//слайд 2 Xamarin — это платформа с открытым исходным кодом, предназначенная для построения современных производительных приложений для iOS, Android и Windows с .NET. Платформа Xamarin представляет собой уровень абстракции, который обеспечивает управление взаимодействием между общим кодом и кодом базовой платформы. Xamarin выполняется в управляемой среде, которая реализует такие возможности, как выделение памяти и сборка мусора.

//слайд 3

* 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.

Благодаря Xamarin в среднем 90 % кода приложения может использоваться без изменений на разных платформах. С помощью этого шаблона разработчик может написать всю бизнес-логику на одном языке (или использовать существующий код приложения), но при этом получить характеристики производительности, оформление и поведение, характерные для каждой соответствующей платформы. Приложения Xamarin можно писать на ПК или Mac и компилировать в собственные пакеты приложений, например в файлы с расширением .apk для Android или .ipa для iOS.

//слайд 5

**На кого ориентирована платформа Xamarin**

Платформа Xamarin ориентирована на разработчиков, перед которыми стоят следующие задачи:

* Совместное использование кода, тестов и бизнес-логики на различных платформах.
* Написание кроссплатформенных приложений на языке C# в Visual Studio.

//слайд 7

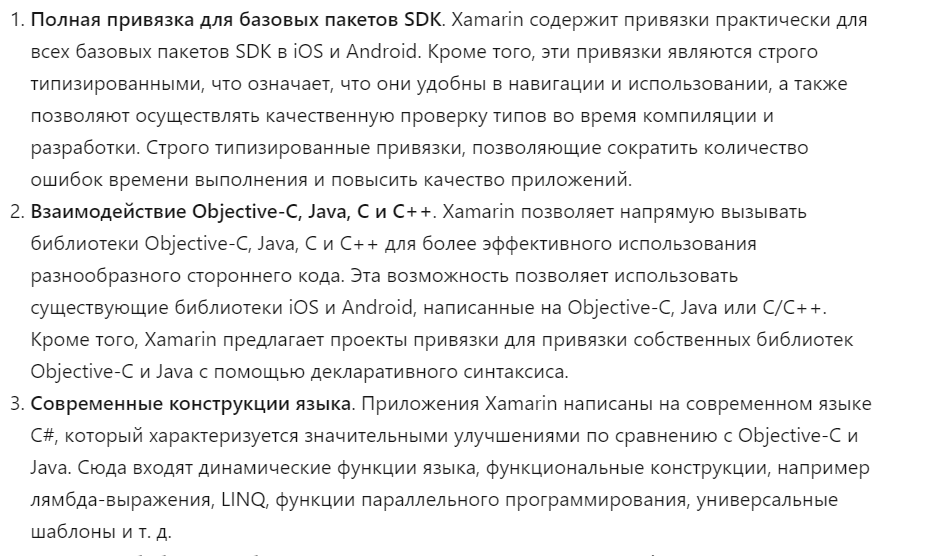


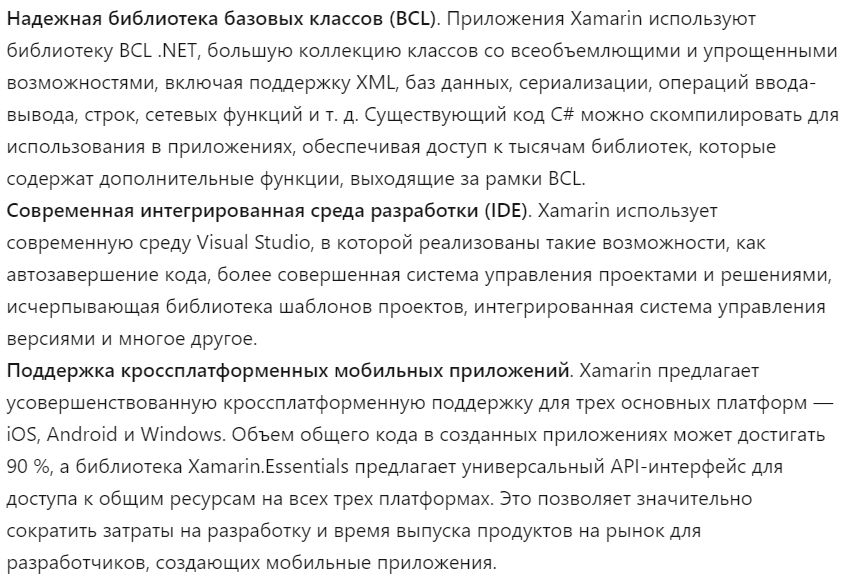
На этой схеме показана общая архитектура кроссплатформенного приложения Xamarin. С помощью Xamarin вы можете создавать собственный пользовательский интерфейс для каждой платформы и писать на языке C# общую бизнес-логику, которая будет использоваться на различных платформах. В большинстве случаев Xamarin позволяет использовать на разных платформах 80 % кода приложения. В основе Xamarin лежит среда .NET, которая автоматически обрабатывает такие задачи, как выделение памяти, сборка мусора и обеспечение взаимодействия с базовыми платформами.

//слайд 8

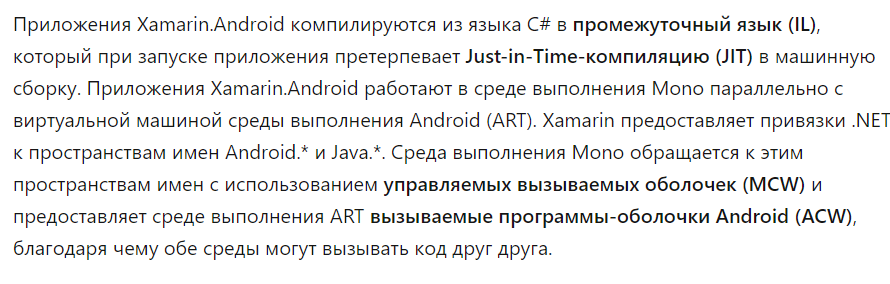
### **Добавленные компоненты**

Xamarin сочетает в себе возможности собственных платформ с добавлением возможностей, к которым относятся:





//слайд 9

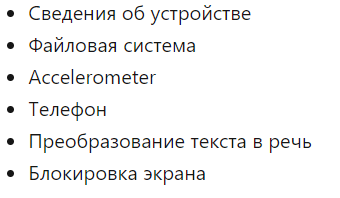


//слайд 10

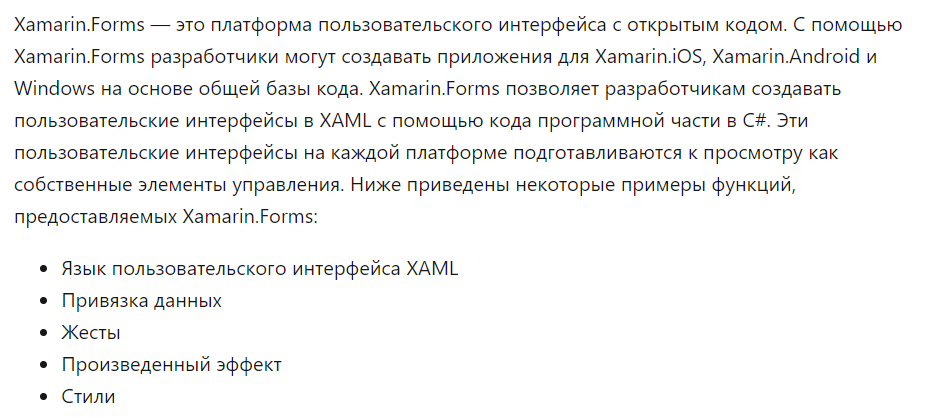
Приложения Xamarin.iOS проходят полную Ahead-of-Time-компиляцию (AOT) из языка C# в собственный код сборки ARM. Xamarin использует селекторы для предоставления кода Objective-C управляемому коду C# и Registrars для предоставления управляемого кода C# коду Objective-C. Селекторы и Registrars в совокупности называются "привязками" и обеспечивают взаимодействие между Objective-C и C#.

