# Design og implementering

## Overoverdnet design

Deployment-diagrammet i **Figur 1** viser, hvordan de interne dele af Smart Fridge interagerer. Som det fremgår, interagerer Bruger kun med enten en *Fridge app* eller *Web app*, og har altså ingen direkte interaktion med den bagvedliggende logik.



**Figur 1** Deployment Diagram for hele SmartFridge-systemet

For et mere detaljeret Deployment Diagram, henvises til **bilag XX**.

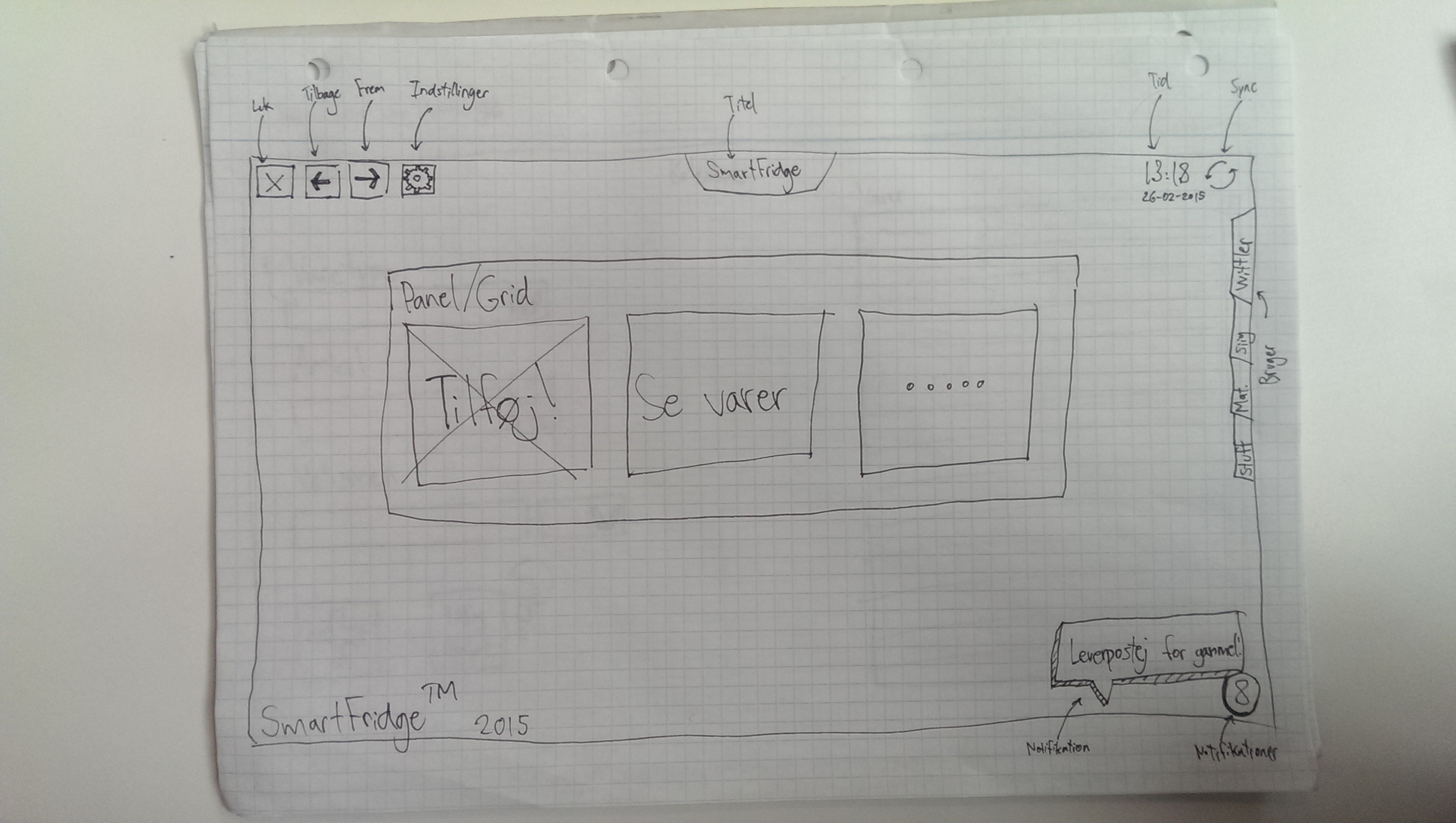
## Fridge app

### Design

Dette afsnit indeholder de grafiske og kodemæssige designovervejelser for de forskellige vinduer i Fridge App. Overvejelserne er baseret på en brainstorm (**bilag XX**). For større billeder, henvises til **bilag XX**.

