Пловдивски университет „Паисий Хилендарски”
Факултет по математика и информатика

Курсова работа

По дисциплина „Програмиране на приложения за мобилни устройства“

На тема: „Телефонен указател“

Изготвил: Александър Йончев
Специалност: Софтуерни технологии и дизайн
Факултетен номер: 1701681058

Проверил:
/ доц. д-р Н. Касъклиев/

Пловдив, 2020

Съдържание

1. Увод

Приложението представлява телефонен указател, съхраняващ в локална база данни информация за контакти – име, телефонен номер, допълнително описание, категория и снимка на контакта.

1. Използвани технологии и библиотеки

- Visual Studio
- C#
- Xamarin.Forms – Технология позволяваща споделянето на огромно количество код за различните мобилни платформи – Android, iOS, Windows Phone, правеща създаването на мобилни приложения изключително лесно, посредством технологията XAML
- SQLite.NET – Малък ORM за работа със SQLite база данни, използван за лесно добавяне, взимане, изтриване, редактиране на данни, както и конвертирането им към домейн обекти
- Xam.Plugin.Media – Използван за избиране на снимка от устройството

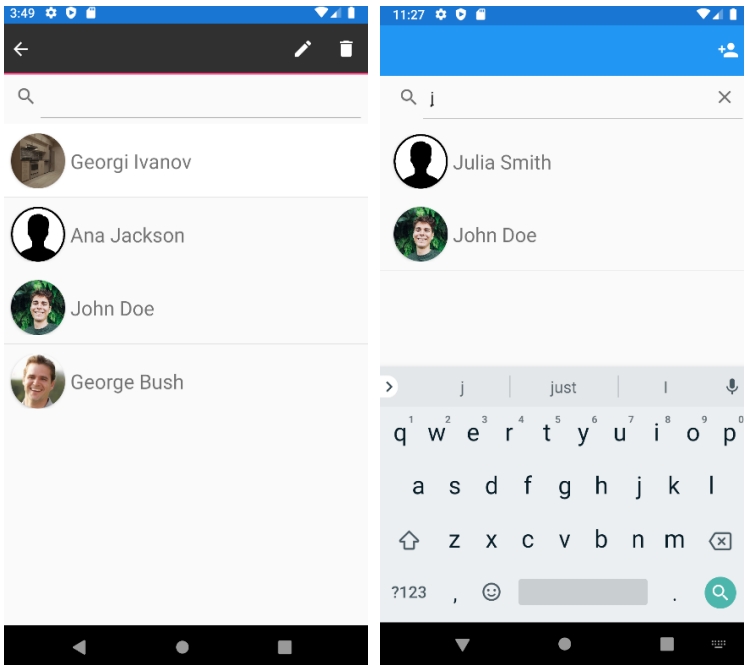
2. Изисквания към приложението

- Да пази информация за контакти в локална база данни
- Да пази име, телефонен номер, допълнително описание и категория на потребителя (познати, колеги, семейство, други)
- Да могат да се редактират, преглеждат и добавят контакти

3. Допълнително разработени функционалности

- Търсене по име на контакт
- Избиране на снимка, съхранявана локално на устройството, която да се изобразява, заедно с другата информация за контакта

