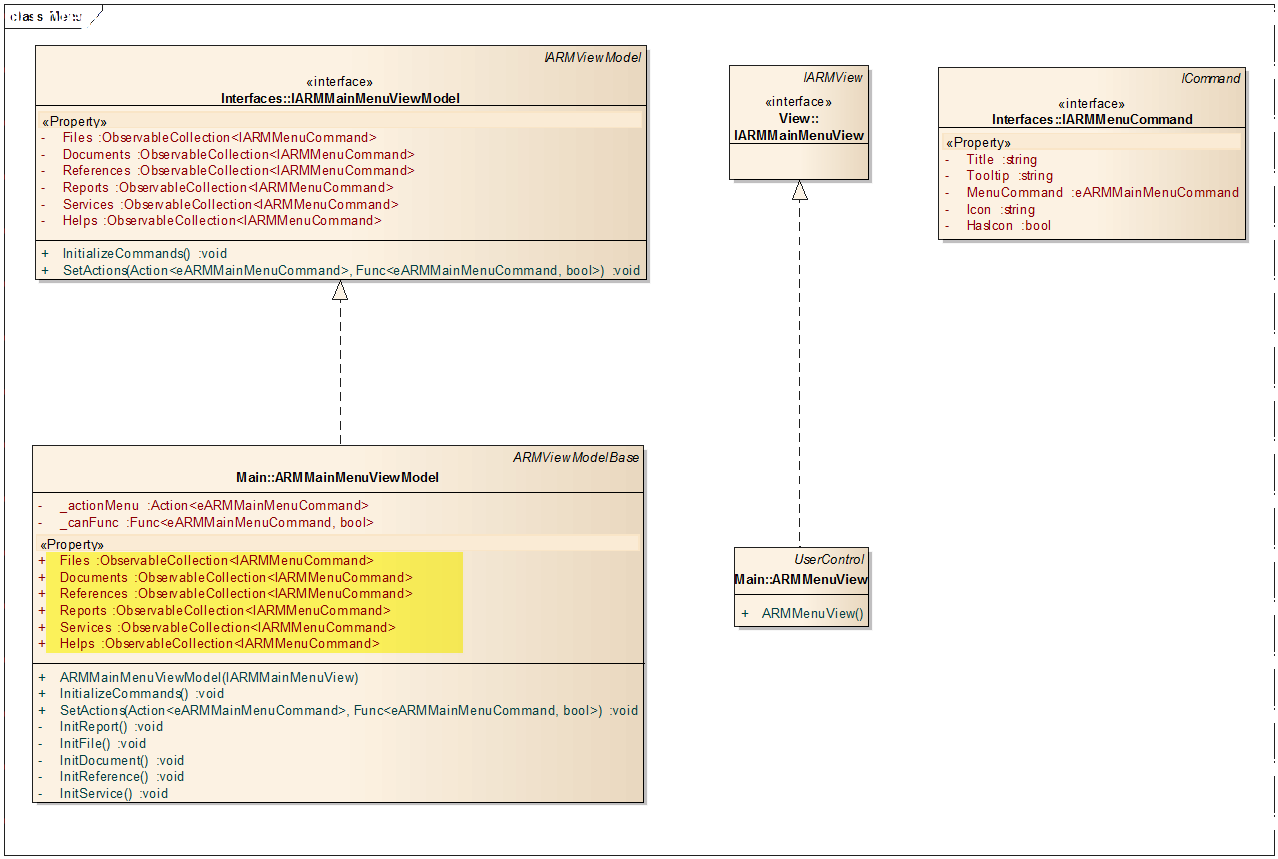
**Розробка ресурсів головного меню.**

На малюнку нижче представлені класи, які формують та забезпечують функціонування головного меню програми.



Клас ARMMainMenuViewModel представляє собою модель представлення головного меню. Цей клас містить шість колекцій команд. Дані команди прив’язуються до пунктів меню графічного меню.

Нижче приведене графічне представлення головного меню в програмі.



