|  |  |  |  |
| --- | --- | --- | --- |
|  |  | | |
| VŨ BÌNH MINH | **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** | | |
|  | | |
| ĐỒ ÁN TỐT NGHIỆP  NGÀNH KỸ THUẬT PHẦN MỀM | | |
|  | | |
| **XÂY DỰNG ỨNG DỤNG BẢO MẬT CHO THIẾT BỊ DI ĐỘNG SAMSUNG KHI SỬ DỤNG BẢN DÙNG THỬ** | | |
|  | | |
|  | | |
| **CBHD:** | **TS. Đặng Trọng Hợp** | |
| KỸ THUẬT PHẦN MỀM | **Sinh viên:** | **Vũ Bình Minh** | |
| **Mã sinh viên:**  **Lớp:**  **Khóa:** | | **2019604575**  **2019DHKTPM03**  **14** |
|  | | |
|  | | |
|  | | |
| Hà Nội – Năm 2023 | | |
|  | | |

**MỤC LỤC**

**LỜI CẢM ƠN**

**MỞ ĐẦU**

**CHƯƠNG 1. TỔNG QUAN VỀ NỘI DUNG NGHIÊN CỨU**

**CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG**

**CHƯƠNG 3. KIỂM TRA HOẠT ĐỘNG CỦA HỆ THỐNG**

**KẾT LUẬN**

**PHỤ LỤC**

**TÀI LIỆU THAM KHẢO**

**DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **STT** | **Viết tắt** | **Dịch nghĩa** |
| 1 | SDK | Software Development Kit |
| 2 | FCM | Firebase Cloud Messaging |
| 3 | IDE | Integrated Development Environment |
| 4 | MVVM | Model - View - ViewModel |
| 5 | JVM | Java Virtual Machine |
| 6 | API | Application Programming Interface |
| 7 | XML | Extensible Markup Language |

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

LỜI CẢM ƠN

MỞ ĐẦU

## Lý do chọn đề tài

Rò rỉ thông tin về sản phẩm trước khi ra mắt trong những năm trở lại đây không còn là vấn đề quá xa lạ. Có nhiều thương hiệu sử dụng việc này như là một chiêu để tiếp thị, gây sự hiếu kỳ, hấp dẫn khách hàng. Tuy nhiên, đa số tổ chức bị tiết lộ thông tin quan trọng này bởi sự tấn công, đánh cắp dữ liệu từ bên ngoài. Điều này mang lại những hậu quả nặng nề, làm mất lợi thế của doanh nghiệp trên thị trường.

Ở Việt Nam, tình trạng ăn cắp ý tưởng và thông tin sản phẩm xảy ra rất phổ biến. Để bảo vệ ý tưởng kinh doanh và sản phẩm của mình, doanh nghiệp có thể tiến hành đăng ký Sở hữu trí tuệ. Tuy nhiên, Trung tâm bản quyền, Cục sở hữu trí tuệ bộ Khoa học Công nghệ chỉ chấp nhận đăng ký sản phẩm cụ thể, thương hiệu đã hiện có. Đây là một bất lợi đối với doanh nghiệp, bởi nhiều trường hợp đánh cắp thông tin xảy ra khi sản phẩm vẫn chưa được giới thiệu ra thị trường.

Giống như hầu hết các thương hiệu lớn khác, Samsung cũng thường xuyên bị rò rỉ các thông tin sản phẩm từ sớm, thậm chí trước cả thời điểm ra mắt nhiều tháng. Lần gần nhất Samsung chịu ảnh hưởng nặng nề vì các tin đồn và hình ảnh rò rỉ loạt sản phẩm ra mắt hồi đầu tháng 8 năm 2022 như Galaxy Z Fold 3, Galaxy Z Flip 3, Galaxy Buds 2. Mặc dù Samsung đã không ít lần gửi thư nhắc nhở đến các cá nhân, đơn vị lan truyền các hình ảnh của sản phẩm mới, thậm chí xử phạt hành chính với các nhân viên làm rò rỉ thông tin theo quy định tại hợp đồng lao động, nhưng điều này có vẻ không hiệu quả. Các tổ chức, cá nhân có thể lợi dụng việc dùng thử nghiệm thiết bị để đánh cắp thông tin sản phẩm.

