**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

**KHÓA LUẬN TỐT NGHIỆP**

**XÂY DỰNG HỆ THỐNG QUẢN LÝ TAXI**

**MGTS-TAXI**

Giảng viên hướng dẫn: **TS. ĐÀM QUANG HỒNG HẢI**

Sinh viên thực hiện: **LÊ KIM HÙNG 08520548**

**NGUYỄN THÀNH TRUNG 08520435**

Lớp : **MMT03**

Khóa : **2008-2012**

***TP. Hồ Chí Minh, tháng 02 năm 2013***

**MỞ ĐẦU**

Ngày nay với sự tiến bộ vượt bậc của khoa học công nghệ, cuộc sống của chúng ta ngày càng được thoải mái và thuận tiện hơn. Một trong những công nghệ phổ biến hiện nay đó là công nghệ định vị thông qua hệ thống định vị toàn cầu. Hiện nay có rất nhiều tổ chức quốc gia phát triển hệ thống này mà tiêu biểu như: Hệ thống Beidou của Cộng Hòa Dân Chủ Nhân Dân Trung Hoa, GLONASS của Nga, QZSS của Nhật, Galileo của liên minh châu Âu EU, nhưng đặc biệt nhất là GPS (Global Positioning System – Hệ thống định vị toàn cầu) do bộ quốc phòng Mỹ thiết kế, xây dựng và được triển khai rộng rãi vào năm 1980. Hệ thống GPS đang được sử dụng rất phổ biến trên toàn thế giới nhờ khả năng định vị chính xác và được cung cấp miễn phí hoàn toàn. Ở Việt Nam, hệ thống GPS không còn quá xa lạ. Nó đang được sử dụng để phục vụ trong một số lĩnh vực như: quản lý điều hành xe, tìm đường, du lịch, thám hiểm, thiết lập bản đồ …

Kể từ khi ra đời màn hình cảm ứng đầu tiên – IBM Simon được ra mắt bởi Bell South vào năm 1993, nhưng thời đại của smartphone tại Mỹ thực sự bắt đầu vào năm 2002 khi công nghệ CPA[1] giúp thực hiện cuộc gọi trên smartphone xuất hiện. Năm 2006, smartphone chỉ chiếm 6% thị phần của Mỹ. Tại thời điểm này, so với tốc độ phát triển của PC, tốc độ "phủ sóng" của smartphone ngày càng rộng lớn hơn. Và cho đến nay, Nielsen cho biết smartphone đã chiếm tới 2/3 số lượng điện thoại di động bán ra, dù đang diễn ra cuộc khủng hoảng kinh tế toàn cầu[[1]](#footnote-1). Với sự phát triển mạnh mẽ như vậy, chip xử lý trong smartphone ngày càng được thu nhỏ, rẻ và có hiệu xuất cao giúp giảm giá thành tạo điều kiện cho smartphone được sử dụng rộng rãi hơn nữa ở các nước đang phát triển như Việt Nam.

Tận dụng khả năng định vị của một số thiết bị GPS. Chúng ta có thể lấy thông tin tức thời và chính xác của thiết bị về vị trí, tốc độ di chuyển. Nhờ vào khả năng này, nó có thể được sử dụng vào trong việc đặt và giao hàng. Chẳng hạn như đề án “Giải pháp đặt hàng, giám sát qua công nghệ GPS giành giải thưởng Nhân tài Đất Việt 2013 [[2]](#footnote-2)” ứng dụng GPS để giúp các doanh nghiệp có thể giám sát toàn bộ quá trình INBOUND - OUTBOUND trong chuỗi cung ứng..

Từ thực tế trên, em đã đưa ra giải pháp là tận dụng khả năng định vị GPS được tích hợp trên hầu hết những smartphone vào việc đặt và giao hàng để. Nội dung chính của giải pháp này là kết nối smartphone đến máy chủ website đặt hàng để đặt hàng bằng ứng dụng đồng thời truyền tải thông tin về vị trí dung cho việc giao hàng. Về phía cửa hàng việc sữ dụng giả pháp này để giúp cho việc giao hàng trở nên tiện .

Hệ thống đặt hàng qua điện thoại di độngđược thực hiện trên nền tảng Android, một hệ điều hành dành cho các thiết bị di động được phát triển bởi Google, bên cạnh đó cũng xây dựng một Web Application thực hiện chức năng điều phối, quản lý đơn hàng và web đặt hàng bằng ngôn ngữ PHP.

Luận văn được cấu trúc như sau. Chương 2 nêu khái quát các kiến thức có liên quan được dùng để phục vụ đề tài, đồng thời khảo sát các công việc có liên quan đến đề tài. Chương 3 giới thiệu cách tiếp cận của chúng em để giải quyết vấn đề. Phần hiện thực hệ thống quản lý taxi cũng như đánh giá được trình bày trong chương 4. Trong chương 5, chúng em tổng kết đề tài và nêu ra một số hướng phát triển tiếp theo của đề tài.

TP. Hồ Chí Minh, ngày 17 tháng 06 năm 2015

Sinh viên

Nguyễn Chí Hiếu

**LỜI CẢM ƠN**

Em xin bày tỏ lòng biết ơn chân thành và sâu sắc nhất đến   
TS. Đàm Quang Hồng Hải, người đã tận tình hướng dẫn và tạo mọi điều kiện tốt nhất cho em trong quá trình định hướng, nghiên cứu và xây dựng đề tài này.

Chúng em xin chân thành cảm ơn các thầy cô Khoa Mạng Máy Tính Và Truyền Thông cũng như tất cả các thầy cô, cán bộ nhân viên trường Đại Học Công Nghệ Thông Tin – Đại Học Quốc Gia Thành Phố Hồ Chí Minh đã tận tình giảng dạy, tạo điều kiện tốt nhất cho chúng em trong suốt quá trình học tập tại trường, giúp chúng em học hỏi và tiếp thu được kiến thức một cách vững vàng.

Cuối cùng chúng em xin bày tỏ lòng biết ơn chân thành đến gia đình, người thân và bạn bè đã giúp đỡ, động viên, khích lệ chúng em trong quá trình hoàn thành đề tài này.

TP. Hồ Chí Minh, ngày 17 tháng 06 năm 2015

Sinh viên

Nguyễn Chí Hiếu

**LỜI CAM ĐOAN**

Em xin cam đoan tất cả nội dung trình bày trong khóa luận tốt nghiệp này có được dựa trên việc nghiên cứu, tổng hợp kiến thức lý thuyết và làm việc thực tế của nhóm chúng em. Mọi thông tin trích dẫn đều được chú thích và liệt kê rõ ràng trong danh mục các tài liệu tham khảo.

Chúng em xác nhận khóa luận tốt nghiệp này là sản phẩm của chúng em xây dựng dưới sự hướng dẫn của TS. Đàm Quang Hồng Hải và sự giúp đỡ của những người khác đã được ghi nhận trong báo cáo này.

TP. Hồ Chí Minh, ngày 17 tháng 06 năm 2015

Sinh viên

Nguyễn Chí Hiếu

**NHẬN XÉT**

**CỦA GIẢNG VIÊN HƯỚNG DẪN**

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

…………………………………………………………………………………………

**NHẬN XÉT**

**CỦA GIẢNG VIÊN PHẢN BIỆN**

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

**MỤC LỤC**

[CHƯƠNG 1. GIỚI THIỆU 1](#_Toc349986741)

[1.1 Tên đề tài 1](#_Toc349986742)

[1.2 Nội dung và giới hạn của đề tài 1](#_Toc349986743)

[1.2.1 Nội dung 1](#_Toc349986744)

[1.2.2 Giới hạn 2](#_Toc349986745)

[1.3 Đối tượng, phạm vi và phương pháp nghiên cứu của đề tài 2](#_Toc349986746)

[1.3.1 Đối tượng nghiên cứu 2](#_Toc349986747)

[1.3.2 Phạm vi nghiên cứu 2](#_Toc349986748)

[1.3.3 Phương pháp nghiên cứu 3](#_Toc349986749)

[1.4 Thời gian nghiên cứu 3](#_Toc349986750)

[1.5 Cấu trúc luận văn 4](#_Toc349986751)

[CHƯƠNG 2. CÔNG NGHỆ GPS, HỆ ĐIỀU HÀNH ANDROID VÀ NHỮNG CÔNG NGHỆ SỬ DỤNG TRONG QUÁ TRÌNH XÂY DỰNG HỆ THỐNG 5](#_Toc349986752)

[2.1 Map API 5](#_Toc349986753)

[2.1.1 Google Map API 6](#_Toc349986754)

[2.1.2 Bing Map API 8](#_Toc349986755)

[2.1.3 Yahoo API 9](#_Toc349986756)

[2.2 Hệ điều hành Android 9](#_Toc349986757)

[2.2.1 Lịch sử phát triển Android 9](#_Toc349986758)

[2.2.2 Sự phát triển của Android 10](#_Toc349986759)

[2.2.3 Cấu trúc của Android 13](#_Toc349986760)

[2.2.4 Các thành phần chính của Android 16](#_Toc349986761)

[2.2.5 Google Map API Android 18](#_Toc349986762)

[2.2.6 Google Place API Android 27](#_Toc349986763)

[2.2.7 Google Direction API Android 29](#_Toc349986764)

[2.2.8 SIP Android 30](#_Toc349986765)

[2.3 Công nghệ GPS 31](#_Toc349986766)

[2.3.1 Tổng quan về GPS 32](#_Toc349986767)

[2.3.2 Những phân đoạn GPS 32](#_Toc349986768)

[2.3.3 Những thế hệ của vệ tinh GPS 33](#_Toc349986769)

[2.3.4 Những vị trí điều khiển 34](#_Toc349986770)

[2.3.5 GPS-một vài khái niệm cơ bản 35](#_Toc349986771)

[2.3.6 Nguyên lý hoạt động của GPS 36](#_Toc349986772)

[2.4 Java Server Page và Servlet 37](#_Toc349986773)

[2.4.1 Java Server Page 37](#_Toc349986774)

[2.4.2 Servlet 38](#_Toc349986775)

[2.4.3 So sánh JSP và Servlet 42](#_Toc349986776)

[2.5 RESTful Web Service 42](#_Toc349986777)

[2.5.1 RESTful Web Service 42](#_Toc349986778)

[2.5.2 Các đặc tính của hệ thống RESTful 42](#_Toc349986779)

[2.5.3 Các thành phần của RESTful Web Service 43](#_Toc349986780)

[2.6 MySQL 5.5 và Apache Tomcat 7 44](#_Toc349986781)

[2.6.1 MySQL 5.5 44](#_Toc349986782)

[2.6.2 Apache Tomcat 7 45](#_Toc349986783)

[2.7 JSON và thư viện mã nguồn mở GSON 47](#_Toc349986784)

[2.7.1 Định nghĩa JSON 47](#_Toc349986785)

[2.7.2 Thư viện mã nguồn mở GSON 48](#_Toc349986786)

[2.8 Công nghệ thoại IP và Tổng đài mã nguồn mở Trixbox 50](#_Toc349986787)

[2.8.1 Công nghệ thoại IP 50](#_Toc349986788)

[2.8.2 Tổng đài mã nguồn mở Trixbox 50](#_Toc349986789)

[2.9 Tổng quan về SIP 51](#_Toc349986790)

[2.9.1 Lịch sử của SIP 52](#_Toc349986791)

[2.9.2 Các thành phần của giao thức SIP 53](#_Toc349986792)

[2.9.3 Thiết lập cuộc gọi qua SIP 53](#_Toc349986793)

[2.9.4 Thông điệp của SIP 55](#_Toc349986794)

[2.10 Kết luận 57](#_Toc349986795)

[CHƯƠNG 3. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG MGTS-TAXI 58](#_Toc349986796)

[3.1 Khảo sát 58](#_Toc349986797)

[3.1.1 GPS Vehicle Tracking Systems - The Shadow Tracker® Line 58](#_Toc349986798)

[3.1.2 *OpenGTS*™ - Open GPS Tracking System 59](#_Toc349986799)

[3.1.3 Hệ thống điều phối taxi VIETMAP 60](#_Toc349986800)

[3.1.4 Chương trình TAXI-Gis 60](#_Toc349986801)

[3.1.5 Ưu điểm của đề tài so với những hệ thống đã có 61](#_Toc349986802)

[3.2 Xác định yêu cầu và các đối tượng trong hệ thống 61](#_Toc349986803)

[3.2.1 Yêu cầu đối với hệ thống 62](#_Toc349986804)

[3.2.2 Các đối tượng trong hệ thống 62](#_Toc349986805)

[3.3 Thiết kế hệ thống 64](#_Toc349986806)

[3.4 Mô hình dữ liệu 66](#_Toc349986807)

[3.5 Thiết kế Web Service 67](#_Toc349986808)

[3.6 Mô hình xử lý của hệ thống 68](#_Toc349986809)

[3.7 Giải pháp hiển thị bản đồ 69](#_Toc349986810)

[3.8 Mô tả ứng dụng client trên Android 71](#_Toc349986811)

[3.8.1 Giao diện tương tác người dùng 71](#_Toc349986812)

[3.8.2 Mô hình xử lý của ứng dụng 72](#_Toc349986813)

[3.9 Giải pháp VOIP 74](#_Toc349986814)

[3.10 Tổng kết 75](#_Toc349986815)

[CHƯƠNG 4. HIỆN THỰC HỆ THỐNG MGTS-TAXI 76](#_Toc349986816)

[4.1 Xây dựng cơ sở dữ liệu 76](#_Toc349986817)

[4.1.1 Mô hình dữ liệu 76](#_Toc349986818)

[4.1.2 Chi tiết cơ sở dữ liệu 77](#_Toc349986819)

[4.2 Xây dựng Web Service 82](#_Toc349986820)

[4.2.1 Web Service hỗ trợ truy vấn dữ liệu 82](#_Toc349986821)

[4.2.2 Web Service lựa chọn xe phù hợp 83](#_Toc349986822)

[4.2.3 Web Service tương tác với Client 86](#_Toc349986823)

[4.3 Xây dựng ứng dụng web quản lý và điều hành taxi 90](#_Toc349986824)

[4.3.1 Tài nguyên hỗ trợ 90](#_Toc349986825)

[4.3.2 Cấu trúc ứng dụng 91](#_Toc349986826)

[4.3.3 Xây dựng các chức năng cho các đối tượng sử dụng hệ thống 92](#_Toc349986827)

[4.3.4 Xây dựng chức năng điều phối 92](#_Toc349986828)

[4.4 Xây dựng hệ thống bản đồ 95](#_Toc349986829)

[4.4.1 Phương pháp thực hiện 95](#_Toc349986830)

[4.4.2 Triển khai các bản đồ trong hệ thống 96](#_Toc349986831)

[4.5 Xây dựng ứng dụng Client 102](#_Toc349986832)

[4.5.1 Các xử lý về thông tin GPS 102](#_Toc349986833)

[4.5.2 Các xử lý và triển khai Google Direction API 103](#_Toc349986834)

[4.5.3 Các xử lý cơ bản trên Google Map API v2 104](#_Toc349986835)

[4.5.4 Truy vấn dữ liệu từ Web Service 110](#_Toc349986836)

[4.5.5 Các bước để thiết lập một phiên truyền thông bằng giao thức SIP 110](#_Toc349986837)

[4.5.6 Giao diện chương trình 114](#_Toc349986838)

[4.6 Triển khai tổng đài Trixbox 117](#_Toc349986839)

[4.6.1 Triển khai tổng đài Tribox 117](#_Toc349986840)

[4.6.2 Xây dựng thành phần quản lý phòng hội thoại 119](#_Toc349986841)

[4.7 Tổng kết 120](#_Toc349986842)

[CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 121](#_Toc349986843)

[5.1 Tổng kết đề tài 121](#_Toc349986844)

[5.2 Những khó khăn và hạn chế 122](#_Toc349986845)

[5.3 Hướng phát triển đề tài 122](#_Toc349986846)

[DANH MỤC TÀI LIỆU THAM KHẢO 123](#_Toc349986847)

**DANH MỤC HÌNH ẢNH**

[Hình 2‑1: Thống kê tỉ lệ sử dụng các Map API 5](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986848)

[Hình 2‑2: Thống kê thị phần các hệ điều hành di động 10](#_Toc349986849)

[Hình 2‑3: Các tầng trong một hệ thống Android 14](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986850)

[Hình 2‑4: Linux Kernel 2.6 15](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986851)

[Hình 2‑5: Android Framework 15](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986852)

[Hình 2‑6: Application Framework 16](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986853)

[Hình 2‑7: Application 16](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986854)

[Hình 2‑8: GPS 32](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986855)

[Hình 2‑9: 3 phân đoạn chính của GPS 33](#_Toc349986856)

[Hình 2‑10: Phân đoạn điều khiển của GPS 35](#_Toc349986857)

[Hình 2‑11: Qui trình thực thi một trang JSP 37](#_Toc349986858)

[Hình 2‑12: Servlet 38](#_Toc349986859)

[Hình 2‑13: Hoạt động của các servlet 39](#_Toc349986860)

[Hình 2‑14: HttpServlet 40](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986861)

[Hình 2‑15: Kiến trúc Tomcat 46](#_Toc349986862)

[Hình 2‑16: Biểu diễn json của một đối tượng 48](#_Toc349986863)

[Hình 2‑17: Biểu diễn json của một mảng 48](#_Toc349986864)

[Hình 2‑18: Sip cơ bản 54](#_Toc349986865)

[Hình 2‑19: Sip qua Registration 54](#_Toc349986866)

[Hình 2‑20: Sip qua Proxy server 55](#_Toc349986867)

[Hình 2‑21: Sip qua redirect serve 55](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986868)

[Hình 3‑1: Shadow Tracker® 58](#_Toc349986869)

[Hình 3‑2: GPS Vehicle Tracking Systems 59](#_Toc349986870)

[Hình 3‑3: Open GPS Tracking System 59](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986871)

[Hình 3‑4: Mô tả thiết kế hệ thống 66](#_Toc349986872)

[Hình 3‑5: Mô hình thực thể kết hợp 67](#_Toc349986873)

[Hình 3‑6: Mô hình xử lý của hệ thống 68](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986874)

[Hình 3‑7: Word Geodetic System 84 71](#_Toc349986875)

[Hình 3‑8: Giao diện tương tác người dùng 72](#_Toc349986876)

[Hình 3‑9: Mô hình giao tiếp giữa ứng dụng và Google Map Service. 73](#_Toc349986877)

[Hình 3‑10: Mô hình giao tiếp giữa ứng dụng và Taxi Manage Web Service 74](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986878)

[Hình 4‑1: Cơ sở dữ liệu 76](#_Toc349986879)

[Hình 4‑2: Web Service lựa chọn xe phù hợp 85](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986880)

[Hình 4‑3: Chức năng đăng nhập 86](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986881)

[Hình 4‑4: Chức năng đăng xuất 87](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986882)

[Hình 4‑5: Giao tiếp, cập nhật trạng thái và trao đổi dữ liệu với trung tâm điều khiển 89](#_Toc349986883)

[Hình 4‑6: Cách thức hoạt động của ứng dụng 91](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986884)

[Hình 4‑7: Ứng dụng web quản lý và điều hành taxi 92](#_Toc349986885)

[Hình 4‑8: Thành phần điều phối xe 93](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986886)

[Hình 4‑9: Hiển thị khách trên bản đồ 94](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986887)

[Hình 4‑10: Trạng thái khách hàng 94](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986888)

[Hình 4‑11: Danh sách khách hàng 95](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986889)

[Hình 4‑12: Mô hình xử lý hiển thị bản đồ 95](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986890)

[Hình 4‑13: Hiển thị vị trí xe 98](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986891)

[Hình 4‑14: Bản đồ trạng thái của hệ thống 99](#_Toc349986892)

[Hình 4‑15: Hiển thị thông tin một xe 99](#_Toc349986893)

[Hình 4‑16: Số lượng xe trong hệ thống 100](#_Toc349986894)

[Hình 4‑17: Hiển thị theo phân loại trạng thái xe 100](#_Toc349986895)

[Hình 4‑18: Giám sát hành trình của một xe 101](#_Toc349986896)

[Hình 4‑19: Lịch sử hoạt động của một xe 101](#_Toc349986897)

[Hình 4‑20: Vị trí trước 108](#_Toc349986898)

[Hình 4‑21: Vị trí sau 108](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986899)

[Hình 4‑22: Màn hình đăng nhập 114](#_Toc349986900)

[Hình 4‑23: Màn hình chính 115](#_Toc349986901)

[Hình 4‑24: Màn hình chỉ đường và tìm địa điểm 115](#_Toc349986902)

[Hình 4‑25: Màn hình nhận khách từ server 116](#_Toc349986903)

[Hình 4‑26: Màn hình khi có cuộc gọi tới và đi 116](#_Toc349986904)

[Hình 4‑27: Mô hình triển khai tổng đài trixbox 117](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986905)

[Hình 4‑28: Zoiper Wep 118](#_Toc349986906)

[Hình 4‑29: Quản lý phòng hội thoại 119](file:///C:\Users\Nguyen%20Trung\Desktop\Quan%20ly%20taxi_%203-2.docx#_Toc349986907)

[Hình 4‑30: Giao diện quản lý phòng hội thoại 120](#_Toc349986908)

**DANH MỤC BẢNG**

[Bảng 1‑1: Kế hoạch thực hiện đề tài 4](#_Toc349986932)

[Bảng 2‑1: Yêu cầu cấu hình tối thiểu cho Android 13](#_Toc349986933)

[Bảng 2‑2: Các Class trong GooleMap 26](#_Toc349986934)

[Bảng 2‑3: Các Class cơ bản trong SIP API 31](#_Toc349986935)

[Bảng 2‑4: Các phương thức của HTTP servlet tương ứng với các yêu cầu HTTP 40](#_Toc349986936)

[Bảng 2‑5: So sánh JSP và Servlet 42](#_Toc349986937)

[Bảng 2‑6: Một số định dạng được quy định sử dụng trong Web Service 44](#_Toc349986938)

[Bảng 2‑7: Các phiên bản Tomcat Server 45](#_Toc349986939)

[Bảng 4‑1: Quan hệ ADMIN 77](#_Toc349986940)

[Bảng 4‑2: Quan hệ BRANCH 77](#_Toc349986941)

[Bảng 4‑3: Quan hệ TEAM 77](#_Toc349986942)

[Bảng 4‑4: Quan hệ DRIVER 78](#_Toc349986943)

[Bảng 4‑5: Quan hệ DRIVERDETAIL 78](#_Toc349986944)

[Bảng 4‑6: Quan hệ OPERATOR 79](#_Toc349986945)

[Bảng 4‑7: Quan hệ TAXI 79](#_Toc349986946)

[Bảng 4‑8: Quan hệ CUSTOMER 80](#_Toc349986947)

[Bảng 4‑9: Quan hệ NOTIFICATION 81](#_Toc349986948)

[Bảng 4‑10: Quan hệ EXTENSION 81](#_Toc349986949)

[Bảng 4‑11: Quan hệ CONFERENCES 81](#_Toc349986950)

[Bảng 4‑12: Quan hệ CONFERENCEDETAIL 82](#_Toc349986951)

[Bảng 4‑13: Quan hệ UPDATETIME 82](#_Toc349986952)

[Bảng 4‑14: Web Service hỗ trợ truy vấn dữ liệu 83](#_Toc349986953)

[Bảng 4‑15: Trạng thái xe trong hệ thống 97](#_Toc349986954)

**DANH MỤC CÁC CHỮ VIẾT TẮT**

|  |  |
| --- | --- |
| 3G | Third-generation technology |
| 3GPP | 3rd Generation Partnership Project |
| ADT | Android Development Tools |
| AJAX | Asynchronous JavaScript and XML |
| API | Application Programming Interface |
| CE | Community Edition |
| CPA | Call Progress Analysis |
| CRUD | Create, Read, Update và Delete |
| CSDL | Cơ sở dữ liệu |
| CSS | Cascading Style Sheets |
| ERP | Enterprise resource planning |
| FOC | Full Operational Capability |
| GPL | General Public License |
| GPRS | General Packet Radio Service |
| GPS | Global Positioning System |
| GSM | Global System for Mobile Communications |
| GUI | Graphical user interface |
| HSDPA | High-Speed Downlink Packet Access |
| HTML | HyperText Markup Language |
| HTTP | Hypertext Transfer Protocol Secure |
| IDC | International Data Corporation |
| IMS | IP Multimedia Subsystem |
| IOC | initial operational capability |
| Json | JavaScript Object Notation |
| JSP | Java Server Page |
| MCS | master control station |
| MVC | Model–view–controller |
| OHA | Open Handset Alliance |
| PBX | Private Branch Exchange |
| PC | Personal Computer |
| RAS | Remote Access Services |
| REST | Representational State Transfer |
| RFC | Request for Comments |
| SCTP | Stream Control Transmission Protocol |
| SDP | Session Description Protocol |
| SIP | Session Initation Protocol |
| SMTP | Simple Mail Transfer Protocol |
| SOAP | Simple Object Access Protocol |
| TCP/IP | Transmission Control Protocol /Internet Protocol |
| UA | User Agents |
| UAC | User Agent Client |
| UAS | User Agent Server |
| UDP | User Datagram Protocol |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| VOIP | Voice over Internet Protocol |
| WAAS | Wide Area Augmentation System |
| WGS 84 | Geodetic System 84 |
| WSDL | Web Services Description Language |
| XHTML | Extensible HyperText Markup Language |
| XML | Extensible Markup Language |

**DANH MỤC CÁC THUẬT NGỮ ANH VIỆT**

|  |  |
| --- | --- |
| 3G | Là công nghệ truyền thông thế hệ thứ ba. |
| Activity | Cửa sổ chứa giao diện ứng dụng trong Android. |
| Block | Vệ tinh khối |
| Framework | Chương trình khung. |
| Hash | Chuỗi băm. |
| Hexa | Hệ thập lục phân. |
| Latitude | Vĩ độ. |
| Longitude | Kinh độ. |
| Mashup | Là thuật ngữ dùng để chỉ các ứng dụng và web sites có sự kết hợp dữ liệu hoặc chức năng từ hai hay nhiều nguồn khác nhau. |
| Package | Gói chương trình. |
| Smartphone | Điện thoại thông minh. |
| Web Application | Là một ứng dụng được người dùng truy cập thông qua mạng. |
| Web-based | Là những ứng dụng sử dụng trên nền tảng Web. |
| Widget | Ô điều khiển trên màn hình. |
| Window Phone | Là một hệ điều hành dành cho các thiết bị di động được phát triển bởi Microsoft. |

# 

# GIỚI THIỆU

## Tên đề tài

Xây dựng hệ thống đặt hàng qua điện thoại di động – “ORDER-SYSTEM”.

## Nội dung và giới hạn của đề tài

### Nội dung

Cơ sở hạ tầng và ở Việt Nam ngày càng phát triển. Trong đó hệ thống xe taxi phục vụ khách hàng đã trở nên phổ biến và tiện dụng. Việc xe taxi phát triển mạnh mẽ giúp phục vụ việc đi lại của người dùng trở nên dễ dàng.Ta có thể thấy một vài đặc điểm trong việc điều phối taxi hiện nay như nhân viên điều phối liên lạc với các tài xế taxi thông qua hệ thống bộ đàm, các xe được trang bị thiết bị GPS chuyên dụng để định vị vị trí xe tuy nhiên việc trang bị thiết bị này tại Việt Nam hiện nay mất chi phí cao, triển khai phức tạp và chưa phổ biến. Chính điều này đã gây ra một số hạn chế cho nhân viên trong việc điều phối như việc thông báo, cung cấp thông tin của khách hàng cho taxi được thực hiện thông qua hệ thống bộ đàm đến tất cả các taxi trong hệ thống mà không thể lựa chọn được taxi hợp lý nhất để phục vụ khách hàng, hệ thống không trang bị thiết bị GPS chuyên dụng nhân viên không thể giám sát được trạng thái và vị trí của các taxi trong hệ thống… Để khắc phục những khó khăn này luận văn này đưa ra giải pháp xây dựng hệ thống quản lý taxi trong đó Smartphone nền tảng Android được sử dụng giúp cho tài xế có thể gửi thông tin trạng thái và vị trí về trung tâm điều khiển thay thế cho thiết bị GPS chuyên dụng trên xe ô tô, nhận thông báo đón khách từ trung tâm điều khiển, thực hiện việc liên lạc thông qua tổng đài VOIP của trung tâm trên nền mạng internet 3G thay thế cho hệ thống liên lạc bằng bộ đàm. Với sự phát triển mạnh mẽ của Smartphone, cơ sở hạ tầng mạng viễn thông hiện nay sẽ là nền tảng giúp cho hệ thống vận hành và đáp ứng tốt những yêu cầu của người sử dụng.

Đề tài này sẽ thực hiện những công việc sau:

Phía quản lí cửa hàng:

Cho phép khách hàng sữ dụng mobile application hoặc web đặt hàng các sản phẩm của mình:

* Sản phẩm và các thông số (như tên, giá cả, miêu tả) của các sản phẩm được bán.
* Cho phép khách hàng đặt hàng các sản phẩm.
* Quản lý, điều hành việc giao hàng dựa trên nền tảng Web.
* Hệ thống tổng đài Trixbox quản lý việc liên lạc nội bộ giữa tài xế và nhân viên điều phối.

Phía người giao hàng:

* Smartphone chạy hệ điều hành Android có cài đặt ứng dụng “Post Order” của hệ thống với các chức năng như sau:
* Giao tiếp với web server quản lý điều hành thông qua Web application.
* Cập nhật danh sách của các đơn hàng cần giao.
* Cập nhật bản đồ chứa vị trí của các đơn hàng.
* Nhận thông tin đường đi đế đơn hàng cần giao.
* Thực hiện xác nhận giao hàng.
* Thông báo có sự cố xãy ra với khách hàng trực tiếp thông qua tin nhắn

Phía người giao hàng:

* Đặt hàng thông qua web hoặc application đặt hàng trên smartphone chạy hệ điều hành android.
* Xác nhận các thông tin cần thiết để đặt hàng (tên, số điệ thoại địa chỉ giao hàng).

### Giới hạn

Lĩnh vực kinh doanh, đặc biệt là khinh doanh dịch vụ muốn vận hành được thì hệ thống cần phải có sự kết hợp chặt chẽ của nhiều thành phần khác nhau. Ở đây em chỉ tập trung vào việc xây dựng hệ thống đặt hàng qua điện thoại đối với của hàng thức ăn nhanh. Các vấn đề khác như quản lý hành chính, quản lý nhân sự, quản lý tài sản, quản lý kinh doanh… nằm ngoài tầm vóc của luận văn này.

## Đối tượng, phạm vi và phương pháp nghiên cứu của đề tài

### Đối tượng nghiên cứu

* Các yêu cầu kỹ thuật và chức năng của một hệ thống đặt hàng qua điện thoại.
* Tìm hiểu kỹ thuật lập trình trên thiết bị di động nền tảng Android.
* Kỹ thuật định vị người dùng.
* Tìm hiểu về kỹ thuật xây dựng Web Application với PHP, MYSQL và JAVASCRIPT bằng Yii Framework.
* Nghiên cứu và ứng dụng Google Maps API.

### Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài được tập trung vào những phần sau:

* Tập trung vào kỹ thuật phát triển ứng dụng trên Google Maps API trên nền tảng Android kết hợp với định vị ngừời dùng bằng hệ thống GPS.
* Phát triển Web Application với PHP, MYSQL và JAVASCRIPT bằng Yii Framework.
* Ứng dụng Google Map JavaScript API trong việc hiển thị dữ liệu đơn hàng dùng trong phân phối.

### Phương pháp nghiên cứu

Đề tài được thực hiện dựa trên phương pháp khảo sát, phân tích, ứng dụng các đối tượng nghiên cứu để xây dựng hệ thống.

## Thời gian nghiên cứu

Thời gian bắt đầu: 15/04/2012

Thời gian hoàn thành: 17/06/2015

Kế hoạch thực hiện đề tài:

|  |  |  |
| --- | --- | --- |
|  | Thời gian | Nội dung |
| Tuần 1 | 15/08-19/08 | * Đăng ký khóa luận tốt nghiệp |
| Tuần 2 | 20/08-26/08 | * Định hướng đề tài. * Mục tiêu, hướng nghiên cứu đề tài. * Khảo sát yêu cầu đối với hệ thống. |
| Tuần 3 | 27/08-02/09 |
| Tuần 4 | 03/09-09/09 | * Chuẩn bị cơ sở lý thuyết, nghiên cứu những công nghệ sử dụng trong đề tài. * Xác định yêu cầu đề tài, các chức năng của hệ thống. |
| Tuần 5 | 10/09-16/09 |
| Tuần 6 | 17/09-23/09 |
| Tuần 7 | 24/09-30/09 |
| Tuần 8 | 01/10-07/10 | * Thống nhất, Đăng ký đề tài với giảng viên hướng dẫn. |
| ­­Tuần 9 | 08/10-14/10 | Xây dựng đề cương cho khóa luận. |
| Tuần 10 | 15/10-21/10 | Báo cáo, hoàn chỉnh đề cương luận văn với giảng viên hướng dẫn. |
| Tuần 11 | 22/10-28/10 | * Xây dựng cơ sở dữ liệu, thành phần web-based quản lý cơ sở dữ liệu, các đối tượng trong hệ thống. * Xây dựng thành phần bản đồ hiển thị vị trí, trạng thái của xe taxi trên bản đổ google map theo thời gian thực. * Tiếp nhận, xử lý thông tin khách hàng, điều phối khách cho xe taxi cụ thể. * Web Service tiếp nhận, xử lý, phản hồi thông tin tiếp nhận từ client điện thoại. * Xây dựng các thành phần của Client cập nhật trạng thái xe, thông báo nhận khách hàng, hiển thị vị trí khách, hiện thị bản đồ, chỉ đường đến vị trí khách. |
| Tuần 12 | 29/10-04/11 |
| Tuần 13 | 05/11-11/11 |
| Tuần 14 | 12/11-18/11 |
| Tuần 15 | 19/11-25/11 |
| Tuần 16 | 26/11-02/12 | * Triển khai hệ thống, chuẩn bị demo, báo cáo tiến độ với giảng viên hướng dẫn. |
| Tuần 17 | 03/12-09/12 | * Triển khai tổng đài Trixbox. * Xây dựng thành phần quản lý phòng hội thoại cho các tài xế, và nhân viên điều phối. * Xây dựng chức năng gọi thoại cho client. |
| Tuần 18 | 10/12-16/12 |
| Tuần 19 | 17/12-23/12 |
| Tuần 20 | 24/12-30/12 |
| Tuần 21 | 31/12-06/01 |
| Tuần 22 | 07/01-13/01 | * Demo hệ thống |
| Tuần 23 | 14/01-20/01 | * Chuẩn bị hệ thống, demo thực tế cho giảng viên hướng dẫn. |
| Tuần 24 | 21/01-27/01 | * Hoàn chỉnh báo cáo. |

Bảng ‑: Kế hoạch thực hiện đề tài

## Cấu trúc luận văn

Luận văn được cấu trúc như sau.

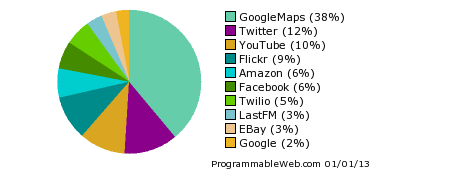
* Chương 1: Giới thiệu
* Chương 2: Công nghệ định vị, Hệ điều hành Android và những công nghệ sử dụng trong việc xây dựng hệ thống
* Chương 3: Phân tích và thiết kế hệ thống ORDER-SYSTEM
* Chương 4: Hiện thực hệ thống ORDER-SYSTEM
* Chương 5: Kết luận và hướng phát triển

# CÔNG NGHỆ GPS, HỆ ĐIỀU HÀNH ANDROID VÀ NHỮNG CÔNG NGHỆ SỬ DỤNG TRONG QUÁ TRÌNH XÂY DỰNG HỆ THỐNG

Chương này nhóm chúng em sẽ trình bày những nền tảng lý thuyết chính được sử dụng để làm cơ sở xây dựng hệ thống quản lý taxi bao gồm: Map API, Hệ điều hành Android, Công nghệ GPS, PHP, MySQL, JSON và thư viện mã nguồn mở GSON.

## Map API

Ngày nay trên các hệ thống websites các bản đồ là một thành phần xuất hiện khá phổ biến. Các bản đồ này được sử dụng để giúp hiển thị vị trí một địa điểm cụ thể lên bản đồ, tìm kiếm một địa chỉ, tìm hướng dẫn chỉ đường đến một địa chỉ và còn nhiều tiện ích khác nữa được các bản đồ hỗ trợ. Một sự vật bất kì chỉ cần nó có thông tin về vị trí thì đều có thể được hiển thị lên bản đồ.

Hiện nay có rất nhiều nhà cung cấp giải pháp về bản đồ như Yahoo! Maps, Bing Maps … nhưng được sử dụng phổ biến nhất vẫn là Google Maps. Theo thống kê ngày 1/1/2013 của website [www.programmableweb.com/apis](http://www.programmableweb.com/apis) Google Maps chiếm tới 38% các nội dung được sử dụng trong các Mashup[[3]](#footnote-3). Mashup là thuật ngữ dùng để chỉ các ứng dụng và web site có sự kết hợp dữ liệu hoặc chức năng từ hai hay nhiều nguồn khác nhau, cách thức này đang ngày càng trở nên phổ biến trong các ứng dụng web.

Hình ‑: Thống kê tỉ lệ sử dụng các Map API

### Google Map API

Google Map API tập hợp các API dành cho các nhà phát triển ứng dụng có thể tương tác, cũng như sử dụng các dịch vụ được Google và được cung cấp một cách miễn phí và có phí. Nền tảng của các API này là bản đồ của Google ở những dạng khác nhau giúp dễ dàng tích hợp nó vào trong các ứng dụng khác nhau như Website, smartphone… Ngoài ra, với Google Map cũng như các dịch vụ khác, chúng ta còn có thể tạo ra một bản đồ chỉ hiển thị một khu vực được định sẵn của riêng mình.

#### Google Maps Javascript API

Google Maps Javascript API được cung cấp dành riêng cho việc sử dụng Google Map trên các website. API này giúp hiển thị cũng như tương tác với bản đồ, kèm theo đó là các dịch vụ như tìm kiếm, chỉ đường… được cung cấp một cách miễn phí. Kết hợp với tài khoản Google, chúng ta có thể đánh dấu các đia điểm của riêng mình kèm theo hình ảnh và nội dung. Website sử dụng API này do Google tạo ra để cung cấp cho người sử dụng là <http://maps.google.com>.

#### Google Maps API Flash

Có chức năng tương tự như Google Maps Javascript API, nhưng sử dụng Flash để hiển thị thay vì Javascript. Google Maps API Flash dùng để phát triển các ứng dụng sử dụng Flash.

#### Google Earth API

Tương tự như Google Map Javascript API nhưng cung cấp thêm tính năng hiển thị hình ảnh 3D kỹ thuật số toàn cầu. Sử dụng API này chúng ta có thể đánh dấu, vẽ đường đi, hiển thị hình ảnh trên bề mặt bản đồ. Ngoài ra với API này, chúng ta còn có thể xây dựng ứng dụng bản đồ hiển thị 3D phức tạp để thêm vào website. Để sử dụng đươc Google Earth API thì yêu cầu trình duyệt phải được cài đặt thêm một số plugin.

#### Google Maps Image API

Là API dùng để cung cấp hình ảnh về một địa điểm cụ thể cho người dùng. Tham số đưa vào là tọa độ của điểm cần hiển thị và kích thước của hình ảnh tạo ra. Google Map Image API gồm hai loại chính là StaticMap và StreetView:

* Static Map: cung cấp cho người dùng hình ảnh tĩnh của bản đồ để hiển thị vào trong website.
* StreetView: cung cấp cho người dùng hình ảnh đường phố tại một địa điểm bất kỳ. Nhưng dịch vụ này chỉ cung cấp trong một số nước nhất định.

