Mobile Programming and Multimedia

The Xamarin Framework

Prof. Ombretta Gaggi
University of Padua

Introduction

Xamarin is a cross-platform framework based on different approaches:

- Interpreted approach for Android and Windows
- Compiled approach for iOS

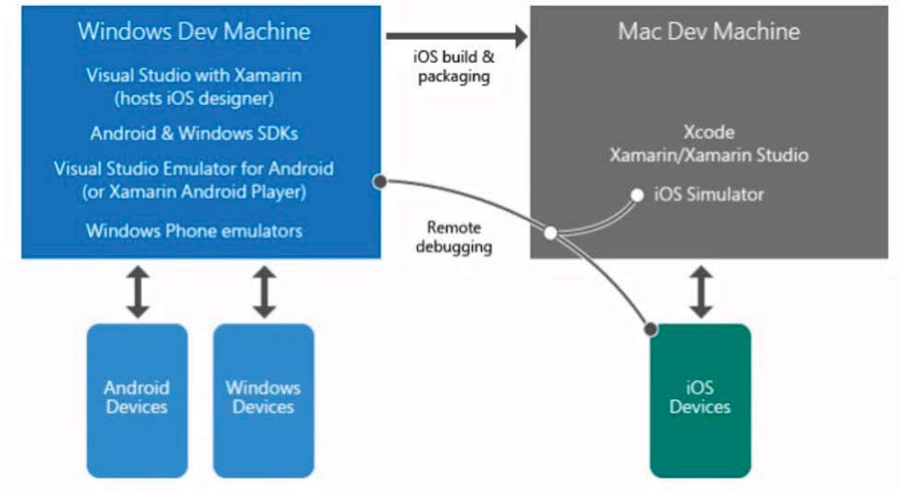
It is a project started in 2014 from Xamarin, a company based in California acquired by Microsoft in 2016

It is a general-purpose framework based on different parts

- Xamarin.Forms
- Xamarin Native
- XAML: XML language for interface building

Based on C#

System architecture



Widely used

How to build an app

Università degli Studi di Padova

Two possibilities available:

- Xamarin Native: allows writing code for a single platform (especially for the interface). Allows to use native APIs
- Xamarin.Forms: provides a set of APIs that can be used for each platform but with a native *look&feel*

To develop applications using Xamarin, it is necessary to install Microsoft Visual Studio

Which one is the best approach? - 1

Xamarin allows several choices:

- C# or XAML
- Native vs. general purpose

Which one is the best approach?

Xamarin.Form is more appropriate for:

- Apps that do not require functionalities specific to the platform
- When it is more important to reuse code instead of interface personalization
- If XAML is already known

Which one is the best approach? - 2

Xamarin.iOS & Xamarin.Android are more appropriate for:

- Apps that require native interactions (native *look&feel* is essential)
- Apps that make abundant use of native APIs
- When personalization of the interface is more important than code reuse with all the platforms

Xamarin.Forms

Projects

It is the combination of two projects, Xamarin.iOS e Xamarin.Android

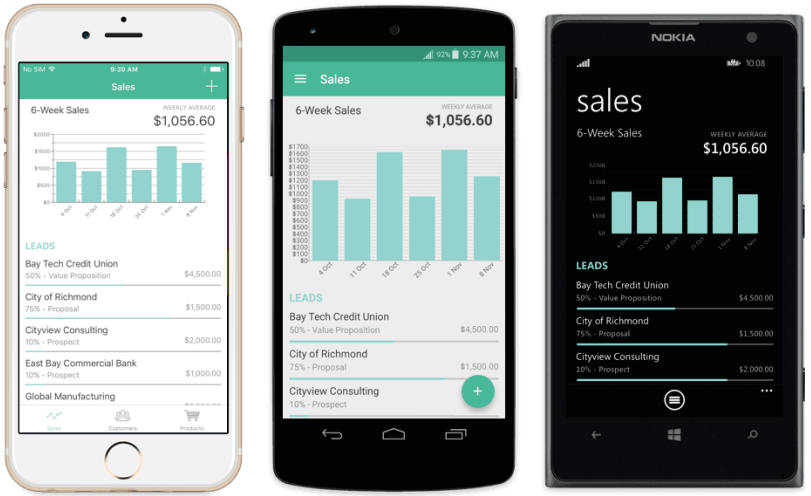
Xamarin.Forms is strongly focused on interfaces, than can be visualized equally everywhere

