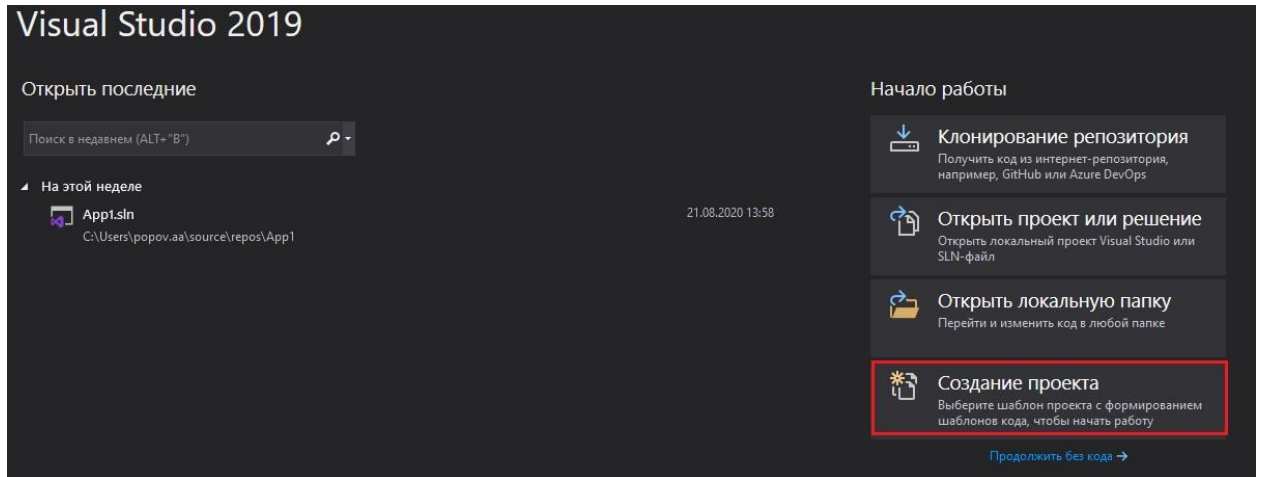


Создание проекта с использованием Xamarin

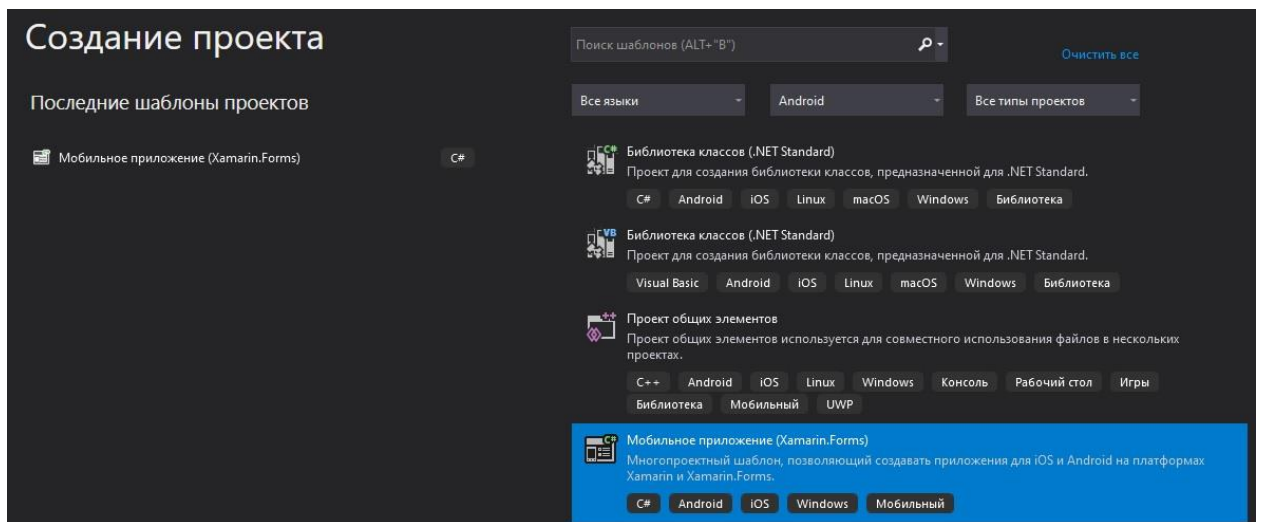
Данное занятие посвящено созданию пробного проекта с использованием платформы Xamarin.

