

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CNTT2

XÂY DỰNG WEBSITE ĐĂNG TIN TUYỂN DỤNG CỦA DOANH NGHIỆP CHO SINH VIÊN TDTU SỬ DỤNG CÔNG NGHỆ NODEJS VÀ REACT

Người hướng dẫn: GV DOÃN XUÂN THANH

Người thực hiện: NGUYỄN HỮU TÂN ĐẠT - 51702075

NGÔ MINH HIẾU - 51702017

Lớp : 17050201

Khoa : 21

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN CNTT2

XÂY DỰNG WEBSITE ĐĂNG TIN TUYỂN DỤNG CỦA DOANH NGHIỆP CHO SINH VIÊN TDTU SỬ DỤNG CÔNG NGHỆ NODEJS VÀ REACT

Người hướng dẫn: GV DOÃN XUÂN THANH

Người thực hiện: NGUYỄN HỮU TÂN ĐẠT - 51702075

NGÔ MINH HIẾU - 51702017

Lớp : 17050201

Khoá : 21

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến thầy **Doãn Xuân Thanh** – Giảng viên hướng dẫn của chúng em trong việc thực hiện dự án công nghệ thông tin 2, và khoa Công Nghệ Thông Tin đã cho em cơ hội thực hiện đề tài này và cũng như là cơ hội em được tiếp xúc với một số công nghệ được xem là trang bị cho em trong môi trường làm việc thực tế, cũng như trong việc hoàn thành đề tài xây dựng trang ứng dụng này.

Vì áp dụng 1 số kiến thức ở trường cũng như tìm hiểu và ứng dụng một số công nghệ mới nên chúng em sẽ không tránh khỏi được những sai sót và sự hạn chế của kiến thức tự tìm hiểu, đánh giá và trình bày về đề tài này. Rất mong nhận được sự quan tâm, góp ý của các thầy/cô để chung em biết mình thiếu sót những gì và giúp cho đề tài của chúng em được đầy đủ và hoành chính hơn. Chúng em xin chân thành cảm ơn.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của GV Doãn Xuân Thanh ;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 03 tháng 12 năm 2022

Tác giả

(ký tên và ghi rõ họ tên)

Đạt

Nguyễn Hữu Tân Đạt

Hiếu

Ngô Minh Hiếu

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

- Giới thiệu các công nghệ và một số thuật ngữ được sử dụng trong quá trình xây dựng trang ứng dụng
 - o Docker, CI/CD, VPS, NGINX, NODEJS
- Xác định các tính năng cần có , xây dựng các bảng dữ liệu lưu trữ cần có để thực hiện hành viết API và giao diện tương ứng
- Quá trình xây dựng API và cách gọi đến API
- Quá trình xây dựng Giao diện.

MỤC LỤC

LỜI CẢM ƠN	i
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN	ii
TÓM TẮT	iii
MỤC LỤC	1
DANH MỤC CÁC BẢNG BIẾU, HÌNH VẼ, ĐỒ THỊ	4
CHƯƠNG 1 – GIỚI THIỆU	8
1. GIỚI THIỆU VỀ ĐỀ TÀI	8
1) Đặc tả	8
2) Usecase	20
3) Activity	41
2. GIỚI THIỆU VỀ CÔNG NGHỆ VÀ THUẬT NGỮ	62
1) Docker	62
2) CI/CD	66
3) VPS	72
4) NGINX	76
5) NODEJS(EXPRESS)	78
6) ORM (PRISMA)	79
7) ReactJS	86
8) Redux	98
9) Typecript	105
10) Antd	118
3. Nội dung của chương này	119
CHƯƠNG 2 – QUÁ TRÌNH THIẾT LẬP CÁC CÔNG CỤ	120
1. Quá trình thiết lập server	120
2. Cách cài đặt gitlab CI/CD vào source code	128
3. Khởi tạo React App	144

4.	Cấu trúc source code front end	145
5.	Cấu hình scss.....	145
6.	Cấu hình router.....	146
CHƯƠNG 3 – KẾT QUẢ VÀ TỔNG KẾT		149
1.	Kết quả	149
1.1	Back-end	149
1.2	Front-end.....	190

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

CÁC CHỮ VIẾT TẮT

CI/CD Chỉ quá trình tích hợp liên tục và phân phối liên tục

JS Là một ngôn ngữ lập trình JavaScript

TS Là một ngôn ngữ lập trình được phát triển bởi Microsoft và có cách viết tương tự như JavaScript nhưng chặt chẽ hơn và dễ phát hiện lỗi sai trong khi viết

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

Hình 1 : Sơ đồ usecase của trang ứng dụng	20
Hình 2 : Sơ đồ Activity – Đăng ký.....	41
Hình 3 : Sơ đồ Activity – Đăng Nhập	41
Hình 4 : Sơ đồ Activity – Tìm Kiếm.....	42
Hình 5 : Sơ đồ Activity – Đăng tải CV	42
Hình 6 : Sơ đồ Activity – Xóa CV	43
Hình 7 : Sơ đồ Activity – Đổi Tên CV	43
Hình 8 : Sơ đồ Activity – Nộp CV	44
Hình 9 : Sơ đồ Activity – Rút CV	44
Hình 10 : Sơ đồ Activity – Đánh Dấu Bài Đăng.....	45
Hình 11 : Sơ đồ Activity – Hủy Đánh Dấu Bài Đăng.....	45
Hình 12 : Sơ đồ Activity – Xem Các Bài Viết Đã Nộp CV.....	46
Hình 13 : Sơ đồ Activity – Xem Các Bài Viết Đã Đánh Dấu.....	46
Hình 14 : Sơ đồ Activity – xóa bài đăng tuyển dụng.....	47
Hình 15 : Sơ đồ Activity – sửa bài đăng tuyển dụng	48
Hình 16 : Sơ đồ Activity – xem các đơn ứng tuyển của ứng viên	49
Hình 17 : Sơ đồ Activity – cập nhật lại mật khẩu mới.....	50
Hình 18 : Sơ đồ Activity – cập nhật lại thông tin của người dùng	51
Hình 19 : Sơ đồ Activity – xem danh sách người dùng trong hệ thống.....	52
Hình 20 : Sơ đồ Activity – xem danh sách bài đăng tuyển dụng của tất cả nhà tuyển dụng	53
Hình 21 : Sơ đồ Activity – cập nhật thông tin nhóm tài khoản	54
Hình 22 : Sơ đồ Activity – thêm mới loại nghề nghiệp	55
Hình 23 : Sơ đồ Activity – cập nhật lại loại nghề nghiệp	56
Hình 24 : Sơ đồ Activity – xóa loại nghề nghiệp.....	57

Hình 25 : Sơ đồ Activity – thêm mới chuyên ngành	58
Hình 26 : Sơ đồ Activity – cập nhật lại thông tin chuyên ngành	59
Hình 27 : Sơ đồ Activity – xóa loại nghề nghiệp.....	60
Hình 28 : Sơ đồ Activity – quên mật khẩu.....	61
Hình 29: Sơ đồ về kiến trúc của docker	64
Hình 30: Chi tiết và cách hoạt động và thao tác với docker	65
Hình 31: Quá trình hoạt động của CI/CD	67
Hình 32: Quá trình phát triển ứng dụng	68
Hình 33: Cách hoạt động của CI/CD	70
Hình 34: Tổng quan về quá trình hoạt động với CI/CD	71
Hình 35 : Nginx là gì.....	76
Hình 36: Mô hình hoạt động của Redux	99
Hình 37: Sơ đồ hoạt động của Redux.....	100
Hình 38: Hình ảnh thuê server	120
Hình 39: Gõ ssh-keygen để tạo ra bộ mã cho riêng mình.....	121
Hình 40: Thư mục .ssh chứa ssh-keygen	121
Hình 41: Hình ảnh tạo ssh key trong thư mục demo.....	122
Hình 42: Hình ảnh thư mục demo chứa ssh key ta vừa tạo với demo	122
Hình 43: Kết nối với server thuê được bằng ssh	123
Hình 44: Thêm ssh key vào máy chủ	124
Hình 45: Nội dung tập tin authorized_keys	125
Hình 46: Ta sẽ copy key trong file private key	126
Hình 47: Hình ảnh thể hiện máy chủ có các container đang hoạt động.....	128
Hình 48: Bước một tiến hành tạo dự án mới trên GitLab	128
Hình 49: Bước hai chọn chạy dự án mở rộng sử dụng CI/CD.....	129
Hình 50 : Bước ba chọn sử dụng dự án từ nền tảng GitHub.....	129
Hình 51: Màn hình yêu cầu ta nhập token ủy quyền	130

Hình 52: Bước bốn quay lại github để cài đặt.....	130
Hình 53: Bước năm chọn vào mục Developer settings.....	130
Hình 54: Tạo token ủy quyền.....	131
Hình 55: Bước tiếp theo điền các thông tin khi tạo token ủy quyền.....	131
Hình 56 : Sau khi khởi tạo token ta sẽ nhận được 1 dãy chữ cái ngẫu nhiên	132
Hình 57: Điền token ủy quyền cho gitlab	132
Hình 58: Màn hình sau khi ủy quyền	133
Hình 59: Dự án từ Github đã được đưa sang GitLab	134
Hình 60 Home Page - Banner	190
Hình 61 Home Page – Top Major	191
Hình 62: Home Page – Featured Job.....	192
Hình 63: Home Page – Recruiment	193
Hình 64: Home Page – How It Work.....	194
Hình 65: Footer	194
Hình 66: Header - UnAuth.....	195
Hình 67: Header - Candidate.....	195
Hình 68: Header – Recruiter	195
Hình 69: Header - Admin.....	195
Hình 70: Search page	196
Hình 71: Job Detail	199
Hình 72: Candidate – Profile.....	200
Hình 73: Candidate – CV Manager.....	201
Hình 74: Candidate – Bookmarks	203
Hình 75: Candidate – Applied Job	203
Hình 76: Candidate – Change passwords	204
Hình 77: Recruiter - Profile.....	204
Hình 78: Recruiter – Post Manager.....	205

Hình 79: Recruiter – New post step one	205
Hình 80: Recruiter – New post step two	206
Hình 81: Recruiter – New post step three	206
Hình 82: Recruiter – Edit Post	207
Hình 83: Recruiter – All Applicant.....	208
Hình 84: Recruiter – Change Password	208
Hình 85: Admin – Account Manager.....	209
Hình 86: Admin - Account Detail – Recruiter	210
Hình 87: Admin - Account Detail – Candidate.....	210
Hình 88: Admin – All Post.....	211
Hình 89: Admin – Account Type Manager.....	211
Hình 90: Admin - Job Type Manager	212

CHƯƠNG 1 – GIỚI THIỆU

1. GIỚI THIỆU VỀ ĐỀ TÀI

Với nhu cầu tìm việc của nguồn lực trường, Trường đại học Tôn Đức Thắng muốn tăng cường cơ hội tìm việc cũng như có thêm một kênh thông tin về cơ hội việc làm cùng với các kênh khác như topcv, glint, hoặc facebook. Nhằm để đáp ứng với nhu cầu đó thì chung em sẽ xem và xây dựng một kênh tìm việc cũng như đăng các tin tuyển dụng của các nhà tuyển dụng lên để tạo cơ hội việc làm cho sinh viên của trường

Kênh tìm kiếm việc này sẽ cho phép sinh viên tạo hồ sơ ứng tuyển của riêng mình cũng như dễ dàng tìm được công việc phù hợp dựa vào nguồn việc làm từ các đối tác lớn của nhà trường cũng như các công ty nhỏ và cần nguồn nhân lực chất lượng cao

1) Đặc điểm

Để đáp ứng với nhu cầu xây dựng một trang ứng dụng cho phép sinh viên có thể tham gia ứng tuyển vào các công việc tiềm năng do các đối tác của nhà trường đăng tin tuyển dụng , Sinh viên có thể tiếp xúc với nguồn việc làm ổn định và chất lượng cũng như cung cấp cho nhà tuyển dụng nguồn nhân lực tốt nhất. Trang ứng dụng này được xây dựng dựa trên dữ liệu trong các bảng như sau

- Loại người dùng (User_Type)
 - id : để định danh - **Bigint autoincrement**
 - user_type_name : Tên của loại người dùng – **Nvarchar(1000)**
 - is_active : Loại người dùng có được kích hoạt hay không – **Bit (0 or 1)**
 - is_delete : Loại người dùng đã được xóa hay không – **Bit(0 or 1)**
 - create_date : Ngày khởi tạo loại người dùng – **Datetime**
 - create_user : Tên tài khoản tạo ra loại người dùng này - **Nvarchar(1000)**
 - update_date : Ngày cập nhật - **Datetime**
 - update_user : Người cập nhật – **Nvarchar(1000)**
 - delete_date : Ngày xóa - **Datetime**

- delete_user : Người xóa – **Nvarchar(1000)**

Bảng Loại người dùng có chức năng chia và phân nhóm có các loại tài khoản được đăng ký ở trang web như là người quản trị (Admin) , Nhà tuyển dụng (Recruiter) và ứng viên (Candidate) giúp cho với mỗi loại người dùng sẽ có các đặc quyền riêng và giúp cho quá trình sử dụng trong trang web một cách thuận tiện nhất

- Tài khoản người dùng (User_Account)

- id : id định danh tài khoản – Bigint acutoincrement
- password : Mật khẩu đăng nhập – **Nvarchar(1000)**
- username : Tên đăng nhập – **Nvarchar(1000)**
- google_id : mã id của google - **Nvarchar(1000)**
- is_active : Tài khoản có được kích hoạt hay không - **Bit (0 or 1)**
- is_delete : Tài khoản có bị xóa hay không - **Bit (0 or 1)**
- create_date : Ngày tạo của tài khoản - **Datetime**
- user_type_id : id loại người dùng - **Datetime**
- first_name : Họ người dùng - **Nvarchar(1000)**
- last_name : Tên người dùng - **Nvarchar(1000)**
- full_name : Họ và tên đầy đủ - **Nvarchar(1000)**
- email : Email của người dùng và email sẽ đóng vai trò là nơi nhận thông báo từ trang ứng dụng - **Nvarchar(1000)**
- number_phone : Số điện thoại của người dùng - **Nvarchar(1000)**
- age : tuổi – **Int**
- gender : Giới tính - **Int**
- province_code : ID của tỉnh - **Nvarchar(1000)**
- district_code : ID của quận/huyện - **Nvarchar(1000)**
- ward_code : ID của phường/xã - **Nvarchar(1000)**

- address : địa chỉ cụ thể - **Nvarchar(1000)**
- avatar : Đường dẫn lưu trữ tấm ảnh của người dùng - **Nvarchar(1000)**

Bảng tài khoản người dùng có chức năng lưu trữ các thông tin của người dùng giúp cho người dùng có thể đăng nhập vào trang web và giao tiếp với nhà tuyển dụng cũng như sử dụng các chức năng có của trang web

- Loại công việc (Job_Type)

- id : id định danh loại công việc - **Bigint autoincrement**
- job_type_name : Tên loại công việc - **Nvarchar(1000)**
- is_active : Loại công việc được kích hoạt hay không – **Bit (0 or 1)**
- is_delete : Loại công việc đã được xóa hay không – **Bit(0 or 1)**
- create_date : Ngày khởi tạo – **Datetime**
- create_user : Người khởi tạo - **Nvarchar(1000)**
- update_date : Ngày cập nhật - **Datetime**
- update_user : Người cập nhật – **Nvarchar(1000)**
- delete_date : Ngày xóa - **Datetime**
- delete_user : Người xóa – **Nvarchar(1000)**

Bảng loại công việc có chức năng lưu trữ các loại công việc được người quản trị khai báo, từ đó trên trang ứng dụng có thể thêm loại công việc mà người tuyển dụng muốn nhắm đến vào bài đăng tuyển dụng, giúp cho ứng viên có thể tìm kiếm các loại công việc mình muốn nộp đơn ứng tuyển như full time hay parttime

- Chuyên ngành (Majors)

- id : id định danh chuyên ngành - **Bigint autoincrement**
- job_type_name : Tên chuyên ngành- **Nvarchar(1000)**
- is_active : Chuyên ngành được kích hoạt hay không – **Bit (0 or 1)**
- is_delete : Chuyên ngành đã được xóa hay không – **Bit(0 or 1)**
- create_date : Ngày khởi tạo– **Datetime**
- create_user : Người khởi tạo - **Nvarchar(1000)**
- update_date : Ngày cập nhật - **Datetime**
- update_user : Người cập nhật – **Nvarchar(1000)**
- delete_date : Ngày xóa - **Datetime**
- delete_user : Người xóa – **Nvarchar(1000)**

Bảng chuyên ngành có nhiệm vụ lưu trữ các chuyên ngành có trên thị trường do người quản trị thêm vào giúp cho nhà tuyển dụng về chuyên ngành đó có thể thêm loại chuyên ngành vào bài đăng của mình cũng như giúp ứng viên học về chuyên ngành đó hoặc muốn có công việc về chuyên ngành đó có thể dễ dàng tìm kiếm được bài tuyển dụng một cách hợp lí

- Hồ sơ ứng tuyển (Apply_Profile)

- profile_id : id định danh hồ sơ - **Bigint autoincrement**
- profile_name : Tên hồ sơ ứng tuyển- **Nvarchar(1000)**
- is_active : Hồ sơ ứng tuyển có được kích hoạt hay không – **Bit (0 or 1)**
- is_delete : Hồ sơ ứng tuyển đã được xóa hay không – **Bit(0 or 1)**
- create_date : Ngày khởi tạo– **Datetime**
- create_user : Người khởi tạo - **Nvarchar(1000)**
- update_date : Ngày cập nhật - **Datetime**
- update_user : Người cập nhật – **Nvarchar(1000)**

- delete_date : Ngày xóa - **Datetime**
- delete_user : Người xóa – **Nvarchar(1000)**
- user_id : Hồ sơ thuộc về tài khoản nào – **Bigint**

Bảng hồ sơ ứng tuyển được dùng vào mục đích là giúp cho ứng viên có thể tạo cho mình một hồ sơ có thể thu hút được nhà tuyển dụng hoặc được dùng để nộp vào các bài đăng phù hợp với mình

- Bài tuyển dụng (Recruitment_Post)
 - id : id định danh chuyên ngành - **Bigint autoincrement**
 - content : Nội dung bài viết - **Nvarchar(4000)**
 - title : Tựa đề bài viết - **Nvarchar(1000)**
 - recruiter_id : Id của nhà tuyển dụng - **Bigint**
 - to_value : Mức lương khởi điểm - **Bigint**
 - from_value : Mức lương tối đa - **Bigint**
 - gender : Giới tính tuyển chọn - **Int**
 - province_code : ID của tỉnh - **Nvarchar(1000)**
 - district_code : ID của quận/huyện - **Nvarchar(1000)**
 - ward_code : ID của phường/xã - **Nvarchar(1000)**
 - address : địa chỉ cụ thể - **Nvarchar(1000)**
 - is_active : Bài đăng có được kích hoạt hay không – **Bit (0 or 1)**
 - is_delete : Bài đăng đã được xóa hay không – **Bit(0 or 1)**
 - create_date : Ngày khởi tạo– **Datetime**
 - create_user : Người khởi tạo - **Nvarchar(1000)**
 - update_date : Ngày cập nhật - **Datetime**
 - update_user : Người cập nhật – **Nvarchar(1000)**
 - delete_date : Ngày xóa - **Datetime**

- delete_user : Người xóa – **Nvarchar(1000)**

Bảng bài tuyển dụng có công dụng là với mỗi nhà tuyển dụng có thể tạo ra bài đăng của mình và tìm được các ứng viên tiềm năng giúp tăng khả năng có được các ứng viên chất lượng đồng thời giúp cho ứng viên hiểu về công việc mà mình có thể nộp đơn vào với mức lương trong khoảng bao nhiêu , của nhà tuyển dụng nào , và cũng như là địa điểm làm việc ở đâu , giới tính mà bài đăng tập trung vào là gì?.

- Lịch sử ứng tuyển (History_Apply_Job)

- id : Mã định danh lịch sử ứng tuyển - **Bigint autoincrement**
- user_id : id của người dùng - **Bigint**
- post_id : id bài tuyển dụng - **Bigint**
- cv_id : id của cv ứng tuyển - **Bigint**
- is_active : Lịch sử ứng tuyển được kích hoạt hay không – **Bit (0 or 1)**
- is_delete : Lịch sử ứng tuyển đã được xóa hay không – **Bit(0 or 1)**
- create_date : Ngày khởi tạo– **Datetime**
- create_user : Người khởi tạo - **Nvarchar(1000)**
- update_date : Ngày cập nhật - **Datetime**
- update_user : Người cập nhật – **Nvarchar(1000)**
- delete_date : Ngày xóa - **Datetime**
- delete_user : Người xóa – **Nvarchar(1000)**

Bảng lịch sử ứng tuyển giúp cho ứng viên có thêm xem lại những cơ hội mình đã nộp đơn

- Loại công việc của bài viết (Recruitment_Post_Job_Type)
 - id : Mã định danh loại công việc của bài viết - **Bigint autoincrement**
 - post_id : id của bài tuyển dụng - **Bigint**
 - job_type_id :id của loại công việc - **Bigint**
 - is_active : Loại công việc của bài viết được kích hoạt hay không – **Bit (0 or 1)**
 - is_delete : Loại công việc của bài viết đã được xóa hay không – **Bit(0 or 1)**
 - create_date : Ngày khởi tạo– **Datetime**
 - create_user : Người khởi tạo - **Nvarchar(1000)**
 - update_date : Ngày cập nhật - **Datetime**
 - update_user : Người cập nhật – **Nvarchar(1000)**
 - delete_date : Ngày xóa - **Datetime**
 - delete_user : Người xóa – **Nvarchar(1000)**

Bảng loại công việc của bài viết có nhiệm vụ lưu trữ tất cả các loại công việc được thêm vào bài viết vì 1 bài viết có thể có nhiều loại công việc nên nhiệm vụ của bảng này là bảng nối để lưu được mối quan hệ 1 – n của bài viết và loại công việc

- Loại chuyên ngành của bài viết (Recruitment_Post_Majors)
 - id : Mã định danh chuyên ngành của bài viết - **Bigint autoincrement**
 - post_id : id của bài viết
 - majors_id : id của chuyên ngành
 - is_active : Loại chuyên ngành của bài viết có kích hoạt hay không – **Bit (0 or 1)**
 - is_delete : Loại chuyên ngành của bài viết đã được xóa hay không – **Bit(0 or 1)**

- `create_date` : Ngày khởi tạo – **Datetime**
- `create_user` : Người khởi tạo - **Nvarchar(1000)**
- `update_date` : Ngày cập nhật - **Datetime**
- `update_user` : Người cập nhật – **Nvarchar(1000)**
- `delete_date` : Ngày xóa - **Datetime**
- `delete_user` : Người xóa – **Nvarchar(1000)**

Bảng loại chuyên ngành của bài viết có nhiệm vụ lưu trữ tất cả các loại chuyên ngành được thêm vào bài viết vì 1 bài viết có thể thuộc về nhiều chuyên ngành nên nhiệm vụ của bảng này là bảng nối để lưu được mỗi quan hệ 1 – n của bài viết và loại chuyên ngành

- Đơn ứng tuyển (cv)

- `id` : Mã định danh đơn - **Bigint autoincrement**
- `file_name_hash` : Tên ngẫu nhiên của cv trên server - **Nvarchar(1000)**
- `file_name` : tên file gốc - **Nvarchar(1000)**
- `user_id` : id user của cv - **Bigint**
- `extname` : Tên mở rộng của file - **Nvarchar(1000)**
- `is_active` : Đơn ứng tuyển có kích hoạt hay không – **Bit (0 or 1)**
- `is_delete` : Đơn ứng tuyển đã được xóa hay không – **Bit(0 or 1)**
- `create_date` : Ngày khởi tạo – **Datetime**
- `create_user` : Người khởi tạo - **Nvarchar(1000)**

Bảng cv có nhiệm vụ với mỗi ứng viên cần có nhiều cv và với mỗi cv trong đó sẽ tóm tắt về bản thân , trình độ học vấn , sở thích , cũng như là quá trình làm việc của bản thân , bảng này sẽ lưu đường dẫn tập tin Người dùng sẽ tải lên cv của cá nhân và bảng sẽ lưu đường dẫn cũng như tên của cv ở server người dùng có thể dùng nó để ứng tuyển , hoặc là tải xuống , xem lại

- Khu vực hành chính (administrative_region)
 - id : id định danh khu vực hành chính - **Int**
 - name : Tên khu vực hành chính - **Nvarchar(1000)**
 - name_en : Tên khu vực hành chính tiếng anh - **Nvarchar(1000)**
 - code_name : Mã khu vực hành chính - **Nvarchar(1000)**
 - code_name_en : Mã khu vực hành chính tiếng anh - **Nvarchar(1000)**

Bảng khu vực hành chính lưu các khu vực của nước Việt Nam như là Đông Nam Bộ , Đồng Bằng Sông Cửu Long

- Đơn vị hành chính (administrative_units)
 - id : id định danh đơn vị hành chính - **Int**
 - full_name : Tên đầy đủ đơn vị hành chính - **Nvarchar(1000)**
 - full_name_en : Tên đầy đủ đơn vị hành chính với tiếng anh - **Nvarchar(1000)**
 - short_name : Tên gọi ngắn của đơn vị hành chính - **Nvarchar(1000)**
 - short_name_en : Tên gọi ngắn tiếng anh đơn vị hành chính - **Nvarchar(1000)**
 - code_name : Mã tên gọi của đơn vị hành chính- **Nvarchar(1000)**
 - code_name_en : Mã tên gọi tiếng anh của đơn vị hành chính- **Nvarchar(1000)**

Bảng Đơn vị hành chính sẽ lưu tên các đơn vị hành chính của nước Việt Nam như là Tỉnh , Thành Phố trực thuộc trung ương,

- Tỉnh thành phố (provinces)
 - id : Mã định danh tỉnh thành phố - **Int**

- code : Mã của tỉnh thành phố - **Nvarchar(1000)**
- name : Tên của tỉnh thành phố- **Nvarchar(1000)**
- name_en: Tên tiếng anh của tỉnh thành phố- **Nvarchar(1000)**
- full_name : Tên đầy đủ của tỉnh thành phố - **Nvarchar(1000)**
- full_name_en : Tên tiếng anh đầy đủ của tỉnh thành phố- **Nvarchar(1000)**
- code_name : Tên mã của tỉnh thành phố- **Nvarchar(1000)**
- administrative_unit_id : Mã đơn vị hành chính - **Int**
- adminstrartive_region_id : Mã khu vực hành chính – **Int**

Bảng tỉnh thành phố sẽ lưu lại tên , mã tỉnh phố , của 63 tỉnh thành của Việt Nam

- Quận/Huyện (districts)

- id : Mã định danh quận huyện - **Int**
- code : Mã của quận huyện- **Nvarchar(1000)**
- name : Tên của tỉnh quận huyện- **Nvarchar(1000)**
- name_en: Tên tiếng anh của quận huyện - **Nvarchar(1000)**
- full_name : Tên đầy đủ của quận huyện - **Nvarchar(1000)**
- full_name_en : Tên tiếng anh đầy đủ của quận huyện- **Nvarchar(1000)**
- code_name : Tên mã của quận huyện - **Nvarchar(1000)**
- administrative_unit_id : Mã đơn vị hành chính - **Int**
- adminstrartive_region_id : Mã khu vực hành chính – **Int**
- province_code : Mã tỉnh quận huyện thuộc - **Nvarchar(1000)**

Bảng Quận/Huyện lưu trữ thông tin của từng quận/Huyện trong 63 tỉnh thành của Việt Nam và đơn vị hành chính

- Phường/Xã (Wards)

- id : Mã định danh phường xã - **Int**

- code : Mã của phường xã- **Nvarchar(1000)**
- name : Tên của phường xã- **Nvarchar(1000)**
- name_en: Tên tiếng anh của phường xã- **Nvarchar(1000)**
- full_name : Tên đầy đủ của phường xã - **Nvarchar(1000)**
- full_name_en : Tên tiếng anh đầy đủ của phường xã- **Nvarchar(1000)**
- code_name : Tên mã của phường xã - **Nvarchar(1000)**
- district_code : Mã Quận/Huyện thuộc về - **Nvarchar(1000)**
- administrative_unit_id : Mã đơn vị hành chính – **Int**

Bảng Phường/Xã sẽ lưu thông tin của phường xã của nhiều Quận/Huyện của tỉnh và đơn vị hành chính

- Bài tuyển dụng đánh dấu của người dùng (Recruitment_Post_User_Like)
 - id : Mã định danh lượt đánh dấu bài tuyển dụng - **Bigint autoincrement**
 - user_id : Mã định danh của người đánh dấu- **Bigint**
 - post_id : Mã định danh của bài viết- **Bigint**
 - is_active : Lượt đánh dấu có kích hoạt hay không – **Bit (0 or 1)**
 - is_delete : Lượt đánh dấu đã được xóa hay không – **Bit(0 or 1)**
 - create_date : Ngày khởi tạo– **Datetime**
 - create_user : Người khởi tạo - **Nvarchar(1000)**
 - update_date : Ngày cập nhật - **Datetime**
 - update_user : Người cập nhật – **Nvarchar(1000)**
 - delete_date : Ngày xóa - **Datetime**
 - delete_user : Người xóa – **Nvarchar(1000)**

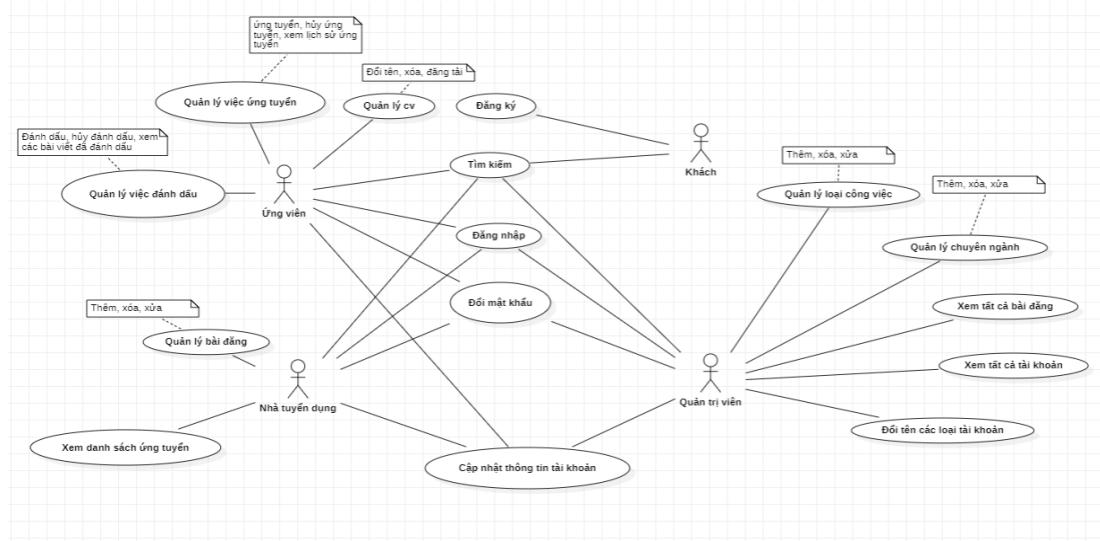
Bảng lưu bài viết được đánh dấu bởi người dùng có chức năng lưu trữ thông tin bài đăng người ứng viên cảm thấy phù hợp nhưng chưa muộn ứng tuyển và sơ việc không tìm thấy được bài đăng thì người ứng viên có thể đánh dấu và ứng tuyển sau

Người dùng sẽ được chia làm 3 nhóm tài khoản và thao tác với một số chức năng nhất định là

- **Ứng viên :** Có thể là sinh viên của trường hoặc là các ứng viên có nhu cầu tham gia vào trang web thông qua việc đăng ký
- **Nhà tuyển dụng :** Là các đối tác của quý nhà trường về các vị trí việc làm chất lượng và có nhu cầu tìm đến các nguồn nhân lực trẻ mới ra trường chưa có kinh nghiệm để tuyển làm thực tập sinh, hoặc là Fresher để đào tạo cho phù hợp với nhu cầu của danh nghiệp
- **Người quản lý:** Có vai trò là người quan sát , giám sát các hoạt động của trang web cũng như khai báo và quản lý các tài nguyên của trang ứng dụng để giúp cho cả 2 loại người dùng là ứng viên và nhà tuyển dụng có thể tiếp xúc với nhau một cách thuận tiện nhất

2) Usecase

- Sơ đồ usecase



Hình 1 : Sơ đồ usecase của trang ứng dụng

- Đặt tả usecase

- Đăng nhập

Use case name	Đăng Nhập	
Scenario	Cần đăng nhập khi muốn thực hiện các tính đăng cù thể của từng loại người dùng	
Triggering Event	Khi người dùng có nhu cầu thao tác với các tác vụ bên trong thì cần thực hiện đăng nhập	
Brief Description	Người dùng có thể chọn trang đăng nhập để đăng nhập	
Actor	Admin, Candidate, Recruiter	
Precondition	Người dùng phải nhập đúng tài khoản và mật khẩu mới có thể vào được chương trình	
Flow of Event	Actor 1. Người dùng vào trang đăng nhập 2. Thực hiện thao tác đăng nhập vào hệ thống thông qua tên tài khoản và mật khẩu đã đăng ký	System 2.1 Hệ Thống sẽ thực hiện việc kiểm tra tên tài khoản và mật khẩu mà nhân viên vừa nhập vào có trong kho dữ liệu hay không

Exception Condition	2.2 Nếu người dùng thực hiện việc đăng nhập không thành công thì sẽ hiện thông báo là tài khoản mật khẩu không hợp lệ
----------------------------	---

- Đăng ký

Use case name	Đăng Ký	
Scenario	Khi người dùng muốn đăng ký tài khoản	
Triggering Event	Khi người dùng có nhu cầu thao tác với các tác vụ bên trong thì cần thực hiện đăng ký tài khoản nếu chưa có tài khoản	
Brief Description	Người dùng có thể chọn trang đăng ký để đăng ký	
Actor	Người dùng thường	
Precondition	Người dùng phải nhập các thông tin cần thiết	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người dùng vào trang đăng ký 2. Đăng ký tên và mật khẩu 3. Điền các thông tin cần thiết 	<ol style="list-style-type: none"> 2.1 Hệ Thống sẽ thực hiện việc kiểm tra tên tài khoản và email xem đã có trong hệ thống chưa, nếu chưa sẽ thêm tài khoản vào hệ thống
Exception Condition	2.2 Nếu người dùng đăng ký với tên đăng nhập và email đã có trong hệ thống sẽ báo với người dùng email hoặc tên tài khoản đã tồn tại	

- Search

Use case name	Search	
Scenario	Tìm kiếm các bài đăng theo nhiều tiêu chí	
Triggering Event	Khi người dùng có nhu cầu tìm kiếm các bài đăng	
Brief Description	Người dùng truy cập trang search	
Actor	Người dùng thường, admin, candidate, recruiter	
Precondition		
Flow of Event	Actor	System
	1. Người dùng vào trang search 2. Tùy chỉnh bộ lọc theo tiêu chí	2.1 Hệ Thống sẽ tìm kiếm và trả về các bài đăng đáp ứng đủ điều kiện mà người dùng đã lọc
Exception Condition		

- Đăng tải CV

Use case name	Upload cv	
Scenario	Candidate upload cv	
Triggering Event	Khi candidate có nhu cầu upload cv	
Brief Description	Người dùng truy cập trang trang candidate và upload cv	
Actor	Candidate	
Precondition	Người dùng phải đăng nhập với loại tài khoản Candidate	
Flow of Event	Actor	System
	1. Người dùng đăng nhập vào hệ thống với tài khoản candidate 2. Người dùng vào trang candidate 3. Vào tab cv manager 4. Upload cv	2.1 Hệ thống sẽ lưu dữ liệu vào cơ sở dữ liệu
Exception Condition		

- Đổi tên CV

Use case name	Đổi tên cv	
Scenario	Candidate đổi tên cv	
Triggering Event	Khi candidate có nhu cầu đổi tên cv	
Brief Description	Người dùng truy cập trang candidate và đổi tên cv	
Actor	Candidate	
Precondition	Người dùng phải đăng nhập với loại tài khoản Candidate	
Flow of Event	Actor	System
	1. Người dùng đăng nhập vào hệ thống với tài khoản candidate 2. Người dùng vào trang candidate 3. Vào tab cv manager 4. Đổi tên cv	2.1 Hệ thống sẽ lưu dữ liệu mới vào cơ sở dữ liệu
Exception Condition		

- Xóa CV

Use case name	Xóa cv	
Scenario	Candidate xóa cv	
Triggering Event	Khi candidate có nhu cầu xóa cv	
Brief Description	Người dùng truy cập trang candidate và xóa cv	
Actor	Candidate	
Precondition	Người dùng phải đăng nhập với loại tài khoản Candidate	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người dùng đăng nhập vào hệ thống với tài khoản candidate 2. Người dùng vào trang candidate 3. Vào tab cv manager 4. Xóa cv 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ xóa cv đó khỏi cơ sở dữ liệu
Exception Condition		

- Nộp CV

Use case name	Nộp cv	
Scenario	Candidate nộp cv	
Triggering Event	Khi candidate có nhu cầu nộp cv vào một bài đăng	
Brief Description	Người dùng truy cập trang search và nộp cv vào 1 bài đăng	
Actor	Candidate	
Precondition	Người dùng phải đăng nhập với loại tài khoản Candidate	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người dùng đăng nhập vào hệ thống với tài khoản candidate 2. Người dùng vào trang search 3. Tìm 1 bài đăng 4. Nộp cv 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ ghi nhận cv nộp vào bài đăng và gửi mail thông báo cho nhà tuyển dụng của bài đăng đó
Exception Condition		

- Rút CV

Use case name	Hủy nộp cv	
Scenario	Candidate hủy nộp cv	
Triggering Event	Khi candidate có nhu cầu hủy nộp cv vào một bài đăng	
Brief Description	Người dùng truy cập trang candidate và hủy nộp cv vào 1 bài đăng	
Actor	Candidate	
Precondition	Người dùng phải đăng nhập với loại tài khoản Candidate	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người dùng đăng nhập vào hệ thống với tài khoản candidate 2. Người dùng vào trang candidate 3. Vào tab applied post 4. Hủy nộp cv 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ xóa ghi nhận về cv đã nộp vào vài viết
Exception Condition		

- Xem lịch sử nộp CV

Use case name	Xem lịch sử nộp cv	
Scenario	Candidate xem lịch sử cv	
Triggering Event	Khi candidate có nhu cầu xem lịch sử nộp cv	
Brief Description	Người dùng truy cập trang candidate và xem lịch sử nộp cv	
Actor	Candidate	
Precondition	Người dùng phải đăng nhập với loại tài khoản Candidate	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người dùng đăng nhập vào hệ thống với tài khoản candidate 2. Người dùng vào trang candidate 3. Vào tab applied post 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ trả về lịch sử nộp cv của user đó
Exception Condition		

- Đánh dấu

Use case name	Đánh dấu bài đăng	
Scenario	Candidate đánh dấu bài đăng	
Triggering Event	Khi candidate có nhu cầu đánh dấu bài đăng	
Brief Description	Người dùng truy cập trang search và đánh dấu bài đăng	
Actor	Candidate	
Precondition	Người dùng phải đăng nhập với loại tài khoản Candidate	
Flow of Event	Actor	System
	1. Người dùng đăng nhập vào hệ thống với tài khoản candidate 2. Người dùng vào trang search 3. Tìm bài đăng 4. Đánh dấu bài đăng	2.1 Hệ thống sẽ ghi nhận và lưu thông tin vào cơ sở dữ liệu
Exception Condition		

- Hủy đánh dấu

Use case name	Hủy đánh dấu bài đăng	
Scenario	Candidate hủy đánh dấu bài đăng	
Triggering Event	Khi candidate có nhu cầu hủy đánh dấu bài đăng	
Brief Description	Người dùng truy cập trang candidate và hủy đánh dấu bài đăng	
Actor	Candidate	
Precondition	Người dùng phải đăng nhập với loại tài khoản Candidate	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người dùng đăng nhập vào hệ thống với tài khoản candidate 2. Người dùng vào trang candidate 3. Vào tab bookmark manager 4. Hủy đánh dấu bài đăng 	2.1 Hệ thống sẽ ghi nhận và xóa thông tin vào cơ sở dữ liệu
Exception Condition		

- Cập nhật thông tin tài khoản

Use case name	Cập nhật thông tin	
Scenario	Người dùng cập nhật thông tin	
Triggering Event	Khi người dùng có nhu cầu cập nhật thông tin	
Brief Description	Người dùng truy cập trang candidate hoặc recruiter và cập nhật thông tin	
Actor	Candidate, Recruiter	
Precondition	Người dùng phải đăng nhập với loại tài khoản Candidate hoặc Recruiter	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người dùng đăng nhập vào hệ thống với tài khoản candidate hoặc recruiter 2. Người dùng vào trang candidate hoặc recruiter 3. Vào tab profile 4. Cập nhật thông tin 	<p>2.1 Hệ thống sẽ ghi nhận và lưu thông tin mới vào cơ sở dữ liệu</p>
Exception Condition		

- Đổi mật khẩu

Use case name	Thay đổi mật khẩu	
Scenario	Người dùng Thay đổi mật khẩu	
Triggering Event	Khi người dùng có nhu cầu thay đổi mật khẩu	
Brief Description	Người dùng truy cập trang candidate hoặc recruiter và thay đổi mật khẩu	
Actor	Candidate, Recruiter	
Precondition	Người dùng phải đăng nhập với loại tài khoản Candidate hoặc Recruiter	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người dùng đăng nhập vào hệ thống với tài khoản candidate hoặc recruiter 2. Người dùng vào trang candidate hoặc recruiter 3. Vào tab change password 4. Thay đổi mật khẩu 	<p>2.1 Hệ thống sẽ ghi nhận và lưu thông tin mới vào cơ sở dữ liệu</p>
Exception Condition		

- Xem danh sách người dùng

Use case name	Xem tất cả danh sách người dùng	
Scenario	Để người quản lý có thể quản lý người dùng	
Triggering Event	Khi người quản trị có thể xem hệ thống có những người dùng nào	
Brief Description	Người quản trị đăng nhập vào giao diện admin (Quản trị) sẽ có thể xem toàn bộ người dùng của hệ thống	
Actor	Admin	
Precondition	Phải là người dùng có quyền admin	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người quản trị đăng nhập vào hệ thống 2. Chọn vào danh sách người dùng 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ lấy ra tất cả danh sách người dùng hợp lệ có trong hệ thống
Exception Condition		

- Xem tất cả bài đăng

Use case name	Xem tất cả bài đăng tuyển dụng trong hệ thống	
Scenario	Để người quản trị có thể quản lý bài đăng tuyển dụng	
Triggering Event	Khi người quản trị có thể xem hệ thống có tất cả bài đăng của những nhà tuyển dụng nào	
Brief Description	Người quản trị đăng nhập vào giao diện admin (Quản trị) sẽ có thể xem toàn bộ danh sách bài đăng của tất cả nhà tuyển dụng	
Actor	Admin	
Precondition	Phải là người dùng có quyền admin	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người quản trị đăng nhập vào hệ thống 2. Chọn vào danh sách bài đăng 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ lấy ra tất cả danh sách bài đăng tuyển dụng hợp lệ có trong hệ thống
Exception Condition		

- Cập nhật loại tài khoản

Use case name	Cập nhật lại thông tin của các loại tài khoản	
Scenario	Để người quản trị có thể thay đổi thông tin của loại tài khoản đăng ký	
Triggering Event	Khi người quản trị có thể thay đổi thông tin của loại tài khoản	
Brief Description	Người quản trị đăng nhập vào giao diện admin (Quản trị) sẽ có thể cập nhật lại tên cho loại tài khoản	
Actor	Admin	
Precondition	Phải là người dùng có quyền admin	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người quản trị đăng nhập vào hệ thống 2. Chọn loại tài khoản cần cập nhật 3. Cập nhật thông tin 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ lưu lại thay đổi của loại tài khoản
Exception Condition		

- Thêm loại nghề nghiệp

Use case name	Thêm mới loại nghề nghiệp		
Scenario	Để người quản trị có thể thêm mới bất kì loại nghề nghiệp mà người quản trị muốn		
Triggering Event	Khi người quản trị có thể thực hiện thao tác thêm mới bất kì loại nghề nghiệp nào		
Brief Description	Người quản trị đăng nhập vào giao diện admin (Quản trị) , chọn thêm mới loại nghề nghiệp , và điền các thông tin cần thiết		
Actor	Admin		
Precondition	Phải là người dùng có quyền admin		
Flow of Event	Actor	System	
	<ol style="list-style-type: none"> 1. Người quản trị đăng nhập vào hệ thống 2. Chọn loại thêm mới loại nghề nghiệp 3. Điền đầy đủ các thông tin của loại nghề nghiệp 4. Lưu 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ lưu lại giá trị của loại nghề nghiệp đó và trên trang web sẽ hiện thị loại nghề nghiệp đó 	
Exception Condition			

- Cập nhật loại nghề nghiệp

Use case name	Cập nhật loại nghề nghiệp	
Scenario	Để người quản trị có thể cập nhật lại thông tin của bất kì loại nghề nghiệp mà người quản trị muốn	
Triggering Event	Khi người quản trị có thể thực hiện thao tác cập nhật thông tin của bất kì loại nghề nghiệp nào	
Brief Description	Người quản trị đăng nhập vào giao diện admin (Quản trị), chọn loại nghề nghiệp, tiến hành cập nhật và điền các thông tin cần cập nhật	
Actor	Admin	
Precondition	Phải là người dùng có quyền admin	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người quản trị đăng nhập vào hệ thống 2. Chọn loại nghề nghiệp cần cập nhật 3. Chính sửa thông tin loại nghề nghiệp theo ý muốn 4. Lưu 	<p>2.1 Hệ thống sẽ lưu lại giá trị đã thay đổi của loại nghề nghiệp đó và trên trang web sẽ hiển thị loại nghề nghiệp đó</p>
Exception Condition		

- Xóa loại nghề nghiệp

Use case name	Xóa loại nghề nghiệp		
Scenario	Người quản trị có thể xóa bất kì loại nghề nghiệp mà người quản trị muốn		
Triggering Event	Khi người quản trị có thể thực hiện thao tác xóa bất kì loại nghề nghiệp nào		
Brief Description	Người quản trị đăng nhập vào giao diện admin (Quản trị) , chọn loại nghề nghiệp, tiến hành cập nhật và điều các thông tin cần cập nhật		
Actor	Admin		
Precondition	Phải là người dùng có quyền admin		
Flow of Event	Actor	System	
	<ol style="list-style-type: none"> 1. Người quản trị đăng nhập vào hệ thống 2. Chọn loại nghề nghiệp cần cập nhật 3. Xóa loại nghề nghiệp theo ý muốn 	2.1 Hệ thống sẽ xóa loại nghề nghiệp đó	
Exception Condition			

