Survey On Application Software

- Industrial Insights -

Project Report

Aarthi Babu
Raja Gunasekaran

University Of Northern British Columbia
Computer Science

Contents

Chapter 1

Introduction

Application Software is program or group of program customised to perform group of activities for end user. Generally software are classified as system software and application software.System software consists of low-level programs that is designed to run computer's hardware and application programs like compilers, loaders, linkers and so on. Application software resides above system software like database programs, word processors, spreadsheets. In this survey, we focus on the architecture and process involved in development of different types of application software.

Software application architecture is the process of designing a structured solution that focuses on how the components in the applications interact with each other, while optimizing common quality attributes such as performance, security, and manageability.The architecture are structured with consideration of user and the business goals.The selection of data structures and algorithms are design concerns which would overlap with architecture concerns.These scenarios where the both of them overlap are discussed in detailed in later chapters.The architecture must be flexible to handle the changes in software or hardware technologies and requirements that are not known in the early stages of design process as the design of the application will evolve during the development stages.

1.1 Command-Line Application

Technology has come long way since the first computers were created, back around the start of World War II. The first generation of software application are command line programs which are mostly single command at a time and uses it to accomplish all the application requirements in that particular loop. These programs are shared as binaries(executables) which can be compiled from the source code, specific to the architecture.

1.2 Desktop Application

In spite of early application which was designed to run from mainframe computers and accessed through terminals devices, the proposal of powerful desktop applications which can be run in the local personal computer dethroned the previous generation application software. The mainframe was replaced with server by the client server model which induced distributed application structure that partitions tasks or workload between the service provider (Server) and the service requester (Client).

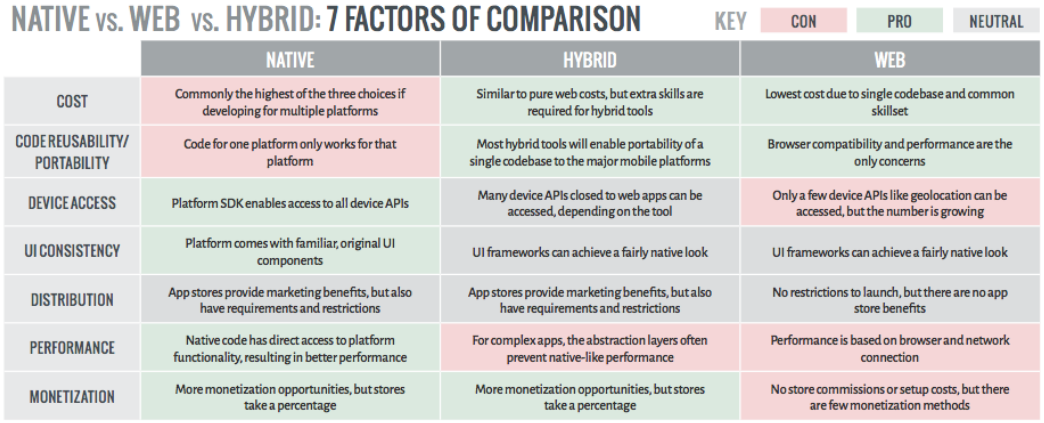
1.3 Web Application

Steady enhancements in hardware specifications and broadband speeds led to major improvements in the quality and quantity of WWW content. Websites inflated their features beyond static web pages by becoming more interactive with the increase of multimedia content. As browsers and development platforms evolved, and more and more people began to use the internet and email, more businesses established their presence in the online world. These businesses leveraged the emerging interactive capabilities of the web to introduce applications that were served directly to a web browser, and these web applications became very popular. Any Web application is a client server application which consist of a client inform of browser and a web server. The business logic of the application is distributed among the server and client where information is exchanged through a channel and the data is stored in the server end.

1.4 Mobile Application

Nowadays, an ever-growing percentage of portable device like smart phones have doubled the number of users accessing the internet on their smart phones.The need of mobile applications became pretty obvious. Mobile applications should be designed in such a way that the power consumption and processing power is reduced. Android and iOS being the major players in the industry, they natively support java and ObjectiveC as the native language.

The mobile applications can be categorised into mobile web application, native application and hybrid application.A native application is developed for a certain mobile device (smart phone, tablet, so on). They're installed directly onto the device. Users typically acquire these applications through an online store or marketplace such as The App Store or Android Apps on Google Play.It was started working as standalone, and in the recent years, we see a lot of integration with the web which is named as hybrid application.