Google Maps Image API được cung cấp hoàn toàn miễn phí và không giới hạn số lượt sử dụng.

#### Maps API Web Service

Map API Web Service là tập hợp các dịch vụ về bản đồ đươc Google cung cấp thông qua giao thức HTTP, người dùng sử dụng các định dạng URL cùng những tham số khác nhau để truy vấn thông tin từ Google. Kết quả trả về là thông tin được lưu theo định dạng Json hay XML. Maps API Web Service gồm các dịch vụ sau đây:

* Direction API: cho phép người dùng tìm kiếm đường đi ngắn nhất giữa hai điểm trên bản đồ tùy thuộc vào loại phương tiện di chuyển. Phương tiện di chuyển được cung cấp gồm có 3 loại là xe ôtô, xe buýt và đi bộ. Kết quả trả về là một tập tin Json lưu trữ thông tin hướng dẫn đường đi, khoảng cách, danh sách các điểm để có thể vẽ đường đi trên Google Map. Đây là một dịch vụ hoàn toàn miễn phí, nhưng ở một số nước trong đó có Việt Nam vẫn chưa cung cấp toàn bộ các dịch vụ này.

Định dạng truy vấn của Direction API là:

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

* Distance Matrix API: là một dịch vụ của google giúp tính toán khoảng cách và thời gian giữa hai hay nhiều điểm trên bản đồ khi di chuyển bằng các phương tiện khác nhau. Dịch vụ này chỉ cho phép tính toán tối đa 100 địa điểm trên một lần truy vấn, 100 địa điểm trong 10 giây, 2500 địa điểm trong một ngày. Đối với phiên bản thương mại thì được sử dụng: 625 địa điểm trong một yêu cầu, 1000 địa điểm trong 10 giây và 100000 địa điểm trong 1 ngày.

Định dạng truy vấn của Distance Matrix API là:

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

* Elevation API: là một dịch vụ của Google dùng để truy vấn thông tin về độ cao, độ sâu của bất kỳ điểm nào trên bản đồ. Giá trị này được tính toán thông qua độ cao trung bình của 4 điểm gần đó nhất. Cũng như Distance Matrix API, Elevation API cũng chỉ cho phép thực hiện tối đa 512 địa điểm trong một truy vấn, 2500 truy vấn trong một ngày. Với phiên bản thương mại có thể thực hiện tối ta 100.000 truy vấn trong một ngày và mỗi truy vấn chứa tối đa 1.000.000 địa điểm.

Định dạng truy vấn của Distance Matrix API là:

http://maps.googleapis.com/maps/api/elevation/outputFormat?parameters

* Geocoding API: là một dịch vụ của Google cho phép chuyển đổi qua lại các địa chỉ cụ thể thực tế thành tọa độ vật lý. Nhờ đó chúng ta có thể tìm kiếm chính xác vị trí trên bản đồ thông qua địa chỉ. Nó được cung cấp miễn phí nhưng cũng rất hạn chế trong việc truy xuất. Mỗi ngày đối với một người dùng chỉ được sử dụng 2500 truy vấn và 10000 truy vấn đối với phiên bản thương mại.

Định dạng truy vấn của Geocoding API theo dạng:

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

* Place API: là dịch vụ của Google cho phép truy vấn để tìm kiếm thông tin về địa điểm, thông tin được cung cấp bao gồm địa chỉ, phân loại, điểm đánh giá… Nhưng Place API chỉ cho phép người dùng thực hiện 1000 truy vấn trong một ngày.

Định dạng truy vấn của Geocoding API theo dạng:

https://maps.googleapis.com/maps/api/place/search/output?parameters

* Time Zone API: là dịch vụ của Goole cho phép truy vấn múi giờ cho một vị trí trên trái đất. Tham số đầu vào là vị trí cần truy vấn thời gian, API sẽ trả về tên và id của múi giờ và thời gian lệch so mới múi giờ gốc.

Định dạng truy vấn của Geocoding API theo dạng:

<https://maps.googleapis.com/maps/api/timezone/outputFormat?parameters>

### Bing Map API

Bing Maps là dịch vụ bản đồ được cung cấp bởi Microsoft. Ngày nay, Bing Map không chỉ là bản đồ dành cho website mà nó còn được sử dụng rộng rãi cho các thiết bị di động sử dụng hệ điều hành Window Phone[[4]](#footnote-4). Các tính năng cơ bản của BingMap thì tương tự như Google Map như ngoài ra nó còn hỗ trợ thêm tính năng nổi bật là bird’s eyes view.

BingMap cũng cung cấp nhiều API, hỗ trợ cho người lập trình có thể tạo ra những ứng dụng dựa trên BingMap API và các dịch vụ đi kèm. Các API của Bing Map bao gồm:

* Bing Maps for Windows Store Apps: dùng để hỗ trợ việc hiển thị bản đồ trong các ứng dụng trên nền tảng windows 8.
* Bing Maps AJAX Control Version 7: là phiên bản mới nhất của Bing Maps Java Script API. Kết hợp giữa AJAX Map Control với Bing Maps REST Services[[5]](#footnote-5) và Bing Spatial Data Services[[6]](#footnote-6) để tạo ra một dịch vụ bản đồ mạnh mẽ trên nền tảng website và thiết bị di động.
* Bing Maps IOS Control: Hỗ trợ việc sử dụng BingMap trên các thiết bị sử dụng nền tảng IOS.
* Bing Maps Silverlight Control: Hỗ trợ cho việc kết hợp giữa Silverlight và BingMap để tạo ra một ứng dụng bản đồ với giao diện đẹp cùng khả năng tương tác cao.
* Bing Maps SOAP services: API này được sử dụng cho các ứng dụng sử dụng dịch vụ SOAP, nó cho phép bạn kết hợp bản đồ với hình ảnh, tìm đường, tính toán khoảng cách… vào trong chương trình hay website của bạn.

### Yahoo API

Yahoo Map đươc quản lý bởi tập đoàn Yahoo nhưng do công ty Cartfact thiết kế và cung cấp dữ liệu và chính thức đi được công bố vào ngày 16/05/2007.

Yahoo Map API giúp chung ta dễ dàng trong việc tích hợp bản đồ vào trong website của chúng ta hay vào các ứng dụng trên PC sử dụng các API mà Yahoo cung cấp bao gồm: Flash, Ajax và Map Image APIs.

Yahoo còn cung cấp các API khác cho phép chúng ta thêm các nội dung vào các tọa độ địa lý trên bản đồ của chúng ta, bao gồm Yahoo Local listings, Traffic and Weather reports, Upcoming events và Flickr Photos. Sử dụng Yahoo Map API làm cho trang web hay ứng dụng của chúng ta trở nên sống động với bản đồ chi tiết, nội dung của các vị trí trên bản đồ và khả năng tương tác của người dùng với bản đồ.

Các API cơ bản của Yahoo Map API bao gồm:

* Ajax API: sử dụng DHTML và JavaScript để giao tiếp với bản đồ của chúng ta. Yahoo cung cấp cho chúng ta các hàm JavaScript để tạo ra bản đồ.
* ActionScript 3 Flash APIs: Tích hợp bản đồ vào trong trang web hay ứng dụng desktop bằng Adobe Flash.
* Rest API: sử dụng để tạo ra một hình ảnh tĩnh từ Yahoo Map và giúp chúng ta tạo ra một bản đồ của riêng mình.
* Geocoding API: dùng để chuyển đổi từ địa chỉ vật lý thành latitude-longitude trên bản đồ và ngược lại.

## Hệ điều hành Android

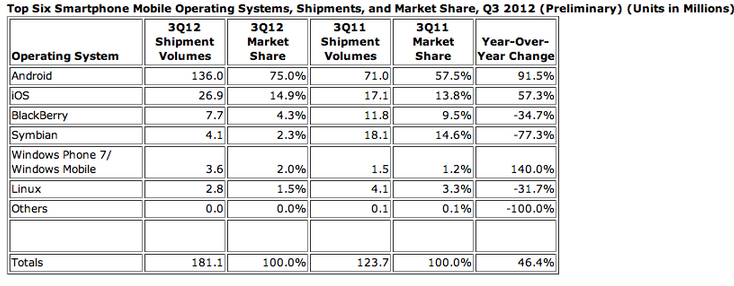
### Lịch sử phát triển Android

Vào khoảng những năm đầu của thế kỷ 21, Andy Rubin[[7]](#footnote-7) đã tạo ra Android như một hệ điều hành mở dành cho thiết bị di động. Năm 2005, Google đã mua sản phẩm này từ tay Andy Rubin và cử ông làm giám đốc phát triển nền tảng di động cho Google. Đây thực sự là một quyết định mang tính đột phá của Google, một mặt giúp Google khẳng định mạnh mẽ hơn vị trí của mình trong lĩnh vực mới được xem là Internet 2.0, nhưng mặt khác cũng tạo ra cuộc cạnh tranh khốc liệt giữa Google và đối thủ nặng ký Apple Iphone, đang là nhà cung cấp được ưu chuộng nhất thời gian này.

Tháng 11 năm 2007, Android chính thức ra mắt phiên bản 1.0 ra đời cùng với sự thành lập tổ chức OHA[[8]](#footnote-8) gồm nhiều công ty phần cứng, phần mềm. Mục tiêu chính của tổ chức này là cùng nhau nghiên cứu phát triển tiêu chuẩn mở cho các thiết bị di động, để giảm giá thành sản phẩm và nâng cao chất lượng dịch vụ. Android là hệ điều hành mở dành cho thiết bị di động đầu tiên, ra đời theo mục tiêu này. Nó dựa trên nền tảng mở Linux và ngôn ngữ lập trình Java của Oracle. Nền tảng Android phải hoạt động như một máy tính chạy trên chip, nghĩa là đủ nhỏ để vừa với thiết bị và đủ mạnh để chạy các ứng dụng. Ngoài ra, Android còn cung cấp các công cụ, tài liệu, thư viện hỗ trợ những nhà phát triển phần mềm và phần cứng phát triển ứng dụng cho thiết bị di động một cách dễ dàng. Trải qua nhiều năm, Android đã phát triển và nâng cấp lên các phiên bản khác nhau và phiên bản mới nhất hiện nay là 4.2. Thiết bị cầm tay cài đặt Android trên thị trường là thiết bị T-Mobile G1 do HTC sản xuất và bán ra thị trường Mỹ. Ngày nay, rất nhiều nhà sản xuất điện thoại di động như HTC, Samsung, LG Electronic, T-Mobile sử dụng Android làm hệ điều hành cho các dòng điện thoại mới của họ.

### Sự phát triển của Android

Có đến 136 triệu điện thoại Android được xuất xưởng trong quý 3 năm nay, tăng 91,5% so với cùng kỳ năm ngoái. Trong quý 3/2011, hệ điều hành Android có mặt trong 71 triệu điện thoại được phát hành và nắm trong tay khoảng 57,5% thị phần[[9]](#footnote-9).



Hình ‑: Thống kê thị phần các hệ điều hành di động

Bên cạnh sư phát triển chóng mặt của Android, có thêm hai hệ điều hành khác đang có đà tăng trưởng nhanh. Đầu tiên phải nhắc đến nền tảng iOS, một nền tảng từng thống trị thị trường smartphone. Theo IDC[[10]](#footnote-10), số lượng điện thoại iOS được xuất xưởng tăng tới 57,3%, trong quý 3 xuất xưởng trên 26.9 triệu chiếc iPhone chiếm 14,9% thị trường, tăng 17,1 triệu chiếc so với cùng kỳ năm ngoái. Tiếp tới là nền tảng Windows Phone cùa Microsoft. Hệ điều hành di động này dù ra đời sau nhưng cũng có mức tăng trưởng thật sự ấn tượng. Thống kê của IDC cho thấy đã có khoảng 3,6 triệu điện thoại Windows Phone được phát hành trong quý 3 nhưng con số này vẫn còn đứng sau 7,7 triệu điện thoại BlackBerry của RIM và 4,1 triệu điện thoại Nokia chạy trên nền tảng Symbian.

Sự phát triển của Android không chỉ thể hiện trên thị trường sản phẩm mà còn trên cả thị trường ứng dụng. Đến tháng 10 năm 2012, đã có khoảng 700.000 ứng dụng được viết cho Android, và số lượng tải về ứng dụng từ Google Play - cửa hàng ứng dụng chính của Android - được ước tính là 25 lượt[[11]](#footnote-11).

Trong lần đầu tiên ra mắt vào ngày 22 tháng 10 năm 2008, hệ điều hành Android bao gồm 12.000.000 dòng mã lệnh trong đó có 3.000.000 dòng XML, 2.800.000 dòng C, 2.100.000 dòng Java và 1.750.000 dòng C++.Từ lúc xuất hiện đến nay thì Android đã cho ra mặt những phiên bản[[12]](#footnote-12):

* Android 1.0 Donut (API level 1): Ngày 15 tháng 9 năm 2008, Android 1.6 SDK ra đời với tên gọi là dubbed Donut dựa trên nền tảng Linux 2.6.29 bao gồm rất nhiều những cập nhật mới. Thiết bị đầu tiên được sử dụng là HTC Dream.
* Android 1.1 (API level 2): Ngày 9 tháng 2 năm 2009, phiên bản cập nhật Android 1.1 ra đời. Thiết bị được sử dụng đầu tiên cũng là HTC Dream. Phiên bản này không có đột phá về mặt ý tưởng, mà chỉ là phiên bản sửa lỗi của phiên bản 1.0.
* Android 1.5 Cupcake (API level 3): Ra đời ngày 30 tháng 4 năm 2009, Android 1.5 được chính thức ra đời dựa trên nền tảng linx 2.6.27. Đây là phiên bản đầu tiên được đặt tên dựa theo những món tráng miệng. Phiên bản này cung cấp những cải tiến mới như Android 1.5 hỗ trợ hiễn thị bàn phím trên màn hình, Widget, ghi âm và hình ảnh.
* Android 1.6 Donut (API level 4): Ra đời ngày 15 tháng 9 năm 2009, Android 1.6 SDK dubbed Donut được ra đời dựa trên nền tảng Linux 2.6.29 bao gồm những cập nhật mới như: chức năng tìm kiếm nhanh, đọc văn bản, xóa nhiều tập tin hình ảnh củng lúc, hỗ trợ màn hình có độ phân giải WVGA…
* Android 2.0 Éclair (API level 5): được chính thức ra đời vào ngày 26 tháng 10 năm 2009, Android 2.0 với tên gọi là Eclair chạy trên nền tảng Linux 2.6.29, với những tính năng ngày càng hoàn thiện và tối ưu hơn chẳng hạn như: cho phép quản lý nhiều tài khoảng Email, tìm kiếm tin nhắn, Google Map 3.1.2, hỗ trợ nhiều kích thước và độ phân giải của màn hình…
* Android 2.01 Éclair (API level 6): Ra đời ngày 3 tháng 12 năm 2009, đây chỉ là phiên bản sửa lỗi cho phiên bản trước đó là Android 2.0
* Android 2.1 Eclair (API level 7): Phiên bản ra đời ngày 12 tháng 1 năm 2010, không có nâng cấp đáng kể trong phiên bản này.
* Android 2.2 Proyo (API level 8): Vào ngày 20 tháng 5 năm 2010, Android 2.2 SDK được ra đời dựa trên nền tảng Linux 2.6.32, với những tính năng bổ sung như: tối ưu hóa tốc độ và sử dụng bộ nhớ, tích hợp Javascript version 8 của Chrome vào trong trình duyệt web, hỗ trợ dịch vụ C2DM, hỗ trợ Adobe Flash…
* Android 2.3 Gingerbread (API level 9): Ra đời vào ngày 6 tháng 12 năm 2010 dựa trên nền tảng Linux 2.6.35 với những thay đổi như: hỗ trợ thiết kế giao diện đơn giản và hiệu quả, hỗ trợ hiển thị trên màn hình có độ phân giải lớn, nhập văn bản thông minh và nhanh hơn, hỗ trợ chức năng sao chép và dán, hỗ trợ tính năng dọn rác để tăng khả năng xử lý cho hệ điều hành, …..
* Android 2.3.3 Gingerbread(API level 10): Ra đời vào ngày 9 tháng 2 năm 2011 được sử dụng để sửa lỗi cho phiên bản trước.
* Android 3.0 Honeycomb (API level 11): Ngày 22 tháng 2 năm 2011 phiên bản Android 3.0 SDK lần đầu tiên xuất hiện trên máy tính bảng chạy hệ điều hành Android dựa trên nền tảng Linux 2.6.36 với những thay đổi lớn như: thêm System Bar và Action Bar, hỗ trơ đa tác vụ, bàn phím ảo được thiết kế để nhập văn bản nhanh hơn, trình duyệt với nhiều tab, hỗ trợ những bộ vi xử lý đa lõi, có khả năng mã hóa tất cả thông tin người dùng, Google Talk hỗ trợ chức năng Video Chat.
* Android 3.1 Honeycomb (API level 12): Ra đời ngày 10 tháng năm 2011 nhằm mục đích cung cấp thêm một số tính năng so với phiên bản trước như: hỗ trợ kết nối USB, hỗ trợ kết nối tay cầm chơi game, hỗ trợ HTTP proxy[[13]](#footnote-13) cho mỗi kết nối Wifi…
* Android 3.2 Honeycomb (API level 13): Ra đời ngày 15 tháng 7 năm 2011 nhưng không có sự thay đổi gì nổi bật ngoại trừ một số cải tiến như: khả năng truy cập vào các tập tin nằm trong thẻ nhớ.
* Android 4.0 Ice Cream Sandwich (API level 14): Được xây dựng trên nền tảng Linux 3.0.1 chính thức công bố vào ngày 19 tháng 10 năm 2011. Phiên bản này kết hợp thiết kế của cả 2 phiên bản là 3.x và 2.x vì vậy nó có thể chạy trên bất cứ thiết bị di động nào dù là điện thoại di động hay máy tính bảng, cùng với đó là những tính năng ưu việc hơn như: giao diện đẹp, Widget có thể thay đổi kích thước, cho phép khóa màn hình …
* Andoid 4.1 Jelly Bean (API level 16): Google công bố Android 4.1 ở hội nghị Google I/O vào ngày 27 tháng 6 năm 2012. Dựa trên nền tảng linux 3.0.31, JellyBean được xem như là một bản cập nhật với mục đích chính là cải thiện chức năng và hiệu xuất của giao diện, tăng khả năng tương tác với người dùng, máy tính bảng Nexus 7 là thiết bị đầu tiên được nâng cấp lên phiên bản này vào ngày 13 tháng 7 năm 2012.
* Android 4.2 Jelly Bean (API level 17): Google dự định công bố phiên bản Android 4.2 vào sự kiện ở thành phố New York nhưng bị hủy và dời vào ngày 13 tháng 12 năm 2012. LG Nexus 4 và SamSung’s Nexus 10 là hai sản phẩm đầu tiên được cập nhật phiên bản này. Các tính năng mới trong phiên bản này như: thêm vào tính năng “Photo Sphere”, bàn phím nhập hiệu quả hơn, màn hình khóa hỗ trợ thêm việc hiển thị Widget và chuyển trực tiếp sang camera, hỗ trợ đa người dùng trên máy tính bảng, hiển thị mạng wireless…

### Cấu trúc của Android

#### Phần cứng

Android là một nền tảng phần mềm được chạy trên các thiết bị di động vì vậy những yêu cầu về phần cứng của Android tương đối thấp, sau đây là những yêu cầu về cấu hình tối thiểu mà Android yêu cầu:

|  |  |
| --- | --- |
| Phần cứng | Những yêu cầu tối thiểu |
| Chip set | ARM-based |
| Ram | 128MB Ram. 256MB Flash External |
| Ổ cứng | Minu hay Micro SD |
| Màn hình chính | QVGA TFT LCD hoặc lớn hơn. |
| Navigation Key | 2MP CMOS |
| Bluetooth | 1.2 hoặc 2.0 |

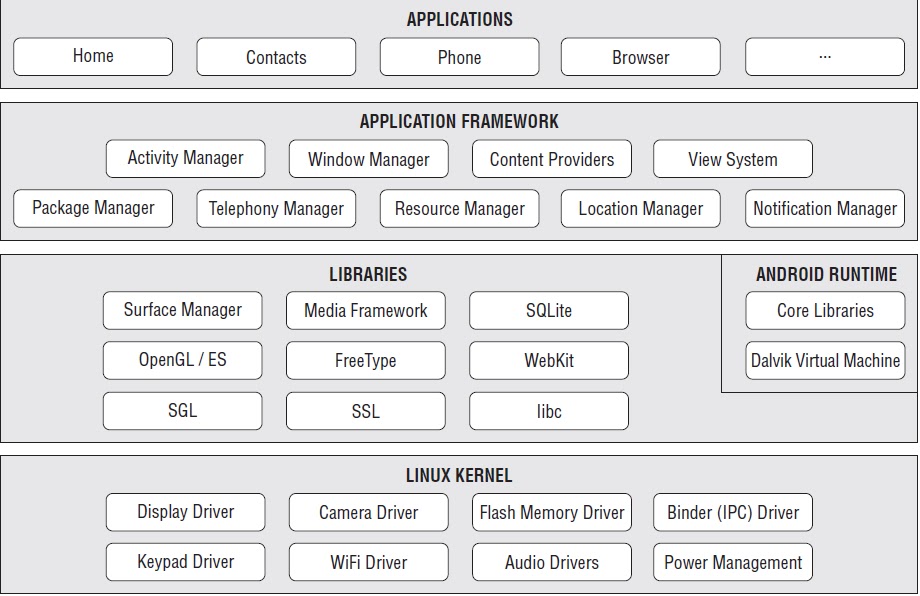
Bảng ‑: Yêu cầu cấu hình tối thiểu cho Android

#### Hệ điều hành Android

Có 5 tầng phân biệt trong một hệ thống Android:

* Tầng lõi ARM Linux: là tầng thấp nhất được xây dựng từ hai thành phần chính là Linux kernel và bộ vi xử lý ARM[[14]](#footnote-14). Bộ vi xử lý ARM hoạt động với hiệu suất cao nhưng tiêu thụ năng lượng rất ít. Còn Linux kernel phiên bản 2.6 được chứng minh là phiên bản có tình ổn định cao. Sự kết hợp giữa hai thành phần này, nhằm giải quyết phần nào giới hạn về nguồn năng lượng sử dụng và tối ưu việc sử dụng các tài nguyên của thiết bị.
* Tầng thư viện: chứa các mã nguồn cấp thấp được sử dụng cho các chức năng cơ bản như: mã hóa và giải mã âm thanh, hình ảnh kĩ thuật số, trình bày các giao diện đồ họa, bảo mật lưu lượng TCP/IP cũng như các thành phần cho trình duyệt Web, hỗ trợ xuất cơ sở dữ liệu (SQLite)…
* Tầng máy ảo Android: đây là một phần mềm dùng để chạy các ứng dụng trên thiết bị Android, bao gồm một tập các thư viện lõi cung cấp hầu hết các chức năng trong thư viện lõi của ngôn ngữ lập trình Java và máy ảo Dalvik[[15]](#footnote-15). Mỗi ứng dụng Android sẽ chạy trong một tiến trình riêng dành cho nó, với một máy ảo Dalvik riêng cũng được thiết lập dành riêng cho mỗi ứng dụng. Dalvik được viết để Android có thể chạy nhiều máy ảo cùng lúc nhằm tăng hiệu xuất.
* Tầng khung ứng dụng Android: tầng này có chức năng định nghĩa các Android API, cung cấp bộ công cụ ở mức cao để lập trình viên Android có thể dễ dàng xây dựng ứng dụng.
* Tầng ứng dụng lõi Android: bao gồm các ứng dụng cơ bản hỗ trợ thêm cho người lập trình như: Webkit brower, Google Calendar, Gmail, Maps.

Hình ‑: Các tầng trong một hệ thống Android

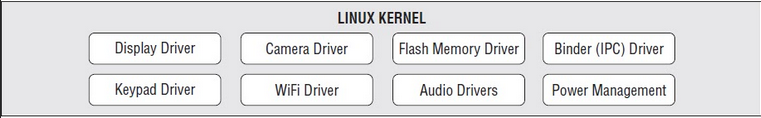
* Application…Chúng ta có thể viết thêm các ứng dụng cho tầng này.

**Linux Kernel 2.6**

Android được xây dựng trên nền tảng Linux kernel 2.6 mã nguồn mở, chứa những trình điều khiển dùng để điều khiển phần cứng như Keypad, Wifi, Camera, Audio, màn hình… được xem là trung tâm chỉ huy của nền tảng Android. Các giải thuật quản lý tài nguyên, chuyển đổi qua lại giữa các tác vụ qua lại, phân chia quyền, giải quyết tranh chấp được cài đặt trong linux kernel đảm nhận trách nhiệm tương tác với các thiết bị ngoại vi. Tuy nhiên người lập trình không thể lập trình trực tiếp trên tầng này.

Những tính năng nổi bật của Linux Kernel:

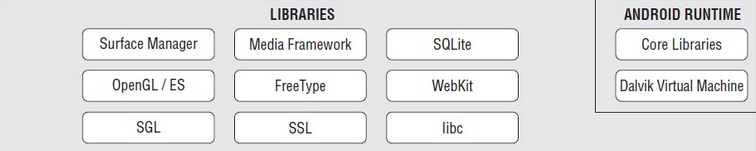
* Mô hình bảo mật tốt.
* Quản lý bộ nhớ tối ưu.
* Quản lý tiến trình hiệu quả.
* Quản lý các kênh kết nối mạng tốt.



Hình ‑: Linux Kernel 2.6

**Android Framework**

Được phát triển trên nền của Linux kernel, tầng này được cấu thành từ hai hay nhiều tầng con, cung cấp những tính năng cho Android:

* Android native libraries: đây là tập thư viện C/C++ và bạn sẽ gọi thư viện này thông qua giao diện Java. Phần này gồm thư viện:
  + Surface Manager dùng đễ tạo ra các cữa sổ giao diện.
  + OpenGL dùng để hỗ trợ xây dựng ứng dụng đồ họa 2D và 3D.
  + Media Framework dùng để xây dựng các ứng dụng âm thanh và hình ảnh.
  + SSL cung cấp giao thức truyền bảo mật.
  + SQLiter cung cấp khả năng quản lý cơ sở dữ liệu.
* Android Runtime: gồm các thư viện lõi của Java và máy ảo Dalvik. Các ứng dụng Android đươc viết bằng ngôn ngữ Java, nên khi biên dịch sẽ thành các tập tin \*.class. Máy ảo Dalvik sẽ làm chức năng đóng gói các tập tin này thành các tập tin nhỏ gọn \*.dex có thể dùng để chạy trên thiết bị, giúp tối ưu bộ nhớ và nguồn năng lượng giới hạn.

**Application Framework**

Hình ‑: Android Framework

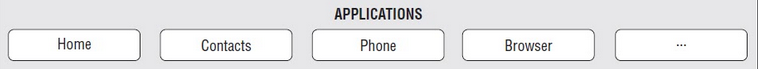
Tầng này chứa các thư viện Java hỗ trợ người dùng giao tiếp với tầng Android framework. Một phần của tầng này là do Google cung cấp sẵn cho chúng ta, một phần là do chúng ta tạo ra. Trong tầng này thành phần quan trọng nhất là Activity Manager dùng để quản lý chu trình sống của Activity.

* Activity Manager: quản lý chu trình sống của Activity trong ứng dụng Android.
* Telephony Manager: cung cấp thư viện để truy xuất đến các dịch vụ điện thoại cũng như thuê bao.
* View System: xử lý giao diện trong Android của bạn.
* Location Manager: cung cấp thư viện hỗ trợ xác định vị trí của chúng ta.



**Application**

Hình ‑: Application Framework

Đây là tầng cao nhất và ứng dụng của bạn sẽ nằm trong tầng này. Tuy nhiên, để tiện lợi cho việc truy cập một số thông tin ở các tầng dưới, Google đã viết một số thư viện tiện ích trong tầng này giúp chúng ta có thể truy xuất thông tin các tầng ở dưới linh hoạt và hiệu quả nhất.

Hình ‑: Application

### Các thành phần chính của Android

Những ứng dụng Android được viết bằng ngôn ngữ Java, sử dụng bộ công cụ Android SDK để biên dịch toàn bộ chương trình thành tập tin \*.apk để có thể cài đặt trên thiết bị chạy Android.

Mỗi ứng dụng sau khi được cài đặt trên thiết bị sẽ có những đặc điểm sau:

* Mỗi ứng dụng sẽ được Android gán cho một User Id phân biệt với nhau.
* Hệ thống sẽ thiết lập quyền hạn riêng cho mỗi ứng dụng trong việc truy nhập vào tập tin của hệ thống.
* Mỗi tiểu trình của ứng dụng sẽ chạy trên các máy ảo khác nhau vì vậy những ứng dụng này sẽ hoạt động độc lập với nhau.

Dù khác nhau ở cách thức hoạt động nhưng bất kỳ chương trình Android nào cũng bao gồm 3 thành phần chính là:

* + Application Components.
  + Tập tin Manifest.
  + Application Resources.

#### Application Components

Application Components đóng vai trò là thành phần giúp giao tiếp giữa ứng dụng của chúng ta và hệ thống. Có 4 loại Application Component chính như sau:

* Activities: Đại diện cho một cửa sổ chứa giao diện của chương trình giao tiếp với người dùng. Activities chứa giao diện và các hàm xử lý của chương trình. Mỗi Activity được tạo ra trong ứng dụng sẽ kế thừa một lớp Activity của Android Platform.Có 3 dạng Activity chính là:
  + Floating Activity: là loại Acitivity nổi, có kích thước không phủ lấp toàn bộ màn hình.
  + Activity Group: là một nhóm các Activity nhúng vào trong Activity khác.
  + Full Screen Activity: là dạng Activity phủ kín toàn bộ màn hình.
  + Một vòng đời của Activity gồm 3 trạng thái chính là:
  + Resumed: là trạng thái chạy của Activity, khi Activity chuyển sang trạng thái này, giao diện của nó sẽ được hiển thị trên màn hình.
  + Paused: là khi một Activity đang chạy mà có một Activity khác được khởi động thì trạng thái của Activity cũ sẽ chuyển sang dạng Paused. Khi ở trạng thái này thì Activity sẽ không còn được hiển thị trên màn hình.
  + Stopped: Khi ở trạng thái này thì Activity sẽ bị hủy nếu có một ứng dụng khác yêu cầu thêm bộ nhớ từ hệ thống.
  + Android cũng cấp cấp một số sự kiện để quản lý 3 trạng thái này của Activity:
  + onCreate(): Sự kiện này được gọi khi Activity lần đầu tiên được tạo. Trong sự kiện này chúng ta phải thực hiện một số công việc như: tạo giao diện cho Activity, tải dữ liệu vào danh sách… Sau sự kiện này thì sự kiện onStart() sẽ được gọi.
  + onStart(): Sự kiện này được gọi khi giao diện của ứng dụng được hiển thị lên màn hình. Lúc này tương tác của người dùng vẵn chưa được thiết lập.
  + onRestart(): Sự kiện này được gọi khi một Activity đang ở trạng thái Stopped nhưng muốn hiển thị và dành quyền điều khiển.
  + onResume(): Sự kiện này được gọi khi Activity bắt đầu cho phép tương tác với người dùng.
  + onPause(): Sự kiện này được gọi khi một Activity khác muốn dành quyền hiển thị và điều khiển của Activity hiện hành.
  + onStop(): Sự kiện này được gọi khi một Activity bị che khất hoàn toàn.
  + onDestroy(): Sự kiện này là sự kiện cuối cùng trước khi Acitivity bị hủy.
* Services: đây là Application Component chạy nền để thực hiện những tác vụ ẩn bên dưới mà không cần phải hiển thị hay chiếm quyền điều khiển. Một Services được tạo ra trong ứng dụng sẽ là một lớp con trong lớp Service trong Android Platform và chúng ta có thể dùng Activity để khởi động Service này.
* Content Providers: được sử dụng để quản lý một tập các dữ liệu chia sẻ ở những định dạng khác nhau như dưới dạng tập tin, cơ sở dữ liệu SQLite, trên Website hay bất cứ nơi nào mà chúng ta có thể truy xuất được. Thông qua Content Provider, những ứng dụng khác có thể truy xuất dữ liệu từ ứng dụng của chúng ta nếu được cho phép.
* Broadcast Receivers: là loại Application Component trong ứng dụng dùng để lắng nghe các thông điệp được gửi đi từ hệ thống. Bản thân những ứng dụng cũng có thể gửi đi những những Broadcast Receivers để những ứng dụng khác biết. Một đối tượng BroadcastReceivers được tạo ra từ lớp cha BroadcastReceiver trong Android Platform.

#### Tập tin Manifest

Được sử dụng để khai báo các Application Component, tập tin Manifest sẽ bao gồm những nội dung sau:

* Quyền hạn của ứng dụng:
* Khai báo các API được sử dụng trong chương trình.
* Khai báo những tính năng phần cứng mà ứng dụng có sử dụng.
* Các thư viện.
* Danh sách các Application Component.

#### Application Resources

Là những tài nguyên được sử dụng cho việc thiết kế giao diện cho chương trình bao gồm hình ảnh, âm thanh, giao diện… Tất cả những Application Resources này được lưu trữ trong thư mục res/. Mỗi thành phần trong Application Resources đều được hệ điều hành Android gắn cho một ID tương ứng và lưu trữ trong tập tin R.java trong thư muc gen/ của ứng dụng.

### Google Map API Android

Google Maps API là thư viện lập trình miễn phí được cung cấp cho người phát triển ứng dụng cùng với đó là các dịch vụ đi kèm như tìm kiếm địa điểm, tìm đường… Mặc dù miễn phí nhưng để có thể sử dụng các dịch vụ này chúng ta cần phải đăng ký và chứng thực với Google.

Trước khi làm việc với Google Map API, chúng ta cần tải API này và chắc chắn rằng mình đã có một Google Map Android API v2 key. Cả API và key đều được cung cấp miễn phí.

Để tạo một ứng dụng android mới sử dụng Google Map Android API v2 chúng ta phải thực hiện theo một số bước khác nhau. Nhiều bước chúng ta chỉ thực hiện một lần nhưng thông tin thu được sẽ hữu ích cho các ứng dụng tương lai sau này. Toàn bộ quá trình thêm một bản đồ đươc cung cấp bởi Google vào trong một ứng dụng Android được thực hiện như sau:

1. Tải và cấu hình Google Play service SDK[[16]](#footnote-16).Google Map Android API là một thành phần của SDK này.
2. Lấy được một API key: Để thực hiện được điều này chúng ta cần phải đăng ký một project trong Google APIs Console và nhận một chữ ký chứng thực cho chương trình của chúng ta:
3. Ghi rõ các cài đặt trong Application Manifest.
4. Thêm một Google Map tới một Android project mới hay đã tồn tại.
5. Xuất bản ứng dụng của chúng ta để có thể được sử dụng rộng rãi.

Google Map Android API v2 là một bộ phận của Google Play Service SDK, chúng ta có thể tải nó từ Android SDK manager. Để sử dụng google Maps android apiv2 trong chương trình, chúng ta phải cài đặt Google Play service SDK trước.

#### Google Map API Android v2 Key

Google maps android api v2 sử dụng một hệ thống quản lý Key mới. Những Key từ phiên bản Google Map Android v1 thông thường được biết với tên gọi là MapView sẽ không hoạt động trên phiên bản v2.

Để truy cập vào dịch vụ Google Maps với Maps API, chúng ta phải thêm một Maps API Key vào chương trình. Key này được cung cấp miễn phí cho tất cả mọi người. Chúng ta có thể dùng nó cho bất kỳ chương trình nào gọi tới Maps API và nó hỗ trợ không giới hạn số người dùng. Chúng ta thu được một Maps API Key từ Google APIs Console bằng việc cung cấp chữ ký chứng thực của chương trình chúng ta và tên của package. Một khi đã có được Key, chúng ta thêm nó vào ứng dụng của chúng ta bằng cách khai báo thêm một thành phần trong file AndroidManifest.xml của chương trình.

Để hiểu được quá trình đăng ký chương trình của chúng ta và thu được một Key yêu cầu một vài kiến thức về quá trình xuất bản một ứng dụng Android và những yêu cầu của nó. Tóm lại, tất cả các ứng dụng android phải đăng ký một chữ ký số mà bạn giữ Key riêng. Bởi vì chứng thực số chỉ có duy nhất, họ cung cấp một cách đơn giản để nhận biết chương trình của chúng ta. Nó rất hiệu quả trong việc tìm kiếm chương trình của chúng ta trong hệ thống cũng như Google Play Store, cũng như dò tìm những tài nguyên mà chương trình của chúng ta sử dụng như Google Maps servers.

Maps API Key được liên kết rõ ràng với cặp chứng thực/package, thay vì cho người sử dụng hay chương trình. Chúng ta chỉ cần duy nhất một Key cho mỗi chứng thực, bất kể bao nhiêu người dùng ứng dụng. Những ứng dụng mà sử dụng chung chứng thực có thể sử dụng chung API Key. Tuy nhiên, nên đăng ký cho mỗi chương trình một chứng thực khác nhau và nhận một khóa khác nhau cho mỗi chương trình.

Để nhận một Key cho một chương trình chúng ta cần thực hiện qua một số bước, những bước thực hiện được nêu ở đây và mô tả cụ thể trong chi tiết của mỗi phần:

1. Lấy lại thông tin về chứng thực của chương trình.
2. Đăng ký một project trong Google APIs Console[[17]](#footnote-17) và thêm vào Maps API như là một dịch vụ của project.
3. Một khi chúng ta có một project, chúng ta có thể yêu cầu một hay nhiều Key.
4. Cuối cùng, chúng ta có thể thêm một hay nhiều Key vào chương trình.

#### Hiển thị thông tin chứng thực

Maps API Key dựa trên một hình thức ngắn của chứng thực số, được biết đến với tên gọi SHA-1 fingerprint. Fingerprint là một chuỗi ký tự duy nhất được tạo ra từ thuật toán Hash[[18]](#footnote-18) của SHA-1[[19]](#footnote-19). Bởi vì fingerprint là một thuật toán đặc biệt nên Google Map sử dụng nó như một cách để chứng thực cho chương trình.

Để hiển thị SHA-1 fingerprint cho chứng thực của chúng ta thì trước tiên chúng ta phải đảm bảo chúng ta đã có chứng thực của mình. Chúng ta có hai cách để nhận chứng thực này:

* Debug certificate: bộ công cụ android SDK tạo ra chứng thực này một cách tự động khi chúng ta thực hiện debug chương trình từ dòng lệnh hay khi chúng ta build hay run chương trình từ eclipse mà không xuất bản chương trình. Chứng thực chỉ được dùng cho một chương trình mà chúng ta đang thử nghiệm. Chúng ta không thể xuất bản chương trình với debug certificate. Chúng ta có thể tạo ra một API Key từ chứng thực này nhưng Key này chỉ được sử dụng cho chương trình thử nghiệm.
* Release certificate: Bộ công cụ Android SDK tạo ra chứng thực này khi chúng ta sử dụng chức năng “release” của IDE. Chúng ta cũng có thể tạo ra chứng thực này thông qua chương trình keytool. Chứng thực này có thể được sử dụng cho một chương trình mà bạn muốn xuất bản. Một khi chúng ta có chứng thực

chính xác mà chúng ta cần, chúng ta có thể hiển thị SHA-1 fingerprint sử dụng chương trình keytool[[20]](#footnote-20).

Để có thể nhận được chính xác loại chứng thực cần thiết thì chúng ta phải thực hiện đẩy đủ một số bước sau đây:

*Đối với Debug certificate*

1. Xác định chính xác vị trí của tập tin “debug.keystore”, mặc định nó nằm chung thư mục với thư mục Android Virtual Device của bạn:
   * OS X and Linux: ~/.android/
   * Windows Vista và Windows 7: C:\Users\your\_username\.android\

Nếu chúng ta sử dụng eclipse với ADT và chúng ta không chắc tập tin này nằm ở đâu thì bạn có thể kiểm tra đường dẫn tại: Windows>Prefs>Android>Build.

