Informatikk Prosjektarbeid I

IT1901 - NTNU
Gruppe 10

Halvorsen, Jørgen

Hansen, Aleksander

Hansen, Ruben

Hansen, Thomas

Haraldsen, Mats

Contents

1 INTRODUKSJON

1.1 FAGET

Fagbeskrivelsen er hentet fra fagsiden til NTNU [1]. I dette kurset skal en gruppe på 4-7 studenter gjennomføre et mellomstort programmeringsprosjekt. Hovedmålet er å gi studentene forståelse for samspillet mellom prosess og produkt-orienterte utfordringer og aktiviteter i et programmeringsprosjekt. Viktige aspekter er kunnskap om metoder og teorier for organisering av gruppe-programmeringsprosjekter (for eksempel CVS for konfigurasjonsstyring, JavaDoc for dokumentasjon, Java Swing for grafisk brukergrensesnitt). Studenter må arbeide med og reflektere over integrasjonen av ulike komponenter (for eksempel utvikling av Java med Eclipse utviklingsverktøy og MySQL database) for å sette sammen et større software-produkt. Oppsummering: Målet med kurset er å forbedre studentenes praktiske ferdigheter i gruppearbeid og i javaprogrammering.

1.2 OPPGAVEN

Prosjektet er et mellomstort programeringsprosjekt som skal fullføres av en gruppe studenter i løpet av et semester. Prosjektet legger stor vekt på utviklingsprossessen, og krever derfor at gruppen samarbeider som om det er et ekte utvikiklingsprosjekt i arbeidslivet, med en ekte kunde som man lager produktet for. Dette fører med seg en rekke urfordringer som er ment som en læringsprosses der gruppa er avhengig av å jobbe sammen og samarbeide for å finne løsninger. Det betyr at det er gruppemedlemmenes evne til å samarbeide er grunnleggende for et velykket prosjekt.

1.3 GRUPPEN

Gruppen vår består av fem studenter fra NTNU. Medlemmene har litt forskjellig kunnskapsnivå og forkunnskaper om prosesser som blir brukt i prosjektet. Fire av gruppemedlemmene går andre året på informatikkstudiet og siste medlemmet går tredje året på en fordypning informatkk som en del av en bachelor i lingvistikk. Alle på gruppa har derfor forkunnskaper om programmering i Java med Eclipse fra tidligere fag. Andre forkunnskaper er derimot litt forskjellig. Det gir oss muligheten til å hjelpe og lære av hverandre.

1.4 BESKRIVELSE AV PROSJEKTET

Det er et mellomstort programmeringsprosjekt som tar utgangspunkt i NTNU-koiene og går ut på å lage et administrasjonssystem for koiene. Systemet skal ta for seg en effektiv drift av koienettet i tillegg til de offentlige tilgjengelige nettsidene [2]. Rapportering står sentralt i det

[1] http://www.ntnu.no/studier/emner/IT1901tab=omEmnet

[2] http://org.ntnu.no/koiene/

nye systemet. Det skal være mulig å ta imot og behandle rapporter etter en bruker av koienettet har vært på besøk på koiene. Brukerne skal kunne si ifra om noe er ødelagt og se status ved diverse mangler. Administrator til systemet skal ha muligheten til å se status på koiene og oppdatere statusen når nytt utstyr er fikset eller kjøpt inn.

Dette systemet skal lages av en gruppe studenter i løpet av et semester. Ved siden av programmeringen står utviklingsmetoden og rapporteringen av prosjektet veldig sentralt. Det er en del utfordringer som gruppen ikke her helt kjent med fra før av. Blant annet utviklingsmetoden Scrum og databaseoppsettet i MySQL. Ingen av medlemmene har jobbet noe spesielt med dette før.

1.5 PROSESSMETODE

Prosessmetode har mye å si for hvordan prosjektarbeidet utføres. Det lønner seg å velge en prosessmetode som passer til prosjektet. Vi har valgt å bruke Scrum som er en smidig prosessmetode. Prosjektet er lagt til rette for en smidig prosessmetode som Scrum, i tillegg er Scrum mye brukt i arbeidslivet og senere i studiet. Scrum krever at prosjektet blir brutt ned i flere klare oppgaver som løses gjennom korte sprinter med tilbakemelding fra kunden på slutten av hver sprint. Dette gjør at det er enklere å være oppdatert og øker sjansen for at kunden får ønsket produkt til avtalt deadline. Vi har derfor valgt å bruke Scrum som prosessmetode.

Som nevnt over er Scrum en smidig prosessmetode. Dette medfører fordeler som jevn kontakt med kunden i utviklingsprosessen. På den måten er det mulig med tilbakemelding underveis, og dette er gunstig siden det ofte er avvik mellom hva kunden beskriver i kravspesifikasjonene, og hva kunden faktisk ønsker seg. Raske sprinter med klart definerte oppgaver fører til at arbeidstimene blir godt utnyttet siden alle vet hvilke oppgaver som skal gjøres til enhver tid. Samtidig gir høy kundeinvolvering som regel mer fornøyde kunder, siden de har mer innflytelse i utviklingsprosessen. Regelmessige Scrum-møter fører også til at alle på gruppa må være involverte. En annen fordel er at dersom prosjektgruppa er bra organisert, vil det være tydelig hva som må gjøres og når. Dette kan også være ulempen med Scrum, da den høye graden av selvstyring kan føre til at ting ikke blir gjort, eller at oppgaver (i produktkøen) er dårlig definert osv. Dette krever disiplin og et kollektivt ansvar hos gruppa, noe Scrum Master må legge til rette for.

2 PROSJEKTORGANISERING

2.1 ANSVARSOMRÅDER

Scrum legger opp til at man skal kunne dele opp oppgaver fortløpende i hver sprint. Derfor vil vi ikke ha noen streng entydig arbeidsfordeling på systemutviklingsoppgavene utover semesteret. Med Scrum er det naturlig at alle gruppemedlemmene skal ha en innvirkning på alle oppgavene. Det er ikke meningen at man skal ha en prosjektleder som bestemmer over gruppa, en som gjør alt av database, en som bare skriver rapport, og en som bare gjør GUI. Legg merke til at det er et viktig skille mellom prosjektleder og Scrum Master, der disse to har ulike oppgaver. Scrum Master sin jobb er å legge opp til at gruppa følger Scrum, han skal ikke ta noen sjefsavgjørelser da alle i gruppa har like stort ansvar.

Når det er sagt har vi endt opp med en viss rolleinndeling. Alle medlemmer av prosjektet har fått et ansvarsområde etter forkunnskaper og interesser. Oppgaven blir da å passe på at oppgavene innenfor deres ansvarsområde blir gjennomført, slik at det sikrer fremgang i prosjektet og unngår misforståelser.

Jørgen Halvorsen

- Scrum Master
- Møteromreservasjon
- Overvåke GUI-oppgaver

Mats Haraldsen

- Overvåke kodeoppgaver
- Oversikt på notater

Aleksander Hansen

- Overvåke databaseoppgaver
- LaTex-ansvarlig

Thomas Hansen

- Javadoc
- Overvåke tester

Ruben Hansen

- Referat-ansvarlig
- Overvåke rapporten

2.1.1 MEDLEMMENES KUNNSKAPSOMRÅDER

Kunnskapsområdene på gruppa er ganske like siden fire av studentene har stort sett hatt samme studieløp så langt. Likevel vil det nok finnes forskjellige erfaringer slik at medlemmene har mulighet til lære av hverandre og finne områder de ønsker å jobbe mer med. Vi har valgt å liste opp kunnskapsområder som er relevante til prosjektet.

Jørgen Halvorsen

- Java
- Java Swing
- Python

Mats Haraldsen

- Java
- Python

Aleksander Hansen

- Java
- LaTex
- Python

Thomas Hansen

- Java
- Python

Ruben Hansen

- Java
- Python

3 PROSESSBESKRIVELSE

3.1 BESKRIVELSE

Scrum er en smidig systemutviklingsmodell som er mye brukt i små og mellomstore programvareutviklingsprosjekter. I et utviklingsprosjekt kan det være vanskelig å vite alle kravene til en løsning før man begynner å implementere dem. Kravene til systemet kan endre seg underveis etterhvert som innblikk og erfaringer gjør at både kunden og utvikler endrer oppfatning av hvordan programmet skal være. I tillegg er det sjeldent at kunden klarer å gi en perfekt beskrive hva de egentlig er ute etter fra begynnelsen. Dette fører til at kravene vanligvis endrer seg underveis.

Scrum følger derfor en iterativ inkrementell prosess der man prøver å lage noe fungerende så fort som mulig, på den måten kan kunden se resultatet og raskt komme med eventuelle endringer som kan føre til at man må gjøre noe annerledes. På denne måten sørger man for at man ender opp på riktig spor med noe kunden faktisk vil ha.

Hver av disse inkrementene kalles sprinter. Etter hver fullførte sprint har utviklingsteamet et møte med kunden. Da blir det vist hva som er gjort og kunden gir tilbakemeldinger på hva de synes så langt. Det fører som regel til et bedre produkt for kunden, når de hele tiden er oppdatert på hvordan det går. Det blir også lettere for utviklingsteamet å holde seg på rett spor. På denne måten er man sikker på at man lager noe kunden ønsker seg gjennom hele prosessen og man vil ikke bruke tid på løsninger som ikke skal være med.

3.2 SPRINT

Vi har møter på mandager og onsdager og har sprinter på ca. 14 dager, ettersom om vi kommer i mål med sprinten til mandagen eller onsdagen. Hvert møte starter med “stand ups” der vi forteller hverandre hva vi har gjort siden sist og om vi har støtt på noen problemer. Videre planegger vi hva vi skal gjøre til neste møte.

På begynnelsen av hver sprint har vi sprintplanleggingsmøte der vi avgjør hva vi skal gjøre i den neste sprinten. Vi lager da en sprintkø i Trello som består av brukerhistorier brutt ned i tasks (oppgaver) slik at vi har en klar liste med oppgaver som må løses for at brukerhistoriene skal bli utført. Hver av disse oppgavene blir gitt et tidsestimat og prioritet.

Etter hver fullførte sprint har vi “sprint review meeting” med studass (som er vår kunde under dette prosjektet) der vi gjennomgår arbeidet som har blitt gjort, eventuelt det som ikke har blitt gjort, og får tilbakemelding fra studass. Etter dette har gruppa et “sprint retrospective meeting” uten studass, der vi går igjennom hva som gikk bra med sprinten, og hva som gikk dårlig med sprinten slik at vi kan forbedre oss på de aktuelle punktene til neste sprint.

3.3 ARBEIDSFORDELING

I Scrum er det slik at alle team-medlemmene har ansvar for fremdriften i prosjektet, det er ingen sjef. Vi har hatt en jevn arbeidsfordeling der medlemmer har valgt ut ansvarsområder og oppgaver etter evne og vilje. Disse ansvarsområdene er beskrevet lenger opp, under prosjektorganisering. For å holde oversikt over arbeidstimer og for å sikre en jevn arbeidsfordeling har vi ført timelister på Google Drive slik at vi har oversikt over antall timer alle på gruppa jobber. Ref til timeliste i vedlegg?

3.4 KILDEKODEDELING

Vi har brukt versjonskontrollsystemet Git for å dele kildekode mellom gruppemedlemmene, og tjenesten GitHub for å hoste vårt git-repository. Når man bruker Git må man passe seg for å ikke få merge-konflikter, dette kan man enkelt unngå om alle holder seg oppdatert. GitHub gir oss en god oversikt over endringer som blir gjort og hvem som har gjort dem. Ved commits passes det på at medlemmer skriver en god beskrivelse av hva de har endret slik at det blir enda mer oversiktlig. Dette gjør det enkelt å holde følge med hvor langt vi har kommet på de forskjellige oppgavene og brukerhistoriene.

4 TIDSESTIMERING

Når vi begynte på prosjektet definerte vi først brukerhistorier som beskrev funkjonaliteten programmet skulle ha. Deretter brøt vi disse opp i flere konkrete oppgaver som vi tildelte en prioritering og et tidsestimat. Oppgavene ble lagt ut på Trello der vi alle hadde mulighet til å merke oppgaver som ferdig etterhvert som de ble gjort, eller endre dem om uforutsette komplikasjoner førte til høyere tidsestimering eller at oppgaven måtte deles opp i flere deler. Dette passet godt med en smidig metode som Scrum siden det ga oss muligheten til å være fleksible etterhvert som endringer og problemer dukket opp.

For å holde oversikt over alt annet enn sprintkøen slik som prosjektorganisering, læringsprosess, rapportskriving brukte vi Google Drive. Vi valgte å bruke Trello og Google Drive for å holde en overordnet oversikt over hva som måtte gjøres. Vi lagde også en work breakdown structure (WBS) for å holde oversikt i tillegg til Trello og Google Drive.

4.1 ESTIMERING I SCRUM

Med Scrum er det ikke meningen at man skal estimere prosjektet fra start til slutt. Det er lagt opp til at man heller skal estimere hver sprint underveis. Dette gjør at man er fleksibel og kan lettere ha realistiske estimeringer. I Scrum er det ikke vanlig å estimere i tid, men heller "effort points" som mål på hvor krevende en oppgave er. Vi har valgt å estimere både i tid og prioritet

fra 1-10. Dette gjorde at vi kunne se hvilke oppgaver som var viktigst for prosjektet og samtidig holde en god tidsoversikt på arbeidet i forhold til timelister og fordeling av arbeid på gruppa.

4.2 ARBEIDSTIMER

Gruppa har hatt møter minst to ganger i uka der det først er et Scrum-møte, og deretter satt av to timer for å jobbe sammen med prosjektet. På den måten har det blitt et minimum av fire timer prosjektarbeid på hvert gruppemedlem hver uke i tillegg til arbeid på egenhånd. Siden programvareutviklingsprosjekter fort overskrider tiden på grunn av uforutsette hendelser, har vi tatt hensyn til dette da vi har valgt arbeidsoppgaver utenfor felles prosjektarbeid på skolen slik at vi ikke overskrider tiden som vi har til rådighet hver uke. Av de ti timene som er planlagt til bruk på prosjektarbeid hver uke har vi derfor satt av to timer til uforutsette hendelser og problemer. Alle timer utenfor felles arbeid på skolen skal være ført opp flittig i timelisten av gruppemedlemmer. Se timeliste i vedlegget.

5 RISIKO I PROSJEKTET

I et prosjekt som dette finnes det gode muligheter for at ting ikke går akkurat som planlagt. I vår risikomatrise har vi prøvd å forutse så godt som mulig de forskjellige problemene som kan oppstå. Blant disse er tap av data, overskridelse av sprint deadline, sykdom, dårlig oppmøte osv. For å unngå problemer som kan være en sinke for arbeidet har vi satt opp tiltak for å håndtere problemene når de eventuelt dukker opp. På den måten kan gruppa om ikke unngå problemer helt, i alle fall minske skaden så mye som mulig. Se risikomatrisa i vedlegget.

