|  |  |
| --- | --- |
| **logo (CMYK)-01** | **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP.HCM** |

**APP TRẮC NGHIỆM LỊCH SỬ VIỆT NAM**

Ngành: **CÔNG NGHỆ THÔNG TIN**

Môn học : Lập trình trên thiết di dộng

Giảng viên hướng dẫn : Trần Duy Thanh

Sinh viên thực hiện : MSSV :Lớp

Võ Vĩnh Nghi 1615060037 16HTH01

Phạm Tuân 1512360164 15CTH02

Lê Nguyễn Minh Tân 1511060629 15DTH04

Nguyễn Quốc Thái 1511061529 15DTH11

Lý Bảo Long 1411061276 14DTHC05

**TP. Hồ Chí Minh, 2019**

## LỜI CẢM ƠN

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều. Trong suốt thời gian từ khi bắt đầu học tập tại trường đến nay, em đã nhận được nhiều sự quan tâm, giúp đỡ của quý Thầy (Cô), gia đình và bạn bè.

Sau hơn thời gian học tập và nghiên cứu đề tài môn học, em đã được mở mang kiến thức và được hiểu biết nhiều hơn về lĩnh vực công nghệ thông tin qua sự quan tâm, hướng dẫn tận tình của Thầy.

Xin chân thành cảm ơn Thầy Trần Duy Thanh– Giảng viên hướng dẫn, đã hướng dẫn em tận tình trong suốt thời gian học tập, em chân thành cảm ơn Thầy đã nhiệt tình truyền đạt những kinh nghiệm quý báu và hơn hết là tình cảm mà Thầy dành cho em trong thời gian hoàn thành môn học.

Trong quá trình học tập, cũng như là trong quá trình làm bài báo cáo do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp của Thầy để em học thêm được nhiều kinh nghiệm.

Xin Chân Thành Cảm Ơn !

## MỤC LỤC

[CHƯƠNG 1: TỔNG QUAN 1](#_Toc4268718)

[1.1 GIỚI THIỆU PHẦN MỀM TRẮC NGHIỆM LỊCH SỬ 1](#_Toc4268719)

[1.2 ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU 2](#_Toc4268720)

[1.3 MỤC TIÊU ĐỀ TÀI 2](#_Toc4268721)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 3](#_Toc4268722)

[2.1 CÁC KHÁI NIỆM 3](#_Toc4268723)

[**2.1.1** **Hệ Điều Hành Mở** 3](#_Toc4268724)

[**2.1.2** **Các ứng dụng có sẵn trên Android** 3](#_Toc4268725)

[**2.1.3** **Truy cập phần cứng** 4](#_Toc4268726)

[**2.1.4** **Dịch vụ chạy nền** 4](#_Toc4268727)

[**2.1.5** **SQL Lite Database** 4](#_Toc4268728)

[**2.1.6** **Tối ưu hóa bộ nhớ và quản lí tiến trình** 4](#_Toc4268729)

[**2.1.7 Android Software Development Kit ( SDK)** 5](#_Toc4268730)

[**2.1.8** **Cấu trúc ứng dụng** 5](#_Toc4268731)

[**2.1.9** **Các thư viện của Android:** 6](#_Toc4268732)

[2.2 CƠ CHẾ HOẠT ĐỘNG 9](#_Toc4268733)

[2.2.1 Các hoạt động 9](#_Toc4268734)

[2.2.2 Tạo một hoạt động 10](#_Toc4268735)

[2.2.3 Bắt đầu một hoạt động 13](#_Toc4268736)

[2.2.4 Quản lý vòng đời của một hoạt động 16](#_Toc4268737)

[CHƯƠNG 3: THIẾT KẾ VÀ TRIỂN KHAI 22](#_Toc4268738)

[3.1 THIẾT KẾ CƠ SỞ DỮ LIỆU 22](#_Toc4268739)

[**3.1.1 Mô hình quan niệm dữ liệu** 22](#_Toc4268740)

[3.1.2 Cơ sở dữ liệu 23](#_Toc4268741)

[3.2. Giao diện 27](#_Toc4268742)

[3.2.1. Giao diện màn hình chính 27](#_Toc4268743)

[3.2.2 Giao diện giới thiệu 28](#_Toc4268744)

[3.2.3 Giao diện trải nghiệm 29](#_Toc4268745)

[3.2.4 Giao diện bắt đầu 30](#_Toc4268746)

[3.2.5 Giao diện kiểm tra 31](#_Toc4268747)

[3.2.6 Giao diện khi lưu thông tin người chơi 32](#_Toc4268748)

[3.2.7 Giao diện kết quả 33](#_Toc4268749)

[3.2.8 Giao diện bảng vàng 34](#_Toc4268750)

[CHƯƠNG 4: KẾT LUẬN KIẾN NGHỊ 35](#_Toc4268751)

[4.1 KẾT QUẢ THỰC NGHIỆM 35](#_Toc4268752)

[4.2 ĐÁNH GIÁ 35](#_Toc4268753)

[4.3 HƯỚNG PHÁT TRIỂN 35](#_Toc4268756)

[TÀI LIỆU THAM KHẢO 37](#_Toc4268757)

# CHƯƠNG 1: TỔNG QUAN

## GIỚI THIỆU PHẦN MỀM TRẮC NGHIỆM LỊCH SỬ

Hiện nay, nước ta đang ngày càng phát triển mạnh mẽ cả về kinh tế lẫn văn hóa xã hội. Trình độ dân trí ngày càng được chú trọng nâng cao. Hội nhập thế giới cũng trở thành nhu cầu tất yếu. Nhu cầu trang bị tốt một kiến thức về lịch sử để có thể trao đổi, tiếp thu các tinh hoa từ các nền văn hóa độc đáo, các tiến bộ khoa học kỹ thuật càng tăng cao. Mà thời buổi công nghệ thông tin mà kiến thức lịch sử lại bị thu hẹp. Vì vậy, để cũng cố kiến thức về lịch sử dân tộc ta nên dùng những gì của công nghệ để thu hẹp khoảng cách đó.

Qua tình hình đó, nhận thấy một phần mềm trắc nghiệm cho mỗi người là cần thiết.Vì vậy, đề tài của chúng em là viết nên một phần mềm trắc nghiệm cho mọi người trên các tiêu chí, dễ nhớ, dễ học và dễ sử dụng.

Các tính năng nổi bật:

+ Trải nghiệm.

+ Thiết lập tính năng.

+ Lịch sử người chơi.

+ Hiển thị câu trả lời ( thông tin chi tiết câu trả lời).

+ Bộ đếm thời gian làm trắc nghiệm

Phần mềm trắc nghiệm giúp cho mọi người dễ dàng học tập và kiểm tra kiến thức của mình một cách tốt nhât, và có khả năng đánh giá độ hiểu biết về lịch sử dân tộc của bản thân và đánh giá mức độ khi người dùng vào làm các bài trắc nghiệm của application trên điện thoại, từ những lần làm bài và được đánh giá bằng cách tính điểm và từ đó sẽ làm thúc đẩy tính ham học, và tính chinh phục độ khó của các câu hỏi, để đạt điểm cao.

## ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

* Đối tượng ứng dụng đề tài:
* Mọi người trên toàn quốc có nhu cầu học hỏi và kiểm tra kiến thức lịch sử của mình.
* Application được sử dụng giành cho tất cả mọi người, mọi lứa tuổi.
* Phạm vi nghiên cứu của đề tài:

Vận dụng công nghệ lập trình đi động Android và không thể thiếu cho một ngôn ngữ lập trình như Java.

## MỤC TIÊU ĐỀ TÀI

Mục tiêu của đề tài phần mềm thi trắc nghiệm giúp cho người dùng học hỏi hỏi và ôn lại các kiến thức đã học mình một cách tốt nhât, và có khả năng đánh giá năng lực và độ hiểu biết của bản thân, và đánh giá xếp loại khi người dùng vào làm các bài trắc nghiệm trên app, từ những lần làm và được đánh giá bằng cách tính điểm và từ đó sẽ làm thúc đẩy tính ham học, và tính chinh phục độ khó của các câu hỏi, để đạt điểm cao.

Đối với người dùng: Chủ yếu là đối tượng học sinh, sinh viên giúp cho các đối tượng này dễ dàng ôn tập kiến thức của mình, Vậy nên trang web phải giúp cho người dùng:

* Tìm kiếm và lựa chọn gói câu hỏi phù hợp với mình.
* Theo dõi kết quả làm được của mình.

Đối với nhà quản lý: là người làm chủ mọi hoạt động của hệ thống có thể truy cập vào Data base nhằm để thực hiện các hình thức sau:

* Quản lý các gói câu hỏi, người dùng một cách dễ dàng.
* Thêm, xóa, sửa thông tin câu hỏi, bộ câu hỏi vào cơ sỡ dữ liệu.
* Kiểm tra và xử lý các chức năng của app.

