**ĐỒ ÁN TỐT NGHIỆP**

**Thiết kế ứng dụng điều khiển cửa cuốn tự động**

**Đinh Duy Anh**

Anh.dd150036@sis.hust.edu.vn

**Ngành Điện Tử Viễn Thông**

**Chuyên ngành Kỹ Thuật Máy Tính**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | ThS. Vũ Song Tùng  Chữ ký của GVHD |
| **Bộ môn:** |  |
| **Viện:** | Điện Tử Viễn Thông |
| **HÀ NỘI, 6/2022** | |

**ĐỀ TÀI TỐT NGHIỆP**

Biểu mẫu của Đề tài/khóa luận tốt nghiệp theo qui định của viện, tuy nhiên cần đảm bảo giáo viên giao đề tài ký và ghi rõ họ và tên.

Trường hợp có 2 giáo viên hướng dẫn thì sẽ cùng ký tên.

Giáo viên hướng dẫn

Ký và ghi rõ họ tên

MỤC LỤC

[**CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI** 4](#_Toc104929560)

[1.1 Vấn đề 4](#_Toc104929561)

[1.2 Internet 4](#_Toc104929562)

[1.2.1 Internet là gì? 4](#_Toc104929563)

[1.2.2 Internet vận hành thế nào? 5](#_Toc104929564)

[1.2.3 Lợi ích 5](#_Toc104929565)

[1.3 Giới thiệu ứng dụng 6](#_Toc104929566)

[1.4 Kết luận 6](#_Toc104929567)

[CHƯƠNG 2. ĐỊNH HƯỚNG GIẢI PHÁP PHÁT TRIỂN ỨNG DỤNG 8](#_Toc104929568)

[2.1 Đặt vấn đề 8](#_Toc104929569)

[2.2 Giải pháp 8](#_Toc104929570)

[2.3 Giới thiệu về công cụ được sử dụng 8](#_Toc104929571)

[2.3.1 Visual Studio 8](#_Toc104929572)

[2.3.2 Xamarin 10](#_Toc104929573)

[2.3.3 Mô hình MVVM 16](#_Toc104929574)

[2.3.4 NoSQL 23](#_Toc104929575)

[2.3.5 Firebase 27](#_Toc104929576)

[2.4 Kết luận 36](#_Toc104929577)

[CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG LANDSLIDE TRÊN XAMARIN 38](#_Toc104929578)

[3.1 Mô hình ứng dụng 38](#_Toc104929579)

[3.2 Phân tích hệ thống 39](#_Toc104929580)

[3.2.1 Usercase của hệ thống 39](#_Toc104929581)

[3.2.2 User đăng nhập 40](#_Toc104929582)

[3.2.3 Usercase Landslide map 40](#_Toc104929583)

[3.2.4 User case cấu hình 40](#_Toc104929584)

[3.3 Hoạt động hệ thống 41](#_Toc104929585)

[3.3.1 Chức năng đăng nhập, đăng ký 41](#_Toc104929586)

[3.3.2 Chức năng landslide map 42](#_Toc104929587)

[3.3.3 Chức năng cấu hình 43](#_Toc104929588)

[3.4 Triển khai trên Xamarin 43](#_Toc104929589)

[3.4.1 tạo project Xamarin trên visual 2019 (Windows) 43](#_Toc104929590)

[3.4.2 Tạo khuôn mô hình MVVM 48](#_Toc104929591)

[3.4.3 Tiến hành triển khai mô hình MVVM 49](#_Toc104929592)

[3.5 Một số chức năng thực hiện 53](#_Toc104929593)

[3.5.1 Chức năng đăng nhập (login) 53](#_Toc104929594)

[DANH MỤC THAM KHẢO 57](#_Toc104929595)

[KẾT LUẬN 58](#_Toc104929596)

DANH MỤC HÌNH ẢNH

[Figure 1: Internet là gì? 8](#_Toc104929597)

[Figure 2: Kết nối mọi người với nhau 9](#_Toc104929598)

[Figure 3: Quá trình phát triển của Xamarin 14](#_Toc104929599)

[Figure 4: Điểm mạnh của Xamarin 14](#_Toc104929600)

[Figure 5: Danh sách các nền tảng hỗ trợ 15](#_Toc104929601)

[Figure 6: Lợi ích của Xamarin 15](#_Toc104929602)

[Figure 7: Đoạn Code màn hình Login 16](#_Toc104929603)

[Figure 8: Build trên 3 nền tảng khác nhau 16](#_Toc104929604)

[Figure 9: Chia sẻ code ở mọi nơi 17](#_Toc104929605)

[Figure 10: Mô hình MVVM 20](#_Toc104929606)

[Figure 11: Mô hình MVVM 21](#_Toc104929607)

[Figure 12: Mô hình MVC 23](#_Toc104929608)

[Figure 13: Mô hình MVP 25](#_Toc104929609)

[Figure 14: Firebase 31](#_Toc104929610)

[Figure 15: Realtime database 32](#_Toc104929611)

[Figure 16: Firebase authentication 32](#_Toc104929612)

[Figure 17: Firebase hosting 33](#_Toc104929613)

[Figure 18: Website firebase 34](#_Toc104929614)

[Figure 19: Firebase auth + google sign in 35](#_Toc104929615)

[Figure 20: Firebase ML 36](#_Toc104929616)

[Figure 21: Opensource firebase 37](#_Toc104929617)

[Figure 22: Firebase + google cloud 39](#_Toc104929618)

[Figure 23: Mô hình ứng dụng 41](#_Toc104929619)

[Figure 24: Usercase ứng dụng Landslide Survey 42](#_Toc104929620)

[Figure 25: Usercase landslide map 43](#_Toc104929621)

[Figure 26: Usercase cấu hình 44](#_Toc104929622)

[Figure 27: Activity đăng nhập, đăng ký 45](#_Toc104929623)

[Figure 28: landslide map 45](#_Toc104929624)

[Figure 29: activity cấu hình 46](#_Toc104929625)

[Figure 30: mở visual studio 2019 47](#_Toc104929626)

[Figure 31: tạo project xamairn trên visual studio 48](#_Toc104929627)

[Figure 32: thông tin cơ bản của project 49](#_Toc104929628)

[Figure 33: loại template cho ứng dụng 50](#_Toc104929629)

[Figure 34: hoàn thành việc tạo project xamairn 51](#_Toc104929630)

[Figure 35: tạo folder theo chuẩn mô hình MVVM 51](#_Toc104929631)

[Figure 36: Tạo pattern MVVM 52](#_Toc104929632)

[Figure 37: Binding context với ViewModel 53](#_Toc104929633)

[Figure 38: BaseViewModel 54](#_Toc104929634)

[Figure 39: form chung của các ViewModel 55](#_Toc104929635)

[Figure 40: giao diện xaml binding data 55](#_Toc104929636)

[Figure 41: model của thông tin User 56](#_Toc104929637)

[Figure 42: UI màn hình đăng nhập 56](#_Toc104929638)

[Figure 43: Properties của LoginViewModel 57](#_Toc104929639)

[Figure 44: khởi tạo các giá trị cơ bản 57](#_Toc104929640)

[Figure 45: xử lý chức năng login 58](#_Toc104929641)

[Figure 46: giao diện đăng nhập 59](#_Toc104929642)

**CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI**

Chương 1 trình bày tổng quan ứng dụng thống kê sạt lở đất, mô tả hoạt động của ứng dụng, cách vận hành của ứng dụng trên điện thoại di động.

* 1. Vấn đề

Trước tình hình biến đổi khí hậu ở khắp nơi trên thế giới nói chung và Việt Nam nói riêng. Việc biến đổi khí hậu diễn ra rất nhiều nơi gây nhiều các hậu quả nghiêm trọng đến đời sống của người dân … Trong đó vấn đề quan trọng không kém bị tác động nặng là những nơi có người dân sinh sống bị sạt lở đất hình thành các hố gây nguy hiểm cho mọi thứ xung quanh đó. Trước tính hình đó việc cập nhật thông tin tin tức cho mọi người và người dân xung quanh là rất cần thiết và cấp bách.

Khi đó, giải pháp cấp bách lúc này là có một thứ gì đó có thể cập nhật ngay lập tức thông tin cho mọi người về những khu vực sạt lở đất bất kể là ai cũng có thể nhận được thông báo.

Trước những diễn biến như vậy tôi cũng tiến hành phát triển một ứng dụng Landslide sẽ cập nhật tất cả các thông tin này cho mọi người sử dụng giữ liệu biết về thông tin các vụ sạt lở đất. Không chỉ dừng lại ở đó ứng dụng còn mang hướng thương mại hóa, mở rộng để có thể phát triển thành một ứng dụng đem lại nhiều sự hứng thú, lợi ích cho mọi người dùng.

* 1. Internet

1.2.1 Internet là gì?

Internet là mạng toàn cầu bao gồm hàng tỷ máy tính và các thiết bị điện tử khác. Với Internet, bạn có thể truy cập hầu hết mọi thông tin, giao tiếp với bất kỳ ai khác trên thế giới và bạn còn làm được nhiều hơn thế nữa.

Bạn có thể làm tất cả điều này bằng cách kết nối máy tính với Internet, hay còn được gọi là going online. Khi ai đó nói rằng máy tính đang trực tuyến, đó là cách nói khác chỉ rằng nó đang được kết nối với Internet.Ứng dụng của Internet.



Figure 1: Internet là gì?

1.2.2 Internet vận hành thế nào?

Internet là một mạng lưới cáp vật lý toàn cầu, có thể bao gồm dây đồng điện thoại, cáp TV và cáp quang. Ngay cả các kết nối không dây như Wi-Fi và 3G/4G/5G cũng dựa vào các loại cáp vật lý này để truy cập Internet.

Khi bạn truy cập một trang web, máy tính của bạn sẽ gửi một yêu cầu qua các dây này tới một máy chủ. Máy chủ là nơi lưu trữ các trang web và nó hoạt động giống như ổ cứng máy tính của bạn. Khi yêu cầu đến, máy chủ sẽ truy xuất trang web và gửi dữ liệu chính xác trở lại máy tính của bạn. Thao tác truy vấn này, tất cả được thực hiện chỉ trong vài giây.

1.2.3 Lợi ích

Một trong những tính năng tốt nhất của Internet là khả năng giao tiếp gần như ngay lập tức với bất kỳ ai trên thế giới. Email là một trong những cách lâu đời nhất và phổ biến nhất để giao tiếp và chia sẻ thông tin trên Internet với hàng tỷ người đang sử dụng nó. Mạng xã hội cho phép mọi người kết nối theo nhiều cách khác nhau và xây dựng cộng đồng trực tuyến.

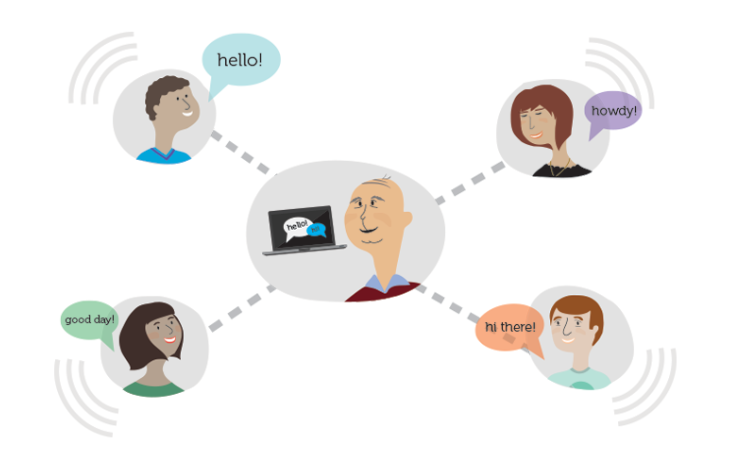


Figure 2: Kết nối mọi người với nhau

Có rất nhiều tính năng khác bạn có thể làm trên Internet. Không chỉ vậy, ngày nay với công nghệ hiện tại rất nhiều các thiết bị hiện đại tích hợp có thể kết nối Internet giúp mọi người dễ dàng tiếp cận và đạt được nhu cầu cá nhân.

1.3 Giới thiệu ứng dụng

Ứng dụng được hướng đến có tên là: “Landslide”. Ứng dụng đáp ứng được yêu cầu quan trọng nhất là cập nhật những thông tin về những vụ sạt lở đất diễn ra trên toàn bộ khu vực. Ngoài ra, ứng dụng có đầy đủ thông tin về người dùng và thông tin về những vụ sạt lở đất.

Tương lai, khi ứng dụng có những thành quả nhất định thì sẽ phát triển mở rộng hơn và hướng tới nhiều chức năng thuận tiện, đáp ứng nhu cầu của người dùng.

1.4 Kết luận

Với những mục tiêu và định hướng được trình bày ở trên thì đề tài mà tôi hướng đến là một Ứng dụng trên điện thoại ở cả hai nền tảng đang được sử dụng nhiều nhất là: Android và IOS.

Để làm được điều đó, ta cần một Framework phù hợp cho việc phát triển ứng dụng với công sức bỏ ra là tối thiểu mà vẫn đạt được hiệu quả cao và mục tiêu. Ngoài ra, còn phải đáp ứng được tính mở rộng, ứng dụng để có thể phát triển và cải thiện liên tục đáp ứng được tất cả các nhu cầu của người dung, khách hàng.

**CHƯƠNG 2. ĐỊNH HƯỚNG GIẢI PHÁP PHÁT TRIỂN ỨNG DỤNG**

Chương 2 trình bày về vấn đề được đưa ra của đề tài, hướng giải quyết và các công cụ để xây dựng hệ thống, ứng dụng thống kê Landslide trên hai nền tảng Android và IOS.

2.1 Đặt vấn đề

Bài toán đưa ra cần một thiết bị nhỏ gọn, dễ sự dụng để mọi người có thể đọc và xem thông tin các khu vực ở bất cứ đâu khi nào, lúc nào. Ngày này chiếc điện thoại di động không còn gì xa lạ với chúng ta, nó không chỉ cuing cấp ứng dụng nghe gọi mà còn nhiều ứng dụng giải trí, hỗ trợ công việc, … Mà ngày nay đi đâu con người cũng gần như mang chiếc điện thoại bên mình. Chính vì thế việc xây dựng ứng dụng Landslide trên nền tang điện thoại mang lại lợi ích và sự thuận tiện rất lớn.

Tuy vậy, ngoài việc phát triển ứng dụng vấn còn cách khác là phát triển một Website có chức năng giống như ứng dụng. Tuy vậy, sẽ gây ra số bất lợi như chúng ta phải nhớ đường dẫn url của website và phải đăng nhập thường xuyên, phải dung trình duyệt. Thay vì thế, việc phát triển ứng dụng tạo lợi thế rất lớn là có thể lưu giữ nhiều data hơn, lưu được đăng nhập, ứng dụng có sẵn trên cửa hang có thể tải xuống.

