Guia de Introdução do WPF

Este laboratório é uma introdução ao desenvolvimento de um aplicativo Windows Presentation Foundation (WPF), que inclui os elementos que são comuns à maioria dos aplicativos WPF: marcação Extensible Application Markup Language (XAML), code-behind, controles, layout, ligação de dados e estilos.

Este laboratório orienta o desenvolvimento de um aplicativo simples WPF usando as seguintes etapas.

- Usar XAML para projetar a interface do usuário (UI).
- Escrever código para gerenciar as ações.
- Adicionar controles ao layout para definir a UI do aplicativo.
- Criar estilos para definir uma aparência consistente em toda interface de usuário do aplicativo.
- Vincular a interface de usuário com os dados e manter os mesmos sincronizados.

No final deste laboratório, você terá criado um aplicativo Windows que permite aos usuários exibir relatórios de despesas para pessoas selecionadas. A aplicação será composta por varias páginas WPF que são hospedadas em uma janela estilo browser.

Criando o projeto de aplicativo

---

1. Abra Application C# e, crie uma aplicação WPF.

Esse arquivo XAML define um WPF aplicativo e os recursos de aplicativo. Você também usa este arquivo para especificar a UI que automaticamente é exibida quando o aplicativo é iniciado; Nesse caso, MainWindow. XAML.

O XAML deve esta aparência no C#

XAML

```
<Application x:Class="Lab01WPF.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="MainWindow.xaml">
    <Application.Resources>

    </Application.Resources>
</Application>
```

2. Abra MainWindow. XAML.

Este arquivo XAML é a janela principal do seu aplicativo e exibe o conteúdo criado na página. A classe Window define as propriedades de uma janela, como título, tamanho ou ícone e trata os eventos, como o fechar ou ocultar.

3. Alterar o Window elemento para um NavigationWindow.

Para navegar para conteúdos diferentes , dependendo da interação do usuário, a janela principal" **Window**" precisa ser alterado para um **NavigationWindow**. NavigationWindow herda todas as propriedades de Window. O elemento NavigationWindow no arquivo XAML cria uma instância da classe NavigationWindow.

4. Alterar as propriedades a seguir sobre o NavigationWindow elemento:
    - Definir a propriedade Title para "Laboratório 01".
    - Definir a propriedade Width para 500 pixels.
    - Definir a propriedade Height para 350 pixels.
    - Remover o elemento Grid entre as marcas NavigationWindow.

XAML

```
<NavigationWindow x:Class="Lab01WPF.MainWindow"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   Title="Titulo da Janela Principal" Height="350" Width="500">
...
</NavigationWindow>
```

5. Abra MainWindow.xaml.cs.

Este arquivo é um arquivo de code-behind que contém o código para manipular eventos declarados na MainWindow. XAML. Este arquivo contém uma classe parcial para a janela definida em XAML.

Se você estiver usando o C#, altere o MainWindow classe derivar de NavigationWindow.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
```

```
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace Lab01WPF
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : NavigationWindow
    {
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

Adição de arquivos para o aplicativo.

Nesta seção, você vai adicionar duas páginas e uma imagem para o aplicativo.

Adicionar uma nova página (WPF) para o projeto, chamada Home.xaml.

Esta página é a primeira página que é exibida quando o aplicativo é iniciado. Ela mostrará uma lista de pessoas a partir da qual um usuário pode selecionar uma pessoa para mostrar um relatório de despesas.

6. Abra Home.xaml.
7. Definir o Title para "Despesas - Home".

XAML

```
<Page x:Class="Lab01WPF.Home"
    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:d="http://schemas.microsoft.com/expression/blend/2008"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="300"
          Title="Despesas - Home">

  <Grid>

  </Grid>
</Page>
```

1. Abra MainWindow. XAML.

Definir a propriedade Source no NavigationWindow para "Home.xaml".

Isso define o Home.xaml para ser a primeira página aberta quando o aplicativo for iniciado.

XAML

```
<NavigationWindow x:Class="Lab01WPF.MainWindow"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   Title="Laboratório 01" Height="350" Width="500" Source=”Home.xaml">

</NavigationWindow>
```

2. Adicionar uma nova página (WPF) para o projeto chamado RelatorioDespesasPage.xaml.

Essa página mostrará o relatório de despesas para a pessoa que está selecionada na Home.xaml.

- Abra RelatorioDespesasPage.xaml.
- Definir o Title para "Despesas do modo de exibição".

