Danmarks
Tekniske Universitet

Web applikation for et Isoleringsfirma

Emil Geutze Høgskilde (s103662)

Afgangsprojekt, Diplom-IT ved DTU

Afleverings dato:14/07/2014

1. ABSTRACT

Purpose: The project will deliver a web application where the total energy savings by replacing the windows in a detached house with new windows are calculated. The calculation shall be delivered as a Web report.

Background: Energy consumption for building operations account for approximately 40% of total energy consumption in Denmark. If the energy consumption is to be reduced significantly, it should be done through energy savings in existing buildings. The web application is developed together with Amon Energy a new Danish company that focuses on energy savings.

Method: The analysis phase identified the information to be provided in order to calculate the energy savings from replacing windows in a detached house. Based on the conduct of business use cases and information on energy saving calculation, was a domain model developed that could be used as a basis for the design of the Web application in the design phase. Within the design phase has a set of requirements and objectives primarily to the applications input surfaces, calculation and reporting of energy been identified. The database tables have been shown in a specific ER diagram.

The implementation chapter is a listing of some of the most important elements that have been developed in the program. During the implementation phase is the setup of the project and the overall structure of the web application described. Communication with the database is described and the description of the web application forms are divided into entry forms for the customer and installer lists of buildings and energy reports and energy report. The system is based on object technology (ASP.NET with C # as code behind) as the front end, and relations technology (MSSQL) as backend.

Result: The web application contains the entry forms from customers and installers (data sources) and it is able to show a report of energy savings. The web application will mostly be used on tablet and communicates with the company's server where the calculations are done.

Conclusion: It is possible to calculate and report energy savings from replacing existing windows in a detached house with new windows. The calculation and reporting can be delivered immediately after the installer has conducted the survey. There are good opportunities to further develop the program.

2. INDHOLDSFORTEGNELSE

3. INTRODUKTION

Projektet skal levere en webapplikation hvor den samlede energibesparelse ved at udskifte vinduerne i et parcelhus med nye vinduer beregnes. Beregningen skal leveres som en webrapport og applikationen skal udformes som et pilotprojekt i forhold til et større sammenhængende produkt, som beregner de økonomiske konsekvenser ved at foretage en optimering af den eksisterende boligmasses energiforbrug, primært parcelhuse fra perioden 1960 – 1990. Beregning af energibesparelsen skal baseres på en offentligt anderkendt beregningsmetode og informationerne som anvendes i beregningen leveres af den person der ejer huset, den person der udskifter vinduerne, producenten af vinduerne og offentlige informationer.

Idégrundlag

Ideen til projektet er opstået i samarbejde med firmaet Amon Energy IS (AE).

Amon Energy IS.

Virksomheden ønsker at medvirke til at det danske ressourceforbrug reduceres. Med det formål er udviklet et koncept som primært fokuserer på at sænke energiforbruget i den del af det eksisterende danske boligareal hvor potentialet for energibesparelser er størst. Energiforbruget til bygningsdrift udgør omkring 40 % af Danmarks samlede energiforbrug. Af dette bruges størstedelen til boligopvarmning. Hvis energiforbruget fra bygningsdrift skal reduceres markant skal det ske gennem energibesparelser i det eksisterende byggeri.

Virksomhedens grundlæggende ide er, at etablere en platform hvor kunder, interesserede i at reducere deres ressourceforbrug, kan henvende sig. Virksomheden vil herefter på baggrund af informationer fra kunden og en vurdering af kundens bygninger levere en rapport med en samlet vurdering af hvilke tiltag det vil kunne betale sig at gennemføre.

Alle beregninger skal baseres på beregninger godkendt af offentlige myndigheder f.eks. DTU. Virksomheden ønsker herudover at indgå en samarbejdsaftale f. eks. med DTU med henblik på at vurdere og dokumentere effekten af energibesparelsesforanstaltninger/virksomheden indsats.

Produktet

Det samlede produkt er således en vurdering af mulige energibesparelser ved drift af kundens ejendom, et tilbud på at gennemføre de energibesparelser som er rentable og gennemførelse af besparelserne i et samlet projekt (bilag 1 og 2).

Ønsket er at dette produkt omfatter alle mulige energibesparelser ved drift af ejendommen, men AE vil i starten fokuseret på besparelser ved isolering af klimaskallen - ydervægge, lofter, rem, sokkel og udskiftning af vinduer.

Kunden vil uden beregning modtage en energimæssig vurdering af deres ejendom. På basis af specifikke data for ejendommen vedrørende isolering og vinduer beregnes energibesparelsen ved at foretage en yderligere isolering og udskiftning af vinduer. Ved beregningen anvendes data fra producenter og offentligt godkendte beregningsmetoder (f.eks. DTU).

Projektet

Projektet er udviklet som afsluttende projekt på diplomingeniør uddannelsen i IT på DTU. Det er afgrænset til udelukkende at fokusere på beregning af energibesparelser ved udskiftning af vinduer.

I forhold til det samlede produkt er disse beregninger helt centrale og det er vurderingen at såfremt disse ikke kan leveres på en overskuelig og troværdig måde vil det ikke være besværet værd at forsætte med etablering af resten af produktet.

Projektet skal således udformes som en webapplikation:

- Hvor den samlede energibesparelse ved at udskifte vinduerne i et parcelhus med nye vinduer beregnes
- Hvor beregningen leveres som en webrapport
- Hvor applikationen er udformet som et pilotprojekt i forhold til et større sammenhængende produkt, som beregner de økonomiske konsekvenser ved at foretage en optimering af den eksisterende boligmasses energiforbrug, primært parcelhuse fra perioden 1960 – 1990.
- Hvor beregning af energibesparelsen baseres på offentligt anderkendte beregningsmetoder
- Hvor informationer som anvendes i beregningen leveres af
    - Den person der ejer huset – webformular - brugerinformationer (Customer)
    - Den person der udskifter vinduerne – webformular - montørinformationer (Montor)
    - Producenten af vinduerne – fabrikantinformationer (Manufacturer)
    - Offentligt certificerede og publicerede – offentlige informationer (State)

Yderligere er projektet afgrænset til alene at indeholde informationer fra en enkelt producent af vinduer.

AE medvirker og har det overordnede ansvar for at bekrive hvilke informationer som skal anvendes i beregningen af energibesparelser og levering af de nødvendige værdier og formler som skal bruges. Det er efterfølgende AE's ansvar sikre at de anvendte værdier, beregningsmetoder og resultater er pålidelige og kan "godkendes" af offentlige instanser.

Interessenter

Det færdigt udviklede produkt vil potentielt kunne få stor samfundsmæssig betydning. Umiddelbart vil det kunne skabe værdi på flere niveauer.

1. Kunderne.

For firmaets målgruppe af kunder med dårligt isolerede huse vil gevinsterne være levering af et tilbud som er overskueligt, let at forstå og let at gennemføre. For alle vil der hvis de gennemfører energirenoveringen være en økonomisk gevinst og det vil betyde at indeklimaet i deres hus forbedres.

2. Amon Energy

For virksomheden vil det være af vital betydning at produktet etableres. Det vil være produktet som adskiller virksomheden fra andre virksomheder på markedet. Det vil være et effektivt arbejdsredskab, som skaber tryghed og tillid hos kunderne og det vil ligeledes være produktet og konceptet bag produktet, som sikrer en speciel interesse for deltage i samarbejdet med virksomheden, blandt de potentielle samarbejdspartnere.

3. Firmaets samarbejdspartnere

For alle virksomhedens samarbejdspartnere er det, specielt i lyset af det fokus der er på den globale opvarmning, af betydning for deres image, at deltage i en proces der medvirker til at CO_2 udledningen reduceres. For alle vil det desuden være forbundet med en økonomisk gevinst at deltage.

For håndværkere, leverandører og producenter vil produktet kunne levere overskuelige projektbeskrivelser med tids og leveringsplaner. Der vil kunne udformes elektroniske og præcise bestillingsinformationer og data vil kunne anvendes i planlægningsværktøjer, f.eks. mandskabsstyring. Deres samlede tidsforbrug vil reduceres.

For banker, kreditforeninger vil det primært have betydning at kundernes økonomi forbedres og for forsikringsselskaberne vil deres risiko vil kunne reduceres i forbindelse med gennemgangen og forbedringen af bygningen. For ejendomsmæglere vil det have tiltagende betydning for salgbarheden af husene, at de lever op til de energimæssige forskrifter.

4. Det danske samfund

For det danske samfund vil produktet kunne medvirke til at borgerne får flere penge til forbrug og at omkostningerne til energi reduceres. Der skabes arbejdspladser og belastningen af miljøet reduceres.

Grundlæggende er det opfattelsen at en væsentlig del af den energi der i dag bruges, anvendes ufornuftig og at alle interessenter vil have en gevinst ved en mere intelligent tilgang, herunder brug og genbrug af elektronisk registrerede data og ensartede anerkendte beregningsmetoder. Hertil tilføjes reduktionen af den miljømæssige belastning.

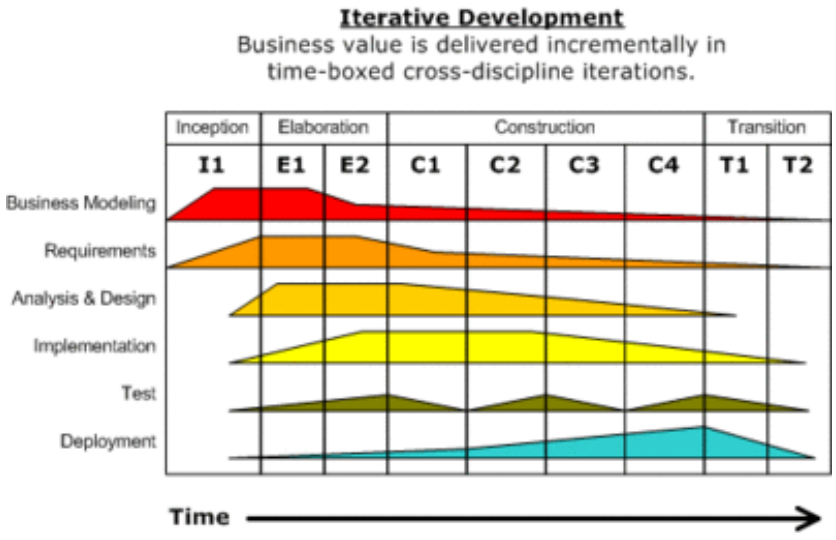
Udviklingsmetode

Projektet er et enkeltmandsprojekt og de fleste metoder til udvikling af IT projekter er baseret på udviklingsteams. SCRUM blev overvejet som udviklingsmetode, men blev forkastet, idet tidsforbruget blev vurderet som betydeligt i forhold til det forventede udbytte ved at bruge metode.

Reelt foregik udviklingen alene som en kommunikation mellem mig og en enkelt person i firmaet, hvor en tilnærmet version af metoden Unified Process (UP) der er en objektorienteret softwareudviklingsproces blev anvendt. UP er en iterativ metode som overordnet kan inddeles fire faser:

- Forberedelse - Inception
- Etablering - Elaboration
- Konstruktion - Construction
- Overdragelse - Transition

Figur 1 (fra wikipedia) viser en illustration af UP's projektforløb.



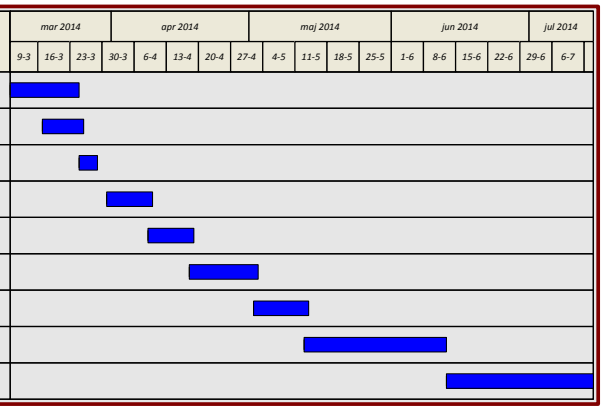
Figur 1. UP tidslinje (Wikipedia)

Analysefasen foregår således hovedsageligt i de 2 første faser, men reelt foregår analysearbejdet i hele projektforløbet. Overgangen mellem faserne er ligeledes flydende, ligesom fasernes indhold er flydende med hovedvægt på forskellige aktiviteter. Der kan således godt programmeres i forberedelsesfasen, såvel som analyseres i konstruktionsfasen.

Projektplanlægning

Initalt blev projektet som varer 18 uger inddelt i 9 opgaver i henhold til nedenstående Gantt diagram

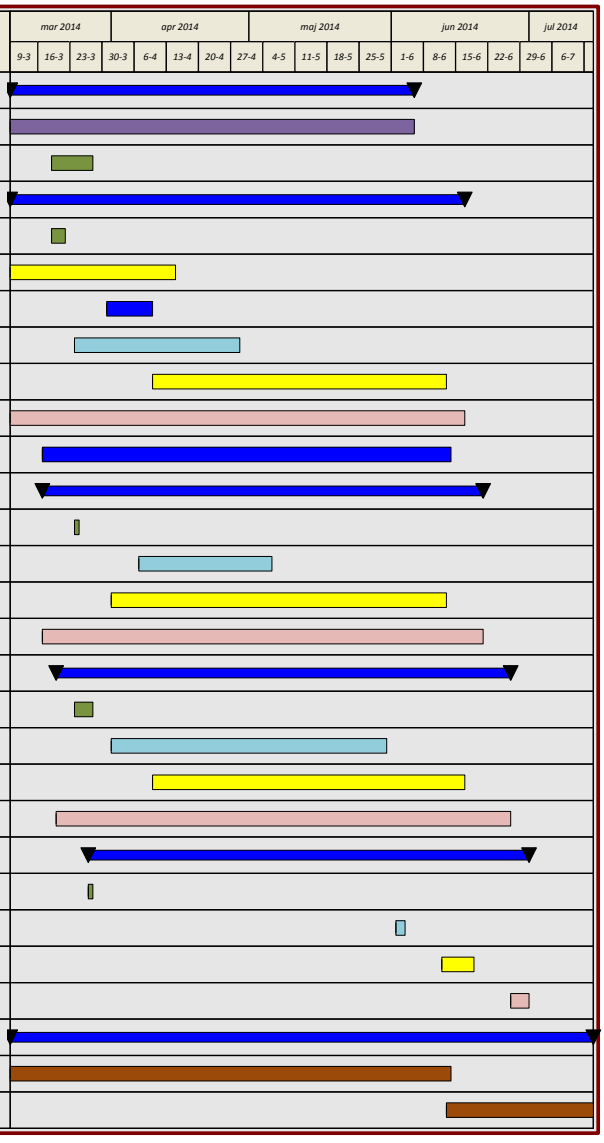
Id	Opgavenavn	Startdato	Slut	Varighed
1	Opsætning af server/udviklingsmiljø	10-03-2014	24-03-2014	11d
2	Brugerhåndtering og sikkerhed	17-03-2014	25-03-2014	7d
3	Kundeformular	25-03-2014	28-03-2014	4d
4	Montørformular	31-03-2014	09-04-2014	8d
5	Udregninger og rapporter - screening	09-04-2014	18-04-2014	8d
6	Udregninger og rapporter - før og efter	18-04-2014	02-05-2014	11d
7	Udregninger og rapporter - Energibesparelse	02-05-2014	13-05-2014	8d
8	Test, korrektioner og optimering	13-05-2014	12-06-2014	23d
9	Rapport	13-06-2014	14-07-2014	22d



Figur 2. Initial projektplanlægning

Planlægning blev i forhold til UP metoden ændret til 6 faser med en række iterationer, som vist i nedenstående Gantt diagram

Id	Opgavenavn	Startdato	Slut	Varighed
1	**Opsætning**	**10-03-2014**	**05-06-2014**	**64d**
2	Opsætning af server/udviklingsmiljø	10-03-2014	05-06-2014	64d
3	Brugerhåndtering og sikkerhed	19-03-2014	27-03-2014	7d
4	**Analyse**	**10-03-2014**	**16-06-2014**	**71d**
5	Login	19-03-2014	21-03-2014	3d
6	Vinduer	10-03-2014	14-04-2014	26d
7	Sammenhæng	31-03-2014	09-04-2014	8d
8	Brugerinformationer	24-03-2014	28-04-2014	26d
9	Vægge	10-04-2014	12-06-2014	46d
10	Beregninger	10-03-2014	16-06-2014	71d
11	Energibesparelse rapport	17-03-2014	13-06-2014	65d
12	**Design**	**17-03-2014**	**20-06-2014**	**70d**
13	Login	24-03-2014	24-03-2014	1d
14	Brugerformular	07-04-2014	05-05-2014	21d
15	Montørformular	01-04-2014	12-06-2014	53d
16	Energibesparelse - rapport	17-03-2014	20-06-2014	70d
17	**Implementering**	**20-03-2014**	**26-06-2014**	**71d**
18	Login	24-03-2014	27-03-2014	4d
19	Brugerformular	01-04-2014	30-05-2014	44d
20	Montørformular	10-04-2014	16-06-2014	48d
21	Energibesparelse - rapport	20-03-2014	26-06-2014	71d
22	**Test**	**27-03-2014**	**30-06-2014**	**68d**
23	Login	27-03-2014	27-03-2014	1d
24	Brugerformular	02-06-2014	03-06-2014	2d
25	Montørformular	12-06-2014	18-06-2014	5d
26	Energibesparelse - rapport	27-06-2014	30-06-2014	2d
27	**Rapport**	**10-03-2014**	**14-07-2014**	**91d**
28	Delelementer	10-03-2014	13-06-2014	70d
29	Sammenskrivning	13-06-2014	14-07-2014	22d



Figur 3. Projekt planlægning (UP)

Risikoanalyse

For projektets gennemførelse blev det hurtigt åbenlyst, at det var forbundet med væsentligste udfordringer.

- Projektets største udfordring var at det skulle foregå i et ikke allerede etableret IT udviklingsmiljø. Der var således ikke mulighed for opbakning fra ingeniører eller programmører med erfaring i
    - Systemopsætning, serveropsætning programvalg og programmering.
    - Afvikling af programmer og kommunikation med databasen
- Det blev også hurtigt klart at det var vanskeligt at fremskaffe
    - anerkendte præcise beregningsmetoder fra det offentlige
    - alle nødvendige værdier fra vinduesproducenterne

De vigtigste risici, deres sandsynlighed og indvirkning på projektet fremgår af nedenstående tabel. Værdierne går fra 1 til 10, hvor 10 er den største sandsynlighed indvirkning og 1 den mindste.

Risikoanalyse	Sandsynlighed	Indvirkning
Sygdom	1	4
Levering af data/beregningsmetoder	6	6
Tab af data	1	5
Kravændringer/kravtilføjelser	4	4
Serveropsætning	8	5
Programvalg/programmering	5	5
Systemopsætning/kommunikation	3	4

Tabel 1. Risikoanalyse

Delkonklusion

Under introduktion er projektet og dets afgrænsninger blevet præsenteret. I projektet udvikles en webapplikation, som leverer en beregning af energibesparelsen ved at udskifte de eksisterende vinduer i et parcelhus med nye vinduer. Applikationen er at betragte som et pilotprojekt for et større sammenhængende produkt, som skal anvendes af virksomheden Amon Energy IS.

Det er virksomhedens vision at ændre standarden for tilbudsgivning således at denne baseres på brug og genbrug af elektronisk registrerede data i det omfang de kan tilvejebringes, at beregninger af energibesparelser baseres på offentligt anerkendte beregningsmetoder og at energirenovering af den eksisterende boligmasse anspores.

4. ANALYSE

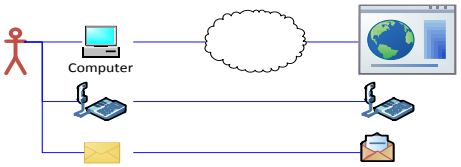
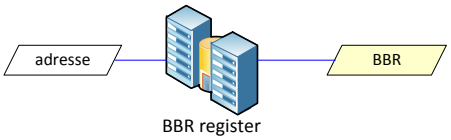
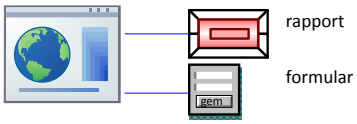
Problemformulering

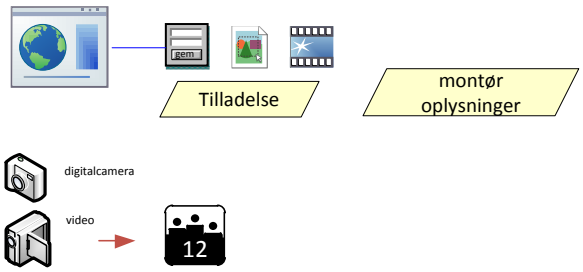
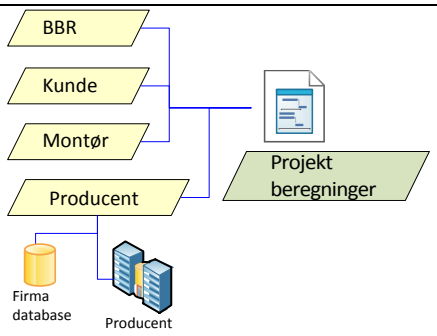
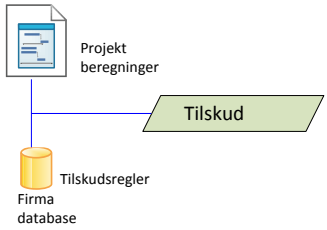
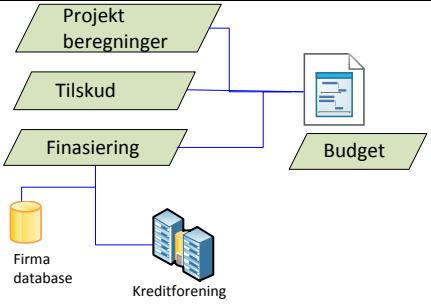
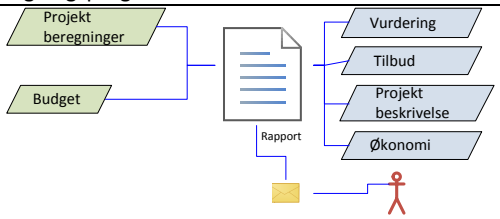
Initialt er det vigtigt at skabe et overblik over de krav som skal håndteres af projektet. Det projektet skal levere er en webapplikation som beregner energibesparelsen ved at udskifte vinduerne i et parcelhus. I den forbindelse kan kravene til projektet inddeles i 3 hovedområder:

- Kontekst. For at kunne levere det bedst mulige produkt er det indledningsvis vigtigt at forstå firmaets forretningsmodel, således at applikationen kan designes under bedst mulig hensyntagen til den kontekst i hvilken den skal fungere.
- Indhold. Webapplikationen skal levere en rapport. Hvorledes sikres et fagligt korrekt indhold. Hvordan foretages beregningerne. Hvilke data er nødvendige og hvordan leveres de.
- System. Baseret på ovenstående ønsker/behov kan der opstilles en række krav til udformning af det system som afvikler webapplikationen. Herunder krav til server, udviklingsmiljø og database.

Forretningsmodel

Firmaets forretningsmodel kan beskrives som en række sammenhængende steps. I nedenstående figur 3 er angivet en forenklet model hvor de opgaver som skal varetages i forbindelse med projektet er markeret med gult.

1. Kunden henvender sig – via hjemmeside, telefonisk eller email	
Opgaver.	Virksomheden skal have hjemmeside På hjemmeside skal der kunne aktiveres en formular hvor kunden indtaster oplysninger. Oplysningerne skal kunne gemmes og anvendes af virksomheden.
2. Virksomheden henter bygningsinformationer fra BBR registeret og fra eventuelle andre registrer.	Ud fra ejendomsoplysninger hentes informationer fra BBR registeret og eventuelle andre registre automatisk. 
Opgaver	Der indgås aftale med leverandør af BBR data
3. De modtagne data bearbejdes og der udformes montørrapport og montørformular. a. Montørrapporten indeholder alle relevante informationer om bygningen i en overskuelig rapport. b. Montørformular. Denne indeholder indtastningsfelter for alle de informationer vi mangler for at kunne tage stilling til et samlet projekt.	
Opgaver	På hjemmesiden skal der kunne aktiveres en formular hvor montøren kan indtaste supplerende oplysninger. Oplysningerne skal kunne gemmes og anvendes af virksomheden.

4. Montøren besøger kunden. a. Her gennemgås bygningen og alle supplerende informationer indsamles b. Specielle forhold noteres, herunder forhold som vil betyde at tidsplan og budget vil ændres i forhold til standard (mindre/større) c. Montøren tager relevante billeder og evt. video. d. Informationerne leveres via montør webformularen e. Kunden anmodes om tilladelse til at hente informationer fra f.eks. kreditforening, såfremt sådanne tilladelser er nødvendige.	
Opgaver	Billeder og video skal kunne overføres fra vores webside. Der skal kunne afholdes videokonference med relevante parter mens montøren er hos kunden med henblik på afklaring af specielle forhold.
5. Informationerne overføres til serveren og beregninger foretages a. BBR b. Kunde informationer c. Montør informationer d. Producent informationer i. Vi sender informationer til producenterne ii. Vi har alle informationer fra producenterne	
Opgaver	Producent informationer skal hentes. Programmer skal udformes. Der skal etableres en server som kommunikerer med hjemmesiden/webapplikationen.
6. Tilskud beregnes ud fra de beregnede besparelser for de dele af projektet det vil kunne betale sig at gennemføre.	
Opgaver	Regler for tilskud skal tilføjes databasen. Beregningsprogrammer skal udformes.
7. Budget for projektet og finansiering beregnes.	
Opgaver	Aftale med kreditforeningen, eventuelt skal vi bruge informationer om hvilken type lån kunden kan tilbydes (hvor store afdrag?). Beregningsprogrammer.
8. Informationerne samles i en rapport/tilbud som sendes dagen efter besøg af montøren til kunden. Indehold: a. Vurdering af mulige energibesparelser b. Kontrakttilbud på gennemførelse af et projekt på baggrund af denne vurdering c. Beskrivelse af projektet d. Budget, finansiering og energibesparelse ved gennemføring af projektet.	
Opgaver	Rapport over den samlede besparelse ved at udskifte vinduerne i huset.

9. Tilbuddet accepteres a. Montøren som foretager tilpasning af projektet efter kommunikation med kunden. F.eks. farve på hus og vinduer. Herunder også eventuelle andre opgaver som skal løses sammen med projektet. b. Informationerne indtastes i en webformular	
10. Endelig kontrakt udformes	

Figur 4. Forretningsmodel - projektopgaver

Indhold af webapplikationen

Webapplikationen skal levere en rapport over energibesparelsen ved at udskifte vinduerne i et parcelhus. For at dette kan lade sig gøre er det nødvendigt at analysere hvordan beregningen foretages og hvilke informationer som skal indgå. Herudover må det klarlægges fra hvilke datakilder informationerne leveres.

Beregning af energibesparelsen

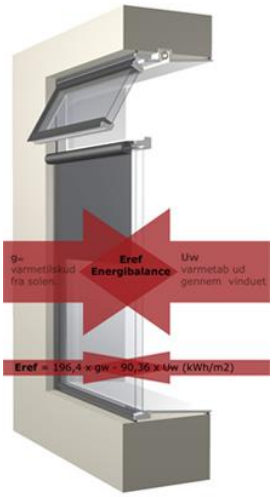
Den samlede energibesparelse på udskiftning af vinduerne er summen af besparelser ved udskiftning af hvert enkelt vindue.

I henhold til bilag 3. "Vinduer – begreber og energiberegning", som indeholder oplysninger leveret af firmaet, vurderes vinduers energiegenskaber ved at beregne energibalancen.

Energibalancen benævnes E_w for et vilkårligt vindue eller E_{ref} for et reference vindue (et vindues gennemsnitlige energiydeevne i referencehuset/Danmarkshuset beskrevet i DS418). Energibalancen (E_{ref}), som måles i kWh/m2/år, angiver hvor mange kWh der enten tabes eller vindes gennem vinduerne pr. kvadratmeter vinduesareal. Er tallet negativt, er der tale om et varmetab gennem vinduet. Er tallet positivt, tilfører vinduet bygningen varme (målt i fyringsæsonen).

Energibalancen (figur 5) for et vindue udtrykker hvor godt vinduet er til at lukke varme fra solen ind i huset og hvor godt det er til at holde på varmen i huset.

Energibalance = **solindstråling** x **vinduets energitransmittans (g_w)**
-
gradtimer for DK x **vinduets varmetab (U_w).**



Figur 5. Vindue - energibalance

Solindstråling

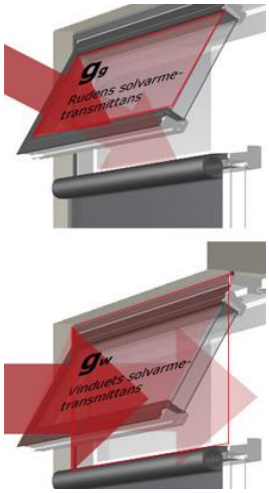
Solindstrålingen er afhængig af vinduets orientering (nord/syd/øst/vest) og hvilken på hvilken breddegrad huset er placeret. Til projektet har firmaet leveret en tabel med solstrålingsværdier 0 – 360 grader for en gennemsnitlig breddegrad i Danmark (tabel i databasen).

Gradtimer

Gradtimer tager udgangspunkt i danske klimaforhold, hvor fyringssæsonen regnes som perioden 24. september til 13. maj. Firmaet har angivet at værdien 90,36 skal benyttes (bilag 4. Vinduers varmetab, raadvad-centeret maj 2002 civilingeniør, arkitekt m.a.a. Thomas Kampmann).

Vinduets energitransmittans – g_w

Vinduets solvarmetransmittans (g_w) (figur 6), angiver vinduets evne til at lukke solens varme ind i bygningen. Solvarmetransmittansen tager højde for vinduets ramme-/karmareal og reducerer rudens varmebidrag, gg, med den procentdel, som ramme-/karmarealet optager. Solvarmetransmittansen kan sidestilles med vinduets U_w i vigtighed. Jo højere g-værdi, jo mere udnyttes solens varme.



Figur 6. Vindue - energitransmittans

Vinduets g-værdi udregnes som:

$$g_w = gg \times A_g/A_w$$

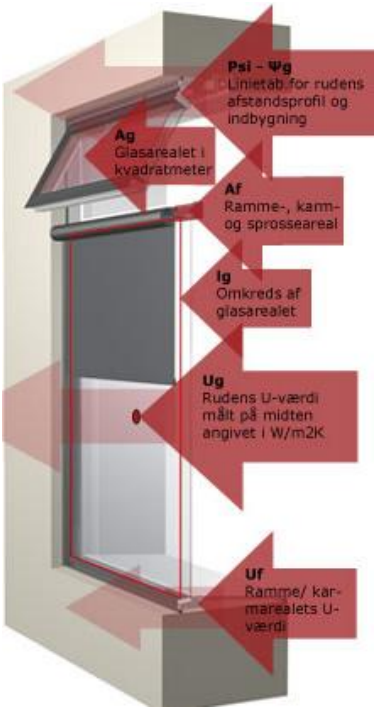
- hvor:

gg: er glassets solvarmetransmittans
A_g: er glassets areal
A_w: er vinduets samlede areal

Vinduets varmetab – U_w

U-værdier beskriver isoleringsgraden af en given konstruktion. Jo lavere værdi - jo mere mindskes varmetabet gennem konstruktionen. Det vinduets samlede isoleringsgrad inklusiv ramme og karm, der er interessant - denne kaldes U_w og måles i W/m2K.

U_w er således afhængig af vinduets størrelse samt rude- og ramme-/karmtype (figur 7). Den præcise U-værdi er afgørende i en energirammeberegning. Typisk angiver vinduesproducenten vinduets U_w-værdi efter en standardstørrelse 1,23 m x 1,48 m, men skal på forespørgsel kunne oplyse U_w på det aktuelle vindue.



Figur 7. Vindue - isoleringsgrad

Vinduets U-værdi udregnes som:

$$U_w = (A_g \times U_g + A_f \times U_f + I_g \times Psi_g) / (Ag + Af)$$

- hvor:

U_g:	= U-værdi - rude	målt på midten	W/m^2K
U_f:	= U-værdi - ramme-/karm		W/m^2K
A_g:	= Areal – glas		m^2
A_f:	= Areal – karm, ramme/sprosser		m^2
Psi:	= Linietab for rudens afstandsprofil + indbygning		W/m^2K
I_g:	= Omkreds af glasarealet		m

Sammenfattende vil det for at kunne beregne energibesparelsen ved at udskifte et vindue vil det være nødvendigt at følgende oplysninger leveres

- Vinduets orientering
  - Værdier for solindstrålingen
- Vinduets udformning så følgende kan beregnes
  - Glasareal
  - Karmareal
  - Rammeareal
  - Postareal
  - Sprosseareal
  - Omkredsen af glasarealet
- Information om

- Glassets solvarmetransmittans
- U værdier for
  - Glasset
  - Karmen
  - Rammen
  - Poster
  - Sprosser
- Linjetabet for rudens indbygning

Værdierne skal kendes både for de eksisterende vinduer som udskiftes og for nye vinduer.