Uses the MVVM model

Allows a fast prototypization

An interface can be developed using C# or XAML (eXstensible Application Markup Language)

Same but different interfaces



Native Look&feel

Xamarin.Forms allows writing the same code for the interfaces of all platforms

Each page and each component are mapped in a specific widget specific for each platform at runtime.

For example, a Xamarin.Forms entry becomes:

- A UITextView on iOS
- A EditText on Android
- A TextBox on Windows

C#

Basic types

Type	Description
bool	true/false
byte	Positive integer, 8 bits
char	characters, 8 bits
int	4 bytes
short	2 bytes
float	4 bytes
double	8 bytes
object	Basic type
string	Sequence of characters

Variables

Variables names must have at least one character, cannot start with a number, and cannot have spaces

string text = "Hello" + "world!"

Array

In C# arrays are defined with [] and can contain only data of the same type

Arrays can contain other arrays

```
int [] grades;
grades = new int[5];
grades[0] = 18;
int numbers [];
numbers = new int[5] {1, 2, 3, 4, 5}
Console.WriteLine(numbers.Lenght);
Console.ReadLine();
```


List

Lists are objects.

Constructor:

- List<type> name = new List<type>();

```
List<string> vegetables = new List<string>();
vegetables.Add(”carrots");
vegetables.Add("zucchini");
vegetables.Remove("zucchini");
```

vegetables.AddRange(otherList) → concatenation

Dictionaries

Dictionaries in C# are associative arrays. Each value has an associated key

```
Dictionary<string, long> numbers = new
                        Dictionary<string, long>();
numbers.Add("Giorgio", 3481111);
numbers["Giovanni"] = 3482222;
numbers.Remove("Giorgio");
numbers.Count;
```


If … then … else

```
if (condition) {
    then instructions
} else {
    else instructions
}
```


Loops

```
while (condition){
    instructions
}

do{
    instructions
} while (condition)
```

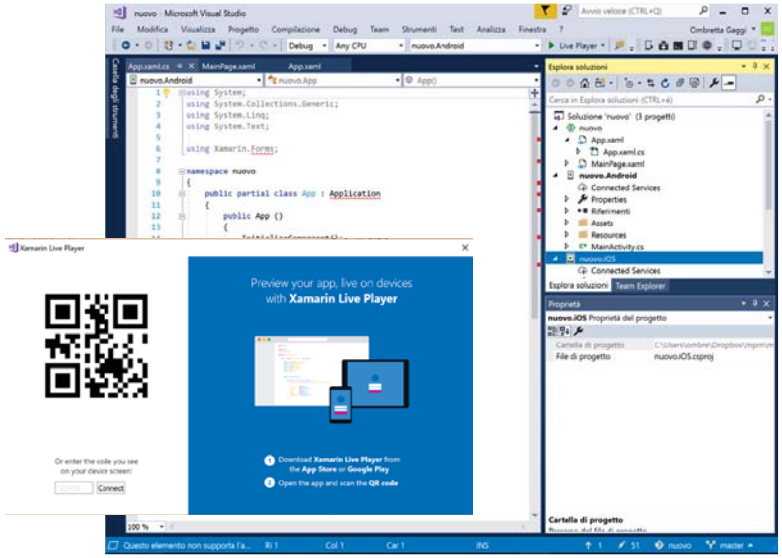
```
for(init; condition; increment){
    instructions
}

foreach (string day in days) {
    MessageBox.Show(day)
}
```

