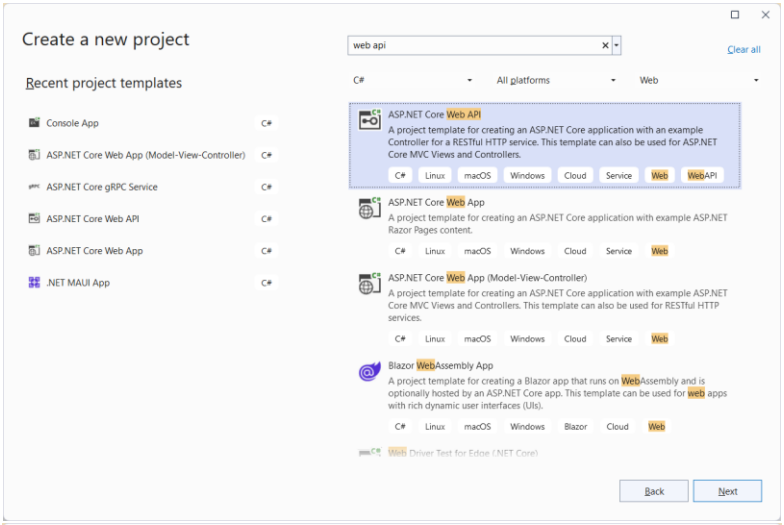
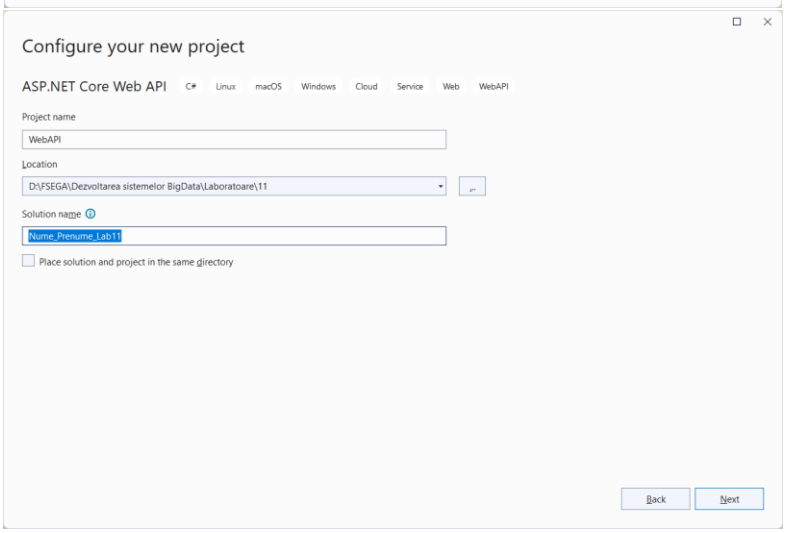
Laborator 11 – Aplicatii multi-platforma cu .NET MAUI - Web API ca si back-end pentru o aplicatie mobila

In laboratorul curent, vom rescrie aplicatia realizata in laboratorul 8 pentru a utiliza o baza de date centralizata MS SQL (in laboratorul 8 am utilizat o baza de date locala pe dispozitivul fizic/emulator). Vom crea un servciu Web care va realiza operatiile CRUD in baza de date MS SQL si va fi utilizat de aplicatia mobila .NET MAUI.

1. Deschidem Visual Studio si cream un nou proiect de tipul ASP.NET Core Web API pe care il denumim WebAPI si il salvam intr-o noua solutie cu numele Nume_Pren_Lab11

