

# TRƯỜNG ĐẠI HỌC TIỀN GIANG

## KHOA CÔNG NGHỆ THÔNG TIN



### BÁO CÁO THỰC TẬP TỐT NGHIỆP

## NGHIÊN CỨU CÔNG NGHỆ MICROSERVICE ỨNG DỤNG LẬP TRÌNH HỆ THỐNG QUẢN LÝ SUẤT ĂN CHO BỆNH VIỆN

Đơn vị thực tập: Trung tâm giải pháp y tế điện tử VNPT Tiền Giang - Phòng giải pháp phần mềm 3

Địa chỉ: Xã Tân Mỹ Chánh, Tân Mỹ Chánh, Thành phố Mỹ Tho, Tiền Giang

Cán bộ hướng dẫn:

SV. Thực Hiện

1. Dương Văn Thích
2. Nguyễn Thái Duy
3. Dương Văn Hiếu

1. Huỳnh Công Kháng 015101060
2. Lê Lâm Khánh Duy 015101007
3. Phạm Trúc Tiên 015101073

Tiền Giang, 30 tháng 03 năm 2019

## **LỜI CẢM ƠN**

Trong suốt quá trình học tập tại trường Đại học Tiền Giang, em đã tiếp thu được rất nhiều kiến thức thiết thực và bổ ích. Điều đó đã giúp em trưởng thành hơn trong cuộc sống và giúp em có thể xác định được con đường đúng đắn mà mình nên đi trong tương lai, đó là cách sống có ích cho bản thân, gia đình, bạn bè và cho toàn xã hội.

Trên thực tế, không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập tại trường cho đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý thầy cô, gia đình và người dùng bè.

Để hoàn thành báo cáo, ngoài sự nỗ lực của bản thân, em trân trọng gửi lời cảm ơn sâu sắc đến:

Thầy Dương Văn Hiếu, người đã tận tình hướng dẫn, giúp em xác định rõ được mục tiêu và phương hướng để hoàn thành báo cáo thực tập. Anh Dương Văn Thích, anh Nguyễn Thái Duy đã vô cùng nhiệt tình, chu đáo đã hướng dẫn em trong quá trình thực tập tại Trung tâm giải pháp y tế điện tử VNPT Tiền Giang - Phòng giải pháp phần mềm 3; bất cứ khi nào em gặp khó khăn trong quá trình thực tập, các anh đã giúp đỡ, chỉ dẫn một cách chi tiết nhất, cụ thể nhất.

Bước đầu đi vào thực tế, tìm hiểu nghiên cứu, kiến thức của em vẫn còn hạn chế và còn nhiều bỡ ngỡ. Do vậy, không tránh khỏi những thiếu sót là điều chắc chắn, em rất mong nhận được những ý kiến đóng góp quý báu, để kiến thức của em trong lĩnh vực này được hoàn thiện hơn.

Một lần nữa em xin chân thành cảm ơn!

Tiền Giang, Ngày 30 Tháng 03 Năm 2019

## LỊCH LÀM VIỆC

Họ và tên sinh viên: Huỳnh Công Kháng  
Lê Lâm Khánh Duy  
Phạm Trúc Tiên

MSSV: 015101060  
MSSV: 015101007  
MSSV: 015101073

Cơ quan thực tập: Trung tâm giải pháp y tế điện tử VNPT Tiền Giang - Phòng giải pháp phần mềm 3

Họ và tên cán bộ hướng dẫn: Dương Văn Thích – Nguyễn Thái Duy

Thời gian thực tập, từ ngày 18 tháng 2 năm 2019 đến ngày 30 tháng 3 năm 2019

Tuần	Nội dung công việc được giao	Tự nhận xét về mức độ hoàn thành	Nhận xét của CB hướng dẫn	Chữ ký của CB HD
1 Từ ngày ..... đến ngày .....				
2 Từ ngày ..... đến ngày .....				
3 Từ ngày ..... đến ngày .....				
4 Từ ngày ..... đến ngày .....				
5 Từ ngày ..... đến ngày .....				
6 Từ ngày ..... đến ngày .....				

Ngày .....tháng..... năm 2017

(Ký tên, đóng dấu)

## MỤC LỤC

Nội dung	Trang
PHẦN 1: TỔNG QUAN VỀ CÔNG TY THỰC TẬP .....	1
1.1 Giới thiệu chung .....	1
1.2 Lĩnh vực kinh doanh.....	1
1.3 Logo.....	2
PHẦN 2: NỘI DUNG THỰC HIỆN .....	3
2.1 Tìm hiểu quy trình khám chữa bệnh.....	3
2.1.1 Tìm hiểu kho Dược.....	3
2.1.2 Sơ đồ kho Dược.....	4
2.1.3 Cận Lâm Sàng .....	5
2.2 Làm quen với các công cụ, ứng dụng hỗ trợ cơ bản. ....	7
2.2.1 GitHub .....	7
PHẦN 3: KẾT QUẢ ĐẠT ĐƯỢC.....	22
3.1 Lý thuyết.....	22
3.1.1 Quản lý kho dược .....	22
3.1.2 Cận Lâm Sàng .....	22
3.1.3 Microservice .....	22
3.1.3 Jhipster (Spring boot + React).....	22
3.2 Phạm vi áp dụng .....	22
3.2 Sản phẩm: .....	23
PHẦN 4 : BÀI HỌC KINH NGHIỆM CHO BẢN THÂN .....	26
TÀI LIỆU THAM KHẢO.....	27

## DANH MỤC HÌNH

<i>Nội dung</i>	<i>Trang</i>
Hình 1.1 Logo.....	2
Hình 2.1 Sơ đồ được.....	4
Hình 2.2 Quy trình cận lâm sàng.....	6
Hình 2.3 Đăng ký.....	7
Hình 2.4 Đăng ký Github .....	8
Hình 2.5 Dịch vụ Github .....	8
Hình 2.6 Chọn chức năng.....	9
Hình 2.7 Bắt đầu tạo project.....	9
Hình 2.8 Tạo project.....	10
Hình 2.9 Code giao diện thêm, sửa, xóa, hủy chức danh.....	10
Hình 2.10 Giao diện chức vụ.....	11
Hình 2.11 Mô phỏng cấu trúc JHipster .....	11
Hình 2.12 Tạo project với JHipster .....	13
Hình 2.13 Mô phỏng Microservices .....	14
Hình 2.14 Mô hình Microservices.....	15
Hình 2.15 Ví dụ sơ mô mô hình Microservice .....	18
Hình 2.16 Mẫu Kiến trúc Microservices .....	19
Hình 3.1 Mô hình cơ sở dữ liệu.....	23
Hình 3.2 Giao diện chính.....	23
Hình 3.3 Giao diện xem chi tiết.....	24
Hình 3.4 Giao diện thêm mới đơn đề xuất .....	24
Hình 3.5 Giao diện sửa thông tin đơn đề xuất.....	25
Hình 3.5 Giao diện xóa đơn đề xuất.....	25

## MỞ ĐẦU

Với sự phát triển khoa học và công nghệ vào những năm gần đây, những công nghệ mới phát triển rầm rộ đã đem lại nhiều tiện ích cho cuộc sống, nó được ứng dụng rộng rãi ở nhiều lĩnh vực, đặc biệt là lĩnh vực thông tin truyền thông. Hệ thống thông tin điện tử, trực tuyến, các Website của các tổ chức, đơn vị, doanh nghiệp được phát triển mạnh mẽ, góp phần hỗ trợ quản lý và điều hành các công việc một cách linh hoạt, nhanh chóng và chính xác.

Ở các bệnh viện lớn, công việc quản lý suất ăn cho bệnh nhân là vô cùng quan trọng đòi hỏi sự nhanh chóng, chính xác và khả năng lưu trữ với số lượng lớn vì thế việc áp dụng Công nghệ thông tin vào hệ thống y tế đã trở thành vấn đề tất yếu phải thực hiện. Đồng thời, việc áp dụng công nghệ thông tin sẽ khắc phục được mọi khuyết điểm của việc lưu trữ thủ công.

## **PHẦN 1: TỔNG QUAN VỀ CÔNG TY THỰC TẬP**

### **1.1 Giới thiệu chung**

Công ty Công nghệ thông tin VNPT (Tên viết tắt: VNPT-IT) được thành lập theo Quyết định số 39/QĐ-VNPT-HĐTV-NL ngày 01 tháng 03 năm 2018 của Chủ tịch Tập đoàn Bưu chính Viễn thông Việt Nam, trên cơ sở tổ chức lại các nhiệm vụ và nguồn lực công nghệ thông tin thuộc Tập đoàn.

VNPT-IT hoạt động trong lĩnh vực nghiên cứu phát triển, tích hợp các sản phẩm dịch vụ công nghệ thông tin phục vụ nội bộ Tập đoàn Bưu chính Viễn thông Việt Nam (VNPT) và các khách hàng bên ngoài VNPT (bao gồm cả khách hàng Quốc tế).

