|  |  |
| --- | --- |
|  | **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
| NGUYỄN HỮU THẢO | ĐỒ ÁN TỐT NGHIỆP NGÀNH KHOA HỌC MÁY TÍNH |
| **XÂY DỰNG ỨNG DỤNG MOBILE THEO DÕI SỨC KHOẺ SỬ DỤNG KOTLIN** |
|  | **CBHD : TS. LÊ VIẾT THẮNG** |
|  | **Sinh viên : Nguyễn Hữu Thảo** |
|  | **Mã số sinh viên: 201960421** |
| KHOA HỌC MÁY TÍNH |  |
|  | Hà Nội – Năm 2023 |

**MỤC LỤC**

[DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT i](#_bookmark0)

[DANH MỤC HÌNH VẼ ii](#_bookmark1)

[DANH MỤC BẢNG BIỂU iii](#_bookmark2)

[LỜI CẢM ƠN iv](#_bookmark3)

[MỞ ĐẦU 1](#_bookmark4)

[CHƯƠNG 1. TỔNG QUAN VỀ NỘI DUNG NGHIÊN CỨU 3](#_bookmark5)

* 1. [Giới thiệu chung 3](#_bookmark6)
  2. [Cơ sở lý thuyết 3](#_bookmark7)
     1. [Tổng quan về lập trình di động 3](#_bookmark8)
     2. [Giới thiệu công cụ lập trình Xcode 5](#_bookmark9)
     3. [Ngôn ngữ lập trình Swift 6](#_bookmark10)
     4. [Một số framework trong lập trình Swift 8](#_bookmark11)
     5. [NoSQL Database và Realm Mobile Database 9](#_bookmark12)
     6. [Tổng quan về mô hình MVVM 16](#_bookmark15)

[CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 20](#_bookmark17)

* 1. [Khảo sát người dùng 20](#_bookmark18)
  2. [Giới thiệu về hệ thống 21](#_bookmark19)
  3. [Phân tích yêu cầu 21](#_bookmark20)
     1. [Về hệ thống 21](#_bookmark21)
     2. [Về người sử dụng 22](#_bookmark22)
     3. [Yêu cầu về chức năng 22](#_bookmark23)
  4. [Thiết kế hệ thống 23](#_bookmark24)
     1. [Mô hình hóa Usecase 23](#_bookmark25)
     2. [Mô tả chi tiết các Usecase 27](#_bookmark31)
     3. [Biểu đồ lớp hệ thống 46](#_bookmark43)
  5. [Thiết kế giao diện 47](#_bookmark45)
     1. [Giao diện Quản lý nhắc nhở 47](#_bookmark46)
     2. [Giao diện Quản lý loại nhắc nhở 53](#_bookmark53)
     3. [Giao diện Thống kê 55](#_bookmark56)
     4. [Giao diện Đăng nhập 56](#_bookmark58)
     5. [Giao diện Đồng bộ 57](#_bookmark60)
     6. [Giao diện Thông báo 58](#_bookmark62)
  6. [Thiết kế cơ sở dữ liệu 59](#_bookmark64)

[CHƯƠNG 3. KẾT QUẢ 60](#_bookmark66)

* 1. [Cài đặt môi trường 60](#_bookmark67)
  2. [Cài đặt chương trình 61](#_bookmark70)
  3. [Kết quả thu được 63](#_bookmark74)
  4. [Kết quả kiểm thử 63](#_bookmark75)

[KẾT LUẬN 66](#_bookmark77)

[TÀI LIỆU THAM KHẢO 67](#_bookmark78)

# DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **STT** | **Viết tắt** | **Dịch nghĩa** |
| 1 | HTML/CSS | HyperText Markup Language/Cascading Style Sheets |
| 2 | SQL | Structured Query Language |
| 3 | API | Application Programming Interface |
| 4 | XML | Extensible Markup Language |
| 5 | JSON | JavaScript Object Notation |
| 6 | RMD | Realm Moblie Database |
| 7 | MVVM | Model - View - ViewModel |
| 8 | MVC | Model-View-Controller |
| 9 | CSDL | Cơ sở dữ liệu |
| 10 | SDK | Software Development Kit |

# DANH MỤC HÌNH VẼ

[Hình 1.1: Biểu đồ các công nghệ cơ sở dữ liệu mới giai đoạn 1994-2014 14](#_bookmark13)

[Hình 1.2: Bảng so sánh tốc độ của Realm so với các CSDL khác 15](#_bookmark14)

[Hình 1.3: Mô hình MVVM 19](#_bookmark16)

[Hình 2.1: Biểu đồ Usecase chính 24](#_bookmark26)

[Hình 2.2: Usecase Quản lý nhắc nhở 25](#_bookmark27)

[Hình 2.3: Usecase Quản lý loại nhắc nhở 25](#_bookmark28)

[Hình 2.4: Usecase Thống kê 26](#_bookmark29)

[Hình 2.5: Usecase Đồng bộ 26](#_bookmark30)

[Hình 2.6: Biểu đồ trình tự Quản lý nhắc nhở 31](#_bookmark33)

[Hình 2.7: Biểu đồ trình tự Quản lý loại nhắc nhở 35](#_bookmark35)

[Hình 2.8: Biểu đồ trình tự Thống kê 38](#_bookmark37)

[Hình 2.9: Biểu đồ trình tự Đăng nhập 41](#_bookmark39)

[Hình 2.10: Biểu đồ trình tự Đồng bộ dữ liệu 44](#_bookmark41)

[Hình 2.11: Biểu đồ lớp hệ thống 46](#_bookmark44)

[Hình 2.12: Giao diện màn hình Trang chủ 47](#_bookmark47)

[Hình 2.13: Giao diện Hiển thị tất cả danh sách nhắc nhở 48](#_bookmark48)

[Hình 2.14: Giao diện Hiển thị danh sách nhắc nhở theo ngày 49](#_bookmark49)

[Hình 2.15: Giao diện Thêm nhắc nhở 50](#_bookmark50)

[Hình 2.16: Giao diện Chi tiết nhắc nhở, Sửa, Xóa nhắc nhở 51](#_bookmark51)

[Hình 2.17: Giao diện Tìm kiếm nhắc nhở 52](#_bookmark52)

[Hình 2.18: Giao diện Hiện thị nhắc nhở theo loại 53](#_bookmark54)

[Hình 2.19: Giao diện Thêm, Sửa loại nhắc nhở 54](#_bookmark55)

[Hình 2.20: Giao diện Hiện thị thống kê 55](#_bookmark57)

[Hình 2.21: Giao diện Đăng nhập 56](#_bookmark59)

[Hình 2.22: Giao diện Đồng bộ dữ liệu 57](#_bookmark61)

[Hình 2.23: Giao diện Thông báo 58](#_bookmark63)

[Hình 2.24: Cơ sở dữ liệu 59](#_bookmark65)

[Hình 3.1: Phần mềm Xcode trên Appstore 60](#_bookmark68)

[Hình 3.2: Cửa sổ terminal khi đang cài đặt homebrew 61](#_bookmark69)

[Hình 3.3: Generate file .xcodeproj thành công 61](#_bookmark71)

[Hình 3.4: Giao diện project trên Xcode 62](#_bookmark72)

[Hình 3.5: Ứng dụng được build thành công trên thiết bị simulator 63](#_bookmark73)

# DANH MỤC BẢNG BIỂU

[Bảng 2.1: Mô tả usecase Quản lý nhắc nhở 27](#_bookmark32)

[Bảng 2.2: Mô tả usecase Quản lý loại nhắc nhở 31](#_bookmark34)

[Bảng 2.3: Mô tả usecase Thống kê 36](#_bookmark36)

[Bảng 2.4: Mô tả usecase Đăng nhập 39](#_bookmark38)

[Bảng 2.5: Mô tả usecase Đồng bộ 42](#_bookmark40)

[Bảng 2.6: Mô tả usecase Thông báo 45](#_bookmark42)

[Bảng 3.1: Kiểm thử các chức năng của ứng dụng 63](#_bookmark76)

# LỜI CẢM ƠN

Trong suốt thời gian học tập và thực hiện đồ án tốt nghiệp em luôn nhận được sự hướng dẫn nhiệt tình từ phía nhà trường và các thầy cô giáo. Vì thế, lời đầu tiên em xin chân thành cảm ơn trường Đại học Công nghiệp Hà Nội, thầy cô giáo khoa Công nghệ thông tin đã tạo điều kiện thuận lợi và truyền dạy kiến thức cho em trong thời gian qua để em hoàn thành đề tài một cách tốt nhất. Và hơn hết em xin bày tỏ lòng biết ơn sâu sắc nhất tới thầy giáo hướng dẫn **TS. Lê Viết Thắng** đã tận tình giúp đỡ, định hướng em rất nhiều trong suốt quá trình tìm hiểu nghiên cứu và hoàn thành đồ án tốt nghiệp.

Khi thực hiện nghiên cứu đề tài, em đã cố gắng hoàn thành tốt nhất báo cáo đồ án tốt nghiệp. Tuy nhiên kiến thức chuyên ngành của bản thân còn nhiều hạn chế. Vì vậy trong báo cáo không tránh khỏi những thiếu sót, em rất mong nhận được sự đóng góp của tất cả các thầy cô giáo để đồ án của em được đầy đủ và hoàn chỉnh hơn.

Em xin chân thành cảm ơn!

# MỞ ĐẦU

## Lý do chọn đề tài

Khi thời đại công nghệ thông tin ngày càng phát triển mạnh, đặc biệt khi xuất hiện đại dịch Covid 19 việc bảo bệ sức khoẻ bằng cách tăng cường sức đề kháng là một trong những giải pháp quan trọng để phòng ngừa nguy cơ nhiễm bện, bên cạnh các biện pháp như đeo khẩu trang, rửa tay thường xuyên với xà phòng hoặc nước sát khuẩn...thì cân bằng dinh dưỡng và rèn luyện thể dục là chìa khoá để tăng sức đề kháng. Việc chủ động tăng sức đề kháng là cần thiết để phòng chống dịch hiệu quả. Để có thể theo dõi thể trạng sức khoẻ và việc rèn luyện hằng ngày cần một thiết bị, hệ thống hỗ trợ điều đó. Chính vì vậy ứng dựng theo dõi sức khoẻ là một giải pháp thiết yếu của mỗi cá nhân giúp cho việc theo dõi sức khoẻ và rèn luyện hàng của chúng ta được tốt và hiệu quả hơn.

Từ nhu cầu thực tế trên nên em đã thực hiện đề tài “Xây dựng ứng dụng mobile theo dõi sức khoẻ sử dụng kotlin”, giúp người dùng theo dõi sức khoẻ và việc ràn luyện thể dục hàng ngày.

## Đối tượng, mục tiêu và phạm vi nghiên cứu

* 1. Mục tiêu nghiên cứu
* Phân tích, hiểu bài toán nghiệp vụ thực tế.
* Tìm hiểu công nghệ xây dựng trên nền tảng Android.
* Xây dựng được một ứng dụng theo dõi sức khoẻ đáp ứng đầy đủ các yêu cầu đặt ra.
* Về phía người dùng, đảm bảo cơ động có thể dễ dàng năm bắt tình trạng sức khoẻ và viẹc rèn luyện thể dục hàng ngày.
  1. Đối tượng nghiên cứu
* Ngôn ngữ lập trình Kotlin
* Công cụ lập trình Android Studio
* Mô hình MVVM
* Cơ sở dữ liệu FireBase
  1. Phạm vi nghiên cứu

Ứng dụng mới xây dựng này sẽ đáp ứng các chức năng như sau:

* Theo dõi số bước đi bộ hằng ngày
* Quản lý thông tin dinh dưỡng hằng ngày
* Có thể tuỳ chỉnh ngôn ngữ
* Đăng nhập
* Đăng ký
* Tham gia các cuộc thi rèn luyện sức khoẻ cộng đồng
* Đặt mục tiêu rèn luyện sức khoẻ cá nhân
* Xem các bài tập video hướng dẫn rèn luyện sức khoẻ

## Phương pháp nghiên cứu

* Phương pháp nghiên cứu về mặt lý thuyết:
  + Nghiên cứu và tiến hành thu thập các tài liệu có liên quan đến đề tài.
  + Tổng hợp các tài liệu đã thu thập và tiến hành phân tích.
  + Chọn lọc các tài liệu nghiên cứu để báo cáo đồ án tốt nghiệp.
* Phương pháp nghiên cứu trong thực nghiệm:
  + Tìm hiểu cách thức, tiêu chí kiểm thử chức năng.
  + Phân tích các ứng dụng có chức năng tương tự.
  + Phân tích và tiến hành xây dựng ứng dụng theo phân tích và yêu cầu thực tế.
  + Tìm hiểu ngôn ngữ lập trình Kotlin, công cụ lập trình Android Studio, mô hình MVVM, cơ sở dữ liệu FireBase.
  + Áp dụng công nghệ vào xây dựng hệ thống.
  + Tổng hợp tất các kiến thức đã học hoàn thành báo cáo.

# CHƯƠNG 1. TỔNG QUAN VỀ NỘI DUNG NGHIÊN CỨU

## Giới thiệu chung

Nhà nghiên cứu David Nieman đã dành 40 năm để xem xét mối liên hệ giữa việc tập thể dục và khả năng miễn dịch của cơ thể. Ông đã nhận thấy nếu bạn tập thể dục vừa phải kết hợp với chế độ ăn những loại quả mọng màu đỏ, đen, xanh, thì có thể tăng cường hệ miễn dịch.

Ngoài chế độ ăn uống giúp bạn tăng cường miễn dịch, các nhà nghiên cứu cũng cho thấy tác dụng của chạy bộ có thể giúp cơ chế chống bệnh tật, tăng vi khuẩn đường ruột có lợi và cải thiện phản ứng kháng thể. Không chỉ vậy thói quen chạy bộ cũng giúp bạn giảm viêm, giảm nguy cơ nhiễm trùng đường hô hấp trên và cúm.

Vậy nên để cá nhân có thể theo dõi qúa trình và những bài học thể dục, theo dõi thể trạng cơ thể để có những phương án rèn luyện phù hợp nhất, việc phát triển ứng dụng theo dõi sức khoẻ là vô cùng cần thiết.

## Cơ sở lý thuyết

## Tổng quan về lập trình di động

Hiện nay có 3 hướng chính xây dựng và phát triển 1 ứng dụng di động là: Native app, Web App và Hybrid app. Mỗi hướng đều có ưu và nhược điểm và kỹ năng riêng.

## Hybrid App

Hybrid App là ứng dụng kết hợp của cả Mobile Web App và Native App. Theo đó, các phần cơ bản của ứng dụng vẫn viết bằng ngôn ngữ web, nhưng được đặt trong native container, nên vẫn có thể đưa lên các App Store.

