**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**ĐỒ ÁN TỐT NGHIỆP**

**Thiết kế hệ thống server và ứng dụng android chặn các cuộc gọi làm phiền dựa trên phản hồi của người dùng**

**NGUYỄN VĂN NAM**

nam.nv162840@sis.hust.edu.vn

**Ngành Công nghệ thông tin**

**Chuyên ngành Kỹ thuật máy tính**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | PGS. TS. Lã Thế Vinh  Chữ ký của GVHD |
| **Bộ môn:** | Kỹ thuật máy tính |
| **Viện:** | Công nghệ thông tin và Truyền thông |
| **HÀ NỘI, 6/2021** | |

**ĐỀ TÀI TỐT NGHIỆP**

1. Thông tin sinh viên

Họ và tên: Nguyễn Văn Nam

Lớp: CNTT1.01-K61

Email: nam.nv162840@sis.hust.edu.vn

Điện thoại: 0356955552

Hệ đào tạo: Kỹ sư chính quy

Đồ án tốt nghiệp được thực hiện tại: Bộ môn Kỹ thuật máy tính – Viện Công nghệ thông tin và Truyền thông – Đại học Bách khoa Hà Nội

Thời gian thực hiện: Từ ngày đến ngày

1. Mục đính nội dung của ĐATN

Triển khai hệ thống xử lý các số điện thoại rác/làm phiền/spam tại điểm cuối thiết bị. Hệ thống gồm một ứng dụng điện thoại (Calling App) có chức năng chính là phát hiện, chặn các cuộc gọi làm phiền và một backend server lưu trữ phản hồi từ người dùng để đánh giá, gợi ý cảnh báo tới tất cả người dùng khác.

1. Các nhiệm vụ cụ thể

Xây dựng ứng dụng điện thoại (Calling App)

Xây dựng tính năng Block Phone Numbers

Xây dựng hệ thống Backend Nodejs kết hợp MYSQL

1. Lời cam đoan của sinh viên

Tôi - Nguyễn Văn Nam - cam kết ĐATN này là sản phẩm được thực hiện bởi tôi dưới sự hướng dẫn của PSG. TS. Lã Thế Vinh.

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

1. Xác nhận của giảng viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ.

Giáo viên hướng dẫn

Ký và ghi rõ họ tên

Giáo viên hướng dẫn

Ký và ghi rõ họ tên

**Lời cảm ơn**

Em xin gửi lời cảm ơn tới tất cả các thầy cô và cán bộ tại trường Đại học Bách Khoa Hà Nội và đặc biệt là các thầy cô trong Viện Công nghệ thông tin và Truyền thông. Thầy cô đã giảng dạy và cung cấp các kiến thức, kĩ năng quan trọng, cần thiết trên con đường học tập, nghiên cứu và làm việc của em sau này. Mang tới cho em khát vọng, sự tự tin về khả năng xây dựng công hiến cho đất nước trong qua trình cách mạng hiện đại hóa công nghiệp. Em rất tự hào khi được học tập dưới mái trường Bách Khoa thân yêu đó sẽ là điểm tựa, động lực giúp em phát triển hoàn thiện hơn.

Em xin cảm ơn chân thành thầy PGS. TS. Lã Thế Vinh, thầy đã trực tiếp hướng dẫn và giúp đỡ em rất nhiều. Thầy đã cung cấp cho em các kiến thức chuyên môn và cách giải quyết đối với các vấn đề phát sinh để em có thể hoàn thành tốt đồ án này.

Em cũng tỏ lời cảm ơn tới những người bạn đã luôn sát cánh bên nhau để viết lên những kí ức tươi đẹp về tuổi sinh viên. Chúng ta đã luôn đồng hành giúp đỡ lẫn nhau đi qua những chặng đường trong quá trình học tập vầ nghiên cứu khoa học tại ngôi trường này.

Sau cùng, em xin được cảm ơn đến gia đình và người thân, mọi người đã cố gắng rất nhiều để em có cơ hội được học tập và theo đuổi ước mơ. Nơi mà luôn cổ vũ động viên và tiếp thêm sức mạnh cho em vượt qua mọi thử thách.

Em xin chân thành cảm ơn!

**Tóm tắt nội dung đồ án**

Với sự phát triển bủng nổ phổ biến của smartphone thì vấn đề giải quyết các cuộc gọi rác, tin nhắn rác làm phiền tới trải nghiệm người dùng cũng được đặc biệt quan tâm. Đã có nhiều giải pháp được các bên đưa ra và áp dụng nhưng vì những khó khăn gặp phải mà vẫn nạn trên vẫn chưa được giiar quyết hợp lý.

Trong đồ án này em đề xuất và triển khai thêm một hế thống để giải quyết bài toán cuộc gọi, tin nhắn rác. Hệ thống là chốt chặn cuối cùng trên chính thiết bị smartphone, tập trung vào sự phản hồi của người dùng nhằm phát hiện vầ cảnh báo sớm cuộc gọi, tin nhắn rác.

Sinh viên thực hiện

Ký và ghi rõ họ tên

Sinh viên thực hiện

Ký và ghi rõ họ tên

**MỤC LỤC**

[CHƯƠNG 1. TỔNG QUAN VÀ CƠ SỞ LÝ THUYẾT 1](#_Toc74445855)

[1.1 Giới thiệu chung 1](#_Toc74445856)

[1.2 Giới thiệu giải pháp 1](#_Toc74445857)

[1.3 Các bước triển khai giải pháp 1](#_Toc74445858)

[1.4 Cơ sở lý thuyết sử dụng trong lập trình android 1](#_Toc74445859)

[1.4.1 Giới thiệu về hệ điều hành Android 2](#_Toc74445860)

[1.4.2 Xây dựng ứng dụng gọi điện 3](#_Toc74445861)

[1.4.3 Chặn cuộc gọi – CallScreeningService 6](#_Toc74445862)

[1.5 Cơ sở lý thuyết về Backend NodeJS 8](#_Toc74445863)

[1.5.1 Giới thiệu về NodeJS 8](#_Toc74445864)

[1.5.2 MySQL trong NodeJS 10](#_Toc74445865)

[1.5.3 RESTful API 12](#_Toc74445866)

[CHƯƠNG 2. QUÁ TRÌNH TRIỂN KHAI HỆ THỐNG 13](#_Toc74445867)

[2.1 Xây dựng ứng dụng Android 13](#_Toc74445868)

[2.1.1 Tạo dự án mới 13](#_Toc74445869)

[2.1.2 Xây dựng ứng dụng điện thoại thông thường 15](#_Toc74445870)

[2.1.3 Tính năng chặn cuộc gọi 20](#_Toc74445871)

[2.2 Xây dựng Server NodeJS 23](#_Toc74445872)

[2.2.1 Triển khai cấu hình chung 24](#_Toc74445873)

[2.2.2 Khởi tạo các đối tượng 24](#_Toc74445874)

[2.2.3 Tạo API 25](#_Toc74445875)

[CHƯƠNG 3. THỰC NGHIỆM VÀ KẾT LUẬN 29](#_Toc74445876)

[3.1 Phương pháp thực nghiệm 29](#_Toc74445877)

[3.1.1 Test trên các SDK khác nhau 29](#_Toc74445878)

[3.1.2 Test trên các thiết bị khác nhau 30](#_Toc74445879)

[3.1.3 Một số trường hợp không hoạt động 30](#_Toc74445880)

[3.2 Đánh giá và kết luận 34](#_Toc74445881)

[3.2.1 Đánh giá 34](#_Toc74445882)

[3.2.2 Kết luận 34](#_Toc74445883)

[3.2.3 Định hướng phát triển 34](#_Toc74445884)

[TÀI LIỆU THAM KHẢO 35](#_Toc74445885)

**DANH MỤC HÌNH VẼ**

[Hình 1.1 Kiến trúc hệ điều hành Android 2](#_Toc74446181)

[Hình 1.2 Khai báo quyền cho ứng dụng gọi điện 3](#_Toc74446182)

[Hình 1.3 Khai báo service InCallService trong manifest 4](#_Toc74446183)

[Hình 1.4 Khai báo màn hình giao diện gọi điện 5](#_Toc74446184)

[Hình 1.5 Hiện thông báo chọn ứng dụng điện thoại mặc định 5](#_Toc74446185)

[Hình 1.6 Hiện thông báo khi có cuộc gọi đến 6](#_Toc74446186)

[Hình 1.7 Đăng ký CallScreeningService trong Manifest 7](#_Toc74446187)

[Hình 1.8 Code xin quyền ROLE\_CALL\_SCREENING 7](#_Toc74446188)

[Hình 1.9 Synchronous và Asynchronous 9](#_Toc74446189)

[Hình 1.10 Cơ chế Event Loop 10](#_Toc74446190)

[Hình 1.11 Cấu hình connect mysql 12](#_Toc74446191)

[Hình 2.1 Logo app 13](#_Toc74446192)

[Hình 2.2 Khai báo quyền READ\_CONTACTS trong Manifest 15](#_Toc74446193)

[Hình 2.3 Dialog yêu cầu quyền READ\_CONTACTS 15](#_Toc74446194)

[Hình 2.4 Giao diện màn hình danh bạ 16](#_Toc74446195)

[Hình 2.5 Giao diện màn hình Lịch sử cuộc gọi 17](#_Toc74446196)

[Hình 2.6 Dialog thiết lập ứng dụng điện thoại mặc định 17](#_Toc74446197)

[Hình 2.7 Màn hình thực hiện cuộc gọi mới 18](#_Toc74446198)

[Hình 2.8 Màn hình cuộc gọi đến 19](#_Toc74446199)

[Hình 2.9 Notification cuộc gọi 19](#_Toc74446200)

[Hình 2.10 Dialog thiết lập ứng dụng chặn cuộc gọi mặc định 20](#_Toc74446201)

[Hình 2.11 Màn hình chặn số điện thoại Local 21](#_Toc74446202)

[Hình 2.12 Màn danh sách số block từ server 22](#_Toc74446203)

[Hình 2.13 Icon share block phone 22](#_Toc74446204)

[Hình 3.1 Dialog thiết lập ứng dụng điện thoại mặc định 31](#_Toc74446205)

[Hình 3.2 Mục “Ứng dụng mặc định” trên thiết bị Joy4 32](#_Toc74446206)

[Hình 3.3 Dialog cấp quyền Call logs (Nhật ký cuộc gọi) 33](#_Toc74446207)

**DANH MỤC BẢNG BIỂU**

[Bảng 1.1 So sánh giữa SQL và MySQL 11](#_Toc74446241)

[Bảng 3.1 Test trên SDK khác nhau 30](#_Toc74446242)

[Bảng 3.2 Test trên nhiều thiết bị 30](#_Toc74446243)

# TỔNG QUAN VÀ CƠ SỞ LÝ THUYẾT

## Giới thiệu chung

Ngày nay với sự phát triển công nghệ và kinh tế smartphone đã ngày càng trở nên phổ biến và là thiết bị cần thiết đối với con người trong quá trình làm việc, học tập và giải trí. Một trong nhưng ứng dụng cở bản của điện thoại là cuộc gọi và tin nhắn để giao tiếp một cách nhanh trong. Vì vậy nó bị lợi dụng làm một kênh quảng cáo hoặc lừa đảo tới người dùng, gây ra rất nhiều phiền phức ảnh hửng đến công việc và trải nghiệm của người dùng.

Vấn nạn này đã được quan tâm nhiều hơn trong thời gian gần đây từ nhà nước đến các đơn vị cung cấp dịnh vụ viễn thông. Với nhiều giải pháp được đưa ra như định danh số thuê bao, ra soát quản lý phát hiện từ nguồn của các đơn vị. Nhưng vì nhưng khó khăn gặp phải trong qua trình cung cấp dịch vụ và sự khôn khéo của các đối tượng thì vấn nạn trên vẫn chưa được giải quyết một cách hợp lý.

