1 Års Projekt

08-06-2018

Erhvervsakadamiet Lillebelt (Vejle) - VEADM17ED17A

Alex Madsen - Daniel Merbt - Jonas Bill Lundqvist - Patrick Lundsgaard

Indledning

Sweethome har brug for at skræddersyet administrativt it-system til at lette arbejdsbyrden i dagligdagen. Dette program skal indeholde funktioner til at vise salgsstatistikker over deres tidligere boligsalg, samt over kvm. Priser af tidligere solgte ejendomme i specifikke områder. Programmet skal desuden indeholde en prisberegner som kan give mæglerne en ide om hvad prisen på en ejendom i et specifikt område skal koste baseret på ejendommens størrelse og områdets prisfaktor. Herudover skal programmet indeholde en funktion som skal hjælpe mæglerne med fordeling af ejendommene i forbindelse med åbent hus, denne funktion skal fordele 30 ejendomme ud på 6 mæglere så alle får en af de 6 dyreste ejendomme mm. Programmet skal indeholde mulighed for CRUD af mæglere, husejere og ejendomme. Til sidst skal programmet kunne vise tiden for London og Danmark på samme tid, da de arbejder sammen med internationale mæglere.

Systemudvikling

Iterationsplan - UP og Projektledelse/Projektdeltagelse

Unified Process

Discipline	Artifact	Inception	Elaboration		Construction		Transition
	Iteration	1	2	3	4	5	6
Requirements							
	Kravsdokument	s	r	r			
	Vision	s	r	r			
	FURPS+	s	r	r			
	Use Case Model	s	r	r			
	Use Case Beskrivelse	s	r	r			
	Glossary	s	r	r			
	SSD-diagrammer		s	r			
Project management							
	Business case	s	r				
	Iterationsplan	s	r				
Business Modeling							
	Domain Model		s	r			
	ER-Diagram		s	r			
Design							
	Use case implementering		s	r			
	Prototype GUI		s	r			
	Class Diagram		s	r			
	Database Design		s	r	r	r	
Testing							
	Test Model			s	r	r	
Implementation							
	Implementation Model			s	r	r	r

Projektet strækker sig over 6 uger - 1-6 i UP planen ovenfor - og faserne har vi vægtet, så vi har anslået at elaboration og construction faserne bliver de tungeste i forhold til arbejdsbyrden. Derfor har de to faser en varighed på 10 dage, eller 2 uger, delt op i 2 iterationer. Tidsplanen over iterationerne findes herunder.

Iterationsplan

Iterationsplan (tidsplan)							
	UP faser	Inception	Elaboration		Construction		Transition
	Iteration	1	2	3	4	5	6
	Varighed (dage)	5	5	5	5	5	5
	Ugeangivelse (Ugen for iterationens start)	18	19	20	21	22	23

Kravsdokument (Daniel)

Sweet Home Ejendomsmæglerne er et traditionelt ejendomsmæglerfirma, som formidler salg af ejendomme i Danmark. Firmaet samarbejder med andre ejendomsmæglere i Danmark og udlandet i forbindelse med formidling af salg af ejendomme.

Vision

Sweet Home Ejendomsmæglernes arbejdsgang er hovedsageligt manuelle med lidt assistance af Office-programmer. Sweet Home Ejendomsmæglerne har brug for et informationssystem, som kan administrere en stor del af deres kunderettede opgaver.

Sweet Home's system mht. til hus oplysninger osv. skal primært ligne den måde som ejendomsmæglerkæden Home Vejles hjemmeside viser.
Systemet skal vise tidszonerne for København og London, grundet firmaets internationale samarbejde.

Oftest starter et salg med, at Sweet Home kontaktes af en husejer, som vil sælge sit hus. Der tilbydes herefter en - ofte gratis og uforpligtende - vurdering af huset af en af Sweet Home's ejendomsmæglere.
I en vurdering indgår dels huset selv: og eksempelvis dets type, antal etager, antal kvadratmeter, alder, nye eller gamle køkkener og badeværelser, ombygningsår osv.
En eventuel have kan også spille en rolle for prisen. Men derudover er en faktor som beliggenheden yderst afgørende for salgsprisen. Ejendomsmæglerne afgør værdien af beliggenheden ud fra fx bydel, eventuel udsigt til fjord eller ådal, afstand til skole og indkøb mv.
Faktisk ønsker ejendomsmæglerne et system, der umiddelbart kan afgøre, om en given bolig ligger i et meget attraktivt-, et ikke attraktivt- eller et neutralt område af byen.

Når vurderingen er foretaget, kan sælgeren vælge at lade Sweet Home overtage arbejdet med at sælge ejendommen.
Ethvert salg køres som en sag. En sag afsluttes, når en ejendom er blevet solgt, eller når sælgeren eventuelt ønsker at lukke sagen eller skifte til et andet mæglerfirma.

Så længe sagen kører, vil mæglerne tilbyde forskellige ydelser til hus sælgeren. Ud over vurderingen kan det være annoncering på hjemmesiden, i boligtillæg og bolig aviser. En anden vigtig serviceydelse er også Sweet Home's "Åbent Hus" arrangementer, som tilbydes helt gratis for sælger. Det er selvfølgelig også fremvisning af huset til mulige købere, og det kan være hele dokument arbejdet i forbindelse med salg og overtagelse (skøde, tinglysning, refusionsopgørelse, overdragelse af nøgle osv.).

Når huset er solgt (eller samarbejdet ophører), får mægleren sit salær (en procentsats af salgsprisen) samt diverse gebyrer for udformning af dokumenter.

Der lægges særligt vægt på, at systemet kan håndtere funktionalitet som:
• Oprettelse, ændring og sletning af en sag vedr. ejendomssalg.
• Prisberegning/vurdering, der tager højde for områdepris faktoren.
• Opstillinger af salgsstatistikker, der viser hvor mange boliger, der er solgt i hvilke områder af byen og til hvilken pris (samt hvornår og af hvilken mægler).

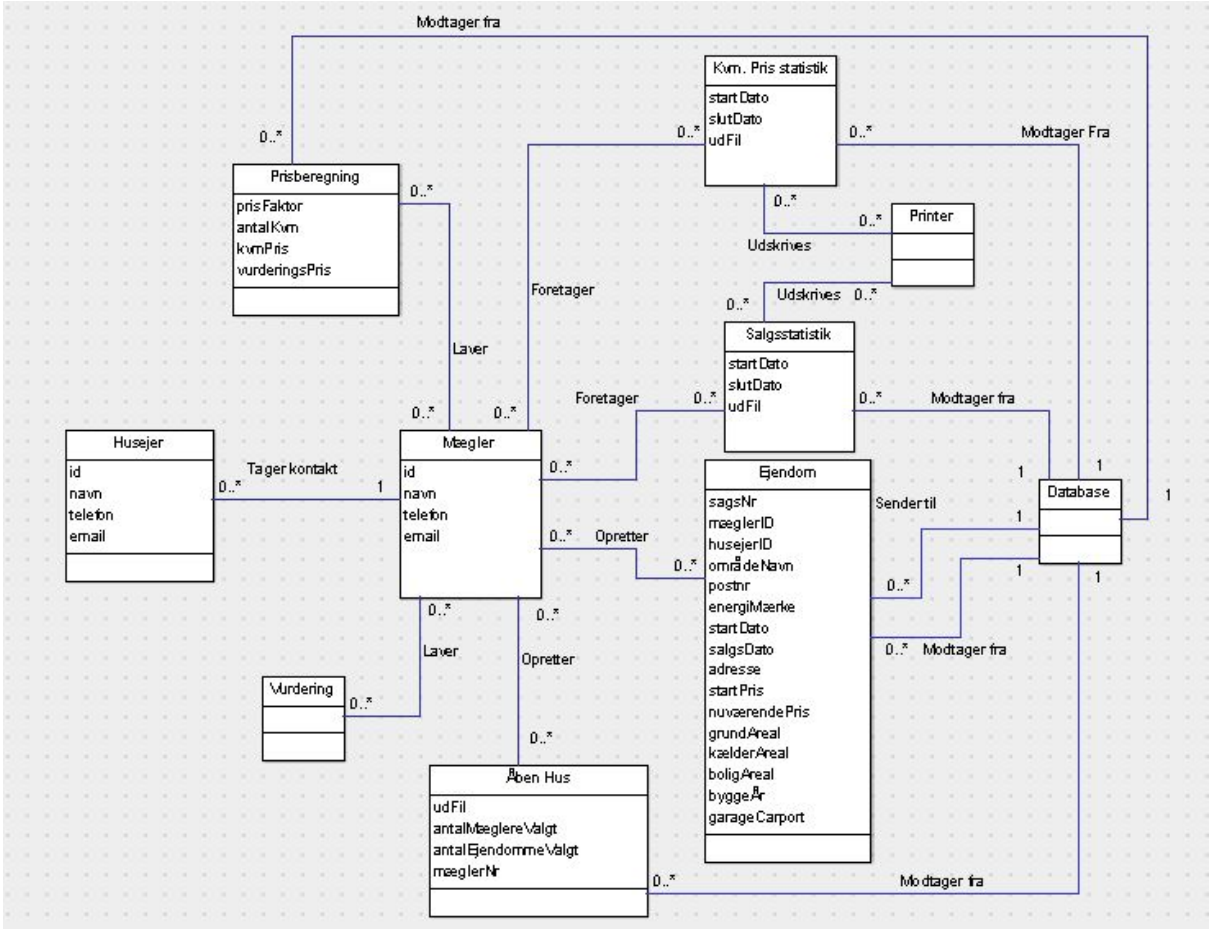
Versions Styring (Jonas)

Til at hjælpe med projektet har vi valgt at bruge versionsstyring, vi har valgt at bruge github, samt github udvidelsen til visual studio som gør det nemt at udføre commit, push og pull til og fra vores repository.
https://github.com/JonasAlexDanielPatrick/SemesterProjekt2

Proces Dagbog

- Inception (Påbegyndt d. 30-04-2018)
    - 02-05-2018: Kravsdokument og visionsdokument påbegyndt.
    - 02-05-2018: FURPS+ påbegyndt.
    - 02-05-2018: Use Cases påbegydt.
    - 02-05-2018: Domæne model påbegyndt.
- Elaboration (Påbegyndt d. 07-05-2018)
    - 08-05-2018: ER Diagram påbegyndt.
    - 08-05-2018: GUI prototype påbetyndt.
    - 08-05-2018: Klassediagram påbegyndt.
    - 08-05-2018: Business case påbegyndt.
    - 15-05-2018: Database mapning påbegyndt.
    - 17-05-2018: SSD Påbegyndt.
- Construction (Påbegyndt d. 18-05-2018)
    - 19-05-2018: Implementering af multithreaded ur afsluttet.
    - 20-05-2018: Implementering af Salgsstatistik afsluttet.
    - 22-05-2018: Implementering af Kvm. Priser afsluttet.
    - 29-05-2018: Implementering af Pris Beregner afsluttet.
    - 01-06-2018: Implementing af Åbent Hus afsluttet.
    - 04-06-2018: Implementing af CRUD afsluttet.
- Transition (Påbegyndt d. 04-06-2018)
    - Afsluttet d. 07-06-2018

Domænemodel (Patrick)



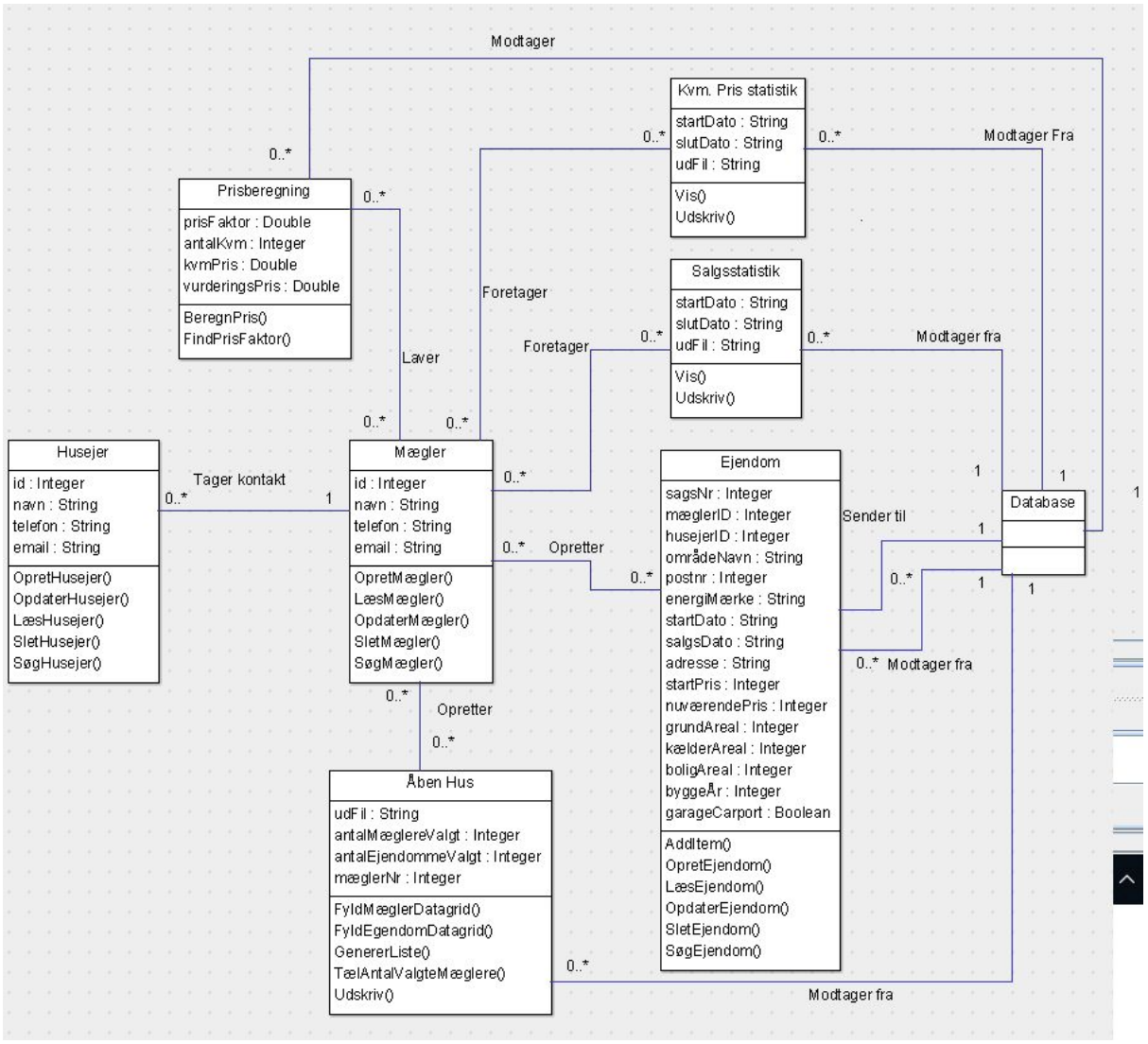
Læseretning:

Husejer tager kontakt til en mægler, som så vil lave en prisberegning af huset for at kunne give husejeren et bud på hvor prisen nogenlunde ligger. Prisberegningen modtager data fra databasen for at kunne lave beregningen. Mægleren vil derefter tage en vurdering og oprette en sag, som sendes til Databasen.

Mægleren kan også foretage en Kvm. Pris statistik eller en Salgsstatistik som begge modtager data fra Databasen, som så skrives op på papirform, så det eventuel kan udskrives på en printer.

Mægleren kan oprette Åben Hus som modtager data fra Databasen.

Klasse diagram (Patrick)



Mægler kan i programmet lave en Prisberegning ud fra attributterne i klassen. Programmet kalder metoden BeregnPris() som så kalder metoden FindPrisFaktor() som henter data fra databasen, for at få fat i prisFaktor værdien. Så fortsætter BeregnPris() metoden processen, laver beregningen og giver mægleren en vurderingsPris.

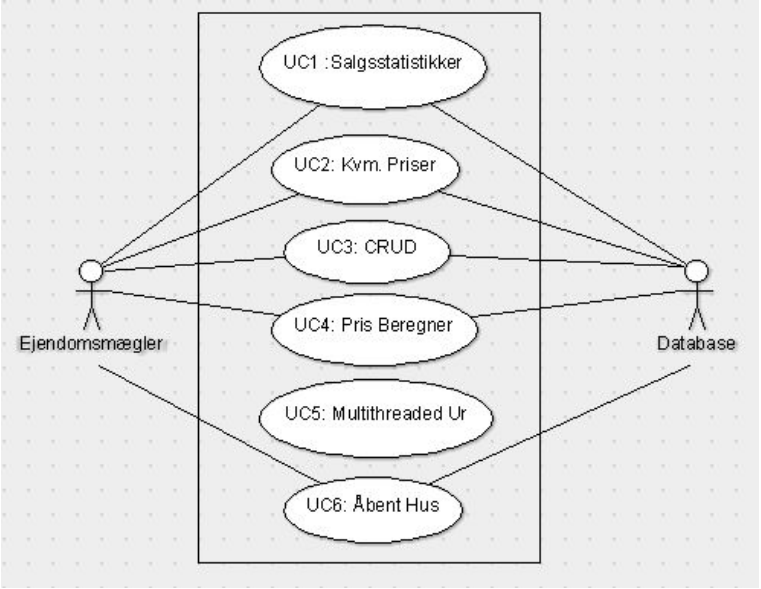
Mægleren kan foretage en Kvm. Pris statistik eller en Salgsstatistik, som begge har de samme metoder Vis() og Udskriv(). Metoden Vis() viser data på skærmen, som modtages fra databasen. Metoden Udskriv() udskriver den data på papirform.

Mægleren kan oprette Åben Hus som bruger metoderne FyldMæglerDatagrid() og FyldEgendomDatagrid() for at vise mægleren datagridet på skærmen. Mægleren kan i Programmet trykke på en udskriv knap som så kalder metoden GenererListe(), som så også kalder TagCheckedMæglerOgEjendomme() og Udskriv().

Der er CRUD metoder på Husejer, Mægler og Ejendom, samt en søg metode, som bliver kaldt når knappen Søg bliver trykket.
AddItem() metoden i Ejendom klassen bliver kaldt en gang i MainWindow klassen. Den giver data fra databasen til en combobox.

Use Cases (Alex)

Use Case Diagram



Use Case Beskrivelse

Brief:

UC1: Salgsstatistikker:
Viser solgte boliger i område af by og pris mm. Som kan gemmes og blive udskrevet.

UC3: CRUD:
Ejendomsmægler foretager CRUD på ejendomme, ejendomsmæglere & husejere.

UC5: Multithreaded Ur:
Et multithreaded ur som vises i TT:MM for København og London.

Casual:

UC2: Kvm. Priser:
Ejendomsmægler vælger en given måned og årstal, herefter trykkes på "søg" hvorefter programmet så vil vise en potentiel udskrift at den givne statistik over kvm. priser på solgte ejendomme på den given dato.
Hvis ejendomsmægler ønsker den valgte måned udskrevet, trykkes der på "Udskriv" og programmet vil udskrive statistikken.

UC4: Pris Beregner:
Ejendomsmægler indtaster kvm på boligen og der vælges en by og et område hvor i boligen ligger, og alt efter hvor, så vil der være en områdepris faktor tildelt det given område.
Ejendomsmægler trykker på "Beregn Pris" og der vil så blive vist en vejledende pris/vurdering som der kan tillægges den given bolig.