VNPT-IT đặt mục tiêu xây dựng một hệ sinh thái tích hợp trọn gói các sản phẩm, dịch vụ công nghệ thông tin và Internet lớn nhất Việt Nam, từ đó mang sản phẩm - dịch vụ của chúng tôi đến với thị trường quốc tế. Để làm được điều này, VNPT-IT đề ra chiến lược phát triển xoay quanh 4 giá trị cốt lõi:

- Con người là chìa khóa
- Khách hàng là trung tâm
- Sáng tạo không ngừng
- Đối tác đáng tin cậy

### **1.2 Lĩnh vực kinh doanh**

- Tổ chức nghiên cứu, phát triển, sản xuất, kinh doanh các sản phẩm, dịch vụ công nghệ thông tin để cung cấp cho nội bộ VNPT và khách hàng bên ngoài VNPT.
- Đầu tư, phát triển, quản lý tài sản các hệ thống, nền tảng công nghệ thông tin; vận hành khai thác hệ thống điều hành sản xuất kinh doanh, đảm bảo an toàn, bảo mật thông tin cho các sản phẩm và các dịch vụ công nghệ thông tin của VNPT cung cấp cho khách hàng.
- Trong mô hình của Công ty VNPT-IT gồm: Ban Tổng giám đốc, Văn phòng các Ban chức năng cùng các Trung tâm trực thuộc và các Trung tâm tại: Hà Nội, Hồ Chí Minh, Đà Nẵng, Hải Phòng và Tiền Giang. Các Trung tâm này là đơn vị hạch toán phụ thuộc của Công ty.
- Là một trong những Đơn vị chủ chốt của Tập đoàn Bưu chính Viễn thông Việt Nam, VNPT-IT luôn phấn đấu không ngừng nâng cao chất lượng sản phẩm dịch vụ về mọi mặt để trở thành thương hiệu có uy tín trong lĩnh vực công nghệ thông tin, góp phần đưa VNPT đạt mục tiêu trở thành Tập đoàn Viễn thông - CNTT hàng đầu quốc gia, giữ vai trò chủ đạo trong lĩnh vực Viễn thông và CNTT Việt Nam.

### 1.3 Logo



Hình 1.1 Logo

Logo VNPT mô phỏng chuyển động của vệ tinh xoay quanh địa cầu, vẽ nên hình chữ V là chữ cái đầu tiên trong tên viết tắt VNPT. Sự uyển chuyển của hình khối kết hợp ngôn ngữ âm dương thể hiện sự vận động không ngừng của thông tin, sự bền vững cùng sự hội nhập thế giới với khoa học và công nghệ hiện đại.



## **PHẦN 2: NỘI DUNG THỰC HIỆN**

### **2.1 Tìm hiểu quy trình khám chữa bệnh**

#### **2.1.1 Tìm hiểu kho Dược**

- Kho dược CTMTQG: là kho quản lý “thuốc”, “vật tư tiêu hao”, “hoá chất” mà bệnh viện không phải trả phí cho nhà cung cấp.

- Kho dược chính: Là kho quản lý “thuốc”, “vật tư tiêu hao”, “hoá chất” theo giá có nghĩa là “Giá nhập” “Giá xuất” phải bằng nhau và bệnh viện khi mua phải thanh toán tiền cho nhà cung cấp.

- Kho Thanh Lý: Kho dùng chứa thuốc, vật tư, hóa chất được thanh lý do hết hạn sử dụng, bị mất, hỏng, vỡ...

- Kho BHYT: Chức năng giúp “khoa Dược” thống kê tình hình xuất thuốc cho bệnh nhân có BHYT khám ngoại trú.

- Kho thuốc nội trú: Quản lý lưu lượng thuốc cũng như điều phối thuốc cho các khoa trong quá trình điều trị đồng thời còn có chức năng chuyển kho cho các tủ thuốc ở các khoa.

Kho VTTH: Quản lý “vật tư tiêu hao” cũng như điều phối “vật tư tiêu hao” cho các khoa và chuyển kho cho các tủ VTTH tại các khoa.

-Kho cận lâm sàng: Xuất cho bệnh nhân theo định mức của từng chỉ định cận lâm sàng cụ thể.

-Tủ thuốc khoa, tủ VTTH khoa : Tủ thuốc của từng khoa, bệnh viện có bao nhiêu khoa thì có bấy nhiêu tủ thuốc

- Nhập kho là khi bệnh viện có yêu cầu cần thêm “thuốc”, “vật tư tiêu hao”, “hoá chất” từ nhà cung cấp vào kho của bệnh viện bao gồm kho dược chính và kho dược CTMTQG.

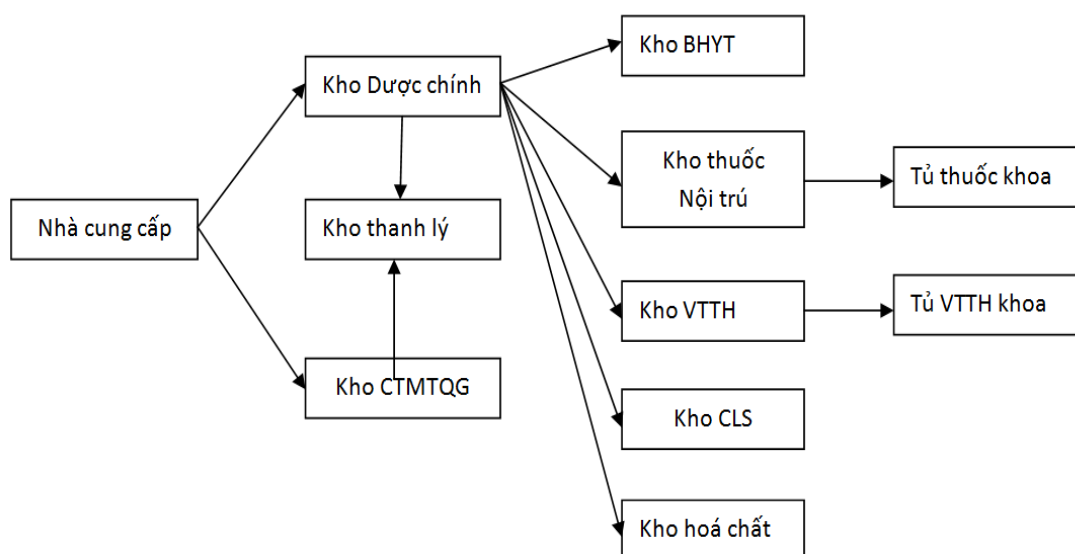
- Chuyển kho là: chuyển giao “thuốc”, “vật tư tiêu hao”, “hoá chất” giữa các kho trong bệnh viện đó.

- Phương thức chuyển kho :

- Kho A yêu cầu chuyển vật tư từ kho B ở ngày X
- Kho B duyệt yêu cầu chuyển kho từ kho A vào ngày Y => lúc này hệ thống sẽ trừ kho B và hiển thị xuất nhập tồn ở kho B là đã trừ đi lượng dược/vật tư đã duyệt ở đơn yêu cầu chuyển vật tư.Nhưng lượng dược/vật tư này chưa được cộng dồn vào kho A.

- Kho A nhận được/ vật tư về kho ở ngày Z => lúc này hệ thống sẽ cộng thêm được/ vật tư vào kho A.
- Phương thức trừ kho nội trú: kho nội trú ra thuốc từ hai nguồn đó là : “tủ trực khoa” và “kho thuốc nội trú”:
- Khi ra thuốc từ tủ trực khoa sẽ trừ kho ngay ngày ra thuốc.
  - Khi ra thuốc từ kho thuốc nội trú của khoa Được vào ngày X, khoa nội trú sẽ tổng hợp dự trừ đến khoa Được vào Y.
  - Khoa Được duyệt giao thuốc hoàn trả vào ngày Z, thì sẽ trừ kho vào ngày khoa được duyệt thuốc.
- Ghi nhớ hai ngày trừ kho trong phiếu chuyển kho :
- Đối với kho Yêu Cầu: ngày nhập được về kho.
  - Đối với kho Cấp Phát: ngày duyệt yêu cầu.
- Chức năng Được Khoa-Phòng dự trừ:Mục đích là dự trừ được từ đơn vị tuyệt dưới lên đơn vị quản lý – ĐVQL
- Chức năng Được Khoa-Phòng hoàn trả: Mục đích là hoàn trả được từ kho được tuyến dưới lên đơn vị tuyến trên – ĐVQL
- Tồn kho:
- + Tồn kho thực : là số lượng được tồn kho có thực tế có trong kho.
  - + Tồn kho ảo: là số lượng được tồn kho trên sổ sách .

### 2.1.2 Sơ đồ kho Được



Hình 2.1 Sơ đồ kho được

### 2.1.3 Cận Lâm Sàng

#### - Cận lâm sàng nội trú

- Bác sĩ lập phiếu điều trị nội trú, chỉ định cận lâm sàng trong phiếu điều trị.
- Y/Bác sĩ/Kỹ thuật viên thực hiện cận lâm sàng kiểm tra phiếu điều trị và thực hiện cận lâm sàng cho bệnh nhân. Sau đó trả kết quả cận lâm sàng cho bệnh nhân.

#### - Cận lâm sàng bệnh án ngoại trú

- Sau khi bác sĩ chỉ định cận lâm sàng thì sẽ in phiếu chỉ định để bệnh nhân đi thực hiện. Y/Bác sĩ/Kỹ thuật viên cận lâm sàng kiểm tra phiếu chỉ định của bệnh nhân và thực hiện cận lâm sàng cho bệnh nhân. Sau đó trả kết quả cận lâm sàng cho bệnh nhân và cập nhật vào chương trình.

