|  |
| --- |
| **TRƯỜNG ĐẠI HỌC HOA SEN**  **KHOA KHOA HỌC VÀ CÔNG NGHỆ**    **BÁO CÁO ĐỀ TÀI**  **CÔNG NGHỆ DỊCH VỤ WEB**  ***Tên đề tài:***  **Xây dựng ứng dụng webservice**  **Hệ thống báo mất xe**  **12/ 2011**  **Giảng viên hướng dẫn : Võ Hoàng Hải**  **Nhóm sinh viên thực hiện : Lê Nguyễn Anh Khoa 080836**    **Nguyễn Hoàng Kim Long 080844**    **Nguyễn Hồng Chương 080794** |

**TRÍCH YẾU**

Công nghệ thông tin ngày càng có một vị trí quan trọng trong đời sống hiện đại ngày nay. Một số công việc từ khó khăn, sẽ trở nên dễ dàng hơn, giải quyết công việc hiệu quả hơn với sự trợ giúp của sức mạnh công nghệ.

Trong xã hội kinh tế hiện đại, việc mất mát tài sản đã trở thành một vấn đề đáng quan tâm. Cái khó cho chúng ta là tìm cách khắc phục hậu quả, hoặc tìm lại được tài sản của mình sao cho hiệu quả và nhanh chóng nhất. Dó cũng chính là mục đích khi nhóm chúng tôi xây dựng ”**Ứng dụng Web Service cung cấp thông tin mất xe**”!

**MỤC LỤC**

[**TRÍCH YẾU** 2](#_Toc311619096)

[**MỤC LỤC** 3](#_Toc311619097)

[**CÁC THUẬT NGỮ VÀ VIẾT TẮT** 4](#_Toc311619098)

[1 Tìm Hiểu Lý thuyết 5](#_Toc311619099)

[1.1 Giới thiệu Web service & SOA 5](#_Toc311619100)

[1.1.1 Tổng quan Web service 5](#_Toc311619101)

[1.1.2 Khái quát SOA 6](#_Toc311619102)

[1.2 Các nguyên tắc thiết kế 7](#_Toc311619103)

[1.1.3 Nhóm nguyên lý thực thi 7](#_Toc311619104)

[1.1.4 Nhóm nguyên lý điều chỉnh 8](#_Toc311619105)

[1.3 Standardized Service contract 8](#_Toc311619106)

[1.4 Service Reusability 8](#_Toc311619107)

[1.5 Service Autonomy 9](#_Toc311619108)

[1.6 Service Statelessness 11](#_Toc311619109)

[1.7 Service Discoverability 12](#_Toc311619110)

[1.8 Service Loose Coupling 13](#_Toc311619111)

[1.9 Service Abstraction 14](#_Toc311619112)

[1.10 Service Composability 15](#_Toc311619113)

[2 Thiết Kế Chương Trình 16](#_Toc311619114)

[2.1 Tổng quan về ứng dụng 16](#_Toc311619115)

[2.2 Các chức năng chính trong ứng dụng 16](#_Toc311619116)

[2.3 Thiết kế CSDL 17](#_Toc311619117)

[3 Hiện thực 19](#_Toc311619118)

[3.1 Hiện thực các chức năng 19](#_Toc311619119)

[3.1.1 Phía Server 19](#_Toc311619120)

[3.1.2 Phía Client 23](#_Toc311619121)

[3.2 Một vài đoạn mã đặt biệt 25](#_Toc311619122)

[I. Kết luận 27](#_Toc311619123)

**CÁC THUẬT NGỮ**

**VÀ VIẾT TẮT**

1. **Enterprise Architecture (EA**): Kiến trúc các hệ thống thông tin của một tổ chức ( “enterprise” được hiểu như là 1 hệ thống bao gồm nhiều nghiệp vụ )
2. **BPM**: Business Process Modelling
3. **EA:** Enterprise Architecture framwork
4. **OOAD:** *Object-Oriented Analysis and Design*
5. **SOAD** :Service Orientation Analysis and Design
6. **Normalization:** là thuật ngữ chỉ đến cách tiếp cận để giảm hay khử sự dư thừa của các data entity và cấu trúc.

# Tìm Hiểu Lý thuyết

## Giới thiệu Web service & SOA

## Tổng quan Web service

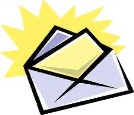
Web services là những application component được khám phá bởi sử dụng UDDI (Universal Description, Discovery and Integration), được sử dụng bởi các ứng dụng khác. Sử dụng ngôn ngữ cơ bản là XML.



**result**



**Get CustomerInfo(id)**



**request**

**Web service**

**Hình1.1 Kiến trúc của Web service**

**Web service gồm 3 thành phần chính:**

* **SOAP:** là giao thức truyền dữ liệu được xây dựng dựa trên XML, HTTP, và SMTP.
* **WSDL:** là các lược đồ XML được dùng để mô tả các chức năng, giao diện (interface) của 1 web service.
* **UDDI:** là tổ chức quản lý các services, nơi có sự tham gia của các tổ chức như IBM, Microsoft, ... tham gia cung cấp các dịch vụ.

(Xem chú thích ở mục “các thuật ngữ và từ viết tắt”)

Tuy nhiên thực tế cho thấy áp dụng những mô hình này là những hướng đi ngắn, mối quan hệ giữa các mô hình hoá này độc lập với nhau. Cần có định hướng cho các quá trình này một cách phù hợp, một cách tiếp cận mới kết hợp các yếu tố này lại với nhau🡪 định hướng SOAD.