//слайд 11

Xamarin.Essentials — это библиотека, которая предоставляет кроссплатформенные API для собственных функций устройства. Как и сама платформа Xamarin, библиотека Xamarin.Essentials представляет собой абстракцию, которая упрощает процесс доступа к собственным функциям. Ниже приведены некоторые примеры функциональных возможностей, предоставляемых Xamarin.Essentials:



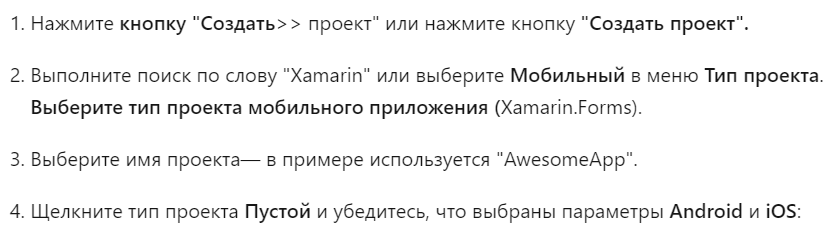
//слайд 12



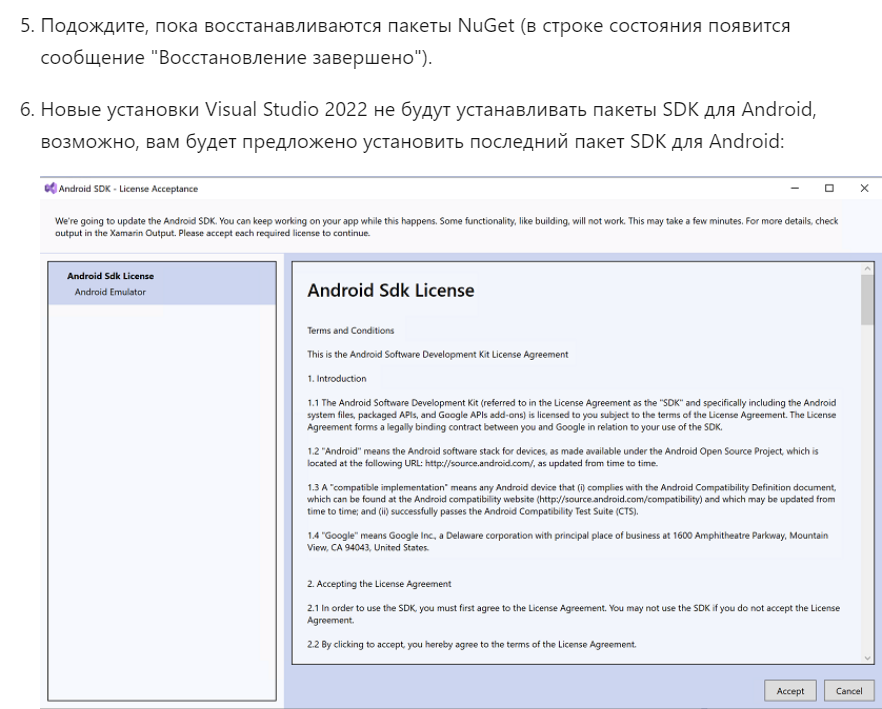
//слайд 13

Далее мы рассмотрим настройку среды, в которой мы будем делать приложение (Visual Studio 2022)

