

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----

****

**BÁO CÁO**

**ĐỒ ÁN TỐT NGHIỆP**

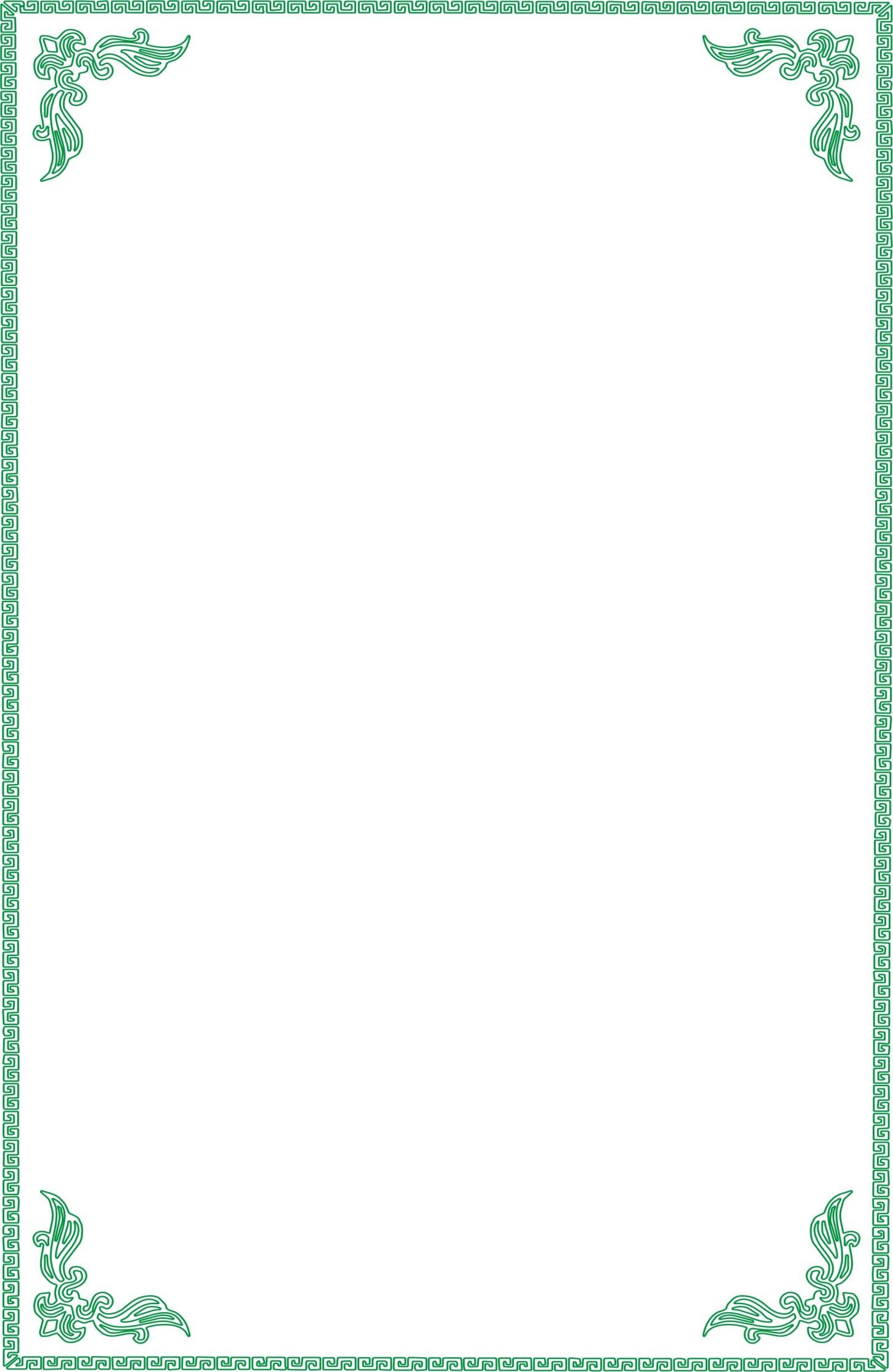
**ĐỀ TÀI: PHÁT TRIỂN ỨNG DỤNG ĐỊNH VỊ GIA ĐÌNH TRÊN NỀN TẢNG IOS**

**Giảng viên HD : ThS Trần Hùng Cường**

**Lớp : KTPM3-K9**

**SV thực hiện : Nguyễn Duy Kiên**

**Hà Nội\_2018**



**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----

****

**BÁO CÁO**

**ĐỒ ÁN TỐT NGHIỆP**

**ĐỀ TÀI: PHÁT TRIỂN ỨNG DỤNG ĐỊNH VỊ GIA ĐÌNH TRÊN NỀN TẢNG IOS**

**Giảng viên HD : ThS Trần Hùng Cường**

**Lớp : KTPM3-K9**

**SV thực hiện : Nguyễn Duy Kiên**

**Hà Nội\_201**

MỤC LỤC

[1](#_Toc512032995)

[Lời nói đầu 8](#_Toc512032996)

[Chương I: Tổng quan về ngôn ngữ lập trình di động 9](#_Toc512032997)

[1.1 Tổng quan về lập trình di động 9](#_Toc512032998)

[1.2. Web App 9](#_Toc512032999)

[*1.2.1. Ưu điểm* 9](#_Toc512033000)

[*1.2.2. Nhược điểm* 10](#_Toc512033001)

[*1.2.3. Kĩ năng cần có* 10](#_Toc512033002)

[1.3. Native App 11](#_Toc512033003)

[*1.3.1 Ưu điểm* 11](#_Toc512033004)

[*1.3.2 Khuyết điểm* 12](#_Toc512033005)

[*1.3.3. Kĩ năng cần có* 12](#_Toc512033006)

[1.4 Hybrid App 12](#_Toc512033007)

[*1.4.1. Ưu điểm* 13](#_Toc512033008)

[*1.4.2. Khuyết điểm* 13](#_Toc512033009)

[*1.4.3. Kiến thức cần biết* 13](#_Toc512033010)

[Chương II: Cơ sở lý thuyết 14](#_Toc512033011)

[2.1. Giới thiệu đề tài 14](#_Toc512033012)

[*2.1.1. Tên đề tài* 14](#_Toc512033013)

[*2.1.2. Nội dung đề tài* 14](#_Toc512033014)

[*2.1.3. Lý do chọn đề tài.* 14](#_Toc512033015)

[*2.1.4 Mục đích thực hiện của đề tài* 14](#_Toc512033016)

[*2.1.5 Phương pháp* 15](#_Toc512033017)

[*2.1.6.Dự kiến kết quả đạt được.* 15](#_Toc512033018)

[2.2 Hệ điều hành IOS 15](#_Toc512033019)

[2.3. Ngôn ngữ lập trình Swift 17](#_Toc512033020)

[*2.3.1: Giới thiệu chung về Swift* 17](#_Toc512033021)

[*2.3.2: Môi trường phát triển XCode* 20](#_Toc512033022)

[*2.3.3: Những lợi ích của Swift* 21](#_Toc512033023)

[*2.3.4: Điểm yếu của Swift* 22](#_Toc512033024)

[2.4. Công nghệ Node.js 23](#_Toc512033025)

[*2.4.1: Giới thiệu chung về Node.js* 23](#_Toc512033026)

[*2.4.2: Ưu điểm của Node.js* 24](#_Toc512033027)

[*2.4.3: Nhược điểm của Node.js* 25](#_Toc512033028)

[2.5. Công nghệ SocketIO 26](#_Toc512033029)

[2.6. Google Firebase 26](#_Toc512033030)

[2.7 Giới thiệu về NoSQL MongoDB 28](#_Toc512033031)

[*2.7.1: Giới thiệu về NoSQL:* 28](#_Toc512033032)

[*2.7.2: Giới thiệu về MongoDB:* 30](#_Toc512033033)

[*2.7.3 Ưu điểm của MongoDB* 33](#_Toc512033034)

[Chương III: Phân tích thiết kế hệ thống & Cơ sở dữ liệu 36](#_Toc512033035)

[3.1 Xác định các Usecase**.** 36](#_Toc512033036)

[*3.1.1. Use tổng quan* 36](#_Toc512033037)

[*3.1.2 Usecase quản lí người dùng theo nhóm* 37](#_Toc512033038)

[*3.1.3 Usecase quản lí người dùng theo nhóm* 37](#_Toc512033039)

[*3.1.4 Quản lí tài khoản người dùng* 38](#_Toc512033040)

[*3.1.5 Quản lí lịch sử người dùng* 39](#_Toc512033041)

[*3.1.6 Quản lý người dung theo vùng* 39](#_Toc512033042)

[3.2: Cơ sở dữ liệu offline 41](#_Toc512033043)

[*3.2.1: Bảng thành viên* 41](#_Toc512033044)

[*3.2.2: Bảng User* 42](#_Toc512033045)

[*3.2.3: Bảng địa điểm* 42](#_Toc512033046)

[*3.2.4: Bảng vòng* 43](#_Toc512033047)

[Chương IV: Ứng dụng định vị gia đình 44](#_Toc512033048)

[4.1.Các chức năng chính: 44](#_Toc512033049)

[*4.1.1 Tạo tài khoản:* 44](#_Toc512033050)

[*4.1.2 Đăng nhập* 44](#_Toc512033051)

[*4.1.3 Quên mật khẩu:* 44](#_Toc512033052)

[*4.1.4 Tự động lưu trạng thái người dùng* 44](#_Toc512033053)

[*4.1.5 Tạo vòng kết nối* 44](#_Toc512033054)

[*4.1.6 Mời vào vòng* 44](#_Toc512033055)

[*4.1.7 Xin vào vòng* 45](#_Toc512033056)

[*4.1.8 Lấy danh sách tham gia vòng của mình* 45](#_Toc512033057)

[*4.1.9 Tạo địa điểm* 45](#_Toc512033058)

[*4.1.10 Sửa địa điểm* 45](#_Toc512033059)

[*4.1.11 Tạo địa điểm* 45](#_Toc512033060)

[*4.1.12 Xem chi tiết địa điểm* 45](#_Toc512033061)

[*4.1.13 CheckIn* 45](#_Toc512033062)

[*4.1.14 SOS* 45](#_Toc512033063)

[*4.1.15 Theo dõi tổng quan các thành viên* 46](#_Toc512033064)

[*4.1.16 Danh sách chi tiết các thành viên* 46](#_Toc512033065)

[*4.1.17 Chi tiết thành viên* 46](#_Toc512033066)

[*4.1.18 Lịch sử di chuyển* 46](#_Toc512033067)

[*4.1.19 Thông báo* 46](#_Toc512033068)

[4.2. Giao diện chương trình 47](#_Toc512033069)

[*4.2.1: Màn hình home* 47](#_Toc512033070)

[*4.1.2 : Màn hình danh sách địa điểm* 48](#_Toc512033071)

[*4.1.3 : Màn hình thêm địa điểm* 49](#_Toc512033072)

[*4.1.4 : Màn hình vòng* 50](#_Toc512033073)

[*4.1.5 : Màn hình lấy mã code* 51](#_Toc512033074)

[*4.1.6: Màn hình tham gia vòng* 52](#_Toc512033075)

[*4.1.7 : Màn hình danh sách thành viên* 53](#_Toc512033076)

[*4.1.8 Màn hình checkIn* 54](#_Toc512033077)

[*4.1.9 : Màn hình danh sách yêu cầu vào vòng* 55](#_Toc512033078)

[*4.1.10 Màn hình xem chi tiết vòng trước khi gia nhập* 56](#_Toc512033079)

[*4.1.11 Màn hình lịch sử di chuyển* 57](#_Toc512033080)

[KẾT LUẬN 58](#_Toc512033081)

[Tài liệu tham khảo 59](#_Toc512033082)

# Bảng danh mục hình ảnh

|  |  |
| --- | --- |
| **Số thứ tự** | **Tên hình ảnh** |
| Hình 1.1 | Các hướng phát triển lập trình di động |
| Hình 1.2 | Web app |
| Hình 1.3 | Native app |
| Hình 1.4 | Hybird app |
| Hình 2.1. | Giao diện XCode |
| Hình 2.2. | Minh họa bộ sưu tập |
| Hình 3.1 | Biểu đồ Usecase tổng quan |
| Hình 3.2 | Biểu đồ usecase quản lí người dùng theo nhóm |
| Hình 3.3 | Usecase quản lí người dùng theo nhóm |
| Hình 3.4. | Biểu đồ usecase quản lí tài khoản người dùng |
| Hình 3.5. | Biểu đồ usecase quản lí lịch sử người dùng |
| Hình 3.6. | Biểu đồ usecase quản lý người dùng theo vùng |
| Hình 4.1 | Màn hình home |
| Hình 4.2 | Màn hình danh sách địa điểmHình |
| Hình 4.3 | Màn hình thêm địa điểm |
| Hình 4.4 | Màn hình danh sách vòng |
| Hình 4.5 | Màn hình lấy code của vòng |
| Hình 4.6 | Màn hình tham giavòng |
| Hình 4.7 | Màn hình danh sách thành viênHình |
| Hình 4.8 | Màn hình checkIn |
| Hình 4.9 | Màn hình yêu cầu tham gia vòng |
| Hình 4.10 | Xem chi tiết vòng |
| Hình 4.11 | Màn hình lịch sử di chuyển |

# Lời nói đầu

Công nghệ thông tin ngày càng phát triển, nhu cầu cuộc sống của mọi người ngày càng phát triển. Do đó nhu cầu chia sẻ thông tin, gắn kết mọi người theo đó cũng tăng theo.

Các thiết bị di động ngày càng phát triển, nó giúp cho mọi người, mọi nhà có thể dễ dàng kết nối với nhau thông qua mạng di động hoặc wifi. Tại Việt Nam nhu cầu giải trí , quan tâm tới các thành viên trong gia đình của mọi người hiện tại rất lớn do đó em đã quyết định làm phần mềm định vị gia đình để mọi người có thể quan tâm chăm sóc người thân ngay trên chiếc điện thoại của mình.

Phần mềm Định vị gia đình trên điện thoại IOS của em giúp mọi người có thể quan tâm, theo dõi các thành viên trong gia đình. Tối ưu trải nghiệm của người sử dụng.

Em xin chân thành cảm ơn ThS Trần Hùng Cường đã hướng dẫn, giúp đỡ em hoàn thành ứng dụng này. Em cũng cảm ơn các thầy cô giáo trong khoa đã tạo điều kiện để cho em hoàn thành được sản phẩm. Mong các thầy cô đóng góp ý kiến để giúp em hoàn thiện hơn.

Hà Nội, ngày 1 tháng 5 năm 2018

Sinh viên

Nguyễn Duy Kiên

# Chương I: Tổng quan về ngôn ngữ lập trình di động

## **1.1 Tổng quan về lập trình di động**

Hiện nay, có 3 hướng chính để phát triển một [ứng dụng di động](https://toidicodedao.com/2015/09/01/tong-quan-ve-lap-trinh-ung-dung-di-dong-phan-1/), đó là: **Web App, Native App và Hybrid App**. Mỗi hướng sẽ cần những [kĩ năng](https://toidicodedao.com/2015/08/18/nhung-ki-nang-can-co-cua-mot-web-developer/) riêng, có những ưu nhược điểm riêng, sẽ được nói rõ hơn bên dưới.



Hình 1.1 Các hướng phát triển lập trình di động

## **1.2. Web App**

Hướng Mobile Web thường được áp dụng khi các bạn đã có sẵn một website đang hoạt động. Ta sẽ **tạo thêm 1 trang web riêng cho mobile**, sử dụng HTML, CSS, một số framework hỗ trợ mobile và responsive (Bootstrap, jQuery Mobile, Materialize). Người dùng sẽ trang web dành cho mobile để dùng ứng dụng.