Для создания проекта необходимо запустить Visual Studio.Net.

Далее необходимо нажать **Создание проекта**.



В диалоговом окне **Создание проекта** выбрать шаблон **Мобильное приложение (Xamarin.Forms)** и нажать кнопку «Далее»



Задать для проекта имя **Phoneword**, определить расположение проекта и нажать кнопку «Создать»

Настроить новый проект

Мобильное приложение (Xamarin.Forms) C# Android iOS Windows Мобильный

Имя проекта

Phoneword

Расположение

C:\Users\popov.aa\Desktop\

Имя решения ⓘ

Phoneword

☒ Поместить решение и проект в одном каталоге

Назад

Создать

В диалоговом окне «**Новое мобильное приложение**» нажать шаблон «**Пустой**» и нажать кнопку «**Создать**», чтобы создать новый проект

×

Новое мобильное приложение

Выберите шаблон приложения.

Всплывающий элемент

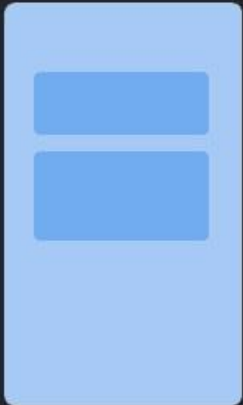
Приложение с боковым меню, которое можно свернуть на небольших экранах.

С вкладками

Приложение, которое использует вкладки для перехода между разделами.

Пустой

Пустое приложение с одним начальным экраном.



Я хочу разработать приложение для следующих платформ:

☒ Android

☒ iOS

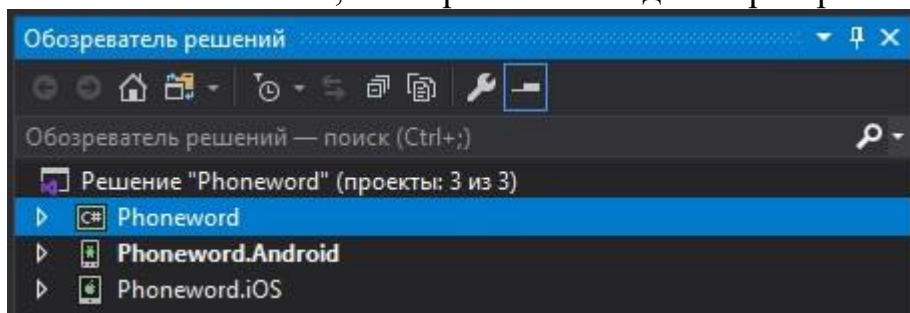
☐ Windows (UWP)

Назад

Создать

После создания проекта перейдите в **Обозревателе решений (Solution Explorer)**.

Мы можем заметить, что в решении созданы три проекта.



Все они имеют одно корневое имя, но у двух есть суффикс (".iOS" или ".Android").

Будем создавать одностраничное программное приложение, которое преобразует введенный пользователем буквенно-цифровую строку в числовую строку, которая отображается в пользовательском интерфейсе.