Datakilder - energiberegning

Energistyrelsen/Fabrikanter

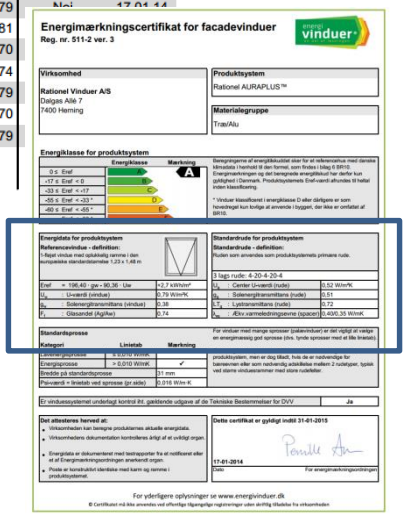
De fleste nye vinduer er i dag certificerede. Den frivillige Energimærkningsordning for vinduer er etableret af Vindues Industrien. Den viser på en skala fra A til F, hvor energirigtigt et vinduessystem er. I dag er der i henhold til Bygningsreglement BR10 lovkrav om, at vælge C-, B- eller A-mærkede energivinduer.

Via energistyrelsens hjemmeside kan man hente en positivliste med de forskellige certificerede vinduer. Nedenstående figur 8 viser et udsnit af denne liste

Ordnet alfabetisk efter virksomhedernes navn — **Se ordforklaringer her: http://energivinduer.dk/certifikatet** — Opdateret 28.04.14

Certifikat nummer	Energi klasse	Produktsystem	Materiale gruppe	Standardsprosse type	Standardsprosse bredde	Referencevindue E_{ref}	U_w	g_w	F_f	T_{oi} °C	Standardrude rude	U_g	g_g	LT_g	DVV http://dvv.dk	Certifikat gyldig fra	Ændret / Nyt
PROTEC Vinduer A/S																	
504-1	B	Protec 7 Multi	Træ/FRP	Lavenergi	20 mm	-17,2	1,01	0,38	0,77	13,5	3-lags	0,72	0,49	0,71	Ja	17.01.14	
504-2.3	C	Protec Classic	Træ/Alu	Energi	20 mm	-26,5	1,32	0,47	0,75	12,2	2-lags	1,14	0,63	0,80	Ja	17.01.14	
504-3.2	A	Protec Classic +	Træ/Alu	Lavenergi	20 mm	+2,6	0,77	0,37	0,75	14,1	3-lags	0,53	0,49	0,71	Ja	17.01.14	
504-4	B	Protec Classic Energy	Træ/Alu	Energi	20 mm	-16,7	1,44	0,58	0,75	12,0	2-lags	1,31	0,77	0,82	Ja	17.01.14	
504-5	A	Protec Classic+ Energy	Træ/Alu	Lavenergi	20 mm	+14,5	0,85	0,47	0,75	13,9	3-lags	0,64	0,62	0,73	Ja	17.01.14	
504-6	A	Protec Dreje-kip Energy	Træ/Alu	Lavenergi	20 mm	+8,2	0,92	0,47	0,75	14,9	3-lags	0,64	0,62	0,73	Ja	17.01.14	
504-7	A	Protec Xframe	FRP	Lavenergi	25 mm	+17,2	0,80	0,46	0,86	14,1	3-lags	0,58	0,53	0,72	Ja	28.04.14	X
Rationel A/S																	
511-1.3	A	AURA®	Træ/2 ØKO	Energi	31 mm	+3,6	0,78	0,38	0,74	14,2	3-lags	0,52	0,51	0,72	Ja	17.01.14	
511-2.3	A	Rationel AURAPLUS ™	Træ/Alu	Energi	31 mm	+2,7	0,79	0,38	0,74	14,5	3-lags	0,52	0,51	0,72	Ja	17.01.14	
511-3	A	Aldus A	Træ/Alu	Ingen sprosse		+4,2	0,91	0,44	0,71	14,0	3-lags	0,58	0,62	0,73	Ja	17.01.14	
511-4	B	Aldus B	Træ/Alu	Ingen sprosse		-9,3	1,06	0,44	0,71	13,7	3-lags	0,80	0,62	0,73	Ja	17.01.14	
511-5	C	Domus	Træ/2 ØKO	Energi	40 mm	-32,7	1,32	0,44	0,70	12,6	2-lags	1,14	0,63	0,80	Ja	17.01.14	
511-6	C	Aldus	Træ/Alu	Energi	40 mm	-31,9	1,31	0,44	0,71	13,0	2-lags	1,16	0,62	0,80	Ja	17.01.14	
511-7	C	Patus	Træ/2 ØKO	Energi	26 mm	-29,9	1,33	0,46	0,73	11,9	2-lags	1,14	0,63	0,80	Ja	17.01.14	
511-8	C	Patus+	Træ/Alu	Energi	26 mm	-31,3	1,33	0,45	0,73	12,7	2-lags	1,16	0,62	0,80	Ja	17.01.14	
511-9	B	Patus+B	Træ/Alu	Lavenergi	25 mm	-15,0	1,15	0,45	0,73	12,9	3-lags	0,91	0,62	0,73	Ja	17.01.14	
Skontoplan																	
530-2	A	Schüco AWS 75.SI	Alu/Alu	Lavenergi	28 mm	+2,5	0,89	0,42	0,67	15,7	3-lags	0,55	0,63	0,73	Ja	03.04.14	
Spar Vinduer ApS																	
520-1	C	2-lags standard	Træ	Lavenergi	25 mm	-31,2	1,34	0,46	0,75	12,2	2-lags	1,13	0,61	0,79	Nej	17.01.14	
520-2	C	2-lags Plus	Træ	Lavenergi	25 mm	-19,2	1,37	0,53	0,75	12,8	2-lags	1,20	0,71	0,81			
520-3	B	3-lags Clima	Træ	Lavenergi	25 mm	-15,4	0,98	0,37	0,76	12,5	3-lags	0,72	0,49	0,70			
520-4	A	3-lags Clima Plus	Træ	Lavenergi	25 mm	+1,9	0,97	0,46	0,76	13,0	3-lags	0,74	0,60	0,74			
520-5	C	2-lags Sapelli Mahogni	Hårdttræ	Lavenergi	25 mm	-31,7	1,41	0,49	0,75	12,1	2-lags	1,13	0,65	0,79			
520-6	C	3-lags Sapelli Mahogni	Hårdttræ	Lavenergi	25 mm	-20,8	1,04	0,37	0,76	12,4	3-lags	0,72	0,49	0,70			
520-7	C	2-lags Træ/Alu standard	Træ/Alu	Lavenergi	25 mm	-33,0	1,36	0,46	0,75	11,3	2-lags	1,13	0,61	0,79			

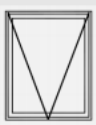
Samlet positivliste over Energimærknings-certifikater


Energimærkningscertifikat for facadevinduer
Reg. nr. 511-2 ver. 3

For hvert enkelt vindue/produktsystem er der et link til en yderligere detaljering af informationen.

Energidata for produktsystem

Referencevindue - definition:
1-fløjet vindue med oplukkelig ramme i den europæiske standardstørrelse 1,23 x 1,48 m



Eref	= 196,40 · gw - 90,36 · Uw	+2,7 kWh/m²
U_w	: U-værdi (vindue)	0,79 W/m²K
g_w	: Solenergitransmittans (vindue)	0,38
F_f	: Glasandel (Ag/Aw)	0,74

Standardrude for produktsystem

Standardrude - definition:
Ruden som anvendes som produktsystemets primære rude.

3 lags rude: 4-20-4-20-4

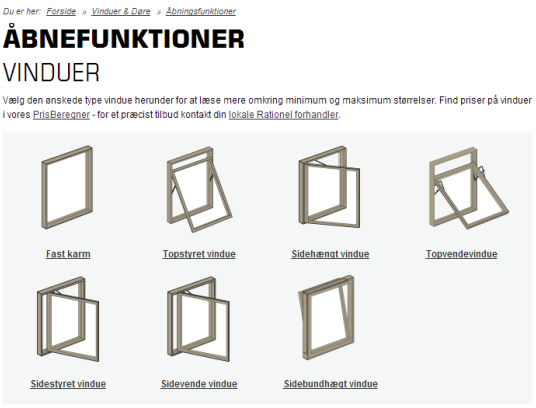
U_g	: Center U-værdi (rude)	0,52 W/m²K
g_g	: Solenergitransmittans (rude)	0,51
LT_g	: Lystransmittans (rude)	0,72
λ_{eq}	: Ækv.varmeledningsevne (spacer)	0,40/0,35 W/mK

Figur 8. Vindue - Certificerede energiværdier

Udover de værdier man kan hente fra positivlisten er det nødvendigt at kende en række oplysninger som ikke fremgår af positivlisten. Det er således ikke et krav at linjetabet og U værdier for karm, ramme, poste og sprosser oplyses. Disse oplysninger skal leveres af fabrikanterne.

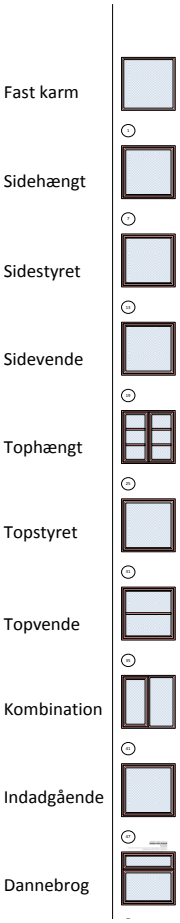
Fabrikanter

Udover at vinduer kan leveres af forskellige fabrikanter, som typisk hver leverer vinduer i forskellige produktsystemer (Træ, Træ/alu, plastik) kan et vindue leveres i mange forskellige udformninger. For at arealerne, som indgår i energibalancen kan beregnes er det nødvendigt at montøren udover størrelsen på vinduet angiver præcist hvordan det skal se ud.



Figur 9. Vindue - åbnefunktioner

Indenfor det enkelte produktsystem leverer producenterne således en række modeller afhængig af vinduets åbnefunktion og antallet af ruder. Figuren fra producenten Rationel vinduer viser en inddeling i en række modeller, som kan vælges for produktsystemerne DOMUS og ALDUS, baseret på vinduet åbnefunktion.

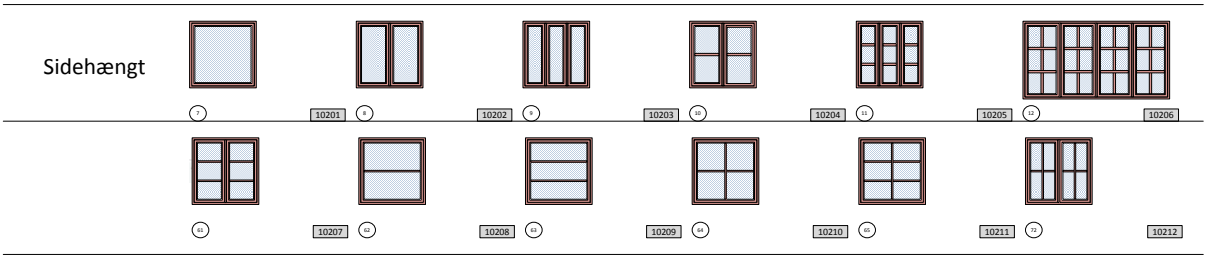


Figur 11. Vindue - modeller

Udover dette leverer Rationel vinduer som kan åbnes indad, dannebrogsvinduer og kombinations vinduer. I figuren til venstre er vist de forskellige modeller som kan leveres af Rationel.

Indenfor den enkelte model kan vinduet herefter leveres i en række versioner afhængig af antallet af poste og sprosser. Poste kan enten være horisontale eller vertikale. De inddeler vinduet i en række ruder. Sprosser kan være i form af energisprosser, som påsættes uden på ruden, og gennemgående sprosser som inddeler vinduet i flere ruder.

Figurerne 10 og 11 viser at modellen sidehængt vindue hos Rationel, som standard kan leveres i 12 forskellige versioner. I version 1 består vinduet af en enkelt oplukkelig rude. I version 2 og er placeret henholdsvis en og 2 horisontale poste som inddeler vinduet i 2 og 3 oplukkelige ruder (2 og 3 fags vinduer). I version 4 er der udover en vertikal post, en horisontal post i hvert fag, således at vinduet opdeles i 4 ruder. I flere af de andre versioner er det angivet at de sidehængte vinduer leveres med både vertikale og horisontale sprosser.



Figur 10. Vindue - versioner

Karme, rammer, poste og sprosser leveres i forskellig størrelse af de forskellige producenter. For at der kan beregnes et areal er det derfor nødvendigt at disse mål kendes for alle producenterne, herunder hvis der kan vælges forskellige størrelser hos den enkelte producent.

Herunder er beskrevet hvilke informationer som er nødvendige og de dimensioner som leveres af producenten Outline. Jeg har medtaget de navne jeg i forbindelse med design/implementering har anvendt for de forskellige variable så det under designet ikke er nødvendigt at vise figurerne igen.

KARM

Alle vinduer har en karm (figur 12). Denne karm er altid 50 mm hos Outline. I vinduesmodellen placeres denne information i tabellen 40_tbl_Manufacturer.

WINDOW_FRAME_HEIGTH

Ses vinduet udefra er rammen opdelt i en

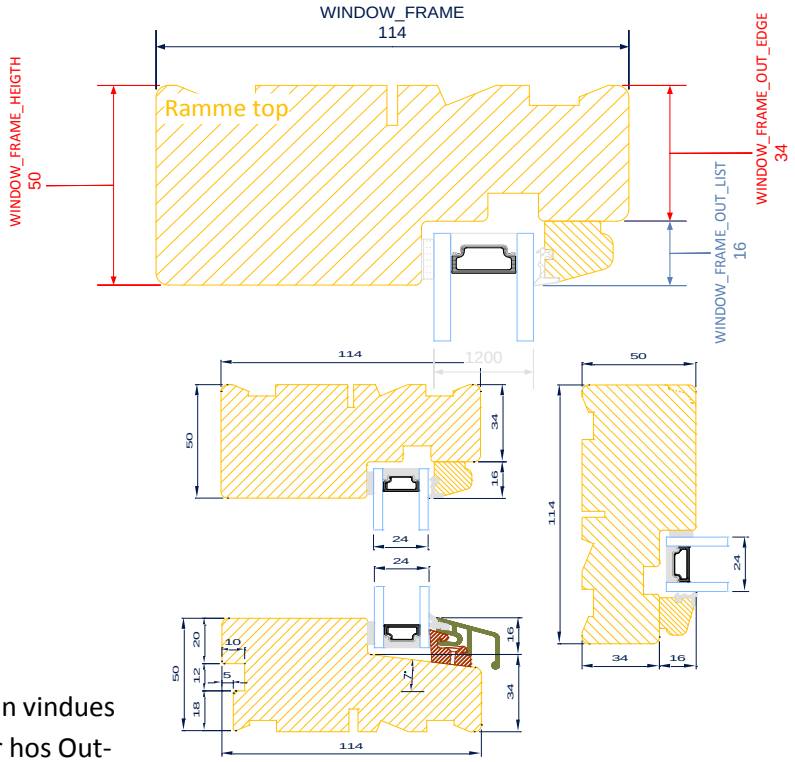
ydre kant **WINDOW_FRAME_OUT_EDGE**

og en

listekant **WINDOW_FRAME_OUT_LIST**.

WINDOW_FRAME_OUT_EDGE = 34 mm og **WINDOW_FRAME_OUT_LIST** = 16 mm placeres I 40_tbl_Manufacturer.

Tykkelsen **WINDOW_FRAME** er afhængig af hvilken vindues model som vælges. For den enkelte model kan der hos Outline vælges mellem 3 forskellige tykkelser. **WINDOW_FRAME** ligger I 31_tbl_Window_Model.



Figur 12. Vindue - karm

POSTE

I vinduet kan der være foretaget en inddeling af ruden i flere elementer. Denne inddeling foretages ved hjælp af poster som kan være horisontale eller vertikale.

Poster er altid 60 mm hos outline. Denne information placeres i 40_tbl_Manufacturer

WINDOW_POST_HEIGTH = 60 mm.

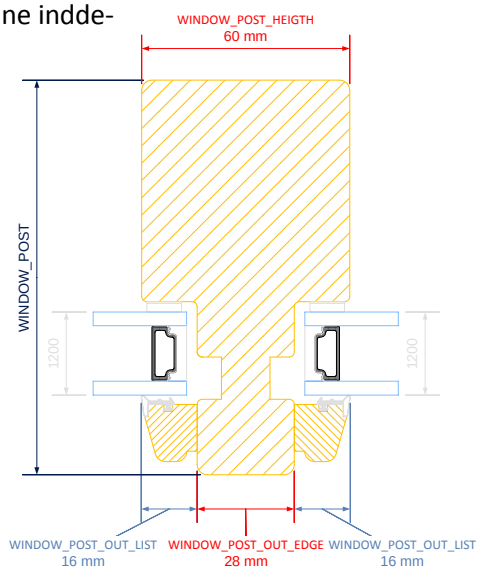
Ses vinduet udefra er posten opdelt i et

midterstykke **WINDOW_POST_OUT_EDGE** = 28 mm

og

2 listekanter **WINDOW_ POST _OUT_LIST** = 16 mm

Informationen placeres I 40_tbl_Manufacturer . Tykkelsen **WINDOW_ POST** kan vælges i forbindelse med hvilken vindues model som vælges.(For den enkelte model kan der hos Outline vælges mellem 3 forskellige tykkelser). **WINDOW_ POST** ligger I 31_tbl_Window_Model.



Figur 13. Vindue - post

VINDUESPANELER

Alle vinduer som kan åbnes er forsynet med ramme/panel omkring glasset (figur 14). Denne er altid 56 mm hos Outline. I vinduesmodellen placeres denne information i tabellen 40_tbl_Manufacturer.

WINDOW_PANE_HEIGTH = 56 mm

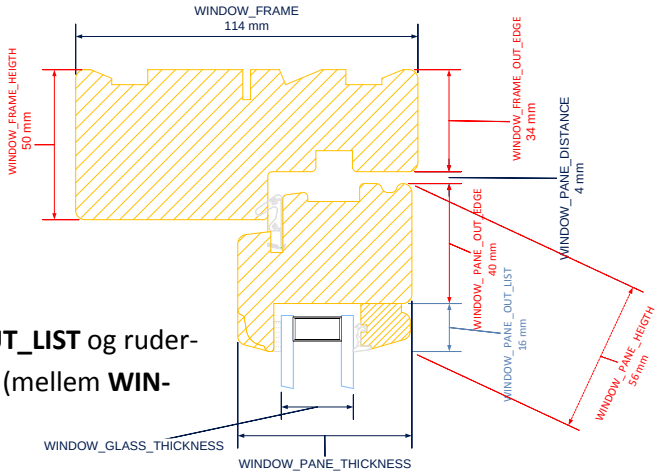
Ses vinduet udefra er ruderammen rammen opdelt i en

ydre kant **WINDOW_ PANE _OUT_EDGE** = 40 mm

og en

listekant **WINDOW_ PANE _OUT_LIST** = 16 mm

Når vinduet skal kunne åbnes fjernes **WINDOW_FRAME_OUT_LIST** og ruderammen placeres i vinduesrammen med en afstand på 4 mm (mellem **WINDOW_FRAME_OUT_EDGE** og **WINDOW_PANE_HEIGTH)**



Figur 14. Vindue - panel

WINDOW_PANE_DISTANCE = 4 mm

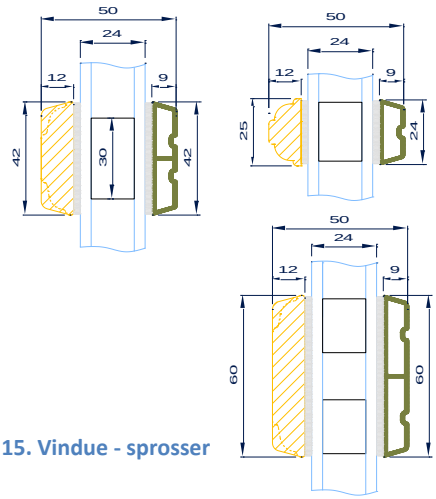
WINDOW_ PANE _OUT_EDGE, WINDOW_ PANE _OUT_LIST og **WINDOW_PANE_DISTANCE** placeres I 40_tbl_Manufacturer.

Tykkelserne **WINDOW_GLASS_THICKNESS** og **WINDOW_PANE THICKNESS** er afhængige af den valgte vinduesmodel. De ligger i 31_tbl_Window_Model.

SPROSSER

Ruden kan inddeles synsmæssig ved hjælp af sprosser (figur 15). Der findes både horisontale og vertikale sprosser. Sprosser er en del af rudens konstruktion.

I forbindelse med at montøren vælger vindues model og version angives automatisk antallet af sprosser som leveres i standard versionen. Sprosser leveres kun i en energi version med udvendigt aluminium i bredderne 24, 42 og 60 mm.



Figur 15. Vindue - sprosser

PLACERING AF POSTER OG SPROSSER.

For poster og sprosser er der angivet en standard placering i forbindelse med valg af vinduesmodel og version. Den præcise placering er afhængig af vinduets højde og bredde. Nedenstående tabel angiver sammenhæng mellem model, version, poster og sprosser for producenten Outline.

Åben funktion	Retning	Placering	Version	Horisontale poster	Vertikale poster	Horisontale sprosser	Vertikale sprosser	Version ID
Ingen			Fast karm	Ingen	Ingen	Ingen	Ingen	FK_01
				1	Ingen	Ingen	Ingen	FK_02
				1	1	Ingen	Ingen	FK_03
				1	Ingen	2	2	FK_04
				Ingen	Ingen	1	1	FK_05

				Ingen	Ingen	2	2	FK_06
En	Udadgående	Side	Sidehængt	Ingen	Ingen	Ingen	Ingen	SH_01
				Ingen	1	Ingen	Ingen	SH_02
				Ingen	2	Ingen	Ingen	SH_03
				Ingen	1	1	Ingen	SH_04
				Ingen	2	2	Ingen	SH_05
				Ingen	3	2	1	SH_06
			Sidestyret	Ingen	Ingen	Ingen	Ingen	SS_01
				Ingen	2	Ingen	Ingen	SS_02
				Ingen	2	Ingen	Ingen	SS_03
				Ingen	2	Ingen	Ingen	SS_04
				1	Ingen	Ingen	Ingen	SS_05
				2	2	Ingen	Ingen	SS_06
			Sidevende	Ingen	Ingen	Ingen	Ingen	SV_01
				Ingen	2	Ingen	Ingen	SV_02
				Ingen	2	Ingen	Ingen	SV_03
				Ingen	2	Ingen	Ingen	SV_04
				1	Ingen	Ingen	Ingen	SV_05
				3	2	Ingen	Ingen	SV_06
		Top	Tophængt	Ingen	1	2	Ingen	TH_01
				Ingen	Ingen	Ingen	Ingen	TH_02
				Ingen	1	Ingen	Ingen	TH_03
				Ingen	2	Ingen	Ingen	TH_04
				Ingen	3	Ingen	Ingen	TH_05
			Topstyret	Ingen	1	1	Ingen	TS_01
				Ingen	Ingen	Ingen	Ingen	TS_02
				Ingen	1	Ingen	Ingen	TS_03
				Ingen	2	Ingen	Ingen	TS_04
				Ingen	3	Ingen	Ingen	TS_05
			Topvende	Ingen	Ingen	1	Ingen	TV_01
				Ingen	2	Ingen	Ingen	TV_02
				Ingen	Ingen	Ingen	Ingen	TV_03
				Ingen	1	Ingen	Ingen	TV_04
				Ingen	2	Ingen	Ingen	TV_05
				Ingen	3	Ingen	Ingen	TV_06
		Kombination	Kombination	Ingen	1	Ingen	Ingen	KOM_01
				1	Ingen	Ingen	Ingen	KOM_02
				1	1	Ingen	Ingen	KOM_03
				2	1	Ingen	Ingen	KOM_04
				2	Ingen	Ingen	Ingen	KOM_05
				2	2	Ingen	Ingen	KOM_06
	Indadgående		Indadgående	Ingen	Ingen	Ingen	Ingen	IND_01
				Ingen	1	Ingen	Ingen	IND_02
				Ingen	2	Ingen	Ingen	IND_03
				Ingen	Ingen	Ingen	Ingen	IND_04
				Ingen	1	Ingen	Ingen	IND_05
				Ingen	Ingen	Ingen	Ingen	IND_06
To	Udadgående		Dannebrog	1	Ingen	Ingen	Ingen	DAN_01
				1	1	Ingen	Ingen	DAN_02
				3	2	Ingen	Ingen	DAN_03
				1	Ingen	2	2	DAN_04
				1	1	3	2	DAN_05
				3	2	2	4	DAN_06

Tabel 2. Outline - poste og sprosser

U-VÆRDIER OG LINIETAB

Udover at det skal være oplyst fra fabrikanterne hvilke vinduer det er muligt at vælge og deres dimensioner så de forskellige arealer som indgår i energiberegning kan beregnes, skal det fra fabrikanten oplyses hvad de forskellige valg betyder for vinduets U-værdier. Hvis der f.eks. kan vælges rammer med forskellig tykkelse skal U-værdien for hver type oplyses. Kan der vælges forskellige glastyper skal de nødvendige værdier som indgår i energiberegning (solindstråling, isolering og lysindfald) oplyses.

Montør

UDFORMNING AF DE ENKELTE VINDUER

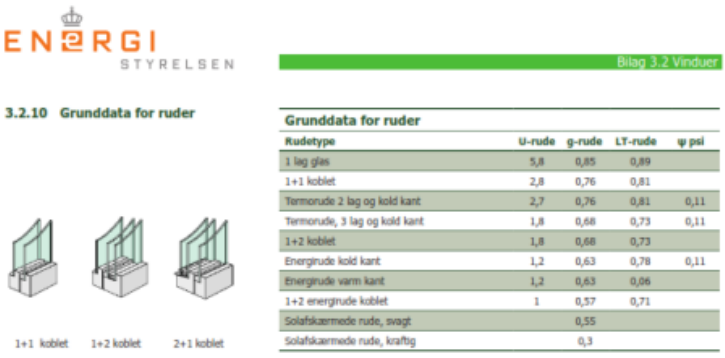
Det skal være muligt for montøren at angive præcist hvordan de enkelte vinduer skal produceres. Han vælger fabrikant, model og version for hvert enkelt vindue. I forhold til standardvinduerne skal han ikke alene kunne angive højde og bredde, men også placering af poste og sprosser såfremt ønskerne til et vindue afviger fra standarden.

ORIENTERING

Montøren skal levere oplysning om i hvilke af parcelhusets vægge vinduet er placeret og dermed orientering i forhold til solen.

DE EKSISTERENDE VINDUER

I projektet er det besluttet at de nye og de eksisterende vinduers udformning er identisk. Typisk vil det være vanskeligt at afgøre fabrikatet af de eksisterende vinduer. Både derfor og fordi vinduerne er af ældre dato vil det ikke være muligt at få præcise værdier som kan anvendes i beregningen af deres energibalance. Firmaet har besluttet at anvende de referenceværdier som oplyses i Bilag 5 "Bilag til håndbog for energikonsulenter". Her er vinduerne inddelt i 6 grupper på baggrund af hvordan de er udformet. Indenfor hver gruppe, er det herudover nødvendigt at montøren afgør hvilken type rude der er isat.



ENERGI STYRELSEN

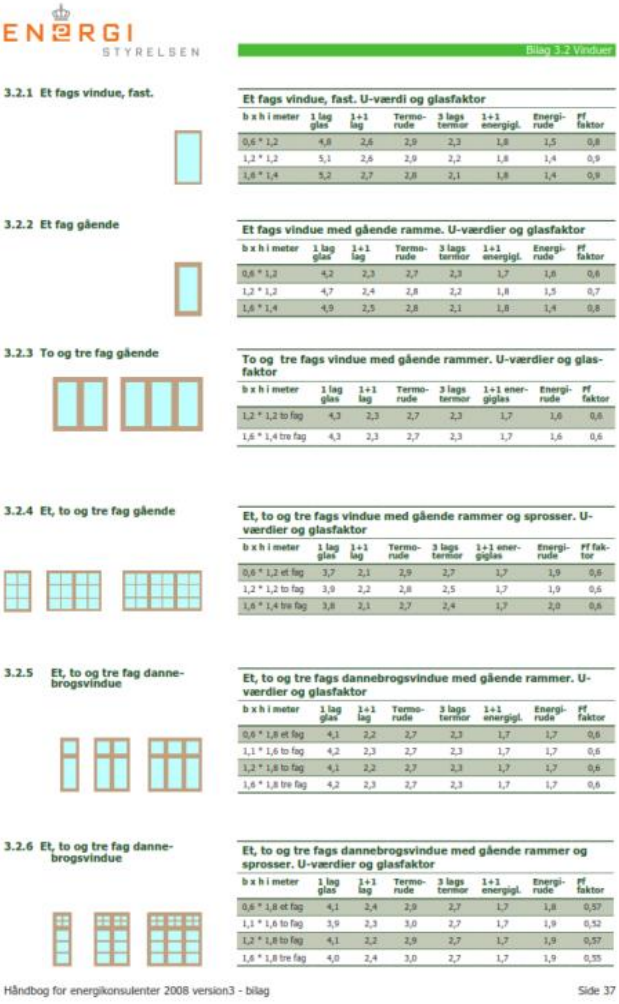
Bilag 3.2 Vinduer

3.2.10 Grunddata for ruder

1+1 koblet 1+2 koblet 2+1 koblet

Grunddata for ruder				
Rudetype	U-rude	g-rude	LT-rude	ψ psi
1 lag glas	5,8	0,85	0,89	
1+1 koblet	2,8	0,76	0,81	
Termorude 2 lag og kold kant	2,7	0,76	0,81	0,11
Termorude, 3 lag og kold kant	1,8	0,68	0,73	0,11
1+2 koblet	1,8	0,68	0,73	
Energirude kold kant	1,2	0,63	0,78	0,11
Energirude varm kant	1,2	0,63	0,06	
1+2 energirude koblet	1	0,57	0,71	
Solafskærmede rude, svagt		0,55		
Solafskærmede rude, kraftig		0,3		

Figur 16. Vindue - energiværdier for eksisterende vinduer



ENERGI STYRELSEN

Bilag 3.2 Vinduer

3.2.1 Et fags vindue, fast.

Et fags vindue, fast. U-værdi og glasfaktor								
b x h i meter	1 lag glas	1+1 lag	Termo-rude	3 lags termor	1+1 energigl.	Energi-rude	Pf faktor
0,6 * 1,2	4,8	2,6	2,9	2,3	1,8	1,5	0,8
1,2 * 1,2	5,1	2,6	2,9	2,2	1,8	1,4	0,9
1,6 * 1,4	5,2	2,7	2,8	2,1	1,8	1,4	0,9

3.2.2 Et fag gående

Et fags vindue med gående ramme. U-værdier og glasfaktor							
b x h i meter	1 lag glas	1+1 lag	Termo-rude	3 lags termor	1+1 energigl.	Energi-rude	Pf faktor
0,6 * 1,2	4,2	2,3	2,7	2,3	1,7	1,6	0,6
1,2 * 1,2	4,7	2,4	2,8	2,2	1,8	1,5	0,7
1,6 * 1,4	4,9	2,5	2,8	2,1	1,8	1,4	0,8

3.2.3 To og tre fag gående

To og tre fags vindue med gående rammer. U-værdier og glasfaktor							
b x h i meter	1 lag glas	1+1 lag	Termo-rude	3 lags termor	1+1 ener-giglas	Energi-rude	Pf faktor
1,2 * 1,2 to fag	4,3	2,3	2,7	2,3	1,7	1,6	0,6
1,6 * 1,4 tre fag	4,3	2,3	2,7	2,3	1,7	1,6	0,6

3.2.4 Et, to og tre fag gående

Et, to og tre fags vindue med gående rammer og sprosser. U-værdier og glasfaktor							
b x h i meter	1 lag glas	1+1 lag	Termo-rude	3 lags termor	1+1 ener-giglas	Energi-rude	Pf fak-tor
0,6 * 1,2 et fag	3,7	2,1	2,9	2,7	1,7	1,9	0,6
1,2 * 1,2 to fag	3,9	2,2	2,8	2,5	1,7	1,9	0,6
1,6 * 1,4 tre fag	3,8	2,1	2,7	2,4	1,7	2,0	0,6

3.2.5 Et, to og tre fag dannebrogsvindue

Et, to og tre fags dannebrogsvindue med gående rammer. U-værdier og glasfaktor							
b x h i meter	1 lag glas	1+1 lag	Termo-rude	3 lags termor	1+1 energigl.	Energi-rude	Pf faktor
0,6 * 1,8 et fag	4,1	2,2	2,7	2,3	1,7	1,7	0,6
1,1 * 1,6 to fag	4,2	2,3	2,7	2,3	1,7	1,7	0,6
1,2 * 1,8 to fag	4,1	2,2	2,7	2,3	1,7	1,7	0,6
1,6 * 1,8 tre fag	4,2	2,3	2,7	2,3	1,7	1,7	0,6

3.2.6 Et, to og tre fag dannebrogsvindue

Et, to og tre fags dannebrogsvindue med gående rammer og sprosser. U-værdier og glasfaktor							
b x h i meter	1 lag glas	1+1 lag	Termo-rude	3 lags termor	1+1 energigl.	Energi-rude	Pf faktor
0,6 * 1,8 et fag	4,1	2,4	2,9	2,7	1,7	1,8	0,57
1,1 * 1,6 to fag	3,9	2,3	3,0	2,7	1,7	1,9	0,52
1,2 * 1,8 to fag	4,1	2,2	2,9	2,7	1,7	1,9	0,57
1,6 * 1,8 tre fag	4,0	2,4	3,0	2,7	1,7	1,9	0,55

Håndbog for energikonsulenter 2008 version3 - bilag Side 37

Andet indhold

Under designet af webapplikationen skal der tages hensyn til at applikationen skal videreudvikles til et større sammenhængende produkt, som beregner de økonomiske konsekvenser ved at foretage en optimering af den eksisterende boligmasses energiforbrug.

