



*Creating mobile Apps with Xamarin  
and MV using Linkar*

**WITH OR WITHOUT WEB SERVICE**

*February 2019*

*by Kosday Solutions*



## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

We are going to develop a simple app with Xamarin that will read data from a MV file and represent it in a mobile device. This app can run in IOS and Android.

We are going to make the connection using two different ways:

- 
- Using a **REST WEB SERVICE** that will receive the app requests, using Linkar we will make requests to the MV database and send back the obtained data to the mobile device (you can read the HowTo about creating web services with Linkar in <https://www.kosday.com/blog/how-to-web-service-with-linkar>)
- Using a **DIRECT CONNECTION** to the Linkar Server. The mobile device will connect directly with Linkar Server. No need to use a Web Service.

### What do you need to create the web services:

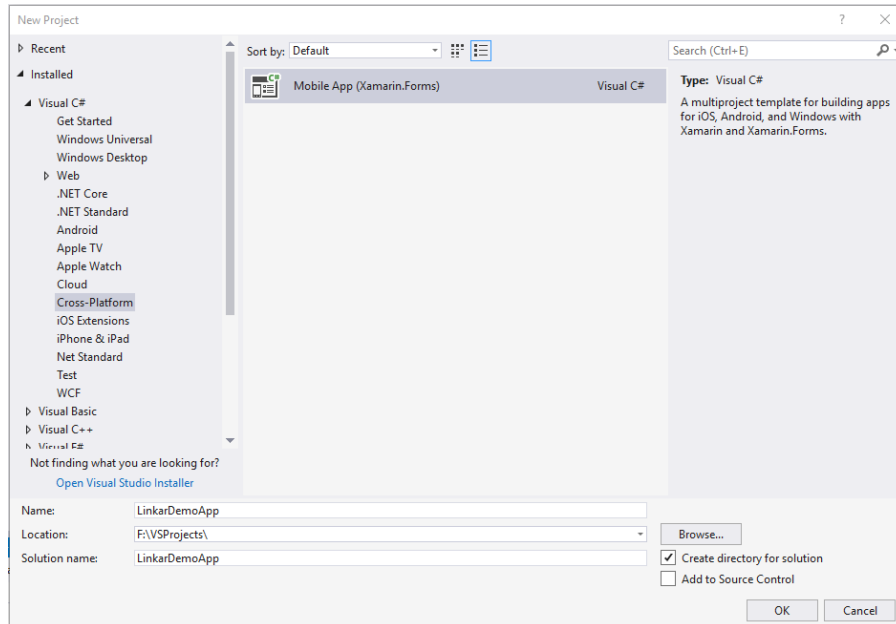
- **Linkar** (the lite free version is enough)
- An MV database (D3, jBase, QM, Unidata or Universe)
- Visual Studio 2017.
- Xamarin (<https://docs.microsoft.com/en-us/xamarin/cross-platform/get-started/installation/windows>)
- The LinkarClientMono.dll library (for the direct connection)

Let's open Visual Studio. We are going to create a new project "Mobile App":

*File → New → Cross-Platform → Mobile.app → Xamarin.Forms*

Let's name the project and the solution **LinkarDemoApp**.

## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE



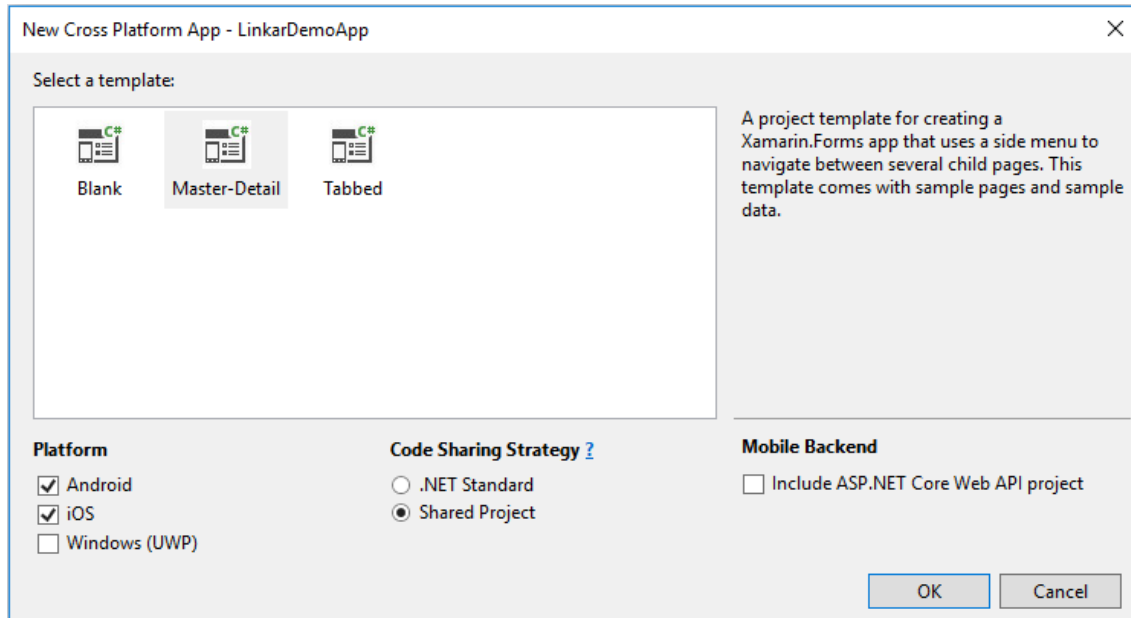
Let's choose the *Master-Detail* template (then we can work on the code example)

Let' choose *Android* and *iOS* as the destination *Platform*.

Let's choose *Shared Project* as *Code Sharing Strategy*.

Do not select *Mobile Backend*. You can see a complete sample of Backend [in this link](#).

## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE



This will generate 3 projects:

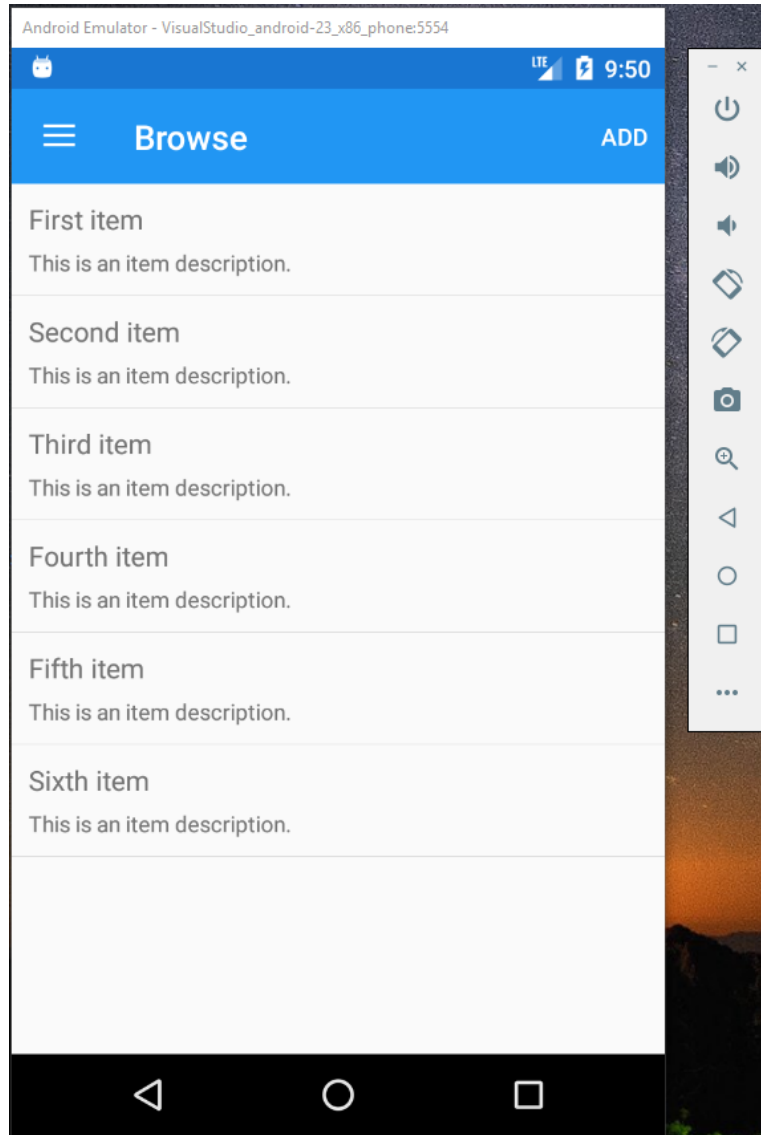
- LinkarDemoApp: Here we will insert most of the code and the app screens.
- LinkarDemoApp.Android: Here we will have the pure Android elements (AndroidManifest, MainActivity) and resources (icons, images ...)
- LinkarDemoApp.iOS: Here we will have the pure iOS elements (Info.plist) and resources (icons, images ...)