#### - Các tình huống xử trí khám bệnh ngoại chẩn

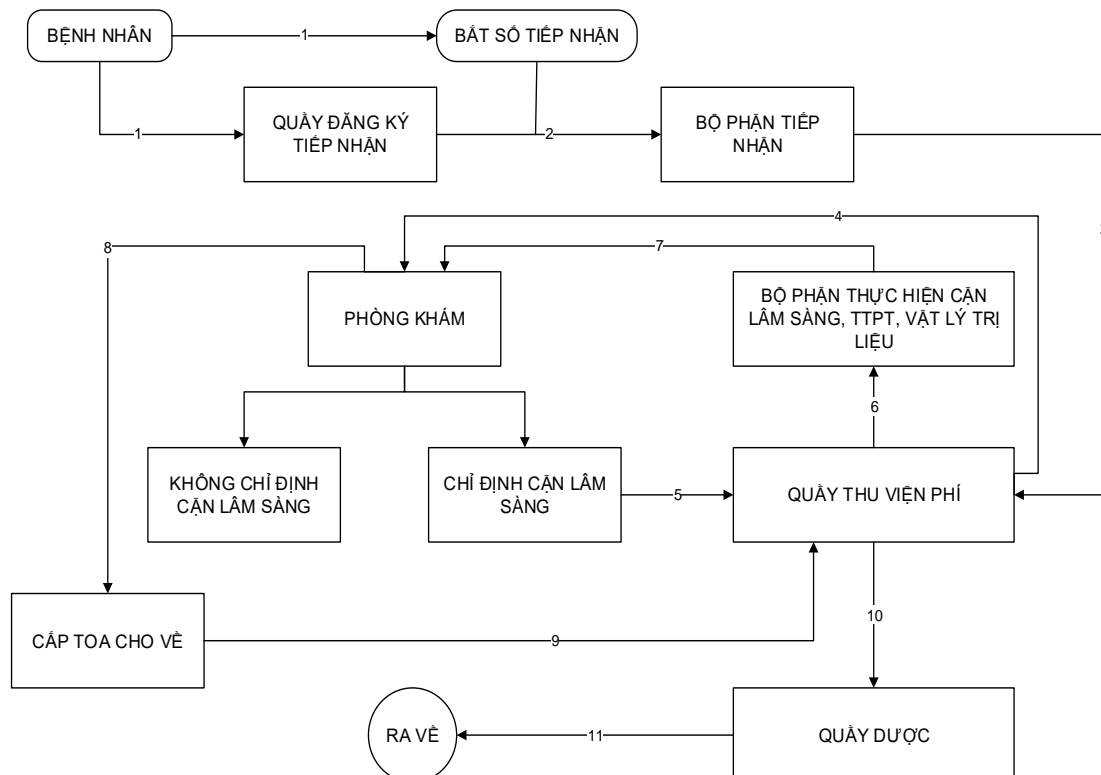
Trường hợp bác sĩ không chỉ định cận lâm sàng, TTPT, vật lý trị liệu

- Bác sĩ khám bệnh cho bệnh nhân, ra chẩn đoán, in giấy chuyển viện (nếu bệnh nhân có nhu cầu chuyển viện lên tuyến cao hơn).
- Bác sĩ khám bệnh cho bệnh nhân, chẩn đoán, chuyển phòng khác khám tiếp.
- Bác sĩ khám bệnh cho bệnh nhân, chẩn đoán, cấp toa thuốc ra về và hẹn tái khám, in.
- Bác sĩ khám bệnh cho bệnh nhân, chẩn đoán, cho nhập viện, lấy số nhập viện, bệnh nhân vào điều trị nội trú (phân hệ nội trú của hệ thống).
- Bác sĩ khám bệnh cho bệnh nhân, chẩn đoán, cho điều trị bệnh án ngoại trú, lấy số nhập viện bệnh án ngoại trú, bệnh nhân vào điều trị bệnh án ngoại trú (phân hệ bệnh án ngoại trú của hệ thống).
- Bác sĩ chỉ định cận lâm sàng, TTPT, vật lý trị liệu
  - Bác sĩ khám bệnh cho bệnh nhân, chỉ định cận lâm sàng (bệnh nhân thực hiện cận lâm sàng xong trở về phòng khám), chẩn đoán, chuyển viện, in giấy chuyển viện. lưu ý khi in các phiếu chỉ định cận lâm sàng cần in riêng bhyt chi và không chi.
  - Bác sĩ khám bệnh cho bệnh nhân, chỉ định cận lâm sàng (bệnh nhân thực hiện cận lâm sàng xong trở về phòng khám), chẩn đoán, cấp toa thuốc ra về.
  - Bác sĩ khám bệnh cho bệnh nhân, chỉ định cận lâm sàng (bệnh nhân thực hiện cận lâm sàng xong trở về phòng khám), chẩn đoán, cấp toa thuốc ra về và hẹn tái khám.
  - Bác sĩ khám bệnh cho bệnh nhân, chỉ định cận lâm sàng (bệnh nhân thực hiện cận lâm sàng xong trở về phòng khám), chẩn đoán, cho nhập viện, tạo phiếu nhập viện cho bệnh nhân, bệnh nhân vào điều trị nội trú

(phân hệ nội trú của hệ thống, kết quả cận lâm sàng có thể được sử dụng lại ở phân hệ nội trú).

- Bác sĩ khám bệnh cho bệnh nhân, chỉ định thủ thuật phẫu thuật, thực hiện thủ thuật phẫu thuật, chẩn đoán, cấp toa thuốc ra về.
- Bác sĩ khám bệnh cho bệnh nhân, chỉ định thủ thuật phẫu thuật, thực hiện thủ thuật phẫu thuật, chẩn đoán, cấp toa thuốc ra về và hẹn tái khám.
- Bác sĩ khám bệnh cho bệnh nhân, chỉ định thủ thuật phẫu thuật, thực hiện thủ thuật phẫu thuật, chẩn đoán, cho nhập viện, tạo phiếu nhập viện cho bệnh nhân, bệnh nhân vào điều trị nội trú (phân hệ nội trú của hệ thống, kết quả thủ thuật, phẫu thuật có thể được sử dụng lại ở phân hệ nội trú).
- Bác sĩ khám bệnh cho bệnh nhân, chỉ định thủ thuật phẫu thuật, thực hiện thủ thuật phẫu thuật, chẩn đoán, cho điều trị bệnh án ngoại trú, tạo phiếu nhập viện bệnh án ngoại trú cho bệnh nhân, bệnh nhân vào điều trị bệnh án ngoại trú.

• Quy trình cận lâm sàng:



Hình 2.2 Quy trình cận lâm sàng

## 2.2 Làm quen với các công cụ, ứng dụng hỗ trợ cơ bản.

### 2.2.1 GitHub

GitHub là một dịch vụ lưu trữ trên web dành cho các dự án có sử dụng hệ thống kiểm soát Git revision. GitHub cung cấp chức năng social networking như feeds, followers và network graph để các Developer học hỏi kinh nghiệm làm việc thông qua lịch sử commit. Nếu comment dùng để mô tả chức năng của đoạn code, thì commit message trên Git dùng để mô tả hành động mà Dev vừa thực hiện trên code. Một tài khoản GitHub với nhiều repositories và tham gia vào những project đa dạng khác nhau đem đến cho người dùng nhiều lợi. Github đã nâng tầm kỹ năng và cả sự nghiệp của tôi cùng người dùng bè tôi lên một mức vượt bậc. Tôi sẽ chia sẻ câu chuyện của mình cùng 3 lợi ích máu chó mà GitHub đã mang lại cho tôi (và sẽ mang lại cho người dùng nếu người dùng cũng có 1 tài khoản GitHub).

Github đã trở thành một yếu tố có sức ảnh hưởng trong cộng đồng phát triển mã nguồn mở. Thậm chí nhiều nhà phát triển đã bắt đầu xem nó là một sự thay thế cho sơ yếu lý lịch và một số nhà tuyển dụng yêu cầu các ứng viên cung cấp một liên kết đến tài khoản Github để đánh giá ứng viên.

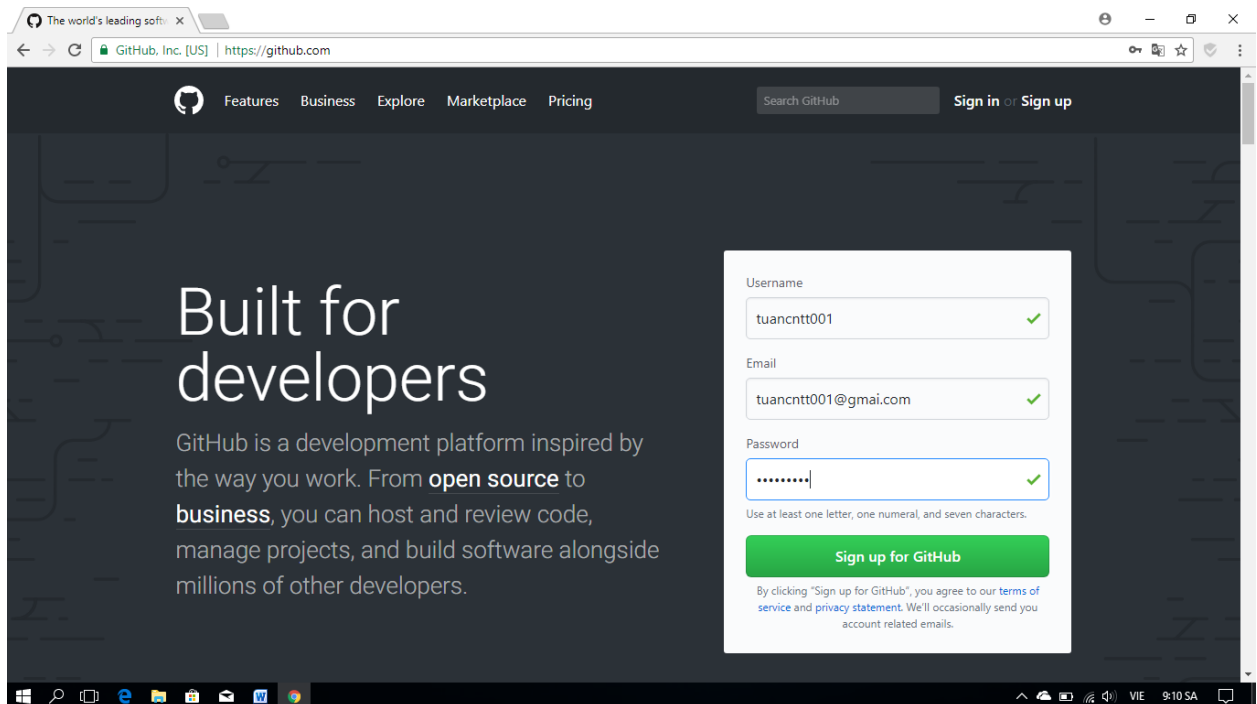
Thực hiện:

Để tạo tài khoản GitHub truy cập vào trang <https://github.com/> sau đó chọn nút Sign up để đăng ký tài khoản GitHub.



