UBND TỈNH BÌNH DƯƠNG

**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**BÁO CÁO CHUYÊN ĐỀ  
THỰC TẬP TỐT NGHIỆP**

Nơi thực tập

**ISC QUANG TRUNG**

**Đề tài: XÂY DỰNG PHẦN MỀM QUẢN LÝ HỌC VIÊN**

NGƯỜI HƯỚNG DẪN TẠI ĐƠN VỊ: **HÀ THANH LIÊM**

GIẢNG VIÊN HƯỚNG DẪN: **VÕ QUỐC LƯƠNG**

SINH VIÊN THỰC HIỆN:

**NGUYỄN NGỌC QUÝ** MSSV: **1624801040046**

NIÊN KHÓA: **2016-2020**

**06/2019**

UBND TỈNH BÌNH DƯƠNG

**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**BÁO CÁO CHUYÊN ĐỀ  
THỰC TẬP TỐT NGHIỆP**

Nơi thực tập

**ISC QUANG TRUNG**

**Đề tài: XÂY DỰNG PHẦN MỀM QUẢN LÝ HỌC VIÊN**

NGƯỜI HƯỚNG DẪN TẠI ĐƠN VỊ: **HÀ THANH LIÊM**

GIẢNG VIÊN HƯỚNG DẪN: **VÕ QUỐC LƯƠNG**

SINH VIÊN THỰC HIỆN:

**NGUYỄN NGỌC QUÝ** MSSV: **1624801040046**

NIÊN KHÓA: **2016-2020**

**06/2019**

**LỜI CẢM ƠN**

Trước tiên chúng em xin chân thành cám ơn thầy Hà Thanh Liêm đã tạo điều kiện tiếp nhận các em thực tập tại doanh nghiệp và tận tình hướng dẫn trong suốt thời gian thực tập tại đây.

Tiếp theo, chúng em xin cám ơn các anh chị trong doanh nghiệp nơi thực tập đã giúp đỡ chúng em trong suốt 8 tuần thực tập. Hỗ trợ tối đa để chúng em hoàn thành báo cáo thực tập này.

Chúng em cũng xin cám ơn quý thầy cô ở khoa Kỹ thuật – Công nghệ, trường Đại học Thủ Dầu Một đã truyền thụ những kiến thức quý báu cho chúng em trong suốt bốn năm học tập tại trường. Những kiến thức nền tảng làm hành trang sau này cho chúng em ra trường.

Chúng em cũng không quên gửi lời cảm ơn tới thầy Võ Quốc Lương đã tận tình, chu đáo hướng dẫn thực hiện báo cáo này.

Mặc dù đã có nhiều cố gắng để thực hiện đề tài một cách hoàn chỉnh nhất. Song do buổi đầu mới làm quen với doanh nghiệp, tiếp cận với thực tế cũng như hạn chế về kiến thức và kinh nghiệm nên không thể tránh khỏi những thiếu sót nhất định mà bản thân chưa thấy được. Chúng em rất mong được sự góp ý của quý anh chị tại doanh nghiệp và thầy Hà Thanh Liêm. Chúng em cũng mong nhận được những góp ý chân thành nhất của quý thầy cô ở Khoa để đề tài được hoàn thiện hơn.

Chúng em xin chân thành cám ơn.

*Sinh viên thực tập*

**Nguyễn Ngọc Quý**

**MỤC LỤC**

[LỜI MỞ ĐẦU 1](#_Toc27762159)

[CHƯƠNG I. GIỚI THIỆU VỀ CƠ SỞ THỰC TẬP 2](#_Toc27762160)

[I.1. Giới thiệu chung 2](#_Toc27762161)

[I.2. Chức năng và nhiệm vụ 4](#_Toc27762162)

[CHƯƠNG II. CƠ SỞ LÍ THUYẾT 5](#_Toc27762163)

[II.1. Giới thiệu về AngularJS 5](#_Toc27762164)

[II.2. Tìm hiểu Githud 8](#_Toc27762165)

[*II.2.1. Githud là gì?* 8](#_Toc27762166)

[*II.2.2. Git là gì ?* 8](#_Toc27762167)

[*II.2.3. Hud là gì?* 9](#_Toc27762168)

[*II.2.4. Repository là gì?* 9](#_Toc27762169)

[*II.2.5. Branch là gì?* 9](#_Toc27762170)

[*II.2.6. Pull Request là gì?* 9](#_Toc27762171)

[*II.2.7. Fork một Repository là gì?* 9](#_Toc27762172)

[*II.2.8. Một số lệnh cơ bản thường sử dụng trong Git* 9](#_Toc27762173)

[*II.2.9. Hướng dẫn cài đặt Git trên windows* 11](#_Toc27762174)

[II.3. Tìm hiểu Spring Boot 18](#_Toc27762175)

[II.4. Tìm hiểu RESTfull API 20](#_Toc27762176)

[*II.4.1. RESTful API là gì?* 20](#_Toc27762177)

[*II.4.2. RESTful hoạt động như thế nào?* 21](#_Toc27762178)

[*II.4.3. Status code* 22](#_Toc27762179)

[II.6. Tìm hiểu về JPA 22](#_Toc27762180)

[*II.6.1. JPA là gì?* 22](#_Toc27762181)

[*II.6.2. Một số khái niệm trong JPA* 23](#_Toc27762182)

[II.7. Tìm hiểu Hibernate 23](#_Toc27762183)

[*II.7.1. Hibernate là gì?* 23](#_Toc27762184)

[*II.7.2. Lợi ích của Hibernate Framework* 25](#_Toc27762185)

[*II.7.3. Hạn chế của Hibernate Framework* 25](#_Toc27762186)

[II.8. Tìm hiểu Agile và Scrum trong phát triển phần mềm 26](#_Toc27762187)

[*II.8.1. Tìm hiểu Agile* 26](#_Toc27762188)

[*II.8.2. Tìm hiểu Scrum* 27](#_Toc27762189)

[CHƯƠNG III. PHẦN THỰC HÀNH 28](#_Toc27762190)

[III.1. Phân tích hệ thống 28](#_Toc27762191)

[III.2. Thiết kế cơ sở dữ liệu 28](#_Toc27762192)

[III.3. Khởi tạo server Spring Boot 36](#_Toc27762193)

[*III.3.1. Công cụ phát triển* 36](#_Toc27762194)

[*III.3.2. Tạo môi trường và project* 37](#_Toc27762195)

[*III.3.3. Sử dụng POSTMAN để Get API kiểm tra dữ liệu trả về* 55](#_Toc27762196)

[*III.3.4. Các Uri thao tác với dữ liệu* 60](#_Toc27762197)

[III.4. Giao diện hệ thống reponsive sử dựng Bootstrap, AngularJS và jQuery 63](#_Toc27762198)

[*III.4.1. Giao diện quản lý học viên* 63](#_Toc27762199)

[*III.4.2. Giao diện quản lý chương trình học* 66](#_Toc27762200)

[*III.4.3. Giao diện quản lý môn học* 67](#_Toc27762201)

[*III.4.4. Giao diện quản lý khoá học* 68](#_Toc27762202)

[*III.4.5. Giao diện quản lý trường học* 68](#_Toc27762203)

[*III.4.6. Giao diện quản lý chuyên ngành* 68](#_Toc27762204)

[*III.4.7. Giao diện quản lý tính năng* 69](#_Toc27762205)

[*III.4.8. Giao diện đăng nhập admin* 69](#_Toc27762206)

[KẾT LUẬN 70](#_Toc27762207)

[1. Kết luận 70](#_Toc27762208)

[2. Kiến nghị 70](#_Toc27762209)

[TÀI LIỆU THAM KHẢO 72](#_Toc27762210)

**DANH MỤC CÁC HÌNH**

[Hình 1.1. Môi trường thực tập tại ISC – Quang Trung 3](#_Toc27762220)

[Hình 1.2. Thế mạnh của ISC – Quang Trung 4](#_Toc27762221)

[Hình 1.3. Chuyên ngành đào tạo tại ISC – Quang Trung 4](#_Toc27762222)

[Hình 2.1. Mô hình cấu trúc AngularJS 7](#_Toc27762223)

[Hình 3.1. Biểu đồ UC tổng quát 28](#_Toc27762224)

[Hình 3.2. Dữ liệu khi GET Người Quản Lí từ server 63](#_Toc27762225)

[Hình 3.3. Giao diện quản lý học viên 63](#_Toc27762226)

[Hình 3.4. Giao diện trang thêm học viên 64](#_Toc27762227)

[Hình 3.5. Giao diện trang thông tin học viên 65](#_Toc27762228)

[Hình 3.6. Giao diện xoá học viên 66](#_Toc27762229)

[Hình 3.7. Giao diện quản lý chương trình học 66](#_Toc27762230)

[Hình 3.8. Giao diện thêm môn học vào chuyên ngành tương ứng 67](#_Toc27762231)

[Hình 3.9. Giao diện xoá môn học ra khỏi chuyên ngành tương ứng 67](#_Toc27762232)

[Hình 3.10. Giao diện quản lý môn học 67](#_Toc27762233)

[Hình 3.11. Giao diện quản lý khoá học 68](#_Toc27762234)

[Hình 3.12. Giao diện quản lý trường học 68](#_Toc27762235)

[Hình 3.13. Giao diện quản lý chuyên ngành 68](#_Toc27762236)

[Hình 3.14. Giao diện quản lý tính năng 69](#_Toc27762237)

[Hình 3.15. Giao diện đăng nhập admin 69](#_Toc27762238)

**DANH MỤC CÁC BẢNG**

[Bảng 2.1. Bảng GIANGVIEN: Lưu trữ thông tin giảng viên 28](#_Toc19662190)

[Bảng 2.2. Bảng CHUYENNGANH: Lưu trữ thông tin chuyên ngành 29](#_Toc19662191)

[Bảng 2.3. Bảng DANHMUCHOCVIEN: Lưu trữ thông tin về loại học viên 29](#_Toc19662192)

[Bảng 2.4. Bảng DAOTAO: Lưu trữ thông tin về chương trình học 29](#_Toc19662193)

[Bảng 2.5. Bảng DIEMMONHOC: Lưu trữ thông tin điểm của học viên 29](#_Toc19662194)

[Bảng 2.6. Bảng DOANHNGHIEP: Lưu trữ thông tin về doanh nghiệp liên kết với trung tâm 30](#_Toc19662195)

[Bảng 2.7. Bảng GOPY: Lưu trữ thông tin góp ý của học viên 30](#_Toc19662196)

[Bảng 2.8. Bảng HOCVIEN: Lưu trữ thông tin học viên 30](#_Toc19662197)

[Bảng 2.9. Bảng HOSO: Lưu trữ thông tin hồ sơ của học viên 31](#_Toc19662198)

[Bảng 2.10. Bảng KHOAHOC: Lưu trữ thông tin khoá học 32](#_Toc19662199)

[Bảng 2.11. Bảng LOAITAIKHOAN: Lưu trữ thông tin loại tài khoản 32](#_Toc19662200)

[Bảng 2.12. Bảng MONHOC: Lưu trữ thông tin môn học 32](#_Toc19662201)

[Bảng 2.13. Bảng NGUOIHUONGDAN: Lưu trữ thông tin danh sách giảng viên hướng dẫn tại trường của học viên 33](#_Toc19662202)

[Bảng 2.14. Bảng NGUOIQUANLY: Lưu trữ thông tin quản trị viên hệ thống 33](#_Toc19662203)

[Bảng 2.15. Bảng PHONGHOC: Lưu trữ thông tin phòng học 34](#_Toc19662204)

[Bảng 2.16. Bảng TINHNANG: Lưu trữ thông tin menu chức năng của trang người dùng dựa vào loại tài khoản (học viên/ giảng viên) 34](#_Toc19662205)

[Bảng 2.17. Bảng THOIKHOABIEU: Lưu trữ thông tin thời khoá biểu cho học viên và giảng viên 34](#_Toc19662206)

[Bảng 2.18. Bảng THONGBAO: Lưu trữ thông tin nội dung thông báo 35](#_Toc19662207)

[Bảng 2.19. Bảng TRUONGHOC: Lưu trữ thông tin trường học của học viên 35](#_Toc19662208)

[Bảng 2.20. Bảng mô tả uri HocVien 60](#_Toc19662209)

[Bảng 2.21. Bảng mô tả uri GiangVien 61](#_Toc19662210)

[Bảng 2.22. Bảng mô tả uri NguoiQuanLi 62](#_Toc19662211)

# LỜI MỞ ĐẦU

Sự ra đời và phát triển của công nghệ thông tin là một trong những thành tựu vĩ đại nhất của con người. Nhịp độ phát triển công nghệ thông tin đang là một vấn đề rất được các ngành khoa học, giáo dục, kinh tế,… quan tâm. Nó hiện hữu với một tầm vóc hết sức mạnh mẽ, to lớn và ngày càng lớn mạnh thêm.

Tin học giữ vai trò đặc biệt trong hoạt động của con người, trong đời sống xã hội, trong khoa học kỹ thuật, trong sản xuất kinh doanh, trong quản lý khách hàng, trong các hoạt động mua bán hàng hoá… Ứng dụng tin học trong việc quản lý dường như không còn xa lạ với các doanh nghiệp, công ty hiện nay. Lợi ích mà các chương trình quản lý mang lại khiến người ta không thể phủ nhận tính hiệu quả của nó.

Nhằm giúp trung tâm ISC Quang Trung có thông tin đầy đủ về công tác đào tạo học viên theo từng khoá học, chúng em xin thực hiện đề tài: “**Xây dựng phần mềm quản lý học viên tại trung tâm đào tạo kỹ sư công nghệ thông tin ISC Quang Trung**” cho chuyên đề thực tập của mình.

# CHƯƠNG I. GIỚI THIỆU VỀ CƠ SỞ THỰC TẬP

## I.1. Giới thiệu chung

“IT Workforce Solution Center” hay còn gọi “ISC – QUANG TRUNG” ra đời với mong muốn mang đến những giải pháp hiệu quả và thực tiễn nhất trong vấn đề bồi dưỡng, đào tạo nguồn nhân lực CNTT chất lượng, có khả năng thực hành và vận dụng kiến thức cao. Với tiêu chí nắm bắt nhu cầu và luôn cập nhật, ISC – QUANG TRUNG giúp học viên bổ sung và nâng cao những kỹ năng thực hành, các kinh nghiệm cần thiết đáp ứng cho yêu cầu công việc tại các dự án CNTT, góp phần giải quyết bài toán thiếu hụt nhân lực CNTT trước mắt và lâu dài cho Việt Nam, trở thành cầu nối hữu hiệu giữa các trường ĐH-CĐ và các đơn vị sử dụng nhân lực của ngành CNTT.

**Thông tin đơn vị thực tập:**

Tên đơn vị : **JISC – QUANG TRUNG**  
Tên viết tắt bằng tiếng Việt : ISC – QUANG TRUNG.  
Tên giao dịch quốc tế : IT Workforce Solution Center.

**Địa chỉ liên hệ:**

Trụ sở chính: Tòa nhà SaigonTech, Lô 14, Đường số 5, Công viên Phần mềm Quang Trung, Q. 12, Tp. HCM, Việt Nam.

Điện thoại văn phòng: (+84) 903 76 71 88

Fax:   
Email: iscquangtrung@saigontech.edu.vn  
Website: http://iscquangtrung.edu.vn



Hình 1.1. Môi trường thực tập tại ISC – Quang Trung

- Cơ sở vật chất hiện đại tại tòa nhà SaigonTech, thiết kế khoáng đạt, trang thiết bị tiện nghi theo tiêu chuẩn Hoa Kỳ, tọa lạc trong Công viên phần mềm Quang Trung - khu Công viên Phần mềm lớn nhất Việt Nam hiện nay.

- Phương tiện học tập được trang bị đầy đủ như tại các dự án CNTT thực tế.

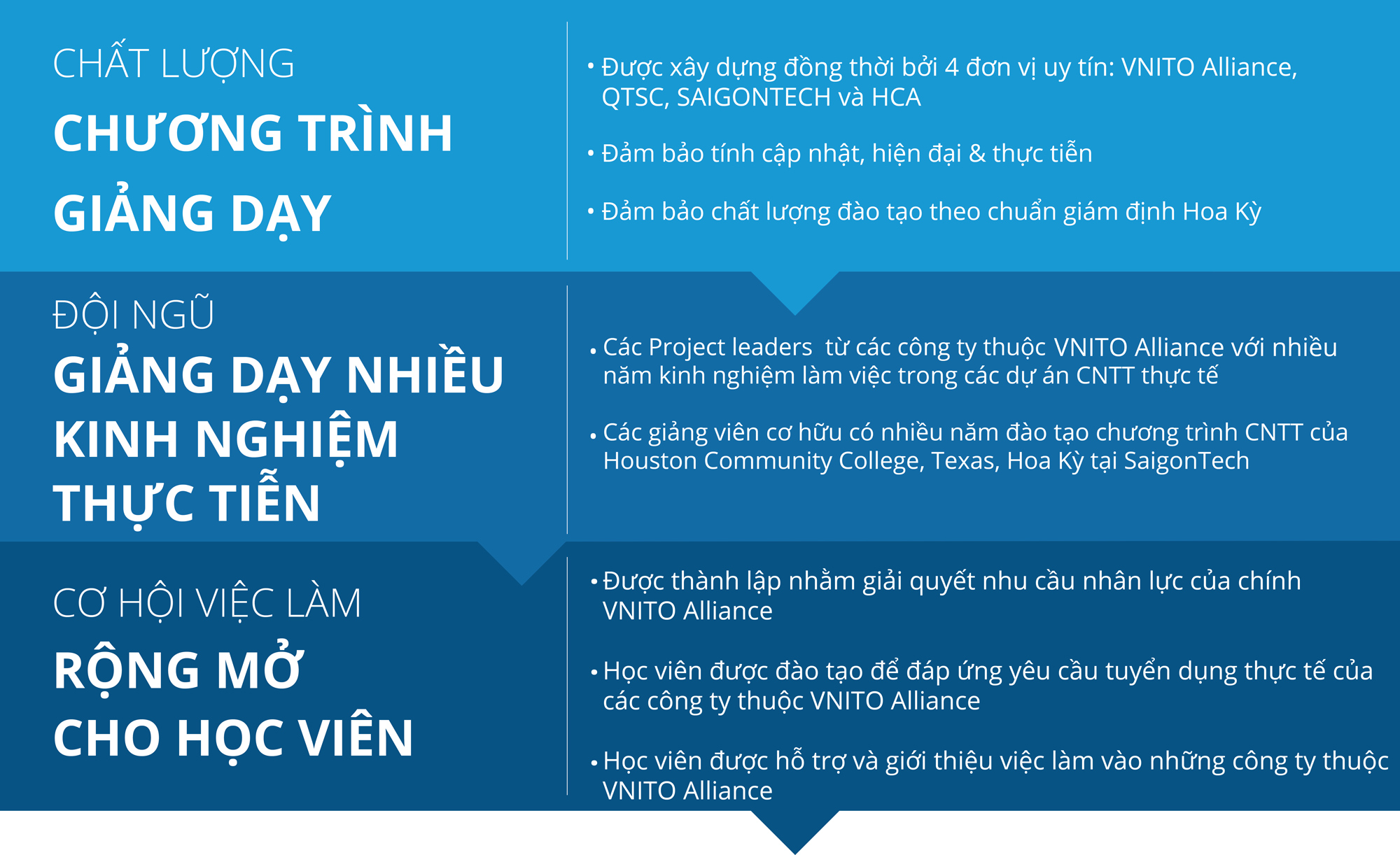
- Qui trình làm việc theo tiêu chuẩn SCRUM (Agile).

