ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN MINH TÂN

TÍCH HỢP NGHIỆP VỤ DỰA TRÊN CÔNG NGHỆ

ESB MIDDLEWARE

LUẬN VĂN THẠC SĨ NGÀNH HỆ THỐNG THÔNG TIN

Hà Nội – 2017

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN MINH TÂN

TÍCH HỢP NGHIỆP VỤ DỰA TRÊN CÔNG NGHỆ ESB MIDDLEWARE

Ngành: Hệ thống thông tin

Chuyên ngành: Hệ thống thông tin

Mã số: 60480104

LUẬN VĂN THẠC SĨ NGÀNH HỆ THỐNG THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS.TS. Nguyễn Ngọc Hóa

Hà Nội – 2017

LỜI CẢM ƠN

Để có thể hoàn thiện được luận văn thạc sỹ của mình, trước tiên, tôi xin bày tỏ lòng biết ơn sâu nhất tới thầy – PGS.TS. Nguyễn Ngọc Hóa (bộ môn Các hệ thống thông tin – trường Đại học Công nghệ – Đại học Quốc gia Hà Nội). Sự gần gũi, khích lệ và nhiệt tình hướng dẫn của thầy là nguồn động lực rất lớn đối với tôi trong suốt thời gian thực hiện luận văn.

Tôi cũng xin được gửi lời cảm ơn chân thành nhất tới tất cả các thầy, cô trong bộ môn Các hệ thống thông tin, cũng như các thầy, cô trong khoa Công nghệ thông tin – trường Đại học Công nghệ – Đại học Quốc gia Hà Nội đã nhiệt tình giảng dạy, cung cấp cho chúng tôi những kiến thức, kinh nghiệm không chỉ trong học tập mà còn trong cuộc sống hàng ngày.

Đồng thời tôi cũng xin được gửi lời cảm ơn đến cha mẹ, người thân trong gia đình, các bạn học viên, đồng nghiệp đã giúp đỡ, động viên, tạo điều kiện tốt nhất cho tôi trong suốt khóa học tại Trường Đại học Công nghệ – Đại học Quốc gia Hà Nội để tôi có thể hoàn thiện tốt luận văn thạc sỹ của mình.

Hà Nội, tháng 11 năm 2017

Học viên

Nguyễn Minh Tân

LỜI CAM ĐOAN

Tôi xin cam đoan luận văn tốt nghiệp *"Tích hợp nghiệp vụ dựa trên công nghệ ESB Middleware"* là công trình nghiên cứu thực sự của bản thân, được thực hiện trên cơ sở nghiên cứu lý thuyết, kiến thức chuyên ngành, nghiên cứu khảo sát tình hình thực tiễn và dưới sự hướng dẫn khoa học của PGS.TS. Nguyễn Ngọc Hóa. Các kết quả được viết chung với các tác giả khác đều được sự đồng ý của tác giả trước khi đưa vào luận văn. Những phần tham chiếu, trích dẫn trong luận văn đều được nêu rõ trong phần tài liệu tham khảo. Các số liệu, kết quả trình bày trong luận văn là hoàn toàn trung thực. Nếu sai tôi xin chịu hoàn toàn trách nhiệm và chịu mọi kỷ luật của khoa và nhà trường đề ra.

Hà Nội, tháng 11 năm 2017

Học viên

Nguyễn Minh Tân

MỤC LỤC

DANH MỤC CÁC HÌNH

DANH MỤC CÁC TỪ VIẾT TẮT

CSDL	Cơ sở dữ liệu
ESB	Enterprise Service Bus
ECM	Enterprise Content Managerment
MOM	Message – Oriented Middleware
RPC	Remote Procedure Call
SOA	Service Oriented Architecture
TTQT	Thanh toán quốc tế

MỞ ĐẦU

Ngày nay, các hệ thống công nghệ thông tin phục vụ cho ngân hàng (Hệ thống quản lý quan hệ khách hàng (CRM), Hệ thống quản lý hiệu quả hoạt động (KPI), Định giá điều chuyển vốn nội bộ (FTP), Quản lý tiền mặt, kho quỹ, tài sản v.v…) thường xuyên được nâng cấp và phát triển, góp phần tăng hiệu quả điều hành và thực thi, cũng như năng lực thanh tra, giám sát. Bên cạnh đó, để mang tính nhất quán và đồng bộ, các hệ thống này phải được giao tiếp với nhau – đây cũng chính là vấn đề khó khăn mà các tổ chức Ngân hàng đang gặp phải. Thực trạng hiện nay, các hệ thống, ứng dụng giao tiếp với nhau qua mô hình tích hợp point-to-point (hai ứng dụng kết nối trực tiếp với nhau) và tích hợp tĩnh (viết mã tích hợp đan xen mã ứng dụng). Theo thời gian, phương thức truyền thống này sẽ tạo ra một kết nối chồng chéo, phụ thuộc chặt chẽ lẫn nhau dẫn tới khó khăn trong chỉnh sửa nghiệp vụ khi có yêu cầu, hệ quả là chi phí tích hợp gia tăng đáng kể. Do đó, trục tích hợp dữ liệu ESB được đưa ra và trở thành giải pháp hàng đầu để giải quyết những khó khăn này.

Với thực trạng như trên, luận văn này sẽ hướng đến mục tiêu là nghiên cứu, khảo sát và đánh giá một số giải pháp tích hợp dịch vụ mã mở dựa trên công nghệ ESB Middleware, từ đó ứng dụng trong tích hợp một số dịch vụ nghiệp vụ tại ngân hàng TPBank.

Từ mục tiêu đó, chúng tôi đã tiến hành thực hiện các công việc của luận văn với những nội dung được thể hiện trong bản thảo với cấu trúc như sau:

Chương 1: Giới thiệu về cơ sở lý thuyết, các vấn đề liên quan đến tích hợp hệ thống và các công nghệ được sử dụng.

Chương 2: Trình bày về ESB, các khái niệm, các thành phần và so sánh một số công cụ ESB Middleware

Chương 3: Trình bày về thực trạng hệ thống công nghệ thông tin tại ngân hàng TPBank, đưa ra phương pháp giải quyết vấn đề. Xây dựng, thử nghiệm và đánh giá hệ thống.

Kết luận chung: Các kết quả đạt được, các điểm còn hạn chế và hướng phát triển kế tiếp.

CHƯƠNG 1. TỔNG QUAN VỀ TÍCH HỢP HỆ THỐNG

1. Giới thiệu

Ngày nay, khi sự phát triển vươn tới những đỉnh cao mới, nhu cầu về làm chủ tri thức của con người được đặt lên hàng đầu. Do đó, thông tin dữ liệu cần phải được dễ dàng truy xuất, độ tin cậy, tính chính xác cao và luôn luôn sẵn sàng phục vụ. Theo thời gian, sự phát triển về công nghệ và nhu cầu của con người đã xuất hiện những hệ thống hoạt động và trao đổi dữ liệu theo những kiến trúc mới và đặt ra thách thức là các kiến trúc mới này cần trao đổi dữ liệu và có thể phối hợp nhịp nhàng với những hệ thống cũ. Do đó người ta đã đưa ra giải pháp tích hợp hệ thống để giải quyết vấn đề này.

1.1. Khái niệm tích hợp hệ thống

Theo Wikipedia, ta có định nghĩa về tích hợp hệ thống như sau:

Về mặt kỹ thuật: *Tích hợp hệ thống được hiểu là quá trình đưa các thành phần của các hệ thống con vào thành một hệ thống chung, và đảm bảo rằng các thành phần này liên kết và hoạt động với nhau thành một thể thống nhất hoàn chỉnh. [5]*

Về mặt công nghệ thông tin: *Tích hợp hệ thống là quá trình liên kết các hệ thống máy tính và các phần mềm với nhau để hoạt động như một hệ thống hoàn chỉnh. [1]*

Vậy *Tích hợp hệ thống là quá trình liên kết, kết nối các hệ thống thông tin, cả về khía cạnh chức năng lẫn hạ tầng tính toán, để hoạt động như một thể thống nhất. [1]*

1.2. Mục tiêu và thách thức

- Mục tiêu

Tích hợp hệ thống giúp chúng ta có thể truy xuất được đúng dữ liệu cần thiết từ đúng hệ thống cần tìm, trong đúng thời điểm mong muốn với chất lượng tuyệt đối và chi phí thấp nhất.

- Thách thức của tích hợp hệ thống

Thực tế, khi một hệ thống ra đời, điều mà tổ chức quan tâm chủ yếu là tạo ra một hệ thống để giải quyết những vấn đề đang tồn tại hoặc theo một nhu cầu của một đơn vị

nào đó trong một khoảng thời gian cho phép. Chính vì thế mà các hệ thống này thường không được tính toán trước để tích hợp, thiết kế thường độc lập và thường theo kiểu "nghĩ đến đâu làm đến đấy", dó đó thường rất khó để kết hợp những thành phần nhỏ để giải quyết bài toán chung.

Hơn nữa, các ứng dụng như dịch vụ Web, ứng dụng cho hệ điều hành Windows, Linux… được phát triển bởi nhiều ngôn ngữ khác nhau như là: C++, Java, dotNet, … cũng như phương thức quản lý dữ liệu khác nhau: Tệp lưu trữ, Dữ liệu quan hệ, Dữ liệu có cấu trúc và phi cấu trúc. Việc vượt qua những điểm khác biệt này để tích hợp các thành phần thành một hệ thống là điều khá khó khăn. Do đó các tổ chức, các chuyên gia tích hợp cần có một kiến thức tổng thể về hệ thống cùng với một nguồn đầu tư kinh phí lớn để có thể thực hiện tốt việc tích hợp.

1.3. Kiểu tích hợp

1.3.1. Tích hợp mức dữ liệu

Đây là kiểu tích hợp dữ liệu ở mức thấp, các ứng dụng/ hệ thống tham gia vào hệ tích hợp sẽ chia sẻ dữ liệu chung với nhau. Ở mức độ tích hợp này, cần tiến hành các công việc sau: [1]

- Định danh dữ liệu: chỉ ra vị trí nguyên thủy trong hệ phân tán.
- Thể loại hóa dữ liệu: phân loại dữ liệu và gán nhãn thể loại
- Xây dựng mô hình siêu dữ liệu (metadata), mô tả dữ liệu về dữ liệu

Một số phương pháp chia sẻ dữ liệu điển hình: Chia sẻ dữ liệu dạng tệp (File-base data sharing), Chia sẻ cơ sở dữ liệu (Shared Database), Đồng bộ tệp (Socket).

1.3.2. Tích hợp mức chức năng

Là phương pháp cho phép các ứng dụng chia sẻ các chức năng (tái sử dụng chức năng) lẫn nhau. [1]
Một số phương pháp điển hình của tích hợp chức năng là:

- Gọi thủ tục từ xa (Remote Procedure Call)
- Đối tượng phân tán (Distributed Object)
- Thông điệp (Message)

1.3.3. Tích hợp mức dịch vụ (quy trình)

Là tích hợp mức cao, cho phép khắc phục những nhược điểm của phương thức thông điệp.

Phương pháp này có 2 loại:

- Tích hợp hệ thống dựa vào tích hợp quy trình nghiệp vụ.
- Tích hợp hệ thống dựa vào kiến trúc hướng dịch vụ.

1.3.3.1. Tích hợp quy trình

Tích hợp mức quy trình đảm bảo mục tiêu tạo mô hình chung giữa các hệ thống liên kết qua dịch vụ và quy trình. Phương pháp này thường được sử dụng trong các hệ thống:

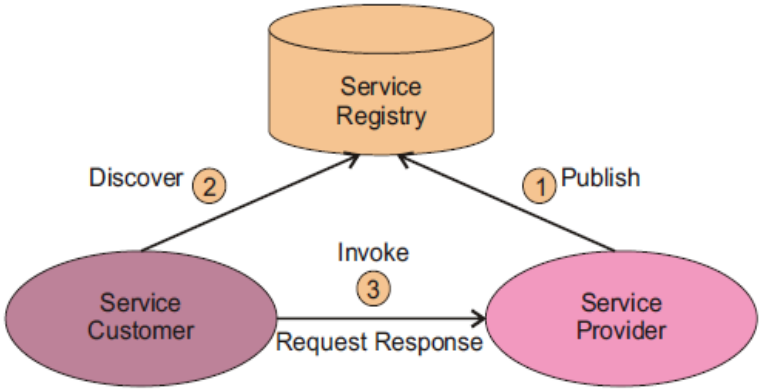
- Dịch vụ khách hàng
- Quản trị nguồn nhân lực
- Giao dịch tài chính.
- Chế tạo

Mô hình quy trình chung thường phải đủ bao quát hết các quy trình trong hệ thống tích hợp. Để đặc tả quy trình nghiệp vụ người ta sử dụng chuẩn ngôn ngữ Business Modeling Language (BPML).

1.3.3.2. Tích hợp hướng dịch vụ (Service-Oriented Architecture)

Kiến trúc hướng dịch vụ (SOA) là mô hình xây dựng ứng dụng dựa trên các dịch vụ đã có trên mạng chuyên biệt, chẳng hạn như Web. SOA giải quyết các vấn đề còn tồn tại của các hệ thống hiện nay như phức tạp, không linh hoạt và không ổn định.

Các thành phần cơ bản của SOA



Hình 1.1. Các thành phần cơ bản của SOA

- Service Registry: tao ra giao diện dịch vụ và cung cấp khả năng truy cập thông tin có sẵn tới Service Customer.
- Service Customer: xác định thông tin của service registry, sau đó liên kết với service provider để gọi dịch vụ.
- Service Provider: tạo ra dịch vụ và cung cấp thông tin về giao diện, truy cập cho Service Registry. Mỗi nhà cung cấp dịch vụ phải quyết định dịch vụ nào sẽ cung cấp, đánh giá giữa vấn đề an ninh bảo mật và tính sẵn sàng, xác định làm sao để bán dịch vụ hoặc làm sao để khai thác dịch vụ miễn phí.

Nguyên lý cơ bản của SOA [2]

- Liên kết không chặt: các dịch vụ có ít sự rằng buộc với nhau tuy nhiên các module có rằng buộc rõ ràng, đảm bảo tính mềm dẻo của SOA
- Tính tự trị: các dịch vụ có quyền kiểm soát dựa vào logic bên trong của dịch vụ đó.
- Khả năng cộng tác: hệ thống có thể giao tiếp với nhau trên nhiều nền tảng và ngôn ngữ khác nhau.
- Đóng gói:
- Sử dụng lại: tái sử dụng lại các dịch vụ giúp loại bỏ những thành phần trùng lặp, tăng độ vững chắc trong cài đặt, đơn giản hóa việc tự trị
- Phi trạng thái: các dịch vụ hoạt động phi trạng thái.
- Có thể tìm thấy: người dùng có thể tìm kiếm dịch vụ và đăng ký sử dụng dịch vụ đó.

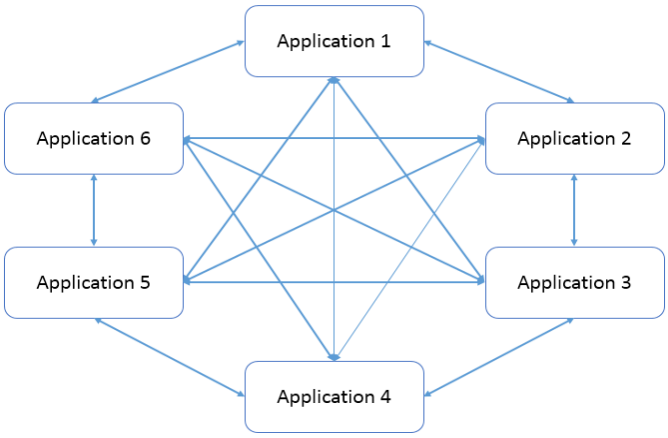
2. Kiến trúc tích hợp hệ thống

Như đã trình bày ở trên, việc tích hợp hệ thống cần một kiến thức tổng thể về lĩnh vực hệ thống, vì vậy việc nắm vững kiến trúc hệ thống là điều vô cùng quan trọng. Kiến trúc của các hệ thống thông tin hiện nay là kiến trúc đa, cụ thể nó bao gồm:

- Client: người dùng hoặc chương trình thi hành các tác vụ, các thao tác trên hệ thống.
- Presentation Layer: tầng giúp client gửi yêu cầu và nhận lại kết quả phản hồi.
- Application Logic: tầng này đảm bảo thực hiện các quy trình nghiệp vụ đồng thời xác lập những thao tác nào được thi hành trên hệ thống
- Resource manager: tầng tương tác mức thấp nhất với tài nguyên dữ liệu của hệ thống. Tầng này có thể là hệ quản trị cơ sở dữ liệu hoặc hệ thống quản trị dữ liệu có khả năng lưu trữ và truy vấn dữ liệu.

2.1. Kiến trúc Point-to-Point

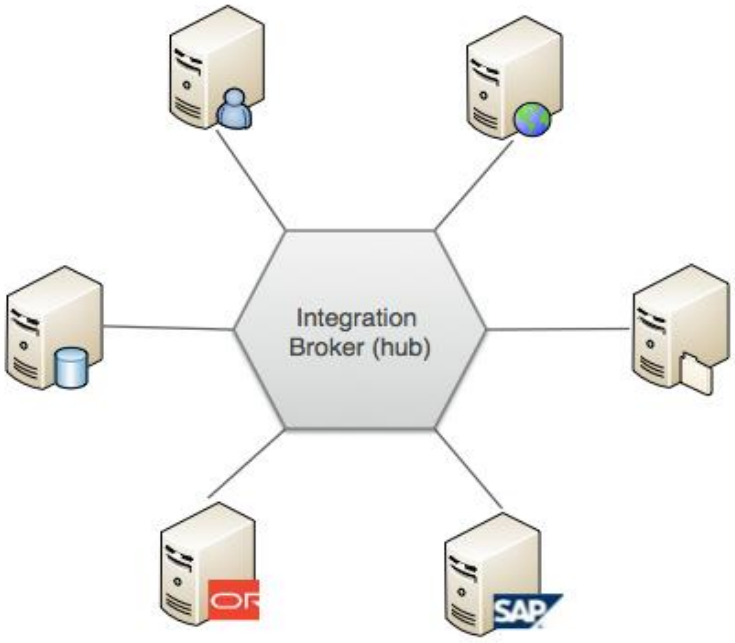
- Các ứng dụng công nghệ thông tin giao tiếp với nhau thông qua các giao diện (interfaces)
- Các giao tiếp này được hỗ trợ bởi các giao diện, nó có thể được thực hiện trong thời gian thực hoặc đồng bộ
- Số lượng giao diện tăng lên khi số lượng ứng dụng công nghệ thông tin tăng lên.
- Phù hợp khi hệ thống có số lượng các ứng dụng cần giao tiếp và tích hợp với nhau không nhiều.



Hình 1.2. Kiến trúc Point-to-Point

2.1.1. Kiến trúc Hub-and-Spoke

Được sử dụng trong các hệ thống tích hợp ứng dụng doanh nghiệp Enterprise Application Integration (EAI), kiến trúc hub-and-spoke được tích hợp từ bộ xử lý trung tâm của hệ thống. [2]

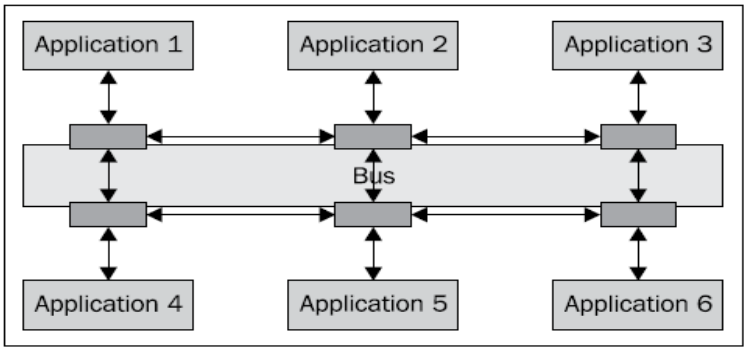


Hình 1.3. Kiến trúc Hub-and-Spoke

- Tất cả hệ thống được tích hợp tại một điểm duy nhất – Hub
- Sử dụng cơ sở dữ liệu chia sẻ
- Để mở rộng hệ thống, hub sẽ được co cụm lại
- Hub có chức năng định tuyến thông điệp (messaging routing) và chuyển đổi dữ liệu (data transformation)
- Phù hợp với tích hợp hệ thống có số lượng ứng dụng vừa và ít.

