

# Nettside, Webshop og Beregningsmodell

---

Hovedprosjekt våren 2009



Peder Sundbø ■ Magnus Eriksen ■ Øyvind Schjelderupsen

**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

## HOVEDPROSJEKT

HOVEDPROSJEKTETS TITTEL <b>Nettside, Webshop og beregningsmodell</b>	DATO 29.05.09
	ANTALL SIDER/BILAG 71 sider/8 bilag over 56 sider
PROSJEKTDeltakere Peder Sundbø, s141795 Magnus Eriksen, s141765 Øyvind Schjelderupsen, s141758	INTERN VEILEDER Torunn Gjester
OPPDRAAGSGIVER Stansefabrikken Products AS – <a href="http://www.stansefabrikken.com/index.php?cid=4148">http://www.stansefabrikken.com/index.php?cid=4148</a>	KONTAKTPERSON Trond Eriksen - Salgssjef

<p>SAMMENDRAG</p> <p>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.</p> <p>Systemet vi har utviklet består av en webshop og en beregningsmodell. Denne rapporten tar for seg utviklingsprosessen og teknologier som ble benyttet for å ferdigstille produktet.</p> <p>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.</p> <p>Det ferdige produktet er overlatt til Stansefabrikken, og de står fritt til å bruke systemet som det er eller videreutvikle det til eget bruk.</p> <p>God fornøyelse.</p>
--

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. Det ferdige produktet er tilgjengelig på <http://www.lugusu.com/stansefabrikken>.

Vi ønsker å takke alle de involverte i vårt prosjekt. Takk til alle ansatte ved Stansefabrikken Products AS for godt samarbeid og til vår veileder Torunn Gjester ved Høgskolen i Oslo for god hjelp, tilbakemeldinger og veiledning under hele prosessen. Takk til John Kåre Schjelderupsen ved Borge AS for opptrykkingen av rapporten.

## Innholdsfortegnelse

Studieprogram: Anvendt Datateknologi .....	2
Forord .....	3
Presentasjon .....	6
Mål Resultatmål.....	7
Effektmål.....	7
Kravspesifikasjonen.....	8
Systembeskrivelse .....	8
Systemet skal inneholde:.....	9
Rammekrav i systemet.....	9
Valg av metode .....	10
Metode .....	10
PROSJEKTPLANLEGGING .....	12
Fremdriftsplan .....	12
Aktivitetskart .....	12
Risikoanalyse.....	12
PROTOTYPING .....	13
Beregningsmodell.....	13
Tanker og Planlegging av beregningsmodellen i papirprototypen .....	14
Webshop .....	15
Etter utvikling av papirprototype .....	17
Designprototype .....	18
Brukertestning av designprototypen .....	18
ANALYSE .....	19
Mål med analysefasen .....	19
Use Case .....	19
Vår Use Case.....	21
Use Case beskrivelser: .....	22
ER-diagram.....	23
Strukturkart .....	24
Dataflytdiagram .....	25
Hvorfor dataflytdiagram?.....	25
Hva er et dataflytdiagram?.....	26
DESIGN .....	27

Mål med designfasen .....	27
Valg av verktøy .....	27
Litt om verktøyene .....	28
Adobe Flash .....	31
Formater .....	32
Fordeler .....	33
Ulemper.....	33
IMPLEMENTASJON .....	34
Mål med implementasjonsfasen .....	34
Webshop.....	34
Presentasjon av Webshop .....	34
Administratorsiden .....	37
PHP .....	39
Forsiden.....	39
Bestill postkasser .....	40
Sikkerhet i PHP.....	43
CSS implementasjon.....	46
Stilark.css .....	46
Beregningsmodellen Presentasjon av beregningsmodellen.....	50
Fremgangsmåte for Flash .....	55
Forklaring av Actionscript.....	57
SAMMENDRAG .....	65
KONKLUSJON .....	66
KILDER .....	67
ORDBOK .....	68

## Presentasjon

Hovedprosjektet har blitt gjennomført ved Høgskolen i Oslo, avdelingen for ingeniørutdanning. Målet med oppgaven var 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 har vi laget 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.

### Effekt mål

Stansefabrikken Products AS 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, og er et datterselskap av Stafa Industrier AS. Virksomheten er salg av postkassesystemer til boligblokker og enkeltstående hus 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 Products 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 vil gi mer automatikk i denne prosessen. Systemet vil kunne minske arbeidsmengden på selgere av postkasser fordi kunder kan henvises til beregningsmodellen når det gjelder plassering av postkasser. Ytterligere 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 stansefabrikken sin 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 å utarbeide en god kravspesifikasjon slik at kunden og utvikler har satt seg klare mål for prosjektet.

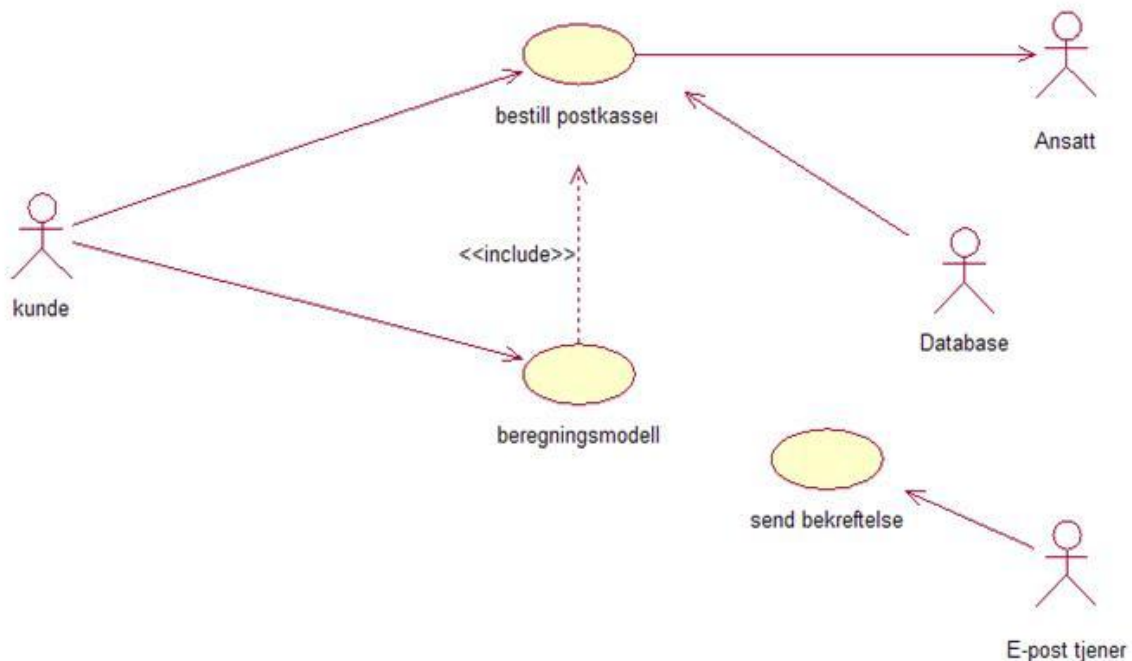
Denne kravspesifikasjonen ble utarbeidet av gruppen sammen med oppdragsgiver, hvor oppdragsgiver måtte godkjenne kravspesifikasjonen før videre arbeid ble fastslått.

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 rammebetingelser er beskrevet i dette dokumentet, som er en instruks for hvordan systemet skal fungere.

Den fullstendige kravspesifikasjonen er vedlagt som vedlegg (vedlegg1).

## 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.
- 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.

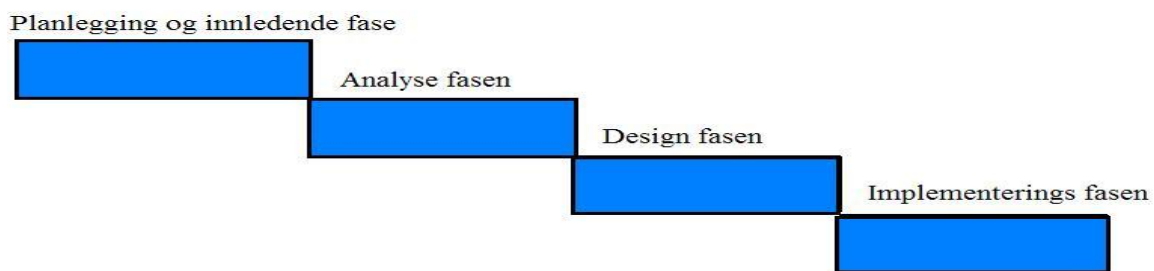
## Valg av metode

### Metode

Fossefallsmetoden er en metode innenfor systemutvikling. Fossefallsmetoden går ut på at man deler opp hele prosjektet i faser. Disse fasene utføres en etter en i den rekkefølgen fasene er satt opp. Når man bruker denne formen for systemutvikling må en fase gjøres ferdig før man begynner på den neste.

Vårt fossefallsdiagram:

### Fossefallsmetoden



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 4 faser.

### *Planlegging og innledende fase*

I denne fasen legger vi mest vekt på å planlegge systemet som skal utvikles. Vi arbeider med å utvikle en kravspesifikasjon for systemet hvor systemets krav blir fastslått. I denne perioden valgte vi å lage en papirprototype for å bestemme hovedtrekkene ved design på 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 benyttet vi ER-diagram for planlegging av databasen. Vi så nærmere på og planla strukturen i systemet og dens relasjoner.

### *Design fasen*

I denne delen av utviklingen til systemet lager vi design prototype og planlegger design av den endelige siden på et dypere nivå. I tillegg fastslår vi valg av verktøy og hjelpemidler til utviklingen. Her jobber vi også 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 planleggingen i de tidligere fasene.

## **PROSJEKTPLANLEGGING**

### **Fremdriftsplan**

Fremdriftsplan er en oversiktlig plan over fremdriften i prosjektet. Fremdriftsplanen brukes til 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.

