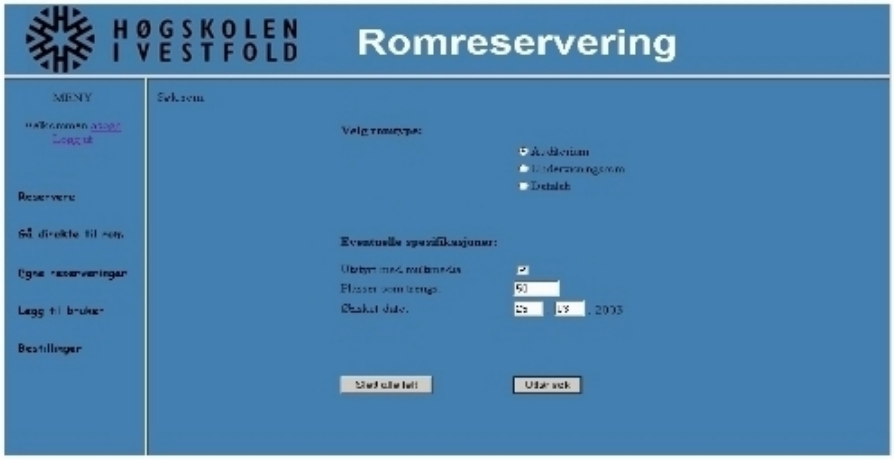
HØGSKOLEN I VESTFOLD
Avdeling for realfag og ingeniørutdanning

Hovedoppgave 2003

Romreserveringsprogram



Utarbeidet av Jørgen Wang Svendsen
Atle Sogn
Terje Havsø

Forord

Denne rapporten er en dokumentasjon på vår hovedprosjektoppgave 2003, som er gitt oss i oppgave av Høgskolen i Vestfold. Rapporten er i hovedsak beregnet på sensor, veileder, timeplanleggere her ved høgskolen og de som kommer til å videreutvikle programmet ved en senere anledning.

Rapporten viser hvordan vårt program er blitt og hva som kan forbedres.

Horten, 06.06.03

Atle Sogn , Jørgen Wang Svendsen , Terje Havsø

Innhold

Innledning

Dette prosjektet ble satt i gang av høgskolen her i Vestfold. Rutinene omkring reservering av rom her ved skolen var altfor gammeldagse, og ikke minst tungvinte. De ønsket seg derfor et nettbasert program som kunne håndtere romreserveringer på en enklere og arbeidsbesparende måte. Programmet skulle bygge på data fra GPUntis, og ha mulighet for eksterne reserveringer utover disse. GPUntis er timeplanprogrammet skolen benytter seg av til å timeplanlegging.

Oppgaven innebar håndtering av database, og opprettelse av en dynamisk hjemmeside. Vi fant tidlig ut at php, mysql og apache var det perfekte trekløver for denne typen web-side. Php er ikke pensum på dataingeniørstudiet, så dette var helt nytt for oss, men erfaring med c fra før, gjorde at dette gikk greit å sette seg inn i. Databaser og spørringer var stoff vi hadde vært innom i dba50-timene i 2.klasse.

Det første vi gjorde var å sette oss inn i problemstillingene, og forsøke å se for oss hvordan vi skulle løse disse problemene. Skolen hadde i utgangspunktet ingen rammer på oppgaven, men i samråd med skolen, ved Maija Ånestad og Helge Herheim, kom vi fram til at dette var elementer som var ønsket i programmet vi skulle lage:

- Innlogging
- Søke etter ledig rom til ønsket dato
- Reservere ønsket rom
- Slette en foretatt reservering
- Begrense antallet personer som har lov til å reservere
- Unngå dobbelbooking av timer
- Godkjenne bestilling av time(r)
- Utskrift av foretatte reserveringer
- Tilbakemelding til den som reserver om godkjent/avslått
- Oppdatering av timeplan

Disse punktene, samt at vi skulle jobbe utifra filer som blir lagd i GPUntis, ble utgangspunktet for oppgaven vår. Programmet har store utviklingsmuligheter utover det vi har fått til på denne tiden som hovedprosjektet har vart, noe vi kommer tilbake til i et senere avsnitt.

Vi satte også som en forutsetning at websiden skulle være enkel, lett forståelig og brukervennlig.

Programvare

Henviser til vedlagt Cd-rom for informasjon om installering av programvare.

Php er gratis scriptspråk som har hentet syntaksen fra c, java og Perl, og er i hovedsak er til å lage dynamiske websider og andre web-applikasjoner. En dynamisk side er en interaktiv side, som kan tilpasses hver enkelt bruker. Php støtter mange forskjellige databaser, som Oracle, Sybase, MySql og ODBC, og det er enkelt å integrere html-kode. Også integrasjonen av eksterne bibliotek gjør at man kan gjøre mye med php, som for eksempel å generere PDF-dokumenter, skrive XML og lage gif-/jpeg-bilder. Koden behøver ikke å kompilereres først, men legges rett ut på nett. Det er snakk om å lage en php-kompilerer, men det er først og fremst for å distribuere php-programmer som andre folk ikke kan endre på. Php er også veldig bra fordi det kan operere på mange plattformer. Fungerer godt på både unix og windows, og til sammen kan det kjøres på 25 forskjellige plattformer, og da vises kodingen helt likt på alle sammen!

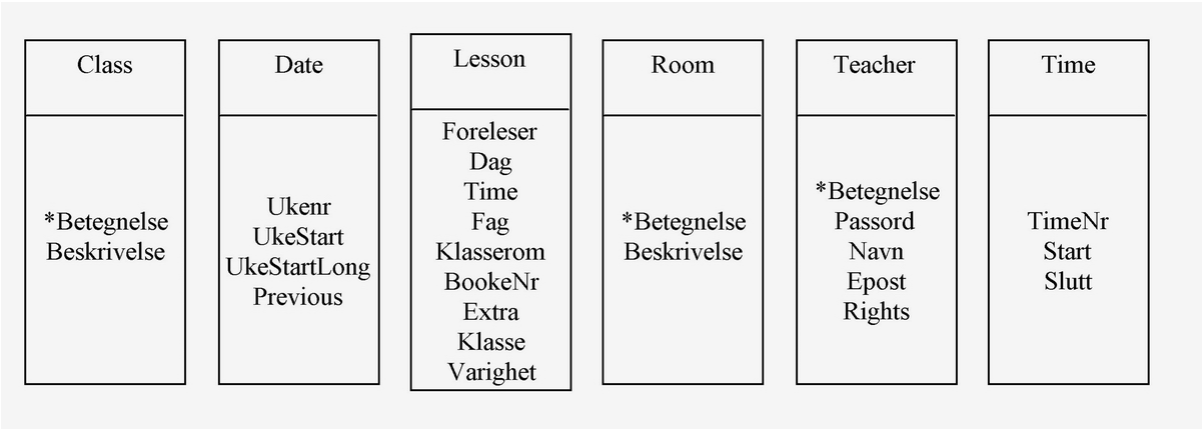
Php har eksistert siden 1994, men først etter at Php 3 ble utgitt i juni 1998, har det blitt allment kjent som et meget bra scriptspråk. Mange store firmaer foretrekker i dag å benytte php til sine websider, bl.a. Mitsubishi, Ericsson og NASA. Gode grunner, utenom de ovennevnte, til å benytte php, er at det går fort å gjøre oppdateringer på feil/mangler i dette språket og det fungerer på alle eksisterende internettservere, som apache, Microsoft og Netscape.