1. Hiển thị thông tin SHA-1 fingerprint
   * Đối với Linux hay OS X, mở giao diện dòng lệnh và nhập vào dòng lệnh sau:

keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android

* + Đối với Vista hay Win 7, chạy dòng lệnh sau:

keytool -list -v -keystore "C:\Users\your\_user\_name\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android

1. Chúng ta sẽ nhìn thấy kết quả như bên dưới đây:

 Alias name: androiddebugkey  
 Creation date: Jan 01, 2013  
 Entry type: PrivateKeyEntry  
 Certificate chain length: 1  
 Certificate[1]:  
 Owner: CN=Android Debug, O=Android, C=US  
 Issuer: CN=Android Debug, O=Android, C=US  
 Serial number: 4aa9b300  
 Valid from: Mon Jan 01 08:04:04 UTC 2013 until: Mon Jan 01 18:04:04 PST 2033  
 Certificate fingerprints:  
      MD5:  AE:9F:95:D0:A6:86:89:BC:A8:70:BA:34:FF:6A:AC:F9  
      SHA1: BB:0D:AC:74:D3:21:E1:43:07:71:9B:62:90:AF:A1:66:6E:44:5D:75  
      Signature algorithm name: SHA1withRSA  
      Version: 3

Dòng bắt đầu bằng chuỗi SHA1 sẽ chứa mã chứng thực SHA-1 fingerprint. Chuỗi chứng thực fingerprint chứa 20 ký tự hexa[[21]](#footnote-21) và được ngăn cách với nhau bởi dấu hai chấm.

*Đối với release fingerprint*

1. Không có vị trí mặc định cho tập tin release keystore, nếu bạn không chỉ định nó khi xuất bản chương trình thì nó sẽ cho ra một tập tin apk không được đăng ký. Bạn phải đăng ký nó trước khi đưa ra cộng đồng sử dụng. Đối với release certificate, chúng ta cần biết về tên, mật khẩu, của keystore và chứng thực của nó. Có thể hiện thị tất cả các tên của khóa trong keystore[[22]](#footnote-22) bằng dòng lệnh:

keytool -list -keystore your\_keystore\_name

1. Thay thế “your\_keystore\_name” bằng đường dẫn và tên của keystore của chúng ta bao gồm cả đuôi .keystore, cần cung cấp mật khẩu của keystore để có thể hiển thị được những thông tin này.

Nhập vào dòng lệnh sau trong giao diện dòng lệnh:

keytool -list -v -keystore your\_keystore\_name -alias your\_alias\_name

1. Thay thế “your\_keystore\_name” bằng đường dẫn và tên của keystore của chúng ta bao gồm cả đuôi .keystore. Thay thế “your\_alias\_name”bằng tên mà chúng ta đã đăng ký khi khởi tạo.

Kết quả hiển thị ra sẽ tương tự như bên dưới:

Alias name: <alias\_name>  
 Creation date: Feb 02, 2013  
 Entry type: PrivateKeyEntry  
 Certificate chain length: 1  
 Certificate[1]:  
 Owner: CN=Android Debug, O=Android, C=US  
 Issuer: CN=Android Debug, O=Android, C=US  
 Serial number: 4cc9b300  
 Valid from: Mon Feb 02 08:01:04 UTC 2013 until: Mon Feb 02 18:05:04 PST 2033  
 Certificate fingerprints:  
      MD5:  AE:9F:95:D0:A6:86:89:BC:A8:70:BA:34:FF:6B:AC:F9  
      SHA1: BB:0D:AC:74:D3:21:E1:43:67:71:9B:62:90:AF:A1:66:6E:44:5D:75  
      Signature algorithm name: SHA1withRSA  
      Version: 3

Dòng bắt đầu bằng chuỗi SHA1 sẽ chứa mã chứng thực SHA-1 fingerprint. Chuỗi chứng thực fingerprint chứa 20 ký tự hexa và được ngăn cách với nhau bởi dấu hai chấm.

#### Tạo một API project

Một khi chúng ta đã đăng ký thành công chứng thực fingerprint, tạo và chỉnh sửa một project cho chương trình của chúng ta trên Google APIs Console và đăng ký Maps API.

Để nhận đươc một project và đăng ký với API chúng ta cần thực hiện một số bước như sau:

1. Trong trình duyệt, chúng ta chuyển hướng tới trang https://code.google.com/apis/console/

* Nếu chúng ta chưa từng sử dụng Google API Console trước đây thì cẩn phải tạo mới một project thông qua nhấp chuột vào nút “Create Project”. Console sẽ tạo cho chúng ta một project có tên là API Project. Ở trang tiếp theo, tên này sẽ được hiện ra phía bên trái, tại đây chúng ta sẽ có thể đổi tên cũng như quản lý Project.
* Nếu trước đây chúng ta đã sử dụng Google API Console thì sẽ thấy ngay danh sách cái project và các dịch vụ. Lúc này chúng ta có thể tạo một project mới hoặc đổi tên project cũ lại để sử dụng

1. Chúng ta sẽ thấy danh sách các dịch vụ nằm ở trung tâm của trang. Nếu vẫn chưa thấy, chọn “service” bên thanh định hướng phía bên trái.
2. Trong danh sách các dịnh vụ này, kéo xuống cho đến khi bạn thấy Google Maps Android API v2. Bật dịch vụ này.
3. Màn hình tiếp theo hiển thị các điều khoảng của dịch vụ. Nếu đồng ý, nhấp vào “Accept” để đồng ý. Sau đó nó sẽ trả lại màn hình chính.

Bây giờ chúng ta đã có thể nhận khóa Maps API.

#### Nhận một API Key

Nếu chương trình của chúng ta đã đăng ký sử dụng dịch vụ Google Android Map v2, thì chúng ta có thể yêu cầu nhận một API Key, có thể đăng ký nhiều khóa trên một project.

Để nhận được Key chúng ta phải thể thực hiện các bước sau đây:

1. Trên thanh định hướng phía bên trái, nhấp chuột vào API Access.
2. Trên trang kết quả, nhấp chuột vào Create New Android Key.
3. Sau đó nhập vào SHA-1 fingerprint và tên của Packet của bạn. ví dụ như:

BB:0D:AC:74:D3:21:E1:43:67:71:9B:62:91:AF:A1:66:6E:44:5D:75;com.example.android.mapexample

1. Google API console trả lại chuỗi gồm 40 ký tự đó là Key của chương trình Android.

AIzaSyBdVl-cTICSwYKrZ95SuvNw7dbMuDt1KG0

1. Lưu Key này để sử dụng cho các chương trình sử dụng dịch vụ Google map API v2.

#### Một số class cơ bản dùng để hiển thị và quản lý trên android

Theo [[1](#Placeholder1)] thì ta có thể tóm tắt các class trong GooleMap như sau:

|  |  |
| --- | --- |
| Tên Class | Chức năng |
| BitmapDescriptor | Dùng để tạo ra một bức hình, đối với marker có thể dùng để tạo ra marker icon, đối với Ground Overlay nó có thể thiết lập một hình ảnh tại một địa điểm trên bề mặt bản đồ. Để có được một Bitmapdescriptor ta sử dụng class BitmapDescriptorFactory. |
| BitmapDescriptorFactory | Dùng để tạo ra BitmapDescriptor |
| CameraPosition | Một class dùng để tập hợp tất cả các thuộc tính của việc hiển thị một điểm trên bản đồ. |
| CameraPosition.Builder | Dùng để xây dựng một camera position. |
| GroundOverlay | Dùng để hiện thị một hình ảnh trên bề mặt bản đồ tại một vị trí xác định. |
| GroundOverlayOptions | Dùng để tập hợp các thuộc tính của GroundOverlay với các thuộc tính cơ bản như:   * Anchor(float u, float v): Xác định điểm neo của hình. * Position(LatLng location, float width, float height): Chỉ ra vị trí, chiều dài và chiều rộng của khung hình. * Image(BitmapDescriptor image): thiết lập hình ảnh hiển thị trong GroundOverlay * Bearing(float bearing): thiết lập góc nhìn nghiêng với tham số là độ lớn của góc, mặc định bằng 0 |
| LatLng | Là class đai diện cho cặp tọa độ latitude/longitude dùng để xác định vị trí của một điểm bất kỳ trên bản đồ trước khi thể hiện nó. Tham số đầu vào là cặp giá trị kiểu double đại diện cho latitude/longitude. |
| LatLngBound | Là class đại diện cho những cặp tọa độ LatLng để tạo thành hình chữ nhật. |
| Marker | Là class dùng để đánh dấu một điểm trên bản đồ bằng một icon. Marker được vẽ theo định hướng màn hình của thiết bị hơn là so với bề mặt bản đồ. Một Marker gồm có bốn thuộc tính cơ bản là:   * Anchor: là điểm trên hình mà sẽ đặt vào vị trí trên bản đồ. Mặc định là 50% về phía bên trái của hình và bên cạnh trên của hình. |
|  | * Position: là giá trị LatLng cho vị trí của marker trên bản đồ. Chúng ta có thể thay đổi vị trí này bất kể khi nào chúng ta muốn di chuyển marker. * Title: Là chuỗi ký tự sẽ hiển thị trên cữa sổ khi chúng ta nhấp vào marker. * Snippet: là chuỗi ký tự hiển thị bên dưới Title. * Icon: là ảnh sẽ hiển thị đại diện cho marker. Nếu không được thiết lập thì marker sẽ sử dụng giá trị mặc định. Chúng ta không thể thay đổi thuộc tính này của marker một khi chúng ta đã tạo ra nó. * Drag status: Nếu chúng ta muốn kéo thả Marker thì thiết lập giá trị này là true.   Visibility: Mặc định giá trị này sẽ thiết lập là true, nếu muốn ẩn marker thì chuyển thành false. |
| MarkerOptions | Là class tập hợp cho các thuộc tính của Marker. |
| Polygon | Là class dùng để vẽ một hình nhiều cạnh trên bề mặt trái đất. Hình này có thể lồi hoặc lõm gồm các thuộc tính sau đây:   * Outline: là danh sách các đỉnh của hình sắp xếp theo chiều ngược của kim đồng hồ. * Holes: là vùng nằm bên trong Polygon không được lấp kín. Holes được xác đinh tương tự theo cách của Outline. Một Outline có thể chứa nhiều holes nhưng các holes này không được chồng lên nhau. * Stroke Width: độ dày của đường nối giữa các đỉnh của polygon. * Stroke color: màu của đường nối giữa các đỉnh của polygon. * Fill color: màu nền của polygon. * Visibility: dùng để thiết lập thuộc tính ẩn hiện cho polygon. * Geodesis status: xác định đường nối các đỉnh của polygon có vẽ theo địa hình hay không. |
| PolygonOptions | Là class tập hợp các thuộc tính của polygon |
| Polyline | Là class đại diện cho danh sách các điểm đươc nối với nhau bằng các đường thẳng. Các thuộc tính cơ bản của polyline bao gồm:   * Points: là danh sách các điểm được nối liên tiếp với nhau. Mặc định polyline không khép kín trừ khi điểm đầu trùng với điểm cuối. * Width: thể hiện chiều rộng của đường nối các điểm. * Color: thể hiện màu sắc của các đường nối các điểm. * Visibility: dùng để thiết lập thuộc tính ẩn hiện cho polyline. * Geodesis status: xác định đường nối các đỉnh của polyline có vẽ theo địa hình hay không. |
| PolylineOptions | Là class thể hiện các thuộc tính của polyline. |
| Tile | Là class thể hiện các thuộc tính của polyline. |
| TileOverlay | Là class dùng để hiện thị hình ảnh trên đỉnh của bản đồ. Những hình ảnh này có thể trong suốt nên cho phép chúng ta có thể thêm những tính năng vào trong bản đồ. |
| TileOverlayOptions | Là class tập hợp các thuộc tính của TileOverlay. |
| TileProvider | Là class dùng để dùng cho việc hiển thị hình ảnh trong TileOverlay. |
| UriTileProvider | Là một bộ phận của TileProvider nhưng nó chỉ yêu cầu một URL trỏ đến hình ảnh được cung cấp. |
| VisibleRegion | Là class dùng để vẽ một polyline bằng việc xác định bốn điểm. |
| CameraUpdate | Xác định sự di chuyển của camera giúp thay đổi góc nhìn của người dùng bằng cách sử dụng các phương thức như animateCamera(CameraUpdate), animateCamera(CameraUpdate,GoogleMap.CancelableCallback) hay moveCamera(CameraUpdate). |
| CameraUpdateFactory | Là class dùng để chứa các phương thức để khởi tạo một đối tượng CameraUpdate. Để thay đổi đối tượng góc nhìn của người sử dụng trong bản đồ chúng ta dùng các phương thức như: animateCamera(CameraUpdate),animateCamera(CameraUpdate, GoogleMap.CancelableCallback) hay moveCamera(CameraUpdate) |
| GoogleMap | Là một interface mà một đối tượng nào đó phải kế thừa khi muốn hiển thị và quản lý bản đồ. |
| GoogleMapOptions | Là class chứa các thiết lập để khởi tạo một đối tượng GoogleMap. |
| MapView | Dùng để hiển thị bản đồ. |
| Projection | Được sử dụng để chuyển đổi giữa vị trí trên màn hình thành tọa độ trên bản đồ. Class này chỉ gồm hai phương thức chính là:   * public LatLng fromScreenLocation (Point point): giúp chuyển một điểm trên màn hình về tọa độ vật lý trên bản đồ. * public Point toScreenLocation (LatLng location): giúp chuyển một tọa độ vật lý trên bản đồ thành tọa độ điểm ảnh trên màn hình. |
| UiSettings | Là class chứa các thiết lập về việc hiển thị các thành phần hỗ trợ trên bản đồ. |

Bảng ‑: Các Class trong GooleMap

### Google Place API Android

Google Places API cho phép chúng ta thực hiện các truy vấn thông tin về địa điểm bao gồm các địa điểm lân cận cũng như chính xác các địa điểm. Google Place API trả về danh sách các địa điểm cùng thông tin tóm gọn về mỗi địa điểm, còn thông tin chi tiết sẽ được cung cấp thông qua truy vấn Place Details. Theo [[2](#Placeholder2)] ta có thể tóm tắt các loại truy vần của Google Place API Android như sau:

#### Truy vấn thông tin gần chúng ta

Cung cấp cho chúng ta thông tin về địa điểm bên trong một khu vực được chỉ định sẵn. Chúng ta có thể chọn lọc kết quả thông qua việc cung cấp thêm các từ khóa hay loại của địa điểm mà chúng ta đang muốn tìm kiếm.

HTTP URL theo dạng:

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

Giá trị tham số output là định dạng văn bản của Google sẽ trả về khi chúng ta gửi yêu cầu lên server bao gồm:

* + Json: chỉ ra giá trị trả về là JavaScript Object Notation(Json)
  + Xml: chỉ ra giá trị trả về là XML.

Chúng ta cần chắc chắn cung cấp đầy đủ giá trị của các tham số quan trọng trong yêu cầu để có thể nhận được chính xác thông tin về địa điểm cần tìm, giống như URLs chuẩn thì các tham số sẽ được cách nhau bằng dấu “&”.

Những tham số yêu cầu cơ bản bao gồm:

* + Key: giá trị của tham số là API Key mà chúng ta được cung cấp thông qua việc đăng ký với google.
  + Language: giá trị của tham số là ngôn ngữ mà Google sẽ trả về.
  + Radius: giá trị của tham số là bán kính tính từ tâm của điểm mà chúng ta muốn truy vấn thông tin.
  + Sensor: giá trị của tham số này là true hay false dùng để xác định có hay không sử dụng những bộ cảm biến như GPS … để xác định vị trí.

Maps API sử dụng cho khách hàng là doanh nghiệp không bao gồm tham số chữ ký trong các truy vấn.

Ví dụ cần truy vấn các địa điểm về ẩm thực tại Sydney trong bán kính 500m kể từ vị trí hiện tại chúng ta sẽ sử dụng câu truy vấn sau:

https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=33.8670522,151.1957362&radius=500&types=food&name=harbour&sensor=false&key=*AddYourOwnKeyHere*

#### Text Search Request

Google Places API Text Search Service là một dịch vụ web dùng để trả lại thông tin về thông tin của một địa điểm cụ thể. Dịch vụ sẽ trả lại danh sách địa điểm phù hợp với truy vấn. Chúng ta có thể sử dụng Google Detail để có thêm được những thông tin cụ thể về các địa điểm được trả về.

* Truy vấn của Place Text Search là một HTTP URL theo dạng:

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

* Giá trị của tham số output xác định kiểu của kết quả trả về là Json hay xml.
* Các tham số yêu cầu cơ bản bao gồm:
  + Query: là một chuỗi truy vấn theo kiểu String dùng để tìm kiếm địa điểm.
  + Key và sensor: là những tham số tương tự với Nearby Search Request.
* Ví dụ chúng ta muốn tìm khách sạn tại Sydney thì chuỗi truy vấn sẽ như sau:

https://maps.googleapis.com/maps/api/place/textsearch/xml?query=restaurants+in+Sydney&sensor=true&key=*AddYourOwnKeyHere*

#### Radar Search Request

Google Places API Radar Search Service cho phép chúng ta tìm trên 200 địa điểm một lần, tuy nhiên thông tin về những địa điểm này sẽ ít hơn bình thường so với sử dụng Text Search hay Nearby Search. Với Radar Search giúp người sử dụng có thể xác định khu vực yêu thích của mình trên bản đồ. Để có thêm thông tin cụ thể về các địa điểm trong danh sách trả về chúng ta có thể sử dụng Google Detail.

* Truy vấn của Radar Search Request là một HTTP URL theo dạng:

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

* Giá trị của tham số output xác định kiểu của kết quả trả về là Json hay xml.
* Các tham số cơ bản bao của Radar Search tương tự như Nearby Search.
* Ví dụ chúng ta muốn tìm kiếm những quán cà phê có tên là vegetarian trong khu vực có bán kính 5000m thì chuỗi truy vấn sẽ như sau:

https://maps.googleapis.com/maps/api/place/radarsearch/json?location=48.859294,2.347589&radius=5000&types=food|cafe&sensor=false&keyword=vegetarian&key=*AddYourOwnKeyHere*

#### Kết quả truy vấn được trả về từ Google Search

* Định dạng của kết quả trả về được chỉ định thông qua tham số output trong truy vấn đến Google Search.
* Kết quả trả về của TextSearch tương tự như NearbySearch ngoại trừ thuộc tính formatted\_address thay thế bằng vicinity.

Một Json trả về thì gồm 4 thành phần chính là:

* “status” chứ metadata của yêu cầu.
* “results” chứa mảng cách địa điểm tra về với thông tin cụ thể của từng địa điểm.
* “http\_attributions” chứa các thiết lập về thẩm quyền của danh sách các địa điểm đang được hiển thị.
* “next\_page\_token” chứa token dùng để hiển thị cho 20 kết quả tiếp theo. Để hiển thị thêm các kết quả này thì chúng ta chỉ cần truyền giá trị này vào tham số pagetoke. Tối đa cho một kết quả trả về từ Goole là 60 địa điểm và đươc chia thành 3 trang khác nhau.

### Google Direction API Android

Google Directions API là một dịch vụ dùng để tính toán đường đi giữa hai điểm trên bản đồ sử dụng các truy vấn HTTP. Chúng ta có thể tìm kiếm đường đi theo các chế độ khác nhau như: xe buýt, đi bộ, xe ôtô. Đường đi được xác định thông qua tọa độ của điểm đầu và điểm cuối được nhập vào thông qua một chuỗi ký tự hay tọa độ latitude/longitude. Google Direction API có thể trả lại nhiều đường đi khác nhau dựa trên danh sách các điểm mốc. Dù là một dịch vụ miễn phí nhưng đối với một người dùng thông thường thì chỉ có thể sử dụng 2500 truy vấn trong một ngày, còn đối với người dùng trong kinh doanh thì có thể sử dụng 100000 truy vấn trong một ngày.

Cấu trúc truy vấn của Google Directions API theo dạng:

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

Giá trị của biến output là Json hay XML.

Các tham số yêu cầu bao gồm:

* Origin: Vị trí điểm đầu của đường đi. Giá trị tham số có thể là tọa độ latitude/longitude hoặc địa chỉ. Chúng ta cần chú ý rằng trong chuỗi giá trị của tham số này không được sử dụng khoảng trắng.
* Destination: Vị trí điểm cuối của đường đi. Giá trị tham số có thể là tọa độ latitude/longitude hoặc địa chỉ. Chúng ta cần chú ý rằng trong chuỗi giá trị của tham số này không được sử dụng khoảng trắng.
* Sensor: Xác định có hay không sử dụng bộ cảm biến cho việc tính toán đường đi.

Các tham số tùy chọn cơ bản bao gồm:

* Mode: dùng để xác định loại của tuyến đường cần tin. Google cung cấp 4 loại chính bao gồm:
  + Driving: loại dùng để tìm đường cho xe ô tô hay các phương tiện lớn.
  + Walking: loại dùng để tìm đường cho người đi bộ.
  + Bicycling: loại dùng để tìm đường cho xe đạp.
  + Transit: loại dùng để tìm đường cho các phương tiện công cộng.
* Waypoints: là danh sách các điểm cố định mà tuyến đường cần phải đi qua.
* Avoid: tham số đưa vào khi tìm đường sẽ tránh bao gồm 2 giá trị là:
  + Tolls: tránh những chỗ có thu phí.
  + Highways: tránh đường cao tốc.
* Alternatives: tham số này xác định có hay không việc hiển thị nhiều hơn một tuyến đường cho người dùng lựa chọn.
* Language: xác định ngôn ngữ của hướng dẫn chỉ đường được trả về. Mặc định sẽ là tiếng Anh.

### SIP Android

Android cung cấp một API dùng để hỗ trợ thực hiện truyền tải dữ liệu sử dụng giao thức SIP. API này bao gồm một hàng đợi SIP và những dịch vụ dùng để quản lý cuộc gọi để giúp chúng ta có thể dễ dàng thiết lập cuộc gọi ra ngoài và nhận cuộc gọi mà không quản lý phiên giao tiếp. Ngoài ra còn cung cấp các dịch vụ khác tương tự như với cuộc gọi thoại thông thường như ghi âm, gọi lại, đổ chuông….

Để sử dụng được giao thức SIP trên Android, chúng ta phải đáp ứng một số yêu cầu sau:

* Sip chỉ có thể hoạt động trên nền tảng Android 2.3 trở lên.
* Sip muốn hoạt động thì yêu cầu phải có kết nối mạng.
* Muốn tham gia truyền thông qua giao thức SIP thì cần phải cung cấp một tài khoản SIP từ nhà cung cấp. Ngoài thị trường có rất nhiều nhà cung cấp dịch vụ SIP miễn phí.

Các lớp cơ bản trong SIP API:

|  |  |
| --- | --- |
| Class | Mô tả |
| SipAudioCall | Điều khiển một cuộc gọi thoại thông qua giao thức SIP |
| SipAudioCall.Listener | Lắng nghe những sự kiện liên quan tới cuộc gọi SIP, như là khi nào nhận cuộc gọi, khi nào kết thúc cuộc gọi. |
| SipErrorCode | Xác định những lỗi trả về trong suốt quá trình hoạt động của giao thức SIP. |
| SipManager | Cung cấp API thực hiện những tác vụ trong giao thức SIP như là thiết lập kết nối, cung cấp phương thức truy nhập vào các dịch vụ của SIP |
| SipProfile | Xác định một hồ sơ cá nhân trong SIP bao gồm một tài khoảng SIP, thông tin về domain và server. |
| SipSession | Đại diện cho một phiên truyền thông SIP. |
| SipSession.Listener | Lắng nghe các sự kiện liên quan đến một phiên truyền thông SIP như khi bắt đầu đăng ký, hay đang thực hiện cuộc gọi ra ngoài. |
| SipSession.State | Xác định trạng thái của phiên truyền thông SIP, |
| SipRegistrationListener | Một interface dùng để lắng nghe các sự kiện liên quan đến đăng ký tài khoảng SIP với server. |

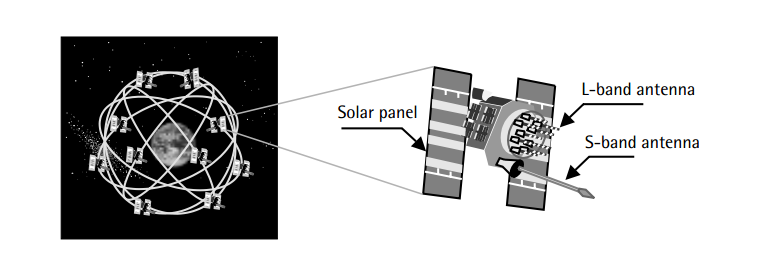
Bảng ‑: Các Class cơ bản trong SIP API

## Công nghệ GPS

Hệ thống định vị toàn cầu (GPS) là một hệ thống vệ tinh điều hướng được phát triển bởi bộ quốc phòng mỹ trong những năm đầu của thập niên 70. Ban đầu, GPS được phát triển như là một hệ thống được sử dụng trong mục đích quân sự. Tuy nhiên, thời gian sau nó được sử dụng thêm vào các mục đích dân sự.

GPS cung cấp liên tục thông tin vị trí và thời gian bất kỳ đâu trên thế giới trong bất kỳ điều kiện thời tiết nào. Nó phục vụ một không hạn chế người dùng cũng như được sử dụng cho các lý do an ninh. GPS là hệ thống một chiều thụ động có nghĩa là những người dùng chỉ có thể nhận tín hiệu từ vệ tinh mà không thể gửi trả. Để sử dụng GPS người dùng chỉ cần có thiết bị thu sóng và không phải trả bất kỳ chi phí nào khác. Sau khi tổng hợp từ [[3](#Placeholder3)] chúng ta có thể đưa ra một số khái niệm cơ bản vể GPS như dưới đâu.

### Tổng quan về GPS

GPS bao gồm một 24 vệ tinh được liên kết với nhau. Hệ thống này được hoàn thành vào tháng 7 năm 1993 và công bố chính thức vào ngày 8 tháng 12 năm 1993. Nó được xem là cơ sở quan trọng của hệ thống GPS với tên gọi “Năng lực hoạt động ban đầu” gọi tắt là IOC(initial operational capability). Để đảm bảo hệ thống vệ tinh này bao phủ toàn bộ trái đất một cách liên tục thì 24 vệ tinh được sắp xếp sao cho cứ mỗi 4 vệ tinh được đặt trong 6 mặt phẳng quỹ đạo (Hình 2-8). Với sự bố trí này sẽ có trên 4 vệ tinh luôn hiện hữu ở bất kỳ vị trí nào trên trái đất, nếu góc ngẩng là 10o. Để định vị tại một vị trí ta cần tối thiểu 4 vệ tinh.

Hình ‑: GPS

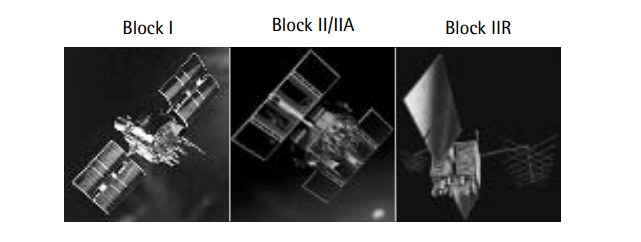
Quỹ đạo của các vệ tinh GPS gần như là hình tròn (Một hình elipse với tâm sai cực đại khoảng 0.01) với độ nghiêng khoảng 550 so với xích đạo. Nửa trục lớn của quỹ đạo GPS có độ dài khoảng 26560 km, tức là vệ tinh có độ cao khoảng 20200km bên trên bề mặt trái đất. Chu kỳ quỹ đạo vệ tinh tương ứng khoảng 12 giờ thiên văn.

Hệ thống GPS đạt được khả năng hoạt động đầy đủ gọi tắt là FOC(full operational capability) vào ngày 17 tháng 7 năm 1995 đảm bảo khả năng hoạt động thực tế tối thiểu 24 vệ tinh GPS. Thực tế thì khi GPS đạt FOC thì trên quỹ đạo hoạt động thường có nhiều hơn 24 vệ tinh.

### Những phân đoạn GPS

GPS bao gồm 3 phân đoạn chính là: phân đoạn không gian, phân đoạn điều khiển, phân đoạn người dùng.

* Phân đoạn không gian chính là nhóm gồm 24 vệ tinh, mỗi vệ tinh GPS phát đi tín hiệu bao gồm những thành phần sau: hai sóng sin là thành phần sóng mang, hai chuỗi dữ liệu số, và một thong điệp điều hướng, dữ liệu số và thông điệp điều hướng. Sóng mang và chuỗi dữ liệu số chủ yếu dùng để xác định khoảng cách từ máy thu của người sử dụng đến những vệ tinh GPS. Thông điệp điều hướng bao gồm tọa độ vệ tinh được biểu diễn dưới hàm biến đổi theo thời gian và các thông tin cần thiết khác. Những tín hiệu này được phát đi bởi các đồng hồ nguyên tử có độ chính xác cao được gắn trên những vệ tinh.
* Phân đoạn điều khiển hệ thống GPS bao gồm một mạng lưới rộng khắp những trạm theo dõi và một trạm điều khiển đặt tại bang Colorado của Mỹ. Nhiệm vụ của phân đoạn này là theo dõi, quản lý các vệ tinh GPS bao gồm vị trí của vệ tinh, tình trạng hệ thống, hoạt động của các đồng hồ nguyên tử.. và một số vấn đề khác. Những thông tin này được truyền tải trong thông tin về vị trí của vệ tinh và được chứa trong máy thu, khi vệ tinh di chuyển thì các thông tin này liên tục được cập nhật lại. Những thông tin này sau đó được đóng gói lại và gửi lên vệ tinh.
* Phân đoạn người dùng bao gồm tất cả người dân và quân đội. Với một bộ thu được kết nối với anten GPS, một người dùng có thể nhận tín hiệu GPS, tín hiệu này dùng để xác định vị trí mà không cần quan tâm vị trí của người đó đang ở đâu. GPS là một dịch vụ được cung cấp cho tất cả người sử dụng mà không phải trả thêm một chi phí nào.



Hình ‑: 3 phân đoạn chính của GPS

### Những thế hệ của vệ tinh GPS

Trong hệ thống ban đầu, nhóm vệ tinh GPS được xây dựng với chuỗi gồm 11 vệ tinh, gọi là những Block I. Vệ tinh đầu tiên trong chuỗi này bắt đầu hoạt động đầu tiên vào ngày 22 tháng 2 năm 1978, cái cuối cùng hoạt động vào ngày 9 tháng 10 năm 1985. Block I chủ yếu được sử dụng với mục đích chủ yếu là thí nghiệm. Góc nghiêng mặt phẳng quỹ đạo của vệ tinh này so với đường xích đạo là 630, sau đó được sửa đổi ở các thế hệ sau. Block I cuối cùng được rút ra vào ngày 18 tháng 11 năm 1995.

Thế hệ thứ 2 của các vệ tinh GPS được biết tới với tên gọi là Block II/IIA. Block IIA là phiên bản nâng cấp của khối II, với việc tăng khả năng lưu trữ dữ liệu thông điệp điều hướng từ 14 ngày đối với khối II đến 180 ngày đối với khối IIA, điều này giúp tang khả năng hoạt động độc lập của vệ tinh mà không cần sự hỗ trợ từ mặt đất. Có tổng cộng 28 Block II/IIA đã hoạt động trong khoảng thời gian từ tháng 2 năm 1989 đến tháng 11 năm 1997. Mặt phẳng nghiêng của Block II/IIA so với đường xích đạo là 550, thời gian hoạt động trong không trung là 7.5 năm nhưng thường chúng tồn tại trên không gian lâu hơn thế rất nhiều. Block II/IIA được tăng cường khả năng bảo mật hơn so với các phiên bản trước bằng việc thêm vào khả năng lựa chọn người dùng và chống lừa đảo trong cấu trúc tín hiệu truyền về máy thu.

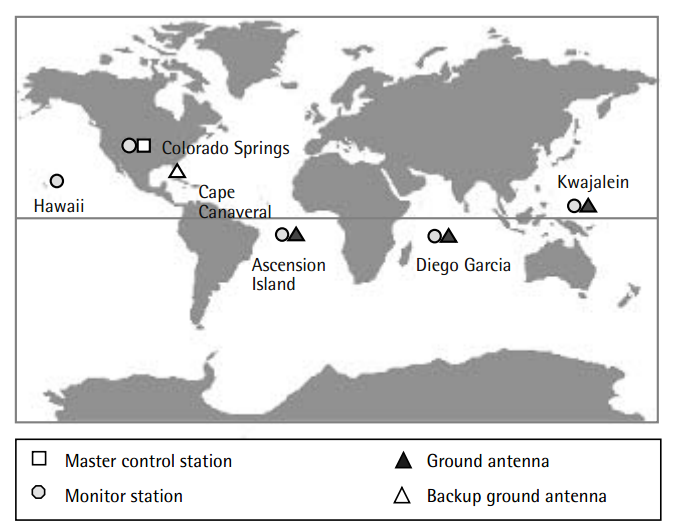
Bản nâng cấp tiếp theo của Block IIA là Block IIR, nó có khả năng tương thích với các Block thế hệ trước đó. Block IIR bao gồm 21 vệ tinh với thời gian sống được thiết kế là 10 năm. Thêm vào đó là độ chính xác cao hơn trước, những Block IIR có khả năng hoạt động độc lập cao nhờ giảm được phương sai của độ chính xác theo thời gian. Khả năng định vị của Block này được nâng cao nhờ chúng có khả năng tự sắp xếp lẫn nhau. Vào tháng 7 năm 2001, 6 vệ tinh thuộc Block IIR đã hoạt động thành công.

Theo sau Block IIR là Block IIF bao gồm 33 vệ tinh, khoảng thời gian sống của vệ tinh là 15 năm. Những vệ tinh thuộc khối này được tăng cường tính chính xác trong định vị.

Hệ thống Block III là hệ thống mới nhất được thiết kế và phát triển bởi tập đoàn Lockheed Martin[[23]](#footnote-23) theo đơn đặt hàng của bộ quốc phòng mỹ, vòng đời của vệ tinh thuộc khối này là 15 năm. Hiện tại đang có 4 vệ tinh đang được phát triển và đến năm 2014 sẽ nâng lên thành 35 vệ tinh. Các vệ tinh này được thiết kế với trọng lượng nhẹ, độ bền cao bằng vật liệu composite[[24]](#footnote-24). Sự thay đổi đầu tiên là trong việc truyền tín hiệu GPS dành cho dân sự sẽ không sử dụng đường truyền có tần số L1 mà thay vào đó bằng tần số L2 (1227,6 MHz), nó được sử dụng cho các Block IIR và các Block sau này. Tín hiệu truyền trên tần số L2 tăng thêm độ chính xác về vị trí cũng như tăng khả năng dẫn đường nhờ vào hệ thống tín hiệu dự phòng giảm khả năng bị nhiễu và mất sóng.

### Những vị trí điều khiển

Phân đoạn điều khiển của GPS bao gồm một trạm điều khiển chính và rộng khắp những trạm giám sát và những trạm điều khiển mặt đất.



Hình ‑: Phân đoạn điều khiển của GPS

Có 5 trạm giám sát và định vị tại Colorado Springs, Hawaii, Kwajalein, Diego Garcia, và đảo Ascension. Vị trí hoặc tọa độ của những trạm giám sát này được xác định chính xác. Những trạm này có mục đích theo dấu vết liên tục tất cả những vệ tinh GPS trong tầm nhìn, upload thông tin, điều chỉnh quản lý hoạt động của các vệ tinh.

### GPS-một vài khái niệm cơ bản

Mỗi vệ tinh GPS phát liên tục một tín hiệu vô tuyến tạo thành từ tổng thể 2 sóng mang[[25]](#footnote-25), 2 mã và một thông điệp điều hướng, bộ thu GPS phân tích thông tin này thu được bao gổm các khoảng cách tới những vệ tinh GPS và tọa độ của vệ tinh xác định qua thông điệp điều hướng.

Theo lý thuyết chỉ duy nhất cần 3 khoảng cách đến 3 vệ tinh mà được theo dấu vết một cách đồng thời. Trong trường hợp này, bộ thu sẽ được định vị tại chỗ giao nhau của ba hình cầu, mỗi hình cầu này có một bán kính tương ứng với khoảng cách vệ tinh – bộ thu và tâm vệ tinh. Tuy nhiên cần đến 4 vệ tinh để mô tả độ lệch giữa đồng hộ bộ thu và đồng hồ vệ tinh. Độ chính xác của phương pháp này là giới hạn trong 100m theo chiều ngang, 156m theo chiều dọc, và 340ns theo thành phần thời gian. Độ chính xác này tương đối thấp vì vậy phương pháp này không còn được sử dụng và được thay thế bằng phương pháp vi sai, trong đó sử dụng hai bộ thu theo dấu vết đồng thời cùng một vệ tinh. Trong trường hợp này, có thể đạt được độ chính xác từ vài centimet đền vài mét.

Lợi ích khác của GPS là khả năng mô tả vận tốc của người sử dụng thông qua một vài phương pháp. Phương pháp được sử dụng rộng rãi nhất là đánh giá tần số Doppler[[26]](#footnote-26) của tín hiệu GPS thu được. Độ dịch Doppler được xem như là chuyển động tương đối giữa bộ thu và vệ tinh.

### Nguyên lý hoạt động của GPS

#### Hoạt động cơ bản

Các vệ tinh GPS bay vòng quanh Trái Đất hai lần trong một ngày theo một quỹ đạo rất chính xác và phát tín hiệu có thông tin xuống Trái Đất. Các máy thu GPS nhận thông tin này và bằng phép tính lượng giác tính được chính xác vị trí của người dùng. Về bản chất máy thu GPS so sánh thời gian tín hiệu được phát đi từ vệ tinh với thời gian nhận được chúng. Sai lệch về thời gian cho biết máy thu GPS ở cách vệ tinh bao xa. Rồi với nhiều quãng cách đo được tới nhiều vệ tinh máy thu có thể tính được vị trí của người dùng và hiển thị lên bản đồ điện tử của máy. Máy thu phải nhận được tín hiệu của ít nhất ba vệ tinh để tính ra vị trí hai chiều (kinh độ và vĩ độ) và để theo dõi được chuyển động. Khi nhận được tín hiệu của ít nhất 4 vệ tinh thì máy thu có thể tính được vị trí ba chiều (kinh độ, vĩ độ và độ cao). Một khi vị trí người dùng đã tính được thì máy thu GPS có thể tính các thông tin khác, như tốc độ, hướng chuyển động, bám sát di chuyển, khoảng hành trình, quãng cách tới điểm đến, thời gian Mặt Trời mọc, lặn và nhiều thứ khác nữa.

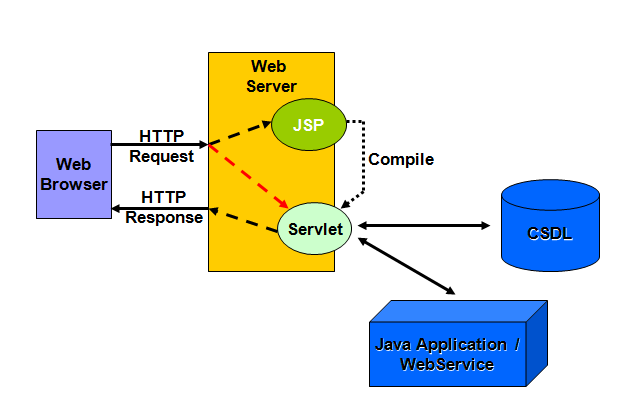
#### Độ chính xác của GPS

Độ chính xác của các thiết bị GPS ngày nay được tăng lên đáng kể, nhờ vào thiết kế hoạt động song song nhiều kênh của nó. Các máy thu 12 kênh song song nhanh chóng bám sát các quả vệ tinh khi vừa mới phát tín hiệu và chúng duy trì chắc chắn quan hệ này dù là trong ở rừng rậm rạp hay trong các tòa nhà cao tầng. Độ chính xác của các máy GPS trung bình khoảng 15 mét, các yếu tố ảnh hưởng đến độ chính xác này là trạng thái khí quyển và các yếu tố gây nhiễu trong không khí khác.Các mày thu phát sóng được trang bị khả năng WAAS (Wide Area Augmentation System) có thể tăng độ chính xác trung bình dưới 3m. WAAS là một dịch vụ sửa lỗi để tăng độ chính xác của các thiết bị GPS. Hệ thống bao gồm một mạng các đài thu tín hiệu GPS và phát tín hiệu đã sửa lỗi bằng các máy phát hiệu chỉnh. Để thu được tín hiệu đã sửa lỗi, người dùng phải có máy thu tín hiệu vi sai bao gồm cả ăn-ten để dùng cùng với máy thu GPS của họ.