NATIVE vs. WEB vs. HYBRID: 7 FACTORS OF COMPARISON

KEY: CON | PRO | NEUTRAL

	NATIVE	HYBRID	WEB
COST	Commonly the highest of the three choices if developing for multiple platforms	Similar to pure web costs, but extra skills are required for hybrid tools	Lowest cost due to single codebase and common skillset
CODE REUSABILITY/ PORTABILITY	Code for one platform only works for that platform	Most hybrid tools will enable portability of a single codebase to the major mobile platforms	Browser compatibility and performance are the only concerns
DEVICE ACCESS	Platform SDK enables access to all device APIs	Many device APIs closed to web apps can be accessed, depending on the tool	Only a few device APIs like geolocation can be accessed, but the number is growing
UI CONSISTENCY	Platform comes with familiar, original UI components	UI frameworks can achieve a fairly native look	UI frameworks can achieve a fairly native look
DISTRIBUTION	App stores provide marketing benefits, but also have requirements and restrictions	App stores provide marketing benefits, but also have requirements and restrictions	No restrictions to launch, but there are no app store benefits
PERFORMANCE	Native code has direct access to platform functionality, resulting in better performance	For complex apps, the abstraction layers often prevent native-like performance	Performance is based on browser and network connection
MONETIZATION	More monetization opportunities, but stores take a percentage	More monetization opportunities, but stores take a percentage	No store commissions or setup costs, but there are few monetization methods

Figure 1.1: Native vs Web vs Hybrid

Internet-enabled applications that have specific functionality for mobile devices are called as mobile web application. They are accessed through the mobile device's web browser (i.e. on the iPhone, this is Safari by default) and they don't need to be downloaded and installed on the device.Although mobile websites and mobile apps aren't the same thing, they generally offer the same features that can help grow business by making it easier for customers to find and reach it.

Chapter 2

Software Application Architecture

Overall structure of the software application can be represented in the logical grouping of components into layers that interact with each other according to the functionality and features of the application. This chapter will focus on how the application is divided into components and the services provided by each layer.These layers help to uniquely identify the different kind of task performed by each components. Each layer would consist of multiple sub layers which performs specific type of task which would greatly help while debugging.

By analysing the types of components that exist in most solutions, you can construct a meaningful map of an application or service, and then use this map as a blueprint for your design. Dividing an application into separate layers that have distinct roles and functionalities helps you to maximize maintainability of the code, optimize the way that the application works when deployed in different ways.Figure 2.1 represents the highest and most abstract level of the software architecture.It gives clear perspective of theirs interaction with users, relationship with other application that requests the services implemented within the application business layer,web services hosted by the application , data sources and the remote service used by the applications offered by other software.

2.1 Basic Architecture Layer

The common three layer design consist of Presentation Layer ,Business Layer and Data Layer.

- **Presentation Layer :** This layer consist of two sub components which contains purely user based functionality that manages the user interaction with the application. This layer acts as a bridge between the user and business logic layer. User Interface and user interface logic are the two major components found in this layer.This layer is responsible for the user input and display

