

A

Seminar Report

On

“Analysis of Cross – Platform App Development Tools”

Kadtan Soham Raju

Roll no: 3159

Class: T.E. (A)

Seminar Guide

Prof.S.S. Varpe

For the partial fulfillment of
Seminar work in T.E. Computer Engineering



Department Of Computer Engineering
Amrutvahini College of Engineering, Sangamner (422605)

2020 - 21

Department of Computer Engineering
Amrutvahini College of Engineering, Sangamner



CERTIFICATE

This is to certify that **Mr. Kadtan Soham Raju** student in Third Year Division A, Computer Engineering has successfully completed his Seminar titled **“Analysis of Cross - Platform App Development Tools”** at Amrutvahini College of Engineering, Sangamner towards partial fulfillment of Seminar Work in Computer Engineering.

Prof S.S. Varpe
Seminar Guide

Prof. Pooja Walunj
Seminar Coordinator

Prof. R.L Paikrao
Head of Department

Dr.M.A Venkatesha
principal

Abstract

With the emergence of different cross-platform alternatives, there is a need to explore the various approaches the developer must take. We perform this study in the hopes of shedding lighter on their differences and highlighting the critical aspects which make them unique. The examination of these WORA (Write Once, Run Anywhere) mobile App tools is done on the basis of different approaches, i.e., Native apps, Web apps, Hybrid apps, Interpreted apps and Widget-based apps. The study performed shall enable us to illustrate the results on the basis of these categories.

Acknowledgments

I want take this chance to thank almighty for blessing me with his grace and taking my job to a successful culmination. I extend my sincere and heartfelt thanks to my esteemed seminar guide, Prof.S.S. Varpe Sir, for providing me with the important guidance and advice at the crucial junctures, while preparing project report and also for guiding me towards right way. I expand my sincere thanks to my Seminar Coordinator Prof. Pooja Walunj Mam, Head of Department prof.R.L. Paikrao Sir, In end, I'd also thankful to Respected Principle Dr.M.A. Venkatesh Sir for the support and encouragement they have given me throughout the course of my work.

Contents

Abstract	3
Acknowledgments	4
Contents	5
List of Figures	6
Abbreviations	7
Introduction	8
Literature Survey	18
Software Requirement	19
System Architecture	21
Application	36
Advantages	37
Disadvantages	38
Conclusion	39
Reference	40

List of Figures

1.1 An introduction to Cross platform Mobile apps	8
1.2 Dart Language Syntax	10
1.3 Kotlin Language syntax	15
1.4 Ruby Language syntax	16
3.1 React Native System Architecture	21
3.2 Flutter System Architecture	23
3.4 Cordova System Architecture	24
3.5 Ionic System Architecture	26
3.6 Xamarin System Architecture	27
3.7 Codename One System Architecture	28
3.8 Ruby Motion System Architecture	29
3.9 QT System Architecture	30
4.1 Model-View-Controller	33
4.2 Model-View-Presenter	34
4.3 Model-View-View-Model	35

Abbreviations

API	Application Programming Interface
APK	Android Package
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
JS	JavaScript, an interpreted Programming language
MVC	Model-View-Controller
MVP	Model-View-Presenter
MVVM	Model-View-View-Model
GUI	Graphical User Interface
SDK	Software Development Kit

Chapter 1

Introduction

Cross – Platform application development is about building a single application that can run on various operating systems, instead of developing different app versions for each platform.

Types of frameworks: -

Apps are made on the basis of different types of frameworks and by the help of these frameworks developers make their own applications.

For Example: -

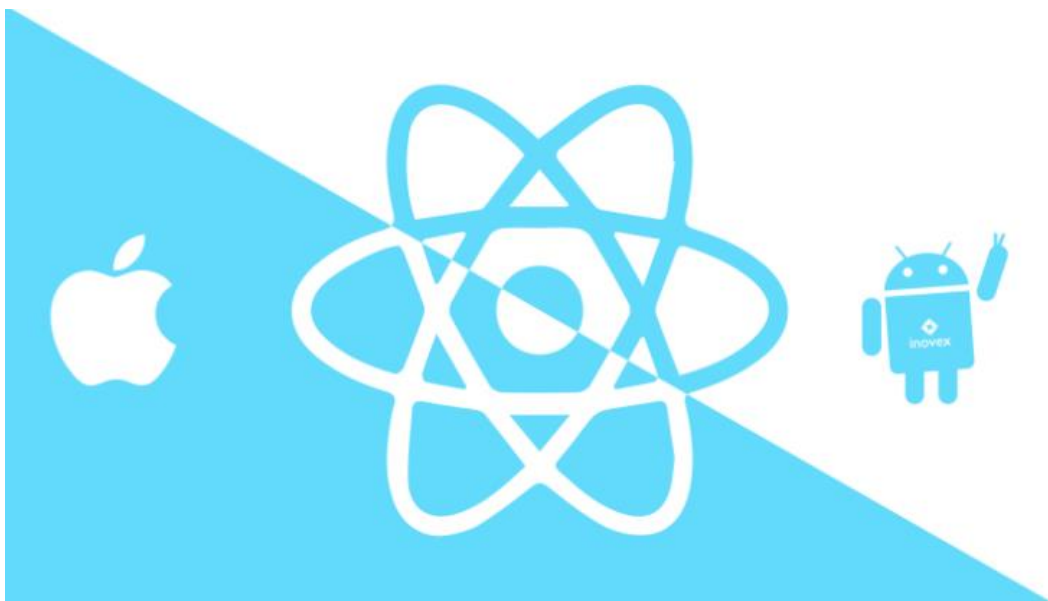
- 1) React Native
 - 2) Flutter
 - 3) Xamarin
 - 4) Cordova
 - 5) Ionic
- and many more....

An Introduction to Cross-Platform Mobile Apps



1) React Native: -

- i) This framework is made by the company name Facebook.
- ii) This framework is based on JavaScript programming language which used to developers to make apps on both iOS and android.
- iii) It is also known as RN.



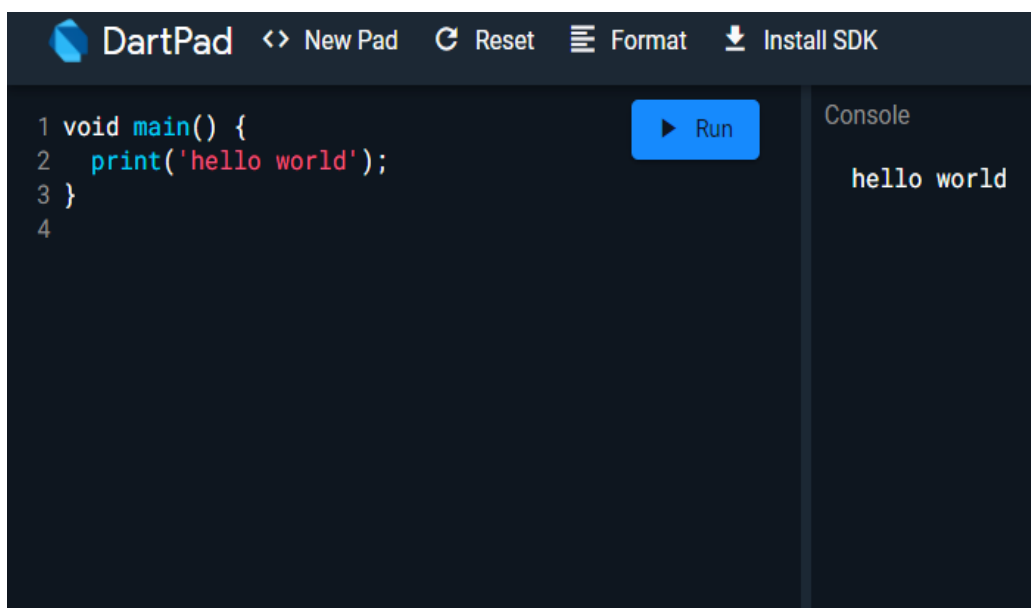
iv) developers can now write mobile applications that look and feel truly “native,” all from the comfort of a JavaScript library that we already know and love. Plus, because most of the code you write can be shared between platforms, React Native makes it easy to simultaneously develop for both Android and iOS.

