**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

 **KHOA CÔNG NGHỆ PHẦN MỀM**

**ĐỒ ÁN MÔN**

**CHUYÊN ĐỀ J2EE**

**Đề tài:**

**WEBSITE ĐĂNG KÝ HỌC TOEIC**

Giảng viên hướng dẫn: ThS. Nguyễn Trác Thức

Giảng viên thực hành : ThS. Thái Thụy Hàn Uyển

Lớp: SE325.H22

Sinh viên thực hiện: Nguyễn Văn Quang 13520675

Bùi Đình Lộc Thọ 13520844

Nguyễn Thanh Hải 13520

***TP. Hồ Chí Minh, ngày 03 tháng 06 năm 2017***

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**ĐỒ ÁN MÔN**

**CHUYÊN ĐỀ J2EE**

**Đề tài:**

**WEBSITE ĐĂNG KÝ HỌC TOEIC**

Giảng viên hướng dẫn: ThS. Nguyễn Trác Thức

Giảng viên thực hành : ThS. Thái Thụy Hàn Uyển

Lớp: SE325.H22

Sinh viên thực hiện: Nguyễn Văn Quang 13520675

Bùi Đình Lộc Thọ 13520844

Nguyễn Thanh Hải 13520

***TP. Hồ Chí Minh, ngày 03 tháng 06 năm 2017***

**Lời Cảm Ơn**

Đầu tiên, nhóm chúng em xin gởi lời cảm ơn chân thành đến tập thể quý Thầy Cô Trường Đại học Công nghệ thông tin – Đại học Quốc gia TP.HCM và quý Thầy Cô khoa Công Nghệ Phần Mềm đã giúp cho nhóm chúng em có những kiến thức cơ bản làm nền tảng để thực hiện đề tài này.

Đặc biệt, nhóm chúng em xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới thầy Nguyễn Trác Thức, cô Thái Thụy Hàn Uyển. Đã giúp nhóm chúng em hoàn thành tốt báo cáo môn học của mình.

Trong thời gian một học kỳ thực hiện đề tài, nhóm chúng em đã vận dụng những kiến thức nền tảng đã tích lũy đồng thời kết hợp với việc học hỏi và nghiên cứu những kiến thức mới. Từ đó, nhóm chúng em vận dụng tối đa những gì đã thu thập được để hoàn thành một báo cáo đồ án tốt nhất. Tuy nhiên, trong quá trình thực hiện, nhóm chúng em không tránh khỏi những thiếu sót. Chính vì vậy, nhóm chúng em rất mong nhận được những sự góp ý từ phía các Thầy Cô nhằm hoàn thiện những kiến thức mà nhóm chúng em đã học tập và là hành trang để nhóm chúng em thực hiện tiếp các đề tài khác trong tương lai.

Nhóm em xin chân thành cảm ơn quý Thầy Cô!

NHẬN XÉT CỦA GIẢNG VIÊN

………………………………………………………………………………………………....

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………………………………

MỤC LỤC

[MỤC LỤC 5](#_Toc484131021)

[CHƯƠNG 1. GIỚI THIỆU 7](#_Toc484131022)

[1.1 Giới thiệu về Java 7](#_Toc484131023)

[1.1.1 Lịch sử phát triển 7](#_Toc484131024)

[1.1.2 Khả năng của ngôn ngữ Java 7](#_Toc484131025)

[1.1.3 Những đặc điểm của ngôn ngữ Java 8](#_Toc484131026)

[1.1.4 Máy ảo Java (JMV – Java Virtual Machine) 8](#_Toc484131027)

[1.2 Giới thiệu về Java EE 9](#_Toc484131028)

[1.2.1 Java EE Là gì 9](#_Toc484131029)

[1.2.2 Thành phần cơ bản 9](#_Toc484131030)

[1.2.3 Cấu trúc Java EE 9](#_Toc484131031)

[1.2.4 Web và ứng dụng Server 10](#_Toc484131032)

[1.2.5 Các Container có trong Java EE 11](#_Toc484131033)

[1.3 Project TOEIC Class 12](#_Toc484131034)

[Chương 2. CÔNG NGHỆ 13](#_Toc484131035)

[2.1 HTML 13](#_Toc484131036)

[2.1.1 HTML là gì 13](#_Toc484131037)

[2.1.2 HTML được xử lý như thế nào 13](#_Toc484131038)

[2.1.3 Cấu trúc của một đoạn HTML 13](#_Toc484131039)

[2.1.4 Vai trò của HTML 14](#_Toc484131040)

[2.2 CSS 14](#_Toc484131041)

[2.2.1 Tổng quan về CSS 14](#_Toc484131042)

[2.2.2 Công dụng CSS 15](#_Toc484131043)

[2.2.3 Cách sử dụng CSS 15](#_Toc484131044)

[2.3 Javascript 17](#_Toc484131045)

[2.3.1 Tổng quan 17](#_Toc484131046)

[2.3.2 Javascript là gì 18](#_Toc484131047)