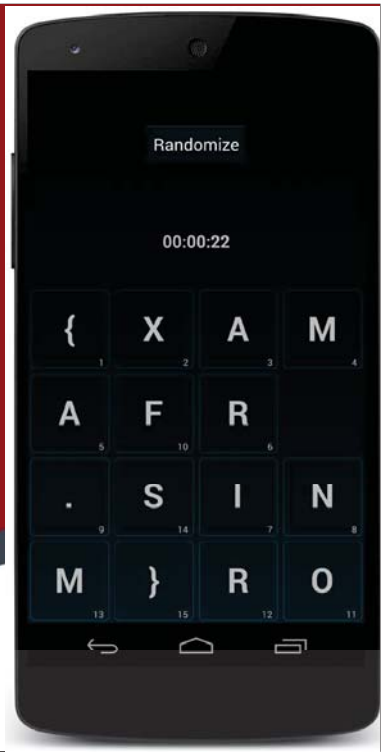

Switch

```
switch(expression){
    case A:
        instructions
        break;
    case B:
        instructions
        break;
    default:
        instructions
}
```

Tools and file management



A simple example



App.cs

```
using System;
using Xamarin.Forms;

namespace Xuzzle{
    public class App : Application{
        public App (){
         MainPage = new XuzzlePage();
       }
   }
}
```


Classe XuzzlePage – main page

```
using System; using System.Threading.Tasks; using Xamarin.Forms;
namespace Xuzzle{
      class XuzzlePage : ContentPage{ //variables definition
            static readonly int NUM = 4;
            XuzzleSquare[,] squares = new XuzzleSquare[NUM, NUM];
            int emptyRow = NUM - 1;  int emptyCol = NUM - 1;
            StackLayout stackLayout;
            AbsoluteLayout absoluteLayout;
            Button randomizeButton;
            Label timeLabel;
            double squareSize;
            bool isBusy;
            bool isPlaying;
            //functions definition
}
```

Constructor

```
public XuzzlePage (){
    // AbsoluteLayout to draw the puzzle
    absoluteLayout = new AbsoluteLayout () {
        HorizontalOptions = LayoutOptions.Center,
        VerticalOptions = LayoutOptions.Center
    };
    // strings
    string text = "{XAMARIN.FORMS}";
    string winText = "CONGRATULATIONS";
    int index = 0;
    …
 }
```


Drawing the interface - 1

```
for (int row = 0; row < NUM; row++) {
    for (int col = 0; col < NUM; col++) {
        if (row == NUM - 1 && col == NUM - 1) break; //do not fill the last square
        XuzzleSquare square = new XuzzleSquare (text [index], winText [index], index) {
            Row = row, //initialization and draw of each  card
            Col = col
        }; // Add tap recognition
        TapGestureRecognizer tapGestureRecognizer = new TapGestureRecognizer {
            Command = new Command (OnSquareTapped),
            CommandParameter = square
        };
        square.GestureRecognizers.Add (tapGestureRecognizer);
        // adding to tile array and  to absoluteLayout for visualization
        squares [row, col] = square;
        absoluteLayout.Children.Add (square);
        index++;
}}
```


Botton for repositioning

```
randomizeButton = new Button {
    Text = "Randomize",
    HorizontalOptions = LayoutOptions.Center,
    VerticalOptions = LayoutOptions.CenterAndExpand
};
randomizeButton.Clicked += OnRandomizeButtonClicked;
```


Text for timer

```
timeLabel = new Label {
        FontSize = Device.GetNamedSize (NamedSize.Large,
                                        typeof(Label)),
        FontAttributes = FontAttributes.Bold,
        HorizontalOptions = LayoutOptions.Center,
        VerticalOptions = LayoutOptions.CenterAndExpand
};
```