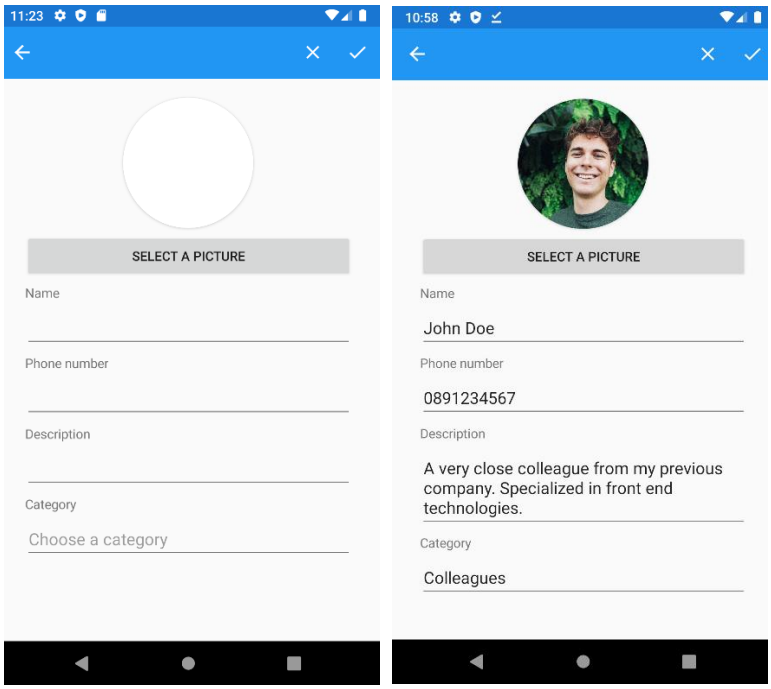
2. Потребителски инструктаж



Фигура 1 Основен екран - списък с контакти и възможност за търсене

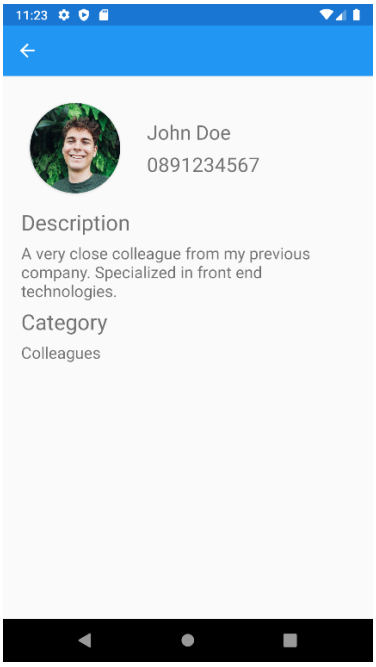
При зареждане на приложението се отваря основната страница, която представлява списък с въведените до момента контакти. Посредством полето за търсене над списъка, се вижда на Фигура 1, че щом въведем някакъв символ в полето, списъка се обновява и показва всички контакти, имената на които започват с въведения от нас низ.

По подразбиране в десния ъгъл в навигационната лента стои бутон за добавяне на нов контакт. При натискане и задържане на кой да е от елементите на списъка обаче, се отваря контекстно меню, от което може да се промени или изтрие съответния контакт. Когато пък се натисне кой да е елемент на списъка се отваря екран с детайли за съответния контакт.



Фигура 2 Екран за добавяне или промяна на контакт

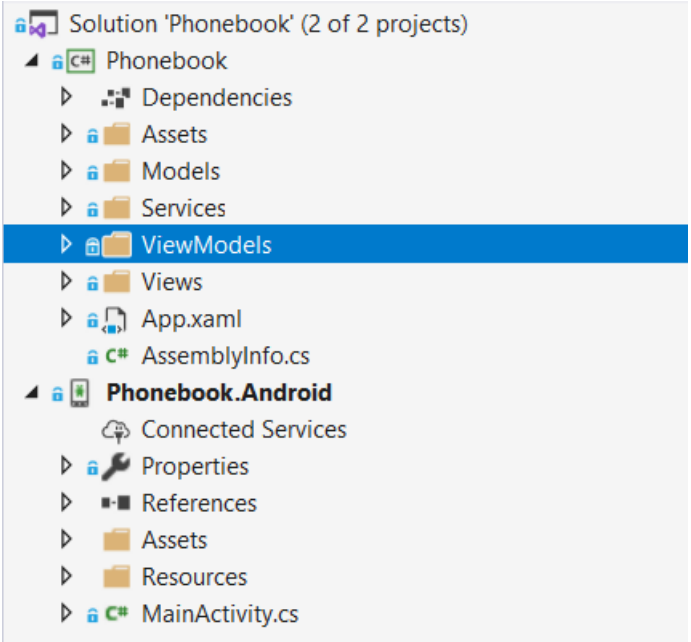
На Фигура 2 е изобразен екрана за добавяне или промяна на контакт, където се въвеждат данните за контакта и се избира снимката, ако потребителя желае да използва такава. Поради възможността полето за описание да бъде попълнено с по-голямо количество данни, екранът е скролируем.