MONTØRFORMULAR

Montørformularen skal designes med henblik på at de øvrige informationer, som montøren skal levere, kan tilføjes efterfølgende. Da vinduerne alle er placeret i en væg og da vinduerne alle identificeres ud fra den væg de er placeret, herunder deres orientering er det besluttet at det i projektet skal være muligt for montører at levere informationer om husets vægge.

BRUGERFORMULAR

Oplysninger fra brugerformularen anvendes til at identificere de forskellige parcelhuse. Brugerformularen indeholder desuden oplysninger som tillader at den potentielt mulige energireduktion for huset vurderes. Montørgennemgangen af huset som er gratis for kunden vil normalt tage flere timer. Såfremt det ikke er sandsynligt, at der vil kunne opnås et fordelagtigt resultat for kunden og dermed en opgave for firmaet vil montørgennemgang af bygning ikke blive tilbudt. F.eks. et stort hus med et meget lavt energiforbrug.

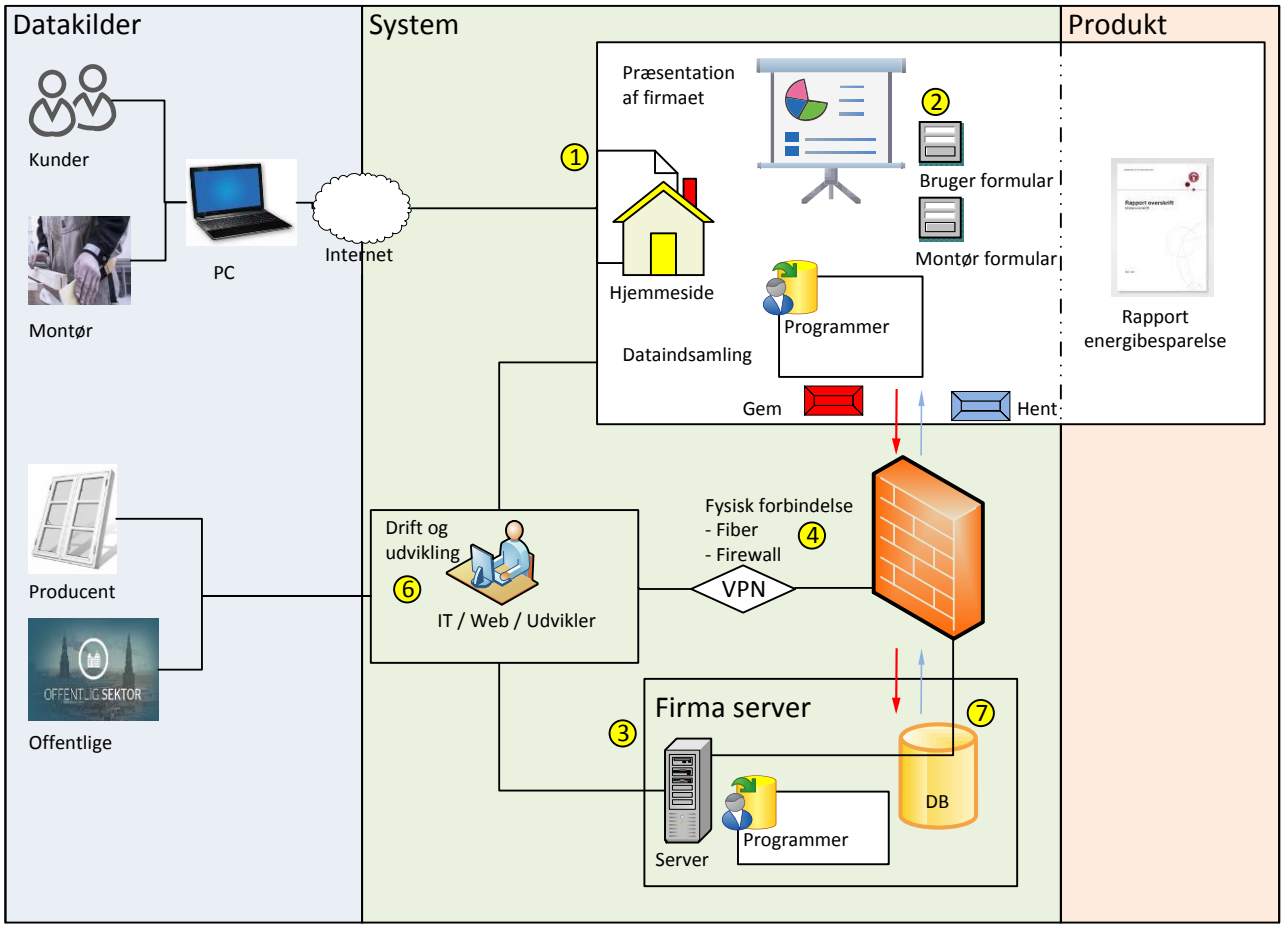
Brugerformularen skal således indeholde kontaktoplysninger til kunden, informationer om husets konstruktion og om familiens energiforbrug.

System

Ud fra forretningsgang og indhold kan der beskrives følgende entiteter som skal håndteres af projektet:

1. Hjemmeside
2. Indtastningsformularer
3. System
    a. Server
    b. Forbindelser, firewall
    c. Udviklingsmiljø og administrationsmiljø
    d. Programmer
    e. Database

Sammenhængen er vist i figur 17



Figur 17. Projekt - entiteter

Hjemmeside

I projektet skal der alene etableres en hjemmeside med en kort præsentation af firmaet og med link til indtastningsformularerne.

Bruger og montørformular

Jeg har valgt at beskrive indholdet af disse i forbindelse med beskrivelsen af design af formularerne.

Udviklingsmiljø, programmeringssprog og database

Der var for kunden side et ønske om at systemet blev udviklet i ASP.NET frameworket. Jeg havde som udgangspunkt forstillet mig at udvikle systemet i Django. Django er et udviklingsmiljø til Python, som jeg har arbejdet med det før og at det er et meget effektivt udviklingsmiljø til indtastningsflader. Udover at Django er godt til indtastningsformularer, er Python et godt sprog til databehandling da det kompiler og kører med stor hastighed. Så valget lå mellem de to udviklingsmiljøer.

ASP.NET med C# som sprog blev dels valgt som følge af at det var det firmaet ønskede, også på den tilgængelige dokumentationen af miljøet. Da Django er et forholdsvist nyt udviklingsmiljø og open source kan dokumentationen mangle eller være meget vanskelig at finde, hvorimod der ligger næsten endeløse mængder dokumentation vedrørende .NET. Selve sproget C# blev også valgt på baggrund af dokumentationen, og fordi det ligger tæt op af JAVA og Python, som jeg har arbejdet med før.

Efter valg af udviklingsmiljø var valg af teksteditor, database og server ligetil. Jeg har valgt Visual Studio som tekst editor, som er produceret til .NET. Som database fald valget på MSSQL, som også fungerer godt sammen med .NET og som ligger tæt op af mySQL som vi har arbejdet med på DTU.

Firmaet har en server, men var ikke sikker på om det kunne betale sig at beholde den og man var ikke tilfreds med styresystemet. Serveren kørte Ubuntu, den var ikke blevet vedligeholdt, hvilket blandt andet betød at boot-partitionen var fyldt op med gamle opdateringer, som ikke var blevet kørt. Det at boot-partitionen var fyldt, resulterede i at man ikke kunne opdatere styresystemet, da boot-partitionen bruges til dette. Jeg kunne ikke få lov til at udvide partitionen da jeg ikke havde en ”magic key” (firmaet viste ikke hvad det var). Så der skulle ske noget med serveren før man kunne få den til at virke.

Da firmaet ikke var sikker på om de ville beholde serveren blev jeg nødt til at foretage en server analyse for at finde ud af om det kunne betale sig at beholde serveren eller om man skulle få webapplikationen hosted et andet sted.

Da man under alle omstændigheder skulle formatere serveren, og den skulle afvikle .NET applikation, var det oplagt at vælge Windows server som styresystem.

Serveranalyse

Herunder er liste en række forhold af betydning som man skal være opmærksom på når der skal vælges server.

1. Hvor meget kan serveren maksimal blive belastet på et givent tidspunkt
    a. Antal brugere der er logget ind samtidigt.
    b. Sender brugerne store mængder data frem og tilbage.
    c. I hvilket omfang skal der ske en behandling af brugerens data og hvor hurtigt skal den foregå.
2. Hvor meget data skal gemmes.

3. Sender brugerne meget data frem og tilbage på månedlig basis.
4. Sikkerhed
5. Har serverren det software man skal bruge.
6. Support.
7. Udgifter

Vedrørende punkt 1, 2 og 3 kunne vi hurtigt konkludere at den server firmaet havde, sagtens ville kunne håndtere dette selv hvis det endelige produkt blev en succes med en omfattende daglig aktivitet.

Med hensyn til sikkerheden er der både fordele og ulemper ved at få siden applikationen hosted eksternt. Fordelene ved webhosting er at der med stor sandsynlighed er en bedre firewall og at der er ansat som holder styr på sikkerheden. På den anden side er der sikkert også man flere der prøver på at "hacke" servere og det kan risikeres at en medarbejder som sælger firmaets data. Sikkerheden blev ikke vurderet som en betydende faktor ved det aktuelle valg af serverløsning.

Software var heller ikke anset som vigtig, da webhosting kan vælges så man kan få den software man gerne vil have, og på ens egen server kan man installere lige præcist hvad man har lyst til.

Supporten er væsentlig bedre hvis applikationen hostes eksternt, men da det er valgt at man gerne vil køre Windows server og en forholdsvis ligetil .Net applikation, er det begrænset hvor meget support man kan få behov for.

Derfor blev valget af serverløsning hovedsageligt baseret udgifterne til de to muligheder. Hvad kostede det at have serveren kørende, hvad kunne man få for at sælge serveren og hvor meget kostede det at siden blev hostet eksternt.

Der er flere omkostninger til at have en server kørende, herunder en god internet forbindelse og en firewall. Disse udgifter ville firmaet imidlertid have alligevel. Den eneste udgift der betød noget var hvor meget serveren brugte i strøm. Jeg satte en energi måler på som viste at serveren kostede dem 318 kr at have kørende om måneden.

Jeg lavede herefter en værdisætning af serveren for at prøve at finde ud af hvad de ville kunne få for den(se bilag). En forsigtig vurdering var i området af 60.000 kr.

Til sidste fandt jeg 2 forslag til eksterne udbydere hvor man kunne få webhotel og de ønskede faciliteter.

Webhotel	Hostnordic A/S	hostforlife.eu
Pris DKK/måned	4.550.00	3.000
Lagerplads (HD) GB	20	1000
RAM GB	4	4
Support	24/7	24/7
link	http://www.hostnordic.com/da-DK/Web-Hosting/Windows-Server/Windows-Server-Priser-og-varianter.aspx	http://hostforlife.eu/European-Windows-Cloud-Server-Hosting-Plans

Tabel 3. Servervalg

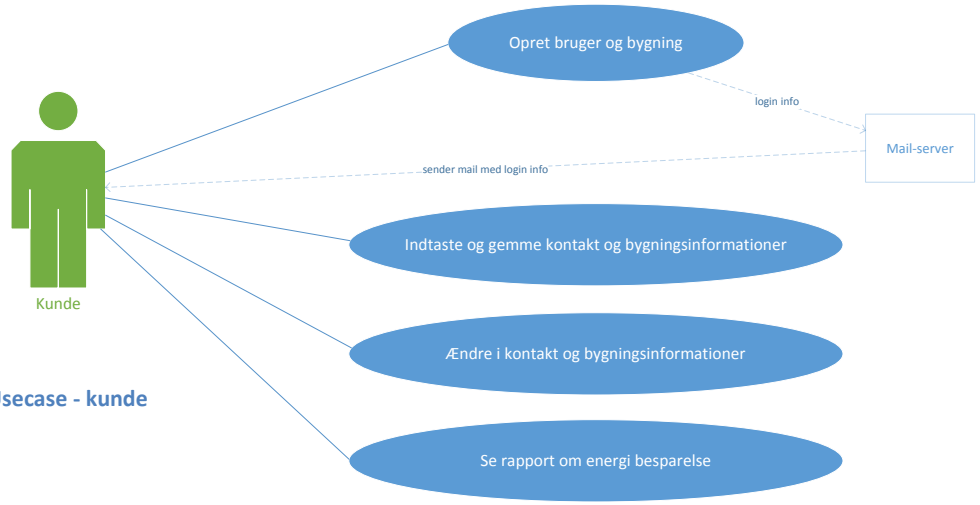
Webhotellerne ville man kunne bruge i nogen tid med mindre at firmaet voksede meget hurtigt.

Konklusion var at det kunne betale sig at beholde serveren. Man ville få en væsentlig bedre server og udgifterne til webhosting ville allerede efter 2 år overstige beløb det man sparede ved at slukke serveren + det beløb serveren kunne indbringe ved et salg.

Usecase

Usecases beskriver brugernes behov i forbindelse med anvendelse af webapplikationen. Der er lavet et usecasediagram for en kunde og en tabel som mere detaljeret beskriver kundens behov og der er lavet et diagram og en tabel for montøren.

Kunde



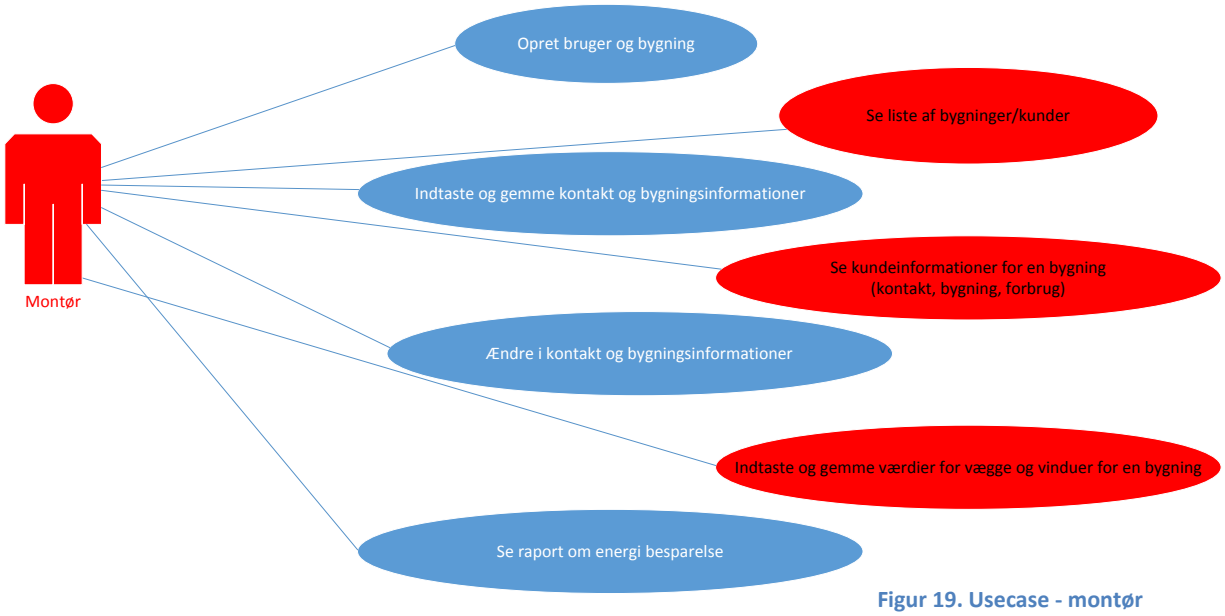
Figur 18. Usecase - kunde

Use Case UC1: Kunde registrer kontakt og parcelhusinformationer

Scope:	Indtastning i webapplikationens brugerformular
Niveau:	User-goal
Primær aktør:	Kunde
Interessenter og interesser:	• Kunden skal have en nem, overskuelig og intuitiv adgang til at levere de nødvendige oplysninger. • Montøren (medarbejderen/samarbejdspartneren) skal let kunne vælge og se information om de forskellige kunder • Firmaet (administrator) skal kunne vælge og overskue de forskellige kunder og deres bygninger således at kunderne kan kontaktes og det kan vurderes for hvilke kunder der er et væsentligt potentiale for at spare energi ved en bygningsrenovering.
Præ - forudsætninger:	Kunden har fundet firmaets webside på sin pc. Brugeren vælger at oprette en bygning. I forbindelse med projektet generer systemet automatisk et brugernavn og password.
Post - forudsætninger:	Kunden har leveret de nødvendige kontakt og bygningsinformationer. Disse er gemt på i databasen på serveren. Både kunde, montør og administrator kan genfinde og se informationerne. Alle vil i forbindelse med det aktuelle projekt have mulighed for at rette de indtastede informationer.
Vigtigste succes senarier:	• Kunden har via firmaet hjemmeside valgt at kontakte firmaet med henblik på en mulig renovering af kundens parcelhus. • Kunden ser en side hvor der kan indtastes kontakt oplysninger, bygnings og forbrugsinformationer. • Kunden indtaster informationerne • Kunden gemmer informationerne på serveren • Kunden/montøren/firmaet kan hente informationerne • Kunden/montøren/firmaet kan ændre og gemme ændringerne • Kunden skal kunne se en rapport hvor den potentielle energibesparelse er beregnet
Specielle krav:	Kunden har forbindelse til internettet. Firmaet har en hjemmeside. Hjemmesiden har forbindelse til serveren.

Tabel 4. Usecase - kunde

Montør



Figur 19. Usecase - montør

UseCase UC2: Kunde registrer kontakt og parcelhusinformationer

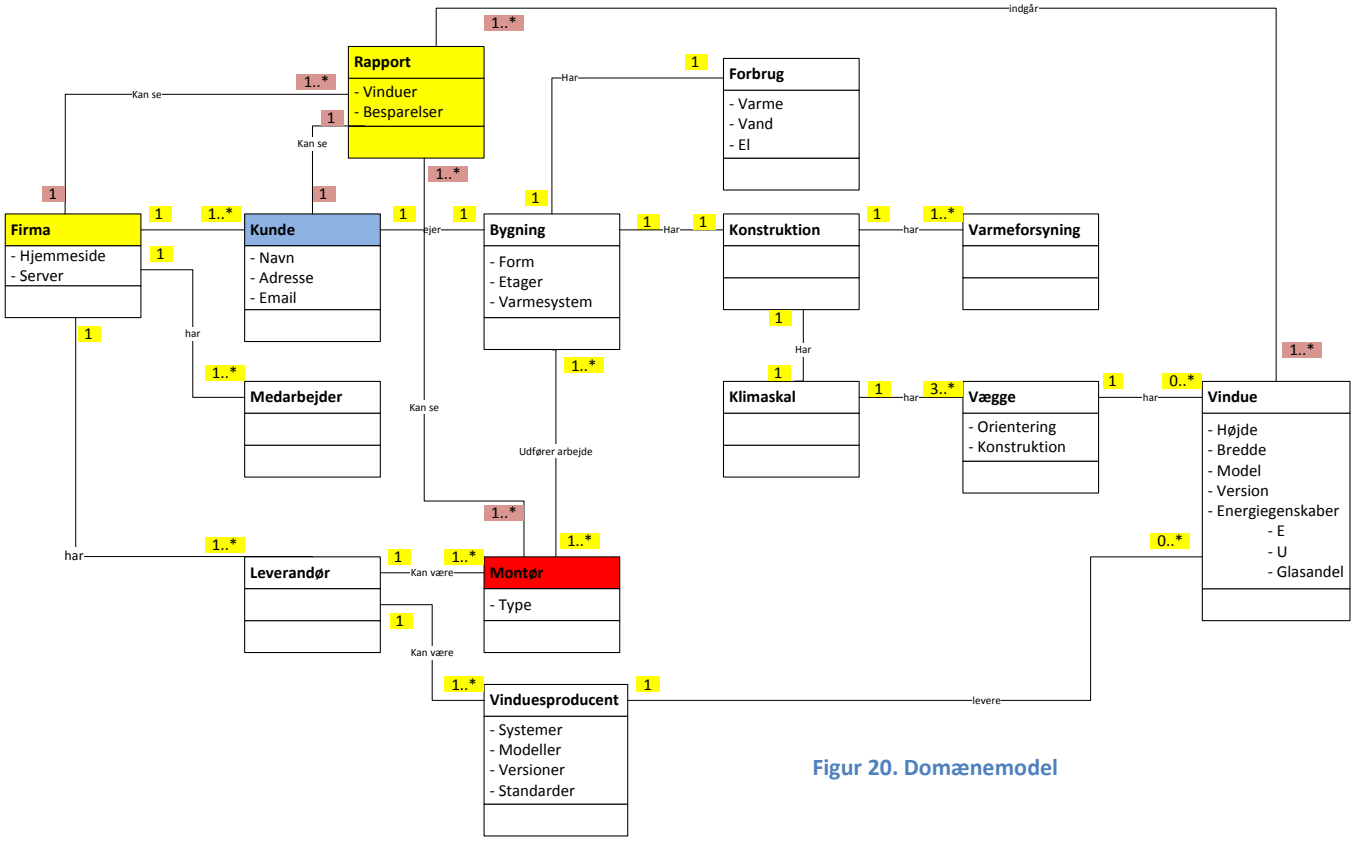
Scope:	Indtastning i webapplikationens montørformular
Niveau:	User-goal
Primær aktør:	Montør
Interessenter og interesser:	• Montøren skal have let kunne vælge og se information om de forskellige kunder • Montøren skal kunne hjælpe kunden med at indtaste kunde og bygningsinformationer • Montøren skal kunne ændre kundeinformationer • Montøren skal kunne indtaste informationer om vinduer som skal udskiftes (og de vægge vinduerne sidder i) • Montøren skal kunne gemme, hente og ændre informationer om vinduer. • Firmaet (administrator) skal kunne vælge og overskue alle informationer leveret af montøren. • Montøren/firmaet skal kunne se en rapport hvor den potentielle energibesparelse er beregnet.
Præ - forudsætninger:	Montøren har fundet kunden via firmaets webside på sin bærbare pc.
Post - forudsætninger:	Montøren har leveret de nødvendige vinduesinformationer. Disse er gemt på i databasen på serveren. Både montør og administrator kan genfinde og se informationerne og de vil begge have mulighed for at rette de indtastede informationer. Kunde, montør og administrator kan en rapport over den potentielle energibesparelse.
Vigtigste succes senarier:	Der foreligger en rapport over den potentielle energibesparelse
Specielle krav:	Montøren har forbindelse til internettet/hjemmesiden på sin bærbare pc. Hjemmesiden har forbindelse til serveren.

Tabel 5. Usecase - montør

Domænemodel

Ud fra forretningsmodel, de værdier som er nødvendige for at foretage en energiberegning og usecases er nedenstående domænemodel udformet (figur 20). Den beskriver behovene i projektet på et overordnet niveau, hvor der fokuseres på stuktur og ikke detaljer. For hver entitet kan beskrives attributter og sammenhæng. I figuren er der angivet eksempler på væsentlige attributter.

Figuren er heller ikke fuldstændig, men indeholder kun hovedentiteterne og relationerne mellem dem. F.eks. skal der også indgå data fra det offentlige, både for nye vinduer og for de eksisterende vinduer.



Figur 20. Domænemodel

Delkonklusion

Analysen har identificeret de informationer som skal tilvejebringes for at foretage en beregning af energibesparelsen ved at udskifte vinduer i et parcelhus. Baseret på forretningsgangen, usecases og informationer om energibesparelsesberegningen har jeg udformet en domænemodel som kan anvendes som basis for design af webapplikationen i designfasen.

Der skal etableres en hjemmeside, hvorfra webapplikationen kan aktiveres. Den skal indeholde indtastningsformularer fra kunder og montører (datakilder) og der skal kunne vise en rapport over energibesparelsen (resultat/produkt). Webapplikationen skal kommunikere med firmaets server hvor beregningerne foretages. Systemet baseres på objekt teknologi (ASP.NET med C# som code behind) som frontend, og relations teknologi (MSSQL) som backend.

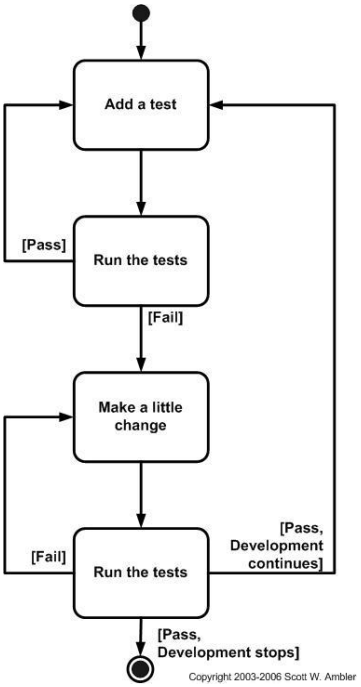
5. DESIGN

I designfasen valgte jeg at bruge domænemodellen som guide. En tilgang inspireret af "Agile/Evolutionary Data Modeling: From Domain Modeling to Physical Modeling" af Scott W. Ambler: (http://agiledata.org/essays/agileDataModeling.html#InitialDomainModel). I samarbejde med firmaet/kunden blev projektet inddelt i en række afgrænsede opgaver. Sammen med firmaet blev der angivet ønsker til indhold og defineret krav til de enkelte opgaver og der blev opstillet en målsætning for hver opgave.

I praksis foregik arbejdet med hver opgave som et mix mellem design og implementering i en TDD (Test – driven development) proces (figur 21), hvor jeg gennem en række iterationer fik indarbejdet krav, UI relaterede ideer, forretningsregler og strukturel information.

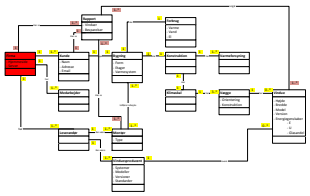
Designet blev inddelt i følgende opgaver:

- Firma
- Kunde
- Vindue
- Montør
- Rapport



Figur 21. TDD - Test Driven Development

Firma



MÅLSÆTNING:

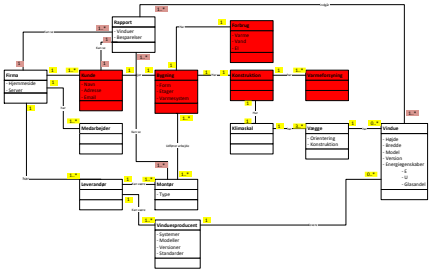
Serveren fungerer og webapplikationen kan hentes på internettet og bruges af kunderne til at levere kundeinformation og til at energibesparelsen ved at udskifte vinduer kan demonstreres.

KRAV:

Det var initialt firmaets ønske at hjemmesiden indeholdt en præsentation af virksomheden og hvad den kunne tilbyde. På grund af projektets omfang blev dette reduceret til ovenstående målsætning.

Opsætning af server og webapplikation er beskrevet under implementering.

Kunde



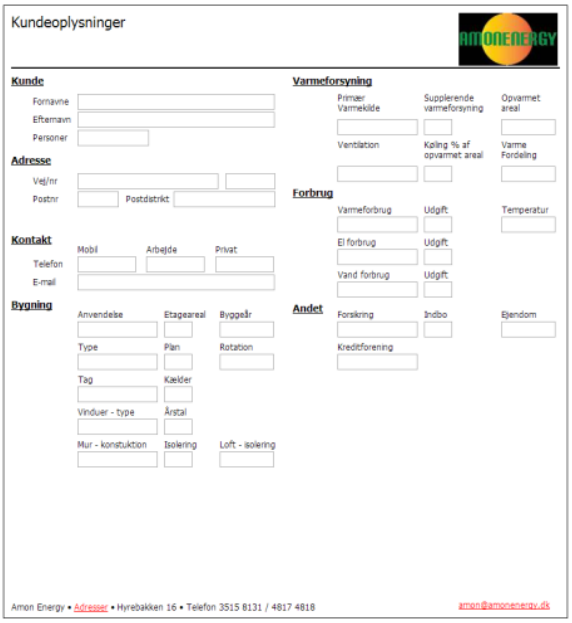
MÅLSÆTNING:

Brugerformularen kan åbnes i webapplikationen. De af firmaet ønskede informationer kan indtastes i en formular som er overskuelig og brugervenlig.

KRAV:

Der skal kunne leveres informationer om kundekontakt, bygningen og forbruget.

Det er firmaets ønske at der i brugerformularen i projektet som udgangspunkt medtages alle de informationer firmaet ønsker leveret fra kunden, uagtet at dette ikke er nødvendigt for at gennemføre projektet. Som første step udarbejdes figur 22



Figur 22. Kundeoplysninger - initialt

Med henblik på at indtastningsfladen kan baseres på et enkelt skærmbillede uden at scrolle besluttes det at udforme indtastningsfladen med 3 horisontale faneblade:

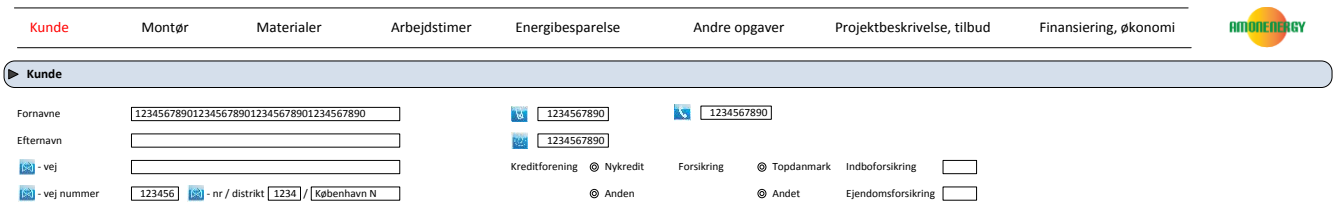
- Kunde
- Bygning
- Varme

Det er firmaets ønske at det ud fra de af kunden leverede oplysninger er muligt at vurdere om det er realistisk at der kan opnås en besparelse på husstandens varmeforbrug og at man kan lave et groft estimat af størrelsen på besparelsen. Udover dette ønskes oplysninger om forsikring og kreditforening da firmaet ved at indgå aftale med et forsikringsselskab og en kreditforening.

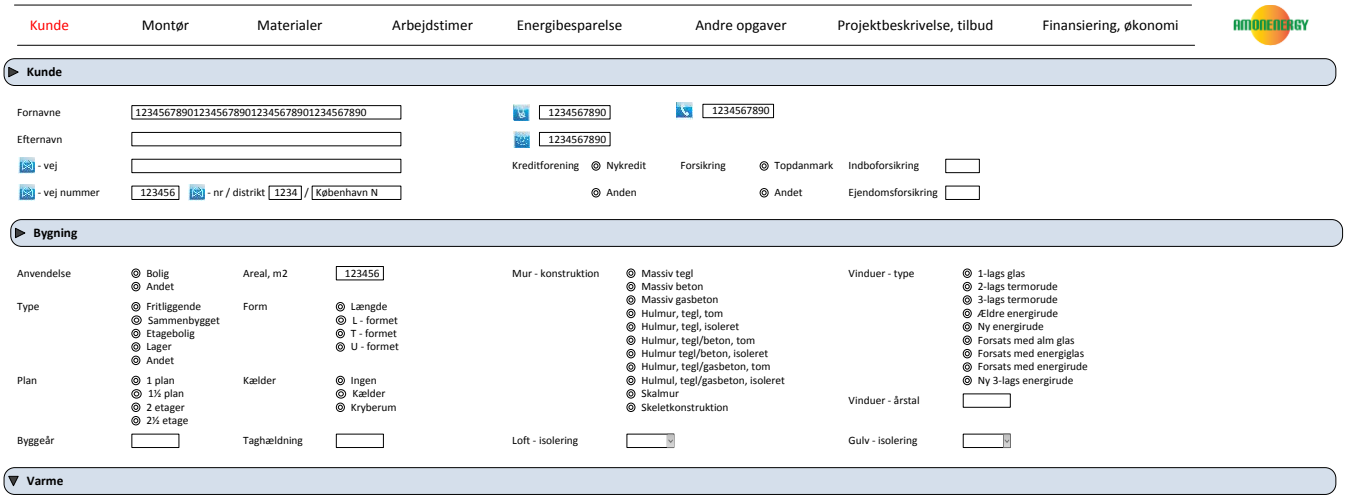
Jeg besluttede at organisere informationerne i 2 tabeller - CUSTOMER og CUS_BUILDING. Indholdet af disse og deres relationer er beskrevet under design af databasemodellen.

Herefter udformede jeg nedenstående design dokumenter i Visio som blev gennemgået med firmaet, således at jeg sikrede at det svarede til forventningerne og kravene til indhold. Firmaet havde enkelte kommentarer og i forbindelse med implementering afveg man fra enkelte elementer, herunder blandt andet at det ikke var hensigtsmæssigt/nødvendigt at firmaets kunde både opgav varmeforbrug(mængde) og udgift til varme.

