

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



**NGUYỄN VIỆT THANH**

**TÍCH HỢP HỆ THỐNG THÔNG TIN Y TẾ  
THEO MÔ HÌNH RESTFUL**

**Ngành: Công nghệ thông tin chất lượng cao**

**HÀ NỘI - 2016**

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**NGUYỄN VIỆT THANH**

**TÍCH HỢP HỆ THỐNG THÔNG TIN Y TẾ  
THEO MÔ HÌNH RESTFUL**

**Ngành: Công nghệ thông tin chất lượng cao**

**Cán bộ hướng dẫn: PGS.TS. Nguyễn Ngọc Hóa**

**HÀ NỘI - 2016**

**VIETNAM NATIONAL UNIVERSITY, HANOI  
UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

**NGUYEN VIET THANH**

**INTEGRATED MEDICAL INFORMATION  
SYSTEM USING RESTFUL MODEL**

**Major: Information Technology**

**Supervisor: Assoc.Prof. Nguyen Ngoc Hoa**

**HÀ NỘI - 2016**

## **LỜI CAM ĐOAN**

Tôi xin cam đoan khóa luận tốt nghiệp do tôi tự mình thực hiện dưới sự hướng dẫn của Thầy Nguyễn Ngọc Hóa, mọi thông tin tham khảo sử dụng trong khóa luận đều được trích dẫn đầy đủ và hợp pháp.

Tôi xin hoàn toàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định của nhà trường cho lời cam đoan của mình.

Hà Nội, ngày      tháng 5 năm 2016

Người cam đoan

Nguyễn Việt Thanh

## LỜI CẢM ƠN

Tôi xin chân thành cảm ơn PGS.TS Nguyễn Ngọc Hóa là giảng viên của Trường Đại học Công Nghệ đã tận tình giúp đỡ tôi về kiến thức, định hướng phát triển và cả về tinh thần cố gắng trong suốt quá trình làm khóa luận tốt nghiệp.

Tôi cũng xin gửi lời cảm ơn đến các thầy cô của khoa Công Nghệ Thông Tin vì đã giảng dạy và hướng dẫn tôi trong suốt 4 năm theo học tại Trường Đại học Công Nghệ.

Cảm ơn những người bạn đã cùng tôi vượt qua quãng thời gian sinh viên đầy những kỉ niệm vui buồn.

Cuối cùng, tôi xin gửi lời biết ơn sâu sắc đến với gia đình vì đã luôn ở bên cạnh ủng hộ tôi trên con đường học tập và nghiên cứu đầy khó khăn.

*Xin chân thành cảm ơn*

Hà Nội, Tháng 5 Năm 2016

Nguyễn Việt Thanh

## TÓM TẮT NỘI DUNG

**Tóm tắt:** . Với thực trạng môi trường và xã hội hiện nay, con người càng phải tự chăm lo nhiều hơn đến sức khỏe và từ đó dẫn đến việc khám, chữa bệnh ngày càng nhiều. Để nâng cao hiệu quả quản trị và điều hành trong các cơ sở y tế, công nghệ thông tin đã được ứng dụng. Thực trạng đó cũng dẫn đến việc có rất nhiều các hệ thống thông tin được xây dựng và triển khai theo những mô hình khác nhau trong ngành Y tế. Tuy nhiên với cách hoạt động hiện tại ở hầu hết các cơ sở y tế, người bệnh vẫn phải trải qua một quy trình khám bệnh rất vất vả, quá trình thăm khám của các bác sĩ phải trải qua nhiều bước chuẩn đoán, xét nghiệm tại nhiều phòng chuyên khoa khác nhau, thời gian để người bệnh chờ đợi và lấy kết quả từ các phòng này là khá dài và mệt mỏi, nhất là với những trường hợp khó chuẩn đoán phải trải qua nhiều xét nghiệm.

Chính vì thế, trong khoá luận tốt nghiệp này, chúng tôi tập chung tìm hiểu và ứng dụng những phương pháp tích hợp hệ thống tiên tiến trong việc hỗ trợ liên thông giữa các ứng dụng khác nhau tại cơ sở Y tế.

Nội dung khóa luận sẽ tập trung trình bày một số phương pháp tích hợp hệ thống, chú trọng đến kỹ thuật tích hợp theo mô hình RESTFUL. Từ đó, ứng dụng để giải quyết bài toán tự động hóa các quá trình lấy kết quả thăm khám của người bệnh dựa vào các phân hệ quản lý thông tin y tế.

**Từ khóa:** Tích hợp, hệ thống thông tin y tế, RESTFUL.

## ABSTRACT

**Abstract:** With the reality of today's society and the environment, the more people have to take care of their health. To improve the efficiency of management and administration in the health facilities, information technology has been applied. That situation also leads to a lot of the information system to be built and deployed according to the different models in the medical industry. However with the current activity in most of the medical establishment, the patient must still undergo a consultation process, which included many diagnostic steps, tests in many different specialty, time people have to wait and get the results from these is quite long and tiring.

Thus, in this thesis we will learn and apply these methods of advanced system integrated to combine different applications in the medical facility in to one.

Content of the thesis will focus on the presentation of a number of systems integration methods, focuses on integrating using RESTFUL model. From then on, applications it to solve the problems of automation the process of taking the results of the patient visit based on the medical information management system.

**Keywords:** integration, healthcare information system, RESTFUL

## MỤC LỤC

<b>TÓM TẮT NỘI DUNG .....</b>	<b>3</b>
<b>ABSTRACT.....</b>	<b>4</b>
<b>MỤC LỤC .....</b>	<b>5</b>
<b>DANH MỤC BẢNG BIỂU.....</b>	<b>7</b>
<b>DANH MỤC HÌNH VẼ.....</b>	<b>8</b>
<b>LỜI MỞ ĐẦU.....</b>	<b>10</b>
<b>CHƯƠNG 1.TỔNG QUAN VỀ TÍCH HỢP HỆ THỐNG.....</b>	<b>12</b>
1.1. TỔNG QUAN VỀ TÍCH HỢP HỆ THỐNG. ....	12
1.1.1. Tích hợp hệ thống là gì ?.....	12
1.1.2. Mục tiêu của tích hợp hệ thống .....	12
1.1.3. Thách thức của tích hợp hệ thống .....	12
1.2. MỘT SỐ PHƯƠNG PHÁP TÍCH HỢP .....	13
1.2.1. Hướng tiếp cận .....	13
1.2.2. Kiến trúc đa tầng trong tích hợp hệ thống.....	13
1.2.3. Tích hợp mức dữ liệu .....	18
1.2.4. Tích hợp mức chức năng.....	20
1.2.5. Tích hợp mức dịch vụ.....	25
1.3. KẾT LUẬN.....	28
<b>CHƯƠNG 2.ỨNG DỤNG RESTFUL TRONG TÍCH HỢP HỆ THỐNG THÔNG TIN Y TẾ .....</b>	<b>29</b>
2.1. TÌM HIỂU VỀ RESTFUL.....	29
2.1.1. Giới thiệu REST .....	29
2.1.2. Các nguyên lý cơ bản của REST.....	29
2.1.3. Một số kiến trúc phù hợp với REST.....	31
2.1.4. Đánh giá ưu và nhược điểm của RESTFUL.....	32
2.2. BÀI TOÁN TÍCH HỢP HỆ THỐNG THÔNG TIN Y TẾ .....	33
2.2.1. Bài toán.....	33
2.2.2. Tổng quan về nghiệp vụ .....	33
<b>CHƯƠNG 3.XÂY DỰNG HỆ THỐNG THÔNG TIN Y TẾ TÍCH HỢP VÀ KẾT QUẢ THỬ NGHIỆM .....</b>	<b>36</b>
3.1. TỔNG QUAN VỀ HỆ THỐNG .....	36
3.2. XÁC ĐỊNH YÊU CẦU.....	37



3.2.1. Yêu cầu của hệ thống .....	37
3.2.2. Chức năng của từng thành phần .....	37
3.2.3. Đặc tả yêu cầu .....	37
3.2.4. Xác định yêu cầu cụ thể.....	38
3.2.5. Yêu cầu phi chức năng .....	38
3.3. THIẾT KẾ KIẾN TRÚC TÍCH HỢP.....	39
3.4. THIẾT KẾ CHI TIẾT TÍCH HỢP .....	41
3.4.1. Xác định các Actor và UseCase .....	41
3.4.2. Đặc tả UseCase.....	42
3.4.3. Thiết kế cơ sở dữ liệu .....	48
3.5. THỬ NGHIỆM VÀ TRIỂN KHAI .....	52
3.5.1. Phát triển hệ thống.....	52
3.5.2. Môi trường thử nghiệm .....	57
3.5.3. Thử nghiệm và đánh giá.....	58
<b>CHƯƠNG 4.KẾT LUẬN, ĐỊNH HƯỚNG NGHIÊN CỨU.....</b>	<b>69</b>
4.1. CÁC KẾT QUẢ ĐẠT ĐƯỢC .....	69
4.2. ĐỊNH HƯỚNG PHÁT TRIỂN TRONG TƯƠNG LAI.....	69
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>70</b>

## **DANH MỤC BẢNG BIỂU**

Bảng 2.1. Các phương thức sử dụng trong REST .....	31
Bảng 3.1: Yêu cầu chức năng của hệ thống .....	38
Bảng 3.2. Bảng yêu cầu phi chức năng của hệ thống .....	38
Bảng 3.3. Bảng users .....	49
Bảng 3.4. Bảng roles .....	50
Bảng 3.5. Bảng permissions .....	50
Bảng 3.6. Bảng permission_role .....	51
Bảng 3.7. Bảng role_user .....	51
Bảng 3.8. Cấu trúc thư mục của dự án .....	53

## DANH MỤC HÌNH VẼ

Hình 1.1 Phân hoạch các tầng trong kiến trúc tích hợp .....	14
Hình 1.2. Kiến trúc 1-tier .....	15
Hình 1.3. Kiến trúc 2-tier .....	16
Hình 1.4. Kiến trúc Middleware.....	17
Hình 1.5. Kiến trúc 3-tier .....	17
Hình 1.6. Chia sẻ dữ liệu tệp .....	18
Hình 1.7. Sockets .....	19
Hình 1.8. Cơ sở dữ liệu dùng chung .....	20
Hình 1.9. Gọi thủ tục từ xa.....	21
Hình 1.10. Hàm đồng bộ .....	21
Hình 1.11. Minh họa gọi thủ tục .....	22
Hình 1.12. Dịch vụ thông điệp .....	23
Hình 1.13. Hàng đợi Điểm-Điểm.....	24
Hình 1.14. Hàng đợi Phát hành và Đăng ký.....	24
Hình 1.15. Thông điệp SOAP .....	24
Hình 1.16. Chuẩn BPEL trên Eclipse.....	26
Hình 1.17. Mô hình SOA .....	26
Hình 1.18. Triển khai SOA dựa trên Web và XML.....	27
Hình 2.1. Thiết kế có trạng thái.....	30
Hình 2.2. Thiết kế phi trạng thái .....	30
Hình 2.3. Ví dụ về Electronic Medical Record [5] .....	34
Hình 2.4. Ví dụ về phân dữ liệu hình ảnh trong DICOM [6].....	35
Hình 3.1. Mô hình phân rã chức năng cho toàn hệ thống .....	38
Hình 3.2. Kiến trúc tổng quan hệ thống .....	40
Hình 3.3. Biểu đồ ca sử dụng tổng quát của hệ thống .....	42
Hình 3.4. Ca sử dụng quản lý thông tin người bệnh .....	42
Hình 3.5. Ca sử dụng quản lý hồ sơ y tế .....	44
Hình 3.6. Ca sử dụng Quản lý hình ảnh DICOM.....	46
Hình 3.7. Lược đồ dữ liệu của hệ thống tích hợp.....	49
Hình 3.8. Đoạn mã mô tả quy trình sử dụng OpenMRS REST API .....	54
Hình 3.9. Quy trình sử dụng Orthanc REST API .....	55
Hình 3.10. Những giao diện làm việc của Chikisa vào trang chủ .....	56
Hình 3.11. Những module trình diễn dữ liệu hình ảnh của DICOM .....	56

Hình 3.12. Giao diện đăng nhập hệ thống.....	58
Hình 3.13. Giao diện trang chủ của Chikitsa .....	59
Hình 3.14. Danh sách người bệnh.....	59
Hình 3.15.. Chức năng thêm người bệnh mới.....	60
Hình 3.16. Giao diện thông tin chi tiết người bệnh và Sửa thông tin người bệnh .....	60
Hình 3.17. Giao diện danh sách người bệnh trong cơ sở dữ liệu về hồ sơ y tế .....	61
Hình 3.18. Một bản ghi kết quả khám của người bệnh.....	61
Hình 3.19. Tạo bản ghi thăm khám mới .....	62
Hình 3.20. Thêm kết quả xét nghiệm.....	62
Hình 3.21. Giao diện tìm kiếm người bệnh của OpenMRS mặc định .....	63
Hình 3.22. Giao diện tìm kiếm hồ sơ y tế của OpenMRS mặc định.....	63
Hình 3.23. Chức năng thêm xét nghiệm của OpenMRS mặc định .....	63
Hình 3.24. Giao diện tải lên tệp DICOM .....	64
Hình 3.25. Giao diện chính của dịch vụ quản lý DICOM.....	64
Hình 3.26. Giao diện trình diễn dữ liệu hình ảnh của DICOM.....	65
Hình 3.27. Giao diện trang chủ của Orthanc, liệt kê danh sách DICOM .....	66
Hình 2.28. Trang thông tin về người bệnh .....	66
Hình 2.29. Trang thông tin Study.....	67
Hình 2.30. Trang thông tin Series .....	67
Hình 2.31. Chức năng Xem hình ảnh của Series DICOM .....	68

# LỜI MỞ ĐẦU

## 1. Đặt vấn đề

### 1.1. Định hướng nghiên cứu

Ngày nay, với việc bước vào công nghiệp hóa, hiện đại hóa, đất nước ta cũng đối mặt với một vấn đề nghiêm trọng đó là ô nhiễm môi trường: nước, không khí, đất, biển đều đang ngày một chịu nhiều nguồn ô nhiễm xâm phạm. Bên cạnh đó là rất nhiều nguồn thực phẩm không an toàn, thuốc lá, rượu bia, tai nạn giao thông... tất cả dẫn đến nhu cầu khám chữa bệnh chưa bao giờ giảm sút.

Tuy nhiên quy trình khám chữa bệnh hiện tại ở các cơ sở khám chữa bệnh của nước ta vẫn khiến bệnh nhân cảm thấy khá vất vả: sau khi khám tổng thể để đưa ra chuẩn đoán sơ bộ, người bệnh có thể được yêu cầu làm thêm các xét nghiệm khác để bác sĩ có thể đưa ra chuẩn đoán chính xác nhất. Khi đó việc khám và chờ đợi kết quả tại các phòng xét nghiệm này rất mất thời gian và làm người bệnh mệt mỏi.

Cần có một giải pháp tự động hóa cho quá trình chuyển giao kết quả xét nghiệm giữa các phòng chuyên môn và bác sĩ nhằm giúp việc khám chữa bệnh của người bệnh trở nên bớt khó khăn hơn.

Để giải quyết vấn đề này, chúng ta có thể lựa chọn phương pháp tích hợp một số module quản lý thông tin y tế có sẵn trở thành một hệ thống đồng nhất để có thể sử dụng đồng thời các dịch vụ mà từng module đem lại.

### 1.2. Định hướng công nghệ

Với sự phát triển của công nghệ thông tin, đã có rất nhiều phần mềm mã nguồn mở với mục đích quản lý các thông tin về y tế và sức khỏe. Tuy nhiên khó có phần mềm nào có thể đảm đương được tất cả các yêu cầu, bởi vậy với những nhu cầu nhất định chúng ta có thể lựa chọn những phần mềm phù hợp để sử dụng.

Mỗi phần mềm lại được viết dựa trên một kiến trúc khác nhau và có cách sử dụng khác nhau, vì vậy việc lựa chọn các phần mềm có sự tương đồng để có thể thuận lợi cho việc tích hợp là một việc cần ưu tiên.

Hiện nay các dịch vụ Web đang càng ngày càng phát triển và có mặt ở hầu hết các lĩnh vực, việc triển khai các dịch vụ viết trên nền Web cũng ngày càng trở nên đơn giản. Đặc biệt từ sau khi REST ra đời, nó đã được đông đảo các công ty

dẫn đầu về cung cấp dịch vụ mạng như Yahoo, Google và Facebook ủng hộ, với những ưu điểm của mình, REST đang ngày càng được ưa chuộng sử dụng hơn cho các dịch vụ Web.

Từ những nhận định trên tôi quyết định sử dụng mô hình tích hợp dựa trên RESTFUL và lựa chọn các phần mềm quản lý thông tin y tế được viết trên nền Web.

## **2. Mục tiêu của khóa luận**

Khoá luận tốt nghiệp này có mục tiêu nghiên cứu tìm hiểu một số phương pháp tích hợp hệ thống và ứng dụng trong việc tích hợp một số hệ thống thông tin cơ bản trong lĩnh vực Y tế (do việc có thể đưa ra một hệ thống hoàn chỉnh cho cơ sở y tế để sử dụng là quá lớn so với quy mô của khóa luận tốt nghiệp).

Mục tiêu trên sẽ được cụ thể hoá thông qua những nội dung thực hiện chính sau:

- Khảo sát, đánh giá một số cách tiếp cận phục vụ tích hợp hệ thống, chú trọng đến phương pháp tích hợp mức dịch vụ theo mô hình dịch vụ Web.
- Áp dụng mô hình dịch vụ Web RESTFUL trong việc xây dựng giải pháp tích hợp một số hệ thống thông tin cơ bản trong ngành Y tế.
- Phát triển hệ thống tích hợp thông tin y tế thử nghiệm theo mô hình giải pháp đã đưa ra ở trên.

## **3. Tổ chức khóa luận**

Khóa luận được thực hiện xuyên suốt trong quá trình từ khi hình thành các khái niệm, ý tưởng, phân tích thiết kế, trình bày cài đặt sản phẩm cho đến khi hoàn thành sản phẩm và kiểm tra kiểm thử đánh giá sản phẩm. Các kết quả chính của khóa luận sẽ trình bày trong 4 chương có nội dung vắn tắt như sau:

- **Chương 1:** Tổng quan về tích hợp hệ thống - các khái niệm cơ bản về tích hợp hệ thống, quy trình chung tích hợp và một số phương pháp tích hợp.
- **Chương 2:** Ứng dụng RESTFUL trong tích hợp hệ thống thông tin y tế.
- **Chương 3:** Xây dựng hệ thống thông tin y tế và kết quả thử nghiệm.
- **Chương 4:** Kết luận, định hướng nghiên cứu.

## **CHƯƠNG 1. TỔNG QUAN VỀ TÍCH HỢP HỆ THỐNG**

### **1.1. Tổng quan về tích hợp hệ thống.**

#### **1.1.1. Tích hợp hệ thống là gì ?**

Trong thời đại ngày nay thông tin ngày càng được quan tâm hơn, cần phải dễ dàng truy xuất, có độ sẵn sàng cao. Các tổ chức thường có sẵn các ứng dụng nghiệp vụ, sử dụng nhiều kiến trúc, công nghệ khác nhau, chưa được định hướng để phối hợp cùng nhau tạo ra một hệ thống thông tin cụ thể. Hơn nữa theo thời gian các ứng dụng mới được tạo ra với những công nghệ và kiến trúc hiện đại hơn, tuy nhiên những ứng dụng mới cần phải phối hợp tốt với nhau và với những ứng dụng cũ. Từ đó người ta đưa ra những phương pháp, kỹ thuật, công nghệ và mẫu (pattern) để có thể liên kết và giúp các ứng dụng này phối hợp hoạt động cùng nhau.