### Khái quát SOA

**Định nghĩa SOA:**

* Theo DotNetGuru : SOA là khái niệm về hệ thống mà trong đó mỗi ứng dụng được xem là một nguồn cung cấp dịch vụ.
* SOA là tập hợp các dịch vụ kết nối 'mềm dẻo' với nhau, có giao tiếp được định nghĩa rõ ràng và độc lập với nền tảng hệ thống, và có thể tái sử dụng.
* SOA là cấp độ cao hơn của phát triển ứng dụng.
* SOA thiết kế tách riêng phần thực hiện dịch vụ và giao tiếp gọi dịch vụ

**Liên hệ giữa SOA và web service:**

* Dịch vụ web là đặc tả công nghệ
* SOA là triết lý thiết kế phần mềm
* SOA và WS có mối quan hệ tương hỗ: sự phổ biến của dịch vụ web giúp thúc đẩy sự phát triển của SOA, và kiến trúc tốt của SOA sẽ giúp dịch vụ web thành công.
* Đặc điểm chính của SOA là tách rời phần giao tiếp với phần thực hiện dịch vụ

🡪 SOA và WS trong có vẻ giống nhau

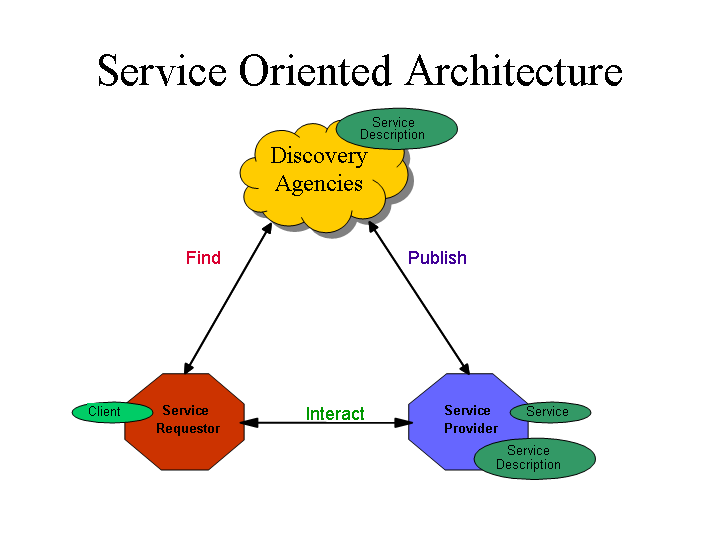
* SOA là kiến trúc phần mềm phát xuất từ:
  + Định nghĩa giao tiếp
  + Xây dựng toàn bộ mô hình ứng dụng như là mô hình các giao tiếp
* Web Service được truy cập thông qua giao tiếp, thường theo cách thức yêu cầu - đáp trả (request - response).
* Với yêu cầu dịch vụ 1 chiều thì Web Service vẫn là yêu cầu trực tiếp có chủ đích từ một phần mềm này đến một phần mềm khác

**Có 3 level chính của sự trừu tượng hoá trong SOA:**

* Operations: Các operation trong SOA tương đương với các method trong hướng đối tượng.
  + Chúng có một interface có cấu trúc cụ thể và trả về response có cấu trúc
* Services: thể hiện của một nhóm operation.
  + VD: xem thông tin customer là 1 service, thì kèm theo nó là sự kết hợp các operation: tra khách hàng theo tên, tra khách hàng theo số điện thoại, lưu dữ liệu cho khách hàng mới
* Business Process (BP) : Là tập hợp các action hay các activity thực hiện với một mục đích nghiệp vụ cụ thể. Business Process bao gồm nhiều yêu cầu service.
  + VD: Business Process : Tạo một employee mới, bán các sản phẩm hay dịch vụ, điền thông tin đơn hàng
  + Trong SOA, thì BP bao gồm một loạt các operation được thực thi tuần tự theo một tập hợp rule nghiệp vụ

🡪 Từ cái nhìn chuẩn về mô hình hoá, mang lại thách thức là bằng cách nào có những cái nhìn trừu tượng về operation, service, Business Process định rõ đặc điểm và có hệ thống.

**Tổng quan:**

****

**Hình 1.1.2**

* BPM, EA và OOAD là những nguyên tắc mô hình hoá có quan hệ độc lập với nhau
* SOA hướng tới cung cấp các giải pháp nghiệp vụ của doanh nghiệp mở rộng hay thay đổi theo nhu cầu
* Giải pháp SOA là sự kết hợp các service có khả năng sử dụng lại với các interface chuẩn, được công bố và định nghĩa tốt.
* SOA cung cấp cơ cấu tương tác các ứng dụng sẵn có mà không quan tâm đến nền tảng và ngôn ngữ của ứng dụng.

## Các nguyên tắc thiết kế

Có 8 nguyên tắc thiết kế theo hướng kiến trúc và được chi thành 2 nhóm như sau:

### Nhóm nguyên lý thực thi

* **Standardized Service Contract:** dịch vụ đưa ra một thỏa thuận thông tin liên lạc, theo quy định chung bởi một hoặc nhiều mô tả dịch vụ và tài liệu liên quan.
* **Service Reusability:** các dịch vụ sẽ được chi thành các dịch vụ con với mục đích cho việc tái sử dụng các service.
* **Service Autonomy:** các dịch vụ sẽ có các quyền tự chủ.
* **Service Statelessness:** giảm việc lưu trữ các thông tin riêng.
* **Service Discoverability:** được thiết kế theo hướng mô tả bên ngoài cho phép người dùng tìm kiếm và sử dụng dựa trên các cơ chế khám phá có sẵng.

### Nhóm nguyên lý điều chỉnh

* **Service Loose Coupling:** giảm thiểu sự phụ thuộc giữa các service với nhau.
* **Service Abstraction:** che giấu thông tin logic của service khi hiển thị thông tin ra bên ngoài.
* **Service Composability**: tập hợp các service có thể phối hợp và ghép lại với nhau.

## Standardized Service contract

Các Service contract thông thường bao gồm mô tả kỹ thuật và phi kỹ thuật. Các tài liệu thường được miêu tả như một dạng giao diện và được biết như API. Do đó các service trong kho dịch vụ phải tuân theo các chuẩn thiết kế contract giống nhau.

