Pedro Inácio Rodrigues Pontes

Prática 3, Conversor de Unidades de Medida

Belo Horizonte, Brasil

2025

1 Introdução

O objetivo da presente prática foi produzir um conversor de unidades de medida em C# utilizando da interface gráfica Avalonia UI. Aqui estão os requerimentos para a interface:

1. Permitir a seleção do tipo de conversão através de um ListBox
2. Ter um campo de entrada para o valor de origem
3. Ter um campo de saída para o valor convertido
4. Incluir um botão para executar a conversão
5. Tipos de Conversão e Fórmulas

2 Desenvolvimento

Para conseguir alcançar o objetivo da confecção do conversor de unidades de medida, o projeto foi dividido utilizando o padrão MVVM (Model-View-ViewModel). View define a interface do usuário, onde é utilizado o código .xaml do avalonia, Model define as estruturas de dados da aplicação, ViewModel conecta os dois.

2.1 Model

Para a seção model, foi criada a classe ConversionType, a qual define a estrutura básica de uma conversão, que tem nome e sua fórmula, que foi representada utilizando uma *Delgate Property*, a qual consiste em uma propriedade de classe que representa uma função que recebe um valor de parâmetro e devolve outro.

```
namespace ConversorDeUnidadesDeMedida.Models
{
    public class ConversionType
    {
        public string Name { get; set; }
        public Func<double, double> ConversionFormula { get; set; }

        public ConversionType(string name, Func<double, double> conversionFormula)
        {
            Name = name;
            ConversionFormula = conversionFormula;
        }
    }
}
```

2.2 Services

Além da divisão base do modelo MVVM, foi adicionado também o modelo Services, onde se encontra o ConversionService, o qual retorna uma lista com todos os tipos de conversão:

```
namespace ConversorDeUnidadesDeMedida.Services
{
    public static class ConversionService
    {
        public static List<ConversionType> GetConversionTypes()
        {
            return new List<ConversionType>
            {
                new ConversionType("Celsius para Fahrenheit", c => (c * 1.8) + 32),
                new ConversionType("Fahrenheit para Celsius", f => (f - 32) / 1.8),
                new ConversionType("Celsius para Kelvin", c => c + 273.15),
                new ConversionType("Kelvin para Celsius", k => k - 273.15),
                new ConversionType("Metros para Pés", m => m * 3.28084),
                new ConversionType("Pés para Metros", ft => ft * 0.3048),
                new ConversionType("Quilômetros para Milhas", km => km * 0.621371),
                new ConversionType("Milhas para Quilômetros", mi => mi * 1.60934),
                new ConversionType("Quilogramas para Libras", kg => kg * 2.20462),
                new ConversionType("Libras para Quilogramas", lb => lb * 0.453592),
                new ConversionType("Gramas para Onças", g => g * 0.035274),
                new ConversionType("Onças para Gramas", oz => oz * 28.3495),
                new ConversionType("Litros para Galões", l => l * 0.264172),
                new ConversionType("Galões para Litros", gal => gal * 3.78541),
                new ConversionType("Mililitros para Onças Fluidas", ml => ml * 0.03
                new ConversionType("Onças Fluidas para Mililitros", flOz => flOz *
            };
        }
    }
}
```

2.3 View

Para o modelo View, representado principalmente por MainWindow.axaml, foram utilizados os tipos ListBox para fazer a lista de conversões, TextBox para receber e exibir os valores e Button para criar um botão para conversão, além do TextBloc para representar um bloco comum de texto. Segue a parte relevante do código:

```
<ListBox ItemsSource="{Binding ConversionTypes}" SelectedItem="{Binding SelectedCon
         Grid.Row="0" Grid.Column="1" Margin="10" Height="100">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <TextBlock Text="{Binding Name}" />
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
[...]
<TextBox Text="{Binding InputValue}" Grid.Row="1" Grid.Column="1" Margin="10"/>
[...]
<TextBox Text="{Binding OutputValue}" Grid.Row="2"
Grid.Column="1" Margin="10" IsReadOnly="True"/>
[...]
<Button Content="Converter" Command="{Binding ConvertCommand}" Grid.Row="3"
Grid.Column="1" Margin="10" HorizontalAlignment="Right"/>
```

Observa-se principalmente a Palavra *Binding* dentro de chaves. Isso ocorre para chamar elementos da divisão ViewModel, que serão conectados automaticamente a elas, como ConvertCommand e Name. Essa funcionalidade do Avalonia é extremamente útil para separar as lógicas de View e backend.

2.4 ViewModel

3 Resultados

Todos os resultados foram alcançados perfeitamente, com a única ressalva de aparentemente o método Dump permitir a exibição de máximas 16 linhas de toda a seleção.

4 Conclusão

Todos os objetivos foram alcançados com sucesso. Os maiores problemas foram o método Dump não exibir mais de 16 linhas e a necessidade do uso do simbólo ? após alguns valores para permitir valores *null*, e ?? para manipular um valor que será utilizado quando algum for *null*. Outra observação é que o uso de LINQ lembra enormemente ao SQL.

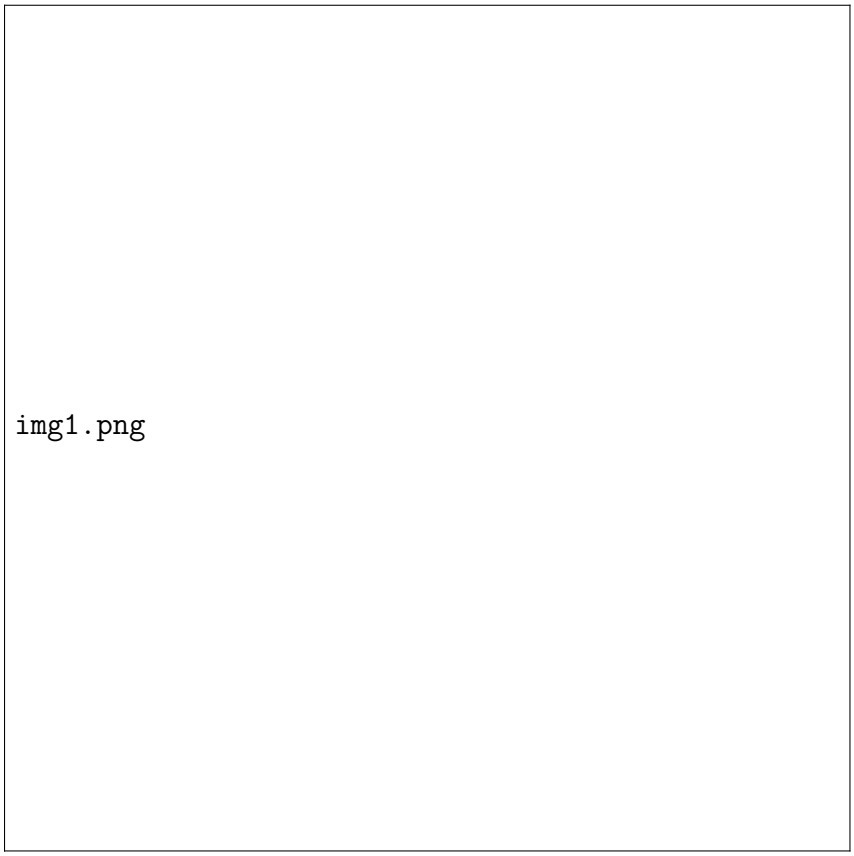

Figura 1 – Resultado do Exércicio 10