Курсова работа

По дисциплина „Програмиране на приложения за мобилни устройства“

На тема: „Мениджър на Тренировки“

Изготвил: Кирил Караколев
Специалност: СИ 3-ти курс 2020
Факултетен номер: 1801321015

Съдържание:

1. Увод

Целта на курсовия проект е да се изгради мобилно приложение за менажиране на тренировки.

2. Основни функционалности

- Съставяне на тренировки, съдържащи упражнения
- Създаване на упражнения
- Маркиране на тренировки като текущи
- Изтриване на тренировки

3. Използвани технологии

- Visual Studio
- Android SDK
- Xamarin.Forms
- SQLite-Net-pcl – използван за работа с база данни
- SQLiteNetExtensions – ORM за SQLite-Net-pcl
- SQLiteNetExtensions-async – асинхронна версия на QLiteNetExtensions

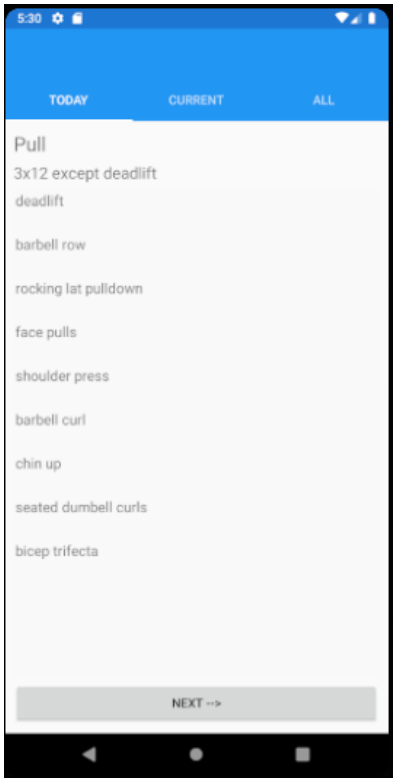
4. Начин на употреба

Основният интерфейс на приложението съдържа три раздела:

1 Днешна тренировка: *показва днешната тренировка, долу има бутон за преминаване към следващата (фиг. 1)*

2 Текущи тренировки: *показва всички текущи тренировки, които могат да бъдат пренаредени чрез натискане на бутона до тях (фиг. 2)*

3 Всички запазени тренировки *(фиг. 3)*



Фиг. 1

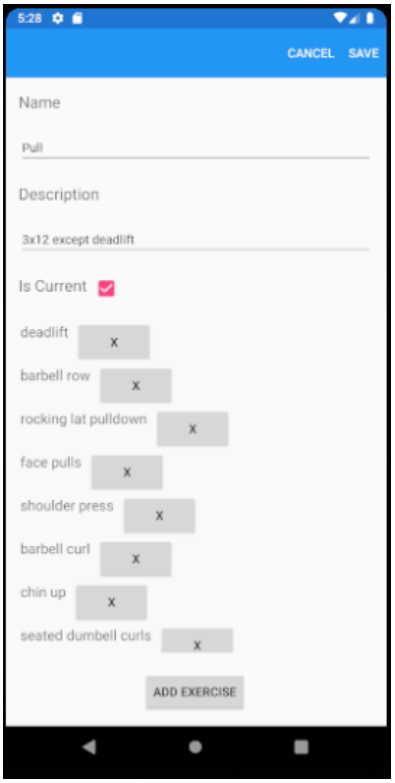
Фиг. 2

Фиг. 3

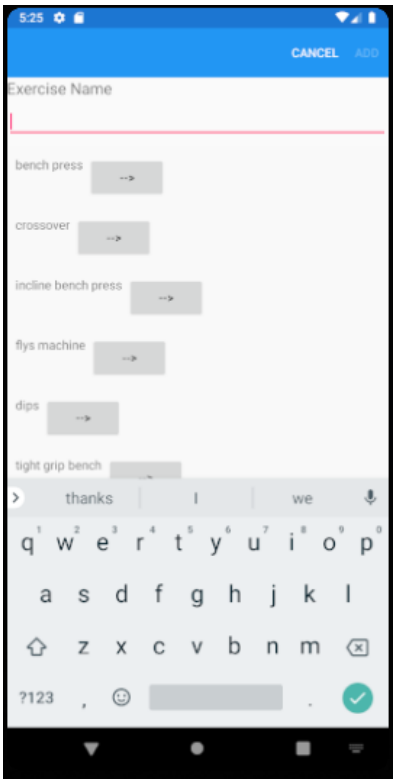
За създаване на тренировка се натиска бутона ADD в горния десен ъгъл. След натискане се отваря страница за създаване на нова тренировка (фиг. 4).

