Nettside, Webshop og Beregningsmodell

Hovedprosjekt våren 2009

høgskolen i oslo
avdeling for ingeniørutdanning

Studieprogram: Anvendt Datateknologi
Postadresse: Postboks 4 St. Olavs plass, 0130 Oslo
Besøksadresse: Holbergs plass, Oslo

Telefon: 22 45 32 00
Telefaks: 22 45 32 05

PROSJEKT NR.
22

TILGJENGELIGHET
Åpen

HOVEDPROSJEKT

HOVEDPROSJEKTETS TITTEL	DATO
Nettside, Webshop og beregningsmodell	29.05.09
	ANTALL SIDER / BILAG
PROSJEKTDELTAKERE Peder Sundbø, s141795 Magnus Eriksen, s141765 Øyvind Schjelderupsen, s141758	INTERN VEILEDER Torunn Gjester

OPPDRAGSGIVER	KONTAKTPERSON
Stansefabrikken Products AS – http://www.stansefabrikken.com/index.php?cid=4148	Trond Eriksen - Salgssjef

SAMMENDRAG

Dokumentet du nå holder i hånden er bacheloroppgaven til tre studenter på studiet Anvendt Datateknologi ved ingeniørutdanningen på Høgskolen i Oslo. Oppgaven er skrevet i tidsrommet januar – mai 2009, og er utarbeidet for og i samarbeid med Stansefabrikken Products AS. Vår veileder i denne prosessen har vært Torunn Gjester, som er foreleser ved HiO.

Systemet består i hovedsak av en beregningsmodell skrevet i flash (Actionscript 3.0) og en web-shop (skrevet i HTML og PHP) med tilknytning til en MySQL database.

Arbeidet har vært tidkrevende og til tider svært utfordrende, spesielt med tanke på at ingen av oss kunne programmere i flash. Mye av tiden har derfor gått med til å tilegne seg denne teknologien.

Det ferdige produktet er overlatt til Stansefabrikken, og de står fritt til å bruke systemet som det er eller videreutvikle det til eget bruk.

God fornøyelse.

3 STIKKORD
Flashapplikasjon, AS 3.0
Web-shop, PHP
Database, MySQL

Forord

Denne rapporten er resultatet av et hovedprosjekt som er gjennomført ved Høgskolen i Oslo, avdeling for ingeniørutdanning. Rapporten tar for seg utviklingen av en webshop og en beregningsmodell for Stansefabrikken AS. I tillegg til å fremstille hele utviklingsprosessen gir også denne rapporten innsyn i teknologier som Adobe Flash og webprogrammering i PHP, samt HTML og CSS.

Innholdsfortegnelse

Presentasjon

Hovedprosjektet gjennomføres ved Høgskolen i Oslo, avdelingen for ingeniørutdanning. Målet med oppgaven er å utvikle en online webshop for bestilling av postkasser. Dette vil lette arbeidsmengden til de ansatte i postkasseavdelingen betraktelig, siden de da slipper å snakke fysisk med hver enkelt kunde. I tillegg skal vi lage en beregningsmodell som gjør det mulig for kunder og arkitekter å beregne mål for plassering av postkasser i eksisterende bygg og nybygg. Disse systemene må stille store krav til brukervennlighet da brukerne ofte kan være uerfarne databrukere.

Mål

Resultatmål

Målet med vår prosjektoppgave er å lage et nettbasert system som gjør det mulig for kunder av Stansefabrikken Products AS å bestille postkasser over internett, med en webshop. Siden skal også ha en beregningsmodell for plassering av postkasser i en oppgang. Denne beregningsmodellen skal kunne gi kunden mulighet til lage et oversiktlig bilde av hvordan målene til kassene er og hva de har plass til i sin egen oppgang.

Effektmål

Stansefabrikken Products ble i 2007 skilt ut som et eget forretningsområde som fokuserer på salg av produkter og løsninger innen et bredt spekter av elektroskap og postkasser. Selskapet er lokalisert på Fornebu og Fredrikstad er et datterselskap av Stafa Industrier AS. Virksomheten er salg av postkassesystemer til boligblokker i Norge. Kassene er produsert i Litauen og Norge. Disse har solgt jevnt over mange år på bekjentskaper og navn. I de senere år er det blitt mer og mer viktig å være på nett med brosjyrer og andre hjelpemidler som bedriftens kunder kan benytte. Stansefabrikken AS har mange kunder med spesielle behov slik som arkitekter, byggefirmaer og forhandlere. De mangler hjelpemidler på nettet for oppsett av kasser i forhold til størrelsen på kassene og de lover som til en hver tid gjelder. Bedriften ønsker å utvikle enkle hjelpemidler for kundene sine i form av tabeller eller regneark som letter arbeidet for begge parter. Samtidig vil det bli behov for å utvikle en netthandel som også vil lette arbeidet. Systemet vil kunne lette arbeidsmengden på selgere av postkasser fordi kunder kan henvises til beregningsmodellen når det gjelder plassering av postkasser. Utligere har de til dags dato ingen form for bestilling over internett, alt salg foregår over telefon og e-post. Dermed vil det å ha en nettbutikk forbedre stansefabrikkens nettsider i forhold til salg betraktelig. Effektmålet er å øke salget ved å gjøre produktene tilgjengelig for kundene via internett.

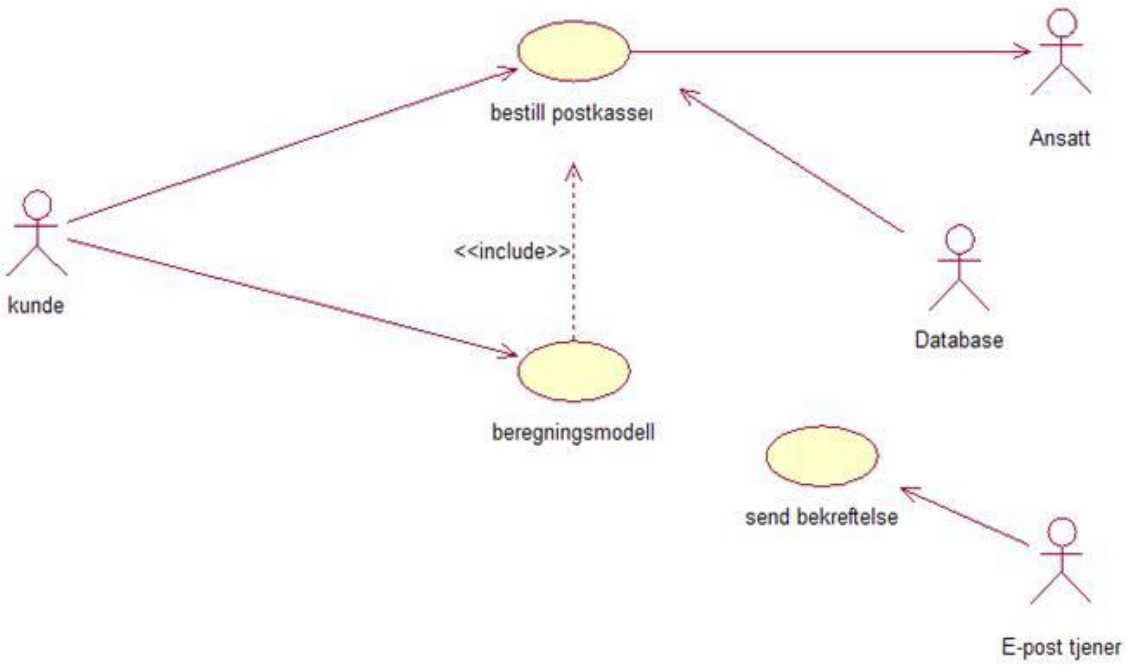
Kravspesifikasjonen

Kravspesifikasjon er et formelt dokument, som brukes som en juridisk bindende kontrakt mellom kunde og utvikler om hva som skal gjøre i prosjektet. Det er viktig og utarbeide en god kravspesifikasjon slik at kunden og utvikler har satt seg klare mål for prosjektet.

Denne kravspesifikasjonen ble utarbeidet av gruppen sammen med oppdragsgiver, hvor oppdragsgiver måtte godkjenne kravspesifikasjonen før videre arbeid ble fastslått. Kravspesifikasjonen er beregnet for de medvirkende i prosjektet, altså oppdragsgiver, gruppemedlemmene og veileder. Den er også beregnet for en sensor som skal evaluere og bedømme prosjektresultatet. Systemets funksjonalitet, spesifikasjoner og rammebetingeleser er beskrevet i dette dokumentet, som er en instruks for hvordan systemets skal fungere.

Systembeskrivelse

Dette bildet viser et overordnet bilde av systemet.



Systemet skal inneholde:

- Mulighet til å bestille postkasser på nett. Det skal være mulig å velge antall og type postkasser.
- En webshop hvor kunder kan bestille ønskelige postkasser uten å benytte beregningsmodellen.
- En beregningsmodell i flash som regner ut hvor mye plass som trengs til postkasser i henhold til standard for plassering av postkasser og mål oppgitt av kunde/arkitekt.
- Mulighet for kunde å sette sammen ønskelige postkasser ved hjelp av ”drag and drop” i flash.
- Database over alle typer postkasser med mål i millimeter.
- Mulighet for å skrive ut resultatet fra beregningsmodellen.
- Bekreftelse til kunde via e-post.
- Ansatte varsles om bestillinger på e-post.

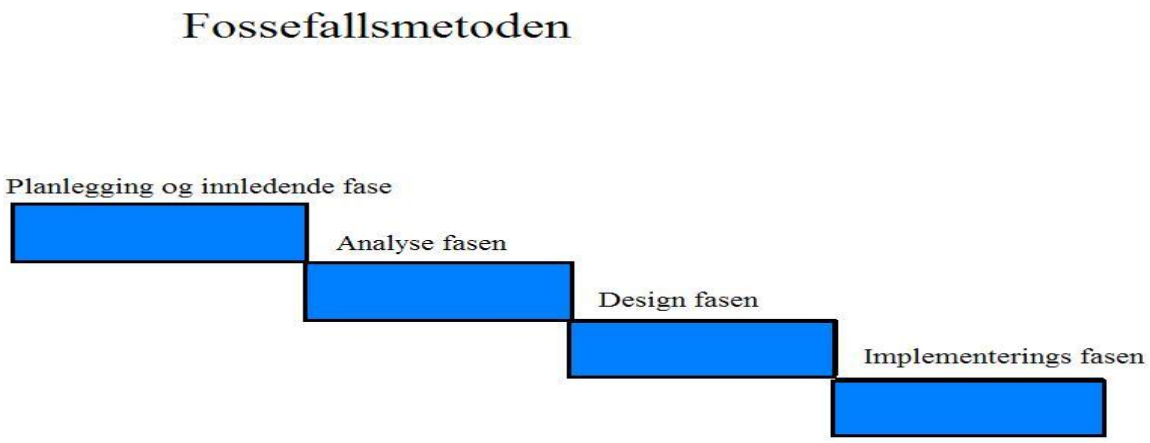
Rammekrav i systemet

- Hoveddelene skal programmeres i PHP (objektorientert).
- Beregningsmodellen skal utarbeides i flash.
- Systemet skal kunne utvides ved at det kan brukes direkte på nettsidene til stansefabrikken. It avdelingen til stansefabrikken skal også kunne gjøre ytterligere forbedringer eller endringer i senere tid

Valg av metode

Metode

Fossefallsmetoden er en metode for system utvikling. Fossefallsmetoden går ut på at du deler opp hele prosjektet i faser. Man utfører disse fasene en etter en i rekkefølgen man har satt opp de forskjellige fasene. Når man bruker denne formen for systemutvikling må en fase gjøres ferdig før man begynner på en ny en. Vårt fossefallsdiagram:



Ved bruk av en slik metode jobber man seg ned hver fase til man kommer til slutten. Vi har valgt å bruke fossefallsmetoden fordi denne er god å bruke på mindre prosjekter hvor man har klare krav og godt dokumentert planlegging. Hvis man jobber på større prosjekter der mange er involvert og kravene endres ofte er ikke fossefallsmetoden den beste utviklingsmetoden.

Vi har delt opp vår utvikling i 5 faser.

- Planlegging og innledende fase:
  I denne fasen setter vi mest vekt på å planlegge systemet som skal utvikles. Vi arbeider med å utvikle en kravspesifikasjon for systemet hvor systemet er mest mulig fastslått på forhånd. I denne perioden valgte vi å lage en papirprototype for å fastslå hovedtrekk ved design av siden.

- Analyse fasen:
  I denne fasen går vi litt dypere i systemet og prøver og analysere på et litt lavere nivå enn i planleggingsfasen. Her benytter vi Er-diagram for planlegging av database. Vi ser på strukturen i systemet og lignende.

- Design fasen:
  I denne delen av utviklingen av systemet lager vi design prototype og planlegger design av den endelige side mye dypere , jobber med kompetanse utvikling av nytt materiale for utvikling av det som skal lages.

- Implementeringsfasen:
  Er den delen av metoden der vi faktisk produserer systemet. I denne fasen lager vi systemet ut i fra planlegging fra tidligere.

- Testing:
  Siste fase er da vi tester det endelige systemet på brukere av systemet.

PROSJEKTPLANLEGGING

Fremdriftsplan

Fremdriftsplan er en oversiktlig plan over fremdriften i prosjektet. Fremdriftsplanen brukes for å sette tidsrammer for hele prosjektet. Dette er et overordnet dokument som hjelper oss og se når vesentlige deler av planlegging, analyse, utvikling og implementering skal være ferdig. Vi valgte å bruke dette styringsdokumentet for å få et overordnet blikk på hva som skulle gjøres og sette frister for disse oppgavene.
Fremdriftsplan (vedlegg 1).

Aktivitetskart

Vi valgte å lage en litt mer detaljert plan for arbeidet vårt under dette prosjektet ut over den allerede utviklede fremdriftsplanen. Dette dokumentet kalles aktivitetskart, der vi puttet inn alle punkter som skulle gjøres under veis. Vi fylte ut i denne planen når ting skulle være ferdig, hvem som hadde ansvaret for punktet, hvorfor punktet skulle utføres og hvor lang tid det egentlig tok og fullføre det. Dette dokumentet kunne da være en ganske enkel 2 ukers plan med oversiktlig blikk på hva som måtte gjøres. (Vedlegg 2)

Risikoanalyse

Risiko analyse brukes til å kartlegge problemer som kan oppstå under et prosjekt eller en oppgave. Man bruker det til å kartlegge hvor stor fare det er for at et problem kan oppstå og konsekvensene av et slikt problem. Dette dokumentet har vi valgt å lage for å ha et oppslagsverk ved eventuelle problemer underveis. (Vedlegg 3)

Prototyping

Papirprototype

Når man lager en papirprototype setter man seg ned med kravene til systemet og lager enkle strektegninger av systemet. Det som er greit og ha med i en papirprototype er litt funksjonalitet slik at man kan se litt hvordan systemet skal reagere på brukernes valg. På den måten kan man få en pekepinn på hvordan deler av et system er og teste den ut, eventuelt bruke det som planlegging for videre arbeid. Det kan også gjøre deg klar over eventuelle problemer man kan møte og hjelper og få start på design av brukergrensesnitt.

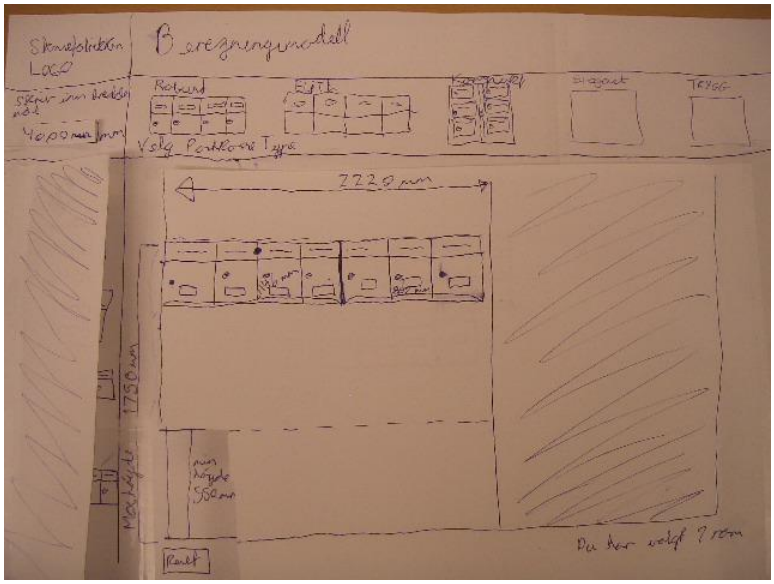
Vi valgte å lage en papirprototype av systemet for og planlegge enkel design og hvordan systemet skal se ut i grove trekk og få en enkel grov skisse over systemet som vi kan bruke til og planlegge systemet sammen med våre arbeidsgivere. Dette vil da sammen med kravspesifikasjonen vår gi oss et godt utgangspunkt til å lage et bra system. Tankene og layout knyttet til papirprototypen er laget i samarbeid med arbeidsgiverne våre.

Siden systemet vårt er delt i 2 deler (beregningsmodell og web-shop) har vi valgt å lage disse hver for seg. Det vil komme en sammenslåing av disse i implementeringsfasen som da vil si at de vil henge sammen ved at de linkes til fra samme side.

