Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Операционные системы и системное программирование(ОСиСП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

Комплексное программное средство для автоматизации учебных процессов. Мобильный клиент.

Студент: гр. 551006 Жизневский В.С.

Руководитель: Трус В. В.

Минск 2017

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 7](#_Toc501522054)

[**1** **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ** 8](#_Toc501522055)

[**1.1** **Surveymonkey** 8](#_Toc501522056)

[**1.2** **Simpoll** 9](#_Toc501522057)

[**1.3** **Постановка задачи** 10](#_Toc501522058)

[**2** **МОДЕЛИРОВАНИЕ ПРИЛОЖЕНИЯ** 11](#_Toc501522059)

[**2.1** **Описание используемых ресурсов и паттернов программирования** 11](#_Toc501522060)

[2.1.1 С# 11](#_Toc501522061)

[2.1.2 Microsoft Visual Studio 12](#_Toc501522062)

[2.1.3 Xamarin 12](#_Toc501522063)

[2.1.4 Xamarin Forms 14](#_Toc501522064)

[2.1.5 XLabs 15](#_Toc501522065)

[2.1.6 Prism 15](#_Toc501522066)

[2.1.7 MVVM 16](#_Toc501522067)

[2.1.8 Dependensy injection. Inversion of control. Container. 17](#_Toc501522068)

[2.1.9 Веб-сервисы 18](#_Toc501522069)

[**3** **ПРОЕКТИРОВАНИЕ. РАЗРАБОТКА** 19](#_Toc501522070)

[**3.1** **Структура приложения** 19](#_Toc501522071)

[**3.2** **Модуль CourceProject.MC** 21](#_Toc501522072)

[3.2.1 Описание функциональности 21](#_Toc501522073)

[3.2.2 Inversion of Control Container в Prism 21](#_Toc501522074)

[3.2.3 Navigation service в Xamarin и Prism 21](#_Toc501522075)

[3.2.4 Привязка страниц к ViewModel в XamarinForms и Prism 23](#_Toc501522076)

[**3.3** **CourceProject.MC.DomainModel** 25](#_Toc501522077)

[3.3.1 Описание функциолаьности 25](#_Toc501522078)

[3.3.2 Модель опроса 25](#_Toc501522079)

[**3.4** **Модуль CourceProject.MC.UI** 26](#_Toc501522080)

[3.4.1 Описание функциональности 26](#_Toc501522081)

[3.4.2 Описание страниц 26](#_Toc501522082)

[3.4.3 SplashPage 29](#_Toc501522083)

[3.4.4 SignUpPage 29](#_Toc501522084)

[3.4.5 HomePage 30](#_Toc501522085)

[3.4.6 GetSurveyPage 30](#_Toc501522086)

[3.4.7 SelectedSurveyPage 31](#_Toc501522087)

[3.4.8 Конструктор представления опроса 31](#_Toc501522088)

[3.4.6.1 Базовые классы QuestionViewBuilderBase и QuestionViewModelBase 33](#_Toc501522089)

[3.4.6.2 SingleLine 33](#_Toc501522090)

[3.4.6.3 MultiLine 35](#_Toc501522091)

[3.4.6.4 CheckBoxGroup 36](#_Toc501522092)

[3.4.6.5 RadioButtonGroup 37](#_Toc501522093)

[3.4.6.6 DateInput 38](#_Toc501522094)

[3.4.9 Custom controllers 39](#_Toc501522095)

[3.4.10 Валидация ответов перед отправкой 40](#_Toc501522096)

[**3.5** **Модуль CourceProject.MC.Common** 41](#_Toc501522097)

[3.5.1 Описание функциональности 41](#_Toc501522098)

[3.5.2 Регулярные выражения 41](#_Toc501522099)

[3.5.3 Атрибуты 41](#_Toc501522100)

[**3.6** **Модуль CourceProject.MC.Resources** 41](#_Toc501522101)

[3.6.1 Описание функциональности 41](#_Toc501522102)

[**3.7** **Модуль CourceProject.MC.Android** 41](#_Toc501522103)

[3.7.1 Описание функциональности 41](#_Toc501522104)

[3.7.2 Структура модуля 41](#_Toc501522105)

[3.7.3 Класс MainActivity 42](#_Toc501522106)

[**3.8** **Модуль CourceProject.MC.iOS** 42](#_Toc501522107)

[3.8.1 Описание функциональности 42](#_Toc501522108)

[3.8.2 Структура модуля 42](#_Toc501522109)

[3.8.3 Стартовая точка приложения. Класс Main. Класс AppDelegate 42](#_Toc501522110)

[**3.9** **Модуль CourceProject.MC.Foundation** 42](#_Toc501522111)

[3.9.1 Описание функциональности 42](#_Toc501522112)

[3.9.2 SurveyWebService 43](#_Toc501522113)

[3.9.3 SurveysManager 43](#_Toc501522114)

[3.9.4 SettingsService 43](#_Toc501522115)

[3.9.5 NetworkManager 43](#_Toc501522116)

[3.9.6 AuthenticationWebService 43](#_Toc501522117)

[**4** **ПРОВЕРКА КОРРЕКТНОСТИ РАБОТЫ ПРОГРАММНОГО СРЕДСТВА** 44](#_Toc501522118)

[**5** **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ** 47](#_Toc501522119)

[**5.1** **Системные требования** 47](#_Toc501522120)

[**5.2** **Запуск программы.** 47](#_Toc501522121)

[**5.3** **Авторизация** 48](#_Toc501522122)

[**5.4** **Главная страница** 49](#_Toc501522123)

[**5.5** **Страница получения опроса** 50](#_Toc501522124)

[**5.6** **Страница прохождения опроса** 51](#_Toc501522125)

[**ЗАКЛЮЧЕНИЕ** 52](#_Toc501522126)

[Список использованных источников 53](#_Toc501522127)

[**ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ** 54](#_Toc501522128)

[**ПРИЛОЖЕНИЕ Б. СХЕМА АЛГОРИТМА ФОРМИРОВАНИЯ ПРЕДСТАВЛЕНИЯ ОПРОСА И ПРИВЯЗКИ К VIEWMODEL.** 89](#_Toc501522129)

# **ВВЕДЕНИЕ**

В современной практике образования достаточно большое значение имеет время, затрачиваемое на выполнение типичных процессов, таких как проведение самостоятельных и контрольных работ, проверка посещаемости и ведение журнала с состоянием сдачи лабораторных работ.   
Данные процедуры — основная часть документационной работы всех преподавателей университета. Так как такие мероприятия необходимы для корректной организации процесса создания систем и техник образования, однако большая часть из них, такие как самостоятельные работы, могут отнимать достаточно большие объёмы времени, так как помимо самого проведения самостоятельных работ, необходимо также в дальнейшем проверить написанные студентами ответы на предмет корректности.   
 В современном мире каждая область нашей жизни проходит процесс перехода к использованию программных и аппаратных средств. Яркими примерами процесса перехода к автоматизированным процессам могут служить такие службы и учреждения, как банки, общественный транспорт, дизайн и т.д., что свидетельствует о широком распространении использования информационных технологий.   
 В процессе получения образования в учреждении образования БГУиР довольной низкий процент имеет использование тех или иных средств автоматизации, соответственно, процессы проведения и организации образовательных мероприятий затрачивает достаточно большой объём времени.   
 Целью данной курсовой работы является создание программного средства, которое будет обладать простотой в использовании, и которое будет способствовать автоматизации образовательных процессов, а также будет иметь возможность дальнейшего переиспользования.   
 Программное средство будет состоять из нескольких частей:

* мобильное приложение для студентов, в котором будут находиться работы, которые студенты могут пройти;
* RESTful API ( в дальнейшем, API ), предоставляемый для использования остальным частям программного средства;
* веб-приложение, которое будет служить для создания шаблонов для самостоятельных работ.

В данной части курсового проекта будет рассмотрено создание мобильного приложения.

# **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ**

Особенностями мобильных приложений систем опроса является то, что практически во все они предназначены для создания опросов, и очень мало мобильных клиентов имеют в себе возможность прохождения данных опросов. Также все они предназначены для решения большого круга задач, и мало какие из них заточены под конкретную предметную область.

## **Surveymonkey**

Данная платформа включает в себя как сайт, так и клиент для мобильных устройств (рисунок 1.1). В частности мобильный клиент позволяет создавать опросы, рассылать по электронной почте, в SMS-сообщениях или через социальные сети. Также мобильный клиент позволяет просматривать результаты опроса в виде всяких графиков и диаграмм.

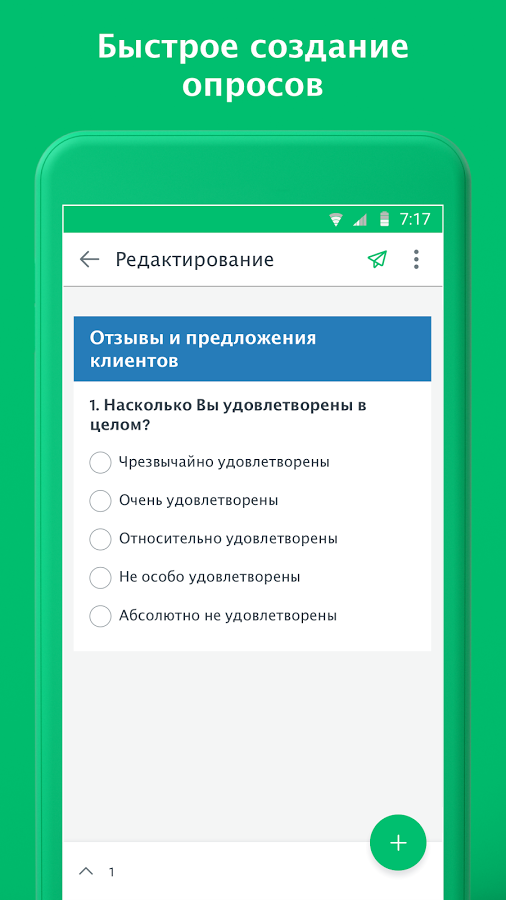


Рисунок 1.1 – Surveymonkey.

Однако данное приложение имеет лишь часть функционала, которое возможно будет реализовано в дальнейшем в последующих версиях курсового проекта, а именно просмотр статистики пользователя. Возможность прохождения опросов данное приложение не имеет.

## **Simpoll**

Данный клиент позволяет создавать вопросы, рассылать данные вопросы и вести статистику ответов (рисунок 1.2).

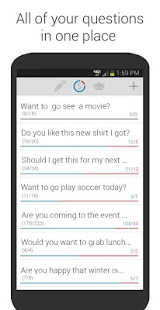
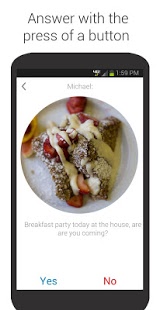


Рисунок 1.2 – Simpoll.

Недостатком данного клиента является отсутствие возможности группировки вопросов в опросы.

На основе анализа аналогов было решено реализовать мобильный клиент для системы автоматизации прохождения опросов.

## **Постановка задачи**

Разработать мобильный клиент, являющийся частью системы автоматизации прохождения опросов. 1-я версия клиента должна реализовывать следующий функционал:

* взаимодействие с сервером;
* аутентификацию пользователей;
* получение опроса по Id;
* формирование представления опроса;
* редактирование ответов пользователем;
* валидация ответов на обязательные вопросы;
* отправка ответов на сервер.

В последующих версиях реализовать следующий функционал:

* просмотр статистики пользователя;
* push-нотификации;
* офлайн-режим;
* создание хотспот точки для раздачи опроса;
* конструктор опросов;
* реализация механизма сессий;
* и т.д.

# **МОДЕЛИРОВАНИЕ ПРИЛОЖЕНИЯ**

Данная система, часть которой будет являться мобильный клиент, будет состоять из сайта, клиента для мобильного телефона (Xamarin + Xamarin.Forms) и сервера(ASP.NET MVC).

В рамках данной курсовой работы будет рассмотрена разработка клиента для мобильных платформ.

В качестве основной платформы для разработки данного приложения было решено взять фрэймворк Xamarin, т.к. он позволяет вести кросс-платформенную разработку приложений для мобильных платформ и создавать нативные приложения, базируясь на общем коде для всех платформ.

Исходя из выбранной платформы, в качестве архитектуры был выбрана архитектура, характерная для приложения, разработанного на Xamarin + Xamarin.Forms (п. 2.1.3, п. 2.1.4, рисунок 2.3).

1. **Описание используемых ресурсов и паттернов программирования**

Разработка данного приложения будет происходить в IDE Microsoft Visual Studio. Основным языком будет C#, а также будет использоваться язык разметки XAML. Основным фреймворком, на котором будет вестись разработка приложения, будет Xamarin, базируемый на платформе Mono. Также будут использоваться фрэймворки: Xamarin Forms, Prism, XLabs.

### С#

C**#** — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Pascal, и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

### Microsoft Visual Studio

Microsoft Visual Studio — продукт компании Microsoft, включающий интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данный продукт позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

### Xamarin

Xamarin — это фреймворк для кроссплатформенной разработки мобильных приложений (iOS, Android, Windows Phone) с использованием языка C#. Xamarin позволяет писать с применением всех основных возможностей С# и платформы .Net, например LINQ, лямбда-выражения, Generic типы и async. При имеется полный доступ ко всем возможностям SDK платформы и нативному механизму создания UI, получая на выходе приложение, которое, практически ничем не отличается от нативных.  
  
 Фреймворк состоит из нескольких основных частей:

* Xamarin.IOS — библиотека классов для C#, предоставляющая разработчику доступ к iOS SDK;
* Xamarin.Android — библиотека классов для C#, предоставляющая разработчику доступ к Android SDK;
* компиляторы для iOS и Android;
* IDE Xamarin Studio;
* плагин для Visual Studio.

Xamarin основан на open-source реализации платформы .NET — Mono. Эта реализация включает в себя собственный компилятор C#, среду выполнения, а так же основные .NET библиотеки. Основное назначение — позволить запускать программы, написанные на C#, на операционных системах, отличных от ОС Windows — Unix-системах, операционной системы Mac OS и других. Разработчиками Mono и Xamarin является компания Xamarin.

Xamarin содержит в себе компиляторы для ОС Android и операционной системы iOS, которые имеют разные принципы работы. С точки зрения исполнения приложений между ОС iOS и ОС Android есть одно ключевое различие — способ их предварительной компиляции. Для выполнения приложений в Android используется виртуальная Java-машина Dalvik. Нативные приложения, которые пишутся на Java, компилируются в некий промежуточный байт-код, который интерпретируется Dalvik`ом в команды процессора в момент исполнения программы. Это так называемая Just-in-time компиляция. В iOS используется другая модель компиляции — Ahead-of-Time (рисунок 2.1).

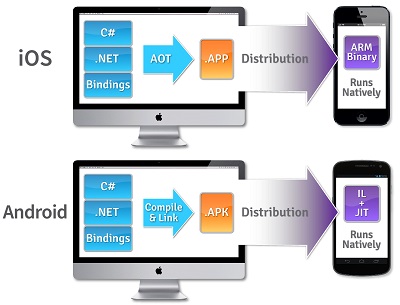


Рисунок 2.1 - Схема создания и функционирования приложений.

В случае ОС iOS принцип выполнения следующий — программный код заранее скомпилирован в машинный. Для этой цели используется AOT компилятор Mono.

Принцип выполнения в ОС Android другой. При компиляции приложения происходит перевод кода на C# в промежуточный байт-код, понятный виртуальной машине Mono и сама эта виртуальная машина также добавляется в упакованное приложение. И Mono и Dalvik написаны на языке Си и работают поверх ядра ОС Linux. При запуске приложения на ОС Android обе виртуальные машины начинают работать бок о бок и обмениваются данными через специальный механизм wrapper`ов (рисунок 2.2).

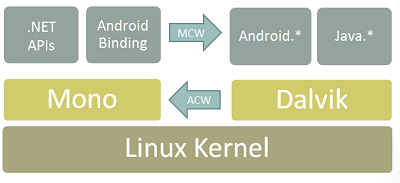
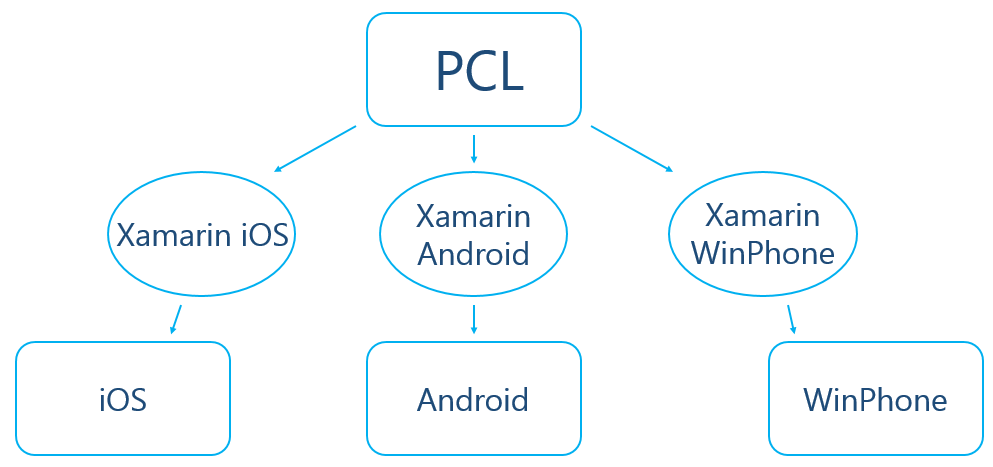


Рисунок 2.2 - Схема взаимодействия Mono и Dalvik.

### Xamarin Forms

Xamarin.Forms — позволяет использовать одну библиотеку элементов (текстовые поля, кнопки, переключатели, надписи, списочные представления и т.д.) для всех систем (Android, iOS, Windows).   
 Xamarin.Forms позволяет создавать элементы в Xamarin приложении, а на конкретной платформе(iOS или Android или Windows) каждого вашего элемента вызывается генерация и рисуются родные элементы(на iOS свои кнопки, для Android свои кнопки и т.д.).   
 Добавлять элементы можно как в Xamarin C# коде так и в XAML-формате.

Структура приложения Xamarin.Forms представлена на рисунке 2.3.



Русунок 2.3 – Структура приложения Xamarin.Forms.