Veja o XAML a seguir:

XAML

```
<Page x:Class="Lab01WPF. RelatorioDespesasPage"
    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:d="http://schemas.microsoft.com/expression/blend/2008"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="300"  Title="Relatórios de despesas">
  <Grid>

  </Grid>
</Page>
```

3. Dê uma olhada nos arquivos Home.xaml.cs e RelatorioDespesasPage.xaml.cs.

Quando você cria um novo arquivo de página, o Visual Studio cria automaticamente um arquivo code-behind. Esses arquivos code-behind manipulam a lógica para responder à entrada do usuário.

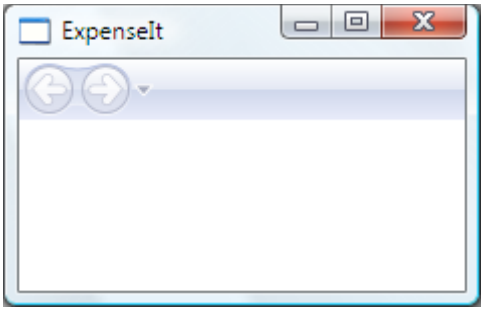
Seu código deve ter esta aparência.

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;


namespace Labo1WPF

{

   ///<summary>/// Interaction logic for RelarorioDespesasPage.xaml

///</summary>publicpartialclass RelarorioDespesasPage : Page

   {

      public RelarorioDespesasPage()

      {

         InitializeComponent();

      }

   }
```

}

4. Adicione uma imagem chamada watermark.png para o projeto. Você pode criar sua própria imagem ou copiar o arquivo de imagem da Web.
5. Construa e execute o aplicativo pressionando F5 ou selecione Start Debugging partir do Debug menu.

A ilustração a seguir mostra o aplicativo com o NavigationWindow botões.



Feche o aplicativo para retornar ao Visual Studio.

Criando um Layout

Os Layouts fornecem uma maneira de organizar os controles em uma UI, e também gerenciar o tamanho e o posicionamento destes controles quando UI é redimensionada. Normalmente, você cria um layout com um dos seguintes controles de layout:

- Canvas
- DockPanel
- Grid
- StackPanel
- VirtualizingStackPanel
- WrapPanel

Cada um desses controles de layout oferece suporte a um tipo especial de layout para seus elementos filho. As páginas de nossa aplicação podem ser redimensionadas, e cada página tem elementos que são arranjados horizontalmente e verticalmente. Conseqüentemente, o Grid é o layout ideal para esta aplicação.

Então, vamos criar uma tabela com três linhas e uma uma coluna, com margem de 10 pixels, adicionando definições de coluna e linha para o Grid em Home.xaml.

1. Abra Home.xaml.

2. Defina a propriedade Margin no elemento Grid para "10,0,10,10" que corresponde às margens esquerda, superior, direita e inferior.
3. Adicione o seguinte XAML entre as marcas Grid para criar definições de linha e coluna.

XAML

```
<Grid.ColumnDefinitions>
   <ColumnDefinition />
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
   <RowDefinition Height="Auto"/>
   <RowDefinition />
   <RowDefinition Height="Auto"/>
</Grid.RowDefinitions>
```

Como você pode ver a propriedade **Height** de duas linhas é definida como Auto que significa que serão dimensionadas de acordo com o conteúdo das linhas. Se duas linhas tem uma altura de "*", eles terão, cada uma a metade de altura do espaço disponível.

O Grid deve agora parecer com o seguinte XAML:

XAML

```
<Grid Margin="10,0,10,10">
   <Grid.ColumnDefinitions>
      <ColumnDefinition />
   </Grid.ColumnDefinitions>
   <Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
      <RowDefinition />
      <RowDefinition Height="Auto"/>
   </Grid.RowDefinitions>
</Grid>
```

Adicionando Controles

Nesta seção, a home page UI é atualizado para mostrar uma lista de pessoas que os usuários podem selecionar para mostrar o relatório de despesas para a pessoa selecionada. Controles são objetos de interface do usuário que permitem aos usuários interagir com seu aplicativo.

Para criar essa UI, os seguintes elementos são adicionados a página Home.xaml:

- ListBox (para lista de pessoas).
- Label (para exibir títulos).
- Button (para executar uma ação).

Cada controle é colocado em uma linha da Grid , definindo a propriedade Grid.Row.

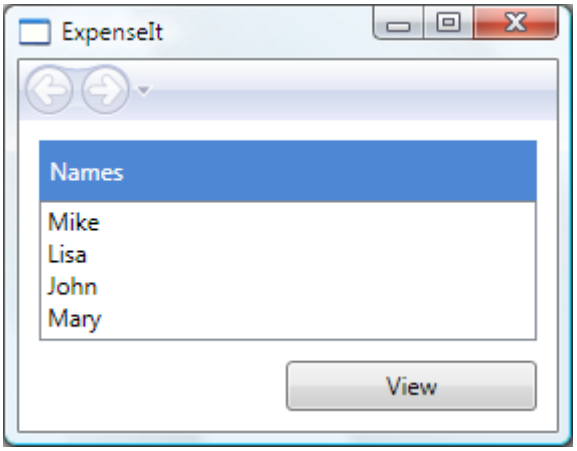
1. Abra o arquivo Home.xaml.
2. Adicione o seguinte XAML entre as marcas Grid.

XAML