[2.3.3 Hiệu ứng và quy tắc JavaScript 19](#_Toc484131048)

[2.3.4 Nhúng JavaScript vào trong trang Web 20](#_Toc484131049)

[2.4 MySQL 22](#_Toc484131050)

[2.4.1 Tổng quan 22](#_Toc484131051)

[2.4.2 Thuật ngữ RDBMS 22](#_Toc484131052)

[2.4.3 MySQL Database 23](#_Toc484131053)

[2.5 Ajax 24](#_Toc484131054)

[2.5.1 Tổng quan 24](#_Toc484131055)

[2.5.2 Ưu điểm 24](#_Toc484131056)

[2.5.3 Nhược điểm 25](#_Toc484131057)

[Chương 3. THIẾT KẾ - CÀI ĐẶT 26](#_Toc484131058)

[3.1 Kiến trúc hệ thống 26](#_Toc484131059)

[3.1.1 Mô hình MVC 26](#_Toc484131060)

[3.1.2 Mô hình Client – Server 31](#_Toc484131061)

[3.2 Sơ đồ Use-case 36](#_Toc484131062)

[3.2.1. Danh sách các actor 36](#_Toc484131063)

[3.2.2. Danh sách các usecase 37](#_Toc484131064)

[3.2.3. Đặc tả usecase 37](#_Toc484131065)

[3.3 Sơ đồ Sequence 45](#_Toc484131066)

[3.3.1 Đăng ký 45](#_Toc484131067)

[3.3.2 Gửi thông tin liên hệ 45](#_Toc484131068)

[3.3.3 Đăng ký khoá học 46](#_Toc484131069)

[3.3.4 Xem khoá học 47](#_Toc484131070)

[3.3.5 Đăng nhập 48](#_Toc484131071)

[3.3.6 Đăng xuất 48](#_Toc484131072)

[3.3.7 Quản lý thông tin liên hệ 49](#_Toc484131073)

[3.3.8 Quản lý học viên 49](#_Toc484131074)

[3.3.9 Quản lý khoá học 50](#_Toc484131075)

[3.3.10 Thống kê 51](#_Toc484131076)

[3.4 Giao diện ứng dụng 52](#_Toc484131077)

[3.4.1 Trang chủ 53](#_Toc484131078)

[3.4.2 Đăng kí 54](#_Toc484131079)

[3.4.3 Liên hệ 55](#_Toc484131080)

[3.4.4 Đăng nhập 56](#_Toc484131081)

[3.4.5 Xem chi tiết khoá học 57](#_Toc484131082)

[3.4.6 Xem khóa học đã đăng kí 58](#_Toc484131083)

[3.4.6 Trang Admin 59](#_Toc484131084)

[3.4.7 Trang Admin – Thống kê 60](#_Toc484131085)

[3.4.8 Trang Admin – Khóa học 61](#_Toc484131086)

[3.4.9 Trang Admin – Học viên 62](#_Toc484131087)

[3.4.10 Trang Admin – Liên hệ 63](#_Toc484131088)

# 

# CHƯƠNG 1. GIỚI THIỆU

## 1.1 Giới thiệu về Java

### 1.1.1 Lịch sử phát triển

Năm 1990, Sun MicroSystems thực hiện dự án Green nhằm phát triển phần mềm trong các thiết bị dân dụng. James Gosling, chuyên gia lập trình đã tạo ra một ngôn ngữ lập trình mới có tên là Oak. Ngôn ngữ này có cú pháp gần giống như C++ nhưng bỏ qua các tính năng nguy hiểm của C++ như truy cập trực tiếp tài nguyên hệ thống, con trỏ, định nghĩa chồng các tác tử…

Khi ngôn ngữ Oak trưởng thành, WWW cũng đang vào thời kỳ phát triển mạnh mẽ, Sun cho rằng đây là một ngôn ngữ thích hợp cho Internet. Năm 1995, Oak đổi tên thành Java và sau đó đến 1996  Java đã được xem như một chuẩn công nghiệp cho Internet.

### 1.1.2 Khả năng của ngôn ngữ Java

Là một ngôn ngữ bậc cao giống như C, C++, Perl, SmallTalk,… cho nên có thể tạo những ứng dụng để giải quyết các vấn đề về số, xử lý văn bản, tạo ra trò chơi và nhiều thứ khác. Có các môi trường lập trình đồ hoạ như Visual Java, Sysmantec Café, Jbuilder, Jcreator,…. Có khả năng truy cập dữ liệu từ xa thông qua cầu nối JDBC (Java Database Connectivity) Hỗ trợ các lớp hữu ích, tiện lợi trong lập trình các ứng dụng mạng (Socket) cũng như truy xuất Web.

Hỗ trợ lập trình phân tán (Remote Method Invocation) cho phép một ứng dụng có thể được xử lý phân tán trên các máy tính khác nhau. Và luôn được bổ sung các tính năng cao cấp khác trong các phiên bản sau.