2) Flutter: -

- i) This framework is made by the company name Google.
- ii) This framework is based on the dart programming language.



- This is the syntax of dart language: -

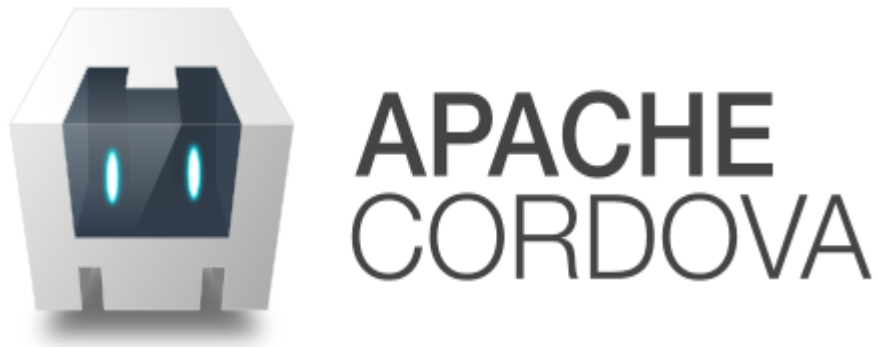
A screenshot of the DartPad web IDE interface. The top toolbar includes the DartPad logo, a "New Pad" button, a "Reset" button, a "Format" button, and an "Install SDK" button. The main editor area contains the following Dart code:

```
1 void main() {  
2   print('hello world');  
3 }  
4
```

A blue "Run" button is positioned to the right of the code. On the right side of the interface, there is a "Console" panel displaying the output "hello world".

3) Cordova: -

- i) Apache Cordova is also named as PhoneGap which is created by Nitobi company.
- ii) This framework is based on the web-based languages called HTML, CSS, JavaScript which helps developer to make their own applications.



- iii) Cordova framework will help to make mobile application and as well as web-based application.

4) Ionic: -

- i) Ionic framework made by drifty company in the year 2013.
- ii) Ionic transforms a single code written in angular.js, HTML into a mobile app.
- iii) Angular.js, CSS, HTML, JS these programming are used in ionic to build applications.



- iv) Ionic uses Cordova and, more recently, Capacitor plugins to gain access to host operating systems features such as Camera, GPS, Flashlight, etc....

5) Xamarin: -

- i) Xamarin is an open-source cross platform UI framework to build apps for various platforms like iOS, android and Windows.
- ii) Xamarin framework written in .NET programming language.



- iii) It builds native apps with high performance and native UI.

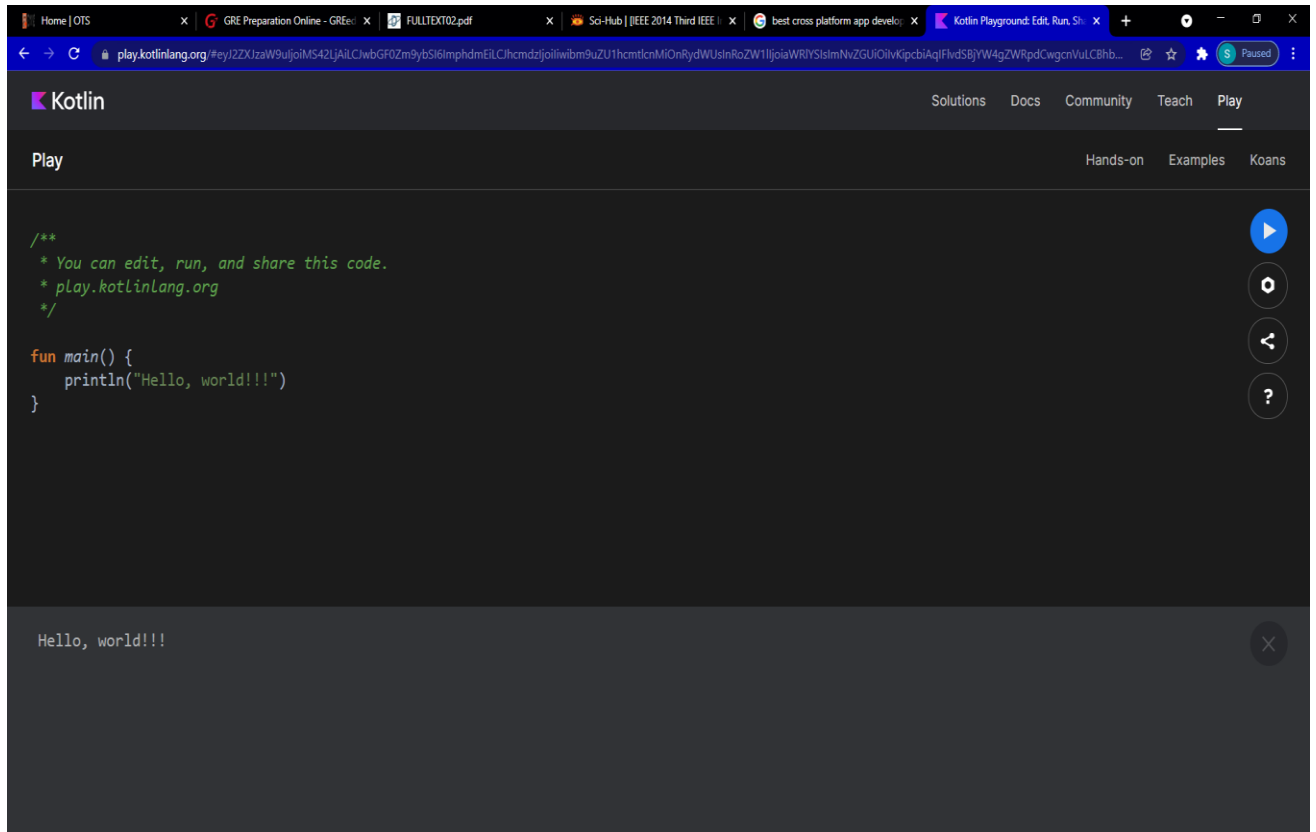
6) Codename One: -

- i) Codename One is an open-source cross-platform framework aiming to provide write once, run anywhere code for various mobile and desktop operating systems (like Android, iOS, Windows, macOS, and others).
- ii) It was created by the co-founders of LWUIT project (Chen Fishbein and Shai Almog) and first announced January 13, 2012.



- iii) Codename One written in Java and Kotlin programming language.

iv) The syntax of Kotlin programming language is shown below.



The screenshot shows a web browser window with the Kotlin Playground interface. The browser's address bar displays the URL `play.kotlinlang.org` followed by a long alphanumeric hash. The page header includes the Kotlin logo and navigation links for Solutions, Docs, Community, Teach, and Play. Below the header, there are tabs for 'Play', 'Hands-on', 'Examples', and 'Koans'. The main area contains a code editor with the following Kotlin code:

```
/**
 * You can edit, run, and share this code.
 * play.kotlinlang.org
 */

fun main() {
    println("Hello, world!!!")
}
```




To the right of the code editor are four circular icons: a blue play button, a settings gear, a share icon, and a question mark. Below the code editor, the output area displays the text "Hello, world!!!".

7) Ruby Motion: -

- i) Ruby Motion is a partially open-source framework that runs on iOS, OS X and Android.
- ii) It is created by Laurent Sansonetti for HipByte.



- iii) Ruby programming language is mainly used for web development. However, with RubyMotion, mobile apps can be made using Ruby.
- iv) Syntax of ruby is shown below.