- Thêm chuyên ngành

Use case name	Thêm mới chuyên ngành	
Scenario	Để người quản trị có thể thêm mới bất kì chuyên ngành mà người quản trị muốn	
Triggering Event	Khi người quản trị có thể thực hiện thao tác thêm mới bất kì chuyên ngành nào	
Brief Description	Người quản trị đăng nhập vào giao diện admin (Quản trị) , chọn thêm mới chuyên ngành, và điền các thông tin cần thiết	
Actor	Admin	
Precondition	Phải là người dùng có quyền admin	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người quản trị đăng nhập vào hệ thống 2. Chọn loại thêm mới chuyên ngành 3. Điền đầy đủ các thông tin của chuyên ngành 4. Lưu 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ lưu lại giá trị của chuyên ngành đó
Exception Condition		

- Cập nhật loại nghề nghiệp

Use case name	Cập nhật chuyên ngành	
Scenario	Để người quản trị có thể cập nhật lại thông tin của bất kì chuyên ngành mà người quản trị muốn	
Triggering Event	Khi người quản trị có thể thực hiện thao tác cập nhật thông tin của bất kì chuyên ngành nào	
Brief Description	Người quản trị đăng nhập vào giao diện admin (Quản trị) , chọn chuyên ngành, tiến hành cập nhật và điền các thông tin cần cập nhật	
Actor	Admin	
Precondition	Phải là người dùng có quyền admin	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người quản trị đăng nhập vào hệ thống 2. Chọn chuyên ngành cần cập nhật 3. Chính sửa thông tin chuyên ngành theo ý muốn 4. Lưu 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ lưu lại giá trị đã thay đổi của chuyên ngành
Exception Condition		

- Xóa loại nghề nghiệp

Use case name	Xóa chuyên ngành	
Scenario	Người quản trị có thể xóa bất kỳ chuyên ngành mà người quản trị muốn	
Triggering Event	Khi người quản trị có thể thực hiện thao tác xóa bất kỳ chuyên ngành nào	
Brief Description	Người quản trị đăng nhập vào giao diện admin (Quản trị) , chọn chuyên ngành, tiến hành cập nhật và điền các thông tin cần cập nhật	
Actor	Admin	
Precondition	Phải là người dùng có quyền admin	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Người quản trị đăng nhập vào hệ thống 2. Chọn chuyên ngành cần cập nhật 3. Xóa chuyên ngành theo ý muốn 	2.1 Hệ thống sẽ xóa chuyên ngành đó
Exception Condition		

- Thêm bài đăng

Use case name	Thêm mới bài đăng	
Scenario	Để nhà tuyển dụng có thể thêm mới bài đăng mà nhà tuyển dụng muốn	
Triggering Event	Khi nhà tuyển dụng thực hiện thao tác thêm mới bài đăng	
Brief Description	Nhà tuyển dụng đăng nhập vào giao diện Recruiter, chọn thêm bài đăng mới, và điền các thông tin cần thiết	
Actor	Recruiter	
Precondition	Phải là người dùng có quyền recruiter	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Nhà tuyển dụng đăng nhập vào hệ thống 2. Chọn thêm bài đăng mới 3. Điền đầy đủ các thông tin của bài đăng 4. Đăng bài 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ lưu bài đăng đó và hiện trên trang tìm kiếm
Exception Condition		

- Cập nhật bài đăng

Use case name	Cập nhật bài đăng	
Scenario	Để nhà tuyển dụng có thể cập nhật bài đăng mà nhà tuyển dụng muốn	
Triggering Event	Khi nhà tuyển dụng thực hiện thao tác cập nhật bài đăng	
Brief Description	Nhà tuyển dụng đăng nhập vào giao diện Recruiter, chọn cập nhật bài đăng, và cập nhật các thông tin cần thiết	
Actor	Recruiter	
Precondition	Phải là người dùng có quyền recruiter	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Nhà tuyển dụng đăng nhập vào hệ thống 2. Chọn bài đăng muốn cập nhật 3. Cập nhật thông tin của bài đăng 4. Lưu 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ lưu lại thay đổi của bài đăng đó và hiện trên trang tìm kiếm
Exception Condition		

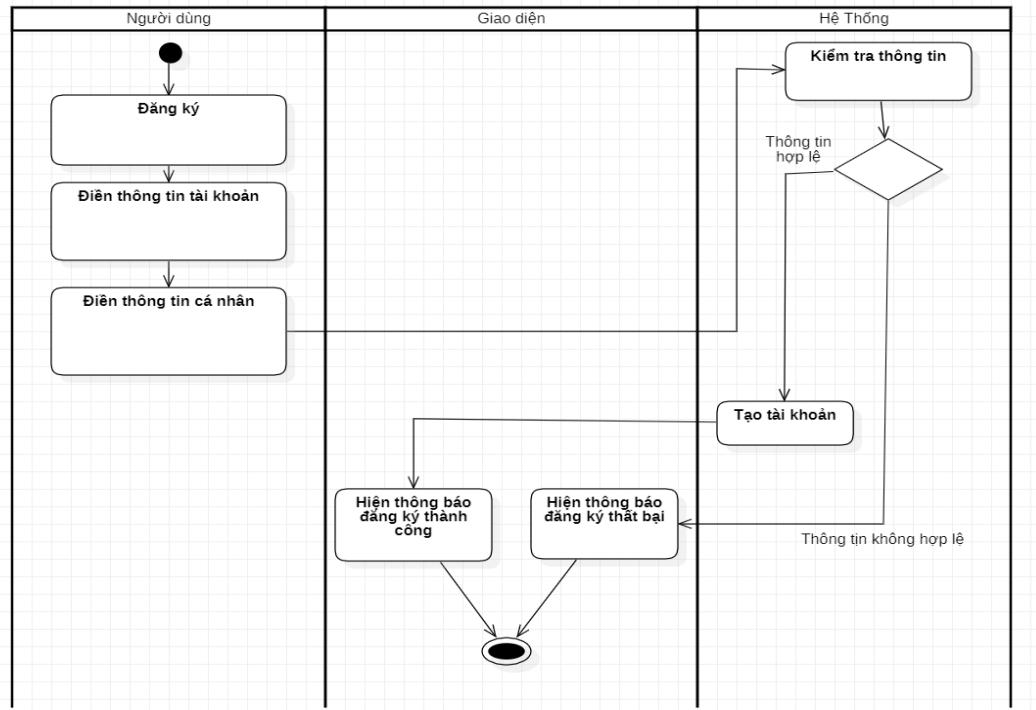
- Xóa bài đăng

Use case name	Xóa bài đăng	
Scenario	Để nhà tuyển dụng có thể xóa bài đăng mà nhà tuyển dụng muốn	
Triggering Event	Khi nhà tuyển dụng thực hiện thao tác xóa bài đăng	
Brief Description	Nhà tuyển dụng đăng nhập vào giao diện Recruiter, và xóa bài đăng.	
Actor	Recruiter	
Precondition	Phải là người dùng có quyền recruiter	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Nhà tuyển dụng đăng nhập vào hệ thống 2. Chọn bài đăng muốn xóa 3. Xóa bài đăng 	<ol style="list-style-type: none"> 2.1 Hệ thống sẽ xóa bài đăng đó và ẩn trên trang tìm kiếm
Exception Condition		

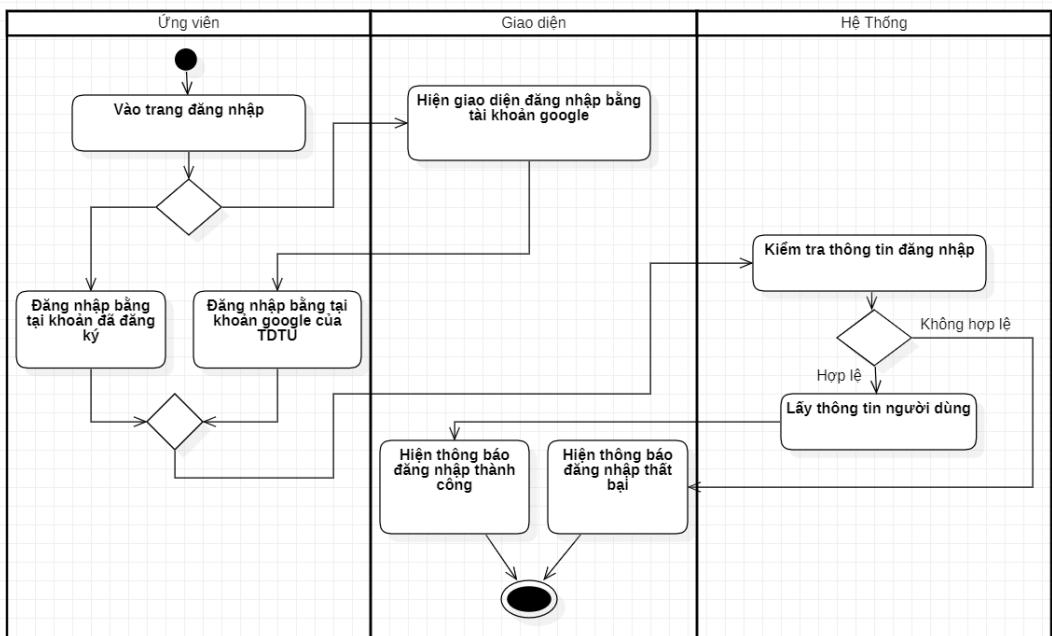
- Xem danh sách ứng tuyển

Use case name	Xem ứng viên đã ứng tuyển	
Scenario	Để nhà tuyển dụng có thể xem ứng viên đã ứng tuyển và bài đăng của nhà tuyển dụng	
Triggering Event	Khi nhà tuyển dụng vào trang xem các ứng viên	
Brief Description	Nhà tuyển dụng đăng nhập vào giao diện Recruiter, và xem các ứng viên	
Actor	Recruiter	
Precondition	Phải là người dùng có quyền recruiter	
Flow of Event	Actor	System
	<ol style="list-style-type: none"> 1. Nhà tuyển dụng đăng nhập vào hệ thống 2. Vào trang xem các ứng viên 	<ol style="list-style-type: none"> 2.1 Hệ thống hiện các ứng viên đã nộp cv vào các post của người dùng
Exception Condition		

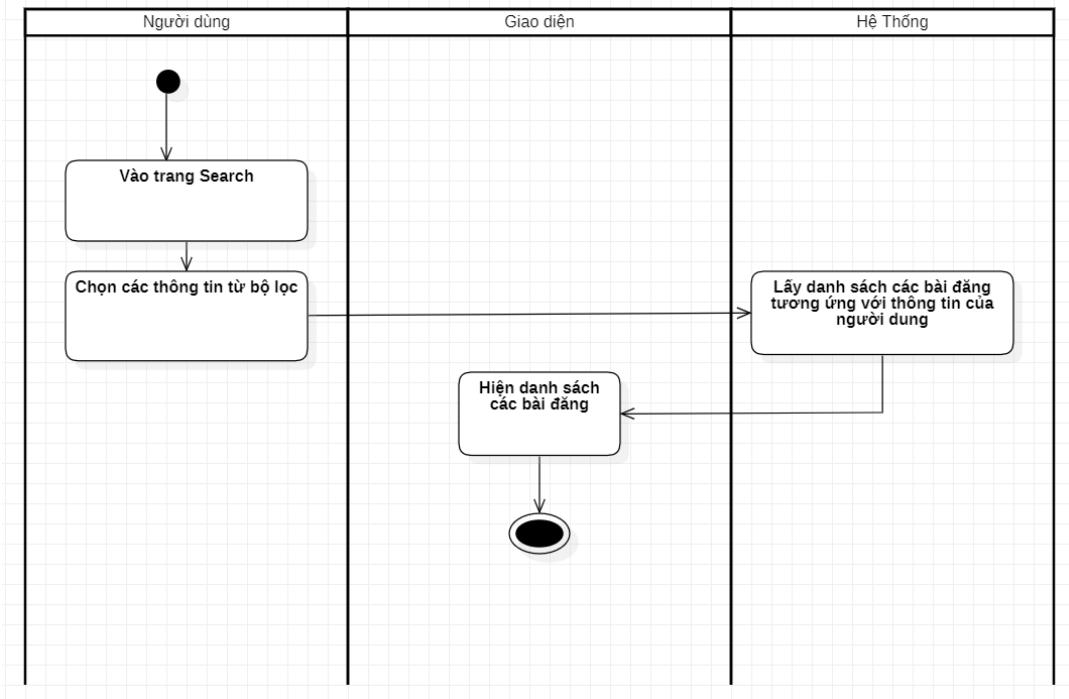
3) Activity



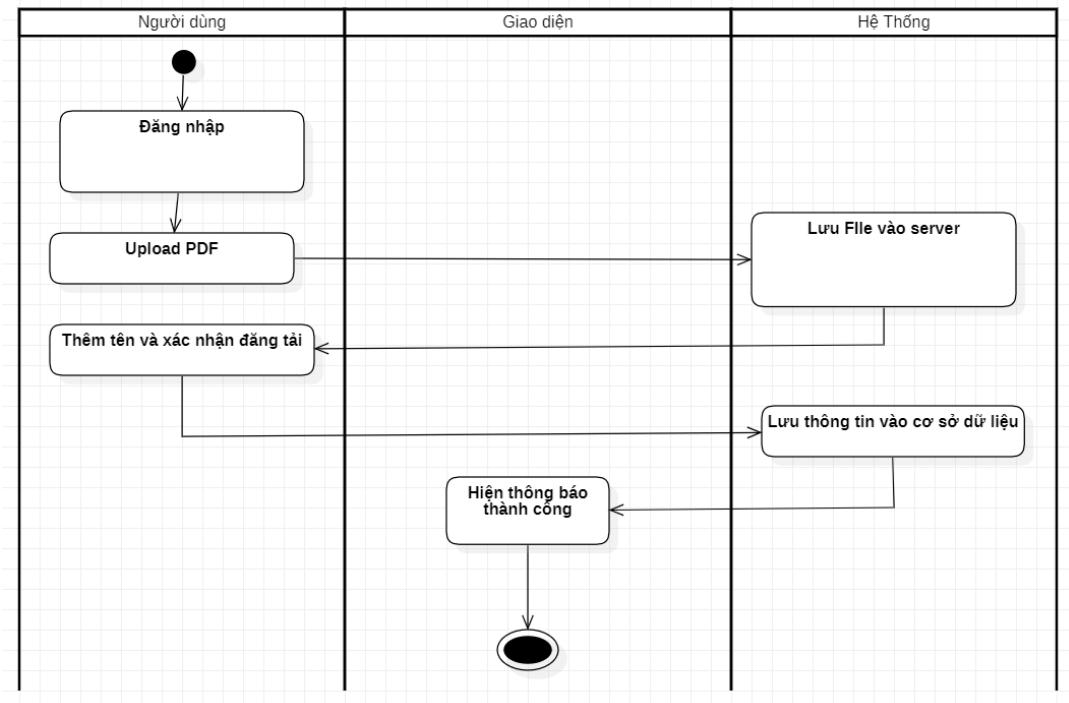
Hình 2 : Sơ đồ Activity – Đăng ký



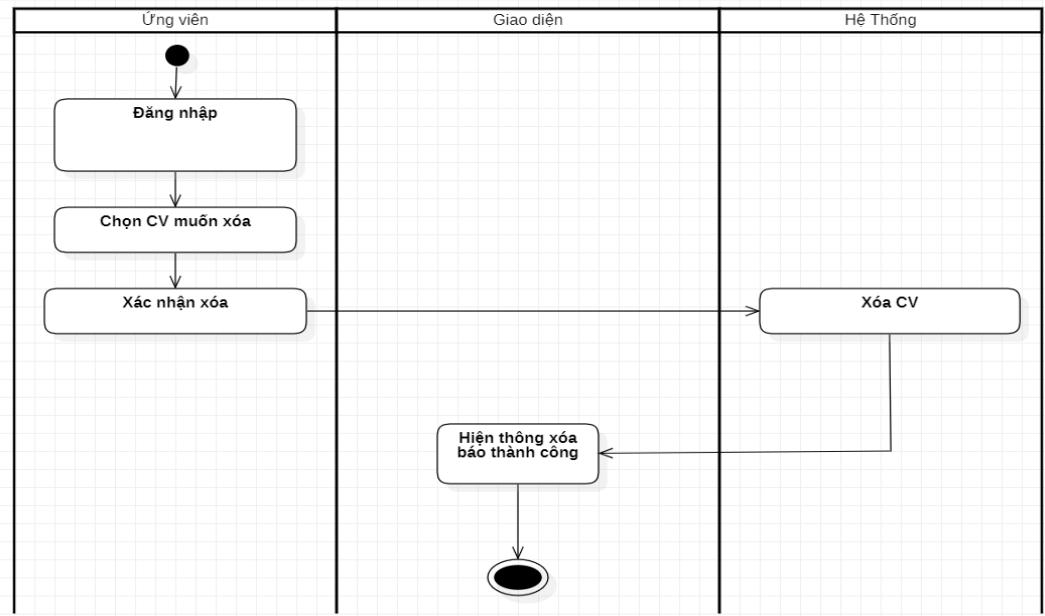
Hình 3 : Sơ đồ Activity – Đăng Nhập



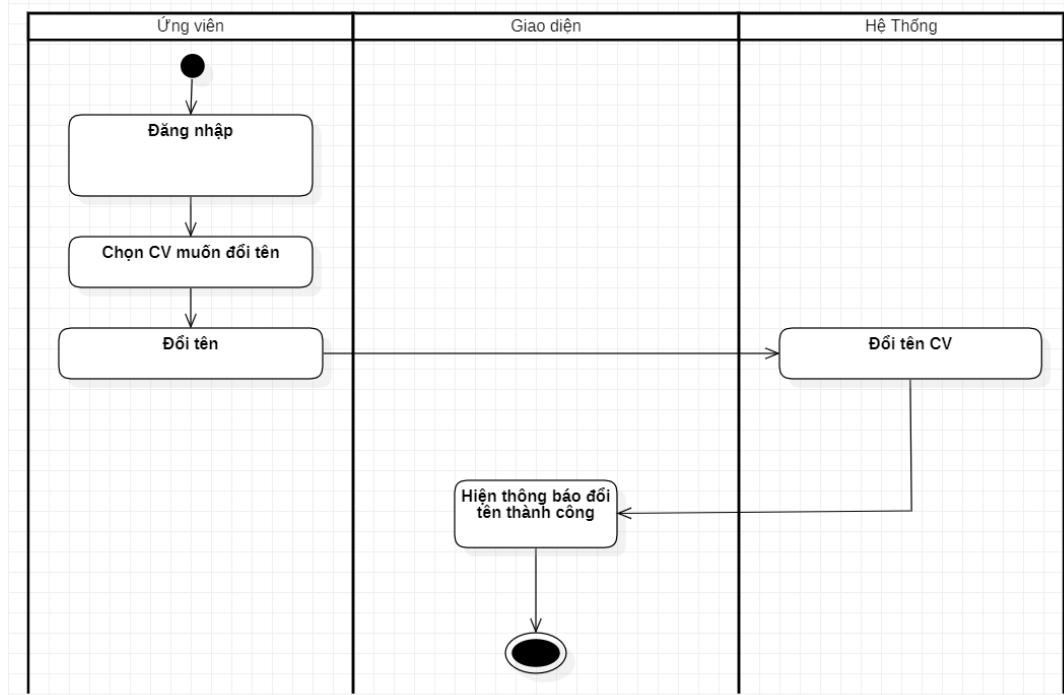
Hình 4 : Sơ đồ Activity – Tìm Kiếm



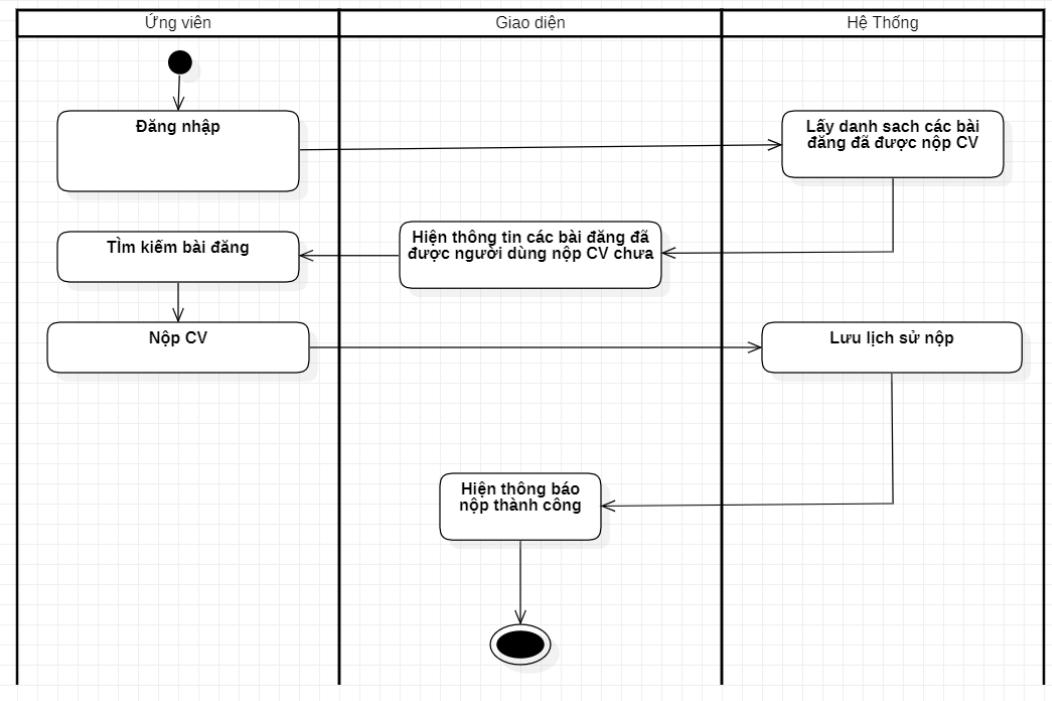
Hình 5 : Sơ đồ Activity – Đăng tải CV



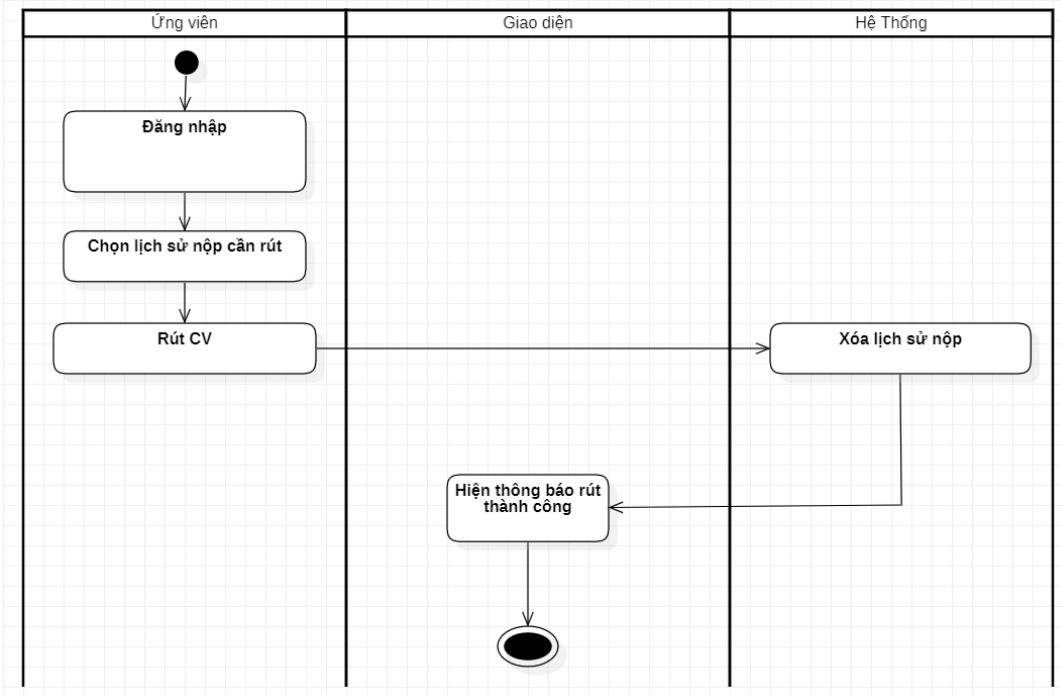
Hình 6 : Sơ đồ Activity – Xóa CV



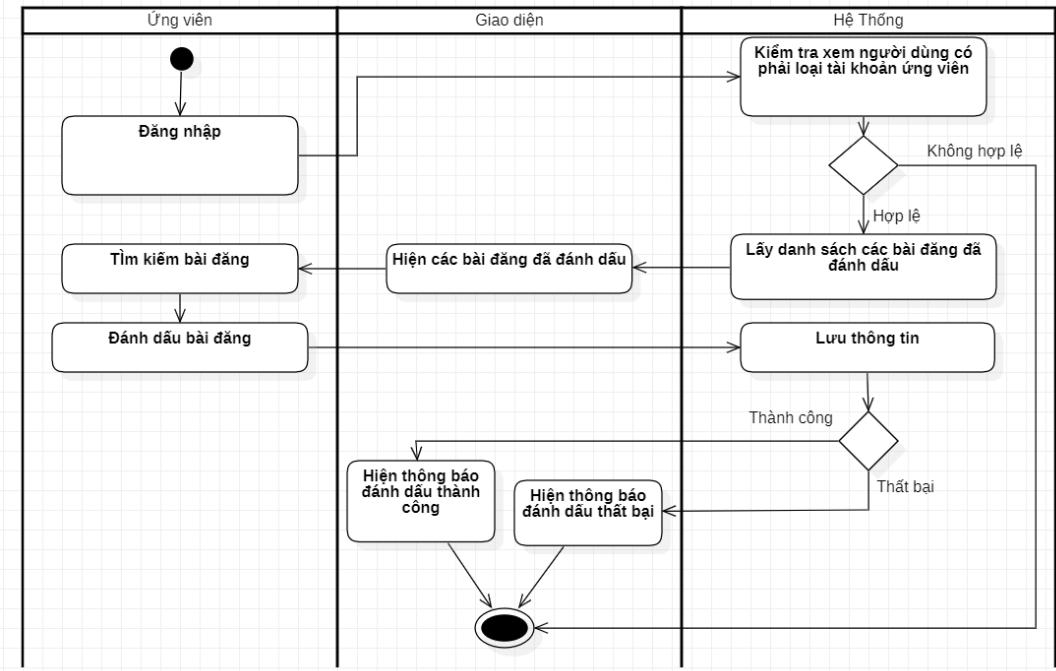
Hình 7 : Sơ đồ Activity – Đổi Tên CV



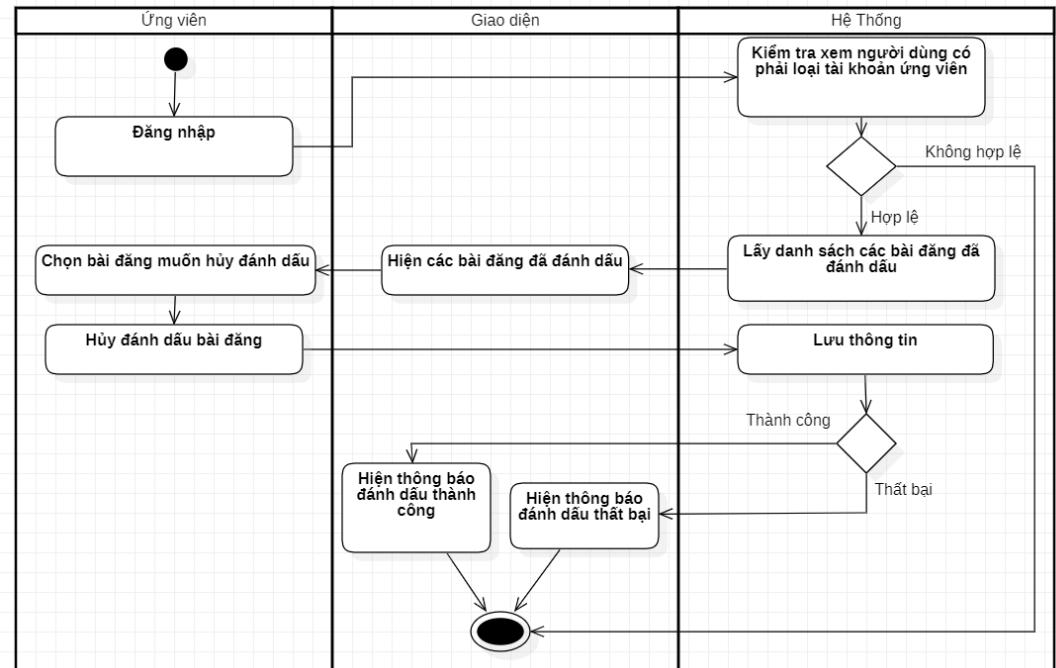
Hình 8 : Sơ đồ Activity – Nộp CV



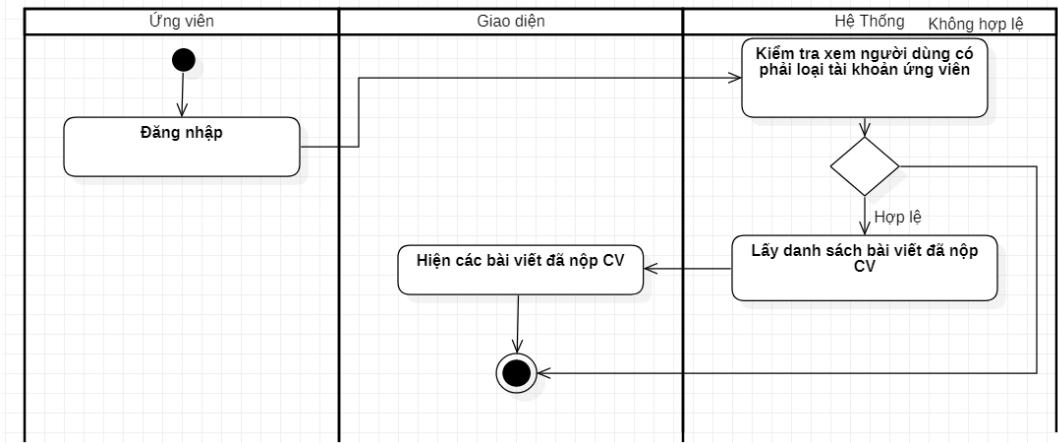
Hình 9 : Sơ đồ Activity – Rút CV



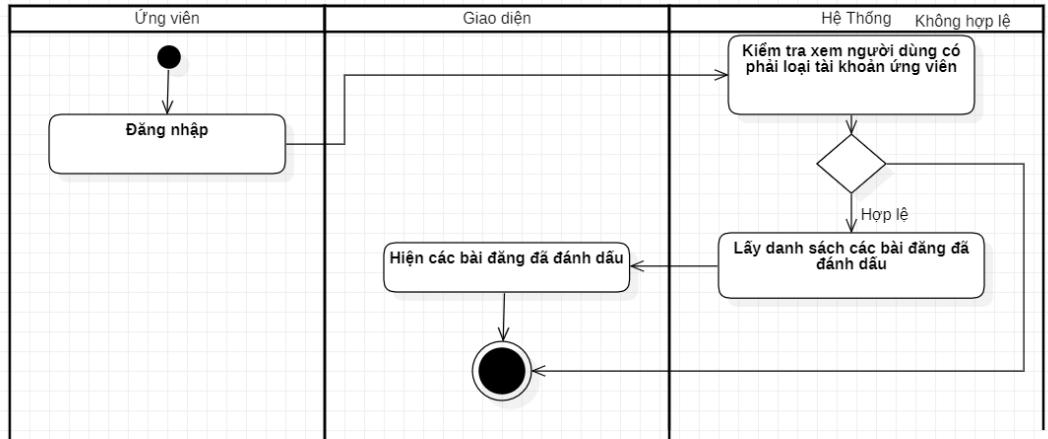
Hình 10 : Sơ đồ Activity – Đánh Dấu Bài Đăng



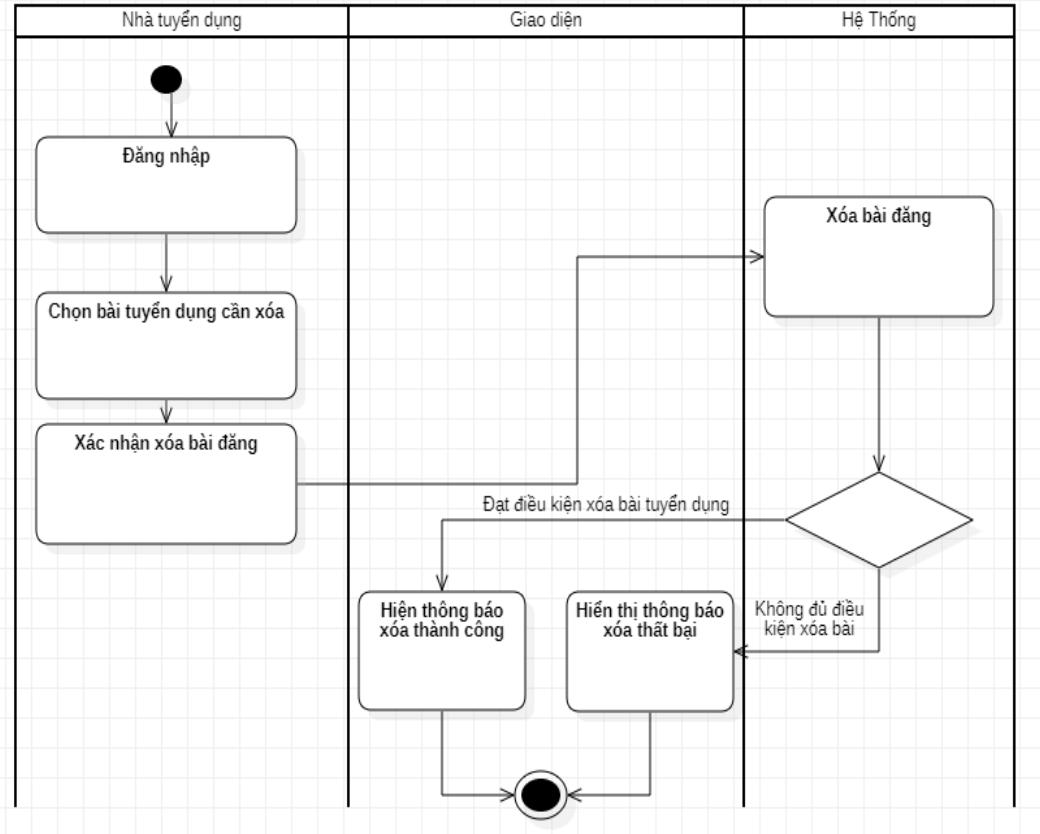
Hình 11 : Sơ đồ Activity – Hủy Đánh Dấu Bài Đăng



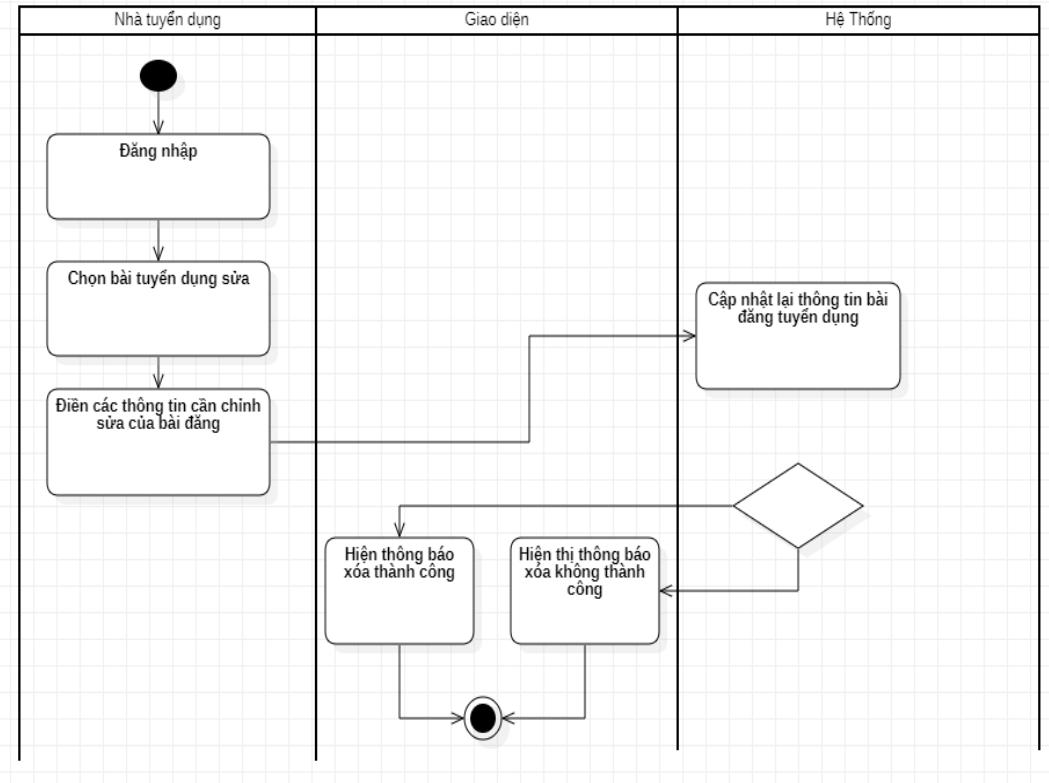
Hình 12 : Sơ đồ Activity – Xem Các Bài Viết Đã Nộp CV



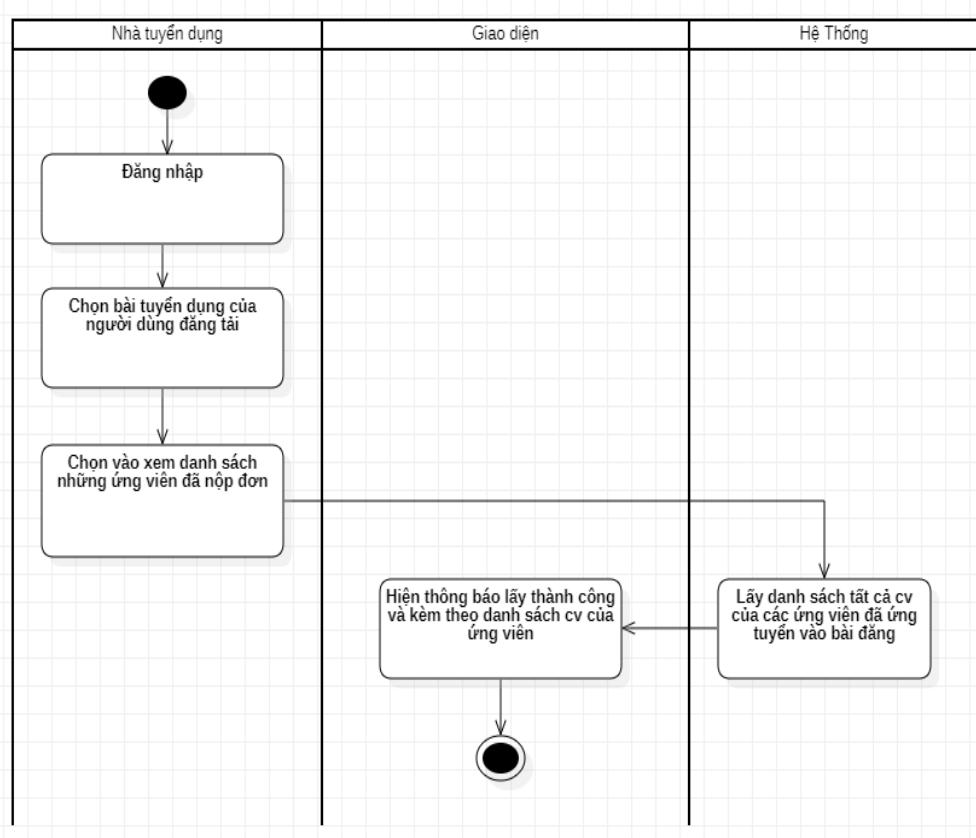
Hình 13 : Sơ đồ Activity – Xem Các Bài Viết Đã Đánh Dấu



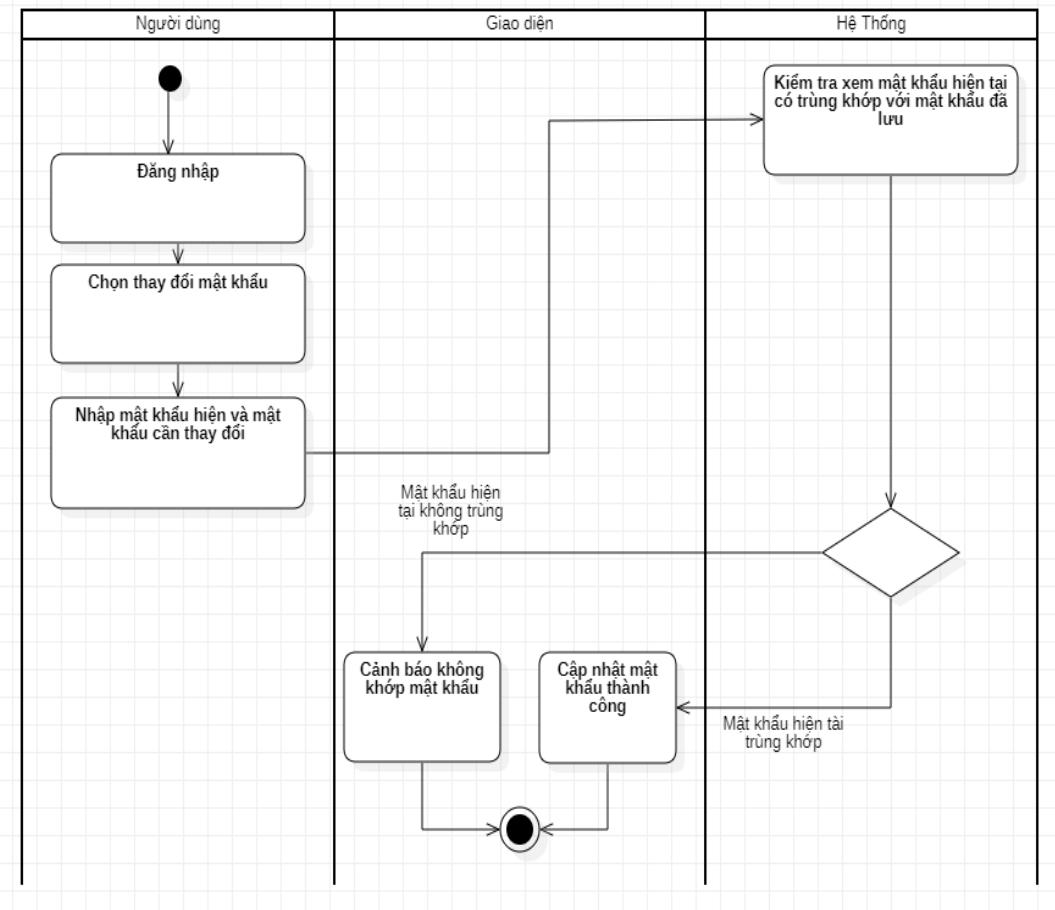
Hình 14 : Sơ đồ Activity – xóa bài đăng tuyển dụng



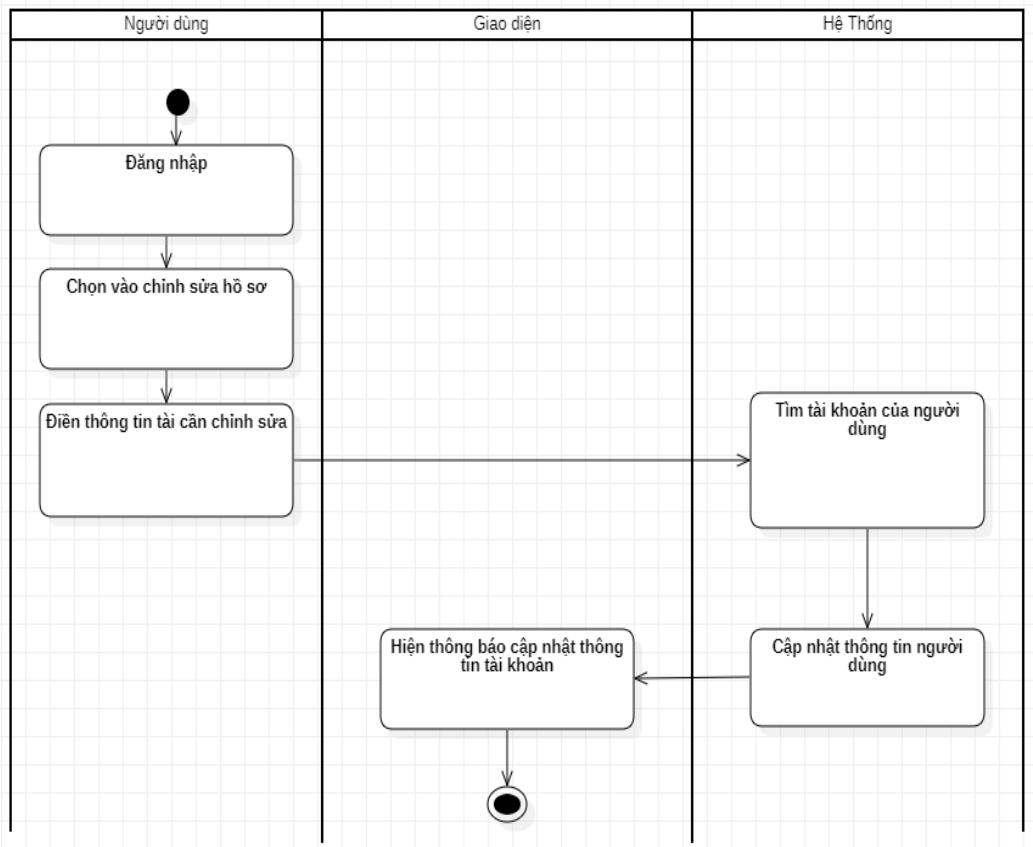
Hình 15 : Sơ đồ Activity – sửa bài đăng tuyển dụng