Theo định nghĩa từ Wikipedia: *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 hệ thống thống nhất.*

#### **1.1.2. Mục tiêu của tích hợp hệ thống**

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

#### **1.1.3. Thách thức của tích hợp hệ thống**

Khi một ứng dụng mới ra đời nó thường không được hoạch định trước kế hoạch tích hợp, bởi vậy thiết kế của nó thường khó có thể dễ dàng kết hợp với những thành phần đã có hoặc những thành phần mới khác. Điều này bắt nguồn từ thực tế các tổ chức chưa quan tâm đến vấn đề tích hợp một cách nghiêm túc, họ thường chỉ tập chung tạo ra sản phẩm mới để giải quyết ngay lập tức vấn đề đang tồn tại.

Bên cạnh đó các ứng dụng đôi khi được viết trên những nền tảng khác nhau như Ứng dụng Web, ứng dụng cho hệ điều hành Windows, Linux...; hay những ngôn ngữ khác nhau: C++, Java, Python...; cũng như phương thức quản lý dữ liệu khác nhau: Tập lưu trữ, Dữ liệu quan hệ, Dữ liệu không cấu trúc – có cấu trúc. Việc vượt qua những khác biệt này để tích hợp chúng rất khó khăn.

Cuối cùng, với những khó khăn trên, cần có kiến thức tổng thể lớn về hệ thống cùng với kinh phí rất cao để có thể thực hiện tốt việc tích hợp.

## 1.2. Một số phương pháp tích hợp

### 1.2.1. Hướng tiếp cận

Từ dưới lên

- Đánh giá hiện trạng và phân loại các hệ thống đã có
- Phân tích những vấn đề cụ thể phát sinh do thiếu sự tích hợp giữa các hệ thống
- Giải quyết những vấn đề đó thông qua những dự án tích hợp không có điều phối, không cần xây dựng kiến trúc tích hợp tổng thể

Từ trên xuống

- Đánh giá hiện trạng và phân loại các hệ thống đã có
- Xây dựng kiến trúc tích hợp tổng thể sớm nhất có thể, đảm bảo cả những khía cạnh về quy trình nghiệp vụ lẫn công nghệ.

Hiện nay người ta thường sử dụng cách tiếp cận Từ trên xuống kết hợp thêm Từ dưới lên.

### 1.2.2. Kiến trúc đa tầng trong tích hợp hệ thống

Kiến trúc đa tầng bao gồm:

**Client :** người dùng hoặc chương trình mong muốn thực hiện tác vụ thông qua hệ thống.

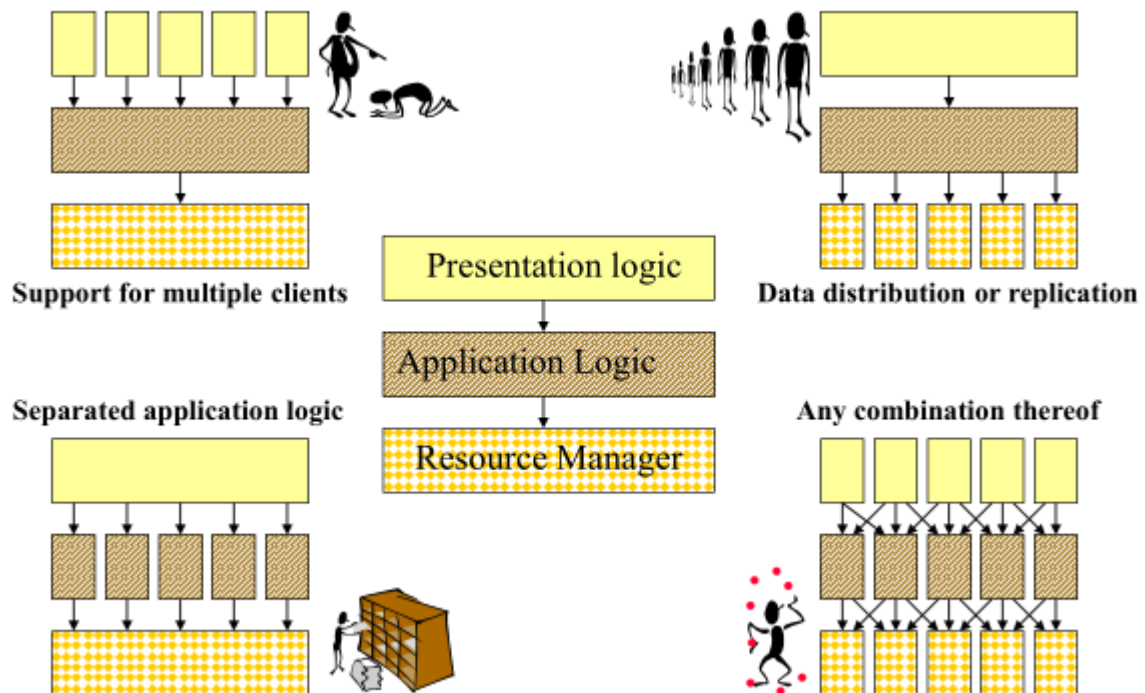
**Presentation layer:** tầng giúp client gửi yêu cầu và nhận kết quả phản hồi.

**Application logic:** tầng đảm bảo thực hiện các quy trình nghiệp vụ đồng thời xác lập những thao tác nào có thể được thực hiện bởi client.

**Resource manager:** tầng tương tác với tài nguyên dữ liệu ở mức thấp. Có thể là một Hệ quản trị cơ sở dữ liệu hoặc hệ thống khác có khả năng lưu trữ dữ liệu và thực hiện truy vấn.



## Phân hoạch tại các tầng

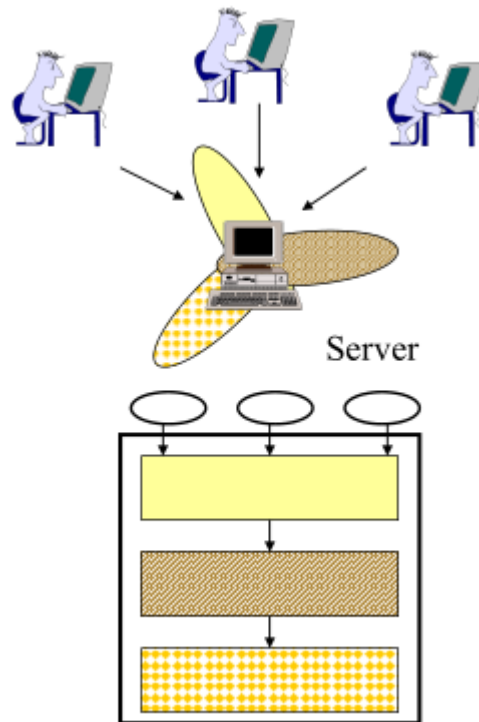


**Hình 1.1 Phân hoạch các tầng trong kiến trúc tích hợp**

Trong mô hình trên, mỗi box thể hiện cho một phần của hệ thống, mỗi arrow thể hiện một kết nối giữa hai phần của hệ thống. Càng nhiều box thì hệ thống càng có nhiều module đồng nghĩa với tăng khả năng làm việc song song và tăng tính đóng gói, tái sử dụng. Tuy nhiên, càng nhiều module cũng có nghĩa hệ thống càng phải quản lý, điều phối nhiều hơn, cũng như có nhiều bước trung gian để hoàn thành công việc, dẫn đến hiệu suất giảm.

### **Kiến trúc 1-tier – Tập trung toàn bộ**

Với kiến trúc 1-tier tất cả các tầng Presentation layer, Application logic, Resource Manager được xây dựng trong cùng một thực thể đồng nhất.

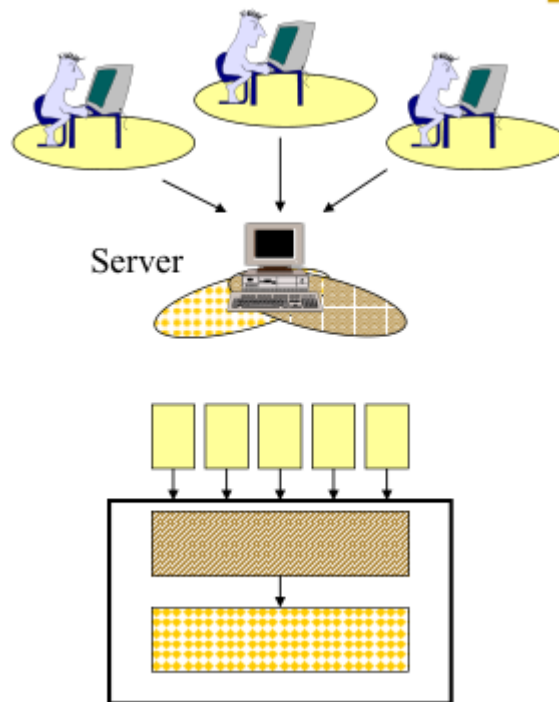


**Hình 1.2. Kiến trúc 1-tier**

Mọi yêu cầu từ client sẽ được thông qua một Presentation duy nhất. Kiến trúc này thường được xây dựng trên các máy trạm có năng lực tính toán lớn, giúp việc quản lý tập trung tài nguyên tốt hơn.

### **Kiến trúc 2-tier**

Tầng Presentation được chuyển về phía client, mỗi client sẽ làm việc với một presentation khác nhau với những hành vi khác nhau, giảm bớt gánh nặng cho Máy chủ. Đưa đến khái niệm API (Application Program Interface), giao diện tương tác với hệ thống từ bên ngoài, cho phép tích hợp một hệ thống phức tạp bằng việc liên kết nhiều hệ thống khác.

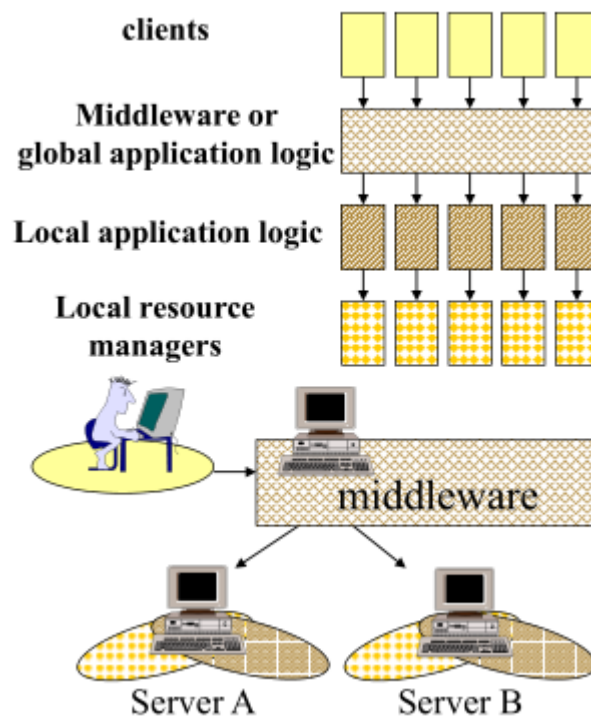


**Hình 1.3. Kiến trúc 2-tier**

Kiến trúc 2-tier duy trì việc xử lý ở trong máy chủ, tầng Resource Manager chỉ quản lý duy nhất một Application logic duy nhất giúp thiết kế chặt chẽ và tối ưu. Tuy nhiên, client bị phụ thuộc vào Presentation layer, nếu client muốn kết nối đến 2 hệ thống khác nhau, phải có 2 presentation để phục vụ, và client cũng là nơi chịu trách nhiệm xử lý nghiệp vụ riêng bởi mỗi hệ thống về cơ bản không có liên hệ gì với nhau. Resource Manager và Application logic gắn chặt với nhau cũng gây ra khó khăn cho việc cải tiến hoặc thay đổi.

### **Middleware**

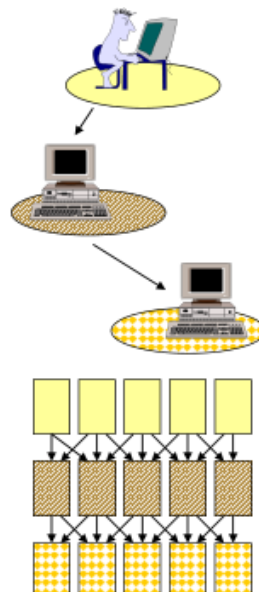
Để giải quyết những vấn đề của kiến trúc 2-tier, ta đưa vào Middleware, là một lớp trung gian giữa client và các tầng khác trong hệ thống. Middleware hoạt động như tầng business logic để cung cấp nghiệp vụ của tất cả các hệ thống. Khi đó Middleware giúp đơn giản hóa thiết kế client, đóng vai trò làm nền tảng cho các chức năng và logic mức cao. Middleware cũng giúp tối ưu phân bổ tài nguyên, truy cập và thu nhận kết quả phản hồi.



**Hình 1.4. Kiến trúc Middleware**

### Kiến trúc 3-tier

Trong kiến trúc 3-tier, ba tầng Presentation layer, Application logic, Resource Manager được tách riêng. Kiến trúc 3-tier có cùng ưu nhược điểm Middleware



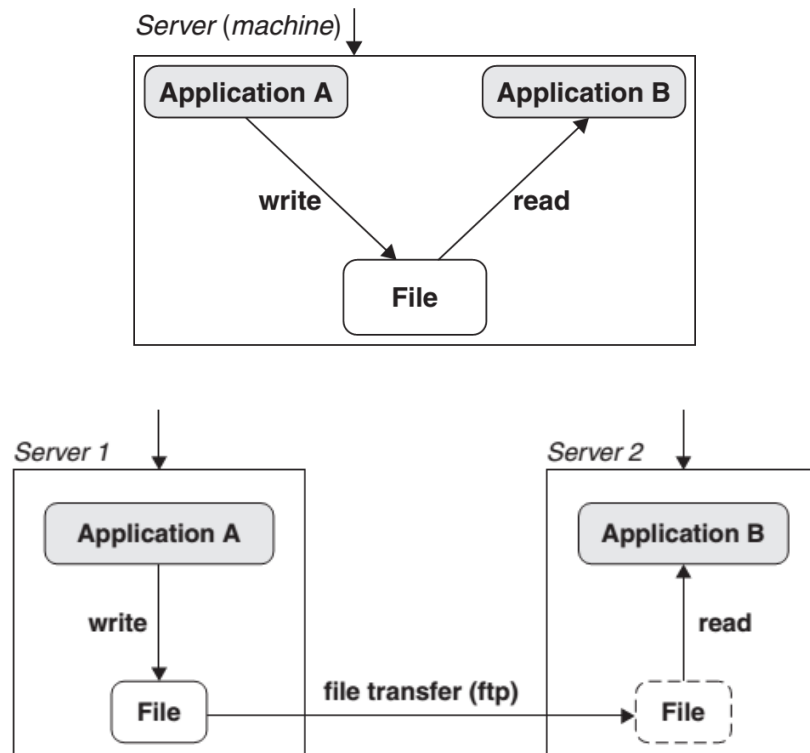
**Hình 1.5. Kiến trúc 3-tier**

### 1.2.3. Tích hợp mức dữ liệu

Đây là kiểu tích hợp ở mức thấp, các ứng dụng/hệ thống tham gia vào hệ tích hợp chia sẻ dữ liệu chung với nhau.

#### 1.2.3.1. Chia sẻ dữ liệu tệp – File-base data sharing

Các ứng dụng/hệ thống dùng chung một tệp để lưu trữ dữ liệu, tệp dữ liệu được luân chuyển giữa các máy chủ qua giao thức ftp.



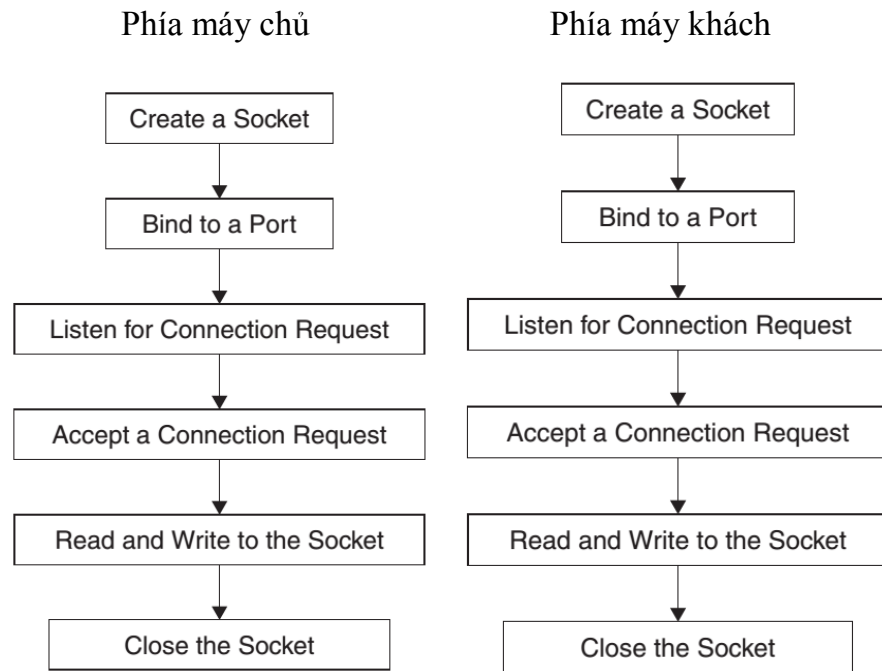
**Hình 1.6. Chia sẻ dữ liệu tệp**

Ưu điểm: Đơn giản, dễ xây dựng.

Nhược điểm:

- Không thể vừa ghi/đọc, có thể gây ra dữ liệu không nhất quán
- Không tin cậy khi cần thao tác nhiều tệp: cần định nghĩa định dạng tệp, quy tắc đặt tên, cách truy xuất tệp; khi triển khai trên nhiều máy tính cần phải xác định ứng dụng/dịch vụ nào chịu trách nhiệm truyền/nhận tệp; toàn bộ nội dung tệp được chia sẻ trong hệ tích hợp [2].

#### 1.2.3.2. Sockets



**Hình 1.7. Sockets**

Sử dụng kết nối trực tiếp để chia sẻ dữ liệu

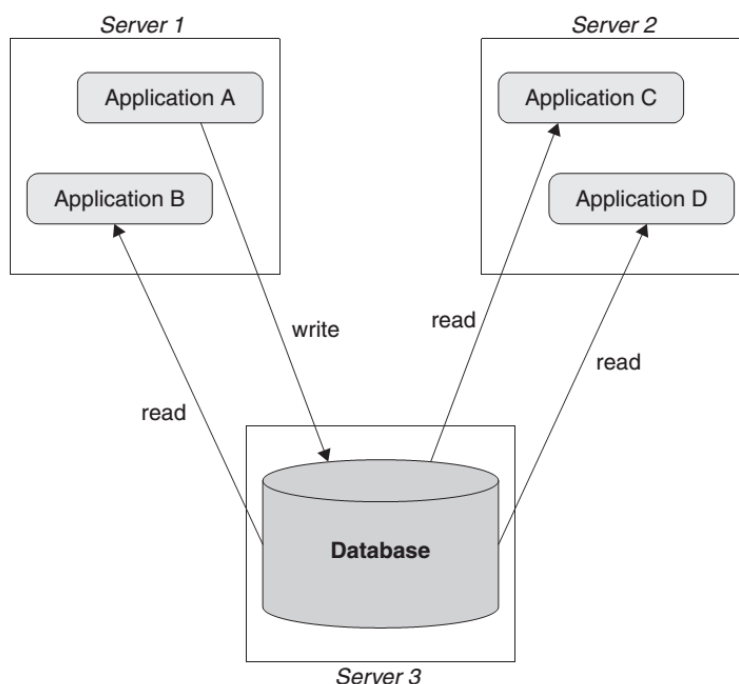
Ưu điểm

- Không phải chia sẻ toàn bộ tệp dữ liệu
- Dữ liệu có thể cập nhật nhanh hơn
- Tạo mô hình kết nối 1-n

Nhược điểm:

- Cần xây dựng hệ thống ở mức thấp
- Cần kết nối chặt chẽ giữa các ứng dụng [2].