* Ưu điểm:
  + Chỉ cần có kiến thức về HTML, CSS, JavaScript …
  + Viết một lần dùng được nhiều nơi.
  + Tận dụng được các chức năng của hệ thống.
  + Có thể chạy được ở chế độ offline
* Nhược điểm:
  + Không ổn định do khó debug. Framework sẽ dịch code thành code native sửa lỗi khá khó khăn không biết được dịch như thế nào
  + Hiệu năng chậm

## Native App

Native App là những ứng dụng được viết riêng cho một loại nền tảng như iOS, Android, Windows Phone bằng các ngôn ngữ tương ứng của mọi nền tảng đó ví dụ Java trên Android, Object C trên iOS.

Mỗi Native App chỉ chạy được trên một nền tảng và không thể mang sang các nền tảng khác

* Ưu điểm:
  + Tận dụng được tính năng có sẵn trên thiết bị như: GPS, Camera, thiết bị thu âm ...
  + Hiệu năng cao vì code native chạy trực tiếp trên máy
  + Có thế chạy được ở chế độ offline
* Nhược điểm:
  + Không thể kết hợp nhiều nền tảng. Mỗi một ứng dụng chỉ chạy trên 1 nền tảng nhất định
  + Mỗi hệ điều hành cân phải viết ứng dụng riêng khó đồng bộ giữa các ứng dụng
  + Việc bảo trì hay nâng cấp sẽ làm mất nhiều thời gian. Do phải sửa chữa từng app trên từng hệ điều hành
  + Xây dựng ứng dụng cần dùng các phần mềm riêng biệt theo hệ điều hành (dùng Xcode trên Mac để phát triển ứng dụng IOS, Android Studio để phát triển ứng dụng Android)

## Web App

Web app được phát triển khi đã có sẵn một website đang hoạt động. Ta tạo thêm một trang web riêng cho các thiết bị di động sử dụng HTML, CSS và một số thư viện khác hỗ trợ.

Web app được thiết kế chạy trên nền tảng web hoặc các trình duyệt của thiết bị di động cho phép người dùng thao tác như thao tác trang web giống nhu thao tác ứng dụng.

* Ưu điểm:
  + Có thể chạy trên tất cả trình duyệt của mobile hỗ trợ phiên bản HTML và javascript
  + Không cần cài đặt trên máy miễn là máy có trình duyệt web
  + Một phiên bản duy nhất cho tất cả, nên giảm chi phí và thời gian cho phát triển, bảo trì, cũng như nâng cấp sau này.
  + Phiên bản được cập nhật liên tục không cần phải cập nhật trên chợ
* Nhược điểm:
  + Hiệu năng không được tốt như native app và luôn phải chạy online.
  + Không thể dùng được các tính năng tích hợp của di động:

Notification, chụp hình, nghiêng máy, định vị GPS, các sensor…

* + Với một số máy đời cũ, Web app sẽ bị vỡ giao diện, hiển thị sai, hoặc javascript không chạy.

## Giới thiệu công cụ lập trình Android Studio

Tháng 5 năm 2013, Google công bố Android Studio, một môi trường phát triển ứng dụng tích hợp (IDE) dành riêng cho Android, mã nguồn mở, dựa trên IDE **Java IntelliJ** của hãng **JetBrains** (đối thủ với Eclipse và Netbeans, vốn khá quen thuộc với dân lập trình Java). **Android Studio** chạy trên Windows, Mac và Linux, nhằm thay thế cho **Eclipse Android Development Tool** (ADT) vốn được sử dụng làm IDE chính trong các năm trước đó.

## Ngôn ngữ lập trình android kotlin

Tháng 5 năm 2013, Google công bố Android Studio, một môi trường phát triển ứng dụng tích hợp (IDE) dành riêng cho Android, mã nguồn mở, dựa trên IDE **Java IntelliJ** của hãng **JetBrains** (đối thủ với Eclipse và Netbeans, vốn khá quen thuộc với dân lập trình Java). **Android Studio** chạy trên Windows, Mac và Linux, nhằm thay thế cho **Eclipse Android Development Tool** (ADT) vốn được sử dụng làm IDE chính trong các năm trước đó.

Swift là một [ngôn ngữ lập trình](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh) [hướng đối tượng](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng) dành cho việc phát triển [iOS](https://vi.wikipedia.org/wiki/IOS(Apple)) và [macOS](https://vi.wikipedia.org/wiki/OS_X), [watchOS](https://vi.wikipedia.org/wiki/WatchOS), [tvOS](https://vi.wikipedia.org/wiki/TvOS) được giới thiệu bởi [Apple](https://vi.wikipedia.org/wiki/Apple_Inc) tại hội nghị [WWDC](https://vi.wikipedia.org/wiki/WWDC) 2014. Swift được mong đợi sẽ tồn tại song song cùng [Objective-](https://vi.wikipedia.org/w/index.php?title=Objective-C&action=edit&redlink=1) [C](https://vi.wikipedia.org/w/index.php?title=Objective-C&action=edit&redlink=1), ngôn ngữ lập trình hiện tại dành cho các [hệ điều hành](https://vi.wikipedia.org/wiki/H%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh) của Apple. Swift được thiết kế để hoạt động với các [framework](https://vi.wikipedia.org/w/index.php?title=Framework&action=edit&redlink=1) [Cocoa](https://vi.wikipedia.org/wiki/Cocoa_(API)) và [Cocoa Touch](https://vi.wikipedia.org/w/index.php?title=Cocoa_Touch&action=edit&redlink=1) của Apple và phần lớn mã Objective-C. Nó được biên dịch với [trình biên dịch](https://vi.wikipedia.org/wiki/Tr%C3%ACnh_bi%C3%AAn_d%E1%BB%8Bch) [LLVM](https://vi.wikipedia.org/w/index.php?title=LLVM&action=edit&redlink=1) và đã được đưa vào [Xcode](https://vi.wikipedia.org/wiki/Xcode) kể từ phiên bản 6, phát hành năm 2014. Trên các nền tảng của Apple, nó sử dụng [thư viện runtime](https://vi.wikipedia.org/w/index.php?title=Th%C6%B0_vi%E1%BB%87n_runtime&action=edit&redlink=1) Objective-C cho phép mã [C](https://vi.wikipedia.org/wiki/C_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)), [Objective-C](https://vi.wikipedia.org/w/index.php?title=Objective-C&action=edit&redlink=1), [C++](https://vi.wikipedia.org/wiki/C%2B%2B) và Swift cùng chạy trong một chương trình.

Apple dự định Swift hỗ trợ nhiều khái niệm cốt lõi liên quan đến [Objective-C](https://vi.wikipedia.org/w/index.php?title=Objective-C&action=edit&redlink=1), đáng chú ý là thu hồi động, các ràng buộc phổ thông, ập trình mở rộng và các tính năng tương tự, nhưng theo cách "an toàn hơn", giúp dễ dàng bắt [lỗi phần mềm](https://vi.wikipedia.org/wiki/L%E1%BB%97i_ph%E1%BA%A7n_m%E1%BB%81m) hơn. Swift có các tính năng giải quyết một số lỗi lập trình phổ biến như [con trỏ rỗng](https://vi.wikipedia.org/wiki/Con_tr%E1%BB%8F_r%E1%BB%97ng) cung cấp [cú pháp đặc biệt](https://vi.wikipedia.org/wiki/C%C3%BA_ph%C3%A1p_%C4%91%E1%BA%B7c_bi%E1%BB%87t) để giúp tránh [kim tự](https://vi.wikipedia.org/w/index.php?title=Kim_t%E1%BB%B1_th%C3%A1p_di%E1%BB%87t_vong&action=edit&redlink=1) [tháp diệt vong](https://vi.wikipedia.org/w/index.php?title=Kim_t%E1%BB%B1_th%C3%A1p_di%E1%BB%87t_vong&action=edit&redlink=1). Swift hỗ trợ khái niệm về khả năng mở rộng [giao thức](https://vi.wikipedia.org/wiki/Giao_th%E1%BB%A9c_(l%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng)), một hệ thống mở rộng có thể được áp dụng cho các kiểu, cấu trúc và [lớp](https://vi.wikipedia.org/wiki/L%E1%BB%9Bp_(l%E1%BA%ADp_tr%C3%ACnh_m%C3%A1y_t%C3%ADnh)), mà Apple khuyến khích như một sự thay đổi thực sự trong mô hình lập trình mà họ gọi là "lập trình hướng giao thức" (tương tự như đặc điểm).

Swift được giới thiệu tại [Worldwide Developers Conference](https://vi.wikipedia.org/wiki/H%E1%BB%99i_th%E1%BA%A3o_nh%C3%A0_ph%C3%A1t_tri%E1%BB%83n_to%C3%A0n_c%E1%BA%A7u_Apple) (WWDC) 2014 của Apple. Nó đã trải qua quá trình nâng cấp lên phiên bản 1.2 trong năm 2014 và nâng cấp lớn hơn cho Swift 2 tại WWDC 2015. Ban đầu, ngôn ngữ [độc](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_s%E1%BB%9F_h%E1%BB%AFu_%C4%91%E1%BB%99c_quy%E1%BB%81n) [quyền](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_s%E1%BB%9F_h%E1%BB%AFu_%C4%91%E1%BB%99c_quy%E1%BB%81n), phiên bản 2.2 được được chuyển sang [phần mềm nguồn mở](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_ngu%E1%BB%93n_m%E1%BB%9F) theo [Giấy](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_Apache) [phép Apache](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_Apache) 2.0 vào ngày 3 tháng 12 năm 2015, dành cho các nền tảng của Apple và [Linux](https://vi.wikipedia.org/wiki/Linux). Thông qua phiên bản 3.0, cú pháp của Swift đã trải qua quá

trình phát triển quan trọng, với nhóm nòng cốt làm cho sự ổn định nguồn trở thành trọng tâm trong các phiên bản sau. Trong quý đầu tiên của năm 2018, Swift đã vượt qua Objective-C về mức độ phổ biến.

Swift 4.0, được phát hành vào năm 2017, đã giới thiệu một số thay đổi đối với một số lớp và cấu trúc tích hợp. Mã được viết bằng các phiên bản trước của Swift có thể được cập nhật bằng chức năng di chuyển được tích hợp trong Xcode

Vào tháng 3 năm 2017, chưa đầy 3 năm sau khi chính thức ra mắt, Swift đã đứng đầu trong bảng xếp hạng TIOBE hàng tháng về các ngôn ngữ lập trình phổ biến nhất. Một tài liệu 500 trang về Swift cũng được phát hành tại WWDC, miễn phí trên iBooks Store.

* Ưu điểm:
* Khả năng phát triển ứng dụng di động nhanh và dễ dàng: Các lập trình viên không phải mất công chạy các chương trình compiler và các chương trình test một cách thường xuyên nữa, mà thay vào đó tập trung vào các phần tích hợp với mức độ khó hơn. Điều này cũng giúp tiết kiệm sức mạnh và sự hao mòn của phần cứng cho các nhà phát triển. Ngoài ra, Swift là một trong những ngôn ngữ lập trình mang lại nhiều tính năng quan trọng như: không xảy ra lỗi array-out- of-bound, không có biến xác định, quản lý bộ nhớ theo phương thức tự động,... từ đó tiết kiệm được tối đa thời gian hơn để hoàn thành một dự án.
* Code ngắn gọn, dễ đọc, dễ hiểu: Điều này giúp cho các lập trình viên tiết kiệm rất nhiều thời gian hoàn thành dự án, cũng như tăng hiệu quá trong quá trình bảo trì và fix các lỗi bug trong tương lai. Swift kế thừa và phát huy tất cả các sức mạnh của Objective-C để lại, nên các nhà lập trình sẽ có một môi trường runtime quen thuộc tuyệt vời, dễ dàng quản lý và phát triển.
* Tốc độ thực thi của các đoạn mã swift nhanh: Theo apple, Swift nhanh gấp 8 lần python. Không những nhanh, mà Swift còn rất mạnh mẽ, giúp cho các nhà lập trình có thể viết được những đoạn mã nâng cao, phức tạp hơn như: generics, closures, tuples, multiple returns, iterators, các mẫu lập trình hàm có sẵn và nhiều hơn thế nữa.
* Swift có thể dùng cho tất cả các hệ điều hành của Apple phát hành như macOS, iOS, tvOS, watchOS,…

## Một số framework trong lập trình Swift

* + - 1. **UIKit**

UIKit framework cung cấp một số class để tạo các tác vụ chạm (touch- based) cho giao diện người dùng, tất cả các ứng dụng iOS đều xây dựng dựa trên UIKit, nhà phát triển không thể xây dựng ứng dụng mà không sử dụng framework này. UIKit cung cấp cơ sở hạ tầng cho việc vẽ các giao diện trên màn hình, điều khiển các sự kiện, tạo ra các thành phần giao diện của ứng dụng. UIKit cũng tổ chức các ứng dụng phức tạp bằng việc quản lý nội dung ứng dụng để hiện thị lên màn hình. UIDynamics là 1 phần quan trọng trong UIKit được apple tích hợp từ năm 2007 cùng với sự xuất hiện của iOS 7. Bằng cách sử dụng UIKit Dynamics, lập trình viên có thể tạo ra các view với những tương tác vật lý như trọng lực, móc nối, va chạm, nảy lên(sau va chạm),... giữa các đối tượng với các thành phần chính: UIDynamicItem protocol, Dynamic behaviors, Dynamic animator.

## Core graphics

Core graphics là một framework cấp độ thấp, dựa trên ngôn ngữ lập trình C, chịu trách nhiệm điều khiển các vector đồ họa chất lượng cao, các nét vẽ path, hình ảnh, quản lý dữ liệu và nhiều tác vụ khác. Tất nhiên việc đơn giản nhất và hiệu quả nhất để tạo đồ họa trong iOS là sử dụng các hình ảnh với khả năng hiện thị chuẩn kết hợp với việc điều khiển của UIKit. Apple khuyến khích

các nhà phát triển nên sử dụng UIKit để vẽ, bởi vì UIKit là framework ở mức độ cao, UIKit cũng cung cấp các thư viện, class phục vụ việc vẽ path, pattern, gradient, images, text.

## Core animation

Core animation là một công nghệ cho phép bạn tạo các hiệu ứng animation mức cao, và hiệu quả trực quan. UIKit cung cấp các animation để xây dựng các Core Animation Technology. Nếu cần một hiệu ứng nâng cao cho ứng dụng vượt khả năng của UIKit thì đó có thể là Core Animation. Core Animation chứa Quartz Core Framework, với việc sử dụng Core Animation bạn có thể tạo ra các phân lớp các đối tượng layer như rotate (xoay), scale ( co giãn), …. Việc sử dụng Core Animation cũng giúp bạn thực hiện các công việc tránh sử dụng các API đồ họa cấp thấp như OPenGL ES.

