Xamarin.Forms raamistiku õppematerjal iOS ja Android mobiilirakenduste arendamiseks

Autor: Tauri Taevik

Sisukord

Sissejuhatus 3

1. Xamarin.Forms projekti loomine 4

2. Ühe leheküljeline rakendus 8

2.1. Kasutajaliidese ning äriloogika failide ülesehitus 8

2.2. Klassi loomine 10

2.3. Andmete kuvamine ListView vaate abil. 12

2.4. Andmete lisamine ListView vaatesse 16

2.5. ObservableCollection loendi kasutamine 18

3. Mitme leheküljeline rakendus 21

3.1. Uue lehekülje loomine 21

3.2. Seadme navigatsiooni elementide lisamine 23

3.3. Andmete lisamine teiselt leheküljelt 27

3.4. OnAppearing() meetodi kasutamine 32

4. SQLite andmebaas 34

4.1. NuGetite lisamine 34

4.2. Sqlite seadistamine android platvormi jaoks 37

4.3. SQLite seadistamine iOS platvormi jaoks 40

4.4. SQLite funktsionaalsused klassis 42

4.5. Andmete sisestamine andmebaasi 43

4.6. Andmete lugemine andmebaasist 44

Soovitusi iseseisvaks õppimiseks 47

Sissejuhatus

Käesolev õppematerjal on mõeldud esmastele Xamarin.Forms raamistiku kasutajatele. Õppematerjali käigus luuakse andmesisestus tüüpi rakendus, mis töötab Android ning iOS platvormil. Õppematerjalis olevad juhised on loodud Windows arvutil ning rakenduse kuvahõived on loodud Android emulaatoril. Mac arvuti kasutajatele on välja toodud õppematerjali välised juhendid kohtades, mis sisaldavad erinevust Windows arvutil arendades.

Õppematerjal on jagatud neljaks peatükiks.

Esimene peatükk sisaldab endas projekti loomist, kus õpitakse looma Xamarin.Forms projekti iOS ja Android platvormi arenduseks ning antakse lühiülevaade projektist.

Teises peatükis luuakse ühe leheküljeline rakendus, mille käigus luuakse postituse lisamise rakendus, kus postituste lisamine ja kuvamine toimub ühelt leheküljelt. Õppijale seletatakse kasutajaliidese ja äriloogika failide ülesehitust. Õpitakse looma klasse ning erinevaid vaateid. Tutvustatakse põhjalikumalt loendi vaate kasutamist, mille käigus luuakse loendi vaate mall ning õpitakse andmete sidumist.

Kolmandas peatükis arendatakse edasi ühe leheküljelist rakendust mitmeleheküljeliseks rakenduseks nii, et postituste lisamine ja kuvamine toimub erinevatelt lehekülgedelt. Kolmandas peatükis õpitakse uue lehekülje loomist, lehekülgedevahelist navigeerimist ning mobiilseadme navigeerimiselementide lisamist. Lisaks õpitakse andmete lisamist teistelt lehekülgedelt ning OnAppearing meetodi kasutamist.

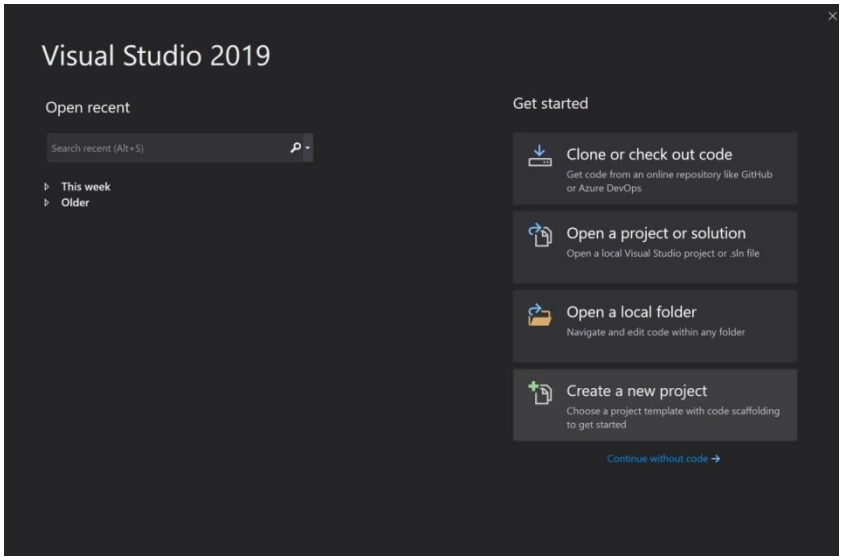
Viimases neljandas peatükis lisatakse rakendusele Sqlite andmebaas. Õpitakse lisama NuGet pakette. Tutvustatakse Sqlite andmebaasi paketi kasutamist, mille käigus õpitakse andmebaasi seadistamist iOS ja Android platvormidele, andmebaasi andmete lisamist, andmebaasist andmete kuvamist ning klassi siseseid SQLite paketi funktsionaalsuseid.

Koodinäidistel kasutatav roheline taustavärv tähendab uute koodiridade lisamist ning punane taustavärv tähendab koodiridade kustutamist.

1. Xamarin.Forms projekti loomine

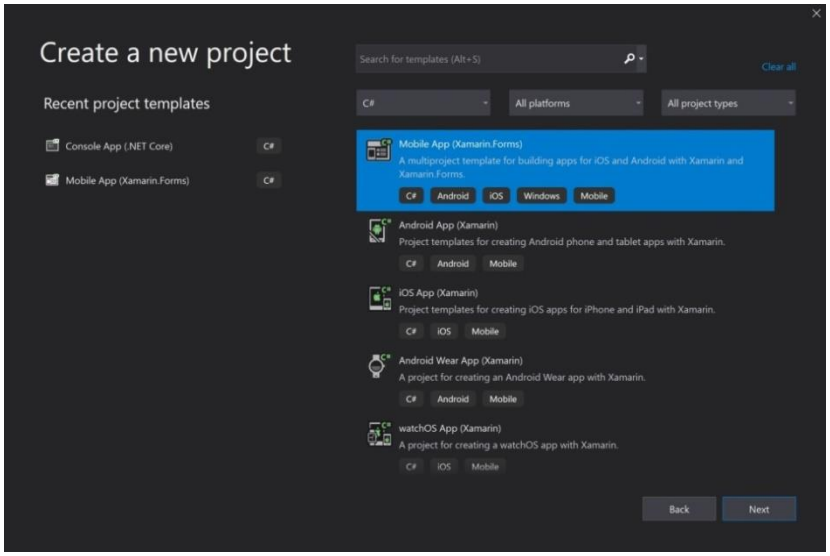
Antud peatükis õpime looma Xamarin.Formsi uut projekti. Eelnevalt on vajalik programmi Visual Studio 2019 olemasolu, millesse on lisatud Xamarin raamistik. Vajalikud juhised Visual studio 2019 koos Xamarin.Forms arendusvahendite allalaadimiseks ning installeerimiseks leiab Microsofti kodulehelt: https://docs.microsoft.com/en-us/xamarin/get-started/installation/windows

1. Käivita Visual Studio 2019 ning vali **Create a new project** (Joonis 1).



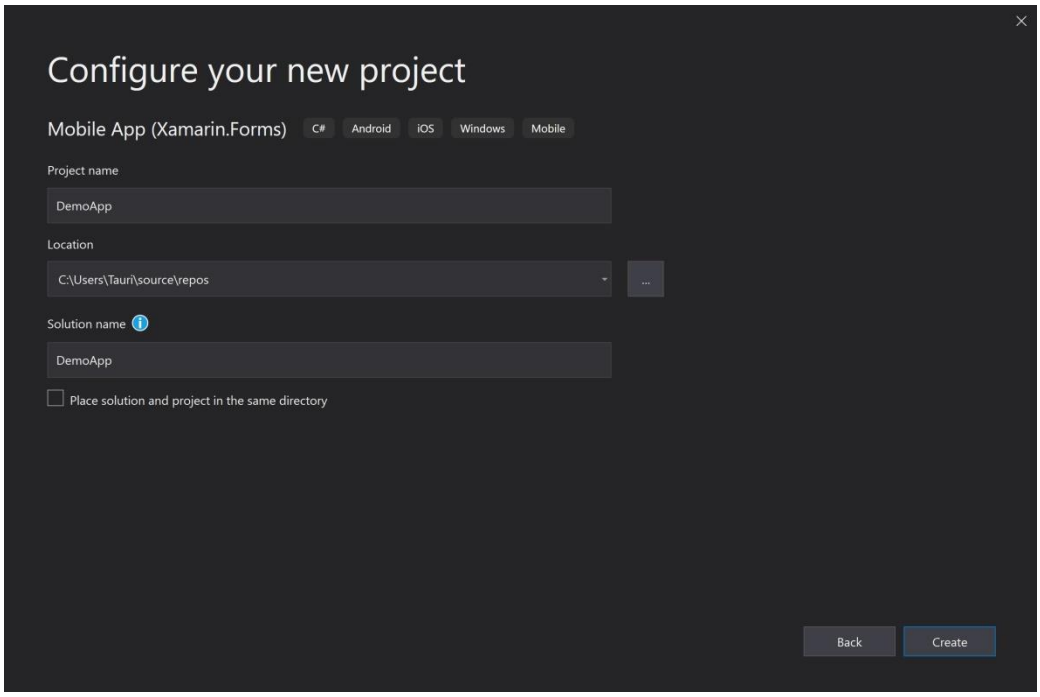
Joonis 1. Visual Studio 2019 avaaken

2. **Create a new project** aknas vali projekti malliks **Mobile App (Xamarin.Forms)**. Lihtsaim viis selle ülesleidmiseks on kasutada **Search for templates** otsinguriba. Pärast malli valimist vajuta **Next** (Joonis 2).



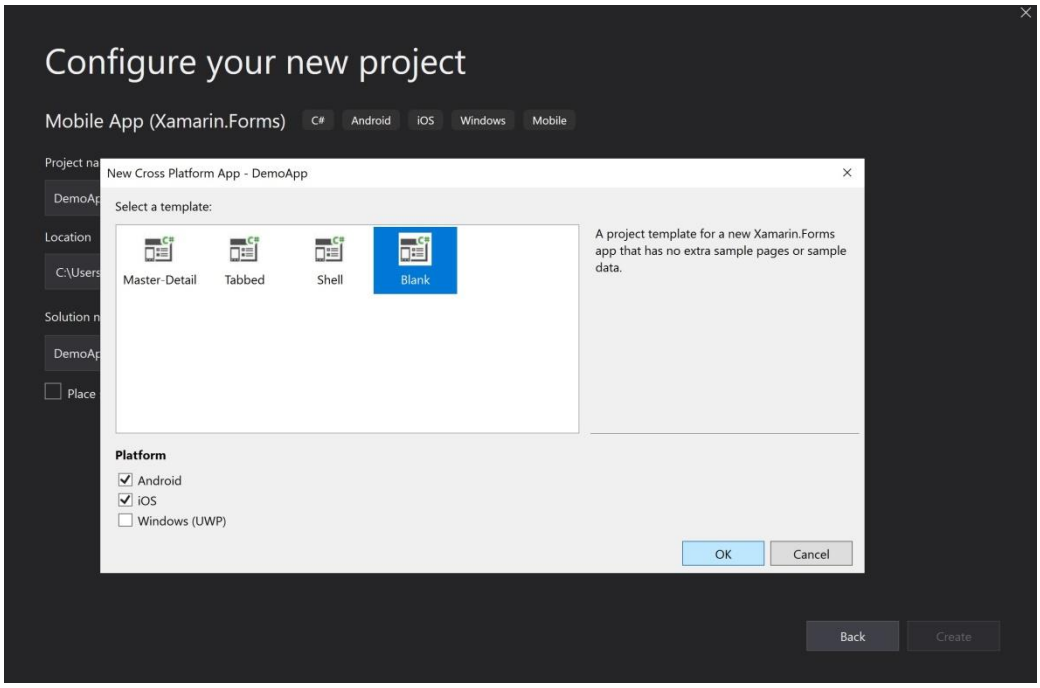
Joonis 2. Create a new project aken

3. **Configure your new project** aknas, sisesta **Project name** lahtrile projekti nimi. Antud rakenduse nimeks saab **DemoApp** (Joonis 3).