//слайд 14

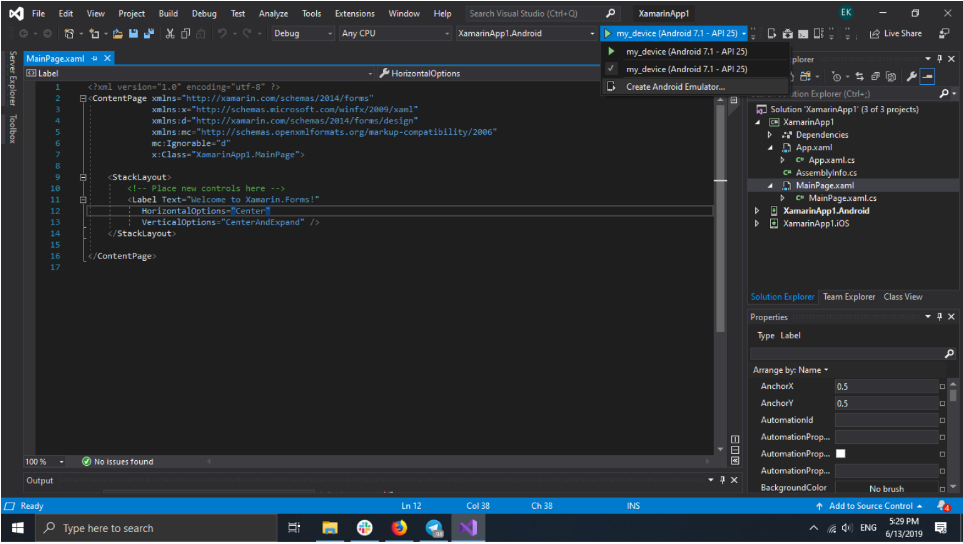


//слайд 15

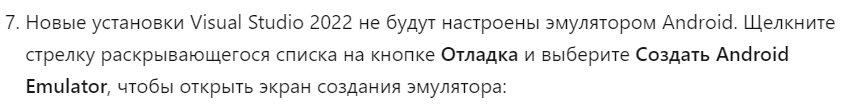


//слайд 16

Сам проект по началу будет выглядеть вот так:



//слайд 17



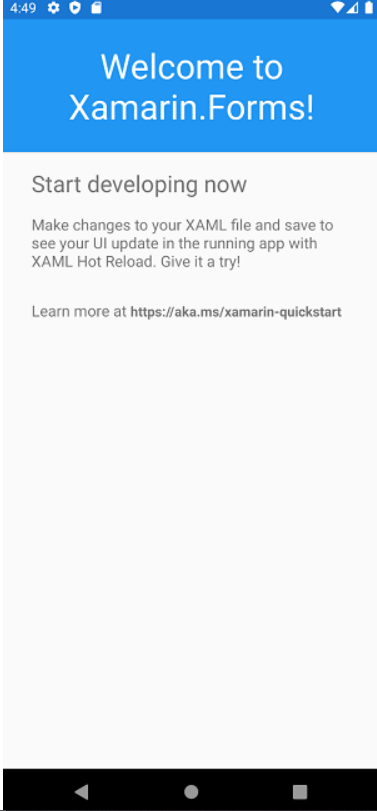


Запускаем эмулятор

//слайд 18

Нажимаем на кнопку Отладка, чтобы создать и развернуть приложение в эмуляторе Android. Вот что мы получим:

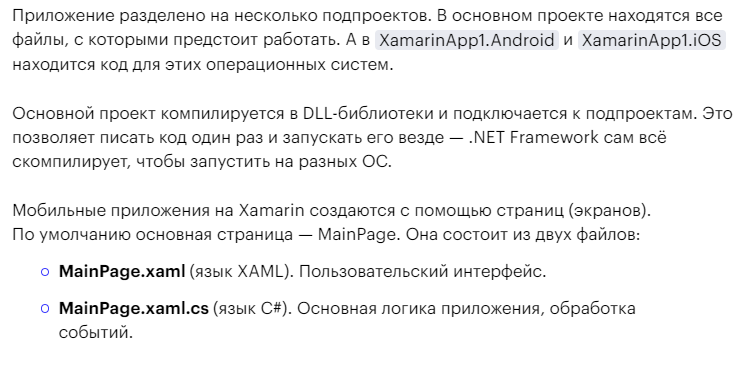
//слайд 19



//слайд 20

Дальше мы рассмотрим структуру проекта чтобы лучше понимать, что и с чем взаимодействует и что для чего надо