- Môi trường học tập thân thiện và thuận tiện cho giao tiếp và làm việc nhóm.

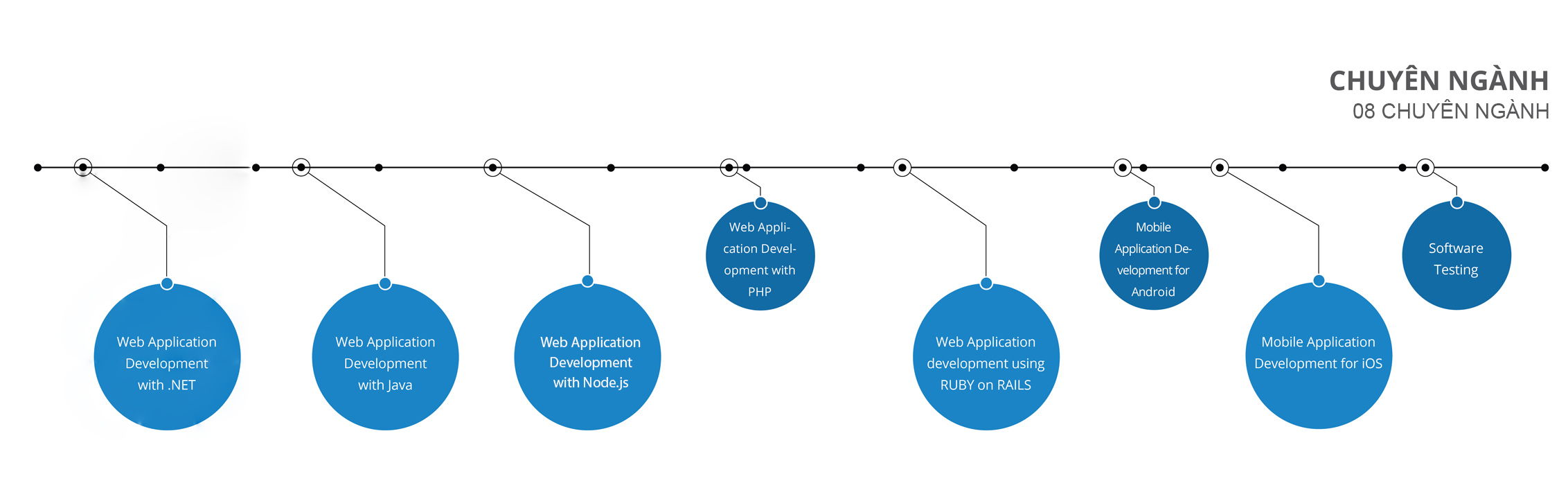
- Có đầy đủ công cụ hỗ trợ như GIT, SVN, Redmine.v.v…

- Có phòng làm việc, máy tính cho từng học viên

## I.2. Chức năng và nhiệm vụ



Hình 1.2. Thế mạnh của ISC – Quang Trung



Hình 1.3. Chuyên ngành đào tạo tại ISC – Quang Trung

# CHƯƠNG II. CƠ SỞ LÍ THUYẾT

## II.1. Giới thiệu về AngularJS



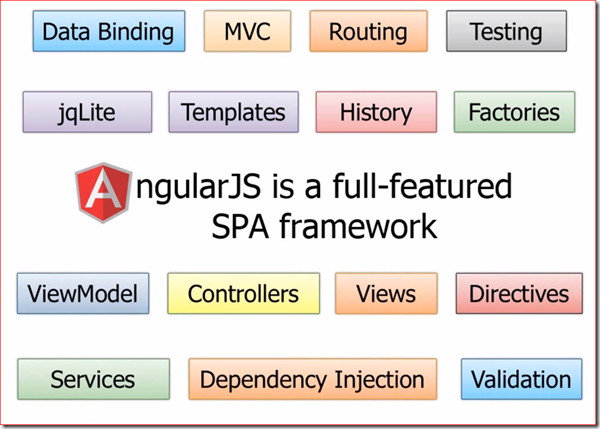
**AngularJS** là một framework có cấu trúc cho các ứng dụng web động. Nó cho phép bạn sử dụng HTML như là ngôn ngữ mẫu và cho phép bạn mở rộng cú pháp của HTML để diễn đạt các thành phần ứng dụng của bạn một cách rõ ràng và súc tích. Hai tính năng cốt lõi: Data binding và Dependency injection của AngularJS loại bỏ phần lớn code mà bạn thường phải viết. Nó xảy ra trong tất cả các trình duyệt, làm cho nó trở thành đối tác lý tưởng của bất kỳ công nghệ Server nào.

**Đặc trưng của AngularJS**

* Phát triển dự trên Javascript
* Tạo các ứng dụng client-side theo mô hình MVC.
* Khả năng tương thích cao, tự động xử lý mã javascript để phù hợp vứi mỗi trình duyệt.
* Mã nguồn mở, miễn phí hoàn toàn và được sủ dụng rộng rãi.

**Các tính năng cơ bản**

* Scope : là đối tượng có nhiệm vụ giao tiếp giữa controller và view của ứng dụng.
* Controller : xử lí dữ liệu cho đối tượng $scope, từ đây bên views sẽ sử dụng các dữ liệu trong scope để hiển thị ra tương ứng.
* Data-binding : tự động đồng bộ dữ liệu giữa model và view
* Service : là singleton object được khởi tạo 1 lần duy nhất cho mỗi ứng dụng, cung cấp các phương thức lưu trữ dữ liệu có sãn. ($http, $httpBackend, $sce, $controller, $document, $compile, $parse, $rootElement, $rootScope …..)
* Filter : Lọc các tập con từ tập item trong các mảng và trả về các mảng mới.
* Directive : dùng để tạo các thẻ HTML riêng phục vụ những mục đích riêng. AngularJS có những directive có sẵn như ngBind, ngModel…
* Temple : một thành phần của view, hiển thị thông tin từ controller
* Routing : chuyển đổi giữa các action trong controller, qua lại giữa các view.
* MVC & MVVM : mô hình thiết kế để phân chia các ứng dụng thành nhiều phần khác nhau (gọi là Model, View và Controller) mỗi phần có một nhiệm vụ nhất định. AngularJS không triển khai MVC theo cách truyền thống, mà gắn liền hơn với Model-View-ViewModel.
* Deep link : Liên kết sâu, cho phép bạn mã hóa trạng thái của ứng dụng trong các URL để nó có thể bookmark với công cụ tìm kiếm. Các ứng dụng có thể được phục hồi lại từ các địa chỉ URL với cùng một trạng thái.
* Dependency Injection: AngularJS có sẵn một hệ thống con dependency injection để giúp các lập trình viên tạo ra các ứng dụng dễ phát triển, dễ hiểu và kiểm tra.



Hình 2.1. Mô hình cấu trúc AngularJS

**Các components chính**

* ng-app : định nghĩa và liên kết một ứng dụng AngularJS tới HTML.
* ng-model : gắn kết giá trị của dữ liệu ứng dụng AngularJS đến các điều khiển đầu vào HTML.
* ng-bind : gắn kết dữ liệu ứng dụng AngularJS đến các thẻ HTML.

**Ưu điểm của angularJS**

* Cung cấp khả năng tạo ra các Single Page Aplication dễ dàng.
* Cung cấp khả năng data binding tới HTML, khiến cho người dùng cảm giác linh hoạt, thân thiện.
* Dễ dàng Unit test
* Dễ dàng tái sử dụng component
* Giúp lập trình viên viết code ít hơn với nhiều chức năng hơn.
* Chạy được trên các loại trình duyệt, trên cả PC lẫn mobile.

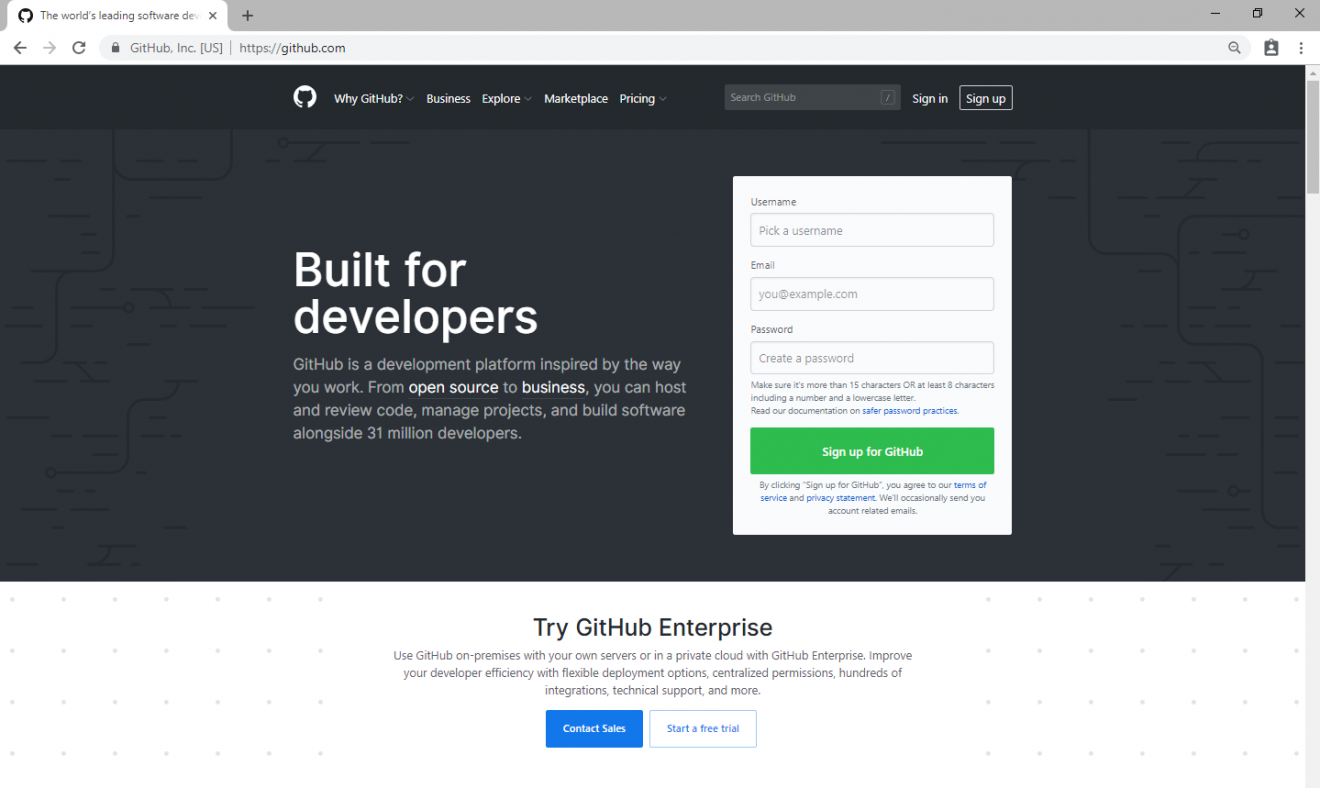
**Nhược điểm**

* Không an toàn : được phát triển từ javascript cho nên ứng dụng được viết bởi AngularJS không an toàn. Nên có sự bảo mật và xác thực phía server sẽ giúp ứng dụng trở nên an toàn hơn.
* Nếu người sử dụng ứng dụng của vô hiệu hóa JavaScript thì sẽ chỉ nhìn thấy trang cơ bản.

II.2. Tìm hiểu Githud

*II.2.1. Githud là gì?*

GitHub là một hệ thống quản lý dự án và phiên bản code, hoạt động giống như một mạng xã hội cho lập trình viên. Nó lập trình viên làm việc nhóm hiệu quả, dễ dàng hơn, nó sử dụng để hợp tác nhiều người lại với nhau, từ mọi nơi trên thế giới, lên kế hoạch, theo dõi và làm chung một dự án.



*II.2.2. Git là gì ?*

Git là một hệ quản trị phiên bản được phát triển bởi Linus Torvalds (người đã tạo ra Linux).

Hệ quản trị phiên bản là công cụ giúp giám sát những thay đổi của code như sửa lỗi, thêm tính năng. Hơn thế nữa, nó còn lưu lại thông tin ai thay đổi gì để có thể khôi phục code cũ bị xóa hoặc code đã từng được sửa. Codes không được ghi đè lên nhau vì Git lưu nhiều phiên bản copies trong repository (thư viện) của nó.

*II.2.3. Hud là gì?*

Hub trong GitHub là nơi biến những dòng lệnh, Git, thành một mạng xã hội khổng lồ cho lập trình viên.

*II.2.4. Repository là gì?*

Repository hay repo là một thư viện nơi chứa các files của dự án. Nó có thể đặt trong bộ lưu trữ của GitHub hoặc trong repository của máy tính local. Chúng ta có thể chứa files code, hình ảnh, âm thanh hoặc mọi thứ liên quan đến dự án trong một repository.

*II.2.5. Branch là gì?*

Branch là một bản sao của repository. Bạn có thể sử dụng Branch để triển khai dự án theo hướng cô lập không ảnh hưởng đến dự án chính hoặc những branches khác. Khi hoàn tất công việc, bạn có thể “Merge” (nhập) branch vào những branch khác khoặc repository chính bằng cách dùng lệnh Pull Request.

*II.2.6. Pull Request là gì?*

Pull request là cách để bạn thông báo với những người khác rằng bạn đã đẩy những thay đổi của Branch lên Repository tổng (master respository). Các cộng tác viên của repository này sẽ có thể chấp nhận hoặc từ chối pull request này. Khi nó được mở ra, bạn có thể thảo luận và xem lại công việc với những người cùng làm khác.

*II.2.7. Fork một Repository là gì?*

Fork một repository (forking a repository) giúp chúng ta có thể tạo một dự án mới dựa trên dự án cũ. Tức là, sao chép hoàn toàn một repository đã tồn tạo, tạo ra các thay đổi cần thiết, và lưu phiên bản mới này dưới dạng một repository độc lập hoàn toàn mới và gọi nó là dự án của riêng bạn.

*II.2.8. Một số lệnh cơ bản thường sử dụng trong Git*

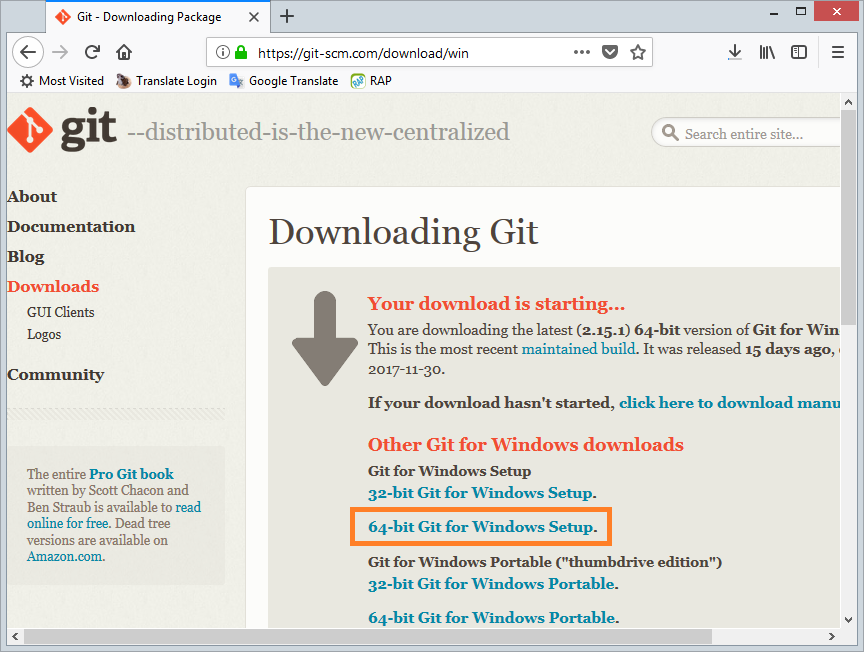
Đầu tiên, bạn sẽ cần phải config username và email mà bạn đã đăng ký trên Github. Thực tế, Git hỗ trợ 3 tuỳ chọn là --global, --system, --local. Tuy nhiên, tôi thường chọn --global.

* “git config --global user.name <username>”
* “git config --global user.email <email>”
* “git config –list” để kiểm tra lại thông tin mà bạn đã config.
* **“git init”** để tạo mới một project trên máy. Bạn có thể tạo ở bất kì đâu tuỳ thích. Sau khi chạy câu lệnh này xong, một thư mục ẩn .git sẽ xuất hiện. Thư mục này chứa thông tin cấu hình của project Git.
* **“git clone <url>”** để copy một repo. Cái remote repo đó sẽ được liên kết với tài khoản mà bạn dùng để clone. Vậy nếu bạn clone một repo thuộc về một người khác, bạn sẽ không push được lên GitHub trừ khi bạn đổi origin của nó.
* **“git status”** để biết những file nào đã bị thay đổi và những file nào được theo dõi.
* **“git add .” , “git add .<tên đuôi file>”, “git add <name\_file\_1> [<name\_file\_2> ... <name\_file\_n>]”** sau khi bạn thay đổi source code: thêm mới, sửa, xoá files,... Bạn cần phải cập nhật lên Staging Area. Câu lệnh đầu tiên dùng để cập nhật tất cả các file, câu lệnh thứ hai dùng để cập nhật dựa vào đuôi file và câu lệnh thứ ba cập nhật dựa vào tên file.
* **“git commit -m 'nội dung tuỳ thích'”** sau lệnh add, bạn cần sử dụng câu lệnh Commit để đẩy thông tin thay đổi lên Local Respository.
* **“git push origin <name\_branch>”** sau câu lệnh Commit, thông tin mới chỉ được cập nhật lên Local Repository. Nếu muốn cập nhật lên server thì bạn phải sử dụng câu lệnh push.
* **“git remote add origin <url repo của bạn>”** nếu chưa tồn tại remote trên server thì bạn cần phải add mới một remote trước rồi mới push, câu lệnh này dùng để khởi tạo một repo
* **“git fetch <name\_branch>”** để lấy source code từ server về Local Repository.
* **“git remote -v”** để xem URL của remote repo, tên mặc định của remote repo đó là origin.
* **“git remote set-url origin <url>”** sử dụng khi bạn cần đổi remote repo. Ví dụ bạn copy một cái repo của người khác và bạn muốn chuyển owner thành tài khoản GitHub của bạn.
* **“git branch”** liệt kê tất cả các branch (nhánh) trong local của bạn.
* **“git branch <tên branch>”** tạo một branch mới.
* **“git checkout <tên branch>”** để chuyển sang một nhánh đã tồn tại khác.
* **“git checkout -b <tên branch>”** để tạo một nhánh và ngay lập tức chuyển sang nhánh đó.
* **“git merge <tên branch>”** để hợp nhất branch của bạn vào branch chính (master). Bạn sẽ checkout nhánh master, rồi chạy lệnh git merge “tên branch cần merge” để hợp nhất nhánh của bạn vào nhánh master.
* **“git pull origin <tên branch>”** để kéo những thay đổi mới nhất về.
* **“git diff”** tất cả những thay đổi chưa được ghi nhận kể từ commit cuối cùng.