Beregningsmodell

Vi startet med å tegne en enkel layout til en webside lik den Stansefabrikken Products AS har i dag, det neste var og lage selve beregningsmodellen, først fant vi ut av hvordan layout den skulle ha. Så valgte vi plassering av de mulighetene den skulle ha. Da vi var ved dette punktet måtte vi ha litt tanker om hvilket utviklingsverktøy som kunne klare å lage den funksjonaliteten vi ønsket. Vi valgte å se litt på nettet etter teknologi som gir støtte for det vi ønsket og lage. Vi kom da frem til at flash (beskrevet om i avsnitt ”hva er flash?”) var et verktøy som ville egne seg til en slik beregningsmodell. Det neste vi gjorde var da å lage en enkel funksjonalitet i papirprototypen vår som viste hvordan den skal reagere i et gitt scenario.

Her er et bilde av papirprototypen, denne viser i grove trekk hvordan sluttproduktet av beregningsmodellen virker og ser ut i grove trekk:



Tanker og Planlegging av beregningsmodellen i papirprototypen

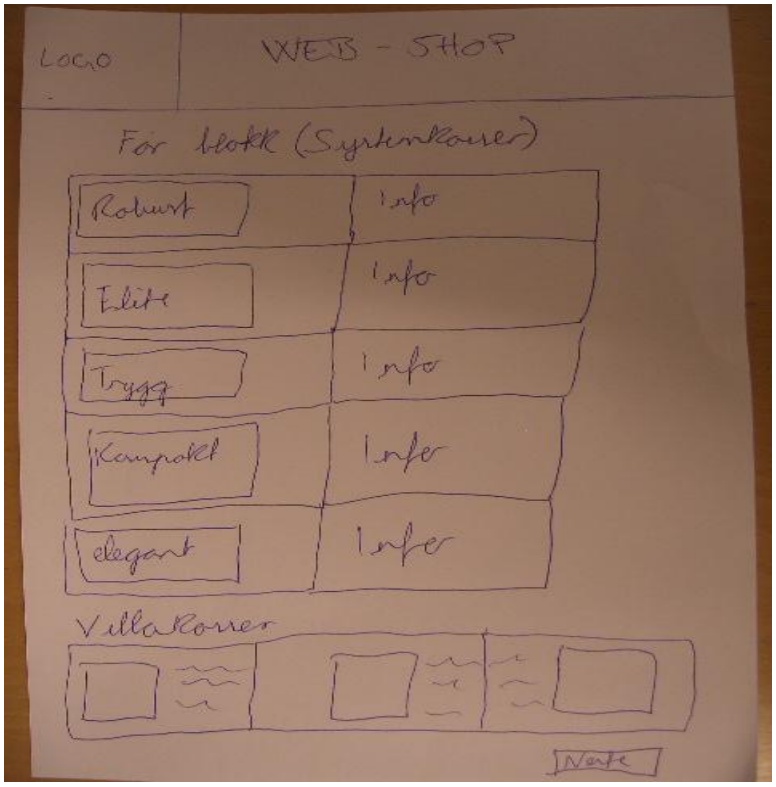
Vi valgte å la produktene til stansefabrikken products AS (postkasser) stå på rekke øverst på nettsiden, siden dette er det stede kunden først ser å det er det mest relevante som må komme frem, når kunden da velger en type postkasse etter ønske, vil han få frem et nytt vindu med alle størrelser av den utvalgte postkassetypen. Her valgte vi å legge til en funksjon som ber kunden skrive inn breddemål (ledig plass) i oppgangen. Når kunden fyller ut dette vil en tegning med mål komme opp til høyere for postkassene. Disse vil ha høydemål fastslått i følge den standard som postverket krever. Det systemet skal gi mulighet for nå er å dra postkasser over til det avgrensede området slik at du får en detaljert plantegning for kassene med mål. Hvis kunden velger å dra en kasse det ikke er plass til i bredde vil systemet forby dette.

WEB-shop

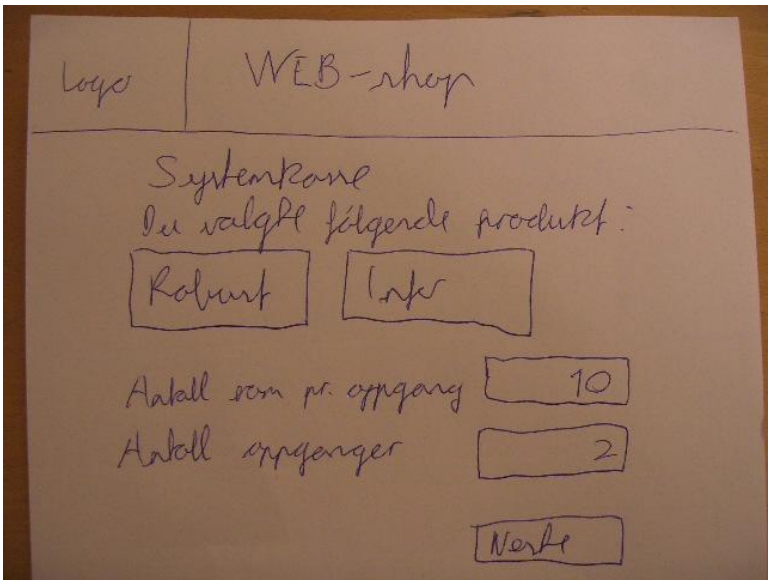
Ved utviklingen av papirprototypen for web-shop delen startet vi med enkel layout for siden som i beregningsmodellen, Siden dette må være likt slik at den eventuelle brukeren av systemet føler at han eller hun er enda på samme side.

Det neste var og få med de tingene vi har beskrevet i kravspesifikasjonen. Først vil siden gi brukeren bilder av alle produktene som vist på bilde ved siden av:

Her valgte vi å sette produktene inn i en tabell siden dette syntes å være den mest oversiktlige måten og plassere de på. Info om postkasser vil da få en god grafisk plassering i forhold til bilde av produktet. Vi valgte å dele opp postkassene i systemkasser og villakasser. Forskjellen på disse to er at systemkasser er kasser ofte relatert til hus med flere postkasser i en oppgang, Villakasser er mer for et hus eller postkassestativ og trenger da mindre planlegging antall oppganger.



Når kunden har valgt en postkasse type vil han eller hun bli sendt til dette skjermbildet:

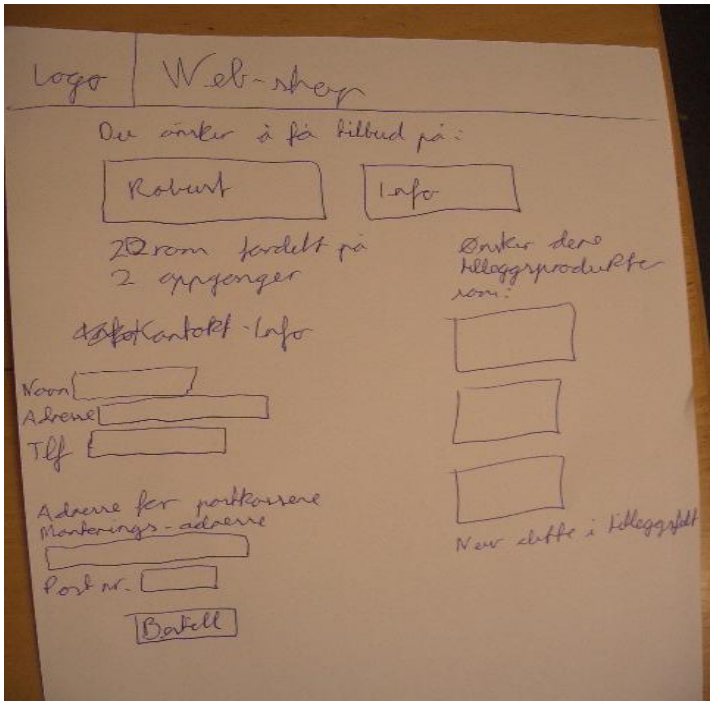


Her kan kunden lese litt om produktet de har valgt, de får nå mulighet til å velge hvor mange kasser de skal ha av produktet.
Vi har valgt å ha flere sider for å gjøre det mer oversiktlig for kunden underveis. Kunder kan fort ha en tendens til og droppe siden hvis personen møter ”a wall of text”, med dette mener vi at hvis siden blir for overflyt til at kunden mister helt fokus og dette vil medføre at han eller hun kanskje dropper å gjennomføre kjøpet.

Etter at kunden har valgt antall postkasser vil personen bli sendt til skjermbildet under.

Her blir kunden nødt til å legge inn informasjon om seg selv, slik at kundebehandler kan kontakte kunden om bestillingen. Vi har valgt å plassere dette på en oversiktlig måte til venstre på siden hvor det skal komme klart frem hva som behøves og fylles ut. Vi vil også forklare at informasjonen ikke vil bli brukt til annet enn å knytte bestillingen til kunden.

Når kunden har gjort dette vil systemet sende bestilling og en bekreftelse til kunden.



Etter utvikling av papirprototype:

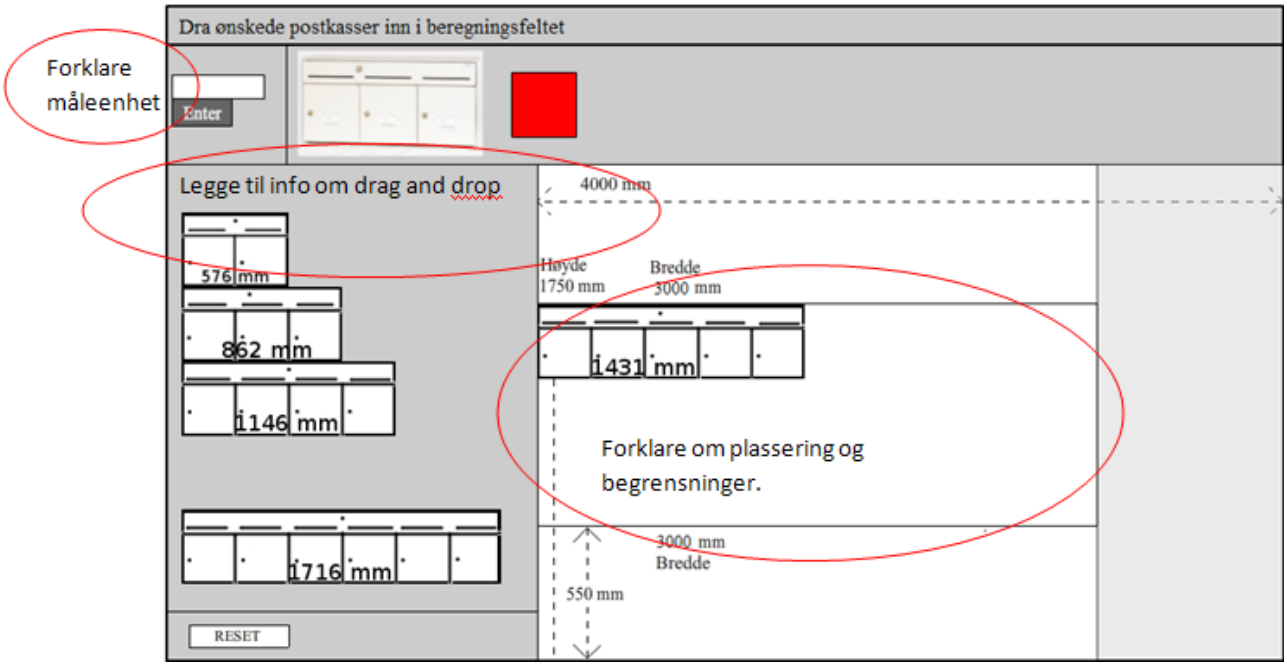
Når papirprototypene var ferdige, valgte vi og ta en enkel brukertest på systemet vårt på våre arbeidsgivere for å høre med de om dette var slik de tenkte i grove trekk at systemet skulle virke og se ut. De ga tilbakemelding om at papirprototypen av enkel å manøvrere og virket som er bra layout for systemet som ønskes.

Designprototype

Designprototypen er en prøve av en liten del av systemet der man legger stor vekt på hvordan systemet skal se ut for kunden. Funksjonalitet er ikke det som er mest vektlagt i en designprototype, derimot hvordan designet på brukergrensesnittet er. Vi valgte å lage en design prototype av den delen av systemet vi følte trengte dette. Vi laget en design prototype av beregningsmodellen som vi testet på noen med studenter. Vi testet beregningsmodellen på fire medstudenter fordi de vil ha eventuelt de samme forkunnskapene som en eventuell kunde, eller arkitekt som skal bruke systemet. Dette gjorde vi for å se hvordan folk reagerte på designet vi har valgt og om systemet og brukervennligheten til systemet svarte til våre forventninger.

Brukertesting av designprototypen

Under brukertesting fant vi fort ut av hva som kunne forbedres. Alle våre testsubjekter nevnte at forklaringsfeltet som var plassert øverst var vanskelig å få øye på. Dermed var testsubjektene ikke klar over hva de skulle gjøre tidlig i prosessen. For å motarbeide dette problemet har vi funnet ut at en mer markant skrifttype / størrelse gjør det enklere og se skriften. I tillegg til dette vil vi sette en forklaring på hva kunden skal skrive inn i feltet. En av våre testpersoner mente at hvis vi plasserte forklaringene nærmere eller i samme ramme som der man skal utføre en aktuell oppgave så ville det vært en forbedring. Det viste seg også at brukerne ikke forsto at det var ”drag and drop” funksjon på postkassene i hovedfeltet så her trengs det også å utarbeides et forklarende felt.



Videre utover de nevnte punktene over syntes brukere at systemet var veldig fint og greit og bruke. Kun tilbakemelding om dårlig forklaringer.

ANALYSE

Mål med analysefasen

Målet med denne fasen er å finne ut hvordan prosjektet skal gjennomføres og bygges opp, og hvilke verktøy som skal benyttes for å få til dette på best mulig måte. Denne fasen er hovedsakelig bestående av ulike skjemaer og tabeller for å illustrere hva vårt system skal kunne gjøre og sammenhengen mellom disse.

Use Case

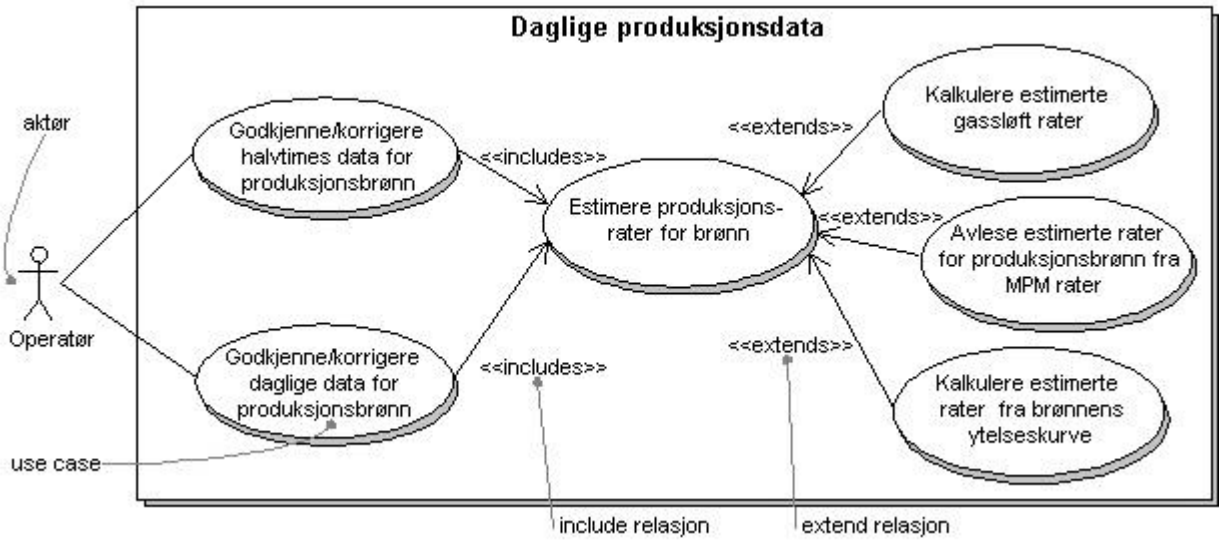
Use Case diagrammer brukes til å illustrere bruken og oppførselen av systemet. Use Case diagrammer forteller hva systemet skal gjøre, men ikke hvordan det skal gjøres. Diagrammene brukes derfor til enten å modellere sammenhengen, oppbygningen av systemet, eller til å vise kravene til systemet.

Use Cases kan forbindes med følgende relasjoner:

- **Include**: En include relasjon mellom Use Case A og Use Case B, hvor pilen peker fra A til B, betyr at A arver oppførselen til B ved en spesifisert lokasjon. Dvs. Use Case A vil alltid inkludere (utføre) Use Case B.