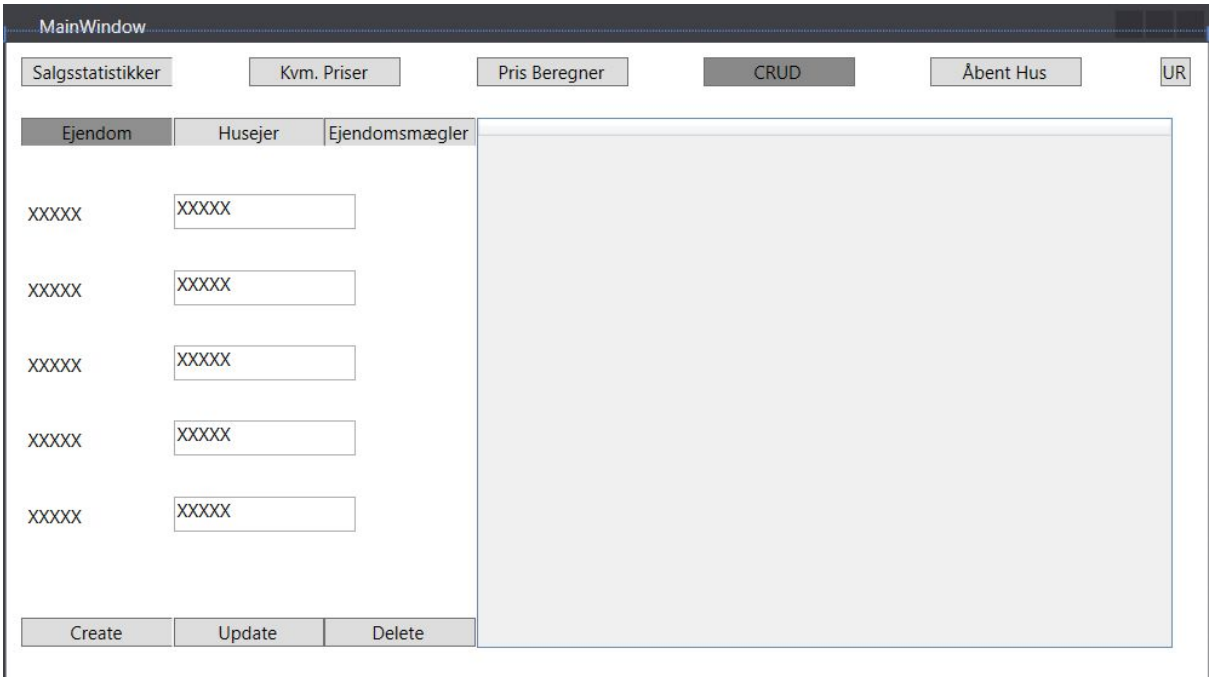
Fully dressed:
UC6: Åbent Hus:
Åbent hus er en funktion hvor der skal findes 6 ejendomsmæglere og 30 ejendomme, hvorpå de 30 ejendomme skal fordeles ud på de 6.

Programmet starter med at vise en kolonne med alle ejendomsmæglere og en kolonne med alle ejendomme fra databasen.
Ejendomsmægler/user vælger minimum 6 ejendomsmæglere til et maximum af 6 ejendomsmæglere fra ejendomsmægler kolonnen, herefter vælges der minimum 30 ejendomme og maximum 30 ejendomme fra ejendoms kolonnen.
Der trykkes nu på "Generere", og hvis kravene ikke er opfyldt vil der komme en pop-up som fortæller at der mangler ejendomsmæglere/ejendomme, eller at der er valgt for mange ejendomsmæglere/ejendomme hvorefter fejlen skal rettes.
Hvis kravene er opfyldt og der trykkes på "Udskriv", så bliver alle 30 ejendomme lagt ind i en list, hvorefter alle ejendomme bliver sorteret efter pris fra højeste pris til laveste pris.
Så bliver ejendomsmæglerne tildelt en placering fra 1-6 og alle ejendomme i deres sorteret form.
Alle ejendommene bliver så smidt ind i et loop som tildeler hver ejendomsmægler en ejendom af gangen til hver ejendomsmægler.
Når loopet er kørt og hver ejendomsmægler er tildelt 5 ejendomme, så gemmes mæglerne og de tildelte ejendomme i et .txt dokument.

Use-case implementeringsplan

	Inception	Elaboration		Construction		Transition
Use-case / Iteration	1	2	3	4	5	6
UC_1: Salgsstatistikker					s	r
UC_2: Kvm. Priser					s	r
UC_3: CRUD				s	r	r
UC_4: Prisberegner				s	r	r
UC_5: Multithreaded ur					s	r
UC_6: Åbent hus				s	r	r

Prototype GUI (Alex)



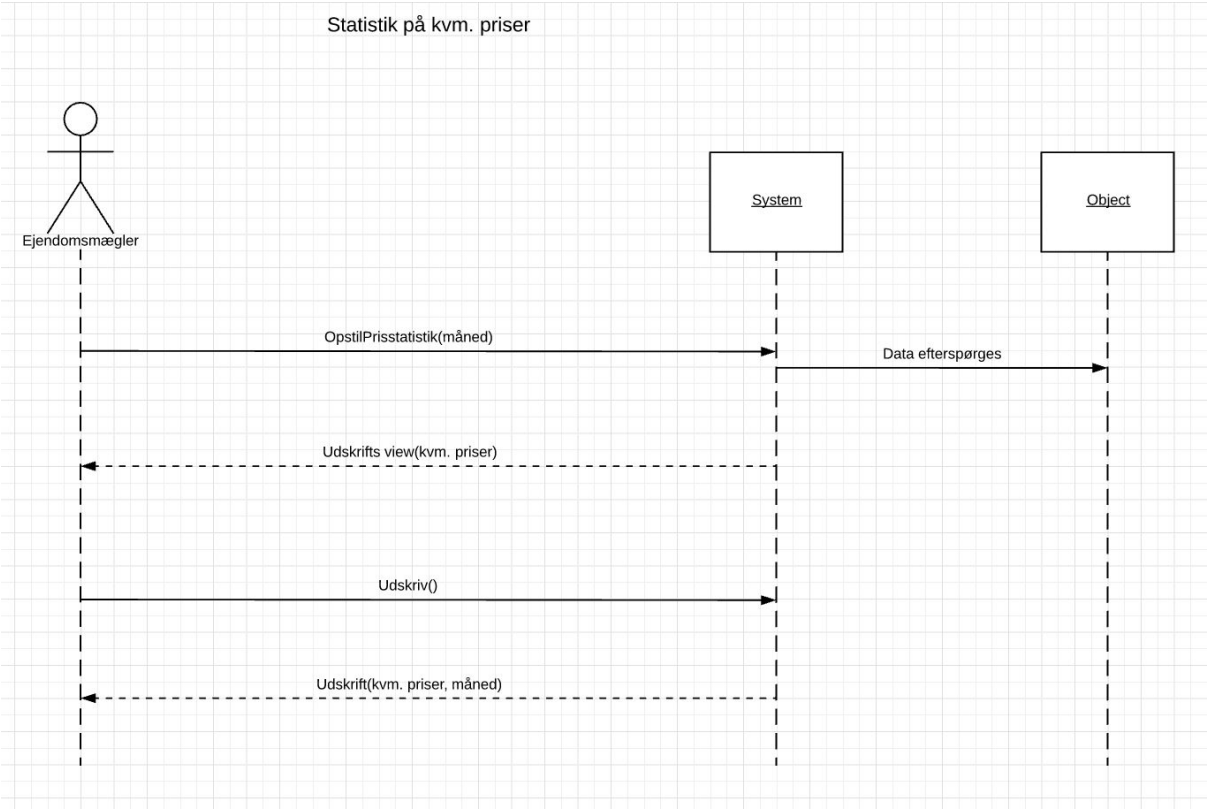
Brugervenlighed

Der er god brugervenlighed, det er overskueligt da der er tekst tilknyttet hver knap/tekstfelt, hvilket gør det let at læse/forstå hvad knappen/tekstfeltet gør.
Det er nemt og overskueligt at se hvad der høre til hvad, da programmet er inddelt i faner.
De knapper og tekst der høre sammen står tæt ved hinanden så der ikke skal ledes efter tingene, man kan sige de kommer i naturlig rækkefølge.
De nævnte ting gør at vi overholder "Loven om nærhed" og "Loven om lukkethed".
Vi kommer ikke så meget ind på reglen om kontrast, fordi vi ikke bruger nogle billeder og programmet ikke er til kunder men personalet.

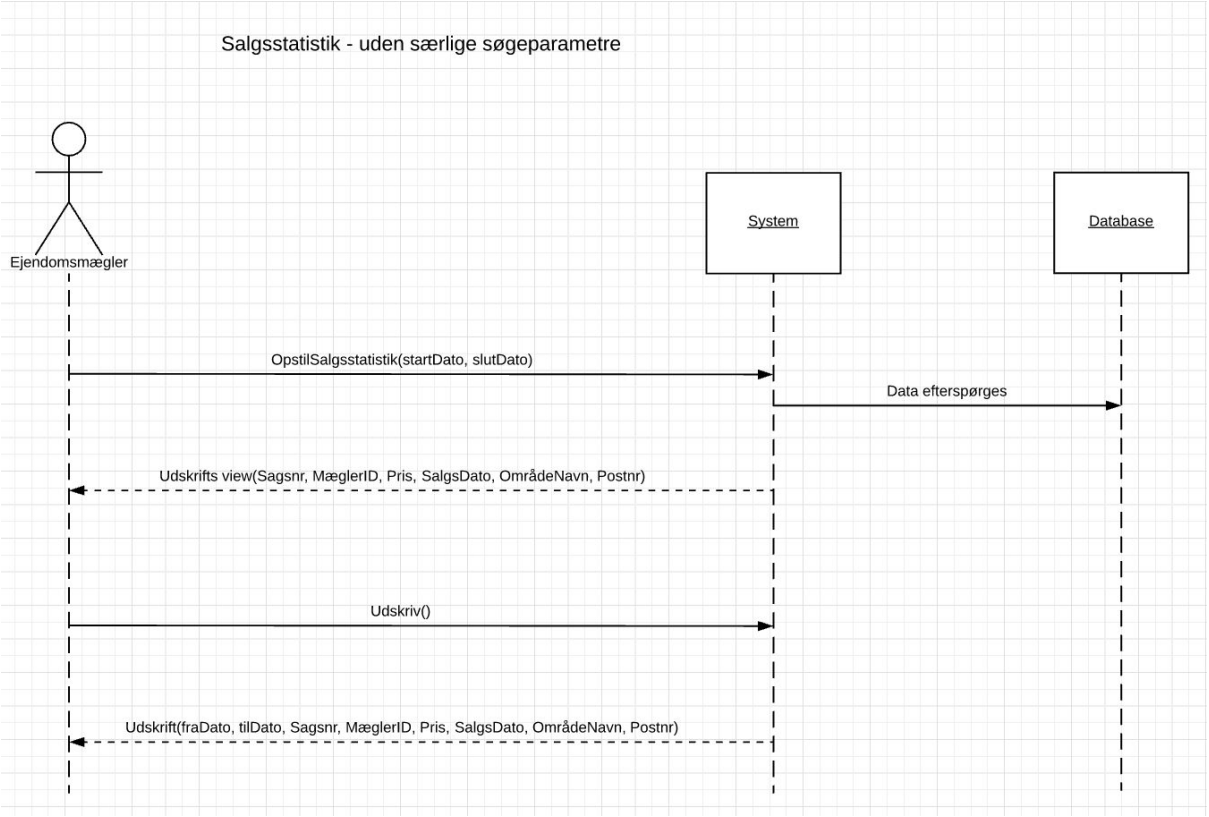
Test af brugervenlighed

Brugervenligheden kunne testes ved at få en medarbejder hos Sweethome til at prøve programmet for første gang uden at sige et ord, give vedkommende en række opgaver som skal løses uden hjælp fra programmøren.
Der er mange inspektions måder man kan foretage sig, ovenstående er blot et eksempel.
På den måde vil man finde ud af hvor brugervenligt programmet er i løbet af programmerings fasen. Afslutningsvis, vil en gennemgang af programmet finde sted i transition-fasen, så brugerne bliver instrueret i programmets funktioner.

Systemsekvensdiagrammer (Daniel)



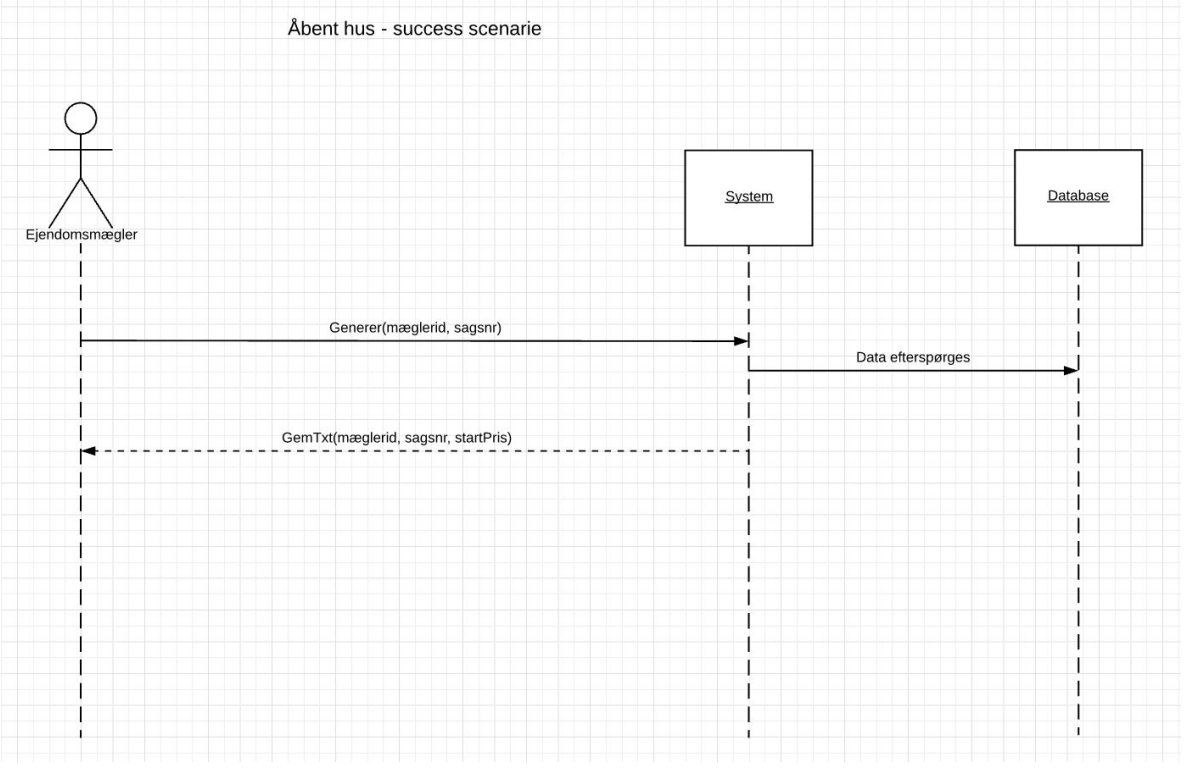
Statistik på kvm. priser
Der laves et metodekald - OpstilPrisstatistik(måned) - som efterspørger data fra databasen. Der returneres et Udskriftsview med kvm. Priser.
Efterfølgende kaldes Udskriv() og der laves en udskrift af statistikken.



Salgsstatistik - uden særlige søgeparametre

OpstilSalgsstatistik(startDato, slutDato) kaldes og databasen får en efterspørgsel om data. Der returneres et udskriftsview.

Efterfølgende kaldes Udskriv og der returneres en udskrift.



Åbent hus - success scenarie

Generer(mæglerid, sagsnr) kaldes og data efterspørges i databasen. Der returneres en txt fil.

Design Patterns (Jonas og Alex)

I forbindelse med ControllerUr klassen bruges et singleton pattern, dette bruges til at skabe en instance af vores view, da ControllerUr klassen har brug for at tilgå dette objekt for at kunne opdatere det. Her er det vigtigt at vi kun har et objekt da det altid er det samme view som skal opdateres.

Arkitektur (Jonas)

I projektet har vi valgt en MVC arkitektur. Her er vores models repræsenteret af constructor klasser samt vores database, vores view er repræsenteret af vores WPF application, og vores controllers er repræsenteret af resten af vores klasser som alle er navngivet med "Controller".
MVC er valgt på baggrund af dens simplicitet samt at den passer godt på denne type opgave. Arkitekturen giver et godt og tydeligt overblik over projektet, og hjælper derved med at holde projektet simpelt.
For at holde arkitekturen ligger models i namespaced Models, controllers ligger i namespaced Controllers, og viewet ligger i namespaced Main, da dette ikke kan omdøbes.

FURPS+ (Jonas)

Der ønskes en fornuftig grad af afkobling til SQL-databasen. (Hvor skal dette være?)

Functionality

- Der skal kunne udføres CRUD på ejendomme i systemet
- Der skal være en prisberegner som tager højde for en områdepris faktor
- Der skal være en "Åbent Hus" funktion i systemet, som kan uddele ejendomme til mæglerne så alle får en af de dyreste boliger, så de næst dyreste osv.
- Der skal kunne laves statistikker over salg af boliger i specifikke områder
- Der skal kunne laves statistikker over kvm. pris på solgte ejendomme i en given måned.
- Systemet skal indeholde et ur som viser klokken i Danmark og i England

Usability

Der skal laves en brugergrænseflade som skal bruges af ejendomsmæglerne, den skal indeholde alle de ovenstående funktionaliteter.

Reliability

- Der er ikke stillet krav til oppetid, men da systemet skal bruges til hverdag af mæglerne, forventes der ikke meget nedetid.

Performance

- Urene som viser klokken i Danmark og England skal kører på sin egen tråd.

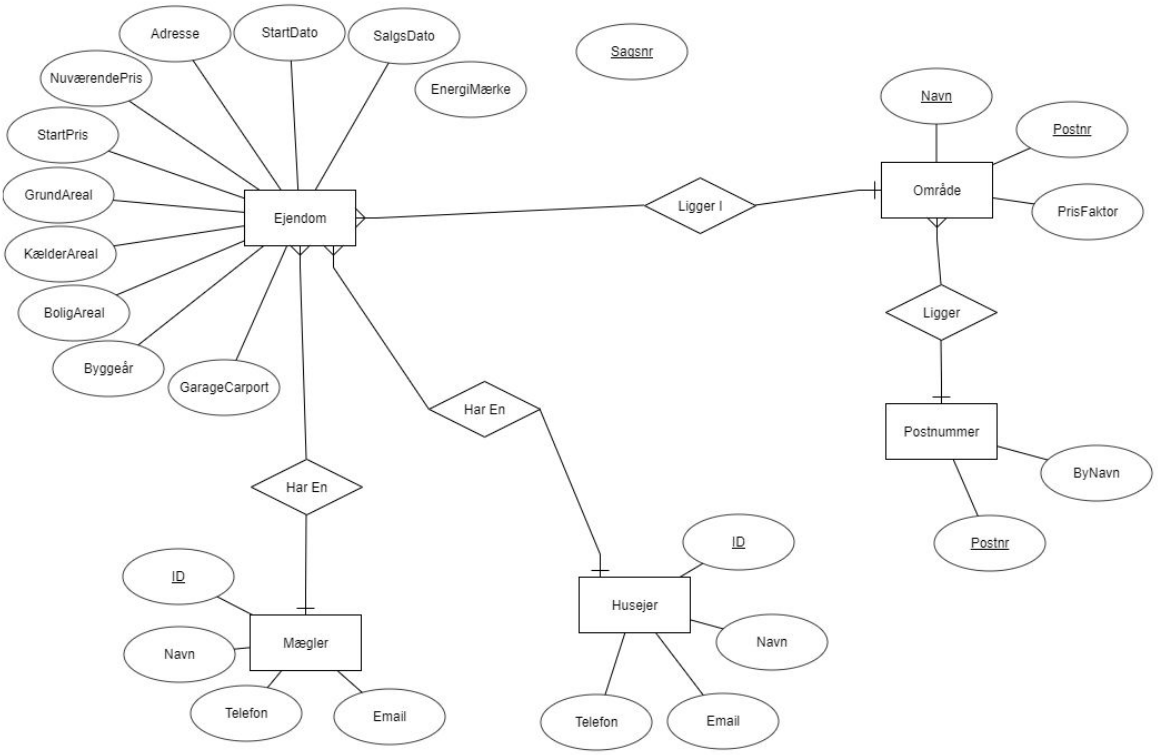
Supportability

- Der er ikke stillet krav om at kunne udvide systemet, dog bør systemets database tage højde for at der kunne være flere brugere som tilgår dataen på samme tid, derfor bør der anvendes en form for transaktionsstyring.