Điều quan trọng trong phần mềm trắc nghiệm phải bảo đảm tuyệt đối độ chính xác của các đáp án và câu hỏi.

Đồng thời phần mềm còn phải có tính dễ nâng cấp, bảo trì, sửa chữa khi cần bổ sung, cập nhật những tính năng mới.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## CÁC KHÁI NIỆM

### **Hệ Điều Hành Mở**

Android được xây dựng trên nhân Linux và được phân phối miễn phí. Không giống như Windows Mobile và Apple iPhone, tuy cả hai đều cung cấp một môi trường phát triển ứng dụng phong phú và đơn giản dễ tiếp cận nhưng luôn có sự ưu tiên cho các ứng dụng mặc định có sẵn của hệ điều hành (native applications). Với Android mọi ứng dụng đều được viết trên cùng một tập API, thế nên không có sự phân biệt giữa các ứng dụng mặc định và các ứng dụng của bên thứ ba. Người dùng hoàn toàn có thể thay thế mọi ứng dụng mặc định bằng các ứng dụng yêu thích của mình, thậm chí ngay cả màn hình thực hiện cuộc gọi và màn hình nhà (home screen).

Các nhà phát triển viết ứng dụng cho Android dựa trên ngôn ngữ Java. Sự ra mắt của Android vào ngày 5 tháng 11 năm 2007 gắn với sự thành lập của liên minh thiết bị cầm tay mã nguồn mở, bao gồm 78 công ty phần cứng, phần mềm và viễn thông nhằm mục đính tạo nên một chuẩn mở cho điện thoại di động trong tương lai.

### **Các ứng dụng có sẵn trên Android**

Một điện thoại Android thông thường sẽ đi kèm với một vài ứng dụng đã được cài đặt sẵn, bao gồm:

* Một trình email tương tích với Gmail.
* Chương trình quản lí tin nhắn SMS.
* Chương trình quản lí thông tin cá nhân, bao gồm cả lịch làm việc, danh bạ, và được đồng bộ với dịch vụ của Google.
* Phiên bản thu gọn của Google Map cho điện thoại, bao gồm Street View, tìm địa chỉ, và chỉ đường đi, tình trạng giao thông…
* Chương trình đa phương tiện ( nghe nhạc, xem ảnh, xem phim…)

Tất cả các ứng dụng có sẵn đều được viết bằng ngôn ngữ Java và sử dụng Android SDK.

Các dữ liệu về thông tin người dùng được các ứng dụng có sẵn sử dụng - như thông tin về danh bạ - vẫn hoàn toàn có thể được sử dụng bởi các ứng dụng của bên thứ ba. Tương tự vậy, ứng dụng của bạn hoàn toàn có thể xử l{ các sự kiện như có cuộc gọi đến, nhận một tin nhắn mới,… thay cho các ứng dụng có sẵn.

### **Truy cập phần cứng**

Android bao gồm thư viện các API giúp đơn giản hóa tối đa việc sử dụng phần cứng của thiết bị. Điều đó đảm bảo rằng bạn không cần phải bận tâm nhiều đến việc ứng dụng của mình có thể chạy như mong đợi trên nhiều thiết bị khác nhau hay không, miễn là thiết bị đó có hỗ trợ Android.

Android SDK bao gồm các API cho phần cứng như: GPS, camera, kết nối mạng, Wi-Fi, Bluetooth, con quay gia tốc, màn hình cảm ứng, quản lý năng lượng,…

### **Dịch vụ chạy nền**

Android hỗ trợ các ứng dụng và dịch vụ được thiết kể để chạy ẩn.

Do kích thước nhỏ của màn hình điện thoại nên tại một thời điểm chỉ có thể thấy được một ứng dụng. Dịch vụ chạy nền giúp tạo ra các thành phần ứng dụng “vô hình” để thực hiện tự động một tác vụ nào đó mà không cần phải có sự tương tác của người dùng. Ví dụ như một dịch vụ chạy nền có chức năng chặn cuộc gọi đến đối với các số điện thoại có trong “black list” chẳng hạn.

### **SQL Lite Database**

Bởi vì tính chất nhỏ gọn và bị hạn chế về phần cứng của điện thoại di động, cho nên đòi hỏi việc lưu trữ và truy xuất dữ liệu phải nhanh chóng và hiệu quả.

Android hỗ trợ hệ quản trị dữ liệu nhỏ gọn SQLite, và cung cấp cho ứng dụng các API để thao tác. Mặc định mỗi ứng dụng đều được chạy trong sandbox - điều này có nghĩa là nội dung của từng database ứng với từng ứng dụng chỉ có thể được truy cập bằng chính ứng dụng đó. Tuy nhiên cũng có các cơ chế để ứng dụng có thể chia sẽ, trao đổi các database với nhau.

### **Tối ưu hóa bộ nhớ và quản lí tiến trình**

Việc quản lí bộ nhớ và tiến trình trong Android có một chút khác biệt. Giống như công nghệ Java và .NET, Android sử dụng một bộ run-time của riêng mình với công nghệ ảo hóa để quản lí bộ nhớ của các ứng dụng đang chạy. Không giống như những nền tảng khác, Android run-time cũng đồng thời quản lí luôn cả thời gian sống của ứng dụng. Android đảm bảo các ứng dụng đều được đáp ứng bằng cách dừng và hủy các tiến trình không cần thiết để giải phóng tài nguyên cho các tiến trình có độ ưu tiên cao hơn.

Trong bối cảnh đó, độ ưu tiên được xác định tùy thuộc vào ứng dụng mà người dùng đang tương tác. Android đảm bảo rằng các tiến trình có thể bị hủy một cách nhanh chóng, nhưng đồng thời cũng có thể khởi động lại nhanh không kém nếu cần. Điều này thực sự quan trọng trong một môi trường mà ở đó bản thân ứng dụng không thể tự kiểm soát được thời gian sống cho chính mình.

**2.1.7 Android Software Development Kit ( SDK)**

Bộ SDK của Android bao gồm mọi thứ cần thiết giúp bạn có thể lập trình, debug, và test ứng dụng Android.

* Android API:Cốt lõi của bộ SDK là thư viện các API. Và Google cũng chỉ sử dụng bộ API này để xây dựng các ứng dụng có sẵn cho And1roid (native application).
* Development Tool:SDK bao gồm rất nhiều công cụ giúp biên dịch, sửa lỗi, và hỗ trợ trong việc lập trình ứng dụng.
* Android Emulator:Trình giả lập một thiết bị chạy Android thực sự với nhiều skin thay thế, cực kì tiện lợi cho việc test ứng dụng Android ngay trên máy tính mà không cần phải thông qua một thiết bị chạy Android thực.

### **Cấu trúc ứng dụng**

Ý tưởng của Android đó là khuyến khích việc tái sử dụng lại các thành phần đã có, cho phép ứng dụng của bạn có thể chia sẻ Activity, Service, Dữ liệu với các ứng dụng khác trong giới hạn truy cập do bạn đặt ra.

Những dịch vụ dưới đây là nền tảng kiến trúc của mọi ứng dụng Android, đây là khuôn khổ cho bạn xây dựng ứng dụng của riêng mình:

**Activity Manager**

Kiểm soát vòng đời của Activity.

**Views**

Xây dựng giao diện người dùng cho các Activity.

**Notification Manager**

Cung cấp một cơ chế thống nhất và an toàn để ứng dụng có thể đưa ra các thông báo cho người dùng.

**Content Providers**

Giúp cho việc trao đổi, chia sẽ dữ liệu giữa các ứng dụng với nhau.

**Resource Manager**

Hỗ trợ quản l{ các loại tài nguyên không là code (non-code resources) như các chuỗi, hình ảnh, hoặc âm thanh,…

### **Các thư viện của Android:**

Android cung cấp các gói API để phát triển ứng dụng. Danh sách các gói core API dưới đây giúp cho bạn có cái nhìn tổng quát về những gì được hỗ trợ sẵn, tất cả các thiết bị chạy Android đều phải hỗ trợ được tối thiểu các API này.

**android.util**

Gói api lõi, chứa các class cấp thấp như containers, string formatters, XML parsing

**android.os**

Truy cập tới các chức năng của hệ điều hành chẳng hạn như: gửi nhận tin nhắn, giao tiếp nội bộ giữa các ứng dụng, thời gian,…

**android.graphics**

Cung cấp các lớp liên quan tới xử l{ đồ họa ở mức thấp. Hỗ trợ các hàm cơ bản như vẽ điểm, vẽ miền, tô màu,… trên một khung canvas.

**android.text**

Cung cấp các hàm phân tích và xử lý chuỗi.

**android.database**

Cung cấp các lớp cấp thấp cần thiết để làm việc với databases.

**android.content**

Dùng để quản lý các tài nguyên, các nội dung, và các gói.

