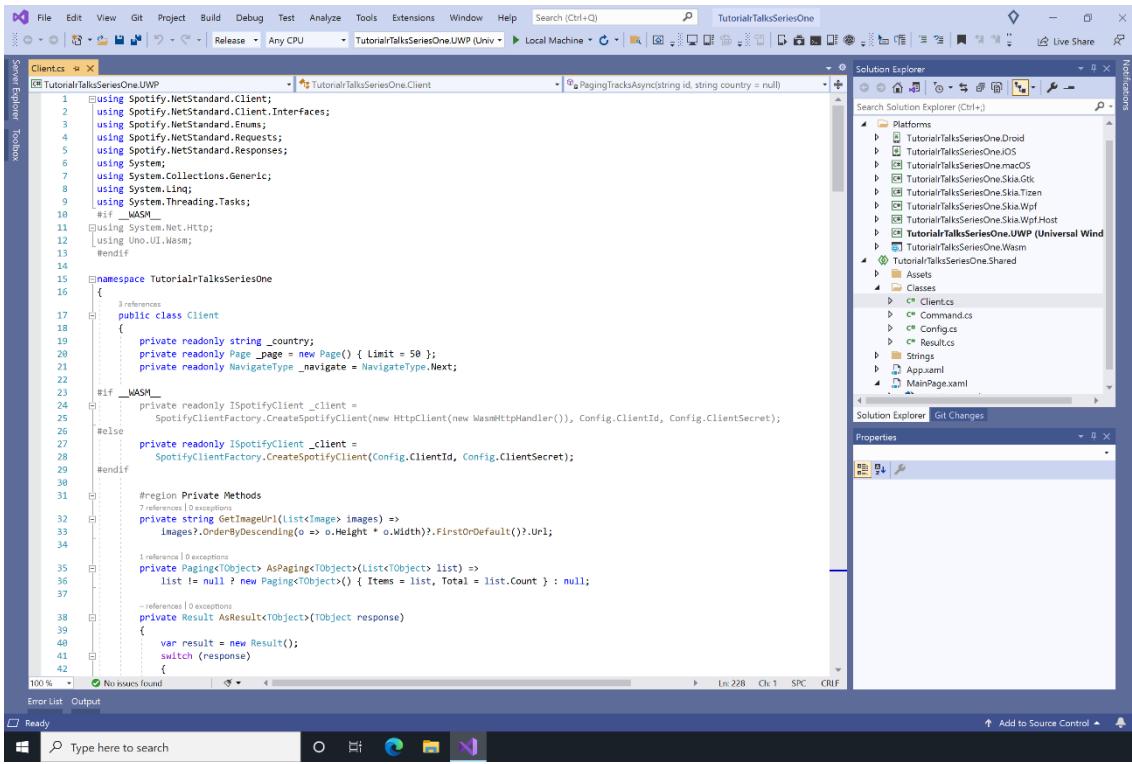


# Demo

# Visual Studio 2019

The Demo uses Visual Studio 2019 with the Uno Platform Solution Templates, Spotify for Developers with Spotify.NetStandard and System Icons from the Fluent Design System.

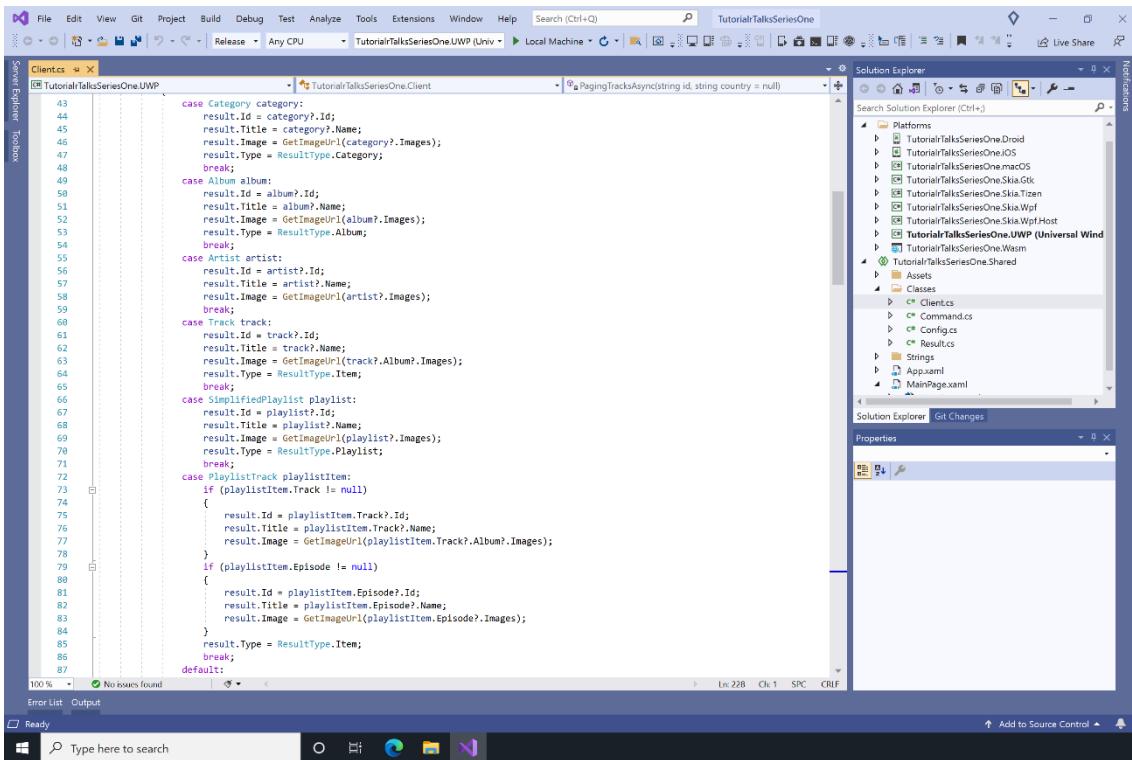
## Client



A screenshot of the Visual Studio 2019 IDE. The main window shows the code for the `Client` class in the `TutorialTalksSeriesOne.UWP` project. The code uses the Spotify for Developers NuGet package (`Spotify.NetStandard`) and HttpClient for WebAssembly. It includes methods for getting image URLs and performing paging operations. The Solution Explorer on the right shows the project structure, including platforms like Droid, iOS, macOS, Skia.Gtk, Skia.Tizen, Skia.Wpf, Skia.Wpf.Host, and UWP. The Properties and Git Changes tabs are also visible.

```
1 //using Spotify.NetStandard.Client;
2 //using Spotify.NetStandard.Client.Interfaces;
3 //using Spotify.NetStandard.Enums;
4 //using Spotify.NetStandard.Requests;
5 //using Spotify.NetStandard.Responses;
6 //using System;
7 //using System.Collections.Generic;
8 //using System.Linq;
9 //using System.Threading.Tasks;
10 //using Wasm;
11 //using System.Net.Http;
12 //using Uno.UI.Wasm;
13 //endif
14
15 namespace TutorialTalksSeriesOne
16 {
17     public class Client
18     {
19         private readonly string _country;
20         private readonly Page _page = new Page { Limit = 50 };
21         private readonly NavigateType _navigate = NavigateType.Next;
22
23         #if WASM_
24         private readonly ISpotifyClient _client =
25             SpotifyClientFactory.CreateSpotifyClient(new HttpClient(new WasmHttpHandler()), Config.ClientId, Config.ClientSecret);
26         #else
27         private readonly ISpotifyClient _client =
28             SpotifyClientFactory.CreateSpotifyClient(Config.ClientId, Config.ClientSecret);
29         #endif
30
31         #region Private Methods
32         #region Exceptions
33         private string GetImageUrl(List<Image> images) =>
34             images?.OrderByDescending(o => o.Height * o.Width)?.FirstOrDefault()?.Url;
35
36         private Paging<Object> AsPaging<Object>(List<Object> list) =>
37             list != null ? new Paging<Object>() { Items = list, Total = list.Count } : null;
38
39         private Result AsResult<ToObject>(ToObject response)
40         {
41             var result = new Result();
42             switch (response)
43             {
44                 case Category category:
45                     result.Id = category?.Id;
46                     result.Title = category?.Name;
47                     result.Image = GetImageUrl(category?.Images);
48                     result.Type = ResultType.Category;
49                     break;
50                 case Album album:
51                     result.Id = album?.Id;
52                     result.Title = album?.Name;
53                     result.Image = GetImageUrl(album?.Images);
54                     result.Type = ResultType.Album;
55                     break;
56                 case Artist artist:
57                     result.Id = artist?.Id;
58                     result.Title = artist?.Name;
59                     result.Image = GetImageUrl(artist?.Images);
60                     break;
61                 case Track track:
62                     result.Id = track?.Id;
63                     result.Title = track?.Name;
64                     result.Image = GetImageUrl(track?.Album?.Images);
65                     result.Type = ResultType.Item;
66                     break;
67                 case SimplifiedPlaylist playlist:
68                     result.Id = playlist?.Id;
69                     result.Title = playlist?.Name;
70                     result.Image = GetImageUrl(playlist?.Images);
71                     result.Type = ResultType.Playlist;
72                     break;
73                 case PlaylistTrack playlistItem:
74                     if (playlistItem.Track != null)
75                     {
76                         result.Id = playlistItem.Track?.Id;
77                         result.Title = playlistItem.Track?.Name;
78                         result.Image = GetImageUrl(playlistItem.Track?.Album?.Images);
79                     }
80                     if (playlistItem.Episode != null)
81                     {
82                         result.Id = playlistItem.Episode?.Id;
83                         result.Title = playlistItem.Episode?.Name;
84                         result.Image = GetImageUrl(playlistItem.Episode?.Images);
85                     }
86                     result.Type = ResultType.Item;
87                     break;
88                 default:
89             }
90         }
91
92         #endregion Exceptions
93     }
94 }
```

The Client Class uses the third-party Spotify for Developers NuGet Package Spotify.NetStandard and supports HttpClient for WebAssembly along with Image and Paging Methods.