## Core data

Core Data là framework quản lý mô hình dữ liệu của ứng dụng. Với việc sử dụng Core Data, bạn có thể tạo các mô hình hóa đối tượng như việc quản lý đối tượng. Bạn quản lý các mối quan hệ giữa nhưng đối tượng này và tạo ra sự thay đổi đến dữ liệu thông qua framework. Core Data cung cấp việc xây dựng ở mức nâng cao trong SQLite, để lưu trữ và quản lý dữ liệu hiệu quả.

## NoSQL Database và Realm Mobile Database

* + - 1. **NoSQL Database**

Thuật ngữ NoSQL được giới thiệu lần đầu vào năm 1998 sử dụng làm tên gọi chung cho các lightweight open source relational database (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 trong một hội thảo về cơ sở dữ liệu nguồn mở phân tán. Thuật ngữ NoSQL đánh dấu bước phát triển của thế hệ database mới: distributed (phân tán) và non-relational (không ràng buộc). Đây là cũng 2 đặc tính quan trọng nhất.

* Có một vài lí do chứng minh cho sự ra đời của NoSQL:
* Sở dĩ người ta phát triển NoSQL suất phát từ yêu cầu cần những database có khả năng lưu trữ dữ liệu với lượng cực lớn, truy vấn dữ liệu với tốc độ cao mà không đòi hỏi quá nhiều về năng lực phần cứng cũng như tài nguyên hệ thống và tăng khả năng chịu lỗi.
* Giải quyết được một số vấn đề mà relational database không giải quyết được.
  + Một số đặc điểm chung về NoSQL:
* High Scalability: Gần như không có một giới hạn cho dữ liệu và người dùng trên hệ thống.
* High Availability: Do chấp nhận sự trùng lặp trong lưu trữ nên nếu một node (commodity machine) nào đó bị chết cũng không ảnh hưởng tới toàn bộ hệ thống.
* Atomicity: Độc lập data state trong các operation.
* Consistency: chấp nhận tính nhất quán yếu, có thể không thấy ngay được sự thay đổi mặc dù đã cập nhật dữ liệu.
* Durability: dữ liệu có thể tồn tại trong bộ nhớ máy tính nhưng đồng thời cũng được lưu trữ lại đĩa cứng.
* Deployment Flexibility: việc bổ sung thêm/loại bỏ các node, hệ thống sẽ tự động nhận biết để lưu trữ mà không cần phải can thiệp bằng tay. Hệ thống cũng không đòi hỏi cấu hình phần cứng mạnh, đồng nhất.
* Modeling flexibility: Key-Value pairs, Hierarchical data (dữ liệu cấu trúc), Graphs.
* Query Flexibility: Multi-Gets, Range queries (load một tập giá trị dựa vào một dãy các khóa).
  1. *Key value stores*

Lưu trữ kiểu key-value là kiểu lưu trữ dữ liệu NoSQL đơn giản nhất sử dụng từ một API. Chúng ta có thể nhận được giá trị cho khóa, đặt một giá trị cho một khóa, hoặc xóa một khóa từ dữ liệu. Ví dụ, giá trị là ‘blob’ được lưu trữ thì chúng ta không cần quan tâm hoặc biết những gì ở bên trong. Từ các cặp giá trị được lưu trữ luôn luôn sử dụng truy cập thông qua khóa chính và thường có hiệu năng truy cập tốt và có thể dễ dàng thu nhỏ lại.

* 1. *Column oriented database*

Cơ sở dữ liệu column-family lưu trữ dữ liệu trong nhiều cột trong mỗi dòng với key cho từng dòng. Column families là một nhóm các dữ liệu liên quan được truy cập cùng với nhau. Ví dụ, với khách hàng, chúng ta thường xuyên sử dụng thông tin cá nhân trong cùng một lúc chứ không phải hóa đơn của họ. Cassandra là một trong số cơ sở dữ liệu column-family phổ biến. Ngoài ra còn có một số cơ sở dữ liệu khác như HBase, Hypertable và Amazon DynamoDB. Cassandra có thể được miêu tả nhanh và khả năng mở rộng dễ dàng với các thao tác viết thông qua các cụm. Các cụm không có node master, vì thế bất kỳ việc đọc và ghi nào đểu có thể được xử lý bởi bất kỳ node nào trong cụm.

* 1. *Graph database*

Kiểu đồ thị này cho phép bạn lưu trữ các thực thể và quan hệ giữa các thực thể. Các đối tượng này còn được gọi là các nút, trong đó có các thuộc tính. Mỗi nút là một thể hiện của một đối tượng trong ứng dụng. Quan hệ được gọi là các cạnh, có thể có các thuộc tính. Cạnh có ý nghĩa định hướng; các nút được tổ chức bởi các mối quan hệ. Các tổ chức của đồ thị cho phép các dữ liệu được lưu trữ một lần và được giải thích theo nhiều cách khác nhau dựa trên các mối quan hệ. Thông thường, khi chúng ta lưu trữ một cấu trúc đồ thị giống như trong RDBMS, nó là một loại duy nhất của mối quan hệ. Việc tăng thêm một

mối quan hệ có nghĩa là rất nhiều thay đổi sơ đồ và di chuyển dữ liệu, mà không phải là trường hợp khó khi chúng ta đang sử dụng cơ sở dữ liệu đồ thị. Trong cơ sở dữ liệu đồ thị, băng qua các thành phần tham gia hoặc các mối quan hệ là rất nhanh. Các mối quan hệ giữa các node không được tính vào thời gian truy vấn nhưng thực sự tồn tại như là một mối quan hệ. Đi qua các mối quan hệ là nhanh hơn so với tính toán cho mỗi truy vấn.

* 1. *Document Oriented databases*

Tài liệu là nguyên lý chính của cơ sở dữ liệu kiểu dữ liệu. Dữ liệu lưu trữ và lấy ra là các tài liệu với định dạng XML, JSON, BSON,… Tài liệu miêu tả chính nó, kế thừa từ cấu trúc dữ liệu cây. Có thể nói cơ sở dữ liệu tài liệu là 1 phần của key-value. Cơ sở dữ liệu kiểu tài liệu như MongoDB cung cấp ngôn ngữ truy vấn đa dạng và cúc trúc như là cơ sở dữ liệu như đánh index,… Một số cơ sở dữ liệu tài liệu phổ biến mà chúng ta hay gặp là MongoDB, CouchDB, Terastore, OrientDB, RavenDB.

* + Ưu điểm của NoSQL:
* Không sử dụng SQL
* Không khai báo ngôn ngữ truy vấn dữ liệu
* Không định nghĩa schema
* Có 1 số nhóm dạng: Key-Value pair storage, Column Store, Document Store, Graph databases
* Dữ liệu phi cấu trúc và không thể đoán trước
* Ưu tiên cho hiệu năng cao, tính sẵn sàng cao và khả năng mở rộng

## Realm Mobile Database

Trong mỗi ứng dụng thì phần quan trọng không kém chính là Cơ sở dữ liệu. CSDL phổ biến nhất được sử dụng hiện nay trên hầu hết các thiết bị là SQLite bởi vì nó khá quen thuộc với đại đa số các lập trình viên do sử dụng câu truy vấn SQL. Tuy nhiên, SQLite cũng có những mặt hạn chế nhất định như tốc

độ truy vấn khá chậm khi mà dữ liệu phình to ra cũng như khi mà thực hiện phép JOIN. Hơn thế nữa, với mỗi ngôn ngữ khác nhau thì việc thiết lập SQLite có thể tốn khá nhiều công sức.

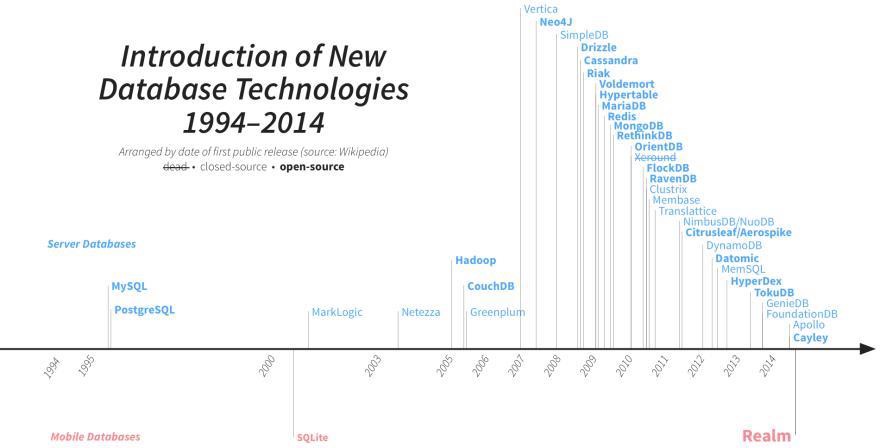
Trên cơ sở đó, Realm Mobible Database ra đời với mục đích cung cấp cho lập trình viên một lựa chọn có thể thay thế cho SQLite hiện nay nhưng vẫn đảm bảo mọi chức năng cần thiết của một CSDL thông thường:

* Realm Moblie Database ( gọi tắt là RMD) là một NoSQL ( Not Only SQL). Nó hướng tới việc xây dựng một ứng dụng theo hướng Offline database first. Điều này có nghĩa là ứng dụng vẫn có thể hoạt động dù cho không có kết nối mạng, dữ liệu sẽ được lưu trực tiếp trên thiết bị, người dùng vẫn có thể tiến hành mọi việc một cách thuận lợi.
* RMD lưu trữ dữ liệu dưới dạng Object và nó cũng cung cấp các hàm và phương thức để có thể truy vấn dữ liệu mà không cần thông qua câu truy vấn SQL.
* Phần core của RMD được viết bằng C++ và là mã nguồn mở, người dùng có thể tùy chỉnh lại theo ý muốn cá nhân.
* Cross-flatform và đã có phiên bản cho các ngôn ngữ sau: Swift, Java, Objective – C, Xamarin, React Native.
* Cung cấp miễn phí.

Realm đơn giản hóa kiến trúc ứng dụng của bạn trong khi cung cấp cho bạn các tính năng rất mạnh mẽ. Nền tảng Realm thực sự được tạo thành từ hai thành phần chính. Cơ sở dữ liệu Realm và Máy chủ đối tượng Realm. Theo tài liệu chính thức, hai thành phần này hoạt động cùng nhau để tự động đồng bộ hóa dữ liệu cho phép nhiều trường hợp sử dụng khác nhau, từ ứng dụng ngoại tuyến đến tích hợp phụ trợ phức tạp.

Một trong những yếu tố chính khiến Realm trở nên đặc biệt là thực tế là nó được xây dựng cho nhu cầu phát triển di động hiện đại của chúng tôi và rất

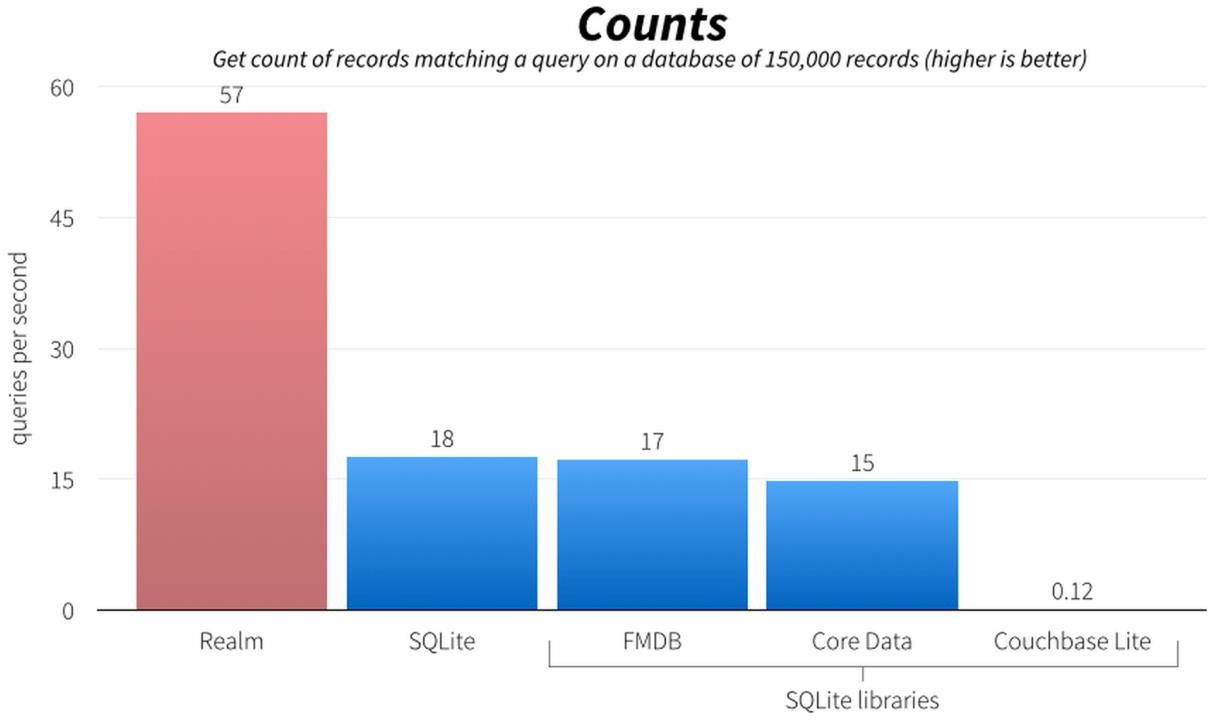
nhiều giải pháp chúng tôi sử dụng không thực sự được thiết kế cho điện thoại, máy tính bảng hoặc thiết bị đeo được:



*Hình 1.1: Biểu đồ các công nghệ cơ sở dữ liệu mới giai đoạn 1994-2014*

* + Điểm mạnh của Realm mobile database:
* Dễ cài đặt và dễ sử dụng: RMD khá dễ cài đặt và sử dụng. Đối với IOS, có thể sử dụng thư viện quản lý CocoaPods để cài đặt và sử dụng. Để sử dụng RMD, lập trình viên chỉ cần tạo một class như bình thường, và kế thừa từ class “Object” của RMD.
* Tốc độ truy vấn nhanh: RMD được tối ưu hóa để sử dụng bộ nhớ một cách ít nhất nhưng hiệu suất vẫn vượt trội so với các CSDL khác.

Dưới đây là bảng so sánh tốc độ của Realm so với các CSDL khác.