## Giới thiệu giải pháp

Để giải quyết các cuộc gọi, tin nhắn rác ngoài các biện pháp chốt chặn tại nguồn thì tại đồ án này em đề xuất thêm một giải pháp làm chốt chặn ngay tại thiết bị của người dùng đây sẽ là chốt chặn cuối cùng và người dùng có thể can thiệp và quyết định loại bỏ nó. Hệ thống sẽ cung cấp các công cụ để chặn các số thuê bao gây phiền phức và tiếp nhận các thông tin phản hồi từ người dùng. Hệ thống sễ lưu trữ, đánh giá và gửi cánh báo lại cho tất cả các người dùng khác. Giải pháp này sẽ có hiệu quả ngay lập tức cho mội người dùng nhất định và sẽ rất hiệu quả khi có lượng người dùng đủ lớn để phân tính đánh giá.

## Các bước triển khai giải pháp

1. Xây dựng ứng dụng cuộc gọi cơ bản

Để đáp ứng được nhu cầu cơ bản của một chiếc điện thoại là thực hiện cuộc gọi, nhận cuộc gọi tin nhắn thì cần xây dưng một ứng dụng cuộc gọi đáp ứng đước các tác vụ trên. Ứng dụng có thể nhận và thực hiện cuộc gọi một cách bình thường ngoài ra cần có một giao diện đẹp và tối ưu cho các thao tác từ người dùng.

1. Triển khai tính năng chặn cuộc gọi

Ngoài các tính năng cơ bản trên, ứng dụng có khả năng chặn cuộc các cuộc gọi được người dùng vào. Các cuộc gọi này sẽ không sẽ được tự động hủy đi và sẽ chỉ hiện trong lịch sử của danh sách cuộc gọi.

1. Triển khai hệ thống backend server

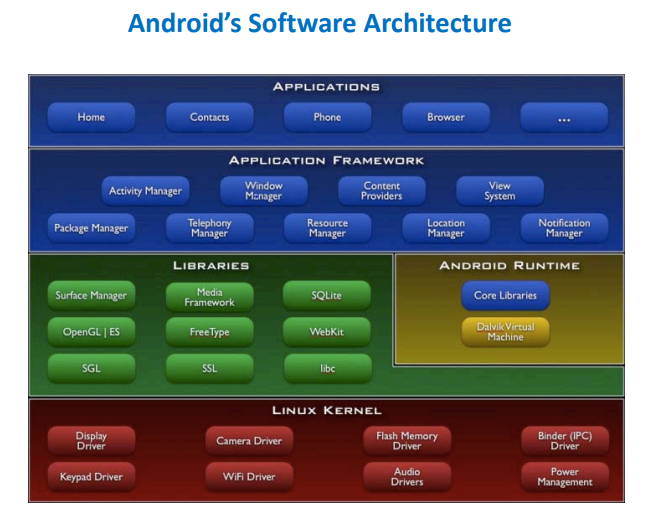
Xây dựng hệ thống server để lưu trữ lại các báo cáo phản hồi từ người dùng gửi từ ứng dụng gửi lên. Hệ thống sẽ phân loại đánh giá và quyết định cho vào danh sách hạn chế. Sau đó danh sách hạn chế sẽ được gửi lại cho tất cả người dùng để cảnh báo trước.

## Cơ sở lý thuyết sử dụng trong lập trình android

### Giới thiệu về hệ điều hành Android

Android là một hệ điều hành mã nguồn mở được phát triển bởi google cho các thiết bị thông minh, thiết bị di động có màn hình cảm ứng như điện thoại thông minh, máy tính bảng, smart tivi,... Với yếu tố mã nguồn mở, sử dụng trên ngôn ngữ lập trình Java với cộng động lập trình viên đông đảo đã thu hút các nhà phát triển ứng dụng và nhanh chóng đưa Android trở thành hệ điều hành phổ biến và có kho ứng dụng lớn nhất thế giới. Android là lựa chọn tốt nhất cho mục tiêu phổ cập thiết bị thông minh trên thế giới. Năm 2017, Android đã chiếm 87,7% thị phần điện thoại thông minh trên thế giới với hơn 2 tỷ thiết bị đã được kích hoạt.

Kiến trúc của hệ điều hành android - gồm 5 tầng:



Hình . Kiến trúc hệ điều hành Android

* Tầng Applications: Tầng này chứa các ứng dụng cơ bản cho người dùng gọi điện, danh bạ, trình duyệt, camera,…
* Tầng Framework: Tầng chứa các API cần thiết để cho việc xây dựng app trên hệ điều hành. Quản lý các view, activity, lấy các thông tin như danh bạ, tin nhắn
* Tầng Libraries: Tầng gồm các thư viện cốt lõi được viết bằng c/c++ kết nối với phần cứng và các bộ phận ngoại vi như Surface Manager, camera, mic, …
* Tầng Android Runtime: Tầng thực hiện convert các ứng dụng sang định dạng nén Dalvik Executable (.dex), cung cấp các tính năng như thread, low-level memory management.
* Tầng Kernel: Là nhân lõi của hệ điều hành dựa trên hệ điều hành linux, chứa các tập lệnh, driver.

Ngôn ngữ lập trình và môi trường phát triển:

Lập trình android là quá trình lập trình viên sử dụng ngôn ngữ lập trình nào đó để xây dựng lên ứng dụng, tiện ích hoạt động trên hệ điều hành android.

Lập trình android có thể sử dụng nhiều ngôn ngữ lập trình như Java, Kotlin, Dart(Flutter), c++, python, …

Java chính là ngôn ngữ đơn giản và phổ biến nhất cho việc lập trình android, là ngôn ngữ hướng đối tượng với khả năng xứ lý tốt, cấu trúc rõ ràng công với kho thư viện, API khổng lồ được cung cấp bởi cộng đồng lập trình viên đông đào thì việc xử lý các yêu cầu cần độ phức tạp cao sẽ trở lên đơn giản hơn.

Kotlin: có nhiều nét tương đồng với Java, đây là ngôn ngữ có cú pháp đơn giản, ngắn gọn hơn. Kể từ 2017, nó được google ưu tiên hỗ trợ nhiều hơn.

Môi trường phát triển(IDE) dành riêng cho android chính là Android Studio. Đây là IDE mã nguồn mở có thể chạy trên cả Windows, Mac và Linux.

### Xây dựng ứng dụng gọi điện

Tại trang web dành cho lập trình viên, google đã cung cấp tài liệu rất chi tiết về ứng dụng gọi điện. Đây là ứng dụng gọi điện cho phép người dùng nhận và thực hiện cuộc gọi âm thanh hoặc video. Nó có thể xây dựng giao diện khác với ứng dụng mặc định của hệ thống để có thể đáp ứng nhưng yêu cầu khác nhau.

Android framework sẽ chứa các gói thư viện android.telecom, các lớp cần thiết để lập trình viên có thể xây dựng một ứng dụng hoản chỉnh.

* Ứng dụng sẽ tương tác chính xácvới hệ thống viễn thông của thiết bị.
* Ứng dụng có thể giao tiếp được với các ứng dụng gọi điện khác nếu cũng viết theo tiểu chuẩn này.
* Framework sẽ giúp ứng dụng quán lý định tuyến âm thanh và video.
* Framework sẽ giúp ứng dụng quản lý phân phối độ ưu tiên của cuộc gọi.

1. Khai báo quyền trong Manifest

Trong file manifest của project cần khai báo các quyền cho hệ thống như:

* MANAGE\_OWN\_CALLS
* READ\_CALL\_LOG
* READ\_PHONE\_STATE
* READ\_PHONE\_NUMBERS



Hình . Khai báo quyền cho ứng dụng gọi điện

1. Triển khai service InCallService

Ứng dụng sẽ thông qua service này để quản lý các cuộc gọi điện thoại. Để có thể làm điều này thì ứng dụng cần được người dùng cài đặt làm ứng dụng điện thoại mặc định.

Ứng dụng điện thoại mặc định(default dialer/phone app) sẽ cung cấp giao diện cho thiết bị khi kết nối cuộc gọi và cung cấp các tiên ích như bắt đầu/kết thúc cuộc gọi, xem lịch sử cuộc gọi. Mỗi thiết bị sẽ có một ứng dụng default dialer/phone app mặc định của hệ thống. Để ứng dụng của chúng ta có thể thay thế ứng dụng hệ thống cần sử dụng RoleManager để yêu cầu đảm nhận vai trò RoleManager.ROLE\_DIALER.

Để đáp ứng vai trò RoleManager.ROLE\_DIALER ứng dụng cần đáp ứng yêu cầu:

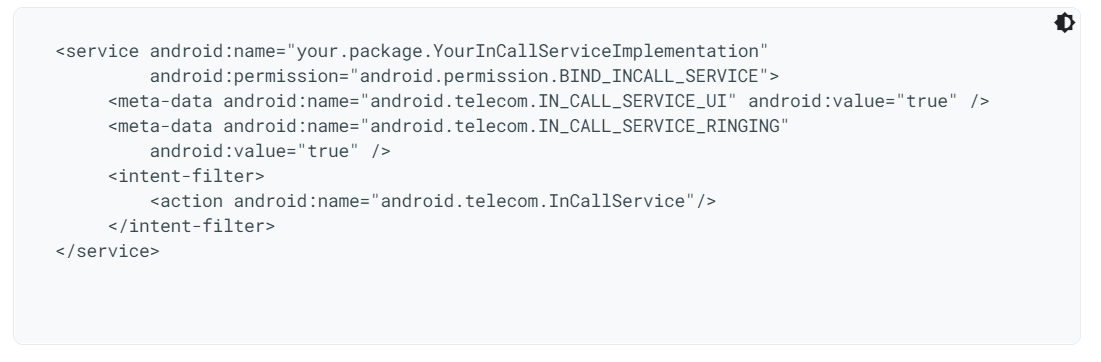
* Xử lý định hướng Intent#ACTION\_DIAL. Có bàn phím số để người dùng có thể thực hiện quay số.
* Triển khai đây đủ API InCallService và có giao diện cho cuộc gói đến, cuộc gọi đi.

Nếu trong quá trình hoạt động ứng dụng gặp lỗi khi triển khai API InCallService thì hệ thống sẽ tự động chuyển định hướng cho ứng dụng mặc định ban đầu của hệ thống thực thi. Và thông báo tới người dùng đã gặp sự cố nhưng cuộc gọi của họ vẫn được tiếp túc trên ứng dụng hệ thống.

Khai báo service trong manifest là điều bắt buộc:

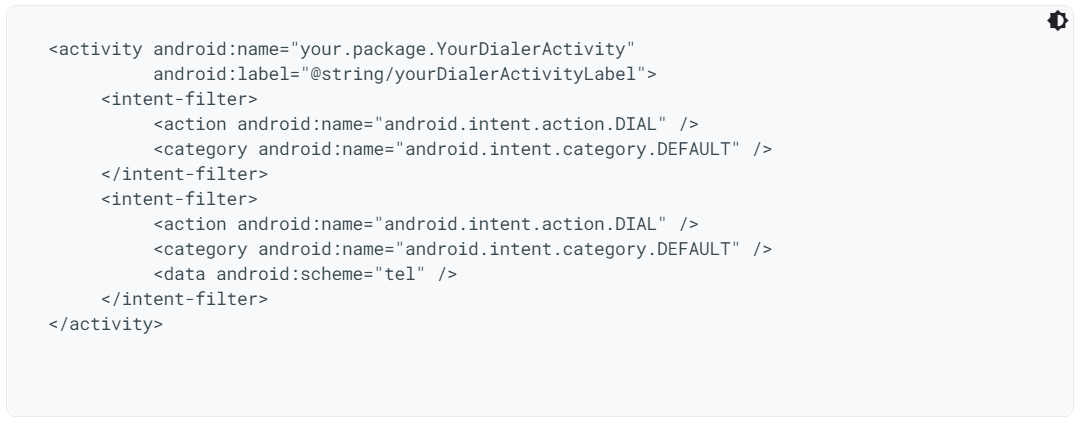
TelecomManager#METADATA\_IN\_CALL\_SERVICE\_UI để thông báo sẽ thay đổi giao diện mặc định.