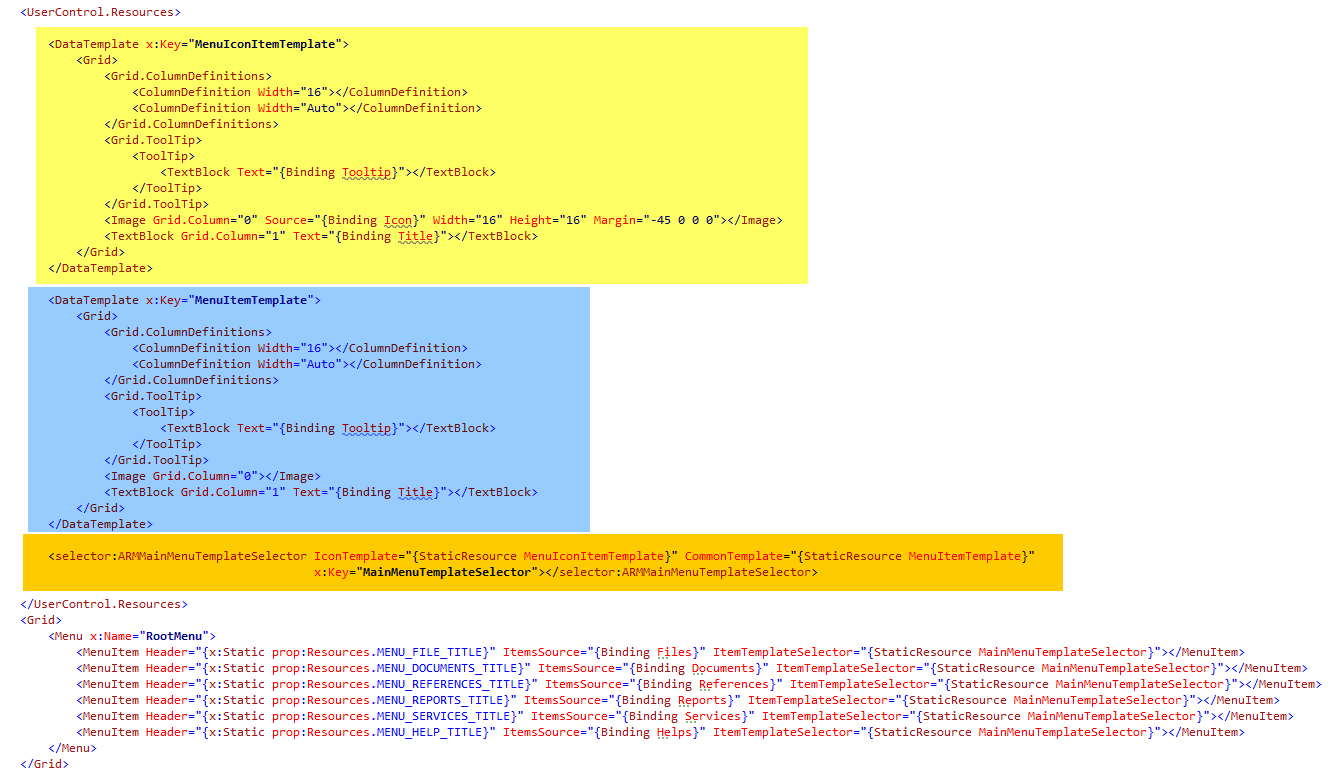
Нижче перераховані колекції команд:

1. ObservableCollection<IARMMenuCommand> Files { get; } – дана колекція містить команди, які відображаються в меню «Файл».
2. ObservableCollection<IARMMenuCommand> Documents { get; } – дана колекція містить команди, які відображаються в меню «Документи».
3. ObservableCollection<IARMMenuCommand> References { get; } – дана колекція містить команди, які відображаються в меню «Довідники».
4. ObservableCollection<IARMMenuCommand> Reports { get; } – дана колекція містить команди, які відображаються в меню «Звіти».
5. ObservableCollection<IARMMenuCommand> Services { get; } – дана колекція містить команди, які відображаються в меню «Сервіси»
6. ObservableCollection<IARMMenuCommand> Helps { get; } – дана колекція містить команди, які відображаються в меню «Допомога».

Графічне меню представлене інтерфейсом IARMMainMenuView та класом ARMMenuView.