*Hình 1.2: Bảng so sánh tốc độ của Realm so với các CSDL khác*

* Có browser giúp việc quản lý dữ liệu, thêm, sửa, xoá dữ liệu trực tiếp trên giao diện phần mềm được thực hiện một cách dễ dàng.
* Cross-platform: Realm hiện nay đã có phiên bản hỗ trợ trên các ngôn ngữ lập trình di động phổ biến. Có thể sử dụng một database duy nhất cho tất cả các phiên bản trên các hệ điều hành di động khác nhau. Hiện nay Realm đã có hỗ trợ cho các ngôn ngữ sau: Java, Swift, Objective – C, Xamarin, React Native.
* Realm hỗ trợ *custom migrating* - việc cập nhật CSDL, thêm/bớt các bảng, trường đều được hỗ trợ một cách thuận tiện, đơn giản nhất bằng cách kế thừa lớp RealmMigration.
* Các lớp của Realm hỗ trợ nhiều trong việc tránh lỗi Null Pointer Exception. Khi truy vấn không có kết quả, đối tượng RealmResults được trả về sẽ là rỗng, và phương thức size() sẽ cho ra 0.
  + Điểm yếu của RMD:
* Không hỗ trợ trường tự tăng giá trị để làm khoá chính.
* Không thể custom initializer.
* Không thể override hàm getter, setter.

## Tổng quan về mô hình MVVM

MVVM là viết tắt của Model - View - ViewModel, đây là mô hình hỗ trợ two-way data binding giữa View và View Model. Cụ thể mô hình MVVM được trình bày như sau:

* *View*

Tương tự như trong mô hình MVC, View là phần giao diện của ứng dụng để hiển thị dữ liệu và nhận tương tác của người dùng. Một điểm khác biệt so với các ứng dụng truyền thống 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 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

* *View Model*

Là lớp trung gian giữa View và Model. View Model có thể được xem là thành phần thay thế cho Controller trong mô hình MVC. Nó chứa các mã lệnh thực hiện Data Binding, Command. Một điểm cần lưu ý là trong mô hình MVVM, các tầng bên dưới sẽ không biết được các thông tin gì về các tầng trên của nó.

* *Một định nghĩa khác về ViewModel trong MVVM:*

ViewModel 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 ViewModel lại không biết thông tin gì về View. nó được ẩn giấu qua cách sử dụng Data-binding và cơ chế của mô hình Observer, một ViewModel có thể được ánh xạ từ nhiều View

**Lưu ý**: Trong mô hình MVVM, các tầng bên dưới sẽ không biết được các thông tin gì về tầng bên trên nó. ViewModel cũng không hề biết gì về View, một ViewModel có thể sử dụng cho nhiều View thường được gọi là binding data, có thể là 1 chiều hoặc 2 chiều tùy nhu cầu của ứng dụng.

Thông thường khi sử dụng với MVVM chúng ta nên tạo 3 thư mục chính chứa các file code liên quan

* *Views*

Tại đây chứa 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 đồng nghĩa điều đó sẽ phá vỡ quy ước của 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 tạo các tầng chứa dữ liệu và bất kỳ liên kết validation, logic nghiệp vụ để chắc chắn tính toàn vẹn của data, bạn có thể tách

ra như một Repositories khác, chúng được dùng như một phần của mô hình MVVM

* *ViewModels*

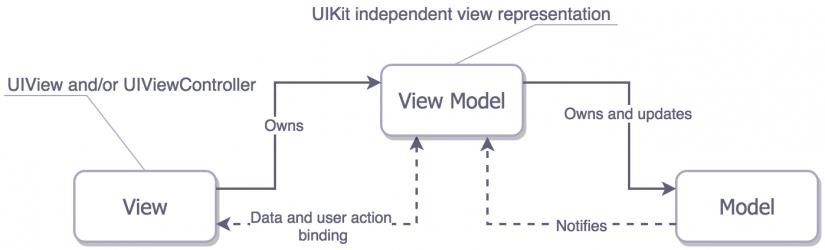
Thông thường trong một file giao diện thì ta tạo ra một class View Models tương ứng (có đôi lúc ta tạo nhiều class phụ giúp tinh giản file code và gọi chúng trong class ViewModels chính)

ViewModels sẽ sử dụng các models 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. ViewModel cần chứa các chức năng của ứng dụng. ViewModels định nghĩa properties, commands và events để chuyển đổi controls trong view cần data- bind

- Data binding

Binding Data trong MVVM là điều không bắt buộc, một số implement chỉ đơn giản làm ViewModel như một lớp trung gian giữa Model-View, lớp này giữ nhiệm vụ format data hoặc mapping trạng thái của View. Tuy nhiên cách này theo mình khiến cho ViewModel trở thành Presenter và đưa kiến trúc này về MVP.



*Hình 1.3: Mô hình 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

# CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## Khảo sát người dùng

* Câu 1: Bạn gặp khó khăn hay vấn đề gì trong việc quản lý sắp xếp công việc?

Trả lời:

* + Chưa biết sắp xếp hợp lý, vẫn lộn xộn, không liệt kê đủ công việc cần thực hiện
  + Hãy có thói quen trì hoãn, không biết việc nào quan trọng, việc nào sắp đến deadline
  + Nhiều việc quá sắp xếp không được hiệu quả
* Câu 2: Bạn có thường dễ quên và bỏ lỡ các công việc quan trọng không?

Trả lời:

* + Có
* Câu 3: Bạn thường dùng cách nào để sắp xếp quản lý công việc khoa học?

Trả lời:

* + Sổ ghi chép, app hoặc tự nhớ
  + Dùng google calendar, trello
  + Ghi lại trên ứng dụng Note
  + Dùng thời gian biểu, dùng các công cụ để sắp xếp lịch trình
* Câu 4: Bạn đã dùng ứng dụng nhắc nhở nào?
  + Google Calendar
  + Trello
  + Chưa dùng app nào
  + Dùng các ứng dụng sẵn của điện thoại
* Câu 5: Bạn hay gặp khó khăn gì khi sử dụng ứng dụng nhắc nhở?
  + Lúc báo, lúc không
  + Công việc rối, không biết làm cái nào trước
  + Giao diện hơi khó dùng
  + Chưa phù hợp với bản thân nên khó trong việc sử dụng
* Câu 6: Bạn thích nhất chức năng gì trong một ứng dụng nhắc nhở công việc?
  + Nhắc công việc khi chưa hoàn thành và sẽ không tắt thông báo cho đến khi hoàn thành
  + Có thể thay đổi linh hoạt, nhiều chức năng để tích hợp với từng hoàn cảnh, người dùng khác nhau
  + Thông báo trước một khoảng thời gian trước khi chúng ta cần phải thực hiện công việc đó
  + Giao diện đẹp, trực quan. Âm thanh thông báo tuỳ vào mức độ quan trọng của việc chẳng hạn

## Giới thiệu về hệ thống

Qua khảo sát và trên cơ sở phân tích ở các giai đoạn trước cần xây dựng một hệ thống nhắc nhở hoàn chỉnh và chuyên dụng. Ứng dụng sẽ cho phép người sử dụng tạo nhắc nhở định kỳ, tạo loại nhắc nhở tùy theo đặc điểm và ngành nghề khác nhau. Cơ động có thể dễ dàng quản lý được thông tin các ngày nhắc nhở. Thông báo nhắc nhở công việc khi chưa hoàn thành kịp thời và đúng lúc. Đồng thời có thể xem thống kê công việc: đã hoàn thành, quá thời hạn và vẫn đang tiến hành. Hệ thống cũng giúp ghi lại những kế hoạch định làm trong tương lai nhưng chưa định thời gian, hệ thống sẽ có hình thức gợi nhớ cho người dùng những công việc này.

## Phân tích yêu cầu

## Về hệ thống

Ứng dụng nhắc nhở sẽ hướng tới một hệ thống có giao diện đẹp, dễ sử dụng, dễ quản lý và đem lại một sự trải nghiệm tuyệt vời nhất cho người sử dụng. Bên cạnh đó, ứng dụng cũng hướng tới sự tối ưu hiệu suất, tiết kiệm thời gian nhất cho người sử dụng.

## Về người sử dụng

Người dùng có thể nắm bắt được các công việc, quản lý nhắc việc. Ngoài việc có thể xem danh sách thông tin tất cả các công việc thì người dùng có thể xem chi tiết theo loại nhắc nhở, xem theo ngày, hay chỉ xem của hôm nay và thực hiện tạo mọi lời nhắc theo ý muốn trong ứng dụng.

## Yêu cầu về chức năng

* Các chức năng chính của hệ thống:
* Quản lý nhắc nhở
* Quản lý loại nhắc nhở
* Thống kê
* Đăng nhập
* Đồng bộ dữ liệu
* Thông báo
* Yêu cầu phi chức năng:
* Giao diện đồ họa
* Hoạt động của hệ thống
* Ngôn ngữ
* Độ tin cậy của ứng dụng
* Hiệu năng của ứng dụng

## Thiết kế hệ thống

## Mô hình hóa Usecase

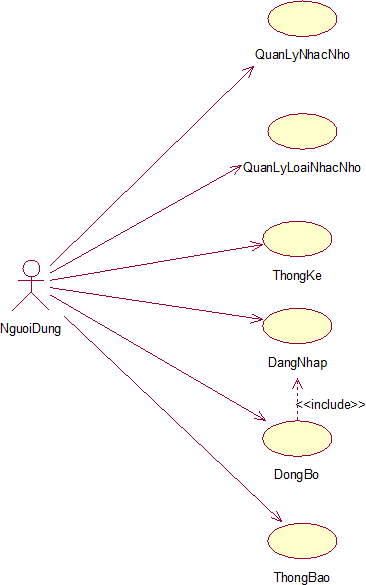
* + - 1. **Xác định các tác nhân**
* Người dùng

## Xác định các Usecase

* Quản lý nhắc nhở
* Quản lý loại nhắc nhở
* Thống kê
* Đăng nhập
* Đồng bộ dữ liệu
* Thông báo

## Biểu đồ Usecase

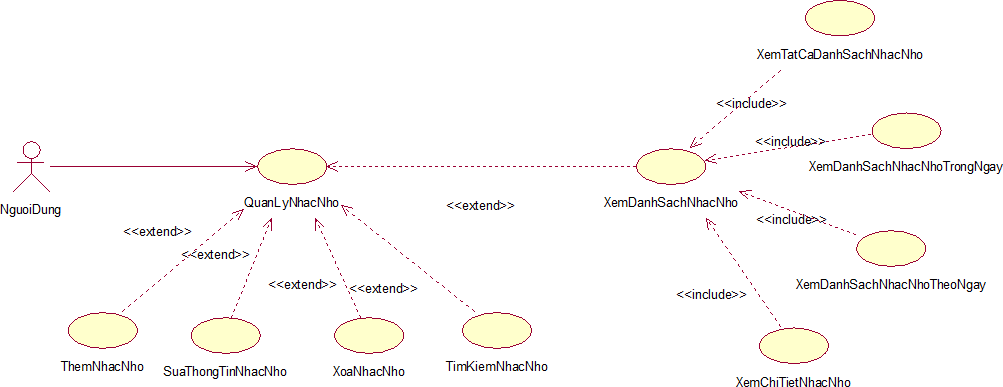
### *Biểu đồ Usecase chính*



*Hình 2.1: Biểu đồ Usecase chính*

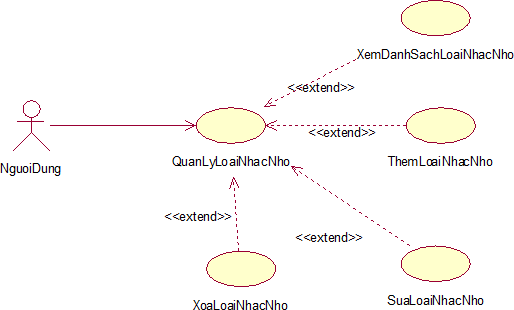
### *Biểu đồ Usecase thứ cấp*

*Phân rã Usecase <Quản lý nhắc nhở>*



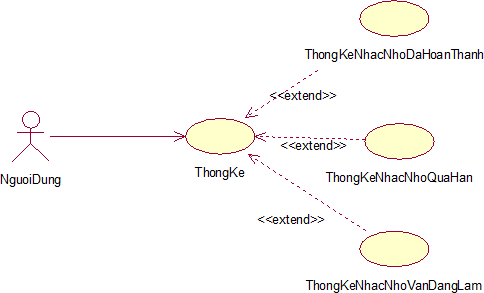
*Hình 2.2: Usecase Quản lý nhắc nhở*

*Phân rã Usecase <Quản lý loại nhắc nhở>*



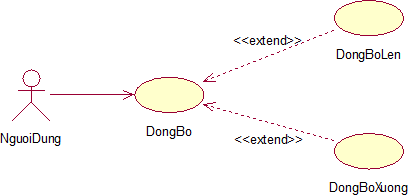
*Hình 2.3: Usecase Quản lý loại nhắc nhở*

*Phân rã Usecase <Thống kê>*



*Hình 2.4: Usecase Thống kê*

*Phân rã Usecase <Đồng bộ>*



*Hình 2.5: Usecase Đồng bộ*

## Mô tả chi tiết các Usecase

* + - 1. **Mô tả Usecase <Quản lý nhắc nhở>**

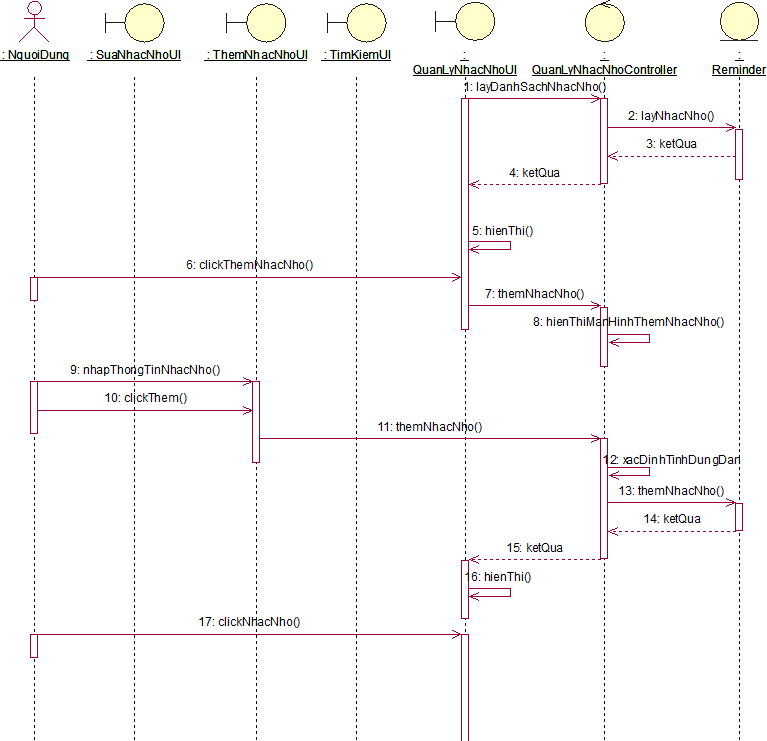
*Bảng 2.1: Mô tả usecase Quản lý nhắc nhở*