Qua thời gian thực tập và làm việc tại Samsung, cá nhân em nhận thấy việc bảo mật thông tin sản phẩm mới còn nhiều mặt hạn chế. Từ đó, em đưa ra giải pháp phát triển ứng dụng tên là *Giải pháp bảo mật thiết bị* để giảm thiểu việc vi phảm bảo mật trong trạng thái dùng thử nghiệm thiết bị mới và nó sẽ được cài đặt trên tất cả các thiết bị mới trước khi đưa đến tay người dùng thử nghiệm. Vì vậy, em lựa chọn đề tài: “Xây dựng ứng dụng bảo mật cho thiết bị di động Samsung khi sử dụng bản dùng thử” là đề tài cho đồ án tốt nghiệp.

## Mục tiêu nghiên cứu

- Phân tích, hiểu nghiệp vụ bài toán thực tế.

- Nghiên cứu về Phát triển phần mềm trên thiết bị di động, Android SDK, Samsung Knox SDK, Google FCM.

- Phân tích và thiết kế hệ thống.

- Phát triển, kiểm thử và vận hành hệ thống đảm bảo hoạt động ổn định.

- Xây dựng báo cáo và tài liệu kèm theo.

## Đối tượng nghiên cứu

- Ngôn ngữ lập trình Java

- Công cụ lập trình Android Studio IDE

- Bộ công cụ phát triển phần mềm Android SDK, Samsung Knox SDK ,Google FCM.

## Phạm vi nghiên cứu

Sản phầm của đề tài này sẽ đáp ứng các yêu cầu sau:

- Hiển thị dấu mờ để ngăn chặn rò rỉ màn hình giao diện người dùng.

- Chặn truy cập mạng của các ứng dụng trong danh sách đen (tránh thu thập và chia sẻ thông tin cấu hình của thiết bị).

- Khóa thiết bị bằng FCM khi điện thoại bị tháo rời/ mất.

## 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:

* Phân tích các sản phẩm có chức năng tương tự.
* Phân tích và tiến hành xây dựng sản phẩm theo phân tích và yêu cầu thực tế.
* Tìm hiểu ngôn ngữ lập trình Android, công cụ lập trình Android Studio IDE, bộ công cụ phát triển phần mềm Android SDK, Samsung Knox SDK, Google FCM.
* 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

Hiện nay, công nghệ di động đang phát triển một cách vượt bậc. Các thiết bị đi động đang trở thành một phần thiết yếu cho tất cả các ngành công nghiệp trên toàn thế giới. Với sự hiện diện của mình, các thiết bị di động đã thúc đẩy hoạt động kinh doanh, doanh thu, lượng người tiêu dùng tăng lên theo cấp số nhân. Các công ty hoạt động trong lĩnh vực công nghệ không ngừng cải tiến sản phẩm cũ và nghiên cứu cho ra mắt sản phẩm mới đón đầu xu hướng. Việc cạnh tranh trở nên gay gắt, các chiêu trò thủ đoạn vô cùng tinh vi. Một trong những hành vi cạnh tranh không lành mạnh, trái pháp luật nhưng lại rất phổ biến hiện nay là ăn cắp ý tưởng.

Sở hữu ý tưởng kinh doanh mới hay những cải tiến trong sản phẩm là một lợi thế vô cùng lớn cho doanh nghiệp, quyết định sự phát triển của một thương hiệu trên thị trường. Ý tưởng về sản phẩm mới bị rò rỉ là mối nguy đối với doanh nghiệp. Các doanh nghiệp khác có thể sao chép ý tưởng và tung sản phẩm ra thị trường sớm hơn, có lợi thế trong hoạt động quảng bá sản phẩm tốt hơn. Doanh nghiệp sẽ đánh mất vị trí người tiên phong và sự chủ động của mình trên thị trường. Bên cạnh đó, thị phần của sản phẩm mới có thể bị ảnh hưởng nghiêm trọng khi ra mắt, khiến doanh thu bán hàng bị sụt giảm. Từ thực tế đó, ứng dụng bảo mật thông tin thiết bị trong trạng thái dùng thử ra đời để giải quyết các vấn đề liên quan đến rò rỉ thông tin sản phẩm mới sắp được ra mắt.