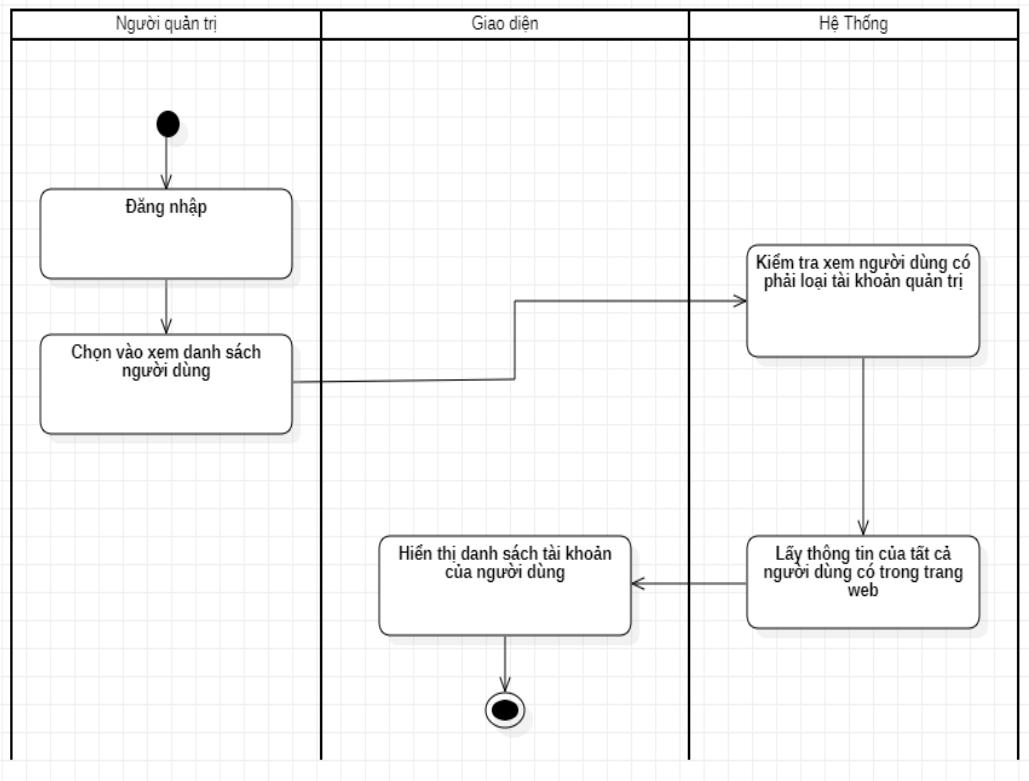
Hình 16 : Sơ đồ Activity – xem các đơn ứng tuyển của ứng viên



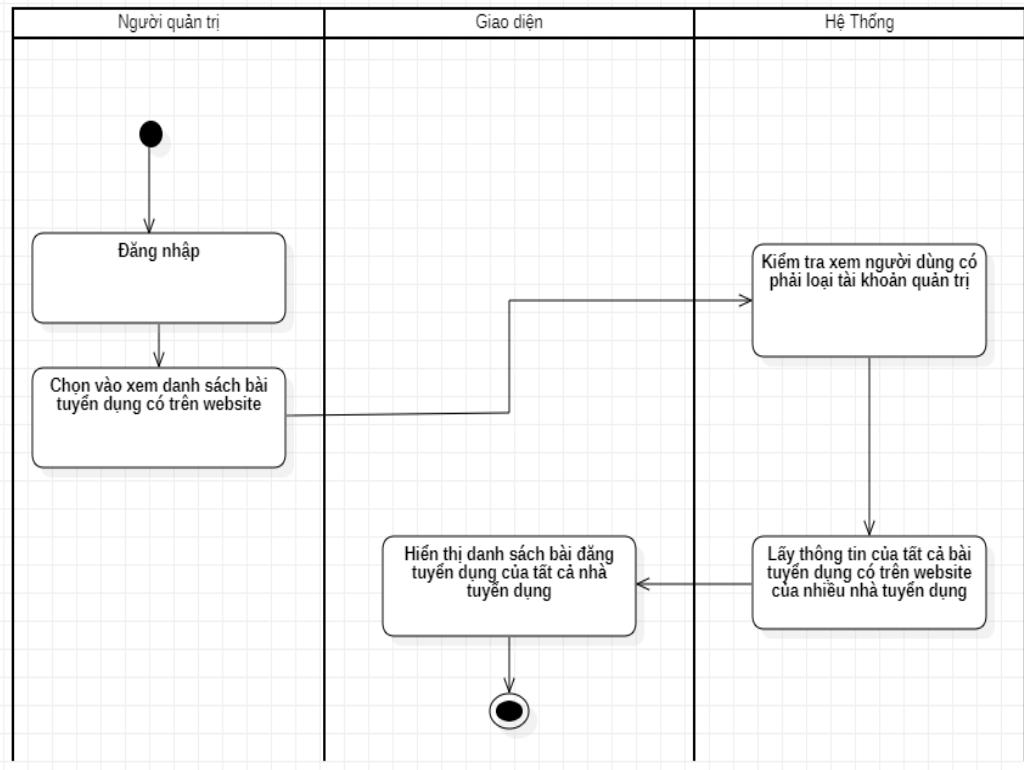
Hình 17 : Sơ đồ Activity – cập nhật lại mật khẩu mới



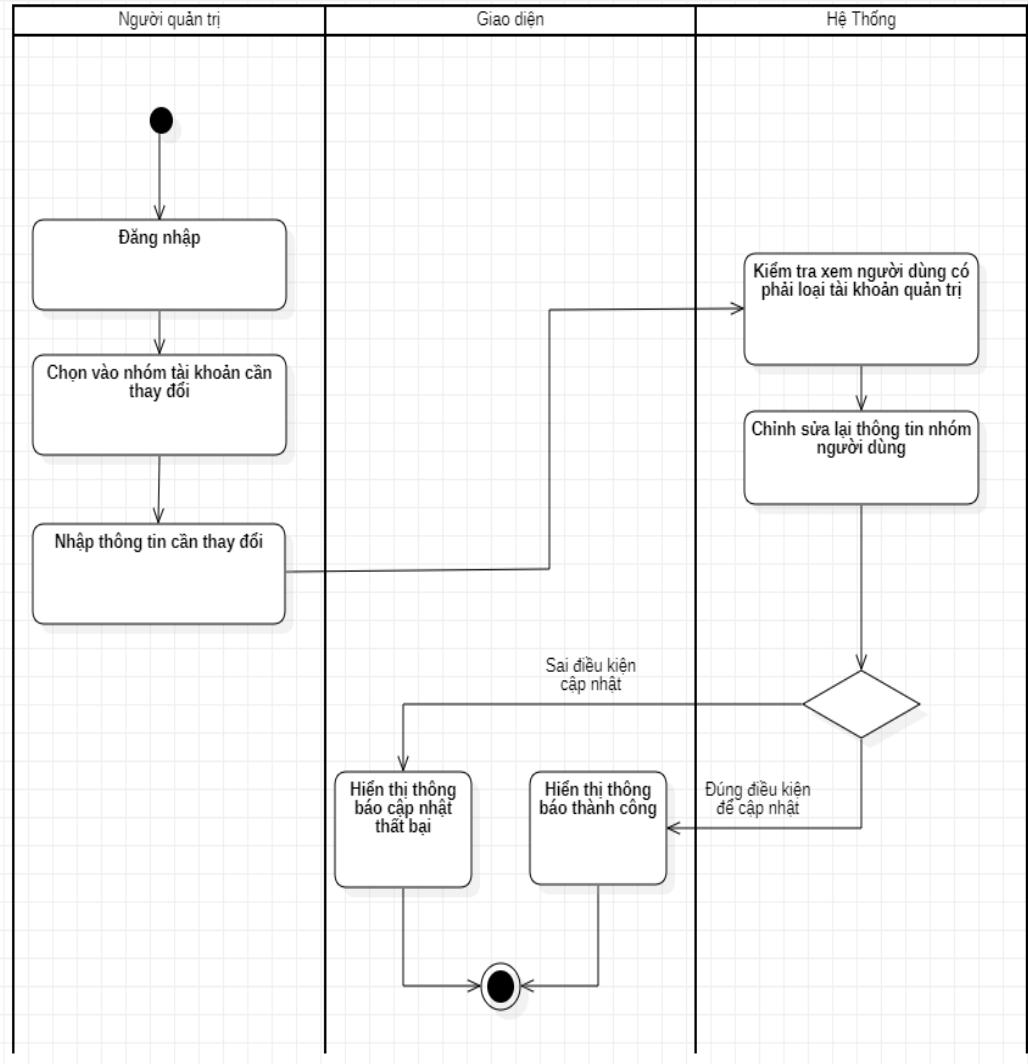
Hình 18 : Sơ đồ Activity – cập nhật lại thông tin của người dùng



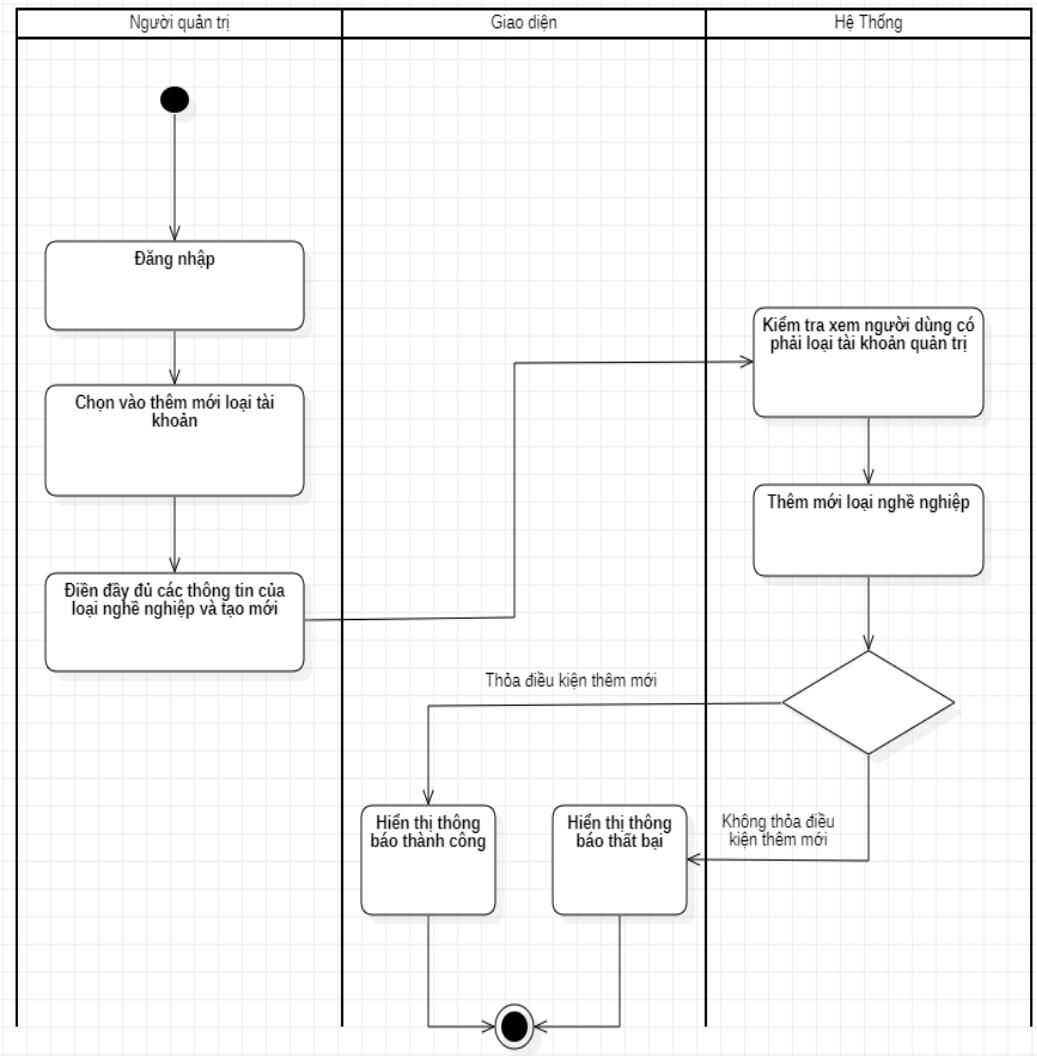
Hình 19 : Sơ đồ Activity – xem danh sách người dùng trong hệ thống



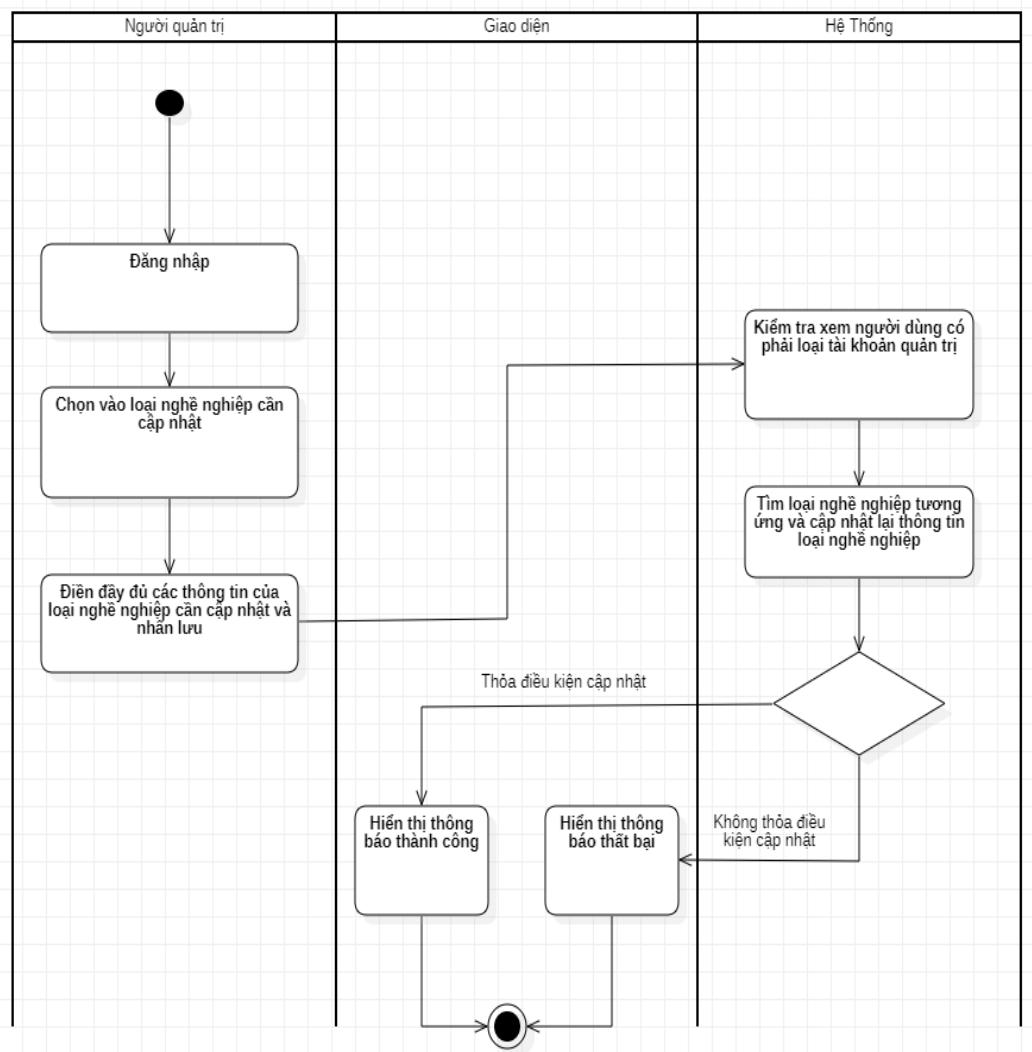
Hình 20 : Sơ đồ Activity – xem danh sách bài đăng tuyển dụng của tất cả nhà tuyển dụng



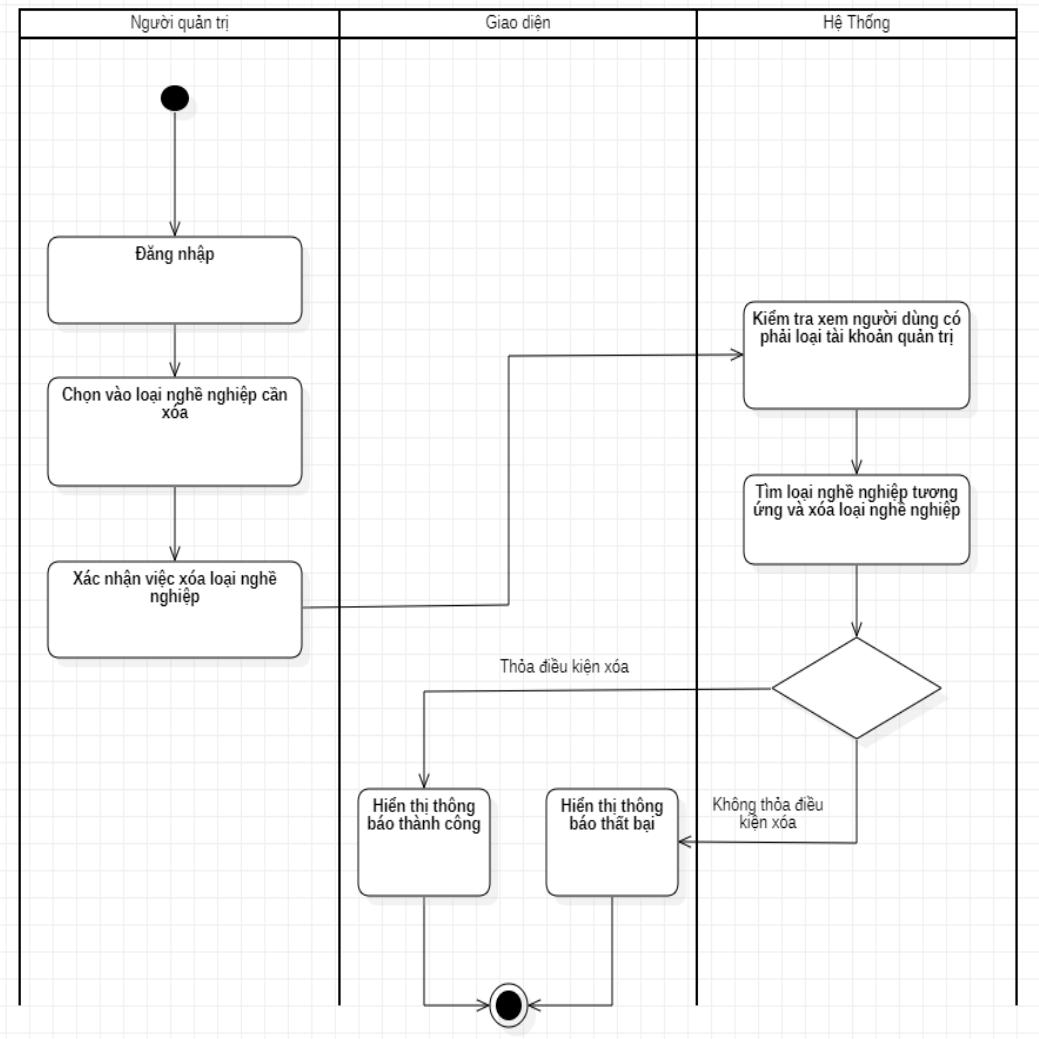
Hình 21 : Sơ đồ Activity – cập nhật thông tin nhóm tài khoản



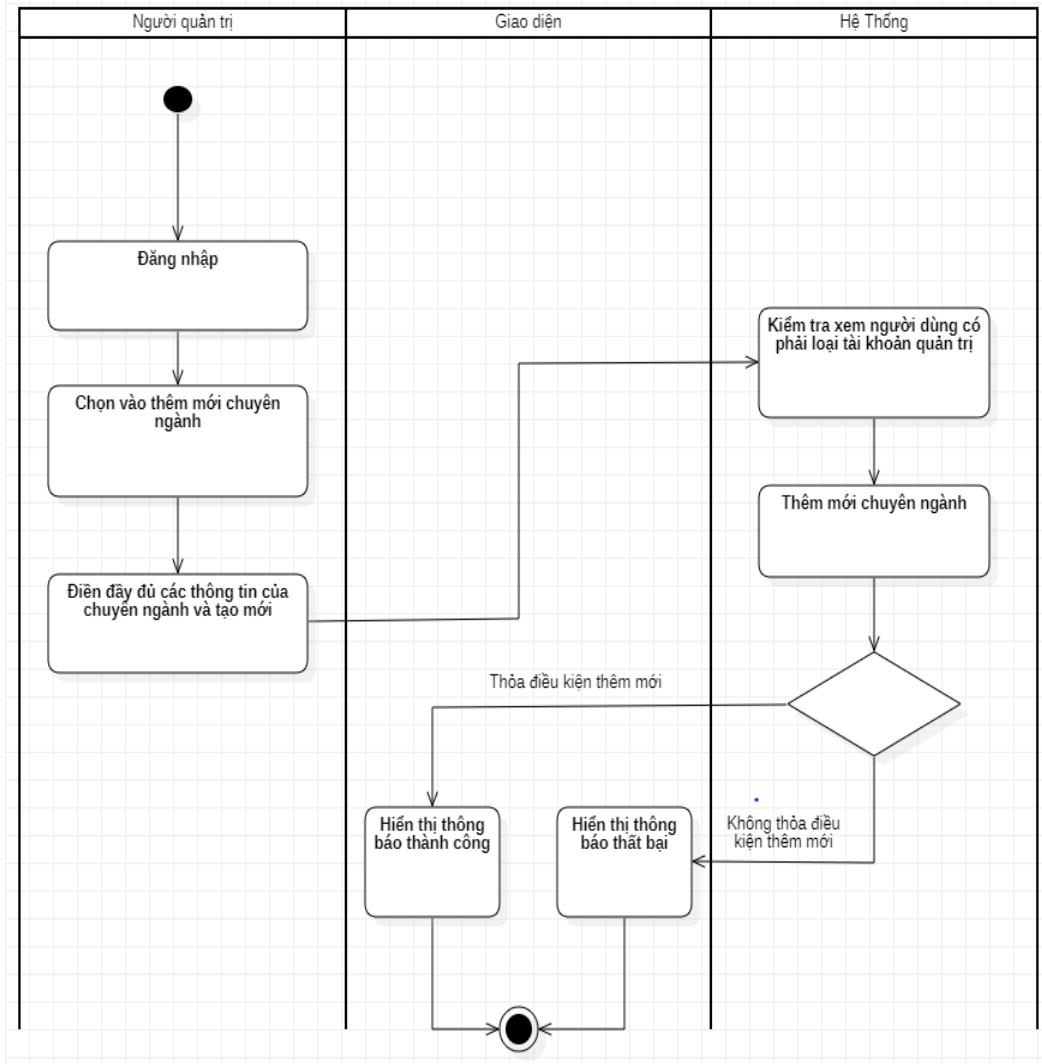
Hình 22 : Sơ đồ Activity – thêm mới loại nghề nghiệp



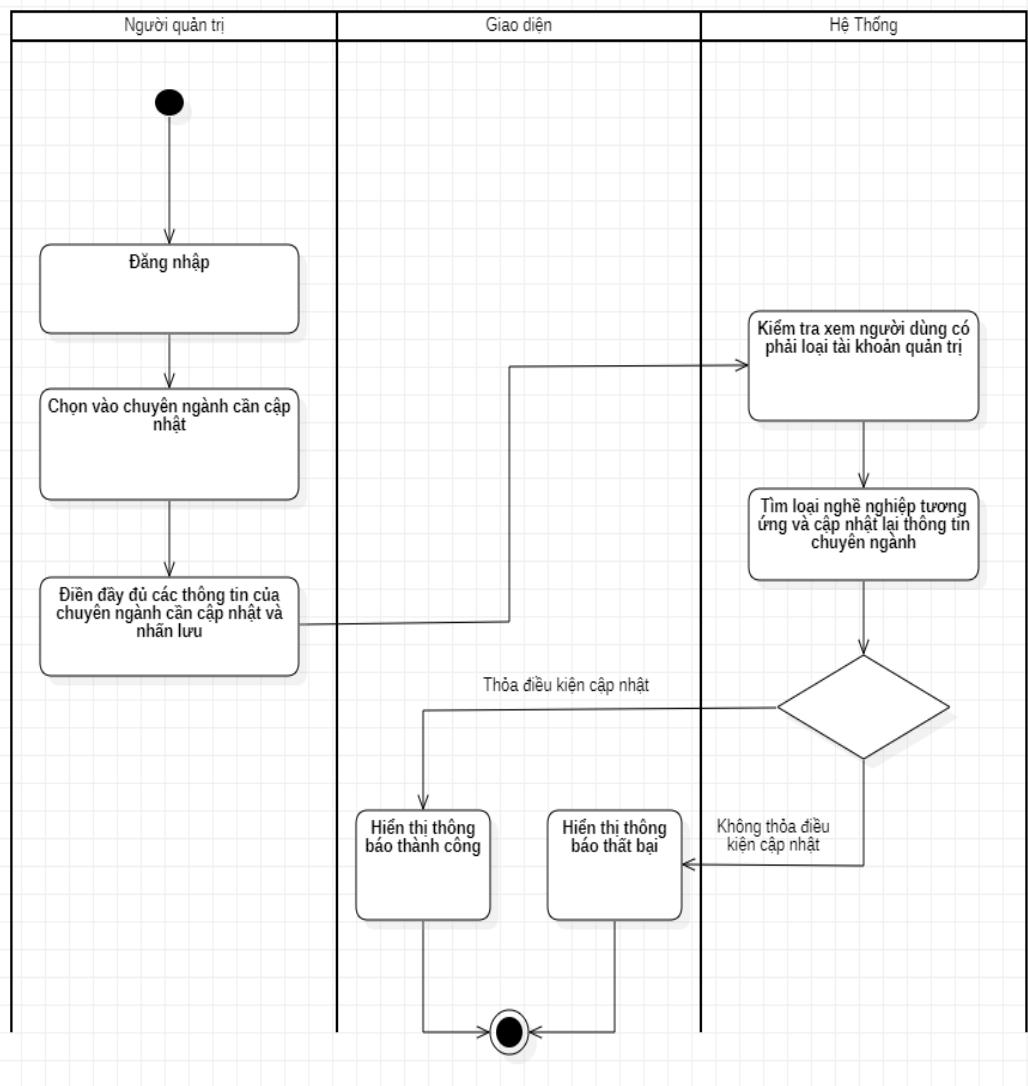
Hình 23 : Sơ đồ Activity – cập nhật lại loại nghề nghiệp



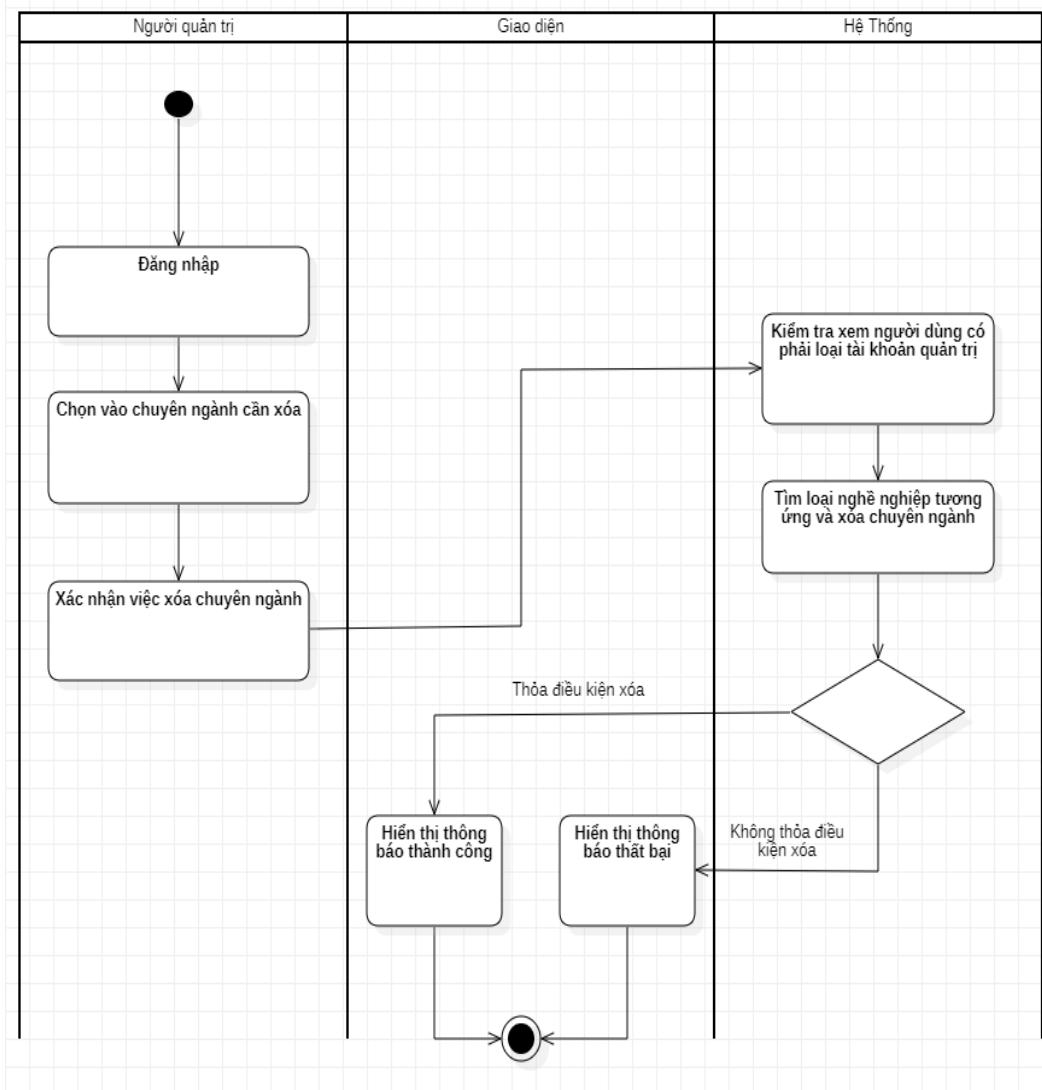
Hình 24 : Sơ đồ Activity – xóa loại nghề nghiệp



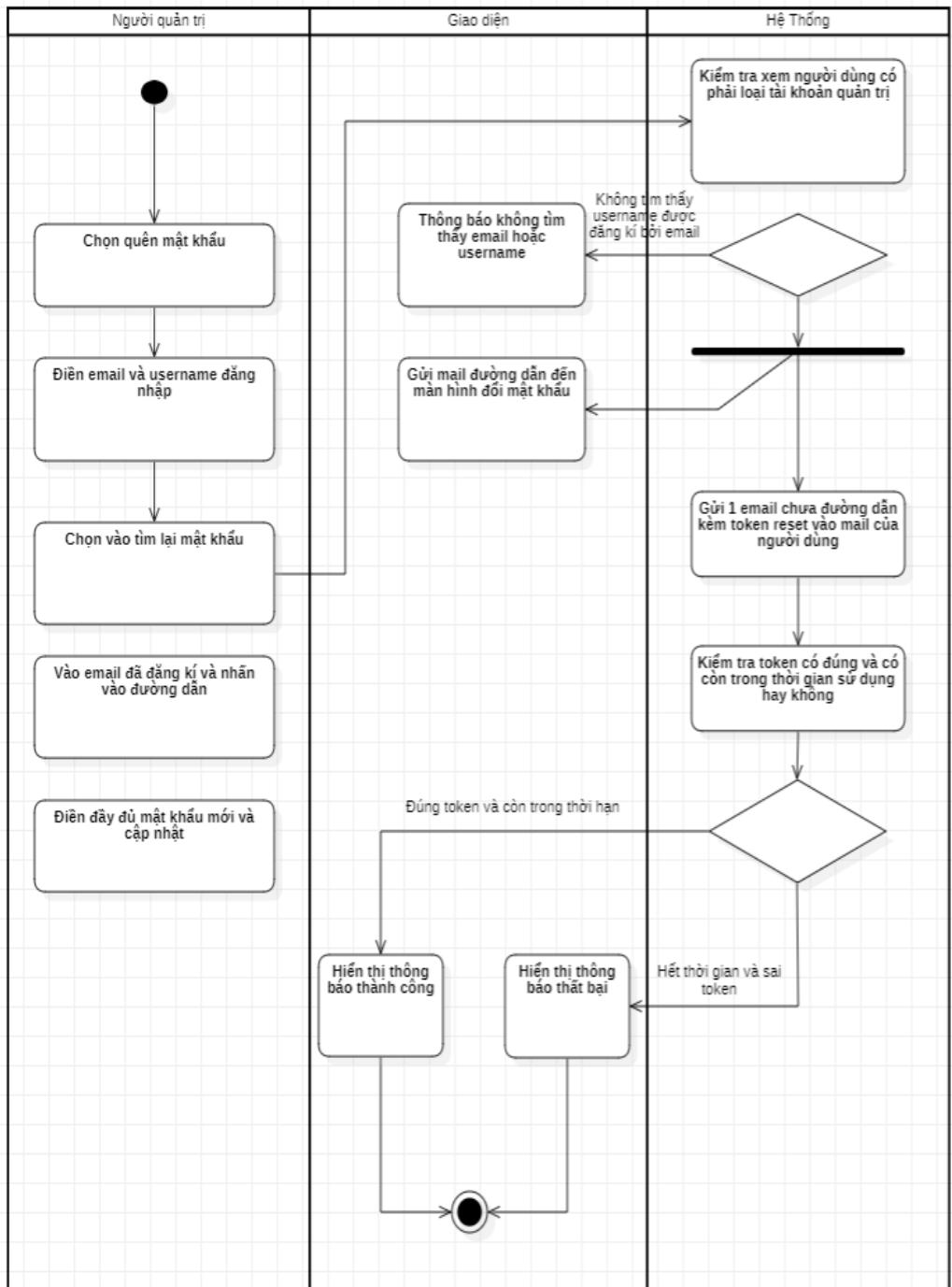
Hình 25 : Sơ đồ Activity – thêm mới chuyên ngành



Hình 26 : Sơ đồ Activity – cập nhật lại thông tin chuyên ngành



Hình 27 : Sơ đồ Activity – xóa loại nghề nghiệp



Hình 28 : Sơ đồ Activity – quên mật khẩu

2. GIỚI THIỆU VỀ CÔNG NGHỆ VÀ THUẬT NGỮ

1) Docker

Để một source code được khởi động và hoạt động trơn tru ở một máy tính đòi hỏi máy tính đó phải có đầy đủ tài nguyên như môi trường , các biến , một số thư viện , cũng như một số công cụ để hỗ trợ cho ngôn ngữ và công nghệ đó hoạt động. Với Docker bạn có thể quản lý các tài nguyên của source code một cách dễ dàng và giảm độ trễ giữ việc viết code với việc chạy chúng trong sản phẩm của bản thân.

VD: Nếu một máy tính đang chạy một phiên bản môi trường ví dụ như NodeJs ở phiên bản 16 nhưng ở một máy của người đồng nghiệp lại chạy NodeJs ở phiên bản cao hơn hoặc thấp hơn điều này sẽ dẫn đến việc là khi source code chạy ở máy mình hoạt động bình thường nhưng khi sang máy của bạn hoặc đồng nghiệp lại không được vì không hỗ trợ cách viết JS đó hoặc thiếu gì đó và dẫn đến mất thời gian rất nhiều

Chính vì thế thì Docker(Container) được sinh ra để giải quyết những vấn đề này

- Docker là một công cụ để cung cấp cho việc Build , Deploy, và Run source code ứng dụng hay là một trang web nào đó một cách dễ dàng. Và docker ban đầu được viết bằng ngôn ngữ Python và sau này là GoLang
- Để thực hiện các việc như Build , Deploy , Run như thế thì Docker áp dụng 1 thuật ngữ là Containers . Containers cho phép người lập trình viên đóng gói source code với tất cả các điều kiện mà source code cần để chạy được như là thư viện , môi trường , hay là công cụ hỗ trợ.
- Dựa vào Container sau khi ta khai báo và đóng gói các package cần thiết thì ta có thể đưa nó vào máy chủ chạy bất kì hệ điều hành nào cũng được và phổ biến nhất ở đây là Linux, Ubuntu , Centos,.... Và có thể là Windows của Microsoft tùy thuộc vào máy chủ mà người lập trình viên đó thao tác với nó

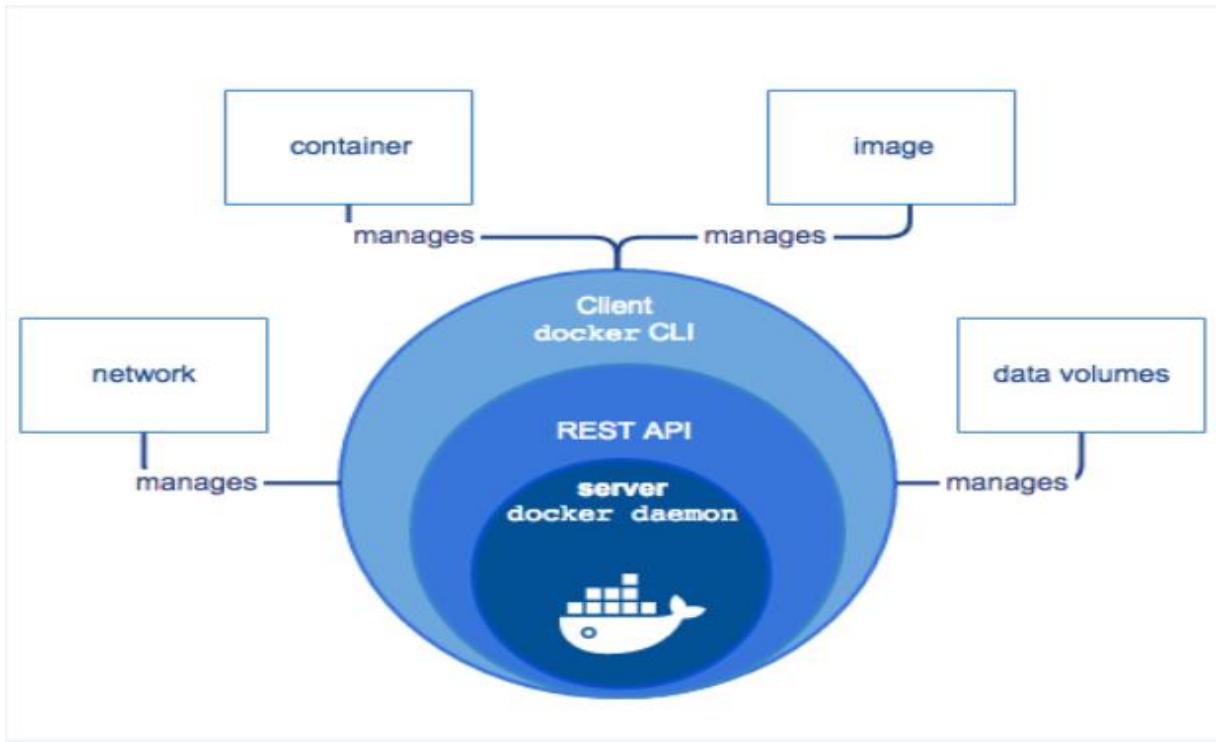
- Sự độc lập và bảo mật cho phép bạn chạy riêng biệt nhiều Container cùng một máy chủ. Các container hoạt động chiếm một lượng dung lượng của máy chủ tùy theo độ lớn nhỏ của dự án mà ta đưa vào nó.
- Docker cung cấp các công cụ và nền tảng để quản lý quá trình hoạt động và vòng đời container của bạn.
 - Phát triển ứng dụng của bạn và các thành phần hỗ trợ trong containers
 - Container cũng đóng vai trò là đơn vị cung cấp và thử nghiệm ứng dụng hay trang web của bạn
 - Nếu bạn có nhu cầu đưa ứng dụng hay trạng web của mình lên môi trường sản phẩm và đưa ra cho mọi người sử dụng dưới dạng là 1 container hoặc là một dịch vụ nào đó cho dù môi trường đó là nằm ở máy của bạn hay là nằm trên một máy chủ khác ở điện toán đám mây hoặc có thể là sử dụng cả hai.

Docker Engine là gì ?

Một tiến trình Docker daemon có nhiệm vụ các bước ban đầu để tạo và quản lý các container, image , volume , network

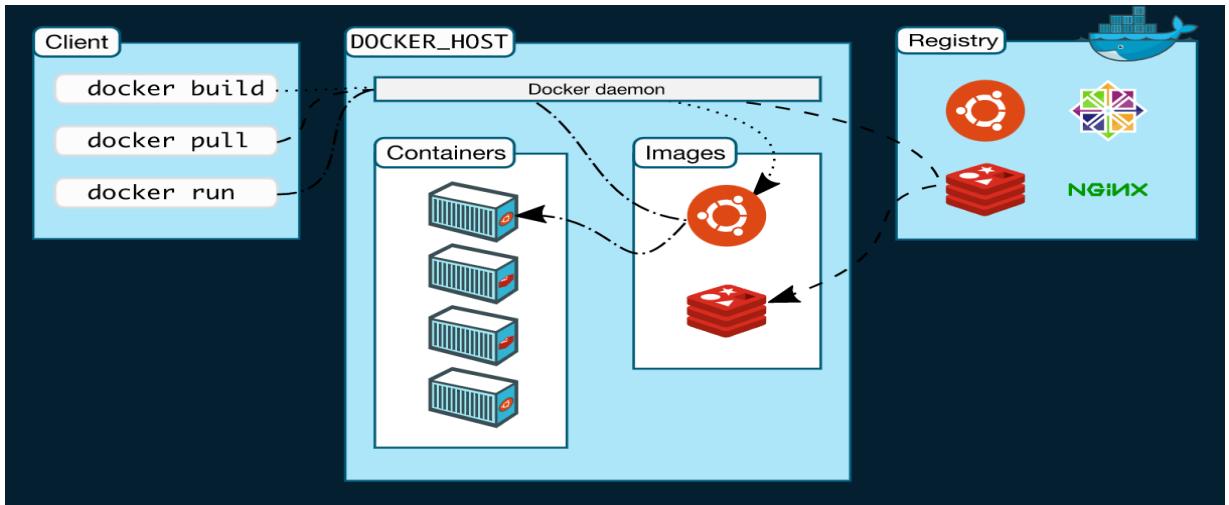
Docker Client sẽ giao tiếp với Docker daemon thông qua API theo chuẩn REST (tức có nghĩa sẽ có các phương thức GET , POST , PUT , DELETE) để yêu cầu Docker daemon thực thi các lệnh mà người dùng gọi đến

Docker CLI được xem là Client hay được hiểu là giao diện tương tác giữa người dùng ở đây là sử dụng các công cụ hỗ trợ như là Command Line (CMD), Window Terminal, Để gửi các yêu cầu tương ứng đến docker daemon thông qua API



Hình 29: Sơ đồ về kiến trúc của docker

Kiến trúc của Docker là Client – Server và giao tiếp qua REST API (Cổng mạng và một số phương thức khác) để có thể gửi các yêu cầu đến Docker daemon. Việc giao tiếp này thực hiện các công việc như Run , Deploy , Build các docker container. Ta có thể cấu hình cho Docker Client – Docker Daemon trên cùng 1 máy , hoặc có thể dùng Docker Client để giao tiếp từ xa với Docker Daemon. Và minh họa cho kiến trúc của docker như sau



Hình 30: Chi tiết và cách hoạt động và thao tác với docker

Một số thành phần khác của docker

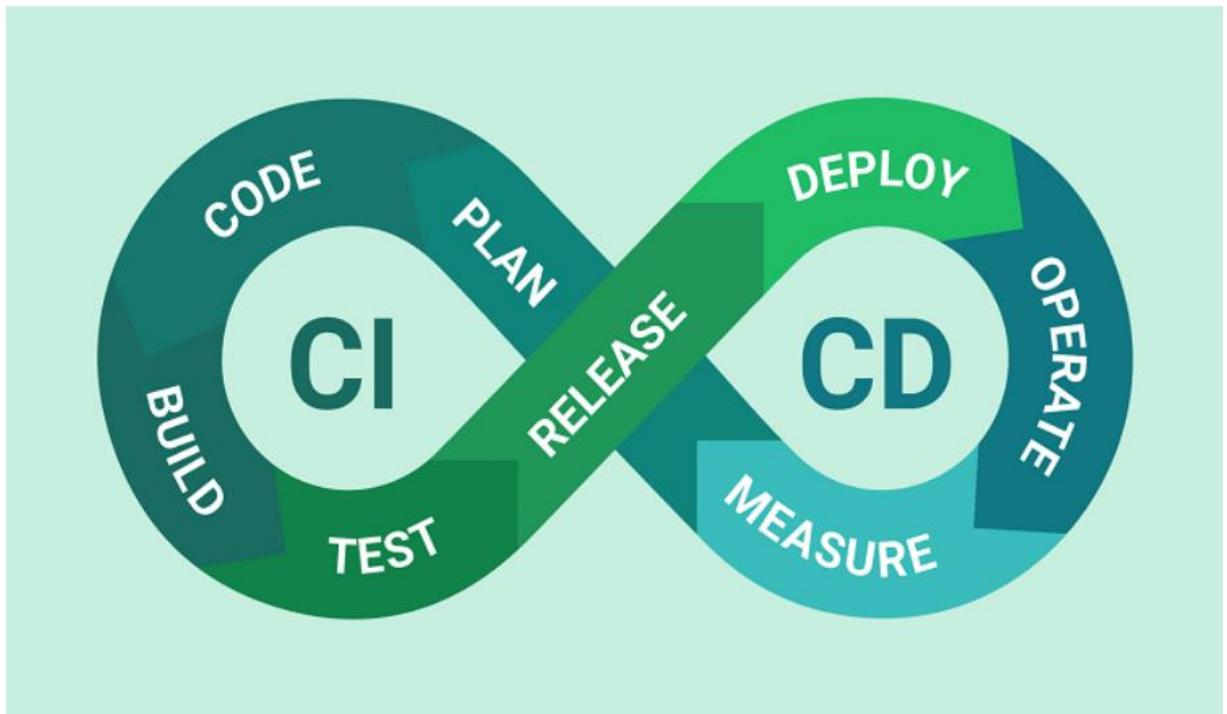
- Daemon Docker (dockerd) lắng nghe các yêu cầu của người dùng thông qua Docker API và quản lý các Docker đối tượng như image, container, network và volume. Một daemon cũng có thể giao tiếp với các daemon khác để quản lý dịch vụ Docker.
- Docker client (docker) là cách tốt nhất để người thiết lập docker giao tiếp với nó. Khi bạn sử dụng các câu lệnh như docker run, client thông qua Docker API sẽ gửi các câu lệnh này đến dockerd, thực hiện chúng. Docker client có thể giao tiếp với nhiều hơn một daemon Docker.
- Docker registry sẽ là nơi lưu trữ các hình ảnh Docker. Docker Hub là nơi lưu trữ Docker image public (public registry) mà bất kỳ ai cũng có thể sử dụng và Docker được định cấu hình mặc định để tìm kiếm hình ảnh trên Docker Hub. Ngoài ra, người dùng có thể cấu hình các tư vấn đăng ký khác nhau để lưu trữ hình ảnh Docker.

Khi người dùng sử dụng câu lệnh “docker pull” hoặc “docker run”, các image only sẽ được tải về dựa trên registry đã được cấu hình trước đó. Khi người dùng sử dụng

câu lệnh “docker push”, image cũng sẽ được tải lên registry mà người dùng đã cấu hình từ trước đó.

