**Xamarin Forms App Tutorial**

**Xamarin App:** [**GitHub - tshego3/EmployeeManagementXamarinFormsApp: Xamarin Forms App - Employee Management, consuming RESTful API with OAuth.**](https://github.com/tshego3/EmployeeManagementXamarinFormsApp)

**API:** [**GitHub - tshego3/EmployeeManagementAPI: ASP.NET - Employee Management API with OAuth.**](https://github.com/tshego3/EmployeeManagementAPI)

**YouTube:** [**https://youtube.com/playlist?list=PLpbcUe4chE78YvgIMtmgNEmRGyGJcsQdF**](https://youtube.com/playlist?list=PLpbcUe4chE78YvgIMtmgNEmRGyGJcsQdF)

Create a new Mobile App.



Provide a title for the solution, select the blank template and create for iOS, Android & UWP.

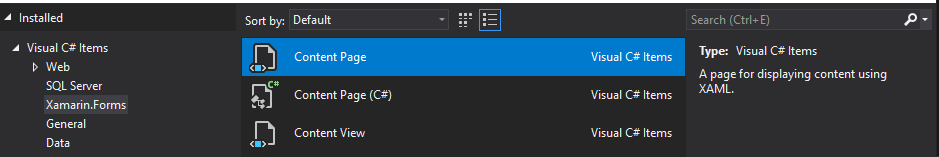
1. Add a new folder “ViewModels” and a class “RegisterViewModel”.



Add the following code in the “RegisterViewModel” Class.

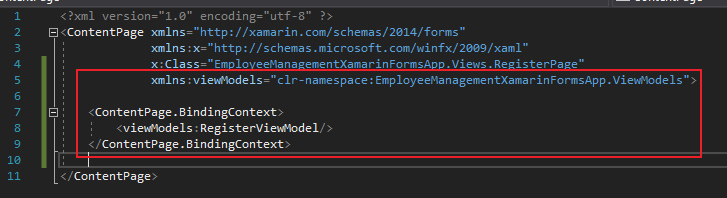


2. Add a new folder “Views” and a Forms Blank Content Page Xaml “RegisterPage”.

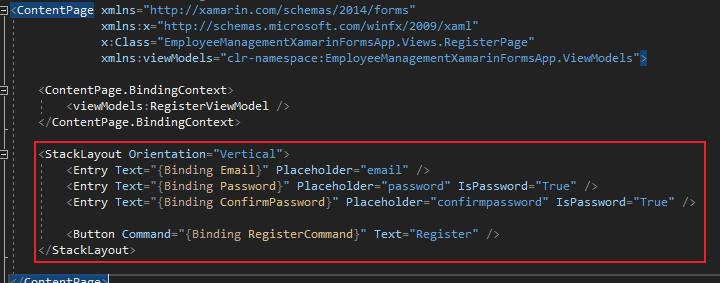




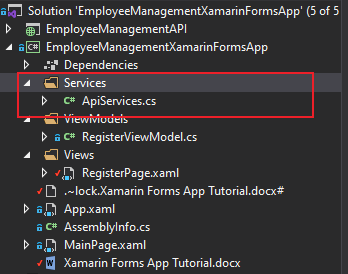
Add the following line in the Content Page, so we can do data binding.



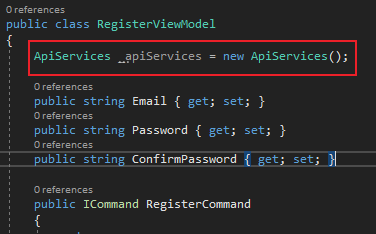
Add the following code in code.



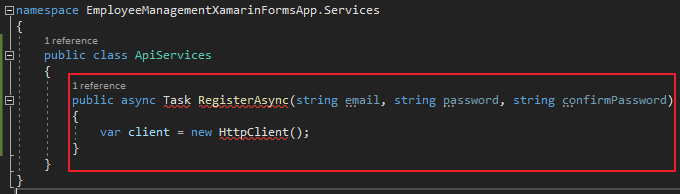
3. Add a new folder “Services” and a Class “ApiServices”.