Фигура 3 Екран с детайли за контакт

На Фигура 3 е изобразен екранът с детайлите за контакта.

3. Структура на проекта



Фигура 4 Дървовидна структура на проекта

На Фигура 4 е представена дървовидната структура на проекта. Понеже проекта е правен на C#, тук под проект визирам solution файла, обединяващ 2 проекта (.csproj файлове) – *Phonebook, Phonebook.Android*. Първият проект съдържа в себе си основната част от кода на логиката на приложението, която би могла да бъде споделена също така с iOS или UWP проект. Вторият проект съдържа конфигурационни файлове свързани с Android платформата. В него съм добавил иконите използвани в приложението, както и код за конфигурация на функционалността за избиране на снимки. Понеже Phonebook е проектът съдържащ по-голямата част от кода на приложението, ще се фокусирам върху него и ще обясня какво съдържа всяка една от директориите в него.

Основни директории и тяхното предназначение

- Assets – Директория, в която се съхраняват статични файлове, в случая единствено снимки, които могат да бъдат използвани от различните платформи.
- Models – Директория съдържаща домейн моделите на приложението. Тези модели се използват и за създаването на различните таблици в базата данни и съдържат в себе си атрибути, предоставени от SQLite.Net, за правила и лимити на различните колони в базата данни (максимална дължина на колона, primary key и др.).
- Services – Тук са включени класове играещи роля на сървиси, за достъпване на базата данни както и за работа с вградената навигация в Xamarin.Forms.
- ViewModels – Класове, които служат за пряка комуникация с потребителския интерфейс (View). Тук се съдържа основната логика на приложението, валидации, използване на базата данни и тн. Служат като медиатор между интерфейса (View) и моделите/данните. Използвани са за имплементацията на MVVM шаблона и да улеснят разрастването на приложението, неговото тестване и да разпределят по-добре отговорностите на отделните компоненти в приложението.

- Views – Тук се съдържат екраните/страниците на приложението структурата и стиловете, на която са описани с технологията създадена от Microsoft – XAML. Заедно с тях, тук е и съпътстващия ги “code-behind”, където се съдържа кода, който обработва различни събития и свързва View класовете със съответните им ViewModel класове.

4. Имплементация

1. Описване на таблица в базата

```
public class Contact : BaseEntity
{
    [MaxLength(50)]
    public string Name { get; set; }

    [MaxLength(20)]
    public string PhoneNumber { get; set; }

    [MaxLength(100)]
    public string Description { get; set; }

    public string PicturePath { get; set; }

    [Indexed]
    public int CategoryId { get; set; }
}
```

Фигура 5 Съдържание на Contact.cs

Contact.cs е домейн клас, който както за използване в приложението, служи и за описване на различните колони в таблицата Contact. Той наследява *BaseEntity.cs*, който е родителски клас на всички домейн класове, съдържащ едно свойство – Id. Чрез атрибутите *MaxLength* задаваме максимална дължина на данните в съответната колона.

