**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ ВАРНА**

***Катедра „Софтуерни и интернет технологии”***



**Курсов проект**

**по Технология на софтуерното производство**

Изготвил: Неджиб Ахмед

Специалност: СИТ

Курс III, Група 1

Факултетен номер: 19621704

05.2022г.

Съдържание

[Техническо задание 2](#_Toc104840677)

[Анализ на проблема 3](#_Toc104840678)

[Функционални изисквания 3](#_Toc104840679)

[Структура на проекта 3](#_Toc104840680)

[Models 4](#_Toc104840681)

[Views 5](#_Toc104840682)

[ViewModels 5](#_Toc104840683)

[Helpers 5](#_Toc104840684)

[Repos 6](#_Toc104840685)

[Resources 6](#_Toc104840686)

[Services 6](#_Toc104840687)

[Проектиране на системата 7](#_Toc104840688)

[Модел на база данни 7](#_Toc104840689)

[Реализация на системата 7](#_Toc104840690)

[Реализация на базата 7](#_Toc104840691)

[Реализация на работата с базата 8](#_Toc104840692)

[Реализация на бизнес логика 9](#_Toc104840693)

[Реализация на графичен дизайн 11](#_Toc104840694)

[Бъдещи разработки 12](#_Toc104840695)

# Техническо задание

Очаква се да бъде разработено мобилно приложение по тема „Фитнес инструктор“. Целта на приложението е да подпомага в извършването на физическа тренировка за специфична мускулна група. Потребителите трябва да имат достъп до цялостна програма за тренировки­­­, както и генератор на единични тренировки, който да създава тренировка от произволни упражнения включени в общата база на приложението. Потребителите също така трябва да имат достъп до цялата база данни от упражнения и да могат да ги преглеждат свободно.

# Анализ на проблема

## Функционални изисквания

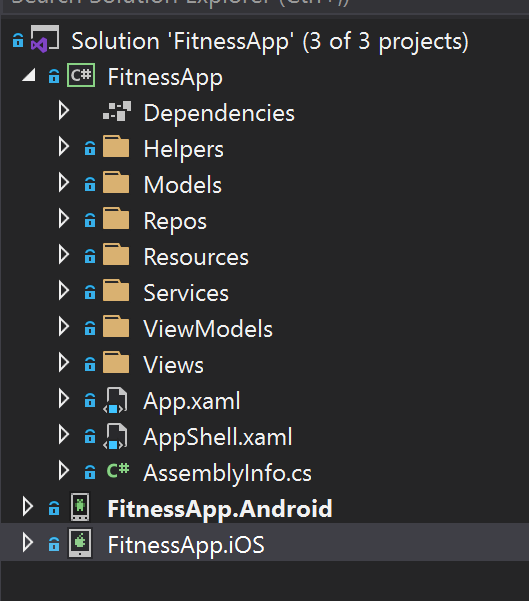
От приложението се изисква потребителите да имат достъп до цялостна програма за тренировки с продължителност до 4 седмици. Приложението ще включва упражнения само за коремната мускулна група. За всеки ден е определена специфична тренировка, както са определени и почивни дни. След избирането на деня за тренировка потребителят трябва да има възможността да прегледа предварително зададените му упражнения преди започване на тренировката. При започване на тренировката се започва таймер, който да отброява времето нужно за всяко упражнение и времето за почивка между упражненията.

Освен цялостна програма потребителите трябва да имат достъп и до генератор на тренировки. Тази функционалност представлява система, която чрез подбиране и разбъркване на наличните упражнения от базата да създаде единична тренировка подобна на тези от програмата. При генериране на тренировка потребителят отново трябва да има възможност да прегледа упражненията и при случай че иска да се създаде нова тренировка трябва да има възможността веднага да го направи без да има нужда да завършва вече генерираната.

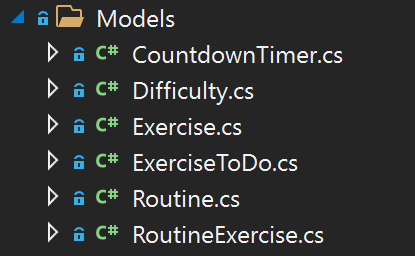
Също така потребителите трябва да имат достъп до пълния списък с упражнения налични в приложението и да могат свободно да ги преглеждат във всеки един момент. Може да се добави и допълнителен екран за външни ресурси, с които потребителите да научат повече за упражненията или да открият допълнителни тренировки.