2.2 Giải pháp

Có được lựa chọn cho vấn đề là xây dựng App trên điện thoại. Ngày nay hệ điều hành của điện thoại khá nhiều: Android, IOS, Windows Phone, …

Trong đó, số lượng người sử dụng hệ điều hành Android/IOS chiếm phần lớn thị trường trong và ngoài nước.

Vì vậy công cụ hiện này phù hợp với mô hình này nhất là: Xamarin trên Visual Studio.

Ngoài ra, để tạo lên sự thống nhất, tối ưu hóa giữa các nền tảng và sự dụng lại các nguồn tài nguyên đã được xây dựng từ trước (Models, Dll, Database) thì ta sẽ chọn một mô hình chung giữa chúng và mô hình phải dễ tiếp cận thuận lợi cho việc Code. Vì vậy mô hình MVVM là lựa chọn phù hợp.

* 1. Giới thiệu về công cụ được sử dụng

2.3.1 Visual Studio

2.3.1.1 Khái niệm Visual Studio

Visual studio là một trong những công cụ hỗ trợ lập trình và thiết kế website rất nổi tiếng nhất hiện nay của Mcrosoft và chưa có một phần mềm nào có thể thay thế được nó. Visual Studio được viết bằng 2 ngôn ngữ đó chính là C# và VB+. Đây là 2 ngôn ngữ lập trình giúp người dùng có thể lập trình được hệ thống một các dễ dàng và nhanh chóng nhất thông qua Visual Studio.

Visual Studio là một phần mềm lập trình hệ thống được sản xuất trực tiếp từ Microsoft. Từ khi ra đời đến nay, Visual Studio đã có rất nhiều các phiên bản sử dụng khác nhau. Điều đó, giúp cho người dùng có thể lựa chọn được phiên bản tương thích với dòng máy của mình cũng như cấu hình sử dụng phù hợp nhất.

Bên cạnh đó, Visual Studio còn cho phép người dùng có thể tự chọn lựa giao diện chính cho máy của mình tùy thuộc vào nhu cầu sử dụng.

2.3.1.2 Một số tính năng mềm Visual Studio

* Biên tập mã: Giống như bất kỳ một IDE khác, Visual Studio gồm có một trình soạn thảo mã hỗ trợ tô sáng cú pháp và hoàn thiện mả bằng các sử dụng IntelliSense không chỉ cho các hàm, biến và các phương pháp mà còn sử dụng cho các cấu trúc ngôn ngữ như: Truy vấn hoặc vòng điều khiển.
* Trình gỡ lỗi: Visual Studio có một trình gỡ lỗi có tính năng vừa lập trình gỡ lỗi cấp máy và gỡ lỗi cấp mã nguồn. Tính năng này hoạt động với cả hai mã quản lý giống như ngôn ngữ máy và có thể sử dụng để gỡ lỗi các ứng dụng được viết bằng các ngôn ngữ được hỗ trợ bởi Visual Studio.

2.3.1.3 Thiết kế

* Windows Forms Designer: Được sử dụng với mục đích xây dựng GUI sử dụng Windows Forms, được bố trí dùng để xây dựng các nút điều khiển bên trong hoặc cũng có thể khóa chúng vào bên cạnh mẫu. Điều khiển trình bày dữ liệu có thể được liên kết với các nguồn dữ liệu như: Cơ sở dữ liệu hoặc truy vấn.
* WPF Designer: Tính năng này cũng giống như Windows Forms Designer có công dụng hỗ trợ kéo và thả ẩn dụ. Sử dụng tương tác giữa người và máy tính nhắm mục tiêu vào Windows Presentation Foundation.
* Web Designer/ Development: Visual Studio cũng có một trình soạn thảo và thiết kế website cho phép các trang web được thiết kế theo tính năng kéo và thể đối tượng.
* Xamarin: Công cụ được thêm vào để xây dựng các ứng dụng di động đa nền tảng.

2.3.1.4 Điểm mạnh của Visual Studio

* Visual Studio hỗ trợ lập trình trên nhiều ngôn ngữ như: C/C++, C#, F#, Visual Basic, HTML, CSS, JavaScript.
* Là một công cụ hỗ trợ việc Debug một cách dễ dàng và mạnh mẽ như: Break Point, xem giá trị của biến trong quá trình chạy, hỗ trợ debug từng câu lệnh.
* Giao diện Visual Studio rất dễ sử dụng đối với người mới bắt đầu lập trình.
* Visual Studio hỗ trợ phát triển các ứng dụng: desktop MFC, Windows Form, Universal App, ứng dụng mobile Windows Phone 8/8.1, Windows 10, …
* Visual Studio hỗ trợ xây dựng ứng dụng một cách chuyên nghiệp bằng các công cụ kéo thả.

Visual Studio được đông đảo lập trình viên trên thế giới sử dụng.

2.3.2 Xamarin

2.3.2.1 Khái niểm về Xamarin

Xamarin là một framework để xây dựng các ứng dụng di động đa nền tảng được tạo ra bởi hãng phần mềm di động cùng tên thành lập vào ngày 16 tháng 5 năm 2011 bởi các kỹ sư đã tạo ra Mono, Mono cho Android và MonoTouch với hơn 500 nghìn nhân lực. Hãng này sử dụng ngôn ngữ C# để viết các ứng dụng chạy trên Mac, Android, ý tưởng triển khai các ứng dụng đa nền tảng sử dụng Common Language Infrastructure (CLI) và Common Language Specifications (thường được gọi là Microsoft .NET).

Đến tháng 11/2013, Microsoft đã mua lại và tích hợp chạy trên môi trường Windows 8. Với cơ sở mã nguồn mở của C#, các lập trình viên có thể sử dụng các công cụ Xamarin để viết các ứng dụng native Android, iOS và Windows với giao diện người dùng native và sự chia sẻ code trên nhiều nền tảng, bao gồm Windows và macOS.

Theo như Xamarin công bố, hơn 1.4 triệu lập trình viên đang sử dụng sản phẩm của Xamarin ở 120 quốc gia trên toàn thế giới tính đến tháng 4 năm 2017

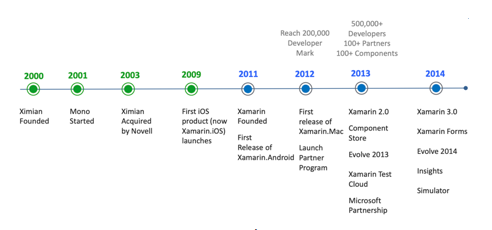


Figure 3: Quá trình phát triển của Xamarin

Công cụ Xamarin được Microsoft giới thiệu vào thị trường Việt Nam vào 18/04/2014 trong cho cộng đồng các giảng viên các trường CĐ, ĐH tại TPHCM và được chuyển giao các tài liệu, bản quyền với giá ưu đãi cho các sinh viên trên nền Microsoft Visual Studio và được chuyên gia Microsoft demo lập trình Xamarin để app chạy trên môi trường iOS, Android, Windows Phone và Windows 8.1 với các Tools khác nhau.

Tại sự kiện Microsoft Build 2016, Microsoft công báo họ sẽ mã nguồn mở Xamarin SDK và cung cấp các công cụ để phát triển ứng dụng sử dụng Xamarin miễn phí trên Visual Studio.

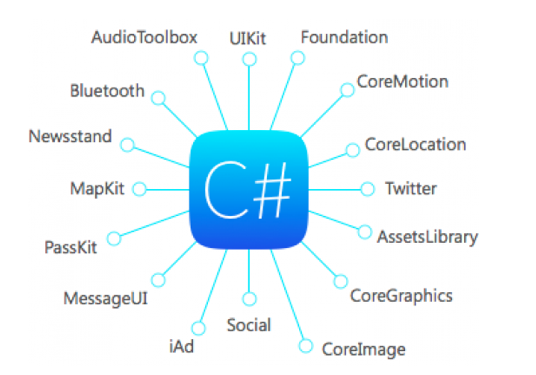


Figure 4: Điểm mạnh của Xamarin

2.3.2.2 Xamarin Forms

Xamarin.Forms là một công cụ giúp code giao diện người dùng có thể được chia sẻ trên các ứng dụng iOS, Android và Universal Windows Platform với 100% C#. Xamarin.Forms bao gồm hơn 40 điều khiển và bố cục, được ánh xạ tới các điều khiển gốc trong quá trình chạy. Nó hỗ trợ các nền tảng sau:

* Android 4.4 +
* IOS 8+
* Windows 10 Universal Windows Platform, Xamarin Forms còn hỗ trợ các nền tảng dưới đây nhưng vẫn đang ở chế độ Preview hoặc do bên thứ ba phát triển.
* Samsung Tizen
* macOS
* GTK#
* WPF

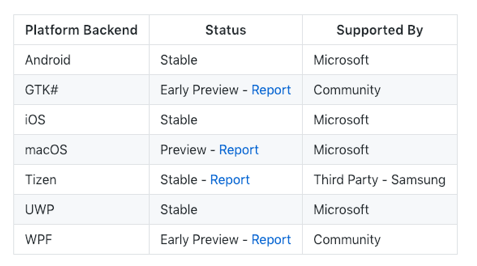


Figure 5: Danh sách các nền tảng hỗ trợ

Dưới đây là hình ảnh so sánh về chia sẻ code của cách sử dụng Xamarin truyền thống và sử dụng Xamarin.Forms. Rõ ràng khi sử dụng Xamarin.Forms, chúng ta đã sử dụng được hầu hết code chung cho toàn bộ ứng dụng.

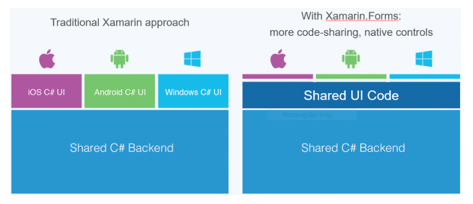


Figure 6: Lợi ích của Xamarin

Giao diện các ứng dụng sử dụng Xamarin được viết trong một file XAML. Nó có cấu trúc của một file XML, khá giống với Winform hay XML của Android.



Figure 7: Đoạn Code màn hình Login

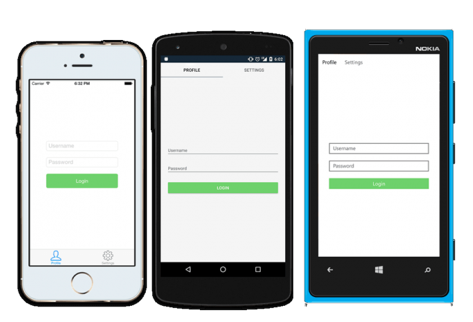


Figure 8: Build trên 3 nền tảng khác nhau

2.3.2.3 Ưu và nhược điểm

2.3.2.3.1 Ưu điểm

* Chia sẻ code ở mọi nơi: Khi viết một ứng dụng sử dụng bộ công cụ của Xamarin thì về cơ bản bạn đang sử dụng một lớp trừu tượng phía trên các SDK thực sự của iOS và Android. Điều này có nghĩa là bạn sẽ thu được kết quả là một ứng dụng native hoàn toàn với giao diện dùng native trên mỗi nền tảng. Khi bạn tạo ứng dụng trên Xamarin, bạn sử dụng cùng ngôn ngữ, API và cấu trúc dữ liệu để chia sẻ trung bình 75% code trên tất cả các nền tảng điện thoại di động. Logic ứng dụng này có thể dễ dàng chia sẻ trên nhiều nền tảng. Qua đó có thể giảm đáng kể chi phí và thời gian phát triển ứng dụng di động cho 3 nền tảng phổ biến nhất.

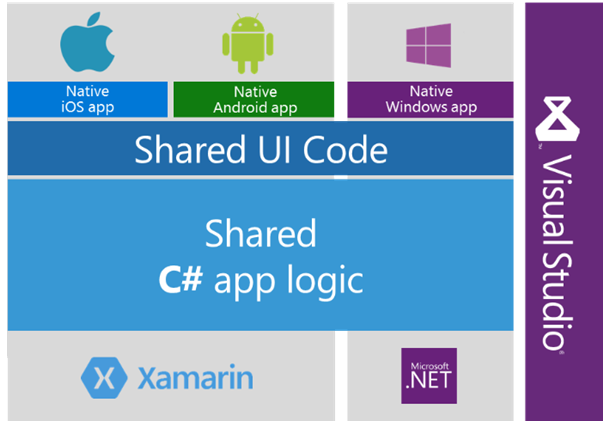
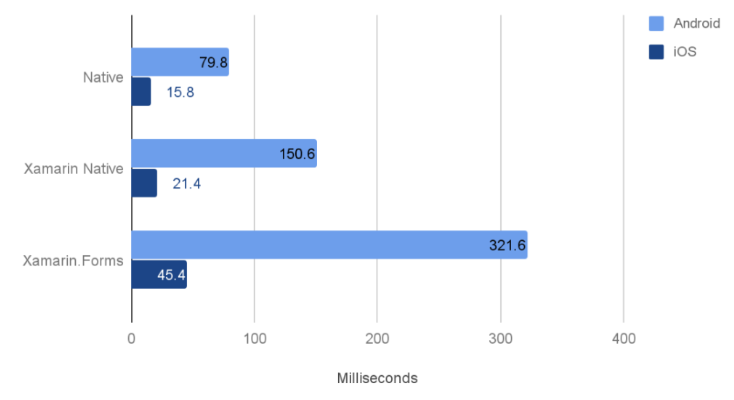


Figure 9: Chia sẻ code ở mọi nơi

* Performance như native: Không giống như phương pháp kết hợp truyền thống dựa trên các công nghệ web, một ứng dụng đa nền tảng được xây dựng với Xamarin cũng có thể xem vào hàng native. Các số liệu performances là tương đương khi so sánh với các số liệu performance của Java, Kotlin cho Android và Objective-C hoặc Swift cho ứng dụng phát triển ứng dụng iOS native. Hơn thế nữa, performance liên tục được cải thiện để phù hợp hoàn toàn với tiêu chuẩn của lập trình native. Nền tảng Xamarin cung cấp là giải pháp để testing và theo dõi hoạt động của ứng dụng. Xamarin Test Cloud kết hợp với công cụ Xamarin Test Recorder cho phép bạn chạy các UI test tự động và xác định các vấn đề về performance trước khi ứng dụng release. Tuy nhiên, dịch vụ này có tính phí.



*Hình 2.12: Tốc độ c hương trình "Hello World" Boot Speed*

* Một số ưu điểm khác như: Lựa chọn UI layout, Tích hợp OAuth, Tích hợp REST APIs từ xa, Công nghệ dẫn đường và xử lý tín hiệu thời gian thực cho ứng dụng định vị, Tích hợp mạng xã hội, Có bộ cơ sở dữ liệu SQLite nhúng, Thư viện XAML cho phép xây dựng một loạt ứng dụng, Hỗ trợ data binding.