## Java Server Page và Servlet

### Java Server Page[[4](#LeT11)]

JSP là một công nghệ hỗ trợ phát sinh nội dung động vào trong một tài liệu HTML bằng các câu lệnh Java được nhúng vào trong một tài liệu HTML bằng các thẻ đặc biệt. Các câu lệnh Java sẽ được thực thi ở phía server.



Hình ‑: Qui trình thực thi một trang JSP[4]

Hai quá trình quan trọng trong qui trình thực thi một trang JSP.

* Biên dịch: Trang JSP được biên dịch thành servlet ở lần đầu tiên phục vụ yêu cầu từ client.
* Thực thi: Servlet được thực thi ở server và kết quả thực thi được gửi về cho client (không phải nội dung của trang JSP).

Có hai loại dữ liệu trong một trang JSP

* Dữ liệu tĩnh: được chuyển trực tiếp vào response.
* Thẻ JSP: mã lệnh được biên dịch và thực thi ở server.

**Java Bean**

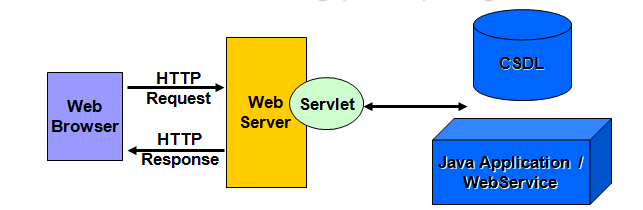
* Java Beans là các thành phần có thể sử dụng lại.
* Dùng để tách biệt business logic và presentation logic.
* Bean thường được tạo ra trong quá trình xử lý business để lưu dữ liệu và được chuyển cho lớp xử lý trình bày để hiển thị dữ liệu.
* Bean = instance của một class.

Java Bean là một class trong Java được thiết kế theo một số nguyên tắc:

* Mỗi thuộc tính trong Bean đều phải có phương thức set/get hoặc chỉ set hoặc chỉ get.
* Tên phương thức set/get trong Bean phải đặt theo mẫu:   
   set<Tên-thuộc-tính>(<parameters>)  
   get<Tên-thuộc-tính>() / is<Tên-thuộc-tính>()  
  Tên thuộc tính bắt buộc chỉ hoa viết ký tự đầu tiên.
* Phải cài đặt phương thức khởi tạo mặc định.

### Servlet[[4](#LeT11)]

Servlet là các chương trình chạy trên máy server (servlet container). Hoạt động như là lớp trung gian giữa máy client (với các yêu cầu HTTP) và CSDL hoặc các ứng dụng khác. Servlet có thể phát sinh các nội dung (HTML) động. Servlet không có GUI mà chỉ cài đặt những thao tác xử lý business logic.



Hình ‑: Servlet[4]

Việc sử dụng Servlet đem lại rất nhiều ưu điểm như:

* Hiệu quả hơn, mạnh mẽ hơn.
* Dễ sử dụng.
* Khả chuyển.
* An toàn hơn.

Ta sẽ tiến hành phân tích một ví dụ cơ bản

import java.io.\*;

import javax.servlet.\*;

public class SimpleGenericServlet extends GenericServlet {

public void service (ServletRequest request,

ServletResponse response)

throws ServletException, IOException {

response.setContentType("text/plain");

PrintWriter out = response.getWriter();

out.println("Hello World");

out.close();

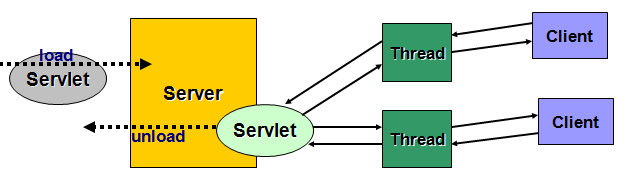
}

}

Ở đây ta thấy service(): là phương thức quan trọng nhất trong một servlet nó xác định nhiệm vụ của servlet Và được tự động gọi thực hiện khi có yêu cầu từ client. Vì vậy ta phải override phương thức này khi lập trình servlet. Service() nhận vào hai tham số là ServletRequest, ServletResponse:

* ServletRequest: Chứa tất cả những thông tin từ browser gửi lên server.
* ServletResponse: Được sử dụng để gửi thông tin về cho browser.

Hoạt động của các servlet được kiểm soát bởi server theo các bước sau:

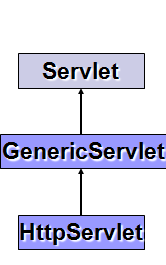


Hình ‑: Hoạt động của các servlet[[4](#LeT11)]

* Server nạp và khởi tạo một servlet. Tạo một thread để thực thi servlet
* Servlet có thể xử lý nhiều yêu cầu đồng thời.
* Server kết thúc hoạt động của một servlet.

Để thực hiện một Servlet ta có các phương thức thực thi sau:

* public void init(): được gọi thực thi một lần duy nhất khi khởi tạo servlet, dùng để khởi tạo trạng thái, mở file, thiết lập kết nối CSDL, …
* public void service(): được gọi thực thi để phục vụ các yêu cầu của client
* public void destroy(): được gọi thực thi một lần duy nhất để hủy một servlet, giải phóng tài nguyên.

Ta phân tích hình sau:

Hình ‑: HttpServlet[4]

Ta thấy Interface Servlet: khai báo các phương thức của một servlet (init(), service(), …) tiếp theo GenericServlet thi công interface Servlet.

Ở đây ta chú ý tới HttpServlet là lớp dẫn xuất từ lớp GenericServlet và bổ sung những tính năng đặc thù của HTTP ta sẽ sử dụng lớp này để thi công các Servlet trong hệ thống.

**HTTP Servlet**

Là lớp dẫn xuất của GenericServlet: chứa phương thức service() tuy nhiên phương thức service() đã được override để phân phối các yêu cầu đến các phương thức tương ứng doGet() và doPost() vì vậy không nên override phương thức service() khi dẫn xuất từ lớp HttpServlet mà chỉ cài đặt phương thức doGet() hoặc doPost() để xử lý yêu cầu từ client:

|  |  |
| --- | --- |
| **Phương thức** | **HTTP request** |
| **doGet** | GET, HEAD |
| **doPost** | POST |
| **...** | **...** |

Bảng ‑: Các phương thức của HTTP servlet tương ứng với các yêu cầu HTTP

Các phương thức trên nhận vào hai tham số là hai đối tượng HttpServletRequest và HttpServletResponse:

* HttpServletRequest: Dẫn xuất từ lớp ServletRequest cung cấp các phương thức truy cập dữ liệu header và các tham số được gửi theo yêu cầu từ client. Các phương thức được sử dụng để lấy dữ liệu: getParameterNames(), getParameter(String name), getParameterValues(String name)
* HttpServletResponse: Dẫn xuất từ lớp ServletRequest, cung cấp hai cơ chế để gửi dữ liệu về cho client getWriter() trả về một đối tượng PrintWriter dùng để gửi dữ liệu text; getOutputStream() trả về một đối tượng ServletOutputStream dùng để gửi dữ liệu nhị phân. Lưu ý: nên đóng PrintWriter và ServletOutputStream sau khi kết thúc quá trình gửi dữ liệu về client

Ví dụ về xử lý yêu cầu POST/GET:

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class SimpleHttpServlet extends HttpServlet {

public void doGet (HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException {

...

}

public void doPost (HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

}

### So sánh JSP và Servlet

Một số đặc điểm khác nhau giữa JSP và Servlet

|  |  |
| --- | --- |
| **JSP** | **Servlet** |
| Hướng tài liệu | Như một chương trình Java |
| Trình bày dữ liệu | Xử lý nghiệp vụ |
| View layer | Controller layer |

Bảng ‑: So sánh JSP và Servlet

## RESTful Web Service

### RESTful Web Service[[27]](#footnote-27)

Representational State Transfer (REST) hiện nay đã được sử dụng rộng rãi trên các hệ thống Web và được coi như giải pháp thay thế cho Web Service kiến trúc SOAP và Web Services Description Language (WSDL). Các nhà cung cấp dịch vụ Web 2.0 điển hình như Goolge, Yahoo, Facebook đã bỏ qua các giao thức dựa trên SOAP và WSDL mà tập trung vào việc sử dụng REST trong việc triển khai các dịch vụ của họ.

REST xuất phát từ luận án tiến sỹ của Roy Fielding được xuất bản năm 2000 "Architectural Styles and the Design of Network-based Software Architectures", là viết tắt của Representational State Transfer. REST không phải là một kiến trúc mà là một nhóm các nguyên lý kiến trúc, REST được sử dụng như một kiểu kiến trúc mềm khi chúng được áp dụng trong việc thiết kế các hệ thống. REST chú trọng đến tài nguyên hệ thống và việc định dạng và phương thức trao đổi dữ liệu thông qua giao thức HTTP đến một số lượng lớn các client mà không phải quan tâm đến vấn đề ngôn ngữ lập trình.

### Các đặc tính của hệ thống RESTful[[5](#Jos09)]

REST được sử dụng trong việc kết hợp tài nguyên mạng và các công nghệ trao đổi dữ liệu qua mạng để xây dựng các ứng dụng phân tán. Một hệ thống RESTful được xây dựng dựa trên những đặc tính sau đây:

* Phải là một hệ thống Client – Server
* Phi trạng thái (Stateless): không cần phải giám sát trạng thái các phiên làm việc của người dùng mà đáp ứng mỗi yêu cầu độc lập của người dùng. Yếu tố này làm cho REST trở nên đơn giản và đơn giản hóa thiết kế của Web Service
* Hỗ trợ bộ nhớ đệm (Cache): Trong cơ sở hạ tầng mạng, hỗ trợ bộ nhớ đệm ở nhiều mức độ khác nhau, phản hồi của server sẽ được client lưu trữ lại.
* Truy cập thống nhất (uniformly accessible): mỗi một tài nguyên trong hệ thống được client truy cập bằng một địa chỉ duy nhất (URI) thông qua các phương thức POST, GET, PUT, DELETE.
* Phải có cấu trúc phân lớp (layer), hỗ trợ khả năng mở rộng
* Code on Demand: các Client sẽ lấy được dữ liệu trong phản hồi khi quá trình xử lý của Server hoàn tất.

Do không phải là một kiến trúc mà chỉ là một nhóm các nguyên lý kiến trúc nên chúng ta có thể triển khai RETSful trên các kiến trúc mạng có sẵn. Ví dụ ta có thể sử dụng các tài nguyên Web có sẵn để triển khai RESTful. Điều này làm cho RESTful có khả năng mở rộng, duy trì và là một ứng dụng phân tán.

### Các thành phần của RESTful Web Service[[28]](#footnote-28)

#### Tài nguyên (Resources)

Có thể là bất kỳ thứ gì có địa chỉ cụ thể thông qua Web, có thể truy cập và trao đổi giữa Client và Server.

#### URI - Uniform Resource Identifier

Là một liên kết (hyperlink) đến một tài nguyên cụ thể, và đó là phương tiện duy nhất để thực hiện trao đổi dữ liệu giữa Client và Server.

#### Các phương thức HTTP sử dụng truy cập tài nguyên

Việc sử dụng các phương thức HTTP cụ thể để truy cập tài nguyên là một trong những đặc điểm quan trọng của RESTful Web Service. REST yêu cầu các lập trình viên sử dụng các phương thức HTTP một cách rõ ràng, phù hợp với định nghĩa của giao thức đó. Nguyên tắc cơ bản của REST là thiết lập một ánh xạ 1-1 giữa các thao tác create, read, update và delete (CRUD) với các phương thức HTTP như sau:

* Sử dụng phương thức POST khi tạo một tài nguyên trên server(POST/create).
* Sử dụng phương thức GET khi muốn lấy xuống một tài nguyên trên server (GET/read).
* Sử dụng phương thức PUT khi thay đổi trạng thái hay cập nhật một tài nguyên trên server (PUT/update).
* Sử dụng phương thức DELETE khi xóa tài nguyên trên server (DELETE/delete).

#### Định dạng dữ liệu trao đổi (XML, JSON, hoặc cả hai)

Xây dựng Web Service với các định dạng được chấp nhận trong phần HTTP Accept header để có thể thống nhất dữ liệu truyền nhận giữa Client và Server.Client có thể gửi các yêu cầu để nhận về tài nguyên với định dạng tương ứng từ Server. Một số định dạng được quy định sử dụng trong Web Service:

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

Bảng ‑: Một số định dạng được quy định sử dụng trong Web Service

Với việc thống nhất dịnh dạng dữ liệu theo các chuẩn cụ thể giúp cho RESTful Web Service có thể đáp ứng yêu cầu của những Client được viết bằng những ngôn ngữ khác nhau, chạy trên các nền tảng và thiết bị khác nhau.

## MySQL 5.5 và Apache Tomcat 7

### MySQL 5.5[[29]](#footnote-29)

MySQl là hệ quản trị cơ sở dữ liệu mã nguồn mở phổ biến nhất trên thế giới. Facebook, Twitter, Wikipedia, YouTube hiện đang sử dụng hệ quản trị cơ sở dữ liệu này. Hầu hết các cơ sở dữ liệu quan hệ một vài thập kỷ trở lại đây đều được thiết kế hướng đến việc xây dựng các ứng dụng ERP phức tạp thì MySQL được thiết kế và tối ưu hóa hỗ trợ cho việc xây dựng những ứng dụng Web. Với hiệu suất khả năng mở rộng, độ tin cậy cao cũng như dễ dàng trong sử dụng, tiết kiệm được chi phí mà MySQL đã trở thành hệ quản trị cơ sở dữ liệu mặc định với các ứng dụng Web-based đối với đa số các lập trình viên.

Một điểm mạnh nữa của MySQL là phần mềm mã nguồn mở miễn phí, rất dễ dàng triển khai trên các hệ điều hành khác nhau như Windows, Linux, Mac OS X, Unix …

Hiện tại MySQL 5.5 được triển khai với những cải thiện đáng kể như: tăng hiệu suất, khả năng mở rộng, tăng độ khả dụng với người dùng, kiến trúc đa xử lý về phần cứng, phần mềm … MySQL 5.5 là sự kết hợp giữa cơ sở dữ liệu MySQL và InnoDB storage engine cung cấp cho chúng ta một giải pháp quản lý dữ liệu mạnh mẽ với những ưu điểm sau:

* InnoDB storage engine là mặc định trong MySQL.
* Tăng hiệu suất và khả năng mở rộng trên Windows.
* Tương thích, tận dụng được tối đa hiệu suất với các hệ thống kiến trúc đa xử lý trên các nền tảng khác nhau.
* Tăng khả năng đáp ứng, hiệu quả, khả năng quả lý, dễ dàng hơn cho người sử dụng...

Với InnoDB mặc định trong MySQL 5.5, những bảng được tạo mới sẽ có Engine Storage mặc định là InnoDB giúp cho phép lập trình các ứng dụng Transaction mà không cần phải thay đổi lại cấu hình. Ngoài ra InnoDB đã được tái cấu trúc giúp tối ưu hóa hiệu suất bằng cách tận dụng tối đa công suất các hệ thống phần cứng đa lõi, đa luồng. Sau đây là các kết quả có được khi so sánh giữa MySQL5.5 và MySQL 5.1:

* Windows: tăng 1500% hiệu suất khi đọc và ghi dữ liệu, 500% khi chỉ đọc.
* Linux: tăng 360% hiệu suất khi dọc và ghi dữ liệu, 200% khi chỉ đọc.

Chính vì những ưu điểm đó mà chúng em quyết định chon MySQL để triển cơ sở dữ liệu cho hệ thống này.

### Apache Tomcat 7[[6](#Ale11)]

Apache Tomcat server được xây dựng trên mã nguồn mở trên nền tảng Java, các Java Web Application Container được tạo ra để thực thi các Servlet và Java Server Pages. Với sự phổ biến của nó hiện nay, các tình nguyện viên từ cộng đồng mã nguồn mở Java đang tích cự hỗ trợ và tăng cường phát triển.

Apache Tomcat hoạt động rất ổn định và hỗ trợ đầy đủ các tính năng của một Web Application Container. Tomcat cũng cung cấp những chức năng bổ sung khác giúp nó là một sự lựa chọn tốt để phát triển các ứng dụng Web như ứng dụng quản lý (Tomcat Manager application), Specialized Realm Implementations, Tomcat valves.

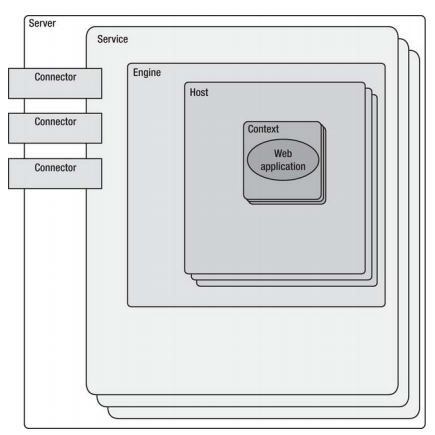
|  |  |  |  |
| --- | --- | --- | --- |
| Apache Tomcat | Servlet API | JSP API | JDK |
| 7.0 | 3.0 | 2.2 | 1.6 |
| 6.0 | 2.5 | 2.1 | 1.5 |
| 5.5 | 2.4 | 2.0 | 1.4 |
| 4.1 | 2.3 | 1.2 | 1.3 |
| 3.0 | 2.2 | 1.1 | 1.1 |

Bảng ‑: Các phiên bản Tomcat Server

**Kiến trúc Tomcat**

Một Servet Tomcat là thành phần cao nhất trong hệ thống Container phân cấp của Tomcat. Chỉ duy nhất một thực thể Tomcat được hoạt động trên một Máy ảo Java cụ thể(Java Virtual Machine). Như vậy, các ứng dụng Java chạy cùng một thiết bị vật lý sẽ cùng hoạt động trên một Server Tomcat. Vẫn có thể triển khai nhiều thực thể Tomcat trên cùng một thiết bị vậy lý, nhưng những thực thể này sẽ hoạt động trên những port phân biệt.

Một Server Tomcat bao gồm một nhóm các Application Containers tồn tại trong một hệ thống phân cấp xác định rõ ràng. Thành phần quan trọng nhất trong hệ thống phân cấp này là Catalina Servlet Engine. Catanila là Java Servlet Container được xây dựng theo các đặc tả trong Java Servlet API. Ví dụ như Tomcat 7 được xây dựng với Servlet API 3.0. Ta có mô tả mối quan hệ của các Tomcat Containers như sau:



Hình ‑: Kiến trúc Tomcat[6]

Trong trường hợp này Tomcat được chia thành các Container sau: Server, Service, Engine, Host, Context. Những Container này được cấu hình bằng cách sử dụng tập tin server.xml của Tomcat với cấu trúc như sau:

<Server>

<Service>

<Connector />

<Engine>

<Host>

<Context> </Context>

</Host>

</Engine>

</Service>

</Server>

#### Server

Là Container đầu tiên được tham chiếu trong đoạn code <Server></Server>. Server đại diện cho tất cả các Catanline Servlet Container và là phần quan trọng nhất trong một Server Tomcat. Trong Container này có thể chứa một hoặc nhiều <Service> Container.

#### Service

Container tiếp theo được tham chiếu trong <Service> </Service>. Là nơi chứa một hoặc nhiều các <Connector> cùng chia sẽ một <Engine> duy nhất.

#### Connector

Được định nghĩa các lớp thực hiện xử lý các yêu cầu và phản hồi lại cho các client.

#### Engine

Container thứ ba của Tomcat <Engine>. Một service chỉ có một Engine duy nhất. Engine sẽ xử lý tất cả các yêu cầu nhận được bởi các Connector chứa trong service chứa Engine này.

#### Host

<Host> xác dịnh các Virtual Host của từng Catalina <Engine> cụ thể. Host có thể chứa một hoặc nhiều ứng dụng cụ thể được tham chiếu trong các <Context>.

#### Context

Là yếu tố được sử dụng phổ biến nhất trong Tomcat. Một Context đại diện cho một ứng dụng Web hoạt động trong một Host. Số lượng Context trong một Host là không hạn chế.

## JSON và thư viện mã nguồn mở GSON

### Định nghĩa JSON[[30]](#footnote-30)

Json là một ngôn ngữ định dạng dữ liệu hoàn toàn độc lập, dùng để lưu trữ và trao đổi các thông tin dạng văn bản. Json giống với Xml nhưng cấu trúc đơn giản hơn, giúp cho việc phân tích, xử lý dữ liệu một cách nhanh hơn và dễ dàng hơn. Json được dựa trên cơ sở ngôn ngữ lập trình JavaScript, chuẩn ECMA-262 3rd tháng 12, 1999. Một thể hiện của json như sau:

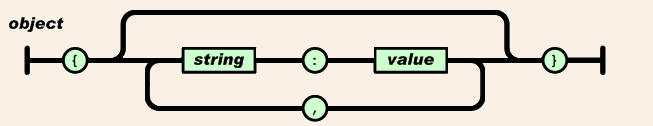
{"employees": [{ "firstName":"John", "lastName":"Doe" },{ "firstName":"Anna", "lastName":"Smith" },{ "firstName":"Peter", "lastName":"Jones" }]}

Json được xây dựng trên hai cấu trúc

* Là một tập hợp các cặp tên – giá trị tương tự như các đối tượng trong các ngôn ngữ lập trình khác.
* Một dãy các giá trị được xắp xếp tương tự như các array, vector, list hay sequence.

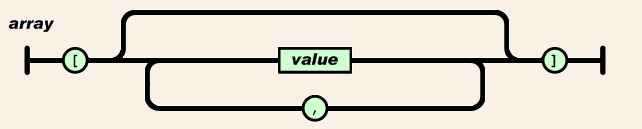
Hai hình thức biểu diễn của Json được sử dụng trong hệ thống này là:

- Object (Đối tượng): Là một dãy các cặp Tên/ Giá trị. Một Object được bắt đầu bằng dấu ngoặc đơn trái ({) và kết thúc bằng dấu ngoặc đơn phải (}). Tên và giá trị ngăn cách bởi dấu hai chấm (:). Mỗi cặp Tên/ Giá trị được ngăn cách bởi dấu phẩy (,).



Hình ‑: Biểu diễn json của một đối tượng

- Array(mảng): Array là một dãy các cặp giá trị, Array được bắt đầu bằng dấu ngoặc vuông trái ([) và kết thúc bằng dấu ngoặc vuông phải (]). Các giá trị được ngăn cách bởi dấu phẩy (,).



Hình ‑: Biểu diễn json của một mảng

Trong hệ thống quản lý taxi này định dạng dữ liệu sẽ sử dụng cho những Web Service chính là Json.

### Thư viện mã nguồn mở GSON[[31]](#footnote-31)

Gson là một thư viện Java được sử dụng trong việc chuyển đổi một đối tượng Java bất kỳ sang định dạng Json. Gson cũng hỗ trợ chuyển đổi từ một chuỗi định dạng Json sang đối tượng Java tương ứng. Đây là một thư viện mã nguồn mở được lưu trữ tại <http://code.google.com/p/google-gson>.

Những ưu điểm của Gson:

* Cung cấp các phương thức giúp việc chuyển đổi qua lại giữa đối tượng Java và Json trở nên đơn giản.
* Hỗ trợ chuyển đổi qua lại các đội tượng bất kỳ.
* Cho phép tùy chỉnh các đối tượng.
* Hỗ trợ các đối tượng phức tạp.
* Tạo ra đối tượng Json gọn nhẹ, dễ phân tích.

Lớp chính trong thư viện này là Gson, được khởi tạo đơn giản bằng cách gọi new Gson(). Ngoài ra còn có một lớp GsonBuilder sử dụng để tạo ra một đối tượng Gson với tùy chỉnh thích hợp. Các đối tượng Gson sẽ không duy trì trạng thái khi chuyển đổi qua lại giữa đối tượng Java và Json vì thế ta có thể sử dụng lại đối tượng Gson này trong các lần chuyển đổi tiếp theo. Dưới đây là ví dụ đơn giản cho thấy cách thức sử dụng đối tượng Gson:

* Chuyển đổi từ đối tượng Java sang Json

Gson gson = new Gson();

gson.toJson(1); ==> prints 1

gson.toJson("abcd"); ==> prints "abcd"

gson.toJson(new Long(10)); ==> prints 10

int[] values = { 1 };

gson.toJson(values); ==> prints [1]

* Chuyển đổi từ Json sang đối tượng Java

int one = gson.fromJson("1", int.class);

Integer one = gson.fromJson("1", Integer.class);

Long one = gson.fromJson("1", Long.class);

Boolean false = gson.fromJson("false", Boolean.class);

String str = gson.fromJson("\"abc\"", String.class);

String anotherStr = gson.fromJson("[\"abc\"]", String.class);

Trong hệ thống này Gson sẽ được sử dụng để chuyển đổi định dạng dữ liệu truy vấn từ cơ sở dữ liệu sang định dạng là Json trong các Web Service.

Phiên bản Google Gson Version 2.2.2 được phép tải miễn phí tại http://code.google.com/p/google-gson/downloads/detail?name=google-gson-2.2.2-release.zip được cập nhật ngày 2/7/2012.

## Công nghệ thoại IP và Tổng đài mã nguồn mở Trixbox

### Công nghệ thoại IP

Công nghệ thoại IP là công nghệ truyền/nhận các dữ liệu thoại (âm thanh) với thời gian thực bằng giao thức IP (Internet Protocol).

Mục đích của việc sử dụng công nghệ thoại IP là:

* Tiết kiệm được chi phí so với sử dụng hệ thống điện thoại thông thường đặc biệt là khi gọi điện thoại đường dài.
* Có thể đưa vào nhiều loại dịch vụ một cách dễ dàng như: quản lý cuộc gọi, hội thoại hội nghị …

PBX (Private Branch Exchange) là một tổng đài nội bộ có nhiệm vụ dịch số điện thoại và quản lý tất cả các cuộc gọi ra vào trong hệ thống.

Một số đặc điểm của PBX:

* Chức năng chính của tổng đài này là chuyển mạch, phân phối cuộc gọi trong toàn hệ thống.
* Những người sử dụng PBX dùng chung một số đường điện thoại ngoài để thực hiện các cuộc gọi ra bên ngoài.
* Có các dịch vụ hướng dẫn, dịch vụ Voice Mail, hệ thống tính cước.

IP PBX là hệ thống PBX chạy bằng phần mềm thực hiện một số nhiệm vụ nhất định và cung cấp những dịch vụ thoại trên mạng IP.

### Tổng đài mã nguồn mở Trixbox

Trixbox CE là hệ thống IP PBX mã nguồn mở trên nền tảng PBX (Private Branch eXchange) mã nguồn mở Asterisk. Trixbox xuất hiện từ tháng 11 năm 2004, Andrew Gillis đã xây dựng phiên bản đầu tiên với tên là Asterisk@Home. Mục tiêu của Andrew lúc này là làm cho việc cài đặt một hệ thống Askterisk nhanh hơn, dễ dàng hơn và hạn chế lỗi. Những năm sau đó Asterisk@Home có thêm những tính năng mới như Asterisk Management Portal giúp cho việc quản lý tổng đài trở nên dễ dàng hơn. Asterisk@Home đã trở thành một phần quan trọng cho những dự án phát triển tổng dài trên nền tảng Asterisk sau này. Vì cụm từ Asterisk này thuộc bản quyền của Digium Ltd nên Asterisk@Home đã được đổi tên thành Trixbox. Đến giữa năm 2007 Fonality đã hoàn thành Trixbox Pro, một phiên bản thương mại dựa trên những công nghệ trên sản phẩm PBXtra của Fonality. Để tránh sự nhầm lẫn với sản phẩm của Fonality, Trixbox được đổi tên thành TrixBox CE(community edition – phiên bản do cộng đồng phát triển).

Trixbox là một sản phẩm được xây dựng dựa trên nhiều thành phần mã nguồn mở khác nhau và được sự cấp phép của GPL. Ưu điểm của Trixbox là việc kết hợp những thành phần lại với nhau. Những thành phần này đã được cấu hình sẵn giúp cho việc cài đặt và vận hành tổng đài trở nên dễ dàng hơn.

* CentOS 5.2: Là phiên bản của hệ điều hành Red Hat Enterprise Linux. Đây chính là phần lõi của TrixBox.
* Asterisk 1.4: Asterisk là một PBX Engine, một gói phần mềm mã nguồn mở được viết bằng ngôn ngữ C chạy trên HĐH Linux (Linux based) đầy đủ chức năng của một hệ thống tổng đài IP-PBX với các phần cứng giao tiếp phù hợp được đóng góp từ những kỹ sư lập trình phần mềm mã nguồn mở ở khắp nơi trên khắp thế giới.
* FreePBX 2.5: Là giao diện giúp người quản trị hệ thống có thể quản lý chức năng của tổng đài bằng cách tạo ra những tập tin cấu hình sử dụng cho Asterisk.
* Flash Operator Panel (FOP): Flash Operator Panel là giao diện giúp cho người sử dụng có thể xem trạng thái của các đầu số trong tổng đài.
* Trixbox CE Dashboard: Là giao diện được phát triển bởi đội ngũ đã phát triển Trixbox, bao gồm các công cụ và tiện ích sử dụng trong việc quản lý và duy trì hoạt động của toàn bộ hệ thống.
* Automated Installation Tools: Tất cả các công cụ, hệ điều hành, Script và tập tin cấu hình sẽ được cài đặt và cấu một cách tự động bằng các sử dụng Setup Script của Trixbox.
* Festival Speech Engine: thực hiện chức năng Text-To-Speech
* Digium Card Auto-Config: Tự động cài đặt Digium Card.

Trixbox là một nền tảng ứng dụng với các công cụ và tiện ích được thiết kế với mục đích giúp cho việc quản lý hệ thống, cũng như các chức năng PBX trở nên dễ dàng hơn. Một số công cụ như sau:

* Package Manager: quản lý, cập nhật các gói cài đặt.
* Endpoint Manager: quản lý, cung cấp điện thoại trong hệ thống.
* Web MeetMe: quản lý các phòng hội thoại.
* RAID:Giám sát và duy trì hoạt động của các module.
* Lưu trữ dữ liệu về các cuộc gọi.
* Sao lưu và phục hồi các module

Một ưu điểm nữa của Trixbox là yêu cầu phần cứng để triển khai không đòi hỏi hỏi cấu hình cao. Cấu hình tối thiểu cho hoạt động thực tế chỉ cần một hệ thống Pentium 4 1GHz. Cấu hình thưc tế đề nghị là P4 2.0 GHz, 512Mb Ram, và tối thiểu 80Ghz. Với cầu hình này chúng ta thấy khi triển khai Trixbox sẽ tiết kiệm được rất nhiều chi phí.

## Tổng quan về SIP

Session Initation Protocol (SIP) là một giao thức báo hiệu lớp ứng dụng dùng để điều khiển các phiên giao tiếp như gọi thoại và gọi video thông qua internet. Giao thức này được sử dụng để tạo, chỉnh sửa, duy trì và kết thúc một hay nhiều phiên truyền thông đa phương tiện với một hay nhiều người tham gia. Các phiên truyền thông đa phương tiện này bao gồm thoại Internet, hội nghị, và các ứng dụng tương tự có liên quan đến truyền âm thanh, hình ảnh, và dữ liệu.

Sip là một giao thức thuộc tầng ứng dụng, nó có thể hoạt động trên giao thức TCP, UDP hay SCTP. SIP dựa theo ý tưởng của SMTP và HTTP. Nó được định nghĩa theo mô hình client-server trong đó bên gọi sẽ đóng vai trò là client để gửi các yêu cầu lên bên bị gọi đóng vai trò là server đáp ứng các yêu cầu này.

Sip hỗ trợ 5 vấn đề thiết lập và kết thúc truyền thông đa phương tiện bao gồm:

* Vị trí người sử dụng: xác định hệ thống đầu cuối sử dụng cho truyền thông, các user có thể di chuyển đến các vị trí khác nhau và truy cập vào hệ thống từ xa. Điều này tương tự như các hệ thống cung cấp bởi RAS trong H.323[[32]](#footnote-32).
* Tính sẵn sàng của người sử dụng: xác định khả năng sẵn sàng của bên nhận tham gia vào truyền thông.
* Năng lực của người sử dụng: xác định phương tiện và các thông số kỹ thuật để sử dụng SIP, chúng sử dụng giao thức SDP để thống nhất tham số truyền thông.
* Thiết lập phiên: thiết lập các thông số dùng để tạo ra phiên gọi ở cả bên gọi và bên nhận theo báo hiệu trực tiếp hay gián tiếp.
* Quản lý phiên gọi: bao gồm việc chuyển và kết thúc phiên, thay đổi các thông số phiên và gọi các dịch vụ.

### Lịch sử của SIP

Sip đầu tiên được thiết kế bởi Henning Schulzrinne[[33]](#footnote-33) và Mark Handley[[34]](#footnote-34) vào năm 1996.SIP được tiêu chuẩn hóa đầu tiên vào tháng 3 năm 1999 trong bộ tiêu chuẩn RFC 2453[[35]](#footnote-35). Tháng 11 năm 2000 được chấp nhận như là một giao thức điều khiển của 3rd Generation Partnership Project[[36]](#footnote-36) và là thành phần của kiến trúc IP Multimedia Subsystem[[37]](#footnote-37) trong dịch vụ truyền thông đa phương tiên trên mạng di động. Phiên bản cuối cùng được chỉ rõ trong RFC 3261[[38]](#footnote-38) vào tháng 5 năm 2002 với 22 lĩnh vự tiêu chuẩn mới của RFC và 21 bản dự thảo của các nhóm làm việc về internet, và hơn 50 văn bản cho các nhóm độc lập không phải là thành viên của các nhóm làm việc đệ trình.

### Các thành phần của giao thức SIP

Các thành phần của Sip được chia ra thành 2 loại:

Thành phần truy suất mạng hay các thiết bị điểm cuối: đó là những thành phần tương tác trực tiếp với người sử dụng để thiết lập cuộc gọi hay tương tác với server cung cấp thông tin. Thành phần này được gọi là User Agents.

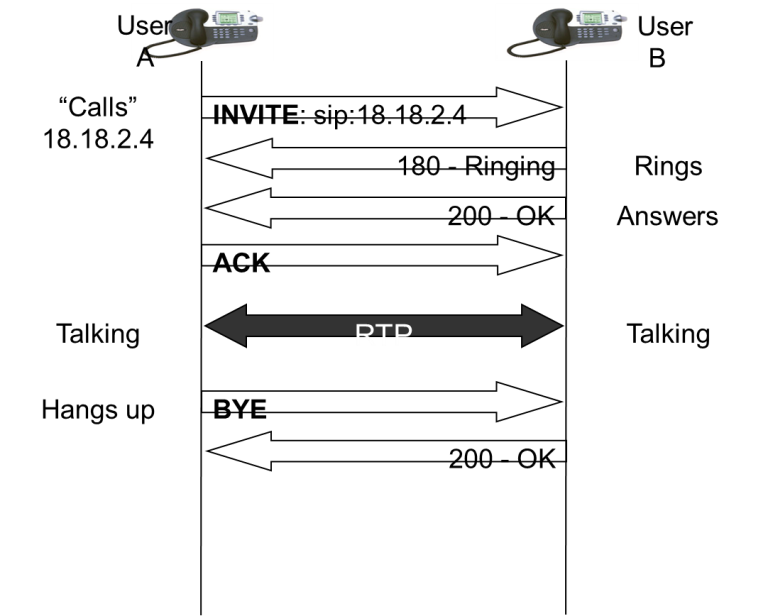
Thành phần lõi của mang Sip bao gồm các thành phần tham gia vào việc truyền, chuyển tiếp các thông điệp. Có 3 loại là: proxy server, registration server và redirect server.

* User Agents: có thể là một phần mềm hoặc thiết bị phần cứng. Các User Agent đóng 2 vai trò: khi nhận các yêu cầu thì nó sẽ xử lý và gửi trả các đáp ứng, lúc này nó đóng vai trò như một Server. Còn khi tạo các yêu cầu gửi đi và khi xử lý các đáp ứng nhận được từ các yêu cầu này thì nó đóng vai trò như một Client.
* Proxy server là thành phần chuyển các thông điệp Sip trực tiếp, Proxy server thực hiện công việc này dựa trên cơ sở dữ liệu được gọi là dịch vụ xác định vị trí. Cơ sở dữ liệu này kết hợp với các địa chỉ host cụ thể mà người dung đang đăng nhập. Cơ sở dữ liệu này được xây dựng bởi registration server nhờ quá trình đăng ký. Các yêu cầu mà proxy server nhận từ client có thể được xử lý hoặc thay mặt cho client gửi đến các server khác.
* Redirect server là server có chức năng chính là chấp nhận một yêu cầu SIP, ánh xạ địa chỉ trong yêu cầu thành một địa chỉ mới và trả lại địa chỉ này về cho client. Không giống như Proxy server, nó không khởi tạo một yêu cầu Sip và không chuyển các yêu cầu đến Server khác hay nhận cuộc gọi.
* Registrar server: Thực hiện quá trình đăng ký, khi UA muốn đăng ký địa chỉ Sip của nó với reigistrar server, nó tạo ra một yêu cầu REGISTER và gởi đến registrar. Registrar nhận và xử lý yêu cầu, lưu các thông tin vào cơ sở dữ liệu gồm địa chỉ SIP và địa chỉ host. Một người sử dụng có thể đăng nhập trên nhiều host. Các User Agent sẽ cập nhật thông tin một cách định kỳ bằng cách gởi lại yêu cầu REGISTER. Nếu sau các khoảng thời gian cụ thể, registrar server không nhận được bất kỳ yêu cầu REGISTER nào, nó sẽ xóa các mục tương ứng từ dịch vụ xác định vị trí.
* Location server:có chức năng giống như DNS server chứa thông tin về địa chỉ của các User Agent trên mạng SIP. Đầu tiên thì User Agent báo vị trí của nó về registrar server, tiếp theo thì registrar server sẽ lưu thông tin này trên location server.

### Thiết lập cuộc gọi qua SIP

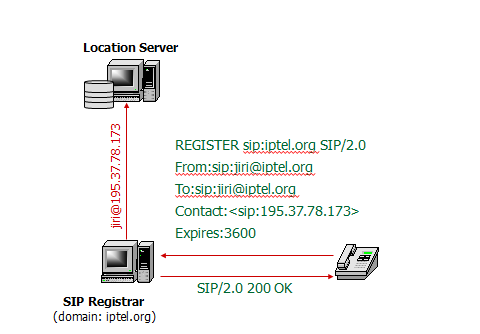
Thiết lập cuộc gọi qua giao thức Sip được chia ra thành các trường hợp sau:

* Sip cơ bản: Là hình thức kết nối của hai User Agent mà không phải thông qua bất kỳ server nào.



