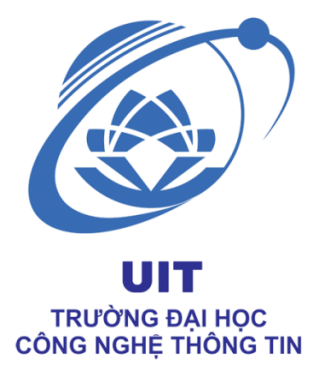
**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**



**GVHD: Nguyễn Trác Thức**

**SVTH: 12520077 – Hà Huy Đức**

**13520446 – Mai Văn Linh**

**ĐỀ TÀI: TÌM HIỂU VỀ RESTAPI WEBSERVICE**

**XÂY DỰNG HỆ THỐNG CẢNH BÁO ÙN TẮC GIAO THÔNG**

**ĐỒ ÁN CHUYÊN NGÀNH**

*TP. Hồ Chí Minh, 10 tháng 06 năm 2017*

**BÁO CÁO**

**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, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập tại trường đại học đến nay, chúng em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý Thầy Cô, gia đình và bạn bè.

Với lòng biết ơn sâu sắc nhất, chúng em xin gửi đến quý Thầy Cô ở Khoa Công nghệ Phần mềm – Trường Đại Học Công Nghệ Thông Tin đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Và đặc biệt, trong học kỳ này, Khoa đã tổ chức cho chúng em được tiếp cận với môn học mà theo chúng em là rất hữu ích đối với sinh viên ngành Công nghệ Phần mềm. Đó là môn học “Đồ án chuyên ngành”

Chúng em xin chân thành cảm ơn thầy Nguyễn Trác Thức đã tận tâm hướng dẫn chúng em thông qua những buổi giảng dạy, nói chuyện và thảo luận. Nếu không có những lời hướng dẫn, dạy bảo của thầy thì chúng em nghĩ bài thu hoạch này sẽ rất khó có thể hoàn thiện được. Một lần nữa, chúng em xin chân thành cảm ơn thầy.

Sau cùng, chúng em xin kính chúc quý Thầy Cô trong Khoa Công nghệ Phần mềm và thầy Nguyễn Trác Thức thật dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau.

Trân trọng.

TP.HCM, ngày 10 tháng 6 năm 2017

**Sinh viên thực hiện**

**12520077 Hà Huy Đức**

**13520446 Mai Văn Linh**

**NHẬN XÉT CỦA GIẢNG VIÊN**

**MỤC LỤC**

[**I.** **LÝ DO CHỌN ĐỀ TÀI** 4](#_Toc486323911)

[**II.** **TỔNG QUAN VỀ RESTful API** 5](#_Toc486323912)

[**1.** **Giới thiệu** 5](#_Toc486323913)

[**2.** **Cấu trúc REST** 5](#_Toc486323914)

[**3**. **Cơ chế hoạt động** 6](#_Toc486323915)

[**4. Ứng dụng của Rest** 7](#_Toc486323916)

[III. TỔNG QUAN VỀ RESTful WEB SERVICE 8](#_Toc486323917)

[**1.** **Phân biệt giữa Web và Web Service** 8](#_Toc486323918)

[**2.** **RESTful Service là gì?** 8](#_Toc486323919)

[IV. GIỚI THIỆU VỀ RETROFIT 13](#_Toc486323920)

[**1.** **Giới thiệu** 13](#_Toc486323921)

[**2.** **Tại sao sử dụng Retrofit** 13](#_Toc486323922)

[**3.** **Cách sử dụng Retrofit trong ứng dụng Android** 13](#_Toc486323923)

[**V.** **HỆ DIỀU HANH ANDROID** 14](#_Toc486323924)

[**1.** **Giới thiệu hệ điều hành Android** 14](#_Toc486323925)

[**2.** **Những đặc trưng của hệ điều hành Android:** 14](#_Toc486323926)

[**3.** **Các tính năng hỗ trợ sẵn trong hệ điều hành Android** 15](#_Toc486323927)

[**4.** **Kiến trúc và các thành phần trong hệ điều hành Android** 16](#_Toc486323928)

[**5.** **Các khái niệm cơ bản trong lập trình ứng dụng Android** 18](#_Toc486323929)

[**6.** **Các thành phần trong một project ứng dụng Android** 23](#_Toc486323930)

[**VI.** **GOOGLE MAP VÀ GOOGLE MAPS API WEB** 24](#_Toc486323931)

[**1.** **Google Map** 24](#_Toc486323932)

[**2.** **Google Maps API Web Services** 27](#_Toc486323933)

[**3.** **Google Geocoding API** 29](#_Toc486323934)

[**4.** **Google Distance Matrix API** 40](#_Toc486323935)

[**5.** **Google Direction API** 47](#_Toc486323936)

[**VII.** **XÂY DỰNG ỨNG DỤNG** 61](#_Toc486323937)

[**1.** **Giớ thiệu đề tài** 61](#_Toc486323938)

[**2.** **Đặc tả use case** 61](#_Toc486323939)

[**a.** **Tìm lộ trình đến địa điểm cụ thể** 62](#_Toc486323940)

[**b.** **Cung cấp thông tin giao thông theo thời gian thực** 63](#_Toc486323941)

[**c.** **Gửi thông tin phương tiện cho server** 64](#_Toc486323942)

[**d.** **Server tổng hợp dữ liệu người dùng** 65](#_Toc486323943)

[**3.** **Mô hình hoạt động** 65](#_Toc486323944)

[**a.** **Quá trình lưu trữ và xử lý dữ liệu** 65](#_Toc486323945)

[**b.** **Quá trình gửi dữ liệu lên server** 66](#_Toc486323946)

[**c.** **Quá trình nhận dữ liệu từ server** 66](#_Toc486323947)

[**4.** **Thiết kế màn hình ứng dụng** 67](#_Toc486323948)

[**VIII.** **TỔNG KẾT** 69](#_Toc486323949)

[**1.** **Đánh giá hệ thống đã xây dựng** 69](#_Toc486323950)

[**2.** **Hướng phát triển** 69](#_Toc486323951)

[**3.** **Kết quả đạt được** 70](#_Toc486323952)

[**4.** **Tài liệu tham khảo** 70](#_Toc486323953)

1. **LÝ DO CHỌN ĐỀ TÀI**

REST định nghĩa các quy tắc kiến trúc để bạn thiết kế Web services chú trọng vào tài nguyên hệ thống, bao gồm các trạng thái tài nguyên được định dạng như thế nào và được chuyển tải qua HTTP thông qua số lượng lớn người dùng và được viết bởi những ngôn ngữ khác nhau. Nếu tính theo số dịch vụ mạng sử dụng, REST đã nổi lên trong vài năm qua như là một mô hình thiết kế dịch vụ chiếm ưu thế. Trong thực tế, REST đã có những ảnh hưởng lớn và gần như thay thế SOAP và WSDL vì nó đơn giản và dễ sử dụng hơn rất nhiều. REST không thu hút được nhiều sự chú ý khi lần đầu tiên giới thiệu vào năm 2000 bởi Roy Fielding trong luận án của ông "Architectural Styles and the Design of Network-based Software Architectures" (Phong cách kiến trúc và thiết kế kiến trúc phần mềm dựa trên mạng) tại Đại học California. Luận án đã phân tích một loạt các nguyên tắc kiến trúc phần mềm sử dụng Web như là một nền tảng tính toán phân tán (xem Tài nguyên liên kết tới luận án). Đến nay, vài năm sau đó, đã xuất hiện các framework chủ đạo cho REST và chúng vẫn đang được tiếp tục phát triển, nó đang được xem xét để đưa vào trong bộ Java™ 6 thông qua tiêu chuẩn JSR-311.

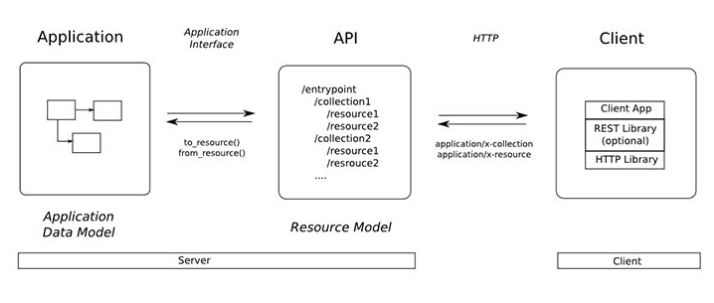
REST (Representational State Transfer) đã được chọn sử dụng rộng rãi thay cho Web service dựa trên SOAP và WSDL. Bằng chứng quan trọng của sự thay đổi này chính là việc các công ty dẫn đầu trong lĩnh vực cung cấp dịch vụ mạng 2.0 như Yahoo, Google và Facebook đã phản đối các giao thức dựa trên SOAP hoặc WSDL và ủng hộ phương thức hướng đến tài nguyên và dễ sử dụng đối với các dịch vụ của họ.

1. **TỔNG QUAN VỀ RESTful API**
2. **Giới thiệu**

* Rest là một loạt hướng dẫn và dạng cấu trúc dùng cho việc chuyển đổi dữ liệu. thông thường, rest hay được dùng cho ứng dụng web, nhưng cũng có thể làm việc được với dữ liệu phần mềm.
* API là viết tắt của application programming interface, phương thức kết nối với các thư viện và ứng dụng khác. windows có nhiều api, và twitter cũng có web API, tuy chúng thực hiện các chức năng khác nhau, với mục tiêu khác nhau.

Nhìn chung, restful API là những api đi theo cấu trúc rest.

1. **Cấu trúc REST**



Một số quy luật bất biếncủa Rest API, như:

* **Sự nhất quán** trong cả API
* Tồn tại không trạng thái (ví dụ, không có server-side session)
* Sử dụng **HTTP status code** khi cần thiết
* Sử dụng **URL endpoint** với logical hierarchy
* **Versioning trong URL** chứ không phải trong HTTP header

REST là một phương thức nhỏ gọn. Nên rất được ưa chuộng cho dữ liệu HTTP. Cũng vì vậy nên REST dần phổ biến trên web, và được xem là lựa chọn “số một” cho phát triển API.

## **3**. **Cơ chế hoạt động**

Người dùng API là lập trình viên web có thể build một script kết nối đến một external API server, rồi dữ liệu cần thiết sẽ chuyển sang HTTP. Lập trình viên khi đó có thể hiển thị dữ liệu lên website mà **không cần đến truy cập cá nhân vào external server**.

Nhìn chung, có **bốn lệnh** dùng để **truy cập RESTful API**:

* GET để truy vấn object
* POST để tạo object mới
* PUT để sửa đổi hoặc thay thế một object
* DELETE để loại bỏ một object

Mỗi phương thức trên phải **được API call thông qua**để gửi chỉ thị cho server phải làm gì.

Đại đa số web API **chỉ cho phép GET request** lấy dữ liệu khỏi một externer server. Authencation không bắt buộc, nhưng nên có khi ta cho phép các lệnh khá “nguy hiểm” như PUT hay DELETE.

* Có hai loại REST API:
* Public API:
* [Reddit API](https://www.reddit.com/dev/api)
* [GitHub API](https://developer.github.com/v3/)
* [Flickr API](https://www.flickr.com/services/api/)
* [Pinterest API](https://developers.pinterest.com/docs/getting-started/introduction/)
* API tự build
* Quá trình xây dựng một API riêng không dễ dàng, nhưng cũng không quá phức tạp như nhiều người nghĩ.

Mỗi API phải**kết nối đến server** để trả dữ liệu. Bạn không những phải viết code để làm điều đó, mà còn phải format return data nữa. Một số yêu cầu khác có thể gồm **authentication**và **rate limiting.**

## **4. Ứng dụng của Rest**

- Mục đích của REST là thiết kế các ứng dụng mạng phân tán sử dụng HTTP như là một giao thức tầng ứng dụng và nó là một mô hình kiến trúc thực sự cho web.

- Sự phát triển ngày càng lớn của các dịch vụ web dẫn tới hệ quả tất yếu là RESTful đã được đưa ra như là một giải pháp để thay thế việc thực hiện triệu gọi từ xa (RPC) thông qua web.

- REST là một kiểu kiến trúc cho hệ thống phân tán như World Wide Web.

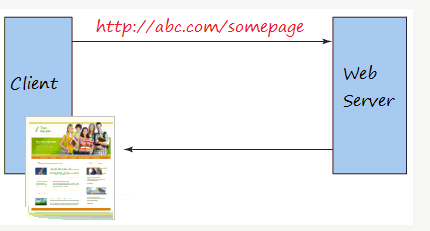
- REST được sử dụng rất nhiều trong việc phát triển các ứng dụng Web sử dụng giao thức HTTP trong giao tiếp thông qua mạng internet. Các ứng dụng sử dụng kiến trúc REST này thì sẽ được gọi là ứng dụng phát triển theo kiểu RESTful.

- REST tập trung vào việc xử lý các tài nguyên. Tài nguyên là mọi thứ như khách hàng, video, tranh ảnh, trang web… Tài nguyên có thể là một đối tượng vật lý cũng có thể là một khái niệm trừu tượng nào đó. Các tài nguyên này giúp ta định nghĩa được các dịch vụ trong hệ thống, kiểu thông tin mà nó trả về, và hành vi xử lý thông tin của nó. Mỗi tài nguyên đều được định danh bởi một ID duy nhất là URI. Nếu một thông tin nào đó không có URI thì nó không phải là tài nguyên và không tồn tại trên mạng.

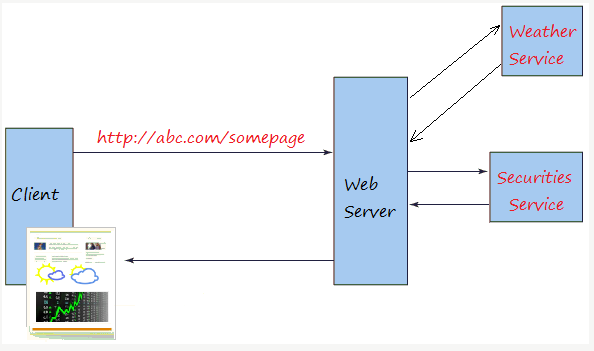
-Dịch vụ kiểu REST cho phép các máy khách chuyển từ trạng thái này đến trạng thái khác bằng cách gửi các liên kết trong các đại diện với nhau, đại diện có thể là siêu âm thanh, có thể là tài liệu mà trong đó không chỉ chứa mỗi dữ liệu mà có thể còn có.

# TỔNG QUAN VỀ RESTful WEB SERVICE

## **Phân biệt giữa Web và Web Service**

Khi bạn gõ vào một URL vào trình duyệt và bạn nhận được một trang web. Đây là một nội dung mà thông thường bạn có thể đọc được, nó là nội dung dành cho người dùng ở đầu cuối.

**Web Service** là một dịch vụ web, nó là một khái niệm rộng hơn so với khái niệm web thông thường, nó cung cấp các thông tin thô, và khó hiểu với đa số người dùng, chính vì vậy nó được sử dụng bởi các ứng dụng. Các ứng dụng này sẽ chế biến các dữ liệu thô trước khi trả về cho người dùng cuối cùng.



## **RESTful Service là gì?**

REST ( REpresentational State Transfer) được đưa ra vào năm 2000 trong luận văn tiến sĩ của Roy Thomas Fielding (đồng sáng lập giao thức HTTP). Trong luận văn ông giới thiệu khá chi tiết về các ràng buộc, quy ước cũng như cách thức thực hiện với hệ thống để có được một hệ thống REST.

REST định nghĩa các quy tắc kiến trúc để bạn thiết kế Web services, chú trọng vào tài nguyên hệ thống, bao gồm các trạng thái tài nguyên được định dạng như thế nào và được truyền tải qua HTTP, và được viết bởi nhiều ngôn ngữ khác nhau. Nếu tính theo số dịch vụ mạng sử dụng, REST đã nổi lên trong vài năm qua như là một mô hình thiết kế dịch vụ chiếm ưu thế. Trong thực tế, REST đã có những ảnh hưởng lớn và gần như thay thế SOAP và WSDL vì nó đơn giản và dễ sử dụng hơn rất nhiều.