6 ARKITEKTURFORKLARING

6.1 HOVEDKOMPONENTER

Systemet vårt består et klientprogram som kommuniserer direkte med en database. Det var ingen spesifikke krav til arkitekturen og ingen av oss hadde erfaring med databaser eller systemarkitektur på forhånd. Derfor valgte vi å implementere systemet på denne måten, fremfor å implementere det som et klient/server-system, som ville vært mer omfattende å implementere. Med denne enkle arkitekturen kunne vi fokusere på å implementere annen funksjonalitet som hadde høyere prioritet. Vi har laget flere sekvensdiagrammer som illustrerer hvordan programmet oppfører seg når det får en ”request”. Se vedlegg for sekvensdiagtam og Deployment diagram.

6.2 KLIENTPROGRAM (PC)

Klientprogrammet er applikasjonen den vanlige brukeren og administrator bruker for å få tak i all informasjon i systemet. Programmet er det samme for administrator og vanlig bruker, men med forskjellig tilgang og bruksområder. Det grafiske brukergrensesnittet ligger i dette programmet, der brukeren kan reservere koie og hente relevant informasjon om koiene. Administrator kan endre status i/på de forskjellige koiene, samt fjerne reservasjoner. Programmet kan kjøres på alle datamaskiner som har Java installert og har en internetttilkobling. I utgangspunktet støtter systemet kun en bruker av gangen, men ved interaksjon i programmet, vil programmet hente nyeste data fra databasen noe som medfører at flere brukere kan være på systemet samtidig. Hele systemet er koblet mot samme database og henter data i sanntid når brukeren “gjør” noe.

6.3 DATABASE (SERVER)

6.3.1 DATABASETEKNOLOGI

Databasen ble implementert som en MySQL-database på studentområdet vårt på serverene til NTNU. MySQL var allerede installert på serverne, og en veiledning var tilgjengelig for hvordan man enkelt kunne sette opp en database.

Vi designet databasestrukturen slik at vi skal kunne representere all informasjon knyttet til systemet som en relasjonsdatabase. Vi brukte MySQLs standard lagringsmotor MyISAM for å håndtere kommunikasjonen med databasen, men denne lagringsmotoren støtter ikke fremmednøkler. Ingen av oss hadde erfaring med MySQL eller andre databaser fra før av, så vi visste ikke hvordan vi kunne endre dette. Dette betyr at relasjoner ikke representeres eksplisitt i databasen, men må opprettholdes av programmet.

6.3.2 DATABASESTRUKTUR

Vi har laget et ER-diagram for å visualisere databasestrukturen, se vedlegg. Inngående forklaring av tabellen finnes under.

Tabellforklaring

I tabellen **bruker** lagrer vi alle brukerne i systemet. En bruker kan være enten en vanlig bruker av koienettet eller en administrator, og hvorvidt brukeren er en vanlig bruker eller en administrator lagres i tabellen. En bruker i systemet har et brukernavn, som for en vanlig bruker må være en epostadresse, og som for en administrator må være en tekststreng som ikke kan tolkes som en epostadresse. Dette brukernavnet brukes som nøkkel i denne tabellen. Hver bruker har også et passord for innlogging, og dette lagres av grunnleggende sikkerhetshensyn som en MD5-hash.

I tabellen **koie** lagrer vi informasjon om koiene som ikke endrer seg under normal bruk av programmet, som for eksempel navn, antall sengeplasser og antall sitteplasser. Dette er attributter ved en koie som kan være fint brukerne av programmet å se, men som ikke er kritisk for funksjonaliteten til systemet. Hver koie har en unik numerisk ID som nøkkel, som vi bruker i andre tabeller i databasen når vi skal referere til en koie.

I tabellen **item** lagres alt utstyr i systemet. Hver ting har en unik numerisk ID som nøkkel. En ting har også et navn og en status som enten i orden, ødelagt eller gjenglemt. Det er også en referanse til den unike numeriske ID-en til koia som tingen tilhører. Dette er for våre formål en fremmednøkkel, selv om lagringsmotoren vi bruker ikke støtter ekte fremmednøkler.

I tabellen **vedstatus** lagrer vi hvor mye ved som er i vedlageret til hver koie. Dette gjør vi ved å lagre en koie-ID (som både nøkkel og fremmednøkkel) og en vedmengde. Siden vi allerede har en koie-tabell, ville det vært lettere å bare legge til et felt i den tabellen. Men vi ønsket å holde statiske koie-attributter i koie-tabellen adskilt fra et koie-attributt som vedstatus, som ville endres over tid.

I tabellen **reservasjon** holder vi styr på alle reservasjonene som gjøres. Hver reservasjon har en unik numerisk ID som nøkkel, en koie-ID som fremmednøkkel som kobler reservasjonen til en koie. Det er også et fremmednøkkel for brukernavn slik at man vet hvem som har reservert koia. Reservasjonsperioden representeres med startdato og sluttdato.

I tabellen **rapport** lagres hvilke brukere som har sendt en rapport om ei koie etter et besøk. Feltene i tabellen er fremmednøkler til koie-ID, reservasjons-ID og brukernavn. Vi valgte å lagre dette i en egen tabell for å senere kunne utvide med kommentar eller andre attributter.

7 PRODUKTBESKRIVELSE

Produktet vårt er et programvaretillegg til nettsidene om NTNU-koiene. Produktet tar i hovedsak for seg rapportering av utstyr på de forskjellige koiene. Det er også mulighet til å reservere en koie, man kan se når koiene er ledige og legge inn en reservasjon. Hvis man har en reservasjon kan man videre få beskjed om det skulle være noen mangler av utstyr eller ved.

8 SYSTEM DESIGN

I et system slik som vi har laget er det naturlig å gi brukeren så mye informasjon som mulig på en ryddig og oversiktlig måte. Vi har valgt å lage et lite og enkelt brukergrensesnitt for å formidle denne informasjonen på en enkel og brukervennlig måte. Vi valgte å kjøre programmet i ett vindu og dele opp informasjonen i flere logiske faner med forskjellig informasjon og funksjonalitet. På denne måten får vi oversiktlig vist frem all relevant informasjon delt inn i faner. Dette er en mye bedre løsning enn et stort rotete vindu med for mye informasjon stappet inn på samme plass. Faner gjør at vi kan slipper rot og unødvendige store dimensjoner. Faner har vært tatt i bruk av andre systemer og er en intuitiv og naturlig del av mange forskjellige programmer.

Man navigerer seg gjennom systemet ved hjelp av knapper som er tydelig merket etter hvilken funksjon de har. Det første vinduet gir brukeren muligheten til å logge inn med en eksisterende bruker eller lage en ny bruker. Det er to valg av brukere, man kan enten være en administrator eller en vanlig bruker. For å registrere seg som vanlig bruker må man fylle in epostadresse og velge et passord, det opprettes en bruker og man kan logge inn. Vanlig brukere har tilgang på tre faner, disse er innlogging, reservasjoner, og reserver koie. Innloggingsfanen gir deg muligheten til å logge ut etter at man er logget inn. Reservasjoner lister opp alle reserveringer din bruker har laget. Reserver koie fanen gir deg mulighet til å velge hvilken koie du vil reservere, dato og lengde på reservasjon og om den valgte koia er ledig i det tidsrommet. Denne fanen har også en oversikt over hvilke fasiliteter og utstyr koiene har. Som administratorbruker får man tilgang på litt mer informasjon. Administrator har tilgang på syv forskjellig faner, der hver fane representerer en gitt funksjon. Disse holder styr på utstyrstatus, vedstatus, kart med administrativ informasjon, sending av mail til brukerne, og en fane som gir muligheten til å fjerne reservasjoner. Administrators fane med reservasjoner skiller seg også fra en vanlig bruker sin siden administartor ser alle reservasjoner fra alle brukere.

Design og fargevalg av det grafiske brukergrensesnittet ble gjort med tanke på at det skulle ha enkle nøytrale farger som ikke skilte seg ut. Den har et tidløst og gjennkjennelig design, med grå bakgrunn og gråblå knapper og kant. Dette gir programmet en trygg og beroligende effekt på brukeren siden det er et enkelt og ufremmed design.

9 VEDLEGG

9.1 DIAGRAMMER

9.1.1 WBS-DIAGRAM

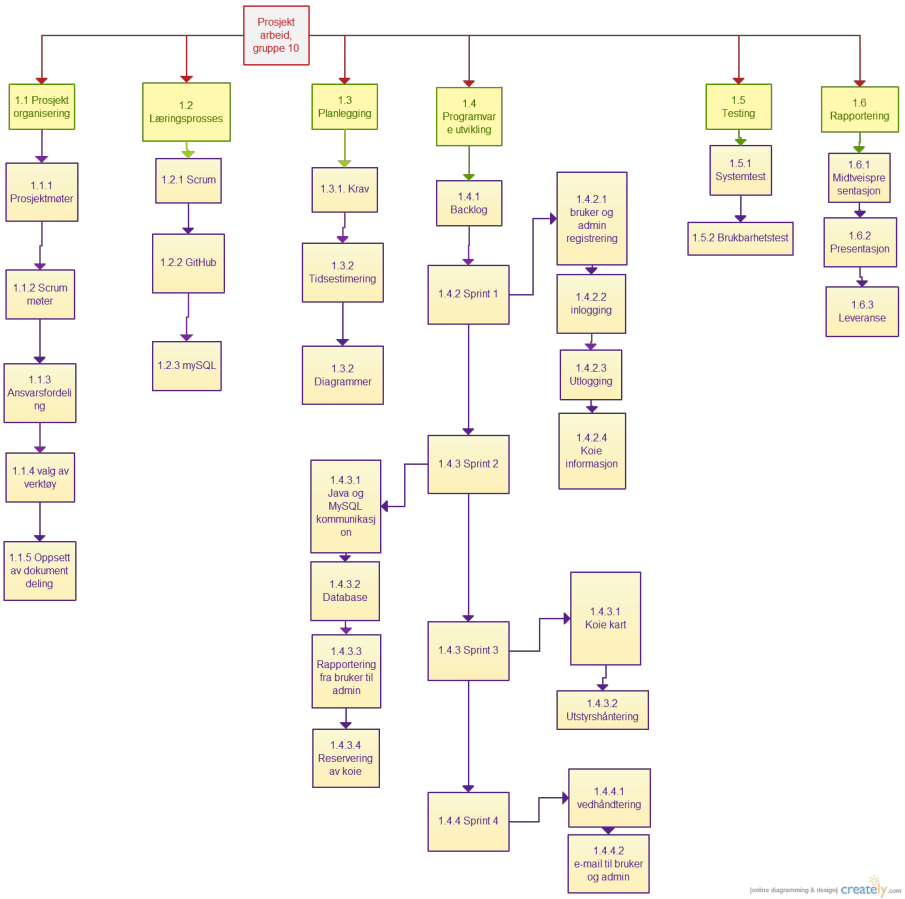
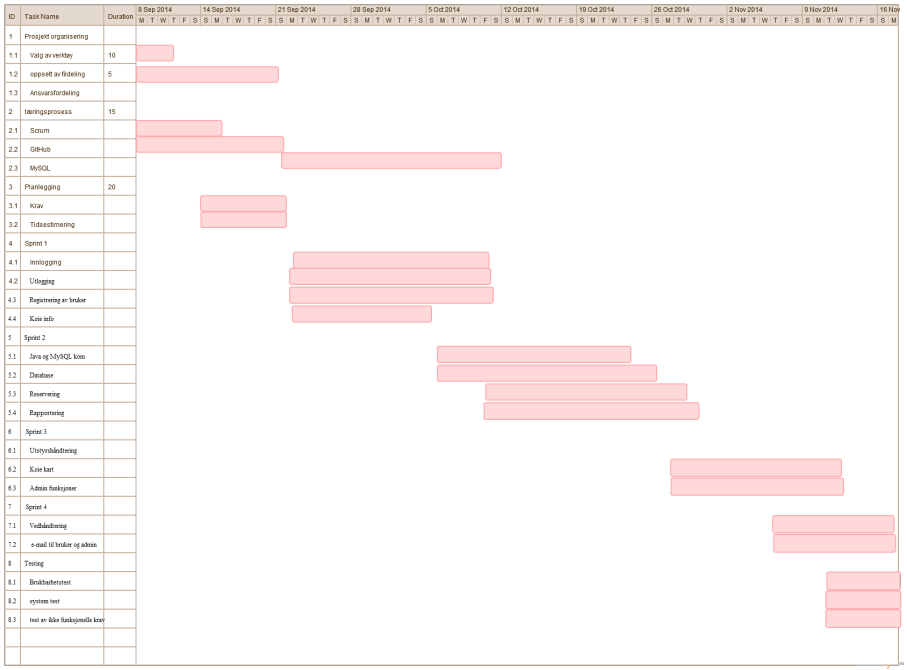


FIGURE 1: WBS

9.1.2 GANTT-DIAGRAM



ID	Task Name	Duration
1	Prosjekt organisering	
1.1	Valg av verktøy	10
1.2	oppsett av fildeling	5
1.3	Ansvarsfordeling	
2	læringsprosess	15
2.1	Scrum	
2.2	GitHub	
2.3	MySQL	
3	Planlegging	20
3.1	Krav	
3.2	Tidsestimering	
4	Sprint 1	
4.1	Innlogging	
4.2	Utlogging	
4.3	Registrering av bruker	
4.4	Koie info	
5	Sprint 2	
5.1	Java og MySQL kom	
5.2	Database	
5.3	Reservering	
5.4	Rapportering	
6	Sprint 3	
6.1	Utstyrshåndtering	
6.2	Koie kart	
6.3	Admin funksjoner	
7	Sprint 4	
7.1	Vedlikehtering	
7.2	e-mail til bruker og admin	
8	Testing	
8.1	Brukbarhetstest	
8.2	system test	
8.3	test av ikke funksjonelle krav	

FIGURE 2: GANTT

9.1.3 TIMELISTE

	Dato	Jørgen	Aleksander	Thomas	Mats	Ruben	SUM
Sprint 1	02.09.2014	1			1	1	3
	03.09.2014						0
	04.09.2014	1		1	1		3
	05.09.2014						0
	06.09.2014						0
	07.09.2014						0
	08.09.2014	2	2	2	2	2	10
	09.09.2014						0
	10.09.2014	2	2	2	2	2	10
	11.09.2014						0
	12.09.2014						0
	13.09.2014						0
	14.09.2014						0
	15.09.2014	2		2		2	6
	16.09.2014						0
	17.09.2014	2	2	2	2		8
	18.09.2014						0
	19.09.2014					1	1
	20.09.2014		3				3
	21.09.2014	2					2
Sum sprint 1		**12**	**9**	**9**	**8**	**8**	**46**