Các xử lý khác liên quan đến [backend](https://toidicodedao.com/2015/08/18/nhung-ki-nang-can-co-cua-mot-web-developer/) như database sẽ được thực hiện phía trên server. Với một số [công nghệ](https://toidicodedao.com/2015/06/25/cach-tiep-can-1-ngon-ngucong-nghe-moi-phan-2/)như [AngularJS](https://toidicodedao.com/2015/07/07/tutorial-tao-ung-dung-chat-voi-50-dong-code-firebase-va-angularjs/), một trang web có thể **giống y hệt một ứng dụng di động thật sự.**

***1.2.1. Ưu điểm***

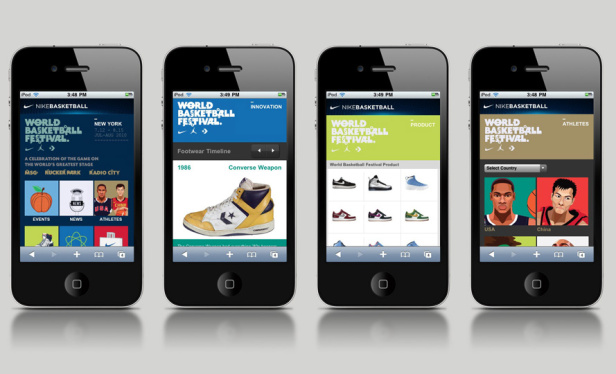
* Chỉ cần có kiến thức về web là viết được
* Viết một lần, chạy được trên mọi hệ điều hành
* Người dùng không cần phải cài app, có thể vào thẳng trang web
* Không cần phải thông qua App Store, **tiết kiệm tiền**
* Dễ nâng cấp (Chỉ việc nâng cấp web là xong)

***1.2.2. Nhược điểm***

* Với một số máy đời cũ, Web App sẽ bị bể giao diện, hiển thị sai, hoặc [javascript](https://toidicodedao.com/tag/javascript-sida/) không chạy.
* **Performance chậm**
* Không thể tận dụng được các tính năng của di động: Push notification, chụp hình, nghiêng máy, định vị GPS…

***1.2.3. Kĩ năng cần có***

* Kiến thức HTML, CSS, Javascript cơ bản.
* Kiến thức về một số framework responsive/mobile như: jQuery Mobile, Bootstrap, …
* Một số framework javascript để viết Single Page Application: [AngularJS](https://toidicodedao.com/2015/07/07/tutorial-tao-ung-dung-chat-voi-50-dong-code-firebase-va-angularjs/), EmberJS, …



Hình 1.2 Web app

## **1.3. Native App**



Hình 1.3 Native app

Viết Native App nghĩa là lập trình viên**sẽ sử dụng IDE, SDK mà nhà sản xuất cung cấp** để lập trình ra một ứng dụng, build ứng dụng đó thành file cài và gửi lên App Store để kiểm duyệt. Người dùng sẽ phải tìm ứng dụng trên App Store, tải về máy và chạy.

Đây là hướng phát triển được áp dụng nhiều nhất, điển hình là game Flappy Bird của Nguyễn Hà Đông. Với những ứng dụng game, xử lý ảnh, cần tính toán nhiều, **Native App là lựa chọn duy nhất**.

Với những hệ thống lớn, cần đồng bộ, ta vẫn phải **viết phần back-end trên server**. Server sẽ đưa ra một số API. Native app lấy dữ liệu về máy, truyền dữ liệu lên server thông qua các API này.

***1.3.1 Ưu điểm***

* Tận dụng được**toàn bộ những tính năng của device**: Chụp ảnh, nghiêng máy, rung, GPS, notification.
* Có thể chạy được offline.
* Performance rất nhanh, vì code native sẽ được chạy trực tiếp.
* Là lựa chọn duy nhất cho các ứng dụng game, xử lý hình ảnh hay video …

***1.3.2 Khuyết điểm***

* Cần cài đặt nặng nề (Eclipse, XCode, Android SDK, …), khó tiếp cận.
* Với mỗi hệ điều hành, ta phải viết một ứng dụng riêng. **Khó đảm bảo sự đồng bộ giữa các ứng dụng** (1 button trên Android sẽ khác 1 button trên iOS, pop cũng khác).
* Cần phải**submit app lên App Store**, mỗi lần update phải thông báo người dùng.
* Code mệt và lâu hơn so với Mobile Web.

***1.3.3. Kĩ năng cần có***

* Ngôn ngữ lập trình: Java cho Android, Objective-C hoặc Swift cho iOS, [C#](https://toidicodedao.com/tag/c-hay-ho/) cho Windows Phone.
* Kiến thức chuyên sâu về ứng dụng: View, Action, Adapter trong Android …
* Cách xây dựng Web Serivce, Restful API, cách gọi API từ device, …

## **1.4 Hybrid App**

Hybrid App **kết hợp những ưu điểm** của Mobile Web và Native App. Ta xây dựng một ứng dụng bằng HTML, CSS, Javascript, chạy trên WebView của mobile. Tuy nhiên, Hybrid App vẫn có thể tận dụng những tính năng của device: chụp hình, GPS, rung, ….

Hybrid App sẽ được viết dựa trên một **cross-platform framework**: Cordova, Phonegap, Titanium, …. Ta sẽ gọi những chức năng của mobile thông qua API mà framework này cung cấp, dưới dạng Javascript. Bạn chỉ cần viết một lần, những framework này sẽ tự động dịch ứng dụng này ra các file cài đặt cho Android, iOS và Windows Phone.

Một số ứng dụng **không quá nặng về xử lý**, cần tận dụng chức năng của device sẽ chọn hướng phát triển này.

***1.4.1. Ưu điểm***

* Chỉ cần biết HTML, CSS, JS (Thế nên mình mới khuyên các bạn nên học [Javascript](https://toidicodedao.com/2015/01/20/hoc-ngon-ngu-lap-trinh-nao-bay-gio-phan-cuoi/)).
* Viết một lần, chạy được trên nhiều hệ điều hành
* Tận dụng được các chức năng của device.

***1.4.2. Khuyết điểm***

* Không ổn định, **khó debug**. Framework sẽ dịch code của bạn thành code native, việc sửa lỗi ứng dụng khá khó vì bạn không biết code sẽ được dịch ra như thế nào.
* Performance chậm.
* Cần cài đặt nhiều thứ (Titanium, Cordova đều bắt phải cài đặt SDK này nọ thì mới build ứng dụng được).

***1.4.3. Kiến thức cần biết***

* Cách dùng một số framework CSS, Javascript: jQuery Mobile, Ionic Framework, AngularJS, Bootstrap, …
* Kiến thức về các **cross-platform framework**: Titanium, Cordova, Phonegap.
* Cách xây dựng Web Serivce, Restful API, cách gọi API từ device, … (Hybrid app cũng sẽ kết nối với server thông qua API như Native App).

[](https://toidicodedao.files.wordpress.com/2015/08/monday_08-04-2013_benefits-of-hybrid-mobile-app-development-e1439996431243.jpg)

Hình 1.4 Hybird app

# Chương II: Cơ sở lý thuyết

## **2.1. Giới thiệu đề tài**

### *2.1.1. Tên đề tài*

Phát triển ứng dụng *định vị gia đình* trên nền tảng IOS và công nghệ Nodejs

### *2.1.2. Nội dung đề tài*

Đề tài gồm 3 phần :

* Phần Client : phần client được viết bằng ngôn ngữ Swift
* Phần Server : phần server được viết bằng ngôn ngữ Node.js
* Phần cơ sở dữ liệu : cơ sở dữ liệu MongoDB

### *2.1.3. Lý do chọn đề tài.*

Khoa học công nghệ trên thế giới ngày càng phát triển mạnh mẽ và có những bước tiến đáng nể. Dễ thấy nhất là sự xuất hiện và thay đổi của những chiếc điện thoại thông minh trong nhiều năm gần đây. Quả không sai khi gọi nó là “smartphone” bởi ngoài những tính năng cơ bản của một chiếc *điện thoại di động* như nghe, gọi, nhắn tin thì chiếc điện thoại thông minh còn sở hữu những ứng dụng vô cùng hiện đại ngang tầm với một chiếc máy tính xách tay. Và những ứng dụng ấy mang lại cho người rất nhiều công dụng quan trọng trong cuộc sống hiện đại này. Ngày nay điện thoại thông minh được sử dụng rất phổ biến trong cuộc sống. Người ta không còn sử dụng nó với mục đích chính là nghe gọi nữa mà điện thoại thông minh được sử dụng nhiều các ứng dụng hiện đại khác mà nó mang lại.

Nhu cầu giải trí trên chiếc điện thoại của người sử dụng ngày càng tăng cao do đó để đáp ứng nhu cầu của người sử dụng em đã thực hiện đề tài này để giúp người dùng xác định vị trí người thân của mình dễ dàng nhất.

### *2.1.4.Mục đích thực hiện của đề tài*

Khi thực hiện đề tài này mục đích mà em hướng tới là có thể nắm bắt được quá trình xây dựng một sản phẩm hoàn thiện phải trả qua những giai đoạn nào. Tìm hiểu rõ hơn về một ngôn ngữ để xây dựng nên một ứng dụng cụ thể. Các giai đoạn đều có những khó khăn riêng nhưng qua đó em cũng đã tích lũy được những kinh nghiệm kiến thức cho bản thân để có thể áp dụng được vào thực tế

Ứng dụng cũng là sản phẩm thực tế đầu tiên nên để có thể hoàn thiện hơn nữa cũng nhờ sự giúp đỡ rất nhiều từ thầy cô bạn bè, qua đó giúp ích cho em trong việc tiếp thu kiến thức, học hỏi và sắp xếp thời gian phân công công việc cần phải làm một cách khoa học hơn.

Ứng dụng định vị gia đình, giúp các thành viên luôn giữ kết nối với nhau, luôn quan tâm chăm sóc được mọi người trong gia đình và hoàn toàn miễn phí. Đảm bảo ứng dụng hoạt động ổn định trên hệ điều hành iOS 10 trở lên các máy iPhone 5 trở lên.

### *2.1.5 Phương pháp*

* Tìm kiếm tìm hiểu thông tin trên mạng.
* Ghi chép nghiên cứu thông tin về sản phẩm.
* Lên ý tưởng, sử dụng kiến thức đã học để phân tích thiết kế hệ thống cho trang ứng dụng.
* Tìm hiểu, xây dựng chức năng cho ứng dụng.
* Áp dụng các công nghệ như swift và Nodejs, SocketIO, FCM để áp dụng cho việc xây dựng ứng dụng.
* Tổng hợp lại tất cả kiến thức đã học và hoàn thành báo cáo.

### *2.1.6.Dự kiến kết quả đạt được.*

* Đề tài sẽ đem lại lượng kiến thức khá đầy đủ về những gì chúng ta cần biết và tìm hiểu về ứng dụng mobile và việc vận dụng mã nguồn mở để xây dựng nó.
* Cung cấp những tài liệu cần thiết về các công nghệ được áp dụng .
* Mang đến một sản phẩm công nghệ với nhiều chức năng thú vị .
* Cung cấp những tài liệu cũng như mã nguồn cần thiết cho những ai cần tìm hiểu cũng như muốn làm một trang web hoàn chỉnh.

## **2.2 Hệ điều hành IOS**

iOS là hệ điều hành di dộng của Apple được cài sẵn trên các thiết bị iPhone, iPad và iPod touch hay Apple TV của hãng và do chính Apple nghiên cứu và phát triển. Ban đầu iOS được gọi là iPhoneOS và đến tháng 6 năm 2010 thì chính thức được đổi gọi tên là “iOS” như hiện nay.

Hệ điều hành iOS được công bố cùng chiếc iPhone 2G tại Macworld Conference & Expo vào ngày 9/1/2007 và chính thức phát hành trong tháng 6 sau đó. Khi ra mắt, hệ điều hành này không không hỗ trợ các ứng dụng từ các nhà phát triển thứ 3 (có thể hiểu là các lập trình viên ứng dụng không thuộc Apple). Nguyên nhân của việc từ chối các ứng dụng từ bên thứ 3 được Steve Jobs đưa ra là các nhà phát triển có thể xây dựng ứng dụng trùng chức năng với các ứng dụng gốc trên iPhone. iOS ra đời đã đánh dấu và mở ra một kỉ nguyên mới cho sự phát triển của điện thoại màn hình cảm ứng.

Đến cuối năm 2011, iOS chiếm 60% thị phần điện thoại thông minh và máy tính bảng. Cuối năm 2012, iOS chiếm 21% thị phần hệ điều hành điện thoại thông minh trên thị trường và 43,6% thị trường hệ điều hành máy tính bảng. Và đến tháng 6 năm 2014, App Store của Apple chứa hơn 1.200.000 ứng dụng iOS, và được tải về hơn 60 tỷ lần.

Giao diện người dùng của iOS được dựa trên khái niệm về thao tác trực tiếp bằng tay, sử dụng các cử chỉ đa cảm ứng. Các yếu tố giao diện điều khiển bao gồm các thanh trượt, công tắc, và các nút. Tương tác với các hệ điều hành bao gồm các cử chỉ như chạm, trượt, vuốt.

Một đặc trưng của hệ điều hành iOS đó là Apple hạn chế việc cài ứng dụng từ bên thứ ba (ngoài AppStore) vào thiết bị, việc này có thể khiến iPhone của bạn mắc phải một số lỗi và không còn mượt mà nữa.

* **Ưu điểm:**

+ Nền tảng ổn định, ứng dụng có khả năng tương thích cao. Ứng dụng trên iOS có thể hoạt động mượt mà vì chỉ được tối ưu hóa cho các thiết bị của Apple.

+ Độ tin cậy và bảo mật cao.

+ Chu kỳ ra mắt là một năm vì vậy bạn sẽ yên tâm không phải chạy đua quá nhiều.

+ Ứng dụng phong phú, chất lượng và cập nhật nhanh hơn khi có phiên bản mới.

+ Do tính phổ biến cũng như không có nhiều loại kích cỡ khác nhau nên điện thoại iPhone được rất nhiều hãng sản xuất phụ kiện ưu ái. Người dùng iPhone có thể dễ dàng lựa chọn rất nhiều mẫu ốp lưng, bao da khác nhau. Thậm chí còn có cả ống kính rời cho camera, đế sạc không dây, ốp pin gắn ngoài, đề can trang trí, đế khuếch đại âm thanh…

* **Nhược điểm**

+ Không thể chạy nhiều ứng dụng cùng lúc trên màn hình.

+ Trao đổi dữ liệu đều phải qua iTunes gây bất tiện và tốn thời gian vì đây là con đường duy nhất.

+ Khả năng tuỳ chỉnh hạn chế.

+ iOS chỉ có thể đi kèm với iPhone, trong khi giá các thiết bị Apple thì không hề dễ chịu tí nào.

Phiên bản mới của iOS được phát hành hàng năm. Bản mới nhất hiện tại là iOS 11.2, được phát hành vào ngày 02/12/2017.

## **2.3. Ngôn ngữ lập trình Swift**

### *2.3.1: Giới thiệu chung về Swift*

**Swift** là một ngôn ngữ lập trình hướng đối tượng dành cho việc phát triển iOS và OS X, được giới thiệu bởi Apple tại hội nghị WWDC 2014. Swift được mong đợi sẽ tồn tại song song cùng Objective-C, ngôn ngữ lập trình hiện tại dành cho các hệ điều hành của Apple. Swift được thiết kế là một ngôn ngữ có khả năng phòng chống lỗi cao. Nó được biên dịch với trình biên dịch LLVM. Trước đây Objective C là “đứa con nối dõi tông đường” của Apple, việc ra đời Swift có thể giúp các lập trình viên dễ dàng tiếp cận hơn, đặc biệt nó là mã nguồn mở (các tài nguyên về Swift ta có thể tải về từ trang <https://swift.org/>). Hiện nay Swift được hỗ trợ biên dịch ở 2 hệ điều hành macOS và Ubuntu (14.04 hoặc 15.10 (64 bit) trở lên).

Sự ra đời của Swift chứng kiến sự hợp tác lịch sử giữa Apple và IBM, đây là sự hợp tác mang lại nhiều lợi ích cho Apple, IBM đặc biệt là các nhà lập trình IOS. Sự tham gia của IBM trong cuộc đầu tư, phát triển Swift được xem là cơ hội tốt để phát triển, khai thác, chia sẻ các nguồn lực của IBM và sử dụng Swift Sandbox để mang lại những trải nghiệm nhanh chóng và tuyệt vời hơn. Trong khi đó, Objective – C nằm ngoài sự hợp tác này dẫn đến việc ngôn ngữ này có rất ít những phiên bản cập nhật có khả năng tương thích cao như Swift.