We are going to use a Windows PC to this development. By comfort, we are going to use LinkarDemoApp.Android as the start project. This will allow us to use the Android Emulator for testing purpose.

We have used the template so now we have a basic app that we can run and test.

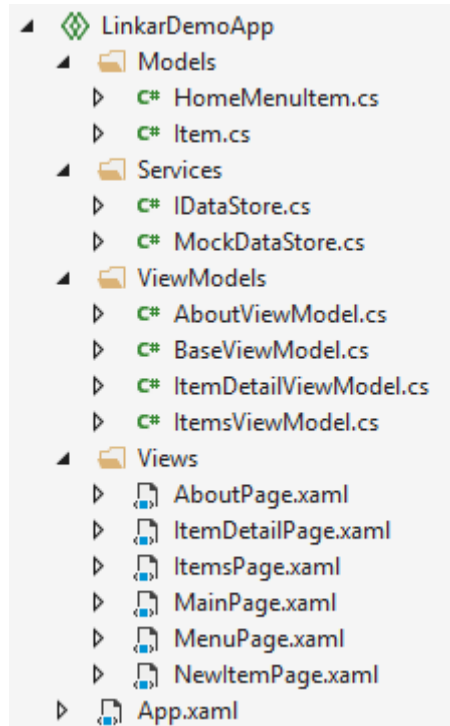


## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE



Let's explain the LinkarDemoApp project structure.

## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE



This template uses the MVVM (Model-View-View-Model) model. If you want to know more about it visit:

<https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>

The App document is the entry point of the shared code and will be call from the Android and iOS projects.

We will define on it the device events and the styles we will use in the different app views. For more details take a look at:

<https://docs.microsoft.com/en-us/dotnet/api/xamarin.forms.application?view=xamarin-forms>

In the *Services* folder, you have a local example about how you must write code to perform the different actions that will be call from the *ViewModels*.

There are two different ways to call Linkar Functions and work with the MV database.

## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

### 1. **WEB SERVICE** (Backend):

This type of connection implies that we have a *web service* between the app and Linkar Server. For instance, if we want an Item List for a MV file we will call the *GetItems* method (HTTP/HTTPS) declared in the web service that will use Linkar Client to select the file items, returns them and send them to the app. You can read the *HowTo* about creating web services with Linkar in <https://www.kosday.com/blog/how-to-web-service-with-linkar> in our blog.

This is a Class that calls a service from the app:

```
using System.Collections.Generic;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using Newtonsoft.Json;
using Sample.Models;

namespace Sample.Services
{
    public class ItemDataStore
    {
        HttpClient client;
        IEnumerable<Item> items;

        public ItemDataStore()
        {
            client = new HttpClient();
            client.BaseAddress = new Uri($"{App.servicesUrl}/");
            items = new List<Item>();
        }

        public async Task<IEnumerable<Item>> GetItemsAsync(bool forceRefresh = false)
        {
            if (forceRefresh)
            {
                var json = await client.GetStringAsync($"api/Sample/GetItems");
                items = await Task.Run(() =>
                    JsonConvert.DeserializeObject<IEnumerable<Item>>(json));
            }

            return items;
        }

        public async Task<Item> GetItemAsync(string id)
        {
            if (id != null)
            {
```

## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

```
var json = await client.GetStringAsync($"api/Sample/GetItem?id=" + id);  
return await Task.Run(() => JsonConvert.DeserializeObject<Item>(json));  
}  
  
return null;  
}  
}  
}
```

### 2. DIRECT CONNECTION WITH LINKAR

This connection uses the LinkarClient MONO library. Then the app interacts with Linkar Server directly.

First of all we must add the *LinkarClientMono.dll* library to **references** in *LinkarDemoApp.Android* and *LinkarDemoApp.iOS* projects.

Now we must add the **using** clauses in all the classes where we will need the client.

```
using LinkarClient;  
using LinkarCommon;
```

We also must declare the Linkar Client. In order to use it in different classes we will declare it as a static variable in **App.xaml.cs** file.

```
public static LinkarClnt linkarClnt = new LinkarClnt();
```

We will also implement a new function in order to control the *LinkarClient exceptions* that may occur. We do it in the same **App.xaml.cs** file:

```
public static string GetException(Exception ex)  
{  
    string msg = "";  
    if (ex.GetType() == typeof(LkException))  
    {  
        LkException lkex = (LkException)ex;  
        msg = "LINKAR EXCEPTION ERROR";  
        if (lkex.ErrorCode == LkException.ERRORCODE.C0003)  
            msg += "\r\nERROR CODE: " + lkex.ErrorCode +  
                "\r\nERROR MESSAGE: " + lkex.ErrorMessage +  
                "\r\nInternal ERROR CODE: " + lkex.InternalCode +  
                "\r\nInternal ERROR MESSAGE: " + lkex.InternalMessage;  
        else  
            msg += "\r\nERROR CODE: " + lkex.ErrorCode +
```



## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

```

"\r\nERROR MESSAGE: " + lhex.ErrorMessage;
}
else if (ex.GetType() == typeof(System.Net.Sockets.SocketException))
{
System.Net.Sockets.SocketException soex = (System.Net.Sockets.SocketException)ex;
msg = "SOCKET EXCEPTION ERROR\r\n" + soex.Message;
}
else
{
msg = "EXCEPTION ERROR\r\n" + ex.Message;
}

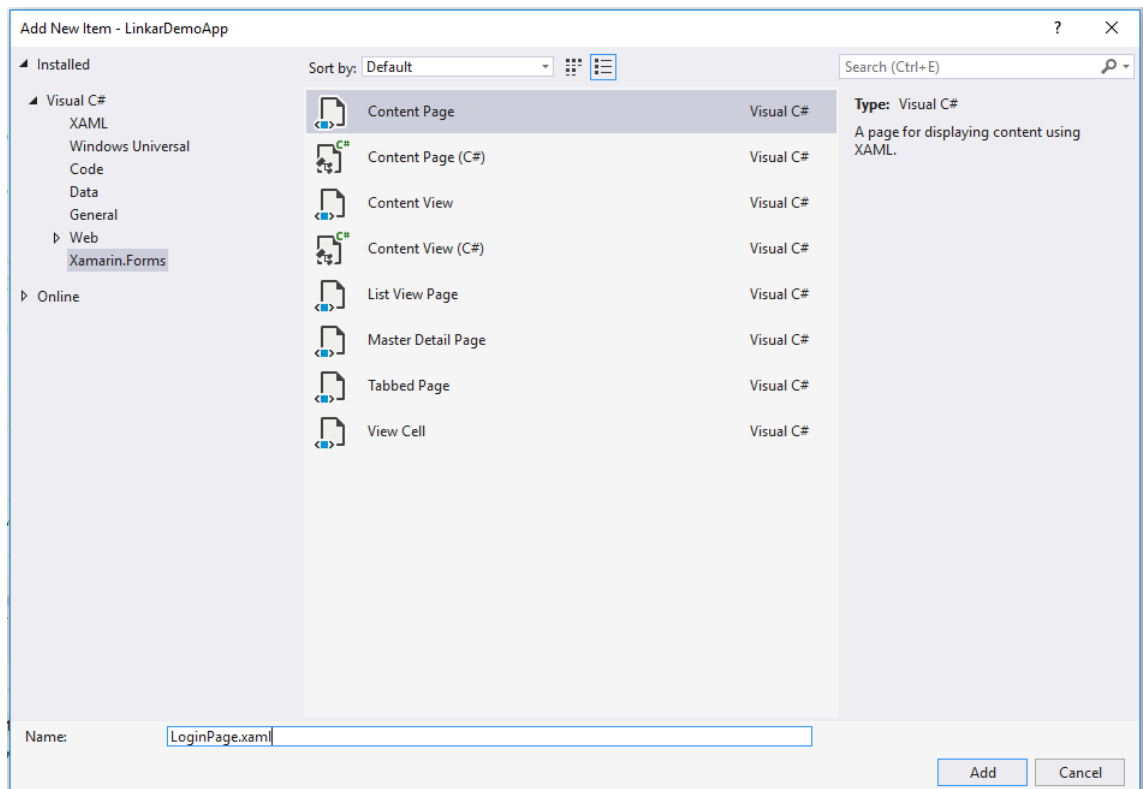
return msg;
}
}

