ĐẠI HỌC QUỐC GIA TP. HCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA CÔNG NGHỆ PHẦN MỀM

BÁO CÁO ĐỒ ÁN MÔN:

CHUYÊN ĐỀ J2EE

SE325.H21

Tên đề tài:

WEBSITE QUẢN LÝ ĐIỀU TRỊ BỆNH TẠI PHÒNG MẠCH

Giảng viên hướng dẫn: ThS. Nguyễn Trác Thức.

Họ và tên	MSSV
Trần Đình Phúc	13520636
Nguyễn Thị Nhơn	13520590
Đỗ Tiến Hậu	13520253
Nguyễn Thị Hằng	13520244

TP.HCM, ngày 01 tháng 06 năm 2017.

NHẬN XÉT CỦA GIẢNG VIÊN

Mục lục

1 Giới thiệu

1.1 Tổng quan về JEE

JEE – Java Enterprise Edition hay Java Phiên bản Doanh nghiệp là một nền tảng (platform) dành cho việc xây dựng những ứng dụng cấp doanh nghiệp. Java EE thường được gọi là Java 2 Platform (phiên bản 2) hay J2EE.

JEE là một nền tảng được sử dụng rộng rãi, chứa một tập hợp các công nghệ được phối hợp vào nhau, làm giảm đáng kể chi phí và độ phức tạp của việc phát triển, triển khai và quản lý các tầng làm việc, các ứng dụng máy chủ trung tâm.

JEE được xây dựng dựa trên nền tảng Java SE và cũng cấp thêm một tập các API (giao diện lập trình ứng dụng) để phát triển và hoạt động các ứng dụng phía máy chủ (Server-Side Applications) một cách mạnh mẽ, có khả năng mở rông, đáng tin cậy và bảo mật.

Một số thành phần cơ bản của JEE:

- Enterprise Java Beans (EJB): một thành phần kiến trúc của các ứng dụng server được quản lý, sử dụng để bao gói các business logic của các ứng dụng. Công nghệ EJB cho phép phát triển nhanh chóng và đơn giản hóa các ứng dụng phân tán, các giao dịch an toàn và di động dựa trên công nghệ Java.
- Java Persistence API (JPA): một framework cho phép nhà phát triển quản lý dữ liệu bằng cách sử dụng ánh xạ đối tượng quan hệ (Object Relational Mapping - ORM) trong các dứng dụng được xây dựng trên nền tảng Java.

1.2 Giới thiệu đề tài

1.2.1 Đặt vấn đề.

1.2.1.1 Hiện trạng

Trong bối cảnh các bệnh viện hiện nay đang quá tải do nhu cầu khám chữa bệnh của người dân tăng không ngừng, các phòng khám đa khoa được lập nên như một giải pháp tất yếu để giải quyết tình trạng đó. Các phòng khám này hoạt động theo mô hình của một bệnh viện nhưng với quy mô nhỏ hơn, đáp ứng yêu cầu khám và điều trị bệnh nhanh, thuận tiện cho mọi người. Và để có thể hoạt động một cách hiệu quả, tiết kiệm chi phí thì mọi phòng khám đều cần có một hệ thống phần mềm quản lý tự động thay cho việc quản lý thủ công bằng giấy tờ, sổ sách. Có thể xem đây là một nhu cầu lớn của thị trường với hàng ngàn phòng khám đa khoa trên cả nước.

Một hệ thống phần mềm quản lý cho phòng khám đa khoa cần phải đáp ứng các yêu cầu: hiệu quả, tiết kiệm chi phí, vận hành đơn giản, đầy đủ các chức năng cần thiết (quản lý khám và điều trị bệnh) và có thể mở rộng các chức năng trong tương lai.

Nắm bắt được hiện trạng và yêu cầu trên, nhóm đã lập kế hoạch xây dựng phần mềm quản lý khám chữa bệnh cho phòng khám đa khoa. Phần mềm sẽ có các chức năng của một phòng khám điển hình và có thể xây dựng thêm các chức năng tùy theo yêu cầu đặc thù của từng phòng khám.

1.2.1.2 Đánh giá hiện trạng

Những vấn đề còn tồn đọng trong việc quản lý khám và điều trị bệnh bằng phương pháp cũ:

- Mất nhiều thời gian lưu trữ, tìm kiếm và bảo quản hồ sơ bệnh nhân; tốn kém nhân lực.
- Các thông tin phức tạp, nếu lưu trên giấy sẽ rất tốn và khó truy hồi.
- Truy xuất dữ liệu, tìm kiếm lâu.
- Hiệu quả trong công việc kém.

1.2.2 Tính cần thiết và mục đích đề tài

Việc tiếp cận máy móc, thiết bị trị liệu cũng như máy tính đều được phổ cập rộng rãi đối với các bác sĩ, y tá để hỗ trợ khám và điều trị bệnh nhân. Đề tài này sẽ là một giải pháp cho các Cơ Sở Y Tế giải quyết được những vấn đề tồn đọng trong việc quản lý bằng phương pháp cũ.

1.2.3 Phương pháp nghiên cứu.

Nghiên cứu qui trình khám và điều trị bệnh, các loại hồ sơ, sổ sách cần có theo yêu cầu của việc quản lý. Từ đó áp dụng kiến thức về phân tích, thiết kế hướng đối tượng và công nghệ .NET xây dựng hệ thống website quản lý điều trị bệnh.

1.2.4 Giới hạn đề tài.

Đề tài nghiên cứu, thiết kế, xây dựng một hệ thống quản lý điều trị bệnh với mục đích quản lý thông tin, qui trình khám và điều trị bệnh của bệnh nhân. Sử dụng kiến thức phân tích , thiết kế hướng đối tượng và công nghệ .NET

1.3 Nghiên cứu sơ bộ

1.3.1 Mô tả nghiệp vụ

1.3.1.1 Mô tả website

Website quản lý điều trị bệnh là một chương trình hệ thống hóa việc quản lý thông tin cũng như quy trình khám và chữa bệnh của bệnh nhân. Theo đó, phần mềm sẽ quản lý công tác khám và điều trị bệnh.

Mục đích là hỗ trợ công tác quản lý hồ sơ bệnh nhân cũng như quy trình khám và điều trị bệnh tại phòng mạch nhằm dễ dàng trong công tác tìm kiếm, thống kê dữ liệu.

1.3.1.2 Phân tích yêu cầu.

Bên cạnh các yêu cầu chức năng, website đảm bảo yêu cầu phi chức năng, tổng quát hóa, chức năng gồm:

1.3.1.2.1 Yêu cầu chức năng.

1.3.1.2.1.1 Danh sách yêu cầu.

STT	Tên yêu cầu	Biểu mẫu	Quy định
1	Ghi nhận thông tin bệnh nhân	BM1: Thông tin bệnh nhân	QĐ1: Giới tính bao gồm: “Nam”, “Nữ”
2	Lập phiếu khám bệnh	BM2: Phiếu khám bệnh	QĐ1 QĐ2: Mã phiếu tạo tự động
3	Lập hóa đơn thanh toán	BM3: Hóa đơn	QĐ 2
4	Lập yêu cầu xét nghiệm	BM4: Phiếu yêu cầu xét nghiệm	QĐ 1 QĐ 3: Tuổi được tính theo ngày sinh của bệnh nhân QĐ 4: Thời gian thực hiện không qua 24h sau thời gian chỉ định
5	Lập phiếu khám chuyên khoa	BM5: Phiếu yêu cầu khám chuyên khoa	QĐ 1 QĐ 3

			QĐ 4
6	Kê đơn thuốc	BM6: Đơn thuốc	QĐ 1
7	Lập giấy nhập viện	BM7: Phiếu điều trị nội trú	QĐ 1
8	Lập giấy xác nhận phẫu thuật, mổ	BM8: Giấy xác nhận phẫu thuật/mổ	QĐ 1
9	Ghi nhận thông tin điều trị định kì	BM9: Hồ sơ điều trị nội trú	QĐ 1
10	Lập phiếu chuyển viện	BM10: Giấy chuyển viện (mẫu chung của bộ Y tế)	
11	Lập phiếu ra viện	BM11: Giấy ra viện	QĐ 5: Thời gian ra viện sau thời gian vào viện
12	Quản lý thông tin thuốc	BM12: Danh mục thuốc	
13	Thống kê, báo cáo định kì	BM13: Báo cáo hoạt động khám bệnh	

1.3.1.2.1.2 Yêu cầu chức năng và biểu mẫu kèm theo.

1.3.1.2.1.2.1 Yêu cầu tiếp nhận bệnh nhân.

THÔNG TIN BỆNH NHÂN

Họ tên: ……

Ngày sinh: …… Giới tính: …….

Nghề nghiệp: …….

Địa chỉ: ……..

Số điện thoại: ……

Lí do khám: ……

1.3.1.2.1.2.2 Lập phiếu khám bệnh.

PHIẾU KHÁM BỆNH

Mã phiếu: …….

Ngày khám: ……..

Họ tên bệnh nhân: ……

Tuổi: …… Giới tính: …….

Tiền sử bệnh: …….

Bác sĩ: ……

CHI TIẾT KHÁM BỆNH

STT	Yêu cầu khám	Khoa phụ trách

Chẩn đoán: ……...

Lời dặn: ……

1.3.1.2.1.2.3 Lập hóa đơn thanh toán.

HÓA ĐƠN

Mã hóa đơn: ……..

Mã phiếu khám bệnh: …….

Họ tên bệnh nhân: ……

Mã bệnh nhân: …….

Ngày khám: …….

CHI TIẾT HÓA ĐƠN

STT	Tên dịch vụ	Thành tiền
	Tổng:	

1.3.1.2.1.2.4 Ghi nhận tình trạng bệnh nhân.

HỒ SƠ ĐIỀU TRỊ NỘI TRÚ

Mã bệnh nhân: ……

Tên bệnh nhân: …….. ……

Tuổi: …….. Giới tính: ……

Chẩn đoán / Tình trạng bệnh: …….

Bác sĩ điều trị: …….

Y tá: ……..……..

Phòng: ……..

Ngày bắt đầu điều trị: ……

Ngày kết thúc điều trị: ……

Ngày	Tình trạng	Ghi chú

1.3.1.2.1.2.5 Lập yêu cầu xét nghiệm.

PHIẾU YÊU CẦU XÉT NGHIỆM

Tên bệnh nhân: ……

Tuổi: …… Giới tính: ……

Chẩn đoán: ……..

Bác sĩ chỉ định: ……

Thời gian chỉ định: ……..

Nơi chỉ định: ……..

Thời gian thực hiện: ……..

Bác sĩ xét nghiệm: ..……

1.3.1.2.1.2.6 Lập phiếu khám chuyên khoa.

PHIẾU YÊU CẦU KHÁM CHUYÊN KHOA

Tên bệnh nhân: …….

Tuổi: …….. Giới tính: …….

Địa chỉ: .....……

Bác sĩ chỉ định: ……

Thời gian chỉ định: ……..

Nơi chỉ định: ……

Thời gian thực hiện: …….

Bác sĩ thực hiện khám chuyên khoa: ……

1.3.1.2.1.2.7 Ghi kết quả khám bệnh.

PHIẾU KHÁM BỆNH

Mã phiếu: …….

Ngày khám: ……..

Họ tên bệnh nhân: ……

Tuổi: …… Giới tính: …….

Tiền sử bệnh: …….

Bác sĩ: ……

CHI TIẾT KHÁM BỆNH

STT	Yêu cầu khám	Khoa phụ trách

Chẩn đoán: ……...

Lời dặn: ……

1.3.1.2.1.2.8 Kê đơn thuốc.

ĐƠN THUỐC

Mã đơn thuốc: ……..

Mã phiếu khám bệnh: ……..

Họ tên bệnh nhân: ……

Mã bệnh nhân: ……

Ngày khám: ……

Chẩn đoán: ……..

STT	Mã thuốc	Tên thuốc	Đơn vị tính	Số lượng	Đơn giá	Thành tiền
					Tổng:	

1.3.1.2.1.2.9 Lập giấy nhập viện,

PHIẾU ĐIỀU TRỊ NỘI TRÚ

Tên bệnh nhân: ……..

Tuổi: ……. Giới tính: ……

Chẩn đoán/Tình trạng: ……

Thời gian nhập viện: ……..

Bác sĩ điều trị: …….

Người tiếp nhận: …….

Ghi chú: …….

1.3.1.2.1.2.10 Yêu cầu khám chuyên khoa

PHIẾU YÊU CẦU KHÁM CHUYÊN KHOA

Tên bệnh nhân: …….

Tuổi: …….. Giới tính: …….

Địa chỉ: .....……

Bác sĩ chỉ định: ……

Thời gian chỉ định: ……..

Nơi chỉ định: ……

Thời gian thực hiện: …….

Bác sĩ thực hiện khám chuyên khoa: ……

1.3.1.2.1.2.11 Yêu cầu xét nghiệm.

PHIẾU YÊU CẦU XÉT NGHIỆM

Tên bệnh nhân: ……

Tuổi: ……. Giới tính: ……

Chẩn đoán: ……..

Bác sĩ chỉ định: ……

Thời gian chỉ định: ……..……..

Nơi chỉ định: ……..

Thời gian thực hiện: ……..…….

Bác sĩ xét nghiệm: ..……

1.3.1.2.1.2.12 Lập giấy xác nhận phẫu thuật, mổ.

GIẤY XÁC NHẬN PHẪU THUẬT/MỔ

Tên bệnh nhân: ……

Tuổi: …… Giới tính: ……

Địa chỉ: ……..

Loại phẫu thuật/mổ: ……..

Bác sĩ chỉ định: ……

Thời gian thực hiện: ……..

Xác nhận của bệnh nhân/người thân

(Kí, ghi rõ họ tên)

1.3.1.2.1.2.13 Ghi nhận thông tin điều trị định kì (ngày, tuần).

HỒ SƠ ĐIỀU TRỊ NỘI TRÚ

Mã bệnh nhân: ……

Tên bệnh nhân: …….. ……

Tuổi: …….. Giới tính: ……

Chẩn đoán / Tình trạng bệnh: …….

Bác sĩ điều trị: …….

Y tá: ……..……..

Phòng: ……..

Ngày bắt đầu điều trị: ……

Ngày kết thúc điều trị: ……

Ngày	Tình trạng	Ghi chú

1.3.1.2.1.2.14 Lập phiếu chuyển viện.

SỞ Y TẾ …….. **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

BỆNH VIỆN …… **Độc lập – Tự do – Hạnh phúc**

GIẤY CHUYỂN VIỆN

Kính gửi: …….

Bệnh viện chúng tôi trân trọng giới thiệu:

Họ và tên người bệnh: …… Tuổi: ……..... Nam, nữ: ……..

Dân tộc:…….Ngoại kiều:……...

Nghề nghiệp:……Nơi làm việc……..

BHYT giá trị từ ngày:…/…./…. Đến …./ ../… . Số:

Địa chỉ:……

Đã được điều trị/ khám bệnh tại:…….

