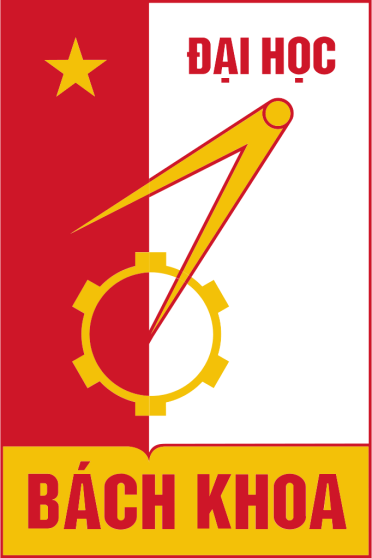
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

\_\_\_\_\_\_\_\_\_\_\_\_\_\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



***Bài tập lớp môn***

***Các phương pháp phát triển phần mềm nhanh***

*Phát triển hệ thống phần mềm hỗ trợ đặt phòng khách sạn*

Danh sách sinh viên nhóm :

Lê Minh Nghĩa 20207694

Nguyễn Minh Đức 20207664

Lê Phạm Long 20207712

Đỗ Minh Hiếu 20207988

Lê Duy Hiếu 20207671

Mục lục

**[I. Thu thập yêu cầu người sử dụng 2](#_gjdgxs)**

[1.1. Lý do chọn đề tài 2](#_30j0zll)

[1.2. Khảo sát địa điểm 3](#_1fob9te)

**[II. Xác lập yêu cầu 4](#_3znysh7)**

[2.1. Cơ sở vật chất 4](#_2et92p0)

[2.2.Mô tả hoạt động hệ thống cũ 5](#_tyjcwt)

[2.3. Yếu điểm của hệ thống cũ 6](#_3dy6vkm)

[2.4. Đề xuất hệ thống mới : 6](#_1t3h5sf)

[2.5. Các giá trị mang lại từ hệ thống mới 6](#_4d34og8)

**[III. Đặc tả yêu cầu phần mềm 7](#_2s8eyo1)**

[3.1. Xác định yêu cầu phần mềm 7](#_17dp8vu)

[3.2. Biểu đồ ca sử dụng 7](#_3rdcrjn)

[3.3. Đặc tả ca sử dụng 8](#_26in1rg)

[3.4. Biểu đồ tuần tự 11](#_lnxbz9)

**[IV. Thiết kế cơ sở dữ liệu 15](#_1ksv4uv)**

[4.1 Xây dựng mô hình liên kết thực thể 15](#_44sinio)

[4.2 Xây dựng mô hình quan hệ dữ liệu 16](#_2jxsxqh)

[4.3 Dữ liệu mẫu 19](#_z337ya)

**[V. Docker / docker compose 21](#_1y810tw)**

[5.1. Giới thiệu Docker 21](#_4i7ojhp)

[5.2. Thành phần của Docker 22](#_2xcytpi)

[5.3. Cài đặt và lệnh cơ bản Docker 23](#_1ci93xb)

[5.4. Triển khai Docker 26](#_3whwml4)

**[VI. Quy trình CI / CD 28](#_2bn6wsx)**

[6.1. Giới thiệu về CICD 28](#_qsh70q)

[6.2.Ưu và nhược điểm của CI/CD: 29](#_3as4poj)

[6.3. Cách thức triển khai của CI/CD 30](#_1pxezwc)

[6.4. Cài đặt và sử dụng Jenkins 31](#_49x2ik5)

# 

# **I. Thu thập yêu cầu người sử dụng**

## **1.1. Lý do chọn đề tài**

- Cùng với sự phát triển của công nghiệp dịch vụ du lịch, hiện nay trên thị trường có rất nhiều phần mềm quản lý khách sạn và những phần mềm đó đã hỗ trợ khá tốt trong công tác quản lý. Trong đó sử dụng nhiều nhất là các trang web quảng bá và hỗ trợ đặt phòng trực tuyến , giúp tăng quy mô và đối tượng khách hàng

- Tuy nhiên với nhiều khách sạn chỉ với quy mô trung bình, vừa và nhỏ, những người chủ hay người quản lý đều chỉ cho thông tin đặt phòng lên một trang web bên thứ ba, thường bị trộn lẫn với các phòng từ khách sạn khác nên hiệu quả không cao như mong muốn .Xuất phát từ nhu cầu thực tiễn để áp dụng công nghệ thông tin vào trong công tác quảng bá để tăng hiệu suất công việc đó, bọn em đã tìm đến đề tài xây dựng phần mềm với chức năng chính là hỗ trợ đặt phòng và quảng bá nhằm giúp các khách sạn mở rộng quy mô hoạt động trên không gian mạng, để từ đó không bỏ lỡ cơ hội cho thuê phòng của khách sạn.

## **1.2. Khảo sát địa điểm**

- Sau khi tham khảo và tìm hiểu 1 số khách sạn, bọn em quyết định lựa chọn khách sạn Minh Khuê tại 57 Nguyễn Lạc, phố Trần Phú, thành phố Quy Nhơn để tham khảo cho bài làm của nhóm. Vì đây là khách sạn nằm tại vùng biển nên hệ thống phần mềm được thiết kế sẽ có một chút tính chất đặc thù dựa trên địa lý .

- Các thông tin cần tìm hiểu bao gồm :

1. Khách sạn có khoảng bao nhiêu nhân viên ? Gồm những bộ phận nào và số nhân viên mỗi bộ phận ?
2. Khách sạn có cơ sở vật chất thế nào ( số phòng , thang máy , các khu vực dịch vụ ,nhà ăn …) ? Điều kiện của các cơ sở vật chất đó thế nào ?
3. Khách hàng khi đặt phòng cần làm những gì, thông qua đâu ?
4. Danh sách của các khách hàng đã đặt phòng, đang ở được lưu trữ ở đâu ? Sửa thông tin khi có sai sót thế nào ?
5. Khách sạn có những dịch vụ tại chỗ gì ?
6. Nếu khách sạn sử dụng một phần mềm quản lí , phần mềm đó phải đảm bảo những điều kiện gì ?
7. Cách thức hoạt động cụ thể của khách sạn ra sao?

- Quá trình thu thập thông tin bao gồm

+) Tìm kiếm thông tin khách sạn trên mạng Internet, đánh giá khách hàng và các trang web giới thiệu.  
 +) Gọi điện và phỏng vấn quản lí/chủ khách sạn trong sự cho phép của họ.

# **II. Xác lập yêu cầu**

## **2.1. Cơ sở vật chất**

- Khách sạn có cơ sở vật chất chỉ ở mức vừa phải gồm tòa nhà 7 tầng, 28 phòng , hầm đỗ xe, nhà ăn , 2 thang máy . Ngoài thuê phòng khách sạn , khách sạn còn có các dịch vụ phục vụ các bữa ăn trong ngày , dọn dẹn phòng . Khách sạn có hầm để xe cho khách kích cỡ vừa cho 3 ô tô 6 chỗ và một vài xe máy. Không có các dịch vụ khác như bể bơi , hội trường do quy mô khách sạn khá nhỏ.

- Về nhân sự có khoảng 30 nhân viên bao gồm nhân viên dịch vụ, kế toán , lễ tân … Số lượng nhiều nhất là nhân viên dịch vụ :

+) Quản lý: có nhiệm vụ quản lý trực tiếp khách sạn. mọi vấn đề của khách sạn đều phải thông qua quản lí và có nhiệm vụ quản lý danh sách hoạt động, quản lý thông tin về phòng, đưa ra chiến lược kinh doanh cụ thể cho khách sạn.

+) Nhân viên lễ tân: có nhiệm vụ tiếp nhận khách đặt phòng và khách thuê phòng, trả lời những thắc mắc của khách hàng. Nhận yêu cầu khi khách muốn sử dụng dịch vụ của khách sạn. Thanh toán hóa đơn khi khách hàng trả phòng.

+) Nhân viên kế toán: quản lý thông tin tài chính thu chi trong khách sạn, có nhiệm vụ lập hóa đơn thanh toán cho khách hàng, lập phiếu chi khi khách sạn cần chi tiêu cho vấn đề sửa chửa hoặc mua sắm phục vụ việc kinh doanh. Hàng tháng nhân viên kế toán phải tiến hành kiểm kê tài chính thu chi và đóng thuế kinh doanh cho khách sạn.

+)Nhân viên bảo vệ: bảo vệ có nhiệm vụ giữ xe, giữ gìn an ninh trật tự trong khách sạn. Theo dõi các thiết bị của khách sạn và chịu trách nhiệm về hệ thống máy móc.

+) Nhân viên dịch vụ: có nhiệm vụ quản lý, cung cấp dịch vụ như dọn dẹp , làm nhà bếp , vận chuyển hành lý cho khách hàng khi khách hàng yêu cầu. Kiểm tra hiện trạng cơ sở vật chất trong phòng khi khách trả phòng.

## **2.2.Mô tả hoạt động hệ thống cũ**

- Về hệ thống , khách sạn có một số hotline và đăng thông tin quảng bá lên các website thứ ba . Khách hàng sẽ xem và đặt phòng trước qua website đó hoặc gọi điện cho lễ tân qua số hotline . Các danh sách khách hàng sẽ được kiểm soát bởi phần mềm quản lí của khách sạn . Thông tin khách hàng được lễ tân gõ lưu lên hệ thống và để đối chiếu khi khách đến nhận phòng . Các dịch vụ hay chi phí phát sinh thì được ghi lại ở mục riêng hoặc trả luôn nên thường bị sai sót . Khách đến thuê phòng đối chiếu nhận phòng với lễ tân thông qua thông tin lưu trên hệ thống đặt rồi được đưa lên phòng bởi nhân viên phục vụ .

- Khách sạn cũng tiếp nhận các đoàn khách đi theo tour đặt trước . Lúc này , tên người đặt phòng sẽ lấy tên đoàn trưởng làm đại diện , không cần lấy tên toàn bộ đoàn . Thông tin của công ty tour và hướng dẫn viên cũng được ghi lại trong danh mục khách đặt phòng.

- Các khiếu nại thường được thông báo trực tiếp với lễ tân và phải chờ phản ánh lại cho quản lý để giải quyết

## **2.3. Yếu điểm của hệ thống cũ**

- Như đã nói ở trên, việc sử dụng trang web bên thứ ba trung gian để quảng bá cho khách sạn sẽ không đem lại hiệu quả lớn.

- Khách hàng không biết rõ được ưu đãi hay các dịch vụ khách sạn vì trang web cũ có giới hạn về không gian.