**android.view**

Views là lớp cha của mọi lớp giao diện người dùng.

**android.widget**

Được kế thừa từ lớp View, bao gồm các lớp để cơ bản để xây dựng giao diện widget như: lists, buttons, layouts. *idget là một dạng ứng dụng mini - hiển thị ở màn hìnhnh* **android.maps**

Gói API cấp cao, dùng để truy cập đến các chức năng của GoogleMap.

**android.app**

Gói API cấp cao, bao gồm lớp Activity và Service - hai lớp cơ sở cho mọi ứng dụng Android.

**android.telephony**

Cung cấp cho bạn khả năng tương tác trực tiếp với các chức năng cơ bản của một chiếc điện thoại như nghe, gọi, tin nhắn.

**android.webkit**

Cung cấp một WebView control trên nền webkit để có thể nhúng vào ứng dụng, cùng với các api điều khiển cơ bản như stop,refresh,cookie,manager.

Hầu hết các ứng dụng trong Android sẽ thuộc vào một trong ba loại sau đây:

* **Foreground Activity**

Ứng dụng hữu dụng chỉ khi nó được tương tác với người dùng ở mặt tiền (foreground), ứng dụng mất tác dụng khi bị ẩn đi. Game, bản đồ, từ điển là các ví dụ điển hình.

* **Background Service**

Ứng dụng không đòi hỏi nhiều đến tương tác của người dùng. Trừ khi được cấu hình, còn phần lớn thời gian là ứng dụng chạy ngầm. Ví dụ thuộc loại này chẳng hạn như: ứng dụng trả lời tin nhắn tự động.

* **Intermittent Activity**

Ứng dụng loại này vừa nhận tương tác của người dùng, và vẫn có thể tiếp tục hoạt động ngầm khi bị ẩn đi. Có thể đưa ra các thông báo cho người dùng khi cần thiết trong lúc đang bị ẩn. Các ứng dụng loại này có thể kể đến như: ứng dụng nghe nhạc, ứng dụng tán gẫu (chat),…

Đối với các ứng dụng lớn và phức tạp, khó để có thể cho vào một loại duy nhất, và thường thì các ứng dụng này bao gồm có khi cả 3 loại trên. Điều quan trọng là khi bạn xây dựng ứng dụng cho mình hãy chú { đến 3 loại trên để đưa ra thiết kế phù hợp. Dưới đây là một số lưu { khi xây dựng ứng dụng thuộc 3 loại trên:

* **Foreground Activities**

Khi xây dựng ứng dụng loại này cần phải xem xét cẩn thận về chu kz sống của Activity (Activity life cycle) để có thể tạo sự chuyển đổi thông suốt và mượt mà giữa trạng thái foreground và background.

Vì ứng dụng không tự nó kiểm soát được chu kz sống, nên điều này đồng nghĩa với việc bạn cần phải lưu lại trạng thái của ứng dụng trước khi nó chuyển từ foreground sang background, và trả lại trạng thái phù hợp trong trường hợp ngược lại.

* **Background Services**

Với ứng dụng loại này, cần chú { là nó sẽ cần rất ít các tương tác trực của người dùng, có khi hoàn toàn không cần. Thay vào đó ứng dụng sẽ lắng nghe các thông điệp, sự kiện phát sinh bởi hệ thống, phần cứng, hoặc từ các ứng dụng khác.

Bạn có thể tạo ra một ứng dụng hoàn toàn “vô hình”, nhưng tốt hơn là nên cung cấp một giao diện cơ bản để người dùng có thể hiệu chỉnh các thiết lập, dừng hoặc thoát hoàn toàn ứng dụng.

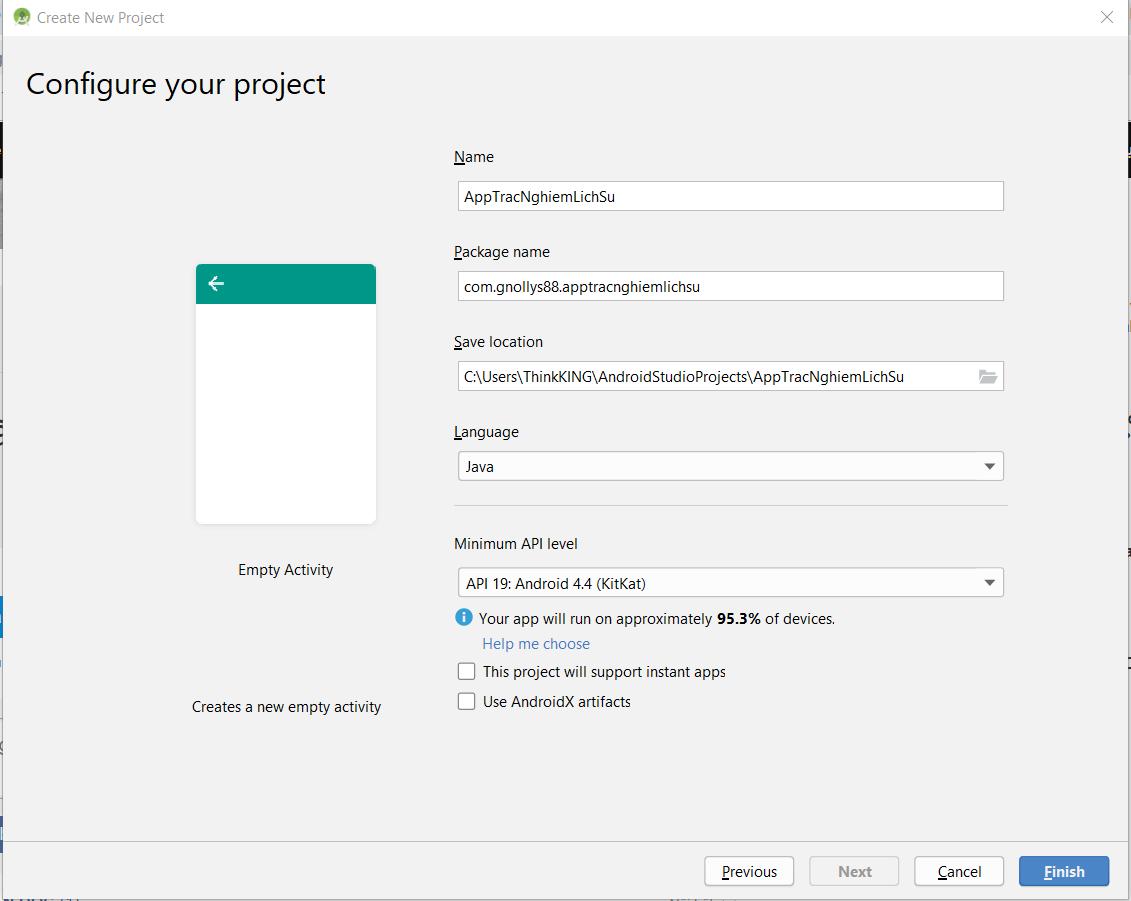
* **Intermittent Activities**

Ứng dụng loại này là sự kết hợp giữa một Foreground Activitie và một Background Service, vậy nên điều cơ bản là ứng dụng cần phải nhận biết được tình trạng hiện tại của mình (foreground/background) để đưa ra các xử l{ phù hợp.

Cũng nên lưu { đến việc gởi các thông báo (có thể thông qua hệ thống notification của Android) đến người dùng khi ứng dụng đang chạy ngầm vào những lúc cần thiết.

## CƠ CHẾ HOẠT ĐỘNG

### 2.2.1 Các hoạt động



[**Activity**](https://developer.android.com/reference/android/app/Activity.html?hl=vi) là một thành phần ứng dụng cung cấp một màn hình mà với nó người dùng có thể tương tác để thực hiện một điều gì đó, chẳng hạn như quay số điện thoại, chụp ảnh, gửi e-mail hoặc xem bản đồ. Mỗi hoạt động được cho trong một cửa sổ là nơi để vẽ giao diện người dùng của nó. Cửa sổ này thường lấp đầy màn hình, nhưng có thể nhỏ hơn màn hình và nổi bên trên các cửa sổ khác.

Ứng dụng thường bao gồm nhiều hoạt động được liên kết lỏng lẻo với nhau. Thường thì một hoạt động trong một ứng dụng sẽ được quy định là hoạt động "chính", nó được trình bày trước người dùng khi khởi chạy ứng dụng lần đầu. Sau đó, mỗi hoạt động có thể bắt đầu một hoạt động khác để thực hiện các hành động khác nhau. Mỗi khi một hoạt động mới bắt đầu, hoạt động trước đó sẽ bị dừng lại, nhưng hệ thống vẫn giữ nguyên hoạt động trong một ngăn xếp ("back stack"). Khi một hoạt động mới bắt đầu, nó được đẩy lên ngăn xếp và chiếm lấy tiêu điểm của người dùng. Ngăn xếp sẽ tuân theo cơ chế xếp chồng cơ bản "vào cuối, ra đầu", vì thế, khi người dùng kết thúc hoạt động hiện tại và nhấn nút Quay lại, nó sẽ được đẩy ra khỏi ngăn xếp (và bị hủy) và hoạt động trước đó sẽ tiếp tục.

