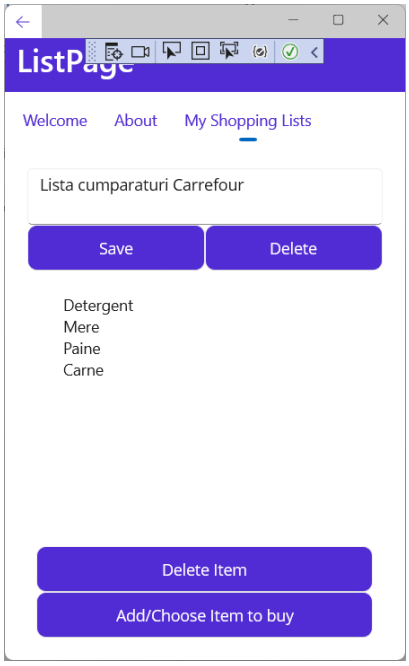
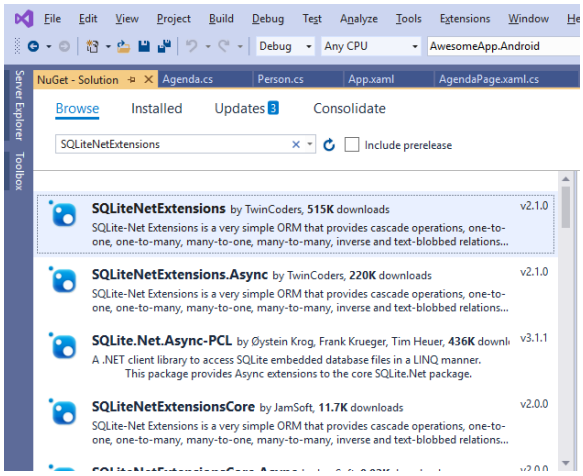
Laborator 9 – Aplicatii mobile multi-platforma cu .NET MAUI

1. In laboratorul curent, vom dezvolta aplicatia realizata in laboratorul 7 si 8. Deschidem Visual Studio si apoi alegem optiunea **Open a project or solution** si cautam proiectul **Nume_Pren_Lab7,** creat anterior. Laboratorul curent il vom dezvolta pe un branch nou. Utilizand pasii indicati in Lab 1, pct. 22 vom crea un branch nou (based on Laborator8) pe care il vom denumi Laborator9.
2. Vom modifica aplicatia realizata in lab 7 si 8 astfel incat produsele pe care dorim sa le cumparam sa nu fie adaugate sub forma de text ci sa le putem alege dintr-o lista de produse predefinite



3. Pentru a crea relatii intre tabele este nevoie sa utilizam adnotari din pachetul **SQLiteNetExtensions**. Astfel, din meniul **Tools ->NuGet Package Manager->Manage NuGet Packages for Solution** si instalam **SQLiteNetExtensions**



4. Adaugam o clasa in folderul: **Models** astfel: click-dreapta pe numele folderului Add->Class si ii dam numele **Product** , care va avea urmatorul continut:

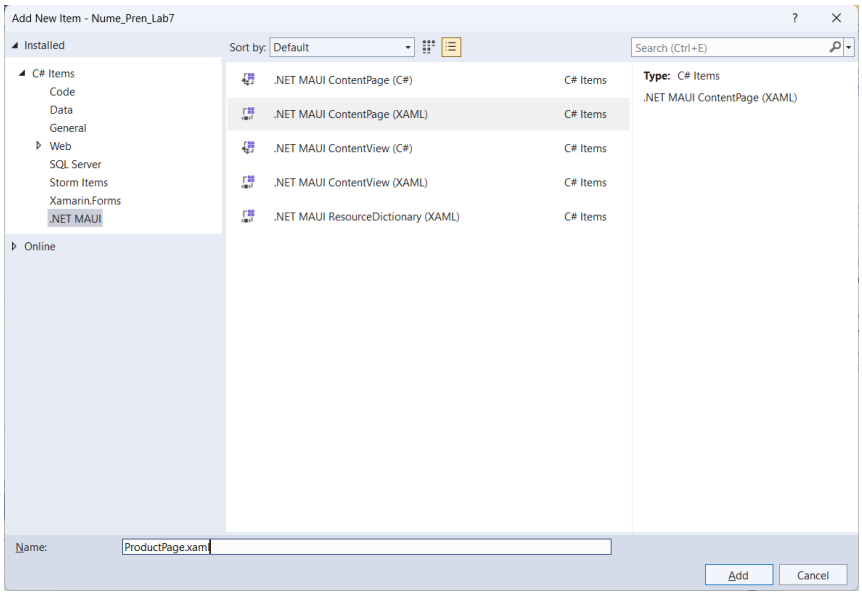
```
using SQLite;
using SQLiteNetExtensions.Attributes;

namespace Nume_Pren_Lab7.Models
{
    public class Product
    {
            [PrimaryKey, AutoIncrement]
             public int ID { get; set; }

             public string Description { get; set; }

             [OneToMany]
             public List<ListProduct> ListProducts { get; set; }
    }
}
```

Aceasta clasa defineste un model Product care va fi folosit pentru a adauga produse in lista de cumparaturi. Proprietatea ListProducts este marcata ca fiind `[OneToMany]` pentru a modela relatia de tip 1-n cu ListProduct

5. Adaugam o clasa in folderul: **Models** astfel: click-dreapta pe numele folderului Add->Class si ii dam numele **ListProduct** , care va avea urmatorul continut:

```
using SQLite;
using SQLiteNetExtensions.Attributes;

namespace Nume_Pren_Lab7.Models
{
            public class ListProduct
    {
            [PrimaryKey, AutoIncrement]
             public int ID { get; set; }

             [ForeignKey(typeof(ShopList))]
             public int ShopListID { get; set; }

             public int ProductID { get; set; }
    }

}
```

6. In proiectul Nume_Pren_Lab7 adaugam o pagina de tip ContentPage. In fereastra Solution Explorer facem click dreapta pe numele proiectului Nume_Pren_Lab7, selectam Add->New Item si din categoria .NET MAUI alegem ContentPage pe care o denumim **ProductPage.xaml**



7. Pagina ProductPage o vom utiliza pentru a crea o lista cu produse predefinite, produse pe care utilizatorul aplicatiei le cumpara frecvent. In cadrul acestei pagini vom avea un control de tip Editor in care vom introduce numele produsului, un buton de Save si Delete si un control Listview din care vom putea alege produsele introduse. Modificam continutul fisierului ProductPage.xaml 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_Lab7.ProductPage"
             Title="ProductPage">
<ContentPage.Content>
        <StackLayout Margin="20">
            <Editor Placeholder="Enter product name" Margin="20"
                Text="{Binding Description}"
                HeightRequest="50" />
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="Auto" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
                <Button Text="Save Product" MaximumWidthRequest="200"
                    Clicked="OnSaveButtonClicked" />
                <Button Grid.Column="1"
                    Text="Delete Product" MaximumWidthRequest="200"
                    Clicked="OnDeleteButtonClicked"/>
                <ListView Grid.Row="1" Grid.Column="0" x:Name="listView"
              Margin="20" IsRefreshing="True" >
                    <ListView.ItemTemplate>
                        <DataTemplate>
```

```
                            <TextCell Text="{Binding Description}" />
                        </DataTemplate>
                    </ListView.ItemTemplate>
                </ListView>
                <Button Grid.Row="2" Grid.Column="0" Text="Add to Shop List"
MaximumWidthRequest="200"
                    Clicked="OnAddButtonClicked" />
            </Grid>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