REST là một bộ quy tắc để tạo ra một ứng dụng Web Service, mà nó tuân thủ 4 nguyên tắc thiết kế cơ bản sau:

* Sử dụng các phương thức HTTP một cách rõ ràng
* Phi trạng thái
* Hiển thị cấu trúc thư mục như các URls
* Truyền tải JavaScript Object Notation (JSON), XML hoặc cả hai.
* Sử dụng các phương thức HTTP một cách rõ ràng

**REST** đặt ra một quy tắc đòi hỏi lập trình viên xác định rõ ý định của mình thông qua các phương thức của **HTTP**. Thông thường ý định đó bao gồm lấy dữ liệu, trèn dữ liệu, cập nhập dữ liệu hoặc xóa dữ liệu. Khi muốn thực hiện một trong các ý định trên hãy lưu ý các quy tắc sau:

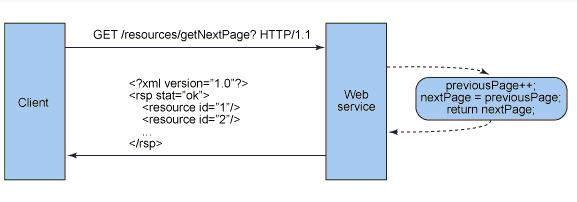
* Để tạo một tài nguyên trên máy chủ, bạn cần sử dụng phương thức POST.
* Để truy xuất một tài nguyên, sử dụng GET.
* Để thay đổi trạng thái một tài nguyên hoặc để cập nhật nó, sử dụng PUT.
* Để huỷ bỏ hoặc xoá một tài nguyên, sử dụng DELETE.

Chú ý rằng các nguyên tắc ở trên là không bắt buộc, thực tế bạn có thể sử dụng phương thức GET để yêu cầu lấy dữ liệu, chèn, sửa hoặc xóa dữ liệu trên Server. Tuy nhiên REST đưa ra các nguyên tắc ở trên mục đích đưa mọi thứ trở nên rõ ràng và dễ hiểu.

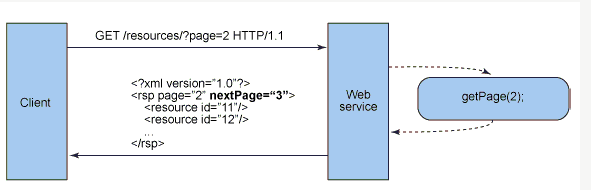
* Phi trạng thái

Một đặc điểm của **REST** là phi trạng thái (stateless), có nghĩa là nó không lưu giữ thông tin của client. Chẳng hạn ta vừa gửi yêu cầu để xem trang thứ 2 của một tài liệu, và bây giờ muốn xem trang tiếp theo (sẽ là trang 3). **REST** không lưu trữ lại thông tin rằng trước đó nó đã phục vụtrang số 2. Điều đó có nghĩa là **REST** không quản lý phiên làm việc (Session).

Hình dưới đây minh họa một ứng dụng có lưu trữ trạng thái, nó biết người dùng đang xem trang số mấy. Và người dùng chỉ cần yêu cầu "Trang Tiếp theo" để nhận được trang mong muốn.



Với các thiết kế phi trạng thái, Client phải gửi yêu cầu rõ ràng, bao gồm số thự tự của trang cần xem.



Như vậy, các thành phần máy chủ phi trạng thái ít phức tạp hơn để thiết kế, viết và phân bổ thông qua máy chủ được cân bằng tải. Dịch vụ phi trạng thái không chỉ hoạt động tốt hơn, nó còn chuyển hầu hết vai trò duy trì trạng thái sang ứng dụng ở máy khách. Trong một dịch vụ mạng**RESTful**, máy chủ chịu trách nhiệm đưa ra các phản hồi và cung cấp một cách thức cho phép Client duy trì trạng thái ứng dụng của chính nó.

* Hiển thị cấu trúc thư mục như các Urls
* REST đưa ra một cấu trúc để người dùng có thể truy cập vào tài nguyên của nó thông qua các URL, tài nguyên ở đây là tất cả những cái mà bạn có thể gọi tên được (Video, ảnh, báo cáo thời tiết,..)
* Bạn cần tạo ra các REST serivce để nó trả về cho người dùng các nguồn tài nguyên tương ứng.
* Các địa chỉ REST service cần phải thật trực quan đến mức người dùng dễ đoán. Hãy nghĩ về một địa chỉ (URI) giống như một gợi ý rõ ràng, dễ đoán rằng nó đang trỏ tới cái gì và cung cấp tài nguyên gì. Tóm lại, cấu trúc của một URI nên được đơn giản, có thể dự đoán, và dễ hiểu.
* Một vài nguyên tắc bổ sung để lưu ý trong khi nói về cấu trúc địa chỉ của Web service RESTful là:
* Giấu các đuôi tài liệu mở rộng của bản gốc trong máy chủ (.jsp, .php, .asp), nếu có, vì vậy bạn có thể giấu một số thứ mà không cần thay đổi địa chỉ Urls.
* Để mọi thứ là chữ thường.
* Thay thế các khoảng trống bằng gạch chân hoặc hoặc gạch nối (một trong hai loại).
* Tránh các chuỗi yêu cầu càng nhiều càng tốt.
* Thay vì sử dụng mã (404 Not Found) khi yêu cầu địa chỉ cho một phần đường dẫn, luôn luôn cung cấp một trang mặc định hoặc tài nguyên như một phản hồi
* Truyền tải JavaScript Object Notation (JSON), XML hoặc cả hai

Khi Client gửi một yêu cầu tới web service nó thường được truyền tải dưới dạng XML hoặc JSON và thông thường nhận về với hình thức tương tự.

Đôi khi Client cũng có thể chỉ định kiểu dữ liệu nhận về mà nó mong muốn (JSON, hoặc XML,..), các chỉ định này được gọi là các kiểu MINE, nó được gửi kèm trên phần HEADER của request.

Dưới đây là các kiểu MINE phổ biến thường sử dụng với REST service.

|  |  |
| --- | --- |
| **MIME-Type** | **Content-Type** |
| **JSON** | application/json |
| **XML** | application/xml |
| **XHTML** | application/xhtml+xml |

# GIỚI THIỆU VỀ RETROFIT

## **Giới thiệu**

Retrofit là một type-safe HTTP client cho Android và Java. Retrofit giúp dễ dàng kết nối đến một dịch vụ REST ở trên web bằng cách dịch API thành các Interface của Java. Trong hướng dẫn này, tôi sẽ chỉ cho bạn cách sử dụng một trong các thư viện HTTP phổ biến nhất và thường được khuyến khích sử dụng cho Android.

Thư viện mạnh mẽ này giúp bạn dễ dàng lấy dữ liệu JSON hoặc XML. Các yêu cầu GET, POST, PUT, PATCH, DELETE đều có thể thực thi.

Giống như hầu hết các thư viện mã nguồn mở khác, Retrofit được xây dựng trên nền tảng của một số thư viện mạnh mẽ và công cụ khác. Đằng sau nó, Retrofit sử dụng OkHttp (từ cùng 1 nhà phát triển) để xử lý các yêu cầu mạng. Retrofit không tích hợp sẵn bộ chuyển đổi JSON để phân tích đối tượng JSON thành JAVA, thay vào đó nó hỗ trợ cho các thư viện chuyển đổi JSON để xử lý vấn đề đó.

## **Tại sao sử dụng Retrofit**

Phát triển thư viện type-safe HTTP của riêng của bạn để giao tiếp với một REST API có thể thật sự rất khó: bạn phải xử lý nhiều khía cạnh, chẳng hạn như kết nối, bộ nhớ đệm, thử lại yêu cầu sai, luồng, phân tích phản hồi, xử lý lỗi và nhiều thứ khác. Mặt khác, Retrofit là một thư viện được tổ chức tốt, tài liệu hướng đầy đủ và đã thử nghiệm sẽ giúp bạn tiết kiệm rất nhiều thời gian quý báu và những đau đầu không cần thiết.

## **Cách sử dụng Retrofit trong ứng dụng Android**

Thêm thư viện Retrofit và GSON để chuyển đổi các đối tượng JSON:

**//retrofit**

**complie ‘com.squareup.retrofit2:retrofit:2.1.0’**

**//json parsing**

**complie ‘com.google.code.gson:gson:2.6.1’**

**complie ‘com.squareup.retrofit2:converter-gson:2.1.0’**

Thêm quyền truy cập internet trong manifest:

<**uses-permission**android:name="android.permission.INTERNET" />

1. **HỆ DIỀU HANH ANDROID**

## **Giới thiệu hệ điều hành Android**

Android là một hệ điều hành di động dựa trên một phiên bản sửa đổi của Linux. Được phát triển vào năm 2005 với một dự án cùng tên “Android”. Như một phần chiến lược của mình để lấn sâu vào lĩnh vực di động Google Android đã mua về toàn bộ quá trình phát triển cũng như đội phát triển nó. Đây là con át chủ bài của Google để cạnh tranh thị phần hệ điều hành di động với Apple.

Google Android muốn mở và miễn phí, vì vậy hầu hết các mã android được được đưa ra dưới dạng mã nguồn mở Apache License, điều này tương đương với việc bất cứ ai muốn sử dụng Android có thể làm như vậy bằng cách tải về mã nguồn android đầy đủ. Hơn nữa các nhà cung cấp (thường là những nhà phát triển phần cứng) có thể thêm phần mở rộng và tùy biến cho android để phân biệt sản phẩm của họ với sản phẩm của những người khác. Điều này đơn giản làm cho mô hình phát triển android rất hấp dẫn và do đó khơi dậy sự quan tâm của nhiều nhà cung cấp. Những nhà sản xuất coi android như một giải pháp – họ sẽ tiếp tục thiết kế phần cứng của riêng mình và sử dụng android như một hệ điều hành chính.

Ưu điểm chính của việc áp dụng android là nó cung cấp một cách tiếp cận thống nhất để phát triển ứng dụng. Các nhà phát triển chỉ cần phát triển cho android và các ứng dụng của họ có thể chạy trên nhiều thiết bị khác nhau, miễn là các thiết bị được hỗ trợ bằng cách sử dụng android. Trong thế giới điện thoại thông minh ứng dụng là một phần quan trọng nhất của chuỗi thành công. Do đó các nhà sản xuất thiết bị coi Android như là hy vọng tốt nhất để thách thức sự tấn công của Apple.

## **Những đặc trưng của hệ điều hành Android:**

* **Application framework:** cho phép sử dụng lại và thay thế các thành phần trong lập trình ứng dụng.
* **Dalvik virtual machine:** tối ưu hóa cho thiết bị di động.
* **Intergrated browser:** trình duyệt tích hợp, dựa trên cơ chế WebKit mã nguồn mở.
* **SQLite:** sơ sở dữ liệu trong môi trường di động.
* **Media support:** hỗ trợ các định dạng audio, video và hình ảnh thông dụng.
* **GSM Telephony:** mạng điện thoại di động (phụ thuộc vào phần cứng).
* **Bluetooth, EDGE, 3G, và WiFi :**các chuẩn kết nối dữ liệu (phụ thuộc vào phần cứng).
* **Camera, GPS, la bàn, và gia tốc kế**: (phụ thuộc vào phần cứng).
* **Môi trường phát triển phong phú**: bao gồm thiết bị mô phỏng, công cụ cho việc dò tìm lỗi, bộ nhớ và định hình hiệu năng và một plugin cho Eclipse IDE.

## **Các tính năng hỗ trợ sẵn trong hệ điều hành Android**

Các tính năng được hổ trỡ tùy thuộc vào cấu hình phần cứng và phần mềm.

* **Storage**: Sử dụng SQLite, một cơ sở dữ liệu quan hệ nhẹ cho việc lưu trữ dữ liệu
* **Connectivity**: Hỗ trợ GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (bao gồm AD2P và AVRCP), Wifi, LTE và WiMAX
* **Messaging**: hỗ trợ cả SMS và MMS
* **Web browser**: Dựa trên mã nguồn mở Webkit, cùng với công nghệ

JavaScript V8 của Chrome

* **Media support**: Bao gồm hỗ trợ các phương tiện truyền thông sau: H.263, H.264 (Trong 3GP hoặc MP4 container ), MPEG-4 SP, AMR, AMR-WB (3GP container), AAC, HE-AAC (MP4 hoặc 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF và BMP
* **Hardware support**: Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor và GPS
* **Multi-touch**: Hỗ trợ màn hình cảm ứng đa điểm
* **Multi-tasking**: Hỗ trợ ứng dụng đa tác vụ
* **Flash support**: Android 2.3 hỗ trợ Flash 10.1
* **Tethering**: Hỗ trợ kết nối internet không dây/có dây

## **Kiến trúc và các thành phần trong hệ điều hành Android**

Mô hình sau thể hiện đầy đủ kiến trúc các thành phần của hệ điều hành Android.



**Linux kernel (nhân Linux) :**

Kernel Linux hoạt động như một lớp trừu tượng hóa giữa phần cứng và tầng dưới của phần mềm. Lớp này chứa tất cả các thiết bị mức thấp điều khiển các thành phần phần cúng khác nhau của một thiết bị Android.

**Libraries :**

Libraries bao gồm một tập hợp các thư viện lập trình chứa mã lệnh cung cấp những tính năng và thao tác chính trên hệ điều hành. Một số các thư viện cơ bản được liệt kê dưới đây:

* **System C library:** a BSD-derived triển khai các thư viện hệ thống ngôn ngữ C chuẩn, được nhúng vào các thiết bị dựa trên hệ điều hành Linux.
* **Media Libraries** – Dựa trên PacketVideo's OpenCORE; thư viện này hỗ trợ cho việc chơi nhạc, quay phim, chụp hình theo các định dạng file MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
* **Surface Manager** – Quản lý truy cập đến các hệ thống con hiển thị cũng như các lớp đồ họa 2D, 3D từ tầng ứng dụng.
* **LibWebCore** – Thư viện được dùng để tạo nên thành phần webview trong Android và có thể nhúng được vào nhiều ứng dụng.
* **SGL** – Thư viện hỗ trợ đồ hoạ 2D.
* **3D libraries** – Thư viện đồ họa 3D.
* **FreeType** - bitmap and vector font rendering.
* **SQLite** – Một cơ sở dữ liệu nhỏ được dùng cho các thiết bị cầm tay có bộ nhớ hạn chế. SQLite không có quan hệ như các cơ sở dữ liệu khác.

**Android runtime:**

Tại cùng một tầng với Libraries, android runtime cung cấp một bộ lõi thư viện cho phép các nhà phát triển viết các ứng dụng android bằng cách sử dụng ngôn ngữ lập trình java. Android runtime cũng bao gồm các máy ảo Dalvik, cho phép mọi ứng dụng android chạy trong tiến trình riêng của mình. Dalvik là một máy ảo chuyên dụng được thiết kế đặc biệt cho android và tối ưu hóa cho các thiết bị điện thoại di động với giới hạn bộ nhớ và CPU.

**Application framework 2.4.3 Android runtime:**

Bằng cách cung cấp một nền tảng phát triển mở, Android cung cấp cho các nhà phát triển khả năng xây dựng các ứng dụng cực kỳ phong phú và sáng tạo. Nhà phát triển được tự do tận dụng các thiết bị phần cứng, thông tin địa điểm truy cập, các dịch vụ chạy nền, thiết lập hệ thống báo động, thêm các thông báo để các thanh trạng thái, và nhiều, nhiều hơn nữa.

Nhà phát triển có thể truy cập vào các API được sử dụng bởi các ứng dụng lõi. Các kiến trúc ứng dụng được thiết kế để đơn giản hóa việc sử dụng lại các API. Đưa ra những khả năng khác nhau của hệ điều hành android vào ứng dụng để sử dụng chúng trong các ứng dụng của mình.