#### Hovedmenu og ramme

Designet af hovedmenuen og rammen er blevet skitseret som i **Figur 2**.



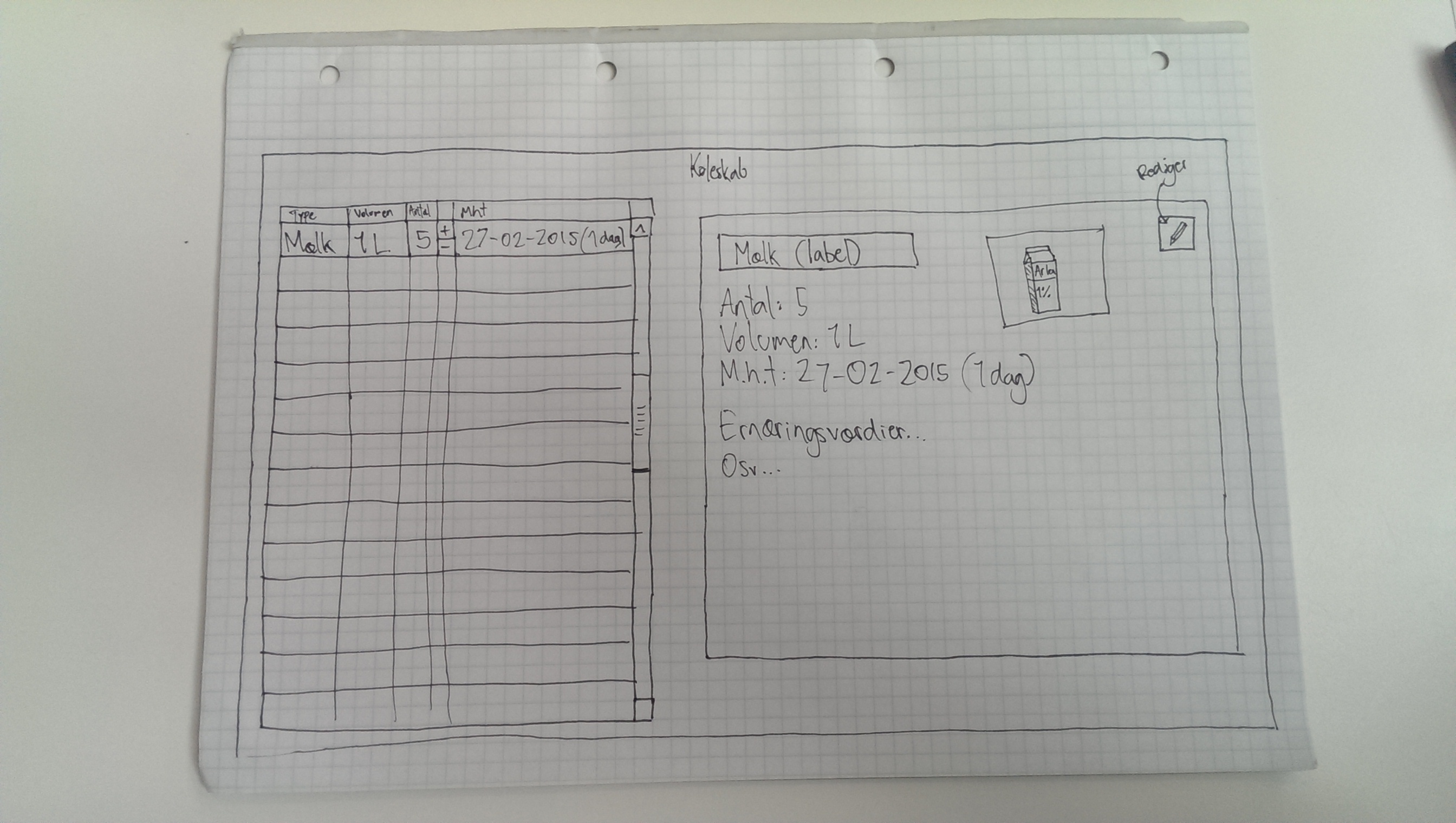
**Figur 2** Skitse af hovedmenu og ramme