(Vedlegg 2)

### **Aktivitetskart**

Vi valgte å lage en litt mer detaljert plan for arbeidet vårt i dette prosjektet ut over den allerede utviklede fremdriftsplanen. Dette dokumentet kalles aktivitetskart, der vi puttet inn alle punkter som skulle gjøres underveis på et mer detaljert nivå enn fremdriftsplanen. I denne planen la vi til informasjon om når ting skulle være ferdig, hvem som hadde ansvaret for punktet, hvorfor punktet skulle utføres og hvor lang tid det tok og fullføre det. Dette dokumentet var en ganske enkel toukers plan med oversiktlig blikk på hva som måtte gjøres.

(Vedlegg 3)

### **Risikoanalyse**

Risikoanalyse brukes til å kartlegge problemer som kan oppstå under et prosjekt eller i en prosess. Dette brukes til det formål 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 4)

## PROTOTYPING

### Papirprototype

Når man lager en papirprototype setter man seg ned med kravene til systemet og lager enkle strektegninger av det. Viktige elementer i en papirprototype er å få frem litt funksjonalitet slik at man kan få innblikk i 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 få et utgangspunkt for designet av brukergrensesnittet.

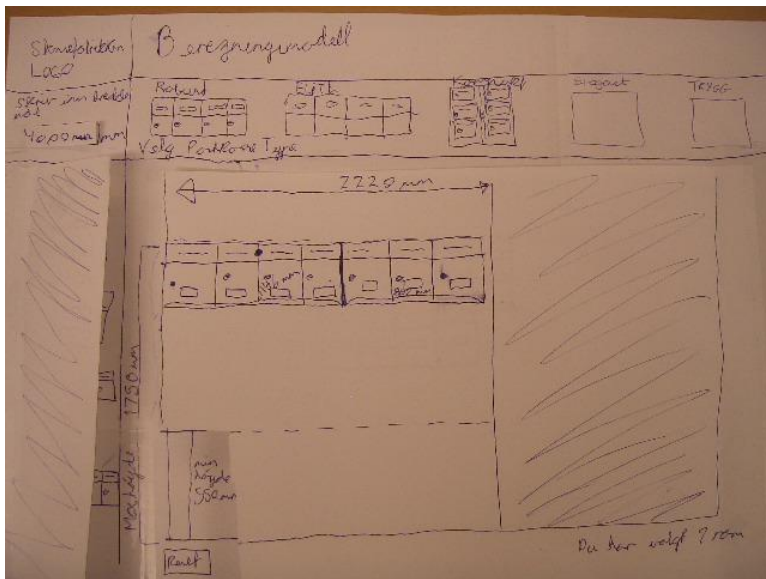
Vi valgte å lage en papirprototype av systemet for å planlegge enkel design og hvordan systemet skal se ut i grove trekk, samt få en enkel grov skisse over systemet som vi kan bruke til planlegging med våre arbeidsgivere. Dette vil sammen med kravspesifikasjonen vår gi oss et godt utgangspunkt til å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

Siden systemet vårt er delt i to deler (beregningsmodell og webshop) har vi valgt å lage disse hver for seg. Det vil komme en sammenslåing av disse i implementasjonsfasen så de blir ett system, ikke to enkeltstående systemer.

### Beregningsmodell

Vi startet med å tegne en enkel layout til en webside lik den Stansefabrikken Products AS har i dag. Neste steg var å lage selve beregningsmodellen. Først fant vi ut av hvilken layout den skulle ha, før vi valgte plasseringen av de mulighetene den skulle ha. Da vi var ved dette punktet måtte vi undersøke hvilke utviklingsverktøy som kunne klare å lage den funksjonaliteten vi ønsket. Etter at vi hadde gjort research på nettet etter teknologi som kunne gi støtte for det vi ønsket å lage, kom vi frem til at Flash var et verktøy som ville egne seg til en slik beregningsmodell. Det neste vi gjorde var å lage en enkel funksjonalitet i papirprototypen vår som viste hvordan den skulle reagere i et gitt scenario.

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



### Tanker og Planlegging av beregningsmodellen i papirprototypen

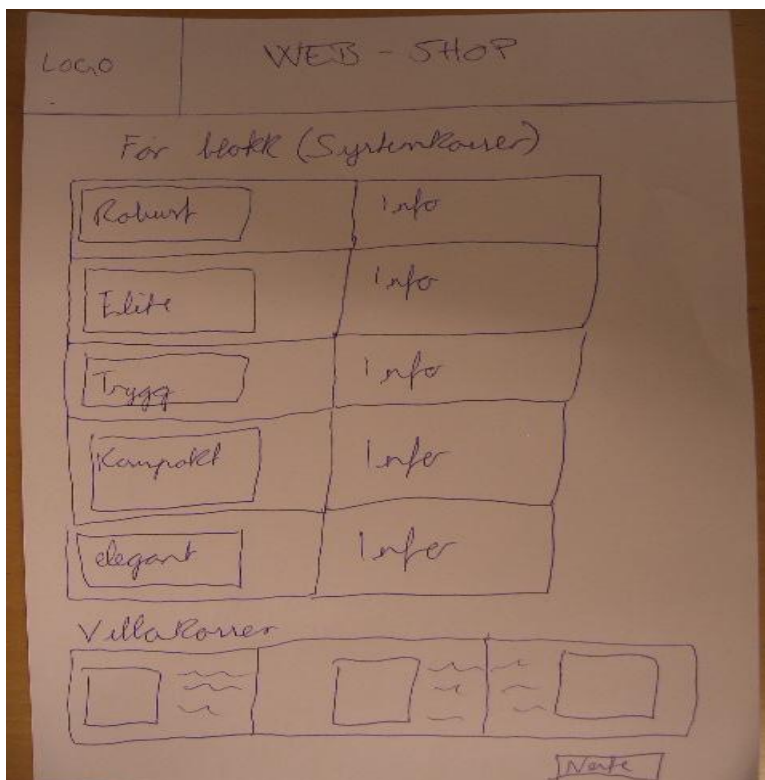
Vi valgte å la produktene til Stansefabrikken Products AS (postkasser) stå på rekke øverst på nettsiden, siden vi ønsker å fremheve produktene kunden kan velge. Når kunden da velger en type postkasse etter ønske, vil han/hun få frem et nytt vindu med alle størrelser av den utvalgte postkassetypen. Her 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 høyre for postkassene. Disse vil ha høydemål som 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.

## Webshop

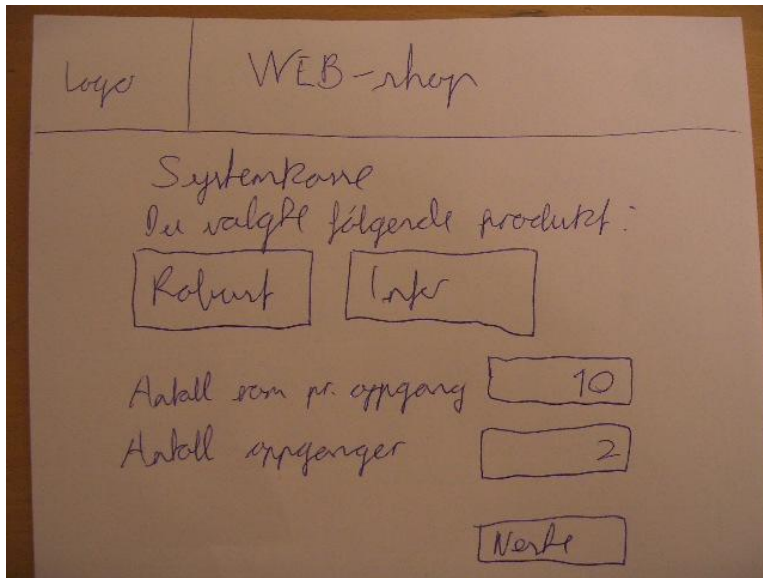
Ved utviklingen av papirprototypen for webshop delen startet vi med en enkel layout for nettsiden, den samme som i beregningsmodellen. Layouten for nettsiden er lik uansett hvor på nettsidene man befinner seg. Dette er for å skape kontinuitet.

Neste steg 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 under.

Her valgte vi å sette produktene inn i en tabell siden dette syntes å være den mest oversiktlige måten og plassere de på. Informasjon om postkasser vil da få en god grafisk plassering i forhold til bildet 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, i motsetning til villakasser som er ment for ett hus eller postkassestativ og trenger da mindre planlegging.



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

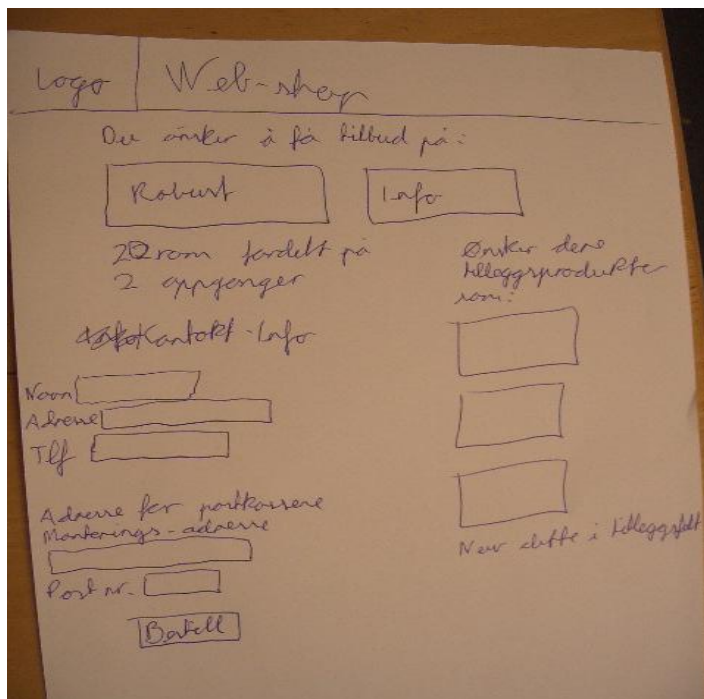


Her kan kunden lese litt om produktet de har valgt. De får nå mulighet til å velge hvor mange kasser de skal ha av produktet.