Từ ngày: ……../……/……. đến ngày: ……../ ……/ ……

TÓM TẮT BỆNH ÁN

Dấu hiệu lâm sàng: …….

……
……

Các xét nghiệm: …….....

……..

……..

Chẩn đoán: ……..

…….

……

Thuốc đã dùng:…… …….

Tình trạng người bệnh lúc chuyển viện:…… …….

Lý do chuyển viện:…….

Chuyển hồi: …. giờ……phút, ngày ……..tháng……năm…….

Phương tiện vận chuyển:……

Họ, tên chức danh người đưa đi: ……

Ngày…… tháng…… năm 20….

BÁC SĨ ĐIỀU TRỊ

Họ tên:……

GIÁM ĐỐC BỆNH VIỆN

Họ tên:

……

1.3.1.2.1.2.15 Lập phiếu xuất viện.

GIẤY RA VIỆN

Họ tên bệnh nhân: …….

Tuổi: …… Giới tính: …….

Địa chỉ: …….

Thời gian vào viện: ……..

Thời gian ra viện: ……

Chẩn đoán: ……

Lời dặn: ……

Ngày …… tháng ……..năm……

BÁC SĨ ĐIỀU TRỊ

1.3.1.2.1.2.16 Lưu trữ thuốc.

STT	Mã thuốc	Tên thuốc	Số lượng	Đơn giá	Số đăng kí	Số lô	Hạn sử dụng

1.3.1.2.1.2.17 Thống kê định kì

BÁO CÁO HOẠT ĐỘNG KHÁM BỆNH

Từ …… đến ……..

1. Hoạt động khám bệnh

Tổng số bệnh nhân khám bệnh:

Vào viên:

Khám theo yêu cầu:

2. Bệnh nhân nhập viện

Tên khoa	Tổng số

3. Bệnh nhân khám theo yêu cầu

Tổng số:

Nội:

Sản:

Nhi:

……. (giờ), ngày……tháng……năm

NGƯỜI LẬP PHIẾU **TRƯỞNG KHOA**

1.3.1.2.2 Yêu cầu phi chức năng.

1.3.1.2.2.1 Yêu cầu bảo mật.

Đăng nhập: Quản trị hệ thống, tài khoản thuộc các nhóm người dùng: Nhân viên tiếp nhận, Bác sĩ đa khoa, Bác sĩ chuyên khoa, Dược sĩ, Y tá.

1.3.1.2.2.2 Yêu cầu phân quyền.

Các bác sĩ, y tá, dược sĩ, nhân viên: Đăng nhập, cập nhật thông tin cá nhân, đổi mật khẩu, tìm kiếm thông tin.

Quản trị hệ thống: Có toàn quyền truy cập vào hệ thống và quản lý tất cả tài khoản đăng nhập hệ thống.

User	**Quyền**
Bác sĩ, y tá, dược sĩ, nhân viên tiếp nhận.	Đăng nhập.
	Đăng xuất.
	Lấy lại mật khẩu.
	Cập nhật thông tin cá nhân (mật khẩu, địa chỉ liên hệ…).
	Tìm kiếm thông tin bệnh nhân.
Admin	Có toàn quyền truy cập vào hệ thống.
	Quản lý tài khoản.

2 Công nghệ

2.1 Maven

2.1.1 Tổng quan về Maven

Maven là một từ trong tiếng Yiddish, nghĩa là bộ đệm kiến thức. Nó là một phần mềm. Ban đầu nó được phát triển để đơn giản hóa các quá trình xử lý trong dự án Jakarta Turbine. Một số dự án phát triển ở một vài nơi khác nhau nên hệ thống file có sự khác nhau.

Điều này làm cho các nhà phát triển rất là khó khăn. Vì vậy, Maven đã được phát triển như là một cái chuẩn để xây dựng các dự án, hệ thống sẽ phát triển cùng một chuẩn sẽ dễ dàng thay đổi cũng như chia sẻ một cách đồng bộ.

Bây giờ Maven được sử dụng để xây dựng và quản lý bất kỳ dự án Java-base nào. Nhà phát triển Maven hy vọng rằng nó sẽ giúp các Java developer dễ dàng hơn trong việc xây dựng cũng như quản lý các dự án Java.

Apache Maven là chương trình quản lý dự án được sử dụng nhiều bởi Java developer, maven thực hiện các công việc như khởi tạo project, biên dịch, đóng gói, chạy test… một cách tự động và nhanh chóng. Nó đặc biệt hữu dụng với các dự án tầm trung và lớn.

- Nó thường được so sánh với **Apache Ant**, nhưng nó hoạt động khác hẳn với **Apcache Ant**.
- Mỗi dự án sẽ được mô tả trong một file có định dạng XML có tên "pom.xml". File này sẽ chứa một số thông tin như tên dự án, các thư mục tài nguyên liên quan, các libraries sử dụng… Khi bắt đầu biên dịch, trình **Maven** sẽ đọc file này và tự động tải các libraries cần thiết từ repositories của nó thông qua mạng hoặc nơi người dùng đã định nghĩa.

Mục tiêu chính của Maven đó là cho phép một developer hiểu được toàn bộ hệ thống trong thời gian ngắn nhất, để được như vậy, Maven phải thành công trong các lĩnh vực sau:

- **Làm cho việc xây dựng hệ thống một cách dễ dàng hơn**
  Khi sử dụng Maven, người dùng không cần phải biết bộ máy phía dưới, Maven che chắn rất tốt các chi tiết cụ thể của hệ thống.
- **Cung cấp cho người dùng một hệ thống xây dựng thống nhất**
  Maven cung cấp rất nhiều thông tin hữu ích mà một phần là được trích từ file POM của java project mà bạn đã build.
- **Cung cấp thông tin dự án một cách chuyên nghiệp**
- **Đưa ra hướng dẫn cụ thể nhất cho việc phát triển thực nghiệm**
  Maven giữ source code thử nghiệm của người dùng trong một nơi riêng

Maven cũng nhằm mục đích hỗ trợ trong công viện quản lý dự án, phát hiện và theo dõi vấn đề.
Hướng dẫn về cách bố trí cây thư mục của dự án để người dùng tìm hiểu có thể dễ dàng hơn.

- **Cho phép tự động update những tính năng mới**
  Maven cung cấp cho người dùng một cách đơn giản để cập nhật những bản update mới mà Maven đã làm.
  Hy vọng một phần kiến thức trên sẽ giúp các bạn một phần trong quá trình tìm hiểu thêm về Maven.

2.1.2 Cách sử dụng Maven

Maven là một phần mềm công cụ, xuất bản bởi Apache Software Foundation, được sử dụng để xây dựng và quản lý các dự án một cách tự động, dựa trên một khái niệm là POM (viết tắt của Project Object Model).

POM là một đơn vị nền tảng của Maven, đó là một file XML chứa đựng các thông tin và cấu hình của dự án. Những thông tin và cấu hình này sẽ được Maven sử dụng để xây dựng nên cấu trúc của dự án.

Maven được sử dụng chính trong Java, nhưng bên cạnh đó còn hỗ trợ cho các ngôn ngữ khác như PHP, C#, Ruby, Scala…

Mục đích chính của Maven là giúp cho các nhà phát triển phần mềm có thể triển khai dự án hoàn thành trong thời gian ngắn nhất và có hiệu quả. Maven có thể quản lý việc xây dựng dự án, báo cáo và tài liệu hóa thông tin dự án.

Ngoài chức năng xây dựng và quản lý thông tin các dự án, Maven còn cho phép tự động download các thư viện và các plug-ins từ một hay nhiều nơi từ trên mạng về. Các nơi này được gọi là Maven remote repositories (các kho chứa). Mặc định thì Maven sẽ lấy thư viện từ remote repository của Maven tại địa chỉ http://repo1.maven.org/maven2.

Các thư viện này sau khi được lấy về từ trên mạng sẽ được lưu trữ tại máy tính để sử dụng cho những lần sau. Về nguyên tắc, khi chạy, Maven sẽ tìm các thư viện ở kho chứa local trong máy tính trước, nếu không có thì sẽ lấy từ trên mạng. Vì vậy, chỉ có lần đầu tiên build chương trình hoặc khi dự án cần sử dụng thư viện mới thì máy tính phải nối kết nối với mạng Internet, còn những lần tiếp sau đó thì có thể chạy offline.

2.2 Spring Framework

2.2.1 Giới thiệu

Spring là một framework mã nguồn mở hỗ trợ phát triển phần mềm bằng Java EE. Những tính năng cốt lõi của Spring có thể được sử dụng để phát triển bất cứ ứng dụng Java nào, ngoài Spring cũng có những phần mở rộng để xây dựng ứng dụng web trên nền Java EE. Mục tiêu của Spring Framework là đơn giản hóa công việc phát triển ứng dựng Java EE bằng cách sử dụng các đối tượng Java đơn giản hay POJO (Plain Old Java Object).

2.2.2 Những lợi ích khi sử dụng Spring Framework

- Sử dụng POJO giúp đơn giản hóa việc phát triển ứng dụng
- Quản lý theo dạng module, nhà phát triển chỉ cần sử dụng những module phù hợp với ứng dụng của mình mà không cần quan tâm đến những thành phần khác của framework
- Nới lỏng ràng buộc giữa các thành phần thông qua việc sử dụng Dependency Injection và viết các interface
- Framework web của Spring được thiết kế theo mô hình MVC giúp xây dựng các ứng dụng web MVC thay thế cho các framework khác nhưng Struts.

2.2.3 Dependency Injection

Dependency Injection (DI) là một công nghệ nổi bật trong Srping Framework. Đây là một cách thực thi Inversion of Control.(IoC). IoC là một design pattern có thể được thực thi theo nhiều cách khác nhau như Delegate, Event… và DI là một trong những cách thực thi đó

Khi xây dựng những ứng dụng phức tạp, các lớp nên độc lập với nhau nhất có thể để tăng khả năng tái sử dụng các lớp và khả năng kiểm thử độc lập giữa chúng (trong unit testin). Lớp A phụ thuộc vào lớp B khi lớp A có thuộc tính có kiểu dữ liệu B.

```
public class Student{

	private Address add;

	public Student() {

		add = new Address();

	}

}
```

Để tránh sự phụ thuộc này, lớp Address sẽ được tiêm (inject) lớp Student. Có hai cách để thực hiện DI là sử dụng constructor của lớp Student (constructor injection) và sử dụng setter (setter injection)

- Construct injection

```
public class Student{

	private Address address;

	public Student(Address add) {

		this.address = add;

	}

}
```

- Setter Injection

```
public class Student{

	private Address address;

	public Student() {

	}

	public void setAddress(Address add){

		this.address = add;

	}

}
```

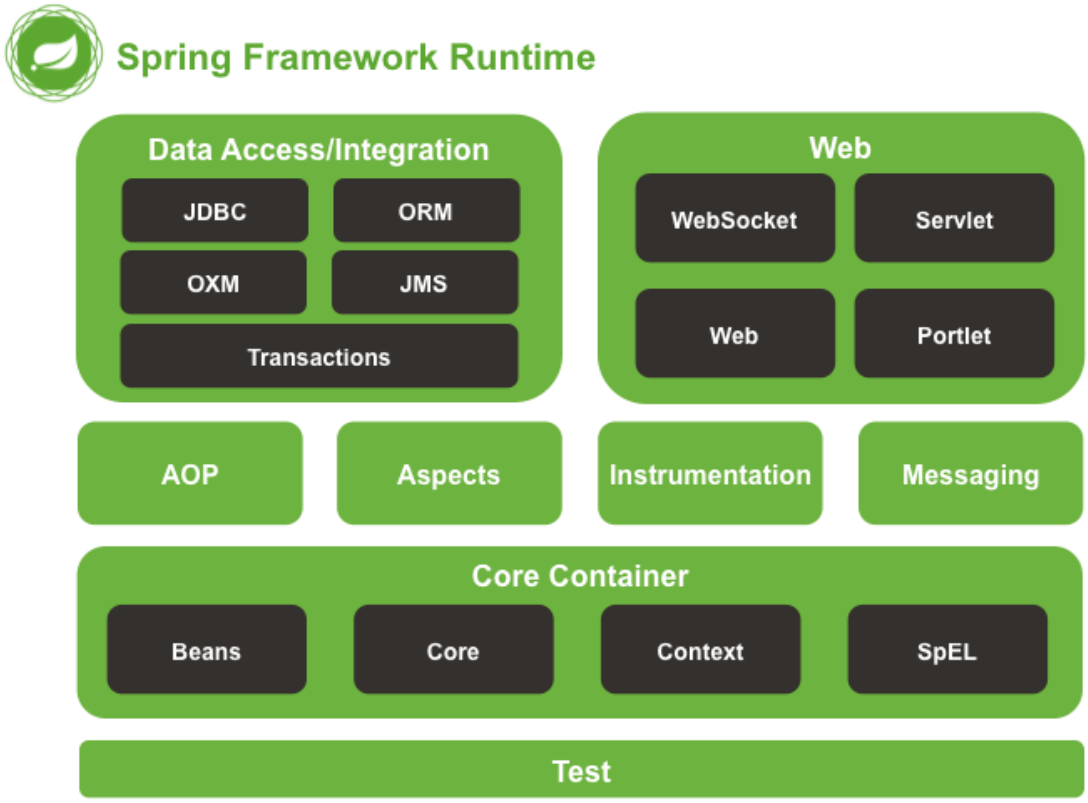
2.2.4 Aspect Oriented Programming

Aspect Oriented Programming (OOP) – lập trình hướng khía cạnh là một phương pháp lập trình hướng đến mục tiêu tăng tính modul bằng các phân tách các cross-cutting concern (những khía cạnh hay chức năng có ảnh hưởng đến toàn hệ thống , cần thực thi xuyên suốt trong hệ thống). AOP được thực hiện bằng cách bổ sung những hành vi vào mã nguồn có sẵn mà không phải sửa đổi nó, thay vì xác định cụ thể mã nào được sửa đổi thông qua một đặc tả "điểm cắt" (pointcut). Điều này cho phép những hành vi không phải là trung tâm của logic nghiệp vụ (ví dụ như logging) được thêm vào chương trình mà không làm rối lõi của mã đến các chức năng.

Module AOP của Spring Framework cung cấp một cách thực thi phương pháp lập trình hướng khía cạnh bằng cách định nghĩa những điểm chặn. Ví dụ có thể thêm vào các hàm bổ sung trước hoặc sau khi một phương thức được thực thi.

2.2.5 Một số module trong Spring Framework

Spring framework có nhiều tính năng khác, các tính năng này được phân thành các module thuận tiện cho việc quản lý và cài đặt. Các module này có thể được nhóm lại với nhau dựa trên những đặc điểm cơ bản của chúng. Các nhóm này bao gồm Core Container, Data Access/Integration, Web, AOP, Instrumentation và Test.



Lược đồ mối quan hệ giữa các module trong Spring Framework

Tài liệu này sẽ giới thiệu về một số module chính, cần thiết cho việc xây dựng một ứng dụng Web Java.