A screenshot of the Visual Studio 2019 IDE showing the continuation of the `Client` class code. It includes a large switch statement for handling different types of results (Category, Album, Artist, Track, Playlist, etc.) and their corresponding properties like Id, Title, Image, and Type. The Solution Explorer and other UI elements are visible on the right.

```
43         case Category category:
44             result.Id = category?.Id;
45             result.Title = category?.Name;
46             result.Image = GetImageUrl(category?.Images);
47             result.Type = ResultType.Category;
48             break;
49         case Album album:
50             result.Id = album?.Id;
51             result.Title = album?.Name;
52             result.Image = GetImageUrl(album?.Images);
53             result.Type = ResultType.Album;
54             break;
55         case Artist artist:
56             result.Id = artist?.Id;
57             result.Title = artist?.Name;
58             result.Image = GetImageUrl(artist?.Images);
59             break;
60         case Track track:
61             result.Id = track?.Id;
62             result.Title = track?.Name;
63             result.Image = GetImageUrl(track?.Album?.Images);
64             result.Type = ResultType.Item;
65             break;
66         case SimplifiedPlaylist playlist:
67             result.Id = playlist?.Id;
68             result.Title = playlist?.Name;
69             result.Image = GetImageUrl(playlist?.Images);
70             result.Type = ResultType.Playlist;
71             break;
72         case PlaylistTrack playlistItem:
73             if (playlistItem.Track != null)
74             {
75                 result.Id = playlistItem.Track?.Id;
76                 result.Title = playlistItem.Track?.Name;
77                 result.Image = GetImageUrl(playlistItem.Track?.Album?.Images);
78             }
79             if (playlistItem.Episode != null)
80             {
81                 result.Id = playlistItem.Episode?.Id;
82                 result.Title = playlistItem.Episode?.Name;
83                 result.Image = GetImageUrl(playlistItem.Episode?.Images);
84             }
85             result.Type = ResultType.Item;
86             break;
87         default:
88             break;
89     }
90 }
```

Then, the Client Class also has a Method for converting the responses from the Spotify Client to the Result Class.

```

    88         throw new NotSupportedException();
    89     }
    90     result.Command = Action != null ? new CommandResult<Action>(Action) : null;
    91     return result;
    92 }
    93
    #endregion Private Methods
    94
    #region Paging Methods
    95
    96     private async Task<Paging<Category>> PagingCategoriesAsync(string country = null) =>
    97     {
    98         await _client.LookupAllCategoriesAsync(country, page: _page);
    99         return Categories;
    100    }
    101
    102    private async Task<Paging<Artist>> PagingArtistsAsync(string id) =>
    103    {
    104        await _client.LookupArtistRelatedArtistsAsync(id);
    105        return Artists;
    106    }
    107
    108    private async Task<Paging<Artist>> PagingArtistsAsync(Paging<Artist> paging) =>
    109    {
    110        await _client.NavigateAsync(paging, paging._navigate);
    111        return Artists;
    112    }
    113
    114    private async Task<Paging<Album>> PagingAlbumsAsync(string country = null) =>
    115    {
    116        await _client.LookupNewReleasesAsync(country, country, page: _page);
    117        return Albums;
    118    }
    119
    120    private async Task<Paging<Album>> PagingAlbumsAsync(Paging<Album> paging) =>
    121    {
    122        await _client.NavigateAsync(paging, paging._navigate);
    123        return Albums;
    124    }
    125
    126    private async Task<Paging<Track>> PagingTracksAsync(string id, string country = null) =>
    127    {
    128        await _client.LookupAsync<Track>(id, LookupType.AlbumTracks, country);
    129    }
    130
    131    private async Task<Paging<Track>> PagingTracksAsync(Paging<Track> paging) =>
    132    {
    133        await _client.PagingAsync(paging, paging._navigate);
    134    }
    135
    136    private async Task<Paging<SimplifiedPlaylist>> PagingPlaylistsAsync(string id = null, string country = null) =>
    137    {
    138        if (id != null)
    139        {
    140            await _client.LookupAsync<ContentResponse>(id, LookupType.CategoriesPlaylists, country, page: _page);
    141        }
    142        else
    143        {
    144            await _client.LookupFeaturedPlaylistsAsync(country, page: _page);
    145        }
    146    }
    147
    148    private async Task<Paging<TObject>> PagingAsync<TObject>(string id = null, string country = null)
    149    {
    150        where TObject : class
    151    }
    152
    153    private async Task<Paging<TObject>> PagingAsync<TObject>(Paging<TObject> paging)
    154    {
    155        switch (typeof(TObject))
    156        {
    157            case Type category when category == typeof(Category):
    158                return await PagingCategoriesAsync(country) as Paging<TObject>;
    159            case Type artist when artist == typeof(Artist):
    160                return await PagingArtistsAsync(id) as Paging<TObject>;
    161            case Type album when album == typeof(Album):
    162                return await PagingAlbumsAsync(country) as Paging<TObject>;
    163            case Type track when track == typeof(Track):
    164                return await PagingTracksAsync(id, country) as Paging<TObject>;
    165            case Type playlist when playlist == typeof(SimplifiedPlaylist):
    166                return await PagingPlaylistsAsync(id, country) as Paging<TObject>;
    167            case Type item when item == typeof(PlaylistTrack):
    168                return await PagingPlaylistItemsAsync(id, country) as Paging<TObject>;
    169            default:
    170                throw new NotSupportedException();
    171        }
    172    }
    173
    174    private async Task<Paging<TObject>> PagingAsync<TObject>(Paging<TObject> paging)
    175    {
    176        switch (typeof(TObject))
    177        {
    178            case Type category when category == typeof(Category):
    179                return await PagingCategoriesAsync(paging as Paging<Category>) as Paging<TObject>;
    180            case Type artist when artist == typeof(Artist):
    181                return await PagingArtistsAsync(paging as Paging<Artist>) as Paging<TObject>;
    182            case Type album when album == typeof(Album):
    183                return await PagingAlbumsAsync(paging as Paging<Album>) as Paging<TObject>;
    184            case Type track when track == typeof(Track):
    185                return await PagingTracksAsync(paging as Paging<Track>) as Paging<TObject>;
    186            case Type playlist when playlist == typeof(SimplifiedPlaylist):
    187                return await PagingPlaylistsAsync(paging as Paging<SimplifiedPlaylist>) as Paging<TObject>;
    188            default:
    189                throw new NotSupportedException();
    190        }
    191    }

```

Next, the Client Class has Methods for getting Categories, Artists, Albums, Tracks and Playlist Items from the Spotify Client Paging Methods.

```

    126
    127
    128
    129    private async Task<Paging<SimplifiedPlaylist>> PagingPlaylistsAsync(Paging<SimplifiedPlaylist> paging) =>
    130    {
    131        await _client.NavigateAsync(paging, paging._navigate);
    132        return Playlists;
    133    }
    134
    135
    136    private async Task<Paging<PlaylistTrack>> PagingPlaylistItemsAsync(string id, string country = null) =>
    137    {
    138        await _client.LookupPlaylistItemsAsync(id, country, page: _page);
    139    }
    140
    141    private async Task<Paging<PlaylistTrack>> PagingPlaylistItemsAsync(Paging<PlaylistTrack> paging) =>
    142    {
    143        await _client.PagingAsync(paging, paging._navigate);
    144    }
    145
    146
    147    private async Task<Paging<TObject>> PagingAsync<TObject>(string id = null, string country = null)
    148    {
    149        where TObject : class
    150    }
    151
    152    private async Task<Paging<TObject>> PagingAsync<TObject>(Paging<TObject> paging)
    153    {
    154        switch (typeof(TObject))
    155        {
    156            case Type category when category == typeof(Category):
    157                return await PagingCategoriesAsync(country) as Paging<TObject>;
    158            case Type artist when artist == typeof(Artist):
    159                return await PagingArtistsAsync(id) as Paging<TObject>;
    160            case Type album when album == typeof(Album):
    161                return await PagingAlbumsAsync(country) as Paging<TObject>;
    162            case Type track when track == typeof(Track):
    163                return await PagingTracksAsync(id, country) as Paging<TObject>;
    164            case Type playlist when playlist == typeof(SimplifiedPlaylist):
    165                return await PagingPlaylistsAsync(id, country) as Paging<TObject>;
    166            case Type item when item == typeof(PlaylistTrack):
    167                return await PagingPlaylistItemsAsync(id, country) as Paging<TObject>;
    168            default:
    169                throw new NotSupportedException();
    170        }
    171    }
    172
    173    private async Task<Paging<TObject>> PagingAsync<TObject>(Paging<TObject> paging)
    174    {
    175        switch (typeof(TObject))
    176        {
    177            case Type category when category == typeof(Category):
    178                return await PagingCategoriesAsync(paging as Paging<Category>) as Paging<TObject>;
    179            case Type artist when artist == typeof(Artist):
    180                return await PagingArtistsAsync(paging as Paging<Artist>) as Paging<TObject>;
    181            case Type album when album == typeof(Album):
    182                return await PagingAlbumsAsync(paging as Paging<Album>) as Paging<TObject>;
    183            case Type track when track == typeof(Track):
    184                return await PagingTracksAsync(paging as Paging<Track>) as Paging<TObject>;
    185            case Type playlist when playlist == typeof(SimplifiedPlaylist):
    186                return await PagingPlaylistsAsync(paging as Paging<SimplifiedPlaylist>) as Paging<TObject>;
    187            default:
    188                throw new NotSupportedException();
    189        }
    190    }

```

Also, the Client Class has Generic Methods that wrap all the Paging Methods for Categories, Artists, Albums, Tracks and Playlist Items.

```

165     return await PagingArtistsAsync(paging as Paging<Artist>) as Paging<TOBJECT>;
166     case Type album when album == typeof(Album):
167         return await PagingAlbumsAsync(paging as Paging<Album>) as Paging<TOBJECT>;
168     case Type track when track == typeof(Track):
169         return await PagingTracksAsync(paging as Paging<Track>) as Paging<TOBJECT>;
170     case Type playlist when playlist == typeof(SimplifiedPlaylist):
171         return await PagingPlaylistsAsync(paging as Paging<SimplifiedPlaylist>) as Paging<TOBJECT>;
172     case Type item when item == typeof(PlaylistTrack):
173         return await PagingPlaylistItemsAsync(paging as Paging<PlaylistTrack>) as Paging<TOBJECT>;
174     default:
175         throw new NotSupportedException();
176     }
177 }
178 #endregion Paging Methods
179
180 #region List Methods
181 private async Task<List<TOBJECT>> ListAsync<TOBJECT>(string id = null, string country = null)
182     where TOBJECT : class
183 {
184     Paging<TOBJECT> paging = null;
185     var list = new List<TOBJECT>();
186     do
187     {
188         paging = paging == null ? await PagingAsync<TOBJECT>(id, country) : await PagingAsync(paging);
189         if (paging.Items != null)
190             paging.Items.ForEach(f => list.Add(f));
191     } while (paging?.Items != null);
192     return list;
193 }
194
195 #endregion List Methods
196
197 #region Public Methods
198 public Client(string country) =>
199     _country = country;
200
201 public async Task<List<Result>> ListCategoriesAsync() =>
202     await ListResultAsync<Category>(_country);
203
204
205
206