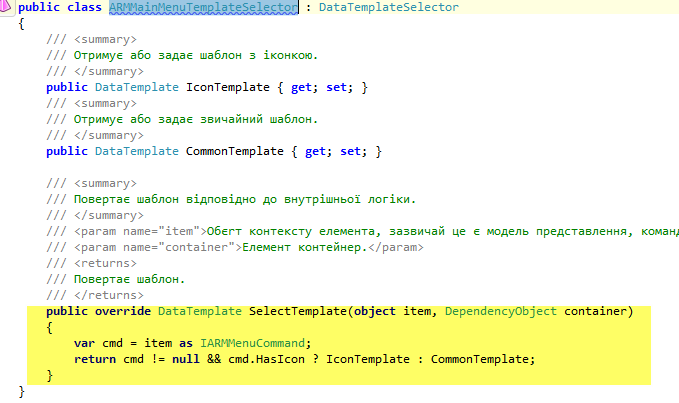
ARMMenuView також наслідується від класу UserControl.

Нижче приведено код класу ARMMenuView:



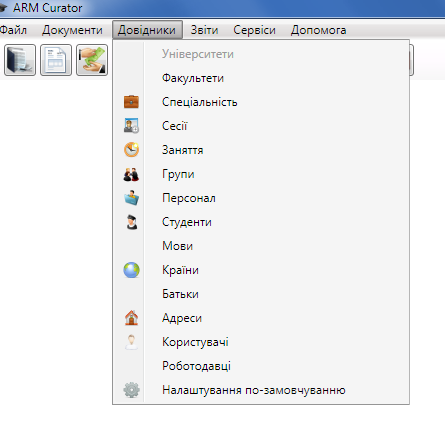
Головне меню складається з елемента з ім’ям «Root». Даний елемент містить 6 елементів MenuItem, які своїми властивостями «ItemsSource» прив’язані до відповідних колекцій модела представлення меню. Пункти меню можуть бути двох видів: з іконкою, так і просте текстовий запис. Ці два типи будуються на основі різних шаблонів. В ресурсах графічного представлення знаходиться два шаблони: жовтим виділено шаблон для пунктів з іконками, синім виділено шаблон, який призначено виключно для текстових пунктів меню. Також в ресурсах ми можемо бачити ще один об’єкт типу ARMMainMenuTemplateSelector. Даний клас призначений для вибору шаблону для певного пункту меню у відповідності до внутрішньої логіки селектора. Даний об’єкт прив’язнаний до кожного пункту меню через властивість «ItemTemplateSelector».

Нижче приведено код класу ARMMainMenuTemplateSelector:



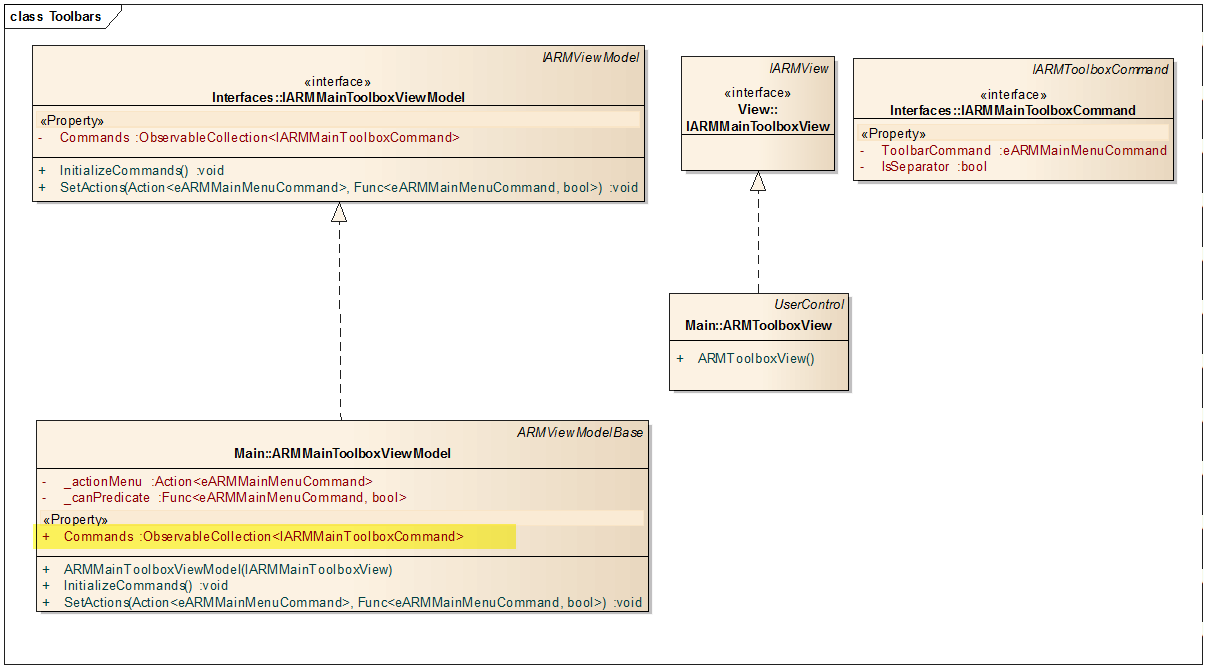
Даний клас має перевизначений метод «SelectTemplate» котрий виділений жовтим кольором. В даному методі програма визначає: якщо команда має іконку то повертає шаблон «IconTemplate», якщо ж іконка відсутня то повертає звичайний шаблон для створення пункту меню.