- Trong trường hợp khách sạn muốn mở rộng trong tương lai sẽ yêu cầu phải tìm thêm cách quảng bá , gây mất thời gian và nguồn lực

- Thông tin khách sạn update sẽ bị chậm do phải phụ thuộc nhiều phía quản lý trang web

## **2.4. Đề xuất hệ thống mới :**

- Thiết kế một trang web đặt phòng riêng biệt để khách hàng dễ nhận diện

- Đăng các thông tin phòng cho thuê , dịch vụ cung cấp và các thông tin cơ bản của khách sạn

- Đăng các bài báo quảng bá và các thông tin ưu đãi và sự kiện hiện có ở khách sạn

- Có thêm thông tin liên hệ để giải đáp thắc mắc khách hàng

## **2.5. Các giá trị mang lại từ hệ thống mới**

- Giá trị nghiệp vụ :

+) Tăng khả năng xử lý các nhu cầu đặt phòng của khách.

+) Giảm bớt gánh nặng nhu cầu phải đi tìm hoặc thỏa thuận mỗi lần khách sạn muốn quảng bá các sự kiện.

+) Giảm bớt sự phụ thuộc vào 1 bên thứ 3 khi truyền thông quảng cáo khách sạn.

- Giá trị kinh tế : Giảm chi phí vận hành hệ thống , giảm thời gian thực hiện hoạt động của nhân viên . Hệ thống giúp nâng cao chất lượng hoạt động đặt phòng khách sạn , từ đó tăng thu nhập cho khách sạn

# **III. Đặc tả yêu cầu phần mềm**

## **3.1. Xác định yêu cầu phần mềm**

- Yêu cầu chức năng phần mềm :

+) Giới thiệu thông tin của khách sạn , có thể đi kèm các bài báo lớn hoặc bài viết giới thiệu trên mạng.

+) Giới thiệu các gói ưu đãi và dịch vụ như đặt tour, ăn uống, thuê xe cho khách hàng tham khảo.

+) Tra cứu tìm kiếm phòng theo loại phòng , giá tiền, thời gian, và các điều kiện khác để khách tham khảo trước

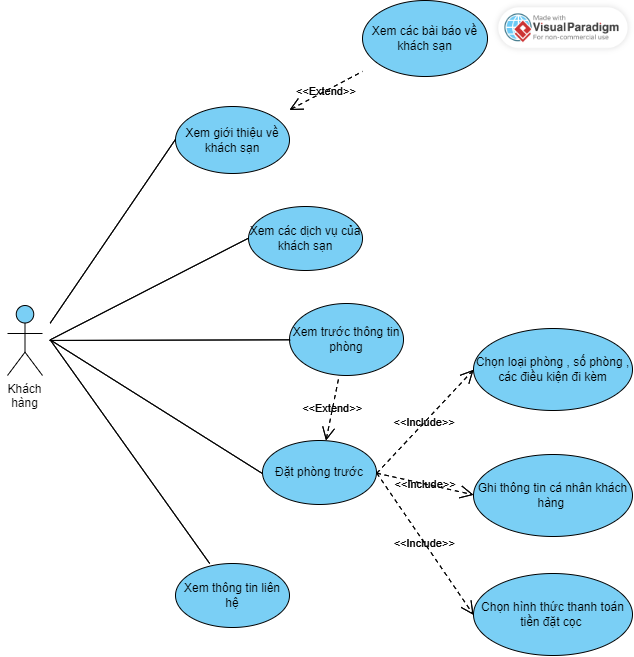
+) Đặt phòng trước và lưu trữ thông tin khách hàng

+) Đưa ra thông tin liên hệ của khách sạn và giải đáp các

thắc mắc khách hàng

## **3.2. Biểu đồ ca sử dụng**

**-** Biểu đổ use case tổng quát



## **3.3. Đặc tả ca sử dụng**

- Đặc tả ca sử dụng của Lễ tân

|  |
| --- |
| Tên ca sử dụng : Đặt phòng |
| Tác nhân chính : Khách hàng |
| Mô tả ngắn gọn : Use case này cho phép khách hàng đặt phòng trước và thực hiện các thủ tục cần thiết |
| Điều kiện tiên quyết : |
| Các mối liên hệ :  +) Tương tác :  +) Bao gồm : Chọn loại phòng và dịch vụ đi kèm ; Ghi thông tin cá nhân ; Chọn hình thức thanh toán tiền đặt cọc  +) Mở rộng : Xem trước thông tin phòng  +) Khái quát hóa : |
| Luồng sự kiện thông thường :   1. Khách hàng bấm vào chọn đặt phòng 2. Trang web hiện lên một form thông tin phòng cho khách hàng điền vào 3. Hệ thống kiểm tra thông tin phòng của ngày mà khách hàng yêu cầu đặt phòng và loại phòng khách yêu cầu. 4. Xác nhận thông tin khách hàng và yêu cầu khách hàng điền thông tin cá nhân cũng như chọn hình thức thanh toán tiền đặt cọc . 5. Khách hàng điền thông tin và xác nhận trả tiền. 6. Hệ thống xác nhận khách hàng đã gửi tiền đặt cọc , gửi email xác nhận cho khách theo địa chỉ email 7. Lưu lại thông tin trên cơ sở dữ liệu của hệ thống. 8. Kết thúc Use case |
| Luồng ngoại lệ / Tương đương :  3.1. Loại phòng mà khách hàng yêu cầu đã hết phòng trống : Hệ thống thông báo hết phòng với loại phòng đã chọn và yêu cầu chọn loại phòng khác.  4.1. Dữ liệu nhập không hợp lệ : Hệ thống thông báo và yêu cầu thực hiện lại. |

|  |
| --- |
| Tên ca sử dụng : Giới thiệu về khách sạn |
| Tác nhân chính : Khách hàng |
| Mô tả ngắn gọn : Use case này cho phép khách hàng xem qua giới thiệu và thông tin về khách sạn |
| Điều kiện tiên quyết : |
| Các mối liên hệ :  +) Tương tác : Khách hàng  +) Bao gồm :  +) Mở rộng : Xem các bài báo về khách sạn  +) Khái quát hóa : |
| Luồng sự kiện thông thường :   1. Khách hàng bấm vào giới thiệu 2. Hệ thống hiển thị trang web với các thông tin về khách sạn và tóm tắt các bài báo khách sạn 3. Khách hàng có thể bấm xem thêm để đọc thêm bài báo   4. Kết thúc Use case |
| Luồng ngoại lệ / Tương đương :  2.1. Khách hàng bấm vào một bài báo : Hệ thống hiển thị link dẫn tới trang web bài báo đó |

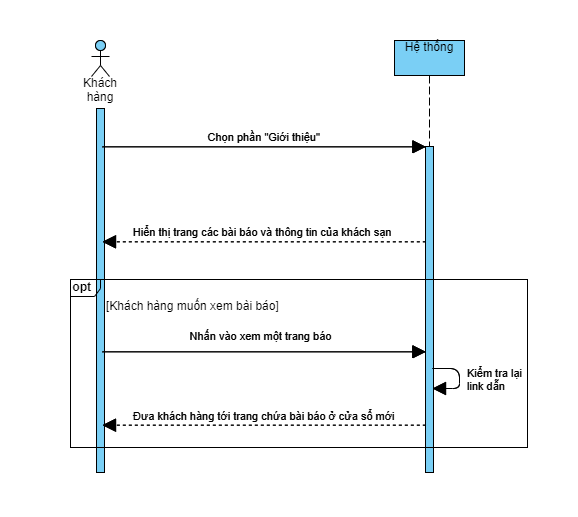
|  |
| --- |
| Tên ca sử dụng : Xem các dịch vụ khách sạn |
| Tác nhân chính : Khách hàng |
| Mô tả ngắn gọn : Use case này cho phép khách hàng xem các dịch vụ khách sạn cung cấp |
| Điều kiện tiên quyết : |
| Các mối liên hệ :  +) Tương tác : Khách hàng  +) Bao gồm :  +) Mở rộng :  +) Khái quát hóa : |
| Luồng sự kiện thông thường :   1. Khách hàng bấm vào nút dịch vụ. 2. Hệ thống hiện thị trang dịch vụ tổng thể của khách sạn. 3. Hệ thống hiện thị cửa sổ cho phép khách hàng xem chi tiết từng dịch vụ.   4. Kết thúc Use case |
| Luồng ngoại lệ / Tương đương :   * 1. Khách hàng bấm vào một cửa sổ dịch vụ : Hệ thống hiển thị trang web về giới thiệu dịch vụ đó |

|  |
| --- |
| Tên ca sử dụng : Xem trước thông tin phòng tin |
| Tác nhân chính : Khách hàng |
| Mô tả ngắn gọn : Use case này cho phép khách hàng xem thông tin các loại phòng của khách sạn. |
| Điều kiện tiên quyết : |
| Các mối liên hệ :  +) Tương tác : Khách hàng  +) Bao gồm :  +) Mở rộng :  +) Khái quát hóa : |
| Luồng sự kiện thông thường :   1. Khách hàng bấm vào loại phòng để vào danh sách phòng 2. Hệ thống hiển thị trang liệt kê các loại phòng 3. Khách hàng có thể bấm vào để xem chi tiết từng loại phòng 4. Kết thúc Use case |
| Luồng ngoại lệ / Tương đương :  3.1. Khách hàng bấm vào một cửa sổ phòng : Hệ thống hiển thị trang web về giới thiệu chi tiết loại phòng đó |

