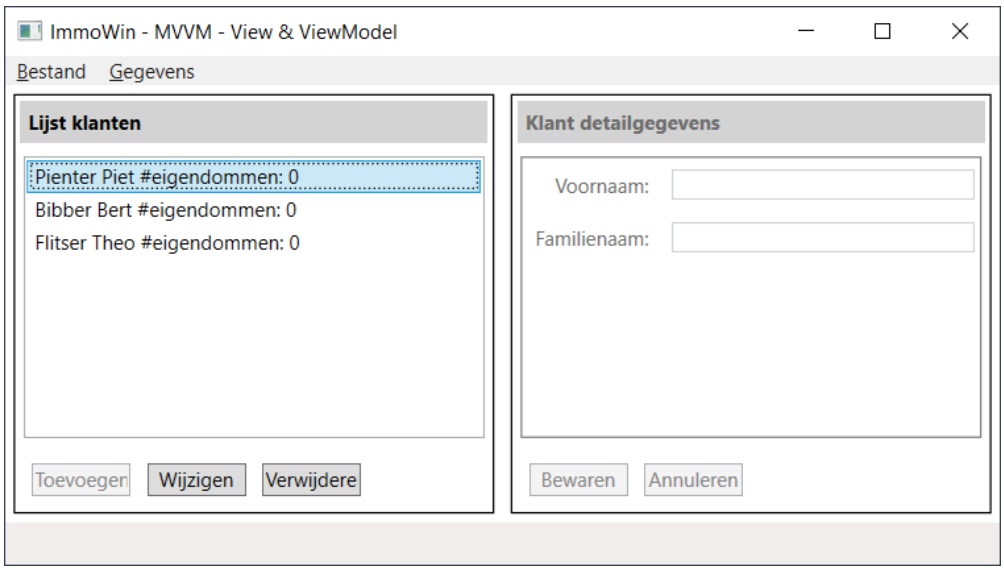
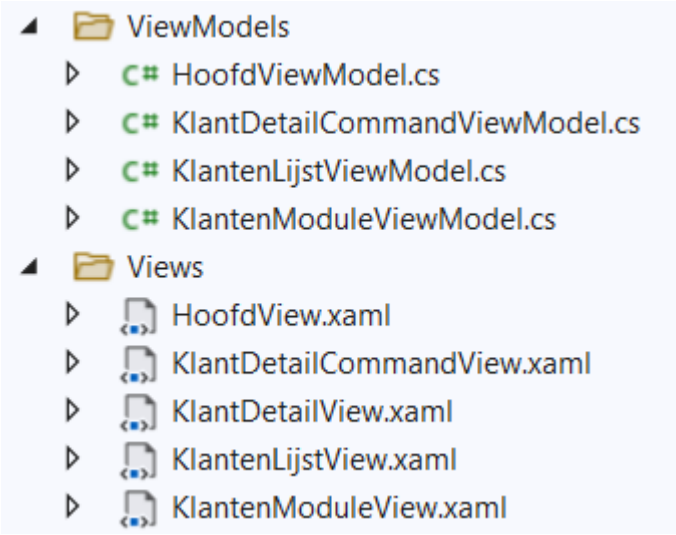
Doel

De bedoeling is om dit scherm te bouwen:



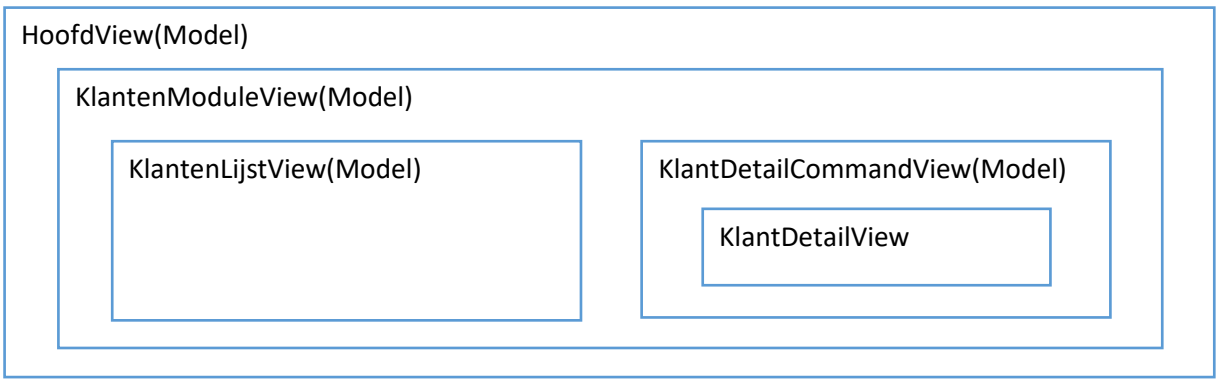
Mogelijke aanpak - Folders en files

Folders en files in de WpfApp:



Mogelijke aanpak - Structuur

Met de voorgestelde Views en ViewModels kunnen we volgende structuur bouwen:



HoofdView is een Window. In App.xaml wordt deze view aangeduid als opstartscherm:

```
StartupUri="Views/HoofdView.xaml"
```

Het HoofdViewModel wordt in de xaml-code van HoofdView aangeduid als datacontext:

```
<Window.DataContext>
        <viewmodels:HoofdViewModel></viewmodels:HoofdViewModel>
</Window.DataContext>
```

In principe worden alle andere Views (=UserControls) getoond in een ContentControl. Via Binding in de Content-property wordt deze gekoppeld aan een property in het ViewModel.

Voorbeelden

HoofdView heeft een ContentControl waarvan de Content via binding gekoppeld is aan property *HuidigeModuleViewModel* van *HoofdViewModel*. Deze property wordt – als in het menu voor 'klanten module' gekozen wordt – opgevuld met een instantie van *KlantenModuleViewModel*. Via een datatemplate in App.xaml wordt *KlantenModuleView* getoond.

KlantenModuleView heeft twee ContenControls waarvan de Content gekoppeld is aan de respectievelijke properties *HuidigeKlantenLijstViewModel* en *HuidigeKlantDetailViewModel* van *KlantenModuleViewModel*. Deze properties worden in de constructor van *KlantenModuleViewModel* opgevuld met instanties van *KlantenLijstViewModel* en *KlantDetailCommandViewModel*. Via datatemplates in App.xaml worden hierdoor respectievelijk *KlantenLijstView* en *KlantenDetailCommandView* getoond.

Opmerking

Voor de klantenlijst (*KlantenListView*) werd geopteerd zowel het presenteren van gegevens (via ListBox) als het implementeren van functionaliteiten (via Buttons) te integreren in één View.

Voor de detailgegevens van een klant werd dit opgesplitst in twee views: *KlantDetailCommandView* voor de functionaliteiten (via Buttons) met daarin *KlantDetailView* voor de gegevens (via TextBoxen). Bijzonder hieraan is dat *KlantDetailView* als control wordt opgenomen in de xaml-code van *KlantDetailCommandView*. Het gevolg hiervan is dat *KlantDetailView* (ook) *KlantDetailCommandViewModel* als datacontext krijgt. *KlantDetailView* heeft (op dit moment) geen eigen ViewModel.