ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

HỒ DANH CHUẨN

TÌM HIỂU ĐÁNH GIÁ CÁC FRAMEWORK PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG ĐA NỀN TẢNG

LUẬN VĂN THẠC SĨ KỸ THUẬT PHẦN MỀM

Hà Nội - 2017

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

HỒ DANH CHUẨN

TÌM HIỂU ĐÁNH GIÁ CÁC FRAMEWORK PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG ĐA NỀN TẢNG

Ngành: Công nghệ thông tin
Chuyên ngành: Kỹ thuật phần mềm
Mã số: 60480103

LUẬN VĂN THẠC SĨ KỸ THUẬT PHẦN MỀM

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. TRẦN THỊ MINH CHÂU

XÁC NHẬN CỦA CÁN BỘ HƯỚNG DẪN	**XÁC NHẬN CỦA CHỦ TỊCH HỘI ĐỒNG**

Hà Nội - 2017

LỜI CAM ĐOAN

Tôi xin cam đoan nội dung và những kết quả của luận văn tốt nghiệp này là do tôi tự nghiên cứu dưới sự hướng dẫn của TS. Trần Thị Minh Châu.

Trong toàn bộ nội dung của luận văn, những nội dung được trình bày là của cá nhân tôi hoặc được tổng hợp từ nhiều nguồn tài liệu khác. Tất cả các tài liệu tham khảo đều được trích dẫn rõ ràng ở phần cuối của luận văn.

Tôi xin cam đoan những lời trên là sự thật. Nếu sai tôi xin hoàn toàn chịu trách nhiệm.

Hà Nội, ngày tháng năm 2017

Học viên

Hồ Danh Chuẩn

LỜI CẢM ƠN

Đầu tiên tôi xin chân thành cảm ơn TS. Trần Thị Minh Châu đã tận tình hướng dẫn và đóng góp những ý kiến quý báu về chuyên môn cũng như các phương pháp nghiên cứu khoa học để tôi có thể thực hiện được luận văn tốt nghiệp thạc sĩ này. Cô cũng là tấm gương trong mọi mặt của cuộc sống để tôi học tập và noi theo.

Trong suốt quá trình học tập tại trường Đại học Công nghệ - Đại học Quốc gia Hà Nội, tôi xin chân thành cảm ơn các thầy, cô giáo đã cung cấp cho tôi những kiến thức hữu ích, bổ trợ cho quá trình phát triển sau này của tôi.

Cuối cùng tôi xin gửi lời cảm ơn tới gia đình tôi đã luôn ủng hộ tôi trên con đường học tập và nghiên cứu với nhiều khó khăn, vất vả. Mặc dù tôi đã cố gắng hết sức trong quá trình làm luận văn nhưng không thể tránh khỏi thiếu sót, rất mong nhận được những góp ý của các thầy cô.

Hà Nội, ngày tháng năm 2017

Học viên

Hồ Danh Chuẩn

MỤC LỤC

DANH MỤC CHỮ VIẾT TẮT

Chữ viết tắt	Diễn giải
IDE	Intergrated Development Environtment
MVC	Model View Controller
API	Application Programming Interface
SDK	Software Development Kit
HTML	Hyper Text Mark-up Language
DOM	Document Object Model
JNI	Java Native Interface
CLR	Common Language Runtime
CLI	Common Language Infrastructure
AOT	Ahead Of Time
JIT	Just In Time
IL	Intermediate Language
REST	Representation State Transfer
JSON	Javascript Object Notation
WCF	Windows Communication Foundation
MSIL	Microsoft Intermediate Language
GPU	Graphics Processing Unit
CPU	Central Processing Unit

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

CHƯƠNG 1: GIỚI THIỆU

1.1. Đặt vấn đề

Ngày nay, hệ sinh thái di động đóng một vai trò quan trọng trong chiến lược kinh doanh của hầu hết các doanh nghiệp. Các doanh nghiệp đều rất nghiêm túc trong việc phát triển và phát hành các ứng dụng phục vụ mục đích kinh doanh của họ. Tuy nhiên dù mục đích của các doanh nghiệp khi phát triển ứng dụng di động là gì thì một vấn đề luôn hiện hữu là việc lựa chọn cách tiếp cận nào là tốt nhất cho họ - công cụ hay phương pháp nào nên được triển khai để họ có thể đưa ứng dụng của mình đến đúng khách hàng, đúng thời điểm mà không cần quan tâm đến thiết bị, nền tảng người dùng cuối đang sử dụng.

Tuỳ thuộc vào ứng dụng thì có các yêu cầu các nhau về tính năng, trải nghiệm người dùng, vòng đời sản phẩm,.... Có ứng dụng có vòng đời ngắn chỉ phục vụ một thời điểm nhất định như là các ứng dụng phục vụ các sự kiện; có ứng dụng lại có vòng đời rất dài, gắn liền với sự tồn tại của doanh nghiệp; có ứng dụng yêu cầu tương tác nhiều với các thành phần của thiết bị; có ứng dụng cần người dùng tương tác nhiều lên thiết bị,... Tuy nhiên, tựu chung lại tất cả đều có một điểm chung: ứng dụng cần được xây dựng càng nhanh càng tốt, càng rẻ càng tốt và có thể chạy trên càng nhiều thiết bị càng tốt. Để lựa chọn một chiến lược phát triển ứng dụng tốt, các bên liên quan phải đánh giá đúng tiềm năng của ứng dụng, cân bằng giữa những yêu cầu và khoảng thời gian cần thiết để đưa ứng dụng ra thị trường.

Sự phát triển của mỗi nền tảng di động phụ thuộc rất nhiều vào trải nghiệm người dùng trên nền tảng đó. Apple chiếm được vị trí rất cao trong thị phần di động nhờ việc cung cấp cho người dùng trải nghiệm đồng nhất với chất lượng cao. Tương ứng, các nhà phát triển ứng dụng đa nền tảng cũng phải hướng đến việc tạo ra ứng dụng đáp ứng được yêu cầu của người dùng. Trong khi trải nghiệm của người dùng trở thành một mục tiêu chính của ứng dụng, thì có đến bốn đến năm nền tảng di động cần cân nhắc phát triển: iOS, Android, BlackberryOS, Windows Phone và Mobile Web để đưa sản phẩm của mình đến được với khách hàng. Trong đó hai nền tảng được chú trọng nhiều nhất là iOS và Android khi Blackberry và Microsoft đã dừng sản xuất các thiết bị chạy BlackberryOS và Windows Phone. Tất cả các hệ điều hành di động đều khác nhau về công nghệ, ngôn ngữ lập trình, cách thức lập trình và tiếp cận thì lập trình viên cần sỡ hữu được lượng kỹ năng đủ lớn để có thể có thể đưa sản phẩm đáp ứng được yêu cầu về độ phủ cũng như tính đa dạng theo cách truyền thống. Nếu như một ứng dụng cần phát triển trên nhiều hơn một hoặc hai nền tảng, thì việc đưa ra một sản phẩm có trải nghiệm trung thành trên tất cả các nền tảng sẽ tiêu tốn rất nhiều thời gian và công sức.

Để giúp các lập trình viên có cái nhìn rõ ràng hơn về các cách phát triển ứng dụng di động, đặc biệt là cách phát triển ứng dụng di động đa nền tảng, luận văn sẽ giới thiệu các cách tiếp cận trong việc phát triển ứng dụng di động. Bên cạnh đó, luận văn cũng đưa ra các so sánh về hai bộ khung phát triển ứng dụng đa nền tảng dựa trên các tiêu chí để các nhà phát triển có thể lựa chọn phương án phù hợp.

1.2. Mục tiêu và phạm vi nghiên cứu

Mục tiêu chính của luận văn là so sánh các phương pháp phát triển ứng dụng di động dựa trên các tiêu chí đánh giá cần thiết hiện nay để có thể đưa một ứng dụng di động thành công ra cộng đồng. Luận văn sẽ giới thiệu các cách tiếp cận phát triển ứng dụng di động hiện nay. Tiếp theo, nghiên cứu lựa chọn hai bộ khung phát triển ứng dụng di động đa nền tảng phổ biến hiện nay là Ionic và Xamarin để tiến hành đánh giá so sánh. Luận văn sẽ cung cấp cái nhìn tổng quan về hai bộ khung phát triển Ionic và Xamarin về kiến trúc, đặc điểm, ưu điểm và nhược điểm. Dựa trên các đặc điểm của hai bộ khung phát triển, luận văn sẽ dựa trên một số tiêu chí cần thiết khi phát triển ứng dụng di động như về giao diện, trải nghiệm người dùng, thiết kế bố cục ứng dụng, cộng đồng, hỗ trợ đa luồng, kiểm thử để so sánh hai bộ khung phát triển nói trên.

Luận văn được chia thành các phần như sau:

Chương hai cung cấp thông tin về các cách phát triển ứng dụng di động tập trung vào hai bộ khung phát triển ứng dụng là Ionic và Xamarin. Đầu tiên sẽ giới thiệu ba cách phát triển ứng dụng di động được sử dụng hiện nay. Sau đó, luận văn sẽ lần lượt giới thiệu từng bộ khung phát triển Ionic và Xamarin về các tính năng, kiến trúc, các đặc điểm nổi bật cũng như ưu điểm và nhược điểm của chúng.

Ở chương ba sẽ đưa ra bảng so sánh khả năng của từng bộ khung phát triển khi so sánh với việc phát triển ứng dụng native. Sau đó sẽ đi sâu hơn vào các tiêu chí mà các nhà phát triển nên quan tâm để lựa chọn bộ khung phát triển phù hợp.

Cuối cùng, ở chương bốn, luận văn sẽ trình bày việc xây dựng ứng dụng thử nghiệm để minh họa khả năng phát triển cũng như hiệu năng đối với hai nền tảng Ionic và Xamarin. Từ đó đưa ra một số khuyến nghị với các nhà phát triển trong việc lựa chọn bộ khung phát triển phù hợp với nhu cầu.

CHƯƠNG 2: NGHIÊN CỨU TỔNG QUAN VỀ PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG ĐA NỀN TẢNG

Hiện nay, các nhà phát triển có ba hướng chính để phát triển một ứng dụng di động, cụ thể là: native, web hoặc hybrid. Hiểu được về ưu và khuyết điểm của mỗi cách tiếp cận sẽ giúp nhà phát triển lựa chọn được hướng phát triển hợp lý nhất cho ứng dụng. Ở chương này, luận văn sẽ đề cập đến các cách tiếp cận này và phân tích cụ thể hai nền tảng là Ionic và Xamarin, đại diện cho hướng phát triển tương ứng là Hybrid và Native.

2.1. Các cách tiếp cận trong việc phát triển ứng dụng di động

Tương ứng với ba cách phát triển ứng dụng di động phổ biến là native, web và hybrid, chúng ta có thể gọi sản phẩm của mỗi cách tiếp cận này tương ứng là ứng dụng native, ứng dụng web và ứng dụng hybrid. Tuỳ thuộc vào kỹ năng của nhà phát triển, tuy nhiên nếu bỏ qua vấn đề chênh lệnh về trình độ hay về mặt thiết kế thì ứng dụng native sẽ đảm bảo ứng dụng có được trải nghiệm người dùng tốt nhất, có được sự đồng nhất với thiết bị nhất nhưng yêu cầu nhiều thời gian và cần có nhiều kỹ năng trên các nền tảng riêng biệt. Trong khi đó, ứng dụng web có thể được triển khai sớm nhưng cũng có nhiều hạn chế. Các ứng dụng hybrid ra đời trong một nỗ lực dung hoà được điểm mạnh của ứng dụng native và ứng dụng web. Các ứng dụng hybrid thường được tạo ra bằng cách kết hợp các native container với các ngôn ngữ lập trình web.

2.1.1. Phát triển ứng dụng native

Đây là cách tiếp cận đơn giản nhất trong việc phát triển ứng dụng nhưng cũng là cách tốn thời gian nhất. Mỗi nền tảng di động cung cấp cho các nhà phát triển một môi trường phát triển hoàn toàn khác nhau, và mỗi nền tảng có một phong cách giao diện và trải nghiệm người dùng tương đối khác nhau. Ví dụ như hệ điều hành Android có hàng phím ảo mà hệ điều hànhiOS không có, thanh tab của Android thường được đặt ở trên của ứng dụng trong khi iOS thì thanh tab được mặc định nằm ở dưới. Apple sử dụng Objective-C/Swift/Objective-C++ hoặc có thể là C++ để phát triển ứng dụng cho các thiết bị iOS, yêu cầu phải có máy tính chạy hệ điều hành MacOS, trong khi đó Android thường sử dụng Java/XML/Kotlin như ngôn ngữ lập trình đi kèm với Android Development Tools được tích hợp với các IDE. Microsoft thì dùng C#/XAML/.Net Framework để phát triển ứng dụng Windows Phone trên Visual Stuido IDE.

Việc sử dụng các công cụ phát triển đươc cung cấp bởi các nhà cung cấp nền tảng đảm bảo cho việc ứng dụng khi phát triển có thể sử dụng được hết tất cả các tính năng của nền tảng với tính nhất quán và hiệu năng cao. Tuy nhiên đây cũng là cách tốn kém nhất cả

về thời gian cũng như tiền bạc. Với mỗi nền tảng, cần phải có riêng một đội phát triển hoặc một đội phát triển với với các kỹ năng trên nhiều nền tảng. Rõ ràng các nhà đầu tư muốn duy trì được một ứng dụng phức tạp trên nhiều nền tảng khác nhau sử dụng phương pháp tiếp cận native thường phải tốn một khoảng chi phí rất lớn. Tuy nhiên với những ứng dụng có vòng đời dài và yêu cầu rất cao về độ ổn định và hiệu năng ứng dụng, thì đây là lựa chọn số một. Trong trường hợp chỉ tập trung vào một nền tảng duy nhất, thì đây cũng là cách rất tốt. Còn nếu muốn ứng dụng được ra mắt sớm, có hỗ trợ nhiều nền tảng, có nhiều ràng buộc về kinh phí, chúng ta có thể nên nghĩ đến các cách tiếp cận khác để xây dựng ứng dụng của mình.

2.1.2. Phát triển ứng dụng web

Cách tiếp cận thứ hai, tương đối phổ biến là cách phát triển sử dụng nền tảng web. Nhiều người nhìn vào sự thành công của công nghệ web trên desktop và tin rằng trên các thiết bị di động cũng sẽ có được sự thành công tương tự. Ứng dụng sử dụng cách tiếp cận này phụ thuộc vào WebKit trên mỗi nền tảng. Về cơ bản, các nhà phát triển sẽ xây dựng một trang web có giao diện và trải nghiệm người dùng giống hệt một ứng dụng trên nền tảng đó. Đây thường là cách phát triển ứng dụng đơn giản nhất, logic của ứng dụng có thể dễ dàng áp dụng chung giữa các nền tảng với nhau.

Các ứng dụng dựa trên nền tảng web thường sử dụng Javascript cho việc lập trình ứng dụng. Javascript là một ngôn ngữ cực kỳ phổ biến trong cộng đồng phát triển web, được hỗ trợ trên tất cả các trình duyệt di động hiện tại. Vì vậy ứng dụng gần như có thể đến tay tất cả người dùng. Bên cạnh đó, các nhà phát triển có thể đơn giản việc lập trình các ứng dụng di động trên nền web bằng cách sử dụng các bộ khung phát triển có sẵn như là AngularJS, JQuery Mobile,... đã được tích hợp sẵn rất nhiều thành phần giao diện và hầu hết đều tuân thủ theo mô hình MVC. Việc tích hợp các bộ khung phát triển Javascript cũng khá đơn giản khi mà chỉ cần thêm tập tin Javascript của bộ khung phát triển vào dự án và tham chiếu đến tập tin đấy trong đoạn mã cụ thể, không cần phải chỉnh sửa gì môi trường phát triển có sẵn.

Các ứng dụng di động dựa trên nền web có thể trông rất giống các ứng dụng native nếu như không tính đến các vấn đề về hiệu năng. Tuy nhiên, bởi vì người dùng cần truy cập các tính năng của ứng dụng thông qua trình duyệt web nên hầu hết các tính năng yêu cầu phải có kết nối mạng và người dùng có thể cảm nhận thấy độ trễ của mạng. Bên cạnh đó, các ứng dụng dựa trên nền tảng web còn có thể gặp các vấn đề về lưu trữ bộ nhớ, khả năng sử dụng các thành phần phần cứng trong thiết bị.

2.1.3. Phát triển ứng dụng hybrid

WebKit đã mở đường cho một cách tiếp cận mới trong việc phát triển ứng dụng di động: phát triển ứng dụng hybrid. Điều này có nghĩa là các ứng dụng được tạo ra dựa trên cách tiếp cận này là sự kết hợp giữa hai cách tiếp cận trên. Các ứng dụng hybrid không hoàn toàn là một ứng dụng native thực sự vì tất cả giao diện được dựng lên thông qua Webview thay cho các bộ khung hỗ trợ phát triển giao diện của từng nền tảng di động, nhưng cũng không phải là một ứng dụng dựa trên nền tảng web thực sự, khi mà chúng được đóng gói để phân phối và có thể sử dụng các API của từng nền tảng. Mô hình hybrid thay thế vỏ bọc WebKit bằng một thành phần khác gọi là được gọi là native container. Thành phần này sẽ chịu trách nhiệm chạy ứng dụng một cách độc lập trên mỗi nền tảng khác nhau.

Thay vì cung cấp một giao diện web giống nhau trên tất cả thiết bị như các ứng dụng web thường làm, các ứng dụng hybrid có thể cung cấp trải nghiệm người dùng khác cho mỗi nền tảng. Nền tảng Titanium của Appcelerator là một trong những framework sớm nhất áp dụng cách tiếp cận này. Hoặc Ionic hiện tại đang nổi lên như một bộ khung phát triển rất có triển vọng trong việc phát triển ứng dụng đa nền tảng. Với cách tiếp cận này, nhà phát triển có thể chọn một bộ kỹ năng cần thiết là có thể xây dựng ứng dụng trên nhiều nền tảng khác nhau. Lợi thế của phương pháp này so với cách phát triển dựa vào WebKit là ứng dụng hoàn toàn có khả năng truy cập vào các thành phần phần cứng của thiết bị. Hiệu năng của các bộ khung phát triển sử dụng cách tiếp cận này thường phụ thuộc vào cách bộ khung phát triển được tổ chức và cách tương tác với hệ điều hành. Tuy nhiên các framework này thường không miễn phí hoàn toàn, có thể đi theo mô hình mua bản quyền, trả phí định kỳ hoặc freemium. Bên cạnh đó sử dụng phương pháp này các nhà phát triển thường phục thuộc vào các bên thứ ba để có thể nhanh chóng đưa ứng này ra thị trường (ví dụ như là các trình cắm cho Cordova/ PhoneGap). Một điều rất quan trọng cần lưu ý, những bộ khung phát triển này là các bộ khung phát triển đa nền tảng chứ không phải mọi nền tảng.