2.2.5.1 Spring Core

Là phần cơ bản nhất của framework, cung cấp những đặc tính IoC và Dependency Injection. Khái niệm cơ bản là BeanFactory - cài đặt factory pattern cho phép bạn móc nối sự phụ thuộc giữa các đối tượng trong file cấu hình.

IoC là một nguyên tắc trừu tượng mô tả một khía cạnh của một số mẫu thiết kế kiến trúc phần mềm đối lập với lập trình thủ tục. IoC là một phương pháp xây dựng phần mềm mà ở đó mã code có thể được dùng lại chung để quản lý việc thực thi các vấn đề cụ thể. Nó có ý nghĩa là nhiều module mã code có thể được phát triển độc lập với nhau sau đó được tái sủ dụng tích hợp trong một ứng dụng duy nhất

IoC thường được sử dụng cho các mục đích sau:

- Mỗi hệ thống có thể tập trung xung quanh những gì mà nó được thiết kế
- Các hệ thống được giả định về các hệ thống khác hoặc phải làm những gì nên làm
- Thay đổi một mắc xích mà không làm ảnh hưởng đến các mắc xích khác

Dependency Injection là các đối tượng định nghĩa sự phụ thuộc của chúng thông qua tham số của phương thức khởi tạo (constructor) hoặc các thuộc tính được thiết lập trên thể hiện (instance) sau khi nó được khởi tạo hoặc trả về từ phương thức factory.

2.2.5.2 Spring Context

Context package cung cấp cách để truy cập đối tượng. Context package kết thừa các đặc tính từ bean package và thêm vào chức năng đa ngôn ngữ, truyền sự kiện, resource-loading,... Thành phần chính của Spring Context là AppicationContext. Một bean factory có thể sử dụng tốt trong những ứng dụng đơn giản, nhưng để tận dụng lợi thế của tất cả sức mạnh của Spring, bạn sẽ có thể load các beans trong ứng dụng của bạn sử dụng nhiều container nâng cao của Spring: Application context.

Nhìn bên ngoài, AppicationContext thì giống với BeanFactory. Cả hai đều load các beans, kết nối và phân phối chúng theo các yêu cầu. Nhưng ApplicationContext còn có:

- Application context cung cấp một cách thức để giải quyết các thông điệp text (text messages), bao gồm hỗ trợ việc quốc tế hóa cho các thông điệp đó.
- Application context cung cấp một cách tổng quát để load các file tài nguyên, chẳng hạn file ảnh.
- Application context có thể xuất bản các sự kiện sang các beans.

Vì các tính năng nổi bật trên mà ApplicationContext được ưa chuộng hơn BeanFactory trong các ứng dụng gần đây.

Một số ApplicationContext phổ biến:

- ClassPathXmlApplicationContext: Load một định nghĩa context từ một file xml trong class path, xử lý các file định nghĩa context như là các class path resources.
- FileSystemXmlApplicationContext: Load một định nghĩa context từ một file xml trong hệ thống.
- XmlWebApplicaitonContext: Load một định nghĩa context từ một file xml trong ứng dụng web.

2.2.5.3 Spring DAO

DAO package cung cấp cho tầng JDBC, bỏ bớt những coding dài dòng của JDBC và chuyển đổi mã lỗi được xác định bởi database vendor. JDBC package cung cấp cách lập trình tốt như declarative transaction management, không chỉ cho các lớp cài đặt các giao tiếp đặc biệt mà còn cho tất cả POJO (plain old Java objects).

Tầng JDBC và DAO đưa ra một cây phân cấp exception để quản lý kết nối đến database, điều khiển exception và thông báo lỗi được ném bởi vendor của database. Tầng exception đơn giản điều khiển lỗi và giảm khối lượng code mà chúng ta cần viết như mở và đóng kết nối. Module này cũng cung cấp các dịch vụ quản lý giao dịch cho các đối tượng trong ứng dụng Spring.

2.2.5.4 Spring ORM

Spring Framework cung cấp tích hợp với Hibernate, JDO, Oracle TopLink, iBATIS SQL Maps và JPA: về quản lý tài nguyên, hỗ trợ thực hiện DAO. Ví dụ đối với Hibernate, có lớp đầu tiên hỗ trợ với nhiều tính năng tiện lợi IoC, giải quyết nhiều vấn đề điển hình của Hibernate mà việc tích hợp đưa ra. Tất cả các gói hỗ trợ cho các O / R (Object Relational) mappers tuân theo giao thức chung của Spring và sự phân cấp DAO exception. Thường có hai phong cách kết hợp: hoặc bằng cách sử dụng DAO của Spring "mẫu" hoặc mã DAO dựa theo Hibernate / JDO / TopLink / API ... Trong cả hai trường hợp, DAO có thể được cấu hình thông qua Dependency Injection và tham gia vào các nguồn tài nguyên của Spring và quản lý giao thức.

2.2.5.5 Spring AOP

Module tích hợp chức năng lập trình hướng khía cạnh vào Spring framework thông qua cấu hình của nó. Spring AOP module cung cấp các dịch vụ quản lý giao dịch cho các đối tượng trong bất kỳ ứng dụng nào sử dụng Spring. Với Spring AOP chúng ta có thể tích hợp declarative transaction management vào trong ứng dụng mà không cần dựa vào EJB component.

Spring AOP module cũng đưa lập trình metadata vào trong Spring. Sử dụng cái này chúng ta có thể thêm annotation vào source code để hướng dẫn Spring nơi và làm thế nào để liên hệ với aspect.

2.2.5.6 Spring Web

Cung cấp đặc tính của web như: chức năng file-upload, khởi tạo IoC container sử dụng trình lắng nghe serlvet và web-oriented application context. Package này để tích hợp với WebWork và Struts.

2.2.5.7 Spring MVC

Cung cấp mô hình MVC cho ứng dụng web. Spring MVC framework cung cấp sự phân biệt rõ ràng giữa domain model và web form - cho phép bạn sử dụng tất cả các đặc tính khác của Spring framework. Đây là module trọng tâm mà ta sẽ cùng tìm hiểu trong bài báo cáo này.

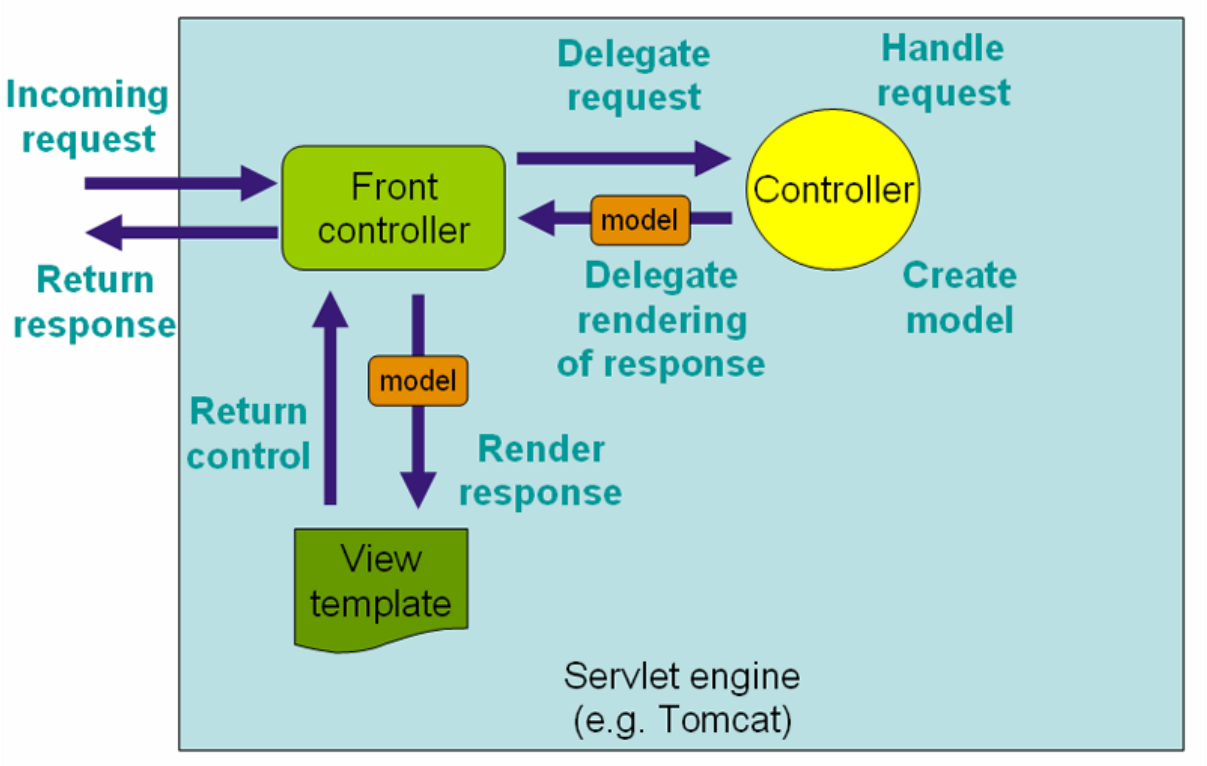
2.2.6 Module Spring MVC

2.2.6.1 Giới thiệu module Spring MVC

Đây là một module giúp cho việc xây dựng ứng dụng web trở nên chặt chẽ và linh động. Mẫu thiết kế Model-View-Controller giúp phân chia rạch ròi 3 công việc business logic, presentation logic, và navigation logic.

- **Model**: gồm các lớp java có nhiệm vụ:
    - Biểu diễn data và cho phép truy cập tới để get và set data trong này (JavaBean), Thường thì phần layer này mô phỏng 1 cách đầy đủ đối tượng từ thế giới thực.
    - Nhận các yêu cầu từ khung nhìn
    - Thi hành các yêu cầu đó (tính toán, kết nối CSDL …)
    - Trả về các giá trị tính toán cho View.
- **View**: bao gồm các mã tương tự như JSP để hiển thị form nhập liệu, các kết quả trả về từ Mô hình…
- **Controller**: Đồng bộ hoá giữa View và Model. Tức là với một trang JSP này thì sẽ tương ứng với lớp java nào để xử lý nó và ngược lại, kết quả sẽ trả về trang jsp nào.Nó đóng vai trò điều tiết giữa View và Model. Như vậy, chúng ta có thể tách biệt được các mã java ra khỏi mã html. Người thiết kế giao diện và người lập trình java có thể mang tính chất độc lập tương đối. Việc debug hay bảo trì sẽ dễ dàng hơn, việc thay đổi các theme của trang web cũng dễ dàng hơn …

2.2.6.2 Các thành phần chính của Spring MVC



Việc xử lý request và response trong Spring MVC Framework được mô tả như sau :

- **Bước 1 :**

  o DispatcherServlet nhận Request.

  o DispatcherServlet tra trong HandlerMapping và gọi Controller kết hợp với Request.

- **Bước 2 :**

  o Controller xử lý Request bằng cách gọi những phương thức dịch vụ thích hợp và sau đó trả về một đối tượng ModelAndView cho DispatcherServlet. Đối tượng ModelAndView này chứa dữ liệu trong đối tượng Model và tên của View.

- **Bước 3 :**

  o DispatcherServlet gửi tên của View đến cho một ViewResolver. ViewResolver sẽ tìm View thực sự cần dùng.

- **Bước 4 :**

  o DispatcherServlet truyền đối tượng Model đến cho View đã xác định để hiển thị kết quả.

- View lấy dữ liệu trong đối tượng Model và hiển thị kết quả cho người dùng.

2.2.6.2.1 DispatcherServlet

Tiếp nhận tất cả các request từ Browser.

Điều khiển luồng xử lý và trung chuyển giữa các thành phần (components) trong MVC. Hơn thế nữa, nó còn tích hợp các Spring IoC container, cho phép ta sử dụng đến tất cả các tính năng của Spring.
Trong hình minh họa trên, DispatcherServlet chính là Front controller. DispatcherServlet thực sự là một Servlet (nó thừa kế từ lớp cơ sở HttpServlet), và được khai báo trong tập tin **web.xml** của ứng dụng web. Các **requests** bạn muốn DispatcherServlet xử lý sẽ phải được ánh xạ bằng cách sử dụng một bộ ánh xạ URL trong cùng một tập tin **web.xml**. Dưới đây là ví dụ về khai báo và định nghĩa DispatcherServlet:

```
<web-app>
    <servlet>
        <servlet-name>example</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servle
t-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>example</servlet-name>
        <url-pattern>*.form</url-pattern>
    </servlet-mapping>
</web-app>
```

Như ta thấy, tất cả các **requests** có đuôi .form sẽ được DispatcherServlet 'example' xử lý.
Khi DispatcherServlet được khởi tạo, framework sẽ gọi file có tên **[tên-servlet]-servlet.xml** trong thư mục **WEB-INF**. Với ví dụ trên, framework sẽ gọi file **example-servlet.xml.**

2.2.6.2.2 Controllers

Controllers là các thành phần được Dispatcher Servlet gọi để xử lý các nghiệp vụ Business.Tất cả các Controller trong Spring đều implement từ org.springframework.web.servlet.mvc.Controller interface.

```
public interface Controller {
    /**
     * Process the request and return a ModelAndView object which
the DispatcherServlet
     * will render.
     */
    ModelAndView handleRequest(
        HttpServletRequest request,
        HttpServletResponse response) throws Exception;

}
```

Controller interface định nghĩa ra 1 phương thức chịu trách nhiệm xử lí 1 request và trả về model và view thích hợp. Ba khái niệm này chính là nền tảng của Spring MVC framework : Model – View – Controller.

Sau đây là các Controller sử dụng được trong Spring.

- SimpleFormController
- AbstractController
- AbstractCommandController
- CancellableFormController
- AbstractCommandController
- MultiActionController
- ParameterizableViewController
- ServletForwardingController
- ServletWrappingController
- UrlFilenameViewController

2.2.6.2.2.1 AbstractController

Khi sử dụng AbstractController làm baseclass cho các controller của bạn, chỉ cần override hàm **handleRequestInternal (HttpServletRequest, HttpServletResponse)**, viết code xử lý, và trả về một đối tượng ModelAndView. Sau đây là ví dụ:

```
package samples;

public class SampleController extends AbstractController {

    public ModelAndView handleRequestInternal(
        HttpServletRequest request,
        HttpServletResponse response) throws Exception {

        ModelAndView mav = new ModelAndView("hello");
        mav.addObject("message", "Hello World!");
        return mav;
    }
}
```

Ta định nghĩa controller này trong ApplicationContext.xml

```
<bean id="sampleController" class="samples.SampleController">
    <property name="cacheSeconds" value="120"/>
</bean>
```

Controller này sẽ buộc client cache dữ liệu cứ mỗi 2 phút trước khi kiểm tra lại. Đồng thời nó cũng trả về một view được code cứng.

