HØGSKOLEN I BERGEN

Semesteroppgave i TOD076
Systemutvikling og webapplikasjoner
Våren 2010

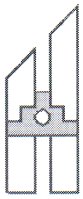
Gruppe E

Kandidater:

Innholdsfortegnelse

Sjekklister

Sjekkliste for nådd LCO-milepæl



HØGSKOLEN I BERGEN
Avdeling for Ingeniørutdanning
Institutt for data- og realfag

Semesteroppgave 2010
TOD076 – Systemutvikling og webapplikasjoner

Gruppe: E

Sjekkliste for nådd milepæl for Inception-fasen (LCO)

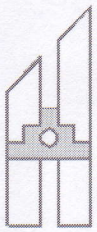
Alle kriterier i tabellen må være oppfylt for at milepælen for Inception-fasen skal regnes som nådd. Kunden signerer denne sjekklisten med evt. merknader når han mener kriteriene er oppfylt. Signering foregår normalt i et kundemøte som markerer slutten av fasen.

En kopi av dette signerte dokumentet vedlegges sluttrapporten og vil telle med i evalueringen av prosjektet.

OK (dato)	Kriterium	Merknad fra kunde
OK	*"Stakeholder concurrence on scope definition and an initial cost/schedule estimate."* Dvs.: At man er enig om omfang og kostnadsestimat (i arbeidstimer). **Visjon** + **Utviklingsplan**.	
OK	*"Agreement that the right set of requirements has been captured and that there is a shared understanding of these requirements."* Dvs.: Resultat av avklaringer med kunde (krav og andre avklaringer) nedfelt i **Visjon, Brukstilfeller, Suppl. krav**, osv ..	
OK	*"Agreement that the cost/schedule estimate, priorities, risks, and development process are appropriate."* Dvs.: At prioriteringer er avklart i **Visjon**. Art utviklingsprosessen beskrevet i **Utviklingsplan** og risikoer beskrevet i **Risikoliste** er fornuftig.	
OK	*"Agreement that initial risks have been identified and that a mitigation strategy exists for each."* Dvs.: **Risikoliste** beskriver tiltak for å redusere risikoene og **Iterasjonsplan(e)** for elaboration viser konkrete planer for å gjennomføre dette.	

Bergen, 31/3 - 2010 Kundens signatur: Lars-Petter Helland

Sjekkliste for nådd LCA-milepæl



HØGSKOLEN I BERGEN
Avdeling for Ingeniørutdanning
Institutt for data- og realfag

Semesteroppgave 2010
TOD076 – Systemutvikling og webapplikasjoner

Gruppe: E

Sjekkliste for nådd milepæl for Elaboration-fasen (LCA)

Alle kriterier i tabellen må være oppfylt for at milepælen for Elaboration-fasen skal regnes som nådd. Kunden signerer denne sjekklisten med evt. merknader når han mener kriteriene er oppfylt. Signering foregår normalt i et kundemøte som markerer slutten av fasen.

En kopi av dette signerte dokumentet vedlegges sluttrapporten og vil telle med i evalueringen av prosjektet.

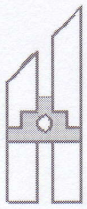
For å unngå splitting av kriterieliste og signering starter tabellen på neste side.

Gruppe: E

OK (dato)	Kriterium	Merknad fra kunde
16/4	*"Are the product Vision and requirements stable?"* Dvs.: At vi er enige om hva som skal lages og at alle krav er beskrevet i detalj i **Brukstilfeller** og **Supplerende Krav**.	
"	*"Is the architecture stable?"* Dvs.: De arkitektoniske elementene beskrevet i **Systemarkitektur** (og demonstrert i **Arkitekturprototypen**) dekker de mest kritiske områdene av applikasjonsarkitekturen.	
"	*"Are the key approaches to be used in testing and evaluation proven?"* Dvs.: Hvordan enhetstesting, funksjonstesting og akseptansetesting skal foregå er beskrevet i **Utviklingsplan**.	
"	*"Have testing and evaluation of executable prototypes demonstrated that the major risk elements have been addressed and resolved?"* Dvs.: Kjøring av de mest arkitektur-kritiske brukstilfellene er gjennomført på en versjon av **Applikasjonen** installert på kundens web-tjener.	
"	*"Are the iteration plans for Construction of sufficient detail and fidelity to allow the work to proceed?"* Dvs.: **Iterasjonsplanen(e) for Construction-fasen** er detaljerte og troverdige nok for å fortsette.	
"	*"Are the iteration plans for the Construction phase supported by credible estimates?"* Dvs.: **Iterasjonsplanen(e) for Construction-fasen** inneholder troverdige estimater.	
"	*"Do all stakeholders agree that ... (the plan will result in the visioned system)?"* Dvs.: Det er samsvar mellom plan og **Visjon**.	
OK	*"Are actual resource expenditures versus planned expenditures acceptable?"* Dvs.: OK.	

Bergen, 16/4-2010 Kundens signatur: Lars Petter Helland

Sjekkliste for nådd IOC-milepæl



HØGSKOLEN I BERGEN
Avdeling for Ingeniørutdanning
Institutt for data- og realfag

Semesteroppgave 2010
TOD076 – Systemutvikling og webapplikasjoner

Gruppe: E

Sjekkliste for nådd milepæl for Construction-fasen (IOC)

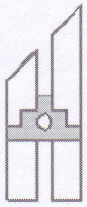
Alle kriterier i tabellen må være oppfylt for at milepælen for Construction-fasen skal regnes som nådd. Kunden signerer denne sjekklisten med evt. merknader når han mener kriteriene er oppfylt. Signering foregår normalt i et kundemøte som markerer slutten av fasen.

En kopi av dette signerte dokumentet vedlegges sluttrapporten og vil telle med i evalueringen av prosjektet.

OK (dato)	Kriterium	Merknad fra kunde
29/4	*"Is this product release stable and mature enough to be deployed in the user community?"* Dvs.: At man har en "ferdig" applikasjon som er klar til beta-testing / kundetesting.	Brukstilfeller: · 5 av 7 MEGET HØY OK. · 2 av 7 —"— har bugs. · Alle HØY implementert. Loddtrekning: · Mener at det virker. Har testet. Kvalitet: · Bra! Ellers: · Har gjort mye ekstra. · Bra brukergrensesnitt.
OK	*"Are all stakeholders ready for the transition into the user community?"* Dvs.: Er kunden klar til å ta applikasjonen i bruk? Opplæring?	
OK	*"Are actual resource expenditures versus planned expenditures still acceptable?"* Dvs.: OK	

Bergen, 29/4-2010 Kundens signatur: Lars P Hellund

Sjekkliste for nådd PR-milepæl



HØGSKOLEN I BERGEN
Avdeling for Ingeniørutdanning
Institutt for data- og realfag

Semesteroppgave 2010
TOD076 – Systemutvikling og webapplikasjoner

Gruppe: E

Sjekkliste for nådd milepæl for Transition-fasen (PR)

Alle kriterier i tabellen må være oppfylt for at milepælen for Transition-fasen skal regnes som nådd. Kunden signerer denne sjekklisten med evt. merknader når han mener kriteriene er oppfylt. Signering foregår normalt i et kundemøte som markerer slutten av fasen.

En kopi av dette signerte dokumentet vedlegges sluttrapporten og vil telle med i evalueringen av prosjektet.

OK (dato)	Kriterium	Merknad fra kunde
6/5	*"Are the users satisfied?"* Dvs.: At applikasjonen er ferdig. Helst bør alle krav med meget høy prioritet være i orden.	Feil/mangler som ble funnet under testing er fikset. Kvalitet og helhetsinntrykk på den ferdige applikasjonen Bra! Har gjort en del ekstra. (Litt smågnikk)
OK	*"Are actual resource expenditures versus planned expenditures acceptable, and, if not, what actions can be taken in future projects to address this issue?"* Dvs.: OK	

Bergen, 6/5-2010 Kundens signatur: Lars P Hellund

Refleksjonsnotat

Hyttestyret ved HiB ønsket et nettbasert bestillingssystem for tildeling av skolens firmahytter. Vi har utviklet en komplett web applikasjon som gir bruker (ansatt som ønsker å leie en av hyttene) mulighet til å reservere en hytte for et gitt tidsrom ved hjelp av en grafisk kalender. Vi har gått langt i å sikre et visuelt godt brukergrensesnitt ved hjelp av CSS, Photoshop og kreativ design. Vi har lagt mye arbeid i at det skal være et komplett profesjonelt system som kan tas i bruk.

Vi har utviklet en avansert og innholdsrik administratorfunksjon som til en hver tid gir hytteansvarlig en fullstendig oversikt over reservasjoner på hyttene, samt mulighet til å endre på alle regler, parametere, informasjon, innstillinger, tekster, bilder osv. Systemet tildeler hyttene automatisk ved hjelp av loddtrekning og håndhever gitte regler for reservasjon og utleie satt av Hyttestyret ved HiB.

Arbeidets gang

Systemet er utviklet ved hjelp av anerkjente WEB-teknologier og etter RUP modellen. Prosjektleder styrte iterasjonsplanene og holdt jevnlige møter, enten på skype eller personlig på HiB. Vi har dermed fått god erfaring i å jobbe i et større utviklingsteam (7 personer), og med å utvikle en reel applikasjon for en reel kunde som krever kvalitet.

Arbeidsfordeling og samarbeid

Vi har organisert oss i en hierarkisk struktur med prosjektleder, analytikere, systemarkitekt, testere og utviklere. Denne strukturen har fungert meget godt i forhold til prosjektet. Vi signerte samarbeidskontrakter, og hadde et godt samarbeid og god fremdrift.

Eventuelle problemer

Siden vi måtte definere roller i semesteroppgaven har læringseffekten vært begrenset til tildelt rolle. Dette betyr at prosjektleder har fått mye erfaring med å lede større prosjekter, analytikere har fått stor innsikt i denne rollen, testerne vet nå hvordan testing foregår, og utviklerne har nå svært gode kunnskaper om programmering i forhold til MVC strukturen (som er en vesentlig del av pensum i TOD076). De som ikke var utviklere kan nok føle at læringseffekten i forhold til MVC har vært mangelfull. Jo større gruppene er, jo dårligere blir breddelæringseffekten.

Valg og løsninger som måtte gjøres om

Vi jobbet etter modellen i RUP. Det betyr at vi har jevnlig iterert og gjort små endringer underveis. Det har ikke vært noen store endringer i forhold til iterasjonsplanene eller visjonsdokumentet.

RUP som arbeidsprosess

RUP gir en klar og strukturert måte å jobbe på som egner seg god for ikke å møte på for mange overraskelser underveis, spesielt ved større prosjekter og mange personer i arbeidsgruppen.

Hvordan fungerte RUP som arbeideprosess?

En sentral filosofi i RUP er å angripe det vanskeligste først. Vi har opplevd RUP som et svært godt hjelpemiddel i vår utviklingsprosess. Samt at en hierarkisk organisering er viktig for at ting skal bli gjort, og for å sikre en kontinuerlig fremdrift.

Hjalp RUP på veien mot implementasjon?

Vi har erfart at inception fasen er svært viktig. Denne er sentral fordi det er her vi legger alle planene om hva vi skal lage etter møter med kunden, og planer for hvordan vi skal klare å lage dette. I denne semesteroppgaven fikk vi en svært kort inception fase, men vi føler allikevel at vi løste inception fasen godt på tross av korte frister. De neste fasene gikk veldig greit etter at gode realistiske iterasjonsplaner ble planlagt av prosjektleder.

Estimering og tidsplanlegging

Tidsplanlegging har vært bra. Vi valgte å implementere svært mye av kundens ønsker og hadde et generelt fokus på å levere et solid og komplett produkt.

Erfaringer

Vår erfaring er at semesteroppgaven tok mye energi og tid siden vi ønsker å gi et godt inntrykk fordi oppgaven teller 30 % av sluttkarakter i faget TOD076.

Hvor godt traff estimatene og planene?

Våre estimater og planer fungerte utmerket. Mye av grunnen til dette er fordi vi jobbet mye med oppgaven, hadde en strukturert og engasjert prosjektleder og fordi vi alle utførte våre tildelte oppgaver innen avtalt tid.

Oppsummering

Vi har lært hvordan det er å jobbe i et større utviklings team som følger RUP modellen for å lage en reel web applikasjon for en reel kunde. Vi har erfart at en hierarkisk struktur med en tilstedeværende prosjektleder er nødvendig for å sikre fremdrift. Som nevnt tidligere er denne rollefordelingen bra for prosjektet sin helhet, men de som ikke var utviklere føler at læringseffekten i forhold til MVC har vært mangelfull.

Visjonsdokument

Formålet med dette dokumentet

Formålet med dette dokumentet er å identifisere, analysere og definere interessentenes behov og egenskaper til HytteWeb. Dokumentet fokuserer på behov og egenskaper i interessenter og brukers øyne, samt hvorfor disse behovene eksisterer. Detaljene for hvordan HytteWeb dekker disse behovene er beskrevet i listen over brukstilfeller og supplerende kravspesifikasjon.

Problembeskrivelse

Problemet ligger i dagens bestillingsløsning basert på kalenderfunksjonen i Microsoft Office for Høyskolen i Bergen sine 2 firmahytter. Dette går utover hytteansvarlig som sitter med mye manuelt arbeid og skolens ansatte som søker om leie av hytte. Dette har sin innvirkning på hytteansvarlig og på ansatte som har opplevd en del uheldige omstendigheter.

En bedre løsning vil være et nettbasert bestillingssystem som tildeler hytter automatisk, gir hytteansvarlig status for utleie av hyttene til en hver tid, gir oversikt over avbestillinger og gir automatisk håndheving av regler forbundet med ansattes historikk for leie samt spesielle regler for ferier.

Visjon

Hyttestyret ved HiB ønsker et nettbasert bestillingssystem for tildeling av skolens firmahytter. Dette systemet skal gi bruker (ansatt som ønsker å leie en av hyttene) mulighet til å reservere en hytte for et gitt tidsrom. Det skal også til en hver tid gi hytteansvarlig en fullstendig oversikt over reservasjoner på hyttene. Systemet skal tildele hyttene automatisk ved hjelp av loddtrekning og skal håndheve gitte regler for reservasjon og utleie satt av Hyttestyret ved HiB. Systemet skal utvikles ved hjelp av anerkjente WEB-teknologier.

Involverte parter

Hyttestyret ved HiB og hytteansvarlig

Denne parten er oppdragsgiver og formulerer kravene til applikasjonen. Parten gir også regelsett for reservasjonssystemet som skal utvikles.

Lars Petter Helland

Kontaktperson mellom utviklere og hytteansvarlig.

Sluttbruker /ansatt ved HiB

Denne parten er sluttbruker av systemet. Systemet skal gi ansatt mulighet til å reservere hytter, sjekke reservasjonshistorikk og avbestille hytter.

Produktsoversikt

Applikasjonen skal gi sluttbruker (ansatt) mulighet til å reservere firmahyttene til HiB elektronisk via nett. Disse kan reserveres for enkelt dager eller perioder. Videre skal applikasjonen uføre en trekning for tildeling av hytte dersom flere ansatte søker om samme periode. Søker skal varsles og bekrefte reservasjoner via e-post generert av systemet. En reservasjon bekreftes ved å besøke en unik link gitt i e-posten. En bruker som er tildelt en hytte varsles via e-post generert av systemet. Bruker bekrefter tildelingen ved å besøke unik link gitt i e-posten.

Systemet opprettholder rettferdighet for tildeling av hytter til brukere ved loddtrekning og gitte karanteneregler.

Systemet skal også håndtere avbestillinger utenfor og innenfor gitte regler. Avbestillinger håndteres via link til avbestilling oppgitt i e-post for reservasjon eller tildeling som bruker har mottatt.

Applikasjonen skal videre håndtere gitte regler forbundet med ferier (høstferie, vinterferie og påskeferie). Administrator skal kunne legge inn nye ferier i systemet via egen nettside for administrator.

Applikasjonen skal til en hver tid enkelt kunne gi hytteansvarlig fullstendige rapporter over hvem som leier, tidsrommet de leier for og hvor mye de skal betale for hver enkelt hytte. Det skal også være mulig å få rapportene for hver enkelt hytte sendt per e-post til personalavdeling som trekker leiebeløpet fra lønnen til leietager.

Administrator av systemet skal via egen nettside kunne endre innstillinger for systemet, endre informasjon og priser for hytter, legge inn reservasjoner for brukere og kunne sperre en hytte for utleie i en gitt periode.

Brukstilfeller og funksjonelle krav

Ansatt /leietager:

- Reservere hytte Jondal
- Reservere hytte Upsete
- Se reservasjoner for en valgt hytte
- Få tilsendt oversikt over egne reservasjoner via epost
- Kunne avbestille reservert/ tildelt hytte
- Lese kontaktinformasjon

Hytteansvarlig /administrator:

- Endre informasjon for hytte Jondal
- Endre reglement for hytte Jondal
- Endre priser for hytte Jondal
- Legge inn sperre for utleie for hytte Jondal
- Endre informasjon for hytte Upsete
- Endre reglement for hytte Upsete
- Endre priser for hytte Upsete
- Legge inn sperre for utleie for hytte Upsete

- Endre kontaktinformasjon
- Endre viktige beskjeder på forside
- Endre tekst på forside
- Hente statistikk for Jondal
- Hente statistikk for Upsete
- Sende statistikk for Jondal til personalavdeling
- Sende statistikk for Upsete til personalavdeling

Andre produktkrav

Systemet skal utvikles med tanke på gjenbruk av kode. Overtager skal kunne utvikle systemet videre uten å gjøre store endringer i arkitektur

Personvern er ikke viktig i systemet da det er brukernavn ved HiB som identifiserer hvem som leier en hytte.

Krav til dokumentasjon

Av arkitekturdokumentet skal all teknisk informasjon rundt systemet dokumenteres på en klar og informativ måte.

Nettsidene til systemet skal være mest mulig selvforklarende sett fra ansatt/kunde og administrators side. Egne hjelpesider skal produseres for mer komplekse regler i forbindelse med utleie av firmahyttene. Med dette menes egne hjelpesider for reglene rundt leie av hytter samt regler for utvelging ved trekning og karantene.

Utviklings- og prosjektplan

Formålet med dette dokumentet

Gi en komplett oversikt over tidsfrister, milepæler, faser, strukturering, roller, ansvar, samt informasjon om organiseringen av prosjektet og gruppen.

Oversikt over prosjektet

Hensikt, mål og omfang

HytteWeb er en applikasjon som er ønsket av HiB for å lettere kunne administrere bestilling/reservasjon av de to (2) hyttene som HiB disponerer. Applikasjonen skal være nettbasert, og plassert på en intern server, slik at alle lett kan utføre bestillinger/ reservasjoner.

Forutsetninger og begrensninger

En forutsetning for prosjektet er at applikasjonen kjører og fungerer i henhold til avtalt kravspesifikasjon.

Leveranser

Leveranse vil skje ved prosjektslutt.

Revidering av planen

Skal minst forekomme en gang pr. iterasjon.

Organisering av prosjektet

Organisasjonsstruktur

Er vel det hele dette dokumentet handler om? Organisasjonskart?

Eksterne kontakter

- Lars-Petter Helland - kunde/lærer/veileder.
- Rigmor Amble Iversen, kunde/HiB.
- Bente Engen, kunde/HiB.
- Yngve Lamo, kunde/HiB.

Roller og ansvar

Rolle	Hvem?	Ansvar
Prosjektleder	Joachim Tingvold	Overordnet ansvar for organisering. Tildele arbeidsoppgaver. Ansvar for prosjektplanlegging og tidsfrister.
Analytiker	-	Ansvar for detaljert utforming av struktur og innhold i prosjektet.
	Kai Andre Hovden	UI.
	Ilona Belevica	Domenemodell.
	André Leknes	Visjon og brukstilfeller.
Arkitekt	Daniel Bjørgo	Utforme arkitekturen til prosjektet.
Utvikler	Stian Djuvsland	Koding. Designmodellering.
	Reidar Liabø	Koding. Databaseansvar.

Utviklingsmetode

Oppgaven bruker RUP (Rational Unified Process) som utviklingsmetode. Det finnes både positive og negative sider ved å bruke RUP, men en av de bedre sidene er at man får kontroll på risikoer veldig tidlig i prosjektet.

Prosjektplan

Fase- og iterasjonsplan

Milepæl 1 - Inception

Fra: Onsdag 17.mars 2010 — Til: Fredag 26.mars 2010

Iterasjon 1

Fra: Onsdag 17.mars 2010 — Til: Fredag 26.mars 2010

Hva?	Hvem?	Når?
UI Prototype v1.0	Kai	Fre 19.mars kl 10:00
Utviklings- og prosjektplan v1.0	Joachim	Tirs 23.mars kl 12:00
Risikoliste v1.0	Joachim	Tirs 23.mars kl 12:00
UI Prototype v1.5	Kai	Tirs 23.mars kl 12:00
Domenemodell v1.0	Ilona	Tirs 23.mars kl 12:00
Supplerende kravspesifikasjon v1.0	Ilona	Tirs 23.mars kl 12:00
Visjonsdokument v1.0	Andre	Tirs 23.mars kl 12:00
Brukstilfeller v1.0	Andre	Tirs 23.mars kl 12:00
UI Prototype v1.6	Kai	Tors 25.mars kl 23:59
Visjonsdokument v1.1	Andre	Tors 25.mars kl 23:59
Brukstilfeller v1.1	Andre	Tors 25.mars kl 23:59
Domenemodell v1.1	Ilona	Tors 25.mars kl 23:59
Supplerende kravspesifikasjon v1.1	Ilona	Tors 25.mars kl 23:59
Utviklings- og prosjektplan v1.1	Joachim	Tors 25.mars kl 23:59
Iterasjonsplan 2 v1.0	Joachim	Tors 25.mars kl 23:59

Milepæl 2 - Elaboration

Fra: Fredag 26.mars 2010 — Til: Fredag 16.april 2010

2. iterasjon

Fra: Fredag 26.mars 2010 — Til: Fredag 9.april 2010

Hva?	Hvem?	Når?
Starte på kodingen av de mest kritiske .jsp-filene, så de matcher UI-prototyp v1.6.	Kai	Tirs 6.april kl 10:00
Systemarkitektur v1.0	Daniel	Tirs 6.april kl 10:00
Databasemodell v1.0	Reidar	Tirs 6.april kl 10:00
Designmodell	Stian	Tirs 6.april kl 10:00
Kommet i gang med koding av applikasjon. Fokus på hyttereservering.	Stian, Reidar, Kai	Tirs 6.april kl 10:00
Brukstilfeller v1.2 (når L-P legger ut mer info på It's Learning vedr. kravspesifikiasjon)	Andre	Tirs 6.april kl 10:00
Supplerende kravspesifikasjon (når L-P legger ut mer info på It's Learning vedr. kravspesifikiasjon)	Ilona	Tirs 6.april kl 10:00
Kjørende prototype av HytteWeb	Stian, Reidar, Kai	Fre 9.april kl 12:00
Iterasjonsplan 3	Joachim	Fre 9.april kl 12:00

3. iterasjon

Fra: Fredag 9.april 2010 — Til: Fredag 16.april 2010

Hva?	Hvem?	Når?
Brukstilfeller i henhold til «Prioriteringslisten» til kunde (lagt ut på It's Learning)	Ilona	Tors 15.april kl 10:00
Systemarkitektur	Daniel	Tors 15.april kl 10:00
Logg for inception-fase/iterasjon 1	Joachim	Tors 15.april kl 10:00
Logg for iterasjon 2	Joachim	Tors 15.april kl 10:00
Plan for iterasjon 4	Joachim	Tors 15.april kl 10:00
Plan for Construction-fasen	Joachim	Tors 15.april kl 10:00
UI-Prototype -- fikse småting	Kai	Tors 15.april kl 10:00
Kjørende prototype av HytteWeb	Stian, Reidar, Kai	Tors 15.april kl 10:00

Milepæl 3 - Construction

Fra: Fredag 16.april 2010 — Til: Fredag 30.april 2010

4. iterasjon

Fra: Fredag 16.april 2010 — Til: Fredag 23.april 2010

Hva?	Hvem?	Når?
Brukstilfelle: Se reservasjon og tildelinger for en hytte	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Reservere hytte (før loddtrekning)	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Reservere hytte (etter loddtrekning, hytte ledig)	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Loddtrekning og tildeling av hytte	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Akseptere tildelt hytte	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Avbestille tildelt hytte før 2-ukers frist	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Se informasjon om hytte	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Avbestille tildelt hytte etter 2-ukers frist	Stian, Reidar	Fre 23.april kl 10:00
Logg iterasjon 3	Joachim	Fre 16.april kl 23:59
Plan iterasjon 5	Joachim	Fre 23.april kl 10:00
Skisse for Transition-fasen	Joachim	Fre 23.april kl 10:00
Testing av alle brukstilfellene som har blitt implementert	Ilona, Kai, André	Fre 23.april kl 10:00
Oppdatere Systemarkitektur-dokumentet i henhold til nye brukstilfeller som skal implementeres	Daniel	Fre 23.april kl 10:00

5. iterasjon

Fra: Fredag 23.april 2010 — Til: Torsdag 29.april 2010

Hva?	Hvem?	Når?
Fullføre brukstilfeller fra iterasjon 4	Reidar, Stian	Tirs 27.april kl 10:00 Tors 29.april kl 10:00
Brukstilfelle: Legge inn ferieperioder (ADMIN)	Reidar, Stian	Tirs 27.april kl 10:00 Tors 29.april kl 10:00
Brukstilfelle: Rapport tildelinger tilbake i tid m/ pris++ (ADMIN)	Reidar, Stian	Tirs 27.april kl 10:00 Tors 29.april kl 10:00
Brukstilfelle: Legge inn relativ frist for avbestilling (ADMIN)	Reidar, Stian	Tirs 27.april kl 10:00 Tors 29.april kl 10:00
Brukstilfelle: Legge inn reservasjoner for andre (ADMIN)	Reidar, Stian	Tirs 27.april kl 10:00 Tors 29.april kl 10:00
Grundig test av alle brukstilfeller som er implementert (ved å følge testplan & malen i testlogg.doc). Utføres på Tirsdag 27.april, slik at evt. feil kan fikses på innen Torsdag 29.april kl 10:00.	Kai, Ilona, Andre	Tirs 27.april kl 23:59
Logg for iterasjon 5	Joachim	Tors 29.april kl 23:59
Iterasjonsplan 6	Joachim	Tors 29.april kl 23:59
Transitionplan	Joachim	Tors 29.april kl 23:59
Tett oppfølging & oppdatering av systemarkitekturdokumentet	Daniel	Tors 29.april kl 10:00
Brukstilfelle: De merket med «middels» og «lav» -- blir ikke tildelt resurser, men om tiden strekker til etter andre kravene er implementert, og testet «OK», så kan man gyve løs på dette.	Reidar, Stian, Joachim	(Tors 29.april kl 10)

Milepæl 4 - Transition

Fra: Torsdag 29.april 2010 — Til: Torsdag 6.mai 2010

6. iterasjon

Fra: Torsdag 29.april 2010 — Til: Torsdag 6.mai 2010

Hva?	Hvem?	Når?
Legge ut mer informasjon enkelte plasser, samt endre informasjon noen plasser, ifbm. tilbakemelding vi fikk fra betatestingen	Kai, Ilona, Andre	Ons 5.mai kl 16:00
BUG: Sessio-timeout gir Nullpointer	Reidar	Ons 5.mai kl 16:00
BUG: EpostDAOImpl/BestillingDAOImpl gir Nullpointer	Joachim	Ons 5.mai kl 16:00
MANGEL/FIX: Rapport tildelinger tilbake i tid skal også kunne ta med betalte	Stian	Ons 5.mai kl 16:00
BUG/FIX: Blank text-input under innstillinger. CSS-relatert.	Joachim	Ons 5.mai kl 16:00
FIX/BUG: Ved oppstart; kjøre trekning på evt. ting som mangler trekning (f.eks. dersom server har vært nede i noen dager).	Reidar, Stian	Ons 5.mai kl 16:00
BUG(?): Blank side ved bekreftelse av bestilling. Refresh, og det funker.	Joachim	Ons 5.mai kl 16:00
FIX: «Bestilling mottatt» - lenke til kalender hvor reservasjonen fant sted	Reidar, Joachim	Ons 5.mai kl 16:00
MANGEL/FIX: Admin skal få epost ved reservasjoner.	Joachim	Ons 5.mai kl 16:00
BUG: Fikk ikke tildelt hytte etter loddtrekning	Stian, Reidar, Joachim	Ons 5.mai kl 16:00
FIX/MANGEL: Egen JSP for exception-feil	Stian, Reidar, Joachim	Ons 5.mai kl 16:00
(FIX: Evt. skrive om DatabaseDAO)	Reidar, Joachim, Stian	Ons 5.mai kl 16:00
Oppdatere Brukstilfeller & Visjon	Andre	Ons 5.mai kl 16:00
Oppdatere domenemodell & begrepsliste	Ilon	Ons 5.mai kl 16:00
Oppdatere Testlogg iht siste tester	Kai, Ilona, Andre	Ons 5.mai kl 16:00
Oppdatere systemarkitektur (!)	Daniel	Ons 5.mai kl 16:00
Oppdatere Databasemodell	Reidar	Ons 5.mai kl 16:00
Oppdatere Designmodell	Stian	Ons 5.mai kl 16:00
TESTING: Teste tingene som har blitt fikset etter betatestingen	Andre, Ilona, Kai	Ons 5.mai kl 16:00
Logg for iterasjon 6	Joachim	Ons 5.mai kl 23:59
(Sluttrapport m/ refleksjonsnotat)	Hele gruppen	Tors 6.mai kl 23:59

Fase- og iterasjonsmål

Milepæl 1 - Inception

1. iterasjon

- Få klarhet i brukstilfeller.
- Produsere de fleste artefakter.
- Kundemøter.
- Enighet om visjon.
- Kostnadsestimat (i timer)
-

Milepæl 2 - Elaboration

- Fastsette endelige brukstilfeller
- Plan for arkitektur, samt strukturering på programmeringen

2. iterasjon

- Kjørende betaversjon
- Finne ut av rette prioriteringer

3. iterasjon

- (Mer) fungerende betaversjon, vises for kunde
- Ha fastsatt rette prioriteringer
- Plan for iterasjon 3
- Plan for construction

Milepæl 3 - Construction

- Programmere ferdig, i prioritert rekkefølge
- Feiltesting

4. iterasjon

- Programmere følgende;
  - Se reservasjon og tildelinger for en hytte — Meget høy
  - Reservere hytte (før loddtrekning) — Meget høy
  - Reservere hytte (etter loddtrekning, hytte ledig) — Meget høy
  - Loddtrekning og tildeling av hytte — Meget høy
  - Akseptere tildelt hytte — Meget høy
  - Avbestille tildelt hytte før 2-ukers frist — Meget høy
  - Se informasjon om hytte — Høy
  - Avbestille tildelt hytte etter 2-ukers frist — Høy
- Testing

5. iterasjon

- Programmere følgende;
  - Legge inn ferieperioder (ADMIN) — Meget høy
  - Rapport tildelinger tilbake i tid m/ pris+++ (ADMIN) — Meget høy
  - Legge inn relativ frist for avbestilling (ADMIN) — Høy
  - Legge inn reservasjoner for andre (ADMIN) — Høy

- Dersom tid, programmere følgende;
  - Reservere hytte (etter loddtrekning, hytte tildelt) — Middels
  - Avbestille reservert ikke-tildelt hytte — Middels
  - Se oversikt over "Mine reservasjoner" — Lav
  - Elektronisk hyttebok — Lav
  - Rapport tildelinger i ferier (ADMIN) — Middels
  - Legge inn viktig hytteinfo, f.eks. priser (ADMIN) — Middels
  - Endre manuelt på reservasjoner og tildelinger (ADMIN) — Middels
  - Stenge hytte for reservasjoner i perioder (ADMIN) — Middels
  - Legge inn parametere for karanteneregler (ADMIN) — Middels
  - Legge inn relativt tidspunkt for loddtrekning (ADMIN) — Lav
  - Legge inn generell hytteinfo, f.eks. tekst og bilder (ADMIN) — Lav
  - Legge til nye hytter i systemet (ADMIN) — Lav
  - Svartelisting av enkeltansatte (ADMIN) — Lav
  - Rapport reservasjoner og tildelinger frem i tid (ADMIN) — Lav

- Redigere maler for e-poster (ADMIN) — Lav

- Testing

Milepæl 4 - Transition

– Ferdig :-)

6. iterasjon

Releases

Versjon 1.0

Fredag 9.april 2010

- Fungerende prototype.
- Fokus på at reservering av hytte fungerer.

Versjon 1.1

Fredag 16.april 2010

- Fungerende alpha
- E-post utsendelse
- Unik URL

Versjon 1.2

Fredag 23.april 2010

- Fungerende beta
- De fleste funksjoner skal være implementert
- Alle brukstilfeller som vi har valgt å prioritere skal fungere

Versjon 1.5 (final)

Fredag 30.april 2010

- Alle funksjoner vi har valgt å implementere skal fungere

Kostnadsbudsjett

Kostnadsbudsjettet er estimert til 2 timer per dag per ansatt. Prosjektet foregår over en periode på 7 uker, som tilsvarer 35 dager. Vi er 7 ansatte i gruppen. Estimert antall timer er da 70 timer per person — totalt 490 timer. Møter kommer utenom, og budsjetteres med totalt 78 timer fordelt over hele prosjektet.

Disse timene skal fordeles slik;

- Inception — 50 timer arbeid + 28 timer møter
  - Iterasjon 1 — 50 timer + 28 timer møter
- Elaboration — 160 timer arbeid + 10 timer møter
  - Iterasjon 2 — 80 timer arbeid + 5 timer møter
  - Iterasjon 3 — 80 timer arbeid + 5 timer møter
- Construction — 210 timer arbeid + 20 timer møter
  - Iterasjon 4 — 105 timer arbeid + 10 timer møter
  - Iterasjon 5 — 105 timer arbeid + 10 timer møter
- Transition —70 timer arbeid + 20 timer møter
  - Iterasjon 6 — 70 timer arbeid + 20 timer møter

Testplan

Målet med testplanen er å avdekke eventuelle logikk- og programmeringsfeil i applikasjonen.

Hvilket system skal vi teste?

HytteWeb.

Hva skal vi teste i systemet?

Alt. Frontend og Admin-sider. Reservering. Loddtrekning. Automatikken. Statistikk-sidene. Rapport-generering. E-post utsending. Mer nøyaktig beskrivelse av hva som skal testes, vil fremgå av test-artefaktet («Testlogg.doc»).

Hva skal vi ikke teste

Brukstilfeller og features som ikke er en del av kravspesifikasjonen. Dersom tiden ikke strekker til, vil brukstifeller merket med «Lav» og evt. «Middels» ikke bli testet/utført.

Godkjenningskriterier

- Hvert brukstilfelle skal tilfredsstille de krav kunden har gitt på forhånd
- Eventuelle feil/bugs, det være seg systemet i seg selv, eller utseende, som ikke har noen kritisk betydning, vil ikke i utgangspunktet bli prioritert rettet.

Ansvarsfordeling

Utviklerne selv skal i hovedsak ta seg av JUnit-testing, men med evt. bistand fra testere. Alle som har Analytiker-rolle i Inception/Elaboration, vil gå over til å være tester i Construction-fasen. De vil fortsatt beholde ansvar for sine respektive artefakter, og holde disse oppdatert.

Tidsplan

Testing vil til en viss grad utføres fortløpende av utvilkere, men testere skal gjennomføre mer omfattende tester underveis som brukstilfeller begynner å bli klare. Plan for når disse testene vil bli utført for hver enkelt iterasjon.

Styring og kontroll

Håndtering av brukerkrav

Brukerens krav skal, så langt det lar seg praktisk gjenomføre, tilfredsstilles.

Kvalitetskontroll

Alle som er delaktige i gruppen.

Rapportering og målinger

Sluttrapport skal inn ved prosjektets slutt.

Håndtering av risiko

Som angitt i risikolisten.

Testlogg

De som har hatt testerrollen, har gjort testinger fortløpende som utviklere har jobbet seg fremover. Det ble gjort omfattende tester før betatestingen, samt etter retting av feil som ble oppdaget under betatesten.

Rapportene 1-8 gjelder krav merket «meget høy».
Rapportene 9-12 gjelder krav merket «høy».
Rapportene 13-16 gjelder bugs funnet under betatesten.

Test nummer	1		
Hvilket system?	HytteWeb		
Hva tester vi?	Se reservasjoner og tildelinger for en hytte		
Hvordan skal det vi tester oppføre seg i forhold planen?	Reservasjoner og tildelinger av en hytte skal vises i kalenderen til den aktuelle hytten med hjelp av fargekoder. Med rødt for tildelt hytte, grønn for ledig hytte osv.		
Hva skal til for at testen er godkjent?	At kalenderen er riktig og viser riktige data i forhold til bestillinger.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Ja		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?	-		
Testen er utført av	Ilona/Kai		
Dato	27.04.2010 kl 13:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	26.04.10	Første utkast	Ilona/Kai

Test nummer	2		
Hvilket system?	HytteWeb		
Hva tester vi?	Reservere hytte (før loddtrekning)		
Hvordan skal det vi tester oppføre seg i forhold planen?	En ansatt skal kunne reservere hytte ved å trykke på de dagene i kalenderen han/hun ønsker å leie hytten, deretter skal man skrive inn sitt brukernavn og bekreftelseslinken blir automatisk sendt til den ansattes email.		
Hva skal til for at testen er godkjent?	Man skal ha mulighet til å velge de dagene man ønsker, Bekreftelseslinken må sendes til riktig epost adresse.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Ved bestilling kan brukeren kun bestille perioder (man-tors eller fre-søn eller hele perioden som er markert som ferie). Det er mulig å bestille èn enkel dag hvis en klikker på nytt på den dagen en vil ha. Hvis en ønsker to dager må man legge inn to bestillinger.		
Kan brukstilfellet markeres som utført?	Ja og Nei		
Dersom nei, er utviklere underrettet om problemet?	Brukstilfelle sier at brukeren velger dager, mens programmet kun gir valg for: èn dag, man-tors, fre-lørd, eller hele ferieperioden.		
Testen er utført av	Ilona/Kai		
Dato	27.04.2010 kl 14:13		
Endringer	**Dato**	**Beskrivelse**	**Person**
	27.04.10	Første utkast	Ilona/Kai

Test nummer	2		
Hvilket system?	HytteWeb		
Hva tester vi?	Reservere hytte (før loddtrekning)		
Hvordan skal det vi tester oppføre seg i forhold planen?	En ansatt skal kunne reservere hytte ved å trykke på de dagene i kalenderen han/hun ønsker å leie hytten, deretter skal man skrive inn sitt brukernavn og bekreftelseslinken blir automatisk sendt til den ansattes email.		
Hva skal til for at testen er godkjent?	Man skal ha mulighet til å velge de dagene man ønsker, Bekreftelseslinken må sendes til riktig epost adresse.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Ved bestilling kan brukeren kun bestille perioder (man-tors eller fre-søn eller hele perioden som er markert som ferie). Det er mulig å bestille èn enkel dag hvis en klikker på nytt på den dagen en vil ha. Hvis en ønsker to dager må man legge inn to bestillinger.		
Kan brukstilfellet markeres som utført?	Ja og Nei		
Dersom nei, er utviklere underrettet om problemet?	Brukstilfelle sier at brukeren velger dager, mens programmet kun gir valg for: èn dag, man-tors, fre-lørd, eller hele ferieperioden.		
Testen er utført av	André		
Dato	04.05.2010 kl 10:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	04.05.10	Andre utkast	André

Test nummer	3		
Hvilket system?	HytteWeb		
Hva tester vi?	Reservere hytte (etter loddtrekning, hytte ledig)		
Hvordan skal det vi tester oppføre seg i forhold planen?	Brukeren bestiller hytte, får tilsendt en mail og bekrefter denne. Bruker får deretter umiddelbart tildelt hytten, og perioden blir markert som opptatt i kalenderen. Hvis flere gjør dette samtidig, vil den som bekrefter først få hytten tildelt.		
Hva skal til for at testen er godkjent?	At mail blir sendt umiddelbart, og at perioden blir markert som opptatt hvis/når bruker bekrefter reservasjonen i mottatt mail.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Kan bestille, men når en klikker på link i mottatt mail kommer en til HytteWeb siden, og følgende feilmelding: Melding: *Bestilling ikke lagt til. Forsøk igjen. Dersom du fortsatt får feil, er reservasjonen mest sannsynlig slettet.* Kalenderen ble ikke oppdatert på noen måte i den bestilte periode.		
Kan brukstilfellet markeres som utført?	Nei		
Dersom nei, er utviklere underrettet om problemet?	Nei – ikke ennå i skrivende stund.		
Testen er utført av	Ilona/Kai		
Dato	27.04.2010 kl 13:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	27.04.10	Første utkast	Ilona/Kai

Test nummer	3		
Hvilket system?	HytteWeb		
Hva tester vi?	Reservere hytte (etter loddtrekning, hytte ledig)		
Hvordan skal det vi tester oppføre seg i forhold planen?	Brukeren bestiller hytte, får tilsendt en e-post og bekrefter denne. Bruker får deretter umiddelbart tildelt hytten, og perioden blir markert som opptatt i kalenderen. Hvis flere gjør dette samtidig, vil den som bekrefter først få hytten tildelt.		
Hva skal til for at testen er godkjent?	At e-post blir sendt umiddelbart, og at perioden blir markert som opptatt hvis/når bruker bekrefter reservasjonen i mottatt e-post.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Kan bestille, men når en klikker på link for å bekrefte reservasjon blir status på hytten satt til reservert, ikke tildelt. Man mottar bekreftelse på reservasjon, men får ikke anledning til å bekrefte tildeling. Tildeling må bekreftes ved å få tilsendt e-post fra ”Mine bestillinger”. Derfra bekreftes tildelingen ved å klikke gjeldende link. Nå får hytten status tildelt. Altså fungerer tildeling, men ikke etter beskrivelse i plan		
Kan brukstilfellet markeres som utført?	Nei		
Dersom nei, er utviklere underrettet om problemet?	Nei – ikke ennå i skrivende stund.		
Testen er utført av	André		
Dato	05.05.2010 kl 14:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	05.05.10	Andre utkast	André

Test nummer	4		
Hvilket system?	HytteWeb		
Hva tester vi?	Loddtrekning og tildeling av hytte		
Hvordan skal det vi tester oppføre seg i forhold planen?	Loddtrekningen foregår en satt dato/klokke av admin for reservert periode hvis flere enn en har bestilt hytten. Da blir mail sendt ut til den heldige (tilfeldig/random) og denne må bekrefte innen en gitt tid. Ellers blir det ny trekning av de andre på listen.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Ikke testet. Mottar mail, men når en klikker på vedlagt link kommer en til HytteWeb siden, og følgende feilmelding: Melding: *Bestilling ikke lagt til. Forsøk igjen. Dersom du fortsatt får feil, er reservasjonen mest sannsynlig slettet.*		
Kan brukstilfellet markeres som utført?	Nei		
Dersom nei, er utviklere underrettet om problemet?	Nei, ikke i skrivende stund.		
Testen er utført av	Ilona/Kai		
Dato	27.04.2010 kl 13:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	27.04.10	Første utkast	Ilona/Kai