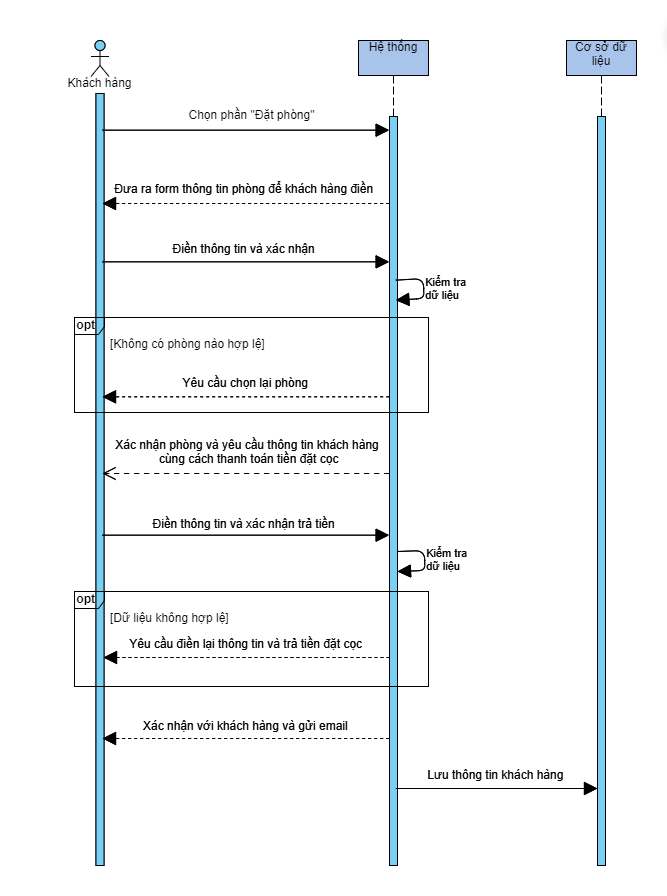
|  |
| --- |
| Tên ca sử dụng : Xem thông tin liên hệ |
| Tác nhân chính : Khách hàng |
| Mô tả ngắn gọn : Use case này cho phép khách hàng liên hệ với bộ phận hỗ trợ khách hàng. |
| Điều kiện tiên quyết : |
| Các mối liên hệ :  +) Tương tác : Khách hàng  +) Bao gồm :  +) Mở rộng :  +) Khái quát hóa : |
| Luồng sự kiện thông thường :   1. Khách hàng bấm vào nút liên hệ. 2. Hệ thống hiển thị cửa sổ yêu cầu nhập thông tin : Tên khách hàng , Số điện thoại, Email, Chú đề, Tin nhắn. 3. Khách hàng điền thông tin và chọn nút “gửi tin” 4. Hệ thống lưu lại thông tin và gửi đến bộ phận hỗ trợ khách hàng. 5. Kết thúc usecase |
| Luồng ngoại lệ / Tương đương :  4.1. Dữ liệu gửi đến không hợp lệ : bộ phận hệ thống bỏ qua tin nhắn và xóa dữ liệu khỏi hệ thống. |

## **3.4. Biểu đồ tuần tự**

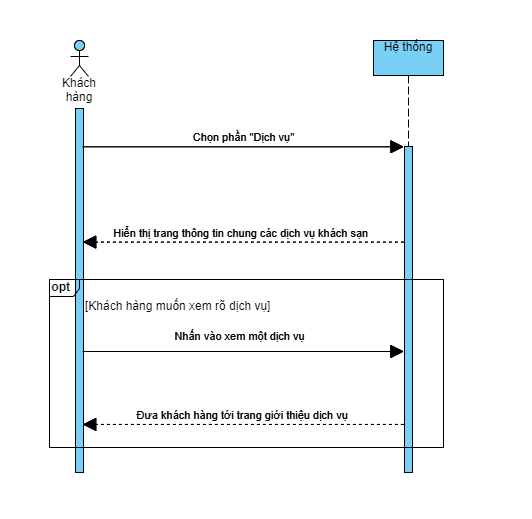
- Giới thiệu về khách sạn



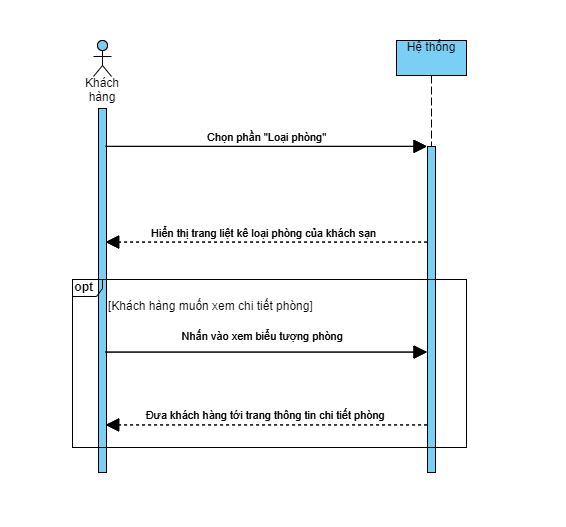
- Đặt phòng



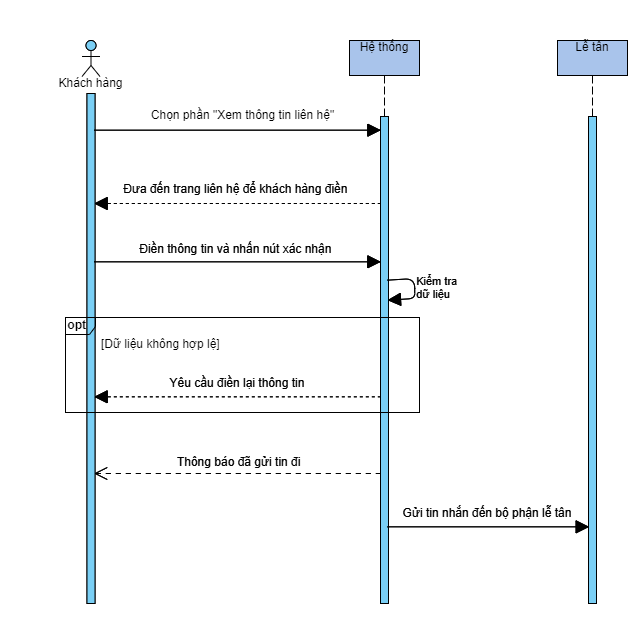
- Xem các dịch vụ khách sạn



- Xem trước thông tin phòng tin



- Xem thông tin và liên hệ



# **IV. Thiết kế cơ sở dữ liệu**

**4.1 Xây dựng mô hình liên kết thực thể**

- Yêu cầu về xác định các thực thể :

+) Phòng cho thuê (Mã phòng, tên phòng, mã loại phòng, trạng thái, mã phiếu)

+) Loại phòng (Mã loại phòng, Tên loại phòng, Số người tối đa, Thông tin phòng, Giá phòng)

+) Phiếu đặt phòng (Mã phiếu, Tên khách, Email, Số điện thoại, Địa chỉ, Ngày đến nhận, Ngày trả, Số lượng khách, Số trẻ em, Ghi chú)

+) Dịch vụ (Mã dịch vụ, Tên dịch vụ, Giá dịch vụ)

+) Hóa đơn (Mã Hóa Đơn, Mã khách hàng, Số thẻ, Tên thẻ, Giá đặt cọc, Ngày lập hóa đơn, Người lập hóa đơn)

**4.2 Xây dựng mô hình quan hệ dữ liệu**

- Bảng Dịch vụ

|  |  |  |
| --- | --- | --- |
| Thuộc tính | Kiểu dữ liệu | Ghi chú |
| MaDichVu | Varchar(5) | Not null,primary key |
| TenDichVu | Varchar(50) | Not null |
| GiaDichVu | int | Not null |

MaDichVu : ID định danh của loại dịch vụ, có dạng S0001

TenDichVu : Tên của dịch vụ cung cấp

GiaDichVu : Giá tiền của dịch vụ

- Bảng Phòng cho thuê

|  |  |  |
| --- | --- | --- |
| Thuộc tính | Kiểu dữ liệu | Ghi chú |
| MaPhong | Varchar(5) | Not null,primary key |
| TenPhong | Varchar(3) | Not null |
| MaLoaiPhong | Varchar(5) | Not null |
| TinhTrang | tinyint | Not null |
| MaPhieu | Varchar(10) | foreigh key |

MaPhong : ID định danh của phòng, có dạng R0001

TenPhong : Số đánh tên của phòng

MaLoaiPhong : ID định danh loại phòng

TinhTrang : cho biết tình trạng của phòng

MaPhieu : ID định danh phiếu người đặt phòng (nếu có)

*Chú thích : TinhTrang có đánh số như sau : 0 là trống, 1 là có người ở, 2*

*là đã được đặt, 3 là đang sửa chữa.*

- Bảng Loại Phòng

|  |  |  |
| --- | --- | --- |
| Thuộc tính | Kiểu dữ liệu | Ghi chú |
| MaLoaiPhong | Varchar(5) | Not null,primary key |
| TenLoaiPhong | Varchar(50) | Not null |
| SoNguoiToiDa | tinyint | Not null |
| GiaPhong | int | Not null |
| ThongTinPhong | Varchar(2000) | Not null |

MaLoaiPhong : ID định danh loại phòng, có dạng T0001

TenLoaiPhong : Tên của loại phòng

SoNguoiToiDa : Số người ở tối đa của loại phòng

GiaPhong: Giá tiền ở một ngày của phòng

ThongTinPhong : tóm tắt các thông tin cơ bản của loại phòng

- Bảng Phiếu đặt phòng

|  |  |  |
| --- | --- | --- |
| Thuộc tính | Kiểu dữ liệu | Ghi chú |
| MaPhieu | Varchar(10) | Not null,primary key |
| TenKhachHang | Nvarchar(50) | Not null |
| SDT | Varchar(15) | Not null |
| Email | Varchar(50) | Not null |
| DiaChi | Nvarchar(200) | Not null |
| SoNguoiLon | tinyint | Not null |
| SoTreEm | tinyint | Not null |
| NgayDenNhan | date | Not null |
| NgayTra | date | Not null |
| MaLoaiPhong | Varchar(5) | Not null,foreigh key |
| SoLuongPhong | tinyint | Not null |
| MaDichVu | Varchar(5) | foreigh key |
| GhiChu | Nvarchar(300) |  |

MaPhieu : ID định danh phiếu

TenKhachHang : Tên của khách hàng

SDT : SĐT của khách

Email : Email của khách hàng

DiaChi : Nơi ở của khách hàng

SoNguoiLon : Số người lớn đi cùng

SoTreEm : Số trẻ em đi cùng

NgayDenNhan : Ngày đến nhận phòng

NgayTra : Ngày trả phòng dự định

MaLoaiPhong : ID định danh loại phòng khách chọn

SoLuongPhong : Số lượng phòng mà khách muốn đặt

MaDichVu : ID định danh của dịch vụ khách chọn

GhiChu : Yêu cầu thêm về phòng của khách

- Bảng Hóa Đơn

|  |  |  |
| --- | --- | --- |
| Thuộc tính | Kiểu dữ liệu | Ghi chú |
| MaHoaDon | Varchar(10) | Not null,primary key |
| MaPhieu | Varchar(10) | Not null,foreigh key |
| SoThe | Varchar(20) | Not null |
| TenThe | Nvarchar(50) | Not null |
| GiaDatCoc | int | Not null |
| NgayLap | date | Not null |
| NguoiLap | Nvarchar(50) | Not null |