2.2.6.2.2.2 SimpleFormController

Việc người dùng nhập các thông tin trong 1 form rồi submit là rất thông thường trong Web Applications. Simple Form Controller được dùng chủ yếu cho việc đó. Xét ví dụ sau :
Đây là thông tin cơ bản của 1 form.

```
  public class EmpInfo{
      private String empName;
      private int empAge;
      private double empSalary;
      // các hàm getter, setter
  }
```

Lưu ý: khi extend **Simple Form Controller** , hàm *doSubmitAction()* thường phải được override vì đây chính là hàm sẽ được gọi khi người dùng submit.

```
  public class EmpFormController extends SimpleFormController{

      public EmpFormController(){
          setCommandClass(EmpInfo.class);
```

```
        }

        public void doSubmitAction(Object command){
            EmpInfo info = (EmpInfo)command;
            process(info);
        }

        private void process(EmpInfo info){
            // do something
        }
    }
```

Giả sử như ta có 2 trang, 1 trang **empInfo.jsp** để người dùng nhập thông tin, 1 trang để thông báo **empSuccess.jsp** sau khi người dùng submit xong, thì ta cấu hình như sau.

```
<bean id = "empForm" class="EmpFormController">

        <property name="formView">
            <value>empInfo</value>
        </property>

        <property name="successView">
            <value>empSuccess</value>
        </property>


</bean>
```

2.2.6.2.3 Handler Mappings

Sử dụng handler mapping bạn có thể **ánh xạ từng requests đến các handler thích hợp.**

Các chức năng cơ bản của một HandlerMapping là đưa ra một HandlerExecutionChain (chuỗi hành động), và có thể chứa một số bộ chặn. Khi requests đến, các DispatcherServlet sẽ trao cho các handler mapping để nó kiểm tra và xuất ra HandlerExecutionChain thích hợp. Sau đó, các DispatcherServlet sẽ gọi handler và bộ chặn trong hệ thống (nếu có).

Phần này ta đề cập đến 2 handler mapping thông dụng nhất của Spring. Cả hai đều kế thừa AbstractHandlerMapping và chia sẻ các thuộc tính sau đây:

- interceptors: danh sách các bộ chặn được sử dụng.
- defaultHandler: bộ xử lý mặc định, được gọi khi handler mapping không trả ra kết quả phù hợp.

- order: Spring sẽ sắp xếp thứ tự tất cả các ánh xạ handler trong ngữ cảnh và gọi handler đầu tiên phù hợp.
- alwaysUseFullPath: true = đường dẫn đến handler được truyền đầy đủ.
- urlPathHelper: nên để default.
- urlDecode: giá trị mặc định cho thuộc tính này là false. HttpServletRequest trả về các request URL và URI không được giải mã. Nếu bạn không muốn chúng được giải mã trước khi HandlerMapping sử dụng chúng để tìm handler thích hợp, hãy set giá trị nó bằng true.
- lazyInitHandlers: cho phép các handler singleton khởi tạo với chế độ "lazy". Mặc định giá trị này là false.

(Lưu ý: trong bốn đặc tính cuối cùng chỉ có sẵn cho các lớp con của org.springframework.web.servlet.handler.AbstractUrlHandlerMapping).

2.2.6.2.3.1 BeanNameUrlHandlerMapping

Một handler mapping đơn giản nhưng rất mạnh mẽ là BeanNameUrlHandlerMapping, nó **ánh xạ các HTTP requests đến tên của các beans**, được định nghĩa trong bối cảnh ứng dụng web. Ví dụ chúng ta muốn cho phép một người dùng tạo một tài khoản, ở đây ta đã có một form controller thích hợp và một trang JSP để hiển thị. Khi sử dụng BeanNameUrlHandlerMapping, ta có thể ánh xạ HTTP requests với địa chỉ http://samples.com/editaccount.form vào form Controller như sau:

```
<beans>
  <bean id="handlerMapping"
class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMap
ping"/>

  <bean name="/editaccount.form"
class="org.springframework.web.servlet.mvc.SimpleFormController">
    <property name="formView" value="account"/>
    <property name="successView" value="account-created"/>
    <property name="commandName" value="account"/>
    <property name="commandClass" value="samples.Account"/>
  </bean>
<beans>
```

Tất cả requests gửi đến yêu cầu địa chỉ /editaccount.form bây giờ sẽ được xử lý bởi form Controller ở trên. Tất nhiên chúng ta phải định nghĩa một servlet-mapping trong **web.xml**, để tất cả các requests đều kết thúc bằng .form.

```
<web-app>
    ...
    <servlet>
        <servlet-name>sample</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
        <load-on-startup>1</load-on-startup>
    </servlet>

   <!-- maps the sample dispatcher to *.form -->
    <servlet-mapping>
        <servlet-name>sample</servlet-name>
        <url-pattern>*.form</url-pattern>
    </servlet-mapping>
    ...
</web-app>
```

2.2.6.2.3.2 SimpleHandlerMapping

```
<web-app>
    ...
    <servlet>
        <servlet-name>sample</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <!-- maps the sample dispatcher to *.form -->
    <servlet-mapping>
        <servlet-name>sample</servlet-name>
        <url-pattern>*.form</url-pattern>
    </servlet-mapping>

    <!-- maps the sample dispatcher to *.html -->
    <servlet-mapping>
        <servlet-name>sample</servlet-name>
        <url-pattern>*.html</url-pattern>
    </servlet-mapping>
    ...
</web-app>
```

Cấu hình web.xml trong đoạn code trên cho phép tất cả requests có kết thúc .form và .html đều được xử lý bởi ‘sample’ dispatcher servlet.

```
<beans>

    <!-- no 'id' required, HandlerMapping beans are automatically
detected by the DispatcherServlet -->
    <bean
class="org.springframework.web.servlet.handler.SimpleUrlHandlerMappi
ng">
        <property name="mappings">
            <value>
                /ex/view*.html=helpController
                /**/help.html=helpController
            </value>
        </property>
    </bean>

    <bean id="helpController"

class="org.springframework.web.servlet.mvc.UrlFilenameViewController
"/>

    <bean id="editAccountFormController"

class="org.springframework.web.servlet.mvc.SimpleFormController">
        <property name="formView" value="account"/>
        <property name="successView" value="account-created"/>
        <property name="commandName" value="Account"/>
        <property name="commandClass" value="samples.Account"/>
    </bean>
<beans>
```

Ánh xạ handler này sẽ chuyển requests yêu cầu ‘help.html’ trong bất cứ thư mục nào đến ‘helpController’. Requests yêu cầu resource bắt đầu với ‘view’ và kết thúc với ‘html’ trong thư mục ‘ex’ sẽ được chuyển đến ‘helpController’.

2.2.6.2.3.3 HandlerInterceptor interface

Cơ chế handler mapping của Spring có khái niệm xử lý đánh chặn, nó có thể cực kỳ hữu ích khi bạn muốn áp dụng các chức năng cụ thể cho một request xác định, ví dụ, kiểm tra điều kiện chẳng hạn.
Ví dụ dưới đây tạo ra một bộ chặn, nó chặn tất cả các requests và redirect người dùng đến một trang xác định nếu thời gian truy cập không nằm từ 9h sáng đến 6h tối.

```
<beans>
    <bean id="handlerMapping"

class="org.springframework.web.servlet.handler.SimpleUrlHandlerMappi
ng">
        <property name="interceptors">
            <list>
                <ref bean="officeHoursInterceptor"/>
            </list>
        </property>
        <property name="mappings">
            <value>
                /*.form=editAccountFormController
                /*.view=editAccountFormController
            </value>
        </property>
    </bean>

    <bean id="officeHoursInterceptor"
          class="samples.TimeBasedAccessInterceptor">
        <property name="openingTime" value="9"/>
        <property name="closingTime" value="18"/>
    </bean>
<beans>
```

```
package samples;

public class TimeBasedAccessInterceptor extends
HandlerInterceptorAdapter {

    private int openingTime;
    private int closingTime;

    public void setOpeningTime(int openingTime) {
        this.openingTime = openingTime;
    }

    public void setClosingTime(int closingTime) {
        this.closingTime = closingTime;
    }

    public boolean preHandle(
            HttpServletRequest request,
            HttpServletResponse response,
            Object handler) throws Exception {

        Calendar cal = Calendar.getInstance();
```

```
        int hour = cal.get(HOUR_OF_DAY);
        if (openingTime <= hour < closingTime) {
            return true;
        } else {

response.sendRedirect("http://host.com/outsideOfficeHours.html");
            return false;
        }
    }
}
```

2.2.6.2.4 ViewResolver

Như đã nói ở phần trên, tất cả controller trong Spring Web MVC framework trả về 1 đối tượng gọi là ModelAndView. Views trong Spring được gán cho 1 "view name " và được phân giải bởi 1 "view resolver".Sau đây là các "view resolver" trong Spring

- BeanNameViewResolver
- FreeMarkerViewResolver
- InternalResourceViewResolver
- JasperReportsViewResolver
- ResourceBundleViewResolver
- UrlBasedViewResolver
- VelocityLayoutViewResolver
- VelocityViewResolver
- XmlViewResolver
- XsltViewResolver

Ở đây ta sẽ đề cập đến 2 ViewResolver chính là "*InternalResourceViewResolver*" và "*BeanNameViewResolver*".

2.2.6.2.4.1 Internal Resource View Resolver

View Resolver này sẽ ánh xạ Logical name của Resource (ModelAndView object được trả về bởi Controller) đến một Physical View(myView.jsp,…). Ví dụ :

```
public class MyController {
         public void handle(){
             if(condition1()){
                 return new ModelAndView("myView1");
             }else if (condition2()){
                 return new ModelAndView("myView2");
             }
```

```
            return new ModelAndView("myView3");
    }
```

Giả sử như Request của người dùng thỏa ***condition1()***, thì view "**myView.jsp**"(trong thư mục " /WEB-INF") sẽ được hiển thị,nếu không thì hoặc "**myView2.jsp**" hoặc "**myView3.jsp**" sẽ được hiển thị.

Ngoài ra, ta còn phải cấu hình file web.xml :

```
<bean id="viewResolver"
class="org.springframework.web.servlet.view.
InternalResourceViewResolver">
    <property name="prefix"><value>/WEB-INF/</value></property>
    <property name="suffix"><value>.jsp</value></property>
</bean>
```

Khi logical View name là myView1, thì view thực sự sẽ là : **prefix + logical View Name + the suffix** , tức là: **/WEB-INF/myView.jsp**

2.2.6.2.4.2 BeanNameViewResolver

Resolver này có nhiệm vụ ánh xạ một "view name" vào một bean trong ApplicationContext.

2.2.7 Ưu, nhược điểm của Spring MVC

2.2.7.1 Ưu điểm

- Spring là một framework Java mạnh mẽ được sử dụng trong những ứng dụng Java có phạm vi lớn. Nó cung cấp những dịch vụ Enterprise đến những Plain Old Java Objects (POJOs). Cơ chế IoC giúp ứng dụng đạt được sự đơn giản hoá và tăng khả năng kiểm tra lỗi.
- Spring MVC cung cấp một sự phân chia rất rõ ràng, rành mạch giữa những Controller, Java Bean models và Views.
- Spring MVC rất linh hoạt, toàn bộ Spring MVC được xây dựng dựa trên những interfaces. Mọi phần của Spring MVC framework được cấu hình thông qua việc lắp ghép những interface, class tiện ích sẵn có, và thậm chí được tạo bởi người dùng.
- Spring không chỉ sử dụng công nghệ JSP mà còn có thể dễ dàng tích hợp các công nghệ view khác như Velocity, XSLT, FreeMarker, XL, …
- Cung cấp cơ chế che dấu nền công nghệ bên dưới, trang web khi hiển thị chỉ có extension là .htm, không thể biết được bên dưới ta sử dụng công nghệ, kỹ thuật gì, JSP hay Velocity, XLST, … thậm chí là những công nghệ view được custom bởi người dùng.

- Spring Controller được cấu hình thông qua IoC như mọi đối tượng khác. Điều này làm chúng dễ dàng được test, và được tích hợp dễ dàng với những đối tượng khác được quản lý bởi Spring.
- Kết buộc trực tiếp các input từ view với những đối tượng domains.

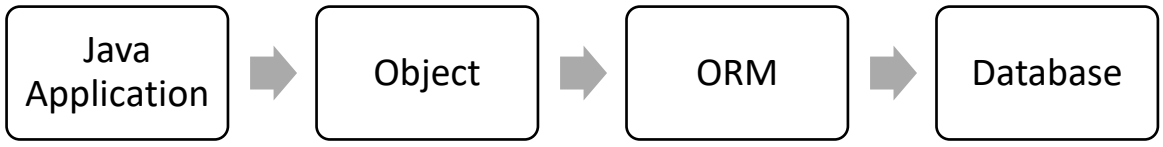
2.2.7.1.1 Nhược điểm

Cấu hình phức tạp và cồng kềnh ➔ không phát huy được sức mạnh khi áp dụng cho các ứng dụng quy mô nhỏ mà có thể ngược lại còn làm cho ứng dụng phức tạp.

2.3 Hibernate Framework

2.3.1 Khái niệm ORM và Persistent layer

- Object Relational Mapping (ORM) là kỹ thuật ánh xạ dữ liệu từ các cơ sở dữ liệu quan hệ đến các đối tượng trong các ngôn ngữ lập trình hướng đối tượng như Java. Trong đó các đối tượng ánh xạ với các bảng và các quan hệ giữa các đối tượng ánh xạ với các ràng buộc giữa các bảng.



- Peristence layer: một ứng dụng có thể được chia làm 3 phần như sau: giao diện người dùng (presentation layer), phần xử lý nghiệp vụ (business layer) và phần chứa dữ liệu (data layer). Cụ thể ra, business layer có thể được chia nhỏ thành 2 layer con là business logic layer (các tính toán nhằm thỏa mãn yêu cầu người dùng) và persistence layer. Persistence layer chịu trách nhiệm giao tiếp với data layer (thường là một hệ quản trị cơ sở dữ liệu quan hệ – Relational DBMS). Persistence sẽ đảm nhiệm các nhiệm vụ mở kết nối, truy xuất và lưu trữ dữ liệu vào các Relational DBMS.

2.3.1.1 Giới thiệu Hibernate Framework

Hibernate là một trong những ORM Framework. Hibernate framework là một framework cho persistence layer. Như vậy, nhờ có Hibernate framework mà giờ đây khi bạn phát triển ứng dụng bạn chỉ còn chú tâm vào những layer khác mà không phải bận tâm nhiều về persistence layer nữa. Hibernate giúp lưu trữ và truy vấn dữ liệu quan hệ mạnh mẽ và nhanh. Hibernate cho phép bạn truy vấn dữ liệu bằng ngôn ngữ SQL mở rộng của Hibernate (HQL) hoặc bằng SQL thuần.