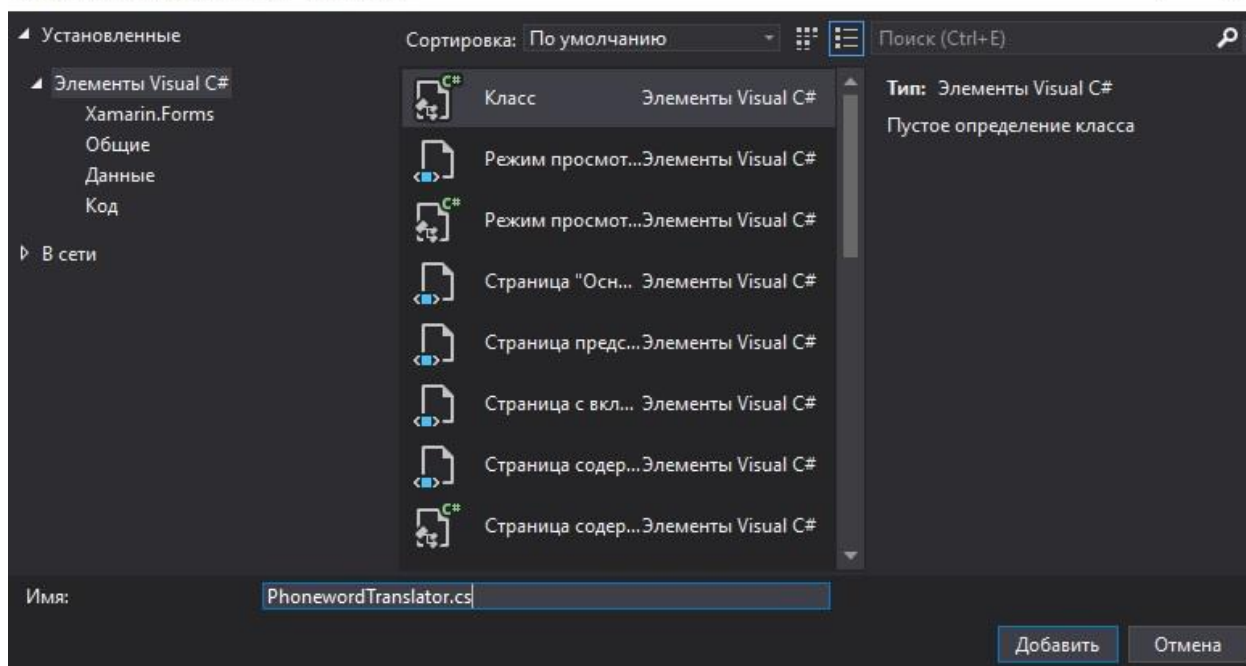
В качестве основы для перевода приложение будет использовать буквы, которые соответствуют цифрам, которые отображаются на клавиатуре кнопочного телефона.



Например, строка «СAB» должна быть преобразована в строку «222», так как цифре «2» на клавиатуре соответствуют все три буквы из строки «ABC».

Добавим в проект новый файл с программным кодом C#.

Для этого необходимо щелкнуть правой кнопкой мыши по проекту с именем Phoneword и выбрать **Добавить>Создать элемент** и в открывшемся диалоговом окне «Добавление нового элемента» выбрать шаблон «**Класс**», ввести имя класса PhonewordTranslator.cs и нажать кнопку «**Добавить**».



Создается пустой класс Си Шарп PhonewordTranslator.