- Docker Image là template read-only (chỉ cho phép đọc) với các hướng dẫn để tạo Docker container. Image sẽ được sử dụng để đóng gói các ứng dụng và các thành phần đi kèm của ứng dụng, được lưu trữ ở server hoặc trên registry. Ví dụ bạn có thể sử dụng Dockerfile để tạo ra một Docker image sử dụng hệ điều hành Ubuntu và cài đặt Apache server với những cài đặt, cấu hình tùy chỉnh của riêng mình.
- Docker Container là 1 “runable instance” của image. Bạn có thể khởi tạo, dừng hoặc xóa container bằng cách sử dụng Docker API hoặc CLI. Bạn có thể kết nối container đến 1 hoặc nhiều network, thư mục lưu trữ, hoặc thậm chí tạo ra 1 image mới dựa trên tình trạng hiện tại của container. Mặc định 1 container được “cách ly” với các container và server nếu người dùng không có các cài đặt gì thêm.
- Docker volume được thiết kế để làm nơi lưu trữ các dữ liệu độc lập với vòng đời của container.
- Docker network : Cung cấp một private network mà chỉ tồn tại giữa container và server, giúp các container có thể giao tiếp được với nhau một cách dễ dàng.
- Docker Service cho phép bạn mở rộng các container thông qua nhiều Docker daemon, chúng giao tiếp với nhau thông qua swarm cluster bao gồm nhiều manager và worker. Mỗi một node của swarm là 1 Docker daemon giao tiếp với nhau bằng cách sử dụng Docker API. Theo mặc định thì service được cân bằng tải trên các node.

2) CI/CD



Hình 31: Quá trình hoạt động của CI/CD

CI (Continous Integration) là cách thức phát triển phần mềm bằng cách tích hợp liên tục có nghĩa là nếu trong một nhóm những người cùng một dự án khi thực hiện một tính năng hay tham gia vào việc bảo trì một tính năng nào đó và sinh ra các đoạn mã mới thay cho đoạn mã cũ và đưa đoạn mã trên nhánh của người đó lên nhánh tổng (master) hoặc là nhánh develop thì ở đây CI sẽ thực hiện công việc của nó là tự động chạy toàn bộ source code cùng với đoạn mã của người viết đoạn mã vừa mới đưa lên và kiểm tra và check xem độ chính xác về mặt quy tắc viết mã của ngôn ngữ mà source đó sử dụng hay không. Nếu không thì sẽ tiếp tục bước tiếp theo nhưng khi có vấn đề xảy ra thì sẽ báo lỗi cho người viết đoạn mã đó chưa khớp với đoạn mã trên nơi tổng mà các nhánh của các thành viên đó về đó, và cũng như đưa ra những gợi ý cho người viết mã là một đề xuất để sửa. CI cũng đảm bảo source code của bạn sau gom mã của thành viên trong nhóm lại sẽ có thể hoạt động ở bất kì điều kiện nào và không xảy ra lỗi. Quá trình Build Run và Deploy sẽ được chạy và cho người quản lý source code tiến trình nó xảy ra như thế nào.

Quá trình đưa mã code từ nhánh cá nhân lên nhánh chính (develop hoặc master) thì quá trình CI hoạt động sẽ được hoạt động tự động nếu người quản lý source code cấu hình và cài đặt trước . Nên quá trình gom source code -> kiểm tra -> phản hồi sẽ diễn ra liền mạch và tự động. Người xây dựng mã người có thể tiết kiệm thời gian để xây dựng tính năng khác hoặc sửa lỗi nếu có và sau khi quá trình kết thúc thì người quản lý có thể xem được quá trình build và test.



Hình 32: Quá trình phát triển ứng dụng.

Lợi ích của việc sử dụng CI

- Giảm thiểu rủi ro nhờ việc phát hiện lỗi và fix sớm, tăng chất lượng phần mềm nhờ việc tự động test và inspect (đây cũng là một trong những lợi ích của CI,

code được inspect tự động dựa theo config đã cài đặt, đảm bảo coding style, chẳng hạn một function chỉ được dài không quá 10 dòng code ...)

- Giảm thiểu những quy trình thủ công lặp đi lặp lại (build css, js, migrate, test...), thay vì đó là build tự động, chạy test tự động
- Sinh ra phần mềm có thể deploy ở bất kì thời gian, địa điểm

CD là gì : Thì đi kèm với CI là việc tích hợp liên tục thì còn một quá trình nữa gọi là chuyển giao liên tục thì nếu source code qua quá trình CI không có lỗi xảy ra thì sẽ sang một giai đoạn gọi là CD để tiến hành kiểm tra những thay đổi về mã nguồn đã được build lên và mã nguồn trong môi trường kiểm thử trước đó. CD cho phép người viết mã nguồn cài đặt những chu kỳ tự động ngắn và liên tục . Từ đó nhanh chóng phát hiện ra lỗi sai và tiến hành chỉnh sửa ngay trong những phần tương tự.

Do sự tiện lợi của 2 quá trình này thì người quản lý mã nguồn thường tích hợp cả hai CI/CD với nhau để tiết kiệm thời gian , chi phí và tài nguyên bộ nhớ cho 2 quá trình này, CD sẽ sử dụng giai đoạn cuối cùng của CI để bắt đầu quá trình chuyển giao của chính nó

Trong quá trình CD thực hiện tiến trình của mình thì có thể mã nguồn sẽ trải qua một số phần testing được người quản lý mã nguồn thiết lập và một số bài test bao gồm UI Testing , API Testing, CD sử dụng một thuật ngữ được gọi là Deployment Pipeline giúp chia quy trình chuyển giao thành các giao đoạn. Mỗi giai đoạn sẽ đáp ứng những mục đích riêng bao gồm xác minh chất lượng của các tính năng từ một số góc độ khác nhau để kiểm định chức năng và tránh lỗi ảnh hưởng đến người dùng. Pipeline sẽ thể hiện phản hồi cho nhóm trong việc cung cấp tính năng mới hoặc sửa chữa các tính năng đã có sẵn. Deployment Pipelien là quy trình để chuyển phần mềm từ version control đến tay người dùng.

Việc kết hợp 2 quá trình này thì ta sẽ được chu trình hoạt động của CICD gói gọn trong 3 bước là Xây dựng -> Kiểm tra -> Triển khai



Hình 33: Cách hoạt động của CI/CD

Lợi ích của CI/CD

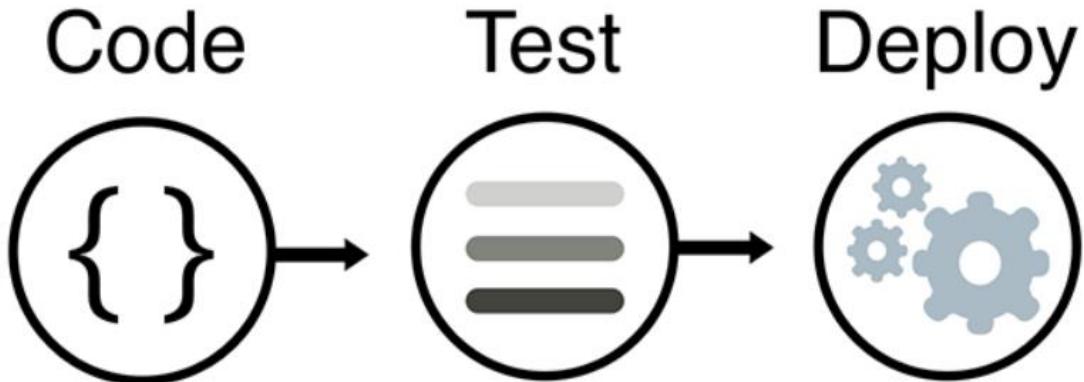
Giảm thiểu rủi ro không đáng có : là một lợi ích vô cùng hữu ích của CI/CD, nó cho phép làm giảm thiểu đi những rủi ro nhờ việc phát hiện và sửa lỗi sớm, giúp tăng chất lượng sản phẩm nhờ khả năng tự động kiểm tra và quan sát. Không những vậy, những quy trình thủ công lặp đi lặp lại hằng ngày cũng được giảm tải, thay vào đó là xây dựng và kiểm thử tự động mà không cần đến sự giúp đỡ của con người. Một đặc điểm nữa của CI CD chính là có thể deploy, triển khai phần mềm ở bất cứ địa điểm và thời gian nào

Thay đổi code nhỏ : Một lợi ích vô cùng lớn của CI/CD chính là cho phép chúng ta tích hợp nhiều loại mã nhỏ cùng một lúc. Những thay đổi mã này được thực hiện một

cách đơn giản và xử lý nhanh hơn so với những đoạn mã không lồ, từ đó làm giảm đi khả năng sinh ra những vấn đề liên quan đến việc thay đổi sau này.

Những sự thay đổi mã nhỏ này có thể được thực kiểm tra ngay sau khi chúng được tích hợp vào kho mã. Các nhà phát triển có thể dễ dàng nhận ra vấn đề trước khi lượng lớn công việc tăng lên một cách chóng mặt. Đây thực sự là một lợi thế đối với những nhóm phát triển lớn hoặc người làm việc từ xa giao tiếp được hiệu quả hơn.

Hạn chế những ảnh hưởng của lỗi: CI/CD được thiết kế với hệ thống sao cho khi có lỗi nào đó xảy ra thì những kết quả tiêu cực sẽ bị giới hạn trong phạm vi ảnh hưởng nhất định nào đó. Việc hạn chế các vấn đề này giúp làm giảm khả năng hư hỏng từ đó làm cho hệ thống được bảo trì và xử lý một cách dễ dàng hơn. Với hệ thống CI CD, có thể đảm bảo cho việc cách ly lỗi sẽ được phát hiện một cách nhanh chóng và dễ dàng thực hiện hơn. Chính vì vậy, hậu quả của các lỗi trong ứng dụng sẽ được giới hạn trong phạm vi ảnh hưởng của nó.



Hình 34: Tổng quan về quá trình hoạt động với CI/CD

Nhược điểm của CI/CD là gì ?

- Bên cạnh các ưu điểm thì hệ thống CI và CD vẫn tồn tại một số điểm trừ nhất định. Vì CI/CD thường xuyên được nâng cấp để tối ưu hơn nên đòi hỏi các nhà lập trình phải nhanh chóng học hỏi các kiến thức mới.
- Ngoài ra, để quá trình tự động hóa diễn ra trơn tu, người sử dụng phải hiểu rõ cách vận hành của chương trình. Và điều này hoàn toàn không đơn giản bởi CI và CD sở hữu bộ kỹ năng khá phức tạp. Để hoàn toàn sử dụng trọn vẹn được CI/CD đòi hỏi quá trình rèn luyện lâu dài.
- Để có thể ứng dụng được CI và CD vào công việc đòi hỏi nhà sản xuất phải đầu tư trước một khoản phí khá lớn. Không nhiều công ty dám mạo hiểm bởi như đã nói ở trên, việc sử dụng CI và CD không dễ dàng. Và nhược điểm cuối cùng là một số hệ thống lập trình không cho phép hỗ trợ thêm CI và CD. Nếu không thể thay đổi hệ thống thì khó lòng sử dụng được chương trình này.

3) VPS

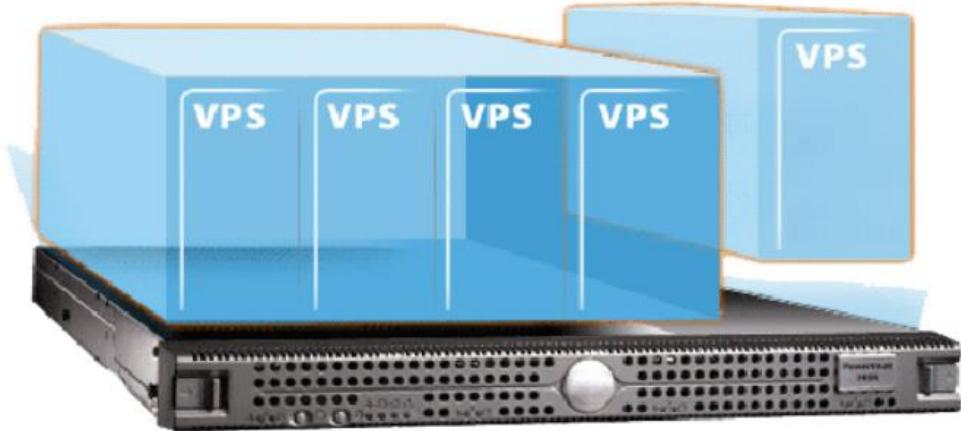
VPS là viết tắt của (Virtual Private Server) hay còn gọi là máy chủ riêng ảo. Được cung cấp và cho thuê bởi các công ty dịch vụ Internet Hosting được tạo ra bởi công nghệ ảo hóa.

VPS hoạt động như một server để người dùng tạo ra trang web hay ứng dụng trên các tài nguyên ta cài đặt được giành riêng

Người cài đặt VPS có toàn quyền kiểm soát bằng superuser (Root , quyền admin), và thông qua SSH key để điều khiển cài đặt các thư viện, hoặc cấu hình VPS theo ý muốn của người cài đặt.

Thường thì VPS sẽ được thuê và có những option các hệ điều hành có sẵn khi người muốn thuê lựa chọn các dịch vụ và trả tiền để thanh toán phí duy trì của cái máy ảo đó ở máy chủ của công ty hosting đó một số hosting phổ biến như AWS , Digital Ocean , Microsoft Azure , Ở Việt Nam thì có Mắt Bão , trên điện toán đám mây

VPS an toàn và ổn định tùy theo tình trạng mạng của bạn kết nối và ổn định hơn so với host website vì bạn không chia sẻ không gian lưu trữ với người khác. Và nó cũng có giá thành thấp hơn so với thuê một server riêng



Thường thì các máy chủ ảo sẽ không có tên miền để thay thế cho việc truy cập đến máy chủ đó mà phải truy cập qua IP của VPS đó được sinh ra khi bạn tiến hành thuê VPS đó

VPS là một server ảo hoạt động tương tự như một máy chủ vật lý. Máy chủ , hay server, là một máy tính chứa source code và cơ sở dữ liệu cho ứng dụng của bạn , Môi trường và các thư viện cần thiết. Bất kể khi nào người dùng truy cập vào ứng dụng của bạn trên trình duyệt thì sẽ gửi yêu cầu đến con server của bạn và lúc này dựa vào đường truyền internet các file cần thiết để chạy 1 trang web sẽ được truyền xuống máy tính của người dùng và hiện thị nó trên trình duyệt

Trên thực tế thì VPS là máy chủ ảo được cấu hình qua mạng như nó sẽ được hình thành trên một server vật lý, và mỗi VPS là sẽ có phần tài nguyên riêng biệt của nó và không bị ảnh hưởng bởi các VPS khác

Các yếu tố lưu ý đến VPS khi tiến hành thuê

- **Ram** : Trong máy tính, RAM là loại bộ nhớ chính, nếu VPS của bạn có nhiều RAM thì khả năng truy xuất dữ liệu càng tốt. Bởi vì khi sử dụng VPS bạn sẽ cần RAM để xử lý các vấn đề như xử lý truy vấn nhập xuất của database
- **SWAP** hay là bộ nhớ ảo có nhiệm vụ lưu lại các hành động trong quá khứ trong trường hợp bộ nhớ RAM bị quá tải (overload). Hay nói cách dễ hiểu hơn, SWAP là nơi lưu trữ trên ổ cứng thay vì một bộ nhớ độc lập. Tuy nhiên, không phải VPS nào cũng hỗ trợ SWAP
- **Disk** : Ổ cứng/Ổ đĩa cứng hay còn gọi là Disk là không gian lưu trữ được sử dụng để lưu các file cài đặt của hệ điều hành và các file của mã nguồn website. Có 2 loại ổ đĩa được sử dụng phổ biến, đó là:
 - o HDD (Hard Disk Drive): Là loại ổ đĩa thông dụng được sử dụng trên máy tính.
 - o SSD (Solid State Drive): SSD hay còn gọi là ổ cứng bán dẫn được sử dụng để lưu trữ dữ liệu nhưng tốc độ truy xuất dữ liệu nhanh hơn so với HDD 300 lần.

Do vậy, giá thành của ổ cứng SSD sẽ có đắt hơn so với loại ổ HDD.

- **CPU Core** : là lõi xử lý của CPU. Một máy chủ riêng sẽ có lượng core nhất định và nó được chia cho các VPS. Thông thường, số core càng cao thì khả năng xử lý dữ liệu càng tốt. Số lượng core trung bình ở các gói VPS là từ 1 core đến 3 core.
- **Bandwidth và transfer** : đều có nghĩa là băng thông, chính là lưu lượng được phép truyền tải dữ liệu đi.
- **IP (Internet Protocol)** : là số lượng địa chỉ IP mà nhà cung cấp dịch vụ VPS sẽ cấp cho bạn.

Các công ty cho thuê dịch vụ máy chủ ảo riêng biệt như thế sẽ cho người thuê toàn quyền quyết định về hệ điều hành, cũng như cho phép chọn cấu hình của phân vùng VPS đó như sử dụng bao nhiêu RAM xử lý , Bao nhiêu GB bộ nhớ , Thì Ram và bộ nhớ

càng cao ứng dụng của bạn sẽ hoạt động một cách trơn tru hơn nhưng bù lại chi phí của của người thuê bỏ ra để duy trì nó sẽ càng cao hơn.



Ưu điểm của VPS là gì ?

- Nhanh và đáng tin cậy hơn server shared hosting.
- Vì được đảm bảo về thông số server như bộ nhớ và sức mạnh vi xử lý, bạn sẽ không gặp phải vấn đề tài nguyên bị người khác dùng hết.
- Các vấn đề về lượng truy cập đột biến tăng cao không ảnh hưởng đến site của bạn.
- Bạn có quyền superuser (root) trên server.
- Có độ riêng tư cao hơn, vì files và databases bị khóa khỏi hệ thống server của các người dùng khác.
- Dễ dàng nâng cấp. Ngay khi website tăng trưởng, bạn chỉ cần nâng cấp gói hosting để nâng tài nguyên lên mà không phải tốn công chuyển dữ liệu hay chuyển server (RAM, CPU, disk space, bandwidth,...).

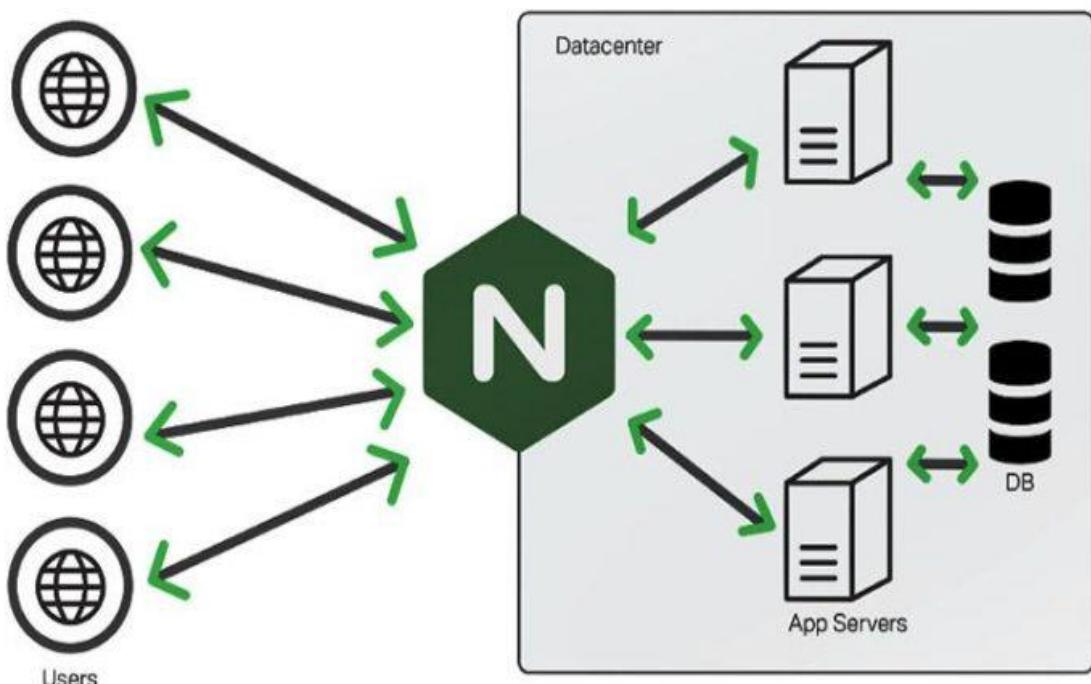
- IP riêng. Chính vì điều này khả năng chống DDoS cũng cao hơn.

Nhược điểm

- Giá cao hơn shared hosting.
- Cần ít nhiều kiến thức kỹ thuật để quản lý.
- Cấu hình server không đúng có thể tạo ra lỗ hổng bảo mật.

4) NGINX

Để đưa trang ứng dụng dịch vụ của mình cho mọi người sử dụng mạng có thể tìm thấy và sử dụng trang web của mình thì công cụ hỗ trợ và cần phải được người cấu hình trang web cài đặt và sử dụng đó là Nginx hoặc là Apache và các công cụ này gọi chung là web server chúng giúp cho trang web của ta có thể được mọi người tìm kiếm hoặc truy cập thông qua các trình duyệt như google chrome hay cốc cốc, ...



Hình 35 : Nginx là gì

Nginx là được biết là một web server mã nguồn mở. Nginx sử dụng kiến trúc đơn luồng với mục đích là tối đa hiệu năng và sự ổn định. Ngoài việc nginx có thể tận dụng được khả năng của máy chủ HTTP còn có thể hoạt động như một proxy server cho email

(với các kiểu gửi mail như IMAP, POP3 và SMTP), reverse proxy, cân bằng tải cho server (load balancing), Lưu lại thao tác với trang web thông qua trình duyệt (HTTP caching). Nginx là một công cụ và thao tác với nginx là một kiến thức cần có đối với những lập trình viên liên quan đến web, quản trị hệ thống (system administrator), Devops

Một số công ty lớn cũng sử dụng Nginx làm web server cho các trang ứng dụng của họ như LinkedIn, Facebook, Cisco, Twitter

Nginx giúp tối ưu hóa về năng suất cho trang web của người lập trình. Đối với các web server truyền thống cứ với mỗi yêu cầu cho trang web thì server sẽ sinh ra 1 luồng xử lý dành riêng cho yêu cầu đó nhưng khi ta tăng số lượng yêu cầu cho server lên rất nhiều thì sẽ sinh ra rất nhiều luồng và khi đó server sẽ dẫn đến việc là thiếu tài nguyên để đáp ứng cho một yêu cầu.

Còn riêng đối với nginx là web server sẽ hoạt động không dựa trên luồng mà Nginx được xây dựng trên kiến trúc xử lý bất đồng bộ và điều hướng tài nguyên cho các yêu cầu một cách linh hoạt.

Một số tiện ích của nginx như

- Tạo ra khả năng xử lý hơn đến 10.000 kết nối cùng một lúc với các bộ nhớ thấp.
- Hỗ trợ phục vụ các tập tin tĩnh và lập ra các chỉ mục tập tin phù hợp.
- Có khả năng tăng tốc reverse proxy bằng các bộ nhớ đệm giúp cân bằng tải đơn giản hơn với khả năng chịu lỗi vô cùng cao.
- Nginx có thể hỗ trợ tăng tốc cùng với bộ nhớ FastCGI, uwsgi, SCGI và những máy chủ memcached vô cùng hiệu quả.
- Kiến trúc modular cho phép bạn tăng tốc độ nạp trang bằng biện pháp nén gzip một cách tự động.
- Nginx có khả năng hỗ trợ thực hiện mã hóa SSL và TLS.

- Cấu hình của Nginx vô cùng linh hoạt giúp lưu lại nhật ký truy vấn một cách dễ dàng.
- Nginx có khả năng chuyển hướng lỗi 3XX-5XX.
- Rewrite URL có thể sử dụng expression.
- Nginx có thể hạn chế tỷ lệ đáp ứng của truy vấn.
- Nginx giúp giới hạn số kết nối đồng thời cũng như truy vấn từ 1 địa chỉ.
- Nginx có khả năng nhúng mã PERL một cách dễ dàng.
- Nginx có thể hỗ trợ và tương thích hoàn toàn với IPv6.
- Nginx có thể hỗ trợ cho websockets.
- Nginx hỗ trợ truyền tải các file FLV và MP4.

5) NODEJS(EXPRESS)

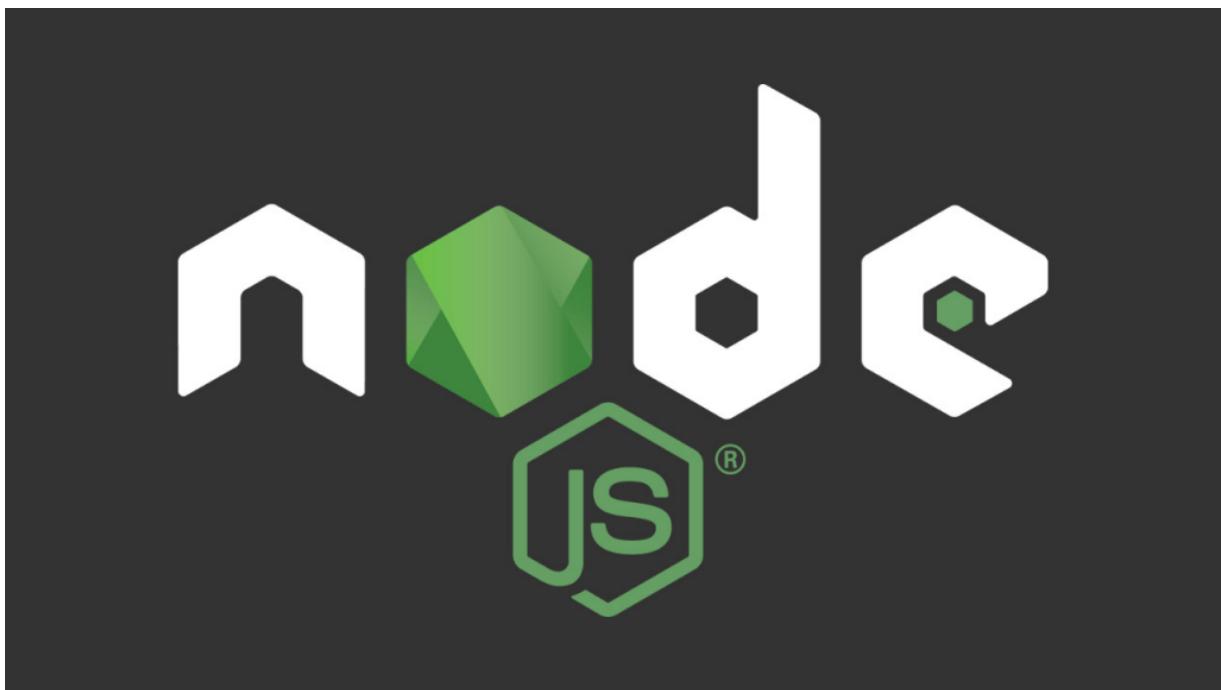
NodeJS được xem là nền tảng để viết và chạy các mã nguồn thuộc về ngôn ngữ JavaScript, Nó cho biến ngôn ngữ JavaScript từ một ngôn ngữ chỉ hoạt động trên các client web thành một thứ mà các lập trình viên có thể sử dụng nó để tạo ra các ứng dụng độc lập trong các lĩnh vực như

- Ứng dụng trò chuyện trong thời gian thực: Nhờ vào cấu trúc không đồng bộ đơn luồng, Node.JS rất thích hợp cho mục đích xử lý giao tiếp trong thời gian thực. Nền tảng này có thể dễ dàng mở rộng quy mô và thường dùng để tạo ra các chatbot.
- Internet of Things (IoT): Các ứng dụng IoT thường bao gồm nhiều bộ cảm biến phức tạp để gửi những phần dữ liệu nhỏ.
- Truyền dữ liệu: Netflix là một trong số những công ty lớn trên thế giới chuyên sử dụng Node.JS cho mục đích truyền dữ liệu.
- Các SPA (Single-page application) phức tạp: Trong SPA, toàn bộ ứng dụng được load vào trong một trang duy nhất, do đó sẽ có một số request được thực hiện trong nền.

- Các ứng dụng REST dựa trên API: JavaScript được sử dụng trong cả frontend lẫn backend của trang. Do đó một server có thể dễ dàng giao tiếp với frontend qua REST API bằng Node.js

ExpressJS là một Framework nhỏ, nhưng linh hoạt được xây dựng trên nền tảng của Nodejs. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile

- Về các package hỗ trợ: Expressjs có vô số các package hỗ trợ nên các bạn không phải lo lắng khi làm việc với Framework này.
- Về performance: Express cung cấp thêm về các tính năng (feature) để dev lập trình tốt hơn. Chứ không làm giảm tốc độ của NodeJS.



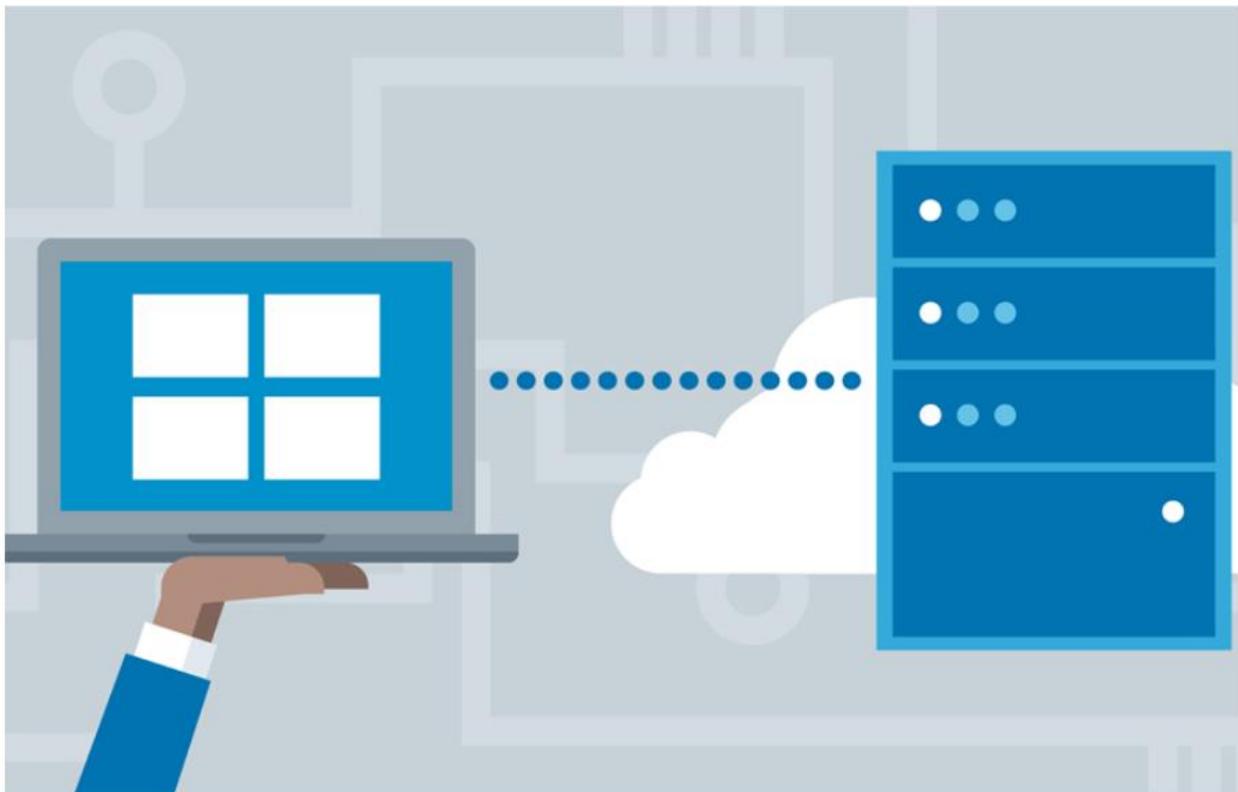
6) *ORM (PRISMA)*

SQL server hay còn được gọi là Microsoft SQL Server, nó từ viết tắt của MS SQL Server. Đây chính là một loại phần mềm đã được phát triển bởi Microsoft và nó được sử dụng để có thể dễ dàng lưu trữ cho những dữ liệu dựa theo tiêu chuẩn RDBMS.

SQL Server có khả năng cung cấp đầy đủ các công cụ cho việc quản lý từ giao diện GUI đến sử dụng ngôn ngữ cho việc truy vấn SQL. Điểm mạnh của SQL điểm mạnh của nó là có thể kết hợp và hoạt động với các nền tảng khác nhau

Để người viết code giao tiếp với SQL server sẽ được giao tiếp với nhau thông qua một ngôn ngữ gọi là SQL , T-SQL

- T-SQL là một trong những loại ngôn ngữ thuộc quyền sở hữu của Microsoft và được gọi với cái tên Transact-SQL. Nó thường cung cấp thêm rất nhiều cho các khả năng khai báo biến, thủ tục lưu trữ và xử lý ngoại lệ,...
- SQL Server Management Studio là một loại công cụ giao diện chính cho máy chủ cơ sở của chính dữ liệu SQL, thông thường thì nó hỗ trợ cho cả môi trường 64 bit và 32 bit.



Thì hiện nay trong thực tế có một số cách thực hiện tạo Database mà không cần thông qua GUI của SQL server mà dùng các đối tượng được định nghĩa và nhờ vào thư

viện để ánh xạ các đối tượng đó vào trong cơ sở dữ liệu để hình thành các bảng và mối quan hệ của chúng thì kĩ thuật đó được gọi là ORM

Vậy ORM là từ viết tắt của (Object Relational Mapping), là một kỹ thuật/cơ chế lập trình thực hiện ánh xạ CSDL sang các đối tượng trong các ngôn ngữ lập trình hướng đối tượng như Java, C# ... (các table tương ứng các class, mối ràng buộc giữa các table tương ứng quan hệ giữa các class ‘has a’ , ‘is a’).

Việc sử dụng ORM cho phép lập trình viên thao tác với database 1 cách hoàn toàn tự nhiên, dễ hiểu thông qua các đối tượng. Lập trình viên không cần tới loại database, kiểu dữ liệu trong database...

Ưu điểm của ORM:

- ORM giúp lập trình viên tập trung vào lập trình hướng đối tượng
- Tính độc lập: Làm việc được với nhiều loại database(hệ quản trị cơ sở dữ liệu), nhiều kiểu dữ liệu khác nhau. Dễ dàng thay đổi loại database hơn. Các câu lệnh SQL không phụ thuộc vào loại database.
- Đơn giản, dễ sử dụng: Hỗ trợ HSQL, cung cấp nhiều API truy vấn.
- Năng suất hơn: viết code ít hơn, dễ hiểu hơn. Phù hợp các case CRUD (Create, Read, Update, Delete)
- Khả năng sử dụng lại code.

Nhược điểm của ORM:

- Khả năng truy vấn bị hạn chế, nhiều trường hợp ta vẫn phải dùng native SQL để truy vấn database.
- Khó tối ưu câu lệnh SQL (do câu lệnh SQL được ORM tự động sinh ra).

Một thư viện hỗ trợ việc ánh xạ dữ liệu từ đối tượng trong khi xây dựng back-end bằng NodeJS đó là thư viện Prisma . Thư viện giúp ánh xạ các đối tượng ta khai báo từ dạng model được quy định thành các bảng trong cơ sở dữ liệu mà ta muốn thì ở đây chúng em chọn SQL Server

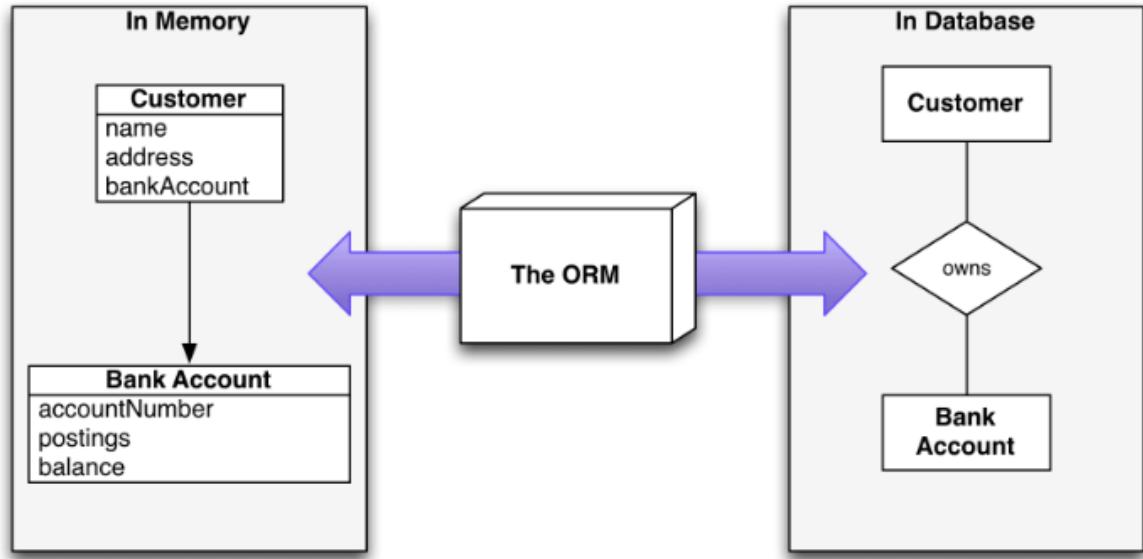
Prisma là một Object Relational Mapping (ORMs) được dùng để xây dựng các máy chủ như GraphQL Server, SQL Server, RESTful APIs, microservice, ... Prisma đơn giản là 1 layer nằm giữa Webserver và Database. Prisma giúp chúng ta giao tiếp với db một cách dễ dàng hơn.

Prisma bao gồm ba phần chính:

- Prisma Client : Trình tạo truy vấn an toàn và được tạo tự động cho Node.js và TypeScript.
- Prisma Migrate : Một hệ thống di chuyển và mô hình hóa dữ liệu khai báo.
- Prisma Studio : Một GUI để xem và chỉnh sửa dữ liệu trong cơ sở dữ liệu của bạn.

Cách truyền thông mà Webserver giao tiếp với Database là thông qua các câu lệnh SQL Query như SELECT, UPDATE hay DELETE. Giờ đây, nhờ vào các công cụ ORMs nói chung và Prisma nói riêng. Chúng tạo ra một tầng abstraction giữa Webserver và Database. Điều này giúp cho lập trình viên dễ dàng trong việc thao tác với Database. Thay vì viết những câu lệnh SQL khô khan, có thể sai sót lúc nào thì chúng ta có thể viết các hàm tương ứng.





```
// Bảng loại người dùng
model User_Type {
    id          BigInt      @id @default(autoincrement())
    user_type_name String
    is_active    Boolean     @default(true)
    is_delete    Boolean     @default(false)
    create_date  DateTime?
    create_user  String?
    update_date  DateTime?
    update_user  String?
    delete_date  DateTime?
    delete_user  String?
    account      User_Account[]
}
```

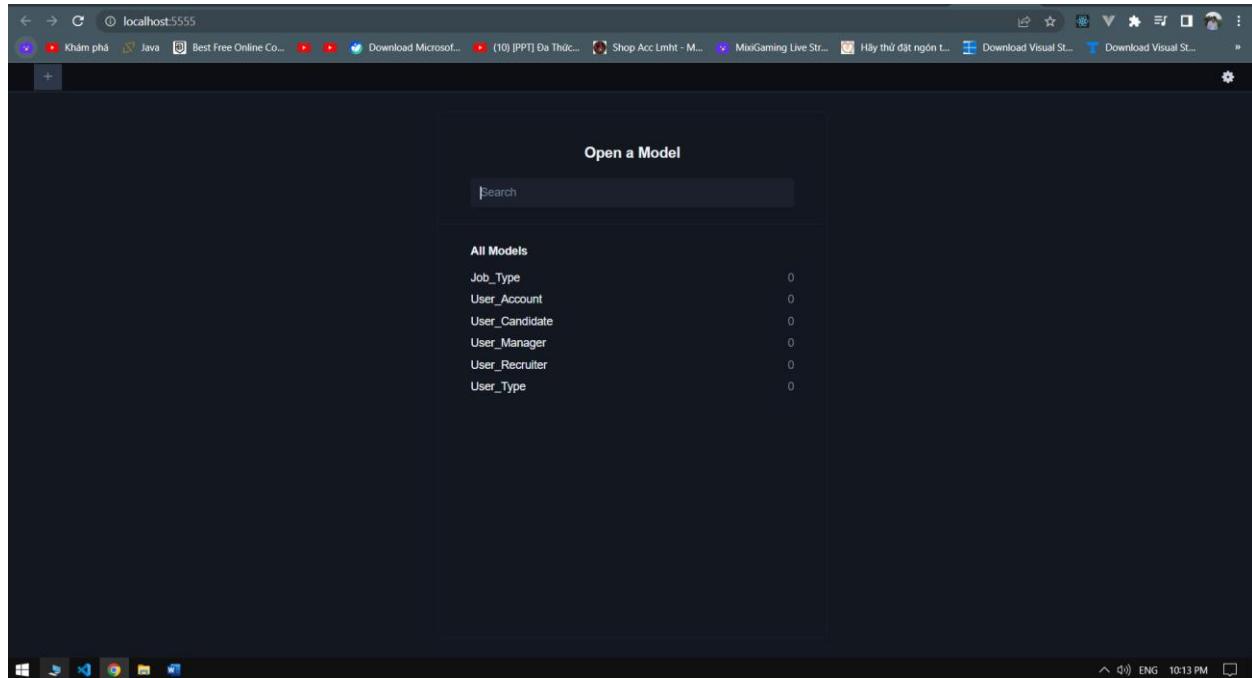
```
// Bảng tài khoản
model User_Account {
    id          BigInt      @id @default(autoincrement())
    password    String
    username    String
    google_id   String?
    is_active   Boolean    @default(true)
    is_delete   Boolean    @default(false)
    create_date DateTime   @default(now())
    user_type   User_Type  @relation(fields: [user_type_id], references: [id])
    user_type_id BigInt
    first_name  String?
    last_name   String?
    full_name   String?
    email       String?
    number_phone String?
    age         Int?
    gender      Int?
    address    String?
    avartar     String?
    logo        String?
}
```

Trên đây là 2 bảng được khởi tạo trong database được khởi tạo thành 2 bảng là User_type và User_Account thì các giá trị bắt buộc không được bỏ trống thì sẽ không có dấu ? sau kiểu dữ liệu

Và cũng như khóa ngoại của bảng User_account sẽ là id trong bảng User_type
Thì với thư viện Prisma ta có thể ánh xạ các model thành các bảng trong cở sở dữ liệu và thư viện này có hỗ trợ một số loại cơ sở dữ liệu phổ biến hiện nay như: PostgreSQL , MySQL, SQLite, MongoDB, SQL Server

Để tạo bảng trong các cơ sở dữ liệu ta chỉ cần thực hiện các thao tác sau

- npx prisma studio : để mở GUI cho người viết code thấy được giao diện của prisma cùng với các bảng của mình



- npx prisma migrate : Để khởi tạo các bảng lưu lại quá trình thay đổi cũng như các file có dạng SQL ứng với mỗi migrate sẽ có 1 file SQL sinh ra và Prisma sẽ dựa vào tất cả các file SQL ở từng thư mục trong migrate để khởi tạo cơ sở dữ liệu cũng như là cập nhật các thuộc tính trong quá trình ta thao tác với các model
- Sau khi chạy câu lệnh này ta trong SQL Server ta khởi tạo và gán địa chỉ vào config để khi tạo từ prisma thì các bảng sẽ vào cơ sở dữ liệu đó sẽ hình thành các bảng như các model mà ta đã khai báo trong Prisma ở NodeJS

7) *ReactJS*

ReactJS là một opensource được phát triển bởi Facebook, ra mắt vào năm 2013, bản thân nó là một thư viện JavaScript chuyên giúp các nhà phát triển xây dựng giao diện người dùng hay UI, Nó được phân loại thành kiểu “V” trong mô hình MVC (Model-View-Controller). Một trong những điểm nổi bật nhất của ReactJS đó là việc render dữ liệu không chỉ thực hiện được trên tầng Server mà còn ở dưới Client nữa.

React được sử dụng bởi hàng trăm công ty lớn trên thế giới, bao gồm Netflix, Airbnb, American Express, Facebook, WhatsApp, eBay, và Instagram.

Hơn nữa, để tăng tốc quá trình phát triển và giảm thiểu những rủi ro có thể xảy ra trong khi coding, React còn cung cấp cho chúng ta khả năng Reusable Code (tái sử dụng code)

a. JSX

JSX (nói ngắn gọn cho JavaScript extension) là một cú pháp mở rộng cho JavaScript. Nó giúp chúng ta dễ dàng thay đổi cây DOM bằng các HTML-style code đơn giản.

JSX = Javascript + XML. Nó transform cú pháp dạng gần như XML về thành Javascript. Giúp người lập trình có thể code ReactJS bằng cú pháp của XML thay vì sử dụng Javascript. Các XML elements, attributes và children được chuyển đổi thành các đối số truyền vào React.createElement.

XML là viết tắt của từ eXtensible Markup Language, hay còn gọi là ngôn ngữ đánh dấu mở rộng do W3C đề nghị với mục đích tạo ra các ngôn ngữ đánh dấu khác. Đây là một tập hợp con đơn giản có thể mô tả nhiều loại dữ liệu khác nhau nên rất hữu ích trong việc chia sẻ dữ liệu giữa các hệ thống. Điểm hình nhất là ngôn ngữ đánh dấu siêu văn bản HTML sử dụng cú pháp của XML để tạo nên và nó có các bộ phần tử và thuộc tính không mềm dẻo nên chỉ có tác dụng trong việc trình bày dữ liệu trên trình duyệt Browser.

Cú pháp của tài liệu XML XML được xây dựng dựa vào cấu trúc NODE lồng nhau, mỗi node sẽ có một thẻ mở và một thẻ đóng như sau:

```
<nodename>nội dung</nodename>
```

Trong đó:

- <nodename> là thẻ mở, tên của thẻ này do bạn tự định nghĩa.
- </nodename> là thẻ đóng, tên của thẻ này phải trùng với tên của thẻ mở.
- content là nội dung của thẻ này

Cú pháp của JSX cũng tương tự như XML.

- Ta có thẻ mở tag:

```
<JSXElementName JSXAttributes>
```

- Đóng tag:

```
</JSXElementName>
```

- Ở đây lưu ý tên của thẻ mở tag và đóng tag phải giống nhau. ví dụ :

```
<MyButton color="blue" shadowSize={2}>
```

Click Me

```
</MyButton>
```

- Ngoài ra JSX cũng có SelfClosingElement:

```
<JSXElementName JSXAttributes/>
```

- Ví dụ:

```
<div className="sidebar" />
```

b. Tại sao lại nên dùng JSX

Việc sử dụng JSX trong ReactJS là không bắt buộc. Bạn có thể sử dụng chỉ JS thuần thô. Thay vì tách biệt các công nghệ một cách giả tạo bằng cách đưa định nghĩa giao diện và logic vào những tệp khác nhau, React tách bạch mối quan hệ bằng những đơn vị rời rạc gọi là “components” chứa cả hai cái trên. Nhưng có rất nhiều lý do cho việc nên sử dụng JSX trong ReactJS đây.