 Execute	 Share	main.rb	STDIN	 Result
<pre>1 # Hello World Program in Ruby 2 puts "Hello World!";</pre>				<pre>\$ruby main.rb Hello World!</pre>

8) QT: -

- i) **Qt** (pronounced "cute") is a widget toolkit for creating graphical user interfaces as well as cross-platform applications that run on various software and hardware platforms such as Linux, Windows, macOS, Android or embedded systems with little or no change in the underlying codebase while still being a native application with native capabilities and speed.
- ii) Qt is currently being developed by The Qt Company, a publicly listed company, and the Qt Project under open-source governance, involving individual developers and organizations working to advance Qt. Qt is available under both commercial licenses and open-source GPL 2.0, GPL 3.0, and LGPL 3.0 licenses.
- iii) QT framework is written in c++ programming language.



Chapter 2

Literature Survey

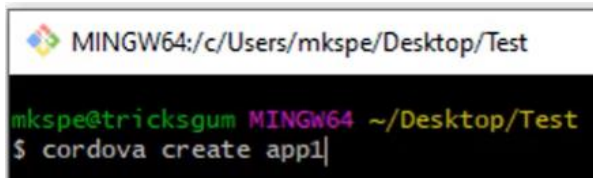
- While our studies sometimes delve into technological know-how from an implementation standpoint, we do not directly discuss the techniques to build an app. The analysis of mentioned frameworks and tools is inspired by the research conducted by N. Huy and D. VanThanh [3] on the criteria for evaluation of apps. The work done by M. Palmieri, I. Singh and A. Cicchetti [4] goes through the architecture and the working of each technology, and then proceeds to draw conclusions based on certain parameters. This paper seeks to perform side-by-side comparison of some of these tools. Our approach on cross-platform mobile development evaluation is different as it takes an example from each of the cross-platform approaches – web, hybrid, interpreted, and widget-based apps and compares them to a native development environment. Many papers offer an in-depth analysis of the cross-platform tools like Wenhao Wu’s [2] thesis on The Flutter framework however there are very few that compare cross-platform tools like web apps and hybrid apps with relatively newer ones like React Native and Flutter

Software Requirements

- 1) Android Studio
- 2) Git
- 3) Command Prompt

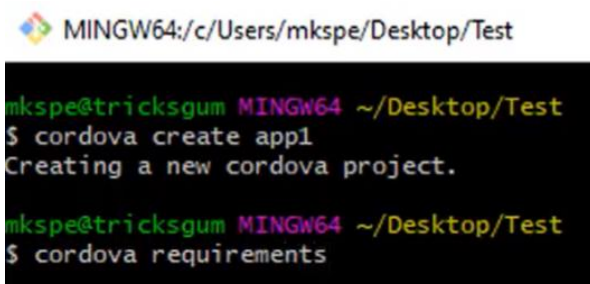
Steps to install Cordova framework by using Git software: -

Step 1) Type Cordova folder name(appl)



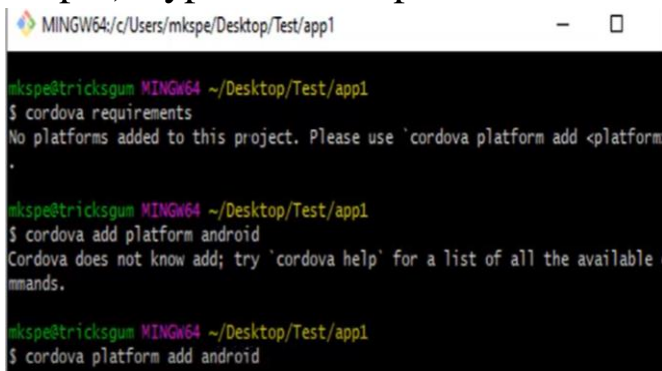
```
MINGW64:/c/Users/mkspe/Desktop/Test
mkspe@tricksgum MINGW64 ~/Desktop/Test
$ cordova create app1
```

Step 2) Type Cordova requirements



```
MINGW64:/c/Users/mkspe/Desktop/Test
mkspe@tricksgum MINGW64 ~/Desktop/Test
$ cordova create app1
Creating a new cordova project.
mkspe@tricksgum MINGW64 ~/Desktop/Test
$ cordova requirements
```

Step 3) Type Cordova platform add android



```
MINGW64:/c/Users/mkspe/Desktop/Test/app1
mkspe@tricksgum MINGW64 ~/Desktop/Test/app1
$ cordova requirements
No platforms added to this project. Please use 'cordova platform add <platform>'
.
mkspe@tricksgum MINGW64 ~/Desktop/Test/app1
$ cordova add platform android
Cordova does not know add; try 'cordova help' for a list of all the available
mmands.
mkspe@tricksgum MINGW64 ~/Desktop/Test/app1
$ cordova platform add android
```

Step 4) Type Cordova build

```
MINGW64:/c/Users/mkspe/Desktop/Test/app1
mkspe@tricksgum MINGW64 ~/Desktop/Test/app1
$ cordova requirements
No platforms added to this project. Please use 'cordova platform add <platform>'
.

mkspe@tricksgum MINGW64 ~/Desktop/Test/app1
$ cordova add platform android
Cordova does not know add; try 'cordova help' for a list of all the available co
mmands.

mkspe@tricksgum MINGW64 ~/Desktop/Test/app1
$ cordova platform add android
Using cordova-fetch for cordova-android@9.0.0
Adding android project...
Creating Cordova project for the Android platform:
  Path: platforms\android
  Package: io.cordova.hellocordova
  Name: HelloCordova
  Activity: MainActivity
  Android target: android-29
Subproject Path: CordovaLib
Subproject Path: app
Android project created with cordova-android@9.0.0
Discovered plugin "cordova-plugin-whitelist". Adding it to the project
Installing "cordova-plugin-whitelist" for android
Adding cordova-plugin-whitelist to package.json

mkspe@tricksgum MINGW64 ~/Desktop/Test/app1
$ cordova build
```