Вверху схемы расположена PCL(Portable class library. По сути это и есть Xamarin.Forms. В общих чертах он представляет собой набор редакторов, навигационных панелей, layout-панелей и так далее. При разработке пользовательского интерфейса большую часть времени работаешь как раз с ними. Однако, данные контролы — это всего лишь абстракция внутри PCL части. Чтобы они смогли как-то отобразиться на устройстве, существуют так называемые рендеры. Располагаются они на следующей ступени иерархии в Xamarin Platform частях.  
 Под PCL частью у нас расположены Xamarin.iOS, Xamarin.Android и Xamarin.WinPhone. Это по сути и есть тот Xamarin, который уже существовал до Xamarin.Forms. Xamarin.iOS, Xamarin.Android и Xamarin.WinPhone содержат в себе C# обертки над нативными классами для каждой платформы. Так вот рендеры — это обертки над соответствующими визуальными компонентами, но которые дополнительно внутри себя содержат ссылки на PCL объекты, умеют читать у них выставленные свойства и применять их у себя.  
 В дальнейшем эти рендеры уже разворачиваются в нативные контролы, которые находятся в модулях, расположенных на нижних уровнях схемы.

Преимущества:

* использование в качестве языка C# и Mono;
* подход к созданию и работе с пользовательским интерфейсом близок WPF. Xamarin Forms поддерживает работу с XAML, биндинги, темплейты, стили и прочие концепции WPF. Однако они несколько урезаны по сравнению с WPF;
* из того, что Xamarin.Forms схож с WPF, вытекает следующий плюс этой платформы: MVVM(Model View ViewModel). Xamarin.Forms имеет XAML, визуальные элементы имеют BindingContext (аналог DataContext в WPF), есть BindableProperty (аналог DependencyProperty). Таким образом, можно связывать View с ViewModel аналогично тому, как в WPF;
* еще одно преимущество данной платформы в том, что, так как UI описывается только в одном месте, приложения под разными системами будут выглядеть очень похоже. Что может быть важно, например, в корпоративных разработках.

Недостатки:

* неполная реализация функционала WPF;
* различное поведение на разных платформах;
* производительность.

### XLabs

Xamarin Forms Labs (XLabs) - библиотека с открытым исходным кодом, целью которого является создание мощного и кросс-платформенного набора элементов управления, адаптированных для работы с Xamarin Forms.

### Prism

Данная библиотека предназначена для упрощения реализации паттерна MVVM в приложении для XAML-based платформ, таких как WPF, Silverlight, Windows Phone, или Windows Store, Xamarin.Forms. Также библиотека предоставляет другие сервисы, как Container, Navigation service т.д.

### MVVM

Паттерн MVVM (Model-View-ViewModel) позволяет отделить логику приложения от визуальной части (представления). Данный паттерн является архитектурным, то есть он задает общую архитектуру приложения.

Данный паттерн был представлен Джоном Госсманом (John Gossman) в 2005 году как модификация шаблона Presentation Model и был первоначально нацелен на разработку приложений в WPF. И хотя сейчас данный паттерн вышел за пределы WPF и применяется в самых различных технологиях, в том числе при разработке под Android, iOS, тем не менее WPF является довольно показательной технологией, которая раскрывает возможности данного паттерна.

MVVM состоит из трех компонентов (рисунок 2.4): модели (Model), модели представления (ViewModel) и представления (View).

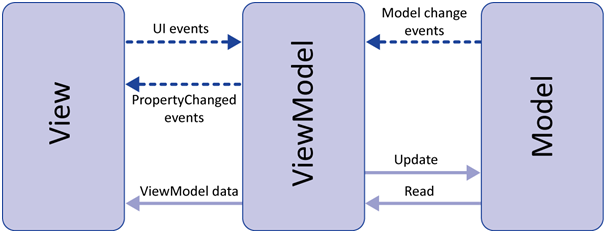


Рисунок 2.4 – схема паттерна MVVM.

Модель (Model) описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления.

Нередко модель реализует интерфейсы INotifyPropertyChanged или INotifyCollectionChanged, которые позволяют уведомлять систему об изменениях свойств модели. Благодаря этому облегчается привязка к представлению, хотя опять же прямое взаимодействие между моделью и представлением отсутствует.

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление - это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов.

Хотя визуальный элемент в Xamarin.Forms может содержать как интерфейс в xaml, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики, кроме разве что конструктора, который вызывает метод InitializeComponent и выполняет начальную инициализацию окна. Вся же основная логика приложения выносится в компонент ViewModel.

Однако иногда в файле связанного кода все может находиться некоторая логика, которую трудно реализовать в рамках паттерна MVVM во ViewModel.

Представление не обрабатывает события за редким исключением, а выполняет действия в основном посредством команд.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны.

ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели.

Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд.

Например, пользователь хочет сохранить введенные в текстовое поле данные. Он нажимает на кнопку и тем самым отправляет команду во ViewModel. А ViewModel уже получает переданные данные и в соответствии с ними обновляет модель.

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

### Dependensy injection. Inversion of control. Container.

Inversion of Control (инверсия управления) — это принцип, набор рекомендаций для написания слабо связанного кода. Суть которого в том, что каждый компонент системы должен быть как можно более изолированным от других, не полагаясь в своей работе на детали конкретной реализации других компонентов.  
 Dependency Injection (внедрение зависимостей) — это одна из реализаций этого принципа (помимо этого есть еще Factory Method, Service Locator). Согласно этому принципу процесс построения внешних зависимостей предоставляется программному компоненту.

IoC-контейнер — это библиотека, фреймворк, программа, которая позволяет упростить и автоматизировать написание кода с использованием данного подхода на столько, на сколько это возможно. IoC-контейнер внедряет зависимости.

### Веб-сервисы

Веб-служба (веб-сервис, англ. web-service) *–* идентифицируемая веб-адресом программная система со стандартизированными интерфейсами. В обиходе веб-сервисами называют услуги, оказываемые в Интернете. В этом употреблении речь идёт о поиске, веб-почте, хранении документов, файлов, закладок и т.п. Такими сервисами можно пользоваться независимо от компьютера, браузера или места доступа в Интернет. Веб-сервисы – это реализация чётко стандартизированных интерфейсов и протоколов обмена данными между различными приложениями, которые могут быть написаны на разных языках и находиться на совершенно разных узлах сети.

# **ПРОЕКТИРОВАНИЕ. РАЗРАБОТКА**

## **Структура приложения**

Данное приложение будет состоять из следующих модулей:

* CourceProject.MC;
* CourceProject.MC.UI;
* CourceProject.MC.DomainModel;
* CourceProject.MC.Common;
* CourceProject.MC.Resources;
* CourceProject.MC.Foundation;
* CourceProject.MC.Android;
* CourceProject.MC.IOS.

CourceProject.MC

Является стартовой точкой приложения. Содержит в себе начальную инициализацию приложения, регистрацию сервисов в контейнере зависимостей, регистрацию страниц в сервисе навигаций.

CourceProject.MC.UI

Содержит в себе часть, отвечающую за UI: страницы, связанные с ними ViewModel, конструктор элементов управления, собственные элементы управления, расширяющие базовые. Также содержит модуль, отвечающий за создание представления для опроса и привязки этого представления к соответствующим ViewModels.

CourceProject.MC.DomainModel

Содержит в себе доменную модель всей системы, в т.ч. связанные с опросом и аутентификацией пользователя.

CourceProject.MC.Common

Содержит константы, общие данные, используемые в остальных модулях (например, используемые регулярные выражения и т.д.), которые не имеют непосредственного отношения к предметной области данного приложенияя.

CourceProject.MC.Resources

Содержит общие ресурсы, используемые в остальных модулях, в виде строк, изображений (например, надписи приложения, иконки и т.д.).

CourceProject.MC.Foundation

Содержит интерфейсы и реализации сервисов, не связанных с UI, но содержащих основную логику приложения. Например, сервис, отвечающий за взаимодействие с сервером, получение от него опросов а также аутентификацию пользователя, сервис конфигурации и настроек приложения.

CourceProject.MC.Android

Содержит в себе классы, отвечающие за преобразование и рендер страниц и управляющих элементов под платформу Android.

CourceProject.MC.IOS

Содержит в себе классы, отвечающие за преобразование и рендер страниц и управляющих элементов под платформу IOS.

Диаграмма зависимостей модулей представлена на рисунке 3.1:

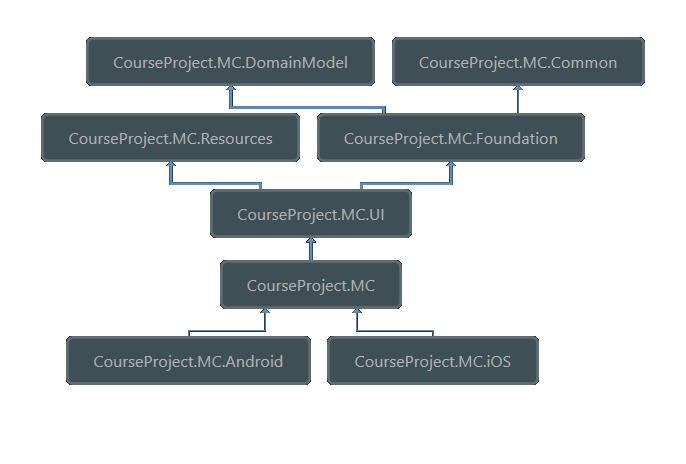


Рисунок 3.1 - Диаграмма зависимостей модулей.

## **Модуль CourceProject.MC**

1. Описание функциональности

Как было описано ранее, модуль CourceProject.MC является стартовой точкой приложения, который содержит в себе начальную инициализацию приложения, регистрацию сервисов в контейнере, регистрацию страниц в сервисе навигаций.

1. Inversion of Control Container в Prism

Значительным вкладом в расширяемость архитектуры приложения можно считать внедрение IoC-контейнера, для реализации паттерна Dependency Injection. С его помощью можно вынести управление зависимостями между различными классами в одно место – это и есть контейнер. Регистрация зависимостей производится на старте приложения. Среди активно используемых способов можно выделить внедрение зависимостей между интерфейсами и реализациями, посредством вызова метода RegisterType, например:

Container.RegusterType<IJsonSerializer, JsonSerializer>(new ContainerControlledLifetimeManager());

Таким образом, можно передавать в конструкторы объектов только интерфейсы, а контейнер будет самостоятельно определять зависимости и передавать объекты с реализацией необходимого интерфейса. Кроме того, контейнер может манипулировать временем жизни объекта, что оптимизирует расходование ресурсов.

В рамках данного курсового проекта будет использоваться UnityContainer, поскольку для библиотеки Prism он подходит лучше всего.

1. Navigation service в Xamarin и Prism

Xamarin Forms предоставляет инструменты для навигации в виде свойства Navigation класса Page. Это свойство представляет интерфейс INavigation, в котором есть следующие методы:

Task PushAsync(Page page)

Task PushModalAsync(Page page)

В качестве параметра здесь передается объект Page - страница, на которую надо осуществить переход.

Второй метод имеет в своем названии слово "Modal" и осуществляет переход на модальную страницу. Обычно модальные страницы используются, когда приложению нужно получить некоторую информацию от пользователя. При этом до получения информации нельзя возвращаться на предыдущую страницу.

Также имеются методы для возврата на предыдущую страницу:

Task<Page> PopAsync()

Task<Page> PopModalAsync()

Исходя из вышеописанного, можно сделать вывод, что организация навигации осуществляется по принципу стека LIFO (Last in – first out).

Отсюда вытекают следующие преимущества:

* для навигации не нужно запоминать полный адрес страницы.

Недостатки:

* для навигации на несколько уровней назад нужно последовательно сделать несколько операций Pop, усложняется логика работы навигации страницы, увеличивается связанность страниц.

Навигация в приложении Prism концептуально отличается от стандартной навигации в Xamarin.Forms. В то время как навигационная система Xamarin.Forms полагается на экземпляр класса страницы для навигации, Prism удаляет все зависимости от типов страниц для достижения слабосвязанной навигации из ViewModel. В Prism концепция навигации к View или навигации к ViewModel не существует. Вместо этого переход осуществляется идентификаторам страниц. В стандартном случае используется система идентификации согласно URI.

Преимущества такого подхода:

* Меньшая связанность страниц, позволяет организовывать быструю навигацию между страницами, т.к. каждая страница независима.

Недостатки данного подхода:

* Для навигации используется URI, которые в некоторых случаях могут иметь сложную структуру, отсутствует проверка валидности URI.

Навигация по страницам в Prism осуществляется при помощи службы INavigationService.

Регистрация страницы NaviagtionService в Prism происходит в App.Xaml.cs в методе RegisterTypes при помощи метода класса Container RegisterTypeForNavigation<T>.

Для последующего использования сервиса он передается в конструктор ViewModel или xaml.cs файл Containerом. Навигация происходит при помощи метода NavigateAsync(URI uri).

Также имеется возможность возврата на предыдущую страницу при помощи метода GoBackAsync.

Также удобной возможностью NavigationService в Prism является возможность передачи параметров на страницу при навигации через NavigationParameters, который представляет собой словарь, содержащий элементы типа: ключ – значение.

Таким образом инструменты навигации в Prism позволяют создавать более гибкие системы и взаимосвязи между страницами.

1. Привязка страниц к ViewModel в XamarinForms и Prism

Xamarin.Forms Привязка данных (data binding) является одним из ключевых моментов платформы Xamarin Forms.

Привязка данных состоит из двух компонентов: источника (source) и цели (target). Привязка осуществляется от свойства источника к свойству цели. И когда происходит изменение источника, механизм привязки автоматически обновляет также и цель.

Цель привязки должна представлять объект BindableObject, а свойство, к которому осуществляется привязка, должно быть свойством BindableProperty. Поскольку большинство визуальных элементов в Xamarin Forms наследуются от класса BindableObject, то в качестве цели привязки будут, как правило, выступать визуальные элементы.

А вот источником привязки может выступать любой объект языка C#. Однако, надо понимать, что цель привязки должна автоматически изменяться при изменении источника, поэтому нам нужно извещать систему о изменении свойств источника привязки. В Xamarin, да и вообще на платформе .NET, в качестве подобного механизма извещения выступает интерфейс INotifyPropertyChanged. То есть нужно реализовать данный интерфейс в объекте-источнике.

Объект BindableObject как раз реализует INotifyPropertyChanged. Поэтому если источником привязки является стандартный визуальный элемент из Xamarin Forms, то автоматически будет изменяться и цель привязки. Но если в качестве источника выступает не BindableObject, а какой-нибудь объект простого класса C#, то, как писалось выше, этот класс должен реализовать INotifyPropertyChanged.

Для установки привязки у объекта цели устанавливается свойство BindingContext. В качестве значения оно принимает источник привязки.

Привязку можно устанавливать как в XAML, так и cs файлах.

Однако Prism предоставляет более удобные инструменты для привязки, основанные на механизмах привязки в Xaramin. Также в Prism имеется возможность привязки не только к свойствам, но и к событиям при помощи механизма команд.

В основном последовательность привязки страницы к ViewModel в Prism состоит в следующем:

1. Создание страницы (xaml + xaml.cs файлы);
2. Создание соответствующей данной страницы ViewModel;
3. Регистрация в классе ViewModelLocationProvider зависимости при помощи метода Register<Page, ViewModel>. Данное действие предпочтительно производить в методе RegisterTypes класса App файла App.xaml.cs;
4. Далее добавить свойство: prism:ViewModelLocator.AutowireViewModel="True" к тэгу страницы в xaml файле.

После выполнения данных действий можно устанавливать привязку как в xaml файле, например Text=”{Binding TextPropertyViewModel, Mode=OneWayToSource}”, так и в xaml.cs файле при помощи метода SetBinding, например SetBinding(TextProperty, nameof(ViewModel. TextPropertyInViewModel), Mode = TwoWay).

В качестве Mode может выступать одно из значений перечисления BindingMode:

* OneWay. Указывает, что изменения должны распространяться только от источника (обычно View Model) до целевого объекта (BindableObject). Это режим по умолчанию для большинства значений BindableProperty;
* OneWayToSource. Указывает, что изменения должны распространяться только от целевого объекта (BindableObject) к источнику (обычно это View Model);
* TwoWay. Изменения распространяются в оба направления;
* Default. Значение по умолчанию, обычно равно OneWay.

Также Xamarin.Forms и Prism имеют в наличии механизмы команд.

Механизм команд позволяет избежать описания обработчиков событий в классе визуального компонента, таким образом достигается большее соответствие паттерну MVVM. Т.е. логика обработчиков выносится в ViewModel.

Т.о. механизм привязки позволяет реализовать паттерн MVVM. Т.о. отделяется представление от логики и данных, что обеспечивает большую взаимозаменяемость модулей и частей приложения, а также увеличивает повторную используемость кода.

В рамках данной курсовой работы будут использованы инструменты, предоставляемые библиотекой Prism.

## **CourceProject.MC.DomainModel**

1. Описание функциолаьности

Модуль CourceProject.MC.DomainModelв себе модели, связанные с опросом и аутентификацией пользователя.

1. Модель опроса

Данная модель представлена классом Survey. Опрос содержит в себе Id опроса, заголовок, Id создателя, время создания, время последней модификации, анонимный ли опрос, случайный ли порядок вопросов, показывается ли количество вопросов, показывать ли прогресс выполнения теста, помечать ли обязательные поля, и коллекцию страниц (рисунок 3.1).

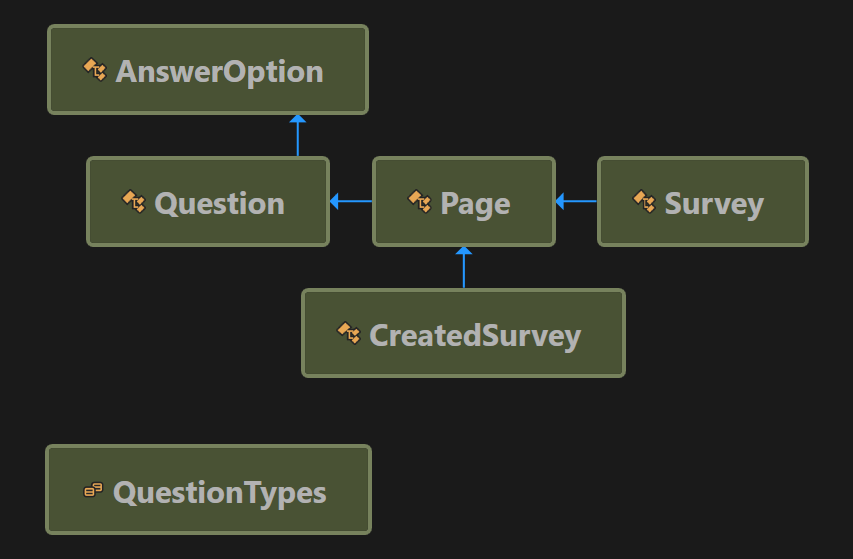


Рисунок 3.1 – Диаграмма модели опроса.

В свою очередь модель страницы представлена классом Page. Модель содержит в себе Id страницы, заголовок страницы и коллекцию вопросов.

Модель вопроса представлена классом Question. Модель содержит в себе Id вопроса, его тип, обязателен ли вопрос и коллекцию ответов. Тип представлен индексом, подробнее обозначение индексов содержится в перечислении QuestionTypes.