Những ứng dụng đã được lập trình bằng ngôn ngữ Objective khó có khả năng được viết lại với ngôn ngữ lập trình mới. Nói cách khác, sự thay đổi gần đây nhất của Objective – C để dễ dàng hơn khi chuyển đổi thành Swift. Theo thông báo của Tim Cook gửi đến toàn bộ nhân viên khi hợp tác với IBM: “Sự hợp tác giữa Apple và IBM sẽ kết hợp được khả năng tương tác thân thiện của phần cứng và phần mềm trong các thiết bị của chúng ta với kinh nghiệm cũng như trình độ chuyên môn sâu trong lĩnh vực phân tích dữ liệu của IBM”. Điều này cũng là nền tảng để  Craig Federighi của Apple tuyên bố: “Chúng tôi nghĩ Swift sẽ là ngôn ngữ lập trình chủ yếu trong thời gian tới, ngôn ngữ mà mọi người sẽ sử dụng để lập trình trong những thế kỉ tiếp”.

**Swift** không chỉ vượt trội hơn Objective-C nhờ vào cầu nối giữa IBM và Apple mà bản thân ngôn ngữ này cũng ít  mã code và legacy hơn. Swift được xem là một ngôn ngữ súc tích để lập trình, điều này giúp Swift có sự sẵn sàng tốt hơn ngôn ngữ khác. Tuy cô đọng nhưng không có nghĩa là Swift đơn giản, thỉnh thoảng ngôn ngữ này rất khó để sử dụng nhưng không thể phủ nhận những lợi ích nó mang lại đặc biệt là có khả năng tái sử dụng cao. Một ví dụ thực tiễn có thể chứng minh được điều này là phần mềm Lyft. Nó được viết theo ngôn ngữ Swift từ những ý tưởng nền tảng. Ý tưởng này khá mạo hiểu bởi vì nhóm tạo ra Lyft bằng việc sử dụng ngôn ngữ Swift từ những ngày đầu Swift  ra đời và trong suốt quá trình không ngừng cải tiến. Ứng dụng có 75000 dòng chứa khoảng 25000 mã code. Sự cải tiến của Swift không ảnh hưởng đến sự vận hành của Lyft, không gây khó khăn cho khách hàng trong quá trình sử dụng. Đây là những nhận định về Swift của nhóm sáng lập và khách hàng của ứng dụng Lyft.

**Swift** là ngôn ngữ lập trình có thể tránh được nhiều lỗi. Điều này được hỗ trợ bởi cú pháp và ngôn ngữ của Swift có thể loại bỏ được nhiều loại lỗi phổ biến của ngôn ngữ Objective. Sự vận hành ổn định này giúp tránh được những gián đoạn không mong muốn. Tuy không hoàn toàn tránh được những mã xấu nhưng lập trình viên hoàn toàn có thể được bảo vệ để tránh mắc lỗi. Kiểm soát được chất lượng  được xem làm một trong những khả năng vượt trội của Swift khiến nó trở thành một ngôn ngữ lập trình an toàn.

Về tốc độ, Swift nhanh hơn với sự hỗ trợ của thuật toán C++. Báo cáo của một nghiên cứu về Swift và hiệu suất C++ cho thấy để tạo nên một câu chuyện dài súc tích hơn, Swift 1.2 chỉ chậm hơn không đáng kể so với những đối thủ cạnh tranh. Tuy nhiên, Apple đang không ngừng nỗ lực để cải thiện tốc độ của Swift và hiện nay đã tạo nên được những khác biệt trong những phiên bản Swift sau này. Swift 2.0 đã đánh bại được C++ trong nhiều thuật toán máy tính như Mandelbrot. Objective-C chậm hơn vì nó chứa  C API legacy.

Được tạo ra để phục vụ trên nền tảng Apple, Swift được xem là một ngôn ngữ mở, dễ thích ứng với các nền tảng khi đang dần dần thích ứng để được xây dựng trên Linux. Nếu so sánh với Objective thì Swift có thể dễ dàng di chuyển trên nhiều nền tảng hơn. Hiện nay cũng có một động thái để giúp Swift tương thích với Android. Yêu cầu này đã được đề cập và chứng thực bởi tác giả Swift Christ Lattner. Ý tưởng này tuy chưa được nhìn nhận cụ thể nhưng chắc chắn sẽ được đề cập đến trong những kế hoạch phát triển Swift  trong tương lai.

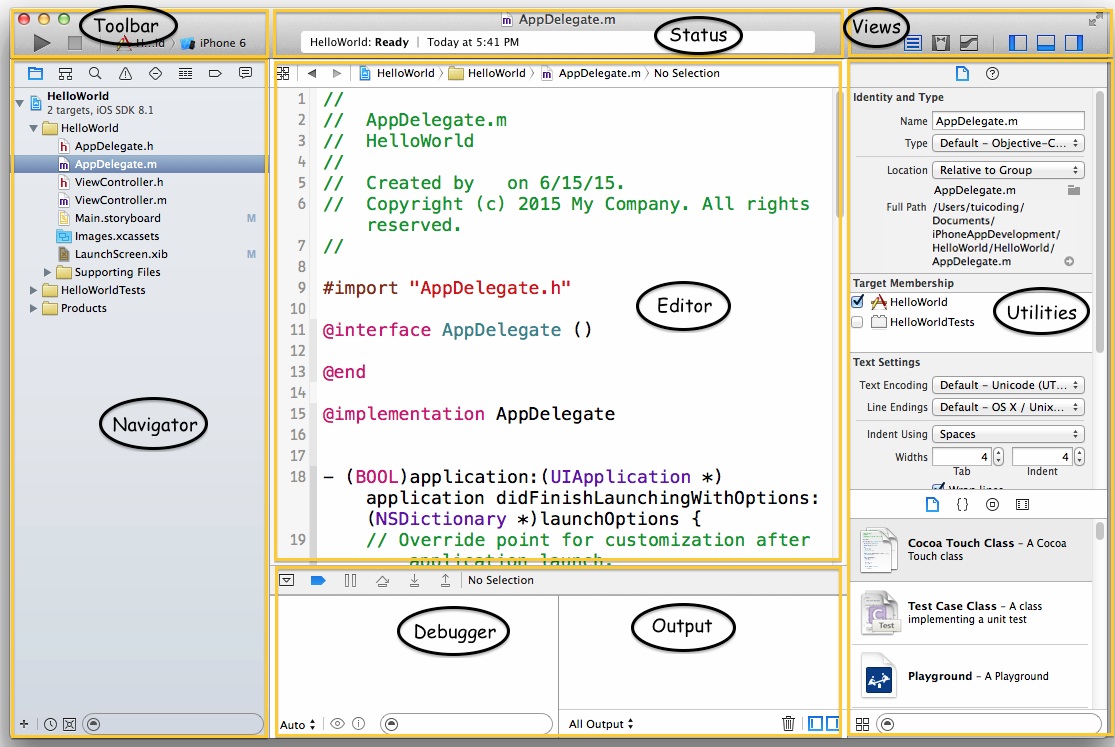
**Swift** là một ngôn ngữ lập trình có khả năng tương tác cao. Swift Playground hay còn gọi là môi trường tương tác riêng cho Swift đã mở ra nhiều cơ hội mới cho các developer. Công cụ này cho phép kiểm tra mã code ngay lập tức mà không cần biên tập những phần lớn hay phải tạo ra một ứng dụng hoàn chỉnh. Môi trường này trực quan hóa dữ liệu và các nhà lập trình có thể nhanh chóng kiểm tra và chỉnh sửa mọi thứ. Điều này đặc biệt được áp dụng cho những cách nhìn nhận quen thuộc và trải nghiệm với mã code.

**Swift** có sự kết nối gần hơn với những nền tảng khác. Đặc điểm này rất quan trọng đặc biệt khi nói về sự hợp tác giữa các nhà lập trình, xây dựng chung những ứng dụng trên nền tảng khác nhau. Ngôn ngữ lập trình hiện đại của Apple cho phép những nhà lập trình không chuyên về lập trình IOS vẫn có thể hiểu được dễ dàng. Nói cách khác, đặc điểm này giúp tiết kiệm được những giải thích và làm rõ, có tác động tích cực đến năng suất làm việc. Hơn nữa, Swift có thể được sử dụng như là một ngôn ngữ viết Script. Đây là một giải pháp thú vị cho cộng đồng IOS để thống nhất cách thức viết của các script. Có thời điểm các **nhà lập trình iOS** bị “chia rẽ” vì hoạt động này. Một vài người viết Script theo Bash, một số người sử dụng Ruby, Python…Swift tạo cơ hội tốt để có thể đáp ứng được tất nhu cầu của một nhà lập trình iOS bằng cách giúp họ cảm thấy dễ dàng hơn lúc thao tác khi sử dụng ngôn ngữ thống nhất và quen thuộc.

### *2.3.2 Môi trường phát triển XCode*

XCode là bộ phát triển phần mềm tích hợp được Apple phát triển chạy trên hệ điều hành Mac để phát lập trình viên có thể phát triển phần mềm chạy trên hệ điều hành Mac và iOS. Phiên bản đầu tiên của XCode được phát hành vào năm 2003 và phiên bản ổn định hiện tại là 9.1 được phát hành vào năm 2017. XCode được phát hành miễn phí cho người dùng Mac download thông qua chợ ứng dụng App Store.

Ở đây tôi chia giao diện ra thành 8 khung nhau như bạn thấy ở hình dưới đây:



Hình 2.1. Giao diện XCode

**Toolbar**: Khung này cung cấp cho bạn các công cụ để chạy ứng dụng (nút hình tam giác phía ngoài cùng tay trái), kết thúc quá trình chạy (nút hình vuông nằm kế), lựa chọn loại thiết bị cho simulator (nút ngoài cùng tay phải).

**Status**: Khung này sẽ cung cấp thông tin cơ bản về dự án như tên dự án, trạng thái của ứng dụng. Ngoài ra khi bạn viết chương trình mà xuất hiện lỗi Xcode cũng sẽ hiển thị một cảnh báo có hình tròn với nền màu đỏ hoặc hình tam giác với nền màu vàng ở phía ngoài cùng tay phải.

**View**: Trong phần này có 6 nút bấm trong đó chia thành 2 nhóm, mỗi nhóm gồm có 3 nút bấm:

* Nhóm tay trái: Nhóm này dùng để điều chỉnh cách hiển thị của khung **Editor**. Mặc định, nút ngoài cùng tay trái sẽ được chọn. Bạn cũng có thể thử với nút ở giữa, khi đó Xcode sẽ hiển thị file .h (file header) và file .m (file implementation) cùng một lúc.
* Nhóm tay phải: Được dùng để ẩn (hiện) các khung **Navigator, Utilities**và**Debugger**, bạn có thể thử click vào các nút này để thấy sự thay đổi.

**Navigator**: Khung này dùng để hiển thị hoặc tìm kiếm theo tên các file và thư mục của dự án. Ngoài ra, nó cũng được dùng để xem thông tin chi tiết về các lỗi xảy ra khi chúng ta lập trình ứng dụng. Để hiển thị nội dung của một file trong khung **Editor**, bạn click chuột lên tên file đó trong khung này. Nếu bạn click đúp lên tên file, Xcode sẽ hiển thị nội dung file trên một cửa sổ mới.

**Editor**: Khung này được dùng để hiển thị và điều chỉnh mã lệnh. Ở đây bạn thấy mã lệnh được hiển thị sử dụng các màu sắc khác nhau. Các **comment** trong mã lệnh được hiển thị với màu xanh dương như bạn thấy từ dòng 1 tới 7. Các dòng này sẽ được bỏ qua khi chương trình được thực thi.

**Utilities**: Khung này chứa các tiện ích khác nhau như giúp lập trình viên xác định tên file (mục **Name**), loại file (mục **Type**), vị trí của file (mục **Full Path**) và các thông tin hỗ trợ khác.

Ngoài ra khi lập trình ứng dụng bạn cũng sẽ tìm thấy các tiện ích rất hữu dụng nằm ở phía cuối khung này. Ví dụ như tiện ích dùng để kéo thả các thư viện, khối lệnh hoặc các thành phần dùng để thiết kế UI có sẵn trong Xcode.

**Debugger**: Khung này được dùng chủ yếu khi bạn chạy ứng dụng. Nó sẽ hiển thị thông tin về lỗi nếu có trước hoặc sau khi chạy chương trình, các thông tin được xuất ra từ chương trình.

### *2.3.3 Những lợi ích của Swift*

Khả năng phát triển ứng dụng di động và kiểm tra thành quả của mình theo thời gian thực sẽ giúp cho quá trình code của các lập trình viên trở nên nhanh và dễ dàng hơn rất nhiều. Ngoài ra, việc không phải chạy chương trình compiler và các chương trình test quá thường xuyên cũng sẽ tiết kiệm được sức mạnh phần cứng cho nhà phát triển. Thay vì tập trung kiểm tra, thử nghiệm các tính năng nhỏ lẻ, lập trình viên Swift có thể dành nhiều thời gian để thực hiện các bài test ở mức độ tích hợp cao hơn một cách kỹ càng hơn.

Lợi ích thứ 2 của Swift là các dòng code rất ngắn và dễ đọc. Trong bài trình bày của Apple, 3 dòng code Objective C có thể gói gọn vào 1 dòng code Swift. Điều này sẽ đẩy nhanh quá trình phát triển ứng dụng và tăng hiệu quả cho quá trình bảo trì/vá lỗi trong tương lai.

Bởi vì “sinh sau đẻ muộn” nên Swift được thừa hưởng nhiều ưu điểm của các ngôn ngữ lập trình khác nhau. Bên cạnh đó, swift cũng rất được Apple ưu ái và thường xuyên nâng cấp, cho đến hiện nay là Swift 4. Vì thế đây sẽ là một ngôn ngữ đa năng, mạnh mẽ, linh hoạt và cũng rất dễ học.

Swift có đầy đủ sức mạnh và những cải tiến của một ngôn ngữ lập trình hiện đại để có thể thay thế Objective-C (vốn đã già cỗi, khó sử dụng và khó học) phát triển các ứng dụng cho iOS, macOS, watchOS và tvOS.

Swift còn được trang bị một tính năng khá hay, đó là Swift Playground. Đây là tính năng giúp chúng ta có thể xem nhanh kết quả thực theo thời gian thực mà không cần phải build hoặc run project.

Cuối cùng và có lẽ là quan trọng nhất, nếu Swift tạo ra các ứng dụng có sức ép phần cứng thấp hơn Objective-C, chúng ta sẽ được tận hưởng trải nghiệm game ấn tượng hơn trên các thiết bị iOS. Khi kết hợp Swift cùng nền tảng Metal mới của Apple, ngay cả vi xử lý A7 cũ kỹ cũng có thể mang tới những trải nghiệm đồ họa ấn tượng nhất.

### *2.3.4 Điểm yếu của Swift*

Điểm yếu lớn nhất của Swift sẽ là tuổi đời quá trẻ của ngôn ngữ lập trình này. Cụ thể hơn, để sử dụng Swift, các nhà phát triển sẽ phải học lại một ngôn ngữ lập trình hoàn toàn mới. Ngành công nghệ thường thích được làm chủ các công nghệ mới nhất, tuyệt vời nhất. Song, mảng ứng dụng iOS cũng đã trở nên lớn mạnh tới mức có hàng nghìn người đã dành hàng tháng trời để làm chủ Objective-C; đào tạo Objective C cho iOS đã trở thành một ngành quan trọng; có những công ty chỉ tập trung vào ngôn ngữ lập trình này (và iOS). Do Swift chắc chắn sẽ thay thế hoàn toàn Objective-C, các nhà phát triển ứng dụng iOS sẽ mất một khoảng thời gian đáng kể để làm chủ ngôn ngữ mới của Apple. Điều này có thể làm giảm tốc độ phát triển của hệ sinh thái ứng dụng Apple.