### 1.1.3 Những đặc điểm của ngôn ngữ Java

Ngôn ngữ hoàn toàn hướng đối tượng. Ngôn ngữ đa nền cho phép một chương trình có thể thực thi trên các hệ điều hành khác nhau (MS Windows, UNIX, Linux) mà không phải biên dịch lại chương trình. Phương châm của java là **"Viết một lần ,  Chạy trên nhiều nền"**  (Write Once, Run Anywhere).

Ngôn ngữ đa luồng, cho phép trong một chương trình có thể có nhiều luồng điều khiển được thực thi song song nhau, rất hữu ích cho các xử lý song song.

Ngôn ngữ phân tán, cho phép các đối tượng của một ứng dụng được phân bố và thực thi trên các máy tính khác nhau.

Ngôn ngữ động, cho phép mã lệnh của một chương trình được tải từ một máy tính về máy của người yêu cầu thực thi chương trình.

Ngôn ngữ an toàn, tất cả các thao tác truy xuất vào các thiết bị vào ra đều thực hiện trên máy ảo nhờ đó hạn chế các thao tác nguy hiểm cho máy tính thật.

Ngôn ngữ đơn giản, dễ học, kiến trúc chương trình đơn giản, trong sáng.

### 1.1.4 Máy ảo Java (JMV – Java Virtual Machine)

Để đảm bảo tính đa nền, Java sử dụng cơ chế **Máy ảo của Java**. ByteCode đó là ngôn ngữ máy của Máy ảo Java tương tự như các lệnh nhị phân của các máy tính thực. Một chương trình sau khi được viết bằng ngôn ngữ Java (có phần mở rộng là .java) phải được biên dịch thành tập tin thực thi được trên máy ảo Java (có phần mở rộng là .class).  Tập tin thực thi này chứa các chỉ thị dưới dạng mã Bytecode mà máy ảo Java hiểu được phải làm gì.

Khi thực hiện một chương trình, máy ảo Java lần lượt thông dịch các chỉ thị dưới dạng Bytecode thành các chỉ thị dạng nhị phân của máy tính thực và thực thi thực sự chúng trên máy tính thực.

Máy ảo thực tế đó là một chương trình thông dịch. Vì thế các hệ điều hành khác nhau sẽ có các máy ảo khác nhau. Để thực thi một ứng dụng của Java trên một hệ điều hành cụ thể, cần phải cài đặt máy ảo tương ứng cho hệ điều hành đó.

## 1.2 Giới thiệu về Java EE

### 1.2.1 Java EE Là gì

Java EE (Java Enterprise Editor) là một nền tảng có chứa một tập hợp các công nghệ khác nhau để tạo ra các ứng dụng mạnh mẽ và di động. Java EE được xây dựng trên nền tảng Java SE cung cấp các API (giao diện lập trình ứng dụng) để phát triển và chạy các ứng dụng phía máy chủ.

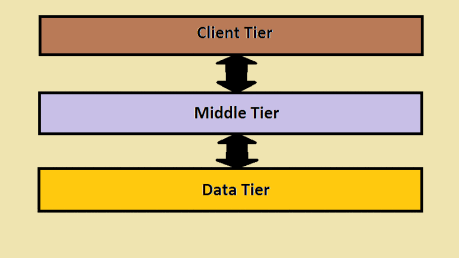
### 1.2.2 Thành phần cơ bản

**Enterprise Java Beans (EJB)**: một thành phần kiến trúc của các ứng dụng server được quản lý, sử dụng để bao gói (encapsulate) các business logic của các ứng dụng. Công nghệ EJB cho phép phát triển nhanh chóng và đơn giản hóa các ứng dụng phân tán, các giao dịch an toàn và di động dựa trên công nghệ Java.

**Java Persistence API (JPA)**: một framework cho phép nhà phát triển quản lý dữ liệu bằng cách sử dụng ánh xạ đối tượng quan hệ (Object Relational Mapping – ORM) trong các ứng dụng được xây dựng trên nền tảng Java.

### 1.2.3 Cấu trúc Java EE

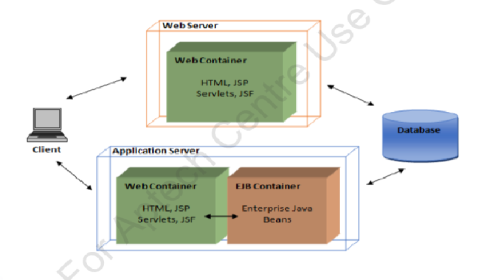
JavaEE làm việc dựa trên kiến trúc ba tầng (3 Tier)



**Nguyên lý hoạt động:**

* Client Tier : Gồm các giao diện người dùng sử dụng để truy cập ứng dụng.
* Middle Tier : Kiểm soát dữ liệu của các ứng dụng thông qua Beans và các tập tin lớp khác của ứng dụng.
* Data Tier : Chứa các dữ liệu cơ sở.