2.1.2. Kiến trúc Pipeline

Trong kiến trúc Pipeline, các hệ thống độc lập được tích hợp với nhau bằng một bus thông điệp. Việc triển khai kiến trúc này tương tự với kiến trúc hub-and-spoke, việc sử dụng các thành phần middlerware thích hợp cho phép truyền thông giữa các hệ thống được chuẩn hóa. Các ứng dụng giao tiếp với bus trung tâm thông qua các giao diện (interfaces) trên đường truyền mạng. [2]

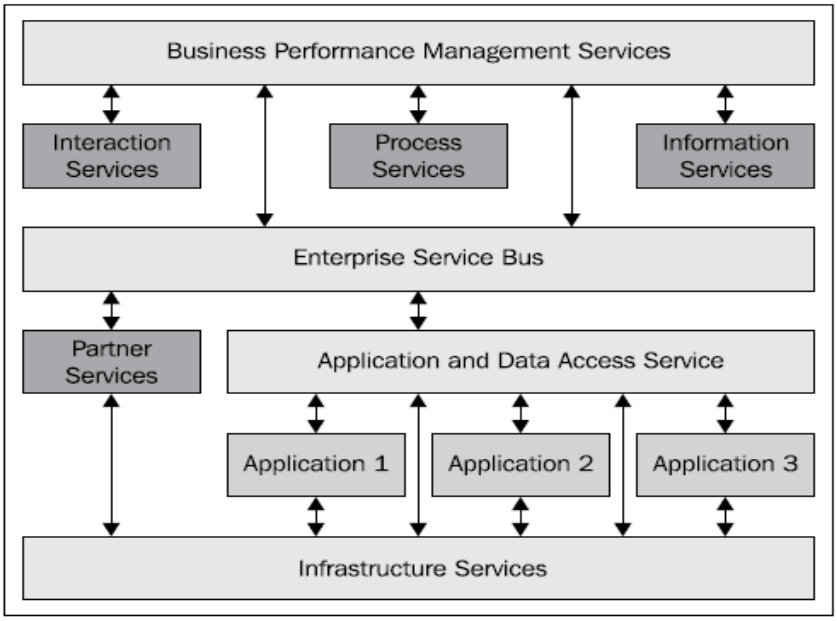


Hình 1.4. Kiến trúc Pipeline

- Kiến trúc linh hoạt, tốn ít chi phí theo dõi vận hành, các hệ thống độc lập có thể được tích hợp hoặc loại bỏ một cách dễ dàng
- Khi khối lượng truyền nhận dữ liệu lớn sẽ có nguy cơ tắc nghẽn, do đó cần thiết lập kênh truyền riêng biệt
- Phù hợp với hệ thống tích hợp hướng sự kiện, phân phối dữ liệu 1-n (kênh phát song), hệ thống sử dụng cơ sở dữ liệu n-1 (kho dữ liệu).

2.1.3. Kiến trúc hướng dịch vụ SOA

Là một hướng tiếp cận với việc thiết kế và tích hợp phần mềm, chức năng, hệ thống theo dạng module và có khả năng truy cập thông qua môi trường mạng. SOA giải quyết các vấn đề còn tồn đọng của các hệ thống hiện nay như: phức tạp, không linh hoạt và không ổn định. [2]



Hình 1.5. Kiến trúc hướng dịch vụ SOA

- Các dịch vụ thực hiện quá trình tương tác chủ yếu thông qua các thành phần giao tiếp. Các thành phần giao tiếp này quy định về những định dạng thông điệp sử dụng trong quá trình trao đổi.
- SOA mang đến khả năng tổng hợp một lớp các ứng dụng mới bằng cách kết hợp chức năng từ những hệ thống sẵn có, cung cấp cho người dùng cuối những chức năng liên kết.
- Phù hợp với hầu hết những hệ thống hướng dịch vụ ngày nay.

SOA có tính mềm dẻo và tùy biến rất cao. SOA mang đến khả năng tổng hợp một lớp các ứng dụng bằng cách kết hợp chức năng từ các hệ thống có sẵn. Phía triệu gọi dịch vụ không cần quan tâm đến địa chỉ hoặc công nghệ nền tảng service, do đó có tính linh hoạt và tăng khả năng triển khai. SOA có hai mô hình ứng dụng chính đó là Web service và ESB. Chương sau luận văn sẽ đề cập rõ hơn về ESB.

3. Công nghệ tích hợp

3.1. Chia sẻ cơ sở dữ liệu

Công nghệ này cung cấp việc truy cập vào cơ sở dữ liệu thông qua một lớp trừu tượng, nó cho phép thay đổi DBMS mà không cần sửa lại mã nguồn của ứng dụng. Nói cách khác, nó cung cấp các mã nguồn để truy cập vào các nguồn dữ liệu khác nhau. Do đó, công nghệ này rất hữu ích để truy xuất dữ liệu từ các DBMS khác nhau.

Các đại diện nổi tiếng cho công nghệ này là: Java Database Connectivity (JDBC) và Java Data Objects (JDO) trên nền tảng Java; Open Database Connectivity (ODBC) và Active Data Objects (ADO.NET) trên nền tảng Microsoft

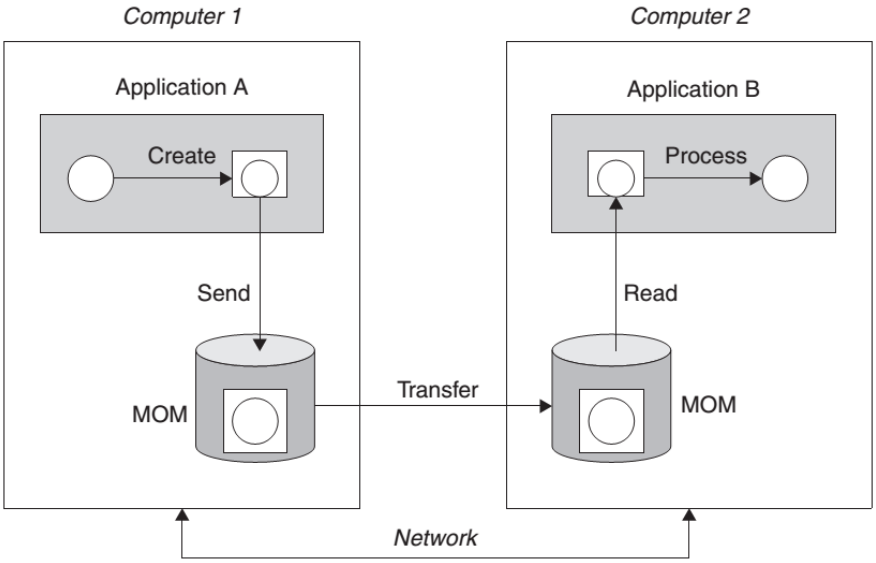
3.2. Message-oriented middleware

Phương thức này giải quyết vấn đề của hai phương thức trên. Phương thức này dựa trên cơ chế gửi thông điệp không đồng bộ - asynchronous message, tức là máy khách gửi yêu cầu tới máy chủ mà không cần chờ kết quả phản hồi từ máy chủ.

Trong phương thức này, các ứng dụng không tương tác trực tiếp với nhau, mà chúng tương tác gián tiếp thông qua hàng đợi. Một hàng đợi là tập hợp các thông

điệp có thể được chia sẻ với nhiều máy tính. Những đoạn mã được xây dựng để kết nối được gọi là Message-oriented middleware (MOM).

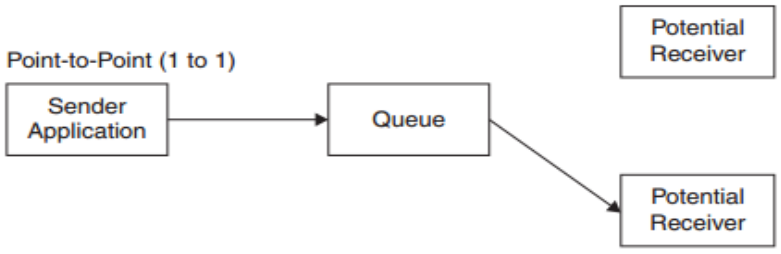
Các thông điệp được gửi và nhận theo cơ chế đồng bộ thông qua hàng chờ/queue. Các thông điệp này phải được gửi đến đích mong muốn, nếu chưa đến đích thì MOM sẽ thực hiện gửi lại ngay. MOM có cơ chế điều phối thông điệp do đó giảm thiểu vấn đề quá tải server



Hình 1.6. Kiến trúc thông điệp

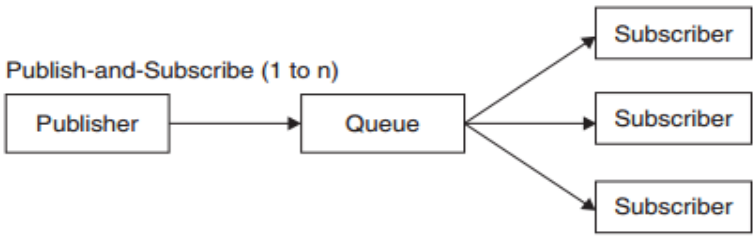
Các thành phần của MOM gồm có

- Hàng đợi/Kênh (Queues/Chanels): sử dụng để truyền dữ liệu. Có hai loại hàng đợi:
    - Point-to-point: chỉ một điểm nhận nhận mỗi thông điệp



Hình 1.7. Hàng đợi Point-to-point

    - Push and Subscribe: thông điệp được gửi tới tất cả các subscriber. Mỗi subscriber như một bản sao lưu của thông điệp và pushlisher không quan tâm tới ai lắng nghe



Hình 1.8. Hàng đợi Push and Subscribe

- Thông điệp (Message): đóng gói dữ liệu (function) cần trao đổi giữa client và server. Thông điệp bao gồm:
    - Header: định nghĩa thông điệp và các thông tin điều khiển.
    - Body: chứa thông tin sẽ được xử lý khi ứng dụng nhận thông điệp.

- Điểm kết thúc (EndPoints): điểm cho phép client/server kết nối được với MOM và để gửi hay nhận một thông điệp.

Ưu điểm:

- Nâng cao tính mở rộng của hệ thống tích hợp
- Đảm bảo độ tin cậy, tính chính xác khi truyền dữ liệu

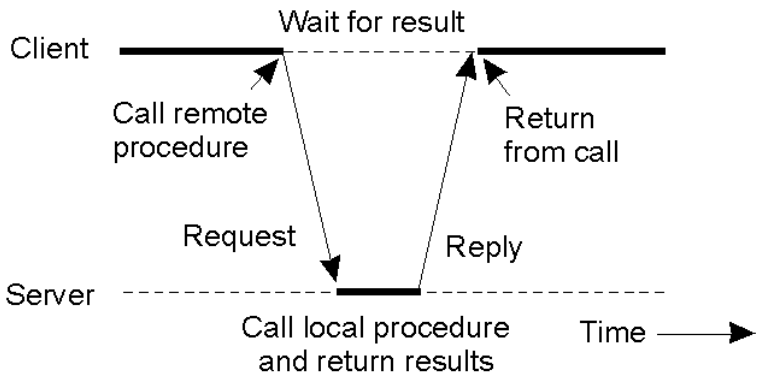
Nhược điểm:

- Sử dụng nhiều MOM cùng một lúc có thể dẫn tới sự không đồng nhất như là giao thức không đồng nhất (HTTP hoặc HTTPS); định dạng thông điệp không đồng nhất.
- Có những ứng dụng cần cả cơ chế gọi hàm đồng bộ và không đồng bộ.

3.3. Remote Procedure Calls

RPC là phương thức tương tác giữa các ứng dụng cho phép ứng dụng này triệu gọi hàm/thủ tục từ ứng dụng khác mà không cần lập trình lại ứng dụng đó.

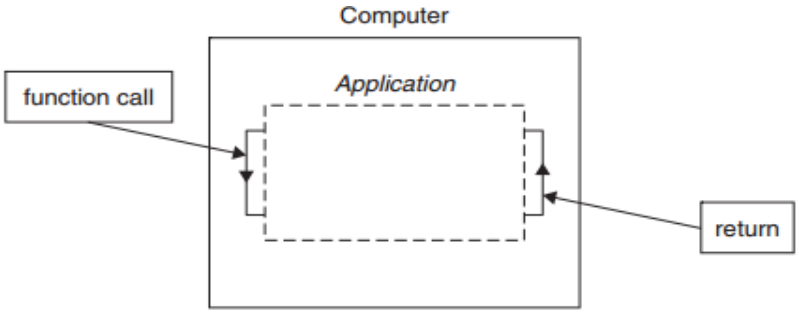
RPC thực hiện theo kiểu đồng bộ chức năng (synchronous functions): ứng dụng gọi đến hàm phải chờ đến khi nhận được kết quả trả về mới tiếp tục thực hiện công việc khác.



Hình 1.9. Gọi thủ tục từ xa (RPC)

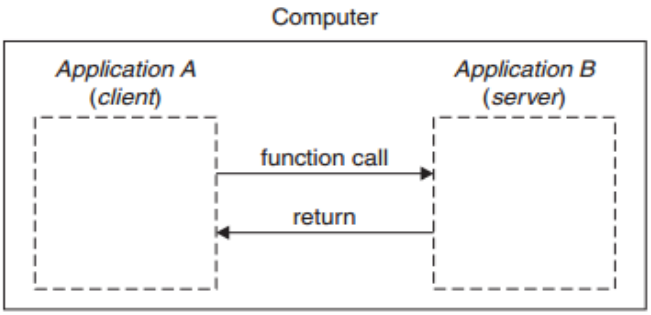
Đồng bộ chức năng có 3 loại hàm gọi:

- Local function call: Hàm gọi và chức năng được gọi cùng trên một ứng dụng



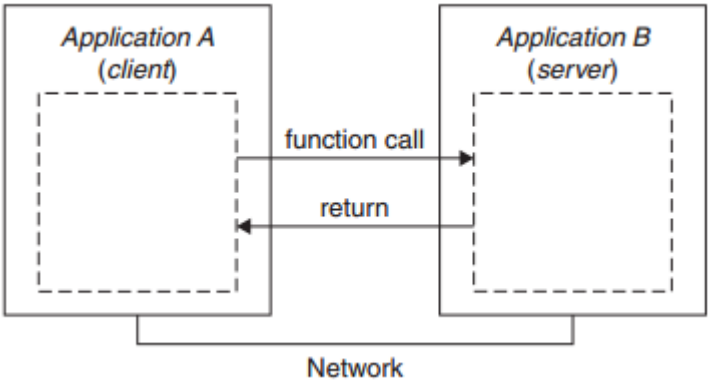
Hình 1.10. Local function call

- Restricted RPC: Hàm gọi và chức năng được gọi trên các ứng dụng khác nhau trên cùng một máy chủ.



Hình 1.11. Restricted RPC

- Loại 3: Ứng dụng client trên một máy chủ gọi hàm trên một máy chủ ứng dụng khác, hai máy chủ này kết nối với nhau qua mạng máy tính



Hình 1.12. Kiến trúc loại 3 của RPC

RPC cho phép ẩn chi tiết truyền thông giữa các lời gọi hàm, trở thành cầu nối giữa các môi trường và nền tảng khác nhau.

Ưu điểm:

- Là phương thức đầu tiên cho phép chia sẻ hàm
- Có thể triển khai với nhiều nền tảng khác nhau

Nhược điểm:

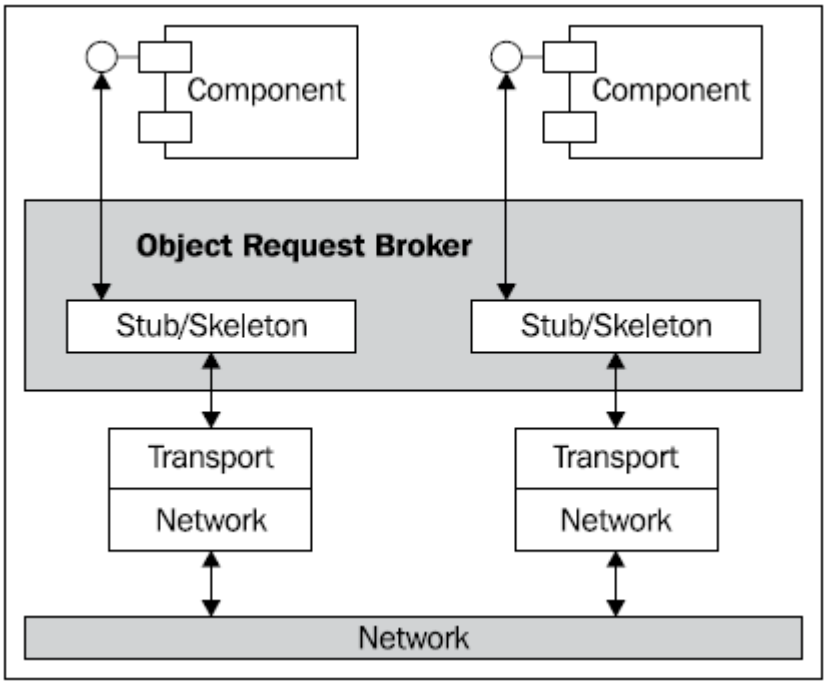
- Cần nắm được sự đặc tả giao tiếp và các phương thức đóng gói dữ liệu.
- Các ứng dụng cần sử dụng chung một ngôn ngữ lập trình
- Cần duy trì kết nối chặt chẽ và liên tục do sử dụng cơ chế hàm đồng bộ
- Rất phức tạp khi có nhiều lời gọi

3.4. Object Request Brokers

Là một công nghệ middleware giúp quản lý và hỗ trợ việc truyền thông điệp giữa các đối tượng phân tán hoặc các thành phần khác mà không cần quan tâm tới nội dung của giao tiếp. ORBs cung cấp tính minh bạch về vị trí, ngôn ngữ lập trình, giao thức và hệ điều hành.

Giao tiếp giữa các đối tượng phân tán và các thành phần thông qua các giao diện (interfaces). Điều này giúp tăng cường khả năng bảo trì và các chi tiết thực hiện được ẩn đi, tăng tính bảo mật của hệ thống, các thông tin giao tiếp thường là đồng bộ.

ORBs thường được kết nối với các dịch vụ định vị cho phép xác định vị trí các thành phần trong mạng. ORBs là công nghệ khá phức tạp, tuy nhiên nó lại che giấu hầu hết những sự phức tạp này. Cụ thể, nó cung cấp một vị trí ảo – sau đó mô hình hóa các thành phần trong cùng địa phương, mặc dù thực tế các thành phần này có thể nằm ở bất cứ đâu trong mạng. Điều này làm cho đơn giản hóa sự phát triển hệ thống nhưng nó lại ảnh hưởng tới hiệu năng. [2]



Hình 1.13. Kiến trúc ORBs

Các hệ thống dựa trên ORB thường rất linh hoạt. Nó có thể di chuyển một số các hàm functions về phía clients và một số chức năng phía máy chủ, hoặc cung cấp các functions này như một tiến trình riêng biệt hoặc thậm chí tích hợp chúng này vào hạt nhân của hệ điều hành. Có ba tiêu chuẩn chính của ORB:

- Tương thích với OMG CORBA ORB
- Java RMI và RMI-IIOP
- Microsoft COM/DCOM/COM+/.NET Remoting/WCF

Có nhiều các sản phẩm của ORB tương tích với yêu cầu kỹ thuật của CORBA ORB và các kế thừa khác nhau của RMI/ RMI-IIOP. Đặc biệt RMI-IIOP rất quan trọng vì nó sử dụng một giao thực để truyền thông giữa các thành phần của CORBA ORB, cụ thể là IIOP (Internet Inter-ORB Protocol). Điều này khiến cho RMI-IIOP tương thích với CORBA.

3.5. Máy chủ ứng dụng

Máy chủ ứng dụng xử lý phần lớn tất cả các tương tác giữa tầng client và tầng trình diễn dữ liệu. Nó cung cấp một tập các dịch middleware cùng với môi trường quản lý – nơi mà triển khai các thành phần logic nghiệp vụ.

Máy chủ ứng dụng hỗ trợ các dịch vụ web (web services), ORBs, MOM, quản lý giao tiếp, bảo mật, cân bằng tải và quản lý tài nguyên. Máy chủ ứng dụng cung cấp một giải pháp toàn diện cho nhu cầu thông tin doanh nghiệp, và nó cũng là một nền tảng tốt để thực hiện tích hợp. Ngày nay, các nhà cung cấp thường xác định máy chủ dịch vụ như là một phần của công cụ tích hợp, hoặc tập trung vận hành các ứng dụng bằng cách bổ sung một số hàm chức năng như là kết nối các hệ thống back-end.

Cho dù được sử dụng để tích hợp hoặc phát triển ứng dụng mới, các máy chủ ứng dụng vẫn là một nền tảng phần mềm. Một nền tảng phần mềm là sự kết hợp của các công nghệ phần mềm cần thiết để vận hành ứng dụng, do đó các máy chủ ứng dụng này xác định cơ sở hạ tầng của tất cả các ứng dụng được phát triển và vận hành trên chúng.