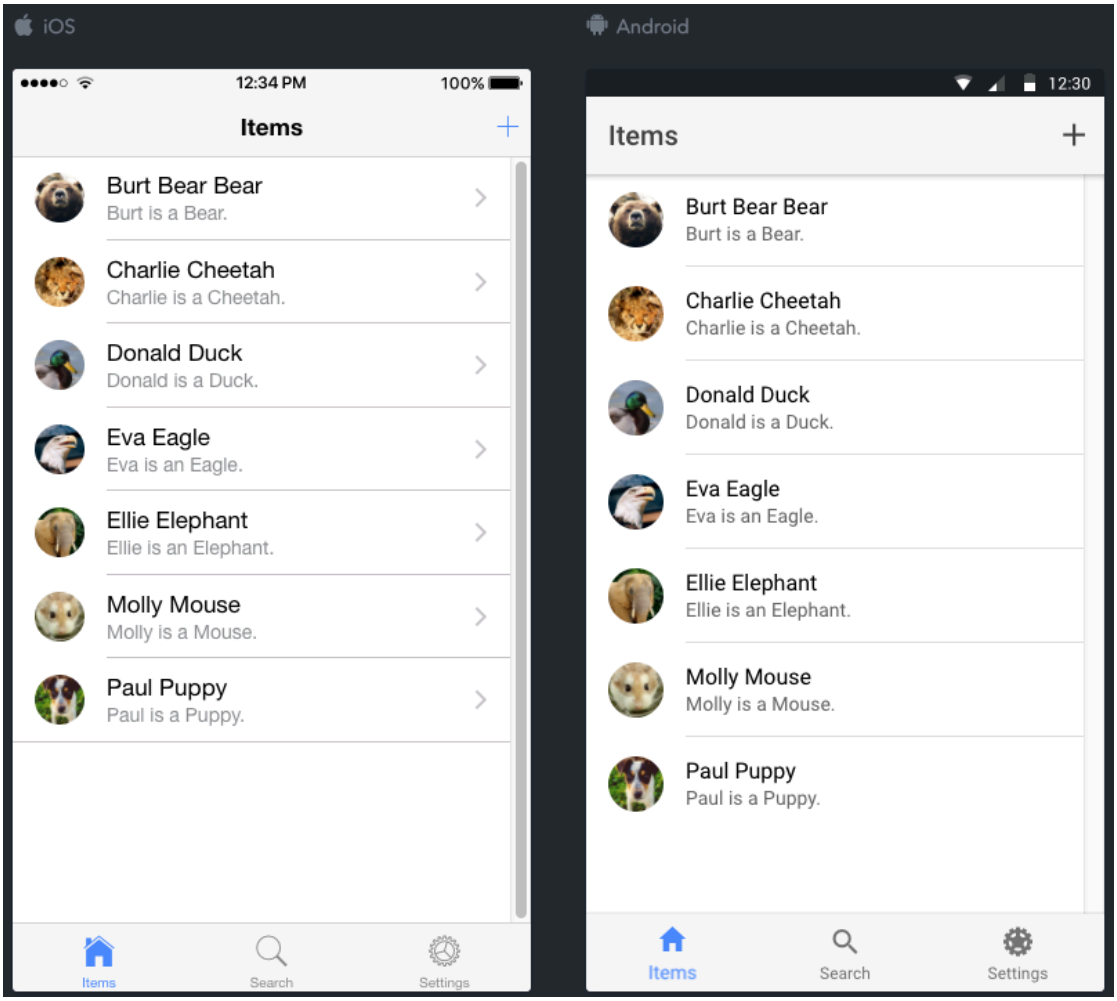
2.2. Ionic framework

2.2.1. Giới thiệu

Ionic là một bộ khung phát triển ứng dụng di động mã nguồn mở được ra đời vào năm 2013, được phát triển bởi công ty Drifty. Được lấy cảm hứng từ sự thành công của nền tảng web trên desktop, các nhà phát triển mong muốn Ionic cũng đạt được thành công tương tự trên các nền tảng di động.

Được xây dựng dựa trên nền tảng AngularJS và Apache Cordova (mặc định), Ionic cung cấp công cụ để phát triển các ứng dụng hybrid trên các hệ điều hành di động sử dụng các công nghệ Web như CSS, HTML5, SASS và Javascript. Ứng dụng được xây dựng dựa vào các công nghệ web như trên và phân phối thông qua các chợ ứng dụng mặc định trên các thiết bị nhờ sự trợ giúp của một trình đóng gói bản địa (Native wrapper).

Ionic có thể được xem như một bộ khung phát triển giao diện cho front-end. Nó sẽ chịu trách nhiệm về giao diện và cách người dùng tương tác với ứng dụng. Bên cạnh đó, Ionic còn hỗ trợ rất nhiều các thành phần native trong các nền tảng di động, cung cấp sẵn một số animation có sẵn. Có một điểm khác biệt của Ionic với các framework khác là các thành phần giao diện trong Ionic có vẻ ngoài và cách hoạt động rất giống với các thành phần giao diện tương tự trong các hệ điều hành di động và tất nhiên điều này diễn ra hoàn toàn tự động, nhà phát triển không cần phải chỉnh sửa bất cứ thứ gì. Ví dụ như thanh tab trong Ionic



Hình 2.1: Thanh Tab bar trong Ionic, biểu diễn trên Android và iOS

2.2.2. Kiến trúc

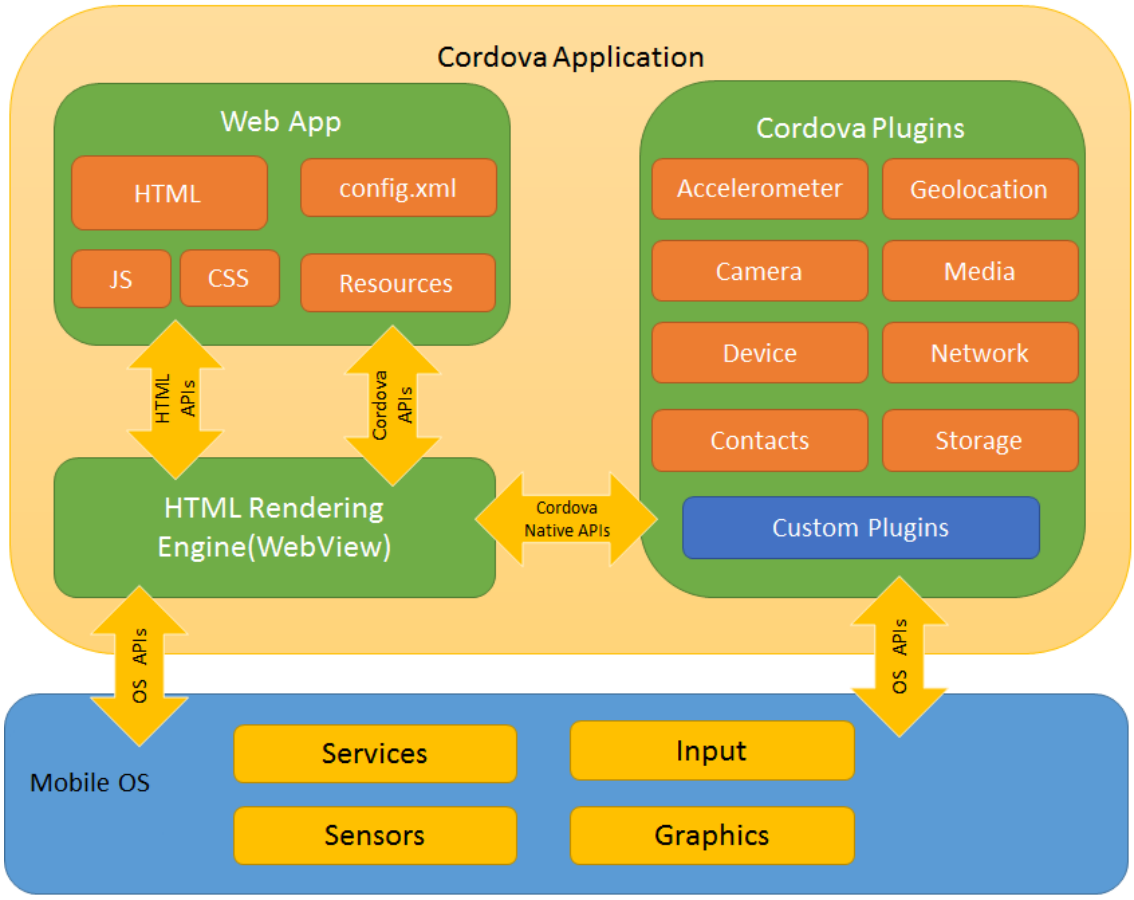
Bộ khung Ionic bao gồm ba thành phần chính:

Một bộ khung phát triển giao diện sử dụng nền tảng SASS được thiết kế và tối ưu cho giao diện trên các thiết bị di động.

Một bộ khung phát triển AngularJS để xây dựng ứng dụng

Một trình biên dịch để đóng gói và biên dịch các thành phần HTML, CSS, JS thành ứng dụng trên các thiết bị di động

Ionic được xây dựng với trình biên dịch mặc định là Cordova. Cordova có nhiệm vụ đóng gói các thành phần HTML, CSS, JS thành ứng dụng trên các hệ điều hành di động. Bên cạnh đó, Cordova còn cung cấp một kiến trúc trình cắm (plugin) cho phép các ứng dụng sử dụng javascript có thể sử dụng các tính năng native trên thiết bị thông qua các đoạn mã Javascript chạy trên trình duyệt. Một ứng dụng Ionic bao gồm một phần chính là ứng dụng web bao gồm mã nguồn giao diện, logic, tài nguyên của ứng dụng. WebView sẽ chịu trách nhiệm biên dịch xây dựng ứng dụng dựa trên mã nguồn trong thành phần ứng dụng web. Để tương tác với các thành phần nền tảng, ứng dụng Ionic sẽ tương tác trực tiếp qua webview hoặc qua các trình cắm được tích hợp vào ứng dụng. Vì vậy, một ứng dụng Ionic sẽ có kiến trúc tương tự với một ứng dụng Cordova như sau:



Hình 2.2: Cấu trúc ứng dụng Ionic/Cordova[1]

Apache Cordova bao gồm các thành phần:

Mã nguồn cho các thành phần native container chỗ mỗi nền tảng hỗ trợ. Các thành phần này sẽ phụ trách việc dựng các ứng dụng Cordova trên thiết bị

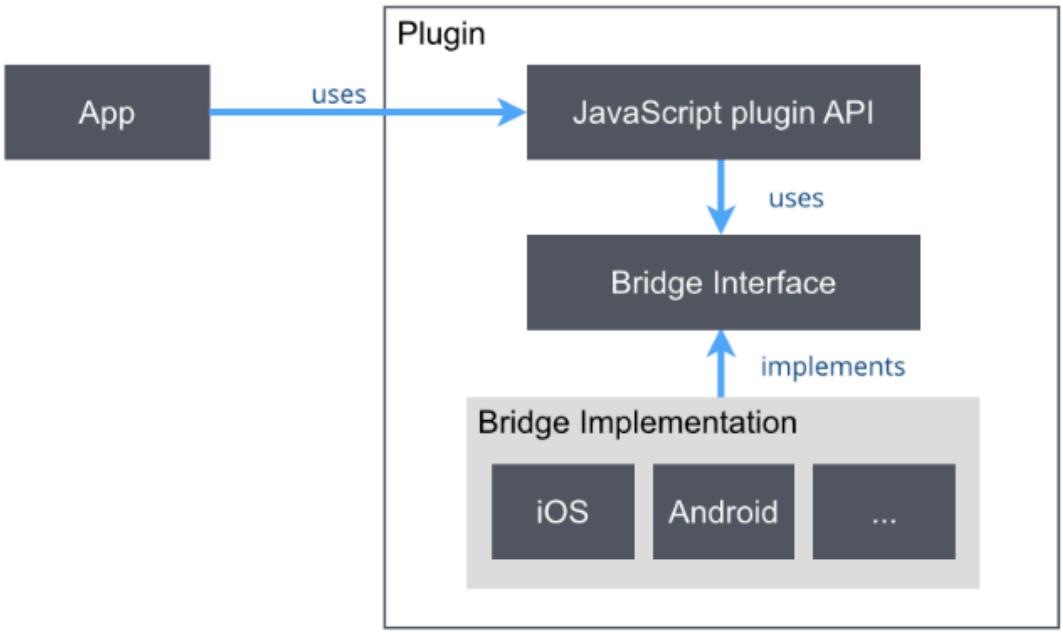
Tập hợp các giao diện lập trình ứng dụng được đóng gói dưới dạng các plugin cho phép các ứng dụng ở trong container có thể sử dụng các tính năng trên thiết bị mà bình thường các trình duyệt không hỗ trợ

Tập hợp các công cụ để quản lý quá trình tạo dự án, quản lý vòng đời các plugin, xây dựng ứng dụng (sử dụng native SDK), và kiểm thử ứng dụng trên trình mô phỏng.

Các trình cắm trong ứng dụng Ionic/Cordova đảm nhiệm vai trò tương tác với một hoặc nhiều các API nền tảng trên ứng dụng. Các trình cắm luôn bao gồm hai phần. Phần thứ nhất là các API được viết bằng Javascript chạy trên các thành phần WebKit để các ứng dụng hybrid có thể sử dụng. Phần thứ hai là các API được viết bằng các ngôn ngữ lập

[1] *https://cordova.apache.org/docs/en/latest/guide/overview/*

trình tương ứng trên các nền tảng. Thành phần này phụ trách gọi các API native trên thiết bị. Kiến trúc của một plugin điển hình sẽ như sau:



Hình 2.3: Kiến trúc của một Cordova plugin[2]

Khi khởi tạo một ứng dụng trên Ionic phiên bản 2.0, cấu trúc của dự án đó sẽ bao gồm một số thành phần được khởi tạo sẵn như sau:

- Thư mục *platform* chứa các dự án iOS and Android của nhà phát triển. Về cơ bản, nhà phát triển không cần thiết phải làm việc với các thư mục này trừ khi cần thực hiện một số thủ thuật cụ thể trên từng nền tảng hoặc là đưa ứng dụng lên cửa hàng ứng dụng.
- Thư mục *hooks* được thiết kế cho các tác vụ tuỳ chỉnh. Thường thì thư mục này có nhiều lợi ích hơn đối với các dự án lớn cần tự động hoá quá trình chạy và chỉnh sửa mã nguồn.
- Thư mục merges dùng để ghi đè các file cho các nền tảng cụ thể. Ví dụ như nếu có hai tệp tin ở hai đường dẫn khác nhau là *'merges/ios/something.js'* và *'www/something.js'*. Khi triển khai ứng dụng trên nền tảng iOS thì tệp tin *something.js* sẽ được thay thế bởi tệp tin trong thư mục *merges*. Thư mục này rất hiệu quả nếu người dùng muốn tuỳ chỉnh một số thứ cho một nền tảng cụ thể.

[2] *https://blog.codecentric.de/en/2014/11/ionic-angularjs-framework-on-the-rise/*

- Thư mục *plugins* là nơi Cordova chứa các plugin mà nhà phát triển thêm vào ứng dụng.
- *SCSS* là thư mục để chứa các file SASS. Việc sử dụng SASS là không bắt buộc khi nhà phát triển xây dựng ứng dụng với Ionic, tuy nhiên bản thân Ionic cũng được xây dựng trên nền tảng SASS. Nhà phát triển có thể nhanh chóng thay đổi nhiều style mặc định của Ionic thông qua SASS mà không cần phải thêm vô số các ghi đè CSS.
- *www* là thư mục làm việc chính. Đây là nơi sẽ viết các đoạn mã chính cho ứng dụng của mình.
    - Thư mục *CSS* chứa các file CSS tự xây dựng hoặc các file SCSS được sinh ra.
    - *Img* là nơi đặt các ảnh của ứng dụng.
    - Mặc định, Ionic tạo sẵn cho các nhà phát triển một số tập tin javascript như một cách để khuyến khích cấu trúc dự án theo đúng kiến trúc MVC. ‘app.js' định nghĩa luồng của ứng dụng; ‘controllers.js’ chứa các controller của các trạng thái ứng dụng; ‘services.js' chứa các service ứng dụng có thể sử dụng như là kết nối đến các API của bên thứ ba để lấy thông tin ứng dụng cần thiết; ‘directives.js' chứa các AngularJS directive. Tuy nhiên, không có gì bắt buộc nhà phát triển phải tự cấu trúc ứng dụng của mình.
    - *Libs* chứa các thư viện ứng dụng sử dụng.
    - *Templates* là nơi chứa các view. Mặc định thì dự án đã có một file index.html trong thư mục ‘www’, nhưng bên cạnh đó, chúng ta cũng có rất nhiều các mẫu có sẵn có thể thêm vào động trong ứng dụng.

Sang phiên bản thứ ba, cấu trúc một ứng dụng Ionic đã trở nên đơn giản rất nhiều nhằm giúp các nhà phát triển tập trung hơn vào các thành phần quan trọng trong việc phát triển ứng dụng

- Thư mục *src* chứa hầu hết mã nguồn của ứng dụng. Đây là nơi mà các lập trình viên dành đa số thời gian để làm việc. Khi các lập trình viên chạy câu lệnh ionic serve thì các mã nguồn trong thư mục src sẽ được biên dịch sang các đoạn mã Javascript mà trình duyệt có thể hiểu được. ĐIều đó có nghĩa rằng các lập

trình viên có thể sử dụng các ngôn ngữ bậc cao như Typescript và nó sẽ được biên dịch xuống các dạng thấp hơn Javascript tương ứng với các trình duyệt.

- Trong thư mục src có một số tệp được Ionic tạo sẵn phục vụ cho việc chạy ứng dụng.
  - src/index.html là điểm truy cập đầu tiên khi một ứng dụng Ionic chạy. Tệp này có nhiệm vụ cấu hình một số thành phần cần thiết cho ứng dụng. Thông thường, các lập trình viên sẽ không thay đổi nhiều đối với file này
  - src/app/app.html. Tệp nàyy định nghĩa mẫu chính của ứng dụng. Người dùng sẽ thấy được giao diện ứng dụng đầu tiên được định nghĩa ở đây
  - src/app/app.module.ts: đây là nơi các lập trình viên định nghĩa các thành phần của một ứng dụng. Mặc định Ionic đã định nghĩa sẵn một số thành phần cần thiết cho ứng dụng như là BrowserModule, IonicModule,...