Набрать внутри класса PhonewordTranslator следующий программный код:

```
public static class PhonewordTranslator
{
    public static string ToNumber(string raw)
    {
        // Проверка отсутствия строки
        if (string.IsNullOrEmpty(raw))
            return null;
        // Перевод символов строки в верхний регистр
        raw = raw.ToUpperInvariant();

        var newNumber = new StringBuilder();

        //Перебор символов строки raw
        foreach (var c in raw)
        {
            // Проверка значения признака
            // вхождения символа в "числовую" строку
            if ("0123456789".Contains(c))
            {
                // Если символ входит в строку, то производится
                // добавление символа к строке newNumber
                newNumber.Append(c);
            }
            else
            {
                // Если символ не входит в строку, то производится
                // обращение к функции перевода буквы в число
                var result = TranslateToNumber(c);
            }
        }
    }
}
```

```

        // проверка результата выполнения
        // функции TranslateToNumber
        if (result != null)
            // добавление цифрового символа
            // к строке newNumber вместо буквы
            newNumber.Append(result);

        else
            // результат не существует, если на вход
            // функции поступил символ, не являющийся цифрой
            // и не соответствующий массиву digits
            return null;
    }
}
return newNumber.ToString();
}

// Функция выполняет поиск символа в строке и
// возвращает признак вхождения (не вхождения)
// символа в строку
static bool Contains(this string keyString, char c)
{
    return keyString.IndexOf(c) >= 0;
}

// Массив для перевода букв в цифры
static readonly string[] digits = {
    "ABC", "DEF", "GHI", "JKL", "MNO", "PQRS", "TUV", "WXYZ"
};

// Функция перевода букв в цифры
static int? TranslateToNumber(char c)
{
    for (int i = 0; i < digits.Length; i++)
    {
        if (digits[i].Contains(c))
            return 2 + i;
    }
    return null;
}
}

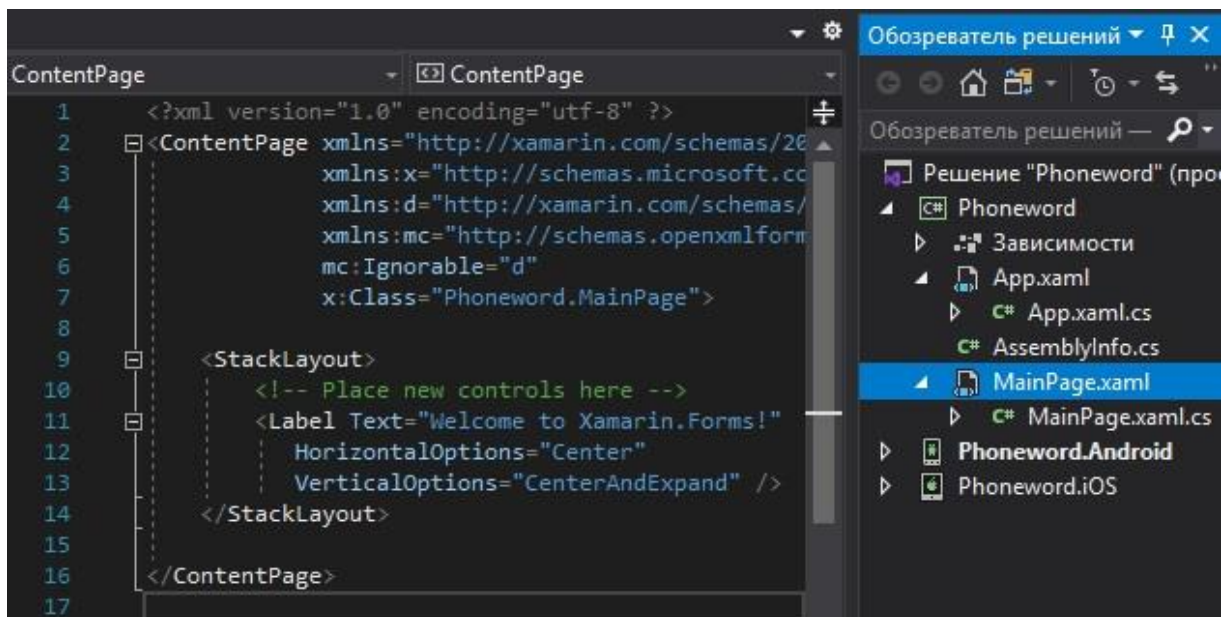
```

Сохранить набранный программный код.

Переделать программный код, находящийся в файле MainPage.xaml и соответствующий пользовательскому интерфейсу.

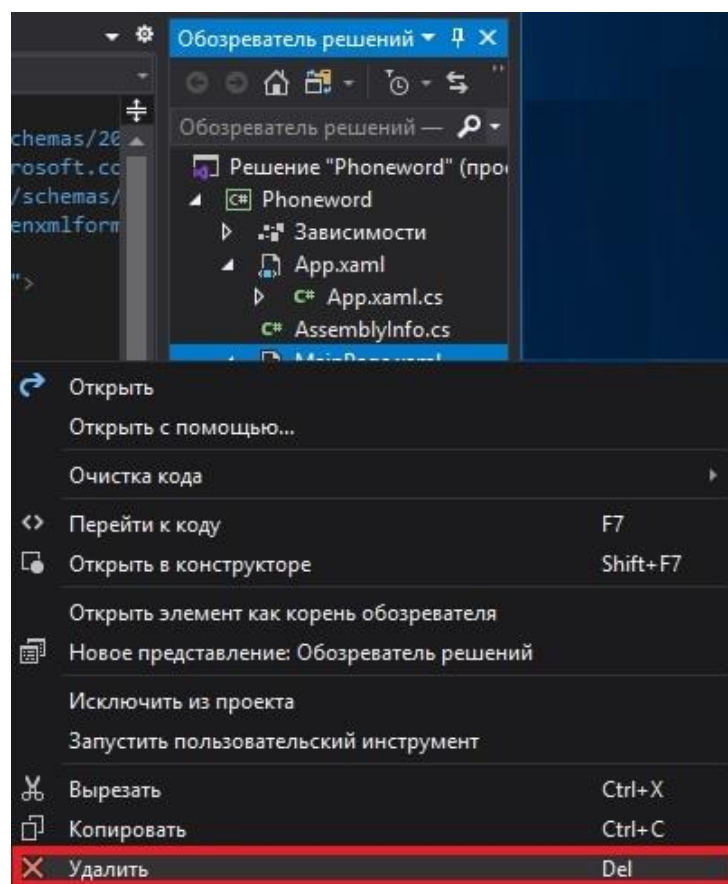
Создать свой собственный пользовательский интерфейс.

