Desarrollo de un Apps para Windows8 RT , con gestión de datos.

App Calendar

Contenido

App Calendario.

Información general

En el presente taller vamos a realizar una aplicación esencial, en cual ustedes podrán tener un calendario de sus eventos, para ello se requiere de tener nociones básicas para crear una aplicación estilo Metro con C#.

Crearemos una aplicación que permita almacenar, modificar, eliminar y listar información registrada por el usuario utilizando la interface de Windows Store y aprendiendo a manejar los parámetros que VisualStudio 2012 ofrece para este tipo de desarrollos.

Expectativas y objetivos Al finalizar el estudiante tendrá la capacidad de crear su propia aplicación respetando los parámetros que se establecen para el Windows Strore, así como se comprenderá la noción de implementar y utilizar el motor de datos SQLite para aplicaciones móviles.

Prerequisitos.

Equipo con sistema operativo Windows 8.

Visual Studio 2012.

Instalar SQLite para utilizarla en la Aplicación.

Imágenes con las especificaciones que se darán más adelante para la personalización de la aplicación.

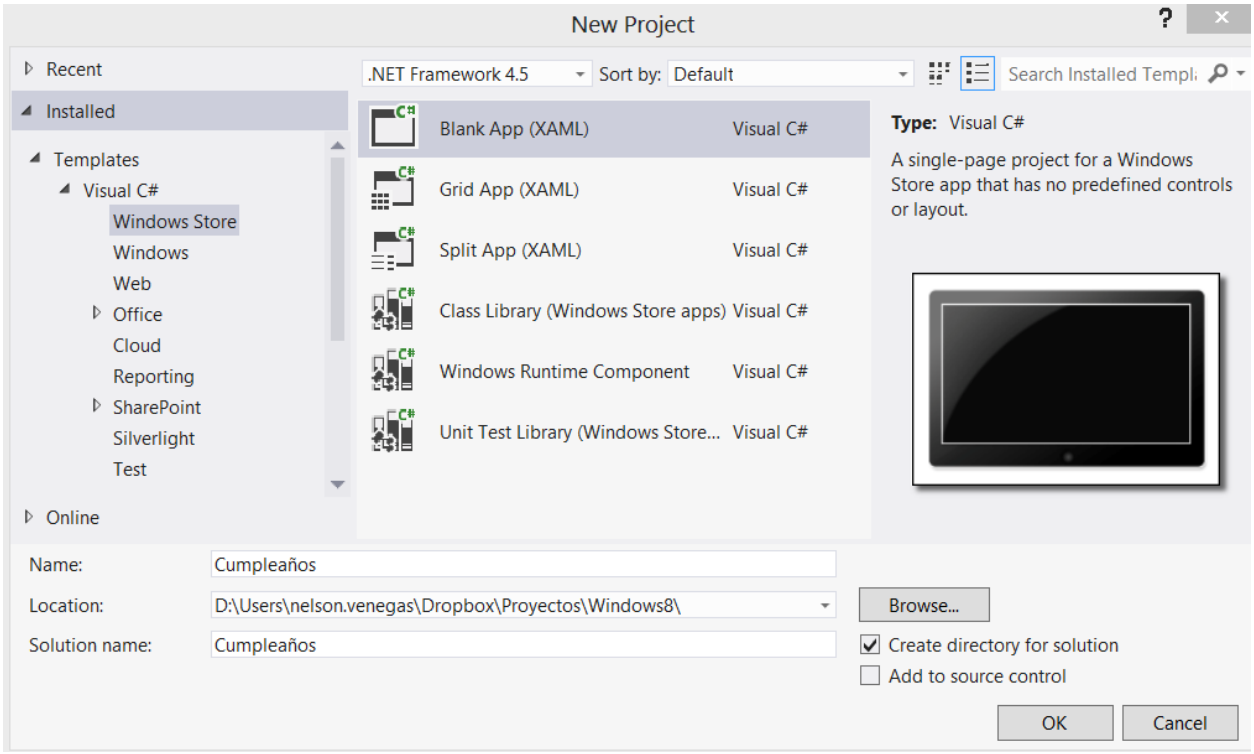
Conocimientos básicos en XAML y C#.

Usaremos XAML, para definir la interfaz de usuario y para escribir la lógica de la aplicación utilizaremos C#.

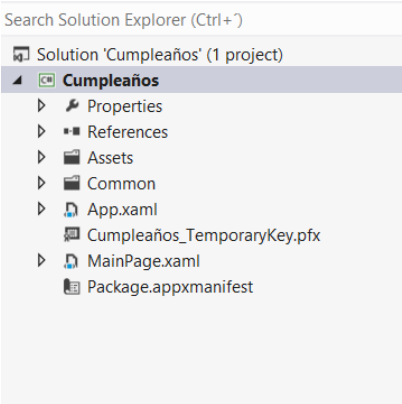
Para crear un nuevo proyecto Windows Store

1. Abre Visual Studio Express 2012 for Windows 8.
2. Selecciona **Archivo** > **Nuevo proyecto**. Se abre el cuadro de diálogo **Nuevo proyecto**.
3. En el panel **Instalado**, expande **Visual C#** o **Visual Basic**.
4. Selecciona el tipo de plantilla **Windows Store**.
5. En el panel central, selecciona **Aplicación vacía (XAML)**.
6. Escribe un nombre para el proyecto. Llama a este proyecto "Cumpleaños".

Así es como se crea un proyecto en Visual Studio Express 2012 for Windows 8.