TelecomManager#METADATA\_IN\_CALL\_SERVICE\_RINGING để thông báo sẽ có nhạc chuông khi có cuộc gọi đến.



Hình . Khai báo service InCallService trong manifest

Ngoài ra còn cần khai báo định hướng Intent#ACTION\_DIAL cho màn hình sẽ làm giao diện gọi điện của ứng dụng.



Hình . Khai báo màn hình giao diện gọi điện

Để ứng dụng của chúng ta làm ứng dụng gọi điện mặc định cần sự đồng ý cho phép đến từ người dùng. Vì vậy khi người dùng truy cập vào ứng dụng cần hiện lên thống báo để người dùng có thể chọn cài đặt ứng dụng gọi điện mặc định.



Hình . Hiện thông báo chọn ứng dụng điện thoại mặc định

Tích hợp với thiết bị đeo tay cần:

* Khai báo quyền MANAGE\_ONGOING\_CALLS trong manifest
* Thiết bị đeo tay cần tích hợp CompanionDeviceManager API
* Triển khai InCallService này với quyền BIND\_INCALL\_SERVICE

1. Hiện thị cuộc gọi đến – InCallService

InCallService#onCallAdded(Call) khi ứng dụng nhận cuộc gọi đến, cần hiện thị giao diện tương ứng cho người dùng. Sử dụng các API NotificationManager để tạo thông báo cuộc gọi đến mới.

TelecomManager#METADATA\_IN\_CALL\_SERVICE\_RINGING nếu được khia báo ứng dụng cần phát nhạc chuông thông báo cuộc gọi đến.



Hình . Hiện thông báo khi có cuộc gọi đến

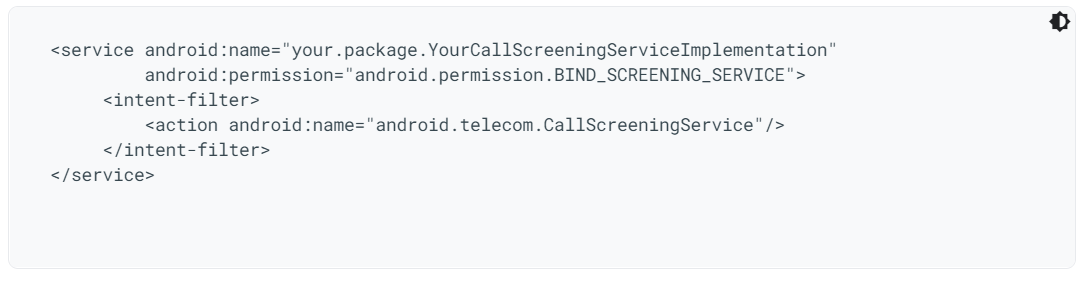
Khi ứng dụng nhận được cuộc gọi đến mới, ứng dụng sẽ tạo thông báo cho cuộc gọi đến và liên kết nó với kênh thông báo cuộc gọi đến của ứng dụng. Chúng ta có thể chỉ định PendingIntent trên thông báo để thiết lập khi chạm vào thông báo sẽ ngay lập tức hiện lên giao diện cuộc gọi toàn màn hình đã được thiết kế của ứng dụng. Nếu người dùng đang sử dụng thì cuộc gọi đến sẽ được hiện thị dưới dạng thông báo, còn người dùng không thao tác trên thiết bị thì cuộc gọi đến sẽ được hiện thị theo cách toàn màn hình.

Một số hàm InCallService cung cấp:

* canAddCall()-boolean: thiết bị có hỗ trợ cuộc gọi?
* getCallAudioState()-CallAudioState: lấy trạng thái hiện tại của âm thanh cuộc gọi
* onBringToForeground(boolean showDialpad): hiện thị bàn phím số lên
* onCallAdded(Call call): hoạt đông khí có thêm cuộc gọi vào phiên quản lý
* onCallAudioStateChanged(CallAudioState audioState): thay đổi trạng thái âm thanh
* onCallRemoved(Call call): hoạt động khi có hành động xóa cuộc gọi khỏi phiên quản lý
* onSilenceRinger(): tắt tiếng chuông điện thoại nếu cuộc gọi đổ chuông
* onUnbind(Intent intent): hoạt động khi đã kết thuốc hoàn toàn cuộc gói đến
* requestBluetoothAudio(BluetoothDevice bluetoothDevice): yêu cầu được định tuyến âm thanh đến thiết bị bluetooth
* Sets the audio route (speaker, bluetooth, etc...): Thiết lập tuyến âm thanh
* setMuted(boolean state): thiết lập trạng thái tắt Mic

### Chặn cuộc gọi – CallScreeningService

Service này được triển khai bởi ứng dụng gọi điện mặc định(Default call phone app) hoặc ứng dụng bên thứ ba đã được cấp quyền để thực hiện các hạnh động trước khi cuộc gọi đến hiện thị cho người dùng. Nó cũng có thể xem các cuộc gọi đi với mục đích cung cấp dịch caller ID services cho cuộc gọi đó.



Hình . Đăng ký CallScreeningService trong Manifest

Hai chức năng chính của một CallScreeningService

* Chặn/sàng lọc cuộc gọi, service có thể lựa chọn cuộc gọi nào sẽ đổ chuông trên thiết bị của người dùng và cuộc gọi nào sẽ được gửi ẩm thâm đến thư thoại.
* Định danh cuộc gọi, service cung cấp chức năng định danh cuộc gọi và có thể hiển thị giao diện tùy ý chưa thông tin định danh cho cuộc gọi.

Triển khai CallScreeningService

Telecom sẽ liên kết với một ứng dụng duy nhất do người dùng chọn, ứng dụng này triển khai API CallScreeningService khi có các cuộc gọi đến và đi mới. Để ứng dụng có khả năng trên thì cần phải yêu cầu quyền thực hiện vai trò sàng lọc cuộc gọi.



Hình . Code xin quyền ROLE\_CALL\_SCREENING

Vòng đời CallScreeningService

RoleManager.ROLE\_CALL\_SCREENING

onScreenCall(android,telecom.Call.Details)

Ứng dụng được người dùng chọn làm ứng dụng chặn cuộc gọi mặc định sẽ nhận được cuộc gọi trước khi đổ chuông và khi cuộc gọi đi được thực hiện. Nó sẽ gọi phương thức onScreenCall để cung cấp cho dịch vụ của chúng ta thông tin chi tiết về cuộc gọi.

responseToCall(android.telecom.CallScreeningService.CallResponse)

Đối với các cuộc gọi đến, CallScreeningService hoạt động trong vòng 5 giây kể từ khi bị ràng buộc để cho nền tảng biết cuộc gọi có nên bị chặn hay không. Cần phải thực hiện các thao tác sàng lọc kịp thời vì thiết bị của người dùng sẽ không bắt đầu đổ chuông cho đến khi nhận được phản hồi(hoặc hết thời gian chờ). Một CallScreeningService có thể chọn thực hiện tra cứu cơ sở dữ liệu cục bộ để giúp xác định xem một cuộc gọi có nên được sàng lọc hay không. Cần chú ý để đảm bảo thời gian chờ không bị nhấn liên tục, gây ra sự chậm trễ trong luồng cuộc gọi đến.

Một số hàm hỗ trợ

* Ibinder onBind(Intent intent): trả về kênh liên lạc của service
* abstract void onScreenCall(Call.Details callDetails): Được gọi khi có cuộc gọi đến hoặc cuộc gọi đi mới
* boolean onUnbind(Intent intent): được gọi khi tất cả các máy khách đã ngắt kết nối

Chi tiết về hàm onScreenCall

Được gọi khi có cuộc gọi đến hoặc cuộc gọi đi mới. Một CallScreeningService phải cho biết cuộc gọi đến có được phép hay không bằng cách gọi CallScreeningService#responseToCall(Call.Details, CallScreeningService.CallResponse). Ứng dụng của chúng ta có thể biết một cuộc gọi có phải là cuộc gọi đến hay không bằng cách kiểm tra xem Call.Details#getCallDirection() có phải là Call hay không.

Lưu ý: Một CallScreeningService phải trả lời cuộc gọi trong vòng 5 giây. Sau thời gian này, khung sẽ hủy liên kết khỏi CallScreeningService và bỏ qua phản hồi của nó.

## Cơ sở lý thuyết về Backend NodeJS

### Giới thiệu về NodeJS

NodeJS là một nền tảng (Platform) được xây dựng trên nền tảng Javascript V8 Engine của Chorme. Được phát triển vào năm 2009 bởi google trong đó phần core sử dụng JavaScript và C++ cho phép xử lý với hiệu năng cao phù hợp với các ứng dụng đòi hỏi real time, xử lý nhanh với cơ chế lập trình bất đồng bộ. Với mục đích xây dựng các ứng dụng mạng nhanh chóng và có thể mở rộng được một cách dễ dàng. NodeJS gọn nhẹ và hiểu quả cho các công việc chuyển sâu về dữ liệu theo thời gian thực, nhờ sử dụng mô hình I/O lập trình theo sự kiện, non-blocking.

Chế độ đồng bộ các dòng lệnh sẽ được thực thi tuần tự, dòng trên thực thi xong thì dòng dưới sẽ bắt đầu được thược thi.

Chế độ bất đồng bộ tất cả dòng lệnh sẽ được thục thi cùng một lúc.

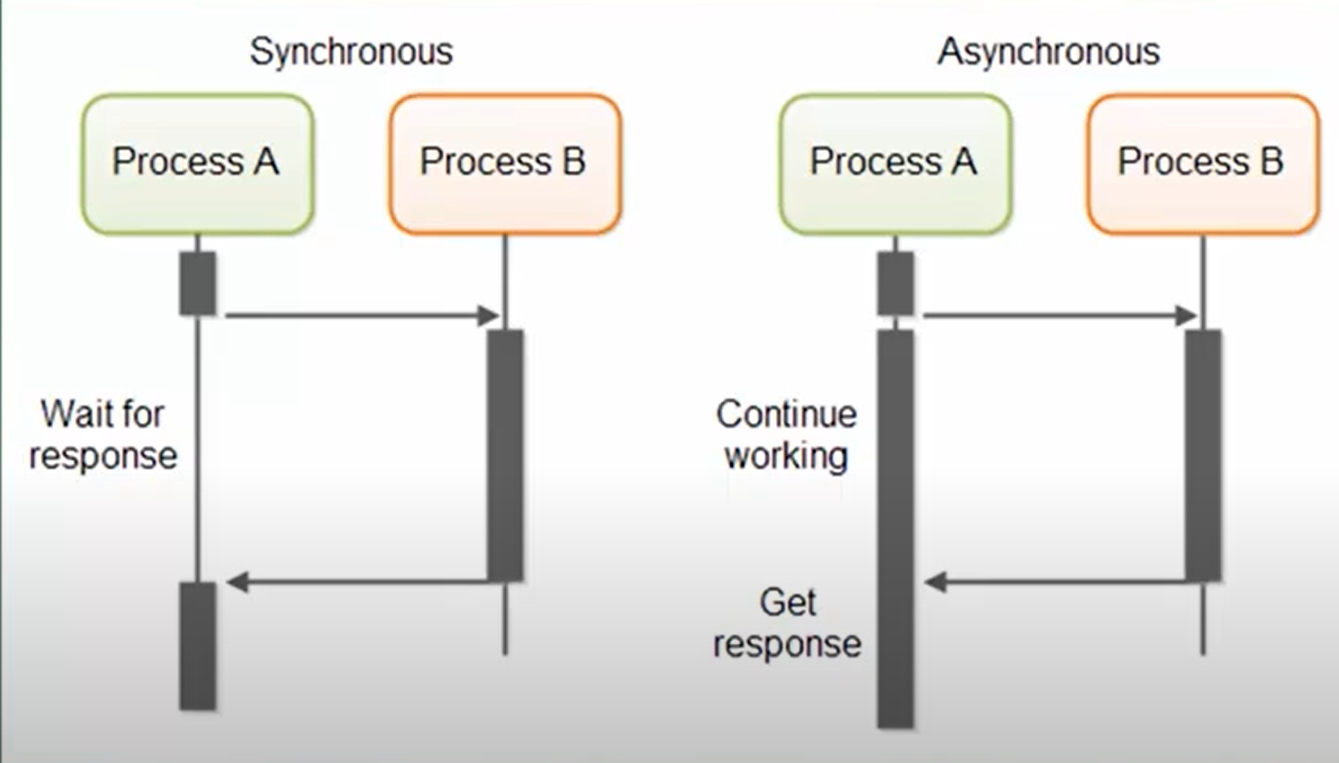
NodeJS là môi trường runtime đa nền tảng mã nguồn mở, thích hợp phát triển các ứng dụng mạng và ứng dụng server-side.