Database

E/R Diagram & Mapning (Jonas, Alex, Patrick og Daniel)

ER Diagram (Jonas)



Mapning (Daniel, Alex, Jonas og Patrick)

Ejendom

Sagsnr	Mægler ID	Husejer ID	Område Navn	Postnr	Energi Mærke	Start Dato	Salgs Dato	Adresse	Start Pris	Nuværende Pris	Grund Areal	Kælder Areal	Bolig Areal	Byggeår	Garage Carport
int	int	int	varchar	int	varchar	varchar	varchar	varchar	int	int	int	int	int	int	bool

Husejer

ID	Navn	Telefon	Email
int	varchar	varchar	varchar

Mægler

ID	Navn	Telefon	Email
int	varchar	varchar	varchar

Område

Navn	Postnr	PrisFaktor
varchar	int	float

Postnummer

Postnr	ByNavn
int	varchar

Normalisering (Patrick, Alex, Jonas og Daniel,)

1 Normalform

Databasen er på første Normalform:

- Vores værdier bliver ikke gentaget.
- Vi har atomare værdier.

2 Normalform

Databasen er på første Normalform:

- Databasen opfylder alle krav for Første Normalform.
- Vi har en sammensat primærnøgle - Alle ikke-primærnøgle felter har fuldt funktionelt afhængighed af primærnøglen.

3 Normalform

Databasen er på Tredje Normalform:

- Databasen opfylder alle krav for Anden Normalform.
- Ingen felter uden for primærnøglen er afhængig af hinanden.

Transaktions styring

Vi bruger transaktionsstyring tre steder, hvor databasen opdateres. Det er i OpdaterHusejer(), OpdaterMægler() og OpdaterEjendom(). Det ser således ud som en query fra OpdaterEjendom() med tilfældige VALUES for eksemplets skyld:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN TRANSACTION;
UPDATE Ejendom SET  MæglerID = 2, Postnr = 7100, StartDato = 2018-06-07 WHERE Sagsnr = 101501
COMMIT TRANSACTION;
```

Årsagen til vi anvender transaktionsstyring på vores updates, er, at vi vil undgå de sjældne scenarier, hvor flere brugere forsøger at opdatere på den samme række, og der blive indsat værdier fra begge brugeres query, og noget fra den ene overskriver noget fra den anden, og der derfor ikke er konsistent data i rækken.
Ved at have transanktionerne som Serializable undgår vi at flere brugere kan opdatere på den samme række, på samme tid.

SQL (Daniel, Alex, Jonas og Patrick)

CRUD (Daniel)

```
SELECT * FROM Ejendom;
```

Ovenstående query henter alle data fra ejendomstabellen. Det resulterer i nedenstående billede, som er et udsnit af tabellen.

	Sagsnr	MæglerID	HusejerID	OmrådeNavn	Postnr	EnergiMærke	StartDato	SalgsDato	Adresse	StartPris	NuværendePris	GrundAreal	KælderAreal	BoligAreal	Byggeår	GarageCarport
1	100001	5	43	Assendrup	7120	F	2018-05-30		Ringdams Kobbel 177	1495000	1345000	576	0	142	1994	1
2	100002	1	38	Assendrup	7120	C	2016-08-19	2018-05-16	Hornemansvej 58	1195000	1145000	545	0	111	2008	1
3	100003	9	24	Vinding	7100	D	2016-03-27	2018-03-04	Skovvænget 141	5095000	5095000	1096	0	274	1905	1
4	100004	1	44	Bredal	7120	C	2015-06-22	2017-08-05	Orionvej 0	1995000	1895000	560	46	137	1935	0
5	100005	6	5	Vinding	7100	E	2016-09-06	2018-01-14	H C Lumbyes Vej 62	3595000	3545000	606	0	195	1969	1
6	100006	4	41	Mølholm	7100	F	2016-03-15		Sdr. Tværkaj 146	2295000	2295000	1157	0	158	1985	0
7	100007	9	2	Grejsdal	7100	A10	2017-06-29		Islandsvej 8	3195000	3195000	503	21	216	2018	1
8	100008	2	6	Vinding	7100	B	2017-01-27	2017-12-13	Grejsvej 85	3495000	3495000	590	18	190	1910	1
9	100009	6	17	Vestbyen	7100	D	2018-05-12		Ulvekæden 147	4595000	4595000	636	19	248	1943	0
10	100010	5	3	Bredballe	7120	E	2018-04-20		Hostrupsvej 30	4695000	4545000	1186	35	211	1959	0
11	100011	1	48	Grejsdal	7100	C	2016-05-09		Aakjærsvej 111	2595000	2595000	1155	0	179	1973	1

```
INSERT INTO Ejendom (MæglerID, HusejerID, Områdenavn, GrundAreal, Byggeår, GarageCarport) VALUES (6,
11, 'Bredal', 144, 1989, 1);
SELECT * FROM Ejendom WHERE Sagsnr = 101502;
```

Ovenstående query opretter en Ejendom med attributterne som ses i eksemplet, og ejendommen ser ud som på nedenstående billede.

	Sagsnr	MæglerID	HusejerID	OmrådeNavn	Postnr	EnergiMærke	StartDato	SalgsDato	Adresse	StartPris	NuværendePris	GrundAreal	KælderAreal	BoligAreal	Byggeår	GarageCarport
1	101502	6	11	Bredal	NULL	NULL	NULL	NULL	NULL	NULL	NULL	144	NULL	NULL	1989	1

Ovenstående ejendom der blev oprettet opdateres via følgende query:

```
UPDATE Ejendom SET MæglerID = 4, EnergiMærke = 'B', StartPris = 1395000;
Select * from Ejendom where Sagsnr = 101502;
```

Det ser således ud:

	Sagsnr	MæglerID	HusejerID	OmrådeNavn	Postnr	EnergiMærke	StartDato	SalgsDato	Adresse	StartPris	NuværendePris	GrundAreal	KælderAreal	BoligAreal	Byggeår	GarageCarport
1	101502	4	11	Bredal	NULL	B	NULL	NULL	NULL	1395000	NULL	144	NULL	NULL	1989	1

```
DELETE FROM Ejendom WHERE Sagsnr = 101502;
```

Dette query sletter ejendommen med Sagsnr 101502.

Salgsstatistik (Alex)

```
select Sagsnr, Adresse, OmrådeNavn, Ejendom.Postnr, Postnummer.ByNavn, NuværendePris, SalgsDato,
Mægler.Navn
from Ejendom, Postnummer, Mægler
where SalgsDato != '' AND Ejendom.Postnr = Postnummer.Postnr AND Ejendom.MæglerID = Mægler.ID and
SalgsDato between '2018-01-01' and '2018-01-30';
```

Ovenstående query giver informationer omkring solgte boligere i perioden 1. Jan. 2018 til 30. Jan. 2018.

	Sagsnr	Adresse	OmrådeNavn	Postnr	ByNavn	NuværendePris	SalgsDato	Navn
1	100005	H C Lumbyes Vej 62	Vinding	7100	Vejle	3545000	2018-01-14	Poul
2	100055	Nørrebjerget 119	Vinding	7100	Vejle	2295000	2018-01-21	Oluf
3	100076	Mølkærparken 19	Bredballe	7120	Vejle Øst	3645000	2018-01-23	Poul
4	100095	Ørstedsgade 17	Bredal	7120	Vejle Øst	2845000	2018-01-06	Karla
5	100125	Gl Viborgvej 195	Bredballe	7120	Vejle Øst	3045000	2018-01-30	Lise
6	100271	Rådhuspassagen 136	Bredal	7120	Vejle Øst	2395000	2018-01-05	Karla
7	100314	Kærvej 49	Vinding	7100	Vejle	4295000	2018-01-03	Niels
8	100365	Stampesvej 106	Vinding	7100	Vejle	3145000	2018-01-10	Leif
9	100476	Nøddevænget 144	Bredballe	7120	Vejle Øst	4145000	2018-01-03	Lise
10	100514	Knudsgade 66	Vinding	7100	Vejle	3645000	2018-01-06	Lotte
11	100531	Adonisvej 96	Assendrup	7120	Vejle Øst	3195000	2018-01-13	Lise

Kvm. Pris (Alex)

```
select Sagsnr, Adresse, OmrådeNavn, Ejendom.Postnr, Postnummer.ByNavn, NuværendePris, BoligAreal,
(NuværendePris/BoligAreal) as 'KvmPris'
from Ejendom, Postnummer, Mægler
where SalgsDato != '' AND Ejendom.Postnr = Postnummer.Postnr AND Ejendom.MæglerID = Mægler.ID and
SalgsDato between '2018-01-01' and '2018-01-30';
```

Ovenstående query giver informationer omkring kvm. Priser på solgte boliger i perioden 1. Jan. 2018 til 30. Jan. 2018.

	Sagsnr	Adresse	OmrådeNavn	Postnr	ByNavn	NuværendePris	BoligAreal	KvmPris
1	100005	H C Lumbyes Vej 62	Vinding	7100	Vejle	3545000	195	18179
2	100055	Nørrebjerget 119	Vinding	7100	Vejle	2295000	132	17386
3	100076	Mølkærparken 19	Bredballe	7120	Vejle Øst	3645000	169	21568
4	100095	Ørstedsgade 17	Bredal	7120	Vejle Øst	2845000	195	14589
5	100125	Gl Viborgvej 195	Bredballe	7120	Vejle Øst	3045000	138	22065
6	100271	Rådhuspassagen 136	Bredal	7120	Vejle Øst	2395000	165	14515
7	100314	Kærvej 49	Vinding	7100	Vejle	4295000	234	18354
8	100365	Stampesvej 106	Vinding	7100	Vejle	3145000	171	18391
9	100476	Nøddevænget 144	Bredballe	7120	Vejle Øst	4145000	188	22047
10	100514	Knudsgade 66	Vinding	7100	Vejle	3645000	199	18316
11	100531	Adonisvej 96	Assendrup	7120	Vejle Øst	3195000	299	10685

Åbent Hus (Jonas)

I Åbenthus delen bruges der SQL til at fylde de to datagrids, nedenfor vises delen for ejendomme, tilsvarende findes også for mæglere.

```
SELECT Sagsnr, Adresse, OmrådeNavn, Postnummer.ByNavn, NuværendePris FROM Ejendom, Postnummer
WHERE Salgsdato = '' AND Ejendom.Postnr = Postnummer.Postnr;
```

Denne query giver en liste af ejendomme hvor salgsdato er tom, det vil sige ejendomme som ikke er solgt endnu. Dette bliver kombineret med postnummer tabellen, da bynavnet også ønskes vist, dette gøres ved kun at vise resultater hvor postnummeret i begge tabeller er ens. Et udklip af resultatet kan ses nedenfor. (Hele resultatet vises ikke da der er 818 linjer)

	Sagsnr	Adresse	OmrådeNavn	ByNavn	NuværendePris
1	100001	Ringdams Kobbel 177	Assendrup	Vejle Øst	1345000
2	100006	Sdr. Tværkaj 146	Mølholm	Vejle	2295000
3	100007	Islandsvej 8	Grejsdal	Vejle	3195000
4	100009	Ulvekæden 147	Vestbyen	Vejle	4595000
5	100010	Hostrupsvej 30	Bredballe	Vejle Øst	4545000
6	100011	Aakjærsvej 111	Grejsdal	Vejle	2595000
7	100017	Ulvekæden 138	Engum	Vejle Øst	2895000
8	100019	Klostergade 77	Grejsdal	Vejle	4295000
9	100020	Tirsbæk Strandvej 160	Assendrup	Vejle Øst	2795000
10	100022	Eskholtvænget 103	Assendrup	Vejle Øst	3095000
11	100024	Rønsled 41	Bredballe	Vejle Øst	2395000

Pris Beregner (Patrick)

```
SELECT Postnr FROM Område;
```

SQL koden kommer fra metoden ComboBoxOpretPostnr() i klassen ControllerPrisBeregner. Den finder alle postnumre fra Område tabellen.

Results Messages

	postnr
1	7120
2	7100
3	7100
4	7100
5	7100
6	7120
7	7120
8	7120

```
SELECT Navn FROM Område WHERE Postnr = 7100;
```

Koden bliver brugt i metoden ComboBoxOpretNavn() i klassen ControllerPrisBeregner. Den finder alle navne som har Postnr værdi.

Results Messages

	Navn
1	Grejsdal
2	Mølholm
3	Vestbyen
4	Vinding

```
SELECT PrisFaktor FROM Område WHERE Navn = 'Vinding' AND Postnr = 7100;
```

SQL koden kommer fra FindPrisFaktor i klassen ControllerPrisBeregner. Koden finder områdets (Navn og Postnumer) pris faktor.

Results Messages

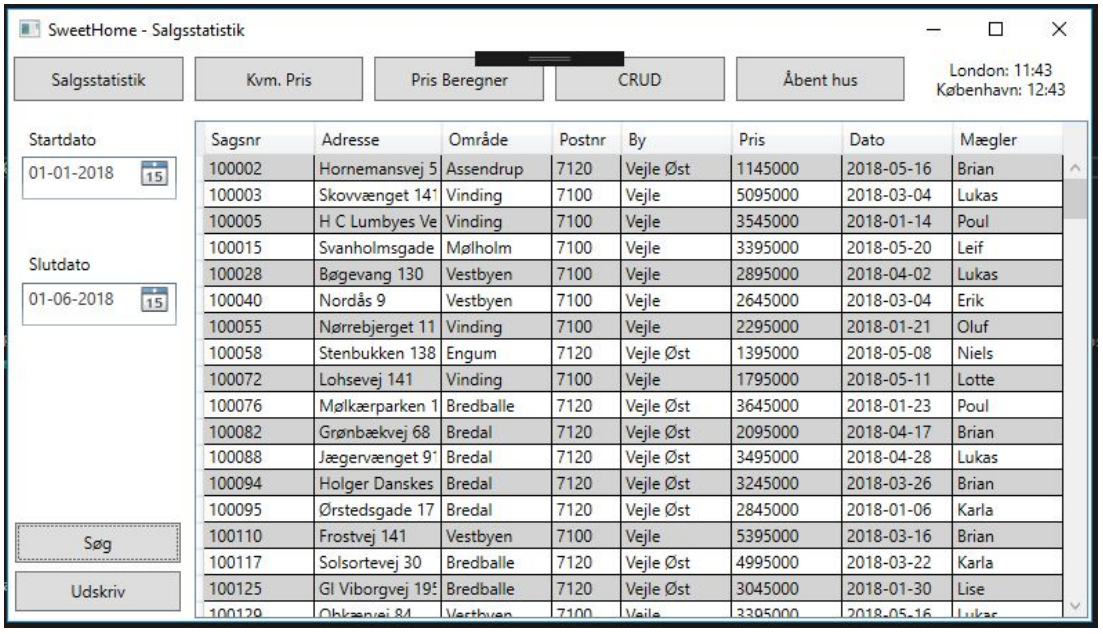
	PrisFaktor
1	1.25

Programmering / Teknik

Salgs Statistik (Jonas og Alex)

```
public static void Vis(DataGrid dg, string startDato, string slutDato)
{
    string sSQL = "select Sagsnr, Adresse, OmrådeNavn, Ejendom.Postnr, Postnummer.ByNavn,
    NuværendePris, SalgsDato, Mægler.Navn from Ejendom, Postnummer, Mægler where SalgsDato
    != '' AND Ejendom.Postnr = Postnummer.Postnr AND Ejendom.MæglerID = Mægler.ID and
    SalgsDato between '" + startDato + "' and '" + slutDato + "';";
    SqlCommand command = new SqlCommand(sSQL, ControllerConnection.conn);
    SqlDataAdapter sda = new SqlDataAdapter(command);
    DataTable dt = new DataTable("Salgsstatistik");
    sda.Fill(dt);
    dg.ItemsSource = dt.DefaultView;
}
```

Koden over er til visning af datagridet som bruger en given SQL select-statement, til at vise de attributter vi skal bruge. I SQL select-statement er der implementeret en start -og slutdato, som bruges til at indikere tidsrummet for solgte boliger vi skal bruge. Metoden bliver kaldt ved at trykke på “Søg”.




```
public static void Udskriv(DataGrid dg, string startDato, string slutDato, string udfil)
    {
        StreamWriter stream = null;

        try
        {
            string sSQL = "select Sagsnr, Adresse, OmrådeNavn, Ejendom.Postnr,
            Postnummer.ByNavn, NuværendePris, SalgsDato, Mægler.Navn from Ejendom, Postnummer,
            Mægler where SalgsDato != '' AND Ejendom.Postnr = Postnummer.Postnr AND
            Ejendom.MæglerID = Mægler.ID and SalgsDato between '" + startDato + "' and '" +
            slutDato + "';";
```

```
            SqlCommand command = new SqlCommand(sSQL, ControllerConnection.conn);
            SqlDataAdapter sda = new SqlDataAdapter(command);
            DataTable dt = new DataTable("Salgsstatistik");
            sda.Fill(dt);

            stream = new StreamWriter(udfil);

            stream.WriteLine("  Sagsnr  |        Adresse       |        Område        | Postnr
            |          By          |     Pris     |  Salgsdato");

            foreach (DataRow row in dt.Rows)
            {
                object[] array = row.ItemArray;

                stream.Write(string.Format("  {0,-7} | ", array[0].ToString()));
                stream.Write(string.Format("{0,-20} | ", array[1].ToString()));
                stream.Write(string.Format("{0,-20} | ", array[2].ToString()));
                stream.Write(string.Format("  {0,-6} | ", array[3].ToString()));
                stream.Write(string.Format("{0,-20} | ", array[4].ToString()));
                stream.Write(string.Format("{0,-12} | ", array[5].ToString()));
                stream.WriteLine(string.Format("{0,-11}", array[6].ToString()));
            }