Chapter 3

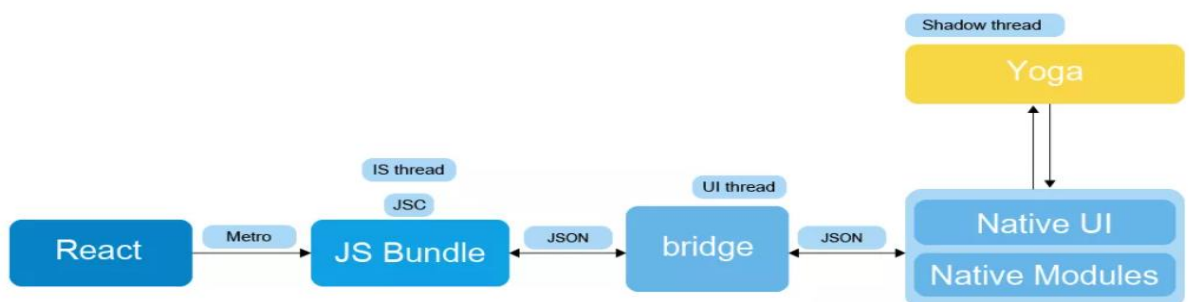
System Architecture

1) React Native: -

React Native architecture is based on 3 major pillars: -

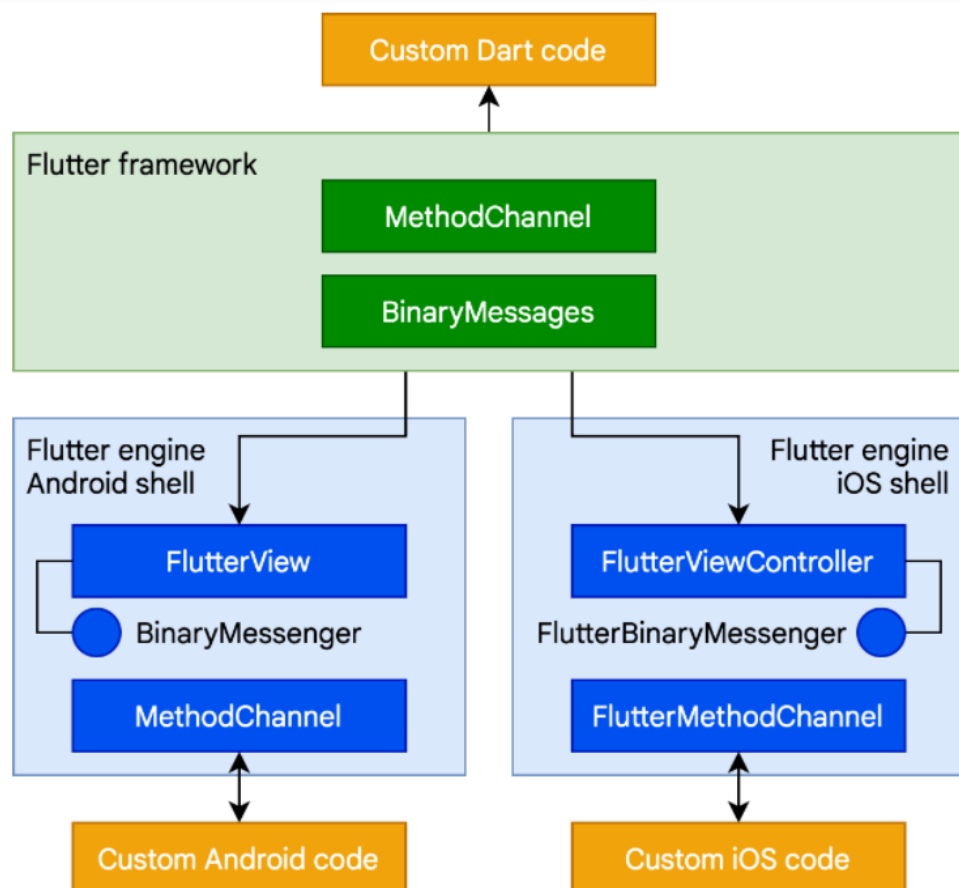
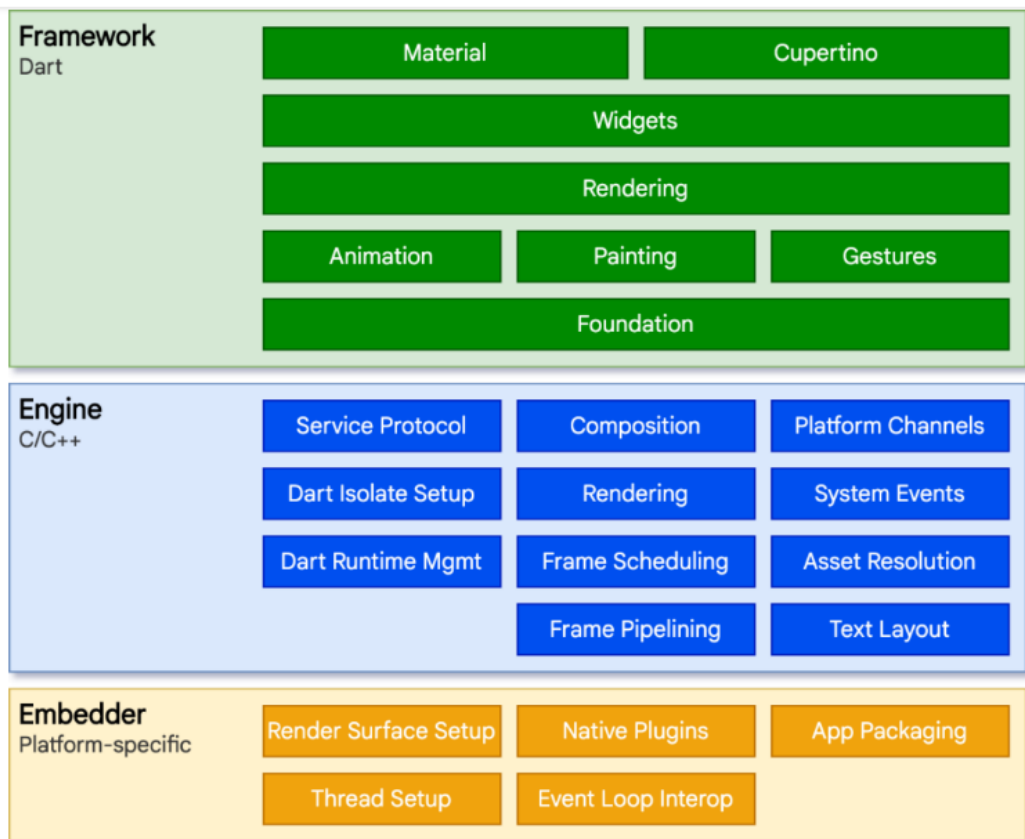
- The JavaScript Thread. This is the place where the entire JavaScript code is placed and compiled. When the app is bundled for production, the JavaScript Core runs the bundle when the user starts the app.
- The Native Thread. This is the place where the native code is executed. This component handles the user's interface and ensures seamless communication with the JS thread whenever the app needs to update the UI, run native functions, etc. All the native modules lie in the startup, which means they will always be bundled if the user wants to access them.
- The Shadow Thread. It is the place where the layout of your application is calculated. This cross-platform framework handles this task with the help of Facebook's own layout engine called Yoga. It transforms flexbox layouts, calculates them and sends them to the app's interface.

