ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Ngọc Minh

ỨNG DỤNG QUẢN LÝ NHÂN VIÊN SỬ DỤNG NHẬN DIỆN KHUÔN MẶT ĐỂ CHẤM CÔNG

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ thông tin

HÀ NỘI – 2022

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Ngọc Minh

ỨNG DỤNG QUẢN LÝ NHÂN VIÊN SỬ DỤNG NHẬN DIỆN KHUÔN MẶT ĐỂ CHẤM CÔNG

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ thông tin

Cán bộ hướng dẫn: TS. Dương Lê Minh

HÀ NỘI - 2022

VIETNAM NATIONAL UNIVERSITY, HANOI

UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Nguyen Ngoc Minh

STAFF MANAGEMENT APPLICATION USING FACE RECOGNITION IN CHECK-IN SYSTEM

THE BS THESIS

Major: Information Technology

Supervisor: PhD. Duong Le Minh

HÀ NỘI – 2022

LỜI CẢM ƠN

Đầu tiên, tôi xin gửi lời cám ơn chân thành tới TS. Dương Lê Minh – Phó Chủ Nhiệm khoa Mạng máy tính và Truyền thông dữ liệu – người đã hướng dẫn tận tình, tỉ mỉ, chu đáo tôi trong suốt quá trình làm đồ án tốt nghiệp. Quãng thời gian được thầy hướng dẫn đã giúp tôi học hỏi, đúc kết được nhiều kinh nghiệm về phương pháp nghiên cứu, kĩ năng giao tiếp, kĩ năng làm việc nhóm, kĩ năng trình bày. Thầy còn truyền cho tôi ngọn lửa yêu nghiên cứu khoa học, niềm tin vượt qua những khó khăn trong cuộc sống và dạy tôi cách vượt qua những khó khăn đó. Tôi cảm thấy tự hào và may mắn khi là một sinh viên được thầy hướng dẫn trong những năm tháng đại học.

Ngoài ra, tôi xin gửi lời cám ơn chân thành đến những người bạn đã giúp đỡ tôi bằng hành động, bằng lời nói mỗi khi tôi gặp khó khăn, thất bại. Bốn năm bên nhau không phải là dài nhưng đối với tôi, đây là quãng thời gian tuyệt vời nhất và không thể nào quên.

Tiếp theo, tôi xin gửi lời cảm ơn đến các thầy cô giảng viên Trường Đại học Công Nghệ - Đại học Quốc Gia Hà Nội – những người đã tận tâm truyền đạt những kiến thức quý báu làm nền tảng để tôi tiếp tục đi xa hơn nữa trong lĩnh vực công nghệ thông tin.

Cuối cùng, tôi xin được cảm ơn gia đình đã nuôi tôi khôn lớn để trở thành người có ích cho xã hội, giúp tôi có một điểm tựa vững chắc để yên tâm học hành trong suốt bao năm qua. Tôi xin gửi lời cám ơn chân thành tới cha, mẹ, em trai đã luôn động viên và cổ vũ tôi mỗi khi tôi gặp khó khăn và thử thách.

TÓM TẮT

Tóm tắt: Đồ án trình bày về ứng dụng quản lý nhân viên, sử dụng công nghệ nhân diện khuôn mặt trong chấm công. Hệ thống ứng dụng Web được xây dựng bằng .Net Core và Angular2, có khả năng hoạt động rất bền bỉ và mạnh mẽ. Công nghệ nhận diện khuôn mặt áp dụng 2 công nghệ là MTCNN trong phát hiện khuôn mặt và model FaceNet nhằm training cho máy học nhận diện khuôn mặt. Trong bối cảnh các công ty ngày có lượng nhân viên ngày càng tăng lên, lượng công việc để đảm bảo việc quản lý quá trình đi làm của từng nhân viên rất lớn, cần một ứng dụng để quản lý dữ liệu đi làm của nhân viên một cách hiệu quả. Ngoài ra, khi mà tác động từ Covid-19 vẫn còn, và có thể trong tương lai sẽ có thể xảy ra các vấn đề tương tự, thì việc nhân viên khi điểm danh phải ký giấy, hoặc phải sử dụng công nghệ vân tay – vốn cần nhân viên phải tiếp xúc trực tiếp với máy đều được hạn chế, thay vào đó chúng ta đang áp dụng công nghệ nhận diện khuôn mặt để giải quyết các vấn đề trên tốt hơn. Để giải quyết 2 vấn đề được nêu trên, ý tưởng về "Ứng dụng quản lý nhân viên, sử dụng công nghệ nhận diện khuôn mặt trong chấm công" cần được nghiên cứu và thực hiện. Từ đó các công ty sẽ có thêm sự lựa chọn để nâng cao việc quản lý giờ giấc đi làm của nhân viên mình tốt hơn, an toàn hơn trong thời buổi dịch bệnh như hiện nay.

Từ khóa: *Quản lý nhân viên, .Net Core, Angular2, MTCNN, FaceNet*

ABSTRACT

Abstract: The thesis proposes an approach for presenting staff management application, using face recognition in check-in system. The Web application system is developed using .Net Core and Angular2, which is very durable and powerful. Its face recognition technology applies two technologies: MTCNN in face detection and FaceNet model to train machine learning to recognize faces. In the context of companies with an increasing number of employees, the amount of work to ensure the management of each employee's commute is very large, an application is needed to manage employee attendance data effectively. In addition, while the impact from Covid-19 is still there, and there may be similar problems in the future, it is necessary for employees to sign a paper when taking attendance, or to use fingerprint technology. – which requires employees to have direct contact with the machine is limited, instead we are applying facial recognition technology to better solve the above problems. To solve the two problems mentioned above, the idea of "Staff management application, using face recognition in check-in system" needs to be researched and implemented. From there, companies will have more options to improve the management of their employees' working time better and safer during the current epidemic.

Keywords: *Staff management, .Net, Angular2, MTCNN, FaceNet*

LỜI CAM ĐOAN

Tôi xin cam đoan rằng những nghiên cứu về ứng dụng quản lý nhân viên sử dụng nhận diện khuôn mặt trong chấm công được trình bày trong đồ án này là của tôi và chưa từng được nộp như một báo cáo khóa luận tại trường Đại học Công Nghệ - Đại học Quốc Gia Hà Nội hoặc bất kỳ trường đại học khác. Những gì tôi viết ra không sao chép từ các tài liệu, không sử dụng các kết quả của người khác mà không trích dẫn cụ thể.

Tôi xin cam đoan công cụ nhận diện khuôn mặt trình bày trong khoá luận là do tôi tự phát triển, không sao chép mã nguồn của người khác. Nếu sai tôi hoàn toàn chịu trách nhiệm theo quy định của trường Đại học Công Nghệ - Đại học Quốc Gia Hà Nội.

Hà Nội, ngày 5 tháng 12 năm 2022

Sinh viên

Nguyễn Ngọc Minh

MỤC LỤC

DANH SÁCH KÝ HIỆU, CHỮ VIẾT TẮT

Từ viết tắt	Từ gốc	Ý nghĩa
CLR	Common Language Runtime	Hệ thống thực thi ảo của .Net
CLI	Common Language Infracstructure	Cơ sở hạ tầng của hệ thống CLR
IL	Intermediate Language	Ngôn ngữ trung gian
JIT	Just-in-Time	Trình biên dịch IL sang mã máy
DTO	Data Transfer Object	Các class đóng gói data để chuyển giữa client - server hoặc giữa các service, đảm bảo client nhận được các thông tin không nhạy cảm.
PWA	Progressive Web Application	Là giải pháp mang một trang web chạy giống như một ứng dụng native dưới mobile
MTCNN	Multi-task Cascaded Convolutional Networks	Là một mạng neuron nhân tạo, có khả năng nhận diện khuôn mặt và các điểm trên khuôn mặt.

DANH SÁCH HÌNH VẼ

Đặt vấn đề

Trong hoạt động của một công ty, việc quản lý công việc của nhân viên là điều gần như không thể thiếu. Đặc biệt là vấn đề đảm bảo các nhân viên của mình đều đi làm đầy đủ, đúng giờ giấc rất được chú trọng vì nó là 1 trong những yếu tố quan trọng để đảm bảo được nhân lực trong các công việc của công ty. Ngoài ra thời gian đi làm từng ngày của nhân viên cũng là 1 yếu tố quan trọng để bộ phận nhân sự đánh giá được chất lượng của từng nhân viên của công ty mình, từ đó tính toán được lương thưởng cho họ sau quá trình làm việc một cách chính xác nhất. Nếu một công ty không có hệ thống chấm công, hoặc hệ thống đó không đủ tốt, vẫn còn phải thủ công thì sẽ dễ xảy ra tình trạng gian lận trong khai báo thời gian đi làm, năng suất suất làm việc giảm, trong khi lương thưởng vẫn phải trả đầy đủ, khiến cho lợi nhuận của công ty bị sụt giảm.

Với tình trạng trên, thực tế rất nhiều công ty đã áp dụng công nghệ trong việc quản lý nhân viên. Các sản phẩm để phục vụ cho việc chấm công, ghi nhận thời gian đi làm của nhân viên đã được ra. Đa phần những giải pháp đó yêu cầu việc nhân viên sẽ phải chấm công bằng cách sử dụng thẻ định danh cá nhân, hoặc sử dụng vân tay[1].

Đối với việc sử dụng thẻ định danh cá nhân để quẹt hoặc quét mã, người dùng sẽ luôn phải nhớ mang thẻ của mình khi đi làm, nếu quên thẻ thì sẽ có thể sẽ phải mất thời gian để lấy, hoặc mất luôn ngày công đó. Ngoài ra sử dụng thẻ vẫn có thể dễ dàng gian lận nếu như để người khác sử dụng thẻ của mình để quẹt hộ. Tuy nhiên nhiều công ty vẫn lựa chọn sử dụng phương pháp này vì lắp đặt dễ dàng, linh hoạt, và người dùng không phải tiếp xúc trực tiếp với máy chấm công.

Còn đối với phương pháp sử dụng vân tay, nó đã giải quyết được các vấn đề của việc sử dụng thẻ cá nhân. Không cần phải mang thẻ, chỉ cần quét vân tay của chính mình, phương pháp này thực sự tiện lợi cho người dùng hơn. Tuy nhiên trong bối cảnh dịch bệnh hiện nay, và trong tương lai các công nghệ đang hạn chế nhiều người dùng tiếp xúc trực tiếp với máy chấm công, thì phương pháp sử dụng vân tay cũng đang được cân nhắc.

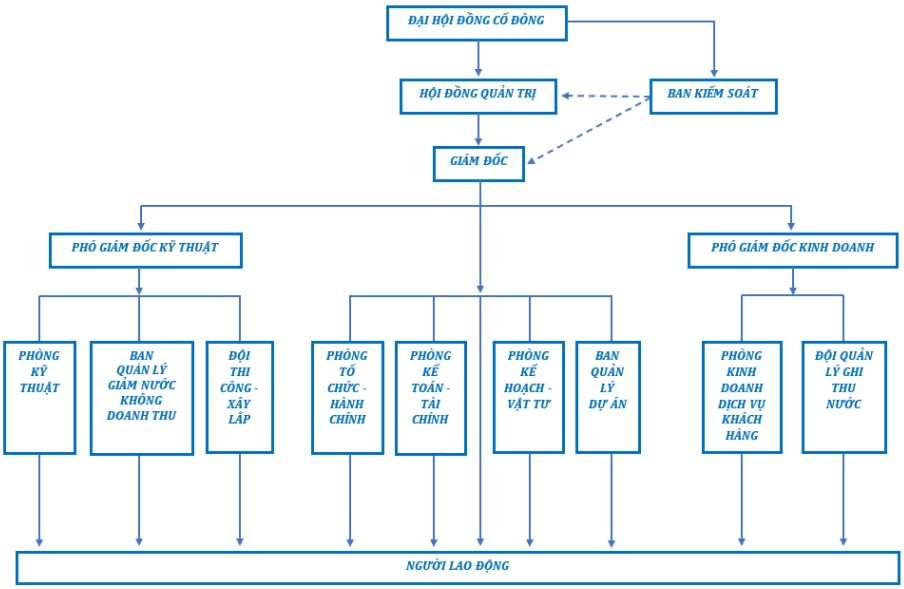
Do đó, ý tưởng về việc sử dụng khuôn mặt để định danh chấm công nhân viên đã ra đời để giải quyết vấn đề của 2 phương pháp trên. Kết hợp những lợi ích của phương pháp nhận diện khuôn mặt và nhu cầu quản lý giờ giấc đi làm của các công ty chính là lý do tôi đã lựa chọn nghiên cứu về đề tài "**Ứng dụng quản lý nhân viên, sử dụng nhận diện khuôn mặt để chấm công**".

Chương 1. Giới thiệu về ứng dụng quản lý nhân viên

Ở phần này, khóa luận trình bày các tính năng tổng quan về ứng dụng quản lý nhân viên và các công cụ được sử dụng trong việc thiết kế ứng dụng. Từ thực trạng hiện nay, đồ án làm nổi bật vai trò quan trọng và cần thiết của việc sử dụng ứng dụng quản lý trong thực tiễn của các công ty.

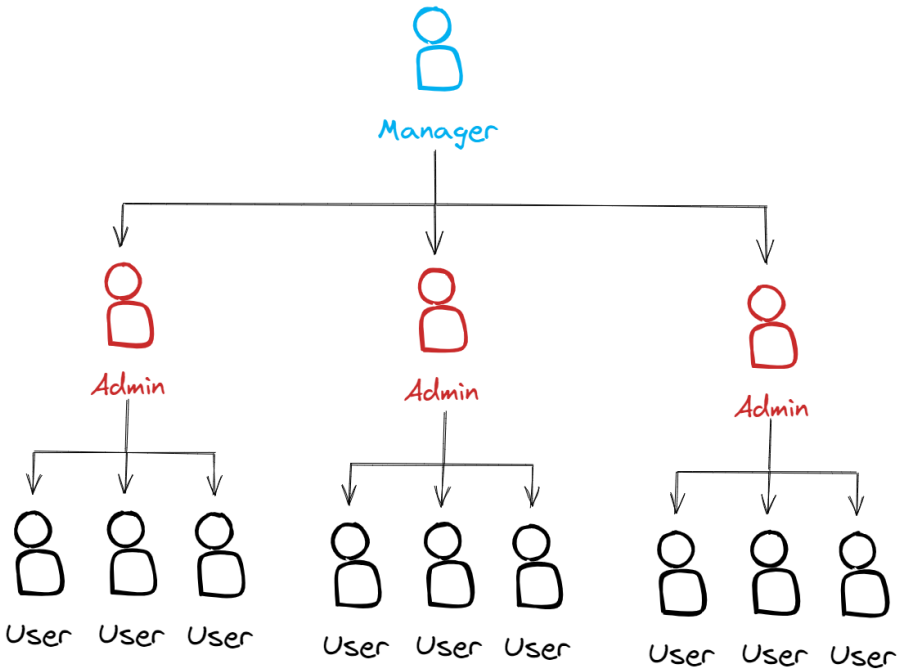
1.1. Mô tả chung về hệ thống

Tại các hệ thống công ty, việc phân cấp người dùng thường rất phức tạp do phân bậc chức vụ và vai trò quản lý trong hệ thống nhân sự có nhiều cấp bậc tùy theo khối lượng nhân sự trong từng công ty. Cho nên để thiết kế một hệ thống chứa đầy đủ các chức vụ, và sắp xếp chúng đúng với vai trò của mình là một điều khá khó, và để ứng dụng hoạt động với đa dạng nhu cầu của nhiều công ty lại càng khó hơn.



Hình 1-1-1 Ví dụ về các chức vụ và cấp bậc trong công ty

Cho nên tại hệ thống của ứng dụng này, tôi dùng 3 cấp chính trong phân quyền người dùng, đủ để vẫn đảm bảo "chia để trị" trong việc quản lý các tài khoản dưới quyền:

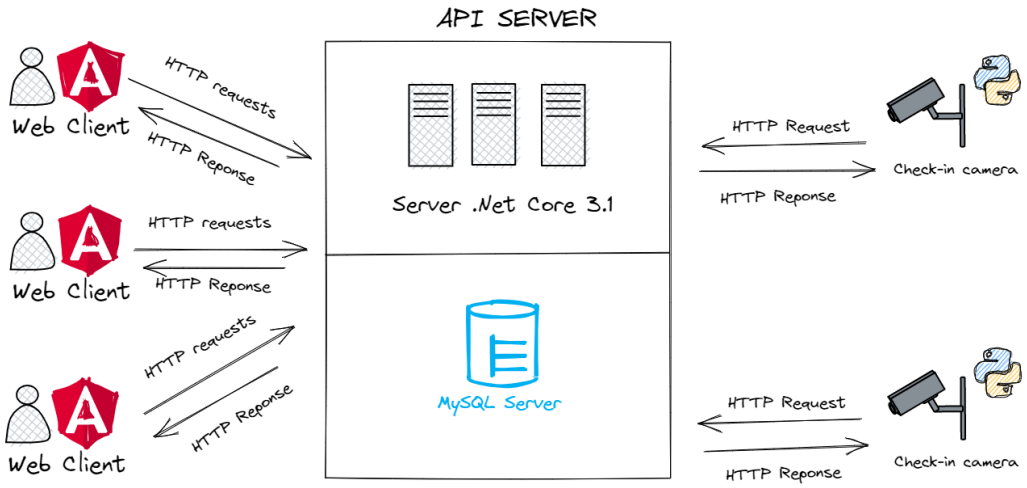


Hình 1-1-2 Phân quyền trong ứng dụng quản lý

- User: Người dùng cơ bản, là các nhân viên trong công ty. User có thể xem thông tin của tài khoản mình, và các thông báo chung của công ty.
- Admin: Là các quản lý các tài khoản User, có thể xem các tài khoản User cũng như quản lý hoạt động của những tài khoản đó.
- Manager: Là người đứng đầu hệ thống, có quyền cao nhất, đồng thời cũng là người cấp quyền và quản lý các tài khoản Admin và các tài khoản dưới quyền.

Hệ thống của ứng dụng hiện tại đang thiết kế với kích thước đủ để mô phỏng nhu cầu của một công ty vừa, bao gồm 3 thành phần chính là:

- Server: Sử dụng .Net Core 3.1 C# và Database là SQL Server
- Client: Sử dụng framework Angular2 ver. 16 để phát triển giao diện web phía người dùng tương tác với Server
- Thiết bị chấm công: Sử dụng ngôn ngữ Python để lập trình máy học, áp dụng 2 công nghệ là MTCNN trong nhận diện khuôn mặt và FaceNet trong xác nhận danh tính người.



Hình 1-1-3 Sơ đồ hệ thống của ứng dụng

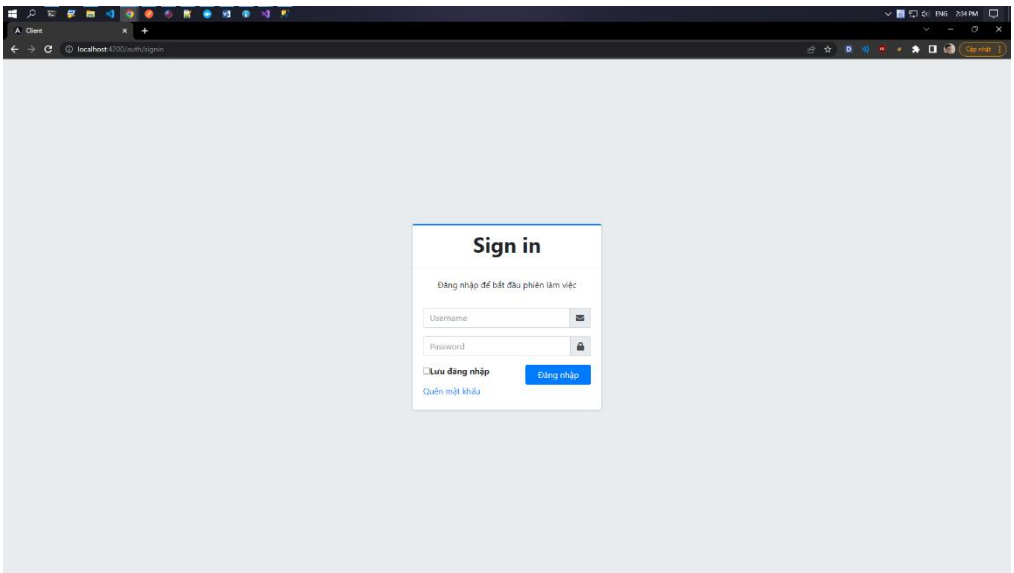
1.2. Nghiệp vụ của người dùng

Ở đây, đồ án sẽ trình bày rõ các vai trò và chức năng mà người dùng sẽ được quyền sử dụng, tùy theo quyền của tài khoản.

1.2.1. Nghiệp vụ của User

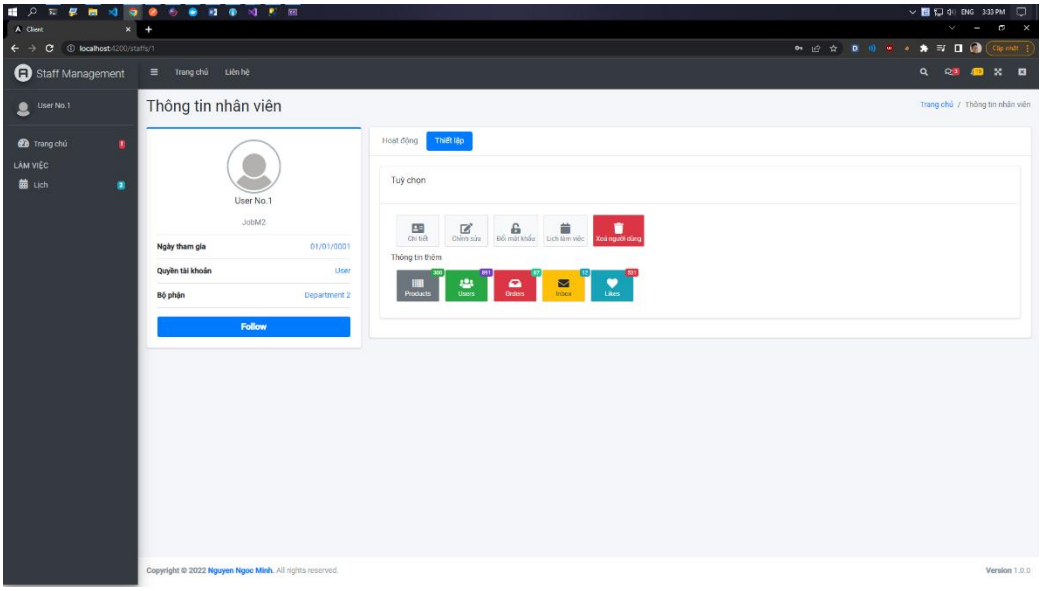
Với User, đây sẽ là kiểu tài khoản dành cho người dùng cơ bản.

User được người dùng quyền Admin hoặc cao hơn cấp tài khoản và khai thông tin theo đúng thông tin của người dùng đó. Điều này đảm bảo việc tài khoản được sinh ra đúng mục đích và số lượng tương ứng với vai trò và nhu cầu của từng người dùng, tránh trường hợp giả mạo tài khoản từ một bên khác nhằm trục lợi.