FIGURE 3: TIMELISTE SPRINT 1

	Dato	Jørgen	Aleksander	Thomas	Mats	Ruben	SUM
Sum sprint 1		**12**	**9**	**9**	**8**	**8**	**46**
Sprint 2	22.09.2014	2	2	2		2	8
	23.09.2014						0
	24.09.2014	2	2	2	2	2	10
	25.09.2014						0
	26.09.2014						0
	27.09.2014						0
	28.09.2014						0
	29.09.2014	2	2	2			6
	30.09.2014	2					2
	01.10.2014	2	2	2	2	2	10
	02.10.2014						0
	03.10.2014				2	1	3
	04.10.2014			2	2	2	6
	05.10.2014		1				1
Sum sprint 2		**10**	**9**	**10**	**8**	**9**	**46**

FIGURE 4: TIMELISTE SPRINT 2

	Dato	Jørgen	Aleksander	Thomas	Mats	Ruben	SUM
Sprint 3	06.10.2014	2	2		2	2	8
	07.10.2014				3		3
	08.10.2014	2		3	2	2	9
	09.10.2014						0
	10.10.2014						0
	11.10.2014						0
	12.10.2014						0
	13.10.2014	2	2	2	2	2	10
	14.10.2014						0
	15.10.2014	2	2	2	2	2	10
	16.10.2014						0
	17.10.2014						0
	18.10.2014						0
	19.10.2014						0
	20.10.2014		2	3			5
	21.10.2014						0
	22.10.2014	3	3	3	2	3	14
	23.10.2014						0
	24.10.2014						0
	25.10.2014						0
	26.10.2014						0
Sum sprint 3		**11**	**11**	**13**	**13**	**11**	**59**

FIGURE 5: TIMELISTE SPRINT 3

	Dato	Jørgen	Aleksander	Thomas	Mats	Ruben	SUM
Sprint 4	27.10.2014	2	2	2	2		8
	28.10.2014		2	3			5
	29.10.2014	3	1	2		3	9
	30.10.2014						0
	31.10.2014						0
	01.11.2014						0
	02.11.2014						0
	03.11.2014	2		8,5		3	13,5
	04.11.2014		2	2			4
	05.11.2014	3	4	3,5	4	2	16,5
	06.11.2014						0
	07.11.2014		3		3	3	9
	08.11.2014						0
	09.11.2014						0
	10.11.2014	2	2	5	2	2	13
	11.11.2014	5	5	1			11
	12.11.2014	9	7	9	3	4	32
	13.11.2014	6		5	3	2	16
	14.11.2014						0
	15.11.2014	7		7			14
	16.11.2014			9	4	4	17
	17.11.2014				3,5	3	6,5
Sum sprint 4		**39**	**28**	**57**	**24,5**	**26**	**174,5**

FIGURE 6: TIMELISTE SPRINT 4

9.1.4 RISIKOMATRISE

Beskrivelse	Sannsynlighet (1-9)	Konsekvens (1-9)	Viktighet	Preventative tiltak	Utbedringstilltak
Tap av data	3	8	24	Verktøy som github og google drive sikrer at data ikke går tapt i tillegg til at man lagrer lokalt.	Antall arbeidstimer økes for å ta igjen det tapte
Et uforesett problem hindrer gruppa i å jobbe videre	5	6	30	Ha gjennomførbare løsninger og ungå flaskehalser	Spør om hjelp fra studass eventuelt orakel tjenesten eller lignende
Skjev arbeidsfordeling	4	5	20	Prøv å fordele like store og like mange oppgaver til alle i gruppa	Fordel arbeidsoppgaver på nytt
Et eller flere gruppemedlemmer er borte over lenger tid	3	3	9	Meld fra til gruppa dersom man planlegger å være borte over lenger tid	Tildel arbeidsoppgaver som gruppemedlemmet må gjøre mens han er borte
Gruppemedlemmer kommer for sent til møte	9	4	36	Faste avtaler på hvor og når vi skal møtes hver uke slik at man ikke glemmer eller møter opp på feil rom	Gruppemedlemmet eller gruppa må jobbe lenger for å ta igjen det tapte
Sprint deadline ikke møtt	4	6	24	Planlegg gjennomførbare sprinter og beregn ekstra tid til eventuelle problemer	Jobb ekstra for å ta igjen det tapte
Et eller flere medlemmer gjør ikke sine arbeidsoppgaver	2	8	16	God planlegging og kommunikasjon i tillegg til verktøy som gir oversikt over arbeidsoppgaver gjør det enklere.	Gruppemedlemmet må jobbe ekstra. Om nødvendig hjelper de andre til

FIGURE 7: RISIKOMATRISE

9.1.5 DEPLOYMENT-DIAGRAM

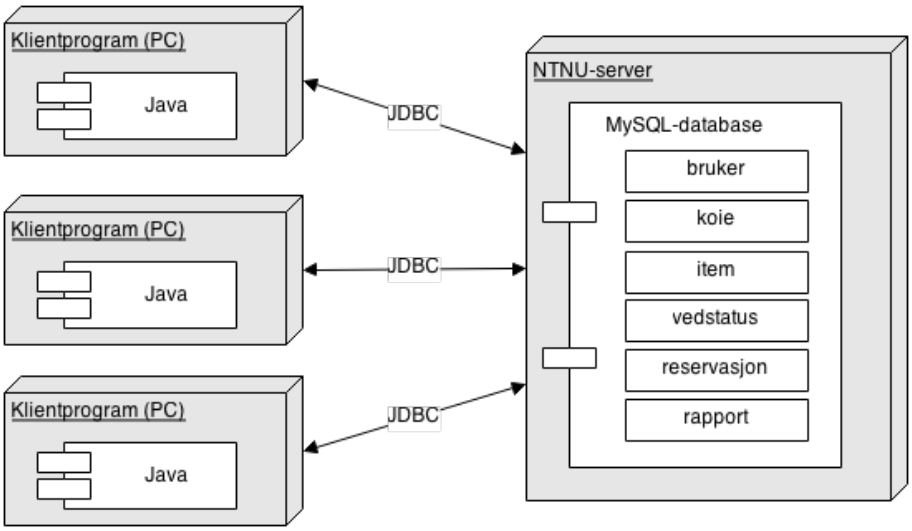


FIGURE 8: DEPLOYMENT-DIAGRAM

9.1.6 ENTITY-RELATIONSHIP DIAGRAM

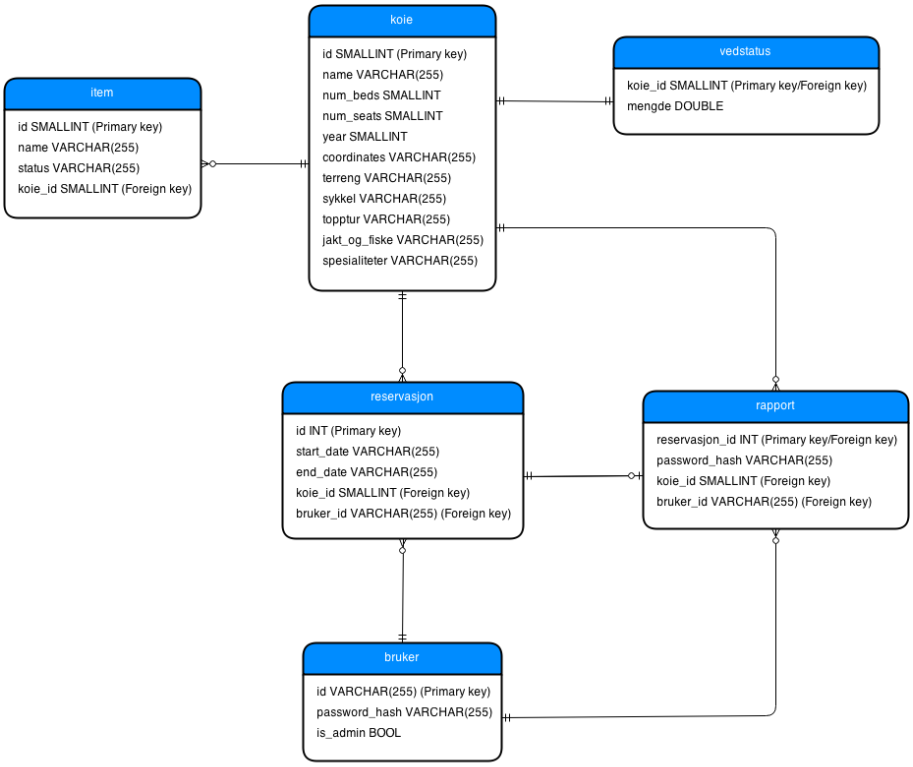
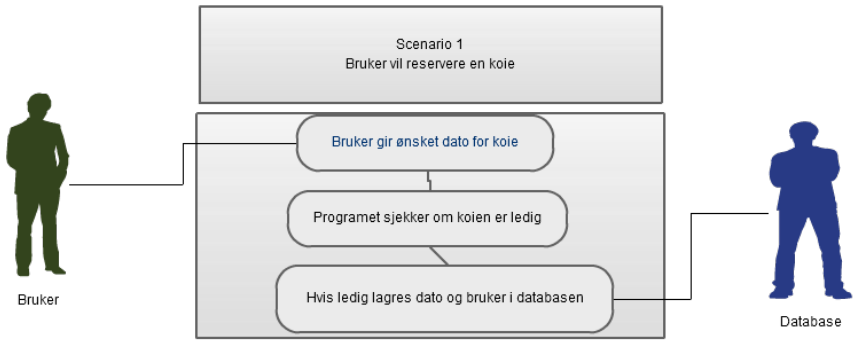
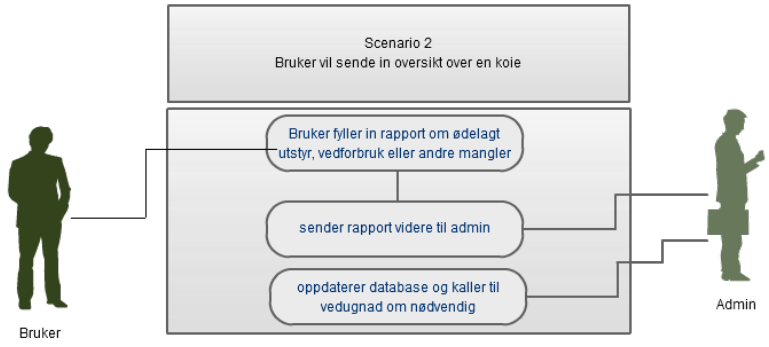
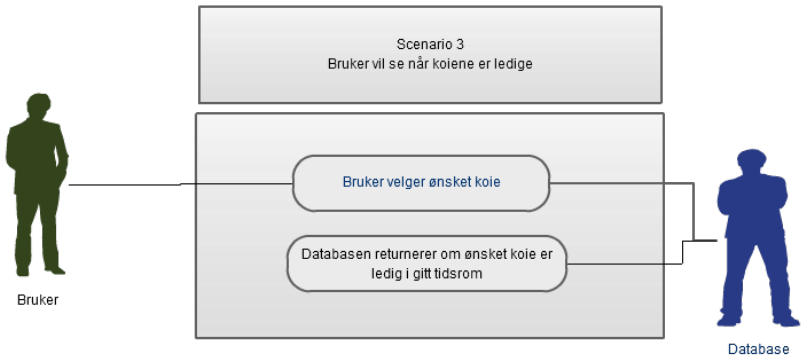