### 2.2.2 Tạo một hoạt động

Để tạo một hoạt động, bạn phải tạo một lớp con của [**Activity**](https://developer.android.com/reference/android/app/Activity.html?hl=vi) (hoặc một lớp con hiện tại của nó). Trong lớp con của mình, bạn cần triển khai các phương pháp gọi lại mà hệ thống gọi khi hoạt động chuyển tiếp giữa các trạng thái khác nhau trong vòng đời, chẳng hạn như khi hoạt động đang được tạo, dừng, tiếp tục, hoặc hủy. Hai phương pháp gọi lại quan trọng nhất là:

[**onCreate()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onCreate(android.os.Bundle))

Bạn phải triển khai phương pháp này. Hệ thống gọi phương pháp này khi tạo hoạt động của bạn. Trong quá trình thực hiện của mình, bạn nên khởi chạy những thành phần thiết yếu cho hoạt động của mình. Quan trọng nhất, đây là lúc bạn phải gọi [**setContentView()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#setContentView(android.view.View)) để định nghĩa bố trí cho giao diện người dùng của hoạt động.

[**onPause()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onPause())

Hệ thống gọi phương pháp này là dấu hiệu đầu tiên về việc người dùng đang rời khỏi hoạt động của bạn (mặc dù không phải lúc nào cũng có nghĩa rằng hoạt động đang bị hủy). Trường hợp này thường là khi bạn định thực hiện bất kỳ thay đổi nào vẫn cần có hiệu lực ngoài phiên của người dùng hiện thời (vì người dùng có thể không quay lại).

#### Triển khai một giao diện người dùng

Giao diện người dùng cho một hoạt động sẽ được cung cấp theo phân cấp dạng xem đối tượng được suy ra từ lớp View. Mỗi chế độ xem kiểm soát một không gian chữ nhật riêng trong cửa sổ của hoạt động và có thể phản hồi trước tương tác của người dùng. Ví dụ, chế độ xem có thể là một nút khởi xướng một hành động khi người dùng chạm vào nó.

Android cung cấp nhiều chế độ xem sẵn có mà bạn có thể sử dụng để thiết kế và tổ chức cho bố trí của mình. "Widget" là những chế độ xem cung cấp những phần tử trực quan (và tương tác) cho màn hình, chẳng hạn như nút, trường văn bản, hộp kiểm, hay chỉ là một hình ảnh. "Bố trí" là những chế độ xem được suy ra từ ViewGroup cung cấp một mô hình bố trí duy nhất cho các chế độ xem con của nó, chẳng hạn như bố trí tuyến tính, bố trí lưới, hoặc bố trí tương đối. Bạn cũng có thể chia thành lớp con View và các lớp ViewGroup (hoặc các lớp con hiện tại) để tạo widget và bố trí của chính mình và áp dụng chúng vào bố trí hoạt động của bạn.

Cách phổ biến nhất để định nghĩa một bố trí bằng cách sử dụng các chế độ xem là dùng một tệp bố trí XML được lưu trong tài nguyên ứng dụng của bạn. Bằng cách này, bạn có thể duy trì thiết kế giao diện người dùng của mình độc lập với mã nguồn định nghĩa hành vi của hoạt động. Bạn có thể đặt bố trí làm UI cho hoạt động của mình bằng setContentView(), chuyển ID tài nguyên cho bố trí. Tuy nhiên, bạn cũng có thể tạo View mới trong mã hoạt động của mình và xây dựng một cấp bậc chế độ xem bằng cách chèn các View mới vào một ViewGroup, sau đó sử dụng bố trí đó bằng cách chuyển root ViewGroup sang setContentView().

#### Khai báo hoạt động trong bản kê khai

Bạn phải khai báo hoạt động của mình trong tệp bản kê khai để hoạt động có thể truy cập được vào hệ thống. Để khai báo hoạt động của mình, hãy mở tệp bản kê khai của bạn và thêm một phần tử [**&lt;activity&gt**;](https://developer.android.com/guide/topics/manifest/activity-element.html?hl=vi) làm con của phần tử [**&lt;application&gt;**](https://developer.android.com/guide/topics/manifest/application-element.html?hl=vi) . Ví dụ:

<manifest ... >  
  <application ... >  
      <activity android:name=".ExampleActivity" />  
      ...  
  </application ... >  
  ...  
</manifest >

Có vài thuộc tính khác mà bạn có thể nêu trong phần tử này, để định nghĩa các thuộc tính như nhãn cho hoạt động, biểu tượng cho hoạt động, hoặc chủ đề mô tả kiểu UI của hoạt động. Thuộc tính [android:name](https://developer.android.com/guide/topics/manifest/activity-element.html?hl=vi#nm) là thuộc tính bắt buộc duy nhất—nó quy định tên lớp của hoạt động. Một khi bạn phát hành ứng dụng của mình, bạn không nên thay đổi tên này, vì nếu bạn làm vậy, bạn có thể làm hỏng một số tính năng, chẳng hạn như các lối tắt của ứng dụng

#### Sử dụng các bộ lọc ý định

Một phần tử [&lt;activity&gt;](https://developer.android.com/guide/topics/manifest/activity-element.html?hl=vi) cũng có thể quy định các bộ lọc ý định khác nhau—bằng cách sử dụng phần tử [&lt;intent-filter&gt;](https://developer.android.com/guide/topics/manifest/intent-filter-element.html?hl=vi) —để khai báo cách thức mà các thành phần khác của ứng dụng có thể kích hoạt nó.

Khi bạn tạo một ứng dụng mới bằng cách sử dụng các công cụ SDK của Android, hoạt động chương trình nhỏ được tạo cho bạn sẽ tự động bao gồm một bộ lọc ý định khai báo hoạt động phản hồi lại hành động "chính" và nên được đặt trong thể loại "trình khởi chạy". Bộ lọc ý định trông như thế này:

<activity android:name=".ExampleActivity" android:icon="@drawable/app\_icon">  
    <intent-filter>  
        <action android:name="android.intent.action.MAIN" />  
        <category android:name="android.intent.category.LAUNCHER" />  
    </intent-filter>  
</activity>

Phần tử [**&lt;action&gt;**](https://developer.android.com/guide/topics/manifest/action-element.html?hl=vi) quy định rằng đây là điểm mục nhập "chính" đối với ứng dụng. Phần tử [**&lt;category&gt;**](https://developer.android.com/guide/topics/manifest/category-element.html?hl=vi)quy định rằng hoạt động này nên được liệt kê trong trình khởi chạy ứng dụng của hệ thống (để cho phép người dùng khởi chạy hoạt động này).

Nếu bạn có ý định cho ứng dụng của mình được độc lập và không cho phép các ứng dụng khác kích hoạt các hoạt động của nó, vậy bạn không cần bất kỳ bộ lọc ý định nào khác. Chỉ một hoạt động nên có hành động "chính" và thể loại "trình khởi chạy" như trong ví dụ trước. Những hoạt động mà bạn không muốn cung cấp sẵn cho các ứng dụng khác không nên có bộ lọc ý định và bạn có thể tự mình bắt đầu chúng bằng cách sử dụng các ý định rõ ràng (như được đề cập trong phần sau).

Tuy nhiên, nếu bạn muốn hoạt động của mình phản hồi lại những ý định ngầm mà được chuyển giao từ các ứng dụng khác (và chính bạn), thì bạn phải định nghĩa các bộ lọc ý định bổ sung cho hoạt động của mình. Với mỗi loại ý định mà bạn muốn phản hồi, bạn phải nêu một [**&lt;intent-filter&gt;**](https://developer.android.com/guide/topics/manifest/intent-filter-element.html?hl=vi) bao gồm một phần tử [**&lt;action&gt;**](https://developer.android.com/guide/topics/manifest/action-element.html?hl=vi) và, không bắt buộc, một phần tử [**&lt;category&gt;**](https://developer.android.com/guide/topics/manifest/category-element.html?hl=vi) và/hoặc một phần tử [**&lt;data&gt;**](https://developer.android.com/guide/topics/manifest/data-element.html?hl=vi). Những phần tử này quy định loại ý định mà hoạt động của bạn có thể phản hồi.

### 2.2.3 Bắt đầu một hoạt động

Bạn có thể bắt đầu một hoạt động khác bằng cách gọi [**startActivity()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#startActivity(android.content.Intent)), chuyển cho nó một **Intent** mà mô tả hoạt động bạn muốn bắt đầu. Ý định này sẽ quy định hoặc hoạt động chính xác mà bạn muốn bắt đầu hoặc mô tả loại hành động mà bạn muốn thực hiện (và hệ thống lựa chọn hoạt động phù hợp cho bạn, thậm chí có thể từ một ứng dụng khác). Một ý định cũng có thể mang theo lượng nhỏ dữ liệu sẽ được sử dụng bởi hoạt động được bắt đầu.