Cơ bản tất cả các ứng dụng là một bộ các dịch vụ và các hệ thống, bao gồm: các View (là dùng để hiển thị thông tin và để người dùng thao tác), Content Provider để chia sẻ dữ liệu giữa các ứng dụng, Resource Manager truy xuất tài nguyên, Notification Manager hiển thị các thông báo, Activity Manager quản lý chu trình sống của ứng dụng và điều hướng Activity.

**Applications:**

Tại lớp trên cùng sẽ là các ứng dụng cho android (như điện thoại, danh bạ, trình duyệt,…) cũng như các ứng dụng được tải về và cài đặt từ AndroidMarket hay bất lý ứng dụng nào bạn viết được tại tầng này.

## **Các khái niệm cơ bản trong lập trình ứng dụng Android**

**Activity :**

Một activity thể hiện một giao diện đồ họa người dùng. Ví dụ một activity có thể biểu diễn một danh sách các menu item để người dùng có thể chọn và có thể hiển thị ảnh cùng với tiêu đề. Một ứng dụng gửi tin nhắn văn bản có thể có một hoạt động là hiển thị một danh sách các liên hệ để gửi tin nhắn tới, hoạt động thứ hai là viết tin nhắn tới liên hệ được chọn, các hoạt động khác nữa là xem lại tin nhắn cũ hay thay đổi cài đặt. Mặc dù chúng làm việc cùng nhau để tạo thành một giao diện người dùng, mỗi activity độc lập với những cái khác. Mỗi activity là một lớp con của lớp cơ sở Activity.

Một ứng dụng có thể gồm chỉ một activity hay nhiều activity. Activity chính phải được hiển thị đầu tiên khi khởi động chương trình. Chuyển từ một activity sang activity khác bằng cách cho activity hiện thời khởi động activity kế tiếp.

Mỗi activity được vẽ vào một cửa sổ trên màn hình, mặc định sẽ lấp đầy màn hình, nhưng nó có thể nhỏ hơn màn hình và nằm trên các cửa sổ khác ví dụ như activity thông báo một thông tin gì đó.

Nội dung trực quan của cửa sổ được cung cấp bởi một cây phân cấp các đối tượng view dẫn xuất từ lớp View. Mỗi view điều khiển một khoảng hình chữ nhật cụ thể bên trong cửa sổ. View cha chứa và tổ chức bố cục các view con. Các view lá vẽ trong hình chữ nhật mà chúng điều khiển và đáp ứng lại các hành động người dùng trực tiếp ở khoảng trống này. Do đó, các view là nơi mà các tương tác của activity với người dùng diễn ra.

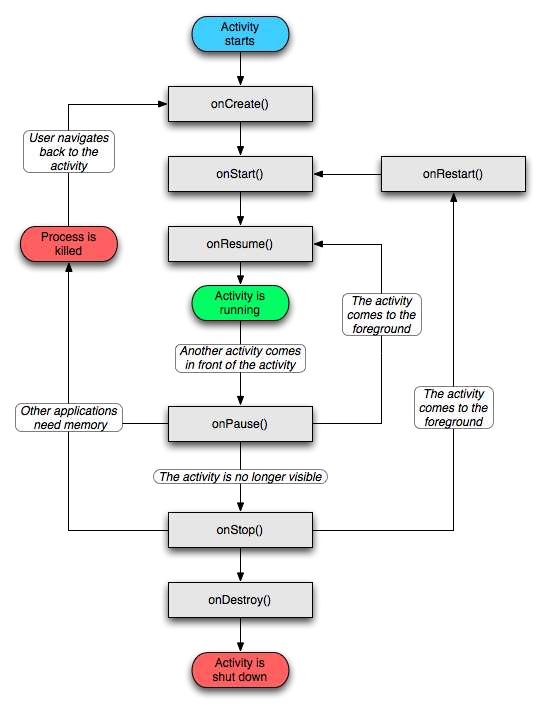
Ví dụ một view có thể hiển thị một hình ảnh nhỏ và khởi tạo một hoạt động khi người dùng nhấn vào hình ảnh đó. Android có một số view đã xây dựng sẵn mà bạn có thể sử dụng – gồm có các buttons, text fields, scroll bars, menu items, check boxes …

Một cây phân cấp view được đặt trong một cửa sổ của activity bằng phương thức Activity.setContentView(). Content view là đối tượng View ở gốc của cây phân cấp.

Class cơ sở Activity định nghĩa một loạt các sự kiện mà điều chỉnh vòng đời của một hoạt động. Class Activity định nghĩa các sự kiện sau đây :

* **onCreate():** Được gọi khi hoạt động được tạo ra lần đầu tiên
* **onStart():** Được gọi khi hoạt động trở nên hữu hình so với người dùng
* **onResume():** Được gọi khi hoạt động bắt đầu tương tác với người sử dụng
* **onPause():** Được gọi để dừng các hoạt động hiện tại và nối lại các hoạt động trước đó
* **onStop():** Được gọi khi hoạt động không còn hiển thị với người dùng
* **onDestroy():** Được gọi trước khi hoạt động bị phá hủy bởi hệ thống (bằng tay hoặc bằng hệ thống để bảo tồn bộ nhớ)
* **onRestart():** Được gọi khi hệ thống đã được dừng lại và khởi động lại một lần nữa

Sau đây là sơ đồ các sự kiện trong vòng đời của một Activity:

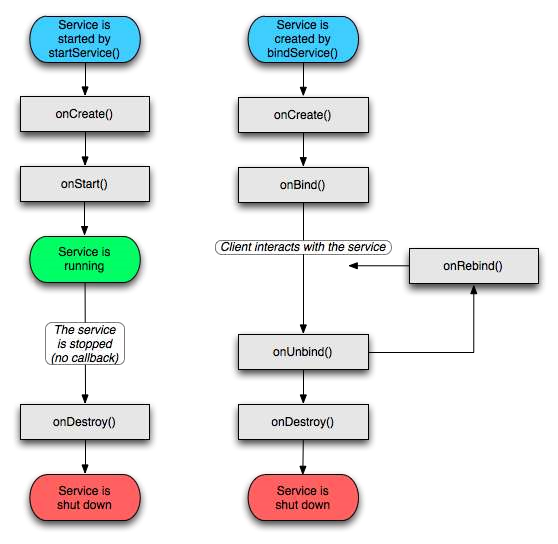


**Service :**

Một service không có giao diện trực quan, nó chạy trên nền trong một khoảng thời gian không xác định. Ví dụ một service có thể chơi nhạc nền, hay nó nạp dữ liệu trên mạng hay tính toán cái gì đó và cung cấp kết quả cho activity cần đến nó. Mỗi service mở rộng từ lớp cơ sở Service.

Trong khi kết nối, người sử dụng có thể giao tiếp với service thông qua giao diện mà service đó trưng ra. Ví dụ như trong service chơi nhạc, giao diện này có thể cho phép người dùng pause, rewind, stop và restart lại playback.

Giống như các activity và các thành phần khác khác, service chạy trong thread chính của tiến trình ứng dụng. Vì thế chúng không thể chặn những thành phần khác hay giao diện người dùng, chúng thường tạo ra các thead khác cho các nhiệm vụ hao tốn thời gian.Sơ đồ các sự kiện trong vòng đời của một service:



**Content provider :**

Một content provider tạo ra một tập cụ thể các dữ liệu của ứng dụng khả dụng cho các ứng dụng khác. Dữ liệu có thể được lưu trữ trong hệ thống file, trong một cơ sở dữ liệu SQLite, hay trong một cách khác nào đó. Content provider mở rộng lớp cơ sở ContentProvider để cài đặt một tập các chuẩn các phương thức cho phép các ứng dụng khác đạt được và lưu trữ dữ liệu của kiểu mà nó điều khiển. Tuy nhiên, các ứng dụng không gọi trực tiếp các phương thức này, chúng sử dụng một đối tượng ContentResolver và gọi các phương thức của nó. Một ContentResolver có thể nói chuyện với bất cứ content provider nào, chúng cộng tác với provider để quản lý giao tiếp liên tiến trình.

**Broadcast Receive :**

Một Broadcast Rreceiver là một thành phần không làm gì ngoài việc nhận và đáp lại các thông báo broadcast. Nhiều broadcast khởi đầu trong mã hệ thống - ví dụ như thông báo múi giờ thay đổi, pin yếu, ảnh đã được chụp, hay người dùng đã thay đổi ngôn ngữ … Các ứng dụng có thể tạo ra các broadcast, chẳng hạn để ứng dụng khác biết được một số dữ liệu đã được tải về thiết bị và sẵn sàng cho việc sử dụng chúng.

Một ứng dụng có thể có một số Broadcast Receiver để đáp lại bất cứ thông báo nào mà nó cho là quan trọng. Tất cả các receiver mở rộng từ lớp cơ sở BroadcastReceiver.

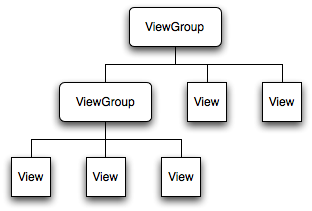
Broadcast Receiver không hiển thị một giao diện người dùng. Tuy nhiên chúng có thể bắt đầu một activity để đáp lại thông tin mà chúng nhận, hay chúng có thể sử dụng NotificationManager để cảnh báo người dùng. Notifications có thể lấy sự chú ý của người dùng bằng nhiều cách, lóe sáng đèn sau, rung, tạo ra âm thanh, vân vân. Chúng thường lấy một biểu tượng bền vững trong thanh trạng thái, cái mà người dùng có thể mở để lấy thông điệp.

**View :**

Trong một ứng dụng Android, giao diện người dùng được xây dựng từ các đối tượng View và ViewGroup. Có nhiều kiểu View và ViewGroup. Mỗi một kiểu là một con của class View và tất cả các kiểu đó được gọi là các Widget.

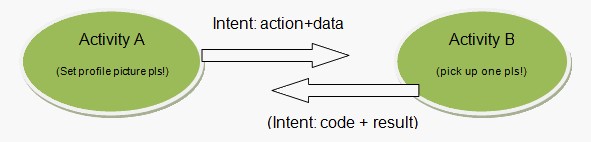
Tất cả mọi widget đều có chung các thuộc tính cơ bản như là cách trình bày vị trí, background, kích thước, lề,… Tất cả những thuộc tính chung này được thể hiện hết ở trong đối tượng View.

Trong Android Platform, các screen luôn được bố trí theo một kiểu cấu trúc phân cấp như hình dưới. Một màn hình là một tập hợp các Layout và các widget được bố trí có thứ tự. Để thể hiện một màn hình thì trong hàm onCreate của mỗi Activity cần phải được gọi một hàm là setContentView(R.layout.main); hàm này sẽ load giao diện từ file XML lên để phân tích thành mã bytecode.



**Intent :**

Là cầu nối giữa các Activity: ứng dụng Android thường bao gồm nhiều Activity, mỗi Activity hoạt động độc lập với nhau và thực hiện những công việc khác nhau. Intent chính là người đưa thư, giúp các Activity có thể triệu gọi cũng như truyền các dữ liệu cần thiết tới một Activity khác. Điều này cũng giống như việc di chuyển qua lại giữa các Forms trong lập trình Windows Form.



1. **Các thành phần trong một project ứng dụng Android**

**AndroidManifest.xml** :

Trong bất kì một dự án Android nào khi tạo ra đều có một file AndroidManifest.xml, file này được dùng để định nghĩa các màn hình sử dụng, các quyền cũng như các giao diện cho ứng dụng. Đồng thời nó cũng chứa thông tin về phiên bản SDK cũng như màn hình chính sẽ chạy đầu tiên.

**File R.java** :

File R.java là một file tự động sinh ra ngay khi tạo ứng dụng, file này được sử dụng để quản lý các thuộc tính được khai báo trong file XML của ứng dụng và các tài nguyên hình ảnh.Mã nguồn của file R.java được tự động sinh khi có bất kì một sự kiện nào xảy xa làm thay đổi các thuộc tính trong ứng dụng.

Có thể nói file R.java hoàn toàn không cần phải đụng chạm gì đến trong cả quá trình xây dựng ứng dụng.

**Thư mục src** :

Là vị trí chứa gói các class trong ứng dụng. Các class có thể là các một Activity hoặc Service hoặc các lớp chức năng nào đó được viết bằng ngôn ngữ Java dựa trên API được cung cấp sẵn của Android. Cần phải có ít nhất một Activity và khai báo là Activity chính để chương trình có thể chạy được.

**Thư mục rec** :

Thư mục chứa tài nguyên ứng dụng. Thư mục này bao gồm 5 thư mục con là: drawable – hdpi, drawable – mdpi, drawable – ldpi,layout,values.

Drawable – hdpi, drawable – mdpi, drawable – ldpi là ba thư mục dùng để chứa các hình ảnh được sử dụng trong quá trình thiết kế giao diện ứng dụng, bao gồm cả icon của ứng dụng. 3 thư mục tương ứng với hình ảnh sẽ được sử dụng ở ba độ phân giải khác nhau lần lược là: cao, trung bình, thấp. Điều này giúp các nhà lập trình có thể thiết kế giao diện ứng dụng phù hợp với nhiều độ phân giải màn hình tương thích với nhiều lại thiết bị.

Thư mục layout chứa các file xml dùng để khai báo và thiết kế giao diện cho một Activity hay một thành phần điều khiển con trong ứng dụng Android.

Thư mục values gồm các file xml chứa các giá trị chuỗi, mã màu … Giúp người lập trình có thể dễ dàng thay đổi những giá trị này trong ứng dụng một cách nhanh chóng mà khôn cần phải sửa trong code của ứng dụng.

1. **GOOGLE MAP VÀ GOOGLE MAPS API WEB**

## **Google Map**

* 1. **Giới thiệu Google Map**

**Google Maps** (thời gian trước còn gọi là **Google Local**) là một dịch vụ ứng dụng và công nghệ bản đồ trực tuyến trên web miễn phí được cung cấp bởi Google và hỗ trợ nhiều dịch vụ dựa vào bản đồ như Google Ride Finder và một số có thể dùng để nhúng vào các trang web của bên thứ ba thông qua Google Maps API. Nó cho phép thấy bản đồ đường sá, đường đi cho xe đạp, cho người đi bộ và xe hơi, và những địa điểm kinh doanh trong khu vực cũng như khắp nơi trên thế giới.



Một sản phẩm liên quan, đó là Google Earth, một ứng dụng độc lập dành cho Microsoft Windows, Mac OS X và Linux cho phép xem các tính năng mở rộng khác.



Google Map là một trong những ứng dụng phổ biến nhất của Google hiện nay. Khi mà việc sử dụng bản đồ giấy đã trở lên lỗi thời thì dịch vụ tìm kiếm bản đồ số, bản đồ vệ tinh, hệ thống GPS là hết sức cần thiết. Giờ đây chúng ta có thể dễ dàng truy cập vào Google Map dù ở bất cứ nơi đâu để tra cứu địa chỉ hoặc sử dụng hệ thống GPS cũng rất phổ biến. Đặt biệt khả năng tích hợp dịch vụ bản đồ Google Map là rất cần thiết và quan trọng.

* 1. **Google Maps for Mobile**

Năm 2006, Google đã giới thiệu một ứng dụng Java có tên là **Google Maps for Mobile.** Mục đích để chạy trên bất kỳ thiết bị di động nào dựa có cũng cấp nền tảng Java. Rất nhiều tính năng chính của phiên bảng Google Maps trên web có mặt trong ứng dụng.