Rammen, som er alt omkring firkanten *Panel/Grid*, vil ikke ændres, når der ændres kontekst.

En senere ændring, i forhold til skitsen, har været at knapperne i *Panel/Grid* vil være direkte henvisende til de eksisterende lister, hvorfra det vil være muligt at tilføje, redigere og fjerne varer direkte.

#### Se varer

Designet af menuen for “Se varer” er blevet skitseret som i **Figur 3**.



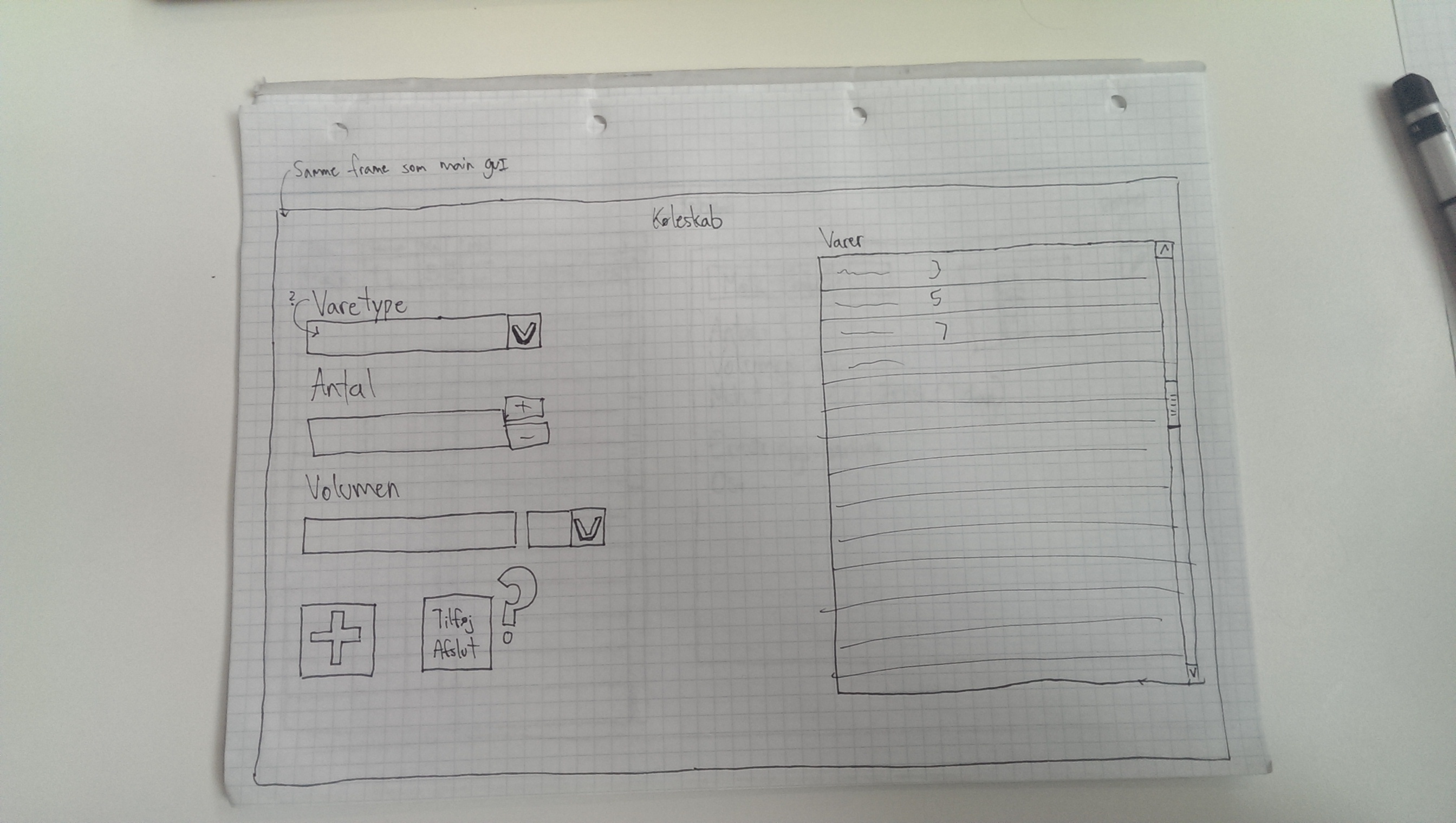
**Figur 3** Skitse af "Se varer"