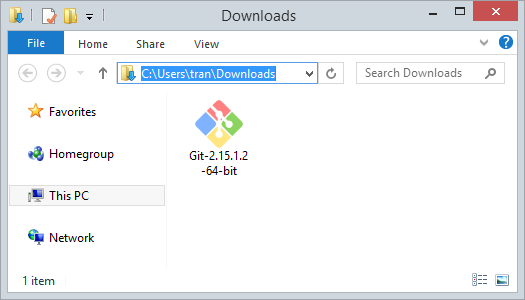
*II.2.9. Hướng dẫn cài đặt Git trên windows*

* Download Git

Truy cập vào địa chỉ dưới đây để download nếu bạn sử dụng hệ điều hành Windows: *https://git-scm.com/download/win*



Kết quả bạn download được:

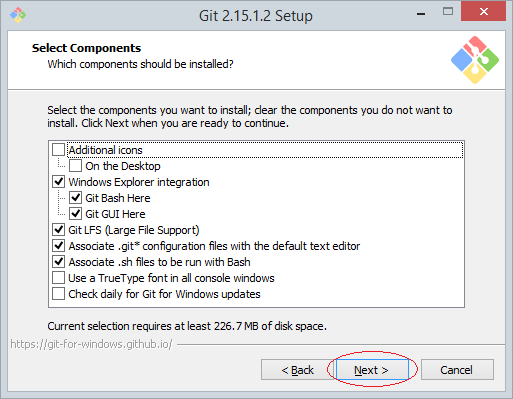
****

* **Cài đặt Git**

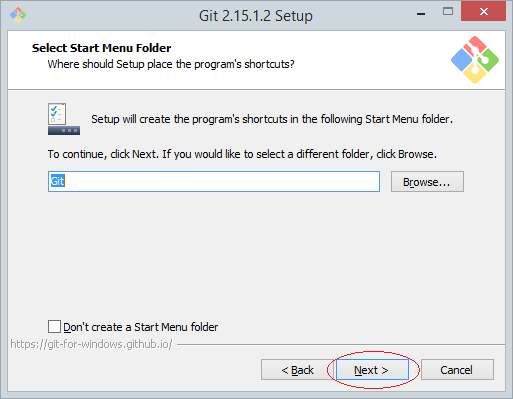
Cài đặt Git trên Windows rất đơn giản, về cơ bản bạn có thể chấp nhận các mặc định và nhấn Next, Next ... cho tới khi hoàn thành.



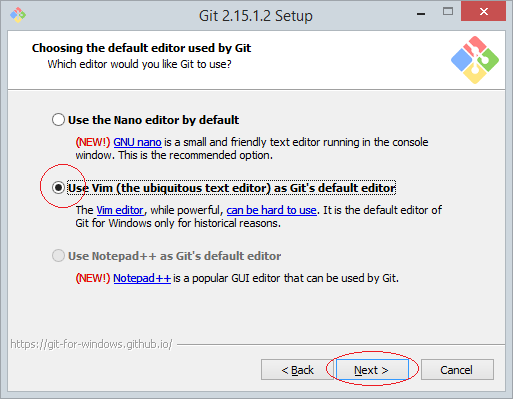
*Chọn Next*



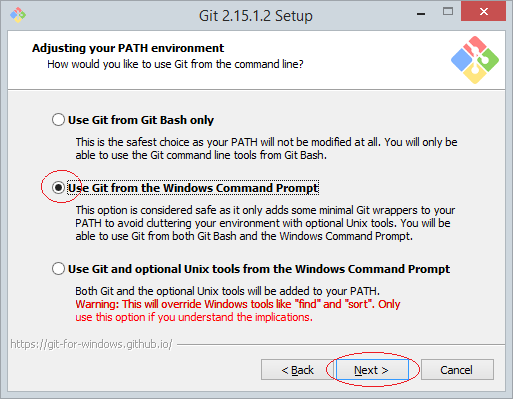
*Chọn Next*



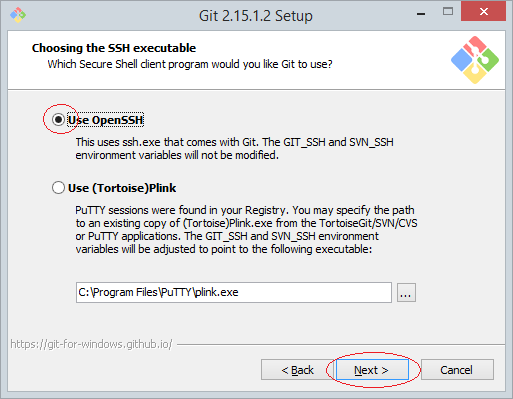
*Chọn Next*



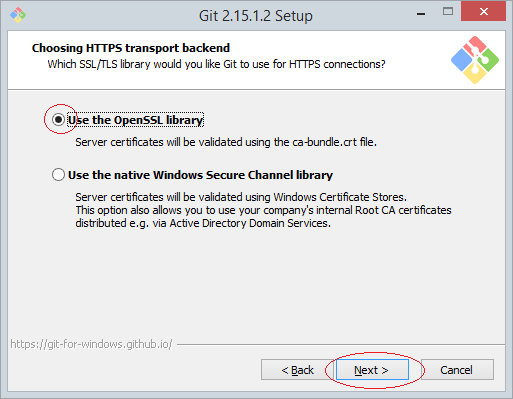
*Chọn User Vim… và chọn Next*



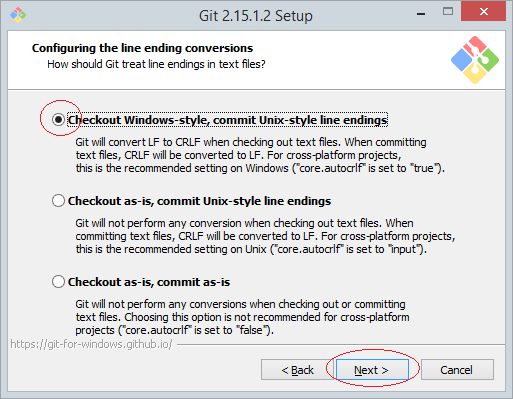
*Chọn Use Git from the Windows Command Prompt và chọ Next*



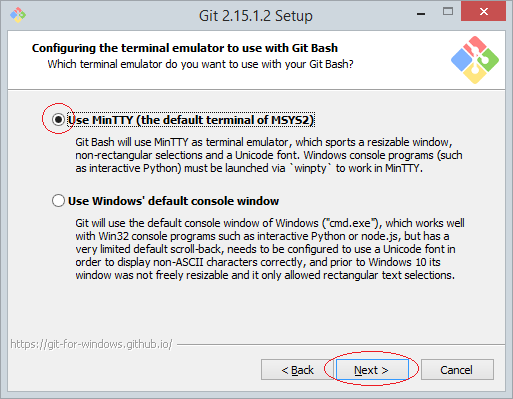
*Chọn Use OpenSSH và chọn Next*



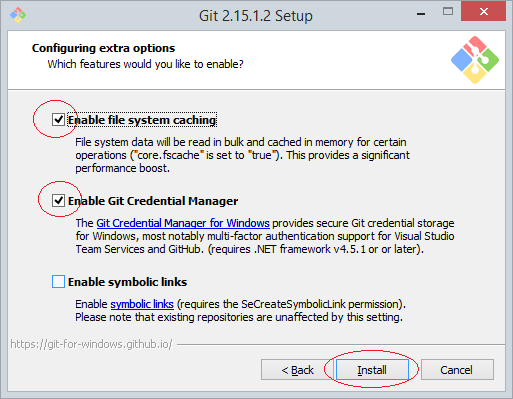
*Chọn Use the OpenSSL library và chọn Next*



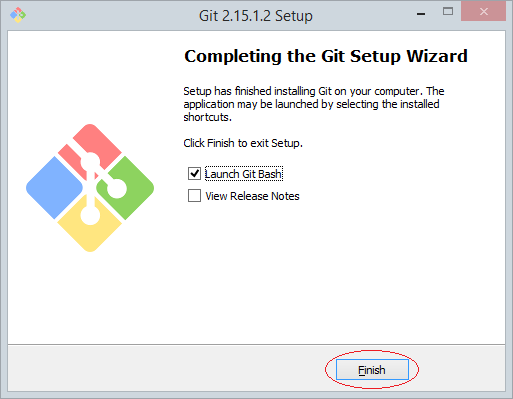
*Chọn Checkout Windows-style,… và chọn Next*



*Chọn Use MinTTY… và chọn Next*

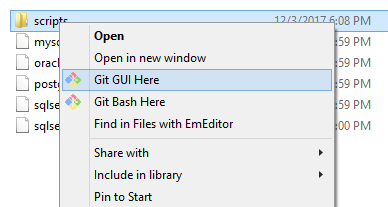


*Check Enable file system cahing, Enable Git Credential Manager và chọn Next*



*Chọn Launch Git Bash và chọn Finish*

Sau khi cài đặt xong, nhấn phải chuột vào một thư mục bất kỳ, một Context-Menu sẽ hiển thị, bạn có thể nhìn thấy các Menu-Item của Git, điều này chứng tỏ rằng bạn đã cài đặt Git thành công.



*Khi click chuột phải vào vùng bất kỳ*

## II.3. Tìm hiểu Spring Boot

* Spring Boot là gì?



Spring Boot là một dự án phát triển bởi JAV (ngôn ngữ java) trong hệ sinh thái Spring framework. Nó giúp cho các lập trình viên chúng ta đơn giản hóa quá trình lập trình một ứng dụng với Spring, chỉ tập trung vào việc phát triển business cho ứng dụng.

Để phát triển một ứng dụng web cơ bản HelloWorld sử dụng Spring framework bạn sẽ cần ít nhất 5 công đoạn sau:

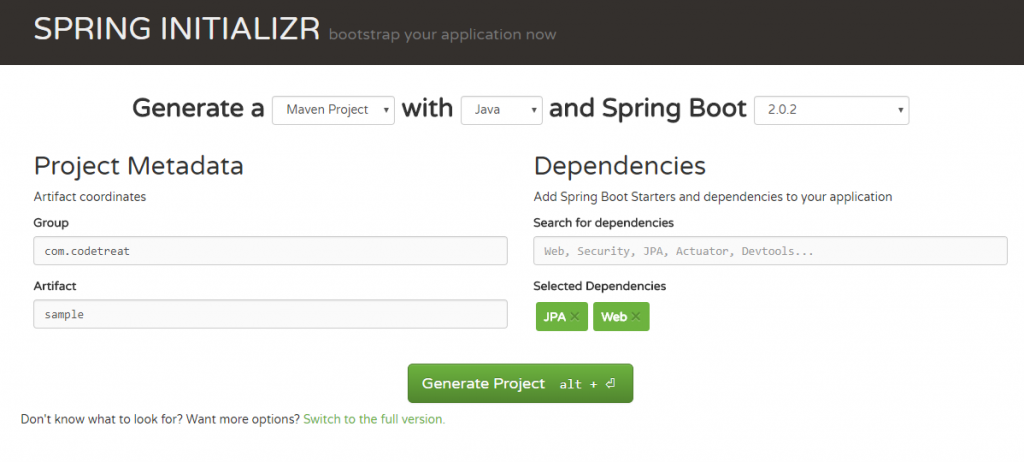
* Tạo một project sử dụng Maven với các dependency cần thiết của Spring MVC và Servlet API.
* Một tập tin web.xml để khai báo DispatcherServlet của Spring MVC.
* Một tập tin cấu hình của Spring MVC.
* Một class Controller trả về một trang “Hello World” khi có request đến.
* Cuối cùng là phải có một web server dùng để triển khai ứng dụng lên chạy.

Trong các công đoạn này, chỉ có công đoạn tạo một class Controller thì có thể khác cho các ứng dụng khác nhau vì mỗi ứng dụng có một yêu cầu khác nhau. Còn các công đoạn khác thì như nhau.

Giờ đây với Spring Boot, chúng ta có thể tạo dự án Spring một cách nhanh chóng và cấu hình cũng đơn giản dùng Sublime Text để phát triển luôn khỏi cần cài đặt eclipse hay netbean.

* **Một số tính năng nổi bật của Spring Boot:**
* Tạo các ứng dụng Spring độc lập
* Nhúng trực tiếp Tomcat, Jetty hoặc Undertow (không cần phải deploy ra file WAR)
* Các starter dependency giúp việc cấu hình Maven đơn giản hơn
* Tự động cấu hình Spring khi cần thiết
* Không sinh code cấu hình và không yêu cầu phải cấu hình bằng XML …
* **Công cụ hỗ trợ:** Spring Tools 4 for Eclipse.

Ngoài ra để tạo nhanh một ứng dụng Spring Boot bạn có thể truy cập vào trang: https://start.spring.io/



II.4. Tìm hiểu RESTfull API

*II.4.1. RESTful API là gì?*

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động…), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.



* **Diễn giải các thành phần**

**API** (Application Programming Interface) là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML.

**REST** (REpresentational State Transfer) là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, vv đến một URL để xử lý dữ liệu.

**RESTful API** là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile…) khác nhau giao tiếp với nhau.

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

### II.4.2. RESTful hoạt động như thế nào?



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

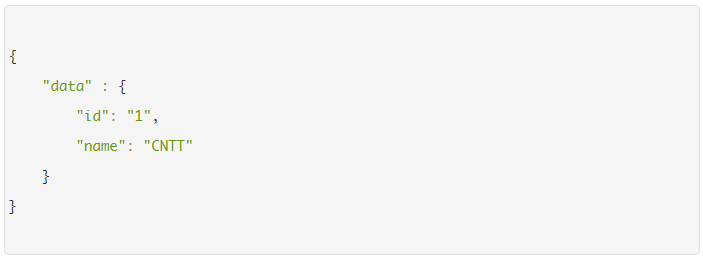
* GET (SELECT): Trả về một Resource hoặc một danh sách Resource.
* POST (CREATE): Tạo mới một Resource.
* PUT (UPDATE): Cập nhật thông tin cho Resource.
* DELETE (DELETE): Xoá một Resource.

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

Hiện tại đa số lập trình viên viết RESTful API giờ đây đều chọn JSON là format chính thức nhưng cũng có nhiều người chọn XML làm format, nói chung dùng thế nào cũng được miễn tiện và nhanh.

* **Authentication và dữ liệu trả về**

RESTful API không sử dụng session và cookie, nó sử dụng một access\_token với mỗi request. Dữ liệu trả về thường có cấu trúc như sau:



*II.4.3. Status code*

Khi chúng ta request một API nào đó thường thì sẽ có vài status code để nhận biết sau:

* 200 OK – Trả về thành công cho những phương thức GET, PUT, PATCH hoặc DELETE.
* 201 Created – Trả về khi một Resouce vừa được tạo thành công.
* 204 No Content – Trả về khi Resource xoá thành công.
* 304 Not Modified – Client có thể sử dụng dữ liệu cache.
* 400 Bad Request – Request không hợp lệ.
* 401 Unauthorized – Request cần có auth.
* 403 Forbidden – bị từ chối không cho phép.
* 404 Not Found – Không tìm thấy resource từ URI.
* 405 Method Not Allowed – Phương thức không cho phép với user hiện tại.
* 410 Gone – Resource không còn tồn tại, Version cũ đã không còn hỗ trợ.
* 415 Unsupported Media Type – Không hỗ trợ kiểu Resource này.
* 422 Unprocessable Entity – Dữ liệu không được xác thực
* 429 Too Many Requests – Request bị từ chối do bị giới hạn

II.6. Tìm hiểu về JPA

*II.6.1. JPA là gì?*

JPA (Java Persistence API) là 1 giao diện lập trình ứng dụng Java, nó mô tả cách quản lý các mối quan hệ dữ liệu trong ứng dụng sử dụng Java Platform.

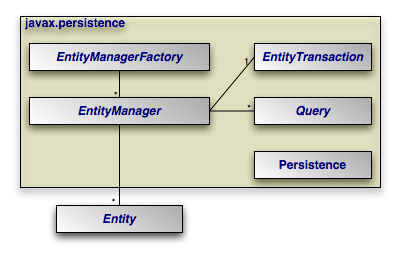
JPA cung cấp một mô hình POJO persistence cho phép ánh xạ các table/các mối quan hệ giữa các table trong database sang các class/mối quan hệ giữa các object.

Ví dụ: table Users với các column (Id, name, age…) sẽ tương ứng với class Users.java với các field Id, name, age… từ đó mỗi khi truy vấn table hay các column ta sẽ truy vấn trực tiếp trên các class, các field của class mà không cần quan tâm tới việc đang dùng loại database nào, dữ liệu database ra sao…

*II.6.2. Một số khái niệm trong JPA*

* Entity: Entity là các đối tượng thể hiện tương ứng 1 table trong cơ sở dữ liệu. Khi lập trình, entity thường là các class POJO đơn giản, chỉ gồm các method getter, setter.
* EntityManager: EntityManager là một giao diện (interface) cung cấp các API cho việc tương tác với các Entity như Persist (lưu một đối tượng mới), merge (cập nhật một đối tượng), remove (xóa 1 đối tượng).
* EntityManagerFactory: EntityManagerFactory được dùng để tạo ra một thể hiện của EntityManager.