MaHoaDon : ID định danh hóa đơn, có dạng HD với 8 chữ số đằng sau

MaPhieu : ID định danh phiếu đặt phòng

SoThe : Số thẻ ngân hàng dùng để trả tiền đặt cọc

TenThe : Tên của thẻ được dùng

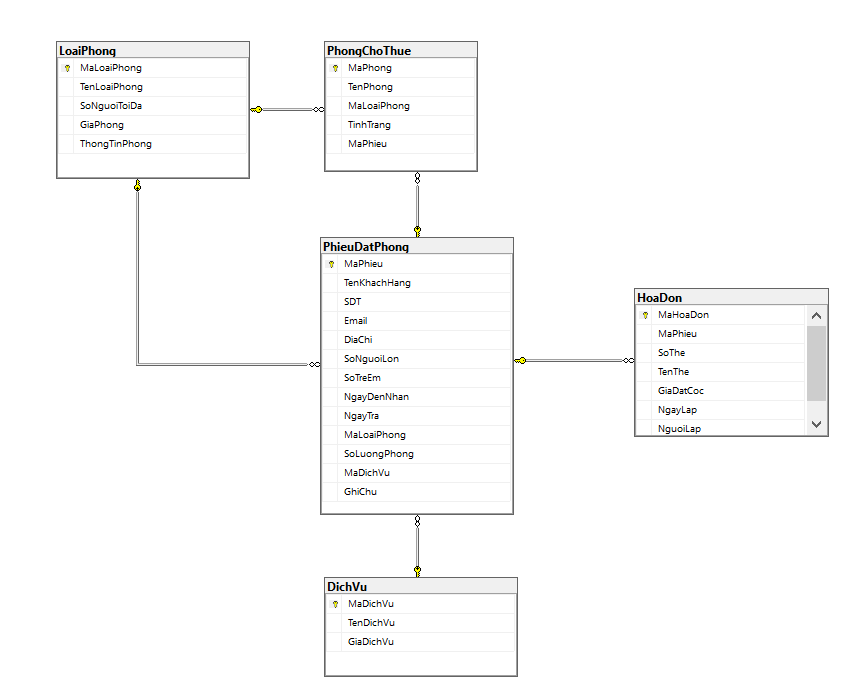
GiaDatCoc : Khoản tiên đặt cọc trước

NgayLap : Ngày lập hóa đơn, cũng như ngày đặt phòng

NguoiLap : Tên nhân viên lập hóa đơn

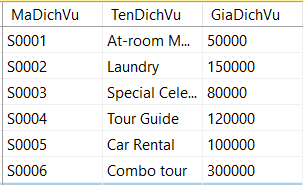
*Chú thích : GiaDatCoc có công thức tính là (giá phòng \* 0.5) x Số phòng + giá dịch vụ đi kèm*

- Sơ đồ quan hệ :

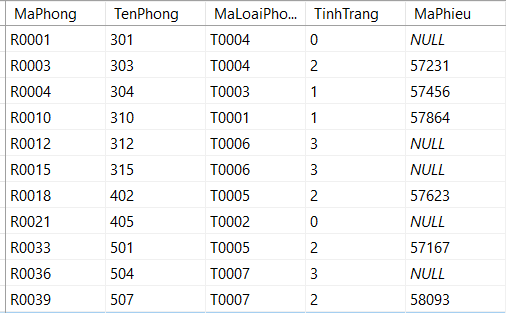


**4.3 Dữ liệu mẫu**

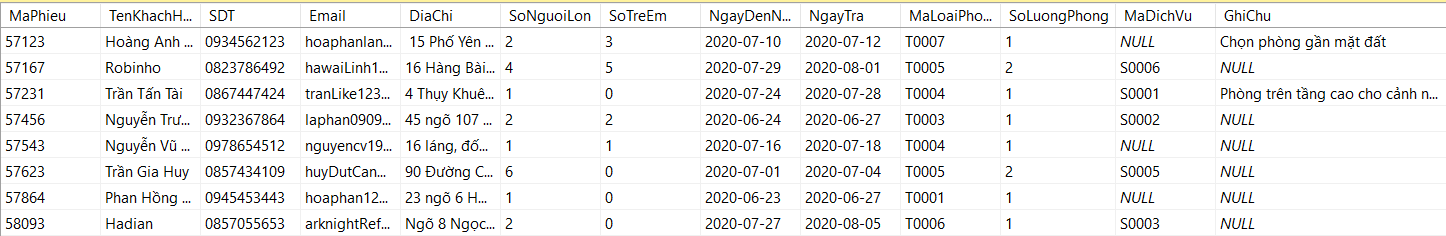
- Bảng Dịch Vụ



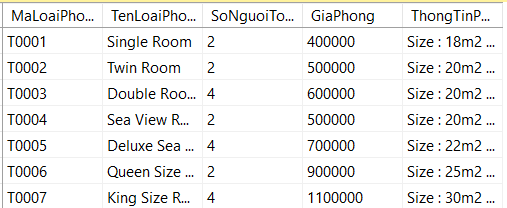
- Bảng Phòng cho thuê



- Bảng Phiếu Đặt Phòng



- Bảng Loại Phòng



- Bảng Hóa Đơn



# **V. Thiết kế giao diện**

## **5.1. Phân tích giao diện**

- Người dùng cuối của hệ thống, ở đây là khách hàng của khách sạn mong muốn đặt phòng khách sạn. Vì thế trang web sẽ phải có thiết kế đơn giản và bố cục ổn định phù hợp cho điều hướng.

- Các tác vụ họ thực hiện theo như đã phân tích ở trên bao gồm xem thông tin khách sạn, xem thông tin dịch vụ , xem thông tin các loại phòng , liên lạc giải đáp thắc mắc và đặt phòng khách sạn.Biểu đồ tuần tự ở phần 3 đã giải thích về quy trình và tương tác của mỗi hành động này thực hiện thế nào.

- Trong quá trình thiết kế giao diện, nhóm sẽ cân nhắc chia tỉ lệ của màn hình ra các khung khác nhau. Điều này tính tới các đầu ra đồ họa có được căn chỉnh để vừa vặn với các giới hạn của thiết bị hiển thị được sử dụng khi giao diện web phải được thiết kế hỗ trợ cả trên điện thoại và các loại máy tính bảng. Một số hình ảnh và khung thông tin quan trọng sẽ được fix cố định vị trí để chúng không bị bỏ qua.

## **5.2. Cấu trúc thiết kế**

- Với mỗi tác vụ ở trên, giao diện trang web sẽ được thiết kế với năm cửa sổ chính là trang chủ , tin tức , dịch vụ , liên hệ và đặt phòng. Các cửa sổ này có thể truy cập qua lại sử dụng một thanh điều hướng chung nằm phía trên phần đầu trang web. Trang chủ sẽ là trang web mặc định vào khi mới truy cập và là trang điều hướng trung gian tới các trang khác.

- Với danh sách loại phòng, người dùng chỉ có thể xem nó khi họ muốn đặt phòng. Trang web sẽ hiển thị hình ảnh và thông tin vắn tắt quan trọng về từng loại phòng. Nếu như khách muốn xem kỹ thông tin phòng, họ có thể bấm vào biểu tượng và nó sẽ đưa họ tới cửa sổ thông tin riêng của loại phòng đó. Giữa các cửa sổ đều có các nút bấm chuyển tiếp qua lại giúp khách không cảm thấy khó khăn.

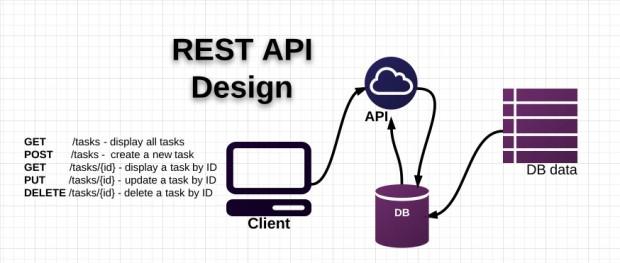
- Trang web sẽ có giao diện thiết kế đơn giản với nhiều khoảng trắng ở hai bên. Những khoảng trống còn gọi là khoảng trắng (Negative space), là những phần họa tiết hoặc hình ảnh có tác dụng làm nền, thu hút sự chú ý từ người đọc. Những khoảng trắng này sẽ không được chứa thông tin hoặc không trang trí nhiều họa tiết bên trong, giúp khách hàng sử dụng trang web cảm thấy dễ nhìn và bắt mắt hơn

# **VI. Thiết kế API**

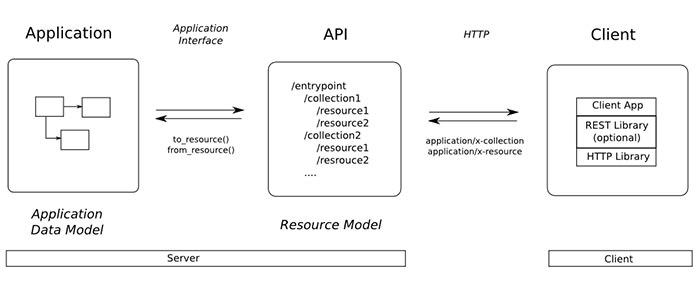
## **6.1. Rest API**

## - Về cơ bản, API (Application Programming Interface) là giao diện lập trình ứng dụng giúp tạo ra các phương thức kết nối với các thư viện và các ứng dụng khác. Đây được xem là một giải pháp thay vì là một công nghệ.

- RestAPI là một loại trong số đó, sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, vv đến một URL để xử lý dữ liệu.



- Chức năng quan trọng nhất của REST là quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE…) và cách định dạng các URL cho ứng dụng web để quản các resource. RESTful không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một RESTful API.



- Về cách hoạt động, REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

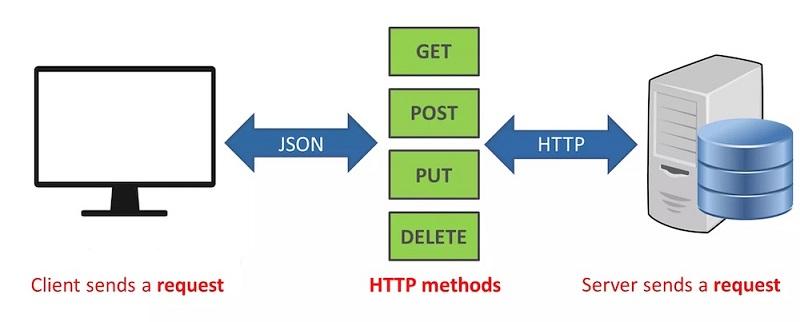
+) GET (SELECT): Trả về một Resource hoặc một danh sách Resource.

+) POST (CREATE): Tạo mới một Resource.

+) PUT (UPDATE): Cập nhật thông tin cho Resource.

+) DELETE (DELETE): Xoá một Resource.