I april 1995 utga Apache Group den første versjonen av web-/http-serveren Apache. Apache Group er et samarbeid mellom frivillige individuelle programutviklere. Av den grunn er Apache, i likhet med php, et gratis produkt, og kildekoden er open-source, altså tilgjengelig for alle. Apache bygger på NCAS webserver og er videreutviklet ved hjelp av samling av informasjon via internett. Per i dag driver Apache godt over halvparten av hele verdens nettsteder. Til sammenligning har Microsoft sin web-server bare en andel på ca 20 %. Versjon 2.0.46 er i dag den beste versjonen og kan benyttes både på Windows og Linux. Apache web-server er spesielt mye brukt fordi den er rask, har god driftsikkerhet, fritt tilgjengelig, og ikke minst fordi den som nevnt er gratis.

MySQL er utviklet av Michael Widenius som jobber ved det svenske firmaet TcX. Widenius laget i 1979 et databaseverktøy UNIREG. Siden den gang er UNIREGskrevet om mange ganger. I 1994 begynte TcX å utvikle webbaserte applikasjoner. Ved å benytte en API som var identisk med MSQL(av David Hughes) kunne mange som hadde kode for mSQL lett flytte den til MySQL.

Oppbygning av databasen

Timeplanene blir fra skolens side lagt inn i timeplanprogrammet GPUntis. Som vi kommer nærmere inn på i konklusjonen er dette ingen database, så det går ikke å jobbe direkte mot disse dataene, men vha. modulen Info-Timeplan kan man eksportere dataene. Da får man ut 7stk .txt-filer. Vi har lagd et script, createTables.txt, som kjøres med kommandoen ”mysql -u localhost -p romreservering < createtables.txt" fra der mysql.exe ligger. Denne kjøres bare første gang, da tabellene i databasen skal opprettes og dataene skal lastes inn. Etter å ha opprettet databasen og lastet inn data for første gang, består databasen Romreservering av tabellene Room, teacher, lesson, time, student, date og class:



Forklaring til tabellene:

---Class har info om hvilke klasser som er med på timeplanen, og betegnelsen for hver klasse.

---Date består av ukenumrene som er tatt med i eksportering. Når man skal eksportere fra GPUntis er startdato den dagen man eksporterer, og man velger for hvor mange uker eksporten skal gjelde. UkeStart er første dato, altså mandag, i hver enkelt uke skrevet på formen ”måned.dag.”.Uke-StartLong er tilsvarende dato, men skrevet på formen”ÅrMånedDag”. Previous er forrige ukenummer, utifra hvert av ukenumrene.

---Lesson er den viktigste tabellen, og denne har info om når timene er. På dataene som legges inn direkte fra GPUntis står det registrert foreleser, klasse, klasserom og fag til hver enkelt time på hvilken dag. I tillegg er ”Varighet” en tekst-streng som består av 53 tegn(antall uker i året), som forteller for hvilke uker denne reserveringen gjelder. ’1’ betyr at reservasjonen gjelder for denne uken,’0’ betyr at den ikke gjelder for denne uken, men ’-’ betyr at denne uken er utenfor den terminen/semesteret som disse dataene er fra. BookeNr og extra er felter som inneholder data, men som vi ikke har sett nødvendigheten i å benytte oss av.
Når det så legges inn eksterne reserveringer, registreres innloggingsnavn/brukernavn som foreleser, dag og rom registreres for hver time , fag blir satt til ’bestilt’ og klasse settes til ’ekstern’. I tillegg lages det en streng som viser for hvilken uke reserveringen

gjelder som legges under varighet. Feltene bookeNr og extra settes bare til ’9999’ og ’0’.

---Room har info om hvilke rom som er med på timeplanen, og betegnelse for hver enkelt.

---Teacher inneholder i utgangspunktet info om alle lærerbetegnelsene og deres fulle navn. Her har vi lagt til feltene ’epost’, ’passord’ og ’rights’. Dette for å kunne bruke den mot innloggingen.

---Time har vi også endret litt på slik at den passer til vårt bruk. Den består av timenummer fra 1 til 13 og når timene starter og slutter.

---Student inneholder ingenting etter eksporteringen, og benyttes derfor aldri.

Oppdatering

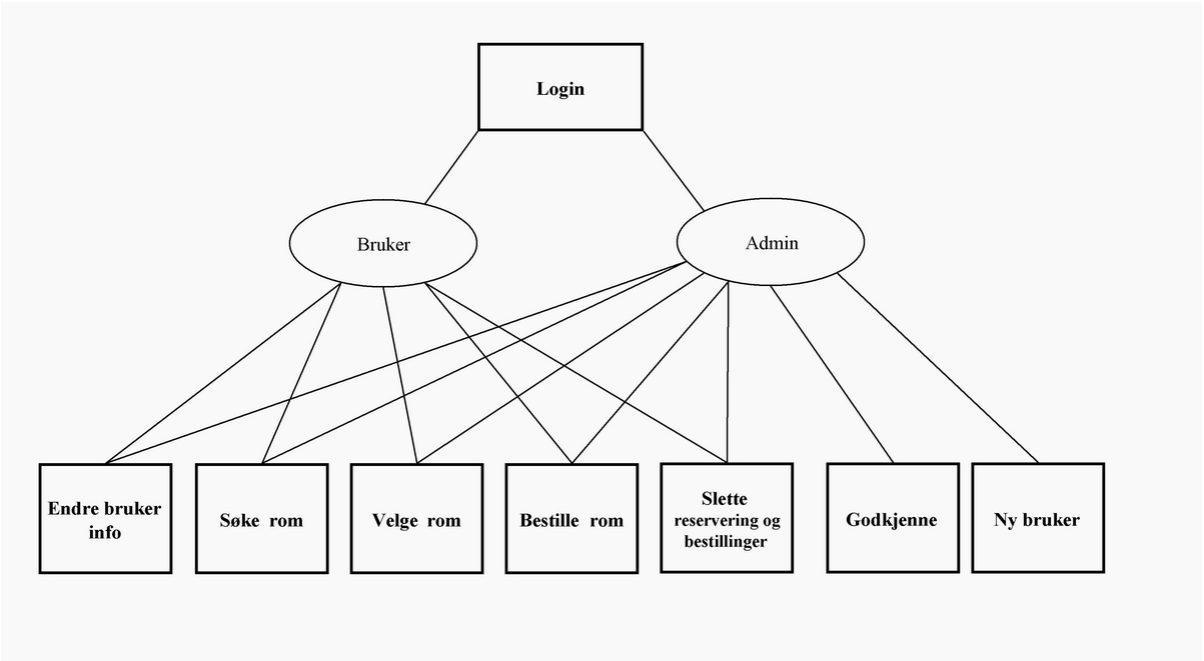
Når det så skal gjøres oppdateringer av timeplan, gjøres dette ved å kjøre ”mysql -u localhost -p romreservering < loaddata.txt" fra der mysql.exe ligger. Da legges de nye dataene inn i databasen. Det første scriptet er altså bare tiltenkt å benyttes èn gang, mens det andre scriptet, loadData.txt, kjøres hver gang man skal oppdatere dataene i databasen.

For å oppdatere benytter man Info-Timeplan i GPUntis og eksporterer til en midlertidig eksportmappe. Derfra flytter man over KUN lesson.txt og time.txt over i mappen ’Romreservering’. Grunnen til at bare disse to skal fornyes, er at ikke alle tabellene behøves oppdatert, da det sjelden eller aldri er store forandringer på disse i løpet av et år.

Innlogging

Begrense antallet som kan bestille

Det var ønskelig fra skolens side å begrense antall personer som kan gå inn å bestille rom. Vi har valgt å begrense det til de som er registrert i fila teacher.txt som blir generert utifra GPUntis, altså i hovedsak skolens forelesere. Det kan enkelt utvides til også å gjelde den vanlige elev ved skolen. De som nå er registrert i teacher.txt blir gitt et rettighetstall og et passord, som de selv kan gå inn å endre i ettertid.



