<https://habr.com/ru/company/devexpress/blog/420999/>

<https://habr.com/ru/company/devexpress/blog/421571/>

<https://metanit.com/sharp/xamarin/>

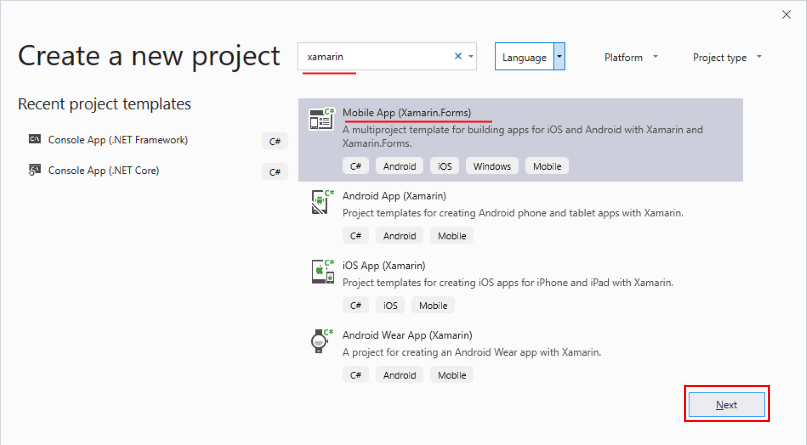
<https://reactnative.dev/docs/getting-started>

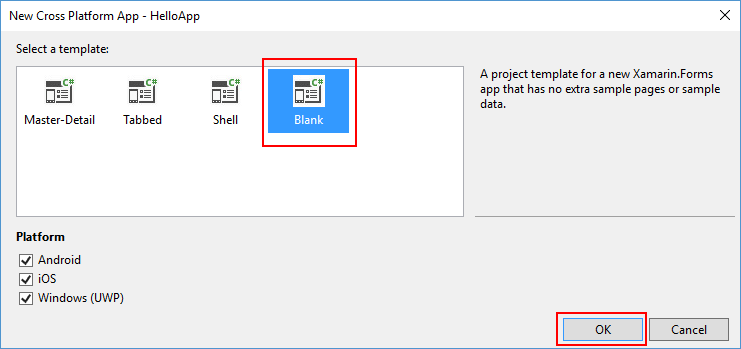
Xamarin:

Разработка на Xamarin ведется на C# в Visual Studio.

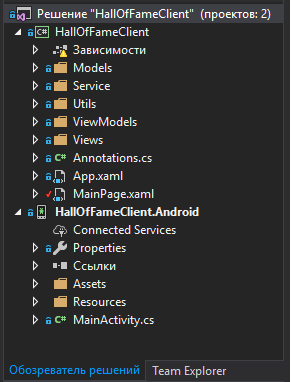
Создание проекта:

Все довольно просто, проект создается кнопкой «Создать проект».



Так же можно выбрать некоторые шаблоны и целевые платформы:

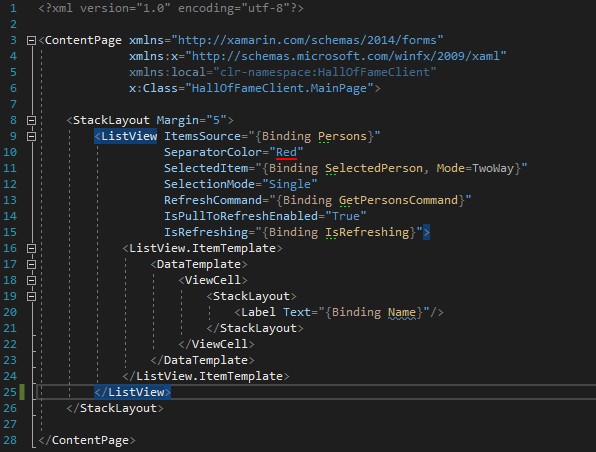
Структура проекта для Xamarin.Forms будет выглядеть подобным образом:



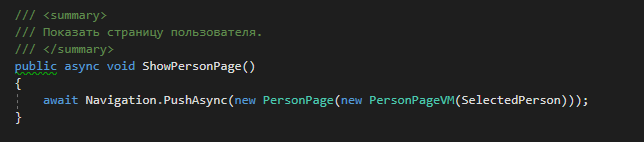
Создается отдельная часть для платформозависимого кода и общая для всей логики приложения. В общей части так же создается и интерфейс.

Синтаксис Xamarin.Forms крайне близок к синтаксису WPF, а сама общая часть написана на .NET Standard. В результате вы получаете возможность использования MVVM подхода при разработке приложения, а также доступ к огромному количеству сторонних библиотек, написанных для .NET Standard и уже лежащих в NuGet, которые вы спокойно можете использовать у себя в Xamarin.Forms приложениях.