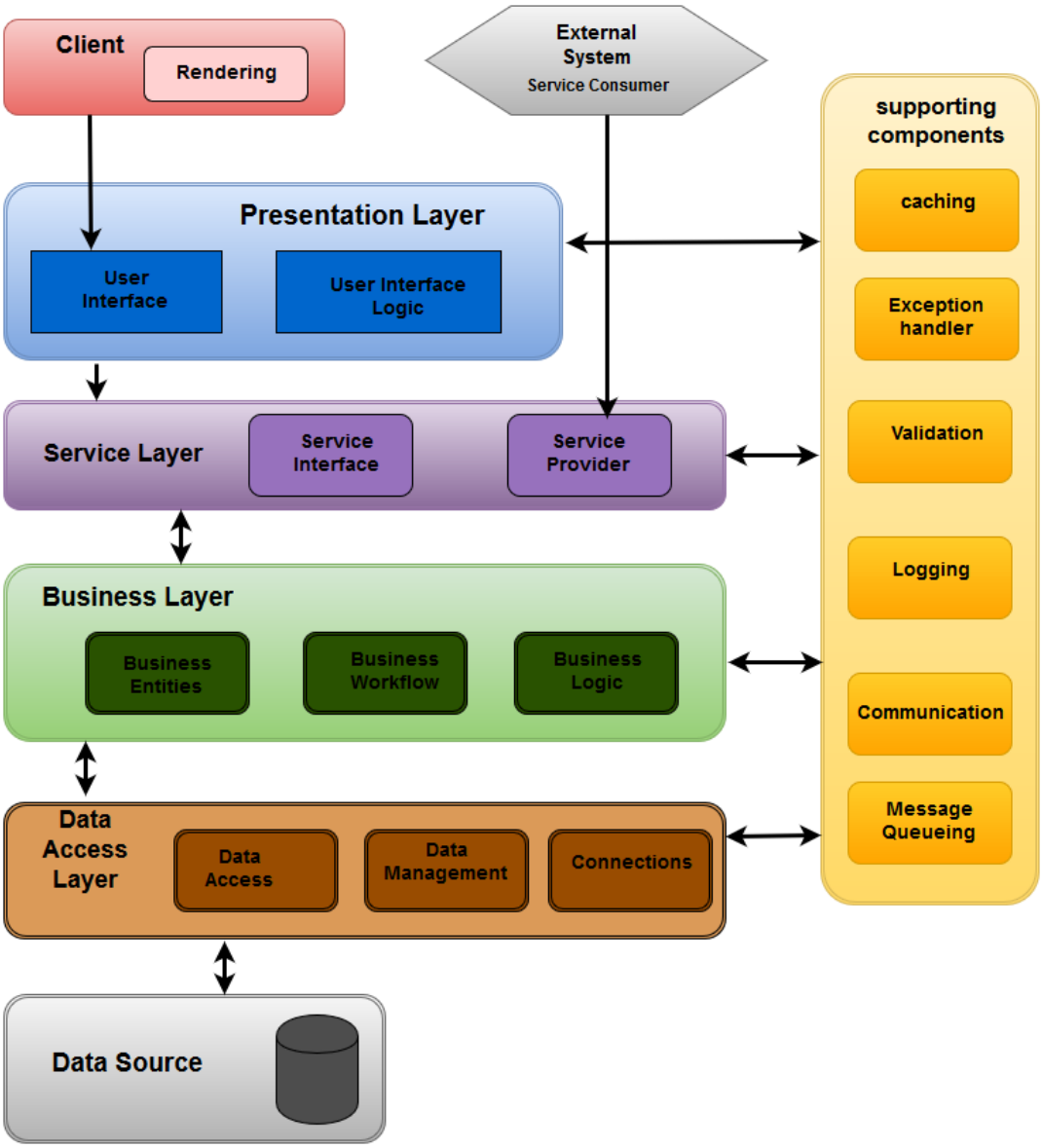


Figure 2.1: Software Application Architecture

in addition to the components that organise user Interaction.To make the code modular the implementation of visual elements which would display data to user and accepts input is separated from the presentation logic which gives rise to the idea of model, view and controller.To handle the design based issues , security issues and improve the application performance and user interface responsiveness, the extra components are introduced which discussed in forth coming chapters.

- **Business Layer :** This layer implements core functionality of the application and is concerned with the processing, transformation, retrieval and management of data.The core implementation include the business rules,entities and work flow of the rules. Application façade is a optional component which acts a simplified wrapper on business logic components by combining multiple operations into a single operation in a logical way. This reduces dependencies and improves modularity. This prevents the external service request from knowing the details of the business components and the inter relationship between the operations.

- **Data Layer :** Data access logic and data store are the major components of this layer. Much more components like object document mapper , object relational Mapper etc which deals with translation of data can be introduced into this layer based on the requirements.This layer provides access to data within the boundaries of the system and the data exposed by the other networked systems through web service.Data access logic components forms a interface that the components in the business layer consumes.

2.2 Requirement-Based Components

Some layers are pluggable in the architecture on requirement basis.When an application must provide service to external system , the service layer is used to expose the business functionality without exposing the operation signature.There are components which are added to the architecture to handle issues.The issues and the corresponding layer which handles the issue can be categorised based on the architecture layers.