Ngày 28 tháng 11, 2007, Google Maps for Mobile 2.0 được phát hành. Nó cung cấp một dịch vụ xác định vị trí giống như GPS nhưng vẫn hoạt động nếu thành phần GPS trong phần cứng không có hoạt không hoạt động. Vị trí hiện tại chỉ được bắt nếu thành phần GPS sẵn sàng hoạt động. Tính năng định vị không cần thành phần GPS dựa vào phần mềm xác định trạm phát sóng không dây và trạm BTS gần nhất. Sau đó phần mềm sẽ tìm trong cơ sở dữ liệu của nó vị trí của các điểm phát sóng WiFi và các trạm BTS mà nó biết được. Bằng cách xác định cường độ tín hiệu từ nhiều trạm phát sóng WiFi và BTS đã biết vị trí, phần mềm sẽ tính toán ra được vị trí hiện tại của người dùng.

Thứ tự ưu tiên tính toán vị trí như sau:

* Thiết bị bắt vị trí GPS
* Trạm WiFi
* Trạm BTS

Tính đến ngày 15 tháng 12 năm 2008, dịch vụ này có sẵn cho các nền tảng sau đây:

* Android
* iOS (iPhone, iPod Touch, iPad)
* PlayStation Vita 3G Models
* Windows Mobile (không hỗ trợ Windows Phone 7 từ 21/12/2011)
* Nokia/Symbian (S60 3rd & 5th)
* Symbian OS (UIQ v3)
* BlackBerry
* Điện thoại hỗ trợ Java-Platform (từ MIDP 2.0 trở lên)
* Palm OS (Centro và mới hơn)
* Palm webOS (Palm Pre và Palm Pixi)

Ngày 4 tháng 11, năm 2009, Google Maps Navigation đã được phát hành trên hệ điều hành Google Android 2.0 Eclair trên điện thoại Motorola Droid, thêm chức năng ra lệnh bằng giọng nói, báo cáo lưu lượng truy cập, và hỗ trợ xem đường phố. Phát hành ban đầu được giới hạn ở Hoa Kỳ. Dịch vụ này đã được đưa ra tại Anh vào ngày 20 tháng 4 năm 2010 và trong phần lớn của lục địa châu Âu vào ngày 09 tháng 6 2010 (bao gồm Áo, Bỉ, Canada, Đan Mạch, Pháp, Đức, Ý, Hà Lan, Bồ Đào Nha, Tây Ban Nha, và Thụy Sĩ ).

Trong tháng 3 năm 2011, Phó Chủ tịch Dịch vụ Location Service, Marissa Mayer thông báo mà Google đã cung cấp dịch vụ bản đồ đến 150 triệu người sử dụng.

Trong tháng sáu năm 2012, Apple thông báo rằng họ sẽ loại bỏ Google Maps trên IOS 6 và sẽ được thay thế bằng dịch vụ bản đồ của riêng của họ.

## **Google Maps API Web Services**

* 1. **Giới thiệu Google Maps API Web Services**

Google Maps API Web Services là một tập các giao diện HTTP cung cấp thông tin địa lý cho ứng dụng.

Google Maps API Web Services bao gồm:

* Directions API
* Distance Matrix API
* Elevation API
* Geocoding API
* Places API
  1. **Sử dụng Google Maps API Web Services**

Google Maps API cung cấp các dịch vụ như là các giao diện phục vụ cho việc yêu cầu dữ liệu địa lý và sử dụng dữ liệu đó trong ứng dụng của nhà phát triền. Các dịch vụ này được thiết kế để sử dụng với một ứng dụng bản đồ.

Các dịch vụ này sử dụng HTTP request, thiết lập các chuỗi URL request để gởi yêu cầu cho web service. Thông thường, web service sẽ trả về kết quả là tập tin JSON hoặc XML. Phân tích cú pháp kết quả trả về để sử dụng.

URL request có dạng:

*http://maps.googleapis.com/maps/api/service/output?parameters*

* **service**: loại dịch vụ.
* **output**: kết quả ở dạng tập tin JSON hay XML.
* **parameters**: các tham số phù hợp.

**SSL Access :**

Có thể truy cập Google Maps API Web Services thông qua HTTPS. Truy cập Maps API Web Service thông qua HTTPS được khuyến khích sử dụng nếu ứng dụng có chứa những dữ liệu nhạy cảm như tọa độ người dùng.

*https://maps.googleapis.com/maps/api/service/output?parameters*

**Theo dõi sử dụng với tham số sensor :**

Sử dụng Google Maps API yêu cầu nhà phát triển phải chỉ ra rằng ứng dụng có sử dụng một cảm biến (ví dụ cảm biến GPS) để xác định vị trí người dùng trong bất cứ yêu cầu nào cho service. Nó khá quan trọng cho điện thoại. Nếu ứng dụng Google Map API sử dụng bất cứ dạng cảm biến để xác định vị trí của thiết bị phải thiết lập giá trị của tham số sensor=true.

Trong trường hợp ứng dụng không sử dụng cảm biến thì vẫn phải gán giá trị sensor=false.

**Xây dựng chuỗi URL request hợp lệ :**

Một chuỗi URL có thể chứa những ký tự đặt biệt. Ví dụ trong trình duyệt web, chuỗi URL được nhập vào thanh địa chỉ chứa những ký tự đặt biệt (ví dụ chứa tiếng Việt có dấu), trình duyệt phải tự chuyển sang kiểu mã hóa khác trước khi gởi đi. Một số trình duyệt có thể nhận những chuỗi ký tự ở dạng mã hóa UTF-8. Quá trình chuyển kiểu mã hóa ký tự đó gọi là URL-encoding.

Cần phải dịch những ký tự đặt biệt bởi vì tất cả các URL cần phải phù hợp với cú pháp đã được qui định ở W3 Uniform Resource Identifier. Có nghĩa là URL chỉ chứa các ký tự thuộc một tập hợp các ký tự ASCII. Các ký tự được sử dụng trong chuỗi URL được qui định trong bảng sau:

|  |  |
| --- | --- |
| **Nhóm** | **Ký tự** |
| Alphanumeric | a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9 |
| Unreserved | - \_ . ~ |
| Reserved | ! \* ' ( ) ; : @ & = + $ , / ? % # [ ] |

Việc chuỗi URL chỉ chứa các ký tự ASCII đã được qui định dẫn đến 2 tình huống:

Ký tự muốn đưa vào URL không nằm trong tập ký tự được cho phép (ví dụ tiếng Việt có dấu)cần được mã hóa bằng các ký tự trên. Ký tự khoảng cách thường được thay thế bằng ký tự “+”.

Ký tự muốn sử dụng theo cách thông thường nằm trong nhóm ký tự dành riêng (Ví dụ ký tự “?”). Nếu muốn sử dụng ký tự dành riêng phải mã hóa.

Ký tự được mã hóa bằng cách sử dụng ký tự “%” theo sau là hai ký tự chứa giá trị hex đại diện cho ký tự hệ UTF-8.

Giới hạn của chuối URL là 2048 ký tự. Hầu hết các URL request ít khi đạt đến ngưỡng này, tuy nhiên vẫn phải cẩn thận để tránh trường hợp web service không trả lời vì chuỗi URL quá dài.

**Processing Responses :**

Kết quả trả về của Google Direction API khá dễ hiểu nhưng lại ít thân thiện với người sử dụng. Khi giởi yêu cầu và nhận được kết quả, tốt nhất là không hiển thị tất cả những gì nhận được mà chỉ cần trích xuất một số thông tin phù hợp. Tóm lại là cần phải phân tích cú pháp kết quả nhận được và chỉ trích xuất một số thông tin có ích.

Phân tích cú pháp kết quả nhận được phụ thuộc vào loại tập tin trả về là XML hay JSON và có thể sử dụng nhiều kỹ thuật và ngôn ngữ lập trình khác nhau.

## **Google Geocoding API**

* 1. **Mã hóa địa lý là gì?**

Mã hóa địa lý là tiến trình chuyển địa chỉ (ví dụ: số 2, đường Nguyễn Đình Chiểu, Vĩnh Thọ, Nha Trang)sang tọa độ địa lý (ví dụ: kinh độ 12.268491, vĩ độ 109.202148) để đánh dấu trên bảng đồ. Google Geocoding API cung cấp một phương thức trực tiếp để truy cập bộ mã hóa địa lý thông qua giao thức HTTP. Ngoài ra dịch vụ Google Geocoding API còn cho phép giải mã từ tọa độ sang địa chỉ.

* 1. **Giới hạn sử dụng**

Google Geocoding API giới hạn ở 2500 truy vấn mỗi ngày. Nếu trả phí con số này tăng lên 100000. Các giới hạn này được đặt ra để ngăn chặn sự lạm dụng dịch vụ. Nếu vượt quá giới hạn này ứng dụng sẽ không thể truy cập được sau 24 giờ, nếu tiếp tục vượt ở 24 giờ tiếp theo thì dịch dụ Geocoding API có thể sẽ chặn ứng dụng.

Sử dụng Geocoding API cần phải kết hợp với một bản đồ Google Map dùng để hiển thị các địa điểm một cách trực quan trên bản đồ.

* 1. **Geocoding API request**

**Định dạng chuỗi request URL :**

Chuỗi request URL có dạng như sau:

*http://maps.googleapis.com/maps/api/geocode/output?parameters*

HTTPS được khuyến cáo sử dụng cho các ứng dụng có dùng những dữ liệu nhạy cảm của người dùng: *https://maps.googleapis.com/maps/api/geocode/output?parameters*

Output có thể là xml hoặc json:

* json (khuyến cáo): kết quả trả về tập tin JSON (JavaScript Object Notation). Loại tập tin này phù hợp hơn cho các ứng dụng bản đồ trên nền web.
* xml: kết quả trả về tập tim XML.

**Tham số của chuỗi request URL :**

Một số tham số là bắt buộc phải được gán giá trị, một số là tùy chọn. Trong một chuỗi URL hợp chuẩn mỗi tham số được phân cách bằng ký tự (&).

* **address** (bắt buộc): địa chỉ muốn mã hóa địa lý.

HOẶC

* **latlng** (bắt buộc): Giá trị kinh độ - vĩ độ của điểm muốn lấy địa chỉ gần nhất.
* **bounds** (tùy chọn): Hình chữ nhật mà kết quả mã hóa địa lý sẽ hiện lên nổi bật.
* **region** (tùy chọn): mã vùng, dùng hệ thống tên miền quốc gia.
* **language** (tùy chọn): thiết lập ngôn ngữ sẽ trả về trong kết quả.
* **sensor** (bắt buộc): cho biết yêu cầu tìm đường có phải đến từ một thiết bị có cảm biến vị trí hay không. True hoặc false.

Chỉ có thể dùng một trong hai tham số address hoặc latlng để tìm. Nếu tham số latlng được sử dụng thì quá trình này tạm gọi là giải mã địa lý.

* 1. **Kết quả mã hóa địa lý**

Geocoding API sẽ trả về một rong hai loại tập tin là JSON hoặc XML.

Trong ví dụ này sẽ yêu cầu Geocoding API mã hóa địa lý địa chỉ “ số 6, đường Biệt Thự, Lộc Thọ, Nha Trang”



Chuỗi request URL:

*http://maps.googleapis.com/maps/api/geocode/json?address=1600+Amphitheatre+P arkway,+Mountain+View,+CA&sensor= false*

***Kết quả trả về ở định dạng JSON:***

*{*

*"results" : [*

*{*

*"address\_components" : [*

*{*

*"long\_name" : "6",*

*"short\_name" : "6",*

*"types" : [ "street\_number" ]*

*},*

*{*

*"long\_name" : "Biệt Thự",*

*"short\_name" : "Biệt Thự",*

*"types" : [ "route" ]*

*},*

*{*

*"long\_name" : "Lộc Thọ",*

*"short\_name" : "Lộc Thọ",*

*"types" : [ "sublocality", "political" ]*

*},*

*{*

*"long\_name" : "Nha Trang",*

*"short\_name" : "Nha Trang",*

*"types" : [ "locality", "political" ]*

*},*

*{*

*"long\_name" : "Khanh Hoa",*

*"short\_name" : "Khanh Hoa",*

*"types" : [ "administrative\_area\_level\_1", "political" ]*

*},*

*{*

*"long\_name" : "Vietnam",*

*"short\_name" : "VN",*

*"types" : [ "country", "political" ]*

*}*

*],*

*"formatted\_address" : "6 Biệt Thự, Lộc Thọ, Nha Trang, Khanh Hoa province, Vietnam",*



*"geometry" : {*

*"location" : {*

*"lat" : 12.23569550,*

*"lng" : 109.19563650*

*},*

*"location\_type" : "ROOFTOP",*

*"viewport" : {*

*"northeast" : {*

*"lat" : 12.23704448029150,*

*"lng" : 109.1969854802915*

*},*

*"southwest" : {*

*"lat" : 12.23434651970850,*

*"lng" : 109.1942875197085*

*}*

*}*

*},*

*"types" : [ "street\_address" ]*

*}*

*],*

*"status" : "OK"*

*}*

***Kết quả trả về ở định dạng XML:***

*<?xml version="1.0" encoding="UTF-8"?>*

*<GeocodeResponse>*

*<status>OK</status>*

*<result>*

*<type>street\_address</type>*

*<formatted\_address>6 Biệt Thự, Lộc Thọ, Nha Trang, Khanh Hoa province,*

*Vietnam</formatted\_address>*

*<address\_component>*

*<long\_name>6</long\_name>*

*<short\_name>6</short\_name>*

*<type>street\_number</type>*

*</address\_component>*

*<address\_component>*

*<long\_name>Biệt Thự</long\_name>*

*<short\_name>Biệt Thự</short\_name>*

*<type>route</type>*

*</address\_component>*

*<address\_component>*

*<long\_name>Lộc Thọ</long\_name>*

*<short\_name>Lộc Thọ</short\_name>*

*<type>sublocality</type>*

*<type>political</type>*

*</address\_component>*

*<address\_component>*

*<long\_name>Nha Trang</long\_name>*

*<short\_name>Nha Trang</short\_name>*

*<type>locality</type>*

*<type>political</type>*

*</address\_component>*

*<address\_component>*

*<long\_name>Khanh Hoa</long\_name>*

*<short\_name>Khanh Hoa</short\_name>*

*<type>administrative\_area\_level\_1</type>*

*<type>political</type>*

*</address\_component>*

*<address\_component>*

*<long\_name>Vietnam</long\_name>*

*<short\_name>VN</short\_name>*

*<type>country</type>*

*<type>political</type>*

*</address\_component>*

*<geometry>*

*<location>*

*<lat>12.2356955</lat>*

*<lng>109.1956365</lng>*

*</location>*

*<location\_type>ROOFTOP</location\_type>*

*<viewport>*

*<southwest>*

*<lat>12.2343465</lat>*

*<lng>109.1942875</lng>*

*</southwest>*

*<northeast>*

*<lat>12.2370445</lat>*

*<lng>109.1969855</lng>*

*</northeast>*

*</viewport>*

*</geometry>*

*</result>*

*</GeocodeResponse>*

Cấu trúc tập tin rả về có hai phần tử gốc:

* **status**: thông báo trạng thái của kết quả.
* **result**: chứa một tập thông tin kết quả.

Thông thường chỉ có một phần tử result trả về trong kết quả, tuy nhiên Geocoding API có thể trả về nhiều thẻ result hơn nếu truy vấn địa chỉ không rõ ràng.

Kết quả trả về cần phải phân tích cú pháp để trích xuất thông tin cần thiết.

