TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN LẬP TRÌNH WEB VÀ ỨNG DỤNG**

**ĐỀ TÀI: RESTFUL WEB SERVICE**

*Người hướng dẫn*: **Th.S DOÃN XUÂN THANH**

*Người thực hiện*: **Huỳnh Minh Thư - 52300068**

**Phan Thúy Vi - 52300084**

Khoá  **: 27**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN LẬP TRÌNH WEB VÀ ỨNG DỤNG**

**ĐỀ TÀI: RESTFUL WEB SERVICE**

*Người hướng dẫn*: **Th.S DOÃN XUÂN THANH**

*Người thực hiện*: **Huỳnh Minh Thư - 52300068**

**Phan Thúy Vi - 52300084**

Khoá  **: 27**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

LỜI CẢM ƠN

Chúng em xin gửi lời biết ơn sâu sắc đến thầy Doãn Xuân Thanh đã dành thời gian và kiến thức để hướng dẫn và hỗ trợ chúng em trong quá trình thực hiện báo cáo này. Sự động viên và sự hướng dẫn từ thầy đã giúp chúng em vượt qua những thách thức và hoàn thiện bài báo cáo một cách tốt nhất có thể. Chúng em rất trân trọng những kiến thức và kinh nghiệm mà thầy đã chia sẻ với chúng em, và chúng em sẽ luôn ghi nhớ điều đó trong suốt hành trình học tập của mình.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Chúng tôi xin cam đoan đây là sản phẩm đồ án của chúng tôi và được sự hướng dẫn của giảng viên Doãn Xuân Thanh. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Huỳnh Minh Thư*

*Phan Thúy Vi*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Restful web services là một phương pháp phổ biến cho việc phát triển các dịch vụ web linh hoạt và hiệu quả. Điều quan trọng nhất của Restful là sự đơn giản và tính nhất quán trong việc sử dụng các phương thức HTTP để thực hiện các hoạt động trên tài nguyên. Bằng cách này, việc tương tác giữa máy chủ và máy khách trở nên dễ dàng hơn, và dữ liệu được trao đổi theo cách mạnh mẽ và chuẩn mực. Với Restful, việc phát triển và tích hợp các ứng dụng trở nên linh hoạt hơn bao giờ hết, giúp tạo ra các hệ thống thông tin mạnh mẽ và hiệu quả. Bài báo cáo này sẽ làm rõ về Restful Web Service và ứng dụng của nó trong việc lập trình web dự báo thời tiết.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc17451)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc25748)

[TÓM TẮT iv](#_Toc26428)

[CHƯƠNG 1 – SƠ LƯỢC VỀ RESTFUL WEB SERVICE 2](#_Toc14605)

[1.1 Mục đích chọn đề tài 2](#_Toc26096)

[1.2 Tổng quan về Web service 2](#_Toc10544)

[1.3 Khái niệm Rest 3](#_Toc12005)

[1.4 Cơ chế hoạt động của Restful 4](#_Toc26586)

[1.5 Nguyên tắc cơ bản của Restful 4](#_Toc30059)

[1.6 So sánh REST và SOAP 5](#_Toc8029)

[CHƯƠNG 2 – TÌM HIỂU VỀ RESTFUL WEB SERVICE 8](#_Toc30150)

[2.1 Khái niệm về RESTful Web Services 8](#_Toc20796)

[2.2 Các phương thức HTTP trong RESTful Web Service 9](#_Toc20604)

[2.3 Cơ chế hoạt động 10](#_Toc21704)

[2.4 Các bước phát triển web service sử dụng Restful 12](#_Toc23787)

[CHƯƠNG 4 - RESTFUL API 14](#_Toc29656)

[4.1 Khái niệm về Restful API 14](#_Toc5571)

[4.2 Thành phần 14](#_Toc24033)

[4.3 Triển khai Restful API 14](#_Toc5652)

[CHƯƠNG 5 - ÁP DỤNG RESTFUL WEB SERVICE VÀO LẬP TRÌNH WEB DỰ BÁO THỜI TIẾT 16](#_Toc11586)

[TÀI LIỆU THAM KHẢO 19](#_Toc5139)