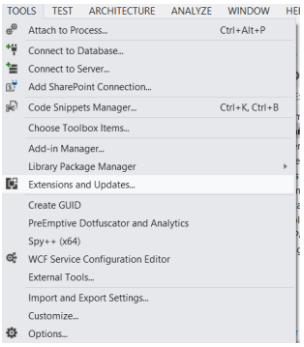


En el explorador se presentara de la siguiente forma:

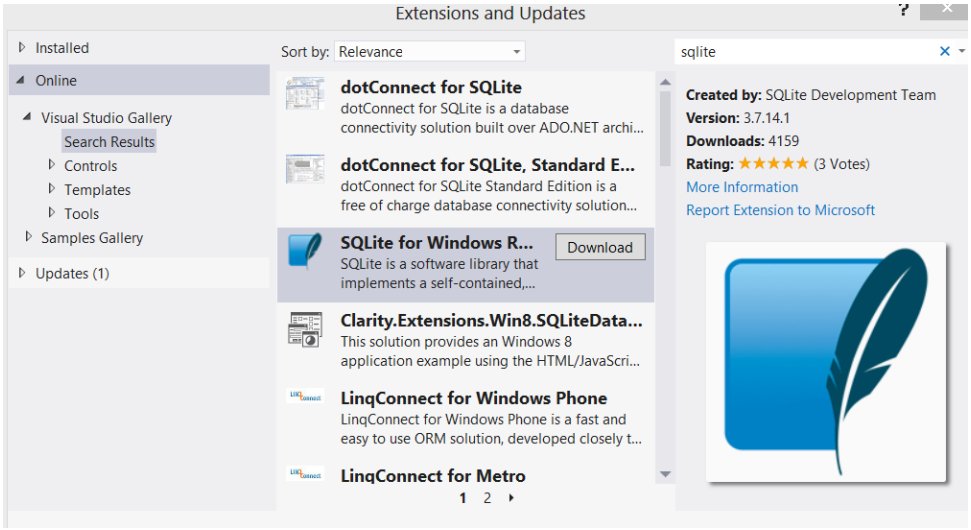


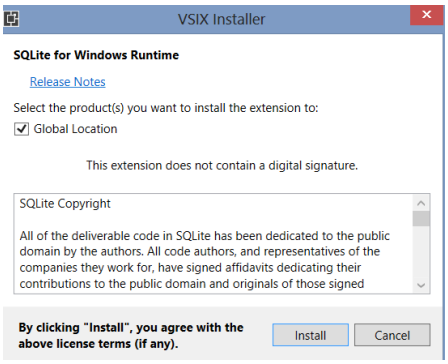
Instalacion de SQLite

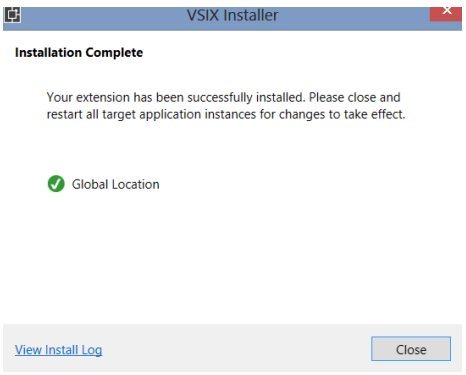
Debido a que vamos a trabajar con datos en la aplicación, vamos a instalar un componente llamado SQLLite, para ello dentro de visual studio 2012, nos vamos al menú Herramientas, vamos a extesiones y actualizaciones:

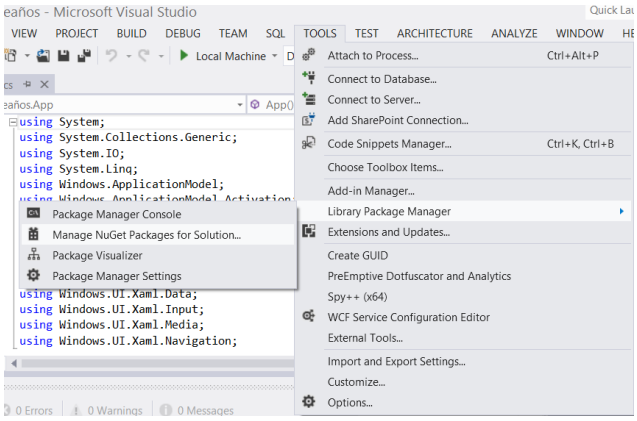


Al presionar extensiones y actualizaciones se nos presentara la siguiente ventana, en donde seleccionaremos EnLinea y en la parte izquierda digitaremos SQLite, y nos aparecerá como aparece acontinuacion y lo descargamos e instalamos.


Extensions and Updates
Installed
Online
Visual Studio Gallery
Search Results
Controls
Templates
Tools
Samples Gallery
Updates (1)
Sort by: Relevance
sqlite
dotConnect for SQLite
dotConnect for SQLite is a database connectivity solution built over ADO.NET archi...
dotConnect for SQLite, Standard E...
dotConnect for SQLite Standard Edition is a free of charge database connectivity solution...
SQLite for Windows R...
Download
SQLite is a software library that implements a self-contained,...
Clarity.Extensions.Win8.SQLiteData...
This solution provides an Windows 8 application example using the HTML/JavaScri...
LinqConnect for Windows Phone
LinqConnect for Windows Phone is a fast and easy to use ORM solution, developed closely t...
LinqConnect for Metro
1 2
Created by: SQLite Development Team
Version: 3.7.14.1
Downloads: 4159
Rating: (3 Votes)
More Information
Report Extension to Microsoft