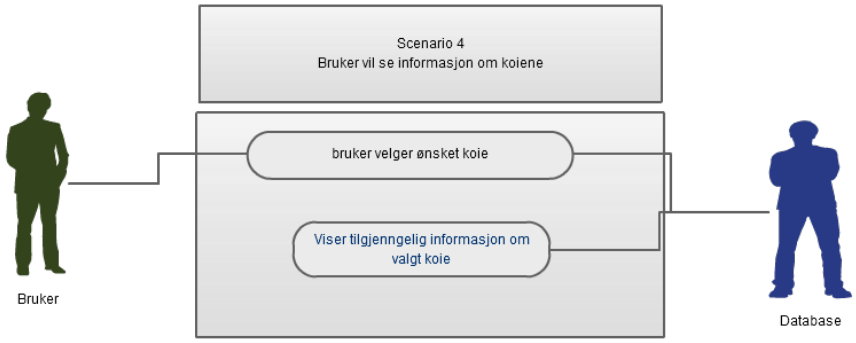
FIGURE 9: ER-DIAGRAM

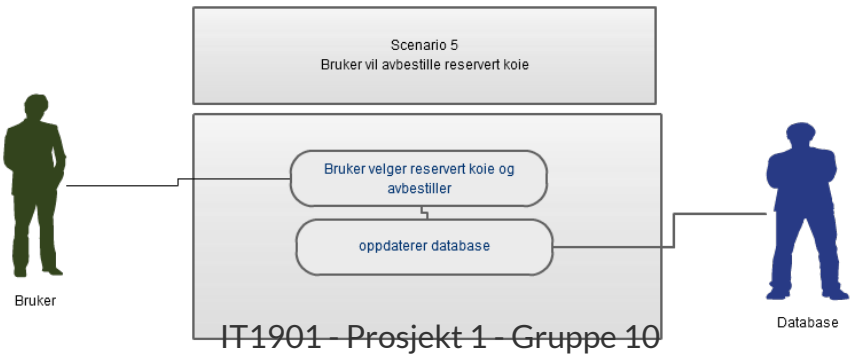
9.1.7 USE CASE-DIAGRAM

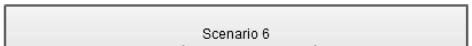












9.1.8 SEKVENSDIAGRAM

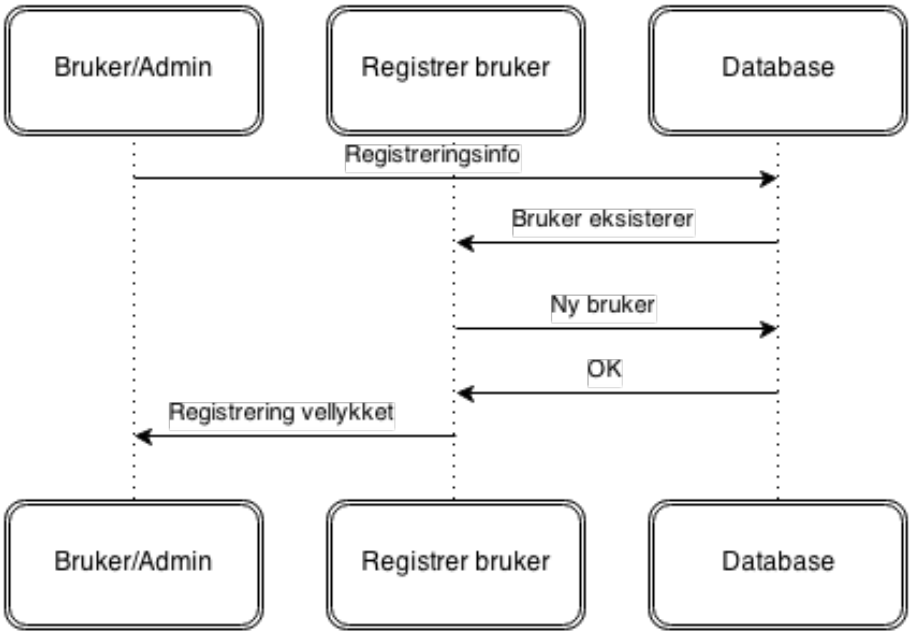


FIGURE 11: REGISTRER BRUKER

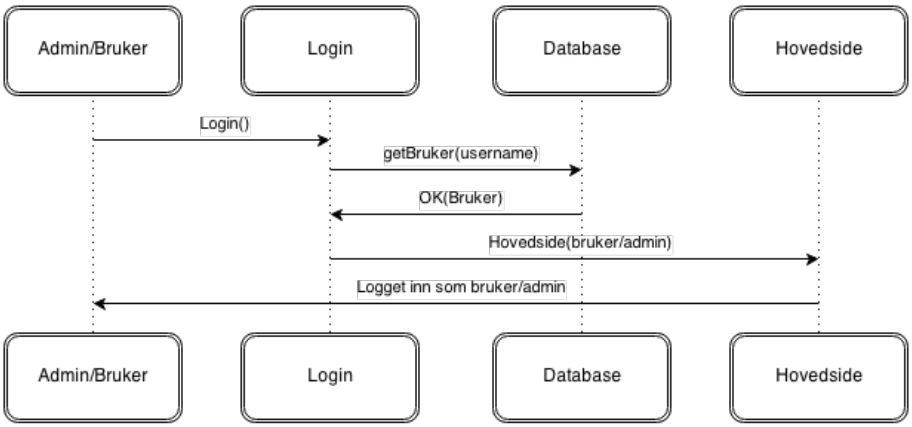


FIGURE 12: LOGG INN

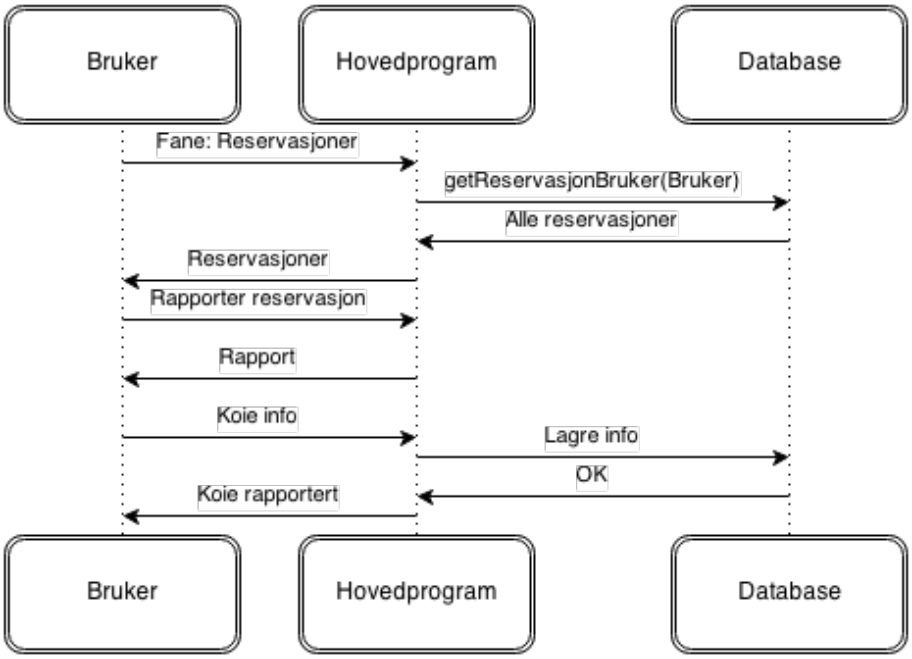


FIGURE 13: BRUKER: RAPPORTER

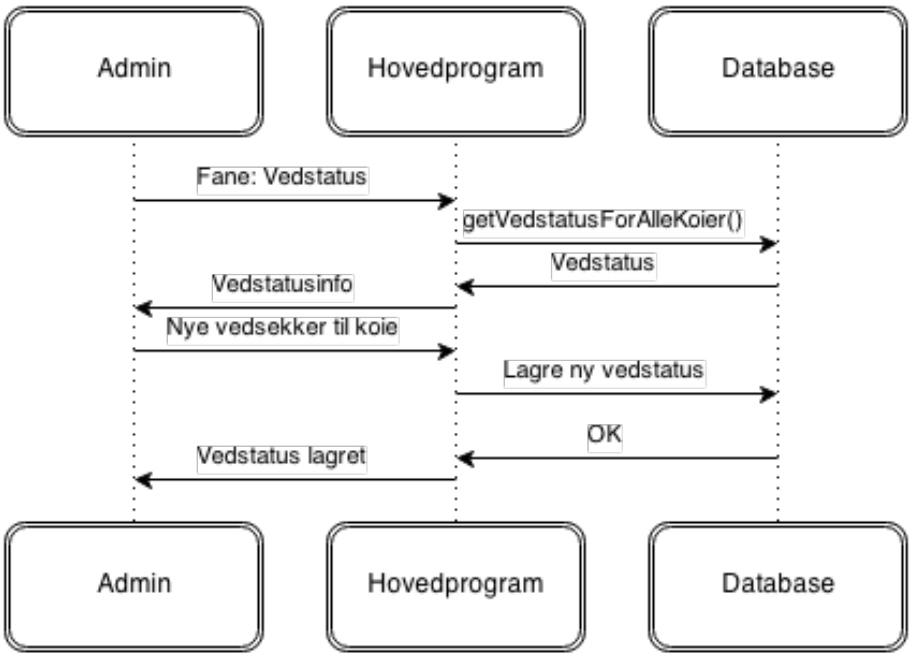


FIGURE 14: ADMININSTRATOR: SE VEDSTATUS

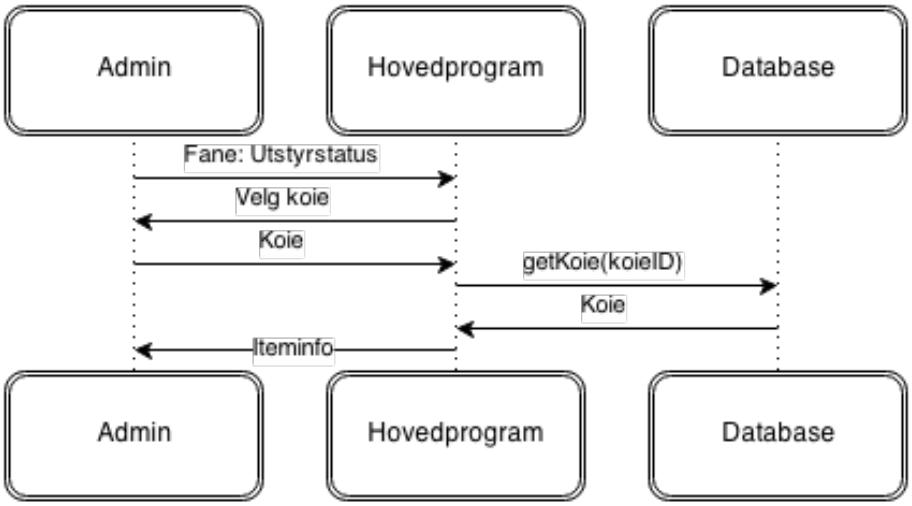


FIGURE 15: ADMININSTRATOR: SE UTSTYRSSTATUS

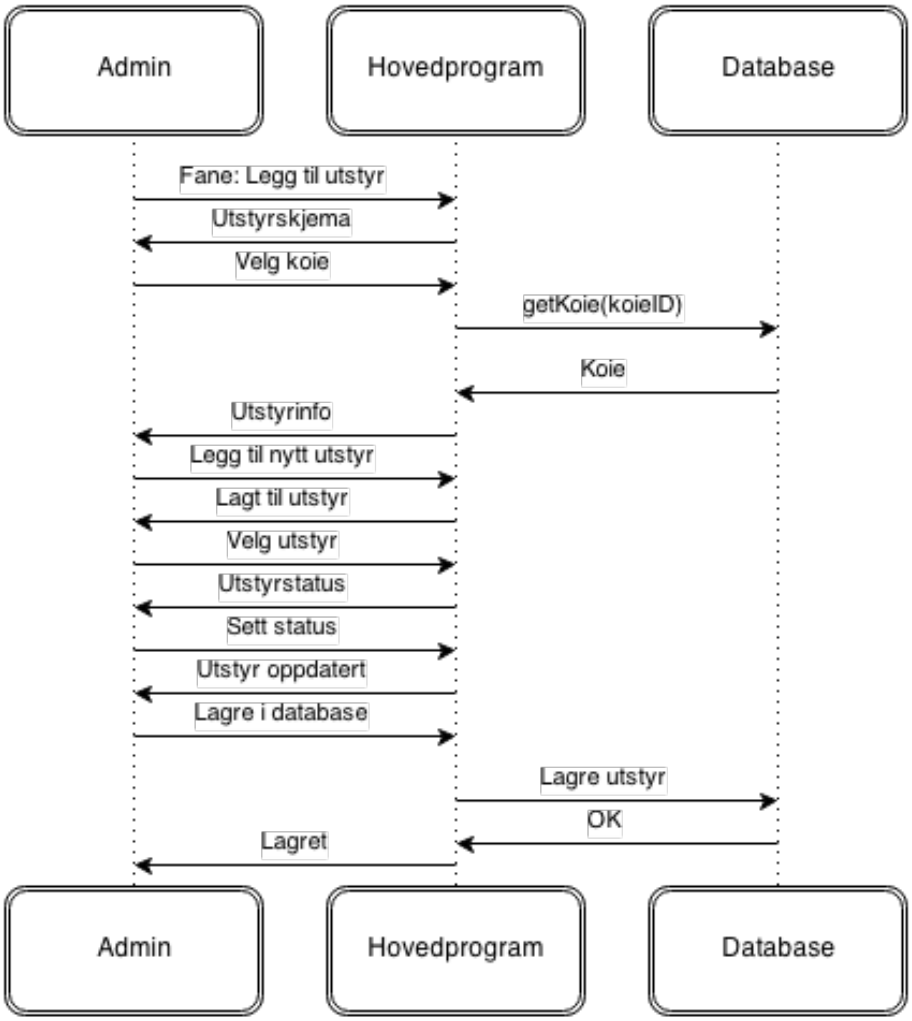


FIGURE 16: ADMININSTRATOR: LEGG TIL UTSTYR

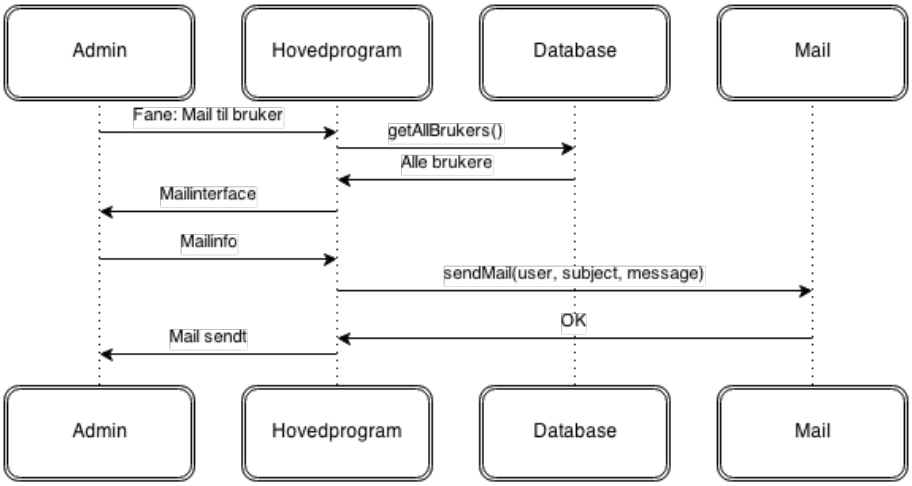


FIGURE 17: ADMININSTRATOR: SEND MAIL TIL BRUKER

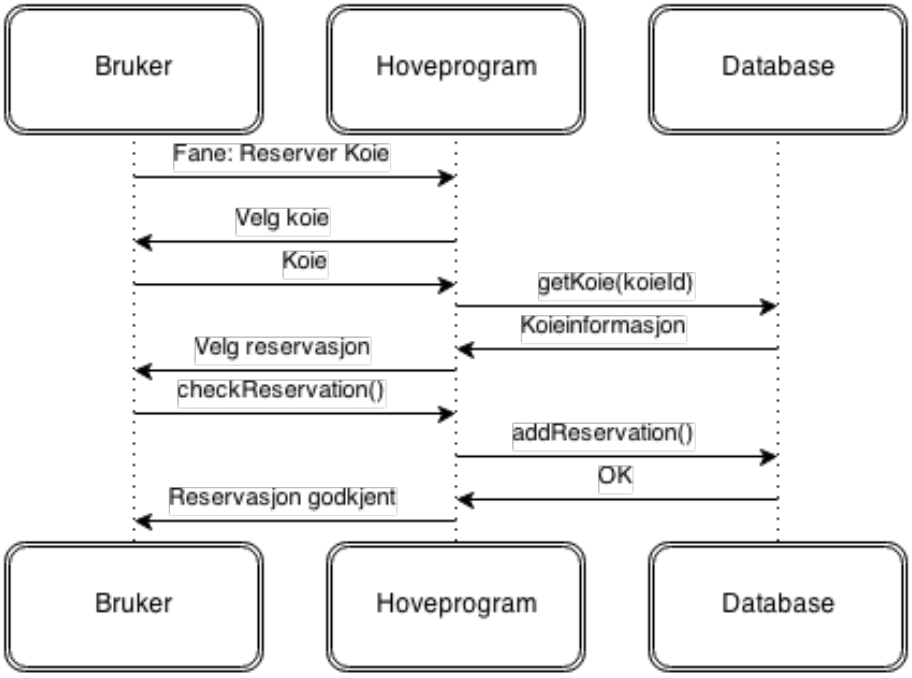


FIGURE 18: BRUKER: RESERVER KOIE

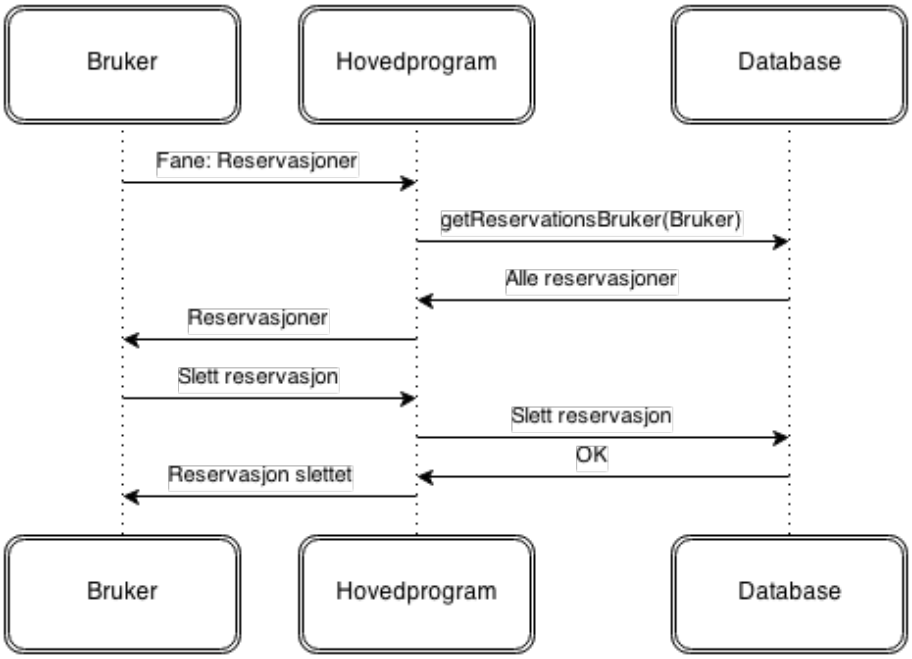


FIGURE 19: BRUKER: SLETT RESERVASJON

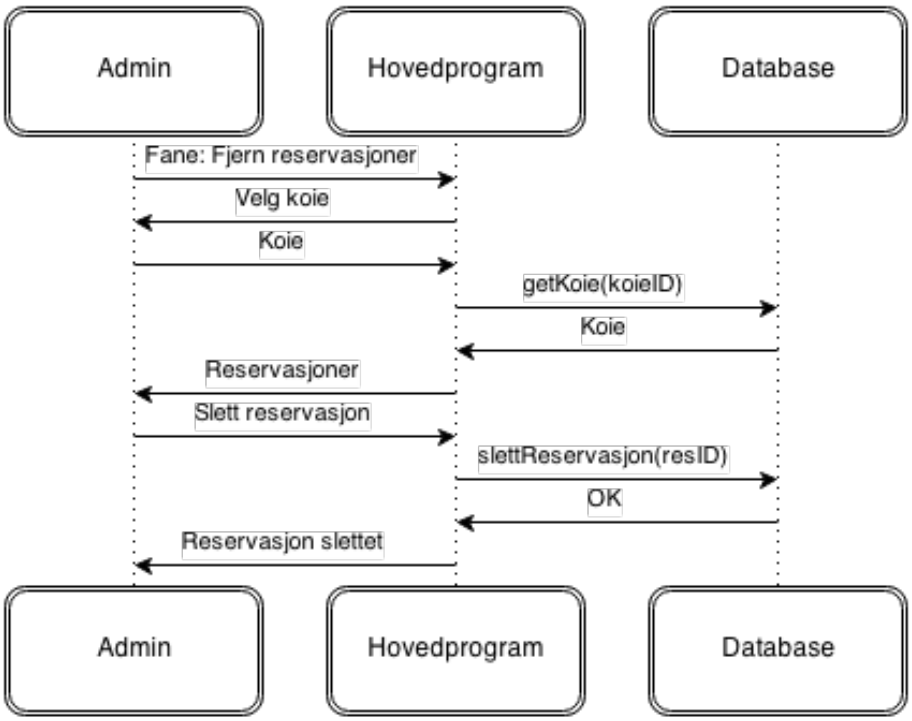


FIGURE 20: ADMININSTRATOR: SLETT RESERVASJON

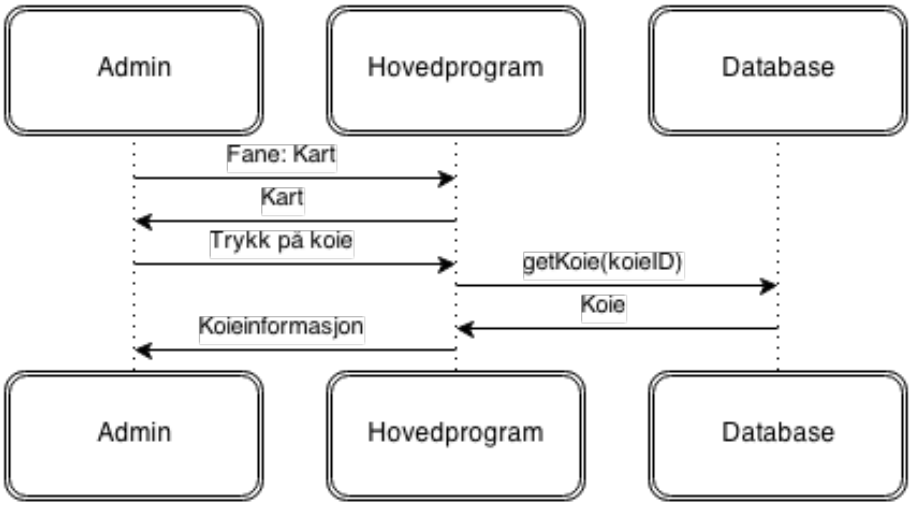


FIGURE 21: ADMININSTRATOR: SE KART

9.2 KRAVSPESIFIKASJON

9.2.1 FUNKSJONELLE KRAV

Vi har valgt å formulere de funksjonelle kravene som brukerhistorier, og delt brukerhistoriene opp i brukerhistorier for vanlige brukere og brukerhistorier for administratorer.

TABLE 1: BRUKERHISTORIER FOR VANLIGE BRUKERE

Nr	Brukerhstorie	Prioritet (1-9)
1	Som bruker vil jeg kunne registrere meg som bruker i systemet.	9
2	Som bruker vil jeg kunne logge meg inn i systemet.	9
3	Som bruker vil jeg kunne logge meg ut av systemet.	9
4	Som bruker vil jeg kunne rapportere om ødelagte ting ved en koie.	8
5	Som bruker vil jeg kunne rapportere vedstatus ved en koie.	7
6	Som bruker vil jeg kunne rapportere om gjenglemte ting ved en koie.	6
7	Som webreservasjonssystem vil jeg kunne legge inn en reservasjon på formatet koie; epostadresse; dato.	5
8	Som bruker skal jeg kunne se når koiene er ledige.	3
9	Som bruker vil jeg kunne reservere en koie.	3
10	Som bruker vil jeg kunne kansellere en koie-reservasjon.	2
11	Som bruker vil jeg kunne se informasjon om koiene.	1

TABLE 2: BRUKERHISTORIER FOR ADMINISTRATORER

Nr	Brukerhistorie	Prioritet (1-9)
12	Som administrator vil jeg kunne registrere meg som administrator i systemet.	9
13	Som administrator vil jeg kunne logge meg inn i systemet.	9
14	Som administrator vil jeg kunne logge meg ut av systemet.	9
15	Som administrator vil jeg kunne se utstyrsstatus for én eller alle koier.	8
16	Som administrator vil jeg kunne legge inn nyinnkjøpt utstyr	8
17	Som administrator vil jeg kunne se vedstatus for én eller alle koier.	7
18	Som administrator vil jeg kunne få beskjed når det er lite ved igjen hos en koie.	7
19	Som administrator vil jeg kunne informere brukere som har reservert koie om at utstyr må fraktes til koia.	6
20	Som administrator vil jeg kunne se et kart over koiene og klikke på koiene for administrativ informasjon.	5
21	Som administrator vil jeg kunne fjerne utstyr.	4
22	Som administrator vil jeg kunne endre tilstanden på utstyr.	4
23	Som administrator vil jeg kunne legge ved til en koie.	4
24	Som administrator vil jeg kunne legge inn en reservasjon for en bruker.	3
25	Som administrator vil jeg kunne slette reservasjoner.	3
26	Som administrator vil jeg kunne resette databasen.	3

9.2.2 IKKE-FUNKSJONELLE KRAV

TABLE 3: IKKE-FUNKSJONELLE KRAV

Nr	Ikke-funksjonelt krav	Prioritet (1-9)
27	Systemet skal være operativt døgnkontinuerlig.	9
28	Systemet skal være tilgjengelig minst 95 % av tiden.	8
29	Systemet må håndtere alle koiene i koienettet.	7
30	Systemet må kunne analysere data fra koier for 6 måneder.	7
31	Ved systemfeil skal systemet ikke være utilgjengelig i mer enn 3 timer.	5
32	Responstid for administrator i systemet skal være maksimalt 0,5 sekund.	3

9.3 SPRINTER

9.3.1 SPRINT 1

Project Backlog Etter det første møte med studass fikk vi et litt bedre innblikk i Scrum og hvordan sprintene skulle fungere. Vi startet med å sette opp User Stories for hele prosjektet, der vi tolket kravene og rangerte de etter viktigheten de hadde i systemet. Dette utgjorde vår Project Backlog.

Sprint Backlog Når Project Bakloggen var ferdig valgte vi ut noen User Stories som skulle bli den første sprinten. Disse delte vi inn i mindre oppgaver og ga et tidsestimat på dem.

User Stories sprint 1:

- Bruker og admin registrering
- Inlogging