React Native Full Architecture



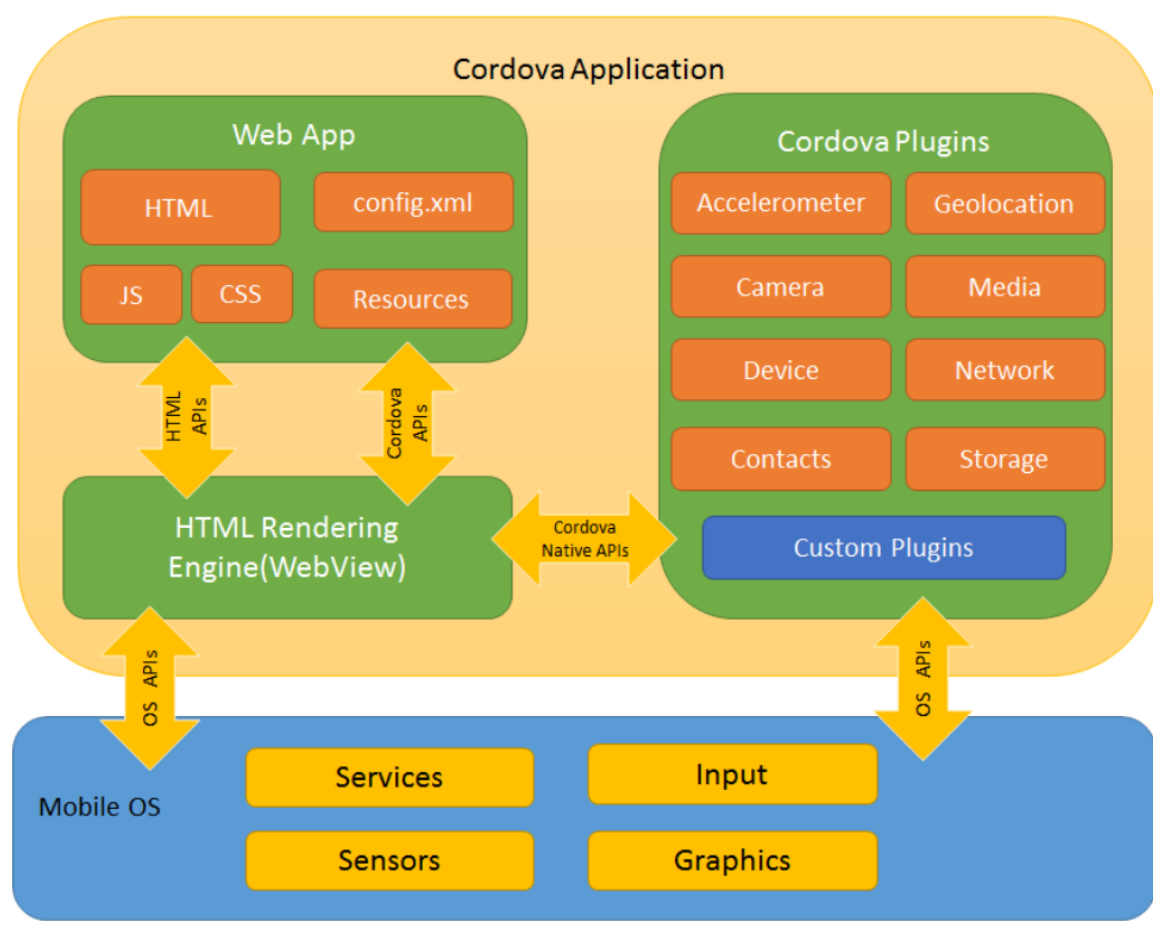
2) Flutter: -

- Flutter is designed as an extensible, layered system. It exists as a series of independent libraries that each depend on the underlying layer. No layer has privileged access to the layer below, and every part of the framework level is designed to be optional and replaceable.
- This overview is divided into a number of sections:
 1. **The layer model**: - The pieces from which Flutter is constructed.
 2. **Reactive user interfaces**: - A core concept for Flutter user interface development.
 3. **An introduction to widgets**: - The fundamental building blocks of Flutter user interfaces.
 4. **The rendering process**: How Flutter turns UI code into pixels.
 5. **An overview of the platform embedders**: - The code that lets mobile and desktop OSes execute Flutter apps.
 6. **Integrating Flutter with other code**: - Information about different techniques available to Flutter apps.
 7. **Support for the web**: - Concluding remarks about the characteristics of Flutter in a browser environment.
- While the general architectural concepts apply to all platforms that Flutter supports, there are some unique characteristics of Flutter's web support that are worthy of comment.
- Dart has been compiling to JavaScript for as long as the language has existed, with a toolchain optimized for both development and production purposes. Many important apps compile from Dart to JavaScript and run-in production today, including the advertiser tooling for Google Ads. Because the Flutter framework is written in Dart, compiling it to JavaScript was relatively straightforward.



3) Apache Cordova: -

- A Web App is defined as a core part where the application code resides. It is simply a webpage that is created using HTML, CSS and JavaScript. By default, a local file i.e., index.html is used to refer to CSS, JavaScript, media files and other resources that are required to run the application. The app is mainly executed in a WebView within the native application wrapper that are distributed to the app stores.
- This container consists a key file i.e., config.xml that are responsible for providing the information about the app.
- Cordova creates an app, in which the web app is initialized in a WebView i.e., HTML Rendering Engine.



- Plugins are an integral part of the Cordova ecosystem. They provide an interface for Cordova and native components to communicate with each other and bindings to standard device APIs. This enables user to invoke native code from JavaScript.
- Apache Cordova project maintains a set of plugins called the Core Plugins. These core plugins provide user application to access device capabilities such as battery, camera, contacts, etc.
- In addition to the core plugins, there are several third-party plugins which provide additional bindings to features not necessarily available on all platforms.

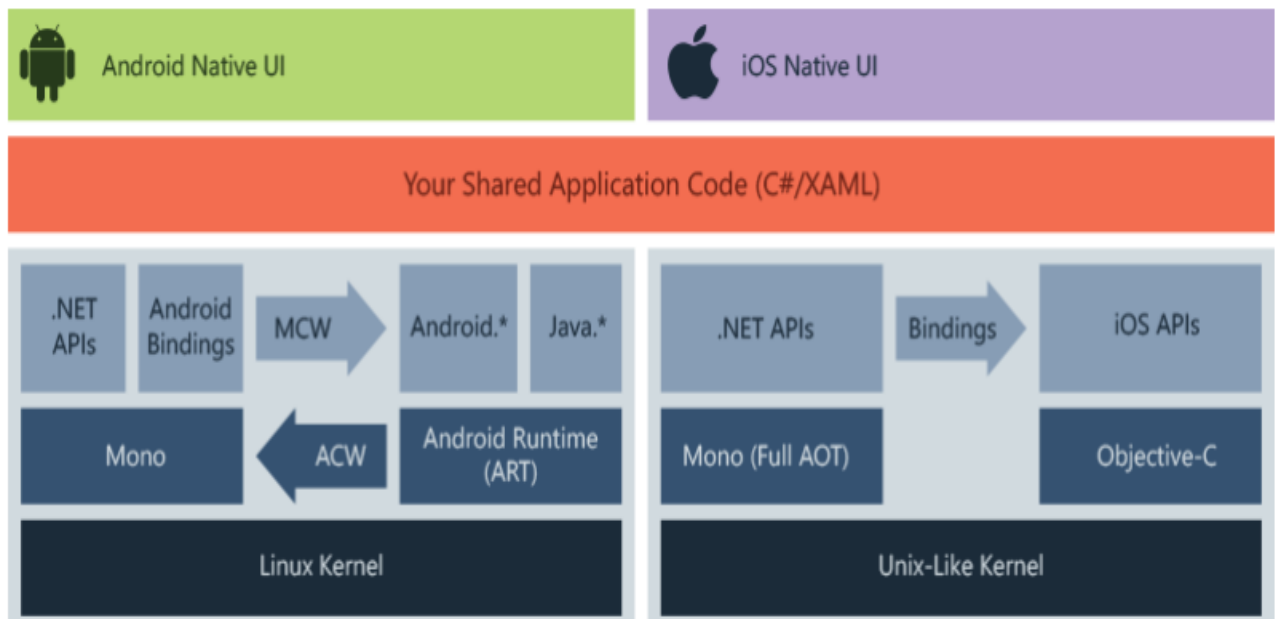
4) Ionic: -

- Ionic apps are built with Cordova. Cordova is a means of packaging html/css/js into apps that can run on mobile and desktop devices and provides a plugin architecture for accessing native functionality beyond the reach of JS run from a web browser. As such, Ionic apps have the Cordova file structure.
- Plugins are an integral part of the Cordova ecosystem. They provide an interface for Cordova and native components to communicate with each other and bindings to standard device APIs. This enables user to invoke native code from JavaScript.
- Apache Cordova project maintains a set of plugins called the Core Plugins. These core plugins provide user application to access device capabilities such as battery, camera, contacts, etc.
- In addition to the core plugins, there are several third-party plugins which provide additional bindings to features not necessarily available on all platforms.