Khi đang làm việc trong ứng dụng của chính mình, bạn thường sẽ cần khởi chạy một hoạt động đã biết. Bạn có thể làm vậy bằng cách tạo một ý định trong đó quy định rõ hoạt động bạn muốn bắt đầu, sử dụng tên lớp đó. Ví dụ, sau đây là cách một hoạt động bắt đầu một hoạt động khác có tên **SignInActivity**:

Intent intent = new Intent(this, SignInActivity.class);  
startActivity(intent);

Tuy nhiên, ứng dụng của bạn cũng có thể muốn thực hiện một số hành động, chẳng hạn như gửi một e-mail, tin nhắn văn bản, hoặc cập nhật trạng thái, bằng cách sử dụng dữ liệu từ hoạt động của bạn. Trong trường hợp này, ứng dụng của bạn có thể không có các hoạt động của chính nó để thực hiện những hành động đó, vì vậy, thay vào đó, bạn có thể tận dụng những hoạt động được cung cấp bởi các ứng dụng khác trên thiết bị mà có thể thực hiện hành động cho bạn. Đây là lúc ý định thực sự có giá trị—bạn có thể tạo một ý định mô tả một hành động bạn muốn thực hiện và hệ thống sẽ khởi chạy hoạt động phù hợp đó từ một ứng dụng khác. Nếu có nhiều hoạt động mà có thể xử lý ý định, vậy người dùng có thể chọn hoạt động nào sẽ sử dụng. Ví dụ, nếu bạn muốn cho phép người dùng gửi e-mail, bạn có thể tạo ý định sau:

Intent intent = new Intent(Intent.ACTION\_SEND);  
intent.putExtra(Intent.EXTRA\_EMAIL, recipientArray);  
startActivity(intent);

[**EXTRA\_EMAIL**](https://developer.android.com/reference/android/content/Intent.html?hl=vi#EXTRA_EMAIL) phụ được thêm vào ý định là một mảng xâu của các địa chỉ e-mail mà e-mail sẽ được gửi tới. Khi một ứng dụng e-mail phản hồi ý định này, nó đọc mảng xâu được cung cấp trong phần phụ và đặt nó vào trường "đến" của mẫu soạn thảo e-mail. Trong trường hợp này, hoạt động của ứng dụng e-mail bắt đầu và khi người dùng làm xong, hoạt động của bạn sẽ tiếp tục.

#### Bắt đầu một hoạt động cho một kết quả

Đôi khi bạn có thể muốn nhận được một kết quả từ hoạt động mà bạn bắt đầu. Trong trường hợp đó, hãy bắt đầu hoạt động bằng cách gọi [**startActivityForResult()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#startActivityForResult(android.content.Intent,%20int))(thay vì [**startActivity()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#startActivity(android.content.Intent))). Rồi để nhận được kết quả từ hoạt động sau đó, hãy triển khai phương pháp gọi lại [**onActivityResult()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onActivityResult(int,%20int,%20android.content.Intent)) . Khi hoạt động sau đó diễn ra xong, nó trả về một kết quả trong một [**Intent**](https://developer.android.com/reference/android/content/Intent.html?hl=vi) cho phương pháp [**onActivityResult()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onActivityResult(int,%20int,%20android.content.Intent)) của bạn.

Ví dụ, bạn có thể muốn người dùng chọn một trong các liên lạc của họ, vì vậy hoạt động của bạn có thể làm gì đó với thông tin trong liên lạc đó. Đây là cách bạn có thể tạo một ý định như vậy và xử lý kết quả:

private void pickContact() {  
    // Create an intent to "pick" a contact, as defined by the content provider URI  
    Intent intent = new Intent(Intent.ACTION\_PICK, Contacts.CONTENT\_URI);  
    startActivityForResult(intent, PICK\_CONTACT\_REQUEST);  
}  
  
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    // If the request went well (OK) and the request was PICK\_CONTACT\_REQUEST  
    if (resultCode == Activity.RESULT\_OK && requestCode == PICK\_CONTACT\_REQUEST) {  
        // Perform a query to the contact's content provider for the contact's name  
        Cursor cursor = getContentResolver().query(data.getData(),  
        new String[] {Contacts.DISPLAY\_NAME}, null, null, null);  
        if (cursor.moveToFirst()) { // True if the cursor is not empty  
            int columnIndex = cursor.getColumnIndex(Contacts.DISPLAY\_NAME);  
            String name = cursor.getString(columnIndex);  
            // Do something with the selected contact's name...  
        }  
    }  
}

Ví dụ này thể hiện lô-gic cơ bản mà bạn sẽ sử dụng trong phương pháp [**onActivityResult()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onActivityResult(int,%20int,%20android.content.Intent)) của mình để xử lý một kết quả hoạt động. Điều kiện đầu tiên kiểm tra xem yêu cầu có thành công không—nếu có thì **resultCode** sẽ là [**RESULT\_OK**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#RESULT_OK)—và liệu yêu cầu mà kiểm tra này đang phản hồi có được biết hay không—trong trường hợp này, **requestCode** phù hợp với tham số thứ hai được gửi bằng [**startActivityForResult()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#startActivityForResult(android.content.Intent,%20int)). Từ đó, mã xử lý kết quả hoạt động bằng cách truy vấn dữ liệu được trả về trong [**Intent**](https://developer.android.com/reference/android/content/Intent.html?hl=vi) (tham số data).

### 2.2.4 Quản lý vòng đời của một hoạt động

Việc quản lý vòng đời các hoạt động của bạn bằng cách triển khai các phương pháp gọi lại rất quan trọng đối với việc xây dựng một ứng dụng mạnh và linh hoạt. Vòng đời của một hoạt động trực tiếp bị ảnh hưởng bởi sự liên kết giữa nó với các hoạt động khác, tác vụ của nó và ngăn xếp (back stack).

Về cơ bản, một hoạt động có thể tồn tại ở ba trạng thái:

***Tiếp tục***

Hoạt động ở tiền cảnh của màn hình và có tiêu điểm của người dùng. (Trạng thái này đôi khi cũng được gọi là "đang chạy".)

***Tạm dừng***

Một hoạt động khác ở tiền cảnh và có tiêu điểm, nhưng hoạt động này vẫn hiển thị. Cụ thể, một hoạt động khác hiển thị ở trên hoạt động này và hoạt động đó trong suốt một phần hoặc không che toàn bộ màn hình. Trạng thái tạm dừng hoàn toàn đang hoạt động (đối tượng [**Activity**](https://developer.android.com/reference/android/app/Activity.html?hl=vi) được giữ lại trong bộ nhớ, nó duy trì tất cả thông tin về trạng thái và thành viên, và vẫn gắn với trình quản lý cửa sổ), nhưng có thể bị hệ thống tắt bỏ trong trường hợp bộ nhớ cực kỳ thấp.

***Dừng***

Hoạt động bị che khuất hoàn toàn bởi một hoạt động khác (hoạt động hiện đang “dưới nền"). Hoạt động dừng cũng vẫn đang hoạt động ([**Activity**](https://developer.android.com/reference/android/app/Activity.html?hl=vi) đối tượng được giữ lại trong bộ nhớ, nó duy trì tất cả thông tin về trạng thái và thành viên, nhưng không gắn với trình quản lý cửa sổ). Tuy nhiên, hoạt động không còn hiển thị với người dùng nữa và hệ thống có thể tắt bỏ hoạt động này khi cần bộ nhớ ở nơi khác.

Nếu một hoạt động bị tạm dừng hoặc dừng, hệ thống có thể bỏ nó khỏi bộ nhớ hoặc bằng cách yêu cầu nó kết thúc (gọi phương pháp [**finish()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#finish()) của nó).

