



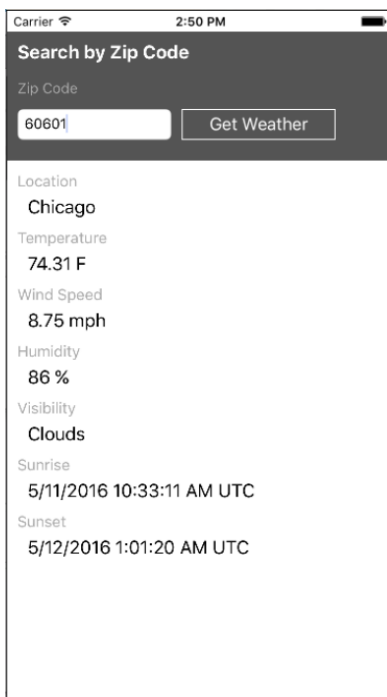
Fonte: <https://msdn.microsoft.com/pt-br/library/Dn879698.aspx>

Criar aplicativos com interface de usuário usando o Xamarin no Visual Studio

Visual Studio 2015

Para obter a documentação mais recente do Visual Studio 2017 RC, consulte a [documentação do Visual Studio 2017 RC](#).

Depois de fazer as etapas [A instalação](#) e [Verifique se o seu ambiente do Xamarin](#), este passo a passo mostra como criar um aplicativo básico (mostrado abaixo) com as camadas de interface de usuário nativas. Com a interface de usuário, código compartilhado reside em uma biblioteca de classes portátil (PCL) e os projetos de plataforma individuais contêm as definições de interface do usuário.





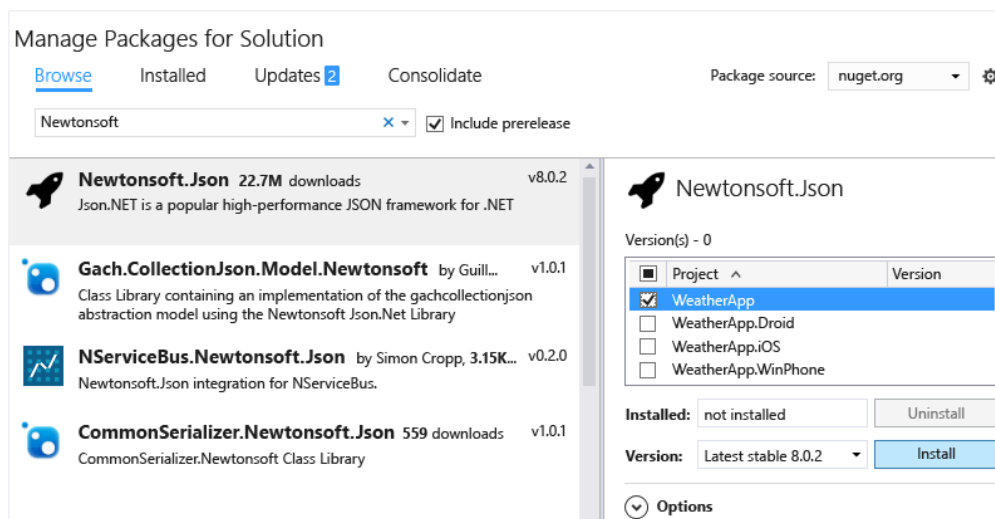
Configurar a sua solução

Estas etapas criam uma solução Xamarin com interface de usuário que contenha uma PCL para o código compartilhado e dois pacotes NuGet adicionados.

1. No Visual Studio, crie um novo **aplicativo em branco (nativo portátil)** solução e nomeie-o **WeatherApp**. Você pode encontrar mais facilmente esse modelo digitando **portátil nativo** no campo de pesquisa.
Se não for, você talvez precise instalar o Xamarin ou habilitar o recurso do Visual Studio 2015.
2. Depois de clicar em OK para criar a solução, você terá um número de projetos individuais:
 - **WeatherApp (portátil)**: PCL onde você escreverá código que é compartilhado entre plataformas, incluindo lógica de negócios comum e código de interface do usuário com xamarin.
 - **WeatherApp.Droid**: o projeto que contém o código nativo do Android. Isso é definido como o projeto de inicialização padrão.
 - **WeatherApp.iOS**: o projeto que contém o código nativo do iOS.
 - **WeatherApp.WinPhone (Windows Phone 8.1)**: o projeto que contém o código nativo do Windows Phone.

Dentro de cada projeto nativo, você tem acesso ao designer nativo para a plataforma correspondente e pode implementar telas específicas de plataforma.