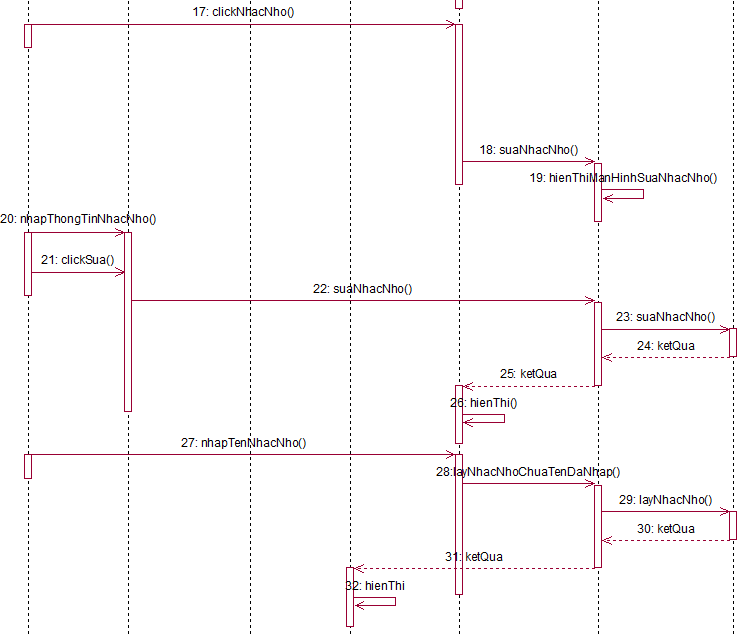
|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase này cho phép người dùng quản lý và xem thông tin nhắc nhở |
| **Tác nhân** | | Người dùng |
| **Tiền điều kiện** | | Người dùng sử dụng ứng dụng |
| **Luồng sự kiện** | **Luồng sự kiện chính** | * Usecase bắt đầu khi người dùng kích hoạt ứng dụng. Khi đó trên màn hình chính của hệ thống hiển thị lên màn hình   quản lý nhắc nhở |
|  |  | 1. Thêm nhắc nhở: |
|  |  | - Người dùng chọn vào button thêm nhắc  nhở trên màn hình, hệ thống hiển thị lên màn hình thêm nhắc nhở. |
|  |  | - Người dùng nhập thông tin và chọn “Thêm nhắc nhở” để hoàn thành. |
|  |  | - Người dùng nhập thiếu thông tin hệ thống chuyển sang luồng A1. |
|  |  | - Hệ thống kiểm tra thông tin và cập nhật thông tin vào cơ sở dữ liệu. |
|  |  | 2. Sửa thông tin nhắc nhở |
|  |  | - Người dùng chọn nhắc nhở cần thay đổi thông tin và ấn vào nó. |

|  |  |  |
| --- | --- | --- |
|  |  | * Hệ thống hiển thị giao diện chi tiết thông tin nhắc nhở đó. * Người dùng nhập thông tin cần thay đổi và chọn “Save” * Hệ thống kiểm tra lại thông tin và cập nhật vào cơ sở dữ liệu  1. Xóa nhắc nhở    * Người dùng click vào nhắc nhở muốn xóa và chọn “Delete”. Hoặc vuốt nhắc nhở đó sang bên trái và chọn “Delete”    * Hệ thống thực hiện xóa và cập nhật lại cơ sở dữ liệu 2. Xem danh sách nhắc nhở    * Người dùng chọn hiển thị nhắc nhở trên màn hình.    * Hệ thống kiểm tra danh sách nhắc nhở.    * Hệ thống hiển thị danh sách nhắc nhở.    * Người dùng click chọn nhắc nhở.    * Hệ thống hiển thị màn hình chi tiết của nhắc nhở đó.    * Người dùng chọn nút “Save” nếu muốn sửa thông tin nhắc nhở.    * Người dùng chọn nút “Delete” để xóa nhắc nhở khỏi danh sách. |

|  |  |  |
| --- | --- | --- |
|  |  | 1. Tìm kiếm nhắc nhở    * Người dùng nhập tên nhắc nhở và nhấn nút tìm kiếm trên màn hình giao diện home.    * Hệ thống hiển thị màn hình danh sách chứa các từ khóa tìm kiếm của người dùng đã nhập.      + Ca sử dụng kết thúc khi người dùng   chọn chức năng khác hoặc thoát khỏi hệ thống. |
| **Luồng rẽ**  **nhánh** | * Luồng A1: Người dùng nhập thiếu thông tin nhắc nhở   - Hệ thống hiển thị button trạng thái disable. |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:





*Hình 2.6: Biểu đồ trình tự Quản lý nhắc nhở*

## Mô tả Usecase <Quản lý loại nhắc nhở>

*Bảng 2.2: Mô tả usecase Quản lý loại nhắc nhở*

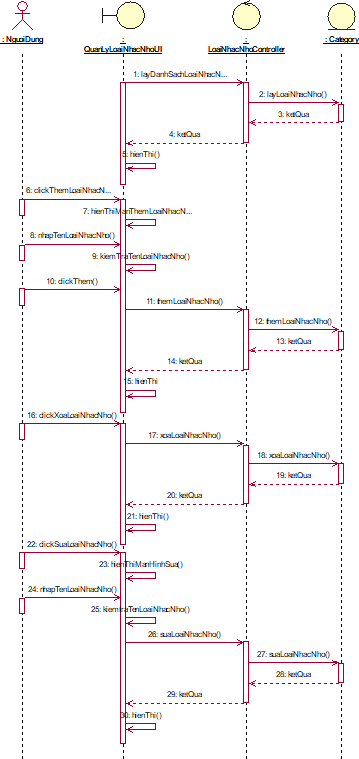
|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase này cho phép người dùng quản lý được thông tin loại nhắc nhở. |
| **Tác nhân** | | Người dùng |
| **Tiền điều kiện** | | Người dùng sử dụng ứng dụng |
| **Luồng sự kiện** | **Luồng sự kiện chính** | * Usecase bắt đầu khi người dùng kích hoạt ứng dụng. Khi đó trên màn hình chính của hệ thống hiển thị lên một mục danh sách loại nhắc nhở. |

|  |  |  |
| --- | --- | --- |
|  |  | 1. Thêm loại nhắc nhở:    * Người dùng chọn vào nút thêm loại nhắc nhở trên màn hình, hệ thống hiển thị lên màn hình thêm loại nhắc nhở.    * Người dùng nhập tên loại nhắc nhở mới, chọn thông tin và chọn “Done” để hoàn thành.    * Hệ thống kiểm tra tính đúng đắn của tên loại nhắc nhở.    * Nếu tên loại nhắc nhở đã có trong danh sách loại nhắc nhở của hệ thống chuyển sang luồng A1. 2. Sửa loại nhắc nhở    * Người dùng chọn loại nhắc nhở muốn sửa.    * Hệ thống hiển thị màn hình danh sách nhắc nhở của loại đó.    * Người dùng chọn nút ba chấm trên màn hình và chọn chức năng Sửa.    * Hệ thống hiển thị lên màn hình sửa loại nhắc nhở, người dùng nhập tên mới chọn lại thông tin.    * Trong trường hợp người dùng sửa tên có tên trùng với loại nhắc nhở đã có trong danh sách loại nhắc nhở hệ thống chuyển sang luồng A2 |

|  |  |  |
| --- | --- | --- |
|  |  | * Nếu người dùng không nhập tên loại mặt hàng hệ thống chuyển sang luồng A4 * Người dùng ấn “Done” hệ thống lưu tên mới của loại mặt hàng vào CSDL  1. Xóa loại nhắc nhở    * Người dùng chọn loại nhắc nhở muốn xóa.    * Hệ thống hiển thị màn hình danh sách nhắc nhở của loại đó.    * Người dùng chọn nút ba chấm trên màn hình và chọn chức năng Xóa.    * Nếu trong loại nhắc nhở đã có danh sách nhắc nhở, hệ thống chuyển sang luồng A4.      + Ca sử dụng kết thúc khi người dùng chọn chức năng khác hoặc thoát khỏi hệ thống. |
| **Luồng rẽ**  **nhánh** | * Luồng A1: Tên người dùng nhập đã có trong danh sách loại nhắc nhở.   - Hệ thống hiển thị thông báo cho người dùng yêu cầu người dùng nhập lại   * Luồng A2: Người dùng sửa tên trùng với tên của loại nhắc nhở đã có   - Hệ thống hiển thị thông báo yêu cầu người dùng nhập lại.   * Luồng A3: Người dùng không nhập tên loại nhắc nhở   - Hệ thống hiển thị button trạng thái  disable. |

|  |  |  |
| --- | --- | --- |
|  |  | * Luồng A4: Trong loại nhắc nhở đã có danh sách nhắc nhở   - Hệ thống hiện thông báo yêu cầu ý kiến của người dùng. |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:



*Hình 2.7: Biểu đồ trình tự Quản lý loại nhắc nhở*

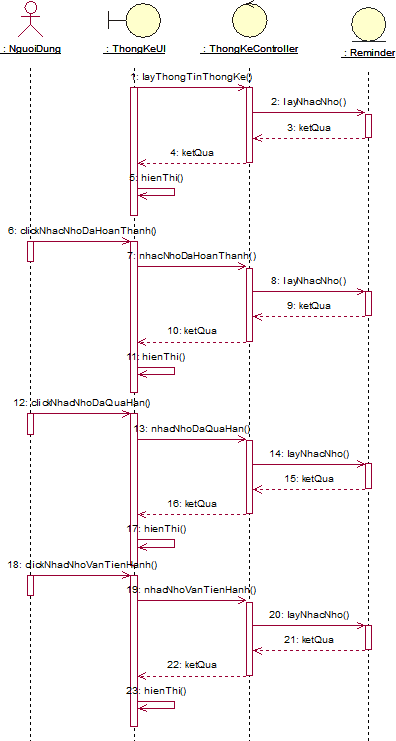
## Mô tả Usecase <Thống kê>

*Bảng 2.3: Mô tả usecase Thống kê*

|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase này cho phép người dùng xem thống kê các nhắc nhở đã hoàn thành, đã quá hạn và vẫn đang tiến hành ở mức độ nào |
| **Tác nhân** | | Người dùng |
| **Tiền điều kiện** | | Người dùng sử dụng ứng dụng |
| **Luồng sự kiện** | **Luồng sự kiện chính** | * Usecase bắt đầu khi người dùng kích hoạt ứng dụng. Khi đó trên màn hình chính của hệ thống hiển thị lên màn hình quản lý nhắc nhở. Người dùng click vào mục “Thống kê”, hệ thống hiển thị lên màn hình thống kê.   1. Xem danh sách nhắc nhở đã hoàn thành  - Người dùng chọn vào nhắc nhở đã hoàn thành. Hệ thống hiển thị lên màn hình danh sách nhắc nhở đã hoàn thành.  2. Xem danh sách nhắc nhở đã quá thời hạn  - Người dùng chọn vào nhắc nhở đã quá thời hạn. Hệ thống hiển thị lên màn hình danh sách nhắc nhở đã quá thời hạn chưa hoàn thành.  3. Xem danh sách nhắc nhở vẫn đang tiến hành  - Người dùng chọn vào nhắc nhở vẫn đang tiến hành. Hệ thống hiển thị lên màn hình |

|  |  |  |
| --- | --- | --- |
|  |  | danh sách nhắc nhở vẫn đang tiến hành theo loại nhắc nhở.   * Ca sử dụng kết thúc khi người dùng chọn chức năng khác hoặc thoát khỏi hệ thống. |
| **Luồng rẽ**  **nhánh** | Không có |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:



*Hình 2.8: Biểu đồ trình tự Thống kê*

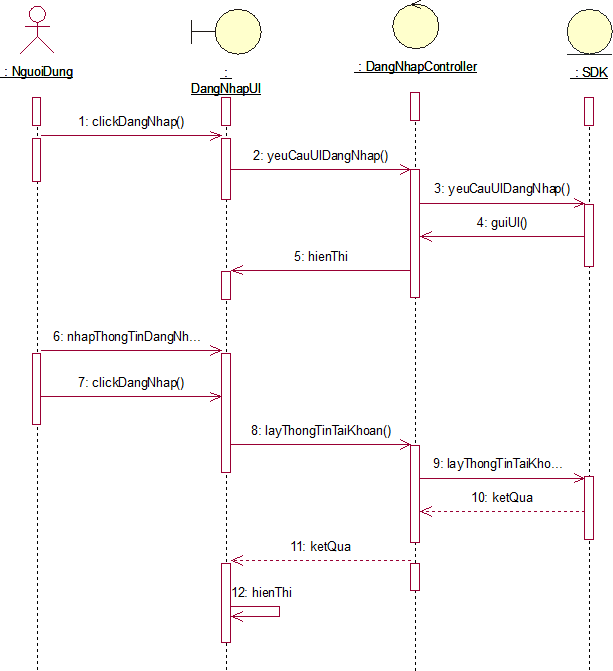
## Mô tả Usecase <Đăng nhập>

*Bảng 2.4: Mô tả usecase Đăng nhập*

|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase này cho phép người dùng đăng nhập để sử dụng chức năng đồng bộ dữ liệu. |
| **Tác nhân** | | Người dùng |
| **Tiền điều kiện** | | Người dùng sử dụng ứng dụng |
| **Luồng sự kiện** | **Luồng sự kiện chính** | * Usecase bắt đầu khi người dùng kích nút avatar trên màn hình home. Khi đó trên màn hình chính của hệ thống hiển thị lên một giao diện đăng nhập. * Đăng nhập: * Người dùng chọn đăng nhập trực tiếp với tài khoản Google hoặc Apple * Hệ thống sẽ mở cửa sổ đăng nhập từ Google SDK / Apple SDK * Người dùng tiến hành nhập thông tin trên màn hình. * Nếu đăng nhập thành công hệ thống sẽ nhận được trạng thái đăng nhập và thông tin của tài khoản sau đó trở về màn hình home. * Nếu đăng nhập không thành công hệ thống chuyển sang luồng A1.   + Ca sử dụng kết thúc khi người dùng đăng nhập thành công, hủy bỏ đăng nhập hoặc thoát khỏi hệ thống. |

|  |  |  |
| --- | --- | --- |
|  | **Luồng rẽ**  **nhánh** | * Luồng A1: Người dùng đăng nhập không thành công.   - Người dùng làm theo hướng dẫn của Google SDK / Apple SDK hoặc hủy bỏ đăng nhập. |
| **Hậu điều kiện** | | Nếu đăng nhập thành công người dùng sử dụng được chức năng đồng bộ dữ liệu. |