#### Triển khai gọi lại một vòng đời

Cùng nhau, những phương pháp này định nghĩa toàn bộ vòng đời của một hoạt động. Bằng việc triển khai những phương pháp này, bạn có thể theo dõi ba vòng lặp lồng nhau trong vòng đời của hoạt động:

* **Toàn bộ vòng đời** của một hoạt động sẽ xảy ra từ thời điểm lệnh gọi đến [**onCreate()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onCreate(android.os.Bundle)) cho tới thời điểm lệnh gọi đến [**onDestroy()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onDestroy())**.** Hoạt động của bạn nên thực hiện thiết lập trạng thái "chung" (chẳng hạn như định nghĩa bố trí) trong [**onCreate()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onCreate(android.os.Bundle)), và giải phóng tất cả tài nguyên còn lại trong [**onDestroy()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onDestroy()). Ví dụ, nếu hoạt động của bạn có một luồng đang chạy ngầm để tải xuống dữ liệu từ mạng, nó có thể tạo luồng đó trong [**onCreate()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onCreate(android.os.Bundle))rồi dừng luồng trong [**onDestroy()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onDestroy())**.**
* **Vòng đời hiển thị** của một hoạt động xảy ra từ thời điểm lệnh gọi đến [**onStart()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onStart()) cho tới lệnh gọi đến [**onStop()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onStop()). Trong thời gian này, người dùng có thể thấy hoạt động trên màn hình và tương tác với nó. Ví dụ, [**onStop()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onStop()) được gọi khi một hoạt động mới bắt đầu và không còn hiển thị nữa. Giữa hai phương pháp này, bạn có thể duy trì các tài nguyên cần để cho người dùng thấy hoạt động. Ví dụ, bạn có thể đăng ký một [**BroadcastReceiver**](https://developer.android.com/reference/android/content/BroadcastReceiver.html?hl=vi) trong [**onStart()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onStart()) để theo dõi các thay đổi tác động tới UI của mình, và bỏ đăng ký nó trong [**onStop()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onStop()) khi người dùng không còn thấy thứ bạn đang hiển thị nữa. Hệ thống có thể gọi [**onStart()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onStart()) và [**onStop()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onStop()) nhiều lần trong suốt vòng đời của hoạt động, khi đó hoạt động luân chuyển giữa trạng thái hiển thị và ẩn với người dùng.
* **Vòng đời ở tiền cảnh** của một hoạt động xảy ra từ thời điểm lệnh gọi đến [**onResume()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onResume())cho tới thời điểm lệnh gọi đến [**onPause()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onPause()). Trong thời gian này, hoạt động sẽ ở phía trước tất cả hoạt động khác trên màn hình và có tiêu điểm đầu vào của người dùng. Hoạt động có thể thường xuyên chuyển tiếp vào và ra tiền cảnh—ví dụ**,**[**onPause()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onPause())được gọi khi thiết bị vào trạng thái ngủ hoặc khi một hộp thoại xuất hiện. Vì trạng thái này có thể chuyển tiếp thường xuyên, mã trong hai phương pháp này nên tương đối nhẹ để tránh chuyển tiếp chậm khiến người dùng phải đợi.

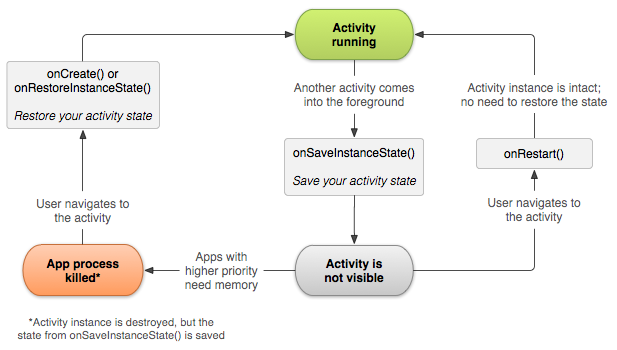


#### Lưu trạng thái của một hoạt động

Phần giới thiệu về [Quản lý Vòng đời của Hoạt động](https://developer.android.com/guide/components/activities?hl=vi#Lifecycle) có đề cập sơ qua rằng khi một hoạt động bị tạm dừng hoặc dừng, trạng thái của hoạt động đó sẽ được giữ lại. Điều này đúng vì đối tượng Activity() vẫn được giữ trong bộ nhớ khi nó bị tạm dừng hoặc dừng—tất cả thông tin về các thành viên và trạng thái hiện tại của nó vẫn hoạt động. Vì thế, bất kỳ thay đổi nào mà người dùng đã thực hiện trong hoạt động đều được giữ lại sao cho khi hoạt động trở về tiền cảnh (khi nó "tiếp tục"), thì những thay đổi này vẫn còn đó.

Tuy nhiên, khi hệ thống hủy một hoạt động để khôi phục bộ nhớ, đối tượng [**Activity**](https://developer.android.com/reference/android/app/Activity.html?hl=vi) bị hủy, vì vậy hệ thống không thể đơn thuần tiếp tục hoạt động với trạng thái không bị ảnh hưởng. Thay vào đó, hệ thống phải tạo lại đối tượng [**Activity**](https://developer.android.com/reference/android/app/Activity.html?hl=vi) nếu người dùng điều hướng trở lại nó. Tuy vậy, người dùng không biết rằng hệ thống đã hủy hoạt động và tạo lại nó và, vì thế, có thể cho rằng hoạt động sẽ vẫn nguyên như cũ. Trong tình huống này, bạn có thể đảm bảo rằng thông tin quan trọng về trạng thái của hoạt động được giữ nguyên bằng cách triển khai một phương pháp gọi lại bổ sung cho phép bạn lưu thông tin về trạng thái của hoạt động của mình: [**onSaveInstanceState()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onSaveInstanceState(android.os.Bundle))**.**

Hệ thống gọi [**onSaveInstanceState()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onSaveInstanceState(android.os.Bundle)) trước khi khiến hoạt động dễ bị hủy. Hệ thống chuyển cho phương pháp này một [**Bundle**](https://developer.android.com/reference/android/os/Bundle.html?hl=vi) trong đó bạn có thể lưu thông tin trạng thái về hoạt động như cặp tên giá trị, bằng cách sử dụng các phương pháp như [**putString()**](https://developer.android.com/reference/android/os/BaseBundle.html?hl=vi#putString(java.lang.String,%20java.lang.String)) và [**putInt()**](https://developer.android.com/reference/android/os/BaseBundle.html?hl=vi#putInt(java.lang.String,%20int)). Sau đó, nếu hệ thống tắt bỏ tiến trình ứng dụng của bạn và người dùng điều hướng trở lại hoạt động của bạn, hệ thống sẽ tạo lại hoạt động đó và chuyển [**Bundle**](https://developer.android.com/reference/android/os/Bundle.html?hl=vi) cho cả [**onCreate()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onCreate(android.os.Bundle)) và [**onRestoreInstanceState()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onRestoreInstanceState(android.os.Bundle))**.** Sử dụng một trong hai phương pháp này, bạn có thể trích xuất trạng thái đã lưu của mình từ [**Bundle**](https://developer.android.com/reference/android/os/Bundle.html?hl=vi) và khôi phục trạng thái của hoạt động. Nếu không có thông tin trạng thái để khôi phục, khi đó [**Bundle**](https://developer.android.com/reference/android/os/Bundle.html?hl=vi) được chuyển cho bạn sẽ rỗng (là trường hợp khi hoạt động được tạo lần đầu).



#### Xử lý thay đổi về cấu hình

Một số cấu hình thiết bị có thể thay đổi trong thời gian chạy (chẳng hạn như hướng màn hình, sự sẵn có của bàn phím, và ngôn ngữ). Khi sự thay đổi đó diễn ra, Android tạo lại hoạt động đang chạy (hệ thống gọi [**onDestroy()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onDestroy()), rồi ngay lập tức gọi [**onCreate()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onCreate(android.os.Bundle))). Hành vi này được thiết kế để giúp ứng dụng của bạn điều chỉnh theo những cấu hình mới bằng cách tự động tải lại ứng dụng của bạn bằng các tài nguyên thay thế mà bạn đã cung cấp (chẳng hạn như bố trí khác cho các hướng và kích cỡ màn hình khác).

Nếu bạn thiết kế hoạt động của mình một cách phù hợp để xử lý khởi động lại do thay đổi hướng màn hình và khôi phục trạng thái hoạt động như nêu trên, ứng dụng của bạn sẽ linh hoạt hơn trước những sự kiện bất ngờ khác trong vòng đời của hoạt động.

Cách tốt nhất để xử lý khởi động lại đó là lưu và khôi phục trạng thái hoạt động của bạn bằng cách sử dụng [**onSaveInstanceState()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onSaveInstanceState(android.os.Bundle))và [**onRestoreInstanceState()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onRestoreInstanceState(android.os.Bundle)) (hoặc [**onCreate()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onCreate(android.os.Bundle))), như đã đề cập trong phần trước.

#### Điều phối hoạt động

Khi một hoạt động bắt đầu một hoạt động khác, cả hai đều trải qua những chuyển tiếp vòng đời. Hoạt động thứ nhất tạm dừng và dừng (tuy nhiên, nó sẽ không dừng nếu vẫn hiển thị được dưới nền), trong khi hoạt động kia được tạo. Trong trường hợp những hoạt động này chia sẻ dữ liệu được lưu vào đĩa hoặc nơi khác, điều quan trọng là phải hiểu rằng hoạt động thứ nhất không bị dừng hoàn toàn trước khi hoạt động thứ hai được tạo. Thay vào đó, tiến trình bắt đầu hoạt động thứ hai chồng lấp với tiến trình dừng hoạt động thứ nhất.