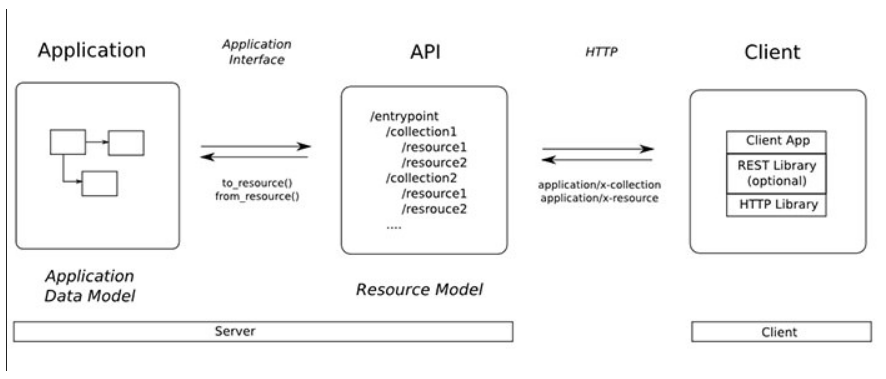
[PHỤ LỤC 20](#_Toc22402)

CHƯƠNG 1 – SƠ LƯỢC VỀ RESTFUL WEB SERVICE

* 1. Mục đích chọn đề tài

RESTful Web Services – một phương pháp mới để tạo web services sử dụng cách thức đơn giản để trao đổi dữ liệu độc lập platform với hiệu suất khá cao. Với sự tiện ích của nó sẽ giúp thực hiện web services với chức năng đơn giản sử dụng RESTful web services. Việc chọn đề tài Restful Web Service mang lại nhiều lợi ích và cơ hội nghiên cứu đa dạng. Điều này giúp nghiên cứu và hiểu sâu hơn về cơ sở lý thuyết và nguyên lý hoạt động của Restful Web Service, cũng như triển khai và thử nghiệm ứng dụng thực tế. Ngoài ra, việc nghiên cứu cũng tập trung vào việc tối ưu hóa hiệu suất, cải thiện bảo mật và an ninh thông tin, tích hợp và mở rộng hệ thống, cũng như theo dõi và áp dụng các công nghệ mới và xu hướng mới trong lĩnh vực này. Tóm lại, việc chọn đề tài Restful Web Service là cơ hội để khám phá và áp dụng kiến thức vào thực tế, đồng thời đóng góp vào sự phát triển của ngành công nghiệp công nghệ thông tin.

* 1. Tổng quan về Web service
* Cơ chế thực hiện và hoạt động của Web Services truyền thống khá phức tạp:
  + - JAX-RPC: người lập trình phải định nghĩa các interface sử dụng ngôn ngữ lập trình hay WSDL. Sau đó thực hiện cài đặt các phương thức đã được định nghĩa trong interface. Thực hiện cấu hình theo đúng chuẩn của SOA và deploy đến server có hỗ trợ web services
    - JAX-WS: người lập trình cài đặt trực tiếp các phương thức sau đó sử dụng các annotation để định ra các phương thức sẽ làm web services và dưới sử hỗ trợ của compiler và các server có hỗ trợ web services để thực hiện deploy ứng dụng
* Cơ chế thực hiện của web services đòi hỏi:
  + - Người dùng sử dụng ngôn ngữ lập trình để gọi services thông qua interface, cụ thể là WSDL, sau đó compiler thực hiện chuyển đổi và parsing lời gọi thông qua WSDL để chuyển đổi thành SOAP dưới sự hộ trợ của các API chuyển đổi tương ứng
    - SOAP được bao bọc bởi HTTP protocol để chuyển đi kết hợp với TCP/IP
    - Server nhận được HTTP thông qua port 80 sẽ trích xuất và nhận dạng SOAP thông qua HTTP
    - Thực hiện validation nội dung của SOAP, dựa trên WSDL, lần nữa thực hiện convert SOAP trở thành object cụ thể ở Server và định vị phương thức thựcthi
    - Kết quả sau khi xử lý phải thực hiện chuyển đổi thành SOAP response hay fault dựa trên WSDL vì tất cả đang sử dụng theo API của ngôn ngữ lập trình tại Server
    - Và bao bọc bởi HTTP truyền đi về client và client phải thực hiện một quá trình như server để lấy thông tin và chuyển về kết xuất của platform ở ngôn ngữ đích
* Cơ chế trên khá phức tạp đối với kết quả xử lý là dạng dữ liệu bình thường như chỉ là chuỗi đúng sai, một số nguyên gửi về sau tính toán
* Bên cạnh đó, HTTP Request và Response là object có khả năng đính kèm dữ liệu, do vậy quá trình convert SOAP là không cần thiết
* Hơn thề nữa, với định dạng đơn giản người sử dụng có thể xử lý uyển chuyển hơn so với định dạng SOAP cố định
  1. Khái niệm Rest
* REST đã nổi lên trong vài năm qua như là một mô hình thiết kế dịch vụ chiếm ưu thế. Trong thực tế, REST đã có những ảnh hưởng lớn và gần như thay thế SOAP và WSDL vì nó đơn giản và dễ sử dụng hơn rất nhiều. Với REST hệ thống không chỉ dễ mở rộng, dễ bảo trì mà còn linh động trong việc cung cấp các kiểu ứng dụng với các dữ liệu khác nhau. Sau đây khái niệm Rest:
* **R**EST là gì? REST là viết tắt của representational State Transfer (dịch nôn na là chuyển trạng thái đại diện) là một kiểu kiến trúc lập trình, nó định nghĩa các quy tắc để thiết kết các web service chú trọng vào tài nguyên hệ thống. Trong kiến trúc REST mọi thứ đều được coi là tài nguyên, chúng có thể là: tệp văn bản, ảnh, trang html, video, hoặc dữ liệu động… REST server cung cấp quyền truy cập vào các tài nguyên, REST client truy cập và thay đổi các tài nguyên đó. Ở đây các tài nguyên được định danh dựa vào URI, REST sử dụng một vài đại diện để biểu diễn các tài nguyên như văn bản, JSON, XML.
* Trong báo cáo sẽ đưa ra ví dụ về sử dụng Restful web service trong lập trình web dự báo thời thiết gửi yêu cầu API và trả về dữ liệu JSON.
  1. Cơ chế hoạt động của Restful