**STATUS :**

* **OK**: kết quả trả về bình thường.
* **ZERO\_RESULTS**: địa chỉ không tồn tại hoặc gí trị kinh độ - vĩ độ không đúng.
* **OVER\_QUERY\_LIMIT**: thông báo đã gởi quá số lượng yêu cầu trong một khoảng thời gian.
* **REQUEST\_DENIED**: ứng dụng đã bị dịch vụ Google Distance Matrix API chặn.
* **INVALID\_REQUEST**: thông báo chuỗi request URL không hợp lệ. **RESULT**
* **type[]:** Một tập thẻ type cho biết kiểu của kết quả trả về. Chứa một hay nhiều kiểu của địa danh, ví dụ khi mã hóa địa lý Nha Trang sẽ trả về hai giá trị type là political và locality.
* **formatted\_address**:là địa chỉ bằng chữ.
* **address\_component[]:** chứa một mảng các thành phần địa lý. Ví dụ Nha Trang là một thành phần, Khánh Hòa là một thành phần. Mỗi thẻ address\_component chứa: o **type[]:** kiểu của thành phần địa lý. o **long\_name:** tên đầy đủ của thành phần địa lý.
  + **short\_name:** tên viết tắt của thành phần địa lý.
* **geometry:** chứa thông tin hình học của địa điểm mã hóa địa lý. Trong thẻ geometry bao gồm các thẻ sau:
  + **location:** chứa kết quả kinh độ vĩ độ của địa điểm. Đây là thông tin chính của kết quả.
  + **location\_type:** chứa thông tin bổ sung về địa điểm . Giá trị của location\_type có thể là một trong danh sách sau:
    - **ROOFTOP**: cho biết kết quả trả về là chính xác đến mức đến độ chính xác đường phố.
    - **RANGE\_INTERPOLATED**: cho biết kết quả nhận được bằng phương pháp nội xuy giữa hai điểm đã biết địa chỉ chính xác.
    - **GEOMETRIC\_CENTER**: cho biết kết quả trả về là trung tâm của kết quả. Kết quả này nhận được khi tìm các địa chỉ dạng vùng (tỉnh, thành phố) hay đường phố.
    - **APPROXIMATE**: chỉ rằng kết quả trả về trả về là gần đúng. o **viewport**: chứa thông tin về khung nhìn của kết quả. Gồm có tọa độ điểm Tây-Bắc và Đông-Nam. Viewport dùng để hiển thị trên bản đồ cho người dùng.
  + **bound** (có thể có trả về hoặc không): chứa thông tin một hình chữ nhật bao toàn bộ kết quả trả về. Có thể khác viewport.
* **partial\_match**: thông báo Geocoding API trả về một kết quả không chính xác, chỉ tìm được một phần so với địa chỉ được yêu cầu. Có thể do lỗi chính tả của địa chỉ nhập vào, hoặc địa chỉ không đầy đủ, hay đường phố không tồn tại trong địa phương được yêu cầu.

Lưu ý rằng trong kết quả trả lại các thành phần địa lý theo cấp độ có thể không nằm ở vị trí tuyệt đối mà phải xem xét kiểu của thành phần địa lý.

Kiểu thành phần địa lý

Một tập thẻ type cho biết kiểu của thành phần địa chỉ. Địa chỉ được mã hóa địa lý có thể có nhiều kiểu và được xem như là tag.

**Các giá trị của type :**

* **street\_address**: địa chỉ đường.
* **route**: địa chỉ tuyến đường (chỉ có ở Mĩ).
* **intersection**: là ngã tư lớn, giao nhau giữa hai con đường chính.
* **political**: địa chỉ hành chính.
* **country**: quốc gia, là địa chỉ cấp cao nhất.
* **administrative\_area\_level\_1**: địa chỉ hành chính cấp 1 dưới cấp quốc gia, ở Việt Nam là tỉnh hoặc các thành phố trực thuộc Trung ương, ở Mỹ là bang.
* **administrative\_area\_level\_2**: địa chỉ hành chính cấp 2 dưới cấp quốc gia. Ở Việt Nam là các quận của thành phố trược thuộc Trung ương.
* **administrative\_area\_level\_3**: địa chỉ hành chính cấp 3 dưới cấp quốc gia. Chỉ có ở Mỹ và một số quốc gia.
* **colloquial\_area**: tên địa chỉ thường dùng.
* **locality**: thành phố hoặc thị xã.
* **sublocality**: địa chỉ hành chính dưới cấp locality.
* neighborhood: khu phố
* **premise**: một địa điểm được đặt tên, thường là một tòa nhà hoặc một cụm các tòa nhà được đặt chung tên.
* **subpremise**: cấp dưới của premise, thường là một tòa nhà nằm trong một cụm được đặt tên.
* **postal\_code**: mã bưu chính.
* **natural\_feature**: indicates một địa điểm tự nhiên.
* **airport**: sân bay.
* **park**: công viên.
* **point\_of\_interest**: một địa điểm nổi tiếng.

*Ngoài ra còn có các giá trị khác của type hiếm khi sử dụng:*

* **post\_box**: hộp thư.
* street\_number: số đường.
* **floor**: tầng nhà trong một tòa nhà.
* **room**: phòng trong một tòa nhà.

**Giới hạn vùng tìm kiếm :**

Có thể giới hạn vùng tìm kiếm địa chỉ bằng cách thêm vào tham số bound gồm có kinh độ vĩ độ - vĩ độ của góc Tây Nam và Đông Bắc của khung tìm kiếm phân tách nhau bằng ký tự (|).

Ví dụ tìm đường Nguyễn Thị Minh Khai:

*http://maps.googleapis.com/maps/api/geocode/json?address=Winnetka&sensor=fals e*

Rất nhiều kết quả nhận được từ chuỗi URL request này. Muốn tìm được đường Nguyễn Thị Minh Khai ở Nha Trang cần phải giới hạn vùng tìm kiếm. *http://maps.googleapis.com/maps/api/geocode/json?address=Nguyễn Thị Minh Khai&bounds=12.211684,109.166822|12.259496,109.225359&sensor=false*

Ngoài ra có thể giới hạn vùng tìm kiếm theo quốc gia.

*http://maps.googleapis.com/maps/api/geocode/json?address=Toledo&sensor=false& region=es*

* 1. **Giải mã địa lý (tìm địa chỉ từ kinh độ-vĩ độ)**

Khái niệm mã hóa địa lý dùng để chỉ việc chuyển từ địa chỉ mà con người đọc được thành vị trí trên bảng đồ. Giải mã địa lý tức là chuyển từ vị trí trên bảng đồ thành địa chỉ mà con người đọc được.

Dữ liệu đầu vào là giá trị kinh độ - vĩ độ. Ví dụ dưới đây sẽ tìm địa chỉ của vị trí có tọa độ 12.268355,109.20116.

*http://maps.googleapis.com/maps/api/geocode/xml?latlng=12.268554,109.202104&s ensor= false&language=vi-VN*

***Kết quả trả về ở định dạng XML:***

*<?xml version="1.0" encoding="UTF-8"?>*

*<GeocodeResponse>*

*<status>OK</status>*

*<result>*

*<type>street\_address</type>*

*<formatted\_address>2 Nguyễn Đình Chiểu, Đại học Nha Trang, Vĩnh Thọ, Nha Trang, Khánh Hòa, Việt Nam</formatted\_address>*

*<address\_component>*

*<long\_name>2</long\_name>*

*<short\_name>2</short\_name>*

*<type>street\_number</type>*

*</address\_component>*

*<address\_component>*

*<long\_name>Đại học Nha Trang</long\_name>*

*<short\_name>Đại học Nha Trang</short\_name>*

*<type>establishment</type>*

*</address\_component>*

*<address\_component>*

*<long\_name>Nguyễn Đình Chiểu</long\_name>*

*<short\_name>Nguyễn Đình Chiểu</short\_name>*

*<type>route</type>*

*</address\_component>*

*<address\_component>*

*<long\_name>Vĩnh Thọ</long\_name>*

*<short\_name>Vĩnh Thọ</short\_name>*

*<type>sublocality</type>*

*<type>political</type>*

*</address\_component>*

*<address\_component>*

*<long\_name>Nha Trang</long\_name>*

*<short\_name>Nha Trang</short\_name>*

*<type>locality</type>*

*<type>political</type>*

*</address\_component>*

*<address\_component>*

*<long\_name>Khánh Hòa</long\_name>*

*<short\_name>Khánh Hòa</short\_name>*

*<type>administrative\_area\_level\_1</type>*

*<type>political</type>*

*</address\_component>*

*<address\_component>*

*<long\_name>Việt Nam</long\_name>*

*<short\_name>VN</short\_name>*

*<type>country</type>*

*<type>political</type>*

*</address\_component>*

*<geometry>*

*<location>*

*<lat>12.2697490</lat>*

*<lng>109.2022615</lng>*

*</location>*

*<location\_type>ROOFTOP</location\_type>*

*<viewport>*

*<southwest>*

*<lat>12.2684000</lat>*

*<lng>109.2009125</lng>*

*</southwest>*

*<northeast>*

*<lat>12.2710980</lat>*

*<lng>109.2036105</lng>*

*</northeast>*

*</viewport>*

*</geometry>*

*</result>*

*<result>*

*…*

Kết quả trả về sẽ nhiều hơn một thẻ result. Geocoding sẽ trả về bất kỳ kết quả nào tìm được.

Danh sách formatted\_address mà Geocodin API tìm được với tọa độ

12.268355,109.20116:

* 2 Nguyễn Đình Chiểu, Vĩnh Phước, Nha Trang, Khánh Hòa, Việt Nam
* Vĩnh Thọ, Nha Trang, Khánh Hòa, Việt Nam
* Nha Trang, Khanh Hoa province, Vietnam
* Khanh Hoa province, Vietnam
* Vietnam

Địa chỉ trả về từ cụ thể nhất đến ít cụ thể nhất. Kết quả quan trọng nhất là địa chỉ cụ thể nhất nằm trong thẻ result đầu tiên.

Geocoding API sẽ mã hóa địa lý bằng cách tìm vị trí gần nhất trong một khoảng sai số nhất định, nếu không tìm thấy Geocoding API sẽ trả về kết quả rỗng.

## **Google Distance Matrix API**

Google Distance Matrix API là một dịch vụ cung cấp khoảng cách và thời gian di chuyển cho một ma trận các điểm gốc và điểm đích. Thông tin trả về phụ thuộc vào đường đi được đề nghị giữa điểm gốc và điểm đích theo tính toán của Google Distance Matrix API, bao gồm các giá trị khoảng cách và thời gian cho mỗi cặp.

Dịch vụ Google Distance Matrix API không cung cấp chi tiết đường đi. Muốn tìm thông tin chi tiết đường đi có thể sử dụng dịch dụ Google Direction API với chỉ một cặp điểm gốc và điểm đích.

* 1. **Giới hạn sử dụng**

Mỗi truy vấn gởi cho Distance Matrix API bị giới hạn số lượng phần tử, cứ mỗi một điểm gốc đến một điểm đích là một phần tử.

Số lượng phần tử bị giới hạn như sau:

* 100 phần tử cho mỗi truy vấn.
* 100 phần tử cho mỗi 10 giây.
* 2500 phần tử cho mỗi giờ.

Google MAP API trả phí sẽ có giới hạn lớn hơn:

* 625 phần tử cho mỗi truy vấn.
* 1000 phần tử cho mỗi 10 giây.
* 100000 phần tử cho mỗi giờ.

Các Distance Matrix API URL giới hạn ở 2048 ký tự trước khi mã hóa. Trong mỗi chuỗi Distance Matrix API URL sẽ có nhiều thông tin địa điểm được đưa vào vì vậy nên chú ý giới hạn này khi xây dựng chuỗi request URL.

Sử dụng Distance Matrix API cần phải kết hợp với một bản đồ Google Map dùng để hiển thị các địa điểm một cách trực quan trên bản đồ.

* 1. **Distance Matrix API request**

**Định dạng chuỗi request URL:**

Chuỗi request URL có dạng như sau:

*http://maps.googleapis.com/maps/api/distancematrix/output?parameters*

HTTPS được khuyến cáo sử dụng cho các ứng dụng có dùng những dữ liệu nhạy cảm của người dùng:

*https://maps.googleapis.com/maps/api/distancematrix/output?parameters*

Output có thể là xml hoặc json:

* **json** (khuyến cáo): kết quả trả về tập tin JSON (JavaScript Object Notation). Loại tập tin này phù hợp hơn cho các ứng dụng bản đồ trên nền web.
* **xml**: kết quả trả về tập tim XML.

**Tham số của chuỗi request URL :**

Một số tham số là bắt buộc phải được gán giá trị, một số là tùy chọn. Trong một chuỗi URL hợp chuẩn mỗi tham số được phân cách bằng ký tự (&).

Các tham số tham gia trong chuỗi URL yêu cầu dịch vụ Distance Matrix gồm có:

* **origins** (bắt buộc): Một hay nhiều điểm gốc mà từ đó muốn tính khoảng cách đến các điểm đích. Mỗi điểm phân cách nhau bằng ký tự (|). Mỗi điểm có thể là chuỗi ký tự địa chỉ hoặc giá trị kinh độ - vĩ độ. Ví dụ:

*origins=89 Vân Đồn, Phước Hòa, Nha Trang|12.213802,109.195588*

* **destinations** (bắt buộc): Một hay nhiều điểm đích muốn tính khoảng cách từ các điểm gốc. Mỗi điểm phân cách nhau bằng ký tự (|). Mỗi điểm có thể là chuỗi ký tự địa chỉ hoặc giá trị kinh độ - vĩ độ.

Ví dụ:

*origins=17 Thái Nguyên, Phước Tân, Nha Trang|12.213802,109.195588*

Để xác định một điểm cần thông tịn địa chỉ hoặc kinh độ - vĩ độ. Nếu sử dụng thông tin địa chỉ, hệ thống Google Map API sẽ tự chuyển địa chỉ đó về giá trị kinh độ - vĩ độ. Nếu sử dụng giá trị kinh độ - vĩ độ để xác định một điểm thì giá trị kinh độ và vĩ độ được phân cách nhau bằng ký tự (,).

* **mode**(tùy chọn, mặc định là driving): xác định kiểu di chuyển, hiện tại hỗ trợ 3 kiểu di chuyển sau: o **driving** (mặc định): lái xe hơi.
  + **walking** : đi bộ, có thể qua những con đường cho người đi bộ hoặc đi trên đường tắt và vỉa hè.
  + **bicycling**: lái xe đạp, chọn đường dành riêng cho xe đạp (chỉ phổ biến ở Mỹ)
* **language**(tùy chọn): xác định ngôn ngữ của kết quả trả về. Để trả về tiếng Việt thì language=vi-VN.
* **avoid** (tùy chọn): chỉ ra tuyến đường sẽ phải tránh đường cao tốc hoăc tránh trạm thu phí.
* **units**(tùy chọn): xác định đơn vị cho kết quả tính toán.
* **sensor**(bắt buộc): cho biết yêu cầu tìm đường có phải đến từ một thiết bị có cảm biến vị trí hay không. True hoặc false.

**Restrictions** Google Distance Matrix sẽ tính toán đường đi với những giới hạn được đặt ra. Dựa vào những giới hạn đó Google Matrix sẽ tính được đường đi thuận lợi hơn cho người dùng. Hiện tại có 2 giá trị của giới hạn được hỗ trợ đó là:

* **avoid=tolls** : tránh trạm thu phí.
* **avoid=highways**: tránh đường cao tốc.

**Unit Systems :**

Trong kết quả Google Distance Matrix trả về có chứa thông tin độ dài tuyến đường của mỗi cặp điểm.

Có thể thay đổi đơn vị khoảng cách mà hệ thống đã lựa chọn bằng cách cung cấp giá trị cho tham số units. Có hai giá trị của tham số units là:

* **metric**: đơn vị hệ mét.
* **imperial** : dùng hệ thống đơn vị của Anh.

Lựa chọn đơn vị độ dài chỉ có hiệu lực đối với dòng ký tự mô tả khoảng cách, trá trị khoảng cách trả về ở dạng số luôn có đơn vị là mét.

* 1. **Kết quả từ Distance Matrix API**

Kết quả trả về của Distance Matrix API phụ thuộc vào output là xml hay json trong cuỗi request URL.

Chuỗi request URL dưới đây sẽ yêu cầu Google Distance Matrix tính toán khoảng cách với hai điểm gốc là Trường Đại học Nha Trang và Tháp Bà Bonaga, hai điểm đích là Ga Nha Trang và khách sạn Lodge.