```
 <!-- People list -->
 <Border Grid.Column="0" Grid.Row="0" Height="35" Padding="5" Background="#4E87D4">
    <Label VerticalAlignment="Center" Foreground="White">Names</Label>
 </Border>
 <ListBox Name="peopleListBox" Grid.Column="0" Grid.Row="1">
    <ListBoxItem>Mike</ListBoxItem>
    <ListBoxItem>Lisa</ListBoxItem>
    <ListBoxItem>John</ListBoxItem>
    <ListBoxItem>Mary</ListBoxItem>
 </ListBox>

 <!-- View report button -->
 <Button Grid.Column="0" Grid.Row="2" Margin="0,10,0,0" Width="125"
Height="25" HorizontalAlignment="Right">View</Button>
```

Crie e execute o aplicativo.

A ilustração a seguir mostra os controles que são criados pelo XAML nesta seção.



Adicionando um título e uma imagem

Vamos adicionar a nossa UI home page uma imagem e um título de página.

1. Abra Home.xaml.
2. Adicione outra coluna para o ColumnDefinitions com Width de 230 pixels.

XAML

```
<Grid.ColumnDefinitions>
   <ColumnDefinition Width="230" />
   <ColumnDefinition />
</Grid.ColumnDefinitions>
```

1. Adicionar outra linha para o RowDefinitions.

XAML

```
<Grid.RowDefinitions>
   <RowDefinition/>
   <RowDefinition Height="Auto"/>
   <RowDefinition />
```

```
  <RowDefinition Height="Auto"/>
</Grid.RowDefinitions>
```

2. Mova os controles para a segunda coluna, definindo Grid.Column como 1. Mova cada controle para baixo uma linha, aumentando a Grid.Row por 1.

XAML

```
 <Border Grid.Column="1" Grid.Row="1" Height="35" Padding="5" Background="#4E87D4">
    <Label VerticalAlignment="Center" Foreground="White">Names</Label>
 </Border>
 <ListBox Name="peopleListBox" Grid.Column="1" Grid.Row="2">
    <ListBoxItem>Mike</ListBoxItem>
    <ListBoxItem>Lisa</ListBoxItem>
    <ListBoxItem>John</ListBoxItem>
    <ListBoxItem>Mary</ListBoxItem>
 </ListBox>

 <!-- View report button -->
 <Button Grid.Column="1" Grid.Row="3" Margin="0,10,0,0" Width="125"
Height="25" HorizontalAlignment="Right">View</Button>
```

3. Definir o Background da Grid para ser o arquivo de imagem de watermark.png.

XAML