- Utlogging
- Koie informasjon

Sprint review møte Når sprinten var ferdig hadde vi et sprint review møte med studass. Vi viste hva vi hadde gjort og hvor langt vi hadde kommet på prosjektet. Videre fikk vi tilbakemeldinger og tips til hvordan vi kunne forbedre oss til neste gang.

Sprint retrospective Etter studass hadde sagt sitt, hadde vi et tilbakeblikk på den første sprinten. Det var noen problemer i starten. Oppsettet av GitHub for kildekodedeling fungerte ikke som vi ville med en gang. Vi måtte også sette oss inn i flere programmer for å få ting igang. Det førte til at ikke alle User Stories ble helt fullført. Vi konkluderte med at vi må sette av mer tid til uforutsette hendelser. Det ble estimert litt for få arbeidstimer til sprint 1.

9.3.2 SPRINT 2

Oppsettet på sprint 2 er så og si identisk med sprint 1, bestående av Sprint Backlog, Sprint review møte og Sprint retrospektiv. Vi velger nye User Stories:

User Stories sprint 2:

- Java og MySQL kommunikasjon
- Database
- Rapportering fra bruker til admin

- Reservering av koie

Denne gangen estimerte vi litt mer tid siden ingen på gruppa var kjent med databaseoppsett. Igjen støtte vi på noen små problemer. Blandt annet kommunikasjonen mellom Java-programmet og MySQL. Men under retrospektiv hadde vi en bedre tidsberegning enn ved sprint 1 og klarte å fullføre alle User Stories innenfor sprinten. Vi beregnet noen ekstra arbeidstimer i og med at vi måtte lære oss databaseoppsettet før vi kunne ta det i bruk.

9.3.3 SPRINT 3

Igjen satt vi opp nye User Stories. Denne gangen beregnet vi litt mindre arbeidsoppgaver på selve programmet fordi vi satte av ekstra tid til å komme i gang med rapporten.

User Stories sprint 3:

- Koie kart
- Utstyrshåndtering

Når sprinten var over hadde vi et nytt møte med studass. Han så gjennom programmet og påpekte noen små mangler. Disse manglene ble utgangspunktet for den fjerde og siste sprinten.

9.3.4 SPRINT 4

Sprint 4 tok for seg de siste kravene til programmet. Arbeidsmengden her er også litt mindre enn ved sprint 1 og 2 fordi vi estimerte litt ekstra tid på GUI og rapport skriving. Som vist på timeslistene ble det endel ekstra arbeidstimer for å finpusse på programmet og rapporten.

User Stories sprint 4:

- Vedhåndtering
- E-mail til bruker og admin

9.4 TESTARBEID

9.4.1 SYSTEMTEST

Vi utførte en systemtest hvor vi kontrollerte at alle funksjonelle og ikke-funksjonelle krav var oppfyllt av programmet.

Test av funksjonelle krav Vi gikk gjennom alle funksjonelle kr

Test av ikke-funksjonelle krav Sette inn time-checks for å måle responstid? La systemet stå på x antall døgn for å teste tilgjengelighet?

9.5 VERKTØY

9.5.1 PROSJEKTHÅNDTERING

Facebook (https://www.facebook.com/)
Siden alle medlemmene på gruppa allerede fantes på facebook, fant vi det naturlig å benytte facebook som kommunikasjonsplatform. E-mail og meldinger gjennom It's learning ble brukt litt i starten når vi skulle prøve å samle gruppa, men da de enten ble ignorert eller sett for sent fant vi ut at å lage en facebook gruppa var mer fornuftig og effektivt, da folk stort sett holder seg oppdatert på nye hendelser på facebook. Facebok gruppa ble brukt til å gi beskjed om hvor og når vi skulle ha møter, deling av nyttig informasjon og lenker, og om noen ikke kunne komme til møte eller andre problemer.

Google Drive (https://drive.google.com/)
Goolge Drive ble brukt til å dele dokumenter, timelister og andre filer. Google Drive har en stor fordel ved at det lar alle på gruppa jobbe på det samme dokumentet om det skulle være nødvendig. Samtidig som det er en lett tilgjengelig lagringsplatform som alle kan bruke. Om man er ordentlig uheldig kan det oppstå problemer og konflikter hvis flere jobber på samme dokument, men dette har ikke vært et problem.

ShareLaTeX (https://www.sharelatex.com/)
Vi valgte å skrive prosjektrapporten i LaTeX. LaTeX er et vertøy som ofte blir brukt i større rapporter og lignende. Det er lettere å opretthold et konsist og fint design gjennom hele rapporten. LaTeX kan blant annet oprette automatisk innholdsfortegnelse. Det gjør at vi kan fokusere mere på innholdet i rapporten og bruke mindre tid på designet.
Vi brukte det nettbaserte verktøyet https://www.sharelatex.com/ som gir muligheten til at flere kan jobbe på samme dokument samtidig. Det førte til en mer effektiv rapportskriving.