```

Let's create a new cover page. We will show this page before the data.

We must right click in:

**Views Add -> New Item ...**



## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

And add a **Content Page**.

Now we change the created code with this:

LoginPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="LinkarDemoApp.Views.LoginPage">

<ContentPage.Resources>
<ResourceDictionary>
<Color x:Key="Primary">#2196F3</Color>
<Color x:Key="Accent">#cce8fe</Color>
<Color x:Key="LightTextColor">#999999</Color>
</ResourceDictionary>
</ContentPage.Resources>

<ContentPage.Content>
<Grid AbsoluteLayout.LayoutBounds="0,0,1,1"
AbsoluteLayout.LayoutFlags="All"
VerticalOptions="FillAndExpand"
HorizontalOptions="FillAndExpand">
<Grid.RowDefinitions>
<RowDefinition Height="Auto" />
<RowDefinition Height="*" />
</Grid.RowDefinitions>
<StackLayout BackgroundColor="{StaticResource Accent}"
VerticalOptions="FillAndExpand" HorizontalOptions="Fill">
<StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="Center">
<ContentView Padding="0,30,0,20" VerticalOptions="FillAndExpand">
<Image Source="linkar_logo.png" VerticalOptions="Center" HeightRequest="64" />
</ContentView>
</StackLayout>
</StackLayout>
<ScrollView Grid.Row="1">
<StackLayout Orientation="Vertical" Padding="16,10,16,10" Spacing="10" >
<Label >
<Label.FormattedText>
<FormattedString>
<FormattedString.Spans>
<Span Text="This app shows Linkar functionality and performance. Our focus is not
to show you how to make pretty interfaces or how to program." />
</FormattedString.Spans>
</FormattedString>
</FormattedString>
</Label.FormattedText>
</Label>
<Label>
<Label.FormattedText>
```

## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

```
<FormattedString>
<FormattedString.Spans>
<Span Text="This app uses Xamarin and a Linkar Server web service." />
</FormattedString.Spans>
</FormattedString>
</Label.FormattedText>
</Label>
<Label>
<Label.FormattedText>
<FormattedString>
<FormattedString.Spans>
<Span Text="You have all the source code in our web in the Linkar Demos folder.
Take a look at it and discover how easy it is." />
</FormattedString.Spans>
</FormattedString>
</Label.FormattedText>
</Label>
<Button Text="Login" Clicked="Button_Clicked" TextColor="White"
BackgroundColor="{StaticResource Primary}" HeightRequest="40"/>
</StackLayout>
</ScrollView>
</Grid>
</ContentPage.Content>
</ContentPage>
```

LoginPage.xaml.cs