## 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.

**1.2.1.1. Native App**

Native app là một ứng dụng di động hoặc máy tính được phát triển và tối ưu hóa cho một nền tảng cụ thể, chẳng hạn như iOS (hệ điều hành của iPhone và iPad) hoặc Android (hệ điều hành của điện thoại thông minh và máy tính bảng sử dụng hệ điều hành Android của Google). Native app được viết bằng ngôn ngữ lập trình đặc thù của nền tảng đó, chẳng hạn Objective-C hoặc Swift cho iOS, hoặc Java hoặc Kotlin cho Android.

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 thể chạy được ở chế độ online hoặc 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).

**1.2.1.2. Hybrid App**

Hybrid app là một loại ứng dụng di động hoặc máy tính được phát triển bằng cách sử dụng các công nghệ đa nền tảng, cho phép chia sẻ mã nguồn giữa các nền tảng khác nhau, chẳng hạn như iOS, Android, và các nền tảng web. Hybrid app thường được viết bằng các ngôn ngữ web phổ biến như HTML, CSS, và JavaScript.

* 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:
  + Hiệu năng chậm .
  + 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.

**1.2.1.3. 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

Android Studio là Môi trường phát triển phần mềm tích hợp (IDE) chính thức để phát triển ứng dụng Android. Nó được ra mắt vào ngày 16 tháng 5 năm 2013 tại hội nghị Google I/O. Android Studio được phát hành miễn phí theo giấy phép Apache Licence 2.0. Android Studio ở giai đoạn truy cập xem trước sớm bắt đầu từ phiên bản 0.1 vào tháng 5 năm 2013, sau đó bước vào giai đoạn beta từ phiên bản 0.8 được phát hành vào tháng 6 năm 2014. Phiên bản ổn định đầu tiên được ra mắt vào tháng 12 năm 2014, bắt đầu từ phiên bản 1.0.

Dựa trên phần mềm IntelliJ IDEA của JetBrains, Android Studio được thiết kế đặc biệt để phát triển ứng dụng Android. Nó hỗ trợ các hệ điều hành Windows, Mac OS X và Linux, và là IDE chính thức của Google để phát triển ứng dụng Android gốc để thay thế cho Android Development Tools (ADT) dựa trên Eclipse.

### Ngôn ngữ lập trình Java

Java là một ngôn ngữ lập trình hướng đối tượng, dựa trên lớp được thiết kế để có càng ít phụ thuộc thực thi càng tốt. Nó là ngôn ngữ lập trình có mục đích chung cho phép các nhà phát triển ứng dụng viết một lần, chạy ở mọi nơi (WORA), nghĩa là mã Java đã biên dịch có thể chạy trên tất cả các nền tảng hỗ trợ Java mà không cần biên dịch lại. Các ứng dụng Java thường được biên dịch thành bytecode có thể chạy trên bất kỳ máy ảo Java (JVM) nào bất kể kiến trúc máy tính bên dưới. Cú pháp của Java tương tự như C và C++, nhưng có ít cơ sở cấp thấp hơn các ngôn ngữ trên. Java runtime cung cấp các khả năng động (chẳng hạn như phản ánh và sửa đổi mã thời gian chạy) thường không có sẵn trong các ngôn ngữ biên dịch truyền thống.