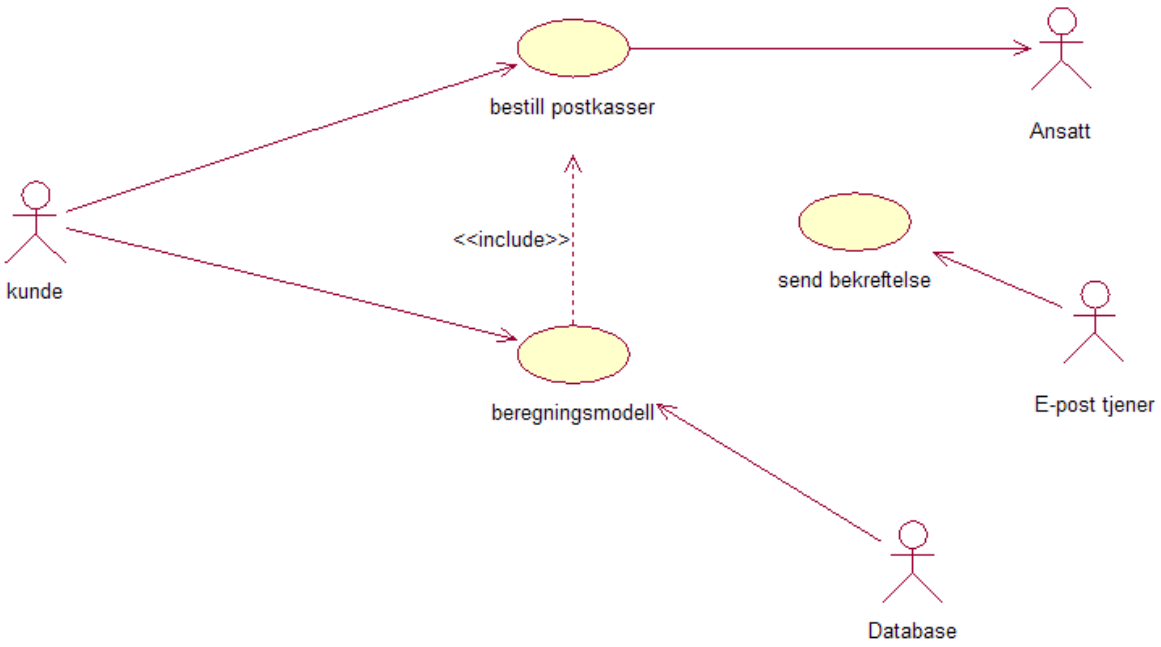
- **Extend**: En extend relasjon fra Use Case A til Use Case B, hvor pilen peker fra A til B, betyr at B legger til oppførselen til A ved en lokasjon spesifisert indirekte i B. Dvs. Use Case B trenger ikke alltid utføre Use Case A. Merk at pilen peker mot Use Casen som blir utvidet.

Eksempel på et Use Case diagram:



Her kan vi se at selve systemet er inne i firkanten, mens aktøren er plassert utenfor. Dette er for å skille handlingene og hvem som utfører de.

Vår Use case



Bilde over viser systemets oppbygning. Der pinnemennene er aktører i systemet. Disse er med på å utføre prosessene. Sirklende er prosessene i systemet. Det er her de forskjellige handlingene er. Pilene mellom aktørene og prosessene viser hvem som utfører de forskjellige prosessene og hvilken retting input eller output av data går.

Use case beskrivelsen under er forklaring på hendelsesforløp i de forskjellige prosessene i Use Case diagrammet. Disse er med i en fullstendig Use Case for å forklare på høyt nivå hvordan systemet skal fungere sammen med aktørene. De er med på å danne et bilde av hvordan systemet skal bli til slutt.

De forskjellige punktene i en Use Case beskrivelse:

1. Use case = Navn på prosessen som beskrivelsen tilhører
2. Aktør = Er hvem som utfører selve prosessen
3. Trigger = Er hva som må til for at prosessen starter.
4. Prebetingelse = betingelse for at denne prosessen kan gjennomføres.
5. Postbetingelse = er resultatet av prosessen blir gjennomført.
6. Normal hendelsesflyt = er hvordan prosessen skal fungere ved normal hendelsesflyt.
7. Variasjoner = er hva som kan skjer i prosessen som ikke er etter normal hendelsesflyt.

Use Case beskrivelser:

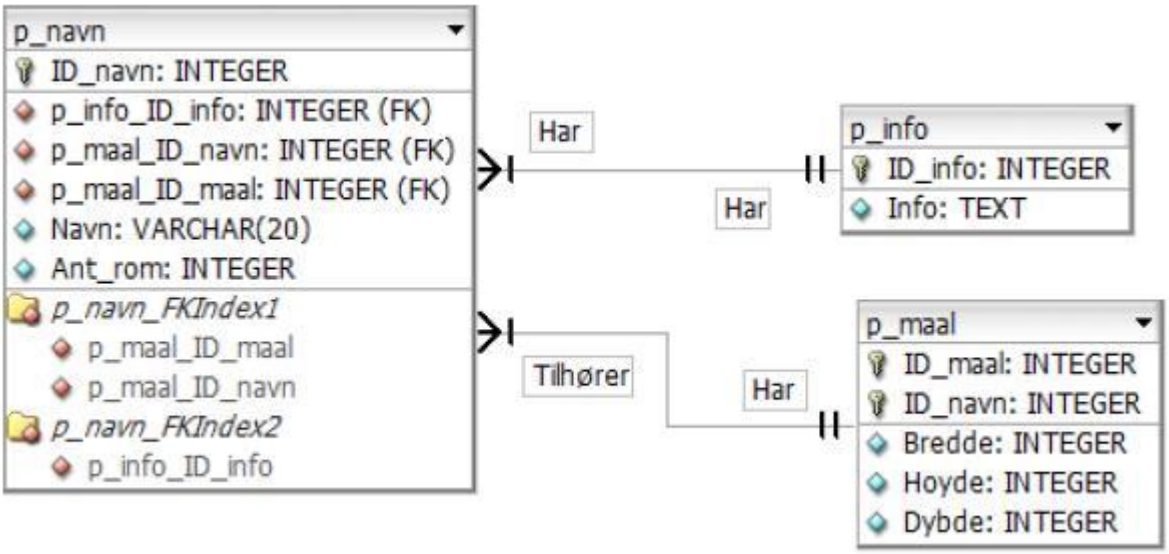
Use Case	Beregningsmodell
Aktør	Kunde
Trigger	Kunde ønsker beregning av plass til postkasser
Prebetingelse	Kunden ønsker å bruke beregningssystem
Postbetingelse	Kunden får en grafisk beregningsmodell av oppgangen med postkasser.
Normal hendelsesflyt	1. Systemet spør om breddemål i oppgangen. 2. Kunden skriver inn breddemål og bekrefter det. 3. Systemet viser postkassetyper. 4. Kunden velger en postkassetype. 5. Systemet viser alle størrelser av valgt postkassetype og setter av et område for plassering av kasser. Den informerer om ”drag and drop” funksjonen. 6. Kunden drar ønskede postkasser over på veggen og godkjenner. 7. Systemet viser tegning av veggen og gir mulighet for utskrift av tegning. Den gir også mulighet for å gå direkte til bestill postkasser.
Variasjoner	6a1. Kunden drar en kasse som ikke passer i forhold til breddemål. Systemet vil informere om at dette ikke er mulig og nekter valget. 6a2. Kunden ønsker flere typer kasser, systemet gjør det mulig og legge til flere forskjellige postkassetyper på veggen.

Use Case beskrivelser for de andre prosessene er vedlagt som vedlegg.

ER-diagram

Et ER-diagram er en illustrasjon av databasen. Denne illustrasjonen viser tabellene i databasen, dets felt og tilhørende datatyper og hvordan de interagerer med hverandre.

Til å begynne med laget vi et ER-diagram med tre forskjellige tabeller. Ideen bak denne strukturen på databasen var å ha en egen tabell for mål på postkassene og en egen infotabell. Dette skulle vise seg å være unødvendig siden info og mål på postkassene er mer eller mindre konstant.



Den første versjonen av vår ER- modell

Vårt endelige ER-diagram er forholdsvis lite, da vi kun har to tabeller. Disse er heller ikke knyttet sammen. Tabellen ”p_navn” inneholder alt av postkasseinfo, som brukes av web-shopen. Den andre tabellen, ”admin”, inneholder brukernavn og passord til administrator for innlogging på websiden. Her kan administrator legge til eller endre produkter/info i databasen direkte fra nettsiden.

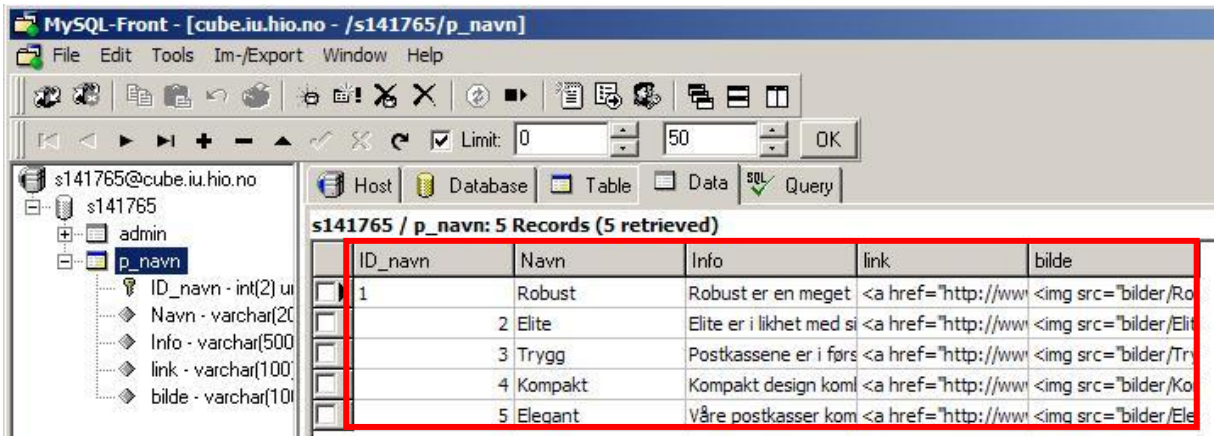
Endelig versjon av ER-modellen

MySQL

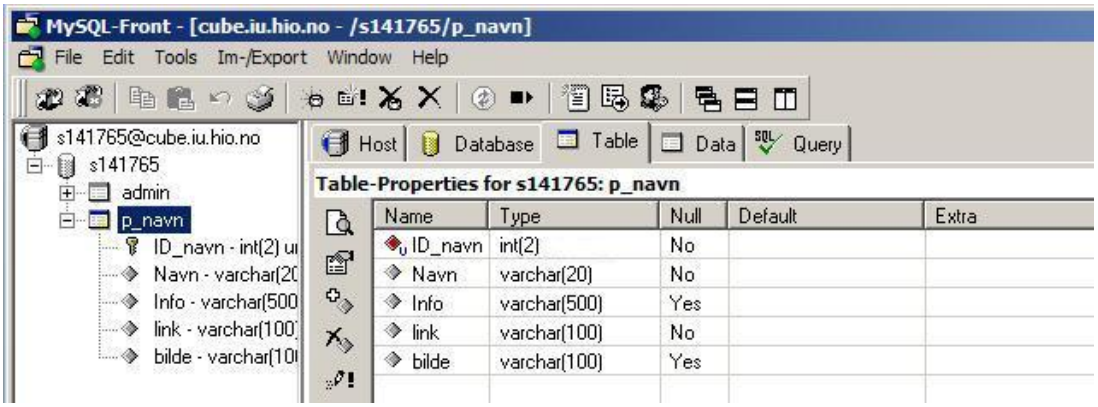
MySQL er det mest populære databasehåndteringssystemet i dag med åpen kildekode (open source). Grunnen til dette er at databasen kan kjøres på de fleste operativsystemer og kan kobles til ved hjelp de aller fleste programmeringsspråkene. Databasen selv er for øvrig skrevet i C og C++. MySQL er også brukt på flere av de mest trafikkerte nettsidene i verden for lagre data og brukerdata, blant annet Google, Youtube og Wikipedia. Dette sier oss at databasen er driftsikker og troverdig.

I vår oppgave benyttes MySQL for å ha et sted å lagre produktdata og hente ut login-info til administrasjonssidene på weben. Webshopen henter da ut info om de forskjellige produktene, inkludert bilder.

Skjermbildet under er tatt fra MySQL-Front, som er et program for å administrere MySQL databaser. Her ser du at de forskjellige postkassemodellene er lagt inn med ett unikt id-nummer, navn, litt info om postkassetypen, link til mer info og til slutt et bilde av den.

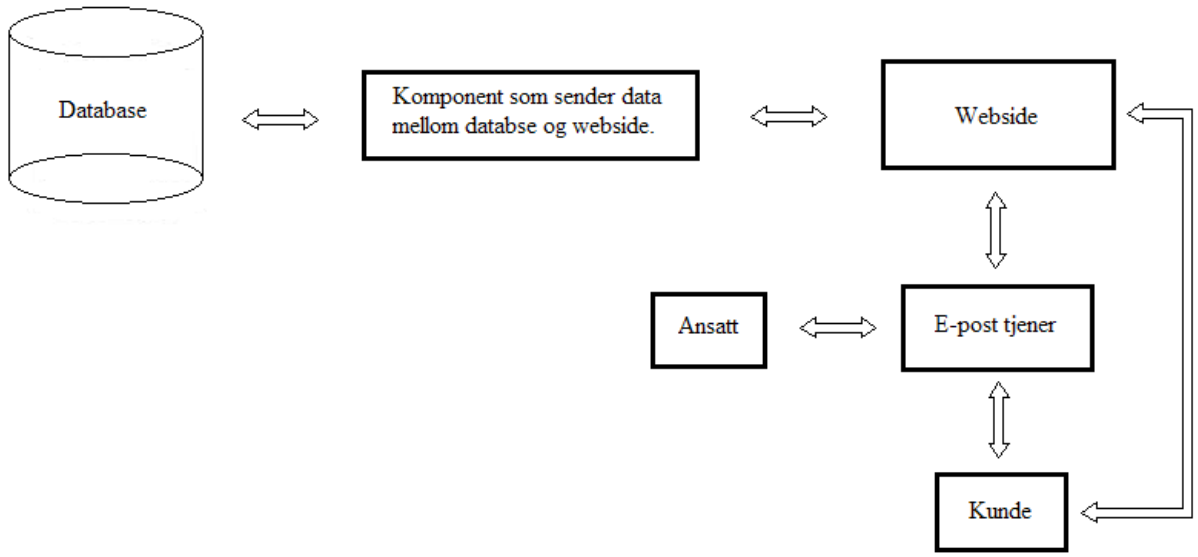


Her kan du se hvordan tabellen, som inneholder dataene du ser i forrige bilde, er bygget opp. Første kolonne (ID_navn) er satt som primærnøkkel. Det vil si at dette nummeret må være unikt, som igjen hindrer dobbeltlagring av produktene. Neste kolonne (Type) sier oss hvilken datatype dette feltet skal inneholde. ”int(2)” indikerer at dette feltet skal ha maks 2 tall. De resterende kolonnene har datatype ”varchar” med et nummer i parentes bak. Dette er skrift med maks antall tegn i parentes. ”P_navn” er navnet på tabellen. Kolonnen ”Null” viser om verdien til det aktuelle feltet kan være null.



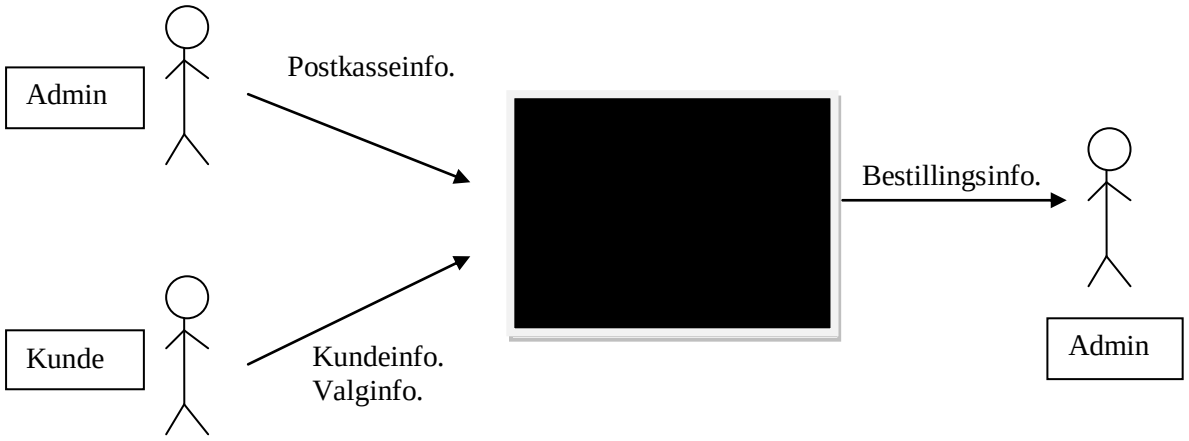
Strukturkart

Et strukturkart er en illustrasjon som viser hvordan systemet henger sammen. Boksene i illustrasjonen simulerer aktører og komponenter. Pilene illustrerer i hvilken retning informasjonsflyten går og mellom hvem/hva. Vi har brukt dette fordi det er et veldig fint og oversiktlig verktøy som gir en klar fremstilling av systemets oppbygging og samhandling.



Illustrasjon over strukturen i systemet.