```
using LinkarDemoApp.ViewModels;
using System;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace LinkarDemoApp.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class LoginPage : ContentPage
    {
        LoginViewModel viewModel;

        public LoginPage()
        {
            InitializeComponent();
            NavigationPage.SetHasNavigationBar(this, false); // Hide nav bar

            BindingContext = viewModel = new LoginViewModel();
        }

        async void Button_Clicked(object sender, EventArgs e)
        {
            await LoginAsync();
        }
    }
}
```

## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

```
}  
  
public async Task LoginAsync()  
{  
    CredentialsOptions credentialsOptions = new CredentialsOptions("127.0.0.1",  
    11300, "DEMOEP", "demo", "demo1234");  
    try  
    {  
        await Task.Run(() => {  
            App.linkarClt.Login(credentialsOptions);  
            App.Current.MainPage = new MainPage();  
        });  
    }  
    catch (Exception ex)  
    {  
        await this.DisplayAlert("ERROR", App.GetException(ex), "OK");  
    }  
}  
}
```

Change the **CredentialsOptions** line with yours.

As we do not use any data capture, nor data binding of that information, we do not need *Model*, nor *ViewModel* structures. We will only use the *button* code as you can see in the example. For a complete Login form you will need all the structure. By now, when the Login is correct we will navigate to the *MainPage*.

Let's go back to the **App.xaml.cs** file and change the constructor of the start app page with our Login page:

```
public App()  
{  
    InitializeComponent();  
    MainPage = new LoginPage();  
}
```

Now, let's change the *Item.cs* model in order to populate the *LK.ITEMS file* data from the LINKAR\_DEMO account in MV.

```
public class Item  
{  
    public string Id { get; set; }  
}
```

## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

```
public string Description { get; set; }  
public double Stock { get; set; }  
}
```

After this change, we must also make small changes in *Views* and *ViewModels*:

We must replace ***NewItemPage.xaml.cs*** constructor with this new code:

```
public NewItemPage()  
{  
    InitializeComponent();  
  
    Item = new Item  
    {  
        Stock = 0,  
        Description = "This is an item description."  
    };  
  
    BindingContext = this;  
}
```

Also we replace the ***ItemDetailPage.xaml.cs*** constructor with:

```
public ItemDetailPage()  
{  
    InitializeComponent();  
  
    var item = new Item  
    {  
        Stock = 0,  
        Description = "This is an item description."  
    };  
  
    viewModel = new ItemDetailViewModel(item);  
    BindingContext = viewModel;  
}
```

And the ***ItemDetailViewModel.cs*** constructor with:

```
public ItemDetailViewModel(Item item = null)
```

## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

```
{  
  Title = item?.Description;  
  Item = item;  
}
```

In the XAML Views (*ItemsPage.xaml*, *ItemDetailPage.xaml* y *NewItemPage.xaml*) we will just change the references we find of the **Text property** with the **Stock property**, for instance:

```
<Label Text="Stock:" FontSize="Medium" />  
<Label Text="{Binding Item.Stock}" FontSize="Small"/>
```

Next step will be to modify the **MockDataStore.cs** file that is in the *Services* folder in order to obtain the item list from Linkar Server. First we will modify the constructor to take out the fixed data that the screen shows and then we will add the **using** clauses for Linkar:

```
public MockDataStore()  
{  
  items = new List<Item>();  
}
```

Now we will modify **GetItemsAsync** method in the same file. We use this method to obtain data from Linkar:

```
public async Task<IEnumerable<Item>> GetItemsAsync(bool forceRefresh = false)  
{  
  if (forceRefresh)  
  {  
    items = null;  
    try  
    {  
      string lkstring = App.linkarClt.Select_Text("LK.ITEMS", "", "BY CODE", "", "",  
        new SelectOptions(false, false, 10, 1, true), DataFormat.TYPE.MV, "", 0);  
      if (!string.IsNullOrEmpty(lkstring))  
      {  
        char delimiter = ASCII_Chars.FS_chr;  
        char delimiterThisList = DBMV_Mark.AM;  
        String recordIds = "";  
        String records = "";
```

## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