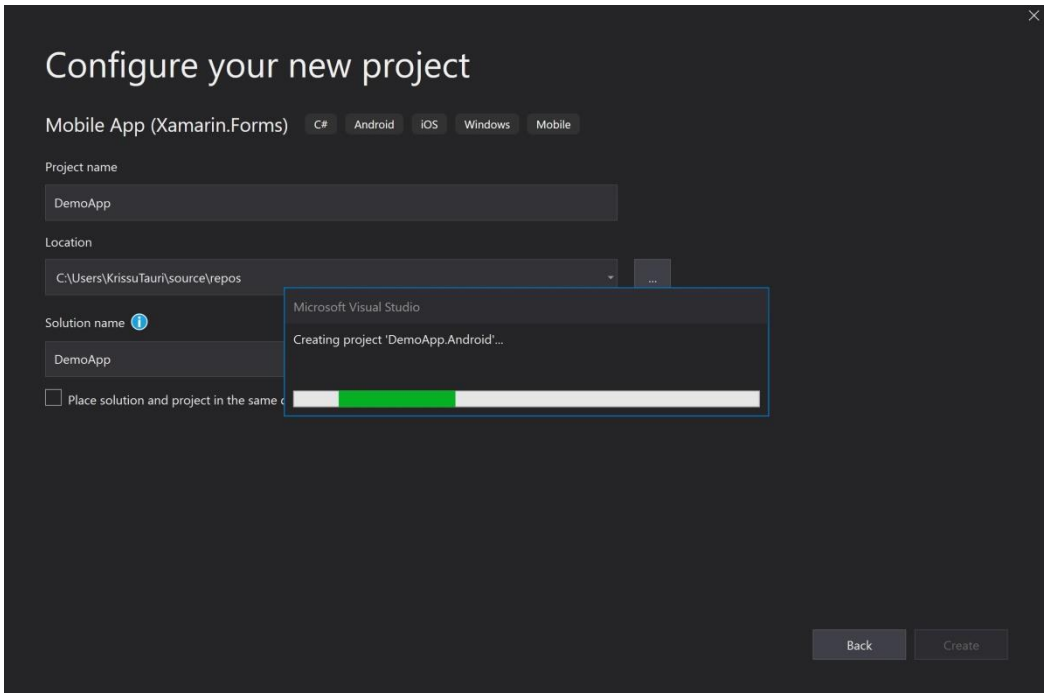
Joonis 3. Configure your new project aken

4. **New Cross Platform App** aknas vali **Blank** mall ning lisa soovitud platvormid. Antud rakenduses kasutame **Android** ja **iOS** platvorme. Kinnita valikud vajutades **OK** nuppu (Joonis 4).



Joonis 4. New Cross Platform App aken

5. **Configure your new project** aknas vajuta **Create** nuppu. Pärast **Create** nupu vajutamist luuakse projekt, mis võib võtta mitu minutit aega (Joonis 5).

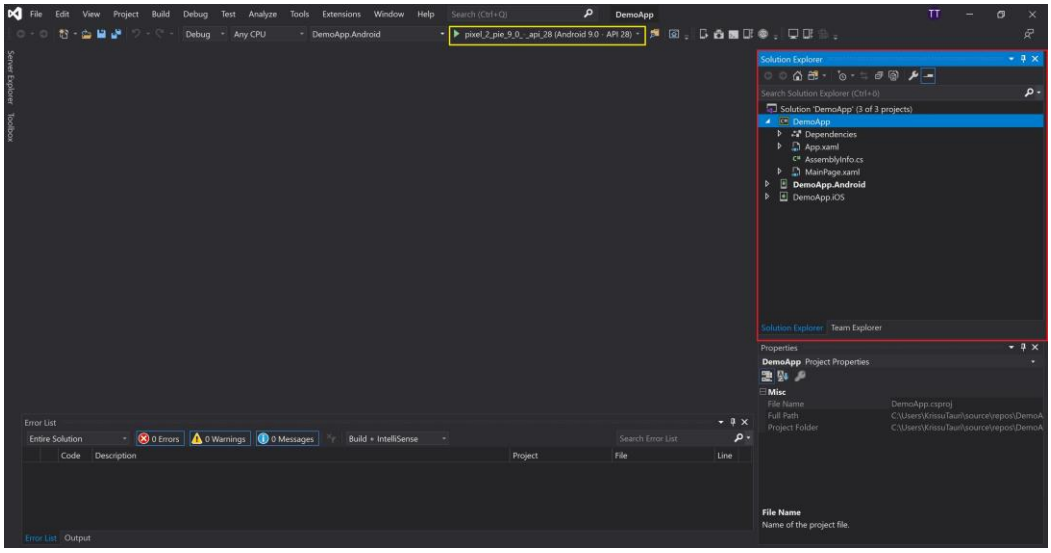


Joonis 5. DemoApp projekti loomine

6. Punase kastiga märgitud alas asub **Solution Explorer**, kust saab ligi projekti failidele. Kollasega märgitud alas asub **emulaator**, kust on võimalik rakendust käivitada ja peatada ning valida soovitud platvorm millel rakendust käivitada (Joonis 6).

   Esialgne emulaatori käivitamine võib võtta kauem aega, soovitatav on arenduse käigus emulaatorit mitte täielikult kinni panna vaid peatada. Pärast käivitamist ilmub emulaatori alas rohelise noole asemel punane kast, kust saab emulaatorit peatada. Peatada on soovituslik pärast igat testimist.

   Selleks, et kasutada iOS emulaatorit Windows arvutil, on vajalik võrku ühendatud Mac arvutit. Vajalikud juhised leiab Xamarini dokumentatsioonist veebiaadressil: https://docs.microsoft.com/en-us/xamarin/ios/get-started/installation/windows/introduction-to-xamarin-ios-for-visual-studio



Joonis 6. Kuvahõive pärast projekti loomist

Antud **DemoApp** lahendus sisaldab endas kolme projekti:

- **DemoApp** – .Net projekt, mis sisaldab kõigi platvormide ühist äriloogika ning kasutajaliidese koodi.
- **DemoApp.Android** – sisaldab Androidi platvormi spetsiifilist koodi ning Androidi rakenduse käivitusinstruktsioone.
- **DemoApp.iOS** – sisaldab iOS platvormi spetsiifilist koodi ning iOS rakenduse käivitusinstruktsioone.

2. Ühe leheküljeline rakendus

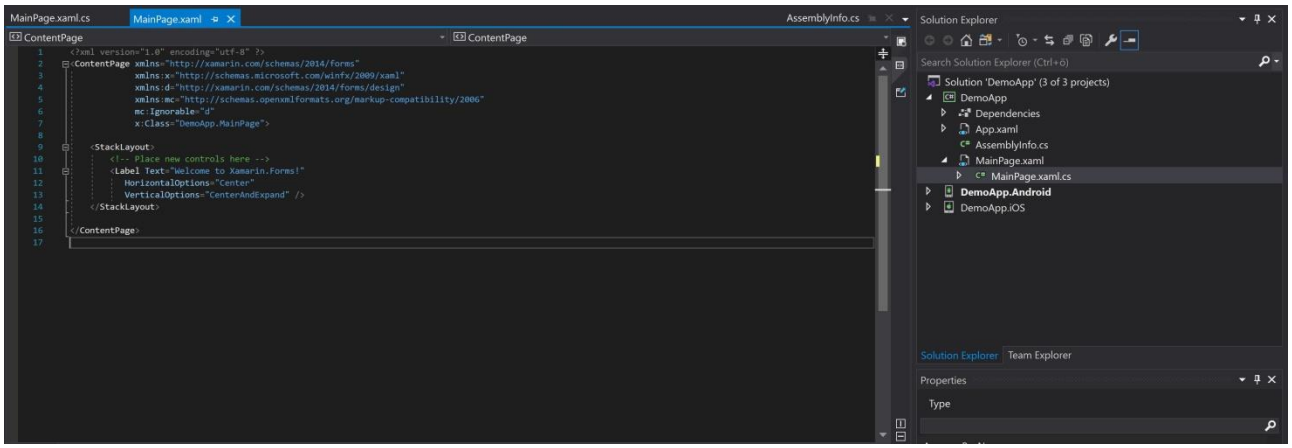
Antud peatükis loome ühe leheküljelise rakenduse, millelt saab lisada postitusi ning neid loendis välja kuvada.

Selles peatükis õpime:

- Kasutajaliidese ning äriloogika failide ülesehitust
- Klassi loomist
- Erinevate vaadete lisamist
- ListView vaate malli loomist
- Andmete kuvamist ListView vaate abil
- Andmete sidumist
- ObservableCollection loendi kasutamist

2.1. Kasutajaliidese ning äriloogika failide ülesehitus

1. Avame **Solution Explorer**-is **MainPage.xaml** faili (Joonis 7).

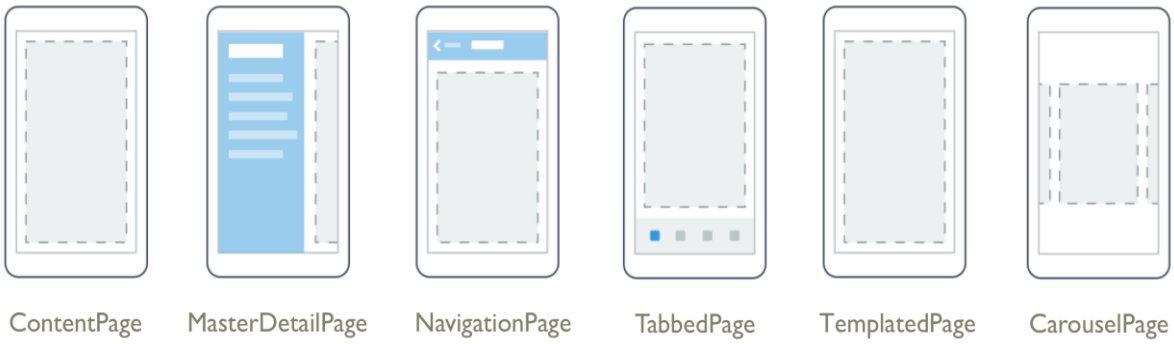


Joonis 7. MainPage.xaml avamine

MainPage.xaml fail sisaldab endas MainPage lehekülje kasutajaliidese koodi. Rakenduse kasutajaliidese loomiseks kasutab Xamarin.Forms nelja peamist juhtrühma.

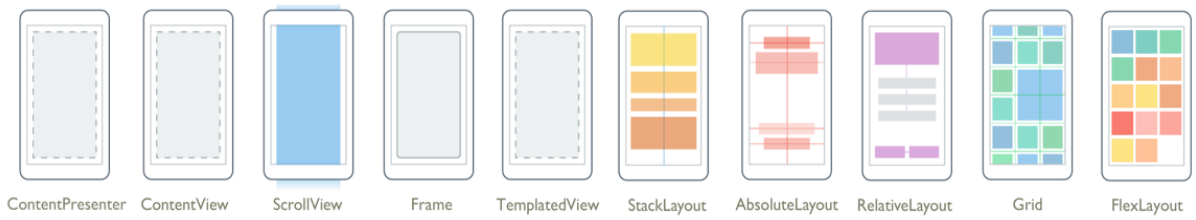
- **Leheküljed (Pages)** – Leheküljed esindavad rakenduse ekraane ning määravad ära lehekülje tüübi. Antud rakendus kasutab ContentPage lehekülje tüüpi, mis on kõige tavalisem lehekülje tüüp, sisaldades endas ainult ühte vaadet.

