TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI

**KHOA CÔNG NGHỆ THÔNG TIN**

A logo of a university of transport technology

Description automatically generated**-----o0o-----**

**MÔN HỌC**

**KIẾN TRÚC VÀ THIẾT KẾ PHẦN MỀM**

**Tên đề tài :**

**Tổng quan về SOA – SOAP – WSDL – WEBSERVICES - REST/RESTFUL, API – RESTFUL API**

|  |  |
| --- | --- |
| **GIẢNG VIÊN HƯỚNG DẪN:** | **NGUYỄN VĂN CƯỜNG** |
| **NHÓM SINH VIÊN SỐ 5:** | **PHAN MINH KHOA**  **NGUYỄN VĂN KHÁNH**  **NGUYỄN MINH VŨ**  **VŨ DOÃN HÀ**  **NGUYỄN MINH HIẾU**  **LƯƠNG MẠNH TIẾN** |
| **LỚP:** | **72DCHT21** |

**HÀ NỘI 2-2024**

**MỤC LỤC**

[**MỞ ĐẦU** 1](#_Toc162099088)

[**TỔNG QUAN VỀ SOA** 2](#_Toc162099089)

[**1.** **SOA** 2](#_Toc162099090)

[1.1. Lịch sử ra đời 2](#_Toc162099091)

[1.2. Lợi ích của SOA 2](#_Toc162099092)

[1.3. Ưu điểm SOA 2](#_Toc162099093)

[1.4. Nhược điểm: 3](#_Toc162099094)

[**TỔNG QUAN VỀ SOAP** 4](#_Toc162099095)

[**2.** **SOAP là gì** 4](#_Toc162099096)

[2.1. Các thành phần của SOAP 4](#_Toc162099097)

[**TỔNG QUAN VỀ WSDL** 5](#_Toc162099098)

[**3.** **WSDL là gì? mô tả và cho ví dụ** 5](#_Toc162099099)

[**TỔNG QUAN VỀ WEBSERVICES** 7](#_Toc162099100)

[**4.** **Webservices là gì? Các thành phần của webservices – các giao thức http liên quan?** 7](#_Toc162099101)

[4.1. Các thành phần của webservices 7](#_Toc162099102)

[4.2. Các giao thức HTTP phổ biến nhất liên quan đến Webservices 7](#_Toc162099103)

[**TỔNG QUAN VỀ REST/RESTFUL, API – RESTFUL API** 9](#_Toc162099104)

[**5.** **Rest/Restful** 9](#_Toc162099105)

[5.1. Kiến trúc của REST 9](#_Toc162099106)

[5.2. Các phương thức của REST 9](#_Toc162099107)

[**6.** **Api – Restful api.** 9](#_Toc162099108)

[6.1. Các thành phần của Restful api 9](#_Toc162099109)

[6.2. Lợi ích của API RESTful 10](#_Toc162099110)

[6.3. Hoạt động của API RESTful 10](#_Toc162099111)

[**KẾT LUẬN** 12](#_Toc162099112)

[**TÀI LIỆU THAM KHẢO** 13](#_Toc162099113)

**MỞ ĐẦU**

Trong thời đại số hóa ngày nay, việc phát triển các ứng dụng phần mềm đang trở nên ngày càng phức tạp và đa dạng. Để đáp ứng nhu cầu này, các nhà phát triển phải chọn lựa các kiến trúc và giao thức phù hợp nhằm xây dựng các hệ thống có khả năng tương tác, linh hoạt và dễ bảo trì. Trong bối cảnh này, Service-Oriented Architecture (SOA), SOAP/WSDL, và REST/RESTful API đã trở thành những phương pháp quan trọng trong việc phát triển các dịch vụ web.

Báo cáo này sẽ tập trung vào việc nghiên cứu, so sánh và phân tích ba trong số những phương pháp quan trọng nhất hiện nay để xây dựng và tương tác với các dịch vụ web: SOA, SOAP/WSDL, và REST/RESTful API. Mục tiêu của báo cáo là cung cấp một cái nhìn tổng quan và sâu sắc về các khái niệm cơ bản, cách thức hoạt động, ưu điểm và nhược điểm của mỗi phương pháp, từ đó giúp người đọc hiểu rõ hơn về cách chúng có thể được áp dụng trong việc phát triển các ứng dụng phần mềm hiện đại.