2.3.1.2 Kiến trúc Hibernate Framework

Mô tả chức năng các file cấu hình trong Hibernate:

- Mỗi table trong database là một object trong Hibernate. Do đó, bạn cần có một java bean cho mỗi table trong database. Các java bean này sẽ có các getters/setters và một số ngoại lệ theo quy ước của Hibernate.
- Mỗi file mapping (ánh xạ) có dạng ***.hbm.xml có nhiệm vụ đặc tả mỗi quan hệ giữa các thuộc tính của đối tượng và các trường trong bảng CSDL
- File Hibernate.cfg.xml: Đây là tập tin được load đầu tiên khi khởi chạy ứng dụng Hibernate. Nó chứa các thông tin sau:
    - Connection JDBC (URL , Driver class, Username, password, pool_size,…).
    - SQL Dialect.
    - Hibernate configuration (show_sql, format_sql, use_sql_comment, default_schema, order_updates,…).
    - Hibernate cache configuration (class cache, collection cache).
    - Hibernate transaction configuration (factory_class, auto_close_sesion, manager_lookup_class)
    - Miscellaneous confguration (current_session_context_class, factory_class,…)
    - Mapping source configuration.

2.3.1.3 Các thành phần của Hibernate project

- Hibernate Core: Cung cấp các chức năng cơ bản của persistent layer cho các ứng dụng java với các APIs và hỗ trợ XML Mapping metadata.
- Hibernate Annotations: các map class với JDK 5.0 Anootations, bao gồm Hibernate Validator.
- Hibernate EntityManager : sử dụng EJB 3.0 API trong JSE hoặc với bất kỳ JEE nào.
- Hibernate Tools: các tool tích hợp với Eclipse và Ant giúp tạo ra các persistent object từ 1 schema có sẵn trong database(reverse-engineering) và từ các file hbm sinh ra các class java thực hiện các persistent object rồi Hibernate tự tạo tác object trong database (forward-engineering).
- NHibernate: Hibernate cho .NET Framework.
- Jboss Seam: Một Java EE 5.0 framework cho phát triển các ứng dụng JSF, Ajax và EJB 3.0 với sự hỗ trợ của Hibernate. Seam hiện rất mới và tỏ ra rất mạnh để phát triển các ứng dụng Web 2.0. Nó tích hợp đầy đủ tất cả các công nghệ "hot" nhất hiện nay.

2.3.1.4 Lợi ích khi sử dụng Hibernate cho persistent layer

Trước tiên cần kể tới những khó khăn khi triển khai ứng dụng business dùng ngôn ngữ SQL thuần túy:

- Mã thì rải rác khắp nơi đến những nơi không quản lý nổi.
- Nếu thiết kế bị thay đổi sau khi phát triển ứng dụng, phải trả giá rất đắt để nhận ra những nơi cần phải thay đổi.
- Khó tìm và fix bug.
- Quản lý các kết nối database là một nhiệm vụ cực kỳ khó khăn vì mã SQL nằm tùm lum, kết nối database cũng vậy.
- Quản lý transaction là một nhiệm vụ phức tạp.

Lợi ích khi sử dụng Hibernate (với HQL – Hibernate Query Language):

- Tìm kiếm và sắp xếp nhanh.
- Làm việc được với dữ liệu lớn.
- Làm việc trên nhóm dữ liệu.
- Joining, aggregating.
- Chia sẽ nhiều người dùng và nhiều vùng.
- Giải quyết tương tranh (Transaction)
- Hỗ trợ cho nhiều ứng dụng.
- Bảo đảm toàn vẹn.
- Ràng buộc nhiều cấp độ.
- Tách biệt giao tác.

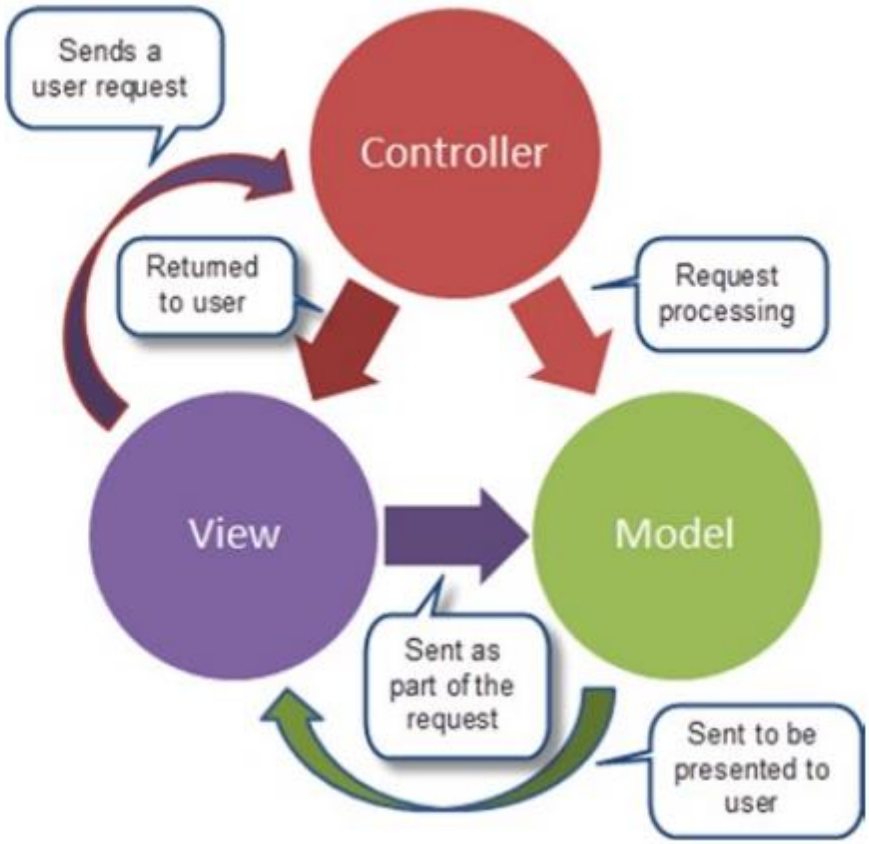
2.3.1.5 Những hệ quản trị cơ sở dữ liệu được hỗ trợ

Hibernate hỗ trợ hầu hết các hệ quản trị cơ sở dữ liệu phổ biến hiện nay, bao gồm: MySQL, Microsoft SQL Server, Oracle, DB2, PostgreSQL, FrontBase, Sybase, Informix Dynamic Server.

3 Thiết kế - Cài đặt

3.1 Kiến trúc hệ thống

Kiến trúc của hệ thống được triển khai trên nền tảng Spring MVC gồm 3 module cơ bản: Model, View và Controller.



Mô hình MVC

Thành phần	Diễn giải
Model	Là một phần của ứng dụng, các đối tượng này thiết lập logic phần dữ liệu của ứng dụng. Nói cách khác, Models chứa tất cả các xử lý mang tính nghiệp vụ, tính logic trong truy vấn cơ sở dữ liệu, cũng

	như tính hợp lệ trong ứng dụng.
View	Được xem là giao diện của ứng dụng, View có tác dụng trả về cho trình duyệt trang HTML khi người dùng sử dụng website của chúng ta.
Controller	Controllers đảm nhiệm việc xử lý logic ở phía ứng dụng bao gồm việc nhận giá trị đầu vào của ứng dụng, phát sinh các lệnh thực thi, nhận dữ liệu từ thành phần Model, và đưa người dùng đi đến các UI khác nhau.

3.2 Mô tả chi tiết từng thành phần trong hệ thống.

3.2.1 View

Đây là các thành phần dùng để hiển thị giao diện người dùng (UI). Views dùng để hiển thị cả các thông tin tĩnh và động cho phép người sử dụng giao tiếp với hệ thống để cập nhật thông tin.

Phần View sử dụng JSP để sinh ra các trang HTML. Mỗi đối tượng ánh xạ sang Database sẽ có tối đa một View. Các view tổng thế bao gồm các chức năng cơ bản như thêm, xoá, sửa.

Ngoài ra, View còn sử dụng các ngôn ngữ front-end web như: HTML, CSS, Javascript, Jquery và sử dụng kỹ thuật AJAX.

3.2.2 Controller

Trong hệ thống "Quản lý điều trị bệnh tại phòng mạch", Controller là một thành phần trong MVC, chuyên xử lý nghiệp vụ của hệ thống. Cũng là thành phần quan trọng xử lý các hành động của người dùng để làm việc với Model, và giao tiếp với phần View, là cầu nối giữa View và Model.

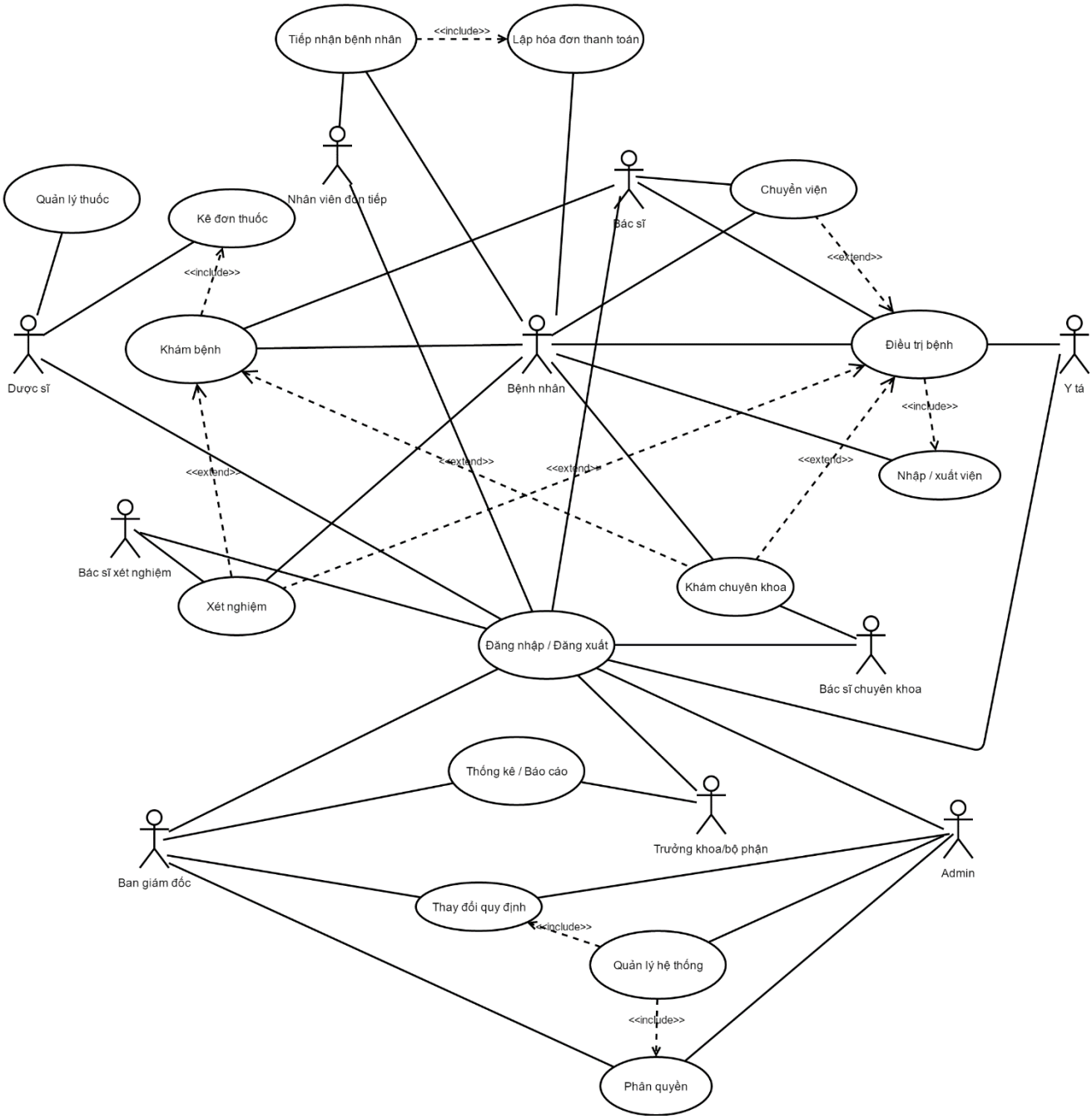
Các Controller trong quản lý điều trị bệnh tại phòng mạch: *BenhNhanController, UserController, DichVuController, BoPhanController…*

3.2.3 Model

Yêu cầu của người dùng được xuất phát từ View, View sẽ nhờ Controller để điều khiển các hành động, các giao tiếp, chuyển những thông tin từ Model xuống Cơ sở dữ liệu, tương tự, từ CSDL, thông qua Model thì Controller sẽ điều khiển để hiển thị trên View.

3.3 Sơ đồ Use case

3.3.1 Sơ đồ



3.3.2 Danh sách các actor

STT	Tên Actor	Vai trò/ Ghi chú
1	Admin	Chịu quản lý hệ thống trên máy tính.
2	Bác sĩ	Có quyền đăng nhập, xem thông tin cá nhân, thay đổi mật khẩu, tìm kiếm bệnh nhân đã chữa trị.
3	Dược sĩ	Có quyền đăng nhập, xem thông tin cá nhân, thay đổi mật khẩu, tìm kiếm các đơn thuốc đã kê.
4	Y tá	Có quyền đăng nhập, xem thông tin cá nhân, thay đổi mật khẩu, tìm kiếm thông tin bệnh nhân đã chăm sóc.
5	Nhân viên	Có quyền đăng nhập, xem thông tin cá nhân, thay đổi mật khẩu.

3.3.3 Danh sách các use case

STT	Tên Use-case	Ý nghĩa/ Ghi chú
1	Quản lý tiếp nhận bệnh nhân.	Giúp phòng khám quản lý tiếp nhận bệnh nhân.
2	Khám bệnh.	Lưu trữ dữ liệu bệnh án của

		bệnh nhân.
3	Thanh toán.	Use case này giúp phòng mạch lưu trữ thông tin thánh toán.
4	Xét nghiệm.	Use case này giúp lưu trữ thông tin thực hiện xét nghiệm của bệnh nhân.
5	Khám chuyên khoa.	Use case này giúp lưu trữ thông tin thực hiện khám chuyên khoa của bệnh nhân.
6	Kê đơn thuốc.	Use case này giúp lữu trữ thông tin đơn thuốc của bệnh nhân.
7	Điều trị bệnh.	Use case này giúp lưu trữ thông tin bệnh nhân đang thực hiện điều trị tại phòng khám.
8	Nhập/ xuất viện.	Use case này để lưu trữ thông tin xác nhận nhập/xuất viện của bệnh nhân tại phòng mạch.
9	Chuyển viện.	Use case để lưu trữ thông tin xác nhận chuyển viện của bệnh nhân.
10	Quản lý thuốc.	Use case để lưu trữ thông tin thuốc tại phòng mạch.
11	Thống kê/ báo cáo.	Use case này để trình bày các số liệu định kì về hoạt động của phòng khám.
12	Đăng nhập/ đăng xuất.	Use case này để xác thực

		người dùng truy cập hệ thống.
13	Quản lý hệ thống.	Use case này để giúp cho admin truy cập hệ thống và quản lý tài khoản người dùng.
14	Phân quyền.	Use case này để phân quyền truy cập hệ thống.
15	Thay đổi quy định.	Use case này để thay đổi các quy định của hệ thống theo yêu cầu của ban giám đốc.