Test nummer	4		
Hvilket system?	HytteWeb		
Hva tester vi?	Loddtrekning og tildeling av hytte		
Hvordan skal det vi tester oppføre seg i forhold planen?	Loddtrekningen foregår en satt dato/klokke av admin for reservert periode hvis flere enn en har bestilt hytten. Da blir mail sendt ut til den heldige (tilfeldig/random) og denne må bekrefte innen en gitt tid. Ellers blir det ny trekning av de andre på listen.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Ja		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?			
Testen er utført av	André		
Dato	05.05.2010 kl 16:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	27.04.10	Første utkast	Ilona/Kai
	05.05.10	Andre utkast	André

Test nummer	5		
Hvilket system?	HytteWeb		
Hva tester vi?	Akseptere tildelt hytte		
Hvordan skal det vi tester oppføre seg i forhold planen?	At bruker klikker på vedlagt link i mail fra HytteWeb systemet, og at deretter perioden blir markert som opptatt i systemet (websiden / databasen)		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Mottar mail, men når en klikker på vedlagt link kommer en til HytteWeb siden, og følgende feilmelding: Melding: *Bestilling ikke lagt til. Forsøk igjen. Dersom du fortsatt får feil, er reservasjonen mest sannsynlig slettet.*		
Kan brukstilfellet markeres som utført?	Nei		
Dersom nei, er utviklere underrettet om problemet?	Nei, ikke i skrivende stund.		
Testen er utført av	Ilona/Kai		
Dato	27.04.2010 kl 13:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	27.04.10	Første utkast	Ilona/Kai

Test nummer	5		
Hvilket system?	HytteWeb		
Hva tester vi?	Akseptere tildelt hytte		
Hvordan skal det vi tester oppføre seg i forhold planen?	At bruker klikker på vedlagt link i mail fra HytteWeb systemet, og at deretter perioden blir markert som opptatt i systemet (websiden / databasen)		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Mottar e-post om tildeling og følger vedlagt link. Man får kvittering i nettleser og per e-post om at tildeling er bekreftet. Dagene markeres som tildelt i kalender		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?			
Testen er utført av	André		
Dato	05.05.2010 kl 11:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	05.05.10	Andre utkast	André

Test nummer	6		
Hvilket system?	HytteWeb		
Hva tester vi?	Avbestille tildelt hytte før 2-ukers frist		
Hvordan skal det vi tester oppføre seg i forhold planen?	At bruker klikker på ”mine bestillinger”, fyller inn sin epost, klikker på vedlagt link i epost fra HytteWeb systemet. Bruker kommer så inn på en side med alle sine bestillinger. Her skal bruker kunne avbestille en periode/dag.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Linken ”mine bestillinger” gir feilmelding – ”**HTTP Status 404 – side ikke funnet”**		
Kan brukstilfellet markeres som utført?	Nei		
Dersom nei, er utviklere underrettet om problemet?	Nei, ikke i skrivende stund.		
Testen er utført av	Ilona/Kai		
Dato	27.04.2010 kl 13:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	27.04.10	Første utkast	Ilona/Kai

Test nummer	6		
Hvilket system?	HytteWeb		
Hva tester vi?	Avbestille tildelt hytte før 2-ukers frist		
Hvordan skal det vi tester oppføre seg i forhold planen?	At bruker klikker på ”mine bestillinger”, fyller inn sin e-post, klikker på vedlagt link til kansellering i e-post fra HytteWeb systemet. Systemet kansellerer reservasjonen og fra link i e-post vises en kvittering i nettleser og bruker mottar e-post om mottatt kansellering.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Ja		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?			
Testen er utført av	André		
Dato	05.05.2010 kl 12:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	05.05.10	Andre utkast	André

Test nummer	7		
Hvilket system?	HytteWeb		
Hva tester vi?	Legge inn ferieperioder		
Hvordan skal det vi tester oppføre seg i forhold planen?	At admin kan legge inn ferieperioder.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Testet ok, men siden hvor admin legger inn dette (”ferier”) er litt forvirrende å forstå. Pris legges inn to ganger, og en må ”registrere” to ganger. MEN siden ser bra ut. Kanskje litt forklarende tekst eller lignende kan hjelpe…		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?	-		
Testen er utført av	Ilona/Kai		
Dato	27.04.2010 kl 13:00		
Endringer	Dato	Beskrivelse	Person
	27.04.10	Første utkast	Ilona/Kai

Test nummer	8		
Hvilket system?	HytteWeb		
Hva tester vi?	Rapporter tildelinger tilbake i tid m/pris++		
Hvordan skal det vi tester oppføre seg i forhold planen?	Admin skal kunne velge å liste ut de som ikke har betalt eller alle.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Siden viser kun de som ikke har betalt.		
Kan brukstilfellet markeres som utført?	Nei		
Dersom nei, er utviklere underrettet om problemet?	Nei, ikke i skrivende stund.		
Testen er utført av	Ilona/Kai		
Dato	27.04.2010 kl 13:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	27.04.10	Første utkast	Ilona/Kai

Test nummer	8		
Hvilket system?	HytteWeb		
Hva tester vi?	Rapporter tildelinger tilbake i tid m/pris++		
Hvordan skal det vi tester oppføre seg i forhold planen?	Admin skal kunne velge å liste ut de som ikke har betalt eller alle.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Alt fungerte som det skulle.		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?			
Testen er utført av	Daniel		
Dato	5.05.2010 kl 10:00		
Endringer	Dato	Beskrivelse	Person
	27.04.10	Første utkast	Ilona/Kai
	05.05.2010	Andre test	Daniel

Test nummer	9		
Hvilket system?	HytteWeb		
Hva tester vi?	Se informasjon om hytte		
Hvordan skal det vi tester oppføre seg i forhold planen?	At vi får kort informasjon om valgte hytte med bilde. Ved klikk på ”mer informasjon” – kommer ny side med mer informasjon.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Mangelfull tekst, ellers veldig bra. Bør ha link nederst på ”mer informasjon” som kan ta brukeren direkte til Bestill/Sjekk ledighetsstatus.		
Kan brukstilfellet markeres som utført?	Ja og nei (mangelfull tekst)		
Dersom nei, er utviklere underrettet om problemet?	Nei, ikke i skrivende stund.		
Testen er utført av	Ilona/Kai		
Dato	27.04.2010 kl 13:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	27.04.10	Første utkast	Ilona/Kai

Test nummer	9		
Hvilket system?	HytteWeb		
Hva tester vi?	Se informasjon om hytte		
Hvordan skal det vi tester oppføre seg i forhold planen?	At vi får kort informasjon om valgte hytte med bilde. Ved klikk på ”mer informasjon” – kommer ny side med mer informasjon.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Er kommet bra med tekst nå, fungerer bra. Kunne hatt en link til kalenderen på slutten av informasjonen.		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?			
Testen er utført av	Daniel		
Dato	05.05.2010 kl 10:00		
Endringer	Dato	Beskrivelse	Person
	27.04.10	Første utkast	Ilona/Kai
	05.05.2010	Andre utkast	Daniel

Test nummer	10		
Hvilket system?	HytteWeb		
Hva tester vi?	Avbestille tildelt hytte etter 2-ukers frist		
Hvordan skal det vi tester oppføre seg i forhold planen?	At bruker kan gå inn på ”mine bestillinger” og avbestille hyttereservasjonen sin.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Linken ”mine bestillinger” gir feilmelding – ”**HTTP Status 404 – side ikke funnet**”		
Kan brukstilfellet markeres som utført?	Nei		
Dersom nei, er utviklere underrettet om problemet?	Nei, ikke i skrivende stund.		
Testen er utført av	Ilona/Kai		
Dato	27.04.2010 kl 13:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	27.04.10	Første utkast	Ilona/Kai

Test nummer	10		
Hvilket system?	HytteWeb		
Hva tester vi?	Avbestille tildelt hytte etter 2-ukers frist		
Hvordan skal det vi tester oppføre seg i forhold planen?	At bruker klikker på ”mine bestillinger”, fyller inn sin e-post, klikker på vedlagt link til kansellering i e-post fra HytteWeb systemet. Bruker får nå en kvittering via link og e-post at tildeling er kansellert.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	ja		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?			
Testen er utført av	André		
Dato	05.05.2010 kl 13:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	05.05.10	Andre utkast	André

Test nummer	11		
Hvilket system?	HytteWeb		
Hva tester vi?	Legge inn relativ frist for avbestilling		
Hvordan skal det vi tester oppføre seg i forhold planen?	At admin kan bestemme avbestillingsfristen for perioden		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Valgene er tilstedet for admin under ”instillinger”		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?	-		
Testen er utført av	Ilona/Kai		
Dato	27.04.2010 kl 13:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	27.04.10	Første utkast	Ilona/Kai

Test nummer	12		
Hvilket system?	HytteWeb		
Hva tester vi?	Legge inn reservasjoner for andre		
Hvordan skal det vi tester oppføre seg i forhold planen?	At admin kan reservere hytter for andre.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Linken ”reservasjoner” fungerer ikke. Ved å gå inn på hytter, og velge hytte, deretter ”Legge inn reservasjon” kan admin reservere hytte til en bruker ved å oppgi brukers epost.		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?	-		
Testen er utført av	Ilona/Kai		
Dato	27.04.2010 kl 13:00		
Endringer	**Dato**	**Beskrivelse**	**Person**
	27.04.10	Første utkast	Ilona/Kai

Test nummer	13		
Hvilket system?	HytteWeb		
Hva tester vi?	Reservasjon over 2 mnd		
Hvordan skal det vi tester oppføre seg i forhold planen?	At vi får reservert dager som går over 2 måneder.		
Hva skal til for at testen er godkjent?	Se over.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Reservasjonen vil bli gjort, og du får mail om bekreftelse og kan bekrefte. Men det blir ikkje markert ordentlig i kalenderen.		
Kan brukstilfellet markeres som utført?	Nei.		
Dersom nei, er utviklere underrettet om problemet?			
Testen er utført av	Daniel		
Dato	05.05.2010 kl 10:00		
Endringer	Dato	Beskrivelse	Person
	05.05.2010	Første utkast	Daniel

Test nummer	14		
Hvilket system?	HytteWeb		
Hva tester vi?	Etter loddtrekning fikk man ikke tildelt hytte		
Hvordan skal det vi tester oppføre seg i forhold planen?	Hytte skal tildeles etter loddtrekning er utført.		
Hva skal til for at testen er godkjent?	At hytte blir tildelt etter trekning.		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Ja		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?	-		
Testen er utført av	André		
Dato	05.05.2010 kl 13:20		
Endringer	**Dato**	**Beskrivelse**	**Person**
	05.05.10	Første utkast	André

Test nummer	15		
Hvilket system?	HytteWeb		
Hva tester vi?	Nullpointer ved avbestilling av tildeling		
Hvordan skal det vi tester oppføre seg i forhold planen?	Ved avbestilling skal en kvittering vises etter man klikker på link til avbestilling i mottatt e-post fra systemet.		
Hva skal til for at testen er godkjent?	At ovenstående fungerer uten feilmelding		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Ja		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?	-		
Testen er utført av	André		
Dato	05.05.2010 kl 13:30		
Endringer	**Dato**	**Beskrivelse**	**Person**
	05.05.10	Første utkast	André

Test nummer	16		
Hvilket system?	HytteWeb		
Hva tester vi?	Adminrapport tilbake i tid m/ visning av betalende		
Hvordan skal det vi tester oppføre seg i forhold planen?	Rapport tilbake i tid skal også vise betalende.		
Hva skal til for at testen er godkjent?	At ovenstående fungerer uten feilmelding		
Hva var resultatet av testen? Fungerte alt slik det skulle?	Ja		
Kan brukstilfellet markeres som utført?	Ja		
Dersom nei, er utviklere underrettet om problemet?	-		
Testen er utført av	André		
Dato	05.05.2010 kl 13:45		
Endringer	**Dato**	**Beskrivelse**	**Person**
	05.05.10	Første utkast	André

Risikoliste

Formålet med dette dokumentet

Analysere og gi en oversikt over risikoer som kan oppstå i løpet av prosjektperioden, hva slags konsekvenser de kan få, hvor stor sannsynlighet det er for at risikioen inntreffer, samt forebyggende tiltak og tiltak ved eventuell inntreffelse. Dokumentet skal også gi oss en oversikt over de mest alvorlige risikioene.

Oversikt over de mest alvorlige risikoene

ID	Risiko	Betydning
5	Kravspesifikasjon endres underveis	48
16	Loddtrekning fungerer ikke som planlagt	45
15	E-post utsending fungerer ikke	40
8	Ikke ferdig til milepæler	36
7	Satt for høye mål for produktet	32
14	Får ikke testet produktet skikkelig	25

Risikoer i prosjektet

ID	1		
Risiko	Ikke bli ferdig i tide		
Konsekvens	Ikke godkjent oppgave?		
Sanns.het (1-10)	1		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	10		
Tiltak	God planlegging.		
Tiltak ved inntreffelse	Ikke godkjent oppgave?		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim
	30.04.10	Revidert	Joachim

ID	2		
Risiko	Sykdom		
Konsekvens	Mindre resurser. Rekker ikke milepæler.		
Sanns.het (1-10)	4		
Alvorlighet (1-10)	6		
Betydning (sanns. * alvor)	24		
Tiltak	Alle skal ha kjennskap til hva de forskjellige gjør, slik at andre kan ta over arbeidet dersom sykdom inntreffer.		
Tiltak ved inntreffelse	Konflikthåndtering. Samarbeidsavtalen. Reforhandle kravspesifikasjon. Evt. kontakte veileder/kunde.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	3		
Risiko	Mye fravær		
Konsekvens	Mindre resurser. Rekker ikke milepæler. Dårlig miljø/stemning.		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	5		
Betydning (sanns. * alvor)	10		
Tiltak	God planlegging. Få alle til å føle seg inkludert.		
Tiltak ved inntreffelse	Snakke med vedkommende før det går ut over resten av gruppen. Konflikthåndtering. Samarbeidsavtalen. Reforhandle kravspesifikasjon. Evt. kontakte veileder/kunde.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	4		
Risiko	Tap av data		
Konsekvens	Mister arbeid. «Går tilbake i tid».		
Sanns.het (1-10)	1		
Alvorlighet (1-10)	9		
Betydning (sanns. * alvor)	9		
Tiltak	Backup. Hyppig bruk av «commit».		
Tiltak ved inntreffelse	Hente inn backup/revisjon.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	5		
Risiko	Kravspesifikasjon endrer seg underveis		
Konsekvens	Kode/logikk må endres.		
Sanns.het (1-10)	8		
Alvorlighet (1-10)	6		
Betydning (sanns. * alvor)	48		
Tiltak	Sette en baseline før utvikling starter.		
Tiltak ved inntreffelse	For store endringer ift. baseline kan ikke godkjennes. Små endringer og/eller tillegg kan gjøres dersom det er praktisk gjennomførbart.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	6		
Risiko	Enkeltpersoner «melder seg ut»		
Konsekvens	Mister resurser. Dårlig miljø.		
Sanns.het (1-10)	1		
Alvorlighet (1-10)	7		
Betydning (sanns. * alvor)	7		
Tiltak	Få alle til å føle seg viktige.		
Tiltak ved inntreffelse	Få vedkommende til å føle seg viktig. Eventuelt sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	7		
Risiko	Satt for høye mål for produktet		
Konsekvens	Når ikke målet. Rekker ikke bli ferdig.		
Sanns.het (1-10)	4		
Alvorlighet (1-10)	8		
Betydning (sanns. * alvor)	32		
Tiltak	Sette realistiske mål. Prioritere de viktigste funksjonene.		
Tiltak ved inntreffelse	Kutte funksjonaliteter.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim
	23.04.10	Revurdere sannsynlighet/alvor	Joachim

ID	8		
Risiko	Ikke ferdig til milepæler		
Konsekvens	Forsinket. Oppgaver blir forskjøvet til neste milepæl.		
Sanns.het (1-10)	6		
Alvorlighet (1-10)	6		
Betydning (sanns. * alvor)	36		
Tiltak	God planlegging. God oppfølging.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	9		
Risiko	Kompabilitetsproblemer ved overlevering til kunde		
Konsekvens	Kunde sitter uten fungerende program.		
Sanns.het (1-10)	1		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	10		
Tiltak	Forhøre oss med kunden om at applikasjonen kan kjøres på deres systemer.		
Tiltak ved inntreffelse	Kjøre på eksternt system, eller sette opp ny tjener.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	10		
Risiko	Mangel på resurser		
Konsekvens	Rekker ikke tidsfrister. Rekker ikke bli ferdig.		
Sanns.het (1-10)	1		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	10		
Tiltak	God planlegging. Ha nok resurser.		
Tiltak ved inntreffelse	Hente inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	11		
Risiko	Konflikter		
Konsekvens	Uenighet. Forsinkelser. Utestengelse av deltaker.		
Sanns.het (1-10)	1		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	10		
Tiltak	Samarbeidskontrakt.		
Tiltak ved inntreffelse	Konflikthåndtering iht. samarbeidskontrakt.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	12		
Risiko	Sitter fast i programmeringen		
Konsekvens	Får ikke implementert alle funksjoner.		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	9		
Betydning (sanns. * alvor)	18		
Tiltak	Sørge for å ha folk med kompetanse.		
Tiltak ved inntreffelse	Skaffe folk med kompetanse. Forenkle logikken/funksjonene/ brukstilfellene i programmet.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim
	23.04.10	Revurdere sannsynlighet/alvor	Joachim

ID	13		
Risiko	Produktet fungerer ikke slik som forventet		
Konsekvens	Ikke-fungerende applikasjon		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	8		
Betydning (sanns. * alvor)	16		
Tiltak	Sette av god tid til kravspesifikasjon og testing.		
Tiltak ved inntreffelse	Testing. Fikse feilene. Forenkle logikken. Enklere funksjonaliteter, så lenge det ikke går på bekostning av kravspesifiseringen.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	14		
Risiko	Får ikke testet produktet skikkelig		
Konsekvens	Feil. Bugs. Ikke-fungerende applikasjon.		
Sanns.het (1-10)	5		
Alvorlighet (1-10)	5		
Betydning (sanns. * alvor)	25		
Tiltak	Sette av tid til feiltesting.		
Tiltak ved inntreffelse	Sette inn ekstra resurser for feiltesting.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	15		
Risiko	E-post utsending fungerer ikke		
Konsekvens	Feil. Får ikke bekreftet reservering av hytte		
Sanns.het (1-10)	4		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	40		
Tiltak	Implementere det korrekt. Sette av tid til testing.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	16.04.10	Første utkast	Joachim
	23.04.10	Revurdere sannsynlighet/alvor	Joachim

ID	16		
Risiko	Loddtrekning fungerer ikke som planlagt		
Konsekvens	Feil reservasjoner. Kaos.		
Sanns.het (1-10)	5		
Alvorlighet (1-10)	9		
Betydning (sanns. * alvor)	45		
Tiltak	Implementere det korrekt. Sette av tid til testing.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	16.04.10	Første utkast	Joachim
	23.04.10	Revurdere sannsynlighet/alvor	Joachim

ID	17		
Risiko	Avbestilling før/etter tildeling fungerer ikke		
Konsekvens	Folk får ikke avbestillt		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	20		
Tiltak	Implementere det korrekt. Sette av tid til testing.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.04.10	Første utkast	Joachim

ID	18		
Risiko	Instillinger på admin-siden		
Konsekvens	Får ikke satt mange variabler som mange andre funksjoner er avhengig av.		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	20		
Tiltak	Implementere det korrekt. Sette av tid til testing.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.04.10	Første utkast	Joachim

ID	19		
Risiko	Ikke godkjent akseptansetesting		
Konsekvens	Kan ikke levere applikasjon til kunde. Rekker ikke deadline.		
Sanns.het (1-10)	1		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	10		
Tiltak	Testing. Betatesting.		
Tiltak ved inntreffelse	Forandre på ting som evt. har gjort at akseptansetestingen ikke ble godkjent.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	30.04.10	Første utkast	Joachim

ID	20		
Risiko	Bugs/feil som ikke har blitt funnet		
Konsekvens	Uønskede feil kan oppstå.		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	20		
Tiltak	Testing. Betatesting. Akseptansetesting.		
Tiltak ved inntreffelse	Lokalisere, og deretter fikse problemet.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	30.04.10	Første utkast	Joachim

Iterasjonsplan 1

Mål for iterasjonen

Sentrale mål for fasen

- En fungerende prototype skal lages
- Fokus på at hyttereservasjon fungerer
- Få avklart kravspesifikasjon (med fokus på logikk/loddtrekning-delen)

De mest alvorlige risikoene

- Ingen spesiellle alvorlige risikoer funnet
- Startfasen av prosjektet — det meste er åpent her

Valgte brukstilfeller

- Ingen — bli enige om brukstilfeller etc sammen med kunden

Andre krav

- Finne kundens behov
- Holde tidsbudsjettet

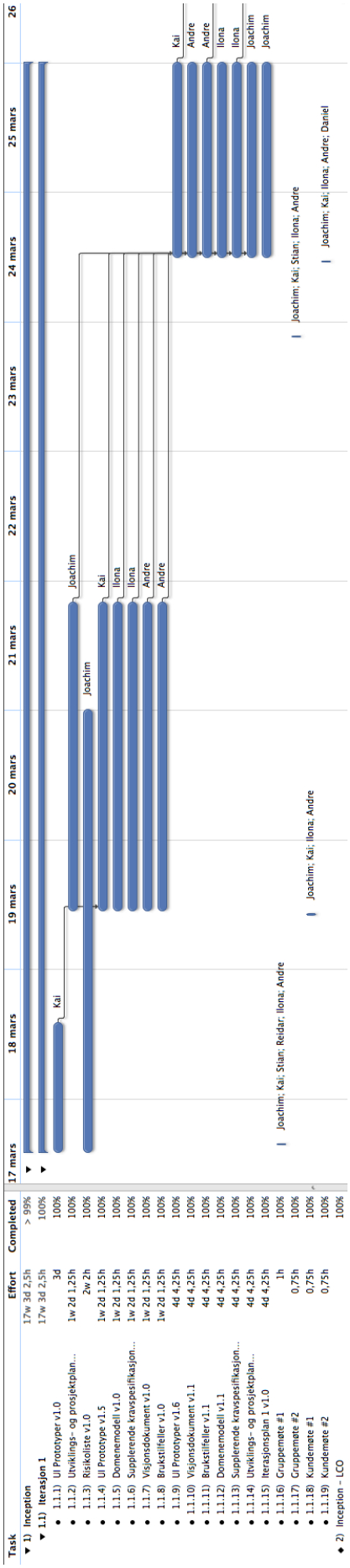
Bemanning og budsjett

5 personer. Snitt på 2 timer per dag. Frem til 26.mars. 5 dager. 10 timer per person. 50 timer totalt. Videre er 28 timer satt av til møter.

Aktivitetsplan

Arbeidsfordeling

Hva?	Hvem?	Når?
UI Prototype v1.0	Kai	Fre 19.mars kl 10:00
Utviklings- og prosjektplan v1.0	Joachim	Tirs 23.mars kl 12:00
Risikoliste v1.0	Joachim	Tirs 23.mars kl 12:00
UI Prototype v1.5	Kai	Tirs 23.mars kl 12:00
Domenemodell v1.0	Ilona	Tirs 23.mars kl 12:00
Supplerende kravspesifikasjon v1.0	Ilona	Tirs 23.mars kl 12:00
Visjonsdokument v1.0	Andre	Tirs 23.mars kl 12:00
Brukstilfeller v1.0	Andre	Tirs 23.mars kl 12:00
UI Prototype v1.6	Kai	Tors 25.mars kl 23:59
Visjonsdokument v1.1	Andre	Tors 25.mars kl 23:59
Brukstilfeller v1.1	Andre	Tors 25.mars kl 23:59
Domenemodell v1.1	Ilona	Tors 25.mars kl 23:59
Supplerende kravspesifikasjon v1.1	Ilona	Tors 25.mars kl 23:59
Utviklings- og prosjektplan v1.1	Joachim	Tors 25.mars kl 23:59
Iterasjonsplan 2 v1.0	Joachim	Tors 25.mars kl 23:59



(Merk at «Effort»-feltet ikke stemmer overens med hva vi har planlagt i skjemaet under).

Kriterier for iterasjonens ulike delmål

Samsvarer med kundens ønske

Status og måloppnåelse ved iterasjonens slutt

Mål for iterasjonen ble nådd. Kundens ønske iht UI og brukstilfelle ivaretatt. Videre er en liste over estimert vs. reell medgått tid;

Hva?	Planlagt	Reell
UI Prototype v1.0	7	7,5
Utviklings- og prosjektplan v1.0	3	2,5
Risikoliste v1.0	3	3,0
UI Prototype v1.5	5	4,75
Domenemodell v1.0	3	1,3
Supplerende kravspesifikasjon v1.0	3	2,25
Visjonsdokument v1.0	3	3,25
Brukstilfeller v1.0	3	3,75
UI Prototype v1.6	5	4,0
Visjonsdokument v1.1	3	5,0
Brukstilfeller v1.1	3	1,25
Domenemodell v1.1	2	3,30
Supplerende kravspesifikasjon v1.1	2	1,25
Utviklings- og prosjektplan v1.1	2	2,0
Iterasjonsplan 2 v1.0	3	1,75
Møter	28	24,5

TOTALT	78	71,35

Iterasjonsplan 2

Mål for iterasjonen

Sentrale mål for fasen

- En fungerende prototype skal lages
- Fokus på at hyttereservasjon fungerer
- Få avklart kravspesifikasjon (med fokus på logikk/loddtrekning-delen)

De mest alvorlige risikoene

ID	5		
Risiko	Kravspesifikasjon endrer seg underveis		
Konsekvens	Kode/logikk må endres.		
Sanns.het (1-10)	8		
Alvorlighet (1-10)	6		
Betydning (sanns. * alvor)	48		
Tiltak	Sette en baseline før utvikling starter.		
Tiltak ved inntreffelse	For store endringer ift. baseline kan ikke godkjennes. Små endringer og/eller tillegg kan gjøres dersom det er praktisk gjennomførbart.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	13		
Risiko	Produktet fungerer ikke slik som forventet		
Konsekvens	Ikke-fungerende applikasjon		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	8		
Betydning (sanns. * alvor)	16		
Tiltak	Sette av god tid til kravspesifikasjon og testing.		
Tiltak ved inntreffelse	Testing. Fikse feilene. Forenkle logikken. Enklere funksjonaliteter, så lenge det ikke går på bekostning av kravspesifiseringen.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

Valgte brukstilfeller

- Hyttereservasjon

Andre krav

- Fungerende prototype
- Holde tidsbudsjettet

Bemanning og budsjett

7 personer. Snitt på ca 2,3 timer per dag. Frem til 9.april. Minus noen dager i påskeferien. 5 dager. 11-12 timer per person. 80 timer totalt. 5 timer er avsatt til møter.

Aktivitetsplan

Arbeidsfordeling

Hva?	Hvem?	Når?
Starte på kodingen av de mest kritiske .jsp-filene, så de matcher UI-prototyp v1.6.	Kai	Tirs 6.april kl 10:00
Systemarkitektur v1.0	Daniel	Tirs 6.april kl 10:00
Databasemodell v1.0	Reidar	Tirs 6.april kl 10:00
Designmodell	Stian	Tirs 6.april kl 10:00
Kommet i gang med koding av applikasjon. Fokus på hyttereservering.	Stian, Reidar, Kai	Tirs 6.april kl 10:00
Brukstilfeller v1.2 (når L-P legger ut mer info på It's Learning vedr. kravspesifikiasjon)	Andre	Tirs 6.april kl 10:00
Supplerende kravspesifikasjon (når L-P legger ut mer info på It's Learning vedr. kravspesifikiasjon)	Ilona	Tirs 6.april kl 10:00
Kjørende prototype av HytteWeb	Stian, Reidar, Kai	Fre 9.april kl 12:00
Iterasjonsplan 3	Joachim	Fre 9.april kl 12:00

(Merk at «Effort»-feltet ikke stemmer overens med hva vi har planlagt i skjemaet under).

Kriterier for iterasjonens ulike delmål

- Fungerende prototype av programmet.

Status og måloppnåelse ved iterasjonens slutt

Mål for iterasjonen ble nådd. Kjørende prototype av HytteWeb på plass. Ser ingen umiddelbare forandringer som trengs å gjøres på risikolisten. Videre er en liste over estimert vs. reell medgått tid;

Hva?	Planlagt	Reell
Starte på kodingen av de mest kritiske .jsp-filene, så de matcher UI-prototyp v1.6.	5	7,5
Systemarkitektur v1.0	5	2
Databasemodell v1.0	5	7
Designmodell	5	8
Kommet i gang med koding av applikasjon. Fokus på hyttereservering.	20	24
Brukstilfeller v1.2 (når L-P legger ut mer info på It's Learning vedr. kravspesifikiasjon)	2	1,25
Supplerende kravspesifikasjon (når L-P legger ut mer info på It's Learning vedr. kravspesifikiasjon)	2	0,75
Kjørende prototype av HytteWeb	34	31,75
Iterasjonsplan 3	2	1,75
Møter	5	9,75

TOTALT	85	93,75

Iterasjonsplan 3

Mål for iterasjonen

Sentrale mål for fasen

- En fungerende prototype skal lages
- Fokus på at reservering av hytte gjennomføres

De mest alvorlige risikoene

ID	5		
Risiko	Kravspesifikasjon endrer seg underveis		
Konsekvens	Kode/logikk må endres.		
Sanns.het (1-10)	8		
Alvorlighet (1-10)	6		
Betydning (sanns. * alvor)	48		
Tiltak	Sette en baseline før utvikling starter.		
Tiltak ved inntreffelse	For store endringer ift. baseline kan ikke godkjennes. Små endringer og/eller tillegg kan gjøres dersom det er praktisk gjennomførbart.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	13		
Risiko	Produktet fungerer ikke slik som forventet		
Konsekvens	Ikke-fungerende applikasjon		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	8		
Betydning (sanns. * alvor)	16		
Tiltak	Sette av god tid til kravspesifikasjon og testing.		
Tiltak ved inntreffelse	Testing. Fikse feilene. Forenkle logikken. Enklere funksjonaliteter, så lenge det ikke går på bekostning av kravspesifiseringen.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

Valgte brukstilfeller

- Hyttereservasjon
- Unik ID

Andre krav

- Fungerende prototype
- Holde tidsbudsjettet

Bemanning og budsjett

7 personer. Snitt på ca 2,3 timer per dag. Frem til 15.april. 5 dager. 11-12 timer per person. 80 timer totalt. 5 timer er avsatt til møter.

Aktivitetsplan

Arbeidsfordeling

Hva?	Hvem?	Når?
Brukstilfeller i henhold til «Prioriteringslisten» til kunde (lagt ut på It's Learning)	Ilona	Tors 15.april kl 10:00
Systemarkitektur	Daniel	Tors 15.april kl 10:00
Logg for inception-fase/iterasjon 1	Joachim	Tors 15.april kl 10:00
Logg for iterasjon 2	Joachim	Tors 15.april kl 10:00
Plan for iterasjon 4	Joachim	Tors 15.april kl 10:00
Plan for Construction-fasen	Joachim	Tors 15.april kl 10:00
UI-Prototype -- fikse småting	Kai	Tors 15.april kl 10:00
Kjørende prototype av HytteWeb	Stian, Reidar, Kai	Tors 15.april kl 10:00

(Merk at «Effort»-feltet ikke stemmer overens med hva vi har planlagt i skjemaet under).

Kriterier for iterasjonens ulike delmål

- Fungerende prototype av programmet.

Status og måloppnåelse ved iterasjonens slutt

Mål for iterasjonen ble nådd. Tidspunkt for iterasjon 3/construction-fasen måtte fremskyndes pga. kunde ikke kunne møte fredag 16.april. Kjørende prototype av HytteWeb på plass. Ser ingen umiddelbare forandringer som trengs å gjøres på risikolisten. Videre er en liste over estimert vs. reell medgått tid;

Hva?	Planlagt	Reell
Brukstilfeller i henhold til «Prioriteringslisten» til kunde (lagt ut på It's Learning)	2	1,75
Systemarkitektur	5	1,5
Logg for inception-fase/iterasjon 1	1	0,75
Logg for iterasjon 2	1	0,25
Plan for iterasjon 4	1	0,5
Plan for Construction-fasen	2	1,25
UI-Prototype -- fikse småting	5	6,75
Kjørende prototype av HytteWeb	63	65,5
Møter	5	7

TOTALT	85	85,25

Iterasjonsplan 4

Mål for iterasjonen

Sentrale mål for fasen

- Programmering starter for alvor
- Brukstilfeller skal implementeres iht prioriteringslisten

De mest alvorlige risikoene

ID	15		
Risiko	E-post utsending fungerer ikke		
Konsekvens	Feil. Får ikke bekreftet reservering av hytte		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	20		
Tiltak	Implementere det korrekt. Sette av tid til testing.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	16.04.10	Første utkast	Joachim

ID	16		
Risiko	Loddtrekning fungerer ikke som planlagt		
Konsekvens	Feil reservasjoner. Kaos.		
Sanns.het (1-10)	4		
Alvorlighet (1-10)	9		
Betydning (sanns. * alvor)	36		
Tiltak	Implementere det korrekt. Sette av tid til testing.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	16.04.10	Første utkast	Joachim

ID	7		
Risiko	Satt for høye mål for produktet		
Konsekvens	Når ikke målet. Rekker ikke bli ferdig.		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	8		
Betydning (sanns. * alvor)	16		
Tiltak	Sette realistiske mål. Prioritere de viktigste funksjonene.		
Tiltak ved inntreffelse	Kutte funksjonaliteter.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	8		
Risiko	Ikke ferdig til milepæler		
Konsekvens	Forsinket. Oppgaver blir forskjøvet til neste milepæl.		
Sanns.het (1-10)	6		
Alvorlighet (1-10)	6		
Betydning (sanns. * alvor)	36		
Tiltak	God planlegging. God oppfølging.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	13		
Risiko	Produktet fungerer ikke slik som forventet		
Konsekvens	Ikke-fungerende applikasjon		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	8		
Betydning (sanns. * alvor)	16		
Tiltak	Sette av god tid til kravspesifikasjon og testing.		
Tiltak ved inntreffelse	Testing. Fikse feilene. Forenkle logikken. Enklere funksjonaliteter, så lenge det ikke går på bekostning av kravspesifiseringen.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

Valgte brukstilfeller

- Se reservasjon og tildelinger for en hytte — Meget høy
- Reservere hytte (før loddtrekning) — Meget høy
- Reservere hytte (etter loddtrekning, hytte ledig) — Meget høy

- Loddtrekning og tildeling av hytte — Meget høy
- Akseptere tildelt hytte — Meget høy
- Avbestille tildelt hytte før 2-ukers frist — Meget høy
- Se informasjon om hytte — Høy
- Avbestille tildelt hytte etter 2-ukers frist — Høy

Andre krav

- Deploye på eple.hib.no
- Holde tidsbudsjettet

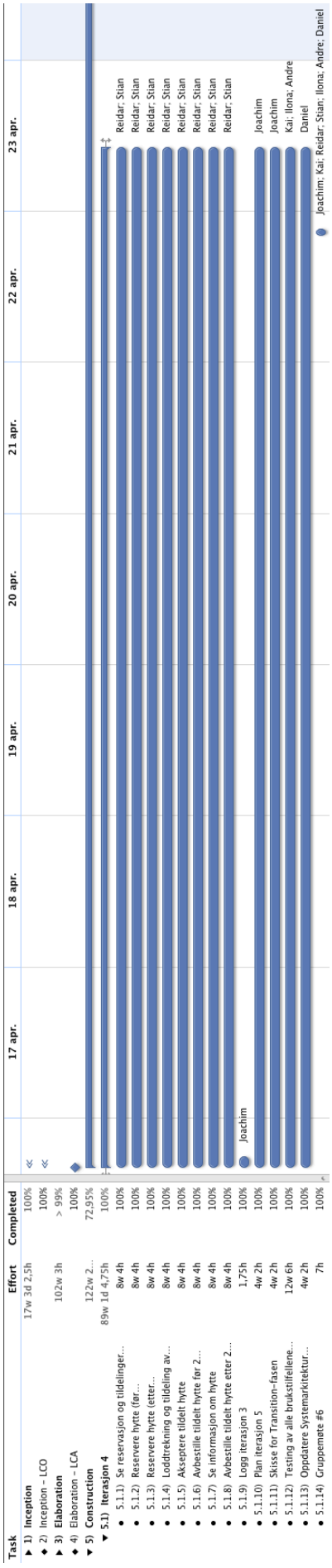
Bemanning og budsjett

7 personer. Snitt på 3 timer per dag. Frem til 23.april. 5 dager. 15 timer per person. 105 timer totalt. Videre er 10 timer satt av til møter.

Aktivitetsplan

Arbeidsfordeling

Hva?	Hvem?	Når?
Brukstilfelle: Se reservasjon og tildelinger for en hytte	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Reservere hytte (før loddtrekning)	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Reservere hytte (etter loddtrekning, hytte ledig)	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Loddtrekning og tildeling av hytte	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Akseptere tildelt hytte	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Avbestille tildelt hytte før 2-ukers frist	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Se informasjon om hytte	Stian, Reidar	Fre 23.april kl 10:00
Brukstilfelle: Avbestille tildelt hytte etter 2-ukers frist	Stian, Reidar	Fre 23.april kl 10:00
Logg iterasjon 3	Joachim	Fre 16.april kl 23:59
Plan iterasjon 5	Joachim	Fre 23.april kl 10:00
Skisse for Transition-fasen	Joachim	Fre 23.april kl 10:00
Testing av alle brukstilfellene som har blitt implementert	Ilona, Kai, André	Fre 23.april kl 10:00
Oppdatere Systemarkitektur-dokumentet i henhold til nye brukstilfeller som skal implementeres	Daniel	Fre 23.april kl 10:00



(Merk at «Effort»-feltet ikke stemmer overens med hva vi har planlagt i skjemaet under).

Kriterier for iterasjonens ulike delmål

- Testet «OK» iht. testplan

Status og måloppnåelse ved iterasjonens slutt

Målet for iterasjonen ble nådd. Vi er kommet i god gang med programmering og implementering av brukstilfeller. Etter forelesning onsdag 21.april, ble vi enige om å flytte noen datoer. Iterasjon 5/construction skal være ferdig torsdag 29.april. Da vil vi foreta betatesting med kunde, selv om dette egentlig hører til transition-fasen. Transition-fasen starter da på torsdag 29.april. Videre vil kunde foreta akseptansetesting torsdag 6.mai, og resultatene av dette vil fremligge like etter lunch. Vi har derfor forskjøvet sluttdato for transition-fasen til torsdag 6.mai. Videre er det foretatt forandringer i risikolisten.

Liste over estimert vs. reell medgått tid;

Hva?	Planlagt	Reell
Brukstilfelle: Se reservasjon og tildelinger for en hytte	5	4,75
Brukstilfelle: Reservere hytte (før loddtrekning)	10	13,75
Brukstilfelle: Reservere hytte (etter loddtrekning, hytte ledig)	10	14,5
Brukstilfelle: Loddtrekning og tildeling av hytte	10	15,25
Brukstilfelle: Akseptere tildelt hytte	10	0
Brukstilfelle: Avbestille tildelt hytte før 2-ukers frist	10	9,75
Brukstilfelle: Se informasjon om hytte	10	11,5
Brukstilfelle: Avbestille tildelt hytte etter 2-ukers frist	10	
Logg iterasjon 3	1	0,75
Plan iterasjon 5	2	1,5
Skisse for Transition-fasen	2	0,5
Testing av alle brukstilfellene som har blitt implementert	23	20
Oppdatere Systemarkitektur-dokumentet i henhold til nye brukstilfeller som skal implementeres	2	3,5
Møter	10	7,5

TOTALT	115	103,5

Iterasjonsplan 5

Mål for iterasjonen

Sentrale mål for fasen

- Programmere ferdig krav merket «meget høy» og «høy»
- Grundig testing av implementerte brukstilfeller

De mest alvorlige risikoene

ID	18		
Risiko	Instillinger på admin-siden		
Konsekvens	Får ikke satt mange variabler som mange andre funksjoner er avhengig av.		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	20		
Tiltak	Implementere det korrekt. Sette av tid til testing.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.04.10	Første utkast	Joachim

ID	17		
Risiko	Avbestilling før/etter tildeling fungerer ikke		
Konsekvens	Folk får ikke avbestillt		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	20		
Tiltak	Implementere det korrekt. Sette av tid til testing.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.04.10	Første utkast	Joachim

ID	15		
Risiko	E-post utsending fungerer ikke		
Konsekvens	Feil. Får ikke bekreftet reservering av hytte		
Sanns.het (1-10)	4		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	40		
Tiltak	Implementere det korrekt. Sette av tid til testing.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	16.04.10	Første utkast	Joachim
	23.04.10	Revurdere sannsynlighet/alvor	Joachim

ID	16		
Risiko	Loddtrekning fungerer ikke som planlagt		
Konsekvens	Feil reservasjoner. Kaos.		
Sanns.het (1-10)	5		
Alvorlighet (1-10)	9		
Betydning (sanns. * alvor)	45		
Tiltak	Implementere det korrekt. Sette av tid til testing.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	16.04.10	Første utkast	Joachim
	23.04.10	Revurdere sannsynlighet/alvor	Joachim

ID	7		
Risiko	Satt for høye mål for produktet		
Konsekvens	Når ikke målet. Rekker ikke bli ferdig.		
Sanns.het (1-10)	4		
Alvorlighet (1-10)	8		
Betydning (sanns. * alvor)	32		
Tiltak	Sette realistiske mål. Prioritere de viktigste funksjonene.		
Tiltak ved inntreffelse	Kutte funksjonaliteter.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim
	23.04.10	Revurdere sannsynlighet/alvor	Joachim

ID	8		
Risiko	Ikke ferdig til milepæler		
Konsekvens	Forsinket. Oppgaver blir forskjøvet til neste milepæl.		
Sanns.het (1-10)	6		
Alvorlighet (1-10)	6		
Betydning (sanns. * alvor)	36		
Tiltak	God planlegging. God oppfølging.		
Tiltak ved inntreffelse	Sette inn ekstra resurser.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	12		
Risiko	Sitter fast i programmeringen		
Konsekvens	Får ikke implementert alle funksjoner.		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	9		
Betydning (sanns. * alvor)	18		
Tiltak	Sørge for å ha folk med kompetanse.		
Tiltak ved inntreffelse	Skaffe folk med kompetanse. Forenkle logikken/funksjonene/ brukstilfellene i programmet.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim
	23.04.10	Revurdere sannsynlighet/alvor	Joachim

Valgte brukstilfeller

- Legge inn ferieperioder (ADMIN) — Meget høy
- Rapport tildelinger tilbake i tid m/ pris+++ (ADMIN) — Meget høy
- Legge inn relativ frist for avbestilling (ADMIN) — Høy
- Legge inn reservasjoner for andre (ADMIN) — Høy

- Dersom tid;
  - Reservere hytte (etter loddtrekning, hytte tildelt) — Middels
  - Avbestille reservert ikke-tildelt hytte — Middels
  - Se oversikt over "Mine reservasjoner" — Lav
  - Elektronisk hyttebok — Lav
  - Rapport tildelinger i ferier (ADMIN) — Middels
  - Legge inn viktig hytteinfo, f.eks. priser (ADMIN) — Middels
  - Endre manuelt på reservasjoner og tildelinger (ADMIN) — Middels
  - Stenge hytte for reservasjoner i perioder (ADMIN) — Middels
  - Legge inn parametere for karanteneregler (ADMIN) — Middels
  - Legge inn relativt tidspunkt for loddtrekning (ADMIN) — Lav
  - Legge inn generell hytteinfo, f.eks. tekst og bilder (ADMIN) — Lav
  - Legge til nye hytter i systemet (ADMIN) — Lav

Andre krav

- Deploye på eple.hib.no
- Holde tidsbudsjettet
- Alle brukstilfeller markert som «meget høy» og «høy» **må** være implementert
- Grundig testing

Bemanning og budsjett

7 personer. Snitt på 3 timer per dag. Frem til 29.april. 5 dager. 15 timer per person. 105 timer totalt. Videre er 10 timer satt av til møter.