Máy chủ ứng dụng hỗ trợ những nền tảng chuẩn, mở và được sử dụng rộng rãi như là Java Enterprise Edition. Các khía cạnh quan trọng nhất của các nền tảng này là:

- Nền tảng phần mềm, kiến trúc của các ứng dụng được phát triển cho nền tảng đó, khả năng tương tác, khả năng mở rộng, tính cơ động, tính khả dụng, độ tin cậy, phạm vi hợp đồng với khách hàng, khả năng phát triển và thích ứng với các giải pháp mới. Về mặt tích hợp, khả năng tương tác với những hệ thống khác.
- Tính mở rộng cho phép các nhà cung cấp máy chủ ứng dụng và các công ty bên thứ ba có một vài khả năng tác động đến sự phát triển của nền tảng này.
- Khả năng tương tác giữa các nền tảng khác nhau là điều rất quan trọng đối với việc áp dụng máy chủ ứng dụng. Đặc biệt là các nền tảng điều chỉnh bổ sung và sửa đổi. Việc các nền tảng kết hợp với nhau càng chặt chẽ thì cơ hội thành công và mở rộng thị trường ngày càng lớn. Tuy nhiên, mỗi một nền tảng cần cung cấp sự khác biệt giữa các máy chủ ứng dụng với nhau.
- Chi phí cho nền tảng cũng là một yếu tố quan trọng và có lẽ là điều khó đánh giá nhất vì nó bao gồm cả chi phí cài đặt máy chủ ứng dụng và các phần mềm

phát triển khác. Chi phí phần cứng, đào tạo, và phí duy trì các ứng dụng trong vòng đời của chúng.

- Cuối cùng là sự phát triển của nền tảng. Nền tảng càng phát triển thì nó càng được kiểm thử và chứng minh rằng nó phù hợp với các quy mô lớn.

3.6. Dịch vụ web

Dịch vụ web là công nghệ phân tán mới nhất hiện nay. Nó cung cấp nền tảng công nghệ để đạt được khả năng tương tác giữa các ứng dụng cho dù có khác nhau về nền tảng sử dụng, hệ điều hành và ngôn ngữ lập trình.

Các đặt trưng cơ bản của dịch vụ web dựa trên là SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) và UDDI (Universal Description, Discovery, and Integration). SOAP, WSDL và UDDI dựa trên XML giúp cho các thông điệp và giao thức dịch vụ web trở nên dễ dàng đối với người lập trình. Các hoạt động trong dịch vụ web dựa trên việc trao đổi các thông điệp XML. Nó là một tập hợp dữ liệu đầu vào, đầu ra và mã lỗi, và sự kết hợp của các tin nhắn này xác định một loạt các hoạt động (một chiều, request/response, yêu cầu trả lời, hoặc thông báo).

Các dịch vụ web hỗ trợ các tương tác đồng bộ và không đồng bộ. Các dịch vụ web không có trạng thái và không sử dụng giao thức chuẩn như là HTTP (Hyper Text Transfer Protocol), SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol), và MIME (Multipurpose Internet Mail Extensions). Do đó, việc kết nối thông qua Internet đơn thuần, ngay cả kết nối có được xác lập tường lửa thì cũng không gặp vấn đề.

Ngoài các ưu điểm, dịch vụ web cũng có những nhược điểm. Một trong số đó là hiệu năng – nó sẽ không được tốt như kiến trúc phân tán sử dụng giao thức nhị phân để truyền thông. Do các dịch vụ web không cung cấp cơ sở hạ tầng và chất lượng dịch vụ (QoS) như bảo mật và các dịch vụ khác. Thay vào đó, các dịch vụ web lại đưa ra các thành phần bổ sung:

- **WS-Security**: Xác định địa chỉ và bảo mật thông điệp, cho phép truyền thông an toàn.
- **WS-Coordination**: Xác định phạm vi tương tác giữa các dịch vụ web và là nền tảng cho WS-AtomicTransaction và WS-BusinessActivity.

- **WS-AtomicTransaction và WSBusinessActivity**: Hỗ trợ cho các tương tác phân tán với các dịch vụ web. AtomicTransaction: chỉ định thời gian ngắn, các tương tác ACID và BusinessActivity xác định tương tác nghiệp vụ.
- **WS-Reliable Messaging**: hỗ trợ giao tiếp tin cậy và phân phối thông điệp giữa các dịch vụ web thông qua giao thức truyền tải khác nhau.
- **WS-Addressing**: Chỉ định sự phối hợp và định danh thông điệp
- **WS-Inspection**: hỗ trợ cho các nội dung "động" trong các mô tả của dịch vụ web
- **WS-Policy**: Chỉ định các chính sách được tuyên bố và các trao đổi giữa các dịch vụ web hợp tác
- **WS-Eventing**: Định nghĩa mô hình sự kiện cho thông báo không đồng bộ của các bên quan tâm trong dịch vụ web.

3.7. Trục tích hợp dịch vụ tổng thể (Enterprise Service Buses)

ESB là một cơ sở hạ tầng phần mềm hoạt động như là một lớp trung gian middleware để giải quyết các yêu cầu mở rộng hệ thống mà dịch vụ web không làm được, ví dụ như tích hợp giữa các dịch vụ web với các ứng dụng middleware khác, cũng như giải quyết các vấn đề về an ninh, quản lý, kiểm soát các dịch vụ truyền thông. ESB giải quyết các vấn đề trên, đồng thời nó tăng tính linh hoạt trong giao tiếp giữa các dịch vụ, làm đơn giản hóa việc tái sử dụng các dịch vụ.

ESB cung cấp cơ sở hạ tầng truyền thông mạnh mẽ, đáng tin cậy, an toàn và khả năng mở rộng giữa các dịch vụ. Nó cũng cung cấp kiểm soát truyền thông và kiểm soát việc sử dụng các dịch vụ. ESB cung cấp khả năng định tuyến để điều hướng các thông điệp tới các dịch vụ khác nhau dựa trên nội dung, nguồn gốc, hoặc các thuộc tính khác và khả năng chuyển đổi để biến đổi thông điệp trước khi chúng được truyền tới đích. Đối với các thông điệp định dạng XML, những chuyển đổi như vậy thường được thực hiện bằng cách sử dụng XQuery của XSLT ((Extensible Stylesheet Language for Transformations) hoặc XQuery.

ESB cung cấp kiểm soát việc triển khai, sử dụng và bảo trì dịch vụ. Ngoài ra nó còn cho phép kiểm soát, cân bằng tải, tối ưu hiệu năng, triển khai phân tán, ước tính chi phí dịch vụ, cấu hình trực tuyến v.v…Các tính năng quản lý quan trọng khác bao gồm định nghĩa tương quan giữa các thông điệp, định nghĩa các đường truyền tin cậy, định nghĩa các rằng buộc về bảo mật v.v… Chương sau luận văn sẽ trình bày rõ hơn về công nghệ ESB.

4. Kết chương

Trong chương này, luận văn đã trình bày tổng quan về tích hợp hệ thống bao gồm: các khái niệm cơ bản, kiến trúc của tích hợp hệ thống và một công nghệ tích hợp hệ thống. Trong chương sau luận văn sẽ đi sâu vào mô hình ESB.

CHƯƠNG 2. TÍCH HỢP DỊCH VỤ DỰA TRÊN CÔNG NGHỆ ESB

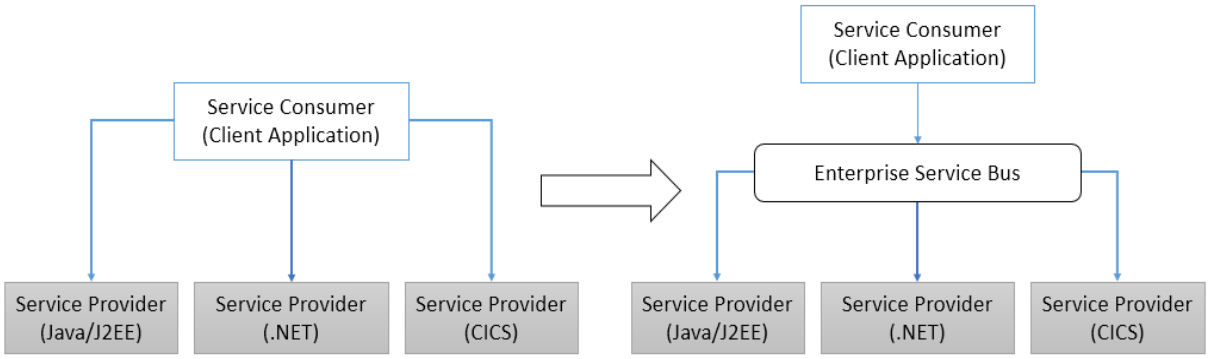
1. Khái niệm trục dịch vụ tổng thể ESB

1.1. Khái niệm ESB và Middleware

Middleware là phần mềm máy tính với nhiệm vụ kết nối các thành phần phần mềm hoặc các ứng dụng với nhau. Phần mềm loại này bao gồm một tập các dịch vụ cho phép sự tương tác giữa các tiến trình chạy trên một hoặc nhiều máy khác nhau. Công nghệ middleware đã được phát triển để cung cấp khả năng hoạt động tương hỗ, phục vụ cho các kiến trúc phân tán thường được để hỗ trợ và đơn giản hóa các ứng dụng phân tán phức tạp.

ESB là một kiến trúc phần mềm trung gian (middleware) dựa trên phương thức truyền thông điệp, nó cung cấp một cơ sở hạ tầng tích hợp phục vụ cho các dịch vụ định tuyến, gửi và nhận phản hồi yêu cầu để tạo điều kiện thuận lợi cho việc tương tác giữa các ứng dụng một cách an toàn, tin cậy và hiệu quả cao.

1.2. Kiến trúc cơ bản ESB



Hình 2. 1. Kiến trúc ESB

Bên yêu cầu và bên phản hồi không cần phải cùng một kiểu định dạng tin nhắn, giao thức truyền tin hay thậm chí là địa chỉ đích. Các ứng dụng yêu cầu mới có thể được kết nối tới hệ thống mà không cần phải thay đổi các service phản hồi (providers) và ngược lại, những provider có thể được gọi đến mà không cần thay đổi các yêu cầu. Những thay đổi về bên yêu cầu sẽ không làm ảnh hưởng tới bên cung cấp (providers) cũng như việc tác động đến provider cũng không làm ảnh hưởng tới bên yêu cầu. Các

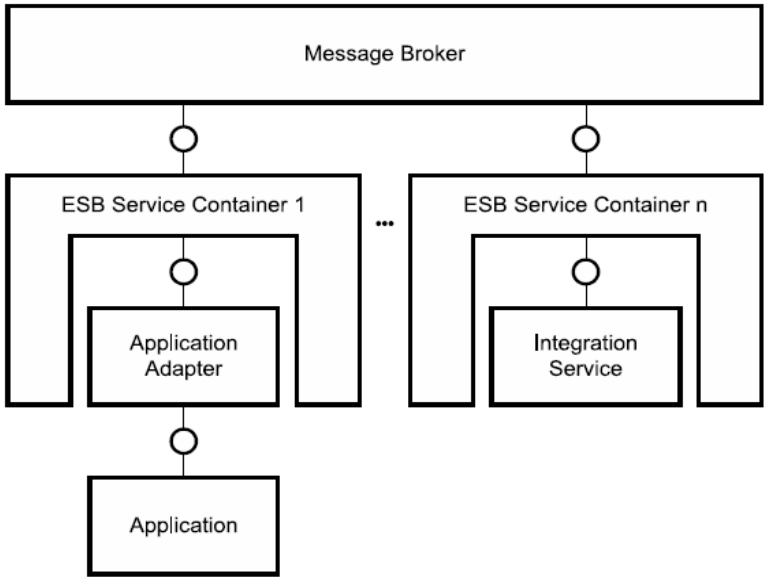
vấn đề về bảo mật và quản lý có thể được bổ sung, thực thi hay nâng cấp bởi ESB, giúp nâng cao tính cơ động của hệ thống.

1.3. Mô hình hóa luồng dữ liệu trong ESB

ESB thường được thực hiện qua lớp dịch vụ (services containers) và được phân phối thông qua môi trường mạng. Các container này cung cấp các dịch vụ tích hợp như là định tuyến, chuyển đổi định dạng, điều hướng dứng dụng (application adapter) hoặc các cầu nối MOM, và cung cấp các dịch vụ này một cách rộng rãi trên môi trường giao tiếp.

Trong các giải pháp ESB hiện nay, phần hạ tầng kiến trúc thường được xây dựng dựa trên kiến trúc JMS Middleware để đảm bảo thông điệp được truyền đi (JMS cung cấp API chuẩn hóa cho các thông điệp được truyền đi một cách tin cậy và nó hỗ trợ các hệ thống truyền thông điệp middleware) [4].

Các ứng dụng được kết nối tới các bus bằng cách sử dụng bộ điều hướng ứng dụng (application adapter) hoặc một cơ chế hỗ trợ tổ chức thông điệp. Để hỗ trợ SOA thì các services container cần bao gồm những công nghệ liên quan tới webservice cơ bản nhất. Ngoài ra, các thành phần của ESB cũng như cơ chế xử lý các nguồn tài nguyên kết nối phải dựa trên tiêu chuẩn mở để đảm bảo khả năng tương tác cũng như đảm bảo khả năng an ninh, bảo vệ hệ thống.



Hình 2. 2. Một kịch bản của ESB. Một Service Container có thể chứa nhiều dịch vụ và các thành phần khác nhau.

Tất cả các ứng dụng được kết nối tới trung tâm hệ thống hàng đợi thông điệp (message broker) thông qua một interface thống nhất phục vụ cho việc gửi và nhận thông điệp. Các message broker này có thể lưu trữ các thông điệp giúp cho người gửi và người nhận không cần phải kết nối với nhau tại cùng một thời điểm nhất định. Hơn nữa, các message broker này còn có chức năng biến đổi thông điệp truyền đi sao cho phù hợp với các yêu cầu của ứng dụng nhận tin.

1.4. Phân loại ESB Middleware

- ESB dựa trên thông điệp: Hỗ trợ trao đổi thông điệp đồng bộ và không đồng bộ. Có khả năng hỗ trợ tích hợp mở rộng hệ thống, triển khai trên mô hình rộng, đồng thời hỗ trợ đa nền tảng lập trình (Java, C/C++…). Tuy nhiên nó lại tốn chi phí triển khai cài đặt, và cần cấu hình phức tạp.

- ESB dựa trên máy chủ ứng dụng: Nó dựa trên công nghệ tích hợp máy chủ ứng dụng. Phù hợp với hệ thống có định dạng ngôn ngữ XML hoặc Java. Ưu điểm của loại này là dễ sử dụng, dễ cài đặt và thiết lập, và phù hợp với triển khai hệ thống vừa và nhỏ.

1.5. So sánh ESB với các phương pháp tích hợp khác.

Phương pháp tích hợp	**Mô tả**	**Ưu điểm**	**Nhược điểm**
Trao đổi dữ liệu qua tập tin	Nguyên tắc của giải pháp này là một hệ thống sẽ định kỳ xuất dữ liệu ra một tệp tin (đặt trên một thư mục trọng mạng cục bộ hoặc Web nào đó), sau đó hệ thống nhận dữ liệu cũng sẽ định kỳ quét thư mục (thông qua giao thức chuyển tập tin như FTP) hoặc	- Đơn giản, thường được sử dụng trong tích hợp ứng dụng quy mô không lớn và không phức tạp - Cho phép thực hiện cơ chế không đồng bộ, các hệ thống thông tin thực hiện trao đổi dữ liệu không phải chờ nhau và cũng không cần	- Việc trao đổi dữ liệu sẽ chỉ được tiến hành theo một chiều

	truy nhập đường liên kết để tải tập tin về.	sẵn sàng tại mọi thời điểm.	
Trao đổi dữ liệu bằng dịch vụ truyền thông điệp	- Một hệ thống sẽ gửi dữ liệu dưới dạng các thông điệp tới một hệ thống trung gian (ứng dụng quản lý thông điệp - Message Broker) và sau đó được chuyển tiếp tới hệ thống nhận dữ liệu. Đối với các hệ thống dựa trên Java, dịch vụ truyền thông điệp được cung cấp dưới dạng một giao diện lập trình API chuẩn để truy cập và giao tiếp với các hệ thống thông tin.	- Giải pháp này cho phép dữ liệu có thể được chia sẻ hai chiều.	- Giải pháp này chỉ được áp dụng nếu hai hệ thống (cho và nhận dữ liệu) cùng có khả năng sử dụng cùng một loại dịch vụ truyền thông điệp nào đó, ví dụ dịch vụ truyền thông điệp Java (Java Message Service).
Kết nối trực tiếp đến cơ sở dữ liệu	Đây là cách thức mà một hệ thống kết nối và truy cập trực tiếp vào cơ sở dữ liệu của hệ thống khác, với điều kiện được cấp quyền kết nối vào cơ sở dữ liệu (sử dụng các giao thức kết nối cơ sở dữ liệu Database Connectivity như JDBC, ODBC). Trường hợp cơ sở dữ liệu này không cung cấp giao thức kết nối cơ sở dữ liệu cho các hệ thống thông tin	- Giải pháp này cho phép dữ liệu có thể được chia sẻ hai chiều.	- Nếu dữ liệu nhận được từ hệ thống thông tin chia sẻ dữ liệu là dữ liệu thô (trực tiếp, không qua xử lý) thì cần xem xét các yếu tố liên quan đến đường truyền và tính toàn vẹn của dữ liệu nhận về.

	khác, thì có thể phân tích cơ sở dữ liệu và xây dựng Web-service đặt tại đầu 2 hệ thống.		
Trao đổi dữ liệu E-mail	Cùng với việc sử dụng các giao thức chuẩn SMTP, IMAP và POP3, giải pháp này cho phép việc trao đổi, giao tiếp giữa các ứng dụng dựa trên E-mail được thực hiện một cách dễ dàng. Theo đó, mô-đun thực hiện trao đổi dữ liệu sẽ được cài đặt trên hệ thống chia sẻ dữ liệu và thực hiện gửi tới một địa chỉ E-mail định sẵn nội dung dữ liệu (có thể dưới định dạng XML). Hệ thống thông tin nhận dữ liệu sẽ định kỳ kiểm tra E-mail để lấy dữ liệu ra.	- Các vấn đề liên quan đến sự khác biệt về hệ thống mạng, tường lửa, mạng riêng ảo hay độ sẵn sàng của các hệ thống thông tin sẽ gián tiếp được giải quyết nhờ vào tính chất hàng đợi (Queue) của E-mail. - Đạt hiệu quả trong môi trường mạng nội bộ hoặc yêu cầu về an toàn, an ninh thông tin là tối thiểu. - Giải pháp này tương đối dễ thực hiện và cũng cho phép thực hiện cơ chế không đồng bộ, phù hợp với trường hợp không quan tâm đến vấn đề hiệu năng và tốc độ của việc kết nối, trao đổi dữ liệu.	- Không phù hợp với những hệ thống lớn đòi hỏi hiệu năng, tốc độ kết nối và bảo mật dữ liệu
Kết nối qua Web-service	Giải pháp này dựa trên nhiều giao thức như: SOAP, XML/HTTP, RESTful HTTP, CORBA; các phương thức trao đổi liên lạc như: HTTP, JMS, JBI...; các chuẩn dịch vụ Web, bao gồm:	- Giải pháp này có tính toàn diện cao, cung cấp khả năng kết nối, liên thông không chỉ cho dữ liệu mà còn nghiệp vụ, bảo đảm tính toàn vẹn của dữ liệu và an toàn an ninh thông tin.	Phụ thuộc vào cách thức việc hệ thống thông tin chia sẻ dữ liệu xây dựng sẵn các Web-service và cho phép hệ thống thông tin nhận dữ liệu sử dụng các Web-service này hay không. Trong

	SOAP, WS-I Basic Profile, WSDL, WS-Addressing, WS-Policy, WS-ReliableMessaging, WS Security, WS-SecurityPolicy, WS-SecureConversation, và WS-Trust.		một số trường hợp, hệ thống thông tin Địa phương sẽ được bổ sung các bộ chuyển đổi (Adaptor) để giao tiếp với hệ thống thông tin tại Trung ương dựa trên tập hợp các Web-service do hệ thống tại Trung ương cung cấp.

ESB là giải pháp tích hợp tổng hợp toàn bộ những ưu điểm của các phương pháp tích hợp trên. ESB hiện đang là một xu hướng tích hợp cho các hệ thống không chỉ vừa và nhỏ, mà các hệ thống lớn như chính phủ, ngân hàng… cũng đang nghiên cứu và triển khai.