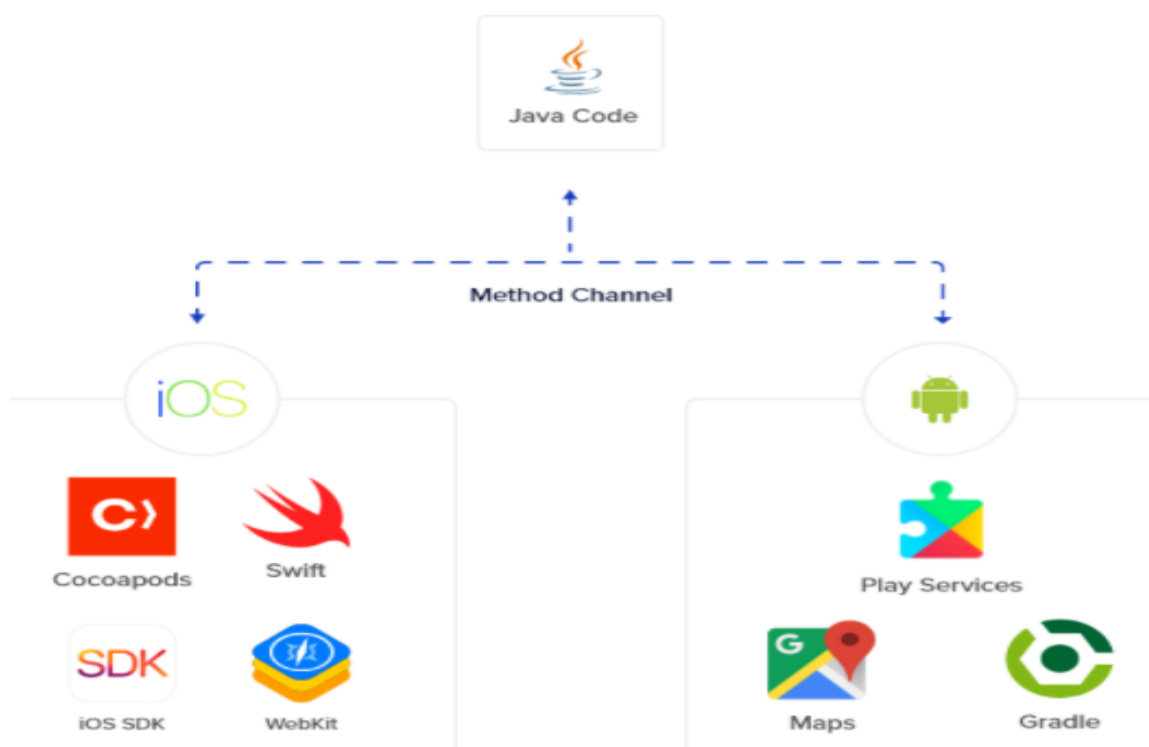
5) Xamarin: -

- Xamarin allows you to build native Android, iOS, and Windows applications using .NET. Common patterns, such as MVVM, combined with good application layering, will maximize code sharing and result in an application that is easier to understand, test, and maintain.
- Xamarin. Android applications run within the Mono execution environment. This execution environment runs side-by-side with the Android Runtime (ART) virtual machine. Both runtime environments run on top of the Linux kernel and expose various APIs to the user code that allows developers to access the underlying system.



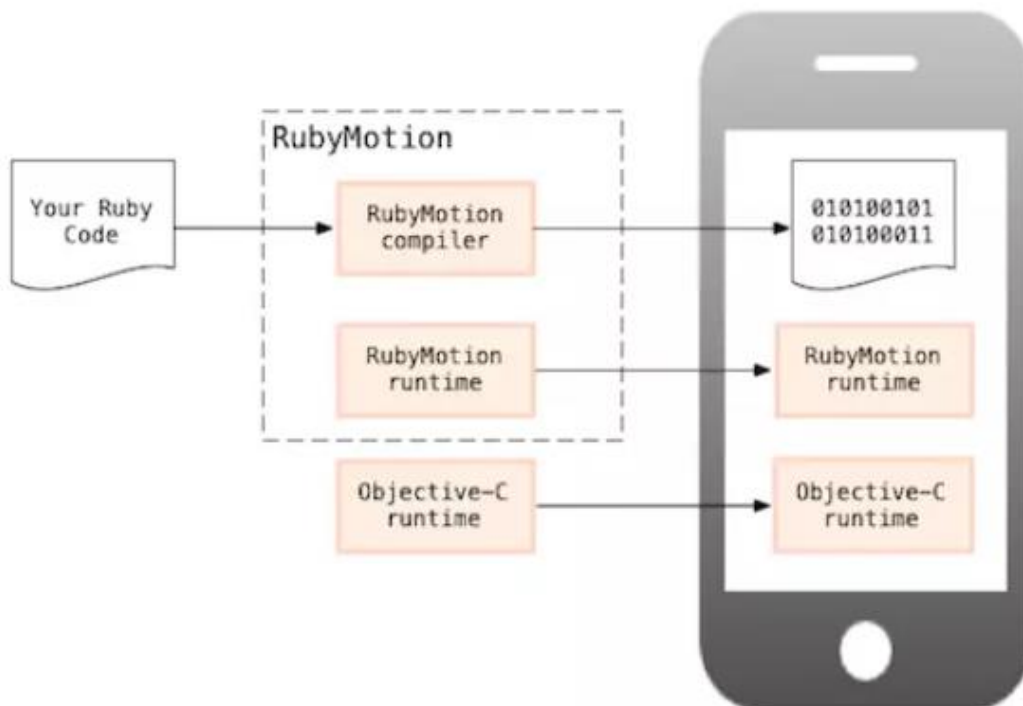
6) Codename One: -

- v) Codename One is an open-source cross-platform framework aiming to provide write once, run anywhere code for various mobile and desktop operating systems (like Android, iOS, Windows, macOS, and others).
- vi) It was created by the co-founders of LWUIT project (Chen Fishbein and Shai Almog) and first announced January 13, 2012.
- vii) Codename One written in Java and Kotlin programming language.
- viii) Codename One is a revolutionary mobile development solution started by ex-Sun Microsystems developers based on the work that started within Sun.
- ix) Its core appeal is its unrestricted access to the native platform allowing developers to write native code directly from Java or Kotlin and access everything that the native mobile platform can provide.



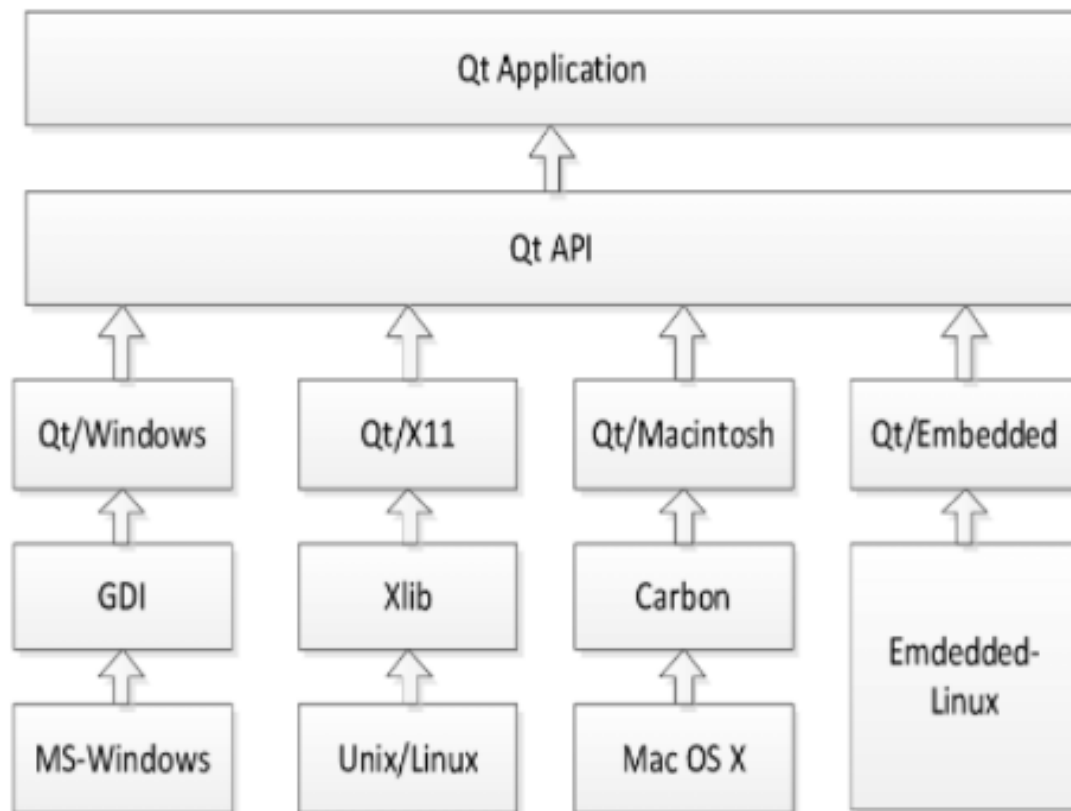
7) Ruby motion: -

- Ruby programming language is mainly used for web development. However, with RubyMotion, mobile apps can be made using Ruby.
- RubyMotion is based on MacRuby, an implementation of Ruby created and maintained previously at Apple. RubyMotion adapted and extended MacRuby to work on platforms beyond MacOS.
- Apps written in RubyMotion call into the native platform APIs and function in the same manner as platform native language. RubyMotion apps are created from the terminal command-line using any text editor.