## Структура на проекта

Проектът е изграден с помощта на framework – Xamarin Forms, който позволява изграждането на мобилни приложения на C#. Приложенията изградени на този framework се изграждат с MVVM архитектура и проекта е разделен на няколко модул. Основните модули са – Model, View и View Model модула. Тези модули включват графичния интерфейс на приложението, основните обекти, които ще се използват в него както и връзката между тези два модула. Останалите модули са помощни, които включват ресурсите използвани в приложението, източника на данни както и помощни класове използвани в бизнес логиката.

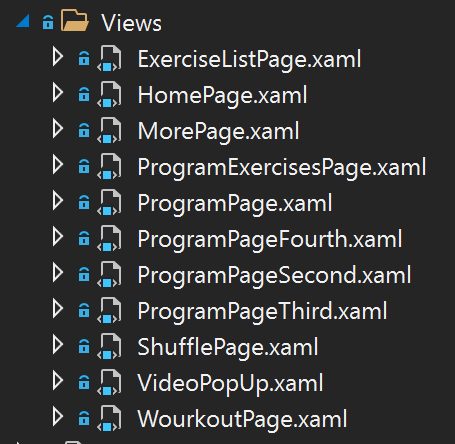


### Models



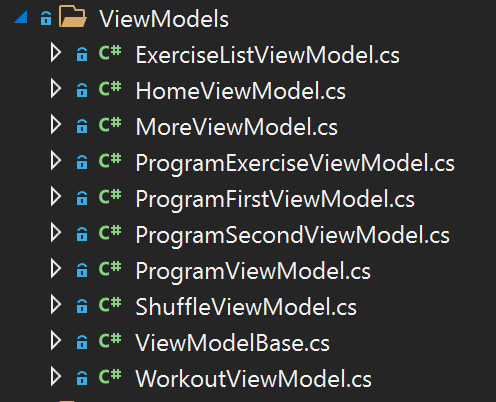
В този модул за класовете на обектите, които ще се използват в приложението, като например обектите за упражнение, трудности и т.н.

### Views



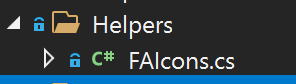
Views модула включва изгледите на приложението или всички екрани за графичния интерфейс, които потребителят може да види и които може да взаимодейства.

### ViewModels



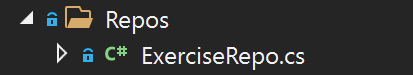
Този модул включва класовете, които преработват събитията извикани от графичния интерфейс при взаимодействието с потребителя. Като събития могат да бъдат натискането на бутони сменянето на екрани и всичко друго, с което потребителят може да влияе на графичния интерфейс на приложението.

### Helpers



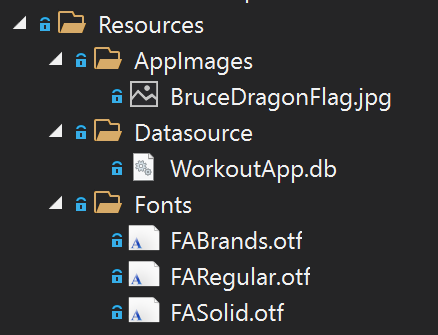
Модул за помощни класове като тук е включен само един клас, който съдържа информация за иконите използвани в графичния интерфейс на приложението.

### Repos



Repository модул, който включва клас, който съдържа данните на приложението. В случая тук само имаме общото хранилище на упражненията.

### Resources



Ресурсите представляват външни файлове, които се използват в приложението и които, ще бъдат запазени в устройството на потребителя при инсталиране на приложението. Това включва вътрешни изображения, базата данни и шрифтовете, към които са включени и иконите.