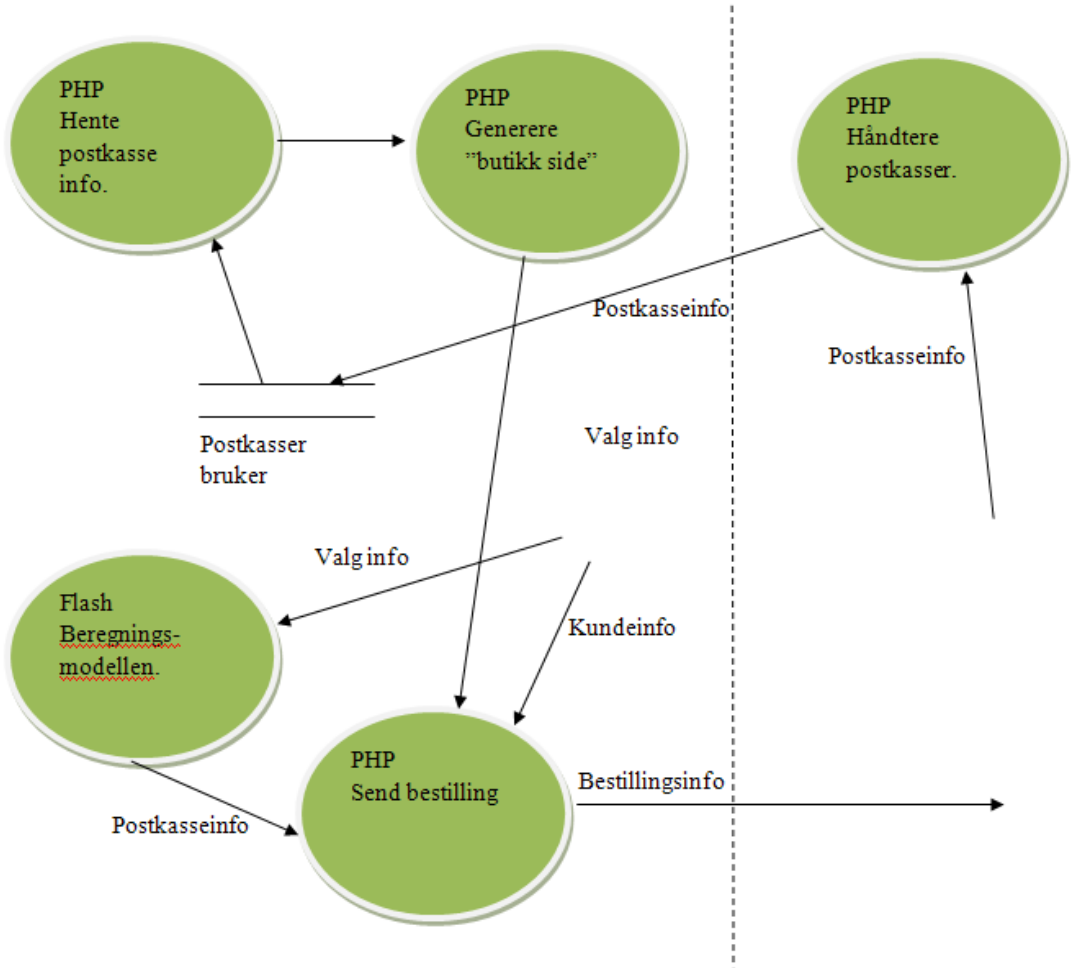
Dataflytdiagram



Et kontekstdiagram gir en oversikt over dataflyten i et informasjonssystem. I kontekstdiagrammet representerer den svarte boksen systemet. Pilene representerer dataflyten mellom aktørene og systemet. Diagrammet viser hvilken informasjon som går fra aktørene til systemet og hvilken informasjon aktørene mottar fra systemet. Aktøren ”Kunde” sender ”Kundeinfo” som er kundens personalia og ”Valginfo” som er kundens valg av postkasser, til systemet. Aktøren ”Admin” sender data om postkasser i ”Postkasseinfo” til systemet. ”Admin” mottar også bestillingsinformasjon fra kunder gjennom ”Bestillingsinfo” fra systemet.

Hvorfor dataflytdiagram:

Vi har laget et dataflytdiagram til vår oppgave for å se hvordan systemets dataflyt henger sammen. Dette vil da gi oss et oversiktlig bilde av hvordan datastrømmen er og hvilke prosesser som er i systemet. Vi kan da se sammenhengen mellom delene i systemet.



Hva er dataflytdiagram:

Dataflydiagrammet viser sammenhengen mellom forskjellige prosesser, datalagring, og ekstern entitet. Dens oppgave er å gi et oversiktlig blikk på hvordan prosessene er bygget opp og sammenhengen mellom dem.

- En sirkel er en prosess. En prosess beskriver en handling eller en transformering av data.
- En pil viser i hvilken retning dataflyten går. Dette vil vise at dataen går fra eller til en prosess.
- To like streker rett over hverandre er et datalager, her er data lagret og pilen fra et slikt lager viser om data hentes ut eller blir puttet inn i datalageret.
- En ekstern entitet er personen eller aktøren som er utenfor systemet og er den som starter de forskjellige prosessene.

DESIGN

Valg av verktøy

Vår kunde har opplyst oss om at vi står fritt til å bruke de løsningene vi synes passer best til å utvikle dette systemet. Vi har valgt å utvikle systemet primært i PHP siden dette er et programmeringsspråk som alle på gruppen har vært borte i. Når det gjelder design på siden har vi valgt å bruke CSS, javascript, flash og html for å utvikle et brukervennlig grensesnitt. For redigering av bilder/grafiske enheter bruker vi GIMP. IBM Rational Rose brukes i planleggingsperioden for å få et konkret overblikk over systemets oppbygning og hendelsesforløp. Databasen er en mySQL database og designes i DB designer, hvor blant annet alle tabeller og deres sammenhørighet planlegges.

Litt om verktøyene

HTML
HTML **(**"HyperText Markup Language") er et programmeringsspråk for det å lage nettsider. I et slik dokument kan man legge til den informasjonen som skal vises på en nettside. Vi bruker dette språket til å utvikle nettsiden.

PHP
PHP **(** "Hypertext Preprocessor") er et verktøy som brukes til å utvikle dynamiske nettsider. PHP er programkode som kjøres på serveren ikke på brukerens maskin. Dette vil si at når du som bruker besøker en nettside vil koden bli kjørt på serveren du kontakter, mens resultatet av denne kjørte koden blir sendt til deg. PHP gjør det mulig og koble nettsiden din opp mot database og kan motta input fra bruker og gjøre operasjoner på dataen.

Vi har valgt å bruke PHP i vårt prosjekt fordi det er et programmeringsspråk som vi har vært borti før og vi vet om dens muligheter innen nettside bygging. Likevel trenger vi også og lære mer om sikkerhet i php, siden vårt prosjekt krever noe mer sikkerhet en vi har laget tidligere.

CSS
CSS **(**"Cascading Style Sheets") brukes til og forme design og utseende til en nettside. Nettsider er skrevet i html eller xml, ved bruk av CSS som suplemang til slike filer kan du ha mye av utforming, design og utseende valg i CSS.

Vi bruker CSS for nettopp dette formål, det og kunne forme sidene slik vi ønsker de skal være ut mot kunden.

GIMP
Gimp er et program for manipulering å behandle digital grafikk og fotografier.

Vi bruker GIMP til å produsere bilder for design av nettsiden, vi bruker også dette programmet til og lage bilder til beregningsmodellen i prosjektet, dette mest fordi bildene som brukes i flash animasjonen skal stemme overens med virkelige mål. Bildene må da lages presist i størrelser.

MySQL er det mest populære databasehåndteringssystemet i dag med åpen kildekode (open source). Grunnen til dette kan være at databasen kan kjøres på de fleste operativsystemer og kan kobles til ved hjelp de aller fleste programmeringsspråkene. Databasen selv er for øvrig skrevet i C og C++. MySQL er også brukt på flere av de mest trafikkerte nettsidene i verden for lagre data og brukerdata, blant annet Google, Youtube og Wikipedia. Dette sier oss at databasen er driftsikker og troverdig.

I vår oppgave benyttes MySQL for å ha et sted å lagre produktdata og hente ut login-info til administrasjonssidene på weben. Webshopen henter da ut info om de forskjellige produktene, inkludert bilder.

Adobe Flash

Historie

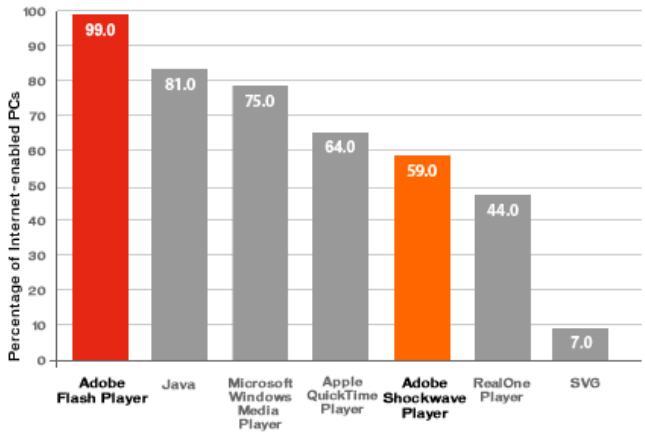
Flash er en av de hotteste innholds-teknologiene på weben i dag. Programmet så dagens lys i juli 1996 under navnet FutureSplash Animator. På samme tid var Macromedia også i gang med å lage sitt eget system under navnet Shockwave. Ressursene til Macromedia og produsenten av FutureSplash Animator ble slått sammen til ett program — Flash 1. i desember 1996. Siden det har programmet utviklet seg til å bli et meget kraftig verktøy for å lage webanimasjoner.

Flash brukes i dag til å lage alle mulige typer av webinnhold som: spill, tegnefilmer, logoer og annonser, men har med litt scripting også mulighetene for å kunne være en "stand alone" webløsning hvor alt innholdet for en side er presentert i Flash istedenfor HTML. Flash kan også knyttes opp mot databaser og presentere XML data.

Worldwide Ubiquity of Adobe Flash Player by Version - December 2008

	Flash Player 7	Flash Player 8	Flash Player 9	Flash Player 10
Mature Markets[1]	99.1%	99.0%	98.6%	55.9%
US/Canada	99.1%	99.1%	98.9%	54.5%
Europe[2]	99.1%	98.9%	98.2%	56.5%
Japan	99.0%	98.8%	98.3%	59.3%
Emerging Markets[3]	98.7%	98.6%	98.1%	55.9%

Oversikt over hvilke versjoner av flash som benyttes i dagens marked (per desember 2008)



Flash er verdens mest utbredte software plattform (basert på markedet i USA, Canada, Storbritannia, Frankrike og Tyskland).

Formater

Når du lager en Flash film bruker du kanskje både bilder, lyd, tekst og animasjon som du pakker sammen i en fil som presenteres på weben.

Flashprogrammet som du lager dette i kan lagre disse filene i to forskjellige formater:

- **fla**
  "Work in progress" filen som brukes av Flash for å lagre originalfilen. Denne kan være relativt stor i størrelse (kb).

- **swf** (uttales *swiff*)
  Den ferdige filen som eksporteres fra Flash for bruk på weben. Dette er en komprimert fil som tar mye mindre plass enn .fla filen.

Når en bruker besøker siden din blir *.swf filen* lastet ned på maskinen hans/hennes. Ved hjelp av en plug-in, eller en programutvidelse for browseren, vil brukeren kunne se din flashanimasjon. For de brukerne som har plug-in vil sidene se helt like ut, uavhengig av maskin og browser. Flash har også sitt eget programmeringsspråk, ”ActionScript”. Foreløpig siste versjon av AS er 3.0, som kom med Flash Player 9.

Noen av de mest kjente programmene for å utvikle flash er Adobe Flash, Adobe Flex, Adobe Air, Swift 3d og Swish.

Fordeler

Flash er bygget på vektorer. All grafikk du lager/tegner i Flash er basert på vektorer, i motsetning til bilder som er bygget opp med pixler. Fordelen med vektor-grafikk er at den er matematisk basert og alltid vil være like sylskarp i kanter og overganger uansett hvor mye du forstørrer grafikken, i motsetning til bilder (gif/jpeg) som vil bli pixelert eller hakkete i kanten når du forstørrer.

Det at den er matematisk basert betyr også at den fysiske størrelsen på grafikken ikke har noe å si for filstørrelse. Du kan lage en firkant fylt med blå farge så stor som et frimerke, så lage en kopi som du forstørrer opp så mye du vil, og lagre begge filene. De er og vil alltid være like store.

At Flash er vektorbasert, betyr at selve grafikkfilene vi lager i programmet er relativt kompakte og raskt nedlastbare. Det som virkelig kan gjøre prosjektet ditt stort er når vi begynner å animere innholdet. Men ved å planlegge Flash-prosjektet grundig på forhånd, kan du lage raskt nedlastbare eller streamede del-filer.

Det at grafikken i Flash kan animeres er selvfølgelig en stor fordel.

Ulemper

En av de største ulempene er også nettopp det at Flash er gøy å jobbe med, og da har ting litt lett for å ta av. Nøyer man seg med statiske små Flashbildefiler vil ikke dette være noe problem, men Flash er et animasjonsprogram, og det er i animasjonsprosessen at filene virkelig legger på seg og blir tungt nedlastbare.

En annen ulempe er at flashprogrammene kan bli for avanserte og dermed virke mot sin hensikt. Designet går på bekostning av ytelsen.

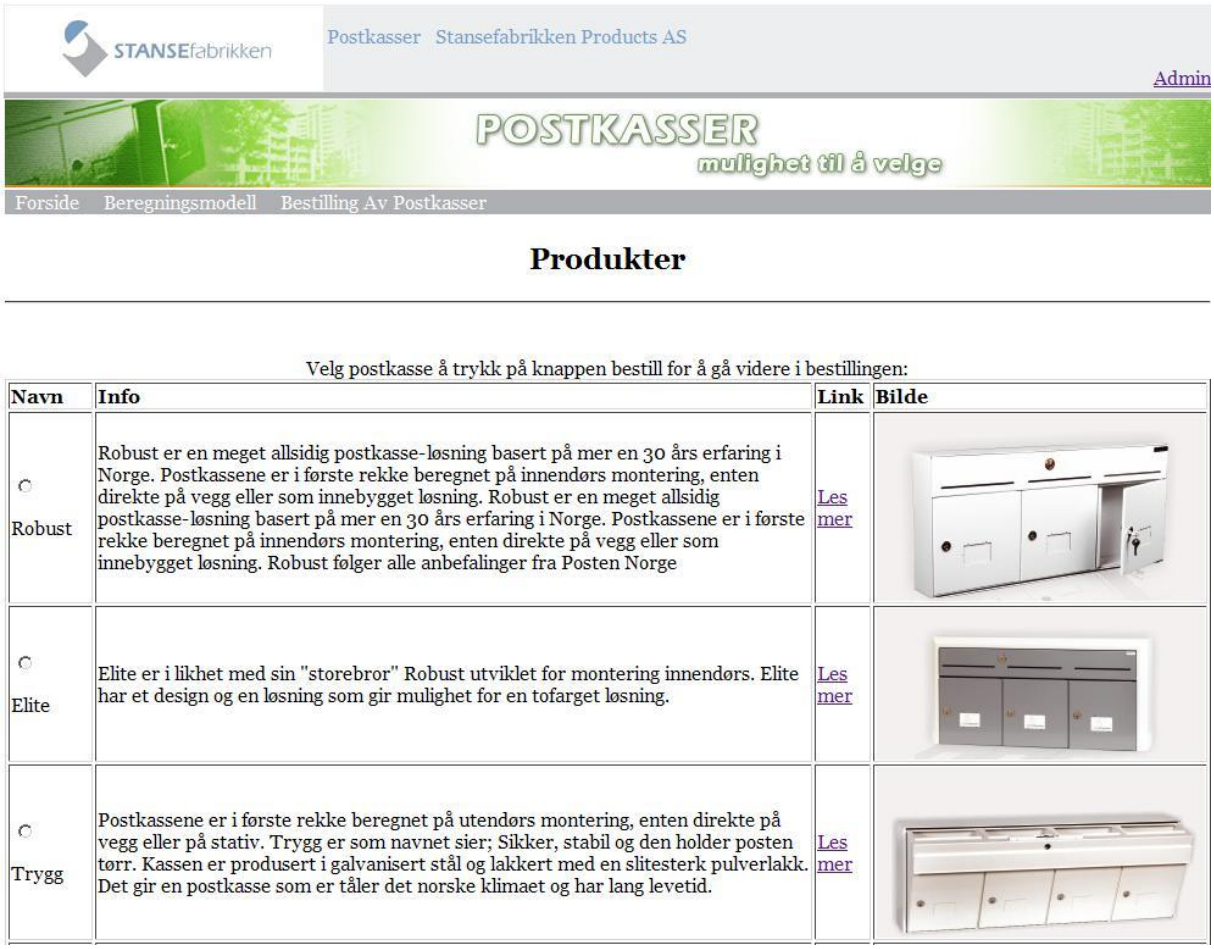
IMPLEMENTASJON

Webshop

Presentasjon av Webshop

Stansefabrikken Products AS ønsket seg en webshop, slik at kunder kunne bestille produkter på nettet for å få mer automatikk i salgsprosessen. Per i dag har de kun informasjon om de forskjellige postkassetypene på nett. Alt salg foregår via telefon og e-post. Stansefabrikken Products AS ønsket et bestillingssystem som gjør det mulig å bestille postkasser på nettet.