Thứ tự gọi lại vòng đời được định nghĩa rõ, cụ thể là khi hai hoạt động trong cùng tiến trình và hoạt động này bắt đầu hoạt động kia. Sau đây là thứ tự thao tác diễn ra khi Hoạt động A bắt đầu Hoạt động B:

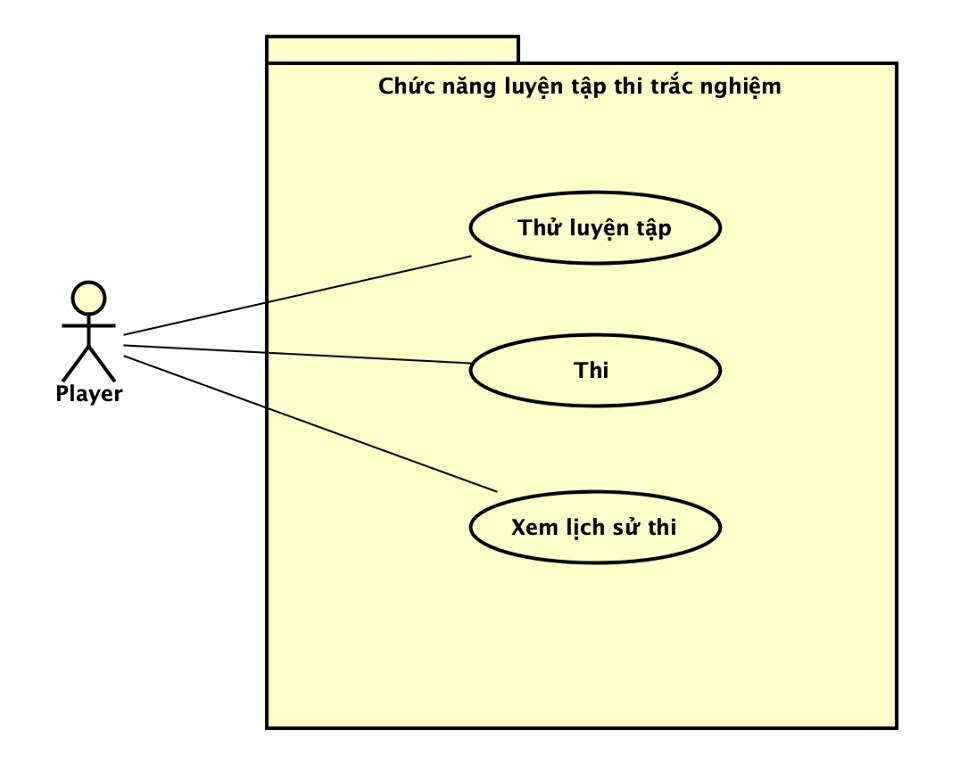
1. Phương pháp [**onPause()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onPause()) của Hoạt động A thực thi.
2. [**onCreate()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onCreate(android.os.Bundle)) của Hoạt động B, [**onStart()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onStart()), và các phương pháp [**onResume()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onResume()) thực thi theo trình tự. (Hoạt động B lúc này có tiêu điểm của người dùng.)
3. Sau đó, nếu Hoạt động A không còn hiển thị trên màn hình, phương pháp [**onStop()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onStop()) của nó sẽ thực thi.

Trình tự gọi lại vòng đời có thể dự đoán này cho phép bạn quản lý chuyển tiếp thông tin từ hoạt động này sang hoạt động khác. Ví dụ, nếu bạn phải ghi vào một cơ sở dữ liệu khi hoạt động thứ nhất dừng sao cho hoạt động theo sau có thể đọc nó, khi đó bạn nên ghi vào cơ sở dữ liệu trong khi [**onPause()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onPause()) thay vì trong khi [**onStop()**](https://developer.android.com/reference/android/app/Activity.html?hl=vi#onStop()).

# CHƯƠNG 3: THIẾT KẾ VÀ TRIỂN KHAI

## THIẾT KẾ CƠ SỞ DỮ LIỆU

### C:\Users\Nghi\Downloads\54523375_320405315284160_5559208550418350080_n.png**3.1.1 Mô hình quan niệm dữ liệu**



### 3.1.2 Cơ sở dữ liệu

#### **SQLite database**

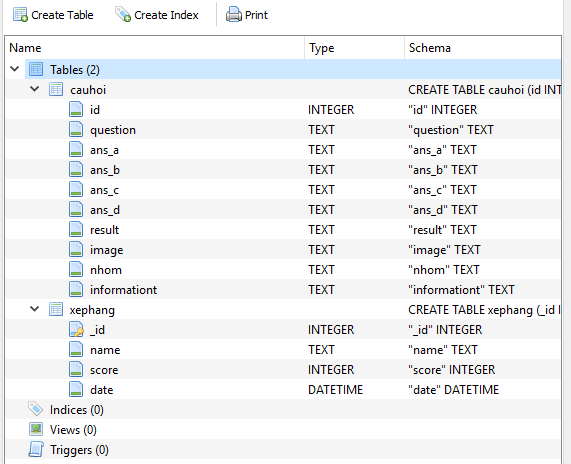
* SQLite là một cơ sở dữ liệu SQL mã nguồn mở, nó lưu trữ dữ liệu vào một tập tin văn bản trên một thiết bị. Nó mặc định đã được tích hợp trên thiết bị Android. Để truy cập dữ liệu này, bạn không cần phải thiết lập bất kỳ loại kết nối nào cho nó như JDBC, ODBC, ... SQLite được Richard Hipp viết dưới dạng thư viện bằng ngôn ngữ lập trình C.
* SQLite có các ưu điểm sau:
  + Tin cậy: các hoạt động transaction (chuyển giao) nội trong cơ sở dữ liệu được thực hiện trọn vẹn, không gây lỗi khi xảy ra sự cố phần cứng
  + Tuân theo chuẩn SQL92 (chỉ có một vài đặc điểm không hỗ trợ)
  + Không cần cài đặt cấu hình
  + Kích thước chương trình gọn nhẹ, với cấu hình đầy đủ chỉ không đầy 300 kB
  + Thực hiện các thao tác đơn giản nhanh hơn các hệ thống cơ sở dữ liệu khách/chủ khác
  + Không cần phần mềm phụ trợ
  + Phần mềm tự do với mã nguồn mở, được chú thích rõ ràng

#### **Sử dụng SQLite trong Android**

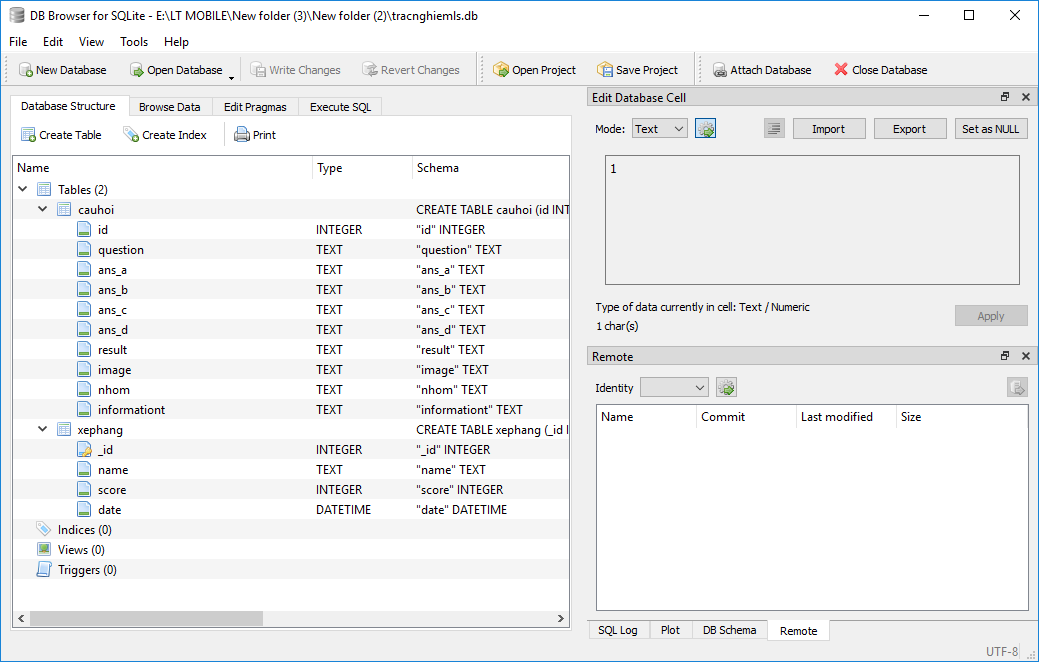
Trong phần này, tôi sẽ hướng dẫn các bạn về cách tạo ra một database trong SQLite và các class để xử lý tất cả các thao tác đối với SQLite.

Sau đây, tôi sẽ lấy một ví dụ về việc lưu trữ thông tin của các sinh viên vào trong cơ sở dữ liệu SQLite.