Trello (https://trello.com/)
Trello er en gratis nettbasert prosjekt forvalter applikasjon (project management application). Trello er som en stor opplsagstavle der man kan henge opp lapper med oppgaver. Disse lappene kan legges i lister med forskjellig status, nye oppgaver kan lett legges til, endres og flyttes etter behov. På den måten kan alle på gruppa ha en ryddig oversiktlig kontroll på hva som er gjort og hva som må gjøres.

GitHub (https://github.com/)
GitHub er en hosting-tjeneste for Git-repositorier, som vi hadde kildekoden vår på.

Git (https://git-scm.com/)

Git er en versjonskontrollsystem for programvareutvikling. Vi brukte Git til å dele på kildekoden mellom gruppemedlemene. Git er lett å bruke, men det kan forekomme merge konflikter om man ikke er forsiktig.

9.5.2 DIAGRAMVERKTØY

draw.io (http://www.draw.io/)
draw.io er et sky-basert diagram-verktøy som kan integreres med andre sky-baserte tjenester, som for eksempel Google Drive.

Creately (http://creately.com)
Creately er et sky-basert diagramverktøy som kan lett brukes til å lage alle verdens diagrammer og lignende visuelle hjelpemidler.

9.5.3 IMPLEMENTERING

Eclipse (http://www.eclipse.org/)
Eclipse er et utviklingsverktøy alle på gruppa er godt kjent med fra TDT4100 og det falt seg ganske naturlig å bruke det. Vi kunne ha tatt i bruk andre programmer men det var ingen som var intresert i å sette seg inn et nytt utviklingsverktøy, da Eclipse fungerer godt.

MySQL (http://www.mysql.com/)
For å kunne gjennomføre dette prosjektet er man avhengig av en database. MySQL er i utgangspunktet lett å sette opp mot Java, selvom det dukket opp komplikasjoner på veien. Det finnes også mye god dokumentasjon til MySQL som gjorde at vi valgte å bruke det.

9.6 BRUKERMANUAL

I denne seksjonen kommer det to brukermanualer. En teknisk brukermanual som administratorer i koiestyret kan bruke for å opprettholde og styre database/program, og en ikke-teknisk der brukeren får en forklaring på hvordan programmet skal brukes.

9.6.1 TEKNISK BRUKERMANUAL

Login på NTNU sine servere og åpne mysql med følgende bruker og passord:
Brukernavn: alekh_IT1901
Passord: abcd1234
Databasenavn: alekh_prosjekt1

Sette opp Java Development Kit:
Gå til følgende side og last ned Java JDK for ditt operativsystem: http://www.oracle.com/technetwork/

java/javase/downloads/jdk7-downloads-1880260.html

Sette opp Eclipse Luna:
Gå til følgende side for å laste ned Eclipse Luna til ditt foretrukne operativsystem:
https://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/lunasr1

Legge inn prosjektet i Eclipse Luna:
Gå til følgende link for oppsett av git i Eclipse: http://wiki.eclipse.org/EGit/User_Guide Cloning_or_adding_Repositories
Repoet for kildekoden ligger i https://github.com/halv00rsen/Prosjekt-I. Når du får beskjed om å velgde branch, velg master for å få ned alle nødvendige filer.

Hente og legge til endringer til og fra GitHub:
For å hente og legge til endringer i kildekoden, gå til følgende link for hvordan man puller og pusher til GitHub:
http://wiki.eclipse.org/EGit/User_GuidePushing_to_other_Repositories.

9.6.2 IKKE-TEKNISK BRUKERMANUAL

Den ikke-tekniske brukermanualen vil forklare hvordan bruker/admin skal bruke programmet. Programmet er delt opp i to deler der den ene delen omfatter funksjonalitet til bruker og den andre omfatter funksjonalitet administrator skal ha tilgang til. Altså det er ikke to forskjellige programmer til admin og bruker, men funksjonaliteten er begrenset til hver av dem.

Kjøring av programmet Gå til prosjektmappa, der vil det ligge en fil ved navn NTNUIKoiene.jar. Start denne fila og programmet vil kjøre.

Oppstartsvindu Ved oppstart av programmet har man to faner oppe. I innloggingsfanen kan man opprette bruker/admin og man kan logge inn i systemet. Hvis man ikke har logget inn kan man uansett se informasjon om de forskjellige koiene og når de er ledig osv.

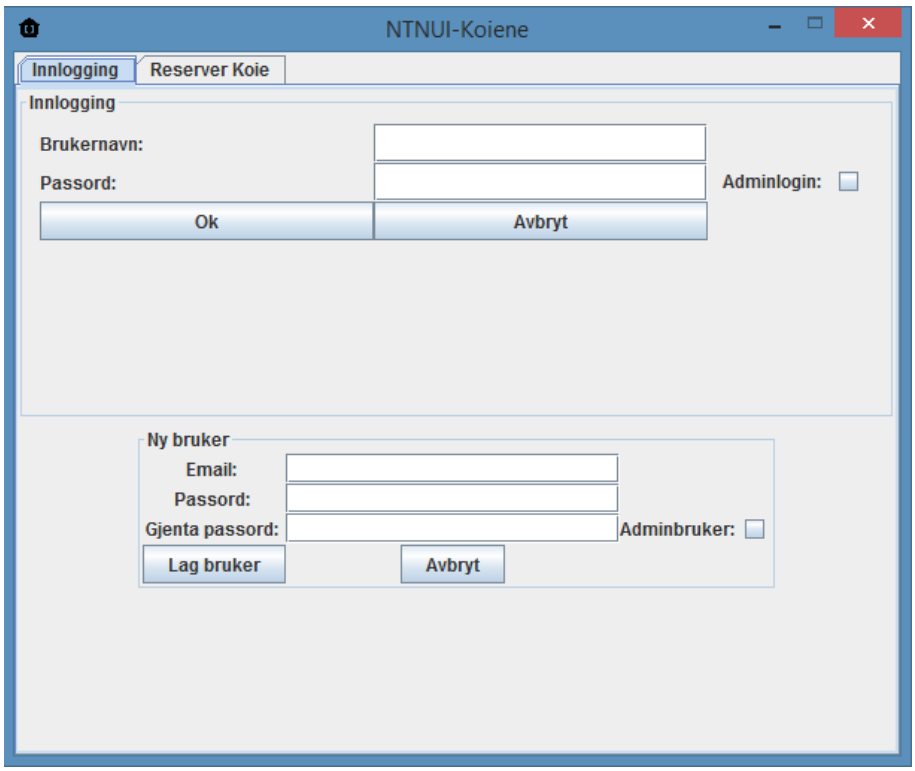


FIGURE 22: OPPSTART

Innlogging - Bruker Hvis man som bruker vil lage en ny bruker i systemet, kan man gjøre det ved oppstart av programmet. Brukeren må ha en epostadresse[1] som brukernavn og et passord[2] på minst 6 tegn. Passordet må også bekreftes[3]. Man vil få beskjed om brukeren ble opprettet eller ei. For at brukeren skal lage en bruker er det viktig at adminbrukerboksen[4] ikke er krysset av.

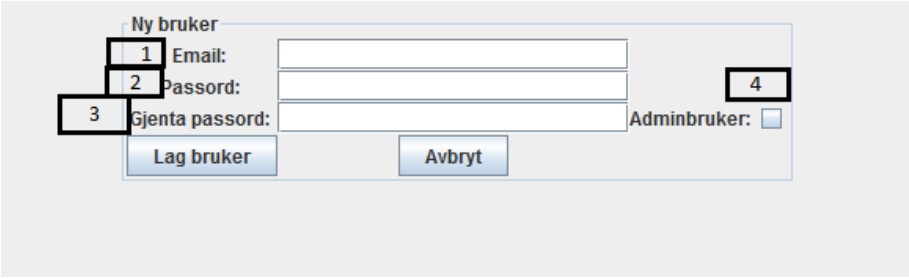


FIGURE 23: NY BRUKER

Når brukeren skal logge inn, skriver brukeren inn sin epostadresse og passord. Viktig at adminloginboksen[1] ikke er krysset av.

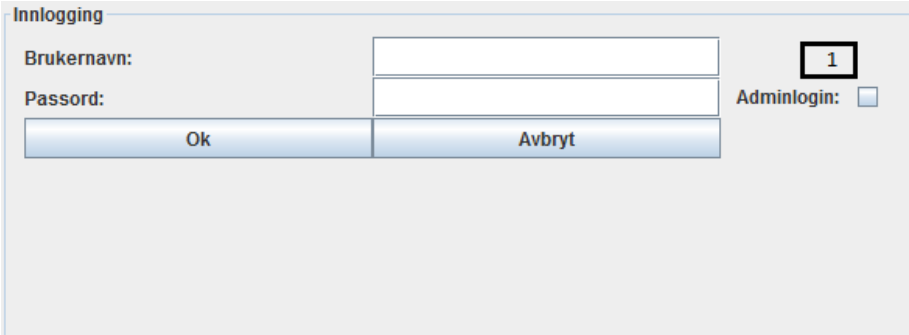


FIGURE 24: LOGIN BRUKER

Når brukeren har logget inn, vil innloggingsfanen vise hvem som har logget inn, og man har en "Log ut" knapp[1]. Når brukeren trykker på knappen[1] vil brukeren logge ut av systemet.

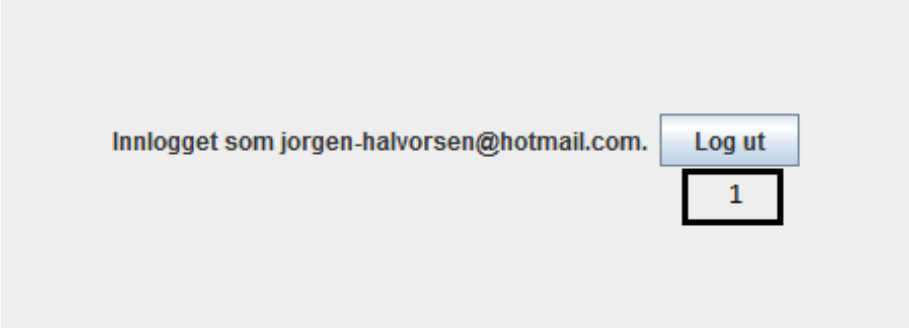


FIGURE 25: LOGOUT BRUKER

Reserver Koie - Bruker Når brukeren er logget inn kan brukeren reservere ei koie. Til høyre i vinduet[1] vil brukeren se all informasjon om valgt koie. Brukeren kan velge koie, bestemme dag, måned og antall dager reservasjonen skal vare[2]. Brukeren vil da se om den aktuelle tidsperioden er ledig[3]. Brukeren trykker så på reserverknappen[4] når han har funnet riktig dato. Brukeren kan reservere ei koie for 0-10 dager.

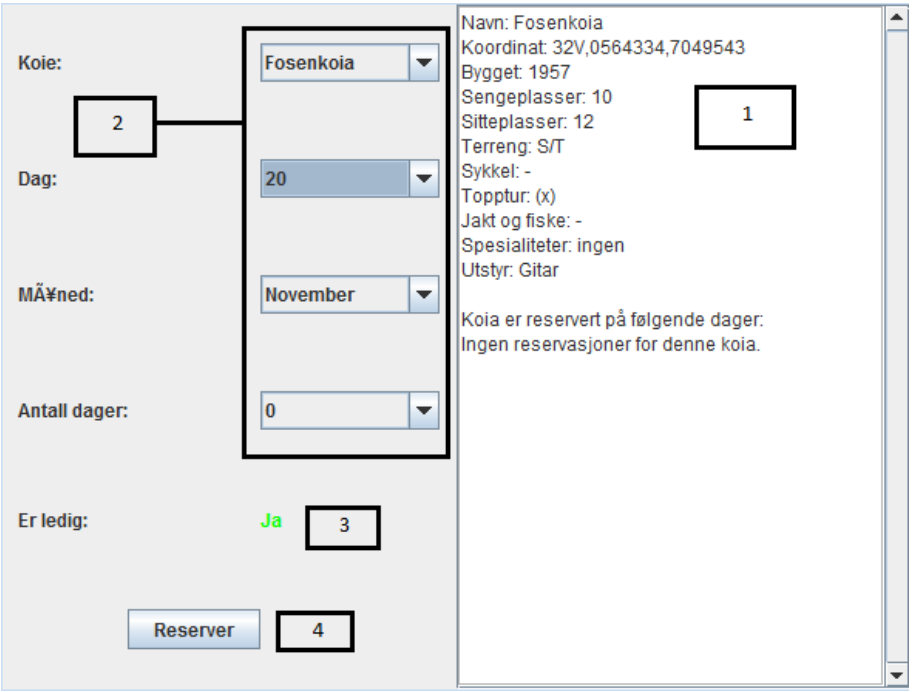


FIGURE 26: RESERVER KOIE

Reservasjoner - Bruker Når en bruker har logget inn, har brukeren tilgang til å se sine reservasjoner. Reservasjonsfanen er delt opp i to deler. Reservasjoner[1] og historie[2]. I reservasjoner ser brukeren alle sine reservasjoner i framtiden[]. Her kan brukeren velge om han vil slette en bestemt reservasjon[3]. I den andre kolonnen ser brukeren sin historikk. Her kan brukeren trykke på rapport[4] på en bestemt reservasjon for å skrive en rapport om koia til systemet.

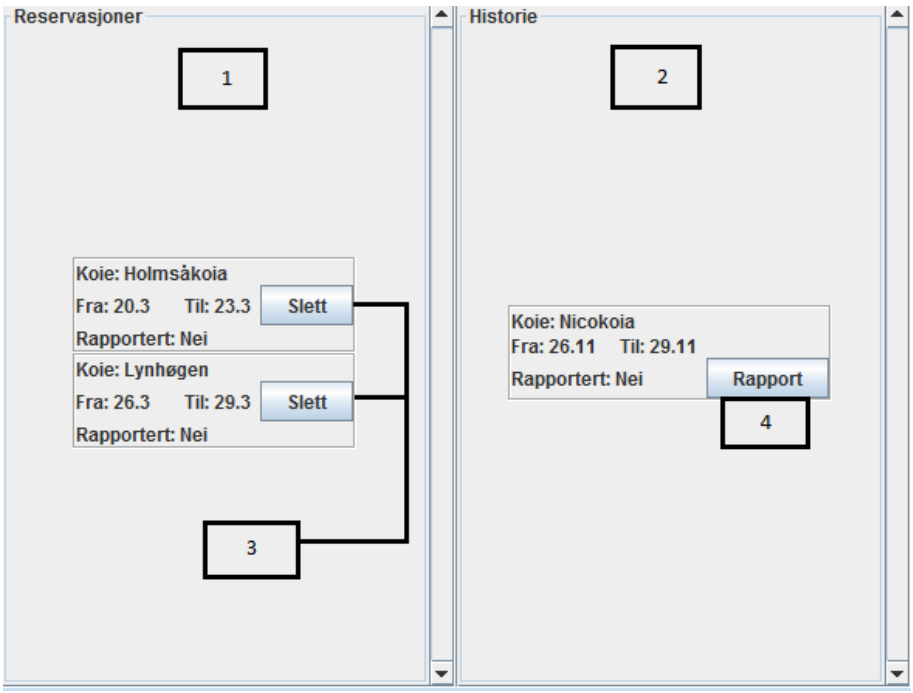


FIGURE 27: RESERVASJONER

Rapport - Bruker Når en bruker vil rapportere inn status til ei koie, vil et nytt vindu dukke opp. Dette vinduet er delt opp i tre faner; koieinfo, ødelagt utstyr og gjenglemt utstyr. Brukeren kan til en hver tid trykke avbryt og ingen informasjon blir lagret. Trykker brukeren på ok, vil all informasjon skrevet inn i rapporten bli sendt inn, og man kan ikke endre på dette etterpå. Koieinfo I denne fanen får bruker opp informasjon om hvilken reservasjon brukeren har valgt å skrive rapport om[1]. Brukeren kan her skrive inn antall brukte vedsekker[2].