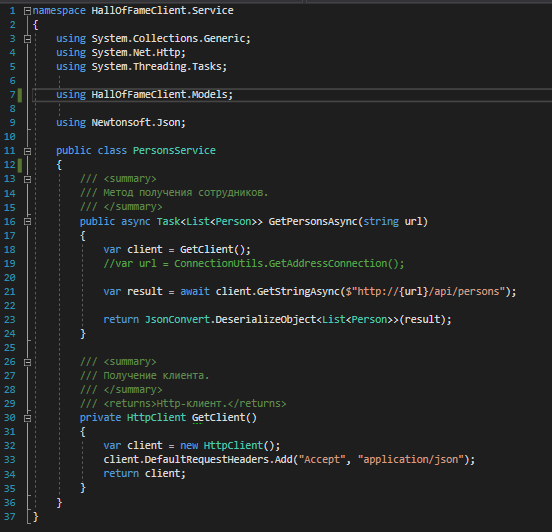
Так будет выглядеть создание списка пользователей с биндинками выбранного и командой обновления.



Для навигации между окнами используется класс Navigation. В примере ниже показано создание страницы пользователя с передачей выбранного:



Для взаимодействия с сервером используются привычные .NET библиотеки:



Запуск приложения на эмуляторе не представляет труда, существует диспетчер виртуальных устройств, в котором можно создать конфигурацию под нужную версию ОС.

Отладка на устройстве тоже не представляет сложности. Необходимо на устройстве выставить режим разработчика и разрешить отладку по USB и в студии появиться вариант запуска на устройстве.

Чтобы создать установочный файл необходимо выбрать значок решения, нажать правой кнопкой и выбрать «Архивировать все», предварительно собрав решение. В итоге вы получите файл apk, который можно перетащить на устройство и установить или добавить в Google Play.

Итого:

Плюсы:

- Единая среда разработки, тестирования, отладки, публикации.

- Возможность комплексной разработки клиент-серверного приложения.

Минусы:

- API, доступное в общей части, достаточно скудное, ибо содержит в себе только то, что является общим для всех платформ.

Для чего можно использовать:

- Высокопроизводительные приложения с нативным внешним видом, где не требуется сложного интерфейса.

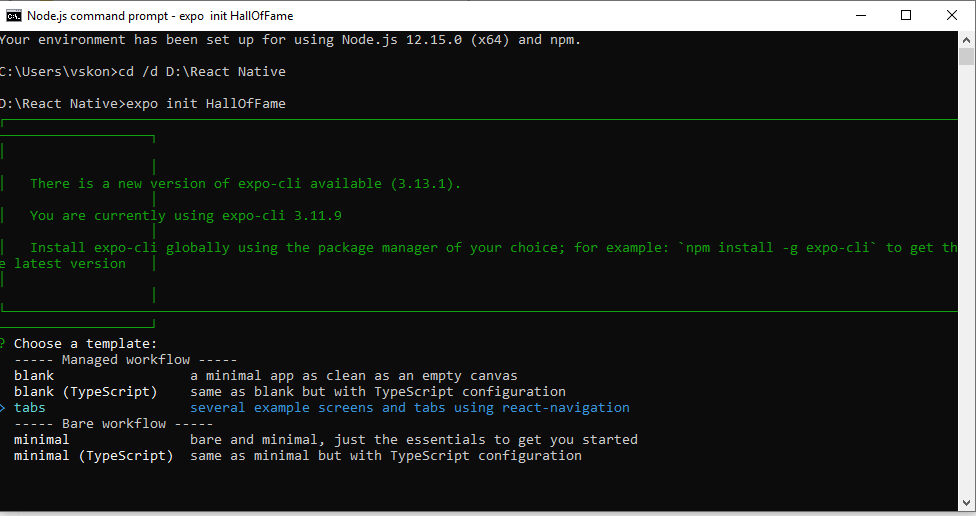
React Native:

Для работы с React Native понадобиться установить Node.js и Git bash. Разработка ведется в любом удобном редакторе кода, например VSCode.

Для создания проекта необходимо запустить следующую команду в командной строке Node.js:

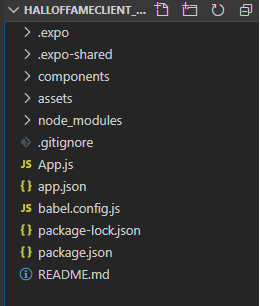
expo init %ProjectName%

Будет предложены шаблонных вариантов проектов:



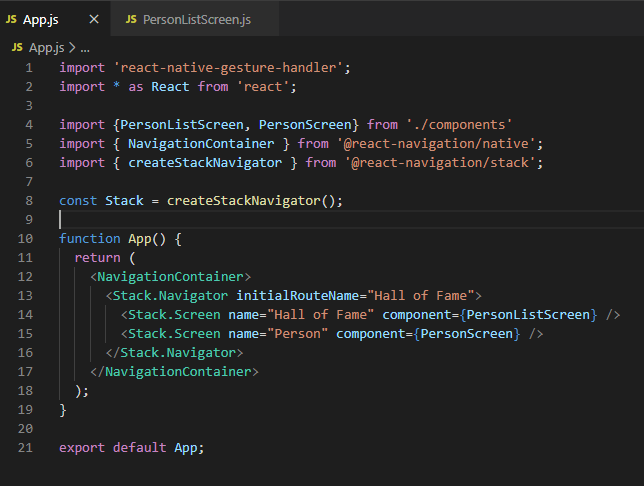
Далее после выбора будет создан проект.