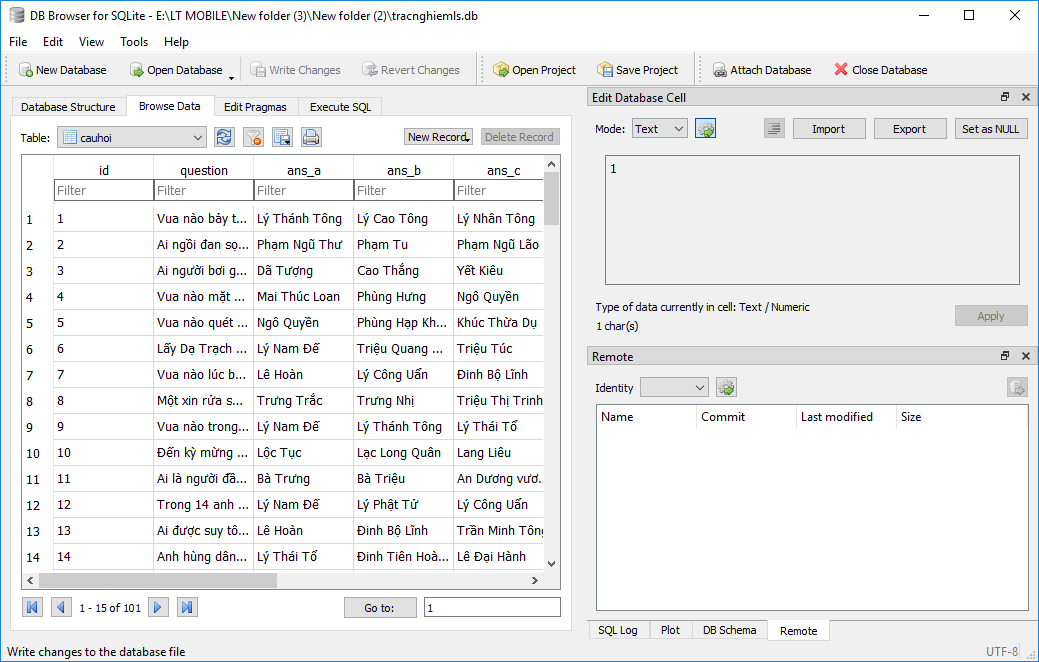
#### **Cấu trúc Table**



#### **Database Structure**



#### **Database**



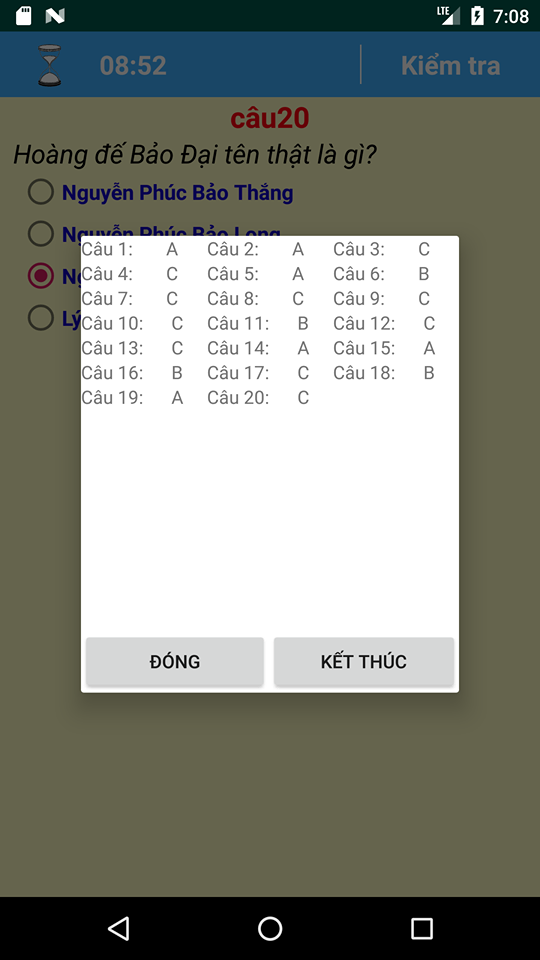
## 3.2. Giao diện

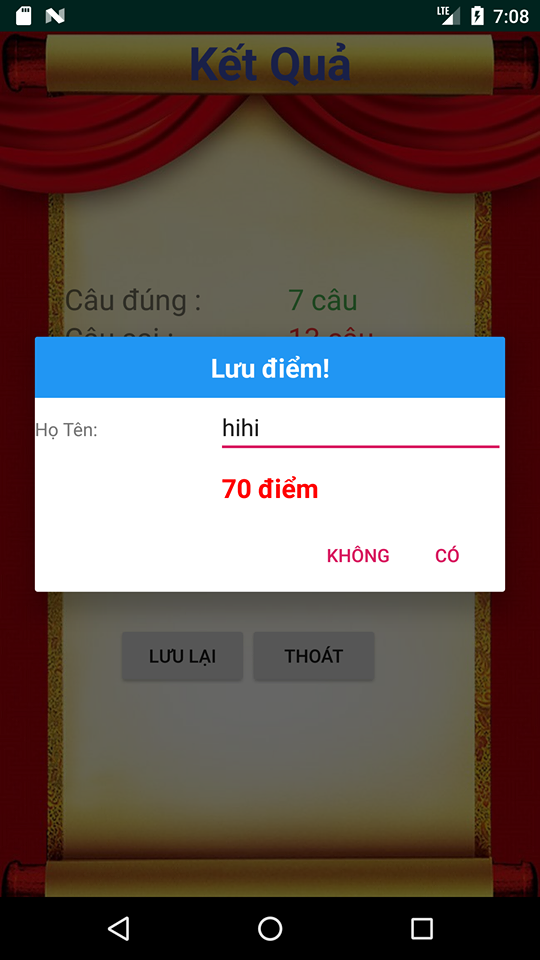
3.2.1. Giao diện màn hình chính

3.2.2 Giao diện giới thiệu

3.2.3 Giao diện trải nghiệm

3.2.4 Giao diện bắt đầu

3.2.5 Giao diện kiểm tra

3.2.6 Giao diện khi lưu thông tin người chơi

3.2.7 Giao diện kết quả

3.2.8 Giao diện bảng vàng

# CHƯƠNG 4: KẾT LUẬN KIẾN NGHỊ

## KẾT QUẢ THỰC NGHIỆM

Trong quá trình thực nghiệm đã giải quyết được yêu cầu cơ bản của người dùng về App trắc nghiệm lịch sử Việt Nam.

Tuy nhiên App đang trong giai đoạn phát triển nên còn nhiều thiếu sót chưa hoàn thiện, nhóm sẽ cố gắng trong thời gian tới để đáp ứng được nhu cầu của người dùng tốt hơn.

## ĐÁNH GIÁ

**Ưu điểm**

- Giúp người dùng có cơ hội kiểm tra kiến thức lịch sử Việt Nam dễ dàng và nhanh chóng.  
- Các câu hỏi bao quát rộng rãi thể hiện toàn cảnh nội dung lịch sử qua các triều đại.  
- Có phần giải thích các đáp án trả lời đúng

**Khuyết điểm**

- App đang trong quá trình thử nghiệm và phát triển nên còn nhiều thiếu sót.  
- Phần thiết lập chưa kịp triển khai.

## HƯỚNG PHÁT TRIỂN

Kiểm tra quy trình: trong mỗi lĩnh vực kinh doanh sẽ luôn luôn có những quy tắc và quy định cụ thể được áp đặt bởi đặt thù của ngành hoặc của cơ quan nhà nước. Điều này có thể trở thành một rào cản trong việc kiểm tra các tính năng của ứng dụng có chính xác theo yêu cầu.

Thị trường cạnh tranh: Các cá nhân, doanh nghiệp yêu cầu viết ứng dụng thường hoạt động trong các lĩnh vực kinh doanh có môi trường cạnh tranh khốc liệt. Chi phí viết app vẫn là yếu tố quan trọng cho khách hàng lựa chọn và quyết định.

Dịch vụ hỗ trợ khách hàng: các hệ thống lớn thì dịch vụ hỗ trợ chuyên nghiệp là mối quan tâm lớn của khách hàng. Đặt biệt là xử lý công việc xảy ra vào ban đêm. Khách hàng không cần gì ngoài hỗ trợ 24/24

Tạo doanh thu: mô hình kinh doanh là một vấn đề lớn đối với việc xây dựng nền tảng ứng dụng theo yêu cầu. Thật sự các nền tảng ứng dụng theo yêu cầu thiếu mô hình kinh doanh hợp lý.

## TÀI LIỆU THAM KHẢO

[1]Nguồn tài liệu của thầy Thanh

[2] <https://unica.vn/>

[3] <https://stackoverflow.com/>