```

Plus, the Client Class has a Generic Method to List everything from the Paging Method. It also has another Generic Method to List everything as a List of Result Class and use this with the List Categories Method.

```

207     public async Task<List<Result>> ListArtistsAsync(string id) =>
208         await ListResultAsync<Artist>(id, _country);
209
210     public async Task<List<Result>> ListAlbumsAsync(string id = null) =>
211         await ListResultAsync<Album>(id, _country);
212
213     public async Task<List<Result>> ListTracksAsync(string id) =>
214         await ListResultAsync<Track>(id, _country);
215
216     public async Task<List<Result>> ListPlaylistsAsync(string id = null) =>
217         await ListResultAsync<SimplifiedPlaylist>(id, _country);
218
219     public Action<Result> Action { get; set; }
220
221 #endregion Public Methods
222
223 #region Public Properties
224 public Action<Result> Action { get; set; }
225 #endregion Public Properties
226
227
228

```

Finally, the Client Class has Methods to List Artists, Albums, Tracks, Playlists and Playlist Items as a List of Result Class along with an Action.

## Command

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar. The status bar at the bottom shows 'TutorialTalksSeriesOne' and other standard developer information.

The main code editor window displays the 'Command.cs' file under the 'TutorialTalksSeriesOne.UWP' project. The code implements the `ICommand` interface with methods `CanExecute` and `Execute`, and an event `CanExecuteChanged`.

```
1  using System;
2  using System.Windows.Input;
3
4  namespace TutorialTalksSeriesOne
5  {
6      public class Command<TAction> : ICommand
7      {
8          private readonly Action<TAction> _execute;
9          private readonly Predicate<bool> _canExecute;
10
11         public event EventHandler CanExecuteChanged;
12
13         public Command(Action<TAction> execute, bool canExecute)
14         {
15             _execute = execute ?? throw new ArgumentNullException("execute");
16             _canExecute = canExecute;
17         }
18
19         public void CanExecute(object parameter) =>
20             _canExecute == null || _canExecute((bool)parameter);
21
22         public void Execute(object parameter) =>
23             _execute((TAction)parameter);
24
25         public void RaiseCanExecuteChanged() =>
26             CanExecuteChanged?.Invoke(this, EventArgs.Empty);
27     }
28 }
```

The Solution Explorer on the right lists various platforms and files for the project, including 'TutorialTalksSeriesOne.UWP (Universal Windows)', 'TutorialTalksSeriesOne.Shared', and 'TutorialTalksSeriesOne.Wasm'. The Properties window is also visible.

The Command Class implements the Methods from the `ICommand` Interface to allow for Commanding to be used with the Action from the Client Class.

## Config

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a search bar. The status bar at the bottom shows 'TutorialTalksSeriesOne' and other standard developer information.

The main code editor window displays the 'Config.cs' file under the 'TutorialTalksSeriesOne.UWP' project. It defines a static class `Config` with properties `ClientId` and `ClientSecret`.

```
1  namespace TutorialTalksSeriesOne
2  {
3      public static class Config
4      {
5          public static string ClientId => "clientid";
6          public static string ClientSecret => "clientsecret";
7      }
8  }
```

The Solution Explorer on the right lists various platforms and files for the project, including 'TutorialTalksSeriesOne.UWP (Universal Windows)', 'TutorialTalksSeriesOne.Shared', and 'TutorialTalksSeriesOne.Wasm'. The Properties window is also visible.

The Config Class has the Client Id and Client Secret which can be obtained from the Spotify for Developers Dashboard and will allow the Client Class to use the Spotify Web API.

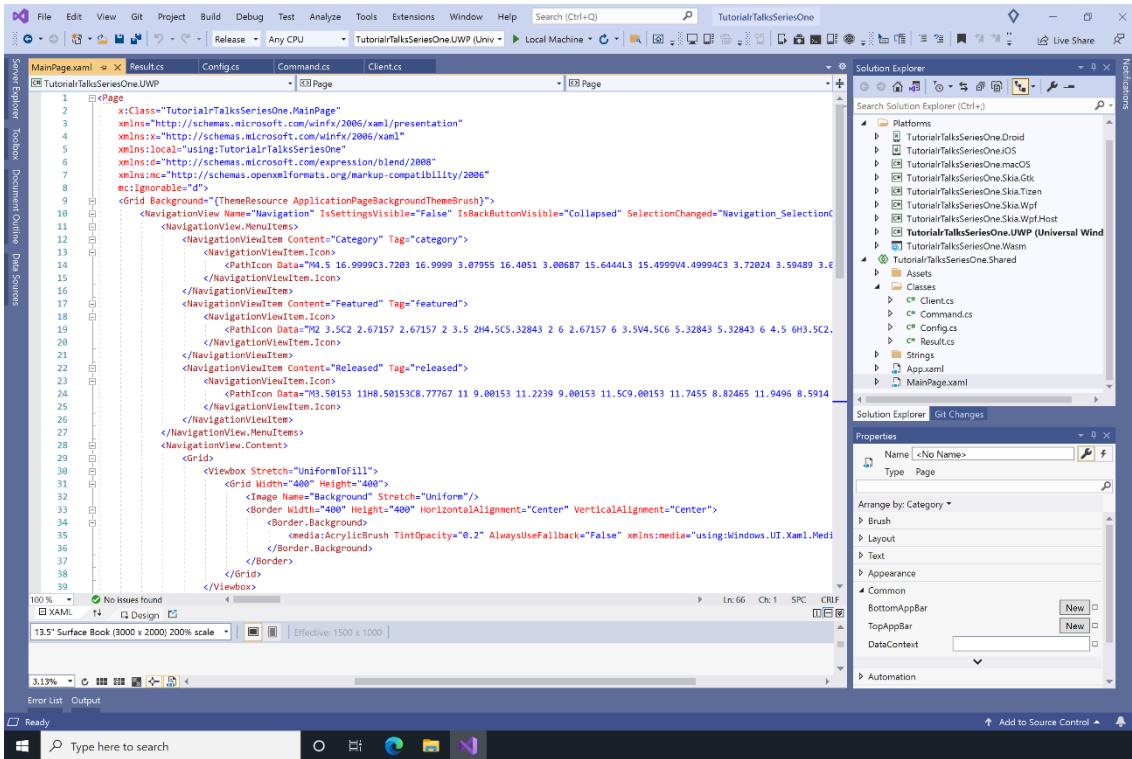
# Result

The screenshot shows the Microsoft Visual Studio interface. The code editor displays a C# file named 'Result.cs' from the 'TutorialTalksSeriesOne.UWP' project. The code defines a class 'Result' with properties for ResultType, Id, Title, Image, and Command. The Solution Explorer panel on the right lists various platforms and files for the project, including 'TutorialTalksSeriesOne.Droid', 'TutorialTalksSeriesOne.iOS', 'TutorialTalksSeriesOne.macOS', 'TutorialTalksSeriesOne.Skia.Gtk', 'TutorialTalksSeriesOne.Skia.Tizen', 'TutorialTalksSeriesOne.Skia.Wpf', 'TutorialTalksSeriesOne.Skia.Wpf.Host', 'TutorialTalksSeriesOne.UWP (Universal Wind...', 'TutorialTalksSeriesOne.Wasm', and 'TutorialTalksSeriesOne.Shared'. The Properties panel at the bottom is visible.

```
1  using System.Windows.Input;
2
3  namespace TutorialTalksSeriesOne
4  {
5      public enum ResultType
6      {
7          Category,
8          Playlist,
9          Album,
10         Item
11     }
12
13    public class Result
14    {
15        public ResultType Type { get; set; }
16
17        public string Id { get; set; }
18
19        public string Title { get; set; }
20
21        public string Image { get; set; }
22
23        public ICommand Command { get; set; }
24    }
25
26 }
```

The Result Class represents a Category, Playlist, Playlist Item, Album or Track and includes ResultType for Category, Playlist, Album, and Item which is used for Playlist Items or Tracks. It also has Properties for the Id, Title, Image and Command for Commanding.