Một mối lo ngại khác có thể kể đến là các ứng dụng rác. Do Swift sẽ giúp quá trình phát triển ứng dụng trở nên dễ dàng hơn trước đây rất nhiều, số lượng ứng dụng chất lượng kém xuất hiện trên gian hàng App Store có thể sẽ ra tăng. Song, đây cũng không hẳn là một mối lo ngại thực sự cho Apple và người dùng của Quả táo. Trải qua giai đoạn chuyển tiếp này, Swift sẽ giúp mang các ứng dụng iOS và Mac lên một tầm cao mới.

## **2.4. Công nghệ Node.js**

### *2.4.1 Giới thiệu chung về Node.js*

**Node.js** là một hệ thống phần mềm được thiết kế để viết các ứng dụng internet có khả năng mở rộng, đặc biệt là máy chủ web. Chương trình được viết bằng JavaScript, sử dụng kỹ thuật điều khiển theo sự kiện, nhập/xuất không đồng bộ để tối tiểu tổng chi phívà tối đại khả năng mở rộng. Node.js bao gồm có V8 JavaScript engine của Google, libUV, và vài thư viện khác.

Node.js được tạo bởi Ryan Dahl từ năm 2009, và phát triển dưới sự bảo trợ của Joyent. Mục tiêu ban đầu của Dahl là làm cho trang web có khả năng push như trong một số ứng dụng web như Gmail. Sau khi thử với vài ngôn ngữ Dahl chọn Javascript vì một API Nhập/Xuất không đầy đủ. Điều này cho phép anh có thể định nghĩa một quy ước Nhập/Xuất điểu khiển theo sự kiện, non-blocking.

Vài môi trường tương tự được viết trong các ngôn ngữ khác bao gồm Twisted cho Python, Perl,ObjectEnvironment cho Perl, libevent cho C và EventMachine cho Ruby. Khác với hầu hết các chương trình Javascript, Nodejs không chạy trên một trình duyệt mà chạy trên Server. Node.js sử dụng nhiều chi tiết kỹ thuật của CommonJS. Nó cung cấp một môi trường REPL cho kiểm thử tương tác.

Node.js được *InfoWorld* bình chọn là "Công nghệ của năm" năm 2012.

Để bắt đầu dùng Node.js, bạn phải hiểu sự khác nhau giữa Node.js với các môi trường   
truyền thống chạy trên server (server side) phổ biến như PHP, Python, Ruby,…

Node.js sử dụng một kiến trúc mô-đun để đơn giản hóa việc tạo ra các ứng dụng phức tạp.   
 Mô-đun giống như các thư viện trong C, hoặc các đơn vị trong Pascal. Mỗi module có chứa một tập hợp các chức năng liên quan đến "đối tượng" của các mô-đun. Ví dụ, các mô-đun http chứa các chức năng cụ thể cho HTTP. Node.js cung cấp một vài mô-đun cơ bản để giúp bạn truy cập các tập tin trên hệ thống tập tin, tạo ra trình điều khiển server HTTP và TCP / UDP và thực hiện các chức năng hữu ích khác. Để gọi một modul thật dễ dàng, chỉ cần gọi hàm require() như sau

Var http = require(‘http’);

Module được đóng gói từng phần mã. Đoạn mã nằm trong một mô-đun chủ yếu là private - có nghĩa là các chức năng và biến được định nghĩa trong họ chỉ có thể truy cập từ bên trong của các mô-đun. Tuy nhiên, bạn có thể tiếp xúc với chức năng và / hoặc các biến được sử dụng từ bên ngoài của mô-đun. Để làm như vậy, phải sử dụng các đối tượng export với các thuộc tính và phương thức của nó với từng phần mã mà bạn muốn gọi từ bên ngoài

Node là một môi trường chạy javascrip với google V8 engine do đó hỗ trợ chạy được ở server side. Do đó bạn cũng nên tuân thủ các kinh nghiệm mà bạn có trong lập trình với các ứng dụng client-side. Ví dụ khi tạo các biến global trong Node không phải lúc nào cũng có thể tạo. Nhưng bạn có thể tạo dễ dàng các biến hoặc hàm global với cách bỏ từ khóa var   
trước các biến

Cộng đồng phát triển Node.js chủ yếu tập trung ở hai nhóm google : nodejs và nodejs-dev, một kênh IRC là #node.js trên mạng freenode. Có một hội thảo về Node.js là NodeConf được tổ chức thường niên.   
 Hiện nay Node.js được sử dụng bởi nhiều công ty trong đó có Linkedin, Microsoft, Yahoo! và Walmart.

### *2.4.2 Ưu điểm của Node.js*

Đầu tiên là ưu điểm về tốc độ thực thi và khả năng mở rộng. Node.js có tốc độ rất nhanh. Đó là một yêu cầu khá quan trọng khi bạn là một startup đang cố gắng tạo ra một sản phẩm lớn và muốn đảm bảo có thể mở rộng nhanh chóng, đáp ứng được một lượng lớn người dùng khi trang web của bạn phát triển lên.

Nếu bạn định viết 1 ứng dụng thể hiện trên 1 trang thì NodeJS rất phù hợp để làm. Với khả năng xử lý nhiều Request đồng thời thời gian phản hồi nhanh. Các ứng dụng bạn định viết không muốn nó tải lại trang, gồm rất nhiều request từ người dùng cần sự hoạt động nhanh để thể hiện sự chuyên nghiệp thì NodeJS sẽ là sự lựa chọn của bạn.

NodeJS sẽ tận dụng tối đa Unix để hoạt động. Tức là NodeJS có thể xử lý hàng nghìn Process và trả ra 1 luồng khiến cho hiệu xuất hoạt động đạt mức tối đa nhất và tuyệt vời nhất.

Các web thông thường gửi HTTP request và nhận phản hồi lại (Luồng dữ liệu). Giả sử sẽ cần xử lý 1 luồng dữ liệu cực lớn, NodeJS sẽ xây dựng các Proxy phân vùng các luồng dữ liệu để đảm bảo tối đa hoạt động cho các luồng dữ liệu khác.

### *2.4.3 Nhược điểm của Node.js*

Nếu cần xử lý các ứng dụng tốn tài nguyên CPU như encoding video, convert file, decoding encryption... hoặc các ứng dụng tương tự như vậy thì không nên dùng NodeJS. Bởi vì NodeJS được viết bằng C++ & Javascript, nên phải thông qua thêm 1 trình biên dịch của NodeJS sẽ lâu hơn 1 chút.

Giống như hầu hết các công nghệ mới, việc triển khai Node.js trên host không phải là điều dễ dàng. Nếu bạn có một web hosting xài chung, bạn không thể đơn giản tải lên một ứng dụng Node.js và mong chờ nó hoạt động tốt. VPS và dedicated server là một sự lựa chọn tốt hơn - bạn có thể cài đặt Node.js trên chúng. Thậm chí dễ hơn là sử dụng một dịch vụ có khả năng mở rộng như là Heroku, và bạn có thể hoàn toàn an tâm để phát triển trang web của mình trên đó - bạn chỉ cần trả tiền khi cần thêm nhiều tài nguyên hơn.

Một nhược điểm lớn khác của Node.js đó là nó vẫn đang trong giai đoạn phát triển ban đầu, điều này có nghĩa là một số đặc trưng sẽ thay đổi trong quá trình phát triển tiếp theo. Trong thực tế, nếu bạn đọc các tài liệu đi kèm, thì nó bao gồm một chỉ số ổn định (stability index), chỉ số này cho thấy mức độ rủi ro khi bạn sử dụng các đặc trưng hiện có.

## **2.5.** **Công nghệ SocketIO**

[SocketIO](http://socket.io/) là một bộ thư viện dành cho các ứng dụng web, mobile realtime. Với đặc trưng mạnh mẽ và dễ sử dụng, **SocketIO** đang dần trở nên quen thuộc với các nhà phát triển (Từ Microsoft Office, Yammer, Zendesk, Trello… tới những đội hackathon, những start up trẻ).

SocketIO cho phép giữ kết nối giữa client và server để cả 2 bên đều luôn lắng nghe được lẫn nhau.

### **2.6. Google Firebase**

  Các chức năng chính của Google Firebase: với Google Firebase, bạn có thể tạo ra các ứng dụng chat như Yahoo Message của ngày xưa hoặc như Facebook Messager của ngày nay trong thời gian cực ngắn như khoảng một ngày thậm chí là vài giờ bởi đơn giản là bạn chỉ cần lo phần client còn phần server và database đã có firebase lo. Firebase là sự kết hợp giữa nền tảng cloud với hệ thống máy chủ cực kì mạnh mẽ tới từ Google, để cung cấp cho chúng ta những API đơn giản, mạnh mẽ và đa nền tảng trong việc quản lý, sử dụng database. Cụ thể hơn Google Firebase cung cấp tới chúng ta những chức năng chính sau:

Cơ sở dữ liệu thời gian thực Firebase lưu trữ dữ liệu database dưới dạng JSON và thực hiện đồng bộ database tới tất cả các client theo thời gian thực. Cụ thể hơn là bạn có thể xây dựng được client đa nền tảng (cross-platform client) và tất cả các client này sẽ cùng sử dụng chung 1 database đến từ Firebase và có thể tự động cập nhật mỗi khi dữ liệu trong database được thêm mới hoặc sửa đổi. Ngoài ra Firebase còn cho phép bạn phân quyền một các đơn giản bằng cú pháp tương tự như javascript.

Hệ thống xác thực của Firebase Với Firebase bạn có thể dễ dàng tích hợp các công nghệ xác thực của Google, Facebook, Twitter, … hoặc một hệ thống xác thực mà bạn tự mình tạo ra vào trong ứng dụng của bạn ở bất kì nền tảng nào như Android, iOS hoặc Web.

Hosting Các bạn có thể triển khai một ứng dụng nền web nhanh chóng với hệ thống Firebase, và các dữ liệu sẽ được lưu trữ đám mây đồng thời được bảo mật thông qua giao thức truy cập SSL. Các ứng dụng sẽ được cấp 1 tên miền dạng \*.firebaseapp.com hoặc bạn có thể trả tiền để sử dụng tên miền của riêng mình.

Những lợi ích từ việc sử dụng Google Firebase Ở phía trên là các chức năng của google firebase, vậy các chức năng đó sẽ đem lại cho bạn những lợi ích gì, có lẽ một số bạn đã mường tượng ra rồi nhưng cũng có bạn có lẽ vẫn còn mơ hồ vì vậy hãy đọc kĩ phần này sẽ biết câu trả lời chính xác nhất.

Triển khai ứng dụng nhanh Với Firebase bạn có thể giảm bớt rất nhiều thời gian cho việc viết các dòng code để quản lý và đồng bộ cơ sở dữ liệu, mọi việc sẽ diễn ra hoàn toàn tự động với các API của Firebase. Không chỉ có vậy Firebase còn hỗ trợ đã nền tảng nên bạn sẽ càng đỡ mất thời gian rất nhiều khi ứng dụng bạn muốn xây dựng là ứng dụng đa nền tảng. Không chỉ nhanh chóng trong việc xây dựng database, Google Firebase còn giúp ta đơn giản hóa quá trình đăng kí và đăng nhập vào ứng dụng bằng các sử dụng hệ thống xác thực do chính Firebase cung cấp.

Bảo mật Firebase hoạt động dựa trên nền tảng cloud và thực hiện kết nối thông qua giao thức bảo mật SSL, chính vì vậy bạn sẽ bớt lo lắng rất nhiều về việc bảo mật của dữ liệu cũng như đường truyền giữa client và server. Không chỉ có vậy, việc cho phép phân quyền người dùng database bằng cú pháp javascipt cũng nâng cao hơn nhiều độ bảo mật cho ứng dụng của bạn, bởi chỉ những user mà bạn cho phép mới có thể có quyền chỉnh sửa cơ sở dữ liệu.

Tính linh hoạt và khả năng mở rộng Sử dụng Firebase sẽ giúp bạn dễ dàng hơn rất nhiều mỗi khi cần nâng cấp hay mở rộng dịch vụ. Ngoài ra firebase còn cho phép bạn tự xây dựng server của riêng mình để bạn có thể thuận tiện hơn trong quá trình quản lý. Việc Firebase sử dụng NoSQL, giúp cho database của bạn sẽ không bị bó buộc trong các bảng và các trường mà bạn có thể tùy ý xây dựng database theo cấu trúc của riêng bạn.

Sự ổn định Firebase hoạt động dựa trên nền tảng cloud đến từ Google vì vậy hầu như bạn không bao giờ phải lo lắng về việc sập server, tấn công mạng như DDOS, tốc độ kết nối lúc nhanh lúc chậm, … nữa, bởi đơn giản là Firebase hoạt động trên hệ thống server của Google. Hơn nữa nhờ hoạt động trên nền tảng Cloud nên việc nâng cấp, bảo trì server cũng diễn ra rất đơn giản  mà không cần phải dừng server để nâng cấp như truyền thống.

Giá thành Google Firebase có rất nhiều gói dịch vụ với các mức dung lượng lưu trữ cũng như băng thông khác nhau với mức giá dao động từ Free đến $500 đủ để đáp ứng được nhu cầu của tất cả các đối tượng. Chính vì vậy bạn có thể lựa chọn gói dịch vụ phù hợp nhất với nhu cầu của mình. Điều này giúp bạn tới ưu hóa được vốn đầu tư và vận hành của mình tùy theo số lượng người sử dụng. Ngoài ra bạn còn không mất chi phí để bảo trì, nâng cấp, khắc phục các sự cố bởi vì những điều này đã có Firebase lo.

## **2.7 Giới thiệu về NoSQL MongoDB**

### *2.7.1 Giới thiệu về NoSQL:*

Với hầu hết các thời kỳ web, Hệ quản trị cơ sở dữ liệu quan hệ dựa trên SQL đã thống trị hầu hết các hệ Quản trị Cơ sở dữ liệu. Tuy nhiên, thời gian gần đây, một cách tiếp cận mới đã bắt đầu biết đến là NoSQL, tạo ra sự thay thế cho các hệ quản trị cơ sở dữ liệu quan hệ truyền thống.

NoSQL còn có nghĩa là Non-Relational - không ràng buộc. Tuy nhiên, thuật ngữ đó ít phổ dụng hơn và ngày nay người ta thường dịch NoSQL thành Not Only SQL - Không chỉ SQL. NoSQL ám chỉ đến những cơ sở dữ liệu không dùng mô hình dữ liệu quan hệ để quản lý dữ liệu trong lĩnh vực phần mềm

Thuật ngữ NoSQL được giới thiệu lần đầu vào năm 1998 sử dụng làm tên gọi chung cho các cơ sở dữ liệu quan hệ nguồn mở nhỏ nhưng không sử dụng SQL cho truy vấn.

Vào năm 2009, Eric Evans, nhân viên của Rackspace giới thiệu lại thuật ngữ NoSQL khi Johan Oskarsson của Last.fm muốn tổ chức một hội thảo về cơ sở dữ liệu nguồn mở phân tán. Thuật ngữ NoSQL đánh dấu bước phát triển của thế hệ CSDL mới: một thế hệ CSDL không ràng buộc, phân tán, nguồn mở, khả năng mở rộng theo chiều ngang, có thể lưu trữ, xử lý từ một lượng rất nhỏ cho tới hàng petabytes dữ liệu trong hệ thống có độ chịu tải, chịu lỗi cao với những đòi hỏi về tài nguyên phần cứng thấp.

Một số đặc điểm nhận dạng cho thế hệ CSDL mới này bao gồm: schema-free, hỗ trợ mở rộng dễ dàng, API đơn giản, nhất quán cuối (eventual consistency), không giới hạn không gian dữ liệu,...

Sau đây là danh sách các CSDL NoSQL:

1. Wide Column Store / Column Families: Hadoop/HBase  –  Apache,  BigTable  –  Google,  Cassandra  - Facebook/Apache, Hypertable - Zvents Inc/Baidu, Cloudera, SciDB, Mnesia, Tablets,…

2. Key-Value Store/Tuple store

a. Key/value cache in RAM: memcached, Citrusleaf database, Velocity, Redis, Tuple space,...

b. Key/value save on disk: Memcachedb, Berkeley DB, Tokyo Cabinet, Redis,...

c. Eventually Consistent Key Value Store: Amazon Dynamo, Voldemort, Dynomite, KAI, Cassandra, Hibari, Project Voldemort,…

d. Ordered key-value store: NMDB, Memcachedb, Berkeley DB,...

e. Distributed systems: Apache River, MEMBASE, Azure Table Storage, Amazon Dynamo,...