Hình 2.3 Đăng ký

Tiếp theo người dùng điền các thông tin cần thiết và chọn Create an account để tạo tài khoản.



Hình 2.4 Đăng ký Github

Người dùng có thể chọn hình thức trả phí để sử dụng hết chức năng mà GitHub mang lại.

**Completed**  
Set up a personal account

**Step 2:**  
Choose your plan

**Step 3:**  
Tailor your experience

**Choose your personal plan**

☒

Unlimited public repositories for free.

☐

Unlimited private repositories for \$7/month. [\(view in VND\)](#)

Don't worry, you can cancel or upgrade at any time.

☐

**Help me set up an organization next**  
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.  
[Learn more about organizations.](#)

**Continue**

**Both plans include:**

- ✓ Collaborative code review
- ✓ Issue tracking
- ✓ Open source community
- ✓ Unlimited public repositories
- ✓ Join any organization

Hình 2.5 Dịch vụ Github

## Tiếp tục chọn Submit

✓ Completed Set up a personal account	📁 Step 2: Choose your plan	⚙️ Step 3: Tailor your experience
--	-------------------------------	--------------------------------------

How would you describe your level of programming experience?

☐ Totally new to programming    ☐ Somewhat experienced    ☐ Very experienced

What do you plan to use GitHub for? (check all that apply)

☐ School projects    ☐ Project Management    ☐ Design  
☐ Research    ☐ Development    ☐ Other (please specify)

Which is closest to how you would describe yourself?

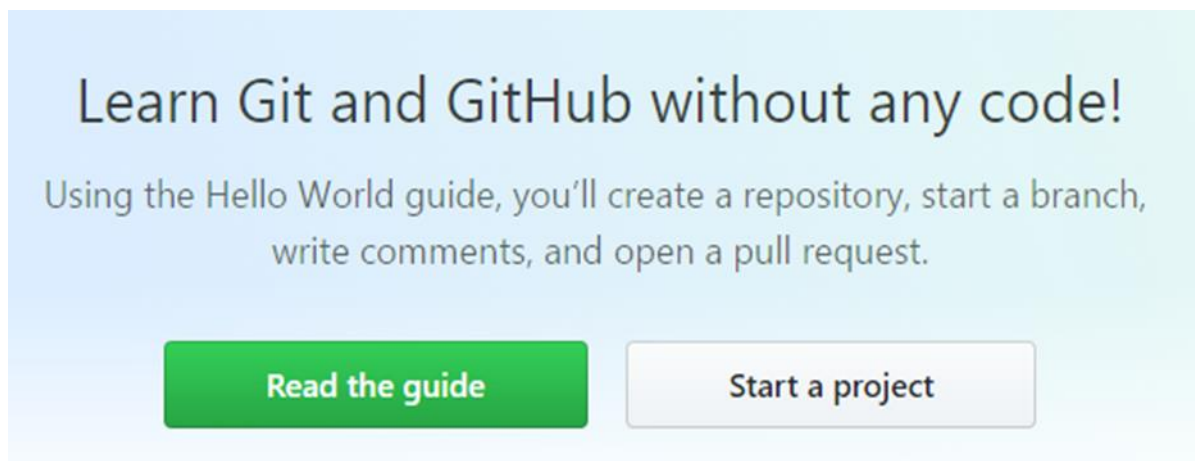
☐ I'm a professional    ☐ I'm a hobbyist    ☐ I'm a student  
☐ Other (please specify)

What are you interested in?

e.g. tutorials, android, ruby, web-development, machine-learning, open-source

[Submit](#) [skip this step](#)

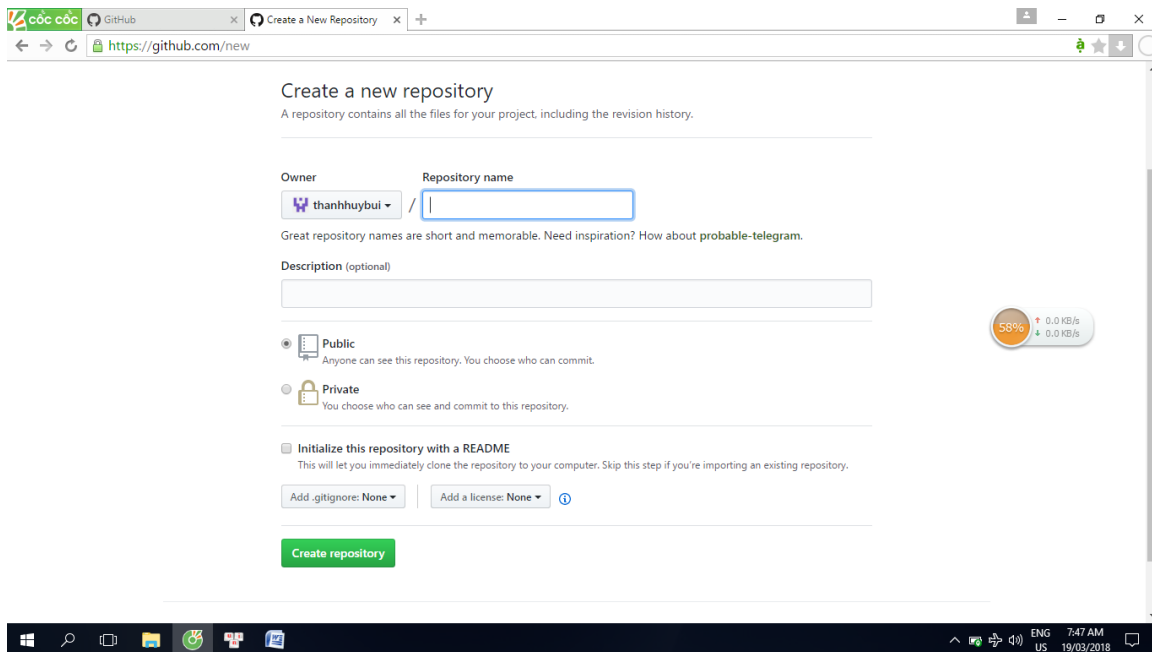
Hình 2.6 Chọn chức năng



Chọn [Start a project](#) để vào GitHub.

Hình 2.7 Bắt đầu Project

## Tạo Repository làm việc

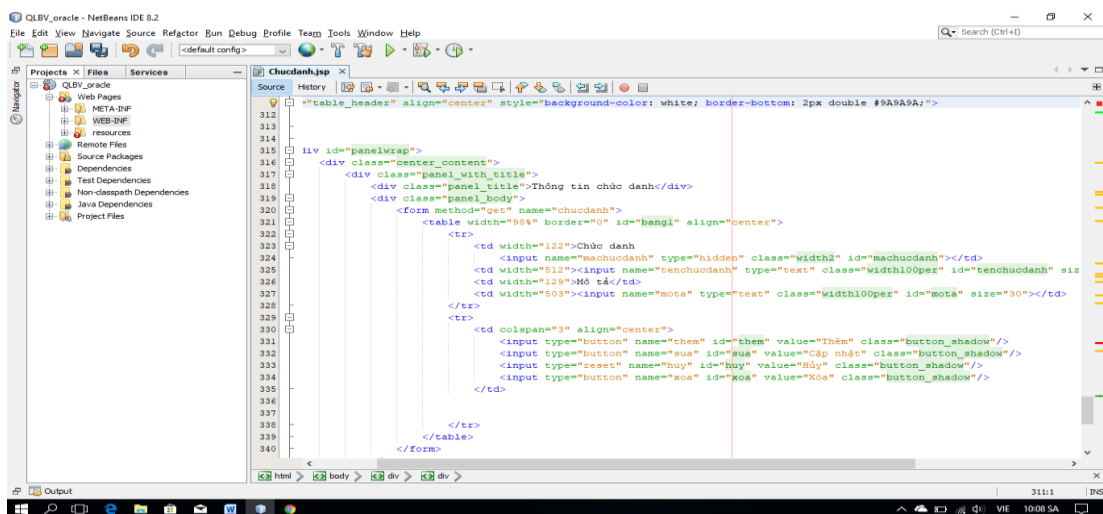


Hình 2.8 Tạo Project

### 2.2.2 JavaSpring MVC

Spring Framework là một bộ khung ứng dụng và bộ chứa đảo ngược điều khiển cho nền tảng Java, là một application framework mã nguồn mở, được giới thiệu vào năm 2002.