# Main Page XAML

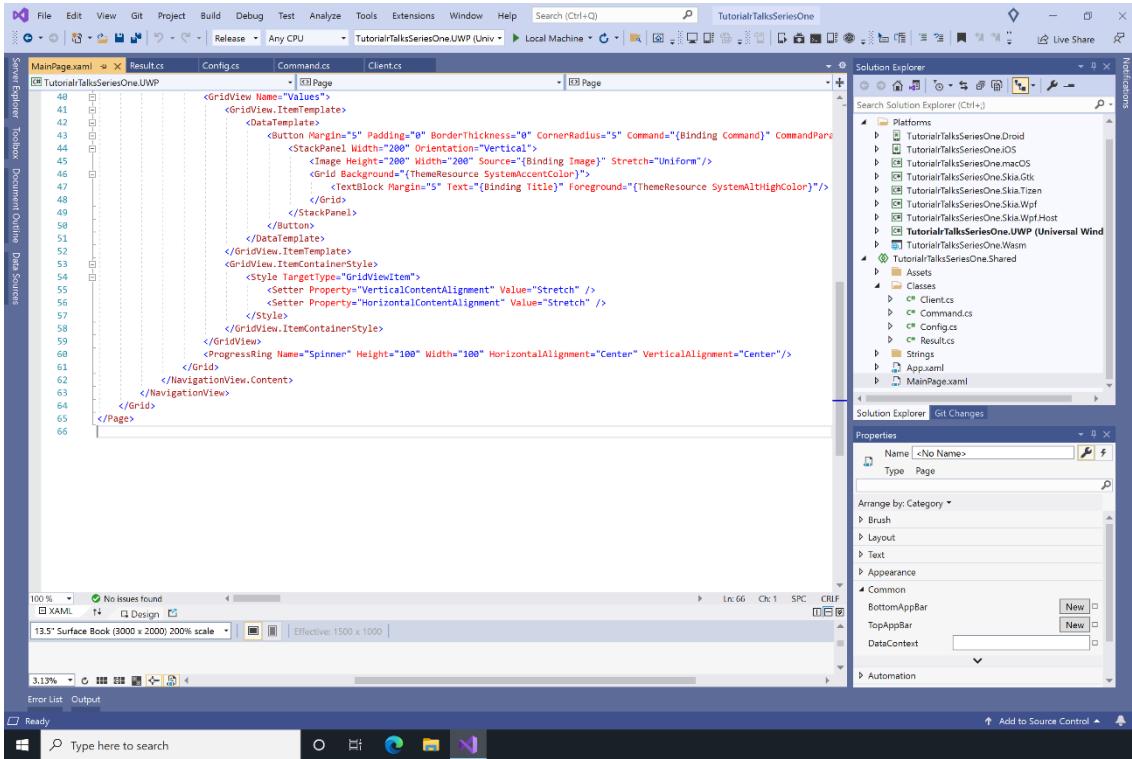


The screenshot shows the Visual Studio IDE with the MainPage.xaml file open. The code defines a NavigationView with three categories: Category, Featured, and Released. Each category has an associated Fluent Design System icon. The content area includes a Viewbox with a Grid and a Border containing an Acrylic Brush background. The properties pane on the right shows the page type as 'Page'.

```
<Page>
<x:Class>TutorialTalksSeriesOne.MainPage</x:Class>
< xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="using:TutorialTalksSeriesOne"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2008"
  mc:Ignorable="d">
<Grid Background="{ThemeResource ApplicationPageBackgroundBrush}">
    <NavigationView Name="NavigationView" IsSettingsVisible="False" IsBackButtonVisible="Collapsed" SelectionChanged="NavigationView_SelectionChanged">
        <NavigationView.MenuItems>
            <NavigationViewItem Content="Category" Tag="category">
                <NavigationViewItem.Icon>
                    <PathIcon Data="#M4.5 16.9999C3.7203 16.9999 3.07955 16.4051 3.08687 15.6444L3 15.4999V4.4999C3 3.72024 3.59489 3.60001L3 3.60001Z" />
                </NavigationViewItem.Icon>
            </NavigationViewItem>
            <NavigationViewItem Content="Featured" Tag="featured">
                <NavigationViewItem.Icon>
                    <PathIcon Data="#M2 3.5C2 2.67157 2.67157 2 3.5 2H4.5C5.32843 2 6 2.67157 6 3.5V4.5C6 5.32843 5.32843 6 4.5 H3.5C2.67157 2 2 2V4.5C5.32843 6 4.5 6H4.5C5.32843 6 4.5 6V4.5C5.32843 6 4.5 6Z" />
                </NavigationViewItem.Icon>
            </NavigationViewItem>
            <NavigationViewItem Content="Released" Tag="released">
                <NavigationViewItem.Icon>
                    <PathIcon Data="#M3.50153 11H8.50153C8.77767 11 9.00153 11.2239 9.00153 11.5C9.00153 11.7455 8.82465 11.9496 8.5914 12.5C9.00153 12.5 9.00153 12.5 9.00153 12.5V12.5C9.00153 12.5 9.00153 12.5 9.00153 12.5Z" />
                </NavigationViewItem.Icon>
            </NavigationViewItem>
        </NavigationView.MenuItems>
        <NavigationView.Content>
            <Grid>
                <Viewbox Stretch="UniformToFill">
                    <Grid Width="400" Height="400">
                        <Image Name="Background" Stretch="Uniform"/>
                        <Border Width="400" Height="400" HorizontalAlignment="Center" VerticalAlignment="Center">
                            <Border.Background>
                                <media:AcrylicBrush TintColor="#0.2" AlwaysUseFallback="False" xmlns:media="using:Windows.UI.Xaml.Media" />
                            </Border.Background>
                        </Border>
                    </Grid>
                </Viewbox>
            </Grid>
        </NavigationView.Content>
    </NavigationView>
</Grid>

```

MainPage includes XAML for a Navigation View with Category, Featured and Released options including use of Fluent Design System Icons. Content includes an Image with a Border using Acrylic Effect from the Fluent Design System for the Background.



The screenshot shows the Visual Studio IDE with the MainPage.xaml file open. The code defines a GridView with an ItemTemplate containing a Button and a TextBlock. The button has a vertical orientation and a rounded rectangle style. The properties pane on the right shows the page type as 'Page'.

```
<Page>
<x:Class>TutorialTalksSeriesOne.MainPage</x:Class>
< xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="using:TutorialTalksSeriesOne"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2008"
  mc:Ignorable="d">
<Grid ViewportHeight="1000" ViewportWidth="1000">
    <GridView Name="Values">
        <GridView.ItemTemplate>
            <DataTemplate>
                <Button Margin="5" Padding="0" BorderThickness="0" CornerRadius="5" Command="{Binding Command}" CommandParameter="{Binding Value}">
                    <StackPanel Width="200" Orientation="Vertical">
                        <Image Height="200" Width="200" Source="{Binding Image}" Stretch="Uniform"/>
                        <Grid Background="{ThemeResource SystemAccentColor}">
                            <TextBlock Margin="5" Text="{Binding Title}" Foreground="{ThemeResource SystemAltHighColor}"/>
                        </Grid>
                    </StackPanel>
                </DataTemplate>
            </GridView.ItemTemplate>
            <GridView.ItemContainerStyle>
                <Style TargetType="GridViewItem">
                    <Setter Property="VerticalContentAlignment" Value="Stretch" />
                    <Setter Property="HorizontalContentAlignment" Value="Stretch" />
                    <Style.Triggers>
                        <Trigger Property="IsSelected" Value="True">
                            <Setter Property="Background" Value="Red" />
                        </Trigger>
                    </Style.Triggers>
                </Style>
            </GridView.ItemContainerStyle>
        </GridView>
        <ProgressRing Name="Spinner" Height="100" Width="100" HorizontalAlignment="Center" VerticalAlignment="Center" />
    </Grid>
</Page>

```

MainPage also includes XAML for a GridView with an Item Template with a Button which has an Image and a TextBlock using a Fluent Design System Colour along with a Progress Ring to indicate Loading.