Hình 1-1-4 Giao diện trang đăng nhập

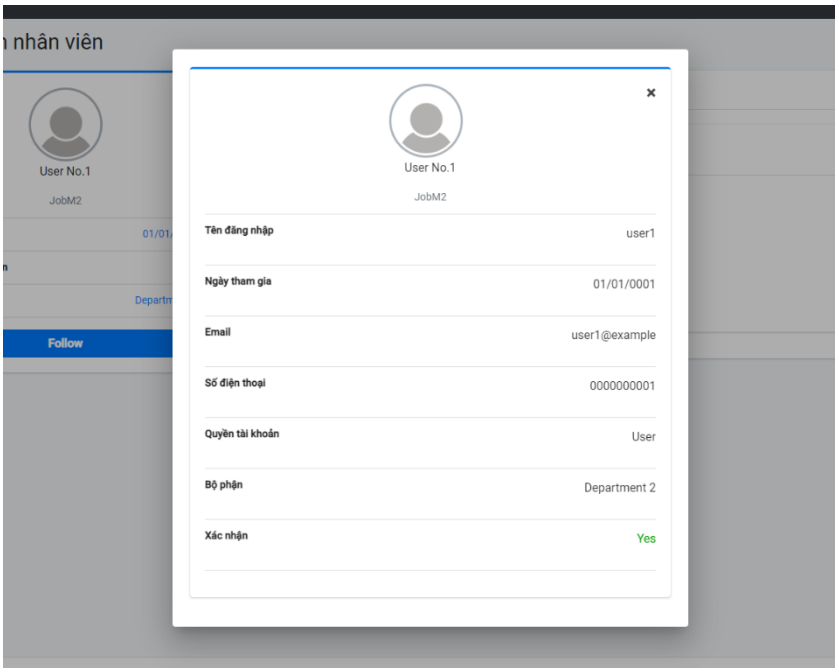
Sau khi đăng nhập, ta được màn hình của User như sau



Hình 1-1-5 Giao diện trang Thông tin nhân viên

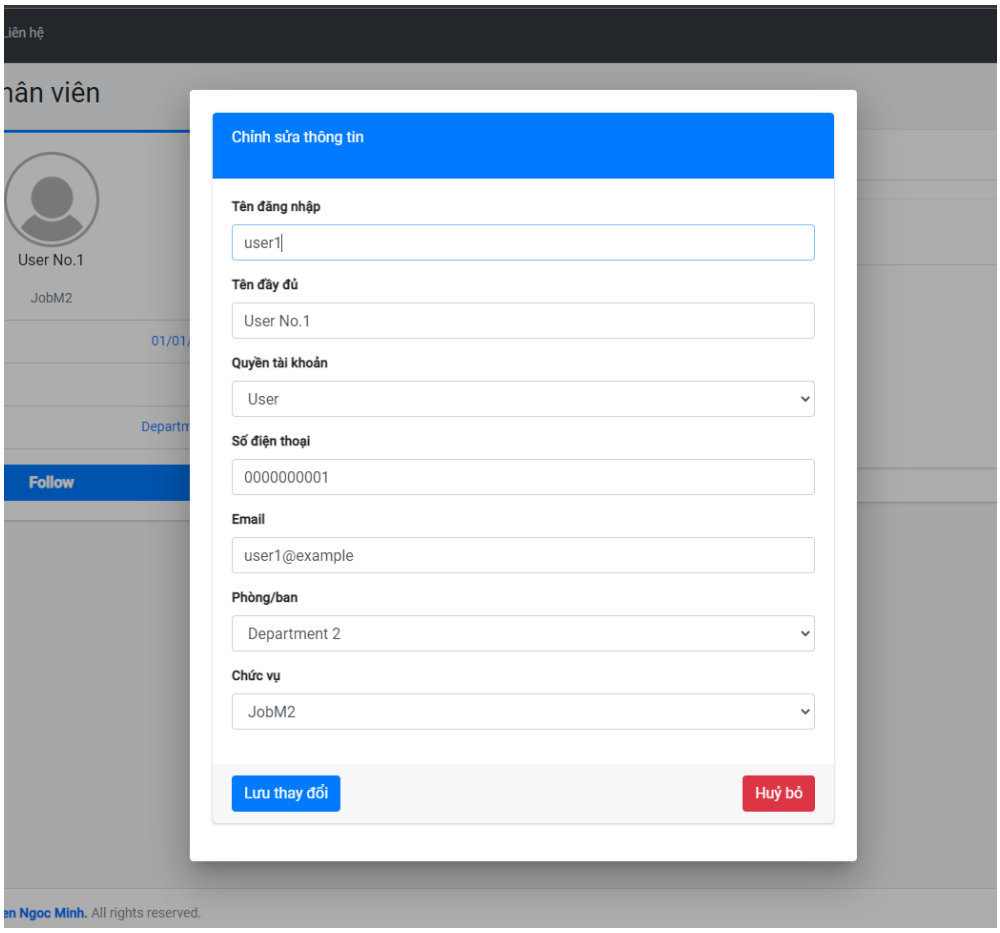
Người dùng chỉ có thể xem thông tin profile của bản thân mình. Ngoài ra tại trang profile, người dùng User có thể được truy cập vào những thao tác sau:

- Xem thông tin chi tiết tài khoản của mình, bao gồm: Họ và tên, Ngày tham gia, Email, Số điện thoại, … (không bao gồm mật khẩu)



Hình 1-1-6 Thông tin chi tiết của nhân viên

- Chỉnh sửa thông tin tài khoản của mình (không bao gồm đổi mật khẩu). Sau khi bất kỳ thông tin nào của người dùng được chỉnh sửa, trạng thái tài khoản sẽ lập tức chuyển sang **Chưa xác nhận**, và phải đợi quản trị viên xác nhận thông tin mới thì tài khoản đó mới được xem là hợp lệ.



Hình 1-1-7 Giao diện biểu mẫu chỉnh sửa thông tin nhân viên

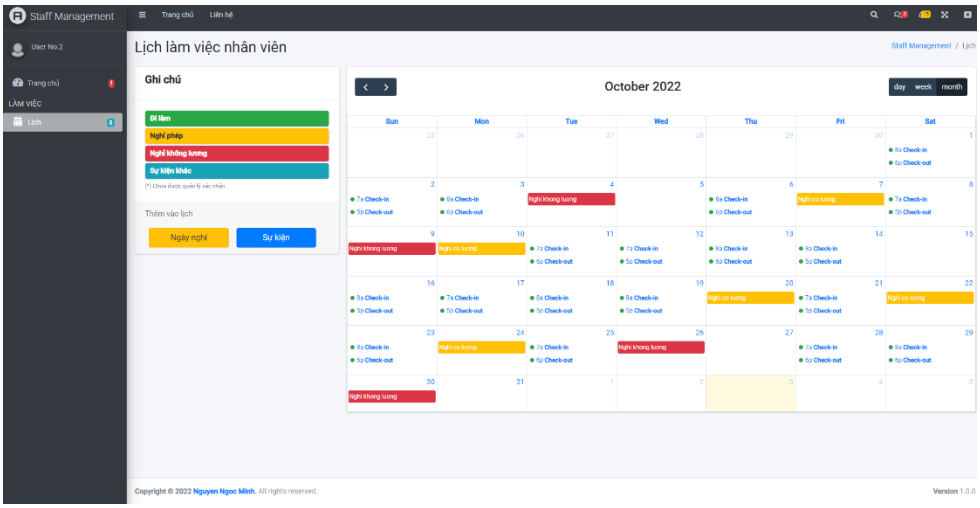
- Đổi mật khẩu tài khoản. Khi muốn đổi mật khẩu, người dùng sẽ phải nhập mật khẩu cũ và mới. Sau khi đổi mật khẩu, trạng thái tài khoản không bị thay đổi.

- Xem lịch làm việc. Người dùng có thể xem, chỉnh sửa, thêm bớt các sự kiện trong lịch làm việc của mình. Chúng ta sẽ nói rõ hơn về cái này ở phần sau

- Xóa người dùng. Người dùng hoàn toàn có thể xóa tài khoản của mình. Chức năng này dành cho trường hợp người dùng đã nghỉ việc thì sẽ tự xóa tài khoản

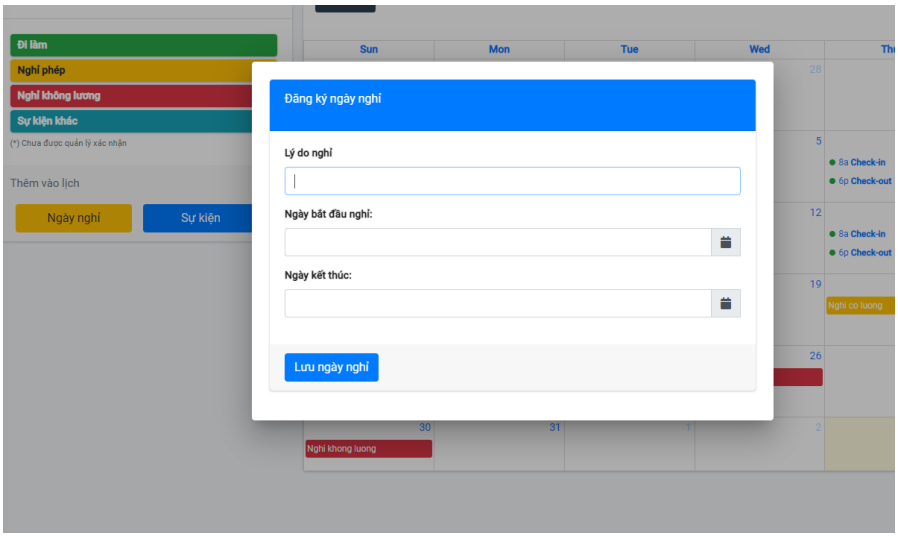
mình trong hệ thống, hoặc với các tài khoản bị tạo lỗi thì mình cũng có thể tự xóa đi.

Tính năng **lịch làm việc** là một trong những tính năng quan trọng nhất trong ứng dụng quản lý nhân viên này. Tất cả người dùng đều có thể truy cập và sử dụng tính năng này.



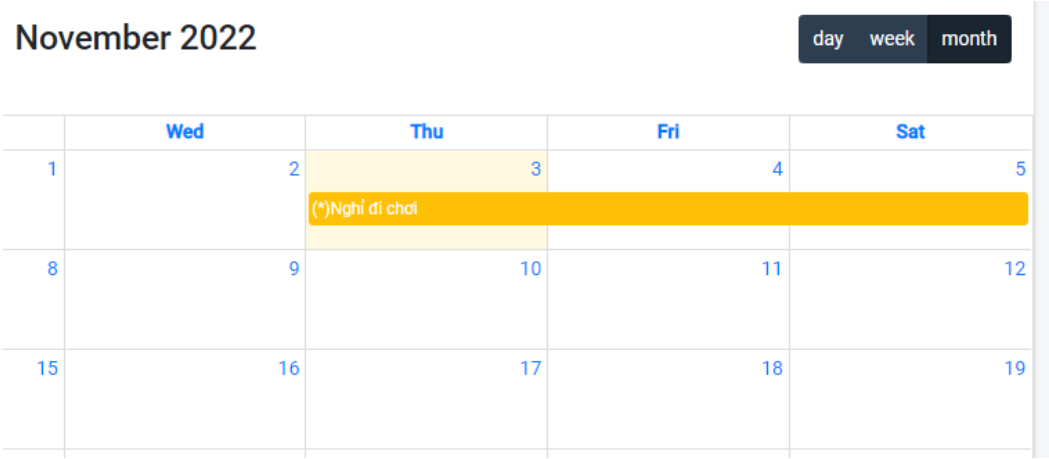
Hình 1-1-8 Giao diện lịch làm việc của nhân viên

Tại đây, người dùng có thể xem lại lịch làm việc của bản thân bao gồm thời gian Check-in/Check-out hằng ngày, ngày mình đã nghỉ, các sự kiện đã được tạo để ghi nhớ, các sự kiện chung của công ty. Ngoài ra tại đây, người dùng có thể đăng ký lịch nghỉ với quản lý trong trường hợp muốn được nghỉ làm một ngày hay một khoảng thời gian nhất định nào đó.



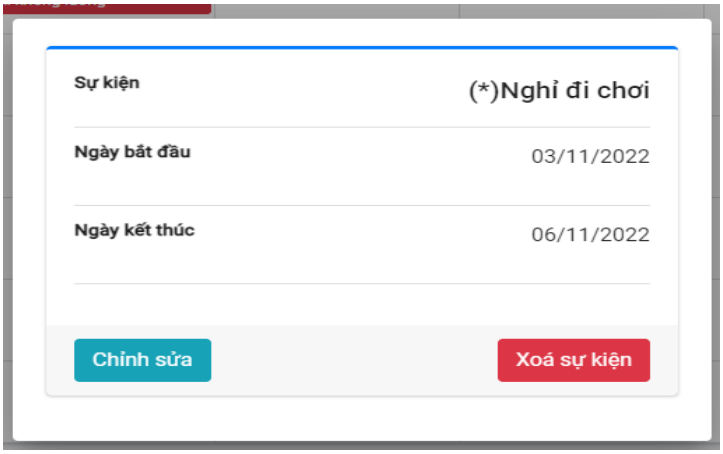
Hình 1-1-9 Giao diện biểu mẫu thêm ngày nghỉ

Sau khi một ngày nghỉ được tạo, Event nghỉ sẽ để ở trạng thái **Chưa xác nhận** để quản lý xác nhận ngày nghỉ này, Event nào chưa xác nhận sẽ có một dấu **(*)** để phân biệt với những Event đã được xác nhận. Đối với một vài công ty có cho phép nhân viên nghỉ phép có lương, ứng dụng sẽ tự động thiết lập ngày nghỉ có lương cho nhân viên dựa theo giới hạn số ngày nghỉ được phép mà công ty thiết lập, Event này sẽ có màu vàng. Các ngày nghỉ còn lại sẽ mặc định là không lương, Event này có màu đỏ. Người quản lý cũng có thể thay đổi kiểu ngày nghỉ của nhân viên mình nếu nó đó không hợp lệ. Khi tạo một ngày nghỉ, người dùng hoàn toàn có thể chỉnh sửa hoặc xóa ngày nghỉ đó trong trường hợp Event đó chưa xảy ra.



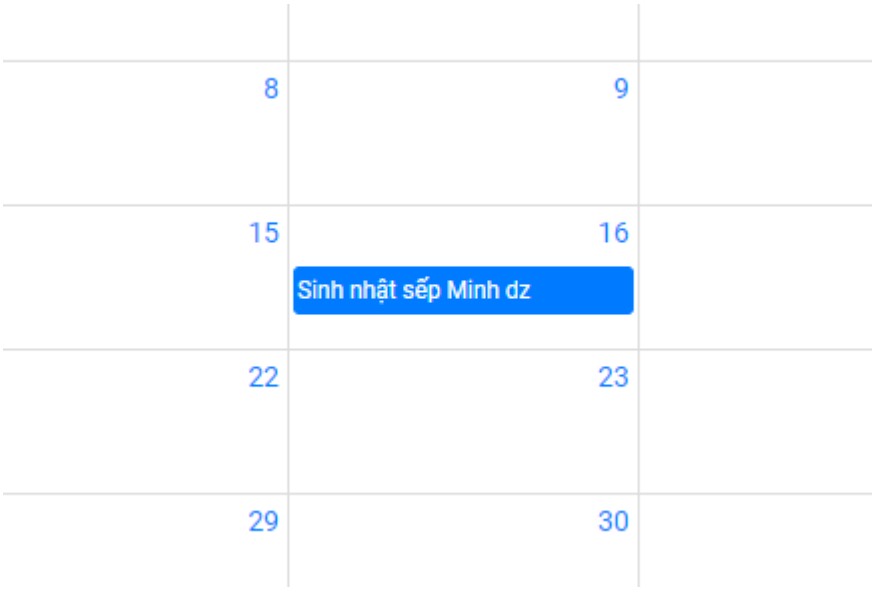
Hình 1-1-10 Ngày nghỉ sau khi được nhân viên tạo phải chờ xác nhận

Trong trường hợp muốn chỉnh sửa ngày nghỉ, Event đó sẽ quay trở lại trạng thái **Chưa xác nhận** nhằm đảm bảo người quản lý có thể cập nhật được sự thay đổi này và xác nhận lại một lần nữa.



Hình 1-1-11 Chi tiết thông tin của một sự kiện

Ngoài ra, lịch cũng cho phép người dùng tạo một sự kiện khác. Sự kiện này dành cho trường hợp nhân viên muốn ghi lại sự kiện, hoặc tạo 1 lời nhắc trong lịch làm việc. Việc tạo Event này tương tự như việc tạo một ngày nghỉ, tuy nhiên Event tạo ra không cần sự xác nhận của người dùng Admin/Manager và hoàn toàn có thể tạo, chỉnh sửa, xóa một cách tự do. Event này có màu nền là màu xanh dương.



Hình 1-1-12 Sự kiện được tạo trên lịch

Ngoài ra lịch cũng hiển thị thông tin Check-in và Check-out hằng ngày của nhân viên. Việc hiển thị rõ ràng như thế này nhằm cho người dùng biết được tiến độ đi làm của mình như thế nào. Từ đó có thể hoàn toàn đối chiếu với các ban khác nếu như có phát sinh vấn đề về trả thưởng lương.

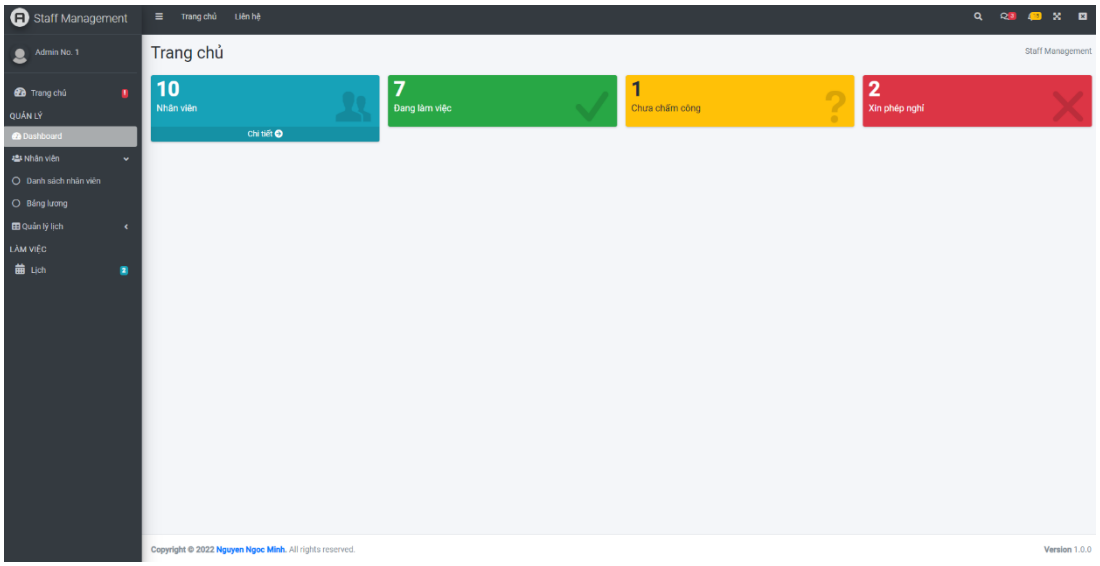
Hình 1-1-13 Sự kiện chấm công của nhân viên

Các Event này đều được sinh ra từ thiết bị chấm công được cài module Nhận diện khuôn mặt để nhận diện người dùng và thông báo lại cho Server để tạo Event chấm công như trên. Người dùng User không thể tạo, chỉnh sửa, xóa những event trên nhằm đảm bảo tính chính xác của việc chấm công, từ đó làm cơ sở để quản lý tính toán lương một cách chính xác nhất.

1.2.2. Nghiệp vụ của Admin

Người dùng Admin sở hữu các tính năng mà người dùng User có thể truy cập được.

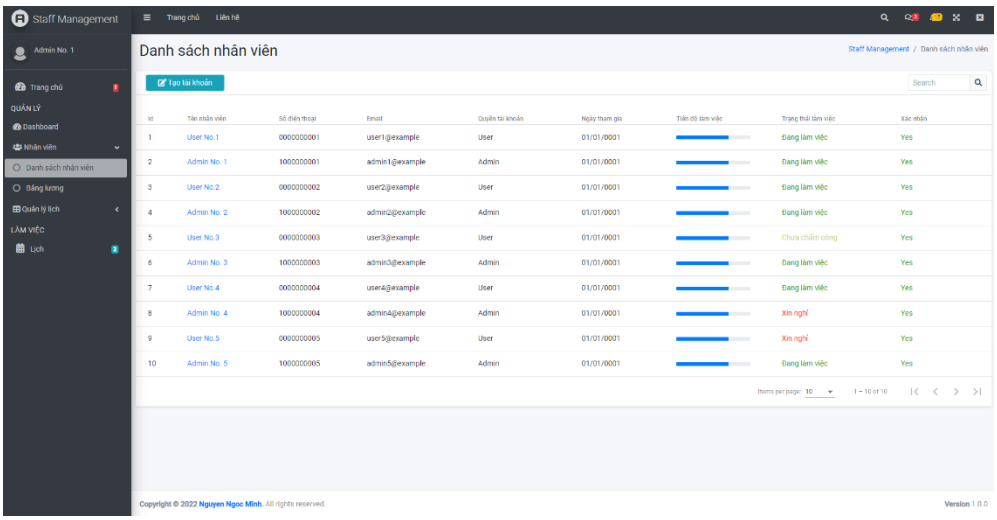
Màn hình làm việc của Admin sẽ có thêm các tính năng chỉ có Admin/Manager thấy được. Người dùng dưới quyền sẽ không thể truy cập tới những tính năng này, kể cả khi sử dụng Url tới tính năng thì cũng sẽ được chuyển hướng về trang chủ để đảm bảo tính phần quyền cho ứng dụng.



Hình 1-1-14 Giao diện Dashboard của Admin

Với vai trò là người dùng đại diện cho người Quản lý, Admin còn có những tính năng dành riêng cho công việc quản lý người dùng. Những tính năng đó bao gồm:

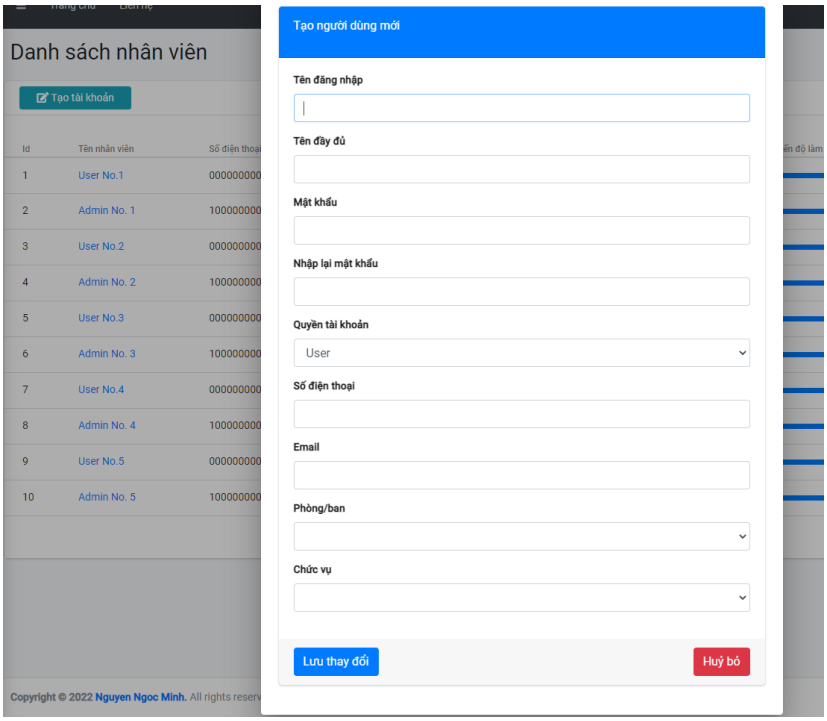
* Quản lý người dùng User: bao gồm việc thêm, xóa, chỉnh sửa thông tin, xác nhận thông tin tài khoản người dùng.



Hình 1-1-15 Giao diện danh sách nhân viên

Admin có thể xem được danh sách nhân viên và trạng thái đi làm của mọi người trong ngày. Tại đây Admin có thể truy cập vào thông tin profile của từng người dùng, thực hiện việc chỉnh sửa thông tin, đổi mật khẩu hay xóa tài khoản như User thực hiện trên chính tài khoản mình.

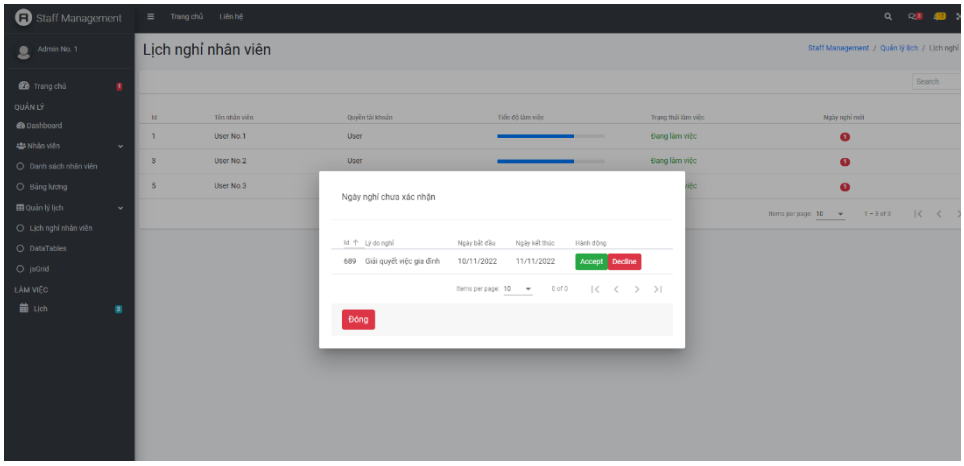
Admin có thể tạo tài khoản người dùng User. Tài khoản sau khi được tạo sẽ tự động chuyển qua chế độ Được xác nhận và có thể được sử dụng ngay.