Erinevate lehekülje tüüpide kohta leiab infot Xamarin.Formsi dokumentatsioonist veebi aadressil:
https://docs.microsoft.com/et-ee/xamarin/xamarin-forms/user-interface/controls/pages



Joonis 8. Xamarin.Forms lehekülje tüübid

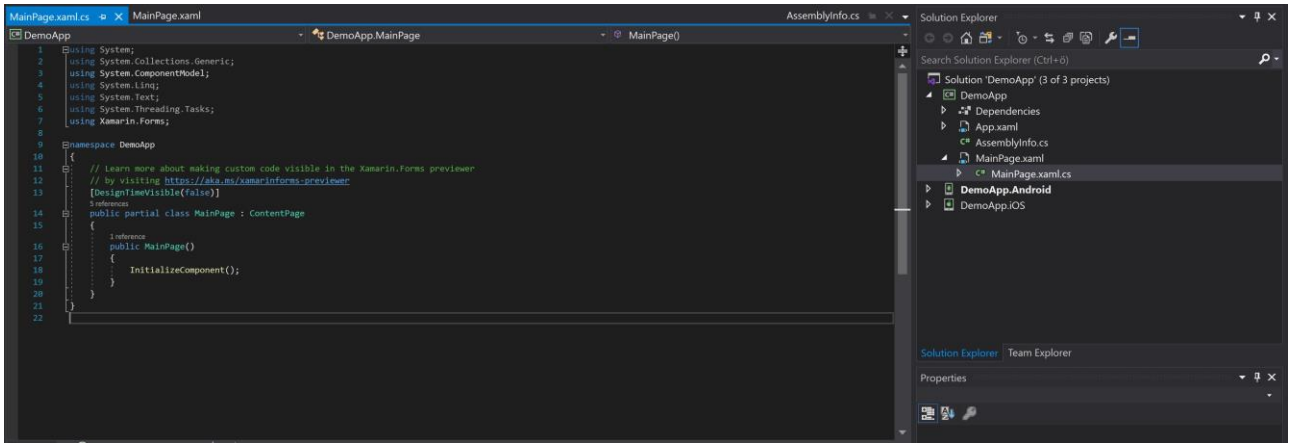
- **Vaated (Views)** – vaated on kasutajaliideses kuvatavad juhtelemendid, nagu näiteks sildid nupud, ning tekstisisestuslahter. Kogu nimekirja Xamarin.Forms kasutatavatest vaadetest leiab Xamarin.Forms dokumentatsioonist veebiaadressil: https://docs.microsoft.com/et-ee/xamarin/xamarin-forms/user-interface/controls/views
- **Paigutus (Layout)** – paigutused on konteinerid, mida kasutatakse vaadete loogiliste struktuuride loomiseks. Antud rakenduses on kasutusel **StackLayout** paigutus, mis paigutab erinevad vaated üksteise alla. Infot erinevate paigutustüüpide kohta leiab Xamarin.Formsi dokumentatsioonist, mis on leitav veebiaadressil: https://docs.microsoft.com/et-ee/xamarin/xamarin-forms/user-interface/controls/layouts



Joonis 9. Xamarin.Formsi paigutustüübid

- **Lahter (Cell)** - lahtrid on spetsiaalsed elemendid, mida kasutatakse loendi kirjete jaoks ning kirjeldavad kuidas loendi iga kirje välja kuvatakse. Lahter ei ole visuaalne element vaid mall visuaalsete elementide kuvamiseks. Lahtreid saab lisada vaadetele **ListView** ja **TableView**. Erinevad lahtri tüüpide nimekirja leiab Xamarin.Forms dokumentatsioonis veebiaadressil: https://docs.microsoft.com/et-ee/xamarin/xamarin-forms/user-interface/controls/cells

2. Avame **Solution Explorer**-is **MainPage.xaml.cs** faili (Joonis 10).

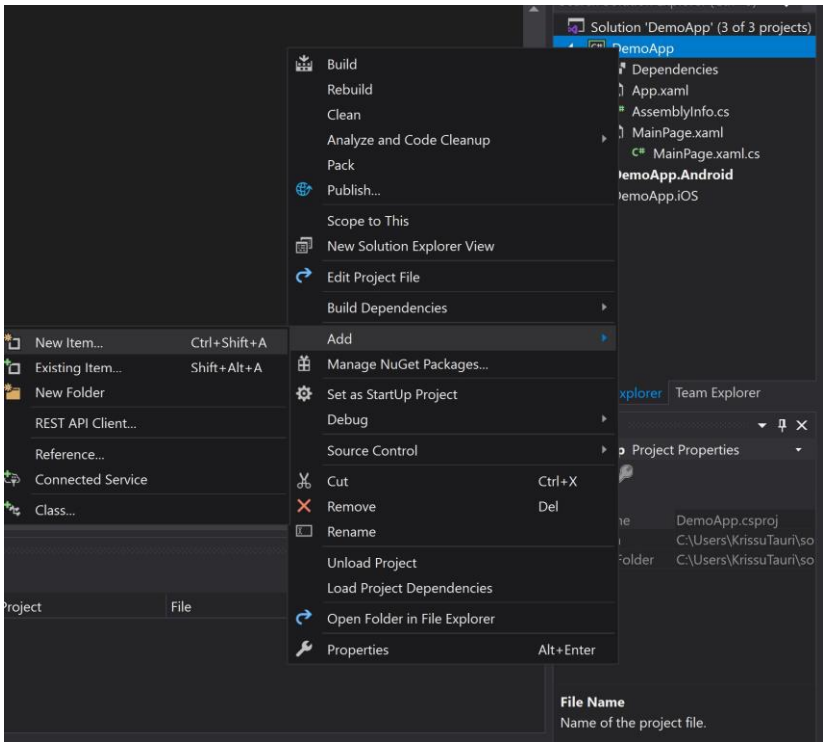


Joonis 10. MainPage.xaml.cs avamine

MainPage.xaml.cs sisaldab endas **MainPage.xaml** äriloogika koodi. Näeme, et hetkel kutsutakse **MainPage()** konstruktoris välja ainult **InitializeComponent()** meetod. InitializeComponent() meetod kutsub välja MainPage.xaml failis olevate elementide loomise.

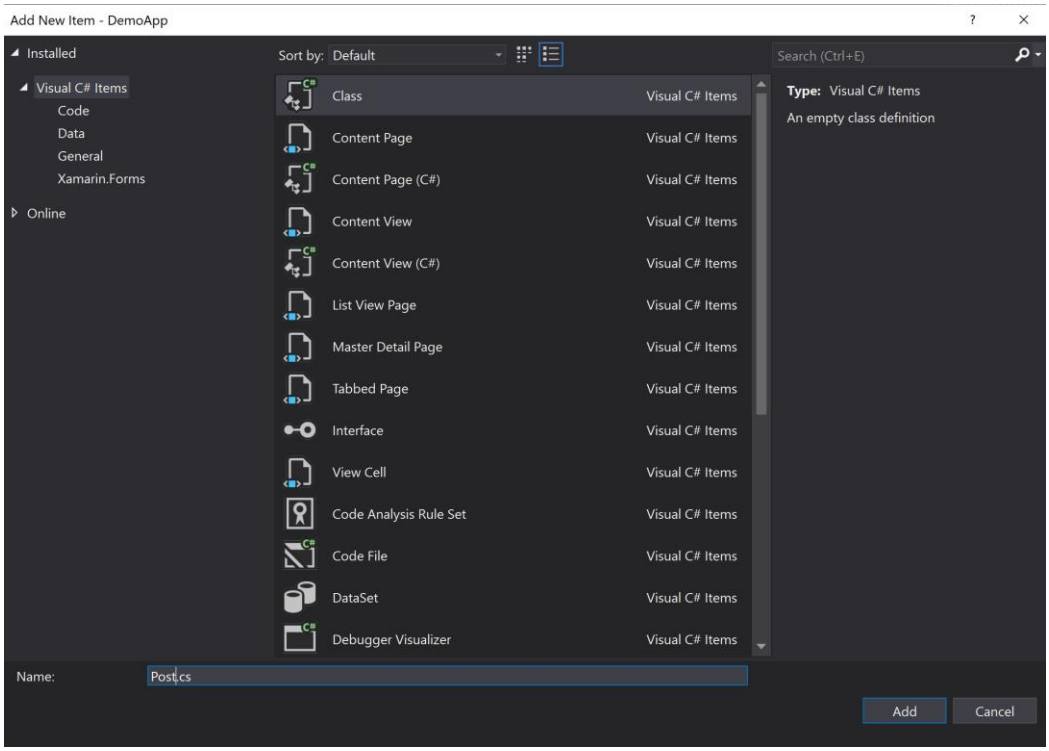
2.2. Klassi loomine

1. Klassi loomiseks vajutame **Solution Explorer**-is hiire parema klikiga **DemoApp** projektil, valime **Add** ning edasi **New Item** (Joonis 11).



Joonis 11. Uue üksuse loomine

2. **Add New Item** aknas valime **Class** (kiiremaks otsinguks võib kasutada **Search** otsinguriba). Sisestame **Name** lahtrisse klassi nime **Post.cs** ning kinnitame valikud vajutades **Add** nuppu (Joonis 12).



Joonis 12. Klassi loomine

3. Avame **Solution Explorer**-is loodud **Post.cs** faili, mis asub **DemoApp** projektis.
4. Muudame klassi avalikuks (**public**). Lisame klassile omadused: **Id**, **Title**, **Date** ning igale omadusele **get, set** meetodid, mille kaudu omadust küsida ning omadust muuta (Koodinäide 1Koodinäide 1. Klassi omaduste lisamine).

```
using System;
using System.Collections.Generic;
using System.Text;

namespace DemoApp
{
    public class Post

    {

        public int Id {
            get;
            set;
        }

        public string Title {
            get;
            set;
        }

        public DateTime Date {
            get;
            set;
        }

    }
```

```
}
```

Koodinäide 1. Klassi omaduste lisamine (Post.cs)

2.3. Andmete kuvamine ListView vaate abil.

1. Avame **MainPage.xaml** faili. Selleks, et kuvada andmed nimekirjana, lisame **StackLayout** elemendi sisse **ListView** vaate. ListView vaatele muutujanime määramiseks kasutame atribuuti **x:Name** ning anname muutjale nimeks **Posts_ListView** mida kasutame hiljem andmete lisamiseks (Koodinäide 2).

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="DemoApp.MainPage">


    <StackLayout>

        <ListView x:Name="Posts_ListView" >

        </ListView>

    </StackLayout>

</ContentPage>
```

Koodinäide 2. ListView elemendi lisamine (MainPage.xaml)

2. **ListView** elemendi sisse lisame **ListView.ItemTemplate** ja **DataTemplate** elemendid, mis määravad ära malli, kuidas igal real kirje kuvatakse (Koodinäide 3).

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="DemoApp.MainPage">


    <StackLayout>

        <ListView x:Name="Posts_ListView" >
            <ListView.ItemTemplate>
                <DataTemplate>

                </DataTemplate>
            </ListView.ItemTemplate>

        </ListView>

    </StackLayout>
```

```
</ContentPage>
```

Koodinäide 3. ListView elemendile kuvatava malli loomine (MainPage.xaml)

3. Lisame **DataTemplate** elemendi sisse **TextCell** lahtri. **TextCell** on lahter teksti kuvamiseks, mis sisaldab endas kahte atribuuti **Text** ja **Detail** ning võimaldab kuvada teksti kuni kahel real. **Text** atribuut võimaldab kuvada teksti esimesel real suures fondis. Lisades **Detail** atribuudi saame võimaluse kuvada teise teksti teisel real väikses fondis.

   Lisame **TextCell** elemendile **Text** ja **Detail** atribuudid (Koodinäide 4).

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="DemoApp.MainPage">


    <StackLayout>

        <ListView x:Name="Posts_ListView" >
            <ListView.ItemTemplate>
                <DataTemplate>
                    <TextCell Text=""
                              Detail="" />
                </DataTemplate>
            </ListView.ItemTemplate>

        </ListView>

    </StackLayout>

</ContentPage>
```

Koodinäide 4. TextCell lahtri lisamine (MainPage.xaml)

4. Avame **Mainpage.xaml.cs** faili. Selleks, et lisada eelnevalt loodud **Posts_ListView** vaatele andmeid, kasutame **ListView** atribuuti **ItemSource**.
5. Omistame **Posts_ListView** vaate atribuudile **ItemSource** uue listi nimega **Post**, millesse lisame näidisandmed (Koodinäide 5).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;



namespace DemoApp
{
    [DesignTimeVisible(false)]
    public partial class MainPage : ContentPage
    {

        public MainPage()
        {
            InitializeComponent();

            Posts_ListView.ItemsSource = new List<Post>()
            {
                new Post() { Title = "post1", Date = DateTime.Now },
                new Post() { Title = "post2", Date = DateTime.Now },
                new Post() { Title = "post3", Date = DateTime.Now },
                new Post() { Title = "post4", Date = DateTime.Now },
                new Post() { Title = "post5", Date = DateTime.Now }
            };

        }

    }

}
```

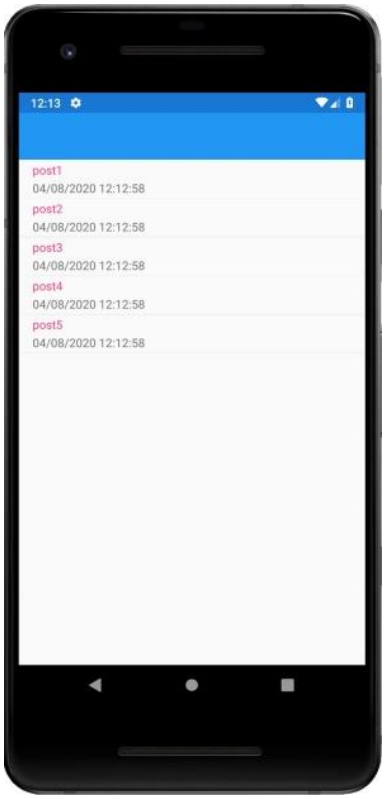
Koodinäide 5. Andmete lisamine ListView vaatele (MainPage.xaml.cs)

6. Avame **MainPage.xaml** faili. Andmete sidumiseks kasutame XAML märgistus laiendit (ingl. markup extension) **Binding**. Märgistus laiend defineeritakse atribuudile loogeliste sulgude vahele. Seome **TextCell** lahtri **Text** atribuudi klassi Post omadusega **Title** ning **Detail** atribuudi omadusega **Date** (Koodinäide 6).