# Main Page Code

The screenshot shows the Visual Studio IDE with the MainPage.xaml.cs file open in the editor. The code implements a partial class MainPage that derives from Page. It contains methods for handling selection changes in a NavigationView and navigating results. The Client class is used to perform asynchronous operations like listing categories, playlists, and albums. The Spinner.IsActive property is used to determine the current state of the spinner.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using Windows.UI.Xaml.Controls;
5 using Windows.UI.Xaml.Media.Imaging;
6 using Windows.UI.Xaml.Navigation;
7
8 namespace TutorialTalksSeriesOne
9 {
10    public sealed partial class MainPage : Page
11    {
12        private readonly Client _client = new Client("GB");
13
14        private async void Navigation_SelectionChanged(NavigationView sender, NavigationViewSelectionChangedEventArgs args)
15        {
16            Spinner.IsActive = true;
17            var results = new List<Result>();
18            Background.Source = null;
19            switch (args.SelectedItemContainer.Tag.ToString())
20            {
21                case "category":
22                    results = await _client.ListCategoriesAsync();
23                    break;
24                case "featured":
25                    results = await _client.ListPlaylistsAsync();
26                    break;
27                case "released":
28                    results = await _client.ListAlbumsAsync();
29                    break;
30            }
31            Values.ItemsSource = results;
32            Spinner.IsActive = false;