**Có 3 loại chuẩn hóa Service contract :**

* **Chuẩn hóa chức năng (Functional Expression Standardization):** Các operation của service khi định nghĩa cần phải áp dụng các qui tắc đặt tên chuẩn, các qui tắc này được áp dụng cho các input message, output message và tên phải phù hợp với ngữ nghĩa của chúng. Điều này nhành giúp diễn giải service contract chính xác hơn, tăng khả năng liên kết nội tại và tính sử dụng lại của service, ngoài ra còn giảm khả năng trùng lắp dữ liệu.
* **Chuẩn hóa mô hình dữ liệu (Data Model Standardization):** yêu cầu phát triển các mô hình dữ liệu phải được chuẩn hóa để tạo kiến trúc dữ liệu được chuẩn hóa nhằm tăng khả năng sử dụng lại các service trong hệ thống, nhưng điều này không yêu cầu chuẩn hóa toàn bộ data model toàn hệ thống.
* **Chuẩn hóa chính sách (Policy Standardization):** các policy mô tả giới hạn sử dụng các service, để một service có thể sử dụng lại thì các behavioral requirement cần được trình bày theo phong cách phù hợp với việc sử dụng các policy expression được chuẩn hóa dựa trên các từ vựng chuẩn và phải phù hợp với những đặc điểm của ngôn ngữ.

## Service Reusability

Là khả năng sử dụng lại của service.

**Mục tiêu:**

Giúp tăng khả năng thu hồi vốn đầu tư ban đầu, tăng tính linh hoạt trong kinh doanh bằng cách thực hiện nhanh chóng các yêu cầu nghiệp vụ trong tương lai thông qua kết hợp các service với quy mô rộng lớn, thực hiện việc hiện thực hoá các agnostic service model, cho phép tạo các kho sevice với tỷ lệ phần trăm các agnostic service cao.

**Đặc điểm thiết kế:**

* Service được xác định bởi các thuyết luôn luôn đúng của dịch vụ.
* Service có tính dùng chung cao.
* Service contract của service có tính dùng chung và có thể mở rộng.
* Các service logic có thể sử dụng đồng thời.

## Service Autonomy

Autonomy thể hiện khả năng tự điều chỉnh, được thể hiện qua sự tự do điều khiển đưa ra quyết định cho bản thân mà không cần sự chấp thuận từ phía bên ngoài dịch vụ.

Để tăng tính tự trị đòi hỏi sự thực thi chương trình phải được cô lập hơn vì vậy sẽ làm tăng mức độ độc lập, kết quả là giúp chương trình tăng độ tin cậy và khả năng dự đoán. Trong lĩnh vực phần mềm, autonomy thể hiện sự độc lập của một chương trình thực thi với xử lý của nó.

Hai lợi ích chính mà tính autonomy mang lại là tăng khả năng dự đoán và khả năng tin cậy của chương trình

**Mục tiêu:**

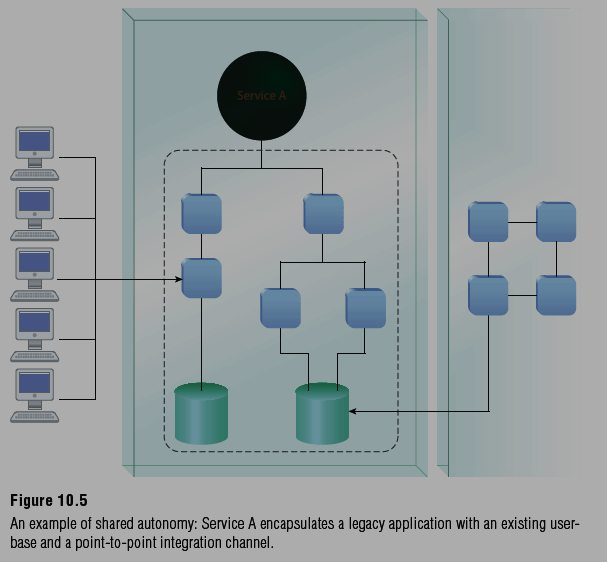
* Gia tăng độ tin cậy, sự thực thi, và khả năng dự đoán tại thời gian chạy của service, đặc biệt là khi được kết hợp và sử dụng lại.
* Tăng quyền điều khiển của một service trên môi trường thực thi của nó.

**Có 2 loại service autonomy:**

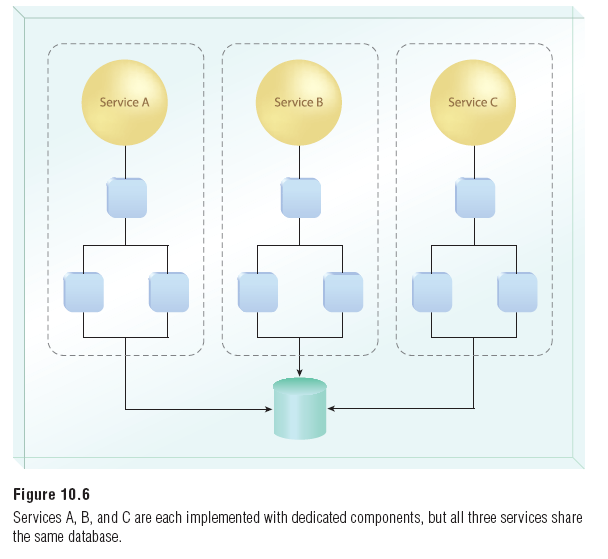
* *Runtime autonomy:* mức độ điều khiển của một service trong quá trình xử lý trong thời điểm mà service được gọi và thực thi.
* *Design-time autonomy:* mức độ tự do của chủ service để tạo sự thay đổi đối với 1 service trong thời gian tồn tại của nó.

**Đo lường khả năng tự trị của service:**

* *Service contract autonomy:* phạm vi khả năng của 1 contract không được trùng lắp với những contract khác. Điều này dễ dàng đạt được khi service contract được chuẩn hóa và tách rời với môi trường thực thi.
* *Shared autonomy.*

**

*Service logic autonomy:* là mức độ phổ biết nhất có thể đạt được khi phát triển tùy chỉnh cho một service mới.