```
<Grid.Background>
  <ImageBrush ImageSource="watermark.png"/>
</Grid.Background>
```

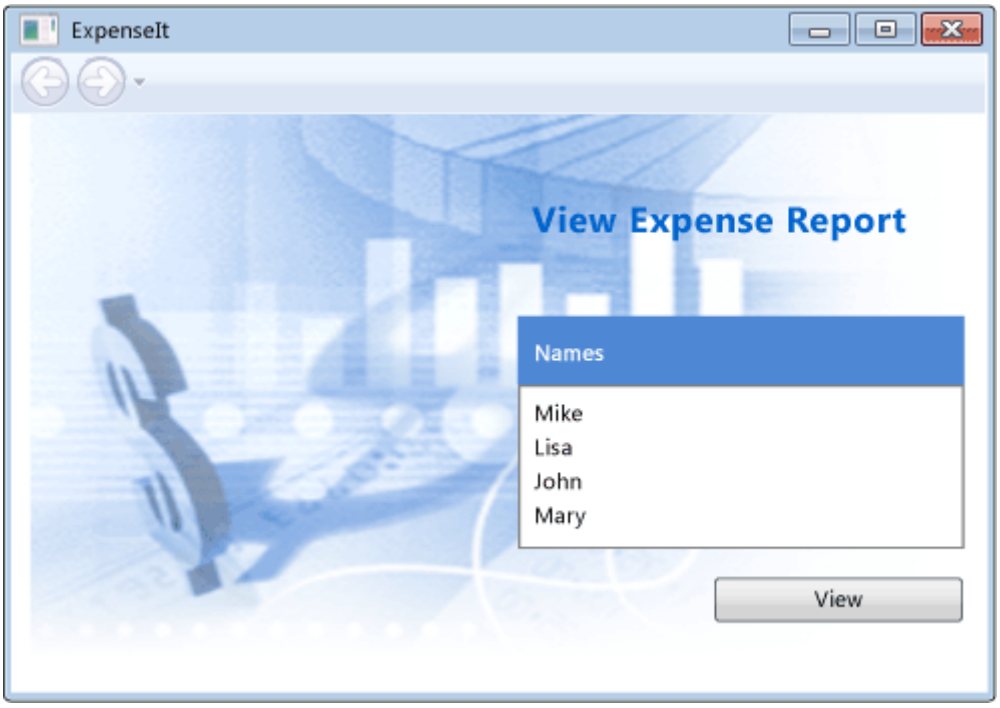
4. Antes de Border, adicione um Label com o conteúdo "Mode de exibição do relatório de despesas" para ser o título da página.

XAML

```
<Label Grid.Column="1" VerticalAlignment="Center" FontFamily="Trebuchet MS"
    FontWeight="Bold" FontSize="18" Foreground="#0066cc">
  Mode de exibição do relatório de despesas
</Label>
```

Crie e execute o aplicativo.

A ilustração a seguir mostra os resultados desta seção.



Adicionando código para manipular eventos

---

5. Abra Home.xaml.

Adicionar um manipulador de eventos Click para o elemento Button.

XAML

```
<!-- View report button -->
<Button Grid.Column="1" Grid.Row="3" Margin="0,10,0,0" Width="125"
```

Height="25" HorizontalAlignment="Right" Click="Button_Click">View</Button>

6. Abra Home.xaml.cs.
7. Adicione o seguinte código para o manipulador de eventos Click, que faz com que a janela navegue até o arquivo RelatorioDespesasPage.xaml.

```
private void Button_Click(object sender, RoutedEventArgs e)

{

   // Vê relatório de despesas

   RelatorioDespesasPage despesasPage = new RelatorioDespesasPage();

   this.NavigationService.Navigate(despesasPage);


}
```

Criando a interface do usuário para RelatorioDespesasPage

RelatorioDespesasPage.xaml exibe o relatório de despesas para a pessoa que foi selecionada em Home.xaml. Então, vamos adicionar os controles para montar a UI para esta página.

1. Abra o arquivo RelatorioDespesasPage.xaml.
2. Adicione o XAML a seguir entre as marcas Grid.

Essa interface do usuário é semelhante à interface do usuário criado em Home.xaml, exceto os dados do relatório são exibidos em um DataGrid.

XAML

```
<Grid.Background>

   <ImageBrush ImageSource="watermark.png" />

</Grid.Background>

<Grid.ColumnDefinitions>

   <ColumnDefinition Width="230" />

   <ColumnDefinition />

</Grid.ColumnDefinitions>
```

```
<Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition />
</Grid.RowDefinitions>



<Label Grid.Column="1" VerticalAlignment="Center" FontFamily="Trebuchet MS"
FontWeight="Bold" FontSize="18" Foreground="#0066cc">
    Relatório despesas para:

</Label>
<Grid Margin="10" Grid.Column="1" Grid.Row="1">
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition />
    </Grid.RowDefinitions>


    <!-- Name -->
    <StackPanel Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="0"
Orientation="Horizontal">
        <Label Margin="0,0,0,5" FontWeight="Bold">Name:</Label>
        <Label Margin="0,0,0,5" FontWeight="Bold">Department:</Label>
    </StackPanel>
```

```
    <!-- Department -->
    <StackPanel Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="1"
Orientation="Horizontal">
        <Label Margin="0,0,0,5" FontWeight="Bold">Department:</Label>
        <Label Margin="0,0,0,5" FontWeight="Bold"></Label>
    </StackPanel>