Programmets innlogging

Vi har valgt å ha med innlogging i programmet vårt, av den grunn at ikke hvem som helst skal kunne gå inn å booke rom, og at vi ved å ha innlogging også lett kan identifisere hvilken bruker som har hvilken booking. Innloggingen bygger på tabellen Teacher i databasen vår. I utgangspunktet inneholder teacher.txt-fila som kommer fra GPUntis bare betegnelse og fullt navn, men vi har lagt til passord, epost og rights, slik at den er egnet som en enkel innloggingsdatabase. Vi har foreløpig valgt å definere to typer brukere, 0 er rettighetstallet for den vanlige bruker, mens administrator(Maija) har rettighetstallet 1. Dette for å kunne gi flere muligheter til administrator enn de andre brukerne. En bruker logger seg inn og booker et rom, bestillingen blir da videresendt administrator som så må godkjenne eller avslå bestillingen. Andre muligheter som skal tilfalle kun administrator er foreløpig å kunne legge til nye brukere. Den vanlige bruker skal i hovedsak bare ha mulighet til å søke på rom, reservere ønskede rom, og eventuelt slette egne bestillinger, samt å kunne endre sin egen brukerinfo.

Innloggingsscriptet som vi bruker i dette programmet er hentet fra www.minlilleverden.net. Med godkjennelse fra Espen Nilsen, administrator for siden, har vi valgt å bruke dette scriptet, fordi ingen på gruppa har spesielt god kjennskap til

innlogging og sikkerhet. Dette scriptet skal være meget bra og sikkert, og passordet blir kryptert med Crypt(). Det er nok fullt mulig å lage en sikrere innlogging, men vi har valgt å ikke prioritere denne delen så veldig, men trur likevel at vi har funnet en dugelig løsning. Ved innlogging sjekker scriptet til enhver tid mot databasen om brukernavn og passord er korrekt, og alle sidene krever at man MÅ være innlogget for å få tilgang. Innloggingen lagrer også input-verdier i sessions. Sessions er globale variabler som lagres ved innlogging, slik at brukeren slipper å taste inn dette flere ganger, og de er veldig praktiske å benytte i programmeringen.

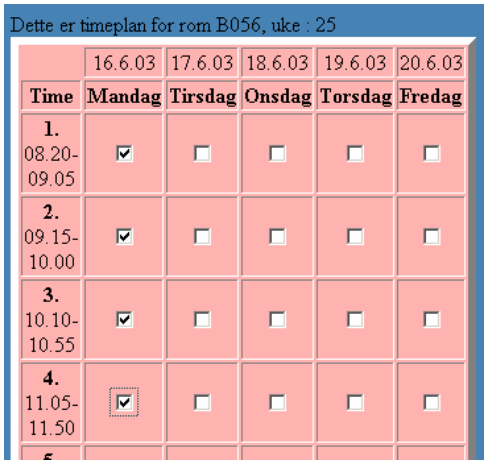
Crypt() har definisjonen string crypt(string input_string[, string salt]) og returnerer en kryptert streng ved å bruke standard Unix DES-basert krypteringsalgoritmer. Argumentene som trengs er en tekststreng som skal krypteres, og en valgfri salt string til å basere krypteringen på. Hvis det argumentet ikke er med, auto-genererer PHP et to-tegns salt. PHP setter en konstant med navnet Crypt_salt_length som forteller deg hvorvidt et to-tegns SALT fungerer på din maskin, eller om et tolv-tegns salt (MD5-basert) er fungerende. Crypt() støtter fire typer algoritmer, men to-tegns og tolv-tegns er de mest vanlige.
Crypt() er et enveis-krypteringsspråk, det vil si at det krypterer passordet en vei, men ikke tilbake igjen. Så når en person skal logge seg inn i programmet vårt, blir passordet vedkommende taster inn kryptert og deretter sammenliknet med det lagrede, krypterte passordet som er registrert på denne brukeren.

Om programmet

Søke etter rom

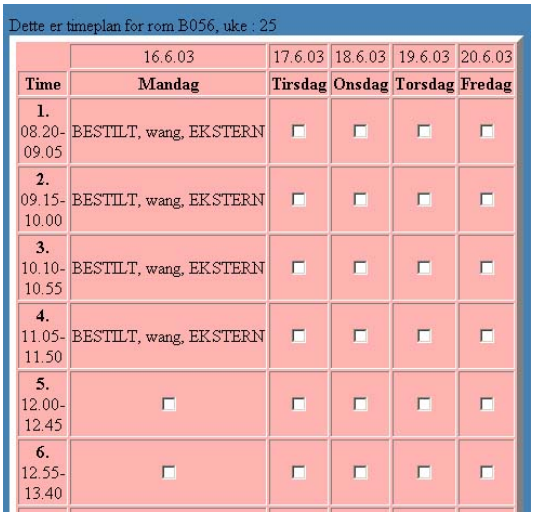
Å finne et rom som passer til det formålet brukeren trenger er helt essensielt for dette programmets brukervennlighet. Her er det satt opp tre valg som muligheter ved valg av romtype, auditorium, datalaboratorium og undervisningsrom. Ved å velge en av disse for så å søke vil man få opp en liste over de aktuelle rom. Man kan begrense det mer ved å velge antall plasser man trenger for at alle skal få plass. Under Auditorium kan man velge om man trenger multimedia utstyr, det vil si prosjektor, datamaskin og så videre. Om ønskelig kan også datoen for den aktuelle reservasjonen tastes inn. Datoen trenges egentlig ikke før seinere men vi følte at det ville være enklere for brukeren å ha all informasjonsinnsamlingen på et sted. Etter å ha tastet inn alle verdier man føler man trenger trykker man på utfør søk-knappen. Det vil føre til at man blir sendt videre til en ny side. Der kommer det opp en oversikt over rom som passer til søket. Hvert av rommene som blir oppgitt står oppført som en link. Denne linker til timeplanen for dette rommet, den uken som datoen man har oppgitt er i. Hvis det ikke ble oppgitt noen dato så vil vi bruke datoen den dagen søket blir utført.

Bestille



Det er viktig å ta hensyn til at det kan være forskjell i timeplanen fra uke til uke. Det er et problem den tidligere nettbaserte timeplanen ikke tok hensyn til. Der ble det ikke tatt hensyn til dynamikken i timeplanen. For at skolen skal få utnyttet sitt fulle potensial, måtte vi legge det opp slik at våre timeplaner plukket opp om det kom noen reserveringer som ikke gjaldt for en gitt uke. I tabellen lesson i vår database er det et felt som vi har valgt å kalle varighet. Her er det en streng som angir om en reservering gjelder for en gitt uke ved å bruke tegnet ”1” for reservert, ”0” for ledig og ”–” for at det er utenfor gjeldende semester. Ved å teste på dette feltet for den angitte uken fant vi frem til om reserveringen gjaldt for den gitte uken. Så hvis et rom er ledig en uke så kan det benyttes av andre. Administrator er eneste som har mulighet til å godkjenne/avslå bestillinger, men alle innloggede kan bestille timer. Da følger man søkeprosedyren som ble beskrevet tidligere. Når man har funnet et passende rom med ledige timer på den aktuelle dagen kan man merke de timene man vil bestille. Tid og

rom-navn er verdier som kommer fra søke-siden og velg rom-siden. Når brukeren har krysset av for de timene som han eller hun trenger, trykker man på knappen ”bestille”(se bilde). Da vil man havne i et script som dobbeltsjekker om det er noen reservasjoner for de aktuelle timene. Hvis time(e) er ledige, bruker vi brukernavn fra innloggingen til å sette hvem bestillingen er på, og resten av verdiene som er relevante vi fikk fra søke-siden. Fra rom_plan sender vi hvilken ukedag og hvilke time som skal lagres. Databasen er laget slik at det blir laget en rad for hvert rom for hver time, dag og uke. Hvis man for eksempel skal bestille første time mandag og første time tirsdag, er dette to rader i databasen, og hvis bestillingen er første og andre time på mandag, så er dette fortsatt to rader. En viktig sak når vi skal lage en bestilling er å lage en tekst streng som angir for hvilken uke reserveringen er. Dette blir gjort i en for-løkke som setter et ”1” tegn for den uken som bestillingen gjelder. Ukenummeret får vi fra tidsvariabelen vår. ”1”-tegnets plassering i tekststrengen angir for hvilken uke bestillinger gjelder. Nå bruker vi all informasjonen vi har samlet og laget til å sette inn en ny rad i tabellen. Etter dette vil timen(e) som har blitt bestilt komme opp når noen vil se på romplanen.(se bilde)