*II.6.3. Kiến trúc JPA*



*Mô hình kiến trúc JPA*

II.7. Tìm hiểu Hibernate

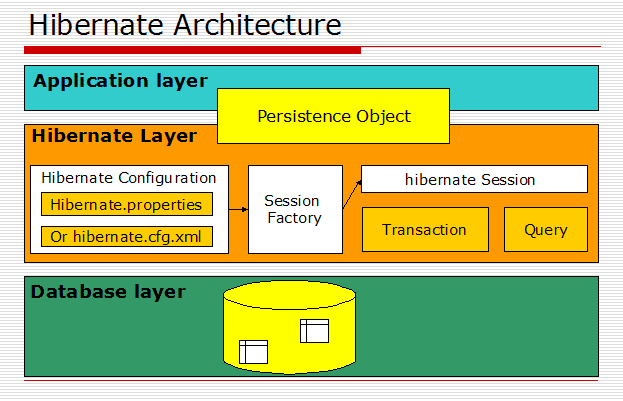
*II.7.1. Hibernate là gì?*

Hibernate là framework được sử dụng nhiều nhất hiện nay để giúp lập trình viên Java có thể map các class (Pojo) với một cơ sở dữ liệu bất kỳ.

Trước khi Hibernate ra đời, chúng ta thường thao tác với cơ sở dữ liệu thông qua JDBC. Theo thời gian, JDBC bộc lộ nhiều điểm yếu như:

* Có nhiều code thừa mà chỉ phục vụ mục đích là lấy dữ liệu.
* Mất nhiều thời gian map dữ liệu vào object Java.
* Sẽ tốn nhiều công sức khi hệ thống thay đổi CSDL (yêu cầu jdbc mới, code mới)
* Giao tiếp giữa các bảng thường khó, thiếu tính OOP trong đó.

Để giảm tải gánh nặng cho dev khi thao tác với database do đó, Hibernate đã ra đời.



Hibernate là một thư viện ORM (Object Relational Mapping) mã nguồn mở giúp lập trình viên viết ứng dụng Java có thể map các objects (pojo) với hệ quản trị cơ sở dữ liệu quan hệ, và hỗ trợ thực hiện các khái niệm lập trình hướng đối tượng với cơ dữ liệu quan hệ.

Hiểu ngắn gọn thì Hibernate là một layer đứng trung gian giữa ứng dụng và database, và chúng ta sẽ giao tiếp với Hibernate thay vì giao tiếp với database.

Để giao tiếp với Hibernate, chúng ta sẽ tạo ra một Class đại diện cho một Table trong database. Và mọi dữ liệu từ Table trong database sẽ được Hibernate bind vào Class đó cho chúng ta.

*II.7.2. Lợi ích của Hibernate Framework*

* + **Mã nguồn mở và nhẹ:** Hibernate Framework là mã nguồn mở có giấy phép LGPL và nhẹ.
  + **Hiệu suất nhanh:** Hiệu suất của Hibernate Framework là nhanh bởi vì bộ nhớ cache được sử dụng trong nội bộ Hibernate Framework. Có hai loại bộ nhớ cache trong Hibernate Framework, gồm bộ nhớ cache cấp một và bộ nhớ cache cấp hai. Bộ nhớ cache cấp một được bật bằng lệnh mặc định.
  + **Truy vấn cơ sở dữ liệu độc lập:** HQL (Hibernate Query Language) là phiên bản hướng đối tượng của SQL. Nó tạo ra các truy vấn cơ sở dữ liệu độc lập. Vì vậy, bạn không cần phải viết các truy vấn cơ sở dữ liệu cụ thể. Trước Hibernate, nếu dự án có cơ sở dữ liệu bị thay đổi, chúng ta cần phải thay đổi truy vấn SQL dẫn đến sự cố bảo trì.
  + **Tạo bảng tự động:** Hibernate framework cung cấp phương tiện để tạo ra các bảng cơ sở dữ liệu tự động. Vì vậy, không cần phải tạo ra các bảng trong cơ sở dữ liệu bằng tay.
  + **Đơn giản lệnh join phức tạp**: Có thể lấy dữ liệu từ nhiều bảng một cách dễ dàng với Hibernate framework.
  + **Cung cấp thống kê truy vấn và trạng thái cơ sở dữ liệu:** Hibernate hỗ trợ bộ nhớ cache truy vấn và cung cấp số liệu thống kê về truy vấn và trạng thái cơ sở dữ liệu.

*II.7.3. Hạn chế của Hibernate Framework*

* Không hỗ trợ các câu truy vấn phức tạp
* Một số trường hợp vẫn phải dùng native SQL do Hibernate không thể cover hết tất cả các cú pháp của các hệ quản trị cơ sử dữ liệu.
* Bị hạn chế sự can thiệp vào câu lệnh SQl do nó được tự động sinh ra.

II.8. Tìm hiểu Agile và Scrum trong phát triển phần mềm

*II.8.1. Tìm hiểu Agile*



Agile là cách thức làm phần mềm linh hoạt để làm sao đưa sản phẩm đến tay người dùng càng sớm, càng tốt và được xem như là sự cải tiến (ở một khía cạnh nào đó) khi đặt cạnh những mô hình cũ như Mô hình “Thác nước (waterfall)”.

Tuyên ngôn Agile (Agile Manifesto)

*“****Cá nhân và sự tương tác*** *hơn là quy trình và công cụ;*

***Phần mềm chạy tốt*** *hơn là tài liệu đầy đủ;*

***Cộng tác với khách hàng*** *hơn là đàm phán hợp đồng;*

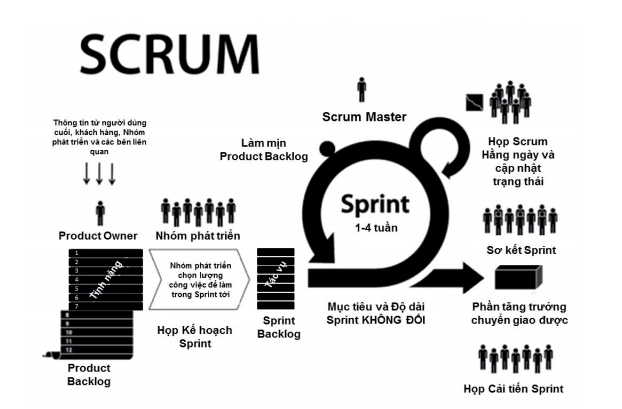
***Phản hồi với các thay đổi*** *hơn là bám sát kế hoạch.”*



*Bốn giá trị của Agile*

*II.8.2. Tìm hiểu Scrum*

Scrum cơ bản là bộ khung làm việc (framework) hay có thể hiểu nôm na là cách thức làm việc để trở nên “linh hoạt” trong phát triển phần mềm.



Hình 2.3. Tóm tắt Scrum

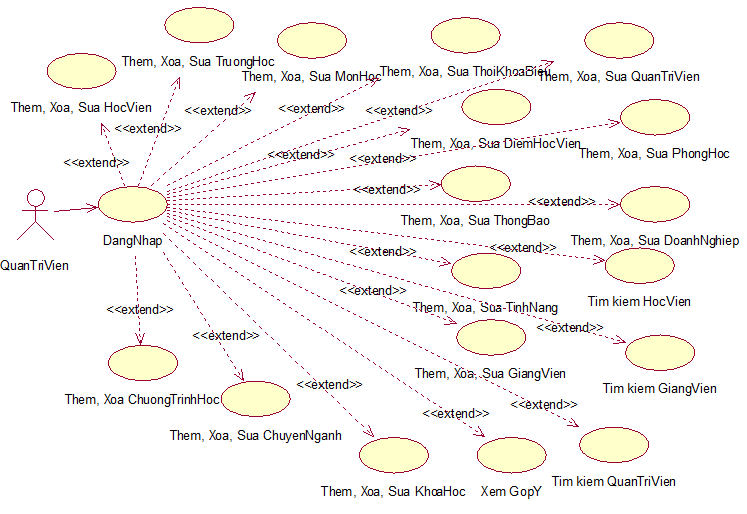
**Vai trò**: Là một cá nhân, hoặc một nhóm tham gia vào trong quá trình vận hành của Scrum. Trong Scrum có 3 vai trò chính: Product Owner, ScrumMaster và Nhóm Phát triển.

**Sự kiện**: Là các hoạt động xảy ra trong suốt vòng đời của quá trình phát triển. Các sự kiện chính trong Scrum là: Sprint, Lập kế hoạch Sprint, Scrum Hằng ngày, Sơ kết Sprint và Cải tiến Sprint.

**Tạo tác**: Là các công cụ hoặc kết quả được tạo ra và sử dụng trong quá trình vận hành Scrum. Các tạo tác trong Scrum bao gồm: Product Backlog, Sprint Backlog, Phần tăng trưởng Sản phẩm.

# CHƯƠNG III. PHẦN THỰC HÀNH

III.1. Phân tích hệ thống



Hình 3.1. Biểu đồ UC tổng quát

III.2. Thiết kế cơ sở dữ liệu

Bảng 2.1. Bảng GIANGVIEN: Lưu trữ thông tin giảng viên

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| MACBGV | String |  |
| HO | String |  |
| TENLOT | String |  |
| TEN | String |  |
| GIOITINH | String |  |
| NGAYSINH | Date |  |
| NOISINH | String |  |
| CMND | Integer |  |
| SDT | String |  |
| EMAIL | String |  |
| DIACHI | String |  |
| LUONG | Interger |  |
| HSLUONG | Integer |  |
| NGAYVAOLAM | Date |  |
| PASSWORD | String |  |
| ACCOUNTTYPE | Integer |  |

Bảng 2.2. Bảng CHUYENNGANH: Lưu trữ thông tin chuyên ngành

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| ID | Integer | Tự động tăng |
| TENCN | String |  |

Bảng 2.3. Bảng DANHMUCHOCVIEN: Lưu trữ thông tin về loại học viên

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| ID | Integer | Tự động tăng |
| TENLOAI | String |  |

Bảng 2.4. Bảng DAOTAO: Lưu trữ thông tin về chương trình học

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| MAMH\_ID | String |  |
| CHUYENNGANH\_ID | Integer |  |

Bảng 2.5. Bảng DIEMMONHOC: Lưu trữ thông tin điểm của học viên

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| ID | Integer | Tự động tăng |
| MAHV | String |  |
| MAGV | String |  |
| DIEMGK | Integer |  |
| DIEMHP | Integer |  |
| MAMH | String |  |
| NGAYNHAP | Date |  |
| KETQUA | String |  |

Bảng 2.6. Bảng DOANHNGHIEP: Lưu trữ thông tin về doanh nghiệp liên kết với trung tâm

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| ID | Integer | Tự động tăng |
| TENDOANHNGHIEP | String |  |
| DIACHI | String |  |
| SDT | String |  |

Bảng 2.7. Bảng GOPY: Lưu trữ thông tin góp ý của học viên

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| ID | Integer | Tự động tăng |
| NGUOIGOPY | String |  |
| NOIDUNG | String |  |
| NGAYNHAN | Date |  |

Bảng 2.8. Bảng HOCVIEN: Lưu trữ thông tin học viên

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| MAHV | String |  |
| HO | String |  |
| TENLOT | String |  |
| TEN | String |  |
| GIOITINH | String |  |
| NGAYSINH | Date |  |
| NOISINH | String |  |
| CMND | Integer |  |
| SDT | String |  |
| EMAIL | String |  |
| DIACHI | String |  |
| PASSWORD | String |  |
| MATRUONG | String |  |
| ACCOUNTTYPE | Integer | Loại tài khoản |
| MAKHOAHOC | String |  |
| TYPE | Integer | Loại học viên |
| TGCOTHEDILAM | Date |  |
| ANHDAIDIEN | String |  |

Bảng 2.9. Bảng HOSO: Lưu trữ thông tin hồ sơ của học viên

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| ID | Integer | Tự động tăng |
| MAHV | String |  |
| BANGDIEM | Boolean |  |
| CMND | Boolean |  |
| DATCOC | Boolean |  |
| HINHANH | Boolean |  |
| CHUTHICH | Boolean |  |

Bảng 2.10. Bảng KHOAHOC: Lưu trữ thông tin khoá học

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| MAKHOAHOC | String |  |
| TENKHOAHOC | String |  |
| NGAYBD | Date |  |
| NGAYKT | Date |  |
| CHUYENNGANH\_ID | Integer |  |

Bảng 2.11. Bảng LOAITAIKHOAN: Lưu trữ thông tin loại tài khoản

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| ID | Integer | Tự động tăng |
| TENLOAI | String |  |

Bảng 2.12. Bảng MONHOC: Lưu trữ thông tin môn học

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| MAMH | String |  |
| TENMH | String |  |
| SOGIO | Integer |  |
| SOTC | Integer |  |

Bảng 2.13. Bảng NGUOIHUONGDAN: Lưu trữ thông tin danh sách giảng viên hướng dẫn tại trường của học viên

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| MANGUOIHDAN | Integer | Tự động tăng |
| HO | String |  |
| TENLOT | String |  |
| TEN | String |  |
| SDT | String |  |
| EMAIL | String |  |
| TRUONG | String |  |

Bảng 2.14. Bảng NGUOIQUANLY: Lưu trữ thông tin quản trị viên hệ thống

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| MANQL | String |  |
| HO | String |  |
| TENLOT | String |  |
| TEN | String |  |
| GIOITINH | String |  |
| NGAYSINH | Date |  |
| NOISINH | String |  |
| CMND | Integer |  |
| SDT | String |  |
| EMAIL | String |  |
| DIACHI | String |  |
| LUONG | Interger |  |
| HSLUONG | Integer |  |
| NGAYVAOLAM | Date |  |
| PASSWORD | String |  |
| ACCOUNTTYPE | Integer |  |

Bảng 2.15. Bảng PHONGHOC: Lưu trữ thông tin phòng học

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| MAPHG | String |  |
| TENPHG | String |  |

Bảng 2.16. Bảng TINHNANG: Lưu trữ thông tin menu chức năng của trang người dùng dựa vào loại tài khoản (học viên/ giảng viên)

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| ID | Integer | Tự động tăng |
| TENMENU | String |  |
| LINK | String |  |
| TYPE | Integer |  |

Bảng 2.17. Bảng THOIKHOABIEU: Lưu trữ thông tin thời khoá biểu cho học viên và giảng viên

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| ID | Integer | Tự động tăng |
| MAKHOAHOC | String |  |
| MACBGV | String |  |
| MAMH | String |  |
| MAPHG | String |  |
| GIOBD | Time |  |
| GIOKT | Time |  |
| NGAYHOC | Date |  |

Bảng 2.18. Bảng THONGBAO: Lưu trữ thông tin nội dung thông báo

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| ID | Integer | Tự động tăng |
| TIEUDE | String |  |
| NOIDUNG | String |  |
| NGUOIDANG | String |  |
| NGAYDANG | Date |  |
| MAKHOAHOC | String |  |

Bảng 2.19. Bảng TRUONGHOC: Lưu trữ thông tin trường học của học viên

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu dữ liệu | **Mô tả** |
| MATRUONG | String |  |
| TENTRUONG | String |  |
| DIACHI | String |  |
| SDT | String |  |

III.3. Khởi tạo server Spring Boot

*III.3.1. Công cụ phát triển*

* Công cụ tạo service (Back End):

Để lập trình back end hay cụ thể là viết code java spring boot chúng ta sử dụng các công cụ sau:

+ JDK 8

Link download: *https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html*

+ Spring Tools 4 for Eclipse

Link download công cụ: *https://spring.io/tools*

Link download: *https://tomcat.apache.org/download-90.cgi*

+ Postman

Link download: *https://www.getpostman.com/downloads/*

* Công tạo giao diện người dùng (Front End)

Để thiết kế giao diện người dùng hay cụ thể là viết code html, css, javascrip, angularJS thì chúng ta có thể sử dụng các công cụ sau:

+ Atom: Một số plugin bổ trợ: atom beautify, atom html preview, less-autocompile, linter-bootlint

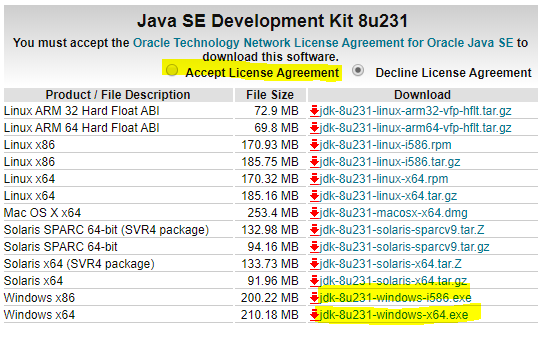
Link download: *https://spring.io/tools*

+ Visual Studio Code:

Link download: *https://code.visualstudio.com/*

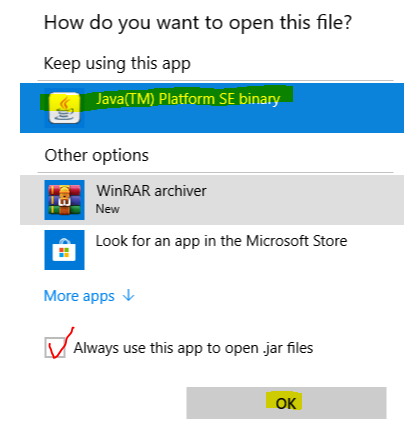
*III.3.2. Tạo môi trường và project*

- Để chạy được java chúng ta cần tạo môi trường để chạy java bằng cách cài đặt JDK 8



*Tiến hành tải và cài jdk 8*

Tiến hành giải nén tập tin Spring Tool Suite đã tải

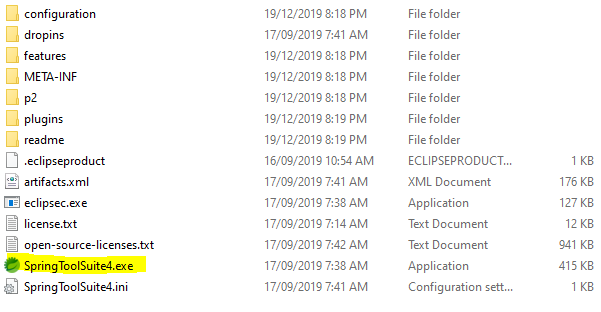


*Chọn ứng dụng chạy file Spring Tool Suite .jar để tiến hành giải nén*



*Spring Tool Suite đã giải nén*

Tiến hành chạy file spring tool suite .exe để khởi động ứng dụng



Chạy file SpringToolSuite4.exe để khởi động ứng dụng

* Tạo Servers Spring Boot

Để khởi tạo dự án nhanh chúng ta có thể vào: <https://start.spring.io/> để tạo project spring boot đúng chuẩn

