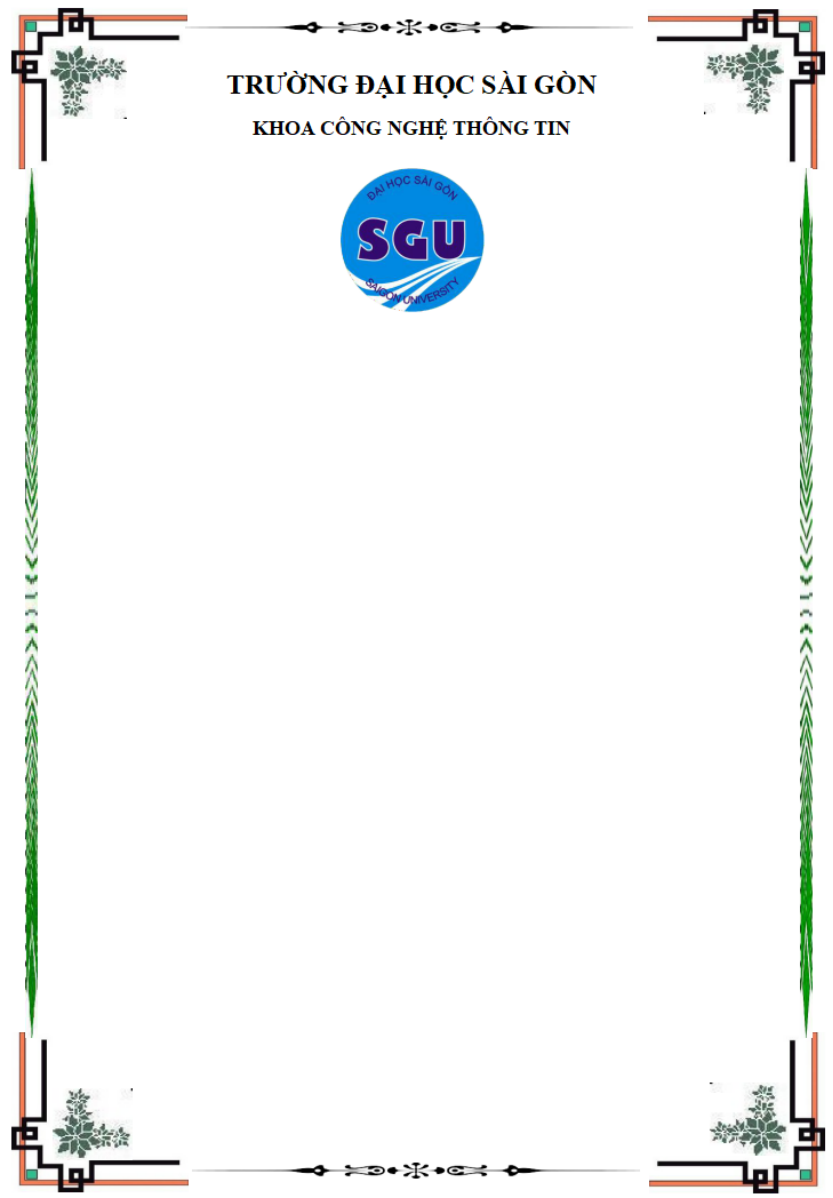
****

**BÁO CÁO ĐỀ TÀI MÔN HỌC**

**NHẬP MÔN PHÁT TRIỂN ỨNG DỤNG TRÊN THIẾT BỊ DI ĐỘNG**

**ĐỀ TÀI: SERVICE TRONG ANDROID**

**Giáo viên hướng dẫn: Phan Nguyệt Minh**

**NHÓM 11**

**Tên Thành viên:**

*Nguyễn Quốc Vương\_3119410499*

*Lê Anh Tú \_ 3119410475*

*Nguyễn Minh Trung\_3119410468*

*Lê Huỳnh Thanh Sang\_3119410344*

# MỤC LỤC

[**MỤC LỤC 1**](#_heading=h.30j0zll)

[**NHẬN XÉT CỦA GIẢNG VIÊN 2**](#_heading=h.3znysh7)

[**CHƯƠNG I. TỔNG QUAN VỀ ĐỀ TÀI 4**](#_heading=h.2et92p0)

[**1.1. Giới thiệu 4**](#_heading=h.tyjcwt)