2. Các thành phần chính trong ESB Middleware

ESB có chức năng phải phối hợp với việc tích hợp các nguồn tài nguyên khác nhau và hỗ trợ sự tương tác giữa các nguồn tài nguyên này. Mục tiêu chung là cung cấp thông điệp và tích hợp các hệ thống mà không cần viết code. Do đó, các thành phần của ESB được cấu hình để đi theo một kịch bản mong muốn.

Ý tưởng của ESB là phục vụ triển khai mô hình kiến trúc hướng dịch vụ SOA vì nó cung cấp các cơ chế tổng quát nhất để kết nối tất cả các dịch vụ cần thiết để xây dựng giải pháp cho doanh nghiệp mà không làm ảnh hưởng tới độ tin cậy, tín an toàn, hiệu suất và khả năng mở rộng của hệ thống.

2.1. Định tuyến – Routing

Định tuyến là khả năng quyết định đích đến của một thông điệp trong quá trình vận chuyển thông điệp đó. Các dịch vụ định tuyến (routing services) là thành phần cốt lõi của ESB, nó cho phép tách các nguồn thông điệp từ các điểm đích

Để kích hoạt tính năng định tuyến và một số tính năng giao tiếp khác thì các điểm đầu cuối của thông điệp cần phải được định danh tham chiếu. Có thể sử dụng URI (Uniform

Resource Identifiers) hoặc các WS-Addressing cho các giải pháp ESB để mô tả và xây dựng các dịch vụ web (webservices)

Việc quyết định một thông điệp sẽ được gửi đến địa chỉ nào dựa trên một số điều kiện được xác định và điều phối bởi một số các bộ định tuyến khác nhau.

- Định tuyến dựa trên nội dung của thông điệp và chuyển tiếp chúng đến các kênh khác nhau dựa vào nội dung của thông điệp. Điều này cho phép người gửi có thể gửi tin nhắn mà không cần chỉ định đến một điểm đến chính xác. Đối với thông điệp dạng XML thì có thể sử dụng XPath để phân tích các thành phần của thông điệp để phục vụ cho việc định tuyến.
- Định tuyến dựa vào bộ lọc tin nhắn: Nó sẽ chuyển tiếp tin nhắn tới địa chỉ đích khi nội dung có chứa một tiêu chí nhất định, nếu không thì tin nhắn sẽ bị xóa.
- Định tuyến dựa trên địa chỉ đã được cấu hình sẵn bằng cách sử dụng một danh sách địa chỉ nhận đã được thiết lập từ trước.

Nếu một thông điệp bao gồm nhiều phần, chức năng spliter được sử dụng để chia một tin nhắn thành nhiều tin nhắn thành phần, còn đối với thông điệp XML, ta có thể sử dụng XPath như là một spliter, sau đó thực hiện biến đổi tin nhắn dựa trên XSL (Extensible Stylesheet Language) để tạo ra các thông điệp riêng biệt.

Trái ngược lại với spliter là aggregator – có chức năng thu thập và lưu trữ các thông điệp. Nếu aggregator nhận được một bộ hoàn chỉnh các tin nhắn thành phần liên quan tới nhau thì nó sẽ gửi một thông điệp là tổng hợp của các tin nhắn thành phần đó tới điểm đích đã được cấu hình sẵn.

Ngoài ra, ta có resequencer – là một router – có chức năng thu thập những tin nhắn liên quan với nhau mà không theo thứ tự, sau đó nó sẽ thực hiện chuyển các tin nhắn thành phần này theo đúng thứ tự. Đối với chức năng resequencer, các tin nhắn cần có một thứ tự nhất định.

Các bộ định tuyến khác nhau có thể được kết hợp để tạo ra các luồng gửi và nhận thông điệp phức tạp hơn.

2.2. Phân giải - Mediation

Mediation đề cập đến tất cả các sự chuyển đổi hoặc biên dịch giữa các nguồn tài nguyên khác nhau, bao gồm cả các giao thức vận chuyển (transport protocol), định dạng và nội dung của thông điệp. Những chuyển đổi này rất quan trọng cho việc tích hợp vì các ứng dụng hiếm khi sử dụng cùng một kiểu dữ liệu chung.

- Message Transformation: là khả năng chuyển đổi cấu trúc và định dạng của services yêu cầu thành kiểu cấu trúc và định dạng phù hợp với services cung cấp. Ví dụ: FIX → Text, HL7 → XML, SWIFT → XML

- Protocol Transformation: là khả năng chấp nhận một loại giao thức từ đầu vào (SOAP/JMS) và truyền tải tới services cung cấp thông qua các loại giao thức khác nhau
  Ví dụ: XML/HTTP → CICS/MQ, SOAP/JMS → SOAP/HTTP, XML/HTTPS → IIOP.
- Service Mapping: là khả năng chuyển đổi một service nghiệp vụ thành các thông tin dịch vụ tương ứng.

2.3. Điều hợp – Adapter

Là thành phần quan trọng nhất của ESB, tất cả yêu cầu đi vào và đi ra đều phải thông qua adapter. Adapter cho phép ESB tương tác với nhiều cơ chế đầu ra. Các giải pháp ESB đều cung cấp một loạt các ứng dụng adapters. Các adapter này có thể được sử dụng để dành cho việc giao tiếp với các ứng dụng phổ biến như là Enterprise Resource Planning (ERP), Supply Chain Management (SCM) và Customer Relationship Management (CRM). Những adapter này kết nối với các interface điều chuyển, các API và các cấu trúc dữ liệu được cung cấp bởi các ứng dụng nghiệp vụ, điều này giúp tái sử dụng tài nguyên nghiệp vụ và dữ liệu.

Thông thường, hầu hết các adapter hoạt động theo cùng một cách là giảm thiểu những kỹ năng cần thiết để có thể tái sử dụng lại những tài nguyên của hệ thống đã kết nối. Sử dụng các adapter có sẵn giúp giảm các công việc cần thiết trong quá trình tích hợp các ứng dụng vào kiến trúc hướng dịch vụ SOA.

2.4. An toàn – Security

Cơ sở hạ tầng của một hệ thống truyền tin của một doanh nghiệp cần phải được bảo vệ. Điều đó có nghĩa là ESB cần có khả năng mã hóa và giải mã nội dung của thông điệp, thực hiện xử lý xác thực, kiểm soát truy cập các thiết bị đầu cuối và sử dụng các cơ chế bảo vệ an toàn liên tục.

2.5. Quản lý – Managerment

Một hệ thống ESB cung cấp các cơ chế ghi log và kiểm tra (audit) để phục vụ mục đích là theo dõi hạ tầng, các kịch bản tích hợp và kiểm soát quá trình vận hành hệ thống. Do đó ESB cần phải có một cơ chế trung tâm để cấu hình và quản trị các bus. Ngoài ra ESB cần hỗ trợ thêm các công cụ đo lường phục vụ việc ghi log.

2.6. Điều phối quy trình - Process Orchestration

Một hệ thống ESB có thể cung cấp các chức năng để thực thi các mô hình nghiệp vụ được mô tả bằng Web Services Business Process Execution Language (WS-BPEL). Các chức năng này được điều khiển bởi mô tả của nghiệp vụ, sau đó nó sẽ phối hợp với các bus đã được kết nối với tài nguyên để thực thi yêu cầu.

2.7. Xử lý các sự kiện phức tạp – Complex Event Processing

Một hệ thống ESB có thể bao gồm thêm các cơ chế để giải thích các sự kiện tương quan cũng như các sự kiện kết hợp với nhau khi có một thông báo trên một kênh truyền tải nào đó.

2.8. Công cụ tích hợp

Đối với những nhà phát triển ESB chuyên nghiệp, cần có những công cụ phát triển ESB cũng như kiểm thử với giao diện trực quan người dùng.

3. Một số ESB Middleware

Hiện nay có rất nhiều những sản phẩm phần mềm hỗ trợ ESB với nhiều tính năng đa dạng và phong phú, do đó ta cần đánh giá mục đích và nhu cầu, sau đó cần đánh giá sản phẩm nào là thích hợp để phát triển nhất. Để lựa chọn một cách đúng đắn, ta cần tuân thủ một số tiêu chí sau: [6]

- Tính dễ sử dụng: Việc cài đặt có phức tạp hay không? Môi trường phát triển có trực quan không?
- Vấn đề bảo trì: Việc bảo trì diễn ra như nào? Có công cụ giám sát trực quan hay không?
- Hỗ trợ: Những tùy chọn hỗ trợ nào được cung cấp (hotline 24/7, email…), việc hỗ trợ có đảm bảo như chúng ta mong muốn hay không?

- Cộng đồng: Có cộng đồng người sử dụng hay không? Sản phẩm có được hỗ trợ bởi nhiều công ty hay không?
- Tính cơ động: Có khả năng thay đổi tùy vào nhu cầu người sử dụng hay không?
- Chức năng: các chức năng yêu cầu có được hỗ trợ hay không?
- Tính mở rộng: Ta có thể mở rộng sản phẩm được không?
- Kết nối: Các bộ chuyển đổi adapter có sẵn sàng theo nhu cầu nghiệp vụ hay không? Có dễ dàng tạo ra bộ chuyển đổi riêng hay không?
- Giá thành: Giá thành sản phẩm và bảo trì
- Bản quyền: Việc nâng cấp, hạ cấp có khả thi, miễn phí hay không?

Phần mềm hỗ trợ trục tích hợp ESB được chia ra làm 2 loại chính đó là: Loại có bản quyền và loại mã nguồn mở. Bảng dưới so sánh các tiêu chí của 2 loại này.

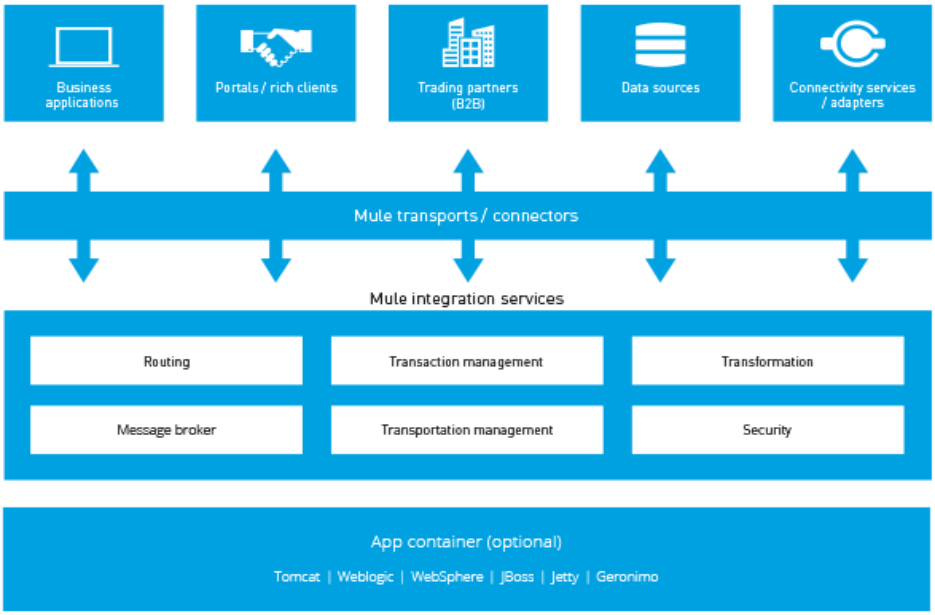
Tiêu chí	Mã nguồn mở	Bản quyền
Dễ sử dụng	- Cài đặt dễ dàng - Sử dụng sau vài phút - Nền tảng hợp nhất	- Cài đặt phức tạp - Cần hỗ trợ, tư vấn
Bảo trì, giám sát	- Công cụ hỗ trợ yếu - Không cần phân tích mã nguồn	- Công cụ hỗ trợ mạnh mẽ - Không cần phân tích mã nguồn
Chức năng	- Các tính năng tích hợp và một vài tính năng khác	- Hỗ trợ nhiều tính năng tích hợp và nhiều tính năng khác
Hỗ trợ	- Hỗ trợ 24/7 của nhà cung cấp. - Sự bảo đảm thấp	- Hỗ trợ 24/7 của nhà cung cấp. - Đảm bảo về chất lượng, mức độ dịch vụ, triển khai diện rộng.
Cơ động	- Mã nguồn mở, thay đổi theo mong muốn	- Tạo yêu cầu thay đổi, thời gian lâu, chi phí phát sinh
Kết nối	- Bộ chuyển đổi công nghệ và nghiệp vụ	- Bộ chuyển đổi công nghệ và nghiệp vụ
Cộng đồng	- Dựa trên dự án mã nguồn mở - Có cộng đồng riêng	- Trả chi phí để có hỗ trợ - Thủ tục khi tham gia diễn đàn, tuy nhiên lại không thực sự có ích
Mở rộng	- Dựa trên các tiêu chuẩn	- Khó can thiệp sâu, hoặc phải trả thêm chi phí

Bản quyền	- Có nâng cấp, hạ cấp - Chi phí thấp	- Chi phí rườm rà, phức tạp
Giá thành	- Thấp	- Khá cao

3.1. Mule ESB

Mule ESB là một trong những trục tích hợp ESB mã nguồn mở thành công đầu tiên. Mule ESB là một trục tích hợp tinh khiết.

Kiến trúc Mule ESB[1]

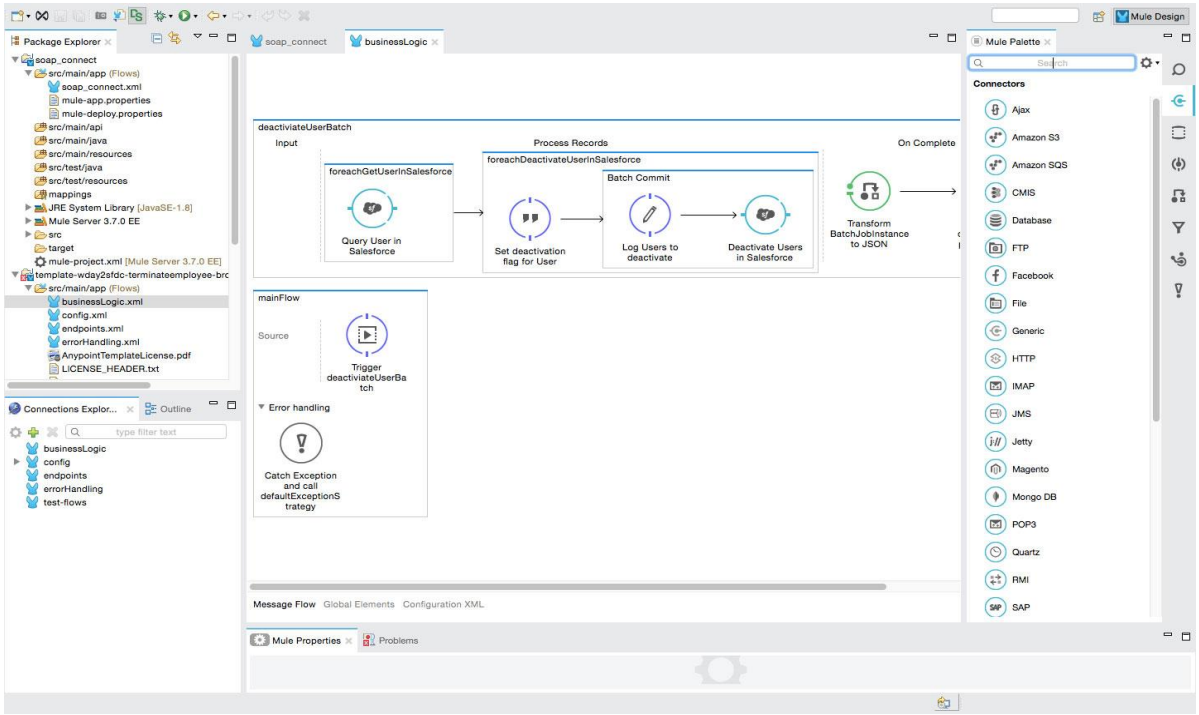


Hình 2. 3. Kiến trúc Mule ESB

- Tạo, lưu trữ và tạo ra các dịch vụ để tái sử dụng, và sử dụng ESB như một chỗ chứa dịch vụ
- Phân giải dữ liệu: che chắn các dịch vụ từ các định dạng tin nhắn, các giao thức và logic nghiệp vụ riêng biệt. Cho phép lời gọi dịch vụ độc lập.
- Định tuyến tin nhắn: định tuyến, lọc, tổng hợp và sắp xếp lại các thông điệp dựa trên nội dung và quy tắc.
- Chuyển đổi dữ liệu: Trao đổi dữ liệu qua các định dạng khác nhau và các giao thức truyền tải.

[1] *https://www.mulesoft.com/resources/esb/what-mule-esb*

- Mule ESB cho phép tái sử dụng các thành phần quan trọng. Không giống như các frameworks khác, Mule cho phép sử dụng các thành phần hiện có mà không cần bất kỳ thay đổi nào cả. Các thành phần (components) của Mule không đòi hỏi bất kỳ đoạn mã cụ thể nào cũng như không yêu cầu bất kỳ lập trình API nào để thực thi. Logic nghiệp vụ được giữ tách biệt hoàn toàn với logic truyền tin.
- Thông điệp có thể ở bất kỳ định dạng nào, có thể từ SOAP cho đến kiểu nhị phân. Mule không ép buộc bất kỳ rằng buộc thiết kế nào đối với lập trình viên, chẳng hạn như việc thống nhất định dạng dịch vụ kiểu XML hay WSDL.
- Có thể triển khai Mule trong một loạt cấu trúc liên kết, không chỉ riêng ESB. Mule giúp tăng năng suất dự án, cung cấp các tính năng bảo mật, khả năng thích nghi với những thay đổi của nghiệp vụ khi cần thiết.
- Kiến trúc Mule là kiến trúc hướng sự kiện giúp nó có khả năng mở rộng cao.
- Mule ESB có công cụ thiết kế luồng dữ liệu tích hợp là Anypoint Studio.



Hình 2. 4 Giao diện Anypoint Studio

Ưu điểm:

- Cung cấp nhiều chức năng có chất lượng tốt như những trục tích hợp khác.
- Dễ dàng cài đặt và sử dụng, dựa trên nền tảng Eclipse nên thân thiện với người dùng
- Nhẹ, dễ dàng mở rộng và tùy chỉnh

- Ngoài phiên bản miễn phí có thêm lựa chọn cho bản thương mại, cung cấp thêm một số tính năng nâng cao với giá thành hợp lý
- Có công cụ Anypoint Studio giúp dễ dàng phát triển ESB – nó giúp dễ dàng kéo thả các thành phần tạo nên flow chuyển đổi dữ liệu
- Có khả năng chạy trực tiếp ESB từ IDE

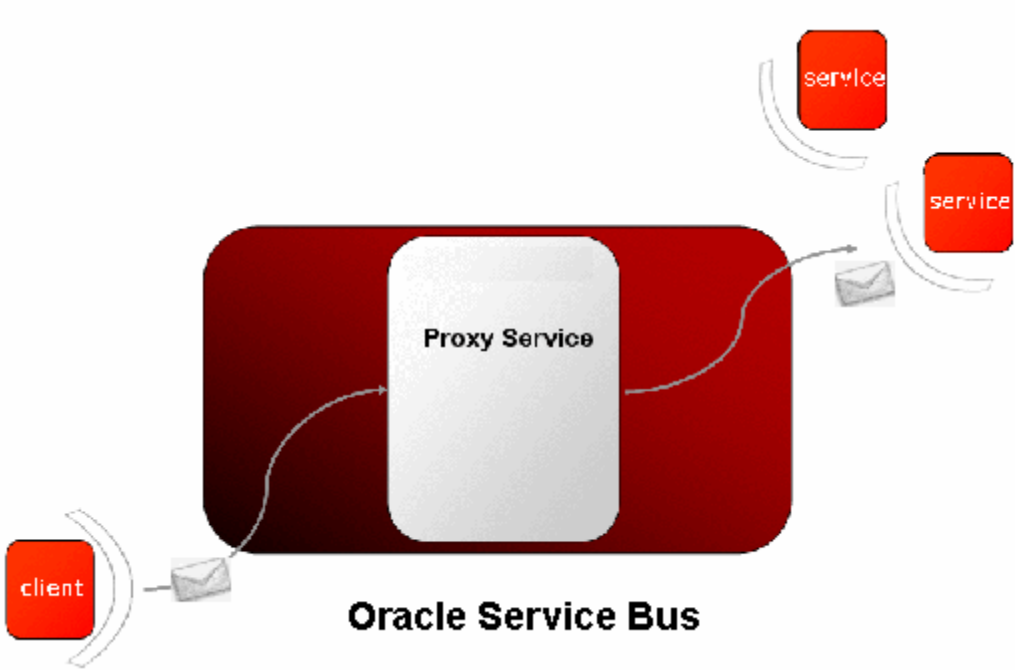
Nhược điểm:

- Chỉ sử dụng để triển khai hệ thống vừa và nhỏ

3.2. Oracle Service Bus

Là trục tích hợp ESB của chính Oracle. Nó là một thành phần của Oracle Fusion Middleware – một bộ công cụ tích hợp mạnh mẽ.