Hình 1-1-16 Biểu mẫu tạo người dùng mới

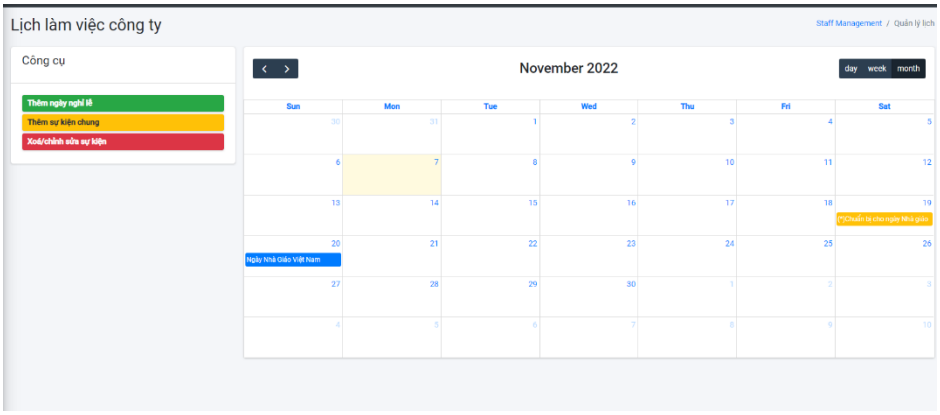
* Quản lý các sự kiện, tạo, xóa, chỉnh sửa sự kiện chung cho công ty, quản lý các sự kiện của người dùng User

Cũng như người dùng User, người dùng Admin cũng là nhân viên, nên sẽ có lịch làm việc cũng như lịch sự kiện riêng, và có các tính năng tạo, chỉnh sửa, xóa ngày nghỉ/sự kiện như một User bình thường.
Ngoài ra, là một Admin, người dùng này còn có thể xem lịch làm việc của tất cả các User, quản lý và xác nhận các thông báo nghỉ của nhân viên.



Hình 1-1-17 Danh sách các ngày nghỉ đang chờ xác nhận

Người dùng Admin cũng là người sẽ thiết lập các sự kiện/ngày nghỉ chung cho toàn công ty.

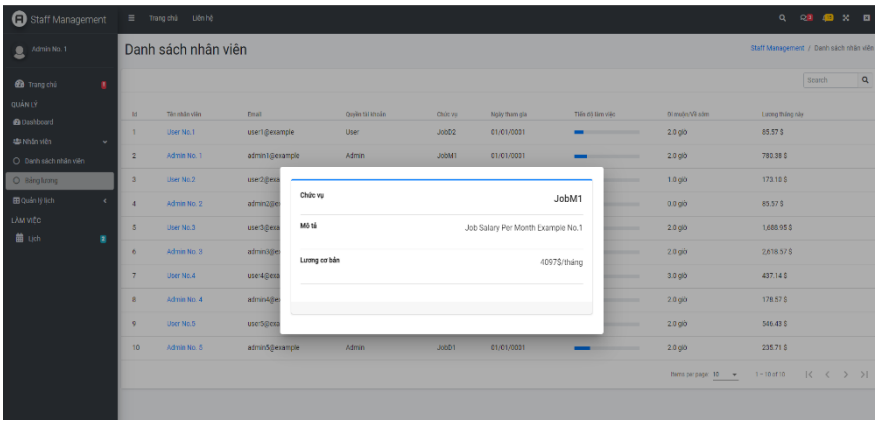


Hình 1-1-18 Giao diện lịch làm việc toàn công ty

Những sự kiện chung được Admin thêm vào sẽ áp dụng cho toàn bộ lịch làm việc của công ty, bao gồm các ngày nghỉ phép (như Tết Nguyên Đán, Quốc

Khánh, Ngày Lao Động, Ngày nghỉ riêng của công ty,….) hay các sự kiện thông thường nhằm mục đích nhắc nhở nhân viên (như Ngày thành lập công ty, Ngày nhận lương, …).

* Xem bảng tính lương của nhân viên: Quản lý sẽ có thể xem bảng tính lương của nhân viên dựa trên ngày đi làm được máy chấm công ghi lại và lương cơ bản của công việc mà họ đảm nhận.



Hình 1-1-19 Bảng tính lương và chi tiết công việc của một nhân viên

Bảng tính lương này sẽ chỉ dựa vào thời gian đi làm, các yếu tố như thuế, thưởng KPI, thưởng hoa hồng, … sẽ chưa được áp dụng vào bảng lương này do yếu tố này khá khác nhau ở nhiều công ty. Tuy nhiên bảng lương này sẽ là một công cụ tốt để tính toán được phần lương từ việc đi làm, giúp giảm bớt công việc tính toán cho bộ phận Hành Chính của công ty.

1.2.3. Nghiệp vụ của Manager

Người dùng Manager sở hữu các tính năng mà người dùng User và Admin có thể truy cập được. Đây sẽ là tài khoản có quyền hạn cao nhất, là người tạo, quản lý các tài khoản Admin cũng như những sự kiện trong lịch làm việc của các loại tài khoản có quyền Admin trở xuống.

1.2.4. Tóm tắt phần nghiệp vụ

Với những giới thiệu về nghiệp vụ của các loại tài khoản trên, ta tóm gọn lại những chức năng mà từng loại tài khoản có thể truy cập được như sau:

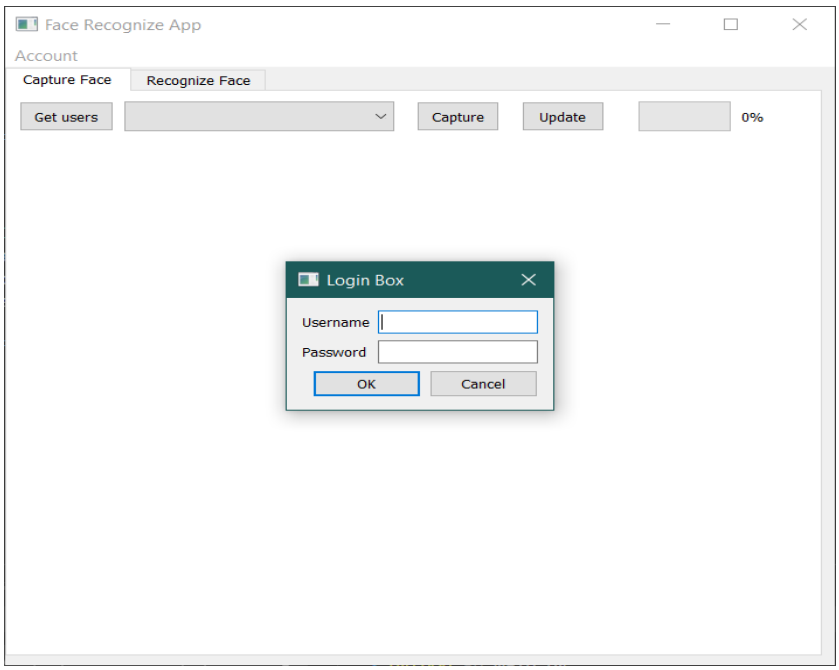
Loại tài khoản	Chức năng

User	- Đăng nhập/Đăng xuất - Xem thông tin cá nhân tài khoản của mình - Chỉnh sửa thông tin cá nhân của mình (yêu cầu xác nhận từ Admin/Manager) - Xóa tài khoản của mình - Xem, thêm các sự kiện trong lịch làm việc - Thêm ngày nghỉ trong lịch làm việc (yêu cầu xác nhận tự Admin/Manager) - Chỉnh sửa/xóa các sự kiện, ngày nghỉ trong lịch làm việc - Điểm danh bằng khuôn mặt
Admin	- Các chức năng của User - Xem, lọc danh sách tất cả người dùng - Tạo tài khoản người dùng User - Xem thông tin cá nhân của các tài khoản khác - Chỉnh sửa thông tin tài khoản User khác - Xóa tài khoản User - Xem lịch làm việc của các tài khoản User khác - Quản lý các sự kiện có trong lịch làm việc của User - Thêm, chỉnh sửa, xóa các sự kiện, ngày nghỉ chung của toàn công ty
Manager	- Các chức năng của Admin - Tạo tài khoản người dùng Admin/User - Chỉnh sửa thông tin tài khoản Admin/User khác - Xóa tài khoản Admin/User - Xem lịch làm việc của các tài khoản Admin/User khác - Quản lý các sự kiện có trong lịch làm việc của Admin/User

1.3. Phần mềm chấm công bằng nhận diện khuôn mặt

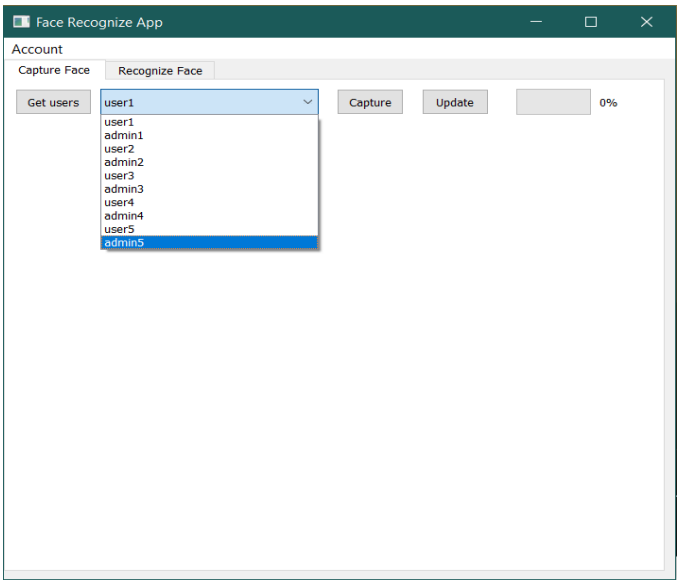
Phần mềm chấm công bằng nhận diện khuôn mặt bao gồm 2 công việc chính là: Đăng ký khuôn mặt và Nhận diện khuôn mặt.

Phần mềm sẽ được kết nối với một camera đặt tại công ty, có thể chạy phần mềm này ngay tại thiết bị có hệ điều hành Windows hoặc hệ điều hành nhân Linux. Phần mềm được thiết kế để chạy được ngay trên cả các thiết bị như Raspberry Pi có gắn kèm Camera như một máy tính thu nhỏ.



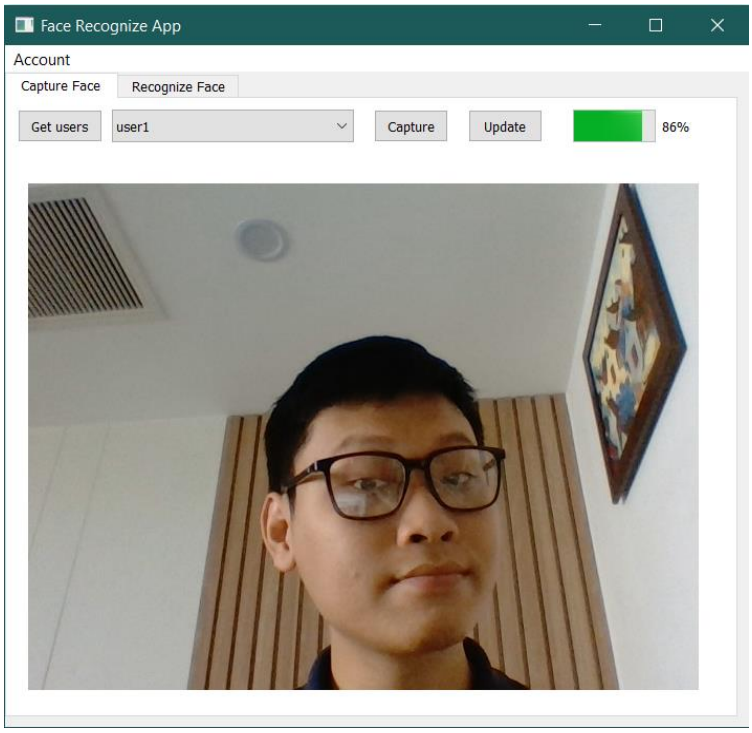
Hình 1-1-20 Đăng nhập vào phần mềm chấm công

Đầu tiên, khi khởi động phiên làm việc của phần mềm, ta cần phải đăng nhập với một tài khoản có cấp độ Admin trở lên vào máy.



Hình 1-21 Lựa chọn người dùng để ghi nhận khuôn mặt

Tại màn hình đăng ký khuôn mặt, ta chọn người dùng cần đăng ký khuôn mặt. Thông tin về người dùng sẽ được ứng dụng lấy từ API để đảm bảo đúng người dùng được nhận diện.



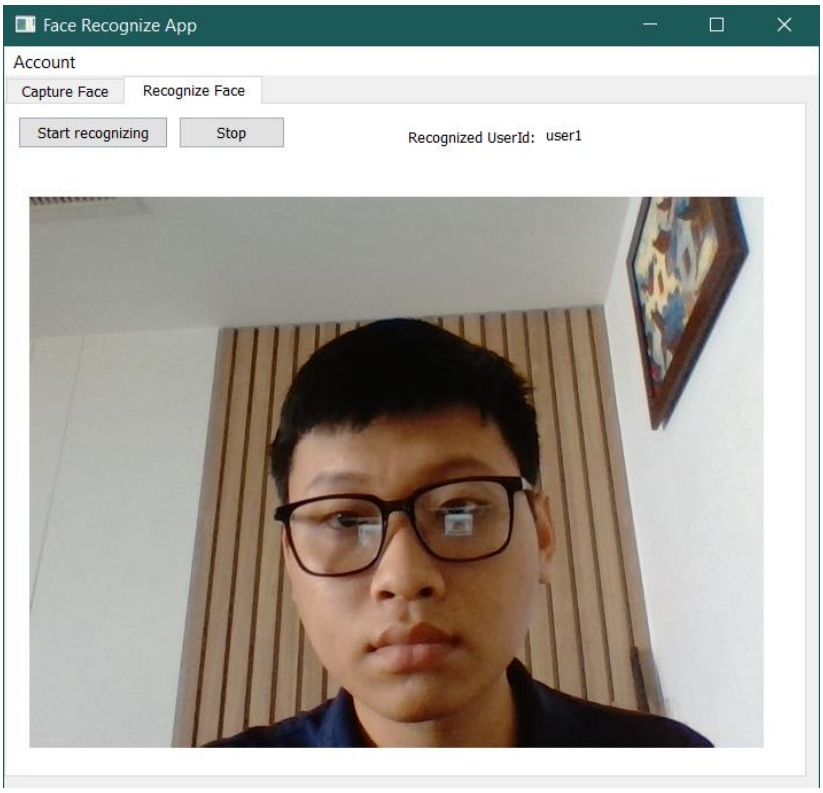
Hình 1-22 Quá trình ghi nhận khuôn mặt

Sau khi chọn xong người dùng, ta tiến hành ghi lại khuôn mặt nhân viên bằng nút **Capture**, công việc này cần nơi có ánh sáng đủ lớn, không tối, và người dùng phải nhìn vào máy. Khi những yếu tố trên, cộng với việc đầy đủ khuôn mặt của nhân viên được hiển thị trên màn hình, phần mềm sẽ tự động ghi nhận khuôn mặt, thanh % sẽ chạy cho tới 100%. Nếu như có sự cố, ví dụ người dùng không tiếp tục để khuôn mặt trong camera, thanh % sẽ tự động dừng, chờ việc tiếp tục ghi hình.

Sau khi thanh chạy đủ 100%, ta tiếp tục chọn **Update** để xác nhận khuôn mặt này lên hệ thống, cũng như cập nhật dữ liệu đã được ghi lại để phần mềm xử lý.

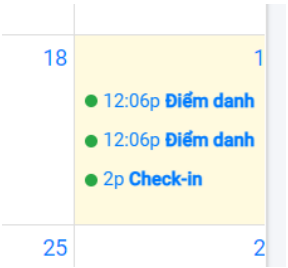
Đối với công việc nhận diện khuôn mặt cho điểm danh, ta lựa chọn tab Regconize Face trên ứng dụng, và lựa chọn **Start recognizing** để phần mềm tiến hành công việc nhận diện. Phần mềm sẽ chạy 24/24, tự động nhận diện các khuôn mặt được đặt trong tầm camera, và nhìn thẳng vào máy.

Phần mềm sẽ tự động nhận diện khuôn mặt nhìn thẳng vào máy trong một cự ly cố định, để đảm bảo chỉ nhận được khuôn mặt có ý muốn điểm danh.



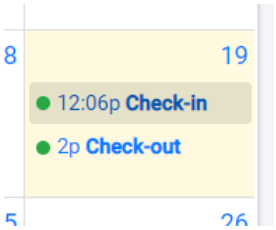
Hình 1-23 Quá trình tự động chấm công những khuôn mặt ghi nhận được

Khi nhận diện thành công, thông tin nhân viên được nhận diện sẽ hiển thị trên phần mềm, đồng thời nó cũng request về API để thông báo cho việc nhận diện này.



Hình 1-24 Các sự kiện điểm danh trong ngày của nhân viên

Các sự kiện điểm danh trong ngày sẽ được để là các sự kiện Điểm danh, vào cuối ngày, BackgroundService ở hệ thống sẽ tự động xử lý tất cả sự kiện trên thành 2 sự kiện cuối cùng là Check-in và Check-out.



Hình 1-25 Các sự kiện điểm danh đã được xử lý ở cuối ngày

Đồng thời hệ thống cũng tự động tính toán ngày hôm đó nhân viên đã làm đủ một ngày công hay chưa dựa theo 2 sự kiện trên, và lưu lại để thực hiện tính lương.

Chương 2. Hệ thống của Ứng dụng quản lý nhân viên

Chương trước ta đã được giới thiệu tổng quan về toàn bộ các module trong Ứng dụng quản lý nhân viên cũng như các nghiệp vụ của người dùng ứng dụng. Chương này sẽ trình bày về hệ thống vận hành ứng dụng này một cách cụ thể.

Như đã giới thiệu tại phần tổng quan, hệ thống ứng dụng được chia làm 3 phần là phần Server, Client và Phần mềm chấm công. Ta sẽ đi vào chi tiết từng phần như sau:

2.1. Server

Server của ứng dụng được phát triển từ .Net Core của ngôn ngữ C#. Đây là một nền tảng khá phổ biến trong việc phát triển các hệ thống lớn, nhiều người dùng.

Đối với việc quản lý dữ liệu của người dùng, server sử dụng SQL Server.

2.1.1. C# và .Net Core

C#[2] là một ngôn ngữ lập trình mạnh mẽ và đa năng, được phát triển bởi đội ngũ kỹ sư của Microsoft vào năm 2000. Nó là một ngôn ngữ lập trình hiện đại, hướng đối tượng và "type-safe". C# được phát triển dựa trên các ngôn ngữ lập trình C (như C và C++) và ngôn ngữ Java.

C# cung cấp nhiều tính năng phù hợp để phát triển một ứng dụng mạnh mẽ và có thể chạy bền bỉ. Có thể kể đến một vài tính năng phổ biến như:

- *Garbage collection*: tự động thu thập các phần bộ nhớ bị chiếm dụng bởi các object không được sử dụng tới, hoặc object không thể truy cập được.
- *Nullable types*: có khả năng tạo giá trị null cho các biến để đề phòng việc một biến chưa được khởi tạo object.
- *Exception handling*: cung cấp cách tiếp cận, phát hiện và khôi phục lỗi trong thực thi một cách có cấu trúc và dễ dàng mở rộng các cách thức trên.
- *Lambda expressions*: chức năng quan trọng trên nhiều ngôn ngữ, hỗ trợ việc lập trình dựa trên kỹ thuật lập trình chức năng (functional programming) mà ta thường thấy trong ngôn ngữ JavaScript.
- *Language Integrated Query (LINQ):* là chức năng rất mạnh mẽ trong C#. Chức năng này hỗ trợ việc chuyển các câu lệnh truy vấn truyền thống sang câu lệnh thực thi trực tiếp trên đối tượng để truy cập nhiều nguồn dữ liệu.

- *Asynchronous operations*: hỗ trợ việc lập trình bất đồng bộ, khiến ngôn ngữ có thể xử lý rất nhiều request một lúc, tận dụng tốt tài nguyên của hệ thống và rút ngắn được thời gian xử lý.
- V…v…

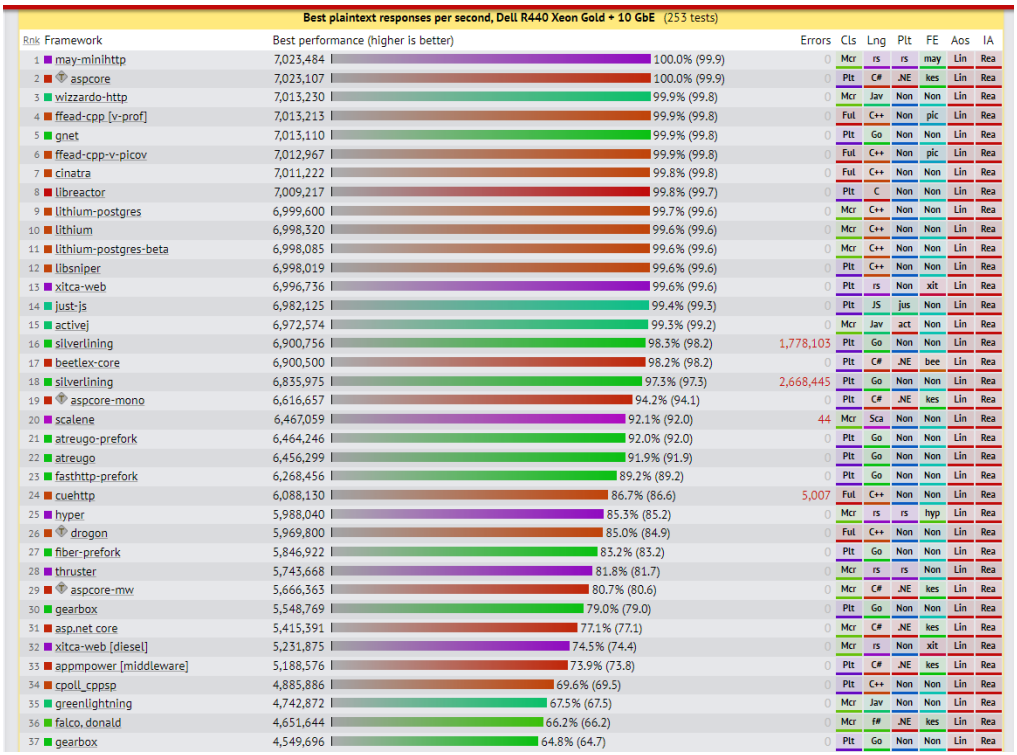
Như đã nói ở trên, C# là một ngôn ngữ lập trình "type-safe". C# thống nhất một kiểu dữ liệu hệ thống chung. Tất cả kiểu dữ liệu trong C#, kể cả các kiểu dữ liệu nguyên thủy như *int* hay *double* đều được kế thừa từ gốc duy nhất là kiểu *object.* Tất cả các kiểu dữ liệu đều chia sẻ một tập hợp các hoạt động chung từ kiểu dữ liệu gốc. Các giá trị thuộc bất kỳ kiểu dữ liệu nào đều có thể được lưu trữ, vận chuyển và vận hành theo một cách nhất quán. Hơn nữa, C# còn hỗ trợ cả kiểu dữ liệu tham chiếu (reference type) và kiểu dữ liệu giá trị (value type). Cấp phát động đối tượng, phương thức và kiểu dữ liệu tổng quát (Generic methods and types), iterators,… cũng được C# cung cấp cho lập trình viên phát triển sản phẩm một cách tối ưu và mạnh mẽ nhất.

Với những thế mạnh trên, C# là một trong những lựa chọn hàng đầu cho lập trình viên khi phát triển các ứng dụng. Do đó C# có một cộng đồng lớn và tích cực hoạt động, tạo nên nhiều thư viện mạnh mẽ để đáp ứng các nhu cầu càng ngày càng nhiều trong nhiều ngành nghề. Và để giúp mọi người dùng dễ dàng sử dụng, mở rộng và tùy biến các thư viện đúng với nhu cầu của bản thân, C# cung cấp cho người dùng tính năng *virtual* và *override.* Ngày nay, C# vẫn được tiếp tục phát triển, đưa ra các bản cập nhật thường xuyên, giúp các lập trình viên yên tâm khi sử dụng công cụ này lâu dài.