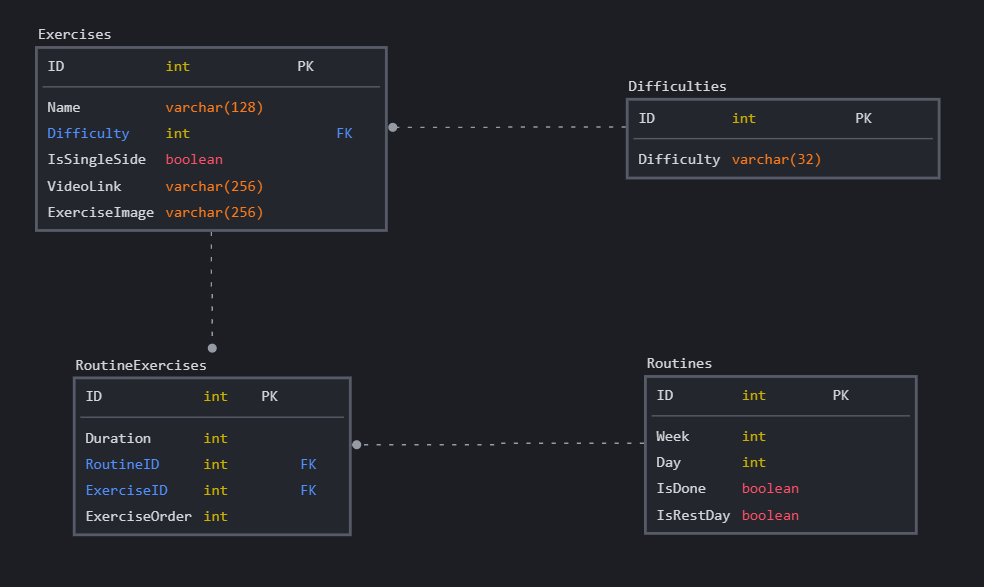
### Services



В Services модула са включени класовете, които отговарят за връзката между базата данни и хранилището на приложението. От тези класове главно се случва четенето на информацията от базата.

# Проектиране на системата

## Модел на база данни



Базата е съставена от четири таблици – Упражнения, Трудности, Рутини и Упражнения за рутини.

Таблицата упражнения държи всички упражнения, които да бъдат използвани в приложението.

Трудностите са за различните трудности на упражненията.

Таблицата за рутини описва всеки ден от програмата, като точния ден, седмицата, дали е завършена тренировката за деня и дали денят е почивен.

Таблицата за упражнения за рутини е таблицата включваща всяко упражнение за определената рутина.

# Реализация на системата

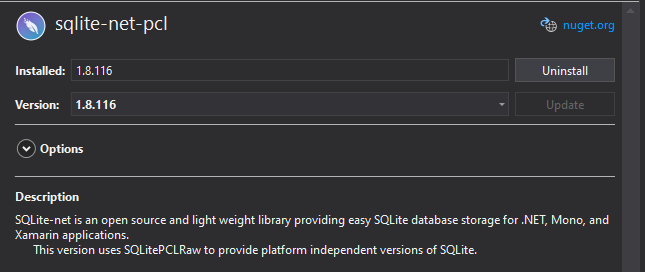
За цялостната реализация на приложението е използван – Xamarin Forms. Това е framework на C#, който позволява изграждането на мобилни, десктоп приложения. Този framework позволява изграждането на мобилни приложения едновременно за Android и iOS, което е голямо предимство, тъй като не е нужно приложението първо да се изгради за едната платформа и после да се адаптира за следващата.

Xamarin Forms използва MVVM (Model-View-ViewModel) модел за разработка. Чрез този модел приложението се разделя на три модула – графичен интерфейс (View), данни (Model) и бизнес логика (View-Model), която да свърже двете. Framework-а има вграденa поддръжка на този модел като позволява лесно свързване на данните между графичния интерфейс и бизнес логиката.