Når man trykker på linken bestilling av postkasser får man skjermbildet som er vist under. Her kan kunden velge en postkassetype han eller hun ønsker å bestille. Nederst på siden har man en videre knapp som sender kunden til neste side for bestilling.



En annen mulighet kunden har på siden over er å lese mer om produktet de ønsker. Hvis kunden ønsker mer informasjon vil linken sende dem til nettsiden til stansefabrikken Products AS hvor all informasjon om postkassetypen er nevnt.

På den neste bestillings siden vil valgt postkassetype vises til kunden. Systemet vil så spørre om antall postkasser som ønskes:

Her fyller kunden ut antall rom og antall oppganger disse er fordelt over. Dette steget blir fullført med neste knappen. Når kunden har fullført dette steget vil han eller hun bli sendt til neste bestillingsside.

Dette er den siste siden kunden trenger å fylle ut før bestilling og bekreftelse på denne blir sendt. Her blir kunden bedt om informasjon vedrørende kontakt fra selger, samt hvor kunden skal ha postkassene montert.

Du ønsker å bestille:

Robust	Robust er en meget allsidig postkasse-løsning basert på mer en 30 års erfaring i Norge. Postkassene er i første rekke beregnet på innendørs montering, enten direkte på vegg eller som innebygget løsning. Robust er en meget allsidig postkasse-løsning basert på mer en 30 års erfaring i Norge. Postkassene er i første rekke beregnet på innendørs montering, enten direkte på vegg eller som innebygget løsning. Robust følger alle anbefalinger fra Posten Norge	Les mer	

Din henvendelse gjelder 10 rom
Din henvendelse gjelder 10 oppganger

Dine personalia

Navn:
Adresse:
E-post adresse:
Tlf:
Monteringsadresse:
Adresse:
Postnr.:
Poststed:
Tilbehør eller andre ønsker:
Ønsker du andre farger eller tilleggsutstyr nevn det her...
Bestill

Nederst på siden har du bestill knappen. Når denne blir trykket på vil bestillingen bli sendt. Kunden vil få en bekreftelse på denne bestillingen både på e-post og på skjermen.

Admin siden

I tillegg til bestillingssystemet har vi utviklet en admin side. Hensikten med denne siden er at Stansefabrikken sine ansatte skal ha mulighet til å legge til nye produkter i salgstabellen. De skal også ha mulighet til å fjerne produkter. Øverst til høyere på siden har vi lagt til en link til denne funksjonaliteten. For at en skal komme til nettsiden der man registrerer nye produkter eller fjerner dem må da en admin trykke her.

Da vil denne personen bli sendt til logg inn vinduet som ser slik ut:

Her blir man da nødt til å skrive inn passord og brukernavn. Når det er gjort trykker man på logg inn. Kun når passordet og brukernavnet er riktig vil man bli sendt til innlogget siden.

Innlogget siden ser slik ut:

Du er inlogget:

Navn: Info Link Bilde

navn | Ønsker du andre farger eller tilleggsutstyr nevn det her... | link | Elegant2

Sett inn

Foreløpig ser tabellen slik ut:

Navn	Info	Link	Bilde	
Robust	Robust er en meget allsidig postkasse-løsning basert på mer en 30 års erfaring i Norge. Postkassene er i første rekke beregnet på innendørs montering, enten direkte på vegg eller som innebygget løsning. Robust er en meget allsidig postkasse-løsning basert på mer en 30 års erfaring i Norge. Postkassene er i første rekke beregnet på innendørs montering, enten direkte på vegg eller som innebygget løsning. Robust følger alle anbefalinger fra Posten Norge	Les mer		slett
Elite	Elite er i likhet med sin "storebror" Robust utviklet for montering innendørs. Elite har et design og en løsning som gir mulighet for en tofarget løsning.	Les mer		slett
Trygg	Postkassene er i første rekke beregnet på utendørs montering, enten direkte på vegg eller på stativ. Trygg er som navnet sier; Sikker, stabil og den holder posten tørr. Kassen er produsert i galvanisert stål og lakkert med en slitesterk pulverlakk. Det gir en postkasse som er tåler det norske klimaet og har lang levetid.	Les mer		slett

Her skal en ansatt hos stansefabrikken kunne legge til nye postkasser med den informasjonen som trengs. Her kan man også se tabellen over innholdet slik den er til en hver tid. Hvis man ønsker og fjerne en post fra tabellen har man en slett knapp på siden til dette formålet.

PHP

Forsiden

På forsiden til Stansefabrikken Products AS brukes det tre linker til å forflytte seg rundt på siden. De er laget med html koden:

Bestilling Av postkasser

Denne koden lager en link. Når linken blir trykket på blir personen sendt til ”index.php?pg=bestill” der pg er variabelen som da inneholder bestill. Variabelen pg vil på denne måten bli sendt som en ”GET” variabel. En GET variabel i PHP er en måte og ta vare på en variabel i headeren fra side til side. Dermed vil variabelen pg inneholde bestill hvis denne linken blir trykket på.

På bildet under ser du php koden som viser innholdet på nettsiden. Det denne koden gjør er at den tar en test på hva variabelen pg inneholder. Hvis pg inneholder bestill vil da php scriptet kjøre koden ”include bestill.php”. Da vil innholdet av bestill1.php vises i index.php sin tabell. På denne måten oppnår vi at utseende på siden rundt hovedvinduet i tabellen ikke forandrer seg uansett hvilken link kunden går til.

```
62          <?php
63                  if($_GET['pg'] == null)
64                  {
65                      include "forside.php";
66                  }
67                  elseif($_GET['pg'] == "beregning")
68                  {
69                      include "Beregningsmodell.html";
70                  }
71                  elseif($_GET['pg'] == "bestill")
72                  {
73                      include "bestill1.php";
74                  }
75                  elseif($_GET['pg'] == "bestill2")
76                  {
77                      include "bestill2.php";
78                  }
79                  elseif($_GET['pg'] == "bestill3")
80                  {
81                      include "bestill3.php";
82                  }
83                  elseif($_GET['pg'] == "bestill4")
84                  {
85                      include "bestill4.php";
86                  }
87                  else
88                  {
89                      include "forside.php";
90                  }
91              ?>
```

For å gjøre en test på innholdet i variabelen pg bruker vi en ”if løkke”. Det en if kommando gjør er at den kjører en test før den lar scriptet kjøre den koden som er inni klammene under.

If($_GET[’pg’]) == ”bestill”)
{ Koden som blir utført }

Koden inne i eksemplet over blir ikke kjørt hvis ikke pg er satt lik bestill.

Bestill postkasser

Under bestilling av postkasser behøves det at nettsiden kan koble opp mot en database der informasjon til de forskjellige postkassene legges. For at PHP skal klare å koble til en database bruker vi koden som er vist i bildet under.

```php
<?php
$tilkobling = mysql_connect( 'localhost', 's141765', '' ) or die( "problemer med tilkobling" );
mysql_select_db( 's141765' , $tilkobling ) or die( "Kunne ikke velge database" );
$rs = mysql_query( "SELECT * FROM p_navn" );
?>
```

Det denne koden gjør er at den oppretter en tilkobling til database serveren og holder denne oppe til vi stenger den igjen. Når dette er utført har man mulighet til å hente ut data fra tabellene. ”$rs = mysql_query (”SELECT * FROM p_navn”);” gjør at alt innholdet i tabellen ”p_navn” blir puttet i variabelen ”rs”.

Under henter vi ut informasjon fra variabelen rs og putter dette i en tabell som blir vist til kunden som er på nettsiden. Det koden under gjør er at den plukker ut hver kolonne i en linje og skriver de til en variabel, som da blir puttet i tabellen. Her brukes en ”while løkke” som gjennomløper hele tabellen i mysql og skriver ut denne til en tabell til kunden. Resultatet av denne koden er at kunden får en tabell med alle postkasseproduktene som er lagt inn i mysql tabellen p_navn.

```php
<?php
while( $obj = mysql_fetch_object( $rs ) )
{
echo "
<tr>
<td><input type='radio' name='systemkasser' value='$obj->Navn'/> <br></br> $obj->Navn</td>
<td>$obj->Info</td>
<td>$obj->link</td>
<td>$obj->bilde</td>
</tr>\n";
}
mysql_free_result( $rs );
mysql_close($tilkobling);
?>
```

Dette blir da den første bestill siden. Siden kunden skal ha mulighet til å velge en postkasse han eller hun måtte ønske å bestille har vi brukt ”FORMs”. Form har vi brukt til og lagre innhold av variabler til neste nettside med POST variabler. Postvariabler er variabler som ligger skjult for brukeren av nettsiden. På samme måte som GET variabler gjør det mulig og sende data fra en side til en annen kan man oppnå det samme med en POST variabel.

Når man bruker koden:

<form id=”form2” name=”form2” method=”post” action=”index.php?pg=bestill2”>
[her kommer html kode som lager de forskjellige variablene]

</form>

Vil dette si at formet får id form 2. Måten formet sender informasjon er ”method=post”, hadde det stått GET her ville den sendt informasjonen med headeren. ”action =” spesifiserer hvor formet ender etter at den blir fullført.

Da vil variablene som blir laget, sendt med hit. De er da mulig og hente opp igjen her på denne måten:

Eksempel:
hvis variabelens navn er postkasser:
$v = $_REQUEST["postkasser"];

I dette tilfellet vil innholdet av variabelen postkasser legge seg i variabelen $v.

Hvis variabelen skulle skrives ut på skjermen kunne man skrevet:
echo ” $_REQUEST[”postkasser”]”

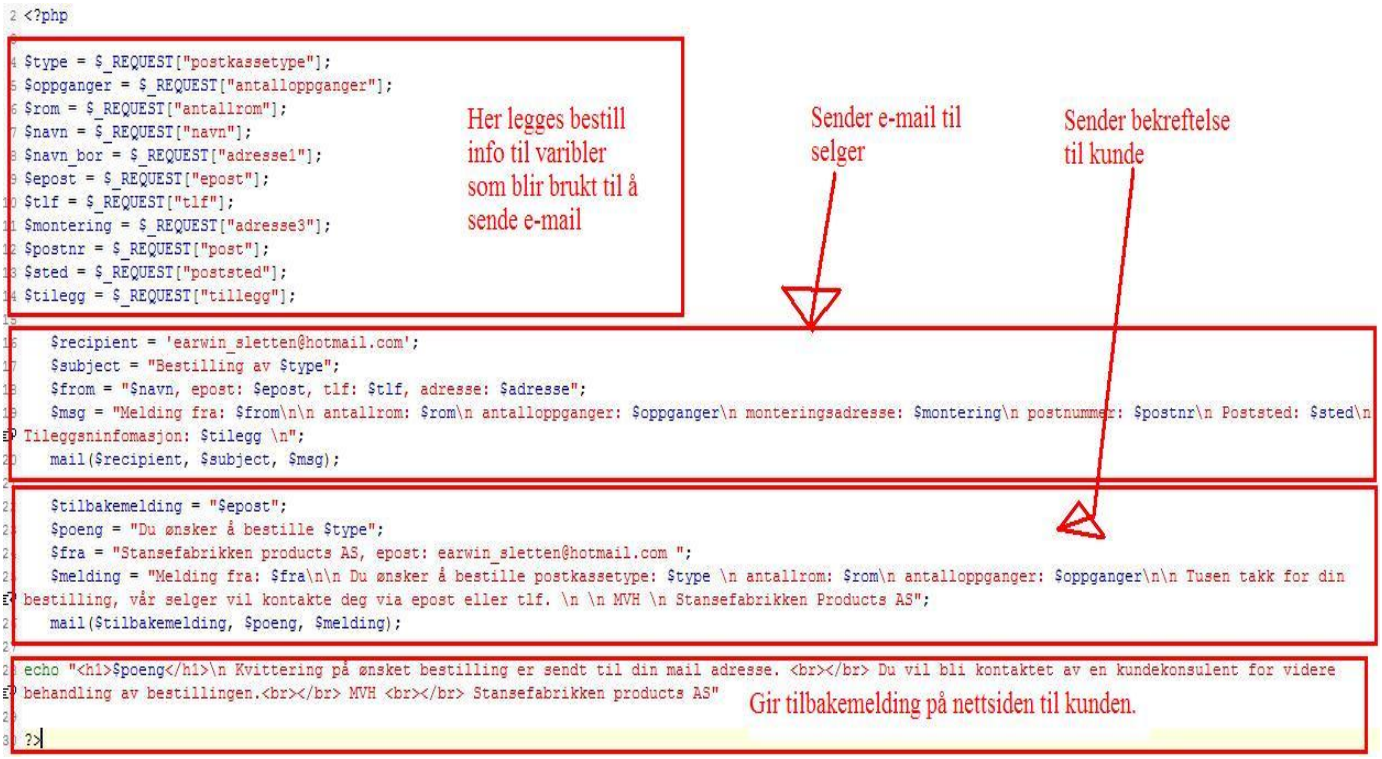
Echo er direkte utskrift. Dette kan være tekst og variabler. Du kan også bruke echo til og skrive ut html tagger. Noe som er veldig praktisk med tanke på utforming av utseende på elementer som blir hentet fra database.

```
<form id="form2" name="form2" method="post" action="index.php?pg=bestill2"
<table border='1'>
<tr>
<td><b>Navn</b></td>
<td><b>Info</b></td>
<td><b>Link</b></td>
<td><b>Bilde</b></td>
</tr>
<?php
```

(Her er php koden i bildet over)

```
?>
</table>
<br></br>
<p><input type="submit" value="Bestill" /></p>
</form>
```

Ved bruk av post variabler sender vi data som trengs til bestillingen videre. Denne informasjonen puttes så til slutt inn i en e-post til kunden for bekreftelse og til selgeren av produktet. Dette gjør vi ved kommandoene som vist i bildet under:




```php
<?php

$type = $_REQUEST["postkassetype"];
$oppganger = $_REQUEST["antalloppganger"];
$rom = $_REQUEST["antallrom"];
$navn = $_REQUEST["navn"];
$navn_bor = $_REQUEST["adresse1"];
$epost = $_REQUEST["epost"];
$tlf = $_REQUEST["tlf"];
$montering = $_REQUEST["adresse3"];
$postnr = $_REQUEST["post"];
$sted = $_REQUEST["poststed"];
$tilegg = $_REQUEST["tillegg"];

    $recipient = 'earwin_sletten@hotmail.com';
    $subject = "Bestilling av $type";
    $from = "$navn, epost: $epost, tlf: $tlf, adresse: $adresse";
    $msg = "Melding fra: $from\n\n antallrom: $rom\n antalloppganger: $oppganger\n monteringsadresse: $montering\n postnummer: $postnr\n Poststed: $sted\n
Tileggsninfomasjon: $tilegg \n";
    mail($recipient, $subject, $msg);

    $tilbakemelding = "$epost";
    $poeng = "Du ønsker å bestille $type";
    $fra = "Stansefabrikken products AS, epost: earwin_sletten@hotmail.com ";
    $melding = "Melding fra: $fra\n\n Du ønsker å bestille postkassetype: $type \n antallrom: $rom\n antalloppganger: $oppganger\n\n Tusen takk for din
bestilling, vår selger vil kontakte deg via epost eller tlf. \n \n MVH \n Stansefabrikken Products AS";
    mail($tilbakemelding, $poeng, $melding);

echo "<h1>$poeng</h1>\n Kvittering på ønsket bestilling er sendt til din mail adresse. <br></br> Du vil bli kontaktet av en kundekonsulent for videre
behandling av bestillingen.<br></br> MVH <br></br> Stansefabrikken products AS"

?>
```

Det vi i korte trekk gjør her er at vi henter ut all informasjon som kommer fra forrige side og putter disse i variabler. Dette er da all informasjon som trengs for å lage en bestilling. Det neste vi gjør er å sende to mail. Den ene er til den som bestiller, dette blir da en bekreftelse på bestilling. Den andre blir sendt til selger.

Mailen sender vi med kommandoen:

Mail(”mottaker”, ”emne”, ”innhold”);

I de forskjellige variablene til det som kommer med i e-posten putter vi da informasjon fra variablene som er spesifisert helt på toppen i bildet over.

Sikkerhet i PHP

Vi har tidligere sett litt hvordan PHP fungerer og hvordan vi har utviklet nettsiden. Når man lager en nettside hvor personer utenfra skal ha slik funksjonalitet er det viktig å tenke litt på sikkerhets aspekter rundt dette.