```
String recordCalculateds = "";
String[] parts = lkstring.Split(delimiter);
if (parts.Length >= 1)
{
String[] ThisList = parts[0].Split(delimiterThisList);
int numElements = ThisList.Length;
for (int i = 1; i < numElements; i++)
{
if (ThisList[i].Equals("RECORD_ID"))
{
recordIds = parts[i];
}
if (ThisList[i].Equals("RECORD"))
{
records = parts[i];
}
if (ThisList[i].Equals("CALCULATED"))
{
recordCalculateds = parts[i];
}
}
}

//Fill all the records with response data
String[] lstids = recordIds.Split(ASCII_Chars.RS_chr);
String[] lstregs = records.Split(ASCII_Chars.RS_chr);
String[] lstcalcs = recordCalculateds.Split(ASCII_Chars.RS_chr);

items = new List<Item>();

for (int i = 0; i < lstids.Length; i++)
{
Item record = new Item();
if (recordCalculateds != null && recordCalculateds != "")
{
record = LkItem.GetRecord(lstids[i], lstregs[i], lstcalcs[i]);
}
else
record = LkItem.GetRecord(lstids[i], lstregs[i], "");

items.Add(record);
}
}
}
catch (Exception ex)
{
string error = App.GetException(ex);
await Xamarin.Forms.Application.Current.MainPage.DisplayAlert("ERROR", error,
"OK");
}
```

## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

```
}  
}  
  
return items;  
}
```

We are only using one of the Linkar functions output types. If you want to see all download Linkar help documentation in:

[https://www.kosday.com/Manuals/en\\_WEB\\_LINKAR/index.html](https://www.kosday.com/Manuals/en_WEB_LINKAR/index.html)

Finally we need to add a **GetRecord** method. We will use it to populate each Item of our list assigning to each class property the value that the function returns.

```
private Item GetRecord(string recordID, string record, string recordCalculated)  
{  
    Item item = new Item();  
  
    //Create empty record  
    string[] reg = new string[2];  
    for (int j = 0; j < reg.Length; j++)  
        reg[j] = "";  
  
    //Fill the record  
    if (record != null && record != "")  
    {  
        string[] aux = record.Split(DBMV_Mark.AM);  
        for (int j = 0; j < aux.Length; j++)  
            reg[j] = aux[j];  
    }  
  
    //Create empty calculated record
```



## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

```
string[] regI = new string[0];

for (int k = 0; k < regI.Length; k++)

regI[k] = "";

//Fill the calculated record

if (recordCalculated != null && recordCalculated != "")

{

string[] auxI = recordCalculated.Split(DBMV_Mark.AM);

int k = 0;

for (; k < auxI.Length; k++)

regI[k] = auxI[k];

}

//Fill the record ID property

item.Id = recordID;

if (record == null || record == "")

return null;

//Fill the class properties

item.Description = LinkarDataTypes.GetAlpha(reg[0]);

item.Stock = LinkarDataTypes.GetDecimal(reg[1]);

return item;

}
```

## Creating mobile Apps with Xamarin and MV using Linkar WITH OR WITHOUT WEB SERVICE

This is only a small example about how you can create a mobile app with Visual Studio and Linkar that logins to the Linkar Server, executes a Select and print the data in the mobile device.

For a much more complete example you can download the folder "Linkar Demos" on our page <https://www.kosday.com/Product/Download/7>, and open the folder LinkarViewFilesDirect within the compressed file sources.zip.

You also can download the app from the **Google Store** or from the **Apple Store** and run it in your device.

The focus of this app is not to show a complex, pretty mobile app, it is just to show you how easy is to use Linkar to create mobiles app with MV databases.

**Android:**

<https://play.google.com/store/apps/details?id=com.kosday.LinkarViewFiles&hl=es> 419

**iOS:**

<https://itunes.apple.com/us/app/linkar-vf/id1447620240?mt=8>

# Thank you very much

## [www.linkarmv.com](http://www.linkarmv.com)

## sl@kosday.com