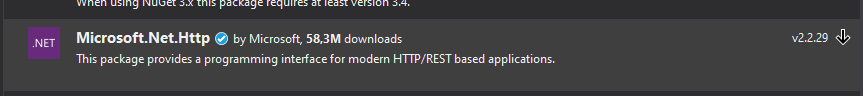
Add the following code in the “RegisterViewModel” Class.



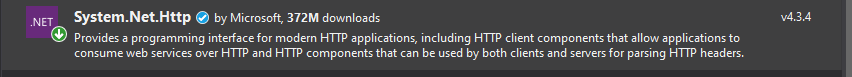
Add the code in the “ApiServices” Class.



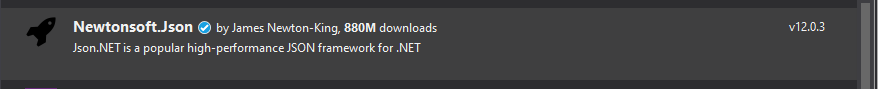
Afterwards install the Microsoft.Net.Http or System.Net.Http and Newtonsoft.JSON NuGet Packages in the “EmployeeManagementXamarinFormsApp” (as well for Android, iOS & UWP projects)via the NuGet Package Manager.



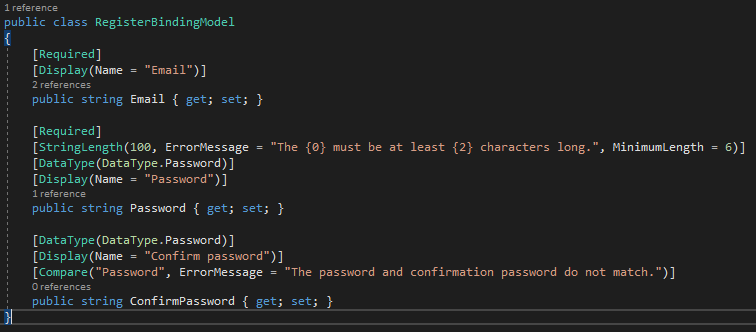
Or



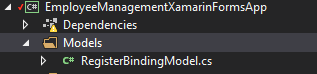
And



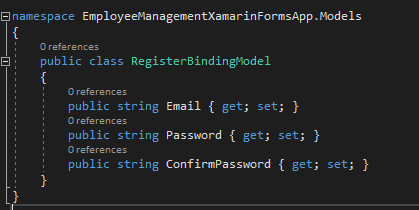
Open the “AccountBindingModels” Class in the Web API (EmployeeManagementAPI) Solution and copy the “RegisterBindingModel” in the class.



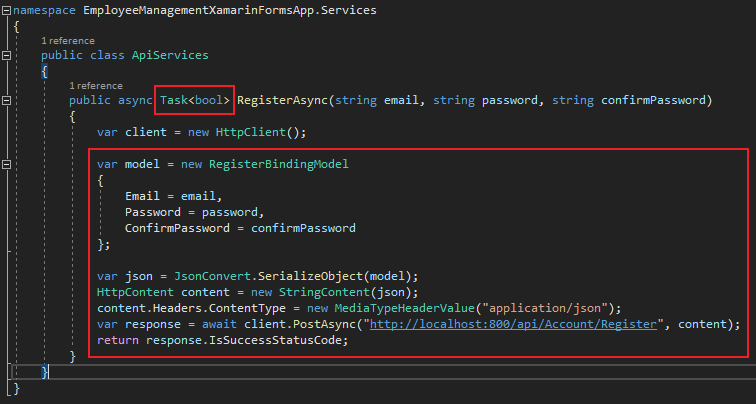
Add a new folder “Models” and a Class “RegisterBindingModel”.