Типы вопроса зависят от вида управляющего элемента, предназначенного для ответа. Поддерживаются следующие типы:

* SingleLine(однострочное поле для ввода);
* MultiLine(многострочное поле для ввода);
* Date(выбор даты);
* RadioButtonGroup(связанная группа RadioButtonов);
* CheckBoxesGroup(связанная группа CheckBoxов).

Модель ответа представлена классом AnswerOption. Модель содержит в себе Id ответа и текстовое представление ответа.

## **Модуль CourceProject.MC.UI**

1. Описание функциональности

Модуль CourceProject.MC.UI в себе часть, отвечающую за пользовательский интерфейс: страницы, связанные с ними ViewModel, конструктор элементов управления, собственные элементы управления, расширяющие базовые. Также содержит модуль, отвечающий за создание представления для опроса и привязки этого представления к соответствующим ViewModels.

1. Описание страниц

Данная версия приложения содержит в себе 5 страниц: страница загрузки (класс SplashPage – рисунок 3.2), страница аутентификации, представленная классом SingUpPage (рисунок 3.3), домашняя страница пользователя, представленная классом HomePage (рисунок 3.4), страница получения опроса, представленная классом GetSurveyPage (рисунок 3.5) и страница отображения опроса, представленная классом SelectedSurveyPage (рисунок 3.6). Также каждая страница имеет связанная с ней ViewModel: InitialViewModel, SignUpViewModel, HomeViewModel, GetSurveyViewModel, SelectedSurveyViewModel соответственно.

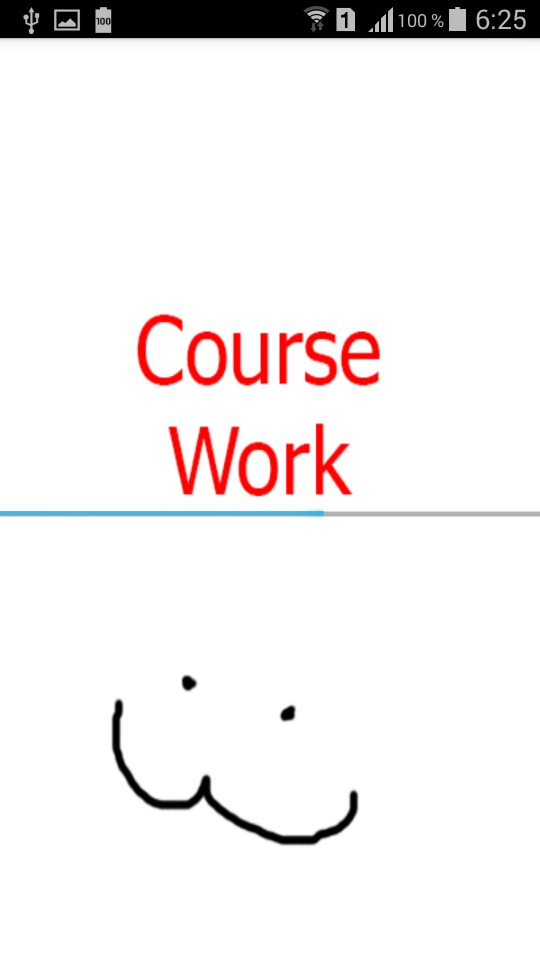
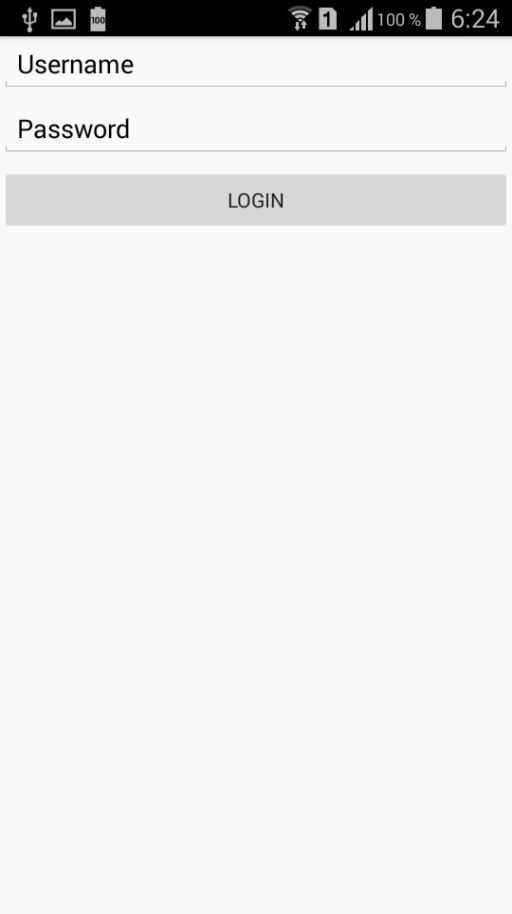
Рисунок 3.2 – Страница загрузки для Android.

Рисунок 3.3 – Страница аутентификации для Android

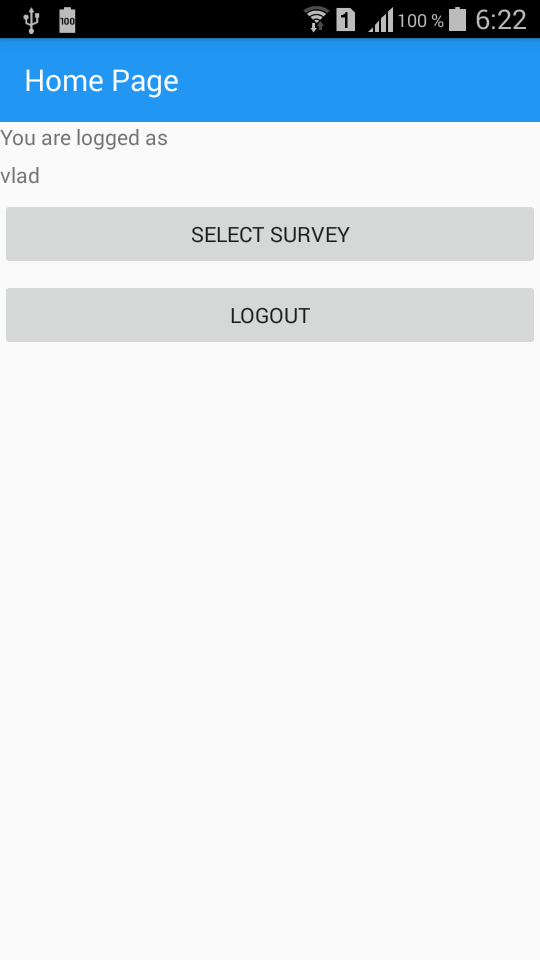
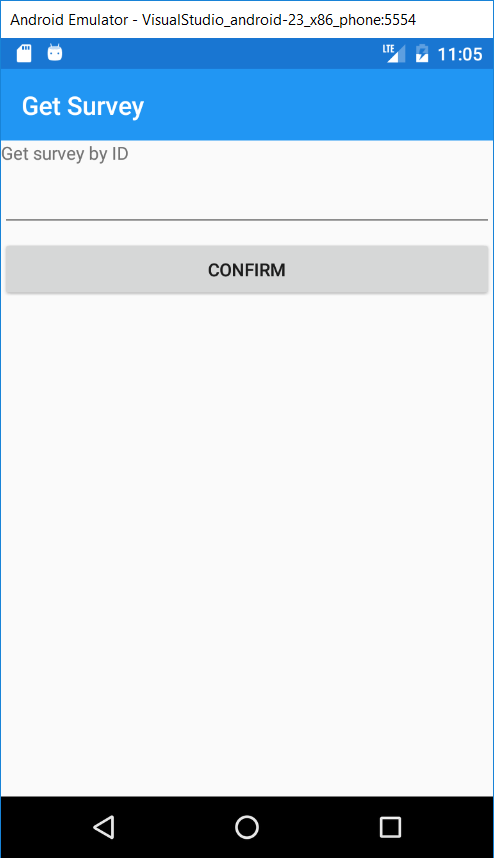
Рисунок 3.4 – домашняя страница пользователя для Android

Рисунок 3.5 – Страница получения опроса для Android.

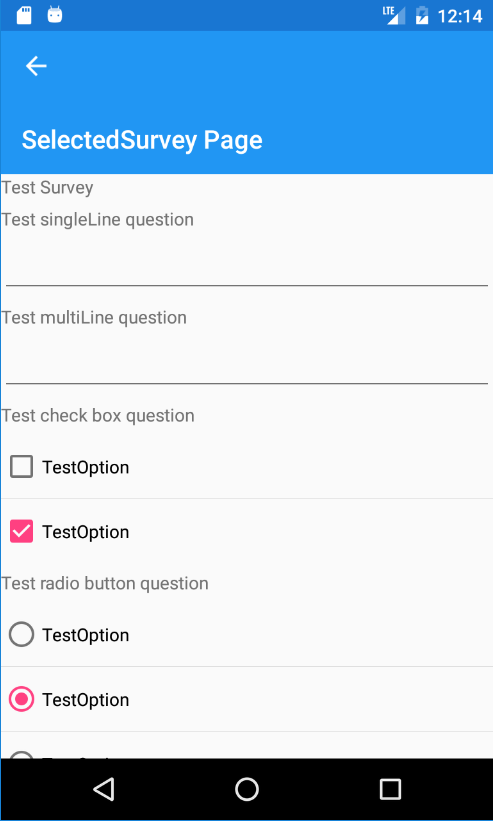


Рисунок 3.6 – Страница опроса для Android.

Далее будут подробно рассмотрены реализации данных страниц.

1. SplashPage

Данная страница загружается первой. На ней имеется показатель

прогресса, который постепенно заполняется, когда на соответствующей ViewModel проверяется соединение с сервером и аутентификация пользователя. Затем, в зависимости от полученных при совершённых действиях результатов происходит перенаправление на страницу аутентификации или домашнюю страницу.

1. SignUpPage

На данной странице пользователь может авторизоваться под необходимыми данными, для входа на домашнюю страницу и получения возможности проходить опрос.

1. HomePage

Попадая на домашнюю страницу, пользователь может увидеть некоторую информацию о себе (на текущий момент логин), а также получает возможность попасть на страницу получения опроса, для его прохождения в дальнейшем.

1. GetSurveyPage

Данная страница (рисунок 3.3) предназначена для получения опроса от сервера по Id данного опроса. В последующих версиях планируется использование Push-нотификаций для предоставления опросов пользователям.

Данная страница содержит поле для ввода Id, кнопку, по нажатию на которую происходит проверка правильности формата введенного Id и, при корректном Id происходит обращение к серверу для получения опроса с соответствующим Id. Если формат введенного Id является некорректным, то пользователю выводится соответствующее сообщение (рисунок 3.5). При неудачной попытке получения опроса от сервера выводится соответствующее сообщение (рисунок 3.7). При успешном получении опроса идёт переход на страницу опроса (рисунок 3.6).

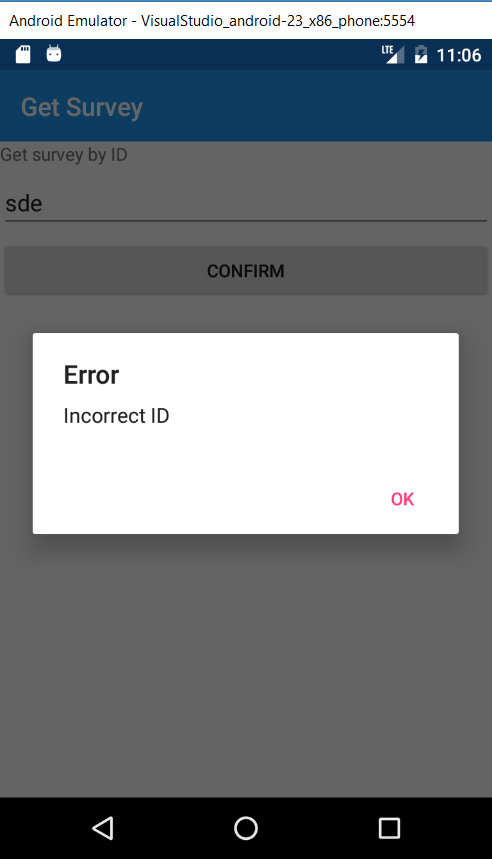


Рисунок 3.7 – Ошибка: “Некорректный Id”, Android.

1. SelectedSurveyPage

Данная страница (рисунок 3.6) предназначена для отображения опроса пользователю с целью дальнейшего прохождения.

Данная страница имеет преставление опроса, сконструированное согласно пришедшему от сервера опросу, также данная страница имеет кнопку для отправки пройденного опроса. При попытке отправки невалидного ответа на опрос будут подсвечены соответствующие невалидные ответы (рисунок 3.17).

1. Конструктор представления опроса

При получении опроса со страницы GetSurvey необходимо, исходя из полученного опроса, построить его представление, также необходимо организовать изменение модели при вводе информации пользователем в сгенерированные управляющие элементы.

Для решения данной задачи был разработан следующий механизм (рисунки 3.7, 3.8, 3.9, 3.10):

Сначала достаются все вопросы из модели опроса, далее для каждой модели вопроса создаётся своё представление. При построении представления для каждого управляющего элемента создаётся своя ViewModel, которая содержит в себе модель вопроса и связывается с соответствующим управляющим элементом. Далее идёт связывание полей ввода с соответствующими свойствами ViewModel, при обращении к которым идёт взаимодействие с моделью вопроса, ссылка на которую хранится в ViewModel.

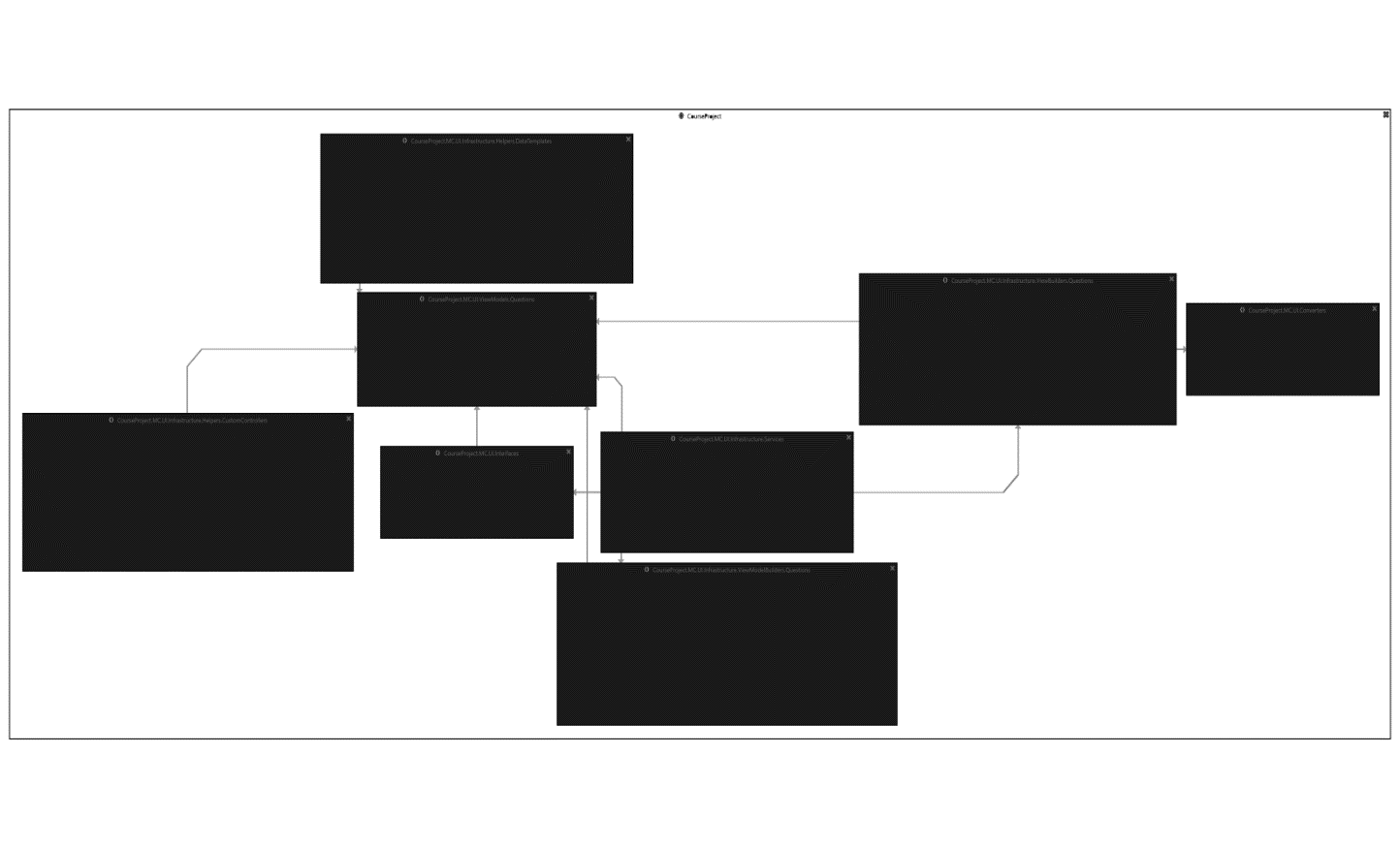


Рисунок 3.8 – Схема взаимосвязи элементов механизма формирования представления опроса.

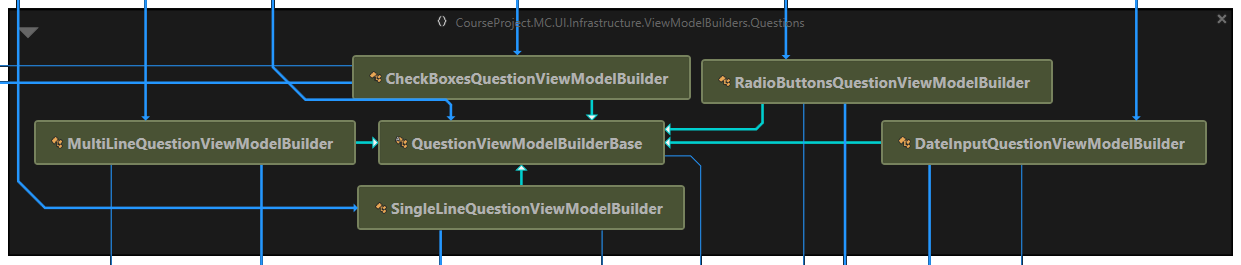


Рисунок 3.9 – Схема взаимосвязи элементов механизма формирования представления опроса. Иерархия ViewModelBuilders.