Siden skitsen, er desuden tilføjet en *Slet*-knap til venstre for hver vare, hvorfra det er muligt at fjerne en vare fra listen. Det skal desuden være muligt inkrementere/dekrementere antallet af vare med én ved tryk på ’+/-’-knapper, som vil befinde sig under knappen *Rediger*.

*Ernæringsværdier* er et eksempel på hvor ekstra information fra eventuelle udvidelser vil befinde sig. Det vil altså ikke være en del af kernefunktionaliteten.

#### Tilføj vare

Designet af menuen for ”Tilføj varer” er blevet skitseret som i **Figur 4**.



**Figur 4** Skitse af "Tilføj varer"

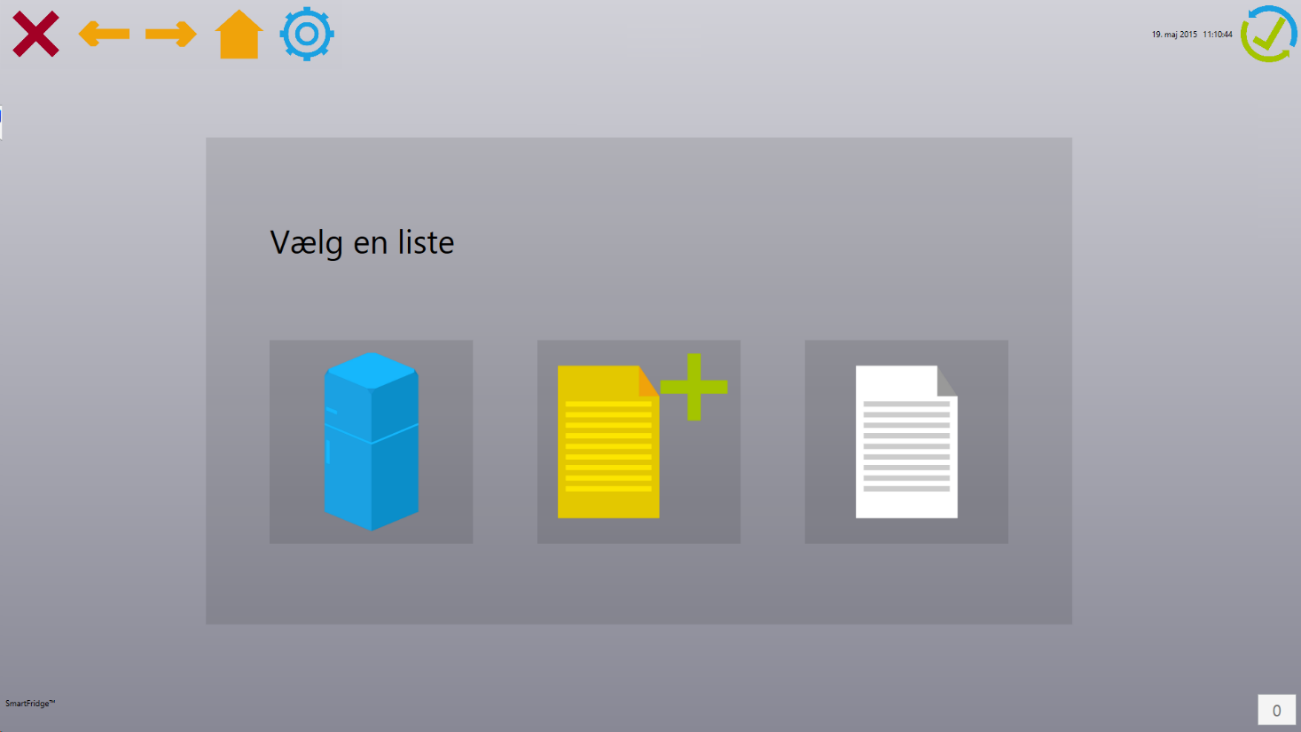
Med implementering af mulighed for at indtaste holdbarhedsdato, er dette felt siden skitsen blevet tilføjet under *Volumen*-feltet.