2.2.3. Điểm nổi bật

a) Cordova

Apache Cordova là một bộ khung phát triển rất phổ biến để xây dựng ứng dụng di động sử cụng CSS3, HTML và Javascript. Có khá nhiều bộ khung phát triển ứng dụng di động được xây dựng dựa trên Cordova như là Ionic, Monaca, Intel XDK,.... Về cơ bản, các ứng dụng Cordova sử dụng CSS và HTML5 để dựng giao diện và Javascript cho logic. HTML5 cung cấp khả năng truy cập vào các thành phần phần cứng của thiết bị như là máy ảnh, gia tốc kế hay GPS. Tuy nhiên khả năng hỗ trợ HTML5 của các trình duyệt là không đồng nhất, đặc biệt là với các thiết bị Android cũ. Vì vậy, Cordova sử dụng một cơ chế nhúng các đoạn mã HTML5 vào native Webview trên thiết bị, sử dụng giao diện hàm ngoại lai (foreign function interface) để truy cập vào các tài nguyên trên thiết bị.

Nhắc đến Cordova không thể không nhắc đến các trình cắm (plug-in). Chúng cho phép các nhà phát triển có thể mở rộng thêm nhiều tính năng cho bộ khung được sử dụng thông qua các Javascript API. Điều đó giúp cho các nhà phát triển có thể tạo một kênh tương tác trực tiếp giữa các trang HTML5 và lớp native của nền tảng.

b) Mã nguồn mở

Ionic là một nền tảng miễn phí và mã nguồn mở được phát hành theo giấy phép MIT. Giấy phép MIT là một trong những giấy phép cho phép sử dụng mã nguồn tự do nhất. Điều này cho phép các nhà phát triển tự do sử dụng Ionic trên tất cả các sản phẩm. Mục đích của các nhà phát triển nên bộ khung Ionic không chỉ là xây dựng một bộ khung

phát triển ứng dụng di động đa nền tảng mà còn mong muốn xây dựng một nền tảng để chia sẻ các kiến thức cho các nhà phát triển, nơi chứa đụng các mẫu thiết kế tốt nhất cho các ứng dụng di động. Điều này thể hiện ở việc Ionic cố gắng nâng cao trải nghiệm của các ứng dụng được xây dựng dựa trên nó bằng cách cung cấp một giao diện thật tự nhiên trên các hệ điều hành khác nhau. Ionic sử dụng Cordova và Angular để cấu trúc dự án.

c) Đánh dấu và trình diễn (Mark up và presentation)

Đây là một lợi thế khá lớn của Ionic khi so sánh với các bộ khung phát triển khác. Ionic luôn cố gắng cung cấp cho các nhà phát triển một trải nghiệm thật tương đồng với các ứng dụng gốc trên các hệ điều hành. Kể từ phiên bản beta 7, ứng dụng được tạo ra bởi Ionic sẽ có bố cục phụ thuộc vào nền tảng mình sử dụng. Ionic chủ động xây dựng các thành phần giao diện cho ứng dụng theo hướng dẫn giao diện trên từng nền tảng. Ionic không những tập trung vào hình ảnh mà còn tập trung vào các hành vi trên từng nền tảng. Ví dụ như hành vi dội lại mặc định khi cuộn trên iOS nhưng không xuất hiện trên Android. Bên cạnh việc cung cấp cho nhà phát triển các mẫu thiết kế (design pattern) có sẵn tốt, Ionic còn cung cấp khả năng kiểm soát tối đa trong quá trình phát triển ứng dụng. Các thành phần CSS có thể hoạt động độc lập không cần sự can thiệp của nhà phát triển nhưng hoàn toàn có thể tuỳ biến để thoả mãn nhu cầu riêng biệt khi cần thiết. Các nhà phát triển có thể thêm các thành phần CSS mới hoặc thay đổi các giá trị mặc định của các thành phần giao diện trong bộ khung phát triển.

d) AngularJS

AngularJS là một dự án mã nguồn mở của Google đã trở nên phổ biến đối với những nhà phát triển ứng dụng web thời gian gần đây. AngularJS cung cấp cho người phát triển khả năng hoàn thiện ứng dụng web một cách nhanh chóng và tạo cấu trúc hợp lí cho ứng dụng. Các bộ khung phát triển cho ứng dụng web dùng Javascript như AngularJS cho phép xây dựng các ứng dụng phức tạp ngay trong trình duyệt mà không cần dựa trên server. Đây chắc chắn là một lợi thế cho các nhà phát triển web khi phát triển ứng dụng hybrid, bởi trình duyệt là nền tảng để tạo nên những ứng dụng này. Nếu nhà phát triển quen thuộc với AngularJS hoặc các bộ khung phát triển Javascript khác như Ember, việc làm quen với việc phát triển ứng dụng di động bằng Ionic sẽ dễ dàng hơn.

Ionic sử dụng AngularJS để tạo ra một bộ khung phát triển tốt nhất cho việc xây dựng các ứng dụng di động đa nền tảng, nó cung cấp một bộ giao diện người dùng (UI) mã nguồn mở miễn phí đi cùng với các tính năng của AngularJS.

Việc xây dựng ứng dụng dựa trên AngularJS đòi hỏi mã nguồn phải có khả năng mở rộng cao để bổ sung các tính năng mới. Tuy nhiên với Ionic, người ta có thể tái sử dụng các chức năng trong ứng dụng trên các nền tảng khác nhau đồng thời vẫn có thể tùy chỉnh giao diện người dùng cho mỗi nền tảng riêng biệt. Các thành phần trong Ionic như danh sách, slide,.. chính là các directive(các thuộc tính của thẻ HTML dùng trong Angular) của AngularJS. Đó là lí do khiến cho Ionic và AngularJS kết hợp rất tốt với nhau.

e) Hiệu suất

Một vấn đề thường gặp đối với các bộ khung phát triển ứng dụng di động đa nền tảng dựa trên nền Web là hiệu suất. Hiện tượng này là kết quả của nhiều nguyên nhân trong đó nguyên nhân chính là hiệu năng của các thành phần Webkit, giới hạn tài nguyên trên các thiết bị di động và cách bộ khung phát triển sử dụng các tài nguyên chưa đủ tốt. Ionic ngay từ lúc bắt đầu được xây dựng đã rất chú ý đến hiệu suất của ứng dụng. Ionic có một giao diện ổn định tốc độ tốt, với các hiệu ứng chuyển động được áp dụng kỹ thuật tăng tốc phần cứng (hardware accelerating) và tối giản các thao tác với DOM. Mặc dù Ionic mặc định không có jQuery, tuy nhiên người dùng có thể dễ dàng thêm vào nếu cảm thấy cần thiết.

Một điểm thú vị của Ionic đồng thời giúp tăng hiệu suất hoạt động của nó, đó là Ionic không cố gắng tự mình thực hiện tất cả công việc. Ví dụ như việc đóng gói ứng dụng, Ionic sử dụng Cordova và tận dụng cấu trúc thư mục mặc định của nó.

2.2.4. Ưu điểm và nhược điểm

a) Ưu điểm

Mã nguồn mở, miễn phí cho tất cả mọi người. Điều này giúp cho Ionic dễ dàng tiếp cận được với số lượng lớn các nhà phát triển, giúp cho Ionic có một cộng đồng đông đảo các nhà phát triển từ khắp nơi trên thế giới. Mã nguồn mở giúp cho Ionic có sức mạnh của cộng đồng, các vấn đề hay lỗi xảy ra trong Ionic có thể được xem xét giải quyết bởi rất nhiều nhà phát triển chứ không bị giới hạn bởi một số lượng nhà phát triển nếu như Ionic là một sản phẩm phần mềm đóng.

Đa nền tảng: các nhà phát triển có thể mang ứng dụng chạy trên nhiều nền tảng chỉ dựa trên một mã nguồn duy nhất. Điều này có thể giúp giảm thời gian phát triển và giảm thời gian và kỹ năng cần thiết để có thể đưa một ứng dụng tương tự lên các nền tảng khác nhau sử dụng cách truyền thống.

Có sẵn các thành phần giao diện có thể sử dụng ngay và tuỳ biến nếu cần

Trải nghiệm người dùng tương tự với các ứng dụng native trên từng nền tảng.

HTML, Javascript và CSS là một chuẩn mực chung của các nhà phát triển web. Ionic giúp các nhà phát triển tận dụng các kỹ năng này mà không cần phải đào tạo lại. Điều này giảm thiểu thời gian phát triển và chi phí phát triển ứng dụng, giúp ứng dụng nhanh chóng được đưa ra thị trường

b) Nhược điểm

Mã nguồn mở cũng có thể coi là nhược điểm của Ionic. Việc đưa Ionic thành một bộ khung phát triển mã nguồn mở, hoàn toàn miễn phí cho mọi người không đảm bảo được sự thành công của chính bản thân Ionic do không có sự ràng buộc với người sử dụng. Tuy nhiên Ionic framework được điều hành bởi công ty có các dịch vụ liên quan chủ yếu đến bản thân các sản phẩm tạo ra bởi Ionic framework nên nhược điểm này có thể không quá lớn.

Đa nền tảng cũng là một nhược điểm có thể được tính đến. Với các bộ khung phát triển đa nền tảng, các nhà phát triển phải hỗ trợ nhiều nền tảng trên một mã nguồn duy nhất. Trong một số trường hợp, các nhà phát triển sẽ cần thực hiện vài tuỳ chỉnh để ứng dụng có thể hoạt động khác đi một chút ở các nền tảng khác nhau. Thực tế, điều này có thể tốn thời gian nếu như gặp phải một vấn đề lạ hoặc cần phải phá bỏ các cách triển khai mặc định của Ionic khi cố gắng làm trang web trông giống như một ứng dụng.

Vấn đề hiệu suất của các bộ khung phát triển ứng dụng di động đa nền tảng sử dụng các ngôn ngữ script như Javascript đã được nhắc đến từ rất lâu. Đây là một trong những vấn đề được các nhà phát triển khá quan tâm trong việc lựa chọn cách phát triển ứng dụng trên các thiết bị di động. Gần đây, với sự phát triển của phần cứng trên các thiết bị di động và nỗ lực tối ưu các thành phần WebKit trên các hệ điều hành của các nhà phân phối, thì vấn đề hiệu suất của các ứng dụng phát triển bằng các bộ khung phát triển sử dụng các ngôn ngữ script đã được cải thiện rất nhiều. Ionic có thể xử lý tốt đa số các ứng dụng không cần xử lý quá nặng về mặt hình ảnh như 3D, videos hay các ứng dụng trò chơi. CSS3 đã hỗ trợ tăng tốc phần cứng giúp cho các nhà phát triển hiếm khi gặp vấn đề với hiệu năng của ứng dụng trừ khi phải làm một số việc ngoài tầm của CSS và chạy trên một số thiết bị đời cũ như iPhone 4 trở về trước. Vấn đề về hiệu nặng ứng dụng trên Android thì rõ ràng hơn so với trên iOS bởi vì cơ chế của hệ điều hành và sự phân mảnh của các thiết bị Android. Để giải quyết điều này, Ionic cung cấp khả năng cho phép tích hợp một trình duyệt khác có hiệu năng cao hơn (Crosswalk) vào ứng dụng để đáp ứng yêu cầu về tốc độ xử lý. Tuy nhiên việc tích hợp thêm Crosswalk vào ứng dụng sẽ làm tăng

kích thước ứng dụng khá nhiều. Tóm lại, Ionic không phù hợp với các ứng dụng nặng cần xử lý hoặc truyền nhiều dữ liệu trên nhiều luồng khác nhau.

Ionic là một bộ khung phát triển sử dụng nền tảng web nên các nhà phát triển có thể dùng trình duyệt để gỡ lỗi các ứng dụng. Tuy nhiên việc gỡ lỗi có thể trở nên khó khăn nếu như các lỗi liên quan đến các thành phần thuộc về các nền tảng như là các thành phần phần cứng vì các trình cắm hoặc một số thành phần không được thiết kế để chạy như là một ứng dụng web.

Kiến trúc plugin là một điểm mạnh của Ionic/Cordova. Tuy nhiên nó cũng làm cho các nhà phát triển bị phụ thuộc vào các plugin này. Điều gì sẽ xảy ra nếu như ứng dụng cần truy cập vào một thành phần phần cứng mới, chưa có plugin nào có sẵn thực hiện điều đó. Hoặc là khi gặp lỗi trên một plugin nào đó. Để giải quyết vấn đề này, các nhà phát triển buộc phải có kỹ năng về cả Javascript và native để có thể giải quyết vấn đề. Điều này có thể mất đi những lợi thế lớn nhất của Ionic trong việc giảm thời gian và chi phí phát triển ứng dụng.

2.3. Xamarin framework

Bộ khung phát triển ứng dụng đa nền tảng Xamarin, tiền thân là dự án mã nguồn mở Mono, được giới thiệu vào ngày 16/5/2011. Xamarin ban đầu thuộc công ty Xamarin, tuy nhiên đã được tập đoàn Microsoft mua lại vào 24/2/2016. Điều này đánh dấu một cột mốc trong vòng đời của bộ khung phát triển này khi Microsoft tuyên bố sẽ mở mã nguồn của Xamarin SDK, cung cấp nó như là một thành phần trong bộ công cụ Microsoft Visual Studio. Các người dùng đang sử dụng bộ công cụ Visual Studio dành cho doanh nghiệp cũng sẽ được sử dụng các tính năng dành cho doanh nghiệp của Xamarin miễn phí. Bên cạnh đó, Microsoft cũng phát hành toàn bộ dự án Mono dưới giấy phép MIT và cũng phát hành toàn bộ các phần mềm Xamarin SDK khác thông qua .NET Foundation dưới giấy phép MIT.

Xamarin là bộ khung phát triển ứng dụng cho phép các nhà phát triển xây dựng ứng dụng trên các nền tảng Android, iOS và Windows sử dụng một ngôn ngữ lập trình chính là C#. Xamarin cung cấp các lớp thư viện, runtime thực thi trên cả ba nền tảng iOS, Android và Windows Phone, trong khi vẫn biên dịch native (không sử dụng các trình thông dịch) và đảm bảo hiệu suất ứng dụng kể cả các ứng dụng yêu cầu khả năng xử lý nặng như các ứng dụng trò chơi. Mặc dù không sử dụng các ngôn ngữ lập trình tương ứng trên các hệ điều hành khác nhau nhưng Xamarin được xem như là một bộ khung phát triển ứng dụng gốc (native framework). Trong việc phát triển ứng dụng di động,

“nativeness" được định nghĩa là hệ sinh thái mà các nhà cung cấp (Apple hoặc Google) chọn để phát triển ứng dụng trên hệ điều hành của họ. Một ví dụ là hệ điều hành Android. Android là sự kết hợp giữa Linux và Java SDK, Java sử dụng JNI để gọi các API C/C++ và cách Xamarin làm việc cũng hoàn toàn tương tự. Xamarin cũng sử dụng một bộ giao diện để tương tác với các API C/C++ được gọi là PInvoke (Platform Invoke), một công nghệ .NET(CLR, CLI) cho phép gọi các API gốc bằng C#. Một điều đặc biệt của Xamarin là nó có hỗ trợ một phần mở rộng có tên là Microsoft's Razor Extension, cho phép các nhà phát triển xây dựng ứng dụng hybrid tận dụng sức mạnh của nền tảng C# trong Xamarin.

Xamarin cung cấp hai sản phẩm là Xamarin.iOS và Xamarin.Android. Cả hai đều được xây dựng dựa trên nền tảng Mono. Đối với nền tảng iOS, trình biên dịch AOT biên dịch ứng dụng iOS trực tiếp thành mã máy ARM. Đối với nền tảng Android, trình biên dịch của Xamarin biên dịch mã nguồn ứng dụng thành IL, sau đó trình biên dịch JIT của Android sẽ chịu trách nhiệm biên dịch các đoạn mã IL thành mã máy khi ứng dụng chạy. Trong cả hai trường hợp, ứng dụng Xamarin sẽ tối ưu thời gian chạy bằng cách tự động xử lý các vấn đề liên quan đến quản lý bộ nhớ, thu dọn rác (garbage collection), các tác vụ bên trong nền tảng (platform interops), v.v…

2.3.1. Điểm nổi bật

- *Liên kết với các SDK của các nền tảng*

Xamarin cung cấp khả năng tương tác với hầu hết các SDK nền tảng của cả iOS và Android. Bên cạnh đó, các liên kết (bindings) này đều là liên kết mạnh, điều đó có nghĩa là các API này dễ điều hướng và sử dụng. Đảm bảo tối ưu thời gian biên dịch kiểm tra kiểu và thời gian phát triển. Điều này giúp làm giảm thiểu lỗi trong thời gian chạy và nâng cao chất lượng của ứng dụng.

- *Tương tác với Objective-C, Java, C/C++*

Xamarin cho phép tương tác với các thư viện Objective-C, Java, C và C++ trực tiếp. Điều này giúp cho các nhà phát triển có thể tận dụng sức mạnh của các thư viện của bên thứ ba có sẵn trên các nền tảng iOS và Android được viết bằng Objective-C, Java hay C/C++. Bên cạnh, Xamarin còn cung cấp các dự án liên kết cho phép các nhà phát triển dễ dàng liên kết các thư việc Objective-C và Java sử dụng các cú pháp khai báo.

- *Sử dụng ngôn ngữ hiện đại*

C# được xem như là một điểm nổi bật của Xamarin khi mà C# có khá nhiều điểm cải tiến đáng chú ý so với các ngôn ngữ cũ hơn như Objective-C hay Java như là Dynamic Language Features, Functional Constructs như là Lambdas, LINQ, Parallel Programming, Generics, v.v…

– *Các lớp thư viện cơ bản*

Các ứng dụng Xamarin có thể sử dụng một số lượng lớn các thư viện .NET được xây dựng sẵn cung cấp các tính năng mạnh mẽ như XML, Database, Serialization, IO, String và Networking, v.v…. Bên cạnh đó, các đoạn mã C# có thể được biên dịch để có thể dùng trong các ứng dụng, cung cấp khả năng truy cập hàng ngàn thư viện mà chưa được triển khai trong các lớp thư viện cơ bản.