3. Adicionar o **newtonsoft** e pacote do NuGet para o projeto PCL, que você usará para processar as informações recuperadas de um serviço de dados de tempo:
 - No Gerenciador de pacotes NuGet (ainda aberta da etapa 2), selecione o **Procurar** guia e procure **Newtonsoft**.
 - Selecione **newtonsoft**.
 - Verifique o **WeatherApp** projeto (esse é o projeto único em que você precisa instalar o pacote).
 - Verifique o **versão** campo é definido como o **estável mais recente** versão.
 - Clique em **instalar**.





4. Repita a etapa 3 para localizar e instalar o **Microsoft.Net.Http** pacote.
5. Compile sua solução e verifique não se há nenhum erro de compilação.

Escrever código de serviço de dados compartilhados

O **WeatherApp (portátil)** projeto é onde você escreverá código para a biblioteca de classes portátil (PCL) que é compartilhado entre todas as plataformas. PCL é automaticamente incluída no aplicativo construir pacotes, o iOS, Android e Windows Phone projetos.

Para executar este exemplo, primeiro você deve se inscrever para uma chave API gratuita na <http://openweathermap.org/appid>.

As etapas a seguir, em seguida, adicionam código para o PCL para acessar e armazenar dados de serviço meteorológico:

1. Clique com botão direito do **WeatherApp** do projeto e selecione **Adicionar > classe....** No **Add New Item** caixa de diálogo, nomeie o arquivo **Weather.cs**. Você usará essa classe para armazenar dados do serviço de dados meteorológicos.
2. Substitua todo o conteúdo de **Weather.cs** com o seguinte:

C#

```
namespace WeatherApp
{
    public class Weather
    {
        public string Title { get; set; }
        public string Temperature { get; set; }
        public string Wind { get; set; }
        public string Humidity { get; set; }
        public string Visibility { get; set; }
        public string Sunrise { get; set; }
        public string Sunset { get; set; }

        public Weather()
        {
            //Because labels bind to these values, set them to a
            //empty string to
            //ensure that the label appears on all platforms by
            //default.
            this.Title = " ";
            this.Temperature = " ";
            this.Wind = " ";
            this.Humidity = " ";
            this.Visibility = " ";
            this.Sunrise = " ";
            this.Sunset = " ";
        }
    }
}
```

3. Adicione outra classe ao projeto PCL chamado **DataService.cs** em que você usará para processar dados JSON do serviço de dados meteorológicos.
4. Substitua todo o conteúdo de **DataService.cs** com o código a seguir:



C#

```
using System.Threading.Tasks;
using Newtonsoft.Json;
using System.Net.Http;

namespace WeatherApp
{
    public class DataService
    {
        public static async Task<dynamic> getDataFromService(string queryString)
        {
            HttpClient client = new HttpClient();
            var response = await client.GetAsync(queryString);

            dynamic data = null;
            if (response != null)
            {
                string json = response.Content.ReadAsStringAsync
().Result;
                data = JsonConvert.DeserializeObject(json);
            }

            return data;
        }
    }
}
```

5. Adicionar uma classe de terceiros para o PCL chamado **Core** onde você colocará compartilhados a lógica de negócios, como lógica que formam uma cadeia de caracteres de consulta com um código postal, chama o serviço de dados de clima e preenche uma instância do **clima** classe.
6. Substitua o conteúdo do **Core.cs** com o seguinte:

C#

```
using System;
using System.Threading.Tasks;

namespace WeatherApp
{
    public class Core
    {
        public static async Task<Weather> GetWeather(string zipCode)
        {
            //Sign up for a free API key at http://openweathermap.org/appid
            string key = "YOUR KEY HERE";
            string queryString = "http://api.openweathermap.org/data/2.5/weather?zip="
+ zipCode + ",us&appid=" + key + "&units=imperial";
        }
    }
}
```



```
dynamic results = await DataService.GetDataFromService(queryString).ConfigureAwait(false);

if (results["weather"] != null)
{
    Weather weather = new Weather();
    weather.Title = (string)results["name"];
    weather.Temperature = (string)results["main"]["temp"] + " F";
    weather.Wind = (string)results["wind"]["speed"] + " mph";
    weather.Humidity = (string)results["main"]["humidity"] + " %";
    weather.Visibility = (string)results["weather"][0]["main"];

    DateTime time = new System.DateTime(1970, 1, 1, 0, 0, 0, 0);
    DateTime sunrise = time.AddSeconds((double)results["sys"]["sunrise"]);
    DateTime sunset = time.AddSeconds((double)results["sys"]["sunset"]);
    weather.Sunrise = sunrise.ToString() + " UTC";
    weather.Sunset = sunset.ToString() + " UTC";
    return weather;
}
else
{
    return null;
}
}
```

7. Exclua MyClass.cs na PCL porque nós não usá-lo.
8. Criar o **WeatherApp** projeto PCL para certificar-se de que o código está correto.

Design da interface do usuário para Android

Agora, podemos criar a interface do usuário, conectá-lo ao seu código compartilhado e, em seguida, execute o aplicativo.