[**1.2. Lý do lựa chọn đề tài 4**](#_heading=h.3dy6vkm)

[**1.3. Mục tiêu nghiên cứu 4**](#_heading=h.1t3h5sf)

[**1.4. Nội dung báo cáo 4**](#_heading=h.4d34og8)

[**1.5. Phạm vi nghiên cứu 4**](#_heading=h.2s8eyo1)

[**1.6. Phương pháp nghiên cứu 5**](#_heading=h.17dp8vu)

[**CHƯƠNG II. CÁC KHÁI NIỆM SERVICE TRONG ANDROID 6**](#_heading=h.3rdcrjn)

[**2.1 Khái niệm 6**](#_heading=h.26in1rg)

[**2.2. Phân loại Service 7**](#_heading=h.lnxbz9)

[**2.2.1 Foreground Service (Unbound Service) 7**](#_heading=h.35nkun2)

[**2.2.2 Background Service (Unbound Service) 8**](#_heading=h.1ksv4uv)

[**2.2.3. Bound Service 8**](#_heading=h.44sinio)

[**2.3. Độ ưu tiên 8**](#_heading=h.2jxsxqh)

[**2.4. Khai báo trong Manifest 8**](#_heading=h.z337ya)

[**2.5. Các giá trị trả về trong Onstartcommand() 9**](#_heading=h.3j2qqm3)

[**2.5.1 START\_NOT\_STICKY 9**](#_heading=h.1y810tw)

[**2.5.2 START\_STICKY 9**](#_heading=h.4i7ojhp)

[**2.5.3 START\_REDELIVER\_INTENT 9**](#_heading=h.2xcytpi)

[**2.5.4 START\_STICKY\_COMPATIBILITY 9**](#_heading=h.1ci93xb)

[**CHƯƠNG III. CÁC PHƯƠNG THỨC TRONG VÒNG ĐỜI ANDROID VÀ CÁC VẤN ĐỀ KHI SỬ DỤNG SERVICE 10**](#_heading=h.3whwml4)

[**3.1. Các phương thức quan trọng trong vòng đời android 10**](#_heading=h.qsh70q)

[**3.1.1. onStartCommand(). 10**](#_heading=h.3as4poj)

[**3.1.2. onBind(). 10**](#_heading=h.1pxezwc)

[**3.1.3. onCreate(). 10**](#_heading=h.49x2ik5)

[**3.1.4. onDestroy(). 10**](#_heading=h.2p2csry)

[**3.2 Các vấn đề khi sử dụng service trong android 11**](#_heading=h.147n2zr)

[**3.2.1. Service đồng thời là Foreground Service và Bound Service 11**](#_heading=h.3o7alnk)

[**3.2.2 Phân biệt Background và Foreground 11**](#_heading=h.23ckvvd)

[**CHƯƠNG IV. ĐÁNH GIÁ VÀ KẾT LUẬN 12**](#_heading=h.ihv636)

[**4.1. Tóm tắt kết quả nghiên cứu 12**](#_heading=h.1hmsyys)

[**4.2. Đánh giá ưu và nhược điểm 12**](#_heading=h.41mghml)

[**4.3. Hướng phát triển tương lai 12**](#_heading=h.2grqrue)

[**4.4. Kết luận 12**](#_heading=h.vx1227)

[**TÀI LIỆU THAM KHẢO 13**](#_heading=h.3fwokq0)

# NHẬN XÉT CỦA GIẢNG VIÊN

# CHƯƠNG I. TỔNG QUAN VỀ ĐỀ TÀI

## 1.1. Giới thiệu

Trong thời đại công nghệ ngày nay, điện thoại di động Android không chỉ là một thiết bị để liên lạc mà còn là một trung tâm giải trí và công việc đa nhiệm. Điều này đã đặt ra nhu cầu ngày càng tăng về các dịch vụ và ứng dụng đa dạng để cung cấp trải nghiệm người dùng tốt nhất. Báo cáo này sẽ tập trung nghiên cứu về đối tượng là "Service trong Android," một khía cạnh quan trọng trong phát triển ứng dụng di động.

## 1.2. Lý do lựa chọn đề tài

Service là một thành phần quan trọng của hệ thống Android, cung cấp khả năng thực hiện các tác vụ dài hạn mà không làm gián đoạn trải nghiệm người dùng. Việc hiểu rõ về cách Service hoạt động là chìa khóa để phát triển các ứng dụng linh hoạt, hiệu quả và có khả năng phản hồi tốt từ người dùng. Đồng thời, nghiên cứu về Service cũng là nền tảng cho việc tối ưu hóa tài nguyên và quản lý vòng đời của ứng dụng Android.

