**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**NGUYỄN HỒNG PHÚC**

**LÊ PHƯỚC THẠCH**

**KHÓA LUẬN TỐT NGHIỆP**

**NGHIÊN CỨU CÔNG NGHỆ RESTFUL WEBSERVICE VÀ PHÁT TRIỂN ỨNG DỤNG QUẢNG BÁ DU LỊCH TRÊN ANDROID**

**KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM**

**TP. HỒ CHÍ MINH 6, 2015**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**NGUYỄN HỒNG PHÚC – 10520278**

**LÊ PHƯỚC THẠCH – 10520204**

**KHÓA LUẬN TỐT NGHIỆP**

**NGHIÊN CỨU CÔNG NGHỆ RESTFUL WEBSERVICE VÀ PHÁT TRIỂN ỨNG DỤNG QUẢNG BÁ DU LỊCH TRÊN ANDROID**

**KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM**

**GIẢNG VIÊN HƯỚNG DẪN**

**THS. TRẦN ANH DŨNG**

**TP. HỒ CHÍ MINH 6, 2015**

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …………………… ngày ………………….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

* 1. …………………………………………. – Chủ tịch.
  2. …………………………………………. – Thư ký.
  3. …………………………………………. – Ủy viên.
  4. …………………………………………. – Ủy viên.

|  |  |
| --- | --- |
| ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH  **TRƯỜNG ĐẠI HỌC**  **CÔNG NGHỆ THÔNG TIN** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc Lập - Tự Do - Hạnh Phúc** |
|  | *TP. HCM, ngày…..tháng…..năm……..* |

**NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP**

**(CỦA CÁN BỘ HƯỚNG DẪN)**

|  |  |  |
| --- | --- | --- |
| **Tên khóa luận:** | | |
| **NGHIÊN CỨU CÔNG NGHỆ RESTFUL WEBSERVICE**  **VÀ PHÁT TRIỂN ỨNG DỤNG QUẢNG BÁ DU LỊCH TRÊN ANDROID** | | |
| **Nhóm SV thực hiện:** | | **Cán bộ hướng dẫn:** |
| Nguyễn Hồng Phúc 10520278 |  |  |
| Lê Phước Thạch 10520204 |  |  |
| **Đánh giá Khóa luận**   1. Về cuốn báo cáo:   Số trang Số chương  Số bảng số liệu Số hình vẽ  Số tài liệu tham khảo Sản phẩm  Một số nhận xét về hình thức cuốn báo cáo:   1. Về nội dung nghiên cứu: 2. Về chương trình ứng dụng: 3. Về thái độ làm việc của sinh viên:   **Đánh giá chung:**  **Điểm từng sinh viên:**  Nguyễn Hồng Phúc **:………../10**  Lê Phước Thạch **:………../10** | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH  **TRƯỜNG ĐẠI HỌC**  **CÔNG NGHỆ THÔNG TIN** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc Lập - Tự Do - Hạnh Phúc** | |  | *TP. HCM, ngày…..tháng…..năm……..* |   **NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP**  **(CỦA CÁN BỘ PHẢN BIỆN)** | **Người nhận xét**  (Ký tên và ghi rõ họ tên) |

|  |  |  |
| --- | --- | --- |
| **Tên khóa luận:** | | |
| **NGHIÊN CỨU CÔNG NGHỆ RESTFUL WEBSERVICE**  **VÀ PHÁT TRIỂN ỨNG DỤNG QUẢNG BÁ DU LỊCH TRÊN ANDROID** | | |
| **Nhóm SV thực hiện:** | | **Cán bộ phản biện:** |
| Nguyễn Hồng Phúc 10520278 |  |  |
| Lê Phước Thạch 10520204 |  |  |
| **Đánh giá Khóa luận**   1. Về cuốn báo cáo:   Số trang Số chương  Số bảng số liệu Số hình vẽ  Số tài liệu tham khảo Sản phẩm  Một số nhận xét về hình thức cuốn báo cáo:   1. Về nội dung nghiên cứu: 2. Về chương trình ứng dụng: 3. Về thái độ làm việc của sinh viên:   **Đánh giá chung:**  **Điểm từng sinh viên:**  Nguyễn Hồng Phúc **:………../10**  Lê Phước Thạch **:………../10** | | |

|  |  |
| --- | --- |
|  | **Người nhận xét**  (Ký tên và ghi rõ họ tên) |
| ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH  TRƯỜNG ĐẠI HỌC  CÔNG NGHỆ THÔNG TIN | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc Lập - Tự Do - Hạnh Phúc** |

**ĐỀ CƯƠNG CHI TIẾT**

|  |  |
| --- | --- |
| **NGHIÊN CỨU CÔNG NGHỆ RESTFUL WEBSERVICE**  **VÀ PHÁT TRIỂN ỨNG DỤNG QUẢNG BÁ DU LỊCH TRÊN ANDROID** | |
| **Cán bộ hướng dẫn: ThS. Trần Anh Dũng** | |
| **Thời gian thực hiện:** Từ ngày 19/03/2015 đến ngày 22/06/2015 | |
| **Sinh viên thực hiện:**  **Nguyễn Hồng Phúc – 10520278**  **Lê Phước Thạch – 1050000** | |
| **Nội dung đề tài:** *Nghiên cứu công nghệ RESTful webservice triển khai kiến trúc một dịch vụ web cung cấp các thông tin, cách thức thao tác với thông tin địa điểm du lịch và xây dựng ứng dụng minh họa trên Android.* | |
| **Kế hoạch thực hiện:**  *Khóa luận được thực hiện trong 16 tuần, nhóm chúng em chia công việc theo tuần để dễ dàng đánh giá kết quả. Kế hoạch thực hiện cụ thể như sau:*   |  |  |  | | --- | --- | --- | | **Tuần** | **Công việc** | **Người thực hiện** | | 1-2 (2 tuần) | * Tìm hiểu một số công nghệ triển khai RESTful và các vấn đề liên quan * Khảo sát và xác định ứng dụng cụ thể sẽ làm. | Phúc  + Thạch | | 3(1 tuần) | * Phát thảo đề cương các nội dung tìm hiểu * Mô tả chi tiết ý tưởng ứng dụng và mô tả các dữ liệu sử dụng | Phúc  + Thạch | | 4-5(2 tuần) | * Chọn lựa công nghệ triển khai, nghiên cứu viết web service mẫu * Thiết kế kiến trúc hệ thống | Thạch | | * Thiết kế dữ liệu, thiết kế database, thiết kế API * Tìm hiểu Google API, viết mã lấy dữ liệu địa điểm sơ bộ | Phúc | | 6-7(2 tuần) | * Tìm hiểu và viết ứng dụng Android mẫu * Viết skeleton cho ứng dụng Android | Thạch | | * Chọn lựa các phương thức bảo mật cho dịch vụ web * Viết ứng dụng mẫu sử dụng phương thức bảo mật trên | Phúc | | 8-10(2 tuần) | * Thực hiện xử lý dữ liệu thô ban đầu, nhập dữ liệu vào cơ sở dữ liệu | Phúc | | * Kiểm tra viết lại dịch vụ web, tích hợp hệ thống bảo mật | Thạch | | 11(1 tuần) | * Viết API | Phúc | |  | * Kết nối dữ liệu và hiển thị trên ứng dụng Android | Thạch | | 12(1 tuần) | * Kiểm lỗi hệ thống và sửa lỗi nếu có | Phúc + Thạch | | 13–14 (2 tuần) | * Thời gian dự trữ cho việc viết báo cáo và sửa lỗi nâng cấp ứng dụng | Phúc + Thạch | | 15–16 (2 tuần) | * Hoàn tất báo cáo. | Phúc + Thạch | | |
| **Xác nhận của CBHD**  (Ký tên và ghi rõ họ tên) | **TP. HCM, ngày….tháng …..năm…..**  **Sinh viên**  (Ký tên và ghi rõ họ tên) |

**LỜI CÁM ƠN**

**MỤC LỤC**

[CHƯƠNG 1. GIỚI THIỆU 3](#_Toc422689235)

[1.1. Giới thiệu đề tài 3](#_Toc422689236)

[1.2. Mục tiêu đề tài 3](#_Toc422689237)

[1.3. Đối tượng nghiên cứu 3](#_Toc422689238)

[1.4. Phạm vi nghiên cứu 3](#_Toc422689239)

[1.5. Nội dung 3](#_Toc422689240)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 5](#_Toc422689241)

[2.1. Giới thiệu dịch vụ web 5](#_Toc422689242)

[2.1.1. Giới thiệu 5](#_Toc422689243)

[2.1.2. Đặc điểm 5](#_Toc422689244)

[2.1.3. Ứng dụng 7](#_Toc422689245)

[2.1.4. Kiến trúc 7](#_Toc422689246)

[2.1.5. Khái quát các mô hình kiến trúc dịch vụ web 11](#_Toc422689247)

[2.1.5.1. Kiến trúc hướng thông điệp (Message Oriented Model) 11](#_Toc422689248)

[2.1.5.2. Kiến trúc hướng dịch vụ (Service Oriented Model) 12](#_Toc422689249)

[2.1.5.3. Kiến trúc hướng tài nguyên (Resource Oriented Model) 13](#_Toc422689250)

[2.1.5.4. Mô hình chính sách (Policy Model) 13](#_Toc422689251)

[2.2. Xu hướng phát triển dịch vụ web 13](#_Toc422689252)

[2.3. REST 14](#_Toc422689253)

[2.4. Các khái niệm chính 14](#_Toc422689254)

[2.4.1. Tài nguyên (Resource) 14](#_Toc422689255)

[2.4.2. Đại diện (Representation) 15](#_Toc422689256)

[2.4.3. Tương tác phi trạng thái (Stateless) 16](#_Toc422689257)

[2.4.4. Giao diện đồng nhất (Uniform interface) 16](#_Toc422689258)

[2.4.5. Hoạt động GET/RETRIEVE 17](#_Toc422689259)

[2.4.6. Hoạt động PUT/CREATE 18](#_Toc422689260)

[2.4.7. Hoạt động POST/UPDATE 19](#_Toc422689261)

[2.4.8. Hoạt động DELETE/DELETE 20](#_Toc422689262)

[2.4.9. Mô hình trưởng thành của Richardson (RMM) 21](#_Toc422689263)

[2.4.9.1. RMM Level 0 22](#_Toc422689264)

[2.4.9.2. RMM Level 1 22](#_Toc422689265)

[2.4.9.3. RMM Level 2 23](#_Toc422689266)

[2.4.9.4. RMM Level 3 23](#_Toc422689267)

[2.5. Mục tiêu chính 25](#_Toc422689268)

[2.6. Ràng buộc 25](#_Toc422689269)

[2.7. Cơ chế hoạt động 28](#_Toc422689270)

[2.8. RESTful API 28](#_Toc422689271)

[2.9. Bảo mật REST API 28](#_Toc422689272)

[2.9.1. JSON Web Token 28](#_Toc422689273)

[2.9.2. Cấu trúc của JSON Web Token 29](#_Toc422689274)

[2.9.2.1. Nội dung (Payload / Claims) 29](#_Toc422689275)

[2.9.2.2. Nội dung (Payload / Claims) 30](#_Toc422689276)

[2.9.2.3. Chữ ký (Signature) 32](#_Toc422689277)

[2.10. REST và SOAP 38](#_Toc422689278)

[2.11. Kết quả so sánh 38](#_Toc422689279)

[2.12. REST trong Java 43](#_Toc422689280)

[CHƯƠNG 3. DỊCH VỤ WEB DU LỊCH 45](#_Toc422689281)

[3.1. Giới thiệu dịch vụ 45](#_Toc422689282)

[3.2. Mô tả 45](#_Toc422689283)

[3.3. Danh sách các yêu cầu 46](#_Toc422689284)

[3.4. Check-in địa điểm 47](#_Toc422689285)

[3.5. Phân tích 48](#_Toc422689286)

[3.5.1. Yêu cầu đăng nhập 48](#_Toc422689287)

[3.5.1.1. Đăng nhập 49](#_Toc422689288)

[3.5.1.2. Đăng xuất 49](#_Toc422689289)

[3.5.1.3. Các yêu cầu khác 49](#_Toc422689290)

[3.5.2. Yêu cầu tìm kiếm 50](#_Toc422689291)

[3.5.3. Bảo mật API 50](#_Toc422689292)

[3.6. Kiến trúc hệ thống 50](#_Toc422689293)

[3.6.1. Mô hình dịch vụ RESTful API 50](#_Toc422689294)

[3.6.2. Kiến trúc xử lý đa tầng 51](#_Toc422689295)

[3.6.3. Mô hình cài đặt chi tiết 53](#_Toc422689296)

[3.7. Thiết kế dữ liệu 55](#_Toc422689297)

[3.7.1. Mô hình ERD 55](#_Toc422689298)

[3.7.2. Mô hình quan hệ 55](#_Toc422689299)

[3.7.3. Thiết kế chỉ mục 56](#_Toc422689300)

[3.7.3.1. Spatial Index 56](#_Toc422689301)

[3.7.3.2. Full search index 57](#_Toc422689302)

[3.8. Thiết kế REST API 58](#_Toc422689303)

[3.9. Kết quả thử nghiệm 62](#_Toc422689304)

[CHƯƠNG 4. ỨNG DỤNG ANDROID QUẢNG BÁ DU LỊCH 63](#_Toc422689305)

[4.1. Xác định yêu cầu 63](#_Toc422689306)

[4.2. Giới thiệu khái quát 63](#_Toc422689307)

[4.3. Khái quát các chức năng 63](#_Toc422689308)

[4.3.1. Chức năng Đăng nhập 63](#_Toc422689309)

[4.3.1.1. Mục tiêu 63](#_Toc422689310)

[4.3.1.2. Kết quả 63](#_Toc422689311)

[4.3.2. Chức năng tìm kiếm địa điểm – quán cafe 63](#_Toc422689312)

[4.3.2.1. Mục tiêu 64](#_Toc422689313)

[4.3.2.2. Kết quả 64](#_Toc422689314)

[4.3.3. Chức năng tìm kiếm địa điểm – nhà hàng 64](#_Toc422689315)

[4.3.3.1. Mục tiêu 64](#_Toc422689316)

[4.3.3.2. Kết quả 64](#_Toc422689317)

[4.3.4. Chức năng tìm kiếm địa điểm – quán ăn đường phố 64](#_Toc422689318)

[4.3.4.1. Mục tiêu 64](#_Toc422689319)

[4.3.4.2. Kết quả 64](#_Toc422689320)

[4.3.5. Chức năng tìm kiếm địa điểm – bệnh viện, dịch vụ nghỉ dưỡng 64](#_Toc422689321)

[4.3.5.1. Mục tiêu 64](#_Toc422689322)

[4.3.5.2. Kết quả 65](#_Toc422689323)

[4.3.6. Chức năng tìm kiếm địa điểm – địa điểm mua sắm 65](#_Toc422689324)

[4.3.6.1. Mục tiêu 65](#_Toc422689325)

[4.3.6.2. Kết quả 65](#_Toc422689326)

[4.3.7. Chức năng tìm kiếm địa điểm – địa điểm tham quan cộng đồng 65](#_Toc422689327)

[4.3.7.1. Mục tiêu 65](#_Toc422689328)

[4.3.7.2. Kết quả 65](#_Toc422689329)

[4.3.8. Chức năng tìm tìm kiếm nhanh 65](#_Toc422689330)

[4.3.8.1. Mục tiêu 65](#_Toc422689331)

[4.3.8.2. Kết quả 66](#_Toc422689332)

[4.3.9. Chức năng chỉ đường đi bằng google map 66](#_Toc422689333)

[4.3.9.1. Mục tiêu 66](#_Toc422689334)

[4.3.9.2. Kết quả 66](#_Toc422689335)

[4.3.10. Chức năng tìm chia sẻ cho cho bạn bè, người thân thông qua mạng xã hội facebook 66](#_Toc422689336)

[4.3.10.1. Mục tiêu 66](#_Toc422689337)

[4.3.10.2. Kết quả 66](#_Toc422689338)

[4.3.11. Chức năng đánh giá địa điểm 66](#_Toc422689339)

[4.3.11.1. Mục tiêu 66](#_Toc422689340)

[4.3.11.2. Kết quả 66](#_Toc422689341)

[4.4. Phân tích 66](#_Toc422689342)

[4.4.1. Lược đồ use case 66](#_Toc422689343)

[4.4.2. Mô tả lược đồ use case 67](#_Toc422689344)

[4.5. Đặc tả một số use case chính 69](#_Toc422689345)

[4.5.1. Use case “Đăng nhập” 69](#_Toc422689346)

[4.5.2. Use case “Tìm kiếm quán cafe” 69](#_Toc422689347)

[4.6. Thiết kế màn hình 71](#_Toc422689348)

[4.6.1. Thiết kế giao diện màn hình Đăng nhập 73](#_Toc422689349)

[4.6.2. Thiết kế giao diện màn hình menu 75](#_Toc422689350)

[4.6.3. Thiết kế màn hình danh sách 76](#_Toc422689351)

[4.6.3.1. Thiết kế giao diện màn hình danh sách quán cafe 76](#_Toc422689352)

[4.6.3.2. Thiết kế giao diện màn hình danh sách nhà hàng 77](#_Toc422689353)

[4.6.3.3. Thiết kế giao diện màn hình danh sách quán ăn 78](#_Toc422689354)

[4.6.3.4. Thiết kế giao diện màn hình danh sách cửa hàng 79](#_Toc422689355)

[4.6.3.5. Thiết kế giao diện màn hình danh sách địa điểm nghỉ dưỡng 80](#_Toc422689356)

[4.6.3.6. Thiết kế giao diện màn hình danh sách địa điểm công cộng 81](#_Toc422689357)

[4.6.4. Thiết kế giao diện màn hình chi tiết 82](#_Toc422689358)

[4.6.4.1. Thiết kế giao diện màn hình chi tiết quán cafe 82](#_Toc422689359)

[4.6.4.2. Thiết kế giao diện màn hình chi tiết nhà hàng 83](#_Toc422689360)

[4.6.4.3. Thiết kế giao diện màn hình chi tiết quán ăn 84](#_Toc422689361)

[4.6.4.4. Thiết kế giao diện màn hình chi tiết cửa hàng 85](#_Toc422689362)

[4.6.4.5. Thiết kế giao diện màn hình chi tiết địa điểm nghỉ dưỡng 86](#_Toc422689363)

[4.6.4.6. Thiết kế giao diện màn hình chi tiết địa điểm công cộng 87](#_Toc422689364)

[4.6.5. Thiết kế màn hình chia sẻ 88](#_Toc422689365)

[CHƯƠNG 5. KẾT LUẬN 89](#_Toc422689366)

[5.1. Kết quả đạt được 89](#_Toc422689367)

[5.2. Hướng phát triển 89](#_Toc422689368)

[TÀI LIỆU THAM KHẢO 90](#_Toc422689369)

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

TÓM TẮT KHÓA LUẬN

MỞ ĐẦU

Công nghệ thông tin phát triển mạnh mẽ đồng hành với nhu cầu trao đổi thông tin và sử dụng các dịch vụ ngày càng tăng, có nhiều phương pháp giúp người dùng có thể sử dụng nhiều dịch vụ, chia sẻ dữ liệu như: rmi, dcom hoặc corba service nhưng còn mang nhiều hạn chế.

Dịch vụ web ra đời là một cuộc cách mạng hóa cách thức hoạt động của các dịch vụ B2B và B2C. Dịch vụ web đã mở ra một hướng mới cho việc phát triển các ứng dụng trên Internet. Web services kết hợp nhiều công nghệ khác nhau cho phép hai ứng dụng có thể độc lập ngôn ngữ, độc lập hệ điều hành có thể trao đổi thông tin với nhau thông qua môi trường Internet. Dich vụ web ngày nay nổi lên hai xu hướng nổi trội đó là dịch vụ web dựa trên mô hình SOAP (Simple Object Access Protocol) và giao thức REST (Representational State Transfer). Cả hai mô hình đều đang và được sử dụng rộng rãi và mang lại những lợi ích to lớn nhất định.

Ngày nay với sự bùng nổ của các thiết bị IoT (Internet of Things), khi mà mỗi đồ vật, vật dụng thông thường hứa hẹn sẽ có thể kết nối không dây và giao tiếp với nhau thông qua Internet. Chúng có thể là điện thoại thông minh, máy tính bảng, đồng hồ đeo tay, kính mác, tivi, tủ lạnh… đã tạo ra viễn cảnh con người có thể kiểm soát mọi đồ vật của mình qua mạng chỉ bằng các thiết bị thông minh nhỏ bé trên tay mà không cần phải tiếp xúc trực tiếp. Với xu hướng phát triển của các thiết bị IoT thì dịch vụ web lại một lần nữa đóng vai trò quan trọng trong việc giao tiếp, trao đổi thông tin. Phát triển dịch vụ web theo mô hình REST ngày càng nổi lên là sự thay thế mô hình SOAP trước đó với những ưu điểm đơn giản, gọn nhẹ và tốc độ thực thi. Bằng chứng là ngày càng nhiều các công ty lớn đã và đang cung cấp dịch vụ web theo mô hình REST như Facebook, Google, Microsoft, Twitter. Nắm bắt được xu hướng đó mục tiêu chính của đề tài là nghiên cứu, tìm hiểu về phát triển dịch vụ web theo mô hình REST và đồng thời ứng dụng những kiến thức đạt được xây dựng một ứng dụng minh họa trên Android.

# CHƯƠNG 1. GIỚI THIỆU

## Giới thiệu đề tài

Dịch vụ web RESTful ngày càng phát triển rộng rãi, cùng với sự bùng nổ của ứng dụng di động. Đề tài đi sâu nghiên cứu về khái niệm và cách thức phát triển dịch vụ web theo mô hình REST, cùng làm sáng tỏ những lợi ích của mô hình REST đem lại.

## Mục tiêu đề tài

## Đối tượng nghiên cứu

* Mô hình phát triển dịch vụ web theo kiến trúc REST.
* Các thành phần, cơ chế hoạt động cùng với các ràng buộc RESTful.
* Phát triển dịch vụ web REST trên nền tảng công nghệ JAX-RS sử dụng ngôn ngữ Java.