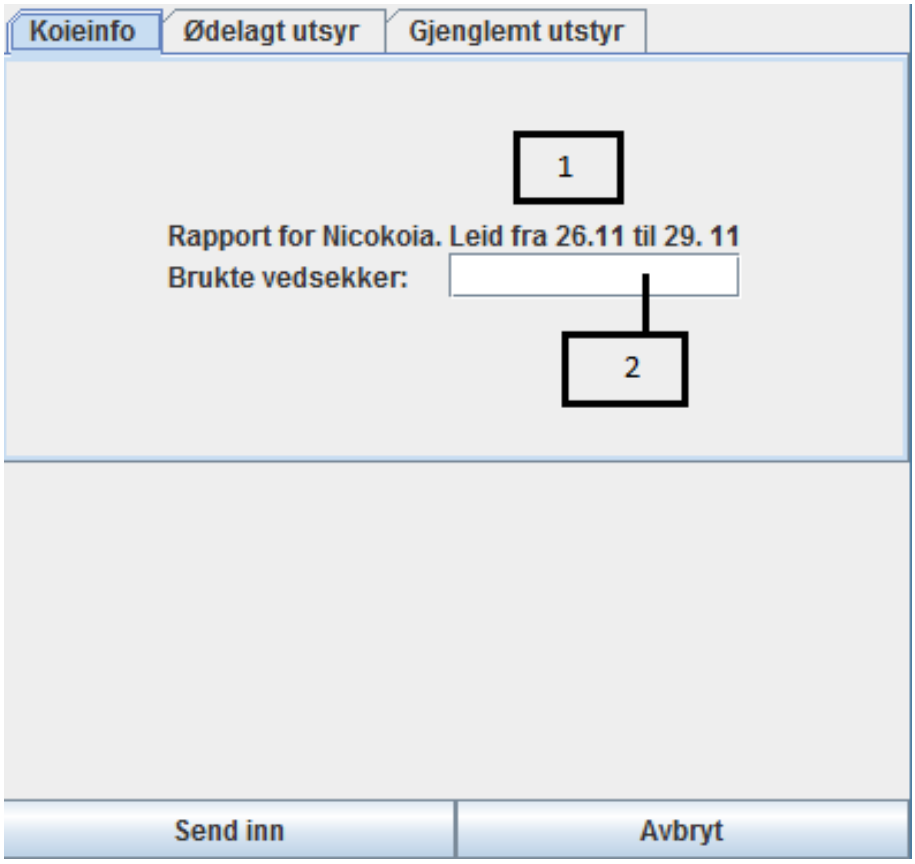


FIGURE 28: RAPPORTINFO

Ødelagt utstyr Her får bruker opp to kolonner med henholdsvis utstyr[1] og ødelagt utstyr[2]. Her kan brukeren markere[5] et eller flere utstyr i en kolonne og flytte den over til den andre ved å trykke på knappene "Legg til"[3] og "Fjern"[4].

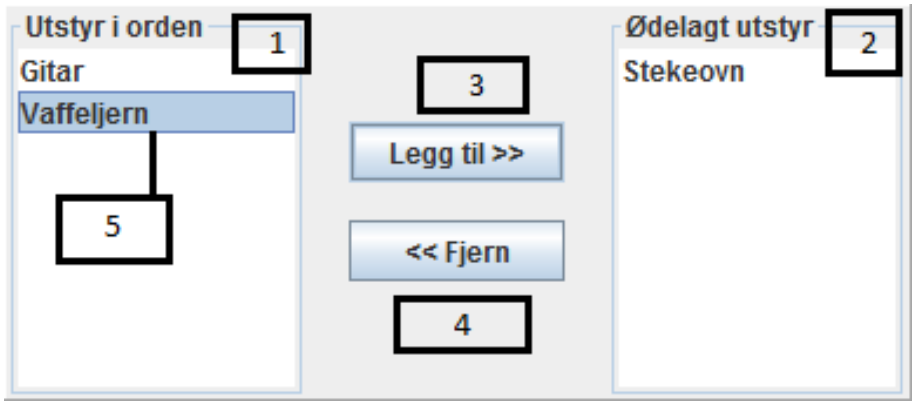


FIGURE 29: RAPPORT UTSTYR

Gjenglemt utstyr Her kan brukeren skrive inn utstyr som brukeren har glemt igjen på koia. Brukeren skriver inn navnet[1] og trykker legg til[2] og det havner i gjenglemt utstyrskolonnen[3]. Brukeren kan også her markere[4] utstyr i ustyrskolonnen og fjerne dem.

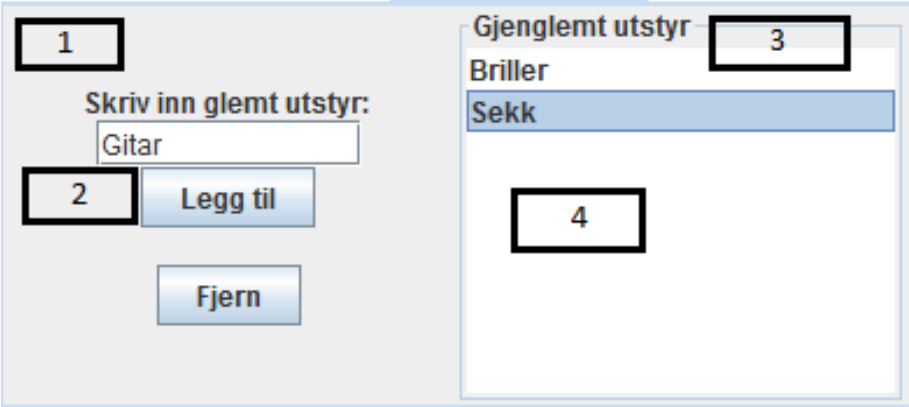


FIGURE 30: RAPPORT GJENGLEMT UTSTYR

Innloggin - administrator Hvis du som admin vil lage en ny administrator, er det viktig at adminbrukerfeltet er krysset av[1]. Admin skal ikke være en emailadresse[2], og adminbrukeren vil ikke bli godkjent hvis den inneholder en alfakrøll. Passordet må være på minst 6 bokstaver[3]. Da vil admin få opp et felt med adminkode[4] der man skal skrive inn en bestemt kode for adminregistrering. Man vil få en bemerkning om brukeren er opprettet i systemet.

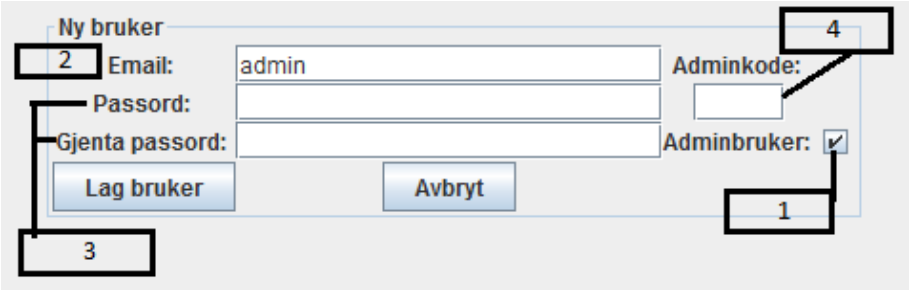


FIGURE 31: NY ADMINBRUKER

Når admin vil logge inn i systemet, er det viktig at adminloginboksen[1] er avkrysset. Skriver så inn brukernavn[2] og passord[3].

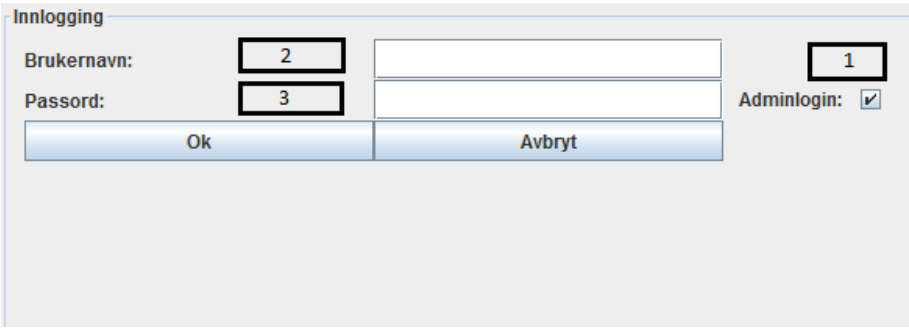


FIGURE 32: LOGIN ADMIN

Når admin har logget inn, vil administrator få opp mange nye faner[1]. Fanen administrator

nå er inne i har to knapper. En knapp som logger admin ut av systemet[2], og en knapp som tilbakestiller databasen[3]. Ved tilbakestilling av databasen, vil systemet "henge" i noen grad, siden det er ganske mange tabeller som skal tilbakestilles. All info i databasen vil bli slettet når databasen blir tilbakestilt, men alle tabeller vil bli opprettet på nytt.

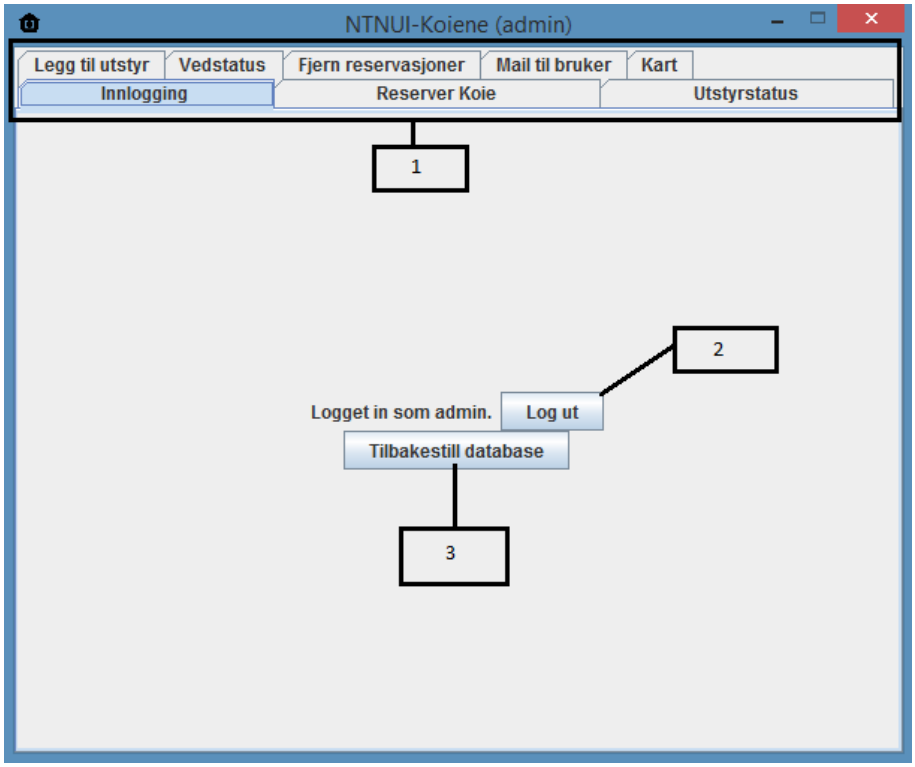


FIGURE 33: LOGOUT ADMIN

Reserver koie - administrator En administrator har ikke mulighet til å reservere ei koie på sin bruker, eller en annen administrator, men administrator kan reservere ei koie for en annen bruker i systemet ved å skrive inn navnet til den aktuelle brukeren i feletet[1]. Admin velger reservasjon[2], sjekker om reservasjonen er ledig[3], og trykker på reserver[4]. Admin kan også se informasjon om koia her[5].

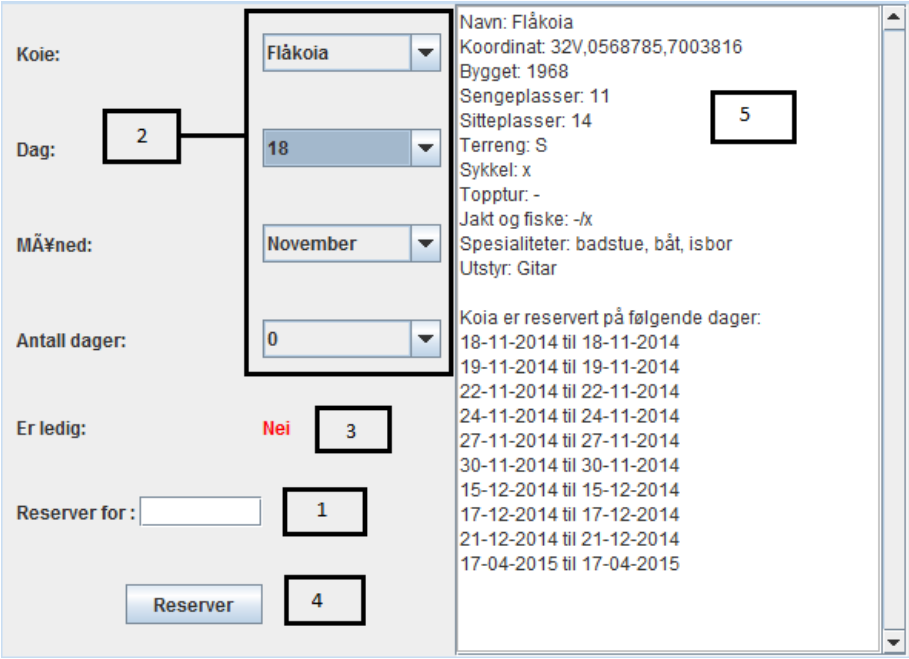


FIGURE 34: RESERVER KOIE

Utstyrstatus - administrator I denne fanen kan admin se utstyrsstatus til ei valgt koie. Fanen er delt opp i to, der admin kan velge koie[1], og under se status på utstyret. Admin kan se om utstyr er i orden[2], om det er ødelagt[3], eller om det er gjenglemt utstyr[4] koia. Denne fanen er kun for oversikt, admin kan ikke endre noen informasjon her.