Для этого необходимо сначала удалить автоматически созданный и находящийся в файле MainPage.xaml программный код пользовательского интерфейса.



Для этого в обозревателе решений необходимо щелкнуть правой кнопкой мыши файл MainPage.xaml и выбрать **Удалить**.

Это действие также удаляет файл программного кода MainPage.xaml.cs.



Создайте класс в новом файле с именем MainPage.cs.

Можно использовать тот же процесс, который описан выше, чтобы создать новый файл класса.

Сделать класс производным от класса `ContentPage`. Добавьте инструкцию `using Xamarin.Forms`, так как `ContentPage` находится в пространстве имен `Xamarin.Forms`.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4  using Xamarin.Forms;
5
6  namespace Phoneword
7  {
8      ссылка: 1
8      class MainPage: ContentPage
9      {
10     }
11 }
```

Следующим шагом является добавление программного кода для подключения пользовательского интерфейса в созданном классе `MainPage.cs`.

Программное приложение использует макет `StackLayout`.

Для работы с ним потребуется создать его экземпляр и убедиться, что задано свойство ориентации, которое необходимо нам для размещения в макете элементов управления сверху вниз.

Если используется ориентация макета `Vertical`, то каждый элемент управления внутри макета заполняет всю ширину макета.

Чтобы макет программного приложения выглядел привлекательно, между его элементами должно оставаться свободное место.

Имеется несколько свойств, которые можно использовать для этого в зависимости от целей, которые решает программное приложение.

Каждый элемент управления имеет свойство «поле» `Margin`, которое учитывается в макетах для задания расстояния между элементами управления, размещенными внутри макета.

Все макеты имеют свойство `Padding`, благодаря которому все элементы управления внутри макета находятся на определенном расстоянии от границы макета.

Для свойства `Padding` страницы установлена значение 20 для всех полей.

Также параметром для задания свободного пространства в пользовательском интерфейсе, является свойство `Spacing` макета `StackLayout`. Это свойство задает пространство между всеми дочерними элементами `StackLayout`.

В программном коде установлено значение свойства `Spacing` макета `StackLayout`, равное 15.

Свойство дополняет собственное поле элемента управления `Margin`, и поэтому фактическим свободным пространством между элементами управления будет совокупное значение поля и интервала (если применены оба свойства).

Элементы управления в макет добавляются с использованием свойства `Children`.

В программном коде созданы и добавлены в макет следующие элементы управления для пользовательского интерфейса программного приложения:

1. Элемент управления `Label` со свойством `Text`, для которого задана вспомогательная надпись «Enter a Phoneword» (Введите номер-слово – буквенно-цифровую строку).

2. Элемент управления `Entry`, который позволяет вводить или редактировать буквенно-цифровую строку и служит для получения от пользователя входных данных. Внутри элемента управления `Entry` с помощью свойства `Text` инициализировано значение «1-855-XAMARIN». Пользователь может заменить исходный текст своим собственным, но предварительное заполнение элемента управления необходимо для тестирования программного приложения.

3. Элемент `Button`, запускающий программную логику для преобразования буквенно-цифровой строки в цифровую строку. Его свойству `Text` необходимо присвоить значение «Translate».

4. Второй элемент `Button` отображает цифровую строку, полученную в результате введенной ранее буквенно-цифровой строки. Его свойству `Text` необходимо присвоить значение «Call». Возможность нажатия на данный элемент `Button` должна быть по умолчанию отключена с помощью задания свойству `IsEnabled` значения `false`.

После построения макета `StackLayout` с расположенными внутри него элементами управления необходимо добавить макет в свойство `Content` страницы `ContentPage`.

В этом случае `MainPage` будет отображать содержимое `StackLayout`, используя правила расположения элементов управления, установленные в макете класса `StackLayout`, а также значения свойств элементов управления и макета.

Далее необходимо подключить события `Clicked` кнопки `Translate` для преобразования буквенной строки в цифровую.

Для этого используется обработчик события `OnTranslate`, который принимает в качестве параметров `object` и `EventArgs`.

Обработчик события должен преобразовать введенное в элементе управления `Entry` буквенно-числовую строку в числовую строку с использованием метода `PhonewordTranslator.ToNumber` ранее разработанного класса `PhonewordTranslator`.

Цифровая строка сохраняется в строке с именем `translatedNumber`.

Буквенно-цифровая строка для дальнейшего преобразования в цифровую строку получается из свойства `Text` элемента управления `Entry`.

Метод `PhonewordTranslator.ToNumber` возвращает `null`, если буквенно-цифровая строка не может быть преобразована в цифровую.

Далее добавим программный код, который предназначен для изменения свойства `Text` кнопки «Call» так, чтобы в ней отображалась цифровая строка, если она успешно преобразована из буквенно-цифровой строки.

Кнопка Call доступна или не доступна для нажатия в зависимости от успешности преобразования буквенно-цифровой строки в цифровую.

Если функция TranslateNumber возвращает null, то кнопка Call становится недоступной, а если преобразование произведено успешно — становится доступной.

Для того, чтобы делать кнопку доступной или недоступной для нажатия используется, как уже указывалось ранее, свойство IsEnabled.

Если не удастся преобразовать буквенно-цифровую строку (в ней содержатся символы, не относящиеся к цифрам и буквам), то преобразование произведено неудачно, и свойство Text кнопки (элемента управления Button2) сбрасывается до «Call», а в поле ввода элемента управления Entry появляется сообщение «Error String».