## Phạm vi nghiên cứu

Luận văn đi sâu vào nghiên cứu về REST full webservice cũng như ứng dụng của REST ful webservice trong thời đại công nghệ thông tin. Ngoài ra, luận văn còn demo ứng dụng của REST full webservice bằng ứng dụng Quảng bá du lịch trên Android.

## Nội dung

Báo cáo gồm có 5 chương:

Chương 1: Giới thiệu mục tiêu, phạm vị, đối tượng đề tài

Chương 2: Tổng quan về dịch vụ web, lịch sử phát triển, các mô hình và đi sâu nghiên cứu về các thành phần của mô hình REST, cơ chế hoạt động, các ràng buộc cũng như cách thức áp dụng trong thực tiễn.

Chương 3: Áp dụng các cơ sở lý thuyết trong chương trên vào phân tích, thiết kế và phát triển một dịch vụ web du lịch theo mô hình REST cung cấp các chức năng tìm kiếm thông tin các địa điểm du lịch, hướng dẫn chỉ đường, đồng thời cho phép người dùng gửi đánh giá, chia sẻ, bình luận đối với địa điểm đó.

Chương 4: Thiết kế, xây dựng ứng dụng sử dụng dịch vụ web du lịch trên nền tảng Android.

Chương 5: Nêu ra kết luận và hướng phát triển cho đề tài.

# CƠ SỞ LÝ THUYẾT

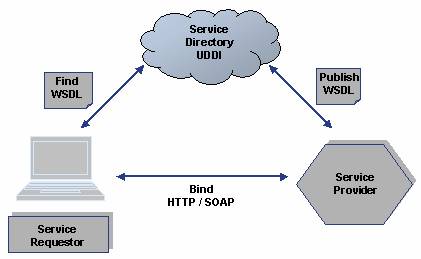
## Giới thiệu dịch vụ web

* + 1. Giới thiệu

Dịch vụ web (Web service) là sự kết hợp các máy tính cá nhân với các thiết bị khác, các cơ sở dữ liệu và các mạng máy tính để tạo thành một cơ cấu tính toán ảo mà người sử dụng có thể làm việc thông qua các trình duyệt mạng. Bản thân các dịch vụ này sẽ chạy trên các máy chủ trên nền Internet chứ không phải là các máy tính cá nhân, do vậy có thể chuyển các chức nǎng từ máy tính cá nhân lên Internet. Người sử dụng có thể làm việc với các dịch vụ thông qua bất kỳ loại máy nào có hỗ trợ web service và có truy cập Internet, kể cả các thiết bị cầm tay.

* + 1. Đặc điểm
* **Cung cấp các chức năng ra ngoài mạng lưới** – Dịch vụ web là các thành phần giao diện chức năng có thể được gọi từ xa thông qua giao thức HTTP, được kích hoạt bằng sử dụng các request HTTP. Một khi được cung cấp ra ngoài mạng lưới thì các ứng dụng thứ ba có thể sử dụng dịch vụ lẫn nhau, mở rộng nhiều cơ hội kinh doanh.
* **Kết nối các ứng dụng khác nhau** – Dịch vụ web cho phép các ứng dụng được phát triển dưới các nền tảng ngôn ngữ, công nghệ, môi trường khác nhau có thể giao tiếp, chia sẻ dữ liệu được với nhau. Chẳng hạn, ứng dụng VB hoặc .NET có thể giao tiếp với dịch vụ web phát triển bằng ngôn ngữ Java và ngược lại. Vì thế, dịch vụ web được sử dụng để phát triển các nền tảng ứng dụng độc lập về công nghệ.
* **Giao thức chuẩn –** Dịch vụ web sử dụng các giao thức chuẩn công nghiệp trong việc truyền thông. Tất cả các tầng (Service Transport, XML Messaging, Service Description và Service Directory layers) đều sử dụng các giao thức chuẩn. Sự chuẩn hóa mang lại nhiều lợi ích nghiệp vụ như sự lựa chọn đa dạng, giảm thiểu chi phí trong cạnh tranh và nâng cao hiệu quả.
* **Truyền thông chi phí thấp –** Dịch vụ web sử dụng SOAP trên nền giao thức HTTP trong việc truyền thông, vì thế chúng ta có thể sử dụng môi trường Internet chi phí thấp trong việc triển khai dịch vụ. Giải pháp này ít tốn kém so với các giải pháp độc quyền như EDI/B2B.
* **Hỗ trợ các phương tiện truyền thông khác –** Bên cạnh SOAP trên HTTP, dịch vụ web cũng có thể được triển khai trên các cơ chế truyền thông tin cậy khác nhau. Vì thế cung cấp sự linh động trong việc chọn lựa sử dụng phương tiện truyền thông phù hợp với từng yêu cầu cụ thể. Ví dụ, một dịch vụ web có thể được triển khai trên giao thức ftp (Web service over FTP).
* **Độc lập về ngôn ngữ, nền tảng, giao thức –** Dịch vụ web là các thành phần ứng dụng tự mô tả đóng gói các chức năng cụ thể của ứng dụng. Dịch vụ web có thể truy cập thông qua các giao thức truyền thông chuẩn như XML và SOAP. Các dịch vụ web có thể được phát triển trên nhiều nền tảng công nghệ khác nhau (như C++, Java, .NET, PHP, Perl,...) và bất kỳ ứng dụng hoặc dịch vụ web khác.
* **Web Services Sharing –** Ngày nay với sự phức tạp của các tổ chức kinh tế sử dụng các nền tảng công nghệ khác nhau như EAI, EDI, B2B, Portals,… trong tính toán phân tán. Dịch vụ web hỗ trợ tất cả các công nghệ trên, do đó giúp cho tổ chức doanh nghiệp sử dụng các đầu tư hiện tại cho các công nghệ khác.
* **Dịch vụ web tự mô tả -** Dịch vụ web là một ứng dụng tự mô tả, thông qua đó cho phép giảm thời gian phát triển ứng dụng, cho phép các thành viên kinh doanh liên quan nhanh chóng phát triển ứng dụng và bắt đầu việc kinh doanh của mình. Giúp cho việc kinh doanh giảm thiểu thời gian và tiền bạc thông qua việc cắt giảm thời gian phát triển ứng dụng.
* **Tự động khám phá –** Dịch vụ web hỗ trợ cơ chế tự động khám phá giúp cho các bên kinh doanh dễ dàng tiềm kiếm các nhà cung cấp dịch vụ. Nó cũng giúp cho khách hàng có thể tìm kiếm dịch vụ mong muốn dễ dàng.
* **Mở rộng cơ hội kinh doanh –** Dịch vụ web mở cửa cho các cơ hội kinh doanh mới bằng cách cung cấp cơ chế dễ dàng kết nối với các bên liên quan.
  + 1. Ứng dụng
* Ngày nay dịch vụ web được sử dụng rất nhiều trong những lĩnh vực khác nhau của cuộc sống như:
* Tìm kiếm các thông tin về các khách sạn ở các thành phố hoặc các trung tâm để liên lạc đặt phòng theo yêu cầu của khách hàng.
* Dịch vụ chọn lọc và phân loại tin tức: Là những hệ thống thư viện kết nối đến các Web Portal để tìm kiếm các thông tin từ các nhà xuất bản có chứa những từ khóa muốn tìm.
* Các ứng dụng dịch vụ du lịch (cung cấp giá vé, thông tin địa điểm,…)
* Thông tin giao dịch trực tuyến như đặt vé máy bay, đặt khách sạn, thông tin thuê xe…
* Thông tin thương mại bao gồm nhiều nội dung, nhiều mục tin như: Dự báo thời tiết, thông tin sức khỏe, lịch bay, tỷ giá cổ phiếu…
  + 1. Kiến trúc

Mô hình hoạt động



Hình 1: Mô hình Web service

* Mô hình dịch vụ web đơn giản định nghĩa cách thức tương tác giữa các thành phần như bên sử dụng dịch vụ (Service Requestor), cung cấp dịch vụ (Service Provider), bên trung lưu trữ thông tin dịch vụ (Service Directory UDDI).
* Bên sử dụng dịch vụ tìm kiếm các thông tin dịch vụ thông qua bên trung gian lưu trữ. Kết quả trả về thông tin dịch vụ (WSDL) – là thông tin đặc tả chi tiết các dịch vụ mà các nhà cung cấp dịch vụ đã đăng ký trước với bên trung gian lưu trữ. Sau khi có được thông tin mô tả dịch vụ (WSDL), Bên sử dụng dịch vụ kết nối đến các nhà cung cấp dịch vụ bằng cách triệu gọi các dịch vụ thông qua giao thức hỗ trợ.

Một số cơ chế tương tác giữa các thành phần này:

**Service**: là cơ chế cho phép bên sử dụng dịch vụ xác định và triệu gọi các dịch vụ từ xa thông qua mạng mà không phụ thuộc vào vị trí địa lý, hệ điều hành sử dụng hay ngôn ngữ lập trình sử dụng.

**Message**: là phương tiện giao tiếp giữa bên cung cấp dịch vụ và bên sử dụng dịch vụ. Một thông điệp có thể là một yêu cầu từ bên sử dụng dịch vụ gửi đến bên cung cấp dịch vụ hay là một phản hồi từ bên cung cấp dịch vụ về cho bên sử dụng dịch vụ. Các thông điệp này được định nghĩa bằng các ngôn ngữ đánh dấu độc lập nền tảng như XML.

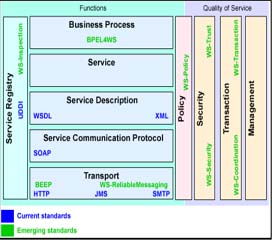
**Dynamic discovery**: là cơ chế cài đặt dựa trên Directory Service. Về phía bên nhà cung cấp, họ sẽ đăng ký những dịch vụ mình cung cấp trên Directory Service. Còn về phía bên sử dụng dịch vụ sẽ tự truy vấn tìm kiếm các dịch vụ theo nhu cầu từ Directory Service thông qua mạng. Điều này làm giảm sự lệ thuộc của bên sử dụng dịch vụ và bên nhà cung cấp dịch vụ.

**Publish**: để có thể sử dụng một dịch vụ web cụ thể nào đó thì bên nhà cung cấp cần phải công bố (mô tả) các dịch vụ của mình để bên sử dụng có thể tìm thấy nó. Việc công bố có thể khác nhau tùy thuộc vào từng ứng dụng cụ thể. Nhưng thông thường, một mô tả dịch vụ (Service description) bao gồm các thông tin sau: các giao diện, các kiểu dữ liệu, các toán tử, thông tin kết nối, vị trí dịch vụ có thể truy cập trên mạng...

**Find**: trong thao tác tìm kiếm bên sử dụng dịch vụ sẽ lấy thông tin mô tả dịch vụ thông qua bên trung gian hoặc trực tiếp với bên cung cấp dịch vụ. Thao tác tìm kiếm này có thể diễn ra trong hai vòng: thiết kế xây dựng (lập trình viên cần biết mô tả, giao diện của dịch vụ) và thực thi (xác định vị trí và tiến hành triệu gọi dịch vụ).

**Bind**: để sử dụng dịch vụ thì cần phải triệu gọi nó. Trong thao tác kết nối, bên sử dụng dịch vụ khi thực thi sẽ gọi hoặc khởi tạo một luồng tương tác với dịch vụ dựa trên các thông tin trong mô tả dịch vụ mà nó thu được trước đó: vị trí dịch vụ, các thức liên lạc và tương tác với dịch vụ …

Kiến trúc của dịch vụ web



Hình 2: Kiến trúc dịch vụ web

Kiến trúc dịch vụ web bao gồm các tầng sau:

Tầng vận chuyển với những giao thức chuẩn như: HTTP, SMTP, JMS

Tầng giao thức tương tác dịch vụ với giao thức chuẩn là SOAP, nằm giữa tầng vận chuyển và tầng mô tả thông tin dịch vụ, SOAP cho phép người dùng triệu gọi các dịch vụ từ xa thông qua thông điệp sử dụng ngôn ngữ đánh dấu XML.

Tầng mô tả dịch vụ với giao thức chuẩn như WSDL và XML. WSDL là một ngôn ngữ mô tả giao tiếp và thực thi dựa trên XML. Dịch vụ web sử dụng WSDL để truyền tải thông tin mô tả tham số, loại dữ liệu, các chức năng mà dịch vụ cung cấp.

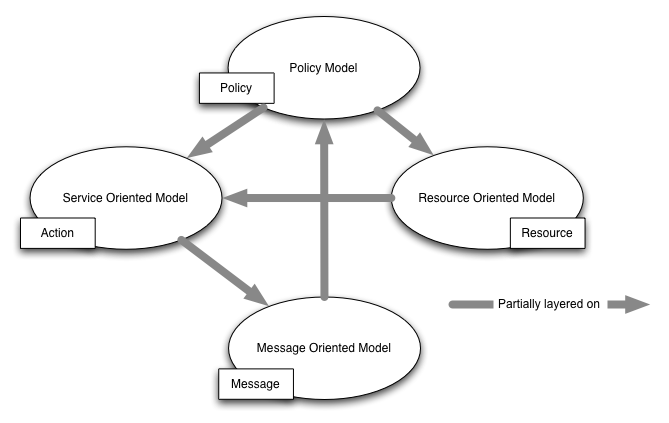
Tầng dịch vụ cung cấp các phương thức dịch vụ của hệ thống.

Tầng đăng ký dịch vụ với công nghệ chuẩn là UDDI. UDDI cho phép cả hai bên cung cấp và sử dụng dịch vụ tiến hành đăng ký và tìm kiếm các thông tin dịch vụ từ bên trung gian lưu trữ thông tin dịch vụ.

Bên cạnh đó còn có thêm các tầng chính sách, an ninh, giao tác, quản lý nhầm tăng cường tính bảo mật, an toàn và toàn vẹn thông tin khi sử dụng dịch vụ.

* + 1. Khái quát các mô hình kiến trúc dịch vụ web

Kiến trúc dịch vụ web có 4 mô hình chính sau:

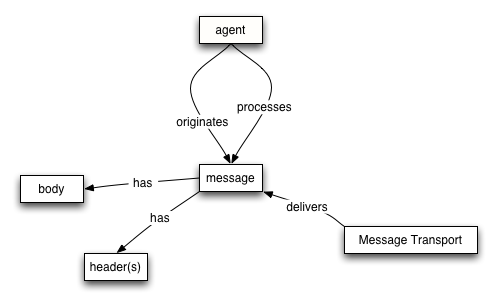


Hình 2.4: Các mô hình kiến trúc dịch vụ web.

Mỗi mô hình trong hình 2.4 được đính kèm một nhãn (hình vuông) được xem như ý niệm chính của mô hình đó.

#### Kiến trúc hướng thông điệp (Message Oriented Model)

Bản chất của mô hình chú trong vào khái niệm thông điệp, cấu trúc thông điệp, sự truyền tải thông điệp giữa các tác nhân trong hệ thống. Cấu trúc thông điệp bao gồm phần tiêu đề và phần thân là cách thức vận chuyển trong hệ thống.

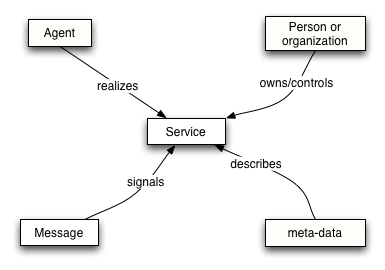


Hình 2.4: Mô hình kiến trúc hướng thông điệp.

Một thông điệp là một đơn vị dữ liệu cơ bản được vận chuyển giữa các tác nhân trong dịch vụ web.

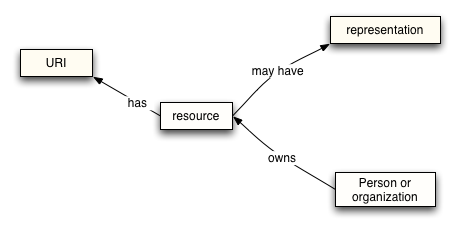
Mỗi thông điệp phải có một địa chỉ đích đến hỗ trợ việc truyền tải một cách chính xác

#### Kiến trúc hướng dịch vụ (Service Oriented Model)



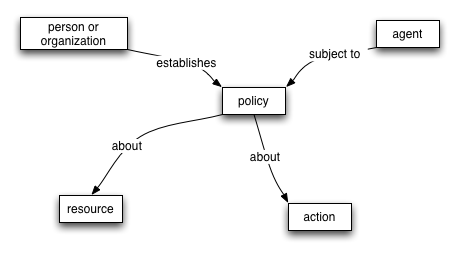
Hình 2.4: Mô hình kiến trúc hướng dịch vụ

#### Kiến trúc hướng tài nguyên (Resource Oriented Model)



Hình 2.4: Mô hình kiến trúc hướng tài nguyên

#### Mô hình chính sách (Policy Model)



Hình 2.4: Mô hình chính sách

## Xu hướng phát triển dịch vụ web

SOAP và RESTful

REST đã được chọn sử dụng rộng rãi thay cho các dịch vụ web dựa trên mô hình SOAP và WSDL. Bằng chứng quan trọng của sự thay đổi này chính là việc hang loạt các công ty dẫn đầu trong lĩnh vực cung cấp dịch vụ mạng 2.0 như Yahoo, Google và Facebook đã nâng cấp các giao thức dựa trên SOAP hoặc WSDL sang sử dụng mô hình hướng tài nguyên REST cho các dịch vụ của họ.

## REST

REST là viết tắt của Representational State Transfer. Các khái niệm của REST được giới thiệu lần đầu trong luận án tiến sĩ năm 2000 của tác giả Roy T. Fielding. Fielding đã đóng góp vào sự phát triển của HTTP 1.0 và là kiến trúc sư chính của HTTP 1.1 và là tác giả của cú pháp chung để định danh tài nguyên (URI). Mặc dù REST bản thân nó không phải là một kiến trúc mà là một tập hợp các ràng buộc mà khi áp dụng trong trong thiết kế, dẫn đến một phong cách kiến trúc phần mềm định nghĩa các quy tắc thiết kế dịch vụ web chú trọng vào các tài nguyên của hệ thống (mô hình hướng tài nguyên), bao gồm các trạng thái tài nguyên được định dạng như thế nào và được truyền tải qua HTTP, đa dạng số lượng người sử dụng và độc lập ngôn ngữ lập trình.

Những ràng buộc sau đây định nghĩa một hệ thống RESTful:

* Mọi thứ đều là tài nguyên
* Tài nguyên được định dạng thông qua một URI
* Giao diện đồng nhất
* Tương tác với tài nguyên thông qua các dạng biểu diễn của nó
* Thông điệp tự mô tả
* Tương tác phi trạng thái
* Sử dụng Hypermedia như công cụ để chuyển đổi trạng thái

## Các khái niệm chính

* + 1. Tài nguyên (Resource)

Tài nguyên là một khái niệm thông tin trừu tượng quan trọng. Bất kỳ các thông tin nào cũng có thể được xem như là một tài nguyên. Ví dụ một tài liệu, hình ảnh, đối tượng phía máy chủ. Một tài nguyên là một ánh xạ đến một tập hợp các thực thể trong một thời điểm nhất định. Một tài nguyên phải được đánh địa chỉ (addressable), có nghĩa là chúng có thể được truy cập và chuyển giao giữa máy khách và máy chủ. Mặc dù một ánh xạ tài nguyên là duy nhất, nhưng nhiều yêu cầu tài nguyên có thể chỉ đến cùng một thực thể được lưu trữ bên trong server. Ví dụ, nếu như một bài viết được đăng lần cuối vào ngày 20/7/2015 thì các yêu cầu như các bài viết vào ngày 20/7/2015 và yêu cầu các bài viết mới nhất đều chỉ đến cùng một thực thể. Theo nguyên tắc REST mỗi tài nguyên (i.e., ánh xạ) có thể được truy cập một cách trực tiếp và không không phụ thuộc, nhưng có thẻ cùng chỉ đến cùng một dữ liệu.

Tài nguyên trong REST được xác định duy nhất bởi các URI. URI là viết tắt của Universal Resource Identifier, mà cũng có thể được coi là sự kết hợp của Uniform Resource Locators (URL) và Universal Resource Names

(URN). REST yêu cầu rằng các URI của tài nguyên không nên được thay đổi theo thời gian. REST thực hiện việc này bằng cách chỉ định mỗi tài nguyên trở thành một ngữ nghĩa xác định một mục tiêu cụ thể (ánh xạ khái niệm) chứ không phải là giá trị tương ứng với những ngữ nghĩa vào thời điểm tham chiếu được tạo ra.

* + 1. Đại diện (Representation)

Một đại diện của một tài nguyên là một dữ liệu hay một nội dung được gửi tương tác giữa máy chủ và máy khách. Một đại diện được xem là một thể hiện tạm thời của đối tượng dữ liệu thực tế được lưu trữ trên máy chủ. Một đại diện bao gồm dữ liệu chính, và có thể có nhiều siêu dữ liệu (metadata) mô tả cho dữ liệu chính hay mô tả cho các siêu dữ liệu khác.

Do sự đa dạng của các kiểu máy khách cùng với khả năng xử lý nội dung, mà đại diện của một tài nguyên có thể nhận được với nhiều hình thức định dạng như HTML, hình ảnh, tập tin, văn bản, XML và JSON. Định dạng dữ liệu của một tài nguyên được biết đến như một loại phương tiện truyền thông, thường được ký hiệu MIME. Căn cứ vào yêu cầu của máy khách, định dạng mong muốn mà tài nguyên sẽ có đại diện phù hợp được chọn một cách tự động (HTML, XML, JSON,…). Nhưng tất cả các đại diện khác nhau của một tài nguyên phải được truy cập thông qua cùng một định danh tài nguyên URI.