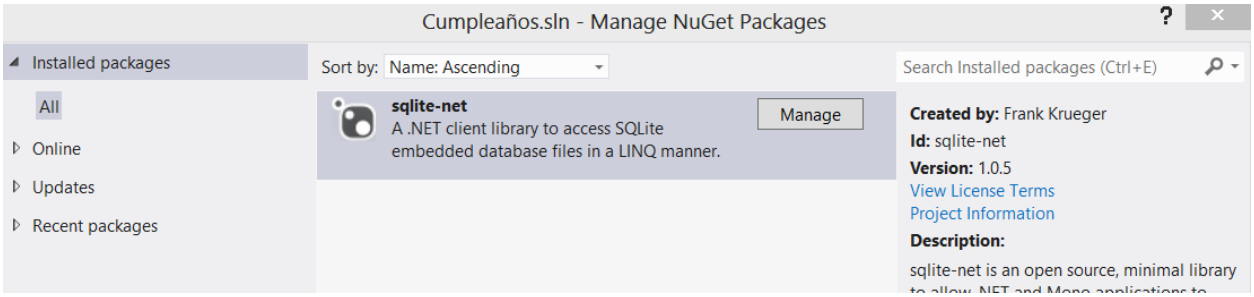

VSIX Installer
SQLite for Windows Runtime
Release Notes
Select the product(s) you want to install the extension to:
Global Location
This extension does not contain a digital signature.
SQLite Copyright
All of the deliverable code in SQLite has been dedicated to the public domain by the authors. All code authors, and representatives of the companies they work for, have signed affidavits dedicating their contributions to the public domain and originals of those signed
By clicking "Install", you agree with the above license terms (if any).
Install
Cancel


VSIX Installer
Installation Complete
Your extension has been successfully installed. Please close and restart all target application instances for changes to take effect.
Global Location
View Install Log
Close

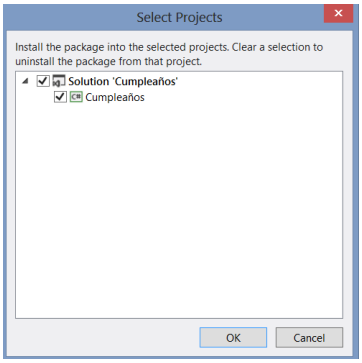

VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW
Local Machine
Attach to Process... Ctrl+Alt+P
Connect to Database...
Connect to Server...
Add SharePoint Connection...
Code Snippets Manager... Ctrl+K, Ctrl+B
Choose Toolbox Items...
Add-in Manager...
Library Package Manager
Extensions and Updates...
Create GUID
PreEmptive Dotfuscator and Analytics
Spy++ (x64)
WCF Service Configuration Editor
External Tools...
Import and Export Settings...
Customize...
Options...
Package Manager Console
Manage NuGet Packages for Solution...
Package Visualizer
Package Manager Settings
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using Windows.ApplicationModel;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
0 Errors 0 Warnings 0 Messages

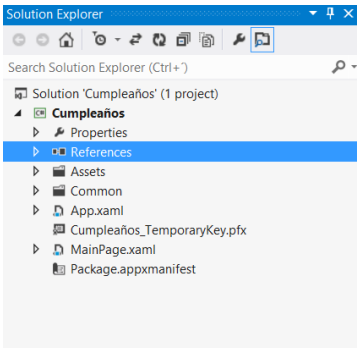
Instalacion de sqlite-net

De la misma forma como descargamos SQLite, procederemos a descargar sqlite-net , la cual es una librería que permite a .Net almacenar en bases de datos de SQLite. Al descargarlo seleccionamos el proyecto y le damos manage sobre sqllite. Nos permite crear el contexto de nuestra base de datos

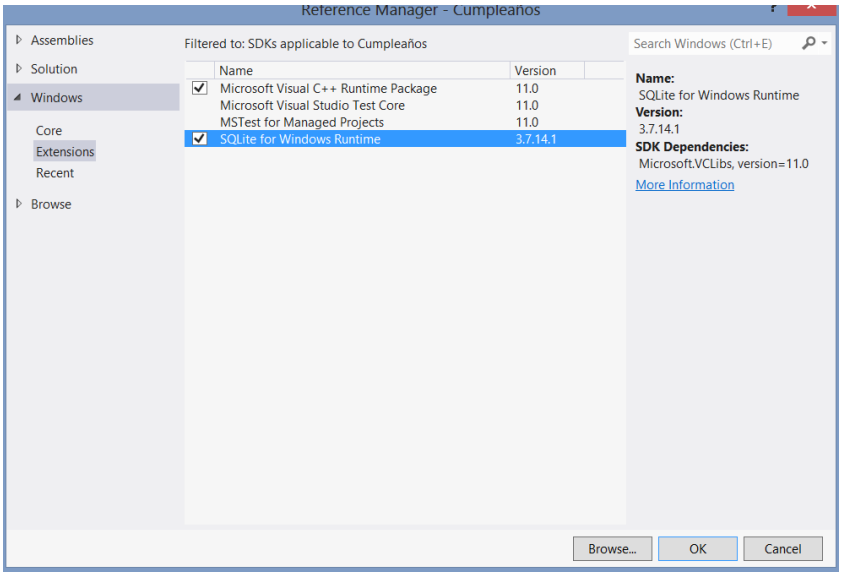


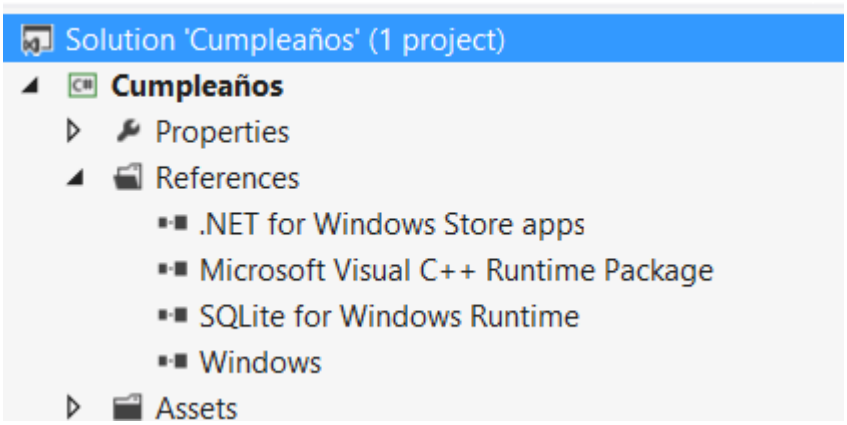
Nos aparece la siguiente ventana la cual nos pregunta en que proyecto desea colocar sqlite, seleccionamos el proyecto cumpleaños.