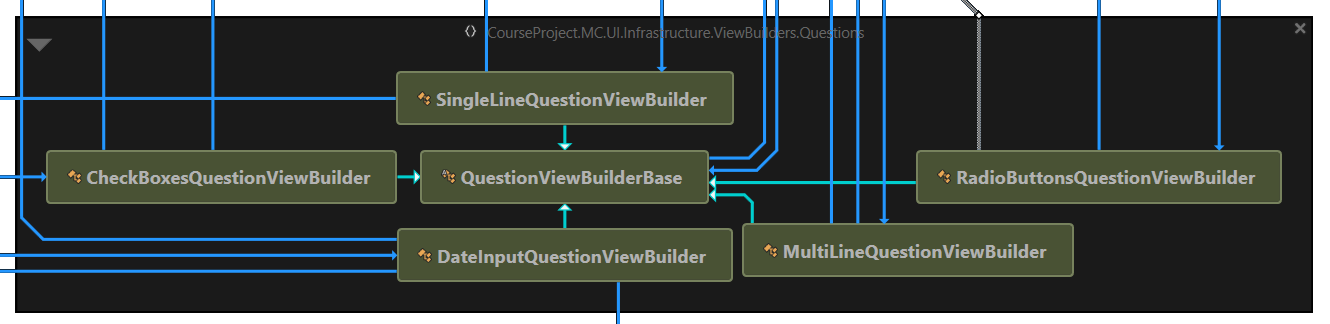


Рисунок 3.10 – Схема взаимосвязи элементов механизма формирования представления опроса. Иерархия ViewBuilders.

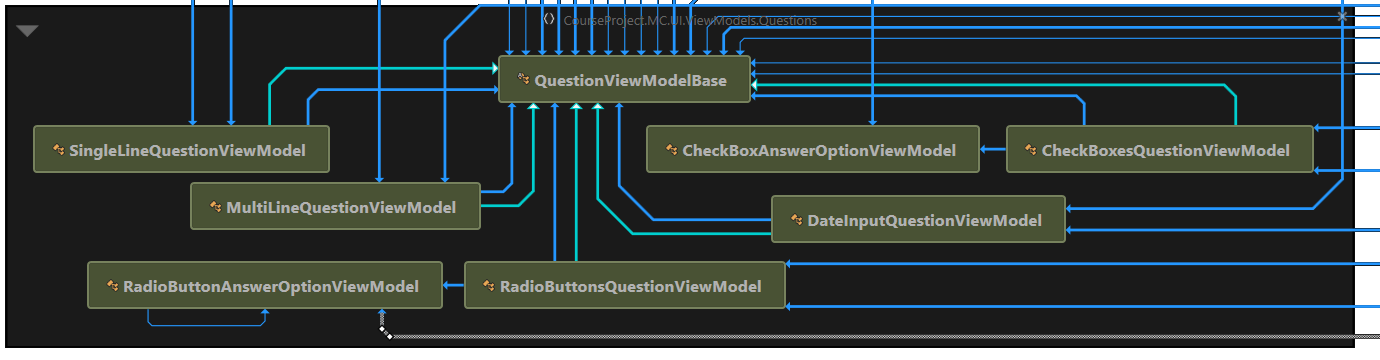


Рисунок 3.11 – Схема взаимосвязи элементов механизма формирования представления опроса. Иерархия ViewModels.

Построение представления для вопроса происходит при помощи класса QuestionViewBuilderService, который реализует интерфейс IQuestionViewBuilderService. Данный класс представляет собой реализацию паттерна “фабрика”.

Создание ViewModel происходит при помощи класса QuestionViewModelBuilder, который реализует интерфейс IQuestionViewModelBuilder. Данный класс представляет собой реализацию паттерна “фабрика”.

Далее будут рассмотрены принципы создания представления и ViewModel для каждого типа вопросов.

1. Базовые классы QuestionViewBuilderBase и QuestionViewModelBase

Класс QuestionViewBuilderBase представляет собой класс, который является базовым для всех ViewBuilders для вопросов. Данный класс содержит метод создания представления, куда передаётся ViewModel, к которой будет происходить привязка.

Класс QuestionViewModelBase представляет собой класс, который является базовым для всех Models для вопросов. Данный класс, содержит себе модель вопроса, который пришёл от сервера. Также он содержит свойство ответа, через которое будет изменяться модель ответа, которая расположена в модели вопроса. Также он содержит остальные свойства, через которые будет осуществляться взаимодействие ViewModel и соответствующего управляющего элемента (например, цвет надписи для вопроса и т.д.).

1. SingleLine

В качестве представления был выбран стандартный элемент управления Entry (рисунок 3.12) из библиотеки Xamarin.Forms.

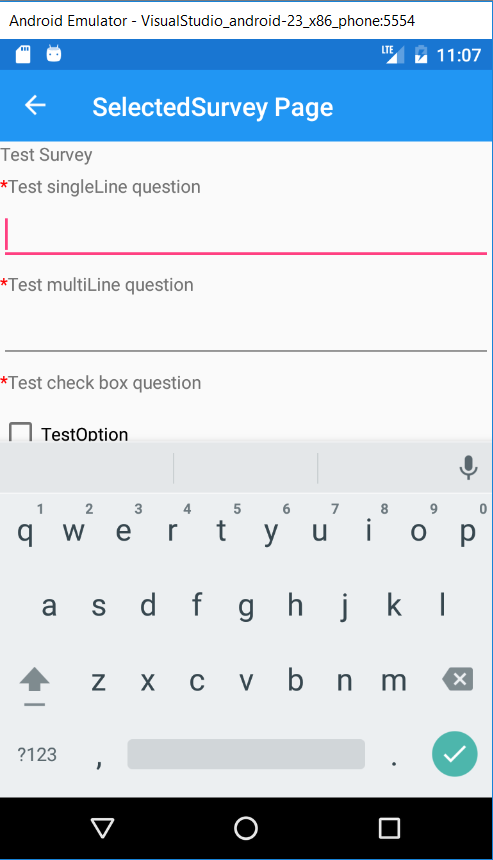


Рисунок 3.12 – Визуальное представление SingleLine. Android.

Логика привязки следующая:

Т.к. по сути, в данном случае фигурирует один управляющий элемент, то свойство Text элемента Entry будет привязываться к свойству Answer ViewModel. В свою очередь оно будет обращаться к нулевому элементу коллекции AnswerOptions, которая содержится в модели вопроса.

1. MultiLine

В качестве представления был выбран стандартный элемент управления Editor(рисунок 3.13) из библиотеки Xamarin.Forms.

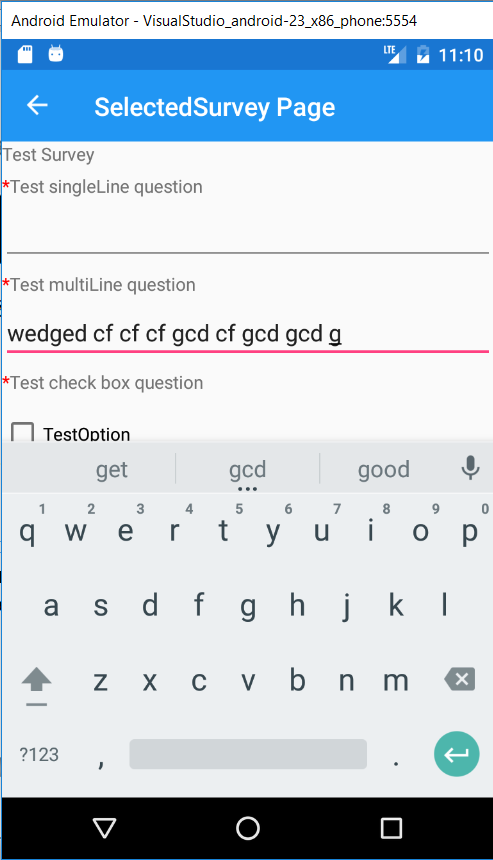


Рисунок 3.13 – Визуальное представление MultiLine. Android.

Логика привязки следующая:

Т.к. по сути, в данном случае фигурирует один управляющий элемент, то свойство Text элемента Editor будет привязываться к свойству Answer ViewModel. В свою очередь оно будет обращаться к нулевому элементу коллекции AnswerOptions, которая содержится в модели вопроса.

1. CheckBoxGroup

В качестве представления был выбран элемент CheckBox из библиотеки Xlabs, т.к. в стандартной библиотеки нету реализации CheckBox (рисунок 3.14).

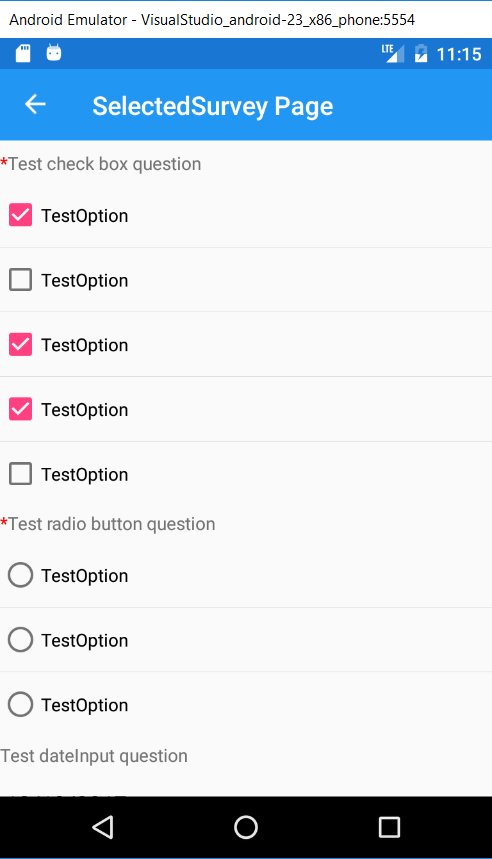


Рисунок 3.14 – Визуальное представление CheckBoxGroup. Android.

Логика привязки следующая:

Вопрос представляет собой общее описание вопроса и несколько опций ответов. Соответственно представлением данного типа вопросов будет коллекция CheckBoxов, каждый из которых будет привязан к соответствующей ему ViewModel, которая содержит в себе модель AnswerOption. Ответ будет представлять собой текстовое представление переменной логического типа: true или false, что будет обозначать отмечена ли вариант ответа или нет.

В качестве ViewModel, соответствующая модели вопроса, представлен класс CheckBoxesViewModel, которая содержит в себе коллекцию CheckBoxViewModel, которая соответствует модели AnswerOption.

В качестве визуального элемента, содержащего в себе коллекцию CheckBoxов выступает стандартный элемент ListView из пространства имён Xamarin.Forms. В качестве ItemTemplate выступает CheckBoxDataTemplate, который представляет собой визуальный элемент Cell, который содержит в себе CheckBox.

1. RadioButtonGroup

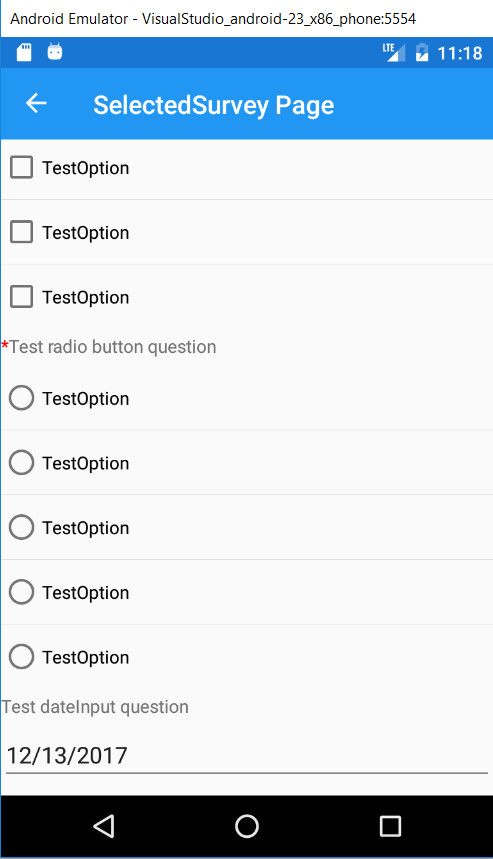
В качестве представления был выбран элемент CustomRadioButton из библиотеки Xlabs, т.к. в стандартной библиотеки нету реализации RadioButton (рисунок 3.15).

Рисунок 3.15 – Визуальное представление RadioButtonGroup. Android.

Логика привязки следующая:

Вопрос представляет собой общее описание вопроса и несколько опций ответов. Соответственно представлением данного типа вопросов будет коллекция RadioButtonов, каждый из которых будет привязан к соответствующей ему ViewModel, которая содержит в себе модель AnswerOption. Ответ будет представлять собой текстовое представление переменной логического типа: true или false, что будет обозначать отмечена ли вариант ответа или нет.

В качестве ViewModel, соответствующая модели вопроса, представлен класс RadioButtonsViewModel, которая содержит в себе коллекцию RadioButtonViewModel, которая соответствует модели AnswerOption.

В качестве визуального элемента, содержащего в себе коллекцию RadioButtonов выступает стандартный элемент ListView из пространства имён Xamarin.Forms. В качестве ItemTemplate выступает RadioButtonDataTemplate, который представляет собой визуальный элемент Cell, который содержит в себе CustomRadioButton.

При изменении значения RadioButtона происходит перерасчёт значения других RadioButtonов, т.к. в группе RadioButtonов значение “отмечен” может одновременно иметь только один элемент.

1. DateInput

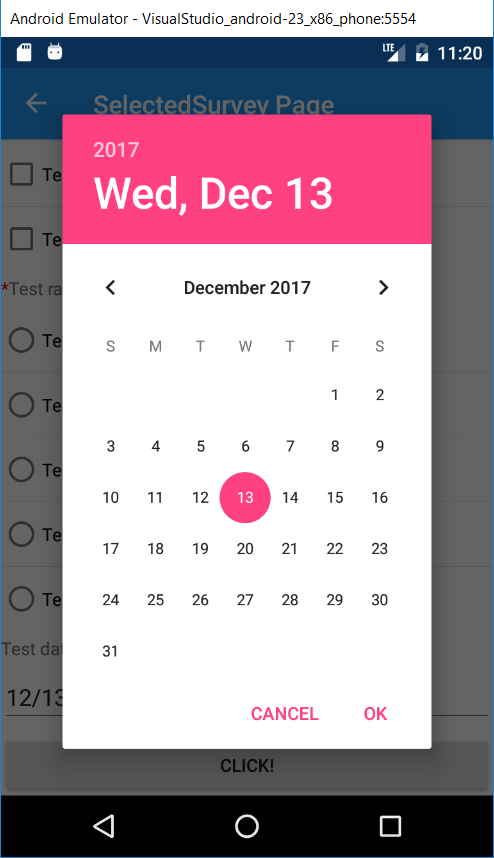
В качестве представления был выбран стандартный элемент управления DatePicker (рисунок 3.16) из библиотеки Xamarin.Forms. Логика привязки следующая:

Рисунок 3.16 – Визуальное представление DateInput. Android.

Т.к. по сути, в данном случае фигурирует один управляющий элемент, то свойство Date элемента DatePicker будет привязываться к свойству Answer ViewModel. В свою очередь оно будет обращаться к нулевому элементу коллекции AnswerOptions, которая содержится в модели вопроса.

1. Custom controllers

Для реализации дополнительной логики взаимодействия с пользователем были разработаны следующие элементы управления:

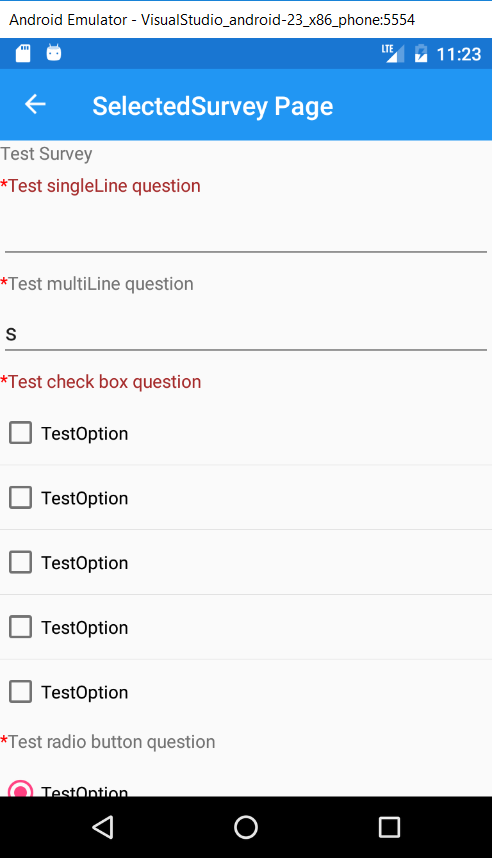
* QuestionDescriptionLabel. Наследуется от класса Label из пространства имён Xamarin.Forms, в качестве дополнительного Bindable свойства имеет свойство цвета текста вопроса. В дальнейшем это свойство будет использоваться в том случае, когда пользователю нужно будет подсветить те вопросы, которые обязательны и которые он оставил пустыми (рисунок 3.17). Также при формировании данного элемента учитывается, обязательный ли вопрос, и в зависимости от этого помечает его или нет.

Рисунок 3.17 – Подсветка обязательных неотвеченных вопросов. Android.

1. Валидация ответов перед отправкой

Перед тем, как отправить данные опроса на сервер, нужно проверить, на все ли обязательные вопросы ответил пользователь. Для этого нужно определить критерии для каждого типа вопроса, по которым будет определяться, является ли ответ валидным:

* Для SingleLine ответ будет являться валидным, если он не пустой;
* Для MultiLine ответ будет являться валидным, если он не пустой;
* Для CheckBoxGroup ответ будет являться валидным, если хотя бы одна опция выбрана;
* Для RadioButtonGroup ответ будет являться валидным, если хотя бы одна опция выбрана.

Подход решения данной задачи:

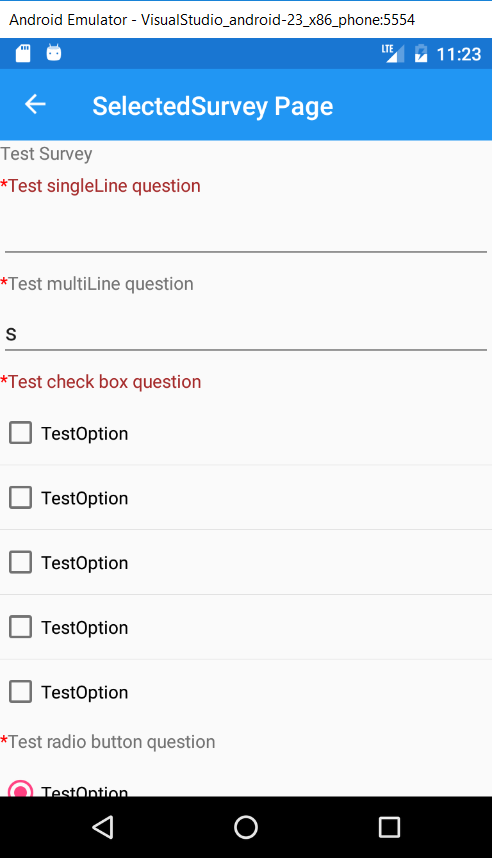
Перед отправлением запроса анализируется коллекция, содержащая в себе ViewModel вопросов. Если вопрос обязательный и ответ является не валидным, согласно типу вопроса и выше описанных правил, то операция отправление запроса отклоняется, а соответствующие невалидные вопросы подсвечиваются для пользователя (рисунок 3.18).