## 1.3. Mục tiêu nghiên cứu

Mục tiêu của đề tài là tìm hiểu sâu rộng về khái niệm Service trong Android, bao gồm cách triển khai, vòng đời, và ứng dụng thực tế trong phát triển ứng dụng di động. Nghiên cứu sẽ tập trung vào các khía cạnh quan trọng như Service thông thường, IntentService, Bound Service, và Foreground Service để hiểu rõ cách chúng tương tác với hệ thống vào nhau.

## 1.4. Nội dung báo cáo

Báo cáo này sẽ được tổ chức thành nhiều chương, mỗi chương tập trung vào một khía cạnh cụ thể của Service trong Android. Sau phần tổng quan này, chương 2 trình bày giới thiệu về Service trong Android. Chương 3 trình bày các phương thức quan trọng trong vòng đời của Service và các vấn đề khi sử dụng Service trong Android. Cuối cùng, chương 4 sẽ đánh giá và kết luận.

## 1.5. Phạm vi nghiên cứu

Phạm vi của nghiên cứu này sẽ tập trung chủ yếu vào Service trong ngữ cảnh của hệ điều hành Android. Các ứng dụng và ví dụ cụ thể sẽ được sử dụng để minh họa các khái niệm và phương pháp được thảo luận trong báo cáo.

## 1.6. Phương pháp nghiên cứu

Phương pháp nghiên cứu sẽ bao gồm việc phân tích tài liệu, xây dựng ứng dụng mẫu, và thực nghiệm trên thiết bị thực tế. Việc này sẽ giúp đảm bảo rằng các kết quả của nghiên cứu là thực tế và có áp dụng trong các tình huống thực tế của phát triển ứng dụng Android.

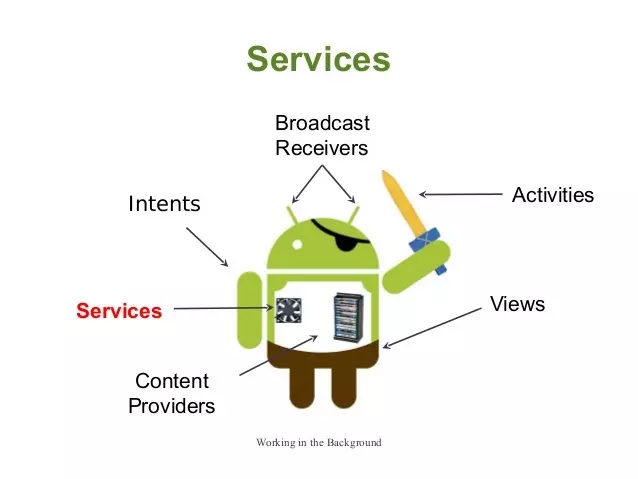
# CHƯƠNG II. CÁC KHÁI NIỆM SERVICE TRONG ANDROID

## 2.1 Khái niệm

Service là một thành phần ứng dụng (application component) có thể thực hiện các hoạt động lâu dài trong background.

Nó không cung cấp giao diện cho người dùng. Sau khi bắt đầu chạy, một Service có thể tiếp tục chạy trong một thời gian, ngay cả khi người dùng chuyển sang ứng dụng khác.

Ngoài ra một thành phần có thể liên kết (bind) với một Service để tương tác với Service đó, thậm chí là thực hiện giao tiếp giữa các quá trình IPC (interprocess communication).



**Hình 2.1**. Cơ cấu Android

Hình 2.1 mô tả một cơ cấu Android với một robot chiến binh. Mỗi phần của robot đại diện cho một thành phần cụ thể trong hệ thống Android:

* Thanh kiếm (Sword) - Activity: Thanh kiếm đại diện cho Activity trong Android. Activity là thành phần chịu trách nhiệm hiển thị giao diện người dùng và tương tác với người dùng thông qua các màn hình.
* Hai chiếc anten (Antennas) - Broadcast Receiver: Hai chiếc ăng ten của robot tượng trưng cho Broadcast Receiver. Broadcast Receiver là thành phần cho phép ứng dụng lắng nghe và phản ứng khi có các thông điệp (Intents) được gửi đi trong hệ thống.
* Cánh tay (Arm) - Intent: Cánh tay của robot biểu thị Intent. Intent là cách mà các thành phần trong hệ thống Android có thể gửi thông điệp và tương tác với nhau.
* Đôi chân (Legs) - View: Đôi chân của robot đại diện cho View trong Android. View là thành phần hiển thị giao diện người dùng, bao gồm các phần tử như nút bấm, hình ảnh, và văn bản.
* Content Provider - Nhiên liệu (Fuel) cho robot: Content Provider giống như một nơi cung cấp nhiên liệu cho robot. Nó là thành phần cho phép ứng dụng lưu trữ và truy cập dữ liệu trong cơ sở dữ liệu chia sẻ.
* Service - Quạt gió (Fan) liên tục chạy: Service là một thành phần không hiển thị giao diện người dùng, được tượng trưng bằng chiếc quạt gió liên tục chạy. Service có khả năng thực hiện các hoạt động lâu dài trong background, ngay cả khi các thành phần khác của ứng dụng đang hoạt động hoặc không nhìn thấy.

Service có thể thực hiện nhiều công việc khác nhau từ background, bao gồm:

* Gửi và nhận dữ liệu qua mạng.
* Phát nhạc và quản lý media playback.
* Thao tác với I/O file.
* Tương tác với Content Provider để truy xuất dữ liệu từ cơ sở dữ liệu

## 2.2. Phân loại Service

### 2.2.1 Foreground Service (Unbound Service)

Foreground Service thực hiện các công việc mà người dùng dễ nhận thấy như phát nhạc, tải xuống dữ liệu, hoặc thậm chí là điều khiển ứng dụng.

Điều quan trọng là Foreground Service phải hiển thị một “Notification” để thông báo cho người dùng biết rằng Service đang hoạt động.

Foreground Service tiếp tục chạy ngay cả khi ứng dụng khác đang hoạt động.

### 2.2.2 Background Service (Unbound Service)

Background Service thực hiện các công việc mà người dùng không nhìn thấy trực tiếp, ví dụ như thu thập dữ liệu hoặc xử lý tác vụ định kỳ.

Đây là loại Service thông thường sử dụng trong các tác vụ nền.

### 2.2.3. Bound Service

Bound Service là loại Service mà các thành phần khác của ứng dụng có thể ràng buộc với bằng lời gọi bindService().

Bound Service cung cấp giao diện Client-Server cho phép tương tác giữa các thành phần và IPC.

Nó chỉ hoạt động khi ít nhất một thành phần ràng buộc với nó, và sẽ bị hủy khi không còn thành phần nào ràng buộc.

## 2.3. Độ ưu tiên

Hệ thống Android bắt buộc phải dừng một Service khi bộ nhớ ít và phải khôi phục tài nguyên hệ thống cho Activity đang được sử dụng. Nếu Service được ràng buộc với một Activity đang sử dụng sẽ ít có khả năng bị hủy, và nếu Service được khai báo và chạy ở chế độ Foreground thì cũng khó có thể bị hủy.

Nếu Service là Started và chạy lâu dài, hệ thống sẽ làm giảm vị trí ưu tiên của nó. Như vậy các loại service sẽ được xếp theo độ ưu tiên sau: Bound Service khó bị hủy nhất, tiếp theo là Foreground Service và Background Service.

***Bound > Foreground > Background***

## 2.4. Khai báo trong Manifest

Bạn muốn dùng một Service bất kỳ thì bắt buộc bạn phải khai báo nó trong Manifest như Activity và các thành phần khác.

<manifest ... >

...

<application ... >

**<service android:name=".ExampleService" />**

...

</application>

</manifest>

Nếu service có thể cho ứng dụng khác ngoài ứng dụng chứa nó sử dụng thì ta cần khai báo thêm thuộc tính *android: exported = "true"* ( mặc định là false).

## 2.5. Các giá trị trả về trong Onstartcommand()

Trong Android, khi các Service tiếp tục chạy trong nền, chúng tiêu tốn bộ nhớ của thiết bị để thực thi. Khi thiết bị Android có nhiều ứng dụng đang chạy và bộ nhớ gần cạn, hệ thống Android sẽ bắt đầu kết thúc các quá trình để giải phóng bộ nhớ.