* *Pure autonomy.*

## Service Statelessness

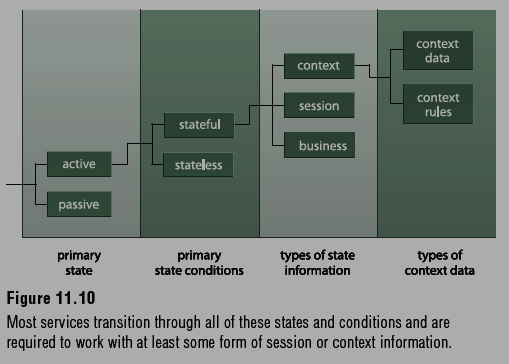
Khi sự kết hợp các service ngày càng phức tạp thì số lượng dữ liệu hoạt động cần được quản lý trong suốt thời gian sống của composition. Đề tăng khả năng mở rộng và sự thực thi trong kho service thì các service và kiến trúc cần được thiết kế hỗ trợ sự ủy nhiệm và trì hoãn các việc quản lý trạng thái.

**Mục tiêu:**

Tăng khả năng mở rộng service, hỗ trợ việc thiết kế agnostic service logic và cải tiến khả năng sử dụng lại service.

Nguyên lý thiết kế: tùy thuộc vào service model và cách tiếp cận trì hoãn được dùng mà có các loại đặc điểm thiết kế khác nhau được thực hiện.

**Các loại state:**

****

* ***Active and Passive***
  + *Qui định của service sau khi nó được gọi và thực thi bởi một chương trình sử dụng được xem như là active state.*
  + *Active service được xem xét để thực hiện các chức năng một cách tích cực trên khả năng của service khi được gọi.*
  + *Một service được load vào bộ nhớ nhưng nó không thực hiện bất kỳ chức năng nào của nó một cách tích cực thì đó là passive state.*
  + *Các service có thể chạy nhàn rỗi (passive) trước, và sau đó để gọi và thực hiện.*
  + *Passive và active là 2 trạng thái chính liên quan đến sự thực thi của nguyên Service Statelessness.*
* ***Stateless and Stateful***
  + Một service là *stateful* nếu service đó active và sử dụng thêm vùng nhớ tạm để lưu trữ tạm thời và xử lý *state data.*
  + Khi *stateful,* service sẽ đang xử lý một hoặc nhiều loại trong 3 loại dữ liệu trạng thái sau: *Session data, Context data, Business data.*
  + Một service là *stateless* nếu service đó active và không sử dụng thêm vùng nhớ tạm để lưu trữ tạm thời và xử lý *state data.*
  + Những loại dữ liệu cần được xem xét để service có thể là stateless:
    - Session data.
    - Context data.
    - Business data.
  + Một service có trở thành stateful hay stateless hay không được quyết định bởi service capability được gọi và các chức năng được kết hợp của service đó.
* **Session and Context Data**
  + ***Session data*** là những thông tin trình bày hay giữ một kết nối hay sự tương quan giữa các quá trình trao đổi message
  + ***Context data*** là thông tin thuộc về và đang xảy ra như một phần của hoạt động service hiện hành

VD:

* + - Số lượng service đang tham gia vào một hoạt động
    - Những service nào đang active trong một hoạt động của service
    - Khoảng thời gian hoạt động của service
    - Có bao nhiều instance của hoạt động đang thực thi
  + ***Context rules*** là những giao thức và những ràng buộc được áp dụng để thực thi một hoạt động của service\

VD:

* + - Khoảng thời gian cho phép của một hoạt động service
    - Số lượng các instance cho phép của một hoạt động của service
    - Số lượng các service được phép tham gia
  + ***Business data*** miêu tả những thông tin liên quan đến business task đang thực thi.
    - Business data được vận chuyển trong SOAP body

## Service Discoverability

Khả năng khám phá, giải thích và trình bày service.

Điều này sẽ giúp cho xác định service đã tồn tại hay cần phải xây dựng mới. Để làm được điều này đòi hỏi thông tin mô tả service phải chính xác thể hiện đầy đủ khả năng của service, nếu mô tả không đầy đủ thì người dùng sẽ mất cơ hội sử dụng lại service điều này dẫn đến sự trùng lắp với các service đã tồn tại.

**Mục tiêu:**

Các service được đặt ở vị trí như các resource có khả năng khám phá cao, mục đích và khả năng của service được trình bày rõ ràng bởi con người cũng như chương trình phần mềm.

**Đặc điểm thiết kế:**

* Trang bị dữ liệu phù hợp.
* Trang bị thêm những thông tin mô tả với mục đích sử dụng của chúng và những khả năng với con người.
* Nếu một service đã được đăng ký thì các bản ghi đăng ký được phổ biến là những meta information.
* Nếu một service không được đăng ký thì các tài liệu sơ lược về service được thiết kế bổ sung cho Service contract và đó là dạng cơ bản cho các bản ghi đăng ký trong tương lai.
* Khả năng khám phá và trình bày dịch vụ.
* Cải thiện sự truyền chất lượng thông tin của dữ liệu mô tả dịch vụ.
* Những loại khám phá:
* Design-time discovery: tiến trình khám phá được thực hiện một cách thủ công bởi con người.
* Runtime discovery: service có khả năng đưa ra những truy vấn discovery động và kết quả là một quá trình tự động hóa.

## Service Loose Coupling

Service loose coupling nhằm giảm sự phụ thuộc vào các service khác mà bản thân service sử dụng, cũng như giảm sự phụ thuộc vào chương trình sử dụng bản thân service.

**Mục tiêu:**

Giảm sự phụ thuộc giữa các service, tăng sự thực thi độc lập.

**Đặc điểm thiết kế:**

Sự tồn tại của một service contract phải tách rời về công nghệ và sự thực thi. Không phụ thuộc vào xử lý bên ngoài, giảm tối thiểu các yêu cầu có sự gắn kết với consumer.