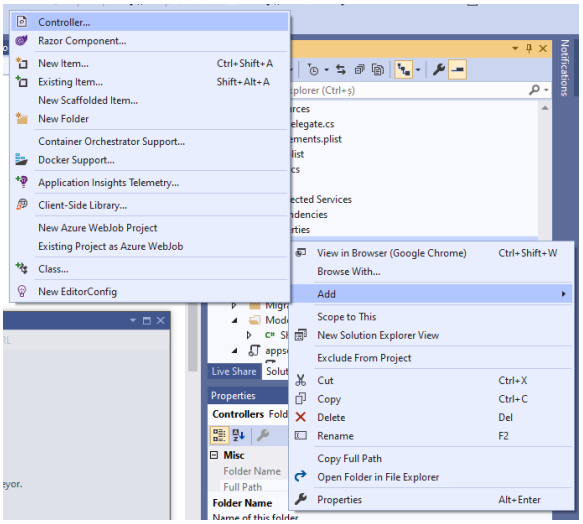




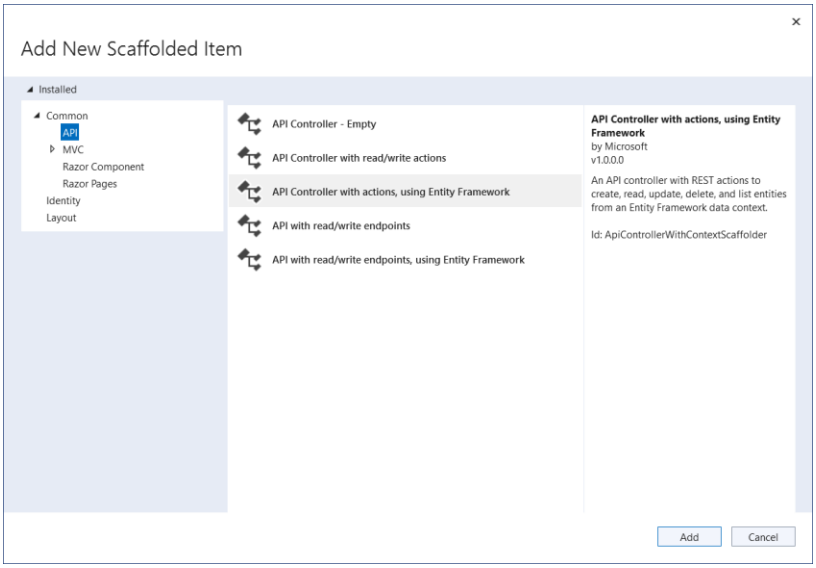
2. In proiectul nou creat adaugam un nou folder cu denumirea Models (click dreapta pe numele proiectului->Add->New Folder)
3. Adaugam o clasa in folderul: **Models** astfel: click-dreapta pe numele folderului Add->Class si ii dam numele **ShopList** , care va avea urmatorul continut:

```
public class ShopList
    {

        public int ID { get; set; }
        public string Description { get; set; }
        public DateTime Date { get; set; }

    }
```

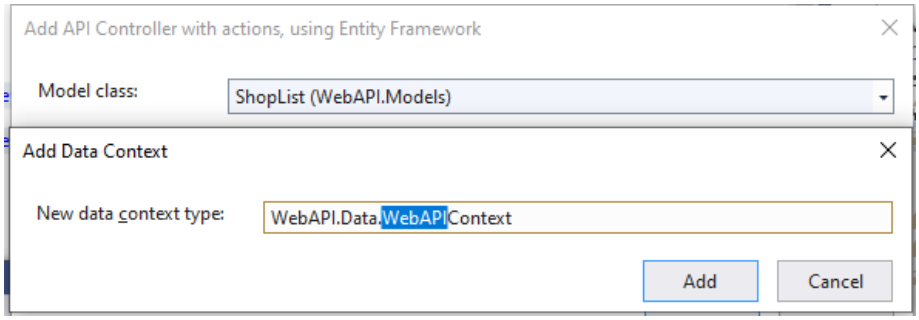
4. Adaugam un Controller (click dreapta pe numele folderului Controllers->Add->Controller)



5. Alegem API Controller with actions using Entity Framework

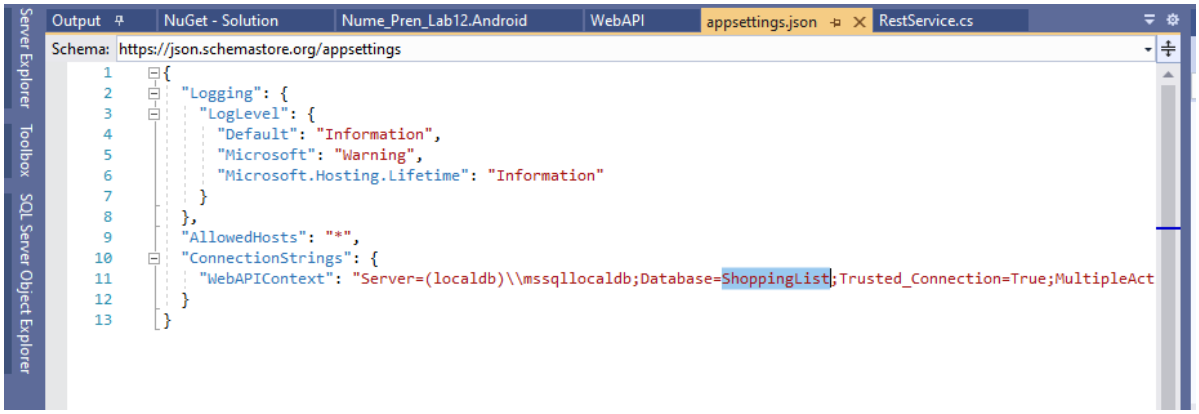


6. In fereastra urmatoare alegem modelul pe baza caruia se va crea controller-ul (WebAPI/Models/ShopList.cs) si apasam butonul “+” pentru a crea o clasa Context care se va numi WebAPIContext, iar controllerul se va numi ShopListsController.