Структура проекта выглядит подобным образом:



Точка входа находиться в файле App.js.

Подобным образом выглядит описание экранов, используемых в приложении:



Компонент это основная единица разработки. Выглядит следующим образом:



В данном случае создается страница содержащая имя пользователя, и перечень его навыков.

В каждом компоненте должен быть метод render, который и содержит отображаемый контент.

Чтобы запустить приложение необходимо выполнить команду npm start в директории приложения. В браузере будет запущен Metro Builder. С помощью приложения Expo можно запустить приложение на устройстве.

React Native поддерживает Hot Reload, что означает, что при внесении изменений в текст приложения и сохранении изменений они мгновенно отобразятся на устройстве.

Дебаг производиться в браузере.

Чтобы собрать приложение и установить его на устройство нужно запустить приложение в одном окне. В другом нужно выполнить команду expo build:android. Будет предложено создать кейстор или выбрать свой. Дальше нужно будет дождаться завершения и все готово.

Итого:

Плюсы:

- Удобная система отладки приложения с помощью expo. При каждом вызове сохранения проект перегружается и происходит это очень быстро.

Минусы:

- Сложно настраиваемая инфраструктура, приходится много работать с командной строкой.

- Исключения зачастую не понятные.

- Приложения имеют проблемы с отображением больших наборов данных.

Использование:

- Подойдет, если надо портировать веб-приложение на мобильное устройство.

Вывод: Нельзя с большой уверенностью сказать, какой фреймворк лучше. Оба предоставляют практически одинаковые возможности. В данном случае нужно отталкиваться от того, на каком языке будет вестись разработка.

**Xamarin.Forms** представляет платформу, которая нацелена на создание кроссплатформенных приложений под Android, iOS и Windows 10.

Преимущества использования Xamarin.Forms:

* В процессе разработки создается единый код для всех платформ
* Xamarin предоставляет прямой доступ к нативным API каждой платформы
* При создании приложений мы можем использовать платформу .NET и язык программирования C# (а также F#), который является достаточно производительным, и в тоже время ясным и простым для освоения и применения

Xamarin.Forms является надстройкой над Xamarin.iOs и Xamarin.Android. После сборки общая часть “разворачивается” в стандартные нативные контролы, так что по сути вы получаете полностью нативные приложения под все поддерживаемые платформы.  
  
Синтаксис Xamarin.Forms крайне близок к синтаксису WPF, а сама общая часть написана на .NET Standard. В результате вы получаете возможность использования MVVM подхода при разработке приложения, а также доступ к огромному количеству сторонних библиотек, написанных для .NET Standard и уже лежащих в NuGet, которые вы спокойно можете использовать у себя в Xamarin.Forms приложениях.

* **Xamarin:** его производительность сопоставима с **[нативными мобильными приложениями](https://www.xicom.biz/offerings/mobile-app-development/?utm_source=SM14" \t "_blank)** , что делает его идеальным вариантом для разработки высокопроизводительного приложения с нативным внешним видом и [**интерфейсом**](https://www.xicom.biz/offerings/mobile-app-development/?utm_source=SM14) .

С технической точки зрения, Xamarin с нативными библиотеками обернут в DOT.NET для кросс-платформенной разработки приложений. Кроме того, Xamarin может использовать все собственные и новейшие API-интерфейсы для использования скрытого потенциала в приложениях Xamarin, таких как **ARKit для iOS** и **Android-Multi Windows** .

**React Native:** Приложения, основанные на React Native, обычно имеют проблемы с отображением больших наборов данных, поскольку они зависят от моста Javascript. Таким образом, повышение скорости работы приложений с помощью сторонних библиотек может быть значительным вариантом, но этого недостаточно для победы над Flutter в конкурентной гонке.

* Вы или ваша команда знакомы с веб-разработкой и JavaScript.
* Вы портируете существующее веб-приложение на iOS и Android.
* Вам нужно нанять разработчиков; JavaScript - самый популярный язык программирования в мире.

**Почему Ксамарин?**

**Как разработчик или технический менеджер, вы можете выбрать Xamarin, потому что:**

* Вы или ваша команда обладаете обширным опытом разработки Microsoft и C #, и / или службы Microsoft являются ядром технической базы вашей компании.
* Это самый зрелый кроссплатформенный набор инструментов для разработки на рынке.
* Он предлагает пользовательский интерфейс приложения.

Возможность повторного использования кода, безусловно, является одним из главных преимуществ использования Xamarin, особенно если ваши приложения интегрируются с серверной средой .NET. Это включает в себя Azure, который, по словам Microsoft, в настоящее время используется более чем 95 процентами компаний из списка Fortune 500.