От страницата за създаване на тренировка могат да се добавят/създават упражнения чрез натискане на бутона за добавяне, който ще отвори нова страница. В новата страница можем да напишем името на упражнението, като вече създадени упражнения започващи със същото име ще се покажат и могат да бъдат добавени (фиг. 5).

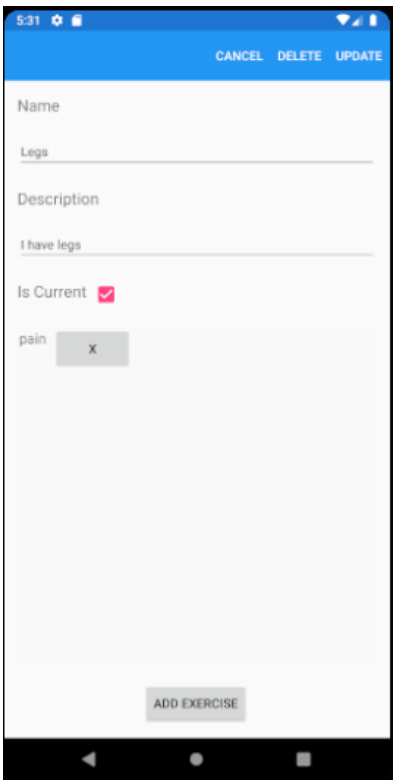
Вече създадена тренировка може да бъде редактирана/изтрита чрез натискане върху нея от разделите за текущи и всички тренировки (фиг. 6).



