Indholdsfortegnelse

Indledning

Denne rapport er skrevet som en del af den afsluttede hovedopgave på datamatiker studiet. Hovedopgaven er lavet i samarbejde med Hanstholm Turistfart.

Hanstholm Turistfart, består af 15 busser og 16 chauffører samt 4 personer på kontoret, der beskæftiger sig med rute og turistfart, såvel indenlands som udenlands.

I øjeblikket fungerer alt administration af busser og chauffører ved hjælp af en stor papir kalender og diverse papirlapper. Hertil kommer en mappe for hver måned i året, hvor der kan findes yderligere information om den pågældende kørsel. Formålet er altså at udvikle et system, der kan hjælpe kontorpersonalet i det daglige arbejde, således at de ikke skal have fat i den fysiske kalender, som måske allerede er i brug af en anden, når der skal foretages ændringer og indskrives indkomne ordrer.

For at lærer og censor har mulighed for at afprøve systemet, har vi fundet et sted på Internettet, hvor systemet kan ligge midlertidigt. Denne udbyder har dog ikke den nyeste MySQL 4.0, så derfor kører systemet ikke optimalt. F.eks. vil der opstå fejl når man prøver at se overlap, og udføre backup funktioner, idet disse er implementeret med metoder der kun understøttes i 4.0.

Adresse til systemet:

http://hovedopgave.dyndns.dk

Brugernavn: cen
Kodeord: censor

Samtidigt har vi vedlagt nogle screenshot af systemet i brug, som kan ses på bilag 1.

Problemformulering

Som det første vil vi udarbejde en projektplan ved hjælp "Strategisk Projektledelse" af Andreas Munk Madsen. Her vil vi uddybe formål og mål for projektet og derved finde en strategi at gå ud fra. Derefter vil vi lave en tidsplan for projektets forløb, på baggrund af aftaler med såvel vejleder, Carsten Sørensen, folkene ved Hanstholm Turistfart og gruppemedlemmerne iblandt.

Systemet som vi har fået til opgave at udvikle, skal være Hanstholm Turistfarts nuværende almindelige papirkalender, i digital form med større udbyggelse af funktionaliteten. Systemet skal køre på lokalt netværk såvel som over Internettet. Systemet skal ligge på kundens egen Linux server.

På baggrund af mødet har vi fundet ud af kalenderen skal indeholde: hvilken chauffør, der kører hvilken bus til hvilken destination for hvem, med hvor mange og hvornår. Derudover var der ønske om at der kunne dannes et overblik over hvilke chaufførerog busser der er fri på hvilke tidspunkter. Endvidere blev der udtrykt ønske om at have en form for brugerniveauer med forskellige rettigheder til kontorpersonale og chaufførerog.

Udover den generelle funktionalitet skal systemet også indeholde en administrationsdel hvor, personer med højere brugerniveau, kan oprette og ændre busser, chaufførerog, rejseledere, kunder og brugere. Derudover skal der være mulighed for at foretage backup og indlæse disse.

Sikkerhed er en væsentlig faktor, da systemet skal være tilgængeligt fra Internettet, og dermed et muligt mål for uvedkommende personer.

To meget væsentlige punkter for kunden, er at systemet skal være brugervenligt og frem for alt fleksibelt, med hensyn til informationer der kan indtastes. Kunden har før haft et system på prøve, der skulle administrere kalenderen, dette system var dog langt fra fleksibelt og blev derfor ingen succes.

Kravspecifikation

I samarbejde med Hanstholm Turistfart har vi fundet frem til hvilken funktionalitet og hvilke oplysninger systemet skal indeholde. Denne er udformet verbalt, idet systemet er relativt overskueligt, på trods af de mange krav. Derfor har vi valgt at undlade at udarbejde en mere formel kravspecifikation med for eksempel brugsmønstre og klassediagrammer.

Chauffør
Navn, adresse, telefonnumre, initialer, hvornår skal vedkommende have fri og hvorfor.

Bus
Nummer, navn, beskrivelse, antal passagerer, registreringsnummer, hvornår bussen skal på værksted og til syn samt ellers ikke er tilgængelig for kørsel.

Rejseleder
Navn, adresse, telefonnumre, initialer.

Kørsel
Hvornår og hvor starter og slutter kørselen, er regningen sendt og af hvem, hvem bestiller kørselen, hvem betaler for kørselen, hvilke busser skal benyttes, hvem skal køre og er kørselen aftalt med dem, hvilke rejseledere skal med, detaljer om kørselen, hvilken type kørsel(rute, transport, udflugt), skal kørselen gentages og hvornår, hvor mange skal med på kørselen, har de et gruppenavn, hvad koster turen, hvornår sluttede turen i forhold til det aftalte.

Kunder
Navn, adresse, telefonnummer.

Dagsnote
Gemme en note, der hører til en bestemt dato.

Priser
Gemme oplysninger om priser på forskellige typer kørsler.

Dette er de oplysninger, som Hanstholm Turistfart ønsker systemet skal indeholde. Derudover skal der være loginoplysninger med brugernavn, password og hvilket rettigheder brugeren har i systemet. Alle disse oplysninger tegnes over i et ER diagram til brug ved design af databasen.
Det skal være muligt at taste alle førnævnte oplysninger ind i systemet. Derudover skal der være mulighed for at se oplysningerne på forskellige måder.

Man skal kunne vælge en dato og se

- Alle kørsler på den valgte dag. Dvs. alle kørsler som starter, slutter eller løber henover dagen. Samt hvilke busser og chauffører, der er tilknyttet. Det skal være muligt at markere regningen som sendt, og så bliver der registreret hvem, der har markeret regningen som sendt. Ved at trykke på en bestemt kørsel, skal det være muligt at se alle oplysninger om den pågældende tur.
- Hvilke chauffører, der skal have fri.
- Hvilke busser, der ikke er tilgængelige for kørsel.
- En note tilknyttet den valgte dag og ændre denne.
- Oprette og ændre kørsler den valgte dag.

Det skal være muligt at se aktuel status for

- Hvilke busser, der er på kørsel
- Hvilke chauffører, der er på kørsel
- Hvilke rejseledere, der er på kørsel

Det skal være muligt at
Vælge en periode og chauffører og få udskrevet dagssedler.

- Se hvilke chauffører og busser, der er dobbeltbooket, dvs. sat på flere kørsler samtidigt.
- Se om der er chauffører eller busser, som er sat på kørsler, samtidig med de er markeret som utilgængelige for kørsel.
- Udføre backup.
- Slette informationer om gamle ture.

Derudover skal systemet være

- Brugervenligt, hurtigt at lære og bruge
- Tilgængeligt fra flere computere samtidigt, både på kontoret og hjemmefra.

Projektplan

Baggrund

I forbindelse med den afsluttende eksamen på IT-akademiet 5. semester hovedopgave, skal der udvikles et produkt med udgangspunkt i en virksomhed. Denne virksomhed er Hanstholm Turistfart, der ønsker at få udarbejdet en IT baseret kalender.

Formål

- At reducere den administrative byrde for kontor personalet på virksomheden.
- At gøre administrationen af busser, chauffører og ture nemmere og mere effektiv.
- Give et større overblik over kørsler på en dag.

Mål

- Et færdigt og køreklart produkt.
- Systemet skal kunne bruges af flere samtidig og uafhængig af arbejdsstation.
- Systemet skal være tilgængeligt 24 timer i døgnet.
- Systemet skal administrere alle informationer om kørsler, busser og chauffører.
- Mulighed for backup.

Strategi

For at få størst muligt samarbejde med kunden har vi valgt den traditionelle projektmodel kombineret med faselinier fra den trinvise projektmodel. Dette betyder, at vi først har analysefasen fra den traditionelle projektmodel, hvorefter vi bevæger os videre til design og konstruktion. Her vil vi inddrage den trinvise model, der går ud på, at dele projektet op i mindre selvstændige moduler. Efterhånden som modulerne bliver færdigudviklet, fremvises de for Hanstholm Turistfart. Dermed sikrer vi os, at vi har kontakt med kunden under hele forløbet, idet vi ved hver trinafslutning har en del af systemet færdigt til at vise kunden. Dermed sikrer vi både at de formelle krav i kravspecifikationen, og kundens personlige ønsker til hvordan systemet skal se ud, opfyldes. Strategien bygger på rapid prototyping, hvilket betyder at det gælder om hurtigt at komme i kontakt med kunden og hele tiden komme med forslag til systemet ved hjælp af prototyper.

Turboanalyse

Mål og vilkår	**Særlig styrke**	**Særlige svagheder**	**Mulige beslutninger**
Teknik			
Linux PHP MySQL Apache	• Gratis • Platformsuafgængigt på klientsiden • Hurtigt		Har vi tid til at sætte os ind i et forholdsvis nyt programmeringssprog
Windows ASP IIS Access	• Platformsuafhængigt på klientsiden • Hurtigt	• Dyrt	Har vi tid til at sætte os ind i et forholdsvis nyt programmeringssprog
Java	• Platformsuafhængigt • Stor erfaring	• Langsomt • Kræver installation af klient	
Udviklerne			
Projektdeltagerne	• Entusiastiske • Deltagerne har arbejdet sammen siden første semester • Stor faglig dygtighed • Selvtillid	• Ikke stor kendskab til PHP og ASP.	• Vidensdeling • Informationssøgning
Resultatet			
	• Brugervenligt • Fleksibelt	-	• Test af systemet

Brugerne			
Kontorpersonale hos Hanstholm Turistfart	• Daglige brugere af IT • Åbne for nye ideer	• Forskellige tilvænningsperioder • Forkert brug af systemet	• Test
Omgivelserne			
	• Grupperum og computere stillet til rådighed på skolen • Frit skema	• Nedbrud af skolens netværk	• Arbejde hjemme

Sammenfatning af Turbo analyse

Vi har valgt at benytte os af Linux, Apache, PHP og MySQL, idet vi mener, at det er mest fordelagtigt i forhold til vores projekt, idet det er gratis og ikke kræver nogen installation på klientsiden. Da vi har valgt at benytte os af PHP, betyder dette, at vi ikke anvender objekt-orienteret design, idet dette ikke er fuldt understøttet af PHP. Derudover har vi valgt at arbejde på skolen, idet det skaber en bedre kommunikation mellem gruppemedlemmerne.

Interessent analyse

Der er to væsentlige grupper interessenter: Hanstholm Turistfart og skolen. Hovedinteressen i vores projekt ligger fra Hanstholm Turistfarts side i selve systemet, hvorimod skolens hovedinteresse ligger i rapporten. Dermed ligger der en vis interessekonflikt, idet Hanstholm Turistfart for så vidt er ligeglade med vores rapport, hvorimod det er rapporten, der danner grundlag for vores eksamen. Derfor må der en vis prioritering til hen ad vejen, hvor det kan blive nødvendigt at nedprioritere programmeringsdelen grundet tidspresset.

Risikoanalyse

Risiko	**Sandsynlighed**	**Konsekvens**	**Håndtering**
Sygdom hos udviklere	• Mellem	• Mindre alvorlig	• Uddelegering af hjemmearbejde. • Kontakt opretholdes vha. email og telefon.
Nedbrud af Linux-server	• Lille	• Generende	• Reinstallation og indlæsning af backup på server.
Ny version af PHP	• Lille/mellem	• Mindre alvorlig	• Opdatering af kode
Nedbrud af skolens netværk	• Lille	• Generende	• Arbejde hjemmefra • Vente på det kommer op igen
Brand på skolen	• Meget lille	• Generende	• Arbejde videre hjemmefra med backup
Kunden ønsker ikke længere at samarbejde	• Lille	• Alvorlig	• Fortsætter udvikling i simuleret forløb

Denne analyse kan vi bruge til at planlægge håndteringen af mulige situationer, der kunne opstå i løbet af projektperioden. På den måde, har vi diskuteret hvordan vi ønsker at håndtere de problemer, der opstår.

Tidsplan

Under hele forløbet vil der være sideløbende opgaveskrivning

Torsdag d. 19. Juni:

Grundlæggende orientering med HT

Analyse

Mandag d. 25. August:

Start på projektet

Start på udvikling af prototype på baggrund af første orienterende møde.

Torsdag d. 29. August:

Udformning af databasen fastlagt

Onsdag d. 3. September:

Kalender funktion færdig.

Torsdag d. 4. September:

Møde med HT, og præsentation af prototype.

Fredag d. 12. September:

Oprette kørsler funktion færdig.

Visning af kalender med dertilhørende info.

Fredag d. 26. September:

Udskrivning af dagssedler for chauffører færdig.

Ændre oplysninger for kørsel færdig.

Onsdag d. 1. Oktober:

Administrationsdel færdig.

Mandag d. 6. Oktober:

Køreklar version færdig.

Mandag d. 13. Oktober:

Installation af systemet hos Hanstholm Turistfart

Periode med brugertest af systemet

Onsdag d. 5. november

Aflevering af projekt

Strategisk Projektledelse

Organisationstype

Organisation er defineret som: ”Et antal mennesker, der arbejder sammen, og som er bundet sammen af aftaler.[1]” i Strategisk Projektledelse. I dette tilfælde er organisationen gruppemedlemmerne. Sagens kerne er at projektdeltagerne har fælles mål med projektet.

Hele vores projekt bindes sammen af aftaler. For eksempel er kravspecifikationen en aftale med kunden om projektets resultat. For at styre projektets forløb, laves der også aftaler mellem de enkelte medlemmer af projektgruppen, som for eksempel arbejdsfordeling, arbejdstider og arbejdsindsats.

På nedenstående figur kan man se, at vores organisation befinder sig i feltet lav opgaveusikkerhed og høj opgavekompleksitet. Dette er fordi at selve at opgaven har været klar fra starten af, da administrationen på Hanstholm Turistfart selv lige fra starten af projektforløbet havde en klar ide om hvilke egenskaber systemet skulle have og/eller hvilke ting de ønskede registreret. Selve platformen systemet skulle køre på og hvilket sprog, der skulle bruges som hovedudviklingssprog, blev ret hurtig klar for os, da vi i de første 2 uger af projektet var af den overbevisning at systemet skulle ligge på Hanstholm Turistfarts eksisterende Linux-server, hvor deres billet bookingsystem kører. Dette blev dog lavet om da udvikleren/administratoren for bookingsystemet ikke syntes om ideen, da der kunne komme softwarekonflikter på maskinen.

Grunden til at vi mener, at opgave kompleksiteten er stor i projektet, er at systemet er omfattende og skal fungere fejlfrit. Derudover er hovedsproget (PHP) noget som vi ikke har den store erfaring indenfor, og skal derfor prøve os en hel del frem.

[1] Strategisk Projektledelse, side 5

Derfor kan man konkludere, at professionel bureaukrati er den rette betegnelse for vores organisation, da vi alle har gennemgået det samme uddannelsesforløb siden folkeskolen, med Handelseksamen og 4 semestre af datamatikeruddannelsen. Dette gør, at vi meget har de samme holdninger til udformningen af et system. Dette kan ses på den type aftaler, der er nævnt i kassen med professionel bureaukrati, hvor der står "fælles holdninger". Dette skal forstås sådan, at der ikke er brug for at lave nær så udspecificerede aftaler projektdeltagerne imellem.

	Lav opgaveusikkerhed	Høj opgaveusikkerhed
Høj opgavekompleksitet	Professionel bureaukrati Fælles holdninger	Projekt Ad hoc-aftaler
Lav opgaveusikkehed	Bureakrati Regler	Simpel organisation Direkte styring

Fig. 1: Sammenhængen mellem opgavetyper, organisationstyper og primære aftaletyper. [2]

[2] "Strategisk Projektledelse", side 5

Projekttype

Ud fra nedenstående figur vil vi prøve at definere typen af projekt vi er i gang med. Som man kan se på nedenstående figur er der tre typer af projekter, nemlig udviklings-, konstruktions- og forhandlingsprojekt. Man kan forholdsvist hurtigt udelukke at der er tale om et forhandlingsprojekt, da der kun gøres brug af denne ved større uenighed i gruppen.

Vi er hovedsageligt i gang med et udviklingsprojekt, da det er et helt nyt system vi skal lave, som vi hverken har tidligere udviklede komponenter til, eller tidligere erfaring med udvikling af et sådant system. Dog blev projektet hurtigt en hybrid af et udviklings- og et konstruktionsprojekt. Da programmeringen foregår samtidig med at konstruktionen af programmet er i fuld gang, er der hele tiden nye tanker og ideer til programmet, der bliver implementeret.

	Udviklingsprojekt	Konstruktionsprojekt	Forhandlingsprojekt
Formål	At skabe fornyelse	At løse en bunden opgave	At finde en løsning der er enighed om
Usikkerhed	Høj	Lav til middel	Lav til høj
Kompleksitet	Middel	Høj	Lav til høj
Ressourcebevidsthed	Lav	Høj	Lav
Udgangspunkt	Et løsnings- og ressourcemæssigt spillerum	Sammenhængende krav og planer	Interessemodsætninger og fællesskab
Ledelsesprincipper	Strategi, motivation og innovation	Resultatorienteret	Tid til at finde langsigtede, fælles interesser

Fig. 2: Forskelle mellem de tre projekttyper. [3]

[3] "Strategisk Projektledelse", side 10

Eksempel på navigering

Med udgangspunkt i kravsspecifikationen beskrives her det primære brugsmønster i systemet ved hjælp af et tilstandsdiagram, som set på figur, i henhold til Craig Larman´s "Statechart Diagram" [4]. Dette diagram har vi brugt til at give et overblik over, hvorledes vi skal opbygge filer og kode.

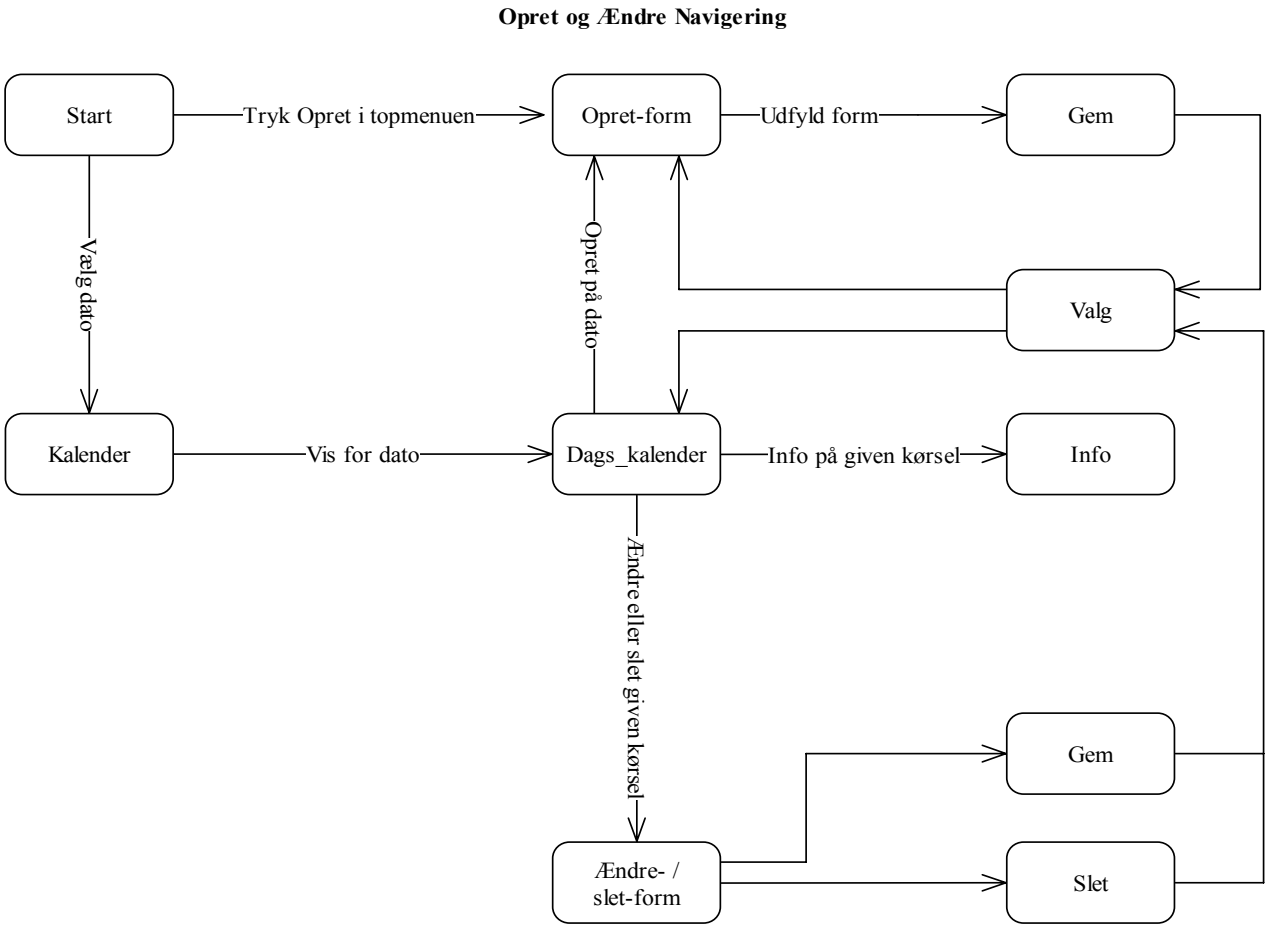


Fig. 3: Opret og ændre navigering

[4] "Applying UML and patterns", side 438.