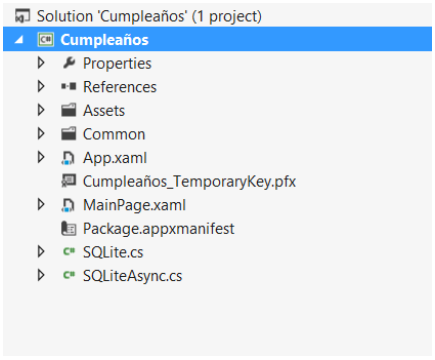
Agregacion de referencias al proyecto.

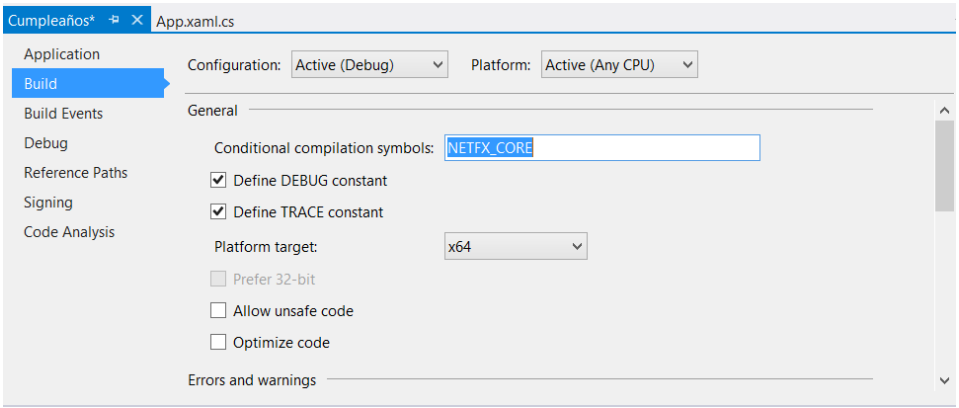




Selección de compilación del Proyecto

Debemos seleccionar la plataforma en la cual vamos a ejecutar nuestro proyecto, para ello vamos a la propiedades de nuestro proyecto , en la pestaña Build seleccionamos x64.





Creación de la lógica de negocio.

Agregue una clase llamada persona la cual va a tener los siguientes campos:

Nombres (tipo string).

Fecha (tipo DateTime).

IdEvento (Tipo int)

```
namespace MyPrimeraBD.Common.Data
{
    public class Eventos
    {
        [SQLite.PrimaryKey, SQLite.AutoIncrement]
        public int idEvento { get; set; }
        public string Nombres { get; set; }
        public DateTime Fecha { get; set; }
    }
}
```

Creacion de la clase que administrara los datos de las tablas.

Con el fin de crear el contexto de la base de datos, vamos a crear una clase llamada contexto , en la cual crearemos la base de datos y lo métodos para CRUD del evento en el calendario. En la clase contexto adiciomos el using de SQLite y dicha clase la Heredamos de SQLiteConnection, por lo que quedaría de la siguiente manera.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SQLite;

namespace Cumpleaños.Common.Data
{
    class Contexto: SQLiteConnection
    {
    }
}
```

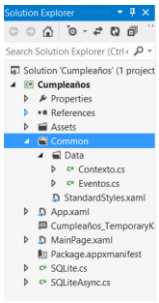
Ahora lo que vamos a crear el contexto en el constructor de la clase, por lo que dicha clase quedaría de la siguiente forma:

```
class Contexto: SQLiteConnection
    {
        public Contexto(string path)
            : base(path)
        {

        }

    }
```

Ahora lo que vamos a crear es la tabla con la cual vamos a trabajar en nuestra aplicación y los respetivos métodos eliminar, adicionar, editar, agregar, la clase quedaría de la siguiente manera:

```
class Contexto: SQLiteConnection
    {
        public Contexto(string path)
            : base(path)
        {
            CreateTable<Eventos>();
            CreateTable<Categoria>();
        }
        #region Eventos
        public int Addevento(Eventos evento)
        {
            return Insert(evento);
        }

        public IEnumerable<Eventos> GetEventos()
        {
        return from c in Table<Eventos>()
        select c;
        }

        public int Deleteevento(int idEvento)
        {
            var a = GetCursos(idEvento).Count();
            if (a > 0)
            {
                throw new Exception("No se puede eliminar el evento");
            }
            else
            {
                return Delete<Eventos>(idEvento);
            }
        }

        public int Editevento(Eventos evento)
        {
            return Update(evento);
        }

        #endregion

        public int AddCurso(Categoria curso)
        {
            return Insert(curso);
        }