Рисунок 3.18 – подсветка невалидных ответов. Android.

## **Модуль CourceProject.MC.Common**

1. Описание функциональности

Содержит константы, общие данные, используемые в остальных модулях (например, используемые регулярные выражения, атрибуты и т.д.).

1. Регулярные выражения

Регулярные выражения - формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. Для поиска используется строка-образец, состоящая из символов и метасимволов и задающая правило поиска. Для манипуляций с текстом дополнительно задаётся строка замены, которая также может содержать в себе специальные символы.

1. Атрибуты

Атрибуты представляют специальные инструменты, которые позволяют встраивать в сборку дополнительные метаданные. Атрибуты могут применяться как ко всему типу (классу, интерфейсу и т.д.), так и к отдельным его частям (методу, свойству и т.д.). Основу атрибутов составляет класс System.Attribute, от которого образованы все остальные классы атрибутов.

## **Модуль CourceProject.MC.Resources**

1. Описание функциональности

Содержит общие ресурсы, используемые в остальных модулях (например, надписи приложения, иконки и т.д.).

## **Модуль CourceProject.MC.Android**

1. Описание функциональности

Содержит в себе классы, отвечающие за преобразование и рендер страниц и управляющих элементов под платформу Andoid.

1. Структура модуля

Данный модуль содержит в себе стартовую точку сборки версии приложения для Android, ресурсы, необходимые для сборки данной версии (иконки, стили, форматирование согласно версии Android, которая является целевой для сборки). Также сборка содержит в себе файлы конфигурации.

В качестве целевой платформы используется опция “Использовать последнюю версию” (на данный момент таковой является Android версия 8.0 , кодовое имя Oreo, уровень API 26).

1. Класс MainActivity

Данный класс является стартовой точкой сборки версии приложения для Android. В данном классе идёт регистрация ресурсов, инициализация компонентов и далее на основе класса App из сборки CourseProject.MC идёт загрузка приложения.

Также в модуль включены библиотеки XLabs, Prism для корректной загрузки приложения, которое использует компоненты из данных модулей.

## **Модуль CourceProject.MC.iOS**

1. Описание функциональности

Содержит в себе классы, отвечающие за преобразование и рендер страниц и управляющих элементов под платформу iOS.

1. Структура модуля

Данный модуль содержит в себе стартовую точку сборки версии приложения для iOS, ресурсы, необходимые для сборки данной версии (иконки, стили, форматирование согласно версии iOS, которая является целевой для сборки). Также сборка содержит в себе файлы конфигурации.

В качестве целевой платформы используется опция “По умолчанию”.

1. Стартовая точка приложения. Класс Main. Класс AppDelegate

В качестве стартовой точки сборки версии приложения для iOS используется класс Main, в котором далее идёт загрузка делегата приложения AppGelegate.

В классе AppGelegate идёт регистрация ресурсов, инициализация компонентов и далее на основе класса App из сборки CourseProject.MC идёт загрузка приложения.

Также в модуль включены библиотеки XLabs, Prism для корректной загрузки приложения, которое использует компоненты из данных модулей.

## **Модуль CourceProject.MC.Foundation**

1. Описание функциональности

Содержит интерфейсы и реализации сервисов, не связанных с UI, но содержащих основную логику приложения. Например, сервис, отвечающий за взаимодействие с сервером, получение от него опросов а также аутентификацию пользователя, сервис конфигурации и настроек приложения.

1. SurveyWebService

Данный сервис предназначен для взаимодействия с сервером, в частности для получения от него опроса по Id. Также данный сервис обеспечивает отправку пройденного опроса на сервер.

1. SurveysManager

Данный сервис предназначен для взаимодействия с моделью опроса, в частности для получения списка моделей вопросов из модели опроса.

1. SettingsService

Данный сервис обеспечивает хранение каких-либо параметров конфигурации приложения, добавление конфигураций, а также получения значения параметра по ключу.

Параметр конфигурации имеет вид: “Ключ” : “Значение”.

1. NetworkManager

Данный сервис обеспечивает взаимодействие, связанное с сервером и подключением к сети интернет, в частности проверка подключения к сети интернет, получения эксемпляра класса, предназначенного для отправки HTTP запросов.

1. AuthenticationWebService

Данный сервис обеспечивает взаимодействие, связанное с сервером и аутентификацией, в частности аутентификацию при помощи логина и пароля или при помощи токенов.

# **ПРОВЕРКА КОРРЕКТНОСТИ РАБОТЫ ПРОГРАММНОГО СРЕДСТВА**

Для проверки корректности работы приложения была проведён процесс тестирования, результаты которого представлены в таблице 4.1.

Таблица 4.1 – Тестирование работы мобильного клиента

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестируемая функция | Ожидаемый результат | Полученный результат |
| 1. | Попытка отправления опроса при пустом поле обязательного вопроса типа SingleLine. | Отправка отклонена. Соответствующий вопрос подсвечен. | Отправка отклонена. Соответствующий вопрос подсвечен. |
| 2. | Попытка отправления опроса при пустом поле обязательного вопроса типа MultiLine. | Отправка отклонена. Соответствующий вопрос подсвечен. | Отправка отклонена. Соответствующий вопрос подсвечен. |
| 3. | Попытка отправления опроса при ни одном неотмеченным вариантом ответа обязательного вопроса типа RadioButtonGroup. | Отправка отклонена. Соответствующий вопрос подсвечен. | Отправка отклонена. Соответствующий вопрос подсвечен. |
| 4. | Попытка отправления опроса при при ниодном неотмеченным вариантом ответа обязательного вопроса типа CheckBoxGroup. | Отправка отклонена. Соответствующий вопрос подсвечен. | Отправка отклонена. Соответствующий вопрос подсвечен. |
| 5. | Попытка получения опроса при пустом поле для ввода Id | Вывод сообщения о некорректности Id. | Вывод сообщения о некорректности Id. |
| 6. | Попытка получения опроса при невалидном значении в поле для ввода Id | Вывод сообщения о некорректности Id. | Вывод сообщения о некорректности Id. |
| 7. | Попытка получения опроса без доступа в интернет | Вывод сообщения о невозможности получения опроса. | Вывод сообщения о невозможности получения опроса. |
| 8 | Попытка отправки валидных ответов на опрос. | Перенаправление на главную страницу. Успешная отправка ответов на опрос. | Перенаправление на главную страницу. Успешная отправка ответов на опрос. |

Таблица 4.1 – Тестирование работы мобильного клиента - продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | Попытка успешной аутентификации | Перенаправление на главную страницу. | Перенаправление на главную страницу. |
| 10 | Попытка неудачной аутентификации | Вывод соответствующего сообщения. | Вывод соответствующего сообщения. |
| 11 | Нажатие кнопки получения опроса на главной странице | Перенаправление на страницу получения опроса. | Перенаправление на страницу получения опроса. |
| 12 | Вывод сообщения о пустом поле для логина. | Вывод сообщения о пустом поле для логина. | Вывод сообщения о пустом поле для логина. |
| 13 | Вывод сообщения о пустом поле для пароля. | Вывод сообщения о пустом поле для пароля. | Вывод сообщения о пустом поле для пароля. |
| 14 | Попытка успешной аутентификации | Перенаправление на главную страницу. | Перенаправление на главную страницу. |
| 15 | Ввод в поле для ответа на вопрос типа SingleLine | Срабатывание изменения соответствующего свойства ViewModel. Изменение соответствующего поля модели вопроса. | Срабатывание изменения соответствующего свойства ViewModel. Изменение соответствующего поля модели вопроса. |

Таблица 4.1 – Тестирование работы мобильного клиента - продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| 16 | Ввод в поле для ответа на вопрос типа MultiLine. | Срабатывание изменения соответствующего свойства ViewModel. Изменение соответствующего поля модели вопроса. | Срабатывание изменения соответствующего свойства ViewModel. Изменение соответствующего поля модели вопроса. |
| 17 | Выбор даты в вопросе типа DateInput. | Срабатывание изменения соответствующего свойства ViewModel. Изменение соответствующего поля модели вопроса. | Срабатывание изменения соответствующего свойства ViewModel. Изменение соответствующего поля модели вопроса. |
| 18 | Клик на CheckBox, соответствующий вопросу типа CheckBoxGroup. | Срабатывание изменения соответствующего свойства ViewModel. Изменение соответствующего поля модели вопроса. | Срабатывание изменения соответствующего свойства ViewModel. Изменение соответствующего поля модели вопроса. |
| 19 | Клик на RadioButton, соответствующий вопросу типа RadioButtonGroup. | Срабатывание изменения соответствующего свойства ViewModel. Изменение соответствующего поля модели вопроса. | Срабатывание изменения соответствующего свойства ViewModel. Изменение соответствующего поля модели вопроса. |

Данная версия клиента успешно прошла тестирование с использованием вышеописанных тест-кейсов. Таким образом, Данная версия клиента является работоспособной и реализует функционал, описанный в п.1.3.

# **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

1. **Системные требования**

Для успешного запуска приложения необходимо иметь:

Для Android: версия Android 4.0.3 и выше.

Также необходимо иметь доступ к интернету.

1. **Запуск программы.**

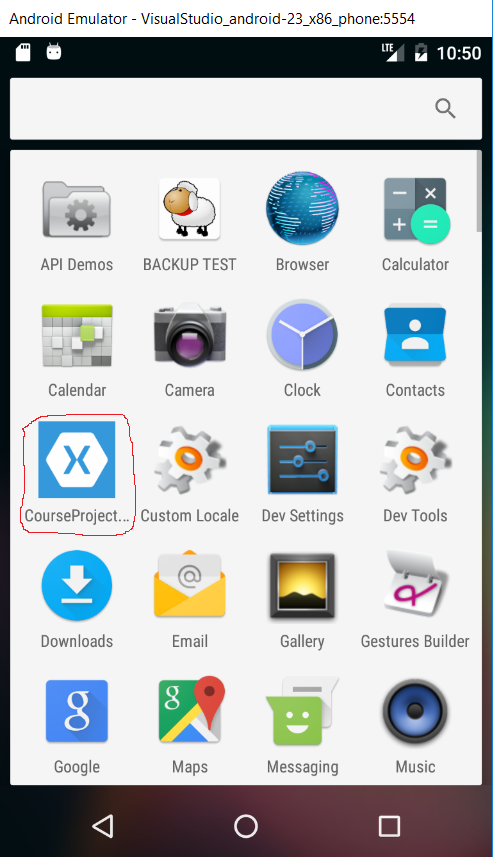
****Для запуска программы необходимо открыть CourceProject.MC (рисунок 5.1) откроется начальное окно программы: страница для аутентификации (рисунок 3.3) или домашняя страница(рисунок 3.4).

Рисунок 5.1 – иконка приложения. Android.

1. **Авторизация**

Для авторизации необходимо ввести в соответствующие поля логин и пароль и нажать кнопку “Логин” (рисунок 5.2). При удачной авторизации произойдёт перенаправление на главную страницу (рисунок 3.4).

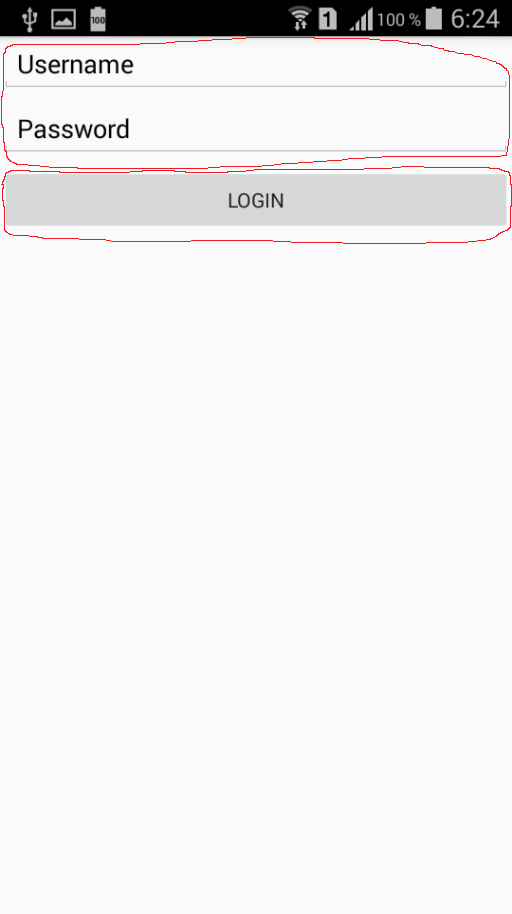
****

Рисунок 5.2 – Домашняя страница. Поля для ввода логина и пароля, кнопка“Логин”. Android.

1. **Главная страница**

На данной странице (рисунок 3.4) имеется информация о пользователе, и кнопки для перехода на страницу получения опроса и для LogOutа (рисунок 5.3).

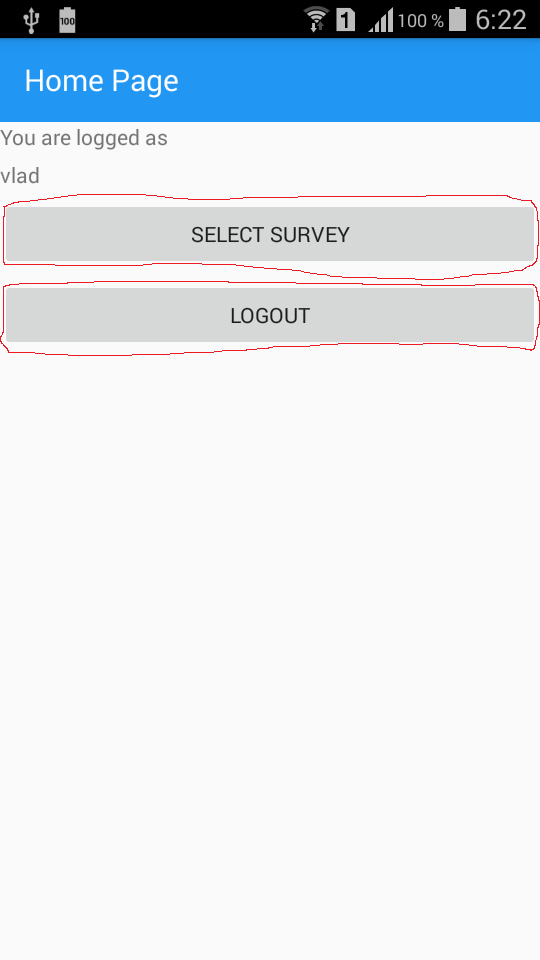
****

Рисунок 5.3 – Домашняя страница. Кнопка “Select survey” , кнопка “Logout”. Android.

1. **Страница получения опроса**

Для получения опроса необходимо ввести соответствующий Id опроса в поле ввода опроса и нажать кнопку “Получение опроса” (рисунок 5.4). При успешном получении опроса произойдёт перенаправление на страницу прохождения опроса (рисунок 3.6).

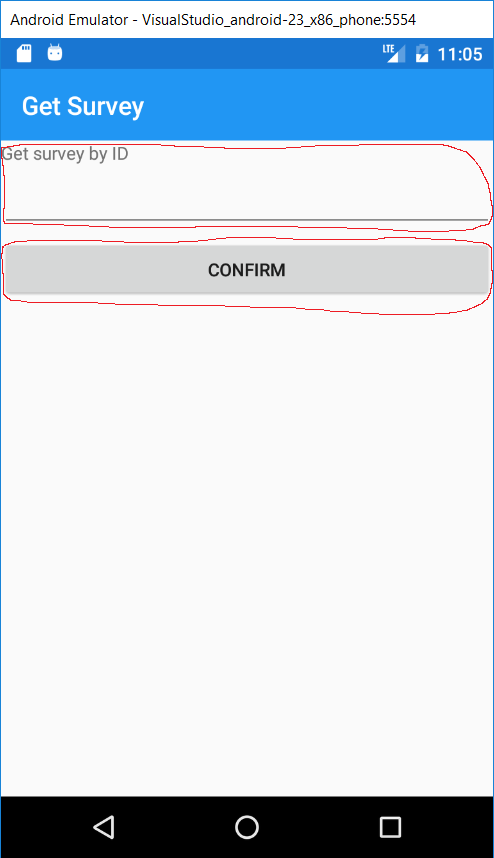


Рисунок 5.4 – Страница получения опроса. Поле для ввода Id, кнопка получения опроса. Android.

1. **Страница прохождения опроса**

Для ответа на вопрос необходимо ввести или выбрать соответствующий ответ в поле для ввода. На вопросы, помеченные символом “\*” необходимо обязательно дать ответ. В противном случае при попытке отправки ответов на сервер попытка будет отклонена, а неотвеченные вопросы будут подсвечены (рисунок 3.18).

Для отправки ответов на сервер необходимо нажать кнопку “Отправить ответы”. При валидных ответах будет произведена отправка ответов на сервер и произойдёт перенаправление на главную страницу (рисунок 3.4).

**ЗАКЛЮЧЕНИЕ**

На данный момент системы для автоматизации прохождения опросов имеют большую популярность, однако все они рассчитываются на большой круг задач, и имеется мало систем, рассчитанных на конкретную предметную область.

В рамках данного курсового проекта было разработано мобильное приложение, которое является частью системы для автоматизации процесса прохождения опросов. В данной версии программы реализован следующий функционал:

* взаимодействие с сервером;
* аутентификацию пользователей;
* получение опроса по Id;
* формирование представления опроса;
* редактирование ответов пользователем;
* валидация ответов на обязательные вопросы;
* отправка ответов на сервер.

При разработке программы в качестве основы использовались фрэймворк Xamarin + Xamarin.Forms.

Поставленная цель выполнена. Программа реализует все поставленные задачи для 1-й версии.

В последующих версиях реализовать следующий функционал:

* просмотр статистики пользователя;
* push-нотификации;
* офлайн-режим;
* создание хотспот точки для раздачи опроса;
* конструктор опросов;
* реализация механизма сессий;
* и т.д.

Список использованных источников

[1] [Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. – 896 с.](http://www.piter.com/collection/yazyk-programmirovaniya-ss-s/product/clr-via-c-programmirovanie-na-platforme-microsoft-net-framework-45-na-yazyke-c-4-e-izd-2)

[2] Каталог API и справочных материалов [Электронный ресурс]. –https://msdn.microsoft.com/library

[3] Стиллмен Э., Грин Д Head First. Изучаем C#. 3-е изд. – 816 с.