Brug af filer til "Opret og Ændre Navigering"

- kalender.php – Selve kalenderen, hvor man kan navigere rundt mellem forskellige datoer.
- viskalender.php – Visning af kørsler og anden information for den valgte dato.
- opret.php – Filen med formen til at oprette en ny tur.
- opret_aendre.php – Filen med formen, hvor de data for den kørsel man vil ændre, bliver indsat.
- opret_gem.php – Her foregår alle indsættelser i databasen for opret.php og opret_aendre.php.
- mere_info.php – Generering af information på valgt kørsel.

Distribuerede systemer

Som man kan se på figur 4 er der tale om et klient/server designmønster, hvor der er en server og et antal tynde kliener, der ikke skal installere andet end Microsoft Explorer, for at benytte sig af systemet.

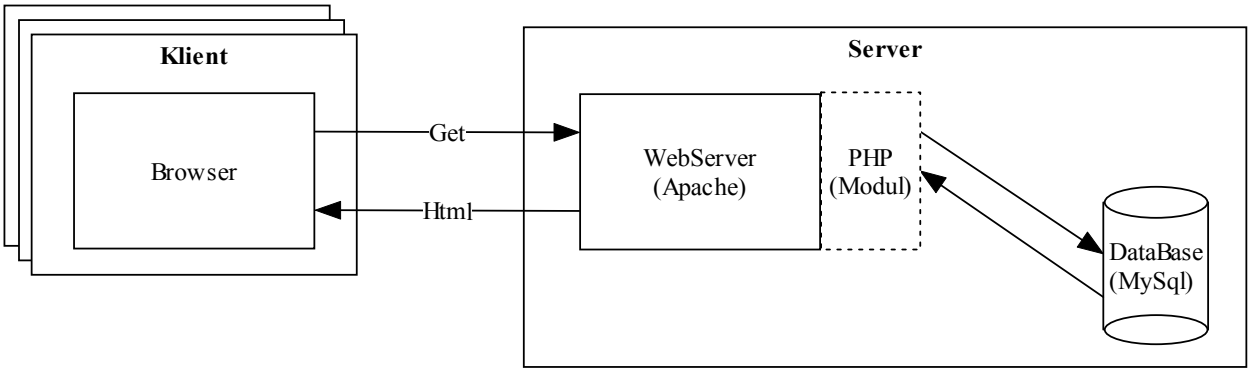


Fig. 4: Kommunikation mellem browser og webserver.

Transparens

Alle processerne, flytning af information/filer og tilgang til data uden for klient maskinens regi, skal ske ubemærket, altså brugeren kan ikke se at der foretages kontakt til andre maskiner, også kaldet transparens.

Vi har registreret Hanstholm Turistfarts IP-nummer ved http://www.dyndns.dk, som er et firma hvor man gratis kan blive indregistreret på deres DNS server. Så vi nu har et såkaldt venligt navn registreret hos dem, så brugerne af systemet i stedet for IP-nummeret, kan bruge dette venlige navn. På denne måde vil man ikke lægge mærke til, at Internetforbindelsen får nyt IP-nummer, hvilket sker da en ADSL-linie som standard får sit IP-nummer tildelt dynamisk.

Skalerbarhed

Her er der tale om at systemet skal være nemt at udvide, altså sætte flere klient maskiner til at bruge systemet. Som sagt har vi valgt at benytte os af en distribueret webløsning, hvor vi har en Apache webserver, der i teorien kan have uendelig mange klienter. Derfor er det utroligt nemt at udvide antallet at af klienter på systemet. Det eneste, der kræves er et netkabel og plads i hubben.

Ud over at have adgang til systemet på det lokale netværk er der også Internetadgang, der gør at personalet hos Hanstholm Turistfart kan benytte sig af systemet hjemmefra. Hvis systemet skulle håndtere et rigtig stort antal klienter, kan det gå ud over ydeevnen på systemet, da det i øjeblikket kun er en maskine, der kører alle server-funktioner. For at løse problemet med et stort antal klienter, kunne man f.eks. vælge at flytte MySQL over på en anden server. Dette er dog ikke nødvendigt, da det er et begrænset antal brugere, der skal bruge systemet hos Hanstholm Turistfart.

Skitse over netværket hos Hanstholm Turistfart

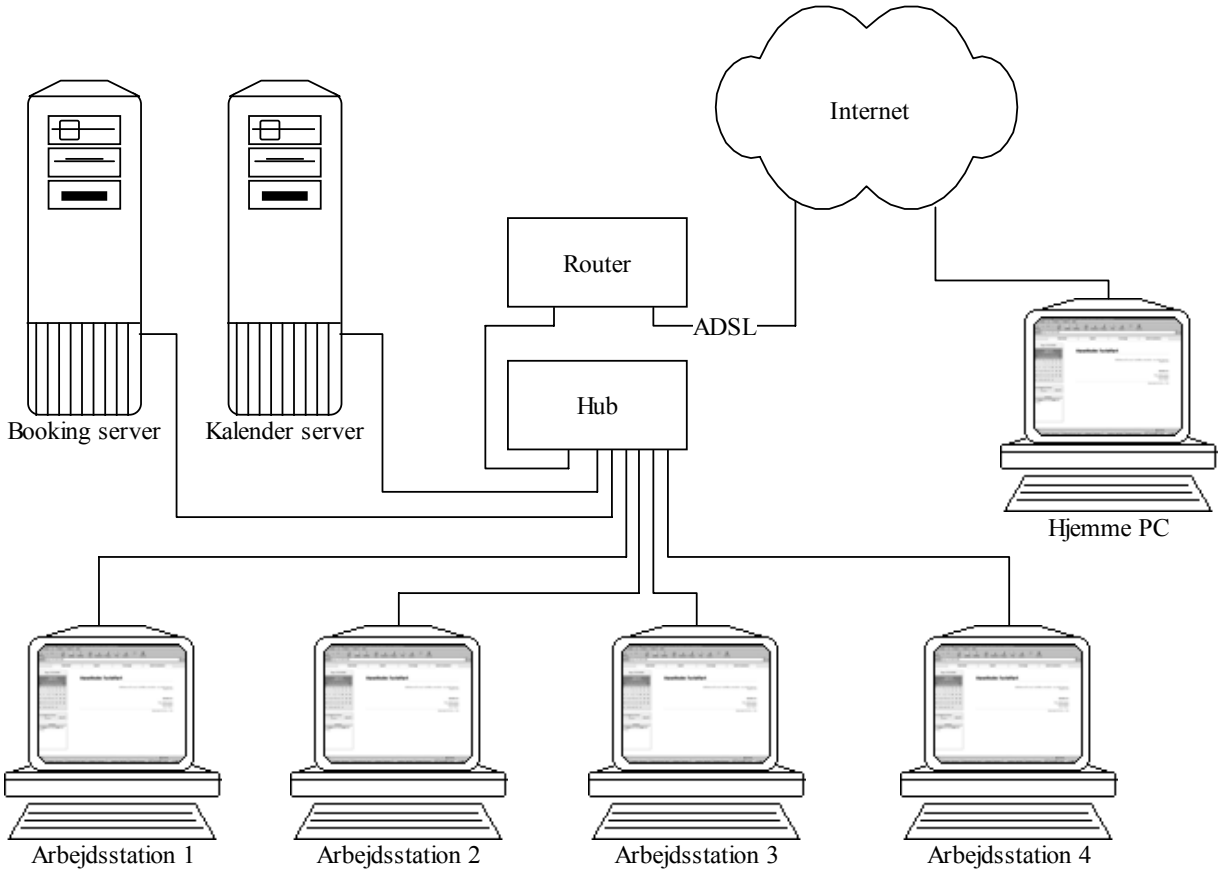


Fig. 5: Skitse over netværket hos Hanstholm Turistfart

Hanstholm Turistfart har i øjeblikket 4 arbejdsstationer på kontoret, som er koblet sammen i en hub. Hub'en er koblet til en router, som har forbindelse til Internet via en ADSL forbindelse. Booking-serveren er en anden Linux-server, som Hanstholm Turistfart i forvejen har kørende. Booking-serveren er koblet til hub'en, ligesom vores system, der ligger på kalender-serven.

I weekenden er det normalt, at en af de fire på kontoret tager den almindelige kalender med hjem, hvis nogen skulle ringe og bestille en tur, eller der opstår problemer. Vi har derfor gjort det muligt at logge på systemet hjemmefra.

Da vi skulle vælge, hvordan systemet skulle køre, havde vi to alternativer. Enten at leje sig ind på et webhotel, eller købe en server, som skulle stå hos Hanstholm Turistfart. Fordelene ved et webhotel er at man slipper for indkøb, konfiguration og vedligeholdelse af en maskine. Da vi har valgt en Linux løsning med gratis software, er er der altså ingen yderligere startomkostninger ved selv at have en server stående, end netop omkostningerne til serveren. Havde vi for eksempel valgt at udvikle i ASP havde der, udover selve serveren, også været store omkostninger til software. Dermed havde der umiddelbart været en økonomisk gevinst ved at vælge et webhotel.

Hvis man vælger et webhotel er der en række faktorer man er nødt til at undersøge.
Det er webhotellets

- Oppetid
- Internetforbindelse
- Hastigheden på hardwaren
- Hvor mange megabyte plads
- Hvilke teknologier understøtter webhotellet
- Pris

Oppetid

Webhotellets oppetid er af afgørende betydning for et system som dette. Hvis ikke Hanstholm Turistfart ikke kan få adgang til deres kalender, kan de jo ikke udføre deres arbejde. Som Susan sagde ”hvis vores kalender forsvinder, kan vi ligeså godt lukke virksomheden”. Selvom webhotellet skulle være nede, er kalenderen ikke nødvendigvis slettet. Men i det øjeblik man ikke kan få adgang til webhotellet, kan man jo sådan set være ligeglad med om kalenderen er slettet eller ej.

Internetforbindelse

Hastigheden på webhotellets Internetforbindelse er vigtig. Webhotellers hastighed kan svinge meget. Et ekstremt eksempel kunne være en studerende, som sidder med sin egen computer på en 256kbit ADSL forbindelse og laver et webhotel. Det er simpelthen for langsomt. Man skal også kigge på hvor mange kunder webhotellet har, og dermed hvor mange der er om at dele Internetforbindelsen.

Hastigheden på hardwaren

Da vi kører med PHP og MySQL, er en vis hastighed på serveren nødvendig for at systemet kører ordentligt. Det nytter jo heller ingenting at webhotellet har den hurtigste Internetforbindelse man kan få, hvis den først skal stå og lave flere minutters opslag i MySQL.

Hvor mange megabyte plads

Da der skal ligge en MySQL database vil det være nødvendigt med en vis mængde plads på webhotellet. Man kan eksempelvis købe webhoteller med 1 megabyte plads, hvilket kan være fint nok hvis man blot ønsker en kort præsentation af sin virksomhed eller hobby. Men når der skal være plads til databasen, skal der flere megabyte til. Selve vores PHP filer fylder under en halv megabyte, og kræver altså ikke særlig meget plads.

Hvilke teknologier understøtter webhotellet

Vi har brug for MySQL og PHP understøttelse. De fleste webhoteller tilbyder understøttelse af dette. Men for at få disse teknologier, er man måske nødt til at vælge en samlet pakke, hvor der er flere megabyte plads end man egentlig har brug for.

Pris

Alle disse faktorer tilsammen vil påvirke prisen. Jo højre oppetid, hurtigere hardware og Internetforbindelse, flere megabyte plads, flere understøttede teknologier vil alt sammen presse prisen opad. Hertil kommer også et oprettelsesgebyr. Hvis man vælger et webhotel, er man som allerede omtalt afhængig af oppetiden på webserveren. Men derudover er man også afhængig af at ens internetforbindelse kører. Webhotellet kunne også blive ramt af strømsvigt. Begge faktorer peger på at det er en god ide, selv at have serveren stående. Dermed er man på kontoret uafhængig af om internetforbindelsen kører. Med hensyn til strømforsyningen kan der selvfølgelig også opstå strømsvigt på kontoret. Dette problem kunne afhjælpes ved at investere i en UPS.

Udfra disse faktorer har vi valgt at investeringen i en maskine, som står på kontoret vil være den mest optimale. Oprettelse af et webhotel med understøttelse af de nødvendige funktioner, en vis mængde plads og høj oppetid vil lynhurtigt blive dyrere end investeringen i en maskine, som står på kontoret og kører kalender-systemet.

Mapning fra E/R model til relationel model

I nogle eksempler er ER diagrammer forsimplet, for større overskuelighed. ER diagrammet er vedlagt som bilag 2.

Trin 1

Regulære entitetstyper

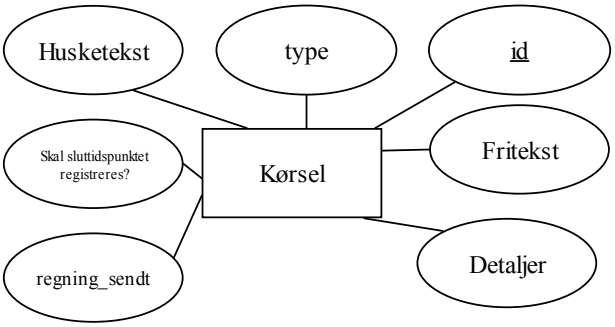
For hver stærk entitetstype E dannes der en relation R der:

Indeholder alle simple attributter i E
For sammensatte attributter medtages kun de simple komponent attributter
Vælg en nøgle for E som primær nøgle for R
Hvis nøglen er sammensat dannes nøglen R af den simple komponentattributter tilsammen

Vi har følgende stærke entiteter i vores E/R model.

- kørsel
- person
- chauffør
- bus
- rejseleder
- dagsnote
- login
- fast_kunde
- person
- tider
- priser

Kørsel entiteten

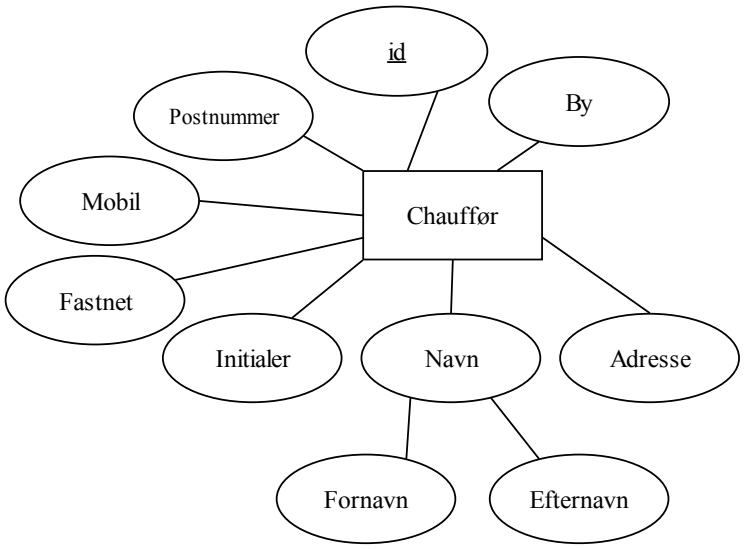


Bliver mappet til nedenstående.

koersel						
id_koersel	husketekst	fritekst	beskrivelse	type	angiv_reel	regning_sendt

ID-attributten bliver navngivet til id_koersel.
Husketeksten er en overordnet beskrivelse af kørslen. Eksempelvis ”Rute 23” eller ”Italienstur”. Friteksten er bruges til specielle korte noter omkring kørslen. Detaljer blev mappet over til en attribut i tabellen, som hedder beskrivelse, hvor der kan angives alle yderligere oplysninger om turens forløb.

Chauffør entiteten

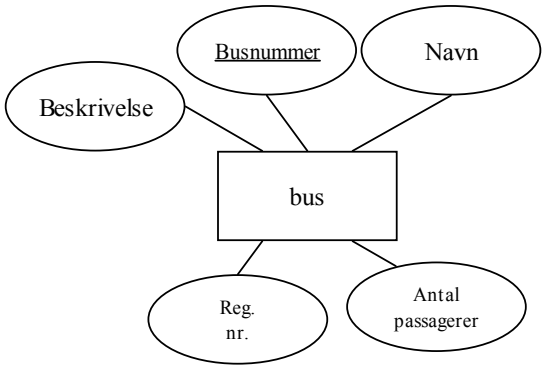


Bliver mappet til følgende.

Chauffoer								
id_chauffoer	initialer	fornavn	efternavn	adresse	postnummer	telefonnummer	mobiltelefon	by

Den sammensatte attribut ”navn” bliver delt op i de to attributter ”fornavn” og ”efternavn”.

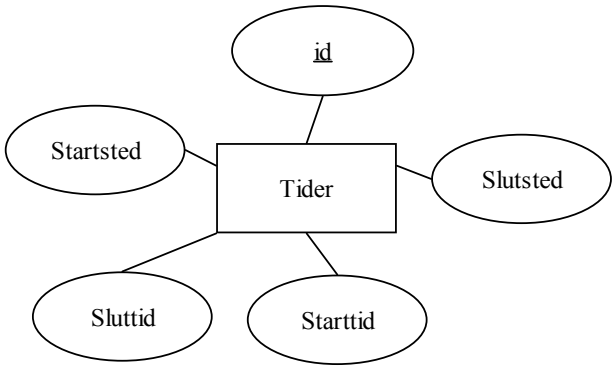
Bus entiteten



Bus				
Bus_nr	bus_navn	antal_passagerer	reg_nr	beskrivelse

Her er der to kandidatnøgler. Vi vælger busnummer som primærnøgle i denne tabel.

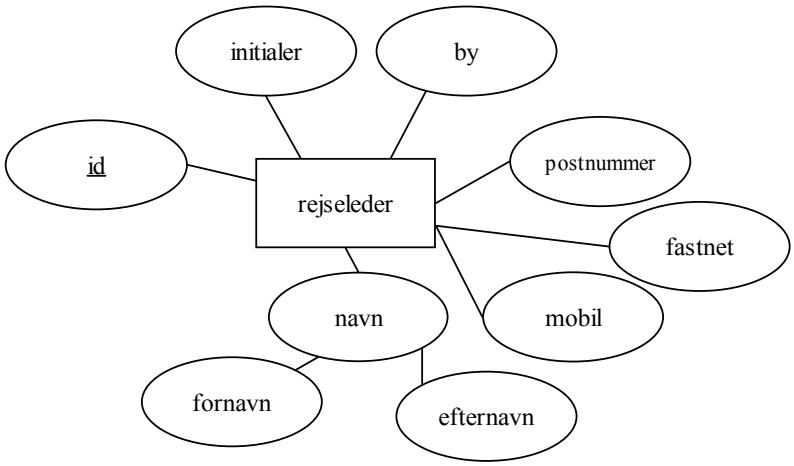
Tider entiteten



Mappes til følgende

Tider				
start_tid	slut_tid	start_sted	slut_sted	id_tid

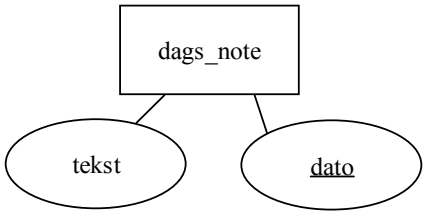
Rejseleder entiteten



Mappes til følgende tabel. ”Navn” attributten deles i ”fornavn” og ”efternavn”.

Rejseleder								
id_rejseleder	initialer	fornavn	efternavn	adresse	postnr	telefonnummer	mobilnummer	by

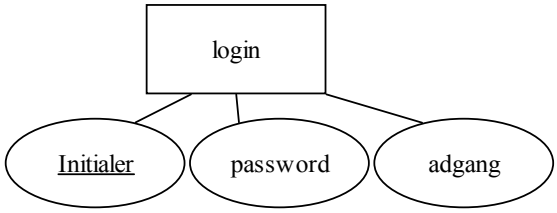
Dagsnote entiteten



Mappes til følgende tabel.

dags_note	
dato	tekst

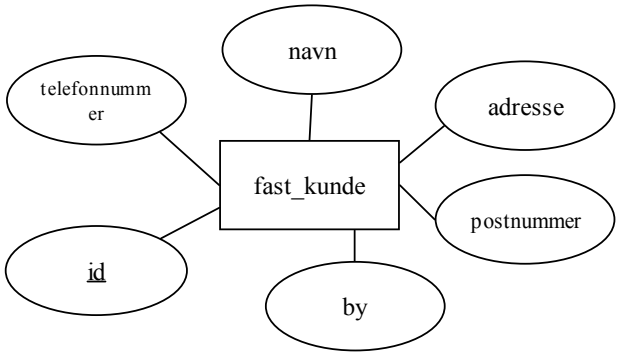
Login entiteten



Mappes til følgende tabel. "Adgang" attributten bruges til at bestemme adgangsniveauet for en bruger af systemet.