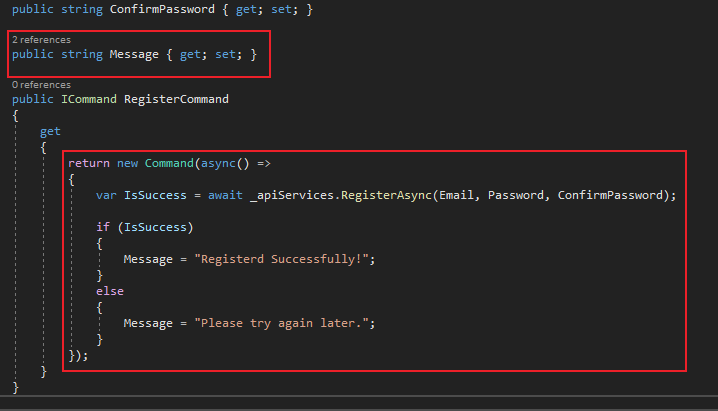
Add the code copied from the “RegisterBindingModel” Class found in the Web API, with the following alterations.



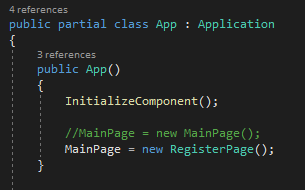
Add the following code in the “ApiServices” Class and alterations on the async Task.



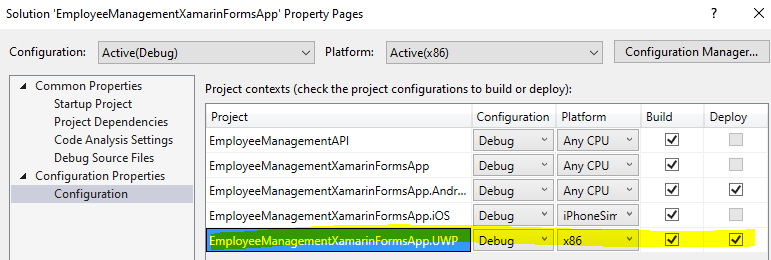
Add the following code in the “RegisterViewModel” Class.



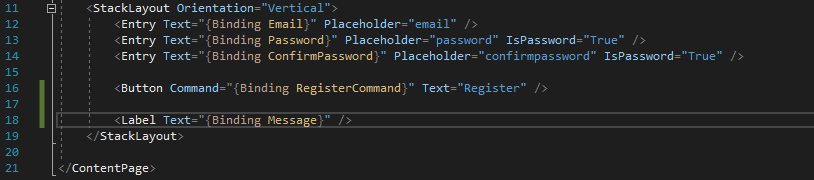
Alter the code in the “App.xaml.cs” Class, so that the app starts up in the Register Page.



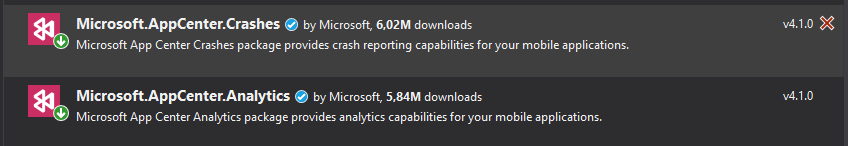
4. Confirm that UWP is deployed by right-clicking the project solution and click “Set Startup Projects...”

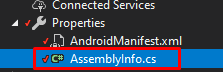


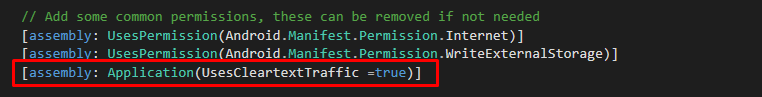
Added the following lable in the “RegisterPage” Content Page and test the app.



Afterwards install the Microsoft.AppCenter.Crashes and Microsoft.AppCenter.Analytics NuGet Packages in the “EmployeeManagementXamarinFormsApp” (as well for Android, iOS & UWP projects) via the NuGet Package Manager.



