# Design og implementering

**Design:**

Her beskrives det mere detaljerede design for hhv. HW og SW.

**Implementering:**

Her beskrives implementeringen i detaljer (kredsløbsberegninger, klassediagrammer, sekvensdiagrammer, state-diagrammer, aktivitetsdiagrammer. sourcekode udsnit m.m.).

**Test:**

Her beskrives de for projektet gennemførte enhedstests. Integrationstests og accepttest kommer i de følgende afsnit.

Evt. testprogrammer vedlægges på CD-ROM.

## 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.

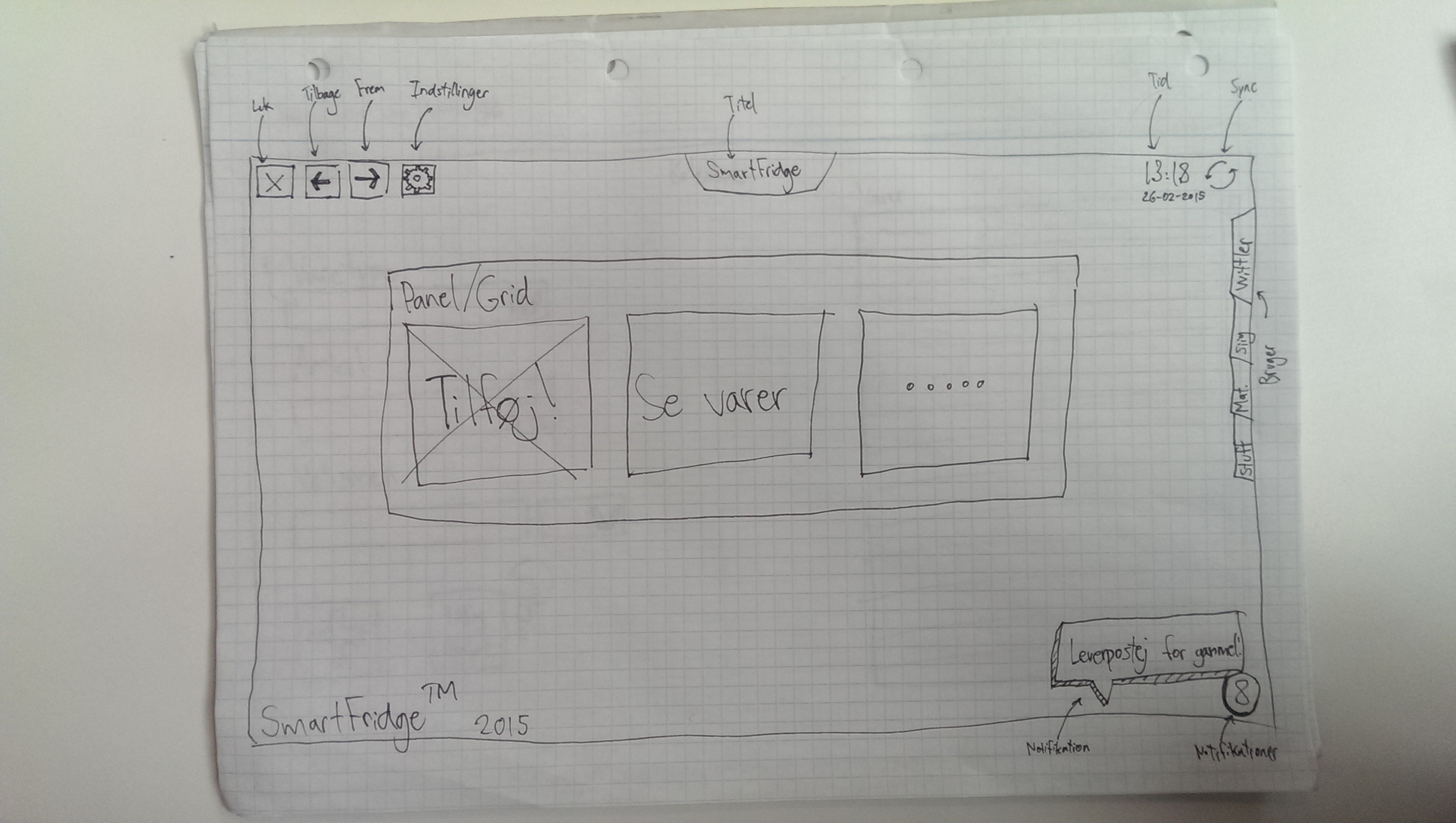
## Fridge app

### 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**.



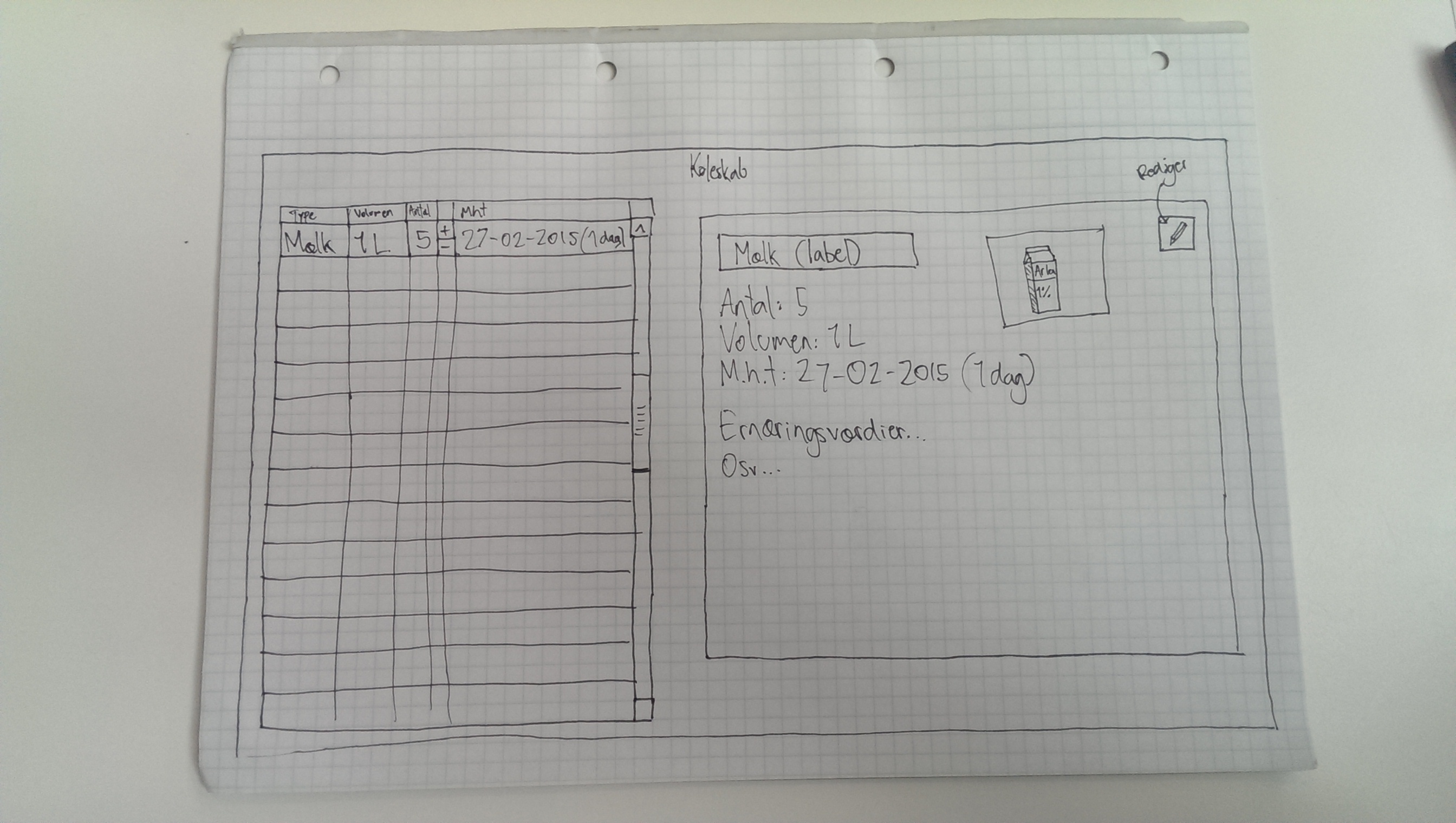
**Figur 2** Skitse af hovedmenu og ramme

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**.