8. In fisierul **Data/ShoppingListDatabase** adaugam cod pentru crearea, citirea, editarea si stergerea datelor pentru clasa Product. Adaugam in Clasa **ShoppingListDatabase** urmatorul continut:

```
using SQLite;
using System.Collections.Generic;
using System.Threading.Tasks;
using Nume_Pren_Lab7.Models;

namespace Nume_Pren_Lab7.Data
{
    public class ShopListDatabase
    {
        readonly SQLiteAsyncConnection _database;

        public ShopListDatabase(string dbPath)
        {
            _database = new SQLiteAsyncConnection(dbPath);
            _database.CreateTableAsync<ShopList>().Wait();
            _database.CreateTableAsync<Product>().Wait();
            _database.CreateTableAsync<ListProduct>().Wait();
        }

public Task<int> SaveProductAsync(Product product)
        {
            if (product.ID != 0)
            {
                return _database.UpdateAsync(product);
            }
            else
            {
                return _database.InsertAsync(product);
            }
        }


        public Task<int> DeleteProductAsync(Product product)
        {
            return _database.DeleteAsync(product);
        }

        public Task<List<Product>> GetProductsAsync()
```

```
        {
            return _database.Table<Product>().ToListAsync();
        }
}
    }
```

9. In fisierul ProductPage.xmls.cs adaugam urmatorul cod:

```
    public partial class ProductPage : ContentPage
        {
            public ProductPage()
            {
                InitializeComponent();
            }

     async void OnSaveButtonClicked(object sender, EventArgs e)
            {
                var product = (Product)BindingContext;
                await App.Database.SaveProductAsync(product);
                listView.ItemsSource = await App.Database.GetProductsAsync();
            }

    async void OnDeleteButtonClicked(object sender, EventArgs e)
            {
                var product=listView.SelectedItem as Product;
                await App.Database.DeleteProductAsync(product);
                listView.ItemsSource = await App.Database.GetProductsAsync();
            }

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

                listView.ItemsSource = await App.Database.GetProductsAsync();
            }
    }
```

10. In fisierul ListPage.xaml adaugam un buton cu care vom putea naviga la pagina ProductPage de unde vom putea alege produse predefinite din lista. Pentru descrierea listei de cumparaturi vom micsora spatial alocat (la rulare se va introduce doar numele listei de cumparaturi (ex. Lista cumparaturi Cora), produsele vor fi adaugate dintr-o colectie de produse)

```
    <StackLayout Margin="20">
                <Editor Placeholder="Enter the description of the shopping list"
                    Text="{Binding Description}"
                    HeightRequest="50" />
<Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>
            <Button Text="Save"
                    Clicked="OnSaveButtonClicked" />
            <Button Grid.Column="1"
                    Text="Delete"
                    Clicked="OnDeleteButtonClicked"/>
```

```
            </Grid>
                <Button Text="Add/Choose Item to buy" MaximumWidthRequest="300"
    Clicked="OnChooseButtonClicked"/>
                <Grid>
    …
```

11. Avem nevoie ca atunci cand navigam la pagina ProductPage sa stim id-ul listei din ShopList, deci adaugam o proprietate de tip ShopList si modificam constructorul clasei ProductPage astfel:

```
public partial class ProductPage : ContentPage
    {
        ShopList sl;
        public ProductPage(ShopList slist)
        {
            InitializeComponent();
            sl = slist;
           }
```

12. In fisierul ListPage.xaml.cs adaugam un urmatorul cod care va naviga la pagina ProductPage:

```
    async void OnChooseButtonClicked(object sender, EventArgs e)
            {

                await Navigation.PushAsync(new ProductPage((ShopList)
    this.BindingContext)
                {
                    BindingContext = new Product()
                });

            }
```