Một số đặc điểm cơ bản NodeJS

Lập trình hướng sự kiện, không đồng bộ: API trong thư viện NodeJS đều không đồng bộ, điều này cũng có nghĩa việc thực thi của một server NodeJS sẽ không chờ một API nào trả về data. Server sẽ chuyển sang API kế tiếp sau khi gọi API trowcs đó và sử dụng cơ chế thông báo cúa Events trong NodeJS để nhận được phản hồi từ các API đã gọi trước đó.

Cực kỳ nhanh chóng: Được xây dưng trên Công cụ JavaScript V8 của google chrome vì vậy thư viện NodeJS có khả năng xử lý mẫ rất nhanh.

Đơn luồng/Single thread: NodeJS sử dụng mô hình đơn luồng với cơ chế vòng lặp sự kiện Events. Cơ chế này cho phép máy chủ phản hồi non-blocking với khả năng cung cấp dịch vụ cho một số lượng yêu cầu lớn hơn so với các máy chủ truyền thống.



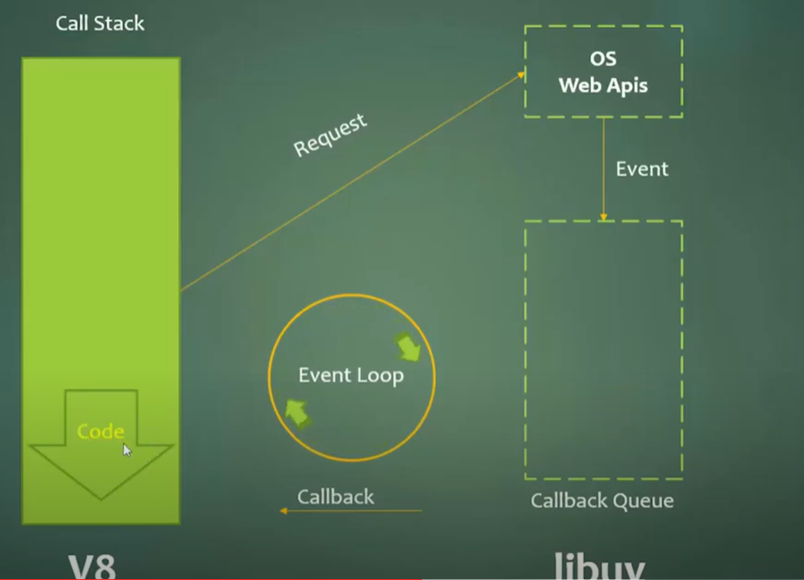
Hình . Synchronous và Asynchronous

Không có buffer: Ứng dụng NodeJS sẽ không có vùng nhớ tam thời(buffer). Cá ứng dụng chỉ đơn giản sẽ xuất dữ liệu theo khối.

License: NodeJS được phát hành theo giấy phép MIT.

Một số thành phần cơ bản

Non Blocking IO-Event Loop: là cơ chế được NodeJS dùng để xử lý đồng thời nhiều request trên single thread mà không cần đợi hoàn thành xử lý của resquest trước đó.



Hình . Cơ chế Event Loop

Package manager: là một kho lưu trữ các package/module để public cho công đồng. Nó dễ dàng cài cặt, quản lý version, quản lý các gói phụ thuộc. Hai package manager phổ biến nhất là npm, yarn

Socket.io: là một công cụ thời gian thực rất nhanh và đáng tin cậy. Nó cho phép giao tiếp 2 chiều theo thời gian thực thông qua các sự kiện. Gồm 2 phần chính là một máy chủ NodeJS và một Javascript client library cho client. Javascript socket.io đã được thêm vào một số ngôn ngữ khác như Java, C++, Dart, Swift.

### MySQL trong NodeJS

Ngôn ngữ SQL

SQL là ngôn ngữ tiêu chuẩn cho các hệ thống quản lý cơ sở dữ liệu quan hệ. Một cơ sở dữ liệu quan hệ sẽ quản lý dữ liệu dưới dạng các bảng. Các câu lệnh SQL sẽ thức thi các tác vụ như cập nhật dữ liệu hoặc truy xuất dữ liệu từ các bảng. Một số hệ quản lý cơ sở dữ liệu sử dụng SQL là MySQL, Oracle, Microsoft SQL Server,..

Hệ quản trị cơ sở dữ liệu MySQL

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở(RDBMS) dựa trên ngôn ngữ truy vấn có cấu trúc(SQL) được phát triển, phân phối và hỗ trợ bởi tập đoàn Oracle. MySQL chạy trên hầu hết tất cả các nền tàng, bao gồm cả Linux, UNIX và Windows.

* MySQL được phát hành theo giấy phép nguồn mở
* MySQL sử dụng dạng chuẩn của ngôn ngữ dữ liệu SQL
* MySQL được tích hợp cho nhiều ngôn ngữ như JAVA, C, C++, NodeJS, PHP
* MySQL hoạt động nhanh và tốt trên các tập dữ liệu lớn. Hỗ trợ lên tới 50 triệu hàng trong 1 bảng. Giới hạn kích thước tệp mặc định cho 1 bẳng là 4GB và có thể nâng thêm theo lý thuyết có thể lên tới 8TB.

Một số ưu điểm của MySQL như dễ dàng sử dụng, tính bảo mật cao, khả năng mở rộng và khôi phục.

So sánh giữa SQL và MySQL:

Bảng . So sánh giữa SQL và MySQL

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **SQL** | **MySQL** |
| Kiểu | Ngôn ngữ truy vấn cấu trúc | Hệ cơ sở dữ liệu sử dụng SQL |
| Hỗ trợ kết nối | SQL không cung cấp trình kết nối | Cung cấp công cụ MySQL workbench |
| Múc đích | Để truy vấn và vận hành hệ thống cơ sở dữ liệu | Cho phép xử lý dữ liệu, lưu trữ, truy vấn theo bảng |
| Sử dụng | Mã và lệnh SQL được sử dụng ở nhiều hệ cơ sở dữ liệu khác nhau | Được sử dụng làm cơ sở dữ liệu RDBMS |
| Cập nhật | Ngôn ngữ là cố định, mã lệnh vẫn được giữa nguyên | Có các bản cập nhật thường xuyên |

Một số câu truy vấn quan trọng

INSERT: chèn dữ liệu vào trong bảng

SELECT: lấy dữ liệu trong bảng

UPDATE: cập nhật dữ liệu đã tồn tại

DELETE: xóa dữ liệu từ một bảng

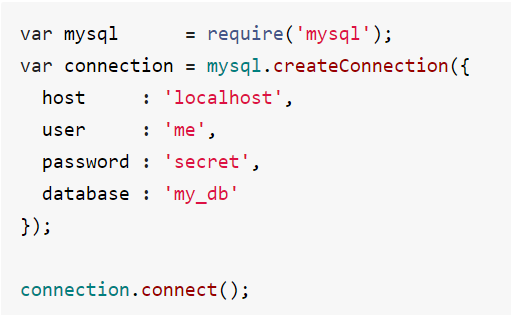
CREATE: tạo ra một bảng dữ liệu

DROP: xóa bảng dữ liệu

Sử dụng MySQL với NodeJS

Tiến hành cài đặt MySQL database và phần mềm quản lý MySQL workbench hoặc Xampp

Cài đặt module mysql qua npm: npm install mysql



Hình . Cấu hình connect mysql

Trong đó:

* host: hostname database của bạn
* user: username để đăng nhập vào mysql
* password: mật khẩu của username để đăng nhập vào mysql
* database: database các bạn cần kết nối

### RESTful API

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng API service. Tập trung vào quản lý trao đổi tài nguyên hệ thống(văn bản, file, âm thanh, video,..), tài nguyên được định danh và truyền tải qua HTTP.

Trong đó:

* API(Application Programming Interface) là một tập các quy tắc và cơ chế để các ứng dụng tương tác qua lại. API sẽ trả về kết quả data dưới dạng cơ bản là JSON, XML,..
* REST(Representational State Transfer) là một dạng chuyển đối cấu trúc dữ liệu, một kiểu kiến trúc cho viết API. Gồm các phương thwucs HTTP đợn giản để giao tiếp giữa các loại thiết bị khác nhau như GET, POST, DELETE.

Chức năng quan trọng nhất của REST là quy định cách sử dụng các HTTP method(GET, POST, PUT, DELETE). RESTful không quy định logic code và không giới hạn bởi ngôn ngữ lập trình, với bất kỳ ngôn ngữ hay framework nào cũng có thể tiếp cận với RESTful API.

REST hoạt động chủ yếu dựa vào giao thức HTTP:

* GET (SELECT): trả về dữ liệu cần thiết cho device user
* POST (CREATE): gửi yêu cầu tạo data mới lên hệ thống
* PUT (UPDATE): cập nhật cho data đã tồn tại trên hệ thống
* DELETE (DELETE): xóa data

Status code trong API khi thực hiện resquest API:

* 200 OK – Trả về thành công cho những phương thức
* 201 Created – Trả về khi một Resouce vừa được tạo thành công.
* 204 No Content – Trả về khi Resource xoá thành công.
* 304 Not Modified – Client có thể sử dụng dữ liệu cache.
* 400 Bad Request – Request không hợp lệ
* 401 Unauthorized – Request cần có auth.
* 403 Forbidden – bị từ chối không cho phép.
* 404 Not Found – Không tìm thấy resource từ URI
* 405 Method Not Allowed – Phương thức không cho phép với user hiện tại.
* 410 Gone – Resource không còn tồn tại, Version cũ đã không còn hỗ trợ.
* 415 Unsupported Media Type – Không hỗ trợ kiểu Resource này.
* 422 Unprocessable Entity – Dữ liệu không được xác thực
* 429 Too Many Requests – Request bị từ chối do bị giới hạn

# QUÁ TRÌNH TRIỂN KHAI HỆ THỐNG

## Xây dựng ứng dụng Android

### Tạo dự án mới

1. Tổng quan project:

* Tên ứng dụng: BlockPhone
* IDE: Android Studio
* applicationId: com.stork.blockspam
* version: 1.0.0
* Support SDK: tất cả các thiết bị kề từ SDK>= 23
* Ngôn ngữ lập trình: Kotlin và Java, với trình biên dịch Kotlin
* Ngôn ngữ ứng dụng: tiếng Anh
* Git: <https://github.com/namnvhustk61/BlockSpamApp>

Icon

Description automatically generated

Hình . Logo app

1. Danh sách màn hình giao diện chính

* Màn hình danh bạ
* Màn hình lịch sử cuộc gọi
* Màn hình danh sách các SĐT chặn
* Màn hình lấy danh sách gợi ý các SĐT chặn trên Server
* Màn hình thêm SĐT chặn thu công
* Màn hình bàn phím phone
* Màn hình khi cuộc gọi đến và đi mới
* Thông báo notification

1. Cấu trúc project

Gồm 3 module: app, http và viewcustom

- Module viewcustom

Gồm các class liên quan đến các thành phần giao diện được tùy biến lại từ các class cơ bản. Bằng cách này sẽ thống nhất giữa các thành phần giao diện dễ dàng thay đổi nâng cấp và thêm các thuộc tính mà các class cơ bản chưa có.