### Implementering

#### View Layer

##### Control Template

Øverste lag i *View Layer* er hovedvinduet, som består af en ramme omkring en *UserControl*, som det fremgår af **Figur 5**.



**Figur 5** Hovedmenu

|  |
| --- |
| **Figur 6** Klassen CtrlTemplate |

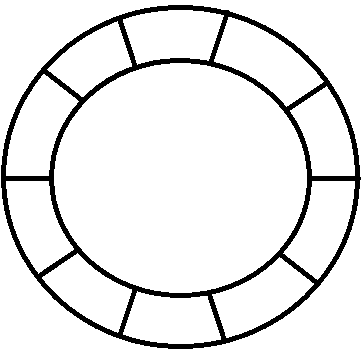
Indholdet i denne *UserControl* styres af *UserControl-*klassen *CtrlTemplate* (**Figur 6**), som ved initialisering indlæser oversigten over tilgængelige lister. På samme tid oprettes en instans af objektet *BLL* (Business Logic Layer), som håndterer al ikke grafisk relateret *code behind*.

Constructoren for enhver af de implementerede *UserControls* tager imod en instans af klassen *CtrlTemplate*. Når et skifte ønskes, kaldes funktionen *ChangeGridContent* i *CtrlTemplate* med den ønskede *UserControl*, som indlæses i stedet for den nuværende, som set i **Kodestump 1**.

|  |
| --- |
| private UserControl \_uc;  public void ChangeGridContent(UserControl uc)  {  \_uc = uc;  CtrlTempGrid.Children.Clear();  CtrlTempGrid.Children.Add(\_uc);  [...]  } |

**Kodestump 1** Skift af UserControl i CtrlTemplate

*CtrlTemplate* lagrer desuden de foregående *UserControls*, som programmet har vist, i en cirkulær buffer (**Figur 7**). Dette muliggør brugen af *Frem*- og *Tilbage*-knapperne, som ses øverst til venstre på **Figur 5**.



[0]

[1]

[2]

[3]

[4]

[5]

[6]

[7]

[8]

[9]

**Figur 7** Cirkulær buffer

Den cirkulære buffer initieres som i **Kodestump 2**. Bemærk at attributternes navne er forkortet her af pladshensyn. For den fulde implementering, henvises til **bilag XX**.

|  |
| --- |
| public UserControl[] NavHisCol { get; private set; }  private int NavHisColPos;  private int NavHisColOrgPos;  public CtrlTemplate()  {  [...]  \_uc = new CtrlShowListSelection(this);  [...]  NavHisCol = new UserControl[10];  NavHisCol[0] = \_uc;  NavHisColPos = 0;  NavHisColOrgPos = NavHisColPos;  } |

**Kodestump 2** Initiering af cirkulær buffer

*NavigationHistoryCollection* er selve arrayet, hvori bufferen lagres.

*NavigationHistoryCollectionPosition* holder styr på hvor den nuværende *UserControl* er placeret, mens *NavigationHistoryCollectionOriginalPosition* holder styr på den foregående *UserControl*. Disse benyttes at at sikre at vi kan finde den korrekte *UserControl*, og at der ikke bevæges til *UserControl*, som ikke længere er gyldig.