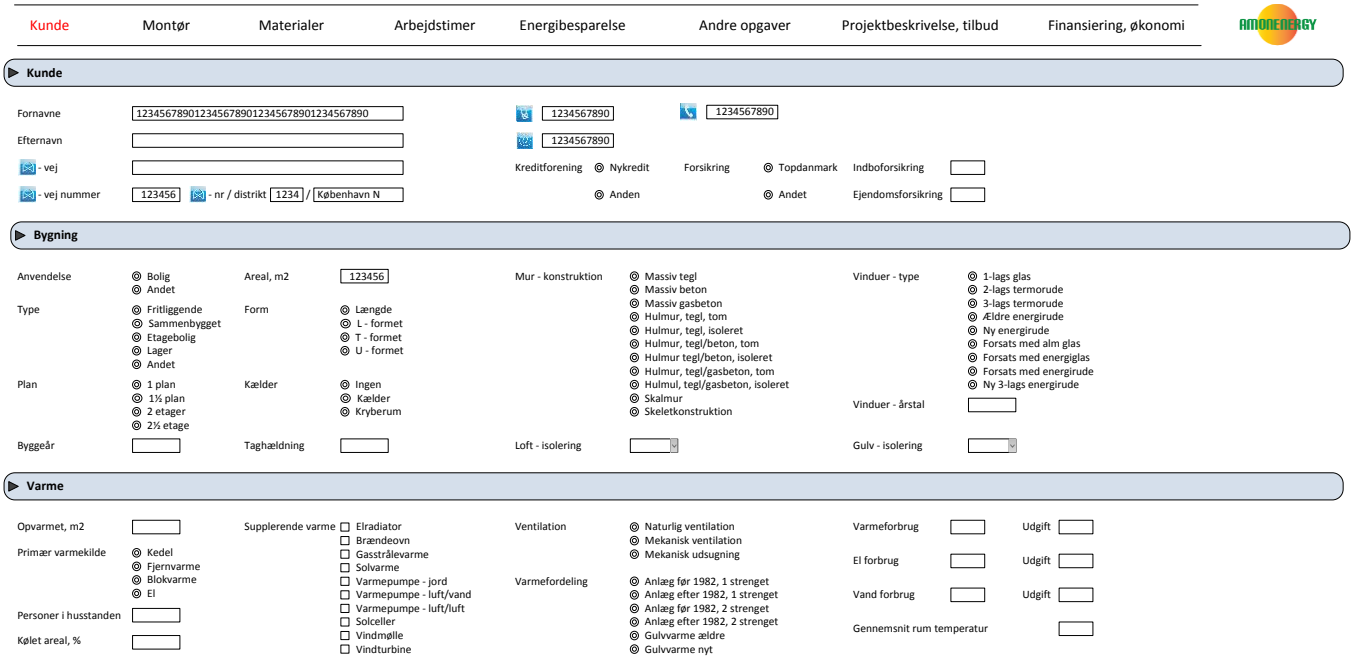
Kunde 1



Kunde 2

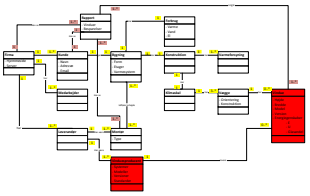


Kunde 3



Figur 23. Kundeoplysninger - designfase

Vindue



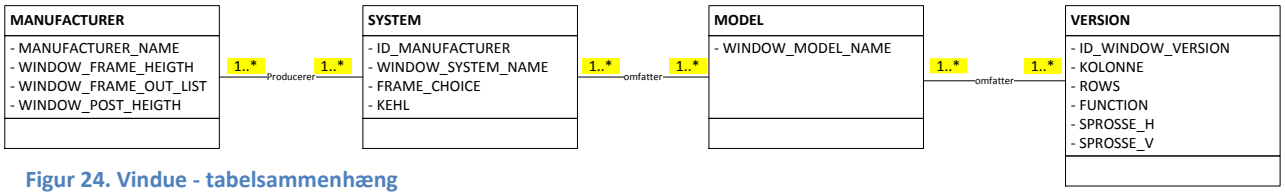
MÅLSÆTNING:

Montøren kan vælge alle typer tilgængelige vinduer og registrere de informationer som er nødvendige for at vinduerne kan bestilles hos producenten. Montøren kan registrere de informationer om de eksisterende vinduer som er nødvendige for at energibesparelsen ved udskiftning af det enkelte vindue kan beregnes.

KRAV:

Ved projektets start var der kun indgået aftale med en enkelt vinduesfabrikant. Som følge af dette var der kun leveret informationer vedrørende vinduernes energivariable for denne producent. I projektet skulle det imidlertid være muligt, at der efterfølgende kunne tilføjes yderligere vinduesproducenter. Herudover skulle der i en vist omfang tages hensyn til webapplikationen senere skulle udvides. Informationerne om de eksisterende vinduer skulle udarbejdes i henhold til kravene i "Håndbog for Energikonsulenter" (bilag 5).

Valg af vindue er baseret på følgende sammenhæng (figur 24):



Figur 24. Vindue - tabelsammenhæng

Vinduerne kan leveres fra flere forskellige producenter (MANUFACTURER). Hver fabrikant producerer vinduer i forhold til en række for dem specifikke dimensioner. Således vil rammehøjde (WINDOW_ FRAME_ HEIGTH) kun produceres i f.eks. 114 mm eller 173 mm og posthøjden (WINDOW_POST_HEIGTH) kun kunne leveres på 60 mm. I det omfang der er valgmuligheder skal det være muligt for montøren at angive disse valg.

Fabrikanterne producerer et eller flere produktsystemer (Træ, Træ/Aluminium, plastik) i en eller flere modeller som montøren skal kunne vælge. Det er de enkelte produktsystemer som er certificerede. Indenfor modellerne, er der en eller flere versioner, hvor det for de enkelte versioner yderligere skal være muligt at angive afvigelser fra standardmålene.

I projektet er det besluttet at de nye vinduer i forbindelse med beregningen af energibesparelsen skal være af samme udseende som de eksisterende vinduer.

Jeg har valgt at beskrive design af vinduer med det dokument jeg i dialog med firmaet udarbejdede som udgangspunkt for implementeringen. Her blev indtastnings flowet og de nødvendig variable identificeret. Der er anvendt foreløbige variabelnavne, som i en del tilfælde er blevet ændret og organiseringen af informationen i tabeller er heller ikke korrekt. Korrekte tabeller, navne og relationer beskrives under datamodellen.

Vinduer

Udskiftning

Under det horisontale faneblad vinduer svares først på om der skal ske en udskiftning

Variabel - Udskiftning (WALL_WINDOW_REPLACEMENT)

01.	Vinduer udskiftes	variable vedrørende udskiftning af vinduer
02.	Glas udskiftes	variable vedrørende udskiftning af glas åbnes

03. Kombination For hver enkelt vindue kan det angives om vinduet eller glasset skiftes
04. Ingen Ingen variable åbnes.

WALL_WINDOW_REPLACEMENT er placeret i tabellen 03_tbl_WALL. Alle øvrige informationer om udskiftning af vinduer eller glas placeres i tabellen 06_tbl_Window.

Alle vinduer har en id og er relateret til en væg som skal være placeret i 03_tbl_Window

03_tbl_Window

ID_WINDOW
FK_ID_WALL

Eksisterende vinduer

Når det er angivet at der skal udskiftes vinduer åbnes en tabel med samme antal rækker som der er vægge i bygningen. Fuldstændig på samme måde som for ydervægge. Eksempelvis har et længehus 4 vægge.

Væg 1	
Væg 2	
Væg 3	
Væg 4	

Hver række i tabellen svarer til et vindue. Da der kan sidde flere vinduer i en væg kan der tilføjes yderligere rækker under hver væg. Det første vinduesnummer i hver væg er tilføjet automatisk.

Væg nr	Vindue nr	
Væg 1	1.1	
Væg 2	2.1	
Væg 3	3.1	
Væg 4	4.1	

Når de første informationer for det første vindue er indtastet får man mulighed for at tilføje yderligere vinduer ved at trykke på det plustegn som vises.

Væg nr	Vindue nr	Information	
Væg 1	1.1	Vindue 1	+
Væg 2	2.1		
Væg 3	3.1		
Væg 4	4.1		

Ved tryk på + åbnes yderligere en række. Vinduet nummeres automatisk

Væg nr	Vindue nr	Information	
Væg 1	1.1	Vindue 1	+
	1.2		
Væg 2	2.1		
Væg 3	3.1		
Væg 4	4.1		

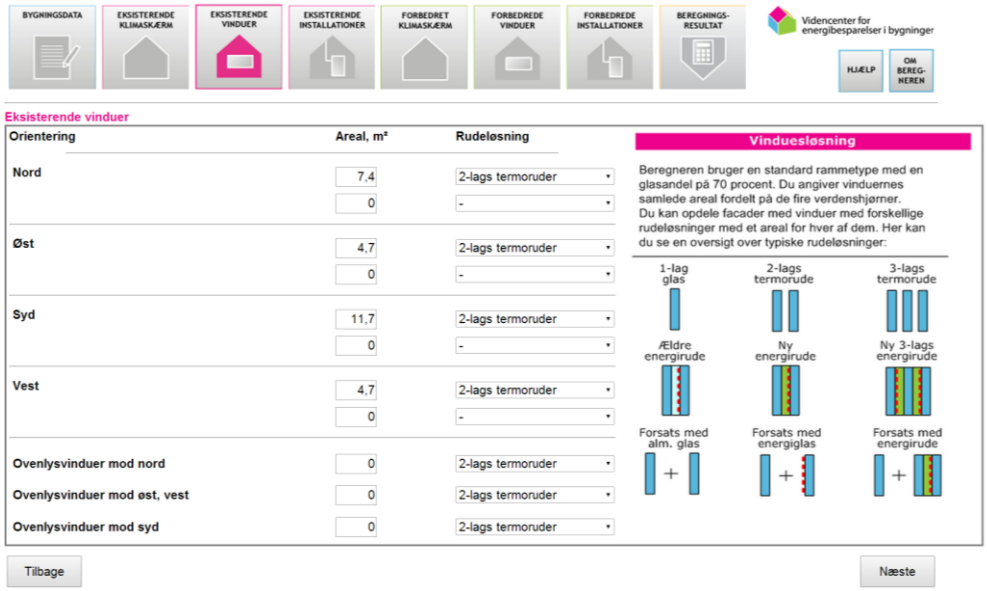
Det eksisterende vindue skal først angives = WINDOW_EXCISTING_ID. Vindues modellen ved opslag i tabel-

len 32_tbl_Window_Version gerne ved at der klikkes på en figur som ligner nedenstående figur 25.

Fast karm						
Sidehængt						
Sidestyret						
Topstyret						
Topvende						
Kombination						
Dannebrog						

Figur 25. Vindue - valg af eksisterende vindue

Dernæst angives hvilken rudeløsning der er anvendt = WINDOW_EXCISTING_GLAS. Her ville det igen være fint hvis man kunne vælge fra en figur som den der anvendes af Videncenter for energibesparelser i bygninger (figur 26).



Figur 26. Vindue - valg af eksisterende rude

Såfremt der skal udskiftes glas skal der tilføjes yderligere informationer om det eksisterende vindue, disse variable er ikke medtaget endnu. Hvis vinduet skal udskiftes er der ikke brug for yderligere informationer, i det højde og bredde anvendes som angivet for det nye vindue. Det er muligt at der ved vindues udskiftningen skiftes model og størrelse, men dette tilføjes som en ekstra opgave og indgår således ikke i beregningen af energibesparelsen.

De 2 variable tilføjes tabellen

Væg nr	Vindue nr	Eksisterende	Nyt vindue

		Model WINDOW_EXCISTING_ID	Rude WINDOW_EXCISTING_GLAS	
Væg 1	1.1	27	03	Vindue 1 +
	1.2			
Væg 2	2.1			
Væg 3	3.1			
Væg 4	4.1			

Tabellen indeholder nu følgende variabel

03_tbl_Window

ID_WINDOW
FK_ID_WALL
WINDOW_NUMBER
WINDOW_EXCISTING_ID
WINDOW_EXCISTING_GLAS

Nye vinduer

For alle nye vinduer skal der angives en højde og en længde. Disse svarer til murhullet med et minimums-fradrag på 11 mm på alle flere sider (mindste afstand findes).

WINDOW_HEIGTH, WINDOW_LENGTH

For hvert vindue skal der angives en type en producent, et system, en model og en version.

Producent og system vælges først, ved at vælge produktsystem fra tabellen med vinduescertifikater. Hvert produktsystem er unikt og identificerer samtidig producenten. Denne sammenhæng registreres i FK_ID_WINDOW_SYSTEM.

Dernæst vises en figur hvor produktsystemet forskellige modeller og versioner er vist.

Nedenstående figur er for produktsystemerne Træ/Alu og Træ/Alu Energy fra producenten Outline

Fast karm						
Sidehængt						
Sidestyret						
Sidevende	Nej					
Tophængt	Nej					
Topstyret						
Topvende						
Kombination						
Indadgående	Nej					
Dannebrog						

Figur 27. Vindue - valg af nyt vindue

Ved tryk på et af ikonerne registreres WINDOW_OUTLINE_ID som består af 5 karakterer.

Karakter 1 – SYSTEM (kan være 1 = Træ, 2 = Træ/Alu, 3 = Træ/Alu – Energy).
Karakter 2 og 3 - MODELLEN (kan være 01 = 01 Fast karm, 02 = 02 Sidehængt, 03 = 03 Sidestyret…)
Karakter 4 og 5 – VERSIONEN (kan være 01 = FK_01, 02 = FK_02, 03 = FK_03 ….)
F.eks.

WINDOW_OUTLINE_ID =	10101 =	Træ, Fast Karm, Fast Karm 01
WINDOW_OUTLINE_ID =	30101 =	Træ/Alu Energy, Fast Karm, Fast Karm 01
WINDOW_OUTLINE_ID =	21057 =	Træ/Alu, Dannebrog, Dannebrog 05

For alle nye vinduer skal der desuden altid registreres følgende variable

WINDOW_FRAME	= Rammestørrelse
WINDOW_KEHL	= +/- kehl
WINDOW_NOT_LYSNING	= +/- not i lysninger
WINDOW_NOT_BUND	= +/- not I bundpladen
WINDOW_COULOR_OUT	= udvendig farve
WINDOW_COULOR_IN	= indvendig farve

De to variable WINDOW_ACCESSORIES og WINDOW_GLAS_TYPE skal formentlig være check bokse. Afhængig af WINDOW_OUTLINE_ID vil der formentlig være forskel på hvad der kan vælges, men ellers skal de formentlig behandles på samme måde som de 6 andre variable.

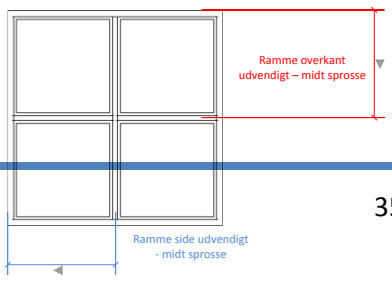
Der er herefter følgende variable i tabellen

03_tbl_Window

ID_WINDOW
FK_ID_WALL
WINDOW_NUMBER
WINDOW_EXCISTING_ID
WINDOW_EXCISTING_GLAS
WINDOW_HEIGTH
WINDOW_LENGTH
FK_ID_WINDOW_SYSTEM
WINDOW_OUTLINE_ID
WINDOW_FRAME
WINDOW_KEHL
WINDOW_NOT_LYSNING
WINDOW_NOT_BUND
WINDOW_COULOR_OUT
WINDOW_COULOR_IN
WINDOW_GLAS_TYPE
WINDOW_ACCESSORIES

Efter at de røde variable er angivet første gang (på det første vindue), sættes de efterfølgende vinduers variable til samme værdi (default). Hvis der for efterfølgende vinduer er forskel for et eller flere rettes til det korrekte.

Ved at angive WINDOW_OUTLINE_ID har man angivet hvilken version af vinduet man vil anvende. For hver version er det angivet om vinduet er forsynet med poster og sprosser og hvor mange. Montøren ind-



taster ikke i disse variable.

Når montøren har valgt en version vises en figur af vinduet (figur 28) dette er forsynet med en angivelse af hvilke mål der kan ændres. F.eks. hvis der er valgt Fast karm i versionen FK_04 vises figuren og

WINDOW_MEASURE_1

WINDOW_MEASURE_2

Figur 28. Vindue - mål som kan ændres

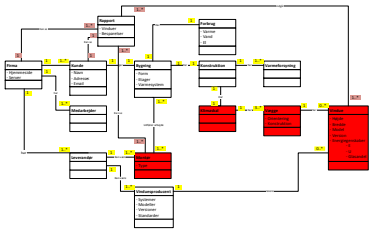
På figuren er det angivet hvad der er WINDOW_MEASURE_1 og WINDOW_MEASURE_2. Hvis montøren ikke indtaster værdier i de 2 variable beregnes placeringen af de 2 poster på baggrund af informationerne i POST_H_1 og POST_V_1 som standard og de beregnede værdier placeres i WINDOW_POST_H_1 og WINDOW_POST_V_1. Hvis montøreren indtaster værdier i WINDOW_MEASURE_1 og WINDOW_MEASURE_2 beregnes de modificerede placeringer af de 2 poster.

I de tilfælde hvor der anvendes sprosser i vinduet skal bredden af disse hvis der er mere end en bredde angives i variablen WINDOW_SPROSSE_WIDTH dropdown

Vi har herefter de nødvendige værdier for de nye vinduer.

03_tbl_Window
ID_WINDOW
FK_ID_WALL
WINDOW_NUMBER
WINDOW_EXCISTING_ID
WINDOW_EXCISTING_GLAS
WINDOW_HEIGTH
WINDOW_LENGTH
FK_ID_WINDOW_SYSTEM
WINDOW_OUTLINE_ID
WINDOW_FRAME
WINDOW_KEHL
WINDOW_NOT_LYSNING
WINDOW_NOT_BUND
WINDOW_COULOR_OUT
WINDOW_COULOR_IN
WINDOW_GLAS_TYPE
WINDOW_ACCESSORIES
WINDOW_MEASURE_1
WINDOW_MEASURE_2
WINDOW_MEASURE_3
WINDOW_MEASURE_4
WINDOW_SPROSSE_WIDTH

Montør



MÅLSÆTNING:

Montørformularen kan åbnes i webapplikationen. De af firmaet ønskede informationer kan indtastes i en formular som er overskuelig og brugervenlig. Alle værdier nødvendige for at kunne foretage en beregning af energibesparelsen er registrerede.

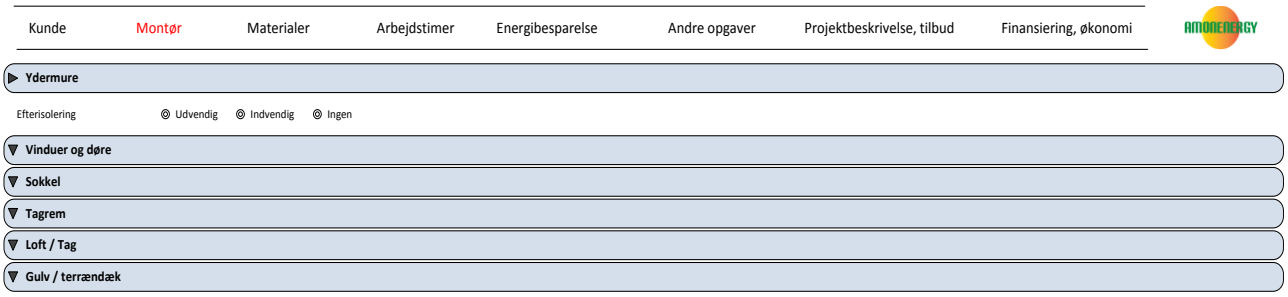
KRAV:

Der skal kunne vælges alle vinduer fra vinduesfabrikanten Outline som angivet i sidste afsnit - Vinduer. Herudover skal der tages hensyn til webapplikationen senere skal udvides. Da vinduerne alle er knyttet til en væg skal det i formularen være muligt at registrere information om vægge og vægisolering og bygningens form og orientering skal kunne registreres.

Som under design af vinduer udarbejdede jeg en beskrivelse af variable og indtastnings flowet udgangspunkt for implementeringen af vægge. Væggene er anvendes i beregningen af energibesparelsen alene til at afgøre vinduernes orientering. Jeg har af pladshensyn og da det egentlig ikke er en nødvendig del af projektet, valgt at vedhæfte dette dokument som bilag. Som for vinduer er anvendt foreløbige variabelnavne, som i en del tilfælde er blevet ændret og organiseringen af informationen i tabeller er heller ikke korrekt.

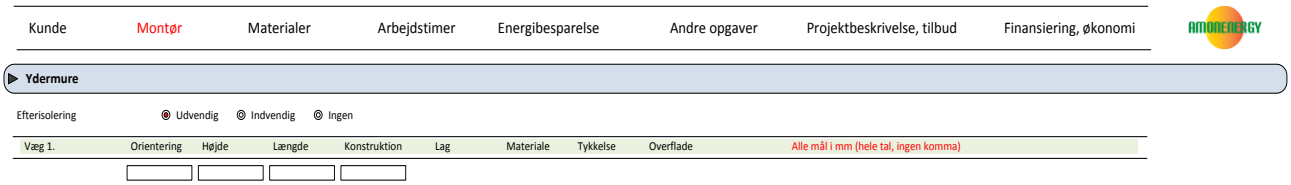
Nedenfor er vist design figurer (figur 29 - flere dele) udarbejdet for væggene. Vinduerne er tidligere beskrevet og den endelige indtastningsformular med vinduer vil blive vist i implementeringsafsnittet. Vinduer og vægge er en del af parcelhusets klimaskærm. I montørindtastningsformularen er de øvrige elementer i klimaskærmen medtaget som horisontale faneblade, men uden indhold.

Montør 1



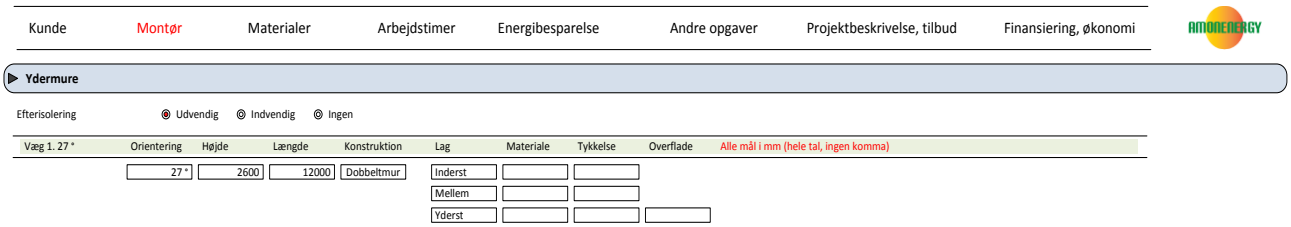
Hvis montøren angiver at der skal foretages en efterisolering

Montør 2



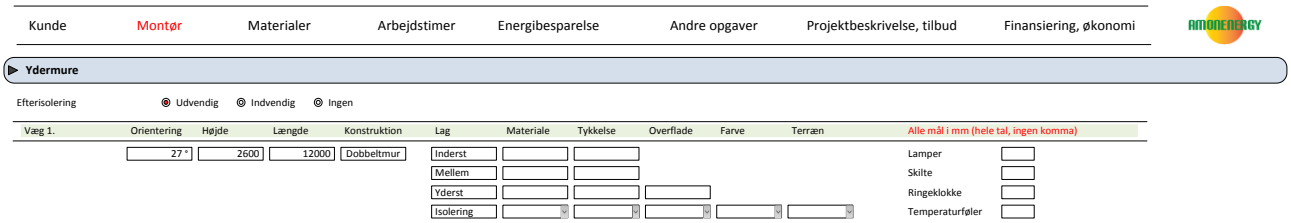
Kan der registreres en række data for væggen.

Montør 3 - A

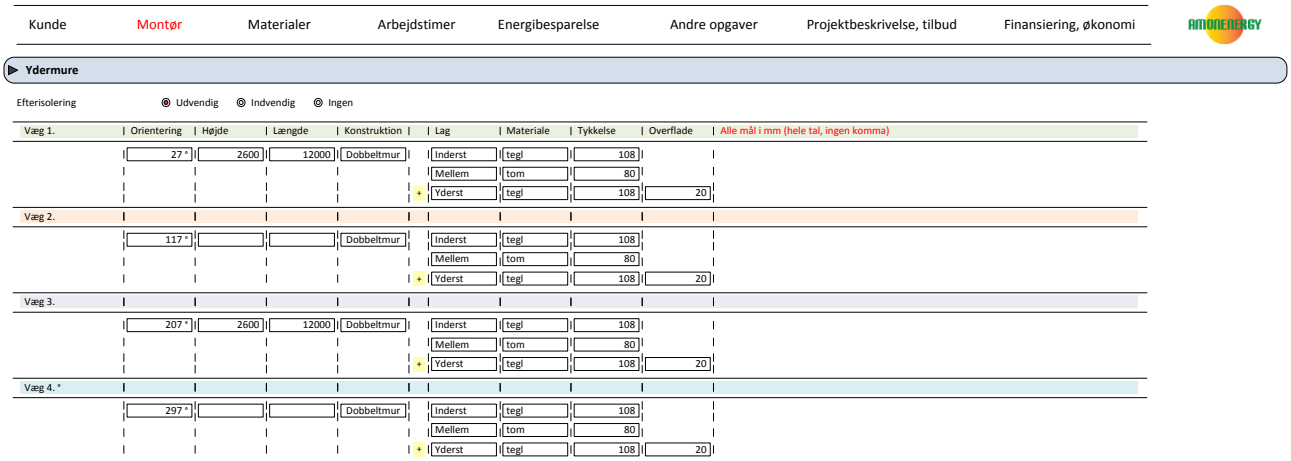


Afhængig af væggens opbygning kan der herefter beskrives op til 3 forskellige lag vedrørende den eksisterende konstruktion.

Montør 3 – B

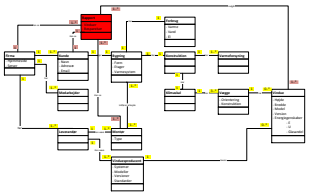


Montør 4



Figur 29. Montør - vægge i design

Rapport



MÅLSÆTNING:

Der vises en rapport hvor den samlede energibesparelse ved udskiftning af vinduerne i et parcelhus og energibesparelsen for de enkelte vinduer kan ses.

KRAV:

For hvert enkelt vindue skal følgende fremgå nummer, type og orientering, højde, bredde og system og energibalance før og efter og energibesparelse.

Tegning af vinduerne

Energiberegningen for et vindue baseres på

Energibalance = **solindstråling x vinduets energitransmittans** - **gradtimer x vinduets varmetab**

For at beregne den første del – vinduets energitransmittans – er det som tidligere beskrevet nødvendigt at beregne areal af glasset og areal af vinduet. For at beregne den anden del – vinduets isoleringsevne – skal areal af glasset, vindue, karm, ramme, poste og sprosser og omkredsen af glasarealet beregnes. Herudover skal man kende vinduets orientering, linjetabet for ruden og U værdier for glas, karm, ramme, poste og sprosser.

Det vil sige at energiberegningen er baseret på beregning af en række arealer og en omkreds og en række værdier som kan slås op i tabeller.

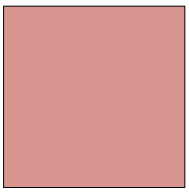
Da firmaet ønsker at energirapporten indeholder en tegning af hver enkelt vindue (ikke en del af projektet), har jeg besluttet at basere beregningen af energibalancen på en tegning af vinduet. Alle vinduer kan tegnes ved hjælp af en række bokse af forskellig størrelse der placeres korrekt i forhold til hinanden (x,y) i en række lag i den rigtige rækkefølge. I det følgende har jeg udeladt figur label af overskueligheds hensyn.

VINDUETS AREAL (MODEL)

Alle vinduer har en størrelse. For alle vinduer skal der altid laves følgende:

Variable	MODEL_AREA		WINDOW_HEIGTH * WINDOW_LENGTH	m^2
Bokse	SHAPE_1	Size	WINDOW_HEIGTH * WINDOW_LENGTH	mm
		Position	(x,y) = 0,0	Lag 1
		Color	WINDOW_COULOR_OUT (RGB 217,150,144)	

Det vil sige for et vindue med højden 1200 mm og bredden 1200 mm tegnes en boks i farven RGB 217,150,144 (rød).



Placeringen (x,y) = 0,0 er angivet ud fra nederste venstre hjørne. Boksen er placeret i det nederste lag (Lag 1).

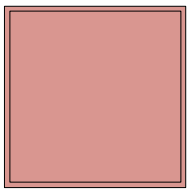
RAMME (RAMME)

Alle vinduer har en karm. Hos outline er karmen altid 50 mm (WINDOW_FRAME_HEIGTH) Denne er delt op i en ydre del på 34 mm og en listedel på 16 mm, disse er angivet i WINDOW_FRAME_OUT_EDGE og WINDOW_FRAME_OUT_LIST (figur). I det følgende benævnes WINDOW_FRAME_OUT_EDGE = WFOE og WINDOW_FRAME_OUT_LIST = WFOL. Vi markerer overgangen mellem den ydre del og listedelen ved at tilføje

yderligere en boks.

Bokse	SHAPE_2	Size	(WINDOW_HEIGTH – 2 * WFOE) * (WINDOW_LENGTH – 2 * WFOE)	mm
		Position	(x,y) = WFOE, WFOE	Lag 2
		Color	WINDOW_COULOR_OUT (RGB 217,150,144)	

Dette resulterer i følgende



Afhængig af hvilken model og version der er valgt for vinduet kan listedelen af rammen være til stede eller fjernet. Hvis den er til stede skal den have samme farve som den ydre del af rammen. Derfor skal SHAPE_2 altid være samme farve som SHAPE_1.

Placeringen (x,y) = WFOE,WFOE er angivet ud fra nederste venstre hjørne. For OUTLINE er WFOE = 34 mm, og x,y er således 34,34. Boksen er placeret i det det næstnederste lag (Lag 2).

Den resterende del af vinduet er afhængig af den valgte model og version. I tabellen er vist hvilke typer bokse som skal kunne tegnes og i hvilken rækkefølge de skal placeres for at alle typer vinduer kan tegnes

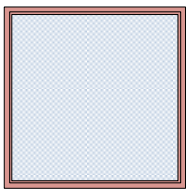
NAVN (SHAPE_NAME)	SHAPE_TYPE	LAG
Model	MODEL	1
Ramme	RAMME	2
Post – vertikal	POSTV	3
Post – horisontal	POSTH	4
Afstand	AFSTAND	5
Panel	PANEL	6
Rude	RUDE	7

I det følgende beskrives en række forskellige vinduer alle baseret samme ovenfor beskrevne MODEL og RAMME.

FAST KARM, 1 FAG, 1 RUDE, INGEN SPROSSER

Her er der kun en fast karm (vinduet kan ikke åbnes). Det eneste der mangler er derfor ruden.

Bokse	SHAPE_3	Size	(WINDOW_HEIGTH – 2 * (WFOE + WFOL)) * (WINDOW_LENGTH – 2 * (WFOE + WFOL))	mm
		Position	(x,y) = WINDOW_FRAME_HEIGTH, WINDOW_FRAME_HEIGTH	Lag 7
		Color	GLAS = RGB(213,223,235) (glas farve)	



Rudens er således vinduets højde – højden af rammen øverst og nederst. Højden af rammen er WFOE + WFOL (34 + 16 mm). Højde = 1200 – 2*50 = 1100 mm. Tilsvarende for bredden.

For vinduet kan de arealer som indgår i energiberegningen nu beregnes.

Total areal = 1200 mm * 1200 mm.
Glas areal = 1100 mm * 1100 mm
Karm areal = Total areal – glas areal.
Omkreds af glasarealet = 1100 mm * 4
Glasandel = Glas areal / Total areal

FAST KARM, 1 FAG, 2 RUDER, INGEN SPROSSER

Her skal der således indsættes en horisontal post som inddeler vinduet i 2 ruder. Hvor højt posten er placeret kan variere. Montøren kan angive højden f.eks. således at den øverste rude er mindre end den nederste. Han angiver dette i forbindelse med registreringen af vinduet i WINDOW_MEASURE_1.

Når vi skal lave boksen for posteren, startes vi med at undersøge WINDOW_MEASURE_1. Hvis den ikke er tom anvendes værdien som y værdi for shapens placering. Hvis WINDOW_MEASURE_1 er tom anvendes standardplacering af posteren. Denne standard findes i variablen WINDOW_POST_H_1 som er en attribut i tabellen over vindues versioner (tbl_VERSION) som skal være udfyldt for denne vinduesversion. Feks.

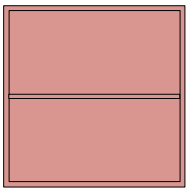
IF WINDOW_MEASURE_1 = 0 THEN WINDOW_MEASURE_1 = WINDOW_HEIGTH/2

Hvor WINDOW_POST_H_1 = WINDOW_HEIGTH/2

SHAPE_3	Size	WINDOW_POST_OUT_EDGE * (WINDOW_LENGTH – 2 * WFOE))	mm
	Position	(x,y) = WFOE, WINDOW_MEASURE_1 – (WINDOW_POST_OUT_EDGE/2)	Lag 4
	Color	WINDOW_COULOR_OUT (RGB 217,150,144)	

Posteren dannes

Og posten SHAPE_3 indsættes



Postens er således: Bredde = WINDOW_POST_OUT_EDGE = 28 mm. Længde = vinduets længe – 2 * ydre ramme bredde = 1132. Posten y placering er : 1200/2 = 600 – 14 = 586 mm. Det vil sige at midten af posten er placeret midt i vinduet.

Hernæst tegnes boksene for de 2 ruder.