Java ban đầu được James Gosling tại Sun Microsystems (sau đó đã được Oracle mua lại) phát triển và được phát hành vào năm 1995 như một thành phần cốt lõi của nền tảng Java của Sun Microsystems. Các trình biên dịch Java, máy ảo và thư viện lớp thực thi gốc và tham chiếu ban đầu được Sun phát hành theo giấy phép độc quyền. Kể từ tháng 5 năm 2007, tuân theo các thông số kỹ thuật của Quy trình Cộng đồng Java, Sun đã cấp phép hầu hết các công nghệ Java của mình theo Giấy phép Công cộng GNU. Oracle cung cấp Máy ảo Java HotSpot của riêng mình, tuy nhiên việc triển khai tham chiếu chính thức là OpenJDK JVM, là phần mềm mã nguồn mở miễn phí và được hầu hết các nhà phát triển sử dụng và là JVM mặc định cho hầu hết các bản phân phối Linux.

### Application Framework trong lập trình Android

Application Framework trong Android là một bộ các thư viện và lớp cơ bản được cung cấp bởi hệ điều hành Android để phát triển ứng dụng di động. Nó cung cấp các công cụ và giao diện lập trình ứng dụng (API) cho các nhà phát triển để xây dựng ứng dụng Android.

Các thành phần chính của Application framework trong Android bao gồm:

* **Activities**: Là các thành phần giao diện người dùng của ứng dụng Android. Nó cho phép người dùng tương tác với ứng dụng thông qua các màn hình, các hoạt động (Activity) của ứng dụng.
* **Fragments**: Là một phần của một Activity, được sử dụng để tạo giao diện người dùng đa màn hình. Fragments cho phép phát triển ứng dụng Android linh hoạt hơn, với khả năng chia sẻ logic và giao diện người dùng giữa các màn hình khác nhau.
* **Views**: Là các thành phần giao diện người dùng như TextView, Button, EditText, ListView, RecyclerView, và nhiều hơn nữa. Views cho phép hiển thị và tương tác với dữ liệu trong giao diện người dùng của ứng dụng Android.
* **Intents**: Là cơ chế để giao tiếp giữa các thành phần khác nhau của ứng dụng Android hoặc giữa các ứng dụng khác nhau. Intents cho phép chuyển dữ liệu và thực hiện các hoạt động như mở một Activity mới hoặc gửi một thông báo.
* **Content Providers**: Là các thành phần dùng để quản lý và cung cấp dữ liệu cho ứng dụng Android, chẳng hạn như lưu trữ dữ liệu trong cơ sở dữ liệu SQLite hoặc chia sẻ dữ liệu giữa các ứng dụng khác nhau.
* **Services**: Là các thành phần chạy nền của ứng dụng Android, cho phép thực hiện các hoạt động không liên quan đến giao diện người dùng, chẳng hạn như phát nhạc, tải dữ liệu từ mạng, hoặc xử lý công việc dài hạn.
* **Notification Manager**: Là các thành phần chạy nền của ứng dụng Android, cho phép thực hiện các hoạt động không liên quan đến giao diện người dùng, chẳng hạn như phát nhạc, tải dữ liệu từ mạng, hoặc xử lý công việc dài hạn.

Ngoài ra, Application framework trong Android còn cung cấp các công cụ và giao diện lập trình khác như Resource Manager, Package Manager, Activity Manager, và System Services, giúpngười phát triển xây dựng ứng dụng Android với nhiều tính năng phong phú và đa dạng, bao gồm đồ họa, âm thanh, định vị, mạng, quản lý tài nguyên, quản lý giao diện người dùng, và quản lý vòng đời ứng dụng.