* Biểu đồ trình tự:



*Hình 2.9: Biểu đồ trình tự Đăng nhập*

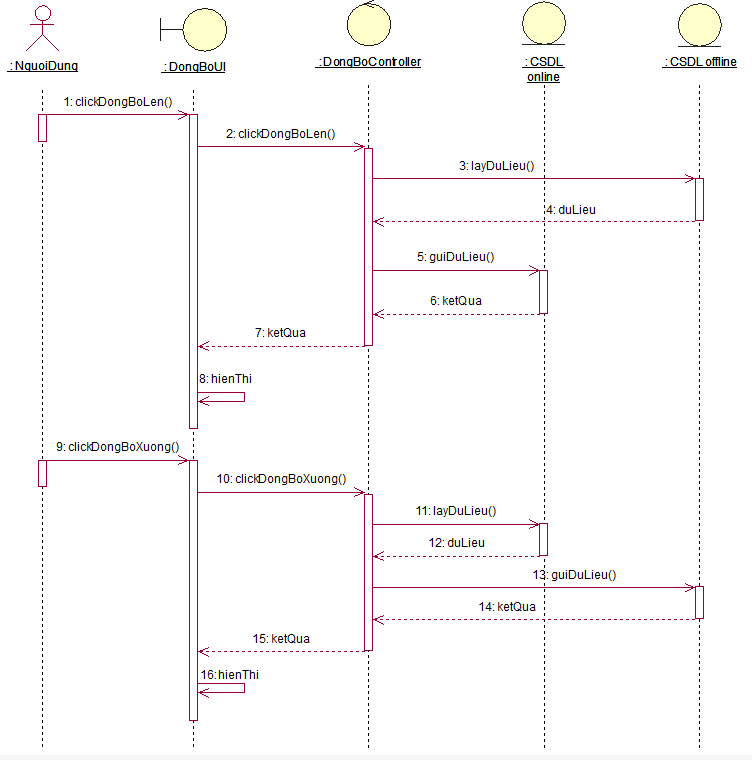
## Mô tả Usecase <Đồng bộ>

*Bảng 2.5: Mô tả usecase Đồng bộ*

|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase này cho phép người dùng đồng bộ dữ liệu giữa điện thoại và máy chủ |
| **Tác nhân** | | Người dùng |
| **Tiền điều kiện** | | Người dùng đăng nhập thành công vào hệ thống |
| **Luồng sự kiện** | **Luồng sự kiện chính** | * Usecase bắt đầu khi người dùng chọn vào button trong popup Account.  1. Đồng bộ từ máy lên sever:    * Người dùng chọn vào “Backup” trên màn hình.    * Dữ liệu sẽ được gửi từ máy lên sever và ghi đè lên sever.    * Đồng bộ thành công hiển thị thông báo.    * Trường hợp mất mạng hệ thống chuyển sang luồng A1. 2. Đồng bộ từ sever về máy:    * Người dùng chọn vào “Pull from server” trên màn hình.    * Máy sẽ download từ sever về thực hiện gộp cả dữ liệu sever và máy.    * Đồng bộ thành công hiển thị thông báo. |

|  |  |  |
| --- | --- | --- |
|  |  | * Trường hợp lỗi mạng hệ thống chuyển sang luồng A1.   + Ca sử dụng kết thúc khi người dùng chọn chức năng khác hoặc thoát khỏi hệ thống. |
| **Luồng rẽ**  **nhánh** | * Luồng A1: Trường hợp mất mạng khi đồng bộ   - Hệ thống hiển thị thông báo, người dùng đồng bộ lại khi có mạng. |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:



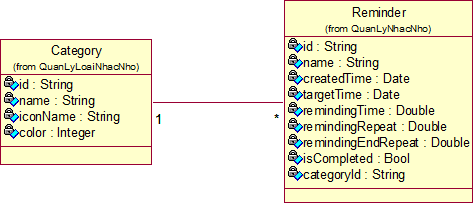
*Hình 2.10: Biểu đồ trình tự Đồng bộ dữ liệu*

## Mô tả Usecase <Thông báo>

*Bảng 2.6: Mô tả usecase Thông báo*

|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase cho phép người dùng cài đặt thông báo nhắc nhở lên trên màn hình |
| **Tác nhân** | | Người dùng |
| **Tiền điều kiện** | | Người dùng sử dụng ứng dụng |
| **Luồng sự kiện** | **Luồng sự kiện chính** | * Usecase bắt đầu khi người dùng sử dụng các chức năng của ứng dụng. * Thông báo: |
|  |  | - Người dùng nhập thông tin nhắc nhở và click tạo. |
|  |  | - Hệ thống thiết lập thời gian thông báo, nội dung thông báo. |
|  |  | - Khi đến thởi gian thì hệ thống sẽ thực hiện đẩy thông báo lên màn hình. |
|  | **Luồng rẽ**  **nhánh** | Không có |
| **Hậu điều kiện** | | Không có |

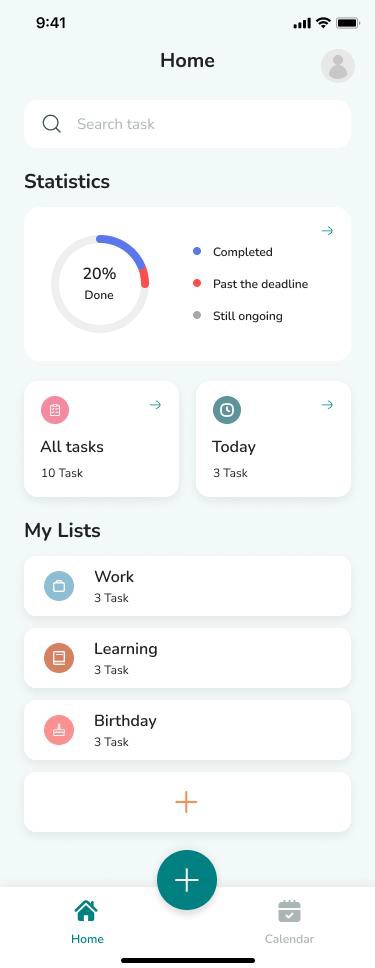
## Biểu đồ lớp hệ thống



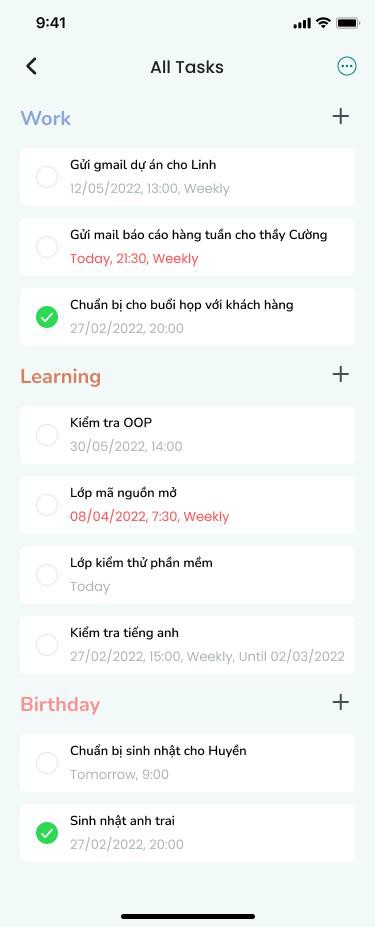
*Hình 2.11: Biểu đồ lớp hệ thống*

## Thiết kế giao diện

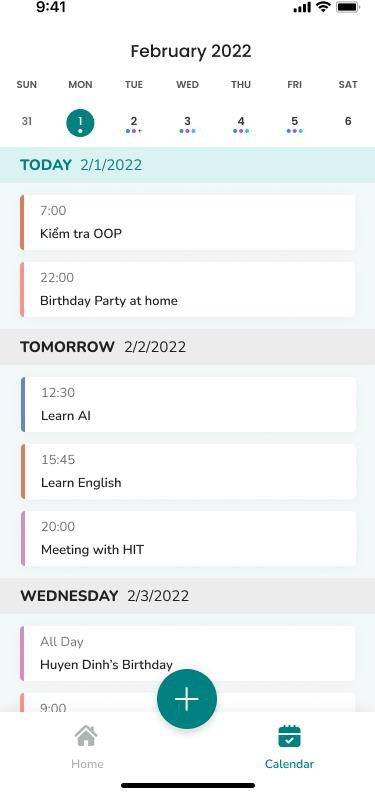
## Giao diện Quản lý nhắc nhở



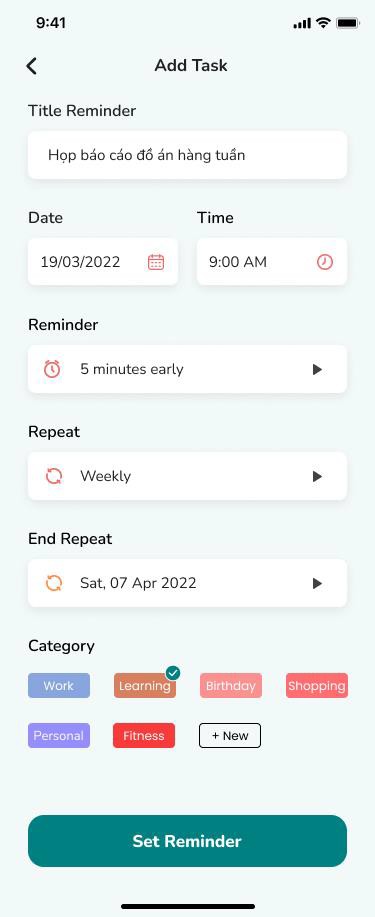
*Hình 2.12: Giao diện màn hình Trang chủ*



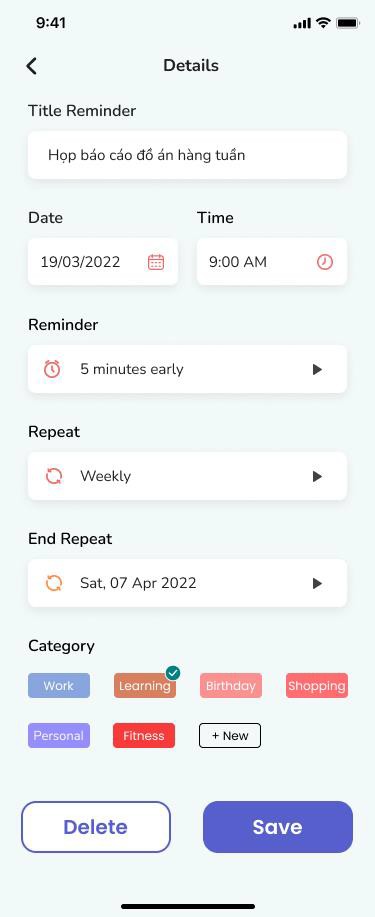
*Hình 2.13: Giao diện Hiển thị tất cả danh sách nhắc nhở*



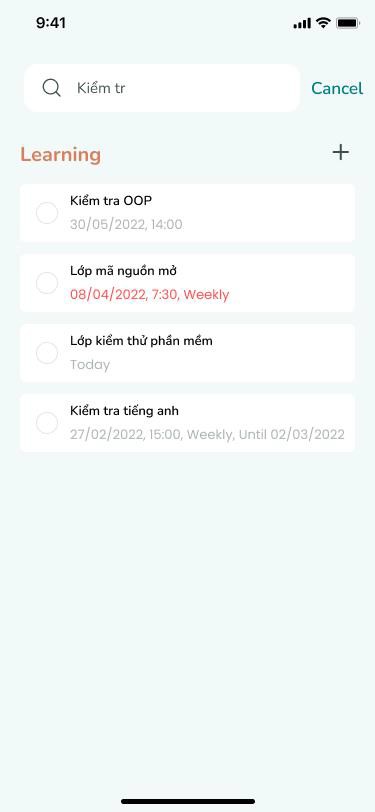
*Hình 2.14: Giao diện Hiển thị danh sách nhắc nhở theo ngày*



*Hình 2.15: Giao diện Thêm nhắc nhở*

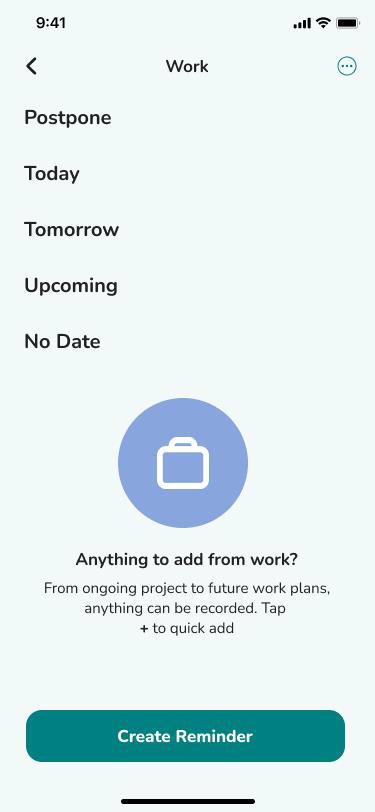


*Hình 2.16: Giao diện Chi tiết nhắc nhở, Sửa, Xóa nhắc nhở*

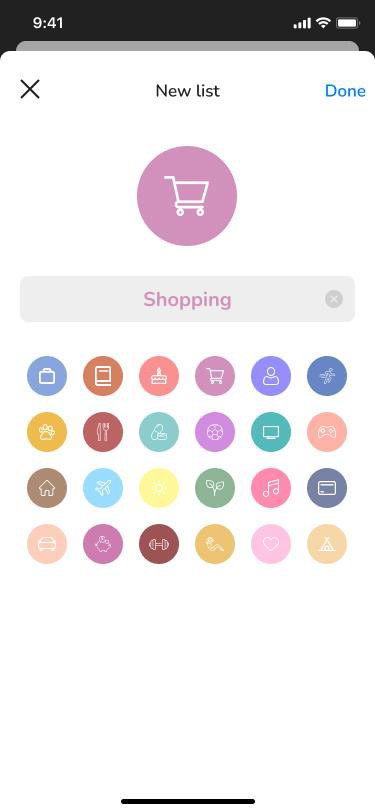


*Hình 2.17: Giao diện Tìm kiếm nhắc nhở*

## Giao diện Quản lý loại nhắc nhở



*Hình 2.18: Giao diện Hiện thị nhắc nhở theo loại*



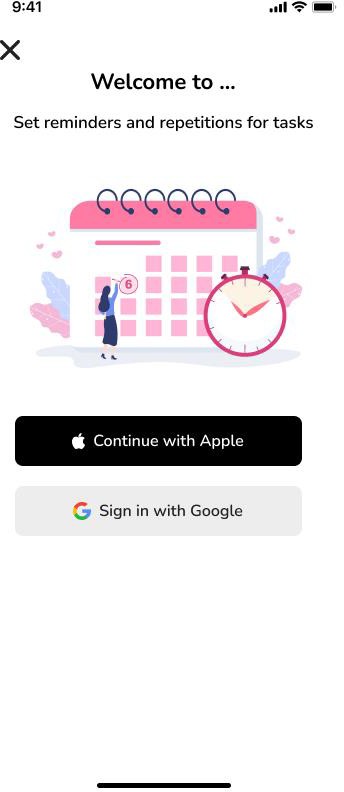
*Hình 2.19: Giao diện Thêm, Sửa loại nhắc nhở*