        public IEnumerable<Categoria> GetCursos(int idevento)
        {
            return from c in Table<Categoria>()
                   where c.idevento==idevento
                   select c;
        }
```

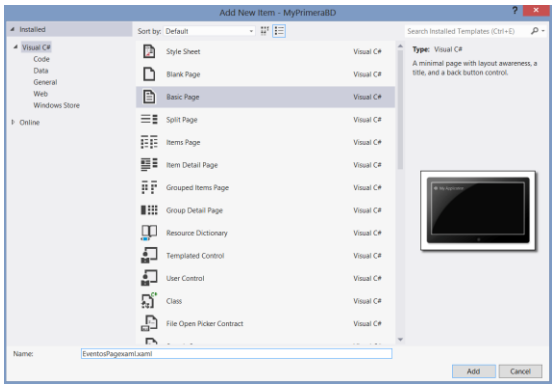
Si abrimos el explorador de nuestra solución, finalmente quedaría de la siguiente manera:



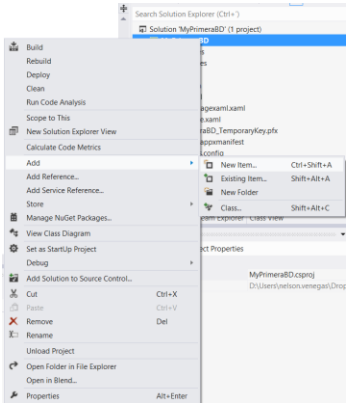
Diseño de la aplicación en XAML:

En esta sección aprenderemos a definir el diseño de la aplicación en XAML, agregaremos algunos controles y visualizaremos la información del calendario. Trataremos conceptos básicos para la creación de la interfaz de usuario en XAML. Para ello crearemos un gestionador de calendario.

Ahora vamos a adicionar una nueva página en el proyecto, para ello damos click botón derecho add, y nos aparecerá un menú de la siguiente manera:



Seleccionamos WindowsStore y luego adicionaremos una plantilla para una pagina básica y le damos el nombre EventosPagexaml.



En la pagina de eventospages.xaml vamos a colocar el siguiente código el cual crea todos los controles que vamos a tener en nuestra pagina inicial.

```
<common:LayoutAwarePage
    x:Name="pageRoot"
    x:Class="MyPrimeraBD.EventosPagexaml"
    DataContext="{Binding DefaultViewModel, RelativeSource={RelativeSource Self}}"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:MyPrimeraBD"
    xmlns:common="using:MyPrimeraBD.Common"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Page.Resources>

        <!-- TODO: Delete this line if the key AppName is declared in App.xaml -->
        <x:String x:Key="AppName">Calendario</x:String>
    </Page.Resources>

    <!--
        This grid acts as a root panel for the page that defines two rows:
        * Row 0 contains the back button and page title
        * Row 1 contains the rest of the page layout
    -->
    <Grid Style="{StaticResource LayoutRootStyle}">

        <Grid.RowDefinitions>
            <RowDefinition Height="140" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
```

```
        <!-- Button de Regreso y Titulo -->
        <Grid>
            <Grid.ColumnDefinitions>
               <ColumnDefinition Width="Auto"/>
               <ColumnDefinition Width="1*"/>
             </Grid.ColumnDefinitions>
            <Button x:Name="backButton" Click="GoBack" IsEnabled="{Binding
Frame.CanGoBack, ElementName=pageRoot}" Style="{StaticResource BackButtonStyle}"/>
            <TextBlock x:Name="pageTitle" Grid.Column="1" Text="{StaticResource AppName}"
Style="{StaticResource PageHeaderTextStyle}"/>
        </Grid>

        <Grid Grid.Row="1">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="2*" MinWidth="320" />
                <ColumnDefinition Width="3*" />
            </Grid.ColumnDefinitions>

            <StackPanel Margin="120,0,0,0"  >
                <TextBlock>Nombres</TextBlock>
                <TextBox x:Name="txtNombre" ></TextBox>
                <TextBlock>Fecha</TextBlock>
                <TextBox x:Name="txtFecha"></TextBox>
                <StackPanel Orientation="Horizontal">
                    <Button x:Name="btnGuardar" Click="btnGuardar_Click_1"
Width="100">Guardar</Button>
                    <Button x:Name="btnConsultar" Click="btnConsultar_Click_1"
Width="100">Consultar</Button>
                </StackPanel>
                <StackPanel Orientation="Horizontal">
                    <Button x:Name="btnEliminar" Click="btnEliminar_Click_1"
Width="100">Eliminar</Button>
                    <Button x:Name="btnEditar" Click="btnEditar_Click_1"
Width="100">Editar</Button>
                    <Button x:Name="btnVerificar" Click="btnVerificar_Click_1"
Width="100">Verificar</Button>
                </StackPanel>
            </StackPanel>


            <ListView x:Name="lstEventos" Grid.Column="1" Margin="173,0,0,0" >

                <ListView.ItemTemplate>
                    <DataTemplate>
                        <StackPanel>
                            <TextBlock Text="{Binding idevento}"></TextBlock>
                            <TextBlock Text="{Binding Nombres}"></TextBlock>
                            <TextBlock Text="{Binding Fecha}"></TextBlock>
                        </StackPanel>
                    </DataTemplate>
                </ListView.ItemTemplate>

            </ListView>
        </Grid>
        <VisualStateManager.VisualStateGroups>

            <!-- Visual states reflect the application's view state -->
```

```
            <VisualStateGroup x:Name="ApplicationViewStates">
                <VisualState x:Name="FullScreenLandscape"/>
                <VisualState x:Name="Filled"/>

                <!-- The entire page respects the narrower 100-pixel margin convention
for portrait -->
                <VisualState x:Name="FullScreenPortrait">
                    <Storyboard>
                        <ObjectAnimationUsingKeyFrames Storyboard.TargetName="backButton"
Storyboard.TargetProperty="Style">
                            <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource
PortraitBackButtonStyle}"/>
                        </ObjectAnimationUsingKeyFrames>
                    </Storyboard>
                </VisualState>

                <!-- The back button and title have different styles when snapped -->
                <VisualState x:Name="Snapped">
                    <Storyboard>
                        <ObjectAnimationUsingKeyFrames Storyboard.TargetName="backButton"
Storyboard.TargetProperty="Style">
                            <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource
SnappedBackButtonStyle}"/>
                        </ObjectAnimationUsingKeyFrames>
                        <ObjectAnimationUsingKeyFrames Storyboard.TargetName="pageTitle"
Storyboard.TargetProperty="Style">
                            <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource
SnappedPageHeaderTextStyle}"/>
                        </ObjectAnimationUsingKeyFrames>
                    </Storyboard>
                </VisualState>
            </VisualStateGroup>
        </VisualStateManager.VisualStateGroups>
    </Grid>
</common:LayoutAwarePage>
```

De igual forma en la clase EventosPage.xaml.cs y vamos a colocar el código de cada uno de los botones que tenemos en la pagina inicial.