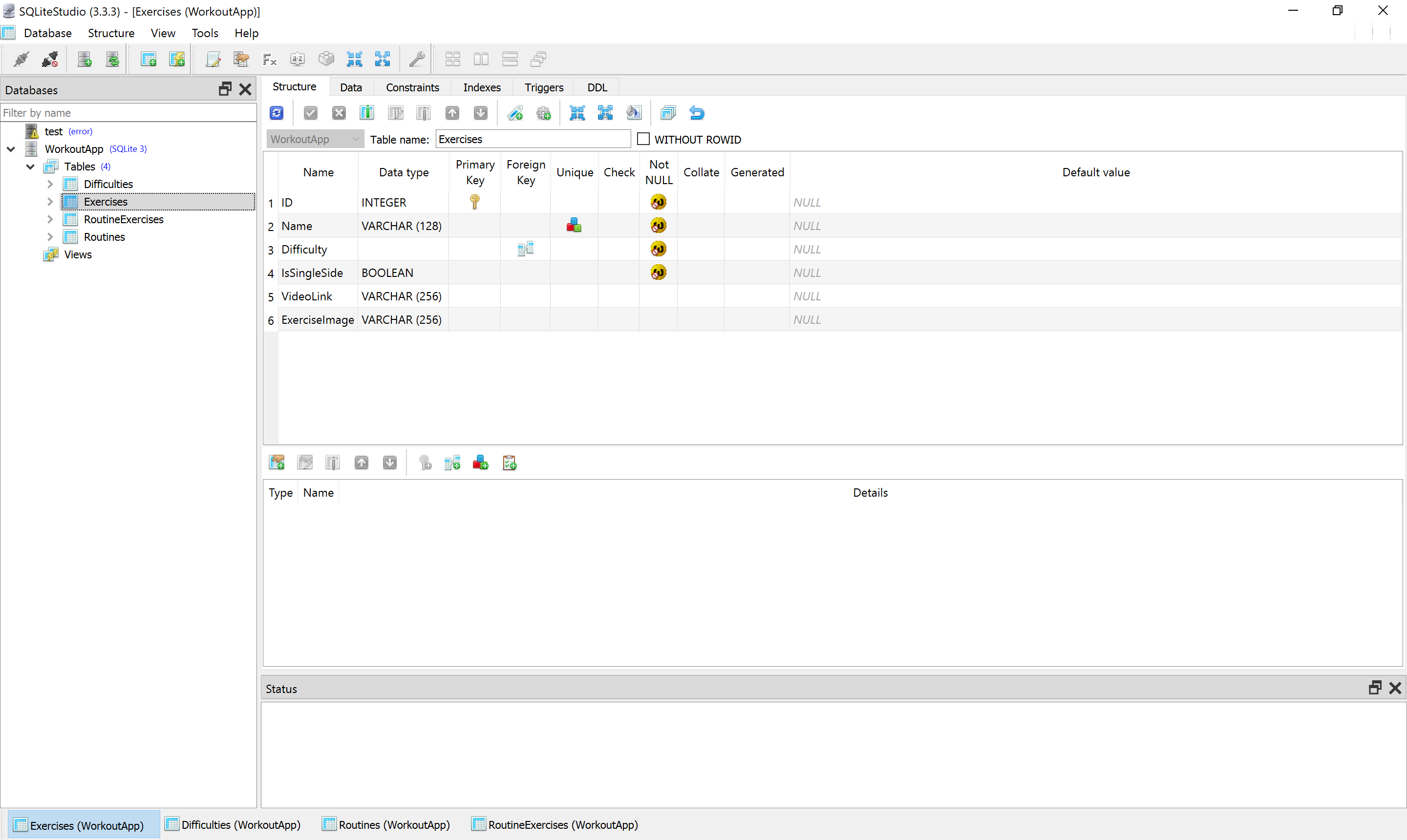
Също така чрез този framework могат да се достъпват и вградените функционалности на всяка платформа.

## Реализация на базата

Базата данни е реализирана с помощта на SQLite, която е изключително бърза и лека база данни. Тя се използва в голям брой машини именно заради тези характеристики, както се използва и във всеки телефон, затова беше естественият избор за това приложение. Данните от приложението се съхраняват локално в телефона и се използва готова библиотека за Xamarin Forms sqlite-net-pcl, която улеснява работата с базата данни в приложението.

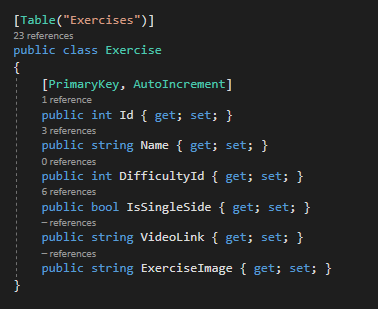


За създаване на самата база е използвана програмата SQLiteStudio, която предоставя удобен графичен интерфейс за по-лесно управление на базата.

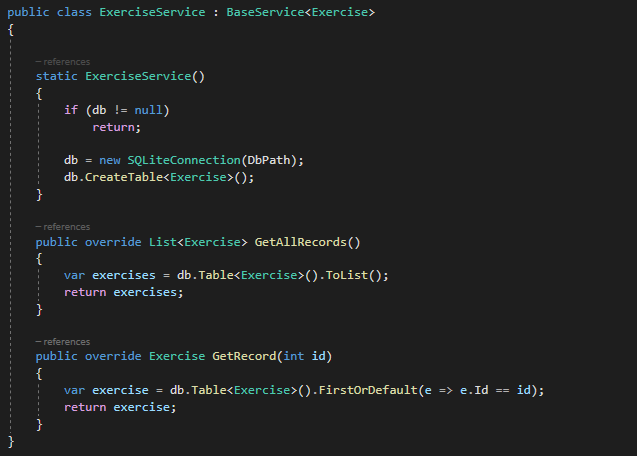


## Реализация на работата с базата

За работа с библиотеката се създават класове, които да отговарят на таблиците в базата и им се поставят атрибути, които да покажат съответните свойства на таблиците от базата (като primary key, връзки и т.н.).



След, което се създават сервизни класове, които служат за обслужване на заявките към базата, като зареждане на данни, редактиране и изтриване.