Kiến trúc Oracle Service Bus[2]



Hình 2. 5. Kiến trúc Oracle Service Bus

- Oracle Service Bus cung cấp các dịch vụ chuyển phát tin dựa trên các tiêu chuẩn bao gồm SOAP, HTTP và JMS.
- Nó được thiết kế để truyền tải thông điệp với độ chính xác cao và đảm bảo đến các máy chủ cung cấp và tiếp nhận dịch vụ. Nó hỗ trợ XML như là một kiểu dữ liệu nguyên thủy đồng thời cung cấp các giải pháp chuyển đổi thành các kiểu dữ liệu khác.

[2] *https://docs.oracle.com/cd/E14571_01/doc.1111/e15020/architecture_overview.htm#OSBCA141*

- Oracle Service Bus cho phép thiết lập mối quan hệ giữa người sử dụng và nhà cung cấp dịch vụ, đồng thời duy trì điểm kiểm soát và giám sát an ninh tập trung.
- Oracle Service Bus là một trung gian xử lý các yêu cầu dịch vụ đến, xác định logic định tuyến và biến đổi các thông điệp để tương thích với các bên nhận dịch vụ khác. Nó nhận tin nhắn thông qua một giao thức truyền tải như HTTP(s), JMS, FPT, và gửi các thông điệp qua cùng một giao thức truyền tải khác

Ưu điểm:

- Cung cấp đầy đủ các chức năng tích hợp
- Mạnh mẽ và ổn định, được Oracle phát triển trong một thời gian dài
- Là một thành phần của Fusion Middleware nên dễ dàng kết nối với các thành phần giả pháp khác như là: SOA, Coherence, Complex Event Processing, BEPL Process Manager, Enterprise Messaging Service, Service Registry, và nhiều hơn thế.
- Hầu hết các sản phẩm đều có trình biên tập đồ họa
- Sự hỗ trợ luôn sẵn sàng cho hầu hết các thỏa thuận mức độ dịch vụ
- Triển khai trên hệ thống doanh nghiệp lớn, có sự chuyên nghiệp

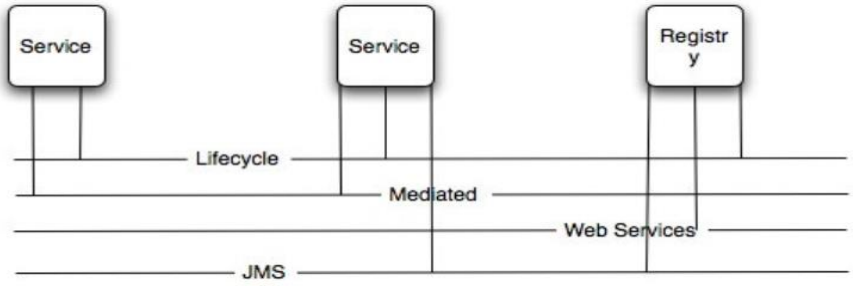
Nhược điểm:

- Giá thành rất cao
- Dung lượng sản phẩm rất cao (có thể vượt quá 20Gb)
- Cài đặt khó khăn
- Chiếm rất nhiều tài nguyên
- Cần có cơ sở hạ tầng tốt mới triển khai được.

3.3. JBoss ESB

JBoss ESB là thế hệ tiếp theo của EAI. JBoss cung cấp các chức năng như: giám sát quy trình kinh doanh (Business Process Monitoring), môi trường phát triển tích hợp (Integrated Development Environment), giao diện trực quan người dùng (Human Workflow User Interface), quản lý quy trình nghiệp vụ (Business Process Management), công cụ kết nối (Connectors), quản lý truyền thông (Transaction Manager), An ninh hệ thống (Security), Messaging Service, kiến trúc phân tán (Distributed Computing Architecture).[3]

[3] *http://jbossesb.jboss.org/resources/WhatIsAnESB.html*



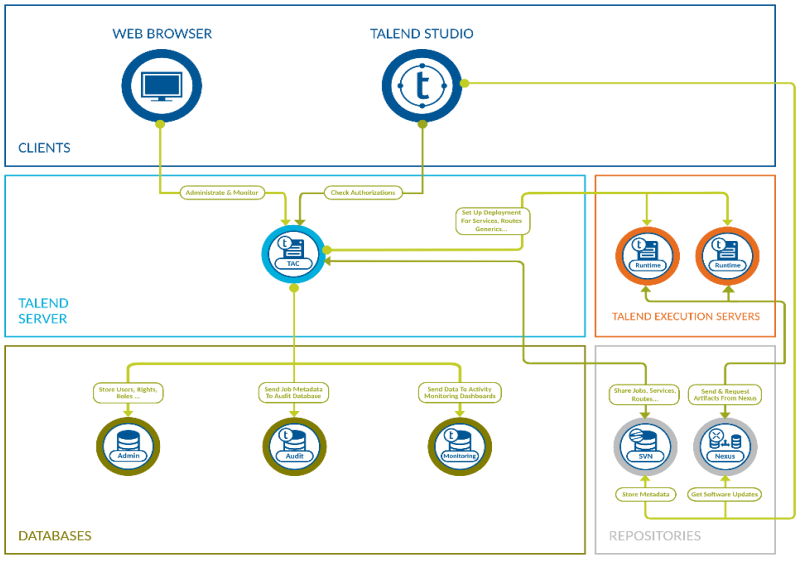
Hình 2. 6 Kiến trúc của JBoss ESB

Kiến trúc của JBossESB là một phần của SOI (Service Oriented Infrastructure). JBoss cung cấp giao diện trực quan người dùng khiến cho quá trình thiết kế, tích hợp và kiểm thử trở nên dễ dàng và thuận tiện, đạt hiệu quả cao.

3.4. Talend Open Studio for ESB

Talend ESB là một phần của bộ công cụ Talend. Tất cả các công cụ của bộ Talend được xây dựng trên nền tảng Eclipse, do đó trực quan đối với việc sử dụng trong Eclipse vẫn giữ nguyên. Talend cung cấp việc thiết kế đồ họa luồng dữ liệu, điều này cho phép thực thi dễ dàng và hiệu quả trong các chương trình tích hợp. Tuy nhiên, chúng ta vẫn có thể viết và thực hiện tích hợp tùy biến về mặt logic đối với từng dự án.

Kiến trúc Talend Open Studio for ESB[4]



Hình 2. 7. Kiến trúc Talend Open Studio for ESB

[4] *https://help.talend.com/reader/eT1URirEr8V_hj9ytQOHvQ/YBUGJ~FNKKLiK_a5pmtvXw*

- Khối Client: bao gồm Talend Studio – công cụ thực hiện quá trình tích hợp dữ liệu hoặc xử lý dịch vụ dữ liệu, phân giải và định tuyến dữ liệu.
- Talend Execution Servers: chứa một hoặc nhiều Talend Runtimes (nơi thực hiện) được triển khai bên trong hệ thống. Talend Runtime cho phép triển khai và thực hiện các Jobs, cấu hình định tuyến và dịch vụ được tạo ra trong Studio. Tất cả các phiên bản của Talend Runtime sẽ giao tiếp với nhau thông qua các Service Locator để xác định một trong nhiều khả năng triển khai và thực hiện chúng.
- Khối Databases: Đại diện cho các cơ sở dữ liệu dùng để theo dõi các hoạt động sử dụng để thu thập thông tin đăng nhập hoặc việc thực hiện các quy trình dữ liệu của người dùng, đồng thời nó dùng để theo dõi các cuộc gọi dịch vụ.
- Quá trình xử lý dữ liệu được ghi lại bằng việc sử dụng các thành phần *tFlowMeterCatcher, tStatCatcher, tLogCatcher*, và để thực hiện tự động các thành phần đó mà không cần sử dụng chúng thì có thể sử dụng chức năng Stats & Logs
- Chức năng giám sát hoạt động dịch vụ: cho phép người dùng theo dõi các cuộc gọi dịch vụ. Nó cung cấp giám sát và hợp nhất thông tin về sự kiện mà người dùng cuối có thể thao tác.

Ưu điểm:

- Talend ESB được xây dựng dựa trên một vài tiêu chuẩn thực tế trong môi trường tích hợp như Apache Camel, Apache CXF, Apache Karaf và Apache Zookeeper.
- Bên cạnh các bộ kết nối dành cho Apache như JMS, HTTP hoặc FTP thì rất nhiều bộ chuyển đổi B2B cũng được cung cấp, ví dụ như là Alfresco, Jasper, SAP, Saleforce hoặc các hệ thống máy chủ.
- Mặc định có khoảng hơn 500 bộ kết nối được bao gồm trong bộ sản phẩm.

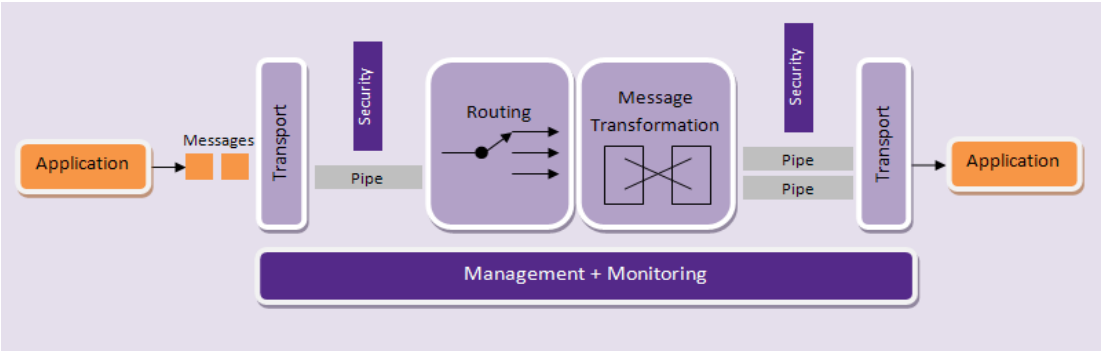
Nhược điểm:

- Có một điểm bất lợi đó là Talend IDE đòi hỏi phần cứng ở mức độ cao hơn so với các đối thủ của nó. Chúng ta không nên cài đặt Talend trên những máy tính quá yếu.
- Một điểm yếu nữa của Talend đó là thiếu các tính năng quản trị SOA.

3.5. WSO2 ESB

WSO2 cung cấp toàn bộ các thành phần hỗ trợ cho việc tích hợp ứng dụng như: Bussiness Process Server, Bussiness Rule Server, Bussiness Activity Monitor hoặc Governace Registry.

Kiến trúc WSO2 ESB:[5]



Hình 2. 8 Kiến trúc WSO2 ESB

- Việc vận chuyển dữ liệu sẽ thông qua một kênh truyền tin – nơi sẽ xử lý các khía cạch chất lượng dịch vụ như là vấn đề bảo mật.
- WSO2 ESB hoạt động ở 2 chế độ :Phân giải tin nhắn và vận chuyển dịch vụ qua các proxy khác nhau.
- Cả 2 dạng chuyển đổi tin nhắn và định tuyến có thể coi là một đơn vị duy nhất. Như trên hình 2.7 ta có thể thấy không có sự tách biệt rõ ràng giữa các thành phần chuyển đổi tin nhắn và các thành phần định tuyến. Trong WSO2 thì đây được gọi là mediation framework.
- Một số việc biến đổi xảy ra trước khi quyết định thực hiện việc định tuyến, trong khi một số khác xảy ra sau khi việc định tuyến kết thúc.

Ưu điểm:

- Toàn bộ nền tảng WSO2 có thể được cài đặt rất dễ dàng và cung cấp một studio phát triển dựa trên Eclipse.
- Giống như Talend và FuseSource, WSO2 sử dụng các dự án mã nguồn mở như Apache Synapse (lightweight ESB), Axis (Web Service Implementation) hoặc ODE (Business Process Engine) vào các thành phần của nó.
- Bên cạnh Talend, WSO2 là nhà cung cấp duy nhất cung cấp một bộ phần mềm hoàn chỉnh dựa trên cơ sở mã nguồn và môi trường phát triển riêng biệt. Do đó, không có gì cản trở quá trình phát phần mềm, từ lúc bắt đầu các tính năng đơn giản nhất, cho đến lúc hệ thống đã có những tính năng phức tạp khác.

[5] *https://docs.wso2.com/display/ESB490/Architecture*

Nhược điểm:

- Điểm yếu là công cụ đồ họa. Nó hỗ trợ tất cả các thành phần cơ bản trong việc tích hợp, nhưng lại không có công cụ trực quan để thiết kế luồng dữ liệu như các đối thủ cạnh tranh khác.

4. Kết luận

ESB giúp mở rộng khả năng triển khai SOA cho hệ thống các doanh nghiệp. Nó không những cung cấp một mô hình chung để triển khai, quản lý cũng như quản trị các ứng dụng, mà nó còn làm giảm gánh nặng thiết kế khái niệm đối với bất kỳ người dùng nào, cải thiện khả năng tái sử dụng kiến trúc.

Lợi ích ESB:

- Thích nghi nhanh và rẻ hơn với hệ thống đang tồn tại
- Tăng sự mềm dẻo; dễ dàng hơn cho việc thay đổi như là thay đổi yêu cầu.
- Dựa trên các chuẩn
- Mở rộng từ giải pháp point–to-point để triển khai rộng rãi cho các doanh nghiệp (Bus phân phối)
- Định nghĩa trước các loại dịch vụ sẵn sàng cho người dùng
- Thêm cấu hình chứ không phải tích hợp mã hóa
- Không có các rules-engine trung tâm, không có môi giới trung tâm

Bất lợi chính của ESB

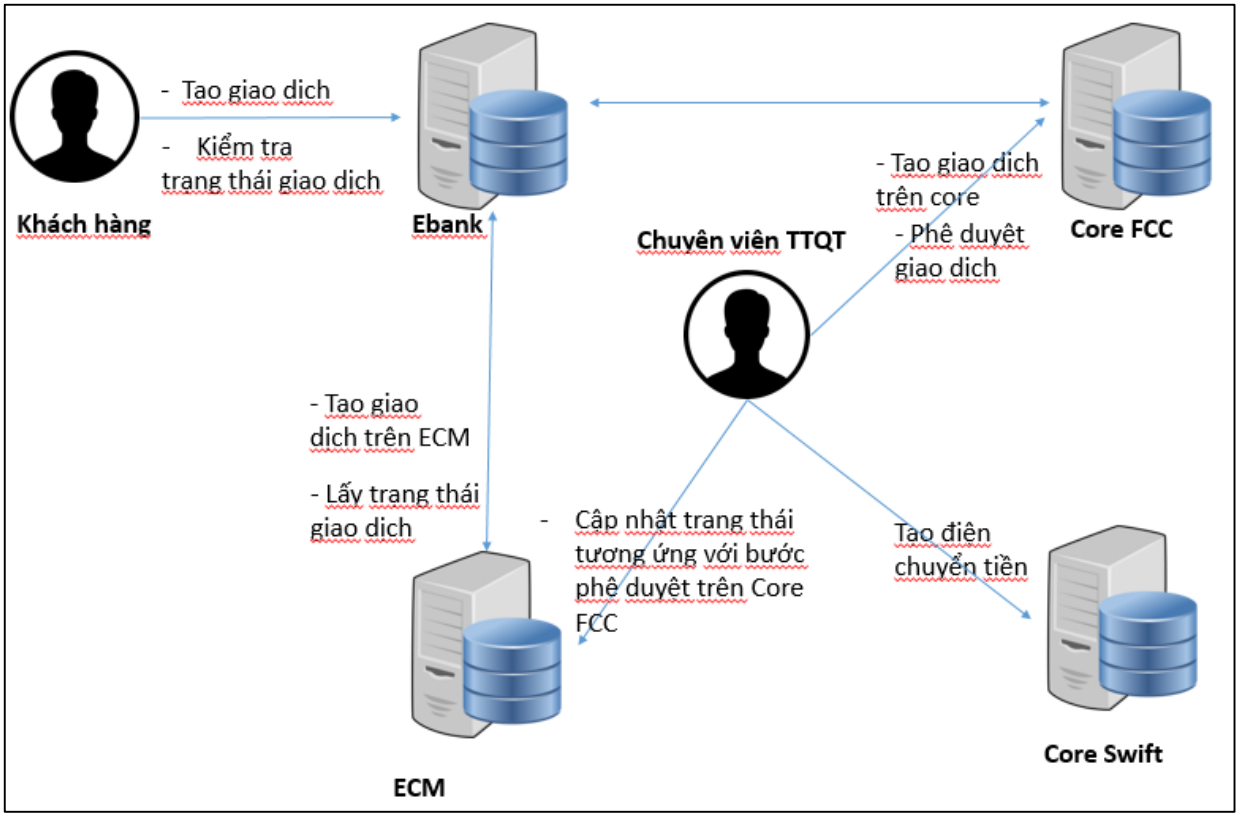
- Thường đòi hỏi mô hình thông điệp doanh nghiệp, kết quả là thêm chi phí quản lý. Những khó khăn tiềm năng khi tích hợp nhiều hệ thống tạp nham để cộng tác thông qua các tiêu chuẩn thông điệp
- Yêu cầu sự quản lý liên tục các phiên bản thông điệp để đảm bảo lợi ích dự kiến của loose coupling. Sự quản lý không chính xác, không đầy đủ hoặc không cần thiết của phiên bản thông điệp có thể dẫn đến sự phụ thuộc chặt chẽ thay vì mục đích là đạt được loose coupling.
- Đòi hỏi phần cứng nhiều hơn là các thông điệp point–to-point đơn giản.
- Kỹ năng phân tích trung gian cần để cấu hình, quản lý và thực hiện ESB.
- Tăng độ trễ gây ra bởi việc các thông điệp phải đi qua thêm các lớp của ESB, đặc biệt khi so sánh với giao tiếp qua mô hình điểm-điểm. Độ trễ tăng lên một phần cũng là do thêm phần xử lý tài liệu XML (ESB thường sử dụng XML là ngôn ngữ giao tiếp).

CHƯƠNG 3. ỨNG DỤNG ESB MIDDLEWARE ĐỂ TÍCH HỢP DỊCH VỤ TẠI NGÂN HÀNG TPBANK

1. Đặt vấn đề

1.1. Thực trạng tại TPBank

Ngày nay với sự phát triển giao thương giữa các tổ chức, cá nhân trong nước với các tổ chức, cá nhân ở nước ngoài thì những hoạt động chuyển tiền quốc tế diễn ra ngày càng nhiều và thậm chí có tới hàng trăm, hàng nghìn giao dịch chuyển tiền quốc tế được thực hiện. Ngân hàng TPBank cũng không ngoại lệ.



Hình 3. 1. Thực trạng ngân hàng TPBank

Các hệ thống tham gia:

- Hệ thống Ebank: Giúp khách hàng thao tác trên phần mềm ebank mà không cần tới các điểm giao dịch để thực hiện.

- Hệ thống ECM (Enterprise Content Managerment): Là hệ thống lưu trữ thông tin giao dịch bao gồm cả các chứng từ, tài liệu đi kèm của giao dịch để thực hiện việc lưu kho, thống kê, báo cáo.
- Hệ thống Core FCC: là hệ thống Core Banking thực hiện các chức năng chính như phê duyệt tính hợp lệ giao dịch và phê duyệt thực hiện chuyển tiền.
- Hệ thống Core Swift: là hệ thống tạo ra điện chuyển tiền (gọi là file swift).

Hế thống chuyển tiền quốc tế trên ebank hiện đang được thực hiện như sau:

Khách hàng sẽ thực hiện tạo yêu cầu chuyển tiền trên hệ thống ngân hàng điện tử Ebank mà không cần tới giao dịch tại quầy. Sau khi xác nhận chuyển tiền, hệ thống ebank sẽ thực hiện gọi service tới hệ thống ECM để thực hiện tạo giao dịch và các thông tin liên quan tợi giao dịch đó. Sau đó, chuyên viên phòng thanh toán quốc tế (TTQT) sẽ thực hiện kiểm tra tính hợp lệ của yêu cầu chuyển tiền. Nếu không hợp lệ, chuyên viên TTQT sẽ thực hiện trả lại giao dịch hoặc hủy yêu cầu trên hệ thống ECM. Khi có lệnh trả về (hoặc hủy), hệ thống ECM sẽ gọi tới service của Ebank và tại đây sẽ thông báo cho khách hàng biết là giao dịch không hợp lệ cùng với lý do. Nếu giao dịch hợp lệ thì chuyên viên TTQT sẽ thực hiện nhập thông tin giao dịch đó trên hệ thống Core Banking FCC rồi thực hiện chuyển phê duyệt các cấp. Khi phê duyệt đồng ý chuyển tiền, hệ thống Core FCC sẽ tạo ra một số REF, chuyên viên TTQT sẽ nhập các thông tin cần thiết và số REF này lên hệ thống Core SWIFT để sinh ra một file điện chuyển tiền. Sau khi đã có số REF và file điện SWIFT chuyển tiền thì chuyên viên TTQT sẽ thực hiện hoàn tất thủ tục trên Core FCC là giao dịch chuyển tiền quốc tế sẽ được thực hiện. Sau khi giao dịch hoàn tất, chuyên viên TTQT sẽ thực hiện hoàn thành giao dịch trên hệ thống ECM để thực hiện chuyển lưu kho giao dịch này. Tại đây, một lần nữa hệ thống ECM sẽ gọi tới service Ebank thông báo giao dịch đã được thực hiện và thông báo tới khách hàng và hoàn tất việc chuyển tiền quốc tế.