В результаті роботи «binding»-а та селектору шаблонів ми отримуємо такий результат:



**Розробка ресурсів головної панелі управління.**

На малюнку нижче представлені класи що формують та забезпечують функціонування головної панелі управління.



Клас ARMMainToolboxViewModel представляє собою модель представлення панелі управління. Він містить властивість «Commands», яка є колекцією команд, які в подальшому будуть прив’язуватись до кнопок панелі управління.

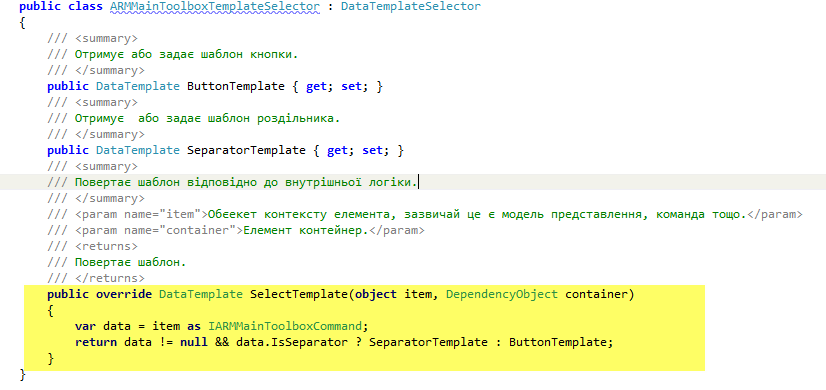
Нижче представлене графічне представлення в програмі:



Графічне представлення описується класом ARMToolboxView. Нижче наведений код представлення:

З коду ми бачимо, що колекція команд прив’язана до властивості «ItemsSource» елемента “Toolbar”. Для формування кожного елемента панелі управління використовуються два шалони: «ButtonToolboxTemplate», «SeparatorToolboxTemplate» та клас-селектор ARMMainToolboxTemplateSelector.