3.3.4 Đặc tả use case

3.3.4.1 Use case "Đăng nhập/ đăng xuất"

Tên chức năng	**Đăng nhập/ đăng xuất**
Tóm tắt	Tất cả nhân viên của phòng khám đều phải đăng nhập để sử dụng hệ thống, đăng xuất khi kết thúc phiên làm việc.
Dòng sự kiện chính	1. Nhập tên đăng nhập và mật khẩu được cấp. 2. Bấm nút đăng xuất khỏi hệ thống.
Dòng sự kiện phụ	
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính phải kết nối với hệ thống.
Trạng thái hệ thống sau khi thực hiện use case	1. Máy tính ở trạng thái sẵn sàng cho phiên làm việc của người dùng. 2. Người dùng chỉ truy cập được những chức năng

	được quy định.
Điểm mở rộng	

3.3.4.2 Use case "Quản lý tiếp nhận bệnh nhân"

Tên chức năng	**Quản lý tiếp nhận bệnh nhân**
Tóm tắt	Use case bắt đầu khi bệnh nhân đến phòng khám, nhân viên đón tiếp thực hiện tiếp nhận bệnh nhân.
Dòng sự kiện chính	1. Nhân viên kiểm tra thông tin bệnh nhân đã tồn tại hay chưa. 2. Bện nhân cung cấp thông tin cần thiết. 3. Nhân viên ghi nhận thông tin vào hồ sơ bệnh nhân. 4. Nhân viên lập phiếu khám bệnh cho bệnh nhân.
Dòng sự kiện phụ	1. Thông tin của bệnh nhân đã tồn tại thì nhân viên chỉ lập phiếu khám bệnh cho bệnh nhân.
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính của nhân viên đón tiếp phải kết nối với hệ thống quản lý. 2. Nhân viên đón tiếp phải đăng nhập vào hệ thống. 3. Cơ sở dữ liệu phải sẵn sàng để nhập liệu. 4. Máy in có đủ giấy để in phiếu cho bệnh nhân.
Trạng thái hệ thống sau khi thực hiện use case	1. Hệ thống phải ở trạng thái tiếp nhận bệnh nhân mới.
Điểm mở rộng	

3.3.4.3 Use case "Khám bệnh"

Tên chức năng	**Khám bệnh**
Tóm tắt	Sau khi nhận phiếu khám bệnh, bệnh nhân vào các phòng khám bệnh. Bác sĩ thực hiện khám bệnh và chẩn đoán.

Dòng sự kiện chính	1. Bệnh nhân mang phiếu khám bệnh đến phòng khám bệnh và chờ gọi tên. 2. Bệnh nhân vào phòng khám, bác sĩ thực hiện khám bệnh, ghi nhận tình trạng và đưa ra chẩn đoán. 3. Bác sĩ (hoặc trợ lý) nhập các thông tin khám bệnh vào hồ sơ bệnh nhân trên hệ thống. 4. Bác sĩ kê đơn thuốc cho bệnh nhân (use case Kê đơn thuốc).
Dòng sự kiện phụ	1. Bác sĩ có thể chỉ định bệnh nhân thực hiện các xét nghiệm. 2. Bác sĩ chỉ định bệnh nhân thực hiện khám chuyên khoa.
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính của bác sĩ phải kết nối với hệ thống. 2. Bác sĩ phải đăng nhập vào hệ thống. 3. Máy tính của bác sĩ có thể xem thông tin bệnh nhân. 4. Cơ sở dữ liệu sẵn sàng để nhập liệu.
Trạng thái hệ thống sau khi thực hiện use case	1. Máy tính ở trạng thái hiện thông tin bệnh nhân tiếp theo.
Điểm mở rộng	

3.3.4.4 Use case "Xét nghiệm"

Tên chức năng	**Xét nghiệm**
Tóm tắt	Bệnh nhân được yêu cầu thực hiện xét nghiệm theo chỉ định của bác sĩ khám bệnh hoặc bác sĩ điều trị.
Dòng sự kiện chính	1. Bác sĩ lập phiếu yêu cầu xét nghiệm sau khi thực

	hiện khám bệnh hoặc trong quá trình điều trị bệnh. 2. Bệnh nhân mang phiếu yêu cầu xét nghiệm đến phòng xét nghiệm và chờ gọi tên. 3. Bác sĩ xét nghiệm thực hiện các xét nghiệm theo yêu cầu. 4. Bệnh nhân nhận kết quả xét nghiệm.
Dòng sự kiện phụ	
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính của bác sĩ phải kết nối với hệ thống. 2. Bác sĩ phải đăng nhập vào hệ thống. 3. Máy tính của bác sĩ có thể xem thông tin bệnh nhân. 4. Cơ sở dữ liệu sẵn sàng để nhập liệu. 5. Máy in có đủ giấy để in phiếu.
Trạng thái hệ thống sau khi thực hiện use case	1. Máy tính ở trạng thái hiện thông tin bệnh nhân tiếp theo.
Điểm mở rộng	

3.3.4.5 Use case "Khám chuyên khoa"

Tên chức năng	**Khám chuyên khoa**
Tóm tắt	Bệnh nhân được yêu cầu thực hiện khám chuyên khoa theo chỉ định của bác sĩ khám bệnh hoặc bác sĩ điều trị.
Dòng sự kiện chính	Tương tự nhi quy trình xét nghiệm.
Dòng sự kiện phụ	
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính của bác sĩ phải kết nối với hệ thống. 2. Bác sĩ phải đăng nhập vào hệ thống. 3. Máy tính của bác sĩ có thể xem thông tin bệnh nhân. 4. Cơ sở dữ liệu sẵn sàng để nhập liệu.

	5. Máy in có đủ giấy để in phiếu.
Trạng thái hệ thống sau khi thực hiện use case	1. Máy tính ở trạng thái hiện thông tin bệnh nhân tiếp theo.
Điểm mở rộng	

3.3.4.6 Use case "Kê đơn thuốc"

Tên chức năng	**Kê đơn thuốc**
Tóm tắt	Bác sĩ khám bệnh kê đơn thuốc cho bệnh nhân sau khi đưa ra chẩn đoán. Dược sĩ nhận đơn và bán thuốc cho bệnh nhân.
Dòng sự kiện chính	1. Bác sĩ kiểm tra các loại thuốc cần thiết có sẵn trong kho thuốc hay không, tiến hành nhập đơn thuốc và in đơn thuốc cho bệnh nhân. 2. Bệnh nhân mang đơn thuốc đến quầy bán thuốc tại phòng khám. 3. Dược sĩ kiểm tra đơn thuốc thuốc và bán thuốc cho bệnh nhân.
Dòng sự kiện phụ	1. Nếu loại thuốc cần thiết không còn trong kho thuốc, bác sĩ kê loại thuốc có công dụng tương tự còn trong kho cho bệnh nhân.
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính của bác sĩ phải kết nối với hệ thống. 2. Bác sĩ phải đăng nhập vào hệ thống. 3. Máy tính của bác sĩ có thể xem danh mục thuốc sẵn có trong kho thuốc. 4. Cơ sở dữ liệu sẵn sàng để nhập liệu. 5. Máy in có đủ giấy để in đơn thuốc.
Trạng thái hệ thống	1. Máy tính ở trạng thái hiện thông tin bệnh nhân tiếp

sau khi thực hiện use case	theo.
Điểm mở rộng	

3.3.4.7 Use case "Điều trị bệnh"

Tên chức năng	Điều trị bệnh
Tóm tắt	Trong qua trình khám bệnh, nếu thấy cần thiết bác sĩ có thể yêu cầu bệnh nhân ở lại điều trị tại phòng khám hoặc bệnh nhân tự yêu cầu điều trị.
Dòng sự kiện chính	1. Bác sĩ yêu cầu bệnh nhân điều trị tại phòng khám hoặc bệnh nhân tự yêu cầu điều trị. 2. Bệnh nhân làm thủ tục nhập viện. 3. Bác sĩ điều trị và y tá thực hiện công việc điều trị, ghi nhận thông tin điều trị hàng ngày vào hồ sơ bệnh nhân. 4. Bác sĩ điều trị có thể yêu cầu bệnh nhân thực hiện các xét nghiệm và khám chuyên khoa cần thiết trong quá trình điều trị. 5. Bệnh nhân thanh toán viện phí và xuất viện.
Dòng sự kiện phụ	1. Bệnh nhân từ chối điều trị. 2. Bệnh nhân từ chối thực hiện xét nghiệm hoặc khám chuyên khoa.
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính của bác sĩ hoặc y tá phải kết nối với hệ thống. 2. Bác sĩ hoặc y tá phải đăng nhập vào hệ thống. 3. Cơ sở dữ liệu sẵn sàng để nhập liệu.
Trạng thái hệ thống sau khi thực hiện use	1. Máy tính ở trạng thái xem danh sách bệnh nhân đang điều trị tại phòng khám.

case	
Điểm mở rộng	

3.3.4.8 Use case "Lập hoá đơn thanh toán"

Tên chức năng	**Lập hoá đơn thanh toán.**
Tóm tắt	Use case bắt đầu khi bệnh nhân yêu cầu hoặc được chỉ định thực hiện một dịch vụ của phòng khám. Bệnh nhân thanh toán tại quầy thu ngân.
Dòng sự kiện chính	1. Bệnh nhân mang các phiếu dịch vụ (khám bệnh, xét nghiệm…) đến quầy thu ngân. 2. Nhân viên phòng khám kiểm tra các phiếu tương ứng trên hệ thống và lập hóa đơn cho bệnh nhân 3. Nhận tiền thanh toán từ bệnh nhân. 4. Nhập số tiền thu. 5. Cập nhật tình trạng thanh toán của các phiếu dịch vụ trên hệ thống.
Dòng sự kiện phụ	
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính của nhân viên đón tiếp phải kết nối với hệ thống quản lý. 2. Nhân viên thu ngân phải đăng nhập vào hệ thống. 3. Cơ sở dữ liệu phải sẵn sàng để nhập liệu. 4. Máy in có đủ giấy để in hóa đơn cho bệnh nhân.
Trạng thái hệ thống sau khi thực hiện use case	1. Hệ thống ở trạng thái nhập hóa đơn mới
Điểm mở rộng	

3.3.4.9 *Use case "Nhập/ xuất viện"*

Tên chức năng	Nhập/ xuất viện
Tóm tắt	Use case được thực hiện khi bệnh nhân bắt đầu hoặc kết thúc điều trị tại phòng khám.
Dòng sự kiện chính	1. Bác sĩ lập giấy nhập viện cho bệnh nhân, chỉ định phòng, khoa điều trị. Sau khi điều trị, bác sĩ lập giấy xuất viện cho bệnh nhân. 2. Bác sĩ và bệnh nhân kí giấy nhập hoặc xuất viện. 3. Bệnh nhân thanh toán các chi phí. 4. Cập nhật thông tin vào hồ sơ bệnh nhân.
Dòng sự kiện phụ	
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính của bác sĩ phải kết nối với hệ thống. 2. Bác sĩ phải đăng nhập vào hệ thống. 3. Cơ sở dữ liệu sẵn sàng để nhập liệu.
Trạng thái hệ thống sau khi thực hiện use case	
Điểm mở rộng	

3.3.4.10 *Use case "Chuyển viện"*

Tên chức năng	Chuyển viện
Tóm tắt	Trong qua trình điều trị, bác sĩ có thể yêu cầu bệnh nhân chuyển viện nếu cần thiết.
Dòng sự kiện chính	1. Bác sĩ lập giấy chuyển viện cho bệnh nhân. 2. Giám đốc phòng khám kí giấy chuyển viện. 3. Bệnh nhân thanh toán các chi phí. 4. Cập nhật thông tin vào hồ sơ bệnh nhân.

	5. Thực hiện chuyển viện.
Dòng sự kiện phụ	
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính của bác sĩ phải kết nối với hệ thống. 2. Bác sĩ phải đăng nhập vào hệ thống. 3. Cơ sở dữ liệu sẵn sàng để nhập liệu.
Trạng thái hệ thống sau khi thực hiện use case	
Điểm mở rộng	

3.3.4.11 Use case "Quản lý thuốc"

Tên chức năng	**Quản lý thuốc**
Tóm tắt	Quản lý số lượng thuốc trong kho thuốc.
Dòng sự kiện chính	1. Nhập thông tin thuốc. 2. Cập nhật số lượng thuốc.
Dòng sự kiện phụ	
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính của dược sĩ phải kết nối với hệ thống. 2. Dược sĩ phải đăng nhập vào hệ thống. 3. Cơ sở dữ liệu sẵn sàng để nhập liệu.
Trạng thái hệ thống sau khi thực hiện use case	
Điểm mở rộng	

3.3.4.12 Use case "Thống kê/ báo cáo"

Tên chức năng	Thống kê/ báo cáo
Tóm tắt	Thống kê, báo cáo định kì về hoạt động của phòng khám.
Dòng sự kiện chính	1. Trưởng khoa/bộ phận lập thống kê, báo cáo. 2. Ban giám đốc duyện báo cáo.
Dòng sự kiện phụ	
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính của người lập báo cáo và người duyệt báo cáo phải kết nối với hệ thống. 2. Bác sĩ hoặc phải đăng nhập vào hệ thống. 3. Cơ sở dữ liệu sẵn sàng để nhập, xuất dữ liệu. 4. Máy in có đủ giấy để in báo cáo.
Trạng thái hệ thống sau khi thực hiện use case	1. Máy tính ở trạng thái xem danh sách các báo cáo.
Điểm mở rộng	

3.3.4.13 Use case "Quản lý hệ thống"

Tên chức năng	Quản lý hệ thống
Tóm tắt	Người quản trị hệ thống (admin) có nhiệm vụ quản lý hoạt động của hệ thống, quản lý cơ sở dữ liệu, theo dõi và khắc phục các sự cố có thể xảy ra, làm việc với nhà phát triển hệ thống để yêu cầu tư vấn, hỗ trợ.
Dòng sự kiện chính	1. Admin thực hiện các công việc quản lý, theo dõi.