* + 1. Tương tác phi trạng thái (Stateless)

Theo các nguyên tắc REST, tất cả các giao tiếp giữa máy khách và máy chủ phải là giao tiếp phi trạng thái. Trong đó hàm ý rằng tất cả các yêu cầu từ máy khách đến máy chủ cần phải chứa tất cả các thông tin cần thiết để xác định và xử lý yêu cầu. Không có thông tin trạng thái nào được giữ ở máy chủ cũng như các thông tin trạng thái của các yêu cầu trước đó. Như vậy, thông tin trạng thái phiên tương tác (session state) được giữ hoàn toàn trên máy khách.

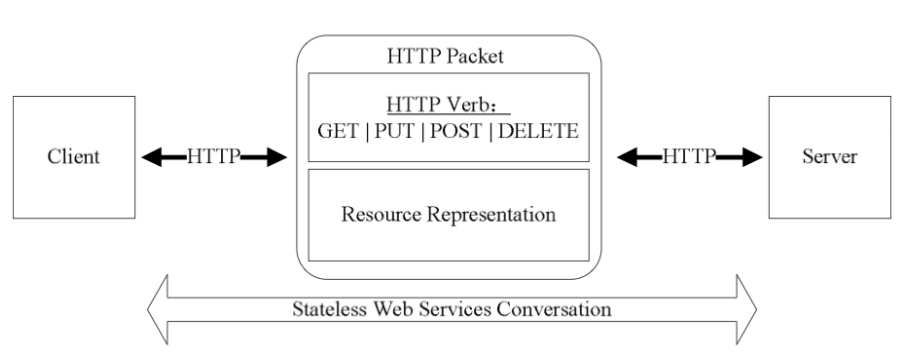
Điều quan trọng là phải lưu ý rằng trạng thái phiên hay trạng thái ứng dụng khác đối với trạng thái tài nguyên. Trạng thái tài nguyên chứa thông tin về tài nguyên và được lưu giữ và duy trì bởi máy chủ.

* + 1. Giao diện đồng nhất (Uniform interface)

Trong các ứng dụng dựa trên nền web hiện đại, các hoạt động cơ bản có thể được phân loại thành bốn loại sau đây: Create, Read, Update và Delete (CRUD). Ứng dụng RESTful cho phép thao tác tới trạng thái của tài nguyên thông qua qua các hoạt động chủ yếu như tạo tài nguyên, sửa đổi, cập nhật và xóa. Mặc dù dịch vụ RESTful có thể cho phép thực thi logic ở cấp độ máy chủ, các hoạt động sẽ là một trong bốn dạng hoạt động cơ bản ở trên.

Mặc dù HTTP hỗ trợ vô số các động từ (hay các phương thức thực thi HTTP) như GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE, CONNECT và PATCH, chỉ có một bộ phận các động từ được sử dụng đủ để đáp ứng các yêu cầu của nhiều ứng dụng phân tán. Vì vậy, xem xét REST cùng các khái niệm của nó như tài nguyên, đại diện, các hoạt động CRUD có thể được ánh xạ đầy đủ đối với các phương thức HTTP như PUT, GET, POST và DELETE.

HTTP GET được sử đụng để lấy được đại diện của tài nguyên, HTTP DELETE được sử dụng để loại bỏ một tài nguyên ra khỏi hệ thống, HTTP PUT được sử dụng để tạo ra một tài nguyên và HTTP POST được sử dụng để cập nhật dữ liệu của một tài nguyên. Các tài nguyên được xác đinh bởi URI của nó và hoạt động trên các tài nguyên được biểu diễn bằng các phương thức mà HTTP được sử dụng.



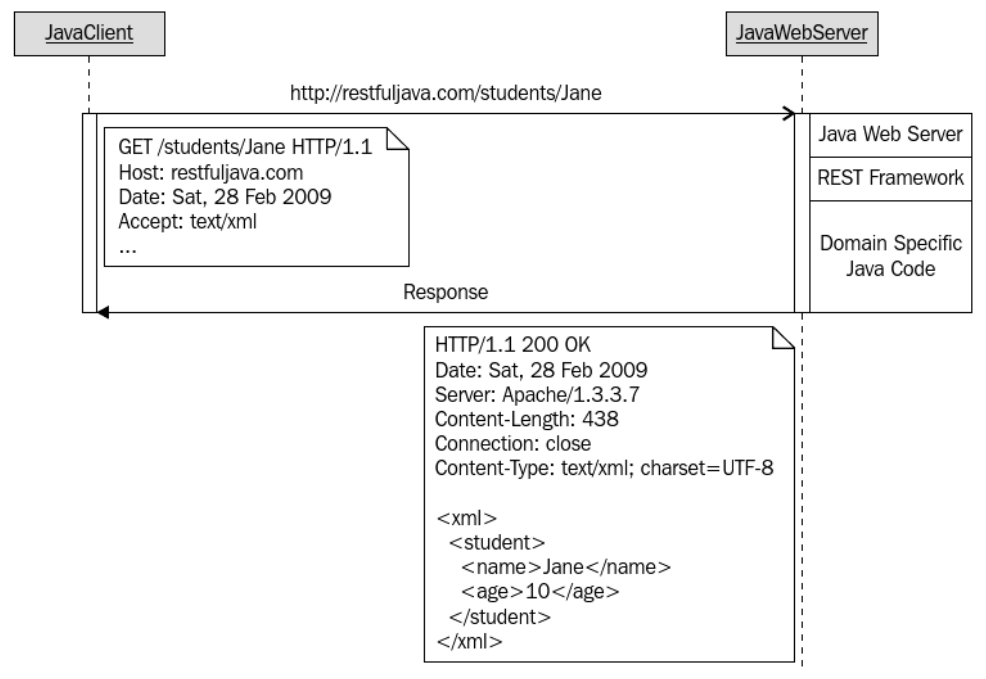
Hình 1: Kiến trúc dịch vụ web RESTful

|  |  |
| --- | --- |
| Hoạt động | Giao thức tương ứng HTTP |
| CREATE | PUT |
| RETRIEVE | GET |
| UPDATE | POST |
| DELETE | DELETE |

Hình 1:Hoạt động CRUD tương ứng đối với HTTP verbs

* + 1. Hoạt động GET/RETRIEVE

HTTP GET được sử dụng để truy xuất một tài nguyên hệ thống. Đây là hoạt động duy nhất mà không cho phép tạo sự thay đổi đến tài nguyên. Hình sau mô tả hoạt động của nó:

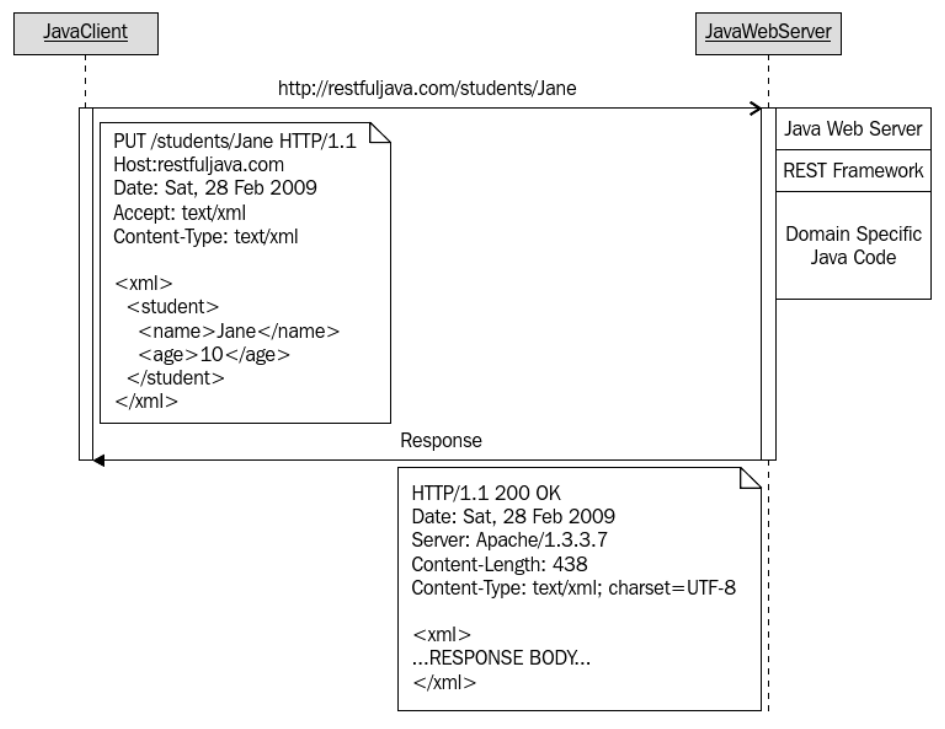


Hình 1:Ví dụ hoạt động GET/RETRIEVE trên hệ thống

Trong ví dụ trên, trình khách Java gửi một yêu cầu HTTP GET cho một tài nguyên có định danh là Jane. Trình khách cũng thông báo với máy chủ rằng định dạng mong muốn của mình là XML thông qua trường Accept trong header của lời yêu cầu. Máy chủ xử lý yêu cầu và sau đó gửi phản hồi chứa định dạng yêu cầu. Như ví dụ trên thực thể Jane hoàn toàn được biểu diễn bởi định dạng XML.

* + 1. Hoạt động PUT/CREATE

HTTP PUT được sử dụng để tạo một tài nguyên. Hình sau mô tả hoạt động của nó:

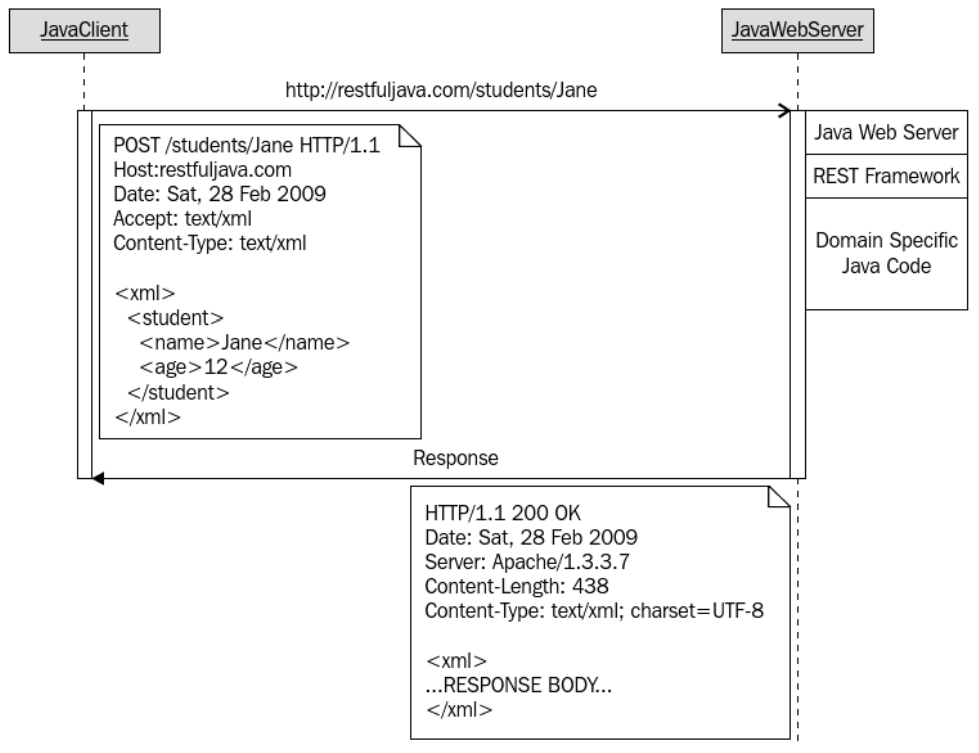


Hình 1:Ví dụ hoạt động PUT/CREATE trên hệ thống

Ở đây, trình khách Java muốn tạo một thực thể Jane trên máy chủ. Trình khách sẽ gửi một yêu cầu HTTP PUT đến máy chủ với nội dung đính kèm là thực thể được biểu diễn dưới định dạng XML. Sau khi nhận được yêu cầu từ trình khách, máy chủ xác định hoạt động mong muốn từ phương thức HTTP và xác định định dạng nội dung đính kèm thông qua trường Content-Type trong header của yêu cầu. Sau đó, máy chủ tiến hành chuyển đổi ngược trở lại nội dung thực thể và gọi chức năng bên dưới để tiến hành tạo đối tượng. Một thông báo phản hồi được gởi lại cho trình khách với mục đích thông báo trạng thái thành công hoặc thất bại của hoạt động.

* + 1. Hoạt động POST/UPDATE

HTTP POST được sử đụng để cập nhật tài nguyên trên hệ thống. Hình sau minh họa hoạt động của nó:

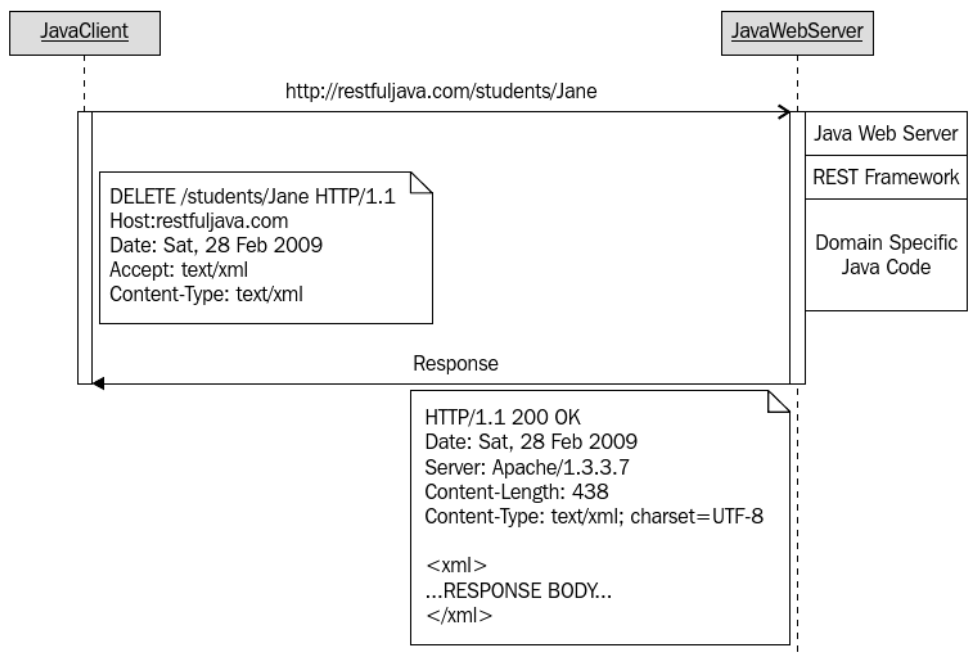


Hình 1:Ví dụ hoạt động POST/UPDATE trên hệ thống

Sau khi trình khách đã chỉnh sửa nội dung trong thực thể Jane được lấy thông qua lời yêu cầu GET được thực hiện lúc trước. Nội dung của thực thể Jane sau khi thay đổi được chuyển đổi sang định dạng mong muốn và đính kèm vào lời yêu cầu POST. Trình khách tiến hành gửi yêu cầu POST để cập nhật sự thay đổi trong thực thể Jane lên hệ thống. Máy chủ sau khi nhận được yêu cầu từ trình khách, tiến hành rút trích và chuyển đổi trở lại nội dung thay đổi và tiến hành cập nhật trên máy chủ. Một thông điệp phản hồi được gởi lại cho trình khách nhằm thông báo trạng thái thành công hay thất bại của hoạt động trên.

* + 1. Hoạt động DELETE/DELETE

HTTP DELETE được sử dụng để xóa tài nguyên trên hệ thống. Tài nguyên được định danh thông qua URI. Hình sau minh họa hoạt động của nó:

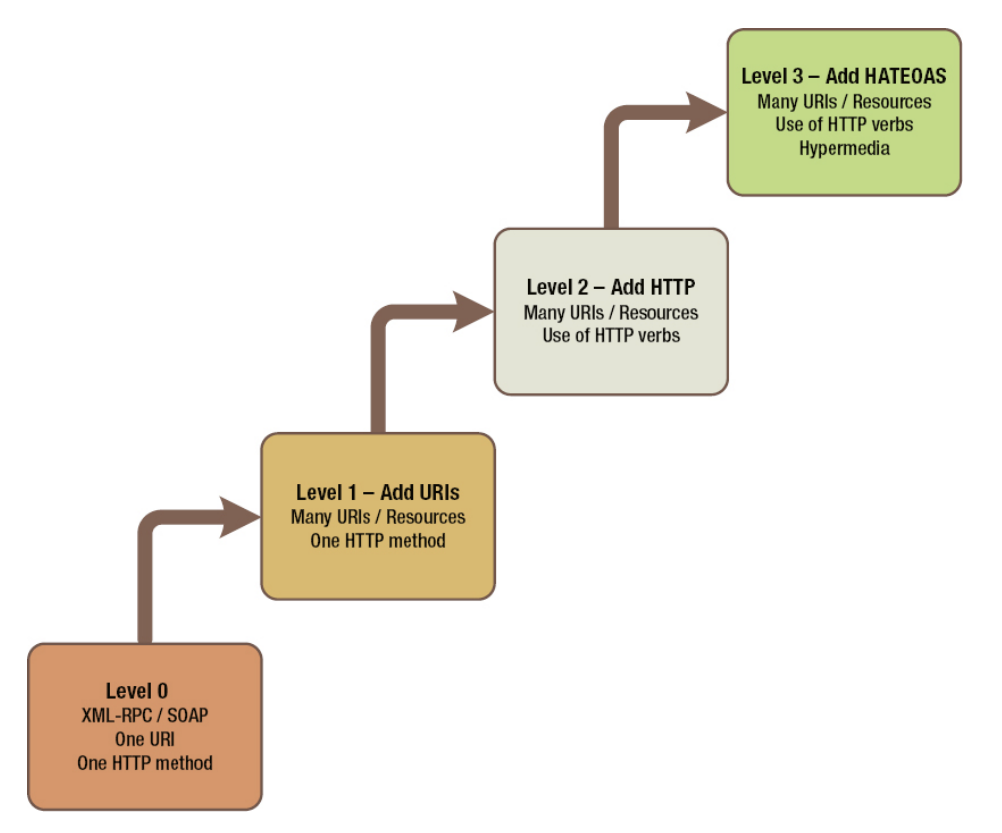


Hình 1:Ví dụ hoạt động DELETE/DELETE trên hệ thống

Trình khách Java gửi lời yêu cầu DELETE đến máy chủ nhầm xóa đối tượng tài nguyên được định danh là Jane. Máy chủ tiến hành xử lý yêu cầu và xác định hoạt động mong muốn thông qua phương thức HTTP sử dụng. Định danh tài nguyên thông qua URI của yêu cầu. Thông điệp phản hồi được gửi lại cho trình khách thông báo trạng thái thành công hay thất bại của yêu cầu.

* + 1. Mô hình trưởng thành của Richardson (RMM)

Leonard Richardson giới thiệu mô hình trưởng thành trong REST vào năm 2008. Nó cung cấp ba mức độ tuân thủ REST (Cấp 1-3) và một mức 0 cho các giao diện theo phong cách RPC. Các mức RMM có thể tóm tắt ở hình sau:



Hình 1: Mô hình trưởn thành của Richardson trong REST.

#### RMM Level 0

Cấp 0 được đặc trưng bởi các dịch vụ có một URI duy nhất và việc sử dụng các phương pháp HTTP đơn lẻ (thường POST). Điều này thường dựa trên phong cách RPC và HTTP chỉ được sử dụng như một cơ chế vận chuyển hay cơ chế đường đường hầm (tunneling) mà không sử dụng bất kỳ các cơ chế của web. Cũng có thể xem như là Plain Old XML (POX) tương tự như SOAP chỉ khác là không có SOAP envelop.

#### RMM Level 1

Cấp 1 mỗi tài nguyên đã có nhiều URIs tham chiếu nhưng chỉ có một phương thức HTTP được sử dụng. Vì thế, sự khác biệt chính là ở các dịch vụ cấp độ 1 đã đưa ra nhiều tài nguyên logic hợp lý hơn so với ở cấp độ 0 tất cả các tương tác đều thông qua một tài nguyên duy nhất phức tạp. Các hoạt động ở cấp độ 1 được thực hiện bằng cách chèn tên hoạt động cùng các thông số khác nhau vào một URI và kêu gọi các dịch vụ từ xa bằng cách sử dụng chỉ với một phương thức HTTP (thường thông qua HTTP GET).

#### RMM Level 2

Tại cấp độ 2, đã có nhiều nguồn tài nguyên URIs và dịch vụ đã hỗ trợ một số phương thức của HTTP. Hoạt động CRUD đã được hỗ trợ bằng cách sử dụng các phương thức của HTTP như POST, GET, PUT và DELETE. Các dịch vụ sử dụng các phương thức từ HTTP và mã trạng thái để phối hợp tương tác và do đó, nỗ lực ở cấp độ 2 đã đạt được độ vững mạnh (robustness) của Web.

#### RMM Level 3

Cấp độ 3 triển khai khái niệm Hypermedia As The Engine Of

Application State (HATEAOS). Trong HATEAOS, một đại diện của tài nguyên chứa các liên kết URI đến các tài nguyên có liên quan khác, và có thể có giá trị đối với người sử dụng. Cũng có thể được xem như các trạng thái khác của tài nguyên. Có thể là cung cấp liên kết đến các trạng thái kế tiếp từ trạng thái hiện hành của tài nguyên. Do đó, dịch vụ dẫn dắt người dùng thông qua một chuỗi các tài nguyên, dẫn đến kết quả tạo ra sự chuyển đổi trạng thái của ứng dụng.

Lợi ích chính của HATEOAS là nó giúp cho máy chủ và trình khách không còn phụ thuộc nhiều vào nhau. Máy chủ có thể dễ dàng thay đổi các sơ đồ URI trong dịch vụ mà không ảnh hưởng đến các triển khai của máy khách. Máy khách sẽ tra cứu các chuyển đổi trạng thái từ dịch vụ nên các mã nguồn ban đầu của máy khách sẽ không phải thay đổi.

Hình sau minh họa một thực thể Order đi kèm các liên kết trạng thái điều khiển có liên quan.



Hình 1:Ví dụ HATEOAS trong thực thể Order.

