# Практические работы №1-3

1. Цель работы
   1. Изучить основные элементы управления Avalonia UI.
   2. Изучить основы применения паттерна MVVM в приложениях Avalonia UI.
2. Литература
   1. Avalonia documentation – Текст : электронный // AvaloniaUI, 2024. – URL: https://docs.avaloniaui.net/
3. Подготовка к работе
   1. Повторить теоретический материал (см.п.2).
   2. Изучить описание лабораторной работы.
4. Основное оборудование
   1. Персональный компьютер.
5. Задание
   1. Создайте новый проект Avalonia
      1. Переименуйте MainView в HomeView
      2. В HomeView добавьте StackPanel и в ней добавьте следующие элементы управления

TextBlock

TextBox

Button

CheckBox

два RadioButton

CalendarDatePicker

ComboBox с двумя вложенными ComboBoxItem

* + 1. Запустите приложение, опробуйте созданные элементы управления
  1. Привязка данных
     1. Создайте класс HomeViewModel, добавьте туда соответствующие поля для значений всех элементов управления
     2. Для полей укажите атрибут [ObservableProperty] для автоматической генерации свойств
     3. Установите HomeViewModel в качестве контекста данных для HomeView (свойство DataContext)
     4. При помощи Binding привяжите значения сгенерированных свойств к элементам управления
  2. Привязка команд
     1. Создайте метод, который будет собирать строку-сообщение из всех данных, указанных при помощи других полей ввода и присваивать это сообщение свойству, привязанному к TextBlock.
     2. Укажите для метода атрибут

[RelayCommand(CanExecute = nameof(метод\_проверки))]

Где метод\_проверки - это второй метод, который возвращает логическое значение, на основе заполненности полей ввода. Если такой метод будет возвращать значение false, конпка, к которой привязана команда автоматически заблокируется.

* + 1. Для свойств полей ввода добавьте атрибут [NotifyCanExecuteChangedFor(nameof(ИмяМетодаCommand))], где ИмяМетодаCommand – сгенерированная команда на основе созданного вами метода. Этот атрибут позволяет оповещать команду об изменении свойств, влияющих на ее возможность выполнения
  1. Валидация свойств
     1. Создайте новую страницу Avalonia (UserControl) – RegistrationView. На странице расположите следующие элементы:

Поле ввода логина

Поле ввода пароля

Поле ввода подтверждения пароля

Поле ввода email

Поле ввода номера телефона

* + 1. Создайте новую ViewModel – RegistrationViewModel. Определите в ней свойства для привязки (используйте полные свойства, а не сгенерированные атрибутом). Установите контекст данных.
    2. Используя атрибуты валидации, установите следующие ограничения:

Логин может содержать только латинские буквы, цифры и нижнее подчеркивание

Пароль должен быть длиной не менее 8 символов, содержать латинские буквы верхнего и нижнего регистра, спецсимволы

Email должен содержать корректный email, а номер телефона - корректный номер.

Все поля должны быть заполнены.

* + 1. В файле App.axaml.cs закомментировать или удалить строку

BindingPlugins.DataValidators.RemoveAt(0);

* + 1. Проверить работу валидаторов полей ввода.
  1. Навигация
     1. Создайте в корне проекта класс со следующим содержимым:

public class ViewLocator : IDataTemplate

{

public bool SupportsRecycling => false;

public Control Build(object data)

{

var name =

data.GetType().FullName

.Replace("ViewModel", "View");

var type = Type.GetType(name);

if (type != null)

{

return (Control)Activator

.CreateInstance(type);

}

else

{

return new TextBlock

{ Text = "Not Found: " + name };

}

}

public bool Match(object data)

{

return data is ViewModelBase;

}

}

* + 1. Добавить в App.axaml следующие строчки:

<Application

xmlns:local=”using:ИмяПроекта”

…

/>

<Application.DataTemplates>

<local:ViewLocator/>

</Application.DataTemplates>

ViewLocator позволяет автоматически создавать объект View для объекта ViewModel в приложении

* + 1. В MainViewModel создайте свойство CurrentPage и присвойте ему значение new HomeViewModel
    2. В MainWindow создайте элемент управления ContentControl и выполните привязку его свойства Content к значению CurrentPage
    3. Создайте в MainWindow две кнопки, создайте и привяжите к ним команды для перехода к определенным окнам приложения.
  1. Отображение списков
     1. Разработать класс User, в котором определены все свойства, определенные на странице регистрации.
     2. Добавить в приложение новую страницу для отображения списка пользователей и ViewModel для нее.
     3. На странице расположите элемент управления ListView, привяжите его ItemsSource к свойству Users (создайте это свойство во ViewModel)
     4. Определите шаблон для элементов списка: отображайте логин, email и кнопку «Удалить»
     5. Определите привязки для элементов управления. Для привязки кнопки «Удалить» необходимо указать более сложный путь привязки через родительский элемент:

{Binding $parent[UserControl].DataContext.Command}

Таким образом будет осуществлена привязка данных к контексту данных страницы.

* 1. Изменение внешнего вида приложения с помощью Material Avalonia
     1. Создать новый проект. Установить в проект nuget-пакет Material.Avalonia
     2. В файле App.axaml добавить пространство имен

xmlns:themes="clr-namespace:Material.Styles.Themes;assembly=Material.Styles"

* + 1. В стили приложения добавить

<themes:MaterialTheme BaseTheme="Light" PrimaryColor="Purple" SecondaryColor="Lime"/>

* + 1. В MainView добавьте StackPanel и в ней добавьте следующие элементы управления

TextBlock

TextBox

Button

CheckBox

два RadioButton

DatePicker

ComboBox с двумя вложенными ComboBoxItem

С некоторыми особенностями и дополнительными возможностями использования элементов вы можете ознакомиться по ссылке:

https://github.com/AvaloniaCommunity/Material.Avalonia/tree/master/Material.Avalonia.Demo/Pages

* + 1. Сравнить внешний вид стандартных элементов управления и стилизованных

1. Порядок выполнения работы
   1. Выполнить все задания из п.5.
   2. Ответить на контрольные вопросы.
2. Содержание отчета
   1. Титульный лист
   2. Цель работы
   3. Описание действий
   4. Вывод