    <Grid Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="2" VerticalAlignment="Top"
          HorizontalAlignment="Left">
        <!-- Expense type and Amount table -->
        <DataGrid  AutoGenerateColumns="False" RowHeaderWidth="0" >
            <DataGrid.ColumnHeaderStyle>
                <Style TargetType="{x:Type DataGridColumnHeader}">
                    <Setter Property="Height" Value="35" />
                    <Setter Property="Padding" Value="5" />
                    <Setter Property="Background" Value="#4E87D4" />
                    <Setter Property="Foreground" Value="White" />
                </Style>
            </DataGrid.ColumnHeaderStyle>
            <DataGrid.Columns>
                <DataGridTextColumn Header="ExpenseType" />
                <DataGridTextColumn Header="Amount"  />
            </DataGrid.Columns>
        </DataGrid>
    </Grid>
</Grid>
```

Crie e execute o aplicativo.

1. Clique no botão Exibir.

Será exibida a página de relatório de despesas.

A ilustração a seguir mostra o resultado da página RelatorioDespesasPage.xaml. Observe que o botão de navegação Voltar está habilitado.

Controles de estilo

A aparência de vários elementos pode geralmente ser o mesmo para todos os elementos do mesmo tipo em um UI. A UI usa os estilos para tornar aparências reutilizáveis através de vários elementos. A capacidade de reutilização dos estilos ajuda a simplificar a criação e gerenciamento do XAML. Esta seção substitui os atributos dos elemento que foram definidos nas etapas anteriores, com estilos.

2. Abra Application.
3. Adicione o seguinte XAML entre as marcas Application.Resources:

XAML

```
<!-- Header text style -->

<Style x:Key="headerTextStyle" TargetType=”{x:Type Label}”>

   <Setter Property="VerticalAlignment" Value="Center"></Setter>

   <Setter Property="Label.FontFamily" Value="Trebuchet MS"></Setter>

   <Setter Property="Label.FontWeight" Value="Bold"></Setter>

   <Setter Property="Label.FontSize" Value="18"></Setter>
```

```
    <Setter Property="Label.Foreground" Value="#0066cc"></Setter>
</Style>

<!-- Label style -->
<Style x:Key="labelStyle" TargetType="{x:Type Label}">
    <Setter Property="VerticalAlignment" Value="Top" />
    <Setter Property="HorizontalAlignment" Value="Left" />
    <Setter Property="FontWeight" Value="Bold" />
    <Setter Property="Margin" Value="0,0,0,5" />
</Style>

<!-- DataGrid header style -->
<Style x:Key="columnHeaderStyle" TargetType="{x:Type DataGridColumnHeader}">
    <Setter Property="Height" Value="35" />
    <Setter Property="Padding" Value="5" />
    <Setter Property="Background" Value="#4E87D4" />
    <Setter Property="Foreground" Value="White" />
</Style>

<!-- List header style -->
<Style x:Key="listHeaderStyle" TargetType="{x:Type Border}">
    <Setter Property="Height" Value="35" />
    <Setter Property="Padding" Value="5" />
    <Setter Property="Background" Value="#4E87D4" />
</Style>

<!-- List header text style -->
```

```
<Style x:Key="listHeaderTextStyle" TargetType="{x:Type Label}">
   <Setter Property="Foreground" Value="White" />
   <Setter Property="VerticalAlignment" Value="Center" />
   <Setter Property="HorizontalAlignment" Value="Left" />
</Style>

<!-- Button style -->
<Style x:Key="buttonStyle" TargetType="{x:Type Button}">
   <Setter Property="Width" Value="125" />
   <Setter Property="Height" Value="25" />
   <Setter Property="Margin" Value="0,10,0,0" />
   <Setter Property="HorizontalAlignment" Value="Right" />
</Style>
```

O XAML antrerior, adiciona os seguintes estilos:

1. headerTextStyle: Para formatar o título da página Label.
2. labelStyle: Para formatar os controles Labels.
3. columnHeaderStyle: Para formatar a DataGridColumnHeader.
4. listHeaderStyle: Para formatar o cabeçalho da lista dos controles Borders.
5. listHeaderTextStyle: Para formatar o cabeçalho da lista Label.
6. buttonStyle: Para formatar o Button em Home.xaml.

Observe que os estilos são recursos, filhos da propriedade Application.Resources. Definindo os estilos desta forma, podemos aplicá-los a todos os elementos da aplicação. Abra a página tHome.xaml.

1. Faça as mudanças a seguir (em negrito) no XAML entre as marcas Grid.

XAML

```
<Grid.Background>
   <ImageBrush ImageSource="watermark.png"  />
</Grid.Background>
```