Når man jobber med input fra brukere er det viktig med sikkerhet. Grunnen til at dette er viktig kommer av at det finnes sikkerhetshull om man ikke tar en test på all input fra brukeren. Et slikt sikkerhetshull er sql-injections. Det en person gjør ved bruk av sql-injection er at han eller hun gir feil input til databasen. Dette kan gi tilgang til deler av databasen som personen ikke skal ha. Det som brukes er tegn som sql databasen tolker som kode slutt eller kommentarer. Dermed kan brukeren skrive inn sin egen kode eller fjerne deler av en spørring, dette resulterer i at feil kode blir kjørt på sql databasen. Sql databasen kan da gi fra seg informasjon om innholdet i tabellene eller kjøre kode som skader eller endrer innholdet. Dette er ikke en ønsket situasjon. For og unngå dette må man gjøre en test på hva brukeren av nettsiden sender inn og fjerne tegn som kan skape problemer.

En måte og gjøre dette på er med koden:

```
15
16 // 'SELECT * FROM p_navn WHERE Navn='' OR ''='';
17 $sql = 'SELECT * FROM p_navn WHERE Navn=\'' . mysql_real_escape_string ( $v ) . '\'';
18 $rs = mysql_query ( $sql );
19 //  or die ("Query failed: " . mysql_error());
20
```

Variabelen $v er i denne situasjonen data som kommer fra brukeren. Her har vi lagt til mysql_real_escape_string, det denne gjør er at den fjerner alle tegn som brukeren kan benytte til og kjøre sin egen kode ved at den legger til skråstrek foran disse tegnene. Dermed vil ikke sql-injection fungere med en enkel input fra brukeren. Slik kode har vi brukt på all input fra brukeren som blir brukt i spørring mot databasen. Dermed vil ikke slike tegn være mulig å utnytte andre steder i koden vår.

Vi har laget en admin side som krever passord og brukernavn for å logge inn. Admin skal ha mulighet til å legge til nye postkasser i systemet og gjøre endring på databasen. Vi trenger da sikkerhet på innlogging.

For å løse dette problemet har vi brukt 2 sikkerhetsmekanismer. Det ene var å kryptere passordet med md5 og bruke et salt. Et salt er en liten string (satt sammen av tall/bokstaver) som legges med når man krypterer passordet. Det md5 gjør er at den lager en hash ut i fra en fil eller noe lignende. En hash er en string med hexsadesimaler, denne blir laget da passordet blir kryptert med md5 og vil være konsist så lenge passordet er det samme. Hvis passordet ikke er det samme vil hash summen bli helt annerledes. Dermed kan vi gjøre en test på hasen vi legger i databasen mot den vi lager av brukerens input.

Fordelen med md5 er at det er nærmest umulig og ta hasen og finne tilbake til passordet. Noe som skaper bedre sikkerhet.

```
10 if(isset($_POST['passord']))
11 {
12 $pass = mysql_real_escape_string(md5($_POST['passord'].$salt));
13 }
14 $sql = "SELECT * FROM admin WHERE username='$username' AND pwd='$pass'";
15
16 $rs = mysql_query($sql);
```

Det som skjer i koden i bildet over er at den lager en hash med md5 og et salt på passordet kunden skriver inn. Så kommer en spørring til databasen hvor den tester passordet som er laget opp mot hashen vi har laget som ligger i databasen. Hvis det finnes en rad i tabellen hvor både passordet og brukernavnet stemmer overens vil variabelen ”$sql” bli satt med innholdet. Så kjører vi ”mysql_query($sql)” som gir variabelen ”$rs” enten innholdet av linjen eller innholdet ”false”.

Etter denne spørringen tar vi en test på variabelen ”$rs”. Koden i linje nummer to i bildet under tester innholdet i variabelen ”$rs”. Hvis ”$rs” er satt vil koden i klammen under bli kjørt. Hvis den ikke er satt vil koden i klammene etter ”else” bli kjørt. Der koden blir kjørt lages en sessioncookie hvor variabelen log = ’in’ eller ’out’. Hvis den er in vil brukeren få tilgang til admin siden, om den er out vil brukeren ikke få tilgang til admin siden.

```
//mysql_num_rows returnerer antall rader, eller false når den ikke finner noe.
if(mysql_num_rows( $rs))
{
//setcookie( 'admin', 1, time()+3600 );
$_SESSION['log'] = 'in';
header( 'location: innlogget.php');
}
else
{
//setcookie( 'admin', 0, time()-3600 );
$_SESSION['log'] = 'out';
header( 'location: index.php');
}
mysql_close($tilkobling);
?>
```

For og unngå at en person bare går rett til siden innlogget.php hvor admin er innlogget tester vi bare om brukeren har session cookien som har innholdet ”in”. Hvis han eller hun ikke har det kaster vi personen tilbake til forsiden.

Denne testen gjøres slik:

```
if ($_SESSION ['log'] == 'in') {
echo "du er innlogget";
} else {
    header( 'location: index.php');
}
```

I slutten av vårt bestillingssystem sendes det en e-post til den som bestiller og selgeren. Et sikkerhetsaspekt ved dette er at det er mulig og skrive HTML og PHP kode inn i e-posten. Vi ønsker at dette ikke er mulig for brukeren å gjøre. For å unngå dette bruker vi strip tags. Strip tags gjør at all HTML kode fjernes fra variabelen. Under bruker vi strip tags for å fjerne dette før informasjonen blir brukt i bekreftelse og bestillings e-posten.

CSS implementasjon

Vi har brukt CSS til å forme utseende på Stansefabrikken sin nettside. For at HTML skal implementere regler som er skrevet for dens utseende i CSS må denne linjen være til stede i dokumentet inne i taggen <head>. Som vist på bilde under:

```
13 <!--her linker vi til CSS dokumentet som inneholder design oppsett-->
14 <link href="stylesheet.css" rel="stylesheet" type="text/css" >
15 <link rel="icon" type="image/ico" href="favicon.ico" >
16
```

Linje 14 på bildet over vil da gjøre at de reglene vi har i Stylesheet.css vil påvirke nettsiden. Der det står href=”stylesheet.css” viser den til filen som skal inneholde css kommandoer.

Stilark.css

I css har du mulighet til å lage egne klasser til deler av html dokumentet, hvor du kan bestemme utseende til en bestemt del.

For eksempel:
body {Her kommer regler for denne delen }

Det som kommer mellom klammene vil da være reglene for body. Siden body er hele hovedfeltet i html dokumentet vil da reglene inne her påvirke hele nettsiden. Det går også an å lage et navn på en slik klasse som da påvirker kun en liten del. Hvis dette var bare en tabell kunne det sett slik ut.

.tabellen { regler}

Fordi man med denne koden lager en egen klasse blir man da nødt til å hente denne opp i html dokumentet før den tas i bruk på denne måten:

<table class=”tabellen”>

Da vil tabellene som har class=”tabellen” bli påvirket av denne klassen.

Den første delen av css dokumentet sier hvordan nettsidens grunnoppsett skal være. Det som er spesifisert i body på bildet under vil da bestemme utseende på alt som ikke blir spesifisert noe annet sted i css dokumentet.

Denne koden gjør at siden bruker skrifttype ”Georgia Fantasy serif”, med skriftstørrelse 10. Fargen på skriften blir da svart over hele siden hvor det ikke er bestemt annet. ”Color : #00000” bestemmer denne fargen. #00000 refererer til hexadesimaler som spesifiserer en enkel farge kode. Med en slik fargekode kan man få nøyaktig den fargen man måtte ønske på et element. Fargen kan påvirke kanter på en tabell, bakgrunnsfarger, tekst og lignende. I tilfellet under er det tekst fargen.

```
/*-------------------------------------------------------------------------------------------------------
klassen for hele siden utenom det som blir spesifisert lengre nede.
-------------------------------------------------------------------------------------------------------*/
body {font-family: "georgia", fantasy, serif;
                font-size: 10pt;
                font-style: normal;
                font-weight: normal;
                color:#000000;
}

/*-----------------------------------------Slutt body-----------------------------------------------*/

```

Den neste delen vi har spesifisert utseende på i stylesheet.css er tabellen til hovedsiden. I denne delen bestemmer vi regler for hvordan hovedtabellen skal se ut.

```
/*-------------------------------------------------------------------------------------------------------
Tabelloppsett og design av tabellen.
-------------------------------------------------------------------------------------------------------*/
/* Denne classen påvirker hele tabellen og bestemmer bredde og plassering*/
.hovedtable {
width: 989px;
margin: auto;
align: center;
border-collapse: collapse;
}

/* denne setter design på Øverste del av tabellen*/
.tdtopp1{
  text-align: right;
  vertical-align: top;
  background-color: #eceef0;
  height: 59px;
}
/* Denne classen plasserer text med farge på øverste td tagg*/
.ptagg{
    text-align: left;
    color: #789eca;
    vertical-align: top;
}

/* denne classen setter en grå border over logo.jpg*/
.tdlogo{
    border-top: 5px solid #aeafb2;
}

/* denne classen lager design på nederste td*/
.tdbunn{
    background-color: #aeafb2;
    color: #ffffff;
    text-align: center;
    font-size: 8pt;
    padding: 5px;
}
/*-----------------------------------------------Slutt Tabell oppsett-----------------------------------------*/

```

”Width: 989px;” bestemmer at bredden på tabellen skal være akkurat 989 piksler bred. En piksel er et lite punkt på skjermen, som sammen med alle de andre utgjør hele skjermbildet. Et skjermbilde kan ha forskjellige oppløsninger. Hvis man spesifiserer bredden på denne måten vil den være konstant uansett hvordan oppløsning brukeren av nettsiden har. Hvis man har en oppløsning på 1600 x 1200 piksler, vil dette da si at skjermbildet er 1600 piksler bredt og 1200 piksler høyt. Dette medfører at tabellen vil dekke 989 piksler av dette.

Bildet over setter regler for de forskjellige delene av en tabell. En tabell er satt sammen av 3 tagger. ”<table>”, ”<td>” og ”<tr>”. Grunnen til at vi har skilt utseende av disse i flere klasser er fordi vi ønsker forskjellig utseende og regler for de forskjellige delene av tabellen. Td står for ”table definition” og påvirker en kolonne mens tr, ”table row” påvirker en linje i tabellen. Slik kan vi da få tabellen til å være plassert og utformet grafisk slik vi ønsker i de forskjellige delene av tabellen.

På bildet under har vi klippet ut noen kodelinjer fra stylesheet.css som påvirker menyen til nettsiden. Siden vi ønsket et annet utseende på linkene som man manøvrerer nettsiden med, har vi laget egne klasser for disse. Klassene under påvirker forskjellige deler av en liste. En liste er laget med to typer tagger i html. Dette er ”<ul>” og ”<li>”, der ul er hele listen, mens li er hver nye linje i listen.

I klassen ”ul#menyliste li a” bestemmer vi utseende til selve linken i hver linje i hver liste. En link lages med taggen” <a>”. Derfor legger vi til ”a” etter navnet på klassen. ”Margin” og ”padding” bestemmer plassering til linken. ”Text-transform: capitalize” gjør at det alltid er stor bokstav på den første bokstaven i hvert ord i linken. ”Text-decoration: none” betyr at det ikke skal være kursiv, bold eller understreket link.

Neste klasse er ”ul#menyliste li a:hover” som er utseende til linken når man holder musen over linken. Her har vi lagt til ”Text-decoration: underline” som gir understrek på teksten når man holder over denne.

```
/*-----------------------------------------------------------------------------------------------------------------------
Her kommer alt for menyen. Dette innebærer design av knapper på meny linjen:
-----------------------------------------------------------------------------------------------------------------------*/

/* denne klassen bestemmel design på ul taggen*/
ul#menyliste {
list-style-type: none;
padding: 0; margin: 0;
line-height: 20px;
background-color: #aeafb2;
}

/* denne setter desin på ul sin li tagg*/
ul#menyliste li {
padding: 0; margin: 0;
display: inline;
padding: 0;
}

/* denne setter design på li taggen sine linker*/
ul#menyliste li a {
margin: 0.5em;
padding: 0em;
text-transform: capitalize;
font-family: "georgia", fantasy, serif;
font-size: 16px;
font-weight: normal;
color: #ffffff;
text-decoration: none;
}

/* denne fremhever linken når muse pekeren er over linken*/
ul#menyliste li a:hover {
text-decoration: underline;
}

/*-------------------------------------------------Slutt Meny-----------------------------------------------------------*/
```

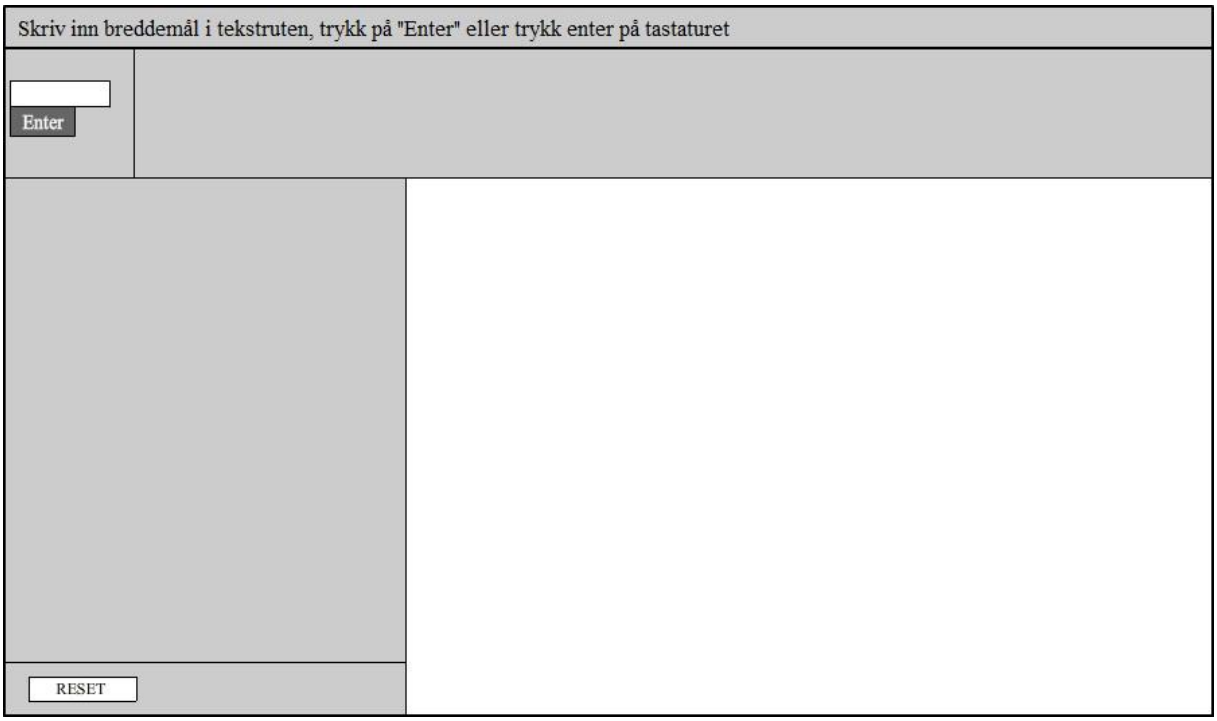
Beregningsmodellen

Presentasjon av beregningsmodellen

Bedriften vi jobbet for hadde et behov for å forenkle arbeidsprosessen for sine selgere. Beregningsmodellen og webshopen vår har til hensikt å gjøre akkurat dette. I tillegg til å lette arbeidsmengden til de ansatte var det også et ønske om å lage et hjelpemiddel for arkitektene som er i kontakt med bedriften. Arkitektene kan bruke løsningen til å plassere ønskede postkasser inn på det området som det er satt av til og deretter får de et bilde som de kan bruke i deres arbeidsprosess. Målet vårt var å skape et solid produkt som oppfylte bedriftens ønsker og som kunne brukes etter prosjektets ferdigstilling.

Beregningsmodellen er en modul som regner ut et område ut ifra en brukers breddemål. Brukeren kan velge mellom ulike typer postkasser og størrelser og dra de inn på det området som er blitt regnet ut. Dette er hovedfunksjonaliteten til modellen.

I utgangspunktet er utseende til beregningsmodellen ganske enkel. Det er få objekter å forholde seg til i utgangspunktet og klare inndelinger av selve rammen. Dette er for å skape en god oversikt og forenkle prosessen for brukeren. Det eneste brukeren trenger å forholde seg til er tekstboksen hvor breddemål skal skrives inn, ”Reset” knappen i bunnen av modellen som tilbakestiller alt til originale innstillinger og hjelpeteksten øverst som forklarer hva brukeren kan foreta seg i den delen av prosessen hvor brukeren befinner seg. Det brukeren må utføre for å kunne gå til neste steg er å skrive inn breddemål og trykke på ”Enter” knappen i modellen eller på ”enter” på tastaturet.