```
namespace MyPrimeraBD
{
    /// <summary>
    /// A basic page that provides characteristics common to most applications.
    /// </summary>
    public sealed partial class EventosPagexaml : MyPrimeraBD.Common.LayoutAwarePage
    {
        public EventosPagexaml()
        {
            this.InitializeComponent();
        }

        /// <summary>
```

```
        /// Populates the page with content passed during navigation.  Any saved state is
also
        /// provided when recreating a page from a prior session.
        /// </summary>
        /// <param name="navigationParameter">The parameter value passed to
        /// <see cref="Frame.Navigate(Type, Object)"/> when this page was initially
requested.
        /// </param>
        /// <param name="pageState">A dictionary of state preserved by this page during
an earlier
        /// session.  This will be null the first time a page is visited.</param>
        protected override void LoadState(Object navigationParameter, Dictionary<String,
Object> pageState)
        {
        }

        /// <summary>
        /// Preserves state associated with this page in case the application is
suspended or the
        /// page is discarded from the navigation cache.  Values must conform to the
serializationon
        /// requirements of <see cref="SuspensionManager.SessionState"/>.
        /// </summary>
        /// <param name="pageState">An empty dictionary to be populated with serializable
state.</param>
        protected override void SaveState(Dictionary<String, Object> pageState)
        {
        }

        private void btnGuardar_Click_1(object sender, RoutedEventArgs e)
        {
            Eventos evento = new Eventos();
            evento.Nombres = this.txtNombre.Text;
            evento.Fecha = Convert.ToDateTime(this.txtFecha.Text);
            var rutaBD =
Path.Combine(Windows.Storage.ApplicationData.Current.LocalFolder.Path,
"MisEventos.sqlite");
            Contexto bd = new Contexto(rutaBD);
            int idevento = bd.Addevento(evento);
            this.txtFecha.Text = "";
            this.txtNombre.Text = "";


        }

        private void btnConsultar_Click_1(object sender, RoutedEventArgs e)
        {
            var rutaBD =
Path.Combine(Windows.Storage.ApplicationData.Current.LocalFolder.Path,
"MisEventos.sqlite");
            Contexto bd = new Contexto(rutaBD);
            this.lstEventos.ItemsSource=bd.GetEventos();
        }

        private void btnEliminar_Click_1(object sender, RoutedEventArgs e)
        {
            Eventos evento = (Eventos)lstEventos.SelectedItem;
```

```
            var rutaBD =
Path.Combine(Windows.Storage.ApplicationData.Current.LocalFolder.Path,
"MisEventos.sqlite");
            Contexto bd = new Contexto(rutaBD);
            bd.Deleteevento(evento.idEvento);
        }

        private void btnEditar_Click_1(object sender, RoutedEventArgs e)
        {

        }

        private void btnVerificar_Click_1(object sender, RoutedEventArgs e)
        {
            var rutaBD =
Path.Combine(Windows.Storage.ApplicationData.Current.LocalFolder.Path,
"MisEventos.sqlite");
            Contexto bd = new Contexto(rutaBD);
            List<Eventos> cumple = new List<Eventos>();
            foreach (Eventos item in  bd.GetEventos().ToList())
            {
                if (item.Fecha.ToString("M/d/yyyy") == DateTime.Now.ToString("M/d/yyyy"))
                {
                    cumple.Add(item);
                }
            }
            this.lstEventos.ItemsSource = cumple;
        }
    }
```

Visualizacion de la aplicacion

Con esta información podemos ejecutar nuestra aplicación y nos mostrara la siguiente ventana.

Cada uno de los botones realizan el crud del evento creado y adicional a ello si doy click en verificar, lo que realiza el software es revisar que tareas se realizan el dia actual.

Declaración de privacidad.

Para poder subir la apps al Windows Store es necesario declara el uso que realiza la aplicación con respecto a la gestión de archivos, uso de internet, micrófono, cámara etc.

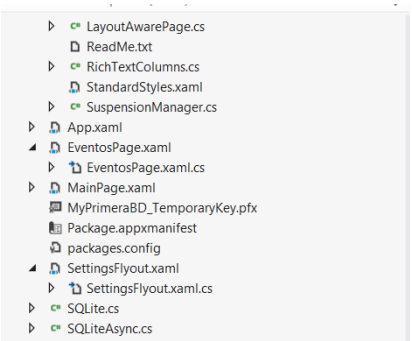
Para ellos debemos crear una pagina llamada SettingsFlyout la cual contiene la siguiente infomacion:

```
<Page
    x:Class=" MyPrimeraBD.SettingsFlyout"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using: MyPrimeraBD "
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid>
        <Grid.Background>
            <ImageBrush ImageSource="Assets/background.jpg" Stretch="None" />
        </Grid.Background>
        <StackPanel>


        <TextBlock Margin="5,30,0,30" Style="{StaticResource SubheaderTextStyle}" >App
desarrollada por:</TextBlock>
            <TextBlock Margin="5,0,0,0" FontSize="24" Style="{StaticResource
SubheaderTextStyle}" >Nombre</TextBlock>
            <TextBlock Margin="5,0,0,0"  FontSize="24" Style="{StaticResource
SubheaderTextStyle}" >Empresa</TextBlock>
            <TextBlock  Margin="5,0,0,0" FontSize="24" Style="{StaticResource
SubheaderTextStyle}" >Ciudad</TextBlock>
            <TextBlock  Margin="5,0,0,0" FontSize="24" Style="{StaticResource
SubheaderTextStyle}" >email</TextBlock>
            <TextBlock  Margin="5,0,0,0" FontSize="24" Style="{StaticResource
SubheaderTextStyle}" >pagina WEB</TextBlock>
```