Afterwards Enable “Clear Text by going into the Android Project, then click on properties.  


Click on “AssemblyInfo” Class, then add the following code.  
 

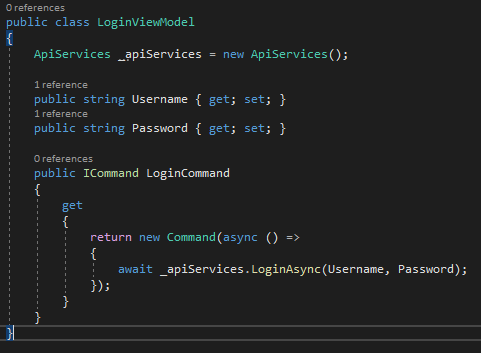
**Enable long file names on Win 10 Pro:**

Search for “Local Group Policy Editor” and open it. Under Local Computer Policy/Computer Configuration/Administrative Templates/System/Filesystem/. Double click on “Enabling Win32 long paths” and set it enabled.

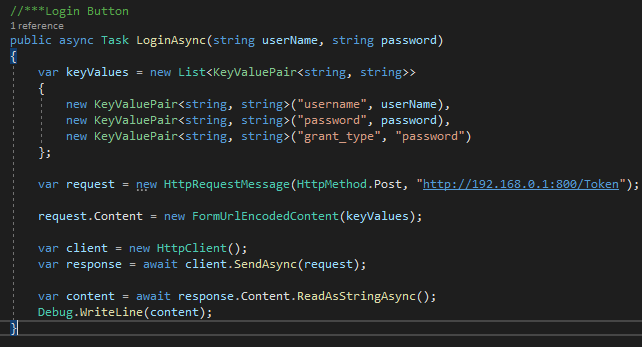
5. Add a new Class “LoginViewModel” in the “ViewModels” folder.



And add the following code in the “LoginViewModel” Class.



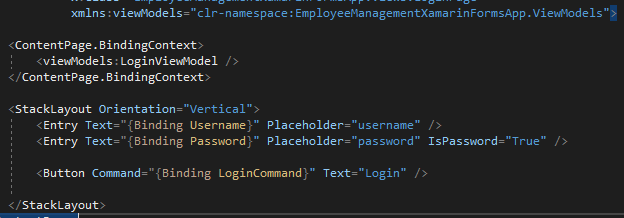
Add the following code in the “ApiServices” Class.



Add a new Forms Blank Content Page Xaml “LoginPage” in the “Views” folder.



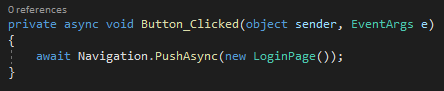
Add the following code in the “LoginPage” Content Page.



Add the following code in the “RegisterPage” Content Page, which allows a user to either register or login when the application starts up.



And the code in the “RegisterPage.xaml.cs” Content Page Class.



Update the code in the “App.xaml.cs” Class and test the app.



6. Add a new Class “EmployeeBindingModel” in the “Models” folder.



And the following code.

public class EmployeeBindingModel

{

public int EmployeeID { get; set; }

public string TbFirstName { get; set; }

public string TbSurname { get; set; }

public string TbTellNo { get; set; }

public string TbEmail { get; set; }

public string UserID { get; set; }

}

Add a new Class “EmployeesViewModel” in the “ViewModels” folder.



And add the following code in the “EmployeesViewModel” Class.

public class EmployeesViewModel : INotifyPropertyChanged

{

ApiServices \_apiServices = new ApiServices();

private List<EmployeeBindingModel> \_employees;

//public string AccessToken { get; set; }

public List<EmployeeBindingModel> Employees

{

get { return \_employees; }

set

{

\_employees = value;

OnPropertyChanged();

}

}

public ICommand GetEmployeesCommand

{

get

{

return new Command(async () =>

{

Employees = await \_apiServices.GetEmployeesAsync(AccessToken);

});

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

Add the following code in the “ApiServices” Class.

