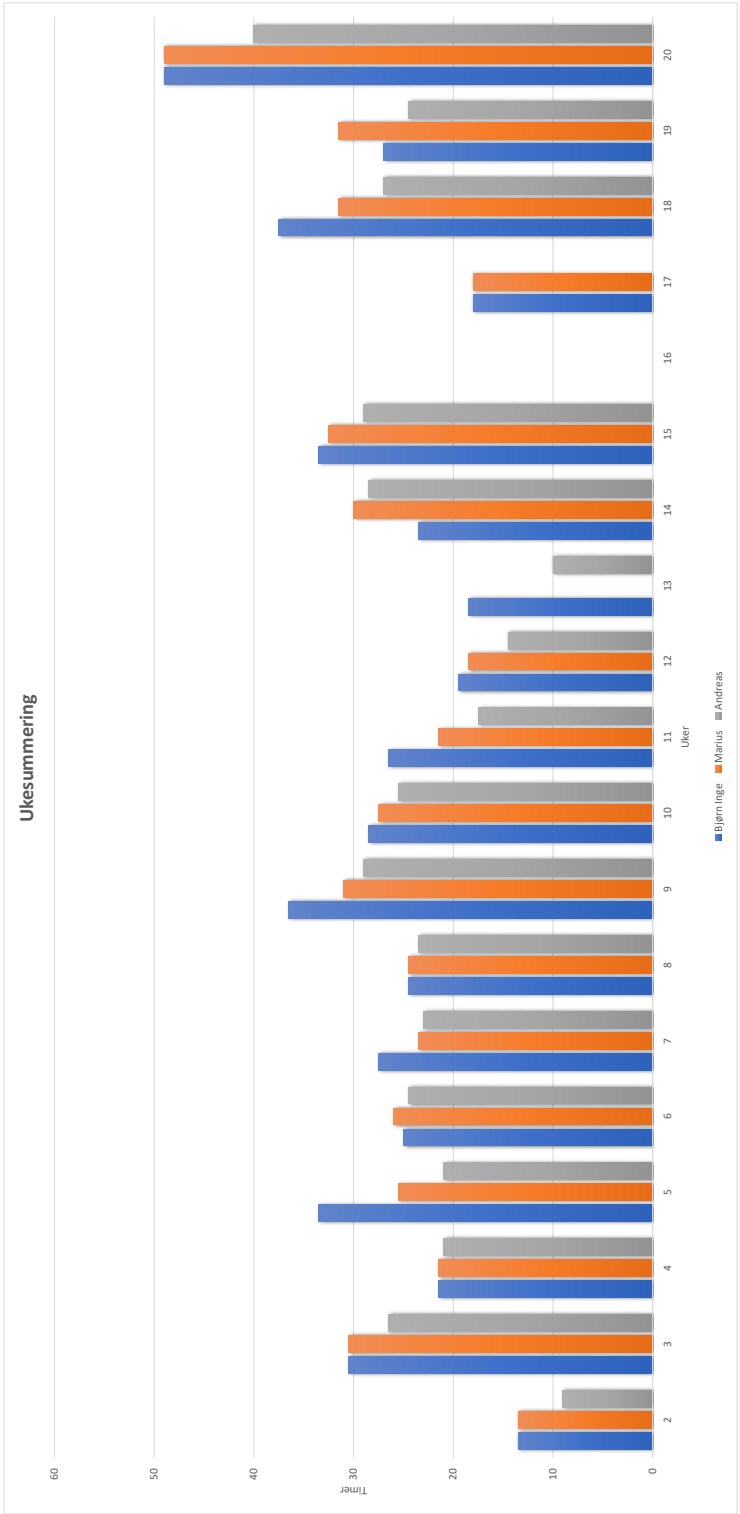
- Bruk tekst foran figurer.
- Ikke bruk parenteser rundt figurreferering. bruk: Figur¡blank¿¡tall¿.
- Flere eksempler på grensesnitt.
- Endre kapittelnavnet til Kodedesign og Implementasjon.
- Må gjøre figurer lesbare!
- Intro til inkrementer i implementasjon
- Dele opp mvc-figur?

- Evaluer risikoanalysen i kapittel 8
- Skriv definisjoner alfabetisk.

J Domenemodell

Denne figuren er plassert i egen zip-fil grunnet konfidensiell informasjon.