– *IDE hiện đại*

Một IDE tốt sẽ giúp đỡ rất nhiều cho các nhà phát triển trong việc phát triển ứng dụng. Và Xamarin Studio trên Mac OS X hay Visual Studio trên Windows là một trong những IDE tốt nhất hiện nay. Nó cũng cấp rất nhiều tính năng cho nhà phát triển như tự động hoàn thiện code, kiểm soát phiên bản (version control), quản lý dự án, các mẫu ứng dụng, v.v…

– *Hỗ trợ đa nền tảng di động*

Xamarin hỗ trợ ba nền tảng di động phổ biến hiện nay là iOS, Android và Windows Phone. Các ứng dụng viết bằng Xamarin có thể chia sẻ tới 90% mã nguồn, và thư viện Xamarin.Mobile cung cấp các API thống nhất để có thể sử dụng các tài nguyên trên tất cả ba nền tảng. Điều này sẽ giúp tiết kiệm thời gian và công sức để đưa ứng dụng ra thị trường hướng tới các nền tảng ứng dụng di động phổ biến nhất hiện nay.

2.3.2. Kiến trúc [3]

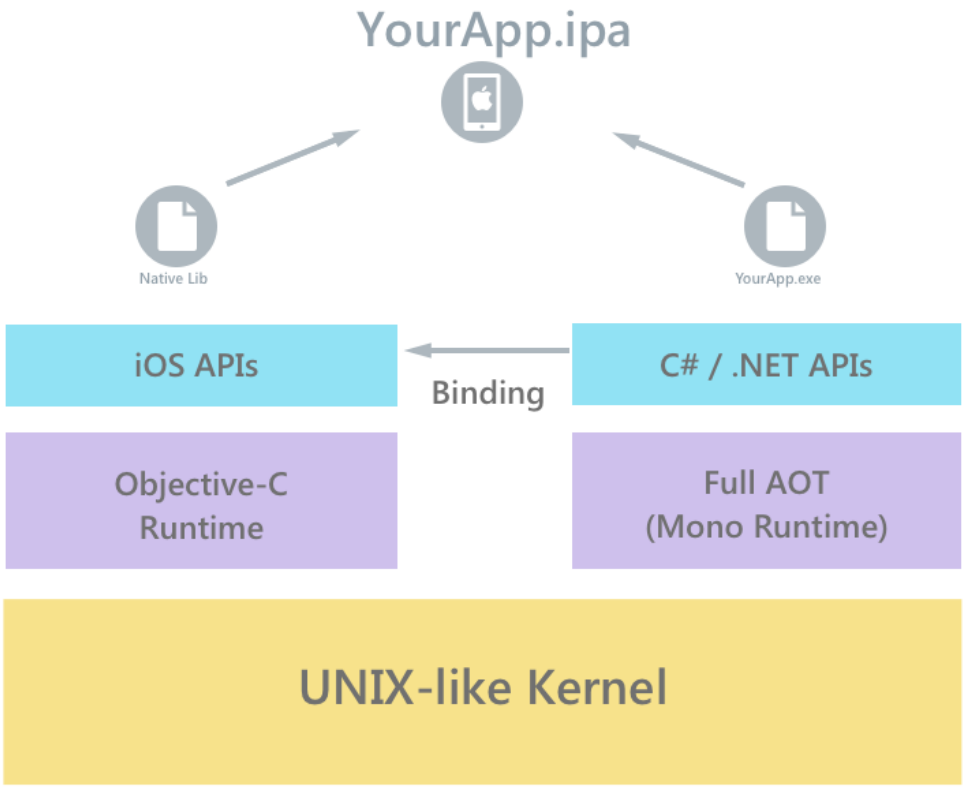
Một ứng dụng được xây dựng trên nền tảng Xamarin tiêu biểu có thể được chia thành sáu tầng:

- Tầng dữ liệu: đảm bảo tính thống nhất và không xung đột của dữ liệu, trong các hệ điều hành di động thì lớp này thường là các cơ sở dữ liệu SQLite
- Tầng truy xuất dữ liệu: cung cấp giao diện cho phép ứng dụng có thể thực hiện các câu lệnh truy vấn vào cơ sở dữ liệu mà không làm lộ ra chi tiết cách các phương thức này hoạt động cụ thể như thế nào

- Tầng nghiệp vụ: là nơi chưa các logic của ứng dụng và các đối tượng trong ứng dụng
- Tầng truy xuất dịch vụ được sử dụng để truy xuất đến các dịch vụ trên đám mây ví dụ như các dịch vụ web như là REST, JSON, WCF, …. Lớp này sẽ đóng gói các hành vi tương tác với các dịch vụ và cung cấp các API đơn giản để cho các tầng ứng dụng và giao diện sử dụng.
- Tầng ứng dụng: mã nguồn ở tầng này thường là những đoạn mã cho từng nền tảng cụ thể hoặc là đoạn mã cho một ứng dụng cụ thể
- Tầng giao diện: cung cấp giao diện cho ứng dụng bao gồm các màn hình, các thành phần giao diện và các trình quản lý phụ trách xử lý chúng

Một ứng dụng không nhất thiết phải chứa tất cả các tầng này ví dụ như tầng truy xuất dịch vụ sẽ không xuất hiện trong các ứng dụng không cần truy cập đến các tài nguyện trên mạng hoặc là các ứng dụng đơn giản có thể gộp hai tầng dữ liệu và tầng truy xuất dữ liệu lại với nhau.

Về cơ bản, Xamarin cố gắng biến các thành phần thuộc các tầng dưới như là tầng truy xuất dịch vụ, tầng nghiệp vụ, tầng truy xuất dữ liệu và tầng dữ liệu thành các thành phần dùng chung giữa các nền tảng trên một ứng dụng hoặc xa hơn là nữa là có thể sử dụng chung giữa nhiều ứng dụng. Ứng dụng Xamarin sẽ chạy trên một môi trường được gọi là Mono. Môi trường này sẽ chạy song song với các môi trường thời gian chạy trên từng nền tảng, cụ thể ở đây là Objective-C Runtime hoặc Android Runtime (ART). Cả hai môi trường chạy này sẽ chạy trên một nhân UNIX hoặc Linux và cung cấp rất nhiều API để cho các lập trình viên có thể sử dụng các thành phần được quản lý hoặc các thành phần native trên các nền tảng. Ở đây các đoạn mã được quản lý là các đoạn mã được chạy dưới sự quản lý của bộ khung Mono (ngôn ngữ trung gian), trong khi đó các đoạn mã native là các đoạn mã được xử lý trực tiếp bởi môi trường native trên từng nền tảng. Mỗi nền tảng có một cơ chế riêng biệt cho phép các lập trình viên có thể truy cập vào các giao diện lập trình native thông qua sự ràng buộc giữa các API trên Xamarin và API native.



Hình 2.4: Phương thức hoạt động của một ứng dụng Xamarin trên iOS[3]

Khi một ứng dụng Xamarin được biên dịch, trình biên dịch Mono C# sẽ chạy và biên dịch các đoạn mã C# hoặc F# thành các đoạn mã trung gian MSIL. Khi ứng dụng chạy trên các nền tảng như Android, iOS hay OSX thì một thành phần được gọi là .NET Common Language Runtime sẽ biên dịch các đoạn mã MSIL sử dụng trình biên dịch JIT. Trình biên dịch JIT có trách nhiệm biên dịch các đoạn mã trung gian thành các đoạn mã native ngay tại thời điểm ứng dụng đang chạy trên từng nền tảng khác nhau

2.3.3. Ưu điểm và nhược điểm

– *Ưu điểm*

Về cơ bản, Xamarin miễn phí cho tất cả mọi người. Xamarin được tích hợp sẵn trong các phiên bản Visual Studio trên các nền tảng Windows và Mac. Xamarin cung cấp ba tuỳ chọn bản quyền tương ứng với ba tuỳ chọn bản quyền cho Visual Studio. Phí được tính vào các dịch vụ đi kèm với các phiên bản Visual Studio.

Đa nền tảng: Xamarin sử dụng C# và XAML cho tất cả các nền tảng hỗ trợ. Trong điều kiện lý tưởng, 96% các đoạn mã có thể sử dụng chung cho cả ba nền tảng nếu sử dụng Xamarin.Forms. Xamarin.Forms cung cấp hơn 40 các thành phần giao diện và điều

[3] *https://developer.xamarin.com/guides/ios/under_the_hood/architecture/*

khiển, được xây dựng sẵn và tự động điều chỉnh phù hợp trên mỗi nền tảng vào thời gian chạy

Xamarin cho phép các nhà phát triển viết các thư viện có thể sử dụng giữa các nền tảng khác nhau hoặc các dự án khác nhau. Những thư viện này được gọi là Portable Class Library. Các nhà phát triển có thể xây dụng một phần logic ứng dụng hoặc các lớp dịch vụ thành các thư viện và sử dụng chúng giữa các nền tảng khác nhau. Xamarin cũng có một số các PCL có sẵn rất thông dụng như SQLite, Json.NET hooặc ReactiveUI trên mọi nền tảng. Bên cạnh lợi ích về có thể tái sử dụng các đoạn mã có sẵn, PCL còn mang lại lợi ích trong việc sử dụng kiểm thử đơn vị và cấu trúc lại mã nguồn, giúp cho việc thay đổi mã nguồn hay tối ưu sẽ trở nên dễ dàng hơn.

Không như các cách tiếp cận truyền thống dựa trên nền tảng web, các ứng dụng được xây dựng dựa trên Xamarin có thể được coi là một ứng dụng native. Hiệu năng của các ứng dụng Xamarin có thể so sánh được với các ứng dụng được viết bằng Java trên Android và Objective-C/Swift trên iOS. Hơn thế nữa, hiệu suất ứng dụng cũng liên tục được cải tiến để đáp ứng được các tiêu chuẩn của các ứng dụng native. Xamarin cũng cung cấp các giải pháp hoàn thiện cho việc kiểm tra và theo dõi hiệu suất ở ứng dụng. Đó là Xamarin Test Cloud kết hợp với Xamarin Test Recorder cho phép các nhà phát triển thực hiện các bài kiểm thử giao diện tự động và xác định các vấn đề về hiệu suất trước khi phát hành. Tuy nhiên các dịch vụ này là các dịch vụ mất phí.

Xamarin cho phép các nhà phát triển tạo ra một trải nghiệm hoàn hảo cho mỗi nền tảng bằng cách sử dụng các thành phần riêng trên mỗi nền tảng. Đối với các ứng dụng sử dụng Xamarin.Forms thì công cụ này sẽ chịu trách nhiệm chuyển đổi các thành phần giao diện thành các thành phần giao diện riêng trên từng nền tảng vào thời điểm chạy. Việc sử dụng Xamarin.Forms sẽ tăng tốc độ phát triển ứng dụng mặc dù việc sử dụng nó cũng sẽ gây ra một số sự suy giảm về hiệu năng bởi vì chúng ta cần một tầng ảo để làm trung gian cho việc chuyển đổi. Đối với các ứng dụng phức tạp cần tuỳ biến nhiều và yêu cầu hiệu năng cao, các nhà phát triển nên sử dụng riêng biệt Xamarin.iOS và Xamarin.Android để phát triển. Sự thân thiện của Xamarin đối với các nền tảng không chỉ dừng lại ở việc hiệu suất của ứng dụng mà còn mở rộng đến việc hỗ trợ các phần cứng khác nhau. Xamarin loại bỏ tất cả các vấn đề tương thích phần cứng khi sử dụng một bộ code cho nhiều nền tảng các nhau bằng cách sử dụng các API riêng biệt và trình cắm. Bên cạnh đó, Xamarin còn cung cấp khả năng sử dụng các thư viện native, tạo sự linh hoạt trong việc tuỳ biến và sử dụng các chức năng native mà không phát sinh thêm nhiều chi phí.

Một điểm mạnh nữa của Xamarin là khả năng của C#, ngôn ngữ chính để phát triển ứng dụng trong Xamarin. C# là một ngôn ngữ hiện đại được phát triển dựa trên C++. C# được cộng đồng phát triển đánh giá là một trong những ngôn ngữ tốt nhất với rất nhiều lợi thế khi so sánh với các ngôn ngữ khác như type safety, Linq, lamda expression, async-await,... Bên cạnh đó C# còn là một ngôn ngữ mà nguồn mở và được sử dụng trong nhiều phần mềm hay nền tảng khác như .Net, Mono, Unity,..

Xamarin được phát hành kèm với các phiên bản Visual Studio, một trong những IDE tốt nhất hiện nay. Một IDE tốt hỗ trợ rất nhiều cho các nhà phát triển trong quá trình phát triển. Bên cạnh đó Xamarin còn có một số các dịch vụ hữu ích để hỗ trợ các nhà phát triển như Profiler hoặc Xamarin Test Cloud dù không miễn phí.

– *Nhược điểm*

Có một sự thật là hiện nay với các bộ khung phát triển phần mềm đa nền tảng thì việc một mã nguồn chạy trên nhiều nền tảng chưa bao giờ đạt đến sự hoàn hảo. Hầu hết các ứng dụng để kinh doanh đều yêu cầu phải tuỳ biến cho từng nền tảng và các nhà phát triển thường xuyên phải viết các đoạn mã riêng biệt cho từng nền tảng để đảm bảo trải nghiệm người dùng hoàn hảo trên mỗi nền tảng. Xamarin.Forms cũng tương tự, nó không thể hoàn toàn đáp ứng được yêu cầu của ứng dụng. Bên cạnh đó có rất nhiều thành phần trong các nền tảng mà Xamarin.Forms không hỗ trợ.

Bản thân Xamarin hoạt động trên hai nền tảng iOS và Android tồn tại những hạn chế riêng về mặt kĩ thuật. Các hạn chế này thường đến từ sự khác biệt về ngôn ngữ phát triển và cách hoạt động giữa Xamarin và các nền tảng, ở đây thường là cách các trình biên dịch hoạt động. Điều này đồng nghĩa với việc một số tính năng có trong C# thì sẽ không hoạt động trên iOS hoặc Android. Chi tiết về các hạn chế kĩ thuật này được Xamarin cung cấp đầy đủ trên trang chủ [18] [19].

Tồn tại một số vấn đề nữa của Xamarin đến từ chính bản thân nó [17]. Xamarin cố ánh xạ tất cả các API native thành API của Xamarin và vấn đề ở đây là điều này không phải lúc nào cũng hoạt động tốt. Xamarin tạo ra một layer để tương tác với môi trường native trên các nền tảng. Tuy nhiên điều này kết hợp với Xamarin AOT compiler gây ra một vấn đề là nhà phát triển không thực sự kiểm soát được cái được tạo ra như là đoạn mã cuối cùng để chạy trên thiết bị.

Bên cạnh đó, một trong những vấn đề của Xamarin là lỗi của chính bộ khung phát triển. Ví dụ nhiều nhà phát triển phàn nàn về việc rò rỉ bộ nhớ xảy ra rất thường xuyên trên các ứng dụng iOS được phát triển bằng Xamarin [20]. Lý do cho điều này được

phỏng đoán là do sự không đồng bộ giữa hai cơ chế giải phóng bộ nhớ của Xamarin là *Garbage Colection* và cơ chế *Automatic Reference Counting* trên iOS. Khá may mắn là đây là một vấn đề khá nghiêm trọng và thu hút được một số lượng lớn các nhà phát triển quan tâm. Và vấn đề này đã được phải quyết bằng cách viết một phương thức bổ sung để duyệt đệ quy qua toàn bộ phần tử và các đối tượng được liên kết để giải phóng chúng lần lượt. Tuy nhiên không phải vấn đề nào cũng dành được sự quan tâm to lớn như thế này. Có khá nhiều các nhà phát triển đưa lên các vấn đề trong việc phát triển ứng dụng của họ mà lỗi được xác định không phải do đoạn mã của họ mà lỗi xuất phát từ chính bản thân Xamarin.

Các ứng dụng phát triển bởi Xamarin thường bị phình to ra hơn so với các ứng dụng native [21]. Để so sánh thì các ứng dụng Xamarin thường chiếm nhiều hơn một vài MB so với các ứng dụng Objective-C/Java thông thường. Và nếu càng dùng nhiều API thì càng tốn nhiều dung lượng hơn. Điều này có thể gây khó khăn cho người dùng cuối khi cài đặt ứng dụng và yêu cầu nhiều bộ nhớ hơn trên thiết bị.

CHƯƠNG 3: SO SÁNH VÀ ĐÁNH GIÁ

Các nhà phát triển khi xây dựng ứng dụng đều mong muốn lựa chọn bộ khung phát triển phù hợp nhất, hỗ trợ tốt nhất với những tính năng được yêu cầu. Các bộ khung phát triển đa nền tảng thường chọn cách tiếp cận trung hoà giữa các nền tảng hỗ trợ, ưu tiên phát triển các tính năng chung trên nhiều nền tảng. Khi đó, có thể có những tính năng đặc biệt ở một nền tảng riêng biệt sẽ chưa được hỗ trợ ngay. Tùy vào yêu cầu thực tế, nhà phát triển có thể lựa chọn bộ khung phát triển phù hợp. Ở chương này sẽ thực hiện việc so sánh các tính năng được hỗ trợ, khả năng khi phát triển của hai bộ khung phát triển là Ionic và Xamarin để các nhà phát triển có thể cân nhắc khi xây dựng ứng dụng.

3.1 So sánh

Dưới đây là bảng so sánh một số tính năng chính trên nền tảng iOS với khả năng đáp ứng tương ứng của hai bộ khung phát triển Ionic và Xamarin.