### 1.2.3.3. Cơ sở dữ liệu dùng chung – Shared Database



**Hình 1.8. Cơ sở dữ liệu dùng chung**

Là sự kết hợp giữa chia sẻ dữ liệu tệp và sockets. Hệ quản trị CSDL sẽ chịu trách nhiệm đảm bảo sự nhất quán và tập chung của dữ liệu chia sẻ giữa các ứng dụng/hệ thống thành phần.

Nhược:

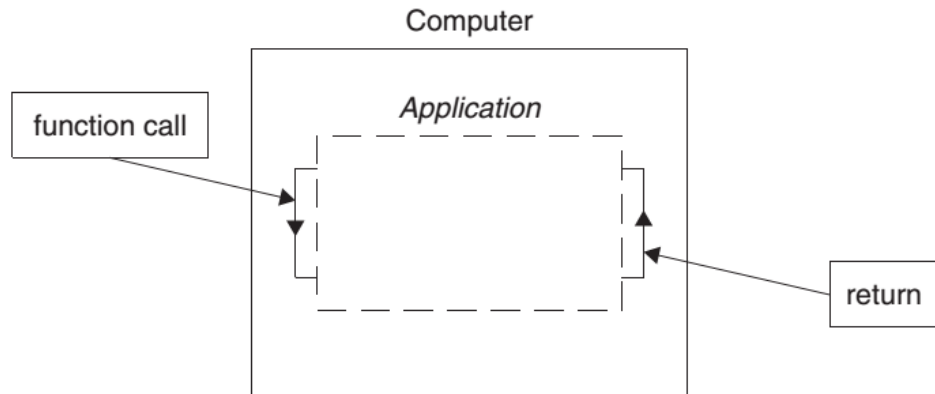
- Không có cơ chế để thông báo cho các ứng dụng/hệ thống thành phần biết khi có thay đổi dữ liệu
- Các ứng dụng/hệ thống sử dụng chung mô hình dữ liệu
- Khi quy mô tích hợp tăng lên hệ quản trị có nguy cơ quá tải [2].

### 1.2.4. Tích hợp mức chức năng

Là phương thức cho phép các ứng dụng chia sẻ các chức năng lẫn nhau (tái sử dụng chức năng).

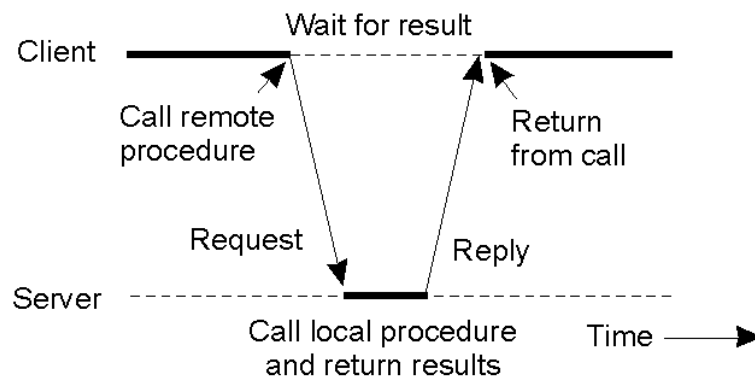
#### 1.2.4.1. Gọi thủ tục từ xa – Remote Procedures Calls-RPC

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 phải lập trình lại trên ứng dụng đó.



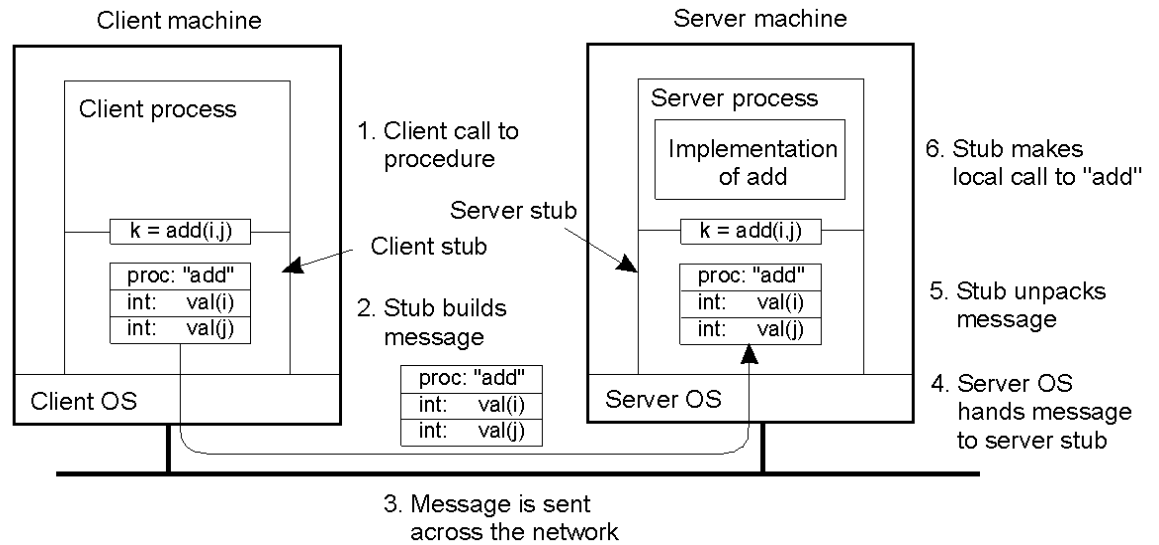
**Hình 1.9. Gọi thủ tục từ xa**

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



**Hình 1.10. Hàm đồng bộ**





**Hình 1.11. Minh họa gọi thủ tục**

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

Ưu điểm:

- Là phương thức đầu tiên cho phép chia sẻ hàm – chức năng giữa các ứng dụng.
- Có thể triển khai với nhiều nền tảng khác nhau

Nhược điểm:

- Cần nắm được đặc tả giao tiếp, các phương thức đóng gói dữ liệu
- Các ứng dụng cần sử dụng chung ngôn ngữ lập trình
- Kết nối chặt chẽ do sử dụng cơ chế hàm đồng bộ
- Trở nên phức tạp nếu có nhiều lời gọi [2].

#### **1.2.4.2. Đối tượng phân tán – Distributed Objects**

RPC có nhược điểm chưa tích hợp được các ứng dụng sử dụng ngôn ngữ lập trình khác nhau và chạy trên nhiều hệ điều hành khác nhau.

Đối tượng phân tán ra đời để giải quyết vấn đề này của RPC.

Đối tượng phân tán có một số mô hình điển hình:

- Common Object Request Broker Architecture (CORBA)

- Microsoft's Distributed Component Object Model (DCOM)
- Java's Remote Method Invocation (RMI)

Ưu điểm:

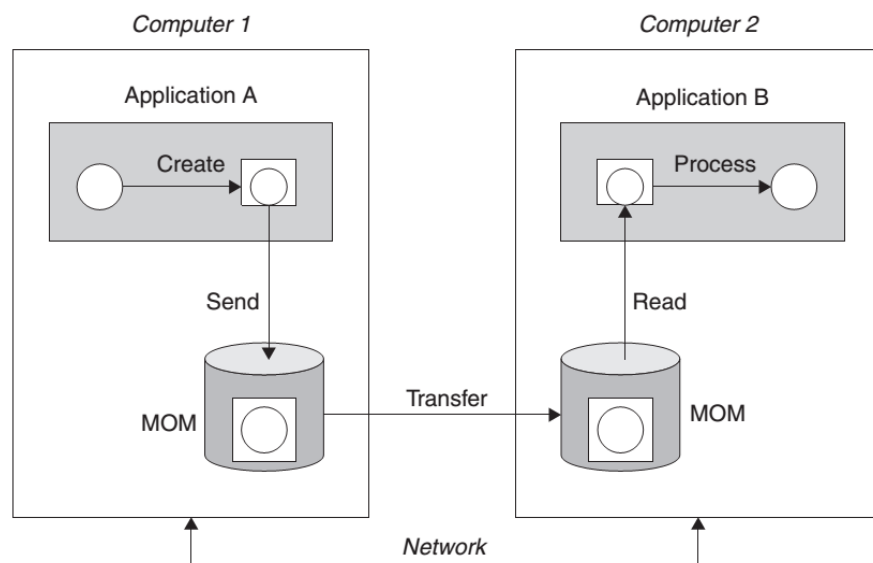
- Khắc phục được những vấn đề của RPC
- Độc lập ngôn ngữ lập trình, độc lập nền tảng môi trường
- Làm mờ máy chủ và máy khách

Nhược điểm:

- Vẫn cần cơ chế hàm đồng bộ
- Phương thức truyền thông không đảm bảo tin cậy: yêu cầu và kết quả có thể không tới được đích mong muốn.

#### 1.2.4.3. Thông điệp – Messaging

Dịch vụ thông điệp - hay Message Oriented Middleware (MOM) – là phương thức cho phép giải quyết các vấn đề của Đối tượng phân tán dựa trên cơ chế gửi thông điệp không đồng bộ - asynchronous message.



**Hình 1.12. Dịch vụ thông điệp**

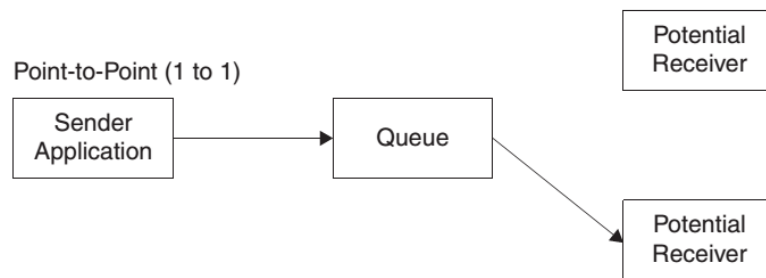
Đặc điểm của Dịch vụ thông điệp

- Các thông điệp được gửi/nhận theo cơ chế không đồng bộ thông qua hàng chờ/queue
- Thông điệp phải được gửi đến đích, nếu không Dịch vụ tiến hành gửi lại

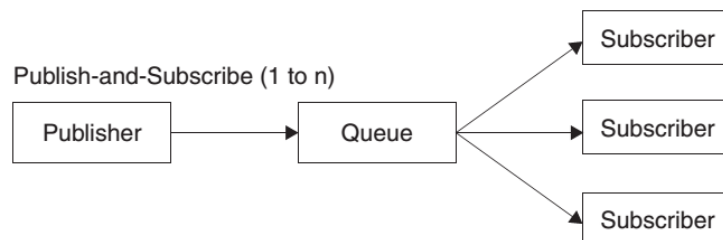
- Dịch vụ có cơ chế điều phối thông điệp để tránh máy chủ quá tải.

Dịch vụ thông điệp gồm có:

- Kênh/Hàng đợi: sử dụng cơ chế Điểm-Điểm và Phát hành-Đăng ký
- Thông điệp: gói dữ liệu cần trao đổi giữa máy khách/máy chủ
- Điểm cuối : điểm cho phép máy khách/máy chủ kết nối tới Dịch vụ thông điệp



**Hình 1.13. Hàng đợi Điểm-Điểm**



**Hình 1.14. Hàng đợi Phát hành và Đăng ký**

Listing 6.1: A SOAP message

```

1  <SOAP-ENV:Envelope xmlns:SOAP-ENV="SOAPEnvelopeURI"
2      SOAP-ENV:encodingStyle="SOAPEncodingURI">
3      <SOAP-ENV:Header>
4      </SOAP-ENV:Header>
5      <SOAP-ENV:Body>
6          <m:GetLastTradePrice xmlns:m="ServiceURI">
7              <tickerSymbol>IBM</tickerSymbol>
8          </m:GetLastPrice>
9      </SOAP-ENV:Body>
10 </SOAP-ENV:Envelope>
  
```

**Hình 1.15. Thông điệp SOAP**

Thông điệp cho phép tích hợp theo cơ chế không đồng bộ, khắc phục được những vấn đề của các phương thức đối tượng phân tán, giúp nâng cao khả năng mở rộng, đảm bảo được độ tin cậy trong giao tiếp giữa các ứng dụng.

Nhược điểm:

- Sử dụng nhiều Dịch vụ thông điệp cùng lúc có thể dẫn tới không đồng nhất, ví dụ như về giao thức: HTTP, HTTPS...
- Có những ứng dụng cần cả cơ chế đồng bộ và không đồng bộ
- Không đồng nhất về định dạng của thông điệp [2].

### **1.2.5. Tích hợp mức dịch vụ**

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

Ta có thể chia làm 2 loại nhỏ

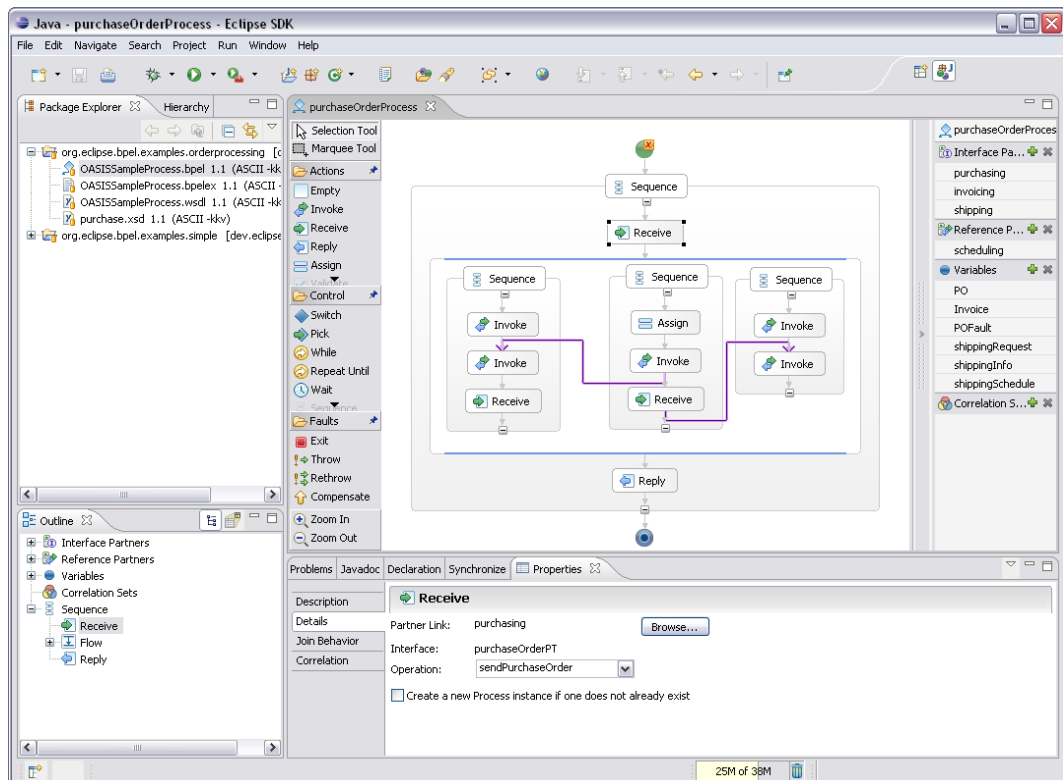
- Tích hợp hệ thống dựa vào tích hợp quy trình nghiệp vụ
- Tích hợp hệ thống dựa vào kiến trúc hướng dịch vụ

#### **1.2.5.1. Tích hợp quy trình**

Tích hợp mức quy trình đảm bảo mục tiêu tạo mô hình nghiệp vụ chung giữa các hệ thống liên kết qua dịch vụ và quy trình. Thường được sử dụng để tích hợp các hệ thống như:

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

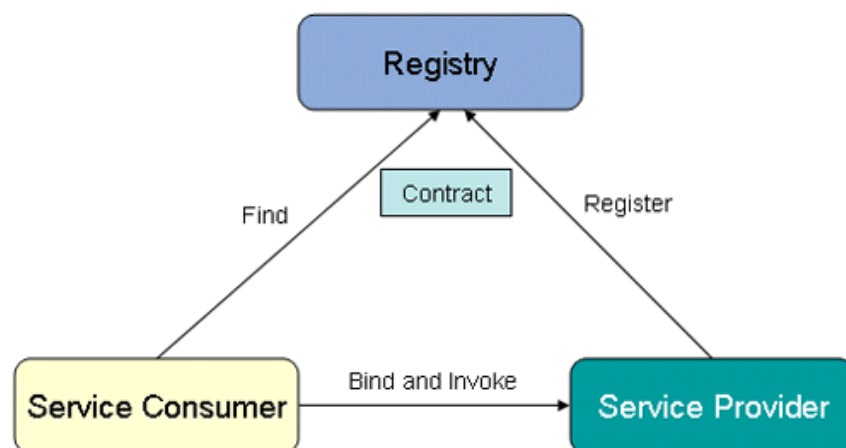
Mô hình quy trình chung phải thiết kế bao quát hết các quy trình trong hệ thống tích hợp. Để đặc tả quy trình nghiệp vụ người ta thường dùng chuẩn ngôn ngữ Business Process Execution Language [3].



**Hình 1.16. Chuẩn BPEL trên Eclipse**

### 1.2.5.2. Tích hợp hướng dịch vụ - Service-Oriented Architecture (SOA)

SOA là mô hình xây dựng ứng dụng dựa trên các dịch vụ đã có trên mạng chuyên biệt, chẳng hạn như Web. SOA giúp các thành phần liên kết với nhau một cách mềm dẻo và hiệu quả. SOA sử dụng mô hình "Tìm kiếm–Trói buộc–Thi hành" - "Find-Bind-Execute"

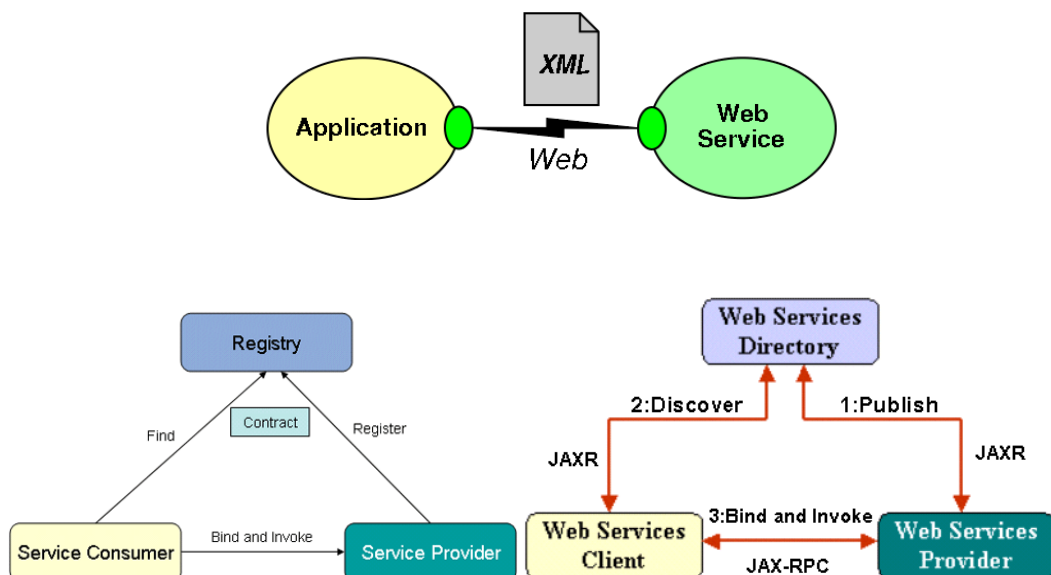


**Hình 1.17. Mô hình SOA**

## Nguyên lý cơ bản của SOA

- Liên kết không chặt chẽ: đảm bảo tính mềm dẻo của SOA, các dịch vụ có ít sự ràng buộc với nhau.
- Quyền tự trị: các dịch vụ có quyền kiểm soát dựa vào logic bên trong của dịch vụ đó
- Chia sẻ hợp đồng: các dịch vụ trong hệ thống đều hoạt động tuân thủ theo một hợp đồng chính thức.
- Đóng gói: các dịch vụ che giấu logic bên trong của mình
- Phi trạng thái: các dịch vụ hoạt động phi trạng thái
- Sử dụng lại: logic bên trong các dịch vụ đều hướng đến mục đích có thể tái sử dụng
- Có thể tìm thấy: người dùng có thể tìm kiếm dịch vụ cần sử dụng và đăng ký sử dụng dịch vụ đó

Ta có thể thấy SOA có tính mềm dẻo rất cao, tất cả các dịch vụ đều là những thành phần của hệ thống với giao diện độc lập, thực hiện một nhiệm vụ nhất định và không có mối ràng buộc chặt chẽ nào với nhau. Các dịch vụ có thể tìm kiếm được. Có thể kết hợp một số dịch vụ lại thành một dịch vụ tổng [2].