2.2.1 Issues and Add-on - Presentation Layer

- **Caching :** Caching is considered to be a best mechanism to improve application performance and user interface responsiveness. Caching is used in presentation layer to avoid frequent data lookups which in turn avoids network round trips and to store the result of the repetitive process to avoid the unnecessary duplicated processing. Its important to make the cache thread-safe when it is used in multi-thread environment.

- **Exception Management :** Employing a centralised exception management is considered to be a consistent way to manage the unexpected exceptions. The exception which propagates across layers are blocking issue which would crash the application. Differentiating the error occurred into system-based or business logic based will be helpful to provide a friendly error message to the users. If it is business errors,a error message can be displayed and allow user

to re-try that particular operation. In case of system error, a error message can be displayed along with the troubleshooting assistance. It is important to ensure no sensitive data is exposed in the error pages, error message and log files.

- **Compositions :** User interface composition patterns and templates supports the creation of the views and the presentation layout at the run times.It is easy to develop and maintain if the presentation layers uses independent modules and views that are composed at run time. These templates helps to minimise the code and library dependencies that would otherwise force recompilation and redeployment of the modules when the dependencies are upgraded.

2.2.2 Issues and Add-on - Business Layer

The business layer includes the previous add-ons like caching

Data Layer Batching : To achieve high performance , the similar queries are batched avoiding multiple database request and execution. Each database request is considered to be a overhead. Batched commands reduces the round trip time to database and minimizes network traffic as well as the size of the connection pool.

- **Authentication :** Security and reliability of an application is important for business layer where authentication comes into picture.If your business layer will be used by multiple application , plugging an authentication mechanism into the business layer ensures the security of the sensitive data. In case of client server application, the application layer lies in client and the business layer lies in server application. Both application communicates over network since they are deployed in different machines. In this case, a centralised platform which handles authentication is mandatory for the reliability of the application

- **Authorization :** In most of the applications the authorization components manages issues like information disclosure and user privileges for specific activity.Authorization are used to allow specific group of user to perform some sensitive business information and view some sensitive data. It promotes hierarchy of user privileges and the related operations ensuring security of the data sources.It is important to implement the authorization code in a modular way which doesn't impact on performance of the application and doesn't have any dependency with business layer .

- **Logging and Auditing :** Maintaining a centralised logging and auditing system would be useful in tracking the issues in production and the illegal user actions in important situations. System Monitoring tools are used to identify

state and performance of the application. Logging helps to track this performance and the load handled by the application. It is important to ensure that logging failure doesn't affect the functionalities of business layer.

2.2.3 Issues and Add-on - Data Layer

- **Batching :** To achieve high performance , the similar queries are batched avoiding multiple database request and execution. Each database request is considered to be a overhead. Batched commands reduces the round trip time to database and minimizes network traffic as well as the size of the connection pool.

- **Connection :** Establishing connection to data sources is considered to be basic part of data access layer.Creating and maintaining data sources connection in inefficient way directly impacts the performance of the applications. Connection pooling within a security model would serve the purpose in efficient way.Holding connection for short period and avoiding the usage of user DSN or system DSN would help to achieve high performance and security.

- **Object Relational Mapping (ORM) :** When the application is designed using object oriented model, It is difficult to translate the relational model to business entities. ORM can be used to create schema which supports the object model and provides a mapping between the database and domain entities.When working with Web applications or services, group entities and support options that will partially load domain entities with only the required data—a process usually referred to as lazy loading. This allows applications to handle the higher user load required to support stateless operations, and limit the use of resources by avoiding holding initialized domain models for each user in memory.

2.2.4 Issues and Add-on - Service Layer

In case of external application acting as service consumer or the application is designed in client server model, the service layer acts as interface between the systems. Service Interface represent the operations supported and their associated parameters and data transfer objects. To avoid multiple call over network, the service interface are designed to serve multiple operations grouped based on business logics. Authentication is used to determine the identity of the service consumer.Authorization is used to determine which resources or actions can be accessed/possessed by an authenticated service consumer. Communication component can be designed based on the requirement like request-response ,duplex communication and asynchronous communication.Message queueing can be used to handle tremendous burst of asynchronous message .