Các phần tử liên kết đại diện cho các trạng thái kế tiếp khả thi hoặc có liên quan đến thực thể Order như các trạng thái payment, cancel, update hay self liên kết chính bản thân thực thể. Các liên kết được tạo ra và cập nhật bởi máy chủ. Do đó mà máy chủ có thể thay đổi sơ đồ địa chỉ hay các trạng thái của ứng dụng bất cứ lúc nào mà không ảnh hưởng đến triển khai của khách hàng. Các thuộc tính URI trong các phần tử liên kết chứa các URI của tài nguyên mà máy khách có thể tương tác để tiến tới trạng thái kế tiếp. Các thuộc tính rel sẽ chứa sẽ chứa các đánh dấu ngữ nghĩa. Ngoài ra các phần tử liên kết còn có thể chứa thuộc tính MediaType bất cứ khi nào các yêu cầu đòi hỏi nội dung cụ thể. Thuộc tính MediaType sẽ khai báo định dạng của nội dung trong một yêu cầu thường tương ứng với một loại MIME.

Cấp độ 3 đã cho thấy sự tuân thủ đầy đủ các nguyên tắc trong REST. Tuy nhiên, vẫn là rất khó khan để đạt được điều đó, do bởi là một công nghệ mới nổi, chưa có một tiêu chuẩn nào là tuyệt đối được áp dụng vào cách triển khai các thông tin điều khiển hypermedia. Ngoài ra còn thiếu các công cụ hỗ trợ như các các nền tảng phát triển web hiện đại đã làm thì chưa có một nguyên tắc HATEAOS nào trong các giao diện API của họ. Hầu hết các hiện thực của HATEAOS được tìm thấy là các nghiên cứu hay phát triển nguyên mẫu mà đang cần được sử dụng trong công nghiệp và phê duyệt.

## Mục tiêu chính

* Khả năng mở rộng của các thành phần tương tác.
* Giao diện tổng quát.
* Triển khai độc lập của các thành phần.
* Sử dụng thành phần trung gian để giảm độ trễ, thực thi an ninh và đóng gói các hệ thống kế thừa.

## Ràng buộc

* **Client-Server** – Một giao diện đồng nhất tách biệt các client ra khỏi các server. Sự tách biệt giúp cho các mã client và server liên kết linh động hơn thông qua các giao diện đồng nhất. Ví dụ như, những đoạn mã client sẽ không cần quan tâm đến việc lưu trữ dữ liệu thật sự bên trong hoạt động của server, từ đó tính di động của mã client được cải thiện. Hoặc mã server sẽ không quan tâm đến giao diện người dùng cụ thể cùng với trạng thái người dùng sẽ giúp cho các máy chủ dễ dàng mở rộng và nâng cấp hơn. Máy chủ và trình khách có thể được thay thế và phát triển một cách độc lập miễn là giao diện giữa chúng là không thay đổi.
* **Phi trạng thái (Stateless)** – Các giao tiếp giữa trình khách và máy chủ không còn bị phụ thuộc vào trạng thái người dùng (session state) được lưu trữ trên máy chủ giữa các yêu cầu. Từ đây mỗi yêu cầu từ phía trình khách bất kỳ đã có chứa đầy đủ các thông tin cần thiết để máy chủ có thể nhận dạng và phục vụ được yêu cầu đó mà không cần phải lưu trữ bất cứ thông tin nào của người dùng và thông tin trạng thái này sẽ được lưu trữ và duy trì ở trình khách. Máy chủ giờ đây có thể dễ dàng mở rộng theo chiều ngang (scale out) khi tuân thủ tốt ràng buộc phi trạng thái này.

**Khả năng lưu trữ tạm thời (Cacheable) –** trong kiến trúc web, máy khách có thể lưu trữ tạm thời (cache) các kết quả trả về từ máy chủ với mục đích giảm số lượng yêu cầu đến máy chủ, thời gan chờ, tải hệ thống cũng như băng thông … Do đó một kết quả trả về phải ngầm định hay chỉ rõ bản thân có thể cache hoặc không và thời gian hợp lệ hiệu quả để ngăn chặn việc client tái sử dụng dữ liệu cũ hoặc không thích hợp để đáp ứng các yêu cầu. Quản lý tốt yêu cầu lưu trữ tạm thời sẽ giúp cho hệ thống giảm thiểu bớt số lượng tương tác giữa client và server từ đó tiếp tục nâng cao khả năng mở rộng cũng như hiệu suất của hệ thống.

**Hệ thống đa tầng (Layered system) –** Trong một hệ thống các thành phần chức năng được nhóm lại và chia thành nhiều tầng chức năng phân cấp, trong đó các tầng thấp hơn sẽ cung cấp chức năng và dịch vụ cho các tầng cao hơn. Nhờ đó hệ thống có thể dễ phát triển hoặc nâng cấp sửa đổi bằng cách thêm các tầng trung gian hoặc mở rộng thêm thông qua các cân bằng tải (load balancer).

**Tùy chọn (Code on demand) –** Máy chủ có thể tạm thời mở rộng hay tùy chỉnh chức năng của trình khách bằng cách chuyển giao các mã thực thi cho trình khách. Ví dụ này có thể là các thành phần biên dịch như Java applets và mã khách như Javascipt có thể mở rộng chức năng bằng cách chuyển giao các đoạn mã javascipt mở rộng cho trình khách. Đây là một ràng buộc tùy chọn trong kiến trúc REST.

**Giao diện thống nhất (Unifom interface) –** Các giao diện thống nhất giữa máy khách và máy chủ cho phép đơn giản hóa giao tiếp và tách biệt hiệu quả cho kiến trúc, nó cho phép các thành phần tiến hóa một cách độc lập. Có bốn nguyên tắc cơ bản sau:

**Định danh các tài nguyên –** Các tài nguyên của hệ thống được định danh cụ thể trong các yêu cầu, ví dụ như việc sử dụng URIs trong kiến trúc web để định danh một tài nguyên trong REST. Bản thân một tài nguyên được tách biệt khỏi kiểu biểu diễn được trả về cho client. Ví dụ một server có thể trả về dữ liệu chứa trong cơ sở dữ liệu dưới dạng HTML, XML hoặc JSON, không một định dạng cụ thể nào ở trên là định dạng lưu trữ tài nguyên bên trong server.

**Thao tác tài nguyên hệ thống thông qua các biểu diễn của nó –** Khi một client nắm giữ một biểu diễn của một tài nguyên, bao gồm bất kỳ metadata đi kèm theo, nó có đủ thông tin để sửa đổi hoặc xóa tài nguyên đó.

**Thông điệp tự mô tả -** Mỗi thông điệp truyền tải trong hệ thống bao gồm đầy đủ thông tin để mô tả cách thức xử lý thông điệp đó. Chẳng hạn, chỉ rõ trình phân tích nào sẽ được sử dụng để xử lý thông điệp bằng cách chỉ định loại phương tiện truyền thông internet ( trước đây được biết đến như là một loại MIME).

**Sử dụng Hypermedia như phương tiện chuyển đổi trạng thái ứng dụng (HATEOAS) –** Các tài nguyên của hệ thống được định danh cụ thể trong các yêu cầu, ví dụ như việc sử dụng URIs trong kiến trúc web để định danh một tài nguyên trong REST. Bản thân một tài nguyên được tách biệt khỏi kiểu biểu diễn được trả về cho client. Ví dụ một server có thể trả về dữ liệu chứa trong cơ sở dữ liệu dưới dạng HTML, XML hoặc JSON, không một định dạng cụ thể nào ở trên là định dạng lưu trữ tài nguyên bên trong server.

Việc tuân thủ đầy đủ các ràng buộc được mô tả bên trên được xem là RESTful. Nếu như một hệ thống nào vi phạm hay không tuân thủ bất cứ ràng buộc nào thì hệ thống đó không được xem là RESTful. Một hệ thống phân tán tuân theo RESTful sẽ có những đặc tính nổi bật như hiệu suất (performance), khả năng mở rộng (scalability), đơn giản (simplicity), dễ thay đổi (modifiablility), tầm nhìn (visibility), tính di động (portability) và độ tin cậy (reliability).

## Cơ chế hoạt động

* Kiến trúc kiểu REST thường bao gồm các máy khách (client) và máy chủ (server)
* Client gửi yêu cầu đến server.
* Server xử lý yêu cầu (request) và gửi trả kết quả (respone) thích hợp.
* Requests và responses được xây dựng xung quanh việc chuyển giao thông qau đại diện của các nguồn tài nguyên.
* Một tài nguyên được gắn kết với một vài nội dung có ý nghĩa mà có thể được gắn với một địa chỉ.
* Một đại diện cho một tài nguyên thường là một tài liệu mà nắm bắt được tình trạng hiện tại hoặc trạng thái tiếp theo của tài nguyên.
* Client bắt đầu gửi một yêu cầu khi nó đã sẵn sang để chuyển sang một trạng thái mới (trong khi một hoặc nhiều yêu cầu đang được giải quyết, client được xem là đang trong quá trình chuyển đổi)
* Đại diện của mỗi trạng thái trong ứng dụng có thể chứa liên kết (links) mà có thể được client chọn và bắt đầu một trạng thái giao dịch mới.

## RESTful API

## Bảo mật REST API

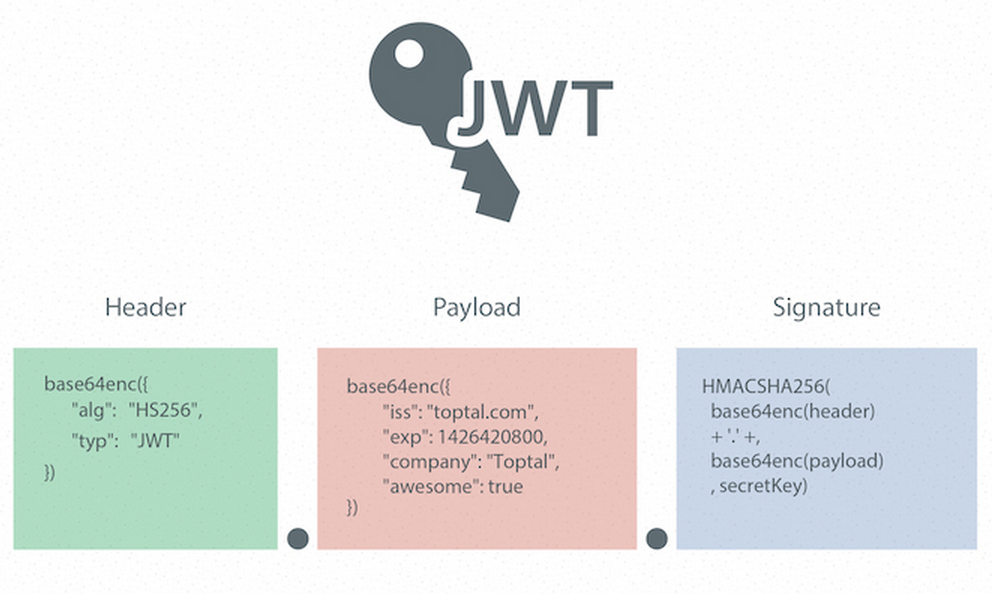
* + 1. JSON Web Token

JSON Web Token hay viết tắt là JWT, được sử dụng để gửi thông tin, và có thể được xác nhận và tin cậy bằng phương pháp ký số. Nó bao gồm một đối tượng JSON nhỏ gọn và URL-safe, được ký số nhầm bảo toàn tính toàn vẹn của thông tin xác thực người dùng, và nó cũng có thể được mã hóa nếu nội dung chứa các thông tin nhạy cảm.

Bởi vì JWT có kích thước nhỏ gọn, JWT thường được sử dụng trong trường Authorization trong HTTP header hoặc đi kèm trên tham số truy vấn URL.

* + 1. Cấu trúc của JSON Web Token

Một JWT được biểu diễn như là một chuỗi các giá trị được mã hóa bởi base64url và các giá trị được phân cách nhau bởi dấu chấm.



Hình 2.4: Cấu trúc của một JSON Web Token

Ví dụ của một JWT:

|  |
| --- |
| eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  eyJpc3MiOiJ0b3B0YWwuY29tIiwiZXhwIjoxNDI2NDIwODAwLCJodHRwOi8vdG9wdGFsLmNvbS9qd3RfY2xhaW1zL2lzX2FkbWluIjp0cnVlLCJjb21wYW55IjoiVG9wdGFsIiwiYXdlc29tZSI6dHJ1ZX0.  yRQYnWzskCZUxPwaQupWkiUzKELZ49eM7oWxAQK\_ZXw |

#### Nội dung (Payload / Claims)

Các tiêu đề chứa các siêu dữ liệu cho các token và tối thiểu có chưa hai loại thông tin loại chữ ký và thuật toán mã hóa

Ví dụ:

|  |
| --- |
| { “alg”: “HS256”, “typ”: “JWT”} |

Tiêu đề trên đây khai báo rằng đối tượng được mã hóa là một JSON Web Token và nó được ký bằng thuật toán HMAC SHA-256.

Và khi được mã hóa bằng base64 encoded, chúng ta có phần đầu tiên của một JWT như sau:

|  |
| --- |
| eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 |

#### Nội dung (Payload / Claims)

Trong bối cảnh của một JWT, một claim có thể được định nghĩa như một khai báo thực thể (thông thường ám chỉ là thông tin người dùng), cũng như các thông tin mở rộng về chính token đó. Trong claim chứa các thông tin mà chúng ta mong muốn truyền tải, và máy chủ có thể sử dụng các thông tin đó xác thực người dùng. Có nhiều loại claim chúng ta có thể cung cấp trên token, có thể claim đã đăng ký, claim công khai cũng như claim riêng tư.

Tên claim đăng ký (Registered Claims)

Sau đây là những tên claim đã được đăng ký tại IANA JSON Web Token Claims registry. Những tên claim này không có tính bắt buộc mà là cung cấp một điểm khởi đầu cho một tập các tên claim hữu ích, mang tính tương thích.

Bao gồm:

* **iss**: Tổ chức phát hành của token
* **sub**: Đối tượng của mã token
* **aud**: Các đối tượng lắng nghe của mã token
* **exp**: Thời gian hết hạn của mã token được định nghĩa trong Unix time
* **nbf**: “Not before” thời gian xác định thời điểm mà mã token không được chấp nhận xử lý nếu như thời gian hiện tại là trước thời điểm đó.
* **iat**: “Issued at” thời điểm mã token được ban hành
* **jti**: “JWT ID” định danh duy nhất cho mã token
* Tên claim công khai
* Các tên claim công khai cần phải có những cái tên tránh trùng lắp. Bằng cách làm đặt tên là một URI hay URN thì việc trùng lắp được giảm thiểu và khi người gửi và người nhận không thuộc một mạng lưới khép kín.
* Một claim có thể là: <http://toptal.com/jwt_claims/is_admin> và một điều khuyến khích là đặt một tập tin tại vị trí đó mô tả chi tiết claim để nó có thể được tài liệu hóa.
* Tên claim riêng tư
* Tên claim riêng tư có thể được sử dụng ở những nơi mà JWTs chỉ được trao đổi bên trong một môi trường khép kín giữa các hệ thống doanh nghiệp. Đây là một claim rằng chúng ta có thể xác định nhân danh của một đối tượng cụ thể, như ID người dùng, vai trò của người dùng hoặc bất kỳ thông tin khác.
* Sử dụng các tên claim riêng tư có thể có ý nghĩa ngữ nghĩa mâu thuẫn trùng lắp với các tên claim bên ngoài hệ thống khép kín hay riêng tư. Do đó sử dụng các tên claim riêng tư một cách thận trọng.
* Một điều quan trọng cần lưu ý chúng ta nên giữ kích thước của một mã token càng nhỏ càng tốt. Do đó, chỉ sử dụng các dữ liệu cần thiết bên trong token.

Ví dụ một payload:

|  |
| --- |
| {  “iss”: “toptal.com”,  “exp”: 1426420800,  “http://toptal.com/jwt\_claims/is\_admin”: true,  “company”: “Toptal”,  “awesome”: true  } |

Ví dụ payload trên có hai tên claim đăng ký (iss, exp), tên claim công khai (http://toptal.com/jwt\_claims/is\_admin), tên claim riêng tư (company, awesome). Sau khi mã hóa base64, chúng ta có phần thứ hai của JWT.

|  |
| --- |
| eyJpc3MiOiJ0b3B0YWwuY29tIiwiZXhwIjoxNDI2NDIwODAwLCJodHRwOi8vdG9wdGFsLmNvbS9qd3RfY2xhaW1zL2lzX2FkbWluIjp0cnVlLCJjb21wYW55IjoiVG9wdGFsIiwiYXdlc29tZSI6dHJ1ZX0 |

#### Chữ ký (Signature)

* JWT tuân thủ theo tiêu chuẩn chữ ký số JSON Web Signature (JWS) để tạo ra mã thông báo đã được ký số. Nó được tạo ra bằng cách kết hợp phần mã hóa JWT tiêu đề và phần mã hóa JWT payload và chúng được mã hóa bằng một thuật toán vững chắc như HMAC SHA-256 để tạo chữ ký số. Khóa bí mật của chữ ký này sẽ được lưu giữ ở máy chủ, nó được dùng để xác minh các chữ ký của mã thông báo hoặc ký mới.

|  |
| --- |
| $encodedContent = base64UrlEncode(header) + “.” + base64UrlEncode(payload);  $signature = hashHmacSHA256($encodedContent); |

Và kết quả chữ ký số cuối cùng của mã token:

|  |
| --- |
| yRQYnWzskCZUxPwaQupWkiUzKELZ49eM7oWxAQK\_ZXw |

* + 1. Bảo mật và mã hóa với JWT

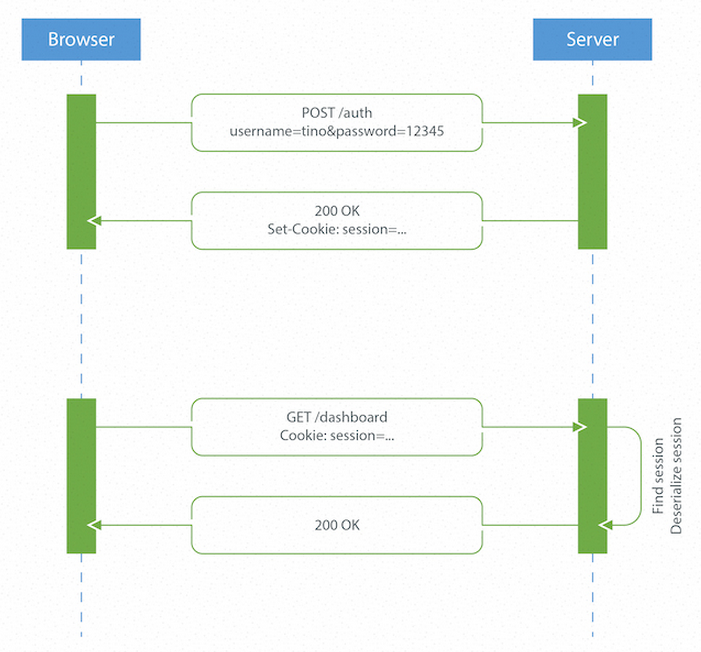
Chúng ta có thể sử dụng TLS/SSL kết hợp với JWT, để ngăn chặn các nguy cơ tấn công man-in-the-middle. Trong hầu hết trường hợp, sẽ là đủ khi chúng ta mã hóa phần nội dung JWT nếu có chứa thông tin nhạy cảm. Tuy nhiên, nếu chúng ta muốn thêm một lớp bảo mật bổ sung, chúng ta có thể mã hóa nội dung JWT bằng cách sử dụng đặc tả kỹ thuật JSON Web Encryption (JWE).

Tất nhiên, nếu chúng ta muốn tránh những chi phí phát sinh khi sử dụng JWE, một lựa chọn khác là chỉ cần giữ các thông tin nhạy cảm trong cơ sở dữ liệu, và sử dụng mã token gọi các API để truy cập các thông tin nhạy cảm.

* + 1. Tại sao sử dụng web token

Trước khi chúng ta có thể nhìn thấy tất cả những lợi ích của việc sử dụng xác thực token, chúng ta nên xem xét lại cách thức xác thực truyền thống.

Xác thực dựa trên máy chủ



Hình 2.4: Mô hình xác thực truyền thống

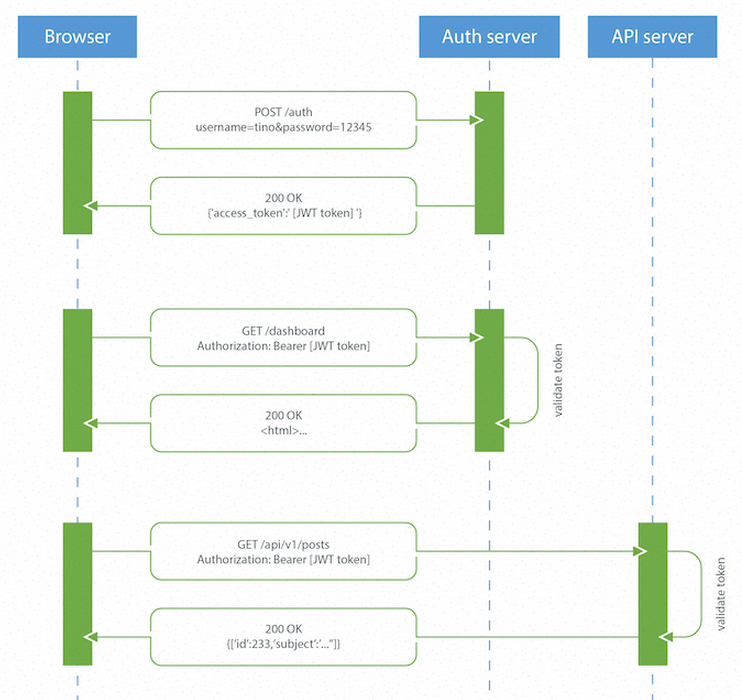
Giao thức HTTP là giao thức phi trạng thái, do đó cần phải có một cơ chế lưu trữ thông tin người dùng và cách thức để xác thực người dùng trên tất cả các lời yêu cầu từ máy khách. Hầu hết các trang web hiện nay sử dụng cookie để lưu trữ mã nhận dạng phiên làm việc (session id) của người dùng và thông tin session sẽ được lưu ở phía máy chủ.