        }
        catch (System.Exception)
        {

        }
        finally
        {
            if(stream != null)
            {
                stream.Close();
            }
        }
    }
```

Koden over er til udskrivning af de rows som er valgt ved forrige metode. Metoden bruger en SQL statement til at vælge hvad vi skal have udskrevet, som er bestemt af en dato som vi har indtastet. Vi smider så datagridet ind i et foreach loop, som så skriver vores rows ud identisk til datagridet. Metoden kaldes ved tryk på “Udskriv”.

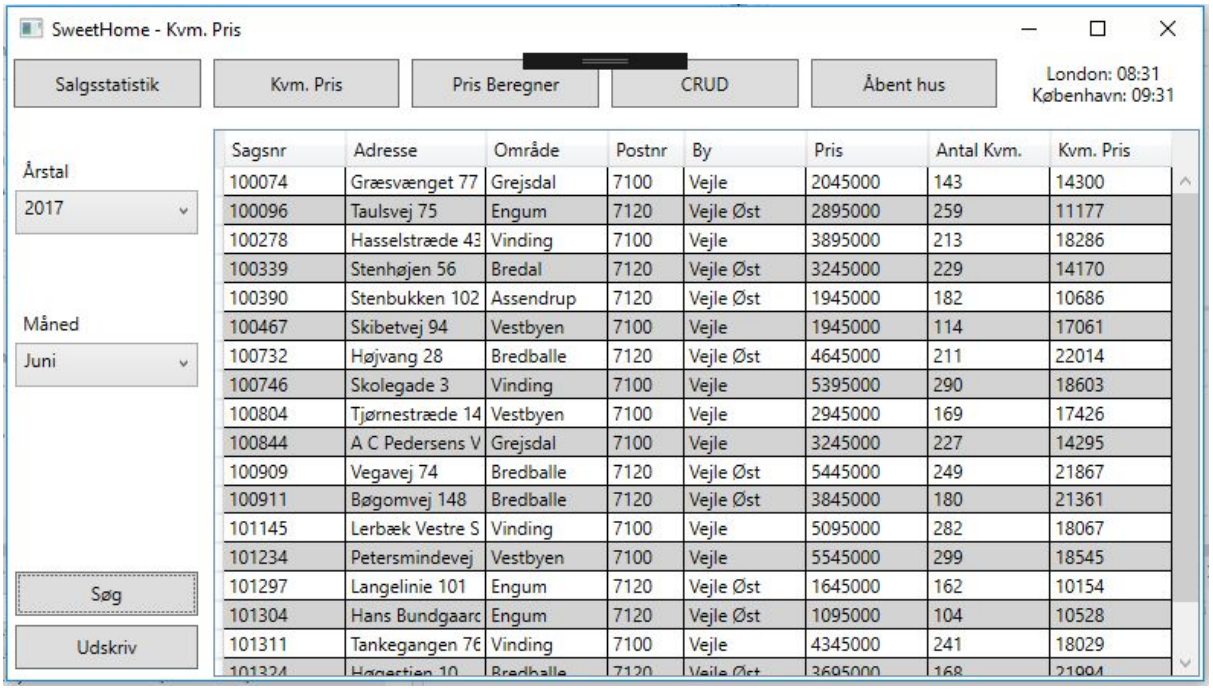
Sagsnr	Adresse	Område	Postnr	By	Pris	Salgsdato
100002	Hornemansvej 58	Assendrup	7120	Vejle Øst	1145000	2018-05-16
100003	Skovvænget 141	Vinding	7100	Vejle	5095000	2018-03-04
100005	H C Lumbyes Vej 62	Vinding	7100	Vejle	3545000	2018-01-14
100015	Svanholmsgade 192	Mølholm	7100	Vejle	3395000	2018-05-20
100028	Bøgevang 130	Vestbyen	7100	Vejle	2895000	2018-04-02
100040	Nordås 9	Vestbyen	7100	Vejle	2645000	2018-03-04
100055	Nørrebjerget 119	Vinding	7100	Vejle	2295000	2018-01-21
100058	Stenbukken 138	Engum	7120	Vejle Øst	1395000	2018-05-08
100072	Lohsevej 141	Vinding	7100	Vejle	1795000	2018-05-11
100076	Mølkærparken 19	Bredballe	7120	Vejle Øst	3645000	2018-01-23
100082	Grønbækvej 68	Bredal	7120	Vejle Øst	2095000	2018-04-17
100088	Jægervænget 91	Bredal	7120	Vejle Øst	3495000	2018-04-28
100094	Holger Danskes Vej 6	Bredal	7120	Vejle Øst	3245000	2018-03-26
100095	Ørstedsgade 17	Bredal	7120	Vejle Øst	2845000	2018-01-06
100110	Frostvej 141	Vestbyen	7100	Vejle	5395000	2018-03-16
100117	Solsortevej 30	Bredballe	7120	Vejle Øst	4995000	2018-03-22
100125	Gl Viborgvej 195	Bredballe	7120	Vejle Øst	3045000	2018-01-30
100129	Obkærvej 84	Vestbyen	7100	Vejle	3395000	2018-05-16
100145	Margrethevej 33	Bredal	7120	Vejle Øst	1545000	2018-04-03
100146	Fidalvej 114	Bredal	7120	Vejle Øst	4145000	2018-02-14
100152	Hjulmagervej 174	Grejsdal	7100	Vejle	2695000	2018-05-04
100153	Fousbjergvej 32	Vestbyen	7100	Vejle	4695000	2018-02-20
100160	Mirabelstræde 143	Assendrup	7120	Vejle Øst	3145000	2018-02-10
100163	Skanderborgvej 162	Grejsdal	7100	Vejle	4345000	2018-02-18
100179	Ferrarivej 137	Assendrup	7120	Vejle Øst	1395000	2018-05-15
100190	Stubdrup Kirkevej 12	Grejsdal	7100	Vejle	3345000	2018-02-07
100195	Teglværksvej 108	Assendrup	7120	Vejle Øst	1645000	2018-05-13
100202	Platan Alle 194	Assendrup	7120	Vejle Øst	1195000	2018-04-14

Kvm. Pris Statistik (Jonas og Alex)

public static void Vis(DataGrid dg, string startDato, string slutDato)

```
        {
            string sSQL = "select Sagsnr, Adresse, OmrådeNavn,
            Ejendom.Postnr, Postnummer.ByNavn, NuværendePris, BoligAreal, (NuværendePris/BoligAreal)
            as 'KvmPris' from Ejendom, Postnummer, Mægler where SalgsDato != '' AND Ejendom.Postnr =
            Postnummer.Postnr AND Ejendom.MæglerID = Mægler.ID and SalgsDato between '" + startDato
            + "' and '" + slutDato + "';";
            SqlCommand command = new SqlCommand(sSQL, ControllerConnection.conn);
            SqlDataAdapter sda = new SqlDataAdapter(command);
            DataTable dt = new DataTable("Salgsstatistik");
            sda.Fill(dt);
            dg.ItemsSource = dt.DefaultView;
        }
```

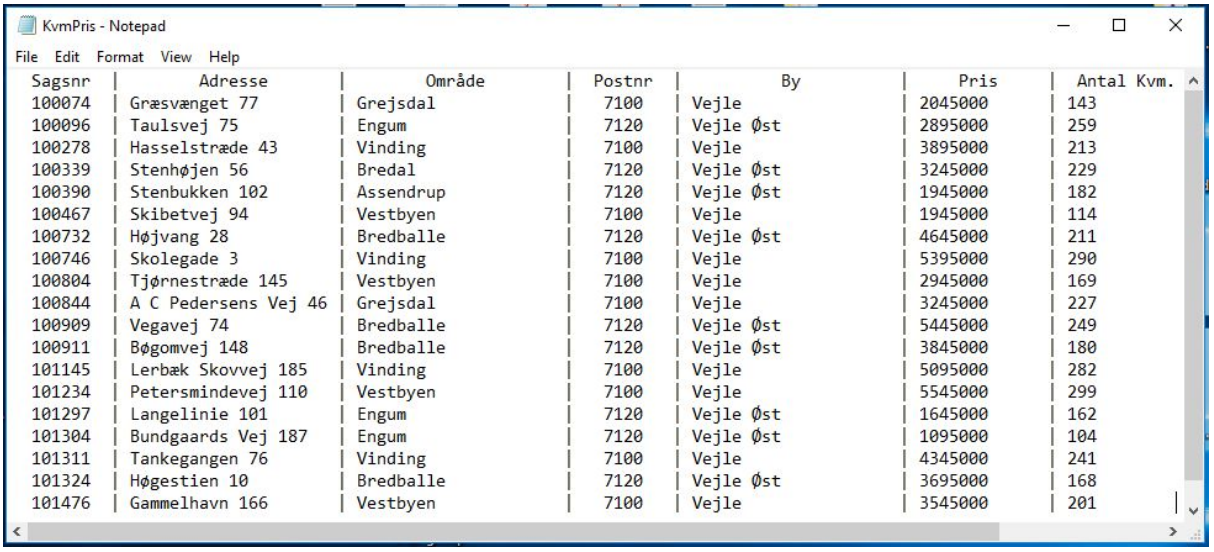
Koden over er til visning af datagridet som bruger en given SQL select-statement, til at vise de attributter vi skal bruge. I SQL select-statement er der implementeret en start -og slutdato, som bruges til at indikere tidsrummet for solgte boliger vi skal bruge. Brugeren vælger en given måned og år hvilket bliver brugt til startdato og slutdato. Metoden bliver kaldt ved at trykke på "Søg".




```
public static void Udskriv(DataGrid dg, string startDato, string slutDato, string udfil)
        {
            StreamWriter stream = null;

            try
            {
                string sSQL = "select Sagsnr, Adresse, OmrådeNavn, Ejendom.Postnr,
            Postnummer.ByNavn, NuværendePris, BoligAreal, (NuværendePris / BoligAreal) as 'KvmPris'
            from Ejendom, Postnummer, Mægler where SalgsDato != '' AND Ejendom.Postnr =
            Postnummer.Postnr AND Ejendom.MæglerID = Mægler.ID and SalgsDato between '" + startDato
            + "' and '" + slutDato + "'; ";
                SqlCommand command = new SqlCommand(sSQL, ControllerConnection.conn);
                SqlDataAdapter sda = new SqlDataAdapter(command);
```

```
            DataTable dt = new DataTable("Salgsstatistik");
            sda.Fill(dt);

            stream = new StreamWriter(udfil);

            stream.WriteLine("  Sagsnr  |        Adresse       |        Område        | Postnr
            |          By          |     Pris     |  Antal Kvm.  |  Kvm Pris.  ");

            foreach (DataRow row in dt.Rows)
            {
                object[] array = row.ItemArray;

                stream.Write(string.Format("  {0,-7} | ", array[0].ToString()));
                stream.Write(string.Format("{0,-20} | ", array[1].ToString()));
                stream.Write(string.Format("{0,-20} | ", array[2].ToString()));
                stream.Write(string.Format("  {0,-6} | ", array[3].ToString()));
                stream.Write(string.Format("{0,-20} | ", array[4].ToString()));
                stream.Write(string.Format("{0,-12} | ", array[5].ToString()));
                stream.Write(string.Format("{0,-12} | ", array[6].ToString()));
                stream.WriteLine(string.Format("{0,-11}", array[7].ToString()));
            }

        }
        catch (System.Exception)
        {

        }
        finally
        {
            if (stream != null)
            {
                stream.Close();
            }
        }
    }
```

Koden over er til udskrivning af de rows som er valgt ved forrige metode. Metoden bruger en SQL statement til at vælge hvad vi skal have udskrevet, som er bestemt af en dato som er valgt via drop-down. Vi smider så datagridet ind i et foreach loop, som så skriver vores rows ud identisk til datagridet. Metoden kaldes ved tryk på “Udskriv”.



Sagsnr	Adresse	Område	Postnr	By	Pris	Antal Kvm.
100074	Græsvænget 77	Grejsdal	7100	Vejle	2045000	143
100096	Taulsvej 75	Engum	7120	Vejle Øst	2895000	259
100278	Hasselstræde 43	Vinding	7100	Vejle	3895000	213
100339	Stenhøjen 56	Bredal	7120	Vejle Øst	3245000	229
100390	Stenbukken 102	Assendrup	7120	Vejle Øst	1945000	182
100467	Skibetvej 94	Vestbyen	7100	Vejle	1945000	114
100732	Højvang 28	Bredballe	7120	Vejle Øst	4645000	211
100746	Skolegade 3	Vinding	7100	Vejle	5395000	290
100804	Tjørnestræde 145	Vestbyen	7100	Vejle	2945000	169
100844	A C Pedersens Vej 46	Grejsdal	7100	Vejle	3245000	227
100909	Vegavej 74	Bredballe	7120	Vejle Øst	5445000	249
100911	Bøgomvej 148	Bredballe	7120	Vejle Øst	3845000	180
101145	Lerbæk Skovvej 185	Vinding	7100	Vejle	5095000	282
101234	Petersmindevej 110	Vestbyen	7100	Vejle	5545000	299
101297	Langelinie 101	Engum	7120	Vejle Øst	1645000	162
101304	Bundgaards Vej 187	Engum	7120	Vejle Øst	1095000	104
101311	Tankegangen 76	Vinding	7100	Vejle	4345000	241
101324	Høgestien 10	Bredballe	7120	Vejle Øst	3695000	168
101476	Gammelhavn 166	Vestbyen	7100	Vejle	3545000	201

CRUD-Program (Daniel)

Jeg har valgt at vise eksempler på CRUD funktioner i ejendomsviewet, men billeder af alle tre views - husejer, mægler, ejendom vil være at se under *Læs*, som kommer efter *Opret.* Grundet til valget af ejendomsviewet er, at Husejer og Mægler praktisk talt er ens i deres funktionalitet og kode, og desuden ikke er så komplekse. Ejendom har en større kompleksitet og er mere interessant at vise og se. Dog er der meget kode.

Opret

ButtonOpret_Click

```
        private void ButtonOpret_Click(object sender, RoutedEventArgs e)
        {
    if (WrapPanelHusejer.IsVisible && TextBoxHusejerNavn.Text != "Navn" &&
              TextBoxHusejerNavn.Text != "" &&
              TextBoxHusejerEmail.Text != "Email" && TextBoxHusejerEmail.Text != "" &&
              TextBoxHusejerTelefon.Text != "Telefon" &&
              TextBoxHusejerTelefon.Text != "")
    {
               ControllerCrudHusejer.OpretHusejer(TextBoxHusejerNavn.Text,
              TextBoxHusejerEmail.Text, TextBoxHusejerTelefon.Text);

               ControllerCrudHusejer.LæsHusejer(DataGridHusejer);

               TextBoxHusejerID.Text = "ID (Autogenereres)";
               TextBoxHusejerNavn.Text = "Navn";
               TextBoxHusejerEmail.Text = "Email";
               TextBoxHusejerTelefon.Text = "Telefon";
    }
    if (WrapPanelMægler.IsVisible && TextBoxMæglerNavn.Text != "Navn" &&
              TextBoxMæglerNavn.Text != "" &&
              TextBoxMæglerTelefon.Text != "Telefon" && TextBoxMæglerTelefon.Text != ""
              && TextBoxMæglerEmail.Text != "Email" &&
               TextBoxMæglerEmail.Text != "")
    {
               ControllerCrudMægler.OpretMægler(TextBoxMæglerNavn.Text,
              TextBoxMæglerTelefon.Text, TextBoxMæglerEmail.Text);

               ControllerCrudMægler.LæsMægler(DataGridMægler);

               TextBoxMæglerID.Text = "ID (Autogenereres)";
               TextBoxMæglerNavn.Text = "Navn";
               TextBoxMæglerTelefon.Text = "Telefon";
               TextBoxMæglerEmail.Text = "Email";

    }
    if (WrapPanelEjendom.IsVisible && TextBoxEjendomMæglerID.Text != "Mægler ID
              (Påkrævet)" && TextBoxEjendomMæglerID.Text != "" &&
              TextBoxEjendomHusejerID.Text != "Husejer ID (Påkrævet)" &&
              TextBoxHusejerID.Text != "")
    {
               ControllerCrudEjendom.OpretEjendom(TextBoxEjendomMæglerID.Text,
              TextBoxEjendomHusejerID.Text, TextBoxEjendomOmrådeNavn.Text,
              TextBoxEjendomPostnr.Text, TextBoxEjendomEnergiMærke.Text,
              TextBoxEjendomStartDato.Text, TextBoxEjendomSalgsDato.Text,
              TextBoxEjendomAdresse.Text, TextBoxEjendomStartPris.Text,
```

```
                TextBoxEjendomNuværendePris.Text, TextBoxEjendomGrundAreal.Text,
                TextBoxEjendomKælderAreal.Text, TextBoxEjendomBoligAreal.Text,
                TextBoxEjendomByggeår.Text, ComboBoxEjendomGarageCarport.Text);

                 ControllerCrudEjendom.LæsEjendom(DataGridEjendom);

                 TextBoxEjendomMæglerID.Text = "Mægler ID (Påkrævet)";
                 TextBoxEjendomHusejerID.Text = "Husejer ID (Påkrævet)";
                 TextBoxEjendomOmrådeNavn.Text = "Områdenavn";
                 TextBoxEjendomPostnr.Text = "Postnr";
                 TextBoxEjendomEnergiMærke.Text = "Energimærke";
                 TextBoxEjendomStartDato.Text = "Startdato (åååå-mm-dd)";
                 TextBoxEjendomSalgsDato.Text = "Salgsdato (åååå-mm-dd)";
                 TextBoxEjendomAdresse.Text = "Adresse";
                 TextBoxEjendomStartPris.Text = "Startpris";
                 TextBoxEjendomNuværendePris.Text = "Nuværende pris";
                 TextBoxEjendomGrundAreal.Text = "Grundareal";
                 TextBoxEjendomKælderAreal.Text = "Kælderareal";
                 TextBoxEjendomBoligAreal.Text = "Boligareal";
                 TextBoxEjendomByggeår.Text = "Byggeår";
                 ComboBoxEjendomGarageCarport.SelectedIndex = 2;

        }
           }
```

ButtonOpret_Click

Her er koden for 'opret' knappen i CRUD delen af vores program. Når brugeren trykker på 'opret' tjekkes der for det første hvilket CRUD view (Husejer, Mægler eller ejendom) der er aktivt, og derefter om der står tekst i tekstboksene på CRUD siden. Hvis ejendomsviewet er det aktive, og der står tekst i textBoxEjendomMæglerID.Text og textBoxEjendomHusejerID.Text, så køres metoden i den klasse der svarer til viewet - ControllerCrudEjendom, og der oprettes enejendom med et automatisk genereret ID og MæglerID og HusejerID og de andre attributter der har en brugerindtastet værdi i tekstboksene . Til sidst indlæses dataen i datagrid'et via controllerCrudEjendom.LæsEjendom() og tekstboksenes indhold bliver nulstillet til deres default værdier via TextBox.Text properties.

OpretEjendom()

```
        public static void OpretEjendom(string mæglerID, string husejerID, string
                                         områdeNavn, string postnr, string energiMærke,
                                         string startDato, String salgsDato, string adresse,
                                         string startPris, String nuværendePris, string
                                         grundAreal, string kælderAreal, string boligAreal,
                                         string byggeår, string garageCarport)
         {
         string tempSSQLOne = "INSERT INTO Mægler (";
         string tempSSQLTwo = ") VALUES (";

         tempSSQLOne += "MæglerID";
         tempSSQLTwo += mæglerID;
         tempSSQLOne += ", HusejerID";
```

```
        tempSSQLTwo += ", " + husejerID;

        if(områdeNavn != "Områdenavn" && områdeNavn != "")
        {
                tempSSQLOne += ", OmrådeNavn";
                tempSSQLTwo += ", " + områdeNavn;
        }

        if (postnr != "Postnr" && postnr != "")
        {
                tempSSQLOne += ", Postnr";
                tempSSQLTwo += ", " + postnr;
        }

        if (energiMærke != "Energimærke" && energiMærke != "")
        {
                tempSSQLOne += ", EnergiMærke";
                tempSSQLTwo += ", " + energiMærke;
        }

        if (startDato != "Startdato" && startDato != "")
        {
                tempSSQLOne += ", StartDato";
                tempSSQLTwo += ", " + startDato;
        }

        if (salgsDato != "Salgsdato" && salgsDato != "")
        {
                tempSSQLOne += ", SalgsDato";
                tempSSQLTwo += ", " + salgsDato;
        }

        if (adresse != "Adresse" && adresse != "")
        {
                tempSSQLOne += ", Adresse";
                tempSSQLTwo += ", " + adresse;
        }

        if (startPris != "Startpris" && startPris != "")
        {
                tempSSQLOne += ", StartPris";
                tempSSQLTwo += ", " + startPris;
        }

        if (nuværendePris != "Nuværende pris" && nuværendePris != "")
        {
                tempSSQLOne += ", NuværendePris";
                tempSSQLTwo += ", " + nuværendePris;
        }

        if (grundAreal != "Grundareal" && grundAreal != "")
        {
                tempSSQLOne += ", GrundAreal";
                tempSSQLTwo += ", " + grundAreal;
        }

        if (kælderAreal != "Kælderareal" && kælderAreal != "")
        {
                tempSSQLOne += ", KælderAreal";
                tempSSQLTwo += ", " + kælderAreal;
        }
```