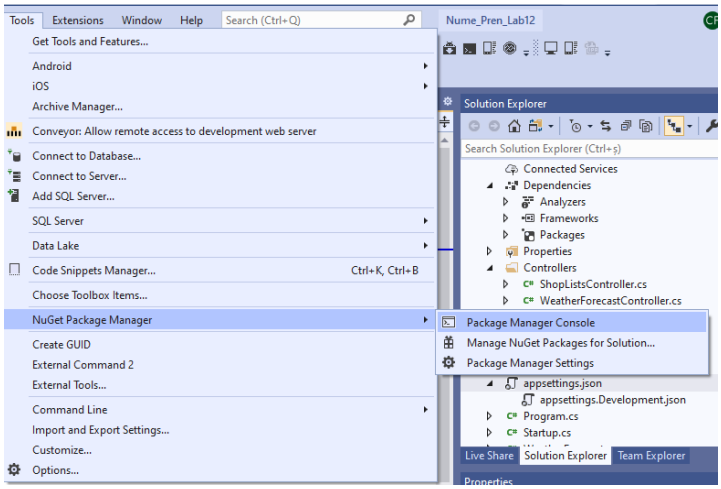


Dupa apasarea butonului Add, observam ca s-au generat automat controller-ul ShopListsController si clasa WebAPIContext.

7. In fisierul appsettings.json modificam numele generat automat al bazei date cu numele ShoppingList



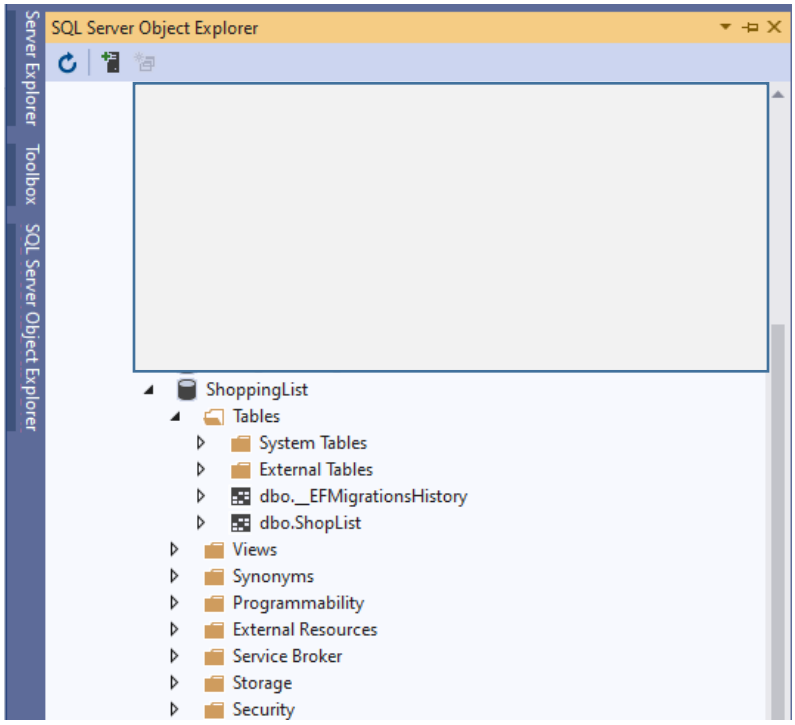
8. Deschidem PMC din meniul Tools->NuGet Package Manager->Package Manager Console



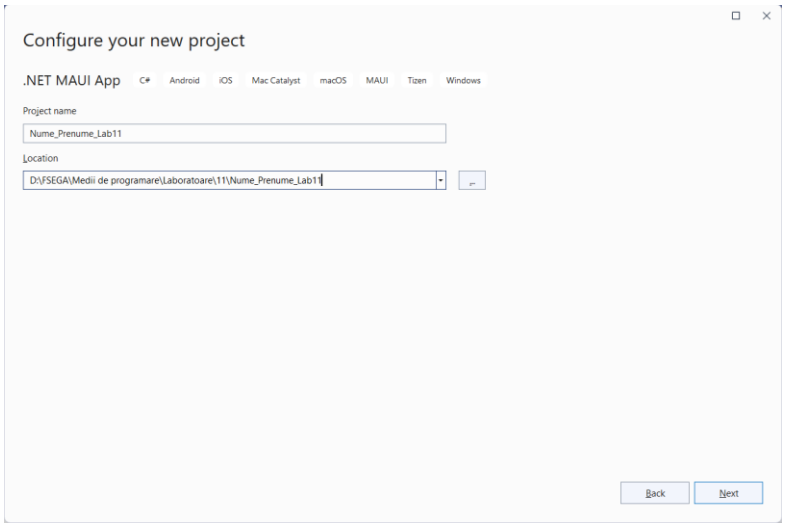
9. Migram modelul creat in baza de date ShoppingList astfel:

Cream un script cu denumirea InitialCreate utilizand comanda : add-migration InitialCreate
Rulam scriptul cu comanda: update-database

Daca nu avem erori vom observa ca s-a creat baza de date ShoppingList in fereastra SQL Server Object Explorer