8) QT: -

- Qt is used for developing cross-platform apps and graphical user interfaces (GUIs) that run on most mobile, desktop or embedded platforms. Qt supports compilers such as Visual Studio, GCC compiler and PHP via an extension.
- Qt tools include the Qt Creator IDE for C++ and Qt Quick which includes a declarative scripting language called QML. Other Qt features include XML parsing, JSON parsing, SQL database access and thread management.



Native Apps

- Native apps live on the device and are accessed through icons on the device home screen. Native apps are installed through an application store (such as Google Play or Apple's App Store). They are developed specifically for one platform, and can take full advantage of all the device features they can use the camera, the GPS, the accelerometer, the compass, the list of contacts, and so on. They can also incorporate gestures (either standard operating-system gestures or new, app-defined gestures). And native apps can use the device's notification system and can work offline.

Hybrid apps

- **Hybrid apps** are part native apps, part web apps. (Because of that, many people incorrectly call them “web apps”). Like native apps, they live in an app store and can take advantage of the many device features available. Like web apps, they rely on HTML being rendered in a browser, with the caveat that the browser is embedded within the app.
- Often, companies build hybrid apps as wrappers for an existing web page; in that way, they hope to get a presence in the app store, without spending significant effort for developing a different app. Hybrid apps are also popular because they allow cross platform development and thus significantly reduce development costs: that is, the same HTML code components can be reused on different mobile operating systems. Tools such as PhoneGap and Sencha Touch allow people to design and code across platforms, using the power of HTML.

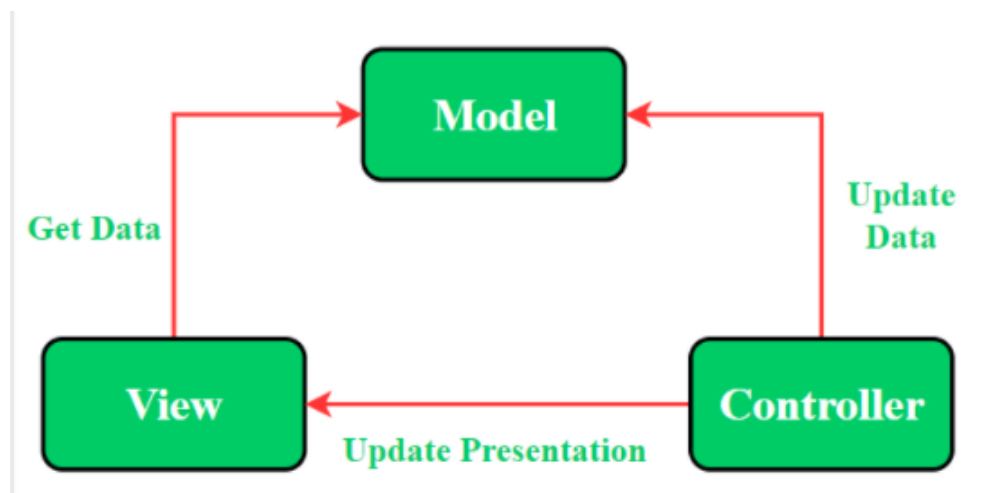
Web Apps

- Web apps are not real applications; they are really websites that, in many ways, *look and feel* like native applications, but are not *implemented* as such. They are run by a browser and typically written in HTML5. Users first access them as they would access any web page: they navigate to a special URL and then have the option of “installing” them on their home screen by creating a bookmark to that page.
- Web apps became really popular when HTML5 came around and people realized that they can obtain native-like functionality in the browser. Today, as more and more sites use HTML5, the distinction between web apps and regular web pages has become blurry.
- Its web app is, in many ways, hard to distinguish from a native app. For instance, there are no visible browser buttons or bars, although it runs in Safari (when accessed from an iPhone). Users can swipe horizontally to move on to new sections of the app. And, due to browser caching, it’s even possible to read the newspaper offline.
- These are all features that are available in HTML5. Also available are the GPS, the tap-to-call feature, and, there is talk about a camera API, although I haven’t seen any web app (or web page) that takes advantage of it so far. There are, however, native features that remain inaccessible (at least from now) in the browser: the notifications, running in the background, accelerometer information (other than detecting landscape or portrait orientations), complex gestures.

Chapter 4: (Model)

Model View Controller (MVC)

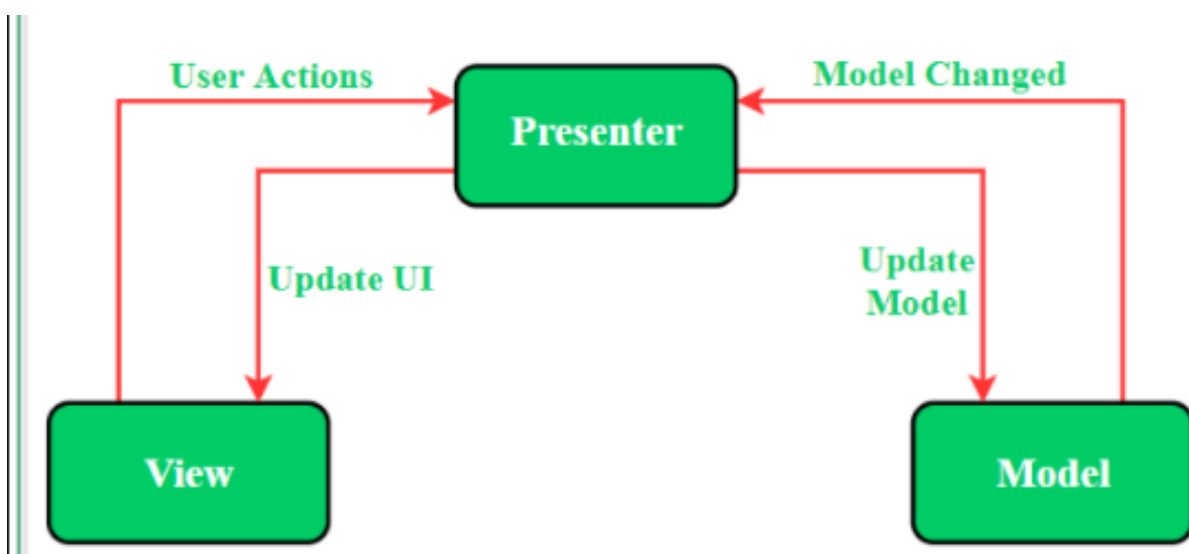
- The MVC pattern suggests splitting the code into 3 components. While creating the class/file of the application, the developer must categorize it into one of the following three layers:
 - 1) Model: This component stores the application data. It has no knowledge about the interface. The model is responsible for handling the domain logic (real-world business rules) and communication with the database and network layers.
 - 2) View: It is the UI (User Interface) layer that holds components that are visible on the screen. Moreover, it provides the visualization of the data stored in the Model and offers interaction to the user.
 - 3) Controller: This component establishes the relationship between the View and the Model. It contains the core application logic and gets informed of the user's response and updates the Model as per the need.