Aktivitetsplan

Arbeidsfordeling

Hva?	Hvem?	Når?
Fullføre brukstilfeller fra iterasjon 4	Reidar, Stian	Tirs 27.april kl 10:00 Tors 29.april kl 10:00
Brukstilfelle: Legge inn ferieperioder (ADMIN)	Reidar, Stian	Tirs 27.april kl 10:00 Tors 29.april kl 10:00
Brukstilfelle: Rapport tildelinger tilbake i tid m/ pris++ (ADMIN)	Reidar, Stian	Tirs 27.april kl 10:00 Tors 29.april kl 10:00
Brukstilfelle: Legge inn relativ frist for avbestilling (ADMIN)	Reidar, Stian	Tirs 27.april kl 10:00 Tors 29.april kl 10:00
Brukstilfelle: Legge inn reservasjoner for andre (ADMIN)	Reidar, Stian	Tirs 27.april kl 10:00 Tors 29.april kl 10:00
Grundig test av alle brukstilfeller som er implementert (ved å følge testplan & malen i testlogg.doc). Utføres på Tirsdag 27.april, slik at evt. feil kan fikses på innen Torsdag 29.april kl 10:00.	Kai, Ilona, Andre	Tirs 27.april kl 23:59
Logg for iterasjon 5	Joachim	Tors 29.april kl 23:59
Iterasjonsplan 6	Joachim	Tors 29.april kl 23:59
Transitionplan	Joachim	Tors 29.april kl 23:59
Tett oppfølging & oppdatering av systemarkitekturdokumentet	Daniel	Tors 29.april kl 10:00
Brukstilfelle: De merket med «middels» og «lav» -- blir ikke tildelt resurser, men om tiden strekker til etter andre kravene er implementert, og testet «OK», så kan man gyve løs på dette.	Reidar, Stian, Joachim	(Tors 29.april kl 10)

(Merk at «Effort»-feltet ikke stemmer overens med hva vi har planlagt i skjemaet under).

Kriterier for iterasjonens ulike delmål

- Testet «OK» iht. testplan

Status og måloppnåelse ved iterasjonens slutt

Målene for denne iterasjonen ble nådd. Vi har fullført alle brukstilfeller, med unntak av «endre manuelt på reservasjoner og tildelinger». Dette ble det ikke gjort noe med, dels fordi det kanskje kan bli «etisk ukorrekt» -- administrator kan da tildele hytte(r) til folk som kanskje egentlig ikke skulle hatt hytten(e), og dels fordi det hadde blitt veldig mye koding/ logikk for å få det til å fungere sømløst. Slik det er nå, er det rettferdig for alle, og alle stiller likt (sånn sett bort ifra karantene og periode-prioritering). Videre ble betatesting-møtet med kunde/veileder (Lars-Petter) gjennomført uten for store feil. Feilene vil bli rettet innen akseptansetestingen.

Liste over estimert tid vs. reell medgått tid;

Hva?	Planlagt	Reell
Fullføre brukstilfeller fra iterasjon 4	28	30,75
Brukstilfelle: Legge inn ferieperioder (ADMIN)	10	12,5
Brukstilfelle: Rapport tildelinger tilbake i tid m/ priss++ (ADMIN)	10	5,50
Brukstilfelle: Legge inn relativ frist for avbestilling (ADMIN)	10	3,25
Brukstilfelle: Legge inn reservasjoner for andre (ADMIN)	10	2,5
Grundig test av alle brukstilfeller som er implementert (ved å følge testplan & malen i testlogg.doc). Utføres på Tirsdag 27.april, slik at evt. feil kan fikses på innen Torsdag 29.april kl 10:00.	20	15,75
Logg for iterasjon 5	1	0,75
Iterasjonsplan 6	1	0,5
Transitionplan	1	0,75
Tett oppfølging & oppdatering av systemarkitekturdokumentet	14	2,25
Brukstilfelle: De merket med «middels» og «lav» -- blir ikke tildelt resurser, men om tiden strekker til etter andre kravene er implementert, og testet «OK», så kan man gyve løs på dette.	0	35,25
Møter	10	7,75

TOTALT	115	117,5

Iterasjonsplan 6

Mål for iterasjonen

Sentrale mål for fasen

- Fikse/fjerne bugs avdekket under betatest
- Mer testing
- Oppatering av artefakter

De mest alvorlige risikoene

ID	1		
Risiko	Ikke bli ferdig i tide		
Konsekvens	Ikke godkjent oppgave?		
Sanns.het (1-10)	1		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	10		
Tiltak	God planlegging.		
Tiltak ved inntreffelse	Ikke godkjent oppgave?		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim
	30.04.10	Revidert	Joachim

ID	9		
Risiko	Kompabilitetsproblemer ved overlevering til kunde		
Konsekvens	Kunde sitter uten fungerende program.		
Sanns.het (1-10)	1		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	10		
Tiltak	Forhøre oss med kunden om at applikasjonen kan kjøres på deres systemer.		
Tiltak ved inntreffelse	Kjøre på eksternt system, eller sette opp ny tjener.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	23.03.10	Første utkast	Joachim

ID	19		
Risiko	Ikke godkjent akseptansetesting		
Konsekvens	Kan ikke levere applikasjon til kunde. Rekker ikke deadline.		
Sanns.het (1-10)	1		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	10		
Tiltak	Testing. Betatesting.		
Tiltak ved inntreffelse	Forandre på ting som evt. har gjort at akseptansetestingen ikke ble godkjent.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	30.04.10	Første utkast	Joachim

ID	20		
Risiko	Bugs/feil som ikke har blitt funnet		
Konsekvens	Uønskede feil kan oppstå.		
Sanns.het (1-10)	2		
Alvorlighet (1-10)	10		
Betydning (sanns. * alvor)	20		
Tiltak	Testing. Betatesting. Akseptansetesting.		
Tiltak ved inntreffelse	Lokalisere, og deretter fikse problemet.		
Endringer	**Dato**	**Beskrivelse**	**Person**
	30.04.10	Første utkast	Joachim

Valgte brukstilfeller

Ingen.

Andre krav

- Rette evt. tilbakemelding/feil fra betatestingen
- Endelig versjon deployet på eple innen torsdag 6.mai.
- Holde tidsbudsjettet
- Grundig testing

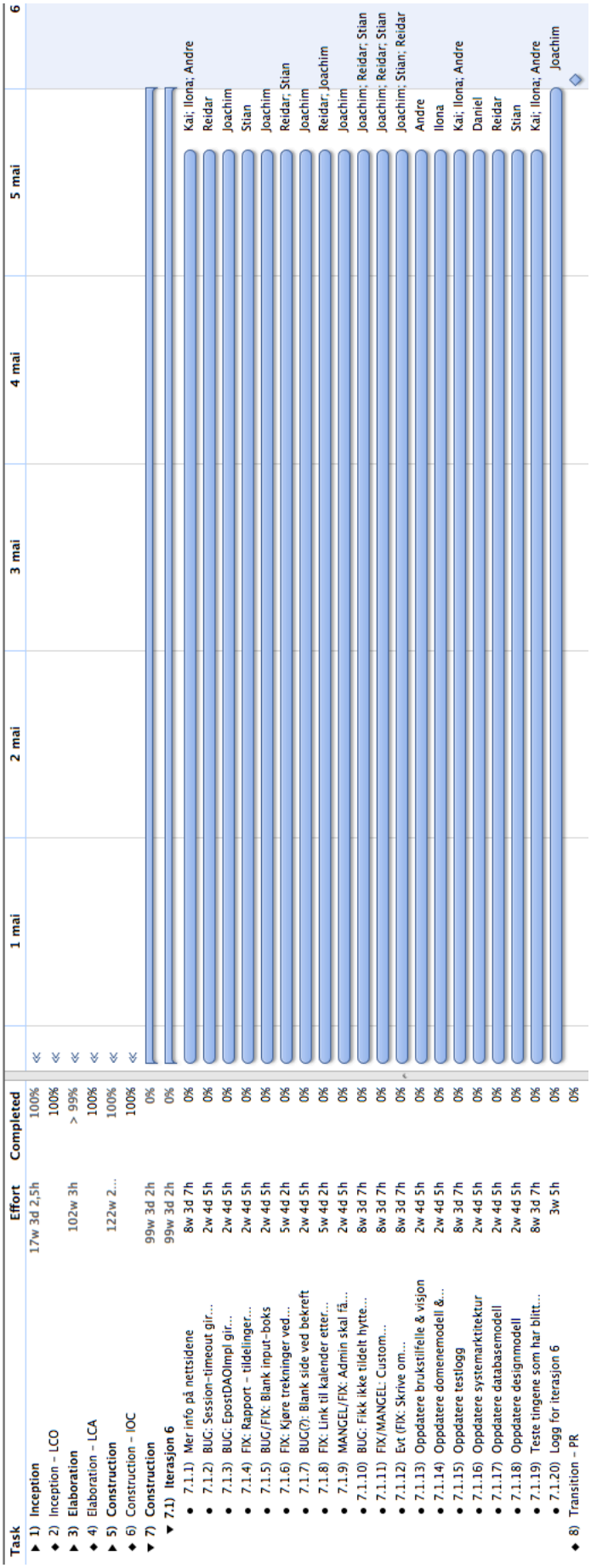
Bemanning og budsjett

7 personer. Snitt på 2-3 timer per dag. Frem til 6.mai. 4 dager. 10 timer per person. 70 timer totalt. Videre er 20 timer satt av til møter.

Aktivitetsplan

Arbeidsfordeling

Hva?	Hvem?	Når?
Legge ut mer informasjon enkelte plasser, samt endre informasjon noen plasser, ifbm. tilbakemelding vi fikk fra betatestingen	Kai, Ilona, Andre	Ons 5.mai kl 16:00
BUG: Sessio-timeout gir Nullpointer	Reidar	Ons 5.mai kl 16:00
BUG: EpostDAOImpl/BestillingDAOImpl gir Nullpointer	Joachim	Ons 5.mai kl 16:00
MANGEL/FIX: Rapport tildelinger tilbake i tid skal også kunne ta med betalte	Stian	Ons 5.mai kl 16:00
BUG/FIX: Blank text-input under innstillinger. CSS-relatert.	Joachim	Ons 5.mai kl 16:00
FIX/BUG: Ved oppstart; kjøre trekning på evt. ting som mangler trekning (f.eks. dersom server har vært nede i noen dager).	Reidar, Stian	Ons 5.mai kl 16:00
BUG(?): Blank side ved bekreftelse av bestilling. Refresh, og det funker.	Joachim	Ons 5.mai kl 16:00
FIX: «Bestilling mottatt» - lenke til kalender hvor reservasjonen fant sted	Reidar, Joachim	Ons 5.mai kl 16:00
MANGEL/FIX: Admin skal få epost ved reservasjoner.	Joachim	Ons 5.mai kl 16:00
BUG: Fikk ikke tildelt hytte etter loddtrekning	Stian, Reidar, Joachim	Ons 5.mai kl 16:00
FIX/MANGEL: Egen JSP for exception-feil	Stian, Reidar, Joachim	Ons 5.mai kl 16:00
(FIX: Evt. skrive om DatabaseDAO)	Reidar, Joachim, Stian	Ons 5.mai kl 16:00
Oppdatere Brukstilfeller & Visjon	Andre	Ons 5.mai kl 16:00
Oppdatere domenemodell & begrepsliste	Ilon	Ons 5.mai kl 16:00
Oppdatere Testlogg iht siste tester	Kai, Ilona, Andre	Ons 5.mai kl 16:00
Oppdatere systemarkitektur (!)	Daniel	Ons 5.mai kl 16:00
Oppdatere Databasemodell	Reidar	Ons 5.mai kl 16:00
Oppdatere Designmodell	Stian	Ons 5.mai kl 16:00
TESTING: Teste tingene som har blitt fikset etter betatestingen	Andre, Ilona, Kai	Ons 5.mai kl 16:00
Logg for iterasjon 6	Joachim	Ons 5.mai kl 23:59
(Sluttrapport m/ refleksjonsnotat)	Hele gruppen	Tors 6.mai kl 23:59



(Merk at «Effort»-feltet ikke stemmer overens med hva vi har planlagt i skjemaet under).

Kriterier for iterasjonens ulike delmål

- Ting skal fungere

Status og måloppnåelse ved iterasjonens slutt

Betatestingen avslørte noen småting som måtte fikses, deriblandt noen NullPointerExceptions. Feilen(e) ble funnet, og rettet på. Noen av tingene som ble påpekt under betatestingen, ble også rettet. Småtweaks i koden er også foretatt. Testerne har så gått igjennom tingene som «feilet» i betatesten, og sjekket at dette er fikset (mao; de har prøvd å replikere feilen(e)). Vi er så og si ferdig med en stor, solid og fungerende applikasjon, og vi har klart å holde oss innenfor tidsrammene vi satt oss fore.

Under finner man en oversikt over planlagt vs. reelt tidsforbruk, samt totaler for hele prosjektet.

Hva?	Planlagt	Reell
Legge ut mer informasjon enkelte plasser, samt endre informasjon noen plasser, ifbm. tilbakemelding vi fikk fra betatestingen	2	1,75
BUG: Sessio-timeout gir Nullpointer	1	0,75
BUG: EpostDAOImpl/BestillingDAOImpl gir Nullpointer	0,5	0,75
MANGEL/FIX: Rapport tildelinger tilbake i tid skal også kunne ta med betalte	1	0,50
BUG/FIX: Blank text-input under innstillinger. CSS-relatert.	0,5	0,25
FIX/BUG: Ved oppstart; kjøre trekning på evt. ting som mangler trekning (f.eks. dersom server har vært nede i noen dager).	5	5,5
BUG(?): Blank side ved bekreftelse av bestilling. Refresh, og det funker.	0,5	0
FIX: «Bestilling mottatt» - lenke til kalender hvor reservasjonen fant sted	1,5	0,25
MANGEL/FIX: Admin skal få epost ved reservasjoner.	2	0,5
BUG: Fikk ikke tildelt hytte etter loddtrekning	5	3,0
FIX/MANGEL: Egen JSP for exception-feil	5	0
(FIX: Evt. skrive om DatabaseDAO)	0	0
Oppdatere Brukstilfeller & Visjon	4	3,75
Oppdatere domenemodell & begrepsliste	2	1,0
Oppdatere Testlogg iht siste tester	4	6,75
Oppdatere systemarkitektur (!)	10	3,25
Oppdatere Databasemodell	5	2,75
Oppdatere Designmodell	5	3,75
TESTING: Teste tingene som har blitt fikset etter betatestingen	5	10,75
Logg for iterasjon 6	1	1,25
(Sluttrapport m/ refleksjonsnotat)	10	15,0
Møter	20	15,75

TOTALT	90	77,25

Hele prosjektet.

Elaboration	78	71,35
Inception	170	179
Construction	230	221
Transition	90	77,25

TOTALT	568	548,6

Systemarkitektur

Arkitekturmessige mål og begrensninger

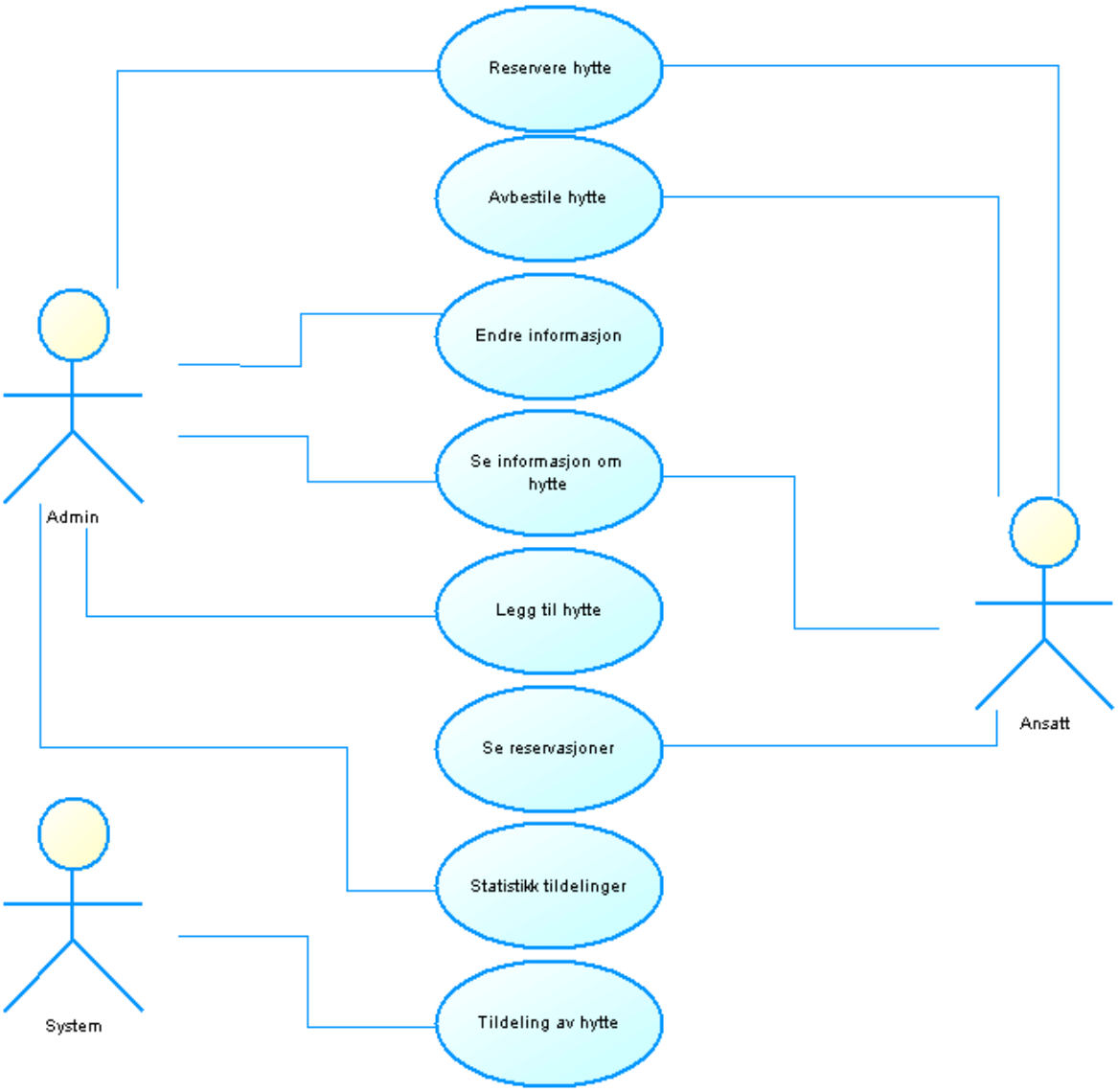
Me har eit system, som kun er basert på web, og har valgt å bruke Java servlets, Jsp, css og PostgreSql som verktøy. Me treng også ein Tomcat server for å kunne kjøre systemet. Me har fått konto på eple som me skal kjøre systemet på, og brukere trenger ei datamaskin på intranettet for å kunne bruke det.

Målet med applikasjonen er at det skal bli felles produkt for ansatte samt dei som skal administere hyttebestilling ved Høgskolen I Bergen. Det skal være ein web basert applikasjon, men den skal kun være tingjengelig for dei som er på intranettet til Høgkolen. Dette er fordi at ikkje kvem som helst skal få tilgang til å bestille hytter.

Me rekner med at ein server vi være nok. Antall ansatte på HiB er ganske stor, men sjølv I dei store feriane, når kanskje 5-600 personer vil prøve å leige hytta samtidig, så vil det gå greit. Tideling av hytter vil skje I forkant av selve ferien, så me vil nok ikkje få nokon store problemar der.

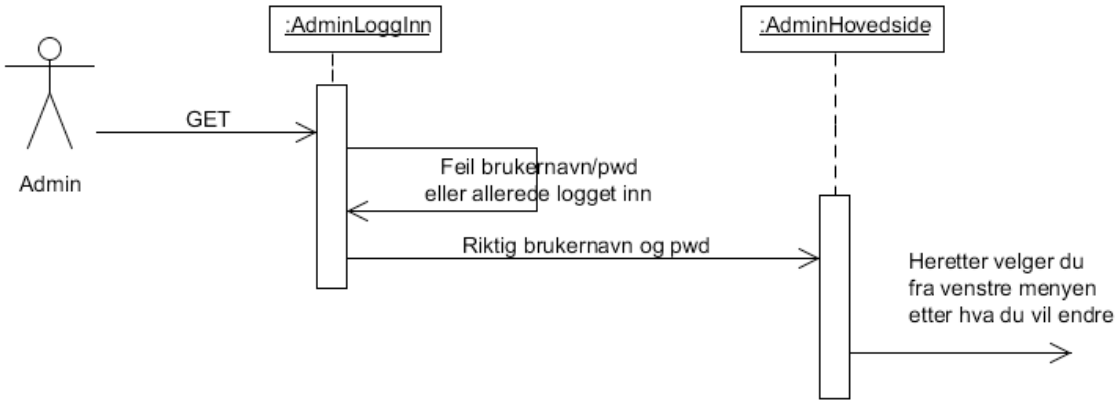
Brukstilfelle-view

Brukstilfellediagram:



Sekvensdiagram:

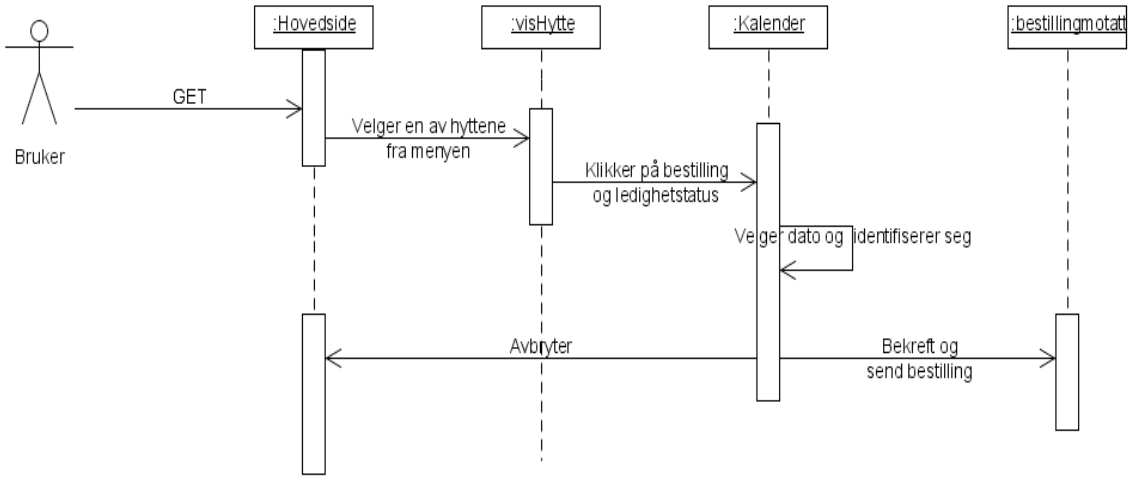
Vise hytte for admin, brukstilfelle 4.1:



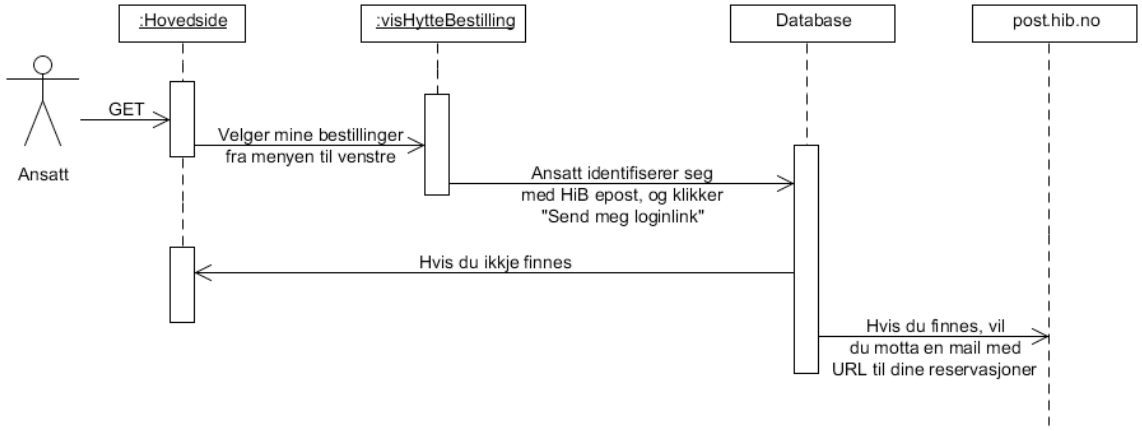
4.3

En ny hytte blir lagt til I database, og vil automatisk bli lagt til I menyen og gjort tilgjengelig ved hjelp av servlet. Informasjon om hytten samt andre ting, kan administreres via admin sidene, via høyremenyen.

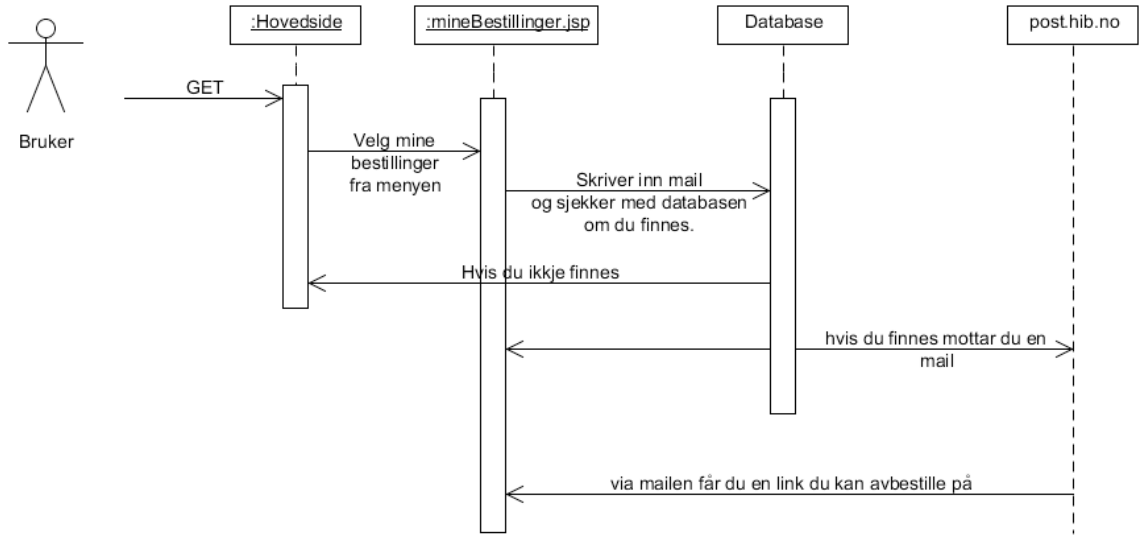
5.1 Reservere hytte:



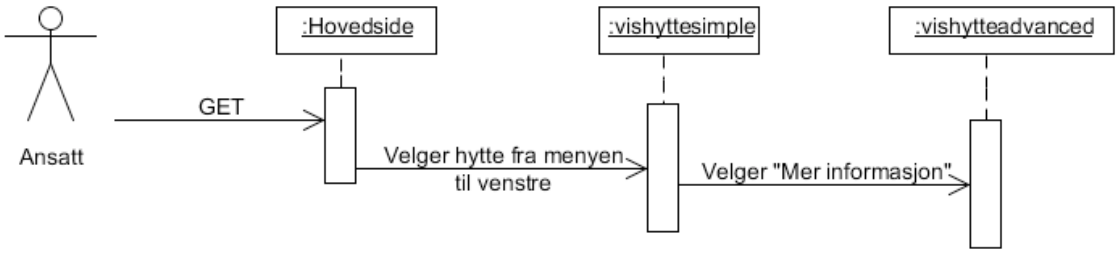
5.2:


:Hovedside
:visHytteBestilling
Database
post.hib.no
Ansatt
GET
Velger mine bestillinger
fra menyen til venstre
Ansatt identifiserer seg
med HiB epost, og klikker
"Send meg loginlink"
Hvis du ikkje finnes
Hvis du finnes, vil
du motta en mail med
URL til dine reservasjoner

5.3:


:Hovedside
:mineBestillinger.jsp
Database
post.hib.no
Bruker
GET
Velg mine
bestillinger
fra menyen
Skriver inn mail
og sjekker med databasen
om du finnes.
Hvis du ikkje finnes
hvis du finnes mottar du en
mail
via mailen får du en link du kan avbestille på

5.4:


:Hovedside
:vishyttesimple
:vishytteadvanced
Ansatt
GET
Velger hytte fra menyen
til venstre
Velger "Mer informasjon"

Logisk view

Beskrivelse av mekanismer:

Kalenderen:

Kalenderen er laget heilt fra scratch. Hver gang du trykker deg inn på ei hytte og kommer til kalenderen, vil det bli opprettet et nytt kalender objekt som du benytter. Det er laget fargekoder for de ulike status tilfellene av hyttas tilgjengelighet. For å bestille hytta må du velge datoene fra kalenderen. Hvis du velger f.eks en lørdag, vil fredag og søndag automatisk også bli markert, men hvis du klikker igjen, vil bare lørdag bli valgt. Det samme vil skje med ukedagene mandag til torsdag. Alle helligdager og ferier er utreknet fra java klassene våre og vil være markert med egne fargekoder. Disse dagene er spesielle og vil danne egne perioder utenom mandag til torsdag og fredag til søndag.

Bestilling av hytter:

Bestilligen av hytter foregår hovedsaklig gjennom kalenderen på siden. Alle ansatte ved Høgskulen I Bergen har mulighet til å bestille/vise interesse for hytten(e) for hvilken som helst periode. Du vil da få en bekreftelse og videre svar via hib mailen din. I første omgang når du bestiller hytten, så er det ikkje sikkert at du får den med en gang. Det er kanskje mange som ønsker hytten samtidig som deg, så mangen ansatte vil bestille den til samme tid. Alle desse vil bli lagt inn I ei liste, og rangert etter når de hadde hytten sist, og om de er “svartelistet” osv. En liten stund før datoen tar sted, vil hytten bli tildelt en av de som har vist interesse.

Akseptering av tildelt hytte:

Hvis du blir tildelt en hytte av systemet, vil du få en mail om dette, som du må bekrefte. Når du har bekreftet denne, er hytten satt av til deg og ingen andre. Men hvis du velger å takke nei til hytten hvis noe har kommet I veien, så har du også mulighet til dette. Da vil hytten bli delt ut til neste ansatt på prioriteringslista, og du vil bli plassert på bunn på lista for hytta på den datoen, og kun den I dette tilfelle.

Arkitekturmessig viktig moduler og pakker:

Hytteweb:

Pakken hytteweb har tre underpakker, model, dao og controller.Desse henger alle sammen, men har ulike vitale roller.

Model:

I denne pakken ligger alle “grunnklassene”. Java klassene som ligger her inneholder henholdsvis variabler og get/set metoder.Klassene som ligge I denne pakken er, Bestilling, Ferie, Hytte, Kalender, KalenderElement, sperre og Periode. Den viktigste og mest komplekse klassen her er Kalender klassen.Den oppretter ein kalender for ei hytte, først og fremst med helligdager, som den rekner ut sjølv. Den legger til helligdager som er samme dato hvert år som foreksempel 1.mai og rekner ut bla. Påskedager som har forskjellige datoer kvart år. Periode klassen setter sammen spesielle datoer og merker av ein periode I kalenderen, desse kan være påskedager, eller sommerferie uker. Kvar dag I

kalenderen har/er eit KalenderElement. Så alle bestillinger som er gjort, tildelt og ikkje tildelt, blir gjort om til kalender element, og lagt til I kalenderen. Ferie klassen lager en tekst ved siden av kalenderen som viser ferie til-fra datoer.

Dao:

I pakken dao har me mange dao klasser(Data access object). Alle desse klassane er interfaces til DAOimpl filene som ligger I underpakka context. I context er alle filene markert med impl(Implementation) på slutten. Oppgavene til desse klassene er å prossesere databasen.

Det ligger noen viktige klasser her. DatabaseDaoImpl med metoder for å koble til databasen, BestillingDaoImpl som henter alle bestilliger fra databasen og fleire liknande.

Controller:

Controller pakken inneholder alle Java servletsene vi bruker. Desse klassene arver fra daoImpl klassene. Servlets klassene jobber sammen med jsp(Java script filene) for å produsere det visuelle, samt ta mot ting som brukeren skriver inn. Det første som skjer når du kommer til siden, er at du gjennom Hovedside servleten blir “sendt til” hovedside jspen, som igjen bruker blant annet venstremeny jsp og hoyreMeny jsp som til slutt blir hovedsiden.

HytteWebUtility:

I denne pakken som ligger rett under src, ligger det noen hjelpeklasser, til blant annet sjekk av datoer, og diverse postfelt I programmet.

WEB-INF:

Web inf pakken inneholder to undermapper; admin og lib. Admin inneholder alle jspene som admin bruker til begrensede instillinger og valg.

Web inf innholder også alle de andre jspene som den vanlige brukeren også benytter, men på et annet nivå enn admin. I tillegg ligger det diverse css filer som lager designet og oppsettet til sidene.

Arkitekturmessige valg:

Servlets:

Vi skal bruke en modell som med flere servlets gjør en gitt oppgave og forwarder brukeren til den rette jspen.

Jsp:

Java script pagene skal følge xhtml transitional standaren. Dei skal produsere det visuelle for brukeren.

Klasser, DAO:

Vi har valgt å bruke DAO arkitekturen. JDBC skal me bruke for å få kontakt med databasen vår.

Tekniske valg:

Database: MySQL
Utviklingsmijlø: Eclipse IDE for Java EE Developers
Server: Tomcat v5.5
SVN: Subversion 1.4.5

Databasemodell

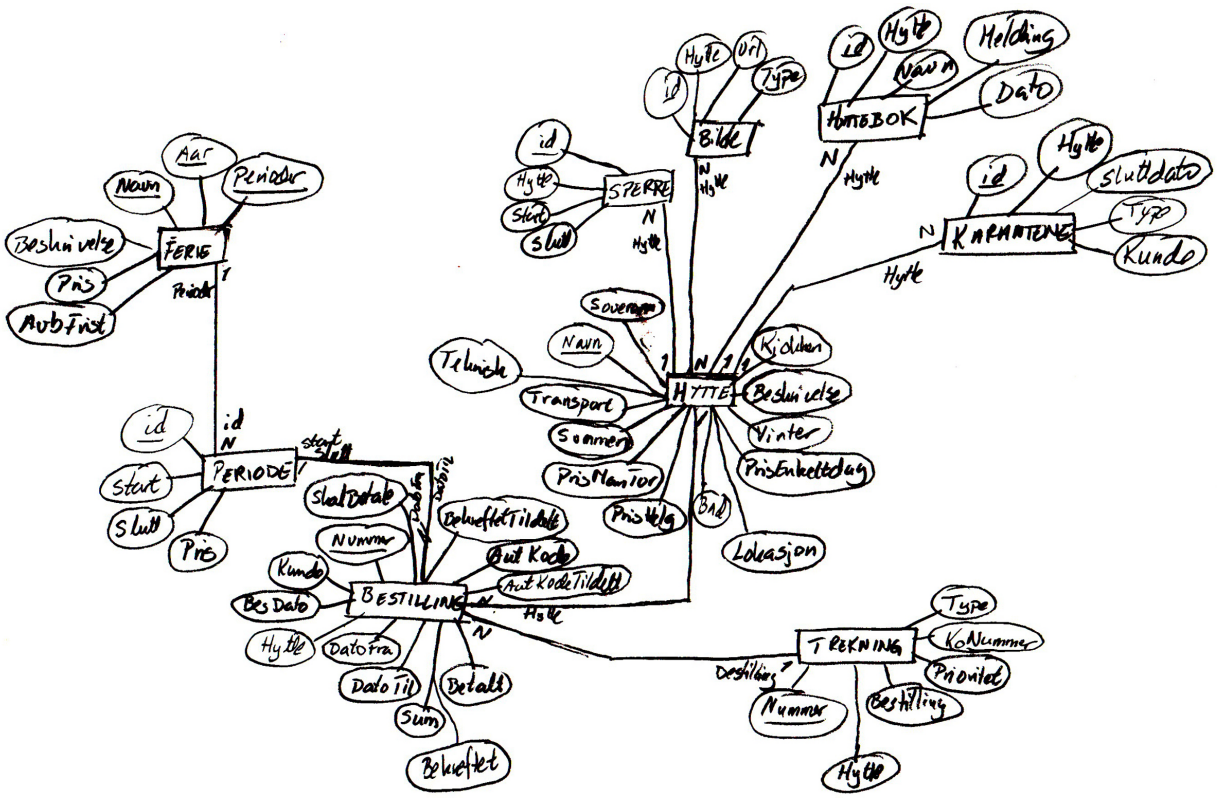
Gi en visuell oversikt over relasjonene mellom de ulike tabellene i databasen. Med dette får man en oversikt over hvordan primærnøkler og fremmednøkler henger sammen. Dokumentet skal også gi en beskrivelse av hvilke entiteter og navn på felt i tabellene.

Dokumentet skal beskrive evt. relasjonsklasser og hvilken Cardinality ration disse har.

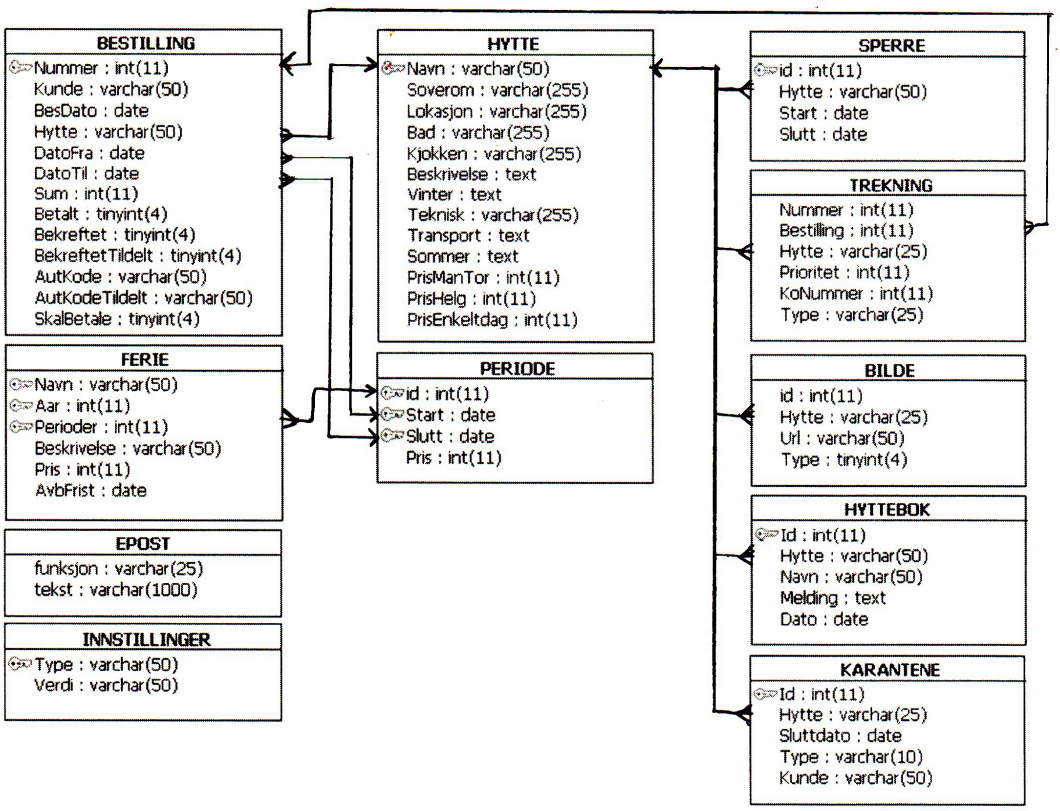
Bakgrunn og betraktninger

Databasemodellen er designet ut fra designmodellen, der behovet for å lagre bestillinger, hytter, ferier, perioder, trekningslister, karantener, hyttebok, url til bilder og datoer for når ei hytte er sperret for bestillinger.