*Hình 1. Mô tả cơ chế hoạt động của RESTFUL*

* REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.
* GET (SELECT): Trả về một Resource hoặc một danh sách Resource.
* POST (CREATE): Tạo mới một Resource.
* PUT (UPDATE): Cập nhật thông tin cho Resource.
* DELETE (DELETE): Xoá một Resource.

Những phương thức hay hoạt động này thường được gọi là **CRUD** tương ứng với Create ( Tạo), Read (Đọc), Update (Cập nhật), Delete( Xóa)

* 1. Nguyên tắc cơ bản của Restful

REST dựa trên 6 nguyên tắc cơ bản, chúng ta tìm hiểu lần lượt từng nguyên tắc:

* **Client-Server**: Tách biệt giữa người dùng (client) và người cung cấp tài nguyên (server) để tăng tính linh hoạt và khả năng mở rộng.
* **Stateless:** Mỗi yêu cầu từ client tới server phải chứa tất cả thông tin cần thiết để hiểu và xử lý yêu cầu đó. Server không lưu trữ trạng thái của client giữa các yêu cầu.
* **Cacheable:** Các phản hồi từ server có thể được đánh dấu là có thể lưu trữ (cache) hoặc không, giúp cải thiện hiệu suất bằng cách giảm số lượng yêu cầu cần thiết tới server.
* **Uniform Interface:** Sử dụng một giao diện thống nhất để đơn giản hóa và giảm thiểu sự phức tạp của hệ thống.
* **Layered System**: Hệ thống có thể được tổ chức thành nhiều lớp, giúp tăng cường bảo mật và khả năng mở rộng.
* **Code on Demand (tùy chọn)**: Cho phép server gửi mã (như JavaScript) để thực thi trên client khi cần thiết.
  1. **So sánh REST và SOAP**

REST (Representational State Transfer) và SOAP (Simple Object Access Protocol) là hai phong cách khác nhau để xây dựng và sử dụng các dịch vụ web. Dưới đây là sự so sánh chi tiết giữa REST và SOAP:

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **REST** | **SOAP** |
| **Kiến trúc và Nguyên tắc Thiết kế** | Phong cách kiến trúc: REST là một phong cách kiến trúc sử dụng các phương thức HTTP (GET, POST, PUT, DELETE) để tương tác với tài nguyên.  Nguyên tắc thiết kế: RESTful services tuân theo các nguyên tắc của REST, bao gồm tính chất không trạng thái (stateless), khả năng lưu trữ bộ đệm (cacheability), và phân lớp hệ thống (layered system). | Giao thức: SOAP là một giao thức dựa trên XML để truyền dữ liệu giữa các ứng dụng thông qua các giao thức mạng như HTTP, SMTP, v.v.  Nguyên tắc thiết kế: SOAP sử dụng các thông báo XML để thực hiện các cuộc gọi thủ tục từ xa (RPC), đảm bảo tính bảo mật và đáng tin cậy thông qua các chuẩn như WS-Security, WS-ReliableMessaging. |
| **Định dạng Dữ liệu** | Hỗ trợ nhiều định dạng dữ liệu như JSON, XML, HTML, và plain text. JSON được sử dụng phổ biến nhất do tính nhẹ và dễ đọc.  Dễ sử dụng: RESTful APIs thường dễ sử dụng và thân thiện với người phát triển nhờ vào tính linh hoạt trong định dạng dữ liệu. | Sử dụng XML cho tất cả các yêu cầu và phản hồi. SOAP messages có cấu trúc phức tạp và nặng hơn.  Tính phức tạp: SOAP messages yêu cầu tuân theo một cấu trúc nghiêm ngặt và phức tạp hơn so với REST. |
| **Truyền thông và Giao thức** | Giao thức truyền thông: Chủ yếu sử dụng HTTP/HTTPS.  Định tuyến và URL: Sử dụng URL để xác định tài nguyên và phương thức HTTP để xác định thao tác.  Không trạng thái: Mỗi yêu cầu từ client đến server phải chứa tất cả thông tin cần thiết để hiểu và xử lý yêu cầu. | Giao thức truyền thông: Có thể sử dụng nhiều giao thức khác nhau như HTTP, SMTP, TCP, v.v.  WSDL: Sử dụng WSDL (Web Services Description Language) để mô tả các dịch vụ và các phương thức mà dịch vụ cung cấp.  Trạng thái: SOAP có thể hỗ trợ các yêu cầu trạng thái (stateful) thông qua WS-\* standards. |
| **Bảo mật** | Bảo mật: Dựa vào các chuẩn bảo mật của HTTP như SSL/TLS cho bảo mật truyền thông, và OAuth cho ủy quyền.  Tính đơn giản: Không có tiêu chuẩn bảo mật riêng, phụ thuộc vào cơ chế bảo mật của HTTP. | Có các tiêu chuẩn bảo mật riêng như WS-Security cho việc xác thực, mã hóa và toàn vẹn thông điệp.  Phức tạp hơn: Cung cấp các cơ chế bảo mật mạnh mẽ và phức tạp hơn so với REST. |
| **Khả năng mở rộng và Hỗ trợ** | Khả năng mở rộng: Tính linh hoạt cao, dễ mở rộng và triển khai.  Hỗ trợ: Rất phổ biến, dễ tìm thấy các công cụ và thư viện hỗ trợ cho nhiều ngôn ngữ lập trình. | Khả năng mở rộng: SOAP có khả năng mở rộng tốt thông qua việc sử dụng WS-\* standards, phù hợp cho các hệ thống yêu cầu tính phức tạp cao.  Hỗ trợ: Được hỗ trợ tốt trong các hệ thống doanh nghiệp và các ứng dụng yêu cầu độ tin cậy và bảo mật cao. |

**Kết luận:**

- REST thường được sử dụng cho các ứng dụng web và mobile do tính đơn giản, nhẹ nhàng và hiệu quả.

- SOAP thường được sử dụng trong các hệ thống doanh nghiệp và các ứng dụng yêu cầu độ bảo mật và tin cậy cao.

=> Sau khi so sánh giữa hai phương thức, ta có thể nhận ra rằng REST được xem là nhẹ, đơn giản và có hiệu suất cao hơn, trong khi SOAP hiệu suất kém hơn vì có độ phức tạp và tính bảo mật cao hơn. Vậy nên REST được sử dụng rộng rãi hơn trong lĩnh vực phát triển dịch vụ web.

CHƯƠNG 2 – TÌM HIỂU VỀ RESTFUL WEB SERVICE

2.1 Khái niệm về RESTful Web Services

* Chính khái niệm phức tạp về Web Services truyền thống, REST được đưa ra để áp dụng cách xây dựng và cơ thế thực hiện web service đơn giản hơn
* REST viết tắt của chữ Representational State Transfer – theo nghĩa cho phép người dùng truy cập tài nguyên – resource (có thể là dữ liệu trước và sau xử lý hay chức năng của ứng dụng) - của ứng dụng trên web services thông qua uri
* Thực hiện cơ chế nói chuyện giữa server và client là điểm điểm – point to point
* Dữ liệu được truyền trực tiếp trên HTTP và được truy cập thông qua HTTP theo đúng định dạng MIME mà không cần thông qua dạng SOAP – giảm bớt sự phức tạp – vì đơn giản chúng ta sẽ coi dữ liệu và chức năng như là tham số truyền hay dữ liệu đính kèm trong HTTP
* Cho phép thao tác dữ liệu dưới 4/7 phương thức được định nghĩa trong HTTP đó là GET, POST, PUT, DELETE để qui định rõ cách thao tác dữ liệu trên server, cái nào được truy cập, các nào được sửa đổi, cái nào được hiển thị và dấu ẩn …
* Các yêu cầu khi sử dụng REST đó là
* Client-Server: Client và Server phải nói chuyện sử dụng cùng interface và protocol.
* Sử dụng dạng kiến trúc layers và không lưu trữ trạng thái sau khi xử lý được response.
* Cache: response được lưu trữ tại client.
* Code on Demand: client lấy được dữ liệu trong response sau khi xử lý của server hoàn tất.
* Uniform Interface: mỗi resource được client truy cập thông qua địa chỉ duy nhất và sử dụng 4 HTTP methods được qui định.
* RESTful Web Services sử dụng HTTP để truyền hay nhận dữ liệu trực tiếp thông qua URI.
* Điều cuối cùng trong tập các ràng buộc khi thiết kế RESTful web service phải làm là định dạng dữ liệu mà ứng dụng và service trao đổi trong phụ tải request/response hoặc trong HTTP body. Cung cấp nhiều đại diện biểu diễn cho tài nguyên cho các request khác nhau. Cụ thể ở đây ta có thể sử dụng các một vài kiểu MIME thông dụng sau:
* JSON
* XML
* XHTML