Bằng cách tập trung vào những điểm đặc trưng của từng phương pháp và điều tra sâu vào cách chúng ảnh hưởng đến quá trình phát triển và triển khai dịch vụ web, chúng em hy vọng rằng báo cáo sẽ cung cấp cái nhìn toàn diện và chi tiết cho các nhà phát triển, kiến trúc sư và các quản lý dự án về sự lựa chọn và triển khai các phương pháp này trong dự án của họ.

**TỔNG QUAN VỀ SOA**

1. **SOA**

SOA là viết tắt của "Service-Oriented Architecture" (Kiến trúc Dựa trên Dịch vụ). Đây là một phương pháp thiết kế phần mềm mà trong đó các ứng dụng được phát triển và triển khai dưới dạng các dịch vụ độc lập mà có thể tái sử dụng được.

Mỗi dịch vụ trong SOA thực hiện một chức năng cụ thể và có thể giao tiếp với các dịch vụ khác thông qua giao thức chuẩn như SOAP (Simple Object Access Protocol) hoặc REST (Representational State Transfer).

* 1. **Lịch sử ra đời**
* SOA (Service-Oriented Architecture) bắt đầu nảy sinh từ những năm 1990 với ý tưởng tách biệt logic kinh doanh từ triển khai kỹ thuật.
* Thập kỷ 1990 đến đầu thập kỷ 2000 chứng kiến sự xuất hiện của các công nghệ như CORBA và DCOM, nhưng chúng gặp khó khăn trong việc triển khai trên mạng Internet.
* Cuối thập kỷ 1990 đến đầu thập kỷ 2000, các nhà sản xuất công nghệ như Microsoft, IBM và Sun Microsystems (nay là Oracle) bắt đầu nghiên cứu và phát triển các tiêu chuẩn và khái niệm liên quan đến SOA.
  1. **Lợi ích của SOA**
* Giảm thiểu chi phí phát triển.
* Giảm thiểu các yêu cầu về đào tạo và kỹ năng.
* Phí bảo trì thấp.
* Chu kỳ phát triển phần mềm nhanh hơn và dễ dàng hơn.
* Định hướng kinh doanh: SOA giống như một bức tranh lớn về toàn bộ quy trình và quy trình kinh doanh của một tổ chức. Do đó, những người đầu tiên kinh doanh có thể hình dung toàn bộ quy trình được xây dựng trên quan điểm công nghệ.
  1. **Ưu điểm SOA**
* \*\*Tính linh hoạt (Flexibility)\*\*: SOA cho phép các dịch vụ được phát triển, triển khai và quản lý độc lập với nhau, giúp tăng tính linh hoạt trong việc cập nhật, mở rộng và thay đổi ứng dụng phần mềm.
* \*\*Tái sử dụng (Reusability)\*\*: SOA thúc đẩy tái sử dụng mã nguồn và dịch vụ. Các dịch vụ có thể được sử dụng lại trong nhiều ứng dụng khác nhau, giảm thiểu thời gian và chi phí phát triển.
* \*\*Tích hợp (Integration)\*\*: SOA cho phép tích hợp dễ dàng giữa các ứng dụng và hệ thống khác nhau thông qua việc truy cập các dịch vụ đã được triển khai.
* \*\*Tính tương thích (Interoperability)\*\*: SOA hỗ trợ tính tương thích giữa các hệ thống khác nhau, giúp chúng có thể giao tiếp và tương tác với nhau một cách hiệu quả.
* \*\*Tính mở rộng (Scalability)\*\*: SOA hỗ trợ việc mở rộng hệ thống một cách linh hoạt, giúp đáp ứng với nhu cầu tăng trưởng của doanh nghiệp một cách dễ dàng.
* định hướng kinh doanh

Nên sử dụng SOA vào:

* những ứng dụng phức tạp
* Tính linh hoạt và mở rộng
* Tích hợp hệ thống
* Tái sử dụng và quản lý dịch vụ
  1. **Nhược điểm:**

Khó khăn trong quản lý: SOA yêu cầu phải quản lý và triển khai nhiều dịch vụ độc lập, điều này đòi hỏi một hệ thống quản lý chặt chẽ để đảm bảo tính đáng tin cậy của hệ thống.