- Chọn project maven

- Chọn ngôn ngữ Java

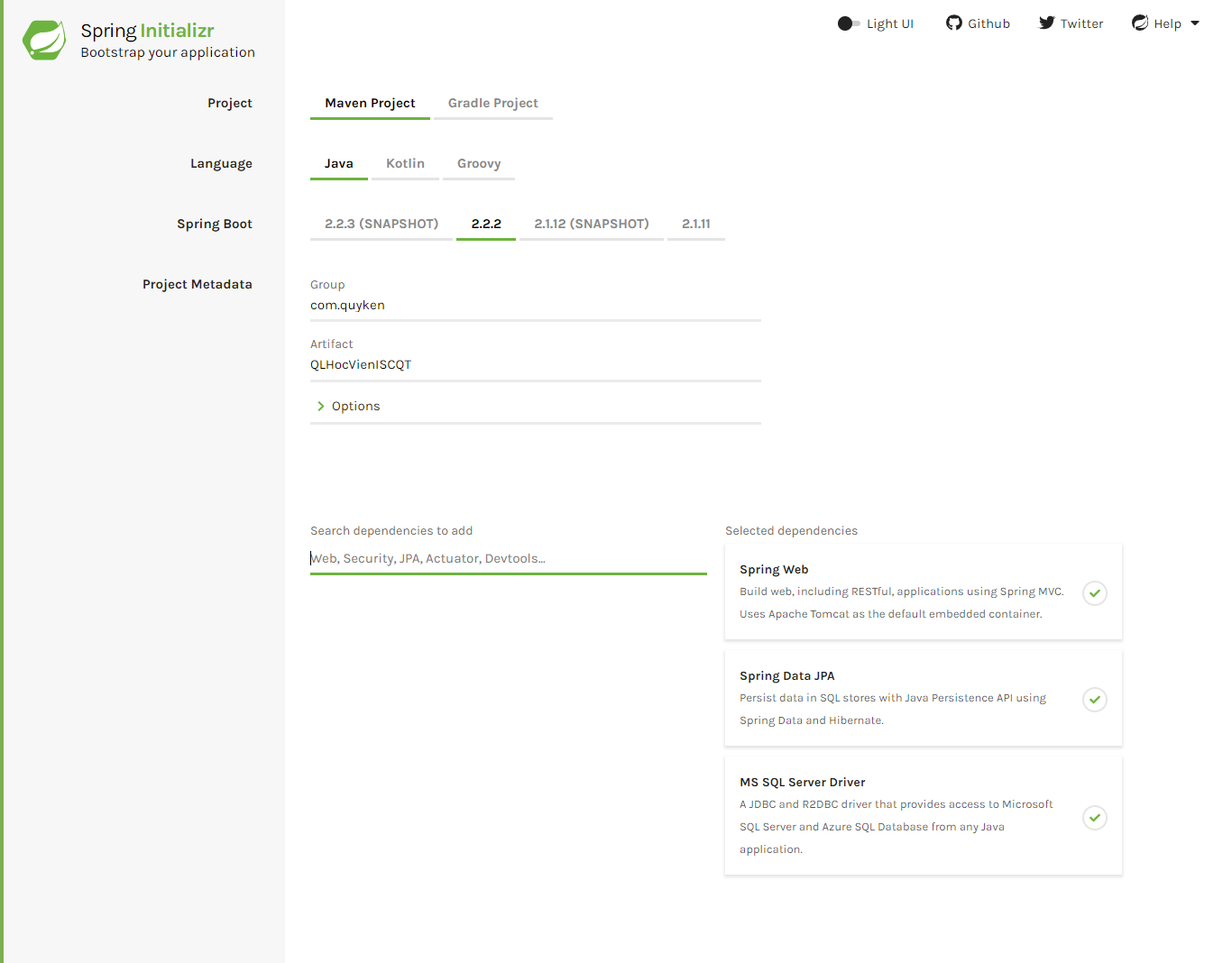
- Phiên bản ứng dụng

- Tên domain (Group)

- Tên project (Artifact)

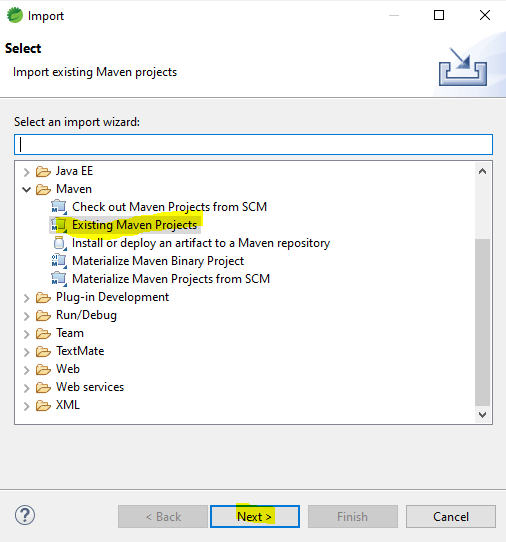
- Tích hợp một số dependencies cần thiết: Spring Web, Spring Data JPA, MS SQL Server Driver.

-> Bấm Generate để tải xuống ứng dụng

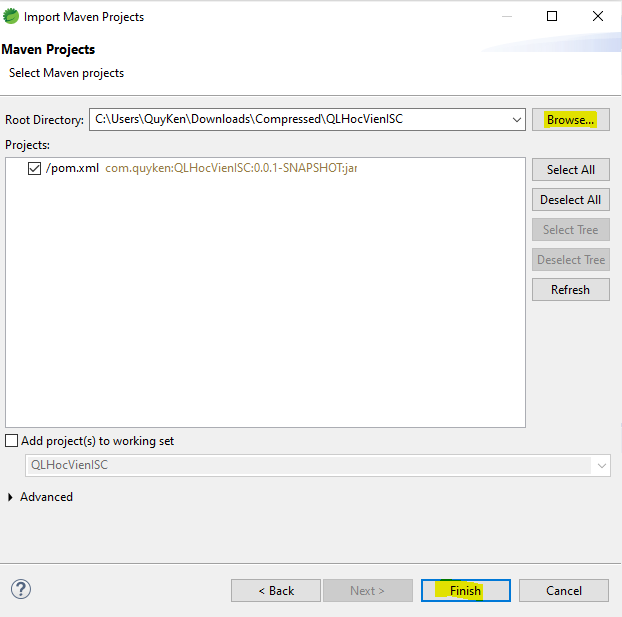


*Tạo project online với* [*https://start.spring.io/*](https://start.spring.io/)

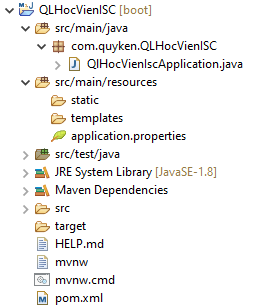
Sau khi tải ứng dụng -> tiến hành giải nén -> vào spring tool suite chọn File -> Import -> Maven -> Existing Maven Projects -> Next



*Tiến hành import ứng dụng*



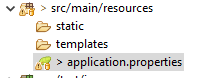
Chọn Browse.. để chọn đến thư mục ứng dụng đã giải nén -> Finish (Đợi quá trình import và tải thư viện cần thiết về máy, quá trình này cần có mạng)



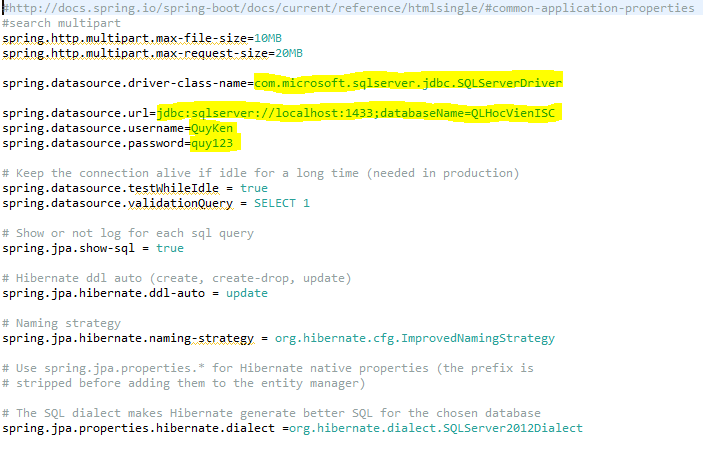
*Cấu trúc của project sau khi tạo xong*

* Xây dựng API
* Cấu hình file kết nối đến database

Vào file application.properties trong src/main/resources để cấu hình file kết nối với SQL Server gồm username, password, port (mặc định 1433), tên database trong SQL Server.



*Thư mục chứ file kết nối database*

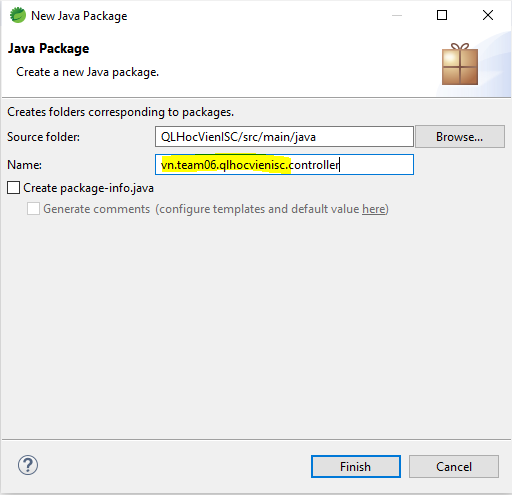


*Nội dung file kết nối đến database*

* Tạo cấu trúc thư mục

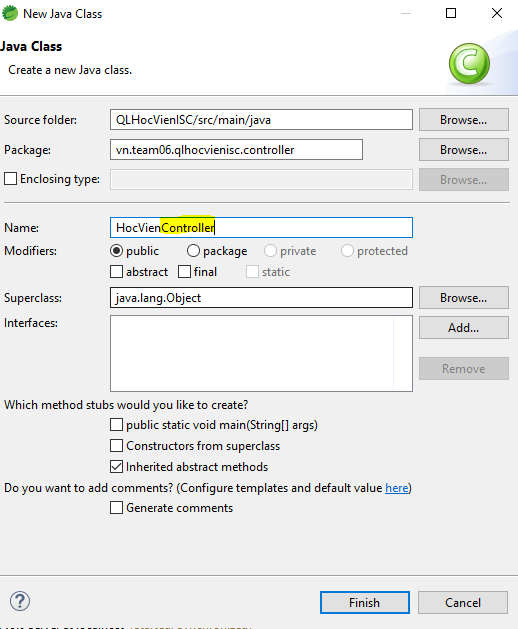


Để tạo các package ta Clich chuột phải vào src/main/java -> New -> Package



Nhập tên package, chúng ta nên đặt tên package theo quy tắt domain + tên package ví dụ: vn.team06.qlhocvienisc.controller -> Finish để tạo package.

Sau khi tạo xong package chúng ta cần tạo các class xử lý bằng cách nhấn chuột phải vào package cần tạo class -> New -> Class



Nhập tên class, tên class chúng ta nên đặt theo nguyên tắt tên class + chức năng của package (Controller, Service, Repository), riêng Entity thì chúng ta chỉ cần có tên class là đủ rồi-> Finish để tạo class.



Chúng ta cần tạo class RestConfig nằm chung với class Application lúc tạo project thì hệ thống đã tạo sẵn class Application, tạo class RestConfig để không bị trình duyệt chặn Cross khi GET, PUT, POST, DELETE, OPTIONS.



*Nội dung file RestConfig*

* Entity: Để khai báo đối tượng và tương tác với với dữ liệu trong Database.

*Một số Annotation sử dụng trong Entity:*

* “@Entity” đánh dấu một class là một Entity.
* “@Table(name=’tên bảng’) đánh dấu class này tương ứng với một bảng trong database. Chúng ta cũng có thể bỏ qua annotation này bằng cách thêm thuộc tính name=’tên bảng’ vào trực tiếp @Entity.
* “@Id” đánh dấu một biến ở dưới annotation này là primary key của table này.
* “@GeneratedValue(strategy = GenerationType.IDENTITY)” tự động tăng giá trị khi insert.
* “@Column(name = "tên trường” để chỉ định tên trường tương ứng như trong database. Bạn cũng có thể bỏ qua annotation này nếu đặt tên trường trong class giống với tên trường trong database. Nếu khi bạn sử dụng annotion này mà trong database không có tên trường chỉ định thì hibernate sẽ tự tạo trường này vào table database tương ứng.

*Code minh hoạ:*

**package vn.team06.qlhocvienisc.entity;**

**import java.sql.Date;**

**import java.util.List;**

**import javax.persistence.CascadeType;**

**import javax.persistence.Column;**

**import javax.persistence.Entity;**

**import javax.persistence.FetchType;**

**import javax.persistence.Id;**

**import javax.persistence.JoinColumn;**

**import javax.persistence.ManyToOne;**

**import javax.persistence.OneToMany;**

**import javax.persistence.Table;**

**import com.fasterxml.jackson.annotation.JsonManagedReference;**

**@Entity**

**@Table(name = "HOCVIEN")//Kết nối đến bảng HOCVIEN trong database**

**public class HocVien {**

**@Id //Gán MAHV là khoá chinh**

**@Column(columnDefinition = "LONGVARBINARY")**

**private String MAHV;**

**private String HO;**

**private String TENLOT;**

**private String TEN;**

**private String GIOITINH;**

**private Date NGAYSINH;**

**private String NOISINH;**

**private int CMND;**

**private int SDT;**

**private String EMAIL;**

**private String DIACHI;**

**private String PASSWORD;**

**private Date TGCOTHEDILAM;**

**private String MATRUONG;**

**private String MAKHOAHOC;**

**private String ACCOUNTTYPE;**

**private String TYPE;**

**private String ANHDAIDIEN;**

**//Quan hệ nhiều một với entity TruongHoc**

**@ManyToOne(fetch = FetchType.EAGER)**

**@JoinColumn(name="MATRUONG",insertable=false, updatable=false)**

**private TruongHoc truonghoc;**

**//Quan hệ nhiều một với entity KhoaHoc**

**@ManyToOne(fetch = FetchType.EAGER)**

**@JoinColumn(name="MAKHOAHOC", insertable=false, updatable=false)**

**private KhoaHoc khoahoc;**

**//Quan hệ nhiều một với entity LoaiTaiKhoan**

**@ManyToOne(fetch = FetchType.EAGER)**

**@JoinColumn(name="ACCOUNTTYPE", insertable=false, updatable=false)**

**private LoaiTaiKhoan loaitaikhoan;**

**//Quan hệ nhiều một với entity DanhMucTaiKhoan**

**@ManyToOne(fetch = FetchType.EAGER)**

**@JoinColumn(name="TYPE", insertable=false, updatable=false)**

**private DanhMucHocVien danhmuchocvien;**

**//Quan hệ một nhiều với entity HoSo**

**@OneToMany(cascade = {CascadeType.REMOVE}, mappedBy = "hocvien")**

**@Column(nullable = true)**

**@JsonManagedReference**

**private List<HoSo> hoso;**

**//Quan hệ một nhiều với entity DiemMonHoc**

**@OneToMany(cascade = {CascadeType.REMOVE}, mappedBy = "hocvien")**

**@Column(nullable = true)**

**@JsonManagedReference**

**private List<DiemMonHoc> diemmonhoc;**

**//Quan hệ một nhiều với entity GopY**

**@OneToMany(cascade = {CascadeType.REMOVE}, mappedBy = "hocvien")**

**@Column(nullable = true)**

**@JsonManagedReference**

**private List<GopY> gopy;**

**//Tạo thêm các Constructor bằng cách bấm tổ hợp phím: Alt + Shift + S -> Generate Getters and Setters… và Generate toString()…**

**}**

* **Controller:** Nhận các yêu cầu từ người dùng và chuyển đến các phương thức xử lý tương ứng và trả về kết quả cho người dùng.

*Một số Annotation sử dụng trong Controller:*

* “@RestController” để khai báo một class thành Controller
* “@Autowired” để khai báo quan hệ giữa các đối tượng (chứa nhau, phụ thuộc nhau) ví dụ khi ta có bean HocVienController để chứa bean HocVienService thì ta khai báo @Autowired trên HocVienService.
* “@RequestMapping(value = "/đường dẫn/{tham số nếu có}")” để nhận các giá trị mà người dùng request đến server.

*Code minh hoạ:*

**package** vn.team06.qlhocvienisc.controller;

**import** java.util.List;

**import** javax.validation.Valid;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpHeaders;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.web.bind.annotation.RestController;

**import** vn.team06.qlhocvienisc.entity.HocVien;

**import** vn.team06.qlhocvienisc.service.HocVienService;

@RestController

**public** **class** HocVienController {

@Autowired

HocVienService hocvienService;

//Trả về danh sách tất cả học viên

@RequestMapping(value = "/hocvien")

**public** List<HocVien> hocvien() {

**return** hocvienService.getAll();

}

//Thêm học viên

@RequestMapping(value = "/hocvien", method = RequestMethod.***POST***)

**public** HocVien createHocvien(@Valid @RequestBody HocVien hocvien)

{

**return** hocvienService.createHocvien(hocvien);

}

//Xoá học viên dựa vào mã học viên

@RequestMapping(value = "/hocvien/{id}", method = RequestMethod.***DELETE***)

**public** ResponseEntity<Object> deleteHocvien(@PathVariable(value = "id") String id)

{

**return** hocvienService.deleteHocvien(id);

}

//Cập nhập học viên dựa vào mã học viên

@RequestMapping(value = "/hocvien/{id}", method = RequestMethod.***PUT***)

**public** HocVien updateHocvien(@PathVariable(value = "id") String id, @Valid @RequestBody HocVien hocvien)

{

**return** hocvienService.updateHocvien(id, hocvien);

}

//Trả về danh sách tất cả học viên có phân trang và sắp xếp

@RequestMapping(value = "/hocvien2", method = RequestMethod.***GET***)

**public** ResponseEntity<List<HocVien>> getAllHocVien(

@RequestParam(defaultValue = "0") Integer pageNo,

@RequestParam(defaultValue = "2") Integer pageSize,

@RequestParam(defaultValue = "1") **int** typeSort,

@RequestParam(defaultValue = "ISC09") String maKH,

@RequestParam(defaultValue = "MAHV") String sortBy)

{

List<HocVien> list = hocvienService.getAllHocVien(pageNo, pageSize, typeSort, maKH, sortBy);

**return** **new** ResponseEntity<List<HocVien>>(list, **new** HttpHeaders(), HttpStatus.***OK***);

}

// Tìm kiếm học viên dựa vào mã học viên

@RequestMapping(value = "/timkiemhocvien/{id}", method = RequestMethod.***GET***)

**public** HocVien timkiemHocvien(@PathVariable(value = "id") String id)

{

**return** hocvienService.timkiemHocvien(id);

}

}

* Service: để tạo ra các phương thức xử lý dữ liệu người dùng gửi lên, tương tương tác với Repository để gọi các hàm xử lý tương ứng trên các đối tượng và trả về kết quả cho Controller để Controller trả kết quả về người dùng.