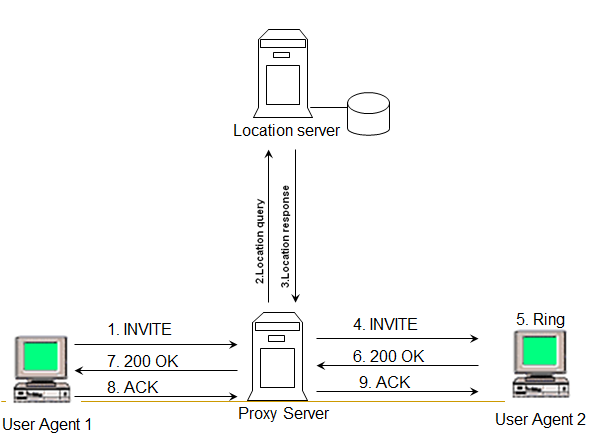
Hình ‑: Sip cơ bản

* Sip qua Registration: là hình thức UAC phải đăng ký qua registar server trước khi thực hiện gọi tới User Agent khác thông qua registrar server.

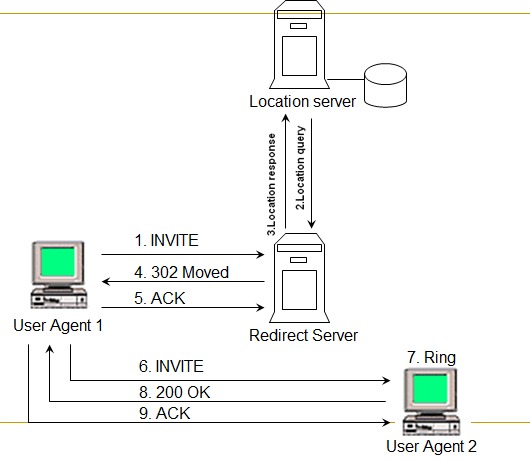


Hình ‑: Sip qua Registration

* Sip qua Proxy server: là hình thức User Agent kết nối với User Agent khác thông qua một proxy server làm trung gian để trung chuyển thông tin giữa 2 bên. Thông tin về User Agent bị gọi sẽ được lấy từ Location server.



Hình ‑: Sip qua Proxy server

* Sip qua redirect server: hình thức này User Agent sẽ nhận thông tin từ location server thông qua redirect server, sau đó sử dụng thông tin này để kết nối trực tiếp đến User Agent bị gọi.

Hình ‑: Sip qua redirect serve

### Thông điệp của SIP

Các thành phần tham gia vào một hệ thống dựa trên giao thức Sip sẽ báo hiệu cho nhau bằng cách gửi các thông điệp Sip. Bằng cách phân tích cấu trúc các thông điệp, chúng ta có thể quản lý các cuộc gọi một cách dễ dàng. Thông điệp của giao thức Sip có cấu trúc tương tự của giao thức HTTP, nó bao gồm hai loại chính là thông điệp yêu cầu và thông điệp trả lời. Dòng đầu tiên của một thông điệp yêu cầu là một phương thức đặc trưng và một Request URI, chỉ ra nơi mà thông điệp yêu cầu sẽ được gửi. Dòng đầu tiên của thông điệp trả lời là mã trả lời.

Đối với thông điệp yêu cầu, được nêu rõ trong RFC 3261[[39]](#footnote-39) sẽ gồm 6 phương thức chính là:

* REGISTER: Được sử dụng để User Agent có thể xác định được địa chỉ IP hiện tại của mình và URLs để nó có thể nhận cuộc gọi.
* INVITE: Được sử dụng để gửi yêu cầu kết nối trong việc thiết lập một phiên truyền thông đa phương tiện giữa hai người sử dụng.
* ACK: Dùng để xác nhận trong quá trình trao đổi thông tin.
* CANCEL: Dùng để kết thúc một yêu cầu đang chờ xử lý.
* BYE: Dùng để kết thúc một phiên làm việc giữa hai người dùng.
* OPTIONS: Dùng để nhận biết thông tin của hai bên với nhau mà không phải thiết lập cuộc gọi.
* Một phương thức mới của giao thức Sip được giới thiệu trong RFC 3262
* PRACK: Prack cải thiện độ tin cậy của mạng bằng cách thêm vào một hệ thống xác nhận các phản hồi tạm thời. Prack được gửi để đáp ứng với các phản hồi tạm thời để đáp ứng các yêu cầu được gửi lên từ User Agent.

Các loại mã đáp ứng của được chỉ rõ trong RFC 3261 gồm các loại sau đâu:

* 100 Trying: Yêu cầu đã được nhận tại một server ở chặng kế tiếp.
* 180 Ringing: Yêu cầu đổ chuông tại User Agent nhận.
* 181 Call Forwarding: Mã này cho thấy thông điệp đã được chuyển tiếp.
* 182 Queued for Service: Các ứng dụng có thể trì hoãn trả lời cuộc gọi cho đến khi nó đã phục vụ các cuộc gọi đang xếp hàng.
* 200 OK: Yêu cầu được thực thi thành công.
* 400 Bad request: Lỗi cú pháp thông điệp.
* 401 User: Yêu cầu xác thực trước khi thực hiện yêu cầu này.
* 403 Forbidden: Nhận được khi cố gắng gọi một số không được chấp nhận từ thuê bao của chúng ta.
* 404 Không tìm thấy user.
* 408 Request time-out: Nhận khi hết thời gian chờ.
* 500 Lỗi server.
* 600 Busy: Được trả về khi User Agent bị gọi đang ở trạng thái bận.
* 603 Decline: Được trả về khi User Agent bị gọi từ chối cuộc gọi.
* 604 Does not exist: Được trả về khi User Agent bị gọi không tồn tại.

## Kết luận

Qua chương này chúng ta đã nắm vững được những công nghệ để xây dựng hệ thống một cách chi tiết và rõ ràng. Bằng việc nắm vững những kiến thức, công nghệ nền tảng như đã trình bày ở trên, chúng em tiến hành chuyển sang việc phân tích thiết kế hệ thống quản lý taxi, phần này sẽ được trình bày trong chương tiếp theo.

# PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG MGTS-TAXI

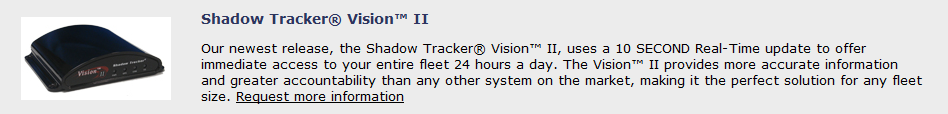
Nội dung chương này sẽ làm rõ những yêu cầu, khó khăn đặt ra cho bài toán quản lý và điều hành Taxi trong thực tế để từ đó có thể tiến hành phân tích những yêu cầu cần phải đáp ứng khi xây dựng hệ thống để từ đó đưa ra phương pháp giải quyết phù hợp trong việc xây dựng hệ thống này.

## Khảo sát

Hiện tại có một số hệ thống giám sát hành trình đã được triển khai. Chúng ta có thể tham khảo những hệ thống sau.

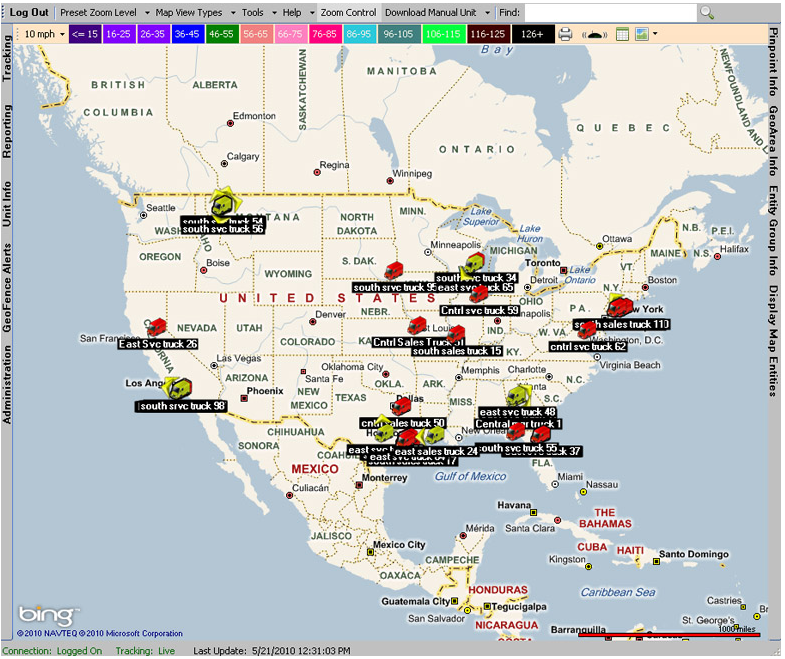
### GPS Vehicle Tracking Systems - The Shadow Tracker® Line[[40]](#footnote-40)

Là hệ thống kiểm soát các phương tiện vận tải. Bao gồm các sản phẩm có thể truyền tải thông tin từ xa bằng USB, sóng radio, và sóng di động thông qua một thiết bị chuyên dụng gọi là Shadow Tracker® thiết bị này sẽ cập nhật thông tin về trung tâm mỗi 10 giây.



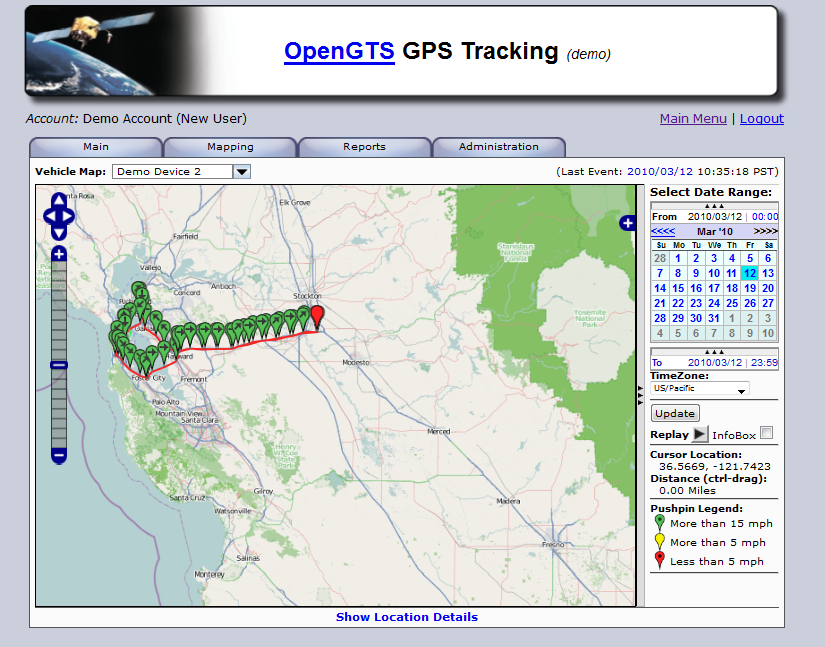
Hình ‑: Shadow Tracker®

Ngoài ra hệ thống này cũng cung cấp một phần mềm theo dõi theo thời gian thực bằng cách kết với Microsoft® Virtual Earth®. Nhà quản lý theo dõi vị trí và trạng thái các xe trong hệ thống.



Hình ‑: GPS Vehicle Tracking Systems

### *OpenGTS*™ - Open GPS Tracking System[[41]](#footnote-41)

Là dự án mã nguồn mở đầu tiên được thiết kế trong việc sử dụng GPS theo dõi các phương tiện giao thông như taxi, xe tải, phương tiện cá nhân, container, tàu, điện thoại di động trên trình duyệt web.

Hình ‑: Open GPS Tracking System

### Hệ thống điều phối taxi VIETMAP

#### Mô tả hệ thống

Hệ thống điều phối taxi VIETMAP[[42]](#footnote-42) là một hệ thống sử dụng quản lý taxi sử dụng thiết bị định vị GPS để xác định vị trí, chỉ dẫn đường đi, thông tin liên lạc 2 chiều giữa trung tâm điều phối và taxi. Hệ thống này gồm các tính năng chính như:

* Có khả năng kiểm soát vị trí xe, theo dõi thời gian thực từ đó phân phối khách hàng đến các taxi một cách hợp lý.
* Quản lý hoạt động, theo dõi lịch trình, báo cáo thống kê doanh số của từng taxi.
* Có khả năng hướng dẫn chỉ đường cho tài xế.

#### Các thiết bị cung cấp

Vietmap đã cung cấp ra thị trường nhiều loại thiết bị GPS với các tính năng khác nhau tùy vào nhu cầu của khách hàng gồm có các thiết bị là:

* GPS Tracking VT300: là thiết bị giám sát với các chức năng như giám sát hành trình chạy, thời gian đỗ, tốt độ xe. Ngoài ra thiết bị có thể kết nối với camera để chụp ảnh hoạt động của xe, ngoài ra còn có thể kế nối đến các cảm biến để thu thập số liệu truyền về cho tổng đài. Giá tham khảo trên thị trường là 3.300.000 đồng[[43]](#footnote-43).
* GPS Tracking VT10S: được sử dụng cho các phương tiện nhỏ dùng để giám sát vị trí, lộ trình hoạt động. Ngoài ra, thiết bị còn có thể Kết hợp với thiết bị phụ kiện là role chuyển mạch để tắt máy, khóa xe thông qua điện thoại di động. Giá tham khảo trên thị trường là 6.100.000 đồng[[44]](#footnote-44).
* GPS Tracking VT300s: có tính năng tương tự như thiết bị GPS Tracking VT300 nhưng bỏ bớt đi một số thành phần như: hình màu, bàn phím. Giá tham khảo trên thị trường là 4.799.000 đồng[[45]](#footnote-45).

### Chương trình TAXI-Gis

#### Mô tả hệ thống

TAXI-Gis[[46]](#footnote-46) là phần mềm quản lý và điều hành taxi được xây dựng bởi công ty cổ phần công nghệ GPSTRACK. TAXI-Gis được xây dựng trên nền internet gồm nhiều phần khách nhau để phục vụ nhiều người dùng với những nhóm chức năng khác nhau. Qua đó những chức năng thu được bao gồm:

* Sử dụng bản đồ số để quản lý xe Taxi.
* Theo dõi doanh thu, giám sát các hành vi gian lận.
* Bảo dưỡng xe.
* Hỗ trợ thông tin kinh doanh và phục vụ khách hàng.

#### Các thiết bị cung cấp

Công ty cổ phần công nghệ GPSTRACK đã đưa ra thiết bị giám sát hành trình AVL0901[[47]](#footnote-47) có chức năng thu thập dữ liệu thời gian thực chuyên sử dụng cho Taxi. Hỗ trợ tính năng GPRS/GSM để truyền tải thông tin về trung tâm quản lý. Ngoài ra còn hỗ trợ tình năng thoại và SMS.

### Ưu điểm của đề tài so với những hệ thống đã có

Thông qua việc tìm hiểu một số hệ thống quản lý taxi đã được triển khai trong thực tế mà tiêu biểu ở đây là hệ thống điều phối taxi VIETMAP và Chương trình TAXI-Gis. Chúng em đã rút ra một số ưu điểm của hệ thống mới sử dụng smartphone so với hệ thống cũ sử dụng các thiết bị định vụ đã có như sau:

* Dễ dàng nâng cấp phần mềm.
* Khả năng hiển thị và tương tác tốt hơn.
* Kết nối đến tổng đài sử dụng công nghệ HSDPA[[48]](#footnote-48) thay vì GPRS[[49]](#footnote-49).
* Tích hợp thêm được nhiều tiện ích hỗ trợ lái xe.
* Tích hợp giao tiếp giữa trung tâm qua voice IP.

Với những ưu điểm vượt trội hơn nhưng chi phí để trang bị một smartphone thay cho một thiết bị định vị GPS vào khoảng 3 đến 6 triệu đồng nên chi phí bỏ ra không chênh lệch nhiều.

Hiện tại việc sử dụng GPS được tích hợp trên Smartphone để thực hiện giám sát hành trình thay cho việc sử dụng các thiết bị giám sát hành trình chuyên dụng là một hướng tiếp cận mới của nhóm chúng em trên các các phương tiện vận chuyển mà cụ thể là phục vụ trong việc quản lý và điều hành taxi. Chúng ta sẽ phân tích những yêu cầu đối với hệ thống này trong phần tiếp theo.

## Xác định yêu cầu và các đối tượng trong hệ thống

Sau đây chúng ta sẽ tiến hành xác định các đối tượng và yêu cầu cần có với một hệ thống quản lý taxi.

### Yêu cầu đối với hệ thống

Đối với một hệ thống quản lý taxi luôn có những yêu cầu bắt buộc hệ thống phải đáp ứng được nếu muốn được đưa vào sử dụng trong thực tế.

Hệ thống của chúng em có khả năng đáp ứng được những yêu cầu sau đây:

* Cho phép người điều hành nhìn thấy bố cục tổng thể của toàn bộ xe Taxi đang hoạt động.
* Trạng thái của hệ thống: số xe có khách, số xe đang chờ đón khách, số xe trên đường đón khách, xe dừng, xe gặp sự cố, xe chạy quá tốc độ…
* Vị trí và tốc độ của xe.
* Tiếp nhận thông tin yêu cầu của khách hàng, tiến hành liệt kê các xe có khả năng đón khách theo ưu tiên về khoảng cách, điều phối xe đón khách.
* Khả năng liên lạc nội bộ giữa tài xế và trung tâm điều khiển, phòng hội thoại đựa trên tổng đài VOIP hoạt động trên nền internet 3G.
* Quản lý, điều hành hệ thống dựa trên nền tảng Web-based.

Đối với việc xây dựng client chuyên dụng để phục vụ các tài xế trong hệ thống sẽ có đáp ứng những yêu cầu sau:

* Là Smartphone chạy hệ điều hành Android có cài đặt ứng dụng client chuyên dụng của hệ thống quản lý taxi với các chức năng như sau
* Giao tiếp với server quản lý điều hành thông qua Web Service.
* Cập nhật trạng thái của xe với server như: xe đang có khách, xe đang đón khách, xe dừng, xe gặp sự cố, xe đã trả khách…
* Cập nhật tốc độ của xe.
* Hiện thị vị trí của xe, vị trí của khách được khi nhận lệnh đón khách từ server. Hướng dẫn chỉ đường từ vị trí của xe đến vị trí của khách hàng.

### Các đối tượng trong hệ thống

Sau khi nắm bắt được các yêu cầu của hệ thống, chúng ta sẽ phân tích những đối tượng hoạt động chủ yếu trong hệ thống này là tài xế, taxi, khách và nhân viên điều phối. Các tài xế, nhân viên điều phối sẽ được quản lý theo từng đội và chi nhánh riêng biệt. Mỗi nhân viên điều phối và tài xế sẽ được cấp một tài khoản dùng để đăng nhập và thực hiện các hoạt động của mình đối với hệ thống.

#### Nhân viên điều phối

Nhân viên điều phối là người trực tiếp làm việc với hệ thống này thông qua trình duyệt web để thực hiện việc quản lý, giám sát và điều phối các tài xế trong hệ thống. Mỗi nhân viên điều phối sẽ được cấp một tài khoản để đăng nhập vào hệ thống. Nhân viên chịu trách nhiệm nhận khác hàng thông qua điện thoại và thực hiện việc điều phối tài xế thích hợp đến phục vụ khách hàng. Quá trình một phiên điều phối diễn ra như sau:

* Nhân viên điều phối nhận yêu cầu phục vụ thông qua điện thoại từ khách hàng.
* Yêu cầu cung cấp thông tin về tên, số điện thoại, địa chỉ nơi khách hàng yêu cầu phục vụ kèm theo những yêu cầu đặc biệt như loại xe, số lượng xe đến đón nếu khách hàng có yêu cầu.
* Thực hiện việc điều phối bằng cách: thông qua bản đồ trạng thái của hệ thống sẽ hiển các tài xế có khả năng phục vụ khách, nhân viên tiến hành tìm địa chỉ của khách hàng trên bản đồ, hệ thống sẽ liệt kê ra những tài xế phù hợp nhất theo khoảng cách so với khách hàng để nhân viên có thể lựa chọn.
* Sau khi lựa chọn được tài xế phục vụ khách hàng, trạng thái của khách sẽ được cập nhật trên bản đồ. Nếu trong trường hợp vì lý do gì đó mà taxi không thể đến phục vụ khách được thì sẽ có một thông báo hiển thị trên bản đồ, khi đó nhân viên sẽ tiến hành lựa chọn một tài xế khác để đến phục vụ khách này.
* Ngoài ra, nhân viên điều phối còn được cung cấp khả năng gọi điện đến từng tài xế hoặc yêu cầu một nhóm tài xế theo đội, theo khu vực tham gia vào phòng hội thoại.
* Có khả năng xem lịch sử hoạt động của các xe taxi, các khách hàng đã phục vụ theo ngày, theo tháng.

#### Tài xế

Là người liên hệ trực tiếp phục vụ khách hàng trong hệ thống. Cái tài xế sẽ phải đăng nhập vào hệ thống thông qua ứng dụng client được cài đặt trên Smartphone chạy hệ điều hành Android. Khi tài xế đăng nhập vào hệ thống, client này sẽ có nhiệm vụ tự động cập nhật thông tin về vị trí, tốc độ và trạng thái của xe về trung tâm điều khiển đồng thời nhận thông tin của khách hàng do nhân viên tại trung tâm điều khiển điều phối xuống.Trong khi phục vụ khách tài xế có nhiệm vụ thông qua client báo cáo tình trạng của khách về trung tâm điều khiển.

Tài xế sẽ nhận khách hàng theo hai phương thức là nhận khách tự do và nhận khách từ trung tâm điều khiển.

Để thuận tiện trong việc liên lạc mỗi tài xế sẽ được cấp một đầu số dùng trong việc liên lạc nội bộ giữa tài xế và trung tâm điều khiển, tài xế và tài xế, tham gia vào các phòng hội thoại khi có yêu cầu từ trung tâm điều khiển.

#### Taxi

Tập hợp tất cả các xe taxi hoạt động trong hệ thống, được chia thành hai loại dựa theo số chỗ ngồi là loại bốn chỗ và loại bảy chỗ. Taxi sẽ được gán với những thông tin như vị trí, tốc độ và trạng thái.

Vị trí và tốc độ của xe sẽ được Smartphone Android ghi nhận lại dựa trên những sensor được hỗ trợ trên Smartphone đó. Vị trí sẽ được ghi nhận bằng GPS được trang bị trên điện thoại sẽ bao gồm hai tọa độ Latitute và Longitude, các tọa độ này sẽ được gửi về trung tâm điều khiển để sử dụng trong việc hiển thị vị trí của xe lên trên bản đồ Google Map.

Trạng thái của xe được quy định giúp cho nhân viên điều phối có thể giám sát được hoạt động của các xe khi hiển thị trên bản đồ và phải được cập nhật về trung tâm điều khiển. Xe sẽ có các trạng thái quy định sau đây:

* Xe sẵn sàng đón khách.
* Xe đang trên đường đón khách.
* Xe đang phục vụ khách.
* Xe phục vụ khách thành công.
* Xe phục vụ khách thất bại.
* Xe gặp sự cố.
* Xe mà tài xế chưa đăng nhập vào hệ thống.

#### Khách hàng

Là đối tượng phục vụ chính của các đối tượng còn lại. Khách hàng muốn được phục vụ thông qua trung tâm điều khiển sẽ liên hệ bằng điện thoại và cung cấp thông tin về địa chỉ của mình cho nhân viên điều phối sau đó chờ tài xế phục vụ.

Khách hàng sẽ được chia thành hai loại là khách do tài xế nhận tự do và khách do trung tâm điều khiển điều phối. Mỗi khách hàng sẽ được quy định những trạng thái sau đây:

* Khách đang chờ phục vụ.
* Khách đang được phục vụ.
* Khách phục vụ thành công.
* Khách phục vụ thất bại.

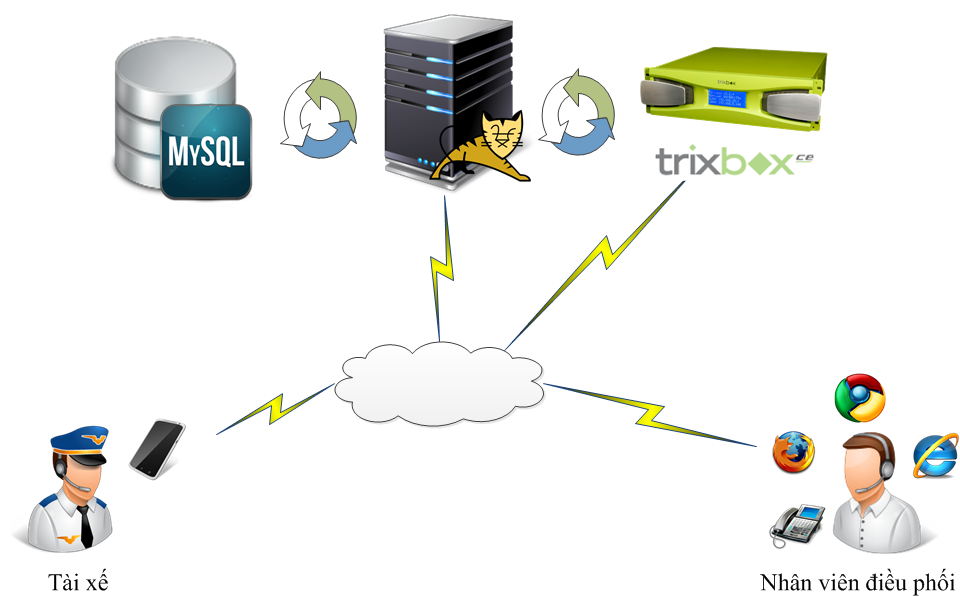
Sau khi đã xác định được yêu cầu đối với hệ thống và các đối tượng trong hệ thống chúng ta sẽ tiến hành mô tả thiết kế của hệ thống ở phần tiếp theo.

## Thiết kế hệ thống

Hệ thống quản lý taxi là hệ thống hoạt động theo mô hình Client – Server bao gốm hai thành phần chính sau đây:

* Server là một ứng dụng web cung cấp các chức năng:
  + Web Application cung cấp các chức năng quản lý và điều hành hệ thống sử dụng thông qua trình duyệt web nên rất tiện dụng và nhanh chóng.
  + Cung cấp cho nhân viên của hệ thống có cái nhìn tổng thể về hệ thống với việc hiển thị vị trí, trạng thái của các xe trong hệ thống, khách hàng lên bản đồ số, phân loại để hiển thị các xe theo từng trạng thái trên bản đồ số giúp tăng hiệu quả trong việc giám sát hoạt động của tài xế.
  + Lưu trữ lịch sử hoạt động bao gồm trạng thái, vị trí và khách hàng của tài xế.
  + Tiếp nhận, xử lý thông tin yêu cầu phục vụ của khách hàng, tiến hành lựa chọn các xe hợp lý theo mức độ ưu tiên để đón khách, cung cấp thông tin khách hàng cho tài xế xe được chọn giúp cho quá trình điều phối diễn ra nhanh chóng, giảm được tình trạng xe phải di chuyển không khách.
  + Hệ thống RESTful Web Service cho phép giao tiếp với ứng dụng client mà tài xế sử dụng giúp client cập nhật trạng thái, vị trí, gửi thông báo đến tài xế khi có khách hàng, cập nhật dữ liệu lên bản đồ số.
  + Tổng đài Trixbox được triển khai phục vụ việc liên lạc giữa nhân viên và tài xế trong hệ thống có thể thay thế cho việc liên lạc bằng bộ đàm truyền thống.
  + Triển khai hệ thống liên lạc thông qua VOIP, tích hợp Soft Phone trên Web Application giúp cho việc liên lạc được dễ dàng, nhanh chóng.
* Client là ứng dụng được cài đặt trên Smartphone chạy hệ điều hành android được trang bị cho các cho tài xế xe taxi với các chức năng:
  + Hiện thị vị trí xe, chức năng tìm đường dựa trên google map.
  + Cung cấp thông tin về vị trí, trạng thái xe, trạng thái khách hàng cho trung tâm điều khiển.
  + Nhận điều phối và thông tin khách hàng từ trung tâm điều khiển, hiển thị vị trí khách và thực hiện việc tìm dường, chỉ dẫn đến vị trí khách.
  + Chức năng gọi VOIP với tài khoản được cung cấp thông quan tổng đài Trixbox của trung tâm diều khiển trên nền 3G.

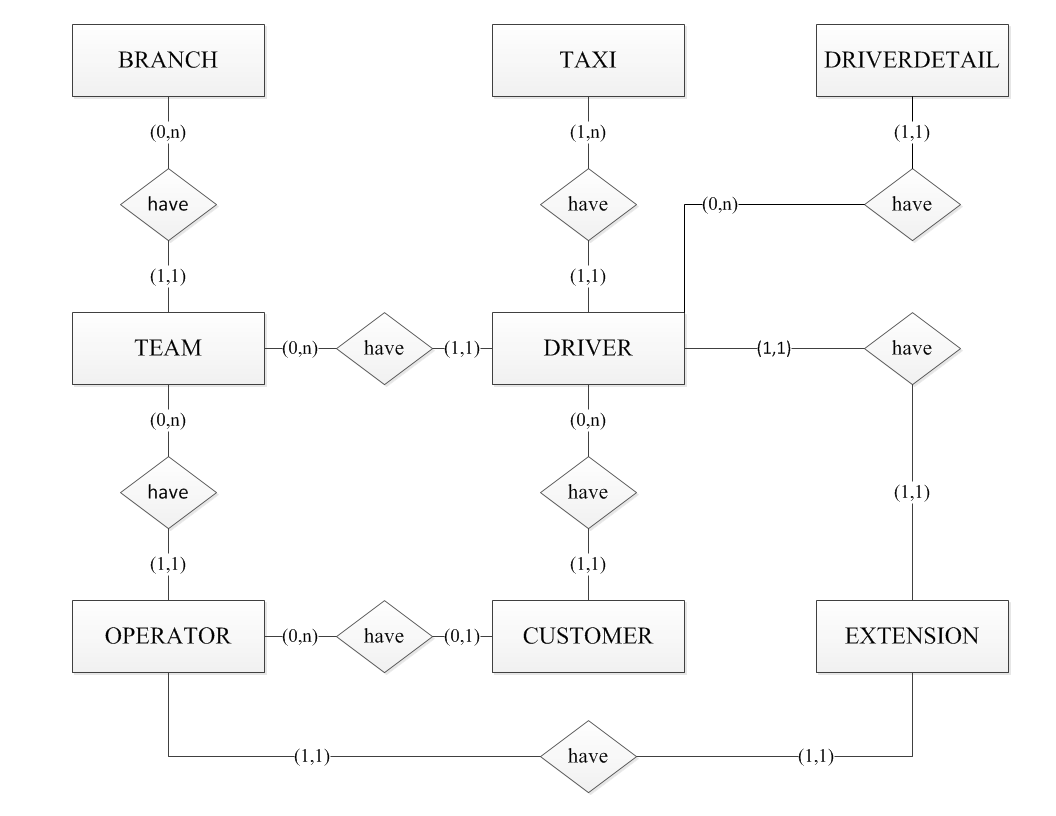
Thiết kế hệ thống được mô tả như hình sau:



Hình ‑: Mô tả thiết kế hệ thống

## Mô hình dữ liệu

Sau khi phân tích được những đối tượng và yêu cầu đối với hệ thống, chúng em đưa ra mô hình thực thể kết hợp ở mức quan niệm để biểu diễn mối quan hệ giữa các đối tượng trong hệ thống.



Hình ‑: Mô hình thực thể kết hợp

Như trong mô hình ta thấy, các tài xế (DRIVER) và nhân viên điều phối (OPERATOR) sẽ được quản lý trong các đội (TEAM), các đội này sẽ thuộc một chi nhánh (BRANCH) cụ thể. Để thực hiện việc liên lạc nội bộ thông qua tổng đài VOIP trong hệ thống mỗi tài xế và nhân viên điều phối sẽ được cấp một số liên lạc (EXTENSION). Mỗi tài xế sẽ được trang bị một taxi trong hệ thống, tuy nhiên một taxi sẽ có thể có nhiều tài xế sử dụng tùy thuộc vào ca làm việc của tài xế. Các hoạt động của tài xế sẽ được ghi nhận lại để nhân viên điều phối có thể xem lại lịch sử hoạt động của tài xế. Các khách hàng (CUSTOMER) trong hệ thống sẽ được phục vụ theo hai trường hợp: do nhân viên điều phối cho tài xế và do tài xế tự do nhận khách.

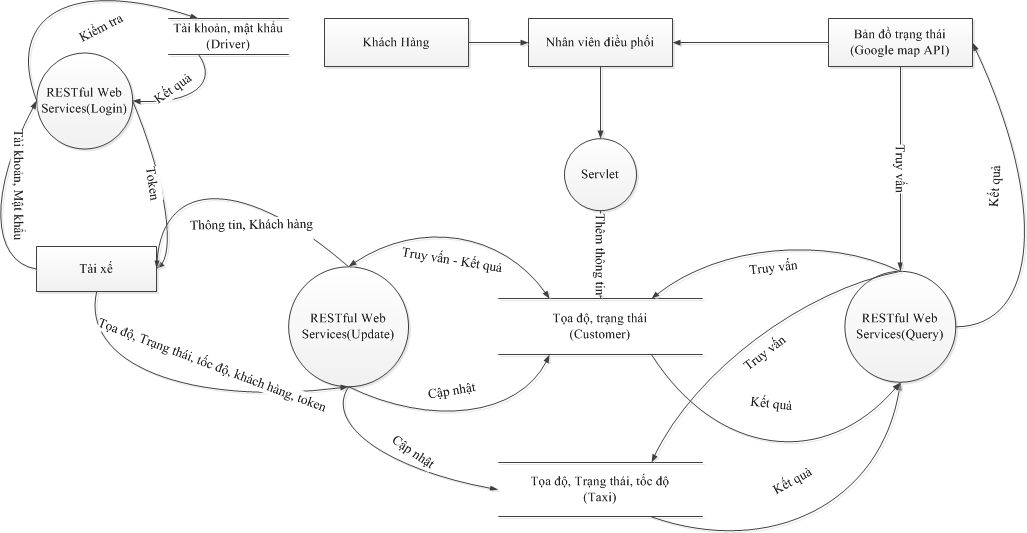
## Thiết kế Web Service

Trong hệ thống quản lý taxi này đòi hỏi phải thực hiện việc liên kết giữa trung tâm điều khiển và client được tài xế, cũng như việc truy suất dữ liệu từ MySQL để cung cấp cho Google Map API trong việc hiển thị vị trí và trạng thái của các đối tượng trong hệ thống lên bản đồ… Để thực hiện những chức năng này chúng em sẽ xây dựng các RESTful Web Service nhằm phục vụ cho những công việc sau đây:

* Thực hiện kiểm soát việt đăng nhập, đăng xuất khỏi hệ thống của các tài xế.
* Thực hiện truy vấn dữ liệu về các taxi, khách hàng, lịch sử hoạt động của taxi trong hệ thống để hỗ trợ việc hiển thị của các bản đồ trong hệ thống.
* Thực hiện việc liên kết giữa trung tâm điều khiển và client, ghi nhận những thông tin do client cập nhật lên như vị trí, trạng thái, tốc độ của taxi… kiểm tra xem tài xế có nhận được yêu cầu đón khách từ nhân viên điều phối hay không, nếu có sẽ phản hồi lại cho tài xế thông tin của khách bao gồm tên, số điện thoại, tọa độ và địa chỉ của khách hàng. Ghi nhận trạng thái của khách hàng. Lưu trữ lại các thông tin cập nhật vào cơ sở dữ liệu phục vụ cho việc truy vấn dữ liệu để hiển thị lên bản đồ và việc hiển thị lịch sử hoạt động taxi.
* Thực hiện việc lựa chọn xe phù hợp nhất để đón khách.

Định dạng dữ liệu sử dụng trong các Web Service này là json. Chi tiết việc xây dựng những Web Service này sẽ được trình bày trong chương tiếp theo.

## Mô hình xử lý của hệ thống

Sau khi đã nắm bắt được những đối tượng và yêu cầu cần có đối với hệ thống này, chúng ta đưa ra mô hình xử lý của hệ thống như được mô tả trong sơ sồ sau:

Hình ‑: Mô hình xử lý của hệ thống

Hệ thống hoạt động theo từng bước sau đây:

* Tài xế đăng nhập vào hệ thống bằng các gửi thông tin tài khoản và mật khẩu để Web Service kiểm tra kiểm tra đăng nhập.
* Nếu thông tin đăng nhập là hợp lệ tài xế sẽ được hệ thống cấp phát một token, tài xế sẽ sử dụng token này để cập nhật thông tin và truy suất những tài nguyên của hệ thống.
* Các tài xế sẽ phải cập nhất thông tin trạng thái, vị trí định kì về trung tâm điều khiển thông qua Web Service.
* Khi có khách gọi điện đến trung tâm điều khiển, nhân viên tại đây sẽ tiến hành lựa chọn taxi và thực hiện lệnh điều phối khách.
* Trong khi gửi thông tin cập nhật về trung tâm điều khiển, Web Service sẽ kiểm tra xem tài xế này có nhận được lệnh điều phối từ nhân viên điều phối hay không, nếu có Web Service này sẽ gửi thông tin của khách hàng về cho tài xế này.
* Tài xế khi nhận thông tin khách hàng sẽ tiến hành phục vụ, trong suốt quá trình này tài xế sẽ cập nhật thông tin trạng thái của khách hàng về trung tâm điều khiển.
* Những thông tin cập nhật tài xế gửi về trung tâm điều khiển sẽ được lưu trữ lại trong cơ sở dữ liệu, để nhân viên điều phối có thể giám sát vị trí, trạng thái của xe hệ thống sẽ sử dụng một Web Service để truy vấn những thông tin này sau đó chuyển những thông tin này sang Google Map API để hiển thị lên bản đồ. Bản đồ sẽ được cập nhật liên tục trạng thái và vị trí của các xe hoạt động trong hệ thống.

Như vậy chúng ta đã nắm được cơ bản những xử lý mà hệ thống cần phải thực hiện. Tiếp theo chúng ta sẽ tiến hành xây dựng client ở phần tiếp theo.

## Giải pháp hiển thị bản đồ

Giải pháp về bản đồ là một trong những thành phần quan trọng trong hệ thống này. Google Maps API có khả năng hỗ trợ lập trình viên khai thác sức mạnh của Google Maps để sử dụng trong các ứng dụng riêng của mình. Cụ thể đây là công cụ giúp hiển thị dữ liệu cụ thể của từng ứng kết hợp với bản đồ rất hiệu quả. Và trong hệ thống này, chúng em sử dụng Google Maps API để hiển thị dữ liệu về vị trí và trạng thái của các taxi lên trên bản đồ giúp cho người điều hành thuận tiện hơn trong việc quản lý, giám sát và điều hành các xe taxi trong hệ thống.

Google Maps được giới thiệu trên Blog post của google trong tháng 2 năm 2005. Ưu điểm tại thời điểm này của Google Maps là cho phép người dùng kéo chuột để di chuyển bản đồ. Chi phí cho các giải pháp về bản đồ lúc này rất tốn kém và đòi hỏi phải có một máy chủ bản đồ đặc biệt và hầu như chỉ được tạo ra nhằm đáp ứng các yêu cầu riêng của Google do đó nó chứ rất nhiều tính năng không cần thiết làm cho API trên nên cồng kềnh hơn.Từ năm 2005, sự bùng bổ của các thiết bị di động khiến API cũ trở nên chậm chạp và không còn phù hợp cho những thiết bị này nữa. Chính vì vậy các nhà phát triển giải pháp bản đồ đã quyết định xây dưng mới lại API này. Trong quá trình phát triển Google đã cung cấp các Api giúp các lập trình viên dễ dàng tiếp cận và tích hợp bản đồ vào trong ứng dụng của mình. Giúp cho Google Maps trở thành giải pháp bản đồ được sử dụng phổ biến như ngày nay.

Trong hệ thống chúng em sử dụng hai Api cụ thể là Google Maps JavaScript API v3 và Google Maps Android API v2.