Konseptuell modell



Relasjonsmodell

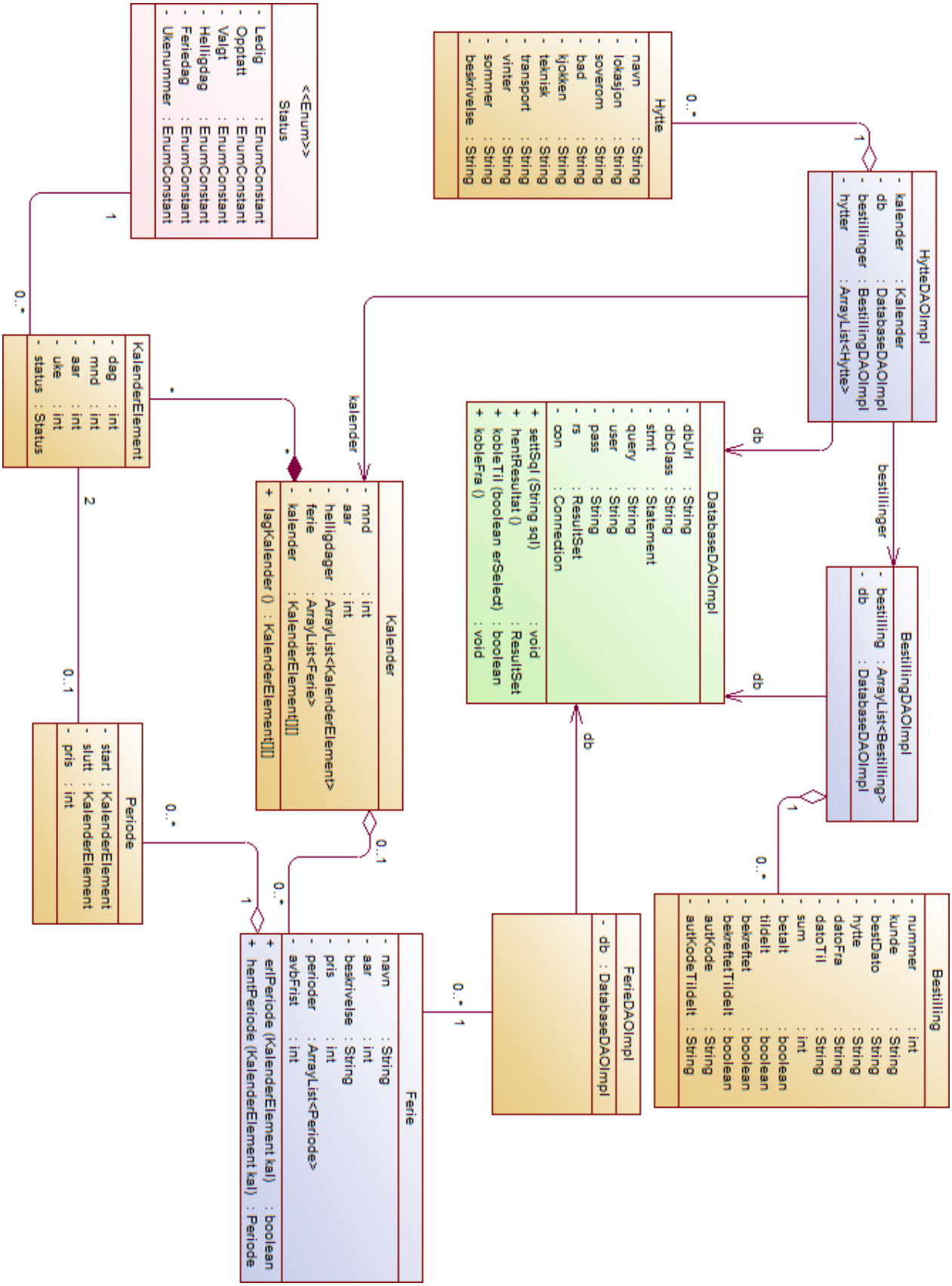


Designmodell

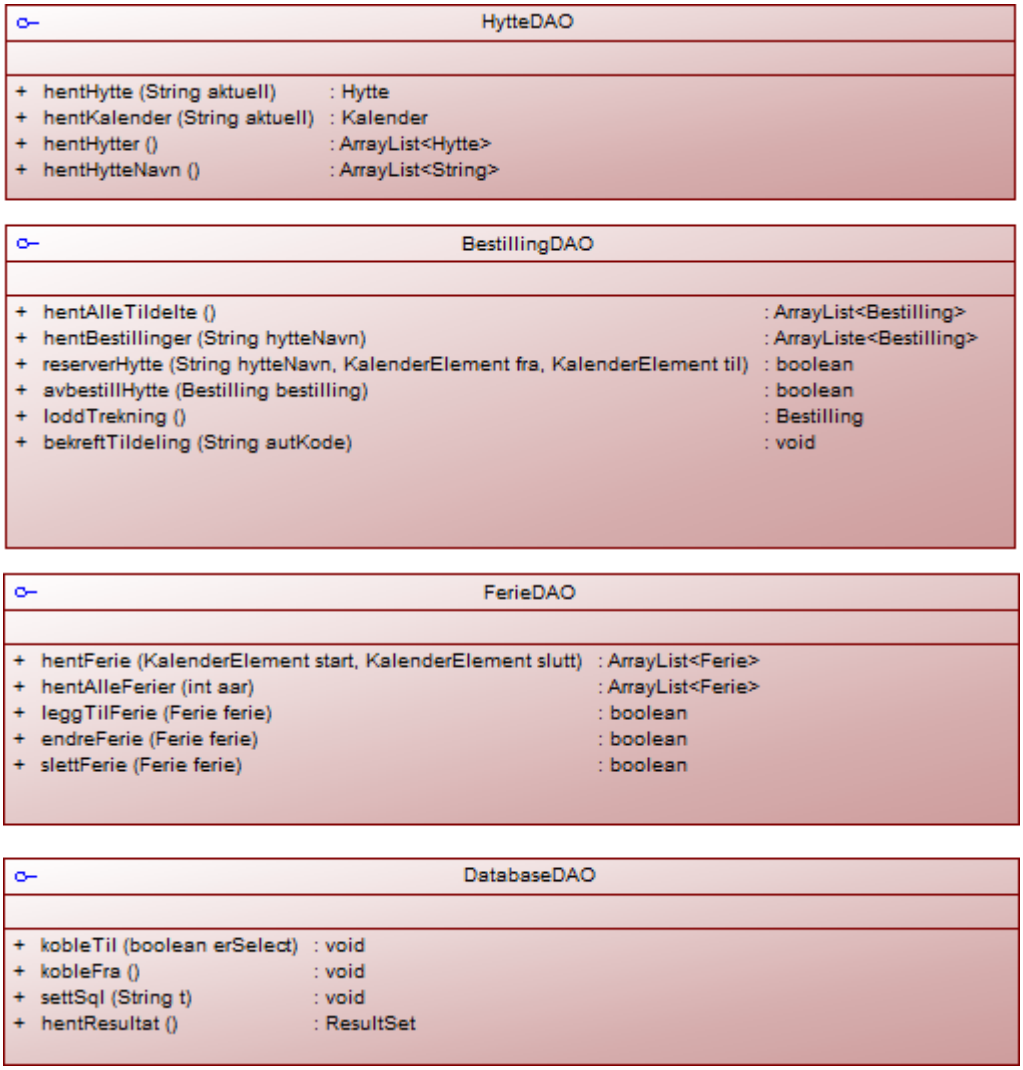
Brukstilfelle “Vise hytte med kalender for registrering”

Applikasjonen skal vise en hytte der det skal foreligge ulik informasjon om selve hytten. Den skal også kunne generere en kalender som har støtte for reservasjon/ registrering.

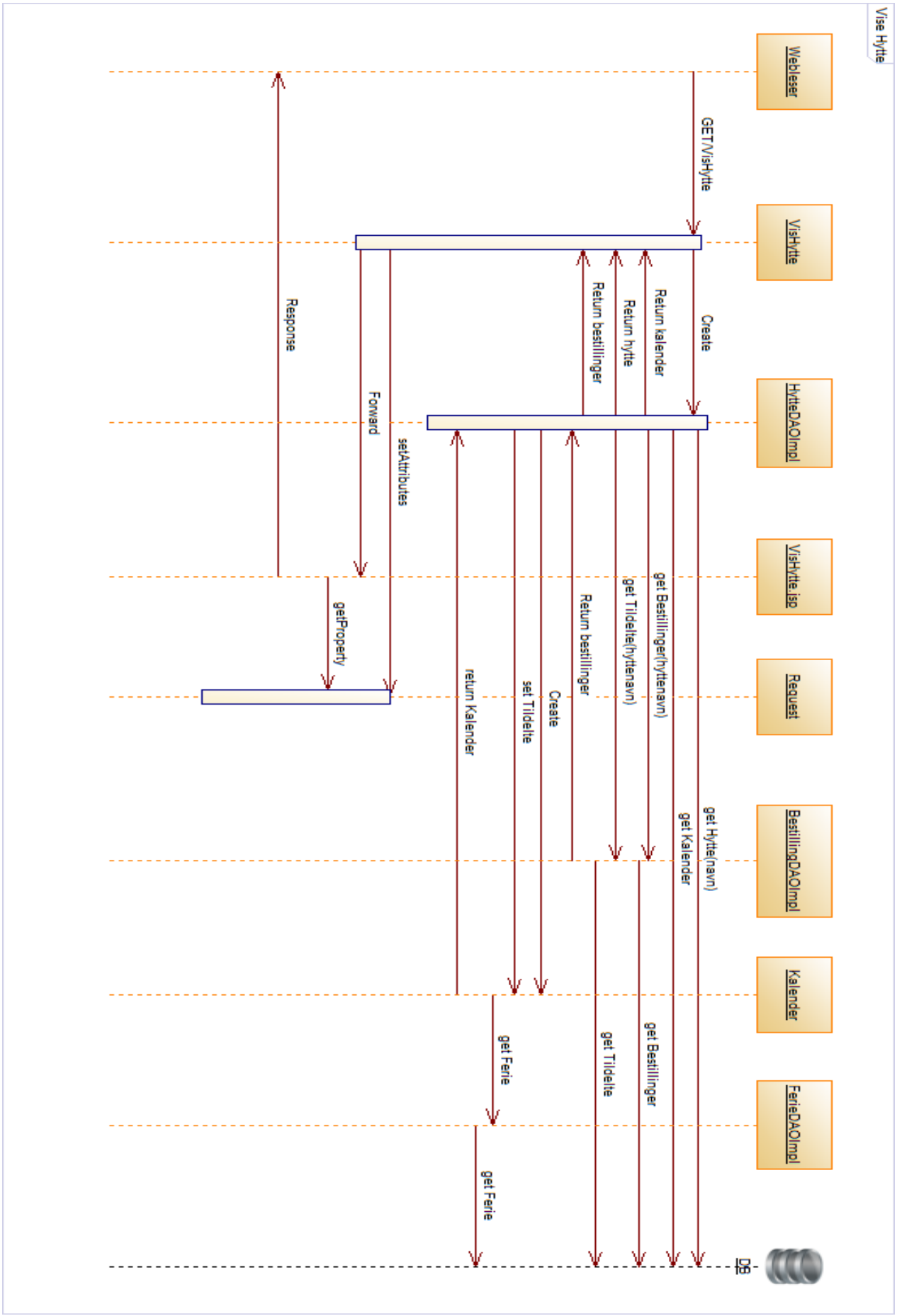
Klassediagram



Interface


HytteDAO
+ hentHytte (String aktuell) : Hytte
+ hentKalender (String aktuell) : Kalender
+ hentHytter () : ArrayList<Hytte>
+ hentHytteNavn () : ArrayList<String>
BestillingDAO
+ hentAlleTildelte () : ArrayList<Bestilling>
+ hentBestillinger (String hytteNavn) : ArrayListe<Bestilling>
+ reserverHytte (String hytteNavn, KalenderElement fra, KalenderElement til) : boolean
+ avbestillHytte (Bestilling bestilling) : boolean
+ loddTrekning () : Bestilling
+ bekreftTildeling (String autKode) : void
FerieDAO
+ hentFerie (KalenderElement start, KalenderElement slutt) : ArrayList<Ferie>
+ hentAlleFerier (int aar) : ArrayList<Ferie>
+ leggTilFerie (Ferie ferie) : boolean
+ endreFerie (Ferie ferie) : boolean
+ slettFerie (Ferie ferie) : boolean
DatabaseDAO
+ kobleTil (boolean erSelect) : void
+ kobleFra () : void
+ settSql (String t) : void
+ hentResultat () : ResultSet

Sekvensdiagram



Kildekode

MODEL

Bestilling.java

Ferie.java

Kalender.java

KalenderElement.java

Periode.java

Sperre.java

DAO

BestillingDAOImpl.java

DatabaseDAOImpl.java

EpostDAOImpl.java

FerieDAOImpl.java

CONTROLLER

Admin.java

Bestilling.java

Trekning.java

JSP

vishyttebestilling.jsp

adminInnstillinger.jsp

MODEL

Bestilling.java

```
package hytteWeb.model;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Bestilling {
         private int nummer;
         private String kunde;
         private String bestDato;
         private String hytte;
         private String datoFra;
         private String datoTil;
         private String ferie;
         private int sum;
         private boolean betalt;
         private boolean tildelt;
         private boolean bekreftet;
         private boolean bekreftetTildelt;
         private String autKode;
         private String autKodeTildelt;

         public int getNummer() {
            return nummer;
         }
         public void setNummer(int newNummer) {
            nummer = newNummer;
         }
         public String getKunde() {
            return kunde;
         }
         public void setKunde(String newKunde) {
            kunde = newKunde;
         }
         public void setFerie(String newFerie) {
                  ferie = newFerie;
         }
         public String getBestDato() {
            return bestDato;
         }
         public void setBestDato(String newBestDato) {
            bestDato = newBestDato;
         }
         public String getHytte() {
            return hytte;
         }
         public void setHytte(String newHytte) {
            hytte = newHytte;
         }
         public String getDatoFra() {
            return datoFra;
         }
         public void setDatoFra(String newDatoFra) {
            datoFra = newDatoFra;
         }
         public String getDatoTil() {
            return datoTil;
         }
         public void setDatoTil(String newDatoTil) {
            datoTil = newDatoTil;
```

```
}
public int getSum() {
   return sum;
}
public void setSum(int newSum) {
   sum = newSum;
}
public boolean getBetalt() {
   return betalt;
}
public void setBetalt(boolean newBetalt) {
   betalt = newBetalt;
}
public boolean getTildelt() {
   return tildelt;
}
public String getFerie() {
         return ferie;
}
public void setTildelt(boolean newTildelt) {
   tildelt = newTildelt;
}
public boolean getBekreftet() {
   return bekreftet;
}
public void setBekreftet(boolean newBekreftet) {
   bekreftet = newBekreftet;
}
public boolean getBekreftetTildelt() {
   return bekreftetTildelt;
}
public void setBekreftetTildelt(boolean newBekreftetTildelt) {
   bekreftetTildelt = newBekreftetTildelt;
}
public String getAutKode() {
   return autKode;
}
public void setAutKode(String newAutKode) {
   autKode = newAutKode;
}
public String getAutKodeTildelt() {
   return autKodeTildelt;
}
public void setAutKodeTildelt(String newAutKodeTildelt) {
   autKodeTildelt = newAutKodeTildelt;
}
public Date getDateFra(){
      Date tmp = new Date();
      SimpleDateFormat fTmp = new SimpleDateFormat("yyyy-MM-dd");
      try {
         tmp = fTmp.parse(datoFra);
   } catch (ParseException e) {
         e.printStackTrace();
   }
      return tmp;
}

public Date getDateTil(){
      Date tmp = new Date();
      SimpleDateFormat fTmp = new SimpleDateFormat("yyyy-MM-dd");
      try {
         tmp = fTmp.parse(datoTil);
   } catch (ParseException e) {
         e.printStackTrace();
   }
      return tmp;
```

```
        }
}
```

Ferie.java

```
package hytteWeb.model;

import java.util.ArrayList;

public class Ferie {
	String navn;
	int aar;
	ArrayList<Periode> perioder;
	String beskrivelse;
	int pris;
	KalenderElement avbFrist;

	public Ferie(){

	}
	public Ferie(String navn, int aar, String beskrivelse, int pris,
KalenderElement avbFrist) {
		this.navn = navn;
		this.aar = aar;
		this.beskrivelse = beskrivelse;
		this.pris = pris;
		this.avbFrist = avbFrist;
		perioder = new ArrayList<Periode>();


		// Bare for testing
//		KalenderElement a = new KalenderElement(12, 4, 2010, 15);
//		KalenderElement b = new KalenderElement(18, 4, 2010, 15);
//		Periode c = new Periode(a, b, 1200);
//		KalenderElement d = new KalenderElement(26, 4, 2010, 17);
//		KalenderElement e = new KalenderElement(2, 5, 2010, 17);
//		Periode f = new Periode(d, e, 1200);
//		perioder.add(c);
//		perioder.add(f);
	}

/
**********************************************************************************
***************
 * Get og set

**********************************************************************************
***************/
	public String getNavn() {
		return navn;
	}

	public int getAar() {
		return aar;
	}

	public ArrayList<Periode> getPerioder() {
		return perioder;
	}

	public String getBeskrivelse() {
		return beskrivelse;
	}
```

```
	public int getPris() {
		return pris;
	}

	public KalenderElement getAvbFrist() {
		return avbFrist;
	}

	public void setNavn(String navn) {
		this.navn = navn;
	}

	public void setAar(int aar) {
		this.aar = aar;
	}

	public void setPerioder(ArrayList<Periode> perioder) {
		this.perioder = perioder;
	}

	public void setBeskrivelse(String beskrivelse) {
		this.beskrivelse = beskrivelse;
	}

	public void setPris(int pris) {
		this.pris = pris;
	}

	public void setAvbFrist(KalenderElement avbFrist) {
		this.avbFrist = avbFrist;
	}


/
**********************************************************************************
****************
 * Metode som legger til en periode
 * @param Periode - Periode som skal legges til

**********************************************************************************
****************/
	public void leggTilPeriode(Periode p) {
		perioder.add(p);
	}

/
**********************************************************************************
****************
 * Metode som returnerer perioden et KalenderElement er i
 * @param k - KalenderElement
 * @return Periode Returnerer Periode dersom det finnes i en periode - ellers
null

**********************************************************************************
****************/
	public Periode finnPeriode(KalenderElement k) {
		boolean funnet = false;
		Periode aktuell = null;
		for(int i=0; i<perioder.size() && !funnet; i++) {
			if(perioder.get(i).finnesIPerioden(k)) {
				funnet = true;
				aktuell = perioder.get(i);
			}
		}
		return aktuell;
	}
```

```
/
**********************************************************************************
****************
 * Metode som returnerer om et KalenderElement er i en periode
 * @param k - KalenderElement
 * @return boolean Returnerer sann dersom den finnes

**********************************************************************************
****************/
      public boolean erIPeriode(KalenderElement k) {
            boolean funnet = false;

            for(int i=0; i<perioder.size() && !funnet; i++) {
                  if(perioder.get(i).finnesIPerioden(k)) {
                        funnet = true;
                  }
            }
            return funnet;
      }

/
**********************************************************************************
****************
 * Metode som returnerer antall perioder i en ferie
 * @return int Returnerer antall perioder

**********************************************************************************
****************/
      public int antallPerioder() {
            return perioder.size();
      }

/
**********************************************************************************
****************
 * Metode som returnerer antall perioder i en ferie
 * @return int Returnerer antall perioder

**********************************************************************************
****************/
      public boolean equals(Ferie f) {
            if(navn.equals(f.getNavn()) && aar == f.getAar())
                  return true;
            else
                  return false;
      }

/
**********************************************************************************
****************
 * Metode som returnerer antall perioder i en ferie
 * @return int Returnerer antall perioder

**********************************************************************************
****************/
      public String toString() {
            String tmp = navn+"<br />";

            for(int i=0; i<perioder.size(); i++) {
                  KalenderElement start = perioder.get(i).getStart();
                  KalenderElement slutt = perioder.get(i).getSlutt();
                  tmp = tmp+" "+start.getDag()+"."+start.getMnd()+" -
"+slutt.getDag()+"."+slutt.getMnd()+"<br />";
            }
            tmp = tmp+"<br />";
```

```
            return tmp;
      }
}
```

Kalender.java

```
package hytteWeb.model;

import hytteWeb.dao.context.BestillingDAOImpl;
import hytteWeb.dao.context.FerieDAOImpl;
import hytteWeb.dao.context.HytteDAOImpl;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;

public class Kalender {
      int mnd;
      int aar;
      ArrayList<KalenderElement> helligdager;
      ArrayList<KalenderElement> valgte;
      ArrayList<Ferie> ferie;
      ArrayList<Bestilling> tildelte;
      ArrayList<Bestilling> ikkeTildelte;
      KalenderElement[][] kalender;
      String hytteNavn;

/
**********************************************************************************
****************
 * Konstruktør
 * @param int mnd - Måned
 * @param int aar - År

**********************************************************************************
*****************/
      public Kalender(int mnd, int aar) {
            this.mnd = mnd;
            this.aar = aar;
            helligdager = new ArrayList<KalenderElement>();
            ferie = new ArrayList<Ferie>();
            valgte = new ArrayList<KalenderElement>();
            tildelte = new ArrayList<Bestilling>();
            ikkeTildelte = new ArrayList<Bestilling>();

            settInnHelligdager();

            FerieDAOImpl f = new FerieDAOImpl();
            ferie = f.hentAlleFerier(aar);
      }

/
**********************************************************************************
****************
 * Setter for mnd og aar
 * @param int mnd - Måned
 * @param int aar - År

**********************************************************************************
*****************/
      public void settManedAr(int mnd, int aar) {
            this.mnd = mnd;
            this.aar = aar;
```

```
            settInnHelligdager();
            FerieDAOImpl f = new FerieDAOImpl();
            ferie = f.hentAlleFerier(aar);
      }

/
**********************************************************************************
****************
 * Getter for ferie
 * @param ArrayList<Ferie> ferie - Alle ferier i et år

**********************************************************************************
****************/
      public ArrayList<Ferie> getFerie() {
            return ferie;
      }

/
**********************************************************************************
****************
 * Setter for mnd og aar
 * @param int mnd - Måned
 * @param int aar - År

**********************************************************************************
****************/
      public void settHyttenavn(String hytteNavn) {
            this.hytteNavn = hytteNavn;
            lagKalender();
      }


/
**********************************************************************************
****************
 * Henter kalenderen som en 2-dimensjonell tabell
 * @return KalenderElement[][] kalender - 2-dimensjonell tabell med alle
kalenderelement

**********************************************************************************
****************/
      public KalenderElement[][] hentKalender() {
            return kalender;
      }

/
**********************************************************************************
****************
 * Setter opp kalenderen med ulike kalender-element.

**********************************************************************************
****************/
      private void lagKalender() {
            GregorianCalendar forsteElement = new GregorianCalendar
(aar,mnd-1,1);
            forsteElement.add(Calendar.DAY_OF_YEAR,-(finnAntallLedigeForan() +
7));
            kalender = new KalenderElement[7][8];

            ArrayList<KalenderElement> bestillinger = bestillingTilKalender();
            ArrayList<KalenderElement> bestillingerIkkeTildelt =
bestillingTilKalenderIkkeTildelt();
            ArrayList<KalenderElement> sperrete = sperrete();

            for (int i=0; i<kalender.length; i++) {
```

```
                  kalender[i][0] = new KalenderElement(forsteElement.get
(Calendar.DAY_OF_MONTH),
                              forsteElement.get(Calendar.MONTH),
forsteElement.get(Calendar.YEAR),
                              forsteElement.get(Calendar.WEEK_OF_YEAR));
                  kalender[i][0].setStatus(Status.Ukenummer);

                  for(int j=1; j<kalender[i].length; j++) {
                        KalenderElement ke = new KalenderElement
(forsteElement.get(Calendar.DAY_OF_MONTH),
                                    (forsteElement.get(Calendar.MONTH)+1),
forsteElement.get(Calendar.YEAR),
                                    forsteElement.get(Calendar.WEEK_OF_YEAR));
                        ke.setStatus(Status.Ledig);

                        // Her må all sjekk på status foregå
                        if(erHelligdag(ke)) {
                              ke.setStatus(Status.Helligdag);
                        }

                        boolean erFerieDag = false;
                        if(ferie.size() > 0) {
                              for(int k=0; k<ferie.size() && !erFerieDag; k++) {
                                    if(ferie.get(k).erIPeriode(ke))
                                          erFerieDag = true;
                              }
                        }

                        if(ke.getMnd() < mnd || ke.getMnd() > mnd) {
                              ke.setStatus(Status.AnnenMnd);
                        }

                        if(erFerieDag)
                              ke.setStatus(Status.Feriedag);

                        // Ikke tildelte
                        boolean funnetBestillingBest = false;
                        for(int n=0; n<bestillingerIkkeTildelt.size() && !
funnetBestillingBest; n++) {
                              if(bestillingerIkkeTildelt.get(n).equals(ke)) {
                                    funnetBestillingBest = true;
                              }
                        }

                        // Tildelte
                        boolean funnetBestilling = false;
                        for(int n=0; n<bestillinger.size() && !funnetBestilling;
n++) {
                              if(bestillinger.get(n).equals(ke)) {
                                    funnetBestilling = true;
                              }
                        }

                        // Ikke tildelte
                        if(funnetBestillingBest) {
                              ke.setStatus(Status.Bestillt);
                        }

                        // Tildelte
                        if(funnetBestilling) {
                              ke.setStatus(Status.Trukket);
                        }

                        for(int h=0; h<valgte.size(); h++) {
                              if(valgte.get(h).equals(ke))
                                    ke.setStatus(Status.Valgt);
```

```
                        }

                        // Sperrete
                        boolean funnetSperret = false;
                        for(int p=0; p<sperrete.size() && !funnetSperret; p++) {
                              if(sperrete.get(p).equals(ke)) {
                                    funnetSperret = true;
                                    ke.setStatus(Status.Opptatt);
                              }
                        }

                        // Her må all sjekk på status foregå slutt
                        kalender[i][j] = ke;
                        forsteElement.add(Calendar.DAY_OF_MONTH,1);
                  }
            }
      }

/
**********************************************************************************
*****************
 * Metode som finner alle ledige plasser foran ei uke
 * @return int antall dager før første kalenderelement i aktuell måned.

**********************************************************************************
*****************/
      private int finnAntallLedigeForan() {
            SimpleDateFormat formatter = new SimpleDateFormat("E");
            GregorianCalendar c = new GregorianCalendar(aar,mnd-1,1);
            if(formatter.format(c.getTime()).equals("ma")) return 0;
            if(formatter.format(c.getTime()).equals("ti")) return 1;
            if(formatter.format(c.getTime()).equals("on")) return 2;
            if(formatter.format(c.getTime()).equals("to")) return 3;
            if(formatter.format(c.getTime()).equals("fr")) return 4;
            if(formatter.format(c.getTime()).equals("lø")) return 5;
            if(formatter.format(c.getTime()).equals("sø")) return 6;
            return 0;
      }

/
**********************************************************************************
*****************
 * Statisk metode som beregner 1.påskedag etter gitt år
 * @param aar - int årstallet for beregningen
 * @return Kalender - GregorianCalender 1.påskedag - Husk at GregorianCalendeer
starter index 0
 * for måned - Januar=0 og ikke 1

**********************************************************************************
*****************/
      public static GregorianCalendar beregnPaaske(int aar) {
            int Y=aar;
            int a = Y % 19;
            int b = Y / 100;
            int c = Y % 100;
            int d = b / 4;
            int e = b % 4;
            int f = (b + 8) / 25;
            int g = (b - f + 1) / 3;
            int h = (19 * a + b - d - g + 15) % 30;
            int i = c / 4;
            int k = c % 4;
            int L = (32 + 2 * e + 2 * i - h - k) % 7;
            int m = (a + 11 * h + 22 * L) / 451;
            int month = (h + L - 7 * m + 114) / 31;
            int day = ((h + L - 7 * m + 114) % 31) + 1;
```

```
            return new GregorianCalendar(aar,month-1,day);
      }

/
**********************************************************************************
****************
 * Statisk metode som beregner helligdager basert på påske - benytter seg av
beregnPaaske(int aar)
 * Palmesøndag -7
 * Skærtorsdag -3
 * Langfredag -2
 * Påskemandag 1
 * Kristi himmelfart 39
 * Pinsedag 49
 * Andre pinsedag 50
 * @param aar - int årstallet for beregningen
 * @param nr - int antall dager som skal flyttes
 * @return Kalender - GregorianCalender 1.påskedag - Husk at GregorianCalendeer
starter index 0
 * for måned - Januar=0 og ikke 1

**********************************************************************************
*****************/
      public static GregorianCalendar beregnPaaskedager(int aar, int nr)     {
            //Hent påske
            GregorianCalendar paaskeAndre=beregnPaaske(aar);
            //Legg til eller trekke fra dager.
            paaskeAndre.add(Calendar.DAY_OF_MONTH,nr);
            return paaskeAndre;
      }

/
**********************************************************************************
****************
 * Metode som setter inn alle helligdager

**********************************************************************************
*****************/
      private void settInnHelligdager() {
            helligdager.add(new KalenderElement(1, 1, 0, 0)); // Nyttårsdag

            // Påskerelaterte helligdager
            Calendar paaske = beregnPaaske(aar);
            paaske.add(Calendar.MONTH, +1);
            helligdager.add(new KalenderElement(paaske.get
(Calendar.DAY_OF_MONTH), paaske.get(Calendar.MONTH), 0, 0)); // 1. Påskedag

            paaske = beregnPaaskedager(aar, 1);
            paaske.add(Calendar.MONTH, +1);
            helligdager.add(new KalenderElement(paaske.get
(Calendar.DAY_OF_MONTH), paaske.get(Calendar.MONTH), 0, 0)); // 2. Påskedag

            paaske = beregnPaaskedager(aar, -2);
            paaske.add(Calendar.MONTH, +1);
            helligdager.add(new KalenderElement(paaske.get
(Calendar.DAY_OF_MONTH), paaske.get(Calendar.MONTH), 0, 0)); // Langfredag

            paaske = beregnPaaskedager(aar, -3);
            paaske.add(Calendar.MONTH, +1);
            helligdager.add(new KalenderElement(paaske.get
(Calendar.DAY_OF_MONTH), paaske.get(Calendar.MONTH), 0, 0)); // Skjærtorsdag

            paaske = beregnPaaskedager(aar, -7);
            paaske.add(Calendar.MONTH, +1);
```

```
            helligdager.add(new KalenderElement(paaske.get
(Calendar.DAY_OF_MONTH), paaske.get(Calendar.MONTH), 0, 0)); // Palmesøndag

            paaske = beregnPaaskedager(aar, 39);
            paaske.add(Calendar.MONTH, +1);
            helligdager.add(new KalenderElement(paaske.get
(Calendar.DAY_OF_MONTH), paaske.get(Calendar.MONTH), 0, 0)); // Kristi
himmelfartsdag

            paaske = beregnPaaskedager(aar, 49);
            paaske.add(Calendar.MONTH, +1);
            helligdager.add(new KalenderElement(paaske.get
(Calendar.DAY_OF_MONTH), paaske.get(Calendar.MONTH), 0, 0)); // 1.Pinsedag
            paaske = beregnPaaskedager(aar, 50);
            paaske.add(Calendar.MONTH, +1);
            helligdager.add(new KalenderElement(paaske.get
(Calendar.DAY_OF_MONTH), paaske.get(Calendar.MONTH), 0, 0)); // 2.Pinsedag

            // Andre helligdager
            helligdager.add(new KalenderElement(1, 5, 0, 0)); // 1. Mai
            helligdager.add(new KalenderElement(17, 5, 0, 0)); // Nasjonaldag
            helligdager.add(new KalenderElement(25, 12, 0, 0)); // 1.juledag
            helligdager.add(new KalenderElement(26, 12, 0, 0)); // 2.juledag
      }

/
**********************************************************************************
****************
 * Metode som sjekker om KalenderElementet er en helligdag
 * @param KalenderElement e - KalenderElement som skal sjekkes for helligdag.

**********************************************************************************
****************/
      private boolean erHelligdag(KalenderElement e) {
            boolean erHellig = false;
            for(int i=0; i<helligdager.size(); i++) {
                  if(e.erLikHelligdag(helligdager.get(i))) {
                        erHellig = true;
                        return erHellig;
                  }
            }
            return erHellig;
      }

/
**********************************************************************************
****************
 * Metode som tar imot trykket KalenderElement og endrer kalender.
 * @param KalenderElement e - KalenderElement som skal sjekkes for helligdag.

**********************************************************************************
****************/
      public void elementTrykket(KalenderElement e) {
            boolean funnetPeriode = false;
            boolean finnesValgte = finnesIValgte(e);

            FerieDAOImpl f = new FerieDAOImpl();
            ferie = f.hentAlleFerier(aar);

            if(!finnesValgte) {
                  // Finner ut om valgte element er i en ferie
                  for(int i=0; i<ferie.size() && !funnetPeriode; i++) {
                        if(ferie.get(i).erIPeriode(e)) {
                              valgte = new ArrayList<KalenderElement>();
                              Periode p = ferie.get(i).finnPeriode(e);
                              valgte = p.hentAlleIPerioden();
```

```
                              funnetPeriode = true;
                        }
                  }

                  if(!funnetPeriode && !finnesValgte) {
                        valgte = new ArrayList<KalenderElement>();
                        GregorianCalendar c = new GregorianCalendar(e.getAar(),
e.getMnd()-1, e.getDag());
                        int dag = c.get(Calendar.DAY_OF_WEEK);

                        if(dag==2) {
                              for(int i=0; i<4; i++) {
                                    valgte.add(new KalenderElement(c.get
(Calendar.DAY_OF_MONTH), c.get(Calendar.MONTH)+1, c.get(Calendar.YEAR), c.get
(Calendar.WEEK_OF_YEAR)));
                                    c.add(Calendar.DAY_OF_MONTH,+1);
                              }
                        }

                        if(dag==3) {
                              c.add(Calendar.DAY_OF_MONTH,-1);
                              for(int i=0; i<4; i++) {
                                    valgte.add(new KalenderElement(c.get
(Calendar.DAY_OF_MONTH), c.get(Calendar.MONTH)+1, c.get(Calendar.YEAR), c.get
(Calendar.WEEK_OF_YEAR)));
                                    c.add(Calendar.DAY_OF_MONTH,+1);
                              }
                        }

                        if(dag==4) {
                              c.add(Calendar.DAY_OF_MONTH,-2);
                              for(int i=0; i<4; i++) {
                                    valgte.add(new KalenderElement(c.get
(Calendar.DAY_OF_MONTH), c.get(Calendar.MONTH)+1, c.get(Calendar.YEAR), c.get
(Calendar.WEEK_OF_YEAR)));
                                    c.add(Calendar.DAY_OF_MONTH,+1);
                              }
                        }

                        if(dag==5) {
                              c.add(Calendar.DAY_OF_MONTH,-3);
                              for(int i=0; i<4; i++) {
                                    valgte.add(new KalenderElement(c.get
(Calendar.DAY_OF_MONTH), c.get(Calendar.MONTH)+1, c.get(Calendar.YEAR), c.get
(Calendar.WEEK_OF_YEAR)));
                                    c.add(Calendar.DAY_OF_MONTH,+1);
                              }
                        }

                        if(dag==6) {
                              for(int i=0; i<3; i++) {
                                    valgte.add(new KalenderElement(c.get
(Calendar.DAY_OF_MONTH), c.get(Calendar.MONTH)+1, c.get(Calendar.YEAR), c.get
(Calendar.WEEK_OF_YEAR)));
                                    c.add(Calendar.DAY_OF_MONTH,+1);
                              }
                        }

                        if(dag==7) {
                              c.add(Calendar.DAY_OF_MONTH,-1);
                              for(int i=0; i<3; i++) {
                                    valgte.add(new KalenderElement(c.get
(Calendar.DAY_OF_MONTH), c.get(Calendar.MONTH)+1, c.get(Calendar.YEAR), c.get
(Calendar.WEEK_OF_YEAR)));
                                    c.add(Calendar.DAY_OF_MONTH,+1);
                              }
```

```
                        }

                        if(dag==1) {
                              c.add(Calendar.DAY_OF_MONTH,-2);
                              for(int i=0; i<3; i++) {
                                    valgte.add(new KalenderElement(c.get
(Calendar.DAY_OF_MONTH), c.get(Calendar.MONTH)+1, c.get(Calendar.YEAR), c.get
(Calendar.WEEK_OF_YEAR)));
                                    c.add(Calendar.DAY_OF_MONTH,+1);
                              }
                        }
                        // Fjerner alle fra periode
                        for(int i=0; i<ferie.size(); i++) {
                              for(int j=0; j<valgte.size(); j++) {
                                    if(ferie.get(i).erIPeriode(valgte.get(j))) {
                                          valgte.remove(j);
                                          j--;
                                    }
                              }
                        }

                  }
                  } else {

                        for(int i=0; i<ferie.size() && !funnetPeriode; i++) {
                              if(ferie.get(i).erIPeriode(e)) {
                                    funnetPeriode = true;
                              }
                        }

                        if(!funnetPeriode) {
                              for(int i=0; i<valgte.size(); i++) {
                                    if(!valgte.get(i).equals(e)) {
                                          valgte.remove(i);
                                          i--;
                                    }
                              }
                        }
                        if(funnetPeriode) {
                              valgte.clear();
                        }
                  }
            // Fjerner alle fra perioder og oppbestillte
            lagKalender();
      }

/
**********************************************************************************
****************
 * Metode som sjekker om elementet finnes i valgte
 * @param KalenderElement e - KalenderElement som skal sjekkes
 * @return boolean funnet - True eller false

**********************************************************************************
****************/
      public boolean finnesIValgte(KalenderElement e) {
            boolean funnet = false;
            for (int i=0; i<valgte.size() && !funnet; i++) {
                  if(valgte.get(i).equals(e)){
                        funnet = true;
                        return funnet;
                  }
            }
            return funnet;
      }
```

```
/
**********************************************************************************
***************
 * Metode som lager alle bestillinger om til KalenderElement
 * @return ArrayList<KalenderElement> bestillinger - Returnerer alle
oppbestillte

**********************************************************************************
***************/
      private ArrayList<KalenderElement> bestillingTilKalender() {
            ArrayList<KalenderElement> bestillinger = new
ArrayList<KalenderElement>();
            BestillingDAOImpl best = new BestillingDAOImpl();
            tildelte = best.hentAlleTildelte(aar, hytteNavn);
            if(tildelte != null) {
                  for(int i=0; i<tildelte.size(); i++) {
                        KalenderElement fra = new KalenderElement(tildelte.get
(i).getDatoFra());
                        KalenderElement til = new KalenderElement(tildelte.get
(i).getDatoTil());
                        Periode p = new Periode(fra, til, tildelte.get(i).getSum
());
                        bestillinger.addAll(p.hentAlleIPerioden());
                  }
            }
            return bestillinger;
      }

/
**********************************************************************************
***************
 * Metode som lager alle bestillinger om til KalenderElement
 * @return ArrayList<KalenderElement> bestillinger - Returnerer alle
oppbestillte

**********************************************************************************
***************/
      private ArrayList<KalenderElement> bestillingTilKalenderIkkeTildelt() {
            ArrayList<KalenderElement> bestillinger = new
ArrayList<KalenderElement>();
            BestillingDAOImpl best = new BestillingDAOImpl();
            ikkeTildelte = best.hentBestillingerIkkeTildelt(hytteNavn, mnd);
            if(ikkeTildelte != null) {
                  for(int i=0; i<ikkeTildelte.size(); i++) {
                        KalenderElement fra = new KalenderElement
(ikkeTildelte.get(i).getDatoFra());
                        KalenderElement til = new KalenderElement
(ikkeTildelte.get(i).getDatoTil());
                        Periode p = new Periode(fra, til, ikkeTildelte.get
(i).getSum());
                        bestillinger.addAll(p.hentAlleIPerioden());
                  }
            }
            return bestillinger;
      }

/
**********************************************************************************
***************
 * Metode som henter alle sperrer
 * @return ArrayList<KalenderElement> sperrete - Returnerer alle sperrete

**********************************************************************************
***************/
      private ArrayList<KalenderElement> sperrete() {
            ArrayList<Sperre> sperrer = new ArrayList<Sperre>();
```

```
            ArrayList<KalenderElement> alleSperrete = new
ArrayList<KalenderElement>();
            HytteDAOImpl h = new HytteDAOImpl();

            sperrer = h.hentAlleSperrer(hytteNavn);
            for(int i=0; i<sperrer.size(); i++) {
                  KalenderElement startTmp = sperrer.get(i).getStart();
                  KalenderElement sluttTmp = sperrer.get(i).getSlutt();
                  KalenderElement arbeid = new KalenderElement(startTmp.getDag
(), startTmp.getMnd(), startTmp.getAar(), 0);
                  sluttTmp = KalenderElement.flyttKalemenderElementDager
(sluttTmp, 1);
                  while(!arbeid.equals(sluttTmp)) {
                        alleSperrete.add(new KalenderElement(arbeid.getDag(),
arbeid.getMnd(), arbeid.getAar(), 0));
                        arbeid = KalenderElement.flyttKalemenderElementDager
(arbeid, 1);
                  }
            }
            return alleSperrete;
      }

/
**********************************************************************************
****************
 * Metode som valgte dager i kalenderen
 * @return ArrayList<KalenderElement> getValgteDager - Returnerer alle valgte
dager

**********************************************************************************
****************/
      public ArrayList<KalenderElement> getValgteDager() {
            return valgte;
      }
}
```

KalenderElement.java

```
package hytteWeb.model;

import java.sql.Date;
import java.util.Calendar;
import java.util.GregorianCalendar;

enum Status { Ledig, Opptatt, Trukket, Bestillt, Valgt, Helligdag, Feriedag,
Ukenummer, AnnenMnd };

public class KalenderElement implements Comparable<KalenderElement> {
      private int dag;
      private int mnd;
      private int aar;
      private int uke;
      private Status status;
      private String mndAar;

      public KalenderElement(int dag, int mnd, int aar, int uke) {
            this.dag = dag;
            this.mnd = mnd;
            this.aar = aar;
            this.uke = uke;
            this.status = Status.Ledig;
            mndAar = hentManedSomTekst(this.mnd, this.aar);
      }
```

```
public KalenderElement(Date d) {
      String dato = d.toString();
      String[] liste = dato.split("-");
      this.aar = Integer.parseInt(liste[0]);
      this.mnd = Integer.parseInt(liste[1]);
      this.dag = Integer.parseInt(liste[2]);
      this.status = Status.Ledig;

      GregorianCalendar c = new GregorianCalendar(aar, mnd, dag);
      this.uke = c.get(Calendar.WEEK_OF_YEAR);
      mndAar = hentManedSomTekst(this.mnd, this.aar);
}

public KalenderElement(String dato) {
      String[] liste = dato.split("-");
      this.aar = Integer.parseInt(liste[0]);
      this.mnd = Integer.parseInt(liste[1]);
      this.dag = Integer.parseInt(liste[2]);
      this.status = Status.Ledig;

      GregorianCalendar c = new GregorianCalendar(aar, mnd, dag);
      this.uke = c.get(Calendar.WEEK_OF_YEAR);
      mndAar = hentManedSomTekst(this.mnd, this.aar);
}

public KalenderElement(String dato, boolean motsatt) {
      String[] liste = dato.split("-");
      this.aar = Integer.parseInt(liste[2]);
      this.mnd = Integer.parseInt(liste[1]);
      this.dag = Integer.parseInt(liste[0]);
      this.status = Status.Ledig;

      GregorianCalendar c = new GregorianCalendar(aar, mnd, dag);
      this.uke = c.get(Calendar.WEEK_OF_YEAR);
      mndAar = hentManedSomTekst(this.mnd, this.aar);
}

public int getDag() {
      return dag;
}

public int getMnd() {
      return mnd;
}

public int getAar() {
      return aar;
}

public int getUke() {
      return uke;
}

public Status getStatus() {
      return status;
}

public void setDag(int dag) {
      this.dag = dag;
}

public void setMnd(int mnd) {
      this.mnd = mnd;
}

public void setAar(int aar) {
```

```
		this.aar = aar;
	}

	public void setUke(int uke) {
		this.uke = uke;
	}

	public void setStatus(Status status) {
		this.status = status;
	}

	public String getMndAar() {
			return mndAar;
		}

	public void setMndAar(String mndAar) {
		this.mndAar = mndAar;
	}

/
**********************************************************************************
****************
 * Metode som sammenligner to KalenderElement som er helligdag for likhet
 * @param e - KalenderElement
 * @return lik - boolean - Returnerer true dersom to helligdag-element er like
på dag og måned

**********************************************************************************
****************/
	public boolean erLikHelligdag(KalenderElement e) {
		if(this.dag == e.getDag() && this.mnd == e.getMnd()) {
			return true;
		} else
			return false;
	}

	public int compareTo(KalenderElement e) {
		if(this.dag == e.getDag() && this.mnd == e.getMnd() && this.aar ==
e.getAar()) {
			return 0;
		} else if(this.aar < e.getAar()) {
			return -1;
		} else if(this.mnd < e.getMnd() && this.aar == e.getAar()) {
			return -1;
		} else if(this.dag < e.getDag() && this.mnd == e.getMnd() &&
this.aar == e.getAar()) {
			return -1;
		} else {
			return 1;
		}
	}

	public boolean equals(KalenderElement e) {
		if(this.dag == e.getDag() && this.mnd == e.getMnd() && this.aar ==
e.getAar())
			return true;
		 else
			return false;
	}

	public String toString() {
		String m = String.valueOf(mnd);
		if(m.length() == 1) m = "0" + m;
		String d = String.valueOf(dag);
		if(d.length() == 1) d = "0" + d;
```

```
            return aar + "-" + m + "-" + d;
      }

      private String hentManedSomTekst(int mnd, int aar) {
            String mndStreng = "";
            switch (mnd) {
                  case 1: mndStreng = "Januar"; break;
                  case 2:     mndStreng = "Februar"; break;
                  case 3: mndStreng = "Mars"; break;
                  case 4:     mndStreng = "April"; break;
                  case 5: mndStreng = "Mai"; break;
                  case 6:     mndStreng = "Juni"; break;
                  case 7: mndStreng = "Juli"; break;
                  case 8:     mndStreng = "August"; break;
                  case 9: mndStreng = "September"; break;
                  case 10: mndStreng = "Oktober"; break;
                  case 11: mndStreng = "November"; break;
                  case 12: mndStreng = "Desember"; break;
                  default: mndStreng = "Januar"; break;
            }
            return mndStreng + " " + aar;
      }

      public static KalenderElement fraGregorian(GregorianCalendar c) {
            return new KalenderElement(c.get(Calendar.DAY_OF_MONTH), c.get
(Calendar.MONTH)+1,
                        c.get(Calendar.YEAR), c.get(Calendar.WEEK_OF_YEAR));
      }

      public static KalenderElement flyttKalemenderElement(KalenderElement e,
int antMnd) {
            KalenderElement temp = e;
            GregorianCalendar c = new GregorianCalendar(e.getAar(), e.getMnd
()-1, e.getDag());
            c.add(Calendar.MONTH,+antMnd);
            temp.setDag(c.get(Calendar.DAY_OF_MONTH));
            temp.setMnd(c.get(Calendar.MONTH)+1);
            temp.setAar(c.get(Calendar.YEAR));
            temp.setUke(c.get(Calendar.WEEK_OF_YEAR));
            temp.setStatus(e.getStatus());
            temp.setMndAar(temp.hentManedSomTekst(temp.getMnd(), temp.getAar
()));
            return temp;
      }

      public static KalenderElement flyttKalemenderElementDager(KalenderElement
e, int dager) {
            KalenderElement temp = e;
            GregorianCalendar c = new GregorianCalendar(e.getAar(), e.getMnd
()-1, e.getDag());
            c.add(Calendar.DAY_OF_MONTH,+dager);
            temp.setDag(c.get(Calendar.DAY_OF_MONTH));
            temp.setMnd(c.get(Calendar.MONTH)+1);
            temp.setAar(c.get(Calendar.YEAR));
            temp.setUke(c.get(Calendar.WEEK_OF_YEAR));
            temp.setStatus(e.getStatus());
            temp.setMndAar(temp.hentManedSomTekst(temp.getMnd(), temp.getAar
()));
            return temp;
      }

}
```