**Các loại service contract coupling:**

* Logic to contact coupling: phạm vi của 1 service được giới hạn bởi contract của nó, đây là dạng liên kết mang tính tích cự.
* Contract to logic coupling: phạm vi của 1 service phụ thuộc vào logic bên dưới, đây là loại liên kết mang tính tích cực vì các chương trình service consumer tạo ra sụ ràng buộc với service contract.
* Contract to technology coupling: là sự phụ thuộc vào công nghệ giao tiếp độc quyền. Loại liên kết này không mang tính tích cực vì khi service phụ thuộc vào công nghệ sẽ dẫn đến consumer cũng sẽ phụ thuộc vào công nghệ.
* Contract to implement coupling: là sự phụ thuộc vào môi trường thực thi, đây cũng là loại liên kết không mang tính tích cực.
* Contract to functional coupling: là sự phụ thuộc của service đối với external logic.

## Service Abstraction

Các service contract chỉ chứa các thông tin cần thiết và các thông tin này sẽ giới hạn những thông tin gì sẽ được publish trong các service contract.

**Mục tiêu:**

Giữ cho số lượng và nội dung thông tin súc tích, cân bằng và ngăn ngừa sự truy cập không cho phép.

**Đặc điểm thiết kế**: thông tin sẽ được trừu tượng về công nghệ, logic vác các chức năng bên ngoài dịch vụ, thông tin về một service sẽ được điều khiển hoặc ẩn trong một môi trường cụ thể.

**Có 4 loại Meta abstraction:**

* Technology information: mô tả thực thi về kỹ thuật bên dưới. Thông tin chi tiết về kỹ thuật sẽ được ẩn đi để tạo sự tự do về thay đổi kỹ thuật mà không ảnh hưởng đến người sử dụng service.
* Function information: metadata sẽ mô tả service có những chức năng nào.
* Programatic logic information: mô tả cách thức dịch vụ thực hiện khả năng của nó. Được áp dụng ở những mức thiết kế chi tiết thấp như thuật toán, xử lý lỗi, ...
* Quality of service information: metadata mô tả service behavior, các giới hạn và yêu cầu tương tác. Chất lượng của service data là một giới hạn của vùng thông tin liên quan đến độ tin cậy.

## Service Composability

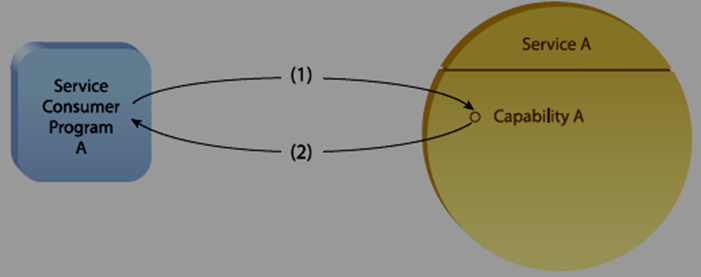
Khả năng kết hợp các service lại với nhau.

**Mục tiêu:**

Tất cả các mục đích của service reusability có thể áp dụng, và là một dạng sử dụng lại dịch vụ.

**Các khái niệm và thuật ngữ tích hợp:**

* **Composition.**
* **Composition Instance:** *Composition instance* miêu tả những gì thực sự xảy ra, khi xảy ra workflow logic được thực hiện bởi tập hợp các instance ở thời điểm runtime.
* **Composition member:** Phân loại theo vai trò được chấp nhận là tạm thời, phân loại theo khả năng được xem có tính bền vững hơn. Vì vai trò là *composition member* có thể chuyển thành *composition controller* nếu sau này *Capability* này kết hợp với các service khác
* **Composition controller.**
* **Composition sub-controller.**
* **Composition member capability.**
* **Composition controller capability.**
* **Designated controller.**
* **Collective composability:** collective measure of composability của tất cả các service thành phần của composition được quyết định bởi chất lượng của service được đưa ra bởi sự điều khiển của composition.
* **Service activity.**

****

Một ví dụ về hoạt động dịch vụ. Chương trình người dùng tương tác với capability A của các dịch vụ A để thực hiện một liên kết *Point-to-Point* cho việc trao đổi dữ liệu

* **Point-to-point.**
* **Primitive composition.**
* **Complex composition.**
* **Composition initiator:** để kích hoạt một *service composition* thì cần chương trình sử dụng service nằm bên ngoài giới hạn gọi *composition controller*. Chương trình này gọi là *Composition Initiator*

# Thiết Kế Chương Trình

## Tổng quan về ứng dụng

Đồ án sẽ mô phỏng hệ thống cớ mất xe trên mô hình Client – Server, Server cung cấp các Service cho các Client kết nối và sử dụng các Service đó. Bên Server sẽ hợp tác với phía công an, nếu bên công an có thông tin về một chiếc xe bị mất sẽ tương tác với Server, nếu chiếc xe đó có lưu trong Server thì thông tin về chiếc xe đó sẽ được cập nhật vào CSDL và chủ xe bị mất sẽ được thông báo thông qua các thông tin cá nhân liên lạc đã được cung cấp trước đó.

**Mục tiêu của đồ án không phải là nghiên cứu nghiệp vụ của một hệ thống hoàn chỉnh mà tìm hiểu phương thức hoạt động, vận hành, tương tác dữ liệu** bao gồm những phần:

* Tìm hiểu các Service kết nối với dữ liệu SQL.
* Tìm hiểu cách Service tổ chức các ServiceContract, OperationContract, DataContract.
* Tìm hiểu cách Service truyền dữ liệu đến Client và ngược lại.
* Tìm hiểu cách Client kết nối đến Service.
* Nghiên cứu cách áp dụng mô hình 3 lớp cũng như các mẫu thiết kế từ windows form application lên Service và Client.