Login		
initaler	password	adgang

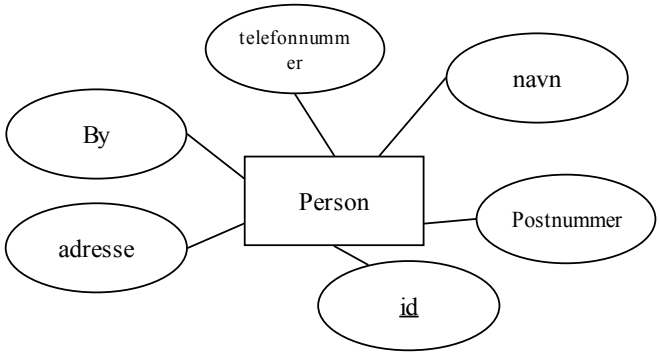
Fast kunde entiteten



Mappes til følgende tabel.

fast_kunde					
id_kunde	navn	adresse	postnummer	telefonnummer	by

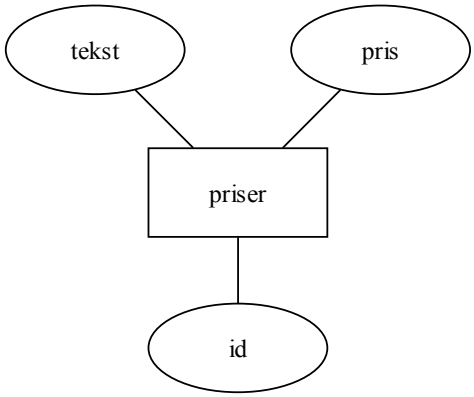
Person entiteten



Mappes til følgende

person					
id_person	navn	adresse	postnummer	telefonnummer	by

Priser entiteten



Mappes til følgende

Priser		
id_priser	tekst	pris

Trin 2

Svage Entitetstyper

For hver svag entitetstype W med den stærke entitetstype E og den identificerende relationstype R dannes en relation RW, der:

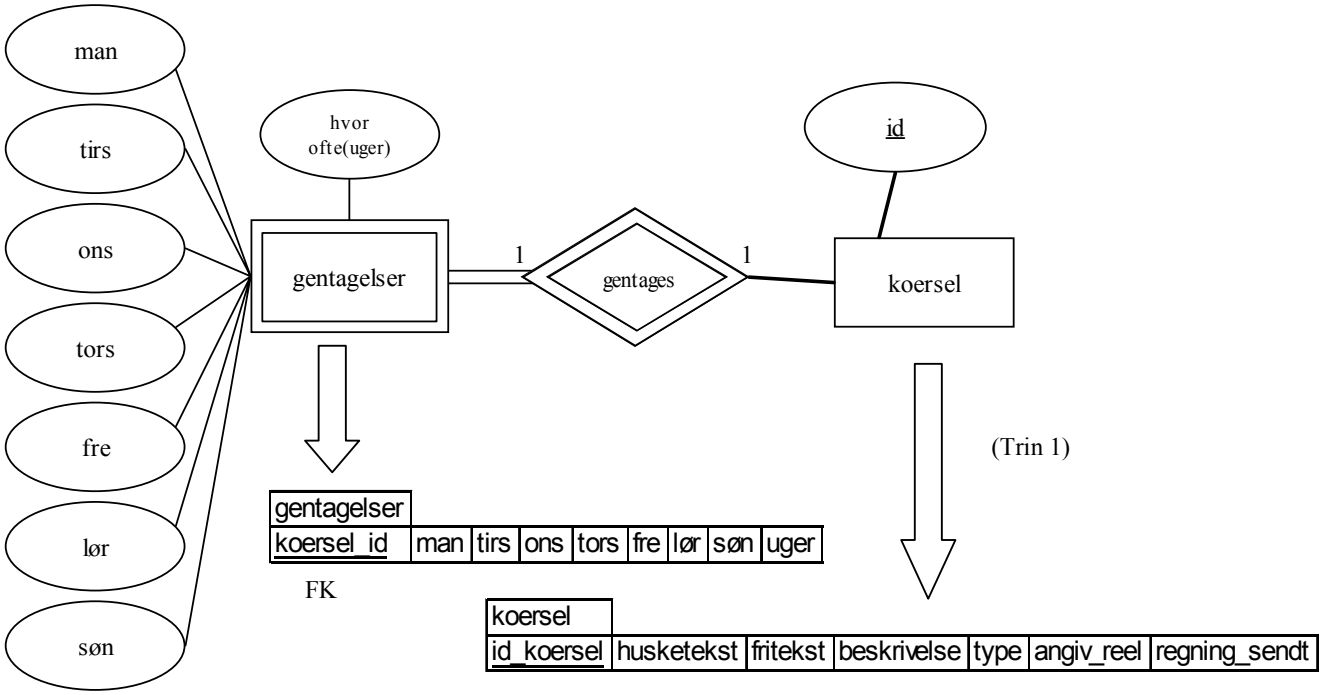
- indeholder alle simple attributter i w
- indeholder som fremmednøgle i RW primærnøglen for den relation der svarer til E
- primærnøglen for Rw er sammensat af primærnøglen for E og den partielle nøgle for W.

Vi har følgende svage entiteter i vores E/R model.

- Gentagelser
- Person_co
- Sluttidspunkt
- Kørsels_info
- Fast_kunde_co
- Chauffør_fri
- Utilgængelig bus

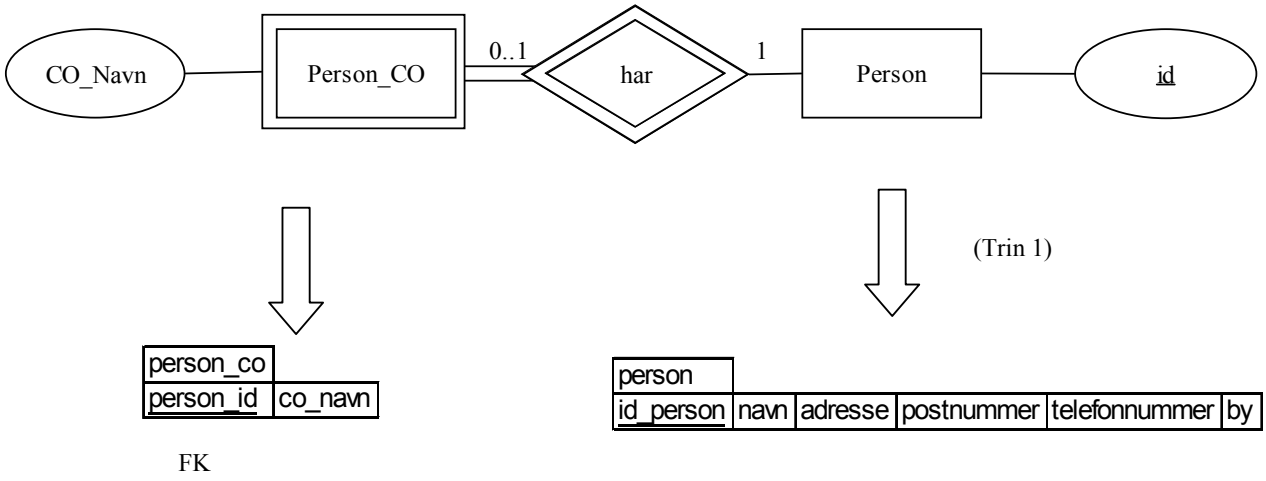
Gentagelser entiteten

Gentagelser kommer ned i en tabel for sig selv. Primærnøglen bliver samtidig fremmednøgle til koersel-tabellen.



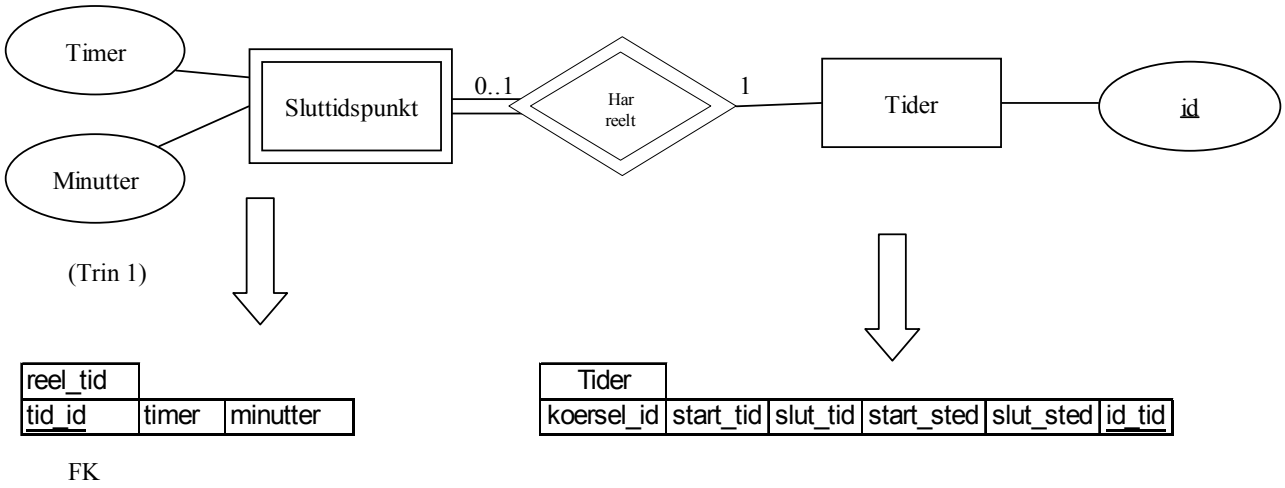
Person_co entiteten

Kun en attribut har vi at gøre med her. Den kommer ned i sin egen tabel, sammen med fremmednøglen "person_id", som refererer til person tabellen.



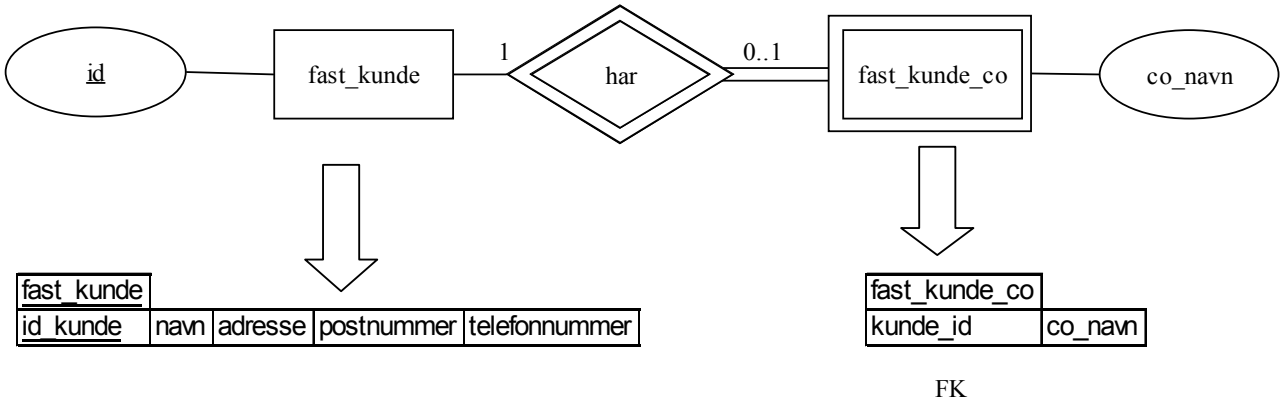
Sluttidspunkt entiteten

Sluttidspunkt attributterne kommer ned i en tabel for sig selv, og får som primærnøgle fremmednøglen til tider.

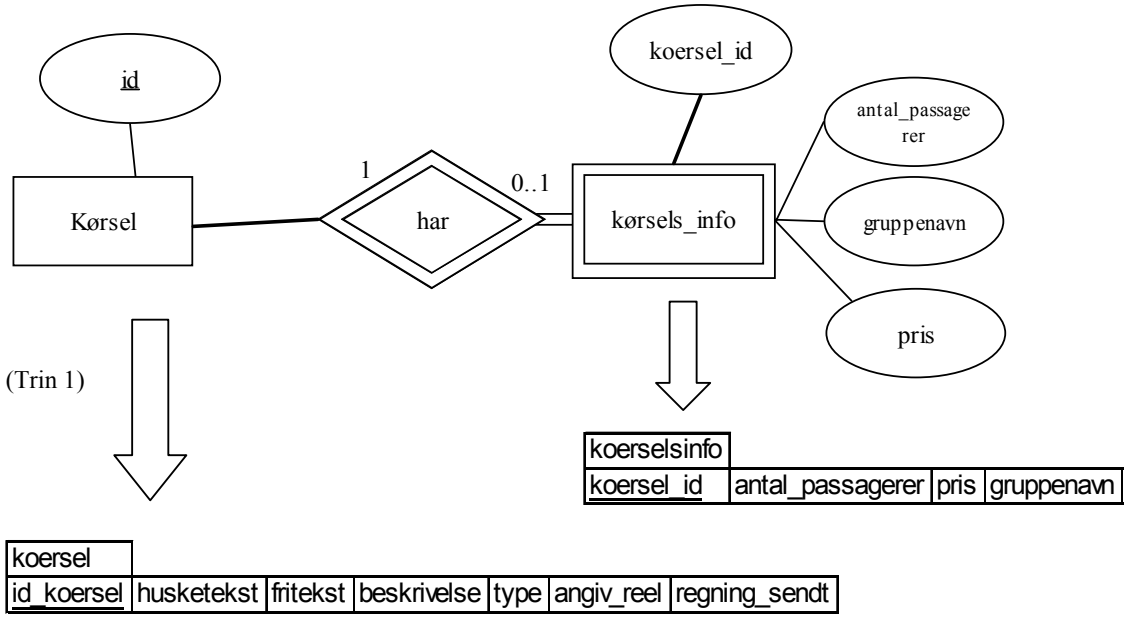


Fast_kunde_co entiteten

Princippet er det samme som person_co entiteten.

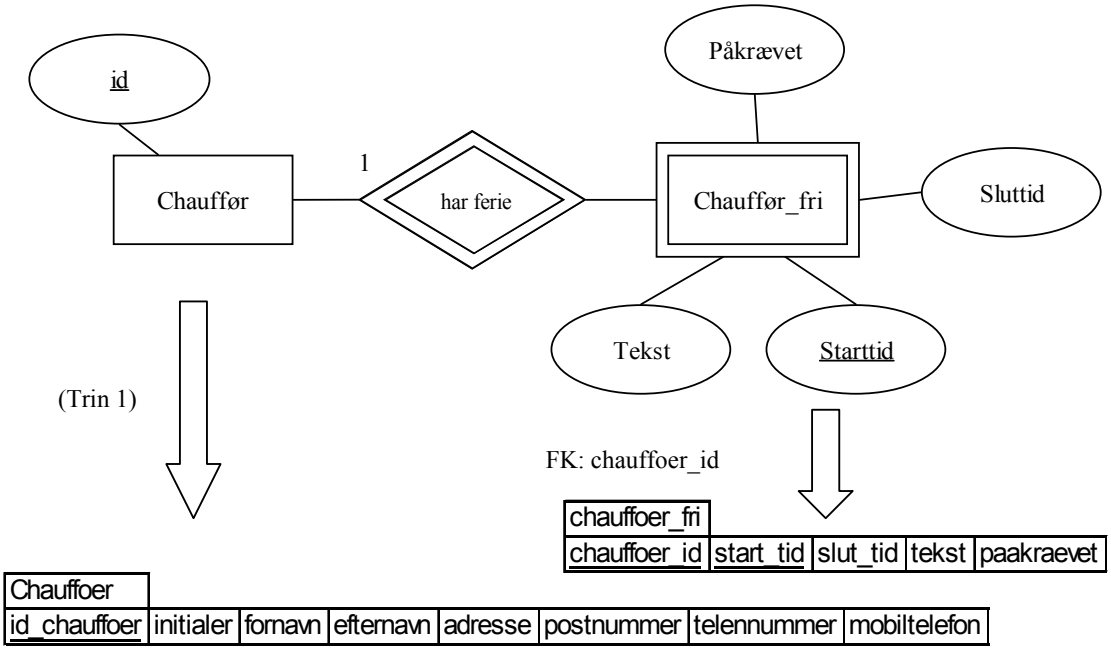


Kørsels_info entiteten



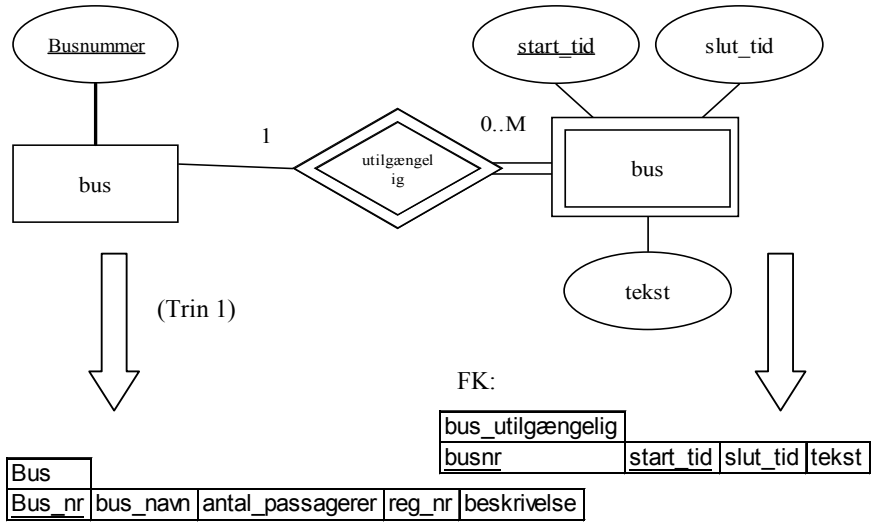
Primærnøglen for kørsels_info tabellen er koersel_id, som referer til kørselstabellen. Primærnøglen for tabellen for den svage entitet skal sammensættes af primærnøglen for den stærke entitet, samt den partielle nøgle for den svage entitet. Da nøglen er den samme i begge tabeller, er det ikke nødvendigt med en sammensat nøgle i den svage entitets tabel.

Chauffør_fri entiteten



Chauffør_fri mappes til en ny tabel, og primærnøglen for den sammensættes af fremmednøglen til chauffør tabellen og starttid.

Bus_utilgængelig entiteten



Princippet for mapningen er det samme som for chauffoer_fri.

Trin 3

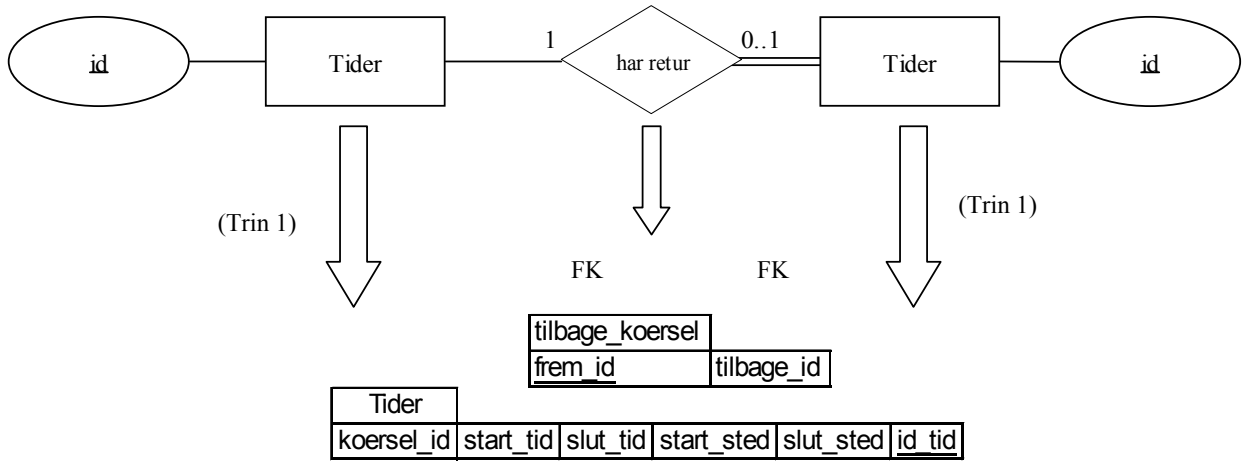
Binære 1:1 relationstyper

For hver binær 1:1 (ikke identificerende) relationstype R med relaterede entitetstyper S og T

- Udvælges en af S og T (foretræk den af entitetstyperne der måtte have total deltagelse i R)
- Inkluder T's primærnøgle som fremmednøgle i RS
- Inkluder R's simple attributter i RS

Tider entiteten i relation med sig selv

En kørsel på et tidspunkt kan godt have en returkørsel tilknyttet. Eksempelvis hvis en skoleklasse skal køres i skoven om formiddagen og hentes om eftermiddagen. Så i stedet for at oprette to forskellige kørsler, oprettes blot en kørsel, som så har et returtidspunkt.



Da det er samme tabel som indgår i relation med sig selv, duer det ikke at tage de simple attributter fra den ene tabel og lægge over i den tabel, som har total deltagelse i relationen. Derfor oprettes en ny tabel, som indeholder fremmednøgler til tider. Her er begge kandidatnøgler, da en kørsel jo ikke kan have mere end en udkørsel eller hjemkørsel, og derfor kun vil optræde en gang i denne tabel.