Một số công nghệ và thư viện khác trong Application framework của Android bao gồm:

* **Android UI Toolkit**: Cung cấp các công cụ và thư viện để xây dựng giao diện người dùng trong Android, bao gồm XML layouts, các lớp View, Drawable, Animator, và nhiều tính năng đồ họa và tương tác khác.
* **Android Resource Manager**: Quản lý tài nguyên trong ứng dụng Android như hình ảnh, âm thanh, dữ liệu chuỗi, màu sắc, kiểu chữ, và các tài nguyên khác.
* **Android Location API**: Cung cấp khả năng định vị địa lý cho ứng dụng Android, cho phép định vị vị trí của thiết bị, lấy dữ liệu về vị trí, hoặc theo dõi vị trí của thiết bị.
* **Android Networking API**: Cung cấp các công cụ và thư viện để thực hiện các kết nối mạng trong ứng dụng Android, bao gồm HTTP, WebSocket, gửi và nhận dữ liệu qua mạng.
* **Android Multimedia API**: Cung cấp khả năng xử lý đa phương tiện như hình ảnh, âm thanh, và video trong ứng dụng Android, bao gồm phát lại, ghi âm, xử lý hình ảnh, và các tính năng đa phương tiện khác.
* **Android Database API:** Cung cấp các công cụ và thư viện để quản lý cơ sở dữ liệu trong ứng dụng Android, bao gồm SQLite và Content Providers để lưu trữ và truy vấn dữ liệu.
* **Android Notification API**: Cung cấp khả năng hiển thị thông báo trong Android, bao gồm tạo, quản lý, và tương tác với các thông báo đẩy hoặc thông báo trong hệ thống.

### Samsung Knox SDK

Samsung Knox là giải pháp bảo mật dành cho doanh nghiệp trên di động được cung cấp bởi Samsung. Knox SDK mở rộng chức năng của SDK Android tiêu chuẩn để cung cấp quyền truy cập chi tiết vào các tính năng của thiết bị, tùy chọn bảo mật, cài đặt tùy chỉnh và hơn thế nữa. Tạo các giải pháp phù hợp bằng cách ánh xạ lại các phím phần cứng, thiết kế kiốt, triển khai chính sách theo vị trí địa lý và tùy chỉnh hoạt ảnh khởi động. Giữ an toàn cho dữ liệu nhạy cảm của doanh nghiệp bằng cách hạn chế quyền truy cập vào cài đặt, định cấu hình trước cài đặt VPN và tường lửa, cho phép và chặn ứng dụng.

### NoSQL Database và Cloud Firestore

**1.2.6.1. 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).

*a. 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.

*b. 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 dễdàng 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.

*c. 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 hệlà 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.

*d. Document Oriented databases*

Tài liệu là nguyên lý chính của cơ sở dữ liệu kiểu dữ liệu. Dữ liệu lưu trữvà lấy ra là các tài liệu với định dạng XML, JSON, BSON,… Tài liệu miêu tảchính 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

**1.2.6.2. Cloud Firestore**

Cloud Firestore là một cơ sở dữ liệu NoQuery được lưu trữ trên đám mây mà các ứng dụng IOS, Android, Web có thể truy cập trực tiếp thông qua SDK.Cloud Firestore cũng có sẵn trong Node.js, Java, Python và Go SDKs, REST và RPC APIs. Cloud Firestore được tổ chức theo mô hình dữ liệu NoQuery của Cloud Firestore, dữ liệu lưu trong các document ánh xạ tới các giá trị. Các document này được lưu trữ trong các collection cho bạn tổ chức dữ liệu và thực hiện truy vấn. Các tính năng chính của Cloud Firestore:

* Tính linh hoạt: Cloud Firestore hỗ trợ các cấu trúc dữ liệu linh hoạt, phân cấp dữ liệu. Lưu trữ dữ liệu của bạn trong các document, được tổ chức thành các collection. Các document có thể chứa các đối tượng phức tạp.
* Truy vấn tượng trưng: Bạn có thể sử dụng các truy vấn để truy xuất các document riêng lẻ hoặc để truy xuất tất cả các document trong collection khớp với các tham số truy vấn của bạn. Các truy vấn của bạn có thể bao gồm nhiều bộ lọc, kết hợp giữa bộ lọc và sắp xếp.
* Cập nhật thời gian thực: Cloud Firestore sử dụng đồng bộ hóa dữ liệu để cập nhật dữ liệu trên mọi thiết bị được kết nối. Nó cũng được thiết kế để thực hiện các truy vấn tìm nạp một lần .
* Hỗ trợ offline: Cloud Firestore lưu trữ dữ liệu tại local, vì vậy ứng dụng có thể viết, đọc, nghe và truy vấn dữ liệu ngay cả khi thiết bị ngoại tuyến. Khi thiết bị trở lại trực tuyến, Cloud Firestore sẽ đồng bộ hóa mọi thay đổi cục bộ lên Cloud Firestore.
* Khả năng mở rộng: Mang đến khả năng từ Google Cloud Platform thiết kế để sử dụng cơ sở dữ liệu khó khăn nhất từ các ứng dụng lớn nhất thế giới.