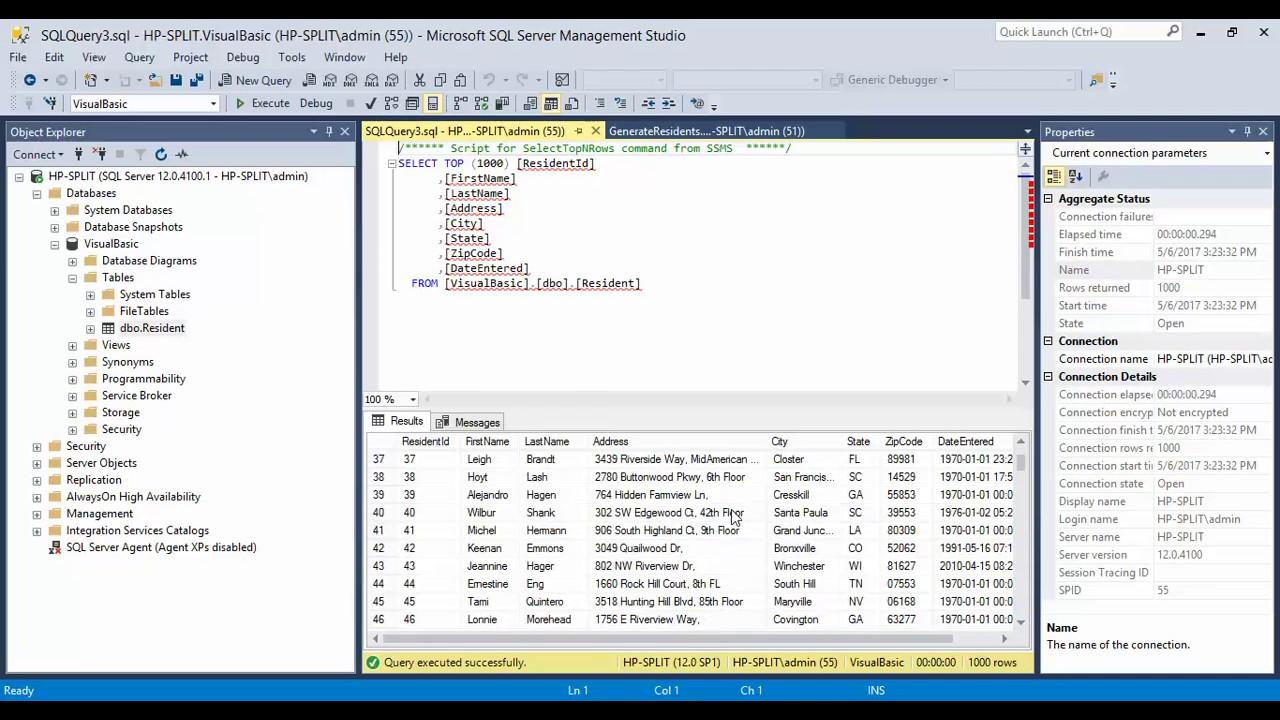
Những phương thức hay hoạt động này thường được gọi là CRUD tương ứng với Create, Read, Update, Delete – Tạo, Đọc, Sửa, Xóa.



- Lý do nhóm chọn Rest API cho bài tập lớn của nhóm vì Code của Rest API đơn giản ngắn gọn so với loại API khác. Các dữ liệu được trả về dưới nhiều định dạng khác nhau: xml, html, json,… và giúp cho ứng dụng rõ ràng và dễ nhìn hơn.

## **6.2. Các công cụ sử dụng**

- Để thiết kế Restful API, nhóm quyết định thiết kế sử dụng Nodejs với cơ sở dữ liệu MSSQL server và code bằng Visual Code Studio. +) SQL server là một hệ quản trị cơ sở dữ liệu phổ biến và mạnh mẽ, hỗ trợ sắp xếp dữ liệu theo các trường khác nhau và nhiều tính năng khác. Sử dụng cơ sở dữ liệu tập trung để lưu trữ thông tin tham chiếu và các trường thông tin khác nhau dễ dàng hơn.



+) Node.js là một [hệ thống phần mềm](https://vi.wikipedia.org/wiki/H%E1%BB%87_th%E1%BB%91ng_ph%E1%BA%A7n_m%E1%BB%81m) được thiết kế để viết các ứng dụng internet có khả năng mở rộng, đặc biệt là [máy chủ web](https://vi.wikipedia.org/wiki/M%C3%A1y_ch%E1%BB%A7_web) được viết bằng c++ và Javascript. Node.js có rất nhiều các module đi kèm trong đó có Express.js. Đây là một framework cho backend web để xây dựng API RESTful với Node.js. Nó được thiết kế để xây dựng các ứng dụng web và API và được gọi là khung máy chủ tiêu chuẩn trên thực tế cho Node.js.



+) Visual Studio Code : đây là trình soạn thảo mã nguồn mở đã quá quen thuộc và nổi tiếng với nhiều developer. Visual Studio Code hỗ trợ sẵn các ngôn ngữ JavaScript và TypeScript cũng như gỡ lỗi phục vụ cho Node.js. Tuy nhiên, để chạy ứng dụng Node.js, bạn sẽ cần cài đặt thời gian chạy Node.js trên máy của mình.

## **6.3. Thiết kế function**

- Các yêu cầu

- Function

# **VII. Docker / docker compose**

## **7.1. Giới thiệu Docker**

- Về cơ bản, docker là một ứng dụng đóng gói phần mềm để cho phép nó chạy bất cứ phần cứng nào. Một phần mềm thường sẽ được phát triển và thiết kế sử dụng một môi trường mà nó tối ưu nhất với các yêu cầu về phần mềm điều kiện phụ thuộc ( ngôn ngữ, cơ sở dữ liệu, thư viện,… ). Câu hỏi là làm sao để tạo lại một trường này trên các môi trường hệ thống khác. Thông thường ta sử dụng máy ảo để mô phỏng phần cứng rồi cài đặt và đóng gói phần mềm vào với mọi thứ mà phần mềm cần để chạy. Tuy nhiên cách này thường chậm và gặp nhiều rào cản do việc mô phỏng trên mỗi máy ảo khác nhau và sử dụng hệ thống khác nhau, chưa tính đến việc phải chia bộ nhớ và ram cho các máy ảo này hoạt động.

- Docker cũng hoạt động tương tự như máy ảo . Tuy nhiên nó chỉ mô phỏng hệ điều hành của máy chủ và các phần mềm chạy trên nền Docker engine chung một kernel. Docker sẽ đóng gói phần mềm điều kiện phụ thuộc này, tách biệt vào các đơn vị tiêu chuẩn hóa tên gọi là [container](https://aws.amazon.com/vi/containers/) , trong đó có thư viện, công cụ hệ thống, mã và thời gian chạy.

- Bằng cách sử dụng Docker, ta sẽ không cần phải nhớ tất cả các phần mềm đi kèm hay phiên bản của chúng mà docker sẽ được thiết lập để chạy giống hệt nhau. Docker là một giải pháp cho phép sự tự do linh hoạt, dùng trên nhiều hệ máy, giảm bộ nhớ yêu cầu cần để mô phỏng môi trường.

## **7.2. Thành phần của Docker**

- Docker có 3 thành phần chính phải để ý tới :

+) Docker file : bản vẽ để xây dựng các image cho docker. Developer đầu tiên xây dựng một tệp docker file để đưa ra định nghĩa và chỉ dẫn môi trường cần xây dựng cho phần mềm

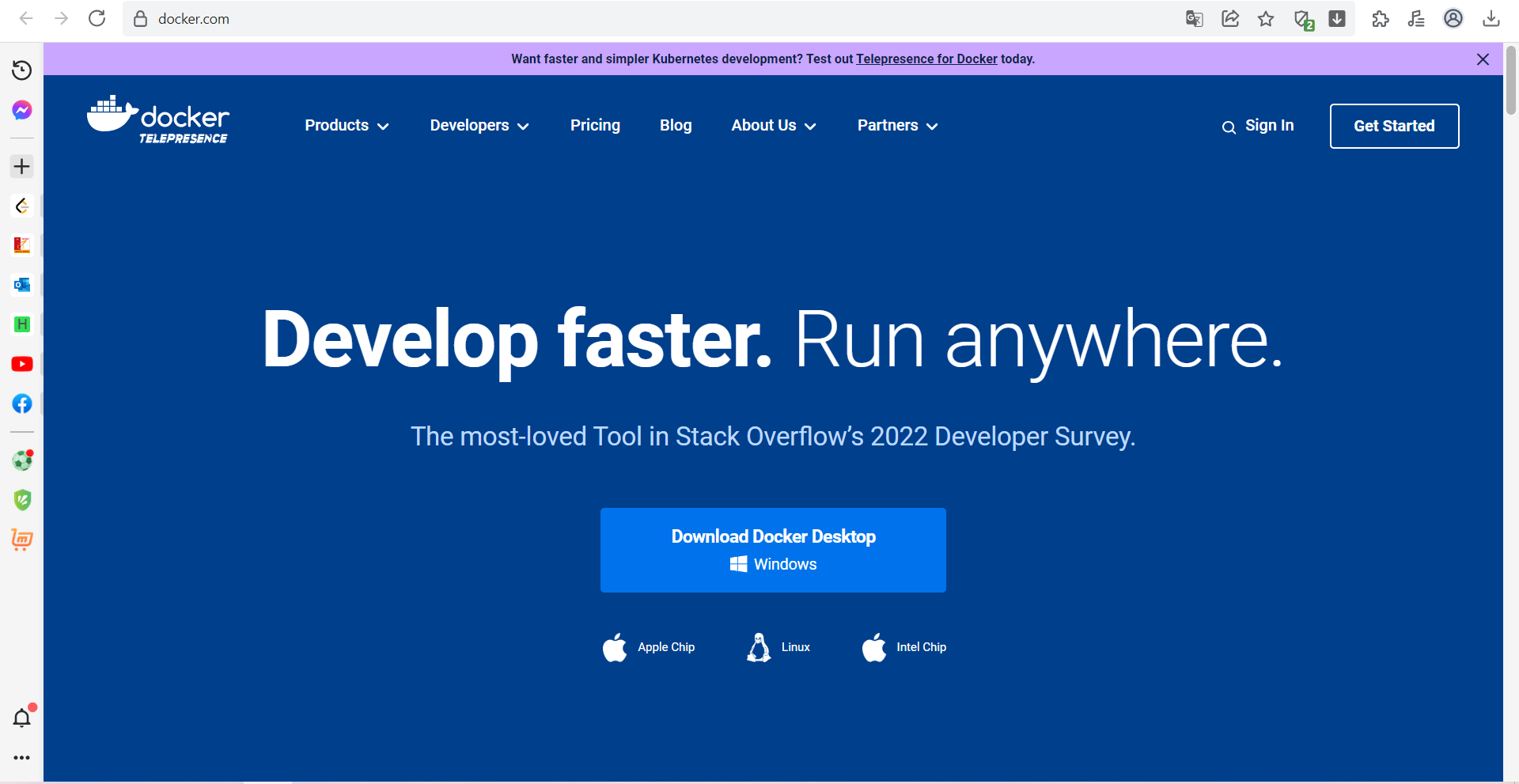
+) Image : chứa các thông số không chỉ của phần mềm mà cả các thông số xây dựng lại môi trường theo docker file và lưu lại dưới dạng một snapshot cố định. Image có thể được tải lên server cloud dưới hình thức pubic hoặc private để cho bất cứ developer nào muốn chạy phần mềm có thể tải xuống để xây dựng các container. Nói cách khác, nó là mẫu hình để chạy các container

+) Container : Chứa các thành phần để chạy phần mềm. Một image có thể tạo ra cùng một container vô số lần khác nhau trên vô số nơi khác nhau.