[4] Сайт Xamarin developers [Электронный ресурс]. – https://developer.xamarin.com/

[5] Сайт Xamarin developers. Руководства [Электронный ресурс]. – https://developer.xamarin.com/guides/

[6] Руководство по программированию для Xamarin Forms [Электронный ресурс]. – https://metanit.com/sharp/xamarin/

[7] XLabs. [Электронный ресурс]. – https://github.com/XLabs/Xamarin-Forms-Labs

**ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ**

**using CourseProject.MC.Common.Serialization;**

**using CourseProject.MC.Foundation.Interfaces;**

**using CourseProject.MC.Foundation.Services.Authentication;**

**using CourseProject.MC.Foundation.Services.Settings;**

**using CourseProject.MC.Foundation.Services.Surveys;**

**using CourseProject.MC.Foundation.Services.Web;**

**using CourseProject.MC.UI.Infrastructure.Services;**

**using CourseProject.MC.UI.Interfaces;**

**using CourseProject.MC.UI.Pages;**

**using CourseProject.MC.UI.ViewModels;**

**using Microsoft.Practices.Unity;**

**using Prism.Mvvm;**

**using Prism.Unity;**

**using Xamarin.Forms;**

**namespace CourseProject.MC**

**{**

**public partial class App : PrismApplication**

**{**

**public App()**

**: this(null)**

**{**

**}**

**public App(IPlatformInitializer initializer)**

**: base(initializer)**

**{**

**}**

**protected override async void OnInitialized()**

**{**

**InitializeComponent();**

**await NavigationService.NavigateAsync(MobileClientPages.SplashPage);**

**}**

**protected override void RegisterTypes()**

**{**

**Container.RegisterTypeForNavigation<NavigationPage>();**

**Container.RegisterTypeForNavigation<SignUpPage>();**

**Container.RegisterTypeForNavigation<SelectedSurveyPage>();**

**Container.RegisterTypeForNavigation<GetSurveyPage>();**

**Container.RegisterTypeForNavigation<SplashPage>();**

**Container.RegisterTypeForNavigation<HomePage>();**

**ViewModelLocationProvider.Register<SignUpPage, SignUpViewModel>();**

**ViewModelLocationProvider.Register<SelectedSurveyPage, SelectedSurveyViewModel>();**

**ViewModelLocationProvider.Register<GetSurveyPage, GetSurveyViewModel>();**

**ViewModelLocationProvider.Register<SplashPage, InitialViewModel>();**

**ViewModelLocationProvider.Register<HomePage, HomeViewModel>();**

**Container.RegisterType<ISurveysWebService, SurveysWebService>(new ContainerControlledLifetimeManager());**

**Container.RegisterType<IQuestionViewBuilderService, QuestionViewBuilderService>(new ContainerControlledLifetimeManager());**

**Container.RegisterType<IQuestionViewModelBuilderService, QuestionViewModelBuilderService>(new ContainerControlledLifetimeManager());**

**Container.RegisterType<ISurveysManager, SurveysManager>(new ContainerControlledLifetimeManager());**

**Container.RegisterType<IDialogService, DialogService>(new ContainerControlledLifetimeManager());**

**Container.RegisterType<IJsonSerializer, JsonSerializer>(new ContainerControlledLifetimeManager());**

**Container.RegisterType<ISettingsService, SettingsService>(new ContainerControlledLifetimeManager());**

**Container.RegisterType<INetworkManager, NetworkManager>(new ContainerControlledLifetimeManager());**

**Container.RegisterType<IAuthenticationWebService, AuthenticationWebService>(new ContainerControlledLifetimeManager());**

**Container.RegisterType<IAuthenticationService, LocalAuthenticationService>(new ContainerControlledLifetimeManager());**

**}**

**protected override void OnStart()**

**{**

**// Handle when your app starts**

**}**

**protected override void OnSleep()**

**{**

**// Handle when your app sleeps**

**}**

**protected override void OnResume()**

**{**

**// Handle when your app resumes**

**}**

**}**

**}**

**<?xml version="1.0" encoding="utf-8" ?>**

**<prism:PrismApplication xmlns="http://xamarin.com/schemas/2014/forms"**

**xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"**

**xmlns:themes="clr-namespace:CourseProject.MC.UI.Themes;assembly=CourseProject.MC.UI"**

**xmlns:prism="clr-namespace:Prism.Unity;assembly=Prism.Unity.Forms"**

**x:Class="CourseProject.MC.App">**

**<Application.Resources>**

**<ResourceDictionary MergedWith="themes:Styles" />**

**</Application.Resources>**

**</prism:PrismApplication>**

**using System;**

**using System.Globalization;**

**using Xamarin.Forms;**

**namespace CourseProject.MC.UI.Converters**

**{**

**public class DateTimeToStringConverter : IValueConverter**

**{**

**public object Convert(object value, Type targetType, object parameter, CultureInfo culture)**

**{**

**var date = value as string;**

**if (String.IsNullOrEmpty(date))**

**{**

**return DateTime.Now.Date;**

**}**

**var enteredDate = DateTime.Parse(date, CultureInfo.InvariantCulture);**

**return enteredDate;**

**}**

**public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)**

**{**

**if (value is DateTime)**

**{**

**var date = (DateTime)value;**

**if (date == DateTime.MinValue)**

**{**

**return String.Empty;**

**}**

**return date.ToString();**

**}**

**return String.Empty;**

**}**

**}**

**}**

**// Helpers/Settings.cs**

**using Plugin.Settings;**

**using Plugin.Settings.Abstractions;**

**namespace CourseProject.MC.UI.Helpers**

**{**

**/// <summary>**

**/// This is the Settings static class that can be used in your Core solution or in any**

**/// of your client applications. All settings are laid out the same exact way with getters**

**/// and setters.**

**/// </summary>**

**public static class Settings**

**{**

**private static ISettings AppSettings**

**{**

**get**

**{**

**return CrossSettings.Current;**

**}**

**}**

**#region Setting Constants**

**private const string SettingsKey = "settings\_key";**

**private static readonly string SettingsDefault = string.Empty;**

**#endregion**

**public static string GeneralSettings**

**{**

**get**

**{**

**return AppSettings.GetValueOrDefault(SettingsKey, SettingsDefault);**

**}**

**set**

**{**

**AppSettings.AddOrUpdateValue(SettingsKey, value);**

**}**

**}**

**}**

**}**

**using CourseProject.MC.UI.Infrastructure.Helpers.CustomControllers;**

**using CourseProject.MC.UI.Infrastructure.Helpers.DataTemplates;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**using Xamarin.Forms;**

**namespace CourseProject.MC.UI.Infrastructure.Helpers**

**{**

**public static class XamarinControllersCreator**

**{**

**public const int EditorMinimumHeight = 80;**

**public static Label GetLabel()**

**{**

**var label = new Label();**

**return label;**

**}**

**public static Label GetLabel(string labelText)**

**{**

**var label = new Label**

**{**

**Text = labelText**

**};**

**return label;**

**}**

**public static Label GetQuestionDescriptionLabel(QuestionViewModelBase questionViewModelBase)**

**{**

**return new QuestionDescriptionLabel(questionViewModelBase);**

**}**

**public static Entry GetEntry()**

**{**

**var entry = new Entry();**

**return entry;**

**}**

**public static Editor GetEditor()**

**{**

**var editor = new Editor();**

**return editor;**

**}**

**public static DatePicker GetDatePicker()**

**{**

**var datePicker = new DatePicker();**

**return datePicker;**

**}**

**public static RadioButtonDataTemplate GetRadioButtonDataTemplate()**

**{**

**var dataTemplate = new RadioButtonDataTemplate();**

**return dataTemplate;**

**}**

**public static CheckBoxDataTemplate GetCheckBoxDataTemplate()**

**{**

**var dataTemplate = new CheckBoxDataTemplate();**

**return dataTemplate;**

**}**

**}**

**}**

**using CourseProject.MC.UI.ViewModels.Questions;**

**using Xamarin.Forms;**

**namespace CourseProject.MC.UI.Infrastructure.Helpers.CustomControllers**

**{**

**public class QuestionDescriptionLabel : Label**

**{**

**private const string RequiredQuestionSymbol = "\*";**

**private readonly Span \_textQuestion;**

**private static readonly Color \_requiredQuestionSymbolColor = Color.Red;**

**public static readonly BindableProperty ForegroundQuestionTextColorProperty = BindableProperty.Create(**

**nameof(ForegroundTextColor),**

**typeof(Color),**

**typeof(QuestionDescriptionLabel),**

**Color.Default,**

**propertyChanging: (bindable, oldValue, newValue) =>**

**{**

**var controller = (QuestionDescriptionLabel)bindable;**

**controller.\_textQuestion.ForegroundColor = (Color)newValue;**

**});**

**public Color ForegroundTextColor**

**{**

**set => SetValue(ForegroundQuestionTextColorProperty, value);**

**get => (Color)GetValue(ForegroundQuestionTextColorProperty);**

**}**

**public QuestionDescriptionLabel(QuestionViewModelBase questionViewModel)**

**{**

**\_textQuestion = GetTextQuestionSpan(questionViewModel.Question.Text);**

**FormattedText = GetFormattedQuestionDescription(\_textQuestion, questionViewModel.Question.IsRequired);**

**BindingContext = questionViewModel;**

**this.SetBinding(ForegroundQuestionTextColorProperty, nameof(QuestionViewModelBase.QuestionDescriptionTextColor));**

**}**

**private static Span GetTextQuestionSpan(string text)**

**{**

**var result = new Span**

**{**

**Text = text**

**};**

**return result;**

**}**

**private static FormattedString GetFormattedQuestionDescription(Span textSpan, bool isRequired)**

**{**

**var resultString = new FormattedString();**

**if (isRequired)**

**{**

**resultString.Spans.Add(**

**new Span**

**{**

**Text = RequiredQuestionSymbol,**

**ForegroundColor = \_requiredQuestionSymbolColor**

**});**

**}**

**resultString.Spans.Add(textSpan);**

**return resultString;**

**}**

**}**

**}**

**using CourseProject.MC.UI.ViewModels.Questions;**

**using Xamarin.Forms;**

**using XLabs.Forms.Controls;**

**namespace CourseProject.MC.UI.Infrastructure.Helpers.DataTemplates**

**{**

**public class CheckBoxDataTemplate : DataTemplate**

**{**

**public CheckBoxDataTemplate()**

**: base(CreateDataTemplate)**

**{**

**}**

**public static ViewCell CreateDataTemplate()**

**{**

**var checkBoxDataTemplate = new ViewCell**

**{**

**View = CreateCheckBoxCellView()**

**};**

**return checkBoxDataTemplate;**

**}**

**private static View CreateCheckBoxCellView()**

**{**

**var checkBox = new CheckBox();**

**checkBox.SetBinding(CheckBox.CheckedProperty, nameof(CheckBoxAnswerOptionViewModel.Answer), BindingMode.TwoWay);**

**checkBox.SetBinding(CheckBox.DefaultTextProperty, nameof(CheckBoxAnswerOptionViewModel.OptionDescription), BindingMode.OneWay);**

**return checkBox;**

**}**

**}**

**}**

**using CourseProject.MC.UI.ViewModels.Questions;**

**using Xamarin.Forms;**

**using XLabs.Forms.Controls;**

**namespace CourseProject.MC.UI.Infrastructure.Helpers.DataTemplates**

**{**

**public class RadioButtonDataTemplate : DataTemplate**

**{**

**public RadioButtonDataTemplate()**

**: base(CreateDataTemplate)**

**{**

**}**

**public static ViewCell CreateDataTemplate()**

**{**

**var radioButtonDataTemplate = new ViewCell**

**{**

**View = CreateRadioButtonCellView()**

**};**

**return radioButtonDataTemplate;**

**}**

**private static View CreateRadioButtonCellView()**

**{**

**var radioButton = new CustomRadioButton();**

**radioButton.SetBinding(CustomRadioButton.CheckedProperty, nameof(RadioButtonAnswerOptionViewModel.Answer), BindingMode.TwoWay);**

**radioButton.SetBinding(CustomRadioButton.TextProperty, nameof(RadioButtonAnswerOptionViewModel.OptionDescription), BindingMode.OneWay);**

**return radioButton;**

**}**

**}**

**}**

**using System.Threading.Tasks;**

**using CourseProject.MC.Resources;**

**using CourseProject.MC.UI.Interfaces;**

**using Xamarin.Forms;**

**namespace CourseProject.MC.UI.Infrastructure.Services**

**{**

**public class DialogService : IDialogService**

**{**

**private const string OkAcceptButton = "OK";**

**public async Task ShowNotificationPopupAsync(string title, string message)**

**{**

**await Application.Current.MainPage.DisplayAlert(title, message, OkAcceptButton);**

**}**

**public async Task ShowErrorPopupAsync(string errorMessage)**

**{**

**await Application.Current.MainPage.DisplayAlert(MCResources.Error, errorMessage, OkAcceptButton);**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using CourseProject.MC.Common;**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.UI.Infrastructure.ViewBuilders.Questions;**

**using CourseProject.MC.UI.Interfaces;**

**using Xamarin.Forms;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Infrastructure.Services**

**{**

**[UsedImplicitly]**

**public class QuestionViewBuilderService : IQuestionViewBuilderService**

**{**

**private readonly Dictionary<QuestionTypes, QuestionViewBuilderBase> \_builders;**

**private readonly QuestionViewModelBuilderService \_questionViewModelBuilderService;**

**public QuestionViewBuilderService(QuestionViewModelBuilderService questionViewModelBuilderService)**

**{**

**\_questionViewModelBuilderService = questionViewModelBuilderService;**

**\_builders = new Dictionary<QuestionTypes, QuestionViewBuilderBase>();**

**InitializeBuilders();**

**}**

**public View CreateFrom(ref List<QuestionViewModelBase> questionViewModels, Question question)**

**{**

**try**

**{**

**var builder = \_builders[(QuestionTypes)question.TypeId];**

**var questionViewModel = \_questionViewModelBuilderService.CreateInstance(question);**

**var questionView = builder.GetInstance(questionViewModel, question);**

**questionViewModels.Add(questionViewModel);**

**return questionView;**

**}**

**catch (Exception ex)**

**{**

**// here can be logger**

**throw;**

**}**

**}**

**private void InitializeBuilders()**

**{**

**\_builders.Add(QuestionTypes.SingleLine, new SingleLineQuestionViewBuilder());**

**\_builders.Add(QuestionTypes.MultiLine, new MultiLineQuestionViewBuilder());**

**\_builders.Add(QuestionTypes.CheckBoxes, new CheckBoxesQuestionViewBuilder());**

**\_builders.Add(QuestionTypes.RadioButton, new RadioButtonsQuestionViewBuilder());**

**\_builders.Add(QuestionTypes.Date, new DateInputQuestionViewBuilder());**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using CourseProject.MC.Common;**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.UI.Infrastructure.ViewModelBuilders.Questions;**

**using CourseProject.MC.UI.Interfaces;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Infrastructure.Services**

**{**

**[UsedImplicitly]**

**public class QuestionViewModelBuilderService : IQuestionViewModelBuilderService**

**{**

**private readonly Dictionary<QuestionTypes, QuestionViewModelBuilderBase> \_builders;**

**public QuestionViewModelBuilderService()**

**{**

**\_builders = new Dictionary<QuestionTypes, QuestionViewModelBuilderBase>();**

**InitializeBuilders();**

**}**

**public QuestionViewModelBase CreateInstance(Question question)**

**{**

**try**

**{**

**var builder = \_builders[(QuestionTypes)question.TypeId];**

**var questionDataContract = builder.GetInstance(question);**

**return questionDataContract;**

**}**

**catch (Exception ex)**

**{**

**// Logger**

**throw;**

**}**

**}**

**private void InitializeBuilders()**

**{**

**\_builders.Add(QuestionTypes.SingleLine, new SingleLineQuestionViewModelBuilder());**

**\_builders.Add(QuestionTypes.MultiLine, new MultiLineQuestionViewModelBuilder());**

**\_builders.Add(QuestionTypes.CheckBoxes, new CheckBoxesQuestionViewModelBuilder());**

**\_builders.Add(QuestionTypes.RadioButton, new RadioButtonsQuestionViewModelBuilder());**

**\_builders.Add(QuestionTypes.Date, new DateInputQuestionViewModelBuilder());**

**}**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using Xamarin.Forms;**

**using CourseProject.MC.UI.Infrastructure.Helpers;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Infrastructure.ViewBuilders.Questions**

**{**

**public class CheckBoxesQuestionViewBuilder : QuestionViewBuilderBase**

**{**

**private const int ParticularCheckBoxItemHeight = 50;**

**private ListView \_listView;**

**protected override StackLayout CreateFrom(QuestionViewModelBase questionViewModelBase, Question question)**

**{**

**var content = new StackLayout();**

**var titleLabel = XamarinControllersCreator.GetQuestionDescriptionLabel(questionViewModelBase);**

**content.Children.Add(titleLabel);**

**\_listView = CreateCheckBoxesGroupListView();**

**content.Children.Add(\_listView);**

**content.MinimumHeightRequest = question.AnswerOptions.Length \* \_listView.RowHeight + titleLabel.Height;**

**BindDataContract(questionViewModelBase);**

**return content;**

**}**

**protected override void BindDataContract(QuestionViewModelBase questionViewModelBase)**

**{**

**BindDataContractCheckBoxes((CheckBoxesQuestionViewModel)questionViewModelBase);**

**}**

**private static ListView CreateCheckBoxesGroupListView()**

**{**

**var checkBoxesGroup = new ListView**

**{**

**ItemTemplate = XamarinControllersCreator.GetCheckBoxDataTemplate(),**

**RowHeight = ParticularCheckBoxItemHeight**

**};**

**return checkBoxesGroup;**

**}**

**private void BindDataContractCheckBoxes(CheckBoxesQuestionViewModel checkBoxesQuestionViewModel)**

**{**

**\_listView.ItemsSource = checkBoxesQuestionViewModel.Answers;**

**}**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.UI.Converters;**

**using CourseProject.MC.UI.Infrastructure.Helpers;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**using Xamarin.Forms;**

**namespace CourseProject.MC.UI.Infrastructure.ViewBuilders.Questions**

**{**

**public class DateInputQuestionViewBuilder : QuestionViewBuilderBase**

**{**

**private DatePicker \_datePicker;**

**protected override StackLayout CreateFrom(QuestionViewModelBase questionViewModelBase, Question question)**

**{**

**var content = new StackLayout();**

**content.Children.Add(XamarinControllersCreator.GetQuestionDescriptionLabel(questionViewModelBase));**

**\_datePicker = XamarinControllersCreator.GetDatePicker();**

**content.Children.Add(\_datePicker);**

**BindDataContract(questionViewModelBase);**

**return content;**

**}**

**protected override void BindDataContract(QuestionViewModelBase questionViewModelBase)**

**{**

**BindDataContractDateInput((DateInputQuestionViewModel)questionViewModelBase);**

**}**

**private void BindDataContractDateInput(DateInputQuestionViewModel dateInputQuestionViewModel)**

**{**

**\_datePicker.BindingContext = dateInputQuestionViewModel;**

**\_datePicker.SetBinding(DatePicker.DateProperty, nameof(dateInputQuestionViewModel.Answer), BindingMode.TwoWay, new DateTimeToStringConverter());**