Periode.java

```
package hytteWeb.model;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;

public class Periode implements Comparable<Periode> {
	KalenderElement start;
	KalenderElement slutt;
	int pris;

	public Periode(){

	}
	public Periode(KalenderElement start, KalenderElement slutt, int pris) {
		this.start = start;
		this.slutt = slutt;
		this.pris = pris;
	}

	// Hent og sett
	public KalenderElement getStart() {
		return start;
	}
	public KalenderElement getSlutt() {
		return slutt;
	}
	public int getPris() {
		return pris;
	}
	public void setStart(KalenderElement start) {
		this.start = start;
	}
	public void setSlutt(KalenderElement slutt) {
		this.slutt = slutt;
	}
	public void setPris(int pris) {
		this.pris = pris;
	}

/
**********************************************************************************
***************
 * Metode som sammenligner to Perioder
 * @param p - Periode
 * @return int - Returnerer -1, 0 eller 1

**********************************************************************************
***************/
	public int compareTo(Periode p) {
		if(start.equals(p.start) && slutt.equals(p.slutt))
			return 0;
		else if(start.compareTo(p.start) < 0)
			return -1;
		else
			return 1;
	}

/
**********************************************************************************
***************
 * Metode som sammenligner to Perioder om de er like
 * @param p - Periode
```

```
 * @return boolean - Returnerer true dersom like, false ellers.

**********************************************************************************
****************/
      public boolean equals(Periode p) {
            if(start.equals(p.start) && slutt.equals(p.slutt))
                  return true;
            else
                  return false;
      }

/
**********************************************************************************
****************
 * Metode som finner ut om et kalenderElement finnes i perioden
 * @param e - KalenderElement
 * @return lik - boolean - Returnerer true dersom elementet er mellom start og
slutt

**********************************************************************************
****************/
      public boolean finnesIPerioden(KalenderElement e) {
            GregorianCalendar st = new GregorianCalendar(start.getAar
(),start.getMnd()-1,start.getDag());
            GregorianCalendar sl = new GregorianCalendar(slutt.getAar
(),slutt.getMnd()-1,slutt.getDag());
            GregorianCalendar akt = new GregorianCalendar(e.getAar(),e.getMnd
()-1,e.getDag());
            sl.add(Calendar.DAY_OF_MONTH,1);
            while(!st.equals(sl)) {
                  if(st.equals(akt))
                        return true;
                  st.add(Calendar.DAY_OF_MONTH,1);
            }
            return false;
      }

/
**********************************************************************************
****************
 * Metode som henter alle kalenderelement i perioden
 * @return ArrayList<KalenderElement> - Returnerer en ArrayList av alle
elementene i perioden

**********************************************************************************
****************/
      public ArrayList<KalenderElement> hentAlleIPerioden() {
            ArrayList<KalenderElement> temp = new ArrayList<KalenderElement>();
            GregorianCalendar st = new GregorianCalendar(start.getAar
(),start.getMnd(),start.getDag());
            GregorianCalendar sl = new GregorianCalendar(slutt.getAar
(),slutt.getMnd(),slutt.getDag());
            sl.add(Calendar.DAY_OF_MONTH,1);
            while(!st.equals(sl)) {
                  temp.add(new KalenderElement(st.get(Calendar.DAY_OF_MONTH),
st.get(Calendar.MONTH),
                              st.get(Calendar.YEAR), st.get
(Calendar.WEEK_OF_YEAR)));
                  st.add(Calendar.DAY_OF_MONTH,1);
            }
            return temp;
      }
}
```

Sperre.java

```
package hytteWeb.model;

public class Sperre {
	int id;
	String hytte;
	KalenderElement start;
	KalenderElement slutt;

	// KonstruktÃ¸rer
	public Sperre() {
		id = 0;
		hytte = "";
		start = new KalenderElement(1,1,2010,1);
		slutt = new KalenderElement(1,1,2010,1);
	}

	public Sperre(int id, String hytte, KalenderElement start, KalenderElement
slutt) {
		this.id = id;
		this.hytte = hytte;
		this.start = start;
		this.slutt = slutt;
	}

	public Sperre(String hytte, KalenderElement start, KalenderElement slutt)
{
		this.id = 0;
		this.hytte = hytte;
		this.start = start;
		this.slutt = slutt;
	}


	// Get and set
	public int getId() {
		return id;
	}

	public String getHytte() {
		return hytte;
	}

	public KalenderElement getStart() {
		return start;
	}

	public KalenderElement getSlutt() {
		return slutt;
	}

	public void setId(int id) {
		this.id = id;
	}

	public void setHytte(String hytte) {
		this.hytte = hytte;
	}

	public void setStart(KalenderElement start) {
		this.start = start;
	}

	public void setSlutt(KalenderElement slutt) {
```

```
            this.slutt = slutt;
      }

      public boolean equals(Sperre obj) {
            if(hytte.equals(obj.hytte) && start.equals(obj.start) &&
slutt.equals(obj.slutt)) {
                  return true;
            } else {
                  return false;
            }
      }
}
```