Cơ chế hoạt động: trình duyệt thực hiện gửi một yêu cầu xác thực POST đến máy chủ chứa thông tin nhận dạng và mật khẩu người dùng. Các máy chủ thực hiện xác thực các thông tin người dùng, nếu hợp lệ máy chú sẽ lưu thông tin phiên làm việc và sẽ phản hồi trở một mẫu cookie chứa mã phiên làm việc hiện hành. Sau đó, trên mỗi lời yêu cầu tiếp theo từ máy khách, máy chủ tiến hành tìm kiếm phiên làm việc trước đó thông qua mã phiên được lấy từ tiêu đề của lời yêu cầu.

Những hạn chế của xác thực người dùng dựa trên máy chủ

* Khó mở rộng: Máy chủ cần phải tạo ra một phiên làm việc cho mỗi người dùng và duy trì nó ở đâu đó ở máy chủ. Điều này có thể được thực hiện trong bộ nhớ hoặc một cơ sở dữ liệu. Nếu chúng ta có một hệ thống phân tán, chúng ta cần phải bảo đảm rằng chúng ta sử dụng một hệ thống lưu trữ phiên làm việc riêng biệt tách biệt với máy chủ ứng dụng.
* Chia sẻ yêu cầu đa nền tảng (Cross-origin request sharing): Khi sử dụng các lời gọi AJAX để lấy thông tin tài nguyên từ một miền khác, chúng ta có thể gặp phải vấn đề các yêu cầu bị ngăn chặn, cấm thực thi, do bởi chính xách mặc định, các yêu cầu HTTP đến các miền khác không chứa sẵn các thông tin cookie, và chính sách thực thi của trình duyệt ngăn chặn các lời gọi đến các miền khác.
* Phụ thuộc chặt chẽ với các khung phát triển web: Khi sử dụng cơ chế xác thực dựa trên máy chủ, chúng ta bị phụ thuộc chặt chẽ tới cơ chế xác thực của nền tảng đó. Sẽ là điều khó khăn, có khi là không thể thực hiện được khi tiến hành chia sẻ dữ liệu phiên giữa các khung phát triển web khác nhau và được viết bởi các ngôn ngữ lập trình khác nhau.

Xác thực dựa trên token



Hình 2.4: Mô hình xác thực dựa trên mã token

Xác thực dựa trên mã token là phi trạng thái, vì thế không cần thiết phải lưu trữ thông tin người dùng trong các phiên làm việc. Điều này mang lại cho chúng ta khả năng mở rộng ứng dụng mà không cần phải quan tâm đến vấn đề nơi nào người dùng đã thực hiện đăng nhập. Chúng ta có thể dễ dàng sử dụng mã token để lấy các tài nguyên an toàn trên các máy chủ khác với máy chủ chúng ta thực hiện đăng nhập ban đầu.

Cơ chế hoạt động: một trình khách browser hay một trình khách di động gửi một lời yêu cầu đến máy chủ xác thực chứa thông tin người dùng đăng nhập. Máy chủ xác thực tiến hành sinh một mã token JWT và gửi trả cho trình khách. Trên mỗi lời yêu cầu đến các tài nguyên được bảo vệ, trình khách phải đính kèm mã token sinh ra bởi máy chủ lên chuỗi truy vấn hoặc tiêu đề Authorization của lời yêu cầu đó. Các máy chủ sau đó tiến hành xác nhận mã token. Nếu mã token là hợp lệ, máy chủ thực hiện xử lý và trả lại kết quả tài nguyên được yêu cầu.

Máy chủ xác thực có thể ký bất kỳ mã token nào sử dụng các phương thức mã hóa an toàn. Ví dụ, một thuật toán khóa đối xứng như HMAC SHA-256 có thể được sử dụng nếu thông tin khóa bí mật có thể được chia sẻ giữa các bên trên một kênh an toàn. Hay một thuật toán khóa bất đối xứng, như RSA, có thể được sử dụng, loại bỏ sự cần thiết phải chia sẻ khóa bí mật.

Những lợi ích của cơ chế xác thực dựa trên mã token:

* + Phi trạng thái và dễ dàng mở rộng: Mã token có chứa tất cả các thông tin để xác thực người dùng, loại bỏ sự cần thiết của việc duy trì trạng thái phiên. Nếu chúng ta sử dụng cân bằng tải, yêu cầu từ phía người dùng có thể đến bất kỳ máy chủ nào, thay vì phải ràng buộc đến cùng một máy chủ mà người dùng đã tiến hành đăng nhập ban đầu.
  + Khả năng tái sử dụng: Chúng ta có thể tái sử dụng mã token để xác thực người dùng trên nhiều máy chủ riêng biệt, chạy trên nhiều nền tảng và miền khác nhau. Điều này giúp chúng ta dễ dàng xây dựng một ứng dụng mà quyền truy cập có thể được chia sẻ với ứng dụng khác.
  + An toàn: Vì chúng ta không sử dụng cookie để xác thực, chúng ta không cần phải bảo vệ chống lại nguy cơ tấn công do bởi Cross-site Request Forgery (CSRF). Chúng ta vẫn nên mã hóa thẻ sử dụng JWE nếu chúng ta đặt bất kỳ các thông tin nhạy cảm nào bên trong mã token, và truyền token trên kênh thông tin an toàn HTTPS để ngăn chặn các cuộc tấn công man-in-the-middle.
  + Hiệu năng: Không cần phải thực hiện tìm kiếm và giải mã phiên làm việc trên mỗi yêu cầu. Điều duy nhất chúng ta phải làm là tính toán HMAC-256 để xác minh nội dung mã token là hợp lệ và đọc nội dung của nó.
    1. Làm thế nào để bảo mật JWT

Có rất nhiều thư viện giúp chúng ta tạo và xác minh JWT, nhưng khi sử dụng JWT vẫn có nhiều điều chúng ta có thể làm để hạn chế các rủi ro bảo mật của ứng dụng:

* Luôn luôn tiến hành xác minh chữ ký trước khi tin tưởng bất kỳ các thông tin trong JWT. Một khả năng gần đây là khi đặc tả JWT cho phép chúng ta đặt trường algorithm là none. Chúng có thể dễ dàng tin tưởng nhận được một mã JWT không có ký số nhưng thực tế nội dung của token đã bị thay đổi. Cách thức tốt nhất là khi tiến hành truyền mã mã khóa bí mật vào hàm thực hiện xác thực một token mà khi đó token có trường algorithm là none. Thì quá trình nên dẫn tới kết quả xác minh thất bại.
* Bảo vệ khóa bí mật dùng để tính toán và xác minh chữ ký. Các khóa bí mật chỉ nên được truy xuất bởi người phát hành (iss) và người sử dụng (sub) được chỉ ra trong claim của token đó.
* Không chứa bất kỳ dữ liệu nhạy cảm nào trong JWT. Những thẻ này thường chỉ được ký số để bảo vệ tính toàn vẹn, mà không được mã hóa an toàn(JWE) nên những dữ liệu này dễ dàng được giải mã để đọc. Nếu lưu trữ các thông tin nhạy cảm trên một JWT chúng ta nên sử dụng đặt tả (JWE) để mã hóa an toàn thông tin.
* Đối với nguy cơ tấn công replay-attacks, chúng ta có thể đính kèm các claim như định danh duy nhất (jti), thời hạn (exp) và thời gian phát hành (iat) để theo dõi thời gian sử dụng của mã token.

## REST và SOAP

Mặc dù sự phổ biến REST nổi lên phổ biến đáng kể trong những năm gần đây, SOAP vẫn là phần được xác định rõ ràng, chuẩn hóa và được hỗ trợ tốt, và được xem là một trong những đại diện công nghệ cho kiến trúc gọi hàm từ xa (RPC). Đã có nhiều cuộc tranh luận kéo dài trong cộng đồng lập trình viên cũng như các nhà khoa học về lợi ích thật sự của REST so với SOAP. REST chỉ là một dạng phong cách thiết kế, là tập hợp chuẩn các quy tắc mà khi áp dụng vào thiết kế sẽ mang lại những hiệu quả nhất định. Nên đáng ra không nên được so sánh SOAP bao gồm nhiều công nghệ và là đại diện của phong cách thiết kế RPC. Nhưng bản so sánh sau đây được đưa ra nhằm vào mục đích đưa ra quyết định lựa chọn đối với một số lĩnh vực cụ thể, và đồng thời là trong kỷ nguyên các thiết bị di động lên ngôi.

## Kết quả so sánh

* + 1. Khả năng mở rộng (Scalability)
* Trong kiến trúc theo phong cách RPC, các dịch vụ web khác nhau có những giao diện khác nhau – mỗi một dịch vụ có một ngữ nghĩa riêng và bao gồm các thông số vận hành của nó. Máy khách muốn giao tiếp được với dịch vụ SOAP thì bắt buộc phải biết tất cả các định nghĩa và ngữ nghĩa của các giao diện dịch vụ. Mặc dù điều đó không phải là vấn đề quá to tát trong các ứng dụng có quy mô nhỏ, nhưng khi trong một môi trường phân tán rộng lớn, nơi hàng trăm dịch vụ web được thực thi, thì phương pháp này có thể dẫn tới việc phụ thuộc chặt chẽ (tight coupling) từ trình khách đối với các giao diện dịch vụ do bởi sự phức tạp và cồng kềnh của nó, gây ra vấn đề nghiêm trọng trong việc ngăn chặn một hệ thống phân tán lớn đạt đến khả năng mở rộng quy mô (Web-scale).
* Với ràng buộc giao diện đồng nhất trong thiết kế hướng REST thì tất cả các tài nguyên của hệ thống đều đưa ra các giao diện tương tự cho máy khách. REST giữ các giao diện giống nhau và lựa chọn các hoạt động phù hợp dựa trên các phương thức mà HTTP sử dụng. Ví dụ, phương thức HTTP GET được sử dụng để đọc, PUT để tạo, DELETE để loại bỏ, UPDATE để cập nhật tài nguyên. Máy khách chỉ cần hiểu được các giao diện dịch vụ cụ thể, và ràng buộc giao diện thống nhất này giúp cho máy khách và máy chủ ít phụ thuộc hơn dẫn tới khả năng tiến hóa một cách độc lập. Máy chủ và máy khách có thể thay đổi một cách tự do miễn sao giao diện chính của nó vẫn không thay đổi. Như vậy, REST làm tăng đáng kể khả năng mở rộng của một hệ thống phân tán.
* Và cũng do bởi ràng buộc tương tác phi trạng thái giữa máy khách và máy chủ trong REST, tất cả các yêu cầu từ phía máy khách đều chứa các thông tin cần thiết để xử lý yêu cầu đó nên các thông tin trạng thái sẽ được lưu trữ ở phía máy khách, và máy chủ không cần phải đồng bộ hóa các thông tin trạng thái đó với máy khách, vậy cho nên máy chủ rất dễ dàng mở rộng theo chiều ngang và cân bằng tải. Nó loại bỏ sự cần thiết cho các nổ lực quản lý các thông tin trạng thái như máy chủ trạng thái, sao chép trạng thái, chia sẻ bộ nhớ giúp cho việc đơn giản hóa khả năng mở rộng.
  + 1. Phụ thuộc dữ liệu (Coupling)
* Trong phong cách thiết kế RPC, các hợp đồng dịch vụ không chỉ định nghĩa các giao diện mà còn định nghĩa các định dạng dữ liệu được sử dụng trong dịch vụ đó. Trong SOAP, định dạng dữ liệu được định nghĩa bên trong WSDL sử dụng ngôn ngữ XML. Sự kết gắn đó giúp ích cho việc tự động hóa sinh mã nguồn cho ứng dụng. Bất kỳ sự thay đổi nào trong kiểu dữ liệu trong dịch vụ đều bắt buộc máy khách phải biên dịch trở lại, điều đó dẫn tới sự phụ thuộc chặt chẽ giữa máy chủ và máy khách.
* Trong REST, các tài nguyên được tương tác thông qua nhiều đại diện của nó. Các định dạng của các đại diện (có thể là HTML, JSON, XML, Text, Images, …) có thể thay đổi dựa trên sự thương lượng giữa máy khách và máy chủ. Dữ liệu được truyền thông qua các thông điệp HTTP mà ở đó các trường content-type và accept trong header của yêu cầu được dùng để chỉ định định dạng dữ liệu truyền tải, REST làm giảm sự phụ thuộc giữa máy chủ và máy khách. REST sử dụng các định dạng dữ liệu chung được xác định thông qua các MIME như là một chuẩn định dạng thống nhất, giúp loại bỏ sự mơ hồ về định nghĩa định dạng dữ liệu và sự phụ thuộc vào chúng.
  + 1. Bảo mật (Security)

SOAP sử dụng giao thức HTTP như một giao thức đường hầm (tunneling protocol). Các thông điệp SOAP được đóng gói dưới dạng một yêu cầu HTTP và được truyền tải bằng phương thức HTTP POST và vì thế nó có thể dễ dàng vượt qua các bức tường lửa sử dụng cổng 80. Điều này hàm ẩn đưa ra những nguy cơ bảo mật một khi các mục đích của các yêu cầu SOAP không thể hiểu được nếu không được phân tích cú pháp, vì thế nếu một thông báo có chứa các yêu cầu độc hại, thì nó không thể được phát hiện bởi các hệ thống tường lửa mà không có giao thức lọc thông điệp đi qua.

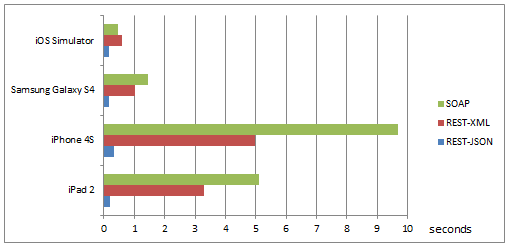
REST cung cấp mỗi tài nguyên với một URI duy nhất và cho phép tương tác bằng các phương thức trong HTTP. Và mỗi phương thức trong HTTP có một ý nghĩa rõ ràng. Máy chủ có thể dễ dàng từ chối một hoạt động cụ thể bằng cách từ chối một URI hay một phương thức cụ thể trong HTTP. Ví dụ, nếu một tài nguyên là chỉ đọc thì chỉ có phương thức HTTP GET là phương thức duy nhất được hỗ trợ, nếu một tài nguyên có thể được xóa bỏ thì phương thức HTTP DELETE sẽ được hỗ trợ. Bốn hoạt động (CRUD) trên bất kỳ tài nguyên nào có thể dễ dàng được cấu hình với chính sách bảo mật thích hợp trên các tường lửa thông qua các URI và phương thức HTTP được sử dụng. Do đó việc thực thi bảo mật với REST được thực hiện dễ dàng hơn rất nhiều.

* + 1. Hiệu năng (Performance)
* REST đơn giản sử dụng các tiêu t được sử dụng rộng rãi trong Web. REST sử dụng giao thức HTTP để truyền tải dữ liệu trực tiếp, vì thế, hiệu quả đạt được do không phải phân tích và đóng gói thành các thông điệp SOAP trên cả máy chủ và khách được đề cập đầu tiên. Một nội dung thông điệp SOAP là nặng hơn nội dung bên trong REST. Khối lượng thông điệp SOAP còn được đính kèm theo các thẻ XML mở rộng và các HTTP header liên quan. Hành động đơn giản nhất là xóa một thực thể có thể được thực hiện bên trong REST bằng cách sử dụng một URI và phương thức DELETE trong HTTP, mà không yêu cầu thêm bất kỳ phần nội dung nào. Tuy nhiên trong SOAP thì sẽ yêu cầu các thẻ XML cụ thể. SOAP do đó được thực hiện kém hơn về độ trễ trong mạng lưới và kích thước gói tin hơn so với REST.
* Một mặt khác REST hỗ trợ lưu trữ tạm thời các phản hồi để tránh các hoạt động không cần thiết để cải thiện hiệu năng. Đó là thông qua các yêu cầu GET của giao thức HTTP và với mục đích đơn nguyên chỉ lấy mà không thay đổi dữ liệu nên các phản hồi có thể được lưu trữ tạm thời bằng cách sử dụng tối ưu khả năng của trình khách chẳng hạn như trình duyệt, …Ngoài ra, REST cung cấp khả năng lựa chọn định dạng gọn nhẹ hơn so với XML là JSON cho phép các máy khách xử lý tối ưu hơn về bộ nhớ và khả năng xử lý. Một ví dụ cụ thể là Amazon.com đã tuyên bố rằng các dịch vụ hướng REST thực thi nhanh hơn sáu lần so với các dịch vụ hướng SOAP.
  + 1. Giao thức trong suốt và độc lập
    2. Khả năng khám phá (Discovery)

Các dịch vụ SOAP được đăng ký và tìm kiếm từ các khu lưu trữ trung tâm như dịch vụ UDDI. REST cung cấp các URI để định danh các nguồn tài nguyên của mình, và lợi ích rõ ràng nhất của việc sử dụng URI là nó chứa đầy đủ các thông tin cần thiết cho các dịch vụ khám phá và triệu gọi toàn cầu mà không cần phải truy vấn thông tin từ trung tâm thông tin UDDI. Một URI thân thiện với con người, có thể được đọc, được phân phối, được đánh dấu hoặc được in. Cùng với nguyên tắc HATEAOS, trạng thái ứng dụng được điều khiển máy chủ. Điều đó có nghĩa là máy khách chỉ cần biết về một URI để bắt đầu dịch vụ. Các trạng thái ứng dụng hay thông tin mở rộng có thể được tìm thấy trong các nội dung phản hồi nhận được từ máy chủ. Điều này giúp cho các dịch vụ REST được khám phá một cách dễ dàng hơn.

* + 1. So sánh thực nghiệm trên ứng dụng di động

Để xây dựng một ứng dụng di động đa chức năng và hiệu suất tối ưu, các dịch vụ web phải phát triển sao cho tối ưu hóa cho khả năng xử lý trên các thiết bị di động. Dịch vụ web RESTful sử dụng định dạng JSON như là định dạng tối ưu được lựa chọn trong kiến trúc tích hợp ứng dụng di động và hệ thống xử lý đầu cuối phía máy chủ. Sau đây là hình ảnh so sánh hiệu năng giữa khả năng xử lý dịch vụ dựa trên nền SOAP với các dịch vụ dựa trên REST trên thiết bị di động.



Hình 1: So sánh hiệu năng SOAP, REST-XML, REST-JSON trên mobile

Như kết quả ở trên, sự khác nhau giữa hiệu năng xử lý của hai nền tảng dịch vụ và rất rõ rệt. Kết luận từ so sánh trên thì dịch vụ web nền REST với kiểu dữ liệu là JSON sẽ có tốc độ thực thi trên các thiết bị di động nhanh hơn từ 9 đến 30 lần. Và cùng với bản chất gọn nhẹ của JSON thì rõ ràng tối ưu về mặt bộ nhớ hơn trên các thiết bị di động hơn so với XML. Hơn nữa, đối với các ứng dụng web sử dụng Javascript thì việc xử lý định dạng JSON là nhanh chóng hiệu quả hơn rất nhiều vì là kiểu dữ liệu đối tượng của Javascript.

## REST trong Java

* + 1. JAX-RS

JAX-RS là một chuẩn đặc tả giao diện lập trình Java được thiết kế để giúp cho việc phát triển các dịch vụ web theo kiến trúc REST một cách dễ dàng hơn. JAX-RS sử dụng tính năng cú pháp chú thích thời gian thực thi (Annotations) được giới thiệu lần đầu trong Java SE 5 để đơn giản hóa việc phát triển và triển khai các ứng dụng dịch vụ web.

Từ phiên bản 1.1 trở đi, JAX-RS là một thành phần chính thức của công nghệ Java EE. Một tính năng đáng chú ý đối với công nghệ Java EE sẽ không cần một cấu hình nào để bắt đầu sử dụng JAX-RS. Còn đối với các môi trường non-Java EE thì chỉ cần một mục nhỏ trong tập tin mô tả *web.xml.*

Phiên bản hiện tại là JAX-RS 2.0 được hình thành vào tháng giêng năm 2011 được phát triển bới nhóm các chuyên gia JSR 339. Các mục tiêu chính là phát triển trở thành một API phổ biến và hỗ trợ Hypermedia theo nguyên tắc HATEOAS trong REST. Cho đến tháng 5 năm 2013, thì phiên bản cuối cùng JAX-RS 2.0 chính thức được phát hành.

Bảng sau tóm tắt một số các cú pháp thông dụng trong JAX-RS:

|  |  |
| --- | --- |
| Cú pháp | Mô tả |
| @Path | Cú pháp @Path xác định đường dẫn tương đối cho một lớp tài nguyên hoặc phương thức |
| @GET,  @POST,  @PUT,  @DELETE | Xác định các loại phương thức yêu cầu trong HTTP đối với một tài nguyên |
| @PathParam | @PathParam là một loại tham số được trích xuất từ một đường dẫn URI và được sử dụng trong phương thức của lớp |
| @QueryParam | @QueryParam là một loại tham số được trích xuất từ các tham số đính kèm theo đường dẫn URI được sử dụng trong phương thức của lớp |
| @Consumes | Quy định chấp nhận xử lý các loại phương tiện truyền thông (MIME) |
| @Produces | Quy định cụ thể các loại phương tiện truyền thông (MIME) được sử dụng như là kết quả trả về |

* + 1. Jersey
* Jersey là một khung phát triển ứng dụng mã nguồn mở có chất lượng để phát triển các dịch vụ web RESTful bằng ngôn ngữ Java, hỗ trợ đặc tả JAX-RS. Sử dụng tính năng chú thích cùng với tập hợp các lớp và giao diện giúp cho việc tạo các dịch vụ web từ các lớp cơ bản POJOs một cách dễ dàng.
* Ngoài các JAX-RS API Jersey cũng cung cấp thêm các chức năng mở rộng khác như hỗ trợ JavaScript Object Notation (JSON), tích hợp với Spring framework và ngôn ngữ mô tả web (WADL).