Efter dette, lagres hver ny *UserControl* på den næste plads i bufferen, så snart den indlæses, og med funktionerne *NavigateBack* og *NavigateForwards*, benyttes *Frem-* og *Tilbage*-knapperne til at navigere i bufferen.

##### Show List Selection

|  |
| --- |
| **Figur 8** Klassen CtrlShowListSelection |

Klassen *CtrlShowListSelection* (**Figur 8**) er den første *UserControl*, som indlæses i *CtrlTemplate*, når sidstnævnte initieres. Den indeholder en oversigt over de tilgængelige lister, og sørger for at oprette den rette *UserControl*, alt efter brugerens valg.

Constructoren for *CtrlShowListSelection* tager imod *CtrlTemplate*-klassen, som har kaldt den, således at den kan benytte *ChangeGridContent*-funktionen til at skifte *UserControl*, når brugeren har trykket på en af knapperne, som set i **Figur 9**.

|  |
| --- |
| private CtrlTemplate \_ctrlTemp;  private void BtnInFridge\_Click(object sender, RoutedEventArgs e)  {  \_ctrlTemp.ChangeGridContent(new CtrlItemList("Køleskab", \_ctrlTemp));  } |

**Figur 9** Skift af UserControl fra CtrlShowListSelection

Samme fremgangsmåde benyttes ved tryk på de andre lister.

Bemærk at listen identificeres på baggrund af en *string*. Denne umiddelbart hårde kodning er foretaget, med øje for muligheden for at udvide systemet til at lade brugeren oprette nye lister, som hver især kan identificeres på baggrund af deres navne. Alternativt kunne listerne oprettes som objekter, men da de alligevel ikke skulle indeholde andet end et navn, blev det valgt at en *string* var tilstrækkelig.