**Hình 1.18. Triển khai SOA dựa trên Web và XML**

### **1.3. Kết luận**

Chương này trình bày về các kiến thức, khái niệm và hiểu biết chung về tích hợp hệ thống. Đây là những thông tin tổng quát và cốt lõi nhất để có thể hiểu và thực hiện tích hợp hệ thống thông tin y tế.

## CHƯƠNG 2. ỨNG DỤNG RESTFUL TRONG TÍCH HỢP HỆ THỐNG THÔNG TIN Y TẾ

### 2.1. Tìm hiểu về RESTFUL

#### 2.1.1. Giới thiệu REST

REST – viết tắt của **representational state transfer** - lần đầu được giới thiệu vào năm 2000 trong luận án tiến sĩ của Roy Fielding với tựa đề "Architectural Styles and the Design of Network-based Software Architectures" (Phong cách kiến trúc và thiết kế kiến trúc phần mềm dựa trên mạng)

REST định nghĩa các quy tắc thiết kế kiến trúc Web Services, chú trọng vào tài nguyên của hệ thống, bao gồm các tài nguyên được định dạng như thế nào và được truyền tải ra sao qua giao thức HTTP. Trong thực tế, REST đã có những ảnh hưởng lớn và gần như đã thay thế SOAP và WSDL vì tính đơn giản và dễ dùng hơn những người đàn anh [4].

#### 2.1.2. Các nguyên lý cơ bản của REST

##### 2.1.2.1. Tài nguyên

Đối tượng cần quan tâm chính của dịch vụ là tài nguyên, được đặt tên bằng một URI, bất cứ thông tin nào có thể đặt tên thì đều có thể là một tài nguyên.

VD: /ws/books, /ws/persons

Tài nguyên không phải là dữ liệu, nó thể hiện cho việc dữ liệu nên được trình bày như thế nào.

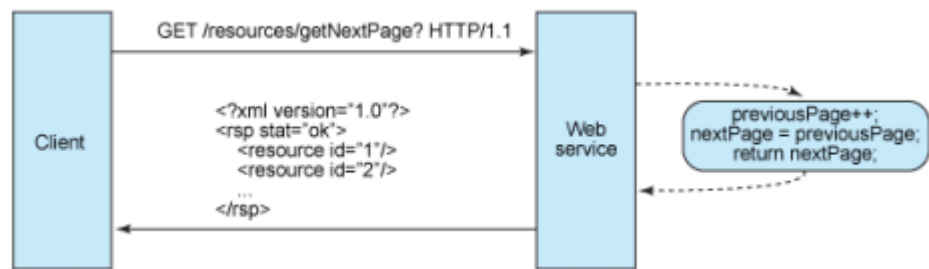
Máy chủ phải gửi phản hồi dưới một định dạng tệp cho trước và bằng một ngôn ngữ định trước

VD: Định dạng: XML, JSON, HTML, PDF, DOC,...

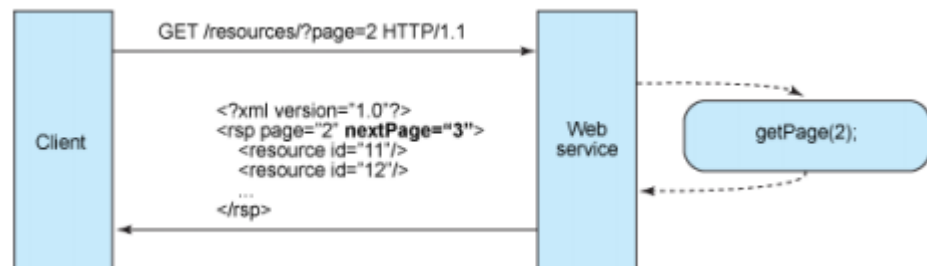
Ngôn ngữ: Tiếng Anh, Tiếng Việt,...



### 2.1.2.2. Phi trạng thái



**Hình 2.1. Thiết kế có trạng thái**



**Hình 2.2. Thiết kế phi trạng thái**

Mọi yêu cầu gửi tới máy chủ bao gồm đầy đủ mọi thông tin cần thiết để máy chủ có thể hiểu được yêu cầu. Mọi yêu cầu HTTP đều được thực hiện hoàn toàn độc lập, kết quả phản hồi của máy chủ không bao giờ bị phụ thuộc vào những yêu cầu trong quá khứ hoặc phụ thuộc vào bất cứ ngữ cảnh nào.

Không có bất cứ một thứ tự nào trong việc máy khách gửi yêu cầu lên – ví dụ như yêu cầu trang số 2 phải được gửi sau yêu cầu trang 1 của một tài nguyên nào đó.

Nếu máy chủ khởi động lại, máy khác chỉ cần gửi lại yêu cầu và tiếp tục làm việc tại trạng thái đó.

Những trạng thái có thể có của máy chủ cũng là một tài nguyên và nên có URI riêng dành cho chúng.

### 2.1.2.3. Sử dụng các phương thức HTTP một cách rõ ràng

Một đặc tính quan trọng của REST đó là sử dụng rõ ràng các phương thức của HTTP, đã được định nghĩa bởi RFC2616 [4].

**Bảng 2.1. Các phương thức sử dụng trong REST**

Phương thức HTTP	Mục đích	Thân yêu cầu	Phản hồi
GET	Truy xuất tài nguyên	Có thể không cần hoặc id của tài nguyên	Máy chủ trả về biểu diễn của tài nguyên
DELETE	Xóa tài nguyên	Id của tài nguyên	Máy chủ có thể trả về thông điệp hoặc không gì cả
PUT	Thay đổi trạng thái của tài nguyên	Máy khách gửi lên id cùng trạng thái của tài nguyên cần thay đổi	Máy chủ có thể trả về thông điệp, hoặc bản sao của tài nguyên được sửa, hoặc không gì cả
POST	Tạo tài nguyên mới	Máy khách gửi lên các trạng thái của tài nguyên cần tạo mới	Máy chủ có thể trả về thông điệp, hoặc tài nguyên được tạo, hoặc không gì cả

### **2.1.3. Một số kiến trúc phù hợp với REST**

REST thường được sử dụng kết hợp với kiến trúc phần mềm khác như Model-View-Control và Event-Base Architectures.

**Model-View-Controller (MVC)** là một kiến trúc phần mềm dựa trên việc chia các thành phần của hệ thống thành 3 phần chính:

- Model: chứa các lớp có nhiệm vụ thao tác với cơ sở dữ liệu, thực hiện việc lấy và tính toán các đối tượng của ứng dụng.
- View: Chịu trách nhiệm trong việc hiển thị các kết quả trả về cho máy khách.
- Controller: là cầu nối giữa Model và View, hoạt động như một Bộ định tuyến URI, controller tiếp nhận URI và phương thức HTTP tương ứng từ đó gọi đến Model để tính toán và trả kết quả về View để hiển thị cho máy khách.

**Event-based Architectures** là kiến trúc phần mềm ở đó tất cả các thao tác, hành động của ứng dụng được điều khiển thông qua việc thay đổi trạng thái – hay một event xảy ra, dữ liệu được chuyển tiếp thông qua các thông báo – notification.

#### **2.1.4.Đánh giá ưu và nhược điểm của RESTFUL**

REST đã đang và sẽ là một kiến trúc có ảnh hưởng lớn đến quá trình phát triển của dịch vụ Web. Tuy nhiên không phải lúc nào chúng ta cũng phải sử dụng REST, tùy vào hoàn cảnh và mục đích sử dụng chúng ta nên có sự lựa chọn phù hợp.

##### **2.1.4.1. Điểm mạnh**

REST đơn giản và rất thuận theo tư duy thông thường khi sử dụng các phương thức HTTP theo chuẩn, bởi vậy REST dễ viết và dễ điều khiển. Sử dụng REST giúp máy chủ không cần phải lo lắng đến việc xử lý phiên, hết thời gian phản hồi hay lưu trữ trạng thái, đồng thời máy khách cũng rất dễ dàng để làm việc tiếp nếu nhờ may máy chủ có sự cố. Với việc mỗi yêu cầu đều độc lập với nhau, chúng ta có thể xử lý mỗi loại yêu cầu ở một máy chủ khác nhau, tiện cho việc triển khai hệ thống. REST cũng có khả năng mở rộng cao, khả năng sử dụng dấu trang rất tốt. Cuối cùng HTTP là một giao thức phi trạng thái, bởi vậy REST sẽ tận dụng được những lợi thế của giao thức này.

##### **2.1.4.2. Điểm yếu**

Không phải dịch vụ Web nào cũng hướng đến cung cấp dịch vụ hướng tài nguyên, thay vào đó là những thao tác tính toán phức tạp, khi đó REST sẽ không phù hợp. Bên cạnh đó đôi khi các thiết đặt bảo mật có thể sẽ không cho phép tất cả các phương thức HTTP được chạy, như vậy chúng ta không thể cung cấp REST một cách đầy đủ.

## 2.2. Bài toán tích hợp hệ thống thông tin y tế

### 2.2.1. Bài toán

Như đã trình bày từ phần mở đầu, bài toán đặt ra là tích hợp một hệ thống thông tin y tế bao gồm một số phần mềm cung cấp nhiều dịch vụ khác nhau nhằm tự động hóa quá trình truy xuất kết quả thăm khám của người bệnh.

Với quy trình thăm khám hiện nay ở các cơ sở y tế, người bệnh sau khi tới làm xét nghiệm hoặc chuẩn đoán hình ảnh tại một phòng chuyên môn tại cơ sở y tế, cần phải chờ đợi tại phòng để có thể lấy kết quả mới có thể tiếp tục làm các xét nghiệm khác và mang trở lại cho bác sĩ để tiến hành chuẩn đoán bệnh.

Mục tiêu của bài toán là có thể bỏ qua bước chờ đợi tại phòng xét nghiệm, các kết quả của người bệnh có thể được bác sĩ truy xuất tự động, người bệnh chỉ cần tới phòng chuyên môn làm xét nghiệm và quay về phòng khám, giảm bớt thời gian và công sức chờ đợi.

Để đáp ứng mục tiêu đó, cần có một hệ thống cung cấp đầy đủ chức năng giúp các bác sĩ có thể nhập dữ liệu thông tin của người bệnh và có thể dễ dàng truy xuất, đồng thời cần phải được triển khai một cách dễ dàng. Cụ thể hệ thống cần đáp ứng các yêu cầu:

- Trực quan hóa: hệ thống cần cung cấp các chức năng một cách tường minh, dễ hiểu và dễ dàng sử dụng.
- Chính xác: các thông số xét nghiệm của người bệnh là cực kì quan trọng, cần đảm bảo không được có sai sót hoặc nhầm lẫn.
- Kịp thời: hệ thống cần đảm bảo việc cập nhật dữ liệu liên tục giữa các thành phần, giúp bác sĩ nhanh chóng có được kết quả của người bệnh để kịp thời chuẩn đoán.

Như vậy dựa trên những yêu cầu thực tiễn và phân tích tổng thể, bài toán đặt ra là ***"tích hợp hệ thống thông tin y tế trực quan hóa, chính xác, và kịp thời để quản lý thông tin bệnh nhân tại cơ sở y tế"***

### 2.2.2. Tổng quan về nghiệp vụ

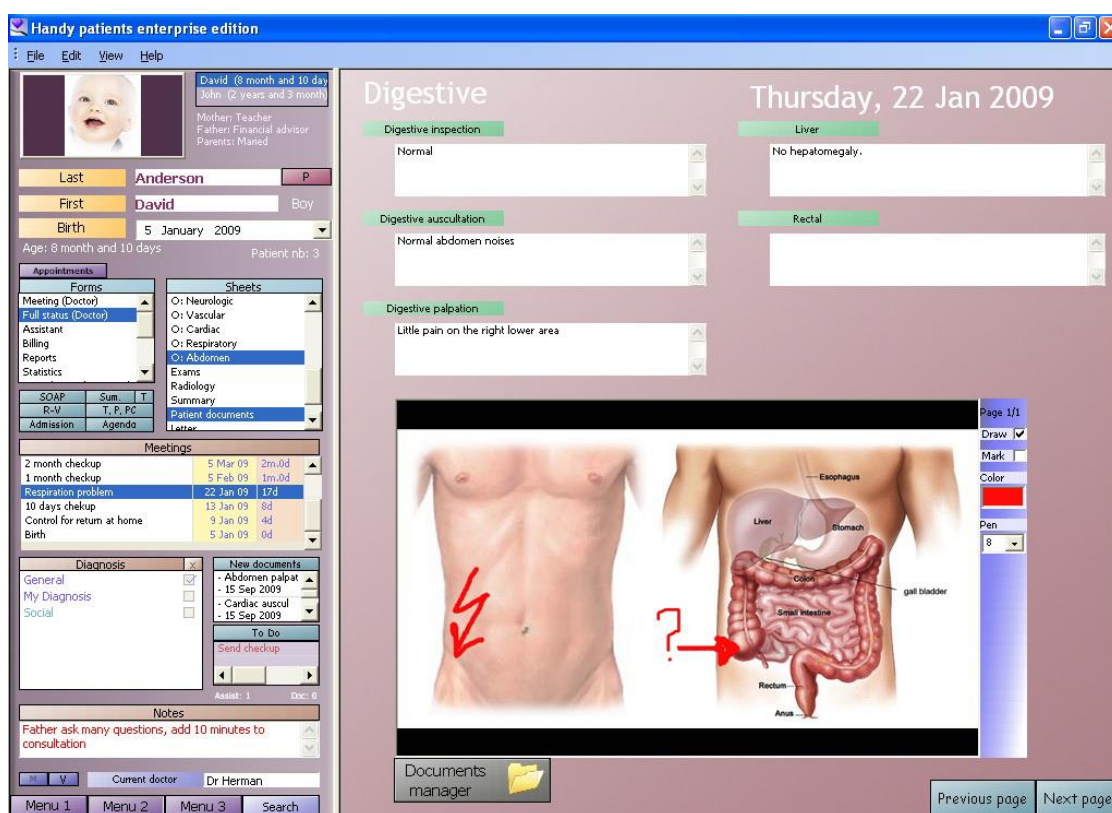
#### 2.2.2.1. Bản ghi y tế điện tử - Electronic Medical Record

Electronic medical record hay electronic health record dùng để chỉ những tập hợp có hệ thống về thông tin y tế của người bệnh được lưu trữ dưới dạng kỹ thuật

số. Những record này có thể chia sẻ giữa các hệ thống trong mạng và giữa các hệ thống thông tin khác.

EMRs có thể bao gồm các thông số xét nghiệm, hình ảnh chụp chiếu, tiền sử bệnh, tác nhân dị ứng, các dấu hiệu sống, thông tin hóa đơn...

EMRs được thiết kế để có thể lưu trữ chính xác và đảm bảo thể hiện được tình trạng của người bệnh qua thời gian. EMRs giúp cho việc tìm kiếm, cập nhật thông tin trở nên tiện lợi hơn. EMRs và các ứng dụng công nghệ thông tin đặc biệt hữu ích trong trích xuất và nghiên cứu dữ liệu của người bệnh trong quá trình theo dõi lâu dài [5].



**Hình 2.3. Ví dụ về Electronic Medical Record [5]**

#### 2.2.2.2. DICOM

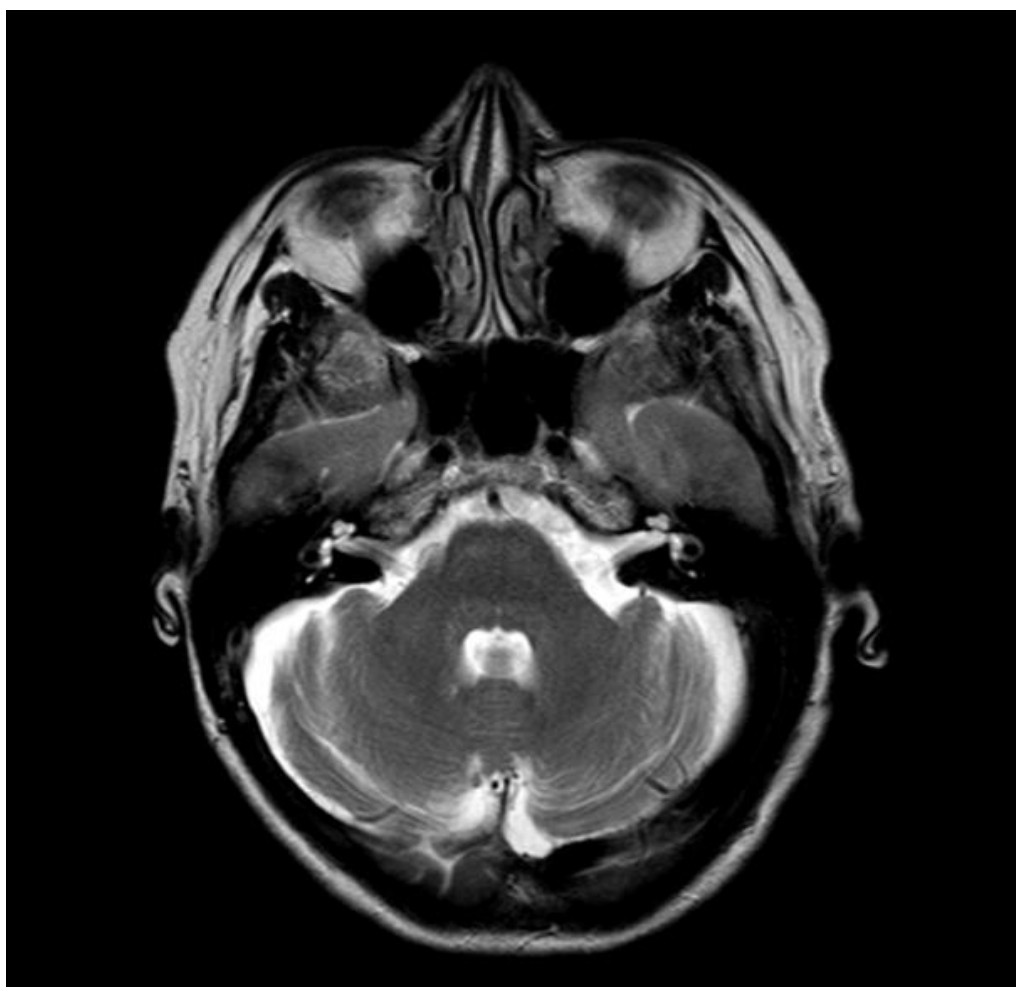
Digital Imaging and Communications in Medicine (DICOM) là một chuẩn để lưu trữ, biểu diễn và truyền tải, in ấn thông tin về hình ảnh y tế. DICOM bao gồm định nghĩa về định dạng tệp và giao thức để giao tiếp trong mạng.

DICOM hỗ trợ việc tích hợp cả phần cứng lẫn phần mềm từ nhiều nhà sản xuất khác nhau. DICOM được áp dụng rộng rãi tại các bệnh viện lớn và dần dần được áp dụng tại các cơ sở y tế quy mô nhỏ hơn.