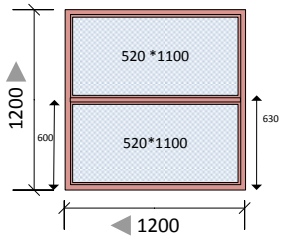
S_4	Size	(WINDOW_MEASURE_1-WFOE–WFOL–WFOL–(WINDOW_POST_OUT_EDGE/2)) * (WINDOW_LENGTH – 2*(WFOE+WFOL))	mm
	Position	(x,y) = WINDOW_FRAME_HEIGTH, WINDOW_FRAME_HEIGTH	L7
S_5	Size	(WINDOW_HEIGTH-WINDOW_MEASURE_1-WFOE–WFOL–WFOL–(WINDOW_POST_OUT_EDGE/2))*(WINDOW_LENGTH–2* (WFOE + WFOL))	mm
	Position	(x,y) = WINDOW_FRAME_HEIGTH, WINDOW_HEIGTH - WINDOW_MEASURE_1 + WPOL+WINDOW_POST_OUT_EDGE/2	L 7

Højden af den nederste rude er

1200/2 - 34 – 16 – 16 – 28/2 = 520 mm

Højden af det øverste er

1200 – 1200/2 - 34 – 16 – 16 – 28/2 = 520 mm



Y placeringen af det øverste vindue = 1200 – 600 + 16 + 28/2 = 630 mm

For vinduet kan de arealer som indgår i energiberegningen nu beregnes.

Total areal	=	1200 mm * 1200 mm
Glas areal	=	(520 mm * 1100 mm) * 2
Post areal		28 mm * 1132 mm
Karm areal	=	Total areal – glas areal – post areal
Omkreds af glasarealet	=	1100 mm * 4 + 520 mm *4
Glasandel	=	Glas areal/ Total areal

SIDEHÆNGT VINDUE, 2 FAG, INGEN SPROSSER

Der egnes et vindue på 1400 mm * 1400 mm. Her skal der således indsættes en vertikal post som inddeler vinduet i 2 ruder som kan åbnes. Igen kan posten placering variere. Montøren kan angive placeringen såle-

des at f.eks. den venstre rude er mindre end den højre. Igen angives dette i WINDOW_MEASURE_1.

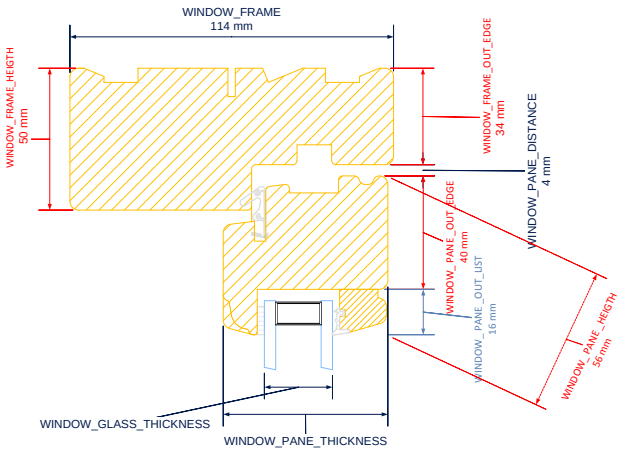
Når vi skal lave boksen for posteren, startes vi med at undersøge WINDOW_MEASURE_1. Hvis den ikke er tom anvendes værdien som x værdi for shapens placering. Hvis WINDOW_MEASURE_1 er tom anvendes standardplacering af posteren. Denne standard findes i variablen WINDOW_POST_V_1 som er en attribut i tabellen over vindues versioner (tbl_VERSION) som skal være udfyldt for denne vinduesversion. Feks.

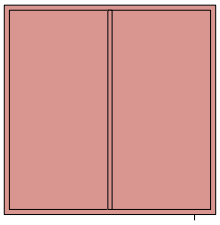
IF WINDOW_MEASURE_1 = 0 THEN WINDOW_MEASURE_1 = WINDOW_LENGTH/2

Hvor WINDOW_POST_V_1 = WINDOW_ LENGTH /2.

SHAPE_3	Size	WINDOW_POST_OUT_EDGE * (WINDOW_HEIGTH – 2 * WFOE)	mm
	Position	WINDOW_MEASURE_1 – (WINDOW_POST_OUT_EDGE/2) , WFOE	Lag 3
	Color	WINDOW_COULOR_OUT (RGB 217,150,144)	

Posten indsættes





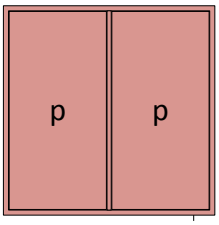
Som det fremgår af figuren til venstre har vinduerne en afstand fra den ydre ramme

= WINDOW_PANE_DISTANCE = WPD.

Hos Outline er den 4 mm. Vi laver derfor 2 afstandsbokse.

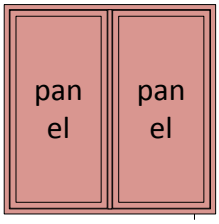
SHAPE_4	Size	(WINDOW_HEIGTH – 2 * (WFOE + WPD)) * (WINDOW_MEASURE_1 – WFOE - WPD - (WINDOW_POST_OUT_EDGE/2) - WPD)	mm
	Pos	WFOE + WPD, WFOE + WPD	L 5
SHAPE_5	Size	(WINDOW_HEIGTH – 2 * (WFOE + WPD)) * (WINDOW_HEIGTH – WINDOW_MEASURE_1 – WFOE - WPD - (WINDOW_POST_OUT_EDGE/2) - WPD)	mm
	Pos	WINDOW_MEASURE_1 + (WINDOW_POST_OUT_EDGE/2) + WPD, WFOE + WPD	L 5

Som placeres som i lag 5 på tegningen



Selve vinduespanelet består af en ydre del WINDOW_PANEL_OUT_EDGE (WPOE) og en inderste del WINDOW_PANEL_OUT_LIST (WPOL). Vi laver først bokse for den ydre del – vinduespanelet.

S_6	Size	(WINDOW_HEIGTH – 2 * (WFOE + WPD + WPOE)) *(WINDOW_MEASURE_1 – WFOE - WPD - WPOE - (WINDOW_POST_OUT_EDGE/2) – WPD - WPOE)	mm
	Pos	WFOE + WPD + WPOE, WFOE + WPD + WPOE	L 6
S_7	Size	(WINDOW_HEIGTH – 2 * (WFOE + WPD + WPOE)) *(WINDOW_HEIGTH – WINDOW_MEASURE_1 – WFOE - WPD - WPOE - (WINDOW_POST_OUT_EDGE/2) – WPD - WPOE)	mm
	Pos	WINDOW_MEASURE_1 + (WINDOW_POST_OUT_EDGE/2) + WPD + WPOE, WFOE + WPD + WPOE	L 6

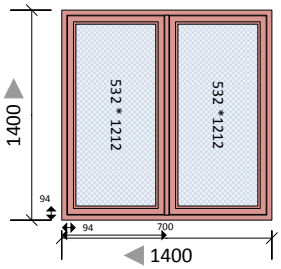


Slutteligt laver vi bokse som afgrænser den panelernes liste og som indeholder glasset.

S_7	Size	(WINDOW_HEIGTH – 2 * (WFOE + WPD + WPOE+WPOL)) *(WINDOW_MEASURE_1 – WFOE - WPD - WPOE – WPOL - (WINDOW_POST_OUT_EDGE/2) – WPD - WPOE– WPOL)	mm
	Pos	WFOE + WPD + WPOE + WPOL, WFOE + WPD + WPOE + WPOL	L 7
S_8	Size	(WINDOW_HEIGTH – 2 * (WFOE + WPD + WPOE+WPOL)) *(WINDOW_LENGTH – WINDOW_MEASURE_1 – WFOE - WPD - WPOE – WPOL - (WINDOW_POST_OUT_EDGE/2) – WPD - WPOE– WPOL)	mm

	Pos	WINDOW_MEASURE_1 + (WINDOW_POST_OUT_EDGE/2) + WPD + WPOE + WPOL, WFOE + WPD + WPOE + WPOL	L 7

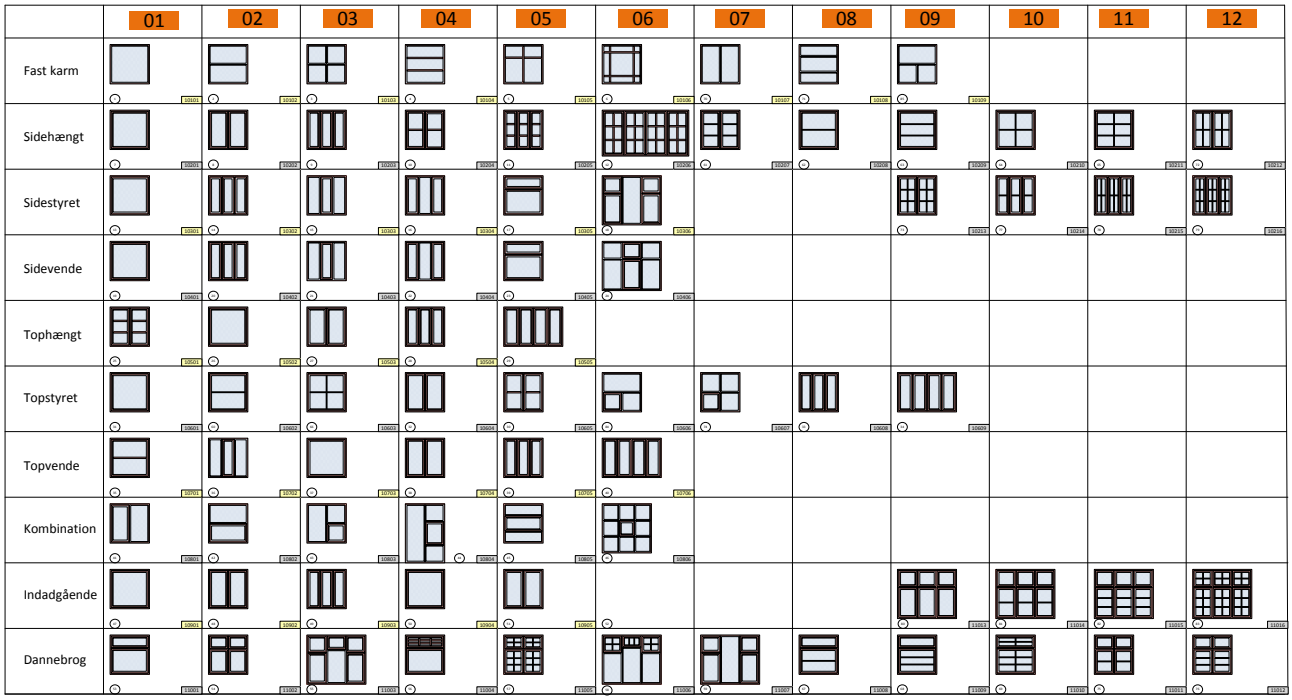
For vinduet kan de arealer som indgår i energiberegningen nu beregnes.



Total areal	=	1400 mm * 1400 mm
Glas areal	=	(532 mm * 1212 mm) * 2
Post areal		28 mm * 1332 mm
Karm areal	=	Total areal – glas areal – post areal
Omkreds af glasarealet	=	1212 mm * 4 + 532 mm *4
Glasandel	=	Glas areal/ Total areal

For fabrikanterne Outline, Rationel og STM vinduer laves en oversigt samtlige standard modeller og versioner som disse 3 fabrikanter producerer. Samtlige vinduer tegnes herefter som beskrevet i de ovenanførte eksempler. Det dokument jeg har udfærdiget er vedhæftet som bilag 6. På det tidspunkt arbejdede jeg med at inddele vinduesversionerne i 18 forskellige grupper (cases). I forbindelse med implementering er dette korrigeret til 12 cases og der er desuden korrigeret en del fejl/uhensigtsmæssigheder ligesom der er arbejdet mere med navngivningen af variable.

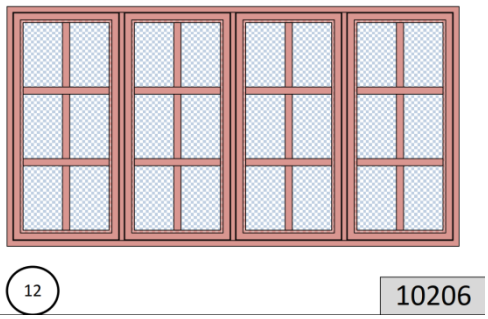
I Visio figuren herunder ses en tegning af samtlige modeller og versioner. For 2 modeller sidehængt og dannebrog er der 16 versioner. De sidste 4 er vist i modellen under og over.



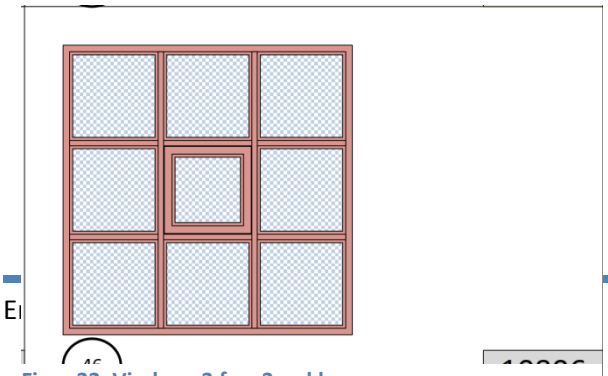
Figur 30. Vindue - versioner som kan vælges i webapplikationen

Ud fra dette besluttede jeg sammen med firmaet at energiberegningsprogrammet skulle kunne håndtere et vindue med maksimalt 4 fag (sidehæng 06)

og med maksimalt 3 rækker ruder (kombination 06)



Figur 31. Vindue - 4 fag, 1 række



Figur 32. Vindue - 3 fag, 3 rækker

Det vil sige et

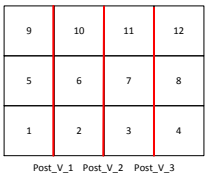
9	10	11	12
5	6	7	8
1	2	3	4

vindue med i alt 12 positioner, nummereret 1 - 12.

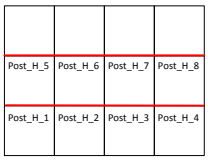
Figur 33. Vindue - 4 fag, 3 rækker. Position 1 - 12

Vinduet ville således kunne have maksimalt

- 12 ruder
- 3 vertikal poste
- 8 horisontale poste
- 9 horisontale sprosser
- 6 vertikale sprosser



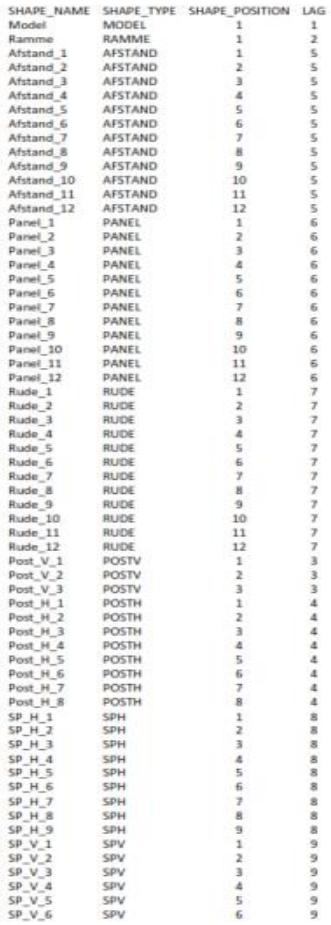
Figur 34. Vindue - maksimalt 3 vertikale poste



Figur 35. Vindue - maksimalt 8 horisontale poste

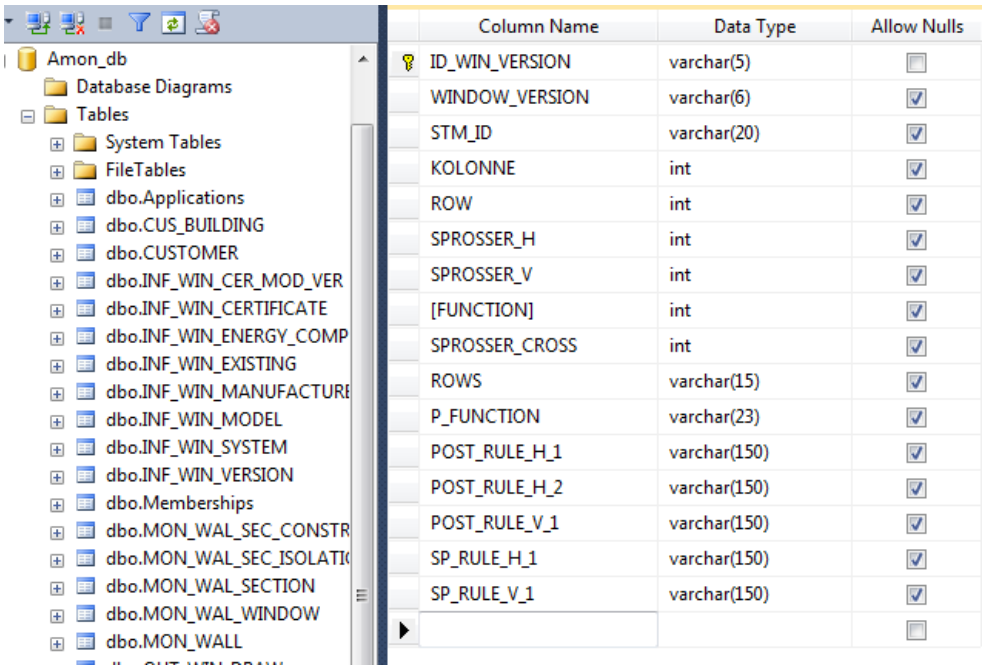
Ud fra dette lavede jeg et datasæt WINDOW_BASE_SHAPES som kan indeholde informationer for alle de shapes som er mulige for det mest komplicerede vindue (4x3).

Figur 36. WINDOW_BASE_SHAPES - mulige shapes - alle



SHAPE_NAME	SHAPE_TYPE	SHAPE_POSITION	LAG
Model	MODEL	1	1
Ramme	RAMME	1	2
Afstand_1	AFSTAND	1	5
Afstand_2	AFSTAND	2	5
Afstand_3	AFSTAND	3	5
Afstand_4	AFSTAND	4	5
Afstand_5	AFSTAND	5	5
Afstand_6	AFSTAND	6	5
Afstand_7	AFSTAND	7	5
Afstand_8	AFSTAND	8	5
Afstand_9	AFSTAND	9	5
Afstand_10	AFSTAND	10	5
Afstand_11	AFSTAND	11	5
Afstand_12	AFSTAND	12	5
Panel_1	PANEL	1	6
Panel_2	PANEL	2	6
Panel_3	PANEL	3	6
Panel_4	PANEL	4	6
Panel_5	PANEL	5	6
Panel_6	PANEL	6	6
Panel_7	PANEL	7	6
Panel_8	PANEL	8	6
Panel_9	PANEL	9	6
Panel_10	PANEL	10	6
Panel_11	PANEL	11	6
Panel_12	PANEL	12	6
Rude_1	RUDE	1	7
Rude_2	RUDE	2	7
Rude_3	RUDE	3	7
Rude_4	RUDE	4	7
Rude_5	RUDE	5	7
Rude_6	RUDE	6	7
Rude_7	RUDE	7	7
Rude_8	RUDE	8	7
Rude_9	RUDE	9	7
Rude_10	RUDE	10	7
Rude_11	RUDE	11	7
Rude_12	RUDE	12	7
Post_V_1	POSTV	1	3
Post_V_2	POSTV	2	3
Post_V_3	POSTV	3	3
Post_H_1	POSTH	1	4
Post_H_2	POSTH	2	4
Post_H_3	POSTH	3	4
Post_H_4	POSTH	4	4
Post_H_5	POSTH	5	4
Post_H_6	POSTH	6	4
Post_H_7	POSTH	7	4
Post_H_8	POSTH	8	4
SP_H_1	SPH	1	8
SP_H_2	SPH	2	8
SP_H_3	SPH	3	8
SP_H_4	SPH	4	8
SP_H_5	SPH	5	8
SP_H_6	SPH	6	8
SP_H_7	SPH	7	8
SP_H_8	SPH	8	8
SP_H_9	SPH	9	8
SP_V_1	SPV	1	9
SP_V_2	SPV	2	9
SP_V_3	SPV	3	9
SP_V_4	SPV	4	9
SP_V_5	SPV	5	9
SP_V_6	SPV	6	9

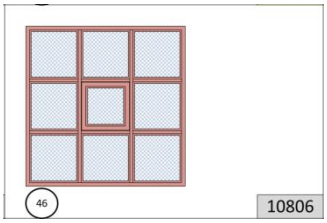
I datasættet som indeholder informationer om de forskellige versioner af vinduerne – i alt de 84 forskellige som er vist i figur 30 herover, skal der tilføjes informationer om hvilke af de forskellige shapes som skal anvendes til at tegne og beregne vinduet. Disse informationer registreres i datasættet INF_WIN_VERSION.



Figur 37. INF_WIN_VERSION - variable

Navn	Anvendelse	Værdier
KOLONNE	**Kolonner**	**1 – 4**
ROW	**Rækker**	**1 – 3**
SPROSSER_H	**Sprosser horisontalt**	**1 – 9**
SPROSSER_V	**Sprosser vertikalt**	**1 – 6**
FUNCTION	**Funktion**	**1 fast, 2 åbne, 3 kombination**
SPROSSER_CROSS	**Sprosser som krydser**	**1 – 9**
ROWS	**Placering - horisontale poster**	**0/1 * 8**
P_FUNCTION	**Placering og funktion - ruder**	**0/1/2 * 12**
Regler for poste og sprosser		
POST_RULE_H_1	**Regel - Post - horisontal**	
POST_RULE_H_2	**Regel 2 - Post - horisontal**	
POST_RULE_V_1	**Regel - Post - vertikal**	
SP_RULE_H_1	**Regel - Sprosse - horisontal**	
SP_RULE_V_1	**Regel - Sprosse - vertikal**	

Eksempel på anvendelse af informationer i INF_WIN_VERSION. Vindue i 3 fag, 3 rækker, 2 vertikale og 6 horisontale poste. 9 ruder, heraf 8 faste og en kan åbnes (position 6).



ID_WINDOW_VERSION	10806	
KOLONNE	3	Vinduet har 3 fag
ROW	3	Vinduet har 3 rækker
ROWS	1.1.1.0.1.1.1.0	Post_H_1 = 1 = ja der er en post Post_H_2 = 1 Post_H_3 = 1 Post_H_4 = 0 = nej der er ingen post Post_H_5 = 1 Post_H_6 = 1 Post_H_7 = 1 Post_H_8 = 0
P_FUNCTION	1.1.1.0.1.2.1.0.1.1.1.0 For de forskellige positioner 1 – 12 angives om der sidder et vindue og om det er fast eller kan åbnes. 0 = nej 1 = fast karm 2 = åbnes Hvis vinduet kan åbnes skal der være en shape for afstand, panel og rude. Hvis vinduet ikke kan åbnes skal der kun være en shape for ruden.	Rude_1 RUDE = ja fast Rude_2 RUDE= ja fast Rude_3 RUDE= ja fast Rude_4 RUDE= nej Rude_5 RUDE= ja fast Rude_6 RUDE = ja åbne Rude_7 RUDE= ja fast Rude_8 RUDE = nej Rude_9 RUDE= ja fast Rude_10 RUDE= ja fast Rude_11 RUDE= ja fast Rude_12 RUDE = nej
FUNCTION	3	Kombinationsvindue. Både vinduer som kan åbnes og vinduer som ikke kan åbnes
SPROSSER_H	0	Ingen
SPROSSER_V	0	Ingen
SPROSSER_CROSS	0	Ingen
POST_RULE_H_1	R + (WINDOW_HEIGTH - RR - POE*2)/3 + PO	
POST_RULE_H_2	WINDOW_HEIGTH * 0.64	
POST_RULE_V_1	R + (WINDOW_WIDTH - RR - POE*2)/3 + PO	
POST_RULE_V_2		
SP_RULE_H_1		
SP_RULE_H_2		
SP_RULE_V_1		
SP_RULE_V_2		

Før jeg kombinerer datasættet med datasættet med shapes organiseres versionerne i 12 cases 1 – 12 hvor nummeret svarer til positionen på den sidste rude i vinduet og antallet af ruder, vertikale og horisontale poste placeres i RUDER, POST_V og POST_H.

KOLONNE	ROW	CASE
1	1	1
1	2	2
1	3	3
2	1	4
2	2	5
2	3	6
3	1	7
3	2	8
3	3	9
4	1	10
4	2	11
4	3	12

Tabel 6. Vindue - cases

```
SELECT
  INF_WIN_VERSION.ID_WIN_VERSION,  INF_WIN_VERSION.WINDOW_VERSION,  INF_WIN_VERSION.STM_ID,
  INF_WIN_VERSION.KOLONNE,  INF_WIN_VERSION.ROW,  INF_WIN_VERSION.SPROSSER_H,
  INF_WIN_VERSION.SPROSSER_V,  INF_WIN_VERSION.[FUNCTION],  INF_WIN_VERSION.SPROSSER_CROSS,
  INF_WIN_VERSION.ROWS,  INF_WIN_VERSION.P_FUNCTION,  INF_WIN_VERSION.POST_RULE_H_1,
  INF_WIN_VERSION.POST_RULE_H_2,  INF_WIN_VERSION.POST_RULE_V_1,  INF_WIN_VERSION.SP_RULE_H_1,
  INF_WIN_VERSION.SP_RULE_V_1,  INF_WIN_VERSION.CASE,  INF_WIN_VERSION.RUDER,
  INF_WIN_VERSION.POST_V,  INF_WIN_VERSION.POST_H,

  WINDOW_BASE_SHAPES.SHAPE_NAME,
  WINDOW_BASE_SHAPES.SHAPE_TYPE,
  WINDOW_BASE_SHAPES.SHAPE_POSITION,
  WINDOW_BASE_SHAPES.LAG
FROM
  INF_WIN_VERSION CROSS JOIN WINDOW_BASE_SHAPES
```

Jeg henter herefter ID_MANUFACTURER for hver vinduesversion og tilføjer informationer fra tbl MANUFACTURER. Nedenfor er angivet de værdier som altid skal hentes fra producenten.

Variabel	Kort	Indhold	Anvendes	Kilde	Størrelse
WINDOW_HEIGTH	W_H	Vinduets højde	Altid	Indtastes - altid	
WINDOW_LENGTH	W_L	Vinduets bredde	Altid	Indtastes - altid	
WINDOW_FRAME_HEIGTH	WF	Vinduets ramme	Altid	Fast	50
WINDOW_FRAME_OUT_EDGE	WFOE	Vinduets ydre ramme	Altid	Fast	34
WINDOW_FRAME_OUT_LIST	WFOL	Vinduets ydre liste	Kun fast ramme	Fast	16
WINDOW_POST	WP	Opdeling af vinduet (ruder)	Poster	Fast	(60)
WINDOW_POST_OUT_EDGE	WPOSTOE	Post – udvendig højde/bredde	Poster	Fast	28
WINDOW_PANE	WP	Vinduet panel	Kun åbne	Fast	56
WINDOW_PANE_DISTANCE	WPD	Afstand mellem ydre ramme og panel	Kun åbne	Fast	4
WINDOW_PANE_OUT_EDGE	WPOE	Panelets ydre ramme	Kun åbne	Fast	40
WINDOW_PANE_OUT_LIST	WPOL	Panelets ydre liste	Kun åbne	Fast	16
WINDOW_SPROSSE	WS	Sprosser er en del af ruden	Sprosser	(Fast)	25/42/60
WINDOW_MEASURE_1	M1	Mål som tilpasser	Poster eller sprosser	Valgfri	
WINDOW_MEASURE_2	M2	Mål som tilpasser	Poster eller sprosser	Valgfri	
WINDOW_MEASURE_3	M3	Mål som tilpasser	Poster eller sprosser	Valgfri	
WINDOW_MEASURE_4	M4	Mål som tilpasser	Poster eller sprosser	Valgfri	

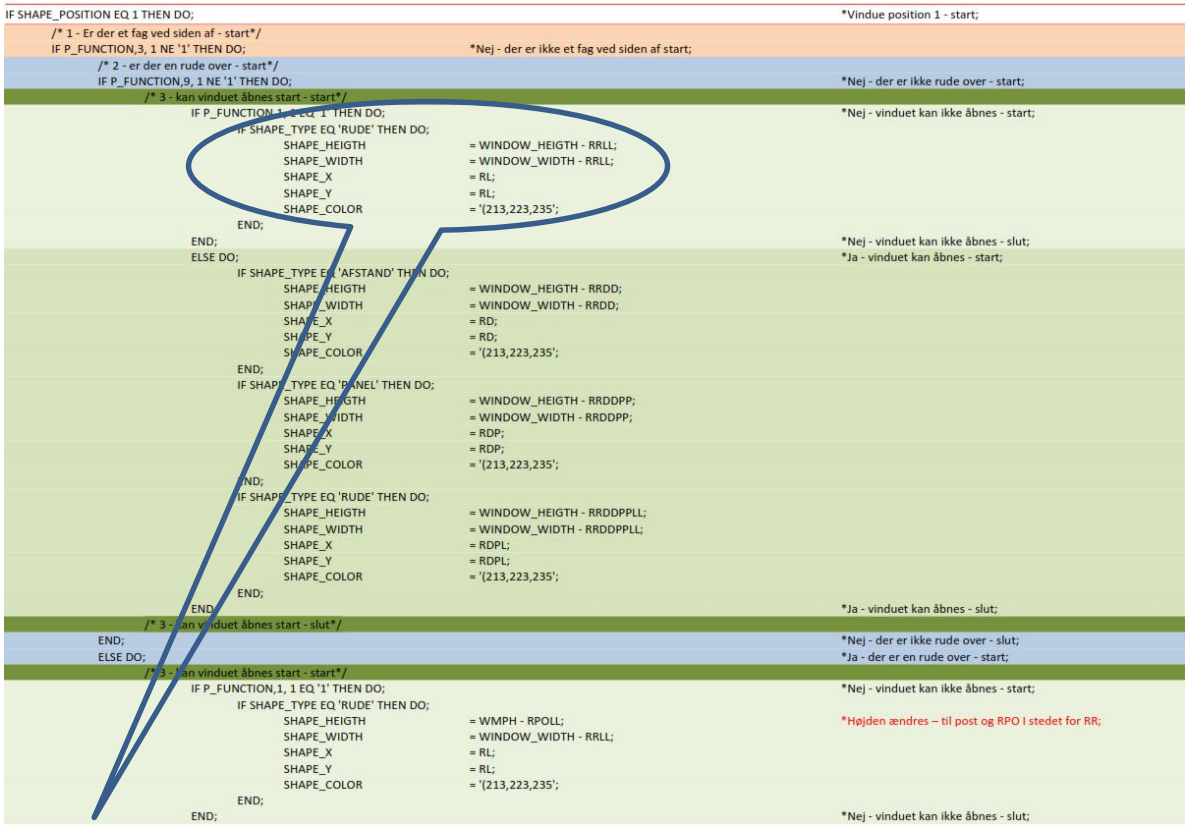
Tabel 7. Værdier som anvendes til at tegne og beregne vinduerne

Det resulterende datasæt VINDUER_SHAPES opdateres kun når der tilføjes nye producenter, vinduessystemer eller vinduesversioner eller hvis der ændres i de eksisterende værdier.

Når montøren har foretaget en registrering af nye vinduer kombineres informationerne med VINDUER_SHAPES og alle shapes beregnes (størrelse og placering). I projektet anvendes alene størrelsen.

Beregningen af shapes baseres på deres position (SHAPE_POSITION) og for vinduernes vedkommende på

indholdet i P_FUNCTION og SHAPE_TYPE. Figur 38 viser et lille udsnit af hvordan det fungerer.



Figur 38. Position 1. Anvendelse af SHAPE_POSITION, P_FUNCTION og SHAPE_TYPE til at beregne vinduets mål

De værdier som anvendes i beregningen af er fundet i forbindelse med tegning af vinduet. Nedenfor er angivet Position 1 værdier, med gult er lilla er markeret værdierne hvis der hverken er ruder over eller ved siden af og hvis vinduet ikke kan åbnes. Alle værdier for alle positioner er vedhæftet i bilag 7. De anvendte forkortelser som alle er baseret på højde, længde af vinduet og producentens standard mål er defineret i forbindelse med koden under implementering.