//\*\*\*Loads All Employees

public async Task<List<EmployeeBindingModel>> GetEmployeesAsync(string accessToken)

{

var client = new HttpClient();

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", accessToken);

var json = await client.GetStringAsync("http://192.168.137.247:800/api/Employees");

var employees = JsonConvert.DeserializeObject<List<EmployeeBindingModel>>(json);

return employees;

}

Add a new Forms Blank Content Page Xaml “EmployeesPage” in the “Views” folder.



Add the following code in the “EmployeesPage” Content Page.

xmlns:viewModels="clr-namespace:EmployeeManagementXamarinFormsApp.ViewModels">

<ContentPage.BindingContext>

<viewModels:EmployeesViewModel />

</ContentPage.BindingContext>

<StackLayout Orientation="Vertical">

<ListView ItemsSource="{Binding Employees}" HasUnevenRows="True">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell>

<StackLayout>

<Label Text="{Binding TbFirstName}"/>

<Label Text="{Binding TbSurname}"/>

<Label Text="{Binding TbTellNo}"/>

<Label Text="{Binding TbEmail}"/>

</StackLayout>

</ViewCell>

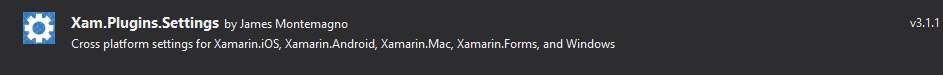
</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackLayout>

Afterwards install the “Xam.Plugins.Settings” NuGet Package in the “EmployeeManagementXamarinFormsApp” (as well for Android, iOS & UWP projects) via the NuGet Package Manager. This plugin stores data as KeyValues and not as database table.



Add a new folder “Helpers” and in that folder, add a Class “Settings”.



Then add this code into the “Settings” Class (the original code is found on the Xam.Plugins.Settings readme.txt file).

using Plugin.Settings;

using Plugin.Settings.Abstractions;

namespace EmployeeManagementXamarinFormsApp.Helpers  
{

public static class Settings

{

private static ISettings AppSettings

{

get

{

return CrossSettings.Current;

}

}

public static string Username

{

get

{

return AppSettings.GetValueOrDefault("Username", "");

}

set

{

AppSettings.AddOrUpdateValue("Username", value);

}

}

public static string Password

{

get

{

return AppSettings.GetValueOrDefault("Password", "");

}

set

{

AppSettings.AddOrUpdateValue("Password", value);

}

}

}

}

public static string AccessToken

{

get

{

return AppSettings.GetValueOrDefault("AccessToken", "");

}

set

{

AppSettings.AddOrUpdateValue("AccessToken", value);

}

}

Add the following code in the “RegisterViewModel” Class.

public ICommand RegisterCommand

{

get

{

return new Command(async() =>

{

var IsSuccess = await \_apiServices.RegisterAsync(Email, Password, ConfirmPassword);

Settings.Username = Email;

Settings.Password = Password;

if (IsSuccess)

{

Message = "Registerd Successfully!";

}

else

{

Message = "Please try again later.";

}

});

}

}

Add the following code in the “LoginViewModel” Class.

public LoginViewModel()

{

Username = Settings.Username;

Password = Settings.Password;

}

Update the code in the “ApiServices” Class.