### Cấu trúc code của dự án HIS



Hình 2.9 Code giao diện thêm, sửa, xóa, hủy chức danh



Thông tin chức danh

Chức danh

Mô tả

Thêm

Cập nhật

Xóa

Hủy

Danh sách chức danh

Chức danh	Mô tả (viết tắt)
ddda	ddd
Bác sĩ chuyên khoa sản phụ khoa	PS
Bác sĩ chuyên khoa nội tổng hợp	NTH
Điều Dưỡng Cao Đẳng	ĐD CĐ
Y Sĩ Đa Khoa	YS DK
Bác sĩ chuyên khoa nhi	BS CK NHI
Test Chức Danh	TESTCD
Test Chức Danh	TCD
CHUNG	.
CĐCNSH	CĐCNSH
Cử nhân luật	CNL
Cao Đẳng Điều Dưỡng	CDĐD
Nhân viên kỹ thuật	NVKT
KTV trung cấp	KTVTC
Kế toán trung cấp	KTTC
Y công	YC

Page 1 of 5 20

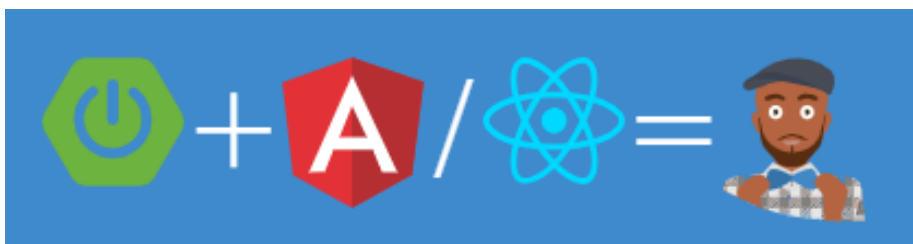
Hình 2.10: Giao diện chức vụ

### 2.2.3 Giới thiệu về JHipster stack

Câu chuyện tiếp cận hay học một công nghệ mới chưa bao giờ hết hot. Nếu người dùng đang loay hoay để bắt đầu khởi tạo project đầu tiên với Spring Boot thì JHipster là giải pháp cho người dùng. JHipster không đơn thuần cho người dùng project có Spring Boot, nó còn phù hợp để người dùng bắt đầu tập tành với Angular hay React với CRUD từ backend đến frontend và nhiều hơn thế.

#### Mô tả tổng quan Jhipster

JHipster là một nền tảng để tạo, phát triển và triển khai các ứng dụng Spring Boot + Angular / React Web và Spring microservices.



Hình 2.11 Mô phỏng cấu trúc JHipster

Nói một cách đơn giản, Jhipster(viết tắt của Java Hipster) là cách đơn giản để chúng ta tạo ra một project xung quanh những công nghệ được ưa thích nhất với Spring technologies và Angular/React. Khi chúng ta bắt đầu dự án chúng ta sẽ quan tâm đến 3 khía cạnh:

- Server side stack sẽ trông như thế nào?
- Client side stack sẽ trông như thế nào?

- Làm sao để chúng ta có thể deploy project của chúng ta?

### **Server side**

Khi chúng ta bắt đầu build phần backend có những câu hỏi mà chúng ta quan tâm đó là:

- Ngôn ngữ chúng ta lựa chọn là gì?
- Tầng dữ liệu sẽ như thế nào?
- Hệ thống sẽ bảo mật ra sao?
- Khả năng bảo trì và mở rộng hệ thống?
- Cách cung cấp API document?
- Kiểm thử ứng dụng thế nào? Câu trả lời sẽ có khi người dùng nhìn vào danh sách công nghệ mà JHipster cung cấp :

### **Client side**

Với những framework frontend mạnh mẽ

Deployment

Deploy dự án dễ dàng

### **Lí do nên sử dụng Jhipster**

Với những gì đã nêu ở trên, tôi hy vọng các người dùng đã biết lý do vì sao nên chọn JHipster. Chúng ta sẽ dễ dàng có được một project đủ mạnh mẽ đầy đủ những thứ cơ bản để bắt đầu với thời gian nhanh nhất. Và nếu người dùng đang tập tành với Spring để trở thành 1 Java Web developer thì người dùng nên quan tâm đến công nghệ này.

### **Tạo project đầu tiên với JHipster**

Những gì người dùng cần có để bắt đầu tạo một project đầu tiên với JHipster:

- Cài đặt Java 8 Oracle website
- Cài đặt Node.js Node.js website
- Cài đặt Yeoman: `npm install -g yo`
- Cài đặt JHipster: `npm install -g generator-jhipster`

### **Note:**

Người dùng cũng có thể sử dụng Yarn/Homebrew/Chocolatey/Docker để cài đặt JHipster.

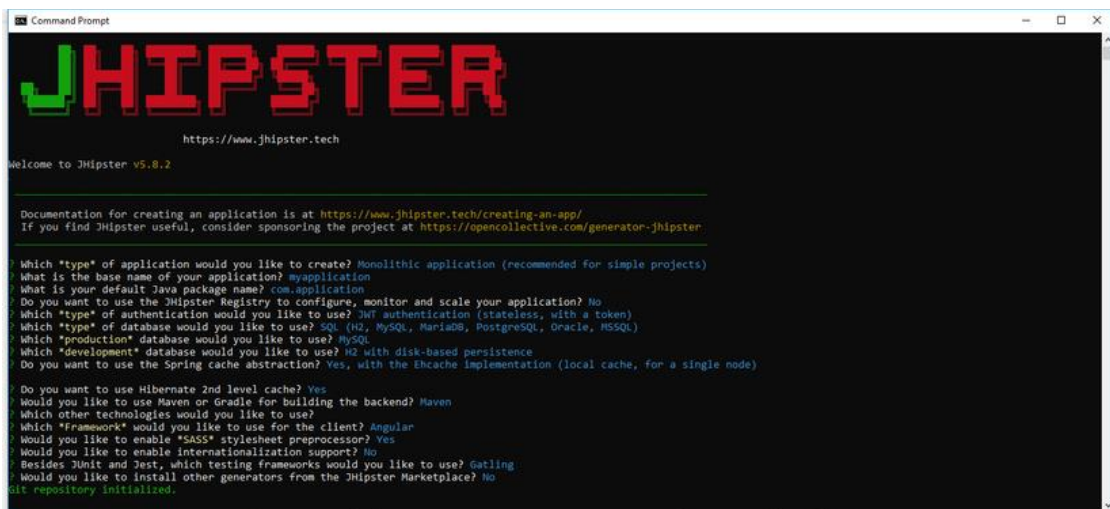
Với bước cài đặt JHipster ở trên nếu người dùng muốn sử dụng phía Client với React thì bản JHipster của người dùng phải là < 5. Khi đó người dùng có thể chạy lệnh sau:

```
npm install -g generator-jhipster@4.14.3
```

## Tạo project:

Thực hiện trên Terminal/cmd:

1. Tạo 1 thư mục trống là nơi sẽ chứa project: `mkdir myapplication`
2. Chuyển đến thư mục vừa tạo: `cd myapplication`
3. Generate ứng dụng: `jhipster`
4. Lựa chọn những thứ phù hợp với project của người dùng



Hình 2.12 Tạo project với JHipster

Bây giờ người dùng đã có 1 project với:

- Backend: Spring Boot + Spring Security
- Database:
  - Mysql (production)
  - H2 with disk-based (development)
- Frontend: Bootstrap + SASS + React (hoặc thấp hơn tùy vào phiên bản JHipster người dùng cài đặt ở trên) Sau khi chạy xong người dùng sẽ nhận được kết quả như bên dưới. Đó cũng là hướng dẫn để người dùng build project của mình trên local.

Chạy Spring Boot application:

`./mvnw` (mvnw if using Windows Command Prompt)

Client application generated successfully. Start your Webpack development server with: `npmstart`

## Những lưu ý khi chọn trong phần config project ở trên

Jhipster hỗ trợ người dùng setup đa ngôn ngữ trong project

- JHipster đã tạo cho người dùng project với cả môi trường của development và môi trường thực tế (production)
- Ở môi trường development: nếu người dùng không muốn cài đặt các database trên local.
- Người dùng có thể sử dụng H2 with disk-based hoặc H2 in -memory. Cả 2 cho phép người dùng sử dụng database ngay trong giao diện của ứng dụng.
- H2 in-memory: data sẽ bị mất khi người dùng restart server.
- H2 with disk-based: data sẽ không bị mất khi người dùng restart server.
- Nếu không dùng 2 loại database trên người dùng sẽ phải tạo schema trên local và sửa lại config với database trong phần code.

Ví dụ người dùng sử dụng Mysql.

Trong thư mục project:

src/main/resources/config/application-dev.yml

datasource:

type: com.zaxxer.hikari.HikariDataSource

url:

jdbc:mysql://localhost:3306/hello?useUnicode=true&characterEncoding=utf8&useSSL=false&useLegacyDatetimeCode=false&serverTimezone=UTC

username: root

password:

Thì hello sẽ là tên schema mà người dùng cần để tạo trên local.

JHipster giúp người dùng với 1 dòng lệnh có đầy đủ CRUD cả phần backend và frontend Người dùng hãy thử tìm hiểu nó trên doc của JHipster

Tổng kết

Trên đây mình đã giới thiệu cho các người dùng cách để tạo và run một project với jhipster. Hãy thử tạo nó và "nghịch" nó, người dùng sẽ thích nó sớm thôi. Mình sẽ chia sẻ vào cụ thể về nó hơn trong những bài viết sau!

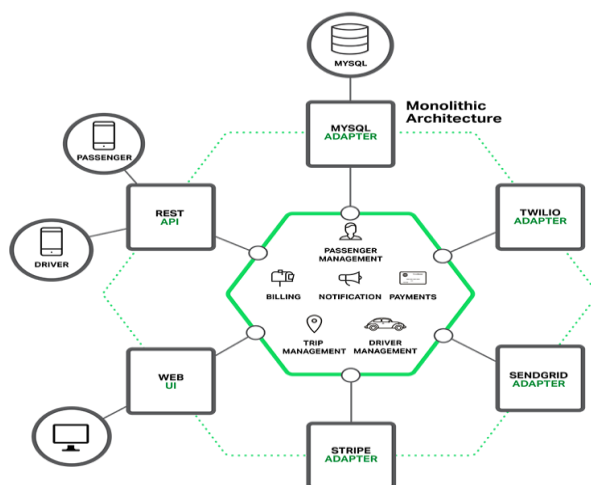
## 2.2.4 Giới thiệu về Microservices



Hình 2.13 Mô phỏng Microservices

Microservices hiện đang nhận được rất nhiều sự chú ý: bài viết, blog, thảo luận trên phương tiện truyền thông xã hội và thuyết trình hội nghị. Họ đang nhanh chóng hướng tới đỉnh cao của kỳ vọng tăng cao trên chu kỳ Gartner Hype. Đồng thời, có những người hoài nghi trong cộng đồng phần mềm bác bỏ microservices như không có gì mới. Naysayers cho rằng ý tưởng chỉ là một thương hiệu của SOA. Tuy nhiên, mặc dù cả sự cường điệu và hoài nghi, Kiến trúc Microservices đều có những lợi ích đáng kể – đặc biệt khi nói đến việc cho phép phát triển nhanh và phân phối các ứng dụng doanh nghiệp phức tạp.