DICOM giúp các bác sĩ có thể dễ dàng hơn trong việc chuẩn đoán tình trạng của người bệnh, đặc biệt với khả năng chia sẻ tới mọi nơi trên thế giới.

Tệp DICOM chứa cả các thông tin về người bệnh, kết quả xét nghiệm, và hình ảnh về xét nghiệm (gọi là PixelData), các thông tin này được lưu trữ và truy xuất qua các **tags**. Một tệp DICOM đơn lẻ gọi là một **instance**. Một **series** DICOM bao gồm một tập các tệp DICOM có cùng SeriesInstanceID.

DICOM được phát triển bởi National Electrical Manufactures Association [10].



*Hình 2.4. Ví dụ về phần dữ liệu hình ảnh trong DICOM [6]*

## CHƯƠNG 3. XÂY DỰNG HỆ THỐNG THÔNG TIN Y TẾ TÍCH HỢP VÀ KẾT QUẢ THỬ NGHIỆM

### 3.1. Tổng quan về hệ thống

Như đã nói, việc xây dựng một hệ thống thông tin y tế hoàn chỉnh với đầy đủ chức năng là khá khó khăn với quy mô của khóa luận. Để minh họa cho kết quả nghiên cứu, tôi xin được lựa chọn 3 module và thực hiện tích hợp một số dịch vụ hữu ích của chúng.

Danh sách các phần mềm y tế mã nguồn mở được cung cấp tại địa chỉ

[https://en.wikipedia.org/wiki/List\\_of\\_open-source\\_health\\_software](https://en.wikipedia.org/wiki/List_of_open-source_health_software)

Một số phần mềm tiêu biểu như:

**OpenEMR** : phần mềm quản lý hoạt động y tế, hồ sơ y tế điện tử, hóa đơn thuốc dựa trên PHP

**GaiaEHR**: phần mềm quản lý hồ sơ y tế điện tử dựa trên PHP và ExtJS

**Open Dental**: phần mềm quản lý hoạt động của phòng khám nha sỹ.

**DHIS**: phần mềm quản lý sức khỏe theo khu vực kiểm kho dữ liệu.

**FreeMED**: phần mềm quản lý hoạt động y tế cho phép theo dõi và nghiên cứu chi tiết về kết quả khám bệnh

**OpenHospital**: quản lý hồ sơ y tế cho trạm xá

Qua quá trình tìm hiểu, tôi quyết định lựa chọn 3 phần mềm cung cấp 3 dịch vụ y tế độc lập sau để tiến hành thử nghiệm tích hợp:

**Chikitsa** – Hospital Management System: phần mềm mã nguồn mở nền Web với chức năng quản lý bệnh nhân, lịch khám, hóa đơn.

**OpenMRS** – Open Medical Record System: phần mềm mã nguồn mở cho phép quản lý Medical Record của bệnh nhân

**Orthanc** - Open source – Lightweight DICOM server: cung cấp công cụ để quản lý, nghiên cứu hình ảnh DICOM của bệnh nhân.

Hệ thống cần có một giao diện chung để có thể sử dụng các dịch vụ được cung cấp một cách tập trung

## **3.2. Xác định yêu cầu**

### **3.2.1. Yêu cầu của hệ thống**

Hệ thống thông tin y tế tích hợp hướng đến người sử dụng là các y bác sĩ, nhân viên của cơ sở y tế nhằm mục tiêu nhanh chóng đồng bộ dữ liệu y tế của người bệnh đến các đơn vị trong cơ sở. Bởi vậy những yêu cầu đặt ra cho hệ thống cần được xác định cho đối tượng người dùng phù hợp.

Hệ thống cần đảm bảo một số chức năng cơ bản:

- Thêm người bệnh, quản lý lịch khám, quản lý thông tin chung về người bệnh: thông tin cá nhân, chi tiết hóa đơn.
- Tạo và quản lý hồ sơ y tế - medical record: bao gồm thông tin thăm khám, chuẩn đoán, hình ảnh từ các phòng chụp chiếu.
- Đồng bộ thông tin giữa các dịch vụ, đảm bảo thống nhất.

### **3.2.2. Chức năng của từng thành phần**

Các thành phần cần cung cấp các chức năng tương ứng như sau:

- Chikisa: quản lý thông tin người bệnh, lịch khám
- OpenMRS: quản lý hồ sơ y tế
- Orthanc: quản lý thông tin hình ảnh dưới định dạng DICOM

### **3.2.3. Đặc tả yêu cầu**

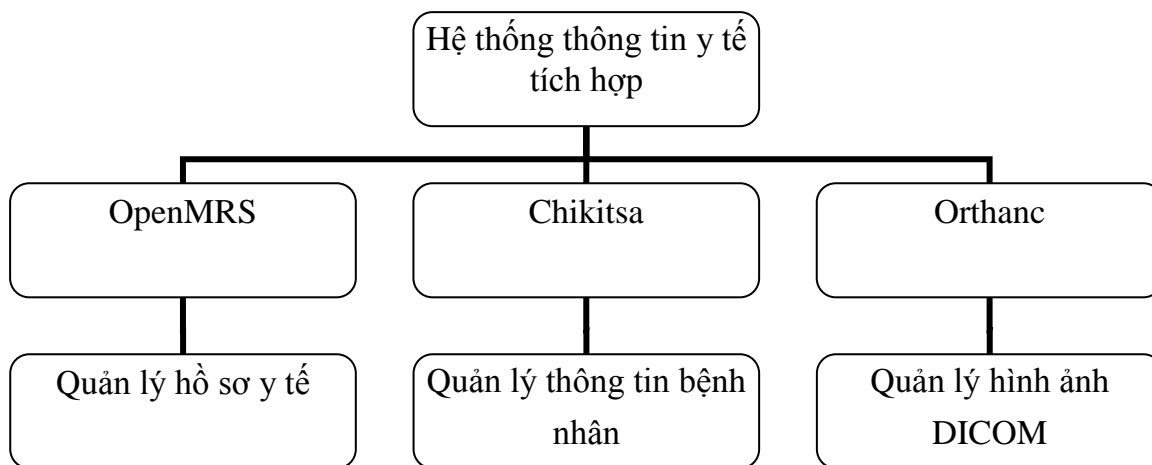
Cuộc sống càng phát triển, con người lại càng phát hiện ra nhiều mối nguy hại với sức khỏe. Đất nước càng phát triển các tác nhân gây nguy hiểm tới sức khỏe con người ngày càng nhiều, nhu cầu bảo vệ sức khỏe và khám chữa bệnh của mỗi chúng ta cũng ngày càng cao. Tuy nhiên quy trình khám chữa bệnh còn những điểm bất cập, gây hao tổn thời gian và sức lực khiến chúng ta gặp nhiều khó khăn. Vì lý do đó, nhu cầu thiết yếu cần tự động hóa những công đoạn có thể lược bỏ được để giúp người bệnh giảm bớt sự vất vả.

Với sự giúp đỡ của hệ thống, người bệnh sẽ có thể bỏ qua giai đoạn chờ đợi kết quả tại các phòng khám trung gian, kết quả sẽ tự động được bác sĩ khám bệnh truy xuất từ phòng khám của mình sau khi bệnh nhân đã làm xét nghiệm xong.



### 3.2.4.Xác định yêu cầu cụ thể

#### 3.2.4.1. Yêu cầu chức năng



**Hình 3.1. Mô hình phân rã chức năng cho toàn hệ thống**

**Bảng 3.1: Yêu cầu chức năng của hệ thống**

<b>Yêu cầu chức năng</b>	<b>Mô tả</b>
Có quản lý người bệnh	Chức năng cho phép người quản lý cơ sở y tế thêm, sửa, xóa người bệnh. Cho phép nhập, sửa các thông tin cá nhân cần thiết.
Có quản lý hồ sơ y tế	Cho phép bác sĩ tìm kiếm, tạo mới và bổ sung thông tin cần thiết vào hồ sơ y tế của người bệnh.
Có quản lý thư viện DICOM	Cho phép các bác sĩ tìm kiếm, tải lên, cập nhật và nghiên cứu các dữ liệu DICOM của người bệnh

### 3.2.5.Yêu cầu phi chức năng

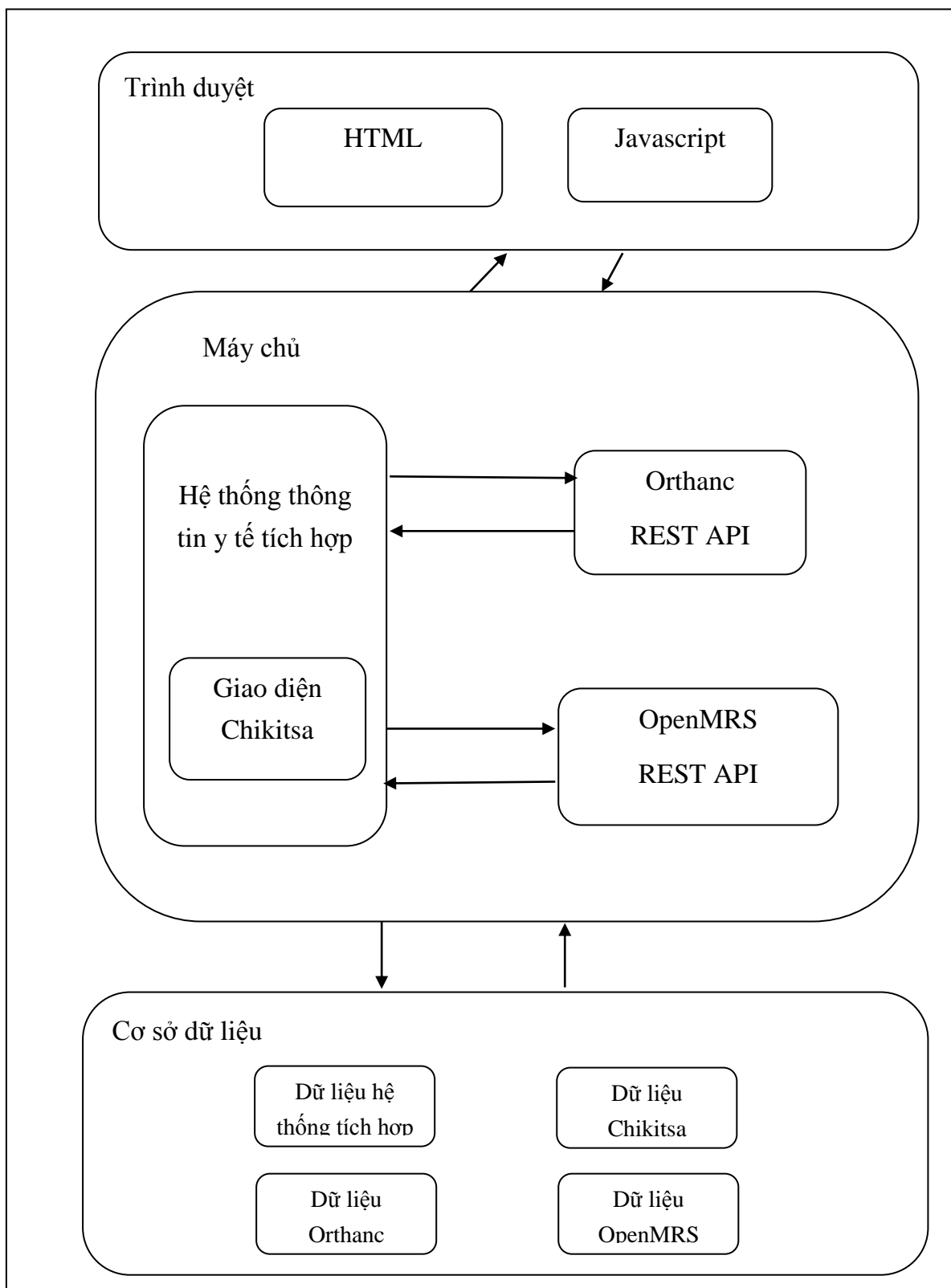
**Bảng 3.2. Bảng yêu cầu phi chức năng của hệ thống**

<b>STT</b>	<b>Yêu cầu phi chức năng</b>	<b>Mô tả</b>
1	Thông tin chính xác	Thông tin về người bệnh luân chuyển giữa các module cần phải được đảm bảo chính xác
2	Thời gian phản hồi	Thời gian đồng bộ thông tin giữa các module phải dưới 5 phút và thời gian đáp ứng yêu cầu chức năng phải dưới 5 giây đối với tác vụ bình

		thường, 10 giây đối với các tác vụ xử lý hình ảnh, phân tích thông tin
3	Tính khả dụng	Hệ thống cần có giao diện trực quan, dễ dàng làm quen và thao tác ngay lập tức
4	Tính tương thích	Hệ thống phải sử dụng được tốt trên các Web browser phổ biến hiện nay là Chrome, Cốc Cốc, Firefox, Opera
5	Công suất tối đa	Hệ thống cần được thiết kế cho ít nhất 10 phòng khám làm việc cùng lúc
6	Độ an toàn	Hệ thống cần đảm bảo an toàn cho thông tin người bệnh

### 3.3. Thiết kế kiến trúc tích hợp

Hệ thống yêu cầu tích hợp chức năng của ba phần mềm con, yêu cầu đảm bảo đáp ứng tốt yêu cầu đã được chỉ ra phía trên. Từ những phân tích này, kiến trúc của hệ thống được thiết kế như hình sau:



**Hình 3.2. Kiến trúc tổng quan hệ thống**

Kiến trúc đa tầng 3-tier được áp dụng cho hệ thống tích hợp với Presentation layer được thể hiện qua giao diện làm việc ở trình duyệt Web, Máy chủ sẽ là nơi xử lý Application logic và Cơ sở dữ liệu sẽ là Resource Manager.

Hệ thống được thiết kế trên nền Web, với HTML và Javascript là ngôn ngữ chính. Hệ thống HIS đóng vai trò là cầu nối giữa người dùng và các phần mềm tích hợp. Giao diện của Chikitsa được tích hợp tại giao diện, 2 dịch vụ Orthanc DICOM và OpenMRS được tích hợp phía máy chủ thông qua REST API.

Các phần mềm con có cơ sở dữ liệu riêng và hoạt động độc lập. Hệ thống tích hợp có vai trò đảm bảo dữ liệu được đồng nhất giữa các phần mềm con, giao tiếp thông qua mô hình RESTFUL.

### **3.4. Thiết kế chi tiết tích hợp**

#### **3.4.1. Xác định các Actor và UseCase**

##### **3.4.1.1. Danh sách các actor**

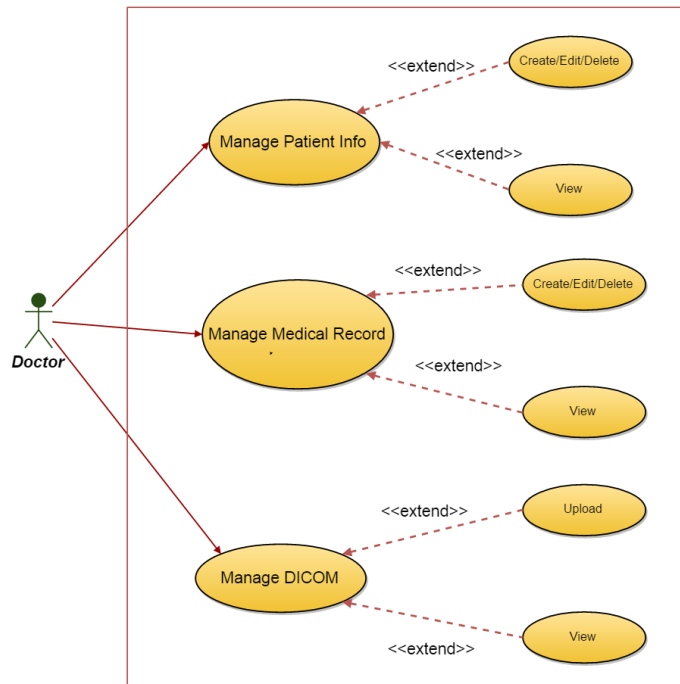
- **Người dùng bình thường:** các bác sĩ trong cơ sở y tế

##### **3.4.1.2. Danh sách các UseCase**

- Thêm, sửa, xóa, xem thông tin người bệnh
- Thêm, sửa, xóa, xem hồ sơ y tế
- Thêm và hiển thị hình ảnh từ DICOM

##### **3.4.1.3. Xây dựng biểu đồ UseCase**

###### **Biểu đồ ca sử dụng tổng quát**

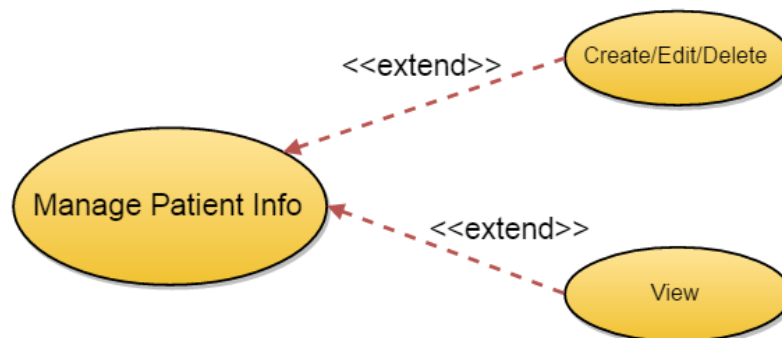


**Hình 3.3. Biểu đồ ca sử dụng tổng quát của hệ thống**

### 3.4.2. Đặc tả UseCase

#### 3.4.2.1. Ca sử dụng Quản lý thông tin người bệnh

Đây là một ca sử dụng lớn, để dễ dàng phân tích chúng ta phân rã ca sử dụng này thành 2 ca sử dụng nhỏ



**Hình 3.4. Ca sử dụng quản lý thông tin người bệnh**

#### Ca sử dụng “Thêm/Sửa/Xóa thông tin người bệnh”

- Mô tả vắn tắt:

Ca sử dụng cho phép người sử dụng tạo mới, thay đổi hoặc xóa thông tin của người bệnh.

- Luồng sự kiện

- Luồng cơ bản:

Ca sử dụng bắt đầu khi người dùng chọn sử dụng chức năng quản lý thông tin người bệnh.

Hệ thống hiển thị thông tin của những người bệnh đã có cùng các nút chức năng Thêm người bệnh, Sửa thông tin, Xóa người bệnh.

Chọn chức năng Thêm người bệnh, người dùng sẽ được chuyển đến cửa sổ mới với giao diện để nhập thông tin người bệnh mới.

Chọn chức năng Sửa thông tin, người dùng sẽ được chuyển đến cửa sổ mới với giao diện chứa các thông tin hiện tại của người bệnh và có thể sửa đổi.

Chọn chức năng Xóa người bệnh, các bản ghi về người bệnh trong cơ sở dữ liệu sẽ được xóa bỏ.

- Luồng rẽ nhánh

Kết nối đến cơ sở dữ liệu không thành công.

Hệ thống không thể kết nối đến cơ sở dữ liệu, thông báo lỗi, ca sử dụng kết thúc.

- Các yêu cầu đặt biệt (không)
- Tiền điều kiện

User đã đăng nhập. Hệ thống kết nối được đến phần mềm quản lý thông tin. Kết nối được đến cơ sở dữ liệu.

- Hậu điều kiện

Nếu ca sử dụng thành công, thông tin về người bệnh mới, thông tin sửa đổi được cập nhật, hoặc bản ghi về người bệnh được xóa khỏi cơ sở dữ liệu và không thể hiển thị được nữa.

Nếu ca sử dụng không thành công hệ thống không thay đổi.

### **Ca sử dụng “Xem thông tin người bệnh”**

- Mô tả vắn tắt

Ca sử dụng cho phép người dùng xem chi tiết thông tin về người bệnh được chọn.