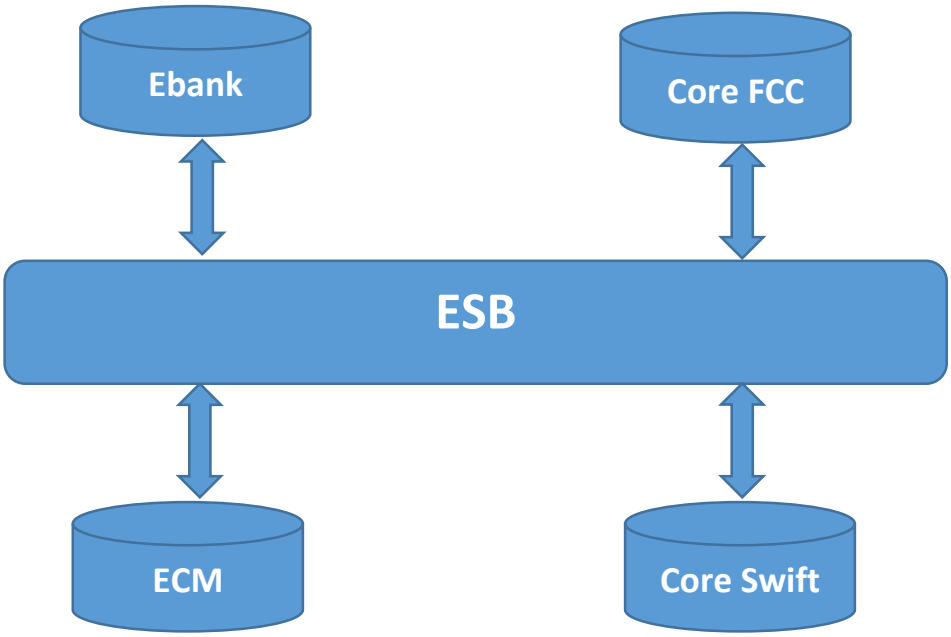
1.2. Bài toán đặt ra

Như ta đã thấy ở mục trên, các hệ thống hoạt động khá là riêng rẽ, chưa có tính thống nhất với nhau, hơn nữa việc chuyển tiền quốc tế mất khá nhiều công đoạn và thời gian liên quan đến nhập liệu, xử lý của chuyên viên phòng TTQT, đồng thời việc kiểm tra tính hợp lệ cũng như chuyển phê duyệt các cấp cũng mất khá nhiều thời gian và công sức, ngoài ra còn chưa kể đến việc sai sót trong quá trình nhập dữ liệu từ hệ thống này sang hệ thống khác (từ ECM sang Core FCC) dễ xảy ra lỗi. Do đó, mục tiêu là cần tích hợp các hệ thống này với nhau thành một hệ thống có tính thống nhất, giảm thiểu

công đoạn nhập liệu đứt đoạn của người dùng, nâng cao tính chính xác và giảm bớt thời gian cũng như công sức của chuyên viên TTQT. Vì vậy luận văn sẽ trình bày giải pháp tích hợp các hệ thống trên thành một thể thống nhất sử dụng ứng dụng ESB Middleware.

2. Giải pháp tích hợp dịch vụ tại TPBank

2.1. Kiến trúc hệ thống tích hợp dịch vụ



Hình 3. 2. Kiến trúc hệ thống tích hợp

Các hệ thống EBank, ECM, Core FCC và Core SWIFT tích hợp với ESB thông qua HTTP Webservice, truy cập vào hệ CSDL dùng chung.

- Khi có một lệnh chuyển tiền trên Ebank, hệ thống này sẽ lưu thông tin lệnh, đồng thời gọi tới hệ thống lưu trữ ECM và sẽ tạo một giao dịch có thông tin tương ứng tại ECM.
- Sau khi tạo giao dịch thành công trên ECM thì hệ thống ECM sẽ gọi tới Core FCC và tạo giao dịch chuyển tiền tương ứng.
- Khi giao dịch hợp lệ, đầy đủ thông tin xác thực và được phê duyệt đồng ý trên Core FCC thì hệ thống Core FCC sẽ thực hiện yêu cầu tới hệ thống ECM thực hiện cập nhật trạng thái tương ứng cho giao dịch, và yêu cầu hệ thống Core Swift để sinh ra điện chuyển tiền.

- Khi điện được chuyển đi, lúc này Core FCC sẽ thực hiện cập nhật thông tin giao dịch tương ứng trên hệ thống ECM để chuyển về quy trình lưu kho, đồng thời hệ thống Ebank sẽ lấy trạng thái giao dịch khi khách hàng thao tác trên Ebank.

2.2. Đặc tả giải pháp

2.2.1. Yêu cầu cụ thể

Mục tiêu của trục tích hợp ESB là giúp cho các hệ thống có khả năng kết nối và trao đổi thông tin dễ dàng hơn, giảm thiểu thao tác của người sử dụng giúp cho tăng tính đúng đắn và nhanh chóng của giao dịch.

Yêu cầu hệ thống sau khi tích hợp là chuyên viên TTQT sẽ thao tác phê duyệt giao dịch chính trên hệ thống Core FCC mà không cần chuyển qua việc hoàn tất thông tin trên các hệ thống khác (hệ thống lưu trữ ECM hay Core SWIFT), nói cách khác: hệ thống Core FCC là trung tâm xử lý nghiệp vụ chính của toàn bộ bài toán.

Như vậy ta thấy chuyên viên TTQT sẽ giảm thiểu thời gian nhập liệu cũng như việc phải phê duyệt thao tác trên các hệ thống khác nhau.

2.2.2. Đặc tả các dịch vụ và chức năng

Các dịch vụ và chức năng chính:

- Khi khách hàng thực hiện một lệnh chuyển tiền trên Ebank, hệ thống này sẽ lưu thông tin lệnh, đồng thời gọi tới hệ thống lưu trữ ECM và sẽ tạo một giao dịch có thông tin tương ứng tại ECM.
- Sau khi tạo giao dịch thành công trên ECM thì hệ thống ECM sẽ gọi tới Core FCC và tạo giao dịch chuyển tiền tương ứng. Còn nếu không hợp lệ, giao dịch sẽ bị trả lại từ hệ thống ECM trở về Ebank để thông báo cho khách hàng biết.
- Khi giao dịch hợp lệ, đầy đủ thông tin xác thực và được phê duyệt đồng ý trên Core FCC thì hệ thống Core FCC sẽ thực hiện kết nối tới hệ thống ECM đê thực hiện cập nhật trạng thái tương ứng cho giao dịch, và kết nối tới hệ thống Core Swift để sinh ra điện chuyển tiền.

- Khi điện được chuyển đi, người dùng sẽ hoàn tất giao dịch chuyển tiền trên Core FCC, Core FCC sẽ thực hiện cập nhật thông tin giao dịch tương ứng trên hệ thống ECM, đồng thời hệ thống Ebank sẽ lấy trạng thái giao dịch khi khách hàng thao tác trên Ebank.

2.2.3. Lựa chọn công nghệ ESB Middleware

Dựa trên khảo sát và đánh giá của 5 công cụ tích hợp ESB Middleware ở mục 3 của chương 2, luận văn sẽ chọn giải pháp tích hợp ESB của Mule (Mule ESB và Anypoint Studio) để thực hiện tích hợp dữ liệu giữa các hệ thống trong ngân hàng lõi để giải quyết bài toán chuyển tiền doanh nghiệp quốc tế.

3. Xây dựng hệ thống thử nghiệm và đánh giá

3.1. Môi trường thực nghiệm

Trục tích hợp ESB sẽ sử dụng công cụ Mule ESB cùng Anypoint Studio (dùng để thiết kế luồng dữ liệu) và được vận hành trên máy chủ cài hệ điều hành Windows Server 2012 Profesional. Cơ sở dữ liệu sử dụng SQL Server 2014 Enterprise. Môi trường sử dụng Java 1.6 để tránh xung đột các thư viện với các hệ thống khác nhau.

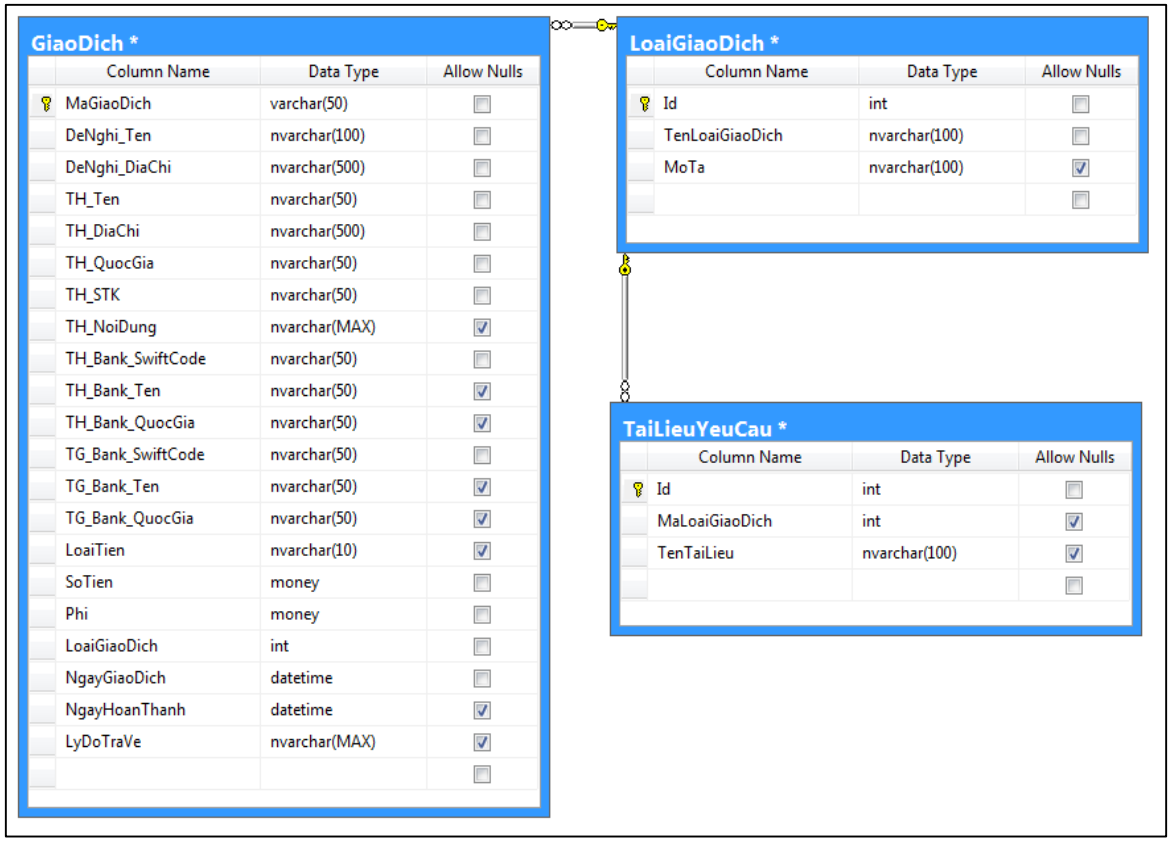
3.2. Luồng thông tin trao đổi

- Khi khách hàng thực hiện một lệnh chuyển tiền trên Ebank, hệ thống này sẽ lưu thông tin lệnh, đồng thời gọi tới hệ thống lưu trữ ECM và sẽ tạo một giao dịch có thông tin tương ứng tại ECM.
- Sau khi tạo giao dịch thành công trên ECM thì hệ thống ECM sẽ gọi tới Core FCC và tạo giao dịch chuyển tiền tương ứng. Còn nếu không hợp lệ, giao dịch sẽ bị trả lại từ hệ thống ECM trở về Ebank để thông báo cho khách hàng biết.
- Khi giao dịch hợp lệ, đầy đủ thông tin xác thực và được phê duyệt đồng ý trên Core FCC thì hệ thống Core FCC sẽ thực hiện kết nối tới hệ thống ECM đê thực hiện cập nhật trạng thái tương ứng cho giao dịch, và kết nối tới hệ thống Core Swift để sinh ra điện chuyển tiền.
- Khi điện được chuyển đi, người dùng sẽ hoàn tất giao dịch chuyển tiền trên Core FCC, Core FCC sẽ thực hiện cập nhật thông tin giao dịch tương ứng trên hệ thống ECM, đồng thời hệ thống Ebank sẽ lấy trạng thái giao dịch khi khách hàng thao tác trên Ebank.

3.3. Mô hình hóa dữ liệu

- Đối với hệ thống Ebank:

Hình 3.3 thể hiện các bảng chính được sử dụng trong quá trình tích hợp

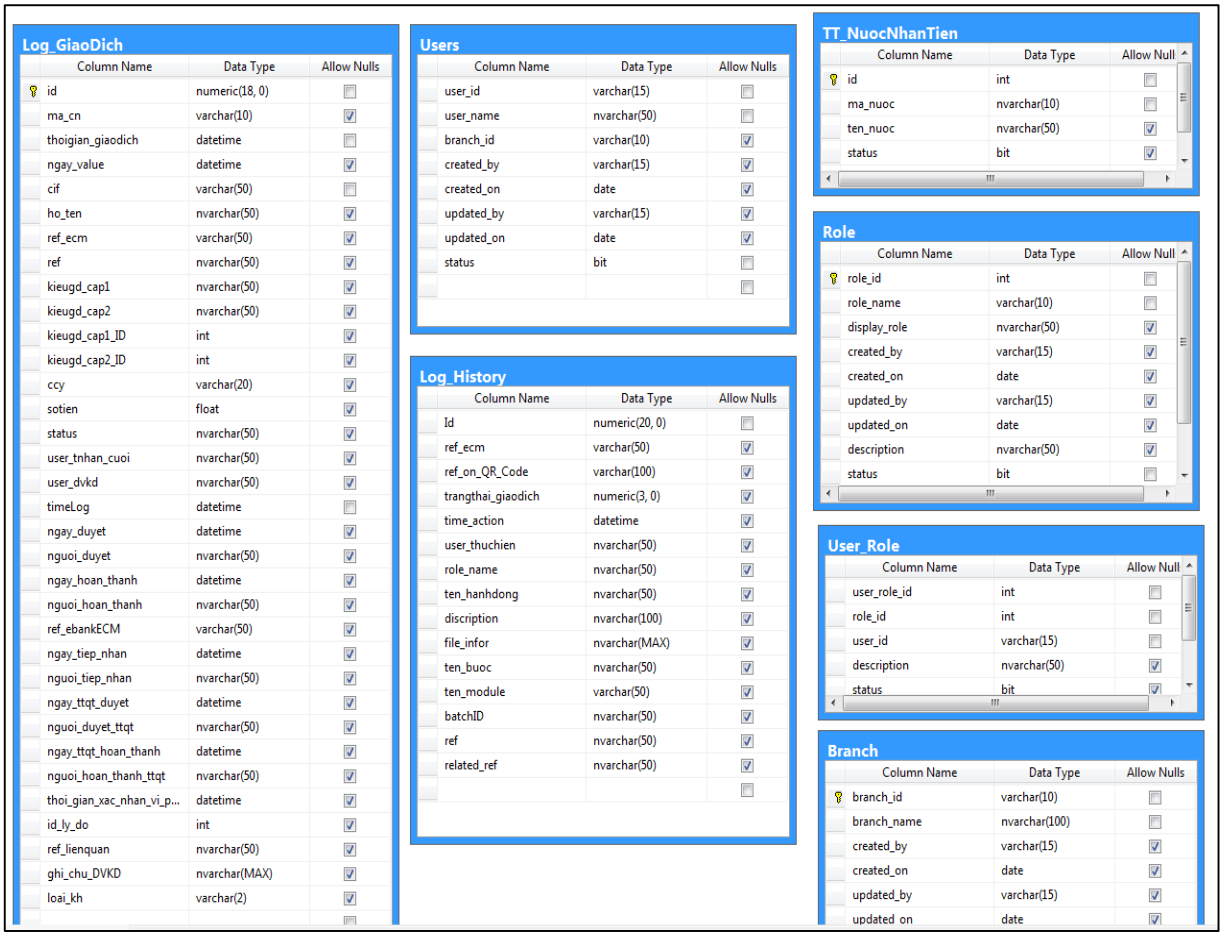


Hình 3. 3. Các bảng dữ liệu chính của hệ thống Ebank được sử dụng để tích hợp

- Bảng GiaoDich: dùng để lưu trữ thông tin giao dịch mà khách hàng thực hiện nhập trên giao diện Ebank
- Bảng LoaiGiaoDich: dùng để lưu trữ thông tin loại giao dịch
- Bảng TaiLieuYeuCau: dùng để lưu trữ thông tin (tên) các loại tài liệu, hồ sơ giấy tờ mà khách hàng cần kê khai cho ngân hàng khi thực hiện giao dịch.

- Đối với hệ thống ECM

Hình 3.4 thể hiện các bảng chính được sử dụng trong quá trình tích hợp

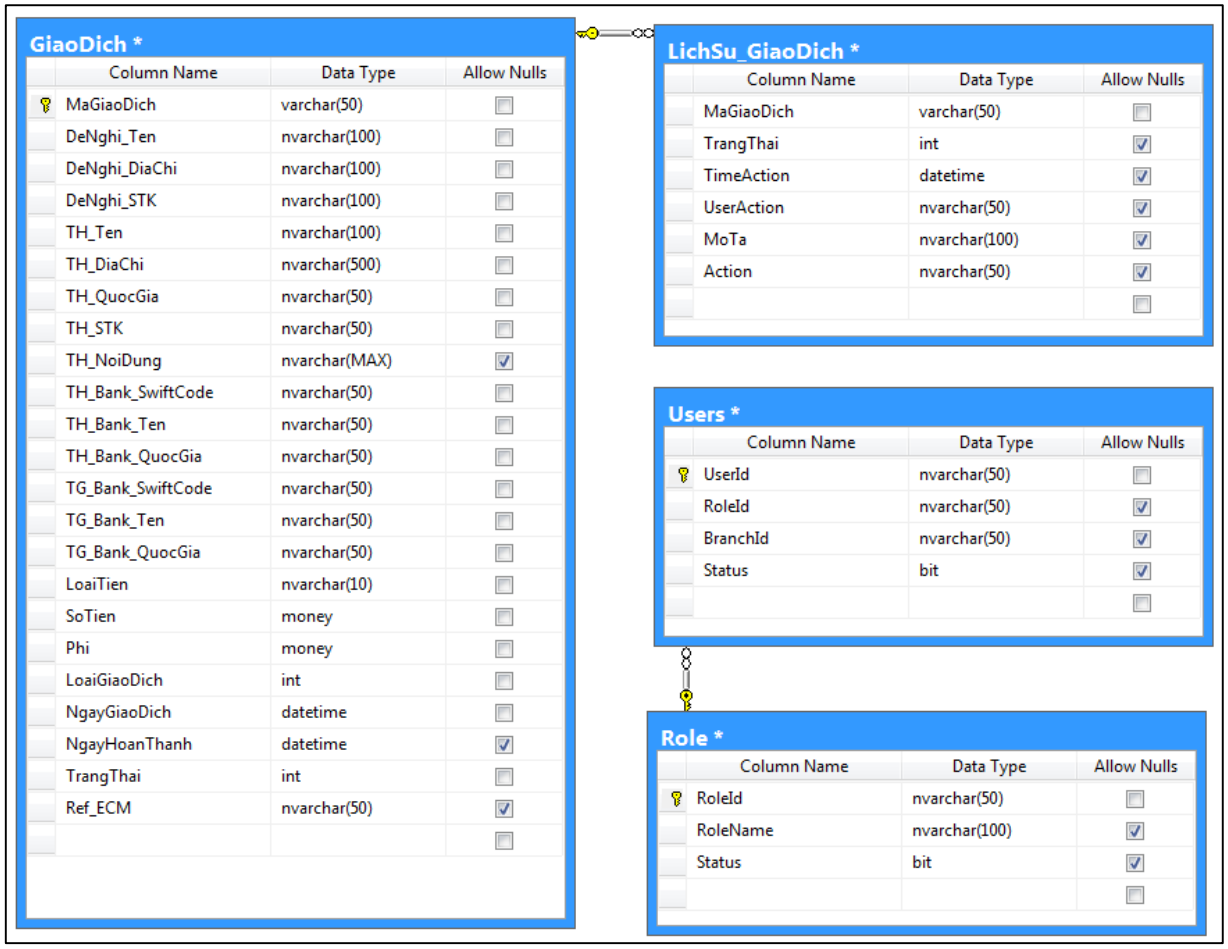


Hình 3. 4. Các bảng dữ liệu chính của hệ thống ECM được sử dụng để tích hợp

- Bảng Log_Giaodich: lưu trữ thông tin giao dịch trên hệ thống ECM phục vụ việc báo cáo.
- Bảng Log_History: lưu trữ thông tin lịch sử chuyển bước của giao dịch, người thực hiện phê duyệt và thời gian thao tác.
- Bảng TT_NuocNhanTien: lưu trữ thông tin quốc gia nhận tiền.
- Bảng User: lưu trữ thông tin người sử dụng (ở đây là tài khoản của nhân viên ngân hàng).
- Bảng Role: lưu trữ quyền của người dùng đối với hệ thống ECM
- Bảng User_Role: lưu trữ thông tin phân quyền của người dùng.
- Bảng Branch: lưu trữ thông tin của tất cả chi nhánh (mã số chi nhánh, tên chi nhánh) của toàn ngân hàng.