Dette er timeplan for rom B056, uke : 25

	16.6.03	17.6.03	18.6.03	19.6.03	20.6.03
Time	**Mandag**	**Tirsdag**	**Onsdag**	**Torsdag**	**Fredag**
1. 08.20-09.05	BESTILT, wang, EKSTERN	☐	☐	☐	☐
2. 09.15-10.00	BESTILT, wang, EKSTERN	☐	☐	☐	☐
3. 10.10-10.55	BESTILT, wang, EKSTERN	☐	☐	☐	☐
4. 11.05-11.50	BESTILT, wang, EKSTERN	☐	☐	☐	☐
5. 12.00-12.45	☐	☐	☐	☐	☐
6. 12.55-13.40	☐	☐	☐	☐	☐

Unngå dobbelbestilling

Det å unngå at timer blir reservert flere ganger av forskjellige personer er et veldig viktig tema. Hvis dette hadde skjedd ville det ført til sterke reaksjoner og oppgitthet for de involverte. De måtte da på kort tid finne et nytt rom som kunne blitt brukt og det er ikke alltid like enkelt. I vårt oversiktsbilde over ledige timer for et gitt rom, finnes det avkrysningsbokser hvor brukeren kan velge timer som personen vil reservere. Men når denne siden blir generert så kan det være en annen bruker som allerede har den siden opp og velger for reservasjon. Siden disse ikke har kommet opp på den andre brukerens side er det mulig for denne personen å velge de samme timene som den andre brukeren. Hvis dette skjer så er det en ekstra sjekk som ser etter om det finnes en reservering for det angitte rommet den angitte dagen, timen og uken. Hvis ikke denne testen gir et resultat så vil settes bestillingen inn i databasen. Hvis vi får resultat som sier at det allerede finnes en reservering, markerer vi en variabel med at det er en bestilling som ikke har gått gjennom. Vi har også to tellere som vi bruker i en utskrift, slik at vi får beskjed om det er noe som kolliderer. Da vil vi for eksempel se at tre timer har blitt reservert men at den fjerde ikke ble reservert. Etter 15 sekunder vil man bli sendt tilbake til siden med rom planen hvis man ikke har trykket på linken som sender deg tilbake til rom planen. Der vil man få den nye timeplanen med oppdateringen.

Her er vi litt usikre på hvor sikkert dette er. Hvis man for eksempel trykker tilnærmet samtidig og de forskjellige skriptene blir kjørt som forskjellige tråder, kan det hende at begge får resultat at det er greit å lagre verdien. Men her er vi nede på microsekund nivå, og med cirka tretti reserveringer i uken burde dette ikke være noe stort problem.

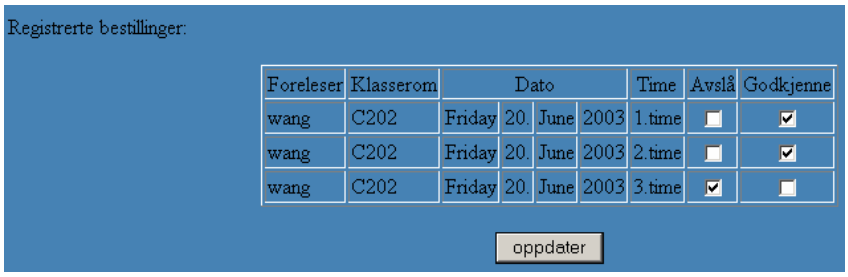
Slette bestillinger og reserveringer

Hvis behovet for timen bortfaller, finnes det en mulighet for å slette egne bestillinger. Dette begrenser også administrasjonen sitt arbeid, for da kan brukeren selv korrigere bestillingene sine. Brukeren får opp en liste over egne reserveringer, hvor det står om bestillingene er godkjent eller fortsatt bare til bestilling. Alle reserveringer vil komme frem i dato rekkefølge. Selv om det ikke er med dato i den faktiske reserveringen så vil brukeren her se hvilken dato reservasjonen er for. For hver time som er bestilt eller reservert er det en avkryssingsboks som man kan bruke til å slette bestillinger og reservasjoner med. Ved å merke av på riktig linje eller linjer vil bestillinger og reservasjoner bli slettet fra databasen. Det vil da ikke være mulig å få den tilbake utenom å bestille den på nytt. Siden som viser egne bestillinger vil bli oppdatert med en gang man trykker på slette knappen, så man får se resultatet med en gang.

Sett fra en mer teknisk synsvinkel, brukes det to script for å gjøre dette, ett for å vise egne reserveringer og ett for å gjøre den faktiske slettingen. For å slette riktig så trengs det en del informasjon slik at den sletter riktig. Vi mente det var for tungvint å hente alle informasjonen over til skriptet som sletter, så vi tok bare med hvilket nummer de var i resultatet for så å foreta en identisk spørring i det neste skriptet. Vi hadde da ikke tenkt så mye på multibruker-problemet, så om en administrator sletter en bestilling så vil indeksene bli forskjøvet. Dette vil føre til at bestillinger som ikke er huket av til sletting kan bli slettet. Dette ville blitt løst hvis slette-delen var i samme script som den som viser egne bestillinger(vis_egne.php).

Godkjenne bestillinger

For at reserveringene ikke skal komme ut av kontroll så vil administrasjonen at alle reserveringer skal gå gjennom dem. Dette er ordnet på den måten at når en bruker bestiller en time, legges ”bestilt” inn i feltet fag for reserveringen. Når administrasjonen skal godkjenne bestillinger bruker vi denne verdien for å finne alle ikke godkjente reserveringer. Ved å krysse av i avkryssingsbokser kan administratoren velge om bestillingen skal slettes eller godkjennes. Deretter blir disse feltene enten slettet eller oppdatert, ettersom hva som har blitt valgt av administratoren. Det blir også her brukt indekser for å slette og oppdatere. Når vi skal oppdatere velger vi posten med de riktige verdiene ved å hente disse fra resultatet av spørringen. Ved godkjennelse av bestilling, oppdateres fag-feltet for reserveringen til verdien ”reservert”. Neste gang vi ser på nye bestillinger, vil denne ikke være med på listen, siden den har blitt godkjent. Hos brukerne vil det stå at bestillingen er reservert i et status felt.



Tilbakemelding til bruker

Brukerne kan se status på sine egne bestillinger når de er logget inn. Likevel sendes det en e-post til bestiller, hvor vi samler opp alle bestillingene som har blitt godkjent eller slettet i denne omgang. Dette er fordi at de fleste sjekker e-posten sin ganske ofte, og at det er enklere å få beskjed der enn å logge seg inn i programmet for å sjekke status. Med e-post vil det antakelig gli lettere inn i den daglige rutinen. Hvis administratoren velger èn og èn bestilling, det vil si at kun èn time blir godkjent eller avslått avgangen, blir det sendt èn e-post for hver bestilling. Hvis derimot flere bestillinger godkjennes/avslås samtidig, vil det bli sendt en e-post for hver bruker som har fått sin bestilling eller bestillinger behandlet.