# DỊCH VỤ WEB DU LỊCH

## Giới thiệu dịch vụ

Travelee là dịch vụ web dựa trên nền tảng REST full, cung cấp cho người dùng chức năng tìm kiếm địa điểm, bình luận, đánh giá địa điểm.

## Mô tả

* Travelee là cộng đồng tin cậy cho mọi người tìm kiếm, đánh giá, bình luận các địa điểm ăn uống: nhà hàng, quán ăn, cafe, bar, karaoke, tiệm bánh, khu du lịch…tại Việt Nam – từ ứng dụng di dộng trên Android. Tất cả thành viên từ Bắc tới Nam, Travelee kết nối những thực khách đến với các địa điểm ăn uống lớn nhỏ cả đất nước.

Hỗ trợ các tính năng:

* **Tìm kiếm dễ dàng –** Công vụ tìm kiếm thông minh bằng cách gõ: tên địa điểm, hoặc tên đường, hoặc tên món ăn, hoặc mục đích, hoặc tên khu vực. Hệ thống tìm kiếm sử dụng gợi ý & xem nhanh thông tin, giúp bạn tìm địa điểm nhanh nhất.
* **Phân loại rõ ràng –** Travelee phân loại các địa điểm ra rất chi tiết: theo mục đích, loại hình, món ăn, giá cả, loại ẩm thực,… Điều này giúp cộng đồng lọc địa điểm theo mục đích của mình rất nhanh chóng.
* **Bình luận & Đánh giá –** Travelee cho phép thành viên chèn hình ảnh vào bình luận, đánh giá địa điểm với 5 tiêu chí: Món ăn, Vị trí, Không gian, Giá cả và Dịch vụ, điều này giúp cho cộng đồng có cái nhìn tổng quan về các tiêu chí của mỗi địa điểm. Do đặc thù của mỗi địa điểm khác nhau, ví dụ Quán cafe, cộng đồng sẽ quan tâm đến Không gian, đối với Quán ăn cộng đồng quan tâm đến chất lượng món,… Đây chính là điểm khác biệt của Travelee so với các dịch vụ khác.

## Danh sách các yêu cầu

* + 1. Đăng nhập

Người dùng có thể dễ dàng đăng ký là thành viên trong cộng đồng bằng cách đơn giản nhất đăng nhập bằng tài khoản Facebook thông qua ứng dụng Androidt, từ đó có thể tham gia tìm kiếm, bình luận, đánh giá về địa điểm mình yêu thích.

* + 1. Tìm kiếm địa điểm

Chức năng tìm kiếm địa điểm là chức năng quan trọng nhất của hệ thống, cho phép các thành viên tìm kiếm nhanh chóng và dễ dàng nhất các địa điểm mong muốn. Bao gồm:

* + 1. Tiêu chí tìm kiếm

Vị trí địa điểm bao gồm (tên đường, tên địa điểm, tên quận, thành phố)

Mục đích (Loại hình, Sở thích, Món ăn, …)

* + 1. Tìm kiếm địa điểm lân cận

Hệ thống sẽ tự động tìm kiếm các địa điểm lân cận theo vị trí hiện tại của người dùng trong một phạm vi người dùng mong muốn.

* + 1. Tìm kiếm địa điểm theo vị trí

Hệ thống sẽ tìm kiếm dựa trên các từ khóa mà người dùng mong muốn– từ khóa có thể là tên địa điểm, tên đường, tên quận hay thành phố.

* + 1. Tìm kiếm địa điểm theo trào lưu và đánh giá

Hệ thống sẽ gợi ý kết quả tìm kiếm thông qua sự đánh giá của các thànhviên khác trong hệ thống bao gồm các địa điểm đang được nhiều người quan tâm, đánh giá tốt.

* + 1. Dẫn đường

Sau khi đã có kết quả tìm kiếm các địa điểm mong muốn, người dùng có thể sử dụng chức năng dẫn đường để có thông tin chi tiết tuyến đường dẫn tới địa điểm mong muốn.

## Check-in địa điểm

Check-in là hành động xác nhận vị trí họ đang ở đâu thông qua sử dụng chức năng định vị toàn cầu GPS. Thông thường trong Facebook người dùng thực hiện việc check-in gắn liền với việc mô tả họ đang làm gì, cảm thấy thế nào, hoặc gắn địa điểm cho một bức ảnh, video.

Ngoài ra Check-in không giới hạn ở việc sử dụng GPS mà có thể mang “tính xã hội” hơn thông qua việc người dùng người dùng có thể check-in tại địa điểm và đặt tên cho địa điểm theo cách riêng của mình (ex: đang ở tại Ngôi nhà hạnh phúc,…) tức là đã thay thế một địa chỉ thương hiệu, cảm xúc để thay thế cho một địa chỉ có tính vật lý.

Ngoài ra việc check-in không chỉ là sở thích, nhu cầu của người dùng mong muốn được lưu giữ những kỷ niệm, những cột mốc, những nơi mà họ đến,… mà còn giúp cho người dùng dễ dàng tìm thấy bạn bè của mình hơn.

* + 1. Đánh giá địa điểm

Người dùng có thể tham gia đánh giá chất lượng của một địa điểm thông qua chức năng đánh giá từ ứng dụng Android. Từ đó hệ thống sử dụng các phản hồi của người dùng để thực hiện đánh giá, gợi ý, phục vụ nhu cầu cho mọi thành viên trong cộng đồng tốt hơn

* + 1. Lưu vào yêu thích

Trong khi đang lướt thông tin các địa điểm cho các dự định du lịch, vui chơi của mình, có thể người dùng sẽ bắt gặp vô số các địa điểm ưng ý và họ muốn lưu lại các thông tin địa điểm đó để có thể truy cập lần sau. Hệ thống hỗ trợ người dùng có thể thêm các địa điểm vào danh sách yêu thích của mình, cũng như tạo ra một bộ sưu tập khác.

* + 1. Bình luận địa điểm

Người dùng có thể đăng bình luận của mình cũng như chia sẻ các cảm xúc đối với từng địa điểm cụ thể.

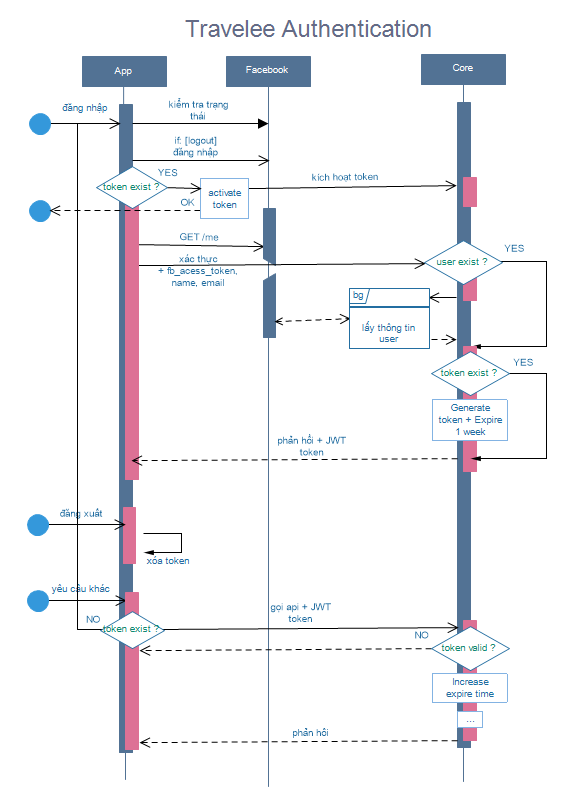
* + 1. Gửi lời mời đến bạn (thông qua facebook)

Người chơi có thể chia sẻ và gửi lời mời đến bạn mình trên Facebook về những địa điểm yêu thích và dự định sẽ đi cùng với nhau.

## Phân tích

* + 1. Yêu cầu đăng nhập

Xác thực người dùng xử dụng JSON Web Token



Hình 4.2: Mô hình xác thực người dùng

Mô tả sơ đồ luồng:

#### Đăng nhập

Người dùng tiến hành đăng nhập thông qua giao diện của ứng dụng Android, ứng dụng Android sẽ kiểm tra trạng thái đăng nhập Facebook bằng cách gửi yêu cầu kiểm tra trạng thái tới Facebook. Người dùng thực hiện đăng nhập Facebook nếu như phản hồi trạng thái là chưa đăng nhập. Còn nếu người dùng đã đăng nhập rồi, sẽ kiểm tra hiện tại mã token có được lưu trong máy hay chưa. Nếu đã mã token đã được lưu trong máy tiến hành gởi yêu cầu kích hoạt mã token. Còn nếu chưa có mã token, ban đầu ứng dụng sẽ lấy thông tin người dùng bằng cách gửi yêu cầu đến máy chủ Facebook, sau đó tiến hành gửi yêu cầu xác thực người dùng với dữ liệu thông tin vừa lấy. Xử lý xác thực ở máy chử, sẽ kiểm tra thông tin người dùng có tồn tại trong hệ thống hay chưa. Nếu thông tin người dùng chưa tồn tại, hệ thống sẽ tạo thông tin người dùng trên hệ thống và đồng thời kích hoạt một xử lý ngầm để lấy các thông tin đầy đủ của người dùng từ máy chủ Facebook. Quá trình sau đó sẽ kiểm tra người dùng đã có mã token hay chưa. Nếu chưa có sẽ tiếp tục xử lý sinh mã token cùng với thời gian hết hạn cho người dùng. Cuối cùng sẽ gửi phản hồi cùng với mã token cho ứng dụng.

#### Đăng xuất

Quá trình đăng xuất diễn ra đơn giản hơn. Ứng dụng phía khách hàng chỉ cần kiểm tra xem mã token hiện có được lưu trong máy hay chưa và nếu có thì xóa mã token được lưu ra khỏi máy.

#### Các yêu cầu khác

Các yêu cầu khác từ ứng dụng được diễn ra giống nhau vì phải cần có mã token mới có thể yêu cầu xử lý nghiệp vụ ở phía máy chủ. Do đó, trước khi gửi yêu cầu lên máy chủ, ứng dụng phải kiểm tra mã token có tồn tại sẵn hay chưa. Còn nếu chưa phải tiến hành bắt buộc người dùng đăng nhập lại. Còn phía máy chủ, mỗi khi nhận được một yêu cầu xử lý nghiệp vụ hợp lệ (bao gồm mã token), hệ thống sẽ tiến hành xác minh mã token có hợp lệ hay không. Nếu như token hợp lệ hệ thống sẽ tăng thời gian hoạt động của mã token đó lên.

* + 1. Yêu cầu tìm kiếm



* + 1. Bảo mật API

Sử dụng JSON Web Token (JWT)

## Kiến trúc hệ thống

* + 1. Mô hình dịch vụ RESTful API

Với yêu cầu của hệ thống là phát triển một dịch vụ web có thể phục vụ cho nhiều ứng dụng khách như Android, IOS, Windows phone hay web browser nên mô hình dịch vụ web dựa trên kiến trúc RESTful sẽ được sử dụng, và hệ thống sẽ cung cấp một API thống nhất để thao tác các chức năng của hệ thống, đồng thời đơn giản hóa việc phát triển giao tiếp giữa các ứng dụng khách đối với máy chủ.



Hình 4.2: Mô hình dịch vụ web RESTful

* + 1. Kiến trúc xử lý đa tầng

Dịch vụ web cũng như các ứng dụng web đôi khi cung cấp các dữ liệu và chức năng tương tự nhau cho các loại ứng dụng khách. Ví dụ như một trang web thương mại điện tử cung cấp giao diện danh mục sản phẩm trực tuyến cho phép người dùng có thể tìm kiếm, xem và mua sản phẩm. Nó có thể hữu ích hơn nữa với việc hỗ trợ một dịch vụ web cho các công ty, các nhà bán lẻ khác và thậm chí là cá nhân có thể thực hiện việc đặt mua hàng tự động. Dịch vụ web cũng hưởng lợi từ việc tách biệt chức năng các thành phần hệ thống để dễ dàng phát triển, bảo trì và nâng cấp từng thành phần riêng lẻ sau này.



Hình 4.2: Kiến trúc đa tầng dịch vụ web RESTful

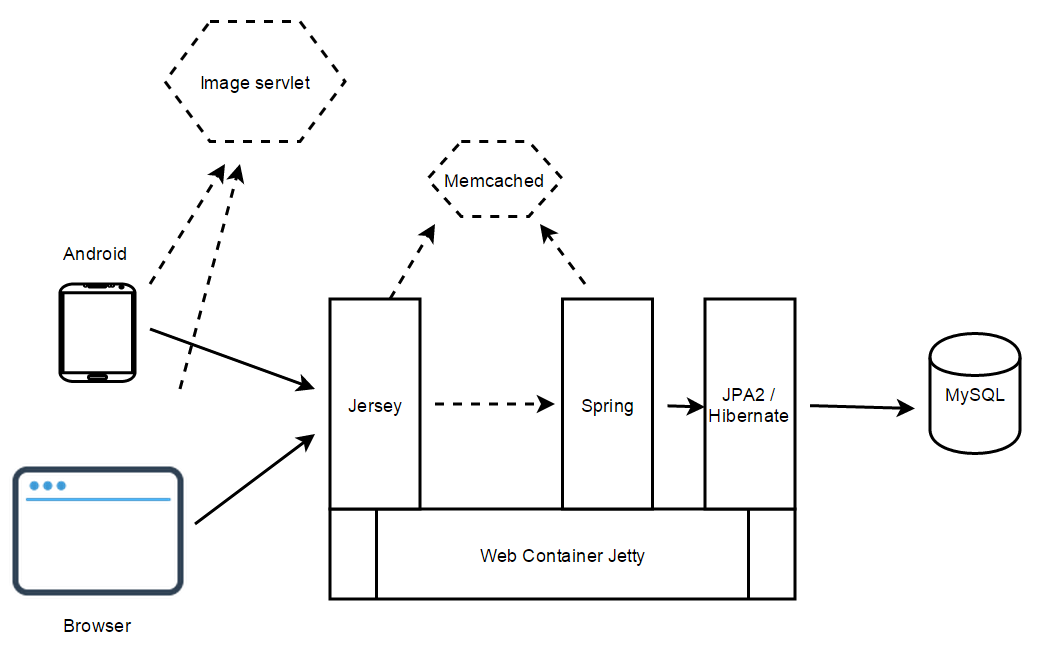
Hệ thống được phân chia thành các thành phần cơ bản như sau:

* **Tầng lưu trữ (Persistence)**: Lớp này thực hiện các nhiệm vụ liên quan đến lưu trữ và truy xuất dữ liệu như đọc (select), lưu, cập nhật (insert, update, delete).
* **Tầng dịch vụ (Business/ Service)**: cung cấp các xử lý nghiệp vụ của hệ thống dưới dạng các dịch vụ cho các tầng phía trên sử dụng. Tầng này trực tiếp giao tiếp với tầng lưu trữ.
* **Tầng biểu diễn (Presentation)**: tầng này trực tiếp tương tác với người dùng hoặc ứng dụng khách thông qua giao diện (API) REST. Đồng thời là tầng biểu diễn dữ liệu theo yêu cầu.

Cơ chế vận hành tương tự như mô hình 3 lớp, các yêu cầu sẽ được xử lý tuần tự qua các tầng. Đầu tiên yêu cầu sẽ được gởi từ ứng dụng khách đến biểu diễn thông qua cổng RESTful. Tại tầng này, các yêu cầu sẽ được kiểm tra sơ bộ, nếu hợp lệ chúng sẽ được chuyển xuống tầng dịch vụ. Tại tầng dịch vụ, các thông tin sẽ được xử lý, tính toán theo đúng yêu cầu đã gửi, nếu không cần tới dữ liệu từ CSDL thì tầng này sẽ trả kết quả về cho tầng presentation, ngược lại tầng dịch vụ sẽ đẩy dữ liệu (thông tin đã xử lý) xuống tầng dữ liệu để thao tác với CSDL. Tầng lưu trữ sẽ thao tác với CSDL và trả kết quả về cho tầng dịch vụ xử lý rồi sau đó gửi lại cho tầng biểu diễn thực hiện biểu diễn các thông tin theo yêu cầu của trình khách.

Ưu điểm mô hình kiến trúc đa tầng đối với dịch vụ web giúp cho dự án được tổ chức cấu trúc rõ ràng, dễ dùng lại. Từ đó việc phát triển và bảo trì hệ thống sẽ thuận lợi hơn. Điều này giúp cho chúng ta tiết kiệm thời gian khi mở rộng chương trình trong tương lai.

* + 1. Mô hình cài đặt chi tiết



Hình 4.2: Mô hình cài đặt chi tiết

Các thành phần và phiên bản cài đặt:

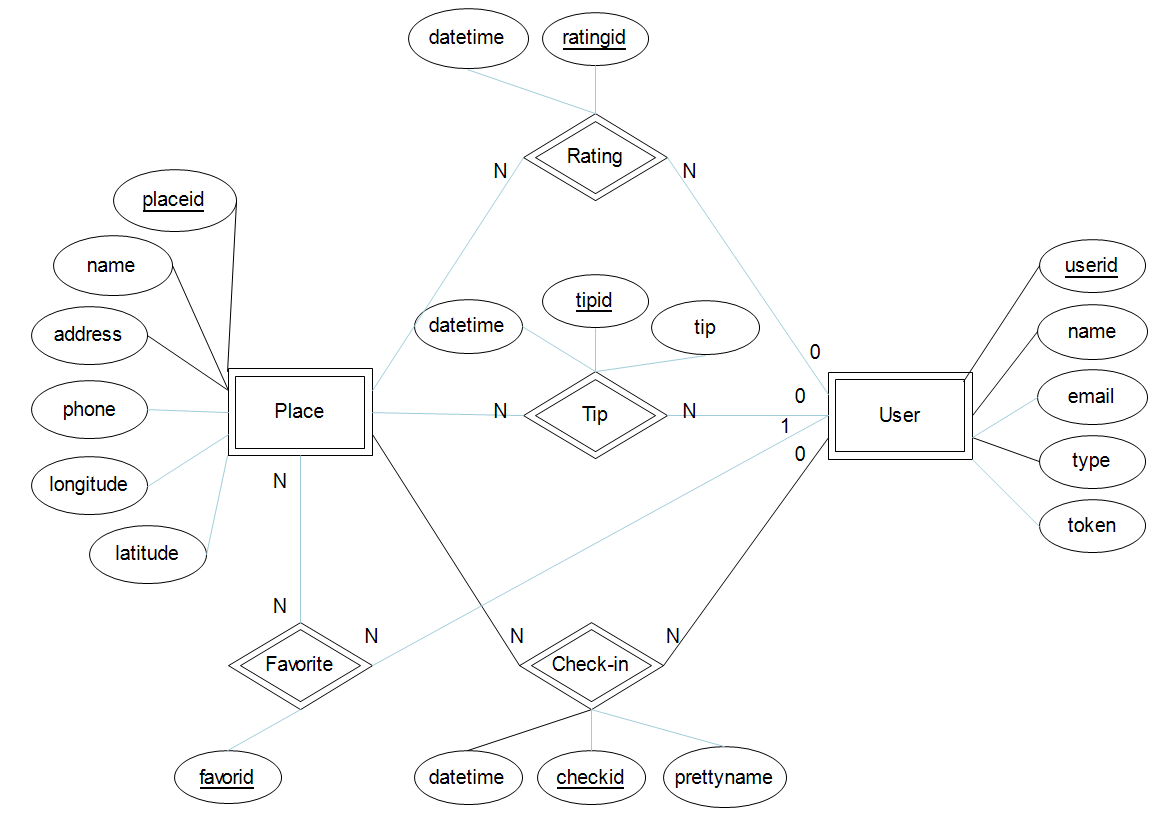
* **Rest API gateway (Jersey)** – Jersey đóng vai trò là một façade giao diện tương tác với trình khách. Jersey là một mã nguồn mở hỗ trợ đơn giản hóa việc phát triển dịch vụ RESTful trong Java. Hỗ trợ các API tuân thủ theo đặc tả JAX-RS (JSR 311 & JSR 339). Phiên bản được sử dụng 2.9
* **Tầng dịch vụ (Spring)** – Thành quả của việc này là nó đã tạo nên một bộ khung, cung cấp một mô hình nhất quán và làm cho nó có thể áp dụng vào hầu hết các loại ứng dụng được tạo trên nền tảng Java ngày nay. Spring cung cấp nhiều mô đun xử lý độc lập giúp cho đơn giản hóa việc phát triển các nghiệp vụ phức tạp.
* **Tầng lưu trữ (JPA 2 / Hibernate)** – Hibernate cung cấp cơ chế ánh xạ ORM (Object Relational Mapping) và nằm ở tầng lưu trữ. Hibernate giúp chúng ta phát triển ứng dụng mà chỉ cần chú tâm vào những tầng nghiệp vụ chính mà không phải bận tâm nhiều về vấn đề truy vấn các thông. Phiên bản được sử dụng 4
* **Cơ sở dữ liệu (MySQL)** – Dữ liệu của hệ thống được lưu trữ trong hệ cơ sở dữ liệu MySQL. Phiên bản được sử dụng 5.7
* **Web container (Jetty)** – Mã nguồn được đóng gói từ chương trình Maven thành các tập tin .war và được triển khai trên Jetty. Phiên bản được sử dụng 9
* **Image Servlet (Resize image on the fly)** – Máy chủ phục vụ yêu cầu về hính ảnh trong hệ thống. Hỗ trợ các tính năng nân cao như chuyển đổi kích thước, định dạng, xử lý khi có yêu cầu giúp cho giảm thiểu gánh nặng quản lý phục vụ ảnh cho các trình khách khác nhau.Hỗ trợ chuyển đổi, xử lý ảnh thông qua chuỗi truy vấn

Ex: image.jpg?width=100&height=100

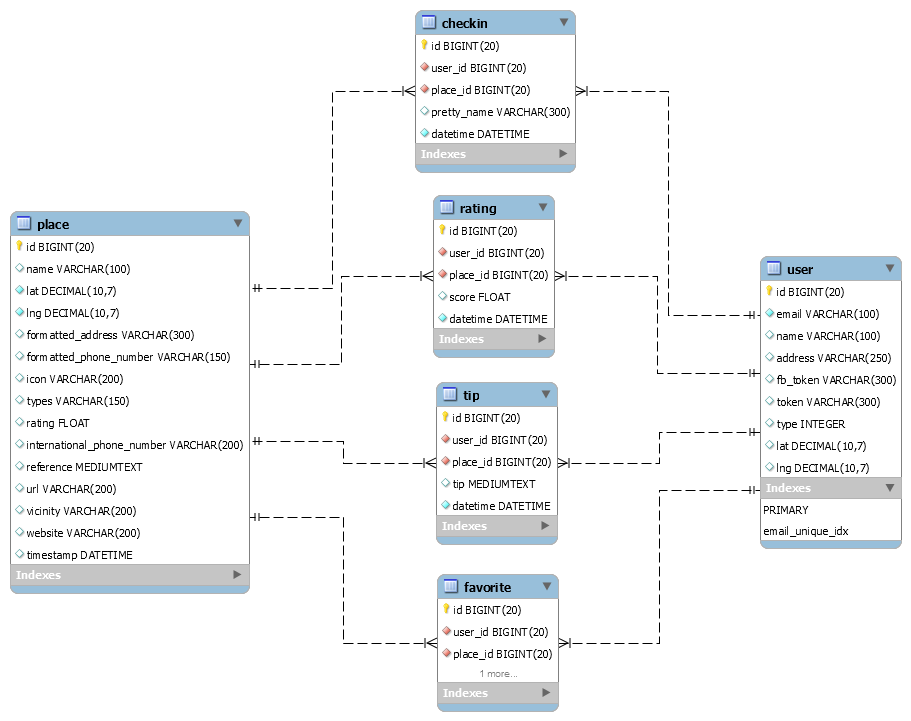
* **Memcached –** nhằm giảm thiểu truy xuất tới cơ sở dữ liệu và phục vụ nhu cầu truy xuất đối tượng nhanh và ít có sự thay đổi. Memcached đóng vai trò trung gian giữa tầng biểu diễn và tầng dịch vụ.