Vi har valgt å ha flere sider for å gjøre det mer oversiktlig for kunden underveis. Kunder kan fort ha en tendens til å droppe siden hvis personen møter ”a wall of text”. Med dette mener vi at dersom det blir for mye informasjon på hver side mister kunden helt fokus og dette kan medføre at han eller hun kanskje dropper å gjennomføre bestillingen.



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 skal 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 til kundebehandler og en bekreftelse til kunden.



### Etter utvikling av papirprototype

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

## 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 designprototype av den delen av systemet vi følte trengte dette. Vi laget en designprototype av beregningsmodellen som vi testet på noen med studenter. Vi testet beregningsmodellen på fire medstudenter fordi de ville ha 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.



## ANALYSE

### Mål med analysefasen

Målet med denne fasen er å finne ut hvordan prosjektet skal gjennomføres og bygges opp.

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

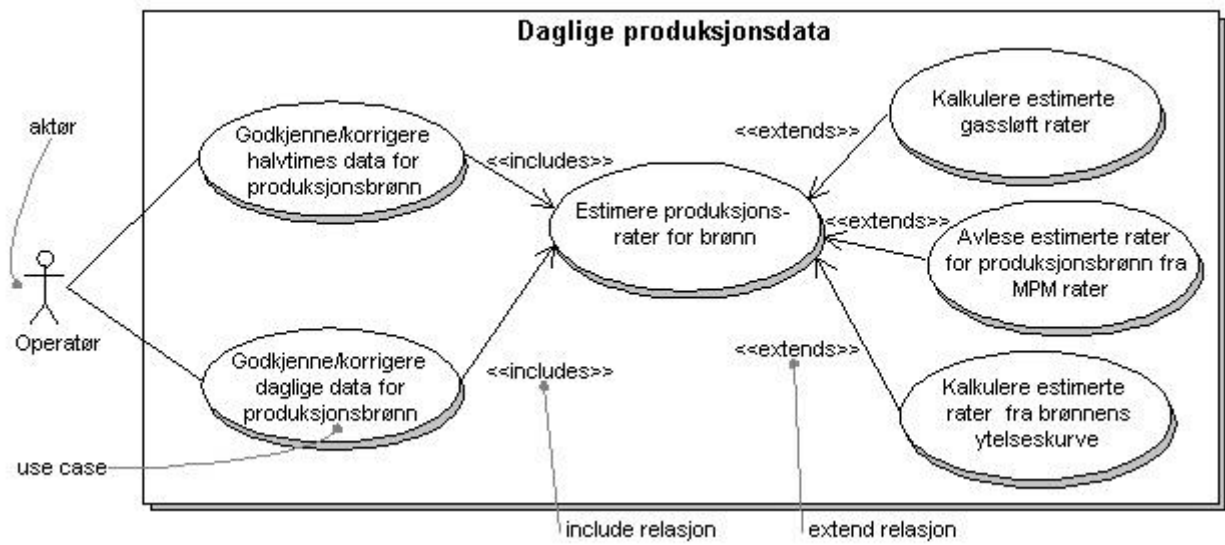
Use Case diagrammer brukes til å illustrere bruken og oppførselen av systemet. Use Case diagrammer forteller hva systemet skal gjøre, men ikke hvordan det skal gjøres.

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:

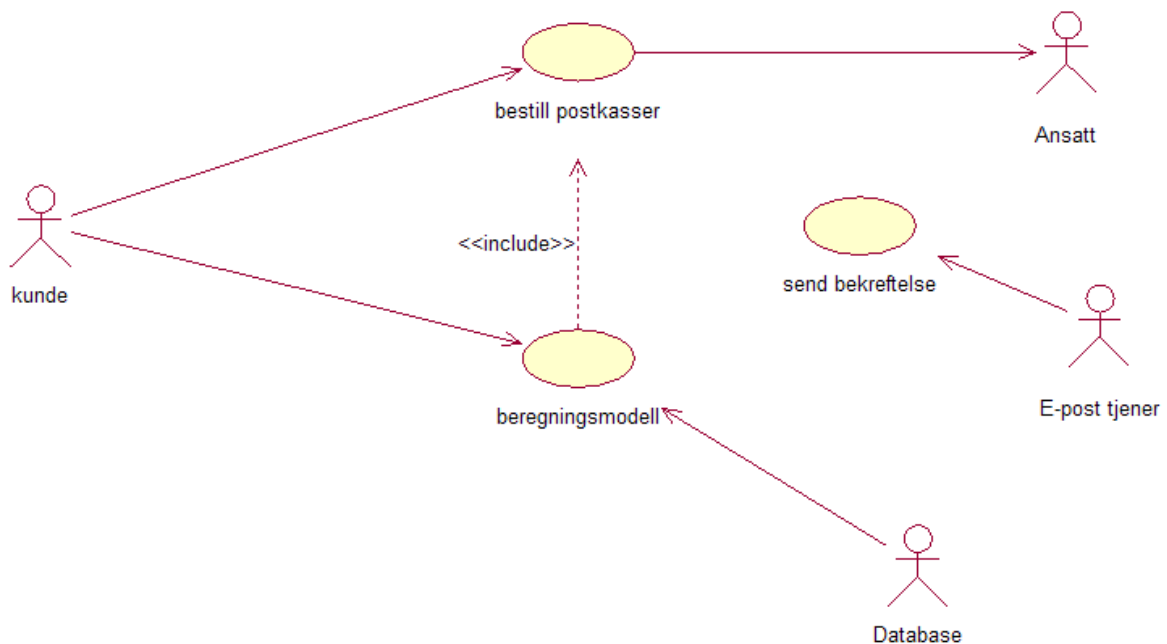
- **Include:** En include relasjon mellom Use Case A og Use Case B, hvor pilen peker fra A til B, betyr at A arver oppførselen til B ved en spesifisert lokasjon. Dvs. Use Case A vil alltid inkludere (utføre) Use Case B.
- **Extend:** En extend relasjon fra Use Case A til Use Case B, hvor pilen peker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



Bildet over viser systemets oppbygning. Der pinnemennene er aktører i systemet. Disse er med på å utføre prosessene. Sirklene 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 retning input eller output av data går.

Use Case beskrivelsen under er en 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 - hvem som utfører selve prosessen
3. Trigger - hva som må til for at prosessen starter.
4. Prebetingelse - betingelse for at denne prosessen kan gjennomføres.
5. Postbetingelse - resultatet av prosessen blir gjennomført.
6. Normal hendelsesflyt - hvordan prosessen skal fungere ved normal hendelsesflyt.
7. Variasjoner - hva som kan skjer i prosessen som ikke er etter normal hendelsesflyt.

## Use Case beskrivelser:

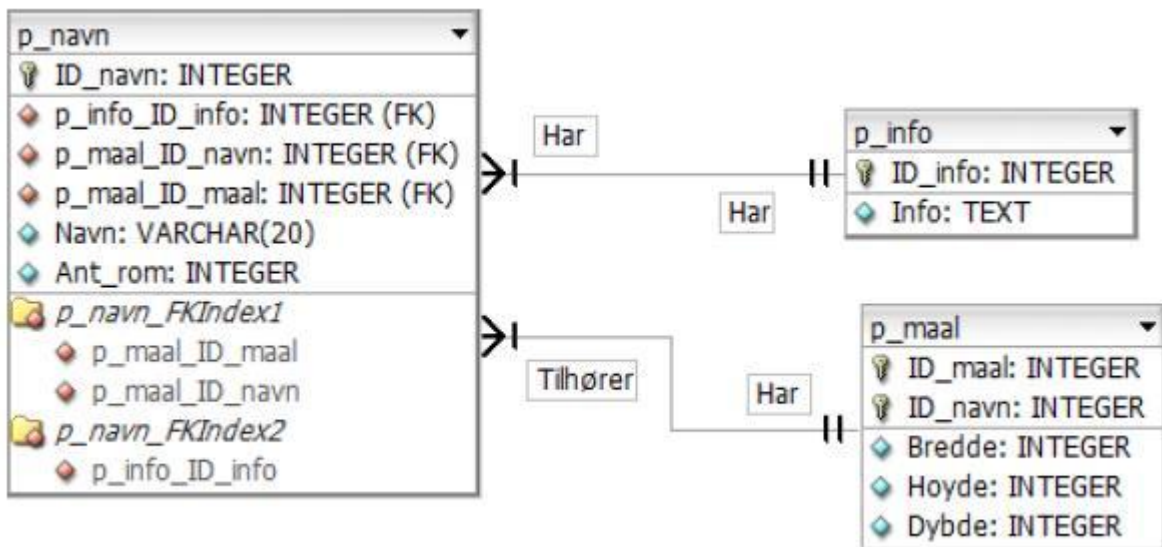
<b>Use Case</b>	Beregningsmodell
<b>Aktør</b>	Kunde
<b>Trigger</b>	Kunde ønsker beregning av plass til postkasser
<b>Prebetingelse</b>	Kunden ønsker å bruke beregningssystem
<b>Postbetingelse</b>	Kunden får en grafisk beregningsmodell av oppgangen med postkasser.
<b>Normal hendelsesflyt</b>	<ol style="list-style-type: none"><li>1. Systemet spør om breddemål i oppgangen.</li><li>2. Kunden skriver inn breddemål og bekrefter det.</li><li>3. Systemet viser postkassetyper.</li><li>4. Kunden velger en postkassetype.</li><li>5. Systemet viser alle størrelser av valgt postkassetype og setter av et område for plassering av kasser. Den informerer om ”drag and drop” funksjonen.</li><li>6. Kunden drar ønskede postkasser over på veggen og godkjenner.</li><li>7. Systemet viser tegning av veggen og gir mulighet for utskrift av tegning. Den gir også mulighet for å gå direkte til bestill postkasser.</li></ol>
<b>Variasjoner</b>	6a1. Kunden drar en kasse som ikke passer i forhold til breddemål. Systemet vil informere om at dette ikke er mulig.