Trin 4

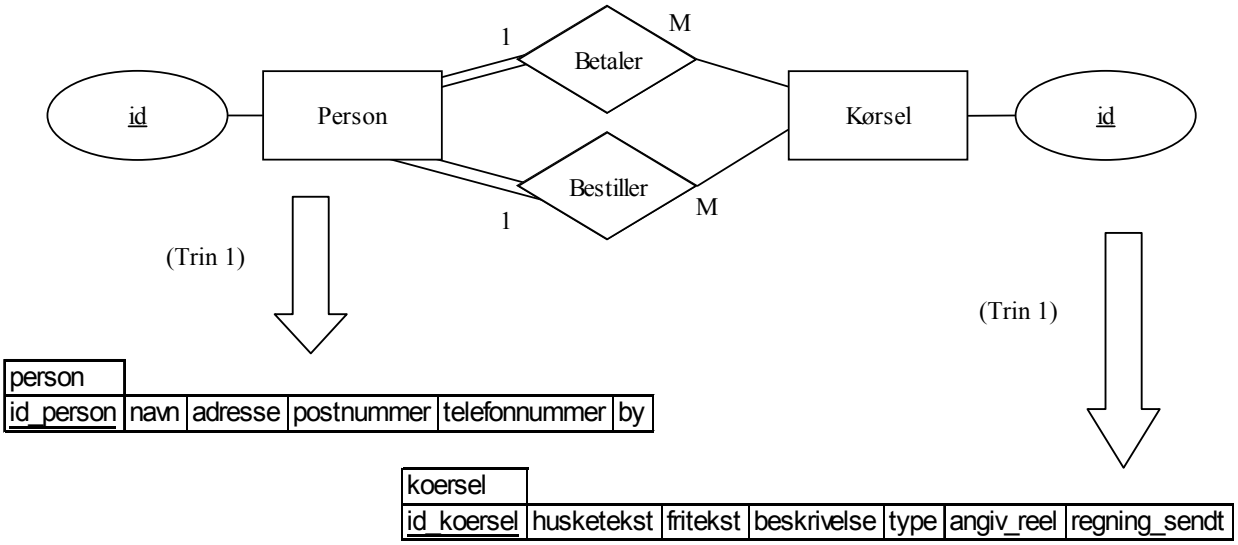
Regulære 1:N relationstyper

For hver binær 1:N (ikke identificerende) relationstype R, med relatere entitetstyper S (N-siden) og T (1-siden):

- Inkluder primærnøglen fra RT som fremmednøgle i RS
- Inkluder de simple attributter fra R i RS

Kørsel-Person relationerne

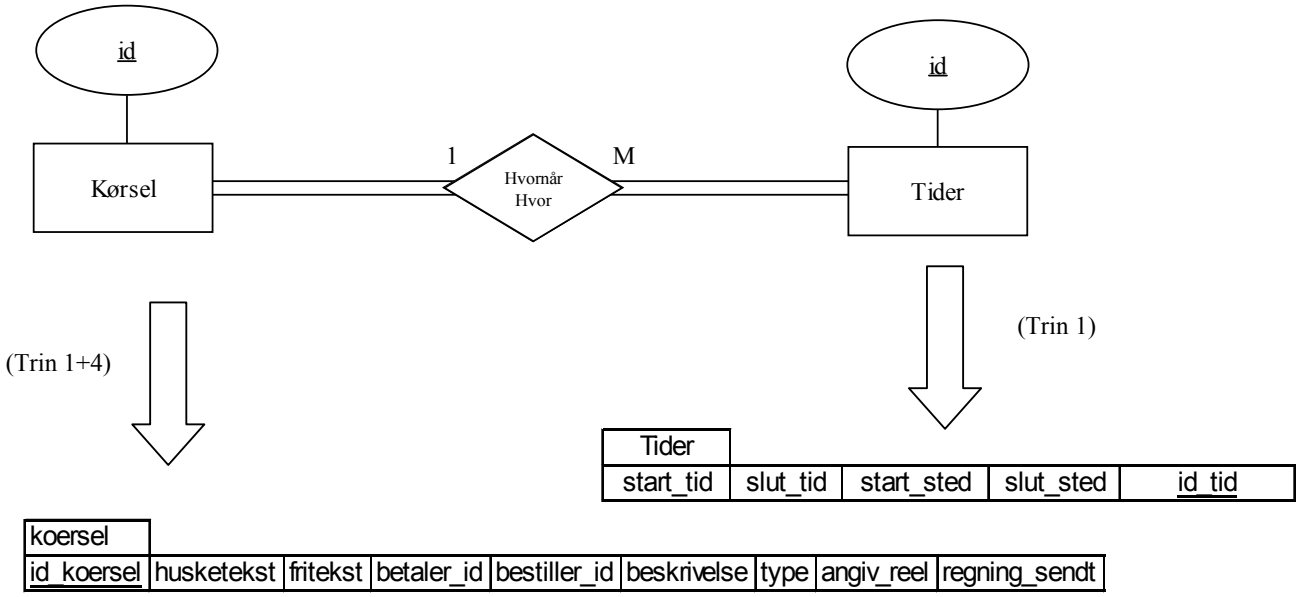
Relationerne bestiller og betaler mellem person og kørsel skal mappes hver for sig. Men da de næsten er ens, tages begge med i samme mapning



Dermed får koersel tabellen fremmednøgler betaler og bestiller som refererer til person tabellen, og kommer dermed til at se sådan ud:

koersel								
id_koersel	husketekst	fritekst	betaler_id	bestiller_id	beskrivelse	type	angiv_reel	regning_sendt
			FK	FK				

Kørsel-Tider relationen



Mange siden skal have fremmednøgle. I dette tilfælde er det tider, som får en fremmednøgle til koersel, og tabellen får følgende udseende.

Tider					
koersel_id	start_tid	slut_tid	start_sted	slut_sted	id_tid

FK

Trin 5

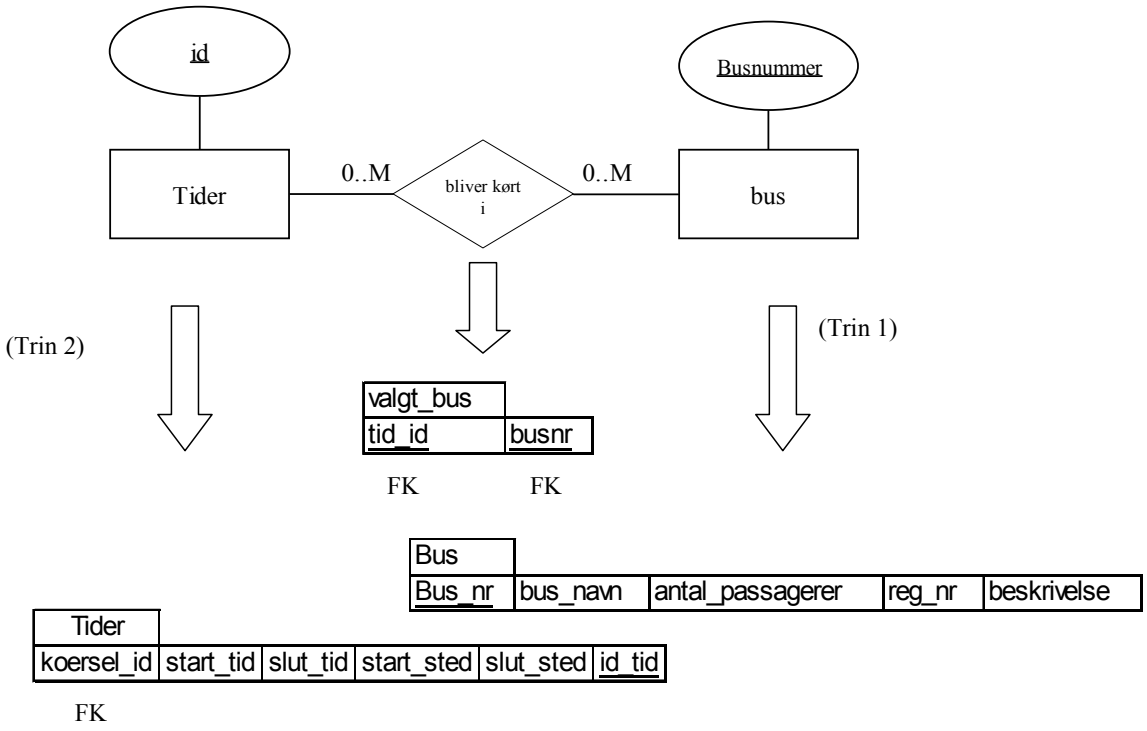
Binære M:N relationstyper

For hver binær relationstype R dannes en relation RR, der:

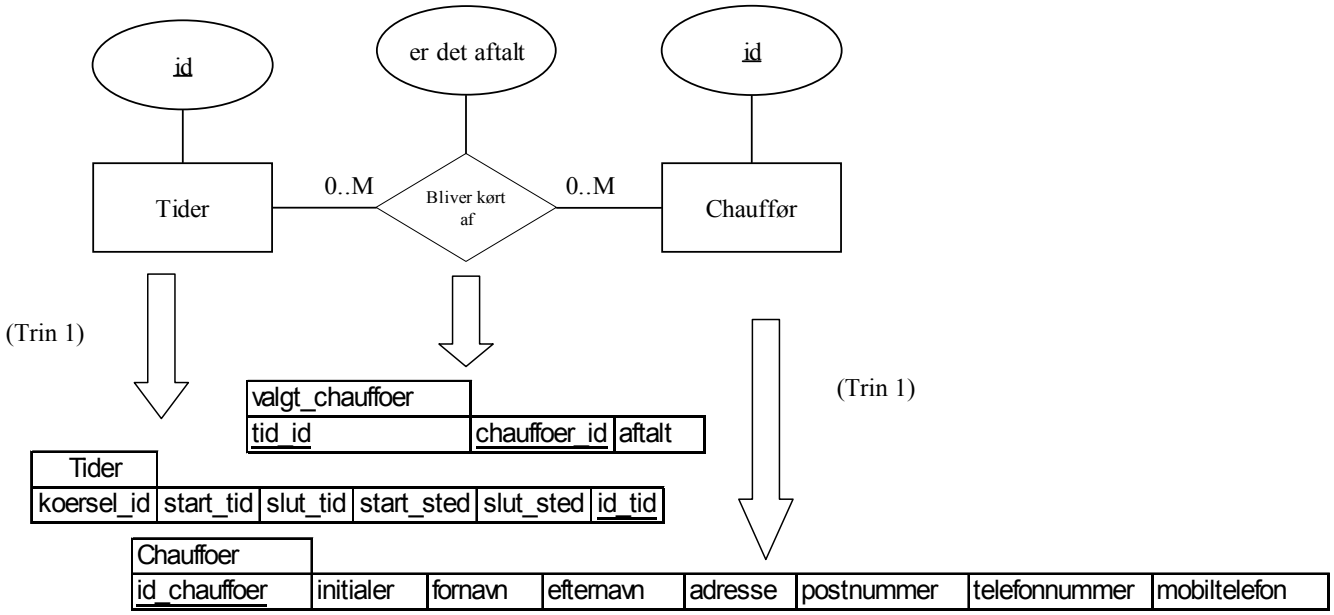
- Indeholder primærnøglerne for RS og RT som fremmednøgle
- De to fremmednøgler bliver primærnøgle for RR
- Indeholder de simple attributter fra R

Vores ER diagram indeholder tre M:N relationer.
Dette er ved tilknytningen af chauffører, busser og rejseledere til tider.
En kørsel på en bestemt tid kan have flere busser, chauffører og rejseleder tilknyttet og de kan også være tilknyttet flere kørsler.

Bliver kørt i relationen

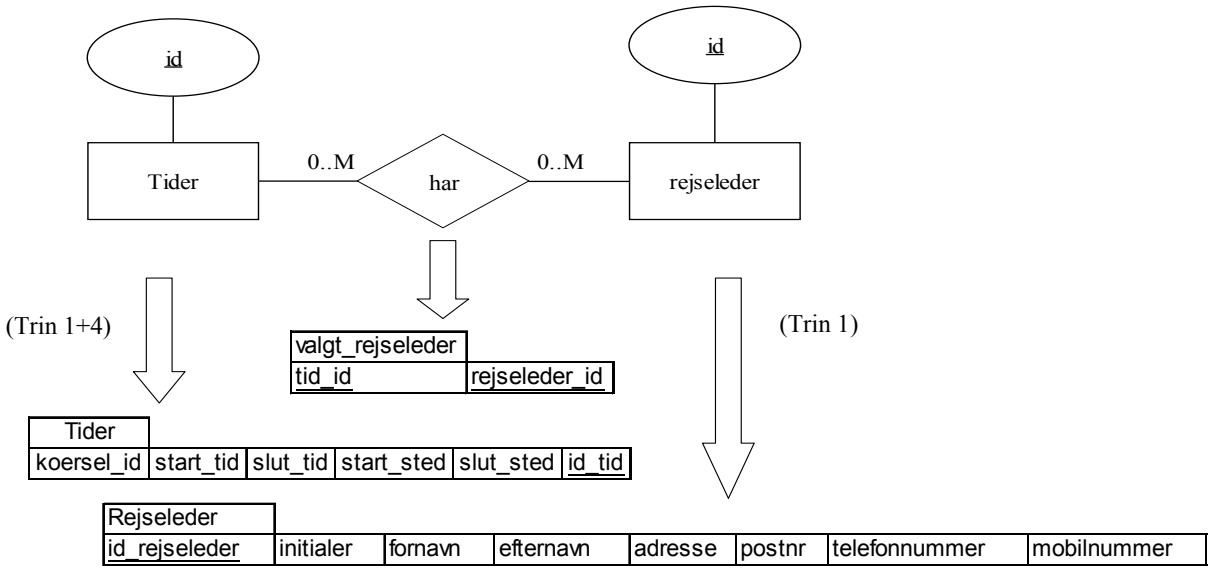


Bliver kørt af relationen



Her oprettes en ny tabel til at forbinde tider og chauffør tabellerne, som hedder valgt_chauffoer. Denne får relationens attribut ”er det aftalt”. Dermed er det muligt at registrere om chaufføren er underrettet om, at han skal køre den pågældende tur.

Har rejseleder relationen



Normalisering

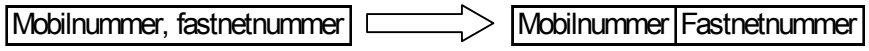
1. Normalform

Enhver tabel skal være en relation. Dvs. hver attribut indeholder atomare værdier.

Da vi tegnede ER-diagrammet, sørgede vi for at vores attributter kun indeholdt udelelige værdier. Dermed er vi allerede på 1. normalform.

Måske kan eksempelvis en chauffør i virkeligheden godt have flere mobilnumre, men det har vi valgt at se bort fra.

Eksempel:



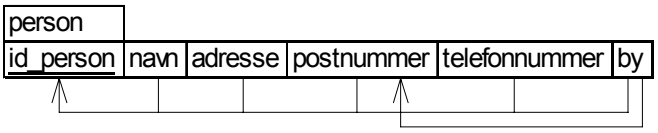
2. Normalform

En attribut skal være fuld funktionel afhængig af hele primærnøglen.

Alle vores attributter er fuld funktionel afhængige af hele primærnøglen og dermed på anden normalform.

3. Normalform

En attribut må ikke være transitiv funktionel afhængig af primærnøglen.



Figuren viser at alle attributter er afhængige af primærnøglen. Men "by" er bestemt af postnummer, og på den måde transitiv funktionel afhængig af primærnøglen. Så derfor kommer by ned i en tabel for sig selv, sammen med postnummer, og postnummer attributten i person bliver til en fremmednøgle i forhold til den nye tabel.

FK

Det samme sker med alle de andre tabeller, som indeholder attributterne postnummer og bynavn.

Modstand mod forandringer

Ved enhver væsentlig ændring i en virksomhed, enten organisatorisk eller teknisk, vil der opstå en vis modstand. Denne modstand bunder i det enkelte individs muligheder for at få sine behov dækket. Typisk vil modstanden tage udgangspunkt i manglende dækning af de nederste behov i Maslows behovspyramide.

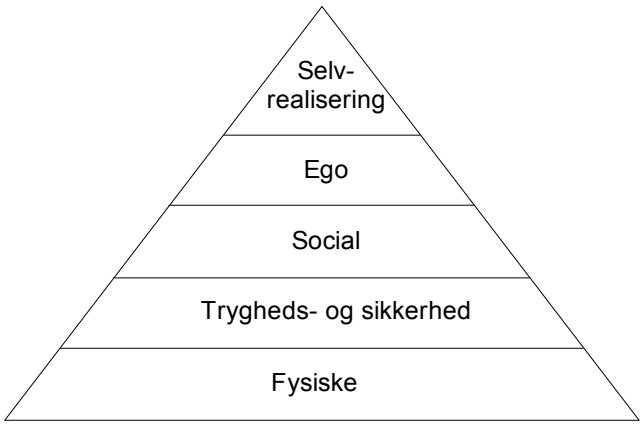


Fig. 6: Maslows behovspyramide

Hvis en produktionsvirksomhed investerer i en ny maskine, kan det blive nødvendigt at afskedige folk, som tidligere har arbejdet med det, som maskinen skal overtage. Dermed vil de folk, som står til at miste deres job reagere negativt mod indkøbet af denne nye maskine. Dette skyldes at de i fremtiden vil få problemer med at få deres behov opfyldt. Eksempelvis tryghedsbehovet, da de vil risikere at komme til at stå uden et job gennem en længere periode. Personer, som måske får lov at blive i virksomheden for at betjene den nye maskine, vil måske også føle at deres tryghedsbehov ikke bliver dækket. De kan måske blive stressede af at tænke på om de kan finde ud af betjene denne nye maskine. På den anden side vil andre føle mulighed for at få deres selvrealiseringsbehov dækket, ved at se betjeningen af den nye maskine som en udfordring. Dette kan også være tilfældet med vores system. Hvis man ser dette nye system som en udfordring.

Ved indførelse af et nyt computersystem bliver nogle medarbejdere måske overflødige, og skal derfor afskediges, og som allerede beskrevet, vil der opstå en vis modstand mod forandring. I vores tilfælde vil fire personer, som sidder på kontoret, blive berørt af indførelsen af dette nye system. Det er de fire personer, som primært skal bruge systemet. Da udviklingen af systemet er foregået i tæt kontakt med to af de fire, kan de andre to måske føle sig udenfor. De to tør måske ikke udtale sig

om egenskaber ved systemet, som de finder uhensigtsmæssige, når det er deres kolleger, som har været med til at udvikle systemet.

Derudover har de fire ikke samme forudsætninger for brug af en pc'er. Alle arbejder dagligt med firmaets booking system, men dette betyder ikke nødvendigvis at de har let ved at sætte sig ind i et nyt system. Der kan så opstå modstand mod forandring, hvis en eller flere personer føler sig usikre på, om de kan finde ud at bruge det nye system. Det er altså en mangel i dækningen af tryghedsbehovet, som resulterer i denne modstand. Hvis man i virksomheden ikke har talt om hvad det nye system skal bruges til, vil en medarbejder måske føle at vedkommendes stilling bliver overflødig, og derfor føle modstand.

En persons modstand mod et nyt system kan komme til udtryk på mange måder. Personen kan måske tage udgangspunkt i mindre fejl i systemet og kritisere dem, som om de betyder systemet er helt uanvendeligt. Bevidst forkert brug af systemet kan også være et udtryk for modstand. Hvis systemet fyldes med forkerte oplysninger, kan systemet gøres uanvendeligt. Ikke fordi systemet går ned, men fordi de data, der ligger i systemet, ikke er brugbare.

Systemet letter dagligdagen på kontoret, ved at flere personer kan arbejde i systemet på samme tid. Ved brug af den gamle papirbaserede kalender, kunne kun en person skrive oplysninger om nye kørsler ind. Oplysningerne er ikke blot tilgængelige for flere på en gang, men også nemmere at overskue. Ud over den almindelige kalender, som indeholdt få oplysninger om hver kørsel, var der på kontoret en mappe med supplerende oplysninger om de fleste ture. Disse oplysninger lægges også ind i systemet, og på den måde skulle systemet gøre dagligdagen nemmere på kontoret.

De øvrige medarbejdere i virksomheden udenfor kontoret vil ikke mærke meget til indførslen af det nye system. I stedet for at blive underrettet mundtligt om hvilke ture de skal køre, får de nu informationerne på et stykke papir.

Så ud fra en rent objektiv synsvinkel er systemet jo en væsentlig hjælp på kontoret. Men da mennesker er forskellige er det ikke sikkert alle ser sådan på det, hvilket kan resultere i forskellige af de nævnte problemer.

Brugervenlighed

Idet systemet vil blive en stor del af virksomhedens dagligdag, har vi lagt meget vægt på, at systemet bliver så brugervenligt som overhovedet muligt. Til dette formål bruger vi Rolf Molichs[5] definition af brugervenlighed:

- Let at lære
- Det at huske
- Effektivt at bruge
- Tilfredsstillende at bruge

Hvis systemet er meget kompliceret og tager lang tid at lære, kan vi risikere at Hanstholm Turistfart hurtigt opgiver at lære systemet at kende, og går tilbage til den måde de kender. For at sikre det, har vi for eksempel benyttet os af gestaltlovene, som vil blive belyst senere.