Hãy tưởng tượng rằng người dùng đang bắt đầu xây dựng một ứng dụng taxi mới toanh có thương hiệu để cạnh tranh với Uber và Hailo. Sau một số cuộc họp sơ bộ và các yêu cầu thu thập, người dùng sẽ tạo một dự án mới từ đầu hoặc bằng cách sử dụng Rails, Spring Boot, Play hoặc Maven. Ứng dụng mới này sẽ có kiến trúc lục giác (hexagonal architecture) kiểu mô-đun, như trong diagram sau:



Hình 2.14 Mô hình sơ đồ Microservices

Core ứng dụng là business logic, được thực hiện bởi các mô-đun có các services, domain objects, and events. Xung quanh lõi là các adapters giao tiếp với thế giới bên ngoài. Ví dụ về adapters bao gồm các database access components, messaging components produce và consume các messages, và các web components cung cấp các API hoặc hiển thị giao diện người dùng.

Mặc dù có một kiến trúc mô-đun hợp lý, ứng dụng được đóng gói và triển khai như một khối nguyên khối. Định dạng thực tế phụ thuộc vào ngôn ngữ và khuôn khổ của ứng dụng. Ví dụ, nhiều ứng dụng Java được đóng gói như các tệp tin WAR và được triển khai trên các máy chủ ứng dụng như Tomcat hoặc Jetty. Các ứng dụng Java khác được đóng gói như các executable JARs.

Các ứng dụng được viết theo phong cách này là cực kỳ phổ biến. Chúng đơn giản để phát triển vì IDE của chúng ta và các công cụ khác tập trung vào việc xây dựng một ứng dụng đơn lẻ. Các loại ứng dụng này cũng đơn giản để test. Người dùng có thể thực hiện end-to-end testing bằng cách đơn giản khởi chạy ứng dụng và kiểm tra giao diện người dùng bằng Selenium. Các ứng dụng nguyên khối cũng đơn giản để triển khai. Người dùng chỉ cần copy ứng dụng đã được build, đóng gói vào máy chủ. Người dùng cũng có thể scale ứng dụng bằng cách chạy nhiều bản sao với load balancer. Trong giai đoạn đầu của dự án, nó hoạt động tốt.

Thật không may, cách tiếp cận đơn giản này có một giới hạn rất lớn. Các ứng dụng thành công có thói quen phát triển theo thời gian và cuối cùng trở nên rất lớn. Trong mỗi lần chạy nước rút, nhóm phát triển của người dùng thực hiện thêm một vài user stories, điều đó, tất nhiên, có nghĩa là thêm nhiều dòng code. Sau một vài năm, ứng dụng nhỏ, đơn giản của người dùng sẽ phát triển thành một khối đá khổng lồ. Để đưa ra một ví dụ cực đoan, gần đây tôi đã nói chuyện với một developer đang viết một công cụ để phân tích sự phụ thuộc giữa hàng nghìn gói JAR trong ứng dụng có hàng triệu dòng code (LOC) của họ.

Một khi ứng dụng của người dùng đã trở thành một khối lớn, phức tạp, công việc phát triển phần mềm của người dùng có lẽ là trong một thế giới của đau khổ. Bất kỳ nỗ lực phát triển nhanh và giao hàng sẽ không còn dễ dàng. Một vấn đề lớn là ứng dụng quá phức tạp. Nó chỉ đơn giản là quá lớn đối với bất kỳ developer nào để hiểu đầy đủ. Kết quả là, sửa lỗi và triển khai các tính năng mới một cách chính xác sẽ trở nên khó khăn và tốn thời gian. Hơn nữa, điều này tạo ra một vòng xoáy cực độ. Nếu

codebase khó hiểu thì các thay đổi sẽ không được thực hiện chính xác. Người dùng sẽ kết thúc với một quả bóng bùn khổng lồ (big ball of mud), không thể hiểu nổi, không thể cứu vãn.

Một vấn đề khác với một ứng dụng nguyên khối phức tạp là sẽ gây trở ngại cho việc triển khai liên tục (continuous deployment). Ngày nay, với các ứng dụng SaaS, chúng thường được push những thay đổi vào production nhiều lần trong ngày. Điều này cực kỳ khó thực hiện với một khối nguyên khối phức tạp vì người dùng phải triển khai lại toàn bộ ứng dụng để cập nhật bất kỳ phần nào của nó. Thời gian khởi động kéo dài mà tôi đã đề cập trước đó cũng không giúp được gì. Ngoài ra, vì tác động của thay đổi thường không được hiểu rõ, có khả năng người dùng phải thực hiện manual testing trên diện rộng (regression test). Do đó, việc triển khai liên tục là không thể thực hiện được.

Các ứng dụng nguyên khối cũng có thể khó mở rộng khi các mô-đun khác nhau có các yêu cầu xung đột về tài nguyên. Ví dụ, một mô-đun có thể triển khai logic xử lý hình ảnh chuyên sâu của CPU và lý tưởng nhất sẽ được triển khai trong các instance của Amazon EC2 Compute Optimized. Một mô-đun khác có thể là một cơ sở dữ liệu trong bộ nhớ và phù hợp nhất với các instance EC2 Memory-optimized. Tuy nhiên, vì các mô-đun này được triển khai cùng nhau, người dùng phải thỏa hiệp về lựa chọn phần cứng.

Một vấn đề khác với các ứng dụng nguyên khối là độ tin cậy. Bởi vì tất cả các mô-đun đang chạy trong cùng một process, một lỗi trong bất kỳ mô-đun nào, chẳng hạn như rò rỉ bộ nhớ, có khả năng có thể làm down toàn bộ process. Hơn nữa, vì tất cả các trường hợp của ứng dụng đều giống hệt nhau, lỗi đó sẽ ảnh hưởng đến tính khả dụng của toàn bộ ứng dụng.

Cuối cùng nhưng không kém phần quan trọng, các ứng dụng nguyên khối làm cho việc áp dụng các frameworks and languages mới trở nên vô cùng khó khăn. Ví dụ, hãy tưởng tượng rằng người dùng có 2 triệu dòng mã được viết bằng cách sử dụng framework XYZ. Nó sẽ cực kỳ tốn kém (cả về thời gian và chi phí) để viết lại toàn bộ ứng dụng để sử dụng framework ABC mới hơn, ngay cả khi framework đó tốt hơn đáng kể. Kết quả là, có một rào cản lớn đối với việc áp dụng các công nghệ mới. Người dùng đang mắc kẹt với bất kỳ lựa chọn công nghệ nào người dùng đã thực hiện khi bắt đầu dự án.

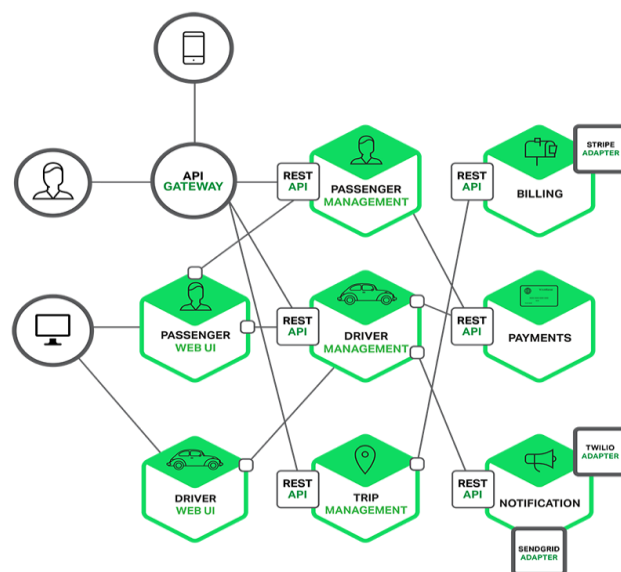
Tóm lại: người dùng có một ứng dụng đã phát triển thành một khối đá khổng lồ mà rất ít, nếu có, các developer hiểu được. Nó được viết bằng cách sử dụng công nghệ lạc hậu, không hiệu quả làm cho việc thuê developer tài năng trở nên khó khăn. Ứng dụng này khó mở rộng và không đáng tin cậy. Kết quả là việc phát triển nhanh và phân phối các ứng dụng là không thể.

### Microservices – Giải quyết sự phức tạp

Nhiều tổ chức, như Amazon, eBay và Netflix, đã giải quyết vấn đề này bằng cách áp dụng những gì bây giờ được gọi là Microservices Architecture pattern. Thay vì xây dựng một ứng dụng đơn khối, ý tưởng là chia ứng dụng của người dùng thành một tập hợp các services kết nối nhỏ hơn.

Một service thường thực hiện một tập hợp các tính năng hoặc chức năng riêng biệt, chẳng hạn như quản lý đơn hàng, quản lý khách hàng, v.v ... Mỗi microservice là một ứng dụng nhỏ có cấu trúc lục giác (hexagonal architecture) riêng bao gồm logic nghiệp vụ cùng với các adapters khác nhau. Một số microservice sẽ cung cấp API được sử dụng bởi các microservices khác hoặc bởi các ứng dụng của khách hàng. Một số Microservices khác có thể triển khai giao diện người dùng web. Khi chạy, mỗi instance thường là một cloud VM or a Docker container.