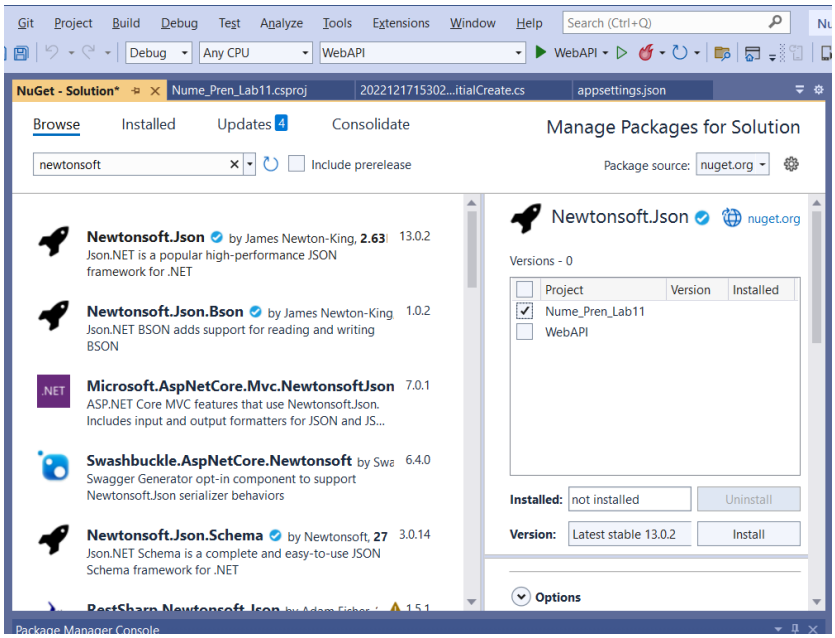


10. In cadrul solutiei existente Nume_Pren_Lab11 cream un nou proiect de tipul .NET MAUI (in Solution Explorer click dreapta pe numele solutiei->Add->New Project) cu numele Nume_Pren_Lab11



11. Serviciul web utilizeaza mesaje de tip JSON pentru a returna date aplicatiei client. Vom instala libraria NewtonSoft pentru a serialize/deserializa mesajele. Astfel de la Meniul Tools alegem Nuget

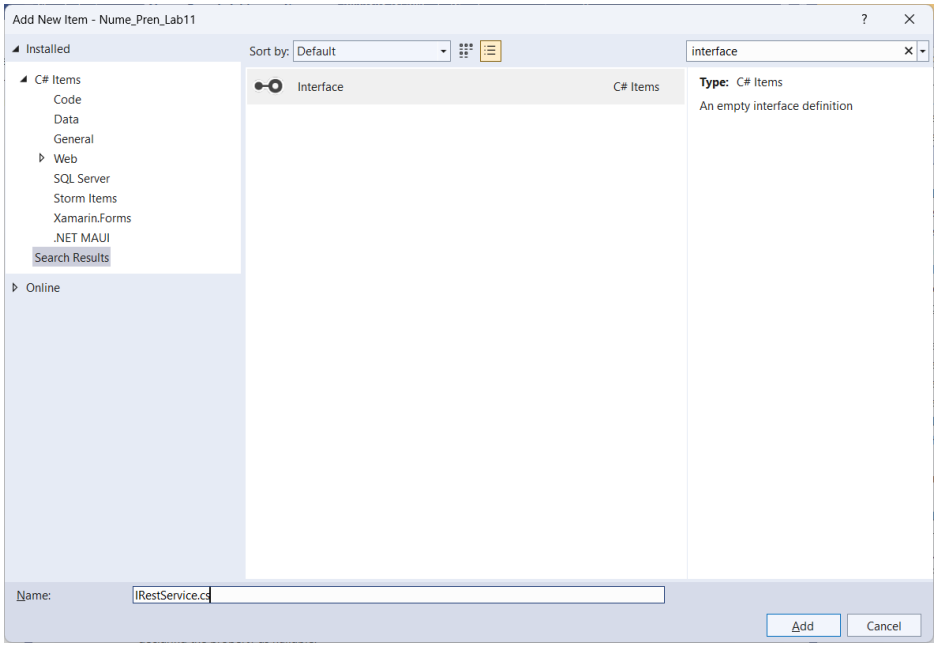
Package Manager->Manage Nuget Packages for Solution cautam libraria NewtonSoft.Json si o instalam pentru proiectul Nume_Pren_Lab11.



12. In proiectul .NET MAUI nou creat, adaugam un nou folder cu denumirea **Models** (click dreapta pe numele proiectului->Add->New Folder)
13. Adaugam o clasa in folderul: **Models** astfel: click-dreapta pe numele folderului Add->Class si ii dam numele **ShopList** , care va avea urmatorul continut:

```
public class ShopList
    {

        public int ID { get; set; }
        public string Description { get; set; }
        public DateTime Date { get; set; }

    }
```

14. In proiectul .NET MAUI adaugam un nou folder cu denumirea **Data** (click dreapta pe numele proiectului->Add->New Folder)
15. In folderul Data adaugam o interfata (click dreapta pe numele folderului ->Add->New Item si alegem Interface pe care o denumim IRestService



16. Aceasta va avea urmatorul continut:

```
using System.Threading.Tasks;
using Nume_Pren_Lab11.Models;


namespace Nume_Pren_Lab11.Data
{
	public interface IRestService
	{
		Task<List<ShopList>> RefreshDataAsync();

		Task SaveShopListAsync(ShopList item, bool isNewItem);

		Task DeleteShopListAsync(int id);
	}
}
```