- Thứ nhất, JSX với cú pháp gần giống XML, cấu trúc cây khi biểu thị các attributes, điều đó giúp ta dễ dàng định nghĩa, quản lý được các component phức tạp, thay vì việc phải định nghĩa và gọi ra nhiều hàm hoặc object trong javascript. Khi nhìn vào cấu trúc đó cũng dễ dàng đọc hiểu được ý nghĩa của các component. Code JSX ngắn hơn, dễ hiểu hơn code JS.
- Thứ 2, JSX không làm thay đổi ngữ nghĩa của Javascript
- Thứ 3, với cách viết gần với các thẻ HTML, nó giúp những developers thông thường (ví dụ như các designer) có thể hiểu được một cách dễ dàng, từ đó có thể viết hoặc sửa code mà không gặp nhiều khó khăn. Ví dụ với đoạn code JSX như sau:

```
const myelement = <h1>I Love JSX!</h1>;
ReactDOM.render(myelement, document.getElementById('root'));
```

- Không có JSX, đoạn code sẽ như sau:

```
const myelement = React.createElement('h1', { }, 'I do not use JSX!');
```

```
ReactDOM.render(myelement, document.getElementById('root'));
```

c. Components

Components cho phép bạn chia UI thành các phần độc lập, có thể tái sử dụng, và hoàn toàn tách biệt nhau. Về mặt khái niệm, components cũng giống như các hàm Javascript. Chúng nhận vào bất kì đâu vào nào (còn được gọi là “props”) và trả về các React elements mô tả những gì sẽ xuất hiện trên màn hình.

Có 2 kiểu components 1 là function components 2 là class components nhưng chúng ta chỉ bàn về function components ở đây.

Cách đơn giản nhất để định nghĩa một component đó là viết một hàm JavaScript:

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Hàm này là một React component hợp lệ vì nó nhận đầu vào là một tham số object “props” (viết tắt của properties) với dữ liệu và trả về một React element. Chúng ta gọi các components này là “function components” vì chúng là các hàm JavaScript theo đúng nghĩa đen.

Component có thể sử dụng các HTML tag

```
function App() {return (<div><Welcome name="Sara" /> );}
```

Chú ý: Luôn luôn bắt đầu tên của component bằng chữ in hoa.

React xử lý các components bắt đầu với chữ thường giống như các DOM tags. Ví dụ, `<div />` biểu diễn HTML div tag, nhưng `<Welcome />` biểu diễn một component và yêu cầu Welcome nằm trong scope.

Các components có thể tham chiếu đến các components khác tại đầu ra của chúng. Điều này cho phép chúng ta sử dụng cùng một component abstraction cho mọi mức độ chi tiết. Một button, form, dialog, màn hình: trong React apps, chúng đều được hiển thị như là các components.

Thông thường, các React apps mới tạo sẽ có một App component ở tầng cao nhất. Thế nhưng, nếu bạn tích hợp React vào ứng dụng hiện có, bạn có thể bắt đầu bằng cách tiếp cận bottom-up với một component nhỏ như là Button và dần dần đi lên các tầng trên cùng của cây kế thừa giao diện.

Đừng ngại ngần việc tách components thành các components nhỏ hơn. Chia các components ngay từ đầu là một công việc không đơn giản, nhưng bù lại chúng ta sẽ có được một tập hợp các components có thể tái sử dụng trong các ứng dụng lớn hơn khác. Một nguyên tắc quan trọng đó là nếu một phần UI của bạn được sử dụng lại nhiều lần

d. Lifecycle

Lifecycle là vòng đời của 1 component react. Chúng ta có thể tương tác lifecycle của 1 component thông qua các giai đoạn của lifecycle:

Có 3 giai đoạn chính **Mounting**, **Updating**, và **Unmounting**.
Mounting: Đây là giai đoạn đầu tiên của 1 component đó là khi component được khởi tạo

Updating: Đây là giai đoạn sau khi component được khởi tạo. mà được cập nhật mỗi khi state hoặc props thay đổi

Unmounting: Đây là giai đoạn khi component được xóa khỏi DOM

e. Props và State

Props là 1 từ viết ngắn gọn của properties. props là 1 đối tượng, nó lưu trữ các giá trị của các attribute (thuộc tính) của một thẻ (Tag). Là cách mà component có thể nhận được các giá trị của thuộc tính truyền vào từ bên ngoài vào, và là cách mà các component có thể nói chuyện với nhau.

Ví dụ :

```
function Welcome(props) {
    return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="ReactJS" />;
ReactDOM.render(element,document.getElementById('root'));
```

Trong đó, name ở {props.name} là property, Welcome là Component. Mỗi property của Component sẽ tương ứng với 1 attribute của thẻ, giá trị của attribute sẽ được truyền vào property của Component.

- Chúng ta sử dụng props để truyền gửi dữ liệu đến các component.
- Mọi component được gọi là hàm javascript thuần khiết (pure function).
- props tương ứng với tham số của pure function javascript .
- Không thể thay đổi dữ liệu của props .

Dữ liệu của React là một chiều có nghĩa là nó chỉ đi từ component cha xuống component con

State có chức năng tương tự như props cũng lưu trữ thông tin về component, nhưng là lưu trữ dữ liệu động của một component.

State là dữ liệu động , nó cho phép một component theo dõi thông tin thay đổi giữa các kết xuất (render) và làm cho nó có thể tương tác.

State chỉ có thể được sử dụng ở trong một component sinh ra nó.

Nếu dự đoán được một component cần quản lý state, thì nó nên được tạo ở trong một class component chứ không phải là một function component. Vì function components là stateless component có nghĩa là nó không có state để lưu trữ thông tin, thay vào đó nó sử dụng các hook để lưu giữ thông tin.

f. Hook

Hooks chính thức được giới thiệu trong phiên bản React 16.8. Nó cho phép chúng ta sử dụng state và các tính năng khác của React mà không phải dùng đến Class

Điều này có nghĩa là từ phiên bản 16.8 trở đi, chúng ta đã có thể sử dụng state trong stateless (functional) component

Hook là 1 function có thể gắn (hook) vào 1 function component để giúp component có thể quản lý các state hay các tính năng vòng đời của 1 component

Hook không hoạt động được trong class component

Hàm useState nhận đầu vào là giá trị khởi tạo của 1 state và trả ra 1 mảng gồm có 2 phần tử, phần tử đầu tiên là state hiện tại, phần tử thứ 2 là 1 function dùng để update state. Ví dụ:

```
const [isLoading, setLoading] = useState(false);
onClick() {setLoading(true)}
```

Trong đó:

- isLoading là state lưu trữ giá trị hiện tại
- setLoading là hàm để thay đổi giá trị của isLoading
- false là giá trị khởi tạo của state

Hàm useEffect dùng để xử lý các side effects. Nó thay thế các hàm cũ trong lifecycle.

Hàm useEffect nhận đầu vào gồm 2 tham số:

Tham số 1 là 1 hàm sẽ được chạy khi các giá trị trong chuỗi ở hàm thứ 2 thay đổi. trong hàm này nếu bạn trả về 1 hàm thì hàm trả về này sẽ được chạy khi component được update

Tham số 2 là 1 chuỗi các biến hoặc hàm, khi các giá trị này thay đổi hàm ở tham số 1 sẽ chạy. Ví dụ:

```
import { callApi } from './actions'

const App = ({ callApi, data }) => {
  const callApiWhenStart = () => {callApi('some_payload_')}
  useEffect(callApiWhenStart,[])
  return(<div>{data.map(item => {// do something })}</div>
)
  export default App
```

Trong đó callApiWhenStart là hàm sẽ chạy khi 1 trong các phần tử của chuỗi ở tham số 2 thay đổi, tuy nhiên vì tham số 2 ở đây là chuỗi rỗng nên hàm callApiWhenStart sẽ chỉ chạy 1 lần khi component được tạo lần đầu tiên

Ở 1 ví dụ khác:

```
useEffect(() => {
    const subscription = props.source.subscribe();
    return () => {subscription.unsubscribe();};
},
[props.source], // giá trị được subscribe
);
```

Trong đó mỗi khi `props.source` thay đổi giá trị thì hàm ở tham số 1 sẽ chạy lại và trước khi hàm ở tham số 1 sẽ chạy lại hàm `return` sẽ chạy để dọn dẹp những dữ liệu của hàm ở tham số 1 để tránh tồn bộ nhớ

`useMemo` giúp ta kiểm soát việc được render dư thừa của các component con bằng cách truyền vào 1 tham số thứ 2 thì chỉ khi tham số này thay đổi thì `useMemo` mới được thực thi. Ví dụ:

```
const NotUsingMemo = ({ products }) => {
    const soldoutProducts = products.filter(x => x.isSoldout === true);
};

const UsingMemo = ({ products }) => {
    const soldoutProducts = useMemo(
        () => products.filter(x => x.isSoldout === true),
        [products]
    );
};
```

Trong đó ở component `NotUsingMemo` mỗi khi component được render thì `soldoutProducts` sẽ được thực hiện lại. Điều này dẫn đến việc phải tính toán lại cả những trường hợp đầu vào và đầu ra không thay đổi. Nếu việc tính toán là 1 hàm nặng tốn nhiều thời gian vậy sẽ gây ảnh hưởng đến trải nghiệm người dùng

Trong khi đó ở component UsingMemo soldoutProducts sẽ chỉ được tính toán lại nếu như products (tham số truyền vào ở tham số thứ 2 của hàm useMemo) thay đổi. Nó giúp ta loại bỏ được các trường hợp đầu vào không thay đổi, giúp giảm tải việc tính toán useCallback có nhiệm vụ tương tự như useMemo nhưng khác ở chỗ function truyền vào useMemo bắt buộc phải ở trong quá trình render trong khi đối với useCallback đó lại là function callback của 1 event nào đó như là onClick chẳng hạn.

useContext sẽ giải quyết vấn đề là việc trên dòng dữ liệu trong react chỉ đi 1 chiều từ cha xuống con vậy làm sao để truyền props từ con lên cha hoặc giữa các component con với nhau. Bạn có thể tạo một context để thêm những shared state và sau đó component nào cần thì bạn gọi context đó ra xài, không cần phải truyền từ cha xuống con nữa. Nó cũng giống như biến global được chia sẻ và sử dụng ở nhiều nơi, nhưng context được quản lý tốt hơn để maintain code dễ hơn.

Ví dụ sử dụng context:

```
import React from 'react';

const ExampleContext = React.createContext();

const App = () => {

  return (
    <ExampleContext.Provider value={{ color: 'red' }}>
      <div className='App'>
        <ChildComponent />
      </div>
    </ExampleContext.Provider>
  );
};

const ChildComponent = () => {
  const { color } = React.useContext(ExampleContext);
```

```

    return <p style={{ color }}>This text is {color}</p>;
};

export default App;

```

Đầu tiên chúng ta sẽ định nghĩa 1 context:

```
const ExampleContext = React.createContext();
```

Sau đó bao bọc toàn bộ thành phần DOM của component bằng thẻ Provider, đồng thời truyền giá trị mà mình muốn chia sẻ đến các component khác

```

<ExampleContext.Provider value={{ color: 'red' }}>
  <div className="App">
    <ChildComponent />
  </div>
</ExampleContext.Provider>

```

Như vậy là chúng ta đã có thể sử dụng context đó trong các component con thông qua useContext:

```

const ChildComponent = () => {
  const { color } = React.useContext(ExampleContext);

  return <p style={{ color }}>This text is {color}</p>;
};

```

Ngoài ra value của hàm Provider còn có thể chứa các function

Ngoài ra chúng ta còn rất nhiều cái hook khác được định nghĩa sẵn bởi React mà các bạn có thể đọc thêm ở trên trang chủ của React hoặc chúng ta có thể tự viết hook cho mình trong trường hợp cái logic đó lặp lại ở các component khác nhau. Các hàm hook luôn được bắt đầu với use (vd: useFetch). Ví dụ:

```
import { useState, useEffect } from "react";

const useFetch = (url) => {
    const [data, setData] = useState(null);

    useEffect(() => {
        fetch(url)
            .then((res) => res.json())
            .then((data) => setData(data));
    }, [url]);

    return [data];
};

export default useFetch;
```

Trong ví dụ trên ta tạo sử dụng hook useFetch để lấy dữ liệu từ data. Hàm useFetch nhận vào url và trả về data và sẽ gọi api mỗi khi url thay đổi (ứng dụng useEffect đã giới thiệu ở trên). Sau đó chúng ta có thể sử dụng useFetch như các hook khác

```

import useFetch from "./useFetch";

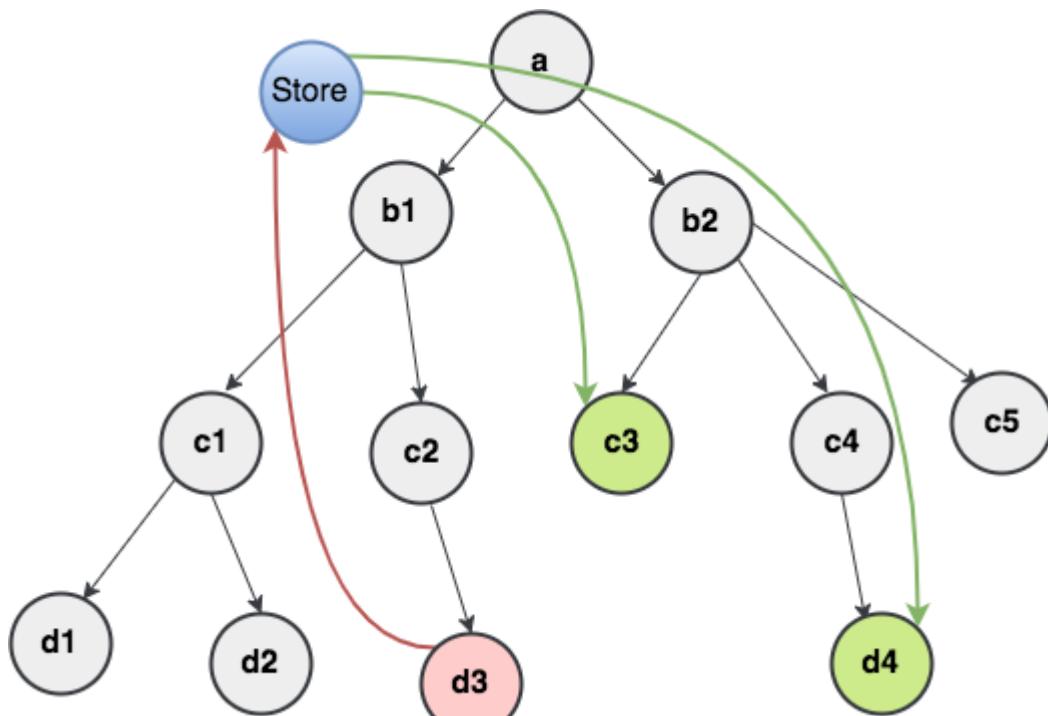
const Home = () => {
    const [data] = useFetch("https://jsonplaceholder.typicode.com/todos");
    return (
        <>
        {data &&
            data.map((item) => {
                return <p key={item.id}>{item.title}</p>;
            })}
        </>
    );
};

```

8) *Redux*

Redux là 1 thư viện Javascript để quản lý state của ứng dụng, thường được sử dụng với javascript framework như React.

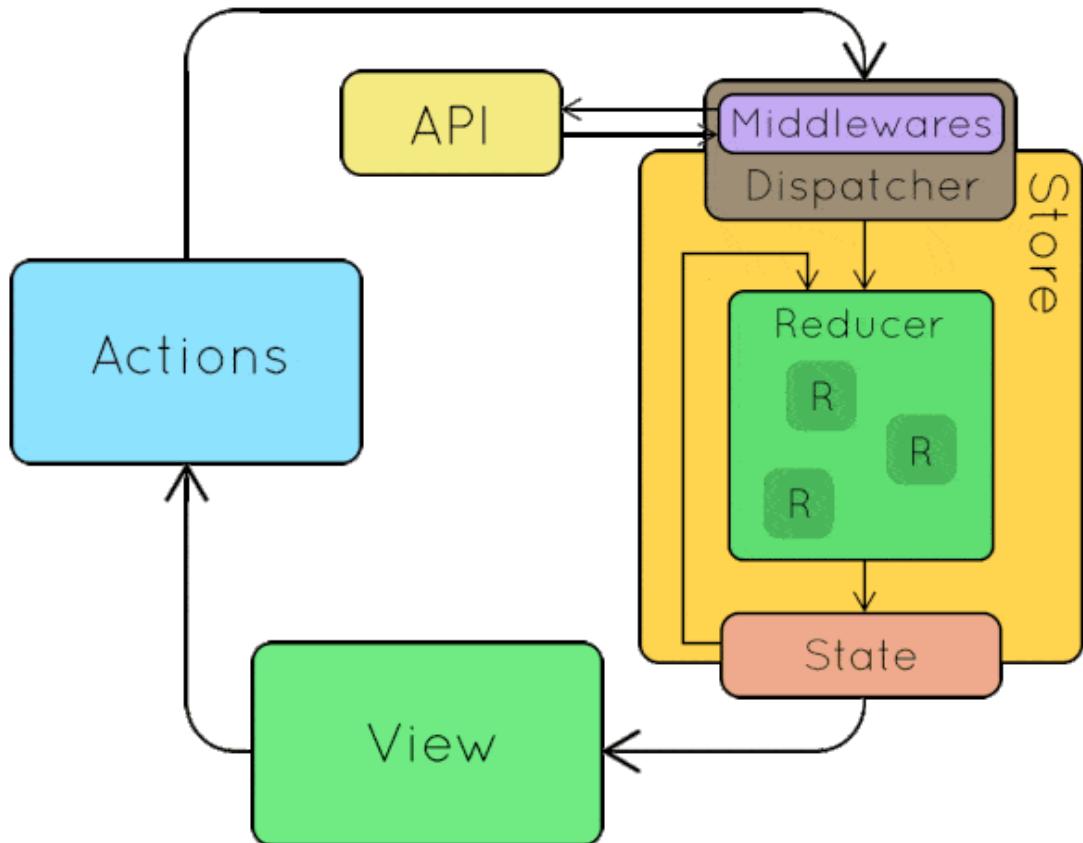
Như đã đề cập bên trên dữ liệu của react chỉ truyền 1 chiều từ cha cho con. Việc truyền dữ liệu giữa con lên cha hoặc giữ các component đồng cấp rất phức tạp. Chúng ta có thể dùng useContext để cập ở trên tuy nhiên useContext sẽ render lại tất cả các component nằm trong provider khi value thay đổi điều này ồn với các ứng dụng nhỏ tuy nhiên với các ứng dụng lớn việc này sẽ giảm hiện xuất của trang web. Do đó redux là 1 giải pháp quản lý state được nhiều người sử dụng khi sử dụng React. Trái với useContext, khi sử dụng Redux các component chỉ render lại những component thay đổi props hoặc state tuy nhiên việc setup Redux đòi hỏi nhiều công sức hơn là useContext.



Hình 36: Mô hình hoạt động của Redux

Cách mà Redux hoạt động là khá đơn giản. Nó có 1 store lưu trữ toàn bộ state của app. Mỗi component có thể access trực tiếp đến state được lưu trữ thay vì phải truyền props từ component này đến component khác.

Cơ chế hoạt động của nó được tóm gọn trong 1 sơ đồ đơn giản:



Hình 37: Sơ đồ hoạt động của Redux

Khi người dùng tương tác với UI sẽ tạo ra 1 action action này sẽ được dispatcher gửi đến reducer (các action bắt đồng bộ ví dụ như gọi api sẽ được xử lý bởi các middleware tại bước này). Reducer sẽ nhận thông tin từ action và state hiện tại và xử lý sao đó trả về một state mới. Khi state thay đổi UI cũng sẽ cập nhật theo.

Thành phần chính của Redux gồm: Actions, Store, Reducers.

a. Action

Actions đơn giản là các events. Chúng là cách mà chúng ta send data từ app đến Redux store. Những data này có thể là từ sự tương tác của user vs app, API calls hoặc cũng có thể là từ form submission.

Actions được gửi bằng cách sử dụng store.dispatch() method, chúng phải có một type property để biểu lộ loại action để thực hiện. Chúng cũng phải có một payload chứa thông tin. Actions được tạo thông qua một action creator. Ví dụ:

```
const setLoginStatus = (name, password) => {
  return {
    type: "LOGIN",
    payload: {
      username: "foo",
      password: "bar"
    }
  }
}
```

Trong đó setLoginStatus là tên của action (function) sẽ được gọi trong hàm dispatch. setLoginStatus trả về 1 object gồm type để reducer nhận biết phải thực hiện hành động nào và payload để truyền data xuống reducer

b. Reducers

Reducers là các function nguyên thủy chúng lấy state hiện tại của app, thực hiện một action và trả về một state mới. Những states này được lưu như những objects và chúng định rõ cách state của một ứng dụng thay đổi trong việc phản hồi một action được gửi đến store. 1 reducer tương đương với 1 state nhưng kèm theo các mô tả state sẽ thay đổi như thế nào khi các action khác nhau được gọi. Ví dụ:

```

export default function counterApp (state = initialState, action) {

  switch (action.type) {

    case INCREASE:

      return {
        increase: ++state.increase,
        decrease: state.decrease
      }

    case DECREASE:

      return {
        increase: state.increase,
        decrease: ++state.decrease
      }

    default:

      return state
  }
}

```

Trong đó counterApp là tên của reducer. Trong 1 ứng dụng có thể có nhiều reducer. Mỗi reducer có 1 tên khác nhau. Reducer nhận vào 2 tham số là state với giá trị default là initialState (giá trị khởi tạo của state) và 1 action

Reducer sẽ dựa theo action type để thay đổi state với payload (nếu có) và trả về state mới. Lưu ý state mới phải là object mới nếu không Redux sẽ không thể cập nhật lại UI

c. Store

Store là nơi lưu trạng thái của ứng dụng và chỉ có duy nhất một Store trong bất kỳ một chương trình ứng dụng Redux nào. Nhiệm vụ của store chính là quản lý, access các state được lưu, update state thông qua dispatch, cho phép truy cập state thông qua Getstate và đăng ký hoặc hủy đăng ký các listeners thông qua helper methods.

d. Sử dụng Redux

Sử dụng useSelector của react-redux để lấy state counter từ store.

Sử dụng useDispatch để trả về function dispatch, truyền increment và decrement vào dispatch để gọi 2 action này

e. Redux-toolkit

Như đã trình bày ở trên việc setup redux khá phức tạp và tốn thời gian. Redux-toolkit là một package được sinh ra nhằm giải quyết phần lớn những vấn đề kể trên, được phát triển bởi chính chủ reduxjs team, giúp chúng ta viết code redux nhanh gọn, hoàn chỉnh theo một quy chuẩn thống nhất.

Hàm configureStore là 1 hàm của redux-toolkit giúp chúng ta đơn giản việc setup store. Như ví dụ ở trên, chúng ta khởi tạo một store bằng hàm createStore của redux với tham số nhận vào là một reducer, configureStore sẽ làm điều tương tự như vậy, chúng ta sẽ khởi tạo store theo phương pháp sau

```
import { configureStore } from '@reduxjs/toolkit';

const store = configureStore({
  reducer: rootReducer
})
```

Về cú pháp thì không khác nhau nhiều. Tuy nhiên thay vì chỉ khởi tạo một store đơn thuần, configureStore sẽ mặc định thiết lập cho phép sử dụng redux devtool để debug và theo dõi quá trình cập nhật state cũng như thiết lập sẵn một số middleware giúp thực hiện các action bất đồng bộ.

createSlice là hàm của redux-toolkit giúp chúng ta định nghĩa 1 reducer và các action liên quan đến nó. Ví dụ:

```
const initialState = {
  value: 0,
}

export const counterSlice = createSlice({
  name: 'counter',
  initialState,
  reducers: {
    increment: (state) => {
      state.value += 1
    },
    decrement: (state) => {
      state.value -= 1
    },
    incrementByAmount: (state, action) => {
      state.value += action.payload
    },
  },
})

export const { increment, decrement, incrementByAmount } = counterSlice.actions
```

```
export default counterSlice.reducer
```

createSlice nhận vào 1 object gồm:

- name: Tên của reducer
- initialState: state khởi tạo của reducer
- reducer: là 1 object định nghĩa các hàm quy định reducer sẽ cập nhật state như thế nào. Bên cạnh đó như đã nhắc ở trên reducer phải trả về 1 object mới tuy nhiên redux-toolkit đã đơn giản hóa việc đó chúng ta có thể thực hiện tính toán ngay trên object cũ

createSlide trả về 1 object chứa actions và reducer. Chúng ta có thể export các action và reducer để sử dụng. Như vậy chỉ 1 hàm đơn giản đã giúp chúng ta tiết kiệm rất nhiều thời gian bằng cách gộp bước tạo action và tạo reducer vào làm 1 .

9) *Typecript*

TypeScript là một phiên bản cao hơn của JavaScript, được thiết kế để xây dựng các ứng dụng lớn và phức tạp. Nó kế thừa nhiều khái niệm từ Java và C#, TypeScript là ngôn ngữ tĩnh (Static typed) có nghĩa là nó nghiêm ngặt và có trật tự trái ngược với free-type. Nó còn được bổ sung thêm lớp hướng đối tượng mà điều này không có ở Javascript. Với TypeScript, ta có thể bê nguyên xi code JavaScript vào trong cùng một file và chạy cùng nhau bình thường, bởi vì TypeScript duy trì cú pháp của JavaScript và mở rộng nó bằng một loạt tính năng mới. Nhờ đó mà hiệu năng làm việc được tăng lên đáng kể.

a. Ưu điểm của Typescript:

Dễ dàng hơn trong phát triển các dự án lớn, được hỗ trợ bởi các Javascript Framework lớn.

Hầu hết các cú pháp hướng đối tượng đều được hỗ trợ bởi Typescript như kế thừa, đóng gói, constructor, abstract, interface, implement, override...v.v

Cách tổ chức code rõ ràng hơn, hỗ trợ cơ chế giúp kiến trúc hệ thống code hướng module, hỗ trợ namespace, giúp xây dựng các hệ thống lớn nơi mà nhiều lập trình viên có thể làm việc cùng nhau một cách dễ dàng hơn.

Hỗ trợ các tính năng mới nhất của Javascript. TypeScript luôn đảm bảo việc sử dụng đầy đủ các kỹ thuật mới nhất của Javascript, ví dụ như version hiện tại là ECMAScript 2015 (ES6).

Một lợi thế của Typescript nữa là mã nguồn mở vì vậy nó miễn phí và có cộng đồng hỗ trợ rất lớn.

Với static typing, code viết bằng TypeScript dễ dự đoán hơn, và dễ debug hơn.

b. Basic Types:

Trong TypeScript chia ra làm 7 loại cơ bản, bao gồm: boolean, number, string, array, enum, any, void.

- Boolean:

```
var isDone: boolean = true;
```

- String:

```
var name: boolean = "nguyen thi A";
```

- Number:

```
var height: number = 8;
```

- Array : có 2 kiểu khai báo tương đương với nhau trong TypeScript

```
1: var list: boolean[] = [true, false];
```

```
2: var isDone: Array<boolean> = [true, false];
```

- Enum: khi khai báo enum một cách thông thường các phần tử sẽ được đánh số từ 0 và tăng dần

```
enum Color{Red, Green, Blue};  
  
var c: Color = Color.Green  
  
var colorName = Color[1] // kết quả sẽ là Green
```

- Khi muốn phần tử đầu tiên là 1 chứ không phải là 0 như mặc định thì cần khai báo như sau:

```
enum Color{Red = 1, Green, Blue};  
  
var c: Color = Color.Green  
  
var colorName = Color[1] // kết quả sẽ là Red
```

- o Any: Any là một kiểu mà bạn có thể gán bất kỳ kiểu nào cho nó.

```

var notSure: any = 4;

notSure = "maybe a string instead";

notSure = false; // khai báo này hoàn toàn được chấp nhận.

// nếu notSure ban đầu khai báo và number thì
// tại đây chắc chắn sẽ có lỗi

var list:any[] = [1, true, "free"]; // nếu sử dụng var list:number[] thì
// tất cả các phần tử trong list sẽ phải là kiểu number

list[1] = 100;

```

- o Void: Cũng giống như any nhưng void được sử dụng là đầu ra của hàm.

```

function warnUser(): void {
    alert("This is my warning message");
}

```

c. Function

TypeScript có khai báo function giống như JavaScript. Nhưng khi khai báo function TypeScript còn hỗ trợ việc khai báo với các kiểu trả ra của function và cũng như kiểu đầu vào của dữ liệu. Không những thế khi sử dụng TypeScript ta có thể khai báo giá trị mặc định của đầu vào ngay khi khai báo function, điều mà JavaScript không có. Không dừng lại ở đó TypeScript còn hỗ trợ việc bỏ qua nhập một hoặc vài đầu vào.

Ví dụ:

```
function buildName(firstName: string, lastName?: string):string {
    if (lastName) return firstName + " " + lastName;
    else return firstName; }
```

Trong đó

- “firstName: string”: Khai báo kiểu dữ liệu string cho firstName
- “lastName?: string”: Khai báo kiểu dữ liệu string lastName tuy nhiên lastName có thể là undefined
- “:string”: là kiểu trả về của function
 - d. Type aliases

Trong TypeScript, có rất nhiều types cơ bản như là number, string,... Ngoài ra, trong TypeScript chúng ta có các types nâng cao và các types nâng cao này được gọi là type aliases. Với type aliases, chúng ta có thể tạo tên mới cho một type nhưng chúng ta không define một type mới. Từ khóa type trong TypeScript là một cách cung cấp type aliases cho các variables, objects và functions của chúng ta. Các type aliases này mô tả dữ liệu của chúng ta trông như thế nào. Để mô tả loại dữ liệu của mình trông như thế nào chúng ta sử dụng các type cơ bản (string, number,...) hoặc bằng cách tạo các type tùy chỉnh của chúng ta.

Ví dụ:

```
// type của Year là một number
type Year = number;

// Biến currentYear là thuộc type Year và phải là number
let currentYear: Year = 2021;
```

```
// custom Person object type
type Person = {
    name: String;
    gender: String;
};
```

e. Interface

Interface trong typescript cho phép bạn định nghĩa thuộc tính là gì và phương thức là gì mà đối tượng cần để được implement. Nếu đối tượng tuân thủ đúng khuôn mẫu interface thì đối tượng đã implement interface ấy sẽ được thực thi đúng. Nếu interface không được thực thi đúng thì typescript sẽ phát sinh lỗi ngay lập tức.

```
interface Person {
    name: string
    age: number
}

const viblo: Person = {
    name: "Viblo",
    age: 31
};
```

f. Types vs Interfaces

Có thể merge interfaces, types thì không

```
interface Person {  
    name: string  
}  
  
interface Person {  
    age: number  
}  
  
const viblo: Person = {  
    name: "Viblo",  
    age: 31  
}
```

Type aliases có thể sử dụng computed properties:

- Từ khóa `in` có thể được sử dụng để iterate tất cả các item bên trong một tập hợp keys. Chúng ta có thể sử dụng tính năng này để tạo mapped types.

Ví dụ sử dụng type aliases

```
type keys = "firstname" | "lastname"
```

```
type Person = {
    [key in keys]: string
}
```

```
const viblo: Person = {
    firstname: "Viblo",
    lastname: "blog"
}
```

Extend và implements:

Trong TypeScript, chúng ta dễ dàng extends và implements interfaces. Nhưng không thể với types. Với interface để kế thừa chúng ta sử dụng extends:

```
interface Animal {
    name: string
}

interface Bear extends Animal {
    honey: boolean
}
```

g. Intersection

Intersection cho phép chúng ta kết hợp nhiều types thành một types duy nhất. Để tạo một intersection types chúng ta dùng &:

```
type Name = {
    name: string
```

```

};

type Age = {
    age: number
};

type Person = Name & Age;

```

Chúng ta cũng có thể làm điều này với interface:

```

interface Name {
    name: "string"
};

interface Age {
    age: number
};

type Person = Name & Age;

```

h. Generics

Generic type trong TypeScript cho phép bạn viết các function, class và interface có thể tái sử dụng và tổng quát hóa. Trong hướng dẫn này, bạn sẽ tập trung vào việc tạo các hàm generic (generic function).

Giả sử ta gặp trường hợp chúng ta phải định nghĩa 1 function trả về ngẫu nhiên 1 phần tử trong 1 mảng. Nếu chúng ta biết mảng vào sẽ là mảng number thì chúng ta có function:

```

function getRandomNumberElement(items: number[]): number {
    let randomIndex = Math.floor(Math.random() * items.length);
    return items[randomIndex];
}

```

Tuy nhiên sẽ ra sao nếu chuỗi vào có thể là mảng number hoặc là mảng string. Trong trường hợp này chúng ta có thể tạo thêm 1 hàm

```
function getRandomStringElement(items: string[]): string {
    let randomIndex = Math.floor(Math.random() * items.length);
    return items[randomIndex];
}
```

Vậy nếu chuỗi vào có thể là bất cứ type nào thì sao? Chúng ta không thể tạo ra tất cả các hàm cho mỗi type được. Vậy phải làm sao đây? Bạn có thể dùng type any tuy nhiên nếu như vậy bạn sẽ không xác định được type trả về hay nói cách khác là nó không an toàn. Do đó typescript cho chúng ta 1 giải pháp cho những trường hợp như vậy đó là generic type.

Chúng ta có thể định nghĩa 1 hàm generic như sau:

```
function getRandomElement<T>(items: T[]): T {
    let randomIndex = Math.floor(Math.random() * items.length);
    return items[randomIndex];
}
```

Hàm này sử dụng biến kiểu T. Kiểu T cho phép bạn chỉ định kiểu dữ liệu tại thời điểm gọi hàm. Ngoài ra, hàm sử dụng biến kiểu T làm kiểu trả về của nó.

Hàm getRandomElement() là hàm generic bởi vì nó có thể làm việc với bất kỳ kiểu dữ liệu nào, ví dụ: chuỗi, số, đối tượng, ...

Theo quy ước, chúng ta sử dụng ký tự T làm kiểu dữ liệu của biến. Tuy nhiên, bạn có thể sử dụng các chữ khác như A, B C, ...

Sau đây là cách sử dụng hàm getRandomElement() với một mảng số:

```
let numbers = [1, 5, 7, 4, 2, 9];
let randomEle = getRandomElement<number>(numbers);
```

```
console.log(randomEle);
```

Ví dụ này chỉ định rõ kiểu number cho T khi gọi hàm getRandomElement().

Trong thực tế, bạn sẽ sử dụng kiểu suy luận cho đối số. Có nghĩa là bạn để trình biên dịch TypeScript tự xác định kiểu dữ liệu của T dựa trên loại đối số mà bạn truyền vào, như thế này:

```
let numbers = [1, 5, 7, 4, 2, 9];
let randomEle = getRandomElement(numbers);
console.log(randomEle);
```

Trong ví dụ này, chúng tôi đã không chỉ định kiểu number một cách rõ ràng khi gọi hàm getRandomElement(). Trình biên dịch sẽ xác định kiểu dữ liệu của T dựa vào kiểu dữ liệu của tham số truyền vào.

Bây giờ, hàm getRandomElement() cũng là kiểu an toàn. Ví dụ: nếu bạn gán giá trị trả về cho một biến kiểu chuỗi, bạn sẽ gặp lỗi:

```
let numbers = [1, 5, 7, 4, 2, 9];
let returnElem: string;
returnElem = getRandomElement(numbers); // compiler error
```

Hàm generic có nhiều kiểu dữ liệu trong TypeScript

Phần sau minh họa cách tạo một hàm generic với hai biến kiểu U và V:

```
function merge<U, V>(obj1: U, obj2: V) {
  return {
    ...obj1,
    ...obj2
  };
}
```

Hàm merge() kết hợp hai đối tượng với kiểu U và V. Nó kết hợp các thuộc tính của hai đối tượng thành một đối tượng duy nhất.

Kiểu suy luận suy ra giá trị trả về của hàm merge() là kiểu giao của U và V, là U & V.

Sau đây minh họa cách sử dụng hàm merge() để hợp nhất hai đối tượng:

```
let result = merge(
  { name: 'John' },
  { jobTitle: 'Frontend Developer' }
);
console.log(result);
```

Đầu ra:

```
{ name: 'John', jobTitle: 'Frontend Developer' }
```

Tuy nhiên trong ví dụ trên Hàm merge() mong đợi đầu vào là hai đối tượng. Tuy nhiên, nó không ngăn bạn truyền vào tham số như thế này:

```
let person = merge(
  { name: 'John' },
  25
);
console.log(person);
```

Đầu ra:

```
{ name: 'John' }
```

Code vẫn chạy mà không có một lỗi nào, tuy nhiên đó không phải là điều chúng ta mong muốn. Chúng ta muốn đầu vào của hàm phải là 2 object vì vậy chúng ta có thể ràng buộc generic type là là type mở rộng của object với từ khóa extends.

```
function merge<U extends object, V extends object>(obj1: U, obj2: V) {
    return {
        ...obj1,
        ...obj2
    };
}
```

Điều này dẫn đến đoạn code dưới đây lỗi và đó là điều chúng ta mong muốn.

```
let person = merge(
    { name: 'John' },
    25
);
```

TypeScript cho phép bạn khai báo một tham số có kiểu dữ liệu bị ràng buộc bởi kiểu dữ liệu của một tham số khác.

Hàm prop() sau chấp nhận hai tham số là một đối tượng và một tên thuộc tính. Nó trả về giá trị của thuộc tính.

```
function prop<T, K>(obj: T, key: K) {
    return obj[key];
}
```

Trình biên dịch gặp lỗi sau:

```
Type 'K' cannot be used to index type 'T'.
```

Để khắc phục lỗi này, bạn thêm ràng buộc cho kiểu dữ liệu K để đảm bảo rằng nó là một khóa của kiểu dữ liệu T như sau:

```
function prop<T, K extends keyof T>(obj: T, key: K) {return obj[key];}
```

Nếu bạn truyền vào hàm prop một tên thuộc tính tồn tại trên obj, trình biên dịch sẽ không báo lỗi. Ví dụ:

```
let str = prop({ name: 'John' }, 'name');
console.log(str);
```

Đầu ra:

```
John
```

Tuy nhiên, nếu bạn truyền một khóa không tồn tại trên đối số đầu tiên, trình biên dịch sẽ phát ra lỗi:

```
let str = prop({ name: 'John' }, 'age');
```

Lỗi:

```
Argument of type '"age"' is not assignable to parameter of type '"name"'.
```

10) Antd

a. Antd là gì

Ant là tập hợp các components của React được xây dựng theo chuẩn thiết kế của Ant UED Team. Tương tự như chuẩn Material Design, Ant cung cấp hầu hết các component thông dụng trong ứng dụng web hiện đại, như Layout, Button, Icon, DatePicket, v.v... Bên cạnh đó Ant cũng có những component riêng thú vị, như LocaleProvider cho phép bạn thay đổi ngôn ngữ trên toàn ứng dụng.

Có thể coi Ant Design cho React là tập hợp của hầu hết các thư viện về React. Nó đáp ứng được hầu hết các yêu cầu của project của bạn mà bạn không phải cài thêm bất cứ thư viện nào nữa. Dưới đây là danh sách các component mà nó cung cấp:

- General: Button, Icon
- Layout: Grid, Layout

- Navigation: Affix, Breadcrumb, Dropdown, Menu, Pagination, Steps
- Data Entry: AutoComplete, Checkbox, Cascader, DatePicker, Form, InputNumber, Input, Mention, Rate, Radio, Switch, Slider, Select, TreeSelect, Transfer, TimePicker, Upload
- Data Display: Avatar, Badge, Collapse, Carousel, Card, Calendar, List, Popover, Tree, Tooltip, Timeline, Tag, Tabs, Table
- Feedback: Alert, Drawer, Modal, Message, Notification, Progress, Popconfirm, Spin, Skeleton
- Other: Anchor, BackTop, Divider, LocaleProvider

b. Cách sử dụng

Import css được antd sử dụng (chỉ cần import 1 lần duy nhất ở file index.js):

```
import 'antd/dist/antd.css';
```

Import component mà ta muốn sử dụng:

```
import { Button } from 'antd';
```

Sử dụng những các component khác

```
<Button>Default Button</Button>
```

Đọc thêm về tất cả các component được thư viện cung cấp tại [trang chủ của antd](#).