*Một số Annotation trong Service:*

* “@Service” để khai báo một class thành service

*Code minh hoạ:*

**package** vn.team06.qlhocvienisc.service;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Optional;

**import** javax.persistence.EntityNotFoundException;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.data.domain.Page;

**import** org.springframework.data.domain.PageRequest;

**import** org.springframework.data.domain.Pageable;

**import** org.springframework.data.domain.Sort;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.stereotype.Service;

**import** vn.team06.qlhocvienisc.entity.HocVien;

**import** vn.team06.qlhocvienisc.repository.HocVienRepository;

@Service

**public** **class** HocVienService {

@Autowired

HocVienRepository hocvienRepository;

//Trả về danh sách tất cả học viên

**public** List<HocVien> getAll() {

**return** (List<HocVien>) hocvienRepository.findAll();

}

//Thêm học viên

**public** HocVien createHocvien(HocVien hocvien) {

**return** hocvienRepository.save(hocvien);

}

//Cập nhập học viên dựa vào mã học viên

**public** HocVien updateHocvien(String Id, HocVien hocvien)

{

HocVien updatedHV;

Optional<HocVien> searchEntity = hocvienRepository.findById(Id);

**if** (searchEntity.isPresent()) {

HocVien sv = searchEntity.get();

sv.setHO(hocvien.getHO());

sv.setTENLOT(hocvien.getTENLOT());

sv.setTEN(hocvien.getTEN());

sv.setGIOITINH(hocvien.getGIOITINH());

sv.setNGAYSINH(hocvien.getNGAYSINH());

sv.setNOISINH(hocvien.getNOISINH());

sv.setCMND(hocvien.getCMND());

sv.setSDT(hocvien.getSDT());

sv.setEMAIL(hocvien.getEMAIL());

sv.setDIACHI(hocvien.getDIACHI());

sv.setPASSWORD(hocvien.getPASSWORD());

sv.setMATRUONG(hocvien.getMATRUONG());

sv.setACCOUNTTYPE(hocvien.getACCOUNTTYPE());

sv.setMAKHOAHOC(hocvien.getMAKHOAHOC());

sv.setTYPE(hocvien.getTYPE());

sv.setANHDAIDIEN(hocvien.getANHDAIDIEN());

sv.setTGCOTHEDILAM(hocvien.getTGCOTHEDILAM());

updatedHV = hocvienRepository.save(sv);

} **else** {

**throw** **new** EntityNotFoundException();

}

**return** updatedHV;

}

//Xoá học viên dựa vào mã học viên

**public** ResponseEntity<Object> deleteHocvien(String Id)

{

Optional<HocVien> hocvien = hocvienRepository.findById(Id);

**if** (hocvien.isPresent()) {

HocVien hv = hocvien.get();

hocvienRepository.delete(hv);

} **else** {

**throw** **new** EntityNotFoundException();

}

**return** ResponseEntity.*ok*().build();

}

//Trả về danh sách tất cả học viên có phân trang và sắp xếp

**public** List<HocVien> getAllHocVien(Integer pageNo, Integer pageSize,**int** typeSort, String maKH, String sortBy)

{

**if**(typeSort==1)

{

Pageable paging = PageRequest.*of*(pageNo, pageSize, Sort.*by*(Sort.Direction.***ASC***, sortBy));

Page<HocVien> pagedResult = hocvienRepository.findCourse(maKH,paging);

**if**(pagedResult.hasContent()) {

**return** pagedResult.getContent();

} **else** {

**return** **new** ArrayList<HocVien>();

}

}

**else**

{

Pageable paging = PageRequest.*of*(pageNo, pageSize, Sort.*by*(Sort.Direction.***DESC***, sortBy));

Page<HocVien> pagedResult = hocvienRepository.findAll(paging);

**if**(pagedResult.hasContent()) {

**return** pagedResult.getContent();

} **else** {

**return** **new** ArrayList<HocVien>();

}

}

}

// Tìm kiếm học viên dựa vào mã học viên

**public** HocVien timkiemHocvien(String Id)

{

Optional<HocVien> hocvien = hocvienRepository.findById(Id);

**if** (hocvien.isPresent()) {

HocVien hv = hocvien.get();

**return** hv;

} **else** {

**throw** **new** EntityNotFoundException();

}

}

}

* Repository: Kế thừa các lớp CrudRepository, PagingAndSortingRepository, JpaRepository để sử dụng các phương thức được định nghĩa sẵn cho việc GET, POST, PUT hay OPTIONS dữ liệu.
* CrudRepository:

Các phương thức CRUD căn bản:

* save(): dùng để lưu 1 hoặc nhiều thực thể (đối tượng).
* findById(): Lấy ra một đối tượng thực thể khi truyền vào primary key của nó.
* findAll(): lấy ra all các đối tượng của thực thể này.
* count(): đếm số lượng bản ghi.
* delete(): xóa đối tượng truyền vào.
* exists(): truyền primary key vào và check xem key đó tồn tại chưa.
* PagingAndSortingRepository

PagingAndSortingRepository kế thừa CrudRepository nên nó có chứa tất carcasc phương thức của CrudRepository và nó có thêm một số phương thức hỗ trợ việc phân trang sau:

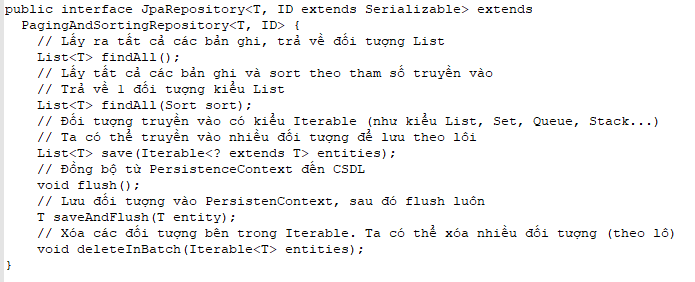
*public interface PagingAndSortingRepository<T, ID extends Serializable>   
extends CrudRepository<T, ID> {  
Iterable<T> findAll(Sort sort);  
Page<T> findAll(Pageable pageable);  
}*

Phương thức findAll(Pageable p) là mấu chốt của việc phân trang. Ta cần truyền vào hàm này 1 đối tượng Pageable, bên trong đối tượng này chỉ ra đc kích thước 1 trang, trang hiện tại muốn lấy dữ liệu, và cơ chế sắp xếp dữ liệu trong trang. Ví dụ ta viết hàm lấy dữ liệu (có phân trang) danh sách HocVien, lấy trang đầu tiên, mỗi trang có 10 kết quả và sắp xếp theo firstName tăng dần:

*Sort sort = new Sort(new Sort.Order(Direction.ASC, "lastName"));  
Pageable pageable = new PageRequest(0, 10, sort);*

* JpaRepository

JpaRepository kế thừa PagingAndSortingRepository nên nó sẽ có chứa tất cả các phương thức của PagingAndSortingRepository và CrudRepository và nó có một số phương thức mở rộng như:



*Một số Annotation trong Repository:*

* “@Repository” để khai báo một class thành Repository
* “@Query” ta sẽ khai báo nó trước các method của interface extends từ JPARepository, CrudRepository , và truyền câu hsql vào bên trong
* “@Param” được sử dụng để khai báo tham số truyền vào câu truy vấn sử dụng “@Query”

Code minh hoạ:

**package** vn.team06.qlhocvienisc.repository;

**import** org.springframework.data.domain.Page;

**import** org.springframework.data.domain.Pageable;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.data.jpa.repository.Query;

**import** org.springframework.data.repository.query.Param;

**import** org.springframework.stereotype.Repository;

**import** vn.team06.qlhocvienisc.entity.HocVien;

@Repository

**public** **interface** HocVienRepository **extends** JpaRepository<HocVien, String>{

@Query(value = "select hv from HocVien hv where hv.MAKHOAHOC=:MAKHOAHOC")

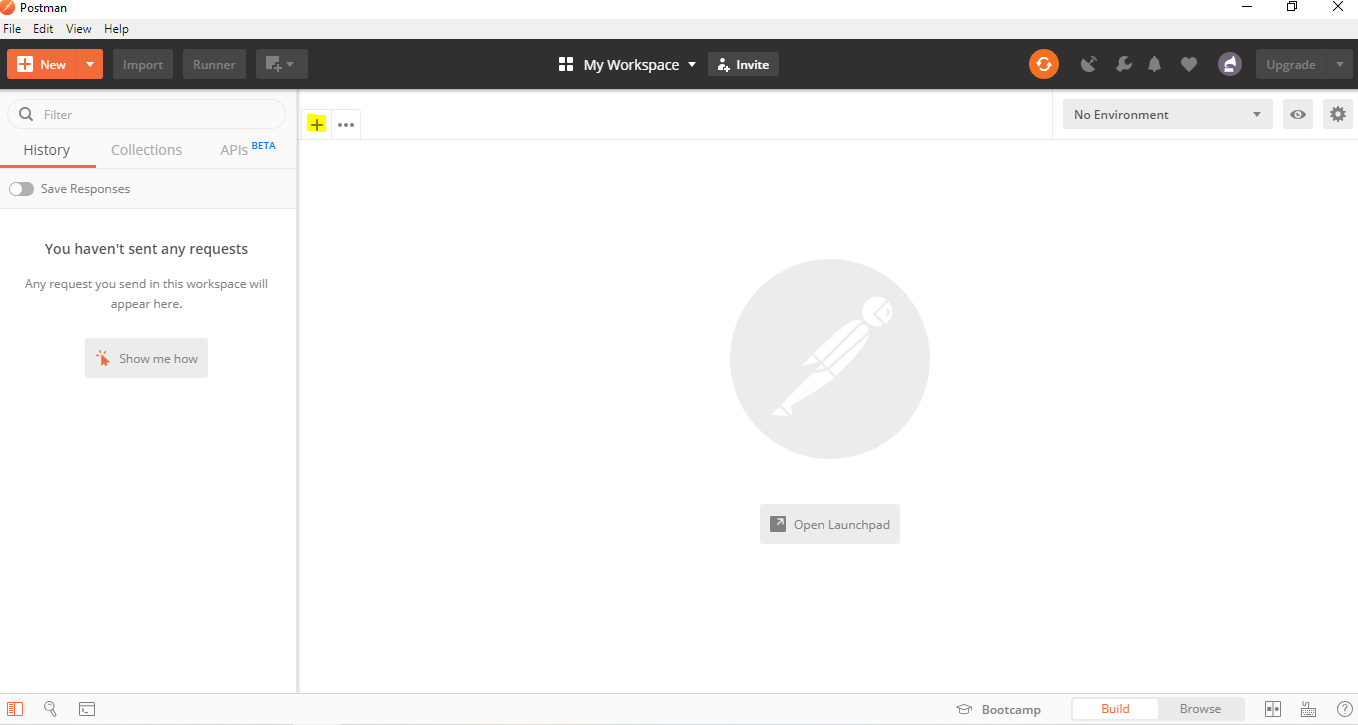
Page<HocVien> findCourse(@Param("MAKHOAHOC") String makhoahoc, Pageable pageable);

}

Các bước tạo cấu trúc server cũng như tạo các API đã hoàn tất bây giờ chúng ta tiến hành chạy server để trả về API bằng cách:

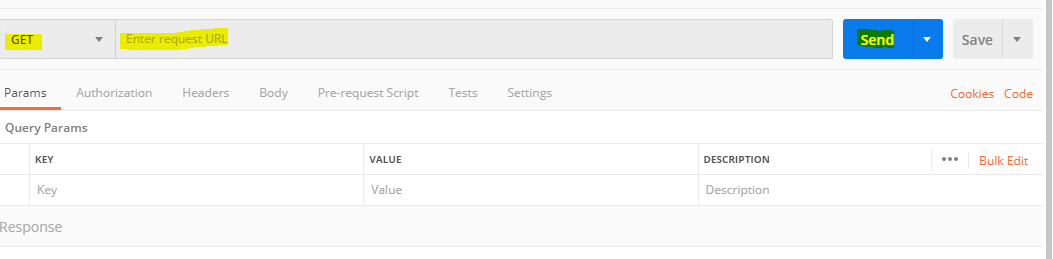
Nhấn chuột phải vào project và chọn Run As -> Spring Boot App

*III.3.3. Sử dụng POSTMAN để Get API kiểm tra dữ liệu trả về*



*Giao diện postman*

Chúng ta click vào dấu cộng mình tô màu vàng như hình trên để bắt đầu

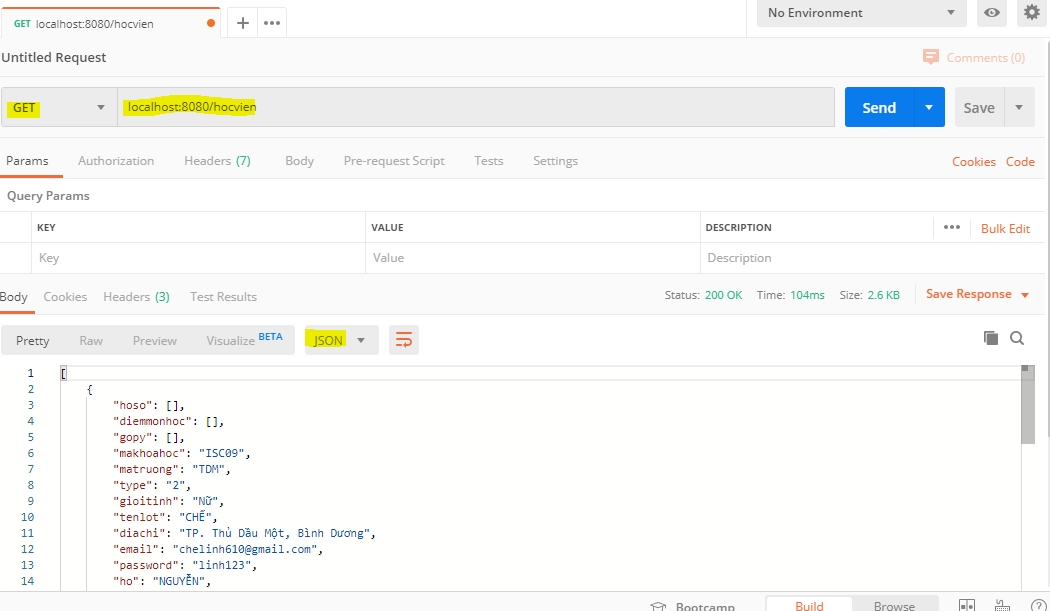


Tiếp theo chúng ta chọn một trong các phương thức GET, POST, PUT... và nhập đường dẫn đến API sau đó nhấn Send

Mặc định Spring Boot chạy đường dẫn mặc định: localhost:8080/{API Controller}/{tham số nếu có}

* *Xem danh sác học viên:*

Khi người dùng gửi request đến API Controller “/hocvien” với phương GET thì Controller sẽ tương tác với phương thức getAll() định nghĩa trong service, trong service phương thức getAll() sẽ gọi Repository với phương thức findAll() định nghĩa sẵn khi kế thừa JpaRepository để lấy ra danh sách tất cả các học viên.



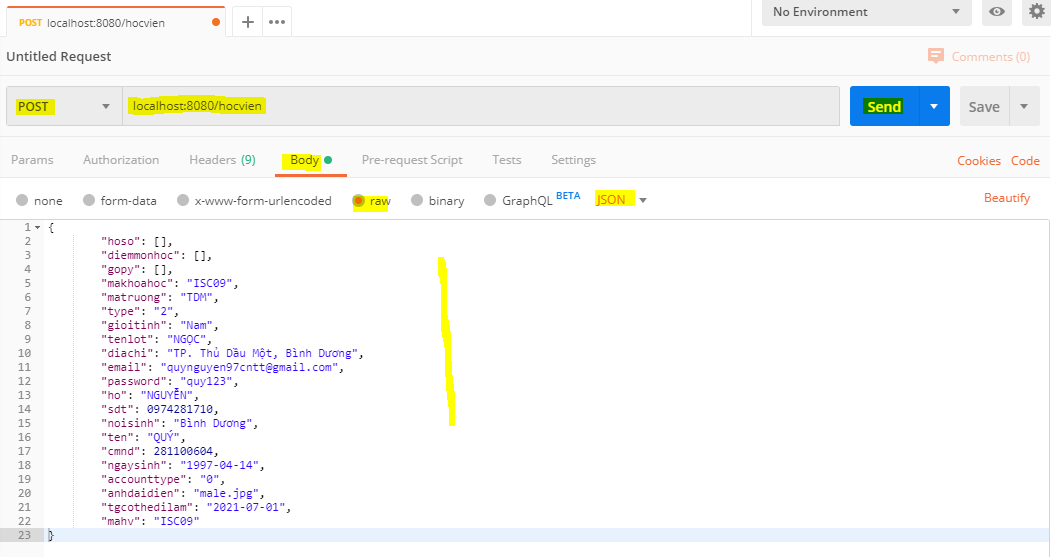
*Danh sách học viên trả về dạng JSON*

* *Thêm học viên*

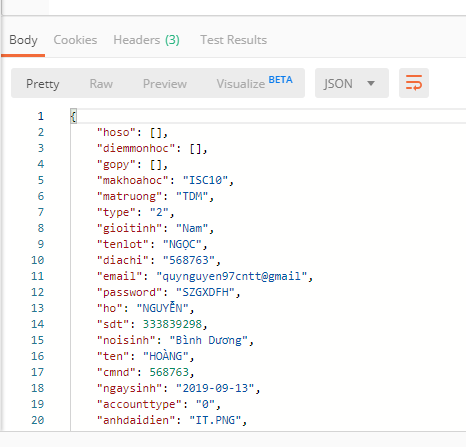
Khi người dùng gửi request đến API Controller “/hocvien” với phương POST thì Controller sẽ tương tác với phương thức createHocvien () định nghĩa trong service, trong service phương thức createHocvien (đối tượng) sẽ gọi Repository với phương thức save(đối tượng) định nghĩa sẵn khi kế thừa JpaRepository để lưu đối tượng đó vào database.

*Trong POSTMAN:*

* Chọn phương thức POST, nhập url: localhost:8080/APIController
* Sau đó chọn thẻ Body và chọn raw chọn định dạng chuỗi JSON và cuối cùng là nhập vào chuỗi JSON và nhấn Send để gửi về server dữ liệu cần thêm.