Position 1.1 Ingen ved siden af, ingen over

POST	0	Bredde	X	Højde	Y
ÅBNE					
	DISTANCE	(BREDDE) – RRDD	RD	HØJDE – RRDD	RD
	PANEL	(BREDDE) – RRDDPP	RDP	HØJDE – RRDDPP	RDP
	RUDE	(BREDDE) - RRDDPPLL	RDPL	HØJDE - RRDDPPLL	RDPL
FAST					
	RUDE	(BREDDE) - RRLL	RL	HØJDE - RRLL	RL

Position 1.2 Ingen ved siden af, rude over

POST	0	Bredde	X	Højde	Y
ÅBNE					
	DISTANCE	(BREDDE) – RRDD	RD	WMPH1 - RPODD	RD
	PANEL	(BREDDE) – RRDDPP	RDP	WMPH1 - RPODDPP	RDP
	RUDE	(BREDDE) - RRDDPPLL	RDPL	WMPH1 - RPODDPPLL	RDPL
FAST					
	RUDE	(BREDDE) - RRLL	RL	WMPH1 - RPOLL	RL

Position 1.3 Rude ved siden af, ingen over

POST	0	Bredde	X	Højde	Y
ÅBNE					
	DISTANCE	(WMPV1) - RPODD	RD	HØJDE – RRDD	RD
	PANEL	(WMPV1) - RPODDPP	RDP	HØJDE – RRDDPP	RDP
	RUDE	(WMPV1) - RPODDPPLL	RDPL	HØJDE - RRDDPPLL	RDPL
FAST					
	RUDE	(WMPV1) - RPOLL	RL	HØJDE - RRLL	RL

Position 1.4 Rude ved siden af, rude over

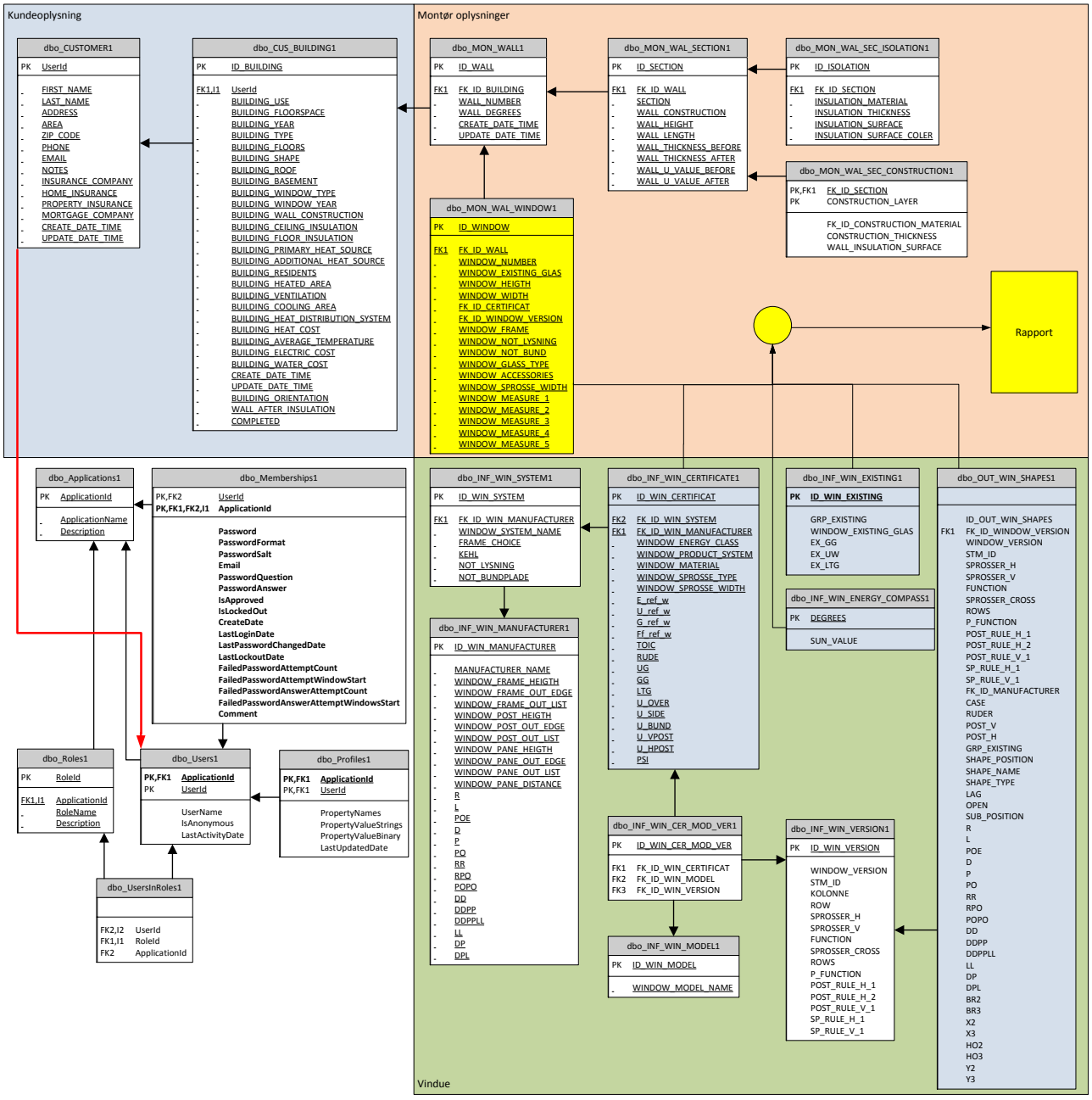
POST	0	Bredde	X	Højde	Y
ÅBNE					
	DISTANCE	(WMPV1) - RPODD	RD	WMPH1 - RPODD	RD
	PANEL	(WMPV1) - RPODDPP	RDP	WMPH1 - RPODDPP	RDP
	RUDE	(WMPV1) - RPODDPPLL	RDPL	WMPH1 - RPODDPPLL	RDPL
FAST					
	RUDE	(WMPV1) - RPOLL	RL	WMPH1 - RPOLL	RL

Tabel 8. Vindue - beregning af størrelser

Datamodel

Databasen overholder første normalform (1NF) – der er ikke data der går igen, der er ingen redundans i tabellerne og alle tuplerne er atomare. Databasen overholder anden normalform (2NF) i det der er en primær nøgle per tabel. For at databasen skal være på tredje normalform skal alle variable i tabellerne være knyttet til den primære nøgle og ingen variable må være tomme. Datasæt fra brugeren og montøren er næsten på 3NF, men enkelte variable f.ex. WALL_INSULATION_SURFACE i MON_WAL_SEC_CONSTRUCTION kan være tomme. I flere vinduestabeller er ID_MANUFACTURER medtaget hvor dette ikke er nødvendigt.

Ved beregning af energibesparelsen indgår data fra montøren dbo_MON_WALL_WINDOW1 og de 4 stationære tabeller dbo_INF_WIN_CERTIFICATE1, dbo_INF_WIN_EXISTING1, dbo_INF_WIN_ENERGY_COMPASS1 og dbo_OUT_WIN_SHAPES1.



Tabel 9. ER diagram over databasetabeller

6. IMPLEMENTERING

Under implementering beskrives først opsætningen af projektet og den overordnede struktur af webapplikationen. Derefter beskrives kommunikationen med databasen. Beskrivelsen af applikationens formularer er opdelt i indtastningsformularerne for kunde og montør og lister med bygninger og energirapporter og energirapporten.

Opsætning

Server, hjemmeside og database

Jeg installerede Microsoft Windows Server 2012 R2, hvor jeg brugte jeg det tilhørende program Server Manager til at installere Web Server(IIS), som hjemmesiden kører på, og anden nødvendig software som ASP.NET 4.5.

Projektet blev lagt i mappen C:\inetpub\wwwroot. Det er den mappe som default bruges til port 80 (Internet porten). Jeg brugte (IIS)manageren til at tilføje siden og give den en Url.

Jeg anvendte SQL Server 2014 Management Studio til opsætningen af databasen og jeg brugte Windows Server's indbyggede (localdb)\v11.0 som SQL server.

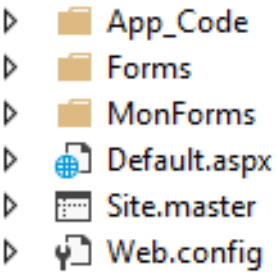
Herefter åbnede jeg op i firewallen. Domænet blev slutteligt registreret hos en domæneudbyderen (ONE.com).

Webapplikationen - struktur

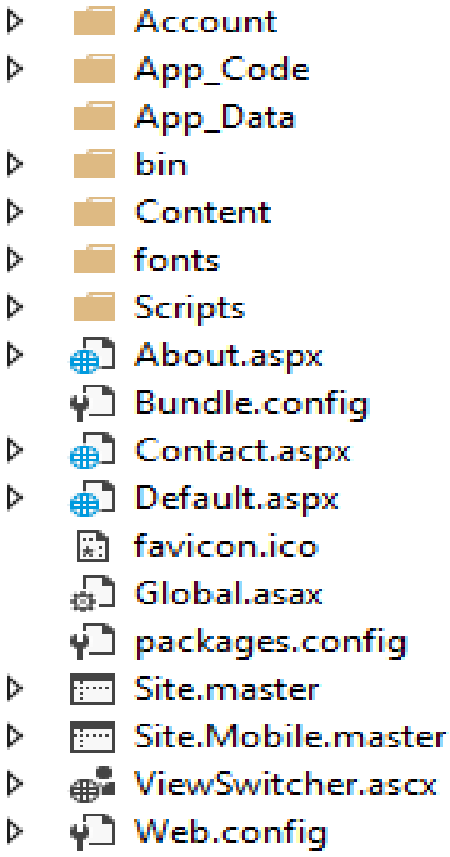
Webapplikationen er en ASP.NET Web Site, oprettet i Visual Studio.

I Visual Studio har jeg taget udgangspunkt i den autogenerede struktur som Visual Studio danner når man opretter en ASP.NET Web Form Site. Strukturen er vist i figur 39.

I forbindelse med projektet har jeg næsten udelukkende ændret i mappen App_Code og filerne Default.aspx, Site.master og Web.config. Derudover har jeg tilføjet mappen Forms som indeholder den webform kunden anvender og MonForms, som indeholder montørens indtastningsformular, energirapport, bruger og rapport lister. Figur 40 viser de mapper og rodfiler hvor jeg har foretaget ændringer af betydning for projektet.



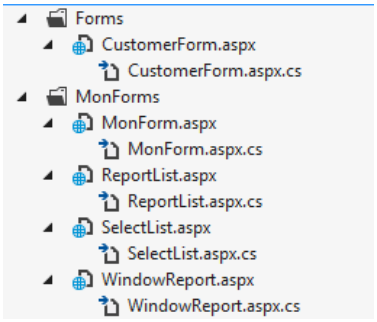
Figur 39. Webapp - tilføjelser og ændringer



Figur 40. Web Form Site - autogenereret struktur

WEBFORMULARER

Jeg har delt websiderne på 2 mapper på grund af brugerhåndteringen. Jeg har derved mulighed for at afgrænse det kunden og det montøren kan se.



Figur 41. Webapp - struktur - sider

Ud over placeringen af filerne er strukturen opbygget svarende til de sider man kan tilgå på hjemmesiden. For hver side er der oprettet en aspx-fil. En aspx-fil er den fil som indeholder sidens struktur, Html og lignende. For hver aspx-fil er der en aspx.cs-fil som indeholder sidens code behind, i dette tilfælde C# koden. Aspx.cs-filen fungerer som en slags kontroller som styrer hvad der sker med aspx-filen, hvilke felter som vises og hvilke data der hentes, gemmes og lignende.

For at forstå hvordan siderne virker, bliver man nødt til have kendskab til ASP Web forms Life-cycle og hvilke Life-cycle Events der bliver kørt og hvornår. Hovedparten af den C# kode jeg har skrevet bliver kørt i Page initialization og Load, hvor jeg har anvendt metoderne `Page_Init`, `Page_Load` og "`Button_Click`". `Page_Init` køres som det første i page initialization stadiet. Det er her data hentes og det her elementer, som senere kan indeholde data, tilføjes. I Page_Load tilføjes elementer som ikke skal modtage data og her sættes værdierne på de elementer som er i .aspx filen. `Button`_Click som også forgår i Load fasen er der hvor jeg gemmer data. I den kode jeg har skrevet har jeg forsøgt at udføre så meget som muligt tidligt i sidens livscyklus.

Stage – Life-cycle
Page request
Start
Page initialization
Load
Validation
Postback event handling
Rendering
Unload

Figur 422. Webside - livscyklus

APP_CODE

De filer som ikke er aspx eller aspx.cs-filer, som jeg har arbejdet med i projektet ligger hovedsageligt i App_Code. App_Code indeholder .cs filer (c#) og de filer som via config filen kommuniker med databasen. Ændringerne beskrives under database og rapport.

SITEMASTER

Ændringerne i Site.master beskrives under indtastningsformularerne.

WEBCONFIG

En .config er ASP.NET konfigurationsfiler, som er skrevet i XML formatet. Web.config er hovedkonfigurationsfilen i projektet. Filen indeholder bland andet en beskrivelse af det framework og de teknologier der anvendes i projektet. Filen indeholder også den information der nødvendige for at kunne kommunikere med databasen. Der er 2 connectionstrings til databasen, en `connectionStrings` som jeg har skrevet og en der er blevet generet i forbindelse med at jeg har produceret .emdx filen, hvilket beskrives nærmere under beskrivelsen af database implementeringen.

Den jeg har skrevet er vist herunder.

```
    <add name="AmondbConnectionString" connectionString="Data Source=(localdb)\v11.0;Initial
Catalog=Amondb;Integrated Security=True" providerName="System.Data.SqlClient" />
```

Strengen består af 3 dele. I den første del (`name`) navngives strengen, således at der kan refereres til den fra andre filer. Den midterste del (`connectionString`) er selve connectionstrengen. Første del af denne angiver hvor datakilde der skal forbindes til er placeret, i dette tilfælde `Data Source=(localdb)\v11.0`. Dernæst angives det hvilken datakilde det er `Initial Catalog=Amondb`. I den sidste del af connectionstrengen er

`Integrated Security=True` hvilket betyder at der anvendes Windows authentication som er sikkerhed for at brugeren må tilgå databasen. Strengens tredje del `providerName="System.Data.SqlClient` sætter providerName til System.Data.SqlClient er en række klasser der bruges til at kommunikere med en SQLdatabase.

NAVNGIVNING

Retningslinjer for navngivningen i projektet.

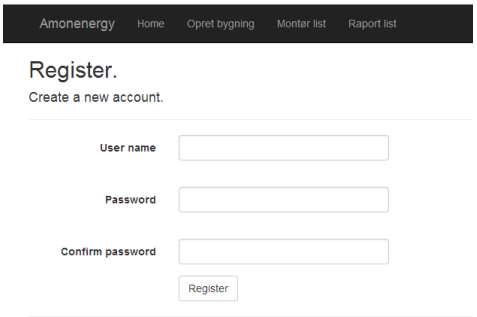
Databasen. Tabeller og variable skrives med store bogstaver med bindestreg mellem ordene. Tabellerne er navngivet så man kan se hvor data kommer fra med en forkortelse på 3 bogstaver. F.eks MON_WAL_SECTION: MON_ fordi det er montøren der indtaster og WAL_ fordi section tilhører en væg. Alle primary keyes starter med ID_ og foreign keys starter med FK_ID_

Applikationen. Pascalcase er anvendt for filnavne. Camelcase er anvendt for variabelnavne og metoder. Pascalcase og camelcase er måder at sammensætte ord på uden bruge bindestreg eller mellemrum. I pascalcase starter navnet med stort forbogstav, mens camelcase for det første ord anvender et lille forbogstav.

Variable som kommer fra databasen eller som skal over i databasen bliver navngiver med stort.

ACCOUNT

Under mappen Account køres filen Register.aspx. Man får herefter mulighed for at oprette en bruger. Herefter opretter Visual Studio automatisk alle user tabeller som vist i ER-diagrammet (tabel 9).



Figur 43. Oprettelse af brugertabeller. Registrering

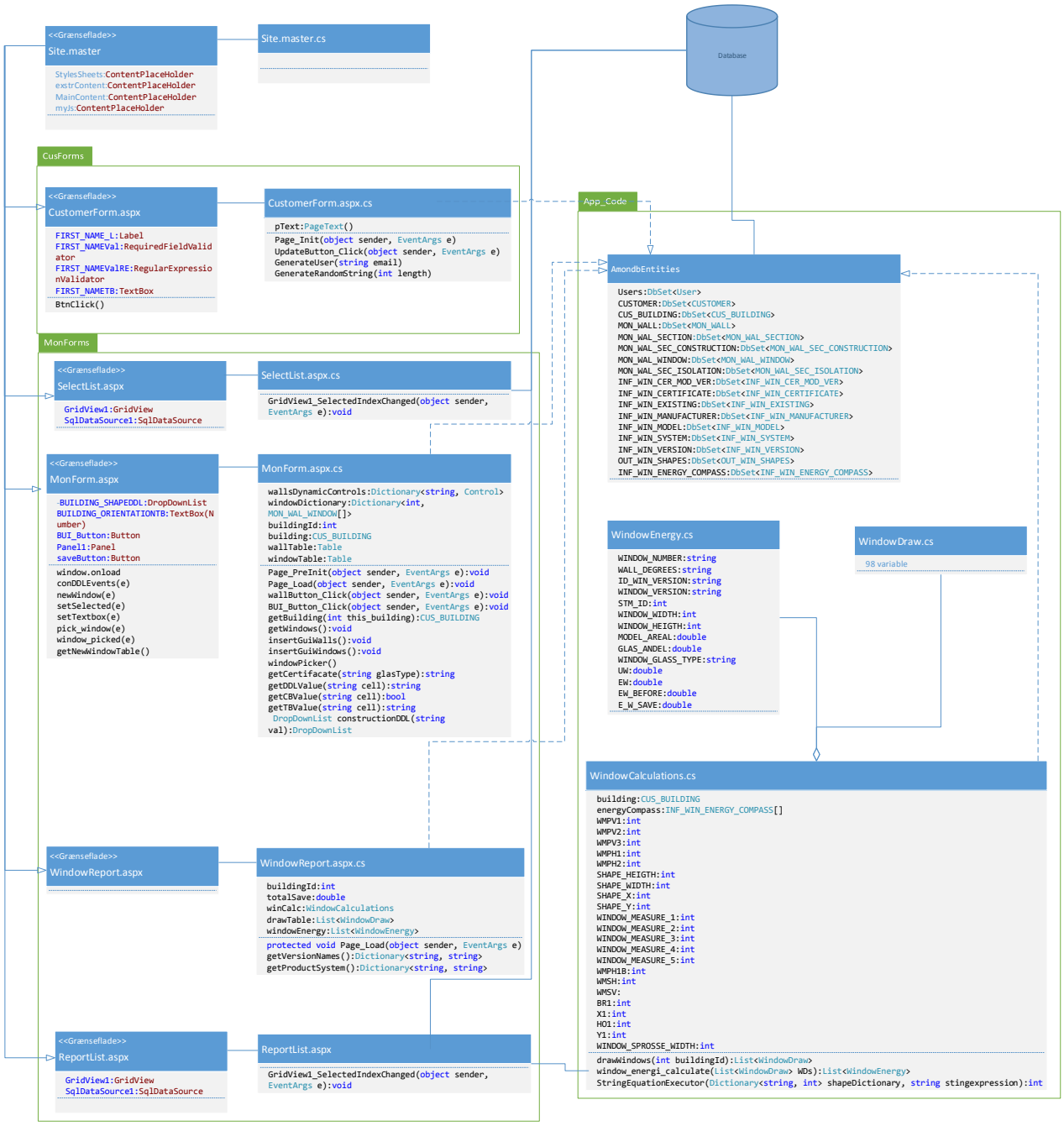
ANDRE ÆNDRINGER

Udover det beskrevne har jeg foretaget mindre ændringer i

- Scripts. Her har jeg tilføjet
  - jquery-1.10.2.js
  - `ui/1.10.4/jquery-ui.js`
- Bin. Her har jeg tilføjet
  - NCalc.dll (som beskrives under energirapporten)
- Content. Her har jeg tilføjet
  - /ui/1.10.4/themes/smoothness/jquery-ui.css
  - Billeder af vinduerne

Klassediagram

Figur xx viser et klassediagram (Overblik) som illustrerer de vigtigste filer i projektet og deres relationer. Jeg har valgt kun at medtage de vigtigste variable og metoder. Alle grænsefladerne nedarver fra Site.master og næsten alle deres code behind filer bruger AmonEntities interfacet til at kommunikere med databasen. WindowDraw bliver brugt som et objekt til at holde 98 variable derfor har jeg skrevet 98 variable.



Figur 44. Klassediagram

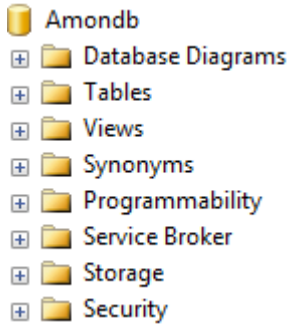
Database

Oprettelse af database og tabeller.

DATABASE

Databasen er implementeret i SQL Server 2014 Management Studio. Databasen oprettes ved at højre klikke på database og derefter klikke på new database.

Den nye database er vist i figur 45. Efter at databasen er oprettet kan database-tabellerne tilføjes.



Figur 45. Database - ny database

DATABASETABELLER

Tabellerne er lavet med SQL queries og føjet til mappen Tables (figur 45). Et eksempel på en query som laver en tabel ses i figur 46. Tabellen der bliver lavet er INF_WIN_ENERGY_COMPASS og den har primary key DEGREES.

```
CREATE TABLE [dbo].[INF_WIN_ENERGY_COMPASS](
    [DEGREES] [int] NOT NULL,
    [SUN_VALUE] [int] NOT NULL,
PRIMARY KEY CLUSTERED
(
    [DEGREES] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

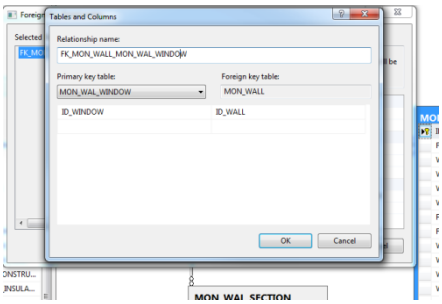
Figur 46. Oprettelse af databasetabel - SQL query

RELATIONER

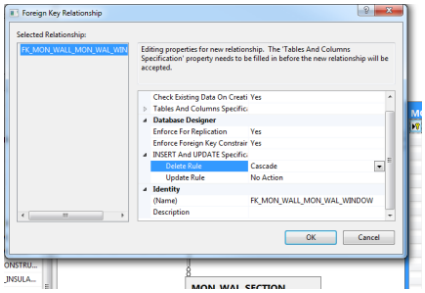
Alle tabellerne er initialt dannet uden relationer.

Relationer blev til føjet gennem et diagram som findes ved at højre klikke på mappen Database Diagrams -> New Database Diagram.

Når man har valgt New Database Diagram kommer der et vindue frem Add Table hvor man kan tilføje de tabeller man skal bruge. Relationerne mellem tabellerne laves ved at man trækker en streg fra en tabel til en anden. Derved kommer der et vindue frem, Tables and Colums, hvor man kan skrive hvad relationen skal hedde, hvilken tabel der er primary key table og hvilken variable der skal relateres. Når Tables and Colums er udfyldt kommer ses vinduet, Foreign Key Relationship, hvor man kan specificere relationen. I Foreign Key Relationship har jeg brugt default værdier da der ikke er nogle specielle relationer. Det eneste jeg har ændret er Delete Rule, som jeg har sat til Cascade på alle de tabeller der relaterer til en bygning både direkte og indirekte. Cascade som delete rule betyder, at hvis en bygning slettes, slettes alle de vægge som tilhører bygningen, hvorefter alle væg sektionerne slettes og så fremdeles.



Figur 47. Navngivning af relation



Figur 48. Anvendelse af delete rule

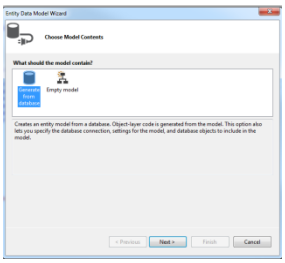
Database kommunikation

Kommunikationen med databasen kan implementeres på flere måder. Jeg har valg at anvende ADO.NET Entity Data Model.

ADO.NET Entity model er implementeret gennem Entity Data Model Wizard i Visual Studio. Jeg tilføjede en ADO.NET Entity Data Model til mappen App_Code, hvilket betød Entity Data Model Wizarden blev aktiveret.

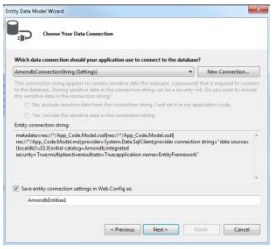
Entity Data Model Wizard

I wizarden vælges først hvad modellen skal indeholde. Her er databasen valgt. Figur 49.



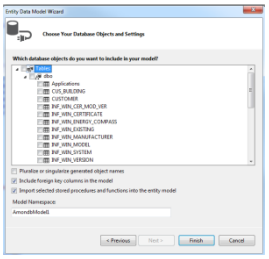
Figur 49. EDMWizard - valg af database

Det næste der et vælges er connetion som skal anvendes for at kommunikere med databasen. Her valgte jeg den connectionString jeg har lavet i min web config. Figur 50.



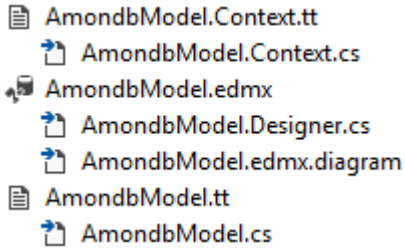
Figur 50. EDMWizard - valg af connection

Herefter vælges de tabeller som skal bruges og at foreign key columns skal inkluderes. Figur 51.



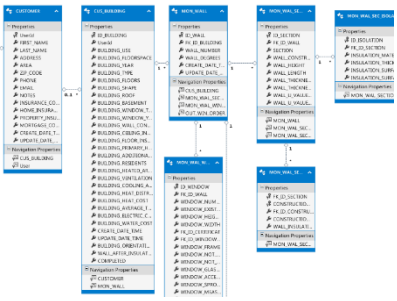
Figur 51. EDMWizard - valg af tabeller

Når wizarden kører bliver der generet 7 filer. Figur 52.



Figur 52. EDMWizard - resultat

AmondbModel.edmx et diagram hvor man kan se tabellerne og deres relationer. Figur 53. Det er her man skal ændre hvis der er lavet ændringer i databasen. Man højre klikker og vælger Update Model From database. Når AmondbModel.edmx opdateres bliver de andre filer også opdateret.



Figur 53. AmondbModel.edmx

Filen AmondbModel.cs indeholder en klasse for hver tabel med variable og relationer. Relationerne mellem tabellerne er implementeret ved brug af ICollection, som er et interface. ICollection for så den klasse som

tabellen relatere til. Et eksempel på en autogenereret klasse vises i figur 54, klassen er den til MON_WALL.

AmondbModel.Context.cs er det eneste sted jeg selv har tilføjet kode. Koden der er tilføjet ligger under OnModelCreating og sikrer at mine delete regler fra databasen også bliver tilføjet til klasserne så der ikke opstår fejl.

Når programmet kommunikerer med databasen anvendes linq queries og at data placeres i objekter af de klasser der er genereret i AmondbModel.cs.

De 3 vigtige filer er AmondbModel.edmx, AmondbModel.Context.cs og AmondbModel.

```
public partial class MON_WALL
{
    4 references
    public MON_WALL()
    {
        this.MON_WAL_SECTION = new HashSet<MON_WAL_SECTION>();
        this.MON_WAL_WINDOW = new HashSet<MON_WAL_WINDOW>();
        this.OUT_WIN_ORDER = new HashSet<OUT_WIN_ORDER>();
    }
    4 references
    public int ID_WALL { get; set; }
    1 reference
    public int FK_ID_BUILDING { get; set; }
    32 references
    public Nullable<int> WALL_NUMBER { get; set; }
    5 references
    public Nullable<int> WALL_DEGREES { get; set; }
    1 reference
    public System.DateTime CREATE_DATE_TIME { get; set; }
    1 reference
    public System.DateTime UPDATE_DATE_TIME { get; set; }
    0 references
    public virtual CUS_BUILDING CUS_BUILDING { get; set; }
    9 references
    public virtual ICollection<MON_WAL_SECTION> MON_WAL_SECTION { get; set; }
    5 references
    public virtual ICollection<MON_WAL_WINDOW> MON_WAL_WINDOW { get; set; }
    1 reference
    public virtual ICollection<OUT_WIN_ORDER> OUT_WIN_ORDER { get; set; }
}
```

Figur 54. AmondbModel.cs – MON_WALL klasse

Indtastningsformularer

GENERELT DESIGN – SITE.MASTER

Jeg har valgt at bruge en masterpage – Site.Master (SM) til alle siderne. Det betyder at alle sidernes stuktur er ens. Alle sider nedarver fra SM. SM indeholder en række vigtige elementer.

Menuen ligger i SM. Det er samme menu for alle sider, dog afhængig af om man er logget ind (Jeg har valgt ikke at medtage loginfunktionen i projektet). Det er også i SM at jeg henter de Javascript biblioteker og css filer jeg skal bruger på alle siderne.

SM indeholder desuden 4 placeholders, som er der hvor koden der er speciel for den enkelte side placeres. De 4 placeholders er

- Stylesheets
- exstraContent
- MainContent
- myJs.

Hver placeholder har sin egen funktion. I Stylesheets placeres css, exstraContent og MainContent indeholder det som skal vises på siden - html og ASP.controls. MainContent er placeret midt på siden og indeholde langt det meste, mens exstraContent er medtaget for at kunne tilføje elementer som ikke skal placeres midt på siden. myJs indeholder Javascript som er specielt for siden.

Når man opretter en ny side (aspx fil) med MasterPage i dette tilfælde Site.Master tilføjes koden vist i figur 55, hvor de 4 contentplaceholdes fremgår.

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.master" CodeFile="xxx.aspx.cs" Inherits="xxx" %>
<asp:Content ID="Content1" ContentPlaceHolderID="Stylesheets" Runat="Server"></asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="exstraContent" Runat="Server"></asp:Content>
<asp:Content ID="Content3" ContentPlaceHolderID="MainContent" Runat="Server"></asp:Content>
<asp:Content ID="Content4" ContentPlaceHolderID="myJs" Runat="Server"></asp:Content>
```

Figur 55. Kode fra Site.Master

BOOTSTRAP

Bootstrap er anvendt i forbindelse med sidens layout. Bootstrap er et front-end framework som gør det nemt at få siderne til at se pæne ud, på alle skærmstørrelser. Bootstrap er en række JS og CSS filer som man linker til, hvilket sker i SM. Når man designer siden, bruger man css klasser til at implementere Bootstrap funktionaliteten.

Kunde form

Kundeformularen dannes af CustomerForm.aspx hvor indtastningsfelterne er organiseret i 3 grupper svarende til de 3 horisontale faneblade beskrevet under design Bruger, Bygning og Forbrug. Koden ligger i placeholderen MainContent.

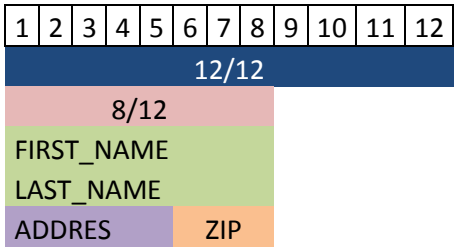
INDTASTNINGSFELTERNE

Indtastningsfelterne er i formularen placeret i kolonner. Opdelingen i kolonner foretages ved brug af html div-elementer, som for en CSS Bootstrap klasse. Figur 56. viser de første 4 indtastningsfelter i gruppen af bruger informationer.

```
1<h3 class="">Bruger <span class="glyphicon glyphicon-user"></span></h3>
2   <div class="row">
3       <div class="col-lg-8 col-md-6">
4           <div class="row">
5               <div class="col-lg-6 col-md-12">
6                   <div class="form-group">
7                       <asp:Label ID="FIRST_NAME_L" CssClass="control-label" runat="server" />
8                       <asp:RequiredFieldValidator ID="FIRST_NAMEVal" ControlToValidate="FIRST_NAMETB" CssCla
9                       <asp:RegularExpressionValidator ID="FIRST_NAMEValRE" ControlToValidate="FIRST_NAMETB"
10                      <asp:TextBox CssClass="form-control" ID="FIRST_NAMETB" runat="server" Text='' />
11                  </div>
12                  <div class="form-group">
13                      <asp:Label ID="LAST_NAME_L" CssClass="control-label" runat="server" />
14                      <asp:RequiredFieldValidator ID="LAST_NAMEVal" ControlToValidate="LAST_NAMETB" CssClass
15                      <asp:TextBox CssClass="form-control" ID="LAST_NAMETB" runat="server" Text='' />
16                  </div>
17                  <div class="row">
18                      <div class="col-lg-8 col-md-8">
19                          <div class="form-group">
20                            <asp:Label ID="ADDRESS_L" CssClass="control-label" runat="server" />
21                            <asp:RequiredFieldValidator ID="ADDRESSVal" ControlToValidate="ADDRESSTB" CssCla
22                            <asp:TextBox ID="ADDRESSTB" CssClass="form-control" runat="server" Text='' />
23                           </div>
24                      </div>
25                      <div class="col-lg-4 col-md-4">
26                          <div class="form-group">
27                              <asp:Label ID="ZIP_CODE_L" CssClass="control-label" runat="server" />
28                              <asp:RequiredFieldValidator ID="ZIP_CODEVal" ControlToValidate="ZIP_CODETB" Cs
29                              <asp:RangeValidator ID="ZIP_CODEValR" runat="server" ControlToValidate="ZIP_CO
28                              <asp:TextBox ID="ZIP_CODETB" CssClass="form-control" runat="server" Text='' />
29                          </div>
```

Figur 56. Indtastningfelter

Med det yderste div, som har klassen row (l.2), opdeles div-ets indhold i 12 dele (kolonner). Det næste div har to klasser, som hver består af 3 elementer. Det første element col angiver en kolonne, den næste element angiver skærmstørrelsen og det tredje angiver kolonnes bredde. "col-lg-8" i linje 3 betyder således at kolonne på en -lg- (stor skæm >1200px) skal være bredden 8, hvilket vil sige at kolonne anvender 8/12 af MainContens bredde. Den anden af klasse (l.3) "col-md-6" angiver at kolonnen på en skærm af mellemstørrelse -md- (>992px) skal være 6/12 bred. Jeg har ikke angivet størrelsen på en lille skærm, hvilket betyder at den anvender hele bredden af MainContent. Jeg kan herefter indenfor div'et foretage en yderligere opdeling indtil jeg har den ønskede struktur på siden. Svarende til opdelingen i linje 5, 18 og 25 placeres de 4 første variable som angivet i figur 57



Figur 57. Indtastningsfelter - bredde

Indtastningsfelterne ligger alle i et div som har klassen form-group (linje 6, 12, 19 og 26). Hver form-group indeholder 3 dele (controller), label (linje 7), validering (linje 8 + 9) og input (linje 10). En control er et asp element, ud fra hvilket der generes html, css og javascrip. F. eks er der en ASP control som hedder `TextBox` som laver et html input felt af typen text (linje 10).