3. Document Store: Apache Jackrabbit, CouchDB, IBM Lotus Notes Storage Format (NSF),  MongoDB,  Terrastore,  ThruDB, OrientDB, RavenDB,...

4. Graph Database: Neo4J, Sones, AllegroGraph, Core Data, DEX, FlockDB, InfoGrid, OpenLink Virtuoso,...

Tuy cùng mang những đặc điểm chung của NoSQL nhưng mỗi CSDL NoSQL cũng có những đặc điểm riêng, và vì thế thường được dùng cho những dự án khác nhau. Ví dụ:

MongoDB và Redis là những lựa chọn tốt cho việc lưu trữ các dữ liệu thống kê ít được đọc mà lại được viết thường xuyên.

Hadoop, một CSDL dạng tự do, phân tán làm tốt công việc lưu trữ các dữ liệu lớn như các con số thống kê thời tiết hoặc công việc phân tích nghiệp vụ.

Memcachedb, một CSDL nhất thời chóng tàn, tuyệt vời trong lưu trữ các phiên làm việc web, các khóa, và các con số thống kê ngắn hạn.

Cassandra và Riak (các lưu trữ dư thừa, tự động tạo bó cluster) làm tốt trong các môi trường với các ứng dụng có tính sẵn sàng cao, khi thời gian sống tối đa là sống còn.

Để tìm hiểu sâu hơn về các CSDL hiện đại NoSQL, chúng ta đi nghiên cứu chi tiết CSDL đặc trưng là MongoDB.

### *2.7.2 Giới thiệu về MongoDB:*

Trong những gương mặt góp phần làm suy tàn đế chế SQL thì MongoDB nổi lên là một CSDL đáng tin cậy và dễ dùng nhất. Mongo viết bằng C++. Nó thích hợp cho các ứng dụng tầm trung trở lên. Nếu tỉ lệ lượng dữ liệu ghi vào CSDL của ứng dụng lớn hơn lượng đọc thì đây càng là lựa chọn hợp lý.

MongoDB là một CSDL có khả năng mở rộng, hiệu suất cao, mã nguồn mở và hướng văn bản.

Trước khi đi vào tìm hiểu kỹ hơn về MongoDB, chúng ta làm quen với một số khái niệm cơ bản của MongoDB:

* Văn bản (Document) là đơn vị cơ bản của dữ liệu trong MongoDB, nó tương đương với một dòng trong CSDL quan hệ
* Bộ sưu tập (Collection) có thể được coi như tương đương với một bảng.
* MongoDB có thể lưu trữ nhiều CSDL độc lập, mỗi CSDL này có các bộ sưu tập và điều khoản riêng của mình
* MongoDB đi kèm với một trình tiện ích JavaScript đơn giản nhưng mạnh mẽ, nó hữu ích trong quản trị và thao tác dữ liệu.
* Mỗi văn bản có một khóa đặc biệt, đó là “\_id”, nó là duy nhất trong bộ sưu tập của văn bản.

**Văn bản**

Văn bản là một khái niệm quan trọng trong MongoDB. Văn bản bao gồm tập hợp các khóa với các giá trị tương ứng.

Ví dụ: {"greeting" : "Hello, world!"}

Văn bản trên gồm một khóa là “greeting”, với giá trị là “Hello, world!”. Các văn bản có thể chứa nhiều cặp khóa/giá trị.

Ví dụ: {"greeting" : "Hello, world!", "foo" : 3}

Một số lưu ý:

* Các cặp khóa/ giá trị trong văn bản được sắp xếp. Văn bản trên sẽ khác với văn bản sau

{"foo" : 3, "greeting" : "Hello, world!"}

* Khóa trong văn bản là một chuỗi
* MongoDB phân biệt chữ hoa chữ thường
* Văn bản trong MongoDB không được chứa những khóa giống nhau. Ví dụ văn bản sau là không hợp lệ

{"greeting" : "Hello, world!", "greeting" : "Hello, MongoDB!"}

**Bộ sưu tập**

Bộ sưu tập là một nhóm các văn bản. Nếu văn bản tương đương với dòng trong CSDL quan hệ thì bộ sưu tập tương đương với bảng.

Bộ sưu tập là một Schema-Free, nghĩa là các văn bản có hình dạng khác nhau có thể cùng được lưu trữ trong 1 bộ sưu tập.

Ví dụ các văn bản sau có thể cùng được lưu trong một bộ sưu tập:

{"greeting" : "Hello, world!"}

{"foo" : 5}

Bộ sưu tập được xác định bởi tên của nó là một chuỗi UTF-8

Các đặc trưng của MongoDB:

* Lưu trữ hướng văn bản: Văn bản theo phong cách JSON với những lược đồ động đơn giản
* Hỗ trợ chỉ mục đầy đủ: chỉ mục trên bất kỳ các thuộc tính
* Tính sao lặp và tính sẵn sàng cao: mở rộng
* Auto-sharding: mở rộng theo chiều ngang mà không ảnh hưởng đến chức năng
* Truy vấn: đa dạng, truy vấn dựa trên văn bản
* Cập nhật nhanh:
* Map/Reduce
* GridFS: lưu trữ file với bất kỳ kích cỡ nào mà không làm phức tạp ngăn xếp
* Hỗ trợ thương mại: hỗ trợ doanh nghiệp, đào tào, tư vấn

Collection là một nhóm các văn bản. Nếu văn bản tương đương với dòng trong CSDL quan hệ thì bộ sưu tập tương đương với bảng.

Collection là một Schema-Free, nghĩa là các văn bản có hình dạng khác nhau có thể cùng được lưu trữ trong 1 bộ sưu tập.

Ví dụ các văn bản sau có thể cùng được lưu trong một bộ sưu tập:

{"greeting" : "Hello, world!"}

{"foo" : 5}

Collection được xác định bởi tên của nó là một chuỗi UTF-8

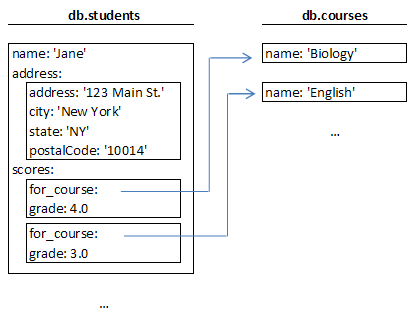
Các đặc trưng của MongoDB:

* Lưu trữ hướng văn bản: Văn bản theo phong cách JSON với những lược đồ động đơn giản
* Hỗ trợ chỉ mục đầy đủ: chỉ mục trên bất kỳ các thuộc tính
* Tính sao lặp và tính sẵn sàng cao: mở rộng
* Auto-sharding: mở rộng theo chiều ngang mà không ảnh hưởng đến chức năng
* Truy vấn: đa dạng, truy vấn dựa trên văn bản
* Cập nhật nhanh:
* Map/Reduce
* GridFS: lưu trữ file với bất kỳ kích cỡ nào mà không làm phức tạp ngăn xếp
* Hỗ trợ thương mại: hỗ trợ doanh nghiệp, đào tào, tư vấn.

Với MongoDB, chúng ta ít phải “chuẩn hóa” hơn so với khi làm việc với lược đồ quan hệ vì trong MongoDB không có khái niệm liên kết (join). Nói chung, với mỗi đối tượng (object) mức cao nhất, ta sẽ có một bộ sưu tập (collection) dữ liệu.

Một bộ sưu tập không phải cho tất cả các lớp (class), thay vào đó, các đối tượng sẽ được nhúng vào đó.

Hình 2.1 minh họa có 2 bộ sưu tập: students và courses. Các văn bản student được nhúng văn bản address và văn bản score. Trong đó, văn bản Score được tham chiếu đến Courses.



Hình 2.2. Minh họa bộ sưu tập

So sánh với lược đồ quan hệ: ta cần lưu Score vào bảng riêng và dùng khóa ngoài liên kết với Student.

Document trong MongoDB có cấu trúc tương tự như kiểu dữ liệu JSON, nghĩa là sẽ có các cặp (key => giá trị) nên nó có tính năng động rất lớn. Document ta có thể hiểu nó giống như các record dữ liệu trong MYSQL, tuy nhiên nó có sự khác biệt là các cặp (key => value) có thể không giống nhau ở mỗi document.

### *2.7.3 Ưu điểm của MongoDB*

Open source: các sản phẩm nguồn mở đưa ra cho những người phát triển với nhiều lợi ích to lớn, đặc biết là việc sử dụng miễn phí. Những lợi ích khác là phần mềm nguồn mở có xu hướng sẽ là tin cậy hơn, an ninh hơn và nhanh hơn để triển khai so với các lựa chọn thay thế sở hữu độc quyền.

Khả năng mở rộng linh hoạt: NoSQL thay thế câu thần chú cũ của các nhà quản trị CSDL về “mở rộng phạm vi” với một thứ mới “mở rộng ra ngoài”. Thay vì bổ sung thêm các máy chủ lớn hơn để điều khiển nhiều tải dữ liệu hơn, thì CSDL NoSQL cho phép một công ty phân tán tải qua nhiều máy chủ khi mà tải gia tăng.

Các CSDL NoSQL khác nhau cho những dự án khác nhau:MongoDB và Redis là những lựa chọn tốt cho việc lưu trữ các dữ liệu thống kê ít được đọc mà lại được viết thường xuyên, như một số đếm truy cập web chẳng hạn. Hadoop, một CSDL dạng tự do, phân tán làm tốt công việc lưu trữ các dữ liệu lớn như các con số thống kê thời tiết hoặc công việc phân tích nghiệp vụ.

NoSQL được các hãng lớn sử dụng: các công ty như Amazon, BBC, Facebook và Google dựa vào các CSDL NoSQL.

NoSQL phù hợp với công nghệ đám mây: NoSQL và đám mây là một sự trùng khớp tự nhiên. Các máy chủ ngày nay là không đắt và có thể dễ dàng mở rộng phạm vi được theo yêu cầu có sử dụng một dịch vụ như là Amazon EC2. Giống như tất cả công nghệ đám mây, EC2 dựa vào ảo hóa. Liên kết yếu của ảo hóa là sự thực thi của I/O, với bộ nhớ và CPU các các kết nối mạnh.

Các CSDL NoSQL hầu hết sử dụng bộ nhớ qua đĩa như là vị trí ghi đầu tiên. Vì thế ngăn ngừa được sự thực thi không ổn định của I/O. Và vì NoSQL lưu trữ dữ liệu thường thúc đẩy được tính mở rộng phạm vi theo chiều ngang thông qua việc ngăn chia, chúng có khả năng tận dụng được việc cung cấp mềm dẻo của đám mây.

#### 2.7.4 Khuyết điểm của MongoDB

Hỗ trợ không đồng đều cho các doanh nghiệp: trong khi các nhà cung cấp chủ chốt của các RMDBMS như SQL Server, Oracle, IBM,… thường đưa ra sự hỗ trợ tốt cho khách hàng thì các nhà cung cấp nguồn mở mới thành lập không thể được mong đợi sẽ cung cấp hỗ tốt hơn.

Chưa đủ “chín” cho các doanh nghiệp: dù chúng đã được triển khai tại một số công ty lớn thì các CSDL NoSQL vẫn đối mặt với một vấn đề về sự tin cậy chính với nhiều doanh nghiệp. Vấn đề lớn của NoSQL là thiếu về độ chín muồi và các vấn đề về tính không ổn định, trong khi đó tính chín muồi, hỗ trợ đầy đủ chức năng và tính ổn định của các RDBMS được thiết lập đã từ lâu.

Những hạn chế về tri thức nghiệp vụ: các CSDL NoSQL không có nhiều sự đeo bám tới các công cụ BI thường được sử dụng, trong khi những yêu cầu và phân tích hiện đại đơn giản nhất thì cũng liên quan khá nhiều tới sự tinh thông về lập trình.

Thiếu sự tinh thông: tính mới mẻ của NoSQL có nghĩa là không có nhiều lập trình viên và người quản trị biết công nghệ này. Như vậy sẽ rất khó khăn cho các công ty tìm người với sự tinh thông phù hợp.

Những vấn đề về tính tương thích: không giống như các CSDL quan hệ, các CSDL NoSQL chia sẻ ít theo cách thức của các tiêu chuẩn. Mỗi CSDL NoSQL có các giao diện lập trình ứng dụng API riêng của mình, các giao diện truy vấn riêng,… Sự thiếu hụt các tiêu chuẩn sẽ gây ra rất nhiều khó khăn khi chuyển từ một nhà cung cấp này sang một nhà cung cấp khác nếu có nhu cầu.

**Chương III: Phân tích thiết kế hệ thống & Cơ sở dữ liệu**

**3.1 Xác định các Usecase.**

*3.1.1. Use tổng quan*

Đầu tiên khi xây dựng Use case cho actor Member:

Xác định chức năng có trong use case:

- Quản lý tài khoản:

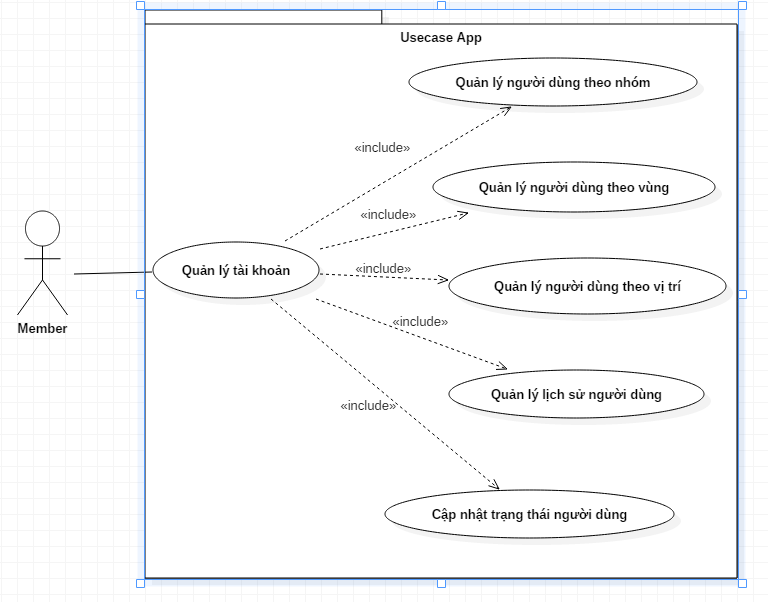
+ Quản lý người dùng theo nhóm

+ Quản lý người dùng theo vùng

+ Quản lý người dùng theo vị trí

+ Quản lý lịch sử người dùng

+ Cập nhật trạng thái người dùng

****

Hình 3.1 Biểu đồ Usecase tổng quan

*3.1.2 Usecase quản lí người dùng theo nhóm*

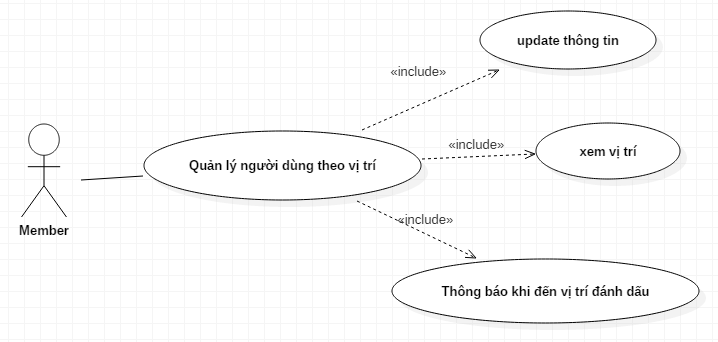
Xác định chức năng có trong use case:

* + Quản lý người dùng theo vị trí

+ Cập nhật thông tin người dùng liên tục

+ Xem vị trí người dùng

+ Thông báo cho mọi người trong vòng khi đến vị trí đặt trước

****

Hình 3.2. Biểu đồ usecase quản lí người dùng theo nhóm

*3.1.3 Usecase quản lí người dùng theo nhóm*

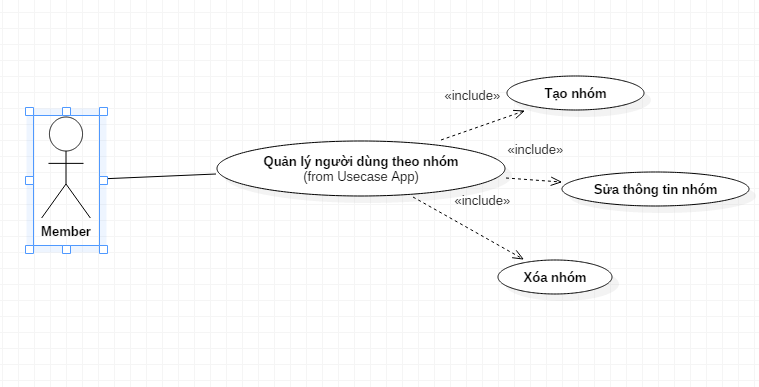
Xác định chức năng có trong use case:

* + Quản lý người dùng theo nhóm

+ Tạo nhóm người dùng

+ Tham gia nhóm

+ Sửa nhóm

****

Hình 3.3 Usecase quản lí người dùng theo nhóm

*3.1.4 Quản lí tài khoản người dùng*

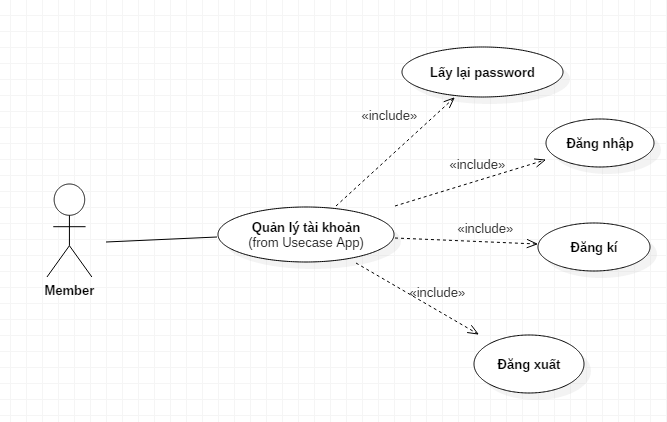
Xác định chức năng có trong use case:

* + Quản lý tài khoản

+ Lấy lại password

+ Đăng nhập

+ Đăng kí

****

Hình 3.4.Biểu đồ usecase quản lí tài khoản người dùng

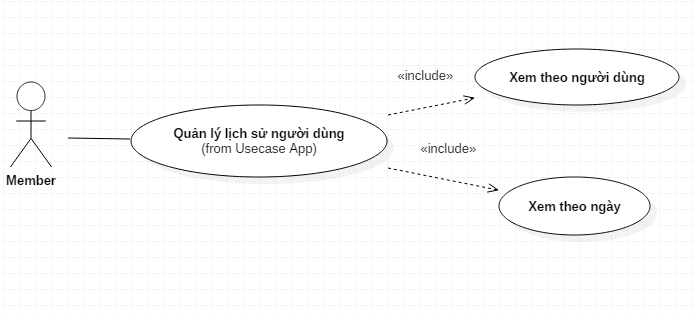
*3.1.5 Quản lí lịch sử người dùng*

Xác định chức năng có trong use case:

* + Quản lý lịch sử người dùng

+ Xem lịch sử của từng người dùng

+ Xem lịch sử theo ngày

****

Hình 3.5.Biểu đồ usecase quản lí lịch sử người dùng

*3.1.6 Quản lý người dùng theo vùng*

Xác định chức năng có trong use case:

* + Quản lý người dùng theo vùng

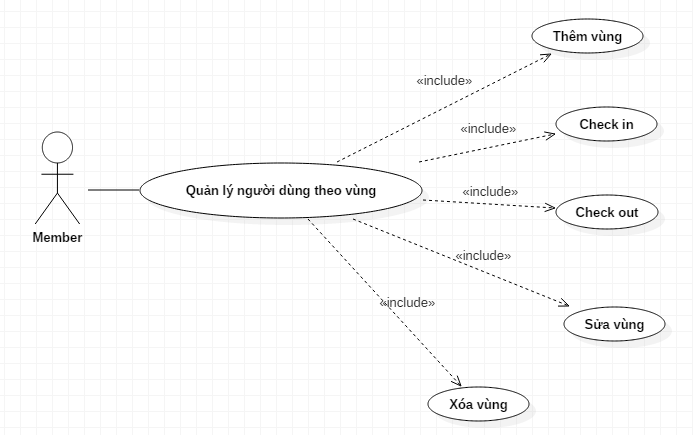
+ Thêm vùng

+ Sửa vùng

+ Xóa vùng

+ Thông báo khi có người vào vùng

+ Thông báo khi có người ra khỏi vùng

****

Hình 3.6. Biểu đồ usecase quản lý người dùng theo vùng

**3.2 Cơ sở dữ liệu offline**

*3.2.1: Bảng thành viên*

Bảng thành viên chứa thông tin các thành viên.

|  |  |  |
| --- | --- | --- |
| Trường | Kiểu dữ liệu | Mô tả |
| \_id | Number | Id của member |
| updatedAt | Date | Ngày cập nhật |
| createdAt | Date | Ngày tạo |
| phone | String | Số điện thoại |
| email | String | Email |
| name | String | Tên thành viên |
| avatar | String | Link avatar |
| Longitude | Double | Kinh độ |
| Latitude | Double | Vĩ độ |
| battery | Int | Phần trăm pin |
| Speed | Int | Tốc độ |
| CheckIn | String | Mô tả địa điểm checkIn |
| SOS | String | Mô tả địa điểm SOS |
| isCharging | Bool | Trạng thái sạc |
| IdCircle | String | Id Của vòng |
| Role | String | Id của chủ vòng |
| LastUpdate | Int | Thời gian lần cuối cập nhật |

Bảng 2.1 Bảng thành viên

### *3.2.2 Bảng User*

Bảng user chứa thông tin của người dùng.

|  |  |  |
| --- | --- | --- |
| Trường | Kiểu dữ liệu | Mô tả |
| \_id | String | Id của diễn viên |
| updatedAt | Date | Ngày cập nhật |
| createdAt | Date | Ngày tạo |
| Name | String | Tên diễn viên |
| avart | String | Link Avatar |
| Email | String | Email |

Bảng 2.2 Bảng user

### *3.2.3 Bảng địa điểm*

Bảng địa điểm chứa thông tin của các địa điểm.

|  |  |  |
| --- | --- | --- |
| Trường | Kiểu dữ liệu | Mô tả |
| \_id | String | Id của thể loại |
| updatedAt | Date | Ngày cập nhật |
| createdAt | Date | Ngày tạo |
| Name | String | Tên thể loại |
| idCircle | String | Id của vòng |
| radius | Int | Bán kính của địa điểm |
| Longitude | Double | Kinh độ |
| Latitude | Double | Vĩ độ |
| typeArea | Int | Loại địa điểm |
| Description | String | Địa chỉ địa điểm |
| Owner | String | Chủ vòng |

Bảng 2.3 Bảng địa điểm

### *3.2.4 Bảng vòng*

Bảng vòng chứa thông tin của các vòng.

|  |  |  |
| --- | --- | --- |
| Trường | Kiểu dữ liệu | Mô tả |
| \_id | String | Id của quốc gia |
| updatedAt | Date | Ngày cập nhật |
| createdAt | Date | Ngày tạo |
| Name | String | Tên quốc gia |
| Owner | String | Chủ vòng |
| Type | String | Loại vòng |
| Code | String | Mã của vòng |
| CreateCode | Int | Thời gian tạo mã |
| avartar | String | Link avatar |
| MembersCount | Int | Số thành viên |

Bảng 2.4 Bảng vòng

# Chương IV: Ứng dụng định vị gia đình

* 1. **Các chức năng chính:**

*4.1.1 Tạo tài khoản:*

Người dùng tạo tài khoản từ các thông tin cơ bản như: số điện thoại, email, tên,…

Tài khoản sẽ được lưu lại cho các lần đăng nhập tiếp theo

*4.1.2 Đăng nhập*

Người dùng đăng nhập bằng tài khoản đã tạo. Sever sẽ trả về thông tin của tài khoản như: Id, tên người dung, token, lần cuối cập nhật, email, số điện thoại,…. Sau đó ứng dụng sẽ vào màn hình chính.

*4.1.3 Quên mật khẩu:*

Người dùng khi quên mật khẩu sẽ nhập số điện thoại, sau đó người dung phải xác nhận số điện thoại bằng ứng dụng Acount Kit của facebook. Người dùng nhập mật khẩu mới và đăng nhập lại.

*4.1.4 Tự động lưu trạng thái người dùng*

Khi người dùng đã đăng nhập vào hệ thống thì các lần mở ứng dụng tiếp theo sẽ không phải đăng nhập lại. Ứng dụng sẽ lưu lại tất cả và thực hiện tự động cho người dung.

*4.1.5 Tạo vòng kết nối*

Người dùng có thể tạo vòng kết nối bằng cách nhập tên vòng, chọn avatar ( không bắt buộc ), Sau đó vòng sẽ được tạo. Người dung có thể mời các người khác vào vòng đó để biết được tình trạng của nhau.

*4.1.6 Mời vào vòng*

Người dùng có thể lấy mã code của vòng gồm 6 kí tự. Ứng dụng có chức năng chia sẻ mã code đó qua các ứng dụng có sẵn trong máy. Khi người khác nhận được mã code đó có thể nhập và xin tham gia vòng của mình.

*4.1.7 Xin vào vòng*

Khi người dùng nhập mã code của vòng nào đó ứng dụng sẽ hiện lên chi tiết của vòng để xem lại có đúng vòng định tham gia không. Người dùng có thể bấm tham gia vòng hoặc hủy yêu cầu.

*4.1.8 Lấy danh sách tham gia vòng của mình*

Người dùng có thể xem danh sách mọi người đã yêu cầu tham gia vòng của mình. Có 2 sự lựa chọn là đồng ý cho vào vòng hoặc hủy yêu cầu đó.

*4.1.9 Tạo địa điểm*

Người dùng có thể tạo một địa điểm trên bản đồ bằng cách gửi lên địa chỉ (tự động lấy), tên địa điểm, bán kính của địa điểm. Ứng dụng sẽ lưu lại địa điểm đó để dùng cho các chức năng sau này.

*4.1.10 Sửa địa điểm*

Người dùng có thể sửa một địa điểm trên bản đồ bằng cách gửi lên địa chỉ (tự động lấy), tên địa điểm, bán kính của địa điểm. Ứng dụng sẽ lưu lại địa điểm đó để dùng cho các chức năng sau này.

*4.1.11 Tạo địa điểm*

Người dùng có thể xóa một địa điểm trên bản đồ bằng cách gửi lên id của địa điểm đó.

*4.1.12 Xem chi tiết địa điểm*

Người dùng có thể xem danh sách các địa điểm đã được tạo, chi tiết của các địa điểm đó bằng cách bấm vào từng địa điểm riêng biệt.

*4.1.13 CheckIn*

Người dùng có thể checkIn các địa điểm xung quanh nơi mình đang ở để thông báo với mọi người rằng mình đang ở gần đó.

*4.1.14 SOS*

Người dùng có thể chọn chức năng SOS thể thông báo tới tất cả người trong vòng rằng mình đang gặp nguy hiểm. thông báo này sẽ được ưu tiên hiển thị.

*4.1.15 Theo dõi tổng quan các thành viên*

Người dùng có thể theo dõi trực quan các thành viên trên bản đồ Apple Map. Vị trí của từng người trên bản đồ sẽ được cập nhật theo thời gian thực.

*4.1.16 Danh sách chi tiết các thành viên*

Người dùng có thể xem danh sách chi tiết các thành viên gồm avatar, tên thành viên, địa chỉ hiện tại của họ, phần trăm pin điện thoại, khoảng cách từ mình tới vị trí của họ, lần cuối cập nhật là bao giờ.

*4.1.17 Chi tiết thành viên*

Người dùng có thể xem chi tiết của 1 thành viên bao gồm gồm avatar, tên thành viên, địa chỉ hiện tại của họ, phần trăm pin điện thoại, khoảng cách từ mình tới vị trí của họ, lần cuối cập nhật cách đây bao lâu, vị trí của họ trên bản đồ.

*4.1.18 Lịch sử di chuyển*

Các thành viên trong vòng có thể xem lịch sử di chuyển của các thành viên khác theo từng ngày bao gồm: lịch sử checkin, lịch sử SOS, lịch sử ra địa điểm, lịch sử vao địa điểm, lịch sử di chuyển.

*4.1.19 Thông báo*

Các thông báo gồm:

* Chủ vòng tạo địa điểm.
* Chủ vòng sửa địa điểm.
* Chủ vòng xóa địa điểm.
* Chủ vòng đồng ý cho vào vòng.
* Có thêm thành viên vào vòng.
* Thông báo SOS.
* Thông báo checkIn.
* Thông báo thành viên vào địa điểm.
* Thông báo thành viên ra khỏi địa điểm.
* Thông báo đã được cho vào vòng.

## **4.2. Giao diện chương trình**

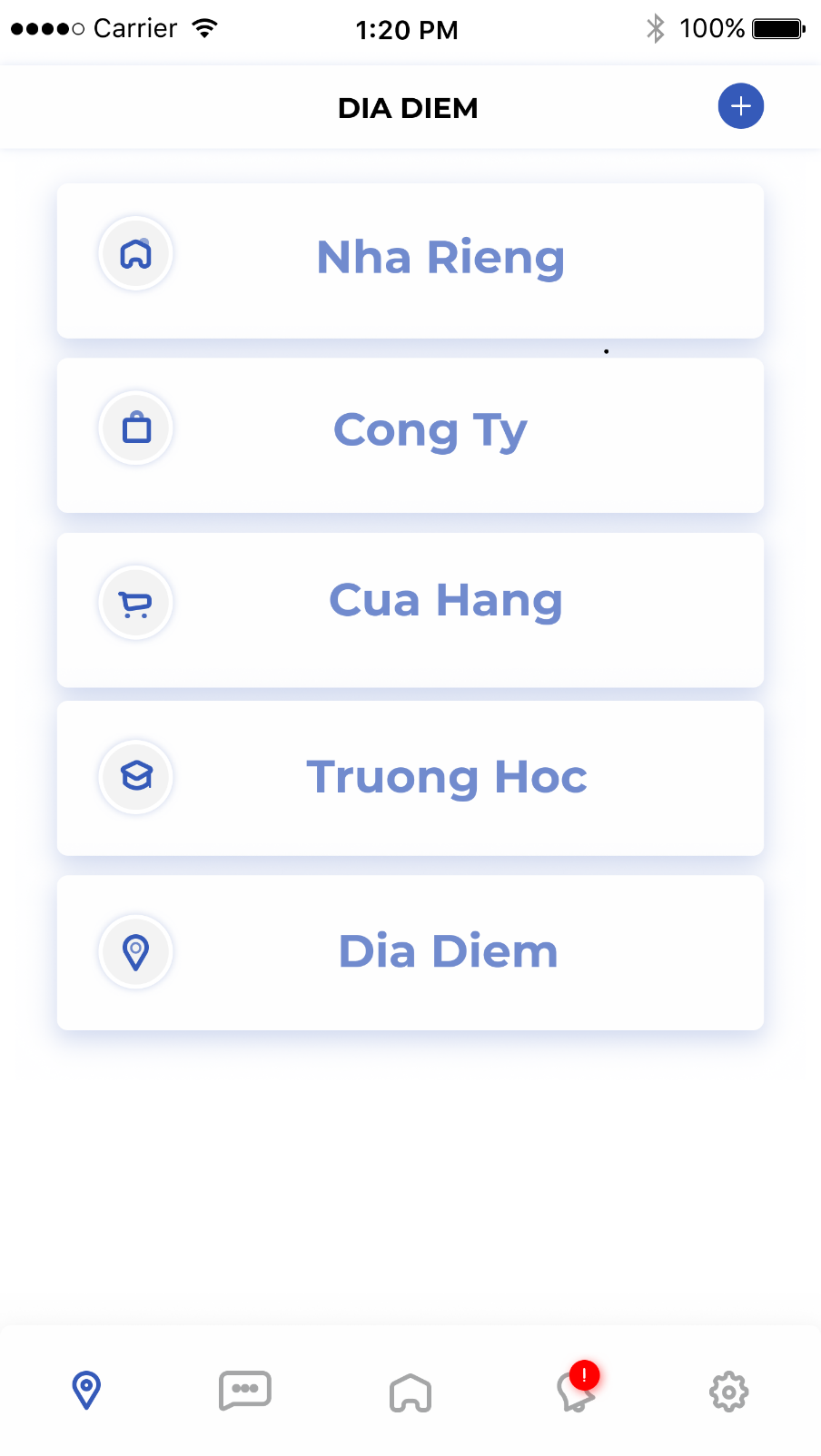
*4.2.1: Màn hình home*



Hình 4.1 Màn hình home

Màn hình home hiển thị vị trí mọi người, các địa điểm trong vòng trên bản đồ trực quan. Tên vòng, số thành viên trong vòng, các chức năng chính như SOS, CheckIn, thêm thành viên, danh sách thành viên, Location.