Tuy nhiên, trong một số trường hợp, có thể đang thực hiện các nhiệm vụ quan trọng bằng các Service, và việc dừng chúng có thể ảnh hưởng đến hoạt động của ứng dụng. Vì vậy, các khái niệm như:

* + START\_NOT\_STICKY,
  + START\_STICKY,
  + START\_REDELIVER\_INTENT,
  + START\_STICKY\_COMPATIBILITY .

được sử dụng để chỉ định hành động cụ thể mà hệ thống Android nên thực hiện khi bộ nhớ thiết bị ổn định trở lại và khi nó sẵn sàng khởi động lại các dịch vụ.

### 2.5.1 START\_NOT\_STICKY

Yêu cầu hệ thống không cần khởi động lại dịch vụ, ngay cả khi có đủ bộ nhớ.

### 2.5.2 START\_STICKY

Yêu cầu hệ thống tạo một bản sao mới của Service, khi có đủ bộ nhớ. Ở đây bạn sẽ mất kết quả mà có thể đã tính toán trước đó.

### 2.5.3 START\_REDELIVER\_INTENT

Yêu cầu hệ thống khởi động lại dịch vụ sau sự cố và cũng phân phối lại các Intents có tại thời điểm xảy ra sự cố.

### 2.5.4 START\_STICKY\_COMPATIBILITY

Giá trị này cũng giống như START\_STICKY nhưng nó không chắc chắn, đảm bảo khởi động lại service.

# CHƯƠNG III. CÁC PHƯƠNG THỨC TRONG VÒNG ĐỜI ANDROID VÀ CÁC VẤN ĐỀ KHI SỬ DỤNG SERVICE

## 

## 3.1. Các phương thức quan trọng trong vòng đời android

Khi tạo một service, phải kế thừa lớp Service của Android cung cấp. Khi thực thi phải override một vài phương thức quan trọng xử lý trong vòng đời của Service và cung cấp một cơ chế cho phép các thành phần liên kết với Service nếu thích hợp. Sau đây là một số phương thức quan trọng đó:

### 3.1.1. onStartCommand().

* Hệ thống gọi phương thức này khi một thành phần khác (Activity chẳng hạn) gọi đến Service bằng câu lệnh startService(). Khi phương thức này được thực hiện, Service được khởi động và có thể chạy trong background vô thời hạn.
* Khi công việc hoàn thành bạn nên stop bằng cách gọi stopService() từ một thành phần khác, hoặc cho chính Service gọi stopSelf(). Nếu chỉ muốn ràng buộc (bind) với Service thì không nên sử dụng onStartCommand().

### 3.1.2. onBind().

* Hệ thống sẽ gọi phương thức này khi một thành phần khác gọi đến Service bằng câu lệnh bindService(). Khi triển khai phương thức này cần phải cung cấp một giao diện để client có thể giao tiếp với Service thông qua một đối tượng IBinder do Service trả về.
* Khi kế thừa từ lớp Service của Android cần phải luôn luôn override phương thức này, nhưng nếu không muốn ràng buộc (bind) với Service, có thể return null.

### 3.1.3. onCreate().

* Hệ thống gọi phương thức này khi Service được khởi tạo, và nó chỉ chạy một lần trước khi onStartCommand() hoặc onBind() được gọi. Nếu Service đã chạy thì phương thức này không được gọi lại lần nào nữa.

### 3.1.4. onDestroy().

* Hệ thống gọi phương thức này khi Service không được sử dụng nữa và đang bị hủy (destroy). Nên giải phóng tài nguyên như các Threads, Listeners hay Receivers ở đây. Đây là phương thức cuối cùng được gọi của Service.
* Nếu một thành phần của ứng dụng khởi động Service bằng cách gọi startService() (dẫn đến một cuộc gọi đến onStartCommand()), thì Service sẽ tiếp tục chạy cho đến khi nó tự dừng với stopSelf() hoặc một thành phần khác dừng nó bằng cách gọi stopService().
* Nếu một thành phần gọi bindService () để tạo Service và onStartCommand() không được gọi, Service chỉ chạy miễn là thành phần được liên kết (bound) với nó. Sau khi Service không bị ràng buộc (unbound) khỏi tất cả các clients của nó, hệ thống sẽ hủy nó.

