## GUI design overvejelser

Under designet af GUI’en dukkede et problem op i form af det er svært at unitteste den, og derfor kan kodens funktionalitet ikke verificeres. Men der findes en løsning, i form af et MVVM[[1]](#footnote-1) pattern.

Ved brug af MVVM bliver bindingen mellem GUI’en og buisness logikken, løsnet idet at viewet primært er defineret i XAML filen, med en begrænset code-behind. Ved at binde en ICommand til view modellen, kan der kaldes funktioner fra viewet nede i view modellen og ved databindings kan der deles data.



Figur 1 - MVVM model[[2]](#footnote-2)

For eksempel, i Pristjek220 kan en forretningsmanager tilføje en ny vare til sin forretning. Når varens navn og pris, er indtastet og der bliver klikket på ’Tilføj vare’, tager view modellen navnet og prisen og forsøger at sende dette videre til modellen. Hvis dette er succesfuldt, bliver varen gemt i databasen og en bekræftelses tekst bliver sat, som et label i GUI’en er bindet til, og derved kan forretningsmanageren se at hans vare tilføjelse var en succes.

Fordi at MVVM laver en opdeling af GUI’en i et view og en viewmodel, er det muligt for teamet at arbejde på begge ting, uden at skulle frygte at ødelægge hinandens kode. Ved at indsætte en driver imellem viewet og viewmodellen, bliver det muligt at unit teste GUI’en.

En anden fordel ved MVVM er, at hvis der i Pristjek220 ønskes at den generede indkøbsliste skal vise prisen i både kroner og euro, så er view modellen oplagt til dette. En view model’s primære opgave er at tage dataen fra modellen og formatere det således at det tilpasser viewet’s ønske. Så i stedet for at tilføje prisen i euro helt nede i databasen, bliver prisen bare konverteret til euro i view modellen og derfra bindet til en ny en kolonne i den generede indkøbsliste.

1. https://msdn.microsoft.com/en-us/library/hh848246.aspx?f=255&MSPPError=-2147217396 [↑](#footnote-ref-1)
2. https://msdn.microsoft.com/en-us/library/hh848246.aspx?f=255&MSPPError=-2147217396 [↑](#footnote-ref-2)