**}**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.UI.Infrastructure.Helpers;**

**using Xamarin.Forms;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Infrastructure.ViewBuilders.Questions**

**{**

**public class MultiLineQuestionViewBuilder : QuestionViewBuilderBase**

**{**

**private Editor \_editor;**

**protected override StackLayout CreateFrom(QuestionViewModelBase questionViewModelBase, Question question)**

**{**

**var content = new StackLayout();**

**content.Children.Add(XamarinControllersCreator.GetQuestionDescriptionLabel(questionViewModelBase));**

**\_editor = XamarinControllersCreator.GetEditor();**

**content.Children.Add(\_editor);**

**BindDataContract(questionViewModelBase);**

**return content;**

**}**

**protected override void BindDataContract(QuestionViewModelBase questionViewModelBase)**

**{**

**BindDataContractMultiLine((MultiLineQuestionViewModel)questionViewModelBase);**

**}**

**private void BindDataContractMultiLine(MultiLineQuestionViewModel multiLineQuestionViewModel)**

**{**

**\_editor.BindingContext = multiLineQuestionViewModel;**

**\_editor.MinimumHeightRequest = XamarinControllersCreator.EditorMinimumHeight;**

**\_editor.SetBinding(Editor.TextProperty, nameof(multiLineQuestionViewModel.Answer), BindingMode.TwoWay);**

**}**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using Xamarin.Forms;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Infrastructure.ViewBuilders.Questions**

**{**

**public abstract class QuestionViewBuilderBase**

**{**

**public virtual View GetInstance(QuestionViewModelBase questionViewModelBase, Question question)**

**{**

**return CreateFrom(questionViewModelBase, question);**

**}**

**protected abstract StackLayout CreateFrom(QuestionViewModelBase questionDataContractBase, Question question);**

**protected abstract void BindDataContract(QuestionViewModelBase questionDataContractBase);**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using Xamarin.Forms;**

**using CourseProject.MC.UI.Infrastructure.Helpers;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Infrastructure.ViewBuilders.Questions**

**{**

**public class RadioButtonsQuestionViewBuilder : QuestionViewBuilderBase**

**{**

**private const int ParticularRadioButtonHeight = 50;**

**private ListView \_radioButtonGroup;**

**protected override StackLayout CreateFrom(QuestionViewModelBase questionViewModelBase, Question question)**

**{**

**var content = new StackLayout();**

**var titleLabel = XamarinControllersCreator.GetQuestionDescriptionLabel(questionViewModelBase);**

**content.Children.Add(titleLabel);**

**\_radioButtonGroup = CreateRadioButtonGroupListView();**

**content.Children.Add(\_radioButtonGroup);**

**content.MinimumHeightRequest = question.AnswerOptions.Length \* \_radioButtonGroup.RowHeight + titleLabel.Height;**

**BindDataContract(questionViewModelBase);**

**return content;**

**}**

**protected override void BindDataContract(QuestionViewModelBase questionViewModelBase)**

**{**

**BindDataContractRadioButtons((RadioButtonsQuestionViewModel)questionViewModelBase);**

**}**

**private static ListView CreateRadioButtonGroupListView()**

**{**

**var radioButtonGroup = new ListView**

**{**

**ItemTemplate = XamarinControllersCreator.GetRadioButtonDataTemplate(),**

**RowHeight = ParticularRadioButtonHeight**

**};**

**return radioButtonGroup;**

**}**

**private void BindDataContractRadioButtons(RadioButtonsQuestionViewModel radioButtonsQuestionViewModel)**

**{**

**\_radioButtonGroup.ItemsSource = radioButtonsQuestionViewModel.Answers;**

**}**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.UI.Infrastructure.Helpers;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**using Xamarin.Forms;**

**namespace CourseProject.MC.UI.Infrastructure.ViewBuilders.Questions**

**{**

**public class SingleLineQuestionViewBuilder : QuestionViewBuilderBase**

**{**

**private Entry \_entry;**

**protected override StackLayout CreateFrom(QuestionViewModelBase questionViewModelBase, Question question)**

**{**

**var content = new StackLayout();**

**content.Children.Add(XamarinControllersCreator.GetQuestionDescriptionLabel(questionViewModelBase));**

**\_entry = XamarinControllersCreator.GetEntry();**

**content.Children.Add(\_entry);**

**BindDataContract(questionViewModelBase);**

**return content;**

**}**

**protected override void BindDataContract(QuestionViewModelBase questionViewModelBase)**

**{**

**BindDataContractSingleLine((SingleLineQuestionViewModel)questionViewModelBase);**

**}**

**private void BindDataContractSingleLine(SingleLineQuestionViewModel singleLineQuestionViewModel)**

**{**

**\_entry.BindingContext = singleLineQuestionViewModel;**

**\_entry.SetBinding(Entry.TextProperty, nameof(singleLineQuestionViewModel.Answer), BindingMode.TwoWay);**

**}**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Infrastructure.ViewModelBuilders.Questions**

**{**

**public class CheckBoxesQuestionViewModelBuilder : QuestionViewModelBuilderBase**

**{**

**public override QuestionViewModelBase GetInstance(Question question)**

**{**

**return new CheckBoxesQuestionViewModel(question);**

**}**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Infrastructure.ViewModelBuilders.Questions**

**{**

**public class DateInputQuestionViewModelBuilder : QuestionViewModelBuilderBase**

**{**

**public override QuestionViewModelBase GetInstance(Question question)**

**{**

**return new DateInputQuestionViewModel(question);**

**}**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Infrastructure.ViewModelBuilders.Questions**

**{**

**public class MultiLineQuestionViewModelBuilder : QuestionViewModelBuilderBase**

**{**

**public override QuestionViewModelBase GetInstance(Question question)**

**{**

**return new MultiLineQuestionViewModel(question);**

**}**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Infrastructure.ViewModelBuilders.Questions**

**{**

**public abstract class QuestionViewModelBuilderBase**

**{**

**public abstract QuestionViewModelBase GetInstance(Question question);**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Infrastructure.ViewModelBuilders.Questions**

**{**

**public class RadioButtonsQuestionViewModelBuilder : QuestionViewModelBuilderBase**

**{**

**public override QuestionViewModelBase GetInstance(Question question)**

**{**

**return new RadioButtonsQuestionViewModel(question);**

**}**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Infrastructure.ViewModelBuilders.Questions**

**{**

**public class SingleLineQuestionViewModelBuilder : QuestionViewModelBuilderBase**

**{**

**public override QuestionViewModelBase GetInstance(Question question)**

**{**

**return new SingleLineQuestionViewModel(question);**

**}**

**}**

**}**

**namespace CourseProject.MC.UI.Infrastructure**

**{**

**public class NavigationParametersKeys**

**{**

**public static string SelectedSurvey = "selectedSurvey";**

**public static string IsOnlineMode = "isOnlineMode";**

**}**

**}**

**using System.Threading.Tasks;**

**namespace CourseProject.MC.UI.Interfaces**

**{**

**public interface IDialogService**

**{**

**Task ShowNotificationPopupAsync(string title, string message);**

**Task ShowErrorPopupAsync(string errorMessage);**

**}**

**}**

**using System.Collections.Generic;**

**using CourseProject.MC.DomainModel;**

**using Xamarin.Forms;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Interfaces**

**{**

**public interface IQuestionViewBuilderService**

**{**

**View CreateFrom(ref List<QuestionViewModelBase> questionViewModels, Question question);**

**}**

**}**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**namespace CourseProject.MC.UI.Interfaces**

**{**

**public interface IQuestionViewModelBuilderService**

**{**

**QuestionViewModelBase CreateInstance(Question question);**

**}**

**}**

**<?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:resources="clr-namespace:CourseProject.MC.Resources;assembly=CourseProject.MC.Resources"**

**xmlns:prism="clr-namespace:Prism.Mvvm;assembly=Prism.Forms"**

**x:Class="CourseProject.MC.UI.Pages.GetSurveyPage"**

**prism:ViewModelLocator.AutowireViewModel="True"**

**Title="{x:Static resources:MCResources.GetSurvey}">**

**<ContentPage.Content>**

**<StackLayout>**

**<Label Text="{x:Static resources:MCResources.GetSurveyById}" />**

**<Entry Text="{Binding SurveyId, Mode=OneWayToSource }" />**

**<Button Text="{x:Static resources:MCResources.Confirm}" Command="{Binding TryGetSurveyCommand}" />**

**</StackLayout>**

**</ContentPage.Content>**

**</ContentPage>**

**using Xamarin.Forms.Xaml;**

**namespace CourseProject.MC.UI.Pages**

**{**

**[XamlCompilation(XamlCompilationOptions.Compile)]**

**public partial class GetSurveyPage**

**{**

**public GetSurveyPage()**

**{**

**InitializeComponent();**

**}**

**}**

**}**

**using Xamarin.Forms.Xaml;**

**namespace CourseProject.MC.UI.Pages**

**{**

**[XamlCompilation(XamlCompilationOptions.Compile)]**

**public partial class SignUpPage**

**{**

**public SignUpPage()**

**{**

**InitializeComponent();**

**}**

**}**

**}**

**<?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:prism="clr-namespace:Prism.Mvvm;assembly=Prism.Forms"**

**x:Class="CourseProject.MC.UI.Pages.SplashPage"**

**prism:ViewModelLocator.AutowireViewModel="True"**

**BackgroundImage="Splash.png">**

**<ContentPage.Content>**

**<StackLayout>**

**<ProgressBar Progress="{Binding Progress}" VerticalOptions="CenterAndExpand" Margin="0,20,0,0" />**

**</StackLayout>**

**</ContentPage.Content>**

**</ContentPage>**

**using Xamarin.Forms.Xaml;**

**namespace CourseProject.MC.UI.Pages**

**{**

**[XamlCompilation(XamlCompilationOptions.Compile)]**

**public partial class SplashPage**

**{**

**public SplashPage()**

**{**

**InitializeComponent();**

**}**

**}**

**}**

**<?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:prism="clr-namespace:Prism.Mvvm;assembly=Prism.Forms"**

**x:Class="CourseProject.MC.UI.Pages.HomePage"**

**xmlns:resources="clr-namespace:CourseProject.MC.Resources;assembly=CourseProject.MC.Resources"**

**prism:ViewModelLocator.AutowireViewModel="True"**

**Title="{x:Static resources:MCResources.HomePage}">**

**<ContentPage.Content>**

**<StackLayout>**

**<Label Text="{x:Static resources:MCResources.LoggedAs}" />**

**<Label Text="{Binding Username}" />**

**<Button Text="{x:Static resources:MCResources.SelectSurvey}" Command="{Binding GoToSelectSurveyCommand}" />**

**<Button Text="{x:Static resources:MCResources.Logout}" Command="{Binding LogoutCommand}" VerticalOptions="End" />**

**</StackLayout>**

**</ContentPage.Content>**

**</ContentPage>**

**using Xamarin.Forms.Xaml;**

**namespace CourseProject.MC.UI.Pages**

**{**

**[XamlCompilation(XamlCompilationOptions.Compile)]**

**public partial class HomePage**

**{**

**public HomePage()**

**{**

**InitializeComponent();**

**}**

**}**

**}**

**using Xamarin.Forms;**

**namespace CourseProject.MC.UI.Pages**

**{**

**public static class MobileClientPages**

**{**

**private const string NavigationPrefix = nameof(NavigationPage) + "\\";**

**public static readonly string SelectedSurveyPage = nameof(Pages.SelectedSurveyPage);**

**public static readonly string GetSurveyPage = nameof(Pages.GetSurveyPage);**

**public static readonly string SignUpPage= nameof(Pages.SignUpPage);**

**public static readonly string HomePage = nameof(Pages.HomePage);**

**public static readonly string SplashPage= nameof(Pages.SplashPage);**

**public static readonly string SelectedSurveyNavigationPage = NavigationPrefix + nameof(Pages.SelectedSurveyPage);**

**public static readonly string GetSurveyNavigationPage = NavigationPrefix + nameof(Pages.GetSurveyPage);**

**public static readonly string SignUpNavigationPage = NavigationPrefix + nameof(Pages.SignUpPage);**

**public static readonly string HomeNavigationPage = NavigationPrefix + nameof(Pages.HomePage);**

**}**

**}**

**<?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:prism="clr-namespace:Prism.Mvvm;assembly=Prism.Forms"**

**x:Class="CourseProject.MC.UI.Pages.SelectedSurveyPage"**

**prism:ViewModelLocator.AutowireViewModel="True"**

**xmlns:resources="clr-namespace:CourseProject.MC.Resources;assembly=CourseProject.MC.Resources"**

**Title="{Binding SurveyId}">**

**<ContentPage.Content>**

**<ScrollView>**

**<StackLayout>**

**<ContentView Content="{Binding Questions}"/>**

**<Button Text="{x:Static resources:MCResources.FinishSurvey}" Command="{Binding TrySendAnswersCommand}" />**

**</StackLayout>**

**</ScrollView>**

**</ContentPage.Content>**

**</ContentPage>**

**using Xamarin.Forms.Xaml;**

**namespace CourseProject.MC.UI.Pages**

**{**

**[XamlCompilation(XamlCompilationOptions.Compile)]**

**public partial class SelectedSurveyPage**

**{**

**public SelectedSurveyPage()**

**{**

**InitializeComponent();**

**}**

**}**

**}**

**<?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:resources="clr-namespace:CourseProject.MC.Resources;assembly=CourseProject.MC.Resources"**

**xmlns:prism="clr-namespace:Prism.Mvvm;assembly=Prism.Forms"**

**x:Class="CourseProject.MC.UI.Pages.SignUpPage"**

**prism:ViewModelLocator.AutowireViewModel="True"**

**Title="{x:Static resources:MCResources.SignUp}">**

**<ContentPage.Content>**

**<StackLayout>**

**<Entry Placeholder="{x:Static resources:MCResources.Username}" Keyboard="Email" VerticalOptions="Center" Text="{Binding Username}" />**

**<Entry Placeholder="{x:Static resources:MCResources.Password}" IsPassword="True" Text="{Binding Password}" />**

**<Label Text="{x:Static resources:MCResources.IncorrectEmailOrPassword}" IsVisible="{Binding ShouldShowErrorMessage}" TextColor="Red" />**