K Timeliste



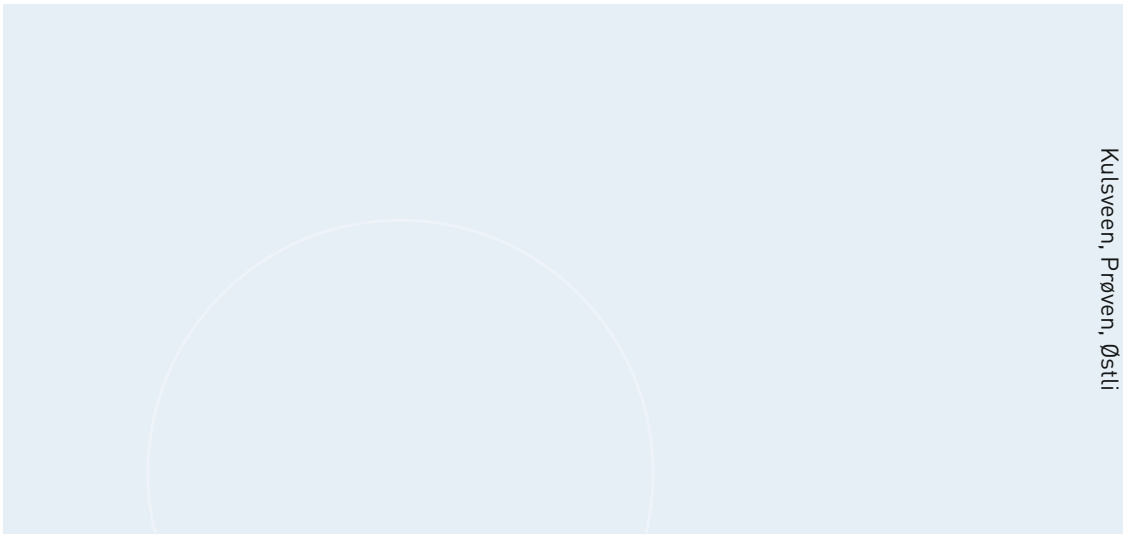
K.2

Uke:	Dato:	Alle Gjøremål:	Bjørn Inge Timer:	 Kommentar:	Marius Timer:	 Kommentar:	Andreas Timer:	 Kommentar:
2	10.01.19	Første møte med ETC	5		5		5	
	11.01.19	Første møte med Frode Oppsett av grupperegler Mal for prosjektplan Installasjon av nødvendig software	8.5		8.5		4	Jobb fra kl 12, Mangler installasjon av software
	Ukeslutt:		**13.5**		**13.5**		**9**	
3	14.01.19	Installasjon av nødvendig software Oppsett av tomcat	6		6		5	
	15.01.19	Oppsett av HTTPS I tomcat Startet på prosjektplanen (1.1,2.1,2.2)	6		6		5	
	16.01.19	Startet på prosjektplanen (1.2,1.3,3,5)	6.5		6.5		6	
	17.01.19	Startet på prosjektplanen (6) Oppsett av Gantt-diagram	6		6		4.5	
	18.01.19	Startet på prosjektplanen (4) Bestemt utviklingsmodell Generell jobbing med prosjektplanen	6		6		6	
	Ukeslutt:		**30.5**		**30.5**		**26.5**	
4	21.01.19	Forbedring av forprosjekt utfra tilbakemelding fra veileder.	6		6		5.5	
	22.01.19	TØL 1011 seminar.	0		0		0	
	23.01.19	TØL 1011 seminar. Lesing og øving med AJAX.	2		2		2	
	24.01.19	Fagområde Generell jobbing med prosjektplanen	5.5		5.5		5.5	
	25.01.19	Gjennomgang av kildekode med ETC Oppsett av utviklingsmiljø Utbedring av oppgavebeskrivelse Signert prosjektavtale	8		8		8	
	Ukeslutt:		**21.5**		**21.5**		**21**	
5	28.01.19	Møte med Frode Jobbing med forprosjekt	5.5		5.5		6	
	29.01.19	Ferdigstilling av forprosjekt	5.5		5.5		4.5	
	30.01.19	Levert forprosjekt Lest litt kode	6		2	Dårlig form	3.5	
	31.01.19	Oppsett av Hovedrapport	6.5		6.5		3	
	01.02.19	Møte med ETC Lesing av kildekode	10	4 timer ektra grunnet oppsett av klassehierarki	6		4	Måtte dra på jobb
	Ukeslutt:		**33.5**		**25.5**		**21**	
6	04.02.19	Cyber dag (Karriere dag) Lese kode	4		4		4	
	05.02.19	Lese kode	4.5		4.5		4.5	
	06.02.19	Lese kode Samtale med MECA Sekvensdiagram for MECA Litt rapportskriving	6.5		6.5		6.5	
	07.02.19	Lese kode Ferdig med klassehierarki (MVC)	5.5		5.5		5	
	08.02.19	Møte med ETC Siste dag med lesing av kode	4.5		5.5	1 time lørdag	4.5	
	Ukeslutt:		**25**		**26**		**24.5**	
	11.02.19	Møte med Frode Innledning Startet på Use Case	5		1	Syk (Lest litt kode)	5	
	12.02.19	Lagt til mere i Use Case	5		5		4	
	13.02.19	Kontaktet Mekonomen og Skruvat Lagt til mere i Use Case	5		5		4	