Độ phức tạp: SOA có thể trở nên rất phức tạp nếu không được thiết kế và triển khai đúng cách. Vì vậy, việc xác định các dịch vụ và mối quan hệ giữa chúng là cực kỳ quan trọng.

Tốc độ xử lý chậm: SOA tạo ra nhiều đặc tả với việc tách chức năng ra thành các dịch vụ độc lập, điều này có thể làm giảm tốc độ xử lý của hệ thống.

Không nên sử dụng SOA vào:

* Dự án nhỏ và đơn giản
* Yêu cầu hiệu xuất cao và thời gian thực tế
* Khả năng tích hợp đơn giản
* Yêu cầu bảo mật cao

# **TỔNG QUAN VỀ SOAP**

1. **SOAP là gì**

SOAP là viết tắt của Simple Object Access Protocol, có nghĩa là giao thức truy cập đối tượng đơn giản.

Được phát triển bởi Microsoft, IBM và một số công ty khác, SOAP được sử dụng để truyền tải [dữ liệu](https://bkhost.vn/blog/data-du-lieu/) giữa các ứng dụng web.

SOAP được phát triển để thay thế các giao thức khác như CORBA, DCOM và [RPC](https://bkhost.vn/blog/rpc-remote-procedure-call/), và nó là một phần của bộ công cụ Web Services của W3C.

SOAP sử dụng ngôn ngữ [XML](https://bkhost.vn/blog/xml-la-gi/) để truyền tải dữ liệu và các yêu cầu truy cập đối tượng. Các thông điệp SOAP có thể được truyền tải qua [HTTP](https://bkhost.vn/blog/http-va-https-la-gi/), [SMTP](https://bkhost.vn/blog/smtp-la-gi/) hoặc TCP

* 1. **Các thành phần của SOAP**
* \*\*Envelope (Bao gói)\*\*: đóng gói tất cả dữ liệu của 1 SOAP message trong 1 thông báo bằng định dạng XML.
* \*\*Header (Đầu)\*\*: Header là một phần tùy chọn của thông điệp SOAP, chứa các thông tin bổ sung về việc xử lý và giao tiếp của thông điệp, chẳng hạn như quyền hạn, mã hóa, hoặc bảo mật.
* \*\*Body (Thân)\*\*: Body chứa dữ liệu thực sự được truyền tải bởi thông điệp SOAP. Đây là nơi mà thông tin hoặc yêu cầu được đặt.
* \*\*Fault (Lỗi)\*\*: Phần lỗi chỉ ra nếu có bất kỳ lỗi nào xảy ra trong quá trình xử lý của thông điệp.

Ví dụ

A screenshot of a computer program

Description automatically generated

# **TỔNG QUAN VỀ WSDL**

1. **WSDL là gì? mô tả và cho ví dụ**

là cụm từ viết tắt của Web Services Description Language (nghĩa là ngôn ngữ mô tả định dạng Web).

Nó là file dữ liệu định dạng XML, cho phép ứng dụng hiểu về hoạt động của dịch vụ web.

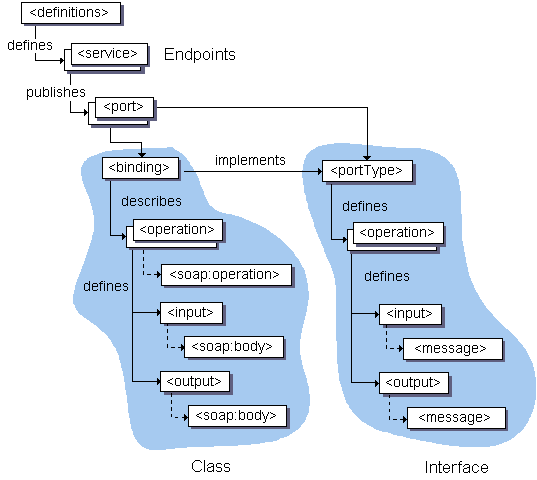
Mục đích thiết lập WSDL nhằm mô tả các chức năng mà dịch vụ web mang tới cho khách hàng. Đồng thời giúp người dùng hiểu được các thông tin kết nối, các tính năng tiêu biểu trong đó.