## Thiết kế dữ liệu

* + 1. Mô hình ERD



* + 1. Mô hình quan hệ

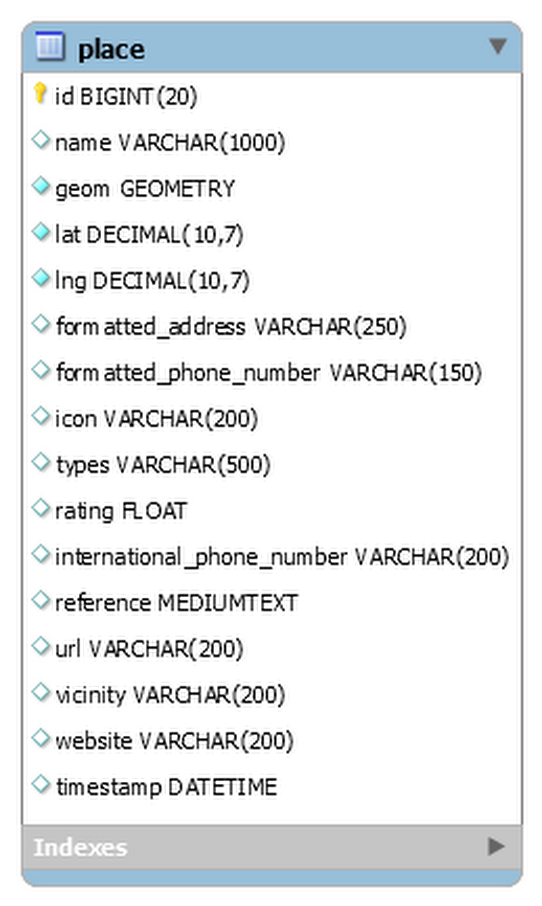


* + 1. Thiết kế chỉ mục

#### Spatial Index

MySQL hỗ trợ đánh chỉ mục không gian đối với các đối tượng hình học. Từ đó giúp cho việc tìm kiếm không gian theo vùng, phạm vi được thực hiện một cách nhanh chóng.

Mỗi dữ liệu địa điểm bên trong bảng Place đều có thông tin kinh độ (lon), vĩ độ (lat) chỉ định tọa độ của địa điểm không gian. Mô hình quan hệ của bảng Place sau khi được thêm chỉ mục không gian:



Hình 4.2: Mô hình quan hệ được đánh chỉ mục không gian

#### Full search index

MySQL hỗ trợ tìm kiếm văn bản bằng lệnh LIKE và các biểu thức tìm kiếm. Tuy nhiên khi cột dữ liệu dạng TEXT có dung lượng lớn và số dòng nhiều thì những phương pháp nêu trên bộc lộ những giới hạn sau:

* Hiệu năng: MySQL phải quét qua toàn bộ bảng để tìm từng dòng theo điều kiện lọc chỉ ra ở lệnh LIKE hay tìm những mẫu văn bản phù hợp với mẫu được chỉ ra trong biểu thức tìm kiếm.
* Tính linh hoạt: Ví dụ đối với yêu cầu tìm kiếm tất cả các sản phẩm có chứa từ ‘car’ trong phần mô tả nhưng lại không chứa từ ‘classic’ thì cả lệnh LIKE và biểu thức tìm kiếm đều bộc lộ hạn chế.
* Xếp hạng tìm kiếm: không chỉ ra được dòng nào trong tập kết quả tìm kiếm là phù hợp với yêu cầu tìm kiếm nhất.

Từ những giới hạn trên, MySQL cung cấp tính năng tìm kiếm ‘toàn bộ’ – full text. Xét về mặt kĩ thuật thì MySQL đánh chỉ mục các từ của cột được tiến hành tìm kiếm full-text. Như thế việc tìm kiếm trên mục này sẽ hiệu quả hơn. MySQL sử dụng những thuật toán phức tạp để xác định xem dòng nào phù hợp với yêu cầu tìm kiếm nhất

Những đặc tính quan trọng của tìm kiếm full-text trong MySQL:

* Giao diện thuần túy SQL: chúng ta sử dụng lệnh LIKE để tìm kiếm full-text.
* Chỉ mục hoàn toàn mang tính động: MySQL tự động lập chỉ mục cột văn bản bất cứ khi nào có diễn ra thay đổi dữ liệu trên cột này.
* Điều tiết dung lượng chỉ mục: không chiếm nhiều dung lượng cho việc lưu trữ chỉ mục.
* Tìm kiếm nhanh chóng ngay cả đối với những câu truy vấn phức tạp.
* Lưu ý: phiên bản hiện tại MySQL 5.6 chỉ hỗ trợ chức năng tìm kiếm full-text đối với bảng dữ liệu kiểu MyISam. Nhưng từ phiên bản mới ra 5.7 thì MySQL đã hỗ trợ đồng thời cả chỉ mục không gian và tìm kiếm toàn văn bản.

## Thiết kế REST API

Bản sau mô tả danh sách các API mà hệ thống hỗ trợ:

|  |  |  |  |
| --- | --- | --- | --- |
| Tài nguyên | URI | Mô tả | HTTP |
| place | /places | Trả về danh sách các địa điểm | GET |
| Tạo mới địa điểm | POST |
|  | PUT |
|  | DELETE |
| /places/{id} | Trả về địa điểm có mã là id | GET |
| Cập nhật | Tạo mới địa điểm | POST |
| Cập nhật | Tạo mới địa điểm | PUT |
| Xóa địa điểm | DELETE |
| /places/textsearch | Trả về danh sách các địa điểm theo các tiêu chí tìm kiếm | GET |
|  | POST |
|  | PUT |
|  | DELETE |
| /places/nearbysearch | Trả về danh sách các địa điểm xung quanh | GET |
|  | POST |
|  | PUT |
|  | DELETE |
| /places/{id}/checkin | Trả về danh sách các checkin người dùng tại địa điểm | GET |
| Tạo mới một đối tượng checkin | POST |
|  | PUT |
|  | DELETE |
| /places/{id}/tip | Trả về danh sách tip của người dùng thuộc địa điểm | GET |
| Tạo mới một đối tượng checkin | POST |
|  | PUT |
|  | DELETE |
| photo | /photos | Trả về danh sách các đối tượng hình ảnh | GET |
| Tạo mới một đối tượng hình ảnh | POST |
|  | PUT |
|  | DELETE |
| /photos/{id} | Trả về thông tin đối tượng photo | GET |
| Tạo mới | Cập nhật thông tin đối tượng photo | POST |
| Tạo mới | Cập nhật thông tin đối tượng photo | PUT |
| Xóa thông tin đố tượng photo | DELETE |
| /places/{id}/photos | Trả về danh sách hình ảnh thuộc về địa điểm | GET |
| Tạo mới đối tượng hình ảnh | POST |
|  | PUT |
|  | DELETE |
| /places/{id}/photos/{id} | Trả về đối tượng hình ảnh thuộc về địa điểm | GET |
| Cập nhật | Tạo mới địa điểm | POST |
| Cập nhật | Tạo mới địa điểm | PUT |
| Xóa đối tượng hình ảnh | DELETE |
| city | /cities | Trả về danh sách các thành phố | GET |
|  | POST |
|  | PUT |
|  | DELETE |
| user | /users/{id} | Trả về thông tin người dùng có mã id | GET |
| Tạo mới | Cập nhật thông tin người dùng | POST |
| Tạo mới | Cập nhật thông tin người dùng | PUT |
| Xóa thông tin người dùng | DELETE |
| checkin | /checkins | Trả về danh sách checkins | GET |
| Tạo mới checkin | POST |
| Tạo mới checkin | PUT |
|  | DELETE |
| rating | /ratings | Trả về danh sách ratings | GET |
| Tạo mới đối tượng rating | POST |
|  | PUT |
|  | DELETE |
| /ratings/{ id} | Trả về thông tin đối tượng rating | GET |
| Tạo mới | Cập nhật đối tượng rating | POST |
| Tạo mới | Cập nhật đối tượng rating | PUT |
| Xóa đối tượng rating | DELETE |
| tip | /tips | Trả về danh sách các tips | GET |
| Tạo mới đối tượng tip | POST |
|  | PUT |
|  | DELETE |
| /tips/{id} | Trả về thông tin tip | GET |
| Tạo mới | Cập nhật đối tượng tip | POST |
| Tạo mới | Cập nhật đối tượng tip | PUT |
| Xóa đối tượng tip | DELETE |
| nearby | /nearby | Trả về danh sách các địa điểm cận kề | GET |
|  | POST |
|  | PUT |
|  | DELETE |

* + 1. Danh sách các mã trạng thái.
    2. Quản lý lỗi hệ thống
* Xử lý lỗi là điều cần thiết trong thiết kế các API REST. Thường có hai nguyên nhân chính dẫn đến xuất hiện lỗi trong hệ thống: hoặc là lỗi xảy ra trên hệ thống, hoặc là lỗi do các yêu cầu không hợp lệ từ phía trình khách.
* Cách thức đầu tiên để cho trình khách biết rằng có lỗi xảy ra là bằng cách gửi mã lỗi HTTP trong bên trong các phản hồi từ phía máy chủ:
* **4xx Client Error** – mã trạng thái thông báo cho trình khách rằng có lỗi xảy ra ở phía trình khách.
* **5xx Server Error** – mã trạng thái thông báo cho trình khách rằng máy chủ thất bại trong việc thực hiện yêu cầu hợp lệ của trình khách.
* Cách thức thứ hai là cung cấp cho trình khách càng nhiều thông tin càng tốt, các thông tin có thể xảy ra lỗi. Một cấu trúc phản hồi thống nhất đi kèm với các siêu dữ liệu có ích cho trình khách cũng như các nhà lập trình hệ thống:

|  |
| --- |
| * { * "status": 400, * "code": 400, * "message": "Provided data not sufficient for insertion", * "link": "http://www.codingpedia.org/ama/tutorial-rest-api-design-and-implementation-with-jersey-and-spring", * "developerMessage": "Please verify that the feed is properly generated/set" * } |

* Các thuộc tính trong cấu trúc phản hồi:
* **status:** chứa thông tin mã trạng thái của HTTP, vì vậy mà nhà phát triển có thể “nhìn thấy” lỗi thông báo ngay trong dữ liệu phản hồi trả về máy chủ mà không cần phải phân tích các tiêu đề của HTTP.
* **code:** đây là mã nội bộ thông báo các trạng thái của ứng dụng.
* **message:** mô tả ngắn gọn của lỗi xảy ra trong hệ thống, từ đó nhà phát triển có thể dễ dàng nhanh chóng biết được nguyên nhân dẫn đến lỗi.
* **link:** trỏ đến nguồn tài nguyên trực tuyến, nơi có thể chứa các thông tin chi tiết liên quan đến lỗi xảy ra trên hệ thống.
* **developerMessage:** thông tin chi tiết, các thông tin bổ sung có liên quan đến nhà phát triển. Điều này chỉ nên xảy ra khi chế độ “debug” được kích hoạt.

## Kết quả thử nghiệm

# ỨNG DỤNG ANDROID QUẢNG BÁ DU LỊCH

## Xác định yêu cầu

## Giới thiệu khái quát

* Ứng dụng Travlee là ứng dụng du lịch phục vụ cho nhu cầu du lịch trong thời đại công nghệ thông tin hiện nay. Với chức năng cho phép tìm kiếm danh sách các địa điểm như nhà hàng, cafe, địa điểm ăn uống đường phố, cửa hàng mua sắm, chỉ đường đi,…
* Ứng dụng sử dụng kỹ thuật lấy dữ liệu địa điểm, bao gồm cả hình ảnh thông qua dịch vụ REST ful webservice, đồng thời cho phép người dùng tương tác với service thông qua thao tác đăng nhập, đánh giá địa điểm.
* Ứng dụng sử dụng facebook sdk để phục vụ cho chức năng Đăng nhập và chia sẻ thông tin thông qua tài khoản facebook.
* Ứng dụng sử dụng google map service để lấy tọa độ hiện của người dùng và chỉ đường đi.

## Khái quát các chức năng

* + 1. Chức năng Đăng nhập

#### Mục tiêu

Sử dụng tài khoản facebook để đăng nhập, thông qua tài khoản facebook có thể review đánh giá địa điểm và chia sẻ cảm xúc, thông tin về địa điểm cho bạn bè và người thân thông qua mạng xã hội facebook.

#### Kết quả

Phát triển thành công chức năng Đăng nhập, cho phép người dùng review, đánh giá địa điểm và chia sẻ thông tin về địa điểm, cảm xúc trải nghiệm cho bạn bè, người thân, thông qua mạng xã hội facebook.

* + 1. Chức năng tìm kiếm địa điểm – quán cafe

#### Mục tiêu

Tìm kiếm danh sách các quán cafe xung quanh khu vực với một tọa độ cho trước hoặc lấy tọa độ hiện tại của người dùng. Danh sách địa điểm quán cafe lấy từ google service có cập nhật thêm và được lưu vào database

#### Kết quả

Phát triển thành công chức năng tìm kiếm địa điểm – quán café.

* + 1. Chức năng tìm kiếm địa điểm – nhà hàng

#### Mục tiêu

Tìm kiếm danh sách các nhà hàng ăn uống xung quanh khu vực với một tọa độ cho trước hoặc lấy tọa độ hiện tại của người dùng. Danh sách địa điểm nhà hàng lấy từ google service có cập nhật thêm và được lưu vào database.

#### Kết quả

Phát triển thành công chức năng tìm kiếm địa điểm – nhà hàng

* + 1. Chức năng tìm kiếm địa điểm – quán ăn đường phố

#### Mục tiêu

Tìm kiếm danh sách các quán ăn đường phố xung quanh khu vực với một tọa độ cho trước hoặc lấy tọa độ hiện tại của người dùng. Danh sách địa điểm nhà hàng lấy từ google service có cập nhật thêm và được lưu vào database.

#### Kết quả

Phát triển thành công chức năng tìm kiếm địa điểm – quán ăn đường phố.

* + 1. Chức năng tìm kiếm địa điểm – bệnh viện, dịch vụ nghỉ dưỡng

#### Mục tiêu

Tìm kiếm danh sách các địa điểm bệnh viện, dịch vụ nghỉ dưỡng xung quanh khu vực với một tọa độ cho trước hoặc lấy tọa độ hiện tại của người dùng. Danh sách địa điểm nhà hàng lấy từ google service có cập nhật thêm và được lưu vào database.

#### Kết quả

Phát triển thành công chức năng tìm kiếm địa điểm – bệnh viện, dịch vụ nghỉ dưỡng.

* + 1. Chức năng tìm kiếm địa điểm – địa điểm mua sắm

#### Mục tiêu

Tìm kiếm danh sách các địa điểm mua sắm xung quanh khu vực với một tọa độ cho trước hoặc lấy tọa độ hiện tại của người dùng. Danh sách địa điểm nhà hàng lấy từ google service có cập nhật thêm và được lưu vào database.

#### Kết quả

Phát triển thành công chức năng tìm kiếm địa điểm – địa điểm mua sắm.

* + 1. Chức năng tìm kiếm địa điểm – địa điểm tham quan cộng đồng

#### Mục tiêu

Tìm kiếm danh sách các địa điểm tham quan cộng đồng như nhà thờ, khu di tích xung quanh khu vực với một tọa độ cho trước hoặc lấy tọa độ hiện tại của người dùng. Danh sách địa điểm nhà hàng lấy từ google service có cập nhật thêm và được lưu vào database.

#### Kết quả

Phát triển thành công chức năng tìm kiếm địa điểm – địa điểm tham quan cộng đồng.

* + 1. Chức năng tìm tìm kiếm nhanh

#### Mục tiêu

Tìm kiếm nhanh trong danh sách các địa điểm đã được tải về thiết bị android và hiển thị trong list view.

#### Kết quả

Phát triển thành công chức năng tìm kiếm nhanh.

* + 1. Chức năng chỉ đường đi bằng google map

#### Mục tiêu

Chỉ đường đi từ địa điểm hiện tại của người dùng đến địa điểm cần đến thông qua google map api.

#### Kết quả

Phát triển thành công chức năng chỉ đường đi.

* + 1. Chức năng tìm chia sẻ cho cho bạn bè, người thân thông qua mạng xã hội facebook

#### Mục tiêu

Người dùng có thể chia sẻ thông tin về địa điểm cũng như cảm xúc trải nghiệm thông qua mạng xã hội facebook.

#### Kết quả

Phát triển thành công chức năng chia sẻ thông qua mạng xã hội facebook.

* + 1. Chức năng đánh giá địa điểm

#### Mục tiêu

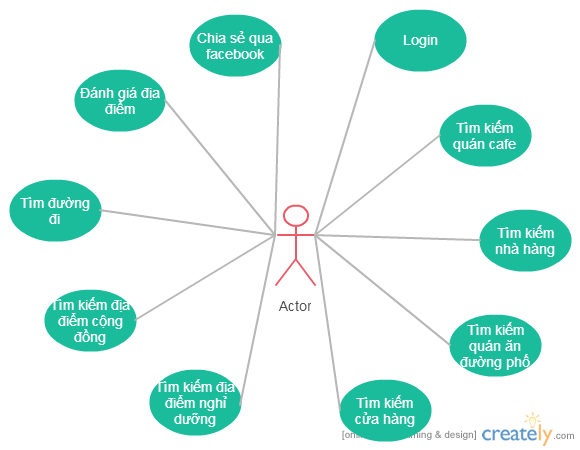
Người dùng có thể review đánh giá địa điểm – chia sẻ cho cộng đồng du lịch, nhằm giúp cho chất lượng dịch vụ được tốt hơn.

#### Kết quả

Phát triển thành công chức năng đánh giá địa điểm.