13. Dorim ca dupa selectarea unui produs aceasta sa se adauge la lista de cumparaturi si sa se intoarca la pagina ListPage. Adaugam in fisierul ProductPage.xaml.cs urmatorul cod:

```
async void OnAddButtonClicked(object sender, EventArgs e)
    {

        Product p;
        if (listView.SelectedItem != null)
        {
            p = listView.SelectedItem as Product;
            var lp = new ListProduct()
            {
                ShopListID = sl.ID,
                ProductID = p.ID
            };
            await App.Database.SaveListProductAsync(lp);
            p.ListProducts = new List<ListProduct> { lp };

            await Navigation.PopAsync();
           }
```

14. In fisierul ListPage.xaml adaugam un ListView in care vom putea vedea produsele predefinite selectate si atasate listei respective de cumparaturi:

```
....
<Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>
            <Button Text="Save"
                    Clicked="OnSaveButtonClicked" />
            <Button Grid.Column="1"
                    Text="Delete"
                    Clicked="OnDeleteButtonClicked"/>
        </Grid>

        <ListView  x:Name="listView"
                Margin="20">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <TextCell  Detail="{Binding Description}" />
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>

        <Button Text="Add/Choose Item to buy" MaximumWidthRequest="300"
Clicked="OnChooseButtonClicked"/>
....
```

15. In fisierul ListPage.xaml.cs metoda OnAppearing va contine urmatorul cod:

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

            var shopl = (ShopList)BindingContext;

            listView.ItemsSource = await App.Database.GetListProductsAsync(shopl.ID);
           }
```

16. Adaugam in fisierul ShopListDatabase.cs:

```
public Task<int> SaveListProductAsync(ListProduct listp)
        {
            if (listp.ID != 0)
            {
                return _database.UpdateAsync(listp);
            }
            else
            {
                return _database.InsertAsync(listp);
            }
        }
```

```
public Task<List<Product>> GetListProductsAsync(int shoplistid)
        {
            return _database.QueryAsync<Product>(
          "select P.ID, P.Description from Product P"
          + " inner join ListProduct LP"
          + " on P.ID = LP.ProductID where LP.ShopListID = ?",
          shoplistid);

         }
```

17. Validarea datelor in .NET MAUI se poate face folosind Behaviours. Acestia permit adaugarea de functionalitati interfetei prin scrierea de clase si atasarea acestora controalelor in XAML.
18. Dorim ca atunci cand utilizatorul nu are descrierea introdusa pentru o lista de cumparaturi background-ul aferent controlului Editor sa se coloreze in Rosu. Adaugam o noua clasa in proiectul Nume_Pren_Lab7: click dreapta pe numele proiectului->Add Class si ii dam numele *ValidationBehaviour.* Aceasta va mosteni clasa de baza Behaviour si va avea urmatorul continut:

```
class ValidationBehaviour : Behavior<Editor>
       {

              protected override void OnAttachedTo(Editor entry)
              {
                     entry.TextChanged += OnEntryTextChanged;
                     base.OnAttachedTo(entry);

              }

              protected override void OnDetachingFrom(Editor entry)

              {

                     entry.TextChanged -= OnEntryTextChanged;
                     base.OnDetachingFrom(entry);

              }

              void OnEntryTextChanged(object sender, TextChangedEventArgs args)

              {

              ((Editor)sender).BackgroundColor =
string.IsNullOrEmpty(args.NewTextValue)?  Color.FromRgba("#AA4A44") :
Color.FromRgba("#FFFFFF");
              }
       }
```

19. Atasam Behaviour-ul creat la controlul de tip Editor din ListPage.xaml

```
        <ContentPage.Content xmlns:local="clr-namespace:Nume_Pren_Lab7">
            <StackLayout Margin="20">
    <Editor Placeholder="Enter the description of the shopping list"
                    Text="{Binding Description}"
                    HeightRequest="50" >
                    <Editor.Behaviors>

                        <local:ValidationBehaviour />

                    </Editor.Behaviors>
                </Editor>
```

Sarcina Laborator: Adaugati un buton cu textul “Delete Item” in pagina ListPage. Implementati functionalitatile aferente astfel incat butonul sa permita stergerea unui produs din lista de cumparaturi curenta.