2.3.2.3.2 Nhược điểm

* Ứng dụng thực hiện chậm hơn và yêu cầu nhiều dung lượng hơn trên thiết bị: Ứng dụng Xamarin lớn hơn, nặng hơn so với ứng dụng native. So sánh với ứng dụng native nó chiếm nhiều hơn vài Mb so với Java/Objective C tương ứng. Kích thước của một ứng dụng code bằng xamarin là 3Mb, trong khi code bằng Objective C chỉ chiếm 172 Kb. Càng sử dụng nhiều API, càng nhiều lưu trữ bị chiếm trên thiết bị.
* Khuyết điểm của AOT Compiler: Xamarin Forms cũng có những khuyết điểm AOT compliler. Nó không được complie các đoạn code gọn gàng như Xcode. Hạn chế của Xamarin ảnh hưởng đến cả phát triển, test tải app lên store. Hạn chế chính đối với iOS là nó không hỗ trợ sinh code tự động. Visual Studio biuld code lâu. Cái này tất cả các lập trình viên Xamarin đều đang phải sống chung với lũ, một lần build sẽ mất rất lâu từ 3 đến 10 phút cho solution với 30 project.
* Cộng đồng hỗ trợ ít: Cộng đồng Xamarin ít hơn so với cộng đồng của iOS hay Android nên để tìm kiếm được 1 developer Xamarin có kinh nghiệm là chuyện không dễ dàng gì dù Xamarin là nền tảng được phát triển nhờ sự hỗ trợ từ Microsoft. Theo nhiều nguồn thông tin, cộng đồng Xamarin chiếm 10% cộng động lập trình mobile toàn cầu dựa trên thực tế là nền tảng này cung cấp tối đa hỗ trợ cho các developer. Cụ thể, có một platform giáo dục chuyên biệt tại ĐH Xamarin, cung cấp kiến thức và khoá huấn luyện cho những ai mới tiếp cận công nghệ này. Kết quả là learning curve của 1 engineer C#/.Net kinh nghiệm sẽ được tối thiểu hóa.

2.3.3 Mô hình MVVM

2.3.3.1 Khái niệm

MVVM (Model-View-ViewModel) có thể nói là mô hình kiến trúc được rất nhiều lập trình viên trong cộng đồng lập trình ưa chuộng sử dụng nó. Mô hình này sinh ra dành cho các ứng dụng sử dụng ngôn ngữ XAML để định nghĩa giao diện ứng dụng như: Windows Phone 8.0, 8.1 Silverlight/RT, WPF, Silverlight, Windows RT, Universal Apps,… Lợi ích từ mô hình này mang lại là rất nhiều. Nhưng lợi ích nổi bật nhất là tách biệt việc thiết kế giao diện và lập trình code logic không phụ thuộc nhau.

MVVM không phải là framework hay thư viện, api… nó chỉ đơn thuần là hướng dẫn bạn định nghĩa cấu trúc ứng dụng của bạn. MVVM được phát triển dựa trên kiến trúc MVP.

Mô hình mvvm cho phép tách biệt dữ liệu (Model), mã thực thi (logic hay ViewModel) và giao diện người dùng (View).

Trong các mô hình truyền thống, chúng ta thường xử lý sự kiện Click và viết mã thực thi trực tiếp ở trên một Button nhưng với mô hình MVVM không cho phép làm điều này.

Trong mô hình MVVM, các điều khiển(control) như Button, ListView, SearchBar, v.v. không thể kết buộc trực tiếp đến dữ liệu mà phải thông qua thuộc tính Command – là một thuộc tính kiểu ICommand.



Figure 10: Mô hình MVVM

2.3.3.2 MVVM được hiểu như nào?

* View: Thành phần giao diện của ứng dụng. Tương tự như ở trong mô hình MVC, View là thành phần duy nhất mà người dùng có thể tương tác được trong chương trình, nó chính là thành phần mô tả dữ liệu.

Một điểm khác biệt so với các mô hình khác là View trong mô hình này tích cực hơn. Nó có khả năng thực hiện các hành vi và phản hồi lại người dùng thông qua tính năng là: binding, command.

* Model: Cũng tương tự như trong mô hình MVC. Model là các đối tượng giúp truy xuất và thao tác trên dữ liệu thực sự.
* ViewModel: Lớp trung gian giữa View và Model. ViewModel có thể được xem là thành phần thay thế cho Controller trong mô hình MVC. Nó chứa các mã lệnh cần thiết để thực hiện data binding, command.

Ngoài ra, để dễ hiểu về ViewModel thì nó sẽ đảm nhận công việc đồng bộ dữ liệu từ model lên View. Mối quan hệ giữa View và View-Model là View sẽ được ánh xạ tới View-Model nhưng View-Model lại không biết thông tin gì về View. Nó được ẩn dấu qua cách sử dụng Data-binding và cơ chế của mô hình Observer. Một View-Model có thể được ánh xạ từ nhiều View.

Lưu ý: Là trong mô hình MVVM, các tầng bên dưới sẽ không biết được các thông tin gì về tầng bên trên nó. ViewModel không hề biết gì về View, một ViewModel có thể được sử dụng cho nhiều View (one-to-many). ViewModel sử dụng Observer design pattern để liên lạc với View (thường được gọi là binding data, có thể là 1 chiều hoặc 2 chiều tùy nhu cầu của ứng dụng).

2.3.3.3 Cấu trúc MVVM trong Project

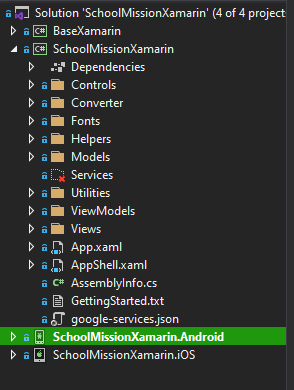


Figure 11: Mô hình MVVM

* Views: Trong thư mục Views chứ các file giao diện. Và mỗi file giao diện đều có class code-behind đi kèm. Đặc biệt file code-behind ta sẽ không sử dụng đến, mọi điều cần làm sẽ chuyển xuống class ViewModel. Tất nhiên là bạn có thể code trong file code-behind của XAML nhưng điều đó sẽ phá vỡ quy ước **MVVM.**Bạn có thể khai báo thuộc tính datacontext hoặc vài thiết lập khác nhưng nên hạn chế tối thiểu code ở đây. Views được sử dụng để kết hợp với các mô hình MVVM… Nó dùng để cung cấp một sự chia tách gọn gàng của khái niệm giữa UI và presentation logic và data.
* Models: Trong thư mục Models trong đó tạo các class chứa data và bất kỳ liên kết validation, logic nghiệp vụ để chắc chắc tính toàn vẹn của data, bạn có thể tách ra thư mục Repositories khác. Chúng được dùng như một phần của mô hình MVVM.
* ViewModels: Tương tự ta cũng tạo một thư mục ViewModels. Thông thường một file giao diện thì ta tạo một class ViewModels tương ứng (có đôi lúc ta tạo nhiều class phụ giúp tinh giản file code và gọi chúng trong class ViewModels chính). ViewModels sẽ sử dụng các model nếu cần định nghĩa dữ liệu. Sự liên kết giữa View-ViewModel giúp chúng gửi và nhận dữ liệu, để hiểu rõ ta cần tìm hiểu các khái niệm về Binding, DataContext, Behaviors SDK. Nhờ đó ta tách code-behind của View và đưa xuống View Model. Ngoài ra một lớp ViewModels chứa presentation logic và state của ứng dụng. ViewModels cần chứa các chức năng của ứng dụng. ViewModels định nghĩa properties, commands, và events, để chuyển đổi controls trong view cần data-bind.

2.3.3.4 Data Binding

Data Binding là kĩ thuật dùng để tạo gắn kết giữa phần giao diện (UI) và dữ liệu thông qua phần business logic. Nhờ Data Binding, UI có thể tự động cập nhật lại để hiển thị các thay đổi trong dữ liệu. Ngoài ra, Data Binding trong WPF còn hỗ trợ các chiều khác nhau, nghĩa là các thay đổi có thể cập nhật từ UI vào dữ liệu. Kĩ thuật binding trong mô hình mvvm thực sự là một bước tiến mới, thỏa mãn những điều mà hầu hết lập trình viên mong đợi.

Nếu như tìm hiểu về tính năng này, bạn sẽ không ngạc nhiên gì khi nhiều người nói rằng data binding là thành phần cốt lỗi tạo nên các cơ chế hoạt động trong WPF. Bạn có thể binding dữ liệu nguồn và đích từ bất kì đối tượng nào: như cửa sổ, các control đơn giản như TextBlock cho đến một usercontrol phức tạp.

Tất cả được thực hiện một cách dễ dàng, nhanh chóng, hiệu quả và có thể không cần dùng đến bất kì dòng code-behind (C#, VB.NET, …) nào.

2.3.3.5 Data Template

Data Template là kĩ thuật dùng để tạo ra một khuôn mẫu giao diện. Template chỉ được áp dụng cho các Control. Một template trong WPF xác định cách thức và cấu trúc mà dữ liệu hoặc control sẽ được hiển thị ra màn hình.

Nói riêng về Data Template, chức năng này giúp cho dữ liệu (thuộc dạng non-visual) được gắn vào một cấu trúc bao gồm một hoặc nhiều thành phần có khả năng hiển thị. Và do đó, dữ liệu sẽ được hiển thị lên cửa sổ một cách trực quan theo ý muốn của **lập trình viên**. Cũng như Data Binding, tính năng này không yêu cầu bạn phải biết trong code-behind của ứng dụng.

2.3.3.6 Command

Data Binding và Data Template trong wpf giúp cho người dùng thấy được những gì có trong dữ liệu và có thể cập nhật lại dữ liệu đó. Tuy nhiên để nhận được tương tác từ người dùng và xử lý, WPF cung cấp một tính năng gọi là command. Các command có thể được xem như dữ liệu và được cung cấp cho người dùng thông qua chức năng binding.

Một command binding cho phép bạn tùy ý xác định các phương thức xử lý, phím tắt hoặc thao tác chuột để kích hoạt.

2.3.3.7 Ưu điểm và nhược điểm của MVC, MVP và MVVM

2.3.3.7.1 Ưu nhược điểm của MVC

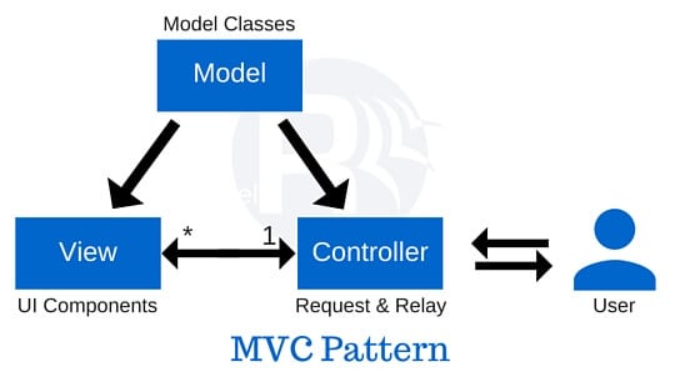


Figure 12: Mô hình MVC

Ưu điểm: Mô hình MVC có rất nhiều ưu điểm, cụ thể như là:

* Nhẹ, tiết kiệm băng thông: MVC không tiêu tốn nhiều viewstate nên rất tiết kiệm băng thông. Các thao tác gửi, nhận dữ liệu được diễn ra liên tục. Vì vậy, sử dụng mô hình này website/ ứng dụng hoạt động ổn định hơn.
* Có thể kiểm tra, phát hiện lỗi phần mềm một cách dễ dàng.
* Dễ dàng trong việc phân tách các phần Model và View.
* Mô hình này có kết cấu đơn giản. Không cần quá am hiểu về kỹ thuật cũng có thể sử dụng được.
* Hỗ trợ tốt cho các nền tảng phát triển SEO: Bạn có thể dễ dàng tạo ra các mã SEO URL để thu hút lượng truy cập đối với 1 ứng dụng bất kỳ.

Nhược điểm: Bên cạnh những ưu điểm nên bên trên thì MVC cũng tồn tại một số nhược điểm:

* Controller và View có sự liên quan với nhau. Vì vậy, khi thay đổi ở View thì đồng nghĩa bạn sẽ phải thay đổi ở Controller.
* Khó thực hiện chạy unit test do Controller và Android API có sự liên hệ chặt chẽ với nhau.
* Theo thời gian, Controller sẽ trở nên khó kiểm soát vì càng ngày càng có nhiều code được viết thêm vào.
* MVC phù hợp với các dự án lớn. Với các dự án nhỏ, mô hình này khá cồng kềnh và tốn nhiều thời gian trong việc trung chuyển dữ liệu.
* Làm khó khăn trong quá trình điều hướng code của dự án.

2.3.3.7.2 Ưu nhược điểm của MVP

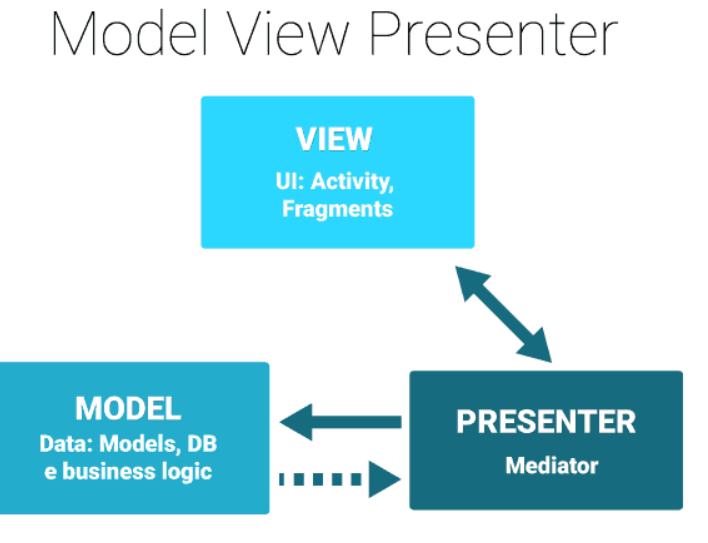


Figure 13: Mô hình MVP

Ưu điểm:

* Chúng ta dễ dàng viết unit test cho presenter vì nó không gắn với bất cứ view, nó hoạt động độc lập với View và không gắn với bất cứ API nào của Android.
* MVP có cấu trúc code rõ ràng hơn so với MVC nên khá dễ hiểu và dễ dùng. ít bug hơn, dễ dàng review code.

Nhược điểm:

* Mô hình MVP theo thời gian, Presenter sẽ dần lớn lên do bị thêm các business logic rải rác. Người dùng sẽ rất khó để kiểm soát và chia nhỏ code khi Presenter đã quá lớn.
* Nó sẽ trở lên rườm rà khi ta xây dựng với các ứng dụng nhỏ, hoặc với các Activity đơn giản.

Khó sử dụng lại logic code trong Presenter cho các View khác.