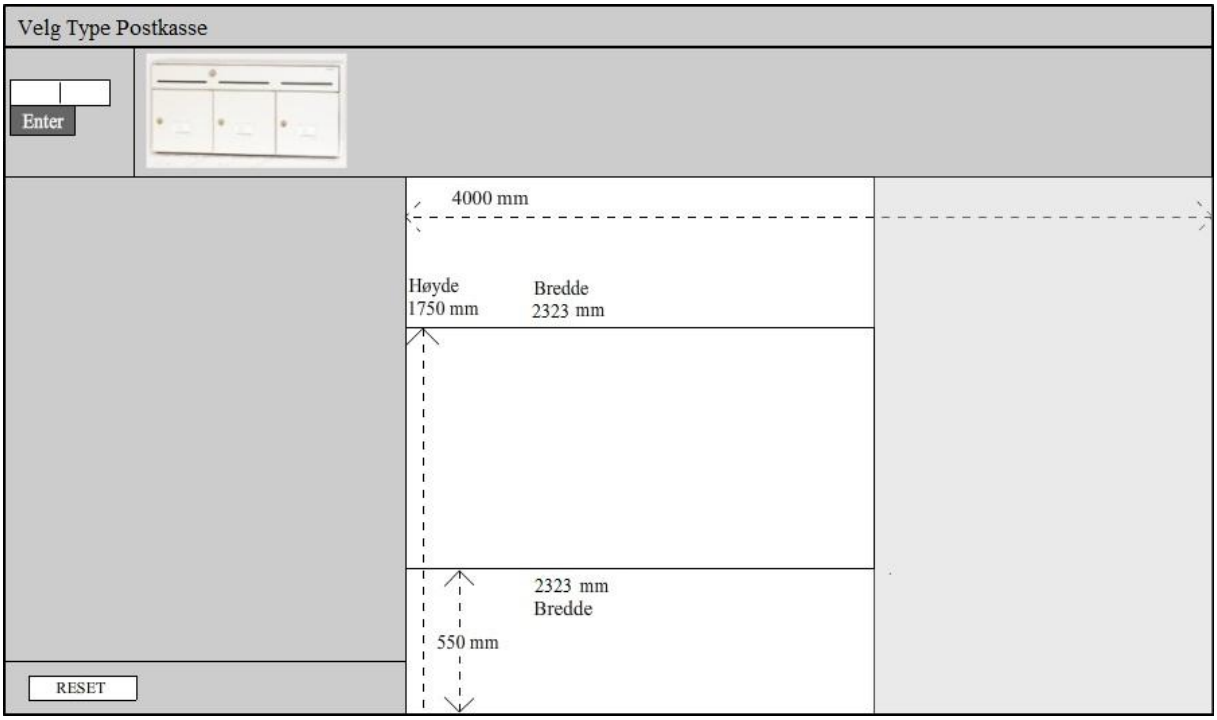
Utseende til beregningsmodellen slik den er i utgangspunktet.

Etter at brukeren har tastet inn breddemål kommer neste steg i prosessen. Her har modellen regnet ut det området som brukeren har til rådighet for å plassere postkasser. Området som ikke kan brukes skraveres vekk og solide streker viser rammen som brukeren må forholde seg til. Tillegg ser man den høyden som er lovregulert i forhold til plassering av postkasser. Loven sier at bunnen av kassene ikke kan være lavere enn 550 mm fra gulvet og toppen av kassene ikke kan være plassert høyere enn 1750 mm fra gulvhøyde.

Modellen viser også bredden på hele rammen, som er 4000 mm. Grunnen til at 4000 mm er maks bredde som brukeren kan velge er at det er ekstremt sjeldent at det er behov for mer. I følge de ansatte på Stansefabrikken AS har det fortsatt ikke skjedd at en kunde ønsker en bredde over 4000 mm.

Over den rammen som viser det området som brukeren kan dra postkasser inn på, er det plassert en annen ramme som viser de ulike postkassetypene som brukeren kan velge imellom. De ulike typene er illustrert med bilder og fungerer som en knapp når man trykker på dem med musepekeren. I eksempelet som er avbildet nedenfor er det bare bilde av en type postkasse. Grunnen til dette er at under hele utviklingsprosessen så fikk vi alt til fungere først med en type postkasse og deretter la vi til flere typer. Bildet er tatt under den delen av prosessen hvor vi fortsatt bare benyttet oss av en type.

Hjelpeteksten har forandret til ”Velg type postkasse”, som er det brukeren må utføre for å komme til neste steg i prosessen.

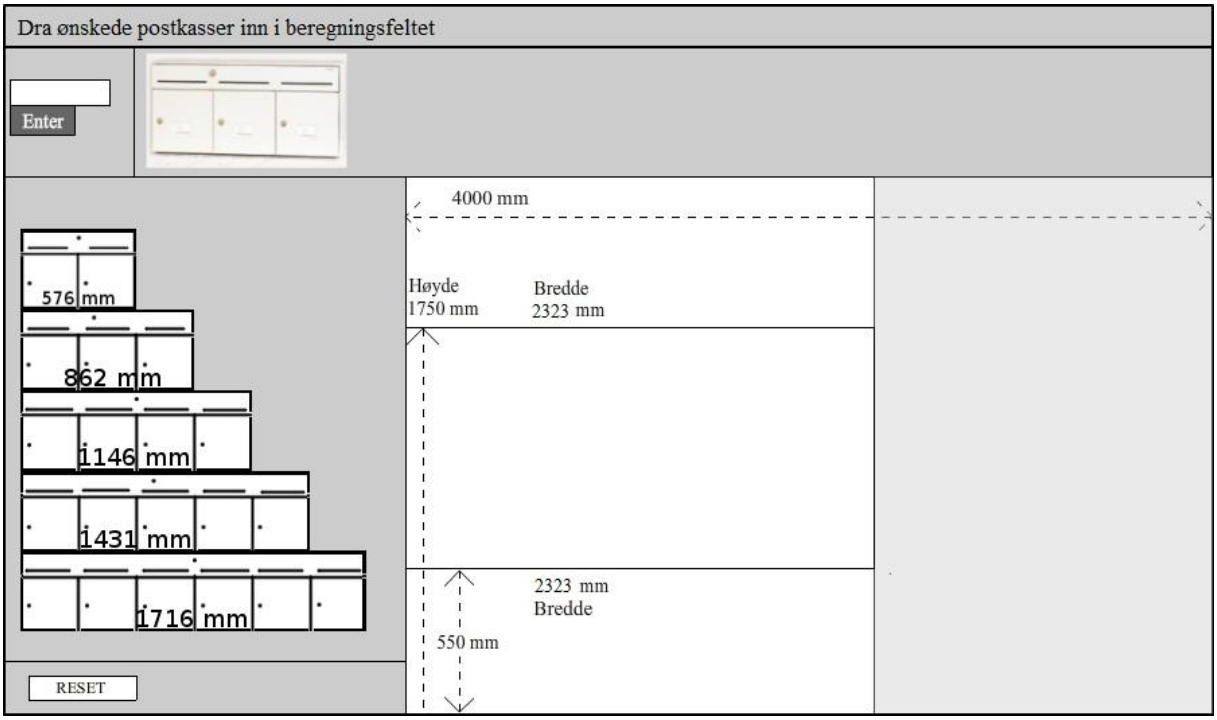


Slik ser modellen ut etter inntastet breddemål. I dette tilfellet er breddemålet på 2323 mm.

Alle de forskjellige typene av postkasser som Stansefabrikken AS selger kommer i forskjellige størrelser, det vil si forskjellige antall rom. Man kan velge postkasse type ”Robust” og de ulike størrelsene for denne typen er en, to, tre, fire, fem eller seks roms. For eksempel på en fem roms er det fem ulike rom i postkassen.

Etter at brukeren har valgt postkasse type, dukker det opp de størrelsesalternativene som det er mulig å velge imellom for den typen i en ramme til venstre. Disse er plassert slik at de rangert fra minst til størst. Alle postkassene er merket med breddemål i mm og er tegnet slik at arkitekter kan plassere dem i en plantegning.

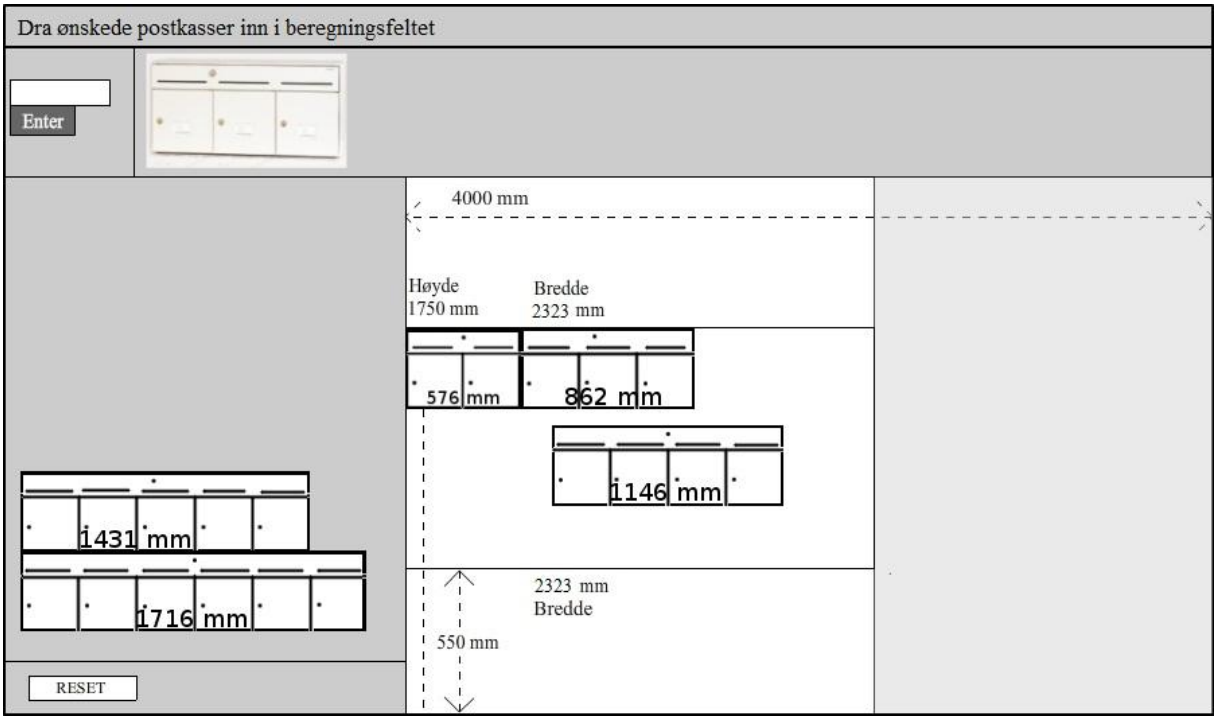
Hjelpeteksten er forandret til ”Dra ønskede postkasser inn i beregningsfeltet.” og er det som brukeren må gjennomføre i dette steget i prosessen. Brukeren kan da dra de ønskelige postkassene inn i feltet som er regnet ut ifra breddemålet som er tastet inn tidligere i prosessen.



Her har brukeren valgt type postkasse, og de ulike størrelsene dukker opp i feltet til venstre.

I bildet nedenfor har brukeren plassert postkasser i feltet ved å ta ”tak” i dem med musepekeren og dra dem inn i ønskelige posisjoner. Brukeren må holde seg innenfor det feltet som er regnet ut ifra det angitte breddemålet for at det skal være gyldig. I høyden er det akkurat plass til tre postkasser, dette er på grunn av lover og regler som er fastsatt, så det eneste som forandrer seg er bredden på feltet.

Etter at brukeren er fornøyd med plasseringen av postkassene og er ferdig med denne delen av prosessen velger brukeren å trykke på ”Neste”, som er en knapp som tar et bilde av det feltet som brukeren har plassert postkasser på og viser det. Brukeren kan deretter velge å lagre, skrive ut eller fjerne bildet.



Her har brukeren plassert postkasser i feltet.

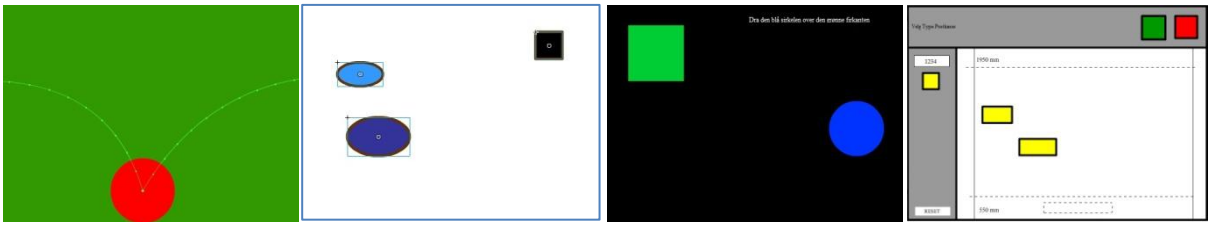
Alle disse stegene som er nevnt ovenfor utgjør hovedfunksjonaliteten til beregningsmodellen. Det er relativt få steg som brukeren må igjennom for å få det ferdige resultatet og modulen skal være oversiktlig og lett å forstå. Hensikten er å gjøre arbeidet lettere og avlaste arbeidsmengden for de ansatte i Stansefabrikken AS.

Fremgangsmåte for Flash

Flashløsningen vår er utarbeidet i Actionscript 3.0 i programmet Adobe Flash CS4 Professional. Funksjonene som forklares er funksjoner som tilbys av det programmet vi benytter. Beskrivelsene av Actionscript 3.0 er ikke avhengig av dette programmet og vil være det samme selv om man benytter et annet program for scripting. Noen visuelle objekter er laget i GIMP 2.6 og deretter importert inn i Adobe Flash, dette er fordi GIMP gir mye bedre muligheter for utvikling av visuelle objekter.

Fremgangsmåten for utvikling av ett flashprosjekt kan deles hovedsakelig i to områder. Vi valgte å benytte den løsningen som bygger på bruk av actionscript, som er programmeringsspråket for flash. Denne metoden krever at løsningen stort sett kodes fra bunnen av, uten store hjelpemidler. Den andre metoden å utvikle en løsning på er å benytte en funksjon og fremgangsmåte som heter ”timeline”. Denne funksjonen gir brukeren mulighet til å utvikle løsningen uten å benytte mye actionscript, men heller bruke visuelle hjelpemidler som tilbys av programmet. Vårt valg falt på bruk av actionscript siden dette gir mye større frihet for å kunne utvikle prosjektet akkurat slik vi ønsker og gir i tillegg mye mer til oss som studenter innen kompetanseheving og selvutvikling.

Ingen av gruppemedlemmene hadde noen erfaring fra flash eller actionscript før dette prosjektet, så for å bygge kompetanse begynte vi med å lese introduksjoner, veiledninger og diverse fremgangsmåter. På denne måten fikk vi mer kunnskap om hvordan flash og actionscript fungerte og hvilke muligheter vi kunne benytte for utvikling av løsningen vår. Illustrasjonene nedenfor er bilder av flashprosjekter tidlig i kompetansebyggingsfasen. Den første tegningen ble laget under timeline funksjonen og resultatet er en ball som spretter over skjermen. De to neste bildene er skrevet i actionscript og hensikten med disse var å få mer kunnskap om hvordan ”drag and drop” fungerer. ”Drag and drop” er en betegnelse for å kunne ta tak i et objekt med musepekeren og dra den over skjermen til et ønskelig punkt. Det siste bildet viser en litt mer avansert løsning som inkluderer ”drag and drop” og ulike løsninger for knapper og input tekst. Fremgangsmåten innebar at vi tok små steg av gangen for å få alle funksjonene vi ønsket og jobbet oss systematisk frem til den løsningen vi hadde visjoner om å utvikle.



Bilder av tidlige flashløsninger, hvor hensikten var å øke kompetanse innen flash og actionscript.

Det første man gjør når man setter i gang med en ny flashløsning er at man definerer og setter opp en ”stage”. Stagen er selve rammen hvor alle visuelle objekter og effekter fungerer på, altså som en bakgrunn for hele løsingen. Man definerer størrelse i piksler og ønskelig farge på stagen.

Etter at stagen er definert, oppretter man objekter og plasserer de på stagen. Objektene kan være alt fra bilder og tegninger til skrift, tekstbokser og andre objekter, og kan enten lages i ett annet program og deretter importeres inn i Adobe Flash, eller så kan man lage objektene direkte i Adobe Flash. Opprettede objekter tildeles ett ønskelig ”instance name” som er et unikt navn på objektet slik at man kan kalle på det fra actionscriptet og dermed benytte på objekter i funksjoner man har skrevet i actionscriptet. Det neste man gjør med objektene er å transformere dem til symboler slik at de kan interagere med stagen, andre objekter og actionscriptet. Man kan velge hva slags symboler man skal transformere til, og i vårt tilfelle benyttet vi oss av typen ”movieclips” for samtlige av våre objekter. Grunnen til at vi benyttet movieclips er at denne typen passet vårt bruksområde best siden vi trengte å skjule, flytte og endre nesten alle våre objekter, og hvis vi ville bruke et movieclip som tekstknapp definerte vi det bare i actionscriptet. Fremgangsmåten vår for objekter og funksjonalitet gikk ut på at vi ville først få til den funksjonen vi ønsket, så vi prøvde først med enkle tegninger og knyttet dem til funksjonene i actionscriptet, og så byttet vi de ut med mer detaljerte grafiske objekter når funksjonaliteten vi ønsket var oppnådd. Deretter la vi til nye objekter og knyttet de til andre funksjoner, og så videre. På denne måten fikk vi god oversikt over alle objektene og hvilke funksjoner fra actionscriptet som interagerte med hvilke objekter.

Forklaring av Actionscript

Det første som var nødvendig i scriptet var å hindre flashløsingen i og ”loope”. Loop betyr at animasjonen blir kjørt om igjen og om igjen uten stans. Siden denne løsningen skal tilby funksjoner som drag and drop må vi stoppe denne loopen, hvis ikke vil objektene som drag and drop har en innvirkning på forsvinne hver gang siden oppdaterer seg selv.

```
1   stop();
```

Denne kommandoen stopper loop.