2. Инициализиране и добавяне на записи по подразбиране

```
private void InitializeDatabase()
        var categoryList = new List<Category>
        {
           new Category() { Name= "Friends"},
           new Category() { Name= "Family"},
           new Category() { Name= "Colleagues"},
           new Category() { Name= "Other"}
        };
        db.Initialize(new Type[] { typeof(Category), typeof(Contact) });
        db.Seed(categoryList, categoryList.Count);
}
```

Фигура 6 Създаване и вмъкване (seed) на първоначални записи в App.xaml.cs

App.xaml.cs е стартиращият приложението клас, съдържащ различни конфигурации. Използван е *Database* класа и неговите методи *Initialize* и *Seed* за създаване на базата (ако вече не е създадена на устройството) и добавяне на първоначални категории. Методите са създадени синхронни, за да се осигури, че таблиците ще са създадени и първоначалните записи ще са добавени, когато за пръв път отваряме приложението.

3. Изтегляне на записи от базата

```
public async Task LoadData() {
        var contacts = await Database.GetItems<Contact>();
        categories = await Database.GetItems<Category>();
        initialContacts = contacts.Select(c => new ContactViewModel
        {
           Id = c.Id,
           Name = c.Name,
           Description = c.Description,
           PhoneNumber = c.PhoneNumber,
           PicturePath = c.PicturePath,
           Picture = ImageSource.FromFile(c.PicturePath),
           Category = categories.FirstOrDefault(ct => ct.Id == c.CategoryId)
        }).ToList();
        Contacts = new ObservableCollection<ContactViewModel>(initialContacts);
}
```

Фигура 7 Изтегляне на данни от базата в ContactsListViewModel.cs

На Фигура 7 е показано изтеглянето на данни в *ContactsListViewModel* и тяхното конвертиране към *ContactViewModel* обекти. Използва се свойство от тип Database, което е имплементирано в *BaseViewModel* класа, който е родител на всички ViewModel класове. **Важно е да се спомене, че работата с базата данни е винаги асинхронна с изключение на методите *Initialize* и *Seed*.**

```
public ContactsListViewModel ViewModel
{
        get { return BindingContext as ContactsListViewModel; }
        set { BindingContext = value; }
}
public ContactsListPage()
{
        InitializeComponent();

        ViewModel = new ContactsListViewModel();
}

protected async override void OnAppearing()
{
        await ViewModel.LoadData();
        base.OnAppearing();
}
```

Фигура 8 Създаване на връзката между ViewModel и View в ContactsListPage.xaml.cs

4. Визуализиране на данните

Последното нещо, което остава е да визуализираме данните и това става като създадем клас от тип *ContactsListViewModel* и го присвоим на *BindingContext* свойството на *ContactsListPage.xaml.cs*. По този начин правим връзката между потребителския интерфейс и данните. Детайлите по изобразяването на данните са описани в *ContactsListPage.xaml* файла, чийто код няма да показвам, понеже е доста дълъг и специфичен.

5. Заключение

Чрез това приложение успях да се запозная доста добре с технологията Xamarin.Forms и да разширя кръгозора си в .NET технологиите.

За тестването на приложението съм използвал както андроид емулатор, така и OnePlus 7 телефон, но е добре в бъдеще да се добавят и unit тестове. Оттук насетне към приложението

могат да се добавят множество функционалности – записване на записите в контактите на телефона, изпращане на съобщения, мейли и т.н.

Кодът на приложението е публикуван на GitHub Repository.

6. Библиография

Madeshvaran, S. (н.д.). *Xamarin.Forms MVVM: How to Work with SQLite DB(C# — Xaml)*. Извлечено от https://medium.com: https://medium.com/swlh/xamarin-forms-mvvm-how-to-work-with-sqlite-db-c-xaml-26fcae303edd

Microsoft. (н.д.). *Data Binding Basics*. Извлечено от https://docs.microsoft.com: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/xaml/xaml-basics/data-binding-basics

Microsoft. (н.д.). *Frame*. Извлечено от https://docs.microsoft.com: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/layouts/frame

Microsoft. (н.д.). *From Data Bindings to MVVM*. Извлечено от https://docs.microsoft.com: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/xaml/xaml-basics/data-bindings-to-mvvm

Microsoft. (н.д.). *Images in Xamarin.Forms*. Извлечено от https://docs.microsoft.com: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/images?tabs=windows