Effektiviteten er den vigtigste faktor i forbindelse med brugervenligheden i vores system. Den angiver, hvor lang tid det tager at udføre en bestemt opgave. Hvis man f.eks. skal igennem mange sider for at oprette en tur, eller hvis svartiden på systemet er meget høj, vil brugerne blive trætte af systemet, og i værste fald skrotte det. For at minimere svartiden, har vi sikret os, at systemet skal køre på en ny hurtig computer, som udelukkende er beregnet til dette formål. På den måde sikrer vi os også imod, at evt. andre processer vil sløve computeren eller i værste fald få den til at bryde ned. En anden måde at holde svartiden nede på er, at vi har valgt at bruge MySQL til at udvælge de data, som skal vises på siden, hvilket er et utrolig hurtigt og optimeret DBMS. Dette vil vi komme ind på senere under afsnittet MySQL.

For at sikre subjektiv tilfredshed, sørger vi for, at brugerne er med under hele udviklingsforløbet. Vi holder løbende møder med Hanstholm Turistfart, hvor de er med til at bestemme hvordan siderne skal se ud og hvilke informationer, der skal med på hver enkelt side. Under disse møder forsøger vi at få meninger fra alle de personer, der skal bruge systemet.

[5] "Brugervenlige edb-systemer", side 17

De fem gyldne regler

For at udvikle et system som brugerne anser for at være brugervenligt, har vi brugt Rolf Molichs "5 gyldne regler", som han anser for at være nøglen til at udvikle et brugervenligt system[6].

- Kend brugere
- Inddrag brugere
- Lær af andre
- Koordiner systemets dele
- Afprøv og ret systemet

Kend brugerne

Før vi overhovedet begyndte at udvikle systemet, havde vi et møde hos Hanstholm Turistfart, hvor vi blandt andet skulle finde ud af, hvem brugerne af systemet ville være, og hvilke arbejdsopgaver de udfører i løbet af en dag i forhold til planlægning af ture. Dette møde var baseret på interview, hvor vi forsøgte at klarlægge hvilke oplysninger, der bliver registreret om de forskellige ture. Samtidig fik vi et kopi af en side fra deres nuværende kalender med hjem, således vi kunne bruge den som en hjælp under udviklingsarbejdet.

Inddrag brugere

Idet vi har valgt at benytte os af den traditionelle projektmodel kombineret med faselinier fra den trinvise projektmodel, sikrer vi os, at vi har kontakt med brugerne under hele udviklingsforløbet, da vi holder møder ved hver trinafslutning.

Allerede tidligt i udviklingsforløbet havde vi designet selve brugergrænsefladen, dog med begrænsede funktioner. På den måde kunne de få et overblik over, hvordan det færdige systemet ville komme til at se ud, og samtidig kunne de komme med forslag til forbedringer. Ved at have en grænseflade færdigt så tidligt i forløbet gør, at forvirrende elementer vil blive rettet. Ellers vil disse være med til at forvirre under afprøvning af andre moduler, og måske være medvirkende til at et modul bliver kasseret.

[6] Brugervenlige edb-systemer, side 27

Lær af andre

Dette princip går ud på, at lade sig inspirere af andre eksisterende programmer, der arbejder med samme problemstilling som en selv. Dette kan både være positivt og negativt. Det vil sige, at man måske har set noget i et andet program, som man kan blive inspireret af, til at lave noget ligende i ens eget system. Vi har eksempelvis fundet inspiration i Microsoft Outlook til hvordan man angiver gentagne ture.

Det modsatte kan også være gældende, det vil sige, at man måske har set noget i et eksisterende system, som man kan sige til sig selv, at det skal vi i hvert fald lave anderledes i vores system.

Koordiner systemets dele

Det er vigtigt, at når der er flere programmører på samme system, at man husker at koordinerer begreberne mellem hinanden. I vores tilfælde er vi tre programmører, så derfor er det vigtigt, at hvis en ændrer i en overskrift eller et menupunkt, så skal det koordineres med de andre, sådan at der på siden for eksempel ikke henvises til et menupunkt som rent faktisk ikke eksisterer.

Afprøv og ret systemet

Grundet den projektmodel vi har valgt, vil der løbende være kontakt med kunden, hver gang et modul er færdigt. Her vil dette modul blive afprøvet, hvor de vil prøve at lægge nogle realistiske data ind på systemet. På den måde, vil man hurtigt kunne finde ud af, om der er noget der skal laves om.

Brugergrænsefladen

I dette afsnit vil vi beskæftige os med brugergrænsefladen. Her vil vi bruge Rolf Molichs retningslinier for udformning af skærmdialog[7].

- Gør systemet intuitivt forståelig
- Støt brugerens hukommelse
- Fortæl hvad der sker
- Vær hjælpsom, når brugeren har problemer
- Forebyg problemer

Gør systemet intuitivt forståelig

Intuitiv forståelse går ud på at få, at få billederne i systemet til at signalere hvilken funktion der gemmer sig bag den. Dette kan gøre, at brugerens opmærksomhed ledes hen på nogle muligheder han ikke var klar over der findes. Ifølge Rolf Molich kaldes dette Affordance[8], hvilket han mener er et at de vigtigste begreber inden for brugervenlighed.

Affordance er også en vigtig faktor i forbindelse med vores system, for at sikre sig at brugerne ikke bruger systemet forkert. Derfor har vi gjort meget ud af at gøre systemet så intuitivt som overhovedet muligt, dvs. at få f.eks. link og knapper til at sige så meget om deres funktion, at man ikke er i tvivl om hvad den gør. Et eksempel på dette kan ses på figur 7. Hvis man ønsker at udskrive siden, vil man intuitivt trykke på printer ikonet.

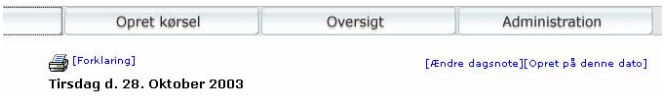


Fig. 7: Affordance

En anden ting, der er gjort for at gøre siden intuitiv forståelig, er at den er delt op i 4 hovedkategorier: Kalender, opret kørsel, oversigt, administration. På den måde, vil man hurtigt kunne finde ud af, hvilken kategori det man ønsker at foretage sig, hører under. Uanset hvilken side man er inde på, vil menuen i toppen altid være den samme, således man hurtigt kan komme tilbage til noget man kender, hvis man er "faret vild".

[7] "Brugervenlige edb-systemer", side 51

Støt brugerens hukommelse

Dette afsnit går ud at, at man skal støtte brugernes hukommelse, således at brugerne hele tiden bliver mindet om, hvad de forskellige menupunkter gør. F.eks. vil der i menuen, hvis man holder musen over knappen, komme en forklaring på hvad der vil ske, hvis man trykker på knappen. Det samme er gjort på kalenderen, hvor der er lavet et link [Forklaring], der vil åbne en pop-up side, hvor alle farver og symboler på kalenderen bliver forklaret.

En anden ting, der er gjort for at støtte brugerens hukommelse, er at vi benytter os at ensartethed. Dvs. at hver gang der skal indtastes en dato på siden, vil formatet altid være det samme.

Denne måde er også med til at nedsætte genindlæringstiden, hvis man har været væk fra systemet i længere tid.

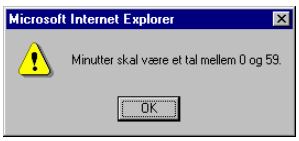
Fortæl hvad der sker

Tilbagemelding er en vigtig del af brugervenlighed. På den måde, er brugerne altid klar over hvad det er der sker. For eksempel når en bruger vælger at slette en chauffør, vil der når chaufføren er slettet komme en besked i toppen af siden, hvor der står: "Chauffør er blevet slettet.", hvilket gør at brugeren er helt sikker på at funktionen er blevet udført.

I tilfælde af, at der bliver lavet nogle kritiske ændringer, der kan skade systemet, som for eksempel sletning af busser, chaufførerer, ture og ikke mindst indlæsning af backup, vil der komme en boks op før kommandoen udføres, der fortæller hvad man er i gang med, og om man er sikker på at man ønsker at udføre denne kommando. Dette minimerer også risikoen for, at man kommer til slette nogle oplysninger ved et uheld.

Vær hjælpsom, når brugeren har problemer

I vores system gør vi brug af handlingsmeddelelse, hvis brugeren har indtastet et ugyldigt argument. F.eks. hvis man indtaster et minuttal, der ikke er imellem 0 og 59, vil der komme en meddelelse der

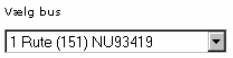


[8] "Brugervenlige edb-systemer", side 53

forklarer at minutter skal være mellem 0 og 59. Denne meddelelse forklarer præcis hvilket felt der er indtastet fejl i, og hvad gyldige værdier er.

Forebyg problemer

I stedet for at brugeren får en masse handlingsmeddelser, når han prøver at oprette en tur, ville det være bedre hvis man gjorde det rigtigt første gang. F.eks. hvis man ønsker at sætte en bus til at være optaget, har vi lavet således, at man skal vælge hvilken bus man ønsker at arbejde på, ved hjælp af en dropdown boks. Hvis det i stedet var lavet således at man selv skulle indtaste hvilket bus nummer man ønsker at arbejde med, ville der være mulighed for indtastningsfejl. På denne på sikrer man, at man kun kan vælge en bus, der findes i systemet. Dette princip bliver også brugt når man skal vælge chauffør, bus og rejseleder på en kørssel. Her vil der komme en popup, hvor man kan afkrydse for eksempel hvilke busser man ønsker, der skal køre denne tur. På den måde eliminerer man også muligheden for, at man kommer til at vælge en bus, der ikke eksisterer.



Design af brugergrænseflade

Idet systemet vil blive stor del af Hanstholm Turistfarts hverdag, og er noget de vil tilbringe meget tid med, er det vigtigt at designet er behageligt at se på. Derfor har vi forsøgt at holde designet så enkelt så muligt, uden et utal af animationer og farver, som man meget hurtigt bliver træt af at sidde og se på. Selve opbygningen af kalenderen og de overordnede menupunkter er designet med bløde afrundede hjørner, således at det skaber et pænt og behageligt skærmbillede.

Farvevalg

Vi valgt at bygge hele systemet op i en neutral grålig nuance, hvilket skaber et meget behageligt og roligt skærmbillede.

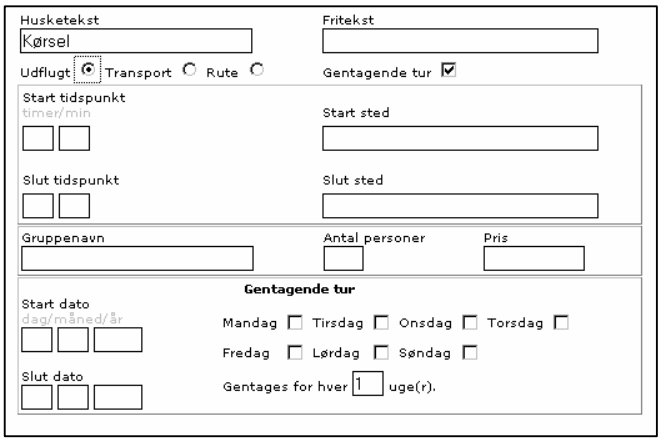
Rød er en farve der signalerer stop eller fare[9]. Denne har vi brugt på administrationssiden, når man skal indlæse en backup. Dette er en funktion, hvor der er en potentiel risiko for at man mister nogle oplysninger, hvis man kommer til at indlæse en gammel backup. Derfor har vi valgt at skrive en advarsel med rød ved siden af

[9] "Brugervenlige edb-systemer", side 70

menupunktet, hvilket resulterer i, at brugerens øjne vil falde på denne advarsel først, fordi denne farve adskiller sig så meget fra resten af farverne på siden.
Vi har også brugt farver til at fremhæve dage og måneder i kalenderen. For eksempel vil den aktuelle måned som bliver vist i kalenderen, være markeret med blå skrift i kontrast til de andre der er skrevet med hvid skrift. Det samme er tilfældet med dags dato, hvilken vil være fremhævet med en grønlig nuance.

Gestaltlove

Gestaltlovene hjælper med at få systemet til at hænge sammen visuelt, og siger noget om, hvordan vi opfatter helheder[10]. Ved at bruge nogle af disse love, kan man gøre systemet nemmere at overskue, og dermed også nemmere at lære.
I vores system har vi brugt lovene om lukkethed og nærhed. Loven om lukkethed siger, at symboler, der står i samme ramme, opfattes som hørende sammen. Når man opretter en tur, vil f.eks. alt hvad der vedrører gentagende ture, stå i en kasse for sig selv. Det samme er tilfældet for tidspunkter, tilbageture og køber/betaler. Derfor overholder disse også loven om lukkethed. På den måde ved man, at når man arbejder på noget, der står i en ramme for sig selv, så ved man at alt i den ramme er noget, der hører til det man arbejder med lige nu. For eksempel vil opret siden virke meget forvirrende, hvis der ikke er nogle rammer, idet det vil være meget svært at overskue de mange forskellige datofelter.



Reglen om nærhed siger, at symboler der er anbragt nær hinanden, opfattes som hørende sammen. Dette er tilfældet for knapperne Udflugt/Transport/Rute, som står ved siden af hinanden for at tydeliggøre, at disse knapper hører sammen.

[10] "Brugervenlige edb-systemer", side 76

Tænke højt afprøvning

Med udgangspunkt i bogen "Brugervelige edb-systemer"[11] vil vi beskrive processen ved at udvikle et både brugervenligt og funktionelt system vha. "Tænke højt afprøvning". Som Molich skriver vil der altid, lige meget hvor omhyggelig man er, være nogle alvorlige og uforudsete uhensigtsmæssigheder. Vi afprøver systemet mens den er i produktion og vi vil have flere brugersessioner for at inddrage brugerne mest mulig og få et så fleksibelt system som muligt.

Netop fleksibiliteten i systemet var noget som brugerne satte i højsædet, så vi mente det var vigtigt at få mange brugersessioner, for at danne os et nuanceret billede af brugerens ønsker og synspunkter på hvordan et brugervenligt, effektivt og fleksibelt system skal se ud.

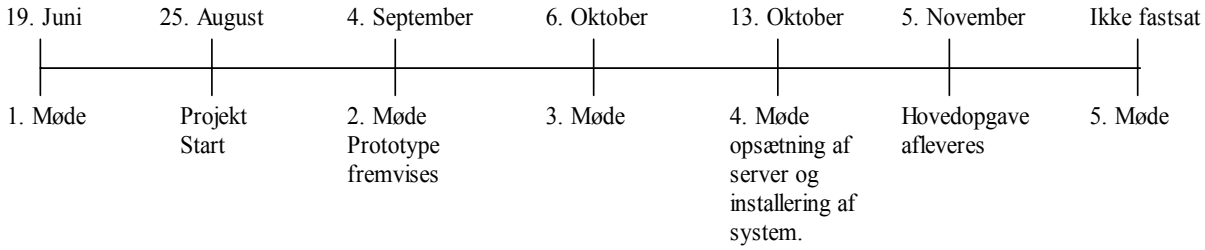


Fig. 8: Tidslinie over møder med Hanstholm Turistfart

En tænke højt afprøvning består af følgende trin (revideret udgave af bogens metode):[12]

- Udformning af opgaver
- Udvælgelse af deltagere
- Gennemførelse af afprøvningerne
- Dataanalyse

[11] "Brugervenlige edb-systemer", side 100.
[12] "Brugervenlige edb-systemer", side 102.

Udformning af opgaver

Det er en god ide, før man lader brugeren afprøve system, at opstille nogle brugerscenarier. Ved 2. møde, se figur 8, havde vi udarbejdet en prototype med en kalender, en foreløbig oversigt over kørsler og en form, hvor man kan oprette en kørsel. Vi havde derfor en opgave med generel navigering og det at oprette en tur. Ved 3. møde hvor system var cirka 90% færdigt var der udover en udvidet oprettelse af ture opgave også opgaver omkring generel administration og informationssøgning på oversigt.

Udvælgelse af deltagere

Vi har valgt at bruge udvalgte personer fra målgruppen som deltagere i vores tænke højt test. På kontoret er der i skrivende stund 4 personer inklusiv de 2 før nævnte. Derfor må det siges at de 2 deltagere danner et godt billede af den gennemsnitlige bruger af systemet.

Gennemførelse af afprøvningerne

Vi har valgt at fremlægge "opgaverne", som brugeren skulle udføre, mundtligt da vi ikke ville være for formelle og skabe en eksamensstemning. Derfor foregik afprøvningen ved at vi fremlagde en opgave, brugeren udførte eller forsøgte at udføre opgaven, imens vi observerede og tog notater. Efter brugeren har udført de valgte opgaver, tager man en såkaldt eftersnak med denne, om hvorledes den generelle oplevelse af systemet var, og om forslag til forbedringer.

Dataanalyse

Da systemet skal afspejle og forbedre den nuværende fysiske papirkalender på alle punkter, har brugerne en meget god ide til opbygningen af systemet og funktionaliteten. Derfor var det ikke noget problem at få brugerne til at beskrive manglerne ved systemet under afprøvningerne.

Ved 2. møde, hvor vi havde lavet en prototype, var ikke en decideret afprøvning, men mere et bud på opbygningen af systemet fra vores side, og en måde at få brugerne til at brainstorme med forbedringer og ikke mindst nye påfund. Brugerne havde mange ønsker til udvidelser af systemet.

Ved 3. møde, godt en måned efter vi fremviste prototypen, lavede vi en afprøvning med de 2 brugere, hvor de fik stillet opgaver verbalt og derefter kom de med forslag til forbedringer.

Sikkerhed

En utrolig vigtig faktor når man snakker webbaserede systemer, eller systemer der skal være adgang til via nettet, er sikkerhed. I dag kan enhver person for et mindre beløb få adgang til Internettet og er derved en potentiel trussel for Internetvirksomheder og privatpersoner, der udveksler følsomme oplysninger på Internettet. Derfor vil vi beskrive hvordan vi bedst kan sikre os imod ubudne gæster på systemet, og analysere hvilke mulige løsninger der er nødvendige og realistiske, for systemet.

Misbrugere af systemet

Interne brugere

Ved denne gruppe af brugere er der hovedsagelig tale om, at "angrebene" på systemet/serveren sker ved en fejltagelse eller af ren og skær uvidenhed. F.eks. kan en person på Hanstholm Turistfart komme til at slukke for serveren på et tidspunkt, hvor denne er i brug, og derved beskadige eller gøre data inkonsistent. Et andet eksempel kunne være ved forkert indsættelse af data, dette tages der dog hånd om fra vores side af, ved hjælp af diverse tjek af brugerinput og formatering af data, der skal gemmes i databasen. Vi har lavet det sådan, at brugerne skal logge ind via en login startside, før de kan benytte systemet.

Endvidere har vi lavet to bruger niveauer, 1 og 2. 1 er det højeste niveau, hvor en bruger kan oprette/ændre og slette brugere. Hvorimod niveau 2 ikke kan dette og heller ikke indlæse backup, da dette vil overskrive databasen, men kan dog stadig gå ind og ændre sit eget password. Begge niveauer kan dog foretage backup.

Crackere

Denne gruppe af personer er langt sværere at tage højde for, blandt andet fordi der ikke er et system der er 100% sikret imod angreb over Internettet. Dog vil vi bruge diverse hjælpemidler som PHP, MySQL og Linux generelt stiller til rådighed, for at gøre systemet så sikkert som mulig.

Vi vil gøre brug af login så brugerne fra Hanstholm Turistfart kan komme ind på systemet hjemmefra, ved hjælp af denne, dette gør at system kan blive genstand for angreb af hackere. Dog prøver vi at mindske denne risiko ved blandt andet at køre en MD5 hash på det password brugerne skal benytte sig af og gemme dette i databasen. Ud over dette laver vi brugerniveauer, så det kun er

”superbrugere”(højeste brugerniveau ud af 2), der kan oprette andre brugere og sætte deres brugerniveau.
Ud over dette har vores Apache server SSL support hvilket vil sige at der krypteres med et 128bit på alle informationer der sendes imellem browseren og serveren, der sikrer at en cracker ikke kan opsnappe og udnytte informationerne der bliver sendt imellem disse, blandt andet password og information om diverse database opslag og oprettelser. SSL vil vi dog komme nærmere ind på andet steds.

Konkurrenter

Ligesom crackere, kunne man forestille sig at konkurrenter kunne have til formål at bryde ind på serveren eller finde det korrekte brugernavn og password, så de kunne komme ind på systemet og slette, spionere eller bare lave om på vigtige data.

Tyve

En faktor, der mere berører udviklerne end brugerne af systemet er eventuelle tyve af selve systemets kode, til videresalg eller brug af mindre dele af koden i egne systemer. Derfor har vi valgt ikke at installere Telnet på Linux-serveren, da dette ville kunne give brugere udefra adgang til serveren og dermed alle filer på denne.

Debian software stadier[13]

I Debian har man mulighed for at vælge, hvilke type pakker man ønsker at installere og opdatere.

Woody (stable)
Den stabile version af Debian, denne version udgør den nyeste udgave af Debian. Her er de lidt ældre, men mest robuste og gennemtestede pakker.

Sarge (testing)
Den næste udgave af Debian, som er under konstant udvikling, sarge erstatter woodys plads som stable i fremtiden. Pakkerne er nyere end i stable, men lidt ældre end unstable.