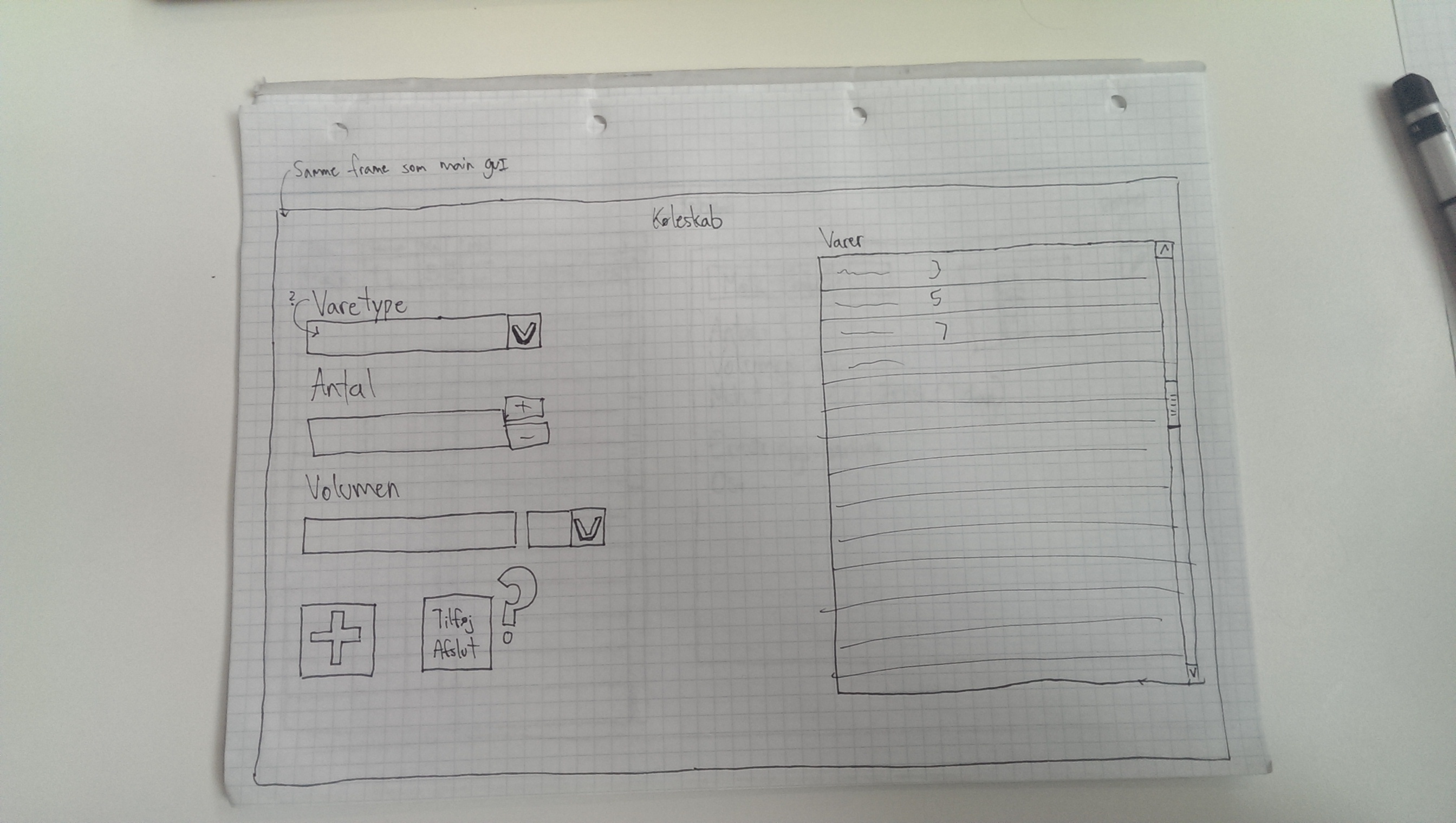
**Figur 3** Skitse af "Se varer"

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"