Bảng 3.1: Bảng so sánh các tính năng hỗ trợ của Ionic và Xamarin trên nền tảng iOS

iOS	Ionic	Xamarin
App extensions	Không hỗ trợ	Có hỗ trợ nhưng tồn lại một số giới hạn như không thể truy cập vào camera hay microphones của thiết bị, không thể nhận dữ liệu AirDrop
Handoff	Không hỗ trợ	Có hỗ trợ
Document Picker	Không hỗ trợ	Có hỗ trợ
AirDrop	Không hỗ trợ	Có hỗ trợ
Hiển thị, xử lý văn bản Hỗ trợ nhiều loại bàn phím khác nhau cho nhiều mục đích	Sử dụng các thành phần html để hiển thị các đoạn văn bản, sử dụng Javascript để xử lý các đoạn văn bản Hỗ trợ nhiều kiểu bàn phím và các sự kiện liên quan với các plug in hỗ trợ	Sử dụng các thành phần có sẵn như Label, Entry, Editor để hiển thị văn bản Có khả năng tuỳ biến các văn bản theo mong muốn Hỗ trợ nhiều kiểu bàn phím khác nhau và các sự kiện liên quan

Cut/Copy/Paste	Có plug in hỗ trợ các tác vụ sao chép/ cắt/ dán nhưng chỉ hỗ trợ văn bản	Bắt buộc phải viết các Dependency service cho từng nền tảng cụ thể
Hỗ trợ các giao diện nhập khác	Sử dụng trình cắm để hiển thị các giao diện phụ trên bàn phím	Hỗ trợ các giao diện nhập khác
Đa tác vụ Tác vụ chạy ngầm định	Hỗ trợ việc chạy các tác vụ khi ứng dụng ở chế độ nghỉ sử dụng trình cắm Sử dụng Webworker để thực hiện các tác vụ trên các luồng khác nhau mặc dù có một số hạn chế	Hỗ trợ chạy các tác vụ khi ứng dụng ở chế độ nghỉ Sử dụng các API .Net hoặc các API native để tạo, quản lý các luồng tác vụ khác nhau
Autolayout	Thiết kế đáp ứng	Sử dụng một hệ thống các layout đẻ thiết kế giao diện phù hợp với nhiều thiết bị
Dịch vụ nhận thông báo	Hỗ trợ cả thông báo cục bộ và thông báo từ xa	Hỗ trợ cả thông báo cục bộ và thông báo từ xa
Nhận diện cử chỉ	Hỗ trợ rất nhiều cử chỉ có sẵn	Hỗ trợ rất nhiều cử chỉ có sẵn
Tương tác với danh bạ	Không cung cấp một thành phần có sẵn để tương tác với danh bạ trên thiết bị. Các nhà phát triển phải tự xây dựng giao diện phù hợp nhất cho tất cả các nền tảng	Xamarin hỗ trợ tương tự iOS
Tương tác với lịch	Tương tự như với danh bạ, lập trình viên cũng phải tự xây dựng giao diện cho ứng dụng của mình	Xamarin hỗ trợ tương tự iOS
VolP	Ionic tồn tại một số nhược điểm trong việc xây dựng một ứng dụng VolP do giới hạn của trình duyệt	Xamarin hỗ trợ tương tự iOS

Các dịch vụ ngang hàng	Ionic có các trình cắm hỗ trợ các kết nối ngang hàng sử dụng các giao diện WebRTC hoặc bluetooth	Xamarin hỗ trợ tương tự iOS
Đồ hoạ	Bắt buộc sử dụng GPU để đảm nhiệm các tác vụ liên quan đến đồ hoạ Có thể phải tích hợp thêm một trình cắm đặc biệt để hỗ trợ việc xử lý đồ hoạ trên thết bị Ionic không hỗ trợ các API bậc thấp hỗ trợ việc dựng các thành phần như văn bản hay ảnh nên không phù hợp với các ứng dụng cần xử lý nhiều ảnh và văn bản cùng một lúc tương tự như Facebook	Xamarin hỗ trợ tương tự iOS

Bảng so sánh trên thể hiện được một phần về đặc điểm của hai bộ khung phát triển. Ionic phụ thuộc nhiều vào các plugin, trong khi đó Xamarin hỗ trợ gần như đầy đủ các tính năng trên iOS nhờ khả năng tương tác tốt với các thư viện native.

3.2 Đánh giá

Trong phần này, chúng ta sẽ đánh giá hai bộ khung phát triển Ionic và Xamarin dựa trên một số tiêu chí cụ thể. Bộ tiêu chí ban đầu được đề xuất dựa trên thảo luận giữa các nhà phát triển ứng dụng di động. Các tiêu chí này cũng được phát triển thêm dựa vào một số bài báo khoa học [22][23][24].

Các nhà phát triển ứng dụng đều mong muốn có thể đưa được sản phẩm của mình đến càng nhiều người dùng càng tốt. Đặc biệt là đối với hệ sinh thái di động, nơi có sự đa dạng rõ rệt giữa các thiết bị, các hệ điều hành, các nhà phân phối, thì khi lựa chọn bộ khung phát triển ứng dụng da nền tảng họ sẽ quan tâm nhất đến khả năng đáp ứng của bộ khung phát triển đấy đối với nhiều thiết bị và nhiều hệ diều hành khác nhau. Điều này dẫn đến điều họ quan tâm là khả năng tùy biến giao diện, khả năng phản hồi với các thiết bị có

kích cỡ màn hình khác nhau, đây là hai tiêu chí được chú trọng đầu tiên vì chúng liên quan đến giao diện, tương tác trực tiếp và mang tới trải nghiệm cho người dùng. Ngoài ra những tiêu chí về việc hỗ trợ bởi bên thứ ba, dịch vụ web, hoạt ảnh, đa luồng… cũng là những yếu tố cần được cân nhắc.

- Giao diện người dùng và trải nghiệm người dùng (User Interface và User eXperimence)

Cả Xamarin và Ionic đều cung cấp cho người dùng một số các thành phần giao diện được thiết kế sẵn với khả năng tự động thay đổi theo từng nền tảng nhằm mang lại trải nghiệm tốt nhất cho người dùng trên mỗi nền tảng. Số lượng các thành phần này trên cả Ionic và Xamarin thường không đủ đáp ứng yêu cầu của các nhà phát triển nếu như ứng dụng của họ yêu cầu nhiều chức năng đặc biệt trên từng nền tảng hoặc sử dụng nhiều giao diện tuỳ biến. Nếu chỉ xét về số lượng thì Xamarin chiếm ưu thế so với Ionic. Xamarin cung cấp một phương pháp cho phép các nhà phát triển có thể dễ dàng tuỳ chỉnh giao diện và hành vi của các thành phần mặc định của Xamarin Forms [16]. So sánh với Ionic thì việc tuỳ biến giao diện và hành vi của các thành phần có sẵn trên Xamarin dễ dàng và được hỗ trợ tốt hơn, phù hợp với nhiều nhà phát triển. Về cơ bản các API trên Xamarin được ánh xạ từ các API native trên các nền tảng nên về lý thuyết, các nhà phát triển có thể tham khảo các tài liệu về nền tảng hoặc các thư viện để thực hiện trên Xamarin. Tất nhiên, điều này không phải luôn đúng do sự khác biệt giữa cách triển khai, cách hoạt động của các trình biên dịch trên Xamarin và các nền tảng. Trong khi đó việc tuỳ biến trên Ionic không dễ dàng do các nhà phát triển phải tương tác với WebKit để thực hiện các tác vụ mong muốn. Điều này có thể rất khác so với việc phát triển ứng dụng thông thường. Vì vậy thường các nhà phát triển bình thường sẽ phụ thuộc vào sự hỗ trợ của cộng đồng các nhà phát triển chuyên nghiệp có kiến thức nền tảng tốt.

- Thiết kế giao diện (Layout)

Một vấn đề quan trọng trong việc phát triển các ứng dụng di động là việc phải thiết kế được một layout giao diện có thể chạy tốt trên nhiều thiết bị, nhiều kích cỡ màn hình. Điều này vốn đã khó khăn trên mỗi nền tảng (ví dụ như nền tảng Android có sự phân mảnh rất rõ rệt trên các thiết bị vì có quá nhiều thiết bị với đủ loại kích thước và phần cứng cũng như phần mềm khác nhau), thì đối với phát triển ứng dụng đa nền tảng thì còn khó hơn do còn phải hỗ trợ nhiều nền tảng khác nhau. Bởi vì Ionic được xây dựng dựa trên công nghệ web nên việc thiết kế layout trên các thiết bị di động sử dụng thiết kế đáp ứng (responsive design) còn Xamarin sử dụng các layout có sẵn như StackLayout,

Relaytive Layout, AbsoluteLayout, v.v… khá tương tự như Android để thiết kế giao diện cho ứng dụng. Hiện nay tất cả các cách tiếp cận để giao diện của các ứng dụng có thể đáp ứng tất cả các thiết bị khác nhau đều không hoàn toàn hoàn hảo. Các lập trình viên có thể cần sử dụng các thành phần giao diện một cách linh động, hoặc là có thể phải viết các đoạn mã để tuỳ chỉnh riêng cho các thiết bị mà một layout chung không đáp ứng được. Một điểm tốt là các thành phần giao diện trên Ionic hỗ trợ và tối ưu responsive toàn diện nên các khó khăn trên gần như không ảnh hưởng đến hiệu năng của ứng dụng. Cả Ionic và Xamarn đều cho phép các lập trình viên có thể tạo các giao diện trong thời gian chạy. Đối với các lập trình viên yêu thích việc thiết kế giao diện bằng cách kéo thả thì Ionic có phần tốt hơn so với Xamarin. IDE mặc định của Xamarin không hỗ trợ việc thiết kế giao diện bằng cách kéo thả, các nhà phát triển bắt buộc phải viết các đoạn mã giao diện bằng XAML, trong khi đó Ionic có hỗ trợ kéo thả giao diện thông qua Ionic Creator. Mặc dù nó có khá nhiều giới hạn khi sử dụng phiên bản miễn phí ví dụ như không thể xuất giao diện thành các đoạn mã HTML, không thể thêm các đoạn mã định hướng navigation,…. Xamarin hỗ trợ *live preview* cho phép các nhà phát triển thấy được thay đổi của giao diện khi họ cập nhật các đoạn mã, trong khi đó Ionic hỗ trợ *live renderer* cho phép thể hiện tất cả các thay đổi của ứng dụng trên trình duyệt ngay sau khi các nhà phát triển lưu lại tệp mã nguồn.

– Hoạt ảnh (Animation)

Đối với ứng dụng di động, ngoài giao diện tĩnh thì các animation cũng đóng vai trò rất quan trọng trong việc nâng cao trải nghiệm người dùng. Ionic chủ yếu sử dụng Ionic animation để thực hiện các chuyển động của các đối tượng. Tuy nhiên, khác với việc sử dụng các animation trên nền tảng web thông thường sử dụng CSS3 animation, trên Ionic thì các nhà phát triển cần sử dụng một thành phần CSS đặc biệt khác để đưa việc xử lý animation cho GPU bởi vì CPU trên di động không đảm bảo đủ hiệu suất cho các hoạt ảnh sử dụng CSS. Đây được gọi là tăng tốc phần cứng hoạt ảnh (hardware accelerated animation). Tuy nhiên việc phụ thuộc quá nhiều vào GPU có thể là một điểm yếu vì có thể dẫn đến tiêu hao nhiều pin. Tương tự như Ionic, Xamarin cũng cung cấp nhiều các hành động hoạt hoạ có sẵn và cho phép tuỳ biến thời gian hay kết hợp các hành động với nhau thành một chuỗi hành động.

– Các dịch vụ và thư viện từ bên thứ ba

Hiện nay, có rất nhiều ứng dụng di động để hoạt động tốt phụ thuộc rất nhiều vào các dịch vụ web. Vì vậy việc tích hợp các dịch vụ web vào các ứng dụng di động trở nên

rất quen thuộc đối với các nhà phát triển. Nền tảng Xamarin hỗ trợ nhiều công nghệ dịch vụ web khác nhau như RESTful, ASMX và WCF dựa trên các thư viện được tích hợp sẵn và thư viện từ bên thứ ba. Ionic tất nhiên với nền tảng là công nghệ web nên cũng hỗ trợ rất tốt các dịch vụ web dựa trên AngularJS. Bên cạnh các dịch vụ web thì các ứng dụng di động còn cần tích hợp các tính năng hoặc dịch vụ khác từ bên thứ ba như push notification, crash reporting hay analytics. Cả Xamarin và Ionic đều cung cấp dịch vụ cho phép gửi các thông báo đến tất cả nền tảng theo một định dạng duy nhất. Điều này giúp giảm công việc cho phần backend không phải tách ra hỗ trợ cho từng nền tảng riêng biệt. Ngoài một số phần cấu hình cần phải làm riêng biệt trên từng nền tảng thì Ionic xây dựng phần triển khai trên các nền tảng thành một plug-in. Các nhà phát triển chỉ cần viết một đoạn mã để xử lý thông báo được gửi từ server. Điều này làm cho Ionic có vẻ có tính đa nền tảng hơn Xamarin khi mà các nhà phát triển phải sử dụng những đoạn mã nguồn khác nhau cho mỗi nền tảng. Cả Ionic và Xamarin đều hỗ trợ nếu nhà phát triển muốn sử dụng một dịch vụ khác phổ biến của bên thứ ba để gửi thông báo như Firebase, mặc dù việc triển khai có thể rắc rối hơn một chút đối với Ionic, còn Xamarin thì vẫn làm tương tự như với dịch vụ Xamarin cung cấp. Đối với dịch vụ như phân tích hay báo cáo lỗi (crash reporting) thì mọi việc trở nên khó khăn hơn với Ionic khi chưa có plugin nào cho Ionic hỗ trợ tốt cho Firebase, các nhà phát triển có thể phải lựa chọn một dịch vụ khác như Fabric. Điều này thể hiện một điểm yếu rõ ràng của Ionic khi bị phụ thuộc nhiều vào cộng đồng phát triển các plugin khi chưa có đủ khả năng để có thể xử lý vấn đề tích hợp các tính năng mới vào nền tảng. Xamarin nhờ cộng đồng lớn hơn và khả năng tương thích tốt với các thư viện native nên dễ dàng hỗ trợ các tính năng mới cho ứng dụng hơn. Firebase và Fabric đều có các gói có sẵn trên thư viện của Xamarin. Bên cạnh các dịch vụ nói trên thì các nhà phát triển còn quan tâm đến cách tích hợp quảng cáo để mang lại doanh thu cho ứng dụng. Hiện nay có rất nhiều nhà cung cấp quảng cáo trên các ứng dụng di động, các nhà phát triển cũng thường tích hợp rất nhiều các quảng cáo từ các nhà cung cấp khác nhau trên ứng dụng của mình để đảm bảo doanh thu. Danh sách các nhà cung cấp quảng cáo phổ biến được Ionic hỗ trợ dưới dạng plugin có thể kể đến Admob, Facebook Audience Network, Mopub, Appodeal. Cũng tương tự như việc tích hợp các SDK phân tích và báo cáo lỗi, việc tích hợp các SDK quảng cáo phụ thuộc vào các nhà phát triển chuyên nghiệp có kĩ năng cao phát triển các trình cắm cho cộng đồng. Cộng đồng phát triển càng mạnh thì sẽ càng thu hút được nhiều nhà phát triển chuyên nghiệp để hỗ trợ cho cộng đồng. Khả năng tương thích tốt với native tiếp tục giúp cho Xamarin ghi điểm trong việc hỗ trợ các SDK quảng cáo. Các nhà phát triển hoàn toàn có thể thực hiện theo các

hướng dẫn trên trang chủ của các nhà cung cấp quảng cáo để thực hiện trong lúc Xamarin chưa hỗ trợ cho SDK đó.

– Đa luồng

Trên các thiết bị di động, các tài nguyên đều tương đối hạn chế bao gồm cả khả năng xử lý của CPU, GPU hay dung lượng bộ nhớ, RAM, v.v… Ví dụ đối với iOS, để đảm bảo ứng dụng hoạt động mượt mà thì ứng dụng phải đảm bảo được số khung hình trên một giây là 60 khung hình. Tương đương với việc mỗi khung hình các nhà phát triển sẽ có khoảng 16.67ms để xử lý dữ liệu để hiển thị trên khung hình tiếp theo. Điều này đồng nghĩ với việc đối với các ứng dụng phức tạp thì nhà phát triển không thể chỉ sử dụng một luồng duy nhất để thực thi việc xử lý. Trên thực tế các ứng dụng lớn đều đưa phần xử lý dữ liệu chính sang một luồng khác, luồng chính chỉ được sử dụng để thực hiện các thay đổi trên giao diện, vốn được yêu cầu bắt buộc thực hiện trên luồng chính. Do vậy việc hỗ trợ đa luồng trên các bộ khung phát triển là môt điểm quan trọng cần cân nhắc đến khi các nhà phát triển muốn phát triển một ứng dụng trên các bộ khung đấy đặc biệt là các ứng dụng enterprise. Xamarin.iOS cho phép các nhà phát triển sử dụng các API luồng của .Net bao gồm cả việc sử dụng trực tiếp các lớp System.Threading.Thread hay System.Threading.ThreadPool hoặc sử dụng gián tiếp thông qua các mẫu bất đồng bộ (Asynchronous Programming Pattern) hoặc phương thức BeginXXX cũng như các API hỗ trợ Task Parallel Library. Trong đó TPL được đề nghị bởi Xamarin để phát triển các ứng dụng có sử dụng nhiều luồng nhờ một số lợi ích. Trình lập lịch TPL mặc định sẽ uỷ nhiệm việc thực thi các tác vụ cho thread pool [14], có nhiệm vụ điều phối số lượng thread cần cho các tiến trình thực hiên. Điều này giúp cho ứng dụng tránh cho việc quá nhiều luồng sẽ cạnh tranh thời gian thực thi của CPU. Thread pool sẽ chịu trách nhiệm tăng số lượng từ từ dựa vào số lượng nhân CPU có thể sử dụng trong hệ thống, tải của hệ thống và nhu cầu của ứng dụng. Đối với Ionic thì mọi chuyện có vẻ trở nên phức tạp hơn, vì Javascript không được thiết kế để hỗ trợ đa luồng. Các hàm như setTimeout, setInterval,.. không thực sự tạo ra các thread mới. Chúng chỉ đơn giản gọi các đoạn mã ấy muộn hơn ở một luồng duy nhất. Tương tự như các nhà phát triển web thì Ionic có thể dùng các thư viện như WebWorker để có thể đưa các tác vụ nặng xử lý ở các luồng chạy nền mà không ảnh hưởng đến trải nghiệm người dùng. Tuy nhiên WebWorker cũng có một số hạn chế ví dụ như không thể truy cập đến các thành phần DOM từ trang web (vì nó không thread safe), không thể truy xuất các biến toàn cục và các hàm Javascript từ trang web, không thể gọi hàm alert hay confirm, các đối tượng như window, document và parent không thể truy cập trong web worker. Qua đây, có thể thấy được, việc sử dụng các

ngôn ngữ khác nhau với mục đích ban đầu khác nhau tạo ra sự khác biệt trong việc xử lý đa luồng của Xamarin và Ionic. Javascript ban đầu được thiết kế để chạy trên các ứng dụng web. Điều đó có nghĩa là Javascript luôn chạy trên một luồng duy nhất trên một trang web [11] và các trình duyệt hiện đại đều thực thi một luồng đơn trên mỗi trang trong khi Javascript runtime là đa luồng với một luồng khác nhau trên mỗi trang. Đa luồng không thực sự cần thiết với Javascript so với các ngôn ngữ khác. Vì vậy khi sử dụng Javascript để phát triển một ứng dụng di động, nơi việc xử lý đa luồng khá là phổ biến thì Ionic gặp một số khó khăn. Xamarin sử dụng C#, một ngôn ngữ được sử dụng rộng rãi trên nhiều nền tảng như desktop, mobile, web. C# được thiết kế có thể xử lý tốt các vấn đề liên quan đến chia sẻ tài nguyên giữa các luồng. Vi vậy Xamarin có sẵn các thiết kế, thư viện để xử lý việc đa luồng trong ứng dụng. Kết luận lại đối với việc xử lý đa luồng, Xamarin có lợi thế hơn Ionic nhờ bản chất, thiết kế, mục tiêu ban đầu của ngôn ngữ phát triển.