## Phân tích

* + 1. Lược đồ use case



Hình 4.1: Lượt đồ use case

* + 1. Mô tả lược đồ use case

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Use case ID** | **Tên chức năng** | **Mô tả** |
| 1 | UC\_001 | Đăng nhập | Người dùng sử dụng tài khoản facebook để đăng nhập, nếu chưa có tài khoản facebook có thể đăng kí. |
| 2 | UC\_002 | Tìm kiếm quán cafe | Người dùng có thể tìm kiếm danh sách quán café thông qua địa điểm cho trước hoặc vị trí hiện tại của người dùng. |
| 3 | UC\_003 | Tìm kiếm nhà hàng | Người dùng có thể tìm kiếm danh sách nhà hàng thông qua địa điểm cho trước hoặc vị trí hiện tại của người dùng. |
| 4 | UC\_004 | Tìm kiếm quán ăn đường phố | Người dùng có thể tìm kiếm danh sách quán ăn đường phố thông qua địa điểm cho trước hoặc vị trí hiện tại của người dùng. |
| 5 | UC\_005 | Tìm kiếm cửa hàng mua sắm | Người dùng có thể tìm kiếm danh sách cửa hàng mua sắm thông qua địa điểm cho trước hoặc vị trí hiện tại của người dùng. |
| 6 | UC\_006 | Tìm kiếm địa điểm nghỉ dưỡng, chăm sóc sức khỏe | Người dùng có thể tìm kiếm danh sách địa điểm chăm sóc sức khỏe, nghỉ dưỡng thông qua địa điểm cho trước hoặc vị trí hiện tại của người dùng. |
| 7 | UC\_007 | Tìm kiếm địa điểm tham quan công cộng | Người dùng có thể tìm kiếm danh sách địa điểm tham quan công cộng thông qua địa điểm cho trước hoặc vị trí hiện tại của người dùng. |
| 8 | UC\_008 | Tìm đường đi | Tìm đường đi từ địa điểm hiện tại đến địa điểm tìm được |
| 9 | UC\_009 | Đánh giá địa điểm | Đánh giá địa điểm bằng tài khoản facebook. |
| 10 | UC\_010 | Chia sẻ qua facebook | Chia sẻ cảm xúc trải nghiệm về địa điểm cho bạn bè người thân qua mạng xã hội facebook |

## Đặc tả một số use case chính

Vì có rất nhiều use case do đó nhóm chúng em chọn và đặc tả cho một số use case chính như sau:

* + 1. Use case “Đăng nhập”

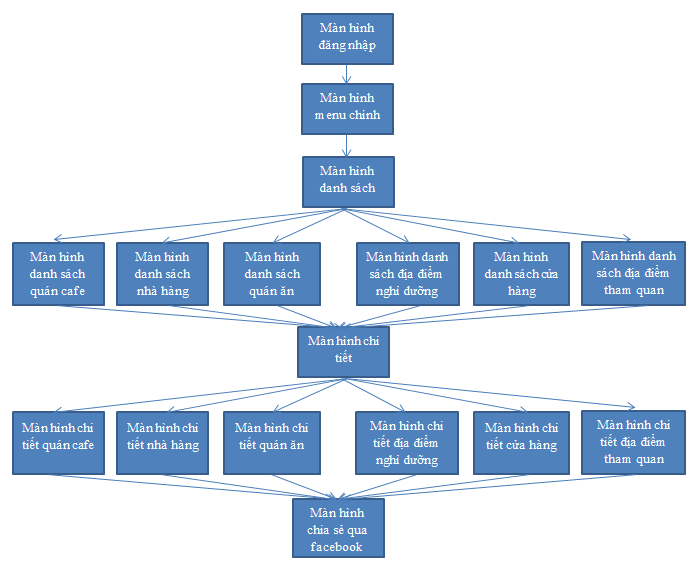
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case ID:** | UC\_001 | | | |
| **Tên use case:** | Đăng nhập | | | |
| **Người tạo:** | Lê Phước Thạch | | **Người cập nhật cuối:** | ……………… |
| **Ngày tạo:** | ………………… | | **Ngày cập nhật cuối:** | ……………… |
| **Mô tả:** | | Người dùng Đăng nhập vào ứng dụng bằng tài khoản facebook | | |
| **Điều kiện kích hoạt:** | | Click vào nút chọn “Đăng nhập bằng tài khoản facebook. | | |
| **Điều kiện đầu:** | | Người dùng đã có tài khoản facebook | | |
| **Điều kiện sau:** | | Nhập đúng tài khoản facebook | | |
| **Tiến trình thường:** | | 1. Nhập đúng tên đăng nhập và mật khẩu tài khoản facebook 2. Click chọn nút “Đăng nhập” | | |
| **Tiến trình phụ:** | | Người dùng có thể bỏ qua bước đăng nhập bằng tài khoản facebok mà vào thẳng ứng dụng | | |
| **Các ngoại lệ:** | | 1. Người dùng chưa có tài khoản facebook 2. Đăng nhập sai thông tin | | |
| **Các use case liên quan:** | | 1. Đánh giá địa điểm 2. Chia sẻ trên facebook | | |
| **Yêu cầu riêng biệt:** | |  | | |

* + 1. Use case “Tìm kiếm quán cafe”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use case ID:** | UC\_002 | | | |
| **Tên use case:** | Tìm kiếm quán cafe | | | |
| **Người tạo:** | Lê Phước Thạch | | **Người cập nhật cuối:** | ……………… |
| **Ngày tạo:** | ………………… | | **Ngày cập nhật cuối:** | ……………… |
| **Mô tả:** | | Người dùng tìm kiếm các quán cafe lân cận hoặc qua một địa điểm cho trước | | |
| **Điều kiện kích hoạt:** | | Click vào nút chọn “cafe” trên màn hình menu | | |
| **Điều kiện đầu:** | | Người dùng đã đăng nhập bằng tài khoản facebook hoặc bỏ qua đăng nhập để vào màn hình menu | | |
| **Điều kiện sau:** | |  | | |
| **Tiến trình thường:** | | 1. Đăng nhập thành công bằng tài khoản facebook hoặc bỏ qua để vào màn hình menu 2. Click chọn nút “cafe” | | |
| **Tiến trình phụ:** | |  | | |
| **Các ngoại lệ:** | |  | | |
| **Các use case liên quan:** | | 1. Tìm kiếm nhà hàng 2. Tìm kiếm quán ăn đường phố 3. Tìm kiếm địa điểm nghỉ dưỡng 4. Tìm kiếm cửa hàng 5. Tìm kiếm địa điểm tham quan công cộng | | |
| **Yêu cầu riêng biệt:** | |  | | |

## Thiết kế màn hình

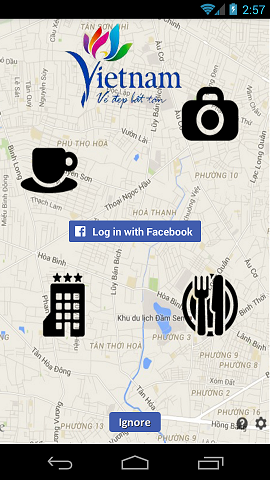
Sơ đồ các màn hình:



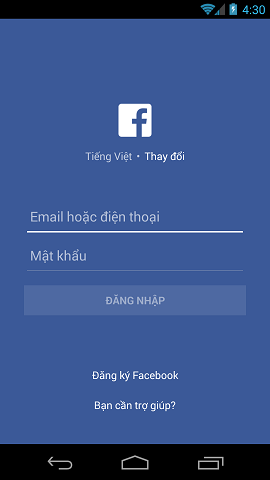
*Bảng mô tả*

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên màn hình** | **Ý nghĩa/ghi chú** |
| 1 | Màn hình đăng nhập | Cho phép người dùng chọn đăng nhập bằng tài khoản facebook hoặc bỏ qua để vào menu chính |
| 2 | Màn hình menu chính | Màn hình hiển thị danh sách các mục địa điểm tìm kiếm bao gồm: coffee, restaurant, food, health, establishment, store |
| 3 | Màn hình chi tiết quán cafe | Hiển thị thông tin chi tiết quán café, cho phép đánh giá địa điểm, tìm đường đi và chia sẻ thông tin qua facebook |
| 4 | Màn hình chi tiết nhà hàng | Hiển thị thông tin chi tiết nhà hàng, cho phép đánh giá địa điểm, tìm đường đi và chia sẻ thông tin qua facebook |
| 5 | Màn hình chi tiết quán ăn | Hiển thị thông tin chi tiết quán ăn, cho phép đánh giá địa điểm, tìm đường đi và chia sẻ thông tin qua facebook |
| 6 | Màn hình chi tiết địa điểm nghỉ dưỡng | Hiển thị thông tin chi tiết địa điểm nghỉ dưỡng, cho phép đánh giá địa điểm, tìm đường đi và chia sẻ thông tin qua facebook |
| 7 | Màn hình chi tiết cửa hàng mua sắm | Hiển thị thông tin chi tiết cửa hàng, cho phép đánh giá địa điểm, tìm đường đi và chia sẻ thông tin qua facebook |
| 8 | Màn hình chi tiết địa điểm công cộng, tham quan | Hiển thị thông tin chi tiết địa điểm công cộng, cho phép đánh giá địa điểm, tìm đường đi và chia sẻ thông tin qua facebook |
| 9 | Màn hình chi tiết tìm đường đi | Hiển thị thông tin chi tiết tìm đường đi, cho phép đánh giá địa điểm, tìm đường đi và chia sẻ thông tin qua facebook |
| 10 | Màn hình chia sẻ facebook | Màn hình chia sẻ cho bạn bè, người thân qua facebook |

* + 1. Thiết kế giao diện màn hình Đăng nhập



|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Button login* | Facebook login button | Đăng nhập bằng tài khoản facebook |
| 2 | *Button Ignore* | Imagebuttion | Sử dụng các chức năng tìm kiếm không cần đăng nhập (không cho phép đánh giá) |

****

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Edit text Username* | Facebook login username | Hiển thị yêu cầu người dùng nhập email hoặc số điện thoại sử dụng facebook |
| 2 | *Edit text Password* | Facebook login password | Hiển thị yêu cầu người mật khẩu cho tài khoản sử dụng facebook |
| 3 | *Button đăng nhập* | Facebook login button | Thực hiện đăng nhập bằng tài khoản facebook |

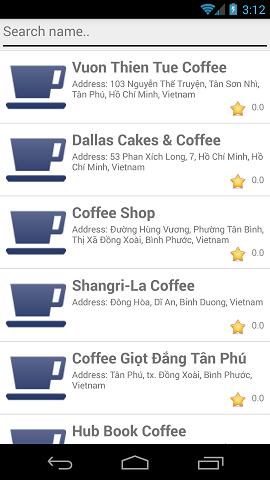
* + 1. Thiết kế giao diện màn hình menu



|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Image button Coffee* | Imagebutton | Hiển thị danh sách quán caffe khi người dùng chọn |
| 2 | *Image button food* | Imagebutton | Hiển thị danh sách quán ăn khi người dùng chọn |
| 3 | *Image button restaurant* | Imagebutton | Hiển thị danh sách nhà hàng khi người dùng chọn |
| 4 | *Image button store* | Imagebutton | Hiển thị danh sách cửa hàng khi người dùng chọn |
| 5 | *Image button health* | Imagebutton | Hiển thị danh sách địa điểm nghỉ dưỡng khi người dùng chọn |
| 6 | *Image button establishment* | Imagebutton | Hiển thị danh sách địa điểm công cộng khi người dùng chọn |

* + 1. Thiết kế màn hình danh sách

#### Thiết kế giao diện màn hình danh sách quán cafe



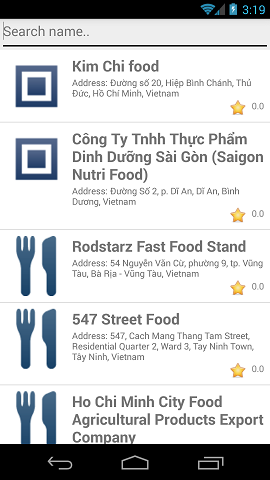
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Danh sách quán cafe* | List view | Hiển thị danh sách quán cafe, người dùng có thể xem chi tiết khi click chọn |
| 2 | *Trường tìm kiếm* | Edit text | Tìm kiếm quán café theo tên |

#### Thiết kế giao diện màn hình danh sách nhà hàng



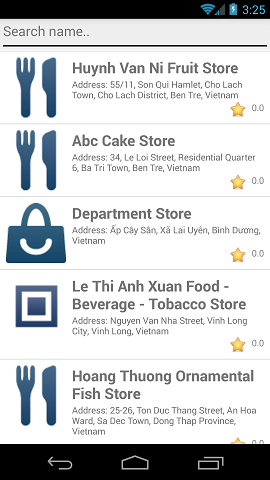
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Danh sách nhà hàng* | List view | Hiển thị danh sách quán cafe, người dùng có thể xem chi tiết khi click chọn |
| 2 | *Trường tìm kiếm* | Edit text | Tìm kiếm nhà hàng theo tên |

#### Thiết kế giao diện màn hình danh sách quán ăn



|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Danh sách quán ăn* | List view | Hiển thị danh sách quán ăn, người dùng có thể xem chi tiết khi click chọn |
| 2 | *Trường tìm kiếm* | Edit text | Tìm kiếm quán ăn theo tên |

#### Thiết kế giao diện màn hình danh sách cửa hàng



|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Danh sách cửa hàng* | List view | Hiển thị danh sách cửa hàng người dùng có thể xem chi tiết khi click chọn |
| 2 | *Trường tìm kiếm* | Edit text | Tìm kiếm cửa hàng theo tên |

#### Thiết kế giao diện màn hình danh sách địa điểm nghỉ dưỡng



|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Danh sách địa điểm nghỉ dưỡng* | List view | Hiển thị danh sách địa điểm nghỉ dưỡng, người dùng có thể xem chi tiết khi click chọn |
| 2 | *Trường tìm kiếm* | Edit text | Tìm kiếm địa điểm nghỉ dưỡng theo tên |

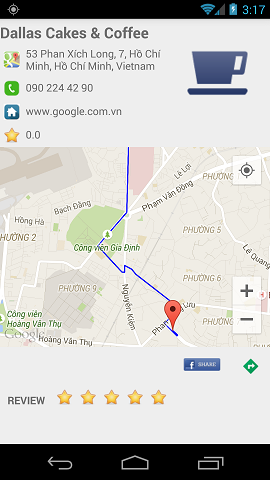
#### Thiết kế giao diện màn hình danh sách địa điểm công cộng



|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Danh sách địa điểm công cộng* | List view | Hiển thị danh sách địa điểm công cộng, người dùng có thể xem chi tiết khi click chọn |
| 2 | *Trường tìm kiếm* | Edit text | Tìm kiếm địa điểm công cộng theo tên |

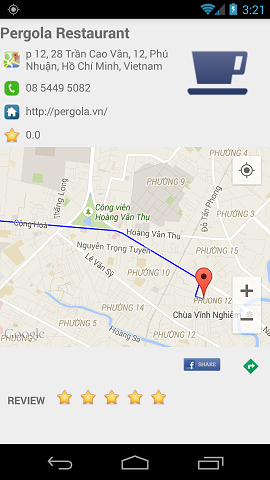
* + 1. Thiết kế giao diện màn hình chi tiết

#### Thiết kế giao diện màn hình chi tiết quán cafe



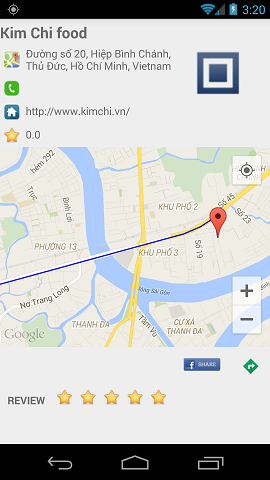
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Tên* | Textview | Hiển thị tên quán cafe |
| 2 | *Địa chỉ* | Textview | Hiển thị địa chỉ quán cafe |
| 3 | *Số điện thoại* | Textview | Hiện thị số điện thoại liên lạc |
| 4 | *Website* | Textview | Hiển thị website |
| 5 | *Đánh giá* | Textview | Hiển thị đánh giá của người dùng |
| 6 | *Hình ảnh* | Imageview | Hiển thị hình ảnh của quán |
| 7 | *Bản đồ* | Google map | Hiển thị địa điểm trên bản đồ, chỉ đường |
| 8 | *Chia sẻ facebook* | Button | Chia sẻ cho bạn bè, người thân qua facebook |
| 9 | *Chỉ đường* | Button | Chỉ đường từ địa điểm hiện tại |

#### Thiết kế giao diện màn hình chi tiết nhà hàng



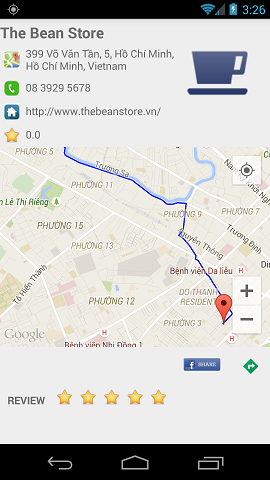
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Tên* | Textview | Hiển thị tên nhà hàng |
| 2 | *Địa chỉ* | Textview | Hiển thị địa chỉ nhà hàng |
| 3 | *Số điện thoại* | Textview | Hiện thị số điện thoại liên lạc |
| 4 | *Website* | Textview | Hiển thị website |
| 5 | *Đánh giá* | Textview | Hiển thị đánh giá của người dùng |
| 6 | *Hình ảnh* | Imageview | Hiển thị hình ảnh của nhà hàng |
| 7 | *Bản đồ* | Google map | Hiển thị địa điểm trên bản đồ, chỉ đường |
| 8 | *Chia sẻ facebook* | Button | Chia sẻ cho bạn bè, người thân qua facebook |
| 9 | *Chỉ đường* | Button | Chỉ đường từ địa điểm hiện tại |

#### Thiết kế giao diện màn hình chi tiết quán ăn



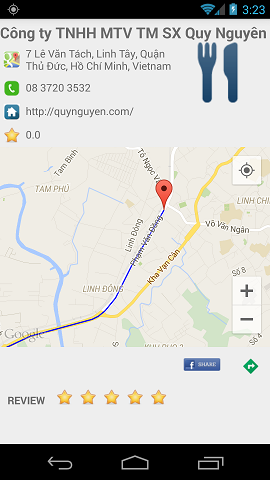
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Tên* | Textview | Hiển thị tên quán ăn |
| 2 | *Địa chỉ* | Textview | Hiển thị địa chỉ quán ăn |
| 3 | *Số điện thoại* | Textview | Hiện thị số điện thoại liên lạc |
| 4 | *Website* | Textview | Hiển thị website |
| 5 | *Đánh giá* | Textview | Hiển thị đánh giá của người dùng |
| 6 | *Hình ảnh* | Imageview | Hiển thị hình ảnh của quán ăn |
| 7 | *Bản đồ* | Google map | Hiển thị địa điểm trên bản đồ, chỉ đường |
| 8 | *Chia sẻ facebook* | Button | Chia sẻ cho bạn bè, người thân qua facebook |
| 9 | *Chỉ đường* | Button | Chỉ đường từ địa điểm hiện tại |

#### Thiết kế giao diện màn hình chi tiết cửa hàng



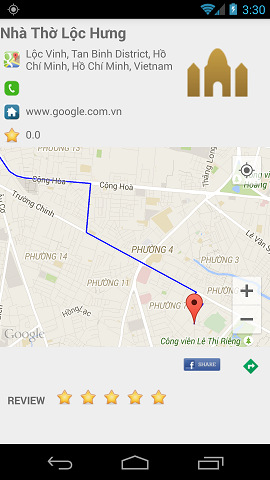
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Tên* | Textview | Hiển thị tên cửa hàng |
| 2 | *Địa chỉ* | Textview | Hiển thị địa chỉ cửa hàng |
| 3 | *Số điện thoại* | Textview | Hiện thị số điện thoại liên lạc |
| 4 | *Website* | Textview | Hiển thị website |
| 5 | *Đánh giá* | Textview | Hiển thị đánh giá của người dùng |
| 6 | *Hình ảnh* | Imageview | Hiển thị hình ảnh của cửa hàng |
| 7 | *Bản đồ* | Google map | Hiển thị địa điểm trên bản đồ, chỉ đường |
| 8 | *Chia sẻ facebook* | Button | Chia sẻ cho bạn bè, người thân qua facebook |
| 9 | *Chỉ đường* | Button | Chỉ đường từ địa điểm hiện tại |

#### Thiết kế giao diện màn hình chi tiết địa điểm nghỉ dưỡng



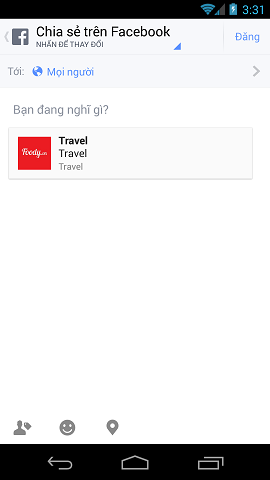
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Tên* | Textview | Hiển thị tên địa điểm nghỉ dưỡng |
| 2 | *Địa chỉ* | Textview | Hiển thị địa chỉ địa điểm nghỉ dưỡng |
| 3 | *Số điện thoại* | Textview | Hiện thị số điện thoại liên lạc |
| 4 | *Website* | Textview | Hiển thị website |
| 5 | *Đánh giá* | Textview | Hiển thị đánh giá của người dùng |
| 6 | *Hình ảnh* | Imageview | Hiển thị hình ảnh của địa điểm nghỉ dưỡng |
| 7 | *Bản đồ* | Google map | Hiển thị địa điểm trên bản đồ, chỉ đường |
| 8 | *Chia sẻ facebook* | Button | Chia sẻ cho bạn bè, người thân qua facebook |
| 9 | *Chỉ đường* | Button | Chỉ đường từ địa điểm hiện tại |

#### Thiết kế giao diện màn hình chi tiết địa điểm công cộng



|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Tên* | Textview | Hiển thị tên địa điểm công cộng |
| 2 | *Địa chỉ* | Textview | Hiển thị địa chỉ địa điểm công công |
| 3 | *Số điện thoại* | Textview | Hiện thị số điện thoại liên lạc |
| 4 | *Website* | Textview | Hiển thị website |
| 5 | *Đánh giá* | Textview | Hiển thị đánh giá của người dùng |
| 6 | *Hình ảnh* | Imageview | Hiển thị hình ảnh của địa điểm công cộng |
| 7 | *Bản đồ* | Google map | Hiển thị địa điểm trên bản đồ, chỉ đường |
| 8 | *Chia sẻ facebook* | Button | Chia sẻ cho bạn bè, người thân qua facebook |
| 9 | *Chỉ đường* | Button | Chỉ đường từ địa điểm hiện tại |

* + 1. Thiết kế màn hình chia sẻ



|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên** | **Loại** | **Ý nghĩa** |
| 1 | *Chia sẻ* | Facebook edit text | Người dùng nhập chia sẻ cho bạn bè người thân để đăng tài bằng tài khoản facebook |
| 2 | *Đăng* | Facebook button | Thực hiện thao tác đăng lên trang cá nhân của người dùng facebook |
| 3 | *Tùy chọn bảo mật* | Facebook list view | Tùy chọn đối tượng chia sẻ |
| 4 | *Đánh dấu* | Facebook list view | Đánh dấu tài khoản facebook |
| 5 | *Cảm xúc* | Facebook list view | Tùy chọn cảm xúc |
| 6 | *Địa điểm* | Facebook list view | Đánh dấu địa điểm |

# CHƯƠNG 5. KẾT LUẬN

## Kết quả đạt được

## Hướng phát triển

TÀI LIỆU THAM KHẢO

**TÀI LIỆU TIẾNG VIỆT**

[1] ………………………………………………………………………………..

**TÀI LIỆU TIẾNG ANH**

[2] ………………………………………………………………………………...

[3] ………………………………………………………………………………..

[4] ………………………………………………………………………………..

**WEBSITE TIẾNG VIỆT**

[5 ………………………………………………………………………………

**WEBSITE TIẾNG ANH**

[6] <http://www.ateam-oracle.com/performance-study-rest-vs-soap-for-mobile-applications/>

[7] ………………………………………………………………………………

[8] ………………………………………………………………………………