Фиг. 4



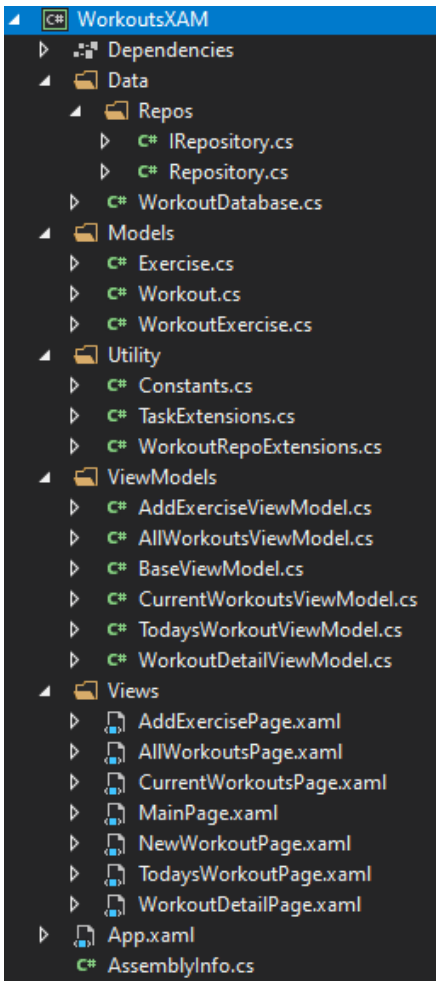
Фиг. 5



Фиг. 6

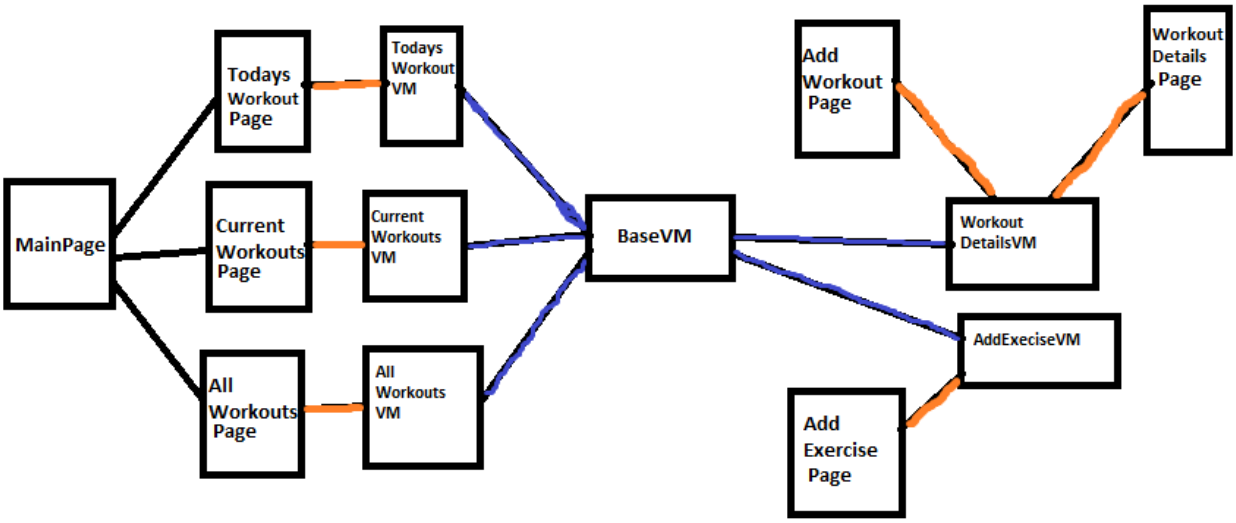
5. Архитектура

Фиг 7. Дървовидна структура на проекта



- **Data** – абстрактно ниво за работа с база данни
  - WorkoutDatabase – клас отговорен за работа с базата от данни, създаването и поддържането ѝ. Подсигурява, че имаме само една връзка към базата.
  - I/Repository – интерфейс и клас от общ тип дефиниращи достъп до дадена таблица от базата от данни.
- **Models** – съдържа всичките основни модели с които ще работи БД и приложението.
- **Utility** - съдържа помощни класове:
  - Constants – константи за БД като файлов път и опционални флагове.
  - TaskExtensions – разширителни методи за тип Task.
  - WorkoutRepoExtensions – разширителни методи за „repo“-та от тип Workout.
- **ViewModels** – съдържа класовете отговарящи за основната логиката на приложението.
- **Views** – съдържа класовете представящи потребителския интерфейс. Всеки клас е свързан към „view model“, чието състояние и данни визуализира, същевременно предавайки му инпута на потребителя

Фиг. 8 Свързаност на класовете **Синьо** – наследяване **Оранжево** – страница и съответния и view model



6. Имплементации

- Инициализация на БД

  - Фиг. 9 Инициализация на връзката към базата

```
static readonly Lazy<SQLiteAsyncConnection> lazyInitializer = new
Lazy<SQLiteAsyncConnection>(() =>
        {
            return new SQLiteAsyncConnection(Constants.DatabasePath, Constants.Flags);
        });

static SQLiteAsyncConnection Database => lazyInitializer.Value;
```

За връзка към базата се използва статично пропърти, което връща връзката от “lazy” инициализатор, т.е. връзката към базата се създава чак когато се използва за първи път.

  - Фиг. 10 Инициализация на класа

```
public WorkoutDatabase()
{
       InitializeAsync().SafeFireAndForget(false, (e) => { Debug.WriteLine("\n\n\n\n db
       init error: " + e.Message); });
}
```

Не можем да изчакваме (await) асинхронни методи в констуктора на клас, съответно не можем да прихванем хвърлени изключения (exception). За решаване на проблема използвам разширителен метод, който в случай на грешка изпълнява даден метод.

    - Фиг. 10.1 Разширителен метод

```
public static async void SafeFireAndForget(this Task task,
bool returnToCallingContext,
Action<Exception> onException = null)
{
        try
       {
        await task.ConfigureAwait(returnToCallingContext);
       }
       catch (Exception ex) when (onException != null)
       {
               onException(ex);
       }
}
```

  - Фиг. 11 Създаване на таблица в базата

```
if (!Database.TableMappings.Any(m => m.MappedType.Name == typeof(Workout).Name))
{
  await Database.CreateTableAsync(typeof(Workout),CreateFlags.None).ConfigureAwait(false);
}
```

Ако не съществъва таблица с име на съответния тип създаваме такава.

- Връзка между View(activity, ГПИ) и View Model

  - Фиг. 12 OnPropertyChanged и SetProperty методи

```
protected bool SetProperty<T>(ref T backingStore, T value,
    [CallerMemberName]string propertyName = "",
    Action onChanged = null)
{
    if (EqualityComparer<T>.Default.Equals(backingStore, value))
        return false;

    backingStore = value;
    onChanged?.Invoke();
    OnPropertyChanged(propertyName);
    return true;
}

public event PropertyChangedEventHandler PropertyChanged;
protected void OnPropertyChanged([CallerMemberName] string propertyName = "")
{
    var changed = PropertyChanged;
    if (changed == null)
        return;

    changed.Invoke(this, new PropertyChangedEventArgs(propertyName));
}
```

Всяка ГПИ страница (activity, view) е свързана към съответен “view model” (VM), чието състояние визуализира. Всеки VM трябва да инплементира „INotifyPropertyChanged“ интерфейса, който задължава да има “OnPropertyChanged” метод.
Всяко пропърти, чиято стойност бива използвана от ГПИ, би трябвало да извиква OnPropertyChanged при всяка промяна на стойноста си, но това може да доведе до проблем, ако стойнста му издва от ГПИ, тъй като при всяка промяна VM-а извиква метода, съответно ГПИ обновява стойноста и в резултат методът отново се извиква, създава се безкраен цикъл. Затова преди извикване на метода трябва да се сравнява дали има различна нова стойност. За удобство извикването на метода и проверката са обединени в един метод с име SetProperty.