Phiên bản 3 của Google Maps JavaScript API là phiên bản được làm mới hoàn toàn so với các phiên bản trước của Google. Api này cho phép nhúng trực tiếp Google Maps vào trang web riêng của chúng ta, nó được thiết kế đặc biệt để hoạt động nhanh và nhiều tính năng hơn cho các thiết bị di động và trình duyệt web. Api này cung cấp các tiện ích giúp cho việc thao tác bản đồ (Giống như trang http://maps.google.com), việc xử lý nội dung trên bản đồ được thực hiện thông qua các Web Service do Google cung cấp giúp chúng ta có thể tạo ra các ứng dụng bản đồ một cách dễ dàng và mạnh mẽ hơn.

Google Maps JavaScript thực chất là sự kết hợp của HTML, CSS và JavaScript. Bản đồ được chia thành từng ô nhỏ là những hình ảnh của bản đồ. Những ảnh này được tải về bằng Ajax sau đó chèn vào một thẻ Div trong trang HTML. Khi người dùng thao tác với bản đồ, API sẽ gửi tọa độ mới và mức độ zoom của bản đồ để gọi Ajax để lấy về những hình ảnh phù hợp với thao tác trên bản đồ.

API này bao gồm các tập tin JavaScript có chứa các lớp với các thuộc tính và phương thức để chúng ta có thể sử dụng khi thao tác với bản đồ. API này được thiết kế chú trọng đến việc tăng hiệu suất cả khi sử dụng trên thiết bị di động và trên các máy tính. Để thực hiện điều này, với phiên bản cũ trình duyệt phải tải và thực thi rất nhiều Script trước khi có thể hiển thị được bản đồ thì phiên bản mới được module hóa sẽ tải về những code cần thiết trước, hiển thị bản đồ sau đó tải tiếp những thứ còn lại sau khi đã hiện thị bản đồ.

Hệ thống tọa độ được sử dụng trong Google Maps là Word Geodetic System 84 (WGS 84), cùng chung hệ thống với Global Postioning System. Vì vậy sẽ rất tiện lợi khi kết hợp bản đồ Google maps với GPS trên Smartphone vì chúng ta không cần tính toán chuyển đổi tọa độ. Để xác định một điểm trong hệ tọa đồ này ta sẽ sử dụng vĩ độ(Latitude) và kinh dộ (Longtitude), hai giá trị này thể hiện bằng số thập phân. Các giá trị vĩ độ luôn đứng trước kinh độ (vĩ độ, kinh độ). Ví dụ như (40.714,-47.005)



Hình ‑: Word Geodetic System 84[[7](#Sve10)]

## Mô tả ứng dụng client trên Android

### Giao diện tương tác người dùng

Giao diện chính của chương trình là bản đồ của Google, ngoài ra để đáp ứng mục đích của ứng dụng. Giao diện còn được thêm vào một số nút và các tùy chọn trong Menu.

Chương trình chỉ bao gồm 2 activity chính là activity được sử dụng dụng để đăng nhập và activity chính dùng để hiển thị bản đồ và thực hiện các chức năng khác. Việc hiện thị các thông tin hay thông báo đều thực hiện qua các dialog.

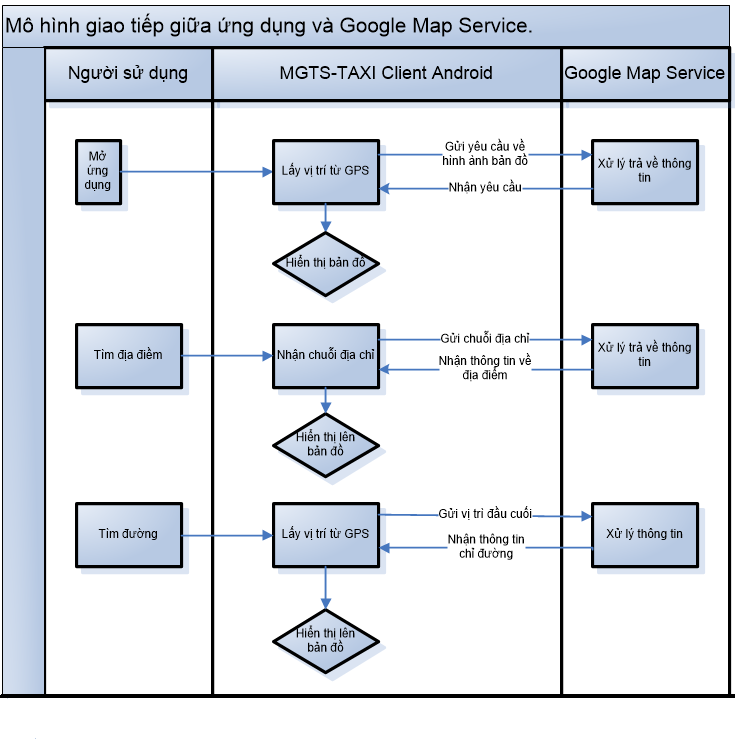


Hình ‑: Giao diện tương tác người dùng

### Mô hình xử lý của ứng dụng

Ứng dụng được kết nối đến 2 server chính là Google Map Service và MGTS-TAXI Web Service. Vì vậy, muốn hoạt động được thì thiết bị cần kết nối đến internet. Sau đây là quá trình xử lý đối với từng web service:

* Google Map Service: Cung cấp hình ảnh, dữ liệu để có thể hiễn thị Google cũng như các dịch vụ đi kèm như: tìm kiếm địa điểm, chỉ đường,…. Chúng ta có thể tóm tắt quá trình kết nối qua mô hình sau:



Hình ‑: Mô hình giao tiếp giữa ứng dụng và Google Map Service.

MGTS-TAXI Web Service: là web server được phát triển bởi nhóm thực hiện nhằm mục đích lưu trữ các thông tin về vị trí, xử lý việc đón trả khách của các phương tiện taxi. Chính vì mục đích này nên ứng dụng kết nối liên tục đến web server này để gửi các thông tin về vị trí lên server. Quá trình giao tiếp với Server nhằm thực hiện 3 hoạt đông chính bao gồm:

* + Nhận khách từ Server
  + Bắt khách trên đường đi
  + Trả khách tại địa điểm yêu cầu.

 Chúng ta có thể tóm tắt quá trình này qua mô hình sau:

Hình ‑: Mô hình giao tiếp giữa ứng dụng và Taxi Manage Web Service

## Giải pháp VOIP

Trong quá trình quản lý và giám sát taxi việc thực hiện liên lạc giữa tài xế và nhân viên điều phối được thực hiện một cách thường xuyên và giữ vai trò quan trọng. Các nhà cung cấp dịch vụ taxi hiện tại việc liên lạc giữa trung tâm điều khiển và tài xế chỉ được thực hiện thông qua hệ thống bộ đàm. Trong hệ thống của chúng em sẽ đưa ra giải pháp thay thế cho việc sử dụng bộ đàm bằng cách triển khai một tổng đài VOIP để các nhân viên và tài xế.

Việc sử dụng liên lạc bằng bộ đàm sẽ gặp khó khăn trong việc liên lạc cá nhân một gọi một, gọi theo nhóm dựa theo từng khu vực cụ thể, và chi phí đầu tư để triển khai khá cao. Vì vậy chúng em triển khai một tổng đài mã nguồn mở Trixbox trong hệ thống này để thực hiện việc liên lạc. Việc triên khai tổng đài Trixbox có những thuận lợi sau đây:

* Là tổng đài mã nguồn mở nên chi phí triển khai thấp. Không phải trang bị hệ thống bộ đàm trên mỗi taxi vì Client hoạt động trên Smartphone android có thể thực hiện cuộc gọi thông qua tổng đài VOIP.
* Cung cấp đầu số cho mỗi nhân viên trong hệ thống giúp việc liên lạc nội bộ, từng thành phần gọi nhau một cách dễ dàng.
* Kết hợp với bản đồ vị trí các xe, giúp cho nhân viên điều phối có thể gom nhóm các xe trong cùng khu vực để yêu cầu tham gia vào phòng hội thoại.

Với những ưu điểm của Trixbox như đã giới thiệu ở chương hai chúng em tin tưởng rằng tổng đài này hoạt động tốt giúp chúng ta có thêm lựa cho trong việc liên lạc giữa các thành phần trong hệ thống.

## Tổng kết

Qua chương này chúng ta đã thấy được chi tiết hơn những yêu cầu và đối tượng trong hệ thống. Bước đầu xác định các thành phần, các xử lý chính của hệ thống. Chúng ta sẽ đi vào phần thiết kế chi tiết của hệ thống ở chương tiếp theo.

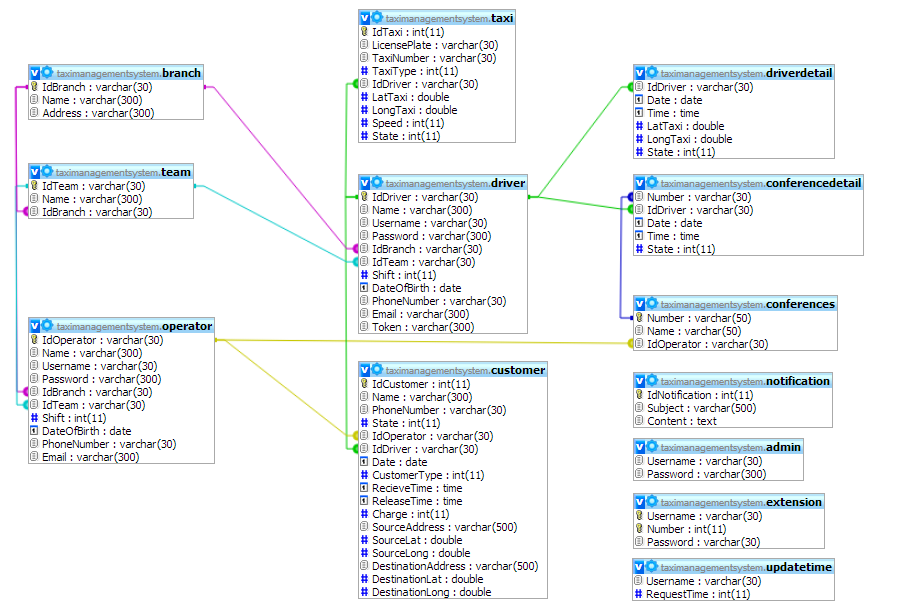
# HIỆN THỰC HỆ THỐNG MGTS-TAXI

Sau khi đã nắm vững cơ sở lý thuyết và tiến hành phân tích yêu cầu, đưa ra phương pháp giải quyết bài toán mà đề tài đặt ra, chương này sẽ trình bày chi tiết việc thiết kế, xây dựng và mô hình triển khai hệ thống.

## Xây dựng cơ sở dữ liệu

### Mô hình dữ liệu

Dữ diệu của hệ thống được lưu trữ tại cơ sở dữ liệu taximanagementsystem trên hệ quản trị cơ sở dữ liệu MySQL Server và được thiết kế như sau:



Hình ‑: Cơ sở dữ liệu

### Chi tiết cơ sở dữ liệu

#### Quan hệ ADMIN

Ý nghĩa: lưu giữ thông tin chứng thực đăng nhập của quản trị hệ thống.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| **Username** | Tài khoản đăng nhập hệ thống | varchar(30) | notNull |  |
| Password | Mật khẩu đăng nhập | varchar(300) | notNull |  |

Bảng ‑: Quan hệ ADMIN

#### Quan hệ BRANCH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| **IdBranch** | Mã chi nhánh | varchar(30) | notNull |  |
| Name | Tên chi nhánh | varchar(300) | notNull |  |
| Address | Địa chỉ chi nhánh | varchar(300) | notNull |  |

Ý nghĩa: lưu trữ thông tin các chi nhánh trong hệ thống.

Bảng ‑: Quan hệ BRANCH

#### Quan hệ TEAM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| **IdTeam** | Mã đội | varchar(30) | notNull |  |
| Name | Tên đội | varchar(300) | notNull |  |
| IdBranch | Mã chi nhánh của đội | varchar(30) | notNull |  |

Ý nghĩa: lưu trữ thông tin về các đội xe trong hệ thống.

Bảng ‑: Quan hệ TEAM

#### Quan hệ DRIVER

Ý nghĩa: lưu trữ danh sách các tài xế trong hệ thống bao gồm các thông tin cá nhân, thông tin tài khoản đăng nhập hệ thống của tài xế.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| **IdDriver** | Mã tài xế | varchar(30) | notNull |  |
| Name | Tên tài xế | varchar(300) | notNull |  |
| Username | Tài khoản đăng nhập hệ thống | varchar(30) | notNull |  |
| Password | Mật khẩu đăng nhập hệ thống | varchar(300) | notNull |  |
| IdBranch | Mã chi nhánh của tài xế | varchar(30) | notNull |  |
| IdTean | Mã đội của tài xế | varchar(30) | notNull |  |
| Shift | Ca làm việc | int | notNull |  |
| DateOfBirth | Ngày sinh | date | notNull |  |
| PhonneNumber | Số điện thoại | varchar(30) | notNull |  |
| Email | Email tài xế | varchar(300) | notNull |  |
| Token | Token kiểm tra trạng thái đăng nhập của tài xế | varchar(300) | notNull |  |

Bảng ‑: Quan hệ DRIVER

#### Quan hệ DRIVERDETAIL

Bảng ‑: Quan hệ DRIVERDETAIL

Ý nghĩa: Lưu trữ thông tin về lịch sử hoạt động của tài xế như ngày, giờ, vị trí, trạng thái của xe taxi do tài xế điều khiển sẽ được tự động cập nhật khi tài xế đăng nhập hệ thống.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| **IdDriver** | Mã tài xế | varchar(30) | notNull |  |
| Date | Ngày hoạt động của tài xế | date | notNull |  |
| Time | Thời gian hoạt động của tài xế | time | notNull |  |
| LatTaxi | Tọa độ Latitude của tài xế | double | notNull |  |
| LongTaxi | Tội độ Longitude của tài xế | double | notNull |  |
| State | Trạng thái của tài xế | int | notNull |  |

#### Quan hệ OPERATOR

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| **IdOperator** | Mã nhân viên điều phối | varchar(30) | notNull |  |
| Name | Tên nhân viên điều phối | varchar(300) | notNull |  |
| Username | Tài khoản đăng nhập hệ thống | varchar(30) | notNull |  |
| Password | Mật khẩu đăng nhập hệ thống | varchar(300) | notNull |  |
| IdBranch | Mã chi nhánh của nhân viên | varchar(30) | notNull |  |
| IdTean | Mã đội của nhân viên | varchar(30) | notNull |  |
| Shift | Ca làm việc | int | notNull |  |
| DateOfBirth | Ngày sinh | date | notNull |  |
| PhonneNumber | Số điện thoại | varchar(30) | notNull |  |
| Email | Email tài xế | varchar(300) | notNull |  |

Ý nghĩa: Lưu trữ thông tin, tài khoản đăng nhập của các nhân viên điều phối trong hệ thống.

Bảng ‑: Quan hệ OPERATOR

#### Quan hệ TAXI

Bảng ‑: Quan hệ TAXI

Ý nghĩa: lưu trữ thông tin của các xe taxi hoạt động trong hệ thống. Vị trí của các xe sẽ được cập nhật khi tài xế điều khiển xe đó đăng nhập vào hệ thống.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| **IdTaxi** | Mã xe taxi trong hệ thống | int | notNull |  |
| LicensePlate | Biển số xe taxi | varchar(30) | notNull |  |
| TaxiNumber | Số hiệu taxi | varchar(30) | notNull |  |
| TaxiType | Loại xe taxi | int | notNull |  |
| IdDriver | Mã tài xế của taxi | varchar(30) | notNull |  |
| LatTaxi | Tọa độ Latitude của taxi | double | notNull |  |
| LongTaxi | Tọa độ Longtitue của taxi | double | notNull |  |
| Speed | Tốc độ của xe | int | notNull |  |
| State | Trạng thái của xe | int | notNull |  |

#### Quan hệ CUSTOMER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| **IdCustomer** | Mã số khách hàng | int | notNull |  |
| Name | Tên khách hàng | varchar(300) | notNull |  |
| PhoneNumber | Số điện thoại khách hàng | varchar(30) | notNull |  |
| State | Trạng thái của khách hàng | int | notNull |  |
| IdOperator | Mã nhân viên điều phối | varchar(30) | notNull |  |
| IdDriver | Mã tài xế phục vụ | varchar(30) | notNull |  |
| Date | Ngày phục vụ khách | date | notNull |  |
| CustomerType | Phân loại khách hàng | int | notNull |  |
| RecieveTime | Thời gian nhận khách | time | notNull |  |
| ReleaseTime | Thời gian trả khách | time | notNull |  |
| Charge | Chí phí | int | notNull |  |
| SourceAddress | Địa chỉ đón khách | varchar(500) | notNull |  |
| SourceLat | Tọa độ Latitude nơi đón khách | double | notNull |  |
| SourceLong | Tọa độ Longitude nơi đón khách | double | notNull |  |
| DestinationAddress | Địa chỉ nơi trả khách | varchar(500) | notNull |  |
| DestinationLat | Tọa độ Latitude nơi trả khách | double | notNull |  |
| DestinationLong | Tọa độ Longitude nơi trả khách | double | notNull |  |

Ý nghĩa: lưu trữ thông tin trạng thái, vị trí, thông tin trong quá trình phục vụ khách hàng.

Bảng ‑: Quan hệ CUSTOMER

#### Quan hệ NOTIFICATION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| **IdNotification** | Mã số thông báo | int | notNull |  |
| Subject | Chủ đề thông báo | varchar(500) | notNull |  |
| Content | Nội dung thông báo | text | notNull |  |

Ý nghĩa: lưu trữ nội dung các thông báo trong hệ thống

Bảng ‑: Quan hệ NOTIFICATION

#### Quan hệ EXTENSION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| **Username** | Tài khoản của nhân viên | varchar(30) | notNull |  |
| Number | Số nội bộ của nhân viên | int | notNull |  |
| Password | Mật khẩu xác thực số nội bộ với trixbox | varchar(30) | notNull |  |

Ý nghĩa: lưu trữ thông tin về tài khoản SIP trong hệ thống VOIP nội bộ.

Bảng ‑: Quan hệ EXTENSION

#### Quan hệ CONFERENCES

Ý nghĩa: Lưu trữ thông tin các phòng hội thoại trong hệ thống.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| **Number** | Số phòng hội thoại nội bộ | varchar(50) | notNull |  |
| Name | Tên phòng | varchar(50) | notNull |  |
| IdOperator | Id nhân viên tạo phòng | varchar(30) | notNull |  |

Bảng ‑: Quan hệ CONFERENCES

#### Quan hệ CONFERENCEDETAIL

Ý nghĩa: Lưu trữ thông tin các tài xế trong một phòng hội thoại

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| Number | Số phòng hội thoại | varchar(30) | notNull |  |
| IdDriver | Mã tài xế thuộc phòng | varchar(30) | notNull |  |
| Date | Ngày tạo phòng | date | notNull |  |
| Time | Thời gian được mời vào phòng | time | notNull |  |
| State | Trạng thái của tài xế đối với phòng | int | notNull |  |

Bảng ‑: Quan hệ CONFERENCEDETAIL

#### Quan hệ UPDATETIME

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Giá trị** | **Ghi chú** |
| **Username** | Tài khoản đăng nhập | varchar(30) | notNull |  |
| RequestTime | Thời gian yêu cầu | int | notNull |  |

Ý nghĩa: Lưu trữ giá trị thời gian yêu cầu tài xế phải cập nhật về trung tâm điều khiển.

Bảng ‑: Quan hệ UPDATETIME

## Xây dựng Web Service

### Web Service hỗ trợ truy vấn dữ liệu

Việc hiển thị dữ liệu của hệ thống lên Google Map đòi hỏi việc phải truy vấn dữ liệu được lưu trữ trong cơ sở dữ liệu MySQL. Dữ liệu cần phải truy vấn ở đây chủ yếu là thông tin về xe taxi trong hệ thống theo từng yêu cầu cụ thể của nhân viên điều phối, thông tin khách hàng phục vụ cho việc thống kê danh sách khách hàng của hệ thống, thông tin về lịch sử hoạt động của taxi trong hệ thống.

Để thực hiện việc truy vấn dữ liệu hệ thống xây dựng RESTful Web Service để thực hiện việc truy vấn dữ liệu từ MySQL theo yêu cầu của từng chức năng trong hệ thống. Dữ liệu truy vấn sẽ được chuyển thành định dạng Json với sự hỗ trợ của thư viện Gson và được phản hồi lại theo từng yêu cầu.

|  |  |  |
| --- | --- | --- |
| Phương thức | URI | Chức năng |
| GET | /resources/query/getAllCustomerInDate | Lấy thông tin tất cả khách hàng trong ngày |
| GET | /resources/query/getAllCustomerInMonth | Lấy thông tin tất cả khách hàng trong tháng |
| GET | /resources/query/ | Lấy thông tin toàn bộ taxi trong hệ thống |
| GET | /resources/query/ getCustomerOfOperatorByDate | Khách hàng của một nhân viên trong ngày |
| GET | /resources/query/ getCustomerOfDriverByDate | Khách hàng của một tài xế trong ngày |
| GET | /resources/query/ getCustomerOfDriverByMonth | Khách hàng của một tài xế trong tháng |
| GET | /resources/query/ getCustomerOfOperatorByMonth | Khách hàng của một nhân viên trong tháng |
| GET | /resources/query/ getTaxi | Thông tin một xe cụ thể |
| GET | /resources/query/ getTimeLineTaxi | Thông tin lịch sử hoạt động của xe |
| GET | /resources/query/ joinConference | Giám sát thông tin tham gia phòng hội thoại |

Bảng ‑: Web Service hỗ trợ truy vấn dữ liệu

### Web Service lựa chọn xe phù hợp

Trong quá trình điều phối của nhân viên tại trung tâm điều khiển thì việc lựa chọn các xe taxi phù hợp và tối ưu nhất để phục vụ khách hàng chiếm một vị trí rất qua trọng. Yêu cầu đối với việc thực hiện lựa chọn taxi là cho phép nhân viên diều phối lựa chọn theo loại taxi, chọn số lượng taxi phụ hợp mà hệ thống có thể gợi ý lựa chọn, cho phép điều một taxi hoặc nhiều taxi cùng một lúc theo thứ tự ưu tiên về khoảng cách từ taxi đến vị trí khách hàng.

Nhân viên điều phối sẽ lựa chọn xe bằng cách kết hợp những tùy chọn sau: chọn loại xe, chọn số lượng xe gợi ý, chọn điều một xe hoặc nhiều xe cùng lúc. Tiếp theo nhân viên điều phối sẽ phải cung cấp thông tin về tọa độ khách hàng. Những thông tin này sẽ được Web Service tiếp nhận và xử lý. Quá trình xử lý được thực hiện theo các bước sau đây:

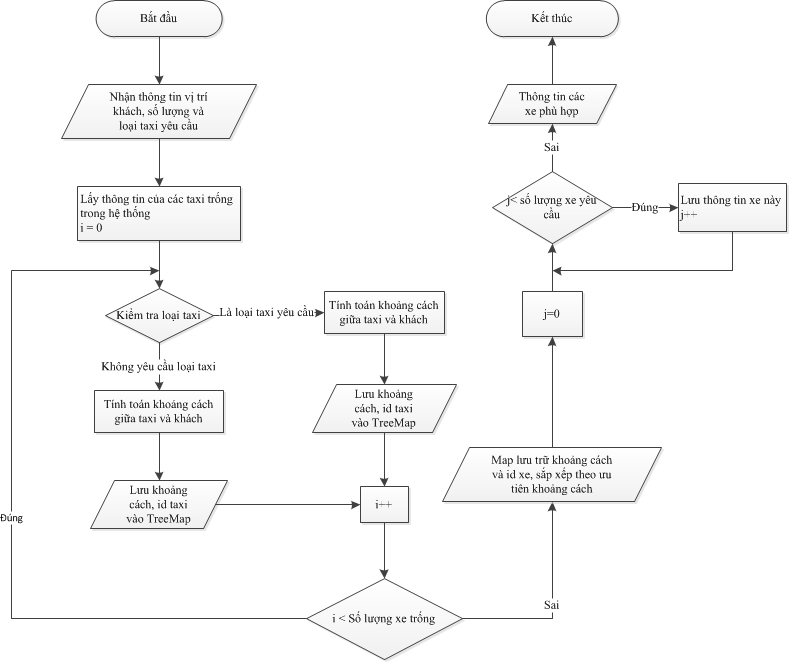
* Tiếp nhận thông tin từ nhân viên điều phối.
* Lấy thông tin về các xe trong trong hệ thống.
* Phân loại theo loại taxi yêu cầu, tính toán khoảng các từ taxi đến vị trí của khách hàng. Xắp xếp ưu tiên theo khoảng cách gần nhất.

Khoảng cách giữa hai tọa độ A(latA,lngA) và B(latB,lngB) với bán kính trái đất R=6371Km được tính theo công thức sau:

d=arccos(sin(latA)\*sin(latB)+cos(latA)\*cos(latB)\*cos(lngA-lngB))\*R

* Dựa theo số lượng taxi trong yêu cầu để lựa chọn số lượng cho phù hợp sau đó gửi thông tin về những taxi phù hợp cho nhân viên điều phối.

Quá trình xử lý được mô tả trong sơ đồ sau:

Chi tiết Web Service:

Hình ‑: Web Service lựa chọn xe phù hợp

* Phương thức: GET
* URI: /resources/query/getNearTaxi
* Danh sách tham số

|  |  |
| --- | --- |
| Tham số | Mô tả |
| lat | Latitute của khách hàng |
| tng | Longitude của khách hàng |
| no | Số lượng xe yêu cầu gợi ý |
| isMultiTaxi | Chọn nhiều xe hay không |
| taxiType | Loại xe yêu cầu gợi ý |

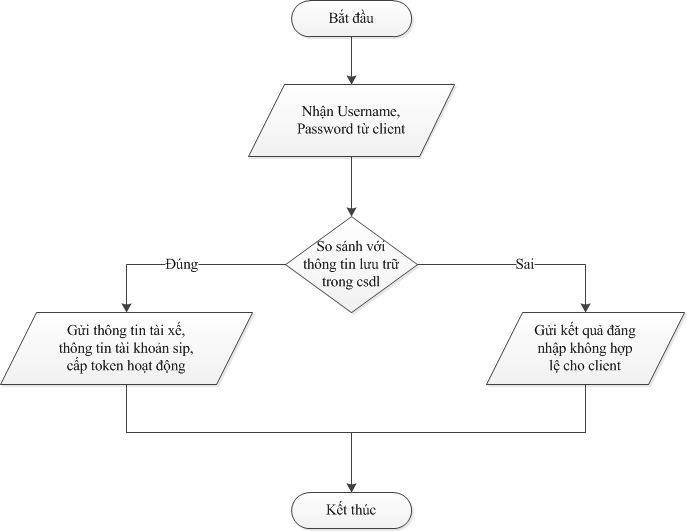
### Web Service tương tác với Client

#### Chức năng đăng nhập, đăng xuất

*Đăng nhập:*

Để nhân viên điều phối có thể quản lý và giám sát các taxi trong hệ thống thì bắt buộc các tài xế taxi phải truy cập vào hệ thống thống bằng cách cung cấp username và password cho ứng dụng client. Client sẽ gửi thông tin này lên server điều khiển để xác thực sự tồn tại của username này. Sau khi đã kiểm tra thông tin là hợp lệ hệ thống sẽ phát sinh ngẫu nhiên và duy nhất một token cho username vừa thực hiện đăng nhập sau đó gửi trả về cho client phía tài xế. Token này được tạo bằng các sử dụng lớp java.util.UUID để tạo ra một token với giá trị có độ dài 128 bit.

Khi nhận được thông tin token này, client sẽ phải sử dụng token này cho việc truy cập các tài nguyên và cập nhật thông tin về server điều khiển vì server sẽ chỉ đáp ứng những yêu cầu có kèm đúng token được cấp phát khi tài xế đăng nhập hệ thống. Ngoài ra, khi đăng nhập thành công hệ thống còn cung cấp thêm thông tin về tài xế, thông tin về tài khoản sip để tài xế sử dụng trong việc liên lạc thông qua tổng đài Trixbox tại trung tâm điều khiển.

Mô hình hoạt động của của Web Service xử lý đăng nhập được thể hiện trong hình sau:

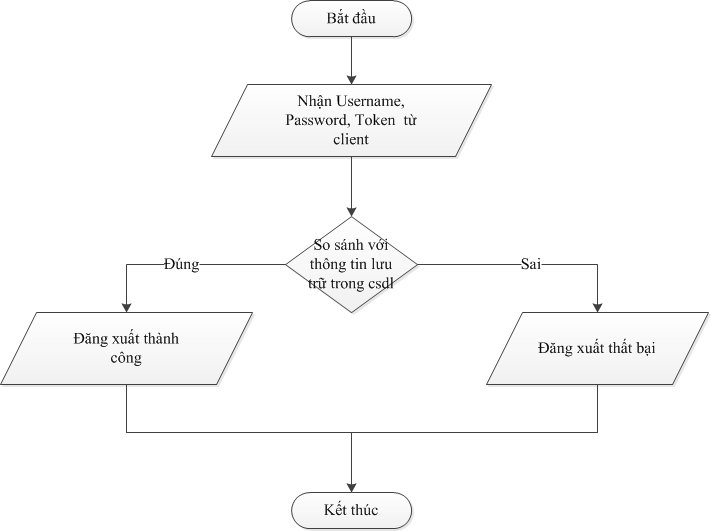
Hình ‑: Chức năng đăng nhập

Chi tiết Web Service:

* Phương thức: GET
* URI: /resources/login
* Danh sách tham số

|  |  |
| --- | --- |
| Tham số | Mô tả |
| username | Tài khoản đăng nhập hệ thống |
| password | Mật khẩu |

*Đăng xuất:*

Việc kết thúc hoạt động, tài xế sẽ phải thực hiện việc đăng xuất khỏi hệ thống để nhân viên điều phối có thể kiểm soát được những taxi nào đang hoạt động và đã ngưng hoạt động. Thông tin chứng thực được gửi lên server điều khiển bao gồm: username, password, và token đã được cấp phát lúc đăng nhập. Quá trình xử lý đăng xuất của tài xế được thực hiện thông qua Web Service được mô tả như hình sau đây.

Hình ‑: Chức năng đăng xuất

Chi tiết Web Service:

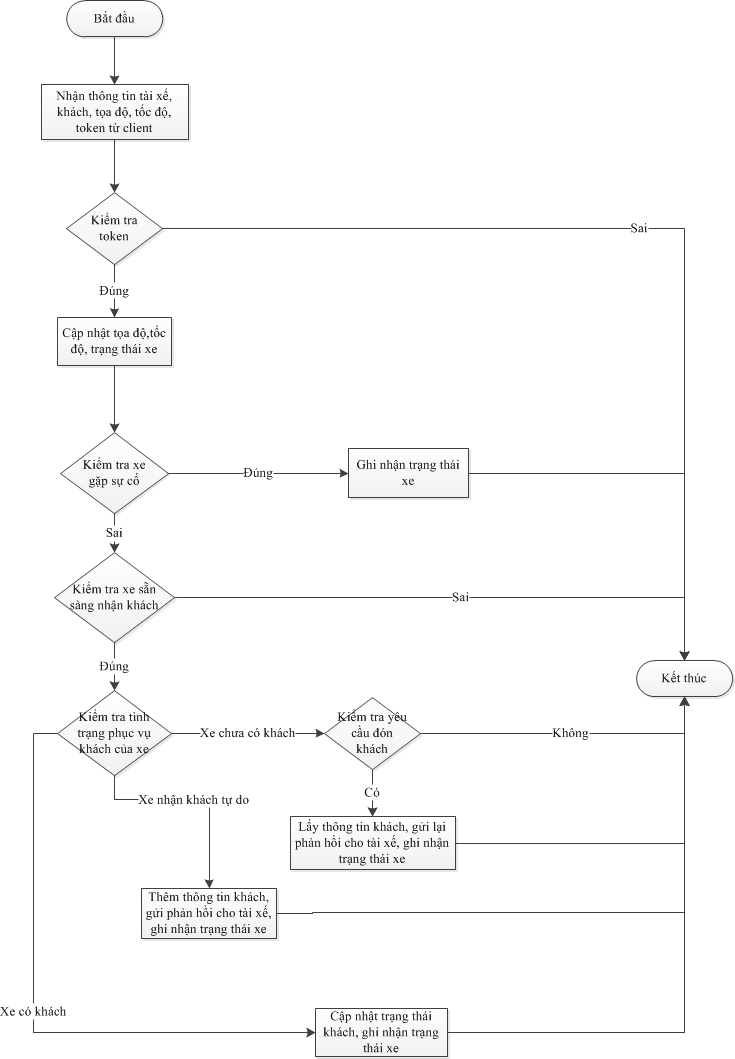
* Phương thức: GET
* URI: /resources/logout
* Danh sách tham số

|  |  |
| --- | --- |
| Tham số | Mô tả |
| username | Tài khoản đăng nhập hệ thống |
| password | Mật khẩu |
| token | Token được cấp phát khi đăng nhập |

#### Giao tiếp, cập nhật trạng thái và trao đổi dữ liệu với trung tâm điều khiển

Trong việc quản lý và giám sát taxi, việc liên lạc trao đổi thông tin giữa trung tâm điều hành và tài xế chiếm vai trọ rất quan trọng. Để đáp ứng yêu cầu này của hệ thống, chúng ta sẽ phải thiết kế một Web Service nhằm thực hiện việc giao tiếp, cập nhật trạng thái và trao đổi dữ liệu với trung tâm điều khiển. Client hoạt động trên mỗi taxi có nhiệm vụ cập nhật các thông tin về vị trí, tốc độ, trạng thái xe cũng như tình trạng của khách nếu xe đang phục vụ khách. Khi nhận được những thông tin cập nhật từ client, web service này sẽ tiến hành xử lý những thông tin này như sau:

* Ghi nhận trạng thái xe, cập nhật tọa độ vị trí, tốc độ của xe vào cơ sở dữ liệu.
* Tùy vào trạng thái của xe để có thể cung cấp thông tin phản hồi đến client cho phù hợp.
* Cung cấp thông tin khách hàng cho tài xế khi xe có yêu cầu đón khách từ nhân viên điều phối.
* Ghi nhận trạng thái và thông tin khách hàng nếu xe đang phục vụ khách.
* Cung cấp thông tin về phòng hội thoại khi tài xế được yêu cầu tham gia phòng từ nhân viên điều phối.
* Cung cấp thông tin về thời gian phải gửi cập nhật về trung tâm điều khiển.



Hình ‑: Giao tiếp, cập nhật trạng thái và trao đổi dữ liệu với trung tâm điều khiển

Chi tiết Web Service:

* Phương thức: GET
* URI: /resources/query/update
* Danh sách tham số

|  |  |
| --- | --- |
| Tham số | Mô tả |
| username | Tài khoản của tài xế cập nhật |
| lat | Latitute của xe |
| tng | Longitude của xe |
| speed | Tốc độ của xe |
| state | Trạng thái của xe |
| idCustomer | Mã khách hàng đang phục vụ |
| token | Token được cấp phát khi đăng nhập |

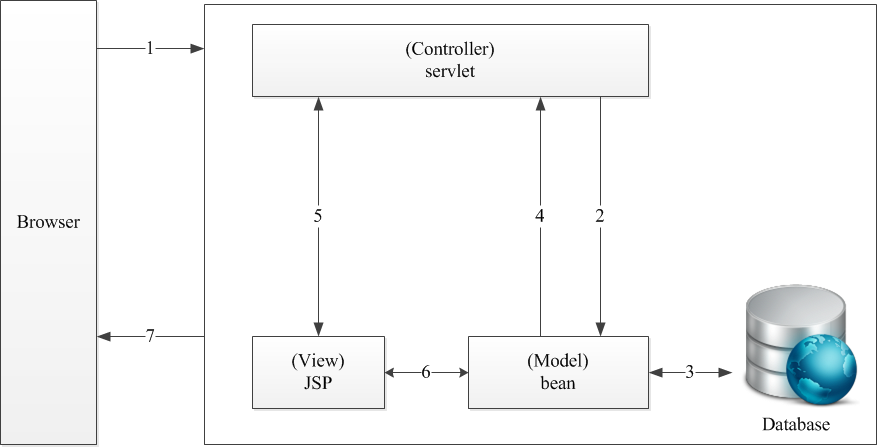
## Xây dựng ứng dụng web quản lý và điều hành taxi

Để giúp nhân viên sử dụng các tài nguyên trong hệ thống vào việc điều phối xe, chúng em xây dựng một ứng dụng web phục vụ việc quản lý, giám sát và điều phối xe taxi trong hệ thống.

### Tài nguyên hỗ trợ

* **IDE:** NetBeans IDE 7.0.1
* **SERVER:** Apache Tomcat 7.0.14.
* **JAVA EE VERSION**: Java EE 6 Web.
* **JAVA PLATFORM:** JDK 1.6 (Java Development Kit) đây là bộ thư viện, công cụ, tài liệu để phát triển ứng dụng viết bằng ngôn ngữ java được cung cấp bởi Oracle. Bao gồm các thành phần chính sau:
  + Development tool: bộ công cụ giúp phát triển, thực thi, debug và tạo java doc cho ứng dụng java
  + Runtime Enviroment: môi trường thực thi ứng dụng java bao gồm máy ảo java, các thư viện hỗ trợ.
  + Source code: bao gồm toàn bộ source code của JDK để ngừời sử dụng có thể nắm bắt đựợc các xử lý của bộ thư viện.
* **JDBC Driver for MySQL (Connector/J)**: MySQL cung cấp một thư viện chuẩn được gọi là driver để các framework như JDBC, ODBC, .NET tạo một ứng dụng có thể thao tác, xử lý trên hệ quản trị cơ sở dữ liệu này. Phiên bản được sử dụng là 5.1.20 download miễn phí tại: <http://www.mysql.com/downloads/connector/j/>.

### Cấu trúc ứng dụng

Ứng dụng hỗ trợ là một Java Web Application với cơ sở dữ liệu là MySQL đươc xây dựng theo mô hình MVC(Model – View – Controller). Hình sau mô tả cách thức hoạt động của các thành phần trong ứng dụng:

Hình ‑: Cách thức hoạt động của ứng dụng

Controller: package servlet chứa các Servlet xử lý và phản hồi lại các yêu cầu được thực hiện bởi người dùng.

Model: package bean bao gồm các lớp java thực hiện việc ánh xạ dữ liệu từ cơ sở dữ liệu MySQL sang các lớp đối tượng java

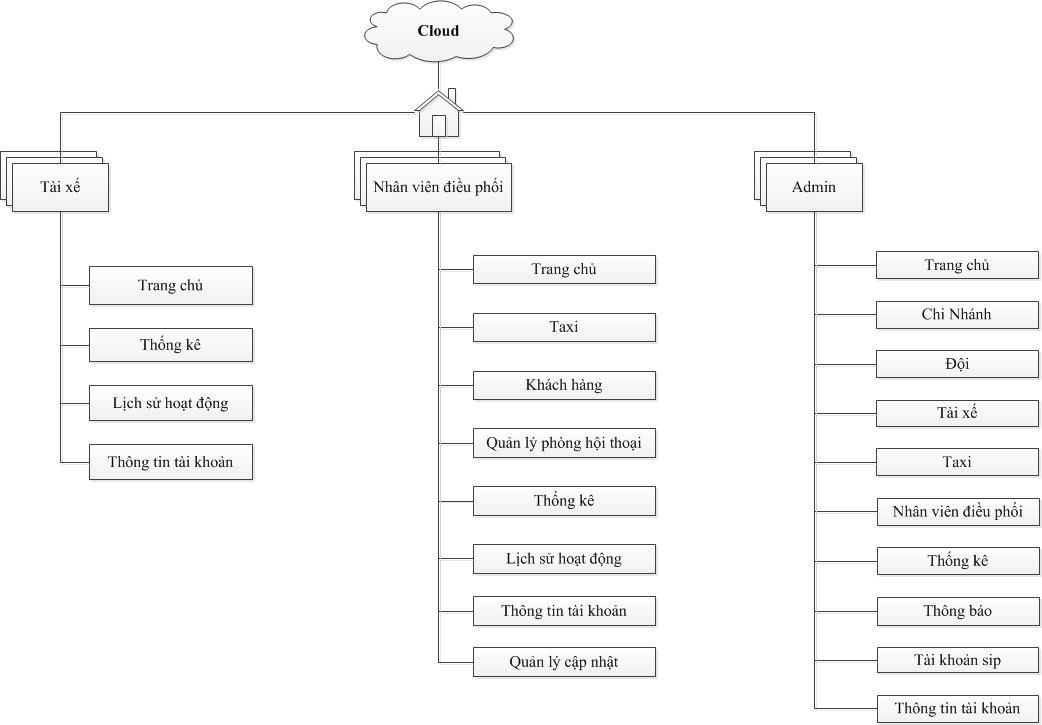
View: pakage chứa các trang JSP có nhiệm vụ trình bày dữ liệu ra màn hình.