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="DemoApp.MainPage">


    <StackLayout>

        <ListView x:Name="Posts_ListView" >
            <ListView.ItemTemplate>
                <DataTemplate>
                    <TextCell Text="{Binding Title}"
                              Detail="{Binding Date}" />
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>

    </StackLayout>

</ContentPage>
```

Koodinäide 6. Andmete sidumine (MainPage.xaml)

Pärast rakenduse käivitamist on tulemus selline (Joonis 13Joonis 13. Rakenduse kuvahõive):



Joonis 13. Rakenduse kuvahõive

2.4. Andmete lisamine ListView vaatesse

1. Avame MainPage.xaml faili. Lisame **StackLayout** elemendi sisse uue tekstirea kasutades **Label** elementi ning teksti kuvamiseks kasutame Label atribuuti **Text** (Koodinäide 7).

```
<StackLayout>

    <Label Text="Lisa postitus"/>

    <ListView x:Name="Posts_ListView" >
        <ListView.ItemTemplate>
            <DataTemplate>
                <TextCell Text="{Binding Title}"
                          Detail="{Binding Date}" />
            </DataTemplate>
        </ListView.ItemTemplate>

    </ListView>

</StackLayout>
```

Koodinäide 7. Tekstirea lisamine kasutades Label elementi (MainPage.xaml)

2. Tekstisisestuselahtri kasutamiseks lisame elemendi **Entry** ning omistame talle muutujanimeks **postTitle**, mille kaudu saame hiljem sisestatud väärtuse küsida. Kohahoidja teksti lisamiseks kasutame atribuuti **Placeholder** (Koodinäide 8).

```
<StackLayout>

    <Label Text="Lisa postitus"/>
    <Entry x:Name="postTitle" Placeholder="Postitus"></Entry>

    <ListView x:Name="Posts_ListView" >
        <ListView.ItemTemplate>
            <DataTemplate>
                <TextCell Text="{Binding Title}"
                          Detail="{Binding Date}" />
            </DataTemplate>
        </ListView.ItemTemplate>

    </ListView>

</StackLayout>
```

Koodinäide 8. Tekstisisestuslahtri lisamine (MainPage.xaml)

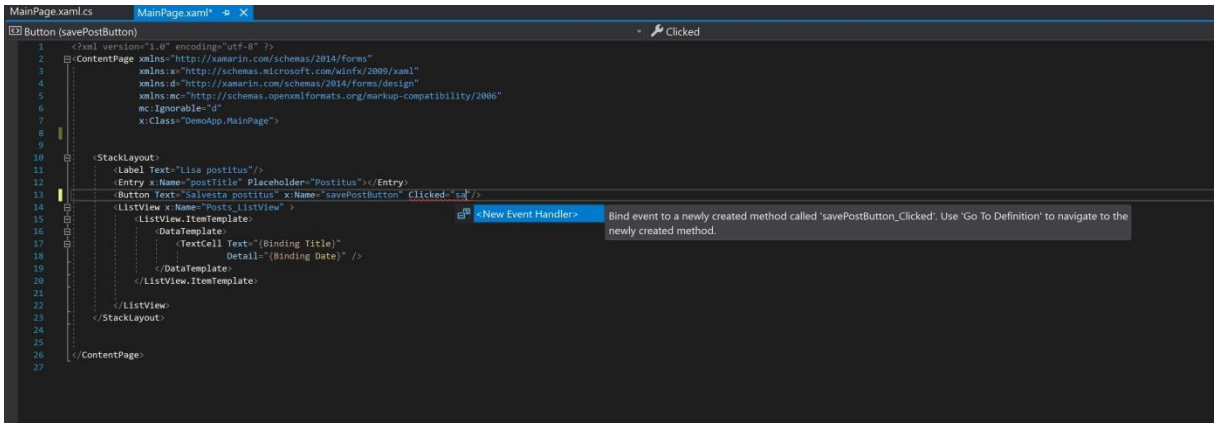
3. Nupu lisamiseks kasutame elementi **Button**. Lisame nupul kuvatava teksti kasutades **Text** atribuuti ning kasutades atribuuti **x:Name** omistame muutujanime **savePostButton**.

Lisame nupule sündmuse **Clicked**, mis käivitaks **savePostButton_Clicked** meetodi (Koodinäide 9).

Meetodit on võimalik luua Visual Studios automaatselt .xaml failis vajutades tabulaatori nuppu meetodi nime kirjutades. Pärast tabulaatori vajutamist luuakse meetod .xaml.cs faili (Joonis 14. Meetodi genereerimine automaatselt).

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="DemoApp.MainPage">


    <StackLayout>

        <Label Text="Lisa postitus"/>
        <Entry x:Name="postTitle" Placeholder="Postitus"></Entry>
        <Button Text="Salvesta postitus"
                x:Name="savePostButton"
                Clicked="savePostButton_Clicked"/>

        <ListView x:Name="Posts_ListView" >
            <ListView.ItemTemplate>
                <DataTemplate>
                    <TextCell Text="{Binding Title}"
                              Detail="{Binding Date}" />
                </DataTemplate>
            </ListView.ItemTemplate>

        </ListView>

    </StackLayout>

</ContentPage>
```

Koodinäide 9. Nupu lisamine ning meetodi omistamine (MainPage.xaml)



Joonis 14. Meetodi genereerimine automaatselt

2.5. ObservableCollection loendi kasutamine

Andmete lisamiseks võtame kasutusele **ObservableCollection** loendi, mis käitub samamoodi nagu tavaline loend, kuid ta suudab **ListView** vaate **ItemSource** atribuuti teavitada loendis toimunud muudatustest ning uuendada **Listview** vaadet automaatselt. **ObservableCollection** kasutab **System.Collection.ObjectModel** teeki, mistõttu on see vajalik lisada.

1. Avame MainPage.xaml.cs. Lisame **System.Collections.ObjectModel** teegi ning loome uue **ObservableCollection**i nimega **Posts**, mis lubab kasutada Post klassi tüüpi andmeid. Määrame uueks **Posts_ListView** vaate **ItemSource** atribuutile uue loodud ObservableCollection **Posts** ning lisame mõned näidisandmed (Koodinäide 10).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

using System.Collections.ObjectModel; //teek ObservableCollection loendi kasutamiseks

namespace DemoApp
{

    [DesignTimeVisible(false)]
    public partial class MainPage : ContentPage
    {
        ObservableCollection<Post> Posts { get; set; } = new ObservableCollection<Post>();


        public MainPage()
        {
            InitializeComponent();

            Posts_ListView.ItemsSource = new List<Post>()
            {
                new Post() { Title = "post1", Date = DateTime.Now },
                new Post() { Title = "post2", Date = DateTime.Now },
                new Post() { Title = "post3", Date = DateTime.Now },
                new Post() { Title = "post4", Date = DateTime.Now },
                new Post() { Title = "post5", Date = DateTime.Now }
            };

            Posts_ListView.ItemsSource = Posts;

            Posts.Add(new Post() { Title = "post1", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post2", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post3", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post4", Date = DateTime.Now });
```

```
        }

        private void savePostButton_Clicked(object sender, EventArgs e)
        {

        }
    }

}
```

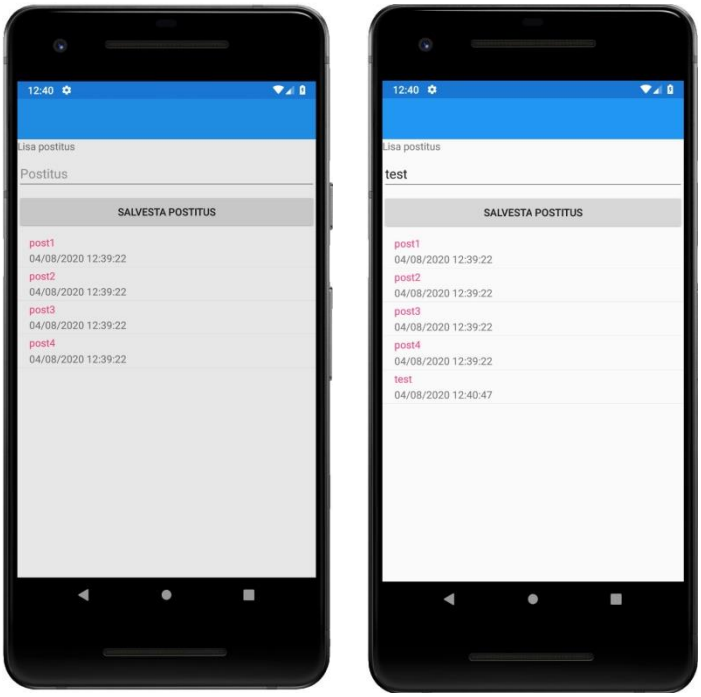
Koodinäide 10. ObservableCollection loendi lisamine (MainPage.xaml.cs)

2. Lisame eelnevalt loodud **savePostButton_Clicked** meetodile andmete lisamise ObservableCollection **Posts** listi. MainPage.xaml failis sisestatud **postTitle** lahtri väärtuse saame küsida kasutades **Text** atribuuti (Koodinäide 11).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

using System.Collections.ObjectModel; //lubab kasutada ObservableCollection listi

namespace DemoApp
{

    [DesignTimeVisible(false)]
    public partial class MainPage : ContentPage
    {
        ObservableCollection<Post> Posts { get; set; } = new ObservableCollection<Post>();


        public MainPage()
        {
            InitializeComponent();


            Posts_ListView.ItemsSource = Posts;

            Posts.Add(new Post() { Title = "post1", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post2", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post3", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post4", Date = DateTime.Now });

        }

        private void savePostButton_Clicked(object sender, EventArgs e)
        {
            Posts.Add(new Post() { Title = postTitle.Text, Date = DateTime.Now });
        }
    }

}
```

Koodinäide 11. Andmete lisamine ObservableCollection loendisse (MainPage.xaml.cs)

Pärast rakenduse käivitamist näeme, et postituste lisamisel uuendatakse postituste loendit automaatselt (Joonis 15).



Joonis 15. Üheleheküljelise rakenduse kuvahõive

3. Mitme leheküljeline rakendus

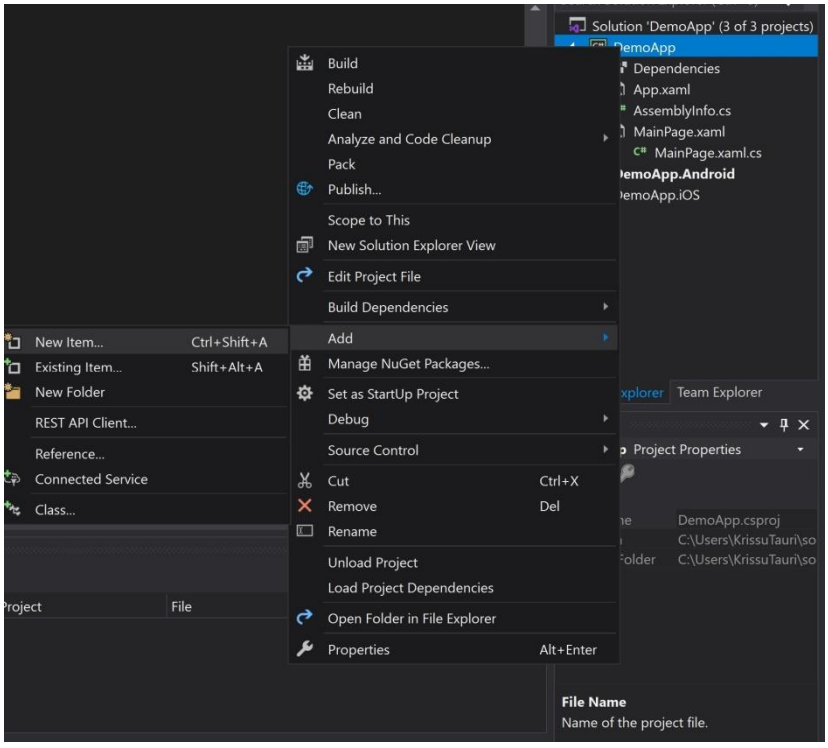
Antud peatükis muudame eelnevalt loodud rakenduse mitmeleheliseks rakenduseks. Eesmärgiks on luua uus leht, kust toimuks postituste lisamine pealehele.