.Net[3] hay **dotnet** là một nền tảng mã nguồn mở miễn phí, chạy đa nền tảng, được sử dụng để phát triển nhiều loại ứng dụng. Các ứng dụng có thể phát triển từ .Net rất đa dạng, từ web, mobile, desktop app, tới game, IoT,….

.Net cũng được chính đội ngũ kỹ sư từ Microsoft phát triển từ năm 2002, nay thuộc tổ chức phi lợi nhuận .Net Foundation[4] , là tổ chức được thành lập nhằm mục đích hỗ trợ phát triển, quảng bá một hệ sinh thái của nền tảng .Net. Nhờ đó, .Net được rất nhiều tổ chức lớn trên thế giới sử dụng để phát triển các sản phẩm quan trọng của họ, ví dụ như: Microsoft, StackOverFlow, Accenture, Agoda, Samsung, JP Morgan Chase, …[5]

.Net sử dụng các ngôn ngữ như F#, Visual Basic, và phổ biến nhất là C#. Với những điểm mạnh đã được kể trên, sức mạnh của C# đã tạo nên một nền tảng hình thành cực kỳ mạnh mẽ cho .Net. Điều đó được chứng minh trong kết quả đo điểm số hiệu năng của .Net (trong những công việc như xử lý JSON, truy cập database, render template phía server,…) so với các Web Framework khác như sau[7]:



Best plaintext responses per second, Dell R440 Xeon Gold + 10 GbE (253 tests)

Rnk	Framework	Best performance (higher is better)	Errors	Cls	Lng	Plt	FE	Aos	IA
1	may-minihttp	7,023,484 100.0% (99.9)	0	Mcr	rs	rs	may	Lin	Rea
2	aspcore	7,023,107 100.0% (99.9)	0	Plt	C#	.NE	kes	Lin	Rea
3	wizzardo-http	7,013,230 99.9% (99.8)	0	Mcr	Jav	Non	Non	Lin	Rea
4	ffead-cpp [v-prof]	7,013,213 99.9% (99.8)	0	Ful	C++	Non	pic	Lin	Rea
5	gnet	7,013,110 99.9% (99.8)	0	Plt	Go	Non	Non	Lin	Rea
6	ffead-cpp-v-picov	7,012,967 99.9% (99.8)	0	Ful	C++	Non	pic	Lin	Rea
7	cinatra	7,011,222 99.8% (99.8)	0	Ful	C++	Non	Non	Lin	Rea
8	libreactor	7,009,217 99.8% (99.7)	0	Plt	C	Non	Non	Lin	Rea
9	lithium-postgres	6,999,600 99.7% (99.6)	0	Mcr	C++	Non	Non	Lin	Rea
10	lithium	6,998,320 99.6% (99.6)	0	Mcr	C++	Non	Non	Lin	Rea
11	lithium-postgres-beta	6,998,085 99.6% (99.6)	0	Mcr	C++	Non	Non	Lin	Rea
12	libsniper	6,998,019 99.6% (99.6)	0	Plt	C++	Non	Non	Lin	Rea
13	xitca-web	6,996,736 99.6% (99.6)	0	Plt	rs	Non	xit	Lin	Rea
14	just-js	6,982,125 99.4% (99.3)	0	Plt	JS	jus	Non	Lin	Rea
15	activej	6,972,574 99.3% (99.2)	0	Mcr	Jav	act	Non	Lin	Rea
16	silverlining	6,900,756 98.3% (98.2)	1,778,103	Plt	Go	Non	Non	Lin	Rea
17	beetlex-core	6,900,500 98.2% (98.2)	0	Plt	C#	.NE	bee	Lin	Rea
18	silverlining	6,835,975 97.3% (97.3)	2,668,445	Plt	Go	Non	Non	Lin	Rea
19	aspcore-mono	6,616,657 94.2% (94.1)	0	Plt	C#	.NE	kes	Lin	Rea
20	scalene	6,467,059 92.1% (92.0)	44	Mcr	Sca	Non	Non	Lin	Rea
21	atreugo-prefork	6,464,246 92.0% (92.0)	0	Plt	Go	Non	Non	Lin	Rea
22	atreugo	6,456,299 91.9% (91.9)	0	Plt	Go	Non	Non	Lin	Rea
23	fasthttp-prefork	6,268,456 89.2% (89.2)	0	Plt	Go	Non	Non	Lin	Rea
24	cuehttp	6,088,130 86.7% (86.6)	5,007	Ful	C++	Non	Non	Lin	Rea
25	hyper	5,988,040 85.3% (85.2)	0	Mcr	rs	rs	hyp	Lin	Rea
26	drogon	5,969,800 85.0% (84.9)	0	Ful	C++	Non	Non	Lin	Rea
27	fiber-prefork	5,846,922 83.2% (83.2)	0	Plt	Go	Non	Non	Lin	Rea
28	thruster	5,743,668 81.8% (81.7)	0	Mcr	rs	rs	Non	Lin	Rea
29	aspcore-mw	5,666,363 80.7% (80.6)	0	Mcr	C#	.NE	kes	Lin	Rea
30	gearbox	5,548,769 79.0% (79.0)	0	Plt	Go	Non	Non	Lin	Rea
31	asp.net core	5,415,391 77.1% (77.1)	0	Mcr	C#	.NE	kes	Lin	Rea
32	xitca-web [diesel]	5,231,875 74.5% (74.4)	0	Mcr	rs	Non	xit	Lin	Rea
33	appmpower [middleware]	5,188,576 73.9% (73.8)	0	Plt	C#	.NE	kes	Lin	Rea
34	cpoll_cppsp	4,885,886 69.6% (69.5)	0	Plt	C++	Non	Non	Lin	Rea
35	greenlightning	4,742,872 67.5% (67.5)	0	Mcr	Jav	Non	Non	Lin	Rea
36	falco.donald	4,651,644 66.2% (66.2)	0	Mcr	F#	.NE	kes	Lin	Rea
37	gearbox	4,549,696 64.8% (64.7)	0	Plt	Go	Non	Non	Lin	Rea

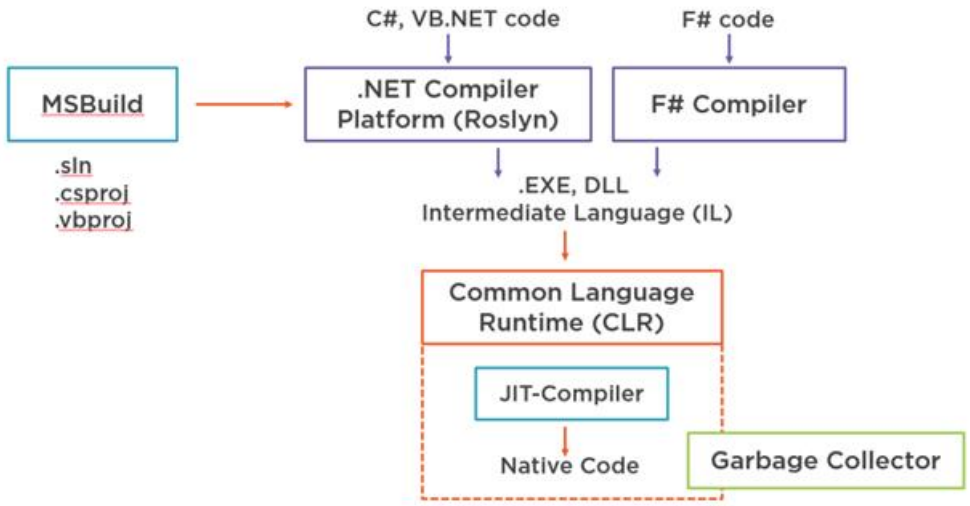
Hình 2-2-1 Bảng benchmarck khả năng xử lý của các Web Framework

Kết quả trên cho thấy sự khác biệt trong hiệu năng hoạt động của các framework trên nền tảng .Net rất mạnh, đặc biệt là so với những framework thường dùng khác.

Kiến trúc của .Net bao gồm một hệ thống thực thi ảo CLR và tập hợp các lớp thư viện. CLR được Microsoft thực thi nhờ vào cơ sở hạ tầng của hệ thống thực thi là CLI dựa vào các tiêu chuẩn quốc tế. Do đó, CLI là cơ sở để tạo môi trường thực thi và phát triển trong đó các ngôn ngữ và thư viện hoạt động liền mạch với nhau.

Mã nguồn được viết bằng C# sẽ được biên dịch sang ngôn ngữ trung gian (IL) phù hợp với đặc điểm kỹ thuật của CLI. Mã IL và các tài nguyên, chẳng hạn như bitmap và strings, được lưu trữ trong một *assembly*, thường có phần mở rộng là *.dll*. Một *assembly* chứa một tệp khai báo cung cấp thông tin về kiểu dữ liệu , phiên bản và đặc điểm nhận biết của *assembly* đó.

Khi một chương trình C# được thực thi, các *assembly* được nạp vào trong CLR. CLR sử dụng trình biên dịch JIT để biên dịch mã IL thành mã máy tự nhiên. CLR cũng



Hình 2-2-2 Mô hình toolchain của .Net (cũ)

cung cấp các dịch vụ hỗ trợ những công việc trên như tự *garbage collection*, *exception handling*, và quản lý tài nguyên.

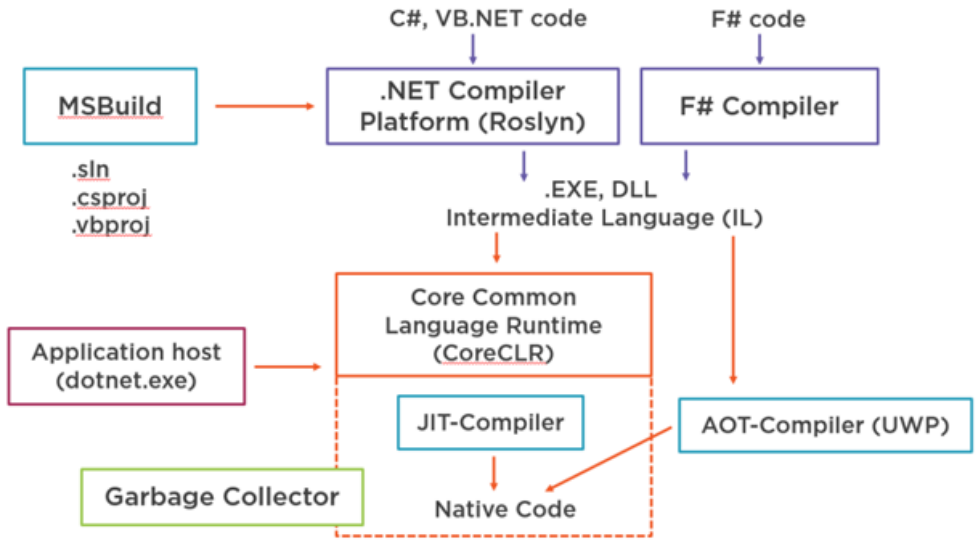
Khả năng tương tác với ngôn ngữ là một tính năng quan trọng của .Net. Mã IL được tạo từ C# hoàn toàn có thể tương tác với mã được tạo ra bởi các phiên bản .Net của F#, Visual Basic và C++. Một *assembly* có thể chứa nhiều module được viết bởi nhiều ngôn ngữ sử dụng .Net khác nhau. Các kiểu dữ liệu của chúng đều có thể tham chiếu với nhau như cách chúng tham chiếu với các kiểu dữ liệu khác khi được lập trình trong một ngôn ngữ.

Ngoài các dịch vụ thực thi, .NET cũng bao gồm các thư viện mở rộng. Những thư viện đó có thể hỗ trợ nhiều khối lượng công việc khác nhau. Chúng được tổ chức thành các *namespace*, *cung* cấp nhiều chức năng hữu ích. Các thư viện đó bao gồm mọi thứ, từ xử lý đầu vào và đầu ra tệp, thao tác chuỗi để phân tích cú pháp XML, cho đến các *framework* ứng dụng Web, và tùy biến Windows Forms.

Trong hệ sinh thái .Net,

.Net Core là một nền tảng mã nguồn mở, đa nền tảng được triển khai từ .Net, được cách tân, cải tiến để phù hợp với thời đại mà Điện toán đám mây (Cloud-computation) đang trở nên phổ biến, mà vẫn có thể tương thích đáng kể với .Net Framework (.Net cũ). Tuy nhiên cho tới bây giờ, bởi có những cải tiến đáng kể, .Net Core được xây dựng mới hoàn toàn và độc lập so với .Net Framework. Nó là một hệ thống trọn vẹn bao gồm nền tảng dịch mã trung gian và thực thi ứng dụng, các framework để phát triển các loại ứng dụng, và là hệ thống thư viện hỗ trợ.

Môi trường thực thi của .NET Core được gọi là CoreCLR. CoreCLR có thể chạy trên nhiều hệ điều hành. Hiện nay CoreCLR có thể hoạt động trên Windows, Linux và macOS. CoreCLR sử dụng một trình biên dịch trung gian tương tự JIT compiler của .NET Framework. Mặc dù tên gọi giống nhau nhưng JIT compilter của .NET Core không phải là JIT của .NET Framework. JIT của .NET Core có thể dịch mã IL sang mã máy của 3 nền tảng nó hỗ trợ[8].



Hình 2-2-3 Mô hình toolchain của .Net Core

Một sự khác biệt nữa về *runtime* nằm ở chỗ, CoreCLR và mã máy được tải và kích hoạt bởi một tiến trình khác như *dotnet.exe*, trong khi CLR được kiểm soát bởi hệ điều hành Windows. Với vai trò framework, .NET Core cung cấp khung sườn cho lập trình viên phát triển ứng dụng web (ASP.NET Core), phát triển ứng dụng *desktop* trên windows (Windows Forms và WPF, từ .NET Core 3.1). Về tính năng này, .NET Core và .NET Framework gần như tương đương nhau.

Với vai trò thư viện, .NET Core cũng cung cấp hệ thống class cho các ngôn ngữ nó hỗ trợ (hiện nay có C#, VB.NET và F#). Tuy nhiên ở đây cần nhấn mạnh rằng,

Thứ nhất, hệ thống thư viện của .NET Core và .NET Framework là hoàn toàn độc lập nhau. Tuy nhiên chúng đều là các file chứa mã trung gian IL cho nên về lý thuyết chúng có thể sử dụng thư viện của nhau. Trên thực tế, bắt đầu từ .NET 2.0 bạn có thể tham chiếu tới các thư viện viết trên .NET Framework. Điều này giúp việc chuyển đổi sang .NET Core dễ dàng hơn.

Tuy nhiên, việc tham chiếu này cũng có giới hạn. Thư viện xây dựng trên .NET Framework sẽ không chạy được trên .NET Core nếu nó phụ thuộc vào những API

không được .NET Core hỗ trợ. Thứ hai, đội ngũ phát triển .NET Core sử dụng lại nguyên vẹn của các thư viện cơ bản của .NET Framework. Điều này giúp lập trình viên dễ dàng chuyển đổi từ .NET Framework sang .NET Core.

Lấy ví dụ, cả trong .NET Framework và .NET Core đều có class Console (System.Console), trong đó đều có các phương thức như *Write, WriteLine, Read, ReadLine*. Do vậy, nếu bạn đã thành thạo C# (trên .NET Framework), bạn tiếp tục sử dụng nó trên .NET Core mà không cần học thêm gì về ngôn ngữ lập trình nữa. Nói chung, nếu bạn có thư viện ở dạng các POCO (Plain Olde C# Object) viết trên .NET Framework, bạn có thể dễ dàng tham chiếu tới và sử dụng nó trong project .NET Core.

Vậy tại sao tại ứng dụng này, tôi lại sử dụng .Net Core, cụ thể là phiên bản 3.1 cho việc lập trình hệ thống Server mà không dùng các ngôn ngữ hay framework khác, hay tôi lại không dùng .Net Framework ngay trong chính hệ sinh thái .Net này? Những lý do có thể kể đến là[9]:

- Hiệu năng của .Net core là rất tốt, khả năng xử lý các request rất nhanh, chịu được thời gian hoạt động dài, và đặc biệt bền bỉ khi xử lý các công việc nặng trong một thời gian, rất tốt so với các framework khác.
- Đối với .Net Framework, .Net Core rõ ràng là một bản nâng cấp đáng kể. Khả năng hoạt động đa nền tảng tốt hơn, cũng như hiệu năng và khả năng mở rộng cũng tốt hơn nhiều. Ngoài ra .Net core cũng được tích hợp Docker container, phù hợp với việc deploy tự động.

2.1.2. SQL Server

SQL Server[6] hay còn gọi là Microsoft SQL Server, viết tắt là MS SQL Server, là một phần mềm quản trị cơ sơ dữ liệu quan hệ được phát triển bởi Microsoft bao gồm: tạo, duy trì, phân tích dữ liệu,… dễ dàng sử dụng để lưu trữ cho các dữ liệu dựa trên tiêu chuẩn RDBMS – Relational Database Management System.

SQL Server được xây dựng dựa trên SQL, được tối ưu để có thể chạy trên môi trường cơ sở dữ liệu rất lớn lên đến Tera – Byte cùng lúc phục vụ cho hàng ngàn user. SQL Server cung cấp đầy đủ các công cụ cho việc quản lý từ nhận diện GUI đến sử dụng ngôn ngữ cho việc truy vấn SQL.

Để tạo nên một giải pháp hoàn chỉnh cho việc phân tích và lưu trữ dữ liệu một cách dễ dàng, SQL đã bao gồm các thành phần cơ bản như sau:

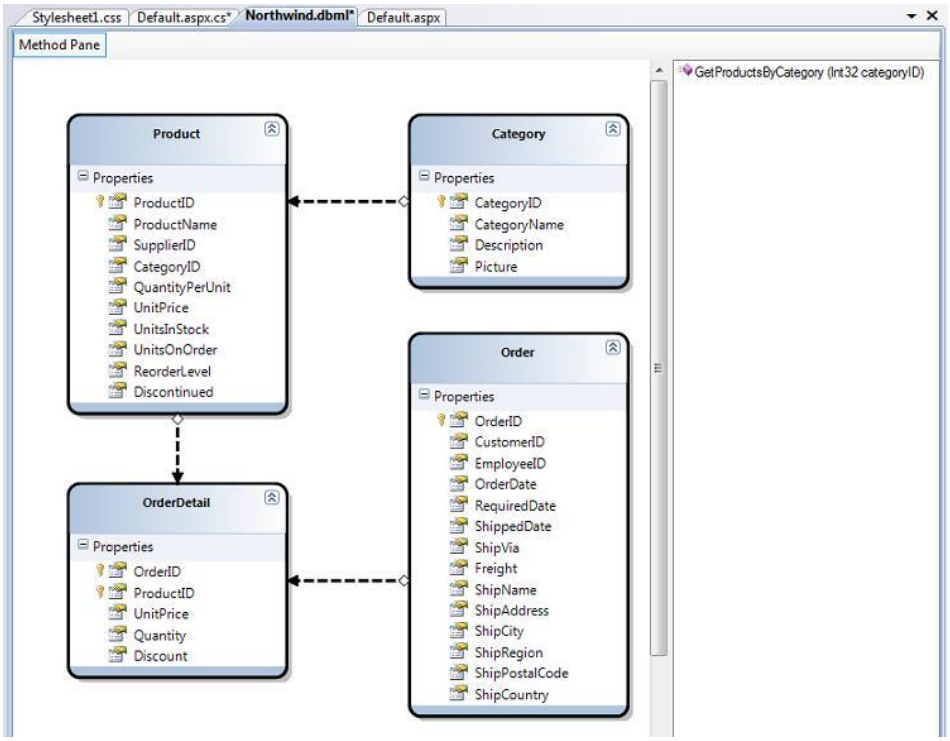
Hình 2-2-4 Các thành phần cơ bản của SQL Server

- **Database Engine**: chính là khả năng lưu trữ dữ liệu ở nhiều dạng quy mô dưới dạng *support* và *table*. Những dữ liệu được lưu trữ đều có thể tự điều chỉnh. Nó là dịch vụ cốt lõi để lưu trữ, xử lý và bảo mật dữ liệu. Database Engine cung cấp khả năng truy cập có kiểm soát và xử lý các request nhanh chóng để đáp ứng yêu cầu của các ứng dụng sử dụng dữ liệu khắt khe nhất.
- **Integration Services**: là một nền tảng để xây dựng các giải pháp tích hợp dữ liệu và chuyển đổi dữ liệu ở quy mô lớn. Các hệ thống sử dụng Integration Services để giải quyết các vấn đề phức tạp bằng cách sao chép hoặc tải xuống tệp, tải kho dữ liệu, dọn dẹp và khai thác dữ liệu cũng như quản lý các đối tượng và dữ liệu SQL Server.
- **Analysis Services**: là một công cụ phân tích dữ liệu được sử dụng trong hỗ trợ quyết định và phân tích một bài toán, chẳng hạn như các bài toán trong kinh doanh. Nó cung cấp các mô hình dữ liệu quy mô lớn cho các báo cáo và các ứng dụng phía client như Power BI, Excel, các báo cáo từ Reporting Services, và các công cụ trực quan hóa dữ liệu khác.
- **Notification Services:** đây là nền tảng cho sự phát triển và triển khai các ứng dụng soạn và gửi thông báo, có chức năng gửi thông báo theo dịch thời đến hàng ngàn người đăng ký sử dụng trên nhiều loại thiết bị khác nhau.

- **Reporting Services**: cung cấp một loạt các công cụ và dịch vụ để tạo, triển khai và quản lý các báo cáo.
- **Full Text Search Services**: Dịch vụ này cung cấp khả năng truy vấn và đánh chỉ mục dữ liệu văn bản không cấu trúc được lưu trữ trong các cơ sở dữ liệu của SQL Server.
- **Service Broker:** là một tính năng của SQL Server giám sát việc hoàn thành các tác vụ, thường là các thông báo lệnh, hoặc giữa hai ứng dụng khác nhau trong cơ sở dữ liệu. Nó chịu trách nhiệm về việc chuyển message một cách an toàn và đáng tin cậy từ đầu này đến đầu khác.

Với những đặc điểm trên, tôi có thể đưa ra một vài lý do của mình khi chọn SQL Server là công cụ quản lý Database cho Ứng dụng này như sau:

- Có nhiều công cụ hỗ trợ tốt, đặc biệt là Microsoft SQL Server Management.
- Hỗ trợ rất tốt khi hoạt động trên nền tảng .Net (thông qua Entity Framework Core)
- Được hỗ trợ tốt khi sử dụng trên Visual Studio
- Nhiều công cụ tích hợp tốt, đặc biệt với *Integration Services* chỉ cần kéo thả các bảng là tạo được database như ý muốn một cách rất trực quan



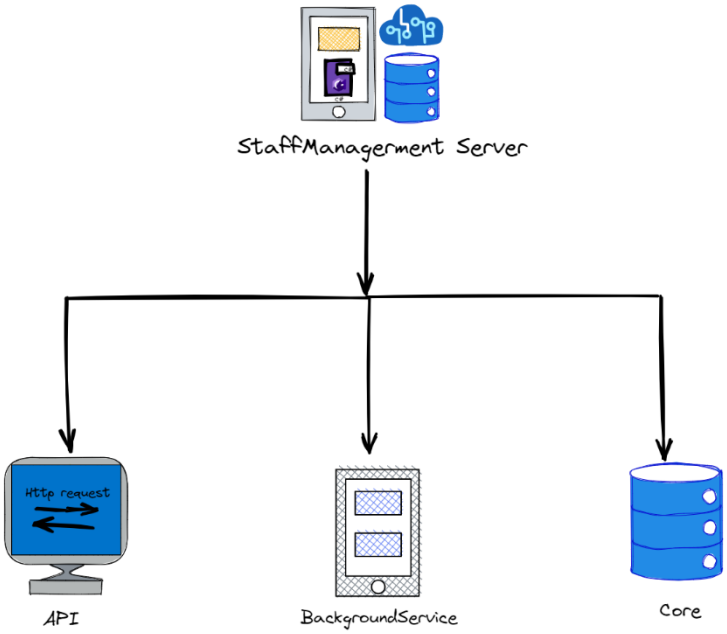
Hình 2-2-5 Ví dụ về khả năng kéo thả tạo Database của SQL Server

- v…v…

2.1.3. Chi tiết cấu tạo hệ thống Server

Như đã được giới thiệu ở trên, nền tảng để phát triển Server của hệ thống chính là .Net Core dùng C# và SQL Server. Khi phát triển Server, tôi đã chia thành 3 phần như sau:

StaffManagement.Core bao gồm những thành phần cơ sở như DbContext, Models, Helpers, Services, Repositories,… để cấu tạo nên các thành phần khác. Hay thành phần này chính là những phần chung của các thành phần còn lại, đảm bảo việc các thành phần khác có thể tái sử dụng các phần mã nguồn.