17. In acelasi folder Data cream o noua clasa RestService.cs care va implementa interfata creata mai sus:

```
using Newtonsoft.Json;
using System.Threading.Tasks;
using Nume_Pren_Lab11.Models;

namespace Nume_Pren_Lab11.Data
{
    public class RestService : IRestService
    {
```

```
        HttpClient client;

       //se va modifica ulterior cu ip-ul si portul corespunzator
        string RestUrl = "https://192.169.0.8:45455/api/shoplists/{0}";

        public List<ShopList> Items { get; private set; }

        public RestService()
        {
            var httpClientHandler = new HttpClientHandler();

            httpClientHandler.ServerCertificateCustomValidationCallback =
            (message, cert, chain, errors) => { return true; };

            client = new HttpClient(httpClientHandler);

        }

        public async Task<List<ShopList>> RefreshDataAsync()
        {
            Items = new List<ShopList>();

            Uri uri = new Uri(string.Format(RestUrl, string.Empty));
            try
            {
                HttpResponseMessage response = await client.GetAsync(uri);
                if (response.IsSuccessStatusCode)
                {
                    string content = await response.Content.ReadAsStringAsync();
                    Items = JsonConvert.DeserializeObject<List<ShopList>>(content);
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(@"\tERROR {0}", ex.Message);
            }

            return Items;
        }

        public async Task SaveShopListAsync(ShopList item, bool isNewItem = true)
        {
            Uri uri = new Uri(string.Format(RestUrl, string.Empty));

            try
```

```
            {
                string json = JsonConvert.SerializeObject(item);
                StringContent content = new StringContent(json, Encoding.UTF8,
"application/json");

                HttpResponseMessage response = null;
                if (isNewItem)
                {
                    response = await client.PostAsync(uri, content);
                }
                else
                {
                    response = await client.PutAsync(uri, content);
                }

                if (response.IsSuccessStatusCode)
                {
                    Console.WriteLine(@"\tTodoItem successfully saved.");
                }

            }
            catch (Exception ex)
            {
                Console.WriteLine(@"\tERROR {0}", ex.Message);
            }
        }

        public async Task DeleteShopListAsync(int id)
        {
            Uri uri = new Uri(string.Format(RestUrl, id));

            try
            {
                HttpResponseMessage response = await client.DeleteAsync(uri);

                if (response.IsSuccessStatusCode)
                {
                    Console.WriteLine(@"\tTodoItem successfully deleted.");
                }

            }
            catch (Exception ex)
            {
                Console.WriteLine(@"\tERROR {0}", ex.Message);
            }
```

```
        }
    }
}
```

18. In fisierul App.xaml.cs modificam clasa App astfel:

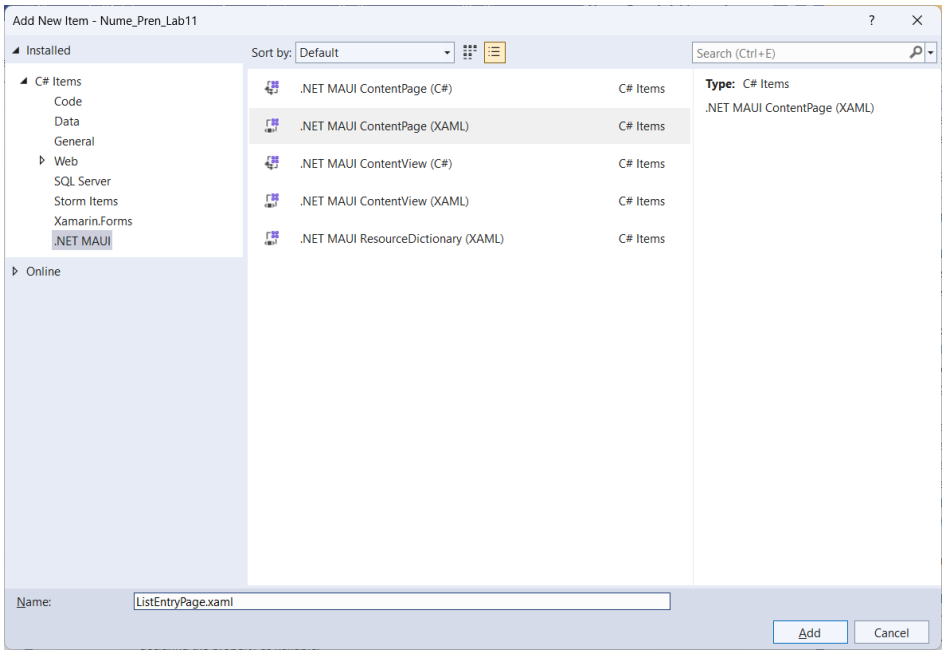
```
using Nume_Pren_Lab11.Data;

namespace Nume_Pren_Lab11
{
    public partial class App : Application
    {
        public static ShoppingListDatabase Database { get; private set; }

        public App()
        {
            Database = new ShoppingListDatabase(new RestService());
            MainPage = new NavigationPage(new ListEntryPage());
        }
```