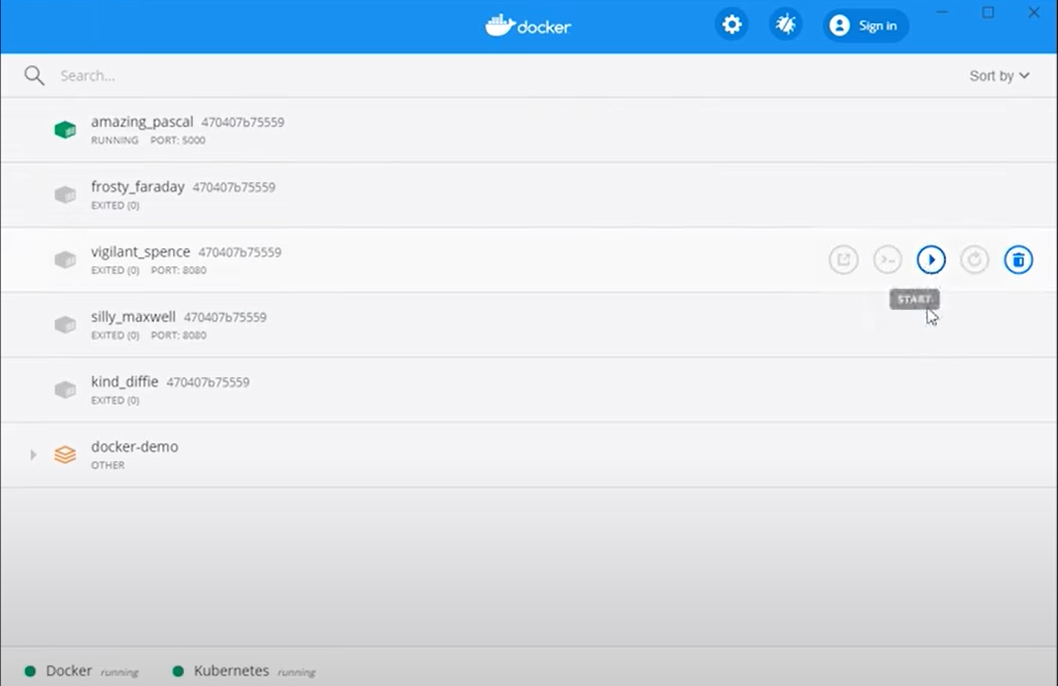
- Một công cụ khác docker cung cấp là docker compose. Bình thường mỗi container chỉ phục vụ cho 1 tiến trình chạy. Nếu phần mềm yêu cầu chạy nhiều tiến trình thì ta cũng cần nhiều container chạy cùng lúc. Lúc này ta cần dùng docker compose để có thể chạy và kiểm soát nhiều container cùng lúc thông qua một tệp docker-compose.yml.

## **7.3. Cài đặt và lệnh cơ bản Docker**

- Đầu tiên ta truy cập vào trang chủ chủ docker. Trên nền tảng Windows hoặc Mac, ta nên tải docker desktop để tích hợp tất cả các thư viện và docker engine cần thiết đồng thời hiển thị giao diện GUI của các container đang chạy. Trang web còn cung cấp hướng dẫn các bước cài đặt và các yêu cầu cấu hình và phần mềm máy cho docker. Sau khi cài đặt, ta có thể truy cập docker từ command line trong máy tính hoặc code editor mà ta sử dụng và quản lí các image từ GUI của docker desktop.



- Với các môi trường ide và code editor , bạn nên cài thêm cả extension docker của chúng. Phần extension sẽ hỗ trợ việc viết và thiết lập các docker file, kết nối các remote registry, …

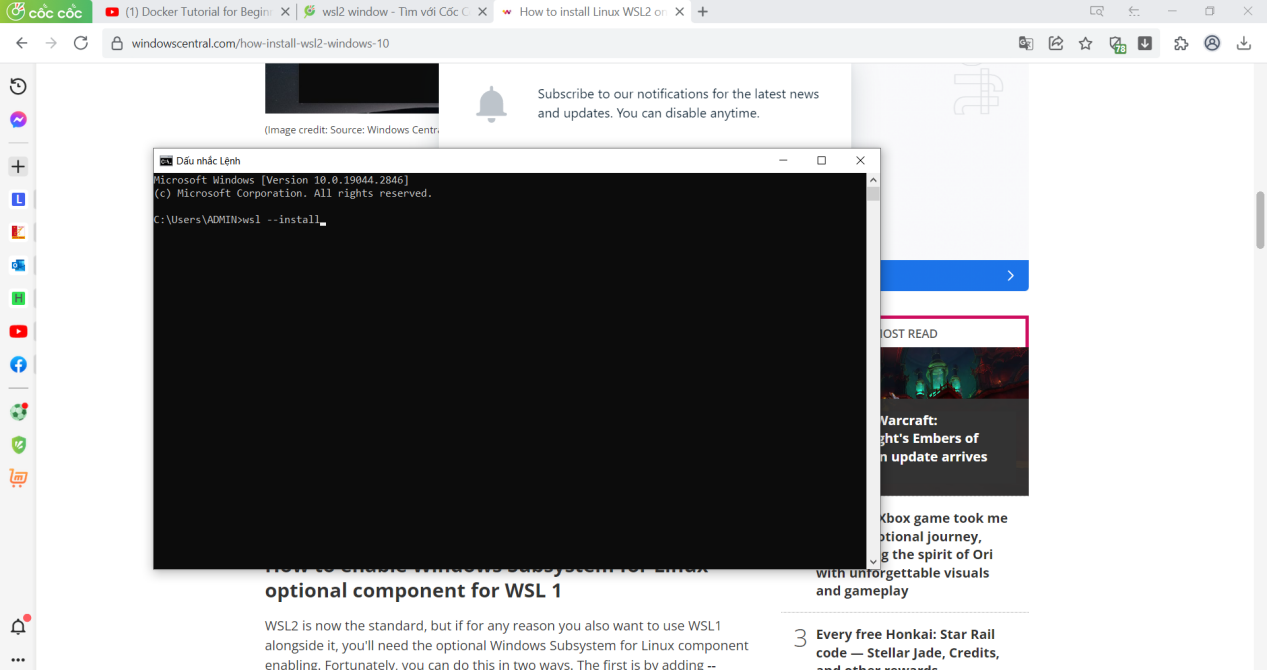


*Giao diện của docker desktop*

**

*Extension docker trên VCS*

*-* Với hệ máy window , người dùng nên cài thêm wsl hay hạt nhân Linux. Ta sử dụng thêm câu lệnh wsl --install trên command line.



- Về cú pháp , câu lệnh docker có cấu trúc :

docker <component> <command>

Component là các thành phần gọi đến tương tác như image, container, network,… Còn mã lệnh tương ứng mỗi đối tượng để thao tác với nó. Một số lệnh thường dùng như :

+) docker ps : hiển thị tất cả các container đang chạy và thông tin của chúng trên hệ thống.

+) docker <component> run : chạy component được đề cập tới.

+) docker <component> stop : tạm dừng component được đề cập tới.

+) docker <component> ls : liệt kê danh sách các đối tượng thuộc loại component nhắc tới.

+) docker <component> prune : xóa component không còn được dùng trong hệ thống nữa.

+) docker start < container > : chạy con container trên docker

## **7.4. Triển khai Docker**

- Bước 1 : viết tệp dockerfile để xây dựng nên image. Tệp dockerfile sẽ có dạng :

FROM {base image}

WORKDIR {địa chỉ phần mềm}

COPY {source} {dest}

RUN command

COPY ..

ENV PORT={số hiệu cổng}

EXPOSE {số hiệu cổng}

CMD [ “ ”,” “]

FROM khai báo **base image** là môi trường để phần mềm chạy trên, có thể lấy từ trên docker hub.

WORKDIR đặt địa chỉ mà các lệnh xong làm việc trong đó như COPY, ADD, RUN.

COPY lấy **source** là vị trí nguồn những tệp muốn lấy đến **dest** là vị trí trong hệ thống file của image. Nếu **dest** chưa tồn tại thì docker sẽ tự tạo.

RUN chạy các command đã viết như python3 install.

COPY .. đưa toàn bộ dữ liệu hiện tại vào đường dẫn tạo ra ở trên

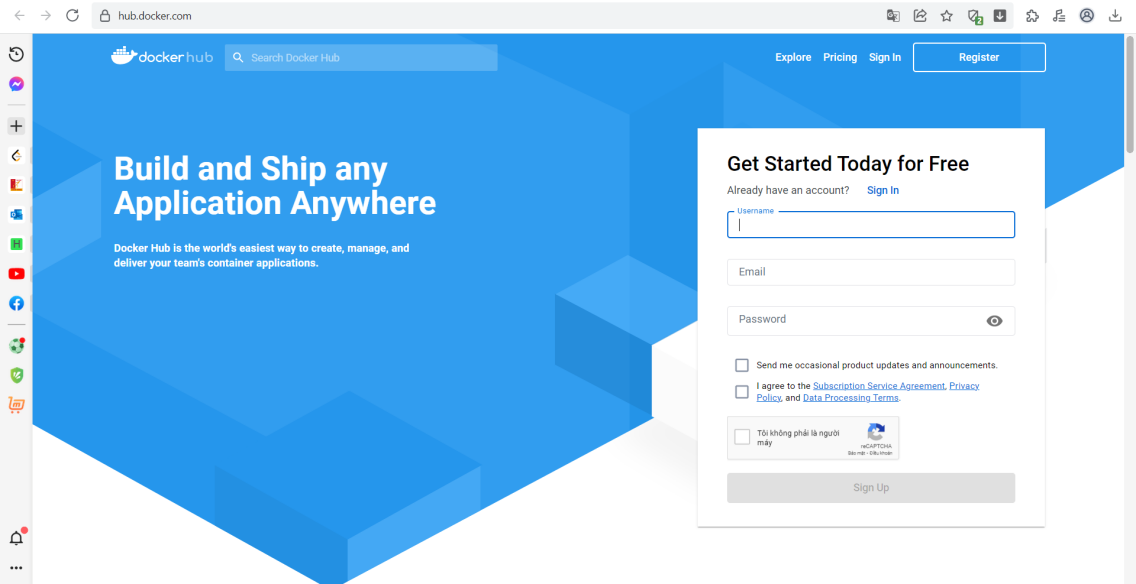
ENV khai báo số hiệu cổng mà qua đó ta có thể truy cập vào phần mềm. Ta có thể dùng với lệnh EXPOSE.