## Giao diện Thống kê



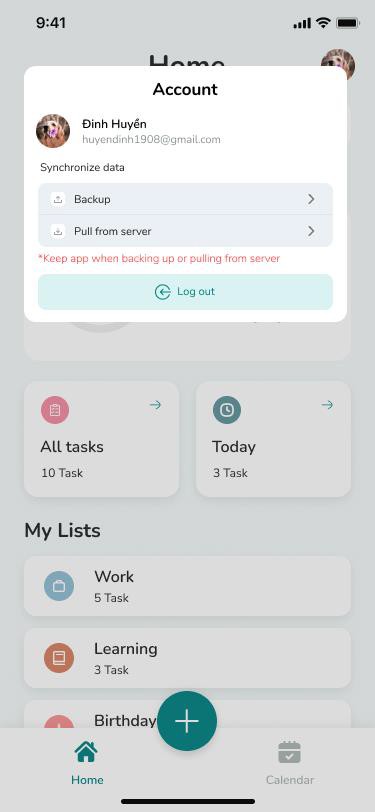
*Hình 2.20: Giao diện Hiện thị thống kê*

## Giao diện Đăng nhập



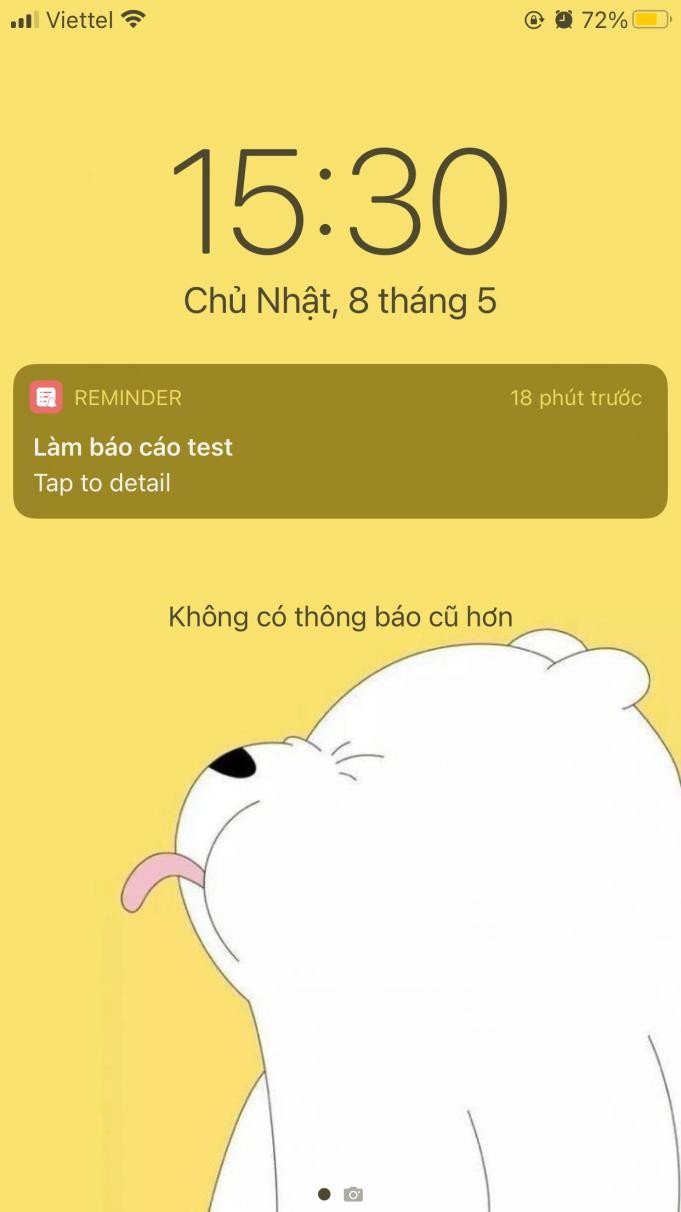
*Hình 2.21: Giao diện Đăng nhập*

## Giao diện Đồng bộ



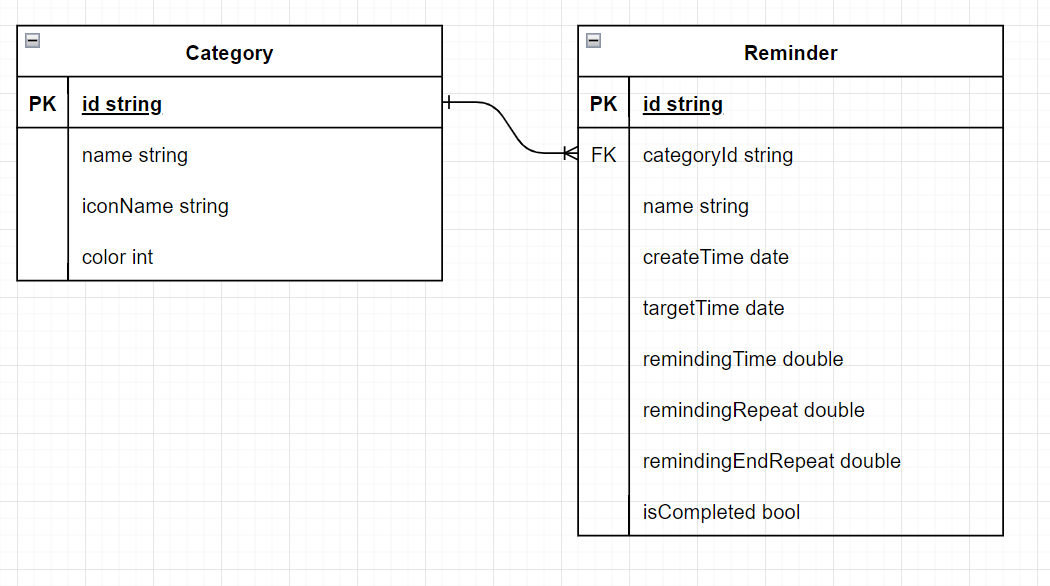
*Hình 2.22: Giao diện Đồng bộ dữ liệu*

## Giao diện Thông báo



*Hình 2.23: Giao diện Thông báo*

## Thiết kế cơ sở dữ liệu



*Hình 2.24: Cơ sở dữ liệu*

# CHƯƠNG 3. KẾT QUẢ

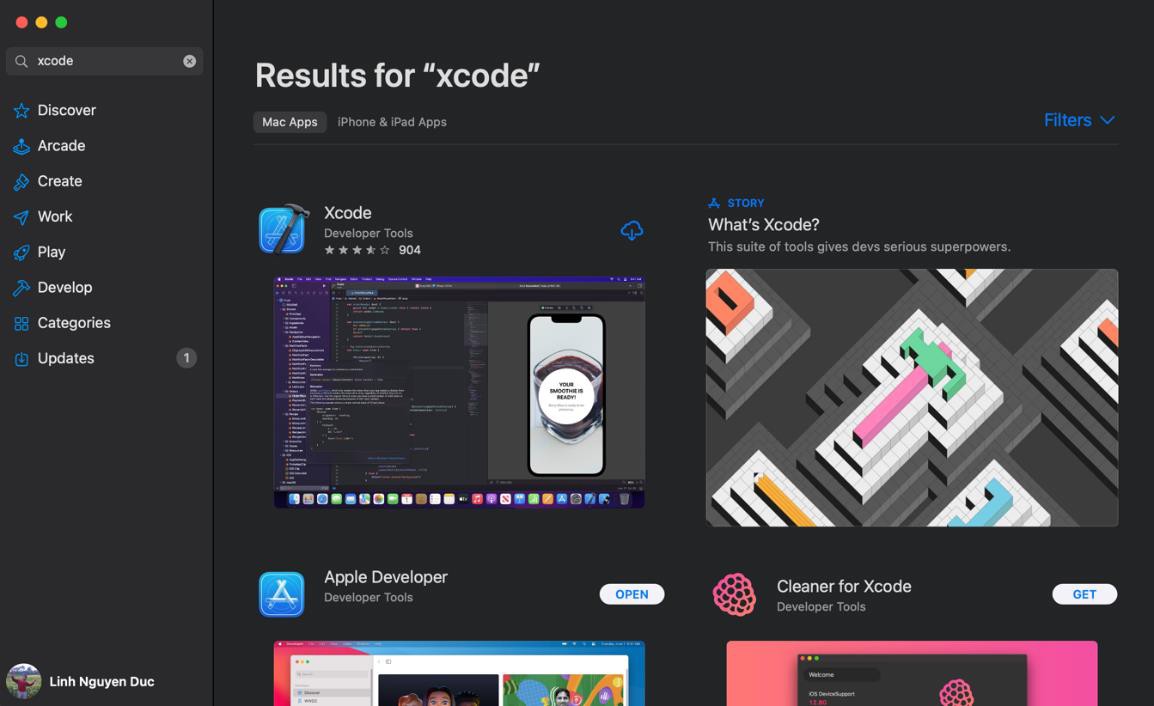
## Cài đặt môi trường

Trong quá trình cài đặt môi trường

* + - Cài đặt xcode

+ Bước 1: Mở Appstore, tìm kiềm Xcode trên thanh tìm kiếm

+ Bước 2: Ấn nút download để thực hiện tải và cài đặt phiên bản mới nhất của Xcode

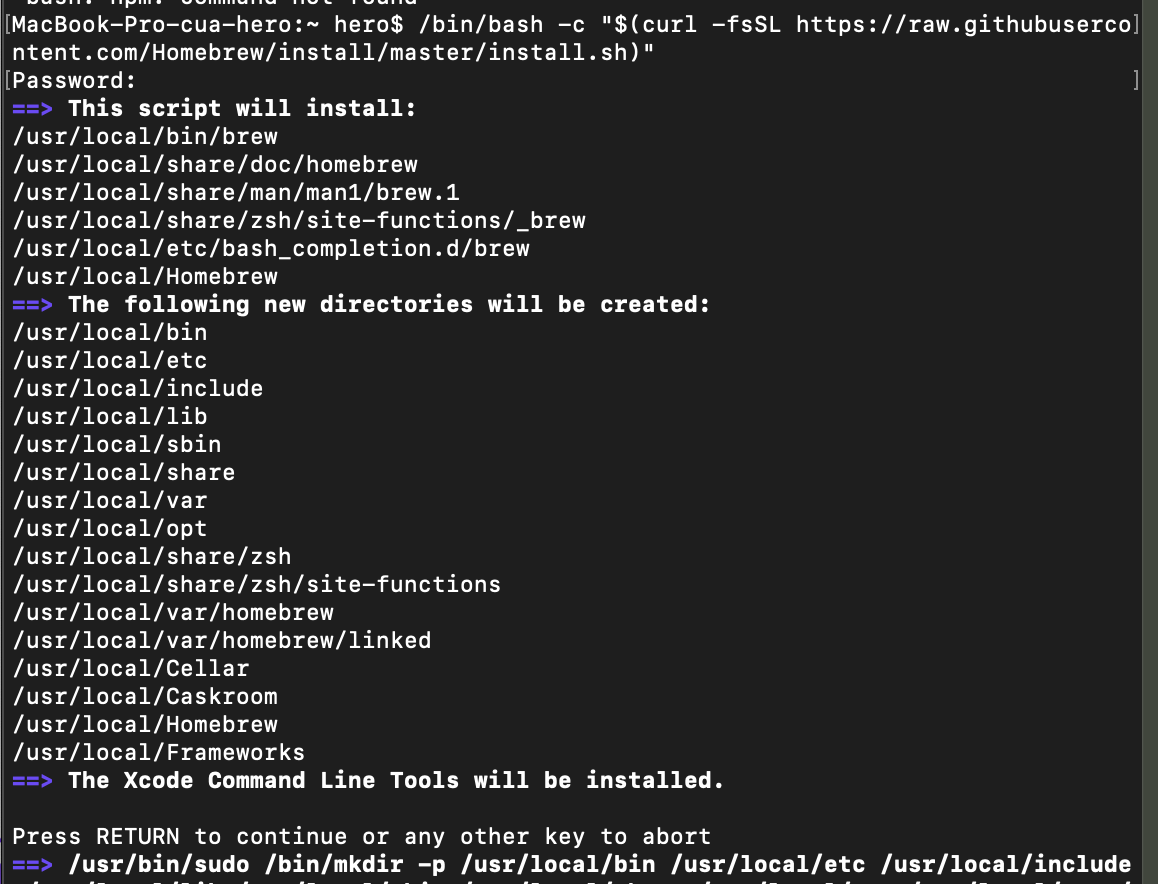


*Hình 3.1: Phần mềm Xcode trên Appstore*

- Cài đặt xcodegen

+ Bước 1: Mở terminal, thực hiện cài homebrew bằng đoạn lệnh sau

*/bin/bash -c "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install.sh>"*



*Hình 3.2: Cửa sổ terminal khi đang cài đặt homebrew*

Khi cài homebrew thành công, thực hiện brew -v để kiểm tra version của homebrew, đồng thời xác nhận việc cài homebrew là thành công.