Drawing the interface - 2

```
stackLayout = new StackLayout {
                Children = {
                    new StackLayout {
                            VerticalOptions = LayoutOptions.FillAndExpand,
                            HorizontalOptions = LayoutOptions.FillAndExpand,
                            Children = {
                                randomizeButton,
                                timeLabel
                            }
                    },
                    absoluteLayout
                }
};
stackLayout.SizeChanged += OnStackSizeChanged; //insert into the page
this.Padding = new Thickness(0, Device.RuntimePlatform == Device.iOS ? 20 : 0, 0, 0);
this.Content = stackLayout;
//end of XuzzlePage() constructor
```


OnStackSizeChanged

```
void OnStackSizeChanged (object sender, EventArgs args){
    double width = stackLayout.Width;
    double height = stackLayout.Height;
    if (width <= 0 || height <= 0) return;
    // check landscape or portrait
    stackLayout.Orientation = (width < height) ?
                                StackOrientation.Vertical:StackOrientation.Horizontal;
    // calculating position and size of each card based on screen size
    squareSize = Math.Min (width, height) / NUM;
    absoluteLayout.WidthRequest = NUM * squareSize;
    absoluteLayout.HeightRequest = NUM * squareSize;
    foreach (View view in absoluteLayout.Children) {
        XuzzleSquare square = (XuzzleSquare)view;
        square.SetLabelFont (0.4 * squareSize, FontAttributes.Bold);
        AbsoluteLayout.SetLayoutBounds (square,
            new Rectangle (square.Col * squareSize,
                           square.Row * squareSize,  squareSize, squareSize));
    }
}
```


OnSquareTapped

```
async void OnSquareTapped (object parameter){
    if (isBusy) return;
    isBusy = true;
    XuzzleSquare tappedSquare = (XuzzleSquare)parameter;
    await ShiftIntoEmpty (tappedSquare.Row, tappedSquare.Col);
    isBusy = false; //check if player wins
    if (isPlaying) {
        int index;
        for (index = 0; index < NUM * NUM - 1; index++) {
            int row = index / NUM; int col = index % NUM;
            XuzzleSquare square = squares [row, col];
            if (square == null || square.Index != index) break;
        } // win
        if (index == NUM * NUM - 1) {
            isPlaying = false;
            await DoWinAnimation ();
        }
}}
```


ShiftIntoEmpty

```
async Task ShiftIntoEmpty (int tappedRow, int tappedCol, int length = 100)
    if (tappedRow == emptyRow && tappedCol != emptyCol) {// Shift columns
        int inc = Math.Sign (tappedCol - emptyCol);
        int begCol = emptyCol + inc;
        int endCol = tappedCol + inc;
        for (int col = begCol; col != endCol; col += inc) {
            await AnimateSquare (emptyRow, col, emptyRow, emptyCol, length);
        }       // Shift rows
    } else if (tappedCol == emptyCol && tappedRow != emptyRow) {
        int inc = Math.Sign (tappedRow - emptyRow);
        int begRow = emptyRow + inc;
        int endRow = tappedRow + inc;
        for (int row = begRow; row != endRow; row += inc) {
            await AnimateSquare (row, emptyCol, emptyRow, emptyCol, length);
        }
    }
}
```

AnimateSquare

```
async Task AnimateSquare (int row, int col, int newRow, int newCol, int length){
    XuzzleSquare animaSquare = squares [row, col]; // card to animate
    //destination rectangle
    Rectangle rect = new Rectangle (squareSize * emptyCol,
                                    squareSize * emptyRow,  squareSize,  squareSize);
    await animaSquare.LayoutTo (rect, length);
    //variables for the new  layout
    squares [newRow, newCol] = animaSquare;
    animaSquare.Row = newRow;
    animaSquare.Col = newCol;
    squares [row, col] = null;
    emptyRow = row;
    emptyCol = col;
}
```