### 1.2.4 Web và ứng dụng Server

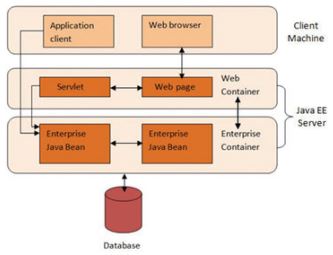


**Nguyên lý hoạt động:**

* Máy chủ Web chấp nhận yêu cầu từ khách hàng thông qua các trang web và  
  đáp ứng những yêu cầu với các trang web thích hợp.
* Các thông tin liên lạc giữa máy khách và máy chủ Web diễn ra  
  thông qua Hyper Text Transfer Protocol (HTTP).
* Các container chứa các thành phần khác nhau cùng cung cấp dịch vụ cho các ứng dụng. Qua đó giảm phức tạp trong khi phát triển các tính năng và cho phép lập trình viên tập trung thực hiện logic.

### 1.2.5 Các Container có trong Java EE

Containers và thành phần cung cấp cho Java EE các dịch vụ nhằm đơn giản hóa lập trình.



Sơ đồ cơ bản các Container làm việc trong Java EE

* **Web Container**: Quản lý sự thực thi của các thành phần web, và container của chúng chạy trong máy chủ J2EE (J2EE server).
* **Enterprise Java Beans Container (EJB)**: Quản lý sự thực thi của các thành phần EJB, các thành phần EJB và container của chúng chạy trong J2EE server.
* **Applet Container**: Quản lý sự thực thi của applets (ứng dụng ký sinh).
* **Applicaton Client Container**: Quản lý sự thực thi các thành phần của ứng dụng client.

## 1.3 Project TOEIC Class

# Chương 2. CÔNG NGHỆ

## 2.1 HTML

### 2.1.1 HTML là gì

HTML hay còn gọi là ngôn ngữ đánh dấu siêu văn bản được sử dụng để tạo nên website, trên một website sẽ có nhiều trang và mỗi trang sẽ được đánh dấu là 1 tập tin HTML.

Một tài liệu hay tập tin HTML được tạo nên bởi các phần tử HTML được quy định bằng các cặp thẻ tag, các cặp thẻ này nằm trong cặp dấu ngoặc nhọn (ví dụ: <html>) và thường khai báo thành một cặp gồm thẻ mở và thẻ đóng (<strong> Xin chào các bạn </strong>) nhưng một số thẻ lại không có thẻ đóng và dữ liệu được khai báo nằm trong các thuộc tính ( ví du: <img>)

### 2.1.2 HTML được xử lý như thế nào

Sau khi một tập tin HTML được hình thành từ các phần tử HTML thì trình duyệt web sẽ đảm nhận việc xử lý nó. Trình duyệt web sẽ đọc hiểu nội dung HTML từ các thẻ bên trong và chuyển nó thành dạng văn bản đã được đánh dấu để đọc, nghe hoặc hiểu.

### 2.1.3 Cấu trúc của một đoạn HTML

HTML sẽ được khai báo bằng các phần tử bởi các từ khóa. Nội dung nằm trong cặp từ khóa là nội dung bạn muốn định dạng với HTML. Ví dụ:

<p> serie học HTML cơ bản </p>

Ngoài ra, trong các thẻ còn có các thuộc tính và thuộc tính sẽ được đặt trong thẻ mở đầu, giá trị của mỗi thuộc tính sẽ được đặt trong dấu ngoặc kép và cách tên thuộc tính bởi dấu (=). Ví dụ:

<form action= "https://acuong.com"></form>

### 2.1.4 Vai trò của HTML

HTML là ngôn ngữ đánh dấu siêu văn bản nên vai trò của HTML trong website là xây dựng cấu trúc siêu văn bản cho website hoặc khai báo các tập tin hình ảnh, video, nhạc…Điều đó không có nghĩa là HTML có thể tạo nên một website mà nó chỉ đóng vài trò hình thành cấu trúc trên website.

* **HTML:** Xây dựng cấu trúc và định dạnh siêu văn bản.
* **CSS:** Định dạng siêu văn bản thô tạo ra từ HTML thành bố cục website có màu sắc, ảnh nền…
* **PHP:** Ngôn ngữ lập trình dùng để xử lý và trao đổi dữ liệu giữa máy chủ với trình duyệt.
* **My SQL:** Hệ quản trị cơ sở dữ liệu truy vấn có cấu trúc.
* **Javascript:** Tạo ra sự kiện tương tác với hành vi của người dùng.

## 2.2 CSS

### 2.2.1 Tổng quan về CSS

Trong tin học, các tập tin định kiểu theo tầng – dịch từ tiếng Anh là Cascading Style Sheets (CSS) – được dùng để miêu tả cách trình bày các tài liệu viết bằng ngôn ngữ HTML và XHTML. Ngoài ra ngôn ngữ định kiểu theo tầng cũng có thể dùng cho XML, SVG, XUL. Các đặc điểm kỹ thuật của CSS được duy trì bởi World Wide Web Consortium (W3C).

Trước đây khi chưa có CSS, những người thiết kế web phải trộn lẫn giữa các thành phần trình bày và nội dung với nhau. Nhưng với sự xuất hiện của CSS, người ta có thể tách rời hoàn toàn phần trình bày và nội dung. Giúp cho phần code của trang web cũng gọn hơn và quan trọng hơn cả là dễ chỉnh sửa hơn.