+ Bước 2: Cài đặt xcodegen bằng đoạn lệnh *brew install xcodegen.*

+ Bước 3: Thực hiện kiểm tra version của xcodegen để đảm bảo quá trình cài đặt được thực hiện thành công bằng câu lệnh *xcodegen – version.*

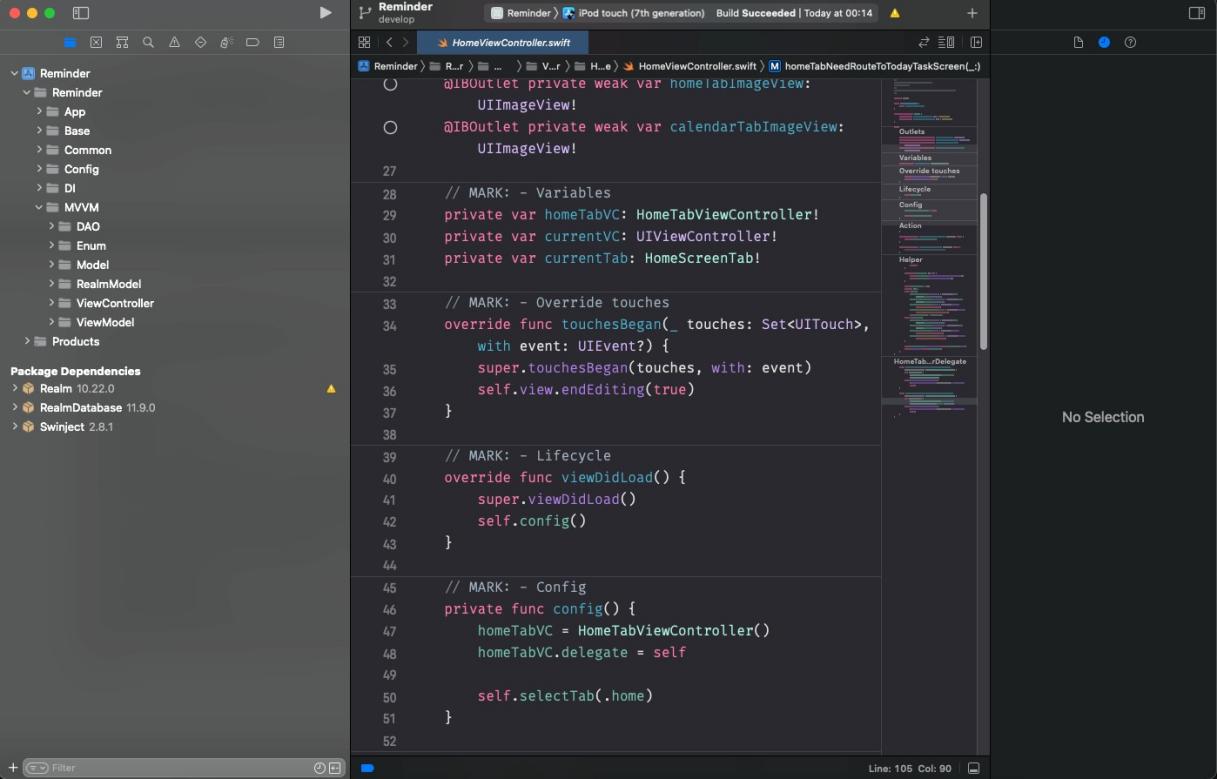
## Cài đặt chương trình

* + - Truy cập vào thư mục project bằng terminal
    - Cài Xcodegen để có generate ra file .xcodeproj. Nếu máy chưa cài xcodegen thì có thể cài thông qua câu lệnh “brew install xcodegen”
    - Chạy câu lệnh “xcodegen” để tạo ra file .xcodeproject



*Hình 3.3: Generate file .xcodeproj thành công*

* + - Mở file .xcodeproj vừa được tạo bằng xcode



*Hình 3.4: Giao diện project trên Xcode*

* + - Ở bên trái, ta thấy cấu trúc thư mục, tệp tin của project
    - Base: là nơi chứa base code, giúp cho việc tổ chức code, thuận tiện hơn
    - Common: chứa các object dùng cho toàn project, các extensions, resources, helper, …
    - MVVM: Nơi tổ chức code theo mô hình MVVM

+ DAO: Chứa các class DAO giúp cho việc tương tác với cơ sở dữ liệu.

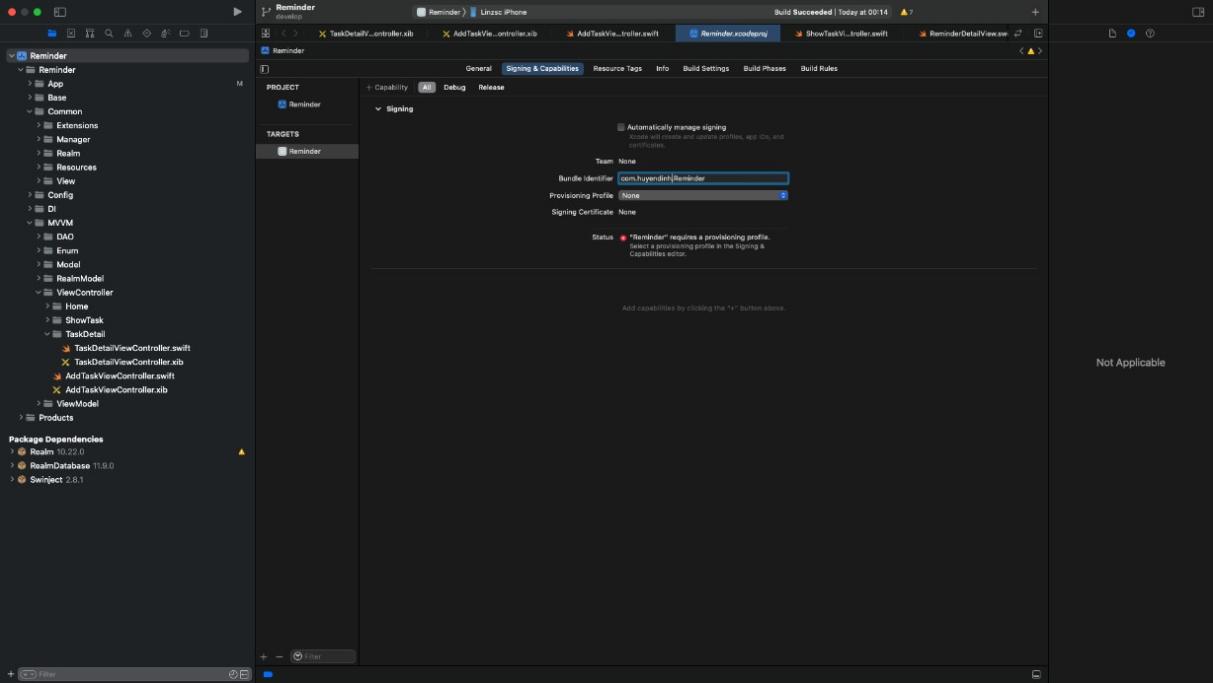
+ Enum: Nơi định nghĩa các enum.

+ Model: Nơi chứa model, là Model trong MVVM.

+ ViewController: nơi chứa các class ViewController và giao diện, chính là View trong MVVM.

+ ViewModel: nơi chứa các class ViewModel, là ViewModel trong MVVM.

* + - Để thực hiện build ứng dụng lên thiết bị di động hoặc thiết bị ảo, ta nhấn phím tắt command + R.



*Hình 3.5: Ứng dụng được build thành công trên thiết bị simulator*

## Kết quả thu được

Hệ thống được xây dựng với những chức năng, đó là:

* Quản lý nhắc nhở
* Quản lý loại nhắc nhở
* Thống kê
* Đăng nhập
* Đồng bộ dữ liệu
* Thông báo

## Kết quả kiểm thử

*Bảng 3.1: Kiểm thử các chức năng của ứng dụng*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Nội dung** | **Mục đích kiểm thử** | **Đầu vào** | **Đầu ra mong muốn** | **Kết quả** |
| 1 | Quản lý nhắc nhở | Kiểm tra chức năng  “Thêm nhắc nhở” | Nhập dữ liệu hợp lệ | Thêm mới  nhắc nhở thành công | Pass |
| Nhập dữ liệu không hợp lệ | Không thêm mới được nhắc nhở | Pass |
| 2 | Kiểm tra chức năng | Nhập dữ liệu hợp lệ | Cập nhật nhắc nhở thành công | Pass |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | “Sửa nhắc nhở” | Nhập dữ liệu không hợp lệ | Hiển thị thông báo yêu cầu nhập lại | Pass |
| 3 | Kiểm tra chức năng “Xóa nhắc nhở” | Đồng ý xóa | Xóa thành công nhắc nhở khỏi danh sách | Pass |
| 4 | Kiểm tra chức năng “Hiển thị  nhắc nhở” | Nhấn các button | Hiển thị đúng danh sách nhắc nhở  theo lựa chọn | Pass |
| 5 | Kiểm tra chức năng “ Tìm kiếm nhắc nhở” | Nhập tên nhắc nhở có tồn tại | Hiển thị danh sách nhắc nhở chứa ký  tự vừa nhập trên màn hình | Pass |
| Nhập tên nhắc nhở không tồn  tại | Không hiển thị bản ghi nào | Pass |
| 6 | Quản lý loại nhắc nhở | Kiểm tra chức năng “Hiển thị danh sách  loại nhắc nhở” | Có trên trang chủ | Hiển thị đủ danh sách loại nhắc nhở trên màn hình home | Pass |
| 7 | Kiểm tra chức năng “Thêm mới loại nhắc nhở” | Nhập dữ liệu hợp lệ | Thêm mới loại nhắc nhở  thành công | Pass |
| Nhập dữ liệu không hợp lệ | Hiển thị thông báo yêu cầu nhập  lại | Pass |
| 8 | Kiểm tra chức năng “Sửa loại nhắc nhở” | Nhập dữ liệu hợp lệ | Sửa thành công | Pass |
| Nhập dữ liệu không hợp lệ | Hiển thị thông báo yêu cầu nhập  lại | Pass |
| 9 | Kiểm tra chức năng | Đồng ý xóa | Xóa thành công loại nhắc nhở | Pass |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | “Xóa loại nhắc nhở” |  |  |  |
| 10 | Thống kê | Kiểm tra chức năng “Thống kê nhắc nhở” | Thống kê mức độ nhắc nhở | Hiển thị danh sách các thống kê nhắc nhở: đã hoàn thành;  quá hạn; vẫn tiếp tục | Pass |
| 11 | Đăng nhập | Kiểm tra chức năng “Đăng nhập  với Google” | Nhập thông tin tài khoản hợp lệ | Đăng nhập thành công hiện thị thông  tin trên màn hình Home | Pass |
|  |  | Nhập thông tin không hợp lệ | Hủy bỏ đăng nhập | Pass |
| 12 | Kiểm tra chức năng “Đăng nhập với Apple” | Chọn thông tin trên màn hình | Đăng nhập thành công hiện thị thông  tin trên màn hình Home | Pass |
| 13 | Đồng bộ dữ liệu | Kiểm tra chức năng “Đồng bộ  lên server” | Chọn nút đẩy lên | Đồng bộ lên thành công | Pass |
| 14 | Kiểm tra chức năng  “Đồng bộ từ server về  máy” | Chọn nút tải từ server | Đồng bộ thành công | Pass |
| 16 | Kiểm tra chức năng khi mất  mạng | Chọn nút đồng bộ | Hiển thị thông báo | Pass |
| 17 | Thông báo | Kiểm tra chức năng “Thông báo” | Tạo nhắc nhở thành công | Thông báo lên màn hình đúng thông tin, đúng giờ | Pass |

# KẾT LUẬN

## Đánh giá kết quả

* *Phần hoàn thành*
  + Về mặt công nghệ, em đã tìm hiểu và nắm bắt quy trình để xây dựng một ứng dụng trên iOS, tìm hiểu cách sử dụng trình soạn thảo Xcode.
  + Hoàn thiện bài khảo sát và phân tích thiết kế hệ thống “Ứng dụng nhắc nhở”.
  + Lập trình thành thạo với ngôn ngữ Swift và framework UIKit.
* *Phần chưa hoàn thành*
  + Cần hoàn thiện thêm giao diện để khách hàng dễ sử dụng.

## Hướng phát triển

* + Thiết kế và lập trình mở rộng các chức năng: Quét tài liệu, Thêm hình ảnh, chụp ảnh vào trong nhắc nhở; Đồng bộ tự động.
  + Hỗ trợ thêm chế độ Dark mode.
  + Hỗ trợ giao diện dạng widget cho ứng dụng trên màn hình chính của điện thoại.
  + Hỗ trợ giao diện ứng dụng trên watchOS, macOS.
  + Cải thiện sự mượt mà của ứng dụng. Nâng cao trải nghiệm người dùng.
  + Em rất mong tiếp tục nhận được sự giúp đỡ và tạo điều kiện của thầy cô và nhà trường để em có cơ hội phát triển, hoàn thiện ứng dụng tốt hơn trong thời gian tới.

# TÀI LIỆU THAM KHẢO

[1]. Tổ HTTT Đại học Công Nghiệp Hà Nội – Giáo trình hệ quản trị cơ sở dữ liệu SQL.

[2]. Tổ HTTT Đại học Công Nghiệp Hà Nội – Giáo trình phân tích thiết kế hệ thống.

[3]. [https://viblo.asia/p/tim-hieu-ve-mo-hinh-mvvm-maGK7vW95j2](https://viblo.asia/p/tim-hieu-ve-mo-hinh-mvvm-maGK7vW95j2?fbclid=IwAR3w4CeEylKpdyEWhW-f_-s8J4T2nPuQ6ZbKpRdp4grMaW2_oSI8gC6G-XE) [online].

[4]. [https://vi.wikipedia.org/wiki/Swift\_(ngôn\_ngữ\_lập\_trình)](https://vi.wikipedia.org/wiki/Swift_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)?fbclid=IwAR1QGHl43ESeotgoylgWwPBRGQvLUZLvYi0maYuDuhydzuNf2WcuTjxGpdQ) [online].

[5]. [https://developer.apple.com/documentation/uikit/](https://developer.apple.com/documentation/uikit/?fbclid=IwAR0Ikl1FoBvHR0V5r_UttCq57R-RNOOuWbiQempp2Dr0zHD_vGn43D0cZ2c) [online].

[6]. [https://medium.com/excellentweb/an-introduction-to-realm-database-](https://medium.com/excellentweb/an-introduction-to-realm-database-2881f0f8c231?fbclid=IwAR1RvWXysNyld5tWSFrB33Rei0AZwi0ntUL-CkGaKTBVAjKp4gPwxzVG5b0) [2881f0f8c231](https://medium.com/excellentweb/an-introduction-to-realm-database-2881f0f8c231?fbclid=IwAR1RvWXysNyld5tWSFrB33Rei0AZwi0ntUL-CkGaKTBVAjKp4gPwxzVG5b0) [online].

[7]. [https://www.mongodb.com/nosql-explained](https://www.mongodb.com/nosql-explained?fbclid=IwAR0fx2OzTuDBMA5PZRWRMcflrRMemVeQwSrrwHUFWREmkcoPVvuFQlVTkOU) [online].

[8]. [https://en.wikipedia.org/wiki/NoSQL](https://en.wikipedia.org/wiki/NoSQL?fbclid=IwAR1ma3a_qNSwTdUVTF1PsDvua3sE9Qb5RB2F1Y3zDFIjiRXKgtBgbuTJ9fw) [online].

[9]. [https://developer.apple.com/documentation/coregraphics](https://developer.apple.com/documentation/coregraphics?fbclid=IwAR1DgWvmMmlaXA3cfZE-Zkek1tbAl4P9mbjIHtj6TquhRXvtgKV-7GibZZ8) [online].