3. Nội dung của chương này

Giới thiệu về đề tài về xây dựng trang ứng dụng đăng tin tuyển dụng cũng như ứng tuyển dành cho sinh viên trường Đại học Tôn Đức Thắng

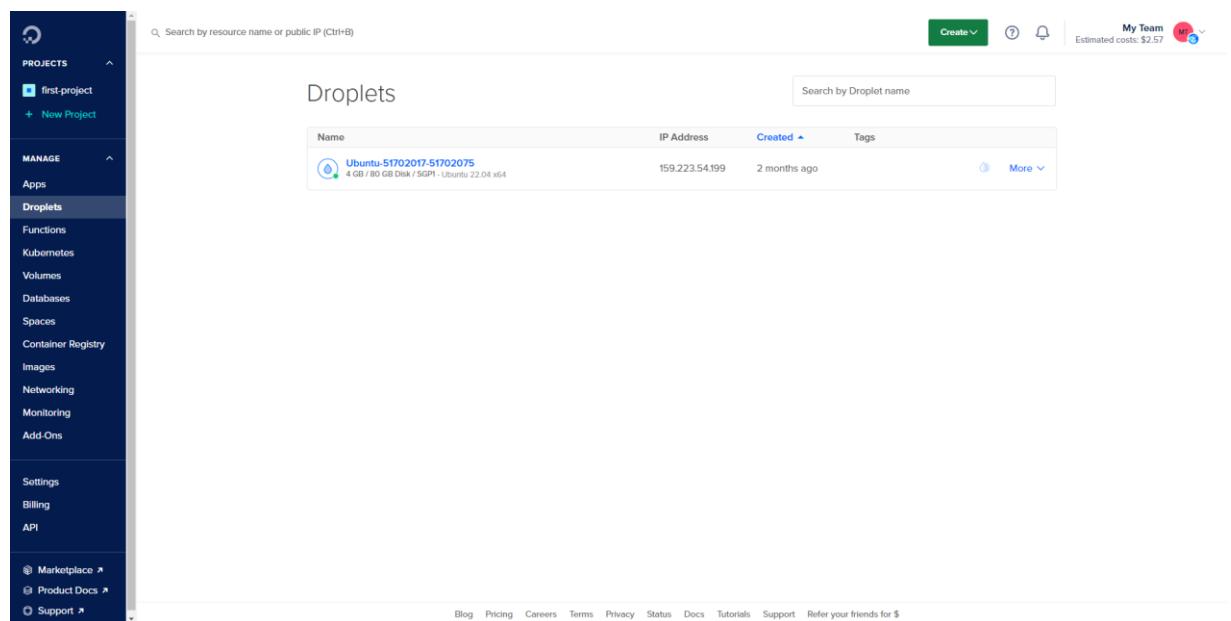
Giới thiệu về các công nghệ sử dụng trong quá trình xây dựng trang ứng dụng này

CHƯƠNG 2 – QUÁ TRÌNH THIẾT LẬP CÁC CÔNG CỤ

1. Quá trình thiết lập server

Để đưa source code lên server ta cần một server ở đây chúng em thuê 1 con server chạy hệ điều hành Ubuntu với 4GB ram và 80GB bộ nhớ của Digitalocean với chi phí duy trì hàng tháng

Quá trình thuê thì sẽ đưa thẻ visa của bản thân vào tài khoản để thuê server và mỗi tháng thì digitalocen sẽ trừ khoản tiền đó



Hình 38: Hình ảnh thuê server

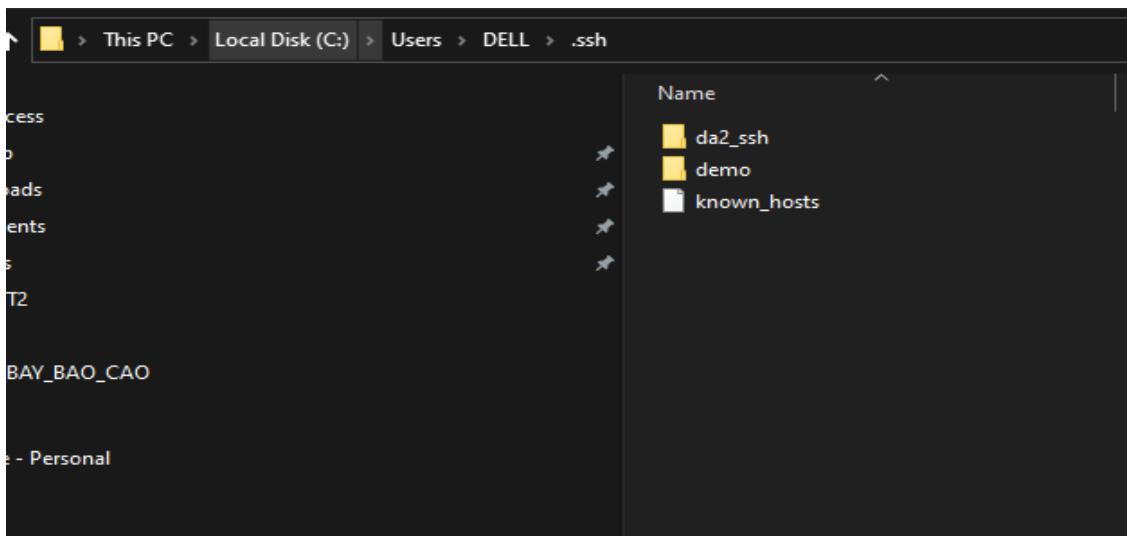
Khi đã có server thì bước tiếp theo thì sẽ cấu hình cho việc kết nối với server đó qua một số công cụ như cmd , Window Terminal,, và các bước như sau:

Bước 1 : ta sẽ sử dụng ssh-keygen để tạo ra 1 chuỗi các ký tự ngẫu nhiên được gọi là key để sau khi thiết lập mỗi lần muốn kết nối với server ta chỉ cần gõ lệnh ssh@root + tên server và nhập mật khẩu của server thì lúc này ta sẽ kết nối được với server của mình vừa thuê

```
PS D:\Workspace\DACNTT2\DA2> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:/Users/DELL/.ssh/id_rsa): |
```

Hình 39: Gõ ssh-keygen để tạo ra bộ mã cho riêng mình

Bước 2 ta vào thư mục C:/Users/(Tên của user)/.ssh và tạo ra 1 thư mục ví dụ ở đây máy cá nhân này có thư mục (demo) để mô phỏng lại quá trình tạo ssh-keygen và da2_ssh đã tạo trong quá trình thực hiện để tài này

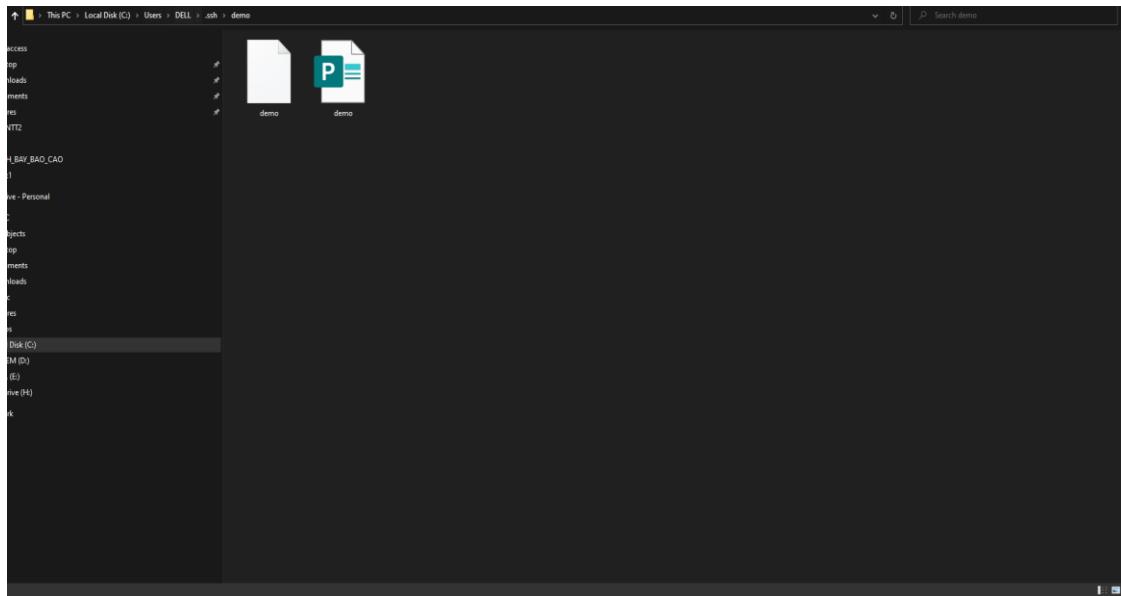


Hình 40: Thư mục .ssh chứa ssh-keygen

Tiếp tục ta ở Window Terminal ta gõ lệnh để với thư mục demo ta gõ lệnh chuyển trả noi tạo file chứa public key và private key

```
PS D:\Workspace\DACNTT2\DA2> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:/Users/DELL/.ssh/id_rsa): C:/Users/DELL/.ssh/demo/demo
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:/Users/DELL/.ssh/demo/demo.
Your public key has been saved in C:/Users/DELL/.ssh/demo/demo.pub.
The key fingerprint is:
SHA256:p44unkCBt9axRF455Ni5r9uV7yuc2mfDrD2YyZC6b6Q dell@DESKTOP-TDOQ9QP
The key's randomart image is:
+---[RSA 3072]---+
| |
| . |
| = o |
| o B |
| . o . OS.. |
| o o + . +o . |
| + * + =.+oB |
| . o * E+ooOb |
| . o**=..o**+ |
+---[SHA256]---+
PS D:\Workspace\DACNTT2\DA2>
```

Hình 41: Hình ảnh tạo ssh key trong thư mục demo



Hình 42: Hình ảnh thư mục demo chứa ssh key ta vừa tạo với demo

- Trong thư mục ta vừa tạo thì sẽ có 2 tập tin bao gồm 1 file demo thường và 1 file demo có đuôi là pub tức nghĩa là public key còn file còn lại là secret key
- Sau khi khởi tạo ssh key ở máy cá nhân ta truy cập với server thông qua ssh bằng dòng lệnh

```

PS D:\Workspace\DACNTT2\DA2> ssh root@159.223.54.199
root@159.223.54.199's password:
Permission denied, please try again.
root@159.223.54.199's password:
Permission denied, please try again.
root@159.223.54.199's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-41-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Sun Dec  4 08:36:48 UTC 2022

System load:                      0.0
Usage of /:                         13.6% of 77.35GB
Memory usage:                      54%
Swap usage:                        0%
Processes:                          130
Users logged in:                   0
IPv4 address for br-dae8ae5879b: 172.19.0.1
IPv4 address for docker0:          172.17.0.1
IPv4 address for eth0:             159.223.54.199
IPv4 address for eth0:             10.15.0.5
IPv4 address for eth1:             10.104.0.2

22 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

*** System restart required ***
Last login: Thu Dec  1 12:58:36 2022 from 14.169.46.238
root@Ubuntu-51702017-51702075:~#

```

Hình 43: Kết nối với server thuê được bằng ssh

- Ta sẽ sử dụng ssh root@(IP) mà nhà cung cấp dịch vụ vps cung cấp cho ta để kết nối và nhập mật khẩu của server để truy cập vào server
- Khi này ta sẽ có quyền thao tác với server lúc này ta sẽ add ssh được sinh ra ở máy cá nhân

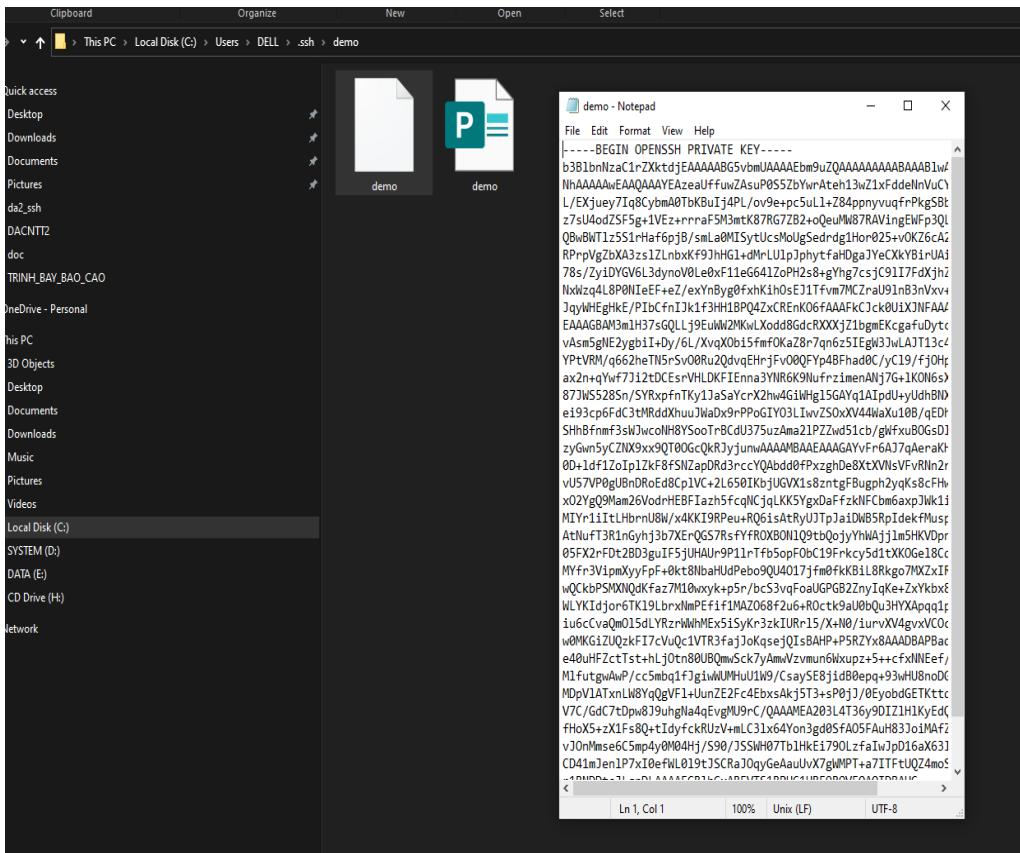
```
*** System restart required ***
Last login: Thu Dec  1 12:58:36 2022 from 14.169.46.238
root@Ubuntu-51702017-51702075:~# cd .ssh
root@Ubuntu-51702017-51702075:~/ssh# ls
authorized_keys  known_hosts
root@Ubuntu-51702017-51702075:~/ssh# vi authorized_keys
root@Ubuntu-51702017-51702075:~/ssh# |
```

Hình 44: Thêm ssh key vào máy chủ

- Khi này ta thực hiện chuyển thư mục bằng `cd .ssh` và `ls` để check xem đã có file `authorized_keys` hay chưa nên chưa ta sẽ tạo file này bằng sử dụng câu lệnh vừa `cd` vừa tạo bằng `nano` “`cd .ssh && sudo nano authorized_keys`” để chuyển đến thư mục `.ssh` và có quyền như root sử dụng lệnh viết `nano` trên tập tin `authorized_keys`
- Ở đây máy chủ đã có tập tin `authorized_keys` nên em sẽ dùng `vim (vi)` để xem nội dung thư mục

Hình 45: Nội dung tập tin authorized_keys

- Nếu là lần đầu thiết lập tập tin này không có gì và ta sẽ thực hiện thao tác nhân nút I trên bàn phím để vào chế độ viết và ta quay lại máy cá nhân thư mục .ssh ta tạo có 2 file public và file thường



Hình 46: Ta sẽ copy key trong file private key

Ta sẽ past private key này vào file ở server thì đây là demo sẽ có nội dung khác với key đã được thêm vào server trước đó

Sau khi thêm private của máy cá nhân vào server thì ta nhấn :x để thoát và lưu lại nội dung đã ghi với vim

Sau khi đã lưu lại private key ta tiến hành update và cài đặt một số thư viện với các dòng lệnh như sau:

- Sudo apt update : Để Ubuntu sẽ tiến hành việc tải các tài nguyên cần thiết để cấu hình server
- Sudo apt upgrade : Để nâng cấp phiên bản nếu tài nguyên sau khi update còn quá cũ

- Sudo apt-get install ca-certificates : Cài đặt tài nguyên này để cho việc chứng thực số cho server này có thể hoạt động trong môi trường mạng
- Sudo apt-get install curl : Để cài đặt 1 công cụ (tool) có thể cài đặt một số thư viện khác ở đây để tải 1 số thư viện
- Sudo apt-get install gnupg : Để cài đặt một phần mềm bảo mật cho server
- Sudo apt-get install lsd-realase : Để cài đặt phiên bản linux tiêu chuẩn nhất giành cho máy chủ nếu phiên bản linux của máy chủ quá cũ
- Ta sử dụng 1 câu lệnh như sau :

Curl -fsSL <https://download.docker.com/linux/ubuntu/gpg> | sudo gpg --dearmor - /usr/share/keyrings/docker-archive-keyring.gpg Để thêm khóa mã hóa cho docker (GPG – GNU Private Guard)

- sudo apt-get install docker-ce docker-ce-cli containerd.io : Để cài đặt các tài nguyên của Docker

Để sử dụng database mà cụ thể là SQL SERVER (2019) bằng docker thì Microsoft đã có images của sql ở docker hub và ta sẽ thực hiện lệnh docker pull để lấy images sql đó để chạy ở server ở trang chủ của docker : https://hub.docker.com/_/microsoft-mssql-server

Ta sẽ thực hiện các lệnh như sau :

- sudo docker pull mcr.microsoft.com/mssql/server:phiên bản-latest thì ở đây máy chủ này đang sử dụng phiên bản 2019 thì sẽ thay phiên bản bằng 2019
- Tiếp theo ta sẽ chạy 1 lệnh nữa để chạy cái images đó thành 1 container trên docker
 - o docker run -e "ACCEPT_EULA =Y" -e "SA_PASSWORD = MatKhau" -p 1433:1433 --name ten --hostname ten -d <https://mcr.microsoft.com/mssql/server:2019-latest>

Để kiểm tra container có hoạt động không ta gõ lệnh docker ps để xem máy chủ có những container nào đang hoạt động

```
root@Ubuntu-51702817-51702875:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
292c7f3565fd7      da2_frontend       "/docker-entrypoint..."   2 days ago        Up 2 days          0.0.0.0:80->80/tcp, :::80->80/tcp           frontda2
8f64ff5913716      da2_cdn           "/docker-entrypoint..."   2 days ago        Up 2 days          0.0.0.0:3002->3002/tcp, :::3002->3002/tcp           cdnda2
7b7e9c792f56       da2_backend        "/docker-entrypoint..."   2 days ago        Up 2 days          0.0.0.0:4000->4000/tcp, :::4000->4000/tcp           backendda2
922ba2001dcc       mcr.microsoft.com/mssql/server:2019-latest  "/opt/mssql/bin/perm..."   4 weeks ago       Up 4 weeks         0.0.0.0:1433->1433/tcp, :::1433->1433/tcp           da2
```

Hình 47: Hình ảnh thể hiện máy chủ có các container đang hoạt động

2. Cách cài đặt gitlab CI/CD vào source code

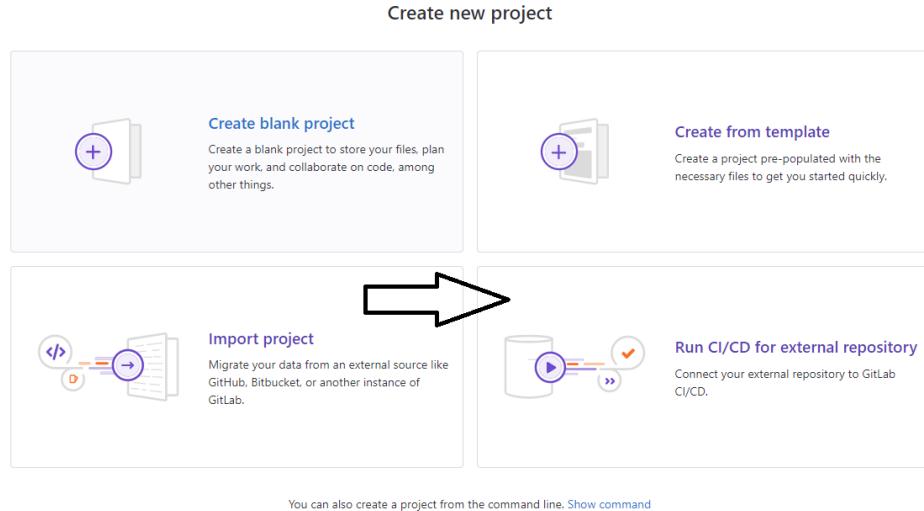
Để thiết lập gitlab CI/CD vào trong mã nguồn của dự án ta cần thực hiện chuyển đổi mã nguồn dự án từ GitHub sang GitLab vì trong Gitlab có tích hợp sẵn và có nhiều tài liệu về việc cấu hình CI/CD trên này ta sẽ thực hiện như sau



Hình 48: Bước một tiên hành tạo dự án mới trên GitLab

Ta vào gitlab.com và đăng nhập vào gitlab

Ở màn hình dự án ta chọn và tạo mới 1 dự án (New Project)

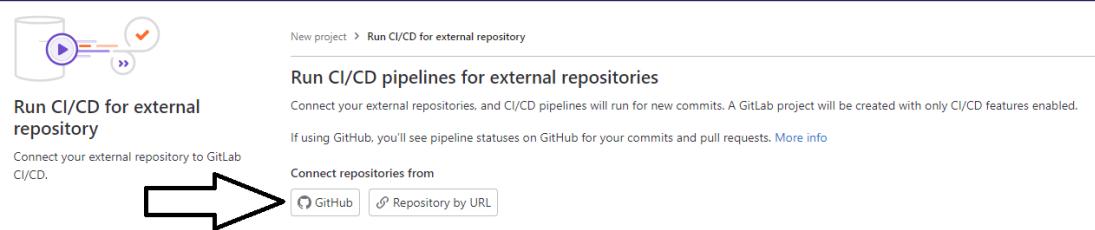


Hình 49: Bước hai chọn chạy dự án mở rộng sử dụng CI/CD

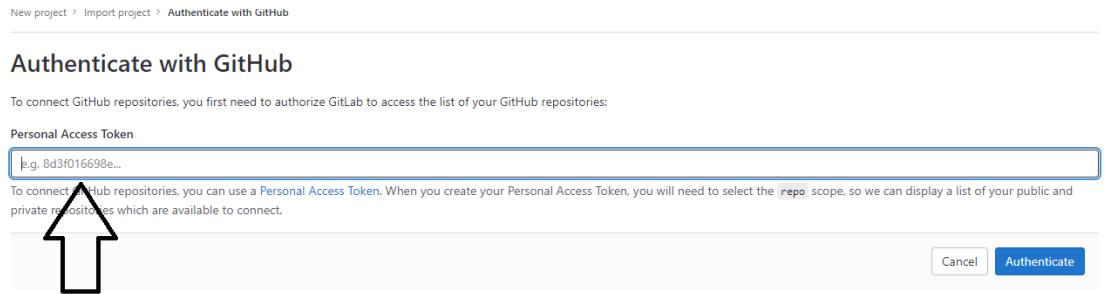
Bước tiếp theo sau khi chọn tạo mới dự án ta sẽ có các lựa chọn như sau :

- Tạo ra 1 dự án trống
- Tạo 1 dự án trống tạo từ mẫu có sẵn
- Thêm dự án bằng câu lệnh
- Chạy dự án mở rộng thêm CI/CD

Với dự án hiện tại ta sẽ chọn lựa chọn là chạy dự án mở rộng thêm CI/CD



Hình 50 : Bước ba chọn sử dụng dự án từ nền tảng GitHub



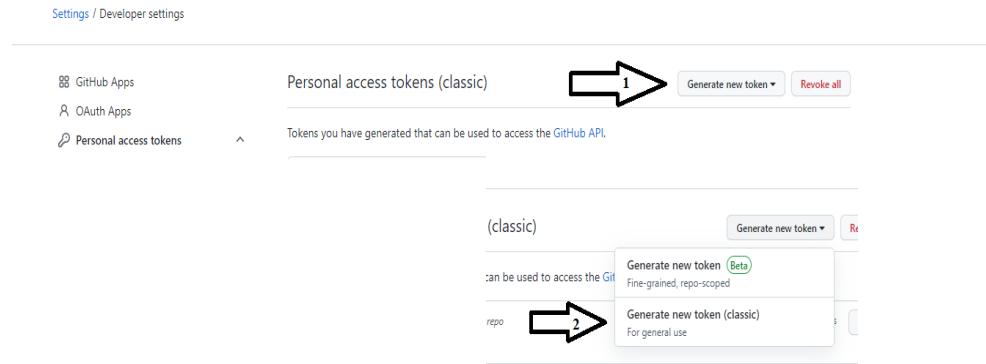
Hình 51: Màn hình yêu cầu ta nhập token ủy quyền

Sau khi chọn vào lựa chọn liên kết với github ta sẽ được chuyển đến màn hình yêu cầu mã token ủy quyền cho gitlab có thẻ liên kết với github

Hình 52: Bước bốn quay lại github để cài đặt

Ta sẽ quay lại github chúa dự án và thực hiện chọn vào settings

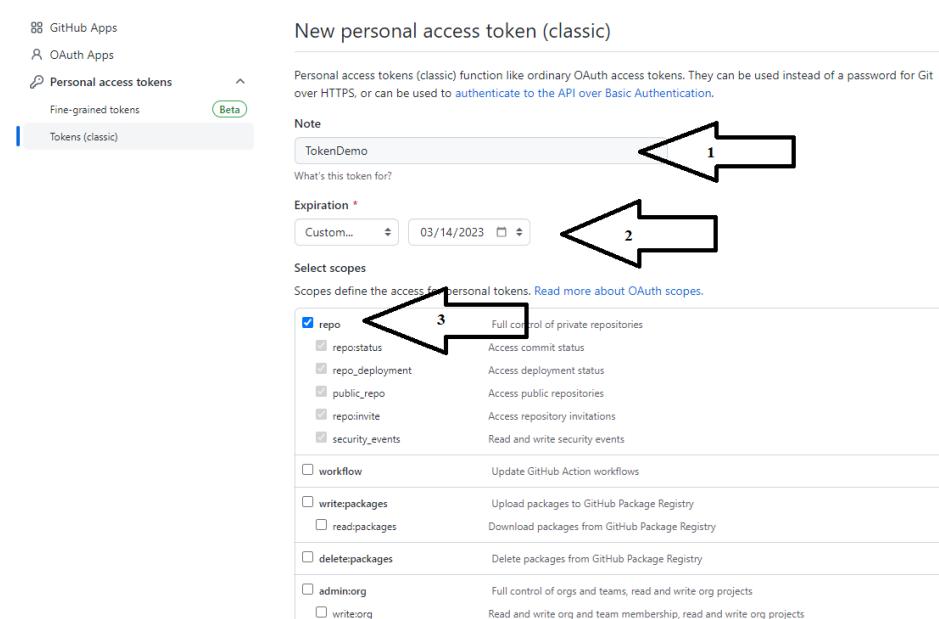
Hình 53: Bước năm chọn vào mục Developer settings



Hình 54: Tạo token ủy quyền

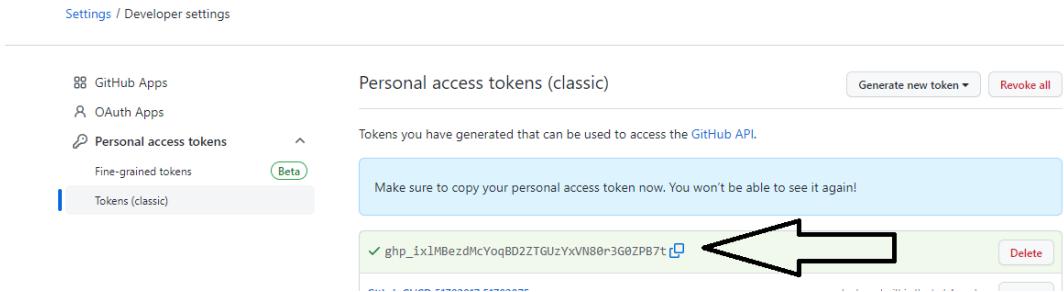
Tại đây ta thực hiện chọn “Generate new token” để tạo mã ủy quyền cho gitlab

(1), Chọn tiếp lựa chọn Generate new token (classic) (2)



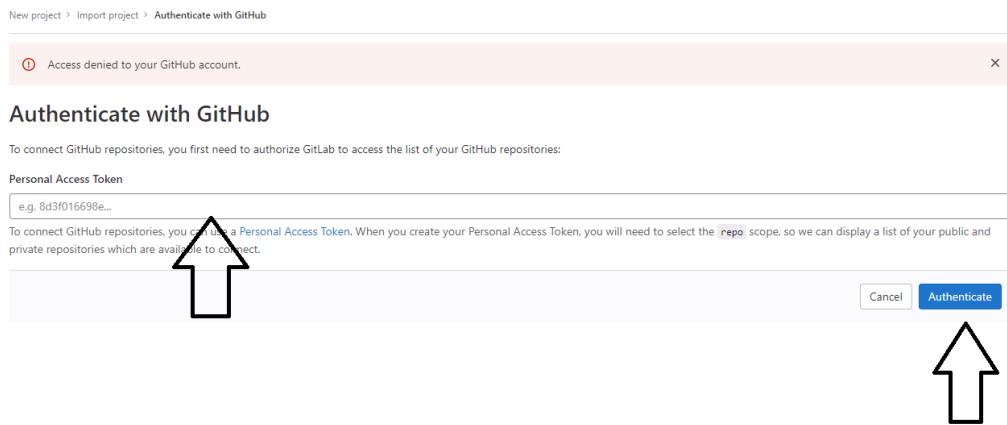
Hình 55: Bước tiếp theo điền các thông tin khi tạo token ủy quyền

Ta sẽ đặt tên cho Token (1) , Chọn thời hạn hết hạn của Token này (2) và cuối cùng là lựa chọn các quyền thao tác với dự án của Token này (3)



Hình 56 : Sau khi khởi tạo token ta sẽ nhận được 1 dãy chữ cái ngẫu nhiên

Ta quay lại màn hình ở gitlab và điền token ủy quyền vào ô và chọn ủy quyền (Authenticate)



Hình 57: Điền token ủy quyền cho gitlab

Cancelling project import failed: The import cannot be canceled because it is finished

Import repositories from GitHub

Select the repositories you want to import

Import 15 repositories

Advanced import settings

⚠️ The more information you select, the longer it will take to import

Import issue and pull request events
For example, opened or closed, renamed, and labeled or unlabeled. Time required to import these events depends on how many issues or pull requests your project has.

Use alternative comments import method
The default method can skip some comments in large projects because of limitations of the GitHub API.

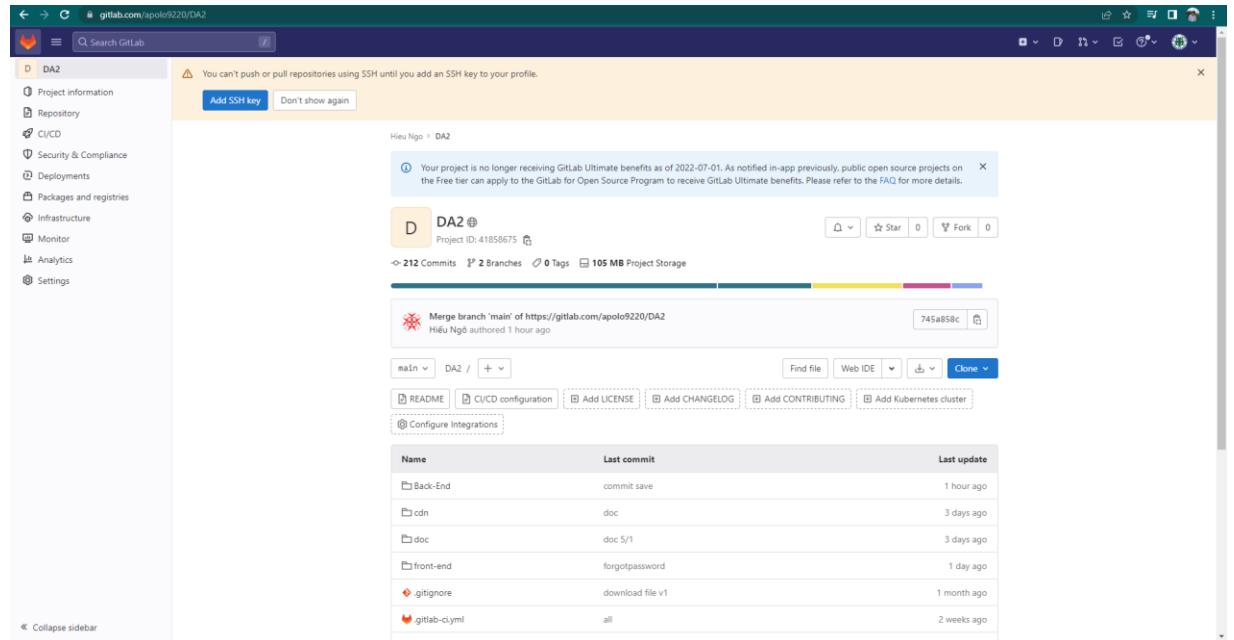
Import Markdown attachments
Import Markdown attachments from repository comments, release posts, issue descriptions, and pull request descriptions. These can include images, text, or binary attachments. If not imported, links in Markdown to attachments break after you remove the attachments from GitHub.

From GitHub	To GitLab	Status
Phoenix22121999/DA2	apolo9220/DA2	Complete 3 Go to project
NolanNgo/Tuhoc1	apolo9220/Tuhoc1	Complete > Details Go to project
NolanNgo/mytraning	apolo9220 / mytraning	Not started 2 Connect
NolanNgo/project-	apolo9220 / project-	Not started Connect
Molvaldrin/Crud Data Slim		

Hình 58: Màn hình sau khi ủy quyền

Sau khi ủy quyền xong thì sẽ hiện ra danh sách các dự án mà tài khoản github tạo Access Token có

Nếu có nhiều dự án ta sẽ sử dụng bộ lọc để lấy được dự án ta cần dùng CI/CD (1), Nếu có dự án ta sẽ chọn Connect và đợi khoản vài phút để Gitlab sẽ lấy dự án từ Github về (2) còn nếu đã kết nối dự án rồi ta sẽ chọn vào Go to project để chuyển hướng đến dự án của ta cần cấu hình CI/CD (3)



Hình 59: Dự án từ Github đã được đưa sang GitLab

Như thế ta hoàn thành xong bước đầu tiên là lấy dự án từ Github sang Gitlab sau đó ta thực hiện việc cấu hình chạy CI/CD trên dự án này với nền tảng Gitlab

Sau khi ta có được dự án ở nền tảng Gitlab ta sẽ tiến hành thiết lập công cụ CI/CD vào trong dự án

Ta sẽ tạo 1 tập tin với tên là `.gitlab-ci.yml` : Đây sẽ là tập tin chứa mã nguồn được viết theo chuẩn của CI/CD để mỗi lần có sự thay đổi về mã nguồn của dự án mà cảng cứ vào đó mà CI/CD dựa vào tập tin này để build cũng như deploy. Cụ thể các đoạn mã trong file này như sau

```
variables:
```

```
IMAGES_NAME : nolanngo20031999
IMAGES_TAG_BACK_END : da2_backend
IMAGES_TAG_FRONT_END : da2_frontend
```

Tag variables : là nơi ta sẽ khai báo các biến dùng chung bao gồm:

`IMAGES_NAME` : Tên trên dockerhub

IMAGES_TAG_BACK_END : Tên của tag để chỉ nơi lưu trữ images khi build của API

IMAGES_TAG_FRONT_END : Tên của tag để chỉ nơi lưu trữ images khi build của Front-end

stages :

- build
- deploy

Tag stages : là nơi ta sẽ định nghĩa các giai đoạn của quá trình CI/CD thao tác trên dự án, Với dự án trên thì ta sẽ khai báo 2 giai đoạn là giai đoạn đóng gói các tài nguyên (Build) và giai đoạn triển khai đưa dự án lên server (Deploy)

```

run_build_jobportal_backend:
  stage : build
  only :
    refs:
      - main
  variables:
    - $CI_COMMIT_DESCRIPTION =~ /(api|all).*/
    - $CI_COMMIT_MESSAGE =~ /(api|all).*/
  image : docker:20.10.16
  services :
    - docker:20.10.16-dind
  variables:
    DOCKER_TLC_CERTDIR : "/certs"
  before_script :
    - docker login -u $REGISTRY_USER -p $REGISTRY_PASS
  script:
    - echo "Runing build image Back End"
    - cd Back-End
    - docker build -t $IMAGES_NAME/$IMAGES_TAG_BACK_END .
    - docker push $IMAGES_NAME/$IMAGES_TAG_BACK_END
    - echo "Build image Back-End success"

```

Đối với api thì ta sẽ khai báo 1 tên đại diện cho quá trình này

- Tên này ta sẽ đặt tùy ý nhưng phải thể hiện chức năng của nó hoạt động là gì
- Stage : ta sẽ khai báo là quá trình này đang trong giai đoạn nào của CI/CD
- Only : refs : - main có nghĩa là sẽ sử dụng nhánh main trong dự án để tiến hành đóng gói tài nguyên

- Variables trong quá trình này sẽ khai báo biến và khi CI/CD nhận thấy sự thay đổi của mã nguồn cùng với tình trạng gộp nhánh (merge) , đưa code lên nhánh (push) thì sẽ thực hiện quá trình này đối với phía Back-End
- Với việc Build image ta sẽ sử dụng 1 image khác thuộc về docker và sử dụng tiện ích dind để build Image của dự án trong chính Image của docker
- Tag before_script : Ta tiến hành login vào docker hub để sau khi tiến hành build images xong thì ta sẽ đẩy images đó lên docker hub để mượn docker hub sẽ lưu trữ images của dự án
- Tag script : những đoạn mã trong tag này sẽ hoạt động tuần tự từ trên xuống dưới
- echo sẽ đẩy ra màn hình message

```

run_deploy_jobportal_backend:
  stage : deploy
  only :
    refs:
      - main
  variables:
    - $CI_COMMIT_DESCRIPTION =~ /(api|all).*/
    - $CI_COMMIT_MESSAGE =~ /(api|all).*/
  before_script:
    - chmod 400 $SSH_KEY
  script:
    - echo "Runing deploy api"
    - echo "Runing deploy api"
    - ssh -o StrictHostKeyChecking=no -i $SSH_KEY root@159.223.54.199 "
      docker stop container backendda2 || true && docker rm backendda2 || true
    &&
      docker rmi $IMAGES_NAME/$IMAGES_TAG_BACK_END || true &&
      docker login -u $REGISTRY_USER -p $REGISTRY_PASS &&
      docker run -d -p 4000:4000 --name backendda2 --hostname backendda2
      $IMAGES_NAME/$IMAGES_TAG_BACK_END"

```

- Quá trình deploy thì cũng khai báo tương tự như quá trình build nhưng sẽ có 1 số thay đổi ở phần stage : sẽ là giai đoạn deploy của CI/CD
- Trước khi chạy đoạn script ta sẽ khai báo mở quyền trên server linux đã thuê bằng câu lệnh **chmod 400 \$SSH_KEY** với **\$SSH_KEY** là key mà ta sử dụng **ssh_key** để tạo và connect với server và quyền 400 là chỉ đọc và không được thực hiện các thao tác ghi gì cả

- Khi bắt đầu deploy ta sẽ thực hiện 1 loạt các câu lệnh để không xảy ra lỗi là ssh **StrictHostKeyChecking = no** sẽ giúp ta trỏ đến server và không bị server kiểm tra key đăng nhập
- Thực hiện dùng docker để tạm ngưng container của API , Xóa container đó đi , và đồng thời xóa images của API có ở server
- Ta Tiến hành đăng nhập vào docker hub
- Và chạy lại container với image khi chạy docker sẽ kiểm tra nếu ở máy chủ chưa có images nào tên đó sẽ thực hiện thao tác là pull images từ docker hub về và build image đó thành container tại máy chủ và như thế sẽ kết thúc quá trình deploy 1 image từ 1 sự thay đổi của mã nguồn dự án do developer thay đổi
- Để có Gitlab CI/CD có thể chạy được thì ta cần cấu hình ở các site trong dự án như Front-end (Giao diện) , Back-end (API) , CDN (Lưu trữ hình ảnh) mỗi site phải cần có 1 DockerFile để đóng gói
- Nhiệm vụ của nó là mỗi khi thao tác với docker để tiến hành đóng gói thì docker sẽ đọc nội dung trong Dockerfile ở site cần đóng gói và thực hiện các đoạn mã được viết trong đó để đóng gói và lấy những tài nguyên cần có để đính kèm vào trong gói mà ta cần để chạy nó ở máy khác hoặc ở server mà không bị lỗi

Đối với DockerFile ở Back-end (API)

```
FROM node:14.17.3-alpine as da2
WORKDIR /app
COPY package*.json .
COPY prisma ./prisma/
RUN npm install
COPY ..
ENV PORT=4000
```

```

ENV ACCESS_TOKEN="51702075DAT_51702017HIEU"
ENV CDN_URL="http://159.223.54.199:3002/"
ENV
DATABASE_URL="sqlserver://159.223.54.199:1433;database=QLHS_TDTU;user=sa;password=51702017Hieu51702075Dat;encrypt=true;trustServerCertificate=true;"
ENV CLIENT_ID_TDTU="454375430021-ins1u408up6msulgvnffrl4f9hn1nlt5.apps.googleusercontent.com"
ENV CLIENT_ID_NORMAL="772360898971-ufhi78bgjm44o9r1megrah02lnfb8f8s.apps.googleusercontent.com"
EXPOSE 4000
CMD ["npm" , "run" , "start"]

```

Ý nghĩa của từng dòng bên trên như sau :

- From :node 14.17.3-alpine as da2 có nghĩa là ta sẽ sử dụng tài nguyên cho site này là node phiên bản 14.17.3 và đặt tên cho nó là da2
- WORKDIR /app có nghĩa là nơi mà thực hiện thao đóng gói sẽ là thư mục /app
- COPY package*.json/ được hiểu là sẽ tiến hành đưa các gói tài nguyên được khai báo trong package và package-lock vào trong gói
- COPY prisma ./prisma/ Ở Dự án này sử dụng công cụ ORM là prisma được đặt trong thư mục ./prisma nên lệnh này được dùng để copy tài nguyên của prisma vào trong gói
- Npm install khi sử dụng node ở bất kì máy nào ta sẽ phải cài đặt bằng câu lệnh này nên khi đóng gói ta sẽ tiến hành cài đặt trước các tài nguyên cho dự án và sau đó đóng gói phần tài nguyên này kèm theo site dự án
- Các biến có ENV = có nghĩa là sẽ khai báo biến môi trường để khi gói được chạy sẽ sử dụng các biến môi trường này vì file .env khai báo biến ở nodejs

nhưng khi ta đóng gói docker sẽ không đóng gói file .env này nên ta cần khai báo biến

- Expose 4000 có nghĩa là khi docker đóng gói xong site dự án này sẽ được đẩy ra là cổng (port) 4000
- CMD [“npm” , “run” , “start”]: đặt các lệnh có thể hoạt động trong quá trình docker chạy gói

Đối với DockerFile ở Front-end (UI)

```

FROM node:14.17.3-alpine AS da2frontend
WORKDIR /app
COPY package*.json .
RUN yarn install
COPY ..

RUN yarn
RUN yarn build

FROM nginx:stable-alpine
COPY --from=da2frontend /app/build /usr/share/nginx/html
RUN rm /etc/nginx/conf.d/default.conf
COPY --from=da2frontend /app/nginx/nginx.conf /etc/nginx/conf.d
# Containers run nginx with global directives and daemon off
EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

```

- Front-end (UI) sẽ có một chút thay đổi do Front-end chạy bằng ReactJs nên DockerFile của Front-end sẽ như sau
- From cũng như Back-end sẽ sử dụng bản node là 14.17.3
- Địa chỉ làm việc sẽ là /app

- Tiến hành việc lấy những tài nguyên trong package , package-lock vào trong gói
- Tiến hành yarn install để cài đặt các tài nguyên được khai báo trong packge.json
- Copy hết các tài nguyên vừa install đó
- Với việc sử dụng ReactJS khi muốn deploy ta sẽ không sử dụng là npm run start , yarn start mà ta sẽ sử dụng lệnh build lúc này ReactJS sẽ build dự án của ta thành 1 folder build trong đó sẽ nén mã nguồn của ta lại thành các file là .html , css , .js để khi react hoạt động chỉ trỏ đến thư mục public và lấy ra tất cả file .html .css và .js để chạy trên trình duyệt web
- Sau khi build thì ta tiến hành giai đoạn tiếp theo trong Docker file thì để deploy được dự án thì sẽ sử dụng một nền tảng khác đó là nginx
- Từ gói ta đã đặt tên docker sẽ copy toàn bộ file build ở gói sang thư mục mặc định mà khi ta start nginx lên là nginx sẽ trả thẳng vào thư mục mặc định này ở đây là /usr/share/nginx/html
- Trong nginx sẽ có 1 file thiết lập mặc định nên ta sẽ tiến hành xóa file thiết lập mặc định này và tiếp theo là lấy toàn bộ file thiết lập của bản thân ra và đưa vào đó

```

server {
    listen 80;
    server_name jobportaltdtu.com;

    location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
        try_files $uri $uri/ /index.html =404;
    }
}

```

```

}

error_page 500 502 503 504 /50x.html;

location = /50x.html {
    root /usr/share/nginx/html;
}

}

```

- File cấu hình nginx sẽ như sau công тро đến nginx sẽ là công 80 và khi ở trên server thì tên của server là jobportaltdtu.com tên miền này được mua ở tenten.vn và khi ta truy cập bên tenten ta cần thiết lập cho tên miền trỏ về địa chỉ ip của server chứa mã nguồn trang web thì khi đó nginx sẽ bắt là khi gõ tên miền jobportaltdtu.com nó sẽ trỏ vào IP của server và chỉ vào công 80 khi này mã nguồn Front-end của trang web sẽ được đặt ở usr/share/nginx/html khi đó trang web của ta sẽ được truy cập thông qua tên miền mà ta cấu hình
- Và các coogn lệnh cho phép hoạt động trong suốt quá trình chạy docker là CMD [“nginx “ “-g” “daemon-off”]
- Phía trên là file thiết lập của từng site là Front-end và Back-end khi start dự án sẽ tiến hành start cả 2 site cùng một lúc thì khi này ở dự án nơi chứa cả 2 site ta sẽ tạo thêm 1 file là Docker-Compose.yml\

```

version: '2.2'

services:

backend:
    build:
        context : ./Back-End
        dockerfile : ./Dockerfile
        # image : 'backendda2'
    container_name: backendda2

```

```

ports :
- 4000:4000

frontend:
build :
context : ./front-end
dockerfile: ./Dockerfile
# image : 'frontda2'
container_name: frontda2

ports :
- 80:80

links :
- "backend:be"

```

- File docker-compose.yml đóng vai trò sẽ chứa các thiết lập mà ta sẽ khai báo khi ta muốn khởi động dự án với những gói nào và các gói riêng biệt được start ở cổng (port) số máy
- version tag được dùng với mục đích là định danh cho quá trình khởi động dự án
- services tag là nơi ta sẽ thiết lập các đoạn mã để build cũng như run các image
- Với dự án này file docker-compose bao gồm 3 services gồm các site là back-end , front-end , cdn (Là nơi chứa các tập tin được upload hoặc là hình ảnh)
- Ví dụ như với site back-end sẽ tiến hành build ở địa chỉ là /back-end và file được docker trỏ đến đọc và build image là Dockerfile được đặt ở site back-end và tương tự các site khác ta khai báo tương tự như vậy và kèm theo port mà người cấu hình docker-compose này muốn site đó được start ở cổng (port) số máy với phần port ta đơn giản hiểu là 3001 : 3001 có nghĩa là ánh xạ cổng 3001 của máy chủ vào cổng 3001 của docker

3. Khởi tạo React App

Sử dụng câu lệnh:

```
npx create-react-app front-end --template typescript
```

Trong đó front-end là tên thư mục của ứng dụng. --template typescript để sử dụng typescript

```
Cd front-end
```

```
Yarn start
```

Web sẽ chạy ở localhost:3000/

4. Cấu trúc source code front end

Src

- apis: chưa các file về api
- assets:
 - images: chứa hình
 - svg: chứa svg
- common: chưa các common component (các component được sử dụng thường xuyên như input hay button)
- components: chứa các component còn lại
- hooks: chứa các custom hook
- pages: chứa các page component
- redux: chưa các file của redux
- routes: chưa các file dùng để thiết lập route
- styles: chứa các css và scss file dùng trong root level
- subpage: chưa các component sub page
- types: chứa các type dùng cho typescript
- utils: chứa các hằng số hoặc hàm được sử dụng thường xuyên
- App.tsx: chứa component App
- index.tsx: thiết lập root của react

5. Cấu hình scss

Cài đặt node-sass:

```
Yarn add node-sass
```

Sử dụng scss:

```
import "./ButtonCommon.scss";
```

6. Cấu hình router

Cài đặt thư viện:

```
yarn add react-router-dom
```

Sử dụng createBrowerRouter để tạo router cho ứng dụng:

```
export const router = createBrowserRouter([
  {
    path: "/",
    element: <AppLayout />,
    errorElement: <Error />,
    children: [
      { path: "/", element: <HomePage />, index: true },
      {
        path: "admin",
        element: <AdminDashboardPage />,
        children: [
          {
            path: "account-type",
            element: <AccountTypeManagerPage />,
          },
        ],
      },
    ],
  },
])
```

```
{
  path: "job-type",
  element: <JobTypeManagerPage />,
},
{
  path: "major",
  element: <MajorManagerPage />,
},
],
},
],
},
]);
}
```

Sau đó sử dụng Component RouterProvider của react-router-dom để thiết lập router cho web:

```
import { RouterProvider } from "react-router-dom";
function App() {

  return (
    <RouterProvider router={router} />
  );
}
```

AppLayout:

```
export function AppLayout(props: HeaderProps) {
  return (
    <div className="app-layout">
      <React.Suspense fallback={<SuspenseLoading size="large" />}>
        <CookiesProvider>
```

```
<ScrollToTop />
<AppHeader />
<React.Suspense fallback={<SuspenseLoading size="large" />}>
  <AppContents>
    <Outlet />
  </AppContents>
</React.Suspense>
<Footer />
</CookiesProvider>
</React.Suspense>
</div> );}
```

Trong AppLayout sử dụng component Outlet để đánh dấu nơi mà các component con sẽ được render. Sau khi thiết lập thi chúng ta truy cập localhost:3000/ chúng ta sẽ truy cập vào Home và localhost:3000/admin để vào trang admin

CHƯƠNG 3 – KẾT QUẢ VÀ KẾT LUẬN

1. Kết quả

1.1 Back-end

Để kiểm tra API và kết quả trả về của API ta có thể sử dụng một số công cụ để gửi request đến API như Postman , Jmeter , Rest-Client , Apiee ,...