*http://maps.googleapis.com/maps/api/distancematrix/xml?origins=2 Nguyễn*

*Đình Chiểu, Vĩnh Phước, Nha Trang|12.265288,109.196167&destinations=42 Trần*

*Phú, Lộc Thọ, Nha Trang, Khanh*

*Hoa|12.248427,109.184052&mode=driving&language=vi-VN&sensor=false* Google Distance Matrix sẽ tính toán 4 phần tử:

|  |  |
| --- | --- |
| Từ ĐH Nha Trang đến KS Lodge | Từ ĐH Nha Trang đến Ga Nha Trang |
| Từ Tháp Bà đến KS Lodge | Từ Tháp Bà đến Ga Nha Trang |

Kết quả trả về sẽ gồm nhiều row, mỗi row sẽ tương ứng với một điểm gốc, mỗi row gồm nhiều element tương ứng với mỗi điểm đích.

***Tập tin kết quả JSON :***

*{*

*"destination\_addresses" : [*

*"42 Trần Phú, Lộc Thọ, Nha Trang, Khánh Hòa, Việt Nam",*

*"24 Thái Nguyên, Phước Tân, Nha Trang, Khánh Hòa, Việt Nam"*

*],*

*"origin\_addresses" : [*

*"2 Nguyễn Đình Chiểu, Vĩnh Phước, Nha Trang, Khánh Hòa, Việt Nam",*

*"126 2 Tháng 4, Vĩnh Thọ, Nha Trang, Khánh Hòa, Việt Nam"*

*],*

*"rows" : [*

*{*

*"elements" : [*

*{*

*"distance" : {*

*"text" : "4,9 km",*

*"value" : 4933*

*},*

*"duration" : {*

*"text" : "6 phút",*

*"value" : 383*

*},*

*"status" : "OK"*

*},*

*{*

*"distance" : {*

*"text" : "3,6 km",*

*"value" : 3564*

*},*

*"duration" : {*

*"text" : "5 phút",*

*"value" : 309*

*},*

*"status" : "OK"*

*}*

*]*

*},*

*{*

*"elements" : [*

*{*

*"distance" : {*

*"text" : "3,2 km",*

*"value" : 3205*

*},*

*"duration" : {*

*"text" : "6 phút",*

*"value" : 346*

*},*

*"status" : "OK"*

*},*

*{*

*"distance" : {*

*"text" : "3,0 km",*

*"value" : 2955*

*},*

*"duration" : {*

*"text" : "4 phút",*

*"value" : 265*

*},*

*"status" : "OK"*

*}*

*]*

*}*

*],*

*"status" : "OK"*

*}*

Để lấy được những thông tin cần thiết cần phải phân tích cú pháp tập tin JSON.

***Tập tin kết quả XML:***

*<DistanceMatrixResponse>*

*<status>OK</status> <origin\_address>2 Nguyễn Đình Chiểu, Vĩnh Phước, Nha Trang, Khánh Hòa, Việt Nam</origin\_address> <origin\_address>126 2 Tháng 4, Vĩnh Thọ, Nha Trang, Khánh Hòa, Việt Nam</origin\_address>*

*<destination\_address>42 Trần Phú, Lộc Thọ, Nha Trang, Khánh Hòa, Việt Nam</destination\_address> <destination\_address>24 Thái Nguyên, Phước Tân, Nha Trang, Khánh Hòa, Việt Nam</destination\_address>*

*<row>*

*<element>*

*<status>OK</status>*

*<duration>*

*<value>383</value>*

*<text>6 phút</text>*

*</duration>*

*<distance>*

*<value>4933</value>*

*<text>4,9 km</text>*

*</distance>*

*</element>*

*<element>*

*<status>OK</status>*

*<duration>*

*<value>309</value>*

*<text>5 phút</text>*

*</duration>*

*<distance>*

*<value>3564</value>*

*<text>3,6 km</text>*

*</distance>*

*</element>*

*</row>*

*<row>*

*<element>*

*<status>OK</status>*

*<duration>*

*<value>346</value>*

*<text>6 phút</text>*

*</duration>*

*<distance>*

*<value>3205</value>*

*<text>3,2 km</text>*

*</distance>*

*</element>*

*<element>*

*<status>OK</status>*

*<duration>*

*<value>265</value>*

*<text>4 phút</text>*

*</duration>*

*<distance>*

*<value>2955</value>*

*<text>3,0 km</text>*

*</distance>*

*</element>*

*</row>*

*</DistanceMatrixResponse>*

**Cấu trúc của kết quả trả về** Các phần từ gốc:

**status**: thông báo trạng thái kết quả. **origin\_addresses**: chứa một tập các địa chỉ điểm gốc đã được API chuyển từ

thông tin đầu vào. Ngôn ngữ sẽ phụ thuộc vào giá trị của language trong request URL. **destination\_addresses**: chứa một tập các địa chỉ điểm đích đã được API chuyển từ thông tin đầu vào. Ngôn ngữ sẽ phụ thuộc vào giá trị của language trong request URL.

**row**: chứa một tập các element.

**STATUS :**

* **OK**: kết quả trả về bình thường.
* **INVALID\_REQUEST**: thông báo chuỗi request URL không hợp lệ.
* **MAX\_ELEMENTS\_EXCEEDED**: thông báo vượt quá giới hạn các phần tử (element)
* **OVER\_QUERY\_LIMIT**: thông báo đã gởi quá số lượng yêu cầu trong một khoảng thời gian.
* **REQUEST\_DENIED**: ứng dụng đã bị dịch vụ Google Distance Matrix API chặn.
* **UNKNOWN\_ERROR**: lỗi không xác định. **ROW**

Khi Distance Matrix API trả về kết quả, thông tin nằm trong một tập các thẻ row. Nếu không có kết quả tìm thấy sẽ trả về tập rỗng.

Các row được xắp xếp phụ thuộc vào giá trị trong chuỗi request URL. Mỗi row sẽ tương ứng với một điểm gốc, mỗi row gồm nhiều element tương ứng với mỗi điểm đích.

**ELEMENT :**

Mỗi element chứa thông tin cụ thể về khoảng cách giữa hai điểm. Mỗi element chứa các phần tử sau:

* **status**: thông báo trạng thái kết quả.

Giá trị của status ở thẻ element o **OK**: kết quả trả về bình thường. o **NOT\_FOUND**: một điểm nào đó không thể mã hóa địa lý để lấy kinh độ

- vĩ độ. o **ZERO\_RESULTS**: không thể tìm thấy đường đi.  **duration**: ước lượng thời gian di chuyển giữa hai điểm, value có giá trị kiểu số nguyên đơn vị là giây, text mô tả bằng chữ, ngôn ngữ phụ thuộc vào giá trị của language trong request URL.

* **distance**: khoảng cách giữa hai điểm, value có giá trị kiểu số nguyên đơn vị là mét, text mô tả bằng chữ, ngôn ngữ phụ thuộc vào giá trị của language trong request URL.

## **Google Direction API**

Google Direction API là dịch vụ tính toán đường đi giữa hai điểm sử dụng HTTP request. Thông tin đầu vào của điểm đầu, điểm kết thúc, điểm trung gian có thể ở dạng tên xác định địa điểm hay giá trị kinh độ và vĩ độ. Direction API có thể trả về nhiều hướng đi khác nhau sử dụng một tập các điểm trung gian.

Dịch vụ này được thiết kế để tính toán đường đi giữa những vị trí xác định rõ thông tin trên bản đồ Google Map. Vì quá trình tính toán đường đi mất thời gian và phức tạp (phụ thuộc vào khoảng cách hai điểm) nên bất cứ khi nào có thể nên thực hiện công việc tính toán trước và lưu trữ tạm thời kết quả nhận được.

Có thể truyền vào một số tham số giá trị địa chỉ hoặc kinh độ - vĩ độ. Nếu truyền vào địa chỉ, hệ thống Google Direction sẽ chuyển nó sang giá trị kinh độ - vĩ độ để tính toán.

* 1. **Giới hạn sử dụng**

Google Direction API cho phép 2500 truy vấn mỗi ngày. Mỗi lần tính toán hướng đi có thể sử dụng tối đa 9 điểm trung gian. Nếu có nhu cầu cao hơn, có thể sử dụng Google Direction API cho doanh nghiệp với tối đa 100000 truy vấn mỗi ngày và 23 điểm trung gian ở mỗi truy vấn.

Ngoài ra mỗi chuỗi URL truy vấn được giới hạn ở 2048 ký tự trước khi mã hóa. Mội số truy vấn tìm đường có thể chứa thông tin các điểm dọc theo đường đi vì vậy cần lưu ý độ dài của nó.

Kết quả trả về của Google Direction API phải được sử dụng kết hợp với bản đồ Goolge Map. Tức là thông tin phải kết hợp với hiển thị đường đi trên bản đồ. Không nên sử dụng thông tin trả về mà không dùng Google Map.

* 1. **Google Direction request**

**Cấu trúc của chuỗi request URL**:

Chuỗi request URL của Google Diriection có dạng như sau:

*http://maps.googleapis.com/maps/api/directions/output?parameters*

Giá trị của output có thể là:

* **json** (Khuyến khích sử dụng): Kết quả trả về ở dạng file JavaScript Object Notation (JSON).
* **xml**: Kết quả trả về ở dạng file xml.

Có thể sử dụng HTTPS để truy xuất Google Direction API nếu chuỗi truy xuất có bao gồm các thông tin nhạy cảm (Ví dụ: trọa độ của người dùng). *https://maps.googleapis.com/maps/api/directions/output?parameters*

**Các tham số của chuỗi URL request :**

Một số tham số là bắt buộc, một số là tùy chọn. Được phân cách bằng ký tự

“&”. Dưới đây là danh sách các tham số có thể tham gia vào một chuỗi Google Direction URL:

* **origin** (bắt buộc): Địa chỉ hay giá trị kinh độ - vĩ độ của điểm mà từ đó bạn muốn tìm đường đi.
* **destination** (bắt buộc): Địa chỉ hay giá trị kinh độ - vĩ độ của điểm muốn tìm đường đi đến.
* **mode** (bắt buộc, mặc định là driving): xác định kiểu duy chuyển (đi bộ, lai xe) khi tính toán hướng đi.
* **waypoints** (tùy chọn): xác định một tập hợp điểm trung gian mà đường đi phải đi qua. Tham số cho điểm trung gian có thể là địa chỉ hay kinh độ - vĩ độ.
* **alternatives** (tùy chọn): nếu được xác định là true, Google Direction sẽ trả về nhiều hơn một tuyến đường để lựa chọn. Tuy nhiên thời gian trả lời của Google Direction sẽ nhiều hơn.
* **avoid** (tùy chọn): chỉ ra tuyến đường sẽ phải tránh đường cao tốc hoăc tránh trạm thu phí. Hiện tại Google Direction chỉ hỗ trợ hai tham số: o **tolls**: tránh các trạm thu phí.

o **highways**: tránh đường cao tốc.

* **units**(tùy chọn): xác định đơn vị cho kết quả tính toán.
* **region**(tùy chọn): mã vùng, bao gồm 2 ký tự xác định vùng.
* **sensor**(bắt buộc): cho biết yêu cầu tìm đường có phải đến từ một thiết bị có cảm biến vị trí hay không. True hoặc false.

**Kiểu di chuyển :**

Khi tính toán hướng đi, có thể qui định kiểu duy chuyển. Kiểu mặc định khi không có tham số trong request url là driving (lái xe hơi). Hiện tại tham số cho kiểu duy chuyển hỗ trợ 3 giá trị:

* **driving** (mặc định): lái xe hơi.
* **walking** : đi bộ, có thể qua những con đường cho người đi bộ hoặc đi trên đường tắt và vỉa hè.
* **bicycling**: lái xe đạp, chọn đường dành riêng cho xe đạp (chỉ phổ biến ở Mỹ)

Hướng dành cho đi bộ và đi xe đạp đôi khi không bao gồm đường chỉ dẫn rỏ ràng cho người đi bộ và đi xe đạp, vì vậy Google Direction cung cấp thông tin cảnh báo trong **kết quả trả về và phải hiển thị cho người dùng.**

**Sử dụng điểm trung gian :**

Khi tính toán tuyến đường đi sử dụng Google Direction API, sử dụng các điểm trung gian để tính toán đường đi đến đích có đi qua những điêm trung gian đã qui định.

Có thể sử dụng nhiều điểm trung gian, mỗi điểm phân cách nhau bằng ký tự “|”.

Ví dụ dưới đây là chuỗi request URL để tìm đường đi từ tháp Trầm Hương đến trường ĐH Nha Trang đi qua cầu Hà Ra:

*http://maps.googleapis.com/maps/api/directions/json?origin=Boston,MA&destination =12.268743,109.201986&waypoints=12.260722,109.190088&sensor=false*

Mặc định Google Direction sẽ tính toán đường đi qua các điểm trung gian dựa trên sự sắp xếp địa điểm trên request URL. Tuy nhiên có thể sử dụng tham số optimize:true để Google Direction xắp xếp các điểm trung gian tối ưu nhất (là một ứng dụng của bài toán Travelling Salesman Problem).

**Vùng giới hạn :**

Google Direction sẽ tính toán đường đi với những giới hạn được đặt ra. Dựa vào những giới hạn đó Google Direction sẽ tính được đường đi thuận lợi hơn cho người dùng. Hiện tại có 2 giá trị của giới hạn được hỗ trợ đó là:

* avoid=tolls : tránh trạm thu phí.
* avoid=highways: tránh đường cao tốc.

**Đơn vị hệ thống :**

Trong kết quả Google Direction trả về có chứa thông tin độ dài từng chặn đường và tuyến đường. Mặc định đơn vị độ dài được sử dụng sẽ phụ thuộc vào vùng (qui định bởi giá trị region).

Ví dụ: tìm đường từ Chicago, IL đến Toronto,ONT sẽ trả về giá trị khoảng cách ở đơn vị dặm (miles), trong khi tìm đường ở Việt Nam kết quả sẻ ở đơn vị ki lô mét (km).

Có thể thay đổi đơn vị khoảng cách mà hệ thống đã lựa chọn bằng cách cung cấp giá trị cho tham số units. Có hai giá trị của tham số units là:

* metric: đơn vị hệ mét.
* imperial : dùng hệ thống đơn vị của Anh.

Lựa chọn đơn vị độ dài chỉ có hiệu lực đối với dòng ký tự mô tả khoảng cách, trá trị khoảng cách trả về ở dạng số luôn có đơn vị là mét.

**Vùng giới hạn địa lý :**

Có thể qui định cho Goolgle Direction trả về kết quả ở những vùng địa lý riêng biệt bằng cách qui định qía trị cho tham số “region”. Giá trị của tham số “region” là tên miền quốc gia (country code top-level domain). Tên miền quốc gia của Việt Nam là “vn”.

Việc đặt giá trị để qui định vùng riêng biệt se giúp cho Google Direction tìm kiếm nhanh và chính xác hơn.

* 1. **Kết quả trả về Google Direction**

**Cấu trúc kết quả trả về :**

Kết quả Google Direction trả về sẽ ở 2 dạng tập tin là xml và json.

Mặc định chỉ có một giá trị route nhận được trong kết quả trả về, trừ khi muốn kết quả trả về hơn hai tuyến đường (alternatives=true).