Selles peatükis õpime:

- Uue lehekülje loomist
- Seadme navigeerimiselementide lisamist
- Lehekülgede vahel navigeerimist
- Andmete lisamist teistelt lehekülgedelt
- OnAppearing() meetodi kasutamist

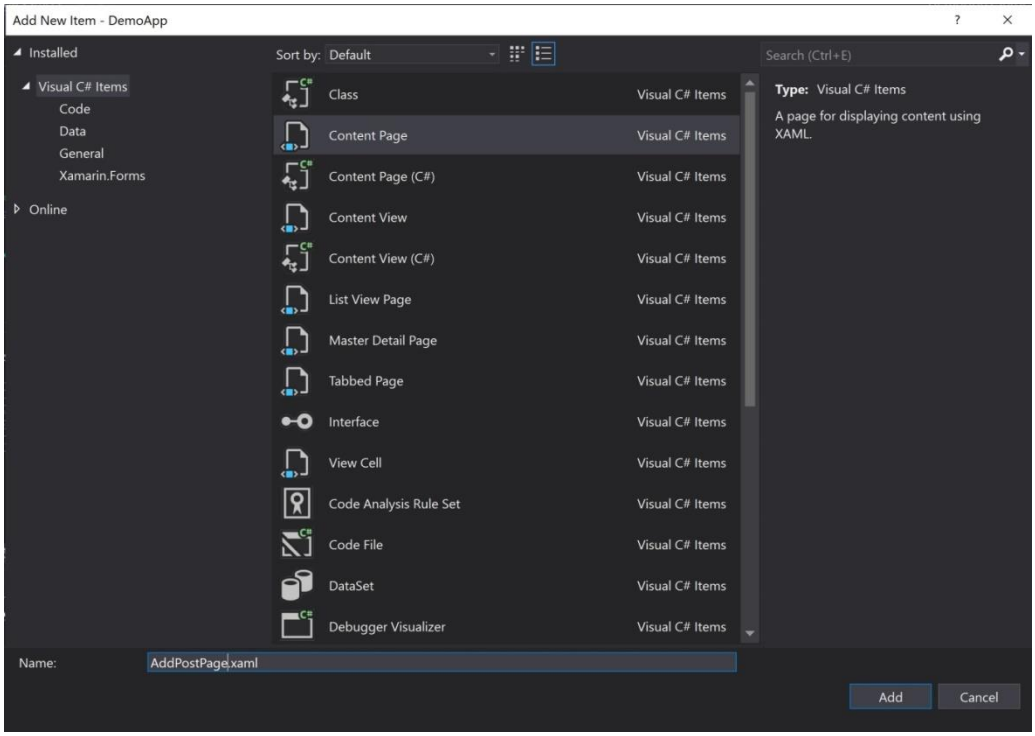
3.1. Uue lehekülje loomine

1. Selleks, et lisada uut lehekülje vaadet, toimime samamoodi nagu klassi lisamisel. Vajutame **Solution Explorer**-is hiire parema klikiga **DemoApp** projektil, valime **Add** ning edasi **New Item.**



Joonis 16. Uue üksuse lisamine

2. **Add New Item** aknas valime **Content Page** ning **Name** lahtris anname lehele nimeks **AddPostPage.xaml**. Kinnitame valikud **Add** nupu vajutusega, mille tulemusena lisatakse **DempApp** projektile **AddPostPage.xaml** kui ka **AddPostPage.xaml.cs** fail.



Joonis 17. Content Page lehekülje tüübi lisamine

3.2. Seadme navigatsiooni elementide lisamine

Selleks, et kasutada platvormile omaseid seadme navigatiooni elemente nagu näiteks seadme tagasiliikumise nupp või navigatsiooniribal olev tagasiliikumise nupp, lisame rakendusele navigatsiooni lehekülje.

1. Navigatsiooni lehekülje lisamiseks avame **DemoApp** projektis **App.xaml.cs** faili. Omistame **MainPage** muutujale uue **NavigationPage** tüüpi lehekülje, millele anname kaasa lehekülje, kust soovime navigatsiooni alustada (Koodinäide 12).

```
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace DemoApp
{
    public partial class App : Application
    {



        public App()
        {
            InitializeComponent();

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

        protected override void OnStart()
        {
        }

        protected override void OnSleep()
        {
        }

        protected override void OnResume()
        {
        }
    }
}
```

Koodinäide 12. Navigatsiooni lehekülje lisamine(App.xaml.cs)

2. Avame **MainPage.xaml** faili. Selleks, et tuua esile navigeerimisriba lisame uue elemendi **ContentPage.ToolbarItems**, mille sisse loome elemendi **ToolbarItem**. **ToolbarItem** on spetsiaalne nupu tüüp, mis on kasutatav ainult navigatsiooniribal. Lisame nupul kuvatava nime kasutades **Text** atribuuti ning **Clicked** sündmuse, mis käivitaks **addPostToolbarItem_Clicked** meetodi. Loome automaatselt ka vastava meetodi Mainpage.xaml.cs faili (Koodinäide 13).

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="DemoApp.MainPage">
    <ContentPage.ToolbarItems>
        <ToolbarItem Text="Lisa postitus"
                     Clicked="addPostToolbarItem_Clicked">
        </ToolbarItem>
    </ContentPage.ToolbarItems>

    <StackLayout>
        <Label Text="Lisa postitus"/>
        <Entry x:Name="postTitle" Placeholder="Postitus"></Entry>
        <Button Text="Salvesta postitus"
                x:Name="savePostButton"
                Clicked="savePostButton_Clicked"/>

        <ListView x:Name="Posts_ListView" >
            <ListView.ItemTemplate>
                <DataTemplate>
                    <TextCell Text="{Binding Title}"
                              Detail="{Binding Date}" />
                </DataTemplate>
            </ListView.ItemTemplate>

        </ListView>
    </StackLayout>


</ContentPage>
```

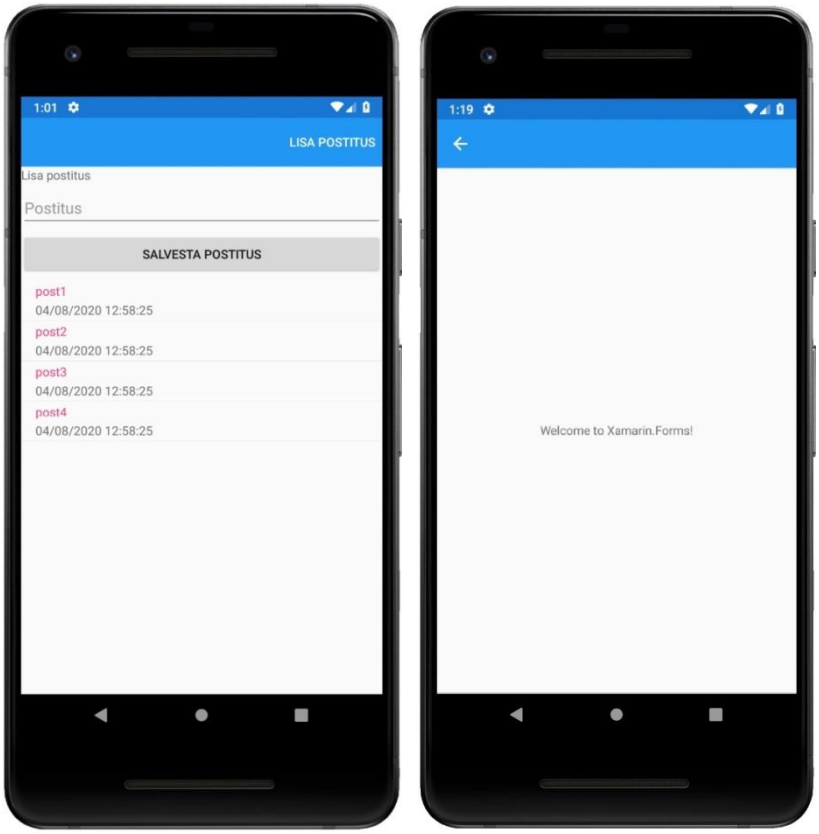
Koodinäide 13. Navigeerimisribale nupu lisamine(MainPage.xaml)

3. Avame **MainPage.xaml.cs** faili ning lisame **addPostToolBarItem_Clicked** meetodisse Xamarin.Formsi meetodi nimega **NavigationPage.PushAsync**, millele anname kaasa lehekülje, kuhu soovime navigeerida (Koodinäide 14).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

using System.Collections.ObjectModel;

namespace DemoApp
{

    [DesignTimeVisible(false)]
    public partial class MainPage : ContentPage
    {
        ObservableCollection<Post> Posts { get; set; } = new ObservableCollection<Post>();

        public MainPage()
        {
            InitializeComponent();

            Posts_ListView.ItemsSource = Posts;

            Posts.Add(new Post() { Title = "post1", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post2", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post3", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post4", Date = DateTime.Now });

        }

        private void savePostButton_Clicked(object sender, EventArgs e)
        {
            Posts.Add(new Post() { Title = postTitle.Text, Date = DateTime.Now });
        }

        private void addPostToolbarItem_Clicked(object sender, EventArgs e)
        {
            Navigation.PushAsync(new AddPostPage());
        }
    }

}
```

Koodinäide 14. Lehekülgede vaheline liikumine(MainPage.xaml.cs)

Pärast rakenduse käivitamist saame rakenduses liikuda AddPostPage leheküljele, ning navigeerida tagasi MainPage leheküljele (Joonis 18).



Joonis 18. Rakenduse kuvahõive pärast lehekülje ning navigatsiooni lisamist

3.3. Andmete lisamine teiselt leheküljelt

1. Viime üle postituse lisamise MainPage leheküljelt AddPostPage leheküljele (Koodinäide 15 ja Koodinäide 16).

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="DemoApp.MainPage">
    <ContentPage.ToolbarItems>
        <ToolbarItem Text="Lisa postitus"
                     Clicked="addPostToolbarItem_Clicked">
        </ToolbarItem>
    </ContentPage.ToolbarItems>

    <StackLayout>
        <Label Text="Lisa postitus"/>
        <Entry x:Name="postTitle" Placeholder="Postitus"></Entry>
        <Button Text="Salvesta postitus"
                x:Name="savePostButton"
                Clicked="savePostButton_Clicked"/>

        <ListView x:Name="Posts_ListView" >
            <ListView.ItemTemplate>
                <DataTemplate>
                    <TextCell Text="{Binding Title}"
                              Detail="{Binding Date}" />
                </DataTemplate>
            </ListView.ItemTemplate>

        </ListView>
    </StackLayout>


</ContentPage>
```

Koodinäide 15. Postituse lisamise elemendite üle viimine (MainPage.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"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="DemoApp.AddPostPage">
    <ContentPage.Content>
        <StackLayout>
            <Label Text="Lisa postitus"/>
            <Entry x:Name="postTitle" Placeholder="Postitus"></Entry>
            <Button Text="Salvesta postitus"
                    x:Name="savePostButton"
                    Clicked="savePostButton_Clicked"/>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

Koodinäide 16.Postituse lisamise elemendite üle viimine (AddPostPage.xaml)

2. Viime üle mainpage.xaml.cs SavePostButton_Clicked meetodi üle AddPostPage.xaml.cs lehele (Koodinäide 17 ja Koodinäide 18).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

using System.Collections.ObjectModel;

namespace DemoApp
{

    [DesignTimeVisible(false)]
    public partial class MainPage : ContentPage
    {
        ObservableCollection<Post> Posts { get; set; } = new ObservableCollection<Post>();

        public MainPage()
        {
            InitializeComponent();

            Posts_ListView.ItemsSource = Posts;

            Posts.Add(new Post() { Title = "post1", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post2", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post3", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post4", Date = DateTime.Now });

        }

        private void savePostButton_Clicked(object sender, EventArgs e)
        {
            Posts.Add(new Post() { Title = postTitle.Text, Date = DateTime.Now });
        }

        private void addPostToolbarItem_Clicked(object sender, EventArgs e)
        {
            Navigation.PushAsync(new AddPostPage());
        }
    }

}
```

Koodinäide 17. savePostButton_Clicked meetodi üleviimine (MainPage.xaml.cs)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

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



namespace DemoApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class AddPostPage : ContentPage
    {


        public AddPostPage()
        {
            InitializeComponent();
        }

        private void savePostButton_Clicked(object sender, EventArgs e)
        {

            Posts.Add(new Post() { Title = postTitle.Text, Date = DateTime.Now });
        }

    }
}
```

Koodinäide 18. savePostButton_Clicked meetodi üleviimine (AddPostPage.xaml.cs)

Praeguses etapis projekti käivitades saame veateate, kuna **AddPostPage** leheküljelt ei pääse ligi **MainPage** leheküljel deklareeritud loendile nimega Posts. Selleks, et pääseda ligi loendile kõigilt lehekülgedelt, loome uue globaalse loendi **App** klassi.

3. Avame **Solution Exploreris** faili **App.Xaml.cs** ning lisame uue loendi nimega **Posts**. Kasutame tavalist loendit ObservableCollection loendi asemele, et näidata hiljem kuidas loendit esile kutsuda kasutades OnAppearing() meetodit. Vajalik on lisada ka System.Collections.Generic teek (Koodinäide 19).