2.3.3.7.3 Ưu nhược điểm của MVVM

Ưu điểm:

* Thực hiện Unit testing bây giờ sẽ rất dễ dàng, vì bạn thực sự không phụ thuộc vào view.
* MVVM sẽ tạo sự tương tác hiệu quả giữa designer và developer.
* Tăng khả năng sử dụng lại các thành phần hay việc thay đổi giao diện chương trình mà không cần phải viết lại code quá nhiều.
* Phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp, bảo trì…

Nhược điểm :

* Khả năng duy trì khi view có thể gán cả biến và biểu thức, các logic không liên quan sẽ tăng dần theo thời gian, ảnh hưởng đến việc thêm code vào XML.
* Đối với dự án nhỏ việc áp dụng mô hình MVVM gây cồng kềnh, tốn thời gian trong quá trình phát triển. Tốn thời gian trung chuyển dữ liệu của các thành phần.
* Đối với dự án lớn hơn, nó gây khó khăn và mất thời gian để thiết kế các ViewModel.
* Việc liên kết dữ liệu cho tất cả các thành phần gây khó khăn trong việc debug khi cơ sở dữ liệu phức tạp.

2.3.3.7.4 Ứng dụng của mô hình MVC, MVP, MVVM

Bạn nên dùng MVP nếu không thể binding dữ liệu qua DataContext (ví dụ như với Windows Forms).

Mô hình MVVM thì nên dùng trong trường hợp có thể binding dữ liệu thông qua DataContext (ví dụ như: WPF).

Còn đối với MVC nên sử dụng khi việc kết nối giữa View và các phần còn lại của ứng dụng không phải lúc nào cùng available (ví dụ như web API). Khi đó, bạn không thể sử dụng MVP hoặc **MVVM** hiệu quả được và bắt buộc phải chọn mô hình MVC.

MVVM có thể nói là **mô hình kiến trúc** được rất nhiều lập trình viên ưa chuộng và sử dụng.  MVVM đã kế thừa những ưu điểm vốn có của MVP, kết hợp với những lợi thế của data binding đem đến một pattern có khả năng phân chia các thành phần với từng chức năng riêng biệt, dễ dàng trong việc maintain, redesign. MVVM cũng đem lại khả năng test rất dễ dàng, giúp làm việc hiệu quả hơn.

2.3.4 NoSQL

2.3.4.1 Khái niệm

Cơ sở dữ liệu NoSQL là Cơ sở dữ liệu được xây dựng dành riêng cho mô hình dữ liệu và có sơ đồ linh hoạt để xây dựng các ứng dụng hiện đại. Cơ sở dữ liệu NoSQL được công nhận rộng rãi vì khả năng dễ phát triển, chức năng cũng như hiệu năng ở quy mô lớn. Trang này có các tài nguyên giúp bạn hiểu thêm về cơ sở dữ liệu NoSQL và bắt đầu sử dụng.

2.3.4.2 Hoạt động

Cơ sở dữ liệu NoSQL sử dụng nhiều mô hình dữ liệu để truy cập và quản lý dữ liệu. Các loại cơ sở dữ liệu này được tối ưu hóa dành riêng cho các ứng dụng yêu cầu mô hình dữ liệu linh hoạt có lượng dữ liệu lớn và độ trễ thấp, có thể đạt được bằng cách giảm bớt một số hạn chế về tính nhất quán của dữ liệu của các cơ sở dữ liệu khác.

Hãy xem ví dụ về mô hình sơ đồ dành cho cơ sở dữ liệu sách đơn giản:

* Trong cơ sở dữ liệu quan hệ, hồ sơ về một cuốn sách thường được phân tách (hay còn gọi là "chuẩn hóa") và lưu trữ trong các bảng tách biệt nhau, còn mối quan hệ được quy định bằng các ràng buộc khóa ngoại và khóa chính. Trong ví dụ này, bảng Sách có các cột cho ISBN, Tên sách và Số phiên bản, bảng Tác giả có các cột cho ID tác giả và Tên tác giả và cuối cùng, bảng Tác giả–ISBN có các cột cho ID tác giả và ISBN. Mô hình quan hệ được thiết kế để cho phép các cơ sở dữ liệu này thực thi tính toàn vẹn tham chiếu giữa nhiều bảng trong cơ sở dữ liệu, được chuẩn hóa để giảm dư thừa và thường được tối ưu hóa cho mục đích lưu trữ.
* Trong cơ sở dữ liệu NoSQL, hồ sơ về một cuốn sách thường được lưu trữ dưới dạng văn bản JSON. Với từng quyển sách, mục, ISBN, Tên sách, Số phiên bản, Tên tác giả và ID tác giả được lưu trữ dưới dạng thuộc tính trong một văn bản duy nhất. Trong mô hình này, dữ liệu được tối ưu hóa cho việc phát triển trực quan và khả năng thay đổi quy mô theo chiều ngang.

2.3.4.3 Ưu điểm

* Linh hoạt: Cơ sở dữ liệu NoSQL thường cung cấp các sơ đồ linh hoạt giúp công đoạn phát triển nhanh hơn và có khả năng lặp lại cao hơn. Mô hình dữ liệu linh hoạt biến cơ sở dữ liệu NoSQL thành lựa chọn lý tưởng cho dữ liệu không được tổ chức thành cấu trúc hoặc có cấu trúc chưa hoàn chỉnh.
* Khả năng thay đổi quy mô: Cơ sở dữ liệu NoSQL thường được thiết kế để tăng quy mô bằng cách sử dụng các cụm phần cứng được phân phối thay vì tăng quy mô bằng cách bổ sung máy chủ mạnh và tốn kém. Một số nhà cung cấp dịch vụ đám mây xử lý các hoạt động này một cách không công khai dưới dạng dịch vụ được quản lý đầy đủ.
* Hiệu năng cao: Cơ sở dữ liệu NoSQL được tối ưu hóa theo các mô hình dữ liệu cụ thể và các mẫu truy cập giúp tăng hiệu năng cao hơn so với việc cố gắng đạt được mức độ chức năng tương tự bằng cơ sở dữ liệu quan hệ.
* Cực kỳ thiết thực: Cơ sở dữ liệu NoSQL cung cấp các API và kiểu dữ liệu cực kỳ thiết thực được xây dựng riêng cho từng mô hình dữ liệu tương ứng.

2.3.4.4 Sự khác biệt giữa SQL và NoSQL

|  |  |  |
| --- | --- | --- |
| Đặc điểm | SQL | NoSQL |
| Khối lượng công việc tối ưu | Cơ sở dữ liệu quan hệ được thiết kế dành cho các ứng dụng xử lý giao dịch trực tuyến (OLTP) trong giao dịch có độ ổn định cao và thích hợp để xử lí phân tích trực tuyến (OLAP). | Các cơ sở dữ liệu NoSQL được thiết kế cho các mẫu truy cập dữ liệu, bao gồm các ứng dụng có độ trễ thấp. Cơ sở dữ liệu tìm kiếm NoSQL được thiết kế để phục vụ phân tích dữ liệu có cấu trúc chưa hoàn chỉnh. |
| Mô hình dữ liệu | Mô hình quan hệ chuẩn hóa dữ liệu vào bảng được hình thành từ hàng và cột. Sơ đồ quy định rõ ràng bảng, hàng, cột, chỉ mục, mối quan hệ giữa các bảng và các thành tố cơ sở dữ liệu khác. Cơ sở dữ liệu sẽ thực thi tính toàn vẹn tham chiếu trong mối quan hệ giữa các bảng. | Các cơ sở dữ liệu NoSQL cung cấp nhiều mô hình dữ liệu khác nhau như khóa-giá trị, tài liệu và biểu đồ, được tối ưu hóa để đạt hiệu năng và quy mô tối ưu. |
| Thuộc tính ACID | Cơ sở dữ liệu quan hệ có các thuộc tính mang tính nguyên tố, nhất quán, tách biệt và bền vững (ACID):   * Tính nguyên tố đòi hỏi giao dịch phải được thực thi đầy đủ hoặc hoàn toàn không thực hiện. * Tính nhất quán đòi hỏi rằng khi giao dịch được thực hiện, dữ liệu phải nhất quán với sơ đồ cơ sở dữ liệu. * Tính tách biệt đòi hỏi rằng các giao dịch diễn ra đồng thời phải được thực thi tách biệt với nhau. * Tính bền vững đòi hỏi phải có khả năng phục hồi từ tình trạng hư hỏng hệ thống hoặc mất điện đột ngột về trạng thái đã biết cuối cùng. | Cơ sở dữ liệu NoSQL thường phải đánh đổi bằng cách nới lỏng một số thuộc tính ACID này của cơ sở dữ liệu quan hệ để có mô hình dữ liệu linh hoạt hơn có khả năng thay đổi quy mô theo chiều ngang. Việc này biến các cơ sở dữ liệu NoSQL thành lựa chọn tuyệt vời cho các trường hợp sử dụng cần thông lượng cao, độ trễ thấp cần thay đổi quy mô theo chiều ngang vượt qua giới hạn của một phiên bản duy nhất. |
| Hiệu năng | Hiệu năng thường phụ thuộc vào hệ thống con của ổ đĩa. Thông thường, việc tối ưu hóa các truy vấn, chỉ mục và cấu trúc bảng bắt buộc phải được thực hiện để đạt mức hiệu năng tối đa. | Hiệu năng thường được xem là chức năng của kích cỡ cụm phần cứng ngầm, độ trễ mạng và ứng dụng đưa ra lệnh gọi. |
| Quy mô | Cơ sở dữ liệu quan hệ thường tăng quy mô bằng cách tăng năng lực điện toán của phần cứng hoặc tăng quy mô bằng cách thêm bản sao của khối lượng công việc chỉ đọc. | Cơ sở dữ liệu NoSQL thường có tính phân mảnh cao do các mẫu truy cập khóa-giá trị có khả năng tăng quy mô bằng cách sử dụng kiến trúc được phân phối để tăng thông lượng, đem đến hiệu năng ổn định với quy mô gần như không giới hạn. |
| API | Yêu cầu lưu trữ và truy xuất dữ liệu được truyền đạt bằng cách sử dụng các truy vấn nhất quán với ngôn ngữ truy vấn có cấu trúc (SQL). Các truy vấn này được phân tích và thực thi bởi cơ sở dữ liệu quan hệ. | API trên cơ sở đối tượng cho phép các nhà phát triển ứng dụng dễ dàng lưu trữ và truy xuất cấu trúc dữ liệu trong bộ nhớ. Khóa phân mảnh tìm kiếm các cặp khóa-giá trị, tập hợp cột hoặc văn bản có cấu trúc chưa hoàn chỉnh có chứa đối tượng và thuộc tính của ứng dụng được xếp theo chuỗi. |

2.3.5 Firebase

2.3.5.1 Khái niệm

Firebase là một nền tảng để phát triển ứng dụng di động và trang web, bao gồm các API đơn giản và mạnh mẽ mà không cần backend hay server.

Firebase là dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây – cloud. Kèm theo đó là hệ thống máy chủ cực kỳ mạnh mẽ của Google. Chức năng chính là giúp người dùng lập trình ứng dụng bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu.

Cụ thể là những giao diện lập trình ứng dụng API đơn giản. Mục đích nhằm tăng số lượng người dùng và thu lại nhiều lợi nhuận hơn.

Đặc biệt, còn là dịch vụ đa năng và bảo mật cực tốt. Firebase hỗ trợ cả hai nền tảng Android và IOS. Không có gì khó hiểu khi nhiều lập trình viên chọn Firebase làm nền tảng đầu tiên để xây dựng ứng dụng cho hàng triệu người dùng trên toàn thế giới.



Figure 14: Firebase

2.3.5.2 Cách thức hoạt động firebase

2.3.5.2.1 Firebase Realtime Database là gì?

Khi đăng ký một tài khoản trên Firebase để tạo ứng dụng, bạn đã có một cơ sở dữ liệu thời gian thực. Dữ liệu bạn nhận được dưới dạng JSON. Đồng thời nó cũng luôn được đồng bộ thời gian thực đến mọi kết nối client.

Đối với các ứng dụng đa nền tảng, tất cả các client đều sử dụng cùng một cơ sở dữ liệu. Nó được tự động cập nhật dữ liệu mới nhất bất cứ khi nào các lập trình viên phát triển ứng dụng. Cuối cùng, tất cả các dữ liệu này được truyền qua kết nối an toàn SSL có bảo mật với chứng nhận 2048 bit.

Trong trường hợp bị mất mạng, dữ liệu được lưu lại ở local. Vì thế khi có mọi sự thay đổi nào đều được tự động cập nhật lên Server của Firebase. Bên cạnh đó, đối với các dữ liệu ở local cũ hơn với Server thì cũng tự động cập nhật để được dữ liệu mới nhất.

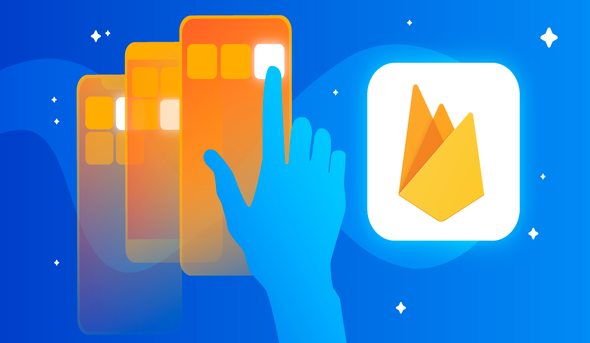


Figure 15: Realtime database

2.3.4.2.2 Freebase Authentication là gì?



Figure 16: Firebase authentication

Hoạt động nổi bật của Firebase là xây dựng các bước xác thực người dùng bằng Email, Facebook, Twitter, GitHub, Google. Đồng thời cũng xác thực nặc danh cho các ứng dụng. Hoạt động xác thực có thể giúp thông tin cá nhân của người sử dụng được an toàn và đảm bảo không bị đánh cắp tài khoản.

2.3.4.2.3 Firebase Hosting là gì?



Figure 17: Firebase hosting

Cách thức hoạt động cuối cùng của Firebase được đề cập trong bài viết này là cung cấp các hosting. Hosting được phân phối qua tiêu chuẩn công nghệ bảo mật SSL từ mạng CDN. Vậy CDN là gì?

CDN viết tắt của Content Delivery Network là mạng lưới máy chủ lưu giữ bản sao của các nội dung tĩnh bên trong website và phân phối đến nhiều máy chủ PoP. Mạng lưới máy chủ CDN được đặt ở khắp mọi nơi trên toàn cầu. Từ PoP (Points of Presence), dữ liệu sẽ tiếp tục được gửi đến người dùng cuối. Thông qua CDN, bản sao nội dung trên máy chủ gần nhất sẽ được trả về cho người dùng khi họ truy cập website.