Chapter 3

Presentation Layer

The design of presentation layer begins by identifying the potential customer base, and understanding the objectives of the customer and task the user wish to accomplish when using the application. As the main focus of the application design must be user centred, the sequencing of tasks or operations should be designed as per the user expectation. It may be a structured step-by-step user experience or unstructured experience where they can perform more than one tasks simultaneously. One important aspect that has great influence on the choice of technology is functionality required for the user interface. Prioritizing the requirements like rich functionality, user interaction, user responsiveness , user interface , personalization requirements and graphical support will help to choose User Interface Type.

Most of the user interface requirements needs more than one UI type.The frameworks, tool and implementation language used for UI requirements differ according to the application platform.

3.1 Mobile Application

Mobile application can be classified as thin client and rich client application.Rich client is designed to support the native or hybrid applications which supports offline or intermittent online scenarios. Web or thin clients supports only online scenarios.Resources constraint should e taken into account while designing user interface for mobile application.

Many developers or vendors want to write an application once using single codebase , the run that application in multiple platform with a very small change in codebase. This approach is famously known as Write Once, Run Anywhere (WORA).WORA removes the redundancy in development at the cost of user experience and application performance.This application is called as hybrid mobile application which has native application experience. Improving the performance and user experience are in future roadmap.

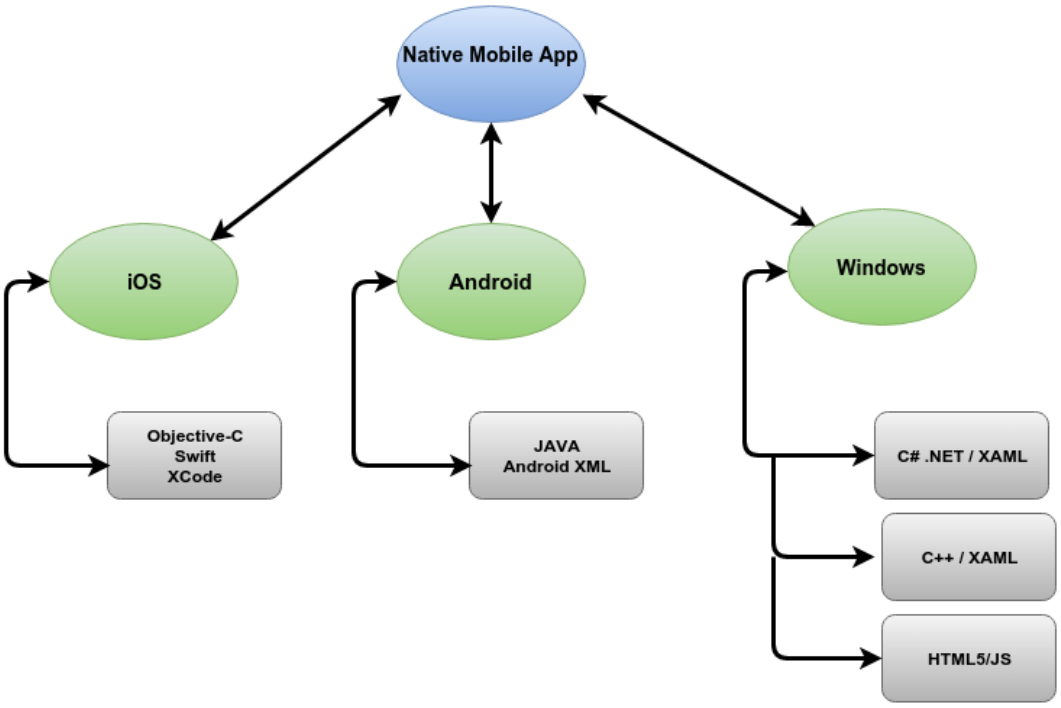


Figure 3.1: Languages - Native Moblie Application

3.1.1 Apache Cordova - Frameworks

Apache Cordova is an open-source mobile cross-platform development framework which allows you to use standard web technologies - HTML5, CSS3, and JavaScript . In mobile development, this framework extends more than one platform avoiding the implementation time taken for each platform.Applications execute within wrappers targeted to each platform, and rely on standards-compliant API bindings to access each device's capabilities such as sensors, data, network status, etc. It is flexible in developing application which has combination of native application components with a web view that can access device level APIs using Core Plugins.