For alle de tekst felter som ikke indeholder normal tekst (f.eks. årstal, e-mail), er der lavet en input addon, som angiver enheden/formatet af input. Man kommer et div rundt om inputfeltet med klassen Input-group og den enhed man ønsker i en span med klasse input-group-addon (figur 58).

Figur 58. Indtastningsfelter - tekst addon

Opret bygning

Når man trykker på linket "Opret bygning" bliver der lavet en request og CustomerForm begynder at compilere. Det første der kode, fra code behind filen CustomerForm.aspx.cs, som afvikles er Page_Init. Page_Init kontrollerer om det modtagne request er et postback - kommer requestet kommer fra CustomerForm. Da dette ikke er tilfældet hentes teksten til siden, som ligger i klassen PageText. Teksten oversættes til html, css og JS og sendes den til klienten.

Tekst klasse

PageText er en klasse som indeholder en masse tekst strenge og get'ers til strengene. Den er lavet fordi firmaet godt kunne tænke sig at udvide til udlandet et sted langt ude i fremtiden. Det blev hurtigt besluttet at fokusere på målsætningen om at få lavet en energiberegning for udskiftning af vinduer. Tekst klassen er ikke anvendt andre steder i programmet.

Gem - Validering

Når kunden har udfyldt formularen og trykker på knappen "Opret" bliver siden først valideret på clientsiden. Den valideres på baggrund af javascript placeret i valideringsdelen af hver form-group.

Jeg har bruger 2 forskellige valideringstyper RequiredFieldValidator og RegularExpressionValidator. På figur 59 se en RequiredFieldValidator, som validerer feltet BUILDING_USERB. RequiredFieldValidator fungere ved at den får ControlToValidate hvorefter det kontrolleres om input feltet er udfyldt.

```
<asp:RequiredFieldValidator ID="BUILDING_USEVal" ControlToValidate="BUILDING_USERB"
    CssClass="error" runat="server" ValidationGroup="AllValidators">*</asp:RequiredFieldValidator>
```

Figur 59. RequiredFieldValidator

Den anden type validering er en RegularExpressionValidator. Den fungere som RequiredFieldValidator ved at den tildeles ControlToValidate. Forskellen er at den ikke kun kontrollerer om inputtet har et indhold, men også hvordan inputtet ser ud. Hvordan inputtet skal se ud defineres i ValidationExpressionvariablen i RegularExpressionValidator. F.eks. er ValidationExpression for de felter der skal indeholde årstal \b\d{4}\b, hvilket betyder at de skal være 4 karakterer lange.

Hvis der er valideringsregler som ikke bliver overholdt vises en pop-up liste med label på de felter hvor reglerne brydes. Udover listen tilføjes en stjerne og CSS klassen bliver sat til has-error for alle input med fejl, hvilket betyder at de bliver røde. Alle valideringsreglerne har en variable som hedder ValidationGruop som jeg har tildelt værdien AllValidators. I bunden af aspx filen ligger en control som hedder ValidationSummary det er den som laver pop-up listen når ShowMessageBox er sat til true. I ValidationSummary er ValidationGroup = AllValidators, hvilket betyder resultater at de enkelte felters validering kendes. Opretknappen har en OnClientClick attribut som er tildelt funktionen BtnClick(). BtnClick() er en javascriptmetode som resulterer i at alle felter med fejl bliver røde.

BtnClick kører Page_ClientValidate(), som først kontrollerer om der er fejl på siden. Hvis der er fejl på siden returneres false. Figur 60 vises at der herefter loopes gennem alle page_validators. For de felter hvor der findes en fejl bliver has-error tilføjet til feltets form-group.

```
for (; i < Page_Validators.length; i++) {
    if (!Page_Validators[i].isvalid) {
        $("#" + Page_Validators[i].controltovalidate)
            .parent()
            .addClass("has-error");
```

Figur 60. Validering - has-error

For felter uden fejl fjernes has-error hvis den er sat i en tidligere validering.

Når siden er korrekt udfyldt returneres den til serveren. Page_Init køres. Da requestet er en postback hentes teksten til siden ikke. Herefter køres UpdateButton_Click (fra tryk på opretknappen) som validerer siden igen, samme validering som på clientsiden, for det tilfælde at javascript er slået fra på kunden pc. Hvis der findes en fejl resulterer det i projektet alene i at man returneres til hjemmesidens forside.

Hvis valideringen returnerer sand oprettes en bruger.

OPRET BRUGER

Oprettelse af brugeren sker i metoden GenerateUser, som kun skal bruge en e-mail. Figur 61.

```
protected string[] GenerateUser(string email)
{
    MembershipCreateStatus createStatus;

    string username = GenerateRandomString(5);
    string password = Membership.GeneratePassword(7, 3);
    string[] usernamePasswordID = new string[3] { username, password, "" };
    MembershipUser user = Membership.CreateUser(username, password, email, "ikke noget", "ja", true, out createStatus);
    usernamePasswordID[2] = user.ProviderUserKey.ToString();
    switch (createStatus)
    {
        case MembershipCreateStatus.Success:
            break;
        case MembershipCreateStatus.DuplicateUserName:
            usernamePasswordID = GenerateUser(email);
            break;
        case MembershipCreateStatus.DuplicateEmail:
            break;
    }
    return usernamePasswordID;
}
```

Memberships er en klasse, som bruges til at oprette brugere. Klassen kommer fra System.Web.Security, som er en del af ASP.NET. Bruger navnet bliver genereret af i metoden GenerateRandomString, som modtager en længde hvorfra der lavers en tilfældig streng ved brug af ASP.NET Random klassen. Metoden kører iterativt så hvis brugernavnet allerede anvendes kører den igen indtil at der kan oprettes en bruger med et unikt bruger navn.

OPDATERING AF DATABASE

Når brugeren er oprettet laver laves der en ny kunde (CUSTOMER) og bygning (CUS_BUILDING) i databasen som relateres til den nye bruger. Kunden og bygningen bliver oprettet ved brug af AmondbEntities.

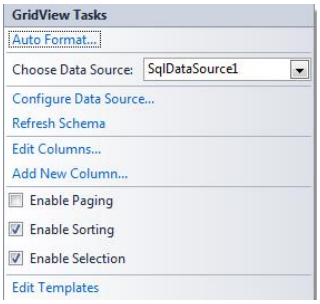
Montørform

SelectList

Når en montør skal indtaste data på et hus vælges bygningen fra en liste, denne liste heder SelectList. SelectList har 3 elementer GridView Control og SqlDataSource Control og GridView1_SelectedIndexChanged.

GridView control

Listen laves i en GridView control. En GridView er en control som bruges til at præsentere rækker af data. Jeg har anvendt Visual Studios Smart Tasks Panel til at genere koden. Smart Task panelet er vist i figur 61.



Figur 61. Smart Task Panel

SqlDataSource Control

Først tilføjes en datasource. Jeg har valgt en SqlDataSource. Jeg kunne også have brugt en EntityDataSource eller en LinqDataSource. I første omgang valgte jeg en EntityDataSource, fordi jeg bruger entitys i resten af applicationen. Det fungerede, men efter et stykke tid fik jeg en system fejl, som krævede at jeg lagde koden over i et nyt projekt. Efter jeg havde lagt projektet om et par gange, hvor det virkede i nogen tid men herefter gav samme fejl, besluttede jeg at bruge en SqlDataSource. Jeg valgte SqlDataSource frem for LinqDataSource fordi min connectionString bruges direkte, hvilket gør at der ikke er så mange steder hvor der kan ske fejl.

SqlDataSource er konfigureret så at den henter alle bygninger i CUS_BUILDING med et par af de vigtigste variable. Efter jeg havde konfigureret min SqlDataSource fik jeg mulighed for at vælge hvilke funktionaliteter gridviewet skulle have. Jeg valgte at man skulle kunne vælge en række og sortere i kolonner. Figur 62 viser den kode som er genereret af Smart Task panelet, med en enkelt tilføjelse OnSelectedIndexChanged, som bliver kørt når man vælger en bygning.

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="ID_BUILDING" CssClass="table" DataSourceID="SqlDataSource1" OnSelectedIndexChanged="GridView1_SelectedIndexChanged" A
    <Columns>
        <asp:CommandField ShowSelectButton="True" />
        <asp:BoundField DataField="ID_BUILDING" HeaderText="ID_BUILDING" InsertVisible="False" ReadOnly="True" SortExpression="ID_BUILDING" />
        <asp:BoundField DataField="BUILDING_USE" HeaderText="BUILDING_USE" SortExpression="BUILDING_USE" />
        <asp:BoundField DataField="BUILDING_TYPE" HeaderText="BUILDING_TYPE" SortExpression="BUILDING_TYPE" />
        <asp:BoundField DataField="CREATE_DATE_TIME" HeaderText="CREATE_DATE_TIME" SortExpression="CREATE_DATE_TIME" />
    </Columns>
</asp:GridView>
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:AmondbConnectionString %>"
    SelectCommand="SELECT [ID_BUILDING], [BUILDING_USE], [BUILDING_TYPE], [CREATE_DATE_TIME] FROM [CUS_BUILDING] ORDER BY [CREATE_DATE_TIME]">
</asp:SqlDataSource>
```

Figur 62. Smart Task Panel - generet kode

GridView1_SelectedIndexChanged

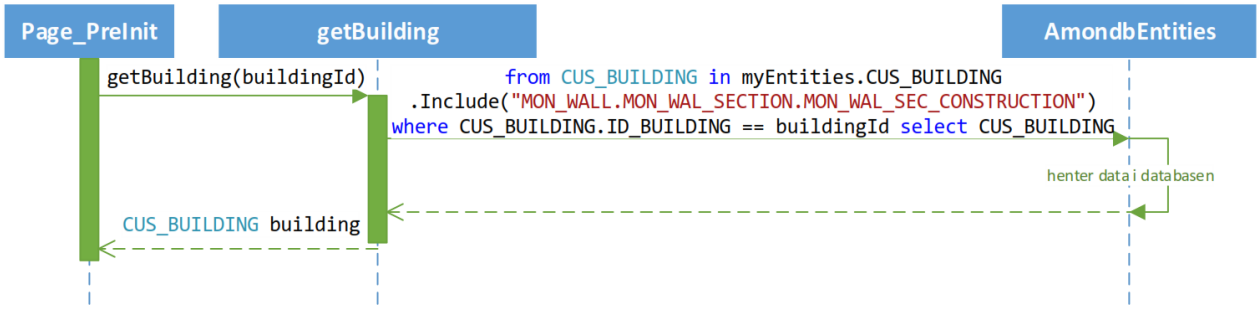
GridView1_SelectedIndexChanged er en enkel metode som redirecter til MonForm og sender en QueryString med ID'et.

Montørformularen

Montør formen er opbygget i 3 dele - bygning, vægge og vinduer. Når montøren første gang går ind på en bygning kan han kun se bygningsdelen.

Bygning

Bygningsdelen består af 3 inputfelter og en knap. Input felterne er type af hus(BUILDING_SHAPE), vinkel i forhold til Nord(BUILDING_ORIENTATION) og efterisolering(WALL_AFTER_INSULATION). Figur xx er et system sekvens diagram over det første der sker når en bruger går ind på montørformularen. Komponenterne repræsenterer klassens metoder. SSD' et viser hvordan bygningen bliver hentet og at der også hentes vægge, vægsektioner og vægkonstruktioner hvis de findes.

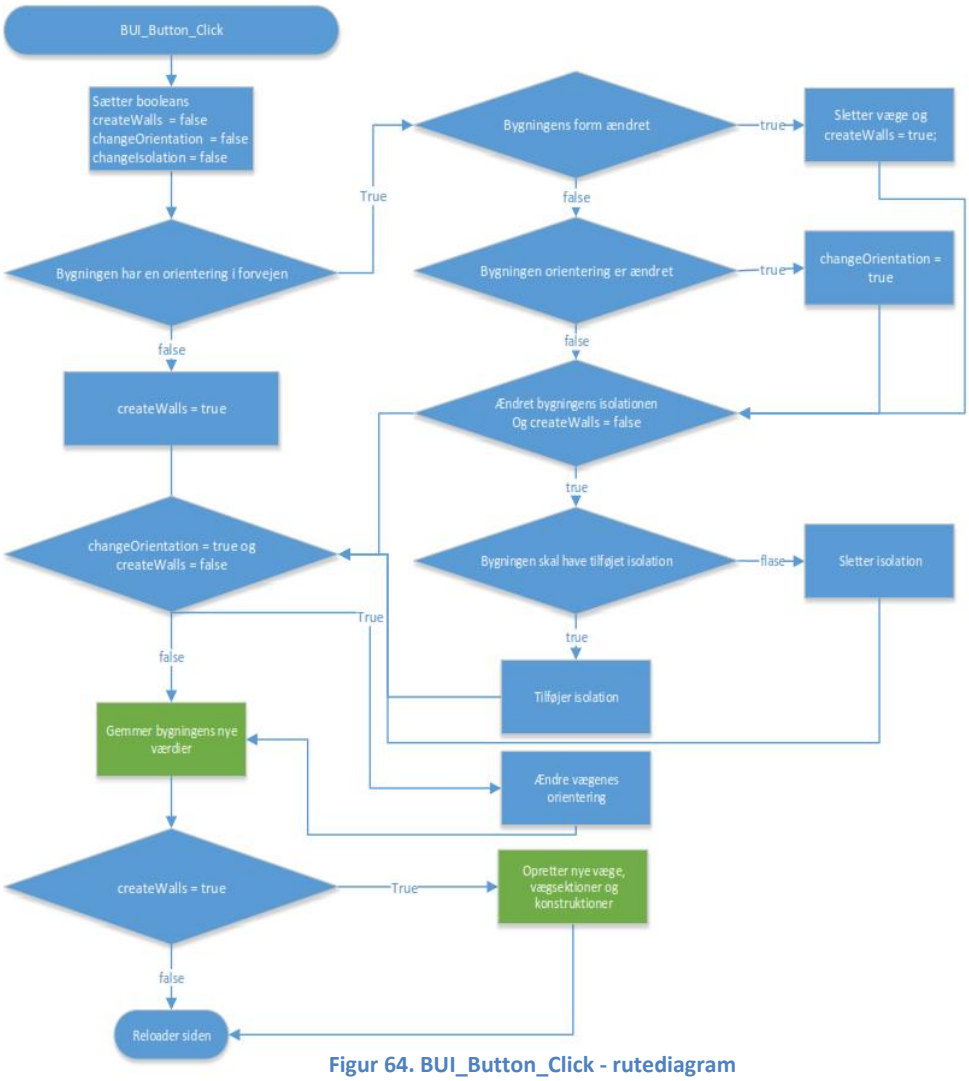


Figur 63. System Sekvens Diagram (SSD) - montør formularen hentes

Når bygningens Data er hentet kontrolleres om bygningen har fået en orientering. Hvis den ikke har fået en orientering er det første gang montøren har åbnet siden, hvilket betyder at der ikke kan være vægge eller vinduer på indtastningsformen. Til sidst bliver værdierne på bygningens 3 inputfelter sat i Page_load.

Når montøren har trykket på knappen, opdater hus og funktionen BUI_Button_Click køres.

BUI_Button_Click kan resultere i flere ting. I starten defineres 3 booleans, createWalls, changeOrientation og changeIsolation. Figur xx er et rutediagram over BUI_Button_Click. Diagrammet er kombineret af normal tekst, som beskriver hvad der sker, men jeg har også valgt medtage de 3 booleans, for at kunne beskrive hvor de bliver sat og brugt. De stadier som er markeret med grønt er dem som køres første gang montøren udfylder på væggene. De andre er med for efterfølgende at kunne ændre på bygningen når væggene er lavet.



Figur 64. BUI_Button_Click - rutediagram

Oprettelse af vægge

Når væggene skal laves køre koden set på figur xx.

```
961  Int32 i = 1;
962  int[] wallDegres = getWallDegres(BUILDING_SHAPEDDL.SelectedValue, Convert.ToInt32(BUILDING_ORIENTATIONTB.Text));
963  foreach (var Degres in wallDegres)
964  {
965      //laver en væg
966      newWall = new MON_WALL();
967      newWall.FK_ID_BUILDING = buildingId;
968      newWall.WALL_NUMBER = i;
969      i++;
970      newWall.WALL_DEGREES = Degres;
971      newWall.CREATE_DATE_TIME = DateTime.Now;
972      newWall.UPDATE_DATE_TIME = DateTime.Now;
973      //laver en section
974      MON_WAL_SECTION newSection = new MON_WAL_SECTION();
975      newSection.SECTION = 1;
976      //laver 3 lag
977      MON_WAL_SEC_CONSTRUCTION newConstruction;
978      string[] constructionLayers = new string[3] { "Inderst", "Mellem", "Yderst" };
979      foreach (string layer in constructionLayers)
980      {
981          newConstruction = new MON_WAL_SEC_CONSTRUCTION();
982          newConstruction.CONSTRUCTION_LAYER = layer;
983          newSection.MON_WAL_SEC_CONSTRUCTION.Add(newConstruction);
984      }
985      MON_WAL_SEC_ISOLATION newISOLATION;
986      if (WALL_AFTER_INSULATIONDDL.SelectedValue != "01")
987      {
988          newISOLATION = new MON_WAL_SEC_ISOLATION();
989          newSection.MON_WAL_SEC_ISOLATION.Add(newISOLATION);
990      }
991      newWall.MON_WAL_SECTION.Add(newSection);
992      building.MON_WALL.Add(newWall);
993  }
994  myEntities.SaveChanges();
```

Figur 65. Oprettelse af vægge - kode

Der er en counter ”i” som holder styr på hvilket væg nummer koden er kommet til. getWallDegrees (linje 962), som anvender bygningsformen og dens rotation, indeholder en switch-case som ser på bygningens form. Alt efter formen laver den et array som indeholder graderne på væggene, hvis bygningen vender stik nord, længden på arrayet afhænger af formen på bygningen. Derefter tilføjes bygningsrotationen til de beregnede grader. Hvis den en vægs vinkel overskrider 359 grader fratrækkes 360 fra vinklen.

Det returnerede array bliver loopet igennem og der bliver lavet en væg for hver vinkel. Væg sektionen, isolation og dens konstruktion bliver lavet i forbindelse med hver væg. Jeg bruger igen mine entities til at lave og kommunikere med databasen. Først bliver der lavet en væg, derefter en sektion og der laver tre konstruktions lag som tilføjes til sektionen (linje 983). På tilsvarende måde med isolationen. Til sidst tilføjer jeg sektionen til væggen og væggen til bygningen (linje 991 og 992). Når alle væggene og deres ”tilbehør” er lavet gemmer jeg det samlet i databasen, så der kun er at database kald.

Jeg laver en vægsektion for at kunne dele væggen op, hvis den har forskellige konstruktioner. Dette og isolationen er endnu ikke implementeret, men skal bruges senere i hovedprojektet.

Hvis en bygning har fået en ny form skal væggene ændres. Dette er implementeret ved at jeg sletter de gamle væge og laver nye, også illustreret i rutediagrammet. Figur 66 viser den kodestump der sletter væggene.

```
883  MON_WALL[] walls = building.MON_WALL.ToArray();
884  foreach (MON_WALL wall in walls)
885  {
886      MON_WAL_SECTION[] secToDelete = wall.MON_WAL_SECTION.ToArray();
887      foreach (MON_WAL_SECTION sec in secToDelete)
888      {
889          myEntities.MON_WAL_SECTION.Remove(sec);
890      }
891      myEntities.MON_WALL.Remove(wall);
892  }
893  createWalls = true;
```

Figur 66. Sletning af vægge

Bygningens vægge og sektionerne bliver loopet igennem sat at de skal fjerne. Jeg behøver ikke at angive at konstruktioner og vinduer skal fjernes. Under `OnModelCreating` (ligger i filen AmondbModel.Context.cs) som er med til at lave entitiesne har jeg angivet at når en væg slettes skal alle data knyttet til væggen også fjernes . En af de linjer der ligger i `OnModelCreating`, som fortæller at hvis en væg bliver slettet skal vinduerne også slettes, er vist herunder.

```
modelBuilder.Entity<MON_WALL>().HasRequired(s => s.MON_WAL_WINDOW).WithMany().WillCascadeOnDelete(true);
```

Indtastningsfladen til vægge og vinduer.

Når bygningen har fået en orientering og væggene er oprettet, dannes felter på formularen således at vægge og vinduer kan indtastes. Vinduer og vægge bliver lagt en på formularen i hver deres tabel. Montøren vil sandsynligvis anvende tablet når han indtaster vægge og vinduer. Derfor er det lavet sådan at væg og vinduestabellerne ikke må kommunikere med serveren under indtastningen.

Vægtabellen

Jeg ligger væggene ind først. Da væggene og deres indhold er hentet i forbindelse med bygningen behøver jeg ikke at gøre dette igen. Jeg genererer vægtabellen i Page_PreInit metoden, som kalder insertGuiWalls().

Når vægstabellen (wallTable) genereres gøres det i kronologisk rækkefølge. Initalt laves en tabel, dernæst headeren, så væg et, to og fremdeles. Tabellen er en ASP.NET control Table. Til layoutet bruger jeg bootstraps table klasse. Table classen og bredden på cellerne er dem som udgør tabellens design. Headeren dannes under anvendelse af et 2d array (headerInfos) som indeholder header teksten og cellebredden, hvor bredden er angivet med en css klasse. Jeg laver en TableHeaderRow hvor jeg tilføjer en celle af gangen, ved at loope igennem headerInfos. Figur 67 viser hvordan en headercelle bliver lavet og lagt ind i TableHeaderRow'en, i den celle i headerInfos loopet er nået til.

```
midheadcell = new TableHeaderCell();
midheadcell.Text = headerInfos[i, 0];
midheadcell.CssClass = headerInfos[i, 1];
wallHeadRow.Cells.Add(midheadcell);
```

Figur 67. Vægtabel - Headercelle

Væggene bliver loopet igennem og lagt i rækker. For hver væg dannes 3 rows, fordi at en væg kan have 3 lag. I væggens første row placeres vægnummer, orientering, højde, længde og konstruktion. Alle rækker har materiale og en tykkelse. Den sidste række har også en overflade. Cellerne kan enten indeholde et input felt eller en tekst. Alle input felterne kopieres til et Dictionary(wallsDynamicControls) som let kan loopes igennem når data skal gemmes.

Figur 69 vises koden som laver konstruktionscellen. Først laves cellen, dernæst laves en dropdownlist.

```
//Konstruktion
midcell = new TableCell();
DropDownList conDDL = constructionDDL(Convert.To
conDDL.ID = wall.WALL_NUMBER + "_" + section.SEC
wallsDynamicControls.Add(conDDL.ID, conDDL);
midcell.Controls.Add(conDDL);
midRow.Cells.Add(midcell);
```

Figur 68. Væg konstruktion - dropdownliste

Dropdownlisten bliver lavet i en metode constructionDDL. Metoden ses i figur 68.

Første laves en dropdownlist, dernæst tilføjes værdier i ListItems. Hvis der ikke har været indtastet data på

```
public DropDownList constructionDDL(string val){
    DropDownList conDDL = new DropDownList();
    conDDL.Items.Add(new ListItem("Dobbeltmur", "01"));
    conDDL.Items.Add(new ListItem("Massivmur", "02"));
    conDDL.Items.Add(new ListItem("Skalmur", "03"));
    conDDL.Items.Add(new ListItem("Skeletkonstruktion", "04"));
    if (val == "") { conDDL.SelectedValue = "01"; }
    else { conDDL.SelectedValue = val; }
    conDDL.Attributes["onchange"] = "conDDLEvents(this);";
    return conDDL;
```

Figur 69. Væg konstruktion – dropdownliste - listitems

væggene før sættes default til dobbelt mur. Til sidst tilføjer den en javascript funktion som kører når der indtastes en ny værdi. Jeg kommer ind denne funktion senere. Det er kun constructionDDL der har en javascript fuction i wallTable.

Dropdownlisten tildeles til sidst et Id. Alle Id i wallTable har samme struktur: vægnummeret _ sektionen og en forkortelse af på navnet af inputvariablen. De felter som indgår i konstruktionen for en ekstra tilføjelse i form angivelse af deres lag.

Alle input variable til walltable lavet på samme måde. Når tabellen er generet tilføjer jeg den til MainContent (figur 70).

```
Master.FindControl("MainContent").Controls.Add(wallTable);
```

Figur 70. Tilføjelse af vægtabel til MainContent

Vinduestabellen

Før tabellen laves hentes vinduerne fra databasen. Vinduerne bliver lagt i arrays, et array for hver væg. Arraysne bliver lagt i et dictionary, hvor arrayets key er vægnummeret. Hvis der ikke tidlige er indtastet vinduer hentes ingen vinduer fra databasen.

Vinduestabellen (windowTable) og dens header dannes på samme måde som beskrevet for vægtabellen. I windowTable ligger vinduerne grupperet væg hvor de sidder (derfor dictionary med arrays). For hvert vindue er der en række. Alle rækkerne som bliver genereret til tabellen får CSS klassen oldwindow. Efter hver gruppe af vinduer er der en knap (nyt vindue).

Inputfelterne og deres id bliver ligeledes dannet samme måde som inputfelterne i wallTabel, men tilføjes ikke til et dictionary, fordi at på clientsiden skal kunne tilføje nye vinduer. Alle inputfelter får en javascript funktion, som bliver kørt når der sker en ændring af feltet. Figur 71 vise at inputfeltet heiTextBox får tildelt en Javastript funktion. Jeg bruger Attributes når ASP controleren ikke har den indbyggede attribut jeg ønsker at sætte.

```
heiTextBox.Attributes["onchange"] = "setTextbox(this);";
```

Figur 71. Tildeling af Javascript

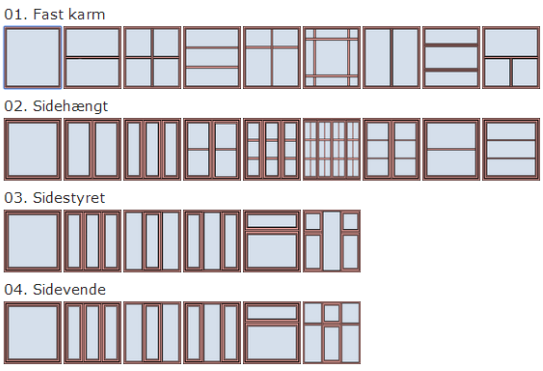
Knappen i bunden af hver vinduesgruppe er til at oprette flere vinduer på samme væg. Knappens id består af 3 dele. Del 1 ”cWB ”som står for create window button, denæst nummeret på væggen mens den sidste del det nummer det får (næste nummer i vindue gruppen), hvis det bliver tilføjet.

Når tabellen er dannet tilføjes den til MainContent.

VALG AF VINDUES VERSION

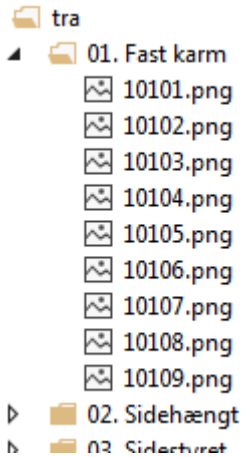
Når montøren skal angive hvilket vindue som skal udskiftes har firmaet ønsket at dette skulle baseres på valg af billeder af mulige vinduesversioner. Jeg har derfor lavet en tabel som indeholder billeder af versionerne. Tabellen bliver lavet under Page_load som kalder metoden windowPicker.

Strukturen på den tabel windowPicker laver, er en overskrift for hver række som indeholder modelnavnet. Herunder er der i rækken placeret billeder i cellerne af de versioner modellen indeholder. Et udsnit af den resulterede tabel ses i figur 72.



Figur 72. Angivelse af version af vindue

Hvert billede er lagt i en Imagebutton control, så man kan trykke på det. Strukturen bliver lavet ud fra den struktur jeg har lavet billedfilerne. Overskriften på rækkerne er hentet fra mappenavnene og id'et på billerne er hentet fra filnavnene. For at hente overskriften og id bruger jeg en kombination af regex og substring. Koden for et id er vist i figur 74. modelPic.ImageUrl er stien til billedet som jeg deler ved hver "/" for så at anvende tager jeg den fjerde del (den sidste). Substring er en funktion kan bruges på alle strenge til hente en del af strengen, hvor første parameter angiver nummeret på den karakter der skal startes med og den anden angiver hvor lang den udtagne streng skal være. Når tabellen er lavet lægges den ind i et panel.



Figur 73. Organisering af vinduesbilleder

```
Regex.Split(modelPic.ImageUrl, @"/")[4].Substring(0, 5);
```

Figur 74. Id for det valgte billede

Klientsiden

Det første der sker når siden er blevet loadet er javascriptfuktionen window.onload kører. Window.onload tager det div, som panelet med vinduesbillederne ligger i og laver det til en dialog som gemmes. En dialog er en jquery funktion, som laver en et pop-up vindue. I bunden af window.onload er linje "return=false" som betyder at der ikke returneret noget til serveren. Alle funktioner har denne return=false linje.

FUNKTIONALITET I VÆGTABELLEN

I vægtabellen køres en funktion som kaldes konstruktionen af en væg ændres. Konstruktionens dropdownlist kalder funktionen hvor den giver "sig selv" med som argument. Funktionen ses i figur 75.

Funktionen finder den del af id'et, som beskriver hvilken væg dropdownlisten tilhører og placeres den i en variabel i linje 51. Hvis dropdown værdien er værdien for massiv mur (linje 52) bliver det yderste og mellemste konstruktionslag skjult (hide) og feltet for overflade vises (show) som en del af den inderste række (linje 53-56). Hvis værdien ikke er for massiv mur vises alle 3 konstruktionslag og kun overflade er synlig for det yderste lag.

```
50      function conDDLEvents(e) {
51          var wallCon = e.id.substring(0, 15);
52          if (e.value == "02") {
53              $("#" + wallCon + "Mellem_row").hide();
54              $("#" + wallCon + "Yderst_row").hide();
55              $("#" + wallCon + "surMel").hide();
56              $("#" + wallCon + "surInd").show();
57
58          }
59          else {
60              $("#" + wallCon + "Mellem_row").show();
61              $("#" + wallCon + "Yderst_row").show();
62              $("#" + wallCon + "surInd").hide();
63              $("#" + wallCon + "surMel").hide();
64              $("#" + wallCon + "surYde").show();
```

Figur 75. Funktion - ændring af vægkonstruktion

FUNKTIONALITET I VINDUESTABELLEN.

Alle felterne kalder en funktion når deres værdi ændres, som betyder at værdi sættes den i html koden. Tekst boksene kører funktionen setTextbox, dropdownlisterne kører setSelected og tjekboksene kører set-CheckBox. Ved alle funktionskald giver felterne "sig selv" med som argument.

Funktionen setTextbox ses i figur 76. Det eneste den gør er værdien value til den nye.

```
function setTextbox(e) {
    $("#" + e.id).attr("value", e.value);
```

Figur 76. Funktionen setTextbox

Funktionen setSelected gemmer index på den valgte værdi, fjerner selected attributten fra listen og sætter den nye.

```
function setSelected(e) {
    var theSelected = e.options[e.selectedIndex].value;
    $('#' + e.id + " option").removeAttr('selected');
    $("#" + e.id + " option[value='" + theSelected + "']").attr("selected", "selected");
```

Figur 77. Funktionen setSelected

Når montøren trykker i feltet hvor vinduesversionen kan vælges køres funktionen pick_window (figur 78). Det første pick_window gør er at den sætter den globale variable windowmodel_boxVar til id'et på den versionstekstboks der har kaldt funktionen. Derefter åbner den dialog boksen som indeholder billederne af vinduerne (figur 72).

Når et vindue er valgt kaldes funktionen window_picked (figur 78). I linje 164-5 bruges den globale variabel windowmodel_boxVar til at gemme id for vinduesversionen. Derefter lukkes dialogboksen.