- Đối với hệ thống Core FCC

Hình 3.5 thể hiện các bảng chính được sử dụng trong quá trình tích hợp



Hình 3. 5. Các bảng dữ liệu chính của hệ thống CoreFCC được sử dụng để tích hợp

- Bảng GiaoDich: dùng để lưu trữ thông tin giao dịch trên hệ thống Core
- Bảng LichSu_GiaoDich: lưu trữ thông tin lịch sử chuyển bước của giao dịch, người thực hiện phê duyệt và thời gian thao tác.
- Bảng Users: lưu trữ thông tin người sử dụng (ở đây là tài khoản của nhân viên ngân hàng) và quyền hạn của người dùng.
- Bảng Role: lưu trữ quyền của người dùng đối với hệ thống CoreFCC

3.4. Xây dựng các bộ chuyển đổi

- Đối với hệ thống EBank:

  Hệ thống Ebank tích hợp với ESB sử dụng Restful thông qua HTTP Webservice: Ebank thực hiện nhận kết quả phê duyệt hợp lệ/ không hợp lệ của giao dịch từ hệ thống ECM và nhận thông báo hoàn tất phê duyệt chuyển tiền trên hệ thống Core FCC để thông báo tới khách hàng.

API được trục ESB cung cấp trong cho chức năng này là:

URI	Phương thức	Giá trị truyền vào	Ghi chú
/esb/ttqt/status	GET	“idGiaoDich”	Lấy trạng thái của giao dịch sau khi hoàn tất việc chuyển tiền để thông báo tới khách hàng.
/esb/ttqt/docinfo	GET	“idGiaoDich”	Lấy thông tin trạng thái giao dịch sau khi phê duyệt hợp lệ trên ECM


```
{
  "data": [{
    "id": "8773713b-1ecf-40f2-aba8-8ef615ee2baf",
    "MaYC": "TTQT0001706060001",
    "attributes": {
      "status": "GD10",
      "description": "HOAN_THANH",
      "created": "2017-10-22T14:56:29.000Z",
      "updated": "2017-10-26T17:56:28.000Z",
      "userTTQT": "ThaoNTP1",
      "MaCN": "000",
      "CustomerCIF": "00438551",
      "Currency": "USD",
      "Amount": "50"
    }
  }]
}
```


Hình 3. 6. Ví dụ dữ liệu trả về của API: /esb/ttqt/staus


```
{
  "data": [{
    "id": "8773713b-1ecf-40f2-aba8-8ef615ee2baf",
    "MaYC": "TTQT0001706060001",
    "attributes": {
      "status": "GD03",
      "description": "CHO_XU_LY_TTQT",
      "created": "2017-10-22T14:56:29.000Z",
      "updated": "2017-10-24T16:56:28.000Z"
    }
  }]
}
```


Hình 3. 7. Ví dụ dữ liệu trả về của API /esb/ttqt/docinfo

- Đối với hệ thống ECM:

Hệ thống ECM tích hợp với ESB sử dụng Restful thông qua HTTP Webservice: ECM thực hiện tạo giao dịch khi có lệnh từ hệ thống EBank yêu cầu, cập nhật

thông tin giao dịch sang trạng thái tương ứng khi chuyên viên TTQT phê duyệt trên Core FCC. Thực hiện lưu file điện chuyển tiền từ hệ thống Core FCC đẩy sang.

API được trục ESB cung cấp trong cho chức năng này là:

URI	Phương thức	Giá trị truyền vào	Ghi chú
/esb/ttqt/create	POST	“soTien”, “soCIF”, “loaiTien”, “loaiGD1”, “loaiGD2”, “listTaiLieu”, “listChungTu”, “noiDung”	Tạo giao dịch lên hệ thống ECM (từ Ebank → ECM)
/esb/ttqt/process	PUT	“idGiaoDich”, “trangThai”	Cập nhật trạng thái của giao dịch trên ECM tương ứng với quá trình phê duyệt trên Core
/esb/ttqt/addfile	PUT	“idGiaoDich”, “listfile”	Cập nhật các file chứng từ đính kèm lên hệ thống ECM


```
{
  "data": [
    {
      "id": "8773713b-1ecf-40f2-aba8-8ef615ee2baf",
      "MaYC": "TTQT0001706060001",
      "result": "SUCCESS",
      "attributes": {
        "status": "GD03",
        "description": "CHO_XU_LY_TTQT",
        "created": "2017-10-20T09:56:29.000Z",
        "updated": "2017-10-20T09:56:29.000Z",
        "userThucHien": "ThaoNTP1",
        "fileId": [
          {
            "file01": "63ae4b04-e6db-4a51-a830-0d8d9a9c29d0",
            "file02": "986ad700-298e-4834-a271-f173dc3f9aa6"
          }
        ]
      }
    }
  ]
}
```


Hình 3. 8 Ví dụ dữ liệu trả về của API: /esb/ttqt/process

```
{
  "data": [
    {
      "id": "8773713b-1ecf-40f2-aba8-8ef615ee2baf",
      "MaYC": "TTQT0001706060001",
      "result": "SUCCESS",
      "attributes": {
        "status": "GD03",
        "description": "CHO_XU_LY_TTQT",
        "created": "2017-10-20T09:56:29.000Z",
        "updated": "2017-10-20T09:56:29.000Z",
        "userTTQT": "",
        "MaCN": "000",
        "CustomerCIF": "00438551",
        "Currency": "USD",
        "Amount": "50",
        "fileId": [
          {
            "file01": "63ae4b04-e6db-4a51-a830-0d8d9a9c29d0",
            "file02": "986ad700-298e-4834-a271-f173dc3f9aa6"
          }
        ]
      }
    }
  ]
}
```


Hình 3. 9. Ví dụ dữ liệu trả về của API: /esb/ttqt/create

- Đối với hệ thống Core FCC:

  Core FCC tích hợp với ESB sử dụng Restful thông qua HTTP Webservice: hệ thống core thực hiện tạo giao dịch với các thông tin có sẵn sau khi việc tạo giao dịch trên ECM hoàn tất và nhận thông tin file điện chuyển tiền swift từ hệ thống Core Swift. Tại đây, chuyên viên TTQT sẽ thực hiện kiểm tra và phê duyệt các cấp các giao dịch chuyển tiền nước ngoài.

  API được trục ESB cung cấp trong cho chức năng này là:

URI	Phương thức	Giá trị truyền vào	Ghi chú
/esb/ttqt/action	POST	“soTien”, “soCIF”, “loaiTien”, “loaiGD1”, “loaiGD2”, “listTaiLieu”, “listChungTu”, “noiDung”	Tạo giao dịch trên hệ thống Core FCC sau khi phê duyệt hợp lệ trên ECM (ECM → Core)
/esb/ttqt/getfile	GET	“idGiaoDich”	Nhận thông tin file điện chuyển tiền swift

```
{
  "data": [
    {
      "id": "8773713b-1ecf-40f2-aba8-8ef615ee2baf",
      "MaYC": "TTQT0001706060001",
      "result": "SUCCESS",
      "attributes": {
        "status": "GD03",
        "description": "CHO_XU_LY_TTQT",
        "created": "2017-10-21T09:56:29.000Z",
        "updated": "2017-10-21T09:58:29.000Z",
        "userTTQT": "",
        "MaCN": "000",
        "CustomerCIF": "00438551",
        "TenKH" : "Nguyen Van A",
        "Currency": "USD",
        "Amount": "50",
        "ref" : "OFT000135805",
        "Swift" "",
        "QuocGia": "Afghanistan",
        "NoiDung": "Chuyen tien tieu dung",
        "fileId": [
          {
            "file01": "63ae4b04-e6db-4a51-a830-0d8d9a9c29d0",
            "file02": "986ad700-298e-4834-a271-f173dc3f9aa6"
          }
        ]
      }
    }
  ]
}
```


Hình 3. 10. Ví dụ dữ liệu trả về của API: /esb/ttqt/action

- Đối với hệ thống Core SWIFT:

  Tích hợp với ESB sử dụng Restful thông qua HTTP Webservice: thực hiện tạo file điện chuyển tiền sau khi có phê duyệt hợp lệ giao dịch trên hệ thống Core FCC.

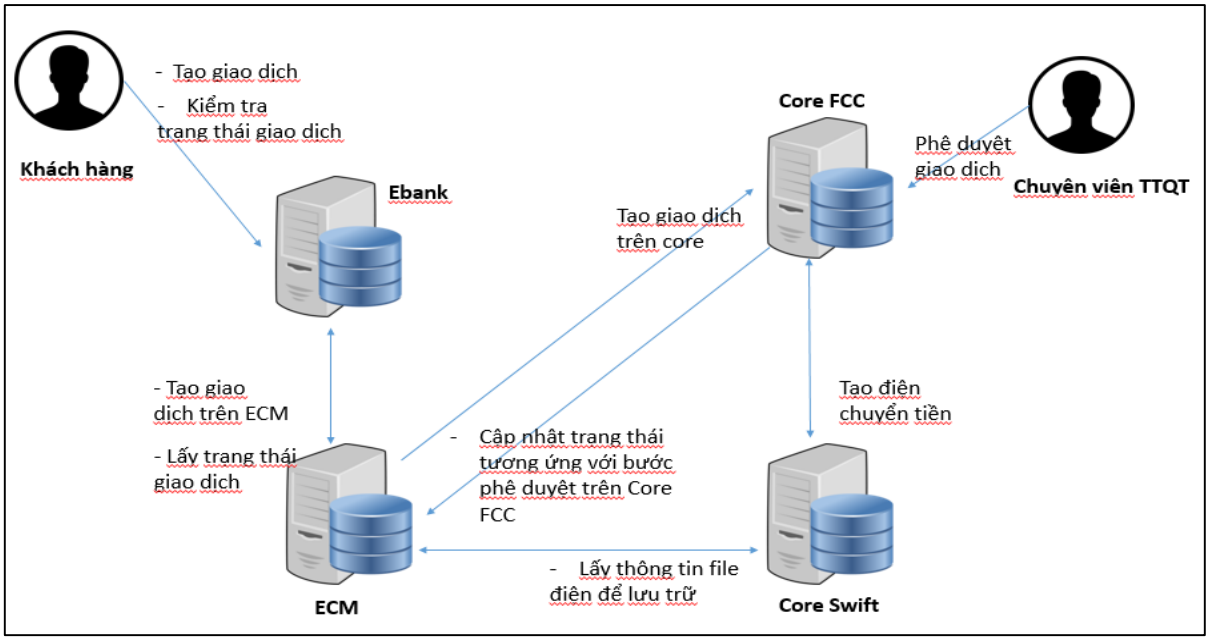
  API được trục ESB cung cấp trong cho chức năng này là:

URI	Phương thức	Giá trị truyền vào	Ghi chú
/esb/ttqt/getswift	POST	“soTien”, “soCIF”, “loaiTien”, “loaiGD1”, “loaiGD2”, “listTaiLieu”, “listChungTu”, “noiDung”	Tạo file điện chuyển tiền sau khi phê duyệt trên hệ thống Core FCC

```
{
  "data": [
    {
      "id": "8773713b-1ecf-40f2-aba8-8ef615ee2baf",
      "MaYC": "TTQT0001706060001",
      "result": "SUCCESS",
      "attributes": {
        "status": "GD09",
        "description": "DA_CHUYEN_TIEN",
        "created": "2017-10-21T09:56:29.000Z",
        "updated": "2017-10-21T09:58:29.000Z",
        "userTTQT": "TraNTH",
        "CustomerCIF": "00438551",
        "Currency": "USD",
        "Amount": "50",
        "ref" : "OFT000135805",
        "QuocGia": "Afghanistan",
        "Swift" : "371ee98e-834b-421b-b693-7d6e8753aef0"
      }
    }
  ]
}
```


Hình 3. 11. Ví dụ dữ liệu trả về của API: /esb/ttqt/getswift

Hình dưới mô tả quá trình tương tác giữa các hệ thống



Hình 3. 12. Quá trình tương tác giữa các hệ thống

3.5. Thiết kế giao diện người dùng

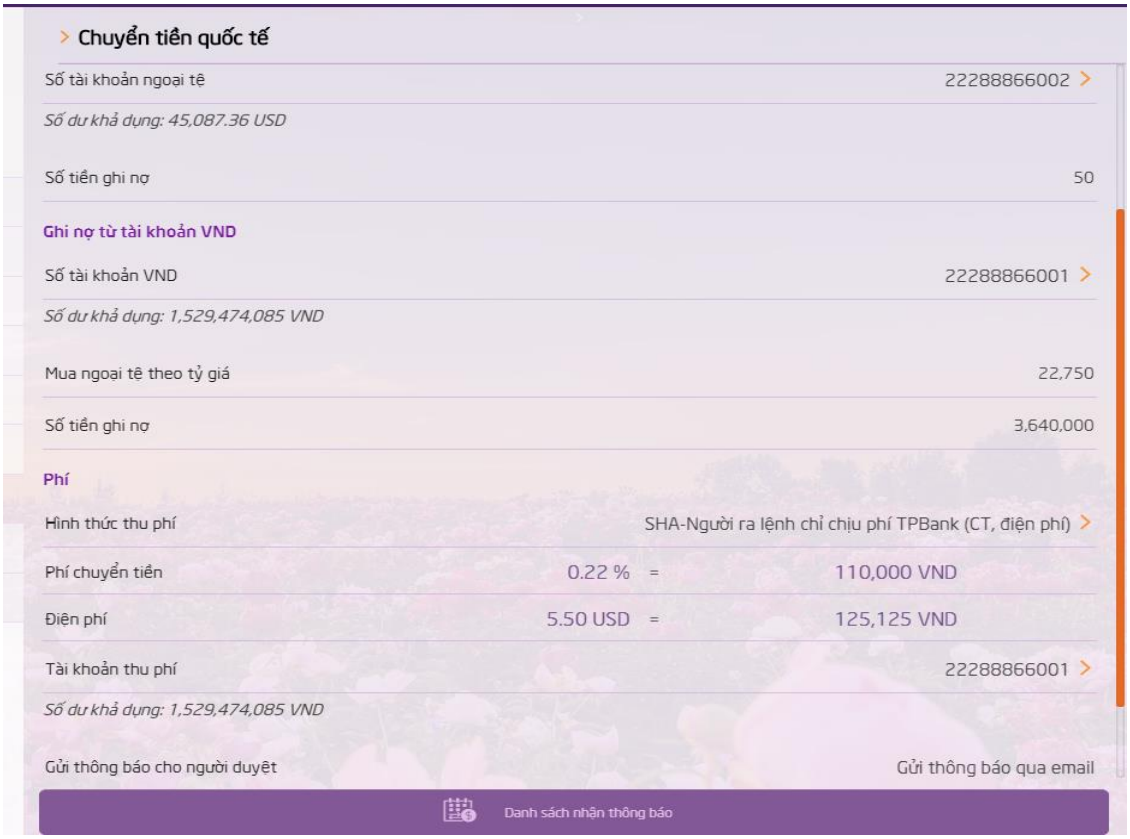
- Hệ thống Ebank: đối với hệ thống này, giao diện cần phải được hiển thị tốt trên môi trường trình duyệt web cũng như môi trường trên các thiết bị di động, do đó sẽ sử dụng HTML5, Javascript và CSS3 để xây dựng.

- Hệ thống ECM: đối với hệ thống này, giao diện chỉ cần đáp ứng nhu cầu hiển thị trên môi trường trình duyệt web, ngoài ra để phù hợp với các chức năng hệ thống thì giao diện hệ thống sẽ được sử dụng Dojo framework, html và css. Giao diện phục vụ nội bộ ngân hàng nên cần đơn giản, dễ thao tác và hạn chế chứa các mã script thao tác phức tạp, tốn tài nguyên của trình duyệt.
- Hệ thống CoreFCC và hệ thống Core Swift: sử dụng JavaForm mặc định của hệ thống.

3.6. Kết quả thử nghiệm

Sau khi áp dụng trục tích hợp ESB, chuyên viên TTQT sẽ giảm thiểu thời gian nhập liệu cũng như việc phê duyệt lệnh chuyển tiền . Kịch bản thử nghiệm giao dịch chuyển tiền quốc tế trên hệ thống Ebank như sau:

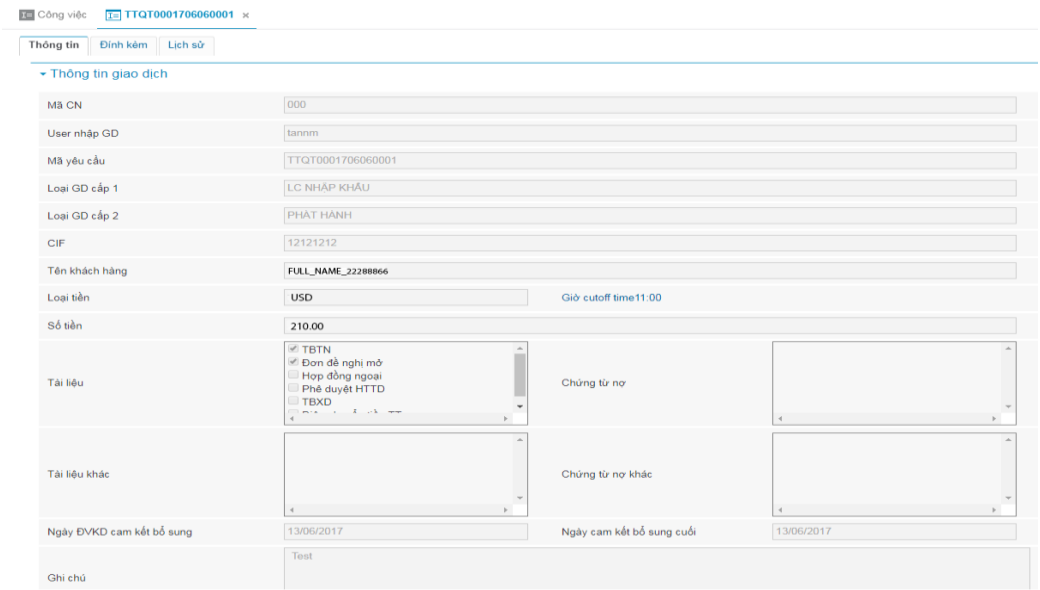
- Khách hàng Nguyễn Văn A đại diện cho công ty TNHH SimpleVN thực hiện chuyển 50 USD sang ngân hàng New Kabul ở Afghanistan. Khách hàng đăng nhập vào hệ thống Ebank dành cho doanh nghiệp, thực hiện nhập thông tin giao dịch.