Điều này cho phép các service sử dụng bởi các client viết bởi các ngôn ngữ khác nhau, chạy trên nhiều nền tảng và thiết bị khác nhau. Sử dụng các kiểu MIME cho phép client chọn dạng dữ liệu phù hợp với nó.

**2.2 Các phương thức HTTP trong RESTful Web Service**

Trong RESTful Web Service, các phương thức HTTP chính được sử dụng để thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên các tài nguyên là:

**- GET:**

**Chức năng:** Lấy thông tin của tài nguyên.

**Đặc điểm:**

An toàn (safe): Không thay đổi trạng thái của tài nguyên.

Idempotent: Thực hiện nhiều lần kết quả vẫn như nhau.

**Ví dụ:** Lấy thông tin người dùng với ID = 1

GET /users/1

**- POST:**

**Chức năng:** Tạo mới tài nguyên.

**Đặc điểm:**

Không an toàn: Có thể thay đổi trạng thái của server.

Không idempotent: Thực hiện nhiều lần có thể tạo ra nhiều tài nguyên mới.

**Ví dụ:** Tạo một người dùng mới

POST /users

Dữ liệu gửi kèm: { "name": "John", "email": "john@example.com" }

**- PUT:**

**Chức năng:** Cập nhật toàn bộ tài nguyên hoặc tạo mới nếu tài nguyên không tồn tại.

**Đặc điểm:**

Idempotent: Thực hiện nhiều lần kết quả vẫn như nhau.

**Ví dụ:** Cập nhật thông tin người dùng với ID = 1

PUT /users/1

Dữ liệu gửi kèm: { "name": "John", "email": "john\_updated@example.com" }

**- PATCH:**

**Chức năng:** Cập nhật một phần tài nguyên.

**Đặc điểm:**

Không hoàn toàn idempotent: Tuỳ thuộc vào ngữ cảnh và việc triển khai.

**Ví dụ:** Cập nhật email của người dùng với ID = 1

PATCH /users/1

Dữ liệu gửi kèm: { "email": "john\_updated@example.com" }

**- DELETE:**

**Chức năng:** Xoá tài nguyên.

**Đặc điểm:**

Idempotent: Thực hiện nhiều lần kết quả vẫn như nhau (nếu tài nguyên đã bị xoá thì không có gì thay đổi).

**Ví dụ:** Xoá người dùng với ID = 1

DELETE /users/1

**+ Các phương thức khác ít được sử dụng hơn trong RESTful Web Service:**

**- OPTIONS:**

**Chức năng:** Lấy các tùy chọn liên quan đến tài nguyên, ví dụ như các phương thức HTTP được hỗ trợ.

**Ví dụ:** Kiểm tra các phương thức hỗ trợ cho /users/1

OPTIONS /users/1

**- HEAD:**

**Chức năng:** Lấy phần header của phản hồi mà không cần phần body.

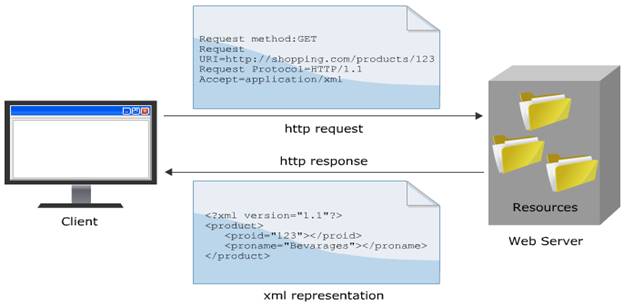
**Ví dụ:** Lấy metadata của tài nguyên /users/1

HEAD /users/1

Các phương thức này giúp RESTful Web Service có thể thực hiện đầy đủ các thao tác CRUD trên tài nguyên, tuân thủ nguyên tắc của REST và đảm bảo tính linh hoạt cũng như hiệu quả trong việc quản lý dữ liệu.