```
using System;
using System.Collections.Generic;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace DemoApp
{
    public partial class App : Application
    {

       public static List<Post> Posts = new List<Post>();

        public App()
        {
            InitializeComponent();
```

```
            //MainPage = new MainPage();
            MainPage = new NavigationPage(new MainPage());
        }

        protected override void OnStart()
        {
        }

        protected override void OnSleep()
        {
        }

        protected override void OnResume()
        {
        }
    }
}
```

Koodinäide 19. Globaalse loendi lisamine App klassi (App.xaml.cs)

4. Avame **MainPage.xaml.cs** faili. Kustutame **ObservableCollection Posts** loendi, kuna seda ei kasutata enam. Kustutame ka **ObservableCollection Posts** loendisse lisatud näidisandmed. Omistame **Posts_ListView** vaate **ItemSource** atribuudile uue eelnevalt loodud loendi **Posts**, mis asup **App** klassis (Koodinäide 20).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

using System.Collections.ObjectModel;

namespace DemoApp
{
    [DesignTimeVisible(false)]
    public partial class MainPage : ContentPage
    {
        ObservableCollection<Post> Posts { get; set; } = new ObservableCollection<Post>();

        public MainPage()
        {
            InitializeComponent();

            Posts_ListView.ItemsSource = App.Posts;

            Posts.Add(new Post() { Title = "post1", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post2", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post3", Date = DateTime.Now });
            Posts.Add(new Post() { Title = "post4", Date = DateTime.Now });



        }


        private void addPostToolbarItem_Clicked(object sender, EventArgs e)
        {
            Navigation.PushAsync(new AddPostPage());
```

```
        }
    }

}
```

Koodinäide 20. App klassis oleva loendi omistamine Posts_ListView vaatele (MainPage.xaml.cs)

5. Avame faili **AddPostPage.xaml.cs**. Lisame **SavePostButton_Clicked** meetodisse andmete salvestamise **App** klassis olevasse **Posts** loendisse.
   Lisame **SavePostButton_Clicked** meetodisse navigeerimise **MainPage** leheküljele, et pärast postituste lisamist navigeerida **MainPage** leheküljele (Koodinäide 21).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

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



namespace DemoApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class AddPostPage : ContentPage
    {


        public AddPostPage()
        {
            InitializeComponent();

        }

        private void savePostButton_Clicked(object sender, EventArgs e)
        {

            App.Posts.Add(new Post() { Title = postTitle.Text, Date = DateTime.Now });
            Navigation.PushAsync(new MainPage());
        }

    }
}
```

Koodinäide 21. Andmete lisamine posts loendisse ja navigeerimine MainPage leheküljele (AddPostPage.xaml.cs)

Praegusel hetkel pärast rakenduse käivitamist näeme, et uusi postitusi ei lisandu, probleem on selles, et loendit ei kutsuta esile kui lehekülg uuesti laetakse.

3.4. OnAppearing() meetodi kasutamine

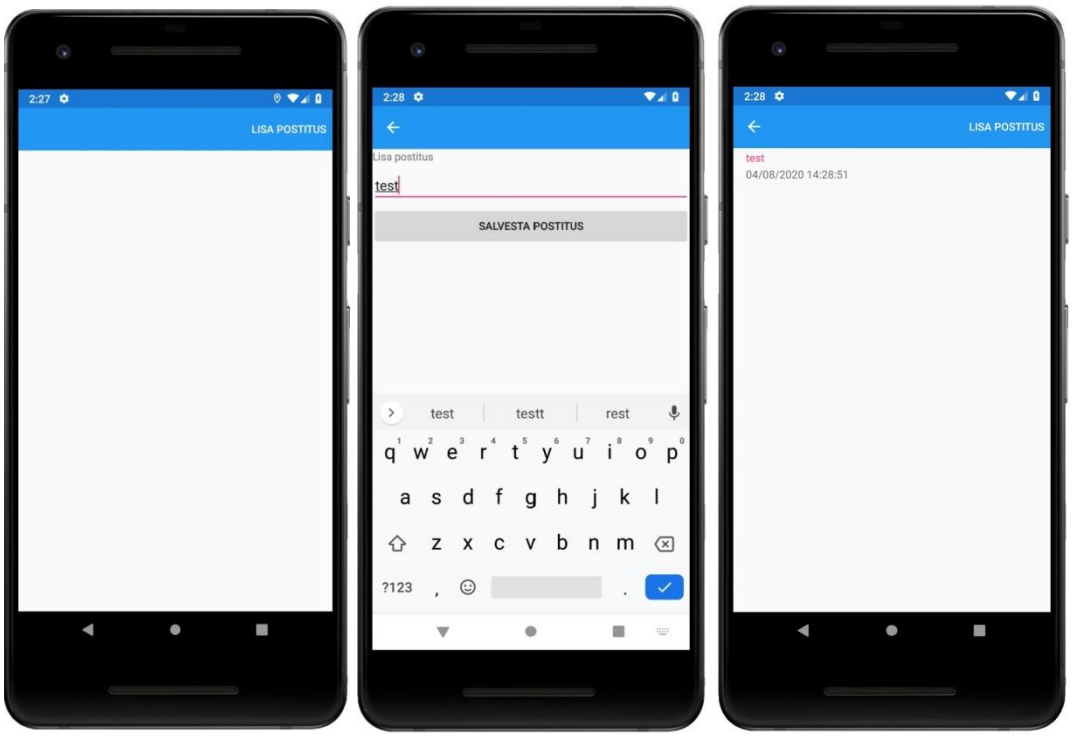
Võtame kasutusele Xamarin.Forms meetodi **OnAppearing()**. OnAppering() on virtuaalne meetod, mis kutsutakse esile igakord enne lehekülje laadimist ning võimaldab sooritada toiminguid enne lehekülje laadimist. Selleks, et ülekirjutada virtuaalseid meetodeid, tuleb kasutada override meetodi tüüpi. Onappearing() meetod kasutab **system.collection.objectmodel** teeki, mis sai eelnevalt lisatud kui kasutasime ObservableCollection loendit.

1. Avame MainPage.xaml.cs faili ning lisame uue meetodi **OnAppearing()**. OnAppearing() meetodis kutsume esile andmete omistamine **Posts_ListView** vaatele, mis uuendab andmeid automaatselt iga kord kui **MainPage** lehekülg kutsutakse esile (Koodinäide 22).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

using System.Collections.ObjectModel;

namespace DemoApp
{

    [DesignTimeVisible(false)]
    public partial class MainPage : ContentPage
    {


        public MainPage()
        {
            InitializeComponent();

        }

        protected override void OnAppearing()
        {

            Posts_ListView.ItemsSource = App.Posts;

        }


        private void addPostToolbarItem_Clicked(object sender, EventArgs e)
        {
            Navigation.PushAsync(new AddPostPage());
        }
    }

}
```

Koodinäide 22. Posts_ListView vaate andmete uuendamine kasutades OnAppearing() meetodit (MainPage.xaml.cs)

Pärast rakenduse käivitamist, näeme, et Post_ListView vaade kutsutakse uuesti esile, ennem lehekülje laadimist ning uued lisatud andmed on nähtaval (Joonis 19).



Joonis 19. Mitme leheküljelise rakenduse kuvahõive

4. SQLite andmebaas

Antud peatükis lisame mitmelehelisele rakendusele andmetele lisamise ning kuvamise andmebaasist.

Selles peatükis õpime:

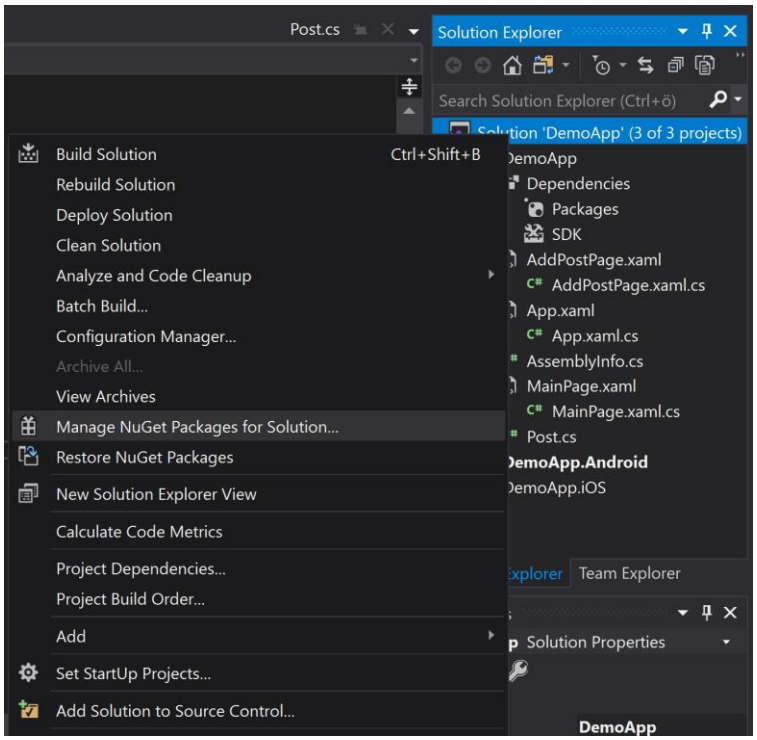
- NuGet pakketide lisamist
- SQLite paketi seadistamist Android platvormile
- SQLite paketi seadistamist iOS platvormile
- SQLite paketi klassi siseste funktsionaalsuste kasutamist
- Andmete sisestamist andmebaasi kasutades SQLite paketti
- Andmete kuvamist andmebaasist kasutades SQLite paketti

4.1. NuGetite lisamine

NuGet on paketihaldus tööriist, mille kaudu saavad arendajad luua, jagada ja kasutada kasulikku koodi teiste arendajatega. Jagatav kood kompileeritakse NuGet pakettideks, mida teised arendajad saavad lisada enda projekti. Pärast NuGet paketi lisamist on võimalik kasutada paketi funktsionaalsust enda projektis.