```
        if (boligAreal != "Boligareal" && boligAreal != "")
        {
                tempSSQLOne += ", BoligAreal";
                tempSSQLTwo += ", " + boligAreal;
        }

        if (byggeår != "Byggeår" && byggeår != "")
        {
                tempSSQLOne += ", Byggeår";
                tempSSQLTwo += ", " + byggeår;
        }

        if (garageCarport != "Garage/carport" && garageCarport != "")
        {
                tempSSQLOne += ", GarageCarport";
                tempSSQLTwo += ", " + garageCarport;
        }

        string sSQL = tempSSQLOne + tempSSQLTwo + ");";

        SqlCommand command = new SqlCommand(sSQL, ControllerConnection.conn);
        command.ExecuteNonQuery();
        }
```

```
INSERT INTO Mægler (MæglerID, HusejerID, GarageCarport) VALUES(mæglerID, husejerID, garageCarport);
```

OpretEjendom()

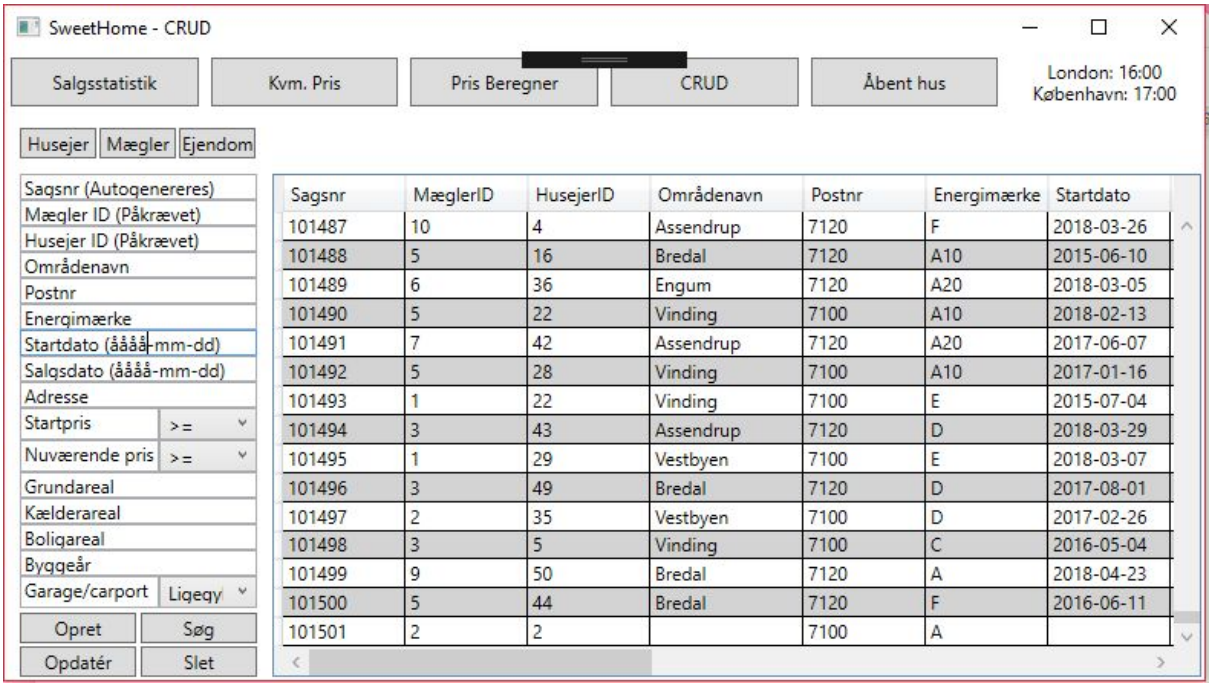
Her ses OpretEjendom() som bliver kaldt når Ejendoms panelet er valgt. Metoden har to midlertidige strings; tempSSQLOne og tempSSQLTwo. TempSSQLOne indeholder den del af SQL stringen der tilføjer kolonnerne og tempSSQLTwo indeholder den del af SQL strengen der tilføjer værdierne i kolonnerne.

OpretEjendom() får tekstboksene på ejendomspanelets indhold som inputparametre og tjekker via if-statements for ændret tekstindhold i tekstboksene. Der hvor der er ændret indhold indsættes der SQL strings i tempSSQLOne og tempSSQLTwo via += og til sidst sættes tempSSQLOne og tempSSQLTwo sammen, og afsluttes med *");",* hvorefter queryen bliver kørt.

I tilfældet der ikke er data i de påkrævede tekstbokse sker der ikke noget.

Som nævnt, ved forrige eksempel med 'opret' knappen, er det efter OpretEjendom() metoden har kørt datagrid'et vil blive opdateret via crudControllerEjendom.LæsEjendom() og tekstboksenes indhold bliver sat til default værdierne.

Oprettet ejendom



Oprettet ejendom

Her ses en ejendom oprettet via sql queryen: INSERT INTO Ejendom (MæglerID, HusejerID, Postnr, EnergiMærke) VALUES (2, 2, 7100, A);

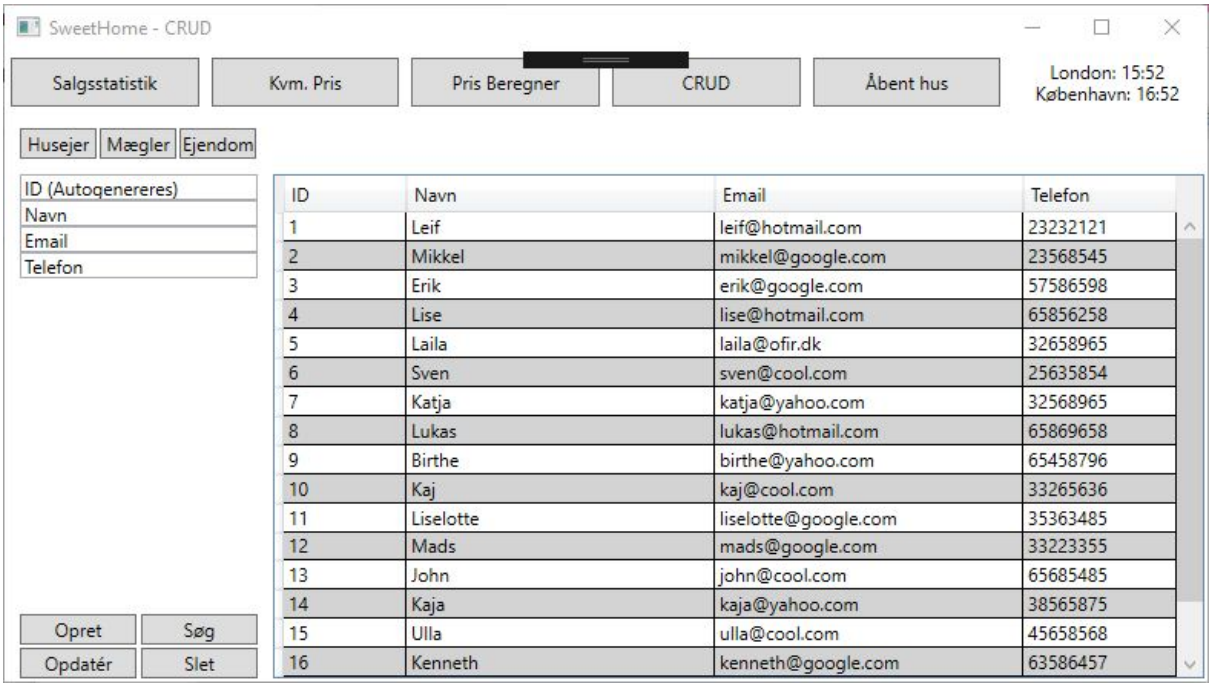
VALUES er fra input i tekstboksene. C# koden der kører queuryen er udeladt, da jeg kun ville vise SQL stringen og det er at finde i ovenstående kodeeksempler.

Læs

De følgende tre billeder vil være det view der ses, når man henholdsvis klikker på Husejer, Mægler og Ejendom. Når du vælger CRUD ved de øverste "menu" knapper, bliver Husejer som standard vist. Dette sker via LæsHusejer(), LæsMægler() og LæsEjendom() metoderne.

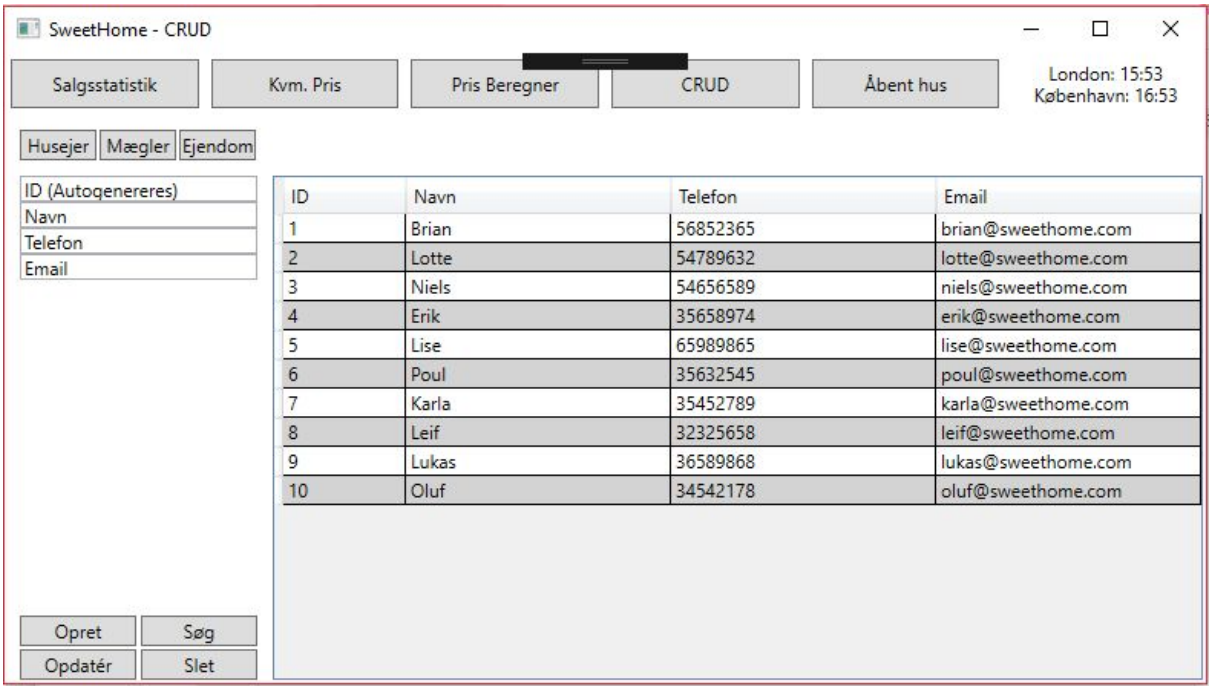
Jeg vil kort forklare hvad jeg mener med et view. Et view i CRUD funktionen består af et WrapPanel de ridneholder teksboskene ude til venstre. Derudover består det af datagrid'et der fylder resten af billedet. *Husejer, Mægler, Ejendom, Opret, Søg, Opdatér,* og *Slet* er generelle for, og synlige ved både husejer, mægler og ejendomsviewet.

CrudHusejer



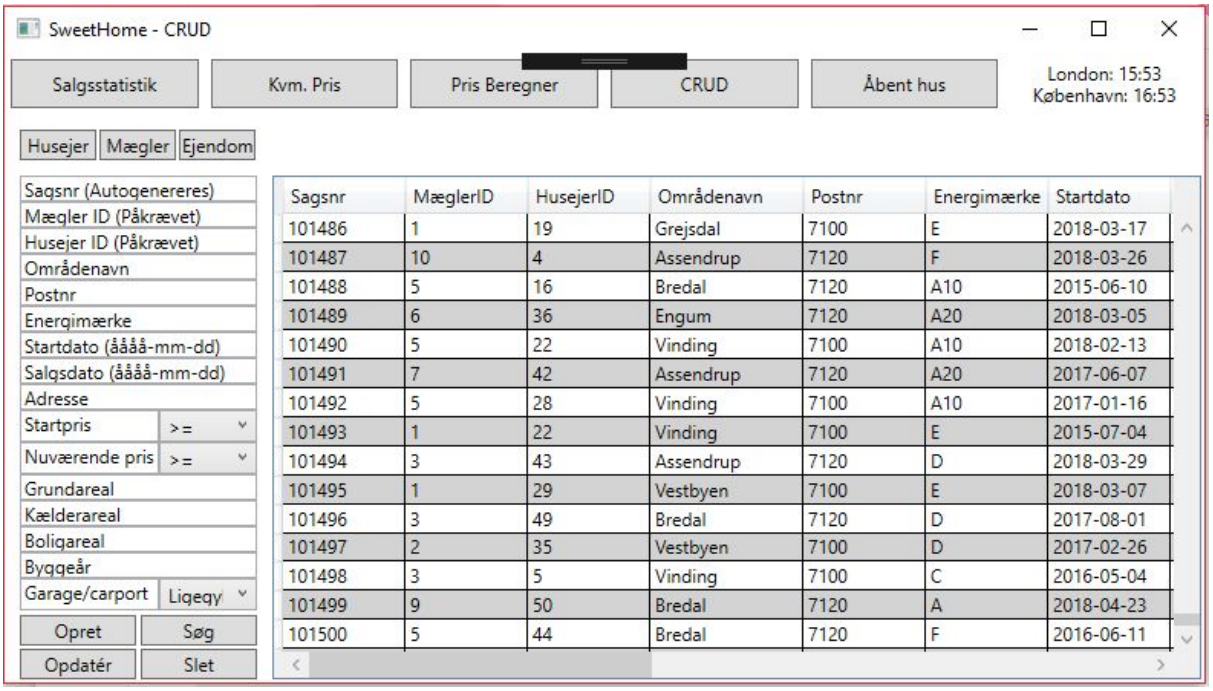
ID	Navn	Email	Telefon
1	Leif	leif@hotmail.com	23232121
2	Mikkel	mikkel@google.com	23568545
3	Erik	erik@google.com	57586598
4	Lise	lise@hotmail.com	65856258
5	Laila	laila@ofir.dk	32658965
6	Sven	sven@cool.com	25635854
7	Katja	katja@yahoo.com	32568965
8	Lukas	lukas@hotmail.com	65869658
9	Birthe	birthe@yahoo.com	65458796
10	Kaj	kaj@cool.com	33265636
11	Liselotte	liselotte@google.com	35363485
12	Mads	mads@google.com	33223355
13	John	john@cool.com	65685485
14	Kaja	kaja@yahoo.com	38565875
15	Ulla	ulla@cool.com	45658568
16	Kenneth	kenneth@google.com	63586457

CrudMægler



ID	Navn	Telefon	Email
1	Brian	56852365	brian@sweethome.com
2	Lotte	54789632	lotte@sweethome.com
3	Niels	54656589	niels@sweethome.com
4	Erik	35658974	erik@sweethome.com
5	Lise	65989865	lise@sweethome.com
6	Poul	35632545	poul@sweethome.com
7	Karla	35452789	karla@sweethome.com
8	Leif	32325658	leif@sweethome.com
9	Lukas	36589868	lukas@sweethome.com
10	Oluf	34542178	oluf@sweethome.com

CrudEjendom



ButtonEjendom_Click

```
private void ButtonEjendom_Click(object sender, RoutedEventArgs e)
        {

        WrapPanelHusejer.Visibility = Visibility.Hidden;
        DataGridHusejer.Visibility = Visibility.Hidden;
        WrapPanelMægler.Visibility = Visibility.Hidden;
        DataGridMægler.Visibility = Visibility.Hidden;
        WrapPanelEjendom.Visibility = Visibility.Visible;
        DataGridEjendom.Visibility = Visibility.Visible;
        ControllerCrudEjendom.LæsEjendom(DataGridEjendom);
        }
```

ButtonEjendom_Click

Når der klikkes på ButtonEjendom (betegnet som bare Ejendom ovenfor), vil de relevante controls blive sat til *visible* og de ikke-relevante (de controls der danner de to andre views) sat som *hidden*. LæsEjendom() bliver kaldt og fylder datagrid'et med data.

LæsEjendom()

```
public static void LæsEjendom(DataGrid dg)
        {
        string sSQL = "SELECT * FROM Ejendom";
        SqlCommand command = new SqlCommand(sSQL, ControllerConnection.conn);
        SqlDataAdapter sda = new SqlDataAdapter(command);
        DataTable dt = new DataTable("Ejendom");
        sda.Fill(dt);
        dg.ItemsSource = dt.DefaultView;
        }
```

LæsEjendom() læser meget simpelt al dataen fra en database tabel over i datagridet i viewet. Det man ser, er CrudEjendom billedet ovenfor.

Søg

Søgefunktionaliteten relaterer sig til læsefunktionaliteteten, derfor nævner jeg den lidt i forlængelse af *Læs*.