```
<Grid.ColumnDefinitions>
   <ColumnDefinition Width="230" />
   <ColumnDefinition />
</Grid.ColumnDefinitions>

<Grid.RowDefinitions>
   <RowDefinition/>
   <RowDefinition Height="Auto"/>
   <RowDefinition />
   <RowDefinition Height="Auto"/>
</Grid.RowDefinitions>

<!-- People list -->

<Label Grid.Column="1" Style="{StaticResource headerTextStyle}" >
   View Expense Report
</Label>

<Border Grid.Column="1" Grid.Row="1" Style="{StaticResource listHeaderStyle}">
   <Label Style="{StaticResource listHeaderTextStyle}">Names</Label>
</Border>
<ListBox Name="peopleListBox" Grid.Column="1" Grid.Row="2">
   <ListBoxItem>Mike</ListBoxItem>
   <ListBoxItem>Lisa</ListBoxItem>
   <ListBoxItem>John</ListBoxItem>
   <ListBoxItem>Mary</ListBoxItem>
</ListBox>
```

<!-- View report button -->

<Button Grid.Column="1" Grid.Row="3" Click="Button_Click" **Style="{StaticResource buttonStyle}"**>View</Button>

As propriedades como VerticalAlignment e FontFamily que definem a aparência de cada controle são removidas e substituídas aplicando estilos. Por exemplo, o headerTextStyle é aplicado para o Label ".

1. Abra o arquivo RelatorioDespesasPage.xaml.
2. Modifique o XAML como a seguir.

XAML

<Grid.Background>

  <ImageBrush ImageSource="watermark.png" />

</Grid.Background>

<Grid.ColumnDefinitions>

  <ColumnDefinition Width="230" />

  <ColumnDefinition />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

  <RowDefinition Height="Auto" />

  <RowDefinition />

</Grid.RowDefinitions>

<Label Grid.Column="1" Style="{StaticResource headerTextStyle}">

  Expense Report For:

</Label>

<Grid Margin="10" Grid.Column="1" Grid.Row="1">

  <Grid.ColumnDefinitions>

    <ColumnDefinition />

```
    <ColumnDefinition />
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" />
    <RowDefinition />
  </Grid.RowDefinitions>

  <!-- Name -->
  <StackPanel Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="0"
Orientation="Horizontal">
    <Label Style="{StaticResource labelStyle}">Name:</Label>
    <Label Style="{StaticResource labelStyle}"></Label>
  </StackPanel>

  <!-- Department -->
  <StackPanel Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="1"
Orientation="Horizontal">
    <Label Style="{StaticResource labelStyle}">Department:</Label>
    <Label Style="{StaticResource labelStyle}"></Label>
  </StackPanel>

  <Grid Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="2" VerticalAlignment="Top"
      HorizontalAlignment="Left">
    <!-- Expense type and Amount table -->
    <DataGrid ColumnHeader Style="{StaticResource columnHeaderStyle}"
          AutoGenerateColumns="False" RowHeaderWidth="0" >
      <DataGrid.Columns>
```

```
            <DataGridTextColumn Header="ExpenseType" />
            <DataGridTextColumn Header="Amount"  />
          </DataGrid.Columns>
      </DataGrid>
   </Grid>
</Grid>
```

Crie e execute o aplicativo.

Após adicionar o XAML nesta seção, o aplicativo procura os mesmos como fazia antes sendo atualizados com estilos.

Ligação de dados para um controle.

Agora, vamos cria o XML dos dados que serão vinculados aos controles.

1. Abra Home.xaml.
2. Adicione o seguinte XAML para criar um XmlDataProvider que contém os dados para cada pessoa.

Os dados são criados como um recurso. Normalmente, isso seria carregado como um arquivo, ou vindo de um banco de dados, mas para simplificar este exemplo, vamos adicionar os dados in-line.

XAML

Coloque este código no início do Grid.

```
<Grid.Resources>
...
<!-- Expense Report Data -->
<XmlDataProvider x:Key="ExpenseDataSource" XPath="Expenses">
   <x:XData>
      <Expenses xmlns="">
```