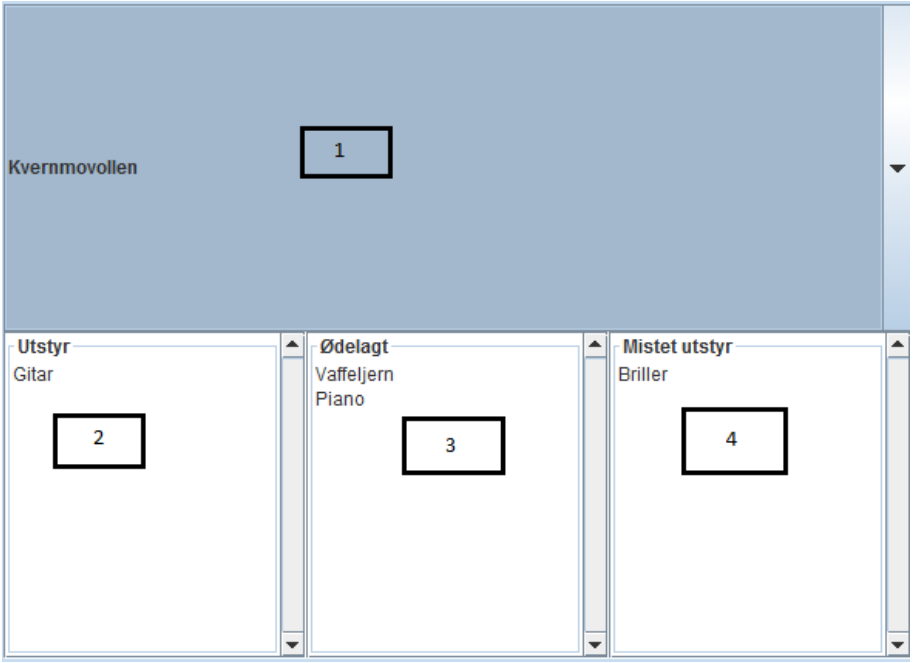


FIGURE 35: UTSTYRSSTATUS

Legg til utstyr - administrator Her kan admin endre status på utstyr i ei bestemt koie, eller legge til utstyr. Admin velger først koie[1]. Deretter kan admin legge til nytt utstyr hvis ønsket.

Skriver inn navnet på det nye utstyret[2], trykker deretter på "legg til"[3] for å legge til utstyret i koia. Til venstre kan admin velge en ting[4] i koia. Under denne kan admin bestemme status til det valgte utstyret[5], enten IN ORDER, BROKEN eller LOST AND FOUND. Til høyre er det et lite tekstfelt[6] som vise valgt utstyr, og i hvilken tilstand det utstyret er i. Admin kan velge å fjerne det valgte utstyret ved å trykke på "Fjern"[7] knappen. Admin kan velge å lagre endringene i databasen ved å trykke på "Lagre"[8], eller admin kan resette alle endringene ved å trykke på "Reset"[9]. Da vil ingen av endringene bli gjennomført.

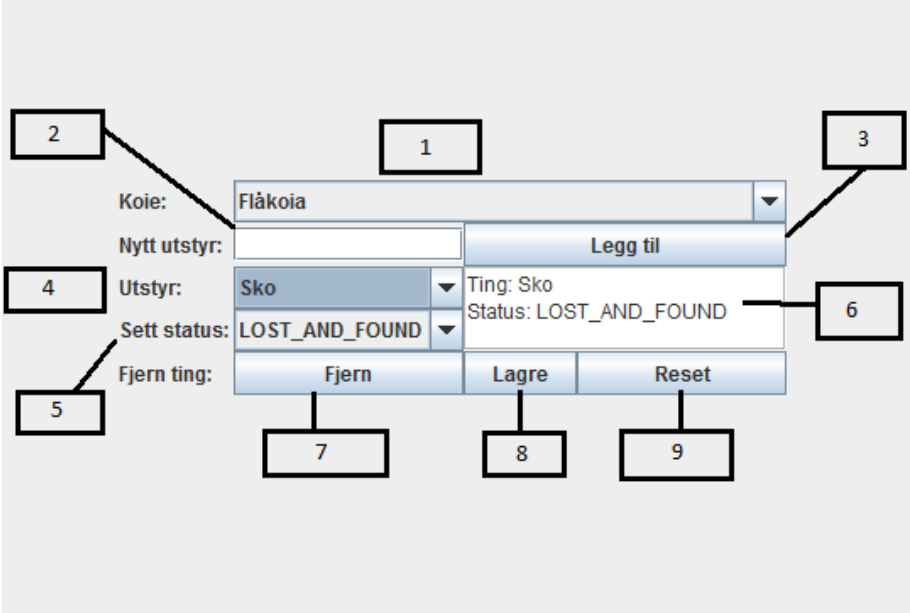


FIGURE 36: UTSTYRSBEHANDLING

Vedstatus - administrator I denne fanen kan administrator se vedstatus[1] til alle koiene i systemet. Hvis ei koie snart trenger veddugnad, vil det også stå i dette feltet[2]. Admin kan velge ei bestemt koie[3] og skrive legge til vedsekker til koia. Dette gjøres ved at admin skriver inn antall vedsekker[4] og trykker på "Legg til"[5]. Når "legg til" blir trykket på, vil dette oppdatere databasen med en gang.

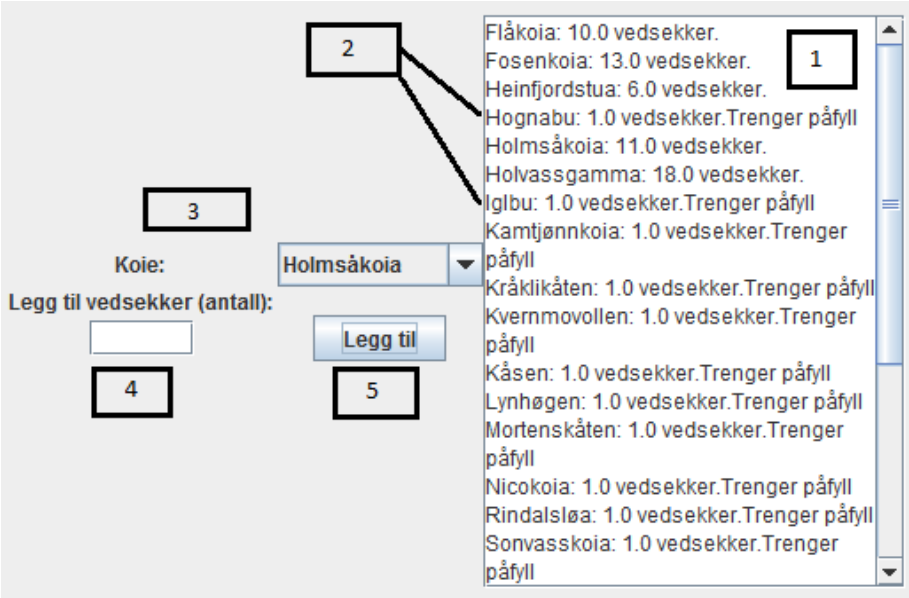


FIGURE 37: VEDSTATUS

Fjern reservasjoner - administrator I denne fanen kan administrator velge å fjerne en eller flere reservasjoner til ei bestemt koie. Dette gjøres ved at admin først velger hvilken koie det er snakk om[1], deretter vil admin få opp alle reservasjoner i ei git koie[2]. Admin kan da velge å slette en eller flere reservasjoner ved å trykke på "Slett"[3]. Admin vil da få opp en en boks som bekrefter at denne reservasjonen skal slettes.

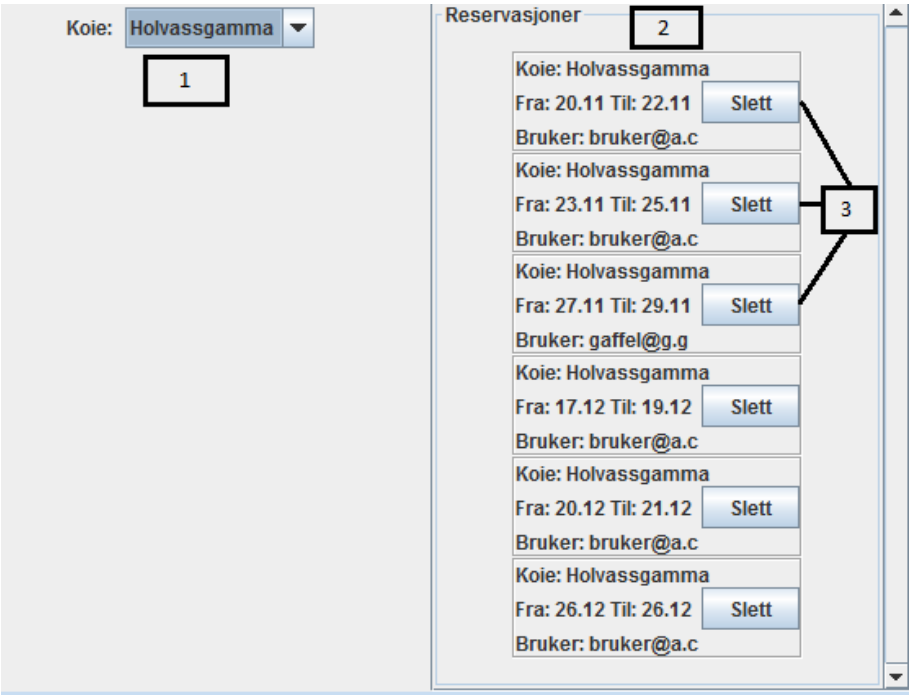


FIGURE 38: FJERN RESERVASJONER

Mail til bruker - administrator I denne fanen kan admin sende mail til en bestemt bruker i

systemet. Dette kan være seg ting som skal tas med på ei koie elr lignende. Admin velger først hvilken bruker han skal sende til[1], deretter skriver admin inn hva som skal stå i emnefeltet[2] i mailen. Deretter skriver admin inn den informasjonen som brukeren skal få[3]. Admin kan da velge å trykke på "Send"[4] for å sende mailen, eller han kan trykke "Avbryt"[5]. Ved trykk på "Avbryt" vil all tekst bli fjernet fra feltene.

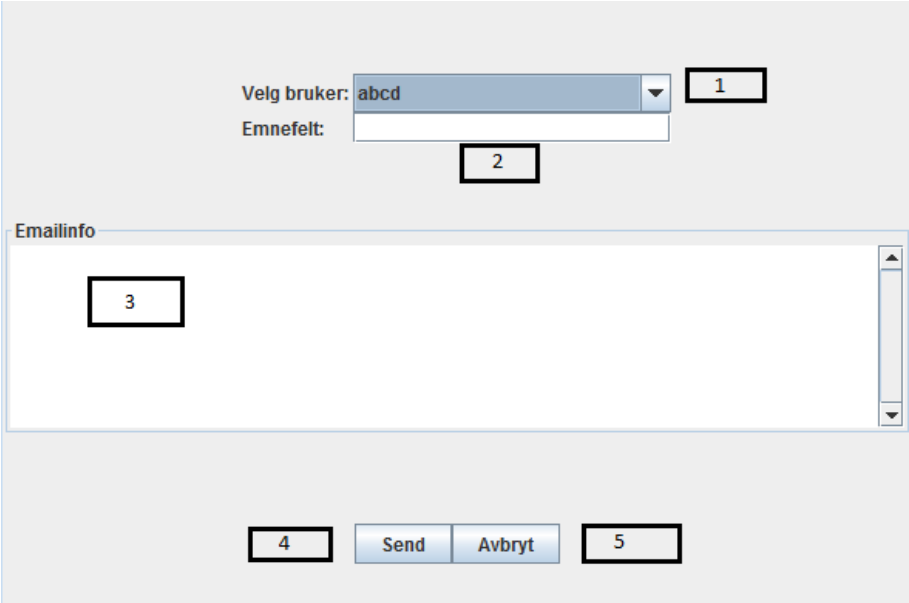


FIGURE 39: MAIL

Kart - administrator I denne fanen kan administrator få opp relevant informasjon om ei bestemt koie. Fanen er delt opp i to, der på venstre side er det et kart[1] over alle koiene. Til høyre[2] er det et tekstfelt som viser all relevant informasjon om ei bestemt koie. Denne informasjonen får man ved å trykke på koiesymbolene på kartet[3]. Hvert symbol representerer ei bestemt koie.

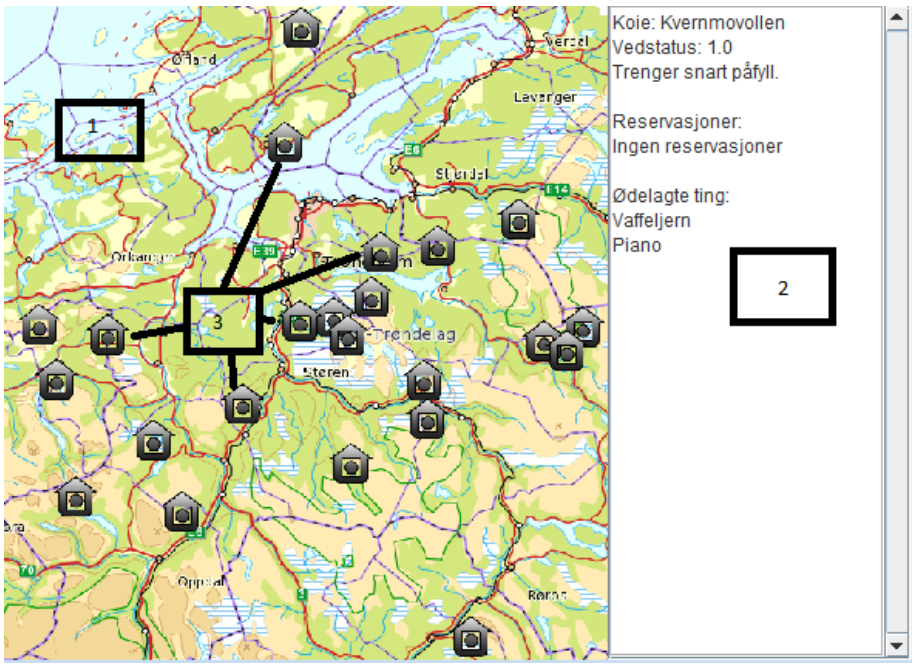


FIGURE 40: KART