### 2.2.2 Công dụng CSS

Hạn chế tối thiểu việc làm rối mã HTML của trang Web bằng các thẻ quy định kiểu dáng (chữ đậm, chữ in nghiêng, chữ có gạch chân, chữ màu), khiến mã nguồn của trang Web được gọn gàng hơn, tách nội dung của trang Web và định dạng hiển thị, dễ dàng cho việc cập nhật nội dung.

Tạo ra các kiểu dáng có thể áp dụng cho nhiều trang Web, giúp tránh phải lặp lại việc định dạng cho các trang Web giống nhau.

### 2.2.3 Cách sử dụng CSS

#### 2.2.3.1 Cục bộ

Cách định dạng cục bộ là bạn sử dụng code CSS chèn trực tiếp vào thẻ HTML và chỉ có tác động lên thẻ đó.

<p style=”font-size:16pt; font-weight:bold; color:blue;”>

Nếu bạn thêm code CSS vào trong thẻ này nó sẽ thay đổi cách hiển thị trên trình duyệt

</p>

#### 2.2.3.2 Nhúng vào trang web

Cách thứ hai là khai báo code CSS nằm trong cặp thẻ <style></style> được đặt ở phần <head></head> của tài liệu. Về cơ bản nó cũng chỉ có tác động cục bộ lên file chèn đoạn code CSS này. Nó có phạm vi ảnh hưởng lớn hơn là cách chèn cục bộ được nói ở trên, nhưng nó không có ảnh hưởng đến các file khác trong cùng một trang web.

<head>

<title>Nhúng vào trang</title>

<meta http-equiv=”Content-type” content=”text/html; charset=utf-8″ />

<style type=”text/css”>

h1 {font-size: 16px;}

p {color:blue;}

</style>

</head>

#### 2.2.3.3 Liên kết đến một file biệt lập

Thực tế cách cuối cùng này mới là thế mạnh thực sự của CSS bởi vì bạn chỉ cần tạo ra một file CSS và viết code chỉ một lần. Nó sẽ ảnh hưởng đến toàn bộ trang web chứ không chỉ ảnh hưởng đến một file .html đơn lẻ hoặc chỉ một thẻ trong hàng ngàn thẻ mà bạn có. Bằng cách này chúng ta có thể tách rời hoàn toàn những thành phần trình bày và nội dung ra khỏi nhau. Nếu viết toàn bộ code CSS trong một file và liên kết nó đến những trang còn lại, thì việc thay đổi sửa chữa trang web trở nên cực kỳ đơn giản.

Để liên kết file CSS đến tất cả các file .html, chỉ cần chèn đoạn code sau vào giữa thẻ <head></head> của tài liệu:

<link href="style\_sheet.css" rel="stylesheet" type="text/css" />

## 2.3 Javascript

### 2.3.1 Tổng quan

JavaScript là một ngôn ngữ kịch bản (scripting language) được dùng để tạo các script ở máy client (client-side script) và máy server (server-side script). Các script ở máy client được thực thi tại trình duyệt, các script ở máy server được thực hiện trên server. Chương này sẽ giới thiệu cho chúng ta về ngôn ngữ Javascript, và cách chèn một script vào trong tài liệu HTML.

HTML lúc đầu được phát triển như là một định dạng của tài liệu có thể chuyển dữ liệu trên Internet Tuy nhiên, không lâu sau đó, trọng tâm của HTML nặng tính hàn lâm và khoa học dần chuyển hướng sang người dùng thường nhật vì ngày nay người dùng xem Internet như là một nguồn thông tin và giải trí. Các trang Web ngày càng mang tính sáng tạo và đẹp mắt hơn nhằm thu hút nhiều người dùng hơn. Nhưng thực chất kiểu dáng và nội dung bên trong vẫn không thay đổi. Và người dùng hầu như không thể điều khiển trên trang Web mỗi khi nó được hiển thị.

Javascript được phát triển như là một giải pháp cho vấn đề nêu trên. Javascript là một ngôn ngữ kịch bản được Sun Microsystems và Netscape phát triển. Nó được dùng để tạo các trang Web động và tương tác trên Internet. Đối với những người phát triển HTML, Javascript rất hữu ích trong việc xây dựng các hệ thống HTML có thể tương tác với người dùng.

**JavaScript** là một trong những ngôn ngữ lập trình phổ biến nhất thể giới, nó là ngôn ngữ cho HTML, web, server, PC, laptop, tablet, smart phone…

* JavaScript là một ngôn ngữ kịch bản
* Ngôn ngữ kịch bản là một ngôn ngữ lập trình nhỏ.
* JavaScript là mã lập trình có thể được chèn vào các trang HTML.
* JavaScript được chèn vào các trang web có thể được chạy bởi tất cả các trình duyệt web hiện đại.
* JavaScript là một ngôn ngữ lập trình dễ học.