OnRandomizeButtonClicked - 1

```
async void OnRandomizeButtonClicked (object sender, EventArgs args) {
    Button button = (Button)sender;
    button.IsEnabled = false;
    Random rand = new Random ();
    isBusy = true;
    // Simulate some fast crazy taps
    for (int i = 0; i < 100; i++) {
        await ShiftIntoEmpty (rand.Next (NUM), emptyCol, 25);
        await ShiftIntoEmpty (emptyRow, rand.Next (NUM), 25);
    }
    button.IsEnabled = true;
    isBusy = false;
    …
}
```


OnRandomizeButtonClicked - 2

```
async void OnRandomizeButtonClicked (object sender, EventArgs args) {
    … // preparation of the game
    DateTime startTime = DateTime.Now;
    Device.StartTimer (TimeSpan.FromSeconds (1), () => {
        // Round duration and get rid of milliseconds.
        TimeSpan timeSpan = (DateTime.Now - startTime) + TimeSpan
                                    .FromSeconds(0.5);
        timeSpan = new TimeSpan (timeSpan.Hours, timeSpan.Minutes,
                                    timeSpan.Seconds);
        if (isPlaying) // shows the duration
            timeLabel.Text = timeSpan.ToString ("t");
        return isPlaying;
    });
    this.isPlaying = true;
}
```


DoWinAnimation

```
async Task DoWinAnimation (){
    //blocking input
    randomizeButton.IsEnabled = false;
    isBusy = true;
    for (int cycle = 0; cycle < 2; cycle++) {
        foreach (XuzzleSquare square in squares)
            if (square != null)
                await square.AnimateWinAsync (cycle == 1);
            if (cycle == 0)
                await Task.Delay (1500);
    }
    //restarting input
    randomizeButton.IsEnabled = true;
    isBusy = false;
}
```

XuzzleSquare

```
using System;
using System.Threading.Tasks;
using Xamarin.Forms;

namespace Xuzzle{
    class XuzzleSquare : ContentView{
        Label label;
        string normText, winText;
        //constructor and functions
        // current position
        public int Index { private set; get; }
        public int Row { set; get; }
        public int Col { set; get; }
    }
}
```


Constructor - 1

```
public XuzzleSquare (char normChar, char winChar, int index){
    this.Index = index;
    this.normText = normChar.ToString ();
    this.winText = winChar.ToString ();
    // each card is a frame with two labels
    label = new Label {
        Text = this.normText,
        HorizontalOptions = LayoutOptions.Center,
        VerticalOptions = LayoutOptions.CenterAndExpand
    };
    Label tinyLabel = new Label {
        Text = (index + 1).ToString (),
        FontSize = Device.GetNamedSize (NamedSize.Micro, typeof(Label)),
        HorizontalOptions = LayoutOptions.End
    };
    …
}
```


Constructor - 2

```
public XuzzleSquare (char normChar, char winChar, int index){
    …
    this.Padding = new Thickness (3);
    this.Content = new Frame {
    OutlineColor = Color.Accent,
    Padding = new Thickness (5, 10, 5, 0),
        Content = new StackLayout {
            Spacing = 0,
            Children = {
                label,
                tinyLabel,
            }
        }
    };
    // blocks touch event that is managed by this object and not by the ones below
    this.BackgroundColor = Color.Transparent;
}
```


AnimateWinAsync

```
public async Task AnimateWinAsync (bool isReverse) {
    uint length = 150;
    await Task.WhenAll (this.ScaleTo (3, length),
                        this.RotateTo (180, length));
    label.Text = isReverse ? normText : winText;
    await Task.WhenAll (this.ScaleTo (1, length),
    this.RotateTo (360, length));
    this.Rotation = 0;
}
```

SetLabelFont

```
public void SetLabelFont(double fontSize,
                         FontAttributes attributes){
    label.FontSize = fontSize;
    label.FontAttributes = attributes;
}
```


References

Official site

- https://www.xamarin.com/

Documentation

- https://developer.xamarin.com/guides/

Puzzle example

- https://developer.xamarin.com/samples/xamarin-forms/Xuzzle/