Criar a aparência de seu aplicativo

1. Em **Solution Explorer**, expanda o **WeatherApp (Android)** > **recursos** > **layout** pasta e abra **Main.axml**. Isso abre o arquivo no visual designer.
2. Selecione e exclua o botão padrão que aparece no designer.
3. Do **Toolbox**, arraste um **RelativeLayout** controle para o designer. Você usará esse controle como um contêiner para outros controles.
4. No **propriedades** janela, defina o **fundo** propriedade **#545454**.
5. Do **Toolbox**, arraste um **TextView** controle para o **RelativeLayout** controle.
6. No **propriedades** janela, defina essas propriedades:



Propriedade	Valor
texto	Pesquisa por CEP
id	@+id/ZipCodeSearchLabel
layout_centerVertical	True
layout_marginLeft	10dp
textColor	@android:color/white
estilo de texto	Bold

7. Do **Toolbox**, arraste um **TextView** controle para o **RelativeLayout** de controle e posicione-o abaixo do controle ZipCodeSearchLabel. Para fazer isso, soltando o novo controle na borda do controle existente apropriado.
8. No **propriedades** janela, defina essas propriedades:

Propriedade	Valor
texto	CEP
id	@+id/ZipCodeLabel




Propriedade	Valor
layout_marginLeft	10dp
layout_marginTop	5dp

9. Do **Toolbox**, arraste um **número** controle para o **RelativeLayout**, posicioná-lo abaixo para o **CEP** rótulo. Em seguida, defina as seguintes propriedades:

Propriedade	Valor
id	@+id/ZipCodeEntry
layout_centerVertical	true
layout_marginLeft	10dp
layout_marginBottom	10dp
largura	165dp

10. Do **Toolbox**, arraste um **botão** para o **RelativeLayout** de controle e posicione-o à direita do controle ZipCodeEntry. Defina as propriedades:



Propriedade	Valor
id	@+id/weatherBtn
texto	Obtenha o clima
layout_marginLeft	20dp
layout_alignBottom	@id/zipCodeEntry
largura	165dp
 Observação	
Se o layout parece não ser exibidos corretamente, tente salvar o arquivo e alternar entre o Design e fonte guias.	

11. Agora você tem experiência suficiente para criar uma interface do usuário básica usando o designer Android. Mas você também pode criar uma interface do usuário, adicionando marcas diretamente para o arquivo .asxml da página. Vamos criar o restante da interface do usuário com isso.
12. Na parte inferior do designer, escolha o **fonte** guia.
13. Cole a marcação a seguir abaixo do `</RelativeLayout>` marca.

XML

```
<TextView
    android:text="Location"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

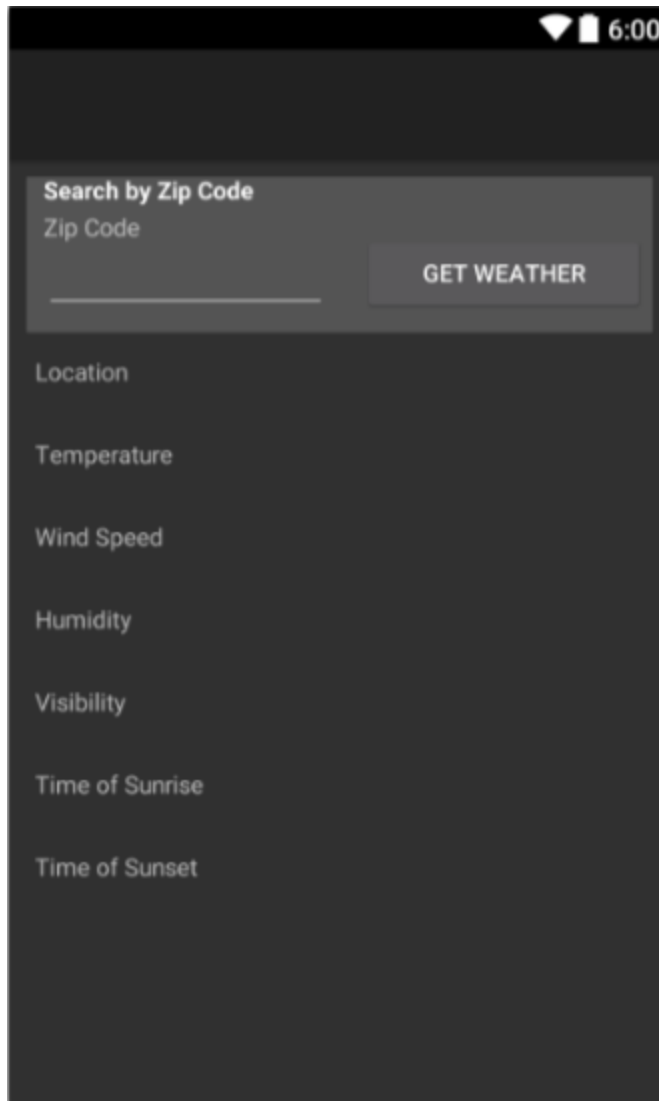