```
            <TextBlock Margin="5,30,0,30" Style="{StaticResource SubheaderTextStyle}"
>Política de privacidad</TextBlock>
            <TextBlock  Margin="5,0,0,0" FontSize="18" Style="{StaticResource
SubheaderTextStyle}" >La aplicación (nombre de la Aplicaicon) hace uso de la conexión a
internet, única y exclusivamente, para descargar información relativa de la misma desde
nuestros servidores.</TextBlock>
            <TextBlock  Margin="5,0,0,0" FontSize="18" Style="{StaticResource
SubheaderTextStyle}" >Ninguna información de la máquina o del usuario es enviada de
vuelta a ellos ni a ningún otro servidor a través de internet ni de ningún otro medio por
parte de esta aplicación.</TextBlock>
            <TextBlock  Margin="5,0,0,0" FontSize="18" Style="{StaticResource
SubheaderTextStyle}" >Es responsabilidad del usuario el manejo que este haga de los datos
que son registrados a través de la aplicación.</TextBlock>

        </StackPanel>
    </Grid>
</Page>
```

```
namespace MyPrimeraBD
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class SettingsFlyout : Page
    {
        public SettingsFlyout()
        {
            this.InitializeComponent();
        }

        /// <summary>
        /// Invoked when this page is about to be displayed in a Frame.
        /// </summary>
        /// <param name="e">Event data that describes how this page was reached.  The
Parameter
        /// property is typically used to configure the page.</param>
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
        }
    }
}
```

Si vemos el explorador de proyectos vemos que la clase queda de la siguiente manera:



Para finalizar y poder relizar el contrato desde configuración y poder visualizar el contrato de confidencialidad se deben agregar alguno elementos a la clase que corresponde a lpagina que da inicio a la aplicación, en nuestro caso `EventosPagexaml.cs, se deben agregar varios espacios de nombre para lo cual acontinuacion especifico el contenido general de la clase.`

```
using MyPrimeraBD.Common.Data;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using Windows.Foundation;
using Windows.UI.ApplicationSettings;
using Windows.UI.Popups;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Media.Animation;


// The Basic Page item template is documented at
http://go.microsoft.com/fwlink/?LinkId=234237

namespace MyPrimeraBD
{
    /// <summary>
    /// A basic page that provides characteristics common to most applications.
    /// </summary>
    public sealed partial class EventosPagexaml : MyPrimeraBD.Common.LayoutAwarePage
    {

        private bool isEventRegistered;

        // Used to determine the correct height to ensure our cust-om UI fills the
screen.
        private Rect windowBounds;

        // Desired width for the settings UI. UI guidelines specify this should be 346 or
646 depending on your needs.
        private double settingsWidth = 646;
```

```
        // This is the container that will hold our custom content.
        private Popup settingsPopup;


        public EventosPagexaml()
        {
            this.InitializeComponent();
             windowBounds = Window.Current.Bounds;

            // Added to listen for events when the window size is updated.
             Window.Current.SizeChanged += OnWindowSizeChanged;

            //Evento de lanzamiento de settings
            //rootPage.NotifyUser("Defaults command added", NotifyType.StatusMessage);
            if (!this.isEventRegistered)
            {
                // Listening for this event lets the app initialize the settings commands
and pause its UI until the user closes the pane.
                // To ensure your settings are available at all times in your app, place
your CommandsRequested handler in the overridden
                // OnWindowCreated of App.xaml.cs
                SettingsPane.GetForCurrentView().CommandsRequested +=
EventosPagexaml_CommandsRequested;
                this.isEventRegistered = true;
            }
        }

        void EventosPagexaml_CommandsRequested(SettingsPane sender,
SettingsPaneCommandsRequestedEventArgs args)
        {
            UICommandInvokedHandler handler = new
UICommandInvokedHandler(onSettingsCommand);

            SettingsCommand generalCommand = new SettingsCommand("DefaultsId", "Acerca
de", handler);
            args.Request.ApplicationCommands.Add(generalCommand);
        }

        private void onSettingsCommand(IUICommand command)
        {
            //rootPage.NotifyUser("Defaults command invoked", NotifyType.StatusMessage);

            // Create a Popup window which will contain our flyout.
            settingsPopup = new Popup();
            settingsPopup.Closed += OnPopupClosed;
            Window.Current.Activated += OnWindowActivated;
            settingsPopup.IsLightDismissEnabled = true;
            settingsPopup.Width = settingsWidth;
            settingsPopup.Height = windowBounds.Height;

            // Add the proper animation for the panel.
            settingsPopup.ChildTransitions = new TransitionCollection();
            settingsPopup.ChildTransitions.Add(new PaneThemeTransition()
            {
                Edge = (SettingsPane.Edge == SettingsEdgeLocation.Right) ?
                       EdgeTransitionLocation.Right :
                       EdgeTransitionLocation.Left
            });
```

```
            // Create a SettingsFlyout the same dimenssions as the Popup.
            SettingsFlyout mypane = new SettingsFlyout();
            mypane.Width = settingsWidth;
            mypane.Height = windowBounds.Height;

            // Place the SettingsFlyout inside our Popup window.
            settingsPopup.Child = mypane;

            // Let's define the location of our Popup.
            settingsPopup.SetValue(Canvas.LeftProperty, SettingsPane.Edge ==
SettingsEdgeLocation.Right ? (windowBounds.Width - settingsWidth) : 0);
            settingsPopup.SetValue(Canvas.TopProperty, 0);
            settingsPopup.IsOpen = true;
        }


        void OnWindowSizeChanged(object sender,
Windows.UI.Core.WindowSizeChangedEventArgs e)
        {
            windowBounds = Window.Current.Bounds;
        }