Om koden

Sok_rom.php

Denne siden er laget for å hente inn informasjon om hva slags rom brukeren trenger. Det er tre forskjellige kategorier. Auditorium, undervisningsrom og datalaboratorium. Auditoriet kan ha multimedia utstyr. Man kan også spesifisere antall plasser og hvilken dato rommet skal reserveres for. Det tre romtypene velges vha ”radio”-knapper. Det betyr at man bare kan velge en av dem av gangen. Det er en avkrysningsboks for å velge multimedia utstyr og tekstfelt for antall plasser og dato. All informasjonen som blir samlet inn her blir sendt videre når man trykker på en knapp med teksten ”Utfør søk”. Ved å trykke på denne knappen vil man sende all informasjonen og bli sendt videre til siden som bruker disse dataene til å finne rom som passer til søket. Alt som er av interesse her er lagd i html, men siden programmet skal skjermes for uvedkommende, trenger vi noen php-funksjonaliteter, slik at vi kan sjekke om den som prøver å benytte siden er en gyldig bruker.

Direkte.php

Hvis en bruker vet hvilket rom han eller hun trenger, så kan det være irriterende å måtte gå gjennom hele søkeprosessen hver gang. Derfor har vi laget et script som finner frem til det rommet man vil ha med en gang. Det er bare å taste inn betegnelsen på rommet i et tekstfelt. Betegnelse er for eksempel ”C202”. For å komme til riktig rom plan er det også to felt til, disse brukes til å taste inn dag og måned. Ved hjelp av disse dataene kan vi nå produsere en link til riktig rom til riktig tid. På siden, under alle valgene, er det en knapp hvor det står ”Utfør søk”. Når denne knappen blir trykt eller det blir trykket enter i tekstboksen hvor vi angir betegnelsen på rommet, så vil søket bli utført.

```
SELECT * FROM room WHERE betegnelse LIKE '${velg_rom}';
```

Her finner vi rommet som tilsvarer betegnelsen som brukeren har tastet inn og som ligger i variabelen $velg_rom. Nå vil vi lage en link som tar med seg dato og romnavn inn i den neste siden(rom_plan.php). Vi vil så skrive ut betegnelsen på rommet som en link. Når linken blir trykket på vil man bli sendt videre til rom_plan.php-siden.

Finn_rom.php

Etter å ha fått bekreftet at dette er en reel bruker, så vil hoveddelen av scriptet settes i gang. Vi er nå koblet til databasen og skal hente de aktuelle rommene som brukeren er interessert i. Først må vi sjekke om brukeren har valgt noe i det hele tatt. Hvis vi ikke har fått inn noen data så må vi lage en dato og da bruker vi datoen for i dag. Datoen blir ikke brukt her men det virker mer oversiktlig hvis all informasjon blir samlet inn på en gang. Deretter legger vi inn verdien for romtypen brukeren skal ha, inn i en spørring. Resultatet av denne spørringen er det som vil bli skrevet ut, med få modifikasjoner. Disse kommer vi tilbake til senere. Spørringen ser ut som dette:

```
SELECT * FROM room WHERE beskrivelse LIKE '%${pc}%' AND beskrivelse
LIKE '%${rom}%';
```

Dette betyr at vi skal ha ut alle feltene i tabellen room, der hvor tekst strengen beskrivelse innholder ordet PC. Dette betyr at rommet har multimedia utstyr. Hvis ikke multimedia utstyr er valgt på søkesiden så vil ikke denne variabelen ha noen verdi, og denne variabelen heller ikke forårsake noen begrensninger. Feltet beskrivelse må også innholde tekststrengen som angir om hva slags rom bruker trenger. Det er tre kategorier. Auditorium, undervisningsrom og datalaboratorium. Variabelen rom innholder den verdien som i tabellen angir hva slags rom det er. Det resultatet som vi får av denne spørringen evaluerer vi videre. I feltet ”betegnelse” ligger også antall plasser. Her må vi parse tekstfeltet for å trekke ut antall plasser for så å sjekke denne verdien mot den verdien brukeren tastet inn under innsamlingen av informasjon tidligere. Hvis denne verdien er større enn den verdien som blir tastet inn så vil vi skrive ut denne siden, ellers så hopper vi over den. Når rommet blir skrevet ut generer vi en url. Denne innholder navnet på rommet og tiden for den aktuelle reserveringen. Ved å trykke på linken med rom nummeret, så vil vi komme til rom_plan.php med de aktuelle verdier.

Rom_plan.php

Her bruker vi verdiene vi får inn til å lage en spørring som returnerer verdien for hver time. Det blir brukt en dobbel løkke, en som teller dager og en som teller timer. For hver time hver dag har vi en spørring som ser ut som dette:

```
SELECT lesson.fag, lesson.foreleser, lesson.klasse, lesson.varighet FROM lesson
WHERE klasserom='${rom}' AND dag=${dagteller} AND time=${timeteller};
```

Her har vi angitt hvilke verdier vi vil ha ut fra tabellen lesson. Vi skal bare ha med de verdiene som har det klasseromnavnet vi alt valgte på forrige side. Den skal også ha riktig time og riktig dag. Resultat fra denne spørringen blir skrevet ut, men det må gjøres en evaluering av feltet varighet. Ved å bruke datoen vi får inn, finner vi frem til ukenummeret for den aktuelle uken. Hvis tegnet i tekststrengen som tilsvarer den aktuelle uken i er ”1”, skrives den ut på timeplanen, hvis ikke kommer det opp en ”checkbox”, fordi da er ingen av resultatene er gyldige for den spesifiserte uken. Hvis vi ikke får noen verdier tilbake fra spørringen så lages det en ”checkbox”. En ”checkbox” er en avkryssingsboks som man kan velge å markere hvis denne timen er aktuell å reservere. Siden vi skriver ut for hele uka så kan det hende at dagen allerede har vært. Derfor vil avkryssingsboksen bli gjort umulig å bruke hvis dagen allerede har vært, ved at den blir grået ut. Når resultatene for hver dag og hver time er skrevet ut, lages det to linker som blir plassert i bunnen. Disse gir muligheten til å se planen for neste uke eller uken før. I bunnen av siden er det også en knapp som sender alle de timene som har blitt markert til et nytt skript som reserverer disse timene.

Registrer.php

Nå har vi samlet inn nok innformasjon til å lage en reservasjon og det er det dette scriptet gjør. Først deler vi opp verdien eller verdiene vi får fra avkryssingsboksene. Disse innholder dag(er) og time(r) som skal reserveres. Dette gjøres ved å parse, det vil si dele opp med hensyn på et tegn. På grunn av måten rader er lagret i tabellen lesson må vi lage en tekststreng som angir hvilken uke reserveringen er for. Dette gjøres ved å bruke en løkke som setter den til ikke brukt når en teller er ulik uke nummer som vi får fra dato. Når uke nummer er likt med telleren så setter den at den er opptatt. Deretter, for å dobbeltsjekke om det er noen som har laget en reservasjon mellom det tidspunktet vi fikk generert siden og vi trykket lagre, sjekker vi med en spørring om det er noen reservasjoner der. Spørringen ser slik ut.

```
SELECT varighet FROM lesson WHERE dag=${dag} AND time=${time} AND
klasserom LIKE '${rom}' AND varighet LIKE '${search_varighet}';
```

Her prøver vi å se om det er noe er lagret på den dagen, på den timen på det rommet på den uken. Hvis dette gir tilbake 0 rader, så er det ingen reservasjon ved dette rommet for denne dagen, timen og uka. Hvis det blir returnert noe annet enn 0 rader, så vil det ikke lages noen reservasjon for denne kombinasjonen av rom, dag, time og uke. Får vi 0 rader lager vi den nye reservasjonen. Det gjøres med denne SQL kommandoen:

```
INSERT INTO lesson (foreleser, dag, time, fag, klasserom, bookenr, extra, klasse,
varighet) VALUES ('$brukernavn','$dag','$time','BESTILT','$rom','9999','0',
'EKSTERN', '$varighet');
```

Her velger vi at den skal settes inn i tabellen lesson og angir hvilke felter som skal settes inn. Verdiene i klammen bak VALUES angir hvilken verdi som skal inn i de respektive feltene som er angitt over. Alt dette gjøres for hver dag-time-kombinasjon vi har valgt i siden rom_plan.php. Når alle reservasjonene er lagret så blir man automatisk sendt tilbake til rom_plan.php siden. Når dette skjer så vil siden bli oppdatert med de nye reservasjonene. Brukeren vil aldri se noe av arbeidet som blir gjort i dette skriptet. For brukeren så vil det bare se ut som rom_plan siden blir oppdatert.