– Ứng dụng đặc biệt

Bên cạnh các ứng dụng thông thường thì các ứng dụng đặc biệt như là các ứng dụng trò chơi cũng là một thể loại được các nhà phát triển quan tâm. Các khái niệm trong việc phát triển ứng dụng trò chơi có một sự khác biệt rõ ràng so với việc phát triển các ứng dụng thông thường. Các nhà phát triển khi mà đi từ việc phát triển các ứng dụng thông thường sang các ứng dụng trò chơi thì chắc chắn phải đối mặt với các khái niệm mới và các cách phát triển khác. Vì những lý do đó thì thường để phát triển các ứng dụng trò chơi thì các nhà phát triển sẽ sử dụng các engine chuyên biệt để phát triển game như Unity, Unreal Engine. Tuy nhiên việc sử dụng các framework phát triển ứng dụng thông thường để phát triển game không phải là điều không thể. Cả Ionic và Xamarin đều có thể được sử dụng để xây dựng các ứng dụng game, tuy mức độ hỗ trợ là khác nhau. Ionic là một bộ khung phát triển ứng dụng thông thường nên Ionic không phù hợp với việc phát triển các ứng dụng trò chơi yêu cầu nhiều đồ hoạ. Tuy nhiên các nhà phát triển hoàn toàn có thể kết hợp với các engine làm game khác như Phaser.io để tạo ra các game có sự kết hợp giữa đồ hoạ và các thành phần giao diện như một ứng dụng thông thường như menu. Xamarin cũng giới thiệu khác nhiều các công nghệ như CocosSharp, MonoGame, UrhoSharp, v.v…. Các công nghệ này có thể sử dụng với Xamarin.iOS và Xamarin.Android đi kèm với các ví dụ có sẵn để các nhà phát triển có thể dễ dàng làm quen và sử dụng.

– Cộng đồng phát triển

Đối với các nhà phát triển thì cộng đồng luôn là một sự trợ giúp đắc lực. Một cộng đồng mạnh sẽ đảm bảo cho sự thành công của một nền tảng. Để so sánh thì với thẻ đánh dấu ‘ionic' trên stackoverflow cho ra khoảng 20000 kết quả, còn với thẻ đánh dấu ‘xamarin' thì có khoảng hơn 40000 kết quả. Điều này cũng phản ánh một phần nào sự phổ biến của hai nền tảng. Song song với việc cộng đồng lớn hơn thì số lượng các thư viện mã nguồn mở được viết bởi cộng đồng người dùng của Xamarin cũng lớn hơn Ionic tương đối. Một số các thư viện của bên thứ ba nổi tiếng ngoài phiên bản trên các nền tảng native thì cũng hỗ trợ nền tảng Xamarin như Realm, Charts, … Bên cạnh đó, khả năng cho phép sử dụng các thư viện native của Xamarin cũng được đánh giá cao hơn Ionic. Xamarin cung cấp tài liệu chi tiết hướng dẫn việc sử dụng các thư viện native trên Android và iOS [15] trong ứng dụng. Đối với Ionic công việc dường như khó khăn hơn rất nhiều vì các nhà phát triển gần như phải viết lại thư viện dưới dạng một plugin và việc này đòi hỏi nhà phát triển phải có kinh nghiệm trong việc sử dụng cả javascript và các ngôn ngữ native.

– Khả năng kiểm thử:

Người dùng di động hoàn toàn có thể là những khách hàng khó tính nhất hiện nay. Họ mong muốn có một sản phẩm phản hồi tốt, không có lỗi và giá rẻ. Các ứng dụng không đạt được những điều này rất có thể sẽ bị gỡ bỏ và đi kèm theo đó là bị đánh giá thấp trên chợ ứng dụng. Cách hiệu quả nhất để kiểm tra tính đúng đắn của ứng dụng là chạy nó và sử dụng nó. Nếu như ứng dụng hoạt động đúng như mong muốn của nhà phát triển mà không bị dừng ứng dụng hoặc trả về kết quả không mong muốn thì ứng dụng đấy có được một trạng thái rất tuyệt vời và nhà phát triển có thể hoàn toàn yên tâm đưa nó đến với người dùng. Một ứng dụng di động cần được kiểm thử hai thành phần quan trọng nhất là logic và UI/UX của ứng dụng. Hiện tại, để đảm bảo chất lượng của sản phẩm không phải chỉ cần đội ngũ QA mà các lập trình viên cũng tham gia một phần nào đó trong việc đảm bảo chất lượng sản phẩm. Các lập trình viên ngày nay trở nên độc lập hơn, họ tham gia nhiều hơn vào việc kiểm soát chất lượng sản phẩm. Để kiểm thử logic của ứng dụng thì Unit Test và Test Driven Development cũng trở nên quen thuộc với các lập trình viên. Ngoài việc thiết kế kiến trúc ứng dụng cho phép dễ dàng thực hiện kiểm thử đơn vị, nhà phát triển còn cần có một thư viện kiểm thử đơn vị tốt hỗ trợ họ. Cách đơn giản nhất để các nhà phát triển thực hiện kiểm thử đơn vị là viết các bộ các ca kiểm thử tương ứng cho từng nền tảng: NUnit cho phiên bản desktop của ứng dụng, MonoTouch.NUnitLite có thể được dùng cho Xamarin.iOS và Andr.Unit cho Xamarin.Android. Cách đơn giản nhất này cần đến nhiều công sức khi các lập trình viên phải viết lập đi lập lại nhiều test case cho từng nền tảng khác nhau. Xamarin vẫn chưa có một giải pháp nào có thể giải quyết vấn đề

này bởi vì sự khác biệt giữa các nền tảng. Và các nhà phát triển vẫn đang chấp nhận việc chạy một bộ test case trên một nền tảng và hi vọng rằng nó sẽ chạy tốt trên các nền tảng khác hoặc sử dụng một số cách tương đối "linh hoạt" để có thể tiết kiệm thời gian và công sức cho việc kiểm thử của mình. Trên Ionic, logic của ứng dụng được viết bằng ngôn ngữ Javascript dựa trên bộ khung AngularJS. Vì vậy để thực hiện kiểm thử đơn vị trên Ionic, các lập trình viên thường sử dụng luôn các thư viện kiểm thử đơn vị trên AngularJS ví dụ như Jasmine và Karma. Jasmine và Karma là hai bộ khung kiểm thử được khuyến nghị bởi Angular, đã được chứng minh về chức năng và sự tiện lợi. Kiểm thử đơn vị trên Ionic thì các lập trình viên không phải suy nghĩ nhiều về việc phải thực hiện trên các nền tảng khác nhau. Các logic sẽ được xử lý giống nhau trên tất cả các nền tảng bởi vì các ứng dụng Ionic đều được chạy trên nền tảng web, nó ít phụ thuộc vào nền tảng hệ điều hành của ứng dụng. Như vậy, đối với việc kiểm thử đơn vị thì Ionic có lợi thế hơn Xamarin nhờ sự thống nhất hơn về môi trường chạy của ứng dụng.

Bên cạnh việc kiểm thử các logic trong ứng dụng bằng cách sử dụng kiểm thử đơn vị thì kiểm thử chấp nhận giao diện (UI Acceptance Testing) cũng rất quan trọng đối với các lập trình viên để đảm bảo chất lượng của ứng dụng, Tuy nhiên, quá trình này thường rất tiêu tốn nhiều tài nguyên bởi vì bản chất thủ công của nó. Trước đây, quá trình kiểm thử giao diện cần rất nhiều sức người để triển khai ứng dụng và sử dụng nó, kiểm tra xem ứng dụng có hoạt động đúng không, sự bố trí trên giao diện có bị xô lệch trên các thiết bị khác nhau không. Sự phức tạp càng tăng lên khi việc kiểm thử cần được thực hiện trên mỗi bản build trên mỗi nền tảng, cũng như càng nhiều thiết bị càng tốt mà ứng dụng đấy có thể hoạt động. Điều này trở nên quá sức với kể cả những công ty lớn chứ chưa tính đến các nhà phát triển độc lập. Để thực hiện kiểm thử giao diện trên Xamarin thì các nhà phát triển sẽ sử dụng một thư viện trong bộ công cụ Xamarin Test Cloud có tên là Xamarin.UITest để thực hiện. Xamarin.UITest là một bộ khung kiểm thử cho phép thực hiện quá trính kiểm thử chấp nhận giao diện tự động được viết bằng NUit để chạy trên các thiết bị iOS và Android. Các ca kiểm thử có thể tương tác với giao diện người dùng như môt người dùng như là nhập văn bản, tương tác với các thành phần điều khiển, thực hiện các cử chỉ như là vuốt, kéo thả,... Thông thường, mỗi trường hợp kiểm thử được viết như là một phương thức. Mỗi lớp chứa các trường hợp kiểm thử được gọi là 'test fixture'. Mỗi 'test fixture' có thể chứa một trường hợp kiểm thử hoặc chứa một bộ các trường hợp kiểm thử và chịu trách nhiệm trong việc khởi tạo, thiết lập để các phương thức kiểm thử chạy và giải phóng các tài nguyên khi phiên kiểm thử kết thúc. Xamarin.UITest cho phép các lập trình viết các test case và chạy trên thiết bị hoặc có thể tải lên dịch vụ Test Cloud để

thực hiện quá trình kiểm thử trên nhiều thiết bị hơn. Để thực hiện kiểm thử giao diện trên các bộ khung phát triển ứng dụng sử dụng nền tảng web như Xamarin, thông thường, các nhà phát triển có thể thực hiện kiểm thử trên trình duyệt thông thường như Chrome, mô phỏng giao diện ứng dụng di động hoặc kiểm thử trực tiếp trên thiết bị. Để kiểm thử trên trình duyệt thông thường, chúng ta có thể sử dụng bộ khung kiểm thử Jasmine như kiểm thử đơn vị kết hợp với Protractor [12], một bộ khung kiểm thử giao diện cho phép tương tác với các trình duyệt như người dùng, được thiết kế cho các ứng dụng sử dụng Angular. Tuyệt vời hơn nữa, để kiểm thử trực tiếp trên ứng dụng, thì các nhà phát triển có thể sử dụng Appium [13]. Appium là bộ công cụ kiểm thử tự động có thể sử dụng với các ứng dụng native, hybrid hay web. Appium có một đặc điểm khá thú vị là cho phép các nhà phát triển có thể viết test case bằng bất cứ ngôn ngữ nào tuỳ ý, nó chỉ phục thuộc vào các giao diện lập trình ứng dụng HTTP để tương tác với các thành phần kiểm thử tự động trên mỗi nền tảng.

Tổng kết lại, các bộ khung kiểm thử trên Xamarin và Ionic đều hỗ trợ tốt hai nền tảng chính là Android và iOS. Quá trình kiểm thử không phức tạp và có thể thực hiện trên cả thiết bị và máy ảo. Khi so sánh việc kiểm thử tự động trên Xamarin và Ionic thì Xamarin có những lợi thế nhất định. Quá trình kiểm thử tự động trên Ionic có sự thống nhất cao hơn vì đều chạy trực tiếp trên nền tảng web, trong khi đó Xamarin bởi vì trình biên dịch của Xamarin chịu trách nhiệm phiên dịch mã nguồn trên Xamarin sang các đoạn mã native trên từng nền tảng. Vì vây việc kiểm thử trên các nền tảng trở nên phân tán, để đảm bảo được chất lượng ứng dụng thì các nhà phát triển cần phải chạy các ca kiểm thử trên từng nền tảng. Việc đó cần nhiều công sức hơn đối với nhà phát triển.

CHƯƠNG 4: ỨNG DỤNG THỬ NGHIỆM

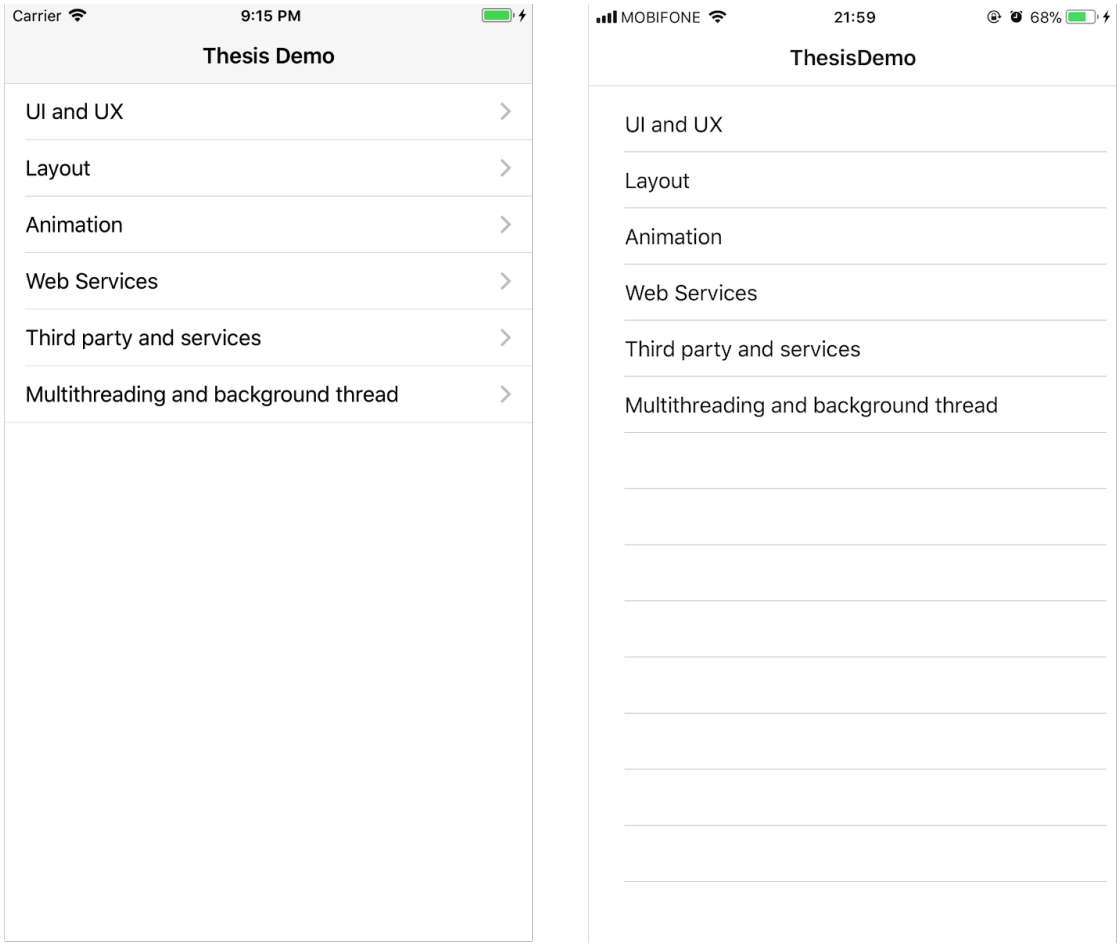
Qua việc tổng hợp những nghiên cứu và phân tích ở các chương trước, để minh họa khả năng phát triển cũng như hiệu năng đối với hai nền tảng Ionic và Xamarin, tôi thực hiện xây dựng một ứng dụng nhỏ theo các tiêu chí so sánh ở chương 3. Bên cạnh đó tôi cũng xây dựng một ứng dụng để so sánh hiệu năng của ba bộ khung phát triển Ionic, Xamarin và native. Từ đó có thể có những kết luận cũng như khuyến nghị cho các nhà phát triển khi xây dựng ứng dụng.

4.1 Ứng dụng so sánh khả năng phát triển trên hai nền tảng

Trong phạm vi luận văn, tôi đã thực hiện xây dựng ứng dụng thực nghiệm nhỏ trên cả hai nền tảng Ionic và Xamarin theo từng tiêu chí đã phân tích ở chương ba để có thể thực hiện việc so sánh và đánh giá một cách khách quan và cụ thể nhất.