33        }
34
35        private async void NavigateResult(Result result)
36        {
37            Spinner.IsActive = true;
38            switch (result.Type)
39            {
40                case ResultType.Category:
41                    Values.ItemsSource = await _client.ListPlaylistsAsync(result.Id);
42                    Background.Source = new BitmapImage(new Uri(result.Image));
43                    break;
44            }
45        }
46    }
47
48    public MainPage()
49    {
50        this.InitializeComponent();
51        _client.Action = (item) => NavigateResult(item);
52    }
53
54    protected override void OnNavigatedTo(NavigationEventArgs e)
55    {
56        Navigation.SelectedItem = Navigation.MenuItems.First();
57    }
58
59    protected override void OnNavigatedFrom(NavigationEventArgs e)
60    {
61    }
62
63    protected override void OnNavigatingFrom(NavigatingEventArgs e)
64    {
65        e.Cancel = true;
66    }
67}
```

The Main Page Class uses the Client Class, where the Country is set for Great Britain, there is an Event Handler for Selection Changed of the Navigation View to List Categories for Category, List Playlists for Featured or List Albums for Released plus a Method for Navigation Result.

This screenshot shows the continuation of the MainPage.xaml.cs code. It includes a switch statement for ResultType to handle Playlist, Album, and Category items. The Action property of the Client class is set to the NavigateResult method. The OnNavigatedTo event is overridden to set the Navigation.SelectedItem to the first item in the menu. The OnNavigatingFrom event is also overridden to cancel the navigation.