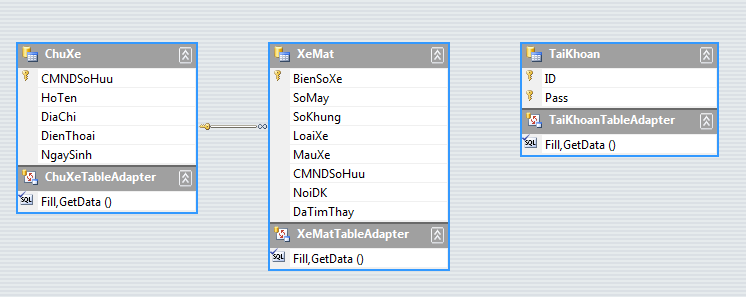
## Các chức năng chính trong ứng dụng

Chính vì nghiên cứu cách vận hình của Web Service là chính chương trình chỉ thực hiện một số tính năng cơ bản và xoáy mạnh vào các nghiệp vụ Web Service, các tính năng cơ bản đó bao gồm:

* Hỗ trợ server tạo các Endpoint cho client kết nối đến.
* Hỗ trợ client kết nối đến server để sử dụng dịch vụ.
* Hỗ trợ người dùng phía client báo mất xe.
* Hỗ trợ người dùng phía client xem thông tin xe của mình đã tìm thấy hay chưa.
* Hỗ trợ người dùng phía server truy cập vào hệ thống dữ liệu của server.
* Hỗ trợ người dùng phía server tạo tài khoản truy cập server.

## Thiết kế CSDL

Về cơ sở dữ liệu, hệ thống sẽ sử dụng file Database trực tiếp trên Visual Studio 2010 mà không sử dụng Mircosoft SQL Server. Việc làm này giúp cho việc cài đặt của người sử dụng trở nên rất dễ dàng vì việc cài đặt SQL và attach database là công việc rất khó khăn nhất là đối với những người không biết có chuyên môn công nghệ thông tin. Khi Build Solution, Visual studio 2010 sẽ tự động sinh ra một cơ sở dữ liệu cùng với file thực thi của chương trình, chương trình sẽ tự động kết nối với cơ sở dữ liệu này mà không cần thông qua bất cứ bước cài đặt khác nào. Riêng đối với người phát triển, việc sử dụng trực tiếp cơ sở dữ liệu SQL trên Visual giúp họ chỉ cần tương tác trên một cửa sổ mà không cần phải mở them SQL Server.



**Bảng:**

Bảng ChuXe là bảng dùng để lưu thông tin cá nhân của người cớ mất xe.

|  |  |  |
| --- | --- | --- |
| ChuXe | | |
| CMNDSoHuu | nchar(10) | Chứng minh nhân dân của chủ xe |
| HoTen | nchar(100) | Họ tên |
| DiaChi | nchar(100) | Địa Chỉ |
| DienThoai | nchar(100) | Điện thoại |
| NgaySinh | DateTime | Ngày sinh |

Bảng XeMat là bảng dùng để lưu thông tin của chiếc xe được cớ mất.

|  |  |  |
| --- | --- | --- |
| XeMat | | |
| BienSoXe | nchar(100) | Biển Số Xe |
| SoMay | nchar(100) | Số Máy |
| SoKhung | nchar(100) | Số Khung |
| LoaiXe | nchar(100) | Loại Xe |
| MauXe | nchar(100) | Màu Xe |
| CMNDSoHuu | nchar(10) | Chứng minh nhân dân của chủ xe |
| NoiDangKy | nchar(100) | Nơi đăng ký xe |
| DaTimThay | bit | Đã tìm thấy hay chưa |

Bảng TaiKhoan là bảng dùng để lưu các tài khoản của admin có khả năng đăng nhập vào hệ thống phía server.

|  |  |  |
| --- | --- | --- |
| TaiKhoan | | |
| ID | nchar(100) | ID |
| Pass | nchar(100) | Mật khẩu |

**Và các Store Procedures để truy xuất dữ liệu như:**

SP\_CheckDaTimThay: Thực hiện sửa trạng thái DaTimThay của đối tượng XeMat từ False sang True.

SP\_Loggin: Thực hiện kiểm tra tài khoản .

SP\_TaoTaiKhoan: Thực hiện việc tạo tài khoản.

SP\_ThemXe: Lưu các thông tin người dùng nhập, bao gồm thông tin cá nhân và thông tin chiếc xe bị mất vào CSDL.

SP\_TimChuXeTheoCMND: Tìm thông tin chủ xe dựa trên CMND.

SP\_TimXeTheoCMND: Tìm thông tin xe theo CMND.

SP\_TimXeTheoBienSo: Tìm thông tin xe theo biển số xe.

SP\_XemXeDaTimThay: Liệt kê những chiếc xe đã được tìm thấy.

SP\_XemXeChuaTimThay: Liệt kê những chiếc xe chưa được tìm thấy.

# Hiện thực

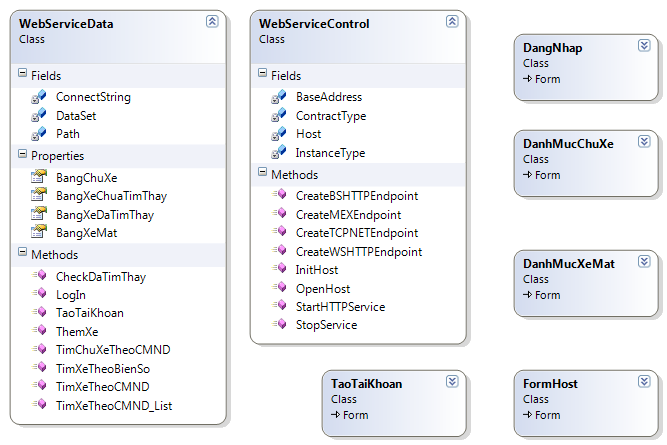
## Hiện thực các chức năng

### Phía Server

#### Cấu Trúc:

Chương trình được thực hiện theo mô hình 3 lớp, mỗi một lớp sẽ thực hiện 1 chức năng riêng biệt:

* Lớp Interface (các window form): chứa các windows form thực hiện tương tác với người dùng, tiếp nhận thông tin input từ người dùng và hiển thị các thông tin output chuyển đến người dùng.
* Lớp Control ( WebServiceControl.cs): thực hiện các chức năng điều khiển hệ thống, ở đây lớp control sẽ thực hiện tạo host, tạo các endpoint để client kết nối đến.
* Lớp Data (WebServiceData.cs) : thực hiên tương tác dữ liệu xuống cơ sở dữ liệu SQL, là nguồn cung cấp data cho lớp Control và Interface.