**<Button Text="{x:Static resources:MCResources.Login}" Command="{Binding LoginCommand}" />**

**</StackLayout>**

**</ContentPage.Content>**

**</ContentPage>**

**using Xamarin.Forms;**

**using Xamarin.Forms.Xaml;**

**namespace CourseProject.MC.UI.Themes**

**{**

**[XamlCompilation(XamlCompilationOptions.Compile)]**

**public partial class Styles : ResourceDictionary**

**{**

**public Styles()**

**{**

**InitializeComponent();**

**}**

**}**

**}**

**<?xml version="1.0" encoding="utf-8" ?>**

**<ResourceDictionary xmlns="http://xamarin.com/schemas/2014/forms"**

**xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"**

**x:Class="CourseProject.MC.UI.Themes.Styles">**

**<Color x:Key="MobileClientBlueColor">Blue</Color>**

**<Style x:Key="TestEntry" TargetType="{x:Type Entry}">**

**<Setter Property="TextColor" Value="{StaticResource MobileClientBlueColor}"/>**

**</Style>**

**<Style x:Key="TestLabel" TargetType="{x:Type Label}">**

**<Setter Property="TextColor" Value="{StaticResource MobileClientBlueColor}"/>**

**</Style>**

**</ResourceDictionary>**

**using System.Threading.Tasks;**

**using CourseProject.MC.Foundation.Interfaces;**

**using CourseProject.MC.UI.Infrastructure;**

**using CourseProject.MC.UI.Pages;**

**using Prism.Mvvm;**

**using Prism.Navigation;**

**namespace CourseProject.MC.UI.ViewModels**

**{**

**public class InitialViewModel : BindableBase, INavigationAware**

**{**

**private const double FakeProgressInitial = 0.1;**

**private const double FakeProgressAfterLogin = 0.6;**

**private const double FakeProgressFinish = 0.9;**

**private readonly IAuthenticationService \_authenticationService;**

**private readonly INavigationService \_navigationService;**

**private readonly INetworkManager \_networkManager;**

**private double \_progress;**

**public double Progress**

**{**

**get => \_progress;**

**set => SetProperty(ref \_progress, value);**

**}**

**public InitialViewModel(**

**IAuthenticationService authenticationService,**

**INavigationService navigationService,**

**INetworkManager networkManager)**

**{**

**\_authenticationService = authenticationService;**

**\_navigationService = navigationService;**

**\_networkManager = networkManager;**

**}**

**public void OnNavigatedFrom(NavigationParameters parameters)**

**{**

**}**

**public async void OnNavigatedTo(NavigationParameters parameters)**

**{**

**Progress = FakeProgressInitial;**

**//var isServerAvailable = await \_networkManager.CheckConnectionToServer();**

**await Task.Delay(1000);**

**//if (!isServerAvailable)**

**//{**

**// var parameter = new NavigationParameters{{ NavigationParametersKeys.IsOnlineMode, false} };**

**// await \_navigationService.NavigateAsync(MobileClientPages.HomePage, parameter);**

**//}**

**var isAuthenticated = await \_authenticationService.CheckIfUserAuthenticated();**

**Progress = FakeProgressAfterLogin;**

**await Task.Delay(2000);**

**Progress = FakeProgressFinish;**

**await Task.Delay(1000);**

**if (isAuthenticated)**

**{**

**await \_navigationService.NavigateAsync(MobileClientPages.HomeNavigationPage);**

**}**

**else**

**{**

**await \_navigationService.NavigateAsync(MobileClientPages.SignUpPage);**

**}**

**}**

**public void OnNavigatingTo(NavigationParameters parameters)**

**{**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Windows.Input;**

**using CourseProject.MC.DomainModel;**

**using CourseProject.MC.Foundation.Interfaces;**

**using CourseProject.MC.UI.Infrastructure;**

**using CourseProject.MC.UI.Interfaces;**

**using CourseProject.MC.UI.Pages;**

**using CourseProject.MC.UI.ViewModels.Questions;**

**using Prism.Commands;**

**using Prism.Navigation;**

**using Prism.Mvvm;**

**using Xamarin.Forms;**

**namespace CourseProject.MC.UI.ViewModels**

**{**

**public class SelectedSurveyViewModel : BindableBase, INavigationAware**

**{**

**private readonly ISurveysWebService \_surveysWebService;**

**private readonly INavigationService \_navigationService;**

**private readonly IQuestionViewBuilderService \_questionViewBuilderService;**

**private readonly ISurveysManager \_surveysManager;**

**private readonly IDialogService \_dialogService;**

**private int \_surveyId;**

**private StackLayout \_questionViews;**

**private Survey \_selectedSurvey;**

**private List<Question> \_questions;**

**private List<QuestionViewModelBase> \_questionViewModels;**

**public StackLayout Questions**

**{**

**get => \_questionViews;**

**set => SetProperty(ref \_questionViews, value);**

**}**

**public int SurveyId**

**{**

**get => \_surveyId;**

**set => SetProperty(ref \_surveyId, value);**

**}**

**public ICommand TestCommand { get; protected set; }**

**public ICommand TrySendAnswersCommand { get; protected set; }**

**public SelectedSurveyViewModel(**

**ISurveysWebService surveysWebService,**

**INavigationService navigationService,**

**IQuestionViewBuilderService questionViewBuilderService,**

**ISurveysManager surveysManager,**

**IDialogService dialogService)**

**{**

**\_surveysWebService = surveysWebService;**

**\_navigationService = navigationService;**

**\_questionViewBuilderService = questionViewBuilderService;**

**\_surveysManager = surveysManager;**

**\_dialogService = dialogService;**

**TrySendAnswersCommand = new DelegateCommand(TrySendAnswers);**

**\_questionViewModels = new List<QuestionViewModelBase>();**

**Questions = new StackLayout();**

**}**

**public void OnNavigatedFrom(NavigationParameters parameters)**

**{**

**}**

**public void OnNavigatedTo(NavigationParameters parameters)**

**{**

**}**

**public void OnNavigatingTo(NavigationParameters parameters)**

**{**

**\_selectedSurvey = parameters[NavigationParametersKeys.SelectedSurvey] as Survey;**

**if (\_selectedSurvey != null)**

**{**

**\_questions = \_surveysManager.GetQuestionsFromSurvey(\_selectedSurvey);**

**SurveyId = \_selectedSurvey.Id;**

**foreach (var question in \_questions)**

**{**

**var questionView = \_questionViewBuilderService.CreateFrom(ref \_questionViewModels, question);**

**Questions.Children.Add(questionView);**

**}**

**}**

**}**

**private async void TrySendAnswers()**

**{**

**//var answers = GetAnswersFromViewModels();**

**if (IsValidAnswers())**

**{**

**var sendResult = await \_surveysWebService.SendAnswersOnSurvey(\_selectedSurvey);**

**if (sendResult)**

**{**

**await \_dialogService.ShowNotificationPopupAsync("Success", "Your answers successfully sent!");**

**await \_navigationService.NavigateAsync(new Uri(MobileClientPages.HomeNavigationPage, UriKind.Absolute));**

**}**

**else**

**{**

**await \_dialogService.ShowErrorPopupAsync("Passed survey couldn't be send to server. Please, try again.");**

**}**

**}**

**else**

**{**

**await \_dialogService.ShowErrorPopupAsync("All required questions should be answered!");**

**}**

**}**

**private List<Answer> GetAnswersFromViewModels()**

**{**

**var answers = new List<Answer>();**

**foreach (var questionViewModel in \_questionViewModels)**

**{**

**answers.Add(**

**new Answer**

**{**

**QuestionId = questionViewModel.Question.Id,**

**Value = questionViewModel.Answer**

**});**

**}**

**return answers;**

**}**

**private bool IsValidAnswers()**

**{**

**var isAllAnswersValid = true;**

**foreach (var questionViewModel in \_questionViewModels)**

**{**

**if (!questionViewModel.IsValid())**

**{**

**isAllAnswersValid = false;**

**}**

**}**

**return isAllAnswersValid;**

**}**

**}**

**}**

**using System;**

**using System.Windows.Input;**

**using CourseProject.MC.Common.Validation;**

**using CourseProject.MC.Foundation.Interfaces;**

**using CourseProject.MC.UI.Pages;**

**using Prism.Commands;**

**using Prism.Mvvm;**

**using Prism.Navigation;**

**namespace CourseProject.MC.UI.ViewModels**

**{**

**public class SignUpViewModel : BindableBase**

**{**

**private string \_username;**

**private string \_password;**

**private bool \_shouldShowErrorMessage;**

**private readonly INavigationService \_navigationService;**

**private readonly IAuthenticationService \_authenticationService;**

**public string Username**

**{**

**get => \_username;**

**set => SetProperty(ref \_username, value);**

**}**

**public string Password**

**{**

**get => \_password;**

**set => SetProperty(ref \_password, value);**

**}**

**public bool ShouldShowErrorMessage**

**{**

**get => \_shouldShowErrorMessage;**

**set => SetProperty(ref \_shouldShowErrorMessage, value);**

**}**

**public ICommand LoginCommand { get; protected set; }**

**public SignUpViewModel(INavigationService navigationService, IAuthenticationService authenticationService)**

**{**

**\_navigationService = navigationService;**

**\_authenticationService = authenticationService;**

**LoginCommand = new DelegateCommand(Login);**

**}**

**private async void Login()**

**{**

**if (Username.Trim() != String.Empty && Password.Trim() != String.Empty)**

**{**

**ShouldShowErrorMessage = false;**

**await \_authenticationService.TryAuthenticateByCredentials(Username, Password);**

**await \_navigationService.NavigateAsync(MobileClientPages.HomeNavigationPage);**

**}**

**else**

**{**

**ShouldShowErrorMessage = true;**

**}**

**}**

**}**

**}**

**using System;**

**using System.Windows.Input;**

**using CourseProject.MC.Foundation.Interfaces;**

**using CourseProject.MC.Resources;**

**using CourseProject.MC.UI.Infrastructure;**

**using CourseProject.MC.UI.Interfaces;**

**using CourseProject.MC.UI.Pages;**

**using Prism.Commands;**

**using Prism.Mvvm;**

**using Prism.Navigation;**

**using Xamarin.Forms;**

**namespace CourseProject.MC.UI.ViewModels**

**{**

**public class GetSurveyViewModel : BindableBase, INavigationAware**

**{**

**private readonly ISurveysWebService \_surveysWebService;**

**private readonly INavigationService \_navigationService;**

**private readonly IDialogService \_dialogService;**

**private string \_surveyId;**

**public string SurveyId**

**{**

**get => \_surveyId;**

**set => SetProperty(ref \_surveyId, value);**

**}**

**public ICommand TryGetSurveyCommand { get; protected set; }**

**public GetSurveyViewModel(**

**ISurveysWebService surveysWebService,**

**INavigationService navigationService,**

**IDialogService dialogService)**

**{**

**\_surveysWebService = surveysWebService;**

**\_navigationService = navigationService;**

**\_dialogService = dialogService;**

**TryGetSurveyCommand = new DelegateCommand(TryGetSurvey);**

**}**

**public void OnNavigatedFrom(NavigationParameters parameters)**

**{**

**}**

**public void OnNavigatedTo(NavigationParameters parameters)**

**{**

**}**

**public void OnNavigatingTo(NavigationParameters parameters)**

**{**

**SurveyId = String.Empty;**

**}**

**private async void TryGetSurvey()**

**{**

**if (SurveyIdIsValid())**

**{**

**var survey = await \_surveysWebService.GetSurveyById(Convert.ToInt32(SurveyId));**

**if (survey != null)**

**{**

**var parameter = new NavigationParameters{{NavigationParametersKeys.SelectedSurvey, survey}};**

**await \_navigationService.NavigateAsync(MobileClientPages.SelectedSurveyPage, parameter);**

**}**

**else**

**{**

**await \_dialogService.ShowErrorPopupAsync(MCResources.CannotGetSurvey);**

**}**

**}**

**else**

**{**

**await \_dialogService.ShowErrorPopupAsync(MCResources.IncorrectId);**

**}**

**}**

**private bool SurveyIdIsValid()**

**{**

**if (int.TryParse(SurveyId, out int numericSurveyId))**

**{**

**return numericSurveyId > 0;**

**}**

**return false;**

**}**

**}**

**}**

**using System;**

**using System.Windows.Input;**

**using CourseProject.MC.Foundation;**

**using CourseProject.MC.Foundation.Interfaces;**

**using CourseProject.MC.UI.Pages;**

**using Prism.Commands;**

**using Prism.Mvvm;**

**using Prism.Navigation;**

**namespace CourseProject.MC.UI.ViewModels**

**{**

**public class HomeViewModel : BindableBase, INavigationAware**

**{**

**private readonly ISettingsService \_settingsService;**

**private readonly INavigationService \_navigationService;**

**private readonly INetworkManager \_networkManager;**

**private readonly IAuthenticationService \_authenticationService;**

**private string \_username;**

**public string Username**

**{**

**get => \_username;**

**set => SetProperty(ref \_username, value);**

**}**

**public ICommand GoToSelectSurveyCommand { get; }**

**public ICommand LogoutCommand { get; }**

**public HomeViewModel(**

**ISettingsService settingsService,**

**INavigationService navigationService,**

**INetworkManager networkManager,**

**IAuthenticationService authenticationService)**

**{**

**\_navigationService = navigationService;**

**\_settingsService = settingsService;**

**\_networkManager = networkManager;**

**\_authenticationService = authenticationService;**

**GoToSelectSurveyCommand = new DelegateCommand(GoToSelectSurvey);**

**LogoutCommand = new DelegateCommand(Logout);**

**}**

**public void OnNavigatedFrom(NavigationParameters parameters)**

**{**

**}**

**public void OnNavigatedTo(NavigationParameters parameters)**

**{**

**Username = \_settingsService.GetValueOrDefault(AuthSettingsKeys.Username, String.Empty);**

**}**

**public void OnNavigatingTo(NavigationParameters parameters)**

**{**

**}**

**private async void GoToSelectSurvey()**

**{**

**await \_navigationService.NavigateAsync(MobileClientPages.GetSurveyPage);**

**}**

**private async void Logout()**

**{**

**\_authenticationService.Logout();**

**await \_navigationService.NavigateAsync(new Uri(MobileClientPages.SignUpPage, UriKind.Absolute));**

**}**

**}**

**}**

**using Prism.Mvvm;**

**using CourseProject.MC.DomainModel;**

**using Xamarin.Forms;**

**namespace CourseProject.MC.UI.ViewModels.Questions**

**{**

**public abstract class QuestionViewModelBase : BindableBase**

**{**

**private readonly Color \_defaultQuestionDescriptionColor = Color.Default;**

**private readonly Color \_errorQuestionDescriptionColor = Color.Brown;**

**private Color \_questionDescriptionTextColor;**

**public Question Question { get; set; }**

**public virtual string Answer { get; set; }**

**public virtual Color QuestionDescriptionTextColor**

**{**

**get => \_questionDescriptionTextColor;**

**set => SetProperty(ref \_questionDescriptionTextColor, value);**

**}**

**protected QuestionViewModelBase(Question question)**

**{**

**Question = question;**

**\_questionDescriptionTextColor = Color.Default;**

**}**

**protected void ChangeTextColor(bool isCorrect)**

**{**

**QuestionDescriptionTextColor = isCorrect ? \_defaultQuestionDescriptionColor : \_errorQuestionDescriptionColor;**

**}**

**public abstract bool IsValid();**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using CourseProject.MC.DomainModel;**

**using Prism.Mvvm;**

**namespace CourseProject.MC.UI.ViewModels.Questions**

**{**

**public class RadioButtonAnswerOptionViewModel : BindableBase**

**{**

**private readonly AnswerOption \_answerOption;**

**private readonly List<RadioButtonAnswerOptionViewModel> \_otherAnswersViewModels;**

**public bool Answer**

**{**

**get => Convert.ToBoolean(\_answerOption.Value);**

**set**

**{**

**var changedAnswerValue = false;**

**SetProperty(ref changedAnswerValue, value);**

**\_answerOption.Value = Convert.ToString(changedAnswerValue);**

**if (changedAnswerValue)**

**{**

**ResetOtherOptions();**

**}**

**}**

**}**

**public string OptionDescription**

**{**

**get**

**{**

**//todo: correct model of AnswerOption: need to add option description field**

**return "TestOption";**

**}**

**}**

**public RadioButtonAnswerOptionViewModel(AnswerOption answerOption, List<RadioButtonAnswerOptionViewModel> otherOptions)**

**{**

**\_answerOption = answerOption;**

**\_otherAnswersViewModels = otherOptions;**

**}**

**private void ResetOtherOptions()**

**{**

**foreach (var answerOption in \_otherAnswersViewModels)**

**{**

**if (!this.Equals(answerOption))**

**{**

**answerOption.Answer = false;**

**answerOption.RaisePropertyChanged(nameof(Answer));**

**}**

**}**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using CourseProject.MC.DomainModel;**

**namespace CourseProject.MC.UI.ViewModels.Questions**

**{**

**public class RadioButtonsQuestionViewModel : QuestionViewModelBase**

**{**

**public List<RadioButtonAnswerOptionViewModel> Answers { get; set; }**

**public RadioButtonsQuestionViewModel(Question question)**

**: base(question)**

**{**

**Answers = InitializeAnswers(Question.AnswerOptions);**

**}**

**public override bool IsValid()**

**{**

**foreach (var option in Question.AnswerOptions)**

**{**

**if (Convert.ToBoolean(option.Value))**

**{**

**ChangeTextColor(true);**

**return true;**

**}**

**}**

**ChangeTextColor(false);**

**return false;**

**}**

**private static List<RadioButtonAnswerOptionViewModel> InitializeAnswers(IEnumerable<AnswerOption> oldAnswers)**

**{**

**var answersOptionsViewModels = new List<RadioButtonAnswerOptionViewModel>();**

**foreach (var answer in oldAnswers)**

**{**

**answersOptionsViewModels.Add(new RadioButtonAnswerOptionViewModel(answer, answersOptionsViewModels));**

**}**

**return answersOptionsViewModels;**

**}**

**}**

**}**

**using System;**

**using CourseProject.MC.DomainModel;**

**using Xamarin.Forms;**

**namespace CourseProject.MC.UI.ViewModels.Questions**

**{**

**public class SingleLineQuestionViewModel : QuestionViewModelBase**

**{**

**public SingleLineQuestionViewModel(Question question)**

**: base(question)**

**{**

**}**

**public override string Answer**

**{**

**get => Question.AnswerOptions[0].Value;**

**set**

**{**

**try**

**{**

**var changedAnswerValue = String.Empty;**

**SetProperty(ref changedAnswerValue, value);**

**Question.AnswerOptions[0].Value = changedAnswerValue;**

**}**

**catch (Exception ex)**

**{**

**// Logger**

**}**

**}**

**}**

**public override bool IsValid()**

**{**

**var isFieldEmpty = Question.AnswerOptions[0].Value.Trim().Length > 0;**

**ChangeTextColor(isFieldEmpty);**

**return isFieldEmpty;**

**}**

**}**

**}**

**using System;**

**using CourseProject.MC.DomainModel;**

**using Prism.Mvvm;**

**namespace CourseProject.MC.UI.ViewModels.Questions**

**{**

**public class CheckBoxAnswerOptionViewModel : BindableBase**

**{**

**private readonly AnswerOption \_answerOption;**

**public bool Answer**

**{**

**get => Convert.ToBoolean(\_answerOption.Value);**

**set**

**{**

**var changedAnswerValue = false;**

**SetProperty(ref changedAnswerValue, value);**

**\_answerOption.Value = Convert.ToString(changedAnswerValue);**

**}**

**}**

**public string OptionDescription**

**{**

**get**

**{**

**//todo: correct model of AnswerOption: need to add option description field**

**return "TestOption";**

**}**

**}**

**public CheckBoxAnswerOptionViewModel(AnswerOption answerOption)**

**{**

**\_answerOption = answerOption;**

**}**

**}**

**}**

**using System.Collections.Generic;**

**using System.Linq;**

**using CourseProject.MC.DomainModel;**

**namespace CourseProject.MC.UI.ViewModels.Questions**

**{**

**public class CheckBoxesQuestionViewModel : QuestionViewModelBase**

**{**

**public List<CheckBoxAnswerOptionViewModel> Answers { get; set; }**

**public CheckBoxesQuestionViewModel(Question question)**

**: base(question)**

**{**

**Answers = InitializeAnswers(Question.AnswerOptions);**

**}**

**public override bool IsValid()**

**{**

**var isAtLeastOneChecked = Answers.Count(a => a.Answer) != 0;**

**ChangeTextColor(isAtLeastOneChecked);**

**return isAtLeastOneChecked;**

**}**

**private static List<CheckBoxAnswerOptionViewModel> InitializeAnswers(IEnumerable<AnswerOption> oldAnswers)**

**{**

**var answersOptionsViewModels = new List<CheckBoxAnswerOptionViewModel>();**

**foreach (var answer in oldAnswers)**

**{**

**answersOptionsViewModels.Add(new CheckBoxAnswerOptionViewModel(answer));**

**}**

**return answersOptionsViewModels;**

**}**

**}**

**}**

**using System;**

**using CourseProject.MC.DomainModel;**

**namespace CourseProject.MC.UI.ViewModels.Questions**

**{**

**public class DateInputQuestionViewModel : QuestionViewModelBase**

**{**

**public DateInputQuestionViewModel(Question question)**

**: base(question)**

**{**

**}**

**public override string Answer**

**{**

**get => Question.AnswerOptions[0].Value;**

**set**

**{**

**try**

**{**

**var changedAnswerValue = String.Empty;**

**SetProperty(ref changedAnswerValue, value);**

**Question.AnswerOptions[0].Value = changedAnswerValue;**

**}**

**catch (Exception ex)**

**{**

**// Logger**

**}**

**}**

**}**

**public override bool IsValid()**

**{**

**ChangeTextColor(true);**

**return true;**

**}**

**}**

**}**

**using System;**

**using CourseProject.MC.DomainModel;**

**namespace CourseProject.MC.UI.ViewModels.Questions**

**{**

**public class MultiLineQuestionViewModel : QuestionViewModelBase**

**{**

**public MultiLineQuestionViewModel(Question questionViewModel)**

**: base(questionViewModel)**

**{**

**}**

**public override string Answer**

**{**

**get => Question.AnswerOptions[0].Value;**

**set**

**{**

**try**

**{**

**var changedAnswerValue = String.Empty;**

**SetProperty(ref changedAnswerValue, value);**

**Question.AnswerOptions[0].Value = changedAnswerValue;**

**}**

**catch (Exception ex)**

**{**

**// Logger**

**}**

**}**

**}**

**public override bool IsValid()**

**{**

**var isFieldEmpty = Question.AnswerOptions[0].Value.Trim().Length > 0;**

**ChangeTextColor(isFieldEmpty);**

**return isFieldEmpty;**

**}**

**}**

**}**

**ПРИЛОЖЕНИЕ Б. СХЕМА АЛГОРИТМА ФОРМИРОВАНИЯ ПРЕДСТАВЛЕНИЯ ОПРОСА И ПРИВЯЗКИ К VIEWMODEL.**