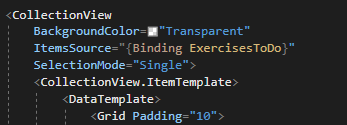


При извличането на данни от базата, получените данни могат да се преобразуват в колекция от тип списък за по-лесно използване в цялостното приложение.

## Реализация на бизнес логика

Бизнес логиката на приложението е реализирана във ViewModel модула. В този модул всеки отделен екран от приложението има отговарящ ViewModel, който да реализира бизнес логиката на определения екран. Тук се случва цялата обработка, когато потребителят взаимодейства с някой от екраните на приложението.

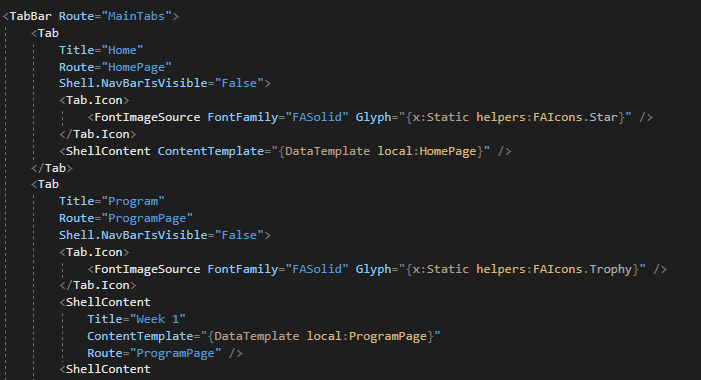
Един от основните цели на приложение с повече от един екран е свързването на данни между тези различни екрани. В Xamarin Forms лесното свързване на данните става чрез така наречения Data Binding позволява екраните в приложението да бъдат синхронизирани с лекота. Това е възможно именно заради MVVM модела на разработка и функцията на Data Binding елиминира нуждата от използване на събития свързани към всички данни от графичния интерфейс и изпращането на известия при промяната на тези данни. Обикновено това свързване на данни се прави в модула на графичния интерфейс на приложението, за да спести повече код от backend-a.





Ето тук графичният елемент за списъка с данни от интерфейса се свързва с колекцията от данни от ViewModel-а за да се изобрази пълният списък с упражнения в приложението.

Навигацията в приложението се реализира в ViewModel модула и View модула. Като във View модула при създаване на екраните и отделните tab-ове на приложението в AppShell-а, който е контейнер, съдържат навигационните пътища на приложението, се задават и имената на пътищата им чрез route атрибута.



А екрани, които не присъстват в тези основни tab-ове на приложението трябва да имат регистрирани пътища в backend-a на AppShell-a.

## Реализация на графичен дизайн

В Xamarin Forms графичният интерфейс на приложението се реализира чрез Extensible Application Markup Language или накрактко XAML. Това е език базиран на XML, който цели да опрости създаването на графични интерфейси. Основните предимства на XAML са че разделя UI от бизнес логиката и че позволява едновременно да са работи върху логиката и интерфейса на една програма с различни инструменти. Елементите на XAML са директно свързани със CLR обекти, а атрибутите – с CLR свойства и събития върху тези обекти.



Писането на XAML наподобява писането на HTML, като самият екран се състои от елементи, които са описани с отварящ и затварящ таг и тези елементи могат да притежават разнообразни свойства. В случая тук имам елемент за списък CollectionView, което да представя данните в списъчна структура и вътре в нея е описана оформлението на всяка единица данни в този списък. Също така е важно да се отбележи задаването на BindingContext – това е елемент, който задава ViewModel-а, който ще е свързан към този елемент.

# Бъдещи разработки

За бъдеще в приложението може да се разработи по-детайлен генератор на упражнения, който да позволява избиране на различни нива на трудност на тренировките, за което може да се наложи по-сложен алгоритъм за подбор на упражненията, така че тренировките да удовлетворяват нуждите на потребителите по по-коректен начин. Също така може да се включи система за профили на различни видове потребители – например треньори и клиенти. Като треньорите да имат възможността да създават свои собствени тренировки с помощта на приложението, както и да вкарват допълнителни упражнения към общата база. Може да се добави и възможност за избиране на повече от една тренировъчна програма, както и повече мускулни групи. При наличието на повече данни в приложението, нужна разработка е комуникация на приложението със сървър, в който да се съхраняват данните. Това може да стане с помощта на REST API .

Също така направих опит да включа функционалност за пускане на клипове при натискане на изображението на упражнението. Но за жалост библиотеката, която използвах не проработи и за бъдеща разработка би било доста полезна функционалност да се вмъкне възпроизвеждането на клипове на упражнения по техния URL от YouTube.