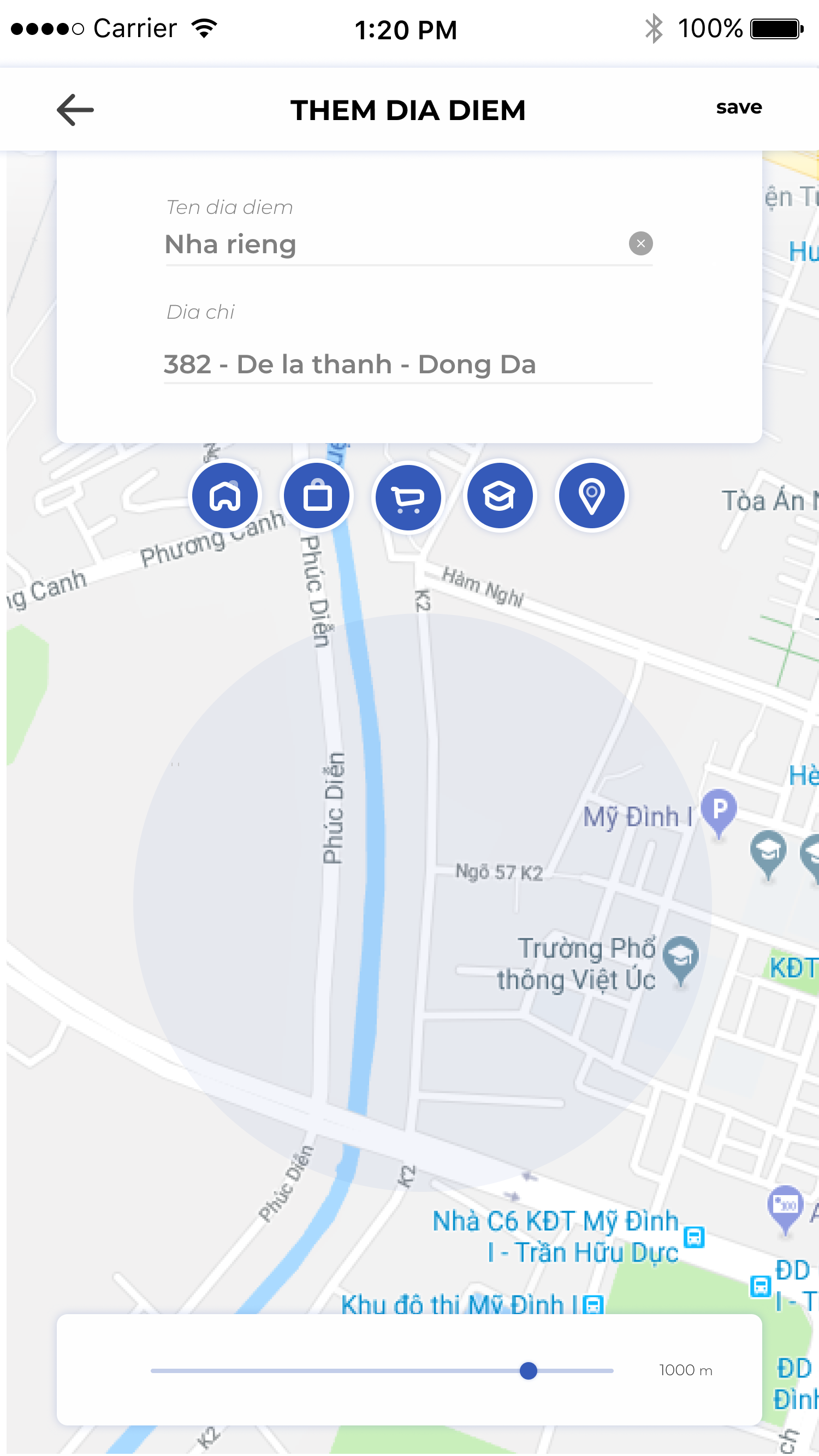
### *4.1.2 Màn hình danh sách địa điểm*



Hình 4.2 Màn hình danh sách địa điểm

Màn hình danh sách địa điểm hiển thị danh sách gồm tên địa điểm, Avatar địa điểm.

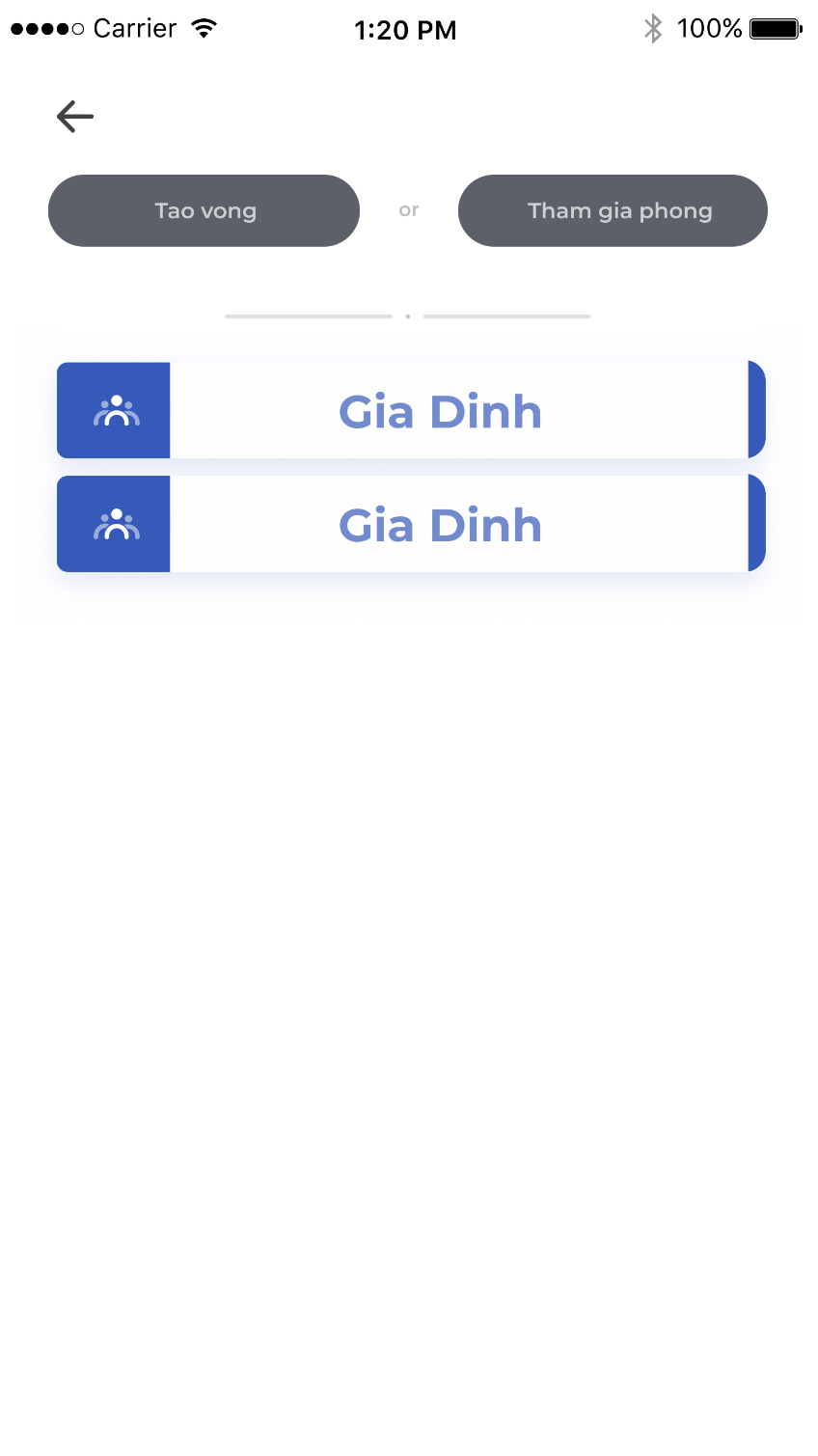
### *4.1.3 Màn hình thêm địa điểm*



Hình 4.3 Màn hình thêm địa điểm

Màn hình thêm địa điểm gồm nhập tên địa điểm, chọn bán kính địa điểm, chọn loại địa điểm.

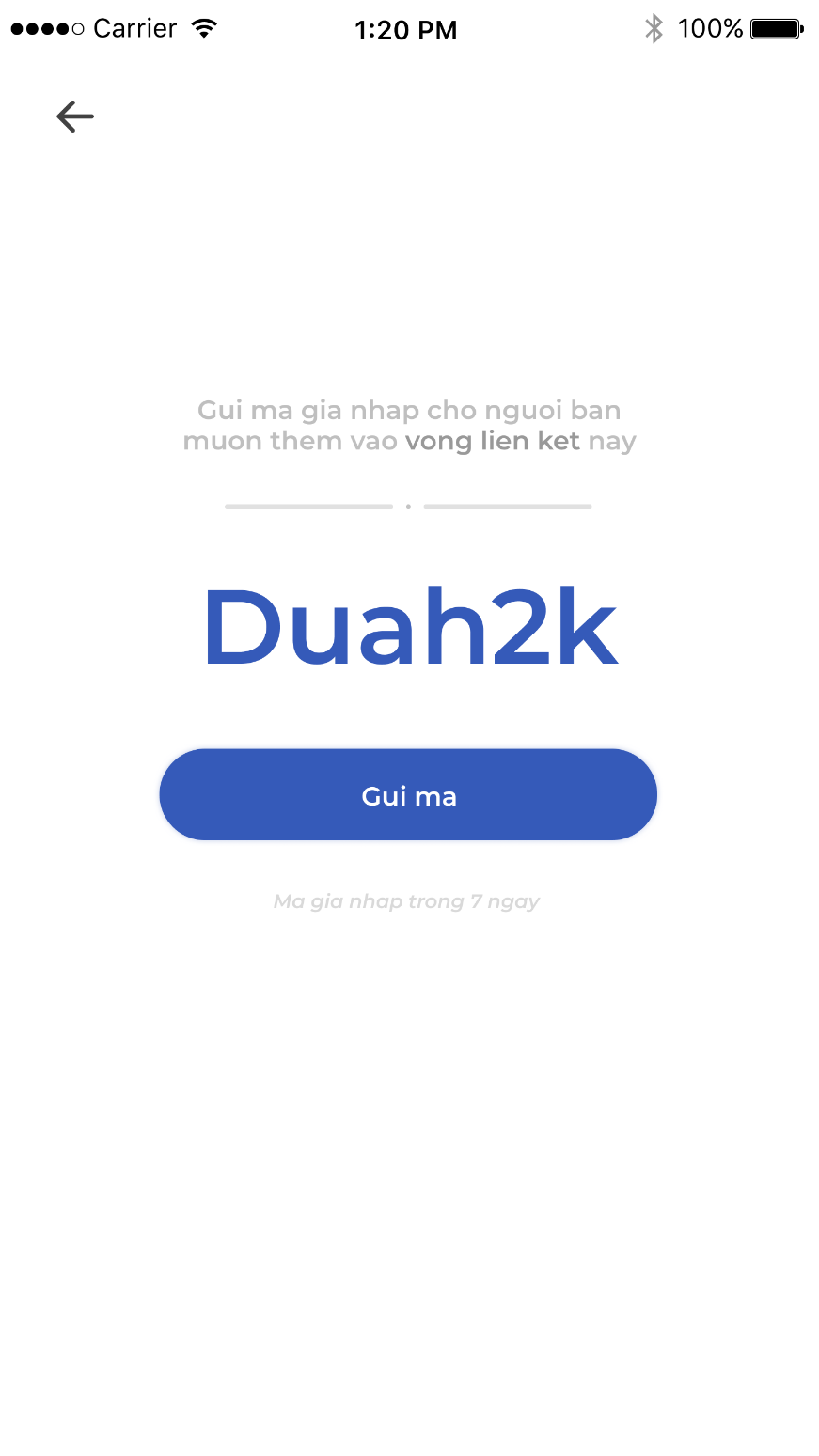
### *4.1.4 Màn hình vòng*



Hình 4.4 Màn hình danh sách vòng

Màn hình gồm danh sách các vòng đã tạo, chức năng tạo vòng và tham gia vòng.

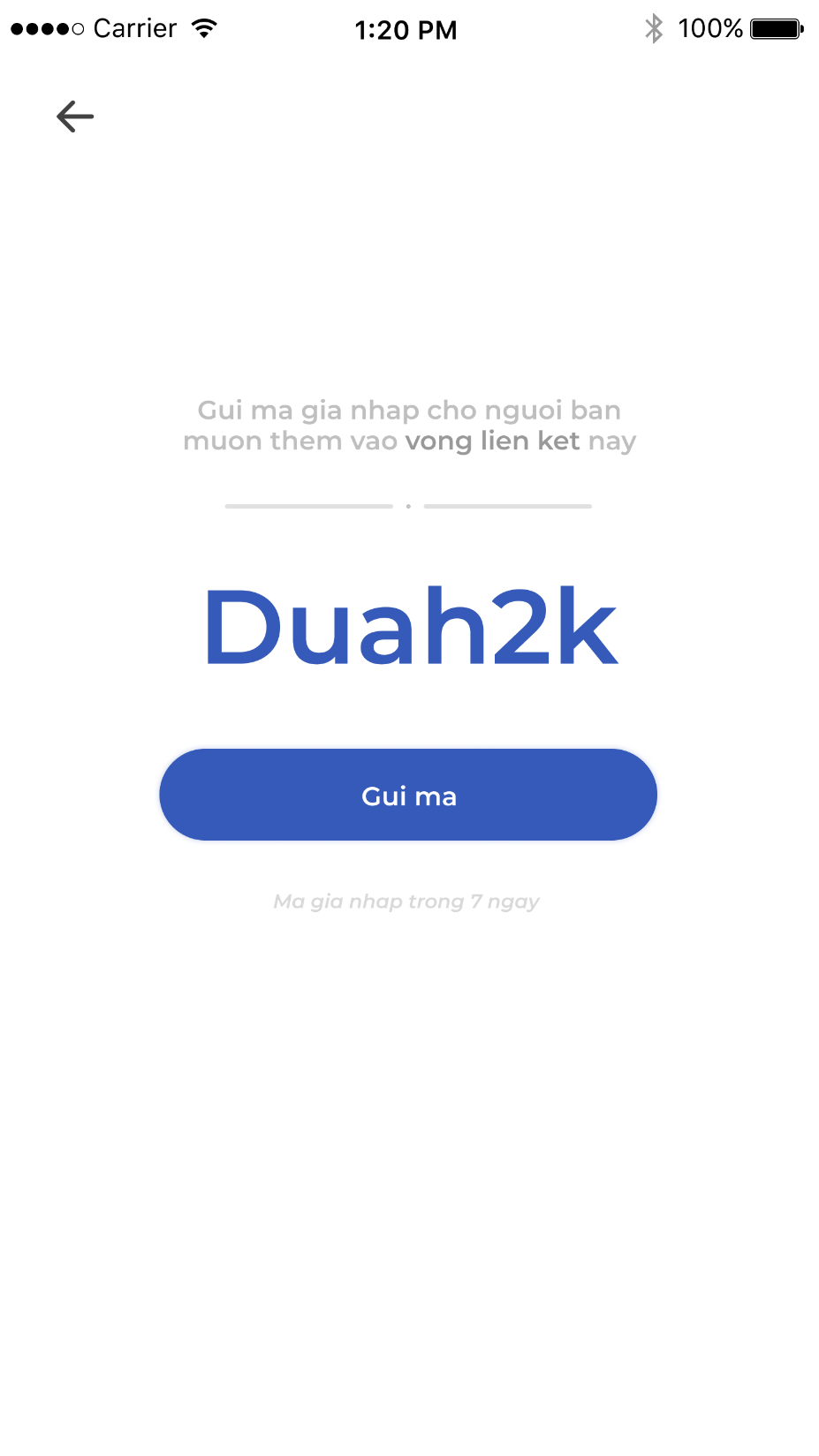
### *4.1.5 Màn hình lấy mã code*



Hình 4.5 Màn hình lấy code của vòng

Màn hình sẽ hiển thị mã code của vòng, có chức năng gửi mã vòng bằng các ứng dụng khác có sẵn trong máy.