3.1.2 Ionic - Frameworks

Ionic is a free open source framework facilitates the development of hybrid native mobile using web technologies like HTML,Javascript and CSS. Its core is built up with AngularJS javascript framework and supports SASS (Syntactically Awesome Style Sheets) CSS extension . This features qualifies Ionic as a unique framework.It simplifies development and testing of the application by providing client side model view controller architecture. User Interface response is pretty faster than Backbone and Knockout and it possesses a large number of 3rd party plug ins and extensions. Currently Ionic supports iOS and Android devices, Windows

phones and FirefoxOS support are considered to be in future roadmap. This framework can be only used for hybrid mobile application.Ionic is built on top of Apache Cordova which is open source mobile development framework.

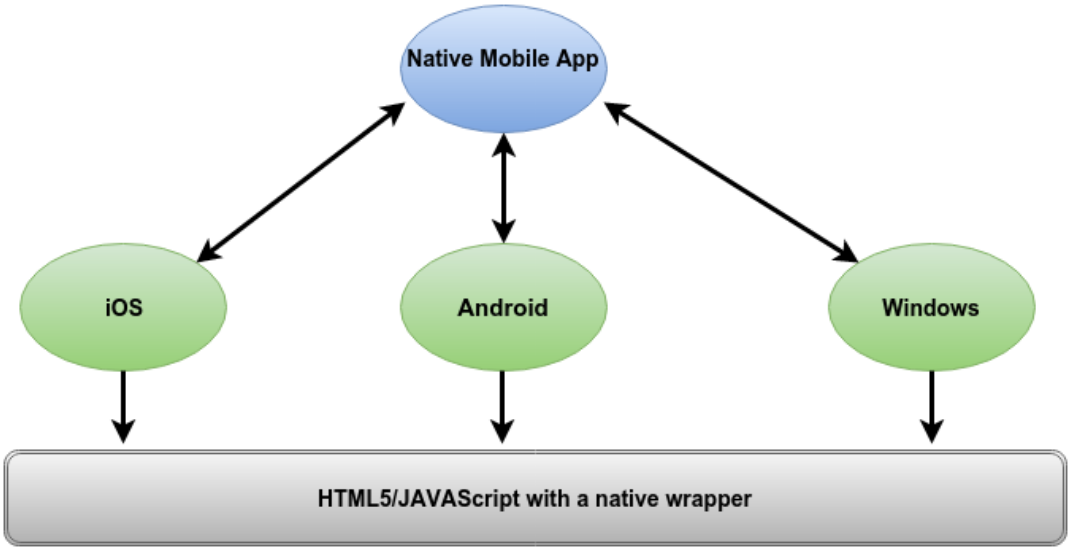


Figure 3.2: Cordova Languages - Hybrid Moblie Application

3.1.3 Mobile Angular UI - Frameworks

Mobile Angular UI is an HTML 5 framework which uses bootstrap 3 and AngularJS with better user response. In other words it combines the best of the web framework which enables you to create hybrid application and web application. This framework works well with all browsers including older versions and offers feature of customizing build workflow with GRUNT. Grunt is used run automation using a task runner with zero effort.

3.1.4 Xamarin - Frameworks

Xamarin applications are written in C# and compiled to binaries compatible with native device giving access to all the device API through C# wrappers.To develop application that look like native, Xamarin uses native UI mechanisms for each platform. Recently Xamarin.Forms was released which enables the use of common UI technology, XAML and work on cross platforms using native controls. Xamarin takes advantage of all the third party libraries written for .NET stack