- Bước 2 : xây dựng image. Vào trong command prompt hoặc terminal của editor, ta dùng lệnh docker build -t {tên người dùng}/{tên app} . Nó sẽ chạy qua các lệnh ta đã viết trong dockerfile và xây dựng nên image.

+) Đương nhiên, image cũng có thể lấy trực tiếp dùng từ docker Hub. Docker hub là một registry chính thức của công ty cung cấp để tìm image phù hợp với loại và phiên bản ngôn ngữ lập trình và phần backend của nhóm làm. Ta có thể kiểm tra danh sách image với lệnh docker image. Sau đó dùng câu lệnh để lấy image :

docker pull {name} : {tag}

với name là tên của image, tag là phiên bản của nó.



*Giao diện docker Hub*

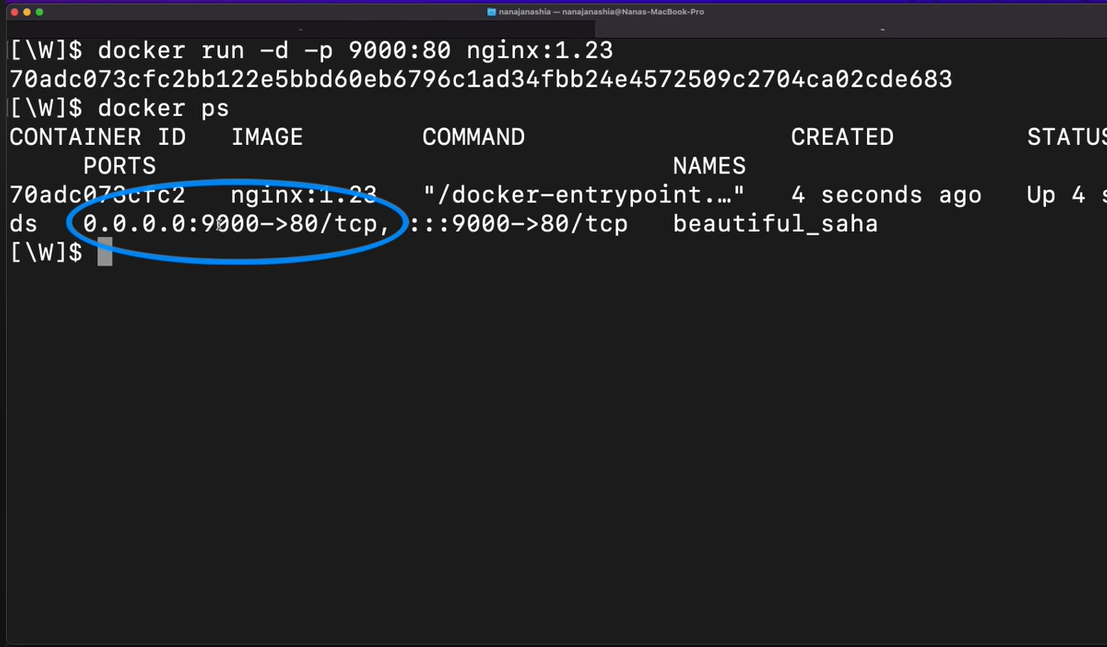
Khai báo image trong terminal rồi chạy lệnh docker run {name}:{tag} để chạy container dựa trên image tương tự như trên.

- Bước 3 : post binding để có thể access được thông tin container. Ta sử dụng câu lệnh

docker run -p {host port}:{local port} {name}:{tag}

VD : docker run -p 8080:8080 -p 50000:50000 jenkins:lts

Host port do người lập trình chọn, local port là cổng container được định sẵn trên máy.



*Ví dụ : binding local port 80 với host port 9000*

Giờ ta có thể truy cập vào container thông qua cổng mới.

- Bước 4 : giờ ta có thể chạy image và truy cập container sử dụng lệnh docker run -p {host port}:{local port} . Giờ cổng localhost sẽ hiển thị thông tin và chạy phần mềm.

# **VIII. Quy trình CI / CD**

## **8.1. Giới thiệu về CICD**

- CI là viết tắt của Continuous Integration (tích hợp liên tục), CD là viết tắt của Continuous Delivery (chuyển giao liên tục) hoặc Continuous Deployment (triển khai liên tục).

- Khái niệm CI/CD thường đề cập đến việc tự động hóa trong quy trình phát triển phần mềm và chuyển giao sản phẩm, giúp cho việc tích hợp diễn ra nhanh hơn và sản phẩm hoàn thiện được chuyển đến người dùng trong thời gian ngắn nhất.

- Các developer thường hoạt động độc lập với nhau phần lớn thời gian, đặc biệt ở các công ty lớn và chỉ ngồi lại khi tích hợp triển khai lên môi trường thực tế. Vì thiếu đi sự tương tác thường xuyên nên quá trình tích hợp này thường xảy ra rất nhiều lỗi do conflict code. Trong những trường hợp xấu nhất, code của hàng trăm hàng ngàn lập trình viên bị conflict dẫn đến tiêu tốn thời gian và tiền bạc.

- Áp dụng CI/CD là cách triệt tiêu hiệu quả các bước thủ công trong quy trình phát triển phần mềm/ứng dụng. Việc của Developer chỉ là commit code, còn lại tất cả quy trình bao gồm chạy build, test, deploy sẽ được tự động thực hiện hoàn toàn bởi công cụ (tool) CI/CD. Việc tích hợp liên tục này nếu có thể kết hợp thêm với automation test thì quy trình sẽ chặt chẽ và hạn chế được tối đa các lỗi phát sinh (ví dụ: lỗi phát triển tính năng mới làm hỏng tính năng cũ). Hiện nay, CI/CD đã được áp dụng rộng rãi vào quy trình làm việc của các doanh nghiệp làm trong lĩnh vực IT, song hành cùng với DevOps và Agile.

## **8.2.Ưu và nhược điểm của CI/CD:**

- Tránh được những lỗi không đáng có: chẳng hạn như lỗi compile (khi đưa code lên) hoặc các lỗi phát sinh liên quan đến môi trường build sản phẩm. *Ví dụ: khi làm thủ công, cùng 1 source code nhưng sẽ có sự khác biệt khi bạn A build trên máy bạn A, bạn B build trên máy bạn B.*

- Đảm bảo logic (vì quy trình CI/CD có phần automation test), khi Developer xây dựng tính năng mới sẽ không gây ảnh hưởng đến tính năng cũ.

- Giúp tập trung vào công việc bởi quy trình CI/CD mang tính tự động cao nên Developer không cần phải thực hiện việc build và deploy phần mềm/ứng dụng trên máy cá nhân nữa.

- Nâng cao chất lượng code thông qua quy trình, Developer có thể cài đặt những ràng buộc ngay từ đầu. *Ví dụ: pull request khi được tạo ra thì không được quá lớn, không được vượt quá X thay đổi…, điều này góp phần giúp chất lượng pull request ngày càng tốt hơn.*

- Phát triển kỹ năng unit test cho Developer thông qua các chỉ số ràng buộc về code coverage (% code đã được cover) được cài đặt trong quy trình CI/CD. Nghĩa là khi phát triển tính năng mới, để không làm giảm chỉ số code coverage, Developer phải ý thức được tầm quan trọng của unit test và chủ động học hỏi, nâng cao các kỹ năng liên quan.

- Tối ưu tốc độ phát triển của sản phẩm thông qua việc theo dõi thời gian build pipeline (các bước chạy test, build, chạy static code analytics (lint check)).

- Bên cạnh những ưu điểm giúp quy trình CI/CD đáng được cân nhắc để áp dụng trong tổ chức thì CI/CD vẫn có những hạn chế cần phải lưu ý như:

* Trong một dự án nếu có quá nhiều Developer cùng tham gia phát triển sản phẩm, sẽ có thời điểm phát sinh nhiều pull request cần được merge vào branch. Lúc này, các thành viên phải chờ pull request của người trước được merge hoàn tất, sau đó thực hiện update (cập nhật) lại source code (trong trường hợp có thông báo conflict từ Git repository) và phải trải qua các bước test lại từ đầu. Hệ quả là làm gián đoạn thời gian phát triển sản phẩm.
* Vì sử dụng dịch vụ CI/CD của bên service thứ 3 nên nếu service đó gặp vấn đề và bị crash, bị khai tử thì những dự án áp dụng CI/CD cũng bị ảnh hưởng khá nghiêm trọng.

**8.3. Cách thức triển khai của CI/CD**

- Quá trình CI gồm 6 giai đoạn chính bao gồm

+) Check out code : khi Developer tạo ra bất kỳ sự thay đổi trên hệ VCS thì VCS sẽ phát đi thông báo đến CI/CD tool là có thay đổi như thế. Phản hồi với thông báo, phía CI/CD sẽ tự động thực hiện các thao tác tải bản code mới đó về

+) Static code analysis : Code mới được review bằng tool (ví dụ như Sonarqube ) để phát hiện các chỗ chưa được tối ưu như lặp đoạn code, hàm quá nhiều tham số , lệnh if else quá dài.

+) Unit test : các đơn vị nhỏ nhất như hàm , class , biến được test trên CI server để xem chúng hoạt động đúng như yêu cầu không.