19. In folderul Data cream clasa ShoppingListDatabase.cs care va avea urmatorul continut:

```
using System.Threading.Tasks;
using Nume_Pren_Lab11.Models;

namespace Nume_Pren_Lab11.Data
{
   public class ShoppingListDatabase
    {
        IRestService restService;

        public ShoppingListDatabase(IRestService service)
        {
            restService = service;

        }

        public Task<List<ShopList>> GetShopListsAsync()
        {
            return restService.RefreshDataAsync();
        }


        public Task SaveShopListAsync(ShopList item, bool isNewItem = true)
        {
```

```
            return restService.SaveShopListAsync(item, isNewItem);
        }

        public Task DeleteShopListAsync(ShopList item)
        {
            return restService.DeleteShopListAsync(item.ID);
        }

    }

}
```

20. In proiectul Nume_Pren_Lab11 adaugam o pagina de tip ContentPage. In fereastra Solution Explorer facem click dreapta pe numele proiectului Nume_Pren_Lab11, selectam Add->New Item si din categoria .NET MAUI alegem .NET MAUI ContentPage(XAML) pe care o denumim **ListEntryPage.xaml**



21. Vom utiliza un ListView care utilizeaza Databinding pentru a afisa toate listele de cumparaturi din aplicatie, selectarea uneia dintre aceste liste va duce la ListPage care permite modificarea acesteia. Vom folosi ToolbarItem pentru a crea o noua nota la apasarea acestuia. Din fereastra **Solution Explorer** deschidem fisierul ListEntryPage.xaml si modificam continutul astfel:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Nume_Pren_Lab11.ListEntryPage"
             Title="ListEntryPage">
<ContentPage.ToolbarItems>
```

```
        <ToolbarItem Text="Add Shopping List"
                     Clicked="OnShopListAddedClicked" />
    </ContentPage.ToolbarItems>
    <ListView x:Name="listView"
              Margin="20"
              ItemSelected="OnListViewItemSelected">
        <ListView.ItemTemplate>
            <DataTemplate>
                <TextCell Text="{Binding Description}"
                          Detail="{Binding Date}" />
            </DataTemplate>
        </ListView.ItemTemplate>
    </ListView>
</ContentPage>
```

22. Din fereastra **Solution Explorer** deschidem fisierul ListEntryPage.xaml.cs si adaugam la continutul acestuia codul de mai jos:

```
...
public ListEntryPage()
    {
        InitializeComponent();
    }

protected override async void OnAppearing()
        {
            base.OnAppearing();

            listView.ItemsSource = await App.Database.GetShopListsAsync();
        }

async void OnShopListAddedClicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ListPage
            {
                BindingContext = new ShopList()
            });
        }

async void OnListViewItemSelected(object sender, SelectedItemChangedEventArgs e)
        {
            if (e.SelectedItem != null)
            {
                await Navigation.PushAsync(new ListPage
                {
                    BindingContext = e.SelectedItem as ShopList
                });
            }