//\*\*\*Login Button

public async Task<string> LoginAsync(string userName, string password)

{

var keyValues = new List<KeyValuePair<string, string>>

{

new KeyValuePair<string, string>("username", userName),

new KeyValuePair<string, string>("password", password),

new KeyValuePair<string, string>("grant\_type", "password")

};

var request = new HttpRequestMessage(HttpMethod.Post, "http://192.168.137.247:800/Token");

request.Content = new FormUrlEncodedContent(keyValues);

var client = new HttpClient();

var response = await client.SendAsync(request);

//var content = await response.Content.ReadAsStringAsync();

//Debug.WriteLine(content);

var jwt = await response.Content.ReadAsStringAsync();

JObject jwtDynamic = JsonConvert.DeserializeObject<dynamic>(jwt);

var accessToken = jwtDynamic.Value<string>("access\_token");

Debug.WriteLine(jwt);

return accessToken;

}

Update the following code in the “LoginViewModel” Class.

public ICommand LoginCommand

{

get

{

return new Command(async () =>

{

var accesstoken = await \_apiServices.LoginAsync(Username, Password);

Settings.AccessToken = accesstoken;

});

}

}

Comment and add the following code in the “EmployeesViewModel” Class.

//public string AccessToken { get; set; }

public List<EmployeeBindingModel> Employees

{

get { return \_employees; }

set

{

\_employees = value;

OnPropertyChanged();

}

}

public ICommand GetEmployeesCommand

{

get

{

return new Command(async () =>

{

//Employees = await \_apiServices.GetEmployeesAsync(AccessToken);

var accesstoken = Settings.AccessToken;

Employees = await \_apiServices.GetEmployeesAsync(accesstoken);

});

}

}

Update and add the following code in the “App.xaml.cs” Class.

public App()

{

InitializeComponent();

//MainPage = new MainPage();

//MainPage = new NavigationPage(new RegisterPage());

SetMainPage();

}

private void SetMainPage()

{

if (!string.IsNullOrEmpty(Settings.AccessToken))

{

MainPage = new NavigationPage(new EmployeesPage());

}

else if (!string.IsNullOrEmpty(Settings.Username) && string.IsNullOrEmpty(Settings.Password))

{

MainPage = new NavigationPage(new LoginPage());

}

else

{

MainPage = new NavigationPage(new RegisterPage());

}

}

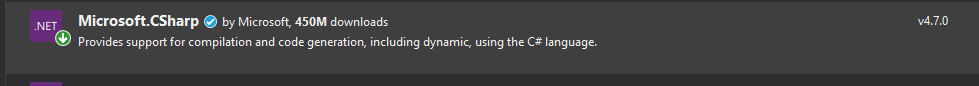
Added the following code in the “EmployeesPage” Content Page for testing.

<StackLayout Orientation="Vertical">

<Button Command="{Binding GetEmployeesCommand}" Text="Get All Employees" />

<ListView ItemsSource="{Binding EmployeeBindingModel}" HasUnevenRows="True">

Afterwards install the “Microsoft.CSharp” NuGet Packages in the “EmployeeManagementXamarinFormsApp” (as well for Android, iOS & UWP projects)via the NuGet Package Manager.



5. Add the following code in the “ApiServices” Class.

//\*\*\*Adding New Employees

public async Task PostEmployeesAsync(EmployeeBindingModel employeeBindingModel,string accessToken)

{

var client = new HttpClient();

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", accessToken);

var json = JsonConvert.SerializeObject(employeeBindingModel);

HttpContent content = new StringContent(json);

content.Headers.ContentType = new MediaTypeHeaderValue("application/json");

/\*var response = \*/await client.PostAsync("http://192.168.0.139:800/api/Employees", content);

}

Add a new Class “AddNewEmployeeViewModel” in the “ViewModels” folder.



Add the following code in the Class “AddNewEmployeeViewModel”.

public class AddNewEmployeeViewModel

{

ApiServices \_apiServices = new ApiServices();

public string TbFirstName { get; set; }

public string TbSurname { get; set; }

public string TbTellNo { get; set; }

public string TbEmail { get; set; }

public ICommand AddEmployeeCommand

{

get

{

return new Command(async () =>

{

var accesstoken = Settings.AccessToken;

var employee = new EmployeeBindingModel

{

TbFirstName = TbFirstName,

TbSurname = TbSurname,

TbTellNo = TbTellNo,

TbEmail = TbEmail

};

await \_apiServices.PostEmployeesAsync(employee,accesstoken);

});

}

}

}

Add a new Forms Blank Content Page Xaml “AddNewEmployeePage” in the “Views” folder.