* Thống nhất về màu, size. font giữa các view
* Thống nhất hiểu ứng khi click
* Thêm thuộc tính radius background cho các view
* Thêm các view mới: LoadingView, AlertDialogView, PopupView, SwipeLayoutView,…
* Module http

Gồm các class và thư viên liên quan đến thực thi API HTTP của cả project. Module sẽ sử dụng thưu viên Retrofit để tạo các yêu cầu http client.

Thư viện Retrofit là một HTTP client type-safe cho Android và Java. Nó giúp dễ dàng kết nối dịch vụ REST trên web bằng cách chuyển đổi API thành Java Interface và rất mạnh mẽ xử lý dữ liệu JSON hoặc XML sau đó sẽ phấn tích thành các Objects với các phương thức GET, POST, PUT, PATCH và DELETE.

Folder gồm 3 chính:

* Các class và interface triển khai http client: cấu hình connect, định nghĩa các phương thức sẽ sử dụng
* Model: gồm các class định nghĩa đối tương Objects của client và server là Request Objects , Response Objects
* Class cung cấp các phương thức bất động đồng bộ trong quá trình gọi API
* Module app

Đây là mudule chính của project để khơi chạy. Chứa tất cả các file cấu hình, giao diện và chức năng riêng biệt.

Module sẽ chia thầy các thành phần:

* Manifest: quản lý định nghĩa các thành phần trong project như permission, activity, service, broadcast receiver,..
* Thư mục res: tập các file xml thiết kế đồ họa giao diện như layout, drawable, values,…
* Thư mục java/ui: tập các file ánh xạ điều khiển hoạt động cho layout
* Thư mục java/service: chứa các class có chức năng thư thi cuộc gọi và chặn cuộc gọi: InCallService, CallScreeningService (mục 1.4.2, 1.3.3)
* Các file khác: gồm các file phụ trợ với các tính năng riêng biệt

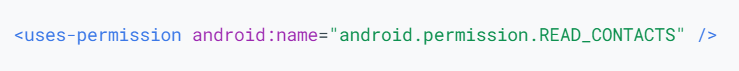
### Xây dựng ứng dụng điện thoại thông thường

Một ứng dụng điện thoại thông thường cần phải có các tính năng cơ bản để hỗ trợ đáp ứng nhu cầu thự hiện cuộc gọi và nhận cuộc gọi mới:

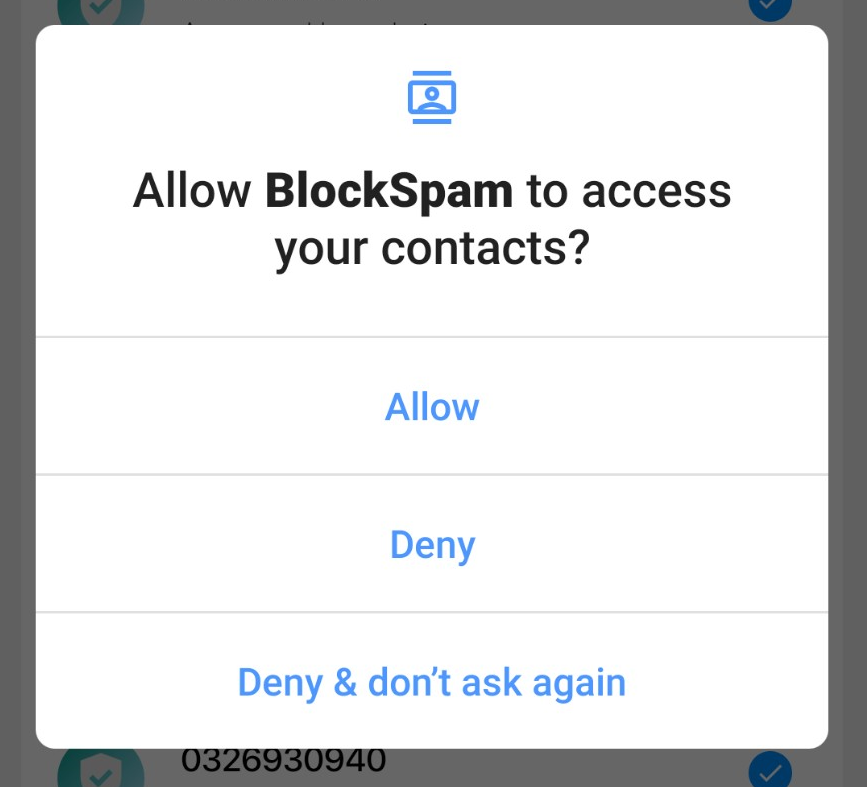
* Hiện thị danh bạ người dùng
* Hiện thị lịch sử cuộc gọi
* Hiển thị màn hình cho cuộc gọi
* Hiện thi thông báo notification khi có cuộc gọi tới

1. Hiển thị danh bạ người dùng

Để lấy danh bạ người dùng từ thiết bị chung ta cần yêu cầu người dùng cấp quyền try cập đọc dạnh bạ READ\_CONTACTS permission. Bằng cách khai báo trong Manifest và hiển thị Dialog yêu cầu cấp quyền từ người dùng.



Hình . Khai báo quyền READ\_CONTACTS trong Manifest

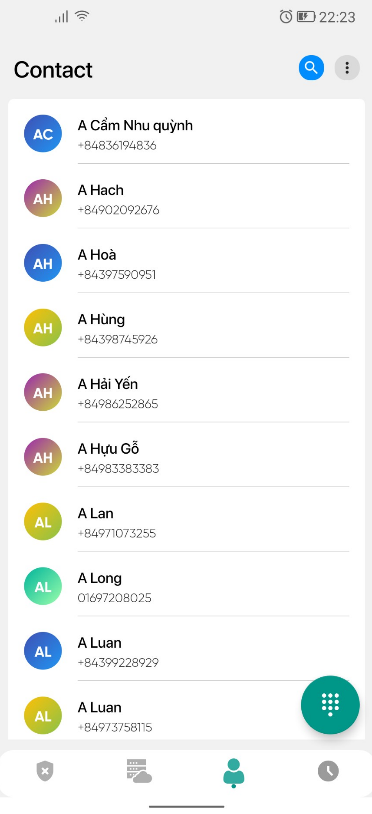


Hình . Dialog yêu cầu quyền READ\_CONTACTS

Danh bạ contact trong Android được lưu trong Database và cung cấp các phương thức để lập trình viên có thể làm việc với nó thông qua Content Provider. Thông tin của một Contact sẽ gồm 13 thành phần:

* Email (address, email type (home, work, mobile, other))
* Instant message (protocol (qq, icq, skype etc), im id).
* Nickname. 4 Organization (company, department, title, job description, office location).
* Phone (number, phone type (home, mobile, work, other)).
* Address (address, address type (home, work, other)).
* Name (display name, given name, family name).
* Postal Address (country, city, region, street, postcode).
* Identity (namespace(SSN, passport), number).
* Photo.
* Group (contact belongs group id).
* Website (website url, website type()).
* Note.

Trong ứng dụng này chúng ta sẽ lấy ra 3 thông tin quan trong nhất là number phone, name và photo.



Hình . Giao diện màn hình danh bạ

1. Hiển thị lịch sử cuộc gọi

Để lấy danh bạ người dùng từ thiết bị chung ta cần yêu cầu người dùng cấp quyền đọc nhật ký cuộc gọi READ\_CALL\_LOG permission. Bằng cách khai báo trong Manifest và hiển thị Dialog yêu cầu cấp quyền từ người dùng.

Nhật ký cuộc gọi trong Android cũng được lưu trong Database và cung cấp các phương thức để lập trình viên có thể truy xuất được nó. Thông tin của một nhật ký cuộc gọi gồm:

* Calls.\_ID: index định dạnh trong database
* Calls.NUMBER: number phone của cuộc gọi
* Calls.CACHED\_NAME: tên đã lưu của số điện thoại
* Calls.CACHED\_PHOTO\_URI: đường dẫn lưu trữ ảnh avatar của số điện thoại
* Calls.DATE: thời gian bắt đầu cuộc gọi
* Calls.DURATION: thời lượng của cuộc gọi
* Calls.TYPE: loại cuộc gọi

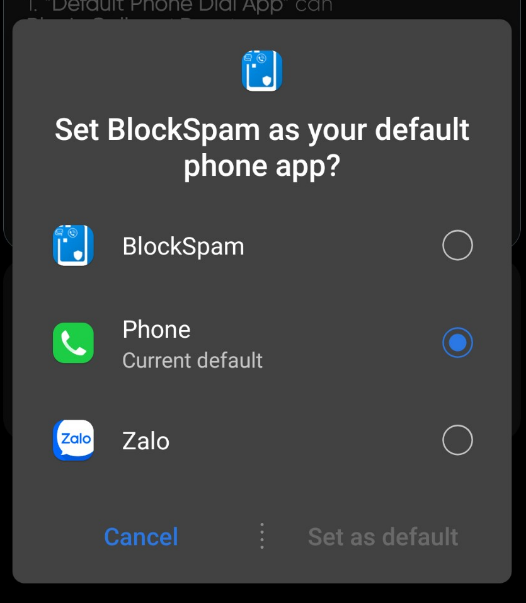
Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình . Giao diện màn hình Lịch sử cuộc gọi

1. Hiển thị màn hình cuộc gọi

Ứng dụng cần phải được người dùng đặt là ứng dụng điện thoại mặc định để có thể thực hiện được các thao tác trong cuộc gọi.



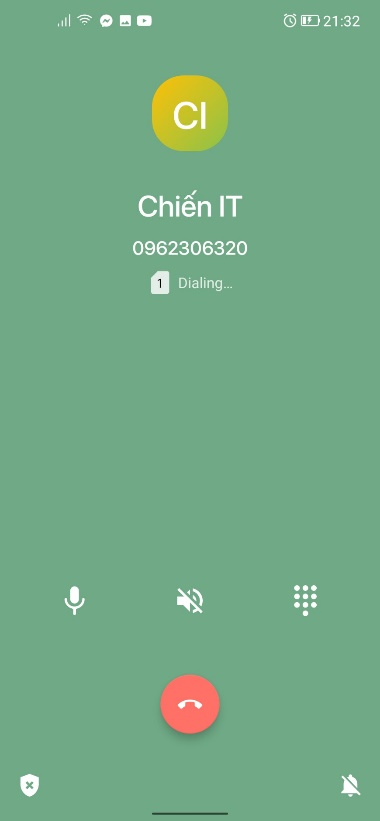
Hình . Dialog thiết lập ứng dụng điện thoại mặc định

Khai báo các quyền permission cần thiết trong Manifest để thống báo cho hệ thống.