    }
```

23. In proiectul Nume_Pren_Lab11 adaugam o pagina de tip ContentPage. In fereastra Solution Explorer facem click dreapta pe numele proiectului Nume_Pren_Lab11, selectam Add->New Item si din categoria .NET MAUI alegem .NET MAUI ContentPage (XAML) pe care o denumim **ListPage.xaml**

24. Vom utiliza un Editor pentru a introduce textul si doua butoane Save si Delete. Cele doua butoane sunt dispuse orizontal intr-un grid cu doua coloane.

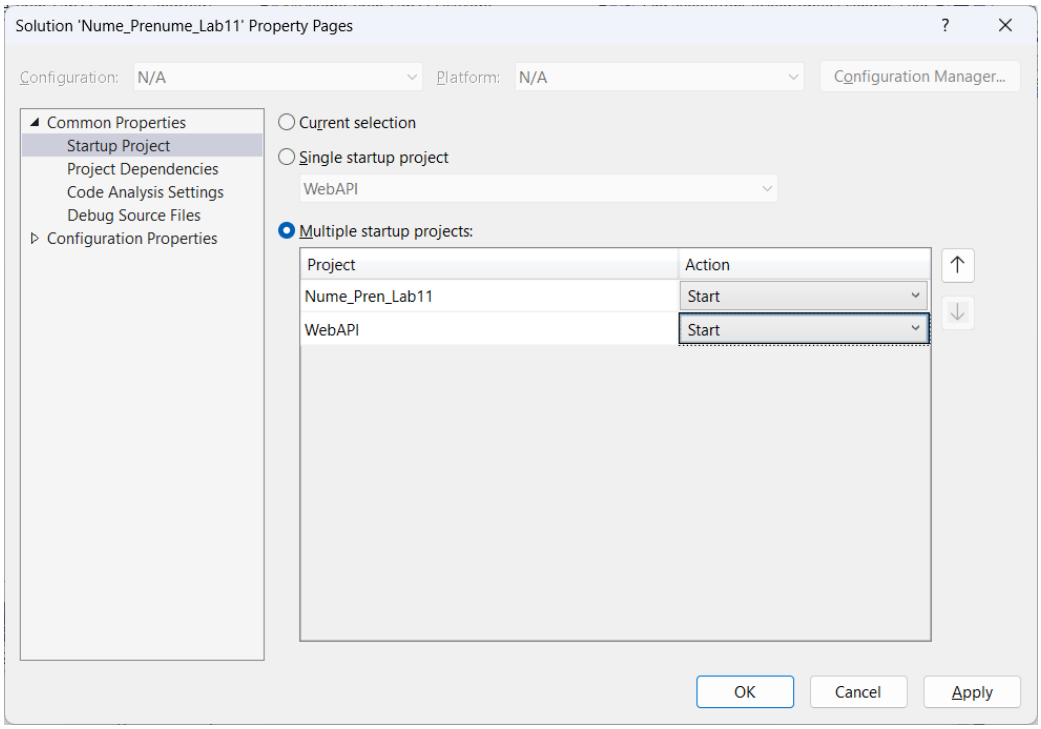
```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Nume_Pren_Lab11.ListPage"
             Title="ListPage">
<ContentPage.Content>
<StackLayout Margin="20">
        <Editor Placeholder="Enter the description of the shopping list"
                Text="{Binding Description}"
                HeightRequest="100" />
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>
            <Button Text="Save"
                    Clicked="OnSaveButtonClicked" />
            <Button Grid.Column="1"
                    Text="Delete"
                    Clicked="OnDeleteButtonClicked"/>
        </Grid>
    </StackLayout>
</ContentPage.Content>
</ContentPage>
```

25. Din fereastra **Solution Explorer** deschidem fisierul ListPage.xaml.cs si adaugam la continutul acestuia codul de mai jos. Cand este apasat butonul Save, handler-ul de eveniment `OnSaveButtonClicked` este executat, lista de cumparaturi este salvata in baza de date si aplicatia navigheaza la pagina precedenta. La executarea handlerului de eveniment `OnDeleteButtonClicked` , lista de cumparaturi este stearsa din baza de date si aplicatia navigheaza la pagina precedenta .

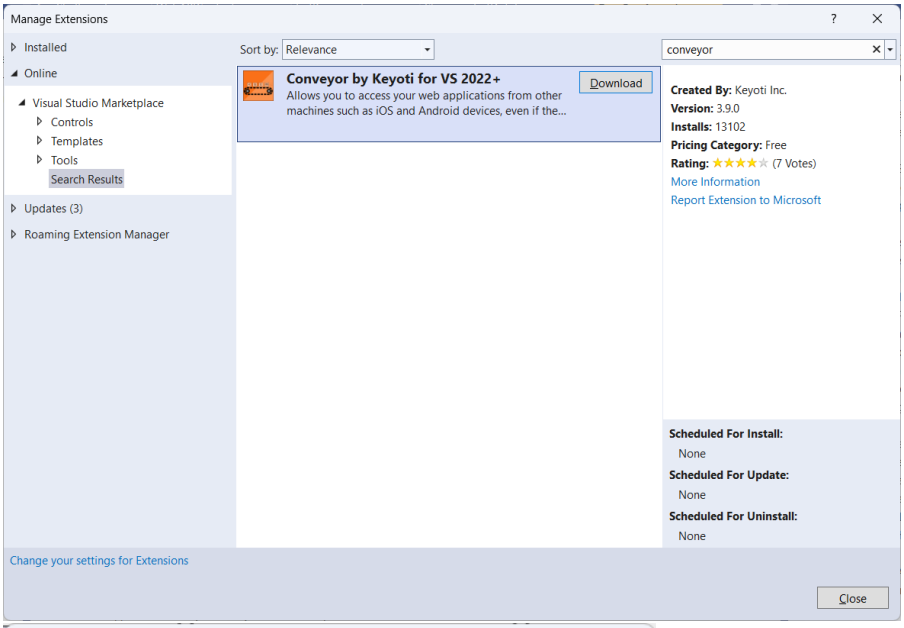
```
using Nume_Pren_Lab10.Models;
…
…
public ListPage()
       {
              InitializeComponent();
       }
async void OnSaveButtonClicked(object sender, EventArgs e)
{
    var slist = (ShopList)BindingContext;
    slist.Date = DateTime.UtcNow;
```

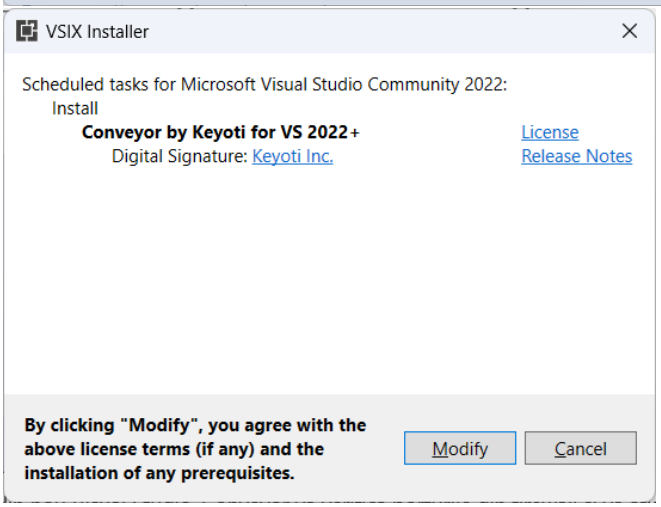
```
    await App.Database.SaveShopListAsync(slist);
    await Navigation.PopAsync();
}