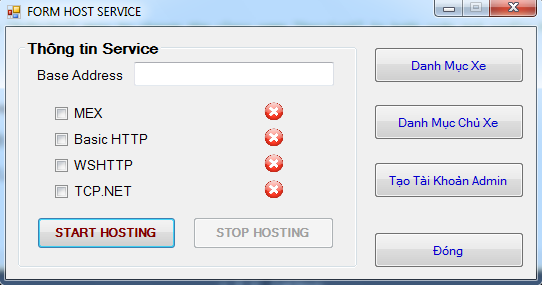
Cả 3 lớp được thiết kế sao cho có thể tương tác lẫn nhau qua các hàm static, có nghĩa là một lớp bất kì sẽ có thể truy cập, điều khiển 2 lớp còn lại, tạo sự đồng bộ và linh hoạt cho hệ thống. Bản thiết kế sẽ được mô tả trong Class Diagram và được Generate tự động thành các lớp chi tiết:

### Chức năng:

**Server sẽ có các chức năng chính đó là:**

* Seft-Host các service của mình lên một địa chỉ Host do người dùng chỉ định đồng thời tạo các Endpoint do người dùng tùy chọn để Client kết nối đến.
* Cho phép xem và chỉnh sửa trực tiếp lên Database của hệ thống.
* Tạo tài khoản đăng nhập vào hệ thống.

1. **Self-Host các Service**



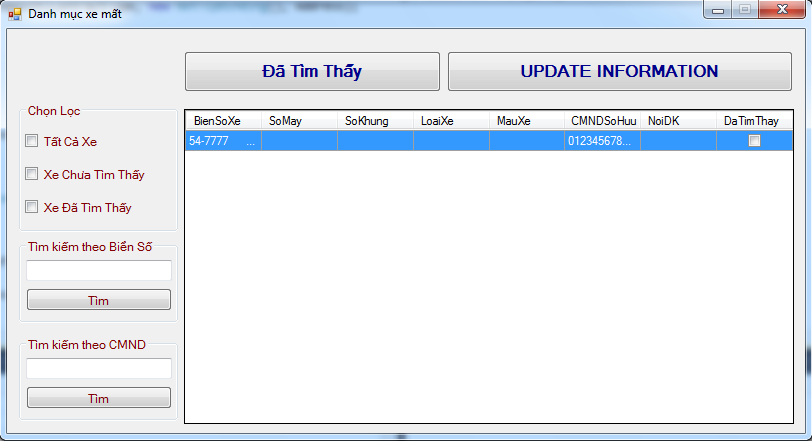
Người dùng sẽ nhập địa chỉ host vào ô base address và chọn các kiểu binding mà mình muốn, chương trình sẽ cung cấp 3 kiểu binding cơ bản đó là Basic HTTP, WS HTTP và TCP.NET.

Nếu người dùng muốn tạo Mex Endpoint thì chọn check box MEX.

Sau khi người dùng tùy chọn xong có thể ấn nút START HOSTING để host service lên địa chỉ đã chỉ định. Tùy theo tùy chọn của họ mà FormHost sẽ gọi các hàm tương ứng trong lớp WebServiceControl để tạo các Endpoint.

1. **Xem và chỉnh sửa Database**

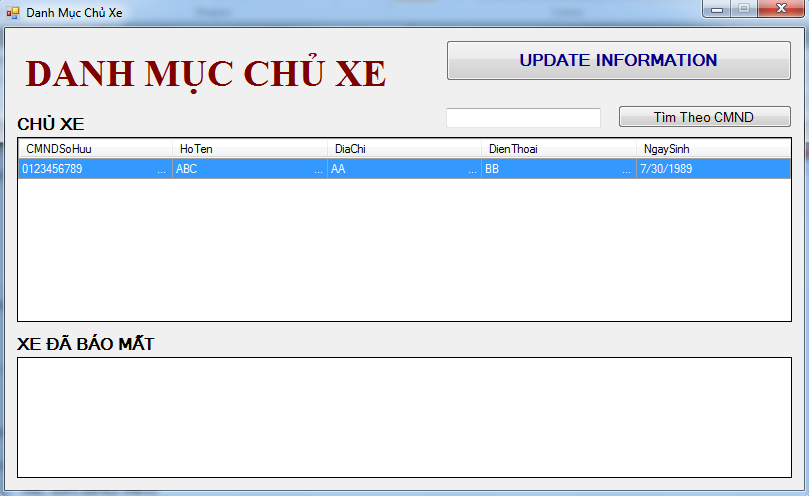
Người dùng có thể xem các danh mục trong CSDL thông qua các gridview bằng cách chọn Danh Mục Xe hoặc Danh Mục Chủ Xe.



Trong danh mục xe mất, người dùng sẽ đuợc cung cấp các chức năng chọn lọc tùy theo thuộc tính của các xe ( như hình trên ).

Khi người dùng muốn cập nhật rằng một chiếc xe nào đó đã được tìm thấy, họ có thể chọn chiếc xe đó và click vào nút Đã Tìm Thấy.

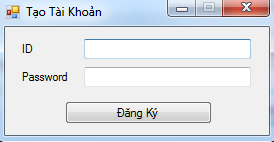
Khi người dùng muốn cập nhật data cho GridView họ có thể click vào nút Update Information.



Tương tự như Danh mục xe mất, danh mục chủ xe cũng cho phép người dùng tìm thông tin chủ xe theo CMND và cho phép họ update data cho Gridview bằng cách nhấn nút Update Information.

Khi người dùng click vào một người chủ sỡ hữu nào đó, Gridview phía dưới sẽ hiện thị thông tin chiếc xe mà người đó đã cớ mất.

1. **Tạo tài khoản đăng nhập hệ thống**



Người dùng có thể click vào Tạo tài khoản Admin từ FormHost để mở giao diện đăng ký, người dùng nhập ID và Pass muốn đăng ký, hệ thống sẽ kiểm tra và thông báo cho người dùng.