4.1.1. Nội dung ứng dụng

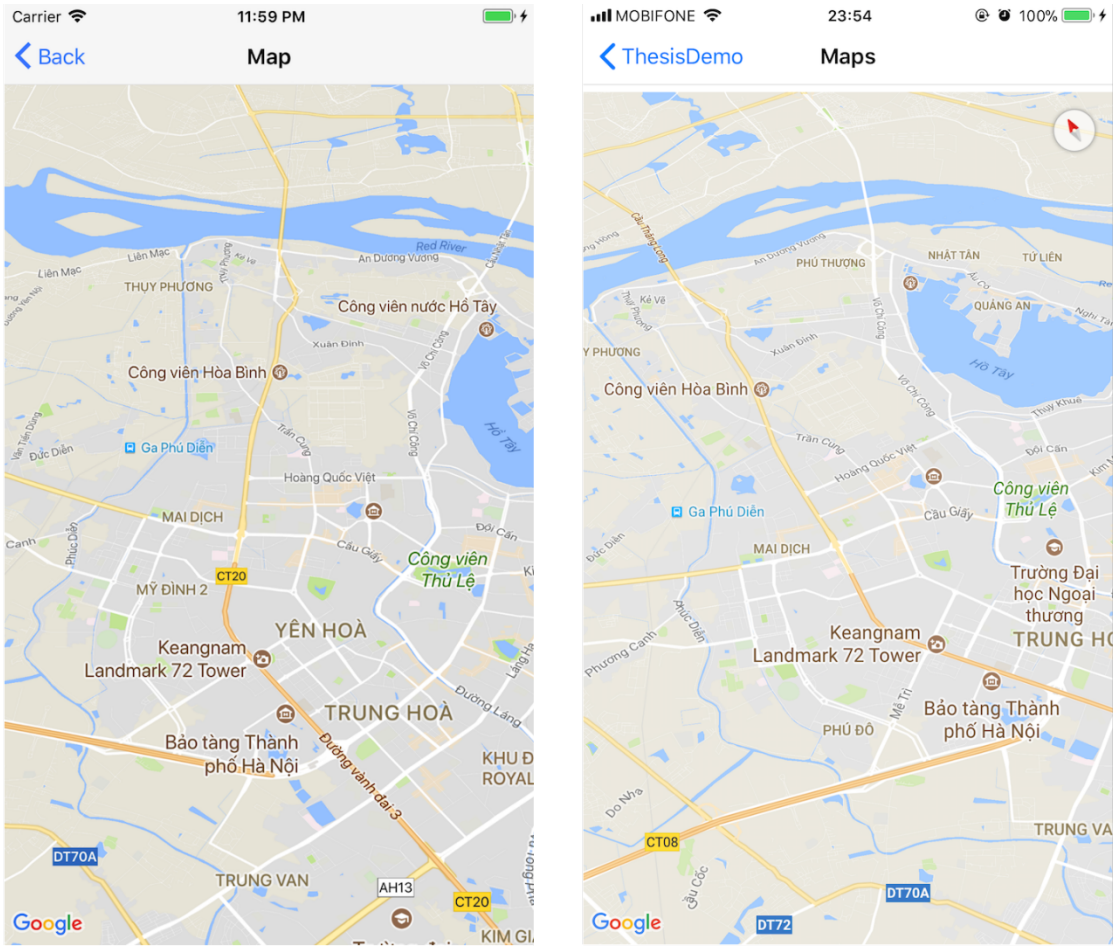
Ứng dụng được xây dựng minh họa việc phát triển các chức năng lần lượt là UI và UX, layout, animation, hỗ trợ của bên thứ ba, đa luồng. Mỗi chức năng được phát triển trên nền tảng iOS dựa trên cả hai bộ khung phát triển là Ionic và Xamarin để qua đó so sánh được mức độ khả thi khi phát triển ứng dụng trên hai nền tảng này.



a. Ionic *b. Xamarin*

Hình 4.1: Ứng dụng thực nghiệm minh họa việc phát triển các chức năng trên Ionic và Xamarin

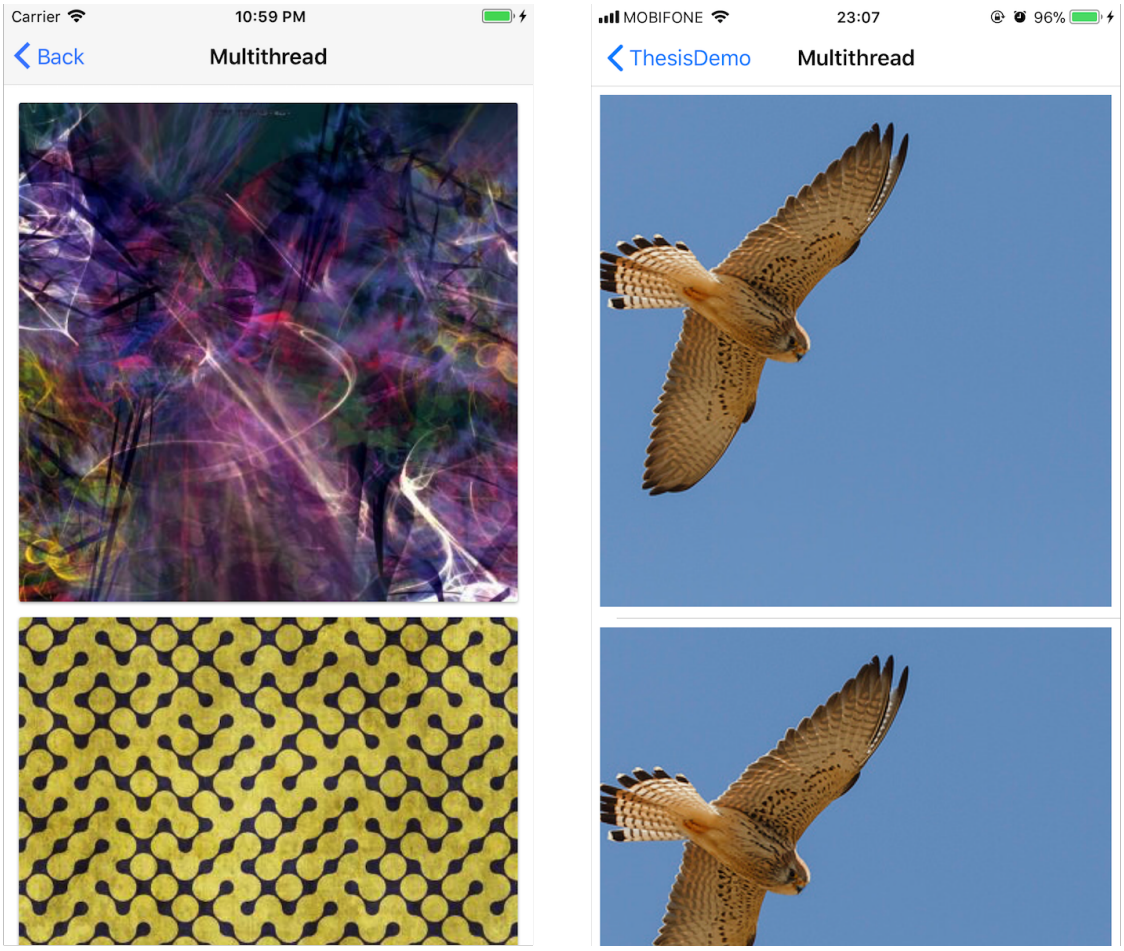
- UI và UX: Ứng dụng được phát triển minh họa việc chuyển đổi bằng các animation tùy chỉnh giữa hai trang giao diện
- Layout: Ứng dụng thực nghiệm được xây dựng với giao diện dạng lưới (chứa các đoạn văn bản dạng text hoặc hình ảnh), giao diện có thể thay đổi kích thước hàng, cột tự động để hiển thị phù hợp với nhiều kích thước màn hình khác nhau.
- Animation: ứng dụng minh họa việc thực hiện animation lung lay khi chạm vào một nút bấm.
- Dịch vụ và thư viện từ bên thứ ba: Ứng dụng thực hiện chức năng tích hợp bản đồ Google Maps và hiển thị trên giao diện.



a. Ionic *b. Xamarin*

Hình 4.2: Giao diện màn hình hiển thị bản đồ Google Maps

– Đa luồng: Để minh họa chức năng này, ứng dụng xây dựng một danh sách cuộn cho phép tải ảnh ở các luồng chạy nền trong khi vẫn thực hiện tương tác với giao diện trên luồng chính



a. Ionic *b. Xamarin*

Hình 4.3: Giao diện màn hình hiển thị danh sách ảnh

4.1.2. Kết quả thực nghiệm

Bảng 4.1: Bảng so sánh đối với từng chức năng trên hai nền tảng Ionic và Xamarin

Chức năng	Ionic	Xamarin
UI và UX	Tôi phải tương tác với các thành phần WebKit để thực hiện tính năng tuỳ chỉnh quá trình chuyển đổi trang. Quá trình này khá phức tạp và cần có kiến thức về cả nền tảng web và native. tôi đã viết đoạn mã để chụp lại ảnh màn hình, sau đó sử dụng thư viện	Sử dụng cơ chế hỗ trợ có sẵn của Xamarin là Custom Renderer [16] kết hợp với tìm hiểu tài liệu về tuỳ chỉnh chuyển đổi trên iOS để thực hiện tính năng này hoàn toàn tương tư như các cách nhà phát triển thực hiện trên nền tảng iOS. Cụ thể tôi sử dụng thư viện CoreAnimation có sẵn

	CoreAnimation có sẵn trong iOS để thực hiện quá trình chuyển đổi. Bên cạnh đó, tôi cũng cần phải viết đoạn mã wrapper bằng javascript để có thể gọi tính năng này từ ứng dụng Ionic. Toàn bộ mã nguồn thực hiện tính năng này là 299 dòng	trong iOS để thực hiện quá trình chuyển đổi "lật" giữa hai trang. Toàn bộ mã nguồn thực hiện tính năng này là 26 dòng
Layout	Cả hai bộ khung phát triển đều có cơ chế riêng để các thành phần giao diện có thể tự động tính toán và thay đổi kích thước đáp ứng lại với các kích cỡ thiết bị khác nhau	
Animation	Để thực hiện animation này thì cần kết hợp các animation riêng lẻ thành một chuỗi và thực hiện lần lượt. Cả hai bộ khung phát triển đều hỗ trợ tốt tính năng này	
Dịch vụ và thư viện từ bên thứ ba	Cả hai bộ khung phát triển đều hỗ trợ cho phép tích hợp Google Maps SDK vào ứng dụng	
Đa luồng	Để thực hiện chức năng cho phép tải ảnh ở các luồng chạy ngầm thì cần có sự kết hợp giữa các API native (NSMutableURLRequest) cho phép thực hiện các yêu cầu ở tầng dưới và thông báo (sử dụng Rxjs) cho các thành phần web cập nhật giao diện khi nội dung đã sẵn sàng.	Sử dụng .NET Api [13] tích hợp sẵn trong Xamarin để tạo các tác vụ chạy ngầm và sử dụng API (Device.InvokeOnMainThread) cho phép tương tác với các thành phần giao diện trên luồng chính để cập nhật giao diện khi nộik dung đã sẵn sàng

Bằng ứng dụng thực nghiệm, chúng ta có thể thấy được phần nào quá trình phát triển các chức năng có thể cần có của một ứng dụng. Cả Ionic và Xamarin đều cung cấp cho các nhà phát triển khả năng để có thể tích hợp các chức năng phổ biến vào ứng dụng của mình. Tuy nhiên, qua ứng dụng thực nghiệm, chúng ta có thể thấy được việc triển

khai một chức năng mà chưa được Ionic hỗ trợ đối với một nhà phát triển thông thường thì phức tạp và khó khăn hơn khá nhiều so với Xamarin. Điều này đến từ sự khác biệt về cách tiếp cận giữa hai bộ khung phát triển. Ứng dụng Xamarin về bản chất vẫn là một ứng dụng native, kết hợp với khả năng tương tác tốt với nền tảng native. Các API của Xamarin còn được xây dựng tương ứng với các API native ở từng nền tảng cụ thể. Điều này giúp cho các lập trình viên có thể sử dụng các tài liệu có sẵn để triển khai các tính năng chưa được hỗ trợ mặc định. Trong khi đó, ứng dụng Ionic chạy trên nền tảng web, các lập trình viên phải tương tác với các thành phần trên nền tảng qua một tầng trung gian là bộ WebKit. Điều này dẫn đến việc các nhà phát triển cần phải có một bộ kĩ năng rộng bao gồm cả về nền tảng web và native để có thể xử lý các vấn đề ở mức độ sâu. Chính lý do này cũng dẫn đến một trong những nhược điểm lớn nhất của Ionic là việc phụ thuộc vào plugin, nói chính xác hơn là phụ thuộc vào một số nhỏ các lập trình viên có kinh nghiệm, có khả năng triển khai các tính năng mới.

4.2. Ứng dụng so sánh hiệu năng

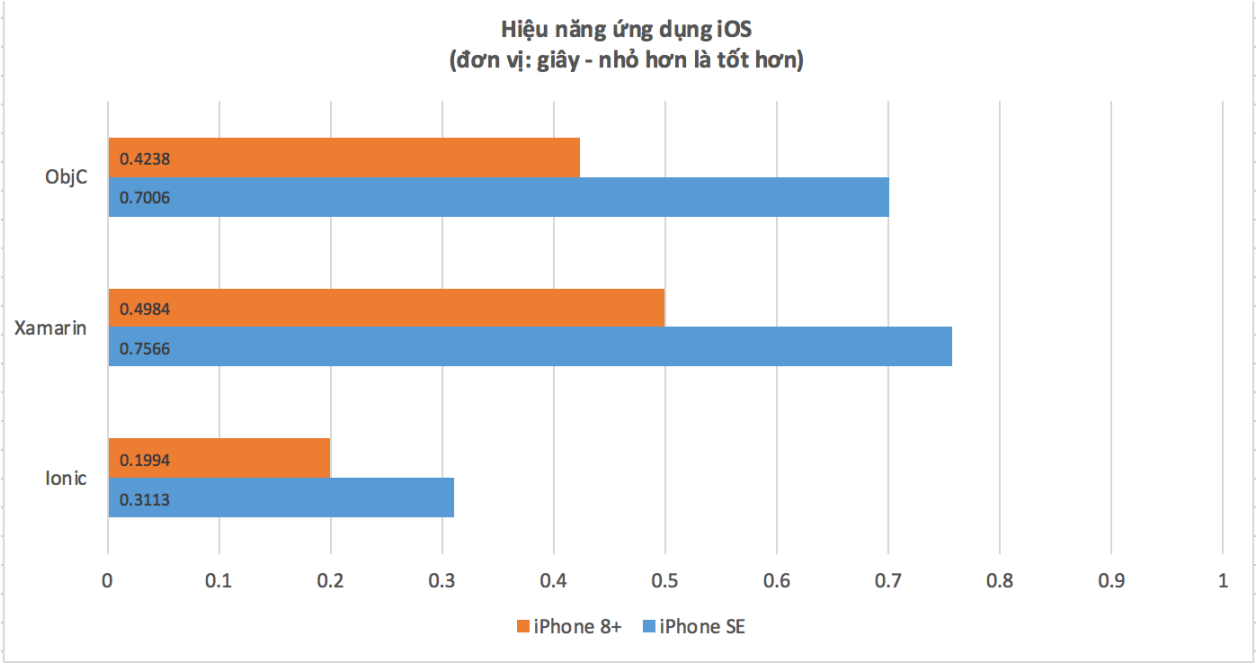
4.2.1. Nội dung thực nghiệm

Khi nhắc đến các bộ khung phát triển ứng dụng đa nền tảng sử dụng công nghệ web, nhiều nhà phát triển sẽ phân vân về hiệu năng của các ứng dụng xây dựng dựa trên các bộ khung phát triển này. Tuy nhiên, hiện tại các bộ khung phát triển ứng dụng di động dựa trên các công nghệ web đã có bước phát triển rất nhanh chóng về hiệu năng nhờ quá trình tối ưu các bộ khung phát triển, sức mạnh phần cứng mạnh hơn của các thiết bị, quá trình tối ưu các engine web trong từng hệ điều hành. Để so sánh hiệu năng của các bộ khung phát triển ứng dụng đa nền tảng, tôi đã viết một ứng dụng nhỏ và đo thời gian các ứng dụng được xây dựng dựa vào các bộ khung xử lý tác vụ trong ứng dụng.

Ứng dụng mô phỏng quá trình xử lý dữ liệu trong một ứng dụng thể thao, cho phép xử lý dữ liệu vị trí của người dùng. Dữ liệu ứng dụng được biểu diễn dưới dạng JSON và CSV bao gồm 531 dòng dữ liệu thể hiện quá trình di chuyển của người dùng. Ứng dụng được xây dựng dựa trên ba nền tảng công nghệ là native iOS, Xamarin và Ionic với cùng một logic. Ứng dụng được thực nghiệm trên hai thiết bị là iPhone SE và iPhone 8+ chạy nền tảng iOS 11. Ứng dụng được thực nghiệm 10 lần với 1000 vòng lặp mỗi lần và lấy kết quả trung bình.

4.2.2. Kết quả thực nghiệm

Biểu đồ dưới đây thể hiện thời gian xử lý của các ứng dụng giống nhau trên nền tảng iOS sử dụng Objective-C, Xamarin và Ionic.



Hình 4.4: So sánh hiệu năng ứng dụng iOS phát triển bằng ObjC, Xamarin và Ionic

Trong phạm vi thử nghiệm, ứng dụng tập trung vào việc xử lý các chuỗi ký tự và tính toán các toạ độ. Qua biểu đồ chúng ta có thể thấy được hiệu năng của ứng dụng phát triển bằng Xamarin khá tương đương với ứng dụng được phát triển bằng cách truyền thông sử dụng các native SDK. Trong khi đó hiệu năng của ứng dụng thử nghiệm khi phát triển bằng Ionic tốt hơn khoảng 2 lần so với ứng dụng phát triển bằng Xamarin và native Objective-C. Điều này tuy không chứng tỏ được rằng hiệu năng của các ứng dụng phát triển bằng Ionic sẽ tốt hơn so với các ứng dụng phát triển bằng Xamarin hay Objective-C nhưng cũng thể hiện được khả năng của Ionic trong việc tối ưu hiệu năng của ứng dụng trong một số trường hợp.

4.3 Khuyến nghị

Ionic và Xamarin đều là những bộ khung phát triển ứng dụng di động đa nền tảng tốt và có những lợi thế nhất định của mình. Nếu như Xamarin mạnh về khả năng tương tác với các thư viện native, hiệu năng và khả năng tuỳ biến thì Ionic mạnh về tính nhất quán cần có của một bộ khung phát triển đa nền tảng. Việc lựa chọn bộ khung phát triển phù hợp phù thuộc vào yêu cầu của ứng dụng và nền tảng công nghệ của lập trình viên. Nếu như ứng dụng yêu cầu không quá phức tạp, không cần xử lý quá nhiều dữ liệu, chỉ sử dụng các dịch vụ phổ biến, cần có thời gian phát triển nhanh, chi phí hạn chế thì Ionic là một lựa chọn rất tốt. Trong khi đó nếu như ứng dụng yêu cầu hiệu năng tốt, cần xử lý

nhiều tác vụ nặng, cần tuỳ biến nhiều trên từng nền tảng hoặc sử dụng một tính năng đặc biệt nào đó trên một nền tảng hoặc sử dụng nhiều các thư viện native thì Xamarin là lựa chọn tốt hơn.