DAO

BestillingDAOImpl.java

```
package hytteWeb.dao.context;

import hytteWeb.model.Bestilling;
import hytteWeb.model.Ferie;
import hytteWeb.model.KalenderElement;
import hytteWebUtility.HjelpeKlasserDato;
import hytteWebUtility.Utility;

import java.sql.Date;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.UUID;

public class BestillingDAOImpl {
	DatabaseDAOImpl db;
	FerieDAOImpl fDAO;
	HytteDAOImpl hDAO;
	InnstillingDAOImpl iDAO;

	public BestillingDAOImpl(){
		db = new DatabaseDAOImpl();
		fDAO = new FerieDAOImpl();
		hDAO = new HytteDAOImpl();
		iDAO = new InnstillingDAOImpl();
	}

	/
**************************************************************************
	 * Metode som henter bestilling etter gitt id
	 * @param id Den unike id'en til bestillingen som skal returneres
	 * @return Bestilling returnerer bestillings objektet

**************************************************************************/
	public Bestilling hentBestilling(int id){
		db.settSql("SELECT * FROM BESTILLING WHERE Nummer='" + id + "'");
		if(db.kobleTil(true)){
			Bestilling tmp = new Bestilling();
			ResultSet rs = db.hentResultat();
			try {
				while(rs.next()) {
					tmp.setNummer(rs.getInt("Nummer"));
					tmp.setKunde(rs.getString("Kunde"));
					tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
					tmp.setHytte(rs.getString("Hytte"));
					tmp.setDatoFra(rs.getDate("DatoFra").toString());
					tmp.setDatoTil(rs.getDate("DatoTil").toString());
					tmp.setSum(rs.getInt("Sum"));
					tmp.setBetalt(rs.getBoolean("Betalt"));
					tmp.setBekreftet(rs.getBoolean("Bekreftet"));
					tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
					tmp.setAutKode(rs.getString("AutKode"));
					tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
				}
```

```
                } catch (SQLException e) {
                      e.printStackTrace();
                }
                db.kobleFra();
                return tmp;
          }
          db.kobleFra();
          return null;
     }

     /
*****************************************************************************
      * Metode som henter bestilling etter gitt autorisasjonskode for
      * bekreftet og tildelt.
      * @param autKode Autorisasjonskoden for bekreftelse
      * @param autKodeTildelt Autorisasjonskoden for tildeling
      * @return Bestilling returnerer bestillings objektet

*****************************************************************************/
     public Bestilling hentBestilling(String autKode, boolean autKodeTildelt){
          if(autKodeTildelt){
                db.settSql("SELECT * FROM BESTILLING WHERE AutKodeTildelt='" +
autKode + "'");
          } else {
                db.settSql("SELECT * FROM BESTILLING WHERE AutKode='" +
autKode + "'");
          }
          if(db.kobleTil(true)){
                Bestilling tmp = new Bestilling();
                ResultSet rs = db.hentResultat();
                try {
                      while(rs.next()) {
                            tmp.setNummer(rs.getInt("Nummer"));
                            tmp.setKunde(rs.getString("Kunde"));
                            tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                            tmp.setHytte(rs.getString("Hytte"));
                            tmp.setDatoFra(rs.getDate("DatoFra").toString());
                            tmp.setDatoTil(rs.getDate("DatoTil").toString());
                            tmp.setSum(rs.getInt("Sum"));
                            tmp.setBetalt(rs.getBoolean("Betalt"));
                            tmp.setBekreftet(rs.getBoolean("Bekreftet"));
                            tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
                            tmp.setAutKode(rs.getString("AutKode"));
                            tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
                      }
                } catch (SQLException e) {
                      e.printStackTrace();
                }
                db.kobleFra();
                return tmp;
          }
          db.kobleFra();
          return null;
     }

     /
*****************************************************************************
      * Metode som sjekker om bestilling eksisterer etter gitt
autorisasjonskode for
      * bekreftet og tildelt.
      * @param autKode Autorisasjonskoden for bekreftelse
      * @param autKodeTildelt Autorisasjonskoden for tildeling
      * @return boolean returnerer om bestillingen eksisterer eller ikke
```

```
**************************************************************************/
      public boolean finnesBestilling(String autKode, boolean autKodeTildelt){
            if(autKodeTildelt){
                  db.settSql("SELECT * FROM BESTILLING WHERE AutKodeTildelt='" +
autKode + "'");
            } else {
                  db.settSql("SELECT * FROM BESTILLING WHERE AutKode='" +
autKode + "'");
            }
            if(db.kobleTil(true)){

                  db.kobleFra();
                  return true;
            }
            db.kobleFra();
            return false;
      }


      /************************************************************
       * Metode som henter ut alle tildelte bestillinger for
       * et gitt aar
       * @param aar aaret det skal hentes ut bestillinger fra
       * @return ArrayList<Bestilling> returnerer alle bestillinger i en
arraylist
       ************************************************************/
      public ArrayList<Bestilling> hentAlleTildelte(int aar){

            Calendar aarFor = new GregorianCalendar(aar-1, 11, 15);
            Calendar aarEtter = new GregorianCalendar(aar+1, 1, 10);
            String foer = aarFor.get(Calendar.YEAR) + "-" + (aarFor.get
(Calendar.MONTH)+1) + "-" + aarFor.get(Calendar.DATE);
            String etter = aarEtter.get(Calendar.YEAR) + "-" + aarEtter.get
(Calendar.MONTH) + "-" + aarEtter.get(Calendar.DATE);
            System.out.println(foer);
            System.out.println(etter);
            db.settSql("SELECT * FROM BESTILLING WHERE BekreftetTildelt=1 AND
Betalt=0 AND" +
                        " DatoFra > '" + foer + "' AND DatoTil < '" + etter +
"'");
            if(db.kobleTil(true)){
                  ArrayList<Bestilling> bestillinger = new ArrayList<Bestilling>
();
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {

                              Bestilling tmp = new Bestilling();
                              tmp.setNummer(rs.getInt("Nummer"));
                              tmp.setKunde(rs.getString("Kunde"));
                              tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                              tmp.setHytte(rs.getString("Hytte"));
                              tmp.setDatoFra(rs.getDate("DatoFra").toString());
                              tmp.setDatoTil(rs.getDate("DatoTil").toString());
                              tmp.setSum(rs.getInt("Sum"));
                              tmp.setBetalt(rs.getBoolean("Betalt"));
                              tmp.setBekreftet(rs.getBoolean("Bekreftet"));
                              tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
                              tmp.setAutKode(rs.getString("AutKode"));
                              tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
                              bestillinger.add(tmp);
                        }
```

```
                } catch (SQLException e) {
                      e.printStackTrace();
                }
                db.kobleFra();
                return bestillinger;
          }
          db.kobleFra();
          return null;
     }

     /**************************************************************
      * Metode som henter ut alle tildelte bestillinger for
      * en gitt bruker
      * @param bruker Brukernavn det skal hentes ut bestillinger fra
      * @return ArrayList<Bestilling> returnerer alle bestillinger i en
arraylist
      **************************************************************/
     public ArrayList<Bestilling> hentAlleKunde(String bruker){
          db.settSql("SELECT * FROM BESTILLING WHERE Kunde='" + bruker + "'");
          if(db.kobleTil(true)){
                ArrayList<Bestilling> bestillinger = new ArrayList<Bestilling>
();
                ResultSet rs = db.hentResultat();
                try {
                      while(rs.next()) {

                            Bestilling tmp = new Bestilling();
                            tmp.setNummer(rs.getInt("Nummer"));
                            tmp.setKunde(rs.getString("Kunde"));
                            tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                            tmp.setHytte(rs.getString("Hytte"));
                            tmp.setDatoFra(rs.getDate("DatoFra").toString());
                            tmp.setDatoTil(rs.getDate("DatoTil").toString());
                            tmp.setSum(rs.getInt("Sum"));
                            tmp.setBetalt(rs.getBoolean("Betalt"));
                            tmp.setBekreftet(rs.getBoolean("Bekreftet"));
                            tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
                            tmp.setAutKode(rs.getString("AutKode"));
                            tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
                            bestillinger.add(tmp);
                      }
                } catch (SQLException e) {
                      e.printStackTrace();
                }
                db.kobleFra();
                return bestillinger;
          }
          db.kobleFra();
          return null;
     }

     /**************************************************************
      * Metode som henter ut alle bekreftet bestillinger for
      * et gitt aar
      * @param aar aaret det skal hentes ut bestillinger fra
      * @return ArrayList<Bestilling> returnerer alle bestillinger i en
arraylist
      **************************************************************/
     public ArrayList<Bestilling> hentAlleBekreftet(int aar){

          Calendar aarFor = new GregorianCalendar(aar-1, 11, 15);
          Calendar aarEtter = new GregorianCalendar(aar+1, 1, 10);
```

```
            String foer = aarFor.get(Calendar.YEAR) + "-" + (aarFor.get
(Calendar.MONTH)+1) + "-" + aarFor.get(Calendar.DATE);
            String etter = aarEtter.get(Calendar.YEAR) + "-" + aarEtter.get
(Calendar.MONTH) + "-" + aarEtter.get(Calendar.DATE);
            System.out.println(foer);
            System.out.println(etter);
            db.settSql("SELECT * FROM BESTILLING WHERE BekreftetTildelt=0 AND
Bekreftet=1 AND" +
                        " DatoFra > '" + foer + "' AND DatoTil < '" + etter +
"'");
            if(db.kobleTil(true)){
                  ArrayList<Bestilling> bestillinger = new ArrayList<Bestilling>
();
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {

                              Bestilling tmp = new Bestilling();
                              tmp.setNummer(rs.getInt("Nummer"));
                              tmp.setKunde(rs.getString("Kunde"));
                              tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                              tmp.setHytte(rs.getString("Hytte"));
                              tmp.setDatoFra(rs.getDate("DatoFra").toString());
                              tmp.setDatoTil(rs.getDate("DatoTil").toString());
                              tmp.setSum(rs.getInt("Sum"));
                              tmp.setBetalt(rs.getBoolean("Betalt"));
                              tmp.setBekreftet(rs.getBoolean("Bekreftet"));
                              tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
                              tmp.setAutKode(rs.getString("AutKode"));
                              tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
                              bestillinger.add(tmp);
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
                  return bestillinger;
            }
            db.kobleFra();
            return null;
      }
      /************************************************************
       * Metode som henter ut alle tildelte bestillinger for
       * et gitt aar og gitt hytte
       * @param aar aaret det skal hentes ut bestillinger fra
       * @param hyttenavn navnet paa hytten det skal hentes ut bestillinger fra
       * @return ArrayList<Bestilling> returnerer alle bestillinger i en
arraylist
       ************************************************************/
      public ArrayList<Bestilling> hentAlleTildelte(int aar, String hyttenavn){


            Calendar aarFor = new GregorianCalendar(aar-1, 11, 15);
            Calendar aarEtter = new GregorianCalendar(aar+1, 1, 10);
            String foer = aarFor.get(Calendar.YEAR) + "-" + (aarFor.get
(Calendar.MONTH)+1) + "-" + aarFor.get(Calendar.DATE);
            String etter = aarEtter.get(Calendar.YEAR) + "-" + aarEtter.get
(Calendar.MONTH) + "-" + aarEtter.get(Calendar.DATE);

            db.settSql("SELECT * FROM BESTILLING WHERE BekreftetTildelt=1 AND
Hytte='"+ hyttenavn +"' AND" +
                        " DatoFra > '" + foer + "' AND DatoTil < '" + etter +
"'");
```

```
            if(db.kobleTil(true)){
                  ArrayList<Bestilling> bestillinger = new ArrayList<Bestilling>
();
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              Bestilling tmp = new Bestilling();
                              tmp.setNummer(rs.getInt("Nummer"));
                              tmp.setKunde(rs.getString("Kunde"));
                              tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                              tmp.setHytte(rs.getString("Hytte"));
                              tmp.setDatoFra(rs.getDate("DatoFra").toString());
                              tmp.setDatoTil(rs.getDate("DatoTil").toString());
                              tmp.setSum(rs.getInt("Sum"));
                              tmp.setBetalt(rs.getBoolean("Betalt"));
                              tmp.setBekreftet(rs.getBoolean("Bekreftet"));
                              tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
                              tmp.setAutKode(rs.getString("AutKode"));
                              tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
                              bestillinger.add(tmp);
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
                  return bestillinger;
            }
            db.kobleFra();
            return null;
      }

      /**************************************************************
       * Metode som henter ut alle inneværende bestillinger hytte
       * en gitt maaned
       * @param hyttenavn Navnet paa hytten det skal hentes ut bestillinger fra
       * @param maned maaneden det skal hentes ut bestillinger fra
       * @return ArrayList<Bestilling> returnerer alle bestillinger i en
arraylist
       **************************************************************/
      public ArrayList<Bestilling> hentBestillinger(String hytteNavn, int maned)
{
            Calendar tmp2 = new GregorianCalendar();
            Calendar datoStart = new GregorianCalendar(tmp2.get(Calendar.YEAR),
maned-2, 23);
            Calendar datoSlutt = new GregorianCalendar(tmp2.get(Calendar.YEAR),
maned, 7);

            String dStart = datoStart.get(Calendar.YEAR) + "-" +  (datoStart.get
(Calendar.MONTH)+1) + "-" + datoStart.get(Calendar.DATE);
            String dSlutt = datoSlutt.get(Calendar.YEAR) + "-" +  (datoSlutt.get
(Calendar.MONTH)+1) + "-" + datoSlutt.get(Calendar.DATE);

            db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ hytteNavn + "'
AND" +
                        " DatoFra > '" + dStart + "' AND DatoTil < '" + dSlutt +
"'");
            if(db.kobleTil(true)){
                  ArrayList<Bestilling> bestillinger = new ArrayList<Bestilling>
();
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              Bestilling tmp = new Bestilling();
```

```
                              tmp.setNummer(rs.getInt("Nummer"));
                              tmp.setKunde(rs.getString("Kunde"));
                              tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                              tmp.setHytte(rs.getString("Hytte"));
                              tmp.setDatoFra(rs.getTimestamp("DatoFra").toString
());
                              tmp.setDatoTil(rs.getTimestamp("DatoTil").toString
());
                              tmp.setSum(rs.getInt("Sum"));
                              tmp.setBetalt(rs.getBoolean("Betalt"));
                              tmp.setBekreftet(rs.getBoolean("Bekreftet"));
                              tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
                              tmp.setAutKode(rs.getString("AutKode"));
                              tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
                              bestillinger.add(tmp);
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
                  return bestillinger;
            }
            db.kobleFra();
            return null;
      }

      /************************************************************
       * Metode som henter ut alle inneværende bestillinger hytte
       * en gitt maaned som ikke er tildelt
       * @param hyttenavn Navnet paa hytten det skal hentes ut bestillinger fra
       * @param maned maaneden det skal hentes ut bestillinger fra
       * @return ArrayList<Bestilling> returnerer alle bestillinger i en
arraylist
       ************************************************************/
      public ArrayList<Bestilling> hentBestillingerIkkeTildelt(String hytteNavn,
int maned){
            Calendar tmp2 = new GregorianCalendar();
            Calendar datoStart = new GregorianCalendar(tmp2.get(Calendar.YEAR),
maned-2, 23);
            Calendar datoSlutt = new GregorianCalendar(tmp2.get(Calendar.YEAR),
maned, 7);

            String dStart = datoStart.get(Calendar.YEAR) + "-" +  (datoStart.get
(Calendar.MONTH)+1) + "-" + datoStart.get(Calendar.DATE);
            String dSlutt = datoSlutt.get(Calendar.YEAR) + "-" +  (datoSlutt.get
(Calendar.MONTH)+1) + "-" + datoSlutt.get(Calendar.DATE);

            db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ hytteNavn + "'
AND Bekreftet=1 AND BekreftetTildelt=0 AND" +
                        " DatoFra > '" + dStart + "' AND DatoTil < '" + dSlutt +
"'");
            if(db.kobleTil(true)){
                  ArrayList<Bestilling> bestillinger = new ArrayList<Bestilling>
();
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              Bestilling tmp = new Bestilling();
                              tmp.setNummer(rs.getInt("Nummer"));
                              tmp.setKunde(rs.getString("Kunde"));
                              tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                              tmp.setHytte(rs.getString("Hytte"));
```

```
                              tmp.setDatoFra(rs.getDate("DatoFra").toString());
                              tmp.setDatoTil(rs.getDate("DatoTil").toString());
                              tmp.setSum(rs.getInt("Sum"));
                              tmp.setBetalt(rs.getBoolean("Betalt"));
                              tmp.setBekreftet(rs.getBoolean("Bekreftet"));
                              tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
                              tmp.setAutKode(rs.getString("AutKode"));
                              tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
                              bestillinger.add(tmp);
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
                  return bestillinger;
            }
            db.kobleFra();
            return null;
      }

      /*************************************************************
       * Metode som registrerer en hytte for bruker
       * en gitt tidsperiode
       * @param hyttenavn Navnet paa hytten det skal legges inn bestilling paa
       * @param bruker Navnet paa brukeren det skal legges inn bestilling paa
       * @param fra Kalender element fra
       * @param til Kalender element til
       * @return Bestilling returnerer den registrert bestillingen
       *************************************************************/
      public Bestilling reserverHytte(String hytteNavn, String bruker,
KalenderElement fra, KalenderElement til){
            db.settSql("INSERT INTO BESTILLING (Kunde, BesDato, Hytte, DatoFra,
DatoTil, Sum, Betalt, Bekreftet, BekreftetTildelt, AutKode, AutKodeTildelt) "
                        + " VALUES('" + bruker + "', '" +
HjelpeKlasserDato.dagensDato() + "', '" + hytteNavn + "', " +
                        "'"+ fra.toString() + "', '" + til.toString() + "', '"
+beregnPris(hytteNavn, fra, til) + "', '0', '0', '0', '"+genererAutKode()+"',
'"+genererAutKode()+"')");

            if(db.kobleTil(false)){

            }

            db.settSql("SELECT * FROM BESTILLING WHERE Kunde='" + bruker + "'
AND Hytte='" + hytteNavn + "' AND "
                        + "DatoFra='" +  fra.toString() + "'");
            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              Bestilling tmp = new Bestilling();
                              tmp.setNummer(rs.getInt("Nummer"));
                              tmp.setKunde(rs.getString("Kunde"));
                              tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                              tmp.setHytte(rs.getString("Hytte"));
                              tmp.setDatoFra(rs.getDate("DatoFra").toString());
                              tmp.setDatoTil(rs.getDate("DatoTil").toString());
                              tmp.setSum(rs.getInt("Sum"));
                              tmp.setBetalt(rs.getBoolean("Betalt"));
                              tmp.setBekreftet(rs.getBoolean("Bekreftet"));
                              tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
                              tmp.setAutKode(rs.getString("AutKode"));
```

```
                              tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
                              db.kobleFra();
                              return tmp;
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            }
            db.kobleFra();
            return null;
      }

/
**********************************************************************************
*************
 * Metode som beregner pris for en hytte i et gitt tidspunket
 * @param hyttenavn Navnet paa hytten det skal beregnes pris paa
 * @param fra Kalender element fra
 * @param til Kalender element til
 * @return int pris

**********************************************************************************
*************/
      private int beregnPris(String hytteNavn, KalenderElement fra,
KalenderElement til) {
            int sluttPris = fDAO.hentFeriePris(fra, til);
            boolean erFerie = false;
            int PrisManTor = 0;
            int PrisHelg = 0;
            int PrisEnkeltdag = 0;
            if(sluttPris > 0)
                  erFerie = true;

            if(erFerie) {
                  return sluttPris;
            } else {
                  Bestilling b = new Bestilling();
                  b.setDatoFra(fra.toString());
                  b.setDatoTil(til.toString());
                  // Henter priser for hytten
                  db.settSql("SELECT PrisManTor, PrisHelg, PrisEnkeltdag FROM
HYTTE WHERE Navn='" + hytteNavn + "'");
                  if(db.kobleTil(true)){
                        ResultSet rs = db.hentResultat();
                        try {
                              while(rs.next()) {
                                    PrisManTor = rs.getInt("PrisManTor");
                                    PrisHelg = rs.getInt("PrisHelg");
                                    PrisEnkeltdag = rs.getInt("PrisEnkeltdag");
                              }
                        } catch (SQLException e) {
                              e.printStackTrace();
                        }
                        db.kobleFra();
                  }

                  if(HjelpeKlasserDato.erEnkeltDag(b))
                        sluttPris = PrisEnkeltdag;
                  if(HjelpeKlasserDato.erHelg(b))
                        sluttPris = PrisHelg;
                  if(HjelpeKlasserDato.erManTor(b))
                        sluttPris = PrisManTor;
            }
            db.kobleFra();
```

```
            return sluttPris;
      }


      /*************************************************************
       * Metode som registrerer en hytte for bruker
       * en gitt tidsperiode fra admin
       * @param hyttenavn Navnet paa hytten det skal legges inn bestilling paa
       * @param bruker Navnet paa brukeren det skal legges inn bestilling paa
       * @param fra Kalender element fra
       * @param til Kalender element til
       * @param bekreftet Om den er bekreftet
       * @param tildelt Om den er tildelt
       * @return Bestilling returnerer den registrert bestillingen
       *************************************************************/
      public Bestilling reserverHytteAdmin(String hytteNavn, String bruker,
KalenderElement fra, KalenderElement til, int bekreftet, int tildelt){
            db.settSql("INSERT INTO BESTILLING (Kunde, BesDato, Hytte, DatoFra,
DatoTil, Sum, Betalt, Bekreftet, BekreftetTildelt, AutKode, AutKodeTildelt) "
                        + " VALUES('" + bruker + "', '" +
HjelpeKlasserDato.dagensDato() + "', '" + hytteNavn + "', " +
                        "'"+ fra.toString() + "', '" + til.toString() + "', "+
beregnPris(hytteNavn, fra, til) +", '0', '"+bekreftet+"', '"+tildelt+ "',
'"+genererAutKode()+"', '"+genererAutKode()+"')");

            if(db.kobleTil(false)){

            }

            db.settSql("SELECT * FROM BESTILLING WHERE Kunde='" + bruker + "'
AND Hytte='" + hytteNavn + "' AND "
                        + "DatoFra='" +  fra.toString() + "'");
            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              Bestilling tmp = new Bestilling();
                              tmp.setNummer(rs.getInt("Nummer"));
                              tmp.setKunde(rs.getString("Kunde"));
                              tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                              tmp.setHytte(rs.getString("Hytte"));
                              tmp.setDatoFra(rs.getDate("DatoFra").toString());
                              tmp.setDatoTil(rs.getDate("DatoTil").toString());
                              tmp.setSum(rs.getInt("Sum"));
                              tmp.setBetalt(rs.getBoolean("Betalt"));
                              tmp.setBekreftet(rs.getBoolean("Bekreftet"));
                              tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
                              tmp.setAutKode(rs.getString("AutKode"));
                              tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
                              return tmp;
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();

            }

            db.kobleFra();
            return null;
      }

      public Bestilling loddTrekning(){
```

```
            return null;
      }

      /************************************************************
       * Metode som bekrefter en reservasjon
       * @param autKode Autorisasjonskode for bekreftelse
       * @return boolean Returnerer om bekreftelsen gikk gjennom
       ************************************************************/
      public boolean bekreftReservasjon(String autKode){
            db.settSql("UPDATE BESTILLING SET Bekreftet='1' WHERE AutKode='" +
autKode + "'");

            if(db.kobleTil(false)){

                  db.kobleFra();
                  return true;
            }
            db.kobleFra();
            return false;
      }

      /************************************************************
       * Metode som bekrefter en tildeling av en reservasjon
       * @param autKode Autorisasjonskode for tildeling
       * @return boolean Returnerer om tildelingen gikk gjennom
       ************************************************************/
      public boolean bekreftTildeling(String autKode){
            db.settSql("UPDATE BESTILLING SET BekreftetTildelt='1' WHERE
AutKodeTildelt='" + autKode + "'");

            if(db.kobleTil(false)){
                  db.kobleFra();

                  Bestilling best = hentBestilling(autKode, true);
                  if(fDAO.tilhorerFerie(best) != null){
                        // Ferie
                        settKarantene(best.getHytte(), best.getKunde(), "Ferie",
best.getDatoFra());
                  } else {
                        // Periode
                        settKarantene(best.getHytte(), best.getKunde(),
"Periode", best.getDatoFra());
                  }
                  return true;
            }
            db.kobleFra();
            return false;
      }



      /************************************************************
       * Metode som bekrefter en tildeling av en reservasjon (admin)
       * @param bid id for bestilling
       * @return boolean Returnerer om tildelingen gikk gjennom
       ************************************************************/
      public boolean bekreftTildelingAdmin(int bid){
            db.settSql("UPDATE BESTILLING SET BekreftetTildelt='1' WHERE
Nummer='"+ bid + "'");

            if(db.kobleTil(false)){

                  db.kobleFra();
                  return true;
            }
            db.kobleFra();
```

```
            return false;
      }

      public boolean slettReservasjon(Bestilling best){
            if(harTrekning(best)){
                  // Bestillingen er i trekning
                  if(erTildelt(best)){
                        // Bestilling er tildelt
                        if(best.getBekreftetTildelt() == false){
                              // Tildeling er ikke bekreftet

                              return slettUbekreftetTildeling(best);

                        } else {
                              // Tildeling er bekreftet
                              // Skal ikke slette noe -- bruker må bruke
«Kanseller tildeling»-linken
                              return false;
                        }
                  } else {
                        // Bestilling er ikke tildelt
                        System.out.println("ikke tildelt");

                        db.settSql("DELETE FROM BESTILLING WHERE Nummer='" +
best.getNummer() + "'");
                        if(db.kobleTil(false)){
                              db.kobleFra();

                              db.settSql("DELETE FROM TREKNING WHERE
Bestilling='" + best.getNummer() + "'");
                              if(db.kobleTil(false)){
                                    db.kobleFra();
                                    return true;
                              }
                        }
                        db.kobleFra();
                        return false;
                  }
            } else {
                  // Bestillingen er ikke i trekning
                  // Vi trenger bare å slette den fra BESTILLING
                  System.out.println("ikke i trekning");

                  db.settSql("DELETE FROM BESTILLING WHERE Nummer='" +
best.getNummer() + "'");
                  if(db.kobleTil(false)){
                        db.kobleFra();
                        return true;
                  }
                  db.kobleFra();
                  return false;
            }
      }

      /**********************************************************
       * Metode som sletter en tildeling
       * @param best Bestillingsobjektet
       * @return boolean Returnerer om slettingen gikk gjennom
       **********************************************************/
      public boolean slettTildeling(Bestilling best){
            if(best.getBekreftetTildelt()){
                  // Dersom bekreftet tildelt
                  // Slettes kun fra TREKNINGER
                  // Neste på listen velges

                  return slettBekreftetTildeling(best);
```

```
            } else {
                  // Dersom ikke bekreftet tildelt

                  return slettUbekreftetTildeling(best);
            }
      }

      /**************************************************************
       * Metode som sletter en ubekreftet tildeling
       * @param best Bestillingsobjektet
       * @return boolean Returnerer om slettingen gikk gjennom
       **************************************************************/
      private boolean slettUbekreftetTildeling(Bestilling best){
            // Hente trekningsnummer før vi sletter
            int trekningsNr = getTrekningsNr(best);

            db.settSql("DELETE FROM BESTILLING WHERE Nummer='" + best.getNummer
() + "'");
            if(db.kobleTil(false)){
                  db.kobleFra();

                  db.settSql("DELETE FROM TREKNING WHERE Bestilling='" +
best.getNummer() + "'");
                  if(db.kobleTil(false)){
                        db.kobleFra();
                  }
            } else {
                  db.kobleFra();
                  return false;
            }

            int storrelse = -1;
            db.settSql("SELECT COUNT(*) AS storrelse FROM TREKNING WHERE
Nummer='" + trekningsNr + "'");
            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              storrelse = rs.getInt("storrelse");
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            } else {
                  db.kobleFra();
                  return false;
            }

            if(storrelse > 0){
                  // TREKNING er ikke tom

                  Bestilling ny = hentBestilling(getForsteBestNr(trekningsNr));
                  sendTildelEpost(ny);
            }
            return true;
      }

      /**************************************************************
       * Metode som sletter en bekreftet tildeling
       * @param best Bestillingsobjektet
       * @return boolean Returnerer om slettingen gikk gjennom
       **************************************************************/
      private boolean slettBekreftetTildeling(Bestilling best){
            // Hente trekningsnummer før vi sletter
            int trekningsNr = -1;
```

```
            trekningsNr = getTrekningsNr(best);

            db.settSql("DELETE FROM TREKNING WHERE Bestilling='" +
best.getNummer() + "'");

            if(db.kobleTil(false)){
                  db.kobleFra();
            } else {
                  db.kobleFra();
                  return false;
            }

            int storrelse = -1;
            db.settSql("SELECT COUNT(*) AS storrelse FROM TREKNING WHERE
Nummer='" + trekningsNr + "'");
            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              storrelse = rs.getInt("storrelse");
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            } else {
                  db.kobleFra();
                  return false;
            }

            if(storrelse > 0){
                  // TREKNING er ikke tom

                  Bestilling ny = hentBestilling(getForsteBestNr(trekningsNr));
                  if(ny != null){
                        sendTildelEpost(ny);
                  } else {
                        return false;
                  }
            }
            return true;
      }

      /**********************************************************************
       * Genererer en unik kode
       * @return String - Kode
       **********************************************************************/
      public static String genererAutKode() {
            return UUID.randomUUID().toString();
      }

      /**********************************************************************
       * Bekrefter betalinger
       * @return boolean Kode
       **********************************************************************/
      public boolean bekreftBetalinger(ArrayList<Bestilling> best) {
            String setning = "UPDATE BESTILLING SET Betalt='1' WHERE ";
            for(int i = 0; i < best.size();i++){
                  setning += "nummer=" + best.get(i).getNummer();
                  if(i < best.size() - 1) setning += " OR ";
            }
            db.settSql(setning);
            if(db.kobleTil(false)){

                  db.kobleFra();
                  return true;
```

```
            }
            db.kobleFra();
            return false;
      }

/*****************************************************************
 * Genererer en rapport
 * @param id Hyttenavn evt Alle
 * @param aar Aar det skal genereres rapport paa
 * @param tildelt generere for tildelte
 * @param betalt generere for betalte
 * @return ArrayList<Bestilling> Returnerer arraylist av bestillinger
 *****************************************************************/
public ArrayList<Bestilling> genererRapport(String id, int aar, boolean tildelt,
int betalt) {
      // Dersom alle hyttene
      if(id.equals("Alle")){
            if(betalt == 0){
                  if(tildelt){
                        db.settSql("SELECT * FROM BESTILLING WHERE Bekreftet=1
AND Betalt=0 AND BekreftetTildelt=1 AND" +
                                    " DatoFra > '" + aar + "-01-01'");
                  }else{
                        db.settSql("SELECT * FROM BESTILLING WHERE Bekreftet=1
AND Betalt=0 AND" +
                                    " DatoFra > '" + aar + "-01-01'");
                  }
            }else if(betalt == 1){
                  if(tildelt){
                        db.settSql("SELECT * FROM BESTILLING WHERE Bekreftet=1
AND Betalt=1 AND BekreftetTildelt=1 AND" +
                                    " DatoFra > '" + aar + "-01-01'");
                  }else{
                        db.settSql("SELECT * FROM BESTILLING WHERE Betalt=1 AND
Bekreftet=1 AND" +
                                    " DatoFra > '" + aar + "-01-01'");
                  }
            }else{
                  if(tildelt){
                        db.settSql("SELECT * FROM BESTILLING WHERE Bekreftet=1
AND BekreftetTildelt=1 AND" +
                                    " DatoFra > '" + aar + "-01-01'");
                  }else{
                        db.settSql("SELECT * FROM BESTILLING WHERE Bekreftet=1
AND" +
                                    " DatoFra > '" + aar + "-01-01'");
                  }
            }
      // Dersom enkelthytte
      }else{
            if(betalt == 0){
                  if(tildelt){
                        db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Bekreftet=1 AND Betalt=0 AND BekreftetTildelt=1 AND" +
                                    " DatoFra > '" + aar + "-01-01'");
                  }else{
                        db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Bekreftet=1 AND Betalt=0 AND" +
                                    " DatoFra > '" + aar + "-01-01'");
                  }
            }else if(betalt==1){
                  if(tildelt){
                        db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Betalt=1 AND Bekreftet=1 AND BekreftetTildelt=1 AND" +
                                    " DatoFra > '" + aar + "-01-01'");
                  }else{
```

```
                        db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Betalt=1 AND Bekreftet=1 AND" +
                                    " DatoFra > '" + aar + "-01-01'");
                  }
            }else{
                  if(tildelt){
                        db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Bekreftet=1 AND BekreftetTildelt=1 AND" +
                                    " DatoFra > '" + aar + "-01-01'");
                  }else{
                        db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Bekreftet=1 AND" +
                                    " DatoFra > '" + aar + "-01-01'");
                  }
            }
      }

            if(db.kobleTil(true)){
                  ArrayList<Bestilling> bestillinger = new ArrayList<Bestilling>
();
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              Bestilling tmp = new Bestilling();
                              tmp.setNummer(rs.getInt("Nummer"));
                              tmp.setKunde(rs.getString("Kunde"));
                              tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                              tmp.setHytte(rs.getString("Hytte"));
                              tmp.setDatoFra(rs.getDate("DatoFra").toString());
                              tmp.setDatoTil(rs.getDate("DatoTil").toString());
                              tmp.setSum(rs.getInt("Sum"));
                              tmp.setBetalt(rs.getBoolean("Betalt"));
                              tmp.setBekreftet(rs.getBoolean("Bekreftet"));
                              tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
                              tmp.setAutKode(rs.getString("AutKode"));
                              tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
                              bestillinger.add(tmp);
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
                  return bestillinger;
            }
            db.kobleFra();
            return null;
      }

/*****************************************************************
 * Genererer en rapport for ferier
 * @param id Hyttenavn evt Alle
 * @param ferie Navn paa ferie
 * @param aar Aar det skal genereres rapport paa
 * @param tildelt generere for tildelte
 * @param betalt generere for betalte
 * @return ArrayList<Bestilling> Returnerer arraylist av bestillinger
 *****************************************************************/
public ArrayList<Bestilling> genererRapportFerie(String id, String ferie, int
aar, boolean tildelt, boolean betalt) {
      // Dersom alle hyttene
      if(id.equals("Alle")){
            if(betalt){
                  if(tildelt){
```

```
                        db.settSql("SELECT * FROM BESTILLING WHERE Bekreftet=1
AND Betalt=1 AND BekreftetTildelt=1 AND" +
                                    " DatoFra > '" + aar + "-01-01' AND DatoFra
< '" + aar + "-12-31'");
                  }else{
                        db.settSql("SELECT * FROM BESTILLING WHERE Bekreftet=1
AND Betalt=1 AND" +
                                    " DatoFra > '" + aar + "-01-01' AND DatoFra
< '" + aar + "-12-31'");
                  }
            }else{
                  if(tildelt){
                        db.settSql("SELECT * FROM BESTILLING WHERE Bekreftet=1
AND Betalt=0 AND BekreftetTildelt=1 AND" +
                                    " DatoFra > '" + aar + "-01-01' AND DatoFra
< '" + aar + "-12-31'");
                  }else{
                        db.settSql("SELECT * FROM BESTILLING WHERE Betalt=0 AND
Bekreftet=1 AND" +
                                    " DatoFra > '" + aar + "-01-01' AND DatoFra
< '" + aar + "-12-31'");
                  }
            }
      // Dersom enkelthytte
      }else{
            if(betalt){
                  if(tildelt){
                        db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Bekreftet=1 AND Betalt=1 AND BekreftetTildelt=1 AND" +
                                    " DatoFra > '" + aar + "-01-01' AND DatoFra
< '" + aar + "-12-31'");
                  }else{
                        db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Bekreftet=1 AND Betalt=1 AND" +
                                    " DatoFra > '" + aar + "-01-01' AND DatoFra
< '" + aar + "-12-31'");
                  }
            }else{
                  if(tildelt){
                        db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Betalt=0 AND Bekreftet=1 AND BekreftetTildelt=1 AND" +
                                    " DatoFra > '" + aar + "-01-01' AND DatoFra
< '" + aar + "-12-31'");
                  }else{
                        db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Betalt=0 AND Bekreftet=1 AND" +
                                    " DatoFra > '" + aar + "-01-01' AND DatoFra
< '" + aar + "-12-31'");
                  }
            }
      }

            if(db.kobleTil(true)){
                  ArrayList<Bestilling> bestillinger = new ArrayList<Bestilling>
();
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              Date fra = rs.getDate("DatoFra");
                              String fNavn = fDAO.hentFerie(fra);
                              if(!fNavn.isEmpty()){
                                    if(ferie.equals("Alle")){
                                          Bestilling tmp = new Bestilling();
                                          tmp.setNummer(rs.getInt("Nummer"));
                                          tmp.setKunde(rs.getString("Kunde"));
```

```
                                        tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                                        tmp.setHytte(rs.getString("Hytte"));
                                        tmp.setDatoFra(rs.getDate
("DatoFra").toString());
                                        tmp.setDatoTil(rs.getDate
("DatoTil").toString());
                                        tmp.setSum(rs.getInt("Sum"));
                                        tmp.setBetalt(rs.getBoolean
("Betalt"));
                                        tmp.setBekreftet(rs.getBoolean
("Bekreftet"));
                                        tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
                                        tmp.setAutKode(rs.getString
("AutKode"));
                                        tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
                                        tmp.setFerie
(fNavn);
                                        bestillinger.add(tmp);
                                  }else{
                                        if(ferie.equals(fNavn)){
                                              Bestilling tmp = new Bestilling
();
                                              tmp.setNummer(rs.getInt
("Nummer"));
                                              tmp.setKunde(rs.getString
("Kunde"));
                                              tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                                              tmp.setHytte(rs.getString
("Hytte"));
                                              tmp.setDatoFra(rs.getDate
("DatoFra").toString());
                                              tmp.setDatoTil(rs.getDate
("DatoTil").toString());
                                              tmp.setSum(rs.getInt("Sum"));
                                              tmp.setBetalt(rs.getBoolean
("Betalt"));
                                              tmp.setBekreftet(rs.getBoolean
("Bekreftet"));
                                              tmp.setBekreftetTildelt
(rs.getBoolean("BekreftetTildelt"));
                                              tmp.setAutKode(rs.getString
("AutKode"));
                                              tmp.setAutKodeTildelt
(rs.getString("AutKodeTildelt"));
                                              tmp.setFerie
(fNavn);
                                              bestillinger.add(tmp);
                                        }
                                  }
                            }

                      }
                } catch (SQLException e) {
                      e.printStackTrace();
                }
                db.kobleFra();
                return bestillinger;
          }
          db.kobleFra();
          return null;
     }
```

```
/**********************************************************************
 * Genererer en rapport frem i tid
 * @param id Hyttenavn evt Alle
 * @param tildelt generere for tildelte
 * @param betalt generere for betalte
 * @return ArrayList<Bestilling> Returnerer arraylist av bestillinger
 **********************************************************************/
public ArrayList<Bestilling> genererRapportFrem(String id, boolean tildelt,
boolean betalt) {
	String dato = HjelpeKlasserDato.dagensDato();
	// Dersom alle hyttene
	if(id.equals("Alle")){
		if(betalt){
			if(tildelt){
				db.settSql("SELECT * FROM BESTILLING WHERE Bekreftet=1
AND Betalt=1 AND BekreftetTildelt=1 AND" +
					" DatoFra > '" + dato + "'");
			}else{
				db.settSql("SELECT * FROM BESTILLING WHERE Bekreftet=1
AND Betalt=1 AND" +
					" DatoFra > '" + dato + "'");
			}
		}else{
			if(tildelt){
				db.settSql("SELECT * FROM BESTILLING WHERE Bekreftet=1
AND BekreftetTildelt=1 AND" +
					" DatoFra > '" + dato + "'");
			}else{
				db.settSql("SELECT * FROM BESTILLING WHERE Bekreftet=1
AND" +
					" DatoFra > '" + dato + "'");
			}
		}
	// Dersom enkelthytte
	}else{
		if(betalt){
			if(tildelt){
				db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Bekreftet=1 AND Betalt=1 AND BekreftetTildelt=1 AND" +
					" DatoFra > '" + dato + "'");
			}else{
				db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Bekreftet=1 AND Betalt=1 AND" +
					" DatoFra > '" + dato + "'");
			}
		}else{
			if(tildelt){
				db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Bekreftet=1 AND BekreftetTildelt=1 AND" +
					" DatoFra > '" + dato + "'");
			}else{
				db.settSql("SELECT * FROM BESTILLING WHERE Hytte='"+ id
+ "' AND Bekreftet=1 AND" +
					" DatoFra > '" + dato + "'");
			}
		}
	}

		if(db.kobleTil(true)){
			ArrayList<Bestilling> bestillinger = new ArrayList<Bestilling>
();
			ResultSet rs = db.hentResultat();
			try {
				while(rs.next()) {
					Bestilling tmp = new Bestilling();
```

```
                              tmp.setNummer(rs.getInt("Nummer"));
                              tmp.setKunde(rs.getString("Kunde"));
                              tmp.setBestDato(rs.getTimestamp
("BesDato").toString());
                              tmp.setHytte(rs.getString("Hytte"));
                              tmp.setDatoFra(rs.getDate("DatoFra").toString());
                              tmp.setDatoTil(rs.getDate("DatoTil").toString());
                              tmp.setSum(rs.getInt("Sum"));
                              tmp.setBetalt(rs.getBoolean("Betalt"));
                              tmp.setBekreftet(rs.getBoolean("Bekreftet"));
                              tmp.setBekreftetTildelt(rs.getBoolean
("BekreftetTildelt"));
                              tmp.setAutKode(rs.getString("AutKode"));
                              tmp.setAutKodeTildelt(rs.getString
("AutKodeTildelt"));
                              bestillinger.add(tmp);
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
                  return bestillinger;
            }
            db.kobleFra();
            return null;
      }

/*********************************************************************
 * Metode som finner om det skal kjores trekning for ferier
 *********************************************************************/
      public void kjorTrekningFerie(){
            // Trekningstabell av bestillinger
            ArrayList<Bestilling> trekning = new ArrayList<Bestilling>();
            // Henter ut navn paa hyttene
            ArrayList<String> hyttenavn = hDAO.hentHytteNavn();
            // Henter ut tildelte bestillinger dette året
            ArrayList<Bestilling> bestillinger = hentAlleBekreftet(2010);
            // Henter ut ferier
            ArrayList<Ferie> ferier = fDAO.hentFerierUtenPerioder(2010);

            String tmpPer;
            String periodeId = null;
            if(hyttenavn != null && bestillinger != null && ferier !=null){
                  for(int k = 0; k < ferier.size(); k++){
                        if(ferier.get(k).getAvbFrist().toString().equals
(HjelpeKlasserDato.dagensDato())){
                              for(int i = 0; i < hyttenavn.size(); i++){
                                    for(int j = 0; j < bestillinger.size(); j+
+){
                                          if(hyttenavn.get(i).equals
(bestillinger.get(j).getHytte())){
                                                tmpPer = fDAO.tilhorerFerie
(ferier.get(k), bestillinger.get(j).getDatoFra());
                                                if(tmpPer != null){
                                                      trekning.add
(bestillinger.get(j));
                                                      periodeId = tmpPer;
                                                }
                                          }
                                    }
                                    // Dersom det er en eller flere
bestillinger i en trekning
                                    if(trekning.size() > 0){
                                          leggTilTrekningFerie(trekning,
hyttenavn.get(i), periodeId);
                                    }
```

```
                        }

                    }
                }
            }
      }

      /**********************************************************************
       * Lager logikk for arraylist med bestillinger som skal trekkes og legges
i db
       * @param trekning Arraylist av trekninger
       * @param hytte Hyttenavn evt Alle
       * @param periode Periode id til aktuell ferie
       * @return boolean Returnerer om trekningen gikk igjennom
       **********************************************************************/
      public boolean leggTilTrekningFerie(ArrayList<Bestilling> trekning, String
hytte, String periode){
            int nNummer = nesteTrekningsNummer();
            trekning = Utility.randomizeFerie(trekning);
            int pri = 0;
            for(int i = 0; i < trekning.size(); i++){
                  if(erIKarantene(hytte, trekning.get(i).getKunde(), "Ferie")){
                        pri = 3;
                  }else pri = 1;
                  db.settSql("INSERT INTO TREKNING (Nummer, Bestilling, Hytte,
Prioritet, KoNummer, Type)" +
                              " VALUES("+nNummer +", " + trekning.get
(i).getNummer() + ", '" + hytte + "', "+pri+", " + i + ", '"
                              + periode + "')");
                  if(db.kobleTil(false)){

                  }
            }
            db.kobleFra();
            sendTildelEpost(trekning.get(0));
            return true;
      }

      /**********************************************************************
       * Metode som finner om det skal kjores trekning for dager og perioder
       **********************************************************************/
      public void kjorTrekningDager(int fristDag, int fristHelg, boolean brukDb)
{

            int avbFristDag;
            int avbFristHelg;
            int avbFristManTor;

            if(brukDb){
                  avbFristDag = iDAO.hentAvbestillingDag()*7;
                  avbFristHelg = iDAO.hentAvbestillingPeriode()*7;
                  avbFristManTor = avbFristHelg;
            } else {
                  avbFristDag = fristDag;
                  avbFristHelg = fristHelg;
                  avbFristManTor = avbFristHelg;
            }

             // Trekningstabell av bestillinger
             ArrayList<Bestilling> trekningHelg = new ArrayList<Bestilling>();
             // Trekningstabell av bestillinger
             ArrayList<Bestilling> trekningManTor = new ArrayList<Bestilling>();
             // Trekningstabell av bestillinger
             ArrayList<Bestilling> trekningDag = new ArrayList<Bestilling>();
             // Henter ut navn paa hyttene
             ArrayList<String> hyttenavn = hDAO.hentHytteNavn();
```

```
            // Henter ut tildelte bestillinger dette året

            // FIXME
            ArrayList<Bestilling> bestillinger = hentAlleBekreftet(2010);

            if(hyttenavn != null && bestillinger != null){
                  for(int i = 0; i < hyttenavn.size(); i++){
                        for(int j = 0; j < bestillinger.size(); j++){
                              if(hyttenavn.get(i).equals(bestillinger.get
(j).getHytte())){
                                    // Funnet at bestilling er enkelt dag
                                     if(HjelpeKlasserDato.erEnkeltDag
(bestillinger.get(j))){
                                          if
(HjelpeKlasserDato.erAvbestillingsFrist(bestillinger.get(j).getDatoFra(),
avbFristDag)){
                                                trekningDag.add
(bestillinger.get(j));
                                          }
                                     }
                                    // Funnet at bestilling er enkelt helg
                                     if(HjelpeKlasserDato.erHelg
(bestillinger.get(j))){
                                          if
(HjelpeKlasserDato.erAvbestillingsFrist(bestillinger.get(j).getDatoFra(),
avbFristHelg)){
                                                trekningHelg.add
(bestillinger.get(j));
                                          }
                                     }
                                    // Funnet at bestilling er enkelt man - tor
                                     if(HjelpeKlasserDato.erManTor
(bestillinger.get(j))){

                                          if
(HjelpeKlasserDato.erAvbestillingsFrist(bestillinger.get(j).getDatoFra(),
avbFristManTor)){
                                                trekningManTor.add
(bestillinger.get(j));
                                          }
                                     }
                              }

                        }
                        // Dersom det er en eller flere bestillinger i en
trekning
                        if(trekningDag.size() > 0 || trekningHelg.size() > 0 ||
trekningManTor.size() > 0){
                              leggTilTrekningDager(trekningDag, trekningHelg,
trekningManTor, hyttenavn.get(i));
                              trekningDag.clear();
                              trekningHelg.clear();
                              trekningManTor.clear();
                        }
                  }
            }
      }

      /**********************************************************************
       * Lager logikk for arraylist med bestillinger som skal trekkes og legges
i db
       * @param tDag Arraylist av trekninger for enkeltdager
       * @param tHelg Arraylist av trekninger for helg
       * @param tManTor Arraylist av trekninger for man-tor
       * @param hytte Hyttenavn evt Alle
       * @return boolean Returnerer om trekningen gikk igjennom
```

```
      ***************************************************************/
      public boolean leggTilTrekningDager(ArrayList<Bestilling> tDag,
                  ArrayList<Bestilling> tHelg, ArrayList<Bestilling> tManTor,
String hytte){
            // Legger til dersom det foreligger trekninger på dagene
            if(tDag.size()>0) tDag = leggTilEksisterendePeriode(tDag);
            if(tHelg.size()>0) tHelg = leggTilEksisterendeDager(tHelg, "Helg");
            if(tManTor.size()>0) tManTor = leggTilEksisterendeDager(tManTor,
"ManTor");
            // Neste trekningsnummer
            int nNummer = 0;
            int pri = 0;
            // Dersom det kun er enkeltdager det skal foretas trekning paa
            if(tDag.size()>0 && tHelg.size() == 0 && tManTor.size() == 0){
                  nNummer = nesteTrekningsNummer();

                  tDag = Utility.randomize(tDag);
                  for(int i = 0; i < tDag.size();i++){
                        if(erIKarantene(hytte, tDag.get(i).getKunde(),
"Periode")){
                              pri = 3;
                        }else pri = 2;

                        db.settSql("INSERT INTO TREKNING (Nummer, Bestilling,
Hytte, Prioritet, KoNummer, Type)" +
                                    " VALUES("+nNummer +", " + tDag.get
(i).getNummer() + ", '" + hytte + "', " + pri + ", " + i + ", 'Dag')");
                        if(db.kobleTil(false)){

                        }
                  }
                  db.kobleFra();
                  sendTildelEpost(tDag.get(0));

                  return true;
            }
            // Dersom det er helg og/eller mandag til torsdag det skal foretas
trekning paa
            if(tDag.size() == 0 && (tHelg.size() > 0 || tManTor.size() > 0)){
                  tHelg = Utility.randomize(tHelg);
                  nNummer = nesteTrekningsNummer();
                  for(int i = 0; i < tHelg.size();i++){
                        if(erIKarantene(hytte, tHelg.get(i).getKunde(),
"Periode")){
                              pri = 3;
                        }else pri = 1;
                        db.settSql("INSERT INTO TREKNING (Nummer, Bestilling,
Hytte, Prioritet, KoNummer, Type)" +
                                    " VALUES("+nNummer +", " + tHelg.get
(i).getNummer() + ", '" + hytte + "', "+pri+", " + i + ", 'Helg')");
                        if(db.kobleTil(false)){

                        }
                  }
                  if(tHelg.size()>0) sendTildelEpost(tHelg.get(0));

                  tManTor = Utility.randomize(tManTor);
                  nNummer = nesteTrekningsNummer();
                  for(int i = 0; i < tManTor.size();i++){
                        if(erIKarantene(hytte, tManTor.get(i).getKunde(),
"Periode")){
                              pri = 3;
                        }else pri = 1;
                        db.settSql("INSERT INTO TREKNING (Nummer, Bestilling,
Hytte, Prioritet, KoNummer, Type)" +
```

```
                                    " VALUES("+nNummer +", " + tManTor.get
(i).getNummer() + ", '" + hytte + "', "+pri+", " + i + ", 'ManTor')");
                        if(db.kobleTil(false)){

                        }
                  }
                  if(tManTor.size()>0) sendTildelEpost(tManTor.get(0));
                  db.kobleFra();
                  return true;
            }

            // Dersom det kun er helg og enkeltdag det skal foretas trekning paa
            if(tDag.size() > 0 && tHelg.size() > 0 && tManTor.size() == 0){
                  tDag = Utility.randomize(tDag);
                  tHelg = Utility.randomize(tHelg);

                  if(HjelpeKlasserDato.erInnenforHelg(tDag.get(0), tHelg.get
(0))){
                        int teller = 0;
                        nNummer = nesteTrekningsNummer();
                        for(int i = 0; i < tHelg.size();i++){
                              if(erIKarantene(hytte, tDag.get(i).getKunde(),
"Periode")){
                                    pri = 3;
                              }else pri = 1;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tHelg.get
(i).getNummer() + ", '" + hytte + "', "+pri+", "
                                          + i + ", 'Helg')");
                              if(db.kobleTil(false)){

                              }
                              teller = i;
                        }
                        sendTildelEpost(tHelg.get(0));

                        for(int j = teller; j < tDag.size() + tHelg.size()-1;j+
+){
                              if(erIKarantene(hytte, tDag.get(j-teller).getKunde
(), "Periode")){
                                    pri = 3;
                              }else pri = 2;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tDag.get(j-
teller).getNummer() + ", '" + hytte + "', "+pri+ ", " + (j+1) + ", 'Dag')");
                              if(db.kobleTil(false)){

                              }
                        }
                  }else{
                        nNummer = nesteTrekningsNummer();
                        for(int i = 0; i < tHelg.size();i++){
                              if(erIKarantene(hytte, tDag.get(i).getKunde(),
"Periode")){
                                    pri = 3;
                              }else pri = 1;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tHelg.get
(i).getNummer() + ", '" + hytte + "', "+pri+", " + i + ", 'Helg')");
                              if(db.kobleTil(false)){

                              }
                        }
```

```
                        sendTildelEpost(tHelg.get(0));
                        nNummer = nesteTrekningsNummer();
                        for(int j = 0; j < tDag.size(); j++){
                              if(erIKarantene(hytte, tDag.get(j).getKunde(),
"Periode")){
                                    pri = 3;
                              }else pri = 2;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tDag.get
(j).getNummer() + ", '" + hytte + "', "+pri+", " + j + ", 'Dag')");
                              if(db.kobleTil(false)){

                              }
                        }
                        sendTildelEpost(tDag.get(0));
                  }

                  db.kobleFra();
                  return true;
            }

            // Dersom det kun er man-tor og enkeltdag det skal foretas trekning
paa
            if(tDag.size()>0 && tHelg.size() == 0 && tManTor.size() > 0){
                  tDag = Utility.randomize(tDag);
                  tManTor = Utility.randomize(tManTor);
                  if(HjelpeKlasserDato.erInnenforManTor(tDag.get(0), tManTor.get
(0))){
                        int teller = 0;
                        nNummer = nesteTrekningsNummer();
                        for(int i = 0; i < tManTor.size();i++){
                              if(erIKarantene(hytte, tManTor.get(i).getKunde(),
"Periode")){
                                    pri = 3;
                              }else pri = 1;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tManTor.get
(i).getNummer() + ", '" + hytte + "', "+pri+", " + i + ", 'ManTor')");
                              if(db.kobleTil(false)){

                              }
                              teller = i;
                        }
                        sendTildelEpost(tManTor.get(0));
                        for(int j = teller; j < tDag.size() + tManTor.size()-1;j
++){
                              if(erIKarantene(hytte, tDag.get(j-teller).getKunde
(), "Periode")){
                                    pri = 3;
                              }else pri = 2;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tDag.get(j-
teller).getNummer() + ", '" + hytte + "', "+pri+", " + (j+1) + ", 'Dag')");
                              if(db.kobleTil(false)){

                              }
                        }
                  }else{
                        nNummer = nesteTrekningsNummer();
                        for(int i = 0; i < tManTor.size();i++){
                              if(erIKarantene(hytte, tManTor.get(i).getKunde(),
"Periode")){
                                    pri = 3;
```

```
                              }else pri = 1;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tManTor.get
(i).getNummer() + ", '" + hytte + "', "+pri+", " + i + ", 'ManTor')");
                              if(db.kobleTil(false)){

                              }
                        }
                        sendTildelEpost(tManTor.get(0));
                        nNummer = nesteTrekningsNummer();
                        for(int j = 0; j < tDag.size(); j++){
                              if(erIKarantene(hytte, tDag.get(j).getKunde(),
"Periode")){
                                    pri = 3;
                              }else pri = 2;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tDag.get
(j).getNummer() + ", '" + hytte + "', "+pri+", " + j + ", 'Dag')");
                              if(db.kobleTil(false)){

                              }
                        }
                        sendTildelEpost(tDag.get(0));
                  }

                  db.kobleFra();
                  return true;
            }

            // Dersom det foretas trekning paa alle alternativ
            if(tDag.size()>0 && tHelg.size() > 0 && tManTor.size() > 0){
                  tDag = Utility.randomize(tDag);
                  tHelg = Utility.randomize(tHelg);
                  tManTor = Utility.randomize(tManTor);

                  if(HjelpeKlasserDato.erInnenforHelg(tDag.get(0), tHelg.get
(0))){
                        int teller = 0;
                        nNummer = nesteTrekningsNummer();
                        for(int i = 0; i < tHelg.size();i++){
                              if(erIKarantene(hytte, tHelg.get(i).getKunde(),
"Periode")){
                                    pri = 3;
                              }else pri = 1;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tHelg.get
(i).getNummer() + ", '" + hytte + "', "+pri+", " + i + ", 'Helg')");
                              if(db.kobleTil(false)){

                              }
                              teller = i;
                        }
                        sendTildelEpost(tHelg.get(0));
                        for(int j = teller; j < tDag.size() + tHelg.size()-1;j+
+){
                              if(erIKarantene(hytte, tDag.get(j-teller).getKunde
(), "Periode")){
                                    pri = 3;
                              }else pri = 2;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tDag.get(j-
teller).getNummer() + ", '" + hytte + "', "+pri+", " + (j+1) + ", 'Dag')");
```

```
                              if(db.kobleTil(false)){

                              }
                        }
                        nNummer = nesteTrekningsNummer();
                        for(int n = 0; n < tManTor.size();n++){
                              if(erIKarantene(hytte, tManTor.get(n).getKunde(),
"Periode")){
                                    pri = 3;
                              }else pri = 1;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tManTor.get
(n).getNummer() + ", '" + hytte + "', "+pri+", " + n + ", 'ManTor')");
                              if(db.kobleTil(false)){

                              }
                        }
                        sendTildelEpost(tManTor.get(0));
                  }else if(HjelpeKlasserDato.erInnenforManTor(tDag.get(0),
tManTor.get(0))){
                        int teller = 0;
                        nNummer = nesteTrekningsNummer();
                        for(int i = 0; i < tManTor.size();i++){
                              if(erIKarantene(hytte, tManTor.get(i).getKunde(),
"Periode")){
                                    pri = 3;
                              }else pri = 1;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tManTor.get
(i).getNummer() + ", '" + hytte + "', "+pri+", " + i + ", 'ManTor')");
                              if(db.kobleTil(false)){

                              }
                              teller = i;
                        }
                        sendTildelEpost(tManTor.get(0));
                        for(int j = teller; j < tDag.size() + tManTor.size()-1;j
++){
                              if(erIKarantene(hytte, tDag.get(j-teller).getKunde
(), "Periode")){
                                    pri = 3;
                              }else pri = 2;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tDag.get(j-
teller).getNummer() + ", '" + hytte + "', "+pri+", " + (j+1) + ", 'Dag')");
                              if(db.kobleTil(false)){

                              }
                        }
                        nNummer = nesteTrekningsNummer();
                        for(int k = 0; k < tHelg.size();k++){
                              if(erIKarantene(hytte, tHelg.get(k).getKunde(),
"Periode")){
                                    pri = 3;
                              }else pri = 1;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tHelg.get
(k).getNummer() + ", '" + hytte + "', "+pri+", " + k + ", 'Helg')");
                              if(db.kobleTil(false)){

                              }
                        }
```

```
                        sendTildelEpost(tHelg.get(0));
                  }else{
                        nNummer = nesteTrekningsNummer();
                        for(int k = 0; k < tHelg.size();k++){
                              if(erIKarantene(hytte, tHelg.get(k).getKunde(),
"Periode")){
                                    pri = 3;
                              }else pri = 1;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tHelg.get
(k).getNummer() + ", '" + hytte + "', "+pri+ ", " + k + ", 'Helg')");
                              if(db.kobleTil(false)){

                              }
                        }
                        sendTildelEpost(tHelg.get(0));
                        nNummer = nesteTrekningsNummer();
                        for(int i = 0; i < tManTor.size();i++){
                              if(erIKarantene(hytte, tManTor.get(i).getKunde(),
"Periode")){
                                    pri = 3;
                              }else pri = 1;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tManTor.get
(i).getNummer() + ", '" + hytte + "', "+pri+", " + i + ", 'ManTor')");
                              if(db.kobleTil(false)){

                              }
                        }
                        sendTildelEpost(tManTor.get(0));
                        nNummer = nesteTrekningsNummer();
                        for(int j = 0; j < tDag.size(); j++){
                              if(erIKarantene(hytte, tDag.get(j).getKunde(),
"Periode")){
                                    pri = 3;
                              }else pri = 2;
                              db.settSql("INSERT INTO TREKNING (Nummer,
Bestilling, Hytte, Prioritet, KoNummer, Type)" +
                                          " VALUES("+nNummer +", " + tDag.get
(j).getNummer() + ", '" + hytte + "', "+pri+", " + j + ", 'Dag')");
                              if(db.kobleTil(false)){

                              }
                        }
                        sendTildelEpost(tDag.get(0));
                  }
            }
            db.kobleFra();
            return false;
      }

      /**********************************************************************
       * Metode som finner neste trekningsnummer
       * @return int Returnerer neste trekningsnummer
       **********************************************************************/
      public int nesteTrekningsNummer(){
            db.settSql("SELECT MAX( Nummer ) as neste FROM TREKNING");

            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              return rs.getInt("neste") + 1;
                        }
```

```
                } catch (SQLException e) {
                      e.printStackTrace();
                }
                db.kobleFra();
          }
          db.kobleFra();
          return -1;
    }

    /**********************************************************************
     * Metode som legger til bestilling i eksisterende trekning
     * @param best Aktuell bestilling
     * @return boolean Returnere om bestillingen gikk gjennom
     **********************************************************************/
    public boolean leggTilBestillingTrekning(Bestilling best){
          boolean eksisterer = false;
          int trekningsNummer = 0;
          int nesteKo = 0;
          String type = fDAO.tilhorerFerie(best);
          if(type == null){
                if(HjelpeKlasserDato.erEnkeltDag(best)){
                      type = "Dag";
                }else if(HjelpeKlasserDato.erHelg(best)){
                      type = "Helg";
                }else if(HjelpeKlasserDato.erManTor(best)){
                      type = "ManTor";
                }
          }

          db.settSql("SELECT * FROM TREKNING WHERE Hytte='" + best.getHytte()
+ "'");
          if(db.kobleTil(true)){
                ResultSet rs = db.hentResultat();
                try {
                      while(rs.next()) {
                            if(HjelpeKlasserDato.harDagISeg(best,
hentBestilling(rs.getInt("Bestilling")))
                                        || HjelpeKlasserDato.erInnenforHelg
(best, hentBestilling(rs.getInt("Bestilling")))
                                        || HjelpeKlasserDato.erInnenforManTor
(best, hentBestilling(rs.getInt("Bestilling")))
                                        || HjelpeKlasserDato.finnDifferanse
(hentBestilling(rs.getInt("Bestilling")).getDatoFra(), best.getDatoFra()) == 0){
                                  eksisterer = true;
                                  trekningsNummer = rs.getInt("Nummer");
                                  nesteKo = finnNesteKoNr(trekningsNummer);
                            }
                      }
                } catch (SQLException e) {
                      e.printStackTrace();
                }
          }
          if(eksisterer){
                int pri = 0;
                if(type.equals("Dag")){
                      pri = 2;
                }else{
                      pri = 1;
                }
                db.settSql("INSERT INTO TREKNING (Nummer, Bestilling, Hytte,
Prioritet, KoNummer, Type)" +
                            " VALUES("+ trekningsNummer +", " + best.getNummer
() + ", '" + best.getHytte() + "', "+pri+", " + nesteKo + ", '"+type+"')");
                if(db.kobleTil(false)){
                      return true;
                }
```

```
                db.kobleFra();
            }
            db.kobleFra();
            return false;
      }

      public ArrayList<Bestilling> leggTilEksisterendePeriode
(ArrayList<Bestilling> best){
            boolean eksisterer = false;
            int trekningsNummer = 0;
            int nesteKo = 0;
            db.settSql("SELECT * FROM TREKNING WHERE Hytte='" + best.get
(0).getHytte() + "' AND (Type='ManTor' OR Type='Helg') GROUP BY Nummer");
            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              if(HjelpeKlasserDato.erInnenforHelg(best.get(0),
hentBestilling(rs.getInt("Bestilling")))
                                          || HjelpeKlasserDato.erInnenforManTor
(best.get(0), hentBestilling(rs.getInt("Bestilling")))){
                                    eksisterer = true;
                                    trekningsNummer = rs.getInt("Nummer");
                                    nesteKo = finnNesteKoNr(trekningsNummer);
                              }
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            }
            if(eksisterer){
                  best = Utility.randomize(best);

                  for(int j = 0; j < best.size(); j++){
                        db.settSql("INSERT INTO TREKNING (Nummer, Bestilling,
Hytte, Prioritet, KoNummer, Type)" +
                                    " VALUES("+ trekningsNummer +", " + best.get
(j).getNummer() + ", '" + best.get(j).getHytte() + "', 2, " + nesteKo + ",
'Dag')");
                        if(db.kobleTil(false)){
                              nesteKo++;
                        }
                  }
                  best.clear();
                  db.kobleFra();
            }
            db.kobleFra();
            return best;
      }

      public ArrayList<Bestilling> leggTilEksisterendeDager
(ArrayList<Bestilling> best, String type){
            System.out.println("askd: " + best.get(0).getNummer());
            boolean eksisterer = false;
            int trekningsNummer = 0;
            int nesteKo = 0;
            db.settSql("SELECT * FROM TREKNING WHERE Hytte='" + best.get
(0).getHytte() + "' AND Type='Dag' GROUP BY Nummer");
            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              if(HjelpeKlasserDato.erInnenforHelg(hentBestilling
(rs.getInt("Bestilling")), best.get(0))
```

```
                                          || HjelpeKlasserDato.erInnenforManTor
(hentBestilling(rs.getInt("Bestilling")), best.get(0))){
                                    eksisterer = true;
                                    trekningsNummer = rs.getInt("Nummer");
                                    nesteKo = finnNesteKoNr(trekningsNummer);
                              }else if(HjelpeKlasserDato.erEnkeltDag(best.get
(0))){
                                    if(HjelpeKlasserDato.finnDifferanse
(hentBestilling(rs.getInt("Bestilling")).getDatoFra(), best.get(0).getDatoFra())
== 0){
                                          eksisterer = true;
                                          trekningsNummer = rs.getInt("Nummer");
                                          nesteKo = finnNesteKoNr
(trekningsNummer);
                                    }
                              }
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            }
            if(eksisterer){
                  best = Utility.randomize(best);

                  for(int j = 0; j < best.size(); j++){
                        db.settSql("INSERT INTO TREKNING (Nummer, Bestilling,
Hytte, Prioritet, KoNummer, Type)" +
                                    " VALUES("+ trekningsNummer +", " + best.get
(j).getNummer() + ", '" + best.get(j).getHytte() + "', 1, " + nesteKo + ", '"+
type +"')");
                        if(db.kobleTil(false)){
                              nesteKo++;
                        }
                  }
                  best.clear();
                  db.kobleFra();
            }
            db.kobleFra();
            return best;
      }

      /**********************************************************************
       * Metode som finner neste ko nummer
       * @param id Id til aktuell trekning
       * @return int Returnerer neste konummer
       **********************************************************************/
      public int finnNesteKoNr(int id){
            db.settSql("SELECT MAX( KoNummer ) as neste FROM TREKNING WHERE
Nummer=" + id);

            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              return rs.getInt("neste") + 1;
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            }
            db.kobleFra();
            return -1;
      }
```

```
	/**********************************************************************
	 * Metode som finner ut om en person er i karantene for gitt type
	 * @param hytte Aktuell hytte
	 * @param kunde Aktuell kunde
	 * @param type Hvilken type det gjelder, feks. Dag, Helg..
	 * @return boolean Returnere om personen er i karantene
	 **********************************************************************/
	public boolean erIKarantene(String hytte, String kunde, String type){
		db.settSql("SELECT * FROM KARANTENE WHERE Hytte='" + hytte + "' AND
Kunde='" + kunde
				+ "' AND Type='" + type + "'");
		if(db.kobleTil(true)){
			ResultSet rs = db.hentResultat();
			try {
				while(rs.next()) {
					return true;
				}
			} catch (SQLException e) {
				e.printStackTrace();
			}
			db.kobleFra();
		}
		db.kobleFra();
		return false;
	}

	/**********************************************************************
	 * Metode som setter karantene
	 * @param hytte Aktuell hytte
	 * @param kunde Aktuell kunde
	 * @param type Hvilken type det gjelder, feks. Dag, Helg..
	 * @param dato Dato ifra
	 **********************************************************************/
	public void settKarantene(String hytte, String kunde, String type, String
dato){

		KalenderElement date = new KalenderElement(dato);
		if(type.equals("Ferie")){
			int karanteneFerie = iDAO.hentKaranteneFerie();
			date = KalenderElement.flyttKalemenderElement(date,
karanteneFerie*12);
		}else{
			int karantenePeriode = iDAO.hentKarantenePeriode();
			date = KalenderElement.flyttKalemenderElementDager(date,
karantenePeriode*7);
		}

		db.settSql("INSERT INTO KARANTENE (Hytte, Sluttdato, Type, Kunde)
VALUES('"
				+ hytte + "', '" + date.toString() +  "', '" + type +
"', '" + kunde + "')");
		if(db.kobleTil(false)){
			db.kobleFra();
		}
		db.kobleFra();
	}

	/**********************************************************************
	 * Metode som sender tildel epost
	 * @param best Aktuell bestilling
	 **********************************************************************/
	public void sendTildelEpost(Bestilling best){
		// Sender ut epost ved tildeling

		EpostDAOImpl epost = new EpostDAOImpl();
		epost.sendTildeltBekreft(best);
```

```
	}

	/**********************************************************************
	 * Metode som returnerer om en bestilling har trekning
	 * @param best Aktuell bestilling
	 * @return boolean Returnerer om bestilling har trekning
	 **********************************************************************/
	public boolean harTrekning(Bestilling best){

		db.settSql("SELECT Bestilling FROM TREKNING WHERE Hytte='" +
best.getHytte() + "'");

		if(db.kobleTil(true)){
			ResultSet rs = db.hentResultat();
			try {
				while(rs.next()) {
					if(HjelpeKlasserDato.harDagISeg(best,
hentBestilling(rs.getInt("Bestilling")))
						|| HjelpeKlasserDato.erInnenforHelg(best,
hentBestilling(rs.getInt("Bestilling")))
						|| HjelpeKlasserDato.erInnenforManTor(best,
hentBestilling(rs.getInt("Bestilling")))
						|| HjelpeKlasserDato.finnDifferanse
(hentBestilling(rs.getInt("Bestilling")).getDatoFra(), best.getDatoFra()) == 0){
						return true;
					}
				}
			} catch (SQLException e) {
				e.printStackTrace();
			}
			db.kobleFra();
		}
		db.kobleFra();
		return false;
	}

	/**********************************************************************
	 * Metode som returnerer om en bestilling har trekning
	 * @param best Aktuell bestilling
	 * @return String Returnerer hvilken type trekning bestillingen har
	 **********************************************************************/
	public String harTrekningStreng(Bestilling best){

		db.settSql("SELECT Bestilling FROM TREKNING WHERE Hytte='" +
best.getHytte() + "'");

		if(db.kobleTil(true)){
			ResultSet rs = db.hentResultat();
			try {
				while(rs.next()) {
					if(HjelpeKlasserDato.harDagISeg(best,
hentBestilling(rs.getInt("Bestilling")))
						|| HjelpeKlasserDato.erInnenforHelg(best,
hentBestilling(rs.getInt("Bestilling")))
						|| HjelpeKlasserDato.erInnenforManTor(best,
hentBestilling(rs.getInt("Bestilling")))
						|| HjelpeKlasserDato.finnDifferanse
(hentBestilling(rs.getInt("Bestilling")).getDatoFra(), best.getDatoFra()) == 0){
						if(HjelpeKlasserDato.erEnkeltDag(best))
return "Dag";
						if(HjelpeKlasserDato.erHelg(best)) return
"Helg";
						if(HjelpeKlasserDato.erManTor(best)) return
"ManTor";
						return "Ferie";
					}
```

```
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            }
            db.kobleFra();
            return "";
      }

      /**********************************************************************
       * Metode som lager en ny trekning for dager og perioder
       * @param best Aktuell bestilling
       * @param karantene Om kunden er i karantene
       **********************************************************************/
      public void nyTrekningDagerPerioder(Bestilling best, boolean karantene){
            int nNummer = nesteTrekningsNummer();
            String type = fDAO.tilhorerFerie(best);
            if(type == null){
                  if(HjelpeKlasserDato.erEnkeltDag(best)){
                        type = "Dag";
                  }else if(HjelpeKlasserDato.erHelg(best)){
                        type = "Helg";
                  }else if(HjelpeKlasserDato.erManTor(best)){
                        type = "ManTor";
                  }
            }

            db.settSql("INSERT INTO TREKNING (Nummer, Bestilling, Hytte,
Prioritet, KoNummer, Type)" +
                        " VALUES("+nNummer +", " + best.getNummer() + ", '" +
best.getHytte() + "', 1, 0, '"
                        + type + "')");
            if(db.kobleTil(false)){
            }
            db.kobleFra();
            if(type.equals("Dag") || type.equals("Helg") || type.equals
("ManTor")){
                  type = "Periode";
            }else{
                  type = "Ferie";
            }
            if(karantene){
                  settKarantene(best.getHytte(), best.getKunde(), type,
best.getDatoFra());
            }
      }

      /**********************************************************************
       * Metode som returnerer om en bestilling er tildelt
       * @param best Aktuell bestilling
       * @return boolean Returnere om bestilling er tildelt
       **********************************************************************/
      public boolean erTildelt(Bestilling best){
            if(best.getNummer() == getForsteBestNr(getTrekningsNr(best))){
                  // Bestillingen er øverst i trekningen
                  return true;
            } else {
                  return false;
            }
      }

      /**********************************************************************
       * Metode som returnerer trekningsnummer for en bestilling
       * @param best Aktuell bestilling
       * @return boolean Returnerer trekningsnummer for en bestilling
```

```
	 ***********************************************************************/
	public int getTrekningsNr(Bestilling best){
		int trekningsNr = -1;

		db.settSql("SELECT Nummer FROM TREKNING WHERE Bestilling='" +
best.getNummer() + "'");

		if(db.kobleTil(true)){
			ResultSet rs = db.hentResultat();
			try {
				while(rs.next()) {
					trekningsNr = rs.getInt("Nummer");
				}
			} catch (SQLException e) {
				e.printStackTrace();
			}
		}
		db.kobleFra();
		return trekningsNr;
	}

	/***********************************************************************
	 * Metode som returnerer neste i trekning
	 * @param trekningsNr Nummeret paa trekningen
	 * @return int returnerer id paa neste i trekningen
	 ***********************************************************************/
	public int getForsteBestNr(int trekningsNr){
		int bestNr = -1;

		db.settSql("SELECT Bestilling FROM TREKNING WHERE Nummer='" +
trekningsNr + "' ORDER BY Prioritet,KoNummer ASC LIMIT 1");

		if(db.kobleTil(true)){
			ResultSet rs = db.hentResultat();
			try {
				while(rs.next()) {
					bestNr = rs.getInt("Bestilling");
				}
			} catch (SQLException e) {
				e.printStackTrace();
			}
		}
		db.kobleFra();
		return bestNr;
	}

	/***********************************************************************
	 * Metode som returnerer om bestillingen er bestilt fra foer av samme
bruker
	 * @param brukernavn Brukernavnet til kunden
	 * @param fraDato Dato fra
	 * @param tilDato Dato til
	 * @return boolean returnerer om brukeren har bestilt samme bestilling fra
foer
	 ***********************************************************************/
	public boolean bestiltFraFoer(String brukernavn, String fraDato, String
tilDato){

		java.util.Date fraDate =
hytteWebUtility.HjelpeKlasserDato.datoStringToDate(fraDato);
		java.util.Date tilDate =
hytteWebUtility.HjelpeKlasserDato.datoStringToDate(tilDato);

		ArrayList<Bestilling> bestillinger = hentAlleKunde(brukernavn);

		boolean funnet = false;
```

```
            if(bestillinger != null){
                  for(int i = 0; i < bestillinger.size(); i++){
                        if(((fraDate.compareTo(bestillinger.get(i).getDateFra())
>= 0)
                                    && (fraDate.compareTo(bestillinger.get
(i).getDateTil()) <= 0)) ||
                              ((tilDate.compareTo(bestillinger.get(i).getDateTil
()) <= 0)
                                    && (tilDate.compareTo(bestillinger.get
(i).getDateFra()) >= 0)) ||
                              (bestillinger.get(i).getDateFra().compareTo
(fraDate) >= 0)
                              && (bestillinger.get(i).getDateFra().compareTo
(tilDate) <= 0)){
                              funnet = true;
                        }
                  }
            }

            return funnet;
      }

      /*********************************************************************
       * Metode som kjorer en trekning ved oppstart av serveren, for aa
       * ta med bestillinger som evt falt bort under nedetid av serveren
       *********************************************************************/
      public void kjorTrekningOppstart(){
            ArrayList<Bestilling> alleBest = hentAlleBekreftet
(HjelpeKlasserDato.hentDagensAar());

            ArrayList<Bestilling> alleUtenTrekningFerie = new
ArrayList<Bestilling>();
            ArrayList<Bestilling> alleUtenTrekningDag = new
ArrayList<Bestilling>();
            ArrayList<Bestilling> alleUtenTrekningHelg = new
ArrayList<Bestilling>();
            ArrayList<Bestilling> alleUtenTrekningManTor = new
ArrayList<Bestilling>();

            // Legger alle bestillinger som ikke har trekning inn i sine
arraylister
            for(int i = 0; i < alleBest.size(); i++){
                  if(harTrekningStreng(alleBest.get(i)).equals("")){
                        if(HjelpeKlasserDato.erEnkeltDag(alleBest.get(i))){
                              alleUtenTrekningDag.add(alleBest.get(i));
                        }else if(HjelpeKlasserDato.erHelg(alleBest.get(i))){
                              alleUtenTrekningHelg.add(alleBest.get(i));
                        }else if(HjelpeKlasserDato.erManTor(alleBest.get(i))){
                              alleUtenTrekningManTor.add(alleBest.get(i));
                        }else{
                              alleUtenTrekningFerie.add(alleBest.get(i));
                        }
                  }
            }

            // Sjekker om det finnes uten trekning i arraylistene
            if(alleUtenTrekningDag.size()>0){
                  for(int i = 0; i < alleUtenTrekningDag.size(); i++){
                        if(!HjelpeKlasserDato.erOverAvbestillingsFrist
(alleUtenTrekningDag.get(i))){
                              alleUtenTrekningDag.remove(i);
                              i--;
                        }
                  }
            }
```

```
            if(alleUtenTrekningHelg.size()>0){
                  for(int i = 0; i < alleUtenTrekningHelg.size(); i++){
                        if(!HjelpeKlasserDato.erOverAvbestillingsFrist
(alleUtenTrekningHelg.get(i))){
                              alleUtenTrekningHelg.remove(i);
                              i--;
                        }
                  }
            }
            if(alleUtenTrekningManTor.size()>0){
                  for(int i = 0; i < alleUtenTrekningManTor.size(); i++){
                        if(!HjelpeKlasserDato.erOverAvbestillingsFrist
(alleUtenTrekningManTor.get(i))){
                              alleUtenTrekningManTor.remove(i);
                              i--;
                        }
                  }
            }
            if(alleUtenTrekningFerie.size()>0){
                  for(int i = 0; i < alleUtenTrekningFerie.size(); i++){
                        if(!HjelpeKlasserDato.erOverAvbestillingsFrist
(alleUtenTrekningFerie.get(i))){
                              alleUtenTrekningFerie.remove(i);
                              i--;
                        }
                  }
            }

            // Dersom arraylistene inneholder bestillinger skal det kjores
trekning
            ArrayList<String> hytter = hDAO.hentHytteNavn();
            ArrayList<Bestilling> tmpDag = new ArrayList<Bestilling>();
            ArrayList<Bestilling> tmpHelg = new ArrayList<Bestilling>();
            ArrayList<Bestilling> tmpFerie = new ArrayList<Bestilling>();
            ArrayList<Bestilling> tmpManTor = new ArrayList<Bestilling>();

            for(int i = 0; i< hytter.size(); i++){
                  if(alleUtenTrekningDag.size()>0){
                        for(int j = 0; j < alleUtenTrekningDag.size(); j++){
                              if(alleUtenTrekningDag.get(j).getHytte().equals
(hytter.get(i))){
                                    tmpDag.add(alleUtenTrekningDag.get(j));
                              }
                        }
                  }
                  if(alleUtenTrekningHelg.size()>0){
                        for(int j = 0; j < alleUtenTrekningHelg.size(); j++){
                              if(alleUtenTrekningHelg.get(j).getHytte().equals
(hytter.get(i))){
                                    tmpHelg.add(alleUtenTrekningHelg.get(j));
                              }
                        }
                  }
                  if(alleUtenTrekningManTor.size()>0){
                        for(int j = 0; j < alleUtenTrekningManTor.size(); j++){
                              if(alleUtenTrekningManTor.get(j).getHytte().equals
(hytter.get(i))){
                                    tmpManTor.add(alleUtenTrekningManTor.get
(j));
                              }
                        }
                  }
                  if(alleUtenTrekningFerie.size()>0){
                        for(int j = 0; j < alleUtenTrekningFerie.size(); j++){
                              if(alleUtenTrekningFerie.get(j).getHytte().equals
(hytter.get(i))){
```

```
                                    tmpFerie.add(alleUtenTrekningFerie.get(j));
                              }
                        }
                  }
                  leggTilTrekningDager(tmpDag, tmpHelg, tmpManTor, hytter.get
(i));
                  if(tmpFerie.size()>0) leggTilTrekningFerie(tmpFerie,
hytter.get(i), fDAO.tilhorerFerie(tmpFerie.get(0)));
                  tmpDag.clear();
                  tmpHelg.clear();
                  tmpManTor.clear();
                  tmpFerie.clear();

            }
      }
}
```