* ANSWER\_PHONE\_CALLS
* MANAGE\_OWN\_CALLS
* WAKE\_LOCK: lăng nghe các feature trong khi ứng dụng ngủ
* CALL\_PHONE
* VIBRATE: Điều khiển rung

Khai báo InCallService trong manifest để thông báo cho hệ thống. Mỗi khi có cuộc gọi mới hệ thống sẽ thống báo tới Service này và chúng ta có thể điều khiển được các hành động tương ứng của cuộc gọi.

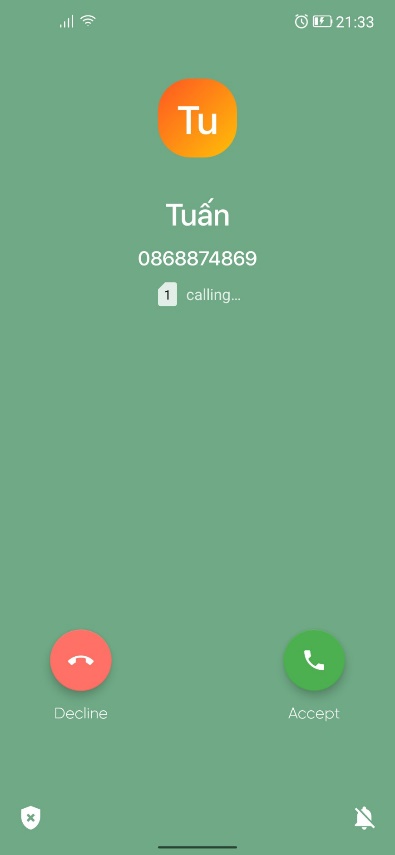
Khi cuộc gọi thông qua đối tượng Call trong hàm onCallAdded(Call call) sẽ lấy được ra 2 thông tin là callPhone và callName. Giao diện người dùng sẽ được mở ra dưới 2 dạng là toàn màn hình hoặc thống báo notification



Hình . Màn hình thực hiện cuộc gọi mới

Màn hình thực hiện cuộc gọi mới sẽ có 4 nút bấm chức năng:

* Kết thục cuộc gọi
* Bật/tắt mic thu
* Bật/tắt loa ngoài
* Hiện bàn phím số



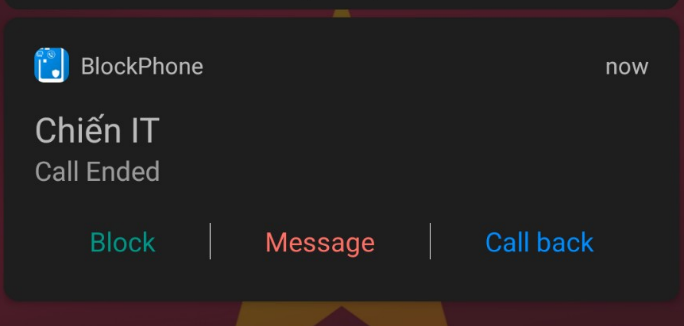
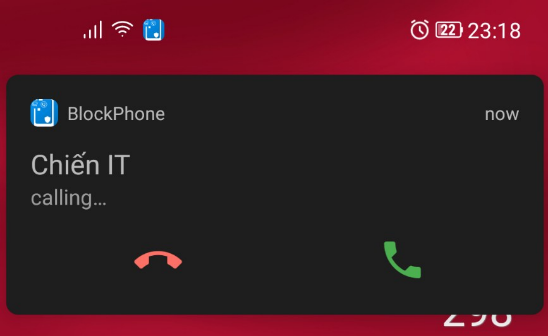
Hình . Màn hình cuộc gọi đến

Màn hình thực hiện nhận cuộc gọi đến mới sẽ có 4 nút bấm chức năng:

* Chấp nhận cuộc gọi
* Kết thúc cuộc gọi
* Tắt chuông và rung
* Chặn số này ngay

1. Hiện thi thông báo notification

Cuộc gọi đến khi ứng dụng ẩn thì giao diện cuộc gọi sẽ là thông báo notification.



Hình . Notification cuộc gọi

Cuộc gọi bắt đầu sẽ có 2 lựa chọn là:

* Accept: Chấp nhận cuộc gọi
* Decline: Hủy cuộc gọi này

Sau khi kết thúc sẽ có thêm các chức năng cho người dùng phàn hồi như:

* Block: thêm số này vào danh sách chặn
* Message: gửi tin nhắn ngay cho số này
* Call back: thực hiện gọi lại ngay

### Tính năng chặn cuộc gọi

1. Thiết lập quyền chặn cuộc gọi

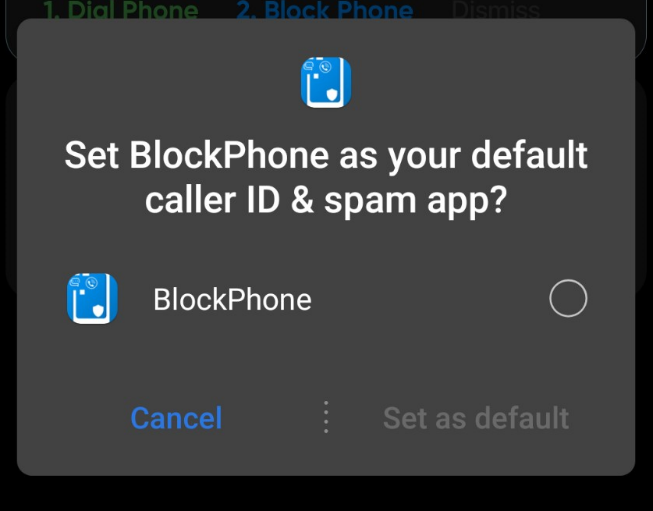
Đây là tính năng cơ bản và quan trọng nhất của ứng dụng. Để có thể chặn được cuộc gọi ứng dụng sẽ tiếp cận với 3 kịch bản sau:

* Ứng dụng là Default Phone App

Vì người dùng đã chọn đặt ứng dụng của chúng ta là ứng dụng điện thoại mặc định vì vậy dùng InCallService ứng dụng có thể can thiệp vào cuộc gọi và Decline nó. Khi cuộc gọi đến hàm onCallAdded(Call call) sẽ được gọi và thông qua CallManager để CallManager.reject().

* Ứng dụng là CallerID&spam app

Với các thiết bị android có SDK >= 26 sẽ có thể triển khai CallScreeningService nếu được người dùng chọn đặt ứng dụng của chúng ta làm ứng dụng chặn cuộc gọi mặc định.



Hình . Dialog thiết lập ứng dụng chặn cuộc gọi mặc định

Ngoài ra ứng dụng còn cần đảm bảo được người dùng cấp thêm 2 quyền permission là: READ\_CONTACTS và READ\_PHONE\_STATE. Để có thể đọc trạng thái cuộc gọi.

Thông qua đối tượng CallScreeningService.CallResponse.Builder() xét các tham số về “true” thì cuộc gọi sẽ được chặn:

* setDisallowCall(true)
* setRejectCall(true)
* setSkipCallLog(true)
* setSkipNotification(true)
* Thiết bị android SDK<26

Với các thiết bị cũ có sdk <26 thì chỉ cần ứng dụng được cấp 2 quyền permission là READ\_PHONE\_STATE và READ\_CALL\_LOG, để có thể gửi intent tới trình gọi điện.

Thông qua đối tượng TelephonyManager và interface Itelephony để thực thi hàm endCall() khi có cuộc gọi tới.

1. Lưu trữ danh sách số bị chặn local

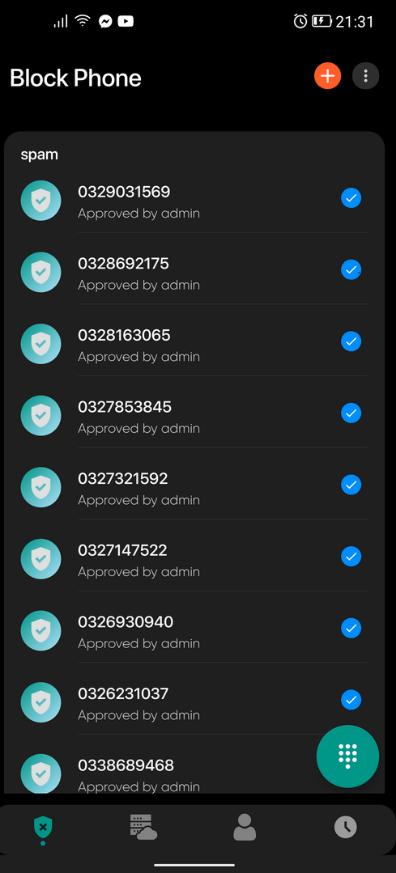
Để lưu trữ các số bị chặn trên thiết bị, ứng dụng đã sử dụng một cơ sở dữ liệu dành cho thiết bị android là RoomDatabase.

RoomDatabase là một thư viện được phát triển và cải tiến từ SQLite, nó giúp đơn giản hóa code và cài đặt liên quán đến cơ sởn dữ liệu hỗ trợ lưu trên thiết bị android. Room database có 3 thành phần chính: Database, DAO(Data Access Object) và entity.

Trong đó entity sẽ là CallPhone với các trường sau:

* Id: trường định dạnh để làm PrimaryKey
* Name: tên của số điện thoại
* Phone: số điện thoại
* Type: phân loại cho block: Local, Online và Other
* Status: với 2 trạng thái hoạt động: Block và UnBlock

CallPhoneDAO sẽ cấp các phương thức để thao tác với database: insert, update, delete, getAll, getByPhone, updateStatusAll.



Hình . Màn hình chặn số điện thoại Local

Khi có cuộc gọi đến ứng dụng sẽ kiểm tra nếu số gọi đến nằm trong danh sách số bị chặn (CallPhone Database) và có Status == Block thì cuộc gọi sẽ bị hủy ngay lập tức.

1. Lấy danh sách Block Server và báo cáo số điện thoại

Dịch vụ Server sẽ cung cấp 2 API:

* getAllPhone

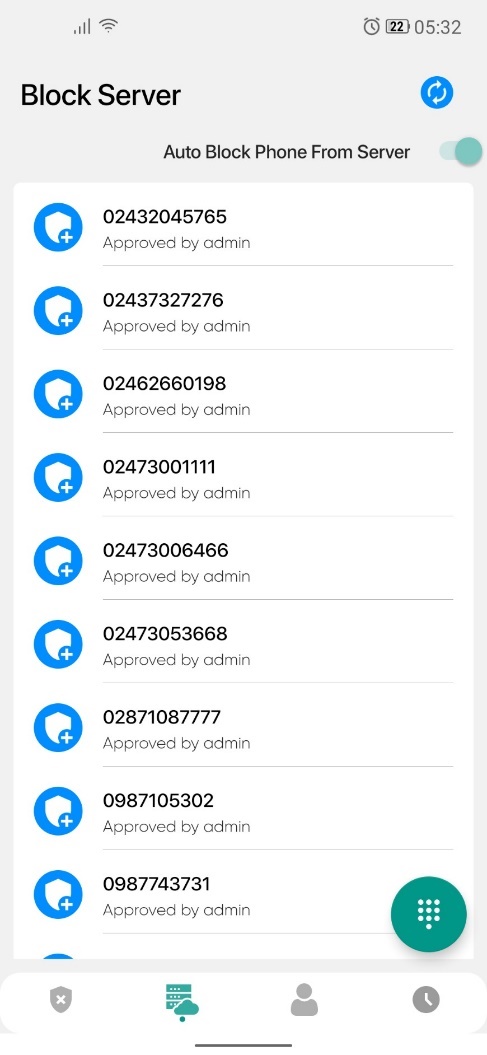
Tất cả các số được hệ thống đánh giá là spam/rác sẽ được lọc và gửi về cho ứng dụng hiển thị.