7	14.02.19	Jobbet med detaljert UseCase Startet på sikkerhets- og operasjonelle krav	6.5		6.5		4	
	15.02.19	Møte med ETC Opreasjonelle og sikkerhetskrav Startet på domenemodell	6		6		6	
	Ukeslutt:		**27.5**		**23.5**		**23**	
8	18.02.19	Møte med Frode Statusrapport 1 Oppsett av Trello	5		5		5	
	19.02.19	TØL1011 - Mappe1	0		0		0	
	20.02.19	Databasemodell Testkoding Rapportskriving	6		6		6	
	21.02.19	Utkast av vår databasemodell Planlegging av design	5.5		5.5		5	
	22.02.19	Møte med ETC Kodet inn: - Menyvalg - Fane for lagerbeholdning Kontaktet 5 nye leverandører (avventer svar)	8		8		7.5	
	Ukeslutt:		**24.5**		**24.5**		**23.5**	
9	25-02-19	Koding av lagerbeholdning, dekkbeholdning og lagerregister	5.5		5.5		5.5	
	26-02-19	Koding av lagerbeholdning, dekkbeholdning og lagerregister	9		4		3	
	27-02-19	Koding av lagerbeholdning, dekkbeholdning og lagerregister	7		6.5		6	
	28-02-19	Koding av lagerbeholdning, dekkbeholdning og lagerregister	8		8		7.5	
	01-03-19	Møte med ETC Koding	7		7		7	
	Ukeslutt:		**36.5**		**31**		**29**	
10	04-03-19	Koding av lagerbeholdning, dekkbeholdning og lagerregister	4		4		4	
	05-03-19	Koding av lagerbeholdning, dekkbeholdning og lagerregister	5.5		5		4	
	06-03-19	Kurs i rapportskriving Koding av lagerbeholdning, dekkbeholdning og lagerregister	5.5		5		5	
	07-03-19	Koding av lagerbeholdning, dekkbeholdning og lagerregister	7		7		6.5	
	08-03-19	Møte med ETC Koding	6.5		6.5		6	
	Ukeslutt:		**28.5**		**27.5**		**25.5**	
11	11-03-19	Møte med Frode Koding Oppsett av VM	5		5		5	
	12-03-19	Koding av dialog for nye dekk Oppsett av VM	5.5		5.5	Jobbintervju fra morgenen	5.5	
	13-03-19	TØL1011	0		0		0	
	14-03-19	Lagerbeholdning så godt som ferdig Versjonskontroll på VM	5		5		5	
	15-03-19	Møte med ETC Oppkobling til MECA Redigering og sletting av dekk	11		6		2	Jobb
	Ukeslutt:		**26.5**		**21.5**		**17.5**	
	18-03-19	Fikset lagring, sletting og redigering av dekk og lagre. Søke skiltnummer og logge inn på meca	12.5		8.5		4.5	