        /// <summary>
        /// We use the window's activated event to force closing the Popup since a user
maybe interacted with
        /// something that didn't normally trigger an obvious dismiss.
        /// </summary>
        /// <param name="sender">Instance that triggered the event.</param>
        /// <param name="e">Event data describing the conditions that led to the
event.</param>
        private void OnWindowActivated(object sender,
Windows.UI.Core.WindowActivatedEventArgs e)
        {
            if (e.WindowActivationState ==
Windows.UI.Core.CoreWindowActivationState.Deactivated)
            {
                settingsPopup.IsOpen = false;
            }
        }


        /// <summary>
        /// When the Popup closes we no longer need to monitor activation changes.
        /// </summary>
        /// <param name="sender">Instance that triggered the event.</param>
        /// <param name="e">Event data describing the conditions that led to the
event.</param>
        void OnPopupClosed(object sender, object e)
        {
            Window.Current.Activated -= OnWindowActivated;
        }



        /// <summary>
```

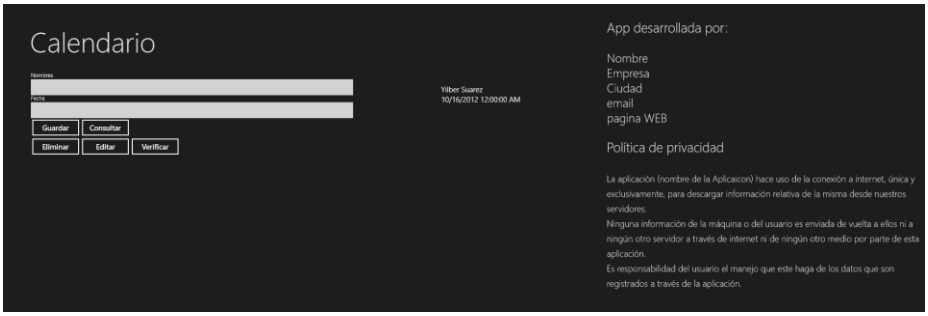
```
        /// Populates the page with content passed during navigation.  Any saved state is
also
        /// provided when recreating a page from a prior session.
        /// </summary>
        /// <param name="navigationParameter">The parameter value passed to
        /// <see cref="Frame.Navigate(Type, Object)"/> when this page was initially
requested.
        /// </param>
        /// <param name="pageState">A dictionary of state preserved by this page during
an earlier
        /// session.  This will be null the first time a page is visited.</param>
        protected override void LoadState(Object navigationParameter, Dictionary<String,
Object> pageState)
        {
        }

        /// <summary>
        /// Preserves state associated with this page in case the application is
suspended or the
        /// page is discarded from the navigation cache.  Values must conform to the
serializationon
        /// requirements of <see cref="SuspensionManager.SessionState"/>.
        /// </summary>
        /// <param name="pageState">An empty dictionary to be populated with serializable
state.</param>
        protected override void SaveState(Dictionary<String, Object> pageState)
        {
        }

        private void btnGuardar_Click_1(object sender, RoutedEventArgs e)
        {
            Eventos evento = new Eventos();
            evento.Nombres = this.txtNombre.Text;
            evento.Fecha = Convert.ToDateTime(this.txtFecha.Text);
            var rutaBD =
Path.Combine(Windows.Storage.ApplicationData.Current.LocalFolder.Path,
"MisEventos.sqlite");
            Contexto bd = new Contexto(rutaBD);
            int idevento = bd.Addevento(evento);
            this.txtFecha.Text = "";
            this.txtNombre.Text = "";


        }

        private void btnConsultar_Click_1(object sender, RoutedEventArgs e)
        {
            var rutaBD =
Path.Combine(Windows.Storage.ApplicationData.Current.LocalFolder.Path,
"MisEventos.sqlite");
            Contexto bd = new Contexto(rutaBD);
            this.lstEventos.ItemsSource=bd.GetEventos();
        }

        private void btnEliminar_Click_1(object sender, RoutedEventArgs e)
        {
            Eventos evento = (Eventos)lstEventos.SelectedItem;
```

```
            var rutaBD =
Path.Combine(Windows.Storage.ApplicationData.Current.LocalFolder.Path,
"MisEventos.sqlite");
            Contexto bd = new Contexto(rutaBD);
            bd.Deleteevento(evento.idEvento);
        }

        private void btnEditar_Click_1(object sender, RoutedEventArgs e)
        {

        }

        private void btnVerificar_Click_1(object sender, RoutedEventArgs e)
        {
            var rutaBD =
Path.Combine(Windows.Storage.ApplicationData.Current.LocalFolder.Path,
"MisEventos.sqlite");
            Contexto bd = new Contexto(rutaBD);
            List<Eventos> cumple = new List<Eventos>();
            foreach (Eventos item in  bd.GetEventos().ToList())
            {
                if (item.Fecha.ToString("M/d/yyyy") == DateTime.Now.ToString("M/d/yyyy"))
                {
                    cumple.Add(item);
                }
            }
            this.lstEventos.ItemsSource = cumple;
        }


    }
}
```

Si vemos al ejecutar la aplicación nos aparecerá lo siguiente:

Al dar click sobre configuración aparece un página desplegable de la siguiente forma:

Finalmente si presionamos Acerca de nos aparecerá la declaración de privacidad la cual es obligatoria para poder subir la aplicación.



La presente guía es una base para la creación de aplicaciones al estilo WinRT Windows Store, pueden construirse a partir de ésta varias aplicaciones con diferentes grados de complejidad, para mayor información les invitamos a inscribirse en el curso correspondiente en http://www.microsoftvirtualacademy.com y revisar el sitio http://msdn.microsoft.com/es-ES/windows/apps