Use Case beskrivelser for de andre prosessene er vedlagt som vedlegg (vedlegg 5).

## 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*

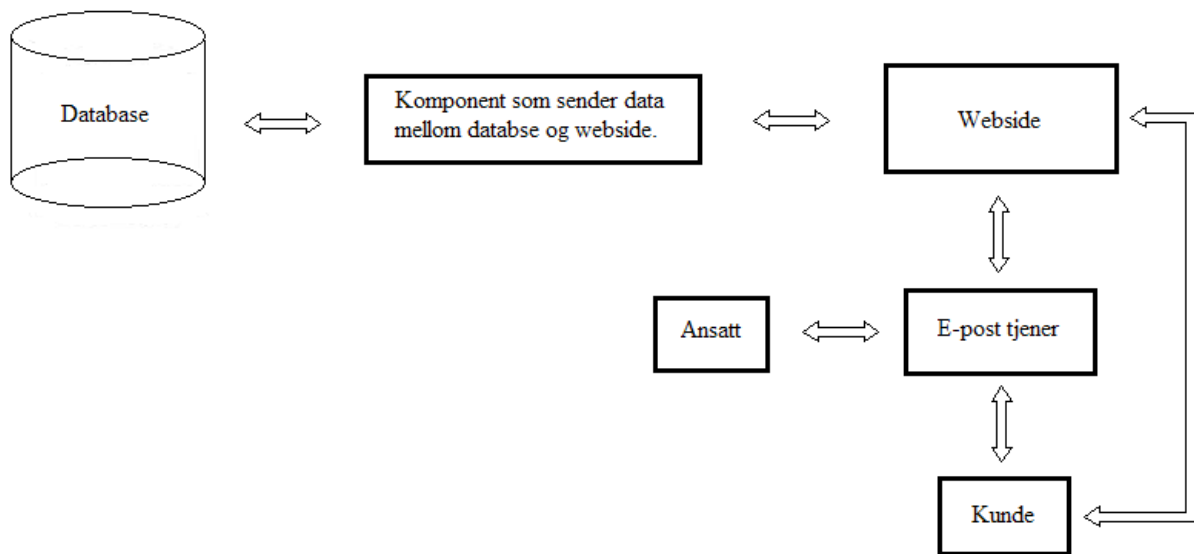
Vårt endelige ER-diagram er forholdsvis lite, da vi kun har to tabeller. Disse er heller ikke knyttet sammen. Tabellen ”p\_navn” inneholder alt av postkasseinfo, som brukes av webshopen. Den andre tabellen, ”admin”, inneholder brukernavn og passord til administrator for innlogging på websiden. Her kan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



*Endelig versjon av ER-modellen*

## Strukturkart

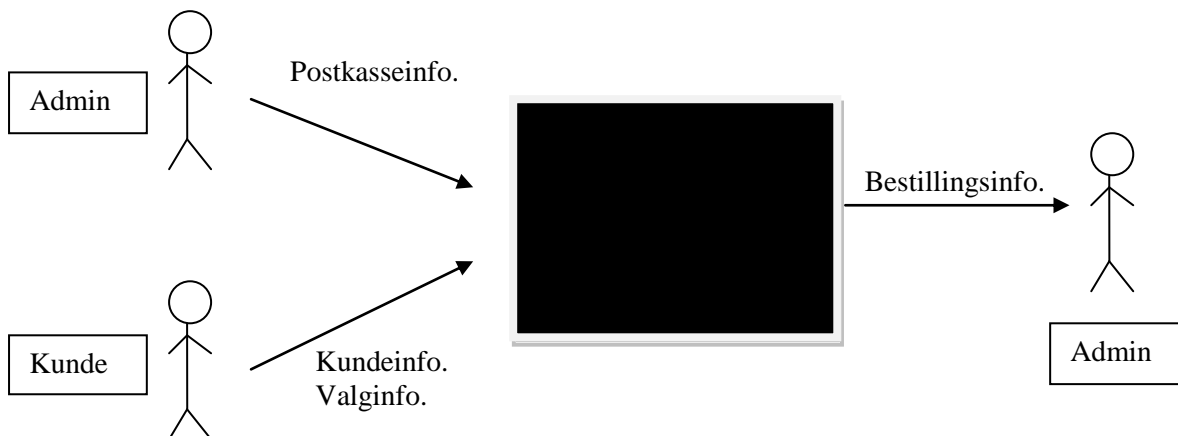
Et strukturkart er en illustrasjon som viser hvordan systemet henger sammen. Boksene i illustrasjonen simulerer aktører og komponenter. Pilene illustrerer i hvilken retning informasjonsflyten går og mellom hvem/hva. Vi har brukt dette fordi det er et veldig fint og oversiktlig verktøy som gir en klar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



*Illustrasjon over strukturen i systemet.*



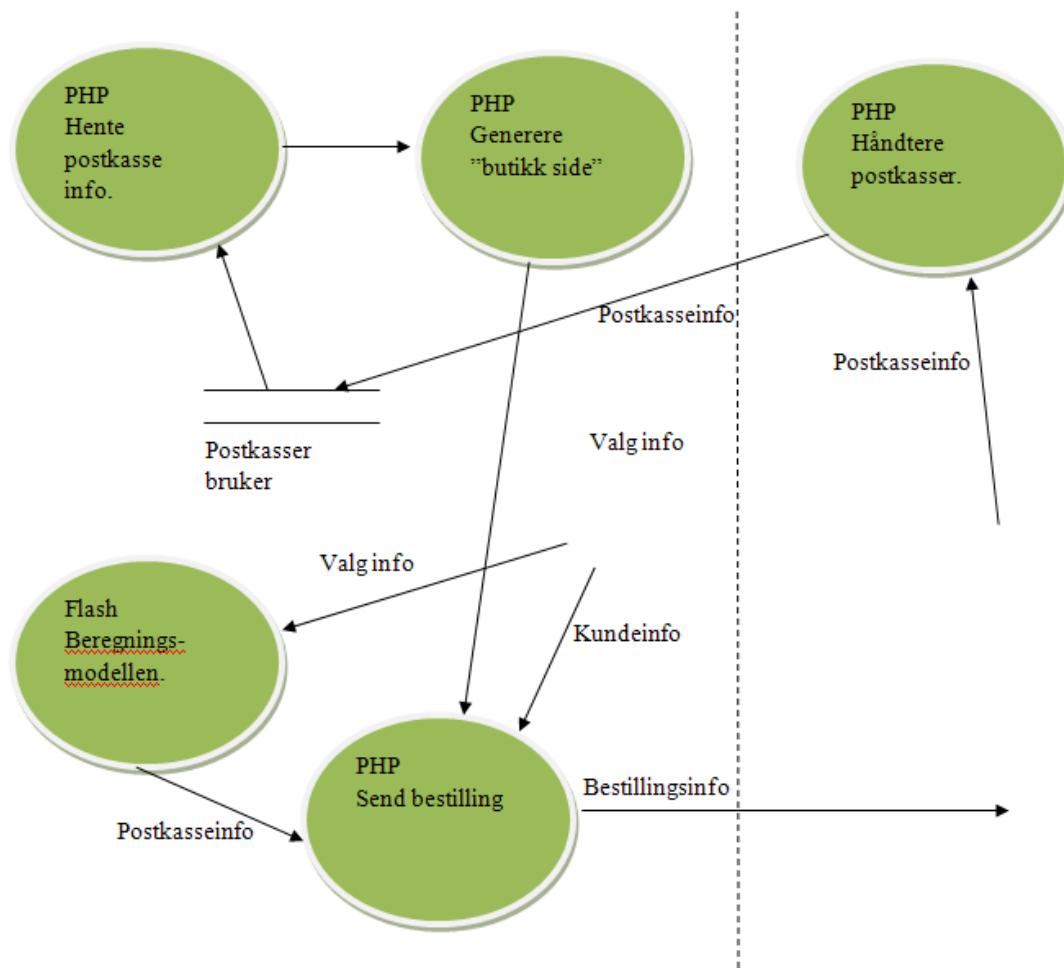
## Dataflytdiagram



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 et dataflytdiagram?

Dataflytdiagrammet viser sammenhengen mellom forskjellige prosesser, datalagring og ekstern entitet. Dens oppgave er å gi et oversiktlig bilde 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 lagt inn i datalageret.
- En ekstern entitet er personen eller aktøren som er utenfor systemet og er den som starter de forskjellige prosessene.

## DESIGN

### Mål med designfasen

Målet med denne fasen er å fastslå hvilken teknologi og hvilke verktøy som skal benyttes for å utarbeide løsningen, samt hvilke utfordringer teknologiene kan gi.

I denne fasen utviklet vi også designet på løsningene og utarbeidet designprototypen. Dette er dokumentert under kapittelet om prototyping.

### Valg av verktøy

Vår samarbeidspartner 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 er kjent med. Når det gjelder design på siden har vi valgt å bruke CSS, 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 sammenhengighet planlegges.

## Litt om verktøyene

### **HTML**

HTML ("HyperText Markup Language") er et programmeringsspråk for det å lage nettsider. I et slikt 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 og 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 har kjennskap til dens muligheter innen nettside bygging. Likevel trenger vi også å lære mer om sikkerhet i PHP, siden vårt prosjekt krever noe mer sikkerhet enn hva vi har laget tidligere.

### **CSS**

CSS ("Cascading Style Sheets") brukes til å forme design og utseende til en nettside. Nettsidene er skrevet i HTML. Vi bruker CSS for nettopp dette formål, det å kunne forme sidene slik vi ønsker de skal være for kunden.

### **GIMP**

Gimp er et program for manipulering og behandling av digital grafikk og fotografier.