Sid (unstable)
En version af Debian der aldrig bliver udgivet. Her kommer de nyeste pakker fra Debian udviklerne ind, hvorefter de senere bliver sendt videre til sarge/testing når de er stabile nok.

Ud fra dette har vi valgt, at vores server skal kun bruge woody pakker. Dermed er vi sikre på at disse har være gennemtestet og de fleste sikkerhedshuller er fundet.

Sikkerhedsproblemer på et netværk
Ifølge Andrew S. Tannenbaum i bogen "Computer Networks" kan sikkerhedsproblemer på et netværk deles op i 4 grupperinger nemlig secrecy, authentication, nonrepudiation og integrity control. Vi vil efterfølgende beskrive disse grupperinger med henblik på at klarlægge hvilke skridt vi har taget for at minimere en eventuel sikkerhedsrisiko.

Secrecy
Her er der tale om hemmeligholde passwords og andre hemmelige data, når de bliver sendt over netværket, dette gøres nemmest ved at kryptere det sendte eller sende det igennem såkaldte sikre forbindelser.

[13] Kilde: http://debianguiden.dk/dists/stable/html/apt-og-dpkg.html

Da vi benytter Linux, som er OpenSource, er kildekoden tilgængelig for alle. Dette udgør en sikkerhedsrisiko da det gør det nemmere for eventuelle crackere at bryde ind i et system hvor de kender kildekoden. Derfor bør man undersøge hvilke services man skal slå fra, for at minimere denne risiko.
Telnet – er en service hvor med man kan logge ind på serveren, i dette tilfælde, og udføre kommandoer som hvis man sad ved computeren, der skal kun bruges brugernavn og password for at gøre dette. Vi har valgt ikke at installere dette da det ville være katastrofalt hvis det lykkedes for en uvedkommende at logge ind på serveren, da dette vil give adgang til alle vores data på denne.
FTP – bruges til at kopiere filer fra og til maskinen. FTP sender brugernavn og password som klartekst og dette er derfor "nemt", for en mere eller mindre professionel cracker, at få fingrene i. Vi har valgt at lukke ned for denne tjeneste, for helt at undgå denne problematik.

Til at kryptere har vi brugt MySQL´s egen metode til dette, nemlig PASSWORD()[14], dette er dog en hash funktion og ikke en decideret kryptering. Som man kan se på nedenstående eksempler udføres dette ved simpelthen at kalde metoden i selve Sql strengen, med det password man vil hashe som argument. MySQL versioner mindre end 4.1 laver en hash værdi, der er 16 byte lang, f.eks. 6f8c114b58f2ce9e, dog er der i den nye MySQL 4.1 gjort brug af en 41 byte lang værdi, dette er selvfølgeligt mere sikkert, men i skrivende stund er denne ikke i en stable(Woody) udgave fra Debians side, derfor vil vi ikke gøre brug af den endnu.

Nedenfor er der to eksempler på brug af PASSWORD() metoden. Nummer 1 bliver der oprettet en ny bruger og nyt hash password bliver lavet. Nummer 2 bliver brugt i forbindelse med login af bruger hvor $passwd variablen er det password som brugeren har indtastet, hvorefter dette bliver hashet af MySQL metoden og sammenlignet med det password der ligger i databasen sammenhørende med de indtastede initialer.

```
INSERT INTO login(initialer, password, adgang) VALUES
('$initialer',PASSWORD('$password1'),'$adgang')
```

```
SELECT * FROM login WHERE initialer = '$login' AND password =
PASSWORD('$passwd')
```

[14] Kilde: http://debianguiden.dk/dists/stable/html/apt-og-dpkg.html

Som nævnt gør vi brug af en Apache webserver, hvor vi har installeret et SSL(Secure Socket Layer)[15] modul, der gør at informationerne der bliver sendt imellem klienten og serveren bliver krypteret med 128bit. SSL bliver f.eks. også brugt i sammenhæng med Dankort transaktioner, hvor man typisk på betalingssider går ind i en tilstand hvor dataene bliver krypteret før afsending.

Authentication

For at sikre at der ikke er nogen uvedkommende personer, der bruger systemer, har vi valgt at beskytte det med login. Som skrevet om under secrecy er det vigtigt at kryptere/hashe password og brugernavn, men dette er dog intet værd, hvis man ikke er sikker på at det er den rigtige person der logger ind. Derfor er det vigtigt at f.eks. Susan (administrationen) ikke bruger 123 eller sin hunds navn som password, men hellere bruger både tal og bogstaver i en sammenhæng der ikke er åbenlys.

Vi har derfor lavet, ved hjælp af session i PHP(nedenstående kode), en test af hvor mange gange en bruger af systemet har tastet forkert brugernavn eller password ind. Man kan også se, hvordan der bliver slået op i databasen og tjekket på rigtigheden af brugernavn og password. Nedenstående eksempel viser hvorledes vi har implementeret test af login.

```
//Funktion til at teste om brugernavn og password er korrekt.
Function auth($login = '', $passwd = '')
{
  //Variablen gøres global så den er tilgængelig udenfor funktionen auth.
  global $login_data;
  //$check bliver sat til true hvis $login ikke er tom eller ikke eksisterende.
  $check = ! empty( $login );

  //Tjekker om session array eksisterer, hvis det gør returneres true og der
bliver ikke tjekket på login
  if (is_array($_SESSION['login_data']))
  {
    return true;
  }
  //Hvis check variablen ikke er tom.
  elseif ( $check )
  {
    //Finder om der er et sted i tabellen login hvor der er overenstemmelse
imellem initialer og password. Her bruges før nævnte MySQL metode PASSWORD().
    $resultat=mysql_query("SELECT * From login WHERE initialer = '$login' AND
password = PASSWORD('$passwd')");

    //Hvis antallet af rows fundet ikke er 0.
    $row = mysql_fetch_array($resultat);
    if ($row)
    {
```

[15] Kilde: http://httpd.apache.org/docs-2.1/ssl/ssl_faq.html

```
      //Associativt array login_data bliver oprettet og registreret som session.
      $login_data = array("login"=>$row['initialer'], "adgang"=>$row['adgang']);
      session_register( 'login_data' );
      session_unregister('forsoeg');
      return true;
    }

    //Hvis session forsoeg ikke er sat sættes den til 1.
    if (!isset($_SESSION['forsoeg']))
    {
      $_SESSION['forsoeg'] = 1;
    }
    //Session forsoeg tælles 1 op ved hvert mislykkedes forsøg.
    else
    {
      $_SESSION['forsoeg']++;
    }
     //Session associativt array bliver sat til ingenting.
     unset( $login_data );
     return false;
    }
    else
    {
      return false;
    }
}

//Ovenstående auth() funktion bliver kaldt med brugernavn og password fra form
som argument.
if ( !auth($_POST['username'], $_POST['password'] ) )
{
  //Hvis session variablen forsoeg er lig med eller over 3 udskrives en
fejlmeddelelse og brugeren skal lukke browseren for at få 3 nye forsøg.
  if ($_SESSION['forsoeg'] >= 3)
  {
    echo "<center><b>Af sikkerhedsmæssige årsager låser systemet efter 3
mislykkede login. For at fortsætte skal du lukke din browser og starte
forfra.</b></center>";
    exit;
}
  //Loginform funktionen bliver kaldt med enten true, for mislykket login, eller
false, for succesfuld login.
  loginform( isset( $_POST['username'] ) );
}
```

Non-repudiation

Digital signatur er hvad der er tale om her, da det i visse tilfælde er vigtigt at være sikker på at det, der bliver sendt ikke er blevet lavet om, eller er læst/brugt af en uvedkommende person. Dette er hovedsageligt i sammenhæng med online betalinger. Det foregår ved at afsenderen bruger sin private nøgle og modtagerens offentlig nøgle til at kryptere det, der skal sendes, derefter skal modtageren bruge sin private nøgle, derefter afsenderens offentlige nøgle til at dekryptere det sendte og er dermed sikker på hvem afsenderen er, da nøglerne er unikke.

At have dette implementeret i vores system er unødvendigt, da vi ikke, udover login, har brug for at registrere handlinger via brugeridentifikation. Dog registrerer vi hvilken bruger, der har sendt en regning for at man kan finde tilbage til den person, hvis der er sket en fejl.

Integrity control

Her er der tale om, at brugerne skal være sikre på at informationen, der bliver sendt imellem dem og andre brugere eller en server, på et netværk, ikke er blevet ændret. Dette har vi brugt når vi genererer en backup af databasen i en tekstfil, som brugeren selv vælger placeringen på. Denne fil er en almindelig tekstfil, hvor diverse kommandoer til at gendanne databasen fra tidspunktet for backup, står i. Først bliver alle data og tabeller slettet, hvorefter alle data og tabeller fra backup tidspunktet bliver indlæst.

Et sikkerhedsproblem med denne backup metode kunne være hvis en person forsøgte at ændre tekstfilen, som bliver genereret, f.eks. kunne en ondsindet bruger finde på at skrive sin egen backup fil, hvor alle tabeller blev slettet i. Derfor har vi i toppen af hver backup fil lavet en checksum, af alle data i filen, der derefter er blevet hashet med PHP metoden crypt(), hvor der i koden er indskrevet en nøgle, der skal bruges til at hashe med.

```
...
//Der udføres en dump af databasen til en fil på serveren der kommer til at hedde, f.eks. 10657704562.sql, der er sekunder siden 1970.
exec("mysqldump -u dbuser –pkodeord --add-drop-table -B ht_web > ".$filnavn);
...
//Der laves en MD5 hash af alle data der er udtrukket af databasen.
$md5 = md5($data);
//En indskrevet "nøgle" bliver brugt i crypt metoden, hvor MD5 summen bliver krypteret.
$password = "hj4HTklh3kj5Y";
$md5 = crypt($md5, $password);
...
```

Når brugeren vil indlæse en backup, som denne har liggende lokalt, gøres dette via en menu under administration. Punktet ”indlæs backup” er kun tilgængeligt for brugere med brugerniveau ”fuld adgang”. Dette har vi valgt at gøre, da det ville være katastrofalt hvis en uerfaren bruger kom til at indlæse en backup af ældre dato, og der samtidigt ikke var taget backup siden de sidste ændringer var blevet foretaget. Det skal dog siges, at det ikke er nødvendigt for brugere at, manuelt, tage backup, eksempelvis efter endt arbejdsdag, da der på serveren bliver lavet automatisk backup hver nat.

Når der indlæses en backup lokalt fra brugeren, tjekkes der i koden, om der er blevet ændret i backupfilen, ved på samme måde som da backup filen blev lavet, at den krypterede MD5 sum hentes ud af filen og gemmes i en variabel. Derefter foretages en tjeksum af de resterende data. Denne tjeksum krypteres på samme måde som da man foretog backuppen med samme kodeord. Herefter sammenlignes de to krypteringer, og hvis de er ens, er man sikker på, at der ikke er blevet ændret i koden.

MySQL

Generelt

MySQL er den mest populære Open Source SQL database i dag[16]. Dette skyldes at den er meget hurtig, pålidelig og nem at bruge. At den er Open Source betyder, at alle kan downloade den fra Internettet og bruge den, uden at skulle betale noget. Hvis man ønsker at ændre noget i selve koden, må man også dette.
MySQL har API til C, C++, Eiffel, Java, Perl, PHP, Python, Ruby og Tcl, hvilket også var en af grundene til at vi valgte at benytte os af PHP.
SQL står for "Structured Query Language", hvilket er den mest brugte fælles standard for at tilgå databaser.

Administration

Her vil vi kort gennemgå opsætningen af MySQL-serveren og hvilke overvejelser vi har gjort os i forbindelse med opsætningen.

Et vigtigt princip i afvikling af programmer på en server er, at man aldrig skal give brugeren flere rettigheder end han behøver. Grunden til dette er, at hvis det skulle lykkes en cracker at finde en bagdør i et program, der bliver kørt som root, vil han lige pludselig have adgang til hele systemet, og vil kunne forsage stor skade.
Derfor bruger vi scriptet "mysqld_safe", der blandt andet starter MySQL-serveren med en anden uprivilegeret bruger. Scriptet sørger også for at logge fejl og genstarte MySQL-serveren, hvis der skulle opstå fejl.

En anden ting vi har gjort for at sikre MySQL-databasen mod angreb udefra er, at vi har tilføjet parameteren " skip-networking" til konfigureringsfilen, hvilket gør at den eneste, der har tilladelser til at bruge serveren er "localhost" (computeren selv), hvilket også er det eneste, der er behov for, idet webserveren også kører lokalt.

[16] http://www.mysql.com/documentation/mysql/bychapter/manual_Introduction.html

Idet MySQL ikke kommer med nogen grafisk brugergrænseflade, har vi valgt at bruge phpmyadmin[17] til at administrere databasen, som er et grafisk værktøj der er udviklet i PHP, til at administrere MySQL-databaser. Dette har vi gjort for at bevare overblikket over databasen, idet den ellers hurtigt kunne blive uoverskuelig at administrere under udviklingsfasen.

Samtidighed

Samtidighed går ud på, at håndtere at flere brugere manipulerer de samme data på samme tid. Dette punkt er ikke noget vi har taget meget hensyn til i vores system. F.eks. vil der opstå problemer hvis 2 brugere går ind og vil ændre den samme tur på samme tid. Hvis f.eks. den ene bruger går ind for at ændre en chauffør, på samme tid som en anden bruger også går ind for at ændre nogle oplysninger. Nu vil de 2 brugere sidde med det nøjagtig samme skærmbillede. Når den ene bruger er færdig med at ændre de oplysninger han skal, vil han trykke gem, og oplysningerne vil blive gemt i databasen. Indtil videre opfører systemet sig som det skal, men nu opstår problemet, at den anden bruger stadig sidder med et skærmbillede, hvor de gamle oplysninger stadig figurerer. Når den anden bruger så trykker på gem, vil den første brugers ændringer blive overskrevet med de nye oplysninger, hvilket kaldes ”lost update”.

Vi har ikke gjort noget for at afhjælpe problemet med lost update, men en måde at gøre det på, er at lade system kontrollere om de gamle og de eksisterende data er ens, før den begynder at opdatere, og hvis de ikke er det, vil der komme en besked til brugeren, der forklarer at oplysningerne er blevet ændret af en anden bruger.

Dog er der enkelte steder, vi har taget højde for samtidighed. Idet MySQL understøtter transaktioner har vi valgt at lave enkelte transaktioner. F.eks. har vi valgt, at når man sletter en tur, vil dette blive udført som en transaktion. Dette har vi gjort fordi, man for at slette en tur, skal udføre en række SQL-sætninger for at få slettet alle informationer omkring en tur fra alle tabeller. Hvis dette ikke var gjort, ville det kunne forekomme, at man har slettet turen fra nogen af tabellerne, men samtidig er der en der opdaterer kalenderen, vil turen stadig forekomme selvom turen rent faktisk er ved at blive slettet, og dermed kan der opstå en række fejl på siden. Når man udfører noget som en transaktion, vil ændringerne ikke fremgå for andre brugere før det hele er blevet udført (før

[17] http://www.phpmyadmin.net/

COMMIT bliver kaldt). For at lave en transaktion i MySQL skal man lave omslutte sine SQL-sætniger med følgende:

```
BEGIN
//SQL-sætninger
COMMIT
```

MySQL er multi-threaded, hvilket betyder at den kan udføre flere ting på en gang, men samtidigt sørger MySQL også selv for, at udelukke 2 tråde fra at skrive til den samme tabel på en gang, hvilket betyder, hvis en tråd er ved at skrive noget til en tabel, vil alle andre tråd vente indtil den første tråd er færdig med at bruge tabellen. Dette betyder, at vi ikke behøver at tænke på samtidigheds-kontrol i forhold MySQL serveren.

Samtidigt holder Apache/PHP også selv styr på, hvilke variable der hører til hvilke bruger, således at selvom 2 brugere åbner den samme side på samme tid med forskellige variabler, finder den selv ud af hvilke, der hører til hvem.

Backup

Uanset hvor mange forholdsregler man tager på serveren i forhold til at sikre sine data, kan man aldrig sikre dem 100 procent. Der vil altid være risiko for brand, tyveri eller at serveren brænder sammen. Derfor har vi implementeret muligheden for at foretage en backup af hele databasen fra systemet. På denne måde kan man kopiere en backupfil ud på et transportabelt medie, som f.eks. en CD-ROM, og tage den med hjem. Hvis det så skulle ske at serveren brænder sammen, vil man så kunne geninstallere programmet på en ny computer, og derefter indlæse backuppen, hvilket vil medføre at man genskaber systemets tilstand fra lige nøjagtig det tidspunkt hvor man foretog backuppen. På den måde kan man minimere tabet ved systemnedbrud, hvis man foretager backup ofte nok.

Til dette formål benytter vi os af et ekstern MySQL program, der hedder mysqldump. Dette program kan bruges til at ”dumpe” hele databasen, dvs. skrive hele databasen i en fil i SQL form. For at kalde dette program, bruger vi PHP-metoden ”exec”, hvilken bruges til at kalde eksterne processer på.

```
<?php
...
exec("mysqldump -u dbuser –pkodeord -add-drop-table -B ht_web > ".$filnavn);
...
?>
```

Denne backup foretager de hos Hanstholm Turistfart hver dag inden de går hjem, hvilken de brænder ud på en CD-ROM og tager med hjem.
Dog er der også lavet på maskinen, sådan at foretages en backup lokalt hver nat kl. fire ved hjælp af cron. Dette er dog blot en ekstra sikkerhedsforanstaltning, for at sikre, at hvis man kommer til at slette hele databasen, og man ikke kan finde backuppen eller mediet er ulæselig eller gået tabt. Så vil der stadig være en backup liggende på serveren, som man kan bruge til at læse ind.

Samtidigt med at der hver nat foretages en backup, findes der en parameter ”—log-update” som vi har valgt at tilføje når vi stater serveren op. Denne gør, at alle ændringer i databasen bliver skrevet i en logfil. Hver nat når der foretages fuld backup, vil denne logfil slettes, således at den starter på en ny fra det tidspunkt som der foretages fuld backup. På den måde, vil man kunne genskabe databasen helt op til det tidspunkt, hvor den blev smadret.

For at indlæse en backup igen, bruger vi SQL programmet MySQL. Ved at dirigere backup-filen til standardinput på MySQL programmet vil hele filen blive eksekveret.

```
<?php
...
exec ("mysql -u dbuser –pkodeord < ".$filnavn);
...
?>
```

Programmeringssprog

PHP

PHP står for "PHP Hypertext Preprocessor", hvilket er et server-side script sprog specielt tilegnet til at udvikle webapplikationer[18]. At sproget er server-side betyder, at scriptet bliver eksekveret på serveren, og kun outputtet bliver sendt til brugerens browser, hvilket gør det ideelt til f.eks. databaseadgang, idet man på den måde kan styre, hvilke oplysninger en bruger må se.

Selvom vi alle i gruppen havde begrænset erfaring med PHP før vi startede med projektet, kom det meget hurtigt, idet selve syntaksen lignede Java meget, som vi til gengæld havde meget erfaring med. Samtidigt findes der meget litteratur såvel fysisk som online der omhandler PHP, så det var ikke noget problem at finde hjælp til diverse problemer, der opstod undervejs.

Ligesom MySQL er PHP også "open source", dvs. gratis. Men fordelen er også, at når koden er tilgængelig for alle, findes der store grupper der udvikler på det, så sikkerhedshuller og fejl findes og rettes hurtigt.

PHP som CGI eller modul

CGI er et interface til at eksekvere programmer. Når man bruger PHP på denne måde, starter webserveren PHP med en selvstændig proces ved siden af webserveren. Altså PHP starter, CGI får PHP filer fra serveren, den returnerer HTML til webserveren, derefter lukkes PHP processen ned og webserveren sender det rene HTML til browseren.

Ved at bruge PHP som et modul er der ikke længere tale om en separat proces. Her er PHP tæt kædet sammen med webserveren, Apache i vores tilfælde, så Apache og PHP kører som én proces. Altså når webserveren modtager en GET fra browseren fortolker Apache, ved hjælp af PHP, scriptet og sender den rene HTML tilbage til browseren.

[18] http://dk.php.net/manual/en/introduction.php

Som man kan se på figur 4 har vi valgt at installere PHP som et modul, da det på det stærkeste opfordres til at man undgår at køre PHP som CGI.

"Note, we consider installing PHP like this suicidal."[19]

Som beskrevet laver CGI en selvstændig proces til hver forbindelse fra hver bruger. Dette gør, at ved flere brugere ville webserveren kunne blive overbelastet. I forhold til PHP som modul hvor der stadig kun er en proces, i sammenhæng med Apache, for alle forbindelser. Udover hastighedsproblemer er sikkerheden ringere i PHP CGI, da folkene bag PHP har valgt at dedikere deres tid til modul baseret PHP. Det vil sige at der kan være problemer og sikkerhedshuller i det mindre udviklede på, CGI.

Javascript

I modsætning til PHP er Javascript client-side, hvilket vil sige, at det udføres på brugerens maskine. Dette bruger vi bla. til at udføre kontrol af indtastninger, således at man ikke skal sende alt til serveren, før kontrollen udføres.
Javascript bruges også til dynamisk at opdatere en side uden at genindlæse. Dette bruger vi til at skjule og vise oplysninger på f.eks. kalenderen.