### 2.3.2 Javascript là gì

Sun Microsystems đã viết ra một ngôn ngữ phức tạp và mạnh mẽ mà chúng ta đã biết đó là ngôn ngữ Java. Mặc dù Java có tính khả dụng cao nhưng nó lại phù hợp nhất đối với các nhà lập trình có kinh nghiệm và cho các công việc phức tạp hơn. Netscape Communications nhận thấy nhu cầu cần một ngôn ngữ thiết kế web có khả năng tương tác vớI ngườI sử dụng hay vớI các Java Applet, dễ sử dụng ngay cả vớI những ngườI lập trình ít kinh nghiệm.

LiveScript là một ngôn ngữ mớI chỉ ở dạng phác thảo, tuy nhiên nó hấp dẫn ngườI sử dụng vì hứa hẹn sẽ đáp ứng tốt những yêu cầu trên. LiveScript được thiết kế theo tinh thần của nhiều ngôn ngữ script đơn giản nhưng nó lại có tính khả dụng cao được thiết kế đặc biệt để xây dựng các trang Web (chẳng hạn như HTML và các form tương tác). Để giúp ‘bán chạy’ ngôn ngữ mới này, Netscape hợp tác với Sun cho ra đời ngôn ngữ Javascript. Trên thực tế, Microsoft là người tiên phong triển khai phiên bản của Javascript (còn có tên là Jscript), nhưng họ không sử dụng các đặc tả chính thức của Javascript.

Mục tiêu của JavaScript là nhằm cung cấp cho các nhà phát triển Web một số khả năng và quyền điều khiển chức năng cho trang Web. Mã Javascript có khả năng nhúng trong tài liệu HTML để điều khiển nội dung của trang Web và kiểm tra sự hợp lệ của dữ liệu mà người dùng nhập vào. Khi một trang hiển thị trong trình duyệt, các câu lệnh được trình duyệt thông dịch và thực thi.

JavaScript

JavaScript là một ngôn ngữ kịch bản dựa trên đối tượng nhằm phát triển các ứng dụng Internet chạy trên phía client và phía server.

Các ứng dụng client được chạy trong một trình duyệt như Netscape Navigator hoặc Internet Explorer, và các ứng dụng server chạy trên một Web server như Microsoft’s Internet Information Server hoặc Netscape Enterprise Server.

### 2.3.3 Hiệu ứng và quy tắc JavaScript

JavaScript là một ngôn ngữ lập trình được nhúng được trong các trang HTML. JavaScript nâng cao tính động và khả năng tương tác cho web-site bằng cách sử dụng các hiệu ứng của nó như thực hiện các phép tính, kiểm tra form, viết các trò chơi, bổ sung các hiệu ứng đặc biệt, tuỳ biến các chọn lựa đồ hoạ, tạo ra các mật khẩu bảo mật và hơn thế nữa.

Chúng ta có thể sử dụng JavaScript để:

**- Tương tác với người dùng**: Chúng ta có thể viết mã để đáp lạI các sự kiện. Các sự này sẽ có thể phát sinh bởi người dùng - - nhấp chuột hay được phát sinh từ hệ thống - - định lại kích thước của trang và v.v.

**- Thay đổi nội dung động:** Mã JavaScript có thể dùng để thay đổi nội dung và vị trí các phần tử một cách động trên một trang nhằm đáp lại sự tương tác với người dùng.

**- Kiểm tra tính hợp lệ dữ liệu**. Chúng ta có thể viết mã nhằm kiểm tra tính hợp lệ của dữ liệu do người dùng nhập vào trước khi nó được gửi lên Web server để xử lý.

Giống như các ngôn ngữ khác, JavaScript cũng tuân thủ một số quy tắc ngữ pháp căn bản. Việc nắm vững các quy tắc ngữ pháp này có thể giúp ta đọc được script và tự viết các script không bị lỗi. Một số trong các luật này bao gồm:

**- Dùng Caps:** JavaScript phân biệt chữ hoa chữ thường.

**- Dùng Pairs:** Trong JavaScript, luôn luôn có cặp ký hiệu mở và đóng. Lỗi sẽ xuất hiện khi bỏ sót hoặc đặt sai một trong hai ký hiệu này.

**- Dùng Spaces (các ký tự trắng).**Như HTML, JavaScript thường bỏ qua ký tự trắng. Trong JavaScript, ta có thể thêm vào các ký tự trắng hoặc các tab giúp cho ta dễ dàng đọc hay sửa các file script.

**- Dùng Chú thích (Comments).**Các chú thích giúp ta ghi chú về chức năng của đoạn script, thờI gian và ngườI tạo ra đoạn script.

Mặc dù cả client-side JavaScript và server-side JavaScript đều dựa trên một ngôn ngữ nền tảng như nhau, nhưng mỗI loạI còn có thêm những tính năng chuyên biệt phù hợp với môi trường mà nó chạy. Nghĩa là, client-side JavaScript bao gồm các đối tượng được định nghĩa sẵn chỉ có thể sử dụng trên trình duyêt, Server-side JavaScript bao gồm các đối tượng và các hàm được định nghĩa sẵn chỉ có thể sử dùng trong các ứng dụng phía server (server-side applications).