#### Các dịch vụ cung cấp:

1. Cớ mất xe

* Chức năng: Cho phép người dùng cớ mất xe khi chiếc xe của mình bị mất, người dùng sẽ nhập thông tin cá nhân, thông tin liên lạc, thông tin xe để khi server nhận được thông báo tìm thấy từ phía công an, bộ phận liên lạc sẽ liên hệ với người mất.
* Input: Thông tin cá nhân, thông tin liên lạc, thông tin xe.
* Output: true/false, nếu hệ thống đưa dữ liệu vào CSDL thành công sẽ trả về true, ngược lại trả về false.

1. Kiểm tra xe đã cớ mất

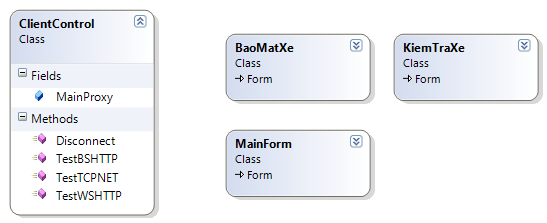
* Chức năng: Khi người dùng muốn kiểm tra xem xe mà mình cớ mất đã được tìm thấy hay chưa, họ có thể nhập vào Chứng minh nhân dân và server sẽ cung cấp thông tin về xe mà họ đã cớ mất.
* Input: Chứng minh nhân dân.
* Output: đối tượng XeMat. Server sẽ gửi đối tượng XeMat từ server đến client, đối tượng này mang toàn bộ thông tin về chiếc xe ấy.

### Phía Client

#### Cấu trúc

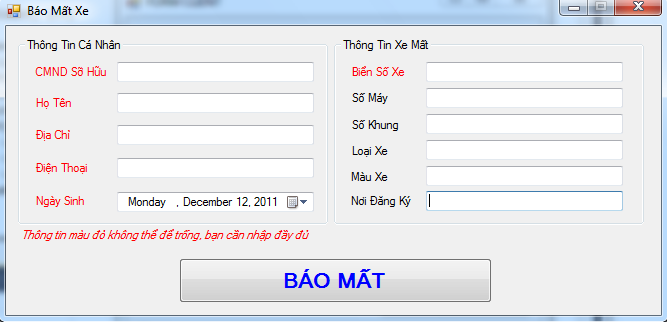
Chương trình cũng được thực hiện theo mô hình 3 lớp như Server nhưng không có lớp Data bởi vì dữ liệu được xử lý từ Server và truyền về Client, Client không cần thiết xử lý dữ liệu.

* Lớp Control ( ClientControl.cs): thực hiện các chức năng điều khiển hệ thống, ở đây lớp control sẽ thực hiện các kết nối đến Endpoint mà Server cung cấp
* Lớp Interface (các window form) : chứa các windows form thực hiện tương tác với người dùng, tiếp nhận thông tin input từ người dùng và hiển thị các thông tin output chuyển đến người dùng.



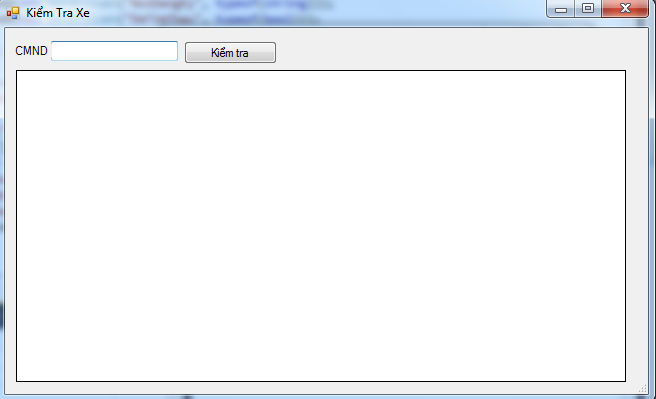
#### Chức năng

1. **Báo mất xe**



Người dùng sẽ nhập thông tin cá nhân, thông tin liên hệ, thông tin xe mất và click Báo Mất, hệ thộng sẽ gửi dữ liệu đến server và trả về kết quả cho người dùng.

1. **Kiểm tra xe đã báo**



Người dùng muốn kiểm tra xem xe mình báo mất đã tìm thấy hay chưa có thể vào form Kiểm Tra Xe và nhập vào Chứng minh nhân dân của mình, nếu đúng thì thông tin về xe mà họ báo mất sẽ được hiển thị.

## Một vài đoạn mã đặt biệt

* Để áp dụng mô hình 3 lớp linh động, các hàm và biến trong mỗi lớp sẽ là static, mỗi lớp đều có khả năng điều khiển 2 lớp còn lại:

public static System.Data.DataTable BangChuXe

{

get

{

using (DataSetOBJTableAdapters.ChuXeTableAdapter Adap = new DataSetOBJTableAdapters.ChuXeTableAdapter { Connection = new System.Data.SqlClient.SqlConnection(ConnectString) })

{

Adap.Fill(DataSet.ChuXe);

return DataSet.ChuXe;

}

}

}

**Ví dụ:**

* Người dùng muốn lấy Bảng Chu Xe từ SQL để đưa vào GridView chỉ cần nhập dòng code:

**GridView.DataSource = WebServiceData.BangChuXe;**

* Tương tự cho lớp Control, khi người dùng muốn Tạo Endpoint Basic HTTP chỉ cần nhập dòng code:

**WebServiceControl.CreateBSHTTPEndpoint();**

# KẾT LUẬN

* Qua quá trình làm luận án dưới sự hướng dẫn của thầy Võ Hoàng Hải, chúng tôi đã hoàn thành được các mục tiêu nghiên cứu được nêu ở trên, đã có những kiến thức cơ bản về phương pháp phát triển ứng dụng Web Service và hiểu được cách thức Web Service vận hành.
* Tuy nhiên đề án của chúng tôi chỉ mang tính ví dụ mà chưa có tính thực tiễn cao, chưa phù hợp với những yêu cầu thực tế ngoài đời và chúng tôi chưa có thời gian tìm hiểu những phần sâu hơn của Web Service.
* Trong tương lai chúng tôi sẽ nghiên cứu thêm các tiện ích nâng cao của Web Service và sẽ áp dụng nó vào các ứng dụng thực tiễn trong tương lai.