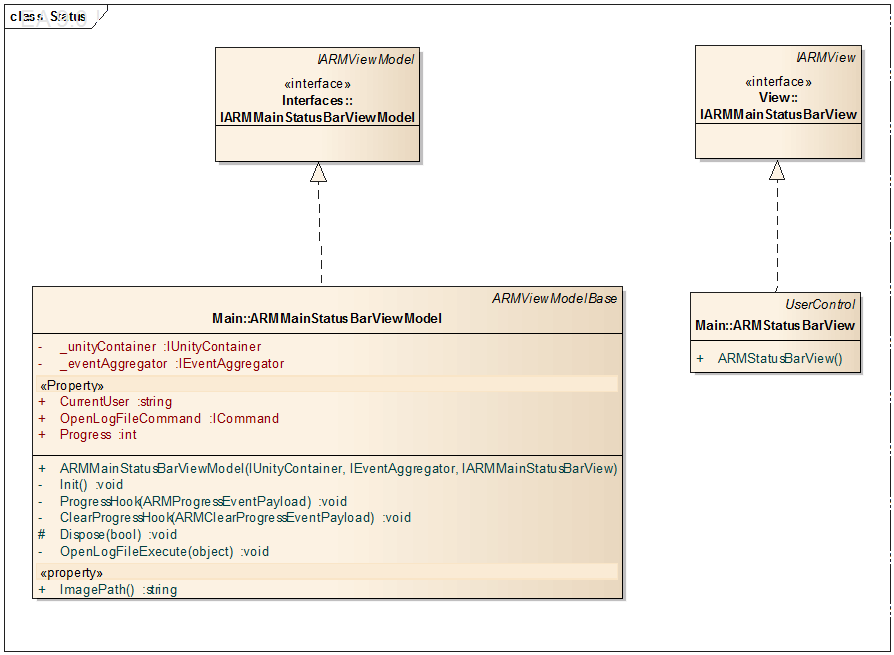
Нижче приведений код класу ARMMainToolboxTemplateSelector.



В даного класу є дві властивості, до яких повинні бути прив’язані шаблони кнопки панелі управління та роздільника панелі управління. Також в даному класі перевизначений метод «SelectTemplate». В даному методі ми перевіряємо чи команда є роздільником, якщо так – то повертаємо шаблон роздільника. При цьому даний пункт відображається у вигляді вертикальної лінії блакитного кольору. Якщо ж команда не є роздільником, то повертається шаблон, який формує кнопку на панелі управління.

**Розробка ресурсів головної панелі статусу.**

Нижче приведені класи що забезпечують формування та функціонування рядка статусу.



Клас ARMMainStatusBarViewModel є моделлю представлення ряжка статусу. Даний клас містить три властивості, які приєднуються доя користувацьго інтерфейсу:

1. CurrectUser – відображає ім’я поточного користувача, який працює з програмою.
2. OpenLogFileCommand – команда, яка прив’язується до кнопки для в відкриття поточного файлу легування.
3. Progress – прив’язана до елементу управління «ProgressBar».

Нижче показане графічне відображення статус рядка в програмі:



Нижче приведений код графічного представлення рядка:



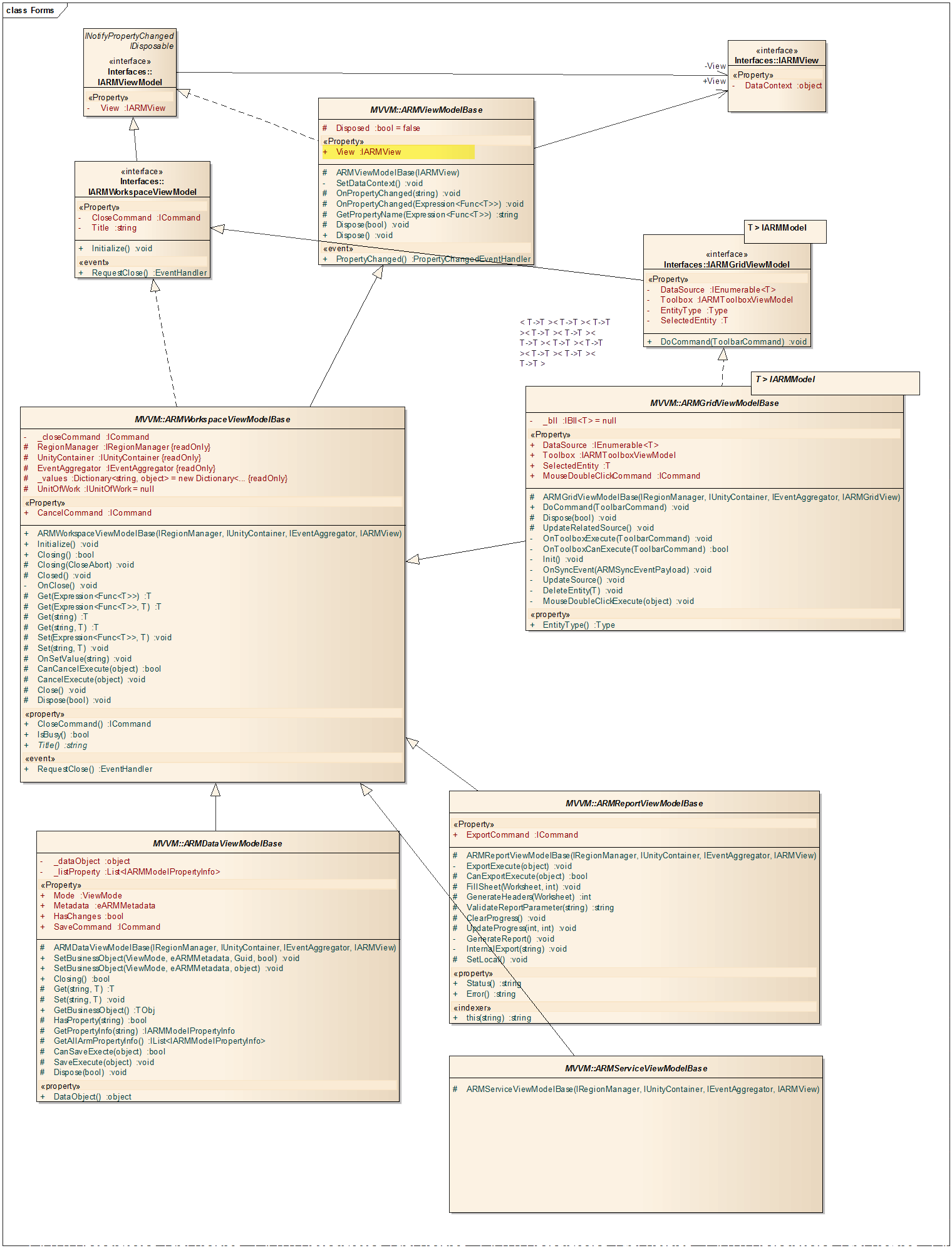
Блок 1 відображає формування представлення поточного користувача на формі. Властивість «CurrentUser» прив’язана до властивості «Text» елемента TextBlock.

В блоці 2 розміщений елемента управління ProgressBar. Даний елемент призначений для відображення прогресу під час експорту даних в Excel.

В блоці 3 представлене створення кнопки, яка призначена для відкриття поточного файл легування.

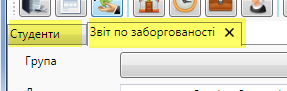
**Розробка шаблонів ресурсів форм.**

Нижче приведене діаграма класів, які беруть участь у створенні головного інтерфейсу програми.



Клас ARMViewModelBase є базовим класом для всіх моделей представлення програми. Даний клас має одну з найважливіших властивостей – «View». Дана властивість зберігає форму представлення даних користувачу.

Клас ARMWorkspaceViewModelBase – відповідає за функціональність закладок в головному вікні програми.



Клас ARMGridViewModelBase<T> - представляє собою форму сітки для відображення набору даних одного типу. Дана модель відрізняється тільки типом даних, і в той же час використовує єдину форму для відображення сітки даних (незважаючи на тип даних). Тому що в даному проекті реалізований елемент управління, який підлаштовується під тип переданих даних. Він може сам визначити які колонки потрібно формувати, в якому порядку і тд. (клас ARMDataGridControl).

Клас ARMDataViewModelBase – призначение для відображення даних одного об’єкта даних. Кожна така модель має свою унікальну форму користувацького інтерфейсу. Їх можна віднайти в просторі імен - ARM.Module.View.References, ARM.Module.View.Documents. Вони зв’язані зі своїми моделями представлення через відповідні унікальні інтерфейси. Але в основі лежить єдиний шаблон, про вибір якого буде розказано нижче.

Таку саму поведінку мають класи успадковані від ARMReportViewModelBase та ARMServiceViewModelBase. Кожен наслідник має свою форму, але в основі лежить єдиний шаблон для кожного класу.