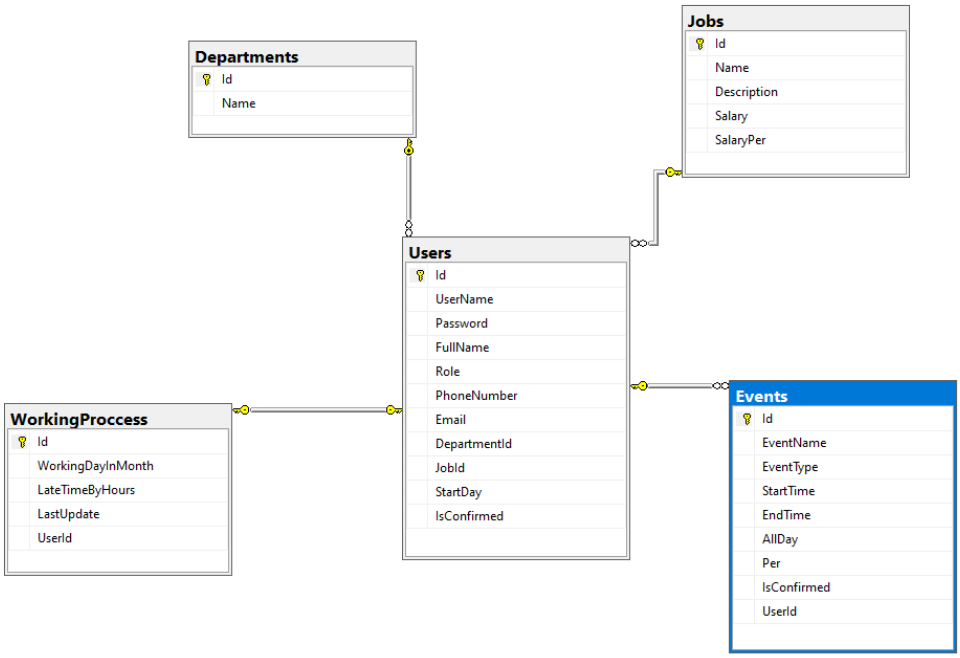


Hình 2-2-6 Cấu trúc chương trình của hệ thống Server

StaffManagement.API chính là phần API của Server. Thành phần này chịu trách nhiệm nhận và xử lý các request HTTP từ các Client và thiết bị chấm công. Đây cũng là thành phần rất quan trọng, đảm bảo việc hoạt động của tất cả các Client.

StaffManagement.BackgroundServices bao gồm các tác vụ chạy ngầm, hoạt động song song với nhau và với API. Các tác vụ này đảm bảo việc xử lý những công việc tốn thời gian, cần một lượng tài nguyên hệ thống nhất định và những công việc hoạt động theo lịch. Bằng cách để những tác vụ này hoạt động ngầm, chúng ta sẽ giảm được gánh nặng lên API, tăng thời gian xử lý các request cho API mà không làm gián đoạn các bước xử lý dữ liệu khác.

Đối với ứng dụng của chúng ta. Database sẽ được thiết kế như sau:



Hình 2-2-7 Diagram thiết kế Cơ sở dữ liệu của ứng dụng

Tên bảng	Cột	Ý nghĩa
User	Id	Mã định danh người dùng
	UserName	Tên tài khoản đăng nhập
	Password	Mật khẩu tài khoản đăng nhập
	FullName	Tên đầy đủ người dùng
	Role	Quyền của tài khoản
	PhoneNumber	Số điện thoại người dùng
	Email	Địa chỉ email của người dùng
	DepartmentId	Id bộ phận người dùng làm việc

	JobId	Id nghề của người dùng
	StartDay	Ngày người dùng bắt đầu làm việc
	IsConfirmed	Tài khoản người dùng đã được xác nhận hay chưa
Event	Id	Mã định danh của Event
	EventName	Tên hoặc mô tả Event
	EventType	Kiểu Event
	StartTime	Ngày bắt đầu Event
	EndTime	Ngày kết thúc Event
	AllDay	Event có diễn ra cả ngày hay không (Đối với Event chỉ có StartTime)
	Per	Event lặp lại theo chu kỳ nào (ngày, tháng năm) ?
	IsConfirmed	Event đã được xác nhận chưa ?
	UserId	Id của người dùng Event thuộc về (Null khi Event chung của Công ty)
WorkingProcess	Id	Mã định danh thông tin tiến trình làm việc
	WorkingDayInMonth	Số ngày công trong một tháng
	LateTimeInMonth	Số giờ đi muộn/về sớm trong một tháng
	LastUpdate	Ngày cuối cùng cập nhật bản ghi
	UserId	Id người dùng tiến trình làm việc này thuộc về
Job	Id	Mã định danh nghề nghiệp

	Name	Tên nghề nghiệp
	Description	Mô tả nghề nghiệp
	Salary	Tiền lương cơ bản
	SalaryPer	Lương cơ bản tính theo giờ, ngày hay tháng ?
Department	Id	Mã định danh bộ phận
	Name	Tên bộ phận

User: Chứa thông tin người dùng của ứng dụng, mỗi bản ghi ứng với một tài khoản.

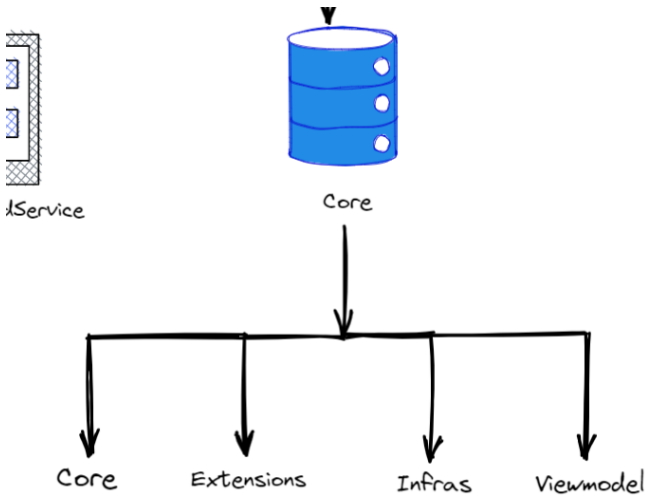
Event: Chứa thông tin của các sự kiện, bao gồm sự kiện ghi nhớ, ngày nghỉ, các thời gian checkin/check-out,…

WorkingProccess: Chứa thông tin tiến độ đi làm của một tài khoản trong tháng hiện tại.

Job: Chứa thông tin về nghề nghiệp trong công ty cũng như cách tính lương của nghề nghiệp đó theo số công

Department: Chứa thông tin về bộ phận, phòng ban trong công ty

2.1.3.1. StaffManagement.Core



Hình 2-2-8 Cấu trúc của Core hệ thống Server

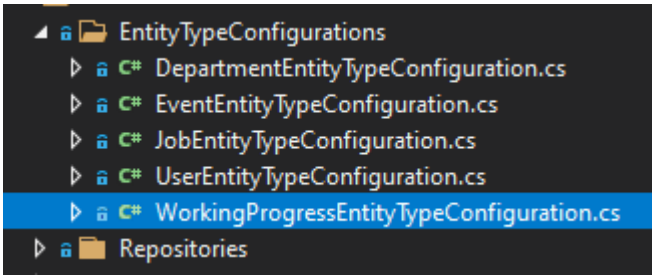
DbContext đại diện cho sự kết hợp của Unit Of Work và Repository Pattern (một trong các Design Pattern) để nó có thể được sử dụng để truy vấn từ cơ sở dữ liệu và nhóm các thay đổi lại với nhau sau đó sẽ được ghi lại vào store dưới dạng một đơn vị. DbContext về mặt khái niệm tương tự như ObjectContext. Có thể hiểu DbContext trong chương trình sẽ đại diện cho một Database ở SQL Server, nó sẽ tham chiếu, tạo các truy vấn tới database thông qua kết nối SQL được đăng ký tại các **Extension.** Trong DbContext cũng sẽ chứa các *DbSet<T>* tương ứng với các bảng trong Database đó. Nhờ đó, khi ta muốn tương tác, sử dụng một bảng trong Database như truy vấn, tạo bản ghi, sửa đổi bản ghi hay xóa, ta chỉ cần tương tác ngày trên chính *DbSet* của bảng đó trong Database.

Đối với Database của chúng ta, ta sẽ có *MsSqlStaffManagementDbContext* đảm nhiệm là DbContext của Database StaffManagement. Trong DbContext này cũng có các DbSet tham chiếu tới các bảng trong Database như sau:

```
0 references | Nguyen Ngoc Minh, 65 days ago | 1 author, 1 change
public DbSet<User> Users { get; set; }

0 references | Nguyen Ngoc Minh, 65 days ago | 1 author, 1 change
public DbSet<Event> Events { get; set; }

0 references | Nguyen Ngoc Minh, 65 days ago | 1 author, 1 change
public DbSet<Department> Departments { get; set; }

0 references | Nguyen Ngoc Minh, 65 days ago | 1 author, 1 change
public DbSet<Job> Jobs { get; set; }

0 references | Nguyen Ngoc Minh, 59 days ago | 1 author, 1 change
public DbSet<WorkingProgress> WorkingProgresses { get; set; }
```

Hình 2-2-9 Các DbSet được khai báo trong DbContext

Ứng với mỗi bảng ở DbSet, chúng ta có một Model để bảng thực hiện việc casting dữ liệu vào đối tượng đó. Ngoài việc tạo Model để lấy giá trị các bản ghi trong bảng, .Net còn hỗ trợ việc thiết lập các thông tin của bảng thành các Entity, hỗ trợ việc xác lập mối quan hệ giữa các Model với nhau như trong Database, không những vậy những thiết lập này còn giúp ta Migrate Database dựa trên chúng, giảm bớt bước tạo từng

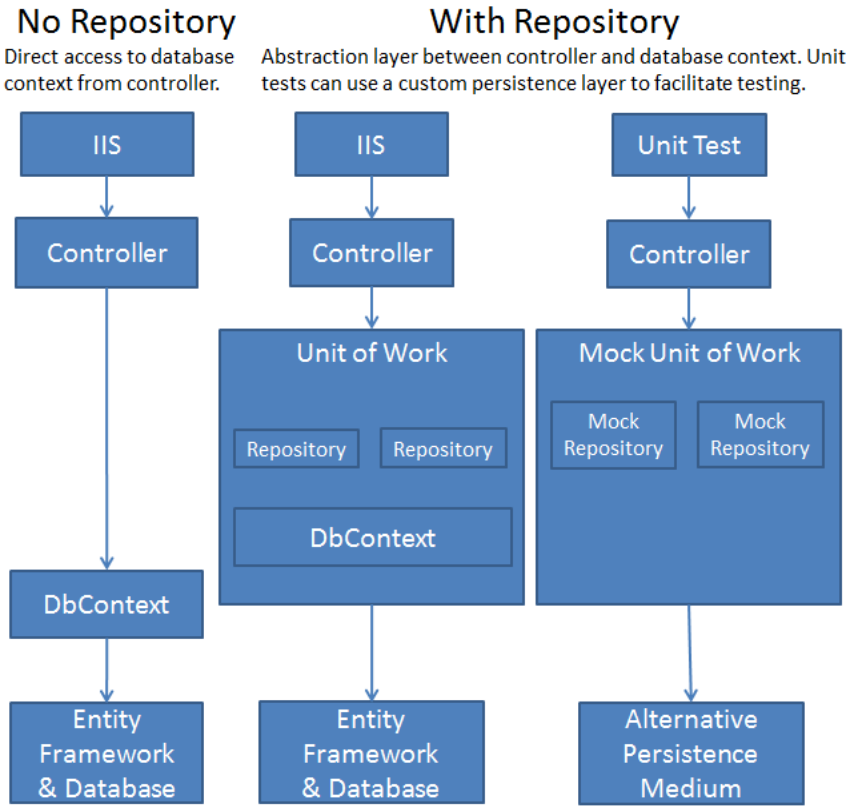


Hình 2-2-10 Các EnityConfiguration của các bảng Database

database, từng bảng một cách thủ công.

Nói về việc Migration Database, tính năng này cũng được .Net chú trọng, vì nó giảm được thời gian đáng kể và tránh được những sai sót so với việc tạo Database thủ công mỗi khi deploy ứng dụng lên một môi trường mới. Tính năng seeding dữ liệu, tạo các dữ liệu giả nhằm mục đích kiểm thử hệ thống khi chưa có người dùng cũng được tích hợp vào công việc Migration Database.

Reposistory là một thành phần cần thiết sau khi thiết lập xong DbContext cho Database trong SQL Server. Repository và Unit of Work nhằm tạo ra một lớp trừu



Hình 2-2-11 Lợi ích của Repository trong việc Mock test

tượng giữa lớp truy cập dữ liệu (data access layer) và lớp logic nghiệp vụ (business logic layer) của một ứng dụng. Việc triển khai các *pattern* này có thể giúp cách ly ứng dụng khỏi những thay đổi trong kho dữ liệu và có thể tạo điều kiện thuận lợi cho việc kiểm tra tự động (automatic unit-testing) hoặc Test-Driven Development (TDD)[10].

Do đó việc có thêm một lớp Repository giữa Services/Controller và DbContext là điều cần thiết. Trong hệ thống của ứng dụng này, Repository cũng sẽ đảm nhiệm là lớp trung gian giữa DbContext và Service. Trong Repository, ta sẽ để dành cho các hàm

thực hiện các công việc thao tác trên Database là các hàm CRUD (Creat, Read, Update, Delete).

```
using StaffManagement.Core.Core.Common;
using System.Threading;
using System.Threading.Tasks;

namespace StaffManagement.Core.Core.Persistence.Repositories
{
    5 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    public interface IBaseRepository<TModel>
    {
        7 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
        void Create(TModel obj);
        16 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
        Task<QueryResult<TModel>> GetAsync(QueryParams<TModel> @params, CancellationToken cancellationToken);
        8 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
        void Update(QueryParams<TModel> @params, TModel obj);
        5 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
        void Delete(QueryParams<TModel> @params);
    }
}
```

Hình 2-2-12 Interface Base Repository - class cha của các Repository khác

Đây là các thao tác cơ bản nhất trong Database. Các Services sẽ đảm nhận việc sử dụng các thao tác này để thực hiện các yêu cầu của chương trình.

Ứng với mỗi bảng trong Database, ta sẽ có một Repository của riêng chúng. Các Reposistory này sẽ kế thừa *BaseRepository* vì chúng đều có chung các phương thức CRUD, điều này sẽ đảm bảo tính DRY (Don't Repeat Yourself) trong mã nguồn dự án.

Kế tiếp của lớp Repository, ta sẽ có các **Service**. Như đã giới thiệu ở trên, Services là lớp đảm nhiệm việc sử dụng các thao tác cơ bản trong Repository để thực hiện các yêu cầu riêng biệt mà ứng dụng cần.

```
11 references | Nguyen Ngoc Minh, 69 days ago | 1 author, 2 changes
public interface IEventService
{
    3 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task CreateEventAsync(Event @event, CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task CreateVacationAsync(Event @event, CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task RegisterEventAsync(string userName, DateTime startTime, CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task<Event> QueryEventByIdAsync(long id, CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task<QueryResult<Event>> QueryEventsByUserIdAsync(QueryEventRequest request, CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 69 days ago | 1 author, 1 change
    Task<QueryResult<Event>> QueryEventsByUserIdAsync(QueryEventRequest request, DateTime date, CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 69 days ago | 1 author, 1 change
    Task<QueryResult<Event>> QueryRegisterEventsByUserIdAsync(long userId, CancellationToken cancellationToken = default);

    1 reference | Nguyen Ngoc Minh, 69 days ago | 1 author, 1 change
    Task<QueryResult<Event>> QueryRegisterEventsByUserIdAsync(long userId, DateTime date, CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task<UserResult> QueryUnconfirmedEventsAsync(CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task<QueryResult<Event>> QueryUnconfirmedEventsByUserIdAsync(QueryEventRequest request, CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task<QueryResult<Event>> QueryMonthEventsByUserId(long userId, CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task<QueryResult<Event>> QueryVacationEventByUserIdAsync(QueryEventRequest request, CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task<QueryResult<Event>> QueryCompanyEventsAsync(CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task UpdateEventAsync(Event request, CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task AcceptEventAsync(Event request, CancellationToken cancellationToken = default);

    2 references | Nguyen Ngoc Minh, 71 days ago | 1 author, 1 change
    Task DeleteEventAsync(QueryEventRequest request, CancellationToken cancellationToken = default);
```

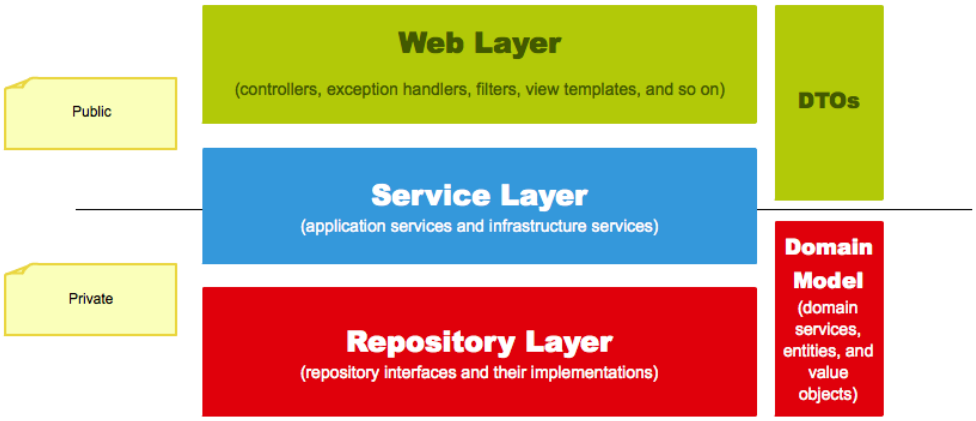
Hình 2-2-13 Các phương thức trong Event Service

Do đó, khác với việc mỗi bảng sẽ có một Repository, Service được tạo ra dựa theo các yêu cầu nên sẽ có rất nhiều các Service khác nhau, tuỳ thuộc vào nhu cầu của ứng dụng. Đặc biệt trong đó là 2 Service:

- **TokenService**: Service dùng để tạo, xác nhận một token, hoặc mã hoá một đoạn thông tin thành token đó.
- **AuthService**: Service dùng để xác nhận người dùng dựa trên token định danh trong request, hoặc để xác nhận người dùng khi đăng nhập.

Đó là 2 Service quan trọng, sử dụng xuyên suốt với các service khác nhằm đảm bảo việc xác thực một request có đúng là của một người dùng hợp lệ hay không.

Và trong các Service, ta có các **DTO**, viết tắt của Data Transfer Object. Đây là các class đóng gói data để chuyển giữa Client - Server hoặc giữa các Service trong hệ thống. Việc đóng gói dữ liệu như vậy để đảm bảo khi dữ liệu được chuyển giao cho Client sẽ không chứa bất kì trường dữ liệu nhạy cảm nào.



Hình 2-2-14 Vị trí các DTO trong các lớp của hệ thống

Như hình trên ta có thể thấy, các Repository khi lấy dữ liệu từ Database thông qua DbContext, các dữ liệu sẽ được chuyển sang các Object có kiểu dữ liệu chính là các Model đã được khai báo trong EntityConfiguration. Tại đây các Object sẽ có tất cả các trường ứng với các cột trong Database. Đây sẽ là kiểu dữ liệu được Repository giao tiếp, vận chuyển giữa chúng và chuyển qua cho Service. Tất cả những thao tác trên đều được thực hiện sâu trong hệ thống nên dữ liệu sẽ không bị lộ ra ngoài, trừ phi được ghi lại bởi công cụ ghi Log của hệ thống đó. Những khi các Service giao tiếp với nhau, đặc biệt là với Client, dữ liệu sẽ được hiển thị rõ ràng trong các response mà API trả về. Ví dụ với Model User, ta có trường nhạy cảm là Password. Khi ta trả về thông tin về User đó cho Client, nếu ta sử dụng luôn Object Model User từ Repository thì Client sẽ lấy được dữ liệu Password từ response, khiến cho việc bảo mật của ứng dụng

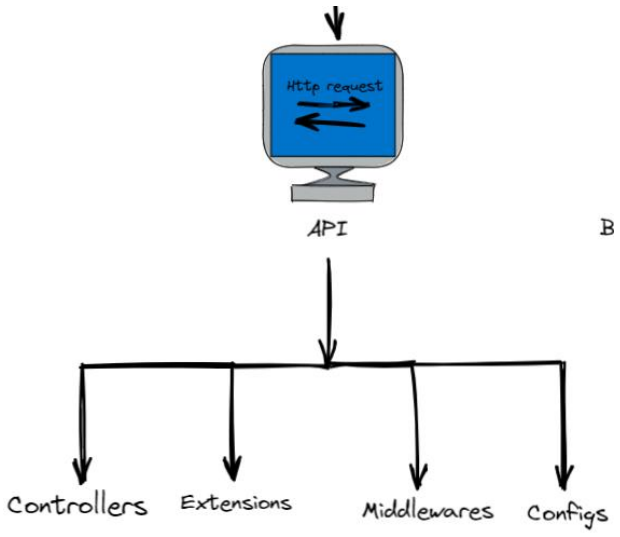
gặp vấn đề lớn. Đấy là lý do ta có thêm một lớp Data nữa sau Model là DTO. Các DTO đều dựa trên các trường dữ liệu của Model nó trường khi chuyển thành, và loại bỏ đi các trường dữ liệu nhạy cảm như Password, số CCCD, số tài khoản,… để khi Client nhận được response sẽ không thể thấy được những dữ liệu đó. Ngoài ra có những dữ liệu không cần thiết cho Client cũng sẽ được loại bỏ đi để giảm bớt dung lượng của các response tới Client.

Ngoài những thành phần quan trọng kể trên, Core còn chứa những thành phần phụ như sau:

- Helpers: tập hợp các phương thức hỗ trợ các thành phần trên, như các hàm tính toán, hàm kiểm tra, ….
- ViewModels: là một lớp Object, thực hiện việc casting các request từ API trước khi chuyển qua DTO cho Service, cũng như định dạng các dữ liệu trong response về Client từ các dữ liệu DTO nhận được từ Service
- Common: Chứa các định nghĩa về các Object khác thường được dùng như Param, Result, Enum, …
- Context: Chứa định nghĩa về các Object chứa dữ liệu về Context

2.1.3.2. StaffManagement.API

Đây là thành phần rất quan trọng của ứng dụng, là phần xử lý các request từ phía Client và máy chấm công.



Hình 2-2-15 Cấu trúc của API hệ thống Server

Ở thành phần API, ta có các **Controller**. Các Controller đảm nhiệm việc quy định các route cho các request HTTP khi gửi tới API. Đồng thời Controller cũng sẽ quy định việc xử lý các request đó dựa trên các phương thức có trong Service như thế nào.

Route	**Ý nghĩa**
Department	
POST: api/department	Tạo một Department
GET: api/department/id	Lấy thông tin về một Department theo ID
GET: api/department	Lấy thông tin tất cả các Department
PUT: api/department	Thay đổi thông tin một Department
DELETE: api/department/id	Xoá một Department theo ID
Job	
POST: api/job	Tạo một Job
GET: api/job/id	Lấy thông tin về một Job theo ID
GET: api/job	Lấy thông tin tất cả các Job
PUT: api/job	Thay đổi thông tin một Job
DELETE: api/job/id	Xoá một Job theo ID
Authentcation Controller	
POST: api/authentication	Xử lý request đăng nhập của Client
User	
POST: api/user/create	Tạo một User
GET: api/user/id	Lấy thông tin về một User theo ID
GET: api/user	Lấy thông tin tất cả các User
GET: api/user/events	Lấy thông tin về các Event của mọi User
GET: api/user/working	Lấy thông tin về WorkingProgress của mọi User
PUT: api/user/edit	Thay đổi thông tin một User
DELETE: api/user/delete/id	Xoá một User theo ID
Event	
POST: api/event/create	Tạo một Event chung của công ty
POST: api/event/create-event	Tạo một Event Sự kiện của User theo

	UserId
POST: api/event/create-vacation	Tạo một Event Ngày nghỉ của User theo UserId
POST: api/event/register	Tạo một Event Check-In của User theo UserId
GET: api/event/id	Lấy thông tin về một Event theo ID
GET: api/event	Lấy thông tin tất cả các Event của công ty
GET: api/event/user/userId	Lấy thông tin tất cả Event của User theo UserId
GET: api/event/unconfirmed	Lấy thông tin tất cả các Event chưa xác nhận
GET: api/event/unconfirmed/user/userId	Lấy thông tin tất cả các Event chưa xác nhận của User theo UserId
PUT: api/event	Thay đổi thông tin một Event
PUT: api/event/confirmed	Xác nhận một Event
DELETE: api/event/id	Xoá một event theo ID

Các Controller sẽ chỉ sử dụng các Service để thực hiện các yêu cầu, các dữ liệu khi được đưa tới Controller đều được casting sang DTO như đã đề cập ở trên.