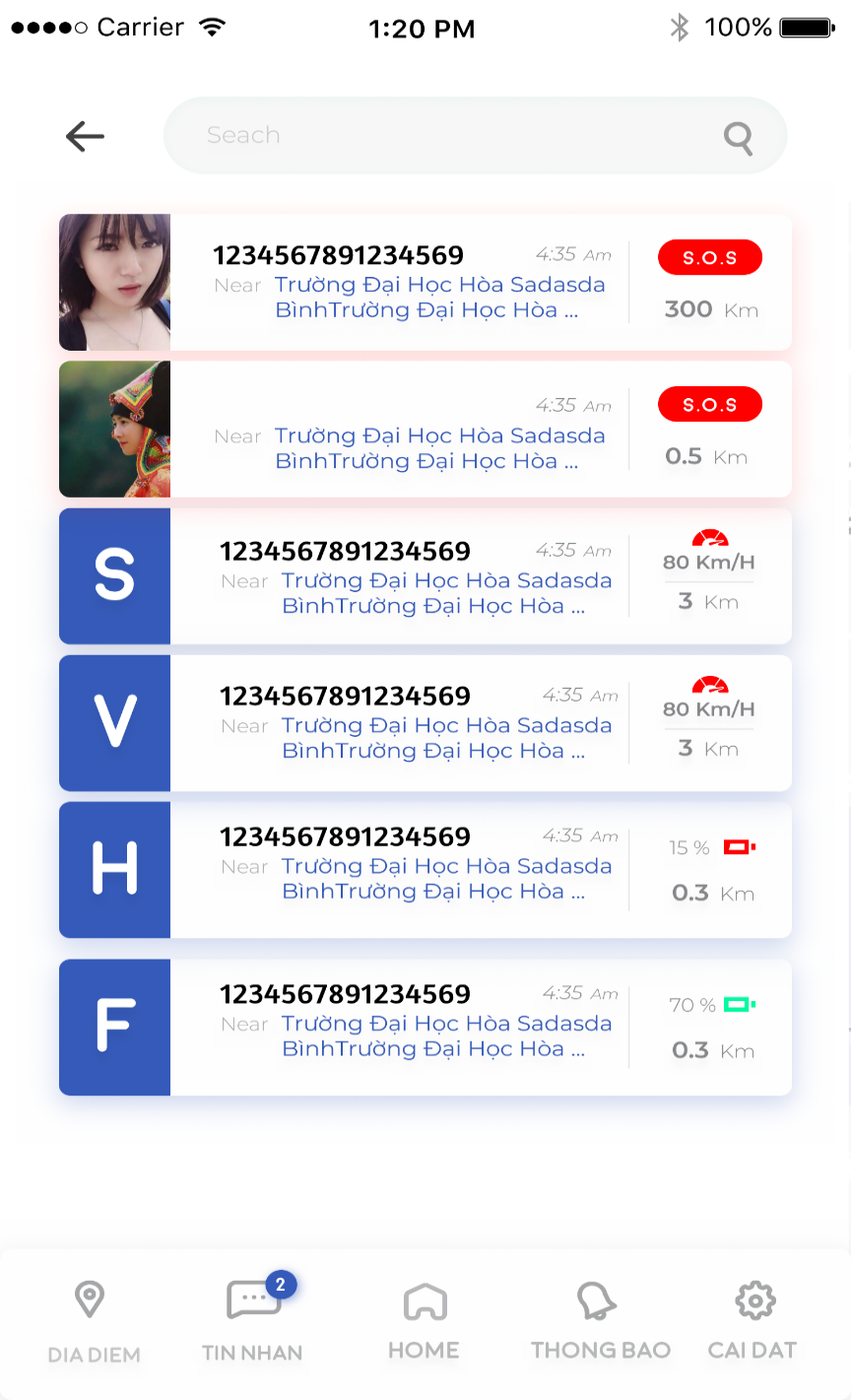
### *4.1.6 Màn hình tham gia vòng*



Hình 4.6 Màn hình tham gia vòng

Màn hình này yêu cầu người dùng nhập code của vòng để có thể gửi yêu cầu tham gia tới chủ của vòng đó.

### *4.1.7 Màn hình danh sách thành viên*



Hình 4.7 Màn hình danh sách thành viên

Màn hình danh sách thành viên hiển thị các thông tin về thành viên như : Avatar, tên thành viên, địa chỉ hiện tại, giờ cập nhật cuối cùng, phần trăm pin, khoảng cách tới mình,….

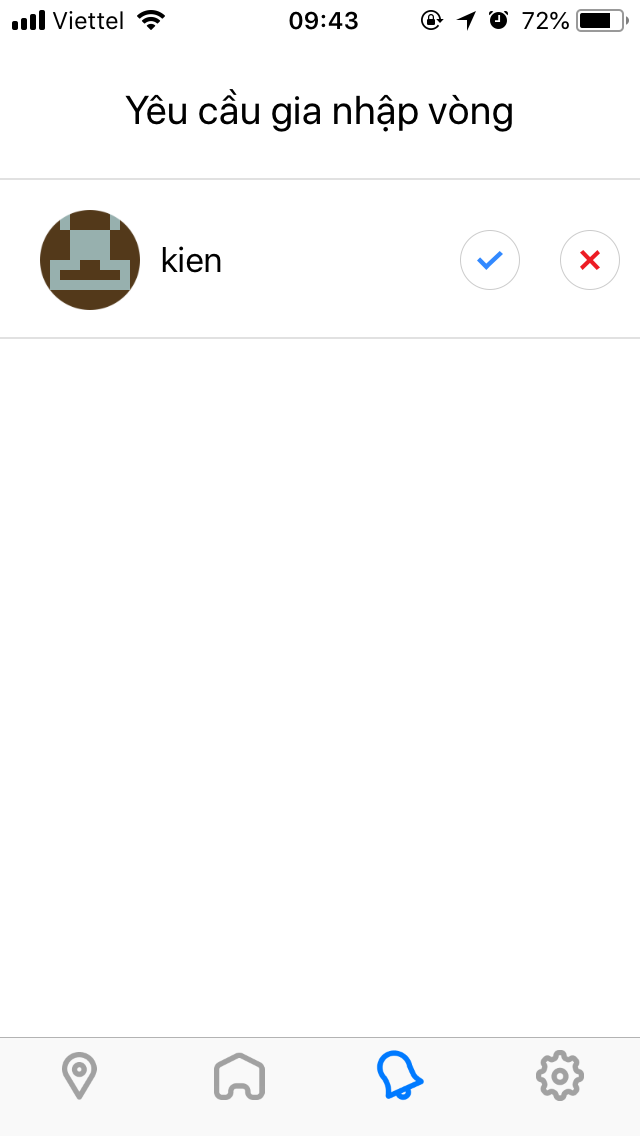
### *4.1.8 Màn hình checkIn*

****

Hình 4.8 Màn hình checkIn

Màn hình checkIn cho phép hiển thị và tìm kiếm các địa điểm xung quanh người dùng để thông báo cho người trong vòng biết mình đang ở gần đó.

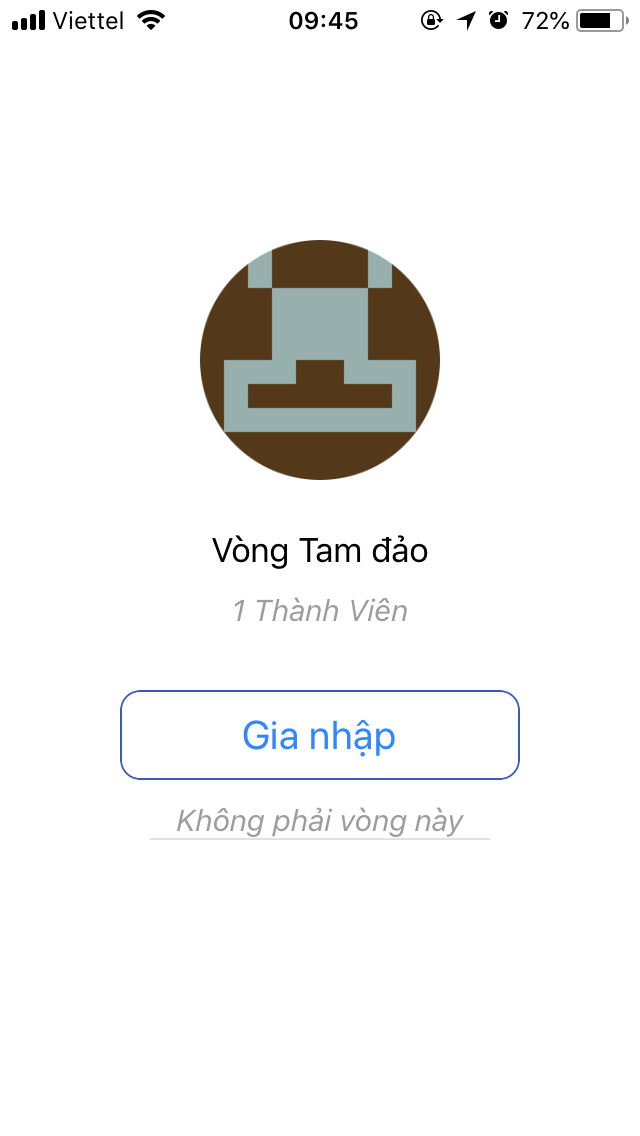
### *4.1.9 Màn hình danh sách yêu cầu vào vòng*



Hình 4.9 Màn hình yêu cầu tham gia vòng

Màn hình yêu cầu tham gia vòng hiển thị các yêu cầu theo vòng hiện tại. Chủ vòng có thể đồng ý hoặc xóa yêu cầu.

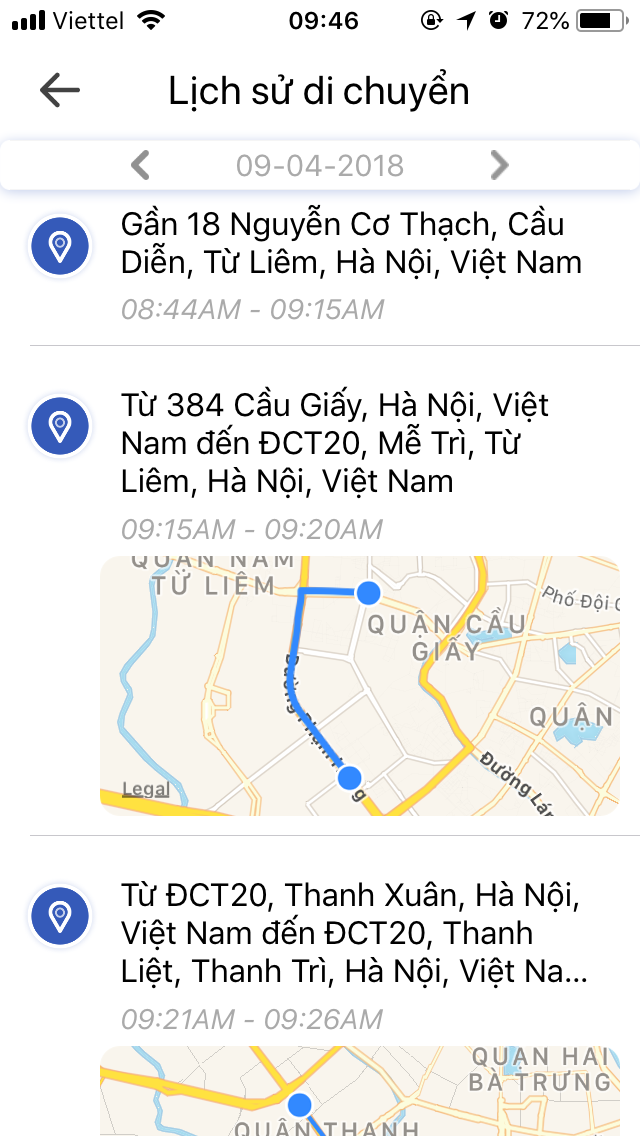
### *4.1.10 Màn hình xem chi tiết vòng trước khi gia nhập*



Hình 4.10 Xem chi tiết vòng

Người dùng có thể xem chi tiết về vòng mình định gia nhập xem có chính xác không. Có thể tham gia hoặc hủy yêu cầu.

### *4.1.11 Màn hình lịch sử di chuyển*

****

Hình 4.11 màn hình lịch sử di chuyển

Màn hình lịch sử di chuyển hiển thị lịch sử của các thành viên trong vòng theo từng ngày. Có thể biết di chuyển từ đâu tới đâu, dừng lại ở đâu vào những giờ nào. Có bản đồ để hiển thị trực quan.

# KẾT LUẬN

Qua thời gian tìm hiểu và nghiên cứu em đã học hỏi được một số công nghệ sau :

* Ngôn ngữ Swift
* SQLite
* SocketIO
* Google Firebase
* Mô hình MVVM

Trong đó đặc biệt em đã vận dụng chính ngôn ngữ Swift, SQLite và mô hình MVVM để hoàn thành sản phẩm của mình

Sau khi nghiên cứu và thực hiện đề tài em đã hoàn thành được sản phẩm ứng dụng ‘Định vị gia đình’ gồm các chức năng sau :

- Tạo tài khoản:

- Đăng nhập

- Quên mật khẩu:

- Tự động lưu trạng thái người dùng

- Tạo vòng kết nối

- Mời vào vòng

- Xin vào vòng

- Lấy danh sách tham gia vòng của mình

- Tạo địa điểm

- Sửa địa điểm

- Tạo địa điểm

- Xem chi tiết địa điểm

- CheckIn

- SOS

- Theo dõi tổng quan các thành viên

- Danh sách chi tiết các thành viên

- Chi tiết thành viên

- Lịch sử di chuyển

- Thông báo

Hướng phát triển sau này để thuận tiện nhất cho việc kết nối các thành viên với nhau em sẽ thực hiện tiếp chức năng nhắn tin và gọi điện trực tiếp giữa các thành viên trong vòng với nhau. Đồng thời sửa các chức năng cũ để phù hợp nhất với người dùng.

Em xin cảm ơn ThS Trần Hùng Cường và các quý thầy cô giáo trong khoa đã tạo điều kiện giúp đỡ em để em hoàn thành được đồ án này.

# Tài liệu tham khảo

[1] *Swift* - <https://developer.apple.com/swift/>

[2]*Swift(programminglanguage)* - <https://en.wikipedia.org/wiki/Swift_(programming_language)>

[3] The Swift Programming Language – Apple.Inc\

[4] *MongoDB* - <https://www.mongodb.com/>

[5] *Học MongoDB cơ bản và nâng cao* - <http://vietjack.com/mongodb/>

[6] *Node.js Introduction* - <https://www.w3schools.com/nodejs/nodejs_intro.asp>