//слайд 21



//слайд 22 Начать можно с файла разметки интерфейса. Напишем простой калькулятор:

<?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:d="http://xamarin.com/schemas/2014/forms/design"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

x:Class="XamarinApp1.MainPage">

<StackLayout >

<Label Text="Calc" Margin="5" FontSize="Large" FontAttributes="Bold"/>

<StackLayout Orientation="Horizontal" Spacing="50" HorizontalOptions="CenterAndExpand">

<Entry x:Name="num1" WidthRequest="100" Keyboard="Numeric"/>

<Label Text="+"/>

<Entry x:Name="num2" WidthRequest="100" Keyboard="Numeric"/>

</StackLayout>

<Label Text="=" Margin="5" HorizontalTextAlignment="Center"/>

<Label x:Name="result" Margin="5" HorizontalTextAlignment="Center"/>

<Button Text="Calculate" Clicked="Button\_Clicked"/>

</StackLayout>

</ContentPage>

Здесь всё находится внутри элемента <ContentPage>, который является аналогом <Window> из [WPF](https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation" \t "_blank). Внутри него содержится <StackLayout>, который позволяет разместить элементы последовательно. С помощью атрибута Orientation указывается направление элементов — вертикальное (по умолчанию) или горизонтальное.

//слайд 23

Так будет выглядеть наше приложение на данный момент. Посмотреть вид приложения можно с помощью инструмента Designer.

Текстовые блоки создаются с помощью элемента <Label>, а поля ввода — с помощью <Entry>. Кнопка — с помощью элемента <Button>. Полям ввода указывается атрибут x: Name, чтобы к ним можно было обратиться в коде C#. А для кнопки указывается обработчик события Clicked.

//слайд 24 Теперь посмотрим, что происходит в коде MainPage.xaml.cs:

using System.Threading.Tasks;

using Xamarin.Forms;

namespace XamarinApp1

{

[DesignTimeVisible(false)]

public partial class MainPage : ContentPage

{

public MainPage()

{

InitializeComponent();

}

private void Button\_Clicked(object sender, EventArgs e) //Обработчик нажатия на кнопку

{

try

{

int n1 = Convert.ToInt32(num1.Text); //Конвертация значений из текстовых полей в числа

int n2 = Convert.ToInt32(num2.Text);

result.Text = (n1 + n2).ToString(); //Сложение полученных чисел, преобразование их обратно в строку и вывод в один из элементов <Label>

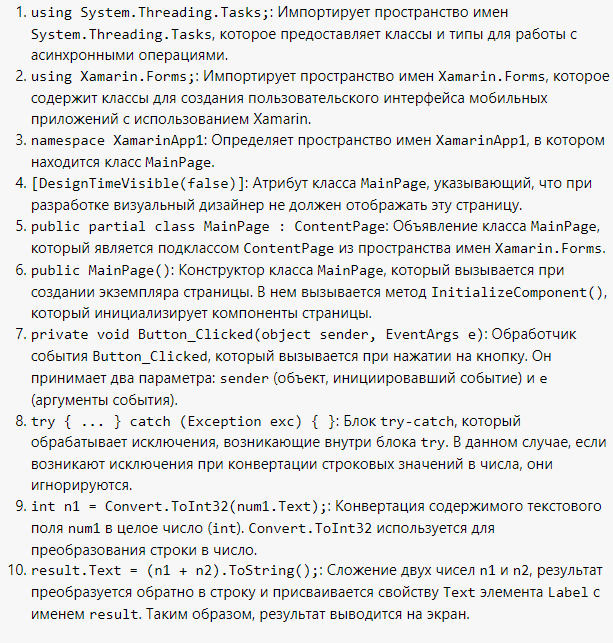
}

catch (Exception exc) { }

}

}

}



//слайд 24 В итоге мы получили вот такой калькулятор:

## //слайд 26 ****Заключение****

После того как приложение будет готово, его можно скомпилировать, чтобы потом передать на мобильное устройство для установки. Установочные пакеты надо искать в папках Bin → Debug необходимой платформы.