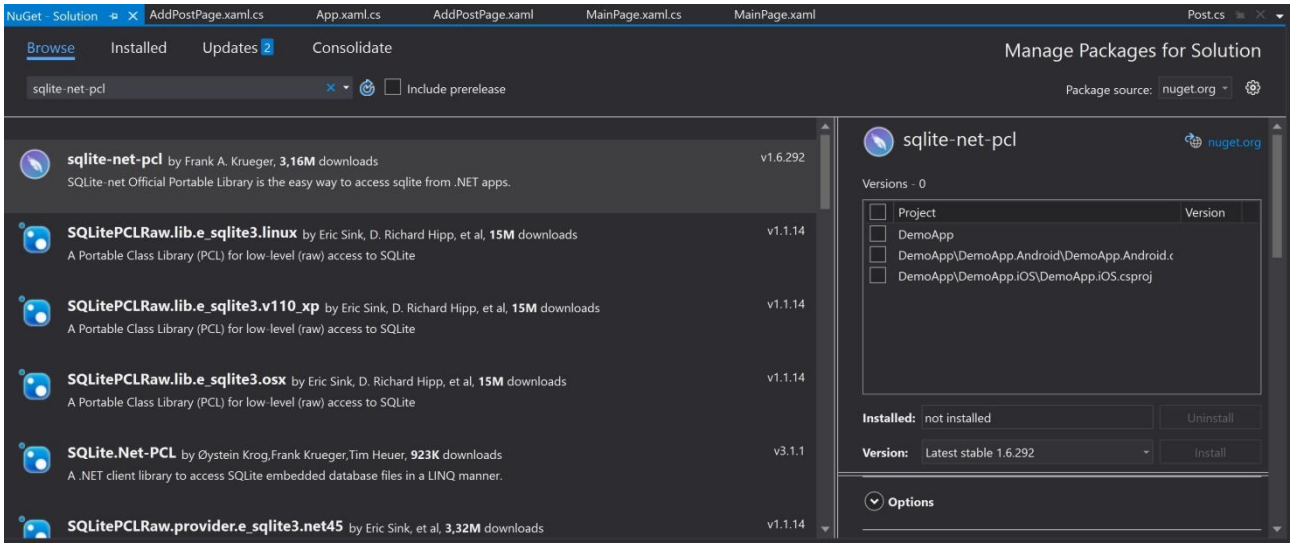
Mac arvutitel tuleb lisada NuGet paketid kõikidele projektidele (DemoApp, DemoApp.Android, DemoApp.iOS) eraldi. Vajalikud juhised leiab Xamarin dokumentatsioonist veebiaadressil: https://docs.microsoft.com/en-us/visualstudio/mac/nuget-walkthrough?view=vsmac-2019

1. NuGet paketihalduri avamiseks vajutame **DemoApp** lahendusel parema hiire klikiga ning valime **Manage NuGet Packages For Solution** (Joonis 20).



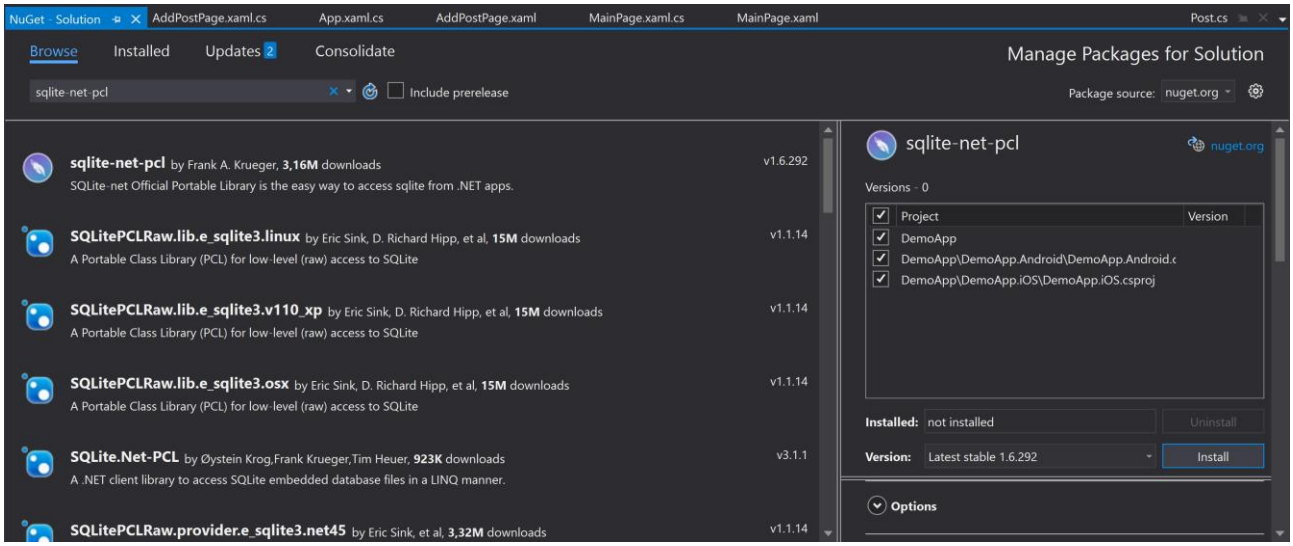
Joonis 20. Nuget pakteihalduri avamine

2. **NuGet Solution** aknas valime **Browse** ning otsime **Search** otsinguribalt **sqlite-net-pcl** paketti. Veendu, et pakett on loodud **Frank A. Krueger** poolt (Joonis 21).



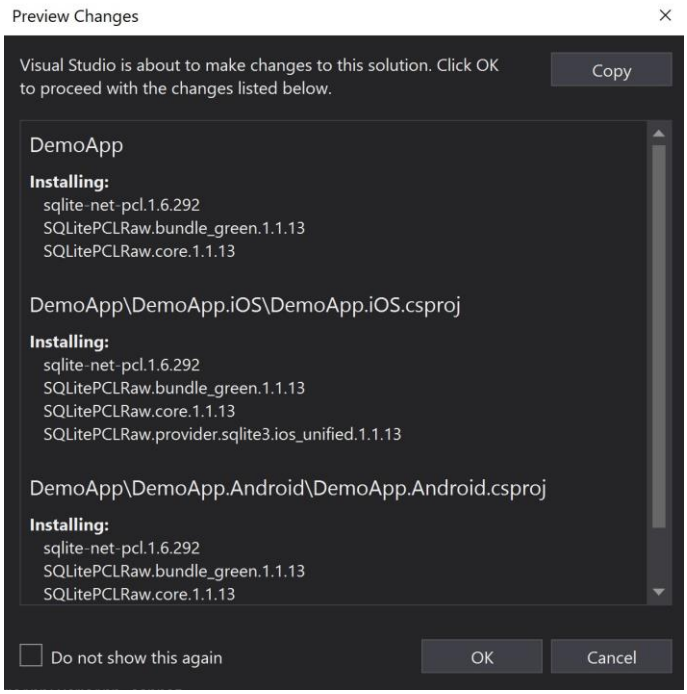
Joonis 21. sqlite-net-pcl paketi otsing

3. Lisame **sqlite-net-pcl** paketi kõikidesse projektidesse, valime kõige viimasema versiooni **1.6.292** ning vajutame **Install** (Joonis 22).



Joonis 22. NuGet paketi lisamine projektidesse

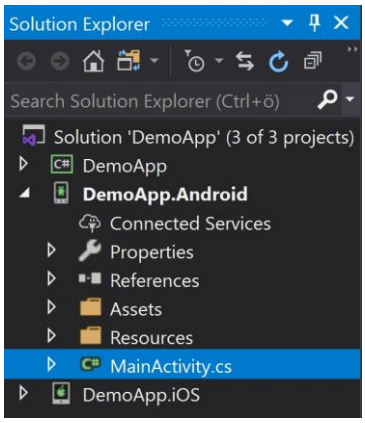
4. **Preview Changes** aknas kinnitame toimingud **OK** nupul vajutades (Joonis 23).



Joonis 23. Preview Changes aken

4.2. Sqlite seadistamine android platvormi jaoks

1. Avame **DemoApp.Android** projektis **MainActivity.cs** faili (Joonis 24).



Joonis 24. MainActivity.cs faili avamine

Android platvormi kasutajaliidese elemendid käivitatakse läbi LoadApplication() meetodi, milles käivitatakse App klass. Selleks, et kasutada SQLite andmebaasi paketti tuleb määrata andmebaasi asukoht.

1. Deklareerime enne **LoadApplication()** meetodit uue string tüüpi muutaja **fileName**, millele anname nimeks **post_db.db3** (Koodinäide 23).
2. Järgmisena deklareerimine uue string tüüpi muutaja **folderPath**, millele anname asukoha kuhu **post_db.db3** fail salvestatakse. Asukoha saamiseks kasutame **System.Environment** klassi meetodit nimega **GetFolderPath().** GetFolderPath() meetod vajab sisendiks kausta nime milleni teekond rajada. Lisame post_db.db3 faili **Personal** kausta, mis on androidi süsteemis juba olemasolev kaust. **Personal** kausta leiame kasutades **System.Environment** meetodit nimega **SpecialFolder**, millele lisame otsitava **Personal** kausta (Koodinäide 23).
3. Järgmisena deklareerimine uue string tüüpi muutuja **completePath**, millese kombineerimine kogu post_db.db3 faili asukoha teekonna. Kombineerimiseks kasutame **Path** klassi meetodit **Combine()**, millele anname kaasa **folderPath** ning **fileName** muutujad. Selleks, et kasutada Path klassi peame lisama **System.IO** teegi (Koodinäide 23).
4. Selleks, et completePath muutujat edasi anda DemoApp projekti anname completePath muutuja edasi LoadApplication() meetodis olevale **App()** klassile (Koodinäide 23).

```
using System;

using Android.App;
using Android.Content.PM;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
using System.IO;

namespace DemoApp.Droid
{
    [Activity(Label = "DemoApp", Icon = "@mipmap/icon", Theme = "@style/MainTheme",
MainLauncher = true, ConfigurationChanges = ConfigChanges.ScreenSize |
ConfigChanges.Orientation)]
    public class MainActivity :
global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
    {
        protected override void OnCreate(Bundle savedInstanceState)
        {
            TabLayoutResource = Resource.Layout.Tabbar;
            ToolbarResource = Resource.Layout.Toolbar;

            base.OnCreate(savedInstanceState);

            Xamarin.Essentials.Platform.Init(this, savedInstanceState);
            global::Xamarin.Forms.Forms.Init(this, savedInstanceState);

            string fileName = "posts_db.db3";
            string folderPath =
System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal);
            string completePath = Path.Combine(folderPath, fileName);

            LoadApplication(new App(completePath));
        }
        public override void OnRequestPermissionsResult(int requestCode, string[]
permissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)
        {
            Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode,
permissions, grantResults);

            base.OnRequestPermissionsResult(requestCode, permissions, grantResults);
        }
    }
}
```

Koodinäide 23. Android süsteemi andmebaasi faili lisamine (MainActivity.cs)

5. Avame **DemoApp** projektis **App.xaml.cs** faili. Näeme, et praegusel juhul **Public App()** konstruktor ei võta sisendiks midagi. Public App() konstruktori üle kirjutamiseks duplikeerimine public App() konstruktori ning anname sisendiks string tüüpi muutuja **filePath** (Koodinäide 24)**.**
6. Deklareerimine uue muutuja FilePath ning ülekirjutatud **public App(string filePath)** konstruktoris omistame FilePath muutujale konstruktorile sisendiks saadud filePath

muutuja, mille tulemusena on nüüd SqLite andmebaasi fail kättesaadav DemoApp projektis (Koodinäide 24).

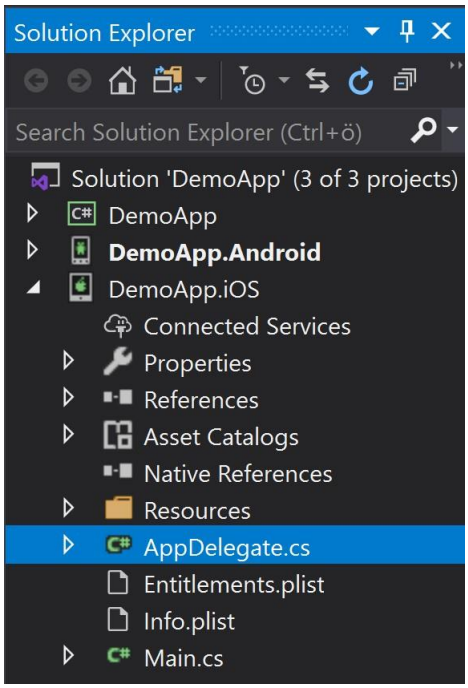
App.xaml.cs

```
using System;
using System.Collections.Generic;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace DemoApp
{
    public partial class App : Application
    {
       public static string FilePath;


       public static List<Post> Posts = new List<Post>();

        public App()
        {
            InitializeComponent();

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

        public App(string filePath)
        {
            InitializeComponent();

            //MainPage = new MainPage();
            MainPage = new NavigationPage(new MainPage());
            FilePath = filePath;
        }

        protected override void OnStart()
        {
        }

        protected override void OnSleep()
        {
        }

        protected override void OnResume()
        {
        }
    }
}
```

Koodinäide 24. Andmebaasi faili asukoha edasi andmine App klassile (App.xaml.cs)

4.3. SQLite seadistamine iOS platvormi jaoks

1. Avame **DemoApp.iOS** projektis **AppDelegate.cs** faili (Joonis 25).



Joonis 25. AppDelegate.cs faili avamine

2. Andmebaasi faili asukoha lisamiseks sooritame samasugused toimingud nagu eelnevalt Androidi platvormi puhul. Lisame muutuja **fileName** ning anname talle nimeks **posts_db.db3** (Koodinäide 25).