IoCRegisterExtensions đảm nhiệm việc đăng ký các thành phần IoC (Inversion of Control) trong chương trình.

```
public static void AddPersistence(this IServiceCollection services, IConfiguration configuration)
{

    services.AddScoped(sp =>
    {
        return new MsSqlStaffManagementDbContext(configuration.GetConnectionString(PersistenceConnectionKey));
    });

    /*
    services.AddDbContext<MsSqlStaffManagementDbContext>(options =>
    {
        options.UseSqlServer(configuration.GetConnectionString(PersistenceConnectionKey));
    });*/

    services.AddScoped<IUnitOfWork>(sp => sp.GetRequiredService<MsSqlStaffManagementDbContext>());

    services.AddScoped<IUserRepository, UserRepository>();
    services.AddScoped<IEventRepository, EventRepository>();
    services.AddScoped<IDepartmentRepository, DepartmentRepository>();
    services.AddScoped<IJobRepository, JobRepository>();
    services.AddScoped<IWorkingProgressRepository, WorkingProgressRepository>();
}

1 reference | Nguyen Ngoc Minh, 71 days ago | 1 author, 2 changes
public static void AddCoreService(this IServiceCollection services)
{
    services.AddScoped<IAuthTokenService, JwtAuthService>();
    services.AddScoped<IAuthUserService, AuthUserService>();
    services.AddScoped<IUserService, UserService>();
    services.AddScoped<IEventService, EventService>();
    services.AddScoped<IJobService, JobService>();
    services.AddScoped<IDepartmentService, DepartmentService>();
    services.AddScoped<IWorkingProgressService, WorkingProgressService>();
}

1 reference | Nguyen Ngoc Minh, 92 days ago | 1 author, 1 change
public static void AddAuthenticationContext(this IServiceCollection services)
{
```

Hình 2-2-16 IoCRegisterExtension thực hiện việc đăng ký các thành phần

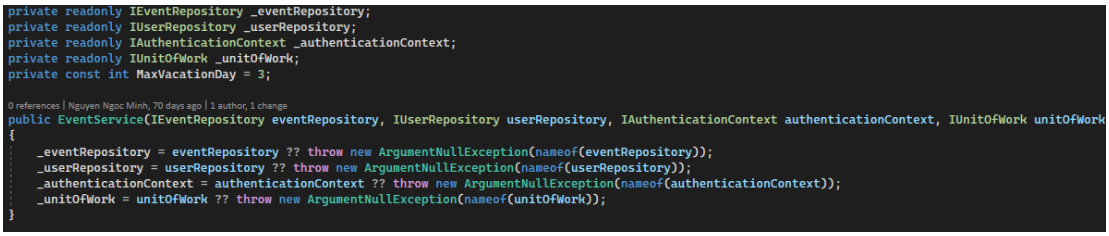
Như đã thấy trong mã nguồn của ứng dụng, ta rất dễ bắt gặp các Interface của một Service hay một Repository. Lý do cho sự có mặt của các Interface này chính là do mã nguồn C# trong ứng dụng được viết tuân theo các quy tắc SOLID[11] là:

- Single responsibility principle
- Open/closed principle
- Liskov substitution principle
- Interface segregation principle
- Dependency inversion principle

Các quy tắc trên được tạo ra cho tất cả các mã nguồn trên thế giới tuân theo nhằm đảm bảo mã nguồn trở thành Clean Code, tức là khi được phát triển, mã nguồn của ứng dụng sẽ dễ đọc, dễ hiểu cho nhiều lập trình viên tham gia vào dự án, dễ mở rộng và bảo trì. Cách sử dụng Interface cho các Service và Repository chính là do áp dụng quy tắc *Dependency Inversion*, hay *Depenency Injection*, hay *Inversion of Control*. Quy tắc này quy định như sau:

- Các module cấp cao không nên phụ thuộc vào các modules cấp thấp. Cả 2 nên phụ thuộc vào abstraction.
- Interface (abstraction) không nên phụ thuộc vào chi tiết, mà ngược lại. (Các class giao tiếp với nhau thông qua interface, không phải thông qua implementation.)

Và chính C# hỗ trợ việc áp dụng Dependency Injection vào mã nguồn, nên việc áp dụng quy tắc này trong phát triển ứng dụng .Net là không thể thiếu. Do đó mỗi lớp của Service hay Repository đều được kế thừa từ một Interface chứa các quy định về các phương thức của lớp đó. Khi một Controller khởi tạo một Service để làm việc, hoặc Service khởi tạo một Repository, ta sẽ gọi tới Interface thay vì chính lớp đó.



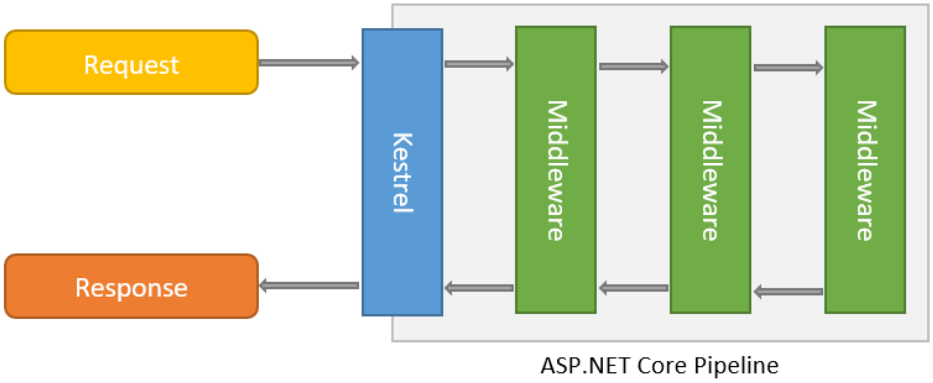
Hình 2-2-17 Ví dụ về khai báo interface trong Service, áp dụng Dependency Injection

Tại IoCRegisterExtensions, ta sẽ đăng ký cho các lớp dựa trên interface nó kế thừa, và vòng đời của từng lớp đó khi được khởi tạo. .Net cung cấp cho chúng ta 3 kiểu vòng đời để khai báo đó là:

- Transient: Mỗi khi gọi đến hàm khởi tạo, sẽ khởi tạo một đối tượng mới hoàn toàn.
- Scoped: Một đối tượng khi được khởi tạo sẽ tồn tại trong một request, kể cả khi gọi tới hàm khởi tạo thì cũng sẽ sử dụng đối tượng được tạo ra cho tới khi kết thúc request đó. Mỗi request, các đối tượng được khởi tạo sẽ khác nhau.
- Singleton: Một đối tượng khi được khởi tạo sẽ tồn tại vĩnh viễn trong quá trình ứng dụng chạy. Các requet đều sử dụng chung một đối tượng.

Tại API này, các Service, Repository, DbContext, Context được đăng ký vòng đời là Scoped nhằm đảm bảo việc tối ưu hoá việc sử dụng đối tượng khởi tạo trong mỗi request khi API xử lý. Do đó, các đối tượng sẽ được tạo và sử dụng trong thời gian xử lý 1 request riêng biệt, không ảnh hưởng tới các request khác. Điều này cũng giúp cho việc xử lý được đồng thời nhiều request với nhau trong việc lập trình bất đồng bộ.

Middlewares là cầu nối giữa tương tác của người dùng và hệ thống. Đóng vai trò trung gian giữa request/response và các xử lý logic bên trong web server. Đây là một thành phần rất phổ biến trong các Web Server.



Hình 2-2-18 Vị trí các Middleware trong hoạt động xử lý request của API

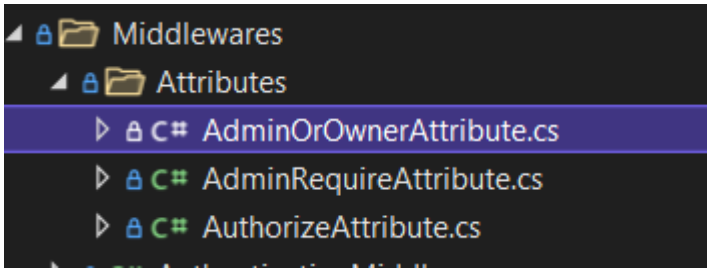
Tại ứng dụng này, ta có 2 Middleware tiêu biểu đó là:

- **Authentication Middleware**: Middleware này sẽ tiền xử lý các request, đọc token bearer nằm trong Header của request, tiến hành xác nhận hợp lệ token, và lấy thông tin người dùng từ token đó để tạo Context định danh cho quá trình xử lý token đó.

- **ErrorHandling Middleware**: Middleware này sẽ xử lý các response tới client chứa thông tin về lỗi, tiến hành định dạng lại thông báo lỗi thành định dạng phù hợp để Client đọc được và hiển thị lên phía màn hình người dùng.

Đối với Authentication Middleware, khi nó nhận được một request và tiến hành tạo một Context chứa định danh của User gửi request đó, Middleware này cũng lưu thông tin về phân quyền của người dùng. .Net cung cấp tính năng tạo các Attribute để tiến hành việc kiểm tra trong Context phân quyền người dùng trước khi quyết định xem request đó có được quyền thực hiện hành động trong Controller không.

Các Attribute khi được định nghĩa, sẽ được đặt ở phía trên những hàm định nghĩa Route trong Controller, nhằm xác định những kiểu người dùng nào được phép truy cập

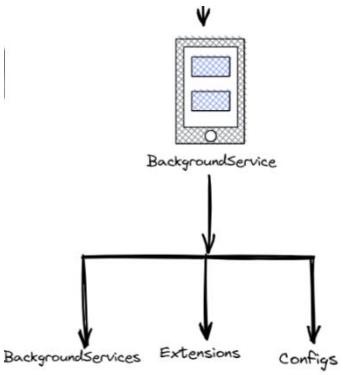


Hình 2-2-19 Các attribute trong API

Route đấy. Những request không hợp lệ sẽ được trả về Exception thông báo không đủ quyền, và tiến hành việc redirect về trang chủ.

2.1.3.3. StaffManagement.BackgroundService

Là một thành phần đảm nhiệm việc thực thi các tác vụ ngầm, đây là thành phần sẽ chạy xuyên suốt, song song với API.



Hình 2-2-20 Cấu trúc BackgroundService trong hệ thống Server

.Net hỗ trợ việc tạo ra các BackgroundService chạy song song với API hay ứng dụng nhằm hỗ trợ việc xử lý các tác vụ ngầm, tối ưu hoá thời gian phản hồi của API tới các Client.

BackgroundService tại ứng dụng này sẽ đảm nhiệm vai trò chạy các tác vụ sau:

- Xử lý các sự kiện điểm danh, chuyển các sự kiện điểm danh trong ngày của nhân viên trở thành 2 sự kiện duy nhất là Check-in và Check-out.
  Khi một nhân viên điểm danh bằng máy, hệ thống sẽ sinh ra sự kiện Check-in. Khi được máy nhận diện nhiều lần trong một ngày, các sự kiện điểm danh sẽ được sinh ra mà không xác định được đâu là lần điểm danh cuối cùng của nhân viên. Do đó tác vụ này sẽ chạy ở cuối ngày để lấy ra sự kiện điểm danh đầu tiên và cuối cùng của nhân viên để đưa về thành sự kiện Check-in và Check-out
- Tính toán WorkingProccess của User dựa theo sự kiện Check-in và Check-out trong ngày nhằm hỗ trợ tính lương tháng đó.

Hai tác vụ này sẽ được thiết lập chạy vào cuối mỗi ngày, đảm bảo tất cả các sự kiện trong ngày đó trở về trước đều được xử lý và cập nhật vào trong Database.

IoCRegistrationExtensions trong BackgroundService cũng làm nhiệm vụ tương tự như ở phía API là đăng ký vòng đời và lớp tổng quát cho các Service hoạt động trong thành phần này. Khác với ở API, khi các vòng đời đều được đăng ký là Scoped để tối ưu hoá việc xử lý các request, BackgroundService được đăng ký hoạt động theo vòng đời Singleton, tức là chỉ khởi tạo đúng một lần trong suốt quá trình nó hoạt động.

```
using (IServiceScope scope = _serviceProvider.CreateScope())
{
    _logger.LogInformation("Event worker is starting .....");

    _userService = scope.ServiceProvider.GetService<IUserService>();
    _eventService = scope.ServiceProvider.GetService<IEventService>();
    _workingProgressService = scope.ServiceProvider.GetService<IWorkingProgressService>();

    var users = await _userService.QueryUsersAsync();

    foreach (var user in users.Users)
    {
        await _eventService.ProccessRegisterEventsByUserIdAsync(user.Id);
        await _workingProgressService.UpdateWorkingDayAsync(user.Id);

        _logger.LogInformation("Update user " + user.UserName + "'s register events successfully!");
    }
}
```

Hình 2-2-21 Sử dụng ServiceProvider để tạo các Scoped trong BackgroundService'

Lý do cho việc đăng ký này là do BackgroundService được .Net hỗ trợ khi đăng ký sẽ tự động hoạt động ở vòng đời Singleton, chính đối tượng được khởi tạo này sẽ chạy ngầm liên tục trong suốt quá trình chạy BackgroundService và không khởi tạo lại ở bất kỳ thời điểm nào. Do đó để sử dụng được các Service hoặc Repository vòng đời Scope, mỗi khi BackgroundService sử dụng chúng trong thời điểm nó hoạt động (là cuối ngày đối với ứng dụng này), nó sẽ xem như đây là một request để xử lý, và dùng ServiceProvider của .Net để tạo nên một vùng Scope chứa các đối tượng vòng đời Scope hoạt động phía trong.

2.2. Client

Client của ứng dụng được phát triển từ Angular2 sử dụng Typescript. Đây là một framework javascript phổ biến trong việc phát triển các ứng dụng Web hay các ứng dụng đa nền tảng.

2.2.1. Angular2

Angular2 (phân biệt với phiên bản AngularJS) là một framework ứng dụng Web mã nguồn mở và miễn phí, sử dụng Type-script, được nghiên cứu và dẫn đầu bởi đội ngũ Angular đến từ Google, và được phát triển bởi đội ngũ và cộng đồng đông đảo lập trình viên đang sử dụng nó.

Angular2 được phát triển như là một framework front-end, phát triển các trang web động. Angular2 sử dụng Type-script, một ngôn ngữ lập trình có nền tảng là Javascript, với những đặc điểm vượt trội hơn như: cung cấp khả năng lập trình hướng đối tượng, cung cấp các tính năng trong lập trình như static, interface, … mà Javascript chưa hề sở hữu. Ngoài ra so với những framework front-end sử dụng Javascript, Angular2 còn có thể phát hiện ra các lỗi biên dịch khi lập trình, nhờ đó giảm thiểu thời gian sửa lỗi trong chương trình. Nhờ những ưu điểm đó, Angular2 sử dụng Type-script có thể loại bỏ những đoạn mã không cần thiết và đảm bảo các ứng dụng nhẹ hơn và nhanh hơn.

Angular2 sử dụng HTML để xây dựng giao diện người dùng của ứng dụng. HTML, so với JavaScript, là một ngôn ngữ ít phức tạp hơn. Nó cũng là một ngôn ngữ trực quan và điều hướng với các lệnh như *ng-app*, *ng-model*, *ng-repeat* và *form control*. Với sự trợ giúp đó, lập trình viên sử dụng Angular2 không cần phải đầu tư quá nhiều thời gian vào các luồng chương trình và quyết định tải cái gì trước. Chỉ cần xác định những gì họ yêu cầu và Angular2 sẽ đảm nhiệm những phần việc còn lại.

Với các ứng dụng chạy trên thiết bị thiết bị di động , Angular2 cung cấp Progressive Web Application (PWA). Đây là một tính năng hữu ích, cho phép các trang web hoạt

động giống như các ứng dụng dành cho thiết bị di động. Nó làm giảm sự phụ thuộc vào mạng giúp cải thiện đáng kể trải nghiệm người dùng của trang web.

Bộ nhớ đệm trong PWA hoạt động hiệu quả và tiết kiệm băng thông bất cứ khi nào có thể. Điều này giảm thiểu rủi ro khi tải các nội dung đã lỗi thời. Hơn nữa, vì nó là một trang web, nó có thể được tối ưu hóa cho hoạt động SEO. Angular2 cũng tạo điều kiện phát triển các ứng dụng trang đơn (SPA) cung cấp khả năng render phía máy chủ, giúp tăng thứ hạng SEO. Nó cũng giúp tải trang đầu tiên nhanh chóng và cải thiện hiệu suất trang web trên thiết bị di động và thiết bị cấu hình yếu.

Angular2 được phát triển có cấu trúc tương đồng với một framework MVVM (Model-View-View-Model). Do đó nó đảm bảo công việc phát triển ứng dụng dễ dàng vì nó loại bỏ các nhu cầu không cần thiết về những phần code. Angular2 có kiến trúc MVC được đơn giản hóa, khiến cho việc viết getters và setters trở nên không cần thiết. Các *directive* có thể được quản lý bởi một số nhóm khác vì đây không phải là một phần của mã ứng dụng. Nói chung, các nhà phát triển được đảm bảo sẽ phải viết phần mã nguồn ngắn hơn, cùng với các ứng dụng nhẹ hơn và nhanh hơn.

Angular tổ chức mã nguồn ứng dụng thành các *bucket*, cho dù đó là *component*, *directive*, *pipeline* hay *service*. Ta cũng có thể gọi các nhóm này là các mô-đun. Các mô-đun làm cho việc tổ chức, sắp xếp các chức năng của ứng dụng trở nên dễ dàng, tách biệt nó thành các tính năng và các khối có thể tái sử dụng. Các mô-đun cũng cho phép sử dụng Lazy-loading, tạo tiền đề cho tính năng tải ứng dụng trong nền hoặc theo yêu cầu.

Angular2 dựa trên các *components* bắt đầu theo cùng một kiểu. Chẳng hạn, mỗi *component* đặt đoạn mã trong một class hoặc định nghĩa một *decorator;* @Component từ *metadata*. Các *component* này là các thành phần giao diện nhỏ độc lập với nhau và cung cấp cho lập trình viên một số lợi ích, bao gồm:

- **Khả năng tái sử dụng**: Cấu trúc dựa trên các *component* của Angular làm cho các *component* có khả năng tái sử dụng cao trên ứng dụng. Chúng ta có thể xây dựng UI (Giao diện người dùng) với các bộ phận thường xuyên có sự thay đổi, đồng thời đảm bảo quá trình phát triển suôn sẻ cho nhà phát triển.
- **Đơn giản hoá quá trình Unit-test**: Do việc nằm độc lập với nhau, các *component* giúp việc unit-test diễn ra dễ dàng hơn.
- **Cải thiện khả năng đọc hiểu mã nguồn**: Tính nhất quán trong mã hóa khiến việc đọc các đoạn mã nguồn trở thành một công việc dễ dàng đối với các nhà

lập trình viên mới trong một dự án đã và đang diễn ra. Điều này giúp cho năng suất của họ và hiệu quả tổng thể của dự án tăng lên rất nhiều.

- **Dễ dàng bảo trì, mở rộng**: Các *component* tách rời có thể thay thế bằng các triển khai của nó một cách tốt hơn. Nói một cách đơn giản, nó cho phép việc cập nhật, bảo trì và mở rộng mã nguồn trở nên hiệu quả hơn.

Hình 2-2-22 Các lợi thế của Angular2

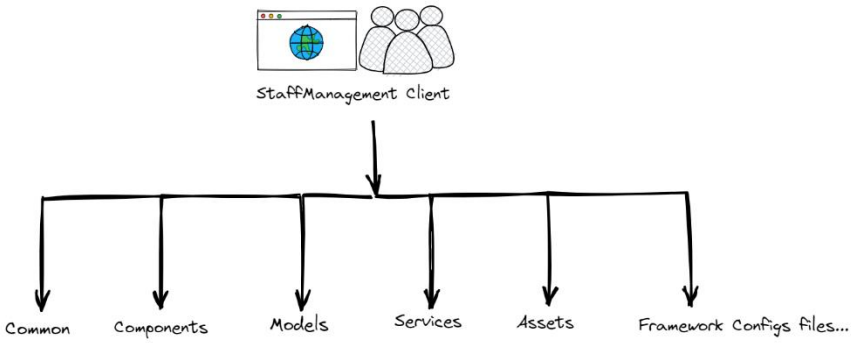
Và cuối cùng, một trong những lợi thế lớn nhất của Angular2 chính là việc nó được hỗ trợ bởi Google. Google cung cấp khả năng hỗ trợ dài hạn (Long-Term Support) cho Angular, cũng là một phần trong kế hoạch của Google trong việc hỗ trợ lập trình viên gắn bó với framework này và những sản phẩm khác nữa trong hệ sinh thái của Angular. Do đó Angular2 cũng có một lượng người dùng rất đông đảo, tạo nên một cộng đồng lớn mạnh, giúp các nhà lập trình viên có thể cùng nhau trao đổi kiến thức, phát hiện lỗi, phát triển công cụ này tương xứng với nhu cầu ngày một cao của thời đại.

Chính nhờ những ưu điểm đã được nêu phía trên, nhiều tập đoàn lớn đã lựa chọn Angular2 làm framework phát triển ứng dụng Web cho bên mình như:

- Microsoft Home Office
- Deutsche Bank Developer Portal
- Deutsche Bank Developer Portal
- IBM
- PayPal
- Google Marketing Platform

2.2.2. Chi tiết cấu tạo hệ thống Client

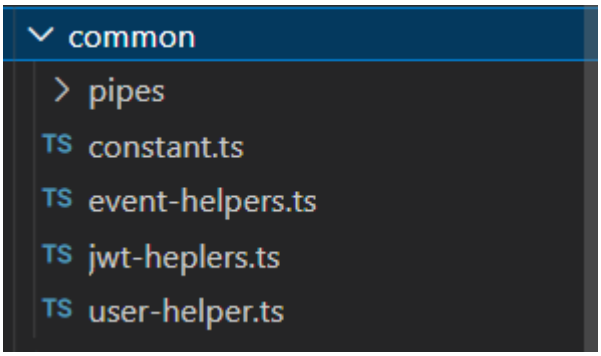
Hệ thống Client, hay Web App của ứng dụng được phát triển bằng Angular2. Dựa vào những ưu điểm đã được nêu phía trên, dễ dàng thấy nhất trong cách thiết kế mã nguồn Angular2, tổ chức các thành phần trong mã nguồn rất khoa học.



Hình 2-2-23 Cấu trúc thành phần trong hệ thống Client

Trong ứng dụng này, ta sẽ chia mã nguồn ra làm 4 phần chính như trên hình. Cách chia thành phần trong mã nguồn Angular2 sẽ có một vài nét tương đồng như trong cách tổ chức thành phần của Server .Net.

Common là thành phần chịu trách nhiệm chứa các file thông dụng, sử dụng nhiều và mang tính hỗ trợ các thành phần khác trong mã nguồn.



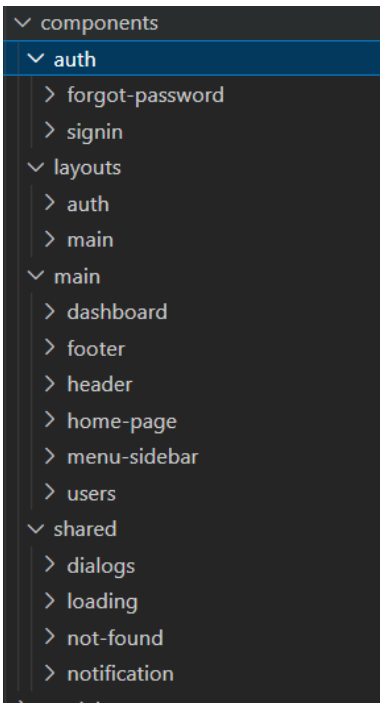
Hình 2-2-24 Cấu trúc thành phần Common của Client

Common bao gồm file chứa các hằng số của các thiết lập, các helper chứa các hàm hỗ trợ tính toán, kiểm tra, xử lý những object mà phải sử dụng nhiều trong các thành phần khác. Ngoài ra Common chứa các định nghĩa về pipeline, là một tính năng của Angular2 hỗ trợ nhằm xử lý các văn bản text, các xâu ký tự trong phần HTML để hiển thị đúng với định dạng mong muốn của lập trình viên, chỉ bằng cách gọi lại pipeline đó sau mỗi phần ký tự cần định dạng.