Дані шаблони описуються в ресурсах класа ARMMainWorkspaceView. Даний клас являє собою робоче поле програми.

Нижче приведено код даного представлення:

<UserControl x:Class="ARM.Module.View.Main.ARMMainWorkspaceView"

             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

             xmlns:xctk="http://schemas.xceed.com/wpf/xaml/toolkit"

             xmlns:xcad="http://schemas.xceed.com/wpf/xaml/avalondock"

             xmlns:armviewmodel="clr-namespace:ARM.Infrastructure.MVVM;assembly=ARM.Infrastructure"

             xmlns:ws="clr-namespace:ARM.Module.Helpers.WorkspaceSelector"

             xmlns:i="clr-namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"

             xmlns:localBeh="clr-namespace:ARM.Infrastructure.Utils;assembly=ARM.Infrastructure">

    <UserControl.Resources>

        <DataTemplate x:Key="GridModelDataTemplate">

            <ContentControl Content="{Binding View}" />

        </DataTemplate>

        <DataTemplate x:Key="DataModelDataTemplate">

            <Grid>

                <Grid.RowDefinitions>

                    <RowDefinition Height="\*" />

                    <RowDefinition Height="Auto" />

                </Grid.RowDefinitions>

                <ContentControl Content="{Binding View}" Grid.Row="0" />

                <StackPanel Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Right">

                    <Button Command="{Binding SaveCommand}" Width="70" Margin="5">OK</Button>

                    <Button Command="{Binding CancelCommand}" Width="70" Margin="5">Cancel</Button>

                </StackPanel>

            </Grid>

        </DataTemplate>

        <DataTemplate x:Key="ServiceModelDataTemplate">

            <Grid>

                <Grid.RowDefinitions>

                    <RowDefinition Height="\*" />

                    <RowDefinition Height="Auto" />

                </Grid.RowDefinitions>

                <xctk:BusyIndicator IsBusy="{Binding IsBusy,UpdateSourceTrigger=PropertyChanged}">

                    <ContentControl Content="{Binding View}" Grid.Row="0" />

                </xctk:BusyIndicator>

            </Grid>

        </DataTemplate>

        <DataTemplate x:Key="ReportModelDataTemplate">

            <Grid>

                <Grid.RowDefinitions>

                    <RowDefinition Height="\*" />

                    <RowDefinition Height="Auto" />

                </Grid.RowDefinitions>

                <xctk:BusyIndicator IsBusy="{Binding IsBusy,UpdateSourceTrigger=PropertyChanged}">

                    <ContentControl Content="{Binding View}" Grid.Row="0" />

                </xctk:BusyIndicator>

                <Grid Grid.Row="1">

                    <Grid.ColumnDefinitions>

                        <ColumnDefinition Width="\*"></ColumnDefinition>

                        <ColumnDefinition Width="Auto"></ColumnDefinition>

                        <ColumnDefinition Width="Auto"></ColumnDefinition>

                        <ColumnDefinition Width="Auto"></ColumnDefinition>

                    </Grid.ColumnDefinitions>

                    <StackPanel Grid.Column="1" Orientation="Horizontal" Margin="5">

                        <TextBlock Text="{Binding Status,UpdateSourceTrigger=PropertyChanged}"></TextBlock>

                    </StackPanel>

                    <Button Grid.Column="2" Command="{Binding ExportCommand}" Width="70" Margin="5">Export</Button>

                    <Button Grid.Column="3" Command="{Binding CancelCommand}" Width="70" Margin="5">Cancel</Button>

                </Grid>

            </Grid>

        </DataTemplate>

        <ws:ARMWorkspaceSelector x:Key="ArmWorkspaceSelector"

                                 GridViewModel="{StaticResource GridModelDataTemplate}"

                                 DataViewModel="{StaticResource DataModelDataTemplate}"

                                 ReportViewModel="{StaticResource ReportModelDataTemplate}"

                                 ServiceViewModel="{StaticResource ServiceModelDataTemplate}"/>

    </UserControl.Resources>

    <Grid>

        <xcad:DockingManager DocumentsSource="{Binding Items,UpdateSourceTrigger=PropertyChanged}"