CHƯƠNG 5: KẾT LUẬN

Luận văn tốt nghiệp đã giới thiệu một cách tổng quát các cách tiếp cận để phát triển một ứng dụng di động. Dựa vào xu thế phát triển của các bộ khung phát triển di động đa nền tảng, luận văn đã lựa chọn giới thiệu, phân tích ưu nhược điểm và so sánh hai bộ khung phát triển là Ionic và Xamarin, đại diện cho hai trường phái phát triển ứng dụng đa nền tảng sử dụng công nghệ web và công nghệ native. Cụ thể, luận văn thực hiện việc phân tích và so sánh dựa trên các tiêu chí cần thiết mà các nhà phát triển quan tâm để phát triển một ứng dụng di động như giao diện, trải nghiệm người dùng, hiệu năng, đa luồng, sự hỗ trợ các dịch vụ của bên thứ ba và kiểm thử tự động. Qua phân tích đã cho thấy khả năng tương thích tốt với các thư viện native, khả năng tuỳ biến trên từng nền tảng của Xamarin, tuy nhiên đi kèm với đó là việc tính đa nền tảng có thể không được đảm bảo cao, dẫn đến việc kéo Xamarin gần trở thành một bộ khung phát triển ứng dụng native hơn là bộ khung phát triển đa nền tảng. Trong khi đó Ionic mặc dù có khả năng tương thích với các nền tảng kém hơn, phụ thuộc nhiều vào các trình cắm và cần các lập trình viên có kinh nghiệm hơn thì lại có tính đa nền tảng cao hơn, nền tảng công nghệ phổ biến hơn.

Để minh hoạ những phân tích và so sánh đã đưa ra ở trên, trong phạm vi luận văn cũng xây dựng một ứng dụng nhỏ dựa theo các tiêu chí so sánh. Luận văn đánh giá khả năng phát triển của hai nền tảng Ionic và Xamarin dựa vào cách tiếp cận và số dòng mã nguồn cần sử dụng để triển khai các tính năng tương tự nhau dựa trên hai bộ khung phát triển trên cùng một nền tảng. Bên cạnh đó, luận văn cũng xây dựng một ứng dụng dựa vào việc xử lý các chuỗi và tính toán dữ liệu để có thể so sánh hiệu năng giữa ba cách tiếp cận trong việc phát triển ứng dụng di động. Kết quả cho thấy được trong một số trường hợp hiệu năng của Ionic và Xamarin là tốt, có thể so sánh với các ứng dụng xây dựng dựa trên nền tảng native.

Tổng kết lại, việc lựa chọn bộ khung phát triển phù hợp phụ thuộc vào yêu cầu của ứng dụng và khả năng của các lập trình viên. Ionic phù hợp với các ứng dụng không quá phức tạp, ít tuỳ biến, không yêu cầu xử lý nhiều dữ liệu, hiệu năng ở mức tương đối, sử dụng các dịch vụ phổ biến hoặc các lập trình viên có sẵn kinh nghiệm nền tảng công nghệ web, muốn tiết kiệm thời gian và chi phí phát triển. Xamarin phù hợp với các ứng dụng lớn hơn, cần tuỳ biến nhiều, hiệu năng tốt, muốn tiết kiệm một phần chi phí và thời gian phát triển ứng dụng.

TÀI LIỆU THAM KHẢO

[1] Ionic team, Ionic docs, http://ionicframework.com/docs/.

[2] Xamarin team, Xamarin docs, http://developer.xamarin.com/guides/cross-platfrom

[3] Xamarin team, Architecture, https://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_2_-_architecture/

[4] Gartner, Market Share, "Mobile communination devices(2012)".

[5] Charland A., Leroux B., "Mobile application development: web vs native," in *ACM 54*, pp. 49-53, 2011

[6] Goadrich M. H., Rogers M.P, "Smart smartphone development: iOS versus Android", in *Proc. SIGCSE 2011*, pp. 607-612, New York, 2011.

[7] Anderson R.S., Gestwicki P., "Hello, worlds: an introduction to mobile application development for iOS and Android". *J. Comput. Sci. Coll. 27*, pp. 32–33, 2011.

[8] Newman B, "Are cross-platform mobile app frameworks right for your business?", 2011, http://mashable.com/2011/03/21/cross-platform-mobile-frameworks/.

[9] Behrens H., "Cross-Platform App Development for iPhone, Android & Co", 2010, http://heikobehrens.net/2010/10/11/cross-platform-app-development-for-iphone-android-co-—-a-comparison-i-presented-at-mobiletechcon-2010/

[10] Cordova team, Cordova guide, https://cordova.apache.org/docs/en/latest/guide/overview/

[11] John Resig, "How javascript timer work", https://johnresig.com/blog/how-javascript-timers-work/.

[12] Tom Buyse, "End to end testing an Ionic application with appium and protractor", http://tombuyse.com/end-to-end-testing-an-ionic-application-with-appium-and-protractor/.

[13] Appium Team, Appium docs, http://appium.io/slate/en/master/?ruby - about-appium

[14] Microsoft Team, Thread Pools, https://msdn.microsoft.com/en-us/library/windows/desktop/ms686760(v=vs.85).aspx

[15] Xamarin Team, Linking native libraries, https://developer.xamarin.com/guides/ios/advanced_topics/native_interop/

[16] Xamarin Team, Custom Renderer, https://developer.xamarin.com/guides/xamarin-forms/application-fundamentals/custom-renderer/introduction/

[17] Estaun.net blog, Some thoughts after (almost) a year of real Xamarin use, http://www.estaun.net/blog/some-thoughts-after-almost-a-year-of-real-xamarin-use/

[18] Xamarin Team, Limitations, https://developer.xamarin.com/guides/ios/advanced_topics/limitations/

[19] Xamarin Team, Limitations, https://developer.xamarin.com/guides/android/advanced_topics/limitations/

[20] Herman Schoenfeld, Xamarin iOS memory leaks everywhere, https://stackoverflow.com/questions/25532870/xamarin-ios-memory-leaks-everywhere

[21] Nexgendesign.com, Xamarin troubles, http://www.nexgendesign.com/xamarin-troubles

[22] Siddharth, 15 important consideration for choosing a web dev framework, 2009, https://code.tutsplus.com/tutorials/15-important-considerations-for-choosing-a-web-dev-framework--net-8035

[23] Daniel Pfeiffer, Which cross-platform framework is right for me?, 2011, https://gowithfloat.com/2011/07/which-cross-platform-framework-is-right-for-me/

[24] Heitk¨otter, H., Hanschke, S., Majchrzak, T.A.: Evaluating cross-platform development approaches for mobile applications. In: Cordeiro, J., Krempels, K.-H. (eds.) Web Information Systems and Technologies. LNBIP, vol. 140, pp. 120–138. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36608-6_8

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Số: 1156 /QĐ-ĐT

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Hà Nội, ngày 23 tháng 11 năm 2017

QUYẾT ĐỊNH
Về việc thành lập Hội đồng chấm luận văn thạc sĩ

HIỆU TRƯỞNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Căn cứ Quy định về Tổ chức và hoạt động của các đơn vị thành viên và đơn vị trực thuộc Đại học Quốc gia Hà Nội ban hành theo quyết định số 3568/QĐ-ĐHQGHN ngày 08/10/2014 của Giám đốc Đại học Quốc gia Hà Nội;

Căn cứ Quy định về Tổ chức và hoạt động của Trường ĐH Công nghệ ban hành kèm theo Quyết định số 520/QĐ-ĐHCN ngày 19/7/2016 của Hiệu trưởng Trường ĐH Công nghệ;

Căn cứ Quy chế đào tạo sau đại học ở Đại học Quốc gia Hà Nội, ban hành kèm theo Quyết định số 1555/QĐ-ĐHQGHN ngày 25/5/2011 và Quyết định sửa đổi, bổ sung số 3050/QĐ-ĐHQGHN ngày 17/9/2012 của Giám đốc Đại học Quốc gia Hà Nội;

Căn cứ Quy chế Đào tạo thạc sĩ tại Đại học Quốc gia Hà Nội, ban hành theo Quyết định số 4668/QĐ-ĐHQGHN ngày 10/12/2014 của Giám đốc Đại học Quốc gia Hà Nội ;

Căn cứ Quyết định công nhận học viên cao học số 660/QĐ-CTSV ngày 18/09/2014 của Hiệu trưởng Trường Đại học Công nghệ;

Căn cứ Công văn số 153/CNTT-ĐT ngày 31/10/2017 và Công văn số 169/CNTT-ĐT ngày 21/11/2017 của Chủ nhiệm Khoa Công nghệ thông tin về việc đề xuất hội đồng chấm luận văn;

Xét đề nghị của Trưởng phòng Đào tạo,

QUYẾT ĐỊNH:

Điều 1. Thành lập Hội đồng chấm luận văn thạc sĩ của học viên Hồ Danh Chuẩn, sinh ngày 10/05/1991 tại Nghệ An là học viên cao học khóa 21,

Ngành: Công nghệ thông tin

Chuyên ngành: Kỹ thuật phần mềm Mã số: 60480103

Tên đề tài luận văn: Tìm hiểu đánh giá các framework phát triển ứng dụng di động đa nền tảng.

Cán bộ hướng dẫn: TS. Trần Thị Minh Châu

Danh sách các thành viên Hội đồng kèm theo quyết định này.

Điều 2. Chủ nhiệm Khoa Công nghệ thông tin có nhiệm vụ tổ chức để học viên bảo vệ luận văn thạc sĩ trước Hội đồng theo đúng Quy chế Đào tạo thạc sĩ ở Đại học Quốc gia Hà Nội và các quy định hiện hành khác. Hội đồng tự giải thể sau khi hoàn thành nhiệm vụ.

Điều 3. Trưởng phòng Hành chính – Quản trị, Trưởng phòng Đào tạo, Chủ nhiệm Khoa Công nghệ thông tin, các Thủ trưởng đơn vị có liên quan, các thành viên Hội đồng và học viên Hồ Danh Chuẩn chịu trách nhiệm thi hành quyết định này./.

KT. HIỆU TRƯỞNG
PHÓ HIỆU TRƯỞNG

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Chử Đức Trình

Nơi nhận:
- Như Điều 3;
- Lưu: VT, ĐT, CH11.

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc
==========

BẢN NHẬN XÉT PHẢN BIỆN LUẬN VĂN THẠC SĨ

Họ và tên cán bộ phản biện: **Võ Đình Hiếu**

Học hàm, học vị: Tiến sĩ

Chuyên ngành: Công nghệ phần mềm

Cơ quan công tác: Khoa CNTT - Trường ĐH Công nghệ

Họ và tên học viên cao học: Hồ Danh Chuẩn

Tên đề tài luận văn: **Tìm hiểu đánh giá các framework phát triển ứng dụng di động đa nền tảng**

Chuyên ngành: Kỹ thuật phần mềm Mã số: 60480103

Ý KIẾN NHẬN XÉT

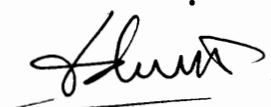
Học viên tìm hiểu về các framework dùng để phát triển các ứng dụng di động đa nền tảng. Cụ thể, học viên đã trình bày về hai framework chính là Ionic và Xamarin. Luận văn thể hiện học viên nắm vững kiến thức về các framework phát triển ứng dụng di động. Cấu trúc luận văn tương đối hợp lý. Luận văn còn những hạn chế sau.

- Luận văn có quá nhiều lỗi ngữ pháp, diễn đạt ("Sự bùng nổ... người dùng", trang 6, "Để giúp các.... đa nền tảng", trang 7)
- Hình vẽ nên có giải thích (ví dụ hình 2.2)
- Sử dụng nhiều từ tiếng Anh
- Nên phân loại các tiêu chí được sử dụng để đánh giá và nêu rõ cách chọn các tiêu chí này

Kết luận: luận văn đáp ứng yêu cầu cơ bản của một luận văn thạc sĩ.

Hà Nội, ngày 2 tháng 12 năm 2017

PHẢN BIỆN



Võ Đình Hiếu

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BẢN NHẬN XÉT PHẢN BIỆN LUẬN VĂN THẠC SĨ

Họ và tên cán bộ phản biện: Lê Quang Minh
Học hàm, học vị: Tiến sỹ
Chuyên ngành: Máy tính và Hệ thống tính toán
Cơ quan công tác: Viện Công nghệ thông tin - ĐHQGHN

Họ và tên học viên cao học: Hồ Danh Chuẩn
Tên đề tài luận văn: Tìm hiểu đánh giá các framework phát triển ứng dụng di động đa nền tảng.
Chuyên ngành: Kỹ thuật phần mềm Mã số:

Ý KIẾN NHẬN XÉT

Đề tài có ý nghĩa khoa học, công nghệ và thực tiễn. Học viên đã tìm hiểu về 2 nền tảng Ionic và Xamarin, là các framework để phát triển trên hệ điều hành iOS đối với các ứng dụng trên di động.

Luận văn đã đánh giá và so sánh 2 nền tảng Ionic và Xamarin, đưa ra các khuyến nghị đối với việc phát triển trên 2 nền tảng này.

Luận văn nêu bố cục, trình bày lại chương ba (chương 3.) và giải thích rõ về cách lựa chọn các tiêu chí đánh giá so sánh với 2 nền tảng Ionic và Xamarin.

Luận văn còn có lỗi đánh máy và trích dẫn tài liệu.

Luận văn đáp ứng các yêu cầu của bản luận văn thạc sỹ chuyên ngành Kỹ thuật phần mềm. Tôi đồng ý cho học viên được bảo vệ trước hội đồng chấm luận văn thạc sỹ.

Hà Nội, ngày 2 tháng 12 năm 2017

XÁC NHẬN CỦA CƠ QUAN CÔNG TÁC **CÁN BỘ PHẢN BIỆN**

Lê Quang Minh

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
------📖------

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc
****

Hà Nội, ngày 02 tháng 12 năm 2017

QUYẾT NGHỊ
CỦA HỘI ĐỒNG CHẤM LUẬN VĂN THẠC SĨ

Căn cứ Quyết định số **1156/QĐ-ĐT, ngày 23 tháng 11 năm 2017** của Hiệu trưởng trường Đại học Công nghệ về việc thành lập Hội đồng chấm luận văn thạc sĩ của học viên **Hồ Danh Chuẩn,** Hội đồng chấm luận văn Thạc sĩ đã họp vào **11h, thứ 7, ngày 02 tháng 12 năm 2017, Phòng 309 , Nhà E3,** Trường Đại học Công nghệ - ĐHQGHN.

Tên đề tài luận văn: **Tìm hiểu đánh giá các framework phát triển ứng dụng di động đa nền tảng**

Ngành: **Công nghệ Thông tin**

Chuyên ngành: **Kỹ thuật phần mềm** Mã số:

Sau khi nghe học viên trình bày tóm tắt luận văn Thạc sĩ, các phản biện đọc nhận xét, học viên trả lời các câu hỏi, Hội đồng đã họp, trao đổi ý kiến và thống nhất kết luận:

1. Về tính cấp thiết, tính thời sự, ý nghĩa lý luận và thực tiễn của đề tài luận văn:

- Hiện nay các thiết bị di động rất đa dạng, nhiều loại. Các ứng dụng phải cần phải hoạt động trên nhiều môi trường khác nhau. Luận văn nghiên cứu về đánh giá các framework phát triển ứng dụng di động đa nền tảng có ý nghĩa thiết thực trong phát triển ứng dụng di động.

2. Về bố cục, phương pháp nghiên cứu, tài liệu tham khảo, ... của luận văn:

Bố cục luận văn gồm 5 chương. Kết cấu hợp lý. Phương pháp nghiên cứu phù hợp, tài liệu tham khảo cập nhật.

3. Về kết quả nghiên cứu:

- Nghiên cứu tìm hiểu các công nghệ đa nền tảng Ionic và Xamarin

- Đánh giá, đề xuất mô hình tổ chức, hiệu quả của các biện pháp
- Xây dựng quy trình điều trị.

4. **Hạn chế của luận văn *(nếu có)*:**

- Luận văn còn có thể bổ
- Nên có cái nhìn của chuyên gia về Đánh giá

5. **Đánh giá chung và kết luận:**

Luận văn đáp ứng yêu cầu của một luận văn thạc sỹ ngành CKII, cần bổ sung KTPH. Chỉnh sửa theo yêu cầu hội đồng

Luận văn đạt 7,2/ 10 điểm. Quyết nghị này được 05 / 05 thành viên của Hội đồng nhất trí thông qua.

THƯ KÝ HỘI ĐỒNG

T. N. Thuần

CHỦ TỊCH HỘI ĐỒNG

Trần Anh Hoàng

XÁC NHẬN CỦA CƠ SỞ ĐÀO TẠO

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

Hà Nội, ngày 08 tháng 12 năm 2017

BẢN XÁC NHẬN SỬA CHỮA CÁC THIẾU SÓT CỦA LUẬN VĂN

Trường Đại học Công nghệ đã có quyết định số 1156/QĐ-ĐT ngày 23 tháng 11 năm 2017 về việc thành lập Hội đồng chấm luận văn Thạc sĩ cho học viên Hồ Danh Chuẩn, sinh ngày 10/05/1991 tại Nghệ An, chuyên ngành Kỹ thuật phần mềm, ngành Công nghệ thông tin.

Ngày 02 tháng 12 năm 2017, Trường Đại học công nghệ (ĐHCN) đã tổ chức cho học viên bảo vệ luận văn Thạc sĩ trước Hội đồng chấm (có biên bản kèm theo). Theo quyết nghị của Hội đồng chấm luận văn Thạc sĩ, học viên phải bổ sung và sửa chữa các điểm sau đây trước khi nộp quyển luận văn cuối cùng cho nhà trường để hoàn thiện hồ sơ sau bảo vệ:

1. Sửa lại lỗi ngữ pháp và trình bầy (trang 6, 7).
2. Bổ sung giải thích về hình minh họa 2.2.
3. Nêu rõ cách lựa chọn các tiêu chí đánh giá: bổ sung nội dung tham khảo lựa chọn các tiêu chí đánh giá dựa vào một số bài báo khoa học.

Ngày 11 tháng 12 năm 2017, học viên đã nộp bản luận văn có chỉnh sửa. Chúng tôi nhận thấy rằng nội dung, hình thức của luận văn và tóm tắt luận văn đã được sửa chữa, bổ sung theo các điểm trên của Quyết nghị.

Đề nghị trường Đại học Công nghệ, ĐHQG HN cho phép học viên được làm các thủ tục khác để được công nhận và cấp bằng Thạc sĩ.

Xin trân trọng cảm ơn!

XÁC NHẬN CỦA HỘI ĐỒNG
ĐỀ NGHỊ HỌC VIÊN SỬA CHỮA LUẬN VĂN

Lê Quang Minh — Võ Đình Thắng — Trương Anh Hoàng

HỌC VIÊN	CÁN BỘ HƯỚNG DẪN	XÁC NHẬN CỦA TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TL. HIỆU TRƯỞNG
Hồ Danh Chuẩn	Trần Thị Minh Châu	

TRƯỞNG PHÒNG ĐÀO TẠO
Nguyễn Phương Thái