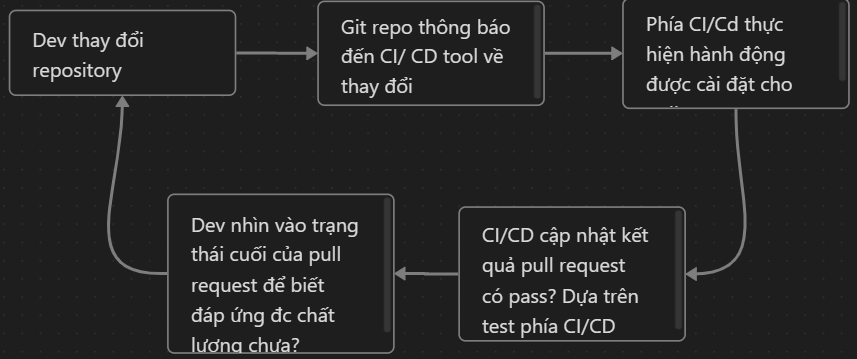
+)Build và Packaging : Code được build sử dụng phần mềm như Maven để đóng gói với thư viện và thông số môi trường để tạo thành container. Kết quả của giai đoạn này là ta có các image và container để triển khai lên môi trường test.

+) Triển khai lên môi trường test : từ các image ta chạy các container để triển khai lên các môi trường thật có kiểm soát kiểm tra độ tích hợp của chúng với nhau và có phát sinh lỗi không. Tùy thuộc vào mỗi môi trường mà có các lệnh test khác nhau.

+) Kiểm tra kết quả : thông qua các test plan và test case được tạo ra từ trước để quy trình CI kiểm tra chức năng của phần mềm chạy trên môi trường test.

- Quá trình CD cũng tương tự như CI chỉ khác là nó thêm hai bước đánh giá kết quả và triển khai lên môi trường production hay môi trường người dùng cuối. Tuy nhiên trong thực tế đa số các developer không thực hiện 2 bước cuối tự động mà do người lập trình tự thực hiện. Lý do vì rủi ro lỗi vẫn còn lớn khi triển khai lên môi trường người dùng cuối kể cả khi phần mềm đã qua tất cả các bài test trước. Vì thế , ở bài tập này nhóm sẽ chỉ dừng lại ở 6 bước đầu.

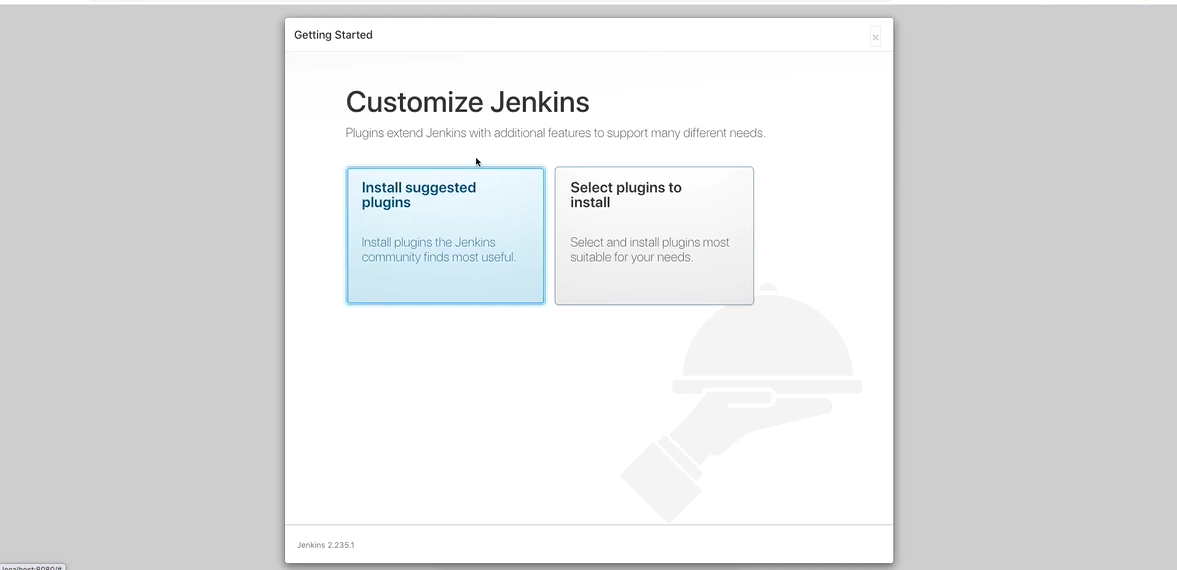
- Nhóm sử dụng Git là hệ VCS, lệnh thực hiện có thể ví dụ như là tạo pull request của Git repository. Sau khi thực hiện tất cả các lệnh đã được cài đặt kể trên thì CI/CD sẽ cập nhật kết quả ngược lại cho Git repository biết là pull request được tạo có vượt qua (pass) hết các quy trình (bao gồm testing) phía CI/CD hay không. Theo đó, khi review code, Reviewer chỉ cần nhìn vào trạng thái cuối cùng của pull request (passed hoặc failed) để biết pull request đã đáp ứng được chất lượng, đã tối ưu hay chưa.



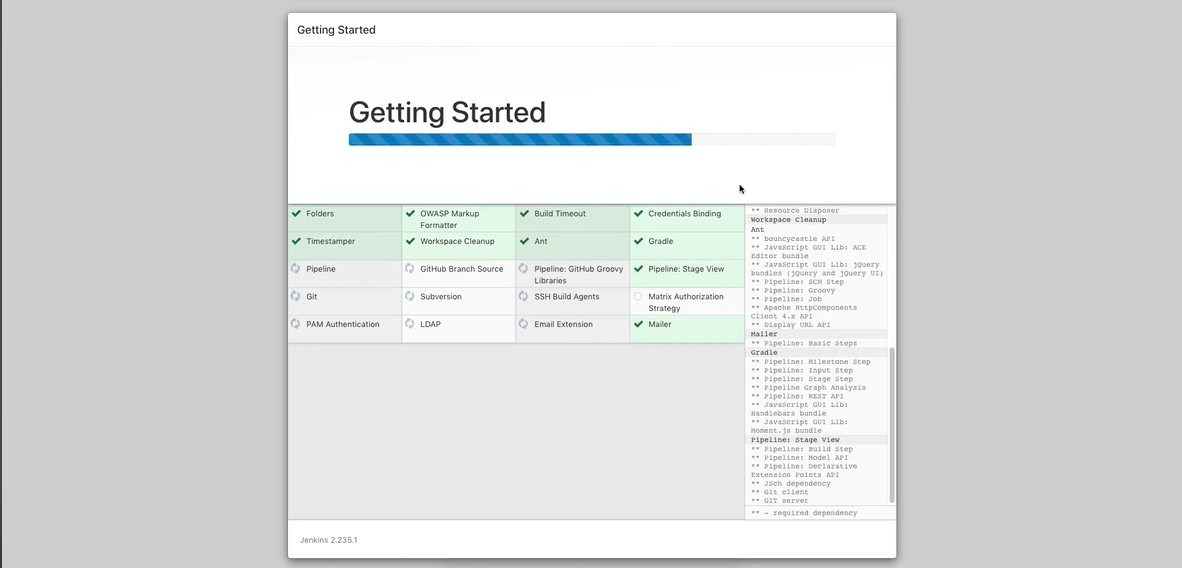
## **8.4. Cài đặt và sử dụng Jenkins**

- Sau khi đã cài đặt Docker ở phần trên, sử dụng Docker Hub để tải image của Jenkin xuống, thiết lập image để chạy container với cổng localhost 8080.

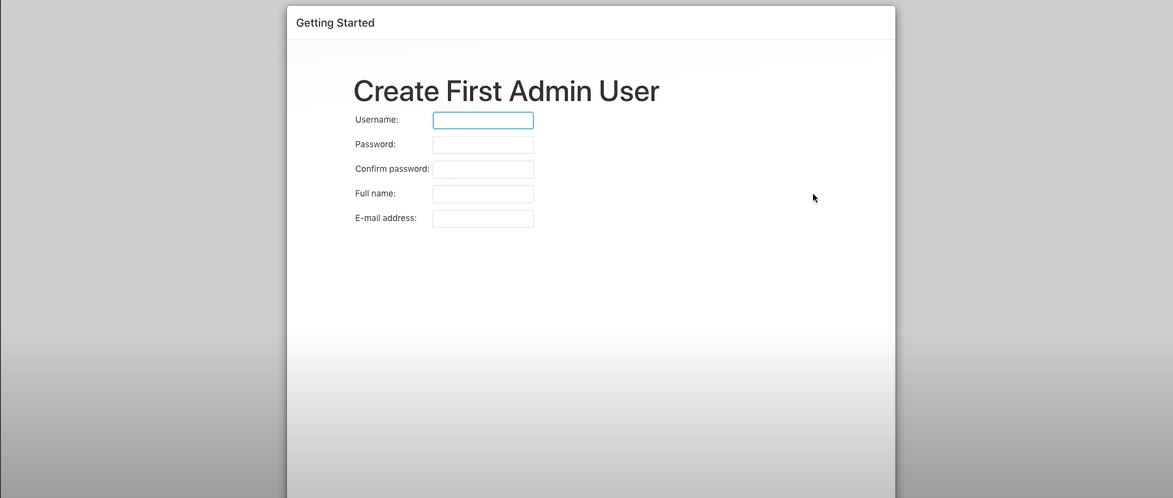
- Khi truy cập địa chỉ này, ta sẽ đưa tới màn hình set up cho Jenkins



- Lựa chọn install plugin theo suggeste và Jenkins sẽ bắt đầu tải về và cài đặt.



- Sau khi hoàn tất việc cài đặt , Jenkins sẽ yêu cầu thiết lập tài khoản cho Jenkins.



- Expose Local Host: Download công cụ ngrok và chạy lệnh ngrok http 8080 để expose local host và tạo 1 public URL

Text

Description automatically generated

- Tích hợp Jenkins và Github

+) Tạo một repo trên github

+) Tạo webhook mới trên github với đường link được public bởi ngrok

+) Truy cập Jenkins và tạo một project mới với source management là git

+) Push code lên repo và github sẽ gửi push request lên cho đường dẫn trên

Graphical user interface, text, application

Description automatically generated

- Jenkin sẽ nhận được sự kiện và kéo code từ github repo mà tạo từ trước về rồi clone repo, build và test code. Nếu build thành công thì báo về

Graphical user interface, text, application, email

Description automatically generated

- Jenkins clone lại nội dung vào container

# **IX. Demo và kết luận**

# **Tài liệu tham khảo**

1. Step by Step: Create Node.js REST API with SQL Server Database : <https://www.telerik.com/blogs/step-by-step-create-node-js-rest-api-sql-server-database>
2. How to Create a Express/Node + React Project | Node Backend + React Frontend - Arpan Neupane
3. Docker documentation : <https://docs.docker.com/desktop/>
4. Learn Docker in 7 Easy Steps - Full Beginner's Tutorial - Fireship
5. Jenkins CI/CD - TechMaster VietNam