                             LayoutItemTemplateSelector="{StaticResource ArmWorkspaceSelector}"

                             ActiveContent="{Binding CurrentWorkspace,Mode=TwoWay,UpdateSourceTrigger=PropertyChanged}">

            <xcad:DockingManager.DocumentHeaderTemplate>

                <DataTemplate>

                    <TextBlock x:Name="TextTitle" Text="{Binding Content.Title,UpdateSourceTrigger=PropertyChanged}" />

                </DataTemplate>

            </xcad:DockingManager.DocumentHeaderTemplate>

            <xcad:LayoutRoot x:Name="RootDockLayout">

                <xcad:LayoutPanel Orientation="Horizontal">

                    <xcad:LayoutDocumentPaneGroup>

                        <xcad:LayoutDocumentPane />

                    </xcad:LayoutDocumentPaneGroup>

                </xcad:LayoutPanel>

            </xcad:LayoutRoot>

            <i:Interaction.Behaviors>

                <localBeh:ARMEventToCommandBehavior Event="DocumentClosing" Command="{Binding ClosingCommand}"

                                                    PassArguments="True" PassSender="False" />

            </i:Interaction.Behaviors>

        </xcad:DockingManager>

    </Grid>

</UserControl>

В даному коді присутні 4 шаблони:

* GridModelDataTemplate – використовується для відображення сіток даних. Він використовується для всіх класів, які наслідують клас ARMGridViewModelBase<T>.
* DataModelDataTemplate – використовується для класів які наслідують клас ARMDataViewModelBase.
* ReportModelDataTemplate – використовується для класів які наслідують клас ARMReportViewModelBase.
* ServiceModelDataTemplate – використовується для класів як наслідують клас ARMServiceViewModelBase.

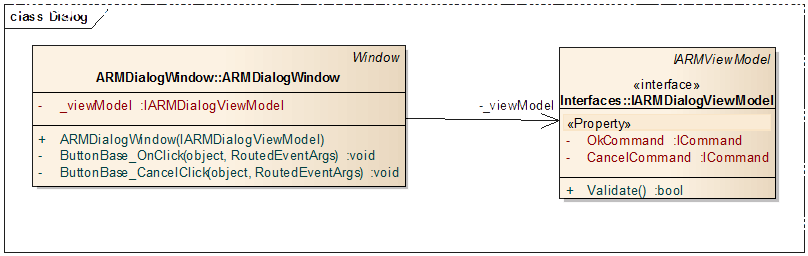
Для вибору відповідного шаблону використовується спеціальний клас ARMWorkspaceSelector. Код якого приведено нижче.

Даний клас має чотири властивості, до яких ззовні приєднуються спеціальні шаблони і він має перевизначений метод «SelectTemplate». Коли створюється нова вкладка, спрацьовує даний селектор. В даному випадку він перевіряє до якого типу належить створена модель і повертає відповідний шаблон відображення. В самих шаблонах ми бачимо що вони представляють загальну модель відображення для певної групи класів. А також вони мають елемент ContentControl. Даний елемент призначений для відображення форми конкретної моделі. З шаблонів ми бачимо що даний елемент використовує дані з властивості «View» (про яку ми згадували раніше, яка є загальною для всіх моделей).



**Розробка ресурсів шаблона діалогового вікна.**

Нижче приведена діаграма класів, які беруть участь у формуванні функціональності діалогових вікон.



Будь яка модель може бути розміщена в діалоговому вікні, при умові що вона реалізовує інтерфейс IARMDialogViewModel.

Нижче приведено код класу ARMDialogWindow.

Графічне представлення:



Код:



З графічного представлення ми бачимо, що діалогове вікно також використовує елемент ContentControl для розміщення форми моделі у вікні. Також воно додає дві кнопки «OK» та «Cancel». При створенні діалогового вінка, в конструктор передається модель. Її форма з властивості “View” поміщається в елемент ContentControl. Після завершення роботи, користувач натискає кнопки «OK» або «Cancel». При цьому закривається діалогове вікно і викликаючи сторона оброблює дані.