Phương thức: GET

Path: “/blockphone/all”

Request: Null

Response: List< BlockPhone >



Hình . Màn danh sách số block từ server

* addPhoneCloud

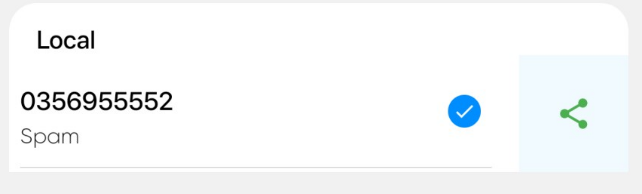
Khi người dùng cho rằng số điên thoại này có thể spam/phiền phức cho nhiều người thì có thể gửi phản hồi cảnh báo nên server bằng cách click vào icon share. Khi đó thông qua API addPhoneCloud server sẽ lưu lại phàn hồi hồi và đánh giá.

Phương thức: POST

Path: “/blockphone/add”

Request: AddPhoneCloud

Response: BlockPhoneRes



Hình . Icon share block phone

## Xây dựng Server NodeJS

Tổng quan project:

* Tên ứng dụng: BlockPhone\_Backend
* IDE: Visual Studio Code
* version: 1.0.0
* Ngôn ngữ lập trình: JavaScript-NodeJS
* Cơ sở dữ liệu sử dụng: MySQL
* Ngôn ngữ ứng dụng: tiếng Anh
* Nền tảng đám mây: Heroku
* url\_host: https://block-spam.herokuapp.com/
* Git: https://github.com/namnvhustk61/BlockSpamBackend

Chức năng chính

Nhận phản hồi cảnh báo các số điện thoại rác/spam. Tiến hành lọc đánh giá số điện thoại theo các tiêu chí khác nhau. Gửi danh sách các số đã được xác nhận là rác/spam cho tất cả người dùng. Thông qua 2 API:

* getAllPhone
* postAddPhone

Cấu trúc Project

root

|- bin

|- model

|- PhoneBlock.model.js

|- UserReport.model.js

|- Result.js

|- routes

|- middleware

|- blockphone.controller.js

|- utils

|- hash

|- validate

|- app.js

|- package.json

Trong đó:

* bin: chứa file cấu hình server “www” cho việc deploy
* model: chứa các file khai báo các đối tượng sẽ được sử dụng
* routes: chứa các file định tuyến cho các kết nối API
* utils: chứa file công cụ: hash(mã hóa) validate(chuẩn hóa)
* app.js: file khởi chạy, mở kết nối của ứng dụng
* package.json: file quản lý khai báo các thư viện được sử dụng

### Triển khai cấu hình chung

Khai báo tên, version project và các thư viện sử dụng trong file package.json:

* name project: blockphone\_backend
* version: 1.0.0
* dependencies: body-parser, cookie-parser, crypto, express, mysql

Cấu hình port trong file “www”:

* port 3000: khi run local
* port process.env.PORT: khi deploy lên server online

Cấu hình kết nối tới MySQL trong file app.js:

* host: “us-cdbr-east-03.cleardb.com”
* user: “bd1afe430caaca”
* password: “\*\*\*\*\*”
* database: “heroku\_962c345ae8a3819”

### Khởi tạo các đối tượng

1. Form data response api chung của cả project

Dữ liệu API trả về từ server cho client dưới dạng kiểu định dạng JSON. Ở đây sẽ khai báo một cấu trúc chung cho tất cả API sau này để đạt được thông nhất cho phía client.

Result.js: cấu trúc chung của tất cả các response api

{

“code”: String,

“message”: String,

“data”: Object

}

* code: mã trạng thái: OK(thành công), E\_xxx(có lỗi xảy ra)
* message: lời nhắn phản hồi
* data: dữ liệu trả về

1. Đối tượng số điện thoại Block

PhoneBlock.Model.js: Đối tượng gồm các trường ứng với các thuộc tính của Bảng PhoneBlock trong cơ sở sữ liệu MySQL. Nó sẽ quản lý các thông tin về số điện thoại spam từ client gửi lên.

Các trường thông tin:

* name\_table: tên bảng tương ứng trên cơ sở dữ liệu MySQL
* idx: chỉ số
* phone: số điện thoại
* name: tên kèm theo
* type: phân loại
* status: trạng thái
* count: số user đã report số này

Các phương thức:

* toJSON: trả về data JSON cho API
* insertDB: tạo bản mới trên database
* getAllDB: lấy hết các đối tượng trong bảng blockphone
* updateIncresmentCount: tăng +1 cho count
* hasPhoneDB: kiểm tra số điện thoại này đã tồn tại trong bảng chưa

1. Đối tượng người dùng report

UserReport.model.js: Đối tượng gồm các trường ứng với các thuộc tính của Bảng UserReport trong cơ sở sữ liệu MySQL. Đối tượng này sẽ chứa id của user và số điện thoại đã report. Bảng UserReport sẽ lưu lại số user và số điện thoại đã báo cáo để tiến hành phân tích và đánh giá.

Các trường thông tin:

* name\_table:
* user\_id: mã định danh của người dùng
* phone: số điện thoại mà người đó báo cáo

Các phương thức:

* insertDB
* hasInDB

### Tạo API

1. Tạo middleware

Middleware là những code trung gian nằm giữa các request và response. Tiếp nhận các request, thi hành các kiểm tra định tuyến tương ứng trên request đó. Sau khi hoàn thành nó response hoặc chuyển kết quả cho một middleware khác trong hàng đợi.

Mid\_check\_content\_type.js: bộ lọc trung gian để kiểm tra content-type gửi lên từ request. Yêu cầu các request gửi data nên với “content-type” : “application/json” được cấu hình trong trường headers của gói tin.

Để khởi chạy middleware cần khai báo trong app.js với câu lệnh

var app = express();

app.use(mid\_check\_jwt\_token);

1. Tạo blockphone.controller.js

File controller.js sẽ là nơi để khai báo định tuyến cho các api hoạt động. Trong dữ án sẽ có 2 API được viết trong file blockphone.controller.js

Để khởi chạy file controller cần khai báo trong app.js với câu lệnh:

var app = express();

app.use('/blockphone', blockphoneRouter);

* Viết API getAllPhone

API này sẽ lấy toàn bộ các result trong bảng blockphone phân tích, xử lý để gửi lại về phía client.

Phương thức: GET

Path: /blockphone/all

Headers: content-type: application/json

Request: None

Response: {

“code”: “OK”,

“data”: [

{

“phone”:”0999999999”,

“name”:”Approved by admin”,

“type”:”spam”,

“status”: “Approved”,

“count”:10,

},

…

]

}

Câu lệnh lấy data trong MySQL:

var sql\_query = `SELECT \* FROM blockphone WHERE count >=10`;

Trà về tất cả các rows trong bảng blockphone có số người dùng report(count) lớn hơn hoặc bằng 10.

* Viết API postAddPhone

API này nhận dữ liệu gửi lên từ phía client tiến hành phân tích và thêm dữ liệu vào trong 2 bảng blockphone và UserReport để lưu trữ với 2 trường hợp.

TH1: Nếu số điện thoại chưa tồn tại sẽ tạo với đối tượng với count = 0

TH2: Nếu số điện thoại đã tồn tại sẽ update count = count +1

Phương thức: POST

Path: /blockphone/add

Headers: content-type: application/json

Request: {

“name”: “Spam”,

“phone”: “0999999999”,

“type”: “”,

“user\_id”: “eff99aa10ffo9”

}

Response: {

“code”: “OK”,

“message”: “Successfully”,

“data”:{

“note”: “The system has confirmed your action, this phone is also blocked by 0 other peoples.\n If number people greater than 10 then this block phone is public for every one”,

“count”: 1

}

}

Trong đó:

Name, phone, type, user\_id: yêu cầu not null

Phone validate: yêu cầu bắt đầu với “+84” hoặc “0” và lớn hơn 10 nhỏ hơn 15 số.

Câu lệnh lấy data trong MySQL:

var sql\_query = `Insert into blockphone (phone, name, type) values (?, ?, ?)`;

Câu lệnh sẽ thêm một hàng trong bảng blockphone

var sql\_query = `UPDATE blockphone SET count=count+1 WHERE phone =?`;

Câu lệnh sẽ update count của số điện thoại thêm 1

var sql\_query = `SELECT \* blockphone WHERE phone = ?`;

Câu lệnh kiểm tra số điện thoại đã tồn tại trong bảng hay chưa?

var sql\_query = `Insert into UserReport (user\_id, phone) values (?, ?)`;

Câu lệnh thêm một hàng trong bảng UserReport

var sql\_query = `SELECT \* UserReport WHERE user\_id = ? and phone=?`;

Câu lệnh kiểm tra số điện thoại và mã người dùng đã tồn tại trong bảng hay chưa?

1. Danh sách Status Code

Trong quá trình thực hiện API sẽ có các nhiều trường hợp phát sinh lỗi xảy ra. Vì vậy phía server sẽ cần kiểm tra và xử lý các lỗi đó rồi gửi phản hồi lại về phía client để tiến hành khắc phục.

OK: “OK”

* E\_001: “E\_001” // header content\_type != application/json
* E\_010: “E\_010” // field is require
* E\_015: “E\_015” // validate
* E\_500: “E\_500” // err connect server

# THỰC NGHIỆM VÀ KẾT LUẬN

## Phương pháp thực nghiệm

Ứng dụng được định nghĩ là ứng dụng điện thoại thông thường và chặn cuộc gọi xác định. Vì vậy cần phải đáp ứng các chức năng sau hoạt động bình thường:

* Thực hiện cuộc gọi
* Nhận cuộc gọi
* Chặn cuộc gọi
* Lấy danh sách danh bạ
* Lấy lịch sử cuộc gọi
* Kết nối tới server

Do Android là một hệ điều hành mã nguồn mở phổ biến do google và nhiều công ty điện thoại phát triển. Vì vậy hệ điều hành nhiều như khác nhau trên các phiên bản SDK khác nhau và trên các thiết bị android khác nhau.

Project này đã sử dụng các service khác nhau được cung cấp bởi google nhưng chỉ hoạt động trên các SDK phiển bản mới.

* InCallService: service gọi điện (tất cả phiên bản SDK)
* CallScreeningService: service chặn cuộc gọi (SDK >= 26)
* BroadcastReceiver: service khi cuộc gọi tới (tất cá phiên bản SDK)

Do đó ta sẽ tiến hành test trên các SDK và loại thiết bị khác nhau với các chức năng đã được liệt kê ở trên thông qua đó có thể đánh giá được sơ bộ về hệ thống.

### Test trên các SDK khác nhau

Trên các phiên bản SDK khác nhau sẽ có một số điểm khác biệt:

* Với SDK <23 ứng dụng chỉ cần khai báo permission trong file manifest, không cần hiện dialog xin quyền từ người dùng. Nhưng với SDK>=23 hiện dialog xin quyền đó là điều bắt buộc.
* Từ SDK>=26 có thêm khái niệm “Ứng dụng chặn cuộc gọi mặc định” và service CallScreeningService có thể hoạt động. Vì vậy với SDK <26 thì ứng dụng sẽ chặn cuộc gọi thông qua BroadcastReceiver.