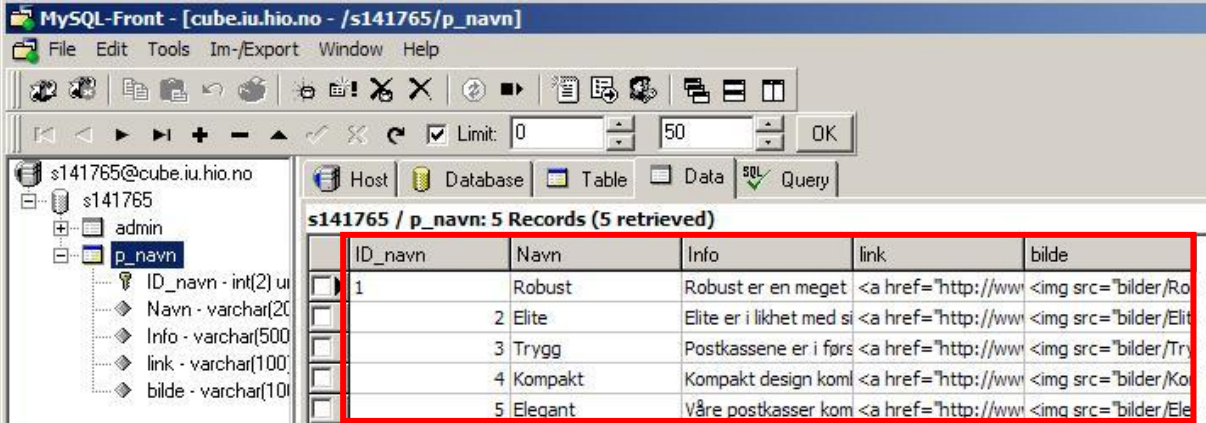
Vi bruker GIMP til å produsere bilder for design av nettsiden. Programmet er også brukt til å lage bilder til beregningsmodellen i prosjektet, dette mest fordi bildene som brukes i flash animasjonen skal stemme overens med virkelige mål. Bildene må da lages i presise størrelser.

## 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.

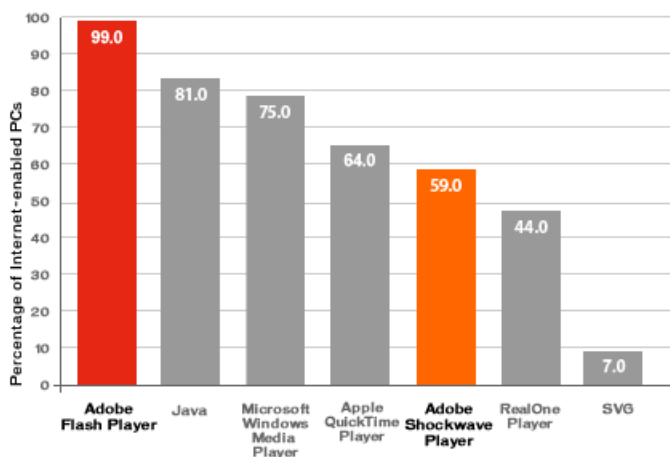
Skjermbildet under er tatt fra MySQL-Front, som er et program for å administrere MySQL databaser. Her ser du at de forskjellige postkassemodellene er lagt inn med ett unikt id-nummer, navn, litt info om postkassetypen, link til mer info og til slutt et bilde av den.



The screenshot shows the MySQL-Front interface. The title bar reads "MySQL-Front - [cube.iu.hio.no - /s141765/p\_navn]". The interface includes a menu bar (File, Edit, Tools, Im-/Export, Window, Help), a toolbar, and a navigation pane on the left showing the database structure for "s141765" with a table named "p\_navn". The main window displays a table with 5 records, highlighted with a red border. The table has columns: ID\_navn, Navn, Info, link, and bilde.

ID_navn	Navn	Info	link	bilde				
1	Robust	Robust er en meget	<a href="http://www	1</sup>	99.1%	99.0%	98.6%	55.9%
US/Canada	99.1%	99.1%	98.9%	54.5%				
Europe <sup>2</sup>	99.1%	98.9%	98.2%	56.5%				
Japan	99.0%	98.8%	98.3%	59.3%				
Emerging Markets <sup>3</sup>	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 nettleseren, vil brukeren kunne se din flashanimasjon. For de brukerne som har plug-in vil sidene se helt like ut, uavhengig av maskin og nettleser. 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 piksler. 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 pikselert 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

## Mål med implementasjonsfasen

Denne fasen består av implementeringen av alt vi utarbeidet i analyse og designfasen. Målet her var å få til å lage systemet slik at det fungerte optimalt i forhold til rammekravene.

## 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 postkassestype han eller hun ønsker å bestille. Nederst på siden har man en "videre" knapp som sender kunden til neste side for bestilling.



### Produkter

Velg postkasse å trykk på knappen bestill for å gå videre i bestillingen:

Navn	Info	Link	Bilde
<input type="radio"/> 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	<a href="#">Les mer</a>	
<input type="radio"/> 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.	<a href="#">Les mer</a>	
<input type="radio"/> 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.	<a href="#">Les mer</a>	

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

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

	Postkasser Stansfabrikken Products AS	<a href="#">Admin</a>
		
Forside   Beregningsmodell   Bestilling Av Postkasser		
<h2>Du ønsker og bestille:</h2>		
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	<a href="#">Les mer</a>
Skriv inn antall postkasserom du ønsker: <input type="text"/>		
Skriv inn antall oppganger henvendelsen gjelder: <input type="text"/>		
<input type="button" value="Neste"/>		
Copyright © STAFA Industrier AS   Privacy Policy   All rights reserved   Member of STAFA INDUSTRIER		

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.

	<a href="#">Postkasser</a> <a href="#">Stansefabrikken Products AS</a>	<a href="#">Admin</a>
 <h2 style="margin: 0;">POSTKASSER</h2> <p style="margin: 0;">mulighet til å velge</p>		
<a href="#">Forside</a> <a href="#">Beregningsmodell</a> <a href="#">Bestilling Av Postkasser</a>		

## Du ønsker å bestille:

Robust	<p>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</p>	<a href="#">Les mer</a>	
--------	---	-------------------------	---

Din henvendelse gjelder 10 rom  
Din henvendelse gjelder 10 oppganger

## Dine personalia

Navn: <input type="text"/>
Adresse: <input type="text"/>
E-post adresse: <input type="text"/>
Tlf: <input type="text"/>
Monteringsadresse:
Adresse: <input type="text"/>
Postnr.: <input type="text"/>
Poststed: <input type="text"/>
Tilbehør eller andre ønsker:
<small>Ønsker du andre farger eller tilleggsutstyr nevnt det her...</small>
<input type="text"/>
<input type="button" value="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.

## Administratorsiden

I tillegg til bestillingssystemet har vi utviklet en administratorside. 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 høyre 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.

The screenshot shows the administrators interface for STANSEfabrikken. At the top left is the logo and name 'STANSEfabrikken'. To its right is the breadcrumb 'Postkasser Stansefabrikken Products AS'. In the top right corner, there is a red-bordered box containing the text 'Admin'. Below this is a green banner with the title 'POSTKASSER' and the subtitle 'mulighet til å velge'. A navigation bar below the banner contains links for 'Forside', 'Beregningsmodell', and 'Bestilling Av Postkasser'. The main content area features a background image of a building and mailboxes. It contains two sections: 'Beregningsmodell' with a description of a calculation tool, and 'Bestill postkasser' with a description of the ordering process. At the bottom, there is a footer with copyright information: 'Copyright © STAFA Industrier AS | Privacy Policy | All rights reserved | Member of STAFA INDUSTRIER'.

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

**Admin**

Brukernavn

Passord




Her blir man 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 innlogget:

Navn:	Info	Link	Bilde
<input type="text" value="navn"/>	<input type="text" value="Ønsker du andre farger eller tilleggsutstyr nevnt det her..."/>	<input type="text" value="link"/>	<input type="text" value="Elegan2"/>
<input type="button" value="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	<a href="#">Les mer</a>		<a href="#">slett</a>
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.	<a href="#">Les mer</a>		<a href="#">slett</a>
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.	<a href="#">Les mer</a>		<a href="#">slett</a>

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:

```
<a href="index.php?pg=bestill">Bestilling Av postkasser</a>
```

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 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 seg 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.

```
1 <?php  
2 $tilkobling = mysql_connect( 'localhost', 's141765', '' ) or die( "problemer med tilkobling" );  
3 mysql_select_db( 's141765' , $tilkobling ) or die( "Kunne ikke velge database" );  
4 $rs = mysql_query( "SELECT * FROM p_navn" );  
5 ?>
```

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  
) 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 å 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 form (som vist under) vil dette si at formet får id form 2. Måten formet sender informasjon på er "method=post". Hadde det stått GET her ville den sendt informasjonen med headeren. "action =" spesifiserer hvor formet ender etter at den blir fullført.