**2.3 Cơ chế hoạt động**

* Cơ chế hoạt động
* Client gửi request yêu cầu resources thông qua HTTP
* Web services dựa trên request để đáp ứng yêu cầu và response kết quả trả về
* Response được trả về theo đúng định dạng yêu cầu của client
* Client xử lý kết quả từ response để đáp ứng yêu cầu của mình



* Một số thành phần của RESTful Web Services
* Resources: có thể là chuỗi, hình ảnh, xml, tập tin …
* JAX-RS
* Cung cấp chức năng và API cho việc xây dựng ứng dụng với RESTful Web Services
* JAX-RS API
* Một API của JavaEE
* Sử dụng tạo web service với các hàm API với annotations
* Định nghĩa Java class như là một Resource class
* Định nghĩa 02 loại resources: root resource và sub resource thông qua @Path. Trong đó sub resource được truy cập thông qua root
* Root resource thông thường là được khai báo ở đầu class
* Sub resource được khai báo trên các method, nghĩa là các method được truy cập thông qua root theo đường dẫn trên server
  + - * Cú pháp của @Path
  + **@Path(“resourcePath/{param1}/…/{paramN}”)**
    - Package chứa các thành phần là javax.ws.rs
    - Các method của HTTP được hỗ trợ thông qua các khai báo annotations

|  |  |
| --- | --- |
| **Annotation** | **Mô tả** |
| @GET | Dùng cho HTTP GET method |
| @POST | Dùng cho HTTP POST method |
| @PUT | Dùng cho HTTP PUT method |
| @Delete | Dùng cho HTTP DELETE method |

*- Để trích xuất dữ liệu từ HTTP request và đưa vào services – resources xử lý:*

@PathParam annotation: Lấy giá trị từ URI trên đường dẫn sau /

@QueryParam annotation

+ Lấy  query parameter từ request tương tự như getParameter của request

+ Lấy giá trị từ URI sau dấu ? và các dấu &

@FormParam annotation

+Lấy query parameter từ request nhưng được truyền từ form parameters

*-Để xác định loại dữ liệu được xử lý trên server hay lấy về từ server:*

@Produces annotation

+ Xác định định dạng dữ liệu khi phương thức xử lý và trả về

+ Định dạng qui định là dạng có trong MIME như text/plain, text/xml, application/xml, hay text/json

+ Áp dụng cho các phương thức @GET, @POST, và @PUT

@Consumes annotation

+ Xác định loại dữ liệu được gửi từ client đến server cho xử lý trên server

+ Các định dạng hỗ trợ text/plain, text/xml, hay application/x-www-form-urlencoded (hỗ trợ dạng phương thức POST)

+ Áp dụng cho đầy đủ 4 phương thức của RESTful

**2.4 Các bước phát triển web service sử dụng Restful**

Bước 1: Tạo Web Application

* + Chọn Web Server là Tomcat
  + Chọn Java EE version  là JavaEE5

Bước 2: Đưa  Restful Web Services vào trong Web Application

* + Bổ sung các thư viện hỗ trợ như JAX-RS 1.1 và Jersey

Bước 3: Định nghĩa và cài đặt các phương thức cho Web Services

Bước 4: Sử dụng annotation để định dạng cho các services được định nghĩa trong bước 3

* + @Path host web service – root resource và sub resource
  + Loại HTTP methods dùng để truy vập
  + Lấy dữ liệu về hay cập nhật dữ liệu trên server @Produces hay @Consumes

Bước 5: Build Web Services, deploy

Bước 6: Testing Web Services, phát sinh WSDL để sử dụng ở client

**CHƯƠNG 4 - RESTFUL API**

**4.1 Khái niệm về Restful API**

- RESTful API là một ứng dụng của RESTful Web Service. API (Application Programming Interface) là một tập hợp các định nghĩa và giao thức cho phép các ứng dụng phần mềm giao tiếp với nhau. RESTful API sử dụng các nguyên tắc của REST để tạo ra các endpoint mà qua đó client có thể tương tác với server. Các RESTful API phổ biến sử dụng JSON hoặc XML để trao đổi dữ liệu.

**4.2 Thành phần**

* **API** (**A**pplication **P**rogramming **I**nterface) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML.
* **REST** (**RE**presentational **S**tate\*\* T\*\*ransfer) là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, vv đến một URL để xử lý dữ liệu.
* **RESTful API** là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile…) khác nhau giao tiếp với nhau.