- Luồng sự kiện

- Luồng cơ bản

Ca sử dụng bắt đầu khi người dùng lựa chọn một người bệnh từ danh sách.

Người dùng được chuyển sang cửa sổ mới hiển thị đầy đủ thông tin đã đăng ký của người bệnh như Họ tên, Địa chỉ, Số điện thoại, Ngày tháng năm sinh, Thư điện tử.

- Luồng rẽ nhánh

Không thể kết nối đến cơ sở dữ liệu.

Kết nối đến cơ sở dữ liệu không thành công, thông báo lỗi và kết thúc ca sử dụng.

- Các yêu cầu đặc biệt (không)
- Tiền điều kiện

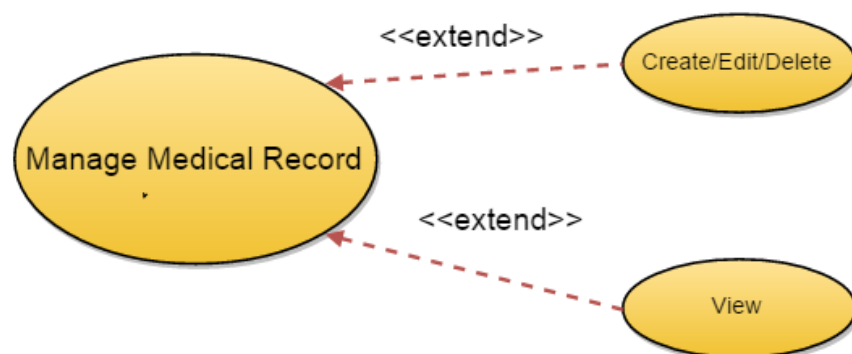
Người dùng đã đăng nhập. Kết nối thành công đến phần mềm quản lý thông tin người bệnh và cơ sở dữ liệu.

- Hậu điều kiện

Hiển thị thông tin người bệnh nếu ca sử dụng thành công.

### 3.4.2.2. Ca sử dụng quản lý hồ sơ y tế

Đây là một ca sử dụng lớn, ta tiến hành phân rã nó thành 2 ca sử dụng nhỏ hơn.



**Hình 3.5. Ca sử dụng quản lý hồ sơ y tế**

#### Ca sử dụng “Thêm/Sửa/Xóa hồ sơ y tế”

- Mô tả vắn tắt:

Ca sử dụng cho phép người sử dụng tạo mới, thay đổi hoặc xóa Hồ sơ y tế của người bệnh

- Luồng sự kiện

- o Luồng cơ bản:

Ca sử dụng bắt đầu khi người dùng chọn sử dụng chức năng quản lý hồ sơ y tế.

Hệ thống hiển thị thông tin của những người bệnh đã có cùng các nút chức năng Thêm hồ sơ, Sửa hồ sơ, Xóa hồ sơ.

Chọn chức năng Thêm hồ sơ, người dùng sẽ được chuyển đến cửa sổ mới với giao diện để tạo hồ sơ mới người bệnh.

Chọn chức năng Sửa hồ sơ, người dùng sẽ được chuyển đến cửa sổ mới với giao diện chứa các thông tin hiện tại của hồ sơ người bệnh và có thể sửa đổi.

Chọn chức năng Xóa hồ sơ, các bản ghi y tế về người bệnh trong cơ sở dữ liệu sẽ được xóa bỏ.

- o Luồng rẽ nhánh

Kết nối đến cơ sở dữ liệu không thành công.

Hệ thống không thể kết nối đến cơ sở dữ liệu, thông báo lỗi, ca sử dụng kết thúc.

- o Các yêu cầu đặt biệt (không)

- o Tiền điều kiện

User đã đăng nhập. Hệ thống kết nối được đến phần mềm quản lý hồ sơ y tế. Kết nối được đến cơ sở dữ liệu.

- o Hậu điều kiện

Nếu ca sử dụng thành công, thông tin về hồ sơ người bệnh mới, thông tin sửa đổi được cập nhật, hoặc bản ghi về hồ sơ người bệnh được xóa khỏi cơ sở dữ liệu và không thể hiển thị được nữa.

Nếu ca sử dụng không thành công hệ thống không thay đổi.

### **Ca sử dụng “Xem hồ sơ y tế”**

- Mô tả vấn đề

Ca sử dụng cho phép người dùng xem chi tiết thông tin về người bệnh được chọn.



- Luồng sự kiện
  - o Luồng cơ bản

Ca sử dụng bắt đầu khi người dùng lựa chọn một người bệnh từ danh sách và chọn xem hồ sơ.

Người dùng được chuyển sang cửa sổ mới hiển thị đầy đủ thông tin đã có về các kết quả thăm khám của người bệnh.

- o Luồng rẽ nhánh

Không thể kết nối đến cơ sở dữ liệu.

Kết nối đến cơ sở dữ liệu không thành công, thông báo lỗi và kết thúc ca sử dụng.

- o Các yêu cầu đặc biệt (không)
- o Tiền điều kiện

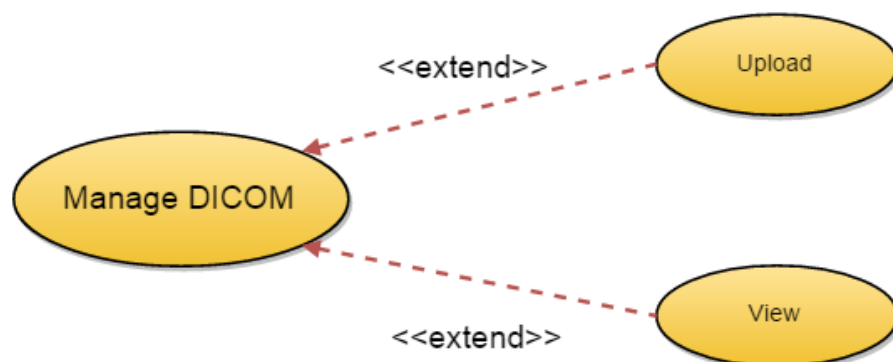
Người dùng đã đăng nhập. Kết nối thành công đến phần mềm quản lý hồ sơ y tế và cơ sở dữ liệu.

- o Hậu điều kiện

Nếu ca sử dụng thành công hiển thị các thông tin khám bệnh của người bệnh.

### 3.4.2.3. Ca sử dụng Quản lý hình ảnh DICOM

Đây là ca sử dụng lớn ta tiến hành phân ra nó ra thành 2 ca sử dụng nhỏ



**Hình 3.6. Ca sử dụng Quản lý hình ảnh DICOM**

#### Ca sử dụng “Tải lên tệp DICOM”

- Mô tả ngắn gọn:

Ca sử dụng cho phép người dùng có thể tải lên các tệp DICOM để lưu trữ và quan sát.

- Luồng sự kiện
  - Luồng cơ bản

Ca sử dụng bắt đầu khi người dùng chọn chức năng Tải lên tệp DICOM.

Hệ thống hiển thị cửa sổ cho phép người dùng kéo thả tập các tệp DICOM để tải lên. Sau khi người dùng nhấn nút Tải lên hệ thống sẽ lưu trữ các tệp DICOM trong cơ sở dữ liệu của mình.

- Luồng rẽ nhánh

Tệp DICOM tải lên không thành công. Thông báo lỗi cho người dùng và kết thúc ca sử dụng

- Các yêu cầu đặc biệt (không)
- Tiên điều kiện

Người dùng đã đăng nhập. Kết nối thành công đến phần mềm quản lý DICOM.

- Hậu điều kiện

Nếu ca sử dụng thành công, các tệp DICOM được lưu trữ trên hệ thống.

Nếu ca sử dụng không thành công hệ thống không có thay đổi.

### **Ca sử dụng “Trình diễn dữ liệu hình ảnh DICOM”**

- Mô tả ngắn gọn

Ca sử dụng cho phép người dùng xem trình diễn dữ liệu hình ảnh của các tệp DICOM đã tải lên

- Luồng sự kiện
  - Luồng cơ bản

Ca sử dụng bắt đầu khi người dùng chọn xem dữ liệu hình ảnh DICOM của người bệnh.

Hệ thống hiển thị dữ liệu hình ảnh của tệp DICOM đã tải lên

- Luồng rẽ nhánh

Không thể kết nối đến cơ sở dữ liệu, thông báo lỗi cho người dùng và kết thúc ca sử dụng.

- Các điều kiện đặc biệt (không)
- Tiên điều kiện

Người dùng đã đăng nhập, kết nối thành công đến phần mềm quản lý DICOM và cơ sở dữ liệu.

- Hậu điều kiện

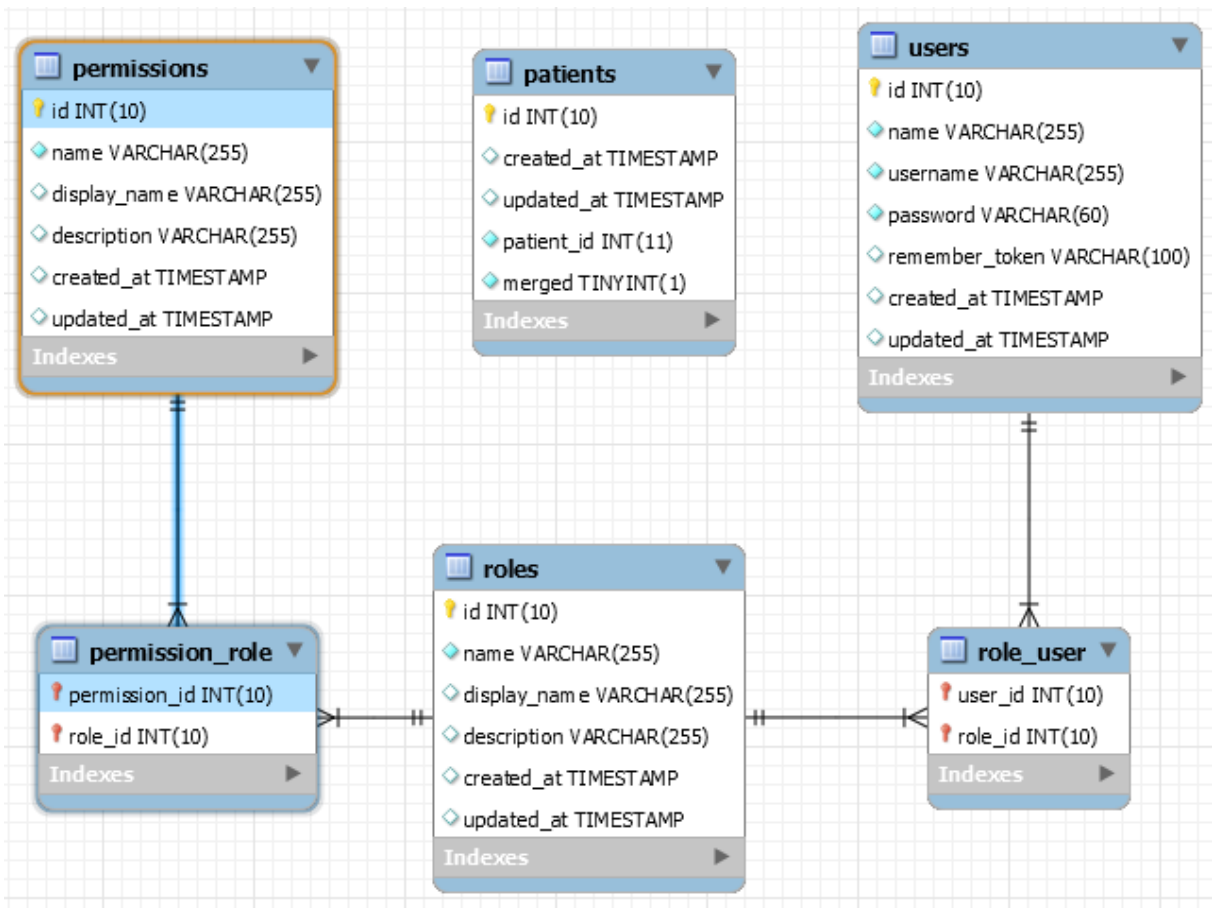
Nếu ca sử dụng thành công dữ liệu hình ảnh DICOM được hiển thị cho người dùng.

Nếu ca sử dụng không thành công hệ thống không có thay đổi.

### **3.4.3. Thiết kế cơ sở dữ liệu**

Các phần mềm tích hợp trong hệ thống hoạt động độc lập và có cấu trúc cơ sở dữ liệu riêng. Các phần mềm trong hệ thống đều sử dụng CSDL quan hệ, trừ Orthanc có cấu trúc lưu dữ liệu dạng tệp riêng.

Hệ thống cần thêm một cơ sở dữ liệu để lưu thông tin đăng nhập của hệ thống tổng thể, đồng thời yêu cầu phải có cơ chế phân quyền để tránh việc lạm quyền trong quá trình làm việc của các y bác sĩ.



**Hình 3.7. Lược đồ dữ liệu của hệ thống tích hợp**

Thông tin đăng nhập người dùng được lưu trữ tại bảng users.

Hệ thống phân quyền hoạt động dựa trên các bảng roles, user\_role, permissions, permission\_role. Người dùng có role và 1 role chứa nhiều permission. Những permission này sẽ quyết định người dùng được phép làm gì và không được làm gì trong hệ thống tích hợp của chúng ta.

Ví dụ một hệ thống phân quyền như sau

**Bảng 3.3. Bảng users**

id	name	username	password
1	Admin	admin	\$2y\$10\$AdttjiOWE0wXE8q5dTwwLukOVmBgb
2	Dr Thai	thaiph	\$2y\$11\$jigeiSDOrih0waE8agejFeGGeieHeeEFEH
3	Ex The	thehn	\$2y\$1085hhugIGiuejgEHJjE954JG94kgDfE58Gh
4	Sf Thanh	thanhnv	\$2y\$10ghGEgeh6Glhei\$ghi\$ihfeoHGIE54GegyEl

***Bảng 3.4. Bảng roles***

id	name	display_name	description
1	doctor	Doctor	Overall doctor
2	examiner	Examiner	Examiner
3	staff	Staff	Normal staff
4	admin	Admin	Administrator

***Bảng 3.5. Bảng permissions***

id	name	display_name	description
1	manage-chikitsa	Manage chikitsa	Can manage patient on Chikitsa
2	view-orthanc	View Orthanc	Can view DICOM on Orthanc
3	upload-orthanc	Upload Orthanc	Upload DICOM to Orthanc
4	view-openmrs	View OpenMRS	Can view patient medical record on OpenMRS
5	create-openmrs	Create OpenMRS	Can create patient medical record on OpenMRS
6	update-openmrs	Update OpenMRS	Can update patient medical record on OpenMRS
7	delete-openmrs	Delete OpenMRS	Can delete patient medical record on OpenMRS
8	manage-user	Manage User	Can manage users of the system

Sau khi đã có 3 bảng users, roles và permissions, ta tiến hành gán Quyền cho Vai trò và gán Vai trò cho người dùng thông qua 2 bảng permission\_role và role\_user

**Bảng 3.6. Bảng *permission\_role***

role_id	permission_id
1	1
1	2
1	4
1	6
2	3
2	4
2	5
2	6
2	7
3	1
4	8

**Bảng 3.7. Bảng *role\_user***

user_id	role_id
1	4
2	1
3	2
4	3

Như vậy với hệ thống cơ sở dữ liệu như vậy, cùng với việc xử lý phân quyền tại máy chủ, chúng ta có 4 người dùng có vai trò khác nhau trong hệ thống.

**Admin:** người quản trị hệ thống chỉ có vai trò cung cấp tài khoản sử dụng cho người dùng, Admin không có quyền tương tác với hệ thống y tế tích hợp.

**Dr Thai:** có vai trò là một bác sĩ tổng thể, người dùng này có thể quản lý bệnh nhân bằng hệ thống Chikitsa và được quyền xem cùng với cập nhật hồ sơ y tế

của người bệnh trên hệ thống OpenMRS, cũng như xem dữ liệu DICOM của người bệnh trên Orthanc.

**Ex The:** có vai trò là xét nghiệm viên, người dùng không được quyền thao tác với Chikitsa để thêm người bệnh, tuy nhiên chỉ người dùng có vai trò này mới có thể tạo mới hồ sơ y tế và xóa hồ sơ y tế của người bệnh. Xét nghiệm viên cũng là người có quyền tải lên tài nguyên DICOM của người bệnh tới Orthanc.

**Sf Thanh:** có vai trò là nhân viên của cơ sở y tế, chỉ có quyền quản lý hệ thống Chikitsa để thêm, sửa thông tin người bệnh, hỗ trợ cho bác sĩ tổng thể.

Ngoài ra bảng patient là một bảng nhỏ hỗ trợ cho việc đồng bộ hóa dữ liệu người dùng giữa 2 dịch vụ Chikitsa và OpenMRS, những người bệnh đã được đồng bộ được đánh dấu trong bảng này.

## 3.5. Thử nghiệm và triển khai

### 3.5.1. Phát triển hệ thống

#### 3.5.1.1. Tổng quan cấu trúc chương trình

Để có thể tiết kiệm được công sức khi bắt tay vào thực hiện minh họa cho khóa luận, chương trình tích hợp được viết trên framework php Laravel 5, là một framework php khá phổ biến thời điểm hiện tại.

Hai phần mềm Orthanc và OpenMRS được triển khai dưới dạng RESTFUL API. Phần mềm Chikitsa được tích hợp và truy xuất từ bên trong thư mục public của framework Laravel.

Phần mềm Orthanc cung cấp một gói dịch vụ độc lập (stand-alone), được cài đặt vào hệ điều hành Windows như một dịch vụ của hệ thống.

Phần mềm OpenMRS được cung cấp dưới dạng tệp WAR, triển khai thông qua dịch vụ máy chủ Tomcat. OpenMRS cũng cung cấp một gói stand-alone bao gồm đầy đủ máy chủ Tomcat, MySQL, tuy nhiên, để đồng bộ với việc triển khai framework Laravel và Chikitsa, tôi quyết định sử dụng module máy chủ Tomcat của XAMPP để triển khai.

Framework Laravel cùng Chikitsa được triển khai dựa trên module máy chủ Apache của gói XAMPP.

Cấu trúc thư mục của hệ thống dựa trên cấu trúc của framework Laravel 5 theo mô hình MVC

**Bảng 3.8. Cấu trúc thư mục của dự án**

app\ controller\ service\ public\ chikitsa_server\ resources\ 	Thư mục chứa các tệp xử lý Logic của hệ thống  Thư mục chứa phần Điều khiển - Controller của hệ thống  Thư mục làm việc với các hệ thống con  Thư mục chứa các tài nguyên tĩnh của hệ thống  Thư mục chứa phần mềm Chikitsa  Thư mục chứa hệ thống Giao diện - View của hệ thống
----------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### **3.5.1.2. Sử dụng RESTFUL để truy cập các dịch vụ thành phần**

Orthanc và OpenMRS là hai dịch vụ có hỗ trợ RESTFUL để sử dụng các tính năng của chúng. Để làm điều này hệ thống sử dụng cURL để gửi yêu cầu và nhận phản hồi. Hệ thống có sử dụng một thư viện hỗ trợ cURL được thuận tiện hơn đó là php-curl của tác giả juanpborda<sup>[9]</sup>.

Với mỗi truy vấn lên máy chủ RESTFUL, một yêu cầu sẽ bao gồm đầy đủ mọi thông tin cần thiết để máy chủ có thể biết được người dùng có quyền gì và có thể sử dụng những dịch vụ nào của nó.