12	19-03-19	Startet på lageroversikt (avansert søkefelt). Hente vareinfo fra meca	7		10		7	
	20-03-19	Jobbsøking	0		0		3	
	21-03-19	Jobbsøking	0		0		0	
	22-03-19	Jobbsøking	0		0		0	
	Ukeslutt:		**19.5**		**18.5**		**14.5**	
13	25-03-19	Endringer i oppkobling mot Meca	5.5		0	Jobbintervju-case	7	
	26-03-19	Fikse innloging til Meca da den tidligere ikke fungerte	6		0	Jobbintervju-case	3	
	27-03-19	TØL 1011 Mappe 2	0		0	Jobbintervju	0	
	28-03-19	Meca - legg varer i handlekurv, hent alle varer fra handlekurv	7		0	TØL1011-Mappe 2	0	Hyttetur
	29-03-19	TØL 1011 Mappe 2	0		0	TØL1011-Mappe 2	0	Hyttetur
	Ukeslutt:		**18.5**		**0**		**10**	
14	01-04-19	Meca - bestilling fungerer Lageroversikt (avansert søkefelt)	9		7		7	
	02-04-19	Lagt til søkfelt under lageroversikt, men funksjonaliteten ikke på plass. Meca - ordre status	8.5		8.5		7	
	03-04-19	Arbeid med avansert søkefelt	6		6		6	
	04-04-19	Statusrapport Koding av ordreoversikt	0	Jobb intervju, Bergen	4		4	
	05-04-19	Kolonne for lagerhylle i dekkbeholdning	0	Bergen	4.5		4.5	
	Ukeslutt:		**23.5**		**30**		**28.5**	
15	08-04-19	Oppsett av BO for PartMeca Kolonne i dekkbeholdning som sier hvilket lager et dekksett befinner seg på	8		5		5	
	09-04-19	Lagt til flere dekktyper i dekkbeholdning Avansert søkefelt	5		5		5	
	10-04-19	Knapp for registrering av deler under registrering av utført vedlikehold	8		8		7	
	11-04-19	Knapp for registrering av deler under registrering av utført vedlikehold	6.5		6.5		6	
	12-04-19	Møte med ETC	6		8	+2 timer lørdag	6	
	Ukeslutt:		**33.5**		**32.5**		**29**	
16	15-04-19	Påskeferie	0		0		0	
	16-04-19	Påskeferie	0		0		0	
	17-04-19	Påskeferie	0		0		0	
	18-04-19	Påskeferie	0		0		0	
	19-04-19	Påskeferie	0		0		0	
	Ukeslutt:		**0**		**0**		**0**	
17	22-04-19	Påskeferie	0		0		0	
	23-04-19	TØL1011 - Mappe 3	0		0		0	
	24-04-19	Oppsett av view for bestilling fra MECA TØL1011 - Mappe 3 Rapport kap.1	5		5		0	Jobb intervju
	25-04-19	Oppsett av view for bestilling fra MECA Rapport kap. 1	6		6		0	Jobb intervju
	26-04-19	View for MECA oppdateres dynamisk Møte ETC Rapport kap. 1 og kap. 2	7		7		0	Jobb intervju
	Ukeslutt:		**18**		**18**		**0**	
	29-04-19	View for MECA steg 1 ferdig Møte med Frode Rapport kap. 1 og kap 2. (nesten ferdig)	10	K.5	9		0	Jobb-samling

18	30-04-19	View for MECA steg 2 under arbeid Statusrapport 3 Rapport kap. 2 klart, men må få tilbakemelding fra Frode Rapport kap. 8 påbegynt Rapport kap. 4	8.5		7.5		8	
	01-05-19	View for MECA steg 2 ferdig Rapport kap 8. Rapport kap 6. Påbegynt Laget databasemodell	8		8		8	
	02-05-19	View for MECA steg 3 ferdig	5		5		5	
	03-05-19	Møte ETC, TØL 1011	6	5t Lørdag	2	1t Lørdag	6	5t Lørdag
	Ukeslutt:		**37.5**		**31.5**		**27**	
19	06-05-19	Frode møte Kap 6.8 Bestilling, MecaConnection Kap 6.5,6.6,6.7 Inkrementer	7		7		6.5	
	07-05-19	Innlevering av mappeoppgaver i TØL1011 Rapport kap 1.3, 6.10, 6.11, 6.12	2.5		2.5		2.5	
	08-05-19	Rapport kap 6.4, 8.1.2, 6.11, 1.3, 1.2	6		7.5		6	
	09-05-19	Rapport kap 6 Telling av kodelinjer	0	Jobb-samling	2.5		3.5	
	10-05-19	Masse rapport skriving	11.5		12		6	
	Ukeslutt:		**27**		**31.5**		**24.5**	
20	13-05-19	Møte med Frode Gantt-diagram etter prosjektet Rettet opp i figurer Lagt inn møtereferater som vedlegg Retting av noen feil i rapporten	8		8		8	
	14-05-19	Lagt til vedlegg Nummerert appendix Annen finpuss	9		9		9	
	15-05-19	Sammendrag og Abstract Annen finpuss	9		9		9	
	16-05-19	Korrektur-retting	7		7		7	
	17-05-19	17.Mai Lørdag og søndag ble brukt til finlesing av rapport	16	3 timer lørdag 13 timer søndag	16	3 timer lørdag 13 timer søndag	7	Jobb lørdag, Jobb søndag 7 timer søndag
	Ukeslutt:		**49**		**49**		**40**	



Kulsveen, Prøven, Østli

Lagermodul

NTNU

Kunnskap for en bedre verden