Tóm lại, thông qua WSDL, chúng ta xác định được vị trí của dịch vụ web mà mình hướng tới. Ngoài ra còn biết được các phương pháp, các chức năng mà dịch vụ web hiển thị.

Một tệp WSDL thường bao gồm các phần sau:

* \*\*Definitions (Định nghĩa) \*\*: Phần này định nghĩa namespace và các phần của tệp WSDL.
* \*\*Types (Kiểu)\*\*: Phần này định nghĩa các kiểu dữ liệu được sử dụng trong các thông điệp của dịch vụ web, thường được mô tả bằng cú pháp XML Schema (XSD).
* \*\*Messages (Thông điệp) \*\*: Phần này định nghĩa các thông điệp được sử dụng trong các phương thức của dịch vụ web, bao gồm các tham số và kiểu dữ liệu của chúng.
* \*\*Port Types (Kiểu Cổng) \*\*: Phần này định nghĩa các phương thức của dịch vụ web và các thông điệp liên quan đến chúng.
* \*\*Bindings (Liên kết) \*\*: Phần này liên kết các phương thức của dịch vụ web với các giao thức cụ thể như SOAP và HTTP, và xác định cách giao tiếp với dịch vụ web.
* \*\*Services (Dịch vụ) \*\*: Phần này định nghĩa các dịch vụ web cụ thể và cách truy cập chúng thông qua các giao thức cụ thể.

Ví dụ:



# **TỔNG QUAN VỀ WEBSERVICES**

1. **Webservices là gì? Các thành phần của webservices – các giao thức http liên quan?**

Web services là tập hợp các giao thức và tiêu chuẩn mở được sử dụng để trao đổi dữ liệu giữa các ứng dụng hoặc giữa các hệ thống.

Chúng ta có thể hiểu ngắn gọn:

* Là phương thức giao tiếp giữa hai thiết bị qua mạng.
* Là ứng dụng hoặc thành phần ứng dụng để giao tiếp
* Là tập hợp các tiêu chuẩn hoặc giao thức để trao đổi thông tin giữa hai thiết bị hoặc ứng dụng.

Một điểm đặc biệt của web service là có thể hoạt động độc lập, không bị phụ thuộc hay trói buộc bởi bất cứ ngôn ngữ nào.

* 1. **Các thành phần của webservices**

Gồm 3 thành phần chính:

* Simple Object Access Protocol - SOAP: Là giao thức dựa trên XML, giúp các ứng dụng có thể trao đổi với nhau thông qua giao thức HTTP hoặc HTTPs. SOAP chính là giao thức truy cập đối tượng đơn giản.
* Web Services Description Language - WSDL: Là ngôn ngữ chuẩn dựa trên XML dùng để mô tả các dịch vụ cũng như cách truy cập.
* Universal Description, Discovery & Integration - UDDI: Cũng dựa trên XML, UDDI dùng để mô tả, xuất bản, tìm kiếm các dịch vụ web.
  1. **Các giao thức HTTP phổ biến nhất liên quan đến Webservices**
* SOAP (Simple Object Access Protocol): SOAP là một giao thức dựa trên XML được sử dụng để truyền tải thông điệp giữa các ứng dụng trên mạng. SOAP thường được sử dụng trong Webservices để truyền tải các yêu cầu và phản hồi giữa client và server. Thông điệp SOAP được đóng gói trong phần thân của các gói HTTP để giao tiếp qua mạng.
* \*\*REST (Representational State Transfer) \*\*: REST không phải là một giao thức, mà là một kiến trúc hoặc phong cách thiết kế cho việc xây dựng các dịch vụ web. REST sử dụng các phương thức HTTP như GET, POST, PUT và DELETE để thực hiện các hoạt động trên các tài nguyên (resource). Các dịch vụ web RESTful sử dụng HTTP để truyền tải các yêu cầu và phản hồi giữa client và server.
* HTTP GET và POST: Cả hai phương thức này đều được sử dụng trong việc gửi yêu cầu từ client đến server và nhận phản hồi từ server trở lại client. Trong Webservices, HTTP GET thường được sử dụng để truy xuất dữ liệu từ server, trong khi HTTP POST thường được sử dụng để gửi dữ liệu đến server để xử lý.
* HTTP PUT và DELETE: Đây là hai phương thức HTTP khác cũng được sử dụng trong Webservices. HTTP PUT thường được sử dụng để cập nhật thông tin trên server, trong khi HTTP DELETE được sử dụng để xóa thông tin khỏi server.