* + - 1. Ưu điểm của Firebase
* Tạo tài khoản và sử dụng dễ dàng: Firebase cho phép người dùng đăng nhập bằng tài khoản Google đơn giản. Gói Spark của Firebase miễn phí và cung cấp nhiều tính năng để giúp các nhà phát triển bắt đầu sử dụng. Khi yêu cầu ngày càng tăng, tôi chọn gói Blaze có trả phí với nhiều tính năng nâng cao hơn.
* Tốc độ phát triển nhanh: Firebase là một tùy chọn phát triển ứng dụng phù hợp có thể giúp các nhà phát triển tiết kiệm thời gian và giảm thời gian tiếp thị ứng dụng. Thông thường, mọi Developer cần có quyền truy cập vào Server và Host để tạo, bảo trì cơ sở dữ liệu và dịch vụ phụ trợ. Do đó, cần có một Backend Developer và một Frontend Developer để xây dựng các ứng dụng. Tuy nhiên, việc này thường có thể dẫn đến lỗi và các vấn đề có thể gây ra sự cố ứng dụng và làm tăng chi phí phát triển. Thông qua việc sử dụng Firebase và Firestore, Frontend Developer có thể quản lý, giảm thời gian cần thiết để hoàn thành tất cả công việc.

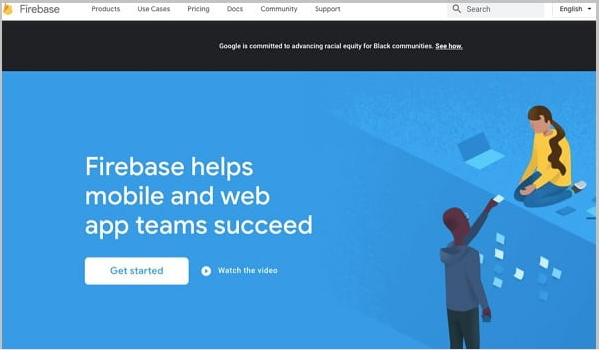


Figure 18: Website firebase

* Nhiều dịch vụ trong một nền tảng: Firebase cũng cung cấp danh sách đầy đủ các sản phẩm để hỗ trợ các Developer trong quá trình phát triển. Hai tùy chọn cơ sở dữ liệu là Firestore và Realtime Database của Firebase. Tương tự như vậy, Firebase cho phép bạn thực hiện lưu trữ Cloud Media dễ dàng. Nó cũng cho phép phát triển ứng dụng không cần máy chủ thông qua việc tích hợp Cloud Functions. Firebase bao gồm toàn bộ chu trình phát triển ứng dụng. Nền tảng này chứa các tính năng để xây dựng, phát hành và giám sát các ứng dụng. Ngoài ra, là bước cuối cùng của chu trình phát triển ứng dụng, nó cung cấp các công cụ để thu hút người dùng và giữ họ sử dụng nó. Đây cũng là ưu điểm của Firebase mà tôi đánh giá khá cao.
* Được cung cấp bởi Google: Firebase được cung cấp bởi Google, một trong những tên tuổi nổi bật và đáng tin cậy nhất trong thế giới công nghệ. Theo tôi được biết, kể từ khi được mua lại, Firebase đã trải qua một loạt thay đổi, phát triển và trở thành nền tảng đáng tin cậy như ngày nay. Nó khai thác sức mạnh của Google Cloud và nhiều dịch vụ của Google. Firebase hiện là một phần của Google Cloud Platform. Nó hoạt động tốt với các dịch vụ Google Cloud khác và tích hợp với nhiều dịch vụ của bên thứ ba.

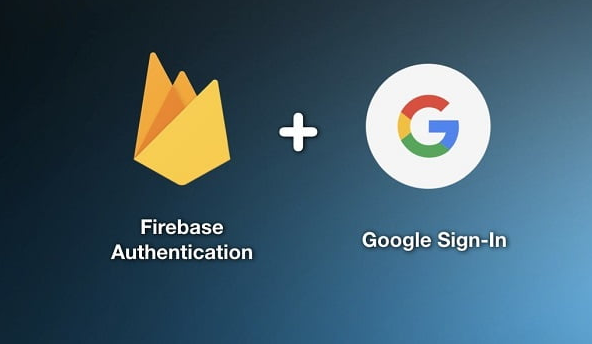


Figure 19: Firebase auth + google sign in

* Tập trung vào phát triển giao diện người dung: Một ưu điểm khác khiến tôi yêu thích Firebase đó là nó cho phép Developer tập trung vào việc tạo mã Frontend cho các ứng dụng di động. Nền tảng này giúp cho việc phát triển ứng dụng trở nên thuận tiện và giúp giảm chi phí đáng kể. Sử dụng Firebase cũng cho phép các công ty, Developer chuẩn hóa môi trường Backend theo một công nghệ duy nhất và dễ học. Mẫu Backend làm giảm số lượng đào tạo cần thiết để hỗ trợ nó và cho phép Developer tập trung vào phát triển giao diện người dùng.
* Firebase không có máy chủ: Tôi cho rằng việc mở rộng hoặc giảm quy mô máy chủ không phải là một nhiệm vụ dễ dàng. Đặc biệt, việc mở rộng một cụm cơ sở dữ liệu là một thách thức và việc tối ưu hóa hiệu suất cho khối lượng công việc khổng lồ đòi hỏi các chuyên gia có kinh nghiệm. Firebase giải quyết vấn đề này và cung cấp một môi trường hoàn toàn không có máy chủ. Firebase đi kèm với kiến ​​trúc không máy chủ (Serverless Architecture). Do đó, bạn sẽ không cần lo lắng về cơ sở hạ tầng máy chủ.
* Học máy (Machine Learning): Theo Gartner, 30% doanh nghiệp sẽ sử dụng Machine Learning (ML) trong một phần quy trình của họ. Firebase cũng có lợi vì nó cung cấp cho các nhà phát triển tùy chọn để dựa vào Machine Learning. Firebase đi kèm với bộ ML với các API sẵn có cho các tính năng khác nhau của nền tảng di động như nhận dạng văn bản, nhận diện khuôn mặt, ghi nhãn hình ảnh, quét mã vạch,…



Figure 20: Firebase ML

* Tạo lưu lượng truy cập: Firebase hỗ trợ lập chỉ mục ứng dụng để cho phép người dùng thu hút lại những khách tìm kiếm trên Google. Cụ thể, nó cung cấp liên kết ứng dụng trên mục tìm kiếm của Google. Xếp hạng ứng dụng cũng có thể được nâng cao trên Tìm kiếm một lần bằng cách lập chỉ mục ứng dụng. Điều này giúp ứng dụng của bạn tiếp xúc với những người dùng mới có thể cài đặt ứng dụng đó.
* Theo dõi lỗi: Tiếp theo, tôi muốn giới thiệu với bạn tính năng Crashlytics của Firebase. Đây là một công cụ tuyệt vời để tìm và khắc phục sự cố một cách nhanh chóng. Firebase có thể theo dõi cả lỗi không nghiêm trọng và lỗi nghiêm trọng. Đồng thời, báo cáo được tạo dựa trên mức độ ảnh hưởng của lỗi đến trải nghiệm của người dùng.
* Sao lưu: Firebase đảm bảo tính bảo mật tối ưu và tính sẵn có của dữ liệu nhờ các bản sao lưu thường xuyên. Các ứng dụng được bảo vệ khỏi mọi khả năng mất dữ liệu bằng cách dựa vào tính năng sao lưu tự động của nền tảng này. Khi sử dụng gói Blaze, tôi cũng có thể dễ dàng định cấu hình Cơ sở dữ liệu thời gian thực của Firebase để thực hiện các bản sao lưu tự động.
  + - 1. Hạn chế của Firebase
* Không phải là mã nguồn mở: Firebase không phải là một tùy chọn mã nguồn mở để phát triển ứng dụng di động. Điều này khiến nó trở thành một lựa chọn không tối ưu cho nhiều nhà phát triển. Người dùng không thể sửa đổi mã nguồn Firebase. Theo tôi thì đây là hạn chế lớn nhất của Firebase và ngăn cộng đồng cải tiến sản phẩm.



Figure 21: Opensource firebase

* Người dùng không có quyền truy cập mã nguồn: Đây là một hạn chế lớn khác của Firebase cho người dùng. Việc không có quyền truy cập vào mã nguồn có thể khá khó khăn đối với một số nhà phát triển. Đặc biệt, đối với các ứng dụng lớn, việc chuyển sang các nhà cung cấp khác không phải là một nhiệm vụ dễ dàng và để thực hiện việc này, toàn bộ Backend sẽ cần được xây dựng lại từ đầu.
* Firebase không hoạt động ở nhiều quốc gia: Firebase là một Subdomain của Google. trang Web chính thức của nó là https://firebase.google.com và bị chặn ở nhiều quốc gia. Theo tôi được biết, Google cùng với các dịch vụ khác của Google bị chặn và không thể truy cập được ở nhiều quốc gia, trong đó có Trung Quốc.
* Chỉ hoạt động với Cơ sở dữ liệu NoSQL: Cả Firestore và Firebase Realtime Database đều cung cấp cấu trúc NoSQL. Chúng không có tùy chọn nào để sử dụng cơ sở dữ liệu quan hệ. Mặc dù Firebase đã thực hiện những cải tiến đáng kể trên Cloud Firestore so với RTDB, nhưng việc chạy các truy vấn phức tạp vẫn là một thách thức đối với một số người dùng nhất định. Với Firebase, người dùng không thể xử lý nhanh chóng việc di chuyển dữ liệu tương tự như cơ sở dữ liệu SQL đơn giản. Firebase sử dụng JSON và hầu như không có tính năng SQL nào. Vì vậy, việc di chuyển từ cơ sở dữ liệu sẽ không dễ dàng.
* Truy vấn chậm: Không phải tất cả cơ sở dữ liệu đều lý tưởng cho mọi trường hợp sử dụng và Cloud Firestore cũng không ngoại lệ. Mặc dù đó là một giải pháp tuyệt vời để mở rộng ứng dụng của bạn, cung cấp hỗ trợ truy vấn ngoại tuyến và cập nhật thời gian thực, nhưng nó cũng có những hạn chế như:
  + - Kích thước yêu cầu API tối đa 10 MiB.
    - Không có truy vấn tổng hợp gốc.
    - Giới hạn kích thước tài liệu là 1 MiB.
    - Kết nối đồng thời tối đa 1M.
    - Truy vấn hiệu suất chậm.
* Không phải tất cả các dịch vụ Firebase đều miễn phí: Các chức năng đám mây chỉ khả dụng trên gói Blaze và người dùng không thể dùng thử dịch vụ trong gói Spark. API Cloud Vision trong tính năng học máy cũng không khả dụng trong gói Spark miễn phí của nền tảng này.. Dịch vụ này chỉ có sẵn cho người dùng của gói Blaze và tính phí 1,5 USD/ k API.
* Firebase khá đắt và giá không ổn định: Firebase là một nền tảng phát triển ứng dụng di động có nhiều tính năng nhưng cũng là một nền tảng đắt tiền đối với một số người. Nó cung cấp một gói miễn phí nhưng có hạn chế nhất định. Firebase cung cấp nhiều tính năng hữu ích hơn nhưng giá cũng sẽ cao hơn so với một nhà cung cấp IaaS thuần túy. Nói chung, tôi cho rằng việc tự lưu trữ một ứng dụng sẽ rẻ hơn so với sử dụng Firebase.

Một khía cạnh khác khiến Firebase trở nên đắt đỏ là nó là một công nghệ độc quyền. Firebase phải bù đắp khoản đầu tư kỹ thuật cho cơ sở người dùng của nó.

Firebase cung cấp dịch vụ theo mô hình định giá dựa trên mức sử dụng và không có cách nào để giới hạn giá cả. Không có bảng giá cố định cho nền tảng và việc theo dõi mức độ sử dụng tài nguyên có thể là một vấn đề phức tạp.

Việc ước tính chi phí của Firebase có thể khó khăn đối với một số người dùng, đặc biệt là khi không thể dự đoán được các yêu cầu về khả năng mở rộng. Việc định giá rất phức tạp và Firebase có giá cụ thể cho mọi tính năng.

Theo thông tin tôi được biết, Firebase từng cung cấp gói Flame. Đó là gói có giá cố định 25 USD / tháng. Nhưng nó đã được gỡ bỏ vào tháng 1/ 2020.

* Chỉ chạy trên Google Cloud: Firebase hiện là một phần của Google và cơ sở hạ tầng của nó chạy hoàn toàn trên Google Cloud. Bạn sẽ không có tùy chọn để chạy Firebase trên các nhà cung cấp đám mây khác như AWS, Azure hoặc Digital Ocean.



Figure 22: Firebase + google cloud

* Thiếu Dedicated Servers và hợp đồng doanh nghiệp: Firebase không có tùy chọn Dedicated Servers hoặc hợp đồng doanh nghiệp. Cách duy nhất để sử dụng Firebase là sử dụng cấu trúc Serverless ít tính linh hoạt hơn. Một hạn chế mà tôi thấy khá bất tiện khi dùng Firebase là việc không cung cấp các gói định giá, hợp đồng hoặc hỗ trợ doanh nghiệp. Tuy nhiên, gói Blaze có thể được coi là một phương án chấp nhận được. Nhưng tôi vẫn hy vọng có thêm nhiều lựa chọn và sự hỗ trợ hơn.
* Không cung cấp các API GraphQL: Firebase không cung cấp API GraphQL như một phần của thiết lập tiêu chuẩn. Mặc dù có những giải pháp thay thế cho việc triển khai GraphQL với Firebase, REST vẫn là tùy chọn mặc định của nền tảng.

2.4 Kết luận

Với hướng giải quyết như trên đã lựa chọn được những công cụ lập trình và dữ liệu hỗ trợ cho quá trình xây dựng ứng dụng app ILearn. Do kiến thức còn chưa vững nên nhưng lý thuyết trên trình bày còn chưa được hoàn thiện và có sự tham khảo trên mạng, sách vở, ...

Chương tiếp ta sẽ tiến hành xây dựng App ILearn trên nền tảng Xamarin với mô hình MVVM.

**CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG LANDSLIDE TRÊN XAMARIN**

Chương 3 sẽ tiến hành thực hiện mô hình MVVM trên Xamarin để tạo nên một khuôn mẫu tổng thể. Tạo tiền đề cho việc phát triển ứng dụng ILearn trên hai nền tảng công nghệ

3.1 Mô hình ứng dụng

Xây dựng một ứng dụng học và làm việc từ xa cần có một nơi lưu giữ liệu trung gian cho mọi người dung và thực hiện các truy vấn cần thiết cho từng request của người dung.