```
ButtonSøg_Click
private void ButtonSøg_Click(object sender, RoutedEventArgs e)
        {
        if (WrapPanelHusejer.IsVisible && (TextBoxHusejerID.Text != "ID
                  (Autogenereres)" && TextBoxHusejerID.Text != "") ||
                  (TextBoxHusejerNavn.Text != "Navn" && TextBoxHusejerNavn.Text != "") ||
                  (TextBoxHusejerEmail.Text != "Email" && TextBoxHusejerEmail.Text != "")
                  || (TextBoxHusejerTelefon.Text != "Telefon" && TextBoxHusejerTelefon.Text
                  != ""))
        {
                ControllerCrudHusejer.SøgHusejer(TextBoxHusejerID.Text,
                  TextBoxHusejerNavn.Text, TextBoxHusejerEmail.Text,
                  TextBoxHusejerTelefon.Text, DataGridHusejer);

                  TextBoxHusejerID.Text = "ID (Autogenereres)";
                  TextBoxHusejerNavn.Text = "Navn";
                  TextBoxHusejerEmail.Text = "Email";
                  TextBoxHusejerTelefon.Text = "Telefon";
        }
        if (WrapPanelMægler.IsVisible && (TextBoxMæglerID.Text != "ID
                  (Autogenereres)" && TextBoxMæglerID.Text != "") ||
                  (TextBoxMæglerNavn.Text != "Navn" && TextBoxMæglerNavn.Text != "") ||
                  (TextBoxMæglerTelefon.Text != "Telefon" && TextBoxMæglerTelefon.Text !=
                  "") || (TextBoxMæglerEmail.Text != "Email" && TextBoxMæglerEmail.Text !=
                  ""))
        {
                ControllerCrudMægler.SøgMægler(TextBoxMæglerID.Text, TextBoxMæglerNavn.Text,
                  TextBoxMæglerTelefon.Text, TextBoxMæglerEmail.Text, DataGridMægler);

                  TextBoxMæglerID.Text = "ID (Autogenereres)";
                  TextBoxMæglerNavn.Text = "Navn";
                  TextBoxMæglerTelefon.Text = "Telefon";
                  TextBoxMæglerEmail.Text = "Email";
        }
        if (WrapPanelEjendom.IsVisible && (TextBoxEjendomSagsnr.Text != "Sagsnr
                  (Autogenereres)" && TextBoxEjendomSagsnr.Text != "") ||
                  (TextBoxEjendomMæglerID.Text != "Mægler ID (Påkrævet)" &&
                  TextBoxEjendomMæglerID.Text != "") ||
                  (TextBoxEjendomHusejerID.Text != "Husejer ID (Påkrævet)" &&
                  TextBoxEjendomHusejerID.Text != "") || (TextBoxEjendomOmrådeNavn.Text !=
                  "Områdenavn" && TextBoxEjendomOmrådeNavn.Text != "") ||
                  (TextBoxEjendomPostnr.Text != "Postnr" && TextBoxEjendomPostnr.Text !=
                  "") || (TextBoxEjendomEnergiMærke.Text != "Energimærke" &&
                  TextBoxEjendomEnergiMærke.Text != "") ||
                  (TextBoxEjendomStartDato.Text != "Startdato (åååå-mm-dd)" &&
                  TextBoxEjendomStartDato.Text != "") || (TextBoxEjendomSalgsDato.Text !=
                  "Salgsdato (åååå-mm-dd)" && TextBoxEjendomSalgsDato.Text != "") ||
                  (TextBoxEjendomAdresse.Text != "Adresse" && TextBoxEjendomAdresse.Text !=
                  "") || (TextBoxEjendomStartPris.Text != "Startpris" &&
                  TextBoxEjendomStartPris.Text != "") ||
```

```
                (TextBoxEjendomNuværendePris.Text != "Nuværende pris" &&
                TextBoxEjendomNuværendePris.Text != "") || (TextBoxEjendomGrundAreal.Text
                != "Grundareal" && TextBoxEjendomGrundAreal.Text != "") ||
                (TextBoxEjendomKælderAreal.Text != "Kælderareal" &&
                TextBoxEjendomKælderAreal.Text != "") || (TextBoxEjendomBoligAreal.Text
                != "Boligareal" && TextBoxEjendomBoligAreal.Text != "") ||
                (TextBoxEjendomByggeår.Text != "Byggeår" && TextBoxEjendomByggeår.Text !=
                "") || ComboBoxEjendomGarageCarport.SelectedIndex == 0 ||
                ComboBoxEjendomGarageCarport.SelectedIndex == 1 ||
                ComboBoxEjendomGarageCarport.SelectedIndex == 2)
        {
                ControllerCrudEjendom.SøgEjendom(TextBoxEjendomSagsnr.Text,
                TextBoxEjendomMæglerID.Text, TextBoxEjendomHusejerID.Text,
                TextBoxEjendomOmrådeNavn.Text, TextBoxEjendomPostnr.Text,
                TextBoxEjendomEnergiMærke.Text, TextBoxEjendomStartDato.Text,
                TextBoxEjendomSalgsDato.Text,
                TextBoxEjendomAdresse.Text, TextBoxEjendomStartPris.Text,
                TextBoxEjendomNuværendePris.Text, TextBoxEjendomGrundAreal.Text,
                TextBoxEjendomKælderAreal.Text, TextBoxEjendomBoligAreal.Text,
                TextBoxEjendomByggeår.Text, ComboBoxEjendomGarageCarport.SelectedIndex,
                ComboBoxEjendomStartPris.SelectedIndex,
                ComboBoxEjendomNuværendePris.SelectedIndex, DataGridEjendom);

                TextBoxEjendomSagsnr.Text = "Sagsnr (Autogenereres)";
                TextBoxEjendomMæglerID.Text = "Mægler ID (Påkrævet)";
                TextBoxEjendomHusejerID.Text = "Husejer ID (Påkrævet)";
                TextBoxEjendomOmrådeNavn.Text = "Områdenavn";
                TextBoxEjendomPostnr.Text = "Postnr";
                TextBoxEjendomEnergiMærke.Text = "Energimærke";
                TextBoxEjendomStartDato.Text = "Startdato (åååå-mm-dd)";
                TextBoxEjendomSalgsDato.Text = "Salgsdato (åååå-mm-dd)";
                TextBoxEjendomAdresse.Text = "Adresse";
                TextBoxEjendomStartPris.Text = "Startpris";
                TextBoxEjendomNuværendePris.Text = "Nuværende pris";
                TextBoxEjendomGrundAreal.Text = "Grundareal";
                TextBoxEjendomKælderAreal.Text = "Kælderareal";
                TextBoxEjendomBoligAreal.Text = "Boligareal";
                TextBoxEjendomByggeår.Text = "Byggeår";
                ComboBoxEjendomGarageCarport.SelectedIndex = 2;
        }
        }
```

ButtonSøg_Click

Når der trykkes på 'søg' bliver der først tjekket hvilket WrapPanel der er synligt, og i det her tilfælde er det Ejendom. Dernæst tjekkes der i et *If-statement* for, om mindst én af tekstboksene har en ændret værdi. Hvis det er tilfældet køres SøgEjendom() og derefter sættes tekstboksenes indhold til default værdierne.

SøgEjendom()

```
public static void SøgEjendom(string sagsnr, string mæglerID, string husejerID,
                            string områdeNavn, string postnr, string energiMærke,
                            string startDato, string salgsDato, string adresse, string
                            startPris, string nuværendePris, string grundAreal,
                            string kælderAreal, string boligAreal, string byggeår, int
                            garageCarport, int comboBoxEjendomStartPris,
                            int comboBoxEjendomNuværendePris, DataGrid dg)
            {
                            string[] CheckIfContains = {"Sagsnr", "MæglerID", "HusejerID",
```

```
                        "OmrådeNavn", "Postnr", "EnergiMærke", "StartDato",
                        "SalgsDato", "Adresse", "StartPris", "NuværendePris",
                        "GrundAreal", "KælderAreal",
                        "BoligAreal","Byggeår", "GarageCarport", "StartPris",
                        "NuværendePris"};

                        string tempSSQL = "SELECT * FROM Ejendom WHERE";

        if (sagsnr != "Sagsnr (Autogenereres)" && sagsnr != "")
        {
                tempSSQL += " Sagsnr = '" + sagsnr + "'";
        }

        if (mæglerID != "Mægler ID (Påkrævet)" && mæglerID != "")
              {
                 if ((CheckIfContains.Any(tempSSQL.Contains)))
                 {
                    tempSSQL += " AND MæglerID = '" + mæglerID + "'";
                 }
                 else
                 {
                    tempSSQL += " MæglerID = '" + mæglerID + "'";
                 }
              }
              if (husejerID != "Husejer ID (Påkrævet)" && husejerID != "")
        {
                 if ((CheckIfContains.Any(tempSSQL.Contains)))
                 {
                    tempSSQL += " AND HusejerID = '" + husejerID + "'";
                 }
                 else
                 {
                    tempSSQL += " HusejerID = '" + husejerID + "'";
                 }
              }
              if (områdeNavn != "Områdenavn" && områdeNavn != "")
              {
                 if ((CheckIfContains.Any(tempSSQL.Contains)))
                 {
                    tempSSQL += " AND OmrådeNavn LIKE '" + områdeNavn + "%'";
                 }
                 else
                 {
                    tempSSQL += " OmrådeNavn LIKE '" + områdeNavn + "%'";
                 }
              }
              if (postnr != "Postnr" && postnr != "")
           {
                  if ((CheckIfContains.Any(tempSSQL.Contains)))
                  {
                    tempSSQL += " AND Postnr LIKE '" + postnr + "%'";
                  }
                  else
                  {
                    tempSSQL += " Postnr LIKE '" + postnr + "%'";
                  }
        }
              if (energiMærke != "Energimærke" && energiMærke != "")
        {
                  if ((CheckIfContains.Any(tempSSQL.Contains)))
                  {
```

```
                tempSSQL += " AND EnergiMærke LIKE '" + energiMærke + "%'";
            }
            else
            {
              tempSSQL += " EnergiMærke LIKE '" + energiMærke + "%'";
            }
    }
        if (startDato != "Startdato (åååå-mm-dd)" && startDato != "")
       {
            if ((CheckIfContains.Any(tempSSQL.Contains)))
            {
              tempSSQL += " AND StartDato LIKE '" + startDato + "%'";
            }
            else
            {
              tempSSQL += " StartDato LIKE '" + startDato + "%'";
            }
    }
        if (salgsDato != "Salgsdato (åååå-mm-dd)" && salgsDato != "")
    {
            if ((CheckIfContains.Any(tempSSQL.Contains)))
            {
              tempSSQL += " AND SalgsDato LIKE '" + salgsDato + "%'";
            }
            else
            {
              tempSSQL += " SalgsDato LIKE '" + salgsDato + "%'";
            }
    }
        if (adresse != "Adresse" && adresse != "")
    {
            if ((CheckIfContains.Any(tempSSQL.Contains)))
            {
              tempSSQL += " AND Adresse LIKE '%" + adresse + "%'";
            }
            else
            {
              tempSSQL += " Adresse LIKE '%" + adresse + "%'";
            }
    }
        if (startPris != "Startpris" && startPris != "")
    {
            if(comboBoxEjendomStartPris == 0)
            {
              if ((CheckIfContains.Any(tempSSQL.Contains)))
              {
                 tempSSQL += " AND StartPris >= '" + startPris + "'";
              }
              else
              {
                 tempSSQL += " StartPris >= '" + startPris + "'";
              }
            }
            if(comboBoxEjendomStartPris == 1)
            {
              if ((CheckIfContains.Any(tempSSQL.Contains)))
              {
                 tempSSQL += " AND StartPris <= '" + startPris + "'";
              }
              else
              {
```

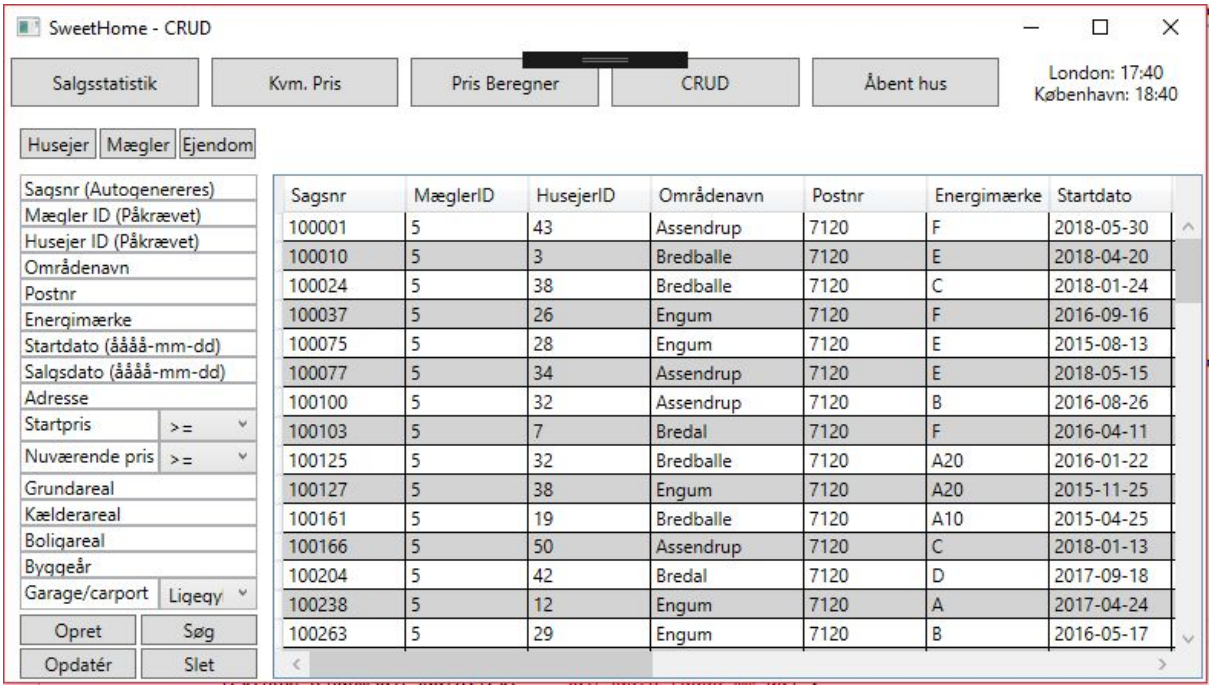
```
                tempSSQL += " StartPris <= '" + startPris + "'";
              }
            }
        }
        if (nuværendePris != "Nuværende pris" && nuværendePris != "")
        {
            if (comboBoxEjendomNuværendePris == 0)
            {
              if ((CheckIfContains.Any(tempSSQL.Contains)))
              {
                tempSSQL += " AND NuværendePris >= '" + nuværendePris + "'";
              }
              else
              {
                tempSSQL += " NuværendePris >= '" + nuværendePris + "'";
              }
            }
            if (comboBoxEjendomNuværendePris == 1)
            {
              if ((CheckIfContains.Any(tempSSQL.Contains)))
              {
                tempSSQL += " AND NuværendePris <= '" + nuværendePris + "'";
              }
              else
              {
                tempSSQL += " NuværendePris <= '" + nuværendePris + "'";
              }
            }
        }
        if (grundAreal != "Grundareal" && grundAreal != "")
        {
            if ((CheckIfContains.Any(tempSSQL.Contains)))
            {
              tempSSQL += " AND GrundAreal LIKE '" + grundAreal + "%'";
            }
            else
            {
              tempSSQL += " GrundAreal LIKE '" + grundAreal + "%'";
            }
        }
        if (kælderAreal != "Kælderareal" && kælderAreal != "")
        {
            if ((CheckIfContains.Any(tempSSQL.Contains)))
            {
              tempSSQL += " AND KælderAreal LIKE '" + kælderAreal + "%'";
            }
            else
            {
              tempSSQL += " KælderAreal LIKE '" + kælderAreal + "%'";
            }
        }
        if (boligAreal != "Boligareal" && boligAreal != "")
        {
            if ((CheckIfContains.Any(tempSSQL.Contains)))
            {
              tempSSQL += " AND BoligAreal LIKE '" + boligAreal + "%'";
            }
            else
            {
              tempSSQL += " BoligAreal LIKE '" + boligAreal + "%'";
            }
```

```
        }
               if (byggeår != "Byggeår" && byggeår != "")
        {
                    if ((CheckIfContains.Any(tempSSQL.Contains)))
                    {
                      tempSSQL += " AND Byggeår LIKE '" + byggeår + "%'";
                    }
                    else
                    {
                      tempSSQL += " Byggeår LIKE '" + byggeår + "%'";
                    }
        }
               if (garageCarport == 0) // ComboBoxEjendomGarageCarport index
        {
                    if ((CheckIfContains.Any(tempSSQL.Contains)))
                    {
                      tempSSQL += " AND GarageCarport = 1";
                    }
                    else
                    {
                      tempSSQL += " GarageCarport = 1";
                    }
        }
               if (garageCarport == 1) // ComboBoxEjendomGarageCarport index
           {
                    if ((CheckIfContains.Any(tempSSQL.Contains)))
                    {
                      tempSSQL += " AND GarageCarport = 0";
                    }
                    else
                    {
                      tempSSQL += " GarageCarport = 0";
                    }
        }

        if (garageCarport == 2) // ComboBoxEjendomGarageCarport index
        {
                    if ((CheckIfContains.Any(tempSSQL.Contains)))
                    {
                      tempSSQL += " AND GarageCarport IN (0, 1)";
                    }
                    else
                    {
                      tempSSQL += " GarageCarport IN (0, 1)";
                    }
        }

        string sSQL = tempSSQL + ";";
        SqlCommand command = new SqlCommand(sSQL, ControllerConnection.conn);
        SqlDataAdapter sda = new SqlDataAdapter(command);
        DataTable dt = new DataTable("Ejendom");
        sda.Fill(dt);
        dg.ItemsSource = dt.DefaultView;
            }
```

SøgEjendom() fungerer ved at den har en tempSSQL string, som der sættes yderligere sql kode ind i enden af, ved at løbe igennem en masse *if-statements* for alle tekstboksene, samt comboboxene, og tjekker for hvor der er ændret tekstindhold. Der, hvor der er ændret tekstindhold, tjekkes så via *CheckIfContains*-arrayet og .Contains, for om tempSSQL stringen har fået sat tekst ind via nogle af de

forrige *if-statements*. Hvis der er blevet sat ind tidligere, så indsættes teksten med " AND" foran. Til slut afsluttes der ved at oprette sSQL stringen og sætte den lig med tempSSQLL + ";", hvorefter queryen køres og dataen læses ind i datagrid'et.

Søgefunktion i ejendom



Dette er et view af en søgning på denne følgende SQL string: "SELECT * FROM Ejendom WHERE MæglerID = 5 AND Postnr LIKE 7120%;

Slet

ButtonSlet_Click

```
private void ButtonSlet_Click(object sender, RoutedEventArgs e)
        {
        if (WrapPanelHusejer.IsVisible && TextBoxHusejerID.Text != "ID
                  (Autogenereres)" && TextBoxHusejerID.Text != "")
        {

ControllerCrudHusejer.SletHusEjer(Convert.ToInt32(TextBoxHusejerID.Text));

                  ControllerCrudHusejer.LæsHusejer(DataGridHusejer);

                  TextBoxHusejerID.Text = "ID (Autogenereres)";
                  TextBoxHusejerNavn.Text = "Navn";
                  TextBoxHusejerEmail.Text = "Email";
                  TextBoxHusejerTelefon.Text = "Telefon";
                }
        if (WrapPanelMægler.IsVisible && TextBoxMæglerID.Text != "ID
                  (Autogenereres)" && TextBoxMæglerID.Text != "")
        {
                  ControllerCrudMægler.SletMægler(Convert.ToInt32(TextBoxMæglerID.Text));

                  ControllerCrudMægler.LæsMægler(DataGridMægler);

                  TextBoxMæglerID.Text = "ID (Autogenereres)";
```

```
                TextBoxMæglerNavn.Text = "Navn";
                TextBoxMæglerTelefon.Text = "Telefon";
                TextBoxMæglerEmail.Text = "Email";
        }
        if (WrapPanelEjendom.IsVisible && TextBoxEjendomSagsnr.Text != "Sagsnr
                (Autogenereres)" && TextBoxEjendomSagsnr.Text != "")
        {

ControllerCrudEjendom.SletEjendom(Convert.ToInt32(TextBoxEjendomSagsnr.Text));

                ControllerCrudEjendom.LæsEjendom(DataGridEjendom);

                TextBoxEjendomSagsnr.Text = "Sagsnr (Autogenereres)";
                TextBoxEjendomMæglerID.Text = "Mægler ID (Påkrævet)";
                TextBoxEjendomHusejerID.Text = "Husejer ID (Påkrævet)";
                TextBoxEjendomOmrådeNavn.Text = "Områdenavn";
                TextBoxEjendomPostnr.Text = "Postnr";
                TextBoxEjendomEnergiMærke.Text = "Energimærke";
                TextBoxEjendomStartDato.Text = "Startdato (åååå-mm-dd)";
                TextBoxEjendomSalgsDato.Text = "Salgsdato (åååå-mm-dd)";
                TextBoxEjendomAdresse.Text = "Adresse";
                TextBoxEjendomStartPris.Text = "Startpris";
                TextBoxEjendomNuværendePris.Text = "Nuværende pris";
                TextBoxEjendomGrundAreal.Text = "Grundareal";
                TextBoxEjendomKælderAreal.Text = "Kælderareal";
                TextBoxEjendomBoligAreal.Text = "Boligareal";
                TextBoxEjendomByggeår.Text = "Byggeår";
                ComboBoxEjendomGarageCarport.SelectedIndex = 2;
        }
        }
```

ButtonSlet_Click

Ved tryk på 'slet' knappen tjekkes der hvilket WrapPanel/view der er aktivt og der tjekkes om der er indtastet et ID eller i dette tilfælde et Sagsnr. Hvis der findes et indtastet køres SletEjendom() og derefter LæsEjendom().

SletEjendom()

```
public static void SletEjendom(int sagsnr)  //
        {
        string sSQL = "DELETE FROM Ejendom WHERE Sagsnr = '" + sagsnr + "';";
        SqlCommand command = new SqlCommand(sSQL, ControllerConnection.conn);
        command.ExecuteNonQuery();
        }
```

SletEjendom()

Her slettes ejendommen via en simpelt SQL query hvor Sagsnr stemmer overens med TextBoxEjendomSagsnr.Text.

Jeg finder det ikke nødvendigt at vise et billede af dette, da det siger sig selv hvad der er sket.

Pris Beregner (Patrick)

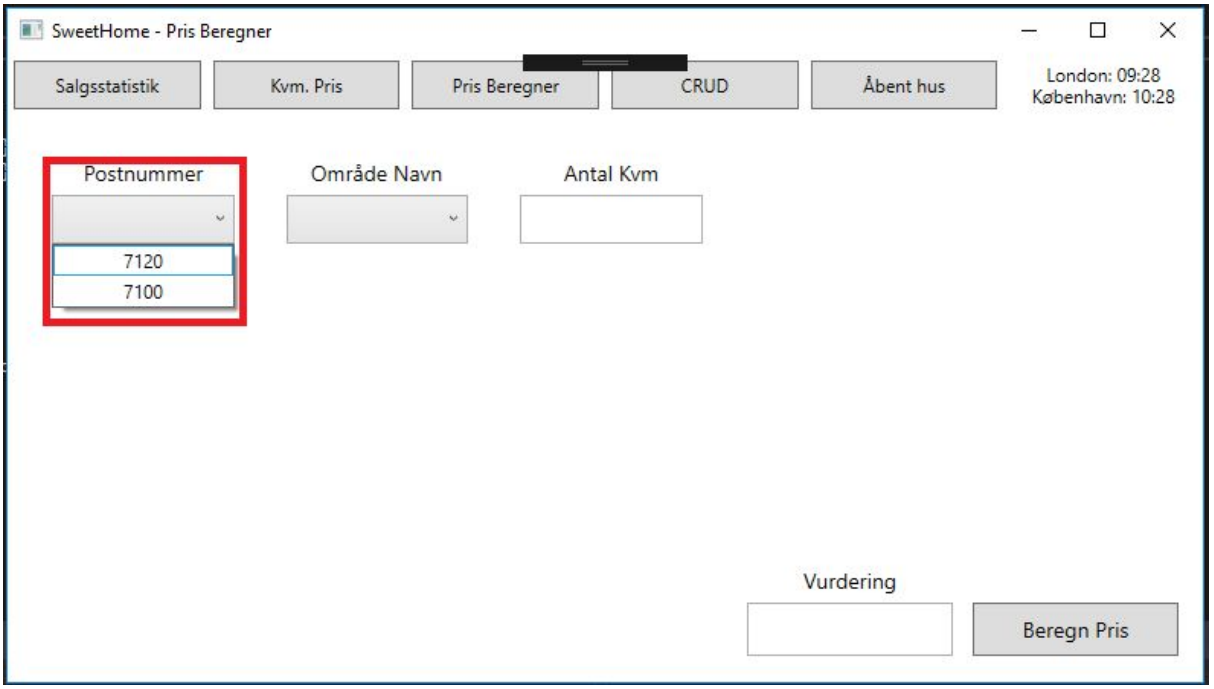
Opret postnumre i ComboBox.