```
157         var windowmodel_boxVar;
158         function pick_window(e) {
159             windowmodel_boxVar = e.id;
160             $("#windowDialog").dialog("open");
161             return false;
162         }
163         function window_picked(e) {
164             $("#" + windowmodel_boxVar).attr("value", e.id.substring(12, 18));
165             $("#windowDialog").dialog("close");
166             return false;
167         }
```

Figur 78. Pick_window og Window_picked

Hvis en af knapperne "Nyt vindue" i vinduestabellen bliver klikket køres funktionen newWindow. (Figur 79)

```
67  function newWindow(e) {
68      $("#" + e.id).closest("tr").before("<tr>" +
69                  "<td>" +
70                  e.id.substring(15, 16) +
71                  "</td>" +
72                  "<td>" +
73                  e.id.substring(17, 18) +
74                  "</td>" +
75                  "<td>" +
76                  "<select id=MainContent_winexi" + e.id.substring(15, 18) + " onchange=setSelected(this); >" +
77                      "<option selected=selected value=01>1-lags glas</option>" +
78                      "<option value=02>2-lags termorude</option>" +"<option value=03>3-lags termorude</option>" +"<option value=04>&ldr
79                  "</select>" +
80                  "</td>" +
81                  "<td>" +
82                  "<input type=text id=MainContent_winhei" + e.id.substring(15, 18) + " class=ncell onchange=setTextbox(this);>" +
83                  "</td>" +
84                  "<td>" +
85                  "<input type=text  id=MainContent_winlen" + e.id.substring(15, 18) + " class=ncell  onchange=setTextbox(this);>" +
86                  "</td>" +
87                  "<td>" +
88                  "<input name=testboxID type=text id=MainContent_winver" + e.id.substring(15, 18) + " onclick=pick_window(this); >" +
89                  "</td>" +
90                  "<td>" +
91                  "<select id=MainContent_winver" + e.id.substring(15, 18) + "  onchange=setSelected(this); >" +
92                      "<option selected=selected value=01>Tinium 2+ 2-lags C</option>" +
93                      "<option value=02>Sapino 2-lags C</option>" + "<option value=03>Mogano 2-lags C</option>" +  "<option value=04>Tin
94                  "</select> " +
95                  "</td>" +
96                  "<td>" +
97                  "<select id=MainContent_winfra" + e.id.substring(15, 18) + " onchange=setSelected(this); >" +
98                  "<option selected=selected value=65>65</option>" +
99                  "<option value=75>75</option>" +
100                 "<option value=85>85</option>" +
101             "</select>" +
102                 "</td>" +
103                 "<td>" +
104                 " <input id=MainContent_winkeh" + e.id.substring(15, 18) + " type=checkbox onchange=setCheckbox(this); /> " +
105                 "</td>" +
106                 "<td>" +
107                 "<input id=MainContent_winnly" + e.id.substring(15, 18) + " type=checkbox onchange=setCheckbox(this); />" +
108                 "</td>" +
109                 "<td>" +
110                 "<input id=MainContent_winnbu" + e.id.substring(15, 18) + " type=checkbox onchange=setCheckbox(this); />" +
111                 "</td>" +
112             "</tr>");
113     var nextWindow = parseInt(e.id.substring(17, 18));
114     nextWindow = nextWindow + 1;
115     $("#" + e.id).attr("id", e.id.substring(0, 17) + nextWindow);
```

Figur 79. Tilføjelse af nyt vindue

I linje 68 ses at funktionen bruger jquerys closest til at identificere den række knappen sider i. Herefter angives at den efterfølgende kodes resultat skal placeres i rækken over knappen. Den efterfølgende kode

laver en ny vinduesrække med alle dens input og funktionaliteter. Knappens id anvendes til at lave id for de nye felter. Når den nye vinduesrække er lavet ændres knappens id, hvor der til det sidste tal i id adderes 1.

Gem vægge og vinduer

Når montøren er færdig med at udfylde formularen trykker han på knappen med teksten "Gem vægge og vinduer". Gem knappen har to klick attributter, en som afvikles på klientsiden og en som køres på serversiden. Den på klient siden kører getNewWindowTable, som tager html'en fra hele vindues tabellen og placere den i et gemt (hidden) tekstboks felt.

SERVER – VÆGGE OG VINDUER

På serveren kører Gem knappen knappens klickattribut update**Wall**and**Windows** . Jeg vil ikke gå i detaljer med vægdelen som er forholdsvis lige til. Der looper gennem dictionaryet wallsDynamicControls, som blev lavet under genereringen af vægtabellen.

Når vinduerne skal gemmes anvendes den streng som blev lavet i på klientsiden af getNewWindowTable. Stengen hentes fra det skjulte felt. Ved anvendelse regular expressions hentes data fra strengen. Først deles strengen op svarende til <tr og placeres i et array (figur 80). Arrayet svarer til de rækker der var i tabellen

```
526    string[] windowRows = Regex.Split(newWindowHtml.Text, "<tr");
```

Figur 80. Vinduesrækker findes

Derefter loopes gennem rækkerne. Strengen deles nu svarende til cellerne (<td>), og placeres i et array. Herefter kontrolleres længden på hvert array. Hvis der ikke er 11 celler (12delinger) i en række bliver den ikke brugt. Hvis en række har 11 celler skal data gemmes.

Først findes væggen vinduet sidder i (celle 1) og vinduets nummer (celle 2). Dernæst laver jeg et vindues objekt. Hvis rækken indeholder teksten oldwindow er det et vindue der har været indtastet før, data vinduet hentes fra databasen og tilføjes vinduesobjektet.

Nu skal dataen hentes ud af inputfelterne. Jeg har lavet 3 metoder som henter værdierne ud fra inputfelterne. En for tekstbokse(`getTBValue`), en til dropdown (`getDDLValue`) og en til tjekbokse (`getCBValue`). Figur 81 viser metoden til dropdownlister.

Tekststrengen opdeles ved <option og lægges i et array. Så loopes dropdownlisten muligheder igennem i det jeg undersøger hvilke der er blevet selected ved brug af ved brug af regex isMatch, som returnerer en boolean(linje 691) . Hvis en mulighed er blevet selected hentes dens værdi, linje 693. Værdien findes ved at bruge af en regularexpression som siger at jeg gerne lige vil have det tal der ligger mellem value=" og ";

```
685    string getDDLValue(string cell)
686    {
687    string value = "";
688    string[] dropdownOptions = Regex.Split(cell, "<option");
689    foreach (string dropdownOption in dropdownOptions)
690    {
691        if (Regex.IsMatch(dropdownOption, @"selected"))
692        {
693            value = Regex.Match(dropdownOption, @"(?<=value=""+)(\d+)(?="")").Groups[0].Value;
694        }
695    }
696    return value;
```

Figur 81. Metoden getDDLValue til dropdownliste

Hvis alt er indtastet korrekt gemmes data for vinduerne i databasen. Hvis der er felter der ikke er udfyldt korrekt vises med rødskrift på siden at der er sket fejl og vinduerne gemmes ikke.

Rapport

Rapport listen

ReportList fungerer på samme måde som selectlisten beskrevet under montør, men med tre forskelle. Figur 82 viser query for de 2 lister

```
"SELECT [ID_BUILDING], [BUILDING_USE], [BUILDING_TYPE], [UPDATE_DATE_TIME] FROM [CUS_BUILDING]"

'SELECT [ID_BUILDING], [BUILDING_TYPE], [UPDATE_DATE_TIME], [CREATE_DATE_TIME] FROM [CUS_BUILDING] WHERE ([COMPLETED] = @COMPLETED) ORDER BY [UPDATE_DATE_TIME]
```

Figur 82. Query for select og rapportlist

For det første forskel er at der linkes til vinduesrapporten - WindowReport. For det andet indeholder rapporten ikke alle bygninger, men kun de bygninger hvor er gemt vægge og vinduer i databasen. Det fungere ved at den tjekker på om completet variablen på bygning er blevet sat til true. Completet bliver sat når vinduer og vægge gemmes. Slutteligt indeholder rapportlisten ikke de samme felter.

Vinduesrapporten (WindowReport)

WindowReport.aspx indeholder intet kode ud over det som leveres fra Master.Site. Det betyder at alt hvad der vises på siden er generet i C#.

WindowReport hentes ved at trykke på bygningens energirapport i Rapportlisten (select). Når der linkes til siden medsendes fra ReportList en querystring med bygningens id. I WindowReport's Page_Load metode oversættes querystrengen til et tal og der laves et objekt af klassen WindowCalculations. WindowCalculations har en metode der hedder drawWindows.

drawWindows

DrawWindows er en funktion som tager bygningen id som argument. Funktionen "tegner" alle de elementer som anvendes til at tegne bygningens vinduer. Den returnerer en liste af objekter af typen WindowDraw. WindowDraw er en klasse med 98 variable. Klassen indeholder alle informationer om vinduerne, herunder alle informationer der er nødvendige for at tegne vinduerne og udregne energibesparelsen ved at udskifte et vindue. Først defineres i drawWindows en række variable, dernæst hentes data i databasen som anvendes i drawWindows til at lave WindowDraw objekterne. Først hentes vægge, vinduer og tabellen INF_WIN_ENERGY_COMPASS. Hele INF_WIN_ENERGY_COMPASS bliver hentet med det samme for at minimere kommunikationen med databasen.

Alt det data der skal bruges i WindowCalculations henter jeg i toppen af drawWindows, da det er lettere at bevare overblikket når al databasekommunikation ligger samlet. Efter at det første data er hentet, startes et loop, som køre væggene igennem. Dernæst loopes over vinduerne. Alle loops er implementeret med foreach. På figur 83 viser det første stykke af vindesloopet.

```
62  foreach (MON_WAL_WINDOW window in windows)
63  {
64      var shapes = from OUT_WIN_SHAPES in myEntities.OUT_WIN_SHAPES where OUT_WIN_SHAPES.FK_ID_WINDOW_VERSION == window.FK_ID_WINDOW_VERSION select OUT_WIN_SHAPES;
65      INF_WIN_CERTIFICATE certificate = (from INF_WIN_CERTIFICATE in myEntities.INF_WIN_CERTIFICATE where INF_WIN_CERTIFICATE.ID_WIN_CERTIFICAT == window.FK_ID_CERTIFICAT select INF_WIN_CERTIFICATE).Single();
66
67      foreach (OUT_WIN_SHAPES shape in shapes)
68      {
69          INF_WIN_EXISTING existingWindowValues = (from INF_WIN_EXISTING in myEntities.INF_WIN_EXISTING
70                                                   where INF_WIN_EXISTING.WINDOW_EXISTING_GLAS == window.WINDOW_EXISTING_GLAS && INF_WIN_EXISTING.GRP_EXISTING == shape.GRP_EXISTING
71                                                   select INF_WIN_EXISTING).Single(); // skal laves så den ikke henter hver gang men henter til heleruden
72          shapeDictionary = new Dictionary<string, int>();
73          //Ligger værdier i dictionary så men kan regne med strenge
74          shapeDictionary.Add("WINDOW_WIDTH", window.WINDOW_WIDTH.Value);
75          shapeDictionary.Add("WINDOW_HEIGTH", window.WINDOW_HEIGTH.Value);
```

Figur 83. drawindow - loop over vinduer

Først i loopet hentes alle de former (shapes) som skal anvendes, for at man kan tegne vinduet. Dernæst hentes vinduernes certifikater og certificerede energiværdier tabellen INF_WIN_CERTIFICATE. Certifikat værdierne bruges ikke i drawWindows. De bliver lagt i WindowDraw objekterne hvor de anvendes under energiberigningerne.

Derefter loopes over alle de fundne shapes og værdier for hver enkelt shape beregnes. Første i loopet henter værdier for de gamle vinduer, som også først skal anvendes under energiberegningen. Den resterende kode for shape loopet kan inddeles i 3 dele.

I del 1 placeres en række værdier i et dictionary. Jeg laver et dictionary fordi nogle de variable som anvendes til at beregne værdier for shapes er strenge som indeholder kode. Jeg har så lavet en funktion (StringEquationExecutor) som kan køre disse strenge som kode.

StringEquationExecutor vises i figur 84. StringEquationExecutor burger NCalc, som er en pakke jeg har hentet gennem Visual Studios NuGet Packages Manager. NCalc har en klasse Expression, som kan bruges til at køre indholdet af strenge som kode. Klassen kræver at alle variable afgrænset af firkantede paranteser (square brackets) og at alle variable tilføjes til klassen parameter dictionary (Parameters).

```
1406  public int StringEquationExecutor(Dictionary<string, int> shapeDictionary, string stingexpression)
1407  {
1408      Expression ex;
1409      foreach (KeyValuePair<string, int> pair in shapeDictionary)
1410      {
1411          if (Regex.IsMatch(stingexpression, pair.Key))
1412          {stingexpression.Replace(@"\b" + pair.Key + "\b", "[" + pair.Key + "]"); }
1413      }
1414      try { ex = new Expression(stingexpression);
1415      }
1416      catch{ return 0;}
1417      foreach (KeyValuePair<string, int> pair in shapeDictionary)
1418      {
1419          ex.Parameters[pair.Key] = pair.Value;
1420      }
1421      return Convert.ToInt32(ex.Evaluate());
```

Figur 84. StringEquationExecutor

I linje 1409 til 1413 tilføjes square brackets til den streng der skal køres som kode. Linje 1414 laves et objekt af klassen Expression med den nye streng. Parametrene tilføjes i linje 1417-1420 og til sidst returneres den værdi som er et resultat af at kodestumpen er blevet kørt.

Del to af shape loopet indeholder en lang række if sætninger, som beregner de præcise værdier der skal anvendes for at tegne vinduerne. If sætningerne indeholder formler for hvordan vindueselementerne skal tegnes. Et eksempel på en IF sætning ses i figur 85, som laver et array med 3 vinduesversion id'er. Koden tjekker om den aktuelle shape har et af disse id'er, shapen position og om shape typen er horisontal sprosse. Alle disse if' er som tegner vinduet skal senere struktureres i nogle switch-cases når man er sikker på at man har få data fra alle vinduesabrikanterne.

```
if (new[] { "10701", "10602", "10208" }.Contains(shape.FK_ID_WINDOW_VERSION))
{
    if (shape.SHAPE_TYPE == "SPH" && shape.SHAPE_POSITION.Value == 1)
    {
        SHAPE_WIDTH = window.WINDOW_WIDTH.Value - shape.RR.Value - shape.DDPPLL.Value;
        SHAPE_X = shape.R.Value + shape.DPL.Value;
        SHAPE_Y = WMSH - WINDOW_SPROSSE_WIDTH / 2;
    }
}
```

Figur 85. IF sætning som beregner sprosseværdier for 3 vinduesversioner

I den sidste del af shapes loopet bliver der lavet at WindowDraw objekt og objektet bliver lagt i en liste. WindowDraw objekt har en construktor som modtager alle variablene på en gang.

Listen bliver returneret til WindowReport. WindowReport kalder så metoden window_energi_calculate der ligger i windowCalculate. Window_energi_calculate som skal have WindowDraw objekterne angivet listen med.

WINDOW_ENERGI_CALCULATE

window_energi_calculate er en metode som en laver en liste af WindowEnergy objekter. Objekterne indeholder de beregnede energiværdier. Værdierne bliver udregnet ved at den efter hvert vindue sorterede liste af WindowDraw objekter loopet således arealerne af vinduets forskellige elementer bliver lagt sammen. Et element kunne være ruder eller sprosser. Sammenlægningen af arealerne ses i figur 86. Når alle arealer for vindue er fundet udregnes de variable der bruges for at beregne Uw og Ew værdierne. Uw og Ew værdier bliver beregnet til sidst. Energibalancen for hvert vindue før og efter renoveringen beregnes og energibesparelsen findes.

Listen med WindowEnergy bliver returneret til WindowReport som laver den om til en tabel og viser den ud til brugeren.

```
foreach(WindowDraw WD in WDs){
    shapeCount++;
    //finder samlede arealer
   if (WD.SHAPE_TYPE == "MODEL")                    { MODEL_AREAL   = (WD.SHAPE_HEIGTH/1000.0) * (WD.SHAPE_WIDTH/1000.0);}
   if (WD.SHAPE_TYPE == "POSTV")                    { POSTV_AREAL   = POSTV_AREAL + (WD.SHAPE_HEIGTH/1000.0) * (WD.SHAPE_WIDTH/1000.0);}
   if (WD.SHAPE_TYPE == "POSTH")                    {POSTH_AREAL    = POSTH_AREAL + (WD.SHAPE_HEIGTH/1000.0) * (WD.SHAPE_WIDTH/1000.0);}
   if (WD.SHAPE_TYPE == "RUDE" )                    { RUDE_AREAL    = RUDE_AREAL + ((WD.SHAPE_HEIGTH/1000.0) * (WD.SHAPE_WIDTH/1000.0));}
   if (WD.SHAPE_TYPE == "SPH" || WD.SHAPE_TYPE =="SPV")    {SPROSSE_AREAL  = SPROSSE_AREAL + ((WD.SHAPE_HEIGTH/1000.0) * (WD.SHAPE_WIDTH/1000.0));}
   if (WD.SHAPE_TYPE == "RUDE" )                    { RUDE_OMKREDS  = RUDE_OMKREDS + ((WD.SHAPE_HEIGTH/1000.0) * 2) + ((WD.SHAPE_WIDTH/1000.0) * 2);}
```

Figur 86. Sammenlægning af arealerne i et vindue

7. TEST

Programmet er løbene blevet testet med en form for White Box Test som jeg kalder ”hul igennem test”, ”debug” test og afsluttende er der gennemført en Explotatory Test.

HUL IGENNEM TEST

En whitebox test er når man tester en del af programmet uafhængigt af resten af programmet. På grund af programmets struktur er det vanskeligt teste et udsnit af koden uden at resten af koden kører. Derfor er testene blevet kørt løbende som en slags hul igennem test, hvor jeg først har lagt et lille element ind som jeg så har testet. Når det mindre kodestykke fungerer, tilføjes et nyt mindre element som så testes. Når de små elementer virker tilføjes gradvist større kodeelementer.

Hvis en tilføjelse ikke virker, går jeg tilbage og tester at de tidligere tilføjede elementer forsat virker små elementer stadig virker hvorefter et det nye mindre element tilføjes i en ændret udgave. På den måde har jeg opbygget programmet og løbende testet med større og større elementer/opgaver.

Da jeg ikke har dokumenteret mine hul igennem test undervejs vil jeg i stedet beskrive nogle eksempler.

Eksempel 1. Kommer den rigtige tekst ind på kunne formen.

1. kan jeg lave en string som bliver vist på kunne formen.
    a. hvis ja, videre til trin 2
2. kan lave en string i min tekst klasse hente den og få den ud på kundeformen
    a. hvis ja, videre til trin 3
3. det samme som trin 2 men nu med 2 variable
    a. hvis ja, videre til trin 4
4. det samme som 2 og 3 nu bare med mange flere variable.
    a. Hvis ja lig alle variable ind og test dem eller gå et trin tilbage og set en enkelt variable på afgangen.

Eksempel 2. Bliver en kunde oprettet med bruger og bygning.

1. Kan jeg oprette bruger hvor jeg har indtastet hans værdier i koden

```
81E  Membership.CreateUser("testperson", password, "test@mail.com", "ikke noget", "ja", true, out createStatus);
```

2. Kan jeg oprette en bruger med mine autogenererede data
3. Kan jeg oprette en bruger og en kunde der relatere til hinanden.
4. Kan jeg oprette en bruger, en kunde og en bygning der relatere til hinanden.

Eksempel 3. Vindues indtastnings form

1. Kan jeg lave en indtastningsformular hvor montøren kan tilføje nye elementer.
2. kan jeg undgå at serveren og klient siden kommunikere undervejs.
3. kan jeg bruge de elementer der er tilføjet på klient siden på serveren
4. Kan jeg gemme dem i databasen
5. Kan jeg hente værdierne fra databasen
6. Vil montøren kunne se de tidligere indtastede værdier når montørformularen for bygningen åbnes igem

”DEBUG” TEST.

Er hvor jeg debugger og hele tiden beskriver for mig selv hvad der skel ske i næste linje og hvad der skal komme ud af den

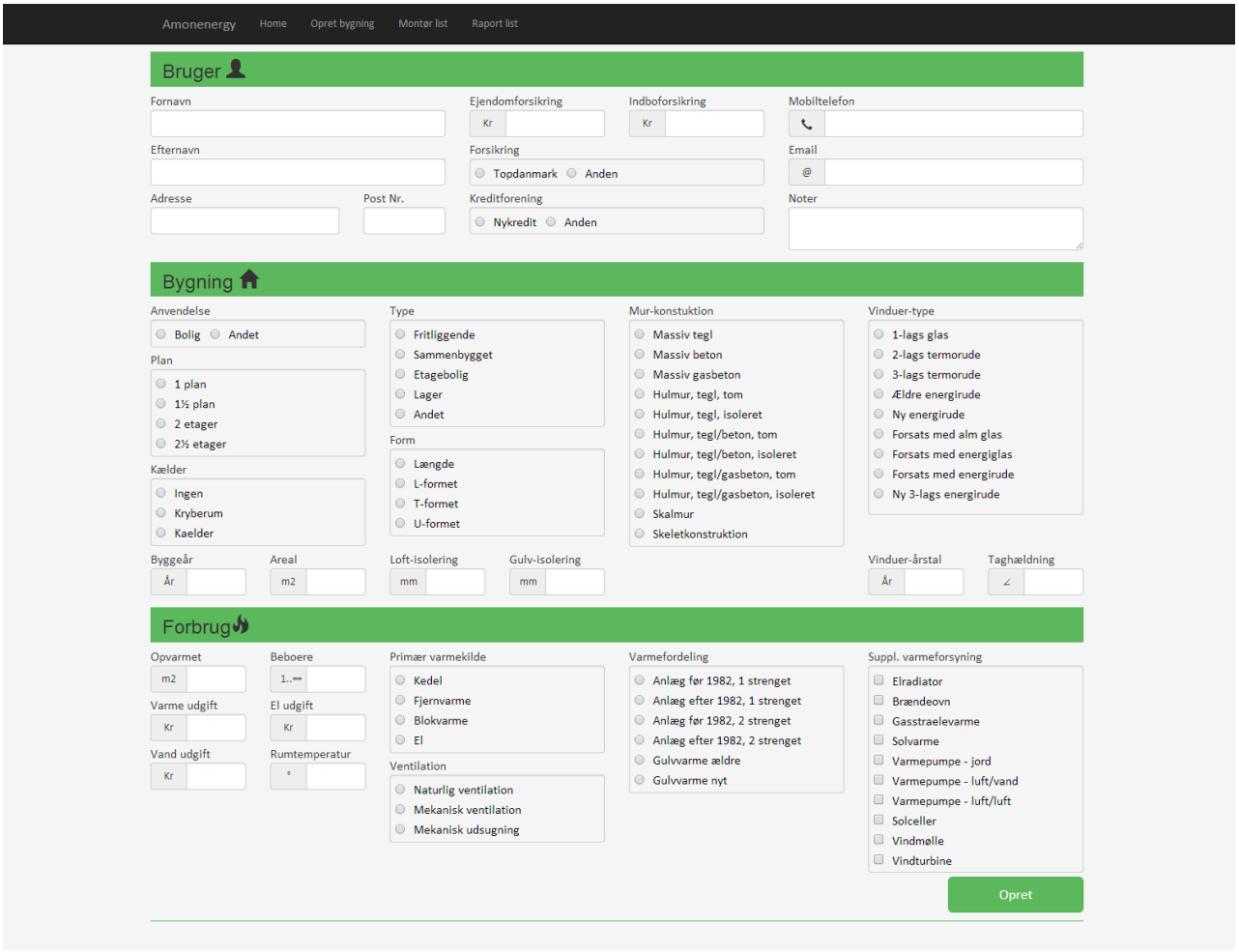
EXPLOTATORY TEST.

En explotatory test er en test metode hvor programmet afprøves på alle de mulige de måder man kan finde på.

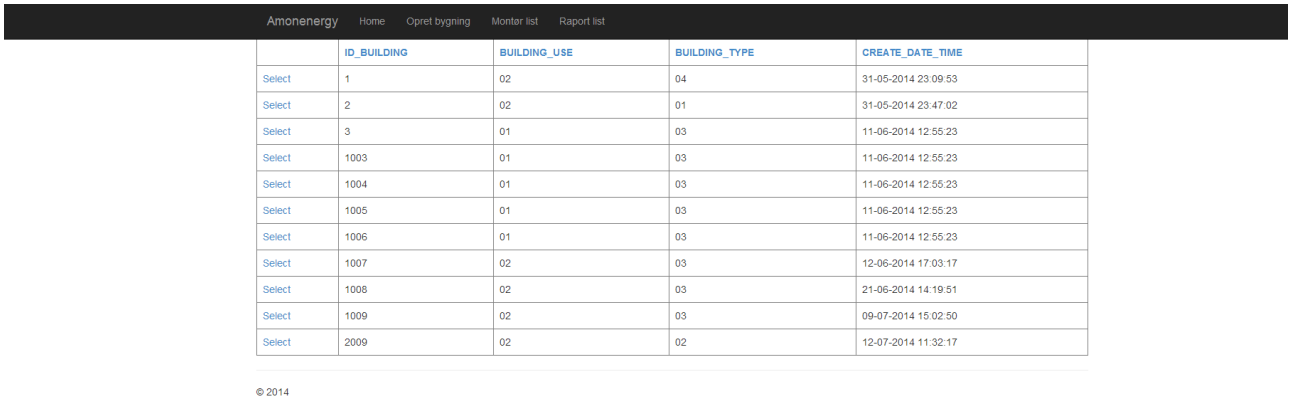
Hvad bliver testet	Hvordan testes det	Resultat af testen
Validering af kundeformen	Prøver at lave flere forskellige indtastnings fejl og prøver det samme hvor scripts er slået fra.	Virker efter hensigt
Bliver en bygning oprettet under kunde formen	Opretter kunde og bygning i kunde formen og ser om byg-ningen kan ses i Montør listen.	Virker efter hensigt.

8. SCREENSHOT

BRUGERFORMULAR

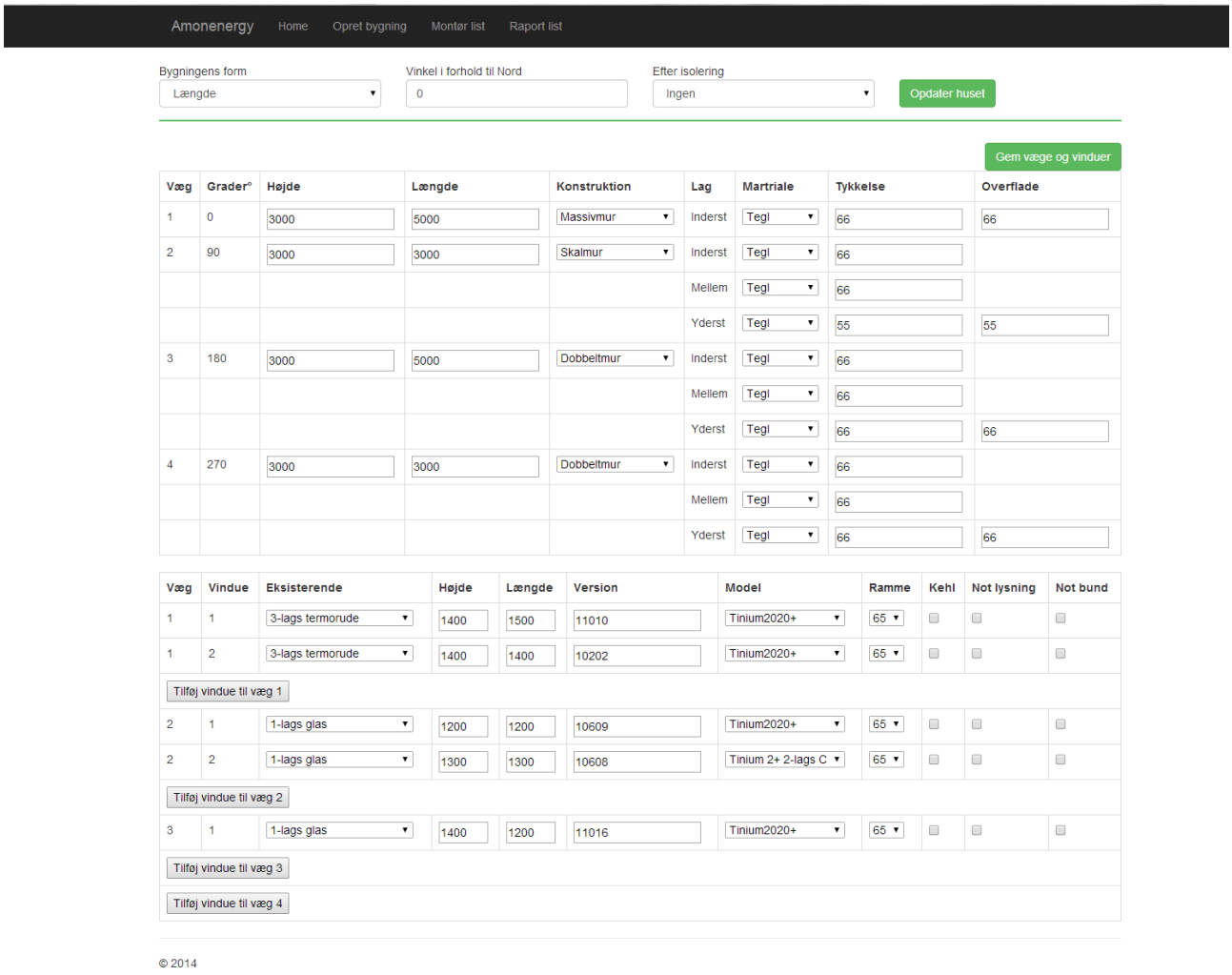


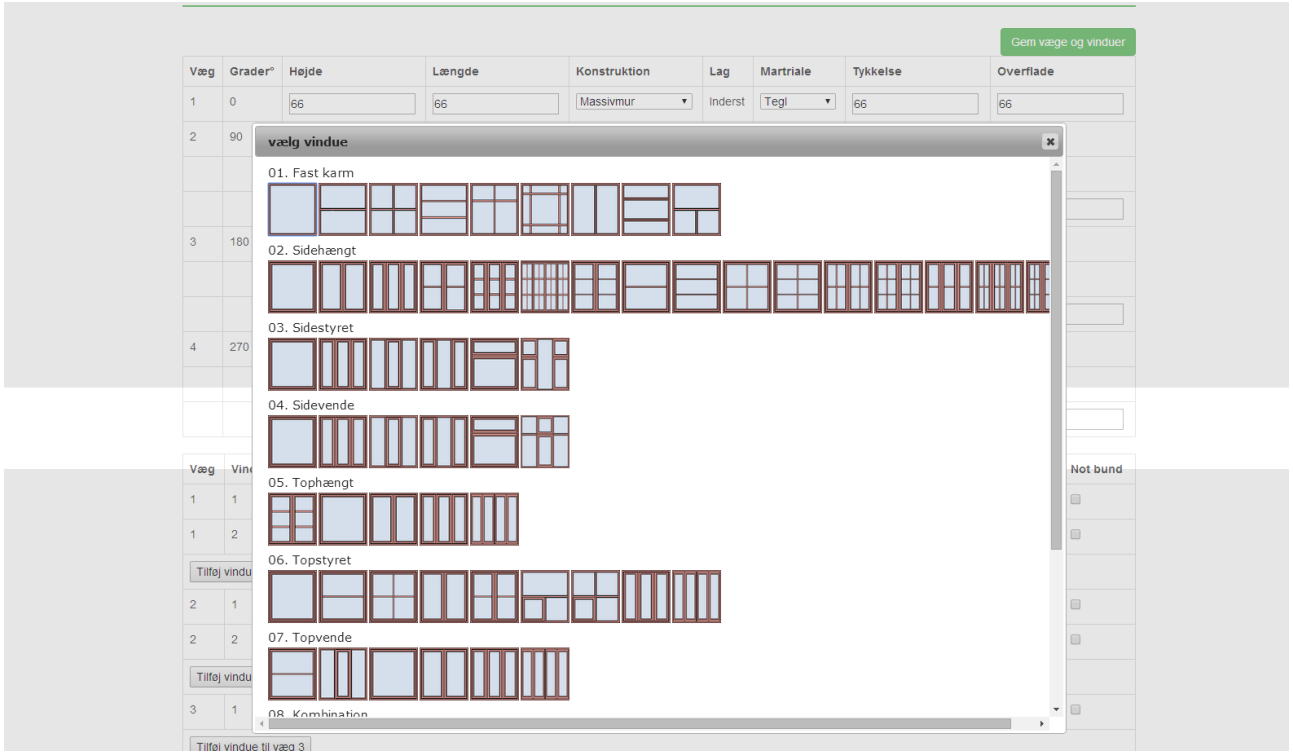
SELECTLISTE



	ID_BUILDING	BUILDING_USE	BUILDING_TYPE	CREATE_DATE_TIME
Select	1	02	04	31-05-2014 23:09:53
Select	2	02	01	31-05-2014 23:47:02
Select	3	01	03	11-06-2014 12:55:23
Select	1003	01	03	11-06-2014 12:55:23
Select	1004	01	03	11-06-2014 12:55:23
Select	1005	01	03	11-06-2014 12:55:23
Select	1006	01	03	11-06-2014 12:55:23
Select	1007	02	03	12-06-2014 17:03:17
Select	1008	02	03	21-06-2014 14:19:51
Select	1009	02	03	09-07-2014 15:02:50
Select	2009	02	02	12-07-2014 11:32:17

MONTØR FORMULAR





ENERGI RAPPORT

Amonenergy Home Opret bygning Montør list Raport list

Orientering	Nummer	Versions navn	VERSION	STM_ID	Højde	Bredde	Modelareal	Glas andel	Type	Uw (Wm2K)	Energi balance efter (kWhm2)	Energi balance før	Energi bespareIse
0	1	1 fags Dannebrog med 10 glas	DAN_10	DB1-EX23/22	1500	1400	2,1	0,61	Tinium2020+	0,91	-54,43	-228,86	174,43
0	2	2 fags sidehængt	SH_02	SH2-1	1400	1400	1,96	0,66	Tinium2020+	0,95	-55,89	-207,95	152,06
90	1	Mangler	TS_05	TS4	1200	1200	1,44	0,4	Tinium2020+	1,2	-68,54	-333,58	265,04
90	2	Mangler	TS_04	TS3	1300	1300	1,69	0,54	Tinium 2+ 2-lags C	1,49	-80,12	-314,61	234,49
180	1	3 fags Dannebrog med 3x10 glas	DAN_16	DB3-EX23/22	1200	1400	1,68	0,38	Tinium2020+	1,09	-27,24	-272,25	245,01

Samlet energi besparrelse er 1071,04 kWh

© 2014

9. BILAG OG KILDER

Bilag 01. Kontraktudkast

Bilag 02. Projektbeskrivelse

Bilag 03. Vinduer - begreber og energiberegning

Bilag 04. Vinduers varmetab

Bilag 05. Bilag til håndbog for energikonsulenter 2008, version 3

Bilag 06. Vinduer - design