Trong đồ án lần này việc gửi request cho API sẽ được sử dụng với thư viện của Visual Studio Code là Rest-Client với thư viện này cho phép người viết mã nguồn kiểm tra API ngay trong chính trình soạn thảo mã nguồn mà không cần phụ thuộc vào công cụ hoạt động dựa trên UI. Việc gửi request bằng Rest-Client sẽ được thực hiện hoàn toàn bằng code theo quy chuẩn của thư viện quy định

Thì với thư viện nay ta sẽ khai báo 1 biến global chứa giá trị là <http://localhost:4000/>

API đăng nhập

- Cách gọi API với đường dẫn api/account/sign-in với phương thức POST và các tham số bên dưới

```
POST {{api_local}}/api/account/sign-in HTTP/1.1
Content-Type: application/json
{
  "user_name": "administrator",
  "password": "admin@123"
}
```

- Thông tin trả về bao gồm thông tin và có AccessToken để thao tác với các API khác

```
{
  "code": 200,
  "status_resposse": true,
  "message": "Đăng nhập thành công",
  "data": {
    "id": "3",
    "username": "administrator",
    "user_type_id": "1",
    "first_name": "Administrator",
    "last_name": "administrator",
    "full_name": "Administrator Account",
    "email": "admin1775@gmail.com",
  },
  "AccessToken": "1 chuỗi ngẫu nhiên và trong đó sẽ chứa các thông tin của user"
}
```

API đăng kí

- Cách gọi API với đường dẫn /api/account/sign-up với phương thức POST và các thông tin bắt buộc như sau

```
POST @{{api_local}}/account/sign-up
Content-Type: application/json

{
    "username" : "administrator5",
    "password" : "admin@123",
    "user_type_id" : 2,
    "first_name" : "Administrator",
    "last_name" : "administrator",
    "full_name" : "Administrator Account",
    "email" : "admin1775@gmail.com"
}
```

- Thông tin trả về với các thông tin dưới

```
{
    "code": 200,
    "status_resposse": true,
    "message": "Đăng kí và đăng nhập thành công",
    "data": {
        "id": "8",
        "username": "administrator5",
        "user_type_id": "2",
        "first_name": "Administrator",
        "last_name": "administrator",
        "full_name": "Administrator Account",
        "email": "admin1775@gmail.com",
    }, "AccessToken": "1 chuỗi ngẫu nhiên" }
```

API cập nhật tài khoản

- Cách gọi API : /account/account-update với phương thức POST

```
POST {{api_local}}/account/account-update HTTP/1.1
Content-Type: application/json
Authorization: Bearer AccessToken được trả về

{
    "first_name" : "Administrator 5 update",
    "last_name" : "Account update",
    "full_name" : "Administrator update",
    "email" : "admin.DA277Saaaaaa@gmail.com",
    "avartar" : "74f62364-72c1-4960-baff-0de7d0945e2c.jpg"
}
```

- Thông tin trả về với các thông tin như sau

```
{
    "code": 200,
    "status_resposse": true,
    "data": {
        "id": "8",
        "username": "administrator5",
        "user_type_id": "2",
        "first_name": "Administrator 5 update",
        "last_name": "Account update",
        "full_name": "Administrator update",
        "email": "admin.DA277Saaaaaa@gmail.com",
        "avartar": "74f62364-72c1-4960-baff-0de7d0945e2c.jpg",
    },
    "message": "Cập nhật tài khoản thành công"
}
```

API thay đổi mật khẩu tài khoản

- Cách gọi API với đường dẫn là /account/change-password và đính kèm token vào trong request để thực hiện chức năng của API này với phương thức POST

```
POST {{api_local}}/account/change-password HTTP/1.1
```

Content-Type: application/json

Authorization: Bearer AccessToken được trả về

```
{
  "current_password": "1234567",
  "new_password": "654321"
}
```

- Thông tin trả về như sau:

```
{
  "code": 200,
  "status_resposse": true,
  "message": "Thay đổi mật khẩu thành công",
  "data": {
    "id": "3",
    "username": "administrator",
    "user_type_id": "1",
    "first_name": "Administrator",
    "last_name": "administrator",
    "full_name": "Administrator Account",
    "email": "admin1775@gmail.com",
  },
  "AccessToken": "chuỗi ngẫu nhiên" }
```

API lấy thông tin chi tiết của tài khoản

- Cách gọi API với đường dẫn /account/get-detail với phương thức GET

```
GET {{api_local}}/account/get-detail
Content-Type: application/json
Authorization: Bearer AccessToken được trả về
```

- Thông tin API trả về như sau

```
{
  "code": 200,
  "status_resposse": true,
  "message": "Lấy thông tin tài khoản",
  "data": {
    "id": "5",
    "username": "administrator",
    "user_type_id": "1",
    "first_name": "Administrator",
    "last_name": "administrator",
    "full_name": "Administrator Account",
    "email": "admin1775@gmail.com",
  }
}
```

API lấy các loại tài khoản

- Cách gọi API với đường dẫn /account-type với phương thức GET

```
GET {{api_local}}/account-type
Content-Type: application/json
Authorization: Bearer AccessToken được trả về
```

- Thông tin trả về như sau:

```
{
  "code": 200,
  "message": "get list account",
  "status_resposse": true,
  "data": [
    {
      "id": "1",
      "user_type_name": "Nhóm Quản Lý",
      "is_active": true, "is_delete": false
    },
    {
      "id": "2",
      "user_type_name": "Nhóm ứng viên",
      "is_active": true, "is_delete": false
    },
    {
      "id": "3",
      "user_type_name": "Nhóm Nhà Tuyển Dụng",
      "is_active": true, "is_delete": false }]
```

API tạo mới 1 nhóm người dùng

- Cách gọi API với đường dẫn là /account-type với phương thức POST

```
POST {{api_local}}/account-type
Content-Type: application/json
Authorization: Bearer AccessToken được trả về

{
    "user_type_name" : "Nhóm quản lý",
    "is_active" : true,
    "is_delete" : false
}
```

- Thông tin API trả về như sau :

```
{
    "code": 200,
    "status_resposse": false,
    "message": "Khởi tạo loại người dùng thành công !"
}
```

API xóa nhóm tài khoản người dùng

- Cách gọi API với đường dẫn /account-type/delete với phương thức PUT

```
POST {{api_local}}/account-type/delete
Content-Type: application/json
Authorization: Bearer string
{
    "user_type_id" : 3
}
```

- Thông tin API trả về như sau

```
{
    "code": 200,
    "status_resposse": true,
    "message": "Xóa tài loại tài khoản thành công!"
}
```

API cập nhật lại thông tin nhóm người dùng

- Cách gọi API với đường dẫn /account-type/update với phương thức PUT

```
PUT {{api_local}}/account-type/update
Content-Type: application/json
Authorization: Bearer AccessToken được trả về

{
    "user_type_id" : 5,
    "user_type_name" : "Cập nhật nè 2",
    "is_active" : true,
    "is_delete" : false }
```

- Thông tin API trả về như sau :

```
{
  "code": 200,
  "status_resposse": true,
  "data": {
    "id": "5",
    "user_type_name": "Cập nhật nè 2",
    "is_active": true,
    "is_delete": false,
    "create_date": "2023-01-01T09:48:12.180Z",
    "update_date": "2023-01-01T00:00:00.000Z"
  },
  "message": "Cập nhật tài loại tài khoản thành công!"
}
```

API tạo 1 đơn cv

- Cách gọi API với đường dẫn /cv với phương thức POST

```
POST {{api_local}}/cv
Content-Type: application/json
Authorization: Bearer AccessToken được trả về
{
  "file_name_hash": "ef6062e0-c2c1-4ced-9ff3-895e91d9f4f5.pdf",
  "file_name" : "2hduthithi800456.pdf",
  "is_active" : true,
  "is_delete" : false
}
```

- Thông tin API trả về như sau :

```
{
  "code": 200,
  "message": "Tạo CV thành công",
  "status_resposse": true,
  "data": {
    "id": "3",
    "file_name_hash": "ef6062e0-c2c1-4ced-9ff3-895e91d9f4f5.pdf",
    "file_name": "2hduthithi800456",
    "user_id": "3",
    "extname": ".pdf",
    "create_date": "2023-01-01T09:56:36.303Z",
    "is_active": true,
    "is_delete": false
  }
}
```

API cập nhật lại tên cho cv

- Cách gọi API với đường dẫn /cv với phương thức PUT

```
PUT {{api_local}}/cv
Content-Type: application/json
Authorization: AccessToken được trả về
{
  "id_cv": 2,
  "file_name_new": "Tập tin mới này 3"
}
```

- Thông tin API trả về như sau:
 - Nếu không có cv có thông tin trả về như sau :

```
{  
    "code": 400,  
    "status_resposse": false,  
    "message": "Không tìm thấy tập tin cv này"  
}
```

- Nếu có cv có thông tin trả về như sau :

```
{  
    "code": 200,  
    "status_resposse": true,  
    "data": {  
        "id": "3",  
        "file_name_hash": "ef6062e0-c2c1-4ced-9ff3-895e91d9f4f5.pdf",  
        "file_name": "Tập tin mới này 3",  
        "user_id": "3",  
        "extname": ".pdf",  
        "create_date": "2023-01-01T09:56:36.303Z",  
        "create_user": null,  
        "is_active": true,  
        "is_delete": false  
    },  
    "message": "Cập nhật tập tin cv thành công"  
}
```

API xóa 1 đơn cv

- Cách gọi API với đường dẫn /cv/delete với phương thức POST

```
POST {{api_local}}/cv/delete
Content-Type: application/json
Authorization: AccessToken được trả về

{
  "cv_id": "2"
}
```

- Thông tin API trả về như sau
 - Nếu không có cv có thông tin trả về như sau :

```
{
  "code": 400,
  "status_resposse": false,
  "message": "Không tìm thấy tập tin cv này"
}
```

- Nếu có cv thì thông tin sẽ trả về như sau :

```
{
  "code": 200,
  "status_resposse": true,
  "data": {
    "id": "3",
    "file_name_hash": "ef6062e0-c2c1-4ced-9ff3-895e91d9f4f5.pdf",
    "file_name": "Tập tin mới này 3",
    "user_id": "3",
    "extname": ".pdf",
    "create_date": "2023-01-01T09:56:36.303Z",
    "update_date": "2023-01-01T09:56:36.303Z"
  }
}
```

```
"create_user": null,  
"is_active": false,  
"is_delete": true  
},  
"message": "Xóa cv thành công"  
}
```

API tải cv về

- Cách gọi API với đường dẫn /cv/download/:id của cv cần tải với phương thức GET

```
GET {{api_local}}/cv/download/:idcv  
Content-Type: application/json  
Authorization: Bearer AccessToken được trả về
```

- Thông tin API trả về như sau : của API sẽ có dạng pipstream và được front-end sử dụng blob để convert thì dạng stream sang file

API lấy danh sách cv của ứng viên

- Cách gọi API với đường dẫn /cv với phương thức GET

```
GET {{ api_local }}/cv
Content-Type: application/json
Authorization: Bearer AccessToken được trả về
```

- o Thông tin API trả về

```
{
  "code": 200,
  "message": "Lấy danh sách thành công",
  "status_resposse": true,
  "data": [
    {
      "id": "4",
      "file_name_hash": "ef6062e0-c2c1-4ced-9ff3-895e91d9f4f5.pdf",
      "file_name": "2hduthithi800456",
      "user_id": "3",
      "extname": ".pdf",
      "create_date": "2023-01-01T10:13:24.293Z",
      "create_user": null,
      "is_active": true,
      "is_delete": false
    },
    {
      "id": "5",
      "file_name_hash": "ef6062e0-c2c1-4ced-9ff3-895e91d9f4f5.pdf",
      "file_name": "aaaaaaaa",
      "user_id": "3",
    }
  ]
}
```

```
    "extname": ".pdf",
    "create_date": "2023-01-01T10:13:32.656Z",
    "create_user": null,
    "is_active": true,
    "is_delete": false
}
]
}
```

API ứng tuyển cv vào bài viết tuyển dụng

- Cách gọi API với đường dẫn /cv/apply-cv với phương thức POST

```
POST {{api_local}}/cv/apply-cv
Content-Type: application/json
Authorization: Bearer AccessToken được trả về
{
    "cv_id": 2,
    "post_id": 1
}
```

- Thông tin API trả về như :

```
{
  "code": 200,
  "status_resposse": true,
  "data": {
    "id": "1",
    "user_id": "4",
    "post_id": "1",
    "cv_id": "2",
    "create_date": "2023-01-03T00:00:00.000Z",
    "create_user": "4",
    "update_date": null,
    "update_user": null,
    "delete_date": null,
    "delete_user": null,
    "is_delete": false,
    "is_active": true,
    "user_account": {
      "id": "4",
      "full_name": "Administrator Account",
      "user_type_id": "2",
      "email": "admin1775@gmail.com",
      "number_phone": null,
      "logo": null,
      "address": null,
      "province_code": null,
      "district_code": null,
      "ward_code": null
    },
    "Recruitment_Post": {
      "id": "1",
      "title": "Bài tuyển dụng testtt",
      "post_job_types": [
        {
          "id": "1",
          "job_type": {
            "id": "2",
            "job_type_name": "Bán thời gian"
          }
        },
        {
          "id": "2",
          "job_type": {
            "id": "1",
            "job_type_name": "Bán thời gian cập nhật"
          }
        }
      ],
      "post_majors": [
        {
          "id": "1",
          "majors": {
            "id": "2",
            "majors_name": "Lập Trình Viên"
          }
        }
      ],
      "cv": {
        "id": "2",
        "file_name_hash": "ef6062e0-c2c1-4ced-9ff3-895e91d9f4f5.pdf",
        "file_name": "hieunm",
        "extname": ".pdf",
        "is_active": true,
        "is_delete": false
      },
      "message": "Quá trình ứng tuyển thành công"
    }
  }
}
```

API rút đơn ứng tuyển

- Cách gọi API với đường dẫn /cv/un-apply-cv với phương thức POST

```

POST {{api_local}}/cv/un-apply-cv
Content-Type: application/json
Authorization: Bearer AccessToken được trả về

{
  "cv_id": 2,
  "post_id": 1
}

```

- Thông tin API trả về như :

```
{
  "code": 200,
  "status_resposse": true,
  "message": "Rút hồ sơ ứng tuyển thành công",
  "data": {
    "id": "4",
    "user_id": "4",
    "post_id": "2",
    "cv_id": "2",
    "create_date": "2023-01-03T00:00:00.000Z",
    "create_user": "4",
    "update_date": null,
    "update_user": null,
    "delete_date": "2023-01-03T00:00:00.000Z",
    "delete_user": null,
    "is_delete": true,
    "is_active": false,
    "user_account": {
      "id": "4",
      "full_name": "Administrator Account",
      "user_type_id": "2",
      "email": "admin1775@gmail.com",
      "number_phone": null,
      "logo": null,
      "address": null,
      "province_code": null,
      "district_code": null,
      "ward_code": null
    },
    "Recruitment_Post": {
      "id": "2",
      "title": "Bài tuyển dụng testtt333",
      "post_job_types": [
        {
          "id": "3",
          "job_type": {
            "id": "2",
            "job_type_name": "Bán thời gian"
          }
        },
        {
          "id": "4",
          "job_type": {
            "id": "1",
            "job_type_name": "Bán thời gian cập nhật"
          }
        }
      ],
      "post_majors": [
        {
          "id": "2",
          "majors": {
            "id": "2",
            "majors_name": "Lập Trình Viên"
          }
        }
      ],
      "cv": {
        "id": "2",
        "file_name_hash": "ef6062e0-c2c1-4ced-9ff3-895e91d9f4f5.pdf",
        "file_name": "hieunm",
        "extname": ".pdf",
        "is_active": true,
        "is_delete": false
      }
    }
  }
}
```

API Lấy danh sách lịch sử đã ứng tuyển của ứng viên

- Cách gọi API với đường dẫn /cv/history-apply

```

GET {{api_local}}/cv/history-apply
Content-Type: application/json
Authorization: Bearer AccessToken được trả về

```

- Thông tin API trả về như sau :

```
{
  "code": 200,
  "message": "Lấy dữ liệu thành công",
  "status_resposse": true,
  "data": [
    {
      "id": "1",
      "user_id": "4",
      "post_id": "1",
      "cv_id": "2",
      "create_date": "2023-01-03T00:00:00.000Z",
      "create_user": "4",
      "update_date": null,
      "update_user": null,
      "delete_date": null,
      "delete_user": null,
      "is_delete": false,
      "is_active": true,
      "user_account": {
        "id": "4",
        "full_name": "Administrator Account",
        "user_type_id": "2",
        "email": "admin1775@gmail.com",
        "number_phone": null,
        "logo": null,
        "address": null,
        "province_code": null,
        "district_code": null,
        "ward_code": null
      },
      "Recruitment_Post": {
        "id": "1",
        "title": "Bài tuyển dụng testtt",
        "post_job_types": [
          {
            "id": "1",
            "job_type": {
              "id": "2",
              "job_type_name": "Bán thời gian"
            }
          },
          {
            "id": "2",
            "job_type": {
              "id": "1",
              "job_type_name": "Bán thời gian cập nhật"
            }
          }
        ],
        "post_majors": [
          {
            "id": "1",
            "majors": {
              "id": "2",
              "majors_name": "Lập Trình Viên"
            }
          }
        ],
        "cv": {
          "id": "2",
          "file_name_hash": "ef6062e0-c2c1-4ced-9ff3-895e91d9f4f5.pdf",
          "file_name": "hieunm",
          "extname": ".pdf",
          "is_active": true,
          "is_delete": false
        },
        "user_id": "4",
        "post_id": "2",
        "cv_id": "1",
        "create_date": "2023-01-03T00:00:00.000Z",
        "create_user": "4",
        "update_date": null,
        "update_user": null,
        "delete_date": null,
        "delete_user": null,
        "is_delete": false,
        "is_active": true,
        "user_account": {
          "id": "4",
          "full_name": "Administrator Account",
          "user_type_id": "2",
          "email": "admin1775@gmail.com",
          "number_phone": null,
          "logo": null,
          "address": null,
          "province_code": null,
          "district_code": null,
          "ward_code": null
        }
      },
      "Recruitment_Post": {
        "id": "2",
        "title": "Bài tuyển dụng testtt333",
        "post_job_types": [
          {
            "id": "3",
            "job_type": {
              "id": "2",
              "job_type_name": "Bán thời gian"
            }
          },
          {
            "id": "4",
            "job_type": {
              "id": "1",
              "job_type_name": "Bán thời gian cập nhật"
            }
          }
        ],
        "post_majors": [
          {
            "id": "2",
            "majors": {
              "id": "2",
              "majors_name": "Lập Trình Viên"
            }
          }
        ],
        "cv": {
          "id": "1",
          "file_name_hash": "ef6062e0-c2c1-4ced-9ff3-895e91d9f4f5.pdf",
          "file_name": "aaaaaaaaaa",
          "extname": ".pdf",
          "is_active": true,
          "is_delete": false
        }
      }
    }
  ]
}
```

API lấy sách ứng viên đã ứng tuyển

- Cách gọi API với đường dẫn /cv/history-applier-post với phương thức GET

```
GET {{api_local}}/cv//history-applier-post
Content-Type: application/json
Authorization: Bearer AccessToken được trả về
```

- Thông tin API trả về như sau

```
{
  "code": 200,
  "message": "Lấy dữ liệu thành công",
  "status_resposse": true,
  "data": [
    {
      "id": "1",
      "user_id": "4",
      "post_id": "1",
      "cv_id": "2",
      "create_date": "2023-01-03T00:00:00.000Z",
      "create_user": "4",
      "update_date": null,
      "update_user": null,
      "delete_date": null,
      "delete_user": null,
      "is_delete": false,
      "is_active": true,
      "user_account": {
        "id": "4",
        "full_name": "Administrator Account",
        "user_type_id": "2",
        "email": "admin1775@gmail.com",
        "number_phone": null,
        "logo": null,
        "address": null,
        "province_code": null,
        "district_code": null,
        "ward_code": null
      },
      "Recruitment_Post": {
        "id": "1",
        "title": "Bài tuyển dụng testtt",
        "post_job_types": [
          {
            "id": "1",
            "job_type": {
              "id": "2",
              "job_type_name": "Bán thời gian"
            }
          },
          {
            "id": "2",
            "job_type": {
              "id": "1",
              "job_type_name": "Bán thời gian cập nhật"
            }
          }
        ],
        "post_majors": [
          {
            "id": "1",
            "majors": [
              {
                "id": "2",
                "majors_name": "Lập Trình Viên"
              }
            ]
          }
        ],
        "cv": {
          "id": "2",
          "file_name_hash": "ef6062e0-c2c1-4ced-9ff3-895e91d9f4f5.pdf",
          "file_name": "hieunm",
          "extname": ".pdf",
          "is_active": true,
          "is_delete": false
        }
      }
    }
  ]
}
```

API lấy danh sách bài đăng tuyển dụng của nhà tuyển dụng

- Cách gọi API với đường dẫn /recruiter-post/list-of-user với phương thức GET

```
GET {{api_local}}/recruiter-post/list-of-user
```

```
Content-Type: application/json
```

```
Authorization: Bearer AccessToken được trả về
```

- Thông tin API trả về như sau :

```
{
  "code": 200,
  "message": "Lấy danh sách bài đăng tuyển dụng thành công",
  "status_resposse": true,
  "data": [
    {
      "id": "5",
      "content": "aaaaaaaaaaaaaaaaaaaa",
      "title": "Bài tuyển dụng testtt",
      "recuiter_id": "8",
      "to_value": "103000",
      "from_value": "2400000",
      "gender": 1,
      "create_date": "2023-01-01T00:00:00.000Z",
      "is_delete": false,
      "is_active": true
    }
  ]
}
```

API tạo bài tuyển dụng

- Cách gọi API với đường dẫn /recruiter-post với phương thức POST

```

POST {{api_local}}/recruiter-post
Content-Type: application/json
Authorization: Bearer AccessToken được trả về

{
    "title" : "Bài tuyển dụng testtt",
    "content" : "caaaaaasssssssssssc",
    "to_value" : 103000,
    "from_value" : 2400000,
    "is_active" : true,
    "is_delete" : false,
    "gender" : 1,
    "list_job_type" : [{"job_type_id" : 1},{ "job_type_id" : 2}],
    "list_major" : [{"majors_id" : 2}]
}

```

- Thông tin API sẽ trả về như sau:

```

{
    "code": 200,
    "message": "Tạo bài đăng thành công",
    "status_resposse": true,
    "data": {
        "id": "5",
        "content": "caaaaaasssssssssssc",
        "title": "Bài tuyển dụng testtt",
        "recuiter_id": "8",
        "to_value": "103000",
        "from_value": "2400000",
    }
}

```

```

"gender": 1,
"create_date": "2023-01-01T00:00:00.000Z",
"is_delete": false,
"is_active": true} }
```

API cập nhật bài đăng

- Cách gọi API với đường dẫn /recruiter-post với phương thức PUT

```

PUT {{api_local}}/recruiter-post
Content-Type: application/json
Authorization: Bearer AccessToken được trả về

{
  "post_id" : "11",
  "title" : "title update",
  "content" : "contentdddsssssss",
  "to_value" : 11111323211,
  "from_value" : 22221122222,
  "is_active" : true,
  "gender" : 1,
  "list_job_type" : [{"job_type_id" : 2}],
  "list_major" : [{"majors_id" : 2}]
}
```

- Thông tin API trả về như sau :

```

{
  "code": 200,
  "message": "Cập nhật bài đăng thành công",
  "status_resposse": true,
  "data": {
    "id": "5",
```

```

    "content": "contentdddssssss",
    "title": "title update",
    "recuiter_id": "8",
    "to_value": "11111323211",
    "from_value": "22221122222",
    "gender": 1,
    "update_user": "8",
    "is_delete": false, "is_active": true
  }
}

```

API xóa bài đăng tuyển dụng

- Cách gọi API với đường dẫn /recruiter-post/delete với phương thức POST

```

POST {{api_local}}/recruiter-post/delete
Content-Type: application/json
Authorization: Bearer AccessToken được trả về
{
  "post_id" : 2
}

```

- Thông tin API trả về như sau :

```

{
  "code": 200,
  "status_resposse": true,
  "data": {
    "id": "5",
    "content": "contentdddssssss",
    "title": "title update",
    "recuiter_id": "8",
    "to_value": "11111323211",
}

```

```

    "from_value": "22221122222",
    "gender": 1,
    "delete_date": "2023-01-01T00:00:00.000Z",
    "delete_user": "8",
    "is_delete": true,
    "is_active": false
},
"message": "Xóa bài đăng thành công" }
```

API lấy danh sách bài đăng

- Cách gọi API với đường dẫn /recruiter-post/{các thông số khác để lọc ra các bài đăng} với phương thức là GET

```

GET {{api_local}}/recruiter-
post?from_value=100000&to_value=100000&page=1&item_per_page=2
Content-Type: application/json
```

- Thông tin API trả về như sau :

```

{ "code": 200, "message": "Lấy danh sách thành công",
"status_resposse": true, "data": { "result": [ { "id": "2",
"title": "Bài tuyển dụng 1", "recuiter_id": "3", "to_value": "100000",
"from_value": "2400000", "gender": 1, "address": null,
"province_code": "79", "district_code": "760",
"ward_code": "26734", "create_date": "2022-11-
29T00:00:00.000Z", "create_user": null, "update_date": null,
"update_user": null, "delete_date": null, "delete_user": null,
"is_delete": false, "is_active": true, "user": { "id": "3",
"full_name": "Administrator Account", "user_type_id": "1",
"email": "admin1775@gmail.com", "number_phone": null,
```

```
"logo": null, "address": null, "province_code": null,  
"district_code": null, "ward_code": null },  
"post_job_types": [ { "id": "1", "job_type": { "id": "2",  
"job_type_name": "Bán thời gian 2" } }, { "id": "2",  
"job_type": { "id": "1", "job_type_name": "Toàn thời gian 2"  
} } ], "post_majors": [ { "id": "1", "majors": { "id": "2",  
"majors_name": "Kế Toán 1" } } ], "provinces": { "id": "28",  
"code": "79", "name": "Hồ Chí Minh", "name_en": "Ho Chi  
Minh", "full_name": "Thành phố Hồ Chí Minh", "full_name_en":  
"Ho Chi Minh City", "code_name": "ho_chi_minh" },  
"districts": { "id": "26", "code": "760", "name": "1",  
"name_en": "1", "full_name": "Quận 1", "full_name_en":  
"District 1", "code_name": "1" }, "wards": { "id": "9996",  
"code": "26734", "name": "Tân Định", "name_en": "Tan Dinh",  
"full_name": "Phường Tân Định", "full_name_en": "Tan Dinh  
Ward", "code_name": "tan_dinh" } } ], "total": 1 } }
```

API đánh dấu bài đăng tuyển dụng

- Cách gọi API với đường dẫn /recruiter-post/bookmark-post với phương thức POST

```
POST {{api_local}}/recruiter-post/bookmark-post
Content-Type: application/json
Authorization: Bearer AccessToken được trả về
{
  "post_id": 1
}
```

- Thông tin API trả về như sau:

```
{
  "code": 200,
  "status_resposse": true,
  "data": {
    "id": "1",
    "user_id": "4",
    "post_id": "1",
    "create_date": "2023-01-03T00:00:00.000Z",
    "create_user": "4",
    "update_date": null,
    "update_user": null,
    "delete_date": "2023-01-03T00:00:00.000Z",
    "delete_user": null,
    "is_delete": false,
    "is_active": true,
    "user_account": {
      "id": "4",
      "full_name": "Administrator Account",
      "user_type_id": "2",
      "email": "admin1775@gmail.com",
      "number_phone": null,
      "logo": null,
      "address": null,
      "province_code": null,
      "district_code": null,
      "ward_code": null
    },
    "Recruitment_Post": {
      "id": "1",
      "title": "Bài tuyển dụng testtt",
      "post_job_types": [
        {
          "id": "1",
          "job_type": {
            "id": "2",
            "job_type_name": "Bán thời gian"
          }
        },
        {
          "id": "2",
          "job_type": {
            "id": "1",
            "job_type_name": "Bán thời gian cập nhật"
          }
        }
      ],
      "post_majors": [
        {
          "id": "1",
          "majors": {
            "id": "2",
            "majors_name": "Lập Trình Viên"
          }
        }
      ]
    },
    "message": "Yêu thích bài viết thành công"
  }
}
```

API bỏ đánh dấu bài đăng tuyển dụng

- Cách gọi API với đường dẫn /recruiter-post/un-bookmark-post với phương thức POST

```
POST {{api_local}}/recruiter-post/bookmark-post
```

Content-Type: application/json

Authorization: Bearer AccessToken được trả về

```
{
  "post_id": 1
}
```

- Thông tin API trả về như sau:

```
{
  "code": 200,
  "status_resposse": true,
  "message": "Bỏ yêu thích bài đăng thành công",
  "data": {
    "id": "1",
    "user_id": "4",
    "post_id": "1",
    "create_date": "2023-01-03T00:00:00.000Z",
    "create_user": "4",
    "update_date": null,
    "update_user": null,
    "delete_date": "2023-01-03T00:00:00.000Z",
    "delete_user": null,
    "is_delete": true,
    "is_active": false,
    "user_account": {
      "id": "4",
      "full_name": "Administrator Account",
      "user_type_id": "2",
      "email": "admin1775@gmail.com",
      "number_phone": null,
      "logo": null,
      "address": null,
      "province_code": null,
      "district_code": null,
      "ward_code": null
    },
    "Recruitment_Post": {
      "id": "1",
      "title": "Bài tuyển dụng testtt",
      "post_job_types": [
        {
          "id": "1",
          "job_type": {
            "id": "2",
            "job_type_name": "Bán thời gian"
          }
        },
        {
          "id": "2",
          "job_type": {
            "id": "1",
            "job_type_name": "Bán thời gian cập nhật"
          }
        }
      ],
      "post_majors": [
        {
          "id": "1",
          "majors": {
            "id": "2",
            "majors_name": "Lập Trình Viên"
          }
        }
      ]
    }
  }
}
```

API lấy danh sách bài đăng được đánh dấu

- Cách gọi API với đường dẫn /recruiter-post/list-bookmark.... Với phương thức GET

```
GET {{api_local}}/recruiter-post/list-bookmark?item_per_page=10&page=1
```

Content-Type: application/json

Authorization: Bearer AccessToken được trả về

- Thông tin API trả về như sau:

```
{
  "code": 200,
  "message": "Lấy dữ liệu thành công",
  "status_resposse": true,
  "data": [
    {
      "id": "1",
      "user_id": "4",
      "post_id": "1",
      "create_date": "2023-01-03T00:00:00.000Z",
      "create_user": "4",
      "update_date": null,
      "update_user": null,
      "delete_date": "2023-01-03T00:00:00.000Z",
      "delete_user": null,
      "is_delete": false,
      "is_active": true,
      "user_account": {
        "id": "4",
        "full_name": "Administrator Account",
        "user_type_id": "2",
        "email": "admin1775@gmail.com",
        "number_phone": null,
        "logo": null,
        "address": null,
        "province_code": null,
        "district_code": null,
        "ward_code": null
      },
      "Recruitment_Post": {
        "id": "1",
        "title": "Bài tuyển dụng testtt",
        "post_job_types": [
          {
            "id": "1",
            "job_type": {
              "id": "2",
              "job_type_name": "Bán thời gian"
            }
          },
          {
            "id": "2",
            "job_type": {
              "id": "1",
              "job_type_name": "Bán thời gian cập nhật"
            }
          }
        ],
        "post_majors": [
          {
            "id": "1",
            "majors": {
              "id": "2",
              "majors_name": "Lập Trình Viên"
            }
          }
        ]
      }
    }
  ]
}
```

API tạo loại nghề nghiệp

- Cách gọi API với đường dẫn /job-type với phương thức POST

```
POST {{api_local}}/job-type
Content-Type: application/json
Authorization: Bearer AccessToken được trả về
```

```
{
  "job_type_name": "Bán thời gian 2"
}
```

- Thông tin API trả về như sau

- Nếu không có thuộc nhóm quản lý mà thực hiện gọi API này

```
{
  "code": 400,
  "message": "Người dùng không hợp lệ",
```

```

    "status_resposse": false
}
```

- Nếu thuộc nhóm quản lý gọi API này

```
{
  "code": 200,
  "message": "Tạo loại công việc thành công",
  "status_resposse": true,
  "data": {
    "id": "2",
    "job_type_name": "Bán thời gian",
    "is_active": true, "is_delete": false }}
```

API lấy danh sách loại nghề nghiệp

- Cách gọi API với đường dẫn /job-type?item_per_page=10& page =1 với phương thức GET

```
GET {{api_local}}/job-type?item_per_page=10&page=1&key_word=
Content-Type: application/json
```

- Kết quả API trả về như sau

```
{
  "code": 200,
  "message": "Lấy danh sách thành công",
  "status_resposse": true,
  "data": [
    {
      "id": "1",
      "job_type_name": "Bán thời gian 2",
```

```

    "is_active": true,
    "is_delete": false,
},
{
    "id": "2",
    "job_type_name": "Bán thời gian",
    "is_active": true,
    "is_delete": false,
}
]
}

```

API cập nhật lại thông tin của loại nghề nghiệp

- Cách gọi API với đường dẫn /job-type với phương thức PUT

```

PUT {{api_local}}/job-type
Content-Type: application/json
Authorization: Bearer AccessToken được trả về
{
    "job_type_id" : 1,
    "job_type_name" : "Bán thời gian cập nhật"
}

```

- Thông tin API trả về như sau:

```

{
    "code": 200,
    "message": "Cập nhật loại công việc thành công",
    "status_resposse": true,
    "data": {
        "id": "1",
        "job_type_name": "Bán thời gian cập nhật",
    }
}

```

```
"is_active": true,  
"is_delete": false,  
}  
}
```

API xóa loại nghề nghiệp

- Cách gọi API với đường dẫn /job-type/delete với phương thức POST

```
POST {{api_local}}/job-type/delete
Content-Type: application/json
Authorization: Bearer AccessToken được trả về

{
  "job_type_id": 1
}
```

- Thông tin API trả về như sau:

```
{
  "code": 200,
  "message": "xóa loại công việc thành công",
  "status_resposse": true,
  "data": {
    "id": "1",
    "job_type_name": "Bán thời gian cập nhật",
    "is_active": false,
    "is_delete": true,
    "create_date": "2023-01-02T00:00:00.000Z",
    "create_user": "5",
    "update_date": "2023-01-02T00:00:00.000Z",
    "update_user": "5",
    "delete_date": "2023-01-02T00:00:00.000Z",
    "delete_user": "5"
  }
}
```

API tạo mới ngành nghề

- Cách gọi API với đường dẫn /majors với phương thức POST

```
POST {{api_local}}/majors
Content-Type: application/json
Authorization: Bearer AccessToken được trả về

{
    "majors_name" : "Kế Toán"
}
```

- Thông tin API trả về như sau :

```
{
  "code": 200,
  "message": "Tạo ngành nghề thành công",
  "status_resposse": true,
  "data": {
    "id": "1",
    "majors_name": "Kế Toán",
    "is_active": true,
    "is_delete": false,
    "create_date": "2023-01-02T00:00:00.000Z",
    "create_user": "5"
  }
}
```

API cập nhật lại thông tin ngành nghề

- Cách gọi API với đường dẫn /majors với phương thức PUT

```
PUT {{api_local}}/majors
Content-Type: application/json
Authorization: Bearer AccessToken được trả về

{
    "majors_id": 1,
    "majors_name": "Kế Toán cập nhật"
}
```

Thông tin API trả về như sau

```
{
    "code": 200,
    "message": "cập nhật ngành nghề thành công",
    "status_resposse": true,
    "data": {
        "id": "1",
        "majors_name": "Kế Toán cập nhật",
        "is_active": true,
        "is_delete": false,
        "create_date": "2023-01-02T00:00:00.000Z",
        "create_user": "5",
        "update_date": "2023-01-02T00:00:00.000Z",
        "update_user": "5",
    }
}
```

API xóa loại nghề nghiệp

- Cách gọi API với đường dẫn /majors/delete với phương thức POST

```
POST {{api_local}}/majors/delete
Content-Type: application/json
Authorization: Bearer AccessToken được trả về
{
  "majors_id": 1
}
```

- Thông tin API trả về như sau

```
{
  "code": 200,
  "message": "Xóa ngành nghề thành công",
  "status_resposse": true,
  "data": {
    "id": "1",
    "majors_name": "Kế Toán cập nhật",
    "is_active": false,
    "is_delete": true,
    "create_date": "2023-01-02T00:00:00.000Z",
    "create_user": "5",
    "update_date": "2023-01-02T00:00:00.000Z",
    "update_user": "5",
    "delete_date": "2023-01-02T00:00:00.000Z",
    "delete_user": "5"
  }
}
```

API lấy danh sách loại nghề nghiệp

- Cách gọi API với đường dẫn /majors với phương thức GET

```
GET {{api_local}}/majors
Content-Type: application/json
```

- Thông tin API trả về như sau:

```
{
  "code": 200,
  "message": "Lấy danh sách thành công",
  "status_resposse": true,
  "data": [
    {
      "id": "2",
      "majors_name": "Lập Trình Viên",
      "is_active": true,
      "is_delete": false,
      "create_date": "2023-01-02T00:00:00.000Z",
      "create_user": "5"
    },
    {
      "id": "3",
      "majors_name": "Quản Trị Nhà Hàng",
      "is_active": true,
      "is_delete": false,
      "create_date": "2023-01-02T00:00:00.000Z",
      "create_user": "5"
    }
  ]
}
```

API Lấy danh sách các tỉnh thành

- Cách gọi API với đường dẫn /address/provinces với phương thức GET

```
GET {{api_local}}/address/provinces
Content-Type: application/json
```

- Thông tin API trả về như sau:

```
{
  "code": 200,
  "message": "Lấy danh sách tỉnh/Thành thành công",
  "status_resposse": true,
  "data": [
    {
      "id": "1",
      "code": "01",
      "name": "Hà Nội",
      "name_en": "Ha Noi",
      "full_name": "Thành phố Hà Nội",
      "full_name_en": "Ha Noi City",
      "code_name": "ha_noi",
      "administrative_unit_id": 1,
      "adminstrative_region_id": 3
    },
    {
      "id": "28",
      "code": "79",
      "name": "Hồ Chí Minh",
      "name_en": "Ho Chi Minh",
    }
  ]
}
```

```

    "full_name": "Thành phố Hồ Chí Minh",
    "full_name_en": "Ho Chi Minh City",
    "code_name": "ho_chi_minh",
    "administrative_unit_id": 1,
    "adminstrative_region_id": 7
}
]
}

```

API lấy danh sách các quận/huyện trong tỉnh

- Cách gọi API với đường dẫn /address/district?province_code={số} với phương thức GET

```

GET {{api_local}}/address/district?province_code=79
Content-Type: application/json

```

- Thông tin API trả về như sau:

```

{
  "code": 200,
  "message": "Lấy danh sách Quận/Huyện thành công",
  "status_resposse": true,
  "data": [
    {
      "id": "26",
      "code": "760",
      "name": "1",
      "name_en": "1",
      "full_name": "Quận 1",
      "full_name_en": "District 1",
    }
  ]
}

```

```

    "code_name": "1",
    "administrative_unit_id": 5,
    "province_code": "79"
  },
  {
    "id": "570",
    "code": "766",
    "name": "Tân Bình",
    "name_en": "Tan Binh",
    "full_name": "Quận Tân Bình",
    "full_name_en": "Tan Binh District",
    "code_name": "tan_binh",
    "administrative_unit_id": 5,
    "province_code": "79"
  }
]
}

```

- API lấy danh sách các phường xã trong quận/huyện
 - o Cách gọi API với đường dẫn /address/ward?district_code = {số} với phương thức GET

```

GET {{api_local}}/address/ward?district_code=760
Content-Type: application/json

```

- o Thông tin API trả về như sau:

```

{
  "code": 200,
  "message": "Lấy danh sách Phường/Xã thành công",
  "status_resposse": true,
  "data": [
    {

```

```
"id": "9996",
"code": "26734",
"name": "Tân Định",
"name_en": "Tan Dinh",
"full_name": "Phường Tân Định",
"full_name_en": "Tan Dinh Ward",
"code_name": "tan_dinh",
"district_code": "760",
"administrative_unit_id": 8
},  {
"id": "9997",
"code": "26737",
"name": "Đa Kao",
"name_en": "Da Kao",
"full_name": "Phường Đa Kao",
"full_name_en": "Da Kao Ward",
"code_name": "da_kao",
"district_code": "760",
"administrative_unit_id": 8
}
]}
```

1.2 Front-end

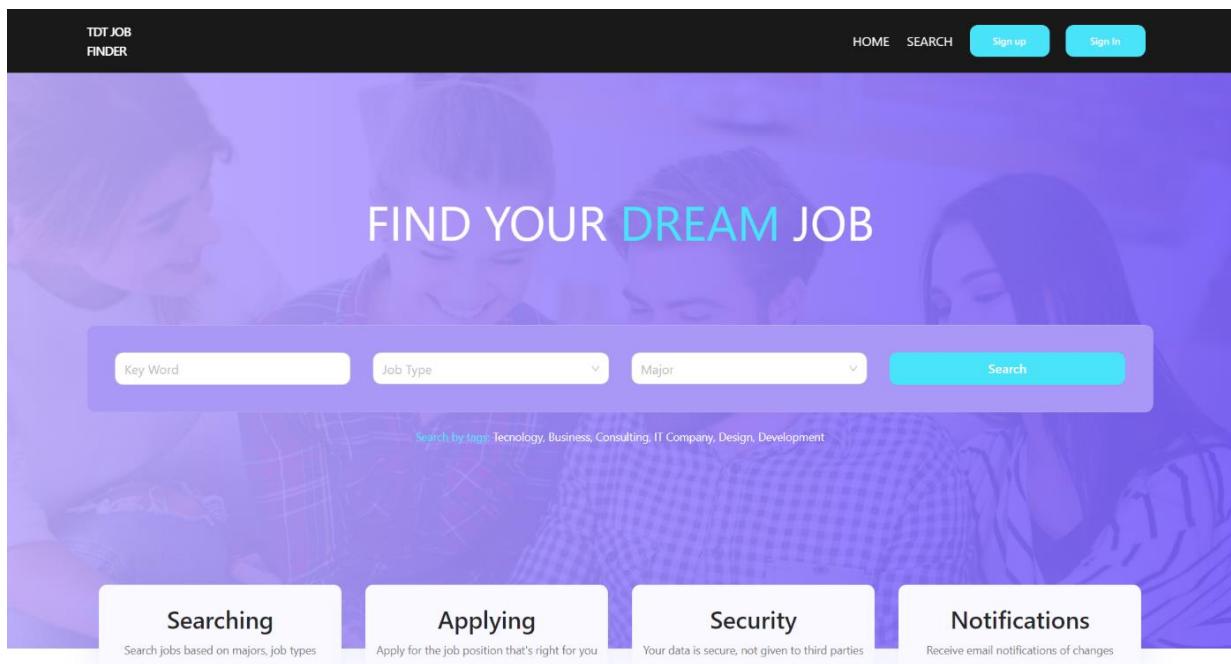
a. Trang chủ

- Banner

Trên cùng của trang chủ là banner, ở giữa banner là thanh tìm kiếm, sao khi chọn các thông tin mà mình muốn và bấm search, người dùng sẽ chuyển đến trang tìm kiếm với thông tin đã chọn.

Bên dưới là các tính năng của trang web bao gồm:

- Searching
- Applying
- Security
- Notification

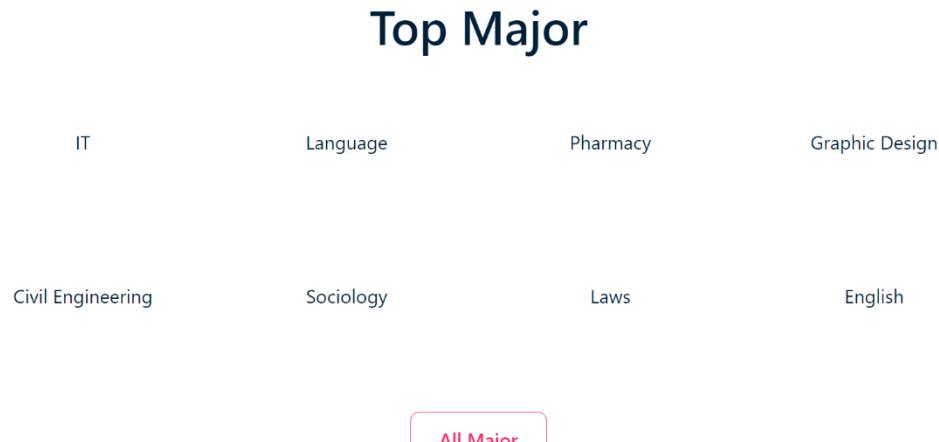


Hình 60 Home Page - Banner

- Top Major

Phần tiếp theo là Top Major sẽ hiện 8 chuyên ngành mới nhất, nếu người dùng bấm vào 1 chuyên ngành sẽ được chuyển đến trang tìm kiếm với filter có chuyên ngành đã chọn.

Nếu người dùng chọn All Major thì sẽ chuyển tới trang tìm kiếm với filter rỗng



Hình 61 Home Page – Top Major

- Feature Job

Phần feature job sẽ hiện 5 bài đăng mới nhất với thông tin chi tiết. Khi người dùng bấm vào view job sẽ được chuyển đến trang Job Detail

Nếu người dùng bấm vào All Job sẽ chuyển đến trang tìm kiếm với filter rỗng.

Featured Job

The screenshot displays four featured job listings from the Zigo platform:

- 1** Hồ Chí Minh Full Time Language Male
Công ty Áo
- 2** Hà Nội Part Time IT Female
Công ty Áo
- 3** Công ty Áo
- 4** Công ty Áo

Each listing includes a small Zigo logo, the job title, company name, a 'View Job' button, and filters for Major (IT), Job Type (Full Time or Part Time), Location (various districts in Hanoi and Ho Chi Minh City), and Salary (0đ - 10,000,000đ).

All Job

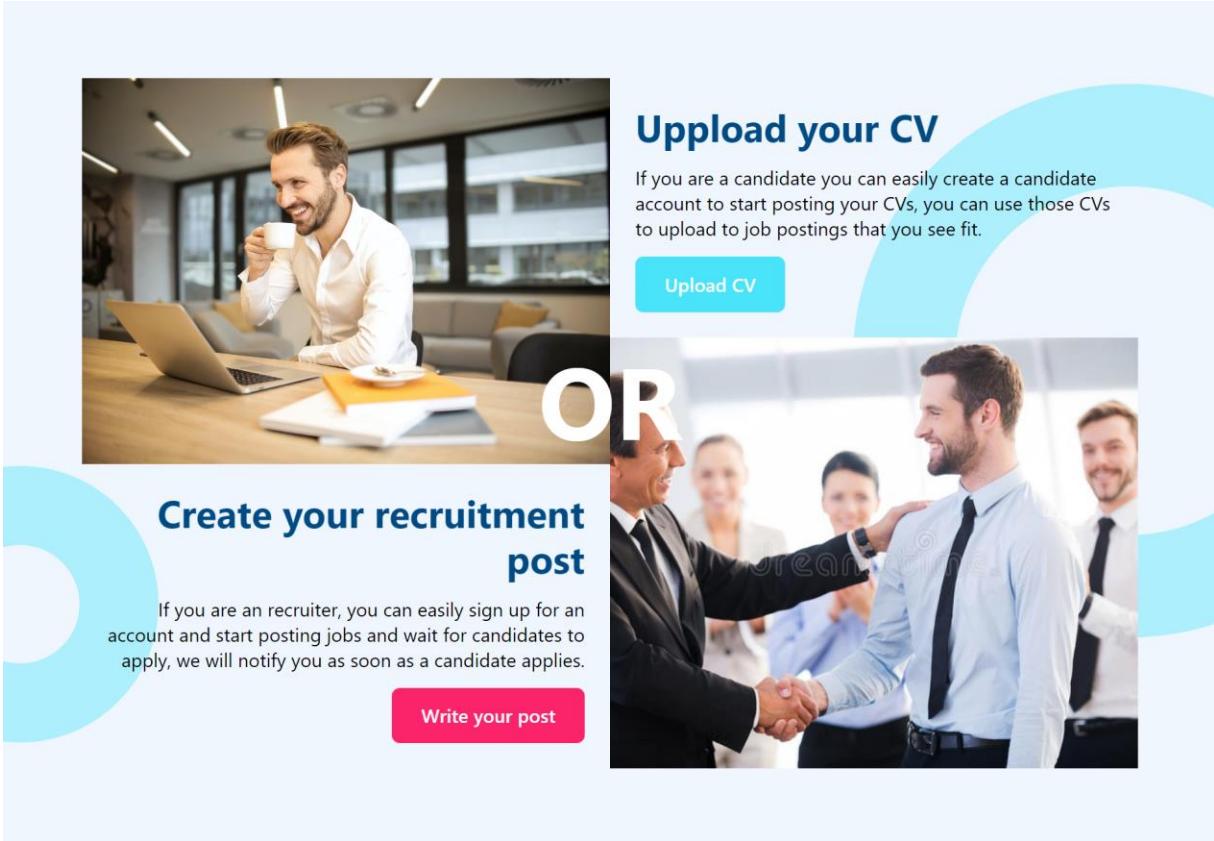
Hình 62: Home Page – Featured Job

• Recruitment

Phần này sẽ khuyến khích các người dùng đăng ký tài khoản và sử dụng các chức năng của trang web. Phần recruitment gồm 2 phần nhỏ:

- Phân cho candidate.
- Phân cho recruiter.