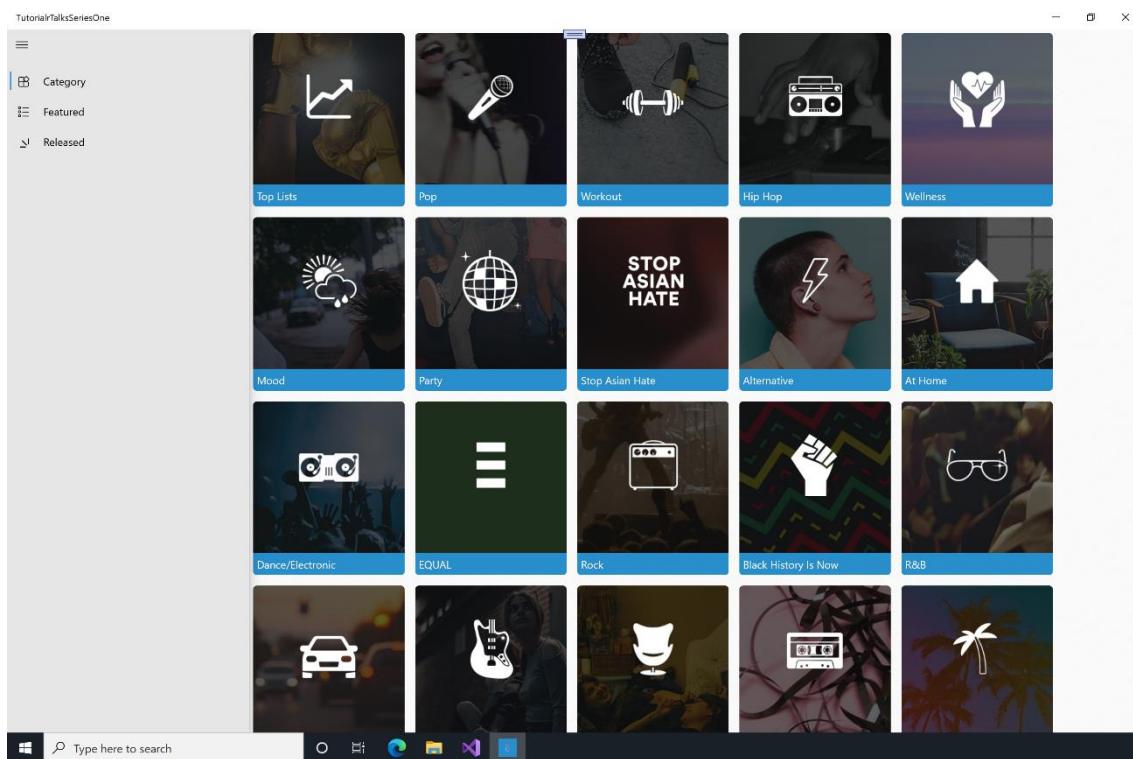
```
44    case ResultType.Playlist:
45        Values.ItemsSource = await _client.ListPlaylistItemsAsync(result.Id);
46        Background.Source = new BitmapImage(new Uri(result.Image));
47        break;
48    case ResultType.Album:
49        var results = await _client.ListTracksAsync(result.Id);
50        results.ForEach(f => f.Image = result.Image);
51        Values.ItemsSource = results;
52        Background.Source = new BitmapImage(new Uri(result.Image));
53        break;
54    }
55    Spinner.IsActive = false;
56}
57
58 protected override void OnNavigatedTo(NavigationEventArgs e)
59 {
60     Navigation.SelectedItem = Navigation.MenuItems.First();
61 }
62
63 protected override void OnNavigatingFrom(NavigatingEventArgs e)
64 {
65     e.Cancel = true;
66 }
```

The Main Page Class also sets the Action to the Navigation Result Method to List Playlists, Playlist Items or Tracks and set the Navigation View to the First option for Category.

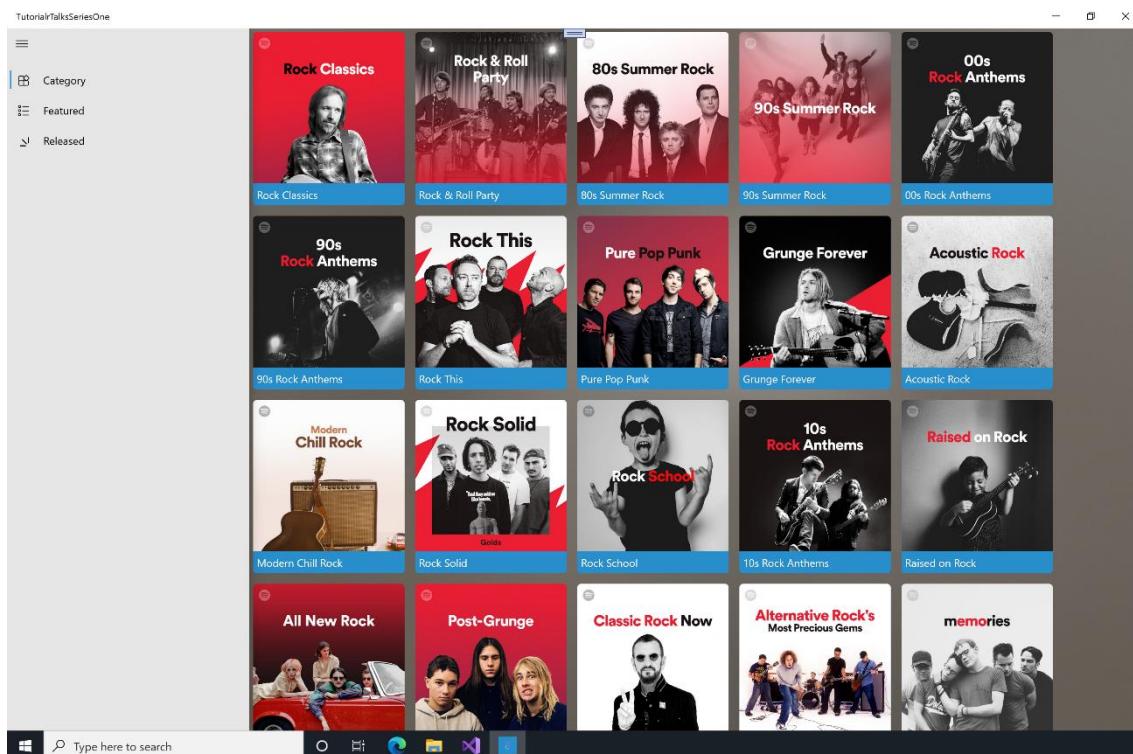
# Universal Windows Platform

Universal Windows Platform Demo shows the Navigation View with Category, Featured and Released options along with the Categories from Spotify.

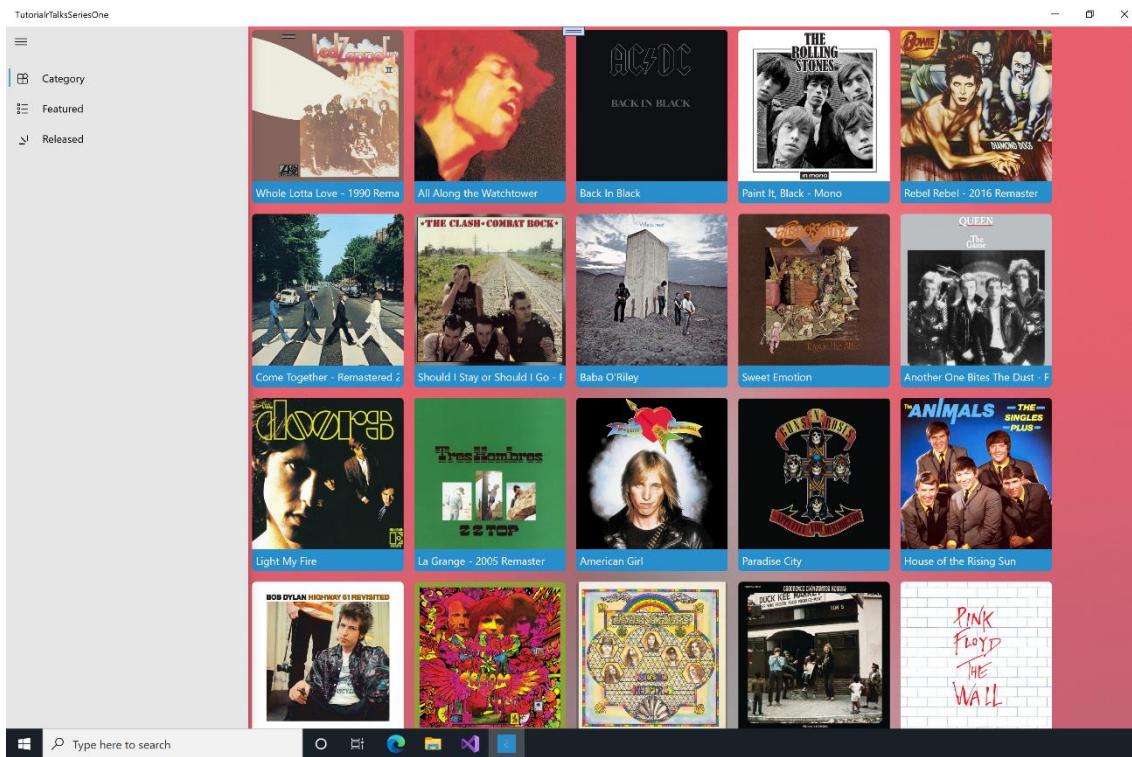
## Category



Category shows the Categories from Spotify such as the Rock Category.

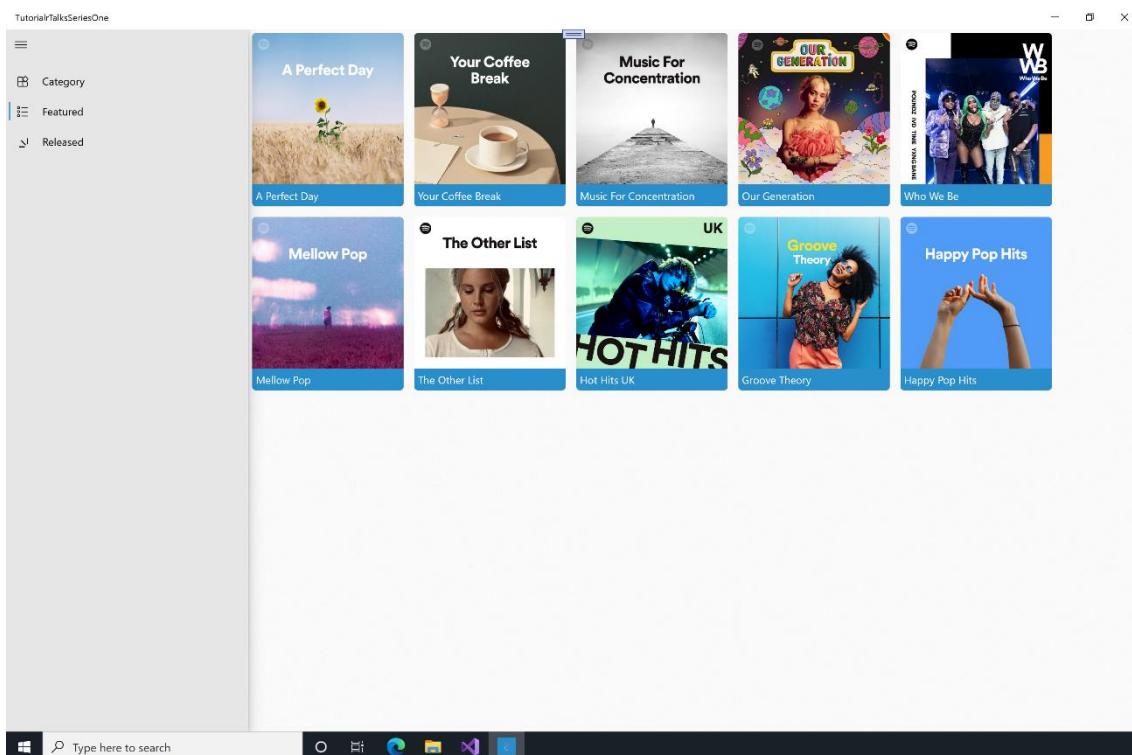


Category when the Rock Category is selected shows Playlists such as the Rock Classics Playlist.

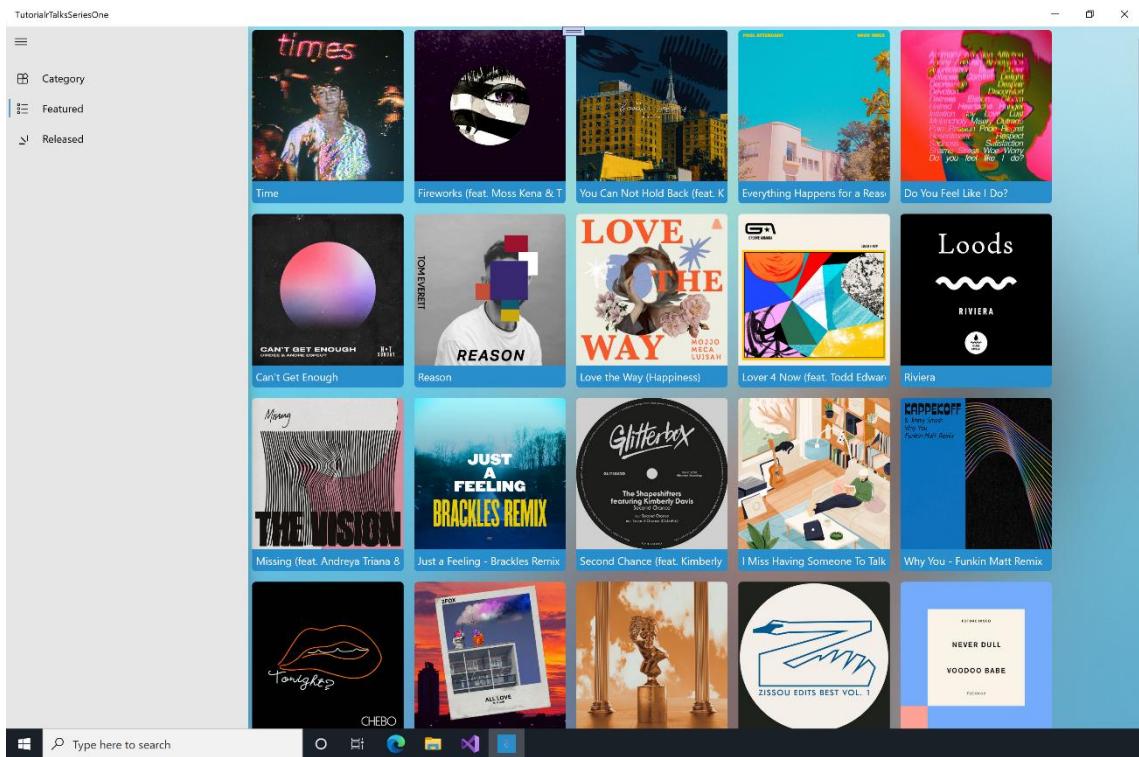


Category when the Rock Classic Playlist is selected shows various Playlist Items.