Component là một thành phần rất quan trọng, chứa các định nghĩa về component của Angular2. Như ta đã được biết Angular2 chứa các components, cấu tạo nên giao diện

của người dùng trong Web App. So với cách truyền thống sử dụng HTML và CSS để style cho giao diện, và viết script trong file Javascript, người dùng cũng sẽ được sử dụng HTML và SCSS (một phiên bản cải tiến của CSS, với những tính năng giúp tái sử dụng các biến và liên kết các file SCSS lại với nhau) để cấu tạo nên một component. Các component này có thể xem như là các mảnh ghép, ghép lại với nhau tạo giao diện người dùng. Những component này đều có thể tái sử dụng lại trong các trang giao diện khác nhau trong cùng một ứng dụng, do đó giảm bớt được độ dài mã nguồn, cũng như tạo nên sự linh hoạt trong cách thiết kế giao diện lên rất nhiều.

Điều tạo nên sự đặc biệt của các component chính là việc Angular2 sử dụng Typescript để thiết kế cách hoạt động của từng component, và cho phép việc phát triển



Hình 2-2-25 Cấu trúc thành phần Component của hệ thống Client

các file HTML động dựa trên những biến số trong file script đó. Ngoài ra, nó cũng cho phép việc các component có thể giao tiếp với nhau, đưa ra các phản ứng dựa trên một event thao tác của người dùng trong, khiến cho trang web hoạt động rất linh hoạt.

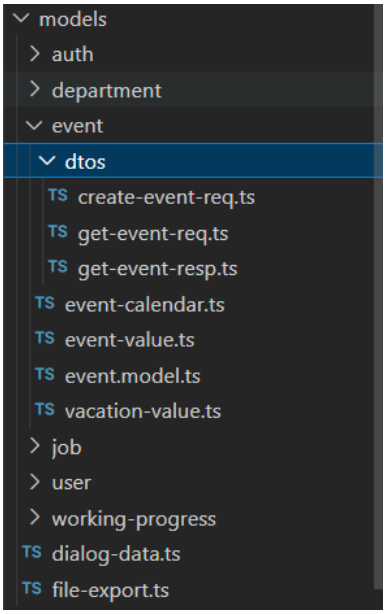
Để đảm bảo cho việc tái sử dụng các component một cách tối ưu nhất, ta phân các component ra thành các thành phần như trên.

- **Layouts** bao gồm các định nghĩa về vị trí các thành phần chính (như header, content, footer) sẽ hiển thị chung trong tất cả các trang áp dụng layout đó. Hiện tại ta có 2 layout là auth – Layout của trang web trước khi đăng nhập, và main –

Layout của trang web sau khi đăng nhập, sẽ có thể làm việc. Các thành phần chính sẽ được hiển thị xuyên suốt các trang web của một layout đó.

- **Auth** bao gồm các component thuộc layout *auth*. Các component này đảm nhiệm việc cấu tạo nên các trang giao diện của người dùng trước khi đăng nhập thành công, bao gồm trang đăng nhập và quên mật khẩu.
- **Main** bao gồm các component thuộc layout *main*. Các component này đảm nhiệm việc cấu tạo nên các trang giao diện của người dùng ở phiên làm việc sau khi đã đăng nhập thành công.
- **Shared** chứa các component cấu tạo nên các thành phần giao diện dùng chung trong các layout như *loading*, *notification*, trang *not found 404*, các *dialog* và các *pop-up*,…

Model chứa các định nghĩa về các thành phần của các object Model, các DTO của Model đó được service sử dụng.

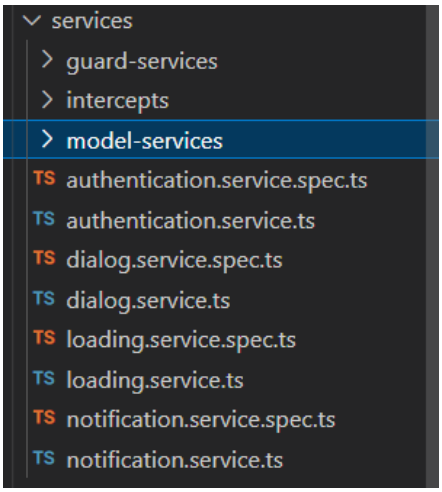


Hình 2-2-26 Cấu trúc thành phần Model trong hệ thống Client

Tương tự như ở API, các Model cũng có các thành phần ứng với những trường trong Database của bảng nó đại diện. Các DTO đảm nhiệm vai trò là các object đại diện cho request và response qua lại với API ứng với từng phương thức HTTP nó.

Service trong Angular2 cũng giống như trong API, là thành phần chứa các phương thức nhằm giải quyết các yêu cầu của phía Client yêu cầu. Do bản thân đây là Web App, là nơi mà những người dùng sử dụng để giao tiếp với máy chủ, nên sẽ có rất nhiều yêu cầu được sinh ra, cho nên lượng Service được sinh ra cũng nhiều hơn so với ở API.

Tại đây, các Service sẽ được phân chia để đảm nhiệm cho từng phần nhiệm vụ mà nó phụ trách.



Hình 2-2-27 Cấu trúc thành phần Service của hệ thống Client

Guard Service đảm nhiệm việc phân quyền cho người dùng, tạo các guard để đảm bảo chỉ người dùng có đủ quyền hạn mới được truy cập vào một trang web nhất định trong ứng dụng, và sẽ redirect những yêu cầu truy cập trang web không hợp lệ.

Intercept Service hoạt động gần giống với một Middleware, khi nó trực tiếp can thiệp vào giữa những request và response HTTP của Client. Ví dụ nó sẽ tự động điền Authentication token vào những request trước khi được gửi cho API, khi người dùng đã đăng nhập và được cấp token lưu trong Context định danh hiện tại. Hay nó cũng xử lý các response chứa lỗi từ API, từ đó hiển thị lên trang web người dùng dưới dạng các pop-up để người dùng hoặc lập trình viên phát hiện lỗi và xử lý. Hoặc đơn giản intercept cũng sẽ có thể phát hiện một request quan trọng của người dùng chưa được phản hồi, thì nó sẽ tiến hành hiển thị *loading* chắn ngang trang web, để nhắc nhở người dùng chờ đợi request được phản hồi.

Model Service thực hiện các công việc liên quan tới các Model object, bao gồm tạo những yêu cầu Tạo, Đọc, Cập nhật, Xoá các object của Model đó thông qua phương thức HTTP tới API, hay xuất kết quả ra dạng file, thay đổi trạng thái đối tượng tại phiên làm việc, …

Các Service còn lại sẽ thực hiện các công việc phụ, bao gồm những công việc liên quan tới quản lý các component trong xây dựng giao diện cho người dùng như: *loading*, hiển thị thông báo, hiển thị *pop up*, hiển thị *dialog*, … hoặc các công việc liên quan tới xử lý việc giao tiếp dữ liệu giữa các component với nhau.

Cuối cùng là các thành phần hệ thống của Angular2. Các thành phần này bao gồm:

- File đảm nhiệm việc thiết lập các đường dẫn tới các trang web trong ứng dụng, cũng như phân quyền truy cập cho từng trang web đó.
- File khai báo các module package được các Component sử dụng trong quá trình xây dựng giao diện.
- File khai báo các package mà Angular2 sử dụng, được tạo ra để Angular2 nhớ và lấy những package đó từ hệ thống npm của NodeJS để chạy chương trình.
- Các *assets* ứng dụng sử dụng.
- Các file chứa các thiết lập của ứng dụng như: môi trường, đường dẫn, …
- …

Chương 3. Phần mềm chấm công

Phần mềm chấm công được viết bởi ngôn ngữ Python phiên bản 3.x, sử dụng 2 công nghệ là MTCNN trong nhận diện khuôn mặt, và FaceNet trong định danh khuôn mặt. Ta sẽ tìm hiểu về 2 công nghệ này ở phần tiếp theo.

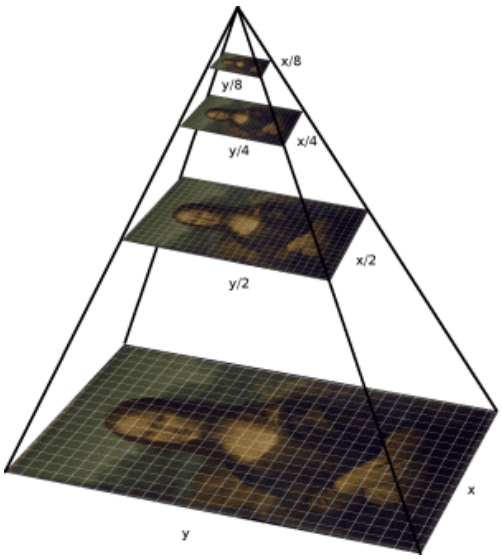
3.1. MTCNN

MTCNN[12] là một mạng neuron nhân tạo, có khả năng nhận diện khuôn mặt cùng với các điểm mốc trên khuôn mặt đó. MTCNN được công bố vào năm 2016 bởi Zhang và những cộng sự.

Mạng MTCNN nhận diện khuôn mặt dựa trên việc nó có ba mạng tích chập riêng biệt, hay còn gọi là ba stage bao gồm: P-Net, R-Net và O-Net[13].

Stage 1: P-Net

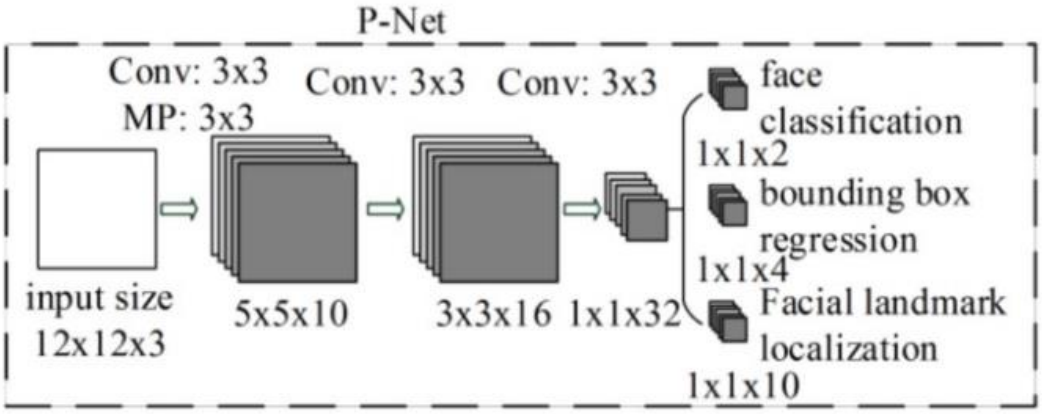
Khi nhận diện khuôn mặt trong một bức ảnh, hay một frame của video, ta thường sẽ có nhiều khuôn mặt trong đó với nhiều kích thước khác nhau. Do đó ta cần một phương thức có thể nhận diện hết tất cả khuôn mặt đó. MTCNN cho ta một giải pháp đó là Resize bức ảnh/frame đó thành hàng loạt các bản copy có kích thước khác nhau, rồi sắp xếp chúng từ lớn đến bé tạo nên một **Image Pyramid.**



Hình 3-1 Image Pyramid trong mạng P-Net

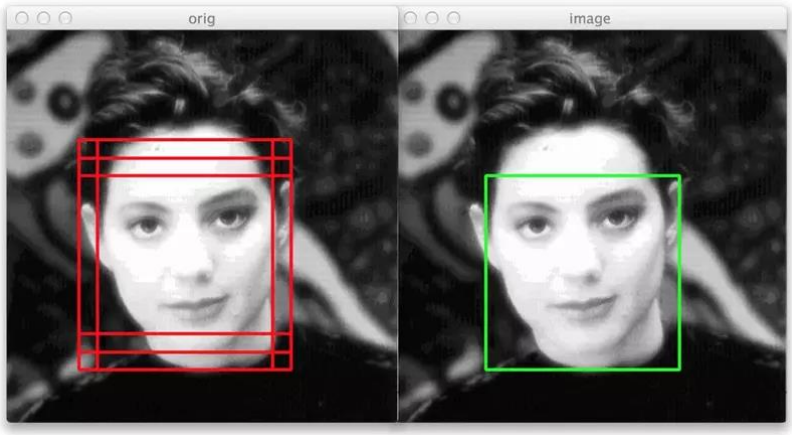
Với mỗi phiên bản đã được resize đó, ta sử dụng một kernel 12x12 và stride = 2 để quét qua tất cả các phần của ảnh và phát hiện các khuôn mặt có trong đó. Do ta có rất nhiều các bản copy của ảnh gốc với nhiều kích thước khác nhau, nên mạng có thể dễ dàng nhận biết được những khuôn mặt có kích thước khác nhau chỉ với 1 kernel kích

thước cố định, do trong các ảnh đã resize sẽ tạo nên các khuôn mặt có kích thước gần nhau và gần với kích thước kernel. Những kernel như trên sẽ được cắt ra và truyền vào mạng P-Net. Kết quả của mạng cho ra một loạt các bounding boxes nằm trong mỗi kernel, mỗi bounding boxes sẽ chứa tọa độ 4 góc để xác định vị trí trong kernel chứa nó (đã được normalize về khoảng từ (0,1)) và điểm confident (Điểm tự tin) tương ứng.



Hình 3-2 Kiến trúc mạng P-Net

Sau đấy ta sẽ cần phải loại bỏ bớt các bounding boxes trên các ảnh và các kernel. Ta sẽ sử dụng 2 phương pháp là lập mức Threshold confident - nhằm xóa đi các box có mức confident thấp và sử dụng NMS (Non-Maximum Suppression) để xóa các box có tỷ lệ trùng nhau (Intersection Over Union) vượt qua 1 mức threshold tự đặt nào đó.

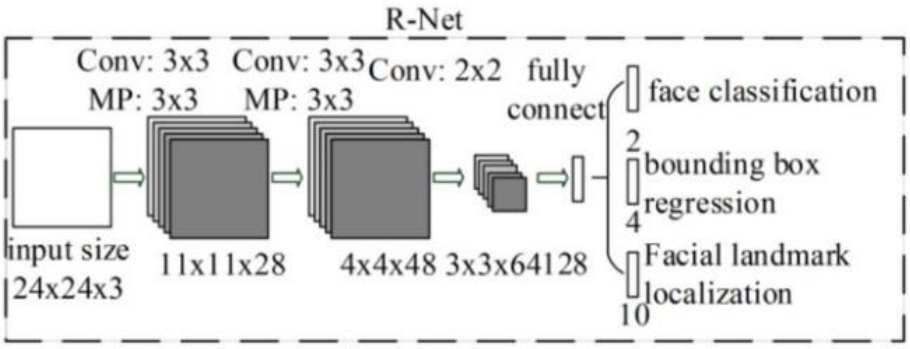


Hình 3-3 Ảnh được xoá bớt các box không hợp lý

Sau khi đã xóa bớt các box không hợp lý, ta sẽ chuyển các tọa độ của các box về với tọa độ gốc của bức ảnh thật. Do tọa độ của box đã được normalize về khoảng (0,1) tương ứng như kernel, cho nên công việc lúc này chỉ là tính toán độ dài và rộng của kernel dựa theo ảnh gốc, sau đó nhân tọa độ đã được normalize của box với kích thước của của kernel và cộng với tọa độ của các góc kernel tương ứng. Kết quả của quá trình

trên sẽ là những tọa độ của box tương ứng ở trên ảnh kích thước ban đầu. Cuối cùng, ta sẽ resize lại các box về dạng hình vuông, lấy tọa độ mới của các box và feed vào mạng tiếp theo, mạng R.

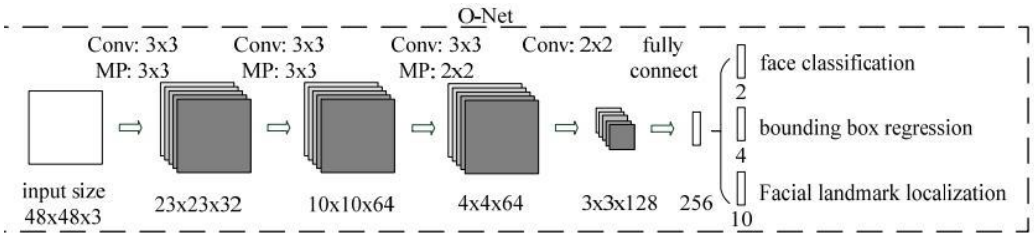
Stage 2: R-Net



Hình 3-4 Kiến trúc mạng R-Net

Mạng R (Refine Network) thực hiện các bước như mạng P. Tuy nhiên, mạng còn sử dụng một phương pháp tên là padding, nhằm thực hiện việc chèn thêm các zero-pixels vào các phần thiếu của bounding box nếu bounding box bị vượt quá biên của ảnh. Tất cả các bounding box lúc này sẽ được resize về kích thước 24x24, được coi như 1 kernel và feed vào mạng R. Kết quả sau cũng là những tọa độ mới của các box còn lại và được đưa vào mạng tiếp theo, mạng O.

Stage 3: O-Net



Hình 3-5 Kiến trúc mạng O-Net

Mạng O (Output Network) cũng thực hiện tương tự như việc trong mạng R, thay đổi kích thước thành 48x48. Tuy nhiên, kết quả đầu ra của mạng lúc này không còn chỉ là các tọa độ của các box nữa, mà trả về 3 giá trị bao gồm: 4 tọa độ của bounding box (out[0]), tọa độ 5 điểm landmark trên mặt, bao gồm 2 mắt, 1 mũi, 2 bên cánh môi

(out[1]) và điểm confident của mỗi box (out[2]). Tất cả sẽ được lưu vào thành 1 dictionary với 3 keys kể trên.

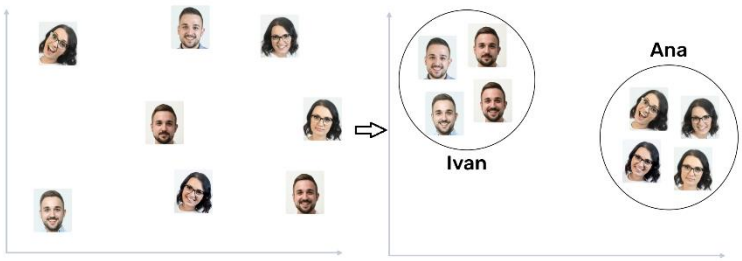
Trải qua 3 Stage trên, ta đã có thể lấy các khuôn mặt từ bức hình. Việc tiếp theo chính là định danh các khuôn mặt đó.

3.2. FaceNet

FaceNet[14] là một mạng neuron nhân tạo sâu được sử dụng để trích xuất các đặc điểm từ hình ảnh khuôn mặt của một người. FaceNet được công bố vào năm 2015 bởi các nhà nghiên cứu của Google là Schroff và các cộng sự.

Với FaceNet, ta cần đánh giá khuôn mặt đã được phát hiện là của ai trong những khuôn mặt được hệ thống ghi nhận trước đó. Trong FaceNet, ta sẽ có vài khái niệm như sau:

- **Embedding Vector**: Là một vector với dimension cố định (thường có chiều nhỏ hơn các Feature Vector bình thường), đã được học trong quá trình train và đại diện cho một tập các feature có trách nhiệm trong việc phân loại các đối tượng trong chiều không gian đã được biến đổi. Embedding rất hữu dụng trong việc tìm các Nearest Neighbor trong 1 Cluster cho sẵn, dựa theo khoảng cách-mối quan hệ giữa các embedding với nhau.
- **Inception V1**: Một cấu trúc mạng CNN được giới thiệu vào năm 2014 của Google, với đặc trưng là các khối Inception. Khối này cho phép mạng được học theo cấu trúc song song, nghĩa là với 1 đầu vào có thể được đưa vào nhiều các lớp Convolution khác nhau để đưa ra các kết quả khác nhau, sau đó sẽ được Concatenate vào thành 1 output. Việc học song song này giúp mạng có thể học được nhiều chi tiết hơn, lấy được nhiều feature hơn so với mạng CNN truyền thống. Ngoài ra, mạng cũng áp dụng các khối Convolution 1x1 nhằm giảm kích thước của mạng, khiến việc train trở nên nhanh hơn.



Hình 3-6 Ví dụ về Triplet Loss trong việc phân loại các khuôn mặt

- Triplet Loss: Thay vì sử dụng các hàm loss truyền thống, khi mà ta chỉ so sánh giá trị đầu ra của mạng với ground truth thực tế của dữ liệu, Triplet Loss đưa ra một công thức mới bao gồm 3 giá trị đầu vào gồm *anchor*: ảnh đầu ra của mạng, positive, ảnh cùng là 1 người với anchor và negative: ảnh không cùng là 1 người với anchor.

Đối với việc sử dụng FaceNet, ta có quá trình training data như sau:

- Sử dụng một tập Dataset với rất nhiều các cá thể người khác nhau, mỗi cá thể có một số lượng ảnh nhất định.
- Xây dựng một mạng DNN dùng để làm Feature Extractor cho Dataset trên, kết quả là 1 embedding 128-Dimensions. Theo như paper của FaceNet, có 2 đại diện mạng là Zeiler&Fergus và InceptionV1.
- Huấn luyện mạng DNN để kết quả embedding có khả năng nhận diện tốt, tối ưu lại các parameters trong mạng màng bằng Triplet Loss.
- Hàm Triplet Loss sẽ sử dụng phương pháp Triplet Selection, lựa chọn các embeddings sao cho việc học diễn ra tốt nhất.

3.3. Chi tiết cấu tạo phần mềm chấm công bằng nhận diện khuôn mặt

MTCNN và FaceNet là 2 mạng rất nổi tiếng trong việc xử lý bài toán Face Recognition nói chung. Và việc kết hợp giữa chúng, khi đầu vào là ảnh/video với rất nhiều người và trong hoàn cảnh thực tế, sẽ đưa ra được kết quả khá tốt. Khi đó, MTCNN sẽ đóng vai trò là Face Detection/Alignment, giúp cắt các khuôn mặt ra khỏi khung hình dưới dạng các tọa độ bounding boxes và chỉnh sửa / resize về đúng shape đầu vào của mạng FaceNet. Còn FaceNet sẽ đóng vai trò là mạng Feature Extractor + Classifier cho từng bounding boxes, đưa ra embedding và tiền hành phân biệt và nhận dạng các khuôn mặt.

Tiếp theo ta sẽ tìm hiểu việc áp dụng 2 công nghệ này vào phần mềm như thế nào.

Phần mềm sẽ có 3 module chính là : Capture Face, Update Data và Recognize Face.

Trong 2 module Capture Face và Recognize Face đều sử dụng MTCNN là công cụ Face Detection/Alignment.

MTCNN có sẵn trong package *facenet_pytorch* của python. Ta import ta và khai báo như sau:

```
device =  torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
```

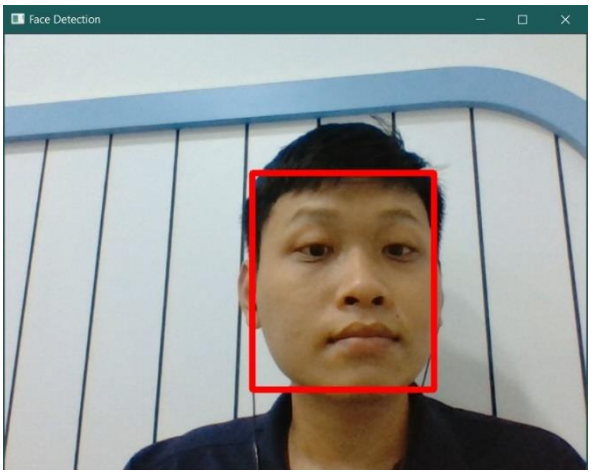
```
mtcnn = MTCNN(margin = 20, keep_all=False, post_process=False, device =
device)
```

MTCNN tự lựa chọn device là CPU hoặc GPU, đảm bảo khả năng nhận diện nhanh nhất.