```
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;

namespace Phoneword
{
    public class MainPage: ContentPage
    {
        Entry phoneNumberText;
        Button translateButton;
        Button callButton;
        string translatedNumber;

        public MainPage()
        {
            this.Padding = new Thickness(20, 20, 20, 20);

            // Задание свойств макета
            StackLayout panel = new StackLayout
            {
                Spacing = 15,
                Orientation = StackOrientation.Vertical
            };

            // Добавление элементов управления в макет StackLayout
            panel.Children.Add(new Label
            {
                Text = "Enter a Phoneword:",
                FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label))
            });
            panel.Children.Add(phoneNumberText = new Entry
            {
                Text = "1-855-XAMARIN",
            });
            panel.Children.Add(translateButton = new Button
            {
```

```

        Text = "Translate"
    });
    panel.Children.Add(callButton = new Button
    {
        Text = "Call",
        IsEnabled = false,
    });

    // Обработка нажатия на кнопку Translate
    translateButton.Clicked += OnTranslate;

    // Добавление макета в страницу ContentPage
    this.Content = panel;
}

private void OnTranslate(object sender, EventArgs e)
{
    string enteredNumber = phoneNumberText.Text;
    translatedNumber =
        Phoneword.PhonewordTranslator.ToNumber(enteredNumber);

    if (!string.IsNullOrEmpty(translatedNumber))
    {
        callButton.IsEnabled = true;
        callButton.Text = "Call " + translatedNumber;
    }
    else
    {
        callButton.IsEnabled = false;
        callButton.Text = "Call";
        phoneNumberText.Text = "Error string";
    }
}
}
}

```

Далее необходимо собрать решение. Для этого необходимо нажать меню Сборка и в выпадающем меню выбрать **Собрать решение**. Если решение уже собиралось, а в программный код вносились изменения, то необходимо выбрать **Пересобрать решение**.

Добиться того, чтобы в диалоговом окне Вывод не было ошибок