Kết quả trả về cần phải được phân tích cú pháp để sử dụng. Ví dụ tìm đường đi từ Tháp Bà đến trường Đại học Nha Trang

**Kết quả trả về ở định dạng JSON:**

*{*

*"routes" : [*

*{*

*"bounds" : {*

*"northeast" : {*

*"lat" : 12.270420,*

*"lng" : 109.202090*

*},*

*"southwest" : {*

*"lat" : 12.265350,*

*"lng" : 109.196220*

*} },*

*"copyrights" : "Dữ liệu bản đồ ©2012 Google",*

*"legs" : [*

*{*

*"distance" : {*

*"text" : "1,1 km",*

*"value" : 1122*

*},*

*"duration" : {*

*"text" : "3 phút",*

*"value" : 161*

*},*

*"end\_address" : "2 Nguyễn Đình Chiểu, Đại học Nha Trang, Vĩnh Thọ, Nha Trang, Khánh Hòa, Việt Nam",*

*"end\_location" : {*

*"lat" : 12.268710,*

*"lng" : 109.202090*

*},*

*"start\_address" : "126 2 Tháng 4, Vĩnh Thọ, Nha Trang, Khánh Hòa, Việt*

*Nam",*

*"start\_location" : {*

*"lat" : 12.265350,*

*"lng" : 109.196220*

*},*

*"steps" : [*

*{*

*"distance" : {*

*"text" : "0,6 km",*

*"value" : 601*

*},*

*"duration" : {*

*"text" : "1 phút",*

*"value" : 43*

*},*

*"end\_location" : {*

*"lat" : 12.270420,*

*"lng" : 109.197950*

*},*

*"html\_instructions" : "Đi về hướng \u003cb\u003eBắc\u003c/b\u003e lên \u003cb\u003e2 Tháng 4/Cầu Xóm Bóng\u003c/b\u003e về phía*

*\u003cb\u003eTháp Bà\u003c/b\u003e\u003cdiv style=\"font-size:0.9em\"\u003eĐi tiếp tục theo 2 Tháng 4\u003c/div\u003e\u003cdiv style=\"font-*

*size:0.9em\"\u003eBăng qua Tháp Bà Ponagar (ở phía bên trái)\u003c/div\u003e",*

*"polyline" : {*

*"points" :*

*"mqziAkknySICqAKyAQqC]gAOmBWe@Ko@Qu@[k@U]OUMoB}@gAe@o@Uc@I"*

*},*

*"start\_location" : {*

*"lat" : 12.265350,*

*"lng" : 109.196220*

*},*

*"travel\_mode" : "DRIVING"*

*},*

*{*

*"distance" : {*

*"text" : "0,3 km",*

*"value" : 338*

*},*

*"duration" : {*

*"text" : "1 phút",*

*"value" : 86*

*},*

*"end\_location" : {*

*"lat" : 12.268840,*

*"lng" : 109.200510*

*},*

*"html\_instructions" : "Rẽ \u003cb\u003ephải\u003c/b\u003e tại Trường Trung Học Kinh Tế lên \u003cb\u003eNguyễn Đình Chiểu\u003c/b\u003e",*

*"polyline" : {*

*"points" :*

*"cq{iAevnySP\_@N\_@Zm@Zk@LUDODM@MBM@OBMBSLSRMXUr@g@FKLQ\\s*

*@Te@AK"*

*},*

*"start\_location" : {*

*"lat" : 12.270420,*

*"lng" : 109.197950*

*},*

*"travel\_mode" : "DRIVING"*

*},*

*{*

*"distance" : {*

*"text" : "0,2 km",*

*"value" : 183*

*},*

*"duration" : {*

*"text" : "1 phút",*

*"value" : 32*

*},*

*"end\_location" : {*

*"lat" : 12.268710,*

*"lng" : 109.202090*

*},*

*"html\_instructions" : "Chếch sang \u003cb\u003ephải\u003c/b\u003e tại*

*Uân Loan\u003cdiv style=\"font-size:0.9em\"\u003eĐiểm đến sẽ ở bên phải\u003c/div\u003e",*

*"polyline" : {*

*"points" : "gg{iAefoyS@CJMJKJOFMFKCOCIEICK?K?UEsACgA?E"*

*},*

*"start\_location" : {*

*"lat" : 12.268840,*

*"lng" : 109.200510*

*},*

*"travel\_mode" : "DRIVING"*

*}*

*],*

*"via\_waypoint" : []*

*}*

*],*

*"overview\_polyline" : {*

*"points" :*

*"mqziAkknyS{AOkFo@uDg@uA]aBq@s@]wDcBo@Uc@I`@\_Av@yARe@F[L\_ALS*

*RMlA}@T]r@yA?OVYR]FKCOISCWIwD"*

*},*

*"summary" : "2 Tháng 4 và Nguyễn Đình Chiểu",*

*"warnings" : [],*

*"waypoint\_order" : []*

*}*

*],*

*"status" : "OK"*

*}*

**Kết quả trả về ở định dạng XML:**

*<?xml version="1.0" encoding="UTF-8"?>*

*<DirectionsResponse>*

*<status>OK</status>*

*<route>*

*<summary>2 Tháng 4 và Nguyễn Đình Chiểu</summary>*

*<leg>*

*<step>*

*<travel\_mode>DRIVING</travel\_mode>*

*<start\_location>*

*<lat>12.2653500</lat>*

*<lng>109.1962200</lng>*

*</start\_location>*

*<end\_location>*

*<lat>12.2704200</lat>*

*<lng>109.1979500</lng>*

*</end\_location>*

*<polyline>*

*<points>mqziAkknySICqAKyAQqC]gAOmBWe@Ko@Qu@[k@U]OUMoB}@gAe@o*

*@Uc@I</points>*

*</polyline>*

*<duration>*

*<value>43</value>*

*<text>1 phút</text>*

*</duration>*

*<html\_instructions>Đi về hướng &lt;b&gt;Bắc&lt;/b&gt; lên &lt;b&gt;2 Tháng 4/Cầu Xóm Bóng&lt;/b&gt; về phía &lt;b&gt;Tháp Bà&lt;/b&gt;&lt;div style=&quot;fontsize:0.9em&quot;&gt;Đi tiếp tục theo 2 Tháng 4&lt;/div&gt;&lt;div style=&quot;fontsize:0.9em&quot;&gt;Băng qua Tháp Bà Ponagar (ở phía bên trái)&lt;/div&gt;</html\_instructions>*

*<distance>*

*<value>601</value>*

*<text>0,6 km</text>*

*</distance>*

*</step>*

*<step>*

*<travel\_mode>DRIVING</travel\_mode>*

*<start\_location>*

*<lat>12.2704200</lat>*

*<lng>109.1979500</lng>*

*</start\_location>*

*<end\_location>*

*<lat>12.2688400</lat>*

*<lng>109.2005100</lng>*

*</end\_location>*

*<polyline>*

*<points>cq{iAevnySP\_@N\_@Zm@Zk@LUDODM@MBM@OBMBSLSRMXUr@g@*

*FKLQ\s@Te@AK</points>*

*</polyline>*

*<duration>*

*<value>86</value>*

*<text>1 phút</text>*

*</duration>*

*<html\_instructions>Rẽ &lt;b&gt;phải&lt;/b&gt; tại Trường Trung Học Kinh Tế lên*

*&lt;b&gt;Nguyễn Đình Chiểu&lt;/b&gt;</html\_instructions>*

*<distance>*

*<value>338</value>*

*<text>0,3 km</text>*

*</distance>*

*</step>*

*<step>*

*<travel\_mode>DRIVING</travel\_mode>*

*<start\_location>*

*<lat>12.2688400</lat>*

*<lng>109.2005100</lng>*

*</start\_location>*

*<end\_location>*

*<lat>12.2687100</lat>*

*<lng>109.2020900</lng>*

*</end\_location>*

*<polyline>*

*<points>gg{iAefoyS@CJMJKJOFMFKCOCIEICK?K?UEsACgA?E</points>*

*</polyline>*

*<duration>*

*<value>32</value>*

*<text>1 phút</text>*

*</duration>*

*<html\_instructions>Chếch sang &lt;b&gt;phải&lt;/b&gt; tại Uân Loan&lt;div style=&quot;font-size:0.9em&quot;&gt;Điểm đến sẽ ở bên phải&lt;/div&gt;</html\_instructions>*

*<distance>*

*<value>183</value>*

*<text>0,2 km</text>*

*</distance>*

*</step>*

*<duration>*

*<value>161</value>*

*<text>3 phút</text>*

*</duration>*

*<distance>*

*<value>1122</value>*

*<text>1,1 km</text>*

*</distance>*

*<start\_location>*

*<lat>12.2653500</lat>*

*<lng>109.1962200</lng>*

*</start\_location>*

*<end\_location>*

*<lat>12.2687100</lat>*

*<lng>109.2020900</lng>*

*</end\_location>*

*<start\_address>126 2 Tháng 4, Vĩnh Thọ, Nha Trang, Khánh Hòa, Việt Nam</start\_address>*

*<end\_address>2 Nguyễn Đình Chiểu, Đại học Nha Trang, Vĩnh Thọ, Nha Trang, Khánh Hòa, Việt Nam</end\_address>*

*</leg>*

*<copyrights>Dữ liệu bản đồ ©2012 Google</copyrights>*

*<overview\_polyline>*

*<points>mqziAkknyS{AOkFo@uDg@uA]aBq@s@]wDcBo@Uc@I`@\_Av@yARe@F*

*[L\_ALSRMlA}@T]r@yA?OVYR]FKCOISCWIwD</points>*

*</overview\_polyline>*

*<bounds>*

*<southwest>*

*<lat>12.2653500</lat>*

*<lng>109.1962200</lng>*

*</southwest>*

*<northeast>*

*<lat>12.2704200</lat>*

*<lng>109.2020900</lng>*

*</northeast>*

*</bounds>*

*</route>*

*</DirectionsResponse>*

Nút gốc trong tài liệu Xml trả về có 2 thẻ con là:

* **status** :thông báo trạng thái của kết quả hoặc thông tin sự cố .
* **route** :(một hay nhiều), mỗi thẻ route chứa một tuyến đường đi riêng biệt từ điểm đầu đến điểm cuối. **STATUS**
* **OK**: kết quả trả về bình thường.
* **NOT\_FOUND**: một điểm nào đó nhập vào không thể mã hóa địa lý để lấy kinh độ - vĩ độ.
* **ZERO\_RESULTS**: không thể tìm thấy đường đi.
* **MAX\_WAYPOINTS\_EXCEEDED**: thông báo vượt quá giới hạn các

điểm trung gian được cho phép. (Bình thường là 8 điểm, nếu trả thêm chi phí sẽ tăng lên tối đa là 23 điểm trung gian)

* **INVALID\_REQUEST**: thông báo chuỗi yêu cầu không hợp lệ.
* **OVER\_QUERY\_LIMIT**: thông báo đã gởi quá số lượng yêu cầu trong một khoảng thời gian.
* **REQUEST\_DENIED**: ứng dụng đã bị dịch vụ Google Direction chặn.
* **UNKNOWN\_ERROR**: lỗi không xác định. **ROUTE**

Mỗi thẻ route chứa thông tin về một tuyến đường phân biệt từ điểm xuất phát đến điểm cần đến. Trong kết quả trả về của Google Direction sẽ có nhiều thẻ route hoặc sẽ không có nếu không thể tìm được đường đi.

Trong mỗi thẻ route sẽ có một hay nhiều thẻ leg phụ thuộc vào số lượng điểm trung gian đã yêu cầu. Ngài ra, bên trong thẻ rout> còn chứa các thông tin cảnh báo, bản quyền …

Trong mỗi thẻ route có thể bao gồm các thẻ con sau:

* **summary**: tên tóm tắc của đoạn đường.
* **legs**[]: chứa một tập hợp các thẻ <leg> mô tả đường đi giữa hai điểm. Nếu không yêu cầu điểm trung gian thì kết quả trả về chỉ có một thẻ <leg> tương ứng với đường đi từ điểm xuất phát đến điểm đích. Nếu có một hoặc một số điểm trung gian được yêu cầu thì mỗi thẻ <leg> sẽ tương ứng với đoạn đường từng cặp điểm lần lược là: điểm xuất phát đến điểm trung gian thứ nhất, điểm trung gian thứ nhất đến điểm trung gian thứ hai,…điểm trung gian cuối đến điểm đích.
* **waypoint\_order**: chứa một mảng qui định thứ tự xắp xếp của các điểm trung gian.
* **overview\_polyline**: chứa một đối tượng giữ một mảng các điểm đã mã hóa đại diện cho một con đường trả về trong kết quả.
* **bounds**: qui định hình chữ nhật bao chung quanh tuyến đường trên bản đồ.
* **copyrights**: chứa thông tin bản quyền.
* **warnings**[] : chứa một tập các cảnh báo cho người dùng.

**LEG :**

Trong mỗi thẻ leg sẽ mô tả một đoạn đường đi giữa hai điểm.Trong những yêu cầu tìm đường không có điểm trung gian sẽ chỉ có một thẻ leg ở mỗi thẻ route. Trong trường hợp có yêu cầu một số điểm trung gian thì mỗi thẻ leg sẽ tương ứng với mỗi đoạn đường tương ứng với từng điểm trung gian.

Trong mỗi thẻ leg có thể bao gồm các thẻ con sau:

* **steps**[]: chứa thông tin từng đoạn đường.
* **distance**: thông tin về độ dài chặn đường được mô tả trong thẻ <leg>. Chứa 2 thẻ con: o **value**: độ dài kiểu số nguyên, đơn vị mét.
  + **text**: mô tả bằng chữ độ dài của đoạn đường, đơn vị độ dài sử dụng phụ thuộc vào vùng địa ký đã thiết lập trong chuỗi url yêu cầu tìm đường, hoặc có thể thiết lập riêng bằng giá trị chủa units.

Thẻ này có thể không có nếu không thể tính toán được khoảng cách.

* **duration**: ước lượng thời gian di chuyển hết chặn đường tương ứng với thẻ leg
  + **value**: giá trị số nguyên đơn vị giây. o **text**: thời gian ước lượng bằng chữ.

Thẻ này có thể không có nếu không thể tính toán được thời gian.

* **start\_location**: chứa kinh độ - vĩ độ điểm đầu của chặn đường tương ứng với thẻ leg. Bởi vì Direction API sẽ chọn một điểm gần với điểm gốc nhất nhưng trên đường giao thông để tính toán đường đi nên tọa độ điểm này có thể không trung với tọa độ điểm gốc(end\_location cũng tương tự như vậy).
* **end\_location**: chứa kinh độ - vĩ độ điểm cuối của chặn đường tương ứng với thẻ leg.
* **start\_address**: địa chỉ điểm đầu bằng chữ tương ứng với thẻ <leg>.
* **end\_addresss**: địa chỉ điểm cuối bằng chữ tương ứng với thẻ <leg>.

**STEP :**

Mỗi thẻ step mô tả một đoạn đường. Đoạn đường thẻ step qui định là “đơn vị” nhất và không cần cần phải tách thêm nữa. Thông thường Google Direction tách ra thành mỗi step khi cần rẽ. Nội dung của thẻ step mô tả độ dài, ước lượng thời gian di chuyển, chỉ đường bằng chữ…

Trong mỗi thẻ leg có thể bao gồm các thẻ con sau:

* **html\_instructions**: hướng dẫn cho đoạn đường bằng HTML.
* **distance** :mô tả độ dài của đoạn đường tương ứng với thẻ <step>.

Chứa 2 thẻ con:

* + value: độ dài kiểu số nguyên, đơn vị mét.
  + text: mô tả bằng chữ độ dài của đoạn đường, đơn vị độ dài sử dụng phụ thuộc vào vùng địa ký đã thiết lập trong chuỗi url yêu cầu tìm đường, hoặc có thể thiết lập riêng bằng giá trị chủa units.

Thẻ này có thể không có nếu không thể tính toán được khoảng cách.  duration: ước lượng thời gian di chuyển hết chặn đường tương ứng với thẻ step

* + value: giá trị số nguyên đơn vị giây. o text: thời gian ước lượng bằng chữ.

Thẻ này có thể không có nếu không thể tính toán được thời gian.

* **start\_location**: chứa kinh độ - vĩ độ điểm đầu của chặn đường tương ứng với thẻ leg. Bởi vì Direction API sẽ chọn một điểm gần với điểm gốc nhất nhưng trên đường giao thông để tính toán đường đi nên tọa độ điểm này có thể không trung với tọa độ điểm gốc (end\_location cũng tương tự như vậy).
* **end\_location**: chứa kinh độ - vĩ độ điểm cuối của chặn đường tương ứng với thẻ leg.