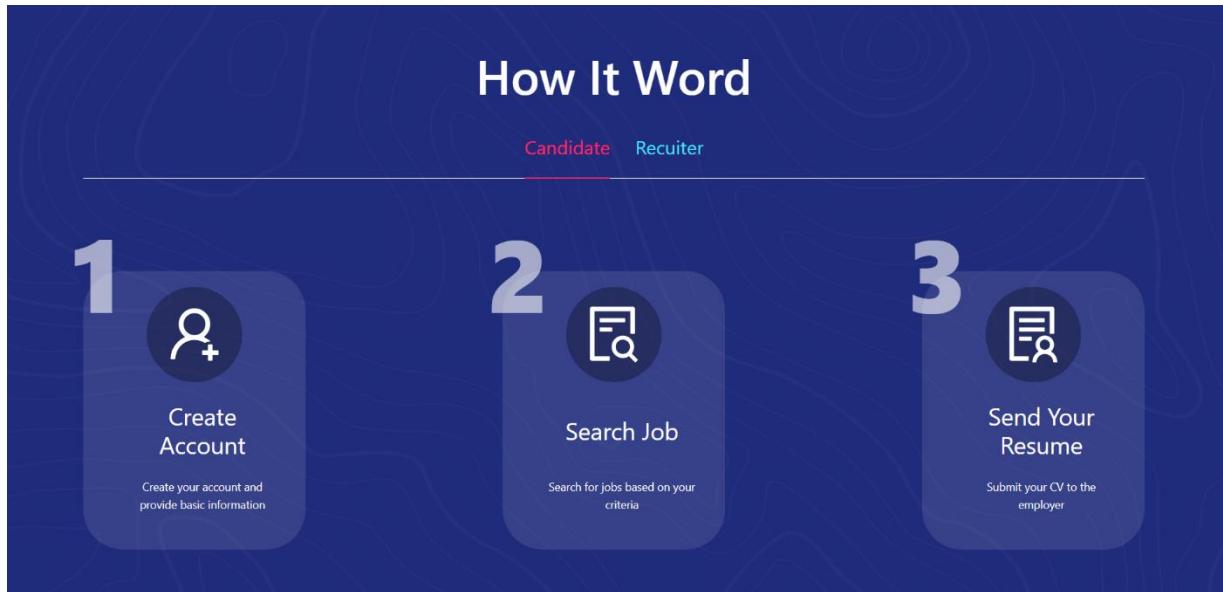
Khi người dùng bấm vào Upload CV hoặc Write your post sẽ được chuyển sang đăng nhập.



Hình 63: Home Page – Recruiement

- How it word

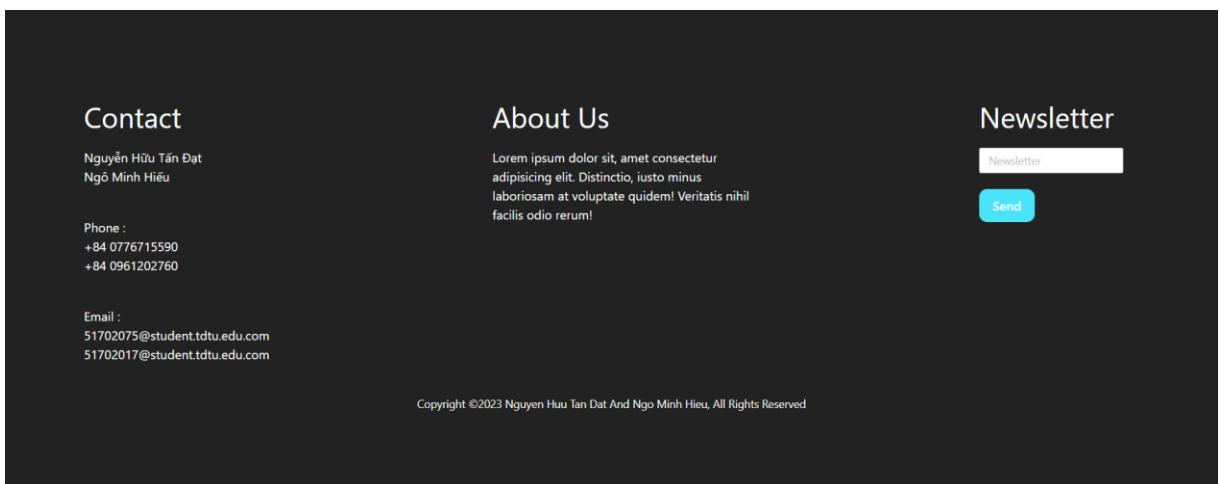
Phần cuối cùng của trang home là How it word, phần này sẽ hiện các bước sửa dụng trang web cho candidate và recruiter



Hình 64: Home Page – How It Work

b. Footer

Phần footer sẽ hiển thị thông tin của Nguyễn Hữu Tấn Đạt và Ngô Minh Hiếu



Hình 65: Footer

c. Header

Phần header gồm logo của trang web nằm bên trái, bên phải là thanh điều hướng. Thanh điều hướng sẽ thay đổi dựa trên loại người dùng đã đăng nhập. Nếu đăng nhập sẽ xuất hiện các trang tương ứng với loại người dùng, nếu chưa đăng nhập sẽ chỉ có 4 trang là home, search, đăng ký, đăng nhập



Hình 66: Hearder - UnAuth



Hình 67: Header - Candidate



Hình 68: Header – Recruiter



Hình 69: Header - Admin

d. Search

Trang search gồm 2 phần bên trái là thanh search filter, bên phải là danh sách các bài post được phân 10 bài 1 trang. Khi người dùng thay đổi thông tin bên thanh search filter thì kết quả sẽ trả về bên phải. người dùng có thể search theo các thuộc tính sau:

- Từ khóa: sẽ trả về các bài đăng có tiêu đề chưa từ khóa mà người dùng nhập
- Vị trí: sẽ trả về các bài đăng có thành phố, quận/huyện, phường/xã mà người dùng nhập
- Chuyên ngành: sẽ trả về các bài đăng có chuyên ngành bao gồm các chuyên ngành mà người dùng chọn, người dùng có thể chọn được nhiều chuyên ngành 1 lúc
- Loại công việc: sẽ trả về các bài đăng có loại là loại công việc là người dùng chọn
- Ngày đăng: sẽ trả về các bài đăng được đăng trong khoảng mà người dùng chọn
- Khoản lương: sẽ trả về các bài post có mức lương nằm trong khoảng mà người dùng chọn
- Người dùng có thể kết hợp các filter với nhau để tìm kiếm

Find Your Job

Home / Search

Showing 1-10 of 12 posts

Create Date ASC

Key Words

Location

Province:

District:

Ward:

Major

Job Type

Full Time

Part Time

Date Post

All

Last 24 Hours

Last 7 Days

Last 14 Days

Last 30 Days

Salary Range

From: To:

Results:

- 1. Hồ Chí Minh Full Time Language Male Công ty Áo
- 2. Hà Nội Part Time IT Female Công ty Áo
- 3. Công ty Áo
- 4. Công ty Áo
- 5. Công ty Áo
- 6. Công ty Áo
- 7. Công ty Áo
- 8. Công ty Áo

< 1 2 >

Hình 70: Search page

Showing 1-10 of 12 posts

Create Date ASC

Người dùng có thể sắp xếp các bài viết theo thứ tự ngày đăng.

Hồ Chí Minh Full Time Language Male
Công ty Ảo

Nếu người dùng đăng nhập với tư cách là ứng viên sẽ có thêm chức năng bookmark bài viết. Khi người dùng click vào bài đăng, bài đăng sẽ mở rộng ra, cho thấy các thông tin chi tiết: chuyên ngành, loại công việc, vị trí, mức lương.

Hồ Chí Minh Full Time Language Male
Công ty Ảo

Major: IT

Job Type: Full Time

Location: Phường Tân Định, Quận 1, Thành phố Hồ Chí Minh

Salary: 0 - 10.000.000 đ

Nếu click vào nút “View Job” thì sẽ chuyển qua trang job detail

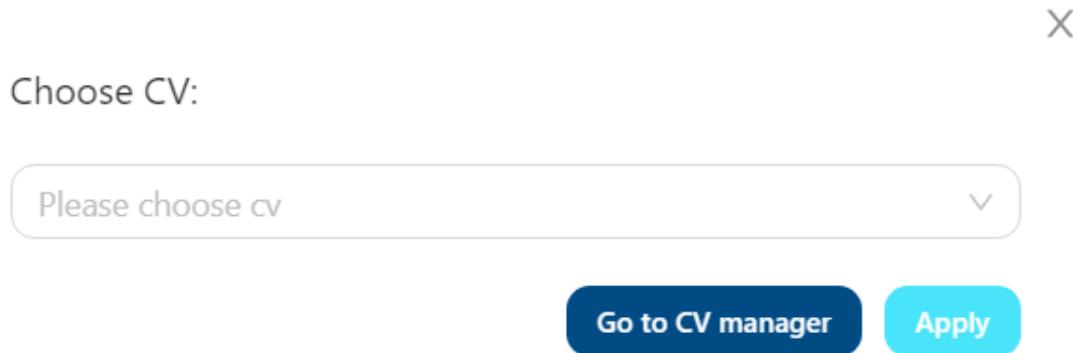
e. Job Detail

Trang Job Detail sẽ hiển thị tất cả thông tin của bài đăng đó, bên trên là banner của bài đăng, bên trái sẽ có nút apply này sẽ thành applied nếu người dùng là ứng viên và đã ứng tuyển vào nút sẽ bị vô hiệu. Nếu người dùng chưa đăng nhập sẽ hiện thông báo yêu cầu đăng nhập với tài khoản ứng viên

Please sign in with candidate account

Sign in Sign up

Trong trường hợp người dùng đã đăng nhập với tài khoản ứng tuyển thì sẽ cho chọn cv để apply



Bên dưới là thông tin của bài đăng, bên trái là nội dung của bài đăng mà nhà tuyển dụng đã viết, bên phải là thông tin gồm:

- Của bài đăng:
 - Ngày đăng
 - Vị trí
 - Loại công việc
 - Chuyên ngành
 - Mức lương
 - Giới tính
- Của công ty đăng bài:
 - Tên công ty
 - Số điện thoại
 - Email
 - Vị trí



Quasi vero, inquit, perpetua oratio rhetorum solum, non etiam philosophorum sit.

 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc omni virtuti vitium contrario nomine opponitur. Num igitur eum postea censes anxio animo aut sollicito fuisse? Sed quid attinet de rebus tam aperitis plura requirere? *Duo Reges: constructio interrete.* Etenim semper illud extra est, quod arte comprehenditur.

Quod autem principium officii querunt, melius quam Pyrrho; Age, inquies, ista parva sunt. Hanc ergo intuens debet institutum illud quasi signum absolvere. Et quod est manus, quod opus sapientiae?

 Sin tantum modo ad indicia veteris memoriae cognoscenda, curiosorum. Vitae autem degendae ratio maxime quidem illis placuit quieta. Recte dicis: *omnia contraria, quo etiam insanos esse vultis.*

 Restatis igitur vos:

 Qui enim existimabit posse se miserum esse beatus non erit.

 Præteritis, inquit, gaudeo.

 Ita fit beatæ vitae domina fortuna, quam Epicurus ait exiguum intervenire sapienti.

 Magna lous.

 Si longus, levis dictata sunt.

 Scrupulum, inquam, abeunt;

 Sumnum enim bonum expositum vacuitatem doloris;

 Quo modo?

 Serpere anguiculos, nare anaticulas, evolare merulas, cornibus uti videmus boves, nepas aculeis.

 Age sane, inquam,

 Iubet igitur nos Pythius Apollo noscere nosmet ipsos.

 1. Quasi ego id curem, quid illi aiat aut neget.

 2. Quodsi ipsum honestatem undique perfectam atque absolutam.

 3. Quid est igitur, cur ita semper deum appetet Epicurus beatum et aeternum?

 4. Sed haec ab Antiocho, familiari nostro, dicuntur multo melius et fortius, quam a Stasea dicebantur.

 5. Iam id ipsum absurdum; maximum malum neglegi.

 6. Te ipsum, dignissimum maioribus tuis, voluptasne induxit, ut adolescentulus eriperes P.

 7. Vide, quantum, inquam, fallere, Torquate.

- Animi enim quoque dolores percipiet omnibus partibus maiores quam corporis.

- Euro, inquit adirende; iniquum, haec quidem de re;

- Non enim, si omnia non sequebatur, idcirco non erat ortus illinc.

- Quorū sine causa fieri nihil putandum est.

- Et nunc quidem quod eam tuetur, ut de vite potissimum loquar, est id extrinsecus;

- Sed tu istuc dixi bene Latine, parum plane.

Atqui si, ut convenire debet inter nos, est quedam appetitio naturalis ea, quae secundum naturam sunt, appetens, eorum omnium est aliquae summa facienda.

Tollit *definitiones*, nihil de dividendo ac partiendo docet, non quo modo efficiatur concludaturque ratio tradit, non qua via captiosa solvantur ambigua distinguuntur ostendit;

Nisi enim id faceret, cur Plato Aegyptum peragravit, ut a sarcodibus barbaris numeros et caelestia acciperet?

Ab his oratores, ab his imperatores ac rerum publicarum principes extiterunt. *Mihi enim erit isdem istis fortasse iam utendum.* Aut unde est hoc contritum vetustate proverbum: *quicum in tenebris?* Neque enim disputari sine reprehensione nec cum iracundia aut pertinacia recte disputari potest. *Sed ego in hoc resisto:* Quod autem ratione actum est, id officium appellamus. Omnes enim iucundum motum, quo sensus hilaretur. *Invidiosum nomen est, infame, suspectum.* Primum quid tu dicas breve? Non queritur autem quid naturae tuae consentaneum sit, sed quid disciplinae. Hoc ipsum elegantius ponи meliusque potuit. Quae in controversiam veniunt, de iis, si placet, disseramus.

Job Overview

Date Posted:
30/11/2022

Location:
141/80 Tâm Danh, Phường Phúc Xá, Quận Ba Đình, Thành phố Hà Nội

Job Type:
Part Time

Job Major:
IT

Salary:
0 ₫-21.000.000 ₫

Gender:
2

Company Overview

Name: Công ty Áo

Phone: 0776715590

Email: nghuuutandat@gmail.com

Location: 141/80 Tâm Danh

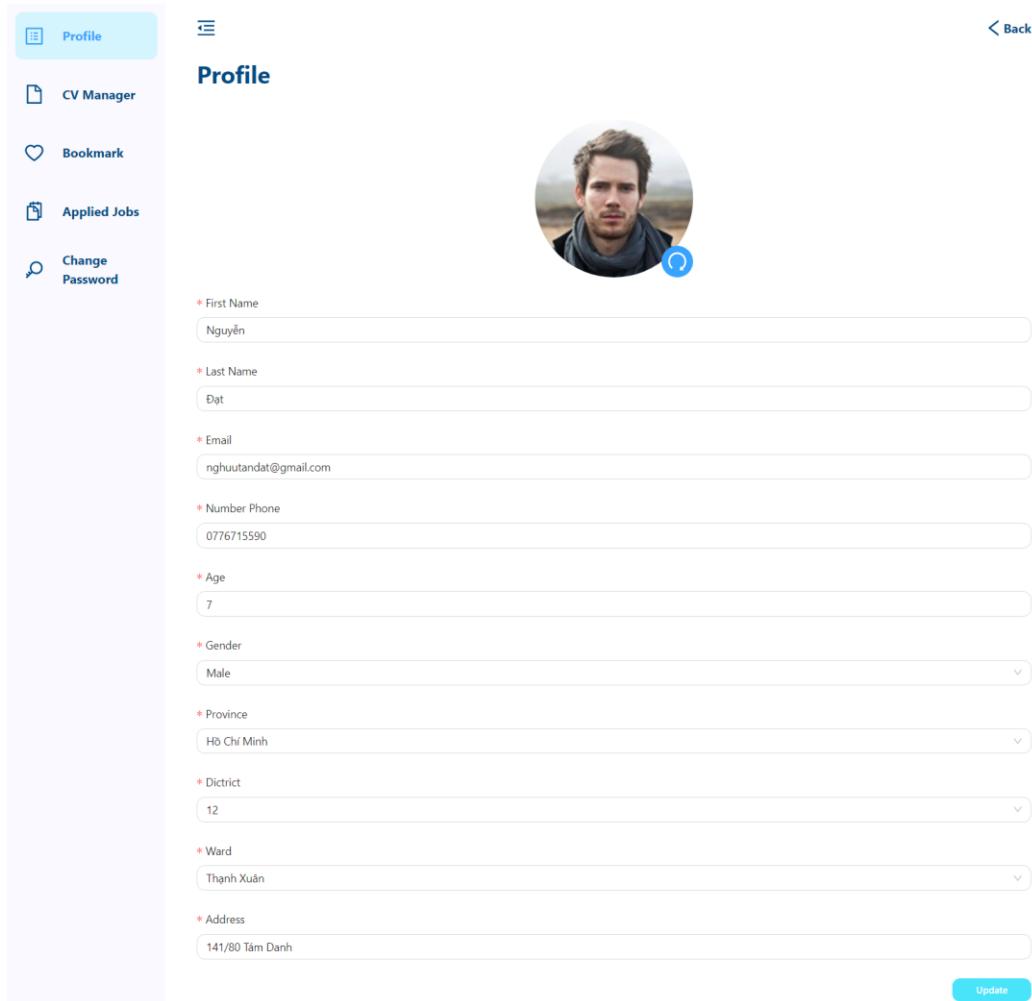
Hình 71: Job Detail

f. Candidate

Bên trái là sidebar dùng để di chuyển giữa các trang con trong trang Candidate

- Profile

Mặc định khi vào trang Candidate sẽ là trang Profile ở trang này sẽ hiện lên các thông tin của người dùng, người dùng có thể thay đổi avatar hoặc các thông tin khác.



The screenshot shows the 'Profile' section of a candidate's account. On the left is a sidebar with links: Profile (selected), CV Manager, Bookmark, Applied Jobs, and Change Password. The main area has a title 'Profile' and a circular placeholder for the user's profile picture, which is currently empty. Below the placeholder are several input fields with validation messages:

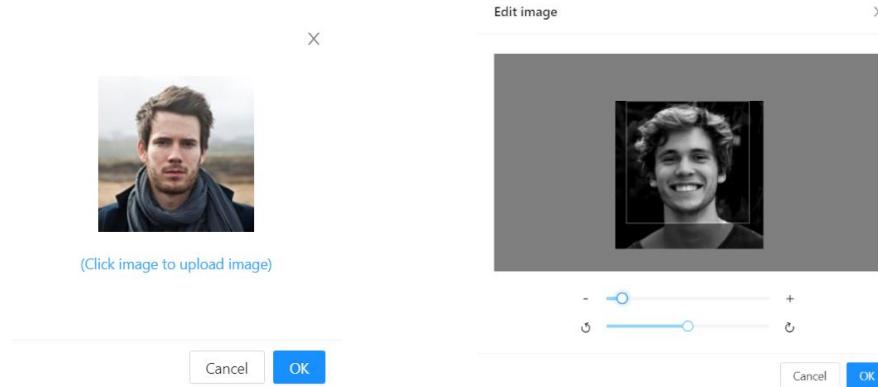
- * First Name: Nguyễn
- * Last Name: Đạt
- * Email: nghuutandat@gmail.com
- * Number Phone: 0776715590
- * Age: 7
- * Gender: Male
- * Province: Hồ Chí Minh
- * District: 12
- * Ward: Thành Xuân
- * Address: 141/80 Tâm Danh

A blue 'Update' button is located at the bottom right of the form.

Hình 72: Candidate – Profile

Khi bấm vào thay đổi avatar sẽ hiện lên modal, khi người dùng click vào avatar hiện tại sẽ được chọn 1 file hình để upload lên, và chỉ được upload hình. Sau khi upload

hình người dùng sẽ được đưa đến trang chỉnh sửa, người dùng có thể cắt hoặc xoay hình, sau đó người dùng có thể tiếp tục bấm Ok để tiếp tục lưu avatar



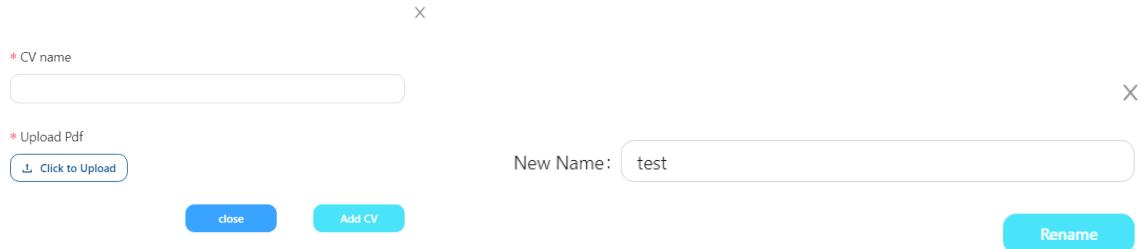
Sau khi chỉnh sửa các thông tin người dùng có thể bấm update để cập nhật thông tin.

- CV manager

Hình 73: Candidate – CV Manager

Ở trang CV manager người dùng có thể thêm, đổi tên, xóa, xem, tải xuống các CV

Khi bấm “Add cv” sẽ hiện modal cho phép người dùng nhập tên, và chọn file pdf để tạo 1 cv



Nếu người dùng chọn Rename, sẽ hiện một modal để người dùng nhập tên mới

Nếu người dùng chọn View, sẽ hiện lên modal với nội dung của file pdf đó

Nếu người dùng chọn delete thì sẽ xóa CV

Nếu người dùng chọn Download, file sẽ được tự động download về máy của người dùng

- Bookmarks

Trang bookmarks sẽ thể hiện các post mà người dùng đã đánh dấu. Người dùng có thể xóa bookmark hoặc xem bài viết, khi bấm vào xem bài viết người dù sẽ được đưa đến trang post detail của bài đăng đó

The screenshot shows a sidebar with navigation options: Profile, CV Manager, Bookmark (selected), Applied Jobs, and Change Password. The main content area is titled 'Bookmark Manager' and displays a table of bookmarked posts. The columns are Post Title, Create Date, and Action. There are seven rows, each with a post title (2, 3, 4, 5, 6, 7) and a creation date (21/12/2022 or 3/1/2023). Each row has two action buttons: a pink heart icon and a blue circular icon with a '@' symbol.

Post Title	Create Date	Action
2	21/12/2022	
3	21/12/2022	
4	21/12/2022	
5	21/12/2022	
6	3/1/2023	
7	3/1/2023	

Hình 74: Candidate – Bookmarks

- Applied Jobs

The screenshot shows a sidebar with navigation options: Profile, CV Manager, Bookmark, Applied Jobs (selected), and Change Password. The main content area is titled 'Applied Job' and displays a table of applied posts. The columns are Post, CV, Create Date, and Action. There is one row for 'Hồ Chí Minh Full Time Language Male' with a CV labeled 'test' and a creation date of '30/11/2022'. The action column contains a blue square icon with a white '@' symbol. Navigation arrows for the table are visible at the bottom right.

Post	CV	Create Date	Action
Hồ Chí Minh Full Time Language Male	test	30/11/2022	

Hình 75: Candidate – Applied Job

Ở trang này sẽ hiện những bài post mà người dùng đã apply và cv họ đã apply, họ có thể xóa để hủy apply vào bài đăng đó

- Change passwords

Người dùng có thể thay đổi passwords ở đây, sau khi change passwords sẽ gửi email thông báo về mail của user

Change Password

* Old Password

* Password

* Confirm Password

Submit

Hình 76: Candidate – Change passwords

g. Recruiter

- Profile

Profile

* Company Name
Công ty Áo

* Email
nghuurandat@gmail.com

* Numberphone
0776715590

* Province
Hồ Chí Minh

* District
1

* Ward
Tân Định

* Address
141/80 Tân Định

Update

Hình 77: Recruiter - Profile

Ở đây người dùng có thể thay đổi thông tin của công ty và logo của công ty

- Post Manager

The screenshot shows a 'Post Manager' section within a recruitment software. On the left is a sidebar with 'Profile', 'Post Manager' (which is selected and highlighted in blue), 'All Applicants', and 'Change Password'. The main area is titled 'Post Manager' with a back arrow and a 'Add new job' button. It displays a table of job posts:

Post Title	Create Date	Active	Action
Hồ Chí Minh Full Time Language Male	30/11/2022	<input checked="" type="checkbox"/>	
Hà Nội Part Time IT Female	30/11/2022	<input checked="" type="checkbox"/>	
1	1/12/2022	<input checked="" type="checkbox"/>	
2	1/12/2022	<input checked="" type="checkbox"/>	
3	1/12/2022	<input checked="" type="checkbox"/>	

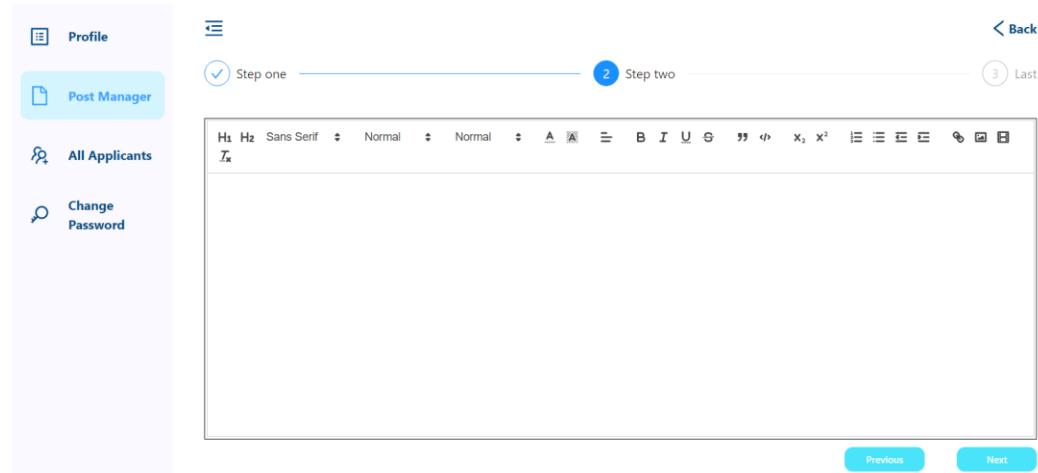
Hình 78: Recruiter – Post Manager

Ở trang post manager người dùng có thể xem các bài đăng hoặc thêm xóa sửa các bài đăng. Khi bấm vào add new job người dùng sẽ được đưa đến trang thêm bài đăng mới.

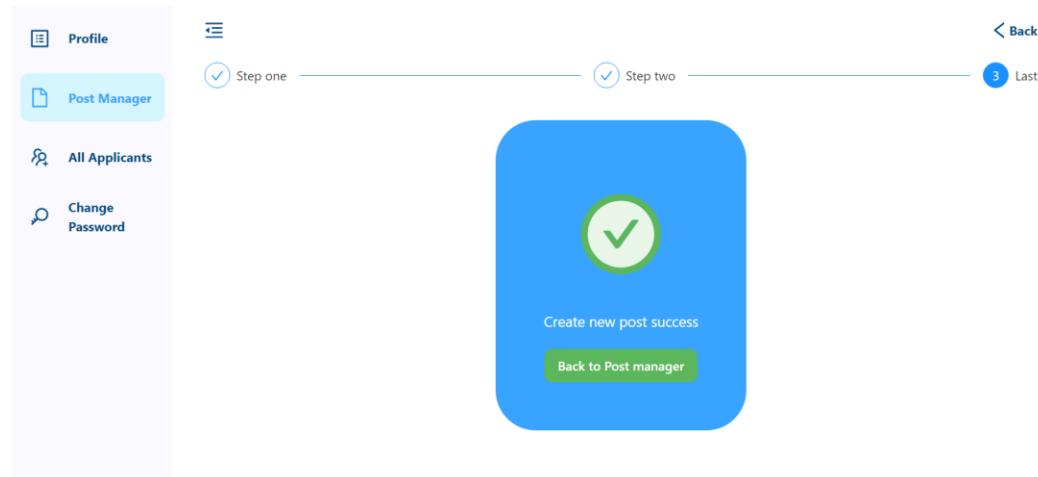
The screenshot shows the 'Step one' of a 'New post' wizard. The sidebar includes 'Profile', 'Post Manager' (selected), 'All Applicants', and 'Change Password'. The main area has three tabs: 'Step one' (selected), 'Step two', and 'Last'. The 'Step one' tab contains the following fields:

- Title: Input field
- Job Type: Select dropdown with placeholder 'Please select job type'
- Major: Select dropdown with placeholder 'Please select majors'
- Gender: Select dropdown with placeholder 'Please select gender'
- Salary: Range slider from 0đ to 50,000,000đ with current value at 0đ
- Province: Select dropdown with placeholder 'Province'
- District: Select dropdown with placeholder 'District'
- Ward: Select dropdown with placeholder 'Ward'
- Address: Input field

Hình 79: Recruiter – New post step one



Hình 80: Recruiter – New post step two



Hình 81: Recruiter – New post step three

Trang tạo bài đăng mới có 3 bước:

- Bước 1: Điền các thông tin thuộc tính của bài đăng
- Bước 2: Thêm nội dung của bài đăng, có hỗ trợ richtext editor
- Bước 3: Chờ kết quả trả về kết quả có thành công hay không

Người dùng còn có thể chỉnh sửa các bài post đã đăng bằng cách bấm vào nút edit

The screenshot shows a user interface for managing job posts. On the left, there's a sidebar with navigation links: Profile, Post Manager (which is active, indicated by a blue background), All Applicants, Change Password, and a search bar. The main area contains fields for a job listing:

- Title:** Hồ Chí Minh Full Time Language Male
- Job Type:** Full Time
- Major:** IT
- Gender:** Male
- Salary:** A slider scale from 0đ to 50.000.000đ, with a midpoint at 10.000.000đ.
- Province:** Hồ Chí Minh
- District:** 1
- Ward:** Tân Bình
- Address:** 141/80 Tân Bình
- Content:** A rich text editor window containing Latin text. The text discusses the nature of work and the author's views on it, mentioning Epicurus and other philosophers.

At the bottom right of the content area is a blue "Update" button.

Hình 82: Recruiter – Edit Post

Sau khi người dùng sửa đổi các thông tin của bài post người dùng có thể nhấn nút update để cập nhật lại các bài đăng. Người dùng còn có thể ẩn bài đăng khỏi việc tìm kiếm hoặc xóa bài đăng

- All Applicant

Hình 83: Recruiter – All Applicant

Trang All Applicant sẽ hiện danh sách các ứng viên đã ứng tuyển vào các bài đăng. Người dùng có thể xem hoặc tải CV được ứng tuyển về

- Change Password

Hình 84: Recruiter – Change Password

Người dùng có thể đổi passwords và sẽ có email thông báo sau khi đổi passwords

h. Admin

- Account Manager

Ở trang này Admin có thể thấy danh sách các tài khoản của người dùng, loại người dùng cũng như ngày tạo. Khi bấm vào View thì admin sẽ được chuyển sang trang account detail. Tùy theo loại người dùng trang account detail sẽ hiện các thông tin khác nhau

The screenshot shows the 'Account Manager' section of a software application. On the left, there is a sidebar with icons and labels: 'All Post', 'User Type', 'Job Type', and 'Majors'. The main area is titled 'Account Manager' and contains a table with the following data:

Username	User Type	Create Date	Action
admin	Admin	30/11/2022	
candidate	Candidate	30/11/2022	
recruiter	Recruiter	30/11/2022	
51702075@student.tdtu.edu.vn	Candidate	3/12/2022	

At the bottom right of the table, there are navigation buttons: '<', '1', and '>'.

Hình 85: Admin – Account Manager

Trang Account Detail bao gồm 2 phần:

Phần đầu sẽ hiển thị các thông tin cơ bản của tài khoản.

- Nếu loại tài khoản là candidate, sẽ hiển thị các thông tin của người dùng như tên, tuổi, giới tính,
- Nếu loại là người dùng là recruiter thì sẽ hiển thị các thông tin của công ty.

Phần hai của trang account sẽ bao gồm các data của người dùng đó:

- Loại người dùng là candidate thì sẽ hiển thị các bảng thông tin: CV, Applied Jobs, Bookmark, admin không có quyền chỉnh sửa những thông tin này.
- Loại người dùng là recruiter thì sẽ hiển thị các bảng thông tin: các bài đăng mà tài khoản này đã tạo, các ứng viên applied vào các bài đăng của tài khoản này. admin không có quyền chỉnh sửa những thông tin này.

Nếu loại người dùng là Admin sẽ không thể xem xem chi tiết account

Information

Id:	3
Username:	recruiter
Age:	7
First Name:	Nguyễn
Last Name:	Đạt
Email:	nghuutandat@gmail.com
Gender:	Male
Numberphone:	0776715590
Province:	Hồ Chí Minh
District:	Full Time
Ward:	Language

Data

Post

Post Title	Create Date	Active
Hồ Chí Minh Full Time Language Male	30/11/2022	<input checked="" type="checkbox"/>
Hà Nội Part Time IT Female	30/11/2022	<input checked="" type="checkbox"/>

All Applicant

Post	CV	Create Date
Hồ Chí Minh Full Time Language Male	test	30/11/2022

Hình 86: Admin - Account Detail – Recruiter

Information

Id:	2
Username:	candidate
Age:	7
First Name:	Nguyễn
Last Name:	Đạt
Email:	nghuutandat@gmail.com
Gender:	Male
Numberphone:	0776715590
Province:	Hồ Chí Minh
District:	Full Time
Ward:	Language

Data

CV

CV Name	Create Date
test	30/11/2022
tmp	4/1/2023

Applied Jobs

Post	CV	Create Date
Hồ Chí Minh Full Time Language Male	test	30/11/2022

Bookmark Post

Post Title	Create Date
Hà Nội Part Time IT Female	3/1/2023
Hồ Chí Minh Full Time Language Male	3/1/2023

Hình 87: Admin - Account Detail – Candidate

- All Post

Trang All Post sẽ hiện tất cả các bài post, kể cả các bài đã được ẩn đi, khi admin nhấp vào view sẽ dẫn tới trang detail tail post hiện các thông tin chi tiết của bài post

Post Title	Company	Create Date	Action
Hồ Chí Minh Full Time Language Male	Công ty Áo	30/11/2022	
Hà Nội Part Time IT Female	Công ty Áo	30/11/2022	
1	Công ty Áo	1/12/2022	
2	Công ty Áo	1/12/2022	
3	Công ty Áo	1/12/2022	
4	Công ty Áo	1/12/2022	

Hình 88: Admin – All Post

- Account Type Manager

Trang account type manager hiện 3 loại người dùng, admin có thể rename các loại người dùng nhưng không thể thêm hoặc xóa.

Type Name	Create Date	Action
Admin	1/1/1970	
Candidate	1/1/1970	
Recruiter	1/1/1970	

Hình 89: Admin – Account Type Manager

- Job Type Manager

Trang job type manager hiện danh sách các loại công việc, admin có thể thêm, xóa, hoặc chỉnh sửa.

The screenshot shows a user interface titled "Job Type Manager". On the left, there is a sidebar with icons for Account Manager, All Post, User Type, Job Type (which is selected and highlighted in blue), and Majors. The main area displays a table with columns for Type Name, Create Date, and Action. Two entries are listed: "Full Time" (Create Date: 30/11/2022) and "Part Time" (Create Date: 30/11/2022). Each entry has edit and delete icons in the Action column. Navigation arrows and a page number indicator (1) are at the bottom right.

Type Name	Create Date	Action
Full Time	30/11/2022	
Part Time	30/11/2022	

Hình 90: Admin - Job Type Manager

- Major Manager

Trang major manager sẽ hiển danh sách các chuyên ngành, admin có thể thêm, xóa chỉnh sửa các chuyên ngành

The screenshot shows a user interface titled "Major Manager". On the left, there is a sidebar with icons for Account Manager, All Post, User Type, Job Type, and Majors (which is selected and highlighted in blue). The main area displays a table with columns for Type Name, Create Date, and Action. Three entries are listed: "IT" (Create Date: 30/11/2022), "Language" (Create Date: 30/11/2022), and "Pharmacy" (Create Date: 30/11/2022). Each entry has edit and delete icons in the Action column. Navigation arrows and a page number indicator (1) are at the bottom right.

Type Name	Create Date	Action
IT	30/11/2022	
Language	30/11/2022	
Pharmacy	30/11/2022	

2. Kết luận

Sau quá trình tìm hiểu cũng như ứng dụng một số công nghệ vào việc thực hiện đề tài xây dựng trang web đăng tin tuyển dụng này, chúng em rút ra cho mình một số bài học như sau:

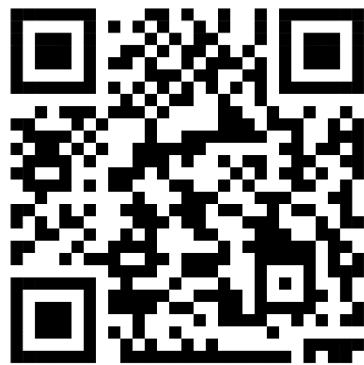
- Tìm hiểu quá trình hoạt động của một trang web bao gồm cả phía giao tiếp với cơ sở dữ liệu (API) và phía giành cho người dùng phổ thông sử dụng là (UI)
- Hiểu thêm 1 phần nhỏ về ngôn ngữ mà mình sử dụng .
- Tìm hiểu về thêm về ánh xạ đối tượng vào cơ sở dữ liệu thay cho việc truy suất dữ liệu thông qua việc viết các câu truy vấn trên nền SQL.
- Tìm hiểu các kiến thức cơ bản về một trong các công cụ hỗ trợ các nhà phát triển website có thể hoạt động ở nhiều môi trường khác nhau là docker.
- Hiểu thêm về như thế nào là một Images , Container , hay các câu lệnh thao tác với docker để chạy một trang ứng dụng có đầy đủ các tài nguyên mà không cần cài đặt quá nhiều.
- Tìm hiểu và xây dựng một bộ CI/CD tự động build mã nguồn và deploy chúng lên server thật.
- Tìm hiểu và thao tác với máy tính server với hệ điều hành Linux bằng các dòng lệnh ở cmd.
- Biết được một số kiến thức cơ bản về việc đưa dự án lên một con server được thuê và cũng như thêm tên miền gắn với trang web của nhóm.
- Sử dụng các công cụ hỗ trợ UI như antd ,.... Đọc tài liệu của các thư viện và ứng dụng nó vào trong dự án.
- Tiếp xúc với một ngôn ngữ mới là Typescript là một dạng nâng cao của JS và đòi hỏi phải viết code một cách chặt chẽ hạn chế lỗi xảy ra.
- Sử dụng React Lazy để tách các component, giúp cho trang web được load nhanh hơn cải thiện trải nghiệm người dùng.

- Sử dụng Redux để quản lý các state của React. Sử dụng react-router-dom để định tuyến phía client.
- Sử dụng google developer console để đăng ký client id dùng trong việc đăng nhập với google.

Tuy nhiên bên cách đó còn một số vấn đề và hạn chế như:

- Do thuê máy chủ server trên mạng nên sẽ phụ thuộc rất nhiều vào tốc độ mạng của server cũng như mạng của người dùng.
- Chỉ mới có một số kiến thức cơ bản về server nên sẽ dễ dẫn đến các tình huống có quá nhiều yêu cầu truy cập từ nhiều . Cân bằng tải cho server .
- Chưa áp dụng được kĩ thuật CI/CD runner vào trong dự án để không phụ thuộc vào docker hub.
- Đối với công khai web thì không đủ kinh phí để mua chứng chỉ SSL để nâng trang web lên https.
- Do chưa có kinh nghiệm về thiết kế UX (User Experience) và UI (User Interface) không được tốt.
- Chưa tối ưu hiệu xuất cho Front End.

Chúng em rất mong nhận được sự góp ý của quý Thầy , Cô và các bạn để có thể xem xét lại cũng như trau dồi thêm, tiếp tục tìm hiểu ứng dụng và hoàn thiện những nội dung chưa được tốt của đề tài này.



jobportaltdtu.com

Hình 91 Link đến Website <http://jobportaltdtu.com/>

TÀI LIỆU THAM KHẢO

Tài liệu bao gồm :

- [1] D. K. Toan, 30 5 2017. [Online]. Available: <https://viblo.asia/p/gioi-thieu-ve-reactjs-phan-i-cac-khai-niem-co-ban-V3m5WzjblO7>.
- [2] [Online]. Available: <https://reactjs.org/>.
- [3] [Online]. Available: https://www.w3schools.com/react/react_jsx.asp.
- [4] n. c. thanh, "Tổng quan về JSX," 28 6 2017. [Online]. Available: <https://viblo.asia/p/tong-quan-ve-jsx-Qbq5QqBL5D8>.
- [5] "TypeScript Documentation," [Online]. Available: <https://www.typescriptlang.org/docs/>.
- [6] B. N. Dan, "Type vs Interface trong typescript," 16 1 2020. [Online]. Available: <https://viblo.asia/p/type-vs-interface-trong-typescript-gJ599Gp5X2>.
- [7] "Usage Guides," [Online]. Available: <https://redux.js.org/usage/>.
- [8] "Ant Design of React," [Online]. Available: <https://4x.ant.design/docs/react/introduce>.
- [9] "Usage Guide," [Online]. Available: <https://redux-toolkit.js.org/usage/usage-guide>.
- [10] T. Blog, "Expressjs là gì? Tất tần tật về Express.js," TopDev, [Online]. Available: <https://topdev.vn/blog/express-js-la-gi/>.
- [11] P. Data, "Prisma schema," [Online]. Available: <https://www.prisma.io/docs/concepts/components/prisma-schema>.
- [12] L. Nam, "SQL Server là gì? cài đặt SQL server ra sao?," [Online]. Available: <https://longvan.net/sql-server-la-gi.html>.
- [13] "VPS LÀ GÌ? VPS ĐƯỢC DÙNG ĐỂ LÀM GÌ?," 2016. [Online]. Available: <http://fit.tdc.edu.vn/blog/2016/04/vps-la-gi-vps-c-dung--lam-gi>.
- [14] "ORM là gì? Tổng quan về ORM Framework," 21 11 2017. [Online]. Available: <https://stackjava.com/uncategorized/orm-la-gi-tong-quan-ve-orm-framework.html>.
- [15] "ORM là gì? Tổng quan về ORM Framework," 09 08 2021. [Online]. Available: <https://hocspringboot.net/2021/08/09/orm-la-gi-tong-quan-ve-orm-framework-2/>.
- [16] T. Q. Chung, ""ORM" LÀ GÌ MÀ NHÀ NHÀ SỬ DỤNG?," 20 02 2021. [Online]. Available: <https://www.goccuachung.com/orm-la-gi-ma-nha-nha-su-dung/>.
- [17] N. M. Dinh, "Node.js Tutorial: Phần 4: Express framework," 26 02 2017. [Online]. Available: <https://viblo.asia/p/nodejs-tutorial-phan-4-express-framework-924IJXpNKPM>.
- [18] N. Hung, "NodeJS là gì? Tổng quan kiến thức về NodeJS," vietnix, 25 04 2022. [Online]. Available: <https://vietnix.vn/nodejs-la-gi/>.
- [19] N. Hung, "VPS là gì? Virtual Private Server dùng để làm gì?," 14 01 2022. [Online]. Available: <https://vietnix.vn/vps-la-gi/>.

- [20] H. G, "VPS là gì? Tất cả các điều cần biết về Virtual Private Server," 29 08 2022. [Online]. Available: <https://www.hostinger.vn/huong-dan/vps-la-gi-tat-ca-cac-dieu-can-biet-ve-may-chu-ao>.
- [21] tenten.vn, "VPS là gì? 6 điều cần biết về Virtual Private Server," 07 04 2022. [Online]. Available: <https://tenten.vn/tin-tuc/vps-la-gi/>.
- [22] Docker, "Docker overview," [Online]. Available: <https://docs.docker.com/get-started/overview/#docker-architecture>.
- [23] T. Blog, "Docker là gì? Tìm hiểu về Docker," [Online]. Available: <https://topdev.vn/blog/docker-la-gi/>.
- [24] TEL4VN, "GIỚI THIỆU TỔNG QUAN VỀ KIẾN TRÚC CỦA DOCKER," 01 01 2021. [Online]. Available: <https://tel4vn.edu.vn/blog/gioi-thieu-tong-quan-ve-kien-truc-cua-docker/>.
- [25] Docker, "Continuous integration with Docker," [Online]. Available: <https://docs.docker.com/build/ci/>.
- [26] Teky, "CI/CD là gì? Thông tin chi tiết về chương trình CI và CD," 26 05 2022. [Online]. Available: <https://teky.edu.vn/blog/ci-cd-la-gi/>.
- [27] ITNavi, "CI/CD là gì? Những lợi ích mà mô hình CI/CD mang lại," 20 07 2021. [Online]. Available: <https://itnavi.com.vn/blog/ci-cd-la-gi/>.
- [28] T. Blog, "Triển khai CI/CD với Gitlab," [Online]. Available: <https://topdev.vn/blog/trien-khai-ci-cd-voi-gitlab/>.
- [29] N. @nghiadd, "CI, CD và ... DevOps ???," 26 07 2017. [Online]. Available: <https://viblo.asia/p/ci-cd-va-devops-07LKXYXDZV4>.
- [30] T. Blog, "NGINX là gì? Tổng quan về NGINX," [Online]. Available: <https://topdev.vn/blog/nginx-la-gi/>.
- [31] N. Hưng, "NGINX là gì? NGINX có thể làm được gì?," 19 02 2021. [Online]. Available: <https://vietnix.vn/nginx-la-gi/>.
- [32] M. Hoang, "Tìm hiểu tổng quan về Nginx," 30 07 2021. [Online]. Available: <https://viblo.asia/p/tim-hieu-tong-quan-ve-nginx-63vKjOExZ2R>.
- [33] D. Erin Glass (Senior Manager, "How To Install Nginx on Ubuntu 20.04," 25 04 2020. [Online]. Available: <https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-20-04>.
- [34] J. E. (. a. a. a. DigitalOcean), "How To Set Up Nginx Server Blocks (Virtual Hosts) on Ubuntu 16.04," 20 05 2016. [Online]. Available: <https://www.digitalocean.com/community/tutorials/how-to-set-up-nginx-server-blocks-virtual-hosts-on-ubuntu-16-04>.
- [35] K. Y. (. a. a. a. DigitalOcean), "How To Run Nginx in a Docker Container on Ubuntu 22.04," 28 10 2022. [Online]. Available: <https://www.digitalocean.com/community/tutorials/how-to-run-nginx-in-a-docker-container-on-ubuntu-22-04>.

- [36] S. (. a. a. a. DigitalOcean), "How To Deploy a Go Web Application with Docker and Nginx on Ubuntu 22.04," 19 11 2022. [Online]. Available: <https://www.digitalocean.com/community/tutorials/how-to-deploy-a-go-web-application-with-docker-and-nginx-on-ubuntu-22-04>.
- [37] "Deploy node application whit docker and nginx," 17 07 2022. [Online]. Available: <https://www.digitalocean.com/community/questions/deploy-node-application-whit-docker-and-nginx>.