Các bước hoạt động của ứng dụng:

* Nhân viên điều phối thực hiện gửi yêu cầu đến server thông qua trình duyệt web.
* Các Servlet (Controller): có nhiệm vụ nhận các yêu cầu, phân tích các giá trị tham số trong yêu cầu để tiến hành xử lý, lựa chọn các Model để thực hiện.
* Model thực hiện các yêu cầu kết nối, truy vấn dữ liệu từ cơ sở dữ liệu MySQL và trả về kết quả cho Controller
* Controller nhận kết quả trả về từ Model thực hiện tạo hoặc lựa chọn View để hiện thị kết quả nhận được.
* Controller gửi phản hồi lại trình duyệt web của nhân viên điều phối để hiện thị kết quả của yêu cầu mà nhân viên điều phối đã thực hiện.

### Xây dựng các chức năng cho các đối tượng sử dụng hệ thống

Ứng dụng web quản lý và điều hành taxi chủ yếu phục vụ ba đối tượng của hệ thống được mô tả như hình sau:



Hình ‑: Ứng dụng web quản lý và điều hành taxi

Mỗi đối tượng sẽ có những chức năng và quyền hạn được quy định cụ thể như sau:

* Tài xế: Có các thành phần cho phép: kiểm tra vị trí, trạng thái xe hiện tại, thống kê khách hàng đã phục vụ của tài xế, xem lại lịch sử hoạt động của taxi theo ngày, chỉnh sửa thông tin tài khoản của tài xế.
* Nhân viên điều phối: có các thành phần cho phép kiểm tra, giám sát vị trí, trạng thái của từng xe hoặc tất cả các xe trong hệ thống, thực hiện việc tiếp nhận, lựa chọn và điều phối xe thích hợp phục vụ khách hàng, thông kê khách hàng phục vụ, quản lý thời gian quy định mỗi tài xế phải thực hiện việc cập nhật thông tin về trung tâm điều khiển, quản lý các phòng hội thoại, việc yêu cầu tham gia phòng hội thoại đến các tài xế
* Quản trị viên: quản lý các đối tượng trong cơ sở dữ liệu, quản lý thông tin tài khoản, tài khoản sip.

### Xây dựng chức năng điều phối

Đây là thành phần chiếm vai trò rất quan trọng trong ứng dụng. Nó giúp cho hoạt động điều phối của nhân viên trở nên tiện lợi, nhanh chóng và chính xác. Nhân viên điều phối sẽ sử dụng các control trên form để cung cấp các thông tin cần thiết đến Web Service lựa chọn xe phù hợp để Web Service này thực hiện việc lựa chọn, sau đó nhận kết quả từ Web Service này để hiện thị cho nhân viên điều phối lựa chọn những xe phù hợp nhất.

Hình dưới đây là giao diện sử dụng của thành phần điều phối xe:



Hình ‑: Thành phần điều phối xe

Từ đó ta có các Control để thực hiện việc điều phối như sau:

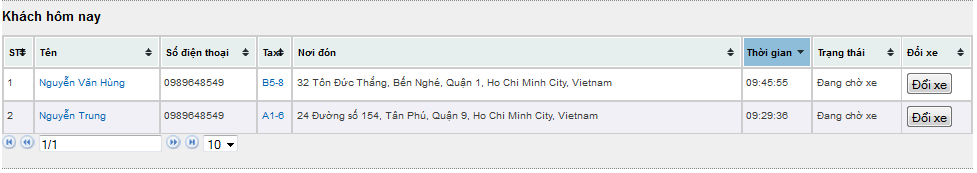
* Loại xe: lựa chọn loại xe điều phối, ở đây ta có thể chọn loại 4 chỗ, 7 chỗ hoặc chọn tất cả các xe.
* Số lượng xe để lựa chọn: đưa ra yêu cầu về số lượng xe mà hệ thống sẽ đưa ra gợi ý để lựa chọn, ví dụ giá trị đây là 5 có nghĩa là hệ thống sẽ chọn ra 5 xe có vị trí gần nhất để cho nhân viên điều phối lựa chọn.
* Điều nhiều xe: tùy chọn cho phép điều một xe hoặc nhiều xe đến một vị trí khách hàng.
* Chọn địa chỉ, Quận: là control giúp cho nhân viên điều phối tìm kiếm địa chỉ, tọa độ của khách hàng trên bản đồ Google Map bằng các sử dụng đối tượng Geocoder của Goolge Map Api.
* Thông tin khách hàng: nhân viên điều phối sẽ nhập tên và số điện thoại của khách hàng. Những thông tin về nơi đón và vị trí khách hàng sẽ được hệ thống tự động điền vào. Những thông tin này sau đó sẽ được gửi đến tài xế được lựa chọn.
* Tài xế: đưa ra một danh sách các tài xế có vị trí gần nhất đối với khách hàng, nhân viên điều phối có thể lựa chọn một hoặc nhiều tài xế để phục vụ khách.

Sau khi đã lựa chọn được xe phù hợp, nhân viên sẽ nhấn chọn nút “Thêm” để thực hiện việc điều phối xe cho khách. Trạng thái khách sẽ được hiện thị ngay trên bản đồ điều phối để giúp nhân viên có thể theo dõi việc phục vụ khách của các xe taxi

Hình ‑: Hiển thị khách trên bản đồ

Các biểu tượng tương ứng với trạng thái của khách hàng được mô tả như hình dưới đây:

Hình ‑: Trạng thái khách hàng

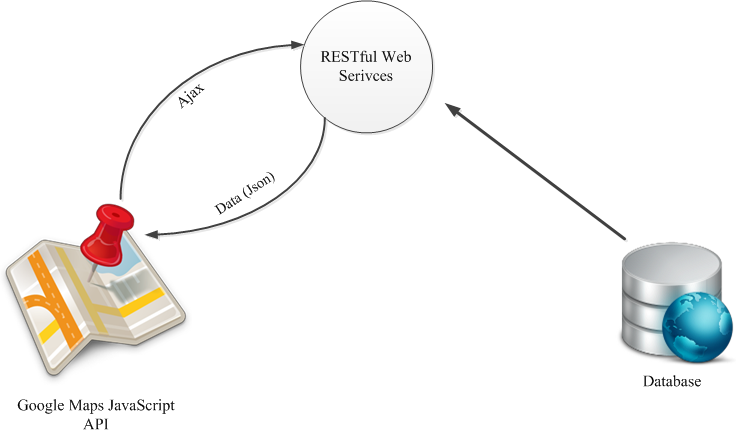
Sẽ có một danh sách khách hàng trong ngày được hiển thị như sau:

Hình ‑: Danh sách khách hàng

Trong trường hợp tài xế sau khi nhận khách không thể thực hiện được việc đón khách sẽ có một bảng thông báo hiện thị trên bản đồ để cho nhân viên điều phối thực hiện việc chuyển đổi tài xế phục vụ cho khách đó.

## Xây dựng hệ thống bản đồ

### Phương pháp thực hiện

Trong hệ thống này,để giúp người điều phối có được một cái nhìn tổng quan và dễ dàng trong việc điều phối các xe thì Server phải có nhiệm vụ hiển thị vị trí và trạng thái các xe đang hoạt động trên hệ thống lên bản đồ số. Vị trí và trạng thái của xe được cập nhật tương ứng với hoạt động của xe theo thời gian thực. Cơ chế hoạt động của thành phần hiển thị bản đồ được mô tả như hình sau:

Hình ‑: Mô hình xử lý hiển thị bản đồ

Chức năng các thành phần:

* Google Maps JavaScript API: hiển thị dữ liệu là vị trí và trạng thái của các xe trong hệ thống lên bản đồ. Dữ liệu lấy về theo phương thức Ajax (Asynchronous JavaScript and XML) với định dạng dữ liệu là Json.
* RESTful Web Service: có nhiệm vụ truy vấn dữ liệu từ cơ sở dữ liệu, chuyển dữ liệu về định dạng Json và gửi dữ liệu về theo yêu cầu.
* Cơ sở dữ liệu: lưu trữ vị trí, trạng thái và các thông tin của các xe trong hệ thống.

Theo như hình mô tả ta thấy dữ liệu được hiển thị lên bản đồ sẽ được lấy về theo phương thức Ajax được thực thi với sự hỗ trợ của thư viện Jquery thông qua RESTful Web Service với định dạng Json. Trong một khoảng thời gian nhất định, yêu cầu lấy dữ liệu sẽ được gửi đến RESTful Web Service. Dữ liệu trả về sẽ được xử lý và hiển thị lên bản đồ.

### Triển khai các bản đồ trong hệ thống

#### Hiển thị bản đồ

Google Maps JavaScript API là bản đồ số được nhúng trực tiếp vào trang web cụ thể, để được quyền sử dụng bản đồ này chúng ta phải đăng ký, đồng ý với các quy định của Google để được cấp một key. Chúng ta có sử dụng đã đăng ký này cho tất cả các Google Maps API. Để tạo key chúng ta sẽ thực hiện theo những bước sau:

* Truy cập và đăng nhập bằng tài khoản của google vào APIs Console tại https://code.google.com/apis/console.
* Trong phần Services sẽ liệt kê tất cả các API được cung cấp bởi Google, chúng ta sẽ kích hoạt Google Maps API v3 tại đây.
* Sau khi kích hoạt, trong phần API Access ta sẽ thấy API key đã được tạo ra
* Sau khi có được key. Việc hiển thị bản đồ trên web rất đơn giản như trong ví dụ sau:
* Chèn MAPs API JavaScript bằng các sử dụng thẻ script như ví dụ sau

<script type="text/javascrip> src="https://maps.googleapis.com/maps/api/js?key=YOUR\_API\_KEY&sensor=SET\_TO\_TRUE\_OR\_FALSE">

</script>

Với YOUR\_API\_KEY chính là key chúng ta đã đăng ký ở phần trên.

* Tạo một thẻ div với tên là “map\_canvas”, bản đồ sẽ được hiển thị trên web bên trong thẻ div này.

<div id="map\_canvas" style="width:100%; height:100%"></div>

Chiều dài, rộng của bản đồ sẽ phụ thuộc vào các thuộc tính của div này

* Chúng ta sẽ tạo một JavaScript Object chứa các thuộc tính của bản đồ

var mapOptions = {

center: new google.maps.LatLng(-34.397, 150.644),

zoom: 8,

mapTypeId: google.maps.MapTypeId.ROADMAP

};

* Khởi tạo đối tượng map sẽ được hiển thị trong div “map\_canvas” với các Option được khỏi tạo trong đối tượng mapOptions

var map = new google.maps.Map(document.getElementById("map\_canvas"),

mapOptions);

Như vậy, về cơ bản chúng ta đã hiển thị được bản đồ số của Google lên trên web. Tiếp theo chúng ta sẽ tìm hiểu về phương thức hiển thị xe, vị trí và trạng thái các xe lên trên bản đồ.

#### Hiển thị xe

Việc tương tác với bản đồ, yêu cầu, xử lý, hiển thị dữ liệu sẽ được thực hiện hoàn toàn bằng JavaScript. Để hiển thị vị trí và trạng thái các xe chúng ta sẽ sử dụng Maker(google.maps.Marker) của API. Đối tượng Maker này có chức năng xác định một vị trí trên bản đồ. Vị trí của xe cũng chính là vị trí của Maker trên bản đồ. Trạng thái của xe hiển thị bằng cách thay đổi icon của Maker, được quy định theo bảng sau đây:

|  |  |
| --- | --- |
| Icon | Trạng thái |
| C:\Users\Nguyen Trung\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\camaro_256.png | Xe có thể phục vụ khách |
| C:\Users\Nguyen Trung\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\Renault laguna1.png | Xe đang trên đường đón khách |
| C:\Users\Nguyen Trung\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\red_256.png | Xe đang phục vụ khách |
| C:\Users\Nguyen Trung\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\aston_martin.png | Xe vượt tốc độ |
| C:\Users\Nguyen Trung\Desktop\IConcar\Car.png | Xe gặp sự cố |
| C:\Users\Nguyen Trung\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\Actions-im-kick-user-icon.png | Xe mà tài xế chưa đăng nhập vào hệ thống |

Bảng ‑: Trạng thái xe trong hệ thống

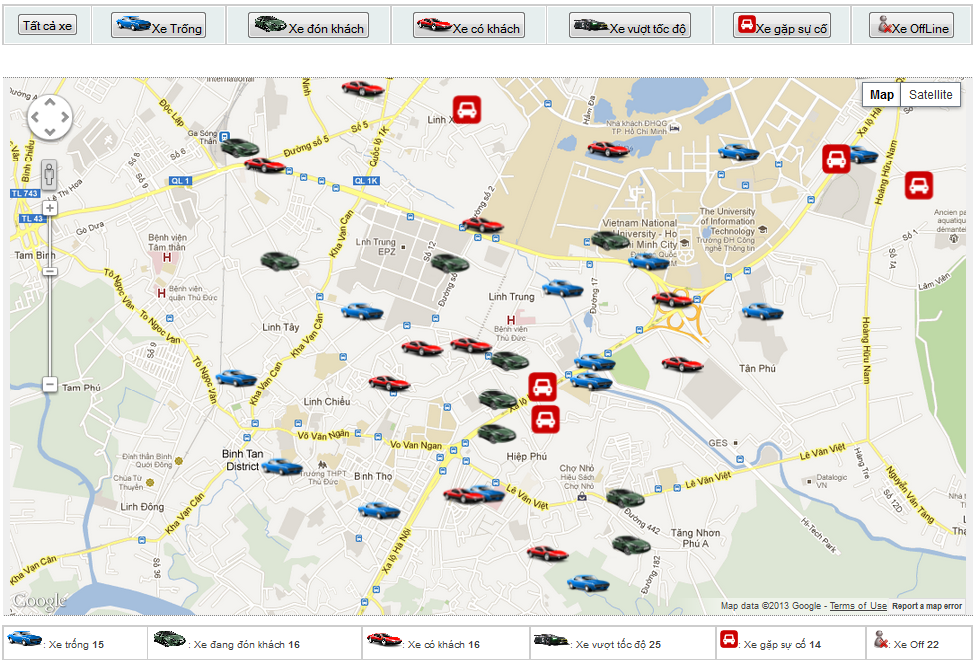
Vị trí của xe sẽ được hiển thị trực tiếp lên bản đồ như hình sau:



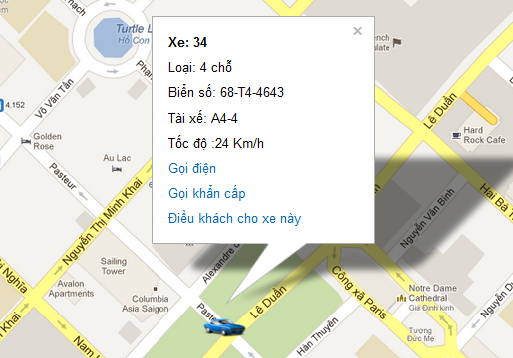
Hình ‑: Hiển thị vị trí xe

Để có thể hiển thị tất cả các xe trong hệ thống lên bản đồ, chúng ta sẽ bằng phương thức Ajax gửi yêu cầu đến các Web Service, Web Service sẽ xử lý yêu cầu, truy suất lấy dữ liệu từ MySQL Server, chuyển dữ liệu về định dạng Json và gửi dữ liệu về lại trình duyệt yêu cầu. Dữ liệu được trả về là một mảng các đối tượng lưu trữ thông tin về vị trí, trạng thái cũng như các thông tin cơ bản của xe taxi.

Tương ứng với mảng các đối tượng xe taxi này chúng ta sẽ tạo một mảng các đối tượng Maker, khởi tạo đối tượng Maker này với thông vị trí và trạng thái được lấy từ mảng dữ liệu thông qua Web Service. Để hiển thị thông tin của mỗi xe, chúng ta sử dụng đối tượng InfoWindow (google.maps.InfoWindow() ) để giúp cung cấp thêm thông tin về xe cho người điều phối. Sau khi đã khởi tạo các đối tượng Maker, InfoWindow, các đối tượng này sẽ được hiển thị lên bản đồ Google Map. Bằng cách này, nhân viên điều phối có thể thấy được vị trí, trạng thái của tất cả các xe trong hệ thống thông qua bản đồ sau:



Hình ‑: Bản đồ trạng thái của hệ thống



Hình ‑: Hiển thị thông tin một xe

Để có thể hiển thị hoạt động các xe một cách liên tục hệ thống sử dụng phương thức setInterval(), phương thức này sẽ gọi thực hiện một phương thức cụ thể lặp lại trong một khoảng thời gian đã được định trước.Chúng ta xẽ sử dụng phương thức này, kết hợp với Ajax để lấy dữ liệu trong cơ sở dữ liệu về trình duyệt. Dữ liệu này sẽ được xử lý để cập nhật vị trí lại cho Maker tương ứng với vị trí do Client gửi lên cho Server.

Ngoài ra, dựa trên trạng thái của xe, hệ thống có khả năng co biết số lượng xe theo trạng thái trong hệ thống, phân loại để hiển thị các xe lên bản đồ theo trạng thái, và một số chức năng có sử dụng bản đồ trong hệ thống như sau:

* Số lượng xe theo trạng thái.



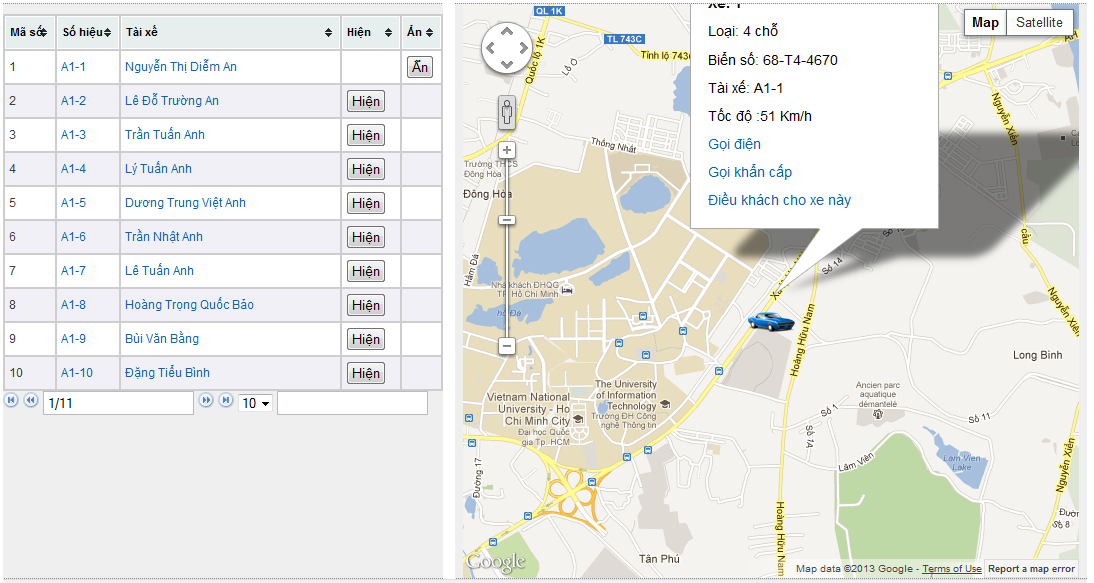
Hình ‑: Số lượng xe trong hệ thống

* Vị trí của các xe trống có khả năng phục vụ khách trong hệ thống.



Hình ‑: Hiển thị theo phân loại trạng thái xe

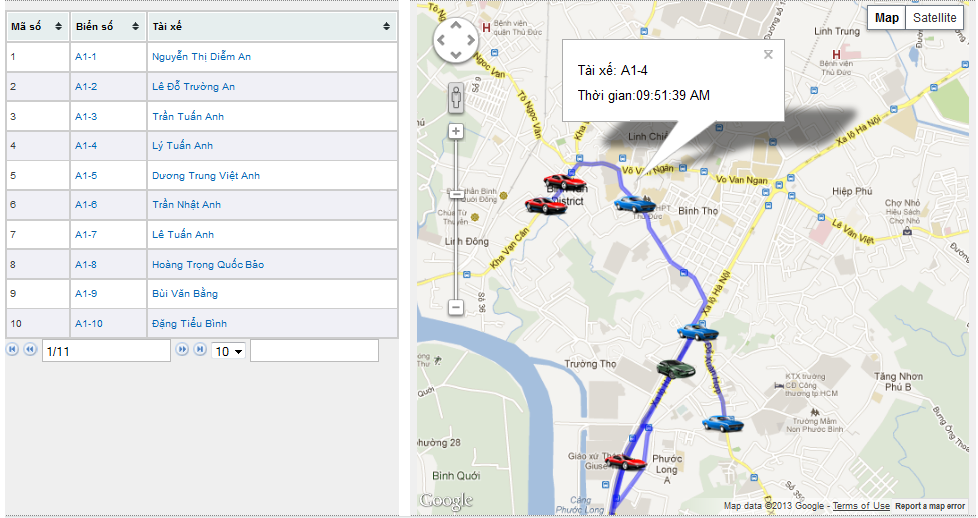
* Hiển thị vị trị và trạng thái của một xe cụ thể: danh sách tất cả các xe được hiện thị trong bảng, chúng ta có thể chọn một xe cụ thể để tiến hành giám sát.



Hình ‑: Giám sát hành trình của một xe

#### Hiển thị lịch sử hoạt động của xe

Trạng thái và vị trí của xe được client cập nhất lên Server sẽ được hệ thống lưu trự lại, thông tin lưu tữ bao gồm mã tài xế, tọa độ xe, ngày, giờ và trạng thái của xe. Dữ liệu được lưu tại MySQL Server và được tải bằng cách sử sụng Ajax kết hợp với Web Service. Trạng thái của xe được thể hiện bằng những Maker với mỗi icon khác nhau cho mỗi trạng thái khác nhau.



Hình ‑: Lịch sử hoạt động của một xe

## Xây dựng ứng dụng Client

### Các xử lý về thông tin GPS

Đa số các điện thoại sử dụng hệ điều hành android đều sử dụng chip xử lý có khả năng nhận tín hiệu GPS từ vệ tinh. Vì vậy android cung cấp một bộ thư viện dùng để giao tiếp và xử lý các thông tin lấy từ GPS nằm trong package android.location. Trong package này gồm có các thành phần như LocationManager, Location… Nhưng quan trọng nhất là lớp location, lớp này đại diện cho tập hợp các thông tin thu từ vệ tinh bao gồm: kinh độ, vỹ độ, các tùy chọn về cao độ, tốc độ, độ chính xác của vị trí….

Cách thức việc lấy thông tin GPS được thực hiện thông qua các nhà cung cấp. Có hai nhà cung cấp chính ở đây là GPS\_PROVIDER và NETWORK\_PROVIDER được đặt trưng bởi lớp provider. Lớp này được khởi tạo thông qua hàm getBestProvider(), Android sẽ tự động chọn nhà cung cấp có độ chính xác cao nhất tại thời điểm hiện tại cho chúng ta. Một số hàm cơ bản để lấy thông tin về Location cơ bản là:

Location mLocation = null;

mLocation.getAccuracy(); // Trả lại thông tin là độ chính xác của GPS.

mLocation.getAltitude(); // Trả lại thông tin là cao độ của vị trí hiện tại.

mLocation.getLatitude(); // Trả lại thông tin là Latitude;

mLocation.getLongitude(); // Trả lại thông tin là Longutude;

mLocation.getSpeed(); // Trả lại thông tin là tốc độ hiện tại;

Để có thể nhận biết sự thay đổi về vị trí khi người dùng di chuyển, chúng ta cần phải thực hiện implement một interface là LocationListener để có được những hàm lắng nghe sự thay đổi về Location. Interface này gồm có 4 hàm chính với chức năng khác nhau nhưng quan trọng nhất là hàm onLocationChanged(Location arg0) dùng để xử lý khi có sự thay đổi về tọa độ.

@Override

**public** **void** onLocationChanged(Location location) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** onProviderDisabled(String provider) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** onProviderEnabled(String provider) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** onStatusChanged(String provider, **int** status, Bundle extras) {

// **TODO** Auto-generated method stub

}

Những quyền cần thiết để có thể truy xuất thông tin về GPS tối thiểu cần có bao gồm:

* Android.permission.ACCESS\_FINE\_LOCATION: quyền này giúp chúng ta có thể sử dụng LocationManager.GPS\_PROVIDER để xác định vị trí thông qua GPS.
* Android.permission.ACCESS\_FINE\_LOCATION hay android.permission.ACCESS\_COARSE\_LOCATION: những quyền này giúp chúng ta có thể sử dụng LocationManager.NETWORK\_PROVIDER để xác định vị trí hiện tại.

### Các xử lý và triển khai Google Direction API

Chức năng tìm đường là một trong các dịch vụ mà google cung cấp cho người sử dụng. Đối với việc phát triển ứng dụng thì google cung cấp Direct API để thực hiện điều này. Hệ thống tìm đường này cung cấp tìm đường cho ba loại phương tiện chính bao gồm: xe hơi, xe buýt và đi bộ. Tuy nhiên tùy thuộc vào quốc gia mà dịch vụ này có được cung cấp đầy đủ và chính xác không. Ở Việt Nam thì chủ yếu sử dụng dịch vụ tìm đường cho xe hơi nhưng độ chính xác vẫn chưa được cao. Direct API đã được giới thiệu cụ thể ở chương trước, vì vậy ở phần này chúng ta chủ yếu tập trung vào các bước thực hiện. Như đã biết, để có thể sử dụng được dịch vụ này chúng ta cần gửi các yêu cầu lên server của Google. Cấu trúc URL như sau:

<http://maps.google.com/maps/api/directions/json?origin=lat1,long1&destination=lat2,long2&language=vi&sensor=true&mode=driving>

Các tham số cần chú ý là:

* Origin= latitude và longitude của điểm bắt đầu.
* Destination= latitude và longitude của điểm kết thúc.
* Language= ngôn ngữ sử dụng cho các chỉ dẫn, google chỉ hỗ trợ một số ngôn ngữ trong đó có tiếng việt.
* Sensor = giá trị này xác định có sử dụng hay không các cảm biến trong việc xác định vị trí và tìm đường.
* Mode= giá trị này chỉ ra chế độ tìm đường đang được sử dụng.

Đối với phiên bản hiện tại mặc định giá trị trả về từ các dịch vụ của Google là tập tim theo dịnh dạng Json. Phân tích thông tin từ tập tin này chúng ta sẽ có được các chỉ dẫn về tìm đường và danh sách các điểm để có thể vẽ đường trên bản đồ. Các thành phần chính từ tập tin trả về bao gồm nhiều thẻ khác nhau, mỗi thẻ chứa một thông tin riêng.

* Thẻ “routes”: là thẻ chính lớn nhất của tập tin Json. Nó chứa toàn bộ thông tin mà chúng ta cần có.
* Thẻ “bounds”: là thẻ chứa tọa độ của hai điểm mút thuộc hướng đông bắc và tây nam.
* Thẻ “copyrights”: là thẻ chứa thông tin về bản quyền thông tin này.
* Thẻ “legs”: là thẻ chứa thông tin về việc tìm đường. Thẻ “legs” bao gồm nhiều thẻ con bên trong:
  + Thẻ “distance”: chứa thông tin về khoảng cách giữa hai điểm.
  + Thẻ “duration”: chứa thông tin về thời gian duy chuyển giữa hai điểm.
  + Thẻ “end\_address”, “end\_location”,”start\_address”,”start\_location”: chứa thông tin về điểm bắt đầu và điểm kết thúc.
  + Thẻ “steps”: chứa thông tin về những đoạn đường nhỏ khác nhau, mỗi đoạn này chứa các thông tin như: khoảng cách, thời gian, điểm bắt đầu, điểm kết thúc, chỉ dẫn, và danh sách các tọa độ dùng để vẽ đoạn đường này trên bản đồ.
  + Thẻ “overview\_polyline”: chứa thông tin về tập hợp tất cả tọa độ của quãng đường, nó đươc chuyển thành mã ascii. Vì vậy muốn sử dụng chúng ta phải giải mã để thu được danh sách các điểm này.

Xử lý các thông tin này chúng ta chúng ta có thể vẽ đoạn đường này trên bản đồ và hiển thị những thông tin cần thiết.

### Các xử lý cơ bản trên Google Map API v2

#### Những cải tiến mới trong Google Map API v2 so với Google Map API v1

Bản đồ được gói trong class MapFragment, và kế thừa từ class Fragment của Android. Với MapFragment chúng ta có thể thêm bản đồ vào một phần của một Activity lớn. Vì vậy chúng ta không những có thể hiển thị bản đồ trong các thiết bị có màn hình nhỏ như thiết bị di động mà còn có thể hiển thị nó trong một giao diện đồ họa phức tạp của các loại thiết bị di động có màn hình lớn như máy tính bảng.

Map API được xây dựng trên nền tảng Vector. Vì vậy khối lượng dữ liệu trả về sẽ ít hơn, việc hiển thị bản đồ trong ứng dụng của chúng ta sẽ nhanh hơn và ít tốn dung lượng hơn.

Bộ đệm dữ liệu được cải thiện, vì vậy người dùng sẽ không còn thấy các khoảng trống khi hiển thị bản đồ.

Bản đồ có khả năng thể hiện 3D. Bằng cách di chuyển điểm nhìn, người sử dụng có thể thấy bản đồ ở các góc nhìn khác nhau theo quy luật xa gần.

API còn bổ sung thêm các đối tượng mới nhằm hỗ trợ tối đa cho người phát triển ứng dụng như Polyline, Polygon, TileOverlay, GroundOverlay…

#### Đánh dấu vị trí trên bản đồ bằng Marker

Marker đại diện cho một vị trí trên bản đồ. Chúng ta có thể thay đổi các tính chất cơ bản của Marker như màu sắc, vị trí, hình đại diện… Marker được hướng dựa vào màn hình của thiết bị hơn là bề mặt bản đồ vì vậy khi ta thực hiện các hành động như xoay, phóng to thu nhỏ, nghiêng thì sẽ không là thay đổi sự định hướng của Marker. Chúng ta có thể tương tác với Marker thông qua một số hành động được cung cấp bởi Google Map API bao gồm click và kéo thả.

Để thêm một marker vào trong bản đồ chúng ta cần phải sử dụng hàm addMarker() của đối tượng GoogleMap với tham số là MarkerOption.

private GoogleMap mMap;  
mMap = ((MapFragment) getFragmentManager().findFragmentById(R.id.map)).getMap();  
mMap.addMarker(new MarkerOptions()  
       .position(new LatLng(0, 0))  
        .title("Hello world"));

Marker Option là class dùng để chứa toàn bộ thông tin về các thuộc tính của Marker, các thuộc tính này bao gồm:

* Position: là thuộc tính cần phải thiết lập đối với bất kỳ marker nào, dùng để xác định vị trí gắn marker.
* Title: chứa thông tin sẽ hiển thị trong tiêu đề của InfoWindow.
* Snippet: chứa thông tin sẽ hiển thị trong nội dung của InfoWindow.
* Draggable: chứa thuộc tính có hay không cho người dùng kéo thả Marker.
* Visible: chứa thuộc tính có hay không hiển thị marker.
* Anchor: chứa điểm trên ảnh được dùng để gắn vào bản đồ, mặc định là giữa ảnh.
* Incon: hình ảnh sẽ được hiển thị tại vị trí đặt marker, chúng ta không thể thay đổi thuộc tính này đối với các marker đã được gắn vào bản đồ.

Tính năng nổi bật trong Google Map API v2 là cung cấp việc hiển thị thông tin của Marker bằng InfoWindow. InfoWindow sẽ được hiển thị thông qua việc nhấn nhẹ vào Marker. Trên bản đồ cùng lúc chỉ có thể hiển thị được một window duy nhất, window hiện tại sẽ ẩn đi nếu một window khác được hiển thị. Chúng ta cũng có thể hiện thị cửa sổ này thông qua hàm showInfoWindow() trên Marker và ngược lại sử dụng hàm hideInfoWindow() để ẩn cửa sổ.

Chúng ta cũng có thể thay đổi nội dung và thiết kế của InfoWindow bằng cách tạo ra một lớp kế thừa của InfoWindowAdapter() interface sau đó gọi hàm GoogleMap.setInfoWindowAdapter() để thiết lập InfoWindow mới tạo thành mặc định. Interface này có hai phương thức chính mà chúng ta phải kế thừa là getInfoWindow(Marker) và getInfoContents(Marker). API sẽ gọi hàm getInfoWindow(Marker) trước, nếu trả lại giá trị null nó sẽ gọi hàm getInforContents(Marker) tiếp theo. Nếu giá trị trả về vẫn là null thì giao diện cửa sổ mặc định sẽ được hiển thị.

Hàm getInfoWindow() cho phép chúng ta thay đổi giao diện của toàn bộ infoWindow. Hàm getInfoContents() cho phép chúng ta thay đổi nội dung của infoWindow còn khung và hình nền vẫn được giữ nguyên.

**class** CustomInfoWindow **implements** InfoWindowAdapter

{

@Override

**public** View getInfoContents(Marker arg0) {

// Các hàm xử lý

**return** **null**;

}

@Override

**public** View getInfoWindow(Marker arg0) {

// Các hàm xử lý

**return** **null**;

}

}

Map API cho phép chúng ta bắt các sự kiện click vào marker thông qua phương thức OnMarkerClickListener, để thiết lập phương thức này trên bản đồ ta gọi hàm GoogleMap.setOnMarkerClickListener(OnMarkerClickListener). Khi người dùng click vào một marker thì mặc định InfoWindow sẽ hiện ra và di chuyển camera bản đồ vào vị trí của marker vừa được click. Phương thức này trả về một giá trị Boolean dùng để xác định người dùng có sử dụng event này không. Ngoài ra, API còn cho phép bắt sự kiện click vào InfoWindow thông qua phương thức OnInfoWindowClickListener. Để thiết lập phương thức này chúng ta gọi hàm GoogleMap.setOnInfoWindowClick(OnInfoWindowClick), khi chúng ta click vào InfoWindow thì hàm onInfoWindowClick(Marker) sẽ được gọi, và info window được click sẽ hiện sáng hơn so với bình thường.

#### Vẽ đường thẳng trên bản đồ bằng Polyline

Polyline là một đối tượng mới trong Google Map API v2 được sử dụng để vẽ hình trên bản đồ bằng các đường thẳng nối với nhau. Chúng ta có thể thay đổi các thuộc tính của Polyline như màu sắc, chiều rộng của nét vẽ, thuộc tính ẩn. Đối tượng Polyline bao gồm danh sách các điểm trên bản đồ, sau đó các điểm này sẽ được nối với nhau bằng những đường thẳng.

Để tạo ra một đối tượng Polyline trước tiên chúng ta cần khởi tạo một đối tượng PolylineOptions dùng để chứa các thuộc tính của Polyline và thêm danh sách các điểm vào nó, những điểm này được xác định trên bản đồ vì vậy chúng ta khởi tạo chúng như những đối tượng LatLng. Đường thẳng nối các điểm thực hiện theo thứ tự của cách điểm trong danh sách lưu trong đối tượng PolylineOptions. Để vào PolylineOption chúng ta sử dụng phương thức PolylineOption.add().

// Khởi tạo đối tượng PolylineOption và thêm danh sách các điểm vào đối tượng vừa khởi tạo.  
PolylineOptions rectOptions = new PolylineOptions()  
        .add(new LatLng(37.35, -122.0))  
        .add(new LatLng(37.45, -122.0))     
        .add(new LatLng(37.45, -122.2))     
        .add(new LatLng(37.35, -122.2))     
        .add(new LatLng(37.35, -122.0));   
  
// thiết lập màu của đường thẳng nối các điểm  
rectOptions.color(Color.RED);  
  
// Thêm Polyline vào trong bản đồ  
Polyline polyline = myMap.addPolyline(rectOptions);

Các thuộc tính cơ bản của Polyline bao gồm:

* Color: dùng để thiết lập màu của đường nối các điểm, giá trị này được thiết lập thông qua phương thức PolylineOptions.color(), chúng ta cũng có thể thay đổi được thuộc tính này sau khi đã thêm Polyline vào bản đồ bằng phương thức Polyline.setcolor().
* Width: dùng để thiết lập bề ngang của đường nối các điểm tính bằng pixel(px) thông qua phương thức PolylineOptions.width().
* Visibility: là giá trị boolea dùng để xác định có hay không việc hiển thị Polyne bằng phương thức PolylineOptions.visible().

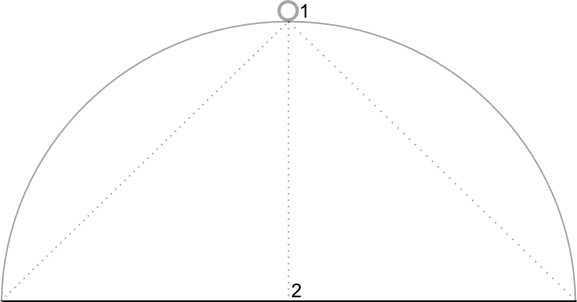
#### Thay đổi, di chuyển góc nhìn trong bản đồ

Google Map API v2 cung cấp thêm một đối tượng hoàn toàn mới là camera được sử dụng để thay đổi góc nhìn của người dùng trong bản đồ. Google Map v2 hỗ trợ hiển thị 3D vì vậy người dùng có thể xoay và nghiêng bản đồ bằng các thao tác đơn giản làm tăng khả năng tương tác của người dùng và bản đồ cũng như khả năng định hướng. Vì vậy đối tượng camera hết sức quan trọng trong Google Map API.

Việc thay đổi góc nhìn của người dùng thông qua đối tượng camera không làm thay đổi bất cứ marker, overlay hay các đối tượng đồ họa khác mà chúng ta đã thêm vào bản đồ trước đó.

CameraPositon là đối tượng dùng để tập hợp các thuộc tính dùng để thiết lập cho Camera, bao gồm các thuộc tính cơ bản như sau:

* Target(Location): dùng để thiết lập tọa độ để đặt góc nhìn, xác định thông qua latitude và longitude.
* Zoom: dùng để thiết lập độ phóng đại, độ phóng đại càng cao thì càng hiển thị rõ chi tiết của bản đồ.
* Tilt: dùng để thiết lập góc nhìn nghiêng trên bản đồ, điểm nhìn sẽ được đặt trên một cung tròn trên bản đồ, Khi sử dụng góc nhìn nghiêng thì bản đồ sẽ tự động xuất hiện la bàn trên góc trên bên trái để khôi phục lại góc nhìn thẳng như mặc định.