OpenMRS yêu cầu xác thực theo phương thức Basic – gửi lên kèm truy vấn một mã hóa base64 của bộ username:password, trong khi đó Orthanc không yêu cầu người dùng phải xác thực thông tin. Bởi vậy để đảm bảo tính an toàn dữ liệu, hệ thống chỉ nên được cài đặt để chạy trong nội bộ cơ sở y tế.



```

$url = static::BASE_PATH . 'person?q=' . $key;
$request = curl::newRequest('GET', $url)->setOption(CURLOPT_USERPWD, static::USERPWD);

$res = $request->send();

if ($res->statusCode == '200') {
    $res = json_decode($res->body);
    $res = $res->results;
    $patients = [];
    foreach ($res as $key => $r) {
        $patient_url = static::BASE_PATH . 'person/' . $r->uuid;
        $request = curl::newRequest('GET', $patient_url)->setOption(CURLOPT_USERPWD, static::USERPWD);

        $res2 = $request->send();

        $patient = [];

        if ($res2->statusCode == '200') {
            $res2 = json_decode($res2->body);
            $patient['uuid'] = $r->uuid;
            $patient['name'] = $res2->display;
            $patient['gender'] = $res2->gender == 'F' ? 'Female' : 'Male';
            $patient['birthdate'] = date('d-m-Y', strtotime($res2->birthdate));
            $patient['age'] = $res2->age;
            $patient['deathdate_estimated'] = $res2->deathdateEstimated;

            array_push($patients, $patient);
        }
    }

    return $patients;
}

```

**Hình 3.8. Đoạn mã mô tả quy trình sử dụng OpenMRS REST API**

Hình phía trên mô tả một truy vấn tới REST API của phần mềm OpenMRS để yêu cầu trả về thông tin của người bệnh với từ khóa trong tên được định trước. Để lấy được thông tin về tài nguyên người dùng cần, phải biết trước được URI của tài nguyên đó. Thông tin trả về đều ở dạng JSON, rất thuận tiện cho việc phân tích cú pháp để đưa vào mảng hiển thị. Khi đã có được URI của tài nguyên, sử dụng cURL để gửi yêu cầu và nhận phản hồi.

```

$patients_url = static::BASE_PATH . '/patients';

$curl = curl::get($patients_url);

if ($curl->statusCode == 200) {
    $res = json_decode($curl->body);

    $patients = [];

    foreach ($res as $key => $r) {
        $patient_info_url = $patients_url . '/' . $r;

        $curl = curl::get($patient_info_url);

        $res = json_decode($curl->body);
        $patient = [];
        $patient['ID'] = $res->ID;
        $res = $res->MainDicomTags;
        $patient['name'] = $res->PatientName;
        $patient['birthday'] = isset($res->PatientBirthDate) ? date('d-m-Y', strtotime($res->PatientBirthDate)) : '';
        $patient['patientID'] = $res->PatientID;

        array_push($patients, $patient);
    }

    return $patients;
}

```

### ***Hình 3.9. Quy trình sử dụng Orthanc REST API***

Tương tự ta cũng sử dụng cURL để làm việc với Orthanc, sau khi nhận phản hồi, ta trích xuất những thông tin cần thiết để gửi trở lại cho Controller xử lý.

#### **3.5.1.3. Xử lý những những giao diện của phần mềm không RESTFUL**

Bên cạnh OpenMRS và Orthanc có hỗ trợ RESTFUL API, Chikitsa đơn thuần là một phần mềm quản lý thông thường và không hỗ trợ RESTFUL, bởi vậy để sử dụng chức năng của Chikitsa bên cạnh các dịch vụ còn lại, giải pháp là những giao diện làm việc của Chikitsa vào trang chủ của hệ thống.

```

@section('content')
<div class="container">
    <div class="row">
        <div class="col-md-12">
            <iframe src="" id="chikitsa_frame" height="550" width="100%" frameborder="0"></iframe>
        </div>
    </div>
</div>
@endsection
@section('script')
<script type="text/javascript">
    $(document).ready(function() {
        $.ajax({
            url: 'chikitsa_server/index.php/login/valid_signin',
            type: 'POST',
            data: {username: 'admin', password: 'admin'}
        })
        .complete(function(res, status) {
            var refresh = res.getResponseHeader('refresh');
            var link = refresh.substring(refresh.indexOf('=')+1);
            $('#chikitsa_frame').prop('src', link);
            $('#chikitsa_frame').show();
        });
    });
</script>
@endsection

```

**Hình 3.10. Những giao diện làm việc của Chikitsa vào trang chủ**

Ta sử dụng một iframe và gọi đến URL của Chikitsa, như vậy giao diện làm việc của Chikitsa sẽ hiển thị tại trang chủ của hệ thống.

```

<div class="row" id="view_dicom_div" style="display:none">
    <div class="col-md-10 col-md-offset-1">
        <div class="panel panel-default">
            <iframe src="" id="view_dicom_frame" height="550" width="100%" frameborder="0"></iframe>
        </div>
    </div>
</div>
</div>
@endsection
@section('script')
<script src="{{asset('/script/orthanc.js')}}" type="text/javascript" charset="utf-8" async defer></script>
<script>
    $(document).ready(function() {
        $('body').on('click', '.view_dicom', function(event) {
            var id = $(this).attr('data-id');
            var href = "http://localhost:8042/web-viewer/app/viewer.html?series=" + id;

            $('#view_dicom_frame').attr('src', href);
            $('#view_dicom_div').show();
        });
        $('body').on('click', '#hide_dicom_div', function(event) {
            $('#view_dicom_div').hide();
        });
    });
</script>
@endsection

```

**Hình 3.11. Những module trình diễn dữ liệu hình ảnh của DICOM**

Việc truy xuất các dữ liệu từ máy chủ của Orthanc được thực hiện tốt qua REST API tuy nhiên việc trình diễn dữ liệu hình ảnh thì không thể thông qua

REST mà phải phụ thuộc vào module WebViewer của Orthanc, bởi vậy, chúng ta cần thêm một bước nhúng giao diện của module này vào màn hình làm việc của hệ thống. Hình trên là bước thực hiện công việc này.

Tương tự với tác vụ tải lên tệp DICOM, với giả định chúng ta đã có tập hợp các tệp DICOM và để có thể xem được dữ liệu hình ảnh từ chúng, ta phải thực hiện tải lên máy chủ Orthanc. Với giao diện kéo thả tiện lợi, việc tải tệp DICOM lên rất dễ dàng với Orthanc, ta chỉ cần nhúng giao diện của Orthanc vào giao diện hiển thị của hệ thống.

### **3.5.2. Môi trường thử nghiệm**

#### **Phần cứng**

Do chưa có nhiều kinh nghiệm và điều kiện thực tế nên tôi chưa thể tính toán được cấu hình phần cứng phù hợp để triển khai hệ thống. Qua kiểm nghiệm tại máy cá nhân, chạy Windows 8 với cấu hình:

- CPU: Intel(R) Core(TM) i5-3210M CPU @ 2.5GHz
- RAM: 4GB
- Hard driver: 500GB

Hệ thống phần nào đáp ứng được yêu cầu với không truy vấn nào có thời gian lớn hơn 5s. Tuy nhiên đây mới chỉ là thử nghiệm đơn lẻ, với tập dữ liệu mẫu nhỏ, chưa thể khẳng định được sự hiệu quả trong hoạt động của hệ thống.

#### **Phần mềm**

Cài đặt trên Máy chủ:

- Operating System: Windows 8 x84\_64
- Xampp 5.5.33
- PHP 5.5.33
- Apache: 2.4.18
- MariaDB: 10.1.10
- Tomcat: 7.0.56
- Composer

- Laravel 5 framework và một số module hỗ trợ Laravel: bootstrap, jquery, php-curl

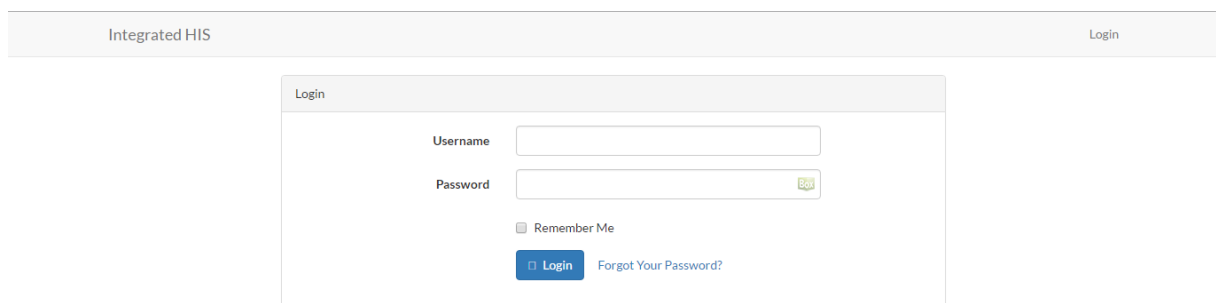
Cài đặt phía Máy khách:

- Chrome 50, Opera 37, Firefox 43, Internet Explorer 10

### 3.5.3. Thử nghiệm và đánh giá

#### 3.5.3.1. Giao diện đăng nhập

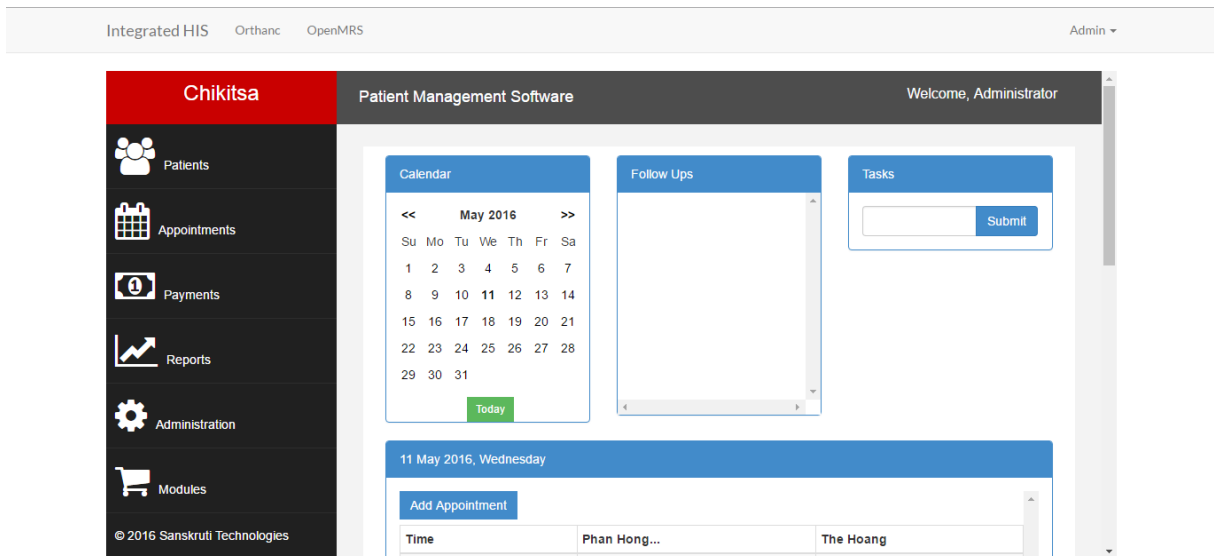
Khi truy cập vào hệ thống, người dùng trước hết phải đăng nhập để có thể sử dụng được các chức năng. Tên đăng nhập và mật khẩu người dùng được nhận từ Quản trị viên.



**Hình 3.12. Giao diện đăng nhập hệ thống**

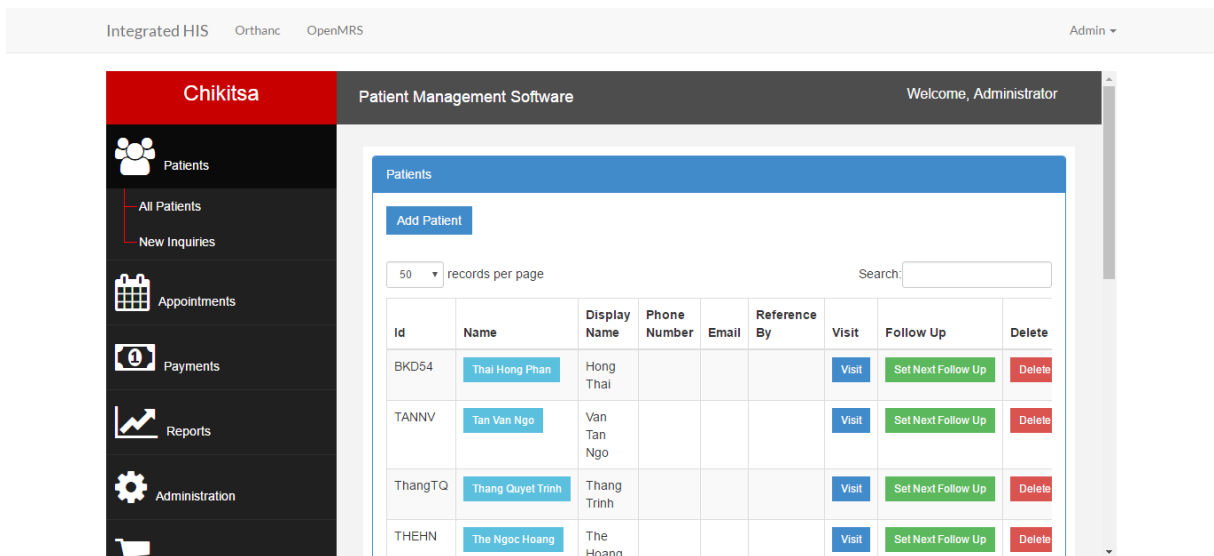
#### 3.5.3.2. Tích hợp giao diện Quản lý người bệnh của chikitsa

Giao diện quản lý thông tin người bệnh sử dụng Giao diện của phần mềm Chikitsa. Chúng ta xử lý nhúng giao diện hoạt động của Chikitsa vào bên trong trang chủ của hệ thống tích hợp, bởi vậy cách sử dụng Chikitsa so với khi nó đứng một mình là tương đồng.



**Hình 3.13. Giao diện trang chủ của Chikitsa**

Giao diện của chương trình bao gồm thanh điều hướng ngang phía trên cùng để truy cập các dịch vụ, phần nội dung bên dưới sẽ là các chức năng để sử dụng.



**Hình 3.14. Danh sách người bệnh**

Với Giao diện của Chikitsa, truy cập các chức năng sử dụng thanh điều hướng ở phía bên trái. Hình 3.14 thể hiện màn hình hiển thị danh sách người bệnh trong hệ thống.

Integrated HIS Orthanc OpenMRS Admin

**Chikitsa** Patient Management Software Welcome, Administrator

**Patient**

Name: First Name Middle Name Last Name

Patient ID: [Text Field]

Display Name: [Text Field] Choose File No file chosen

Gender: ☒ Male ☐ Female

Date Of Birth: [Text Field]

Age: [Text Field] Oops! Could not calculate age!

Address Type: Home

Address Line 1: [Text Field]

© 2016 Sanskruti Technologies

**Hình 3.15.. Chức năng thêm người bệnh mới**

Integrated HIS Orthanc OpenMRS Admin

**Chikitsa** Patient Management Software Welcome, Administrator

**Patient**

Name: Thai Hong Phan

Patient ID: BKD54

Display Name: Hong Thai Choose File No file chosen

Gender: ☒ Male ☐ Female

Date Of Birth: 01-05-2016

Age: Only 10 days old!

Address Type: Home

Address Line 1: My Dinh - Nam Tu Liem

© 2016 Sanskruti Technologies

**Hình 3.16. Giao diện thông tin chi tiết người bệnh và Sửa thông tin người bệnh**

### 3.5.3.3. Tích hợp chức năng quản lý hồ sơ y tế của OpenMRS

Với RESTFUL API của OpenMRS, chúng ta cần xây dựng một giao diện mới để có thể hiển thị những dữ liệu được lấy về.

Integrated HIS
Orthanc
OpenMRS
Admin

Patients

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Name	Gender	Birthdate	Age	Deathdate estimated
Thanh Viet Nguyen	Male	13-03-1994	22	None

1

**Hình 3.17. Giao diện danh sách người bệnh trong cơ sở dữ liệu về hồ sơ y tế**

Chức năng quản lý hồ sơ y tế truy cập dữ liệu từ máy chủ thông qua RESTFUL API. OpenMRS API không cho phép liệt kê tất cả người bệnh mà chỉ cho tìm kiếm, bởi vậy để thuận lợi hơn, giao diện thiết kế bổ sung filter theo chữ cái giúp việc tìm kiếm dễ dàng hơn.

Integrated HIS
Orthanc
OpenMRS
Admin

Patients > Thanh Viet Nguyen

Name	Gender	Patient Birthday	Age
Thanh Viet Nguyen	Male	13-03-1994	22

Patient encounters

Encounter		
ADULTRETURN 09/04/2016	<input type="button" value="Observation"/>	<input type="button" value="Delete"/>
ADULTINITIAL 09/05/2016	<input type="button" value="Observation"/>	<input type="button" value="Delete"/>
PEDSINITIAL 09/05/2016	<input type="button" value="Observation"/>	<input type="button" value="Delete"/>

1

PEDSINITIAL 09/05/2016

X-RAY, SPINE	NORMAL	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
BLOOD TYPING	A NEGATIVE	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
REASON PCP PROPHYLAXIS STOPPED	CD4 COUNT GREATER THAN 15%	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
X-RAY, SHOULDER	NORMAL	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>

**Hình 3.18. Một bản ghi kết quả khám của người bệnh**



Integrated HIS   Orthanc   **OpenMRS**   Admin ▾

Patients > Thanh Viet Nguyen

Add encounter

Name	Gender	Patient Birthday	Age
Thanh Viet Nguyen	Male	13-03-1994	22

Encounter Type: ADULTINITIAL ▾

Submit

***Hình 3.19. Tạo bản ghi thăm khám mới***

Integrated HIS   Orthanc   **OpenMRS**   Admin ▾

Patients > Thanh Viet Nguyen

Add observation

Name	Gender	Patient Birthday	Age
Thanh Viet Nguyen	Male	13-03-1994	22

PEDIINITIAL 09/05/2016

Concept question: BLOOD TYPING ▾


Concept answer: B NEGATIVE ▾

Submit

***Hình 3.20. Thêm kết quả xét nghiệm***

### **So sánh với giao diện mặc định**

OpenMRS cũng cung cấp một giao diện Web cho phép thực hiện các thao tác quản lý đối với hồ sơ y tế tùy nhiên theo đánh giá cá nhân, giao diện mặc định của OpenMRS khá khó dùng và phức tạp, chưa kể đến còn rất nhiều chức năng không thể thực hiện đúng. Bởi vậy việc thiết kế lại giao diện cho hệ thống tích hợp là giải pháp tốt hơn.