- Фиг. 13 Команда

```
OpenWorkoutDetailsCommand = new Command<object>(async (x) => await OpenWorkoutDetailPage(x));
```

При вход от потребителя (натискане на бутон) от ГПИ се използва команда за да се предаде на съответния “view model”.

- Извличане и запис от/в базата

Фиг. 14 Зареждане на текущи тренировки

```
protected async override Task LoadWorkouts()
{
    IsBusy = true;
    try
    {
        Workouts.Clear();
        List<Workout> items = await Database.WorkoutRepo.Get<Workout>(w =>      w.IsCurrent != 0);
        items = items.OrderBy(w => w.IsCurrent).ToList();
        for(int i = 0; i < items.Count; i++)
        {
            items[i].IsCurrent = i + 2;
            await Database.WorkoutRepo.Update(items[i]);
            Workouts.Add(items[i]);
        }
    }
    catch (Exception ex)
    {
        Debug.WriteLine(ex);
    }
    finally
    {
        IsBusy = false;
    }
}
```

Всички методи за работа с базата са асинхронни.

- Визуализация на данните от “view model” в ГПИ

За визуализация на списъка от всички тренировки използвам “CollectionView”, чийто “ItemsSorce” е свързан (bound) към списъка. За визуализация на конкретна тренировка от списъка има дефиниран „Data Template“ (шаблон за данни). В конкретния пример (Фиг. 15) за всяка тренировка се показва името и описанието, допълнително има и „Gesture Recognizer“, който разпознава жестове от потребителя и извиква дадена команда към VM-а.

Фиг. 15 Визуализация на VM

```
<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"
             xmlns:viewModels="clr-namespace:WorkoutsXAM.ViewModels"
             mc:Ignorable="d"
             x:Class="WorkoutsXAM.Views.AllWorkoutsPage"
             Title="{Binding Title}"
             x:Name="AllWorkouts">

    <ContentPage.ToolbarItems>
        <ToolbarItem Text="Add" Command="{Binding OpenAddWorkoutCommand}" />
    </ContentPage.ToolbarItems>

    <RefreshView IsRefreshing="{Binding IsBusy, Mode=TwoWay}" Command="{Binding
LoadWorkoutsCommand}">
        <CollectionView x:Name="WorkoutsCollectionView"
                ItemsSource="{Binding Workouts}">
            <d:CollectionView.ItemsSource>
                <x:Array Type="{x:Type x:String}">
                    <x:String>First Item</x:String>
                </x:Array>
            </d:CollectionView.ItemsSource>
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <StackLayout Padding="10">
                        <Label Text="{Binding Name}"
                                d:Text="{Binding .}"
                                LineBreakMode="NoWrap"
                                Style="{DynamicResource ListItemTextStyle}"
                                FontSize="16" />
                        <Label Text="{Binding Details}"
                                d:Text="Item descripton"
                                LineBreakMode="NoWrap"
                                Style="{DynamicResource ListItemDetailTextStyle}"
                                FontSize="13" />
                        <StackLayout.GestureRecognizers>
                            <TapGestureRecognizer
                                Command="{Binding OpenWorkoutDetailsCommand,
Source={RelativeSource AncestorType={x:Type viewModels:AllWorkoutsViewModel}}}"
                                CommandParameter="{Binding .}"
                                NumberOfTapsRequired="1" />
                        </StackLayout.GestureRecognizers>
                    </StackLayout>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </RefreshView>
</ContentPage>
```

Източници

- Документация на Xamarin от Microsoft:
  - https://docs.microsoft.com/en-us/xamarin
  - https://docs.microsoft.com/en-us/xamarin/get-started/quickstarts/database?pivots=windows
  - https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/data/databases

- SQLite Net Extensions:
  - https://bitbucket.org/twincoders/sqlite-net-extensions/src/master/

- Случайни справки:
  - Навигация в Xamarin: https://stackoverflow.com/questions/43254396/xamarin-form-page-navigation-in-mvvm
  - Поп-ъп в Xamarin: https://stackoverflow.com/questions/52984946/how-do-i-create-an-alert-popup-from-my-viewmodel-in-xamarin
  - Collection View vs List View: https://stackoverflow.com/questions/56268458/what-is-the-difference-between-the-old-listview-and-new-collectionview-in-xamari
  - Как да показвам код в Word: https://stackoverflow.com/questions/387453/how-do-you-display-code-snippets-in-ms-word-preserving-format-and-syntax-highlig/51570356

- Личен опит, особено с WPF