Alle objektene som benyttes i flashprosjektet skal ha et visuelt utgangspunkt i forhold til stagen. Plasseringen av disse objektene bestemmes etter hvilke koordinater på x og y aksen av piksler man tilknytter objektene. Samtlige objekter ble tildelt bestemte koordinater.

```
12  DisplayObject(Robust1).x = 13;
13  DisplayObject(Robust1).y = 167;
```

Eksempel på hvordan man tilknytter koordinater til ett objekt.

Objektene vi brukte i prosjektet ble konvertert til symbol av typen ”movieclips”. Dette var fordi funksjonene vi ville at skulle interagere med objektene krevde denne typen. Når objekter skulle ha virkemåte som knapper, definerte vi dette i scriptet.

```
38  Robust_btn.buttonMode = true;
```

Her definerer vi et objekt som knapp.

Ikke all grafikk skal være synlig i utgangspunktet for brukeren. Ettersom brukeren interagerer med løsningen skal objekter som blir påvirket av brukerens input komme til syne. Alle objekter som ikke skal være synlige som utgangspunkt blir gjort usynlige.

```
45  DisplayObject(Robust1).visible = false;
46  DisplayObject(Robust2).visible = false;
47  DisplayObject(Robust3).visible = false;
```

Kode som gjør objekter usynlige for brukeren.

Siden brukeren skal kunne velge type postkasse og deretter størrelse etter at breddemålet er skrevet inn er det nødvendig med en funksjon som synliggjør de ulike postkassene og de ulike størrelsene. Funksjonen under oppfyller dette. Postkasseobjektene som er tilknyttet typen ”Robust” blir synliggjort og de andre typene blir holdt usynlige. I tillegg bestemmes koordinatene hvor objektene skal plasseres, samt en skrift med hjelpetekst synliggjøres. I funksjonen defineres det også om den skal virke inn på musen eller tastaturet. Dette gjøres i parentesen som inneholder ”event:MouseEvent”.

```
 95  function doitRobust(event:MouseEvent):void {
 96
 97      DisplayObject(Robust1).visible = true;
 98      DisplayObject(Robust2).visible = true;
 99      DisplayObject(Robust3).visible = true;
100      DisplayObject(Robust4).visible = true;
101      DisplayObject(Robust5).visible = true;
102
103
104      DisplayObject(Gul1).visible = false;
105      DisplayObject(Gul2).visible = false;
106      DisplayObject(Gul3).visible = false;
107
108      DisplayObject(Robust1).x = 13;
109      DisplayObject(Robust1).y = 167;
110
111      DisplayObject(Robust2).x = 13;
112      DisplayObject(Robust2).y = 227;
113
114      DisplayObject(Robust3).x = 13;
115      DisplayObject(Robust3).y = 287;
116
117      DisplayObject(Robust4).x = 13;
118      DisplayObject(Robust4).y = 347;
119
120      DisplayObject(Robust5).x = 13;
121      DisplayObject(Robust5).y = 406;
122
123      DisplayObject(overskriftPostkasse).visible = false;
124      DisplayObject(overskriftDra).visible = true;
125
126  }
```

Funksjon som er tilknyttet og interagerer med postkassetypen ”Robust”.

Objektene kan tildeles de ulike funksjonene som er skrevet i actionscriptet. Når brukeren kommuniserer med objektet så utføres den funksjonen som er knyttet til det bestemte objektet. Man nevner også om dette er en ”MouseEvent”. Det vil si om dette er en funksjon som har en virkemåte på museklikk. En knapp som brukeren trykker på med musen er et eksempel på en slik funksjon.

```
129  Red_btn.addEventListener(MouseEvent.CLICK, doit);
130  Robust_btn.addEventListener(MouseEvent.CLICK, doitRobust);
```

Kommandoer som knytter en funksjon til et objekt.

En sentral funksjon i beregningsmodellen er muligheten til å dra de ulike postkassene rundt med musepekeren. Denne ”drag and drop” funksjonen skapes ved å lage en ”startdrag” og en ”stopdrag” funksjon og knytte dem til objektet som skal kunne flyttes på. ”StartDrag” utføres når brukeren presser ned museknappen på objektet og ”stopDrag” utføres når brukeren slipper museknappen igjen.

```
189  function mouseDownHandler(evt:MouseEvent):void {
190      var obj = evt.target;
191      obj.startDrag();
192  }
193
194  function mouseUpHandler(evt:MouseEvent):void {
195      var obj = evt.target;
196          obj.stopDrag();
197  }
```

Funksjonene over skaper en ”drag and drop” effekt.

```
143  Robust1.addEventListener(MouseEvent.MOUSE_DOWN, mouseDownHandler);
144  Robust1.addEventListener(MouseEvent.MOUSE_UP, mouseUpHandler);
```

Kommandoer som knytter ”drag and drop” effektene til objektet.

Funksjon som gir brukeren muligheten til å velge å stille tilbake hele modellen til slik den var i utgangspunktet. Objektene blir usynliggjort og plassert til sine originale koordinater, samt hjelpeteksten forandres.

```
function dotry(event:MouseEvent):void {

    DisplayObject(Gul1).visible = false;
    DisplayObject(Gul2).visible = false;
    DisplayObject(Gul3).visible = false;

    DisplayObject(Robust1).visible = false;
    DisplayObject(Robust2).visible = false;
    DisplayObject(Robust3).visible = false;
    DisplayObject(Robust4).visible = false;
    DisplayObject(Robust5).visible = false;


    DisplayObject(Gul1).x = 36;
    DisplayObject(Gul1).y = 167;

    DisplayObject(Gul2).x = 21;
    DisplayObject(Gul2).y = 234;

    DisplayObject(Gul3).x = 13;
    DisplayObject(Gul3).y = 293;

    DisplayObject(Robust1).x = 13;
    DisplayObject(Robust1).y = 167;

    DisplayObject(Robust2).x = 13;
    DisplayObject(Robust2).y = 227;

    DisplayObject(Robust3).x = 13;
    DisplayObject(Robust3).y = 287;

    DisplayObject(Robust4).x = 13;
    DisplayObject(Robust4).y = 347;

    DisplayObject(Robust5).x = 13;
    DisplayObject(Robust5).y = 406;

    visbreddetext.text="";
    visbreddetext2.text="";
    breddetext.text="";

    DisplayObject(infoNede).visible = false;
    DisplayObject(infoOppe).visible = false;
    DisplayObject(Red_btn).visible = false;
    DisplayObject(Robust_btn).visible = false;
    DisplayObject(overskriftDra).visible = false;
    DisplayObject(overskriftBredde).visible = true;
    DisplayObject(mmOppe).visible = false;
    DisplayObject(mmNede).visible = false;
    DisplayObject(overskriftPostkasse).visible = false;
    DisplayObject(helBreddeText).visible = false;
    DisplayObject(skrav).visible = false;
    DisplayObject(skravTo).visible = false;
}
```

Funksjon hvor resultatet er en "reset" effekt, hvor beregningsmodellen tilbakestilles.

```
258  reset.addEventListener(MouseEvent.CLICK, dotry);
```
Kommandoer som knytter "reset" funksjonen til et objekt.

Det er visse regler som må defineres for tekstfeltet hvor brukeren skal skrive inn breddemålet. Det skal bare være mulig å taste inn tall, det skal ikke være mulig å taste inn mer enn fire siffer og maks størrelse på tallet skal være 4000. Dette er på grunn av at brukeren skal ha maks bredde på 4000. I tillegg fjerner funksjonen det tallet som er skrevet inn i tekstboksen etter at brukeren han trykket på "Enter" eller på "enter" knappen på tastaturet.

```
266  breddetext.restrict = "0-9.";
267
268  //setter maks verdi til 4000
269  function maks(e:Event):void {
270
271      if (Number(breddetext.text) > 4000) {
272          breddetext.text = "";
273      }
274  }
275  breddetext.addEventListener(Event.CHANGE, maks);
```
Funksjon og kommandoer som setter restriksjoner og betingelser på tekstboksen.

En viktig funksjon i beregningsmodellen er den som skal kunne regne ut det området hvor brukeren skal kunne plassere ønskelige postkasser ut ifra brukerens breddemål. Funksjonen som er avbildet er en ”KeyboardEvent” som betyr at den interagerer med tastaturet. I funksjonen begynner man med å definere hvilken knapp på tastaturet som funksjonen skal reagere på, i dette tilfellet er det ”Enter”. For å tegne selve rammen hvor man skal plassere postkassene benytter man ”graphics” funksjonene i flash. Først fjerner man all tidligere ”graphics” som eventuelt har blitt tegnet på den gitte variabelen. Dette er for ikke å lage flere lag av grafikk. Funksjonen henter informasjonen som er skrevet inn i tekstboksen, definerer det som et tall og deretter lagrer det i en variabel som heter ”bredde”. Hele rammen er i utgangspunktet på 600 piksler og selve utregningen for å fastslå området på rammen begynner (4000 / 10)*1.5, hvor resultatet blir antall piksler i bredden og lagrer seg i en variabel. Deretter kommer det en utregning som tar utgangspunkt i ”input” teksten til brukeren, altså hva brukeren skriver inn i

```
282 function keyDownHandler(event:KeyboardEvent):void {
283
284     if (event.keyCode == Keyboard.ENTER) {
285
286         skrav.graphics.clear();
287         DisplayObject(skrav).visible = true;
288         var bredde:Number = Number(breddetext.text);
289         var a:Number = (4000/10)*1.5;
290         var b:Number = (bredde /10)*1.5;
291         var bredde2:Number = a - b;
292
293         skrav.graphics.beginFill(0xD3D3D3, skrav.alpha =0.7);
294
295         skrav.graphics.lineStyle(1,0,1);
296
297         skrav.graphics.drawRect(899, 128, -bredde2, 399);
298
299         skrav.graphics.endFill();
300
301         addChild(skrav);
302
303         skravTo.graphics.clear();
304         DisplayObject(skravTo).visible = true;
305
306         skravTo.graphics.lineStyle(1,0,1);
307         var c:Number = a - bredde2;
308         //var c:Number = a - bredde2;
309         skravTo.graphics.drawRect(299, 240, c, 179);
310
311         skravTo.graphics.endFill();
312
313         addChild(skravTo);
314
315         visbreddetext.text = breddetext.text;
316         visbreddetext2.text = breddetext.text;
317         breddetext.text="";
318         DisplayObject(Red_btn).visible = true;
319         DisplayObject(Robust_btn).visible = true;
320         DisplayObject(infoNede).visible = true;
321         DisplayObject(infoOppe).visible = true;
322         DisplayObject(overskriftBredde).visible = false;
323         DisplayObject(overskriftDra).visible = false;
324         DisplayObject(overskriftPostkasse).visible = true;
325         DisplayObject(mmOppe).visible = true;
326         DisplayObject(mmNede).visible = true;
327         DisplayObject(helBreddeText).visible = true;
328     }
329
330 }
```

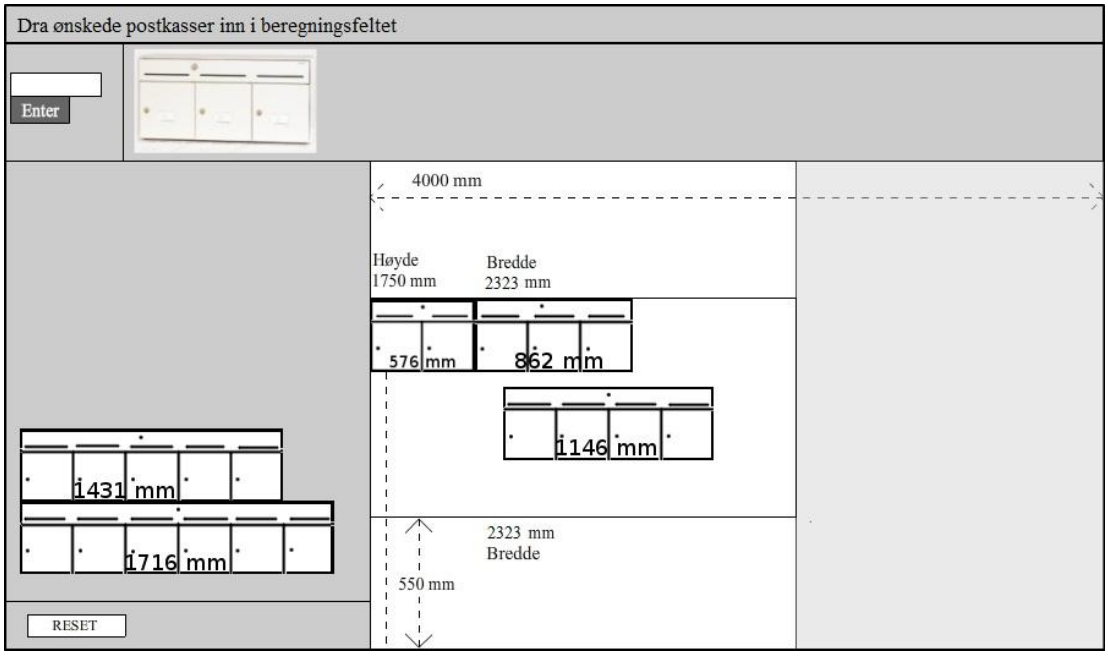
tekstboksen. Utregningen (bredde / 10)*1.5 gir et resultat som er bredden på veggen i forhold til modellen. Så må vi finne antall piksler som skal trekkes fra de 600 pikslene som rammen har i utgangspunktet. Dette gjøres ved å trekke fra svaret fra den første utregningen med svaret fra den andre utregningen. Da får man det området som skal skraveres i modellen.

Etter utregningen kommer den delen som skal tegne selve skraveringen. ”Graphics” er en innebygd funksjon i Adobe Flash og gir muligheten til å tegne på løsingen. Først definerer man farge og gjennomsiktighet som skal være på det som skal tegnes, man legger til synlighet og tykkelse på streken som er omrisset av området og så angir man koordinatene hvor objektet skal tegnes. I dette tilfellet har bredde2, som er en variabel som inneholder verdien av utregningen for området, fått en minus foran seg. Dette er fordi rektangelet som er resultatet av de angitte grafikk kommandoene skal tegne strekene bakover fra angitte koordinater og ikke fremover.

Videre så lages det et nytt rektangel. Her gjelder samme fremgangsmåter og noen av de samme betingelsene som på skraveringsområdet, men koordinatene er forskjellige og dette rektangelet skal ikke ha noen farge. Her har variabelen bredde2 ikke minus foran seg siden rektangelet skal tegnes ut ifra koordinatene og fremover. Resultatet blir et rektangel som går på linje med selve rammen i den delen av beregningsmodellen og med skraveringen og dermed sitter man igjen med det området som brukeren kan dra postkasser inn på. De siste kommandoene i funksjonen setter de ulike objektene til synlig eller usynlig ettersom om de har en oppgave i denne delen av prosessen.

Denne funksjonen er beregnet på en ”KeyboardEvent”, men det er også nødvendig å samme funksjon til en ”MouseEvent” hvis brukeren ønsker å trykke på en knapp i beregningsmodellen med musen istedenfor å trykke på ”Enter” på tastaturet. Innholdet i denne funksjonen er derfor også plassert i en annen funksjon som er definert som en ”MouseEvent” som igjen er tilknyttet en ”Enter” knapp i modellen.

Illustrasjonen under er et bilde av beregningsmodellen. Her ser man hvordan løsningen fungerer og hva slags resultater koden som er beskrevet ovenfor gir.



Kilder

Use Case: http://home.online.no/~moestboe/uml_oversikt.htm

Php : http://no.wikipedia.org/wiki/PHP

CSS: http://no.wikipedia.org/wiki/Cascading_Style_Sheets

Mysql: http://en.wikipedia.org/wiki/Mysql

Flash: http://www.2tp.no/webdesign/flash/index.htm
http://www.adobe.com/products/player_census/flashplayer/

Ordbok for prosjektet

PHP	PHP: ”Hypertext Preprocessor”. Programmeringsspråk. Brukes til å utvikle nettsider.
Flash	Betegnelse som refererer til programvaren Adobe Flash Player. Gir animasjon på nettsider.
Javascript	Skriptspråk. Tilfører dynamiske elementer til nettsider.
GIMP	”GNU Image Manipulation Program”. Blir brukt til å behandle og lage illustrasjoner, digital grafikk og fotografier.
Webshop	Uttrykk for handel på nett.
Netthandel	Norsk utrykk for handel på nett.
”Drag and Drop”	Betyr at brukeren kan interagere med eventuelle elementer ved å ta tak i ett element og plassere det på ønskelig sted.
Database	Samling av data.
mySQL	Versjon av SQL (Structured Query Language). Relasjonsdatabase for behandling av data.
CSS	”Cascading Style Sheets”. Brukes til å definere utseende på filer skrevet i HTML eller XML.
HTML	”HyperText Markup Language”. Markeringsspråk for formatering av nettsider med informasjon som skal vises i en nettleser.

Grensesnitt	Betegnelse for hvordan en bruker kommuniserer med ett system.