OpenMRS

Currently logged in as Super User | [Log out](#) | [My Profile](#) | [Help](#)

[Home](#) | [Find/Create Patient](#) | [Dictionary](#) | [Administration](#)

[Admin](#) | [Manage Persons](#) | [Manage Relationship Types](#) | [Manage Person Attribute Types](#)

### Person

[Create Person](#)

#### Find a Person

Person Name:  ☐ Include Deleted


Viewing results for 'Thanh'

Given	Middle	Family Name	Age	Gender	Birthdate
Thanh	Viet	Nguyen	22	M	14-Mar-1994

Showing 1 to 1 of 1 entries

Show  entries

**Hình 3.21. Giao diện tìm kiếm người bệnh của OpenMRS mặc định**



OpenMRS

Currently logged in as Super User | [Log out](#) | [My Profile](#) | [Help](#)

[Home](#) | [Find/Create Patient](#) | [Dictionary](#) | [Administration](#)

[Admin](#) | [Manage Encounters](#) | [Manage Encounter Types](#) | [Manage Encounter Roles](#)

### Encounter

[Add Encounter](#)

#### Find Encounter

Find by Encounter Id, Patient Identifier or name:  ☐ Include Deleted


Viewing results for 'Thanh' - 1 page

Patient Name	Encounter Type	Form	Provider	Location	Encounter Date
Thanh Viet Nguyen	ADULTRETURN		Super User	Unknown Location	09/04/16
Thanh Viet Nguyen	ADULTINITIAL				09/05/16
Thanh Viet Nguyen	PEDSINITIAL				09/05/16

Showing 1 to 3 of 3 entries

Show  entries

**Hình 3.22. Giao diện tìm kiếm hồ sơ y tế của OpenMRS mặc định**



OpenMRS

Currently logged in as Super User | [Log out](#) | [My Profile](#) | [Help](#)

[Home](#) | [Find/Create Patient](#) | [Dictionary](#) | [Administration](#)

[Admin](#) | [Manage Observations](#)

### Observation

Person\*:

Encounter:  [View/Edit](#)

Order:

Location:

Observation Date\*:  (Format: dd/mm/yyyy)

Question Concept\*:

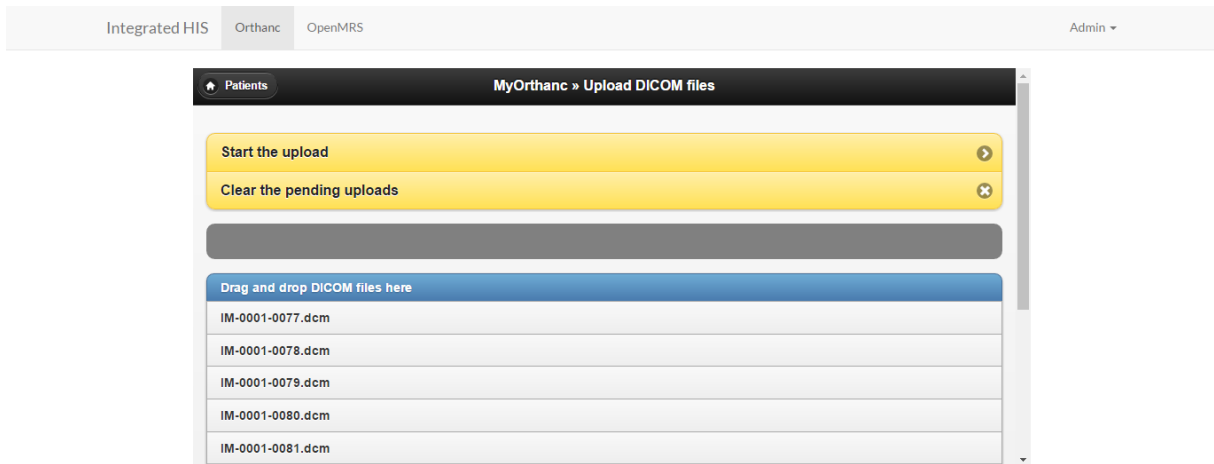
Comment:

**Hình 3.23. Chức năng thêm xét nghiệm của OpenMRS mặc định**

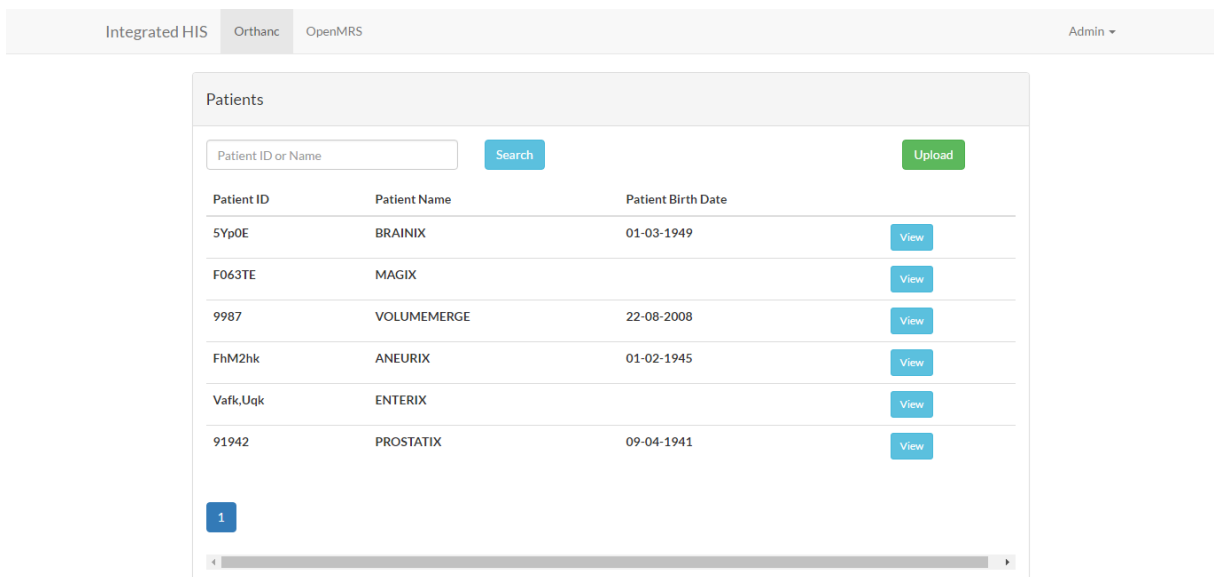
Ví dụ như chức năng thêm xét nghiệm trên đây, giao diện của OpenMRS mặc định buộc người dùng phải nhớ ID của loại xét nghiệm tương ứng mới có thể tìm và thêm vào hồ sơ, như vậy quả thật bất tiện và không hợp lý.

### 3.5.3.4. Tích hợp phần mềm quản lý DICOM Orthanc

Tương tự như quản lý hồ sơ y tế, Orthanc cung cấp các chứng năng quản lý tệp DICOM thông qua RESTFUL API, giao diện được xây dựng nhằm tạo sự tiện lợi cho việc tải lên và trình diễn dữ liệu hình ảnh trong tệp DICOM.



*Hình 3.24. Giao diện tải lên tệp DICOM*



*Hình 3.25. Giao diện chính của dịch vụ quản lý DICOM*

Integrated HIS
Orthanc
OpenMRS
Admin

Patients > BRAINIX

Patient Name	Patient ID	Patient Birthday
BRAINIX	5Yp0E	01-03-1949

Patient series

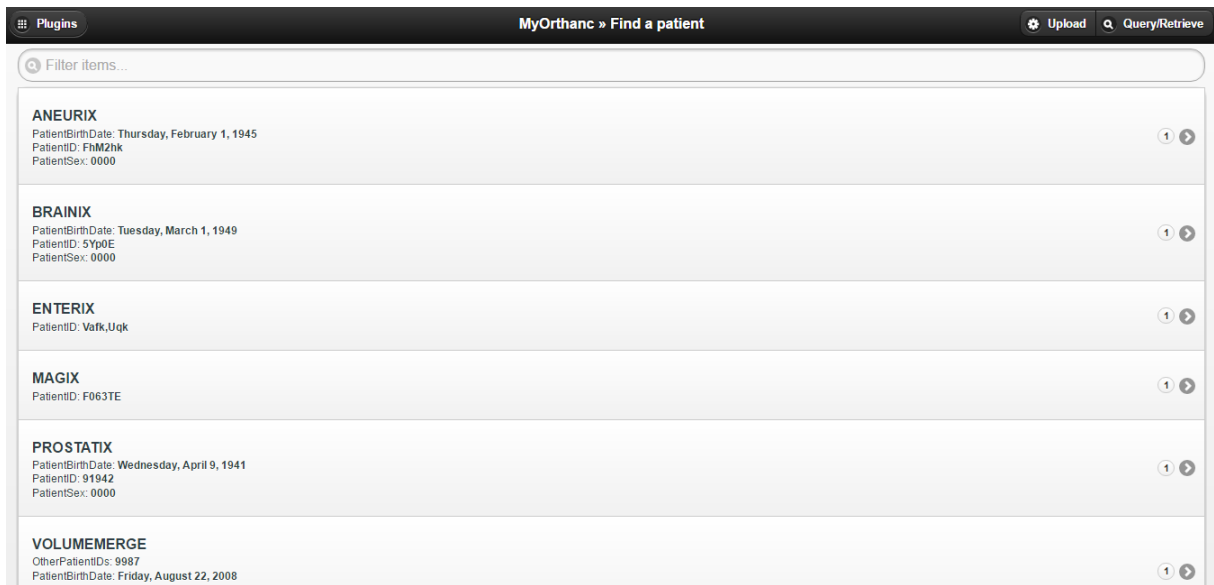
Description	Manufacturer	Modality	Protocol name	Date	
IRM cérébrale, neuro-crâne	Philips Medical Systems	MR	SOUS	01-12-2006	<a href="#">View</a>
IRM cérébrale, neuro-crâne	Philips Medical Systems	MR	T2W/FE-EPI SENSE	01-12-2006	<a href="#">View</a>
IRM cérébrale, neuro-crâne	Philips Medical Systems	MR	sT2/TSE/T SENSE	01-12-2006	<a href="#">View</a>

**Hình 3.26. Giao diện trình diễn dữ liệu hình ảnh của DICOM**

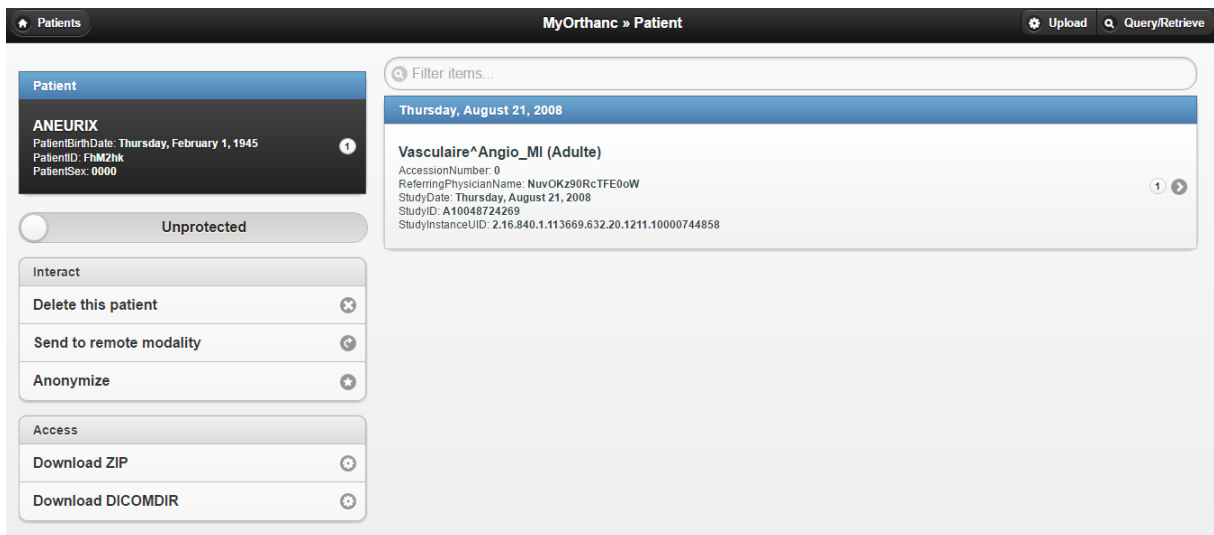
Hình trên mô tả giao diện trình diễn hình ảnh mà một tập các tệp DICOM mang lại, sử dụng con lăn chuột để đi qua các lát cắt.

### So sánh với giao diện mặc định

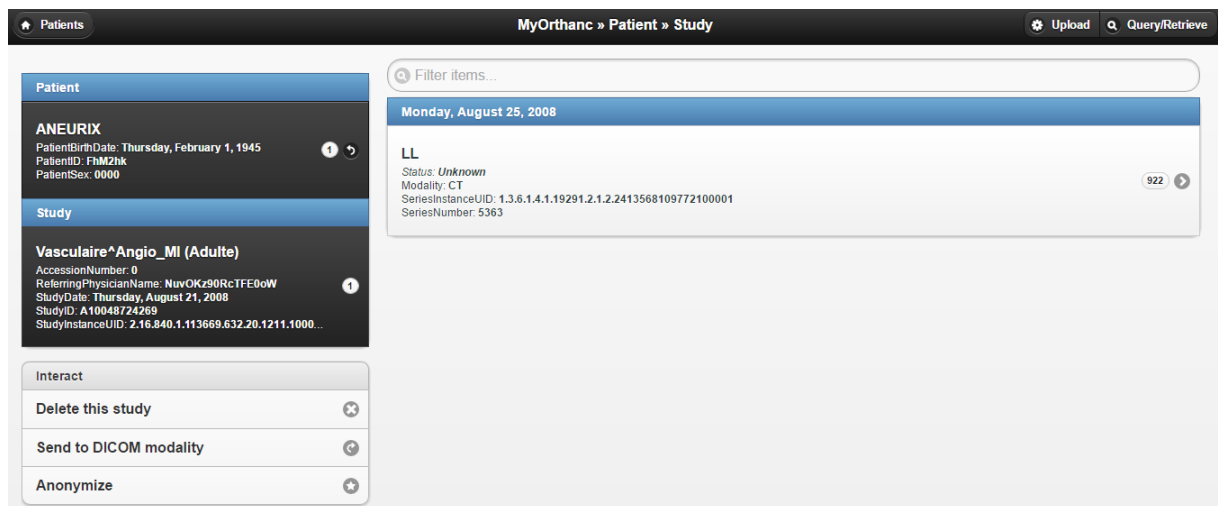
Tương tự OpenMRS, Orthanc cũng đi kèm một giao diện Web để chúng ta có thể sử dụng các chức năng của nó trực tiếp. Và cũng như OpenMRS giao diện mặc định của Orthanc thực sự không ấn tượng và giúp người sử dụng cảm thấy thoải mái.



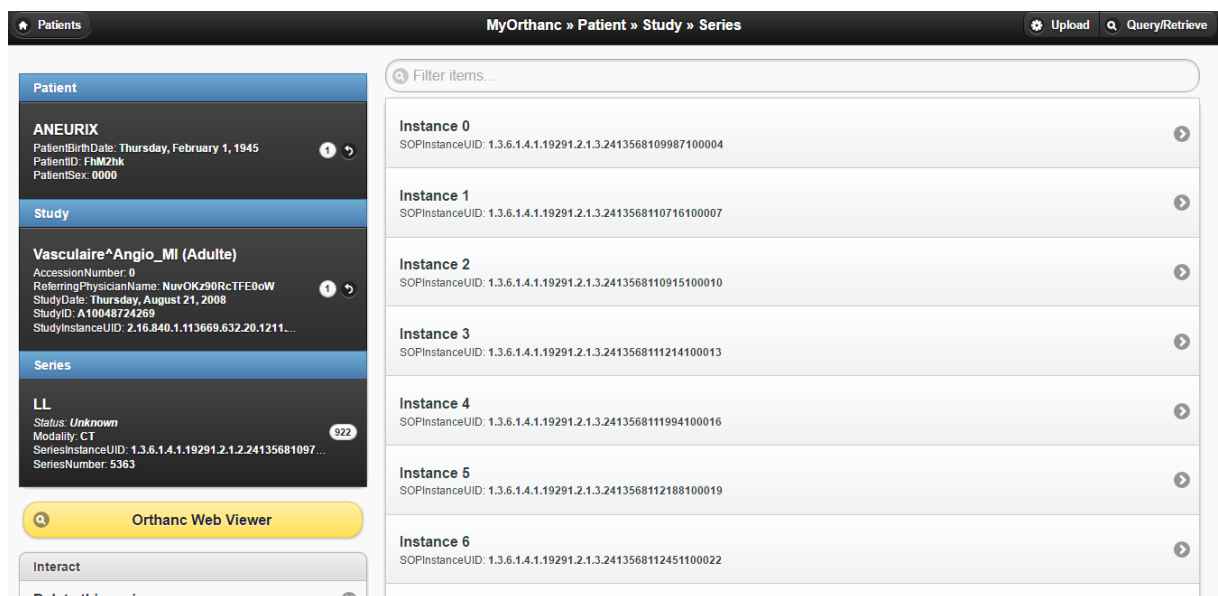
**Hình 3.27. Giao diện trang chủ của Orthanc, liệt kê danh sách DICOM**



**Hình 2.28. Trang thông tin về người bệnh**



*Hình 2.29. Trang thông tin Study*



*Hình 2.30. Trang thông tin Series*



***Hình 2.31. Chức năng Xem hình ảnh của Series DICOM***

Với mục đích là quan sát được dữ liệu hình ảnh DICOM của người bệnh, thiết kế của hệ thống tích hợp có ưu điểm truy cập nhanh chóng và dễ dàng hơn so với giao diện mặc định của DICOM, chúng ta cũng không cần phải chuyển quá nhiều màn hình để có thể quan sát được những hình ảnh này.

## **CHƯƠNG 4. KẾT LUẬN, ĐỊNH HƯỚNG NGHIÊN CỨU**

### **4.1. Các kết quả đạt được**

Qua quá trình nghiên cứu, tìm hiểu và xây dựng một hệ thống thông tin y tế tích hợp, tôi đã bổ sung cho mình nhiều kiến thức, kỹ năng về tích hợp hệ thống, nghiên cứu sử dụng REST API của các phần mềm nguồn mở. Từ đó, mục tiêu cũng như toàn bộ nội dung khoá luận đề ra đã được hoàn thành. Cụ thể, những đóng góp của khoá luận bao gồm:

- Nghiên cứu, tìm hiểu tổng quan về Tích hợp hệ thống, một số phương pháp tích hợp hệ thống từ nhiều thành phần khác nhau như tích hợp mức dữ liệu, mức chức năng và mức dịch vụ.
- Tìm hiểu, đánh giá về mô hình dịch vụ Web RESTFUL, từ đó ứng dụng trong bài toán tích hợp một số hệ thống cơ bản trong lĩnh vực Y tế.
- Xây dựng hệ thống tích hợp thử nghiệm trên nền tảng Laravel, tích hợp được ba dịch vụ cơ bản là Chikitsa để quản lý thông tin bệnh nhân, OpenMRS để quản lý hồ sơ y tế và Orthanc để quản lý hình ảnh chuẩn DICOM.

Ngoài ra, quá trình thực hiện khoá luận tốt nghiệp đã cho phép tôi nâng cao thêm kỹ năng tự học, tự nghiên cứu, tìm giải pháp phù hợp để giải quyết vấn đề đặt ra.

### **4.2. Định hướng phát triển trong tương lai**

Mặc dù đã cố gắng tìm hiểu, trau dồi kiến thức trong quá trình làm nhưng kết quả đạt được vẫn còn hạn chế, cần phải bổ sung và cập nhật trong tương lai. Sau đây là một số định hướng phát triển của khóa luận trong thời gian tới:

- Hiện tại với những phần mềm đã tích hợp chưa thể đáp ứng toàn bộ nhu cầu của một cơ sở y tế thực tế, bởi vậy cần phải bổ sung thêm những thành phần thiết yếu khác của hệ thống.
- Những phần mềm đang được lựa chọn chưa phải là tối ưu, ví dụ nhưng phần mềm Orthanc DICOM trình diễn dữ liệu hình ảnh còn nhiều sai sót và chưa đảm bảo chất lượng tốt nhất, cần nghiên cứu thay thế bằng những phần mềm khác có chất lượng cao hơn.



# TÀI LIỆU THAM KHẢO

## Tiếng Việt

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

## Tiếng Anh

- [2] Ray Harishshankar, SOA-Based Enterprise Integration

- [3] Carl Jones, Do more with SOA Integration: Best of Packt

## Internet

- [4] <https://www.ibm.com/developerworks/vn/library/ws-restful/ws-restful-pdf.pdf>

- [5] [https://en.wikipedia.org/wiki/Electronic\\_health\\_record](https://en.wikipedia.org/wiki/Electronic_health_record)

- [6] <http://www.osirix-viewer.com/datasets/>

- [7] <http://openmrs.org/>

- [8] <http://chikitsa.sanskrutitech.in/>

- [9] <http://www.orthanc-server.com/index.php>

- [10] <http://dicom.nema.org/dicom/geninfo/Brochure.pdf>

- [11] <https://github.com/anlutro/php-curl>