# **TỔNG QUAN VỀ REST/RESTFUL, API – RESTFUL API**

1. **Rest/Restful**

REST (Representational State Transfer) là một kiến trúc phần mềm dựa trên web, thường được sử dụng để xây dựng các dịch vụ web có khả năng mở rộng và linh hoạt. RESTful là cách tiếp cận thiết kế các dịch vụ web sử dụng các nguyên tắc của REST.

* 1. **Kiến trúc của REST**
* Nhất quán xuyên suốt các API. Ví dụ: luôn luôn sử dụng danh từ số nhiều thay vì khi số nhiều, khi số ít
* Tồn tại mà không lưu trạng thái (Stateless existence), ví dụ như không sử dụng session của server. Đây chính là lý do vì sao lại có cụm State Transfer (ST) trong REST. Tính chất này đảm bảo rằng mỗi khi client submit 1 request, request ấy sẽ bao gồm toàn bộ các thông tin mà server cần để xử lý request. Đây là điều cần thiết để build 1 ứng dụng phục vụ lên tới hàng triệu user.
* Sử dụng HTTP status code khi có thể
* Sử dụng URL Endpoint có phân tầng logic
* Đánh version trong URL thay vì trong HTTP Headers
  1. **Các phương thức của REST**

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, Read, Update, Delete – Tạo, Đọc, Sửa, Xóa.

1. **Api – Restful api.**

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động…), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

* 1. **Các thành phần của Restful api**
* **API**(Application Programming Interface) 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](https://topdev.vn/blog/json-la-gi/) hay XML.
* **RESTful API** là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động…), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.
* **REST** (REpresentational State Transfer) 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.
  1. **Lợi ích của API RESTful**
* **Khả năng thay đổi quy mô**: Các hệ thống triển khai API REST có thể thay đổi quy mô một cách hiệu quả vì REST tối ưu hóa các tương tác giữa client và máy chủ. Tình trạng phi trạng thái loại bỏ tải của máy chủ vì máy chủ không phải giữ lại thông tin yêu cầu của client trong quá khứ. Việc lưu bộ nhớ đệm được quản lý tốt sẽ loại bỏ một phần hoặc hoàn toàn một số tương tác giữa client và máy chủ. Tất cả các tính năng này hỗ trợ khả năng thay đổi quy mô mà không gây ra tắc nghẽn giao tiếp làm giảm hiệu suất.
* **Sư linh hoạt**: Các dịch vụ web RESTful hỗ trợ phân tách hoàn toàn giữa client và máy chủ. Các dịch vụ này đơn giản hóa và tách riêng các thành phần máy chủ khác nhau để mỗi phần có thể phát triển độc lập. Các thay đổi ở nền tảng hoặc công nghệ tại ứng dụng máy chủ không ảnh hưởng đến ứng dụng client. Khả năng phân lớp các chức năng ứng dụng làm tăng tính linh hoạt hơn nữa. Ví dụ: các nhà phát triển có thể thực hiện các thay đổi đối với lớp cơ sở dữ liệu mà không cần viết lại logic ứng dụng.
* **Sư độc lập**: Các API REST không phụ thuộc vào công nghệ được sử dụng. Bạn có thể viết cả ứng dụng client và máy chủ bằng nhiều ngôn ngữ lập trình khác nhau mà không ảnh hưởng đến thiết kế API. Bạn cũng có thể thay đổi công nghệ cơ sở ở hai phía mà không ảnh hưởng đến giao tiếp.
  1. **Hoạt động của API RESTful**

