Составил: Садовский Р.В.

Практические работы №1-3

1 Цель работы

1.1 Изучить основные элементы управления Avalonia UI.

1.2 Изучить основы применения паттерна MVVM в приложениях Avalonia UI.

2 Литература

2.1 Avalonia documentation – Текст : электронный // AvaloniaUI, 2024. – URL: https://docs.avaloniaui.net/

3 Подготовка к работе

3.1 Повторить теоретический материал (см.п.2).

3.2 Изучить описание лабораторной работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Создайте новый проект Avalonia

5.1.1 Переименуйте MainView в HomeView

5.1.2 В HomeView добавьте StackPanel и в ней добавьте следующие элементы управления

TextBlock
TextBox
Button
CheckBox
два RadioButton
CalendarDatePicker
ComboBox с двумя вложенными ComboBoxItem

5.1.3 Запустите приложение, опробуйте созданные элементы управления

5.2 Привязка данных

5.2.1 Создайте класс HomeViewModel, добавьте туда соответствующие поля для значений всех элементов управления

5.2.2 Для полей укажите атрибут `[ObservableProperty]` для автоматической генерации свойств

5.2.3 Установите HomeViewModel в качестве контекста данных для HomeView (свойство DataContext)

5.2.4 При помощи Binding привяжите значения сгенерированных свойств к элементам управления

5.3 Привязка команд

5.3.1 Создайте метод, который будет собирать строку-сообщение из всех данных, указанных при помощи других полей ввода и присваивать это

сообщение свойству, привязанному к TextBlock.

5.3.2 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.

5.3.3 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

5.4 Валидация свойств

5.4.1 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

5.4.2 Создайте новую ViewModel – RegistrationViewModel. Определите в ней свойства для привязки. Установите контекст данных.

5.4.3 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.

5.4.4 В файле App.axaml.cs закомментировать или удалить строку

```
BindingPlugins.DataValidators.RemoveAt(0);
```

5.4.5 Проверить работу валидаторов полей ввода.

5.5 Навигация

5.5.1 Создайте в проекте 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;
        }
    }
```

5.5.2 Добавить в App.axaml следующие строчки:

```
<Application.DataTemplates>
    <local:ViewLocator/>
</Application.DataTemplates>
```

ViewLocator позволяет автоматически создавать объект View для объекта ViewModel в приложении

5.5.3 В MainViewModel создайте свойство CurrentPage и присвойте ему значение new HomeViewModel

5.5.4 В MainWindow создайте элемент управления ContentControl и выполните привязку его свойства Content к значению CurrentPage

5.5.5 Создайте в MainWindow две кнопки, создайте и привяжите к ним команды для перехода к определенным окнам приложения.

5.6 Отображение списков

5.6.1 Разработать класс User, в котором определены все свойства, определенные на странице регистрации.

5.6.2 Добавить в приложение новую страницу для отображения списка пользователей и ViewModel для нее.

5.6.3 На странице расположите элемент управления ListView, привяжите его ItemsSource к свойству Users (создайте это свойство во ViewModel)

5.6.4 Определите шаблон для элементов списка: отображайте логин, email и кнопку «Удалить»

5.6.5 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.

5.7 Изменение внешнего вида приложения с помощью Material Avalonia

5.7.1 Создать новый проект. Установить в проект nuget-пакет Material.Avalonia

5.7.2 В файле App.axaml добавить пространство имен

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

5.7.3 В стили приложения добавить

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

5.7.4 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

5.7.5 Сравнить внешний вид стандартных элементов управления и стилизованных

6 Порядок выполнения работы

6.1 Выполнить все задания из п.5.

6.2 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Описание действий

7.4 Вывод