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

```
</form>
```

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

hvis variabelens navn er postkasser:

```
$v = $_REQUEST["postkasser"];
```

eller

```
$v = $_POST["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 å 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. Det vil også bli sendt en bestilling til selgeren av produktet. Dette gjør vi ved kommandoene som vist i bildet under:

```

1 <?php
2
3
4 $type = $_REQUEST["postkassetype"];
5 $soppganger = $_REQUEST["antaloppganger"];
6 $from = $_REQUEST["antallrom"];
7 $navn = $_REQUEST["navn"];
8 $navn_bor = $_REQUEST["adresse1"];
9 $sepost = $_REQUEST["epost"];
10 $tlf = $_REQUEST["tlf"];
11 $montering = $_REQUEST["adresse3"];
12 $postnr = $_REQUEST["post"];
13 $sted = $_REQUEST["poststed"];
14 $tillegg = $_REQUEST["tillegg"];
15
16 $recipient = 'earwin_sletten@hotmail.com';
17 $subject = "Bestilling av $type";
18 $from = "$navn, epost: $sepost, tlf: $tlf, adresse: $adresse";
19 $msg = "Melding fra: $from\n\n antallrom: $rom\n antaloppganger: $soppganger\n monteringsadresse: $montering\n postnummer: $postnr\n Poststed: $sted\n
20 Tilleggsinformasjon: $tillegg \n";
21 mail($recipient, $subject, $msg);
22
23
24 $tilbakemelding = "$sepost";
25 $spoeng = "Du ønsker å bestille $type";
26 $fra = "Stansefabrikken products AS, epost: earwin_sletten@hotmail.com ";
27 $melding = "Melding fra: $fra\n\n Du ønsker å bestille postkassetype: $type \n antallrom: $rom\n antaloppganger: $soppganger\n\n Tusen takk for din
28 bestilling, vår selger vil kontakte deg via epost eller tlf. \n \n MVH \n Stansefabrikken Products AS";
29 mail($tilbakemelding, $spoeng, $melding);
30
31
32 echo "<hi>$spoeng</hi>\n Kvittering på ønsket bestilling er sendt til din mail adresse. <br></br> Du vil bli kontaktet av en kundekonsulent for videre
33 behandling av bestillingen.<br></br> MVH <br></br> Stansefabrikken products AS"
34 ?>

```

Her legges bestill info til variabler som blir brukt til å sende e-mail

Sender e-mail til selger

Sender bekreftelse til kunde

Gir tilbakemelding på nettsiden til kunden.

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 e-poster med innholdet av disse variablene.

Mailen sender vi med kommandoen:

Mail(”mottaker”, ”emne”, ”innhold”);

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

## Sikkerhet i PHP

Vi har tidligere sett litt på 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 sikkerhetsaspekter 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 som kan utnyttes hvis 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 kodeslutt 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, som da kan gi fra seg informasjon om innholdet i tabellene eller kjøre kode som skader eller endrer innholdet. Dette er ikke en ønsket situasjon. For å 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 å 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 hexadesimaler. 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 hashen vi legger i databasen mot den vi lager av brukerens input.

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

```
10 if(isset($_POST['password']))
11 {
12 $pass = mysql_real_escape_string(md5($_POST['password'].$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 opp mot hashen 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 kjøres, lages en sessioncookie hvor variabelen "log" er lik "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.

```
1 //mysql_num_rows returnerer antall rader, eller false når den ikke finner noe.
2 if(mysql_num_rows( $rs))
3 {
4 //setcookie( 'admin', 1, time()+3600 );
5 $_SESSION['log'] = 'in';
6 header( 'location: innlogget.php' );
7 }
8 else
9 {
10 //setcookie( 'admin', 0, time()-3600 );
11 $_SESSION['log'] = 'out';
12 header( 'location: index.php' );
13 }
14 mysql_close($tilkobling);
15 ?>
```

For å unngå at en person bare går rett til siden `innlogget.php` hvor admin er innlogget tester vi bare om brukeren har sessioncookien med innholdet "in". Hvis brukeren 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 å 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.

```
$type = strip_tags ($REQUEST["postkassetype"]);  
$oppganger = strip_tags ($REQUEST["antalloppganger"]);  
$rom = strip_tags ($REQUEST["antallrom"]);  
$navn = strip_tags ($REQUEST["navn"]);  
$navn_bor = strip_tags ($REQUEST["adresse1"]);  
$epost = strip_tags ($REQUEST["epost"]);  
$tlf = strip_tags ($REQUEST["tlf"]);  
$montering = strip_tags ($REQUEST["adresse3"]);  
$postnr = strip_tags ($REQUEST["post"]);  
$sted = strip_tags ($REQUEST["poststed"]);  
$tillegg = strip_tags ($REQUEST["tillegg"]);
```

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

```
.tabellen { regler }
```

Fordi man med denne koden lager en egen klasse blir man nødt til å hente denne 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 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 svart over hele siden hvor noe annet ikke er bestemt. "Color : #000000" bestemmer denne fargen. #000000 refererer til hexadesimaler som spesifiserer en enkel fargekode. 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 tekstfargen.

```
1 /*-----  
2 klassen for hele siden utenom det som blir spesifisert lengre nede.  
3 -----*/  
4 body {font-family: "georgia", fantasy, serif;  
5         font-size: 10pt;  
6         font-style: normal;  
7         font-weight: normal;  
8         color:#000000;  
9 }  
10  
11 /*-----Slutt body-----*/  
12
```

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.

```
14 /*-----
15 Tabelloppsett og design av tabellen.
16 -----*/
17 /* Denne classen påvirker hele tabellen og bestemmer bredde og plassering*/
18 .hovedtable {
19 width: 989px;
20 margin: auto;
21 align: center;
22 border-collapse: collapse;
23 }
24
25 /* denne setter design på Øverste del av tabellen*/
26 .tdtopp1{
27 text-align: right;
28 vertical-align: top;
29 background-color: #ecef0;
30 height: 59px;
31 }
32 /* Denne classen plasserer text med farge på øverste td tagg*/
33 .ptagg{
34 text-align: left;
35 color: #789eca;
36 vertical-align: top;
37 }
38
39 /* denne classen setter en grå border over logo.jpg*/
40 .tdlogo{
41 border-top: 5px solid #aeafb2;
42 }
43
44 /* denne classen lager design på nederste td*/
45 .tdbunn{
46 background-color: #aeafb2;
47 color: #ffffff;
48 text-align: center;
49 font-size: 8pt;
50 padding: 5px;
51 }
52 /*-----Slutt Tabell oppsett-----*/
53
```

”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 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 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 seg rundt på 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 den. Her har vi lagt til "Text-decoration: underline" som gir understrek på teksten når man holder over denne.

```
55 /*-----
56 Her kommer alt for menyen. Dette innebærer design av knapper på meny linjen:
57 -----*/
58
59 /* denne klassen bestemmel design på ul taggen*/
60 ul#menyliste {
61 list-style-type: none;
62 padding: 0; margin: 0;
63 line-height: 20px;
64 background-color: #aeafb2;
65 }
66
67 /* denne setter desin på ul sin li tagg*/
68 ul#menyliste li {
69 padding: 0; margin: 0;
70 display: inline;
71 padding: 0;
72 }
73
74 /* denne setter design på li taggen sine linker*/
75 ul#menyliste li a {
76 margin: 0.5em;
77 padding: 0em;
78 text-transform: capitalize;
79 font-family: "georgia", fantasy, serif;
80 font-size: 16px;
81 font-weight: normal;
82 color: #ffffff;
83 text-decoration: none;
84 }
85
86 /* denne fremhever linken når muse pekeren er over linken*/
87 ul#menyliste li a:hover {
88 text-decoration: underline;
89 }
90
91 /*-----Slutt Meny-----*/
```

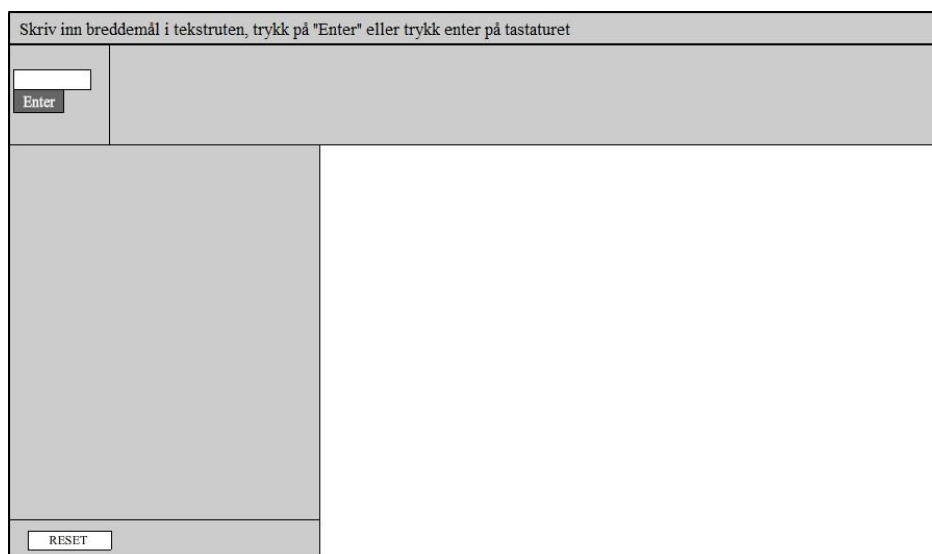
## Beregningsmodellen

### 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.



Skriv inn breddemål i tekstruten, trykk på "Enter" eller trykk enter på tastaturet

<input type="text"/>	
Enter	
RESET	

*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. I 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.

Velg Type Postkasse

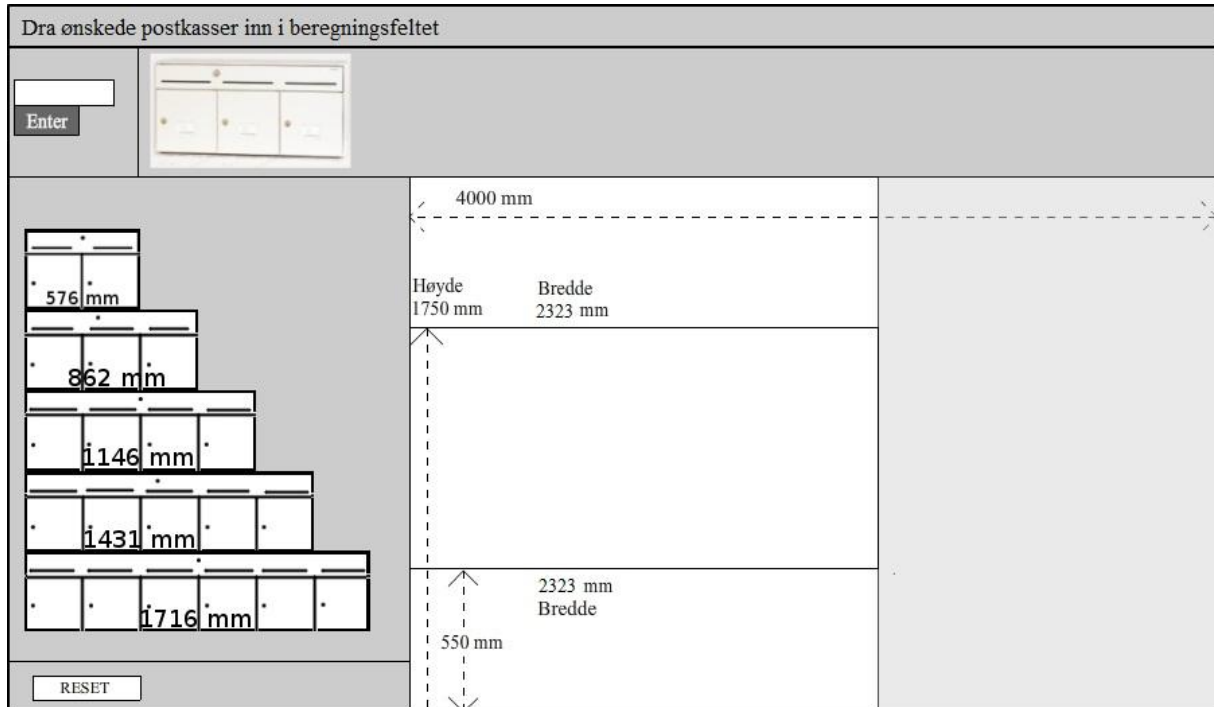
Høyde	Bredde
1750 mm	2323 mm
550 mm	2323 mm

*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.