Vis_egne.php

Vis egne bestillinger har to funksjoner. Den ene er at man skal ha oversikt over sine egne reservasjoner. Den andre er at man kan slette sine egne reservasjoner hvis man må flytte dem eller at det av andre grunner ikke trenger reservasjonen lenger. For å finne frem de forskjellige reserveringene foretar vi en spørring som ser slik ut:

```
SELECT lesson.dag, lesson.time, lesson.klasserom, lesson.varighet, lesson.fag FROM
lesson WHERE foreleser='${brukernavn}' AND fag LIKE 'BESTILT' OR
foreleser='${brukernavn}' AND fag LIKE 'RESERVERT' ORDER BY varighet
DESC, dag, time;
```

Denne spørringen finner alle bestillinger/reserveringer som en lærer har gjort utenom den vanlige timeplanen, eller sagt med andre ord, vedkommendes eksterne bestillinger/reserveringer. Brukernavnet vi bruker for å finne egne bestillinger er hentet fra sessionen som har lagret dette. Resultatet av denne spørringen er ordnet etter tid. Selv om vi ikke har en verdi som er dato så bruker vi verdiene i varighet dag og time. Varighet er ordnet etter motsatt rekkefølge fordi verdien i feltet gjør at det ellers hadde kommet opp med den som var lengst fram i tid først. Nå skriver vi ut de forskjellige verdiene. For at det skal være enklest mulig for brukeren så finner vi fram datoen for reserveringen via en tabell i databasen. Den har oversikt over hvilken dag som er den første i hver uke. Ved å analyser tegnene i feltet varighet så finner vi hvilken uke det er. Slik ser spørringen ut:

```
SELECT ukestart FROM date WHERE ukenr=${j};
```

Ved hjelp av verdien vi får når vi deler opp tekststrengen som er resultatet av spørringen, så lager vi en dato med en funksjon som heter mktime(). Denne verdien er antall sekunder siden 1.januar 1970, ved å regne ut hvor mange sekunder det er i en dag og multiplisere det med hvilken dag i uken reservasjonen er, får vi en verdi som vi kan bruke for å formatere en utskrift som er litt vennligere for øye enn en uke nummer og uke dag. Vi lager også en avkryssingsboks slik at man kan avbestille timen. Vi skriver også ut hva status for timen er, om bestillingen er godkjent eller fortsatt til bestilling. I bunnen er det en knapp som, ved å trykke på den, vil slette de merkede bestillingene. Når brukeren trykker på denne knappen, så vil man starte slett.php fila og sende inn verdiene den trenger for å slette reservasjoner.

Slett.php

Her får vi inn nummeret på hvilken rad som skal bli slettet. Spørringen er da selvfølgelig helt lik den vi hadde på vis_egne.php-siden. Hvis dette ikke hadde stemt, hadde vi fått sletting både her og der. Vi har brukt data fra vis_egne-siden for å finne frem til hver enkelt rad som skal bli slettet.

```
SELECT lesson.dag, lesson.time, lesson.klasserom, lesson.varighet FROM lesson
WHERE foreleser='${brukernavn}' AND fag LIKE 'BESTILT' OR
foreleser='${brukernavn}' AND fag LIKE 'RESERVERT' ORDER BY varighet
DESC, dag, time;
```

Resultatet fra spørringen er lik den på forrige side(vis_egne.php). Dette gjør at vi kan velge ut radene vi skal slette vha disse verdiene. Denne SQL-kommandoen sletter de valgte radene:

```
DELETE FROM lesson WHERE foreleser LIKE '$brukernavn' AND klasserom LIKE
'${klasserom}' AND varighet LIKE '${varighet}' AND time=${time} AND
dag=${dag};
```

Her sletter vi hele raden fra database forutsatt at den er lik de verdiene vi har fått ut av fra spørringen tidligere. Når alt er gjort blir vi sendt tilbake til vis_egne.php som blir

oppdatert til å vise forandringene. Brukeren vil aldri se hva som skjer her inne, men vil bare oppfatte at siden blir oppdatert.

Vis_bestillinger.php

Denne siden er for at de som administrerer romreserveringen skal få muligheten til å godkjenne eller avslå bestillinger. For å få en liste over alle bestillinger bruker vi denne spørringen:

```
SELECT foreleser, klasserom, dag, time, varighet FROM lesson WHERE fag LIKE
'BESTILT' ORDER BY varighet DESC, dag, foreleser, time;
```

Den henter det som er vesentlig for at administratoren skal kunne avgjøre om bestillingen bør godkjennes eller om eden må bli avslått. Når vi lager en bestilling bruker vi feltet fag til å skrive bestilt. Dette bruker vi nå for å finne bestillinger. Vi ordner også alle bestillingene etter dato. Nå når vi har alle dataene, skriver vi de ut med unntak av tidsvariablene dag, time og varighet. Disse bruker vi i en ny spørring som finner frem til den første dagen i uka.

```
SELECT ukestart FROM date WHERE ukenr=${j};
```

Her er $j et tall som angir hvilken uke reservasjonen er for. Vi får her tilbake en tekststreng som vi deler opp og lager en dato verdi av. Datoer er angitt i antall sekunder siden 1. januar 1970. Denne verdien brukes så for å lage en mer brukervennlig utskrift. Vi har to avkryssingsbokser bakerst på hver reservering. Disse brukes av administrator til å godkjenne eller avslå reserveringer. Ved å hake av på en reservering så vil den indeksen bli sendt til et nytt php-script når administratoren trykker på en oppdater knapp i bunnen av siden.

Oppdater.php

Her starter vi med å forsikre oss om at det ikke er noe feil trykk. Siden vi bruker avkryssingsbokser i vis_bestillinger.php, så må vi sjekke om det ved en feiltagelse har blitt krysset av på begge valgene. Hvis dette er tilfellet ignorerer vi den raden. Deretter gjør vi den eksakt samme spørringen som vi gjorde i vis_bestillinger:

```
SELECT foreleser, klasserom, dag, time, varighet FROM lesson WHERE fag LIKE
'BESTILT' ORDER BY varighet DESC, dag, foreleser, time;
```

Nå vil vi få resultatet i samme rekkefølge som i spørringen på den forrige siden. Så vi trenger bare å bruke indekser isteden for å ta med alle dataene. Nå sletter vi de radene som er blitt valgt å fjernes med denne kommandoen:

```
DELETE FROM lesson WHERE fag LIKE 'BESTILT' AND foreleser LIKE
'$betegnelse' AND klasserom LIKE '$klasserom' AND dag=$dag AND time=$time
AND varighet LIKE '$varighet';
```

Dette vil slette den riktige verdien. Verdiene er hentet rett ut fra resultatet av spørringen. Nå blir dette lagt i en dynamisk tabell hvor navnet på læreren blir laget og en annen dynamisk tabell hvor data som sier hvilken time som er slettet blir tatt vare på. Når dette er gjort begynner vi å oppdatere verdiene som har blitt godkjent. Da bruker vi denne kommandoen:

```
UPDATE lesson SET fag='RESERVERT' WHERE foreleser LIKE '$betegnelse'
AND klasserom LIKE '$klasserom' AND dag=$dag AND time=$time AND varighet
LIKE '$varighet';
```