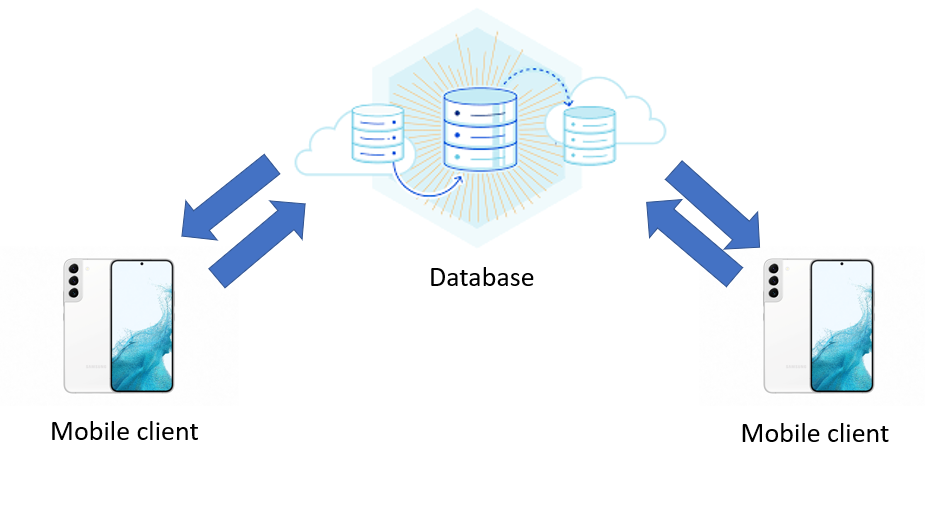


Figure 23: Mô hình ứng dụng

Ứng dụng được tổ chức dưới dạng Database – Client. Điện thoại cài app kết nối Internet sẽ gửi Request lên Database (NoSQL), Database sẽ thực hiện truy vấn và trả lại kết quả cho thiết bị và hiển thị cho người dung. Ứng dụng có thể cung cấp cho rất nhiều người dung truy cập và tiến hành học và làm việc theo nhóm từ xa.

Ứng dụng này xây dựng trên điện thoại môi trường Android 4.0 (Ice Cream Sandwich) trở lên, IOS 8 trở lên. Ứng dụng gồm có chức năng sau:

* Đăng nhập: cho phép người dung truy cập và ứng dụng để tiến hành thống kê các vụ sạt lở đất từ xa.
* Thông tin account: người dung có thể xem được thông tin người dung, avanta.
* Đăng xuất: Thoát tài khoản và chọn các tài khoản đã đăng nhập
* Google map: thống kê tất cả các vụ sạt lở đất trên google map và có thể hiển thị cho người dùng xem các khu vực xung quanh đó.

3.2 Phân tích hệ thống

3.2.1 Usercase của hệ thống

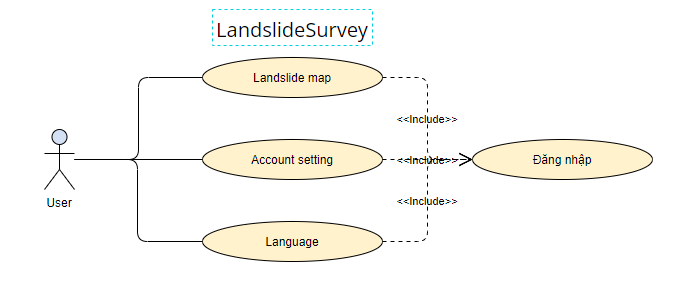


Figure 24: Usercase ứng dụng Landslide Survey

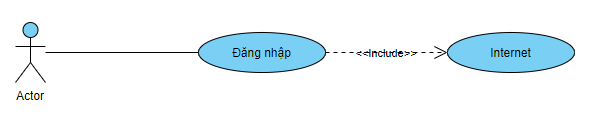
Tác nhân tham gia hệ thống:

* Người dùng: tất cả người dùng sử dụng ứng dụng đều có thể sử dụng tất cả các chức năng của hệ thống

Các tính năng của App:

* Đăng nhập, đăng xuất: cho phép người dung đăng nhập, thoát khỏi tài khoản.
* Landslide map: người dùng có thể xem được tất cả các địa điểm xảy ra các loại sạt lở đất ở nhiều khu vực xảy ra hay ở xung quanh vị trí của mình. Ngoài ra, người dùng có thể thêm các địa điểm xảy ra sạt lở đất để cảnh báo cho người sử dụng khác hay cho chính mình.
* Ngôn ngữ: ứng dụng sẽ thay đổi ngôn ngữ dựa theo sự lựa chọn của người dùng. Trong ứng dụng này tôi tích hợp 2 loại ngôn ngữ là tiếng anh và tiếng việt.
* Thông tin tài khoản: sẽ hiển thị đầy đủ thông tin của người dùng, họ cũng có thể sửa thông tin của chính mình.

3.2.2 User đăng nhập



Mô tả: để đăng nhập vào ứng dụng, người dung bắt buộc phải có kết nối internet để có thể tiến hành đăng nhập hay các thao tác khác trong ứng dụng. Ngoài ra, cũng có thể lấy lại mật khẩu hay là tạo tài khoản đăng nhập.

3.2.3 Usercase Landslide map

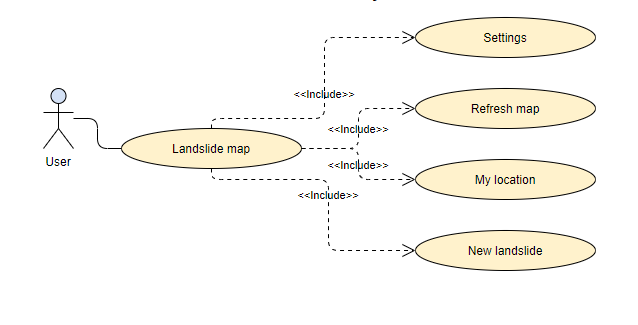


Figure 25: Usercase landslide map

Mô tả: google map sẽ được hiển thị tất cả các vị trí xảy ra sạt lở đất với các thông tin được hiển thị cụ thể. Ngoài ra, còn có thể làm mới lại map để cập nhật các điểm mới. Map cũng lấy được vị trí của chính mình để xác định xem liệu khu vực xung quanh có sạt lở đất hay không. Người dùng có thể chụp ảnh lại vị trí sạt lở đất.

3.2.4 User case cấu hình

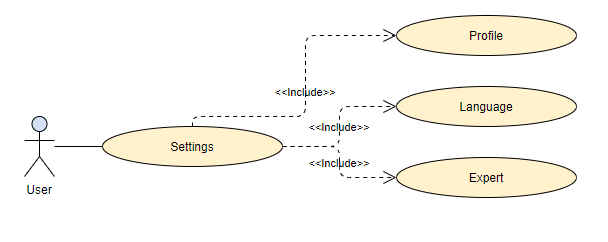


Figure 26: Usercase cấu hình

Mô tả: người dùng có thể thay đổi thông tin tài khoản của mình. Ngoài ra, để phù hợp với nhiều người dùng ứng dụng cũng cho thay đổi ngôn ngữ để thuận tiện sử dụng. Chế độ Expert cho phép người dùng có thể xem được nhiều địa điểm xảy ra sạt lở đất và chỉnh sửa.

3.3 Hoạt động hệ thống

3.3.1 Chức năng đăng nhập, đăng ký

Mô tả biểu đồ: để sử dụng người dùng sẽ phải đăng ký tài khoản. Người dùng sẽ phải chấp nhận các điều khoản sử dụng ứng dụng sau đó điền các thông tin cơ bản về mình để tiến hành tạo tài khoản.

Khi nhập Username và Password phía hệ thống sẽ kiểm tra có tài khoản này không, nếu không tồn tài sẽ có một thông báo “User invalid “, nếu có hệ thống tiếp tục kiểm tra mật khẩu, nếu không đúng sẽ báo “Sai mật khẩu” và yêu cầu nhập lại, khi mật khẩu đúng thì người dùng đăng nhập thành công.

Ngoài ra, khi người dung quên mật khẩu họ có thể chọn “quên mật khẩu” và nhập email của họ để xác nhận và 1 email reset password sẽ được gửi vào email của họ để có thể nhập mật khẩu mới.

App sẽ tự động lưu email người dung nhập vào.

Với chức năng đăng ký người dung sẽ tiến hành nhập các thông tin cơ bản để đăng ký được tài khoản

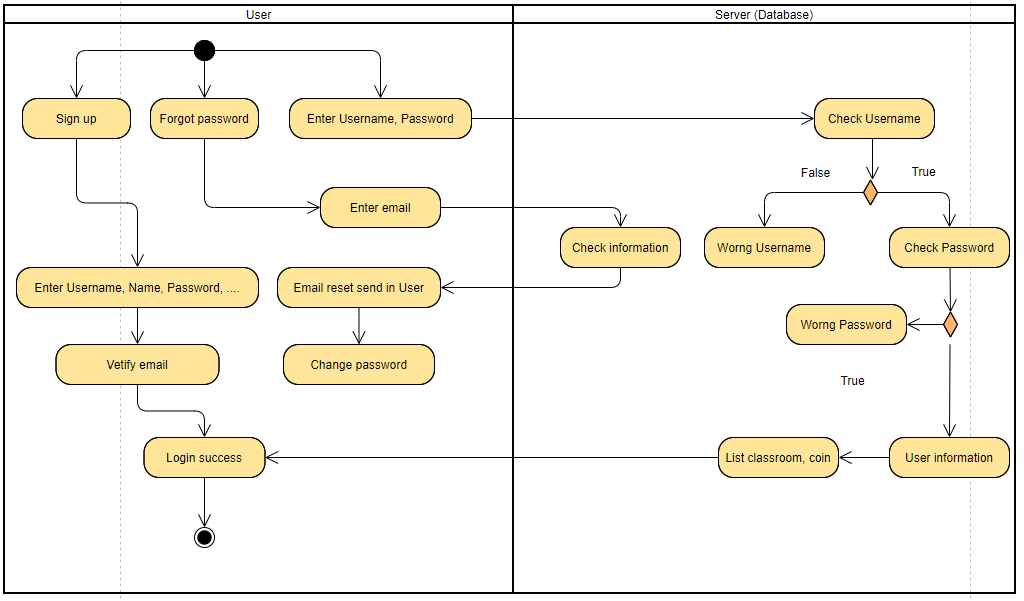


Figure 27: Activity đăng nhập, đăng ký

3.3.2 Chức năng landslide map

Mô tả biểu đồ: người dùng có thể xem được tất cả các địa điểm xảy ra sạt lở đất sẽ được biểu thị trên google map. Người dùng cũng có thể xem vị trí hiện tại của chính mình hay xung quanh xem có các vụ sạt lở đất hay không. Người dùng có thể cập nhật lại map để tải lại những vụ sạt lở mới xảy ra.

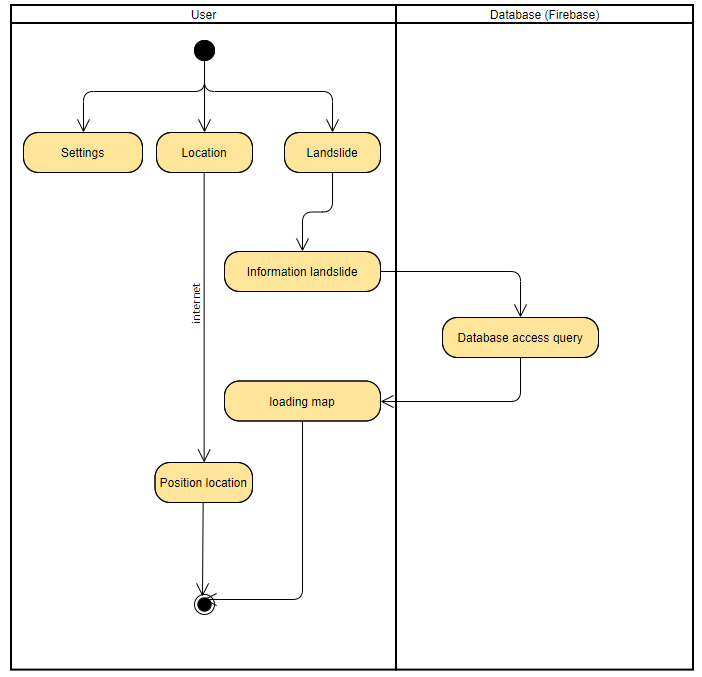


Figure 28: landslide map

3.3.3 Chức năng cấu hình

Mô tả chức năng: người dùng có thể sửa thông tin của chính mình. Ngoài ra, họ cũng có thể đổi ngôn ngữ để thuận tiện cho việc sử dụng ứng dụng. Ứng dụng cũng tích hợp thế tính năng Expert để có thể là tính năng tương lai sẽ được update sau.

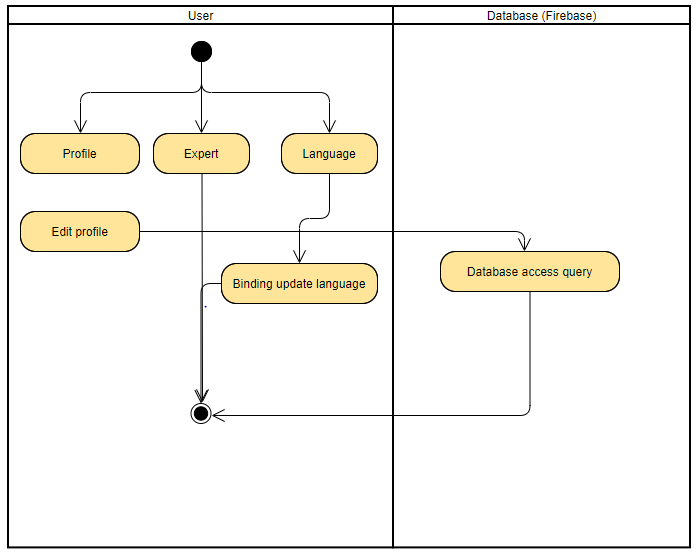


Figure 29: activity cấu hình

3.4 Triển khai trên Xamarin

3.4.1 tạo project Xamarin trên visual 2019 (Windows)

Bược 1: mở visual studio 2019 chọn phần “Create a new project”

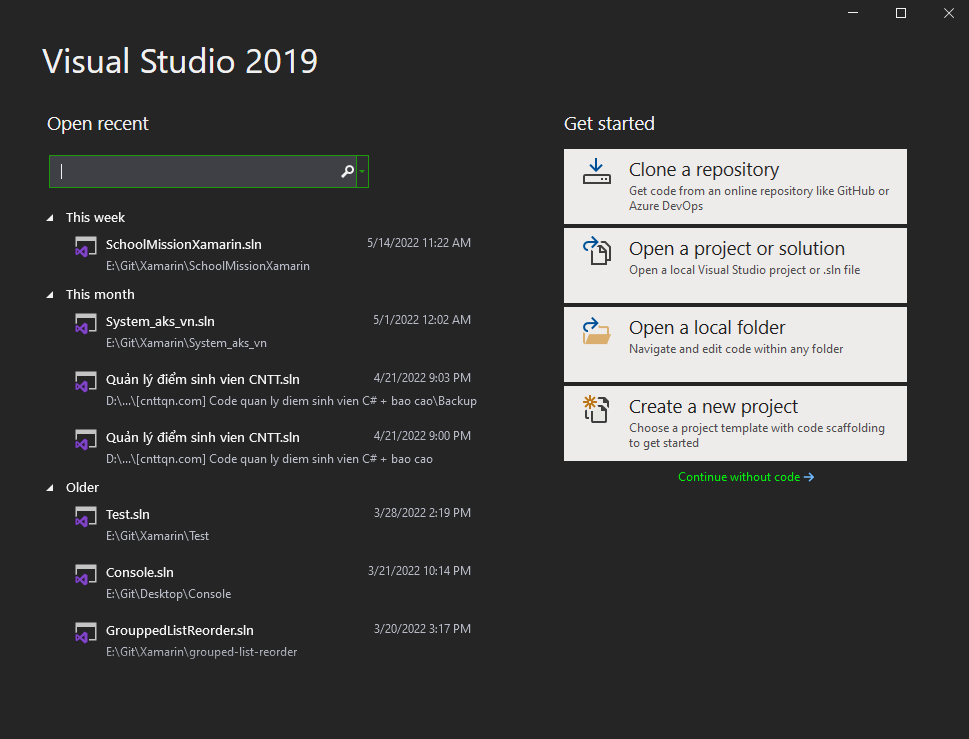


Figure 30: mở visual studio 2019

Bước 2: Search Xamairn và chọn Mobile App (Xamarin.Form)