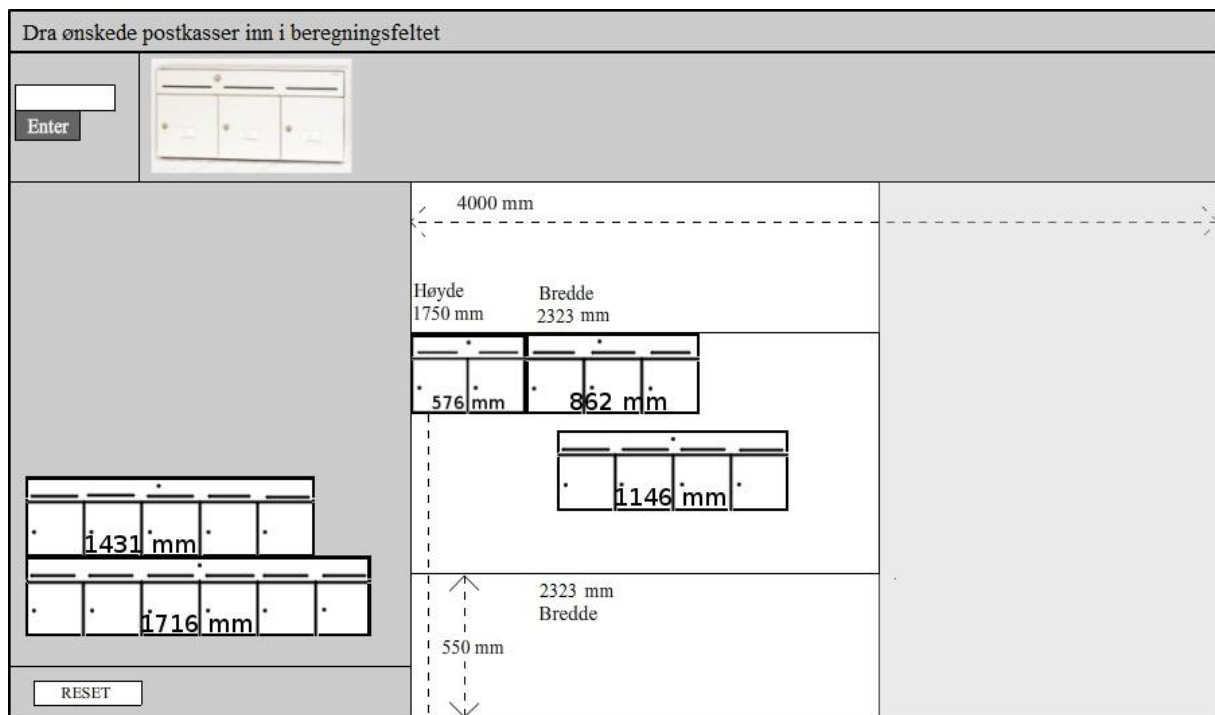
Hjelpeteksten er forandret til ”Dra ønskede postkasser inn i beregningsfeltet.” og er det som brukeren må gjennomføre i dette steget i prosessen. Brukeren kan da dra de ønskelige postkassene inn i feltet som er regnet ut ifra breddemålet som er tastet inn tidligere i prosessen.



*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.

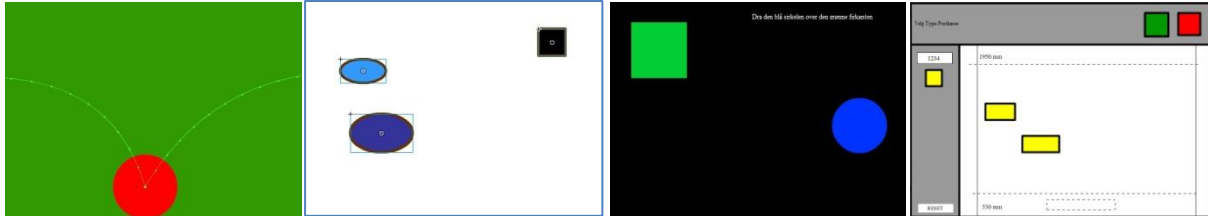
## Fremgangsmåte for Flash

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 løsningen. 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 Actionsript

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 akse 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 parenteser 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(Gull1).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.

```
201 function dotry(event:MouseEvent):void {
202
203     DisplayObject(Gul1).visible = false;
204     DisplayObject(Gul2).visible = false;
205     DisplayObject(Gul3).visible = false;
206
207     DisplayObject(Robust1).visible = false;
208     DisplayObject(Robust2).visible = false;
209     DisplayObject(Robust3).visible = false;
210     DisplayObject(Robust4).visible = false;
211     DisplayObject(Robust5).visible = false;
212
213
214     DisplayObject(Gul1).x = 36;
215     DisplayObject(Gul1).y = 167;
216
217     DisplayObject(Gul2).x = 21;
218     DisplayObject(Gul2).y = 234;
219
220     DisplayObject(Gul3).x = 13;
221     DisplayObject(Gul3).y = 293;
222
223     DisplayObject(Robust1).x = 13;
224     DisplayObject(Robust1).y = 167;
225
226     DisplayObject(Robust2).x = 13;
227     DisplayObject(Robust2).y = 227;
228
229     DisplayObject(Robust3).x = 13;
230     DisplayObject(Robust3).y = 287;
231
232     DisplayObject(Robust4).x = 13;
233     DisplayObject(Robust4).y = 347;
234
235     DisplayObject(Robust5).x = 13;
236     DisplayObject(Robust5).y = 406;
237
238     visbreddetext.text="";
239     visbreddetext2.text="";
240     breddetext.text="";
241
242     DisplayObject(infoNede).visible = false;
243     DisplayObject(infoOppe).visible = false;
244     DisplayObject(Red_btn).visible = false;
245     DisplayObject(Robust_btn).visible = false;
246     DisplayObject(overskriftDra).visible = false;
247     DisplayObject(overskriftBredde).visible = true;
248     DisplayObject(mmmOppe).visible = false;
249     DisplayObject(mmmNede).visible = false;
250     DisplayObject(overskriftPostkasse).visible = false;
251     DisplayObject(helBreddeText).visible = false;
252     DisplayObject(skrav).visible = false;
253     DisplayObject(skravTo).visible = false;
254 }
```

*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 har 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

```
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(mmmOppe).visible = true;
326         DisplayObject(mmmNede).visible = true;
327         DisplayObject(helBreddeText).visible = true;
328     }
329 }
330 }
```

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

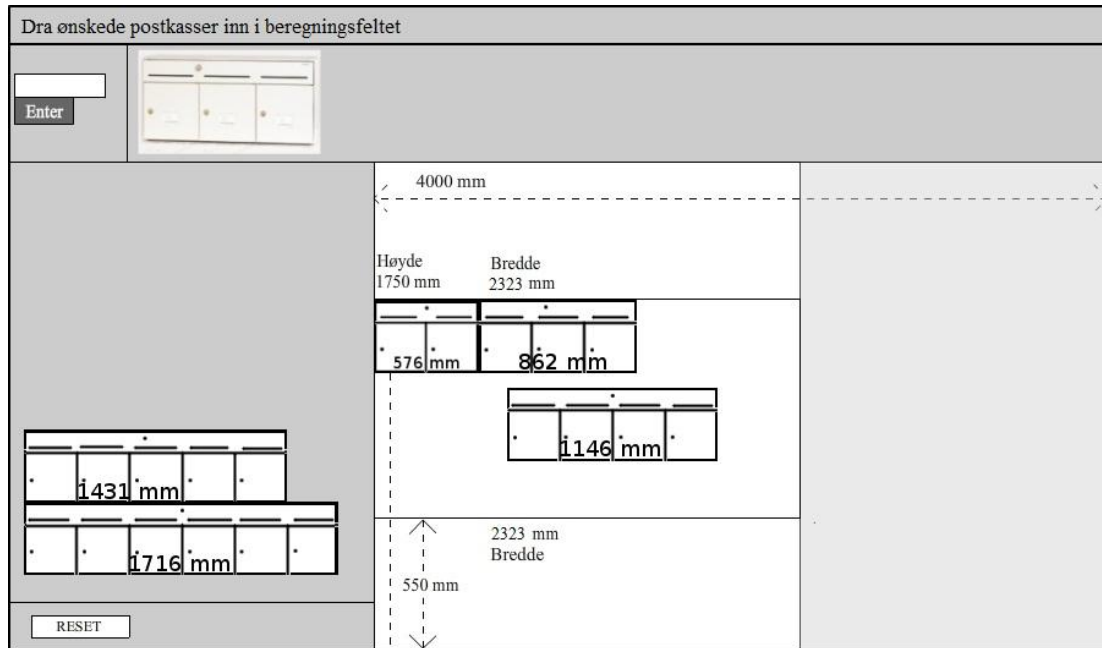
utregningen for å fastslå området på rammen begynner  $(4000 / 10) * 1.5$ , hvor resultatet blir antall piksler i bredden og lagrer seg i en variabel. Deretter kommer det en utregning som tar utgangspunkt i "input" teksten til brukeren, altså hva brukeren skriver inn i tekstboksen. Utregningen  $(\text{bredde} / 10) * 1.5$  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 ha 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.