Verdiene hentes her fra samme spørring. Her blir verdien som stemmer med dataene fra spørringen oppdatert. Fag-feltet i tabellen lesson blir oppdatert fra bestilt til reservert. Det er dette feltet som blir brukt til å bestemme om reserveringen faktisk er godkjent. Også her lagrer vi navnet på foreleser og data om reserveringen i en dynamisk tabell. Vi tar nå dataene og sorterer dem slik at det bare er et data felt for hver foreleser. Dette er fordi at vi skal sende en e-post til foreleser slik at personen vet om reservasjonen har gått i orden eller ikke. Nå er vi klare til å sende e-posten. Vi bruker betegnelsen på foreleseren til å foreta en spørring mot databasen slik at vi finner e-postadressen, spørringen ser slik ut:

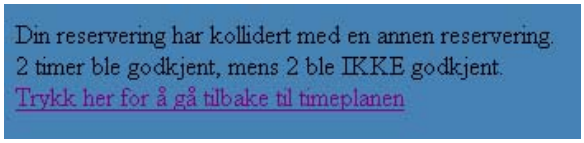
```
SELECT epost FROM teacher WHERE betegnelse LIKE '$ordna_foreleser[$i]'
```

Her slår vi opp i tabellen teacher og bruker foreleser som verdi. Nå når vi har e-postadressen kan vi ordne e-postadressen. Det blir nå sendt en e-post for hver bruker istedenfor for hver eneste reservasjon. Når alt dette er gjort blir man sendt tilbake til vis_bestillinger.php siden som oppdateres slik at slettede og godkjente timer ikke vil vises lenger.

Testing

Etter hvert som programdelene har blitt ferdig, har disse blitt prøvd ut. Dette har ført til at de største feilene i programmet har blitt plukket opp etter hvert. Siden de fleste punktene henger sammen, vil det bli testet på gamle ting for å se at nye ting fungerer. Vi har ikke fått veldig mange utenom om oss i gruppa til prøve ut programmet. Dette er litt synd, siden de muligens ville gjort ting vi ikke hadde tenkt på som kunne ført til nye problemer. Men på grunn av seint avslutnings tidspunkt er det vanskelig å få folk til å teste så nærme innleveringsdato. Vi har prøvd ut programmet med flere personer innlogget på en gang og dette har gått greit. Det har vært tre inne samtidig (det vil si alle på gruppa) uten noen problemer. Større trafikk har vi ikke fått testet. Vi har fått testet flerbrukerproblemer, med litt blandet resultat. Siden vi bruker hvilket nummer i lista som skal slettes, kan dataene i mellomtiden bli forandret og føre til feil i slettingen. Et annet flerbrukerproblem som vi har rukket å fikse, er tilfellet som oppstår når to personer ser på samme timeplan for samme rom. Det vil da komme opp en feilmelding til den som er for seint ute med å reservere hvis de reserverer samme timer. Under ser vi eksempler på dette.

I dette eksempelet ønsker bruker A å bestille 3-6 time mandag 16.juni. Rett før har bruker B rukket å bestille 3. og 4. time. Fordi de lastet romplanen på likt, vil dette altså ikke bli synlig for A før vedkommende laster timeplanen på nytt. Bruker A får da opp feilmeldingen som vist nedenfor.



Konklusjon

I og med at flere av elementene i dette hovedprosjektet var mer eller mindre nye for oss, satte vi av tid i starten til å samle informasjonskilder i form av nettsteder og bøker, og å sette oss inn i php, mysql og databasehåndtering. Etter det satte vi i gang med selve oppgaven, og første punkt var å sette oss inn i GPUntis for å se hvordan vi, på best mulig måte, kunne behandle dataene vi fikk derifra. Vi fant ut at GPUntis er ingen database, og de lagrede data er i et GPUntis spesifikt format. For å eksportere til tabseparerte tekstfiler som vi kunne lastes inn i databasen vår, trang vi en tilleggsmodul som vi, mandag 7.april, bestilte fra Istsoft, forhandler av GPUntis. Denne ankom ikke oss før fredag 25.april. Dette førte til store forsinkelser i prosjektet vårt, da vi hverken hadde noe å jobbe mot eller med, eller visste hva vi kunne forvente av denne modulen. Vi så oss dermed nødt til å endre på oppgaveformuleringen, noe som har resultert i at ikke alle punktene i vår første oppgaveformulering er løst. Derimot har vi utarbeidet forslag til løsning på disse problemene.

Etter hvert som vi kom i gang med programmeringsdelen, ble dette prosjektet utfordrende og interessant å arbeide med. Underveis så vi det nødvendig å fordele arbeidsoppgaver, da denne oppgaven har bestått av forskjellige elementer, men store deler har vi også jobbet samlet. Til slutt endte vi opp med et program vi er svært fornøyd med, og som, slik vi ser det, har et stort potensial for videre utvikling.

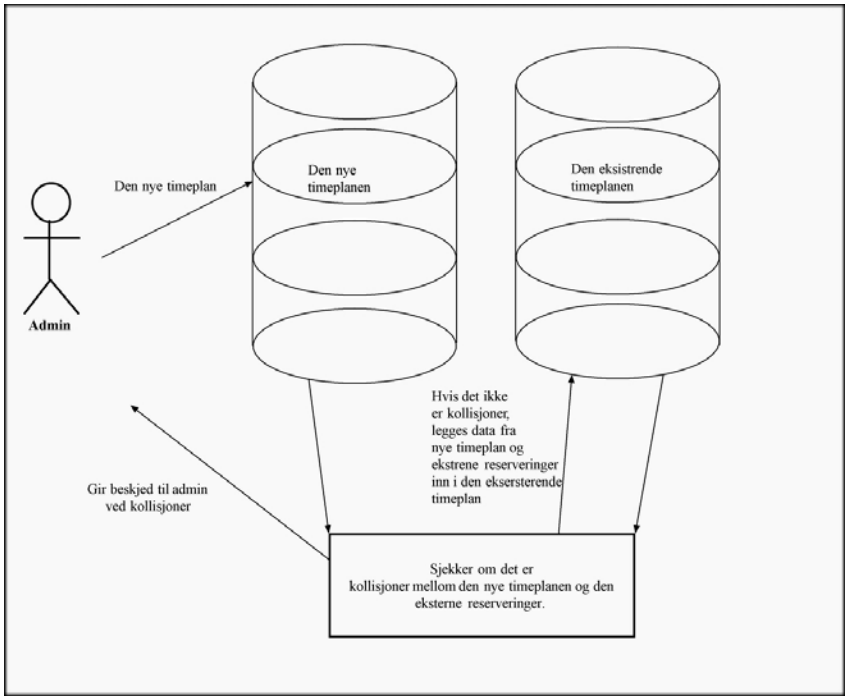
Prosjektet besto av å utvikle et program etter ønske fra kunde, og dette har i stor grad ført til større innsikt i produktutvikling. Også forståelse for andre teoretiske fag vi har hatt tidligere i dette kurset har økt. Prosjektet har også tilført oss kunnskaper om webside-utvikling vha php, forbedret kunnskap om MySQL, og ikke minst om det å samarbeide som gruppe.