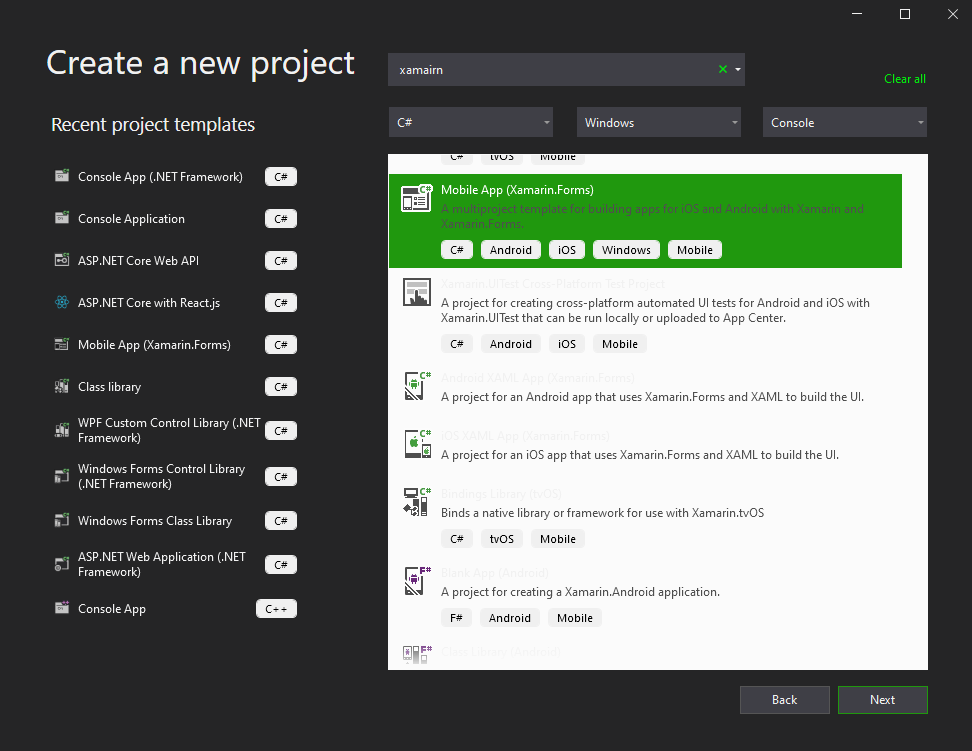


Figure 31: tạo project xamairn trên visual studio

Bước 3: Điền Project Name và vị trí lưu Project và nhấn vào create

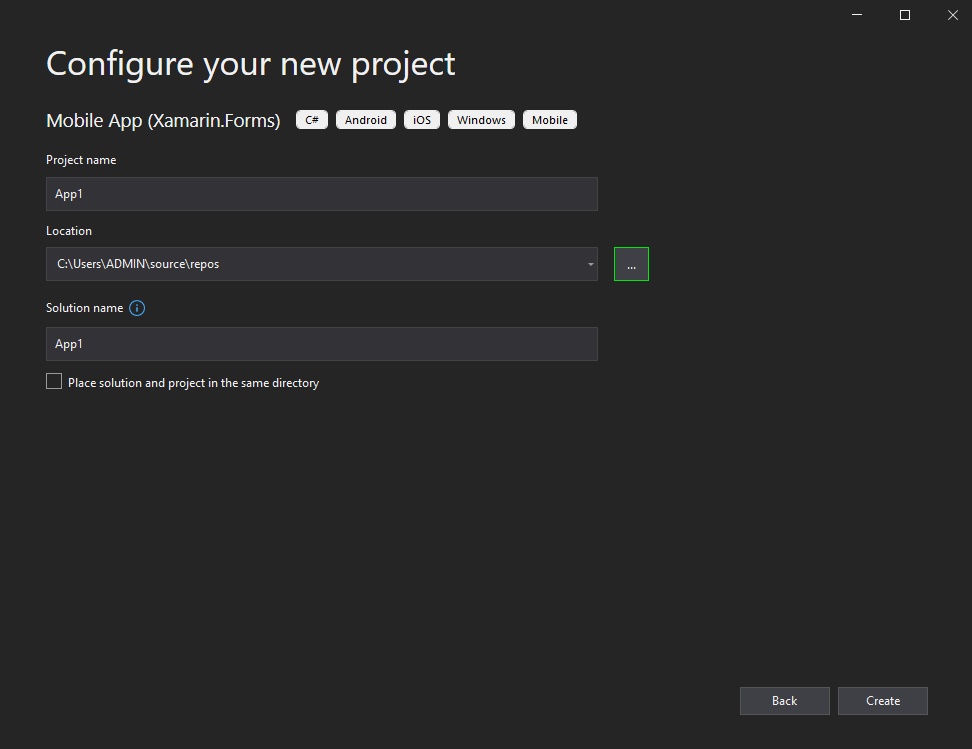


Figure 32: thông tin cơ bản của project

Bước 4: Chọn loại template của project, ở đây ta tạo Blank project

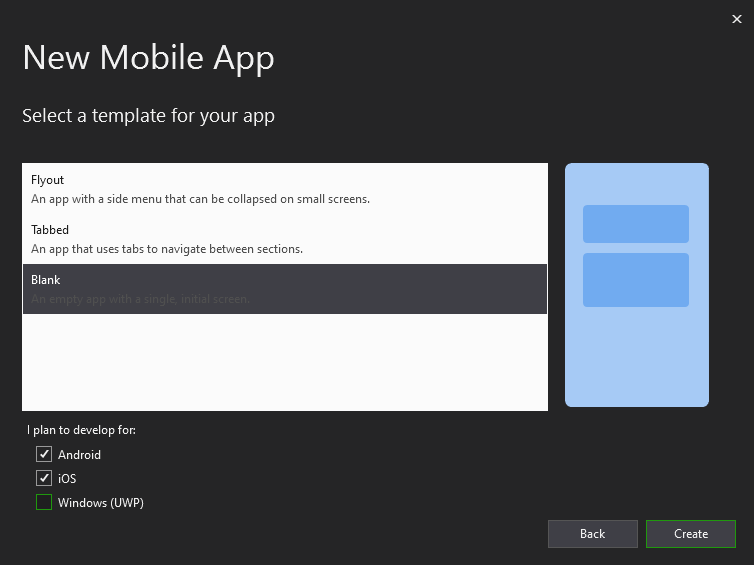


Figure 33: loại template cho ứng dụng

Bước 5: hoàn thành việc tạo ứng dụng Xamarin

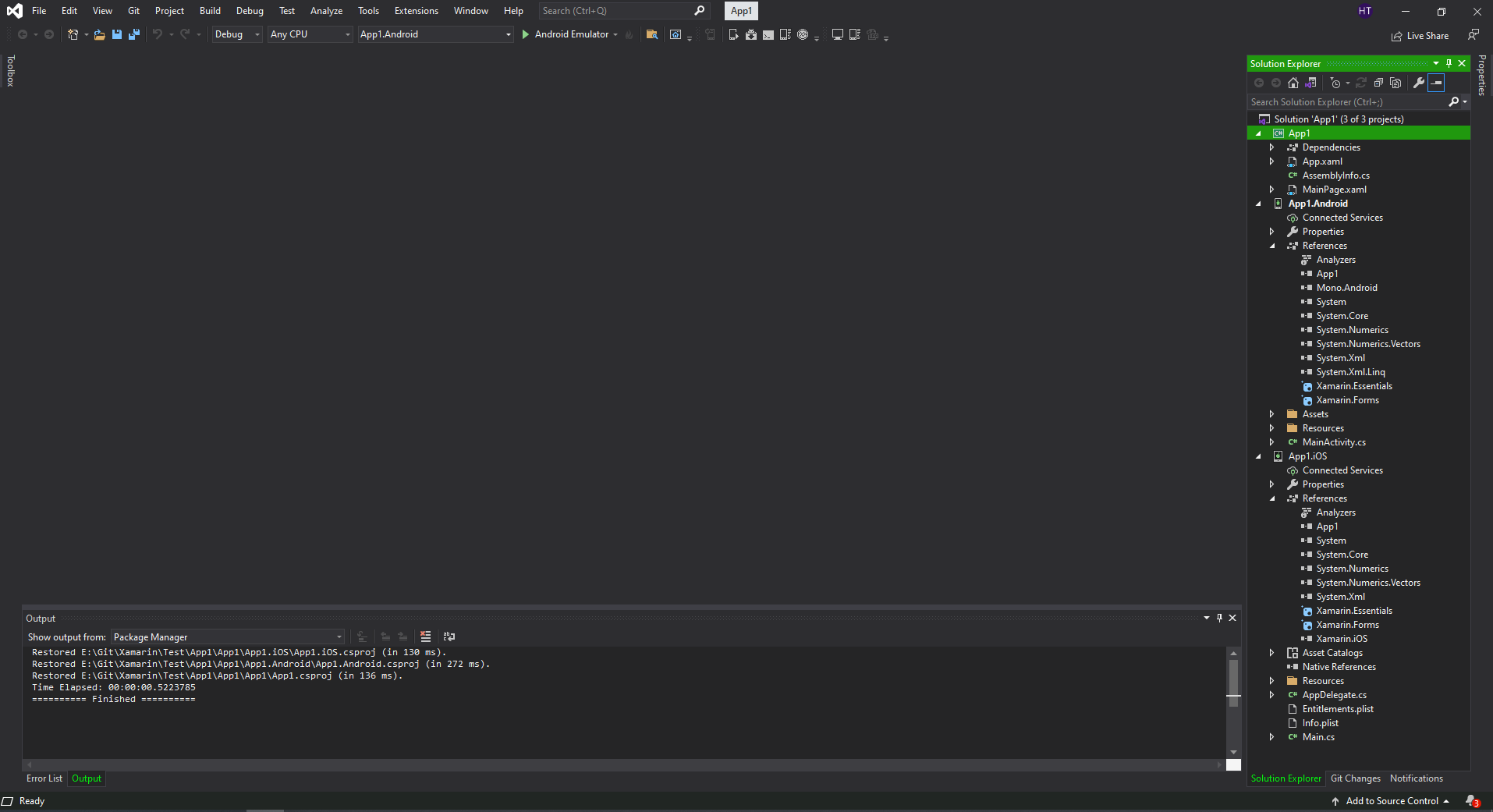


Figure 34: hoàn thành việc tạo project xamairn

3.4.2 Tạo khuôn mô hình MVVM

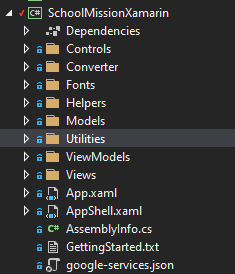


Figure 35: tạo folder theo chuẩn mô hình MVVM

Việc tạo các thư mục này chính là mẫu kiến trúc Model – View – ViewModel được sử dụng khá rộng dãi hiện nay. Việc này giúp việc tạo ứng dụng và phần xử lý được phân tách rạch ròi các khía cạnh của ứng dụng (logic về nhập liệu, logic xử lý tác vụ và logic về giao diện). Mô hình MVVM được sử dụng ở bất kỳ project nào vì tính đa dụng của nó.

Trong đó giao diện (UI logic) thuộc về View, phần xử lý các thao tác của người dung, tương tác chính là ViewModel. Logics các tác vụ ((Business logic – là logic xử lý thông tin, mục đích chính của ứng dụng) thuộc về model. Sự phân chia này giúp bạn giảm bớt được sự phức tạp của ứng dụng và chỉ tập trung vào mỗi khía cạnh cần được cài đặt ở mỗi thời điểm.

3.4.3 Tiến hành triển khai mô hình MVVM

Sau khi đã tạo khuôn maaix mô hình MVVM thì ta cần phải làm cho chương trình hiểu và nó thực thi đúng theo các bước đã được thực hiện theo mô hình MVVM.

Đầu tiên, ta phải tạo 1 file HomePage.xaml ở folder Views file này sẽ chứa giao diện hiển thị nên màn hình của người dung. Sau đó, ở folder ViewModel ta tạo 1 file tên tương tự HomeViewModel.cs, file này có tác dụng là xử lý các thao tác của người dung xử lý tiền giữ liệu.

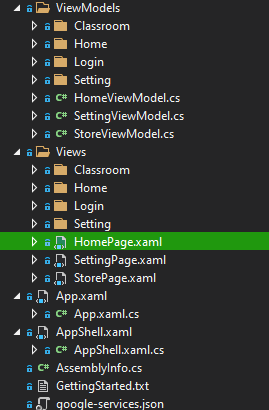


Figure 36: Tạo pattern MVVM

Tiếp đó, ở file HomePage.xaml ta thêm đoạn code bên dưới vào phần giao diện. Mục đính chính llafBindingContext từ View xuống ViewModel. Là cầu nối View với ViewModel.

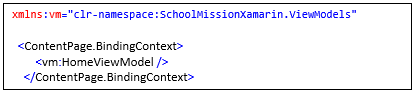


Figure 37: Binding context với ViewModel

Tiếp đó, ở file HomeViewModel.cs sẽ kế thừa 1 class có nội dung cụ thể như bên dưới. Mục tiêu, là tất cả các ViewModel sẽ có chung một cấu trúc đã quản lý và việc xử lý dữ liệu thuận tiện hơn, tiết kiện thời gian, code ngắn hơn.