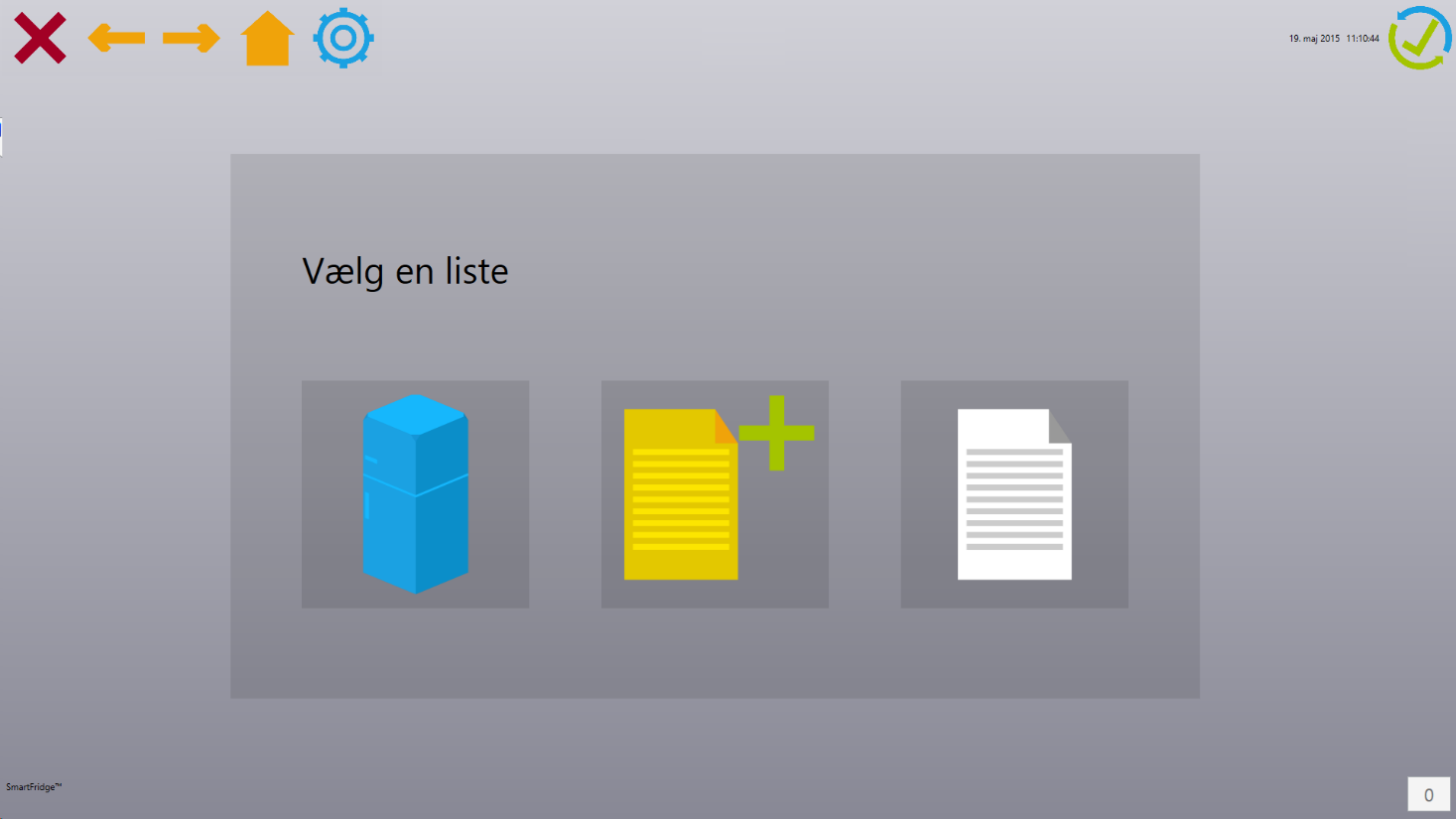
Med implementering af mulighed for at indtaste holdbarhedsdato, er dette felt siden skitsen blevet tilføjet under *Volumen*-feltet.

### 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

Indholdet i denne *UserControl* styres af *UserControl-*klassen *CtrlTemplate*, 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 og , hvor der skiftes fra listeoversigten til listen ”Køleskab”.

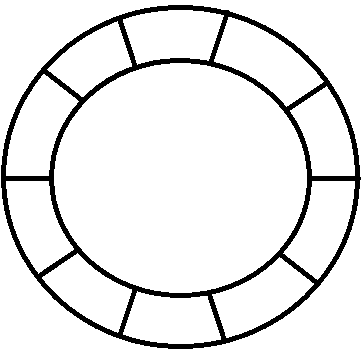
|  |
| --- |
| private CtrlTemplate \_ctrlTemp;  private void BtnInFridge\_Click(object sender, RoutedEventArgs e)  {  \_ctrlTemp.ChangeGridContent(new CtrlItemList("Køleskab", \_ctrlTemp));  } |

**Kodestump 1** Skift af UserControl fra listeoversigten

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

**Kodestump 2** Skift af UserControl i CtrlTemplate

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



[0]

[1]

[2]

[3]

[4]

[5]

[6]

[7]

[8]

[9]

**Figur 6** Cirkulær buffer

Den cirkulære buffer initieres som i . 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 3** 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.

#### Business Logic Layer

### Test

## Web app

### Design

### Implementering

### Test

## Database

### Design

### Implementering

### Test