*Thêm học viên dạng JSON*



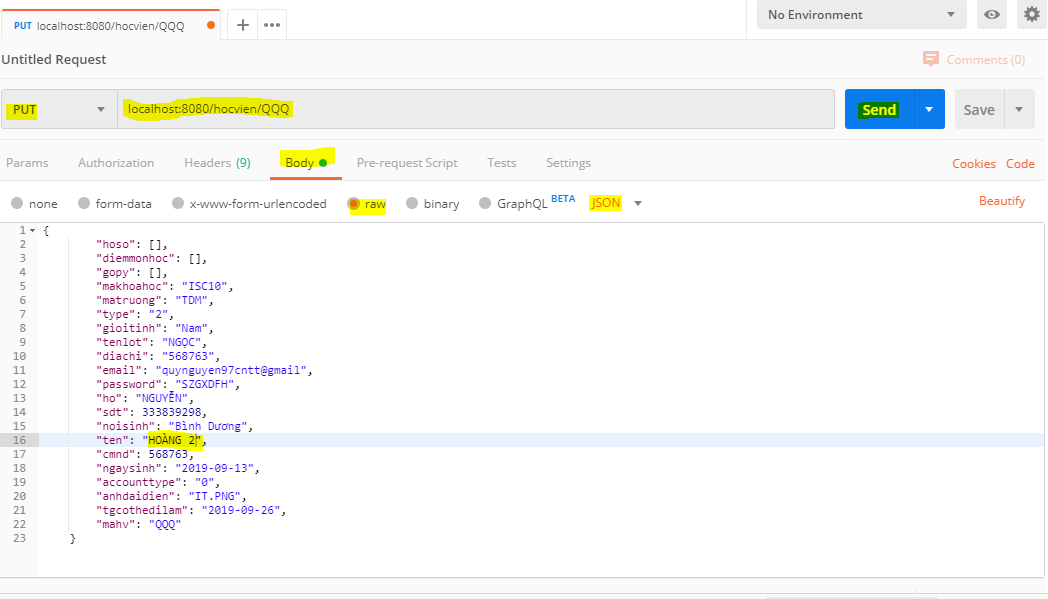
*Kết quả sau khi thêm thành công hiện tại khung xem kết quả phía dưới khung thêm*

* *Cập nhật học viên*

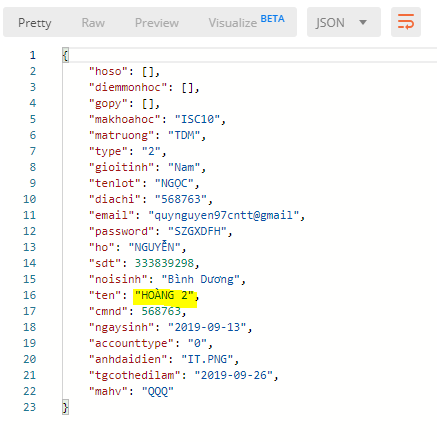
Khi người dùng gửi request đến API Controller “/hocvien/{id}” với phương PUT thì Controller sẽ tương tác với phương thức updateHocvien (đối tượng, id) định nghĩa trong service, trong service phương thức updateHocvien (đối tượng, id) sẽ gọi Repository với các phương thức định nghĩa sẵn khi kế thừa JpaRepository là phương thức findById(id) để tìm đối tượng cần cập nhập và phương thức save(đối tượng) để cập nhật đối tượng trong database.

Trong POSTMAN:

* Chọn phương thức PUT, nhập url: localhost:8080/APIController/{tham số}
* Chọn thẻ Body -> raw -> định dạng dữ liệu JSON-> nhập dữ liệu dạng JSON -> Send để gửi dữ liệu về server



*Cập nhật tên học viên*



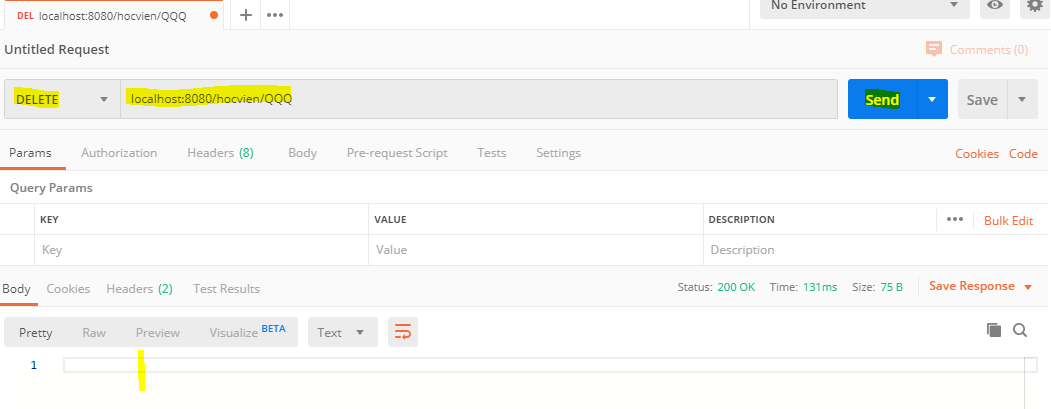
*Kết quả khi cập nhật thành công*

* *Xoá học viên*

Khi người dùng gửi request đến API Controller “/hocvien/{id}” với phương DELETE thì Controller sẽ tương tác với phương thức deleteHocvien(id) định nghĩa trong service, trong service phương thức deleteHocvien(id) sẽ gọi Repository với các phương thức định nghĩa sẵn khi kế thừa JpaRepository là phương thức findById(id) để tìm đối tượng cần xoá và phương thức delete(đối tượng) để xoá đối tượng khỏi database.

Trong POSTMAN:

* Chọn phương thức DELETE, nhập url: localhost:8080/APIController/{tham số}
* Nhấn Send để gửi yêu cầu về server



*Kết quả sau khi xoá thành công*

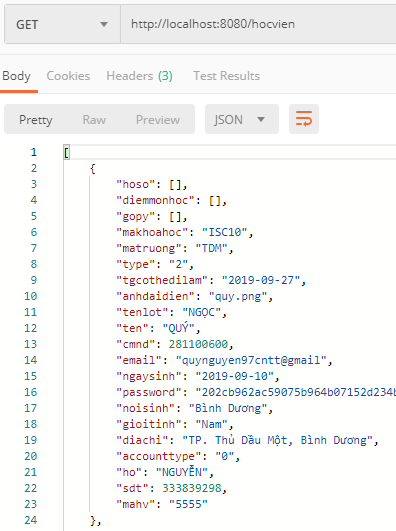
*III.3.4. Các Uri thao tác với dữ liệu*

Web Services chạy localhost Web Server tại [http:/](http://somee.com)/localhost:8080. RESTful cung cấp các uri để thao tác với dữ liệu và trả về dữ liệu kiểu JSON để có thể dễ dàng sử dụng với Javascript. Các Uri được cung cấp bao gồm:

Bảng 2.20. Bảng mô tả uri HocVien

|  |  |  |
| --- | --- | --- |
| Uri | Method hỗ trợ | Mô tả |
| [**http:/**](http://somee.com)**/localhost:8080/hocvien/** | GET | Trả về danh sách tất cả học viên |
| [**http:/**](http://somee.com)**/localhost: 8080/ hocvien /** | POST | Thêm học viên |
| [**http:/**](http://somee.com)**/localhost: 8080/ hocvien /{id}** | PUT, DELETE | Sửa và xóa học viên |
| [**http:/**](http://somee.com)**/localhost: 8080/ timkiemhocvien/{id}** | GET | Tìm kiếm học viên |

Hình ảnh sử dụng postman để test trên localhost:

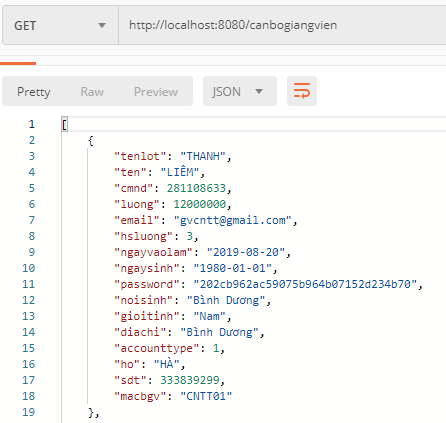


**Hình 3.7. Dữ liệu khi GET Sinh Viên từ server**

Bảng 2.21. Bảng mô tả uri GiangVien

|  |  |  |
| --- | --- | --- |
| Uri | Method hỗ trợ | Mô tả |
| [**http:/**](http://somee.com)**/localhost:8080/giangvien/** | GET | Trả về danh sách tất cả giảng viên |
| [**http:/**](http://somee.com)**/localhost: 8080/giangvien/** | POST | Thêm giảng viên |
| [**http:/**](http://somee.com)**/localhost:8080/giangvien/{id}** | PUT, DELETE | Sửa và xóa giảng viên |
| [**http:/**](http://somee.com)**/localhost: 8080/ timkiemgiangvien/{id}** | GET | Tìm kiếm giảng viên |

Hình ảnh sử dụng postman để test trên localhost:

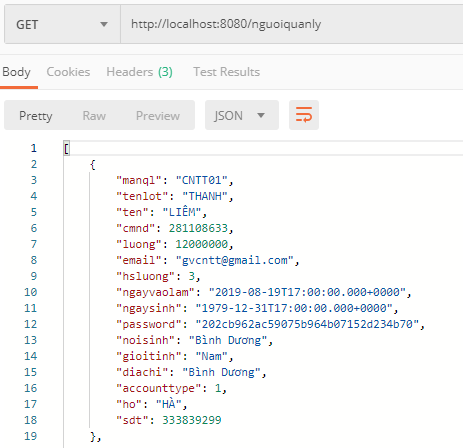


**Hình 3.8. Dữ liệu khi GET Giảng Viên từ server**

Bảng 2.22. Bảng mô tả uri NguoiQuanLi

|  |  |  |
| --- | --- | --- |
| Uri | Method hỗ trợ | Mô tả |
| [**http:/**](http://somee.com)**/localhost:8080/nguoiquanli/** | GET | Trả về danh sách tất cả giảng viên |
| [**http:/**](http://somee.com)**/localhost: 8080/ nguoiquanli/** | POST | Thêm người quản lí |
| [**http:/**](http://somee.com)**/localhost:8080/nguoiquanli/{id}** | PUT, DELETE | Sửa và xóa người quản lí |
| [**http:/**](http://somee.com)**/localhost: 8080/ timkiemnguoiquanli/{id}** | GET | Tìm kiếm người quản lí |

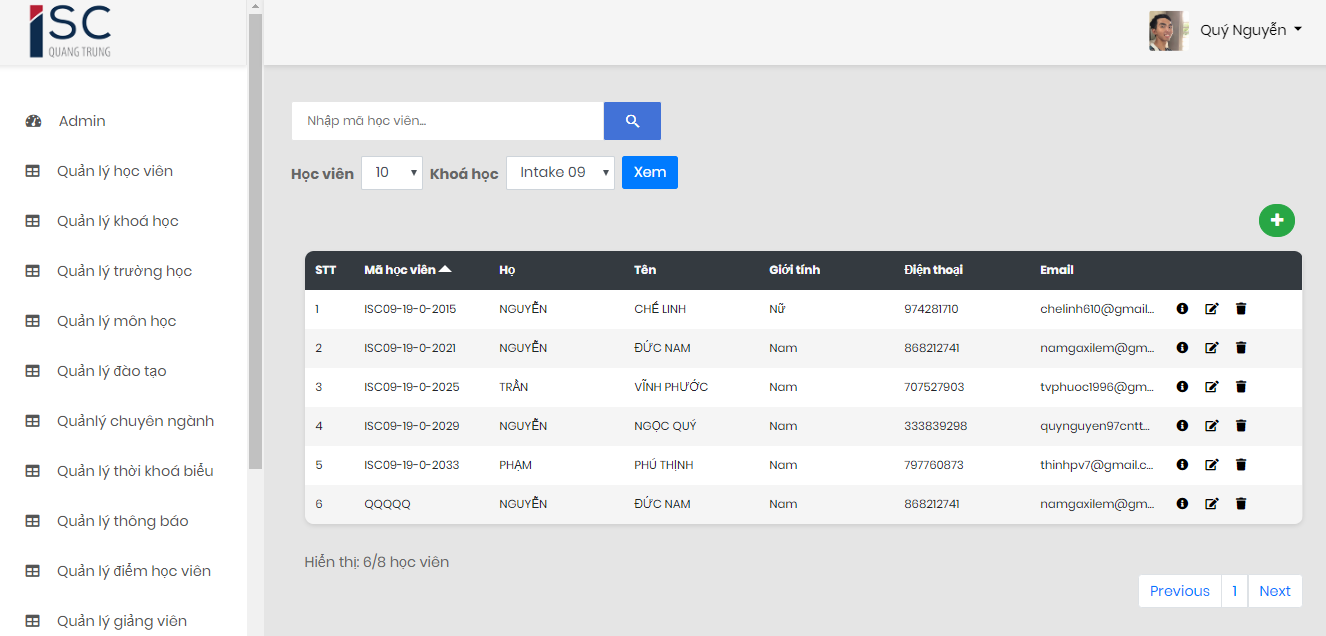
Hình ảnh sử dụng postman để test trên localhost:



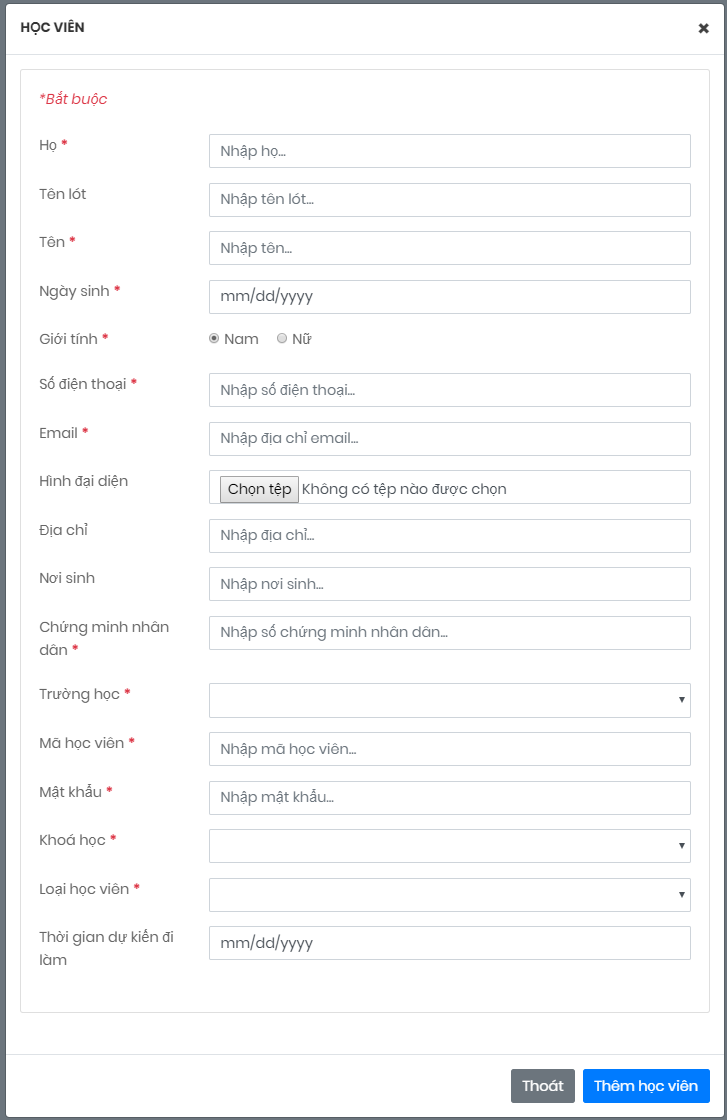
Hình 3.2. Dữ liệu khi GET Người Quản Lí từ server