## Featured

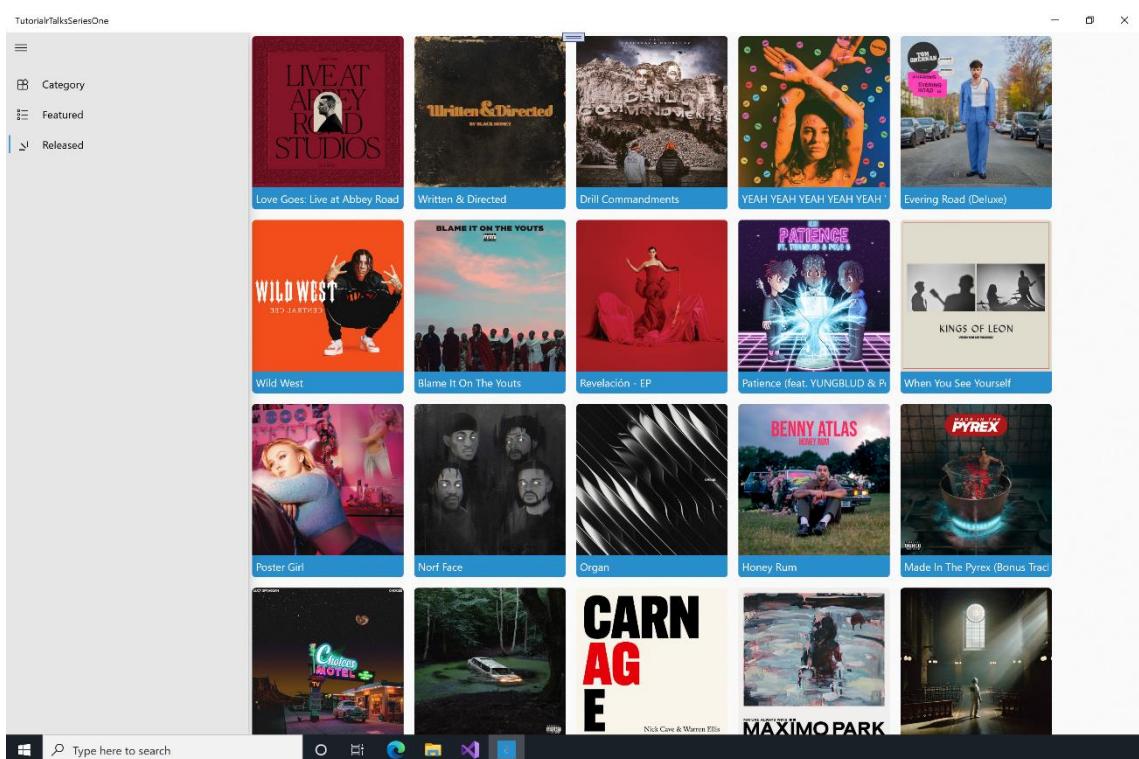


Featured, when selected, shows Playlists from Spotify such as Groove Theory.

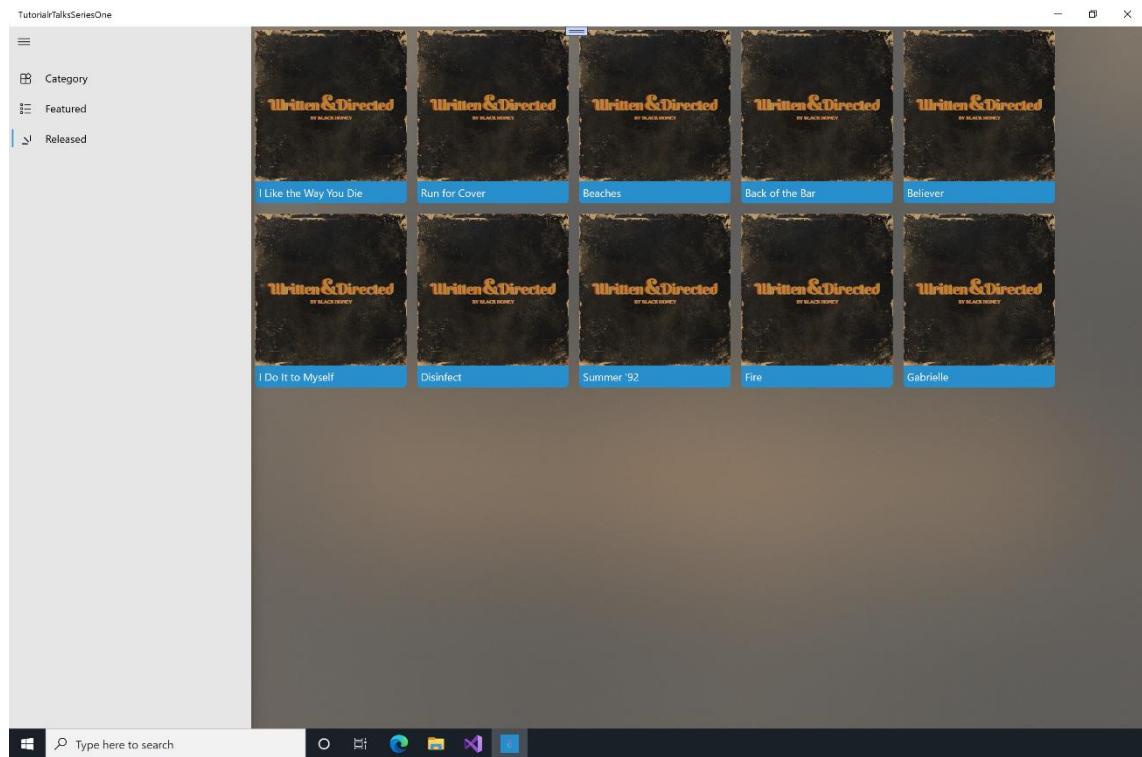


Featured when the Groove Theory Playlist is selected shows various Playlist Items.

## Released



Released, when selected, shows recently released Albums from Spotify such as Written & Directed.

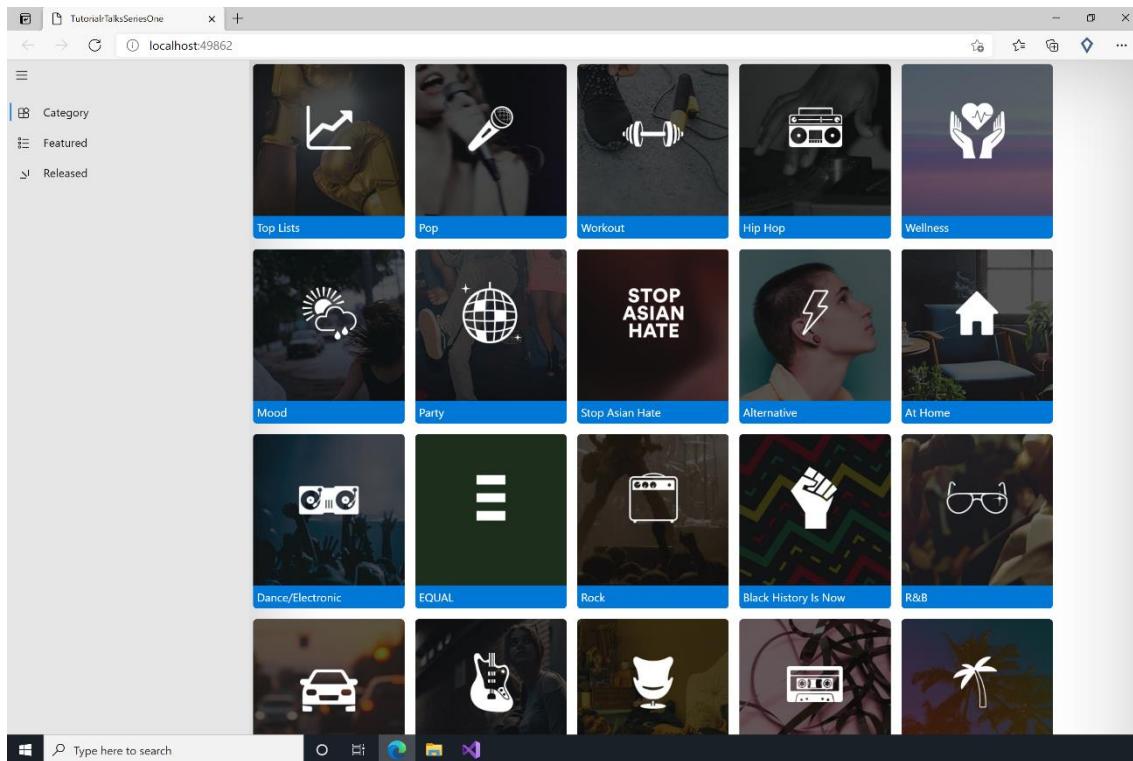


Released when the Written & Directed Album is selected shows the Tracks for the Album.

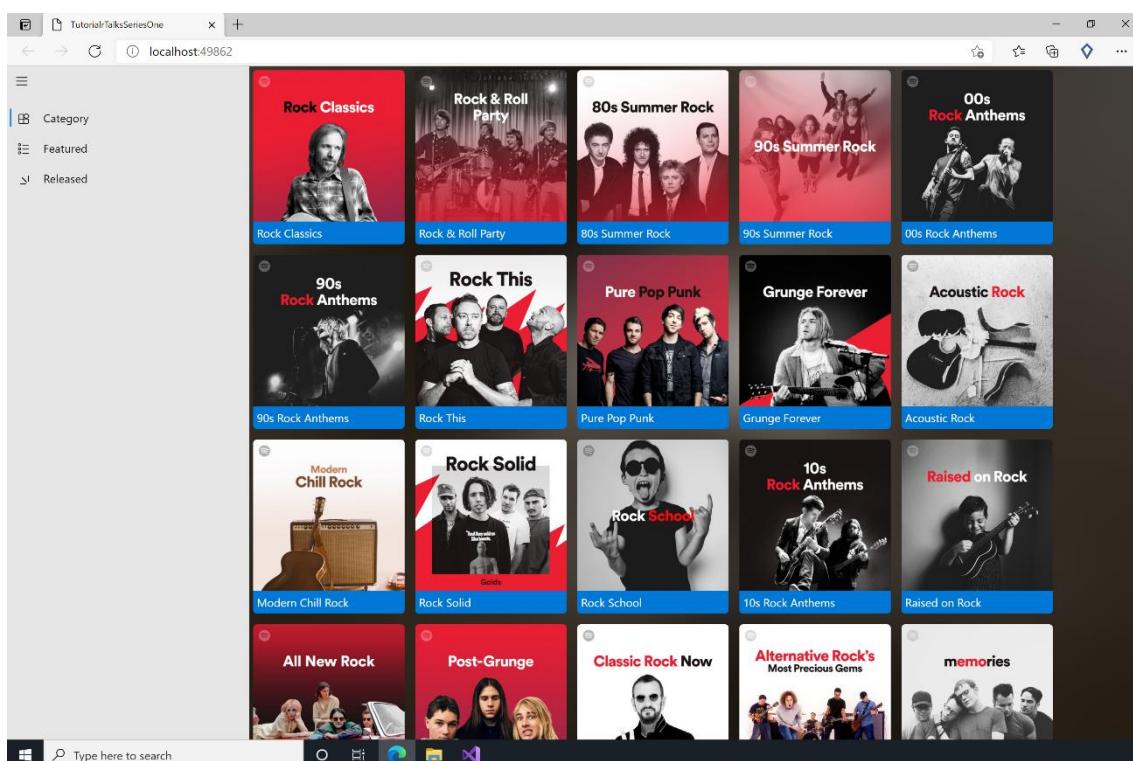
# WebAssembly

WebAssembly Demo in Microsoft Edge shows the Navigation View with Category, Featured and Released options along with the Spotify Categories like the Universal Windows Platform Demo.

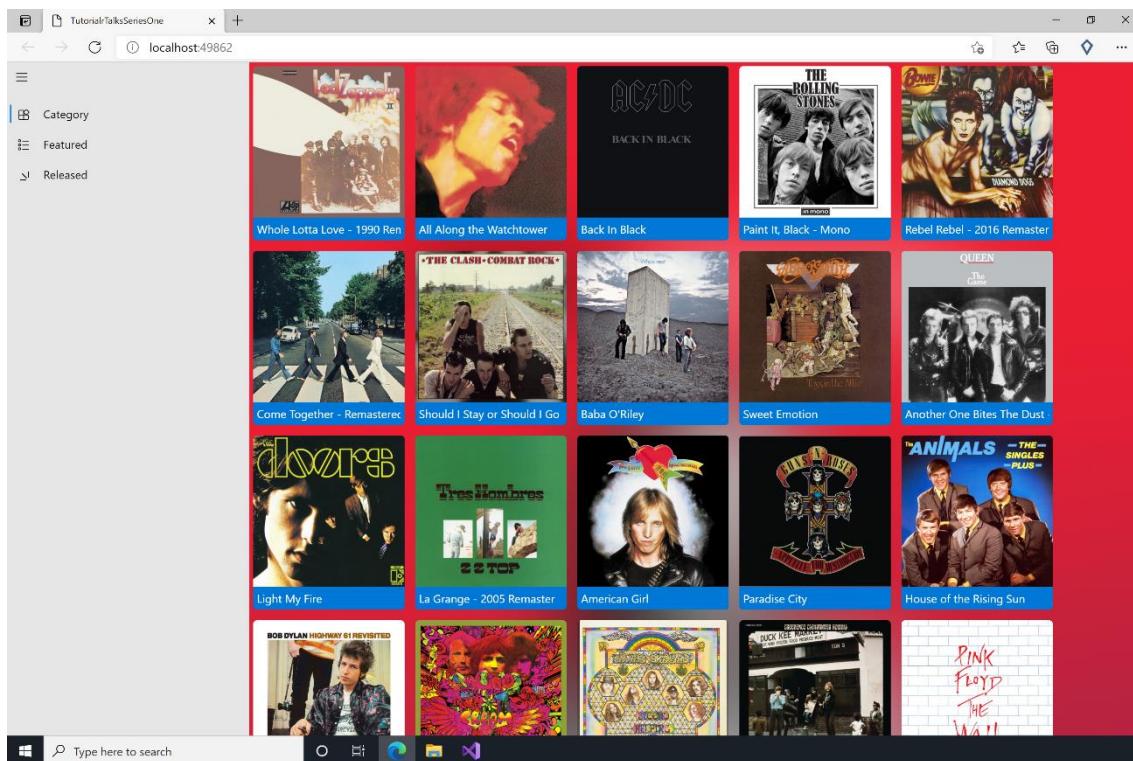
## Category



Category shows Spotify Categories such as Rock.

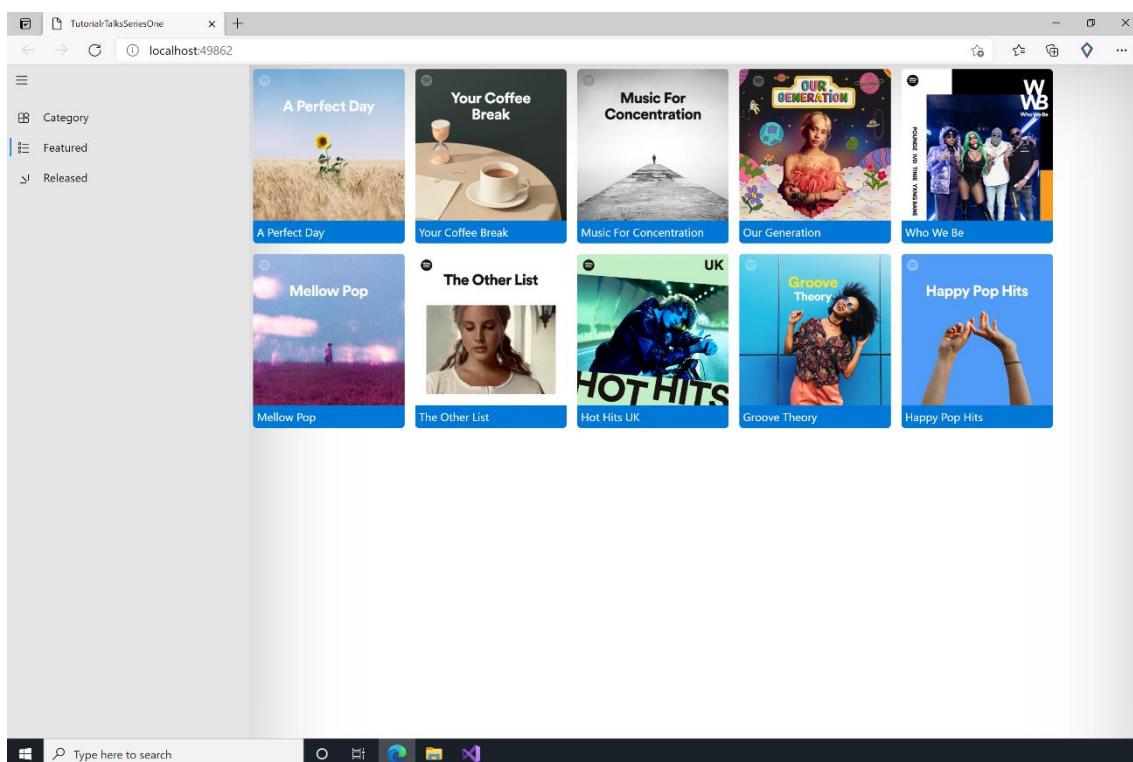


Category when Rock is selected shows Playlists such as Rock Classics.

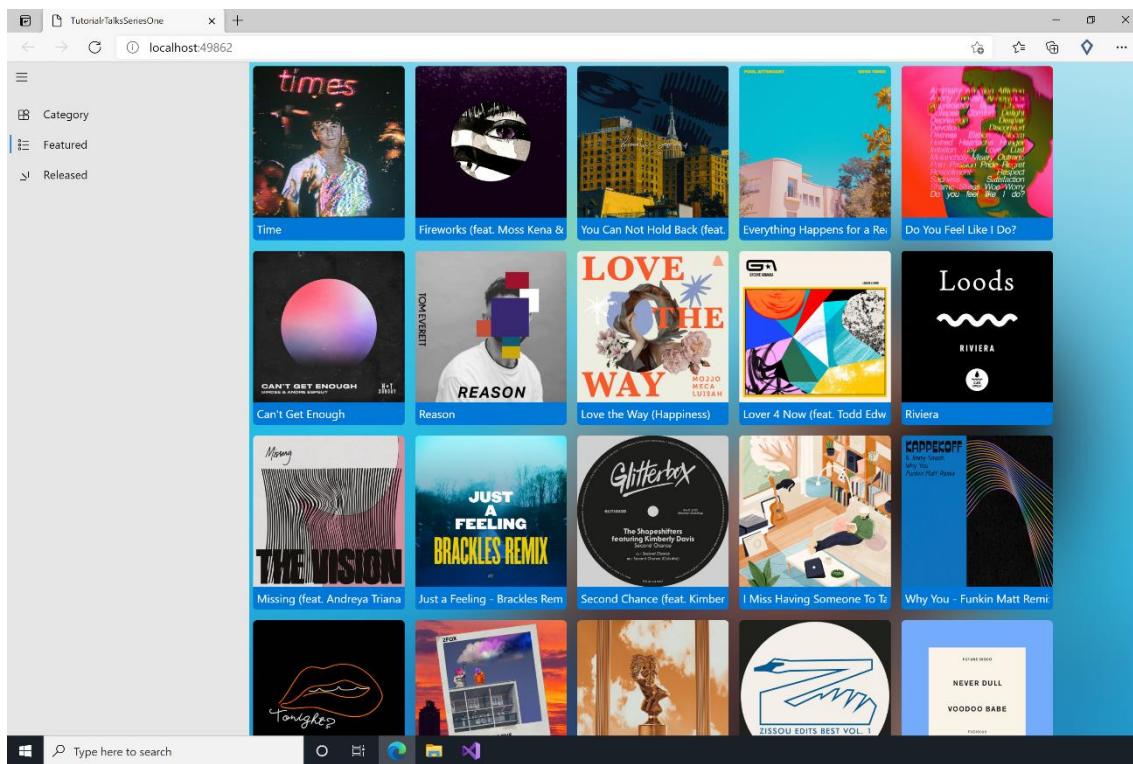


Category when Rock Classic is selected shows the Playlist Items.

## Featured

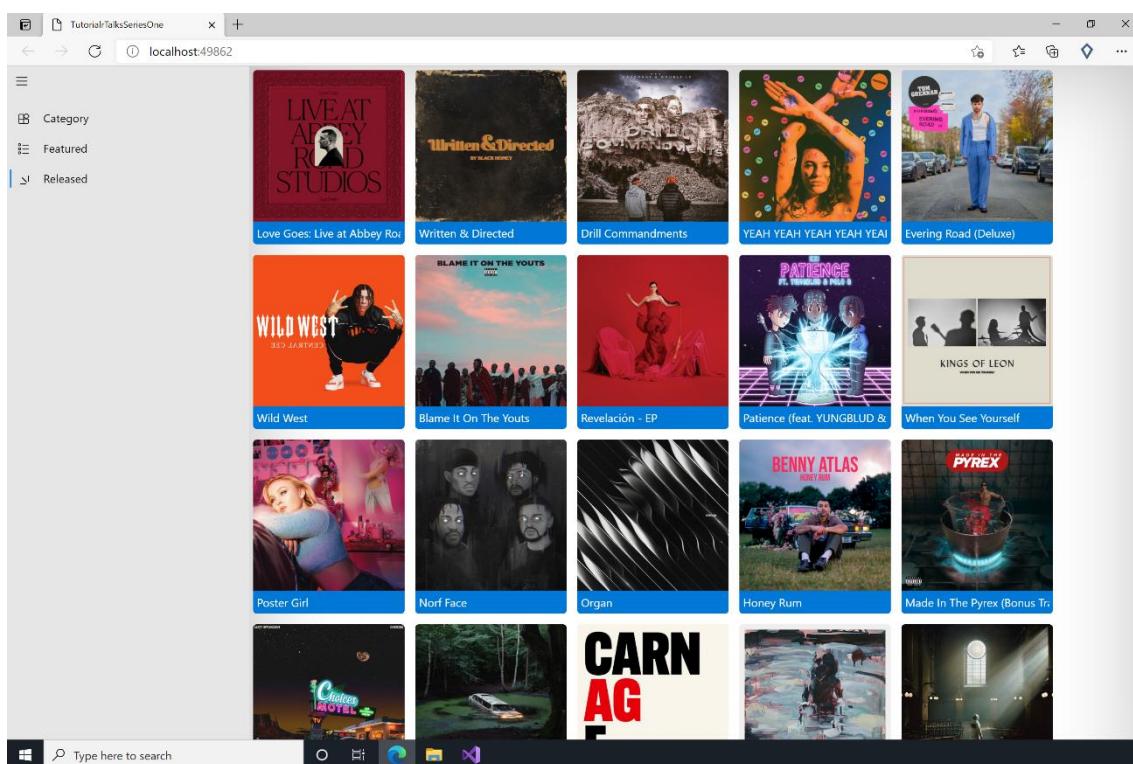


Featured, when selected, shows Featured Spotify Playlists such as Groove Theory like the Universal Windows Platform Demo.

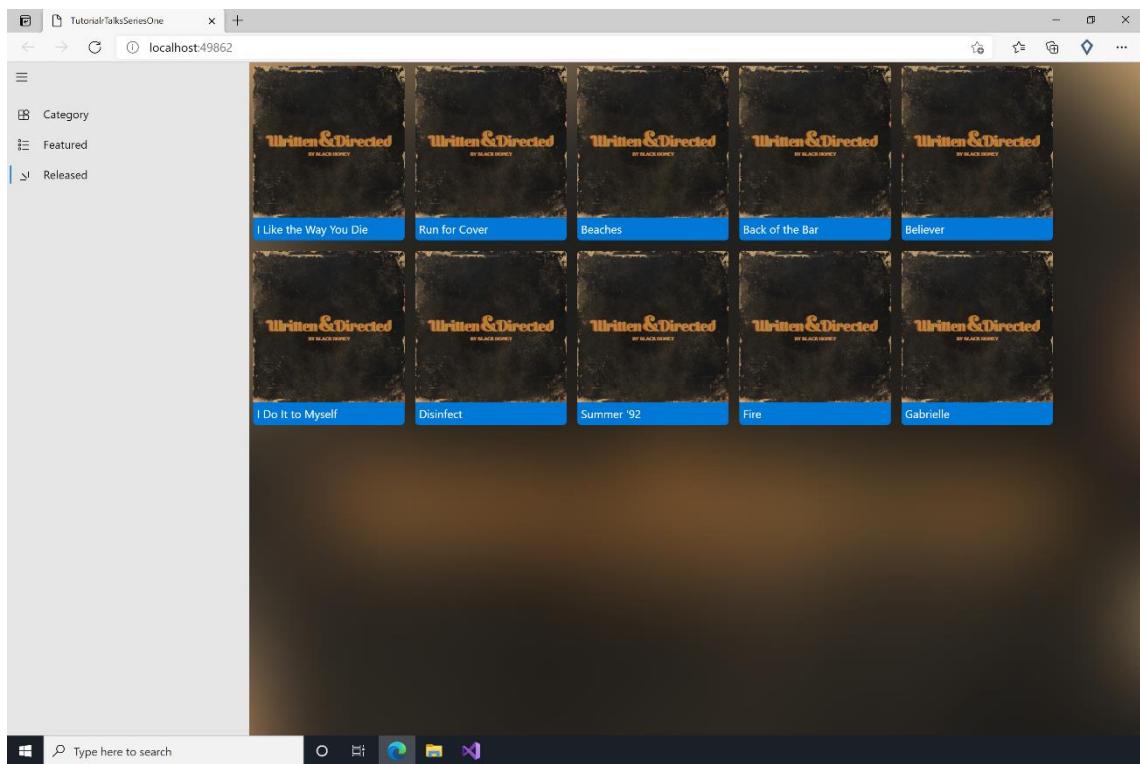


Featured when the Groove Theory is selected shows the Playlist Items.

## Released



Released, when selected, shows Recent Albums on Spotify such as Written & Directed like the Universal Windows Platform Demo.



Released when Written & Directed is selected shows the Tracks for the Album.