Sử dụng một vòng lặp, lặp qua các frame và tiến hành nhận diện khuôn mặt bằng MTCNN như sau:

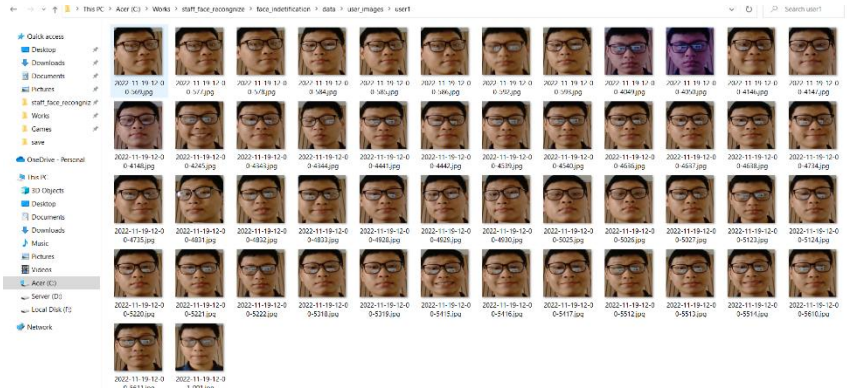
```
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH,640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT,480)
while cap.isOpened():
    isSuccess, frame = cap.read()
    if isSuccess:
        boxes, _ = mtcnn.detect(frame)
        if boxes is not None:
            for box in boxes:
                bbox = list(map(int,box.tolist()))
                frame =
cv2.rectangle(frame,(bbox[0],bbox[1]),(bbox[2],bbox[3]),(0,0,255),6)
```

Kết quả sẽ được như phía dưới, đối với việc sử dụng CPU thông dụng, frame rate sẽ đạt từ 15-20 fps.



Hình 3-7 MTCNN nhận diện được một khuôn mặt từ các frame của camera

Module Capture Face sẽ dựa vào kết quả MTCNN thu được khi nhận diện, từ đó lưu



Hình 3-8 Hình ảnh chụp được lưu lại trong folder ứng với UserName của nhân viên đó

các hình khuôn mặt đã được resize và align lại để Module Update Data tiến hành training.

Tại Module Update Data, việc normalize rất quan trọng trong inference: Khi dữ liệu được chuẩn hóa về 1 khoảng cố định, thuật toán tối ưu Gradient Descent sẽ đưa ra được kết quả converge nhanh chóng hơn và tránh bị các vấn đề liên quan tới Vanishing/Exploding Gradient. Hàm sau là một hàm normalize đơn giản, giúp đưa pixel ảnh về trong khoảng [-1,1] mà vẫn giữ được distribution gốc của nó:

```
def trans(img):
    transform = transforms.Compose([
            transforms.ToTensor(),
            fixed_image_standardization
        ])
    return transform(img)
```

Trong đó, hàm `fixed_image_standardization` được viết như sau:

```
def fixed_image_standardization(image_tensor):
    processed_tensor = (image_tensor - 127.5) / 128.0
    return processed_tensor
```

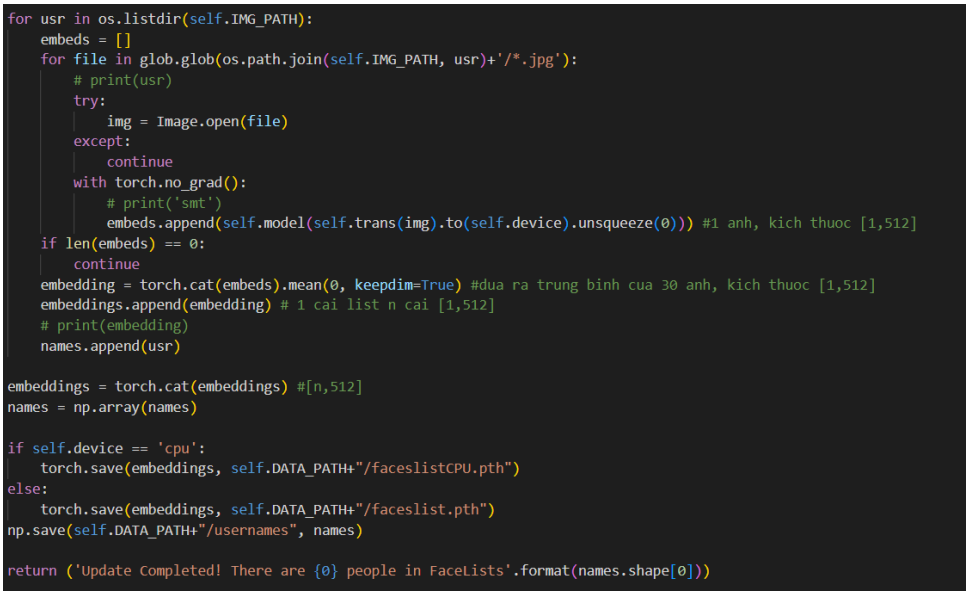
Sau khi Normalize các hình ảnh, ta sẽ cần khai báo model sẽ sử dụng. FaceNet sử dụng cấu trúc mạng InceptionV1 và đầu ra là một Feature Vector 128 chiều. Tuy nhiên do cấu trúc mạng InceptionV1 đã cũ, tỷ lệ chính xác đã bị các mạng bây giờ vượt xa cũng như tốc độ tính toán của máy tính đã có sự cải thiện đáng kể so với năm 2015. Do đó ta sẽ sử dụng kiến trúc mạng mới là InceptionResnetV1 với đầu ra là một Feature Vector 512 chiều. Ngoài ra, mạng được pretrained trên tập dữ liệu mới là VGGFace2 và CASIA-WebFace, trong code mình sử dụng pretrained của tập thứ 2:

```
model = InceptionResnetV1(
            classify=False,
            pretrained="casia-webface"
            #pretrained="vggface2"
        ).to(device)
model.eval()
```

Tham số classify được gán bằng False, nhằm đảm bảo đầu ra sẽ là 1 Feature Vector chứ không phải kết quả đã được classify theo tập pretrained. Lệnh `model.eval()` để khai báo cho PyTorch rằng mình đang evaluation, không phải training.

Tiếp theo ta sẽ tạo lập embed từ các ảnh khuôn mặt đã được chụp. Với mỗi User ta đã chụp, ta xét toàn bộ ảnh, mỗi ảnh sẽ dùng model đã được load pretrained trên để sinh ra một embed với kích thước [1,512]. Ứng với n ảnh (phần mềm chụp 40 ảnh), ta sẽ có n embed của một User. Tuy nhiên, ta chỉ cần một embeds, đại diện cho User đó. Vì vậy, ta sẽ sử dụng hàm torch.cat() để đưa list về 1 tensor 2 chiều và sử dụng hàm torch.mean() để lấy giá trị trung bình cho toàn bộ embeds. Kết quả cuối cùng sẽ được

thêm vào list embeddings, đại diện cho tập hợp các embedding của các user, ứng với đó là 1 giá trị name của user với cùng index trên tập names. Sau cùng, ta sẽ một lần



```
for usr in os.listdir(self.IMG_PATH):
    embeds = []
    for file in glob.glob(os.path.join(self.IMG_PATH, usr)+'/*.jpg'):
        # print(usr)
        try:
            img = Image.open(file)
        except:
            continue
        with torch.no_grad():
            # print('smt')
            embeds.append(self.model(self.trans(img).to(self.device).unsqueeze(0))) #1 anh, kich thuoc [1,512]
    if len(embeds) == 0:
        continue
    embedding = torch.cat(embeds).mean(0, keepdim=True) #dua ra trung binh cua 30 anh, kich thuoc [1,512]
    embeddings.append(embedding) # 1 cai list n cai [1,512]
    # print(embedding)
    names.append(usr)

embeddings = torch.cat(embeddings) #[n,512]
names = np.array(names)

if self.device == 'cpu':
    torch.save(embeddings, self.DATA_PATH+"/faceslistCPU.pth")
else:
    torch.save(embeddings, self.DATA_PATH+"/faceslist.pth")
np.save(self.DATA_PATH+"/usernames", names)

return ('Update Completed! There are {0} people in FaceLists'.format(names.shape[0]))
```

Hình 3-9 Đoạn mã xử lý embedding và lưu lại kết quả thành các FaceList

nữa concatenate list embeddings lại dưới dạng tensor và lưu thành một file .pth, còn tập hợp tên user sẽ được lưu lại vào một folder của chương trình.

Sau khi xong công đoạn này, ta đã có một danh sách khuôn mặt dưới dạng embedding. Bước cuối cùng tại Module Face Regconize, ta sẽ phân loại khuôn mặt dựa trên danh sách các khuôn mặt đã được Module Update Data tạo ra.

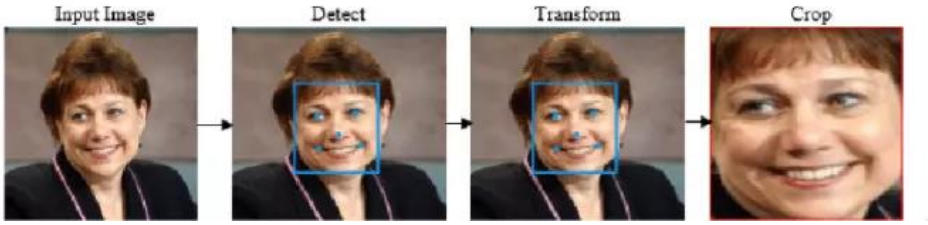
Tương tự với Module Capture Face, ta cũng sẽ duyệt các frame trên camera. Để đảm bảo frame rate ổn định, và thời gian xử lý để nhận diện cho nhân viên, ta sẽ lấy frame sau mỗi 5 frame.

Sau khi MTCNN nhận diện khuôn mặt, ta sẽ tiến hành nhận dạng khuôn mặt đó.

```
boxes, _ = self.mtcnn.detect(frame)
if boxes is not None:
    for box in boxes:
        bbox = list(map(int,box.tolist()))
        face = self.extract_face(bbox, frame)
        idx, score = self.inference(self.model, face, self.embeddings)
        if idx != -1:
            return self.names[idx]
        else:
            return "Unknown"
```

Tại mỗi frame, MTCNN sẽ nhận diện một tập các khuôn mặt, trả vào mảng bbox là mảng toạ độ các khuôn mặt đó. Ta cần lấy khuôn mặt từ các toạ độ trong mảng, nên có hàm `extract_face`.

Khuôn mặt sau khi được extract sẽ tiến hành inference, kết xuất embedding cho từng ảnh mặt từ model mạng InceptionResnetV1 và các embedding load từ việc Update Data.



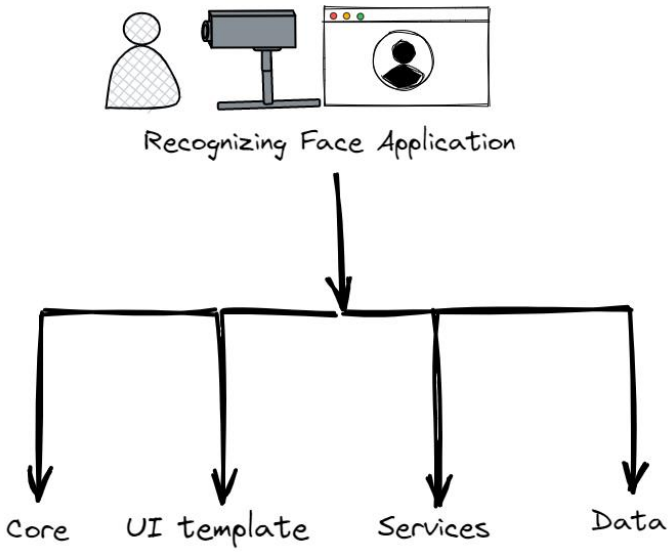
Hình 3-10 Ví dụ về việc crop một khuôn mặt từ ảnh gốc

Phần kết xuất embedding sẽ tương tự như trong phần Update Facelist. Tuy nhiên, ta không lưu lại vào FaceList mà sẽ tiến hành so sánh khoảng cách giữa embedding vừa nhận được với các embeddings khác có trong FaceList.

Kết quả cuối cùng là một dãy các score, mỗi cột là score điểm khác biệt giữa face nhận được với faces trong FaceList. Điểm khác biệt càng nhỏ, độ similarly giữa 2 face càng lớn. Vì vậy chúng ta sẽ trả về index của khuôn mặt trong FaceList có score nhỏ nhất so với face cần inference. Khi đó, giá trị index trả về sẽ là -1, tức sẽ không nhận dạng khi gặp giá trị này và trả về box Unknown.

Từ kết quả thu được ta có UserName của nhân viên đã được nhận diện, việc cuối cùng là phần mềm sẽ gửi một HTTP request về API và tạo sự kiện điểm danh cho nhân viên đó như ta đã thấy ở các ví dụ ở phần API.

Để sử dụng được các Module trên, ta sẽ sử dụng package **PyQt** để thực hiện việc xây dựng GUI cho ứng dụng. Cấu trúc của phần mềm này sẽ như sau:



Hình 3-11 Cấu trúc phần mềm nhận diện

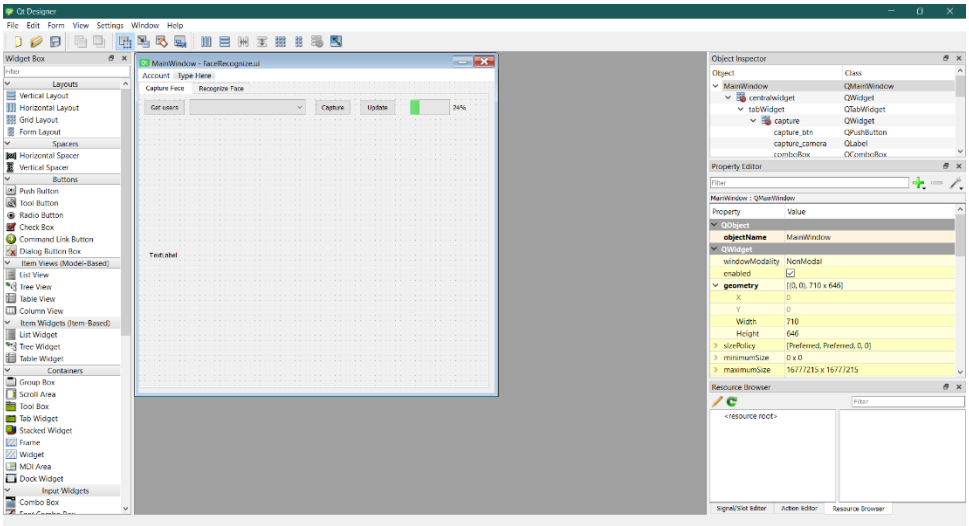
Core sẽ chịu trách nhiệm chứa các dữ liệu là các class chứa các phương thức hoạt động của các Module trong ứng dụng là: Ghi nhận khuôn mặt mới, Cập nhật dữ liệu và training, Nhận diện khuôn mặt, đã được ta phân tích hoạt động ở phần trước.

Services chứa các thành phần hoạt động của ứng dụng. Tại đây ta có 2 service là:

- **UserService**: Hoạt động tương tự UserService của các thành phần trên, chịu trách nhiệm xử lý các yêu cầu về người dùng như: đăng nhập, lấy dữ liệu về nhân viên, yêu cầu điểm danh một nhân viên,…
- **Workers**: Chịu trách nhiệm tạo các luồng cho các Module ở Core hoạt động trong nền của ứng dụng xây dựng bởi PyQt. Ta cũng có 3 Worker của 3 Module là: CaptureWorker, UpdateFaceWorker và RecognizeFaceWorker.

Data là folder chứa các dữ liệu là các ảnh chụp từ Module Ghi nhận khuôn mặt mới và các FaceList từ Module Cập nhật dữ liệu và training, hay các kết quả khuôn mặt dưới dạng embedding dựa trên dữ liệu của những nhân viên đã được đăng ký với máy. Các dữ liệu này sẽ được Module Nhận diện sử dụng để chấm công cho nhân viên.

UI Template chứa các định nghĩa về bố cục và các thành phần trong GUI như vị trí các nút, các form,… Bản thân Qt của PyQt đã hỗ trợ cho các lập trình viên công cụ để xây dựng UI bằng cách kéo thả trực quan một cách dễ dàng là QtDesign.



Hình 3-12 Công cụ xây dựng giao diện ứng dụng

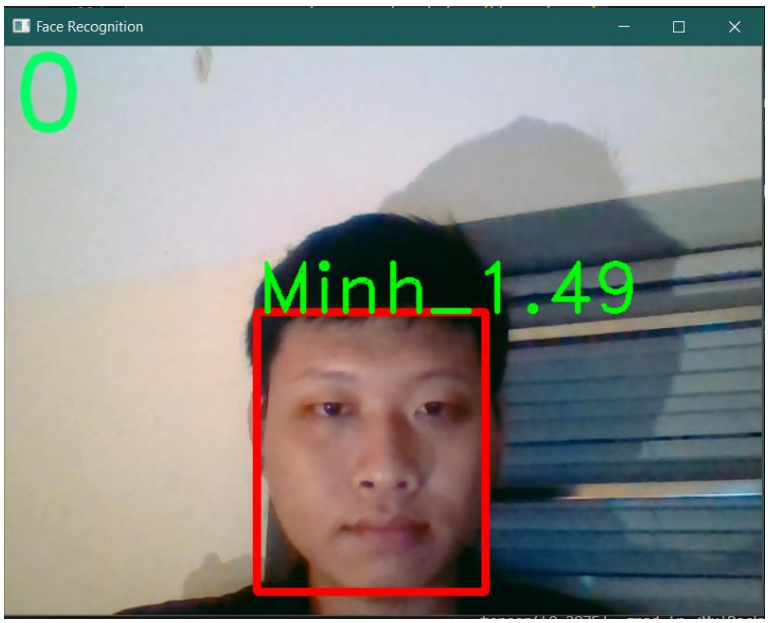
Việc cuối cùng của chúng ta là sẽ thiết lập các sự kiện nhấn nút bằng các hành động đã được định nghĩa cách xử lý trong các Service.

3.4. Thực nghiệm và độ chính xác của FaceNet

Trong các thực nghiệm về độ chính xác của FaceNet, ví dụ như thực nghiệm sử dụng DataSet là các khuôn mặt ở Youtube, độ chính xác đạt tới 95,12%, tốt hơn DeepFace với 91,4% và DeepId với 93,5% với cùng 100 frame. Đây là khi FaceNet sử dụng kiến trúc mạng InceptionV1 và Model pre-trained cũ.

Trong các thực nghiệm mới nhất[18], sử dụng 2 Model pre-trained đã nhắc phía trên là CASIA-WebFace và VGGFace2, và test với nhiều dataset khác nhau, ta đạt được độ chính xác rất ấn tượng. Với Model VGGFace2, ta đạt được độ chính xác 100% ở các dataset là YALE, JAFFE, AT & T, Essex faces95, Essex grimace; 99.375% ở dataset Essex faces94 và tệ nhất là 77.67% ở dataset faces96.

Trên thực tế tại ứng dụng của tôi, FaceNet đạt được độ chính xác rất cao. Tuy nhiên tại các điều kiện như thiếu sáng, máy quay mờ thì độ chính xác giảm đi nhiều.



Hình 3-13 Thực nghiệm trong điều kiện thiếu sáng

3.5. Các hạn chế

Mặc dù tốc độ nhận diện, xử lý rất nhanh, kết quả thu được khả quan, song phần mềm này còn một vài hạn chế như sau:

- Khả năng align khuôn mặt chưa được tốt nên khi người dùng có những góc cạnh khác nhau, embedding tương ứng cũng sẽ bị sai lệch khá nhiều.

- Phần Classifier dựa chủ yếu vào việc tính khoảng cách giữa các embeddings (khá giống thuật toán k-NN với k = 1) cho nên kết quả chưa thực sự quá tốt. Để khắc phục, chúng ta có thể xây dựng một đầu Classifier đơn giản mà mạnh mẽ hơn như SVM hoặc một mạng FCN nhỏ và áp dụng phương pháp Online Training.
- Các yếu tố như ánh sáng, chất lượng camera, góc chụp có thể khiến độ chính xác giảm đáng kể.
- Chưa thể chống gian lận nếu như nhân viên sử dụng ảnh của người khác.
- Chưa thể nhận diện trong tình huống người dùng đeo khẩu trang.

Chương 4. Kết luận

Một cách tổng quan, ứng dụng quản lý nhân viên, sử dụng nhận diện khuôn mặt trong chấm công đã cung cấp và giải quyết cho người dùng những công việc như sau:

- Thuận tiện cho quản lý các công ty có thể quản lý số lượng, tình hình đi làm của tất cả các nhân viên trong công ty, kể cả với lượng nhân viên rất lớn nhờ vào sức mạnh của hệ thống .Net.
- Giúp nhân viên có thêm một công cụ để tra cứu lịch làm việc của bản thân và công ty, cũng như tạo các sự kiện nhắc nhở bản thân trong quá trình đi làm.
- Giảm bớt thời gian, tăng độ chính xác, tin cậy cho công việc tính toán lương thưởng.
- Nhờ sử dụng nhận diện khuôn mặt trong chấm công, công ty có thêm một sự lựa chọn thuận tiện, nhanh chóng, an toàn trong mùa dịch bệnh hơn cho nhân viên trong công ty.
- V…v…

Tuy vậy vẫn còn nhiều hạn chế trong ứng dụng mà trong tương lai, tôi cần phải nghiên cứu thêm, cải tiến tốt hơn nữa như:

- Thêm nhiều tính năng nữa cho các nhà quản lý, cũng như nhân viên như: Thống kê thời gian đi làm của cả công ty, nhắc nhở hoặc khen thưởng nhân viên dựa vào ngày đi làm, tạo kênh liên lạch giữa người quản lý và nhân viên qua dịch vụ tin nhắn,…
- Khắc phục các hạn chế đã nêu ở phần mềm chấm công
- Cải tiến hệ thống tối ưu hơn nữa như sử dụng truy vấn phân trang cho các truy vấn yêu cầu nhiều dữ liệu, đánh chỉ mục cho database, tạo thêm các BackgroundService cho các tác vụ, sử dụng các công cụ logging để theo dõi sự hoạt động của hệ thống, áp dụng nhiều công nghệ hơn trong tạo và xử lý request từ Client và Phần mềm chấm công, …
- Và còn rất nhiều điều cần khắc phục và cải tiến nữa….

Do đó tôi rất mong nhận được sự đánh giá và các ý kiến quý báu của mọi người nhằm cải thiện ứng dụng này ngày một tốt hơn.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Top 5 giải pháp chấm công phổ biến hiện nay - https://ooc.vn/giai-phap-cham-cong-pho-bien-hien-nay/

Tiếng Anh

[2] Giới thiệu về ngôn ngữ C# - https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[3] Giới thiệu về .Net - https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet

[4] Giới thiệu về .Net Foundation - https://dotnetfoundation.org/

[5] Những tổ chức sử dụng .Net nổi tiếng - https://dotnet.microsoft.com/en-us/platform/customers

[6] SQL Server Tutorial - https://www.sqlservertutorial.net/

[7] Bảng benchmark khả năng xử lý các Web Framework - https://www.techempower.com/benchmarks/#section=data-r21&hw=ph&test=plaintext

[8] Kiến trúc của .Net - https://stackify.com/net-ecosystem-runtime-tools-languages/

[9] Lợi thế của .Net và .Net Core - https://www.c-sharpcorner.com/article/difference-between-net-framework-and-net-core/

[10] "Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application" - https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application

[11] Robert C. Martin, "Solid Relevance", 2020 - https://blog.cleancoder.com/uncle-bob/2020/10/18/Solid-Relevance.html

[12] MTCNN Github - https://github.com/ipazc/mtcnn

[13] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, Yu Qiao, "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks", IEEE Signal Processing Letters, Vol. 23, Issue 10, 2016.

[14] FaceNet Github - https://github.com/timesler/facenet-pytorch

[15] Luka Dulčić(2019). Face Recognition with FaceNet and MTCNN - https://arsfutura.com/magazine/face-recognition-with-facenet-and-mtcnn/

[16] Florian Schroff, Dmitry Kalenichenko, James Philbin. “FaceNet: A Unified Embedding for Face Recognition and Clustering”, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) , pp. 3 – 6.

[17] How Does A Face Detection Program Work? (Using Neural Networks) - https://towardsdatascience.com/how-does-a-face-detection-program-work-using-neural-networks-17896df8e6ff

[18] Ivan William, De Rosal Ignatius Moses Setiadi, Eko Hari Rachmawanto, Heru Agus Santoso, Christy Atika Sari, “Face Recognition using FaceNet (Survey, Performance Test, and Comparison)”, 2019 Fourth International Conference on Informatics and Computing (ICIC).