Ví dụ, hệ thống được decompose thành các microservices được mô tả trước đó được thể hiện trong diagram sau:



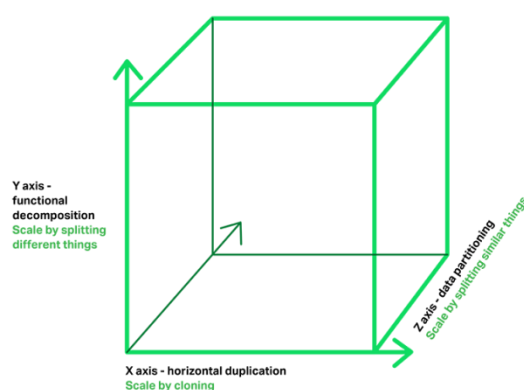
Hình 2.15 Ví dụ sơ đồ mô hình Microservice



Mỗi khu vực chức năng của ứng dụng hiện được thực hiện bởi microservice riêng của nó. Hơn nữa, các ứng dụng web được chia thành một tập hợp các ứng dụng web đơn giản hơn (ví dụ như một cho hành khách và một cho các trình điều khiển trong ví dụ taxi của chúng ta). Điều này giúp dễ dàng triển khai trải nghiệm khác biệt cho người dùng cụ thể, thiết bị hoặc trường hợp sử dụng chuyên biệt.

Mỗi dịch vụ phụ trợ cho thấy các REST API và hầu hết các dịch vụ đều consume các API do các dịch vụ khác cung cấp. Ví dụ: Driver Management sử dụng Notification server để cho tài xế đang rảnh biết về chuyến đi tiềm năng. Các dịch vụ UI gọi các dịch vụ khác để hiển thị các trang web. Các dịch vụ cũng có thể sử dụng giao tiếp không đồng bộ, dựa trên thông điệp (message-based communication). Giao tiếp giữa các dịch vụ sẽ được trình bày chi tiết hơn trong phần sau của loạt bài này.

Một số API REST cũng được tiếp xúc với các ứng dụng dành cho thiết bị di động dành cho tài xế và hành khách. Tuy nhiên, các ứng dụng không có quyền truy cập trực tiếp vào các dịch vụ phụ trợ. Thay vào đó, giao tiếp được trung gian bởi một thành phần trung gian được gọi là API Gateway. API Gateway chịu trách nhiệm về các nhiệm vụ như cân bằng tải, bộ nhớ đệm, kiểm soát truy cập, đo lường API và monitoring. Các bài viết sau trong loạt bài này sẽ trình bày về API Gateway.



Hình 2.16 Mẫu Kiến trúc Microservices

Mẫu Kiến trúc Microservices tương ứng với tỷ lệ trục Y của Scale Cube, là mô hình 3D có khả năng mở rộng từ cuốn sách tuyệt vời có tên The Art of Scalability. Hai trục tỷ lệ khác là chia tỷ lệ trục X, bao gồm chạy nhiều bản sao giống hệt nhau của ứng dụng phía sau bộ cân bằng tải và chia tỷ lệ trục Z (hoặc phân vùng dữ liệu), trong

đó thuộc tính của yêu cầu (ví dụ, khóa chính của một hàng hoặc danh tính của một khách hàng) được sử dụng để định tuyến yêu cầu tới một máy chủ cụ thể.

Các ứng dụng thường sử dụng ba loại chia tỷ lệ với nhau. Trục Y chia tỷ lệ decomposes các ứng dụng thành microservices như được hiển thị ở trên trong hình đầu tiên trong phần này. Khi chạy, quy mô trục X sẽ chạy nhiều instances của từng dịch vụ phía sau bộ cân bằng tải cho thông lượng và tính khả dụng. Một số ứng dụng cũng có thể sử dụng chia tỷ lệ trục Z để phân vùng dịch vụ. Diagram sau đây cho thấy Trip Management service có thể được triển khai với Docker chạy trên Amazon EC2 như thế nào.

Khi chạy, Trip Management service bao gồm nhiều service instances. Mỗi service instance là một Docker container. Để có tính sẵn sàng cao, các containers đang chạy trên nhiều Cloud VMs. Ở phía trước của các service instances là một bộ cân bằng tải phân phối các yêu cầu trên các cá thể. Trình cân bằng tải cũng có thể xử lý các mối quan tâm khác như caching, access control, API metering, and monitoring.

Mô hình Microservices tác động đáng kể đến mối quan hệ giữa ứng dụng và cơ sở dữ liệu. Thay vì chia sẻ một lược đồ cơ sở dữ liệu duy nhất với các dịch vụ khác, mỗi dịch vụ có lược đồ cơ sở dữ liệu riêng của nó. Một mặt, cách tiếp cận này là mâu thuẫn với ý tưởng của một mô hình dữ liệu toàn doanh nghiệp. Ngoài ra, nó thường dẫn đến trùng lặp một số dữ liệu. Tuy nhiên, có một lược đồ cơ sở dữ liệu cho mỗi dịch vụ là điều cần thiết nếu người dùng muốn hưởng lợi từ microservices, bởi vì nó đảm bảo các khớp nối lỏng lẻo(loose coupling). Diagram sau đây cho thấy database architecture cho ứng dụng ví dụ.

Mỗi dịch vụ đều có cơ sở dữ liệu riêng. Hơn nữa, một dịch vụ có thể sử dụng một loại cơ sở dữ liệu phù hợp nhất với nhu cầu của nó, cái gọi là kiến trúc bền bỉ đa điểm (polyglot persistence architecture). Ví dụ: Driver Management cần tìm tài xế gần với hành khách tiềm năng, phải sử dụng cơ sở dữ liệu hỗ trợ truy vấn địa lý hiệu quả.

Nhìn bề ngoài, mô hình Microservices tương tự như SOA. Với cả hai cách tiếp cận, kiến trúc bao gồm một tập hợp các dịch vụ. Tuy nhiên, có thể coi Microservices như là SOA nhưng không có sự thương mại hóa và dấu hiệu nhận biết về các đặc tả dịch vụ web (WS-\*) và Service Service Bus (ESB). Các ứng dụng dựa trên microservice có các giao thức đơn giản, nhẹ hơn như REST, chứ không phải WS-\*.

Microservices cũng rất tránh sử dụng ESB và thay vào đó thực hiện chức năng giống như ESB trong bản thân microservices. Mô hình Microservices cũng loại bỏ các phần khác của SOA, chẳng hạn như khái niệm lược đồ chuẩn (canonical schema).

## **PHẦN 3: KẾT QUẢ ĐẠT ĐƯỢC**

### **3.1 Lý thuyết**

#### **3.1.1 Quản lý kho dược**

Biết được cách quản lý “thuốc”, “vật tư tiêu hao”, “hoá chất” trong bệnh viện.

Hiểu được các khái niệm về các kho, tủ thuốc trong bệnh viện.

Hiểu được quy trình nhập nhập, xuất kho, quy trình về tồn kho thực, tồn kho ảo, phương thức chuyển kho như thế nào và quy tắc định ngày khi chuyển kho.

Biết thế nào là Dược – khoa phòng dự trữ, Dược – khoa phòng hoàn trả .

#### **3.1.2 Cận Lâm Sàng**

Biết được cận lâm sàng nội trú, cận lâm sàng bệnh án ngoại trú, quy trình cận lâm sàng

#### **3.1.3 Microservice**

Hiểu được khuyết điểm của việc xây dựng ứng dụng nguyên khối, phức tạp, thì công việc và nỗ lực phát triển phần mềm sẽ không hề dễ dàng. Nó chỉ đơn giản là quá lớn đối với bất kỳ developer nào để hiểu đầy đủ. Kết quả là, sửa lỗi và triển khai các tính năng mới một cách chính xác sẽ trở nên khó khăn và tốn thời gian.

Hiểu được microservices (kiến trúc nhiều dịch vụ nhỏ) – giải quyết sự phức tạp trên. Mỗi microservice là một ứng dụng nhỏ có cấu trúc lục giác (hexagonal architecture) riêng bao gồm logic nghiệp vụ cùng với các adapters khác nhau. Một số microservice sẽ cung cấp API được sử dụng bởi các microservices khác

#### **3.1.3 Jhipster (Spring boot + React)**

Hiểu được Jhipster là gì, các thành phần của Jhipster, ứng dụng những hiểu biết để xây dựng một module sử dụng JHipster.

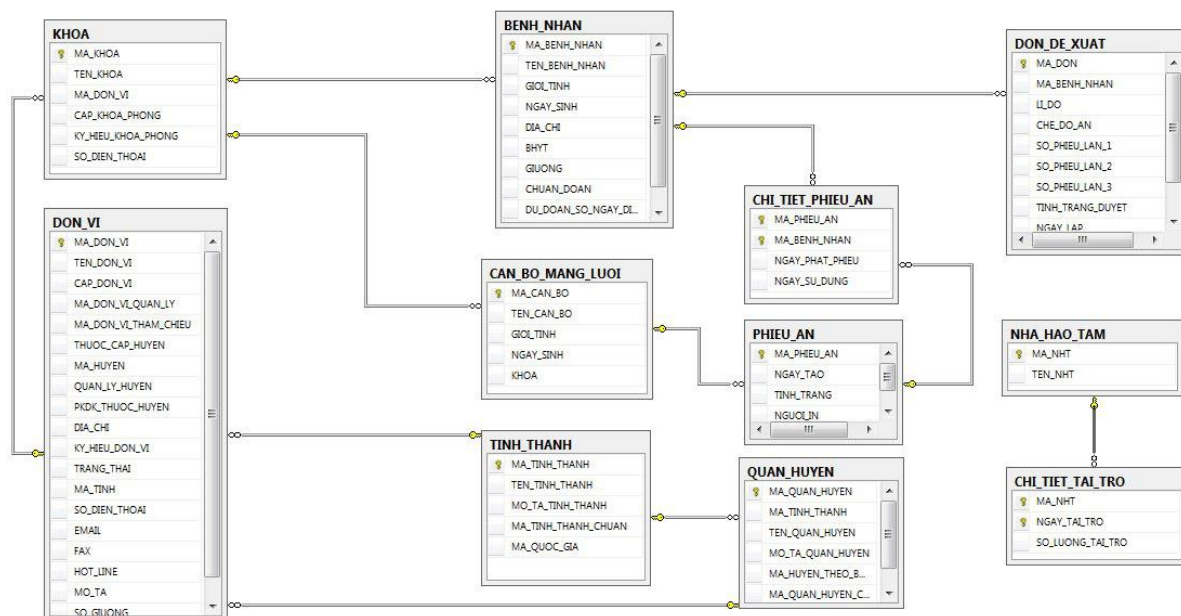
### **3.2 Phạm vi áp dụng**

Đối với quản lý kho dược: Tại các bệnh viện lớn, nhỏ có kho dược của bệnh viện.