Ulempen ved at bruge Javascript er, at brugerens browser skal understøtte Javascript for systemet kan afvikles korrekt. Men idet, at de kun bruger Microsoft Internet Explorer hos Hanstholm Turistfart, er dette ikke noget problem, idet denne understøtter Javascript.

[19] Uddrag af PHP dokumentation Bilag nr. 3

Linux (Debian)

Der findes kun en Linux-kerne, men et utal af Linux-distributioner[20]. Kernen er den, der styrer processer, adgang til disk og meget andet. Derimod er distributionen det firma/organisation, der står for samlingen og udgivelsen. Forskellen på distributioner kan være måden programmer er pakket, installationen og konfigureringsværktøjer. Nogle af de mest brugte distributioner er:

- Red Hat
- Mandrake
- Debian

Vi har valgt at benytte os af Debian. En af grundene til dette, er blandt andet at den indeholder et godt pakkesystem, som gør det utroligt nemt at installere software-pakker. Samtidigt er det meget sikkert. Debian går meget op i at holde distributionen så sikker så muligt, det vil sige, at der normalt ikke mere går end 48 timer efter et sikkerhedshul er blevet rapporteret til at der er en pakke ude, der løser dette problem[21]. Samtidigt kan man automatisere installeringen af sikkerhedspakker ved hjælp af programmet apt-get, således at den f.eks. hver nat undersøger om der en nogle nye sikkerhedspakker til noget af det software der findes på maskinen. Dette har vi gjort på den installation vi har lavet til Hanstholm Turistfartm, idet det er en server der står på Internettet, og dermed et muligt mål for crackere.

En installation af Linux består af flere partitioner. En minimal installation består af en partition til swap og en partition til rod-filsystemet ”/”. I vores installation har vi valgt at lave endnu en partition til ”/var”. Denne partition indeholder alle html- og database-filer. På den måde kan man geninstallere Linux uden at slette systemet.

[20] http://www.linuxbog.dk/friheden/bog/friheden-intro.html
[21] http://www.dk.debian.org/security/

Apache

Apache er også ligesom de andre programmer som vi har beskrevet "open source", hvilket gør det til en meget brugt web-server på Internettet. Faktisk er 66% af alle web-servere pr. den 1. september 2003 en Apache server, hvorimod Microsoft kun har en andel på 23%[22].

Grundene til at vi valgte at benytte os af Apache var hovedsageligt at den gratis. Samtidigt er den fuldt integreret til at køre under Linux, og at den understøttede SSL.

[22] http://www.securityspace.com/s_survey/data/200309/index.html

Systemstatus ved aflevering

Vi havde planlagt fra projektets start, at vi ville prøve at lave et køreklart system, da Hanstholm Turistfart havde stor brug for et sådant system og derfor var meget interesserede i at få et færdigt produkt efter endt projektforløb. I skrivende stund har vi, som før nævnt, opsat en Linux server hos Hanstholm Turistfart, hvor systemet kører fuldt funktionelt. Personalet har dog nogle få ændringer, de gerne vil have fortaget, som nedenstående uddrag af e-mail viser.

"Hej
Vi kører på med systemet og er indtil videre meget glade for det. Vi har nogle små ting som vi gerne vil have ændret, men det kan vi forhåbentlig vende tilbage med senere.
Med venlig hilsen
Susan"

Disse småændringer vil vi imidlertid først foretage efter vi har afleveret rapporten. Blandt andet kan det nævnes at der er tale om at de ønsker at prislisten skal komme i en selvstændig popup og ikke sit eget lille vindue, og at feltet med postnummer skal udvides, så der også kan være udenlanske kunder. Udover dette havde de ønsket at der blev skrevet ved en kørsel hvilken person i administrationen, der havde oprettet denne.

Konklusion

Gennem 4 semestre på datamatiker studiet har vi tilegnet os en bred vifte af viden om de forskellige aspekter vedrørende udvikling af systemer, som vi har gjort brug af i forbindelse med denne hovedopgave.

Først udarbejdede vi en projektplan, for at fastlægge overordnede retningslinierne for selve projektet. Herunder bestemte vi os for at bruge den traditionelle projektmodel kombineret med faselinier fra den trinvise, idet vi gerne ville opnå størst mulig kontakt med Hanstholm Turistfart. Dette opnås ved, at vi ved hver trinafslutning holder møde med dem. Derudover udarbejdede vi en række analyser (TURBO, risiko, interessent), som hjælp under udviklingsprocessen.

Vi har valgt at implementere systemet som web-baseret ved hjælp af PHP. Dette gør at systemet umiddelbart nemt kan udvides med flere klienter, uden yderligere installation på hverken server eller klient. Samtidigt valgte vi at systemet skal køre på en server, som er fysisk placeret hos Hanstholm Turistfart, hovedsageligt grundet omkostninger og tilgængelighed.

For at undgå et uhensigtsmæssigt design af databasen, har vi valgt at tage udgangspunkt i kravspecifikation og tegne et ER-diagram. Diagrammet er så blevet mappet over til en relationel model, som herefter er blevet normaliseret. Hermed har vi sikret optimalt databasedesign.

I projektet er der blevet defineret en mængde aftaler. Før og fremmest den vigtigste formelle aftale med kunden, nemlig kravspecifikationen. Idet projektorganisationen befinder sig i et professionelt bureaukrati, er det ikke nødvendigt at lave aftaler om ethvert aspekt af projektet, deltagerne imellem.

Brugervenlighed er en vigtig del af vores projekt. Med udgangspunkt i Rolf Molichs brugervenlighedsprincipper, har vi i samarbejde med Hanstholm Turistfart udarbejdet en brugergrænseflade. Effektiviteten og brugernes tilfredshed med systemet har været i højsædet, da disse faktorer ellers kan bevirke at et system forkastes af brugerne.

For at sikre at systemet afspejler brugernes ønsker til funktionaliteten og eliminere katastrofer i systemet, har vi gennemført en række brugersessioner. Disse blev gennemført som tænke-højt afprøvninger, hvor resultatet af disse blev brugt til at videreudvikle prototyper.

Idet systemet skal være tilgængelig på Internettet, har vi valgt at lægge stor vægt på sikkerheden. Derfor har vi valg at beskytte siden med login, samtidig med, at vi har krypteret alt kommunikation mellem klient og server.

Vi har valgt at hovedsageligt at benytte os af PHP og MySQL. Samtidigt har vi valgt Linux som platform og Apache som web-server. Dette valg er foretaget udfra en vurdering af kvalitet og hastighed kontra pris.

Tom Oddershede	Kim Dyrholm	Ivan Larsen

Litteraturliste

Christensen, Poul Erik m.fl.
Organisation 2. udgave, Trojka, 2001
ISBN: 87-90701-42-9

Larman, Craig
Applying UML and patterns 2.udgave, Prentice Hall, 2002
ISBN: 0-13-092569-1

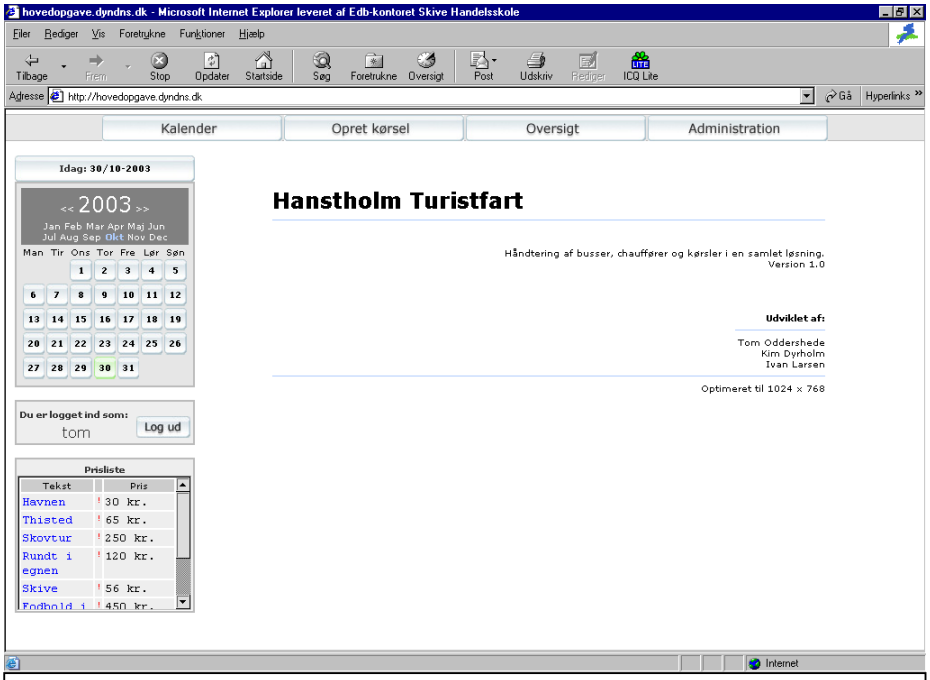
Molich, Rolf
Brugervenlige edb-systemer 2. udgave, Teknisk Forlag, 1996
ISBN: 87-571-1647-4

Munk-Madsen, Andreas
Strategisk Projektledelse, Forlaget Marko, 1996
ISBN: 87-7751-115-8

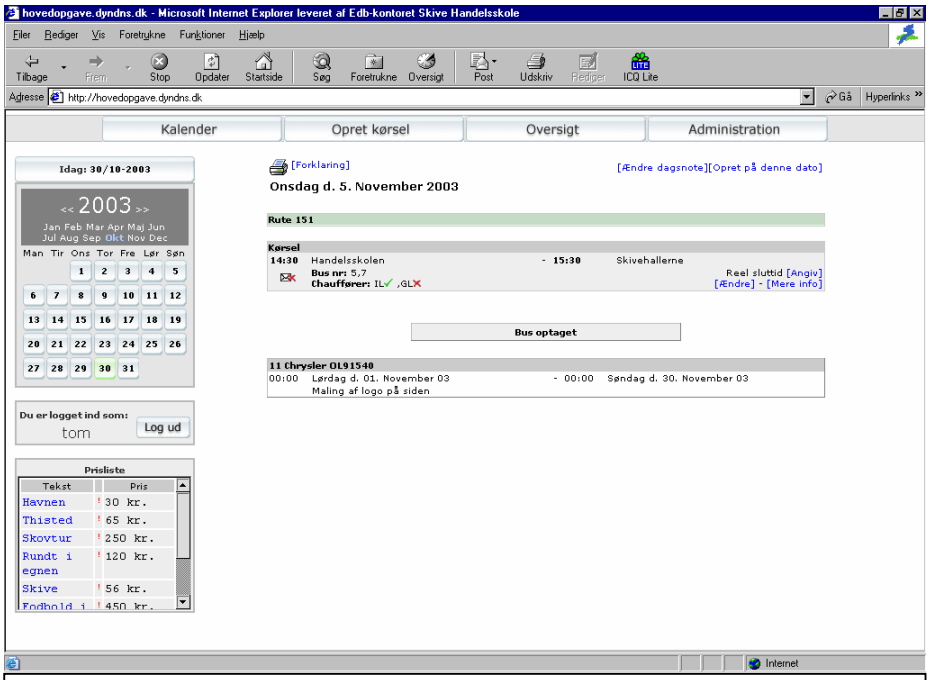
Ramez, Elmasri m.fl.
Fundamentals of Database Systems, 3. udgave, Buschlen/Mowatt Fine Art Ltd.,2000
ISBN: 0-201-54263-3

Tanenbaum, Andrew S.
Computer Networks 3. udgave, Prentice Hall 3,1996
ISBN: 0-13-394248-1

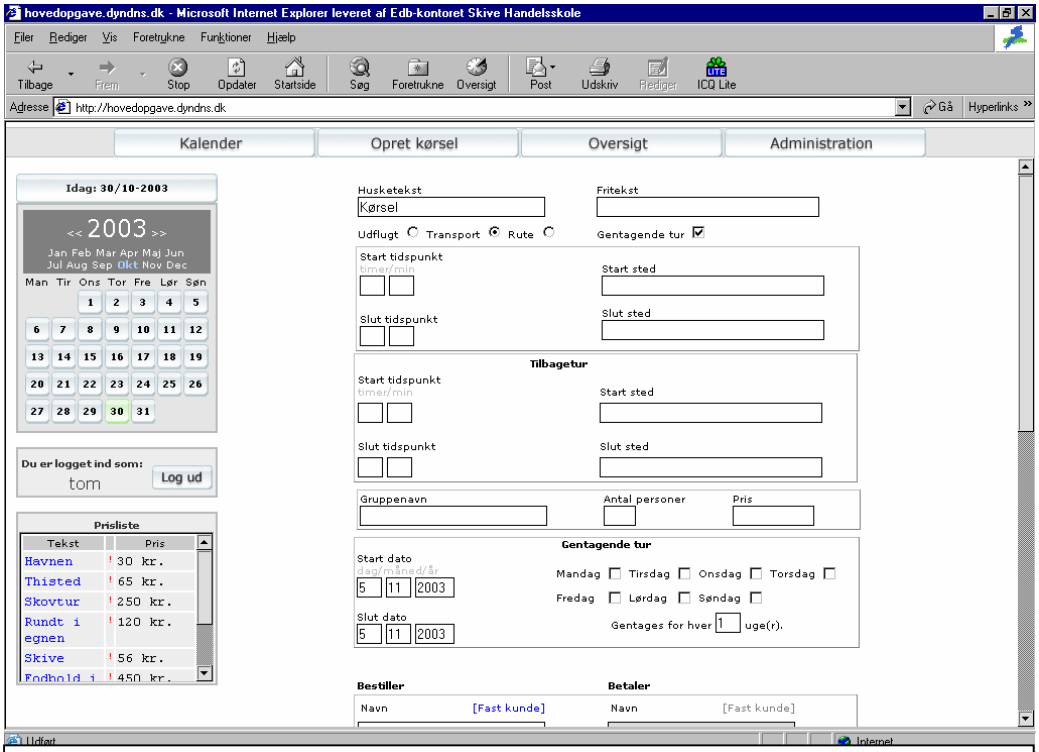
Bilag 1 – Screenshots af systemet



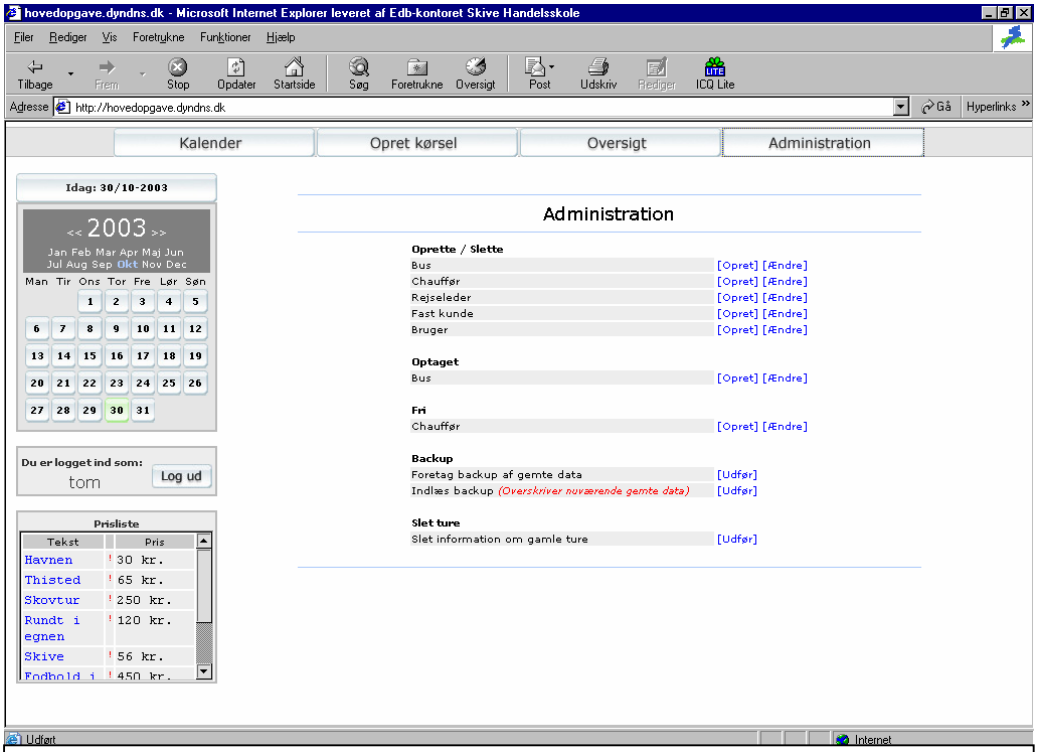
Den første side man ser, når man logger ind



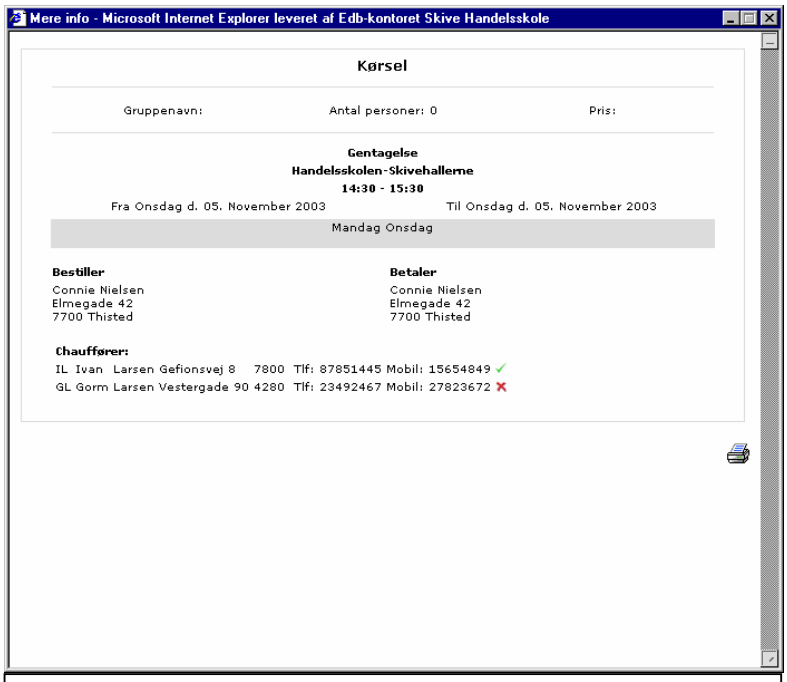
Når man trykker på kalenderen vil den pågældende dag blive vist



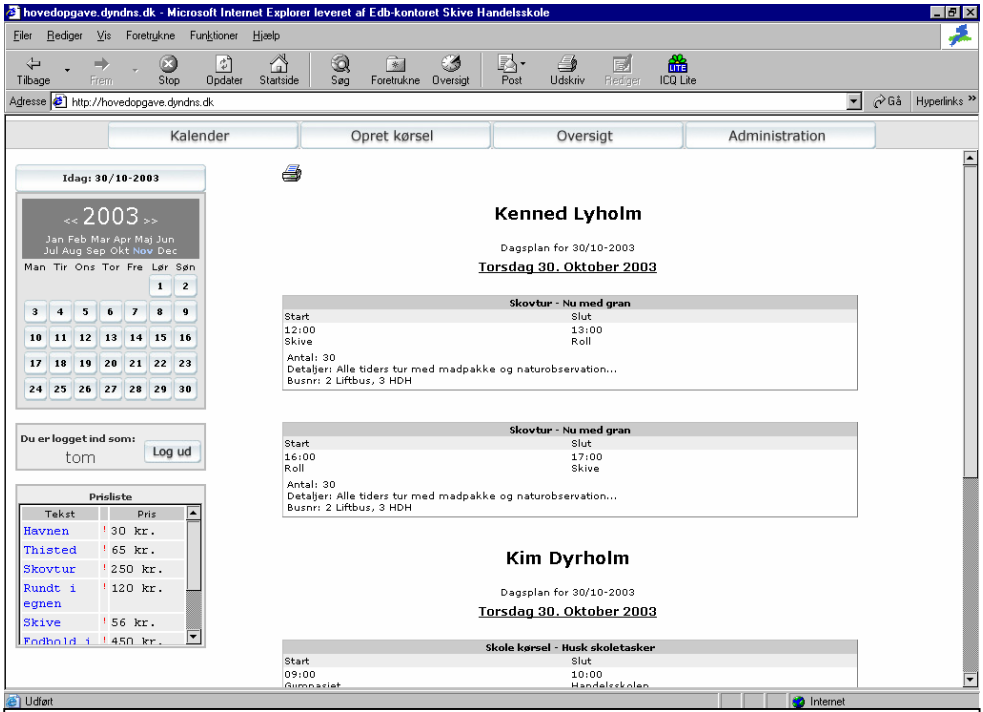
Denne skærm vises, når man vælger at oprette en kørsel