### 2.3.4 Nhúng JavaScript vào trong trang Web

#### 2.3.4.1 Dùng thẻ script

Mã JavaScript cụ thể được nhúng vào trong tài liệu HTML bằng thẻ SCRIPT. Chúng ta có thể nhúng nhiều script vào trong cùng một tài liệu, mỗi script nằm trong một thẻ SCRIPT. Khi trình duyệt gặp phải một thẻ <SCRIPT> nào đó, nó sẽ đọc từng dòng một cho đến khi gặp thẻ đóng </SCRIPT>. Tiếp đến nó sẽ kiểm tra lỗi trong các câu lệnh JavaScript. Nếu gặp phải lỗi, nó sẽ cho hiển thị lỗi đó trong chuỗi các hộp cảnh báo (alert boxes) lên màn hình. Nếu không có lỗi, các câu lệnh sẽ được biên dịch sao cho máy tính có thể hiểu được lệnh đó.

Cú pháp như sau:

<script language="JavaScript">

<!--

JavaScript statements;

//-->

</script>

Thuộc tính language trong thẻ script chỉ ra ngôn ngữ mà trình duyệt sẽ dùng để biên dịch script. Chúng ta cũng có thể chỉ rõ phiên bản JavaScript nào sẽ được trình duyệt sử dụng.

<! - -statements //- -> là các thẻ chú thích. Những thẻ này được dùng để báo cho trình duyệt bỏ qua các câu lệnh chứa trong nó. < ! là thẻ mở của thẻ chú thích, //- -> là thẻ đóng. Các thẻ này không bắt buộc phải có, nhưng ta nên đưa chúng vào trong các đoạn script. Chỉ có Netscape 2.0 và các phiên bản sau đó mới hỗ trợ JavaScript. Các thẻ chú thích đảm bảo các phiên bản cũ hoặc các trình duyệt không hỗ trợ JavaScript sẽ bỏ qua các câu lệnh được nhúng trong tài liệu HTML.

#### 2.3.4.2 Dùng file bên ngoài

Thường các câu lệnh JavaScript được nhúng trong một tài liệu HTML. Tuy nhiên, chúng ta có thể tạo ra một file riêng chứa mã JavaScript. File này có thể được liên kết với một tài liệu HTML. Thuộc tính SRC (source) của thẻ SCRIPT dùng để chỉ ra file chứa JavaScript mà nó cần sử dụng. Khi xác định file nguồn, ta có thể dùng tên đường dẫn tương đối và tuyệt đối ở trong thuộc tính SRC.

<script language="JavaScript" src="filename.js">

</script>

Đó là file văn bản chứa các mã lệnh JavaScript, tên file có phần mở rộng là “.js”. Nó chỉ có thể chứa các câu lệnh và các hàm JavaScript. Ta không thể đưa các thẻ HTML vào trong nó.

#### 2.3.4.3 ****Đặt các biểu thức JavaScript là giá trị cho các thuộc tính của thẻ HTML****

có thể dùng biểu thức JavaScript làm giá trị cho thuộc tính của thẻ HTML. Các giá trị đó được thực hiện một cách động mỗi khi trang được trình duyệt tải vào. Cú pháp như sau:

& {expression};

Trong đó expression là biểu thức JavaScript sẽ được thực hiện.

## 2.4 MySQL

### 2.4.1 Tổng quan

MySQL là cơ sở dữ liệu được sử dụng cho các ứng dụng Web có quy mô vừa và nhỏ. Tuy không phải là một cơ sở dữ liệu lớn nhưng chúng cũng có trình giao diện trên Windows hay Linux, cho phép người dùng có thể thao tác các hành động liên quan đến cơ sở dữ liệu.

Cũng giống như các cơ sở dữ liệu, khi làm việc với cơ sở dữ liệu MySQL, bạn đăng ký kết nối, tạo cơ sở dữ liệu, quản lý người dùng, phần quyền sử dụng, thiết kế đối tượng Table của cơ sở dữ liệu và xử lý dữ liệu.

Tuy nhiên, trong bất kỳ ứng dụng cơ sở dữ liệu nào cũng vậy, nếu bản thân chúng có hỗ trợ một trình giao diện đồ hoạ, bạn có thể sử dụng chúng tiện lợi hơn các sử dụng Command line. Bởi vì, cho dù bạn điều khiển MySQL dưới bất kỳ hình thức nào, mục đích cũng quản lý và thao tác cơ sở dữ liệu.