DatabaseDAOImpl.java

```
package hytteWeb.dao.context;
import hytteWebUtility.Setting;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class DatabaseDAOImpl {
      private String dbUrl;
      String dbClass;
      Statement stmt;
      private String query;
      private String user;
      private String pass;
      private ResultSet rs;
      private Connection con;

      /**************************************************
       * Konstruktør
       * @param String host
       * @param String db
       **************************************************/
      public DatabaseDAOImpl() {
            user = Setting.DB_USER;
            pass = Setting.DB_PASS;
            dbUrl = "jdbc:mysql://" + Setting.DB_HOST + "/" + Setting.DB_DB;
            dbClass = "com.mysql.jdbc.Driver";
            this.query = "";
            rs = null;
            stmt = null;
            con = null;
      }

      /**************************************************
       * Metode for å sette SQL
       * @param String sql
       **************************************************/
      public void settSql(String sql) {
            this.query = sql;
      }

      /**************************************************
       * Metode for å returnere resultat
       * @return ResultSet rs;
```

```
	 ************************************************/
	public ResultSet hentResultat() {
		return rs;
	}

	/************************************************
	 * Metode for å koble til db
	 ************************************************/
	public boolean kobleTil(boolean erSelect) {
		boolean status = false;
		try {
			Class.forName(dbClass);
			con = DriverManager.getConnection (dbUrl, user, pass);
			stmt = con.createStatement();
			if(erSelect){
				rs = stmt.executeQuery(query);
				if(rs.next()){
					status = true;
				}else{
					status = false;
				}
				rs.beforeFirst();
			} else {
				int retur = stmt.executeUpdate(query);
				if(retur > 0) status = true;
				else status = false;

			}
		}

		catch(ClassNotFoundException e) {
			e.printStackTrace();
		}

		catch(SQLException e) {
			e.printStackTrace();
		}
		return status;
	}

	/************************************************
	 * Metode for å koble fra db
	 ************************************************/
	public void kobleFra() {
		try {
			stmt.close();
			if(rs != null) rs.close();
			con.close();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}


}
```

EpostDAOImpl.java

```
package hytteWeb.dao.context;

import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
```

```
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Properties;
import java.util.regex.Pattern;

import hytteWeb.model.Bestilling;
import hytteWebUtility.Setting;
import hytteWebUtility.Utility;

public class EpostDAOImpl {

	private DatabaseDAOImpl db;
	private InnstillingDAOImpl iDAO;
	private BestillingDAOImpl bDAO;
	private String fra;
	private String til;
	private String emne;
	private String melding;
	private Properties props;
	private Authenticator auth;
	private Session mailSession;

	public EpostDAOImpl(){
		db = new DatabaseDAOImpl();
		iDAO = new InnstillingDAOImpl();
		bDAO = new BestillingDAOImpl();
	}

	public void sendReservasjonBekreft(Bestilling best){
		til = Utility.lagEpost(best.getKunde());
		emne = "Bekreft reservasjon";

		melding = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0Transitional//
EN'> \n";
		melding += "<html> \n";
		melding += "<head> \n";
		melding += " <title>Bekreft reservasjon</title> \n";
		melding += "</head> \n";
		melding += "<body> \n";
		melding += "<p>" + hentHeader() + "<br /></p>\n";
		melding += "<p>" + hentReservasjonBekreft() + "</p>\n";
		melding += "<p><u>Bestillingsinformasjon</u><br>\n";
		melding += "Hytte: " + best.getHytte() + "<br>";
		melding += "Fra: " + best.getDatoFra() + "<br>";
		melding += "Til: " + best.getDatoTil() + "<br>";
		melding += "Pris: " + best.getSum() + " kroner</p><br><br>";
		melding += "<a href=\"" + Setting.SYSTEM_URL_ROOT + "/Hytte/
Bestilling?action=confirm&type=reservasjon&key=" + best.getAutKode() +
"\">Bekreft reservasjon</a><br>";
		melding += "<a href=\"" + Setting.SYSTEM_URL_ROOT + "/Hytte/
Bestilling?action=cancel&type=reservasjon&key=" + best.getAutKode() +
"\">Kanseller reservasjon</a>\n";
		melding += "<p>" + hentFooter() + "</p><br>\n";
		melding += "<p>---- <br />";
		melding += "Denne eposten er tiltenkt <i>" + Utility.lagEpost
(best.getKunde()) + "</i>. Dersom du har mottatt denne eposten ved en feil,
vennligst gi beskjed til hyttestyret.</p>\n";
		melding += "</body>\n";
		melding += "</html>\n";
```

```
            try {
                  sendEpost();
            } catch (AddressException e) {
                  e.printStackTrace();
            } catch (MessagingException e) {
                  e.printStackTrace();
            }
      }

      public void sendReservasjonBekreftet(Bestilling best){
            til = Utility.lagEpost(best.getKunde());
            emne = "Din reservasjon er bekreftet";

            melding = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0Transitional//
EN'> \n";
            melding += "<html> \n";
            melding += "<head> \n";
            melding += " <title>Din reservasjon er bekreftet</title> \n";
            melding += "</head> \n";
            melding += "<body> \n";
            melding += "<p>" + hentHeader() + "<br /></p>\n";
            melding += "<p>" + hentReservasjonBekreftet() + "</p>\n";
            melding += "<p><u>Bestillingsinformasjon</u><br>\n";
            melding += "Hytte: " + best.getHytte() + "<br>";
            melding += "Fra: " + best.getDatoFra() + "<br>";
            melding += "Til: " + best.getDatoTil() + "<br>";
            melding += "Pris: " + best.getSum() + " kroner</p><br><br>";
            melding += "<a href=\"" + Setting.SYSTEM_URL_ROOT + "/Hytte/
Bestilling?action=cancel&type=reservasjon&key=" + best.getAutKode() +
"\">Kanseller reservasjon</a>\n";
            melding += "<p>" + hentFooter() + "</p><br>\n";
            melding += "<p>---- <br />";
            melding += "Denne eposten er tiltenkt <i>" + Utility.lagEpost
(best.getKunde()) + "</i>. Dersom du har mottatt denne eposten ved en feil,
vennligst gi beskjed til hyttestyret.</p>\n";
            melding += "</body>\n";
            melding += "</html>\n";

            try {
                  sendEpost();
            } catch (AddressException e) {
                  e.printStackTrace();
            } catch (MessagingException e) {
                  e.printStackTrace();
            }
      }

      public void sendTildeltBekreft(Bestilling best){
            til = Utility.lagEpost(best.getKunde());
            emne = "Bekreft tildeling";

            melding = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0Transitional//
EN'> \n";
            melding += "<html> \n";
            melding += "<head> \n";
            melding += " <title>Bekreft tildeling</title> \n";
            melding += "</head> \n";
            melding += "<body> \n";
            melding += "<p>" + hentHeader() + "<br /></p>\n";
            melding += "<p>" + hentTildelingBekreft() + "</p>\n";
            melding += "<p><u>Bestillingsinformasjon</u><br>\n";
            melding += "Hytte: " + best.getHytte() + "<br>";
            melding += "Fra: " + best.getDatoFra() + "<br>";
            melding += "Til: " + best.getDatoTil() + "<br>";
            melding += "Pris: " + best.getSum() + " kroner</p><br><br>";
```

```
            melding += "<a href=\"" + Setting.SYSTEM_URL_ROOT + "/Hytte/
Bestilling?action=confirm&type=tildeling&key=" + best.getAutKodeTildelt() +
"\">Bekreft tildeling</a><br>";
            melding += "<a href=\"" + Setting.SYSTEM_URL_ROOT + "/Hytte/
Bestilling?action=cancel&type=tildeling&key=" + best.getAutKodeTildelt() +
"\">Kanseller tildeling</a>\n";
            melding += "<p>" + hentFooter() + "</p><br>\n";
            melding += "<p>---- <br />";
            melding += "Denne eposten er tiltenkt <i>" + Utility.lagEpost
(best.getKunde()) + "</i>. Dersom du har mottatt denne eposten ved en feil,
vennligst gi beskjed til hyttestyret.</p>\n";
            melding += "</body>\n";
            melding += "</html>\n";


            try {
                  sendEpost();
            } catch (AddressException e) {
                  e.printStackTrace();
            } catch (MessagingException e) {
                  e.printStackTrace();
            }
      }

      public void sendTildeltBekreftet(Bestilling best){
            til = Utility.lagEpost(best.getKunde());
            emne = "Din tildelingen er bekreftet";

            melding = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0Transitional//
EN'> \n";
            melding += "<html> \n";
            melding += "<head> \n";
            melding += " <title>Din tildeling er bekreftet</title> \n";
            melding += "</head> \n";
            melding += "<body> \n";
            melding += "<p>" + hentHeader() + "<br /></p>\n";
            melding += "<p>" + hentTildelingBekreftet() + "</p>\n";
            melding += "<p><u>Bestillingsinformasjon</u><br>\n";
            melding += "Hytte: " + best.getHytte() + "<br>";
            melding += "Fra: " + best.getDatoFra() + "<br>";
            melding += "Til: " + best.getDatoTil() + "<br>";
            melding += "Pris: " + best.getSum() + " kroner</p><br><br>";
            melding += "<a href=\"" + Setting.SYSTEM_URL_ROOT + "/Hytte/
Bestilling?action=cancel&type=tildeling&key=" + best.getAutKodeTildelt() +
"\">Kanseller tildeling</a>\n";
            melding += "<p>" + hentFooter() + "</p><br>\n";
            melding += "<p>---- <br />";
            melding += "Denne eposten er tiltenkt <i>" + til + "</i>. Dersom du
har mottatt denne eposten ved en feil, vennligst gi beskjed til hyttestyret.</p>
\n";
            melding += "</body>\n";
            melding += "</html>\n";

            try {
                  sendEpost();
            } catch (AddressException e) {
                  e.printStackTrace();
            } catch (MessagingException e) {
                  e.printStackTrace();
            }
      }

      public void sendAdminEpost(Bestilling best){
            til = iDAO.hentEpostAdmin();
            emne = "En hytte har blitt tildelt!";
```

```
            melding = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0Transitional//
EN'> \n";
            melding += "<html> \n";
            melding += "<head> \n";
            melding += " <title>En hytte har blitt tildelt!</title> \n";
            melding += "</head> \n";
            melding += "<body> \n";
            melding += "<p>Hei,<br /></p>\n";
            melding += "<p>En hytte har nå blitt tildelt.</p>\n";
            melding += "<p><u>Bestillingsinformasjon</u><br>\n";
            melding += "Hytte: " + best.getHytte() + "<br>";
            melding += "Kunde: " + best.getKunde() + "<br>";
            melding += "Fra: " + best.getDatoFra() + "<br>";
            melding += "Til: " + best.getDatoTil() + "<br>";
            melding += "Pris: " + best.getSum() + " kroner</p><br><br>";
            melding += "<p>---- <br />";
            melding += "Denne eposten er sendt til deg fordi din epost (<i>" +
til + "</i>) er oppgitt som admin-epost i hyttesystemet.<br>\n";
            melding += "Dersom du har mottatt denne eposten ved en feil,
vennligst gi beskjed til hyttestyret.</p>\n";
            melding += "</body>\n";
            melding += "</html>\n";

            try {
                  sendEpost();
            } catch (AddressException e) {
                  e.printStackTrace();
            } catch (MessagingException e) {
                  e.printStackTrace();
            }
      }

      public void sendAvbestillingMottatt(Bestilling best){
            til = Utility.lagEpost(best.getKunde());
            emne = "Kansellering mottatt";


            melding = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0Transitional//
EN'> \n";
            melding += "<html> \n";
            melding += "<head> \n";
            melding += " <title>Kansellering mottatt</title> \n";
            melding += "</head> \n";
            melding += "<body> \n";
            melding += "<p>" + hentHeader() + "<br /></p>\n";
            melding += "<p>" + hentAvbestilling() + "</p>\n";
            melding += "<p><u>Informasjon</u><br>\n";
            melding += "Hytte: " + best.getHytte() + "<br>";
            melding += "Fra: " + best.getDatoFra() + "<br>";
            melding += "Til: " + best.getDatoTil() + "<br>";
            melding += "Pris: " + best.getSum() + " kroner</p><br><br>";
            melding += "<p>" + hentFooter() + "</p><br>\n";
            melding += "<p>---- <br />";
            melding += "Denne eposten er tiltenkt <i>" + til + "</i>. Dersom du
har mottatt denne eposten ved en feil, vennligst gi beskjed til hyttestyret.</p>
\n";
            melding += "</body>\n";
            melding += "</html>\n";


            try {
                  sendEpost();
            } catch (AddressException e) {
                  e.printStackTrace();
            } catch (MessagingException e) {
                  e.printStackTrace();
```

```
		}
	}

	public void sendLonnRapport(ArrayList<Bestilling> bestillinger){
		til = iDAO.hentEpostLonn();
		emne = "Betalingsrapport for hytter";

		melding = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0Transitional//
EN'> \n";
		melding += "<html> \n";
		melding += "<head> \n";
		melding += " <title>Betalingsrapport for hytter</title>\n";
		melding += "<style>\n";
		melding += "table {\n";
		melding += "font: 11px/24px Verdana, Arial, Helvetica, sans-serif;
\n";
		melding += "border-collapse: collapse;\n";
		melding += "width: 100%;\n";
		melding += "}\n";
		melding += "th {\n";
		melding += "padding: 0 0.5em;\n";
		melding += "text-align: left;\n";
		melding += "}\n";
		melding += "tr.yellow td {\n";
		melding += "border-top: 1px solid #FB7A31;\n";
		melding += "border-bottom: 1px solid #FB7A31;\n";
		melding += "background: #cedeeb;\n";
		melding += "}\n";
		melding += "td {\n";
		melding += "border-bottom: 1px solid #CCC;\n";
		melding += "padding: 0 0.5em;\n";
		melding += "}\n";
		melding += "td:first-child {\n";
		melding += "width: 190px;\n";
		melding += "}\n";
		melding += "td+td {\n";
		melding += "border-left: 1px solid #CCC;\n";
		melding += "text-align: center;\n";
		melding += "}\n";
		melding += "</style>\n";
		melding += "</head> \n";
		melding += "<body> \n";
		melding += "<p>" + hentHeader() + "<br /></p>\n";
		melding += "<p>" + hentLonnrapport() + "</p>\n";
		melding += "<h3>Rapport</h3>\n";
		melding += "<table>\n";
		melding += "<tr>\n";
		melding += "<td>Hytte</td>\n";
		melding += "<td>Ansatt</td>\n";
		melding += "<td>Fra</td>\n";
		melding += "<td>Til</td>\n";
		melding += "<td>Pris</td>\n";
		melding += "</tr>\n";

		for(int i = 0; i < bestillinger.size(); i++){
			melding += "<tr>\n";
			melding += "<td>" + bestillinger.get(i).getHytte() + "</td>
\n";
			melding += "<td>" + bestillinger.get(i).getKunde() + "</td>
\n";
			melding += "<td>" + bestillinger.get(i).getDatoFra() + "</td>
\n";
			melding += "<td>" + bestillinger.get(i).getDatoTil() + "</td>
\n";
			melding += "<td>" + bestillinger.get(i).getSum() + "</td>\n";
			melding += "</tr>\n";
```

```
            }

            melding += "</table>\n";
            melding += "<p>" + hentFooter() + "</p>\n";
            melding += "<p>---- <br />\n";
            melding += "Denne eposten er tiltenkt <i>" + til + "</i>. Dersom du
har mottatt denne eposten ved en feil, vennligst gi beskjed til hyttestyret.</p>
\n";
            melding += "</body>\n";
            melding += "</html>\n";

            try {
                  sendEpost();
            } catch (AddressException e) {
                  e.printStackTrace();
            } catch (MessagingException e) {
                  e.printStackTrace();
            }
      }

      public void sendMineBestillinger(ArrayList<Bestilling> bestillinger){
            til = Utility.lagEpost(bestillinger.get(0).getKunde());
            emne = "Mine hyttereservasjoner";

            melding = "<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0Transitional//
EN'> \n";
            melding += "<html> \n";
            melding += "<head> \n";
            melding += " <title>Mine hyttereservasjoner</title>\n";
            melding += "<style>\n";
            melding += "table {\n";
            melding += "font: 11px/24px Verdana, Arial, Helvetica, sans-serif;
\n";
            melding += "border-collapse: collapse;\n";
            melding += "width: 100%;\n";
            melding += "}\n";
            melding += "th {\n";
            melding += "padding: 0 0.5em;\n";
            melding += "text-align: left;\n";
            melding += "}\n";
            melding += "tr.yellow td {\n";
            melding += "border-top: 1px solid #FB7A31;\n";
            melding += "border-bottom: 1px solid #FB7A31;\n";
            melding += "background: #cedeeb;\n";
            melding += "}\n";
            melding += "td {\n";
            melding += "border-bottom: 1px solid #CCC;\n";
            melding += "padding: 0 0.5em;\n";
            melding += "}\n";
            melding += "td:first-child {\n";
            melding += "width: 190px;\n";
            melding += "}\n";
            melding += "td+td {\n";
            melding += "border-left: 1px solid #CCC;\n";
            melding += "text-align: center;\n";
            melding += "}\n";
            melding += "</style>\n";
            melding += "</head> \n";
            melding += "<body> \n";
            melding += "<p>" + hentHeader() + "<br /></p>\n";
            melding += "<p>" + hentMinebestillinger() + "</p>\n";
            melding += "<h3>Bestillinger</h3>\n";
            melding += "<table>\n";
            melding += "<tr>\n";
            melding += "<td>Hytte</td>\n";
            melding += "<td>Bruker</td>\n";
```

```
            melding += "<td>Fra</td>\n";
            melding += "<td>Til</td>\n";
            melding += "<td>Pris</td>\n";
            melding += "<td>Bekreftet reservert</td>\n";
            melding += "<td>Tildelt</td>\n";
            melding += "<td>Bekreftet tildelt</td>\n";
            melding += "<td>Bekreft</td>\n";
            melding += "<td>Kanseller</td>\n";
            melding += "</tr>\n";

            for(int i = 0; i < bestillinger.size(); i++){
                  melding += "<tr>\n";
                  melding += "<td>" + bestillinger.get(i).getHytte() + "</td>
\n";
                  melding += "<td>" + bestillinger.get(i).getKunde() + "</td>
\n";
                  melding += "<td>" + bestillinger.get(i).getDatoFra() + "</td>
\n";
                  melding += "<td>" + bestillinger.get(i).getDatoTil() + "</td>
\n";
                  melding += "<td>" + bestillinger.get(i).getSum() + "</td>\n";

                  if(bestillinger.get(i).getBekreftet()){
                        melding += "<td>Ja</td>\n";
                  } else {
                        melding += "<td>Nei</td>\n";
                  }

                  if(bDAO.erTildelt(bestillinger.get(i))){
                        melding += "<td>Ja</td>\n";
                  } else {
                        melding += "<td>Nei</td>\n";
                  }

                  if(bestillinger.get(i).getBekreftetTildelt()){
                        melding += "<td>Ja</td>\n";
                  } else {
                        melding += "<td>Nei</td>\n";
                  }

                  if(bDAO.harTrekning(bestillinger.get(i)) || bestillinger.get
(i).getBekreftetTildelt()){
                        melding += "<td><a href=\"" + Setting.SYSTEM_URL_ROOT +
"/Hytte/Bestilling?action=confirm&type=tildeling&key=" + bestillinger.get
(i).getAutKodeTildelt() + "\">Bekreft tildeling</a></td>\n";
                  } else {
                        melding += "<td><a href=\"" + Setting.SYSTEM_URL_ROOT +
"/Hytte/Bestilling?action=confirm&type=reservasjon&key=" + bestillinger.get
(i).getAutKode() + "\">Bekreft reservasjon</a></td>\n";
                  }

                  if(bestillinger.get(i).getBekreftetTildelt()){
                        melding += "<td><a href=\"" + Setting.SYSTEM_URL_ROOT +
"/Hytte/Bestilling?action=cancel&type=tildeling&key=" + bestillinger.get
(i).getAutKodeTildelt() + "\">Kanseller tildeling</a></td>\n";
                  } else {
                        melding += "<td><a href=\"" + Setting.SYSTEM_URL_ROOT +
"/Hytte/Bestilling?action=cancel&type=reservasjon&key=" + bestillinger.get
(i).getAutKode() + "\">Kanseller reservasjon</a></td>\n";
                  }

                  melding += "</tr>\n";
            }

            melding += "</table><br>\n";
            melding += "<p>" + hentFooter() + "</p>\n";
```

```
            melding += "<p>---- <br />\n";
            melding += "Denne eposten er tiltenkt <i>" + til + "</i>. Dersom du
har mottatt denne eposten ved en feil, vennligst gi beskjed til hyttestyret.</p>
\n";
            melding += "</body>\n";
            melding += "</html>\n";

            try {
                  sendEpost();
            } catch (AddressException e) {
                  e.printStackTrace();
            } catch (MessagingException e) {
                  e.printStackTrace();
            }
      }

      public String hentHeader(){
            String tmp = "";

            db.settSql("SELECT * FROM EPOST WHERE funksjon='header'");
            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();

                  try {
                        while(rs.next()) {
                              tmp = rs.getString("tekst");
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            }
            db.kobleFra();
            return tmp;
      }

      public String hentFooter(){
            String tmp = "";

            db.settSql("SELECT * FROM EPOST WHERE funksjon='footer'");
            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();

                  try {
                        while(rs.next()) {
                              tmp = rs.getString("tekst");
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            }
            db.kobleFra();
            return tmp;
      }

      public String hentReservasjonBekreft(){
            String tmp = "";

            db.settSql("SELECT * FROM EPOST WHERE
funksjon='reservasjonBekreft'");
            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();

                  try {
                        while(rs.next()) {
```

```
                        tmp = rs.getString("tekst");
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                }
                db.kobleFra();
            }
            db.kobleFra();
            return tmp;
      }

      public String hentReservasjonBekreftet(){
            String tmp = "";

            db.settSql("SELECT * FROM EPOST WHERE
funksjon='reservasjonBekreftet'");
            if(db.kobleTil(true)){
                ResultSet rs = db.hentResultat();

                try {
                    while(rs.next()) {
                        tmp = rs.getString("tekst");
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                }
                db.kobleFra();
            }
            db.kobleFra();
            return tmp;
      }

      public String hentTildelingBekreft(){
            String tmp = "";

            db.settSql("SELECT * FROM EPOST WHERE funksjon='tildelingBekreft'");
            if(db.kobleTil(true)){
                ResultSet rs = db.hentResultat();

                try {
                    while(rs.next()) {
                        tmp = rs.getString("tekst");
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                }
                db.kobleFra();
            }
            db.kobleFra();
            return tmp;
      }

      public String hentTildelingBekreftet(){
            String tmp = "";

            db.settSql("SELECT * FROM EPOST WHERE
funksjon='tildelingBekreftet'");
            if(db.kobleTil(true)){
                ResultSet rs = db.hentResultat();

                try {
                    while(rs.next()) {
                        tmp = rs.getString("tekst");
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
```

```
            }
            db.kobleFra();
      }
      db.kobleFra();
      return tmp;
}

public String hentAvbestilling(){
      String tmp = "";

      db.settSql("SELECT * FROM EPOST WHERE funksjon='avbestilling'");
      if(db.kobleTil(true)){
            ResultSet rs = db.hentResultat();

            try {
                  while(rs.next()) {
                        tmp = rs.getString("tekst");
                  }
            } catch (SQLException e) {
                  e.printStackTrace();
            }
            db.kobleFra();
      }
      db.kobleFra();
      return tmp;
}

public String hentLonnrapport(){
      String tmp = "";

      db.settSql("SELECT * FROM EPOST WHERE funksjon='lonnrapport'");
      if(db.kobleTil(true)){
            ResultSet rs = db.hentResultat();

            try {
                  while(rs.next()) {
                        tmp = rs.getString("tekst");
                  }
            } catch (SQLException e) {
                  e.printStackTrace();
            }
            db.kobleFra();
      }
      db.kobleFra();
      return tmp;
}

public String hentMinebestillinger(){
      String tmp = "";

      db.settSql("SELECT * FROM EPOST WHERE funksjon='minebestillinger'");
      if(db.kobleTil(true)){
            ResultSet rs = db.hentResultat();

            try {
                  while(rs.next()) {
                        tmp = rs.getString("tekst");
                  }
            } catch (SQLException e) {
                  e.printStackTrace();
            }
            db.kobleFra();
      }
      db.kobleFra();
      return tmp;
}
```

```
	public void setHeader(String header){
		db.settSql("UPDATE EPOST SET tekst='"+ header + "' WHERE
funksjon='header'");
		db.kobleTil(false);
		db.kobleFra();
	}

	public void setFooter(String footer){
		db.settSql("UPDATE EPOST SET tekst='"+ footer + "' WHERE
funksjon='footer'");
		db.kobleTil(false);
		db.kobleFra();
	}

	public void setReservasjonBekreft(String reservasjonBekreft){
		db.settSql("UPDATE EPOST SET tekst='"+ reservasjonBekreft + "' WHERE
funksjon='reservasjonBekreft'");
		db.kobleTil(false);
		db.kobleFra();
	}

	public void setReservasjonBekreftet(String reservasjonBekreftet){
		db.settSql("UPDATE EPOST SET tekst='"+ reservasjonBekreftet + "'
WHERE funksjon='reservasjonBekreftet'");
		db.kobleTil(false);
		db.kobleFra();
	}

	public void setTildelingBekreft(String tildelingBekreft){
		db.settSql("UPDATE EPOST SET tekst='"+ tildelingBekreft + "' WHERE
funksjon='tildelingBekreft'");
		db.kobleTil(false);
		db.kobleFra();
	}

	public void setTildelingBekreftet(String tildelingBekreftet){
		db.settSql("UPDATE EPOST SET tekst='"+ tildelingBekreftet + "' WHERE
funksjon='tildelingBekreftet'");
		db.kobleTil(false);
		db.kobleFra();
	}

	public void setAvbestilling(String avbestilling){
		db.settSql("UPDATE EPOST SET tekst='"+ avbestilling + "' WHERE
funksjon='avbestilling'");
		db.kobleTil(false);
		db.kobleFra();
	}

	public void setLonnrapport(String lonnrapport){
		db.settSql("UPDATE EPOST SET tekst='"+ lonnrapport + "' WHERE
funksjon='lonnrapport'");
		db.kobleTil(false);
		db.kobleFra();
	}

	public void setMinebestillinger(String minebestillinger){
		db.settSql("UPDATE EPOST SET tekst='"+ minebestillinger + "' WHERE
funksjon='minebestillinger'");
		db.kobleTil(false);
		db.kobleFra();
	}

	public void setSvarteliste(String svarteliste){
```

```
            db.settSql("UPDATE EPOST SET tekst='"+ svarteliste + "' WHERE
funksjon='svarteliste'");
            db.kobleTil(false);
            db.kobleFra();
      }

      public boolean svartelistet(String epost){
            // Returnerer sann dersom eposten er svartelistet

            String svarteliste = "";

            db.settSql("SELECT * FROM EPOST WHERE funksjon='svarteliste'");
            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();

                  try {
                        while(rs.next()) {
                              svarteliste = rs.getString("tekst");
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            }
            db.kobleFra();

            if(svarteliste.contains(epost)){
                  return true;
            } else {
                  return false;
            }
      }

      // FIXME
      // Trenger ikke denne kanskje?
      public boolean gyldigSvarteliste(String liste){
            return liste.matches("^((\"(?:[^\"]|\"\")*\"|[^,]*)(,(\"(?:[^\"]|
\"\")*\"|[^,]*))*)$");
      }

      public String fraStringTilHtml(String tekst){
            if(tekst.contains("\n")){
                  return tekst.replaceAll("(\r\n|\r|\n|\n\r)", "<br>");
            } else {
                  return tekst;
            }
      }

      public String fraHtmlTilString(String tekst){
             if (tekst.contains("<br>")){
                   return tekst.replaceAll("<br>", "\n");
             } else {
                   return tekst;
             }
      }

      public String fjernStyggKode(String tekst){
      Pattern pattern = null;
            String regex;
            String friForStyggKode = null;
            friForStyggKode = tekst;

            regex = "<[^>]*>";
            pattern = Pattern.compile(regex);

            return pattern.matcher(friForStyggKode).replaceAll("");
```

```
    }

      private void sendEpost() throws AddressException, MessagingException{

            props = new Properties();

            String type = iDAO.hentEpostType();
            if(type != null){
                  if(type.equals("hib")){
                        fra = "hibhytte@hib.no";
                        props.put("mail.smtp.host",Setting.HIB_HOST);
                        props.put("mail.smtp.port",Setting.HIB_PORT);
                        mailSession = Session.getInstance(props);
                  } else if (type.equals("gmail")){
                        fra = "hibhytte@gmail.com";
                        props.put("mail.smtp.ssl.enable", "true");
                        props.put("mail.smtp.user",Setting.GMAIL_USER);
                        props.put("mail.smtp.host",Setting.GMAIL_HOST);
                        props.put("mail.smtp.port",Setting.GMAIL_PORT);
                        props.put("mail.smtp.starttls.enable","true");
                        props.put("mail.smtp.auth", "true");
                        auth = new SMTPAuthenticator();
                        mailSession = Session.getInstance(props,auth);
                  } else if (type.equals("local")){
                        fra = "hibhytte@hib.no";
                        props.put("mail.smtp.host",Setting.LOCAL_HOST);
                        props.put("mail.smtp.port",Setting.LOCAL_PORT);
                        mailSession = Session.getInstance(props);
                  }
            }

            System.setProperty("mail.mime.charset","UTF-8");

            Message message = new MimeMessage(mailSession);
            message.setFrom(new InternetAddress(fra));
            message.addRecipient(Message.RecipientType.TO, new InternetAddress
(til));
            message.setSubject(emne);
            message.setContent(melding, "text/html; charset=utf-8");

            Transport.send(message);
      }

      private class SMTPAuthenticator extends javax.mail.Authenticator {
            public PasswordAuthentication getPasswordAuthentication() {
                  return new PasswordAuthentication(Setting.GMAIL_USER,
Setting.GMAIL_PASS);
            }
      }
}
```

FerieDAOImpl.java

```
package hytteWeb.dao.context;

import hytteWeb.dao.FerieDAO;
import hytteWeb.model.Bestilling;
import hytteWeb.model.Ferie;
import hytteWeb.model.KalenderElement;
import hytteWeb.model.Periode;

import java.sql.Date;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
import java.util.ArrayList;

public class FerieDAOImpl implements FerieDAO {
	DatabaseDAOImpl db;

	public FerieDAOImpl() {
		db = new DatabaseDAOImpl();
	}

	@Override
	public boolean endreFerie(Ferie ferie) {
		return false;
	}

	public String tilhorerFerie(Ferie f, String datoFra) {
		String periode = null;

		db.settSql("SELECT PERIODE.id, PERIODE.Start FROM PERIODE, FERIE
WHERE PERIODE.id = FERIE.Perioder AND FERIE.AvbFrist='" + f.getAvbFrist() +
"'");
		if(db.kobleTil(true)) {
			ResultSet rs = db.hentResultat();

			try {
				while (rs.next()) {
					if(datoFra.equals(rs.getDate("Start").toString()))
{
						periode = "" + rs.getInt("id");
					}
				}

			} catch (SQLException e) {
				e.printStackTrace();
			}
		}
		db.kobleFra();
		return periode;
	}

	public String tilhorerFerie(Bestilling best) {
		ArrayList<Ferie> f = hentFerierUtenPerioder(2010);
		for(int i = 0; i < f.size(); i++){
			db.settSql("SELECT PERIODE.id, PERIODE.Start FROM PERIODE,
FERIE WHERE PERIODE.id = FERIE.Perioder AND FERIE.AvbFrist='" + f.get
(i).getAvbFrist() + "'");
			if(db.kobleTil(true)) {
				ResultSet rs = db.hentResultat();

				try {
					while (rs.next()) {
						if(best.getDatoFra().equals(rs.getDate
("Start").toString())) {
							db.kobleFra();
							return "" + rs.getInt("id");
						}
					}

				} catch (SQLException e) {
					e.printStackTrace();
				}
			}
		}

		db.kobleFra();
		return null;
	}
```

```
	public ArrayList<Ferie> hentFerierUtenPerioder(int aar) {
		ArrayList<Ferie> tmp = new ArrayList<Ferie>();
		Ferie aktuell = null;

		db.settSql("SELECT * FROM FERIE WHERE Aar='" + aar + "'");
		if(db.kobleTil(true)) {
			ResultSet rs = db.hentResultat();

			try {
				while (rs.next()) {
//					String navn, int aar, String beskrivelse, int
pris, Date avbFrist
					KalenderElement aFrist = new KalenderElement
(rs.getDate("AvbFrist").toString());
					aktuell = new Ferie(rs.getString("Navn").toString
(), rs.getInt("Aar"), rs.getString("beskrivelse").toString(),
							rs.getInt("Pris"), aFrist);
					aktuell.setPerioder(null);
					tmp.add(aktuell);
				}

			} catch (SQLException e) {
				e.printStackTrace();
			}
		}
		db.kobleFra();
		return tmp;
	}

	@Override
	public ArrayList<Ferie> hentAlleFerier(int aar) {
		db.settSql("SELECT * FROM FERIE, PERIODE WHERE FERIE.Perioder =
PERIODE.id AND PERIODE.Start >= '" + aar + "-01-01' AND PERIODE.Slutt <= '" +
aar + "-12-31'");
		Ferie forrige = null;
		Ferie aktuell = null;
		ArrayList<Ferie> tmp = new ArrayList<Ferie>();

		if(db.kobleTil(true)) {
			ResultSet rs = db.hentResultat();

			try {
				while (rs.next()) {
//					KalenderElement start, KalenderElement slutt, int
pris
					Periode p = new Periode(new KalenderElement
(rs.getDate("Start")),
							new KalenderElement(rs.getDate
("Slutt")), rs.getInt("Pris"));

//					String navn, int aar, String beskrivelse, int
pris, Date avbFrist
					KalenderElement aFrist = new KalenderElement
(rs.getDate("AvbFrist").toString());
					aktuell = new Ferie(rs.getString("Navn").toString
(), rs.getInt("Aar"), rs.getString("beskrivelse").toString(),
							rs.getInt("Pris"), aFrist);

					if(forrige != null) {
						if(aktuell.equals(forrige)) {
							tmp.get(tmp.size()-1).leggTilPeriode
(p);
							forrige = tmp.get(tmp.size()-1);
						} else {
							aktuell.leggTilPeriode(p);
```

```
                                    tmp.add(aktuell);
                                    forrige = aktuell;
                              }
                        } else {
                              aktuell.leggTilPeriode(p);
                              tmp.add(aktuell);
                              forrige = aktuell;
                        }
                  }

            } catch (SQLException e) {
                  e.printStackTrace();
            }
      }
      db.kobleFra();
      return tmp;
}

public String hentFerie(Date start) {
      String f = "";
      db.settSql("SELECT * FROM FERIE, PERIODE WHERE FERIE.Perioder =
PERIODE.id AND PERIODE.Start = '" + start + "'");
      if(db.kobleTil(true)) {
            ResultSet rs = db.hentResultat();
            try {
                  while (rs.next()) {
                        f = rs.getString("Navn").toString();
                  }
            } catch (SQLException e) {
                  e.printStackTrace();
            }
      }
      db.kobleFra();
      return f;
}

public String hentFerieBeskrivelse(String start) {
      String f = "";
      db.settSql("SELECT * FROM FERIE, PERIODE WHERE FERIE.Perioder =
PERIODE.id AND PERIODE.Start = '" + start + "'");
      if(db.kobleTil(true)) {
            ResultSet rs = db.hentResultat();
            try {
                  while (rs.next()) {
                        f = rs.getString("Beskrivelse").toString();
                  }
            } catch (SQLException e) {
                  e.printStackTrace();
            }
      }
      db.kobleFra();
      return f;
}

public Ferie hentFerieObjekt(String start) {
      Ferie tmp = new Ferie();

      db.settSql("SELECT * FROM FERIE, PERIODE WHERE FERIE.Perioder =
PERIODE.id AND PERIODE.Start = '" + start + "'");
      if(db.kobleTil(true)) {
            ResultSet rs = db.hentResultat();

            try {
                  while (rs.next()) {
                        tmp.setNavn(rs.getString("Navn"));
                        tmp.setAar(rs.getInt("Aar"));
```

```
                              tmp.setAvbFrist(new KalenderElement(rs.getDate
("AvbFrist")));
                              tmp.setBeskrivelse(rs.getString("Beskrivelse"));
                              tmp.setPris(rs.getInt("pris"));
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            }
            db.settSql("SELECT * FROM FERIE, PERIODE WHERE FERIE.Perioder =
PERIODE.id AND FERIE.Navn='" + tmp.getNavn() + "'");
            if(db.kobleTil(true)) {
                  ResultSet rs = db.hentResultat();
                  ArrayList<Periode> perioder = new ArrayList<Periode>();
                  try {
                        while (rs.next()) {
                              Periode tmpPeriode = new Periode();
                              tmpPeriode.setPris(rs.getInt("PERIODE.Pris"));
                              tmpPeriode.setSlutt(new KalenderElement(rs.getDate
("PERIODE.Slutt")));
                              tmpPeriode.setStart(new KalenderElement(rs.getDate
("PERIODE.Start")));
                              perioder.add(tmpPeriode);
                        }
                        tmp.setPerioder(perioder);
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            }
            db.kobleFra();
            return tmp;
      }

      public boolean datoFinnes(KalenderElement start, KalenderElement slutt,
ArrayList<Periode> p) {
            boolean funnet = false;
            int teller = 0;
            KalenderElement stSjekk = new KalenderElement(start.getDag(),
start.getMnd(), start.getAar(), 0);
            KalenderElement slSjekk = new KalenderElement(slutt.getDag(),
slutt.getMnd(), slutt.getAar(), 0);
            slSjekk = KalenderElement.flyttKalemenderElementDager(slSjekk, 1);

            while(!stSjekk.equals(slSjekk)) {
                  teller++;
                  stSjekk = KalenderElement.flyttKalemenderElementDager(stSjekk,
1);
            }

            if(teller > 70) {
                  funnet = true;
                  db.kobleFra();
                  return funnet;
            }

            if(start.equals(slutt)) {

            } else {
                  KalenderElement st = new KalenderElement(start.getDag(),
start.getMnd(), start.getAar(), 0);
                  KalenderElement sl = new KalenderElement(slutt.getDag(),
slutt.getMnd(), slutt.getAar(), 0);
                  sl = KalenderElement.flyttKalemenderElementDager(sl, 1);
                  while(!st.equals(sl)) {
```

```
                        db.settSql("SELECT * FROM PERIODE WHERE (Start <= '" +
st.toString() + "' AND Slutt >= '" + st.toString() + "')");
                        if(db.kobleTil(true)) {
                              ResultSet rs = db.hentResultat();
                              try {
                                    while(rs.next()) {
                                          funnet = true;
                                          db.kobleFra();
                                          return funnet;
                                    }
                              } catch (SQLException e) {
                                    e.printStackTrace();
                              }
                        }
                        st = KalenderElement.flyttKalemenderElementDager(st, 1);
                  }

                  db.kobleFra();

                  if(!funnet && p.size()>0) {
                        st = new KalenderElement(start.getDag(), start.getMnd(),
start.getAar(), 0);
                        sl = new KalenderElement(slutt.getDag(), slutt.getMnd(),
slutt.getAar(), 0);
                        sl = KalenderElement.flyttKalemenderElementDager(sl, 1);
                        while(!st.equals(sl)) {
                              for(int i=0; i<p.size(); i++) {
                                    if(p.get(i).finnesIPerioden(st)) {
                                          funnet = true;
                                          db.kobleFra();
                                          return true;
                                    }
                              }
                              st = KalenderElement.flyttKalemenderElementDager
(st, 1);
                        }
                  }
            }
            db.kobleFra();
            return funnet;
      }

      public boolean finnesFerien(String navn, int aar) {
            boolean funnet = false;
            db.settSql("SELECT * FROM FERIE WHERE Navn = '" + navn + "' AND Aar
>= '" + aar + "'");
            if(db.kobleTil(true)) {
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              funnet = true;
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
            }
            db.kobleFra();
            return funnet;
      }

      @Override
      public ArrayList<Ferie> hentFerie(KalenderElement start, KalenderElement
slutt) {
            return null;
      }
```

```
	@Override
	public boolean leggTilFerie(Ferie ferie, int avb) {
		ArrayList<Periode> perioder = ferie.getPerioder();
		String[] tabFrister = new String[perioder.size()];
		ArrayList<Integer> tabId = new ArrayList<Integer>();

		// Dersom ferien inneholder perioder
		if(perioder.size() > 0){
			// Legger inn perioder
			for(int i = 0; i < perioder.size(); i++){
				KalenderElement aktuell = new KalenderElement
(1,1,2010,1);
				aktuell.setDag(perioder.get(i).getStart().getDag());
				aktuell.setMnd(perioder.get(i).getStart().getMnd());
				aktuell.setAar(perioder.get(i).getStart().getAar());
				aktuell = KalenderElement.flyttKalemenderElementDager
(aktuell, -(avb));
				tabFrister[i] = aktuell.toString();

				db.settSql("INSERT INTO PERIODE (Start, Slutt, Pris)
VALUES('" + perioder.get(i).getStart().toString()
					+ "', '"+ perioder.get(i).getSlutt
().toString() +"', '" + perioder.get(i).getPris() + "')");

				if(db.kobleTil(false)){
					db.kobleFra();
				}else {
					db.kobleFra();
					return false;
				}


			}

			db.settSql("SELECT id FROM PERIODE ORDER BY id DESC");
				if(db.kobleTil(true)){
					ResultSet rs = db.hentResultat();
					try {
						while(rs.next()) {
							tabId.add(rs.getInt("id"));
						}
					} catch (SQLException e) {
						e.printStackTrace();
					}
				}

				db.kobleFra();

				int j = perioder.size() - 1;

			for(int i = 0; i < perioder.size(); i++){
				db.settSql("INSERT INTO FERIE (Navn, Aar, Perioder,
Beskrivelse, Pris, AvbFrist) " +
					"VALUES('" + ferie.getNavn() + "', '" +
ferie.getAar() + "', '"
					+ tabId.get(j) + "', '" +
ferie.getBeskrivelse() + " periode " + (i+1) + "', '" + ferie.getPris()
					+ "', '" + tabFrister[i] + "')");
				if(db.kobleTil(false)){
					db.kobleFra();
				}else {
					db.kobleFra();
					return false;
				}
				j--;
			}
```

```
                db.kobleFra();
                return true;
            }
            db.kobleFra();
            return false;
      }

/
**********************************************************************************
*******************
 * Metode som sjekker om ferie-perioden eksisterer. Dersom perioden eksisterer
returneres prisen.
 * @return int pris

**********************************************************************************
*******************/
      public int hentFeriePris(KalenderElement start, KalenderElement slutt) {
            int pris = 0;
            boolean finnes = false;
            int periode = 0;

            db.settSql("SELECT * FROM PERIODE WHERE Start='" + start.toString()
+ "' && Slutt='" + slutt.toString() + "'");
            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              pris = rs.getInt("Pris");
                              periode = rs.getInt("id");
                              finnes = true;
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            }

            if(finnes && pris==0) {
                  db.settSql("SELECT Pris FROM FERIE WHERE PERIODER = '" +
periode + "'");
                  if(db.kobleTil(true)){
                        ResultSet rs = db.hentResultat();
                        try {
                              while(rs.next()) {
                                    pris = rs.getInt("Pris");
                              }
                        } catch (SQLException e) {
                              e.printStackTrace();
                        }
                        db.kobleFra();
                  }
            }
            db.kobleFra();
            return pris;
      }


      @Override
      public boolean slettFerie(Ferie ferie) {
            return false;
      }

      public String hentAvbFristGittPeriode(int per){
            String res = "";
            db.settSql("SELECT AvbFrist FROM FERIE WHERE PERIODER = '" + per +
"'");
```

```
            if(db.kobleTil(true)){
                  ResultSet rs = db.hentResultat();
                  try {
                        while(rs.next()) {
                              res = rs.getDate("AvbFrist").toString();
                        }
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }
                  db.kobleFra();
            }
            db.kobleFra();
            return res;
      }
}
```