1. **XÂY DỰNG ỨNG DỤNG**
   1. **Giới thiệu đề tài**

Dựa trên ý tưởng của grab là khách muốn đi taxi/bike thì sẽ gọi qua ứng dụng mobile và mỗi tài xế sẽ có 1 thiết bị (điện thoại di động) để có thể nhận thông báo từ tổng đài về khách hàng: vị trí hiện tại, địa điểm muốn đến, số km…

Khi tài xế tham gia giao thông, từ thời gian và quãng đường mà tài xế đã để tính toán vận tốc hiện tại của phương tiện. Khi tổng hợp được lượng dữ liệu nhất định có thể tính toán lưu lượng xe trên đoạn đường đó và biết được đoạn đường đó có bị ùn tắc, kẹt xe hay không.

Thông tin giao thông sẽ được cập nhập liên tục cho các tài xế, để có thể lựa chọn lộ trình thích hợp, tránh các địa điểm đang có lưu lượng đông, góp phần giảm thiểu tình trạng ùn tắc, kẹ xe đang nhức nhối hiện nay.

* 1. **Đặc tả use case**

**Danh sách usecase**

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên Use-case** | **Ý nghĩa/Ghi chú** |
| 1 | Tìm lộ trình đến địa điểm cụ thể | Người dùng nhập địa chỉ muốn đến, ứng dụng sẽ tìm kiếm tuyến đường thích hợp để hiển thị lên map. |
| 2 | Cung cấp thông tin giao thông theo thời gian thực | Liên tục nhận thông tin giao thông từ server gửi lên để hiển thị thông báo cho người dùng. |
| 3 | Gửi thông tin phương tiện cho server | Liên tục gửi thông tin phương tiện (vận tốc, vị trí) lên server để tổng hợp tính toán tình hình giao thông. |
| 4 | Server tổng hợp dữ liệu người dùng | Liên tục (5 phút một lần) tổng hợp dữ liệu của nhiều phương tiện, tính toán vận tốc trung bình của từng đoạn đường. |

**Đặc tả use case**

### **Tìm lộ trình đến địa điểm cụ thể**

|  |
| --- |
| ***Tên UseCase:*** Tìm lộ trình đến địa điểm cụ thể |
| ***Tóm tắt:*** UseCase bắt đầu khi người sử dụng tiến hành nhập địa điểm muốn tới. Ứng dụng sẽ hiển thị lộ trình thích hợp đi từ vị trí hiện tại đến địa điểm đó. |
| ***Dòng sự kiện chính:***   1. Hệ thống hiển thị giao diện để người dùng nhập địa điểm 2. Người sử dụng nhập địa điểm cần đến vào. 3. Hệ thống sử dụng **direction API** để tìm kiếm lộ trình. 4. Hệ thống hiển thị giao diện lộ trình từ vị trí hiện tại của thiết bị đến vị trí được nhập. 5. Chuyển camera về vị trí hiện tại của thiết bị. |
| ***Dòng sự kiện phụ:***   1. Hệ thống xử lý khi không tìm được địa điểm thích hợp theo từ khóa (hiển thị hộp thoại thông báo, yêu cầu nhập lại địa điểm) |
| ***Yêu cầu đặc biệt:*** Thiết bị đang có kết nối internet và bật GPS. |
| ***Trạng thái hệ thống khi bắt đầu thực hiện UseCase:*** hệ thống đang trong trạng thái đang hoạt động |
| ***Trạng thái hệ thống khi sau khi thực hiện UseCase:*** hệ thống đăng trong trạng thái hiển thị lộ trình từ vị trí hiện tại đến vị trí người sử dụng điền vào. |
| ***Điểm mở rộng:*** không có |

### **Cung cấp thông tin giao thông theo thời gian thực**

|  |
| --- |
| ***Tên UseCase:*** Cung cấp thông tin giao thông theo thời gian thực |
| ***Tóm tắt:*** UseCase bắt đầu khi người sử dụng tiến hành nhập địa điểm muốn tới. Hệ thống sẽ gửi yêu cầu lấy dữ liệu giao thông trên lộ trình hiện tại hiển thị cho người dùng. |
| ***Dòng sự kiện chính:***   1. Người sử dụng nhập địa điểm muốn đến và hệ thống tìm ra lộ trình thích hợp. 2. Hệ thống lấy các con đường mà lộ trình đi qua thông qua các điểm trong poliline lấy từ **direction API**. 3. Hệ thống gửi thông tin những con đường mà lộ trình đi qua lên server. 4. Server nhận thông tin các con đường và truy xuất lấy dữ liệu giao thông tương ứng trong cơ sở dữ liệu trả về cho client. 5. Client nhận dữ liệu giao thông từ server hiển thị cho người dùng và cảnh báo những địa điểm có tình trạng giao thông xấu (vận tốc thấp) để người dùng có thể lựa chọn lộ trình thích hợp. |
| ***Dòng sự kiện phụ:***   1. Hệ thống xử lý khi không tìm được không truy cập được vào internet; thông báo tình trạng. |
| ***Yêu cầu đặc biệt:*** Không có |
| ***Trạng thái hệ thống khi bắt đầu thực hiện UseCase:*** không có |
| ***Trạng thái hệ thống khi sau khi thực hiện UseCase:*** không có |
| ***Điểm mở rộng:*** không có |

### **Gửi thông tin phương tiện cho server**

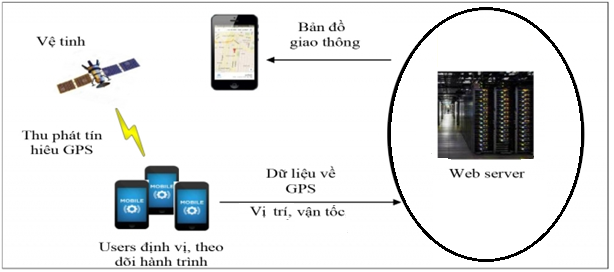
|  |
| --- |
| ***Tên UseCase:*** Gửi thông tin phương tiện cho server |
| ***Tóm tắt:*** UseCase bắt đầu khi người sử dụng nhấp vào button bắt đầu. Ứng dụng sẽ liên tục (1 phút một lần) gửi dữ liệu hiện tại của thiết bị về server. |
| ***Dòng sự kiện chính:***   1. Người sử dụng nhấp vào button bắt đầu. 2. Hệ thống sử dụng **Geocoding API** để lấy thông tin địa điểm hiện tại của thiết bị theo GPS. 3. Hệ thống sử dụng **Google Distance API** để lấy khoản cách giữa hai điểm vị trí của thiết bị hiện tại và vị trí lần trước. 4. Dựa vào khoản cách và thời gian giữa hai lần lấy vị trí (1 phút) để tính toán vận tốc của phương tiện. 5. Gửi dữ liệu phương tiện (thông tin địa điểm, vận tốc) lên server. 6. Lặp lại quá trình trong khoản thời gian cố định (1 phút). |
| ***Dòng sự kiện phụ:***   1. Hệ thống xử lý khi không tìm được vị tí hiện tại của GPS hoặc không truy cập được vào internet; thông báo tình trạng cho người dùng. |
| ***Yêu cầu đặc biệt:*** Thiết bị đang có kết nối internet và bật GPS. |
| ***Trạng thái hệ thống khi bắt đầu thực hiện UseCase:*** trạng thái đang hiển thị lộ trình cho người dùng và trạng thái gửi dữ liệu lên server đang ngừng hoạt động |
| ***Trạng thái hệ thống khi sau khi thực hiện UseCase:*** hệ thống đăng trong trạng thái hiển thị lộ trình và trạng thái gửi dữ liệu lên server đang hoạt động. |
| ***Điểm mở rộng:*** không có |

### **Server tổng hợp dữ liệu người dùng**

|  |
| --- |
| ***Tên UseCase:*** Server tổng hợp dữ liệu người dùng |
| ***Tóm tắt:*** UseCase liên tục thực hiện trên server (5 phút một lần), liên tục tổng hợp dữ liệu được gửi lên từ server để tính toán tình trạng giao thông hiện tại cho từng đoạn đường. |
| ***Dòng sự kiện chính:***   1. Server liên tục thực hiện (5 phút một lần). 2. Server tổng hợp dữ liệu vận tốc theo từng con đường, tính vận tốc trung bình lưu lại vào cơ sở dữ liệu. 3. Xóa rác bản lưu dữ liệu tạm thời nhận từ người dùng |
| ***Dòng sự kiện phụ:***   1. Hệ thống xử lý khi không tìm được vị tí hiện tại của GPS hoặc không truy cập được vào internet; thông báo tình trạng cho người dùng. |
| ***Yêu cầu đặc biệt:*** Không có |
| ***Trạng thái hệ thống khi bắt đầu thực hiện UseCase:*** không có |
| ***Trạng thái hệ thống khi sau khi thực hiện UseCase:*** không có |
| ***Điểm mở rộng:*** không có |

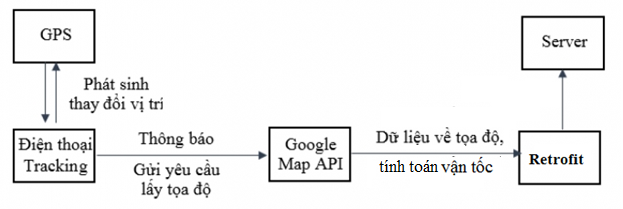
* 1. **Mô hình hoạt động**

1. **Quá trình lưu trữ và xử lý dữ liệu**



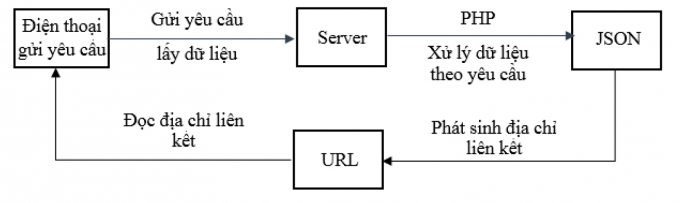
*Quá trình lưu trữ và xử lý dữ liệu*

1. **Quá trình gửi dữ liệu lên server**



*Quy trình quá trình gửi dữ liệu lên server*

1. **Quá trình nhận dữ liệu từ server**

*sơ đồ quá trình tải dữ liệu từ server*

* 1. **Thiết kế màn hình ứng dụng**



*Giao diện ứng dụng*

**Chú thích màu đường**

|  |  |  |
| --- | --- | --- |
| **STT** | **Màu** | **Ý nghĩa** |
| 1 | Xanh dương | Không có dữ liệu |
| 2 | Xanh lá | Vận tốc trung bình >=40km/h |
| 3 | Màu vàng | Vận tốc trung bình <40km/h |
| 4 | Màu đỏ | Vận tốc trung bình <20km/h |

**Các thành phần giao diện**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên Control** | **Kiểu** | **Ý nghĩa** |
| 1 | Địa điểm | Textbox | Địa điểm muốn đến |
| 2 | Tìm kiếm | Button | Tìm lộ trình |
| 3 | Bắt đầu | Button | Bắt đầu gửi dữ liệu |
| 4 | Dừng lại | Button | Kết thúc gửi dữ liệu |

**Danh sách các biến cố**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Điều kiện kích hoạt** | **Xử lý** | **Ghi chú** |
| 1 | Khởi động chương trình | Khởi động chương trình cho phép người dùng tìm kiếm địa điểm, lộ trình |  |
| 2 | Người dùng nhập địa điểm cần đến | Cho phép người dùng nhập địa điểm |  |
| 3 | Click vào Tìm kiếm | Tìm kiếm vị địa điểm người dùng nhập vào và lộ trình đến đó. |  |
| 4 | Click vào Bắ đầu | Bắt đầu trạng thái gửi dữ liệu của người dùng lên server. |  |
| 5 | Click vào kết thúc | Kết thúc trạng thái gửi dữ liệu của người dùng lên server |  |

1. **TỔNG KẾT**
   1. **Đánh giá hệ thống đã xây dựng**

* **Ưu điểm**
  + Hệ thống cơ bản thể hiện được ý tưởng dựa vào GSP của thiết bị người tham gia giao thông để để nắm bắt thông tin ùn tắc giao thông hiện tại.
* **Khuyết điểm**
  + Vẫn còn thiếu sót nhiều chức năng chưa phát triển.
  + Giao diện người dùng chưa được bắt mắt.
  + Chưa có cơ chế cập nhật dữ liệu thông qua mạng, muốn cập nhật dữ liệu phải cập nhật cả phần mềm.
  + Chưa có cơ chế sao lưu dữ liệu người dùng, nếu xóa phần mềm, dữ liệu người dùng sẽ mất.
  1. **Hướng phát triển**
     + **Thêm chức năng tương tác với người dùng:**

Quản lý người dùng

Tiếp nhận và phản hồi trực tuyến các phản ánh của người dân về những bất cập, ùn tắc, tai nạn tai nạn giao thông.

**Cập nhập tiện ích giao thông trên đường:**

Cập nhập tình hình thời tiết, nhiệt độ; các điểm giao thông như thanh tra giao thông, trạm thu phí ...

**Dự doán ùn tắc giao thông**

Dựa vào dữ liệu giao lưu lượng giao thông hiện tại, có thể dự đoán mật độ giao thông trong tương lai để đưa ra các dự đoán về tình hình giao thông, giúp tránh ùn tắc, kẹt xe.

* 1. **Kết quả đạt được**
  + **Về mặt lý thuyết:**
    - Có được các kiến thức về Mobile Pervasive Computing.
    - Có được các kiến thức về lập trình Javascript về công nghệ React Native.
    - Nắm vững quy trình phát triển một web server theo giao thức RESTful API bằng ngôn ngữ PHP.
    - Hiểu sâu hơn về các tầng hoạt động của hệ điều hành Android, các thư viện hỗ trợ không thể thiếu của một ứng dụng Android như Dagger, Retrofit và về việc xây dựng đa nền tảng.
    - Rèn luyện kỹ năng nghiên cứu tài liệu, kỹ năng làm việc nhóm, kỹ năng giao tiếp trong công việc.
  + **Về phần kỹ năng**
    - Thiết kế giao diện trên Android.
    - Lập trình Java trên nền tảng Android.
    - Sử dụng Google API Webservice
    - Sử dụng Google Map
    - Sử dụng JSON
    - Kiến thức hệ thông thông tin địa lý GIS
    - Xây dựng được ứng dụng quản lý trên Android
  1. **Tài liệu tham khảo**
* Các tài liệu sử dụng:

*The Android mobile platform - Benjamin Speckmann*

*Tìm hiểu các Layout cơ bản trong Android - Nguyễn Văn Kiên St*

*Android Application Development - Perth, Western Australia, Reto Meier*

1. Các đường link tham khảo:

[*http://www.baomoi.com/The-gioi-nhay-vot-ve-cong-nghe-thongtin/76/5125452.epi*](http://www.baomoi.com/The-gioi-nhay-vot-ve-cong-nghe-thongtin/76/5125452.epi)

[*http://www.pcworld.com.vn/articles/quan-ly/chuyenmuc/2011/03/1224392/thoi-dai-hau-pc-va-muc-tieu-cho-nganh-cntt/*](http://www.pcworld.com.vn/articles/quan-ly/chuyenmuc/2011/03/1224392/thoi-dai-hau-pc-va-muc-tieu-cho-nganh-cntt/)

[*http://sinhvienit.net/@tut/co-so-du-lieu/xml/67-bai-1-tim-hieu-cau-truc-vacu-phap-cua-xml.html*](http://sinhvienit.net/@tut/co-so-du-lieu/xml/67-bai-1-tim-hieu-cau-truc-vacu-phap-cua-xml.html)

[*http://congdongjava.com/showthread.php?p=8905*](http://congdongjava.com/showthread.php?p=8905)

[*http://developer.android.com*](http://developer.android.com)

*http:// vietandroid.com*

[*http://stackoverflow.com*](http://stackoverflow.com)

[*http://www.dofactory.com*](http://www.dofactory.com)

[*http://genk.vn*](http://genk.vn)

*https://developers.google.com/maps/documentation/webservices*