```
        <Person Name="Mike" Department="Legal">
          <Expenses ExpenseType="Lunch" ExpenseAmount="50" />
          <Expenses ExpenseType="Transportation" ExpenseAmount="50" />
        </Person>
        <Person Name="Lisa" Department="Marketing">
          <Expenses ExpenseType="Document printing"
    ExpenseAmount="50"/>
          <Expenses ExpenseType="Gift" ExpenseAmount="125" />
        </Person>
        <Person Name="John" Department="Engineering">
          <Expense ExpenseType="Magazine subscription"
   ExpenseAmount="50"/>
          <Expense ExpenseType="New machine" ExpenseAmount="600" />
          <Expense ExpenseType="Software" ExpenseAmount="500" />
        </Person>
        <Person Name="Mary" Department="Finance">
          <Expense ExpenseType="Dinner" ExpenseAmount="100" />
        </Person>
      </Expenses>
    </x:XData>
</XmlDataProvider>
...
</Grid.Resources>
```

No Grid recurso, adicione o seguinte DataTemplate, que define como exibir os dados do ListBox.

XAML

```
<Grid.Resources>
```

```
...
<!-- Name item template -->
<DataTemplate x:Key="nameItemTemplate">
   <Label Content="{Binding XPath=@Name}"/>
</DataTemplate>
...
</Grid.Resources>
```

1. Substituir o ListBox com o XAML a seguir.

XAML

```
<ListBox Name="peopleListBox" Grid.Column="1" Grid.Row="2"
      ItemsSource="{Binding Source={StaticResource ExpenseDataSource}, XPath=Person}"
      ItemTemplate="{StaticResource nameItemTemplate}">
</ListBox>
```

Esse XAML vincula a propriedade ItemsSource da ListBox à fonte de dados e aplica o modelo de dados, como o ItemTemplate.

Conectando a dados a cada controles

Agora, vamos escreve o código que recupera o item atual que está selecionado na lista de pessoas na página Home.xaml e passa sua referência para o construtor da página RelatorioDespesasPage durante a instanciação.

2. Abra RelatorioDespesasPage.xaml.cs.
3. Adicione um construtor que leva um objeto que passar os dados do relatório de despesas da pessoa selecionada.

```
public partial class RelatorioDespesasPage : Page
{
   public ExpenseReportPage()
```

```
    {
        InitializeComponent();
    }

    // Constructor personalizado para exibir as despesas
    public RelatorioDespesasPage(object data):this()
    {
        // Bind to expense report data.
        this.DataContext = data;
    }

}
```

1. Abra o arquivo Home.xaml.cs.

Altere o manipulador de eventos Click para chamar o construtor new, passando os dados do relatório de despesas da pessoa selecionada.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    // Exibindo o relatório de despesas
    ExpenseReportPage expenseReportPage = new
ExpenseReportPage(this.peopleListBox.SelectedItem);
    this.NavigationService.Navigate(expenseReportPage);

}
```

Formatando o Estilos dos Dados

Agora, vamos atualizar a UI para cada item de dados vinculado usando modelos de dados.

2. Abra o arquivo RelatorioDespesasPage.xaml.

Vincular o conteúdo de "Nome" e "Departamento" a propriedade do Label apropriado.

XAML

```
<!-- Name -->
<StackPanel Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="0" Orientation="Horizontal">
   <Label Style="{StaticResource labelStyle}">Name:</Label>
   <Label Style="{StaticResource labelStyle}" Content="{Binding XPath=@Name}"></Label>
</StackPanel>

<!-- Department -->
<StackPanel Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="1" Orientation="Horizontal">
   <Label Style="{StaticResource labelStyle}">Department:</Label>
   <Label Style="{StaticResource labelStyle}" Content="{Binding
XPath=@Department}"></Label>
</StackPanel>
```

Agora, vamos fazer a vinculação dos dados ao Grid, adicione os seguintes modelos de dados, para a página RelatorioDespesasPage.XAML que definem como exibir dados de relatório de despesas.

XAML

```
<!--Templates to display expense report data-->
<Grid.Resources>
   <!-- Reason item template -->
   <DataTemplate x:Key="typeItemTemplate">
      <Label Content="{Binding XPath=@ExpenseType}"/>
   </DataTemplate>
   <!-- Amount item template -->
   <DataTemplate x:Key="amountItemTemplate">
```

```
      <Label Content="{Binding XPath=@ExpenseAmount}"/>
   </DataTemplate>
</Grid.Resources>
```

Aplicar os modelos para as colunas do DataGrid que exibem dados de relatório de despesas.