CONTROLLER

Admin.java

```
package hytteWeb.controller;


import hytteWeb.dao.context.BestillingDAOImpl;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class Admin
 */
public class Admin extends HttpServlet {
      private static final long serialVersionUID = 1L;

      protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            // Logg inn
            HttpSession session = request.getSession();
            session.removeAttribute("loginStatus");
            if(request.getSession().getAttribute("admin") != "1") {
                  request.getSession().setAttribute("loginStatus", "Du er ikke
logget inn...");
                        request.getRequestDispatcher("/WEB-INF/admin/
adminLoggInn.jsp").forward(request, response);
            } else {
                  if(request.getParameter("trekning") != null){
                        if(request.getParameter("trekning").equals("yes")){
                              // Kjør trekning
                              if((request.getParameter("fristdag") != null) &&
(request.getParameter("fristhelg") != null)){
                                    // Vi skal bruke egne verdier

                                    if((Integer.parseInt(request.getParameter
("fristdag").toString()) > 0) &&
                                          Integer.parseInt(request.getParameter
("fristhelg").toString()) > 0){
```

```
                                          // Verdien på begge må være større enn
0

                                          BestillingDAOImpl bDAO = new
BestillingDAOImpl();
                                          bDAO.kjorTrekningDager
(Integer.parseInt(request.getParameter("fristdag").toString()),

      Integer.parseInt(request.getParameter("fristhelg").toString()), false);
                                          bDAO.kjorTrekningFerie();
                                          session.setAttribute("trekning",
"Trekning kjørt");
                                          request.getRequestDispatcher("/WEB-
INF/admin/adminHovedside.jsp").forward(request, response);

                                    } else {
                                          session.setAttribute("trekning", "Feil
i URL.");
                                          request.getRequestDispatcher("/WEB-
INF/admin/adminHovedside.jsp").forward(request, response);
                                    }
                              } else {
                                    // Vi skal bruke databaseverdier

                                    BestillingDAOImpl bDAO = new
BestillingDAOImpl();
                                    bDAO.kjorTrekningDager(-1, -1, true);
                                    bDAO.kjorTrekningFerie();
                                    session.setAttribute("trekning", "Trekning
kjørt");
                                    request.getRequestDispatcher("/WEB-INF/
admin/adminHovedside.jsp").forward(request, response);
                              }
                        } else {
                              request.getRequestDispatcher("/WEB-INF/admin/
adminHovedside.jsp").forward(request, response);
                        }
                  } else if(request.getParameter("action") != null){
                        if(request.getParameter("action").equals("logout")){
                              session.removeAttribute("admin");
                              request.getRequestDispatcher("/WEB-INF/admin/
adminLoggInn.jsp").forward(request, response);
                        } else {
                              request.getRequestDispatcher("/WEB-INF/admin/
adminHovedside.jsp").forward(request, response);
                        }
                  } else {
                        request.getRequestDispatcher("/WEB-INF/admin/
adminHovedside.jsp").forward(request, response);
                  }
            }
      }

      protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            HttpSession session = request.getSession();
            session.removeAttribute("loginStatus");

            if(session.getAttribute("admin") == "1") {
                  session.setAttribute("loginStatus", "Du er allerede logget
inn!");
                  request.getRequestDispatcher("/WEB-INF/admin/
adminHovedside.jsp").forward(request, response);
            } else if(request.getParameter("bruker") == "" ||
request.getParameter("pass") == "") {
```

```
                  session.setAttribute("loginStatus", "Alle felt må fylles
ut!");
            request.getRequestDispatcher("/WEB-INF/admin/
adminLoggInn.jsp").forward(request, response);
            } else if(!(request.getParameter("bruker").compareTo("admin") == 0
&& request.getParameter("pass").compareTo("1234") == 0)) {
                  session.setAttribute("loginStatus","Feil brukernavn eller
passord!");
                  request.getRequestDispatcher("/WEB-INF/admin/
adminLoggInn.jsp").forward(request, response);
            } else {
                  session.setAttribute("admin", "1");
                  session.setAttribute("loginStatus","Du er nå logget inn!");
                  request.getRequestDispatcher("/WEB-INF/admin/
adminHovedside.jsp").forward(request, response);
            }
      }
}
```

Bestilling.java

```
package hytteWeb.controller;

import hytteWeb.dao.context.BestillingDAOImpl;
import hytteWeb.dao.context.EpostDAOImpl;
import hytteWeb.dao.context.HytteDAOImpl;
import hytteWeb.model.Hytte;
import hytteWeb.model.Kalender;
import hytteWeb.model.KalenderElement;
import hytteWebUtility.HjelpeKlasserDato;
import hytteWebUtility.Utility;

import java.io.IOException;
import java.util.ArrayList;
import java.util.GregorianCalendar;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class Bestilling
 */
public class Bestilling extends HttpServlet {
      private static final long serialVersionUID = 1L;
      private HytteDAOImpl hDAO = new HytteDAOImpl();
      private BestillingDAOImpl bDAO = new BestillingDAOImpl();

      protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            if(request.getParameter("id") != null){
                  HttpSession session = request.getSession();

                  KalenderElement e = KalenderElement.fraGregorian(new
GregorianCalendar());
                  Kalender kalender = new Kalender(e.getMnd(), e.getAar());

                  // Setter hyttenavn til venstre meny
                  ArrayList<String> hNavn = hDAO.hentHytteNavn();
                  request.setAttribute("liste", hNavn);

                  // Henter parametere fra request
```

```
                  String id = request.getParameter("id").toString();
                  //String action = request.getParameter("action").toString();

                  // Henter aktuell hytte som angitt i ID i request
                  Hytte aktuell = hDAO.hentHytte(id);
                  if(aktuell != null){
                        // Setter hytten paa request
                        request.setAttribute("hytte", aktuell);

                        // Kalender
                        if(session.getAttribute("Dato") != null){
                              e = (KalenderElement)session.getAttribute("Dato");
                        } else {
                              session.setAttribute("Dato", e);
                        }

                        if(session.getAttribute("Kalender") != null){
                              kalender = (Kalender)session.getAttribute
("Kalender");
                        } else {
                              session.setAttribute("Kalender", kalender);
                        }

                        kalender.settHyttenavn(id);
                        request.setAttribute("kalTab", kalender.hentKalender());
                        request.setAttribute("ferier", kalender.getFerie());
                        request.getRequestDispatcher("/WEB-INF/
vishyttebestilling.jsp").forward(request, response);
                  }else{
                        request.setAttribute("feil", "Hytten eksisterer ikke i
systemet.");
                        request.getRequestDispatcher("/WEB-INF/
feil.jsp").forward(request, response);
                  }
            }else if((request.getParameter("action") != null) &&
(request.getParameter("key") != null)
                        && (request.getParameter("type") != null)){
                  if(request.getParameter("action").equals("confirm")){
                        if(request.getParameter("type").equals("reservasjon")){
                              // Bekreft reservasjon
                              if(Utility.uuidSjekk(request.getParameter
("key").toString())){
                                    // UUID gyldig
                                          if(bDAO.finnesBestilling
(request.getParameter("key").toString(), false)){
                                                // Bestilling finnes
                                                hytteWeb.model.Bestilling best =
bDAO.hentBestilling(request.getParameter("key").toString(), false);

                                                if(best.getBekreftet()){
                                                      // Bestilling allerede
bekreftet
                                                      request.setAttribute
("feil", "Reservasjonen er allerede bekreftet.");

      request.getRequestDispatcher("/WEB-INF/feil.jsp").forward(request,
response);
                                                } else {
                                                      // Bestilling ikke
bekreftet
                                                      if(bDAO.harTrekning(best)){
                                                            // «Perioden»
bestillinge tilhører, har en trekning
                                                            if
(bDAO.leggTilBestillingTrekning(best)){
```

```
                                                                        // Bestilling
lagt til trekning
                                                                  } else {
                                                                        // Bestilling
ikke lagt til

      request.setAttribute("feil", "Reservasjonen kunne ikke bekreftes. Forsøk
igjen. Dersom du fortsatt får feil, må du kontakte hyttestyret.");

      request.getRequestDispatcher("/WEB-INF/feil.jsp").forward(request,
response);
                                                                  }
                                                                  if
(bDAO.bekreftReservasjon(request.getParameter("key").toString())){
                                                                        // Bestilling
oppdatert
                                                                        EpostDAOImpl
epost = new EpostDAOImpl();

      epost.sendReservasjonBekreftet(best);


      request.setAttribute("bestilling", best);

      request.setAttribute("epost", Utility.lagEpost(best.getKunde()));

      request.setAttribute("melding", "Din reservasjon er nå bekreftet.");


      request.getRequestDispatcher("/WEB-INF/bekreftelse.jsp").forward(request,
response);
                                                                  } else {
                                                                        // Bestilling
ikke oppdatert

      request.setAttribute("feil", "Reservasjonen kunne ikke bekreftes. Forsøk
igjen. Dersom du fortsatt får feil, må du kontakte hyttestyret.");

      request.getRequestDispatcher("/WEB-INF/feil.jsp").forward(request,
response);
                                                                  }
                                                            } else {
                                                                  // «Perioden»
bestillingen tilhører, har ikke en trekning
                                                                  // Vi må da lage en
trekning etter vi har bekreftet bestillingen

                                                                  if
(hytteWebUtility.HjelpeKlasserDato.erOverAvbestillingsFrist(best)){
                                                                        // Bestillingen
vi har er eneste bestilling
                                                                        // Vi er i
tillegg eneste bestillingen
                                                                        // Vi må derfor
lage en trekning


      bDAO.nyTrekningDagerPerioder(best, false);

                                                                  } else {
                                                                        // Vi er ikke
over avbestillingsfrist
                                                                        // Vi trenger
bare sette bekreftet=1
```

```
                                                                  if
(bDAO.bekreftReservasjon(request.getParameter("key").toString())){
                                                                        //
Bestilling oppdatert

      EpostDAOImpl epost = new EpostDAOImpl();

      epost.sendReservasjonBekreftet(best);


      request.setAttribute("bestilling", best);

      request.setAttribute("epost", Utility.lagEpost(best.getKunde()));

      request.setAttribute("melding", "Din reservasjon er nå bekreftet.");


      request.getRequestDispatcher("/WEB-INF/bekreftelse.jsp").forward(request,
response);
                                                                  } else {
                                                                        //
Bestilling ikke oppdatert

      request.setAttribute("feil", "Reservasjonen kunne ikke bekreftes. Forsøk
igjen. Dersom du fortsatt får feil, må du kontakte hyttestyret.");

      request.getRequestDispatcher("/WEB-INF/feil.jsp").forward(request,
response);
                                                                  }
                                                            }
                                                      }
                                                }
                                          } else {
                                                // Bestilling finnes ikke
                                          request.setAttribute("feil", "Nøkkelen
som er angitt i URL'en eksisterer ikke. Sjekk at det er korrekt nøkkel, og prøv
igjen. Dersom du fortsatt får feil, er reservasjonen mest sannsynlig slettet.");
                                          request.getRequestDispatcher("/WEB-
INF/feil.jsp").forward(request, response);
                                          }
                              } else {
                                    // UUID ugyldig
                                    request.setAttribute("feil", "Nøkkelen som
er angitt i URL'en er feil/ugyldig. Sjekk at det er korrekt nøkkel, og prøv
igjen.");
                                    request.getRequestDispatcher("/WEB-INF/
feil.jsp").forward(request, response);
                                    }
                        } else if(request.getParameter("type").equals
("tildeling")){
                              // Bekreft tildeling
                              if(Utility.uuidSjekk(request.getParameter
("key").toString())){
                                    // UUID gyldig
                                          if(bDAO.finnesBestilling
(request.getParameter("key").toString(), true)){
                                          // Bestilling finnes
                                          hytteWeb.model.Bestilling best =
bDAO.hentBestilling(request.getParameter("key").toString(), true);

                                          if(best.getBekreftetTildelt()){
                                                // Bestilling bekreftet
tildelt
                                                request.setAttribute
("feil", "Tildelingen er allerede bekreftet.");
```

```
      request.getRequestDispatcher("/WEB-INF/feil.jsp").forward(request,
response);
                                                      } else {
                                                            // Bestilling ikke
bekreftet tildelt
                                                            if(bDAO.bekreftTildeling
(request.getParameter("key").toString())){
                                                                  // Funnet & oppdatert

                                                                  EpostDAOImpl epost =
new EpostDAOImpl();

      epost.sendTildeltBekreftet(best);
                                                                  epost.sendAdminEpost
(best);

                                                            request.setAttribute
("bestilling", best);
                                                            request.setAttribute
("epost", Utility.lagEpost(best.getKunde()));
                                                            request.setAttribute
("melding", "Din tildeling er nå bekreftet.");


      request.getRequestDispatcher("/WEB-INF/bekreftelse.jsp").forward(request,
response);
                                                            } else {
                                                                  // Ikke funnet eller
ikke oppdatert
                                                      request.setAttribute("feil",
"Nøkkelen som er angitt i URL'en eksisterer ikke. Sjekk at det er korrekt
nøkkel, og prøv igjen. Dersom du fortsatt får feil, er reservasjonen mest
sannsynlig slettet.");
                                                      request.getRequestDispatcher("/
WEB-INF/feil.jsp").forward(request, response);
                                                            }
                                                      }
                                                } else {
                                                      // Bestilling finnes ikke
                                                request.setAttribute("feil", "Nøkkelen
som er angitt i URL'en eksisterer ikke. Sjekk at det er korrekt nøkkel, og prøv
igjen. Dersom du fortsatt får feil, er reservasjonen mest sannsynlig slettet.");
                                                request.getRequestDispatcher("/WEB-
INF/feil.jsp").forward(request, response);
                                                }
                                    } else {
                                          // UUID ugyldig
                                          request.setAttribute("feil", "Nøkkelen som
er angitt i URL'en er feil/ugyldig. Sjekk at det er korrekt nøkkel, og prøv
igjen.");
                                          request.getRequestDispatcher("/WEB-INF/
feil.jsp").forward(request, response);
                                    }
                              } else {
                                    // Feil i URL
                                    request.setAttribute("feil", "Type som er angitt i
URL'en er feil/ugyldig. Sjekk at den er korrekt, og prøv igjen.");
                                    request.getRequestDispatcher("/WEB-INF/
feil.jsp").forward(request, response);
                              }

                  } else if(request.getParameter("action").equals("cancel")){
                        if(request.getParameter("type").equals("reservasjon")){
                              // Kanseller reservasjon
```

```
                              if(Utility.uuidSjekk(request.getParameter
("key").toString())){
                                    // Gyldig UUID
                                    if(bDAO.finnesBestilling
(request.getParameter("key").toString(), false)){
                                          // Bestilling finnes

                                          hytteWeb.model.Bestilling best =
bDAO.hentBestilling(request.getParameter("key").toString(), false);

                                          if(bDAO.slettReservasjon(best)){
                                                // Reservasjon slettet

                                                EpostDAOImpl epost = new
EpostDAOImpl();
                                                epost.sendAvbestillingMottatt
(best);


                                                request.setAttribute
("bestilling", best);
                                                request.setAttribute("epost",
Utility.lagEpost(best.getKunde()));
                                                request.setAttribute("melding",
"Din reservasjon er nå kansellert.");

                                                request.getRequestDispatcher("/
WEB-INF/bekreftelse.jsp").forward(request, response);
                                          } else {
                                                // Reservasjon ikke slettet
                                                request.setAttribute("feil", "Du
har en hytte som er tildelt. Du må bruke linken «Kanseller tildeling» du fikk
pr. epost.");
                                                request.getRequestDispatcher("/
WEB-INF/feil.jsp").forward(request, response);
                                          }
                                    } else {
                                          // Bestilling finnes ikke
                                          request.setAttribute("feil", "Nøkkelen
som er angitt i URL'en eksisterer ikke. Sjekk at det er korrekt nøkkel, og prøv
igjen. Dersom du fortsatt får feil, er reservasjonen mest sannsynlig slettet.");
                                          request.getRequestDispatcher("/WEB-
INF/feil.jsp").forward(request, response);
                                    }
                              } else {
                                    // Ugyldig UUID
                                    request.setAttribute("feil", "Nøkkelen som
er angitt i URL'en er feil/ugyldig. Sjekk at det er korrekt nøkkel, og prøv
igjen.");
                                    request.getRequestDispatcher("/WEB-INF/
feil.jsp").forward(request, response);
                              }
                        } else if(request.getParameter("type").equals
("tildeling")){
                              // Kanseller tildeling
                              if(Utility.uuidSjekk(request.getParameter
("key").toString())){
                                    // Gyldig UUID
                                    if(bDAO.finnesBestilling
(request.getParameter("key").toString(), true)){
                                          // Bestilling finnes

                                          hytteWeb.model.Bestilling best =
bDAO.hentBestilling(request.getParameter("key").toString(), true);

                                          if(bDAO.slettTildeling(best)){
                                                // Tildeling slettet
```

```
                                                EpostDAOImpl epost = new
EpostDAOImpl();
                                                epost.sendAvbestillingMottatt
(best);

                                                request.setAttribute
("bestilling", best);
                                                request.setAttribute("epost",
Utility.lagEpost(best.getKunde()));
                                                request.setAttribute("melding",
"Din tildeling er nå kansellert.");

                                                request.getRequestDispatcher("/
WEB-INF/bekreftelse.jsp").forward(request, response);
                                          } else {
                                                // Tildeling ikke slettet
                                                request.setAttribute("feil",
"Tildelingen er allerede kansellert.");
                                                request.getRequestDispatcher("/
WEB-INF/feil.jsp").forward(request, response);
                                          }
                                    } else {
                                          // Bestilling finnes ikke
                                          request.setAttribute("feil", "Nøkkelen
som er angitt i URL'en eksisterer ikke. Sjekk at det er korrekt nøkkel, og prøv
igjen. Dersom du fortsatt får feil, er reservasjonen mest sannsynlig slettet.");
                                          request.getRequestDispatcher("/WEB-
INF/feil.jsp").forward(request, response);
                                    }
                              } else {
                                    // Ugyldig UUID
                                    request.setAttribute("feil", "Nøkkelen som
er angitt i URL'en er feil/ugyldig. Sjekk at det er korrekt nøkkel, og prøv
igjen.");
                                    request.getRequestDispatcher("/WEB-INF/
feil.jsp").forward(request, response);
                              }
                        } else {
                              // Feil i URL
                              request.setAttribute("feil", "Type som er angitt i
URL'en er feil/ugyldig. Sjekk at den er korrekt, og prøv igjen.");
                              request.getRequestDispatcher("/WEB-INF/
feil.jsp").forward(request, response);
                        }
                  } else {
                        // Feil i URL
                        request.setAttribute("feil", "Action som er angitt i
URL'en er feil/ugyldig. Sjekk at den er korrekt, og prøv igjen.");
                        request.getRequestDispatcher("/WEB-INF/
feil.jsp").forward(request, response);
                  }
            } else {
                  // Feil i URL
                  request.setAttribute("feil", "URL'en du har skrevet, er feil.
Vennligst dobbeltsjekk URL, og prøv igjen.");
                  request.getRequestDispatcher("/WEB-INF/feil.jsp").forward
(request, response);
            }
      }

      protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            HttpSession session = request.getSession();
            String referer = request.getHeader("Referer");
```

```
            if(session.getAttribute("Kalender") == null || session.getAttribute
("Dato") == null){
                  KalenderElement e = KalenderElement.fraGregorian(new
GregorianCalendar());
                  Kalender kalender = new Kalender(e.getMnd(), e.getAar());

                  session.setAttribute("Dato", e);
                  session.setAttribute("Kalender", kalender);
                  response.sendRedirect(referer);
            } else {
                  if(request.getParameter("action").equals("postnew")){
                        // Dersom bestillingen skal registreres

                        if(Utility.epostSjekk(Utility.lagEpost
(request.getParameter("brukernavn").toString()))){
                              // Brukernavn/epost er OK
                              EpostDAOImpl epost = new EpostDAOImpl();
                              if(epost.svartelistet(Utility.lagEpost
(request.getParameter("brukernavn").toString()))){
                                    // Bruker er svartelistet

                                    // Sett nøytral feilmelding, så de ikke får
«FY! DU ER SVARTELISTET» i trynet
                                    session.setAttribute("feilinput", "Ugyldig
bruker. Kontakt hyttestyret.");

                                    // Send tilbake
                                    response.sendRedirect(referer);
                              } else {
                                    // Bruker er ikke svartelistet

                                    Kalender kalender = (Kalender)
session.getAttribute("Kalender");
                                    ArrayList<KalenderElement> tmp =
kalender.getValgteDager();

                                    if(bDAO.bestiltFraFoer(request.getParameter
("brukernavn").toString(), tmp.get(0).toString(), tmp.get(tmp.size()-1).toString
())){
                                          // Dersom valget er innenfor nåværende
bestilling med samme brukernavn

                                          // Sett feilmelding
                                          session.setAttribute("feilinput",
"<br>Det eksisterer allerede en bestilling på angitt brukernavn, <br> som faller
innenfor valgte dato(er). Vennligst forsøk på nytt.");

                                          // Send tilbake
                                          response.sendRedirect(referer);


                                    } else {
                                          hytteWeb.model.Bestilling best =
bDAO.reserverHytte(request.getParameter("id").toString(),request.getParameter
("brukernavn").toString(),

                                    tmp.get(0), tmp.get(tmp.size()-1));

                                          request.setAttribute("bestilling",
best);
                                          request.setAttribute("epost",
Utility.lagEpost(best.getKunde()));

                                          // Send epost
                                          epost.sendReservasjonBekreft(best);
```

```
                                          // Send videre
                                          request.getRequestDispatcher("/WEB-
INF/bestillingmotatt.jsp").forward(request, response);
                                    }
                              }
                        } else {
                              // Brukernavn/epost ikke ok

                              // Sett feilmelding
                              session.setAttribute("feilinput", "Du skrev inn en
ugyldig verdi.");

                              // Send tilbake
                              response.sendRedirect(referer);
                        }

                  // Dersom kalender skal oppdateres med klikk
                  }else{
                        session.removeAttribute("feilinput");
                        session.removeAttribute("visbestill");

                        if(request.getParameter("neste") != null) {
                              Kalender kalender = (Kalender)session.getAttribute
("Kalender");
                              KalenderElement e = (KalenderElement)
session.getAttribute("Dato");

                              KalenderElement.flyttKalemenderElement(e, 1);
                              kalender.settManedAr(e.getMnd(), e.getAar());
                              session.setAttribute("Dato", e);
                              session.setAttribute("Kalender", kalender);
                        }

                        if(request.getParameter("forrige") != null) {
                              Kalender kalender = (Kalender)session.getAttribute
("Kalender");
                              KalenderElement e = (KalenderElement)
session.getAttribute("Dato");

                              KalenderElement.flyttKalemenderElement(e, -1);
                              kalender.settManedAr(e.getMnd(), e.getAar());
                              session.setAttribute("Dato", e);
                              session.setAttribute("Kalender", kalender);
                        }

                        // Det er trykket på en kalenderknapp
                        if(session.getAttribute("Kalender") != null &&
request.getParameter("btn") != null) {
                              Kalender kalender = (Kalender)session.getAttribute
("Kalender");
                              int d = Integer.parseInt(request.getParameter
("dag"));
                              int m = Integer.parseInt(request.getParameter
("mnd"));
                              int a = Integer.parseInt(request.getParameter
("aar"));
                              int u = Integer.parseInt(request.getParameter
("uke"));
                              KalenderElement ke = new KalenderElement(d, m, a,
u);

                              if(ke != null) {
                                    kalender.elementTrykket(ke);
                              }

                              String datoFra = "";
```

```
                              datoFra += ke.getAar() + "-";
                              datoFra += ke.getMnd() + "-";
                              datoFra += ke.getDag();

                              if(HjelpeKlasserDato.senereEnnDagensDato
(HjelpeKlasserDato.datoStringToDate(datoFra))){
                                    // Valgt fra-dato er senere enn dagens dato
                                    // OK, vi går videre

                                    session.setAttribute("visbestill", true);
                                    session.setAttribute("Kalender", kalender);
                                    response.sendRedirect(referer);

                              } else {
                                    // Valgt fra-dato er før, eller lik, dagens
dato
                                    // FY-FY!

                                    session.setAttribute("visbestill", false);
                                    session.setAttribute("Kalender", kalender);
                                    session.setAttribute("feilinput", "Du har
valgt en fra-dato som er før, eller lik, dagens dato.<br>Vennligst velg en fra-
dato som er etter dagens dato.");

                                    response.sendRedirect(referer);
                              }
                        } else {
                              response.sendRedirect(referer);
                        }
                  }
            }
      }
}
```

Trekning.java

```
package hytteWeb.controller;

import hytteWeb.dao.context.BestillingDAOImpl;

import java.util.Date;
import java.util.Timer;
import java.util.TimerTask;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

public class Trekning implements ServletContextListener  {

      BestillingDAOImpl bDAO = new BestillingDAOImpl();
      @Override
      public void contextDestroyed(ServletContextEvent arg0) {
            System.out.println("Slutter...");
      }

      @Override
      public void contextInitialized(ServletContextEvent arg0) {
            System.out.println("Starter...");
            bDAO.kjorTrekningOppstart();
            Timer timer = new Timer();
            timer.scheduleAtFixedRate( new TimerTask(){
                  public void run(){
                        System.out.println("Nå foregår det en trekning...");
                        bDAO.kjorTrekningDager(-1, -1, true);
```

```
                        bDAO.kjorTrekningFerie();
                  }
            }, new Date(), 86400000); // 86400000 er et døgn
      }
}
```

JSP

vishyttebestilling.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Kalender</title>

<link href="hovedside.css" rel="stylesheet" type="text/css" />
<link href="kalender.css" rel="stylesheet" type="text/css" />

</head>
<body>

<div id="mainDiv">
      <div id="toppDiv"></div>
            <div id="centerDiv">
                  <div id="venstre"><jsp:include page="venstreMeny.jsp" /></div>
                  <div id="midt"><div id="centerMargin">

                        <h3 class="centerOverskrift">${hytte.navn }</h3>
                        <table class="kalender" align="center">
                        <tr>
                              <th width="60px" colspan="2">
                                    <form method="post" action="Bestilling?
action=sel" class="formElem">
                                    <input type="submit" class="NesteForrige"
value="Forrige" name="forrige" />
                                    </form>
                              </th>
                           <th colspan="5">${Dato.mndAar}</th>
                           <th width="60px" colspan="2">
                                    <form method="post" action="Bestilling?
action=sel" class="formElem">
                                    <input type="submit" class="NesteForrige"
value="Neste" name="neste" />
                                    </form>
                              </th>
                        </tr>
                          <tr>
                            <th></th>
                            <th>M</th>
                            <th>T</th>
                            <th>O</th>
                            <th>T</th>
                            <th>F</th>
                            <th>L</th>
                            <th>S</th>
                          </tr>

                              <c:forEach var="i" begin="0" step="1" end="7">
```

```
                                    <tr>
                                          <th>${kalTab[i][0].uke}</th>

                                          <c:forEach items="${kalTab[i]}"
var="kal" begin="1" step="1" end="8">
                                                <c:choose>
                                                      <c:when test="${kal.status
== 'Opptatt'}">
                                                      <td class="$
{kal.status}"><span class="vAlign">${kal.dag}</span></td>
                                                      </c:when>
                                                      <c:otherwise>
                                                            <td>
                                                            <form method="post"
action="Bestilling?action=sel" class="formElem">
                                                            <input type="hidden"
name="dag" value="${kal.dag}" />
                                                            <input type="hidden"
name="mnd" value="${kal.mnd}" />
                                                            <input type="hidden"
name="aar" value="${kal.aar}" />
                                                            <input type="hidden"
name="uke" value="${kal.uke}" />
                                                            <input type="submit"
class="${kal.status}" value="${kal.dag}" name="btn" />
                                                            </form>
                                                            </td>
                                                      </c:otherwise>
                                                </c:choose>
                                          </c:forEach>
                                    <th class="HoyreSpacer"></th>
                                    </tr>
                              </c:forEach>
                              <tr class="BunnSpacer"></tr>
                        </table>

                        <div class="forklaring">

                        <table>
                        <tr><td class="OpptattFarge"></td><td
class="forklaringTekst">Hytten er tildelt</td></tr>
                        <tr><td class="LedigFarge"></td><td
class="forklaringTekst">Ledig - ingen bestillinger</td></tr>
                        <tr><td class="BestilltFarge"></td><td
class="forklaringTekst">Andre har bestillt hytten</td></tr>
                        <tr><td class="FerieFarge"></td><td
class="forklaringTekst">Ferie</td></tr>
                        <tr><td class="SperretFarge"></td><td
class="forklaringTekst">Hytten er sperret for utleie</td></tr>
                        </table>

                        <br />

                        <c:forEach items="${ferier}" var="ferie" begin="0"
step="1">
                        <div>${ferie}</div>
                        </c:forEach>

                        </div>

                        <br class="clearFloat" />

                        <c:choose>
                              <c:when test="${visbestill == 'true'}">
                              <form id="bestillingBruker" action="Bestilling?id=
${hytte.navn }&action=postnew" method="post">
```

```
                                          <p><label><b>Ditt HIB brukernavn: </
b></label><input type="text" name="brukernavn"/>
                                          <input type="submit" name="bestill"
value="Bestill" /> <font color=red>${feilinput}</font></p>
                                          <p>Du vil fÃ¥ en oversikt over hva du
har bestilt, og mÃ¥ deretter
                                          bekrefte din bestilling. </p>
                                    </form>
                              </c:when>
                              <c:otherwise>
                                    <p><font color=red>${feilinput}</font></p>
                              </c:otherwise>
                        </c:choose>

                  </div></div>
                  <div id="hoyre"><jsp:include page="hoyreMeny.jsp" /></div>
                  <br class="clearFloat" />
            </div>
            <div id="bunnDiv"><p>Siden er utviklet for HÃ¸gskulen i Bergen</p></
div>

      </div>

</body>
</html>
```

adminInnstillinger.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>HiB HytteWeb Admin</title>
<link href="hovedside.css" rel="stylesheet" type="text/css" />
<link href="./form.css" rel="stylesheet" type="text/css" />
</head>

<body>

      <div id="mainDiv">
            <div id="toppDiv"></div>
            <div id="centerDiv">
                  <div id="venstre"><jsp:include page="adminVenstreMeny.jsp" /
></div>
                  <div id="midt"><div id="centerMargin">

                  <!-- Endre her...  -->
                        <p class="feilmelding"><i>${melding }</i></p>
                        <h3 class="centerOverskrift">Innstillinger</h3>
                        <p>Her kan du sette ulike innstillinger for
applikasjonen.</p>

                        <form action="Innstillinger" method="post">
                              <br/><h3
class="underoverskrift">Karanteneregler:</h3>
                              <table>
                                    <tr>
                                          <td>Karantene for ferie (Ã¥r)</td>
                                          <td>Karantene for periode (uker)</td>
                                    </tr>
```

```
                                    <tr>
                                          <td>
                                    <select name="karanteneferier" id="aar">
                                          <c:choose>
                                                <c:when test="${karFerie == 1 }">
                                                <option value="1"
selected="selected">1</option>
                                                </c:when>
                                                <c:otherwise>
                                                      <option value="1">1</option>
                                                </c:otherwise>
                                          </c:choose>
                                          <c:choose>
                                                <c:when test="${karFerie == 2 }">
                                                <option value="2"
selected="selected">2</option>
                                                </c:when>
                                                <c:otherwise>
                                                      <option value="2">2</option>
                                                </c:otherwise>
                                          </c:choose>
                                          <c:choose>
                                                <c:when test="${karFerie == 3 }">
                                                <option value="3"
selected="selected">3</option>
                                                </c:when>
                                                <c:otherwise>
                                                      <option value="3">3</option>
                                                </c:otherwise>
                                          </c:choose>
                                          <c:choose>
                                                <c:when test="${karFerie == 4 }">
                                                <option value="4"
selected="selected">4</option>
                                                </c:when>
                                                <c:otherwise>
                                                      <option value="4">4</option>
                                                </c:otherwise>
                                          </c:choose>
                                          <c:choose>
                                                <c:when test="${karFerie == 5 }">
                                                <option value="5"
selected="selected">5</option>
                                                </c:when>
                                                <c:otherwise>
                                                      <option value="5">5</option>
                                                </c:otherwise>
                                          </c:choose>
                                    </select>
                                          </td>
                                          <td>
                                    <select name="karanteneperioder" id="aar">
                                          <c:choose>
                                                <c:when test="${karPeriode == 1 }">
                                                <option value="1"
selected="selected">1</option>
                                                </c:when>
                                                <c:otherwise>
                                                      <option value="1">1</option>
                                                </c:otherwise>
                                          </c:choose>
                                          <c:choose>
                                                <c:when test="${karPeriode == 2 }">
                                                <option value="2"
selected="selected">2</option>
                                                </c:when>
```

```
                                    <c:otherwise>
                                          <option value="2">2</option>
                                    </c:otherwise>
                              </c:choose>
                              <c:choose>
                                    <c:when test="${karPeriode == 3 }">
                                    <option value="3"
selected="selected">3</option>
                                    </c:when>
                                    <c:otherwise>
                                          <option value="3">3</option>
                                    </c:otherwise>
                              </c:choose>
                              <c:choose>
                                    <c:when test="${karPeriode == 4 }">
                                    <option value="4"
selected="selected">4</option>
                                    </c:when>
                                    <c:otherwise>
                                          <option value="4">4</option>
                                    </c:otherwise>
                              </c:choose>
                              <c:choose>
                                    <c:when test="${karPeriode == 5 }">
                                    <option value="5"
selected="selected">5</option>
                                    </c:when>
                                    <c:otherwise>
                                          <option value="5">5</option>
                                    </c:otherwise>
                              </c:choose>
                              <c:choose>
                                    <c:when test="${karPeriode == 6 }">
                                    <option value="6"
selected="selected">6</option>
                                    </c:when>
                                    <c:otherwise>
                                          <option value="6">6</option>
                                    </c:otherwise>
                              </c:choose>
                              <c:choose>
                                    <c:when test="${karPeriode == 7 }">
                                    <option value="7"
selected="selected">7</option>
                                    </c:when>
                                    <c:otherwise>
                                          <option value="7">7</option>
                                    </c:otherwise>
                              </c:choose>
                              <c:choose>
                                    <c:when test="${karPeriode == 8 }">
                                    <option value="8"
selected="selected">8</option>
                                    </c:when>
                                    <c:otherwise>
                                          <option value="8">8</option>
                                    </c:otherwise>
                              </c:choose>
                        </select>
                                    </td>
                              </tr>
                        </table>
                        <input type="submit" value="Lagre innstillinger"
id="sendbutton" name="lagreKarantener" />
                  </form>
                  <form action="Innstillinger" method="post">
```

```
                              <br/><br/><h3 class="underoverskrift">Frister:</
h3>
                              <table>
                                    <tr>
                                          <td>Frist for avbestilling perioder
(uker)</td>
                                          <td>Frist for avbestilling enkeltdager
(uker)</td>
                                    </tr>
                                    <tr>
                                          <td>
                              <select name="fristerperioder" id="aar">
                                    <c:choose>
                                          <c:when test="${avbPeriode == 1 }">
                                          <option value="1"
selected="selected">1</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="1">1</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbPeriode == 2 }">
                                          <option value="2"
selected="selected">2</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="2">2</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbPeriode == 3 }">
                                          <option value="3"
selected="selected">3</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="3">3</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbPeriode == 4 }">
                                          <option value="4"
selected="selected">4</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="4">4</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbPeriode == 5 }">
                                          <option value="5"
selected="selected">5</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="5">5</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbPeriode == 6 }">
                                          <option value="6"
selected="selected">6</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="6">6</option>
                                          </c:otherwise>
                                    </c:choose>
```

```
                                    <c:choose>
                                          <c:when test="${avbPeriode == 7 }">
                                          <option value="7"
selected="selected">7</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="7">7</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbPeriode == 8 }">
                                          <option value="8"
selected="selected">8</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="8">8</option>
                                          </c:otherwise>
                                    </c:choose>
                              </select>
                                          </td>
                                          <td>
                              <select name="fristerenkeltdager" id="aar">
                                    <c:choose>
                                          <c:when test="${avbDag == 1 }">
                                          <option value="1"
selected="selected">1</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="1">1</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbDag == 2 }">
                                          <option value="2"
selected="selected">2</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="2">2</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbDag == 3 }">
                                          <option value="3"
selected="selected">3</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="3">3</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbDag == 4 }">
                                          <option value="4"
selected="selected">4</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="4">4</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbDag == 5 }">
                                          <option value="5"
selected="selected">5</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="5">5</option>
                                          </c:otherwise>
```

```
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbDag == 6 }">
                                          <option value="6"
selected="selected">6</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="6">6</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbDag == 7 }">
                                          <option value="7"
selected="selected">7</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="7">7</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${avbDag == 8 }">
                                          <option value="8"
selected="selected">8</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="8">8</option>
                                          </c:otherwise>
                                    </c:choose>
                              </select>
                                          </td>
                                    </tr>
                              </table>
                              <input type="submit" value="Lagre innstillinger"
id="sendbutton" name="lagreFrister" />
                        </form>
                        <form action="Innstillinger" method="post">
                              <br/><h3
class="underoverskrift">Epostinnstillinger:</h3>
                              <table>
                                    <tr>
                                          <td>Epostutsendelse</td>
                                          <td>Bruk epost-rot</td>
                                    </tr>
                                    <tr>
                                          <td>
                              <select name="epostType" id="type">
                                    <c:choose>
                                          <c:when test="${epostType ==
'gmail' }">
                                          <option value="gmail"
selected="selected">Gmail</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="gmail">Gmail</
option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${epostType == 'hib' }">
                                          <option value="hib"
selected="selected">HiB</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="hib">HiB</option>
                                          </c:otherwise>
                                    </c:choose>
```

```
                                    <c:choose>
                                          <c:when test="${epostType ==
'local' }">
                                          <option value="local"
selected="selected">Localhost</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option
value="local">Localhost</option>
                                          </c:otherwise>
                                    </c:choose>
                              </select>
                                    </td>
                                    <td>
                              <select name="brukEpostRot" id="bool">
                                    <c:choose>
                                          <c:when test="${brukEpostRot ==
true }">
                                          <option value="true"
selected="selected">Ja</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="true">Ja</option>
                                          </c:otherwise>
                                    </c:choose>
                                    <c:choose>
                                          <c:when test="${brukEpostRot ==
false }">
                                          <option value="false"
selected="selected">Nei</option>
                                          </c:when>
                                          <c:otherwise>
                                                <option value="false">Nei</
option>
                                          </c:otherwise>
                                    </c:choose>
                              </select>
                                          </td>
                                    </tr>
                              </table>
                              <br><br><br>
                              <table>
                                    <tr>
                                          <td><p>Epost til lÃ¸nning</p></td>
                                          <td><p>Epost til admin</p></td>
                                    </tr>
                                    <tr>
                                          <td>
                                                <p><input type="text"
name="epostLonn" value="${epostLonn }"/></p>
                                          </td>
                                          <td>
                                                <p><input type="text"
name="epostAdmin" value="${epostAdmin }"/></p>
                                          </td>
                                    </tr>
                              </table>
                              <input type="submit" value="Lagre innstillinger"
id="sendbutton" name="lagreEpost" />
                        </form>

                  <!-- Endre her...  -->

                  </div></div>
                  <div id="hoyre"><jsp:include page="../hoyreMeny.jsp" /></div>
                  <br class="clearFloat" />
```

```
            </div>
            <div id="bunnDiv"><p>Siden er utviklet for HÃ¸gskulen i Bergen</p></
div>
      </div>

</body>
</html>
```