```
public static void ComboBoxOpretPostnr(ComboBox comboboxPrisBeregner_Postnr)
        {
            SqlDataReader reader = null;
            try
            {
                SqlCommand cmd = new SqlCommand("SELECT Postnr FROM Område;",
                ControllerConnection.conn);
                reader = cmd.ExecuteReader();

                while (reader.Read())
                {
                    int sPostnr = reader.GetInt32(0);
                    bool hasItem = comboboxPrisBeregner_Postnr.Items.Contains(sPostnr);

                    if (hasItem)
                    {
                        Debug.WriteLine("Postnummer findes allerede!");
                    }
                    else
                    {
                        comboboxPrisBeregner_Postnr.Items.Add(sPostnr);
                    }
                }
            }
            catch (Exception)
            {
                Debug.WriteLine("Could not find database/table" + "\n");
            }
            finally
            {
                reader.Close();
            }

        }
```

Den ovenstående metode kommer fra ControllerPrisBeregner klassen og bliver kaldt en gang i MainWindow klassen, når programmet starter. Metoden har også en reference til ComboBox comboboxPrisBeregner_Postnr som befinder sig i MainWindow klassen.
Vi har en SqlDataReader som bliver udført efter at SqlCommand har fundet Postnr data fra Område tabellen i databasen. - Læs SQL Prisberegner afsnittet.
Der bliver så løbet igennem Postnr kolonnen og for hver værdi der bliver læst, sætter vi int sPostnr til at være den værdi. sPostnr bliver så tilføjet til ComboBox comboboxPrisBegner_Postnr.

Da der godt kan være flere områder med samme postnumre, har vi lavet en bool som bliver “true” hvis comboboxPrisBeregner_Postnr allerede har den værdi.

Da metoden kræver data fra en Database, har vi brugt try, catch and finally. Så vi får besked hvis forbindelsen til Databasen ikke findes.



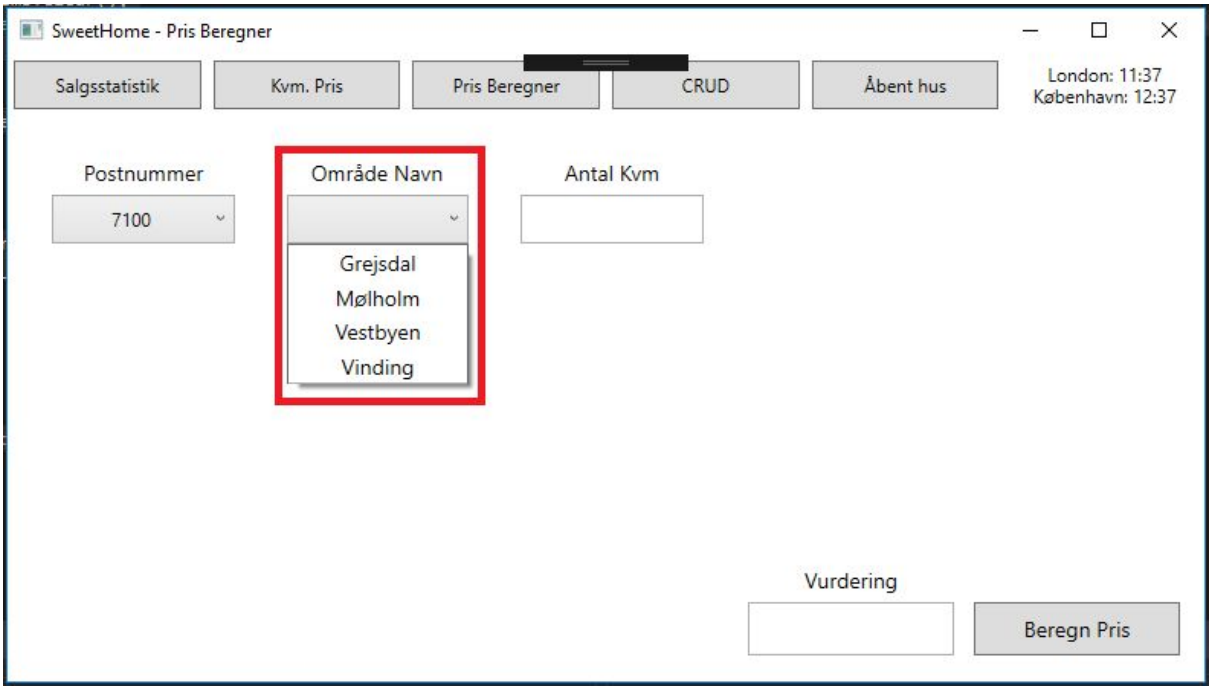
Opret Navne i ComboBox

```
public static void ComboBoxOpretNavn(ComboBox comboboxPrisBeregner_Navn, ComboBox
comboboxPrisBeregner_Postnr)
       {
           comboboxPrisBeregner_Navn.Items.Clear();
           SqlCommand cmd = new SqlCommand("SELECT Navn FROM Område WHERE Postnr = " +
           comboboxPrisBeregner_Postnr.Text + ";", ControllerConnection.conn);
           SqlDataReader reader = null;
           try
           {
               reader = cmd.ExecuteReader();

               while (reader.Read())
               {

                   string sName = reader.GetString(reader.GetOrdinal("Navn"));
                   comboboxPrisBeregner_Navn.Items.Add(sName);

               }

           }
           catch (Exception)
           {
               Debug.WriteLine("Could not find database/table - CLOSE");
           }
           finally
           {
               reader.Close();
           }
       }
```

Metoden ComboBoxOpretNavn() bliver kaldt fra Combobox comboboxPrisBeregner_Postnr via et Close event. Det vil sige at når man har valgt et postnr i GUI og ComboBoxen lukkes, bliver denne metode kaldet. Metoden har referencer til ComboBox comboboxPrisBeregner_Navn og ComboBox comboboxPrisBeregner_Postnr.

Denne metode er meget ligesom ComboBoxOpretPostnr() metoden.
Vi søger Databasen via SqlCommand efter de navne som har postnummeret, der er blevet valgt i comboboxPrisBeregner_Postnr. - Læs SQL Prisberegner afsnittet.
De navne bliver så læst og bliver tilføjet til comboboxPrisBeregner_Navn når SqlDataReader bliver udført.

Igen har vi en try and catch i det tilfælde at der skulle være en fejl med forbindelsen til Databasen.

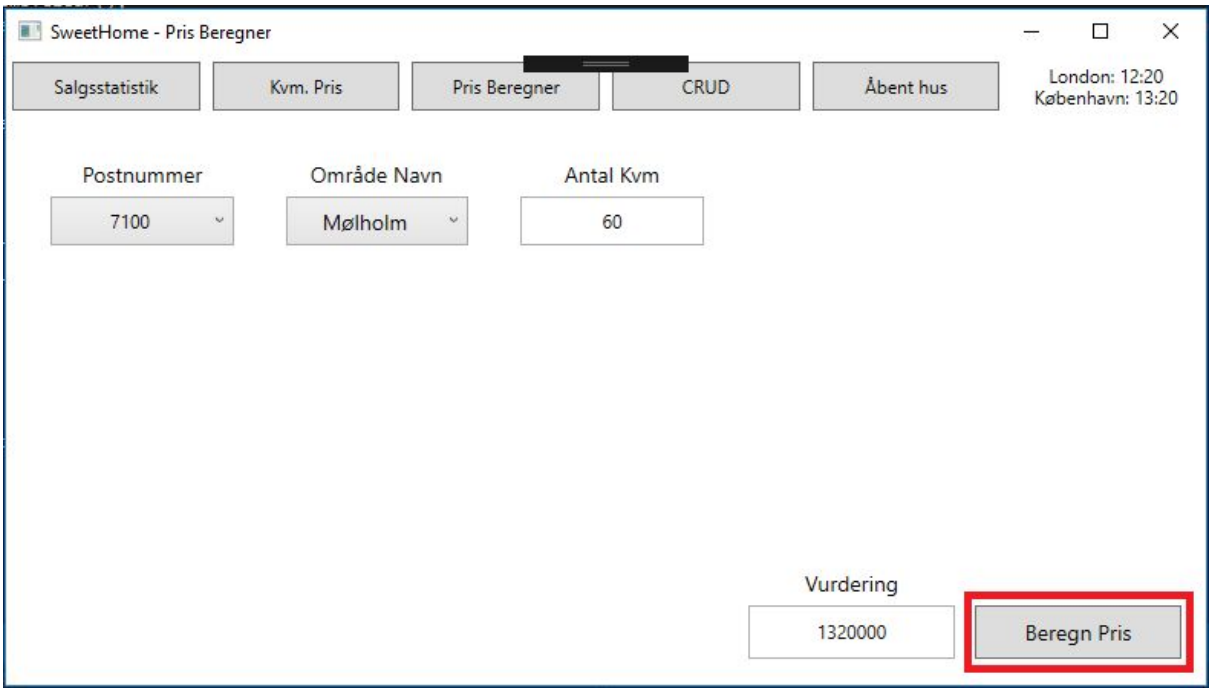


Prisberegner button click metode

```
private void ButtonPrisBeregner_BeregnPrisClick(object sender, RoutedEventArgs e)
      {
          try
          {
              textboxPrisBeregner_Vurdering.Text =
              Convert.ToString(ControllerPrisBeregner.BeregnPris(Convert.ToInt32(
              comboboxPrisBeregner_Postnr.Text), comboboxPrisBeregner_Navn.Text,
              Convert.ToInt32(textbox_PrisBeregnerKVM.Text)));
          }
          catch (Exception)
          {
              Debug.WriteLine("Tager ikke imod string!");
              textbox_PrisBeregnerKVM.Clear();
          }
      }
```

Metoden ButtonPrisBeregner_ReregnPrisClick() kommer fra MainWindow klassen og spørger efter Pris vurderingen ved at kalde metoden BeregnPris() i klassen ControllerPrisBeregner med tre parameter, comboboxPrisBeregner_Postnr.Text, comboboxPrisBeregner_Navn.Text, og textbox_PrisBeregnerKVM.Text. Tre input felter i GUI, som skal bruges i metoden ControllerPrisBeregner.BeregnPris()
textboxPrisBeregner_Vurdering.Text bliver til det returneret værdi fra den kaldte metode

Vi har en try and catch for at forhindre programmet i at stoppe, skulle brugeren indtaste en fejl.



BeregnPris og FindPrisFaktor metoder

```
public static object BeregnPris(int postnr, string navn, int antalKvm)
        {
            double prisFaktor = FindPrisFaktor(postnr, navn);
            double kvmPris = 22000;
            //Udregning
            double vurderingsPris = (antalKvm * kvmPris) * prisFaktor;
            return vurderingsPris;
        }

private static double FindPrisFaktor(int postnr, string navn)
        {

            //SQL code here
            SqlCommand cmd = new SqlCommand("SELECT PrisFaktor FROM Område WHERE Navn = '" +
            navn + "' AND Postnr = " + postnr + ";", ControllerConnection.conn);
            double prisFaktor = 0;
            SqlDataReader reader = null;
            try
            {
                using (reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        prisFaktor = Convert.ToDouble((reader["PrisFaktor"]));
                    }
                }
                return prisFaktor;

            }
            catch (Exception)
            {
                Debug.WriteLine("Failed to connect to the database" + "\n");
                return 0;
            }
            finally
            {
                reader.Close();
            }
        }
```

Først har vi metoden som blev kaldt af Button Click Metoden som vi beskrev før. Vi har de 3 attributter i parameter, som har dataen fra de tre input i GUI.
Men før beregning kan ske, skal vi først finde Pris Faktoren i tabellen Område i databasen.
Den finder vi ved at kalde metoden FindPrisFaktor fra BeregnPris metoden.

I FindPrisFaktor metoden bruger vi igen den samme kode fra tidligere metoder, med små ændringer. SqlCommand bliver kaldt og finder PrisFaktoren ud fra de to parameter i metoden, navn og postnr. - SQL Prisberegner afsnittet.
Der bliver derefter lavet en prisFaktor attribut, som får en værdi efter at SqlDataReader bliver kaldt.

prisFaktor bliver returneret til BeregnPris metoden. Der bliver brugt try and catch, da brugeren kan lave fejl eller at der ingen forbindelse er til Databasen. Skulle det ske, returnere vi 0 i stedet for prisFaktor.
Nu har BeregnPris metoden de værdier den skal bruge til at lave udregningen. kvmPris som vi har sat til 22000, er et gennemsnit af hele landets pris/kvadratmeter.

Beregningen bliver udført og bliver returneret til ButtonPrisBeregner_BeregnPrisClick metoden, som sætter textboxPrisBeregner_Vurdering.Text til den returnerede værdi.

Multithreaded Ur (Jonas og Alex)

```
    public static void København()
    {
        while (true)
        {
            string hour = DateTime.Now.Hour.ToString("00");
            string minute = DateTime.Now.Minute.ToString("00");

            københavn = String.Format("København: {0}:{1}", hour, minute);

            MainWindow.instance.KøbenhavnUr = københavn;

            Thread.Sleep(60000);
        }
    }
```

Ovenstående kode ligger i ControllerUr klassen og metoden bruger sin egen tråd til at holde uret med københavns tid opdateret, tilsvarende metode findes også til londons ur, dog bliver der her altid trukket en time fra, da london er en time bagved københavn. I MainWindow klassen (Vores view) er der en string ved navn KøbenhavnUr som opdateres hvert minut.

```
Thread threadLondonUr = new Thread(new ThreadStart(ControllerUr.London));
Thread threadKøbenhavnUr = new Thread(new ThreadStart(ControllerUr.København));

threadLondonUr.Start();
threadKøbenhavnUr.Start();
```

Den ovenstående kode ligger i MainWindow klassen og viser trådene som håndtere uret til københavn samt uret til london.

```
        internal string LondonUr
        {
            get { return LabelLondon.Content.ToString(); }
            set { Dispatcher.Invoke(new Action(() => {
                  LabelLondon.Content = value; })); }
        }


        internal string KøbenhavnUr
        {
            get { return LabelKøbenhavn.Content.ToString(); }
            set { Dispatcher.Invoke(new Action(() => {
                  LabelKøbenhavn.Content = value; })); }
        }
```

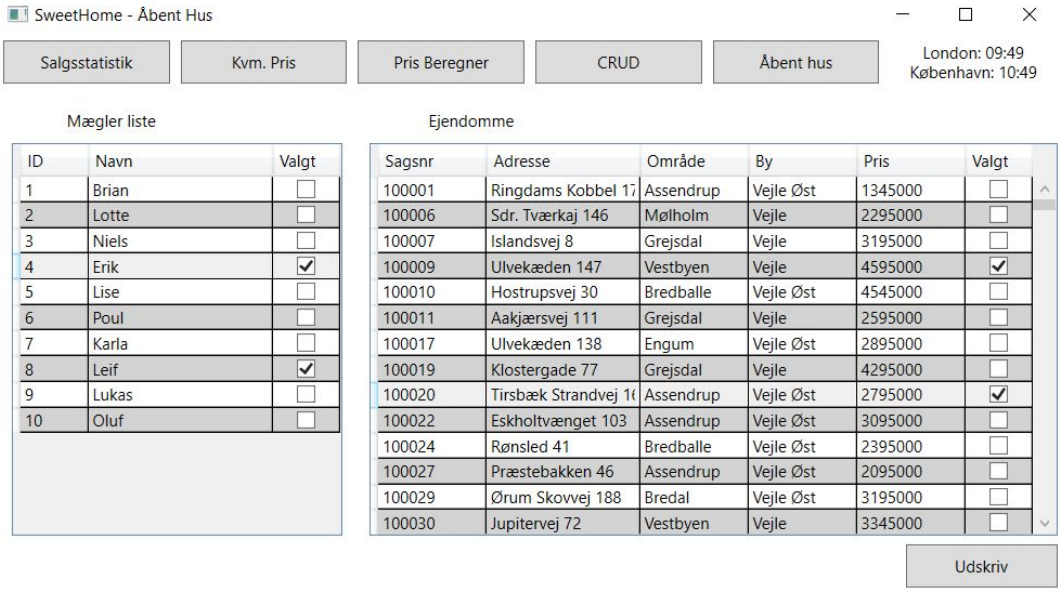
Den ovenstående kode ligger i MainWindow klassen. De to strings bruges til at opdatere teksten på viewet. Da viewet er styret af hoved tråden kan urenes egne tråde ikke tilgå det direkte, derfor bliver der dette uddeligeret til hovedtråden hver gang en stringens værdi ændre sig. Urene opdateres kun en gang i minuttet, derfor lægges der minimalt arbejde over på hoved tråden via denne metode.

"Åbent Hus" (Jonas og Alex)

Forsiden

```
public static void FyldMæglerDatagrid(DataGrid dg)
{
    string sSQL = "select ID, Navn from Mægler;";
    SqlCommand command = new SqlCommand(sSQL, ControllerConnection.conn);
    SqlDataAdapter sda = new SqlDataAdapter(command);
    DataTable dt = new DataTable("Mæglere");
    sda.Fill(dt);

    mæglere.Clear();

    foreach (DataRow row in dt.Rows)
    {
        object[] array = row.ItemArray;
        mæglere.Add(new ModelÅbentHusMægler((int)array[0], array[1].ToString()));
    }
    dg.DataContext = mæglere;
}
```

Den ovenstående kode ligger i ControllerÅbenthus klassen. Metoden kaldes når åbent hus vinduet bliver vist, og bruges til at læse alle mæglere ind fra databasen, og gemme dem over i en liste af mæglere for til sidst at vise dem i data griddet på skærmen. For at gemme mæglerne i en liste bruges modellen ModelÅbentHusMægler, som indeholder mæglers navn og id. Tilsvarende metode findes for ejendomme. Nedenfor ses hvordan viewet ser ud når metoderne er kaldt.

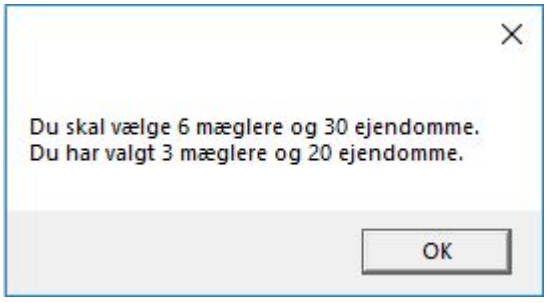


Udskriv

Udskriv metoden består af flere dele, først skal en liste af 30 ejendomme kombineres med en liste af 6 mæglere, herefter skal den så udskrives.