### Google Firebase Cloud Message

Firebase Cloud Messaging (FCM) là một dịch vụ gửi thông báo, tin nhắn đa nền tảng được cung cấp bởi Google, cho phép bạn gửi tin nhắn, thông báo một cách đáng tin cậy và hoàn toàn miễn phí tới các thiết bị đã được đăng ký với FCM.

Nguyên tắc hoạt động: Các thiết bị client sẽ đăng ký device\_token lên cho FCM. Các thông báo, tin nhắn được soạn và gửi từ một ứng dụng, từ Notifications composer của firebase cung cấp, .... FCM sẽ nhận những thông báo này và xử lý gửi về các thiết bị đã đăng ký với FCM từ trước. Khi các thiết bị có kết nối mạng thì thông báo sẽ được gửi về ứng dụng thành công.

Với FCM, có 2 loại thông báo, tin nhắn mà bạn có thể gửi tới ứng dụng, đó là:

* Notification messages: Đôi khi được gọi là "thông báo (tin nhắn) hiển thị", chúng được xử lý tự động bởi FCM SDK. Notification messages chứa các key dữ liệu đã được định nghĩa trước. Sử dụng Notification messages khi bạn chỉ muốn hiển thị các thông báo đến các ứng dụng clients.
* *Data messages:* Là thông báo (tin nhắn) sẽ được xử lý bởi các ứng dụng client. Data messages chứa các cặp key - value do người dùng định nghĩa. Sử dụng Data messages khi bạn muốn xử lý các thông báo trên chính ứng dụng của bạn.

Thông báo Firebase hoạt động khác nhau tùy thuộc vào trạng thái của ứng dụng. Trong android, để nhận Firebase message thì cần phải tạo một service và extend FirebaseMessagingService. Để xử lý các thông báo nhận được bạn cần override phương thức onMessageReceived. Phương thức này xử lý được hầu hết các loại tin nhắn, ngoại trừ các trường hợp sau:

* Nhận Notification messages khi ứng dụng đang ở trạng thái background. Trong trường hợp này, thông báo sẽ được gửi đến khay hệ thống của thiết bị, khi người dùng chạm vào thông báo sẽ mở trình khởi chạy ứng dụng mặc định.
* Nhận thông báo có chứa cả Notification messages và Data messages khi ứng dụng đang ở trạng thái background. Trong trường hợp này, phần Notification sẽ được gửi đến khay hệ thống, còn phần Data sẽ được sử dụng cho trình khởi chạy các Activity khi chạm vào thông báo.

### 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:

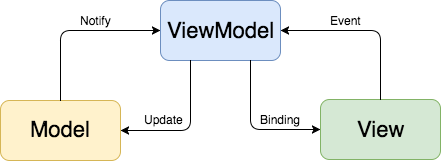
Tương tự như trong mô hình MVC, View là phần giao diện của ứng dụng để hiển thị dữ liệu và nhận tương tác của người dùng. Một điểm khác biệt so với các ứng dụng truyền thống là View trong mô hình này tích cực hơn, nó có khả năng thực hiện các hành vi và phản hồi lại người dùng thông qua tính năng binding, 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.