Fremtidsutsikter

Oppdateringer

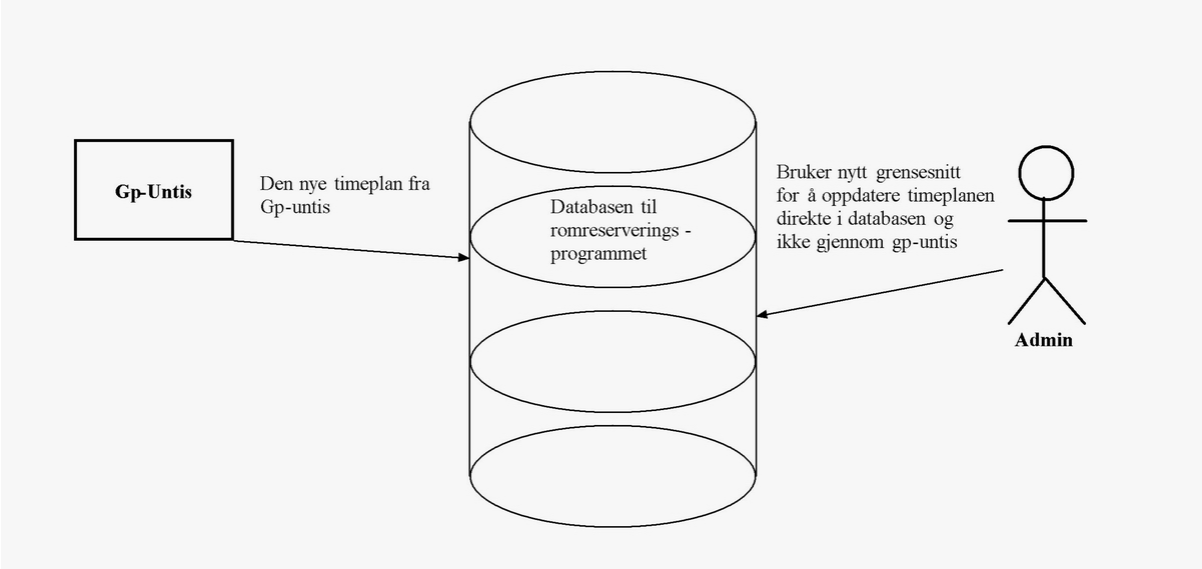
Det store problemet for denne oppgaven er inkompatibiliteten mellom GPUntis og verktøyene benyttet for å vise informasjon på nettet. Dataene i GPUntis er lagret som en fil i et propitiert format. Vi antok lenge at den var en database, men dette viste seg å være feil. Vi har brukt en tilleggsmodul til GPUntis som genererer tabseparerte filer med data fra den aktuelle databasen. Disse dataene importerer vi inn i mySQL hvor det blir lagret i tabeller. Det er i denne overføringen at problemet ligger. Timeplanen er ikke statisk og vil forandres rundt tjue ganger per semester. Siden GPUntis og vår database er adskilt så er det et ganske stort problem og holde dem oppdaterte. Under har vi forklart to metoder som vi ser som mulige løsninger til dette problemet.

For hver gang det skjer en forandring i timeplanen som ligger i GPUntis, så må databasen på nettet oppdateres. I seg selv er ikke dette et stort problem, men problemet oppstår når vi har eksterne reserveringer i den databasen som vi bruker på nettet. Disse er ikke lagret i timeplanen som vi har i GPUntis og eksisterer bare på databasen. Når disse kolliderer med timer som har blitt lagt til i timeplanen får vi et problem. Eksterne reserveringer skal overstyre nye timer, derfor må den nye timeplanen korrigeres slik at ingenting blir overkjørt. For å sjekke om det er noen kollisjoner må man gjennom en ganske tung prosess. Det må opprettes en midlertidig database slik at vi kan foreta SQL spørringer mot den. Deretter vil vi spørre den eksisterende databasen om den har noen eksterne reservasjoner. Hvis den ikke har det er det bare å overskrive hele den gamle databasen med den nye. Hvis det finnes reservasjoner, noe det høyst sannsynlig gjør, blir det litt verre. For hver reservasjon i den gamle databasen må vi sjekke om det er noe i den nye databasen som kollidere. Hvis det er en kollisjon så vil man måtte gjøre om på timeplanen slik at det ikke kolliderer mer. Det vil da selvfølgelig bli skrevet ut når og hvor kollisjonen er. Man må da korrigere slik at kollisjonen ikke lenger blir et problem og så laste den nye fil inn i databasen igjen. Så måtte vi sett etter om det ble noen nye kollisjoner, og repetere dette til det ikke ble flere kollisjoner. Dette kan bli en lang slitsom prosess, men den kan gjøres enklere hvis man tillater at et timeplanlagt rom kan benyttes til noe annet for en uke, da vil ikke



kollisjoner gjøre noe. Men det spørs om dette er praktisk gjennomførbart. Nå når vi har kommet så langt at det ikke er noen kollisjoner kan vi erstatte en gamle databasen med den nye pluss de eksterne reservasjonene som ligger i den gamle databasen. Alt dette kan gjøres i et PHP skript med SQL spørringer mot begge databasene. Det er heller ingen problemer med å implementere disse skriptene inn i systemet forutsatt at den nye databasen blir lastet inn i den temporære mySQL databasen. Det må gjøres manuelt, men det er laget noen filer som gjør denne jobben ganske grei og kan fikses med en kommando i mySQL.

Vi har også tenkt på en annen løsning. Den innebærer at vi flytter all modifikasjon av timeplanen over på databasen som ligger på nettet. Slik at utarbeidingen av timeplanen foregår på GPUntis. Dette er på grunn av at GPUntis har flere tilleggsfunksjoner som gjør dette arbeidet lettere. Mens endringer i timeplan gjennom semesteret gjøres direkte mot databasen. Dette forutsetter at det ikke er for mange for store forandringer. Da kan denne løsningen fort bli uoversiktlig og vanskelig å jobbe med. Vi ville her utvidet administrator-grensesnittet til å inkludere sletting av planlagte timer og innsetting av nye timer. Under dette faller også kollisjonsdeteksjon.



Den første løsningen er antakelig enklest for administrasjon å innføre siden de allerede har kompetanse på GPUntis, men den krever at en lærer seg et lite stykke mySQL kommandoer. Den andre vil være den mest elegante. Vil man bare oppdater databasen for hvert semester fra GPUntis. Men den avhenger av hvor store forandringer det er i timeplanen.

Generelt

Et romreserveringsprogram som dette kan og bør utvikles noe over tid, ettersom det kommer flere ønsker etter hvert.
Som en videreutvikling av programmet bør grupperom og "dørlapper" tas med, dette er ikke tatt med i denne omgang fordi tiden løp fra oss. Grupperom er det for det meste studenter som reserverer, så dette er også en del som bør utvikles mer, med tanke på flere brukergrupper. I denne omgang er det bare lærer og administrator som er tatt med som brukergrupper. Dørlapper er ønskelig fra skolens side, men det er ikke blitt prioritert i denne omgang siden det er klassifisert som "kjekt å ha", men det er jo noe som bør tas med i en videreutvikling.

Litteraturliste

Bøker:

Webb-programmering med PHP
Forfatter : Viktor Jonsson
c Viktor Jonsson och Studentlitteratur 2001
Omslag: BrandtDesign
ISBN 91-44-01941-6

Professional PHP programming
Forfatterliste: Jesus Castagnetto
Harish Rawat
Sascha Schumann
Chris Scollo
Deepak Veliath
c 1999 Wrox Press
ISBN 1-861002-96-3

PHP Bible
Forfatterliste:Tim Converse
Joyce Park
c 2002 by Wiley Publishing ,Inc , Indianapolis,Indiana
ISBN 0-7645-4955-3

Learning SQL
Forfatterliste:Richard Earp
Sikha Bagui
ISBN 0-201-77363-5

Nettsteder:

www.mysql.com
www.php.net
www.minlilleverden.net
www.sindrem.no

www.google.com (groups)

Vedlegg

Kildekode:

1.Adminindex.html
2.Header.php
3.Meny.php
4.Main.php

5.Brukerindex.html
6.Header.php
7.Meny.php
8.Main.php

9.Sok_rom.php
10.Vis_bestillinger.php
11.Vis_egne.php
12.Direkte.php
13.Finn_rom.php
14.Oppdater.php
15.Registrer.php
16.Rom_plan.php
17.Slett.php

18.Logginn.php
19.Logginnaction.php
20.Loggut.php
21.Nybruker.php
22.Nybrukeraction.php
23.Bruker.php
24.Endre_epost.php
25Endre_pas.php

Inc filer:

26.Smtp_mail.inc
27.Mime_mail.inc

Ønskes mer informasjon om oppgaven, henvend deg til Høgskolen i Vestfold, avdeling RI.