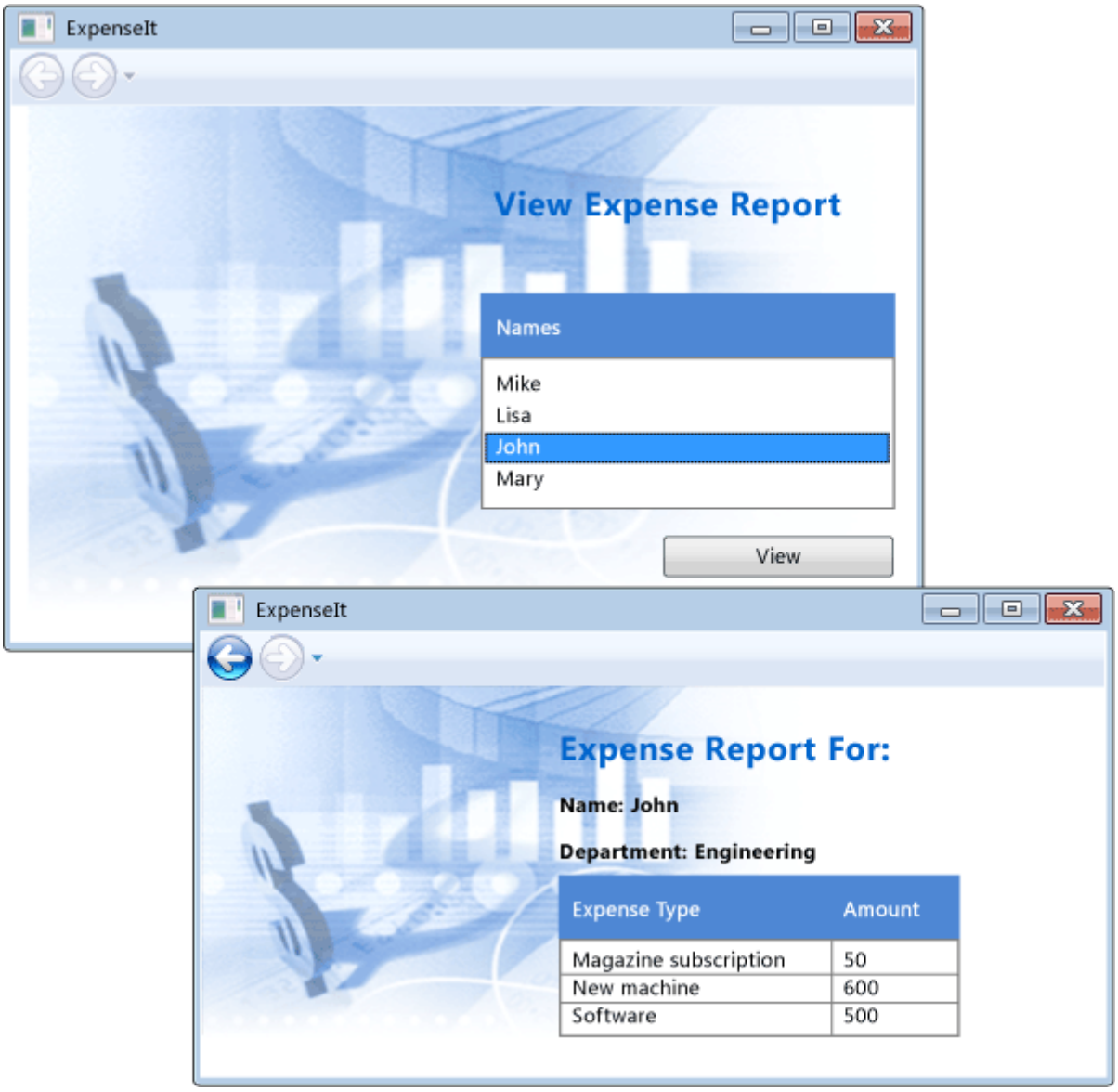
XAML

```
<!-- Expense type and Amount table -->
<DataGrid ItemsSource="{Binding XPath=Expenses}" ColumnHeaderStyle="{StaticResource columnHeaderStyle}" AutoGenerateColumns="False" RowHeaderWidth="0" >

   <DataGrid.Columns>
      <DataGridTextColumn Header="ExpenseType" Binding="{Binding XPath=@ExpenseType}" />
      <DataGridTextColumn Header="Amount" Binding="{Binding XPath=@ExpenseAmount}" />
   </DataGrid.Columns>

</DataGrid>
```

Crie e execute o aplicativo.

Selecione uma pessoa e clique no Exibir botão.

A ilustração a seguir mostra ambas as páginas do aplicativo com controles, layout, estilos, ligação de dados e modelos de dados aplicados.


ExpenseIt
View Expense Report
Names
Mike
Lisa
John
Mary
View
ExpenseIt
Expense Report For:
Name: John
Department: Engineering
Expense Type
Amount
Magazine subscription
50
New machine
600
Software
500