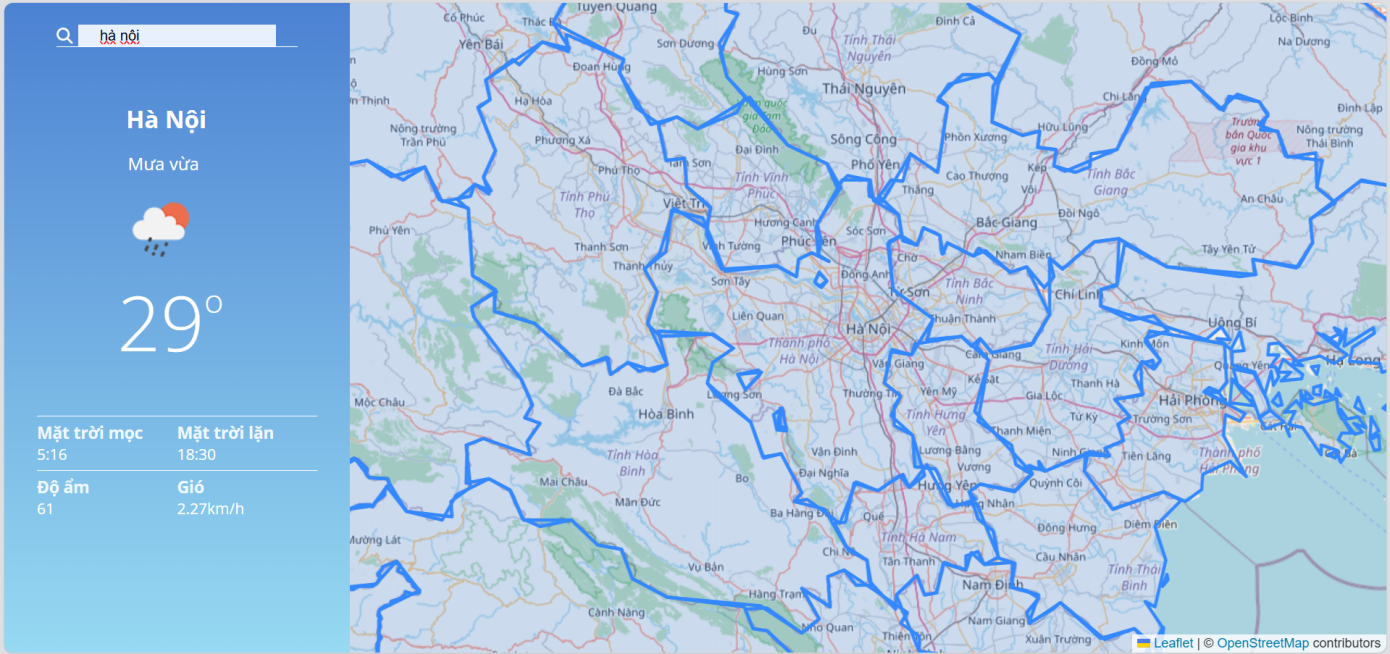
- Chức năng quan trọng nhất của **REST** là quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE…) và cách định dạng các URL cho ứng dụng web để quản các resource. **RESTful** không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một **RESTful API.**

**4.3 Triển khai Restful API**

* Xác định các tài nguyên (resource) của API: Bạn cần xác định các tài nguyên mà API của bạn sẽ cung cấp, ví dụ như danh sách sản phẩm, chi tiết sản phẩm, đơn hàng, tài khoản người dùng,...
* Thiết kế URL: Các URL trong RESTful API được sử dụng để xác định tài nguyên cần truy cập. Bạn cần thiết kế các URL để tương thích với các tài nguyên của bạn.
* Sử dụng các phương thức HTTP: RESTful API sử dụng các phương thức HTTP để thực hiện các hoạt động trên tài nguyên, bao gồm GET, POST, PUT và DELETE. Bạn cần áp dụng các phương thức này để xử lý các yêu cầu từ client.
* Định dạng dữ liệu: RESTful API sử dụng các định dạng dữ liệu chuẩn như JSON hoặc XML để trao đổi dữ liệu giữa client và server. Bạn cần quyết định định dạng dữ liệu mà API của bạn sẽ sử dụng.
* Quản lý lỗi: RESTful API cần cung cấp các mã lỗi và thông điệp lỗi rõ ràng để giúp client xử lý các trường hợp lỗi.
* Bảo mật API: RESTful API nên được bảo mật bằng cách sử dụng các biện pháp bảo mật như xác thực (authentication) và xác thực quyền (authorization) để đảm bảo chỉ các user được ủy quyền mới được truy cập vào tài nguyên.
* Tài liệu hóa API: RESTful API nên được tài liệu hóa cẩn thận để giúp người dùng hiểu rõ cách sử dụng API.