## 3.2 Các vấn đề khi sử dụng service trong android

### 3.2.1. Service đồng thời là Foreground Service và Bound Service

Khi Foreground Service gọi stopSelf() hoặc được gọi stopService() thì service vẫn chưa vào onDestroy(), cần một điều kiện nữa để nó bị hủy đó là các thành phần ràng buộc với Bound Service hủy hết liên kết.

Ngược lại nếu tất cả các thành phần ràng buộc với Bound Service unbind thì Service cũng chưa được hủy cho đến khi Foreground Service tự gọi stopSelf() hoặc được gọi stopService().

### 3.2.2 Phân biệt Background và Foreground

Ứng dụng sẽ được coi là Foreground nếu bất kỳ trường hợp nào trong ba trường hợp dưới đây là đúng:

* Ứng dụng hiện có Activity hiển thị.
* Ứng dụng có Foreground Service đang chạy.
* Ứng dụng được kết nối với một ứng dụng Foreground khác bằng cách ràng buộc Service hoặc bằng cách sử dụng đến các Content Providers của chúng.

Còn trường hợp ngoài 3 cái ở trên thì ứng dụng được coi là ở chế độ Background.

# CHƯƠNG IV. ĐÁNH GIÁ VÀ KẾT LUẬN

## 

## 4.1. Tóm tắt kết quả nghiên cứu

Trong quá trình nghiên cứu về dịch vụ trong Android, chúng ta đã hiểu rõ về các khái niệm cơ bản, kiến trúc và ứng dụng của dịch vụ trong môi trường hệ điều hành Android. Các thành phần Service trong Android đã được khám phá chi tiết, tạo ra một bức tranh toàn diện về cách chúng tương tác trong ứng dụng

## 4.2. Đánh giá ưu và nhược điểm

* Ưu Điểm:
  + Tăng Hiệu Suất Ứng Dụng: Sử dụng dịch vụ giúp tăng hiệu suất bằng cách thực hiện các tác vụ nền mà không làm gián đoạn trải nghiệm người dùng.
  + Tương Tác Linh Hoạt: Dịch vụ cung cấp một cơ chế linh hoạt để tương tác giữa các thành phần khác nhau của ứng dụng.
* Nhược Điểm:
  + Tiêu Tốn Tài Nguyên: Dịch vụ có thể tiêu tốn nhiều tài nguyên hệ thống, đặc biệt là khi chúng chạy trong chế độ Foreground.

## 4.3. Hướng phát triển tương lai

Tối Ưu Hóa Tiêu Tốn Tài Nguyên: Nghiên cứu thêm về cách tối ưu hóa tiêu tốn tài nguyên của dịch vụ, đặc biệt là trong môi trường có giới hạn tài nguyên.

Phát Triển Các Mô Hình Tương Tác Mới: Xem xét và phát triển các mô hình tương tác mới giữa các dịch vụ để cải thiện quản lý và hiệu suất hệ thống.

Nghiên Cứu Về Bảo Mật Dịch Vụ: Đối mặt với nguy cơ bảo mật, nghiên cứu về cách bảo vệ dịch vụ khỏi các mối đe dọa tiềm ẩn.

## 4.4. Kết luận

Trong bài nghiên cứu này, chúng ta đã khám phá sâu rộng về dịch vụ trong Android và nhận thức được vai trò quan trọng của chúng trong phát triển ứng dụng di động. Mặc dù có những thách thức, nhưng hiểu biết về các khái niệm và cách triển khai dịch vụ có thể đóng một vai trò quan trọng trong việc tối ưu hóa hiệu suất và trải nghiệm người dùng. Điều này mở ra những cơ hội mới để phát triển ứng dụng Android mạnh mẽ và linh hoạt hơn trong tương lai.

# TÀI LIỆU THAM KHẢO

1. Slide bài giảng môn Nhập môn phát triển ứng dụng di động giảng viên Phan Nguyệt Minh
2. "Android Programming: The Big Nerd Ranch Guide"- Bill Phillips, Chris Stewart, and Kristin Marsicano
3. Trang chính thức của Android Developers - Service
4. "Android Application Development All-in-One For Dummies" - Barry A. Burd
5. Trang web “Android Developer Documentation” - Google
6. “Hướng dẫn tạo dịch vụ (Service) trong Android” - Tinhte Blog
7. “Làm quen với Service trong Android”- Dạo Không Kỹ Thuật Blog
8. “Hướng dẫn sử dụng Service trong Android” - Trần Đức Thái