Chức năng cơ bản của API RESTful cũng giống như việc duyệt Internet. Client liên hệ với máy chủ bằng cách sử dụng API khi yêu cầu tài nguyên. Các nhà phát triển API giải thích cách client nên sử dụng API REST trong tài liệu về API ứng dụng máy chủ. Đây là các bước chung cho bất kỳ lệnh gọi API REST nào:

Client gửi một yêu cầu đến máy chủ. Client làm theo tài liệu API để định dạng yêu cầu theo cách mà máy chủ hiểu được.

Máy chủ xác thực và xác nhận máy khách có quyền đưa ra yêu cầu đó.

Máy chủ nhận yêu cầu và xử lý trong nội bộ.

Máy chủ trả về một phản hồi đến client. Phản hồi chứa thông tin cho client biết liệu yêu cầu có thành công hay không. Phản hồi cũng bao gồm bất kỳ thông tin nào mà client yêu cầu.

A diagram of a software application

Description automatically generated

# **KẾT LUẬN**

Trong quá trình nghiên cứu và phân tích các phương pháp xây dựng và tương tác với dịch vụ web, chúng tôi đã thấy rằng Service-Oriented Architecture (SOA), SOAP/WSDL và REST/RESTful API đều có những ưu điểm và nhược điểm riêng, phù hợp cho các tình huống và mục tiêu cụ thể của các dự án phần mềm.

- SOA mang lại sự linh hoạt và tái sử dụng thông qua việc chia nhỏ ứng dụng thành các dịch vụ độc lập, tuy nhiên, việc triển khai và quản lý có thể phức tạp hơn do tính phân tán của các dịch vụ.

- SOAP/WSDL cung cấp độ tin cậy cao và mô tả rõ ràng về các dịch vụ, giúp việc tương tác với dịch vụ trở nên đơn giản. Tuy nhiên, đôi khi giao thức SOAP có thể gây ra overhead lớn và không linh hoạt như REST.

- REST/RESTful API mang lại tính đơn giản, linh hoạt và hiệu suất cao, phù hợp với các ứng dụng cần sự mở rộng và tương tác nhanh chóng. Tuy nhiên, việc thiết kế và quản lý các API có thể đòi hỏi sự cẩn trọng để tránh mất kiểm soát và khó khăn trong việc duy trì.

Dù mỗi phương pháp đều có điểm mạnh và điểm yếu riêng, quan trọng nhất là lựa chọn phương pháp phù hợp với yêu cầu cụ thể của dự án phần mềm và mục tiêu kinh doanh. Việc hiểu rõ các nguyên tắc và cách thức hoạt động của mỗi phương pháp sẽ giúp các nhà phát triển và kiến trúc sư đưa ra quyết định chính xác và hiệu quả, từ đó tạo ra các ứng dụng phần mềm chất lượng và đáp ứng tốt nhu cầu của người dùng cuối.

Với sự phát triển liên tục của công nghệ và sự thay đổi trong yêu cầu kinh doanh, việc tiếp tục nghiên cứu và cập nhật kiến thức về các phương pháp này sẽ là chìa khóa để thành công trong việc phát triển các hệ thống phần mềm hiện đại và linh hoạt.

# **TÀI LIỆU THAM KHẢO**

1. <https://www.matbao.net/tin-tuc/soa-la-gi-tong-quan-mo-hinh-kien-truc-huong-dich-vu-134101.html>
2. <https://viblo.asia/p/khai-niem-co-ban-ve-soap-rest-va-cach-phan-biet-chung-oOVlYyoVl8W>
3. <https://bkhost.vn/blog/soap/>
4. <https://lanit.com.vn/wsdl-la-gi.html>
5. <https://topdev.vn/blog/web-service/>
6. <https://text.relipasoft.com/2017/05/co-ban-ve-rest-va-restful-api/>
7. <https://topdev.vn/blog/restful-api-la-gi/>