* 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

## Giới thiệu về hệ thống

Qua thời gian thực tập và làm việc tại Samsung SRV, cá nhân em nhận thấy việc bảo mật thông tin sản phẩm mới là một vấn đề quan trọng. Từ đó, em đưa ra giải pháp cần phát triển một ứng dụng bảo mật để giảm thiểu việc vi phảm bảo mật trong trạng thái dùng thử nghiệm thiết bị mới và nó sẽ được cài đặt trên tất cả các thiết bị mới trước khi đưa đến tay người dùng thử nghiệm. Ứng dụng sẽ cho phép người quản trị quản lý các thiết bị thử nghiệm bằng cách in hình mờ, chặn các ứng dụng trong danh sách đen để ngăn rò rỉ thông tin, theo dõi vị trí thiết bị và khóa thiết bị từ xa khi thiết bị thử nghiệm bị tháo rời/ mất.

## Phân tích yêu cầu

### Về hệ thống

Ứng dụng bảo mật sẽ hướng tới một hệ thống có giao diện đẹp, dễ sử dụng, tối ưu hiệu suất và tiết kiệm thời gian nhất để người quản trị có thể quản lý dễ dàng.

### Về người sử dụng

Người quản trị có thể dễ dàng nắm bắt được thông tin, trạng thái của các thiết bị đang được đưa người dùng thử nghiệm. Ngoài ra, người quản trị cí thể khóa từ xa và truy vết các thiết bị bị tháo rời/mất một cách nhanh chóng, chính xác.

### Về chức năng

* Các chức năng chính của hệ thống
* Thông báo đăng nhập
* In hình mờ
* Chặn ứng dụng trong danh sách đen
* Theo dõi vị trí thiết bị
* Khóa thiết bị từ xa
* Yêu cầu phi chức năng
* Giao diện đồ họa
* Hoạt động của hệ thống
* Ngôn ngữ
* Độ bảo mật của ứng dụng
* Hiệu năng của ứng dụng

## Thiết kế hệ thống

### Mô hình hóa Usecase

**2.3.1.1. Biểu đồ usecase chính**

****

**2.3.1.2. Biểu đồ usecase thứ cấp**

*a. Phân rã UC Xem danh sách ứng dụng trong danh sách đen*

**

*b. Phân rã UC Xem danh sách thiết bị thử nghiệm*

**

*c. Phân rã UC Thay đổi Policy*

**

### Mô tả chi tiết usecase

**2.3.2.1. Mô tả UC Xem Danh sách ứng dụng trong danh sách đen**

|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Use case này cho phép người dùng xem danh sách các ứng dụng trong danh sách đen |
| **Tác nhân** | | Người dùng |
| **Tiền điều kiện** | | Người dùng đăng nhập ứng dụng |
| **Luồng sự kiện** | **Luồng sự kiện chính** | Use case này bắt đầu khi người dùng chọn tùy chọn “Black List” trong giao diện chính của ứng dụng. Khi đó, danh sách các ứng dụng trong danh sách đen hiện lên màn hình.  - Người dùng chọn một ứng dụng trong danh sách, hệ thống sẽ hiển thị thông tin chi tiết của ứng dụng bao gồm tên ứng dụng, kích cỡ, danh mục, nhà phát triển.  Ca sử dụng kết thúc khi người dùng chọn quay lại hoặc thoát khỏi hệ thống. |
| **Luồng rẽ nhánh** | Không có |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:



**2.3.2.2. Mô tả UC Xem Danh sách thiết bị thử nghiệm**

|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Use case này cho phép người quản trị xem danh sách các thiết bị trong trạng thái người dùng dùng thử nghiệm |
| **Tác nhân** | | Người quản trị |
| **Tiền điều kiện** | | Người quản trị đăng nhập vào hệ thống |
| **Luồng sự kiện** | **Luồng sự kiện chính** | Use case này bắt đầu khi người dùng đăng nhập vào ứng dụng. Khi đó, màn hình chính sẽ hiển thị danh sách các thiết bị đang trong trạng thái dùng thử nghiệm.  - Người dùng chọn một thiết bị trong danh sách, hệ thống sẽ hiển thị thông tin chi tiết của thiết bị bao gồm số serial, kiểu máy,vị trí hiện tại…lên màn hình.  Ca sử dụng kết thúc khi người quản trị chọn quay lại hoặc thoát khỏi hệ thống. |
| **Luồng rẽ nhánh** | Không có |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:



**2.3.2.3. Mô tả UC Thay đổi policy**

|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase này cho phép người quản trị thay đổi chính sách bảo mật trên thiết bị thử nghiệm |
| **Tác nhân** | | Người quản trị |
| **Tiền điều kiện** | | Người quản trị đăng nhập vào hệ thống |
| **Luồng sự kiện** | **Luồng sự kiện chính** | Use case này bắt đầu khi người dùng chọn “Thay đổi policy” trong giao diện xem chi tiết thiết bị. Khi đó, màn hình chính sẽ hiển thị danh sách các chính sách áp dụng cho thiết bị thử nghiệm hiện tại:  1. Yêu cầu đăng nhập  - Người quản trị bật áp dụng yêu cầu đăng nhập, ứng dụng sẽ khởi động một dịch vụ kiểm tra trạng thái đăng nhập của người dùng, nếu người dùng chưa thực hiện đăng nhập thì hiện thông báo yêu cầu đăng nhập mỗi phút một lần.  2. In hình mờ  - Người quản trị bật áp dụng in hình mờ, một chuỗi chứa thông tin mã hóa của thiết bị sẽ được chèn mờ vào widget nhằm ngăn chặn rò rỉ giao diện (khi giao diện bị quay/chụp trộm phát tán lên mạng xã hội, người quản trị có thể dựa vào thông tin mã hóa trên màn hình để truy vết người dùng đang dùng thử thiết bị).  3. Chặn ứng dụng trong danh sách đen  - Người quản trị bật áp dụng chặn ứng dụng trong danh sách đen, ứng dụng sẽ khởi động một dịch vụ để lắng nghe trạng thái của các ứng dụng. Khi ứng dụng trong danh sách đen được mở, hệ thống ngay lập tức tắt ứng dụng đó.  4. Theo dõi vị trí thiết bị  - Người quản trị bật áp dụng theo dõi vị trí thiết bị, ứng dụng sẽ khởi động một dịch vụ: sau khi người dùng đăng nhập, ứng dụng sẽ cập nhật vị trí hiện tại của thiết bị mỗi 30 phút 1 lần.  Người quản trị chọn “Apply”, hệ thống sẽ cập nhật và gửi policy mới đến thiết bị qua FCM. Knox SDK nhận tin nhắn cập nhật policy từ FCM sẽ thực hiện xử lý áp dụng policy mới và cập nhật policy lên firestore. |
| **Luồng rẽ nhánh** | Không có |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:



**2.3.2.4. Mô tả UC Khóa thiết bị từ xa**

|  |  |  |
| --- | --- | --- |
| **Mô tả** | | Usecase này cho phép người quản trị khóa thiết bị từ xa khi có dấu hiệu bị tháo rời/ mất. |
| **Tác nhân** | | Người quản trị |
| **Tiền điều kiện** | | Người quản trị đăng nhập vào hệ thống |
| **Luồng sự kiện** | **Luồng sự kiện chính** | 1. Người quản trị chọn “Lock Device” trên giao diện xem chi tiết thiết bị.  - Hệ thống sẽ một gửi tin nhắn khóa qua FCM đến thiết bị.  - Knox SDK nhận tin nhắn khóa từ FCM và thực hiện khóa thiết bị bằng mật khẩu.  2. Người quản trị chọn “Unlock Device” trên giao diện xem chi tiết thiết bị.  - Hệ thống sẽ một gửi tin nhắn mở khóa qua FCM đến thiết bị.  - Knox SDK nhận tin nhắn mở khóa từ FCM và thực hiện mở khóa thiết bị. |
|  | **Luồng rẽ nhánh** | Không có |
| **Hậu điều kiện** | | Không có |

* Biểu đồ trình tự:

****

## Thiết kế giao diện

# CHƯƠNG 3. KẾT QUẢ