Hình ‑: Vị trí trước



Hình ‑: Vị trí sau

Muốn thay đổi vị trí của camera trước tiên chúng ta cần phải xác định chúng ta cần di chuyển camera đến đâu bằng cách sử dụng đối tượng CameraUpdate. Map API cho phép chúng ta tạo ra rất nhiều loại CameraUpdate khác nhau thông qua sử dụng CameraUpdateFactory. Các thuộc tính cơ bản bao gồm:

* Thay đổi độ phóng đại: chúng ta có thể sử dụng hàm CameraUpdateFactory.zoomIn() và CameraUpdateFactory.zoomOut() để thay đổi độ phóng đại của một đối tượng CameraUpdate theo từng cấp độ hay có thể sử dụng hàm CameraUpdateFactory.zoomTo(float) để thay đổi độ phóng đại đến một giá trị xác định.
* Thay đổi vị trí: Có hai phương thức phổ biến để thay đổi vị trí là:
  + CameraUpdateFactory.newLatLng(LatLng): thay đổi vị trí góc nhìn đến một vị trí xác định trên bản đồ bằng giá trị tọa độ Latitude và Longitude. Trong khi giữ nguyên các thuộc tính khác.
  + CameraUpdateFactory.newLatLngZoom(LatLng, float): Kết hợp thay đổi vị trí và độ phóng đại.
    - Phương thức đầy đủ nhất để thay đổi vị trí của camera là sử dụng phương thức:

CameraUpdateFactory.newCameraPosition(CameraPosition) trong đó CameraPosition được khởi tạo thông qua việc sử dụng phương thức new CameraPosition hay CameraPosition.Builder().

Sau khi đã có được đối tương CameraUpdate, Map API cung cấp cho chúng ta hai phương thức để di chuyển góc nhìn trên bản đồ đó là:

* GoogleMap.moveCamera(CameraUpdate): phương thức này thì góc nhìn sẽ được duy chuyển nhanh đến vị trí mà người dùng không cảm thấy được sự di chuyển.
* GoogleMap.animateCamera(cameraUpdate, callback, duration): phương thức này thì góc nhìn sẽ di chuyển mượt hơn, tạo cho người dùng cảm thấy góc nhìn đang di chuyển trên bản đồ đến vị trí cần thiết.Các tham số cần thiết ở đây bao gồm:
  + cameraUpdate: là đối tượng cameraUpdate để xác định vị trí mà camera sẽ di chuyển tới.
  + callback: là phương thức được gọi khi thực hiện việc di chuyển.
  + duration: là thời gian di chuyển từ vị trí ban đầu đến vị trí mới.

Việc Google Map API v2 cung cấp một phương thức giúp thay đổi góc nhìn người dùng giúp tăng khả năng tương tác của người dùng, tạo sự thoải mái và tiện dụng khi sử dụng.

### Truy vấn dữ liệu từ Web Service

Việc truy vấn dữ liệu từ server được sử dụng thông qua phương thức GET của giao thức HTTP, kết quả trả về là tập tin Json. Các truy vấn này giúp Server có thể biết được trạng thái cũng như vị trí hiện tại của taxi. Sau đây là một số truy vấn được sử dụng trong chương trình:

* Khi xe không có khách và ở trạng thái nhận khách từ Server thì truy vấn gửi lên Server sẽ là:

[http://localhost:8084/TaxiManagementSystem/resources/update?username={user}&lat={lat}&long={long}&speed={speed}&state=0&idCustomer=null&token={token}](http://localhost:8084/TaxiManagementSystem/resources/update?username=%7buser%7d&lat=%7blat%7d&long=%7blong%7d&speed=%7bspeed%7d&state=0&idCustomer=null&token=%7btoken%7d)

* Khi xe ở trạng thái đang đi đón khách sau khi nhận khách từ Server:

[http://localhost:8084/TaxiManagementSystem/resources/update?username={user}&lat={lat}&long={long}&speed={speed}&state=3&idCustomer={id}](http://localhost:8084/TaxiManagementSystem/resources/update?username=%7buser%7d&lat=%7blat%7d&long=%7blong%7d&speed=%7bspeed%7d&state=3&idCustomer=%7bid%7d)

* Khi xe ở trong trạng thái có khách thì truy vấn gửi lên server sẽ là:

[http://localhost:8084/TaxiManagementSystem/resources/update?username={user}&lat={lat}&long={long}&speed={speed}&state=1&idCustomer={id}](http://localhost:8084/TaxiManagementSystem/resources/update?username=%7buser%7d&lat=%7blat%7d&long=%7blong%7d&speed=%7bspeed%7d&state=1&idCustomer=%7bid%7d)

* Khi xe trả khách thì truy vấn gửi lên server sẽ là:

[http://localhost:8084/TaxiManagementSystem/resources/update?username={user}&lat={lat}&long={long}&speed={speed}&state=1&idCustomer={id}](http://localhost:8084/TaxiManagementSystem/resources/update?username=%7buser%7d&lat=%7blat%7d&long=%7blong%7d&speed=%7bspeed%7d&state=1&idCustomer=%7bid%7d)

* Khi xe đón khách dọc đường:

[http://localhost:8084/TaxiManagementSystem/resources/update?username={user}&lat={lat}&long={long}&speed={speed}&state=2&idCustomer=unknow&token={token}](http://localhost:8084/TaxiManagementSystem/resources/update?username=%7buser%7d&lat=%7blat%7d&long=%7blong%7d&speed=%7bspeed%7d&state=2&idCustomer=unknow&token=%7btoken%7d)

Khi có lỗi xảy ra trong quá trình truy vấn lên Server sẽ thông báo cho người sử dụng biết và sẽ thực hiện lại truy vấn vừa xảy ra lỗi.

### Các bước để thiết lập một phiên truyền thông bằng giao thức SIP

#### Thiết lập các thông số trong tập tin Manifest

Nếu muốn xây dựng chương trình sử dụng SIP API thì chúng ta cần phải đảm bảo điều kiện là thiết bị phải chạy trên Android phiên bản 2.3 trở lên.

Để có thể sử dụng SIP API chúng ta cần thêm vào hai sự cho phép trong tập tin manifest đó là: android.permission.USE\_SIP và android.permission.INTERNET. Để đảm bảo rằng ứng dụng của chúng ta chỉ có thể được cài đặt trên các thiết bị Android 2.3 trở lên thì cần thêm vào một thiết lập nữa là:

* <uses-sdk android:minSdkVersion="9" />
* <uses-deature android:name=”android.hardware.sip.voip”>Thiết lập này dùng để điều khiển làm sao chương trình có thể lọc được các thiết bị không hỗ trợ SIP API.
* Để có thể nhận được cuộc gọi từ một tài khoảng SIP khác thì chúng ta cần phải chỉ định một lớp con của BroadcastReceiver trong tập tin manifest:
* <receiver android:name=".IncomingCallReceiver" android:label="Call Receiver"/>

#### Thiết lập các thành phần cơ bản trong SIP API

1. Tạo ra một SipManager

Để sử dụng SIP API chúng ta bắt buộc phải tạo ra một đối tượng SipManager. Đối tượng này sẽ làm nhiệm vụ quản lý ứng dụng của chúng ta bao gồm:

* Thiết lập một phiên truyền thông SIP.
* Thiết lập và nhận cuộc gọi.
* Đăng ký và hủy đăng ký với nhà cung cấp dịch vụ SIP.
* Kiểm tra trạng thái phiên kết nối.

Chúng ta sử dụng phương thức sau để tạo ra một đối tượng SipManager:

public SipManager mSipManager = null;

if(mSipManager == null) {

mSipManager = SipManager.newInstance(this);

}

1. Đăng ký với một Sip Server

Một ứng dụng SIP thông thường bao gồm một hay nhiều người dùng, mỗi người dùng sử dụng một tài khoảng SIP. Mỗi tài khoảng SIP được đại diện bởi một đối tượng SipProfile. Tài khoảng được đăng ký trên thiết bị được gọi là local profile. Tài khoản Sip được kết nối tới gọi là peer profile. SipProfile được tạo ra thông qua phương thức SipProfile.Builder:

public SipProfile mSipProfile = null;

...

SipProfile.Builder builder = new SipProfile.Builder(username, domain);

builder.setPassword(password);

mSipProfile = builder.build();

Để có thể nhận cuộc gọi từ bên ngoài chúng ta phải thiết lập một hành động android.Sipdemo.INCOMING\_CALL, được sử dụng để tạo ra bộ lọc các intend khi thiết bị nhận cuộc gọi từ bên ngoài.

Intent intent = new Intent();

intent.setAction("android.SipDemo.INCOMING\_CALL");

PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 0, intent, Intent.FILL\_IN\_DATA);

mSipManager.open(mSipProfile, pendingIntent, null);

Cuối cùng thiết lập một SipRegistrationListener trên SipManager được dùng để xác định tài khoảng Sip có đăng ký thành công với nhà cung cấp dịch vụ sip không.

mSipManager.setRegistrationListener(mSipProfile.getUriString(), new SipRegistrationListener() {

public void onRegistering(String localProfileUri) {

updateStatus("Registering with SIP Server...");

}

public void onRegistrationDone(String localProfileUri, long expiryTime) {

updateStatus("Ready");

}

public void onRegistrationFailed(String localProfileUri, int errorCode,

String errorMessage) {

updateStatus("Registration failed. Please check settings.");

}

Sau khi chương trình sử dụng xong một SipProfile, chúng ta cần đóng nó để giải phóng bộ nhớ và ngắt đăng ký với Server.

public void closeLocalProfile() {

if (mSipManager == null) {

return;

}

try {

if (mSipProfile != null) {

mSipManager.close(mSipProfile.getUriString());

}

} catch (Exception ee) {

Log.d("WalkieTalkieActivity/onDestroy", "Failed to close local profile.", ee);

}

}

1. Thực hiện một cuộc gọi thoại

Để có thể thực hiện được một cuộc gọi thoại trước tiên chúng ta cấn các điểu kiện sau:

* SipProfile thực hiện cuộc gọi và địa chỉ Sip được gọi phải tốn tại và hoàn thành đăng ký với Server.
* Khởi tạo thành công đối tượng SipManager.

Bước tiếp theo chúng ta thiết lập phương thức SipAudioCall.Listener để có thể nhận biết được các sự kiện tương tác với nhau trong quá trình thiết lập cuộc gọi.

SipAudioCall.Listener listener = new SipAudioCall.Listener() {

@Override

public void onCallEstablished(SipAudioCall call) {

call.startAudio();

call.setSpeakerMode(true);

call.toggleMute();

}

@Override

public void onCallEnded(SipAudioCall call) {

}

};

Một khi đã thiết lập xong phương thức SipAudioCall.Listener, chúng ta có thể thiết lập cuộc gọi thông qua phương thức makeAudioCall của đối tượng SipManager. Các tham số bao gồm:

* Địa chỉ Sip dùng đễ gọi.
* Địa chỉ Sip được gọi.
* Một phương thức SipAudioCall.Listener để lắng nghe các sự kiện từ SipAudioCall.
* Thời gian timeout.

call = mSipManager.makeAudioCall(mSipProfile.getUriString(), sipAddress, listener, 30);

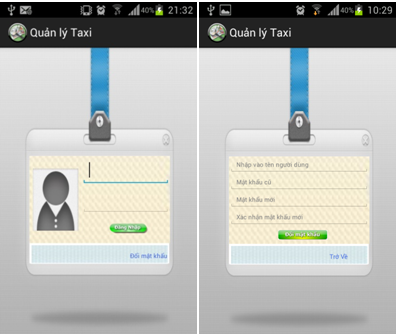
1. Nhận cuộc gọi

Để nhận cuộc gọi, chúng ta phải khai báo một subclass của BroadcastReceiver có khả năng trả lời một intend khi nhận ra nó là một cuộc gọi tới. Sau đây là những thông báo cần thiết đễ thực hiện điều này:

* Trong tập tin Manifest khai báo một thẻ <receiver>.
* Tạo là một class kế thừa từ BroadcastReceiver.
* Thiết lập một tài khoảng SIP kết nối đền intend để có thể nhận cuộc gọi từ bên ngoài.
* Thiết lập một bộ lọc intend để có thể nhận biết được cuộc gọi đến.

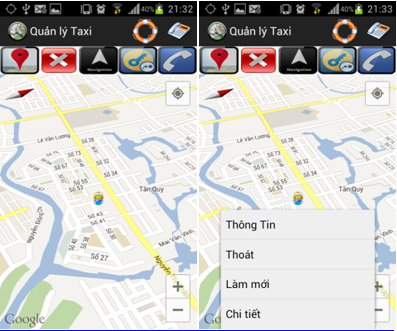
### Giao diện chương trình

Màn hình đăng nhập: được sử dụng để tài xế đăng nhập tài khoản, sau khi chứng thực thành công trên server thì tài xế mới có thể sử dụng được chương trình.



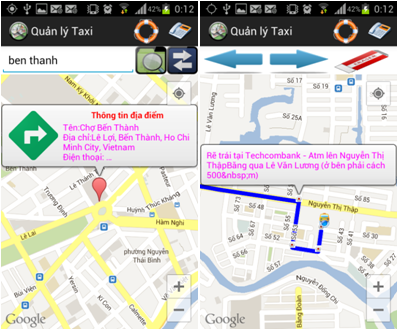
Hình ‑: Màn hình đăng nhập

Màn hình chính: là giao diện của Google Map v2, ngoài ra còn có những nút trên thanh điều hướng để thông báo trạng thái của taxi về trung tâm điều khiển, các tính năng của từng nút đã được trình bày trong hình 3-8.



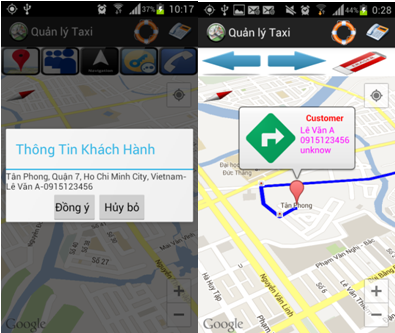
Hình ‑: Màn hình chính

Màn hình chỉ đường và tìm địa điểm: Sau khi tìm kiếm địa điểm bằng cách nhập vào tên địa điểm sau đó bấm nút tìm kiếm. Để có thông tin cụ thể về địa điểm ta tài xế nhấn vào điểm đánh dấu. Khi muốn tìm đường thì người dùng nhấn vào thông tin vừa hiển thị. Giao diện tìm đường sẽ hiện ra, người dùng sử dụng những nút tới lui để xem chỉ dẫn hoặc vào mục chi tiết trong menu.



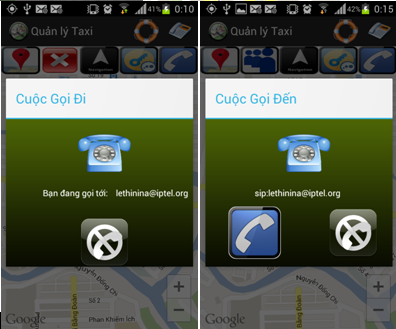
Hình ‑: Màn hình chỉ đường và tìm địa điểm

Màn hình nhận khách từ server: Khi có khách gửi từ server xuống thì có 2 lựa chọn là đồng ý và hủy bỏ. Nếu đồng ý thì chương trình sẽ tự động hiển thị đường đi tới vị trí của khách.



Hình ‑: Màn hình nhận khách từ server

Màn hình khi có cuộc gọi tới và đi: Chức năng này tương tự như thoại thông thường, màn hình giao diện bao gồm 2 nút chính dùng để bắt máy và tắt cuộc gọi.



Hình ‑: Màn hình khi có cuộc gọi tới và đi

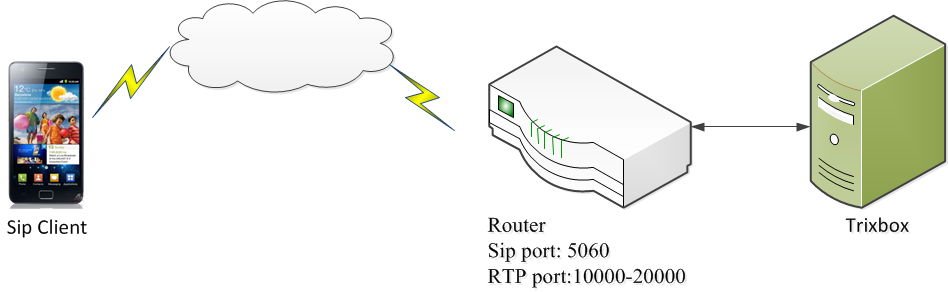
## Triển khai tổng đài Trixbox

### Triển khai tổng đài Tribox

Tổng đài phục vụ trong việc liên lạc giữa tài xế và nhân viên điều phối trong hệ thống được triển khai là tổng đài mã ngồn mở Trixbox CE 2.6 được cài đặt trên hệ thống máy ảo Vmware .Hệ thống này sẽ thay thế cho việc liên lạc bằng bộ đàm truyền thống đang được những nhà cung cấp dịch vụ taxi sử dụng. Giúp cho việc liên lạc trở nên thuận tiện và tiết kiệm chi phí đầu tư cơ sở hạ tầng.

Để tổng đài có thể phục vụ việc liên lạc thông qua mạng dữ liệu 3G chúng ta phải tiến hành public tổng đài này để cho các Sip Client có thể giao tiếp được với tổng đài thông qua mạng internet. Trong hệ thống này tổng đài sẽ được public bằng cách thực hiện phương thức NAT trên Router. Cụ thể chúng ta sẽ Nat các port tương ứng với các giao thức sau:

* Giao thức SIP: port 5060 sử dụng trong việc thiết lập, quản lý và kết thúc cuộc gọi.
* Giao thức RTP: port 10000-20000 sử dụng trong việc truyền dữ liệu trong cuộc gọi.



Hình ‑: Mô hình triển khai tổng đài trixbox

Hình vẽ trên mô tả hệ thống liên lạc thông qua tổng đài Trixbox. Cũng dựa trên tổng đài này hệ thống cung cấp cho nhân viên điều phối có khả năng tạo ra những phòng hội thoại giúp có thể liên lạc theo nhóm các tài xế và nhân viên điều phối.

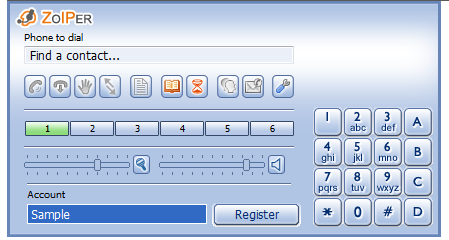
Mỗi nhân viên điều phối và tài xế taxi đều được cung cấp một tài khoản Sip quản lý bởi Trixbox. Thông tin tài khoản này sẽ được cung cấp cho tài xế khi đăng nhập, tài xế sẽ sử dụng chức năng gọi VOIP của client để thực hiện các cuộc gọi. Về phía nhân viên điều phối, sẽ thực hiện các cuộc gọi đến tài xế thông qua Zoiper Web được nhúng trong trình duyệt. Khi cần gọi tài xế nào chỉ cần nhấp chọn Maker của tài xế đó sau đó chọn chức năng “Gọi” hệ thống sẽ tự động cấu hình và thực hiện cuộc gọi đến tài xế. Ngoài lựa chọn trên nhân viên điều phối có thể sử dụng bất kỳ loại soft phone nào trên máy tính có hỗ trợ giao thức Sip để thực hiện việc liên lạc thông qua tổng đài này.

Để thực hiện việc tích hợp Softphone trên trình duyệt ta sẽ sử dụng Zoiper Web[[50]](#footnote-50). Zoiper Web được thiết kế cho phép nhúng trực trực tiếp vào các trang web và có khả năng tương thích với các trình duyệt phổ biến hiện nay. Zoiper Web cho phép người dùng sử dụng trình duyệt web để thực hiện và nhận các cuộc gọi VOIP sử dụng các giao thức SIP hoặc IAX2. API này cho phép lập trình viên kiểm soát Zoiper Web bằng cách sử dụng ngôn ngữ JavaScript.

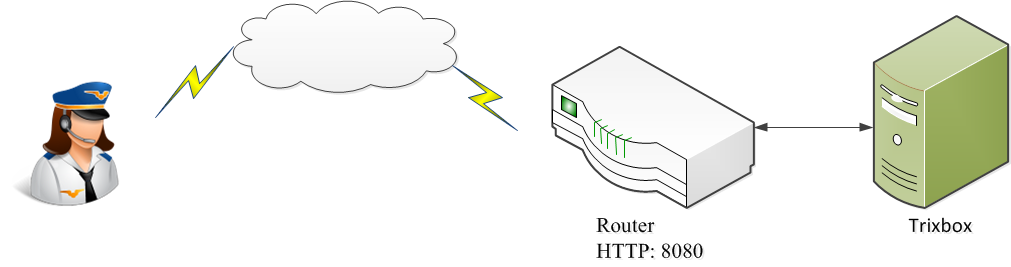
API này cung cấp các lớp với chức năng tạo, quản lý cấu hình, tài khoản, danh bạ và các cuộc gọi. Cung cấp các thông tin phản hồi từ các sự kiện của Zoiper Web đến trình duyệt. Tất cả các thuộc tính, phương thức và các tham số đều sử dụng dữ liệu là “string”.

API được xây dựng trên cơ sở lớp “Phone” cho phép quản lý các đối tượng “Config”, “Account”, “Call” and “Contact”.

Chúng ta sẽ sử dụng JavaScript khởi tạo các đối tượng này để thực hiện các cuộc gọi với Zoiper Web được nhúng vào trình duyệt.



Hình ‑: Zoiper Wep

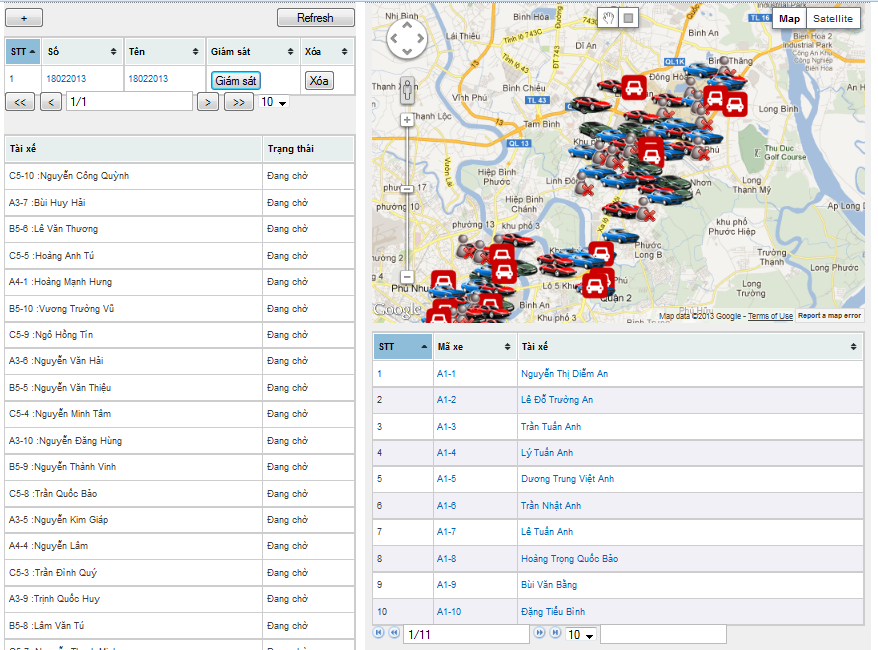
Tất cả các cuộc gọi giữa tài xế và nhân viên điều phối sẽ hoạt động trên nền mạng internet 3G.

Hình ‑: Quản lý phòng hội thoại

### Xây dựng thành phần quản lý phòng hội thoại

Việc sử dụng các phòng hội thoại giúp cho sự liên lạc giữa nhân viên điều phối và tài xế trở nên linh hoạt hơn, nhân viên điều phối có thể chọn liên lạc với các tài xế theo nhóm, theo từng khu vực cụ thể được hiển thị một cách trực quan trên bản đồ giúp hạn chế việc tài xế phải nhận thông báo không phù hợp với vị trí của mình như khi sử dụng bộ đàm. Để thực hiện chức năng cho phép nhân viên điều phối tạo các phòng hội thoại, chúng ta sẽ sử dụng việc khởi tạo và quản lý các conference trong tổng đài Trixbox thông qua giao thức HTTP.

Nhân viên điều phối sẽ sử dụng giao diện web của thành phần quản lý, cung cấp các thông số để khởi tạo conference. Ở đây nhân viên chỉ cần cung cấp số phòng và tên phòng, sau đó server Tomcat sẽ khởi tạo một yêu cầu HTTP trên cổng 8080 đến tổng đài Trixbox để yêu cầu khởi tạo một conference. Sau đó nhân viên điều phối sẽ thực hiện việc lựa chọn nhóm tài xế để gửi yêu cầu tham gia vào conference, nhân viên điều phối cũng có thể giám sát việc tham gia và rời khỏi phòng của tài xế.



Hình ‑: Giao diện quản lý phòng hội thoại

## Tổng kết

Chúng ta đã đi vào thiết kế chi tiết từng thành phần trong hệ thống đảm bảo hệ thống có thể đáp ứng đầy đủ những yêu cầu đã đề ra và có khả năng triển khai hoạt động trong thực tế.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Tổng kết đề tài

Khóa luận đã tổng kết và trình bày một cách hệ thống và cụ thể về kiến thức nền tảng, kiến thức về công nghệ để xây dựng hệ thống quản lý taxi dựa trên nền tảng Google Map Service và một số công nghệ khác. Đề tài đã xây dựng được một ứng dụng hoàn chỉnh trên cả client và server bao gồm các chức năng chính như sau:

Trung tâm điều hành với các chức năng:

* Web Application cung cấp các chức năng quản lý và điều hành hệ thống sử dụng thông qua trình duyệt web nên rất tiện dụng và nhanh chóng.
* Cung cấp cho nhân viên của hệ thống có cái nhìn tổng thể về hệ thống với việc hiển thị vị trí, trạng thái của các xe trong hệ thống, khách hàng lên bản đồ số, phân loại để hiển thị các xe theo từng trạng thái trên bản đồ số giúp tăng hiệu quả trong việc giám sát hoạt động của tài xế.
* Lưu trữ lịch sử hoạt động bao gồm trạng thái, vị trí và khách hàng của tài xế.
* Tiếp nhận, xử lý thông tin yêu cầu phục vụ của khách hàng, tiến hành lựa chọn các xe hợp lý theo mức độ ưu tiên để đón khách, cung cấp thông tin khách hàng cho tài xế xe được chọn giúp cho quá trình điều phối diễn ra nhanh chóng, giảm được tình trạng xe phải di chuyển không khách.
* Hệ thống RESTful Web Service cho phép giao tiếp với ứng dụng client mà tài xế sử dụng giúp client cập nhật trạng thái, vị trí, gửi thông báo đến tài xế khi có khách hàng, cập nhật dữ liệu lên bản đồ số.
* Tổng đài Trixbox được triển khai phục vụ việc liên lạc giữa nhân viên và tài xế trong hệ thống có thể thay thế cho việc liên lạc bằng bộ đàm truyền thống.
* Triển khai hệ thống liên lạc thông qua VOIP, tích hợp Soft Phone trên Web Application giúp cho việc liên lạc được dễ dàng, nhanh chóng.

Ứng dụng client chuyên dụng cho tài xế xe taxi với các chức năng:

* Hiện thị vị trí xe, chức năng tìm đường dựa trên Google Map.
* Cung cấp thông tin về vị trí, trạng thái xe, trạng thái khách hàng cho trung tâm điều khiển.
* Nhận điều phối và thông tin khách hàng từ trung tâm điều khiển, hiển thị vị trí khách và thực hiện việc tìm dường, chỉ dẫn đến vị trí khách.
* Chức năng gọi VOIP với tài khoản được cung cấp thông qua tổng đài Trixbox của trung tâm diều khiển trên nền 3G.
* Có khả năng quản lý vị trí, tốc độ, hoạt động của taxi.
* Thực hiện hoàn chỉnh quá trình nhận và đón trả khách.

Tóm lại, đề tài “XÂY DỰNG HỆ THỐNG MGTS-TAXI” đã hoàn thành các yêu cầu đặt ra và có khả năng triển khai hoạt động trong thực tế. Việc quản lý taxi cũng được thực hiện dễ dàng hơn ngoài ra còn có thể tích hợp thêm một số tính năng hỗ trợ thêm cho tài xế như tìm đường, tìm địa điểm. Hệ thống được triển khai giúp có thêm một hướng tiếp cận mới trong việc quản lý và điều hành taxi trong bối cảnh hiện nay bằng cách tận dụng sự phát triển mạnh mẽ của các công nghệ và thiết bị di động những năm gần đây mà điển hình trong hệ thống này là Smartphone Android và công nghệ truyền tải thông tin thông qua mạng 3G. Đề tài được hoàn thành dựa trên nền tảng nghiên cứu lý thuyết, khảo sát yêu cầu đối với một hệ thống quản lý taxi trên thực tế giúp khắc phục những hạn chế của hệ thống truyền thống, tăng hiệu suất trong quá trình quản lý, giám sát, điều hành hệ thống tại trung tâm điều khiển, giảm được chi phí đầu tư cơ sở hạ tầng so với mô hình truyền thống.

## Những khó khăn và hạn chế

Trong quá trình thực hiện nhóm tác giả đã gặp một số khó khăn khác nhau về công nghệ, thiết bị và việc thử nghiệm đề tài trong thực tế. Cụ thể những khó khăn hạn chế gặp phải như sau:

* Những công nghệ áp dụng cho đề tài còn khá mới mẻ, chưa có tài liệu tiếng việt nên tốn thời gian đầu tư nghiên cứu nhiều, đặc biệt là Google Map API v2.
* Những hạn chế về mặt dịch vụ cũng như độ chính xác của Google Map ở Việt Nam gây ra một số sai lệch trong việc tìm kiếm và phân giải địa chỉ.
* Trong quá trình thực hiện thiếu điều kiện về thiết bị vì để có thể thực hiện chức năng VOIP, smartphone cần chạy trên nền tảng 3.0 trở lên.

## Hướng phát triển đề tài

Đề tài bước đầu đã xây dựng được hệ thống quản lý taxi thông qua thiết bị di động nền tảng Android. Nếu có thêm thời gian và kinh phí đầu tư thêm từ nhà trường và các công ty thì tương lai có thể phát triển hệ thống một cách hoàn chỉnh hơn, không chỉ có chức năng quản lý mà còn hỗ trợ tối đa cho tài xế. Một số hướng phát triển như:

* Giảm độ trễ và lọc tiếng ồn trong VOIP để giúp âm thanh được rõ hơn.
* Thêm khả năng quản lý thông qua việc sử dụng camera của thiết bị di động.
* Xây dựng một bản đồ riêng cho khu vực Hồ Chí Minh để tăng tốc độ xử lý và giảm sai sót trong quá trình phân giải địa chỉ thành tọa độ trên bản đồ.
* Kết nối với những thiết bị cảm biến gắn thêm vào xe để có thể quản lý được tình trạng hiện tại của xe như số xăng trong bình, tình trạng của động cơ, kiểm soát lỗi vi phạm của lái xe taxi như ngắt xung, kích xung. Hướng tới việc phối hợp sản xuất thiết bị giám sát hành trình chạy hệ điều hành Android.
* Phát triển hệ thống có thể tích hợp với những hệ thống, tài nguyên có sẵn của những nhà cung cấp dịch vụ taxi hiện nay.

Màn hình tìm kiếm địa điểm

Màn hình chính

Màn hình đăng nhập

# DANH MỤC TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | (2012, Dec.) Android Map API. [Online]. https://developers.google.com/maps/documentation/android/reference/com/google/android/gms/maps/package-summary |
| [2] | Google Place API. [Online]. https://developers.google.com/places/documentation/ |
| [3] | A. El-Rabbany, "Chaper 1: Introduction to GPS," in *Introduction to GPS - The Global Positioning System*. Boston, London: Artech House, 2002, p. 1. |
| [4] | T. D. Le, *Web Programming*. UIT, 2010. |
| [5] | S. Jose, *RESTful Java Web Services*. Birmingham: Packt Publishing Ltd, 2009. |
| [6] | A. Vukotic and J. Goodwill, *Apache Tomcat 7*. Apress, 2011. |
| [7] | G. Svennerberg, *Beginning Google Maps API 3*. Apress, 2010. |
| [8] | (2012, Dec.) Android Developer. [Online]. http://www.developer.android.com |
| [9] | S. Allamaraju, *RESTful Web Services Cookbook*. O’Reilly Media, Inc, 2010. |
| [10] | B. Burke, *RESTful Java with JAX-RS*. O’Reilly Media, Inc, 2010. |
| [11] | Google. (2012, Dec.) Google Map API v2. [Online]. https://developers.google.com/maps/documentation/android/ |
| [12] | M. Kalin, *Java Web Services: Up and Running*. O’Reilly Media, Inc, 2009. |
| [13] | G. Kerry, *trixbox CE 2.6*. Birmingham: Packt Publishing Ltd, 2009. |
| [14] | R. Leonard and R. Sam, *RESTful Web Services*. O’Reilly Media, Inc, 2007. |
| [15] | T. A. Nguyen, L. T. K. Ton, and N. H. Tran, *Android programming tutorial*. UIT, 2011. |

1. http://www.thongtincongnghe.com/article/35078 [↑](#footnote-ref-1)
2. http://gps.vietmap.vn/giai-phap/taxi [↑](#footnote-ref-2)
3. http://vietbao.vn/Vi-tinh-Vien-thong/Mashup-mon-qua-thu-vi-cua-trao-luu-Web-20/11054949/217/ [↑](#footnote-ref-3)
4. Windows Phone : http://www.windowsphone.com/en-us [↑](#footnote-ref-4)
5. Bing Maps REST Services: http://msdn.microsoft.com/en-us/library/ff701713.aspx [↑](#footnote-ref-5)
6. Bing Spatial Data Services : http://msdn.microsoft.com/en-us/library/ff701734.aspx [↑](#footnote-ref-6)
7. Andy Rubin: http://en.wikipedia.org/wiki/Andy\_Rubin [↑](#footnote-ref-7)
8. Open Handset Alliance: http://www.openhandsetalliance.com/ [↑](#footnote-ref-8)
9. Thị phần các hệ điền hành trên thiết bị di động: http://dantri.com.vn/suc-manh-so/android-dang-thong-tri-thi-truong-smartphone-658334.htm [↑](#footnote-ref-9)
10. IDC : http://en.wikipedia.org/wiki/International\_Data\_Corporation [↑](#footnote-ref-10)
11. Thống kê trên Google Market: http://vi.wikipedia.org/wiki/Android [↑](#footnote-ref-11)
12. Android version history: http://en.wikipedia.org/wiki/Android\_version\_history [↑](#footnote-ref-12)
13. HTTP proxy: http://vi.wikipedia.org/wiki/Proxy\_server [↑](#footnote-ref-13)
14. http://vi.wikipedia.org/wiki/C%E1%BA%A5u\_tr%C3%BAc\_ARM. [↑](#footnote-ref-14)
15. Dalvik: http://en.wikipedia.org/wiki/Dalvik\_(software) [↑](#footnote-ref-15)
16. Google Play service SDK: http://developer.android.com/google/play-services/index.html [↑](#footnote-ref-16)
17. Google APIs Console: https://code.google.com/apis/console/ [↑](#footnote-ref-17)
18. Hash: http://en.wikipedia.org/wiki/Hash\_function [↑](#footnote-ref-18)
19. SHA-1: http://en.wikipedia.org/wiki/SHA-1 [↑](#footnote-ref-19)
20. Keytool: http://docs.oracle.com/javase/1.4.2/docs/tooldocs/windows/keytool.html [↑](#footnote-ref-20)
21. Hexa: http://en.wikipedia.org/wiki/Hexadecimal [↑](#footnote-ref-21)
22. Keystore: http://docs.oracle.com/javase/1.4.2/docs/api/java/security/KeyStore.html [↑](#footnote-ref-22)
23. Lockheed Martin: http://vi.wikipedia.org/wiki/Lockheed\_Martin [↑](#footnote-ref-23)
24. Composite: http://vi.wikipedia.org/wiki/V%E1%BA%ADt\_li%E1%BB%87u\_composite [↑](#footnote-ref-24)
25. Sóng mang trong điều chế tín hiệu: http://vi.wikipedia.org/wiki/%C4%90i%E1%BB%81u\_ch%E1%BA%BF\_t%C3%ADn\_hi%E1%BB%87u [↑](#footnote-ref-25)
26. Doppler: http://vi.wikipedia.org/wiki/Hi%E1%BB%87u\_%E1%BB%A9ng\_Doppler [↑](#footnote-ref-26)
27. http://www.ibm.com/developerworks/webservices/library/ws-restful/ [↑](#footnote-ref-27)
28. http://www.ibm.com/developerworks/webservices/library/ws-restful/ [↑](#footnote-ref-28)
29. Trích tài liệu What’s New in MySQL 5.5 Performance and Scalability [↑](#footnote-ref-29)
30. http://www.json.org/index.html [↑](#footnote-ref-30)
31. https://sites.google.com/site/gson/gson-user-guide [↑](#footnote-ref-31)
32. H.323: http://en.wikipedia.org/wiki/H.323 [↑](#footnote-ref-32)
33. Henning Schulzrinne: http://en.wikipedia.org/wiki/Henning\_Schulzrinne [↑](#footnote-ref-33)
34. Mark Handley: http://en.wikipedia.org/wiki/Mark\_Handley\_(computer\_scientist) [↑](#footnote-ref-34)
35. RFC 2453:http://tools.ietf.org/html/rfc2453 [↑](#footnote-ref-35)
36. 3rd Generation Partnership Project :http://en.wikipedia.org/wiki/3GPP [↑](#footnote-ref-36)
37. IP Multimedia Subsystem: http://en.wikipedia.org/wiki/IP\_Multimedia\_Subsystem [↑](#footnote-ref-37)
38. RFC 3261: http://tools.ietf.org/html/rfc3261 [↑](#footnote-ref-38)
39. RFC 3261: http://www.ietf.org/rfc/rfc3261.txt [↑](#footnote-ref-39)
40. http://www.advantrack.com/gps-tracking-atti-software/software.aspx [↑](#footnote-ref-40)
41. http://opengts.sourceforge.net/ [↑](#footnote-ref-41)
42. http://gps.vietmap.vn/giai-phap/taxi [↑](#footnote-ref-42)
43. http://www.maydinhvi.org/may-dinh-vi-gps-tracker-vt-300 [↑](#footnote-ref-43)
44. http://www.vatgia.com/7030/1932150/h%E1%BB%87-th%E1%BB%91ng-%C4%91%E1%BB%8Bnh-v%E1%BB%8B-vi%E1%BB%87t-map-gps-vt10s.html [↑](#footnote-ref-44)
45. http://www.thanhbinhauto.com/10131/detail-63-Thiet\_bi\_GPS\_Tracking\_VT300s.htma [↑](#footnote-ref-45)
46. http://www.hopdendinhvi.com/index.php/giai-phap-quan-ly-giam-sat-thiet-bi-quan-ly-hanh-trinh/giai-phap-quan-ly-giam-sat-thiet-bi-quan-ly-hanh-trinh-su-dung-cho-xe-taxi/50-quan-ly-xe-taxi-bang-thiet-bi-giam-sat-hanh-trinh [↑](#footnote-ref-46)
47. http://www.hopdendinhvi.com/index.php/thiet-bi-quan-ly-hanh-trinh-hop-den-giam-sat-xe-oto-xe-khach-xe-lu-hanh-xe-taxi-xe-tu-lai-xe-may-xe-moto-phuong-tien-ca-nhan/thiet-bi-quan-ly-hanh-trinh-hop-den-giam-sat-xe-oto-xe-khach-xe-lu-hanh-xe-taxi-xe-tu-lai-xe-may-xe-moto-phuong-tien-ca-nhan [↑](#footnote-ref-47)
48. http://vi.wikipedia.org/wiki/HSDPA [↑](#footnote-ref-48)
49. http://vi.wikipedia.org/wiki/D%E1%BB%8Bch\_v%E1%BB%A5\_v%C3%B4\_tuy%E1%BA%BFn\_g%C3%B3i\_t%E1%BB%95ng\_h%E1%BB%A3p [↑](#footnote-ref-49)
50. Trích tài liệu ZOIPER WEB API [↑](#footnote-ref-50)