III.4. Giao diện hệ thống reponsive sử dựng Bootstrap, AngularJS và jQuery

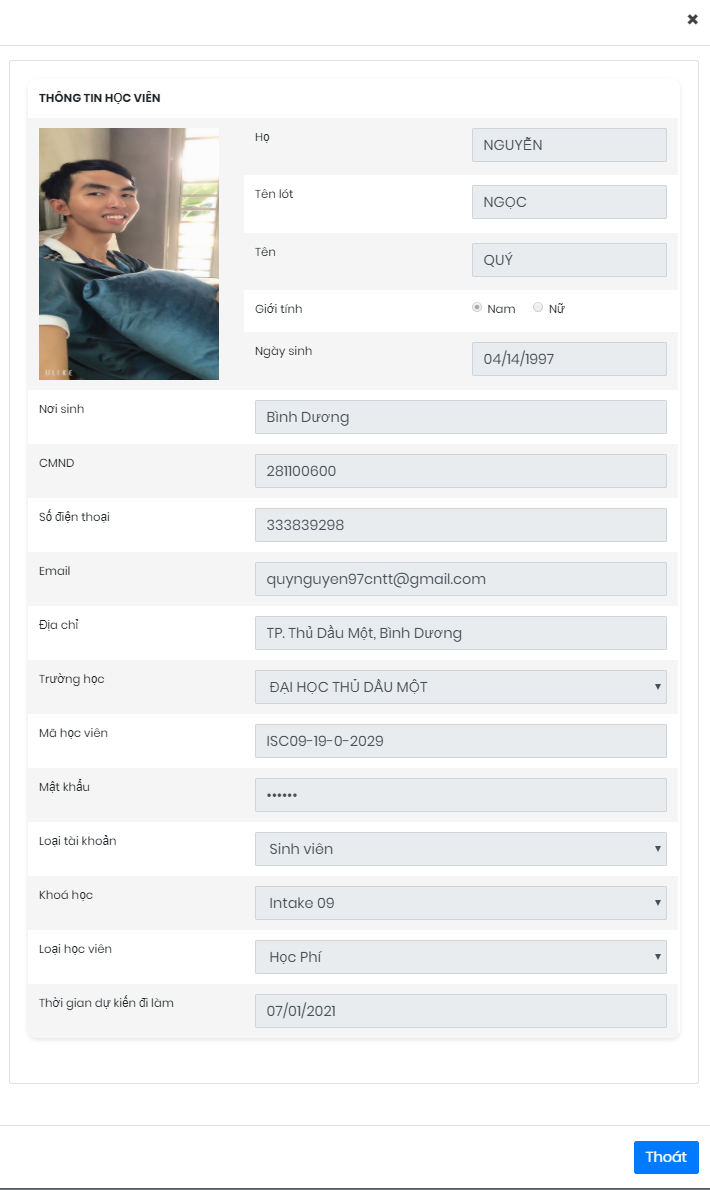
*III.4.1. Giao diện quản lý học viên*



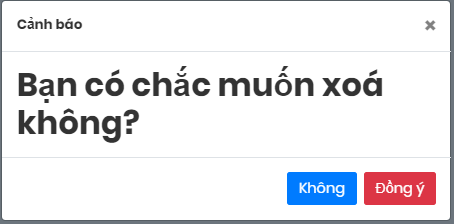
Hình 3.3. Giao diện quản lý học viên



Hình 3.4. Giao diện trang thêm học viên

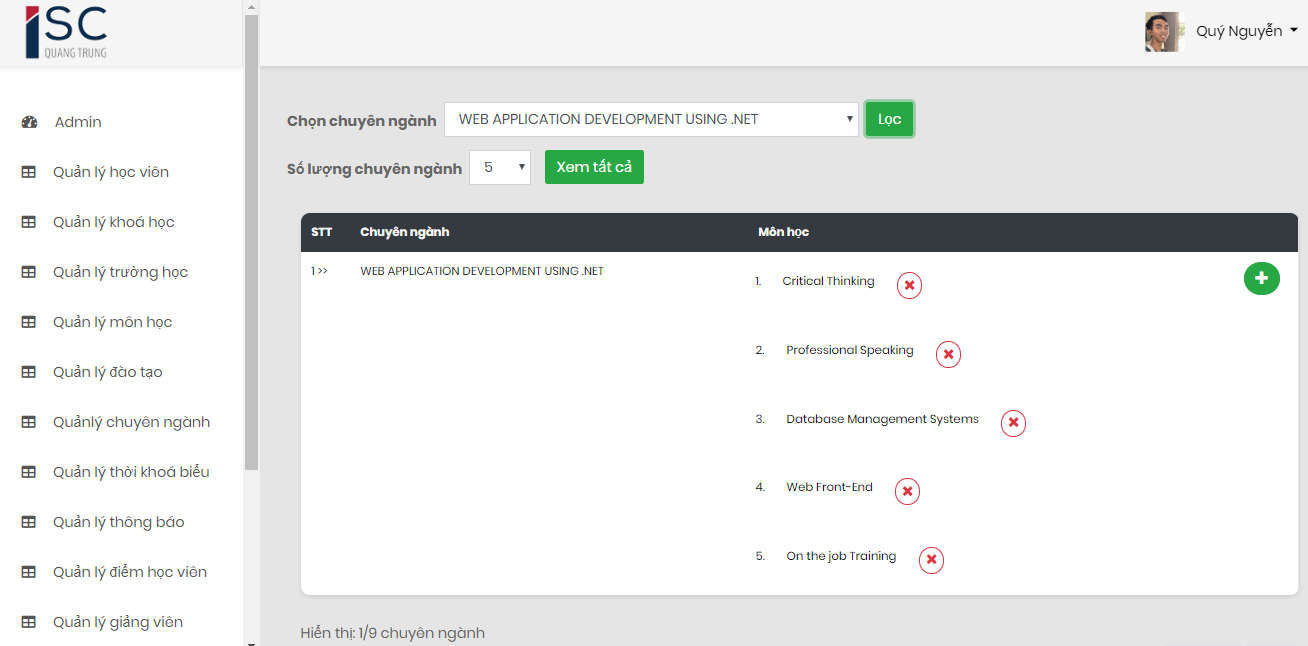


Hình 3.5. Giao diện trang thông tin học viên

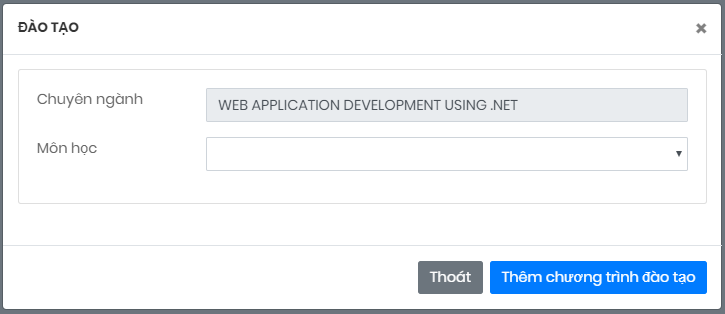


Hình 3.6. Giao diện xoá học viên

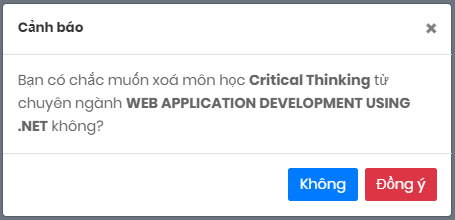
*III.4.2. Giao diện quản lý chương trình học*



Hình 3.7. Giao diện quản lý chương trình học

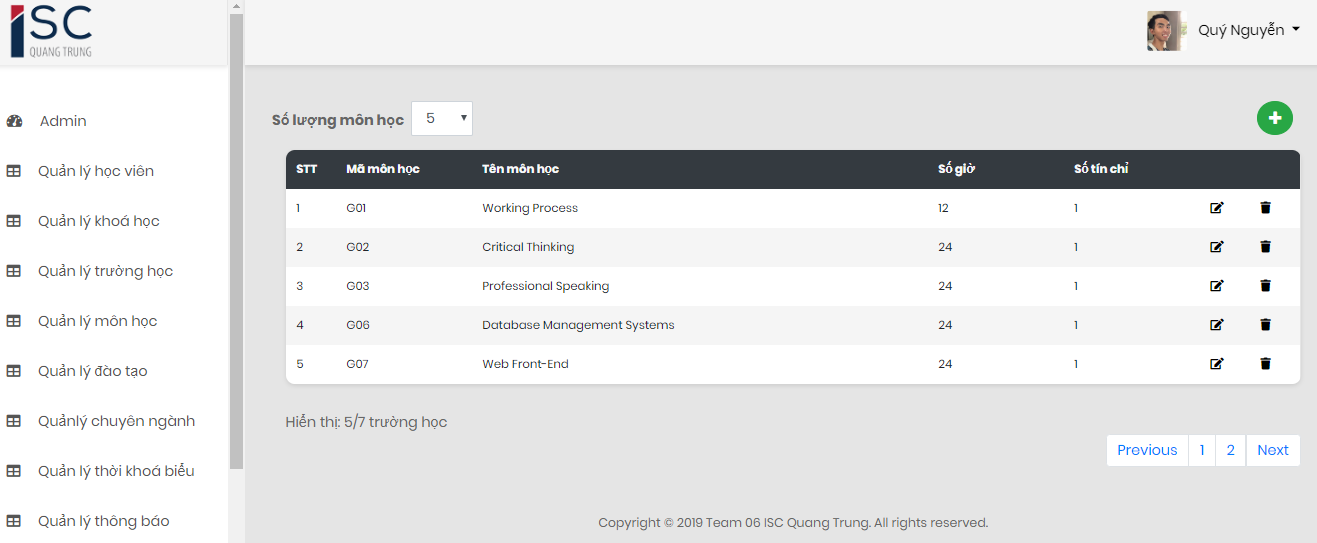


Hình 3.8. Giao diện thêm môn học vào chuyên ngành tương ứng



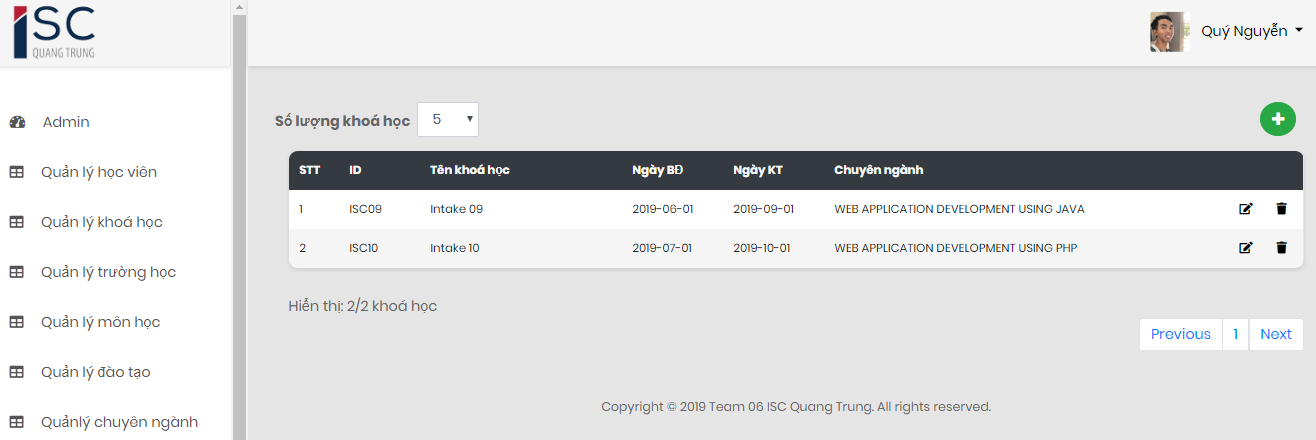
Hình 3.9. Giao diện xoá môn học ra khỏi chuyên ngành tương ứng

*III.4.3. Giao diện quản lý môn học*



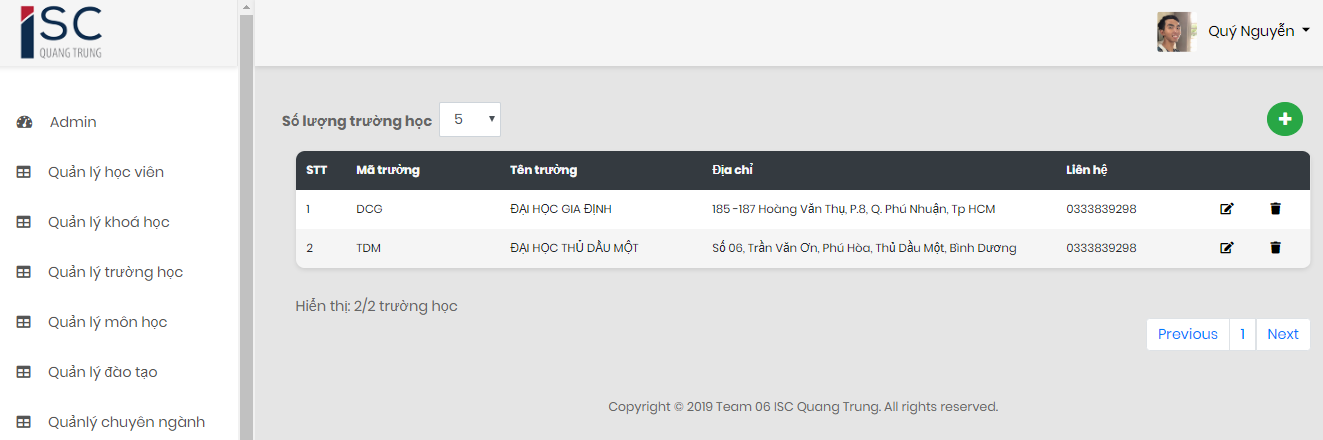
Hình 3.10. Giao diện quản lý môn học

*III.4.4. Giao diện quản lý khoá học*



Hình 3.11. Giao diện quản lý khoá học

*III.4.5. Giao diện quản lý trường học*



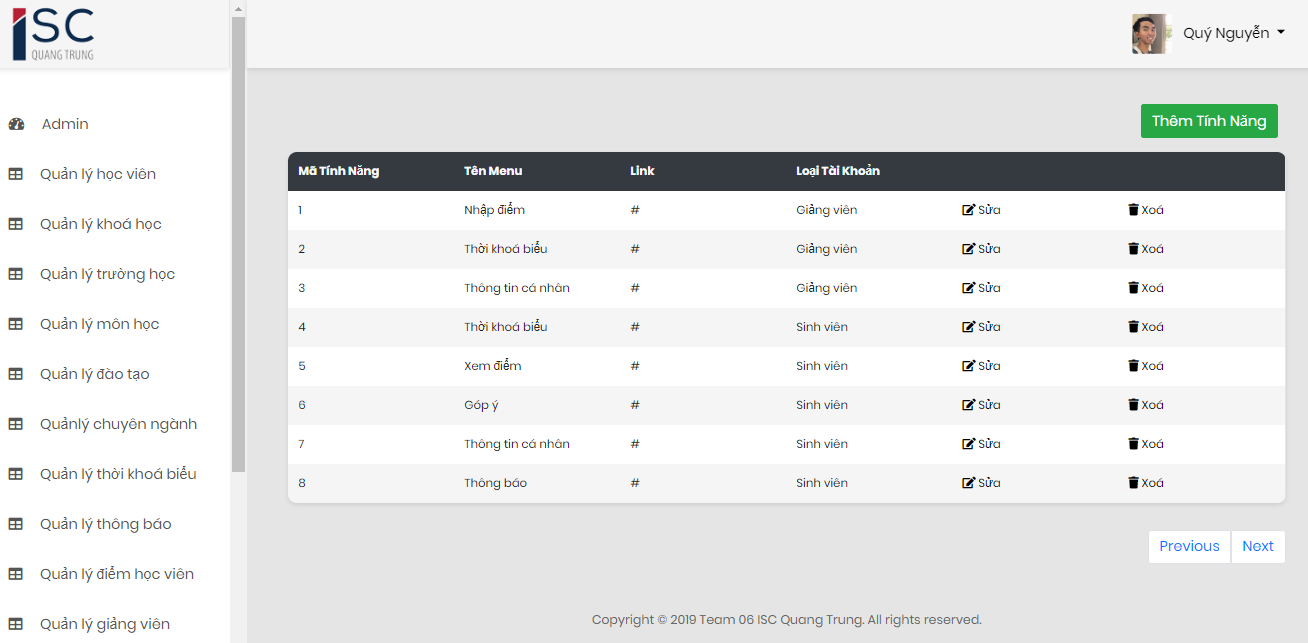
Hình 3.12. Giao diện quản lý trường học

*III.4.6. Giao diện quản lý chuyên ngành*



Hình 3.13. Giao diện quản lý chuyên ngành

*III.4.7. Giao diện quản lý tính năng*



Hình 3.14. Giao diện quản lý tính năng

*III.4.8. Giao diện đăng nhập admin*



Hình 3.15. Giao diện đăng nhập admin

# KẾT LUẬN

## 1. Kết luận

Sau thời gian thực tập tại ISC – Quang Trung nhờ sự giúp đỡ tận tình của thầy Võ Quốc Lương và thầy Hà Thanh Liêm cùng các anh chị trong công ty, đã tạo cho chúng em có được điều kiện tham quan môi trường làm việc thực tế, ứng dụng được những kiến thức đã học ở trường cũng như học hỏi thêm kiến thức mới, công nghệ mới, nghiên cứu, tìm hiểu và hoàn thành báo cáo với đề tài: “Xây dựng phần mềm quản lý học viên tại trung tâm đào tạo kỹ sư công nghệ thông tin ISC Quang Trung”.

Tuy thời gian thực tập không dài nhưng đã cho chúng em thấy được tầm quan trọng của một người kỹ sư công nghệ thông tin trong môi trường làm việc thực tế. Qua đó chúng em học hỏi được nhiều điều quý giá từ đơn vị thực tập cũng như từ những anh chị trong công ty đã hỗ trợ giúp đỡ em về nhiều mặt như:

* **Về kiến thức**: Nắm được quy trình để tạo nên một phần mềm có tính thực tế và ứng dụng hiệu quả. Nắm được những phương pháp để đem những công nghệ mới vào sản phẩm, đồng thời đưa ra hướng đi đúng đắn để tiếp cận đề tài.
* **Về kỹ năng**: Vận dụng được kỹ năng giao tiếp không còn nhút nhát, e dè khi giao tiếp, tiếp xúc với mọi người, phát huy được khả năng phát biểu ý kiến; nắm được kỹ năng thuyết trình sao cho đầy đủ và người nghe không cảm thấy nhàm chán cũng như có thể hiểu được những nội dung trong bài thuyết trình.
* **Về thái độ**: Tuân thủ theo đúng quy định khi đến thực tập tại công ty, tôn trọng, lễ phép cũng như hòa đồng với tất cả anh chị trong công ty.

Vì thời gian hạn hẹp và chưa có nhiều kinh nghiệm thực tế. Vì vậy, đề tài còn nhiều thiếu sót, mong các thầy cô giúp đỡ và thông cảm.

Một lần nữa em xin chân thành cảm ơn các Ban lãnh đạo, Ban giám đốc công ty ISC Quang Trung, các thầy cô tại trường Đại Học Thủ Dầu Một cùng các anh chị trong công ty rất nhiều đã giúp đỡ và tạo cơ hội và điều kiện để thực tập tại đây trong thời gian qua.

## 2. Kiến nghị

Sau khi kết thúc tất cả các môn học tại trường thì tất cả các sinh viên sẽ có khoảng thời gian là 2 tháng để thực tập, tiếp cận với môi trường làm việc thực tế. Tuy nhiên, với khoảng thời gian ngắn như thế và đặc biệt lại nhằm vào thời gian lễ Tết, sinh viên vẫn chưa tích lũy được nhiều kinh nghiệm và kỹ năng làm việc để có thể chuẩn bị cho quá trình ra trường và làm việc sau này, vì vậy chúng em mong bên phía nhà trường có thể xem xét lại thời điểm thực tập cho sinh viên cũng như tạo thêm nhiều điều kiện và cơ hội để sinh viên có thể có thêm nhiều thời gian học hỏi, tích lũy được nhiều kiến thức, kinh nghiệm làm việc.

Về những kiến thức được học ở trường theo chúng em thì chưa thực sự đủ để va chạm vào công việc thực tế mà trước hết là chưa đủ tự tin để trở thành một thực tập sinh. Vì vậy, mong nhà trường có thể tạo thêm nhiều khóa học chuyên về một lĩnh vực nào đó để sinh viên có thể lựa chọn. Song song đó, chúng em cũng mong nhà trường có thể lược bớt lượng thời gian học lý thuyết, đồng thời tăng lượng thời gian thực hành. Hoặc có thể vừa học lý thuyết song song thực hành, như vậy sẽ có hiệu quả và không gây nhàm chán khi học.

Qua quá trình thực tập, chúng em đã có cơ hội trải nghiệm được công việc thực tế, hiểu được rõ hơn công việc của kỹ sư lập trình cũng như hình dung được những công việc mình có thể làm sau khi ra trường.

# TÀI LIỆU THAM KHẢO

1. https://www.ibm.com/developerworks/vn/library/wa-html5fundamentals. Truy cập ngày: 20/06/2019
2. https://en.wikipedia.org/wiki/HTML5. Truy cập ngày: 23/06/2019
3. https://www.w3schools.com. Truy cập ngày: 27/06/2019
4. http://vietjack.com. Truy cập ngày: 05/07/2019
5. http://tutorialspoint.com. Truy cập ngày: 10/07/2019
6. https://scotch.io/tutorials. Truy cập ngày: 23/07/2019
7. https://angularjs.org. Truy cập ngày: 28/07/2019
8. https://viblo.asia. Truy cập ngày: 01/08/2019
9. https://topdev.vn/blog/restful-api-la-gi. Truy cập ngày: 23/07/2019
10. https://itviec.com/blog/agile-la-gi-scrum-la-gi. Truy cập ngày: 24/07/2019
11. https://stackjava.com. Truy cập ngày: 12/08/2019
12. https://kenhlaptrinh.net/hibernate-bai-7-query-trong-hibernate-hibernate-query-language-hql. Truy cập ngày: 14/08/2019
13. https://o7planning.org. Truy cập ngày: 15/08/2019