```
Вывод
Показать выходные данные из: Сборка
1>----- Перестроение всех файлов начато: проект: Phoneword, Конфигурация: Release Any CPU -----
1>Phoneword -> C:\Users\Popov.aa\Desktop\Phoneword\Phoneword\bin\Release\netstandard2.0\Phoneword.dll
2>----- Перестроение всех файлов начато: проект: Phoneword.Android, Конфигурация: Release Any CPU -----
Соединение для связывания с Mac не установлено, поэтому сборка будет проведена автономно. Установите соединение и повт
3>----- Перестроение всех файлов начато: проект: Phoneword.iOS, Конфигурация: Release iPhone -----
3> Executing SayHello Task to establish a connection to a Mac Server.
3> Properties: SessionId=c9862f5e9a7ecd3710da6cc023d2a18f,
3> ServerPort=,
3> ServerAddress=,
3> ServerUser=,
3> ServerPassword=,
3> SSHKey=,
3> SSHPassPhrase=,
3> AppName=Phoneword.iOS,
3> ContinueOnDisconnected=True
3>C:\Program Files (x86)\Microsoft Visual Studio\2019\Professional\MSBuild\Xamarin\iOS\Xamarin.Messaging.targets(56,3)
3> Phoneword.iOS -> C:\Users\Popov.aa\Desktop\Phoneword\Phoneword.iOS\bin\iPhone\Release\Phoneword.iOS.exe
2> Phoneword.Android -> C:\Users\Popov.aa\Desktop\Phoneword\Phoneword.Android\bin\Release\Phoneword.Android.dll
===== Перестроение всех проектов: успешно: 3, с ошибками: 0, пропущено: 0 =====
```

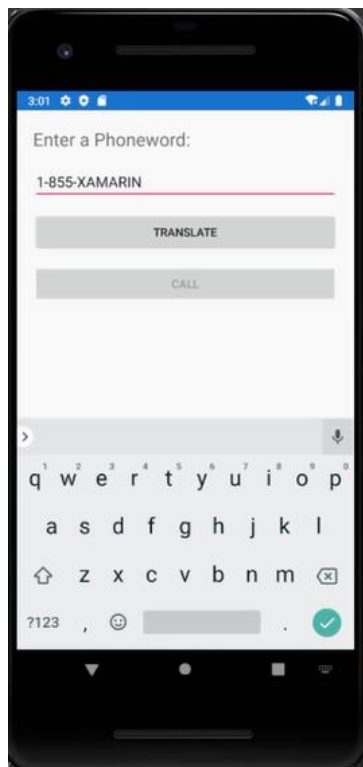
Просмотреть видео, в котором показана работа программного приложения можно, запустив имеющийся видеофайл.

Необходимо протестировать программное приложение, запустив его на настроенном ранее устройстве или эмуляторе Android.

Далее необходимо нажать на кнопку «TRANSLATE» для преобразования буквенно-цифровой строки «1-855-XAMARIN» в цифровую строку.

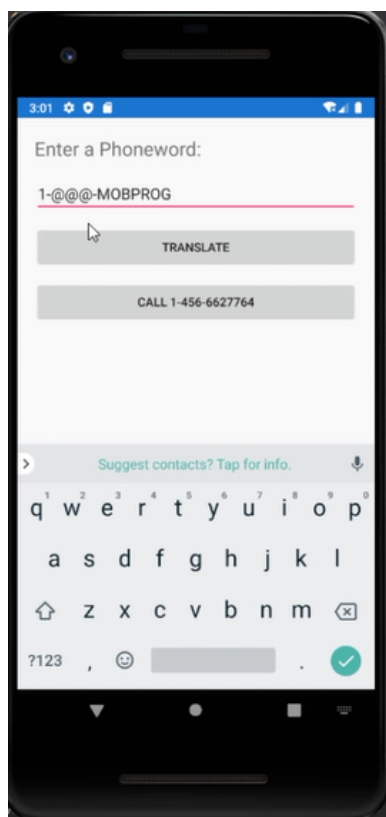
В кнопке с текстом «Call» отображается текстовая строка, состоящая из «Call» и полученной цифровой строки.

При этом нижняя кнопка становится доступной для нажатия





Теперь в элементе управления Entry необходимо ввести «некорректную» буквенно-цифровую строку «1-@@@-MOBPROG», содержащую три символа, не относящихся ни к буквам, ни к цифрам.



Далее необходимо нажать на кнопку с текстом «TRANSLATE» для преобразования буквенно-цифровой строки «1-@@@-MOBPROG» в

цифровую строку. В этом случае преобразование «некорректной» буквенно-цифровой строки будет произведено неудачно, и значению свойства Text нижней присваивается значение «Call», она становится недоступной для нажатия, а в поле ввода Entry появляется строка «Error String» (рис.).