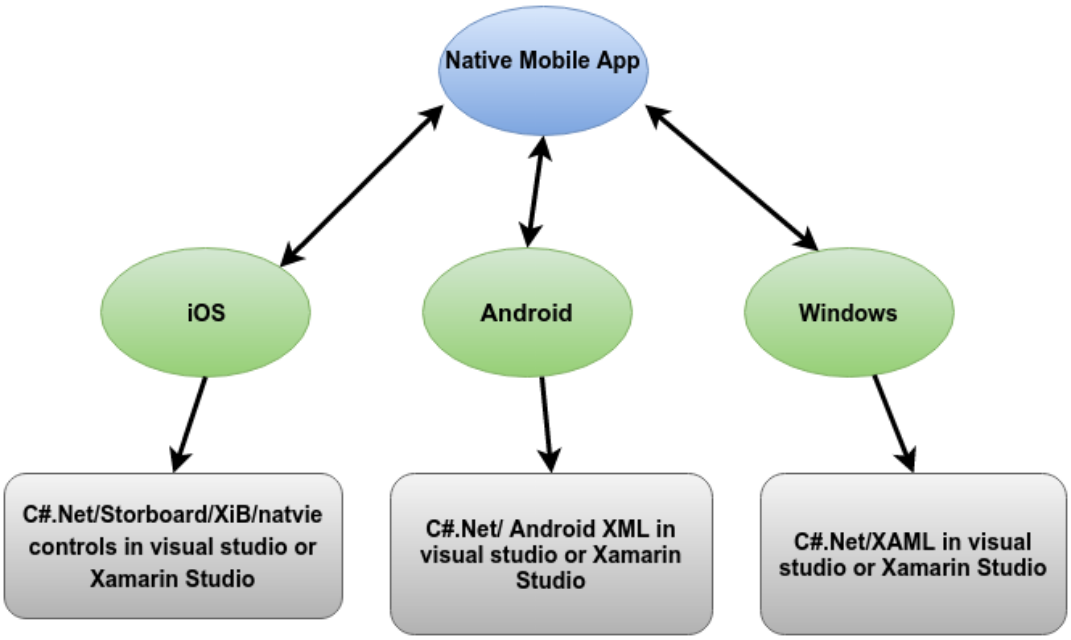


Figure 3.3: Xamarin Languages - Hybrid Moblie Application

3.2 Web Application

3.3 Desktop Application

3.4 General Tools

3.4.1 ReactiveX - libraries

Tremendous increase in need to respond and trigger actions based on data events in mobile application expanded the requirement of handling events in asynchronous manner. Component Object model was one of the previous technology that was used to execute asynchronous tasks. Recently Netflix introduced ReactiveX library which to process asynchronous task with robust architecture. ReactiveX is a library used for developing asynchronous and event based programs using observable sequence.This observer pattern promotes more real time experiences of the mobile or web application. Even modifying a single field in the current page triggers a instant save to back end, for example Twitter follow feature enabled for some profile can be immediately reflected to other connected user and so forth.

3.4.2 picasso - librares

3.4.3 HTML 5 - Languages

3.4.4 Swift - Languages

3.4.5 Java - Language

Chapter 4

Business Layer

4.1 Mobile Back End

4.1.1 Parse

4.1.2 Appcelerator

4.2 Web Server

4.2.1 NGINX

4.2.2 Apache HTTP Server

4.2.3 Microsoft IIS

4.3 Business Process Management

4.3.1 Java Business Process Management

4.4 Real Time Back End

4.4.1 Socket.IO

4.4.2 Firebase

4.5 Platform as a Service

4.5.1 Heroku

4.5.2 Google App Engine

Chapter 5

Data Layer

5.1 Cache

5.1.1 Ehcache

5.2 Message Queue

5.2.1 RabbitMQ

5.2.2 Kafka

5.3 Object Relational Mapper

5.3.1 Hibernate

5.3.2 Doctrine 2

5.4 Managed Memcache

5.4.1 Amazon ElastiCache

5.5 Mobile Database

5.5.1 Realm

5.5.2 Couchbase

5.6 In - Memory Database

5.6.1 Redis

5.6.2 Aerospike

5.7 Platform as a Service

5.7.1 Heroku

5.7.2 Google App Engine

Chapter 6

Data Store

6.1 Database

6.1.1 Mysql

6.1.2 MongoDB

6.1.3 PostgreSql

6.1.4 SQLite

6.1.5 Oracle

6.2 Graph Database

6.2.1 Neo4j

6.2.2 Titan

6.3 Mobile Database

6.3.1 Realm

6.3.2 Couchbase

6.4 In - Memory Database

6.4.1 Redis

6.4.2 Aerospike

6.5 Cloud Storage

6.5.1 Amazon S3

6.5.2 Amazon EBS

6.5.3 Google Cloud Storage

Chapter 7

Conclusion

In case you have questions, comments, suggestions or have found a bug, please do not hesitate to contact me. You can find my contact details below.

Jesper Kjær Nielsen
jkn@es.aau.dk
http://kom.aau.dk/~jkn
Fredrik Bajers Vej 7
9220 Aalborg Ø

Appendix A

Appendix A name

Here is the first appendix