Bảng . Test trên SDK khác nhau

|  |  |  |
| --- | --- | --- |
| **Chức năng** | **SDK >=26** | **SDK <26** |
| Gọi điện | Hoạt động | Hoạt động |
| Nhận cuộc gọi | Hoạt động | Hoạt động |
| Hiện thông báo | Hoạt động | Hoạt động |
| Chặn cuộc gọi | Hoạt động | Hoạt động |
| Danh bạ | Hoạt động | Hoạt động |
| Lịch sử cuộc gọi | Hoạt động | Hoạt động |
| Lấy danh sách từ server | Hoạt động | Hoạt động |
| Báo cáo lên server | Hoạt động | Hoạt động |

### Test trên các thiết bị khác nhau

Ngoài google vẫn hành năm nâng cấp các phiên bản android thuần thì mỗi hãng điện thoại đều dựa trên phiên bản đó để tùy biến lại để có các phiên bản cho riêng mình như VinSmart, Bphone, Samsung, Xiaomi,… do đó ta cần test trên rất nhiều thiết bị để thực nghiệm.

Bảng . Test trên nhiều thiết bị

|  |  |  |  |
| --- | --- | --- | --- |
| **Chức năng** | **Joy4(29)** | **Redmi note10 (30)** | **Redmi note3 (24)** |
| Gọi điện | Hoạt động | Hoạt động | Hoạt động |
| Nhận cuộc gọi | Hoạt động | Hoạt động | Hoạt động |
| Hiện thông báo | Hoạt động | Hoạt động | Hoạt động |
| Chặn cuộc gọi | Hoạt động | Hoạt động | Hoạt động |
| Danh bạ | Hoạt động | Hoạt động | Hoạt động |
| Lịch sử cuộc gọi | Hoạt động | Hoạt động | Hoạt động |
| Lấy danh sách từ server | Hoạt động | Hoạt động | Hoạt động |
| Báo cáo lên server | Hoạt động | Hoạt động | Hoạt động |

### Một số trường hợp không hoạt động

Các chức năng của ứng dụng để hoạt động được cần được cài đặt là ứng dụng mặc định hoặc cấp quyền từ người dùng vì vậy nếu người dùng đã tắt các quyền liên quan sẽ ảnh hướng đến hoạt dộng của nó.

1. Không thực hiện/nhận cuộc gọi mới

Do hệ thống chỉ có thể cấp quyền thực hiện/nhận cuộc gọi mới cho một ứng dụng duy nhất. Vì vậy để ứng dụng của chúng ta có thể làm được điều đó thì bắt buộc phải đươc người dùng đặt là ứng dụng gọi điện thoại mặc định.

Nếu không khi người dùng bắt đầu gọi điện thì hệ thống sẽ tự đọng chuyển hướng sang giao diện của ứng dụng mặc định của thiết bị.

Cách khắc phục:

C1. Thiệt lập trực tiếp tại ứng dụng

Khi người dùng vào ứng dụng sẽ có dialog thiết lập ứng dụng mặc định được hiện lên để người dùng chọn.

A screenshot of a cell phone

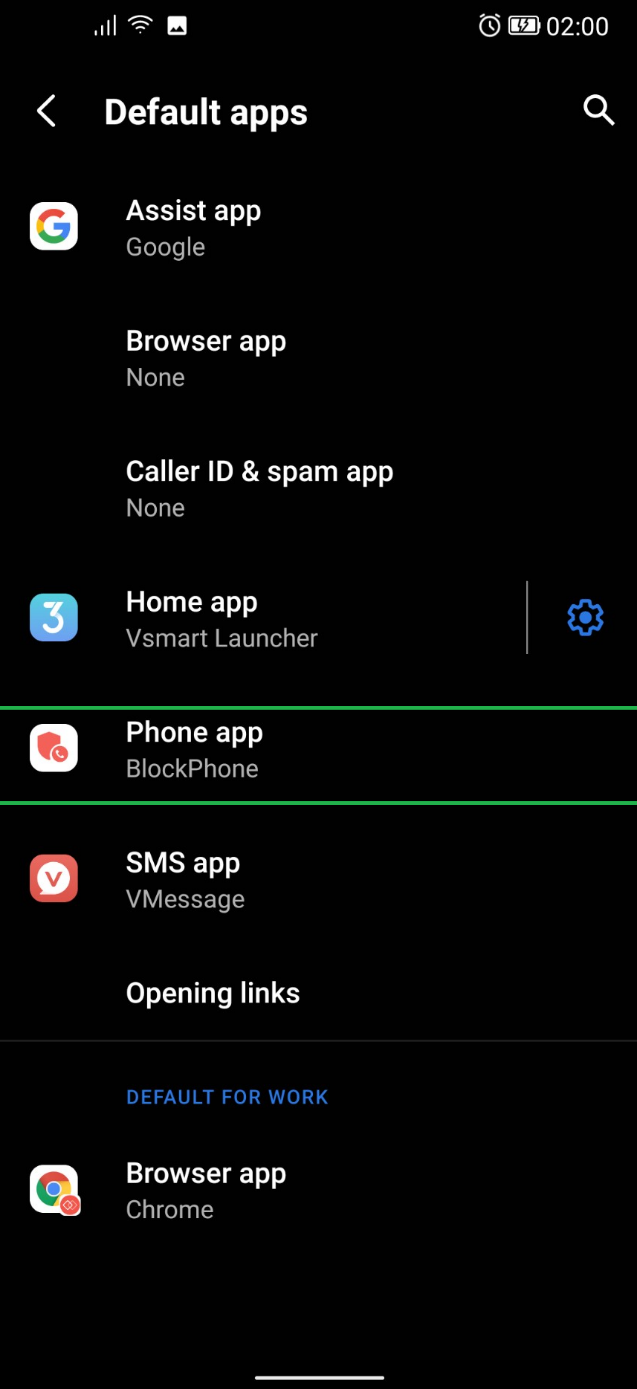
Description automatically generated with medium confidence

Hình . Dialog thiết lập ứng dụng điện thoại mặc định

C2. Thiết lập thủ công tại “Cài đặt” của thiết bị

Nếu người đã tắt và chọn mục “Không hỏi lại” tại Dialog như trên thì ứng dụng sẽ không thể hiện Dialog đó thêm một lần nữa. Vì vậy để thiết lập người dùng cần vào “Cài đặt” của thiết bị để thiết lập thủ công.

Vào “Cài dặt” 🡪 tìm mục “Ứng dụng” 🡪 bấm “Ứng dụng mặc định” 🡪 bấm “Điện thoại” 🡪 chọn ứng dụng bạn muốn thiết lập làm ứng dụng điện hoại mặc định.



Hình . Mục “Ứng dụng mặc định” trên thiết bị Joy4

1. Không Hoạt động do thiếu quyền

Với các thiết bị có phiên bản hệ điều hành SDK>=23 để thực thiện một số chức năng đặc biệt thì ứng dụng cần phải được cấp quyền trực tiệp bởi người dùng.

* Quyền Phone (Điện thoại)

Ứng dụng sẽ không thể thực hiện gọi điện thoại.

* quyền Call logs (Nhật ký cuộc gọi)

Ứng dụng không thể đọc được số điện thoại để chặn cuộc gọi (BlockCallBroadcastReceiver không hoạt động), không thể lấy danh sách lịch sử cuộc gọi.

* quyền Contacts (Danh bạ)

Ứng dụng không thể chặn được cuộc gọi (CallScreeningService không hoạt động), không lấy danh bạ cuộc gọi.

Cách khắc phục:

C1. Thiệt lập trực tiếp tại ứng dụng

Khi người dùng vào ứng dụng tại các màn hình khác nhau sẽ có các giao diện và chức năng khác nhau. Với mỗi chức năng khác nhau ứng dụng sẽ hiện lên dialog cấp quyền tương ứng để người dùng có thể các nhận cho phép. Bằng cách bấm vào “Cho phép” người dùng đã cấp quyền cho ứng dụng thực hiện được các chức năng tương ứng.

A screenshot of a computer

Description automatically generated with medium confidence

Hình . Dialog cấp quyền Call logs (Nhật ký cuộc gọi)

C2. Thiết lập thủ công tại “Cài đặt” của thiết bị

Nếu người đã tắt và chọn mục “Không hỏi lại” tại Dialog như trên thì ứng dụng sẽ không thể hiện Dialog đó thêm một lần nữa. Vì vậy để thiết lập người dùng cần vào “Cài đặt” của thiết bị để thiết lập thủ công.

Vào “Cài dặt” 🡪 tìm mục “Ứng dụng” 🡪 tìm tới ứng dụng BlockPhone 🡪 Tại màn hình “thông tin ứng dụng” chọn “Quyền”🡪 bấm vào quyền tương ứng 🡪 chọn “Cho phép”.

Hoặc nhấn liền vào biểu tượng của BlockPhone 🡪 chọn “Thông tin” 🡪 chọn “Quyền” 🡪 bấm vào quyền tương ứng 🡪 chọn “Cho phép”.

## Đánh giá và kết luận

### Đánh giá

Với các chức năng cơ bản đã được xây dựng ở trên thì người dùng hoàn toàn có thể dùng nó để thay thế cho ứng dụng điện thoại đang sử dụng.

Người dùng có thể sử dụng tính năng chặn cuộc gọi local để đáp ứng ngay nhu cầu chặn các số rác/làm phiền và báo cáo lại cho hệ thống online.

Hạn chế, hiện ứng dụng vẫn đang trong quá trính phát triển và hoàn thiệt vì vậy có thể phát sinh các lỗi trên một số loại thiết bị. Số người dùng còn it nên tính năng phát hiện và cảnh báo người dùng online vẫn chưa thể đạt được hiệu quả cao nhất.

### Kết luận

Trong đồ án tốt nghiệp này, em đã trình bầy tất cả các nội dụng liên quan đến đề xuất và thực hiện giải pháp chặn và phát hiện các số điện thoại rác/gây phiền phức cho người dùng. Cung cấp một ứng dụng ngay lập tức có thể đáp ứng được nhu cầu chắn số rác cho người dùng. Hi vọng sau một thời gian hoạt động với số lượng người dùng lớn hệ thống sẽ đánh giá và phát hiện các số điện thoại rác mốt cách nhanh chóng, hiệu quả và chính xác, góp phần giải quyết vấn nạn sim số rác một cách triệt để.

Trong quá trình thực hiện và triển khải hệ thống còn nhiều thiếu sót và nhiều vấn đề chưa được đề cấp tới. Em rất mong nhận được đánh giá, góp ý từ thầy cô để em có thể hoàn thành đồ án và đề tài này hiệu quả và hoàn thiên hơn.

### Định hướng phát triển

Để hệ thống có thể hoàn thiện và đạt hiệu quả tốt nhất, sẽ cần bổ sung thêm một số thành phần và chi tiết:

+ Nâng cao chất lượng giao diện và tháo tác cho ứng dụng

+ Phát triển hệ thống phân tích, đánh giá chuyên nghiệp hơn trên server

+ Nghiên cứu xây dựng app trên hệ điều hành ios

+ Đưa ứng dụng lên các kho ứng dụng: chplay google, appstore ios, …

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Implementing Block Phone Numbers  https://source.android.com/devices/tech/connect/block-numbers. |
| [2] | Build a calling app  https://developer.android.com/guide/topics/connectivity/telecom/selfManaged |
| [3] | CallScreeningService  https://developer.android.com/reference/android/telecom/CallScreeningService |
| [4] | InCallService  https://developer.android.com/reference/android/telecom/InCallService |
| [5] | Save data in a local database using Room  https://developer.android.com/training/data-storage/room |
| [6] | Simple Dialer  https://github.com/SimpleMobileTools/Simple-Dialer |
| [7] | Getting Started guide NodeJS  https://nodejs.org/en/docs/guides/getting-started-guide |
| [8] | Khóa học nodejs  https://www.youtube.com/watch?v=OR0hBEUk4wI&list=PLqnlyu33Xy-6g7IqU5-3BXOfewcJKoL08 |
| [9] | RESTful API  https://topdev.vn/blog/restful-api-la-gi/ |
|  |  |