Dòng sự kiện phụ	1. Nếu có sự cố xảy ra, admin phải báo cáo với ban giám đốc. 2. Trong trường hợp không thể xử lý được sự cố, admin liên hệ với nhà phát triển hệ thống để yêu cầu hỗ trợ.
Trạng thái hệ thống trước khi thực hiện use case	1. Máy tính phải kết nối với hệ thống. 2. Admin phải đăng nhập vào hệ thống.
Trạng thái hệ thống sau khi thực hiện use case	1. Máy tính ở trạng thái sẵn sàng cho phiên làm việc của admin.
Điểm mở rộng	

3.4 Thiết kế giao diện

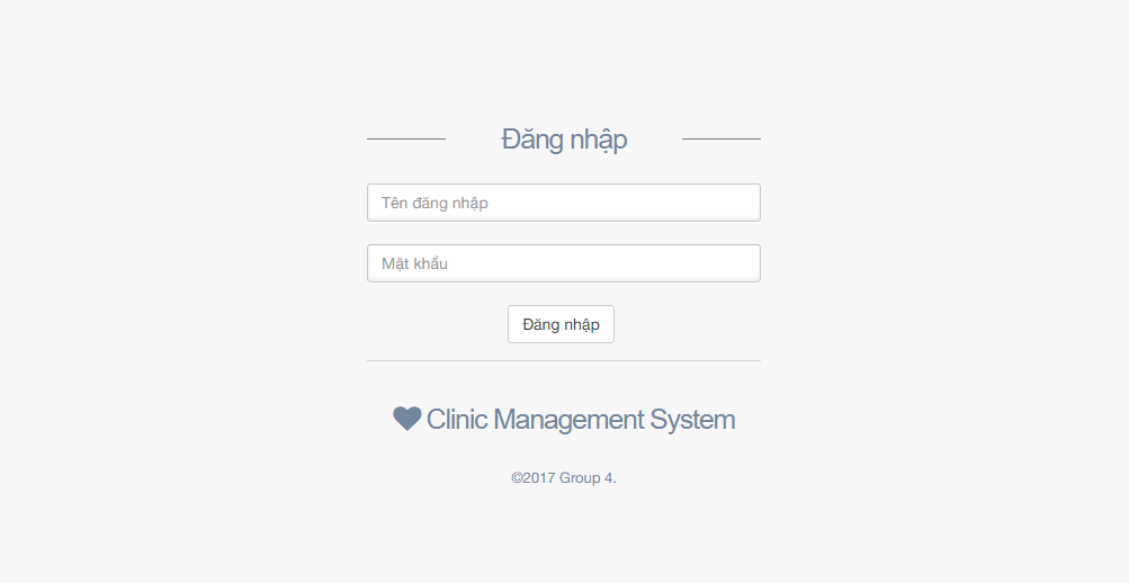
3.4.1 Danh sách các màn hình.

STT	**Tên màn hình**	**Ý nghĩa/Ghi chú**
1	Đăng nhập	Màn hình đăng nhập hệ thống.
3	Trang chủ	Màn hình sau khi đăng nhập vào hệ thống.
4	Quản lý người dùng	Màn hình quản lý danh sách người dùng, thêm, cập nhập, xóa người dùng.
5	Quản lý bộ phận	Màn hình quản lý danh sách các bộ phận trong phòng khám, thêm, cập nhật, xóa bộ phận
20	Màn hình tiếp nhận bệnh nhân	Màn hình hiển thị chức năng tiếp nhận bệnh nhân của bộ nhận tiếp nhận.
21	Màn hình danh sách bệnh nhân	Màn hình hiển thị danh sách bệnh nhân đã khám tại phòng mạch.

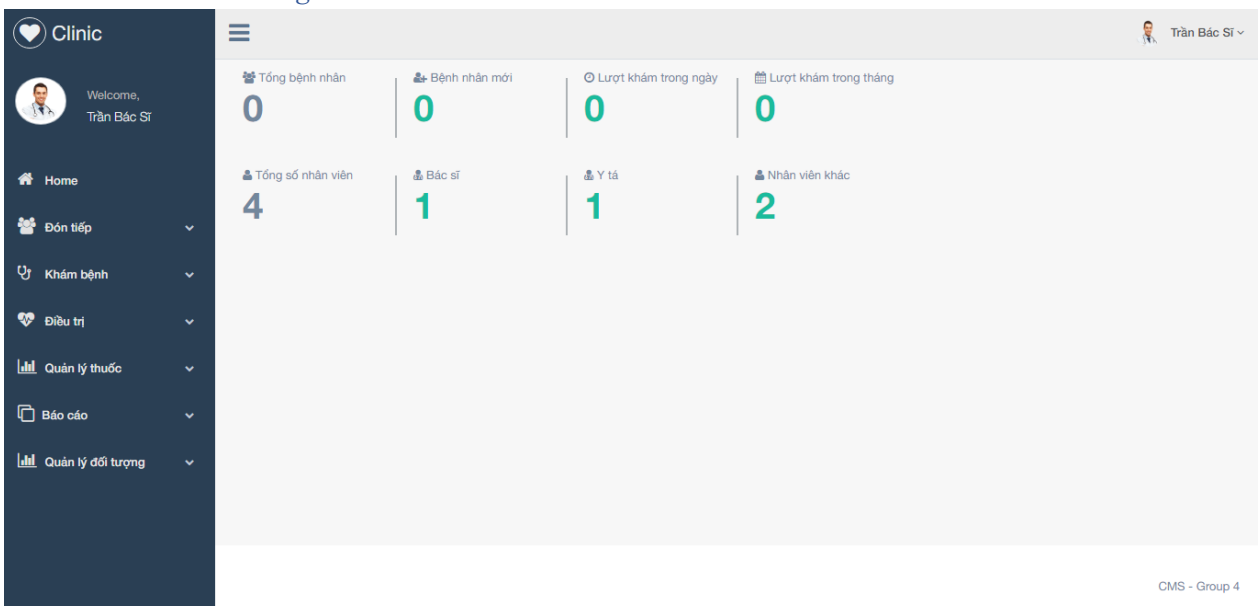
	Khám khám bệnh	Màn hình thực hiện chức năng khám bệnh của bác sĩ.
22	Danh sách điều trị	Mành hình danh sách các bệnh nhân đang điều trị nội trú tại phòng khám, cập nhật trạng thái điều trị, xuất viện.
	Quản lý dịch vụ	Màn hình quản lý danh sách các dịch vụ của phòng khám, thêm, cập nhật, xóa phòng khám.