3. Erinevus tuleb sisse **folderPath** muutuja asukoha lisamisel, kuna iOS süsteemil on piirangud, kus andmebaasi fail võib asuda. Andmebaasi fail **posts_db.db3** peab asuma **Library** kaustas, mis asub samal tasemel Personal kaustaga. Selleks lisame **folderPath** muutujale tagasiliikumise ühe taseme võrra kaustast **Personal** ning edasiliikumise **Library** kausta kasutades **Path.Combine()** meetodit. Lisame ka **System.IO** teegi, et kasutada Path.Combine() meetodit (Koodinäide 25).

4. Lisame **complePath** muutuja, millele omistame andmebaasi faili kogu teekonna nii nagu tegime Androidi puhul.

   Lõpetuseks anname completePath muutuja edasi L**oadApplication()** meetodis olevale **App()** klassile. Siinkohal ei pea me enam App() klassis olevat konstruktorit muutma, kuna see on eelnevalt tehtud Androidi projekti seadistades (Koodinäide 25).

```
using System;
using System.Collections.Generic;
using System.IO;

using System.Linq;

using Foundation;
using UIKit;

namespace DemoApp.iOS
{
    // The UIApplicationDelegate for the application. This class is responsible for
launching the
    // User Interface of the application, as well as listening (and optionally responding)
to
    // application events from iOS.
    [Register("AppDelegate")]
    public partial class AppDelegate :
global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
    {
        //
        // This method is invoked when the application has loaded and is ready to run. In
this
        // method you should instantiate the window, load the UI into it and then make the
window
        // visible.
        //
        // You have 17 seconds to return from this method, or iOS will terminate your
application.
        //
        public override bool FinishedLaunching(UIApplication app, NSDictionary options)
        {
            global::Xamarin.Forms.Forms.Init();

            string fileName = "posts_db.db3";
            string folderPath =
Path.Combine(Environment.GetFolderPath(System.Environment.SpecialFolder.Personal), "..",
"Library");
            string completePath = Path.Combine(folderPath, fileName);

            LoadApplication(new App(completePath));

            return base.FinishedLaunching(app, options);
        }
    }
}
```

Koodinäide 25. iOS süsteemi andmebaasi faili lisamine(AppDelegate.cs)

4.4. SQLite funktsionaalsused klassis

1. Avame **DemoApp** projektis **post.cs** faili. Selleks, et kasutada SqLite paketi funtsionaalsuseid lisame **SQLite** paketi (Koodinäide 26).

   Sqlite funktsionaalsused määratakse atribuutidega. SQLite atribuudid aitavad SQLite paketil aru saada kuidas tabelit luua. SQLite atribuudid sisestatakse ennem klassi omadust kandiliste sulgude sisse. Tabeli nimeks kasutatakse vaikimisi klassi nime, mis on antud juhul Post. Ridade nimeks kasutatakse vaikimisi klassi omaduste nime. Enamkasutatavad SQLite klassi sisesed atribuudid on leitavad veebiaadressil: https://docs.microsoft.com/en-us/xamarin/android/data-cloud/data-access/using-sqlite-orm#sqlite-attributes

2. Kasutades **PrimaryKey** atribuuti saame muuta omaduse **Id** primaarvõtmeks. Lisame juurde ka automaatse juurdekasvu kasutades atribuuti **AutoIncrement**.

   Lisame postituse **Title** omadusele maksimaalse pikkuse 200 tähemärki kasutades atribuuti **MaxLenght()** (Koodinäide 26).

```
using SQLite;
using System;
using System.Collections.Generic;
using System.Text;

namespace DemoApp
{
    public class Post
    {
        [PrimaryKey, AutoIncrement]
        public int Id {
            get;
            set;
        }

        [MaxLength(200)]
        public string Title {
            get;
            set;
        }
        public DateTime Date {
            get;
            set;
        }


    }
}
```

Koodinäide 26. SqLite paketi funktsionaalsuste lisamine klassile (Post.cs)

4.5. Andmete sisestamine andmebaasi

1. Avame **AddPostPage.xaml.cs**. Loome **savePostButton_Clicked** meetodisse uue objekti **post**. Omistame omadusele **Title postTitle** lahtri väärtuse ning omadusele **Date** praeguse kellaaja. Omadust **Id** ei ole vaja deklareerida, kuna Id saab väärtuse automaatselt klassi siseselt (Koodinäide 27).

```
        private void savePostButton_Clicked(object sender, EventArgs e)
        {
            Post post = new Post()
            {
                Title = postTitle.Text,
                Date = DateTime.Now
            };


                App.Posts.Add(new Post() { Title = postTitle.Text, Date = DateTime.Now });
            Navigation.PushAsync(new MainPage());
        }
```

Koodinäide 27. Andmete ettevalmistus andmebaasi lisamiseks (AddPostPage.xaml.cs)

2. SqLite paketi funtsionaalsuste kasutamiseks lisame teegi **SQLite**.
3. Loome **savePostButton_Clicked** meetodisse andmebaasi ühenduse kasutades **using** lause süntaksit. Andmebaasiga ühenduse loomiseks kasutame **SQLiteConnection** klassi, millele anname kaasa andmebaasi faili asukoha. Andmebaasi faili asukoha lisasime eelnevalt **App** klassis olevale **FilePath** muutujale (Koodinäide 28).
4. Selleks, et sisestada andmeid tabelisse, peame esmalt looma tabeli, kasutades **SQLiteConnection** klassi **CreateTable<>** meetodit, millele anname kaasa tabeli, mida soovime luua. Kui eelnevalt on table loodud, siis SQLite pakett on piisavalt tark ning ignoreerib vastavat käsku (Koodinäide 28).
5. Andmete sisestamiseks kasutame **SQLiteConnection** klassi **Insert()** meetodit, millele anname kaasa eelnevalt loodud objekti **post** (Koodinäide 28).
6. Andmebaasi ühenduse sulgemiseks saame kasutada **SQLiteConnection** klassi **Close()** meetodit, kuid kuna kasutasime **using** lause süntaksit, siis **Close()** meetod kutsutakse esile automaatselt ning seda eraldi välja kutsuma ei pea.
7. Kustutame **savePostButton_Clicked** meetodis varasemalt loodud andmete lisamise loendisse (Koodinäide 28).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

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



namespace DemoApp
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class AddPostPage : ContentPage
    {


        public AddPostPage()
        {
            InitializeComponent();

        }

        private void savePostButton_Clicked(object sender, EventArgs e)
        {

            Post post = new Post()
            {
                Title = postTitle.Text,
                Date = DateTime.Now
            };

            using(SQLiteConnection conn = new SQLiteConnection(App.FilePath))
            {
                conn.CreateTable<Post>();
                conn.Insert(post);
            }

                App.Posts.Add(new Post() { Title = postTitle.Text, Date = DateTime.Now });
            Navigation.PushAsync(new MainPage());
        }

    }
}
```

Koodinäide 28. Andmete salvestamine andmebaasi (AddPostPage.xaml.cs)

4.6. Andmete lugemine andmebaasist

1. Avame **MainPage.xaml.cs** ning lisame **SQLite** teegi.
2. OnAppearing() meetodisse loome andmebaasi ühenduse kasutades using lause süntaksit, nii nagu tegime seda andmete sisestamisel andmebaasi (Koodinäide 29).
3. Loome uue tabeli kasutades **SQLiteConnection** klassi **CreateTable<>** meetodit nii nagu tegime seda andmete sisestamisel. See on vajalik rakenduse esmakordsel käivitamisel,

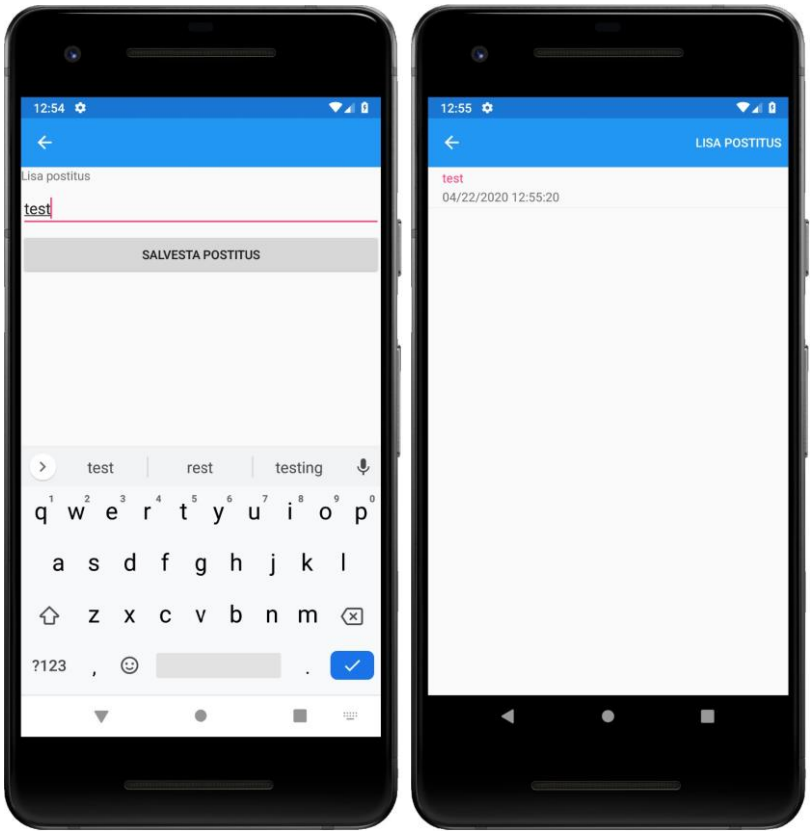
kuna siis pole veel andmebaasis tabelit loodud ning meil ei ole võimalik kusagilt andmeid küsida (Koodinäide 29).

4. Andmete küsimiseks saame kasutada **SQLiteConnection** klassi **Table<>** meetodit, millele anname kaasa soovitud tabeli nime. **Table<>** meetod tagastab kõik tabelis olevad objektid. Loome uue muutuja **posts** ning omistame muutujale **Table<>** meetodi ning viime andmed üle loendi kujule kasutades **System.Linq** klassi meetodit **ToList()**.
   Lõpetuseks omistame **Posts_ListView** vaate atribuudile loendi **posts** (Koodinäide 29).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using SQLite;


namespace DemoApp
{
    // Learn more about making custom code visible in the Xamarin.Forms previewer
    // by visiting https://aka.ms/xamarinforms-previewer
    [DesignTimeVisible(false)]
    public partial class MainPage : ContentPage
    {


        public MainPage()
        {
            InitializeComponent();



        }

        protected override void OnAppearing()
        {
            using (SQLiteConnection conn = new SQLiteConnection(App.FilePath))
            {
                conn.CreateTable<Post>();
                var posts = conn.Table<Post>().ToList();
                Posts_ListView.ItemsSource = posts;
            }


        }


        private void addPostToolbarItem_Clicked(object sender, EventArgs e)
        {
            Navigation.PushAsync(new AddPostPage());
        }
    }

}
```

Koodinäide 29. Andmete lugemine andmebaasist (MainPage.xaml.cs)

Pärast rakenduse käivitamist näeme, et andmed salvestatakse edukalt andmebaasi (Joonis 26).



Joonis 26. Rakenduse kuvahõive pärast andmebaaasi sisestamist

Soovitusi iseseisvaks õppimiseks

Oleme jõudnud edukalt õppematerjali lõppu. Soovitusi iseseisvaks õppimiseks:

- Muuta rakenduse kasutajaliidese kujundust.
  Informatsiooni rakenduse kujundamiseks leiab Xamarin.Forms dokumentatsioonist:
  https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/styles/xaml/
- Lisada postitusele lisamisele lisaks ka muid elemente (näiteks pilt, postituse Id) kasutades loendi vaate kohandatut lahtrit.
  Näiteid leiab Xamarin.Forms dokumentatsioonist:
  https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/listview/customizing-cell-appearance#custom-cells