```
        android:id="@+id/locationLabel"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="10dp" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedi
um"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/locationText"
        android:layout_marginLeft="20dp"
        android:layout_marginBottom="10dp" />
    <TextView
        android:text="Temperature"
        android:textAppearance="?android:attr/textAppearanceSma
l"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/tempLabel"
        android:layout_marginLeft="10dp" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedi
um"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/tempText"
        android:layout_marginBottom="10dp"
        android:layout_marginLeft="20dp" />
    <TextView
        android:text="Wind Speed"
        android:textAppearance="?android:attr/textAppearanceSma
l"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/windLabel"
        android:layout_marginLeft="10dp" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedi
um"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/windText"
        android:layout_marginBottom="10dp"
        android:layout_marginLeft="20dp" />
    <TextView
        android:text="Humidity"
        android:textAppearance="?android:attr/textAppearanceSma
l"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/humidityLabel"
        android:layout_marginLeft="10dp" />
    <TextView
        android:textAppearance="?android:attr/textAppearanceMedi
um"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/humidityText"
        android:layout_marginBottom="10dp"
```



```
        android:layout_marginLeft="20dp" />
<TextView
    android:text="Visibility"
    android:textAppearance="?android:attr/textAppearanceSmall"
1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/visibilityLabel"
    android:layout_marginLeft="10dp" />
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
um"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/visibilityText"
    android:layout_marginBottom="10dp"
    android:layout_marginLeft="20dp" />
<TextView
    android:text="Time of Sunrise"
1"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/sunriseLabel"
    android:layout_marginLeft="10dp" />
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
um"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/sunriseText"
    android:layout_marginBottom="10dp"
    android:layout_marginLeft="20dp" />
<TextView
    android:text="Time of Sunset"
1"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/sunsetLabel"
    android:layout_marginLeft="10dp" />
<TextView
    android:textAppearance="?android:attr/textAppearanceMedium"
um"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/sunsetText"
    android:layout_marginBottom="10dp"
    android:layout_marginLeft="20dp" />
```

14. Alternar para **Design** exibição. A interface do usuário deve aparecer da seguinte maneira:



15. Compile a solução. Observe que adiciona criando controlar IDs para a **Resource.Designer.cs** de arquivos para que se referem a controles por nome no código.

Consumir seu código compartilhado

1. Abra o **MainActivity.cs** arquivo o **WeatherApp** no editor de código do projeto e substitua seu conteúdo pelo seguinte:

C#

```
using System;
using Android.App;
using Android.Widget;
using Android.OS;

namespace WeatherApp.Droid
{
    [Activity(Label = "Sample Weather App", MainLauncher = true,
    Icon = "@drawable/icon")]
    public class MainActivity : Activity
```



```
{
    protected override void OnCreate(Bundle bundle)
    {
        base.OnCreate(bundle);

        SetContentView(Resource.Layout.Main);

        Button button = FindViewById<Button>(Resource.Id.weatherBtn);

        button.Click += Button_Click;
    }

    private async void Button_Click(object sender, EventArgs e)
    {
        EditText zipCodeEntry = FindViewById<EditText>(Resource.Id.zipCodeEntry);

        if (!String.IsNullOrEmpty(zipCodeEntry.Text))
        {
            Weather weather = await Core.GetWeather(zipCodeEntry.Text);
            FindViewById<TextView>(Resource.Id.locationText).Text = weather.Title;
            FindViewById<TextView>(Resource.Id.tempText).Text = weather.Temperature;
            FindViewById<TextView>(Resource.Id.windText).Text = weather.Wind;
            FindViewById<TextView>(Resource.Id.visibilityText).Text = weather.Visibility;
            FindViewById<TextView>(Resource.Id.humidityText).Text = weather.Humidity;
            FindViewById<TextView>(Resource.Id.sunriseText).Text = weather.Sunrise;
            FindViewById<TextView>(Resource.Id.sunsetText).Text = weather.Sunset;
        }
    }
}
```

2. Esse código chama o **GetWeather** método que você definiu no seu código compartilhado. Em seguida, na interface do usuário do aplicativo, ele mostra os dados que são recuperados do método.

Execute o aplicativo e ver a aparência dele

1. Em **Solution Explorer**, defina o **WeatherApp** projeto como o projeto de inicialização.
2. Inicie o aplicativo pressionando a tecla F5.
3. No emulador do Android, digite um CEP válido na caixa de edição (por exemplo: 60601) e pressione **obter clima**. Dados meteorológicos para essa região, será exibida nos controles.