Administrationssiden

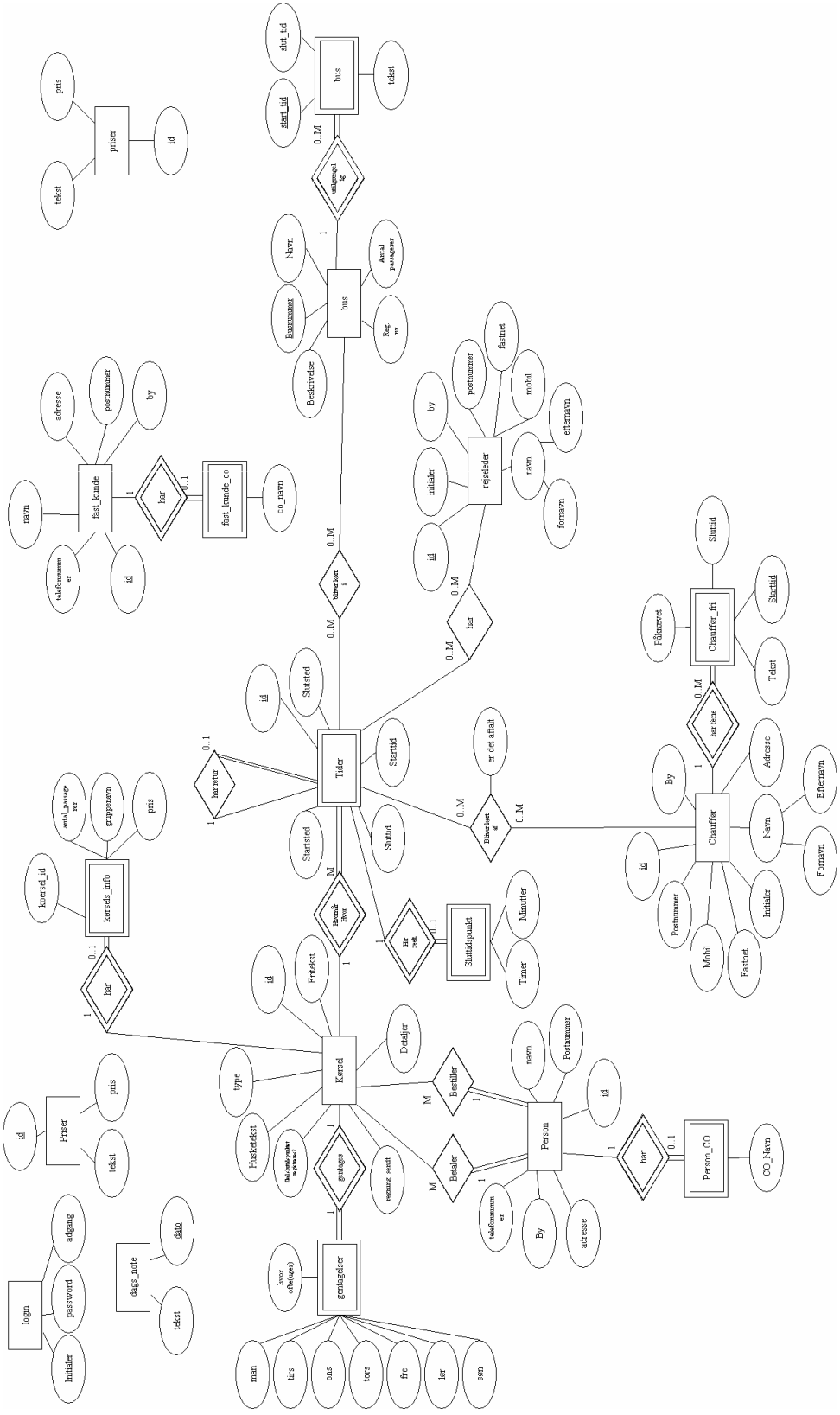


Når man trykker på mere info, vil man se alle informationer



Her vises dagsedlerne op en dag for alle chauffører

Bilag 2 – ER Diagram



Bilag 3 – Uddrag fra PHP installationsdokumentation.

```
Uddrag af PHP documentation vedrørende modul eller CGI (Install.txt)

Installing PHP for Apache as module
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  Now that version 4.1 introduces a safer sapi module, we recommend
  that you configure PHP as a module in Apache.

  To accomplish this, you have to load the php4apache.dll in your
  Apache httpd.conf.

  !! NOTE !!
  Whereever you load php4apache.dll from, php4apache.dll also
  needs the php4ts.dll also included in the PHP4 distribution.
  php4apache.dll depends on php4ts.dll which is loaded as soon as
  Apache loads php4apache.dll. If php4ts.dll can't be found, you
  usually get an error like (also see the "Problems?" section at
  the end of the file):

    Cannot load c:/php/sapi/php4apache.dll into server

  So where does php4ts.dll has to be to be properly loaded ?
  php4ts.dll is searched in the following order:

  1) in the directory where apache.exe is start from
  2) in the directory where php4apache.dll is loaded from
  3) in your %SYSTEMROOT%\System32, %SYSTEMROOT%\system and
     %SYSTEMROOT% directory.
     Note: %SYSTEMROOT%\System32 only applies to Windows NT/2000/XP)
  4) in your whole %PATH%

  Note: What is %SYSTEMROOT% ? Depending on your Windows
        installation this may be for example c:\winnt or C:\windows

  Usually you would just copy it over to %SYSTEMROOT%\System32.
  But if you want to have multiple PHP installations (for
  whatever reason) this is a bad idea. For this circumstance the
  safest thing is to let php4ts.dll reside in the same directory
  where php4apache.dll is loaded from (see point 2 above).


  After you've set up the file layout properly, you're ready to
  finally configure Apache to load the PHP4 module. Just add the
  following lines to your httpd.conf:

   LoadModule php4_module c:/php/sapi/php4apache.dll
   AddModule mod_php4.c
   AddType application/x-httpd-php .php

  Note: Especially newer versions of Apache do not need the
        AddModule directive anymore, your milage may vary.


  Where do I have to put the php.ini ?
   The php.ini files is only searched in two places:
   1) in your Apache installation directory (e.g. c:\apache\apache)
   2) in your %SYSTEMROOT% directory.


  Installing PHP for Apache as CGI binary
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
If you wish to install PHP as a CGI binary, read this first:

  http://www.cert.org/advisories/CA-1996-11.html

and then if you are really sure, insert these lines to your conf file:

 ScriptAlias /php/ "c:/php/"
 AddType application/x-httpd-php .php
 Action application/x-httpd-php "/php/php.exe"

Note, we consider installing PHP like this suicidal.

As a further precaution, we recommend you change the "/php/"
ScriptAlias to something more random, to prevent the binary being
called directly, which is a security risk.

Remember when you have finished to restart the server, for example,
 NET STOP APACHE
followed by
 NET START APACHE

To use the source code highlighting feature, add the following
line to your apache httpd.conf file:

  AddType application/x-httpd-php-source .phps

Note, this will only work when you install php as a sapi module.
If you wish to use this feature with the cgi binary, create a new
file, and use the show_source("path/to/original_file.php"); function.
```

Bilag 4 – Oprettelse af tabeller i MySQL

```
#
# Struktur dump for tabellen `bus`
#

CREATE TABLE bus (
  busnr int(11) NOT NULL default '0',
  bus_navn varchar(25) NOT NULL default '',
  antal_passagerer int(3) NOT NULL default '0',
  reg_nr varchar(7) NOT NULL default '',
  beskrivelse text NOT NULL,
  PRIMARY KEY  (busnr)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `bus_utilgaengelig`
#

CREATE TABLE bus_utilgaengelig (
  busnr int(11) NOT NULL default '0',
  start_tid int(10) NOT NULL default '0',
  slut_tid int(10) NOT NULL default '0',
  tekst varchar(50) NOT NULL default '',
  PRIMARY KEY  (busnr,start_tid,slut_tid)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `chauffoer`
#

CREATE TABLE chauffoer (
  id_chauffoer int(11) NOT NULL auto_increment,
  initialer varchar(5) NOT NULL default '',
  fornavn varchar(50) NOT NULL default '',
  efternavn varchar(50) NOT NULL default '',
  adresse varchar(50) NOT NULL default '',
  postnummer int(4) NOT NULL default '0',
  telefonnummer varchar(8) NOT NULL default '',
  mobiltelefon varchar(8) NOT NULL default '',
  PRIMARY KEY  (id_chauffoer)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `chauffoer_fri`
#

CREATE TABLE chauffoer_fri (
  chauffoer_id int(11) NOT NULL default '0',
  start_tid int(10) NOT NULL default '0',
  slut_tid int(10) NOT NULL default '0',
  tekst varchar(50) NOT NULL default '',
  paakraevet enum('j','n') NOT NULL default 'n',
  PRIMARY KEY  (chauffoer_id,start_tid,slut_tid)
) TYPE=MyISAM;

# --------------------------------------------------------
```

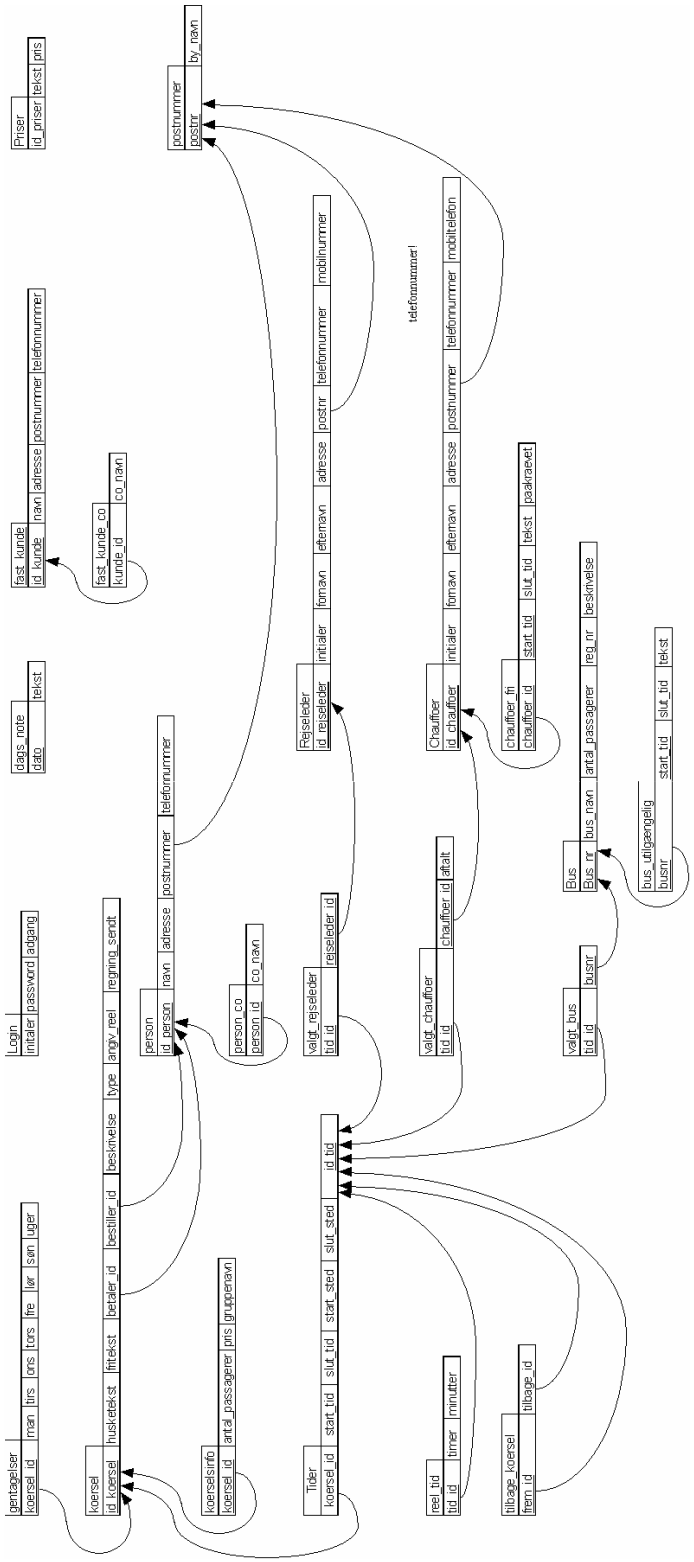
```
#
# Struktur dump for tabellen `dags_note`
#

CREATE TABLE dags_note (
  dato varchar(10) NOT NULL default '',
  tekst text NOT NULL,
  PRIMARY KEY  (dato)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `fast_kunde`
#

CREATE TABLE fast_kunde (
  id_kunde int(11) NOT NULL auto_increment,
  navn varchar(50) NOT NULL default '',
  adresse varchar(50) NOT NULL default '',
  postnummer int(4) NOT NULL default '0',
  telefonnummer varchar(12) NOT NULL default '',
  PRIMARY KEY  (id_kunde)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `fast_kunde_co`
#

CREATE TABLE fast_kunde_co (
  kunde_id int(11) NOT NULL default '0',
  co_navn varchar(50) NOT NULL default '',
  PRIMARY KEY  (kunde_id)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `gentagelser`
#

CREATE TABLE gentagelser (
  koersel_id int(11) NOT NULL default '0',
  man enum('j','n') NOT NULL default 'n',
  tirs enum('j','n') NOT NULL default 'n',
  ons enum('j','n') NOT NULL default 'n',
  tors enum('j','n') NOT NULL default 'n',
  fre enum('j','n') NOT NULL default 'n',
  loer enum('j','n') NOT NULL default 'n',
  soen enum('j','n') NOT NULL default 'n',
  uger int(11) NOT NULL default '0',
  PRIMARY KEY  (koersel_id)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `koersel`
#

CREATE TABLE koersel (
  id_koersel int(11) NOT NULL auto_increment,
  husketekst varchar(50) NOT NULL default '',
  fritekst varchar(50) NOT NULL default '',
  bestiller_id int(11) NOT NULL default '0',
```

```
  betaler_id int(11) NOT NULL default '0',
  beskrivelse text NOT NULL,
  type enum('t','u','r') NOT NULL default 't',
  angiv_reel enum('j','n') NOT NULL default 'n',
  regning_sendt varchar(5) default NULL,
  PRIMARY KEY  (id_koersel)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `koerselsinfo`
#

CREATE TABLE koerselsinfo (
  koersel_id int(11) NOT NULL default '0',
  antal_passagerer int(3) NOT NULL default '0',
  pris varchar(15) NOT NULL default '0',
  gruppe_navn varchar(50) NOT NULL default '',
  PRIMARY KEY  (koersel_id)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `login`
#

CREATE TABLE login (
  initialer varchar(5) NOT NULL default '',
  password varchar(16) NOT NULL default '',
  adgang int(1) NOT NULL default '1',
  PRIMARY KEY  (initialer)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `person`
#

CREATE TABLE person (
  id_person int(11) NOT NULL auto_increment,
  navn varchar(50) NOT NULL default '',
  adresse varchar(50) NOT NULL default '',
  postnummer int(4) NOT NULL default '0',
  telefonnummer varchar(12) NOT NULL default '',
  PRIMARY KEY  (id_person)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `person_co`
#

CREATE TABLE person_co (
  person_id int(11) NOT NULL default '0',
  co_navn varchar(50) NOT NULL default '',
  PRIMARY KEY  (person_id)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `postnummer`
#
```

```
CREATE TABLE postnummer (
  postnr int(4) NOT NULL default '0',
  by_navn varchar(255) default NULL,
  PRIMARY KEY  (postnr)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `priser`
#

CREATE TABLE priser (
  id_priser int(11) NOT NULL auto_increment,
  tekst varchar(30) NOT NULL default '',
  pris varchar(30) NOT NULL default '',
  PRIMARY KEY  (id_priser)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `reel_tid`
#

CREATE TABLE reel_tid (
  tid_id int(11) NOT NULL default '0',
  reel_timer char(2) NOT NULL default '0',
  reel_minutter char(2) NOT NULL default '0',
  PRIMARY KEY  (tid_id)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `rejseleder`
#

CREATE TABLE rejseleder (
  id_rejseleder int(11) NOT NULL auto_increment,
  initialer varchar(5) NOT NULL default '',
  fornavn varchar(50) NOT NULL default '',
  efternavn varchar(50) NOT NULL default '',
  adresse varchar(50) NOT NULL default '',
  postnummer int(4) NOT NULL default '0',
  telefonnummer varchar(8) NOT NULL default '',
  mobiltelefon varchar(8) NOT NULL default '',
  PRIMARY KEY  (id_rejseleder)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `tider`
#

CREATE TABLE tider (
  id_tid int(11) NOT NULL auto_increment,
  koersel_id int(11) NOT NULL default '0',
  start_tid int(10) NOT NULL default '0',
  slut_tid int(10) NOT NULL default '0',
  start_sted varchar(50) NOT NULL default '',
  slut_sted varchar(50) NOT NULL default '',
  aflyst enum('j','n') NOT NULL default 'n',
  PRIMARY KEY  (id_tid),
  KEY tider (start_tid,slut_tid)
```

```
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `tilbage_koersel`
#

CREATE TABLE tilbage_koersel (
  frem_id int(11) NOT NULL default '0',
  tilbage_id int(11) NOT NULL default '0',
  PRIMARY KEY  (frem_id,tilbage_id)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `valgt_bus`
#

CREATE TABLE valgt_bus (
  tid_id int(11) NOT NULL default '0',
  busnr int(11) NOT NULL default '0',
  PRIMARY KEY  (tid_id,busnr)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `valgt_chauffoer`
#

CREATE TABLE valgt_chauffoer (
  tid_id int(11) NOT NULL default '0',
  chauffoer_id int(11) NOT NULL default '0',
  aftalt enum('j','n') NOT NULL default 'n',
  PRIMARY KEY  (tid_id,chauffoer_id)
) TYPE=MyISAM;

# --------------------------------------------------------

#
# Struktur dump for tabellen `valgt_rejseleder`
#

CREATE TABLE valgt_rejseleder (
  tid_id int(11) NOT NULL default '0',
  rejseleder_id int(11) NOT NULL default '0',
  aftalt enum('j','n') NOT NULL default 'n',
  PRIMARY KEY  (tid_id,rejseleder_id)
) TYPE=MyISAM;
```

Bilag 5 – Oversigt over tabeller



Bilag 6 – Liste over filer

admin.js
JavaScript fil der tjekker for fejl i indtastningen på diverse menupunkter under menupunktet "Administration".

admin.php
Selve opbygningen af former og database tilgange for menupunktet "Administration", dog ikke udfør backup og indlæs backup.

aendre_gentagende.php
Ændringer af stamoplysninger til gentagende ture.

aftal_med_chauffoer.php
Ændrer i databasen så en chaufför står til at han er underrettet om at han skal køre den pågældende kørsel. Denne fil indeholder kun database tilgang.

angiv_reel.php
Ændrer i databasen, så den reelle tilbagemeldingstid bliver sat for en kørsel.

backup.php
Laver et "dump" af MySQL databasen til en fil lokalt på brugerens computer.

dagsnote.php
Oprettelse af en dagsnote.

forklaring.htm
Hjælp funktion med forklaring omkring farver og ikoner.

gentagende_gem.php
Gemmer stamdata for gentagende ture.

index.php
Indeholder 3 frames med henholdsvis menu, kalender og visning.

indlaes_backup.php
Indlæser database backup fra angivet fil.

kalender.js
JavaScript fil der laver popups og diverse tjek på visning af kalender (viskalender.php).

kalender.php
Navigering imellem forskellige datoer.

koersel_form.js
JavaScript fil der laver popups og fejltjekker ved oprettelse af en kørsel og ændring af kørsel.

login.css
Stylesheet til login.

login.inc.php
Inkluderer tjek om der er en bruger der er logget ind.

login2.inc.php
Inkluderer login funktionen, når bruger er logget ind.

logout.php
Logger allerede logget ind person ud.

menu.php
Top menuen med generel navigering.

```
mere_info.php
Udskriver alle data om valgt kørsel.

mysql.inc.php
Inkluderes på alle filer med databasetilgang. Den opretter forbindelse til
databasen.

opret.php
Oprettelse af kørsel.

opret_aendre.php
Ændring af allerede oprettet kørsel.

opret_gem.php
Gemmer alle data fra opret.php og opret_aendre.php i databasen.

overlap.php
Tjekker for overlappende chauffører og busser, ved kørsler og fri/ude af drift
på overlappende tidspunkt.

oversigt.php
Oversigt over hvilke busser, chauffører og rejseledere der er på kørsel på
nuværende tidspunkt.

prisliste.php
Oprettelse og slet på prislisten, prislisten er en iFrame.

regning.php
Ændrer om regning er sendt eller ikke sendt. Registrerer også hvilken bruger
ændringen er foretaget af.

samme_betaler.js
JavaScript fil der laver popup med vaelg_kunde.php.

start_side.php
Siden der startes på ved første login.

style.css
Generel brugt stylesheet fil.

style_input.css
Generel brugt stylesheet fil.

udskrivning.php
Sørger for udskrivning af dagsedler, under menupunktet "Oversigt".

vaelg.js
Større JavaScript der håndterer at skjule og markere indtastningsfelter, ved
valgt af type tur.

vaelg_bus.php
Popup hvor der kan vælges busser til kørsel.

vaelg_chauffoer.php
Popup hvor der kan vælges chauffører til kørsel.

vaelg_kunde.php
Popup hvor der kan vælges fast kunder til kørsel.

vaelg_rejseleder.php
Popup hvor der kan vælges rejseledere til kørsel.

valid.js
Generel JavaScript fil til tjek af bruger indtastninger.
```

```
viskalender.php
Viser kørsler og andet info på valgt dato.
```