```
public static void TagCheckedMæglereOgEjendomme()
{
    if (antalMæglereValgt == 6 && antalEjendommeValgt == 30)
    {
        foreach (ModelÅbentHusMægler mægler in mæglere)
        {
            if (mægler.IsChecked == true)
            {
                valgteMæglere.Add(mægler);
            }
        }

        foreach (ModelÅbentHusEjendom ejendom in ejendomme)
        {
            if (ejendom.IsChecked == true)
            {
                valgteEjendomme.Add(ejendom);
            }
        }
    }
}
```

Denne metoder flytter de valgte mæglere og ejendomme over i nye lister, som præcist indeholder 6 mæglere og 30 ejendomme. Hvis brugeren har valgt et andet antal af nogle af disse, vil intet ske i metoden, dog vil der i viewet blive vist en infoboks hvor der står hvor mange du har valgt og hvor mange du bør vælge. Et eksempel kan ses nedenfor.



Efter de to lister er lavet, skal de så samles til en liste hvor de 30 huse fordeles på de 6 mæglere, dette sker i metoden GenererListe.

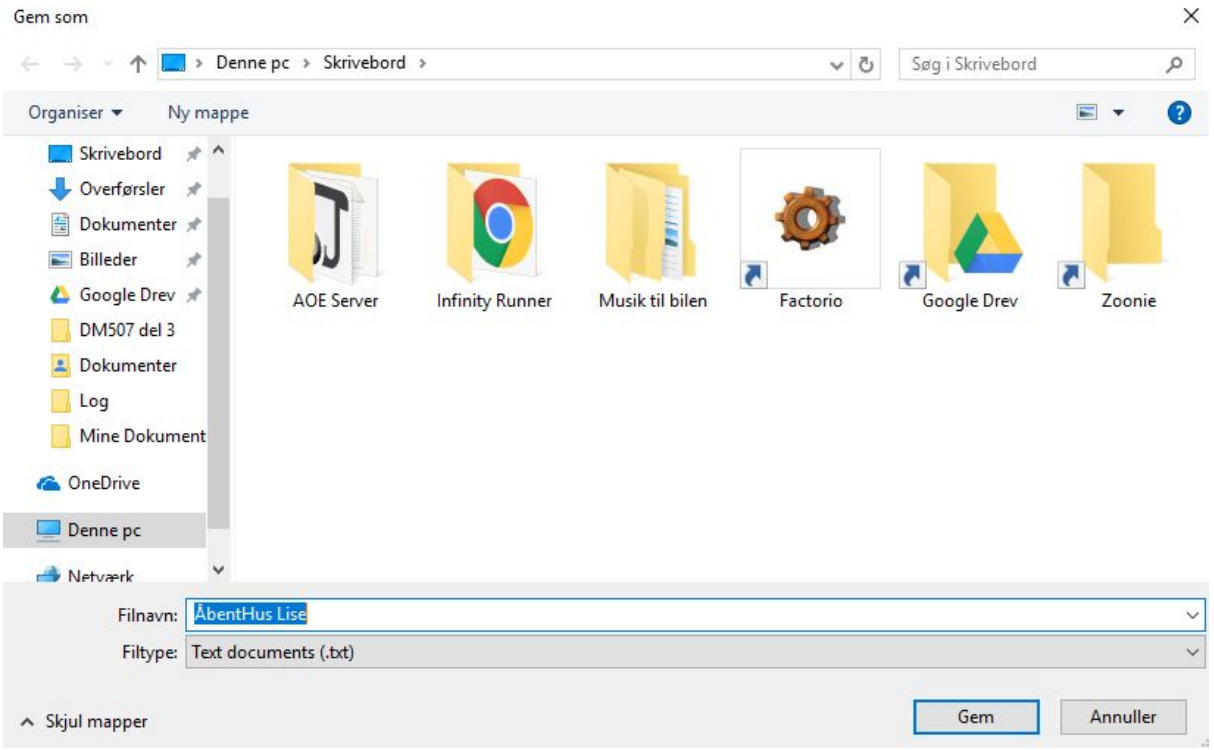
```
public static void GenererListe(string udfil)
{
    List<ModelÅbentHusEjendom> sorteretListeAfEjendomme =
    valgteEjendomme.OrderBy(ejendom => ejendom.Pris).ToList();

    sorteretListeAfEjendomme.Reverse();

    int mæglerNr = 0;
    while (sorteretListeAfEjendomme.Count > 0)
    {
        ModelÅbentHusEjendom ejendom = sorteretListeAfEjendomme[0];
        sorteretListeAfEjendomme.RemoveAt(0);
```

```
            ModelÅbentHusMægler mægler = valgteMæglere[mæglerNr];
            mæglerNr = (mæglerNr + 1) % 6;
            ModelÅbentHus åbenthus = new ModelÅbentHus
            (mægler.ID, mægler.Navn, ejendom.Sagsnr, ejendom.Adresse,
             ejendom.Område, ejendom.By, ejendom.Pris);
            åbentHusListe.Add(åbenthus);
        }
        List<ModelÅbentHus> sorteretÅbentHusListe =
        åbentHusListe.OrderBy(sag => sag.MæglerId).ToList();

        Udskriv(dg, udfil, sorteretÅbentHusListe);
    }
```

GenererListe starter med at sortere listen af ejendomme og derefter reverese den, da det giver en liste med den dyreste bolig først i listen, og den billigste bagerst.
Herefter køres en while lykke hvor den første ejendomme i listen sættes sammen med en mægler og ligges i den nye liste, ejendommen fjernes så fra den gamle liste, så den der nu ligge først er den anden dyreste. MæglerNr variablen lægges der 1 til, og så bruges modulus 6, da vi skal have tallet til at cykle imellem 0 og 5.
Til sidst uden for while lykken, sorteres listen af de 30 ejendomme sammensat med mæglere efter mægler ID da dette giver et bedre overblik over resultatet. Herefter kaldes udskriv metoden som vises nedenfor.

```
    public static void Udskriv(string udfil, List<ModelÅbentHus> liste)
    {
        StreamWriter stream = null;

        try
        {
            stream = new StreamWriter(udfil);
            stream.WriteLine(" Mægler ID  | Mægler Navn | SagsNR |
            Adresse          |   Område   |     By     |   Pris");

            foreach (ModelÅbentHus item in liste)
            {
                stream.Write(string.Format(" {0,-10} | ", item.MæglerId.ToString()));
                stream.Write(string.Format("{0,-11} | ", item.MæglerNavn.ToString()));
                stream.Write(string.Format("{0,-6} | ", item.Sagsnr.ToString()));
                stream.Write(string.Format("{0,-25} | ", item.Adresse.ToString()));
                stream.Write(string.Format("{0,-10} | ", item.Område.ToString()));
                stream.Write(string.Format("{0,-10} | ", item.By.ToString()));
                stream.WriteLine(string.Format("{0,-8}", item.Pris.ToString()));
            }
        }
        catch (System.Exception)
        {

        }
        finally
        {
            if (stream != null)
            {
                stream.Close();
            }
        }
```

Udskriv metoden bruger en StreamWriter til at skrive til en valgt fil, filen vælges via en SaveFileDialog i viewet. Først udskrives en overskrifts linje, og hefter bruges en foreach til at udskrive hvert element i listen af Ejendomme sammensat med mæglere, her bruges der string formatting for at gøre det pænere at se på. Nedenfor ses SaveFileDialogen i bruge, samt et eksemple på en udskrift.



Mægler ID	Mægler Navn	SagsNR	Adresse	Område	By	Pris
1	Brian	100068	Danasvej 53	Bredballe	Vejle Øst	5645000
1	Brian	100009	Ulvekæden 147	Vestbyen	Vejle	4595000
1	Brian	100022	Eskholtvænget 103	Assendrup	Vejle Øst	3095000
1	Brian	100080	Fasanstien 104	Engum	Vejle Øst	2395000
1	Brian	100050	Højen Kirkevej 154	Bredal	Vejle Øst	1495000
4	Erik	100059	Glentevej 22	Vinding	Vejle	5495000
4	Erik	100092	Staldgaardsgade 179	Grejsdal	Vejle	4195000
4	Erik	100086	Hjulmagervej 145	Grejsdal	Vejle	3045000
4	Erik	100024	Rønsled 41	Bredballe	Vejle Øst	2395000
4	Erik	100100	Ulvekæden 189	Assendrup	Vejle Øst	1445000
5	Lise	100042	Ulfsvej 170	Vinding	Vejle	5245000
5	Lise	100043	Horskærvænget 54	Grejsdal	Vejle	3595000
5	Lise	100020	Tirsbæk Strandvej 160	Assendrup	Vejle Øst	2795000
5	Lise	100046	Skorpionen 51	Bredal	Vejle Øst	2195000
5	Lise	100071	Brummersvej 58	Assendrup	Vejle Øst	1445000
6	Poul	100045	Timianvej 193	Vestbyen	Vejle	5195000
6	Poul	100081	Nyborgvej 187	Vinding	Vejle	3395000
6	Poul	100011	Aakjærsvej 111	Grejsdal	Vejle	2595000
6	Poul	100027	Præstebakken 46	Assendrup	Vejle Øst	2095000
6	Poul	100098	Elsdyrvej 54	Grejsdal	Vejle	1345000
7	Karla	100102	Aakjærvænget 153	Vinding	Vejle	5095000
7	Karla	100041	Bagsnoge 111	Mølholm	Vejle	3345000
7	Karla	100065	Elmegårdsvej 49	Mølholm	Vejle	2445000
7	Karla	100052	Pomonavej 96	Bredal	Vejle Øst	1845000
7	Karla	100101	Musvitvej 69	Engum	Vejle Øst	1145000
9	Lukas	100051	Solbakken 38	Vestbyen	Vejle	5045000
9	Lukas	100007	Islandsvej 8	Grejsdal	Vejle	3195000
9	Lukas	100083	Egelundvej 126	Assendrup	Vejle Øst	2395000
9	Lukas	100060	Dalvej 80	Assendrup	Vejle Øst	1595000
9	Lukas	100047	Jernvej 131	Assendrup	Vejle Øst	1145000

Virksomheden

Business case: Sweet Home Ejendomsmæglerne (Daniel)

Cost-benefit analyse

<u>Faktorer med indflydelse på cost/benefit:</u>

- Den samlede udviklingstid
    - 888 mandetimer på 6 ugers projekt.
    - 300 kr. per mandetime – 37 timers uge.
    - § 4 mand – 266.400 kr. for 6 ugers projekt.

- Sweet Homes udgifter til drift af servere, software m.v.
    - Gearhost: Database hosting. 25 dollars/156.92 kr. om måneden.

- Tidsbesparelse i forbindelse med indrapportering og samarbejde med eksterne partnere (tinglysning, banker, m.v.).
    - Vi vurderer at hver ejendomsmægler kan spare 1 time per dag, på at tidligere analoge arbejdsopgaver bliver digitaliserede og effektiviserede. Vi skønner, at der vil blive solgt 10% flere ejendomme ved brugen af det nye system.

- Sweet Home sælger 240 boliger om året i gennemsnit.
    - Der er ansat 10 mæglere.
    - Mæglerne får et salær på 1% af ejendomssalgene og de ligger gennemsnitligt på 30.000 kr.
        - Det giver en indtægt til Sweet Home på 7.200.000 kr. Årligt.
        - Halvdelen af salæret får mæglerne, halvdelen går til firmaet.

- Højere indtægter på salærerne ifht. bedre kvalitet i vurderingsarbejdet.
    - Vi vurderer, at salgsprisen af ejendomme i gennemsnit vil blive 2,5 % højere.

- Administrative besparelser i anvendelsen af integrerede oplysninger i den centrale SQL-database.
    - Nogen bliver fyret, formentlig sekretæren Jonna. Dvs. der spares en månedsløn på 29.000 kr. om måneden.

- Til brug for en beregning af Return Of Investment anvendes at diskonteringsfaktor er 4%.

Tal til beregning af Return of Investment

Udgifter

Engangsudgifter

- 266.400 kr.

Løbende udgifter:

- 156.92 kr. per måned eller 1883,04 kr. per år.
- 3.600.000 kr. årligt.

Besparelser/indtægter

Løbende besparelser/indtægter:

- 7.200.000 kr. årligt.
- 29.000 kr. sparet per måned eller 348.000 kr. per år.

Ændringer i omsætning

- Der sælges 10% flere ejendomme.
- Ejendommene sælges for 2,5% højere beløb.

Særlige detaljer

- Diskonteringsfaktor på 4%.

Cost-benefit model

Samlede udgifter		1. år	2. år	3. år
	Engangsudgifter	-277.056 kr.	0 kr.	0 kr.
	Løbende udgifter pr. år	-3.745.958,36 kr.	-3.895.796,69 kr.	-4.051.628,56 kr.
	Sum	-4.023.014,36 kr.	-3.895.796,69 kr.	-4.051.628,56 kr.
Samlede indtægter		**1. år**	**2. år**	**3. år**
	Engangsindtægter	0 kr.	0 kr.	0 kr.
	Løbende indtægter	8.442.720 kr.	8.780.428,80 kr.	9.131.645,95 kr.
	Sum	8.442.720 kr.	8.780.428,80 kr.	9.131.645,95 kr.
Opgørelse				
	Sum	4.696.761,64 kr.	4.884.632,11 kr.	5.080.017,39 kr.
Return of investment (samlet sum efter løbetiden)		**1. år**	**1. år**	**1. år**
	Sum	4.696.761,64 kr.	9.581.393,75 kr.	14.661.411,14 kr.

Interessent-analyse

Eksterne interessenter	Succeskriterier	Krav til systemegenskaber	Krav til projektegenskaber	Bidrag
Slutbruger (Sweet Home Ejendomsmæglerne)	Større omsætning Fortsat tilfredse kunder Lettere arbejdsgang	Sømløs integration med eksterne systemer	Kontrakten holdes	Krav Viden Forretningsgange
Slutbrugers kunder	Bedre salgspris			
Interne interessenter	**Succeskriterier**	**Krav til systemegenskaber**	**Krav til projektegenskaber**	**Bidrag**
Styregruppe	Indtjening på kontrakten			Beslutninger
Projektdeltagere	God omtale- ros		Godt arbejdsklima God arbejdsfordeling	Ansvar

Risikoanalyse

Risici	Sandsynlighed	Konsekvens	Produkt	Mulige foranstaltninger
1. Sygdom i projektgruppen	2	3	6	Længere arbejdsdage.
2. Manglende supervision (undervisere)	1	3	3	Lav aftaler med undervisere om supervision/vejledning.

Udviklingen af systemet ønskes understøttet af en business case, der dokumenterer systemets berettigelse. Business casen skal som minimum indeholde en cost/benefit analyse med anslåede tal.

Der indregnes tal for eksempel for den samlede udviklingstid gruppen skal bruge på systemet, prisen for udstyr, der anvendes under udviklingen og tal for den software der skal anvendes.

Dertil kan lægges de udgifter Sweet Home skal anvende til drift af systemet i form af servere, software m.v

Det forventes at systemet i sin endelige form vil give besparelser den tid ejendomsmæglerne skal bruge indrapportering og samarbejde med eksterne partnere (tinglysning, banker, m.v.). Systemet vil også kunne give indtægter i forbindelse med bedre kvalitet i vurderingsarbejdet og dermed højere indtægter på salærerne. Og der vil være generelle administrative besparelser i anvendelsen af integrerede oplysninger i den centrale SQL-database.
Til brug for en beregning af Return Of Investment anvendes at diskonteringsfaktor er 4%.

Business case suppleres med projektledelses-oplysninger (Daniel)

Business casen suppleres med projektledelses-oplysninger i forlængelse af projektledelses/UP-overvejelserne under 'Systemudvikling', f.eks. overvejelser omkring risiko og interessenter, typisk i forbindelse med inception-fasen.

BPR (Business Process Re-engineering) (Patrick)

Sweet Home Ejendomsmæglerne ønsker at optimere deres kunderettede opgaver, derfor vil de gerne have et informationssystem, som kan administrere de opgaver. Det effektiviserer arbejdsprocessen og vil reducere virksomhedens omkostninger, men resulterer også i medarbejderes fyringer.

I stedet for at have en medarbejder til at udregne en pris for et hus ud fra hans vurdering, bliver denne process nu automatiseret med Pris Beregning systemet. Systemet kan selv ud fra husets informationer vurdere om huset er attraktivt ellelr ikke attraktivt og kan sammen med ejendomsmæglerens vurdering beslutte sig til en pris. Der er ikke længere brug for en medarbejder til at udregne huset pris og vil derfor, højst sandsynligt, blive fyret. Nu er der en mindre medarbejder der skal have løn, hvilket reducere virksomhedens omkostninger.

Konklusion

Vi har fået oprettet et program til Sweethome, som kan søge og udskrive salgsstatistikker og kvm. priser over tidligere boligsalg. Der er blevet oprettet en prisberegner som kan udregne en vejledende pris på en bolig i et bestemt område baseret på områdets prisfaktor og ejendommes boligareal.
Den ønskede Åbent hus funktion er oprettet. Programmet indeholder CRUD af mæglere, husejere og ejendomme. Multithreading Ur som viser tidszonerne for København og London er implementeret.
Programmet indeholder alle de ønskede funktioner som Sweethome havde.

Perspektivering

Der er en del ting vi kunne tænke os at have kigget på, hvis vi havde haft mere tid.
Det første er GUI'et. Det ville vi gerne have haft tid til at finpudse og pynte på. Bl.a. ville vi have gjort hele vores GUI i stand til at auto resize indholdet, hvilket vi ikke fik gjort rigtigt.
Det næste er, at vi tænker det kunne være sjovt at prøve at gøre prisberegneren mere kompleks, så den tager højde for mere end kun kvm. prisen, grundarealet og prisfaktoren. Der ville f.eks. Kunne tages højde for afstand til skole og indkøb, om der er kort afstand til vand, udsigten osv.
En anden ting vi kunne tænke os at arbejde med, ville være at lave programmet om til en webapplikation. Det tænker vi er mere fleksibelt og ekspanderbart, samt mere brugervenligt, da det ikke kræver installation og opdatering. Det vil også være nemmere at vedligeholde.

Litteraturliste

Der er ingen henvisning til litteratur

Bilag

Der er ingen bilag