Add the following code in the “AddNewEmployeePage” Content Page.

xmlns:viewModels="clr-namespace:EmployeeManagementXamarinFormsApp.ViewModels">

<ContentPage.BindingContext>

<viewModels:AddNewEmployeeViewModel />

</ContentPage.BindingContext>

<StackLayout Orientation="Vertical">

<Entry Text="{Binding TbFirstName}" />

<Entry Text="{Binding TbSurname}" />

<Entry Text="{Binding TbTellNo}" />

<Entry Text="{Binding TbEmail}" />

<Button Command="{Binding AddEmployeeCommand}" Text="Add New Employee" />

</StackLayout>

Add the following code in the “EmployeesPage” Content Page.

<Button Command="{Binding GetEmployeesCommand}" Text="Get All Employees" />

<Button Text="Add New Employee" Clicked="GoToAddNewEmployeePage\_Clicked"/>

<ListView ItemsSource="{Binding Employees}" HasUnevenRows="True">

And the following in the “EmployeesPage.xaml.cs” Content Page Class and test the app.

private async void GoToAddNewEmployeePage\_Clicked(object sender, EventArgs e)

{

await Navigation.PushAsync(new AddNewEmployeePage());

}

6. Add the following code in the “ApiServices” Class.

//\*\*\*Putting (Updating) a Employee

public async Task PutEmployeeAsync(EmployeeBindingModel employeeBindingModel, string accessToken)

{

var client = new HttpClient();

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", accessToken);

var json = JsonConvert.SerializeObject(employeeBindingModel);

HttpContent content = new StringContent(json);

content.Headers.ContentType = new MediaTypeHeaderValue("application/json");

/\*var response = \*/ await client.PutAsync("http://172.20.32.1:800/api/Employees/" + employeeBindingModel.EmployeeID, content);

}

Add a new Class “EditEmployeeViewModel” in the “ViewModels” folder.



Add the following code in the Class “EditEmployeeViewModel”.

public class EditEmployeeViewModel

{

ApiServices \_apiServices = new ApiServices();

public EmployeeBindingModel employeeBindingModel { get; set; }

public ICommand PutEmployeeCommand

{

get

{

return new Command(async () =>

{

var accesstoken = Settings.AccessToken;

await \_apiServices.PutEmployeeAsync(employeeBindingModel, accesstoken);

});

}

}

}

Add a new Forms Blank Content Page Xaml “EditEmployeePage” in the “Views” folder.



Add the following code in the “EditEmployeePage” Content Page.

xmlns:viewModels="clr-namespace:EmployeeManagementXamarinFormsApp.ViewModels">

<!--<ContentPage.BindingContext>

<viewModels:EditEmployeeViewModel />

</ContentPage.BindingContext>-->

<StackLayout Orientation="Vertical">

<Entry Text="{Binding TbFirstName}" />

<Entry Text="{Binding TbSurname}" />

<Entry Text="{Binding TbTellNo}" />

<Entry Text="{Binding TbEmail}" />

<Button Command="{Binding PutEmployeeCommand}" Text="Update" />

</StackLayout>

Add the following in the “EmployeesPage.xaml” Content Page.

<StackLayout Orientation="Vertical">

<Button Command="{Binding GetEmployeesCommand}" Text="Get All Employees" />

<Button Text="Add New Employee" Clicked="GoToAddNewEmployeePage\_Clicked"/>

<ListView ItemsSource="{Binding Employees}" HasUnevenRows="True" ItemTapped="GoToEditEmployeePage\_ItemTapped">

<ListView.ItemTemplate>

<DataTemplate>

Add the following in the “EmployeesPage.xaml.cs” Content Page Class.

private async void GoToEditEmployeePage\_ItemTapped(object sender, ItemTappedEventArgs e)