async void OnDeleteButtonClicked(object sender, EventArgs e)
{
    var slist = (ShopList)BindingContext;
    await App.Database.DeleteShopListAsync(slist);
    await Navigation.PopAsync();
}
```

26. Deoarece e nevoie ca serviciul web sa ruleze atunci cand este accesat de aplicatia client .NET MAUI, setam solutia ca avand proiecte start-up multiple. In Solution Explorer, click dreapta pe numele solutiei Nume_Pren_Lab11->Properties si setam atat proiectul WebAPI cat si Nume_Pren_Lab11 sa porneasca.

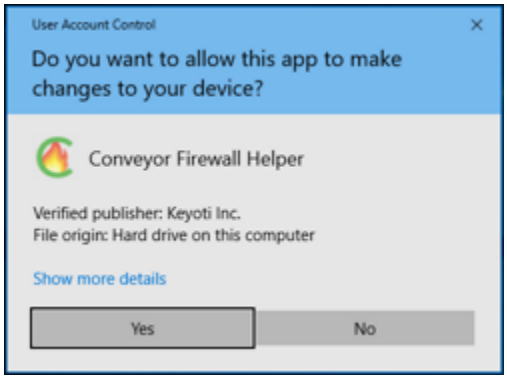


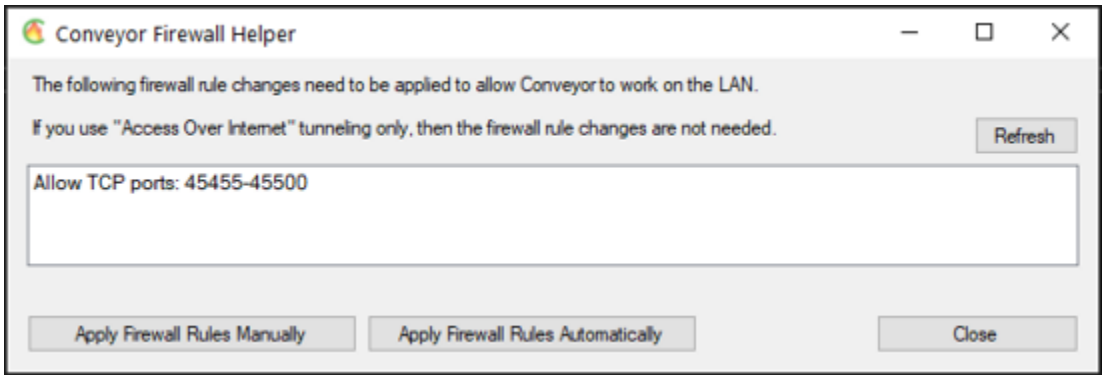
27. Pentru a accesa serviciul web care va rula local din emulator/dispozitiv mobile vom instala o extensie pentru Visual Studio – Conveyor. Astfel de la meniul Extensions alegem Manage Extensions si cautam extensia Conveyor. Pentru a se instala trebuie sa inchidem Visual Studio si se va porni instalarea.





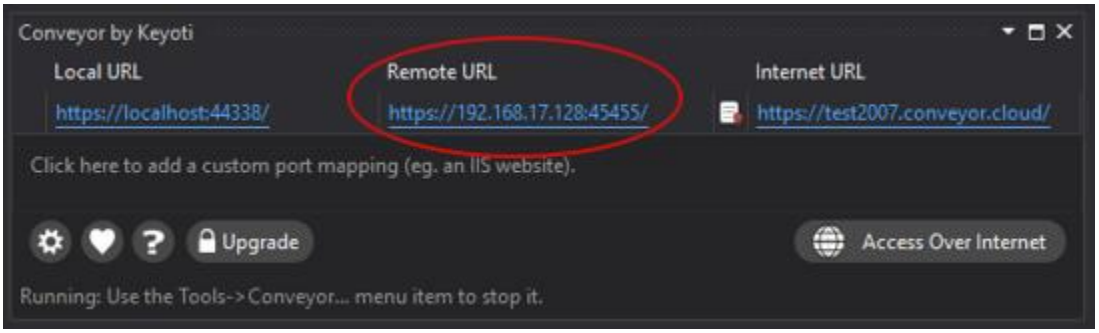
28. Deschidem din nou Visual Studio, Conveyor va verifica porturile din firewall si va afisa o notificare cu setarile necesare.





Alegem "Apply Firewall Rules Automatically"

29. Rulam aplicatia si va aparea Remote URL. Acesta trebuie inlocuit in clasa RestService realizata la punctul 18.



30. Dupa ce am inlocuit URL-ul in clasa RestService, rulam aplicatia (pe Windows Machine si pe Android Emulator) si testam operatiile CRUD pentru ShopList realizate acum prin intermediul serviciului Web
31. Aplicatia poate fi dezvoltata in continuare pe acesta arhitectura, urmand pasii similari din laboratoarele 9-10 si facand analogiile necesare.