Đối với cận lâm sàng: Tại các bệnh viện có điều trị cận lâm sàng, vật lý trị liệu .

Đối với phần mềm quản lý suất ăn từ thiện:

- Tại các bệnh viện, phòng khám, tổ chức từ thiện có trương trình suất ăn từ thiện.
- Với các đơn vị: Khoa lâm sàng có người bệnh khó khăn điều trị nội trú.
- Cán bộ y tế là MLCTXH và cán bộ phụ trách suất ăn tại ĐV.



Hình 3.1 Mô hình cơ sở dữ liệu

### 3.2 Sản phẩm:

**Gateway** 0.0.1-SNAPSHOT Trang chủ Entities Administration Account

**Danh sách đơn đề xuất suất ăn từ thiện** [+ Thêm mới đơn đề xuất](#)

Mã bệnh nhân	Lí do	Tình trạng	Ngày lập	
1	Đơn đề xuất của bệnh nhân 01	Duyệt	21/03/2019	<a href="#">Xem</a> <a href="#">Sửa</a> <a href="#">Xóa</a>
3	Đơn đề xuất của bệnh nhân 03	Không được duyệt	21/03/2019	<a href="#">Xem</a> <a href="#">Sửa</a> <a href="#">Xóa</a>
4	Đơn đề xuất của bệnh nhân 04	Không được duyệt	21/03/2019	<a href="#">Xem</a> <a href="#">Sửa</a> <a href="#">Xóa</a>

Hình 3.2 Giao diện chính



**Gateway** 0.0.1-SNAPSHOT

Nhả phát triển

Trang chủ Entities Administration Account

### Thông tin đơn đề xuất

Mã đơn đề xuất

8

Mã bệnh nhân

1

Lí do

Đơn đề xuất của bệnh nhân 01

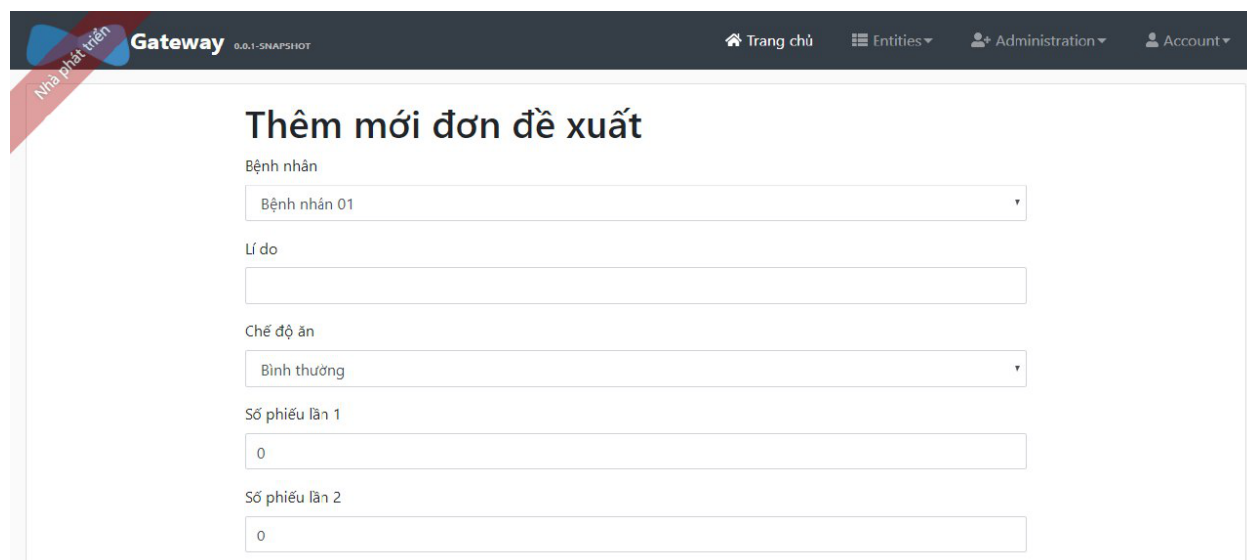
Chế độ ăn

1

Số phiếu lần 1

10

Hình 3.3 Giao diện xem chi tiết



**Gateway** 0.0.1-SNAPSHOT

Nhả phát triển

Trang chủ Entities Administration Account

### Thêm mới đơn đề xuất

Bệnh nhân

Bệnh nhân 01

Lí do

Chế độ ăn

Bình thường

Số phiếu lần 1

0

Số phiếu lần 2

0

Hình 3.4 Giao diện thêm mới đơn đề xuất

**Gateway** 0.0.1-SNAPSHOT

Trang chủ Entities Administration Account

## Cập nhật đơn đề xuất

Mã đơn đề xuất: 8

Bệnh nhân: Bệnh nhân 01

Lí do: Đơn đề xuất của bệnh nhân 01

Chế độ ăn: Bệnh nhân

Số phiếu lần 1: 10

Số phiếu lần 2:

Số phiếu lần 3:

Tình trạng duyệt: Đã được duyệt

Ngày lập: 21/03/2019

Ngày duyệt: 21/03/2019

[← Trở về](#)

Hình 3.6 Giao diện sửa thông tin đơn đề xuất

**Gateway** 0.0.1-SNAPSHOT

Trang chủ Entities Administration Account

## Danh sách đơn đề xuất

[+ Thêm mới đơn đề xuất](#)

Mã bệnh nhân	Lí do	Tình trạng duyệt	Ngày lập	Ngày duyệt	Hành động
1	Đơn đề xuất của bệnh nhân 01	Đã được duyệt	21/03/2019	21/03/2019	Xem Sửa Xóa
3	Đơn đề xuất của bệnh nhân 03	Không được duyệt	21/03/2019	21/03/2019	Xem Sửa Xóa
4	Đơn đề xuất của bệnh nhân 04	Không được duyệt	21/03/2019	21/03/2019	Xem Sửa Xóa

Xác nhận xóa đơn đề xuất

Xóa đơn đề xuất này?

Xóa Không

Hình 3.7 Giao diện xóa đơn đề xuất

## **PHẦN 4 : BÀI HỌC KINH NGHIỆM CHO BẢN THÂN**

Tuy đợt thực tập chỉ kéo dài trong gần 2 tháng, nhưng bản thân em đã học được rất nhiều kinh nghiệm, tiếp thu được nhiều kiến thức mới, rất bổ ích cho công việc trong tương lai.

Thực tập không chỉ là quá trình giúp sinh viên chúng em có được kiến thức, kinh nghiệm thực tế về một lĩnh vực chuyên môn. Thực tập chính là cơ hội để sinh viên chúng em quan sát công việc hàng ngày tại một công ty, văn hóa và môi trường làm việc, cũng là cơ hội để sinh viên hiểu thêm về lĩnh vực, ngành nghề mà mình định hướng. Có thể những gì chúng em nghĩ sẽ hoàn toàn khác với thực tế, vì vậy thực tập là một bước quan trọng để sinh viên có thời gian định hướng và phát triển sự nghiệp sau khi ra trường.

Tuy nhiên, vẫn còn một số hạn chế mà bản thân em chưa thực hiện được đó là: chưa áp dụng hết kiến thức của mình vào trong công việc, giải quyết vấn đề còn thiếu kinh nghiệm, chưa nắm vững về các sản phẩm, phần mềm của công ty. Từ những việc đó, em thấy mình cần phải cố gắng hơn nữa và tích cực chủ động hơn trong công việc, học hỏi thêm từ những anh chị đã đi trước.

Tóm lại, thực tập là bước chuẩn bị, tích lũy quan trọng để sinh viên ra trường phát triển. Riêng bản thân em, sau lần thực tập này, em đã nhận được nhiều ý kiến đóng góp từ thầy cô, các anh chị từ đó thấy được những sai sót của bản thân để có hướng khắc phục và dần hoàn thiện hơn. Đây là cơ hội để cho em có cái nhìn khách quan về bản thân, chuẩn bị và cải thiện mình tốt hơn để có thể phát triển trong ngành nghề bản thân đã chọn.



## TÀI LIỆU THAM KHẢO

- [1] Viễn thông Tiền Giang (<https://vnpttiengiang.vn/index.php?m=info&q=aboutus>)
- [2] Jhiper (<https://www.jhipster.tech/>)
- [3] Mô hình MVC (<http://freetuts.net/mvc-php-mo-hinh-mvc-la-gi-354.html> )
- [4] Reactjs (<https://reactjs.org/>)
- [5] Microservices (kiến trúc nhiều dịch vụ nhỏ)  
(<https://techmaster.vn/posts/33594/gioi-thieu-ve-microservices>)
- [6] Spring Boot (<https://spring.io/projects/spring-boot>