{

var employeeBindingModel = e.Item as EmployeeBindingModel;

await Navigation.PushAsync(new EditEmployeePage(employeeBindingModel));

}

Add the following in the “EditEmployeePage.xaml.cs” Content Page Class and test app.

public EditEmployeePage(EmployeeBindingModel employeeBindingModel)

{

var editEmployeeViewModel = BindingContext as EditEmployeeViewModel;

editEmployeeViewModel.employeeBindingModel = employeeBindingModel;

BindingContext = editEmployeeViewModel;

InitializeComponent();

//var editEmployeeViewModel = BindingContext as EditEmployeeViewModel;

//editEmployeeViewModel.employeeBindingModel = employeeBindingModel;

}

7. Add the following code in the “ApiServices” Class.

//\*\*\*Deleting a Employee

public async Task DeleteEmployeeAsync(int id, string accessToken)

{

var client = new HttpClient();

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", accessToken);

/\*var response = \*/ await client.DeleteAsync("http://test-api.com/api/Employees/" + id);

}

Add the following in the “EditEmployeePage.xaml” Content Page.

<StackLayout Orientation="Vertical">

<Entry Text="{Binding TbFirstName}" />

<Entry Text="{Binding TbSurname}" />

<Entry Text="{Binding TbTellNo}" />

<Entry Text="{Binding TbEmail}" />

<Button Command="{Binding PutEmployeeCommand}" Text="Update" />

<Button Command="{Binding DeleteEmployeeCommand}" Text="Delete" />

</StackLayout>

Add the following code in the Class “EditEmployeeViewModel”.

public ICommand DeleteEmployeeCommand

{

get

{

return new Command(async () =>

{

var accesstoken = Settings.AccessToken;

await \_apiServices.DeleteEmployeeAsync(employeeBindingModel.EmployeeID, accesstoken);

});

}

}

8. Add the following code in the “ApiServices” Class.

*(Employee search feature will be written here)*

9. Update the following code in the “ApiServices” Class, for “LoginAsync” method.

var jwt = await response.Content.ReadAsStringAsync();

JObject jwtDynamic = JsonConvert.DeserializeObject<dynamic>(jwt);

var accessToken = jwtDynamic.Value<string>("access\_token");

var accessTokenExpiration = jwtDynamic.Value<DateTime>(".expires");

Settings.AccessTokenExpiration = accessTokenExpiration;

Debug.WriteLine(jwt);

return accessToken;

Add the following in the “EmployeesPage.xaml” Content Page.

<ContentPage.BindingContext>

<viewModels:EmployeesViewModel />

</ContentPage.BindingContext>

<ContentPage.ToolbarItems>

<ToolbarItem Text="Logout" Command="{Binding LogoutCommand}" Clicked="Logout\_Clicked" />

</ContentPage.ToolbarItems>

Add the following code in the Class “EmployeesViewModel”.

public ICommand LogoutCommand

{

get

{

return new Command(() =>

{

Settings.AccessToken = "";

Settings.Username = "";

Settings.Password = "";

});

}

}

Add the following in the “EmployeesPage.xaml.cs” Content Page Class.

private async void Logout\_Clicked(object sender, EventArgs e)

{

await Navigation.PushAsync(new LoginPage());

}

Then add this code into the “Settings” Class.

public static DateTime AccessTokenExpiration

{

get

{

return AppSettings.GetValueOrDefault("AccessTokenExpiration", DateTime.UtcNow);

}

set

{

AppSettings.AddOrUpdateValue("AccessTokenExpiration", value);

}

}

Update the following code in the “App.Xaml.cs” Class.

private void SetMainPage()

{

if (!string.IsNullOrEmpty(Settings.AccessToken))

{

if (DateTime.UtcNow > Settings.AccessTokenExpiration)

{

var vm = new LoginViewModel();

vm.LoginCommand.Execute(null);

}

MainPage = new NavigationPage(new EmployeesPage());

}