## SAMMENDRAG

Oppgaven vi fikk av Stansefabrikken Products AS gikk ut på å lette arbeidsmengden til de ansatte i bedriften og å gjøre salgsprosessen enklere for deres kunder. Vi valgte å løse dette ved å utvikle en webshop for postkasser og en beregningsmodell for plassering av postkasser i oppganger. Kravene til systemet utarbeidet vi i samarbeid med de ansatte i bedriften, men vi sto ganske fritt til å utvikle løsningen så lenge det ferdige produktet var i henhold til kravspesifikasjonen.

Beregningsmodellen ble utviklet i Flash og webshopen ble utviklet i PHP, HTML og CSS. Ingen på gruppen hadde erfaring fra bruk av Flash før dette prosjektet, så en viktig del av utviklingsprosessen ble å oppnå kompetanse for denne teknologien. I tillegg satt vi oss inn i andre prosesser som for eksempel sikkerhetsaspekter som er nødvendige i forhold til en webshop.

Resultatet av produktet står i sammenheng med kravspesifikasjonen og oppfylder bedriftens ønsker.

## KONKLUSJON

Vi har laget et system som dekker de behovene som Stansefabrikken Products AS ønsket en løsning for. Disse behovene ble utlyst i kravspesifikasjonen som ble utarbeidet i samarbeid med bedriften i oppstarten av prosjektet. Systemet i dag fremstår som ferdig og klart til implementering rett på Stansefabrikken sine nettsider. Det endelige produktet består av et nettbestillingssystem med mulighet for bestilling av produkter over nettet og en ”beregningsmodell” som er en flashapplikasjon for utregning av plass til systemkasser. Beregningsmodellen var til å begynne med tenkt som et hjelpemiddel laget ved hjelp av regneark og tabeller. Dette gikk vi etter hvert bort ifra og foreslo for Stansefabrikken å utvikle denne i Flash for å få en mer visuell og brukervennlig applikasjon. Produktet er nå overlevert til bedriften og de står fritt til å bruke og/eller videreutvikle systemet til eget bruk.

## KILDER

Use Case: [http://home.online.no/~moestboe/uml\\_oversikt.htm](http://home.online.no/~moestboe/uml_oversikt.htm)

PHP: <http://no.wikipedia.org/wiki/PHP>

<http://no2.php.net/manual/en/function.mysql-real-escape-string.php>

<http://no.wikipedia.org/wiki/Md5>

<http://no2.php.net/manual/en/function.mysql-query.php>

CSS: [http://no.wikipedia.org/wiki/Cascading\\_Style\\_Sheets](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/](http://www.adobe.com/products/player_census/flashplayer/)

[http://www.adobe.com/devnet/flash/articles/flash\\_cs4\\_createfla.html](http://www.adobe.com/devnet/flash/articles/flash_cs4_createfla.html)

<http://www.actionscript.org/resources/articles/16/1/Instance-Names/Page1.html>

<http://www.smashingmagazine.com/2008/01/17/adobe-flash-tutorials-best-of/>

<http://www.adobe.com/devnet/flash/?navID=gettingstarted>

[http://edutechwiki.unige.ch/en/Flash\\_drag\\_and\\_drop\\_tutorial](http://edutechwiki.unige.ch/en/Flash_drag_and_drop_tutorial)

<http://www.frenchsquared.com/>

[http://help.adobe.com/en\\_US/Flash/10.0\\_UsingFlash/Wd60f23110762d6b883b18f10cb1fe1af6-7d40a.html](http://help.adobe.com/en_US/Flash/10.0_UsingFlash/Wd60f23110762d6b883b18f10cb1fe1af6-7d40a.html)

<http://www.actionscript.org/forums/showthread.php3?t=174047>

[http://www.adobe.com/devnet/flash/quickstart/drawing\\_commands\\_as3/](http://www.adobe.com/devnet/flash/quickstart/drawing_commands_as3/)

<http://social.msdn.microsoft.com/Forums/en-US/csharp/language/thread/17bd0d8e-ece1-4bee-a926-fe67bccd4081/>

<http://injun.ru/flash10api/flash/text/TextField.html>

<http://www.kirupa.com/forum/showthread.php?t=321969>

[http://www.flashandmath.com/basic/dragdroptour/dd\\_tour1.html](http://www.flashandmath.com/basic/dragdroptour/dd_tour1.html)

## ORDBOK

Actionscript	Programmeringsspråk for Adobe Flash.
Admin	Forkortelse for administrator.
Aktør	Hvem som utfører selve prosessen i en Use Case.
Body	HTML body tag. Definerer body området i et HTML dokument.
C	Programmeringsspråk.
C++	Programmeringsspråk.
Cookie	Informasjonskapsel. Fil som lagres for at nettleseren skal ivareta informasjon.
CSS	”Cascading Style Sheets”. Brukes til å definere utseende på filer skrevet i HTML eller XML.
Database	Samling av data.
Drag and Drop	Betyr at brukeren kan interagere med eventuelle elementer ved å ta tak i ett element og plassere det på ønskelig sted.
Echo	En kommando som gir direkte utskrift.
E-post	Elektronisk post som blir sendt over et nettverk.
ER-diagram	En illustrasjon av databasen.
Extend	Arv av oppførsel mellom to Use Case.
Flash	Betegnelse som refererer til programvaren Adobe Flash Player. Gir animasjon på nettsider.
GIMP	”GNU Image Manipulation Program”. Blir brukt til å behandle og lage illustrasjoner, digital grafikk og fotografier.
Graphics	Actionscript klasse som tilbyr muligheter for å tegne grafiske illustrasjoner.
Grensesnitt	Betegnelse for hvordan en bruker kommuniserer med ett system.
Hash	Sikkerhetsfunksjon. Hindrer uønsket innsyn i informasjon. Blir generert ut i fra en string med tekst.
Hexadesimal	Tallsystem basert på 16 tall. De desimale tallene beskrives med 0 til 15 med sifrene 0 til 9 og bokstavene A til F.

HTML	”HyperText Markup Language”. Markeringsspråk for formatering av nettsider med informasjon som skal vises i en nettleser.
IBM Rational Rose	Program som brukes til å lage Use Case diagram.
If-løkke	Kode som utfører en handling hvis spesielle kriterier er oppfylt.
Include	Kommando som inkluderer data, informasjon eller liknende.
Input	Data fra en bruker til et system.
Instance name	Unikt navn for et objekt i Adobe Flash.
Keyboardevent	Funksjoner som skal reagere på tastetrykk fra tastaturet må kalles på med denne kommandoen.
Layout	Utseende eller designet på et system.
Link	Link (”Hyperlenke”) er en henvisning i et dokument til et annet dokument.
Loop	Noe som gjentas i en løkke til det forstyrres eller utgår.
Margin	Kommando som setter størrelse på området utenfor et element i HTML.
Mouseevent	Funksjoner som skal reagere på museklikk må kalles på med denne kommandoen.
Movieclip	Type symbol med spesielle egenskaper som et objekt kan være i Adobe Flash.
MySQL	Versjon av SQL (Structured Query Language). Relasjonsdatabase for behandling av data.
Netthandel	Norsk uttrykk for handel på nett.
Nettleser	Program som benyttes for å lese nettsider. F. eks Explorer, Firefox, Opera og Safari.
Objektorientert	Metodikk for dataprogrammering
Operativsystem	Grunnleggende system på en datamaskin. Det styrer ressurser og oppgaver som brukeren ønsker å utføre.
Output	Data fra et system til en bruker.
Padding	Kommando som setter størrelse på området innenfor et element i HTML.

PHP	”Hypertext Preprocessor”. Programmeringsspråk. Brukes til å utvikle nettsider.
Piksel	Punkter som sammen utgjør et bilde.
Plug-in	Tilleggsprogram med funksjoner som en bruker kan tilføre andre programmer som allerede er installert på en datamaskin.
Postbetingelse	Resultat av prosessen som blir gjennomført.
Prebetingelse	Betingelse for at en prosess kan gjennomføres.
Prototype	Førsteutkast av et system.
Session	En session cookie lagres i minne og blir slettet etter at nettleseren lukkes. Brukes til og midlertidig lagre informasjon.
SQL-injections	En bruker skriver inn feil type data som blir sendt til databasen, hvor resultatet er negativ innvirkning på databasen.
Stage	Selve rammen hvor en Flash applikasjon utvikles på.
Startdrag	Kommando som definerer når drag and drop skal starte.
Stopdrag	Kommando som definerer når drag and drop skal stoppe.
Stream	Data som blir sendt over Internett til en bruker som ikke trenger å laste ned informasjonen før man får tilgang til den.
String	Datatype som består av tekst.
Table definition	”td” tag som definerer en tabell i HTML.
Table row	”tr” tag som definerer rader i en tabell i HTML.
Timeline	Alternativ fremgangsmåte for utvikling av prosjekter i Adobe Flash.
Trigger	Hva som må utføres for at en prosess skal starte.
Use Case	Beskrivelse av et systems oppførsel.
Variabel	Symbol som representerer en dataverdi.
Vektorgrafikk	Brukes til grafiske illustrasjoner som kan tolkes entydig.
Webshop	Uttrykk for handel på nett.
While-løkke	Kode som utfører noe så lenge angitte kriterier er oppfylt.
XML	Markeringsspråk for HTML.

Åpen kildekode

Kildekoden til et system eller program er offentlig slik at hvem som helst kan ha tilgang til den.