3.4.2 Chi tiết các màn hình.

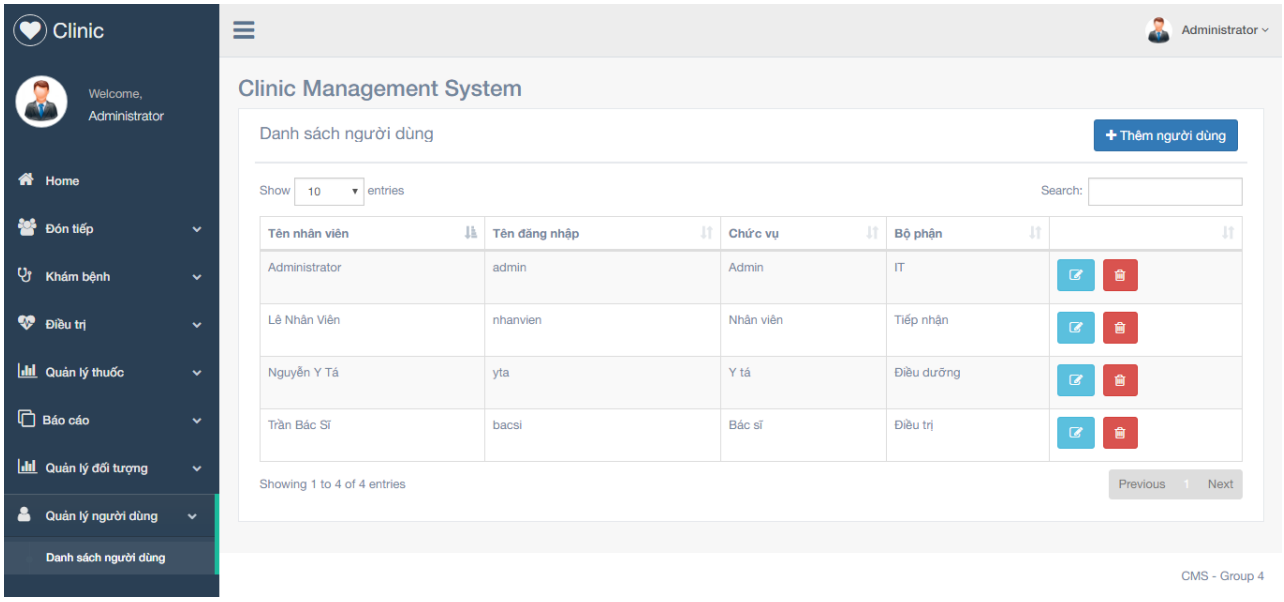
3.4.2.1 Màn hình Đăng nhập

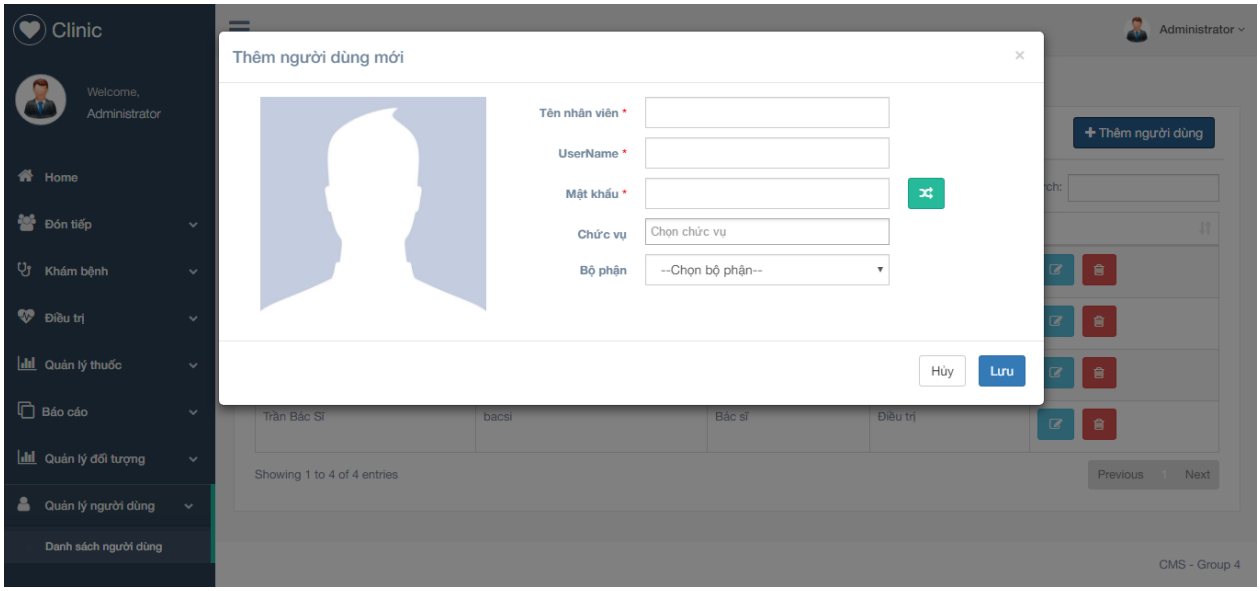


3.4.2.2 *Màn hình Trang chủ*



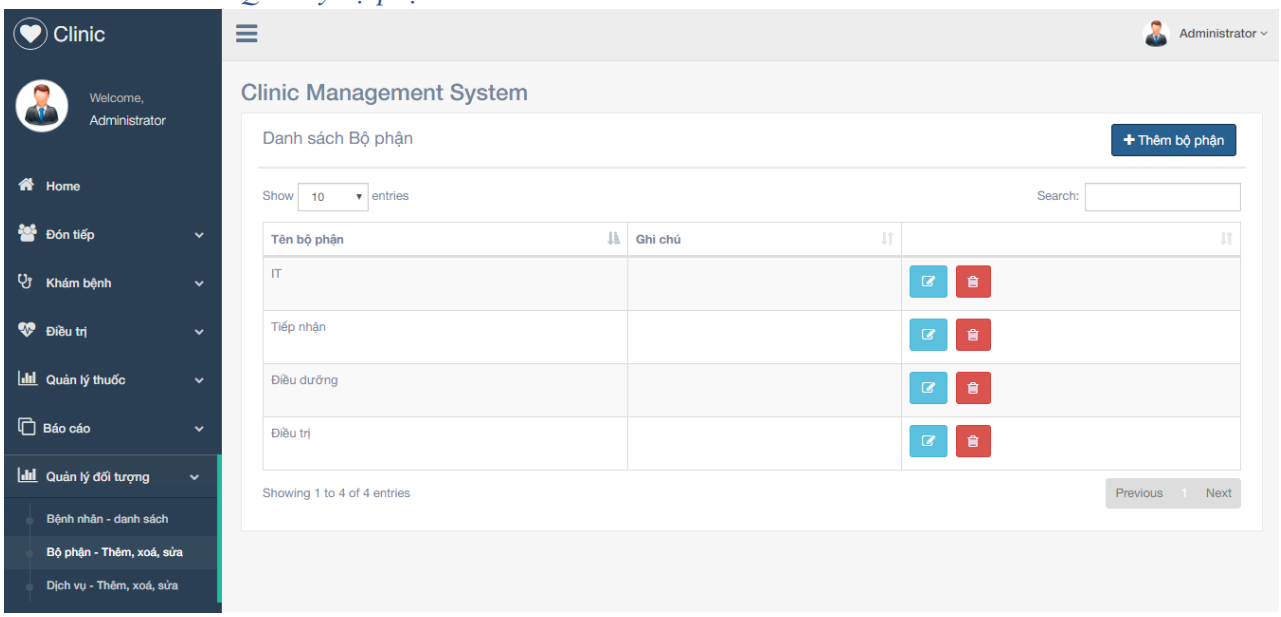
3.4.2.3 *Màn hình Quản lý người dùng*

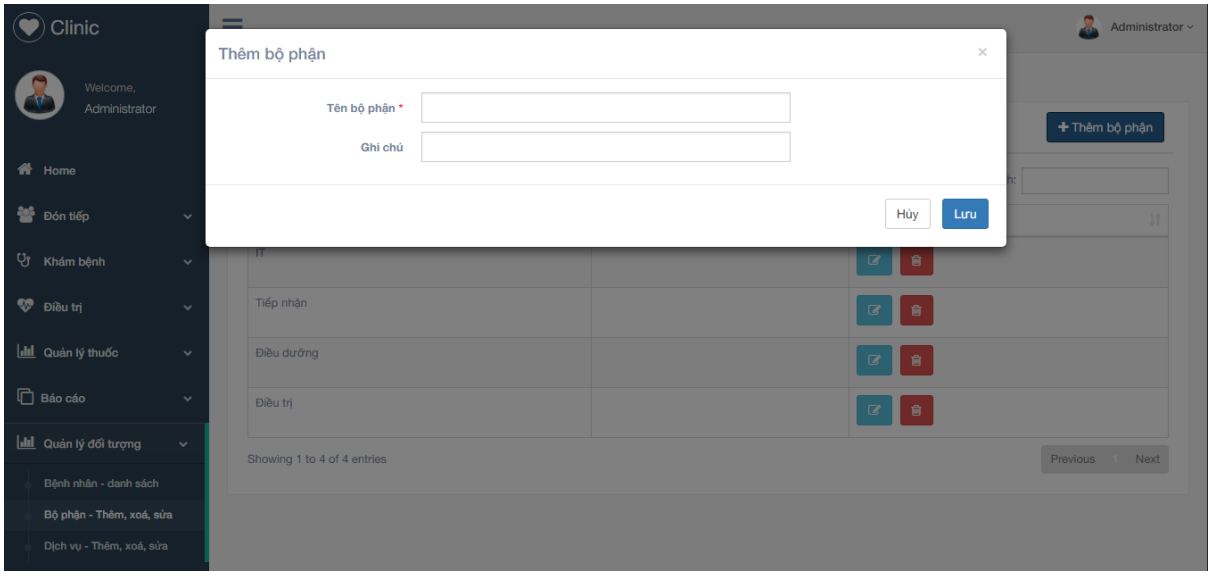




Cửa sổ thêm người dùng mới trong màn hình Quản lý người dùng

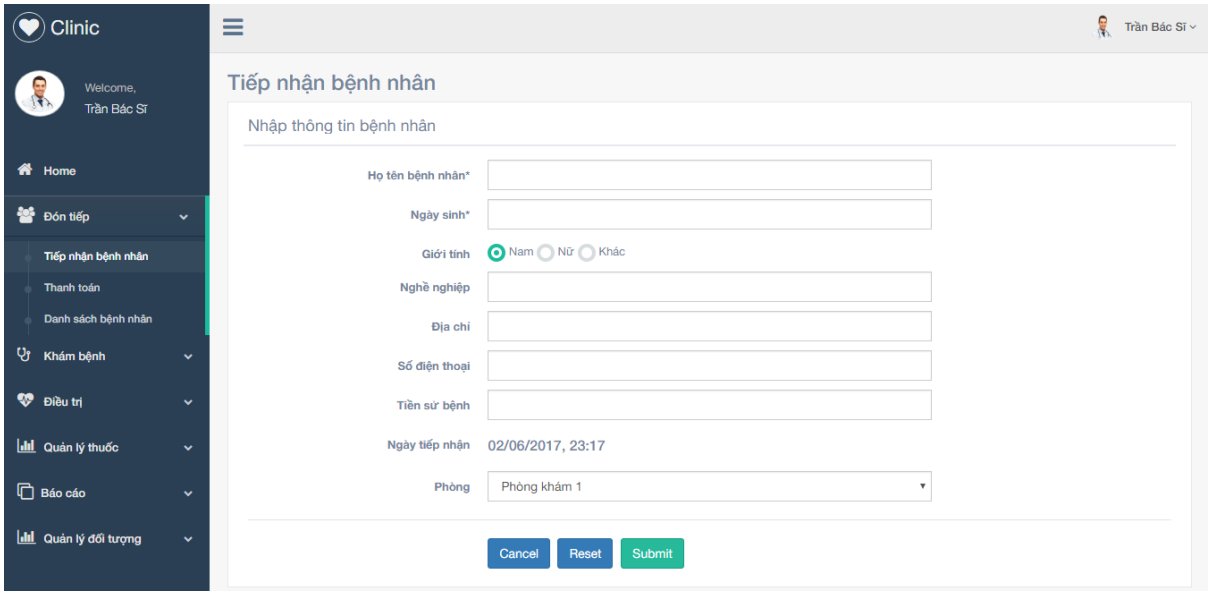
3.4.2.4 Màn hình Quản lý bộ phận



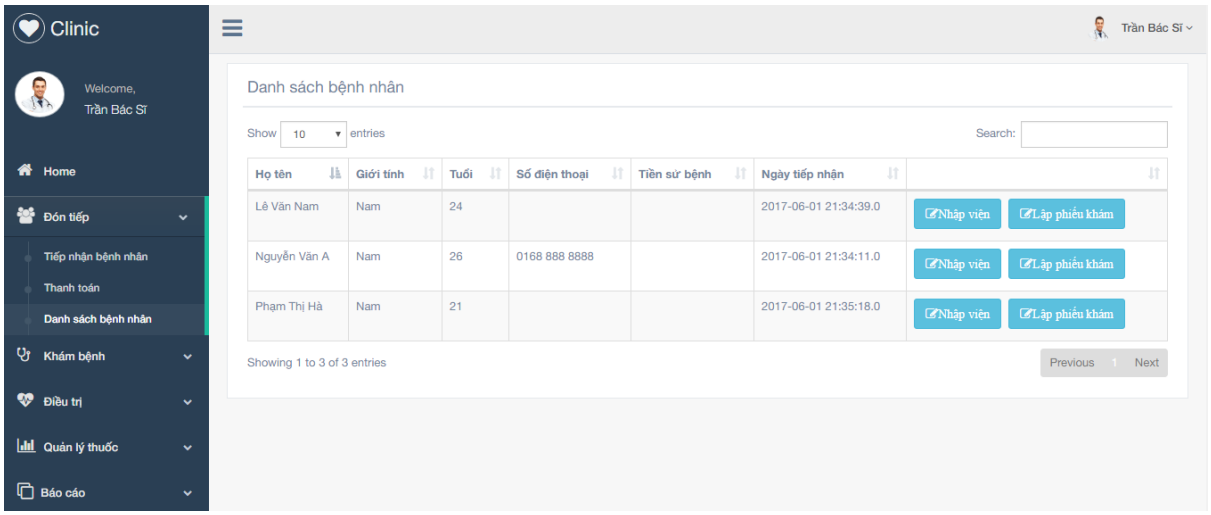


Cửa sổ thêm bộ phân mới trong màn hình Quản lý bộ phận

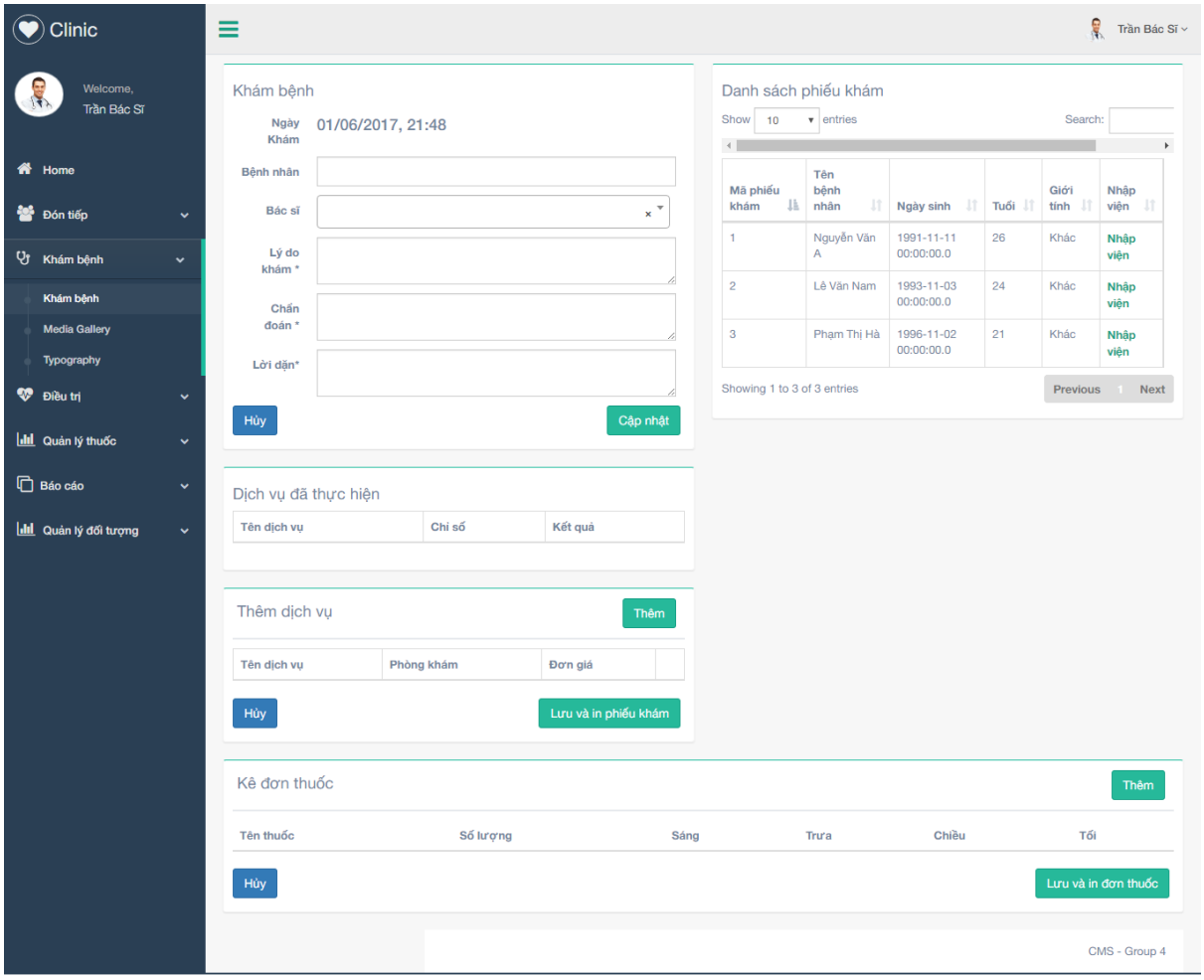
3.4.2.5 Màn hình Tiếp nhận bệnh nhân



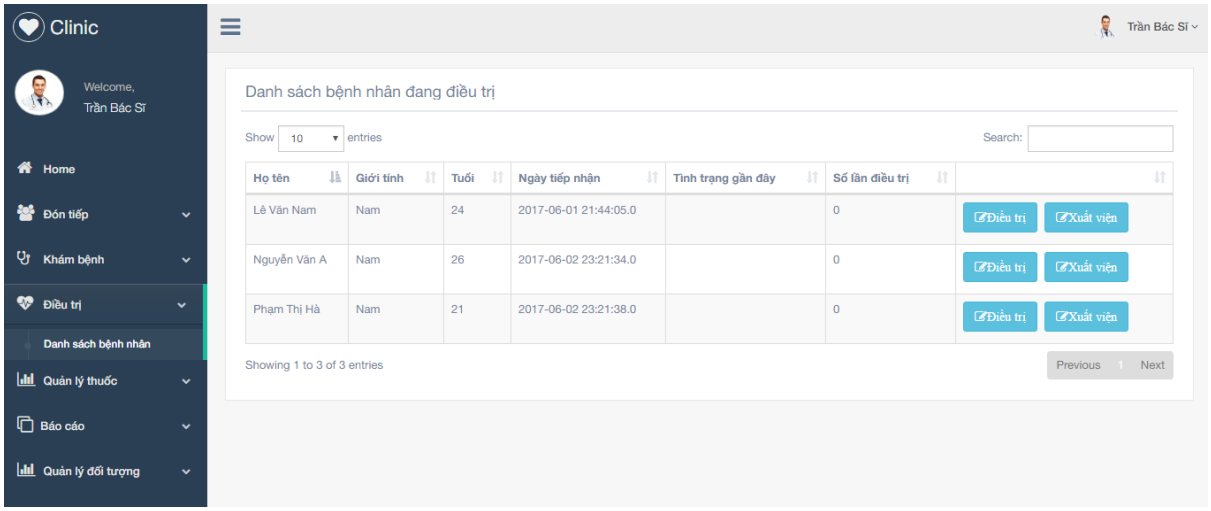
3.4.2.6 *Màn hình Danh sách bệnh nhân*



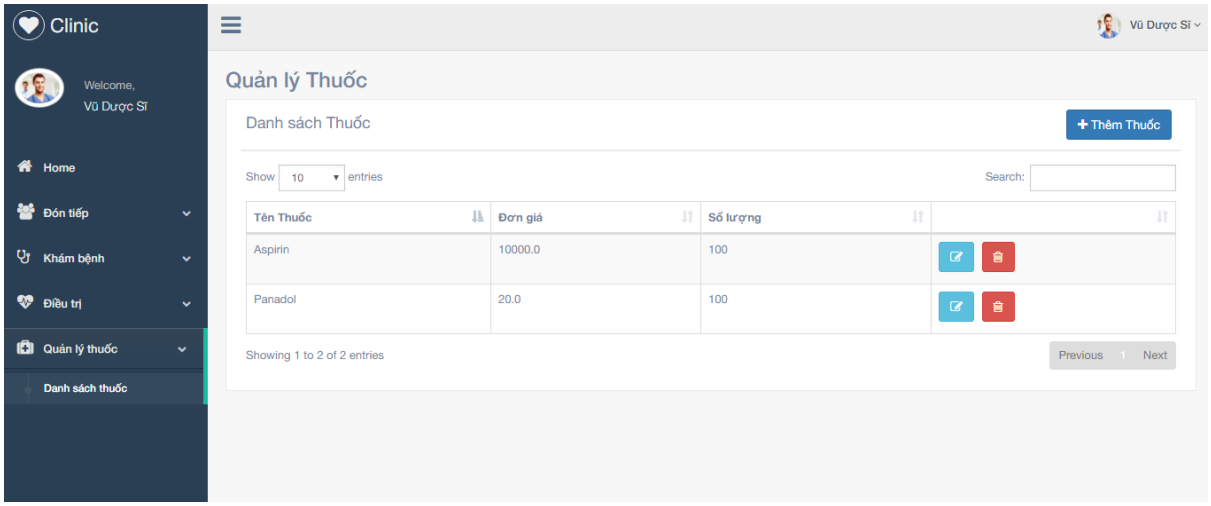
3.4.2.7 *Màn hình Khám bệnh*



3.4.2.8 Màn hình Danh sách điều trị



3.4.2.9 Mành hình Quản lý thuốc



4 Hướng dẫn cài đặt

Project được xây dựng với IDE Eclipse JEE, hệ quản trị cơ sở dữ liệu MySQL.

Các bước cài đặt ứng dụng:

- Khởi chạy Eclipse EE, chọn File ➔ Import… ➔ Exiting Maven Projects ➔ Browse.. ➔ Chọn thư mục chứa project ➔ Finish.
- Chạy file Script1.sql bằng MySQL để khởi tạo cơ sở dữ liệu
- Trên Project Explorer của Eclipse mở file *src\main\webapp\WEB-INF\spring-configuration\config.properties*, thay đổi url của cơ sở dữ liệu, username và password phù hợp với MySQL đang sử dụng.
- Thêm username và password cho người quản trị đầu tiên (người có quyền thêm user mới): mở file src\main\java\com\group4\cms\config\SecurityConfig.java và thêm vào phương thức configureGlobal dòng sau:

  auth.inMemoryAuthentication().withUser("admin_name").password("password").roles("ADMIN");

  Trong đó admin-name là username của người quản trị đầu tiên.

- Click chuột phải vào project chọn Run As ➔ Run On Server ➔ Chọn server Tomcat 8.0 sau đó click Finish.
- Có thể truy cập ứng dụng bằng bất cứ trình duyệt nào, địa chỉ ứng dụng trên server Tomcat có dạng http://localhost:8080/clinic/.
- Sau khi ứng dụng đã khởi chạy và Hibernate đã tự động tạo ra các bảng trong cơ sở dữ liệu, chạy file Script2.sql để thêm một số dữ liệu cần thiết