**CHƯƠNG 5 - ÁP DỤNG RESTFUL WEB SERVICE VÀO LẬP TRÌNH WEB DỰ BÁO THỜI TIẾT**

Sau khi tìm hiểu về RESTful Web Service, nhóm đã áp dụng kiến thức này để xây dựng một trang web dự báo thời tiết. Trang web cho phép người dùng truy cập thông tin thời tiết hiện tại và các thông số liên quan một cách nhanh chóng và tiện lợi.



(Hình 5.1: Ví dụ minh họa web dự báo thời tiết)

Web dự báo thời tiết sử dụng 2 RESTFUL Web Service:

- GEOCODING của openstreetmap.com để trích xuất từ tên địa điểm sang tọa độ địa điểm.

const response = await fetch(`https://nominatim.openstreetmap.org/search?format=json&polygon=1&addressdetails=1&q=${cityNameSearch}`);

Dòng code này thực hiện một yêu cầu HTTP GET đến dịch vụ Nominatim của OpenStreetMap để tìm kiếm thông tin địa lý về một thành phố cụ thể:

Gửi yêu cầu: Dòng code sử dụng `fetch` để gửi một yêu cầu đến URL `https://nominatim.openstreetmap.org/search`.

Kết quả: Dịch vụ Nominatim sẽ trả về một danh sách các kết quả phù hợp với tên thành phố tìm kiếm, bao gồm các thông tin chi tiết về địa chỉ và hình học địa lý (nếu có).

Ví dụ: Nếu `cityNameSearch` là "Hanoi", yêu cầu sẽ tìm kiếm thông tin về Hà Nội và trả về dữ liệu ở định dạng JSON, bao gồm các thông tin chi tiết như tọa độ, tên đầy đủ, và các chi tiết địa chỉ khác.



(Hình 5.2: Minh họa dữ liệu JSON trả về khi gửi yêu cầu với biến CityNameSearch = “Hà Nội”)

- Current Weather Data của openweathermap.com để trích xuất từ tọa độ địa điểm sang dữ liệu thời tiết hiện tại.

        fetch(`https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&appid=${APP\_ID}&units=metric&lang=vi`)

            .then(async res => {

                const data = await res.json();

                updateWeatherUI(data);

            })

            .catch(error => {

                console.error('Đã xảy ra lỗi khi lấy dữ liệu thời tiết:', error);

            });

Hàm này thực hiện một yêu cầu HTTP GET đến API của OpenWeatherMap để lấy dữ liệu thời tiết hiện tại cho một vị trí cụ thể dựa trên tọa độ

Ví dụ: Nếu ‘lat’ là “21.0283334” và ‘lon’ là “105.854041”, yêu cầu sẽ tìm kiếm thông tin thời tiết về tọa độ (Hà Nội) và trả về dữ liệu ở định dạng JSON, bao gồm các thông tin chi tiết như nhiệt độ, độ ẩm, tốc độ gió,v.v...



(Hình 5.3: Minh họa dữ liệu JSON trả về)

Ngoài ra nhóm còn sử dụng một số thư viện như Leaflet để hoàn thành trang web dự báo thời tiết theo ý muốn.

**TÀI LIỆU THAM KHẢO**

1. <https://viblo.asia/p/tim-hieu-ve-restful-web-service-OEqGj5JNM9bL>
2. <https://viblo.asia/p/the-nao-la-restful-web-service-gAm5yR4LKdb>
3. <https://topdev.vn/blog/restful-api-la-gi/#:~:text=REST%20(REpresentational%20State%20Transfer)%20l%C3%A0,giao%20ti%E1%BA%BFp%20gi%E1%BB%AFa%20c%C3%A1c%20m%C3%A1y.>
4. <http://www.kieutrongkhanh.net/2016/08/gioi-thieu-ve-restful-web-services-cong.html>

**PHỤ LỤC**

Phần này bao gồm những nội dung cần thiết nhằm minh họa hoặc hỗ trợ cho nội dung luận văn như số liệu, biểu mẫu, tranh ảnh. . . . nếu sử dụng những câu trả lời cho một *bảng câu hỏi thì bảng câu hỏi mẫu này phải được đưa vào phần Phụ lục ở dạng nguyên bản* đã dùng để điều tra, thăm dò ý kiến; **không được tóm tắt hoặc sửa đổi**. Các tính toán mẫu trình bày tóm tắt trong các biểu mẫu cũng cần nêu trong Phụ lục của luận văn. Phụ lục không được dày hơn phần chính của luận văn