##### Item List

**Figur 10** viser den grafiske repræsentation af *CtrlItemList*, uden den omkringliggende ramme.



**Figur 10** Listen ”Køleskab”

|  |
| --- |
| **Figur 11** Klassen CtrlShowListSelection |

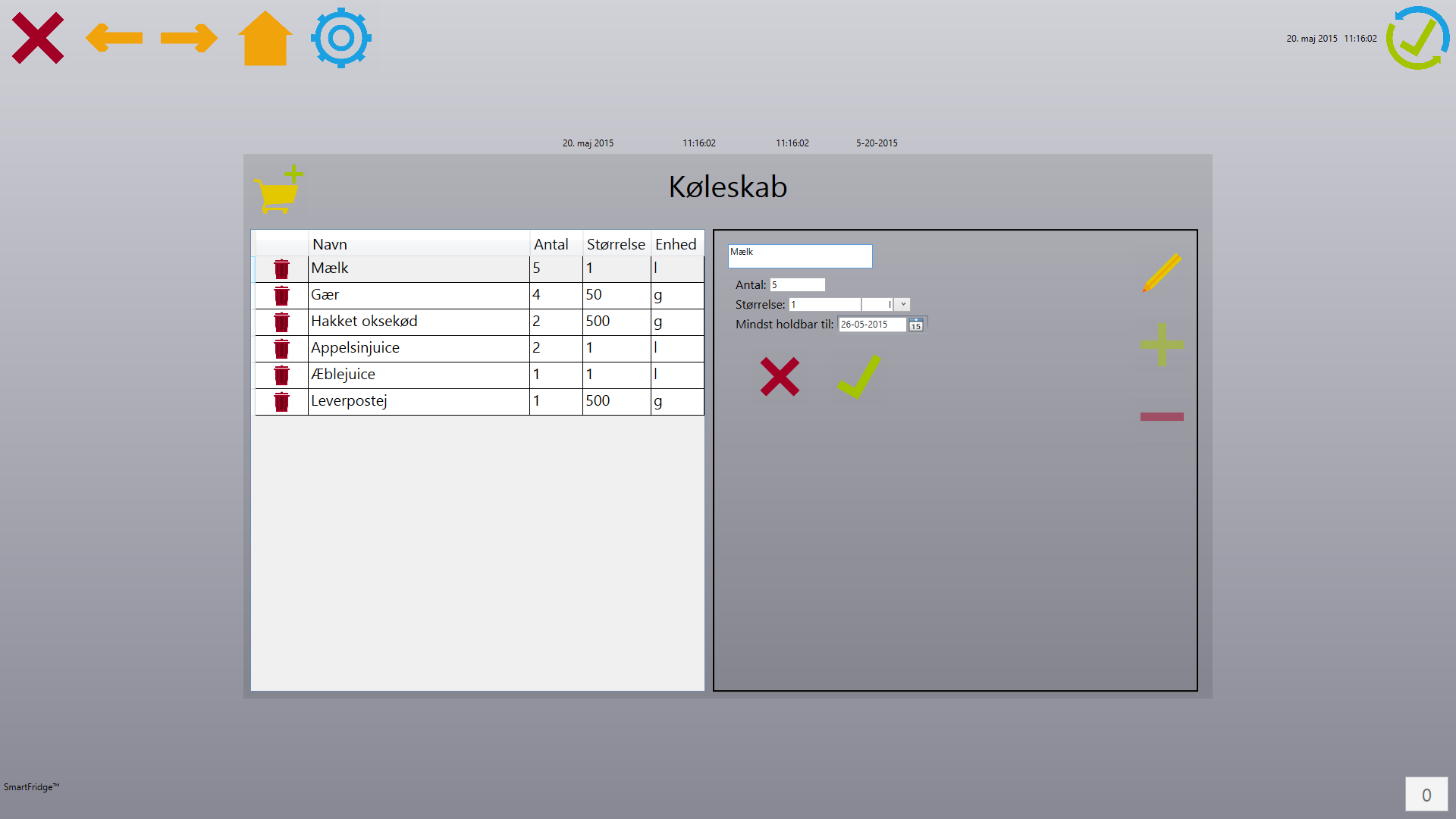
Denne klasse (**Figur 11**) indeholder en del mere funktionalitet end de foregående, og gør derfor i stor stil brug af *Business Logic Layer’*et (*BLL*), som håndterer den bagvedliggende logik, mens *CtrlItemList* selv håndterer den grafiske repræsentation.

*CtrlItemList* viser indholdet af den valgte liste, baseret på den medsendte *string*, som kædes sammen med listen af samme navn i databasen.

Når CtrlItemList oprettes, indlæses alle data (*LoadItemData*) fra den pågældende database i et *DataGrid*, og et klik på en række (*DataGridItems\_SelectionChanged*) udløser nærmere informationer om den enkelte vare, som fremkommer i informationsvinduet i højre side af vinduet.

Til venstre for hver vare, repræsenterer den røde skraldespand muligheden for at slette en vare (*BtnDelete\_Click*).

I informationsvinduet repræsenterer den gule blyant muligheden for at redigere den valgte vare (*BtnEdit\_Click*), hvilket resulterer i at vareinformationerne omdannes til redigerbare tekstblokke, som illustreret i **Figur 12**. Efter redigeringen, er det muligt at bekræfte (*BtnAccept*) ved at trykke på det grønne flueben, hvorved ændringerne persisteres, eller annullere (*BtnCancel\_Click*) ved at trykke på det røde kryds, hvorved ændringerne ignoreres.



**Figur 12** Rediger vare

Ved klik på det grønne plus er det muligt at øge (*BtnInc\_Click*) eller formindske (*BtnDec\_Click*) mængden af den valgte vare med én, uanset om der er trykket på *Rediger*. Disse funktioner syntes relevante at gøre let tilgængelige, for at gøre løbende forbrug, og registrering af ofte brugte varer, hurtigere at registrere.

Øverst til venstre i *CtrlItemList* ses en gul indkøbsvogn med et grønt plus. Denne knap tillader brugeren at tilføje nye varer (*BtnAddItem\_Click*), og fører til den sidste *UserControl*-klasse, *Additem*.

##### Add Item



#### Business Logic Layer

### Test

## Web app

### Design

### Implementering

### Test

## Database

### Design

### Implementering

### Test