Nội dung, BaseViewModel.cs sẽ có:

* Mỗi page sẽ có tiêu đề và Busy để loading
* Mỗi page đều sẽ phải CRUD đến database (firebase)
* Mỗi page đều sẽ có các Field sẽ thay đổi data liên tục thì sẽ phải Notification để ập nhật lại giao diện ngay lập tức.
* Mỗi page đều là các Navigation con của Page Parent

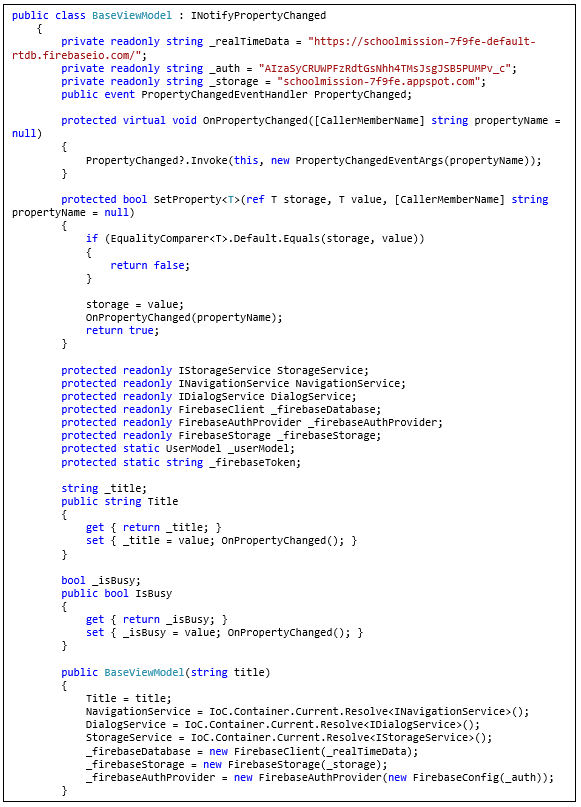


Figure 38: BaseViewModel

Dưới đây là phần code chung của các file trong ViewModel



Figure 39: form chung của các ViewModel

Vậy thì việc Binding dữ liệu diễn ra như nào

Đây là phần code giao diện hiển thị số Point người dung sở hữu. Hãy chú ý đến thuộc tính Text. Nó đang Binding đến biến CurrentUser.Point

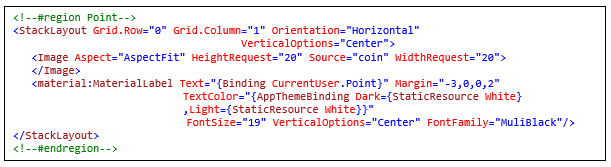


Figure 40: giao diện xaml binding data

Vậy bên ViewModel tức file HomeViewModel.cs sẽ phải có 1 biến tên là CurrentUser. Biến này có kiểu dữ liệu là UserModel và trong thuộc tính của class này có field là Point.

Ngoài ra, để data có thể cập nhật ngay lập tức nên giao diện thì phải có function Setproperty.

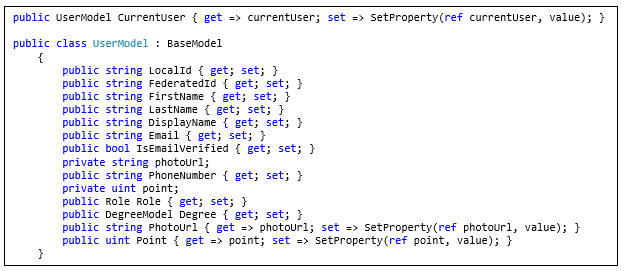


Figure 41: model của thông tin User

3.5 Một số chức năng thực hiện

3.5.1 Chức năng đăng nhập (login)

Bước 1: tạo giao diện màn hình login

Giao diện màn hình login với một số field chính là: Username, password, forgot password, login

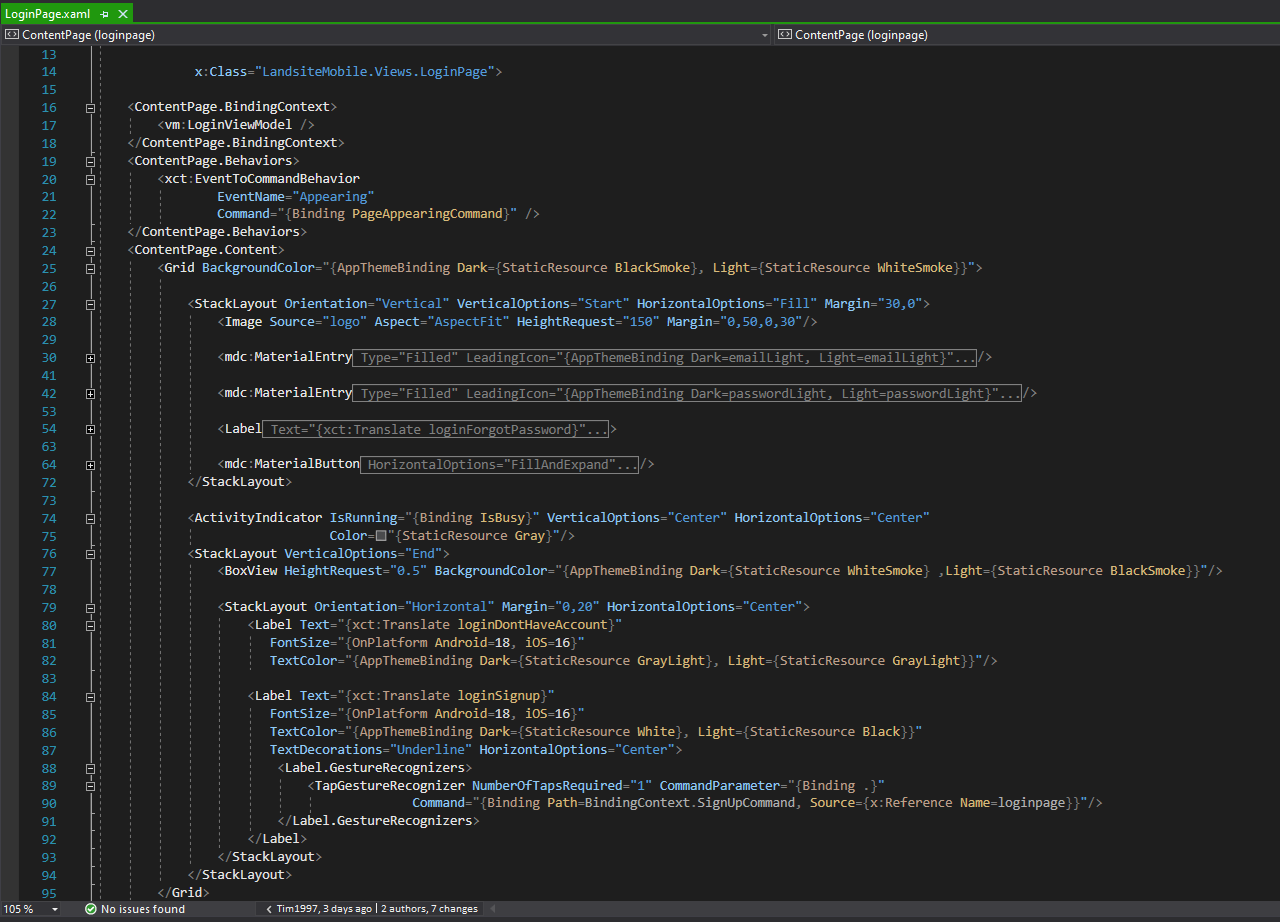


Figure 42: UI màn hình đăng nhập

Tiếp theo đến phần xử lý dữ liệu. Ở LoginViewModel.cs sẽ gồm tất cả các Property cần Binding data nên Views như bên dưới

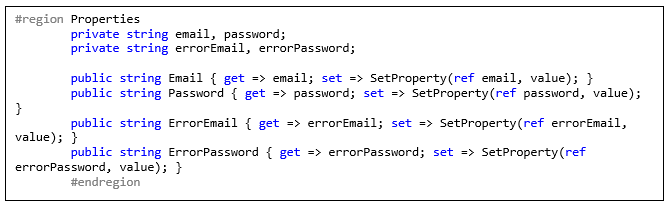


Figure 43: Properties của LoginViewModel

Tiếp đến ta cần phải khởi tạo giá trị cơ bản cho một số Property cần hiển thị data luôn như Email sẽ được load từ data lưu trữ trong App hay phải set background cho app thông qua việc kiểm tra từng khung giờ.

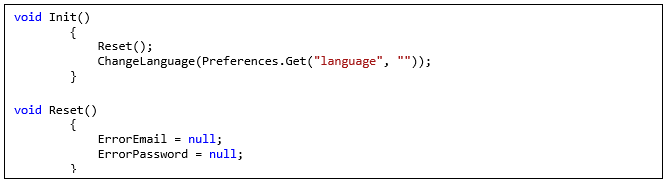


Figure 44: khởi tạo các giá trị cơ bản

Tiếp theo, khi người dung đã nhập Username và Password thì sẽ bấm vào button Login. Khi đó bên ViewModel sẽ call đến Command LoginCommand. Hàm này sẽ kiểm tra các điều kiện cần trước khi gửi query đến database nếu sai sẽ thông báo về cho người dung. Nếu kiểm tra xong hết thì sẽ query đến server và trả dữ liệu về để xử lý.



Figure 45: xử lý chức năng login

Bước 5: Kết quả ta sẽ được giao diện như sau

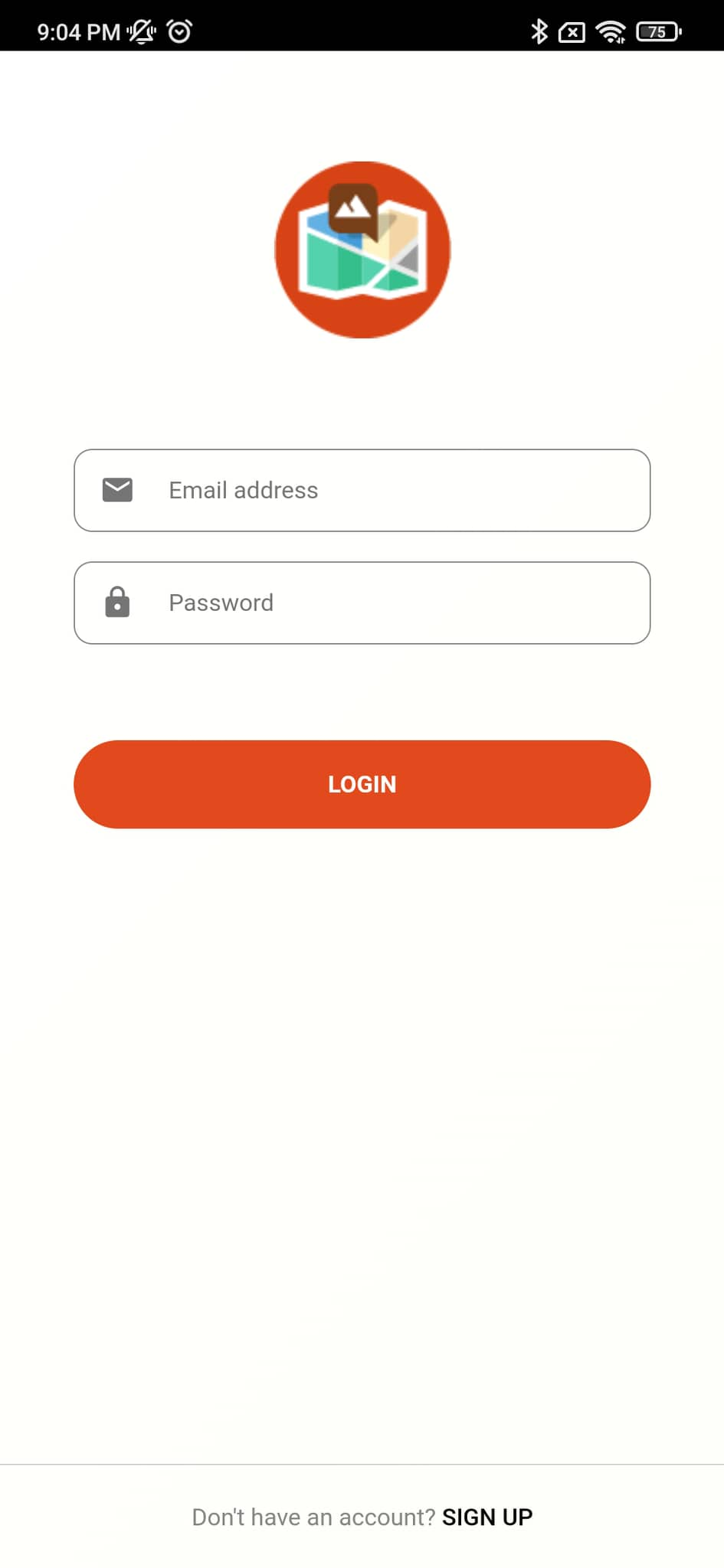


Figure 46: giao diện đăng nhập

*Hình STYLEREF 1 \s 3. SEQ Hình \\* ARABIC \s 1 43: Màn hình Login trên IOS*

*Hình STYLEREF 1 \s 3. SEQ Hình \\* ARABIC \s 1 42:Màn hình Login trên Android*

# DANH MỤC THAM KHẢO

*britannica*. (n.d.). Retrieved from https://www.britannica.com/technology/Internet

Contributor, T. (n.d.). *techtarget*. Retrieved from https://www.techtarget.com/whatis/definition/Model-View-ViewModel

*microsoft*. (n.d.). Retrieved from https://docs.microsoft.com/en-us/xamarin

*mongodb*. (n.d.). Retrieved from https://www.mongodb.com/nosql-explained

*sharpcorner*. (n.d.). Retrieved from https://www.c-sharpcorner.com/article/xamarin-forms-working-with-firebase-realtime-database-crud-operations/

Team, E. (n.d.). *educative*. Retrieved from https://www.educative.io/edpresso/what-is-firebase

*w3schools*. (n.d.). Retrieved from https://www.w3schools.com/cs/cs\_oop.php

# KẾT LUẬN

**Kết luận chung**

Sau quá trình tìm hiểu, phân tích, thiết kế và sự hướng dẫn của cô XXX. Ứng dụng thống kê (Landslide) đã hoàn thành trên cả hai nền tảng Android, IOS và có thể Public trên Store.

Ứng dụng đã đạt được đầy đủ các yêu cầu của học sinh, sinh viên và giáo viên từ giao diện đến chức năng. Hoạt động một cách ổn định (Phụ thuộc vào tốc độ mạng).

Nâng cao khả năng làm việc nhóm, sự liên kết giữa các thành viên trong dự án. Tiếp xúc với công nghệ mới, nền tảng mới đã đem lại rất nhiều kiến thức về chuyên môn.

Được tiếp xúc trực tiếp với dự án thật. Cách tư duy hệ thống, cách phát triển hệ thống theo từng giai đoạn. Làm việc Tester nâng cao hiệu suất công việc và kỹ năng cá nhân.

Tuy vây, do kiến thức còn hạn hẹp và tiếp xúc với công nghệ mới vẫn còn bỡ ngỡ nên cần nhiều thời gian hơn để tìm hiểu và thực hành. Cần sự giúp đỡ khá nhiều cô XXX. Cách bố trí Code vẫn còn xấu và chưa có tính hệ thống.

**Định hướng**

Việc mở rộng và phát triển các chức năng khác của hệ thống là rất dễ dàng.

Ứng dụng này chỉ là một dự án. Việc phát triển các dự án khác là điều có thể làm được.

Cải thiện kỹ năng Code của mình và tư duy hệ thống hiệu quả hơn có tính mở rộng hơn.