Model-View-Presenter (MVP)

- MVP pattern overcomes the challenges of MVC and provides an easy way to structure the project codes. The reason why MVP is widely accepted is that it provides modularity, testability, and a cleaner and more maintainable codebase. It is composed of the following three components:

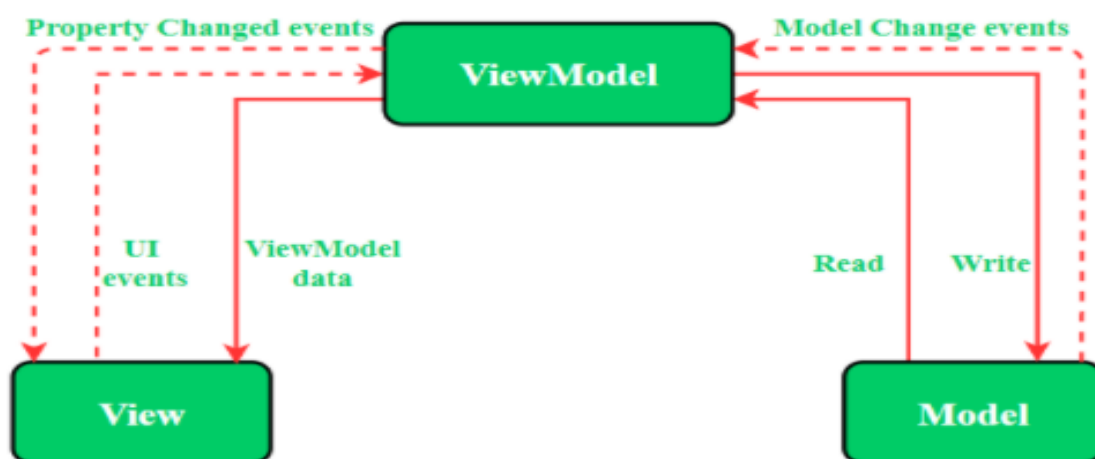
- 1) Model: Layer for storing data. It is responsible for handling the domain logic (real-world business rules) and communication with the database and network layers.
- 2) View: UI (User Interface) layer. It provides the visualization of the data and keep a track of the user's action in order to notify the Presenter.
- 3) Presenter: Fetch the data from the model and applies the UI logic to decide what to display. It manages the state of the View and takes actions according to the user's input notification from the View.



Model -View -View -Model (MVVM)

- MVVM pattern has some similarities with the MVP (Model — View — Presenter) design pattern as the Presenter role is played by the View-Model. However, the drawbacks of the MVP pattern have been solved by MVVM. It suggests separating the data presentation logic (Views or UI) from the core business logic part of the application. The separate code layers of MVVM are:

- 4) Model: This layer is responsible for the abstraction of the data sources. Model and View-Model work together to get and save the data.
- 5) View: The purpose of this layer is to inform the View-Model about the user's action. This layer observes the View-Model and does not contain any kind of application logic.
- 6) View-Model: It exposes those data streams which are relevant to the View. Moreover, it serves as a link between the Model and the View.



Chapter 5

Applications

- The cross-platform must give the possibility to maintain and improve the application
- For example, if new options need to be implemented in an application, the modifications have to always be in the cross-platform level and thereafter deployed in deferent platforms.

Chapter 6

Advantages

- Cross-platform tools are also referred to as WORA tools (Write Once, Run Anywhere) which is the primary advantage of such a method.
- The app is implemented using a single code base but can be deployed to multiple platforms, i.e., it is not restricted to a single platform.
- Cross-platform tools usually use well-known programming languages and syntaxes, which becomes an easy and quick means for development.
- One of the best things about this particular platform is that the entire code can easily be utilized again and again. Rather than developers developing new codes for each and every platform, a single code can simply be reused. Hence, it saves time along with resources as it completely eliminates repetition in the task of keep creating codes.

Chapter 7

Disadvantages

- Native applications are slightly faster than the cross – platform applications.
- Native apps provide a richer user experience. Rendering of high-end graphics is only effectively possible with native app development.
- Cross-compliance during the development phase reduces the speed.

Chapter 8

Conclusion

- The purpose of our paper is to perform a comprehensive analysis of the cross-platform applications. In this paper, we introduce cross-platform development tools and contrast it with native app development. Cross-platform approaches require developers to compromise on the user experience - a native touch and feel, can only be accomplished by the use of truly native components inside the app. A deeper integration with the device and improved performance are invaluable parameters to be considered when developing apps. However, we found that the use of cross-platforms tools usually outweighs the disadvantages that come with it, except for when, for example, creating high-end gaming apps such as Asphalt. Reviewing different types of cross-platform approaches and taking an example from each category gave us an idea about the characteristics of each type of approach and how useful they are in different scenarios. Cross-platform frameworks despite their many advantages are still in their preliminary stages in the market. It has barely been a decade with the advent of interpreted apps and yet newer innovative approaches such as Flutter continue to be developed which propel the future of cross-platform mobile app development. While the ultimate choice of cross-platform tools rests with the developer, we tried to shed some light on the methodology to adopt when making such apps. In the near future, we look forward to carrying out a formal assessment by testing a simple version of a cross-platform app developed from each framework and henceforth comparing our inferences with the quantitative values obtained.

References

- [1] “Number of smartphone users worldwide 2014-2020.” Internet: <https://www.statista.com/statistics/330695/number-of-smartphoneusers-worldwide/> [Oct. 2, 2018].
- [2] “Mobile Operating System Market Share Worldwide.” Internet: <http://gs.statcounter.com/os-market-share/mobile/worldwide> [Oct. 5, 2018].
- [3] N. P. Huy and D. vanThanh, “Evaluation of mobile app paradigms,” in Proc. MoMM ’12 10th International Conference on Advances in Mobile Computing & Multimedia, 2012, pp. 25-30.
- [4] M. Palmieri, I. Singh and A. Cicchetti, "Comparison of cross-platform mobile development tools," 2012 16th International Conference on Intelligence in Next Generation Networks, Berlin, 2012, pp. 179-186.
- [5] W. Wu “React Native vs Flutter, Cross-platforms mobile application frameworks.” B.E. thesis, Metropolia University of Applied Sciences, Finland, 2018.
- [6] S. Helal, J. Hammer, J. Zhang and A. Khushraj, "A three-tier architecture for ubiquitous data access," in Proc. ACS/IEEE International Conference on Computer Systems and Applications, Beirut, Lebanon, 2001, pp. 177-180.
- [7] S. Xanthopoulos and S. Xinogalos, “A comparative analysis of cross-platform development approaches for mobile applications,” in Proc. BCI ’13 6th Balkan Conference in Informatics, 2013. pp. 213-220.
- [8] N. Jain, P. Mangal and D. Mehta. “AngularJS: A Modern MVC Framework in JavaScript.” Journal of Global Research in Computer Science, vol. 5, pp. 17-23, Dec. 2014.
- [9] “Angular Docs.” Internet: <https://angular.io/guide/architecture> [Oct. 10, 2018].
- [10] “Mobile App Services.” Internet: <http://www.directiontech.com/dt7/mobile-apps> [Oct. 10, 2018].
- [11] E. Palme, “How to wrap an Angular app with Apache Cordova.” Internet: <https://medium.com/@EliaPalme/how-to->

wrap-an-angularapp-with-apache-cordova-909024a25d79, Feb. 1, 2018 [Oct. 12, 2018].

- [12] S. Bosnic, I. Papp and S. Novak, "The development of hybrid mobile applications with Apache Cordova," 2016 24th Telecommunications Forum (TELFOR), Belgrade, 2016, pp. 1-4