Hình 3. 13. Thông tin giao dịch trên EBank

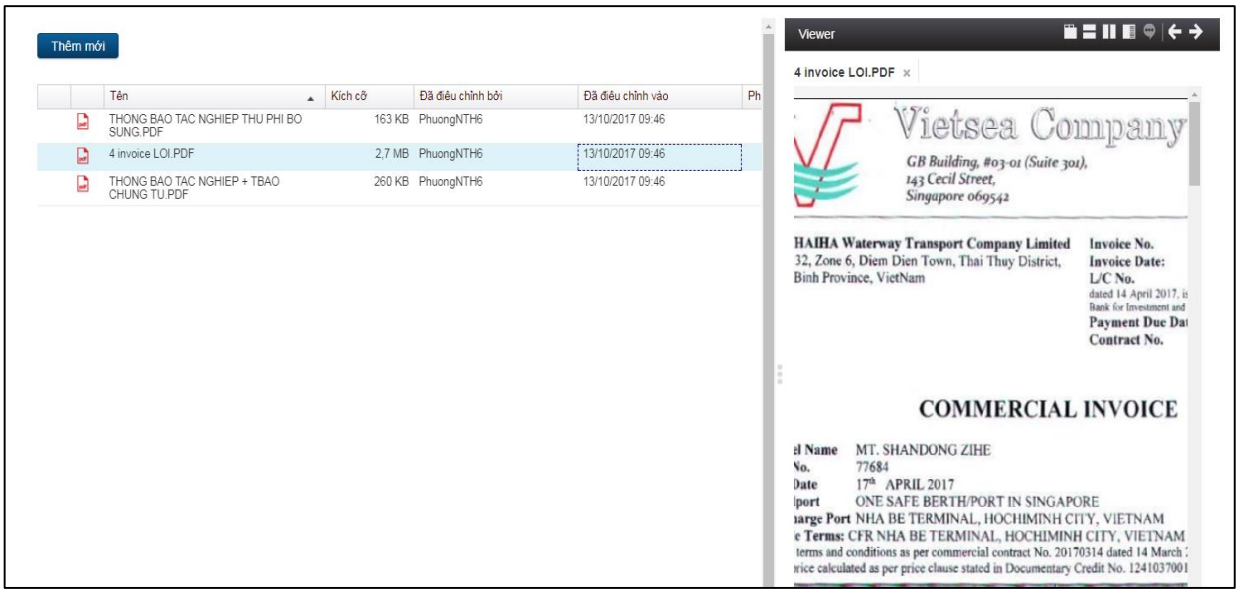
Hình 3. 14. Thông tin các giấy tờ đính kèm

- Sau khi hoàn tất việc nhập thông tin giao dịch, upload các chứng từ cần thiết lên hệ thống, khách hàng sẽ thực hiện yêu cầu lệnh chuyển tiền. Lúc này một giao dịch chứa thông tin tương ứng sẽ được tạo ra trên hệ thống ECM thông qua trục ESB, các file chứng từ đi kèm sẽ được đẩy lên hệ thống ECM để phục vụ việc lưu trữ, đồng thời ECM sẽ thực hiện tạo giao dịch tương ứng trên hệ thống Core FCC, thông tin các giấy tờ lưu trữ trên ECM sẽ được hiển thị tại hệ thống Core FCC.

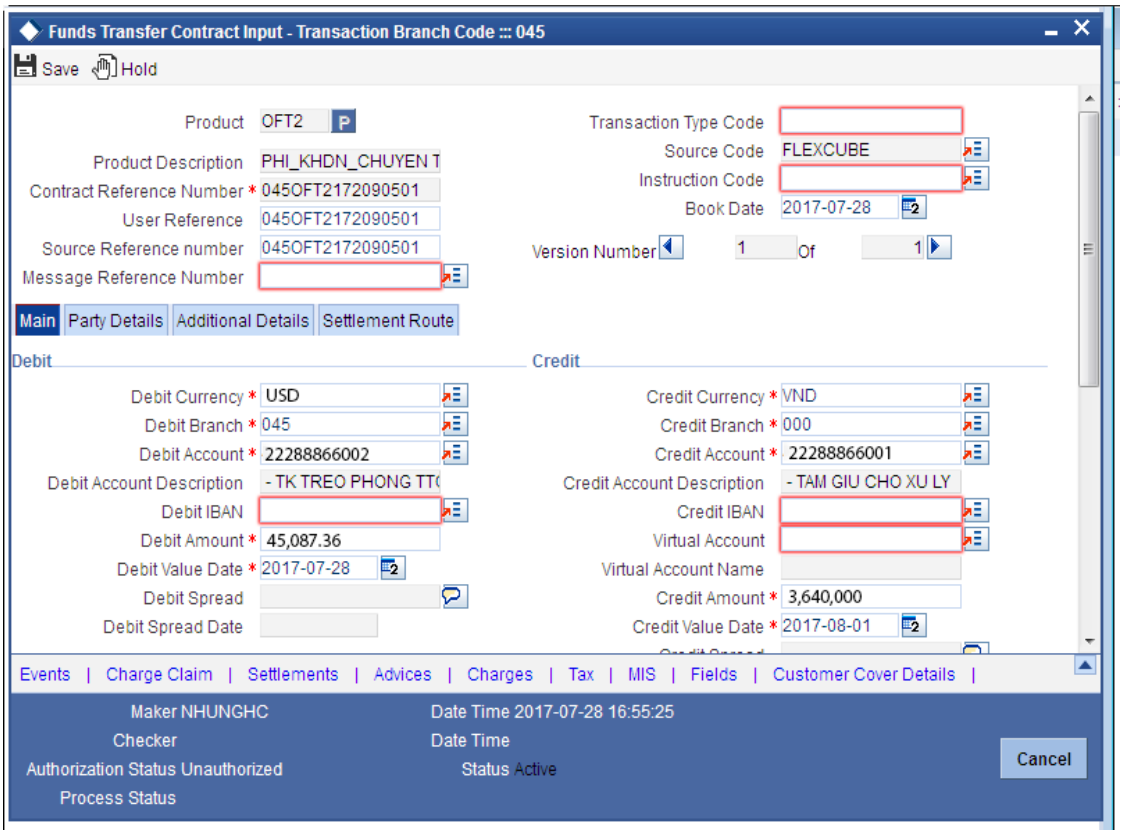


Hình 3. 15. Thông tin giao dịch tương ứng trên hệ thống lưu trữ ECM

- Lúc này hệ thống ECM và hệ thống Core FCC sẽ trao đổi thông tin với nhau thông qua Id của giao dịch. Chuyên viên TTQT vào hệ thống Core FCC, kiểm tra giao dịch và các giấy tờ đính kèm và thực hiện phê duyệt các cấp tại đây.



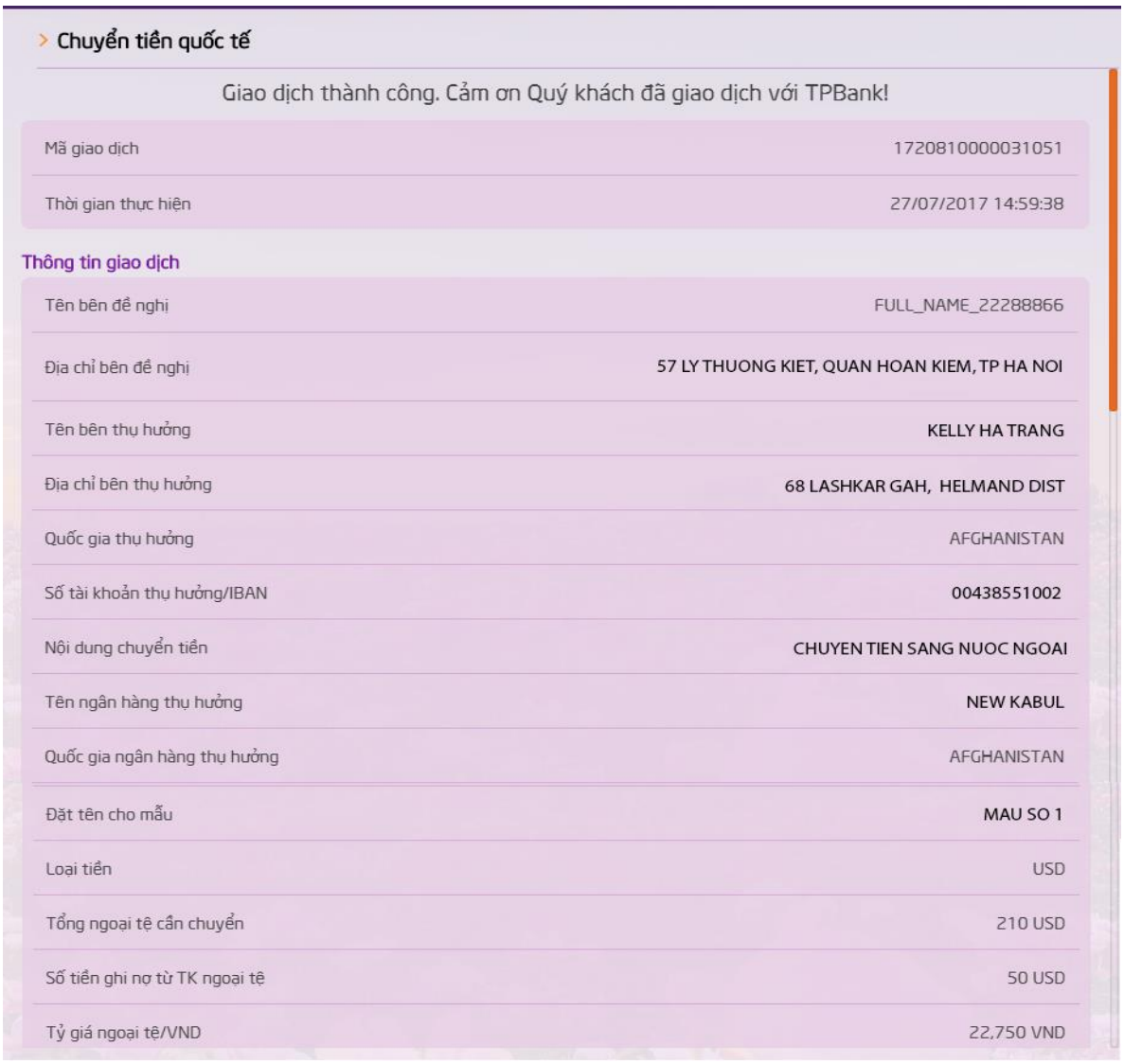
Hình 3. 16. Màn hình danh sách hồ sơ trên Core FCC



Hình 3. 17. Thông tin giao dịch trên hệ thống Core FCC

- Sau khi chuyên viên TTQT thực hiện phê duyệt các cấp trên hệ thống Core FCC, nếu giao dịch hợp lệ, hệ thống Core Swift sẽ sinh ra một file điện swift chuyển tiền. File điện SWIFT này cùng với số REF giao dịch được tạo ra từ Core FCC sẽ là cơ sở để đánh điện chuyển tiền sang ngân hàng nước ngoài.

- Kết thúc việc chuyển điện, Core FCC sẽ thông báo tới hệ thống ECM và EBank để thực hiện lưu kho trên ECM và thông báo hoàn tất chuyển tiền cho khách hàng trên Ebank.



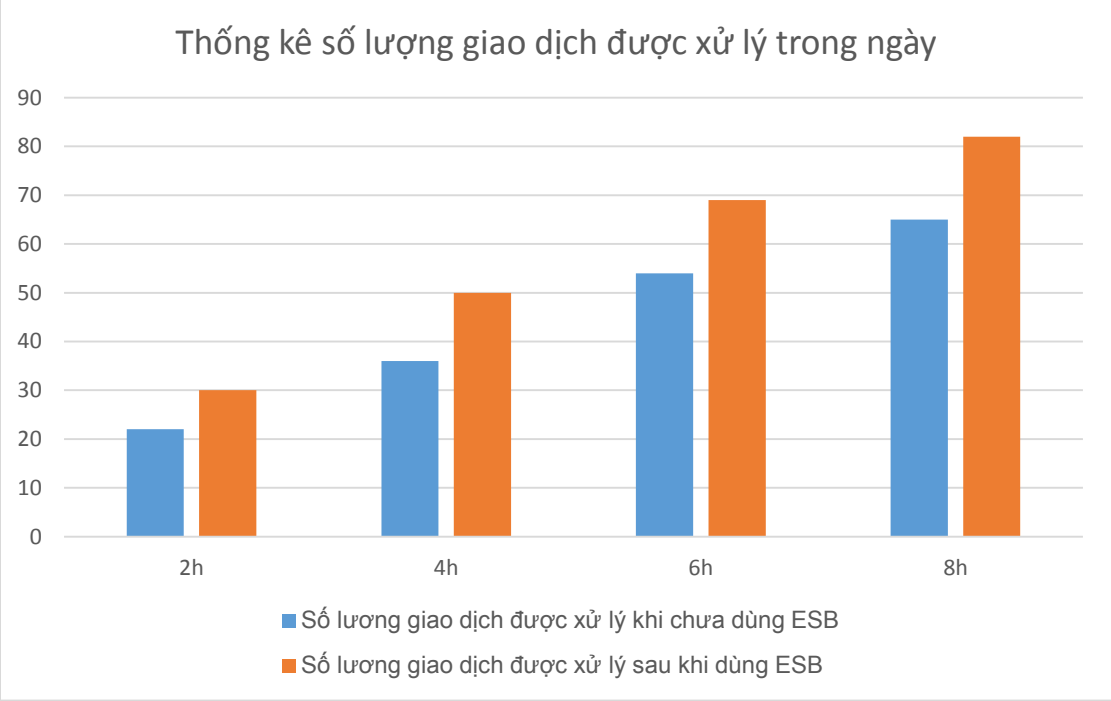
Hình 3. 18. Thông tin giao dịch hoàn tất trên hệ thống Ebank

3.7. Đánh giá kết quả

3.7.1. Kết quả đạt được

Khối lượng công việc và thời gian nhập liệu, kiểm tra giao dịch được giảm thiểu một cách đáng kể. Thay vì phải truy cập vào các hệ thống khác nhau để thực hiện thao tác phê duyệt thì hiện nay người dùng chỉ cần thực hiện phê duyệt tại hệ thống Core FCC, các thông tin khác sẽ được tự động cập nhật trên các hệ thống còn lại. Thời gian giao dịch được giảm xuống, số lương giao dịch và độ chính xác trong các thao tác kiểm tra được tăng lên.

Bảng dưới đây thống kê khảo sát số lượng giao dịch được xử lý sau mỗi 2 giờ làm việc của 5 chuyên viên phòng TTQT. Ta có thể thấy được số lượng giao dịch được xử lý sau sử dụng trục tích hợp ESB được tăng lên khoảng 20% so với hệ thống ban đầu.



3.7.2. Hiệu năng hệ thống

Hệ thống hoạt động tương đối ổn định, quá trình kết nối từ các hệ thống Ebank, ECM tới Core được đảm bảo và thông suốt. Khi thực hiện triển khai pilot hệ thống, với số lượng giao dịch khoảng 400 giao dịch/ngày thì hệ thống vẫn đáp ứng được các thao tác. Dữ liệu được cập nhật tương đối nhanh với độ trễ chưa quá 2 giây. Tuy nhiên khi tăng số lượng giao dịch lên gần 1000 giao dịch (tương

đương với giao dịch chuyển tiền quốc tế của khoảng 20 chi nhánh trong ngày) cùng một thời điểm thì hệ thống đôi lúc phản hồi chưa được nhanh (khoảng hơn 4 giây), đôi lúc bị mất kết nối tới hệ thống core do đó cần tối ưu kết nối tới hệ thống này.

4. Kết chương

Chương 3 luận văn đã trình bày bài toán tích hợp các hệ thống tại ngân hàng TPBank và đề xuất giải pháp tích hợp sử dụng trục tích hợp ESB của Mule ESB để giải quyết bài toán. Bài toán đã được triển khai plilot trên hệ thống UAT (User Acceptance Testing) để giải quyết những vấn đề trong công tác chuyển tiền quốc tế của ngân hàng với những hệ thống tham gia là: hệ thống ngân hàng điện tử EBank, hệ thống lưu trữ chứng từ sổ sách ECM, hệ thống ngân hàng lõi Core FCC và hệ thống tao điện chuyển tiền Core SWIFT.

KẾT LUẬN CHUNG

1. Các kết quả đạt được

Qua quá trình nghiên cứu, tìm hiểu và xây dựng ứng dụng sử dụng ESB để hỗ trợ việc thanh toán quốc tế của ngân hàng, cụ thể là giao dịch chuyển tiền quốc tế, tôi đã bổ sung cho mình thêm nhiều kiến thức cũng như các kỹ năng về tích hợp hệ thống. Mục tiêu mà khóa luận đề ra cơ bản được hoàn thành với các kết quả chính sau:

- Giới thiệu tổng quan về tích hợp hệ thống, các khái niệm cơ bản về lĩnh vực tích hợp hệ thống, đưa ra được lý do tại sao cần phải tích hợp hệ thống, những điểm mạnh và thách thức của việc tích hợp hệ thống cùng hướng tiếp cận vấn đề này. Bên cạnh đó, chương này cũng đã trình bày về các kiến trúc của tích hợp hệ thống cùng một số phương pháp tích hợp phổ biến đang được sử dụng rộng rãi trên toàn cầu.

- Đặc tả chi tiết kỹ thuật tích hợp dịch vụ sử dụng trục dịch vụ tổng thể ESB. Các khái niệm về ESB, kiến trúc cũng như các tính năng cơ bản mà ESB cung cấp cho người phát triển đã được trình bày chi tiết trong luận văn. Đồng thời luận văn cũng đánh giá ưu nhược điểm giữa một số phương pháp tích hợp, bên cạnh đó giới thiệu một số công cụ ESB Middleware phổ biến hiện nay.

- Phân tích và giải quyết bài toán xây dựng ứng dụng hỗ trợ phòng TTQT trong công tác phê duyệt giao dịch chuyển tiền quốc tế; từ đó đề xuất giải pháp sử dụng trục tích hợp ESB của Mule ESB để giải quyết bài toán. Dựa trên giải pháp đó, chúng tôi đã tiến hành triển khai plilot trên hệ thống UAT (User Acceptance Testing) và đã thu được kết quả đánh giá tích cực từ phía người dùng, làm cơ sở để có những định hướng nâng cấp các chức năng trong tương lai.

2. Định hướng phát triển trong tương lai

Hiện tại hệ thống đang hoàn tất quá trình UAT và thực hiện xin phê duyệt để có thể triển khai trên môi trường thực ngiệm Production.
Do thời gian nghiên cứu cũng như kinh nghiệm về tích hợp dữ liệu còn hạn chế, cho nên ứng dụng còn gặp nhiều những điểm bất cập như là: việc chuyển đổi dữ liệu mới

chỉ đơn thuần, cơ chế ghi log còn sơ sài và hiệu năng chưa thực sự tốt lắm, do đó trong tương lai ứng dụng cần cải tiến các mặt sau:

- Nâng cấp việc chuyển đổi dữ liệu sang một số kiểu định dạng khác, thuận tiện cho việc xử lý.
- Đặt thêm cơ chế ghi log trong luồng dữ liệu ESB để kịp thời phát hiện những bất thường xảy ra.
- Thực hiện nâng cấp hiệu năng hệ thống để đáp ứng được số lượng giao dịch ngày càng tăng.

Ngoài ra, tiếp tục nghiên cứu và sử dụng giải pháp trục dịch vụ ESB của Mule ESB để định hướng tích hợp các hệ thống nghiệp vụ khác tại ngân hàng TPBank như:

- Hệ thống thông tin khách hàng: cung cấp thông tin cơ bản của khách hàng như họ tên, số tài khoản, ngày sinh, loại khách hàng là doanh nghiệp hay cá nhân để phục vụ một số yêu cầu của từng hệ thống Core Banking
- Hệ thống báo cáo: Lưu trữ, tổng hợp các báo cáo của các tổ chức tín dụng theo các mẫu và các tiêu chí khác nhau.
- Một số hệ thống liên quan đến chuyển tiền quốc tế hoặc phê duyệt tín dụng.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] PGS.TS. Nguyễn Ngọc Hóa, Bài giảng tích hợp hệ thống.

Tiếng Anh

[2] Carl Jones., 2011. "*Do more with SOA Integration: Best of Packt*", 1st edition, Packt Publishing Ltd, UK, 319-408

[3] Falko Menge, *Enterprise Service Bus*, FREE AND OPEN SOURCE SOFTWARE CONFERENCE 2007

[4] Matt Lucas, *ESB Usage Scenarios and Patterns*, WebSphere Message Broker Architecture and Strategy

[5] T. Sulaeman and Albarda, "Design architecture enterprise service bus to support multi-tenant client and resource provider," 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE), Yogyakarta, 2016, pp. 1-5.

[6] P. Vrba, M. Fuksa and M. Klíma, "JADE-JBossESB gateway: Integration of multi-agent system with enterprise service bus," 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, CA, 2014, pp. 3663-3668.

Internet.

[7] https://vi.wikipedia.org/wiki/Middleware

[8] https://www.infoq.com/articles/ESB-Integration

[9] http://www.1vs.vn/tintuc/15194_tich-hop-he-thong-(phan-1)-cac-muc-do-va-mo-hinh-tich-hop.html

[10] http://www.1vs.vn/tintuc/15195_tich-hop-he-thong-(phan-2)-giai-phap-ky-thuat-tich-hop-cua-1c.html