### 2.4.2 Thuật ngữ RDBMS

Trước khi đi vào khái niệm hệ thống cơ sở dữ liệu MySQL, chúng ta cùng xem lại một số định nghĩa liên quan tới cơ sở dữ liệu:

* **Database:** Một cơ sở dữ liệu là một tập hợp các bảng dữ liệu, với dữ liệu có liên quan.
* **Bảng dữ liệu:** Một bảng là một ma trận dữ liệu. Một bảng trong một cơ sở dữ liệu trông giống như một bảng tính đơn giản.
* **Cột:** Một cột chứa cùng một kiểu dữ liệu, ví dụ như tên khách hàng.
* **Hàng:** Một hàng (row, entry, record) là một nhóm dữ liệu có liên quan.
* **Redundancy:** (có thể hiểu là dữ liệu dự phòng) Dữ liệu được lưu giữ hai lần, để làm cho hệ thống nhanh hơn.
* **Primary Key:** Một Primary Key (Khóa chính) là duy nhất. Một giá trị key không thể xuất hiện hai lần trong một bảng. Với một key, bạn có thể tìm thấy phần lớn trên một hàng.
* **Foreign Key:** Bạn tưởng tượng về Foreign Key như là cái ghim liên kết giữa hai bảng.
* **Compound Key:** Một Compound Key (hay composite key) là một key mà gồm nhiều cột, bởi vì một cột là không duy nhất.
* **Index:** Một chỉ mục trong một cơ sở dữ liệu tương tự như chỉ mục trong một cuốn sách.
* **Referential Integrity:** Đảm bảo rằng một giá trị Foreign Key luôn luôn trỏ tới một hàng đang tồn tại.

### 2.4.3 MySQL Database

MySQL là một RDBMS nhanh và dễ dàng để sử dụng. MySQL đang được sử dụng cho nhiều công việc kinh doanh từ lớn tới nhỏ. MySQL được phát triển, được công bố, được hỗ trợ bởi MySQL AB, là một công ty của Thụy Điển. MySQL trở thành khá phổ biến vì nhiều lý do:

* MySQL là mã ngồn mở. Vì thế, để sử dụng nó, bạn chẳng phải mất một xu nào.
* MySQL là một chương trình rất mạnh mẽ.
* MySQL sử dụng một Form chuẩn của ngôn ngữ dữ liệu nổi tiếng là SQL.
* MySQL làm việc trên nhiều Hệ điều hành và với nhiều ngôn ngữ như PHP, PERL, C, C++, Java, …
* MySQL làm việc nhanh và khỏe ngay cả với các tập dữ liệu lớn.
* MySQL rất thân thiện với PHP, một ngôn ngữ rất đáng giá để tìm hiểu để phát triển Web.
* MySQL hỗ trợ các cơ sở dữ liệu lớn, lên tới 50 triệu hàng hoặc nhiều hơn nữa trong một bảng. Kích cỡ file mặc định được giới hạn cho một bảng là 4 GB, nhưng bạn có thể tăng kích cỡ này (nếu hệ điều hành của bạn có thể xử lý nó) để đạt tới giới hạn lý thuyết là 8 TB.
* MySQL là có thể điều chỉnh. Giấy phép GPL mã nguồn mở cho phép lập trình viên sửa đổi phần mềm MySQL để phù hợp với môi trường cụ thể của họ.

## 2.5 Ajax

### 2.5.1 Tổng quan

AJAX là thuật ngữ viết tắt của Asynchronous Javascript and XML ( JS và XML không đồng bộ). AJAX có thể đọc là "trao quyền cho javascript" và thông qua js để cung cấp một công nghệ phía client-script để gọi ngầm một lệnh background để phía server thực hiện và nhận thông tin trả về, update thông tin của trang nhanh mà không cần phải load lại cả trang, rất mất thời gian load lại những cái không cần update.

**Ví dụ:** Để kiểm tra nickname đã tồn tại trong cơ sở dữ liệu hay chưa thì khi thành viên vừa nhập xong nick name, thông tin trên form lập tức javascript nắm thông tin rồi thực hiện gọi một hàm bằng PHP bên phía server để kiểm tra thông tin sau đó trả kết quả html về cho javascript xuất ra cho người dùng biết tính hợp lệ của nickname, lúc đó ngươid ùng có thể thay đổi ngay chớ không cần đợi submit rồi kiểm tra và nhập lại như dùng PHP truyền thống.

### 2.5.2 Ưu điểm

* Nó giúp việc thiết kế web đa dạng hơn và tăng tính tương tác của website với người dùng.
* Nó sử dụng các công nghệ đã có sẵn nên dễ học và sử dụng
* Nhờ tính phổ biến của nó, đã khuyến khích việc phát triển các khuôn mẫu mà sẽ giúp lập trình viên tránh khỏi các vết xe đổ trước.
* Được hỗ trợ trong các trình duyệt phổ biến hiện nay

### 2.5.3 Nhược điểm

* Bạn không thể bookmark nó vào favourite trên trinh duyệt hay gởi link đến cho bạn bè, vì tất cả quá trình nó thực hiện ngầm và không hiển thị trên address.
* Không thể hiện thị nội dung trên các trang tìm kiếm vì các trang tìm kiếm hiện nay vẫn chưa hỗ trợ tìm vì rất khó tìm và gần như không thể tìm đc.
* Không thể sử dụng nút back vì back cũng là chính nó.

Với một số trình duyệt, do nhu cầu bảo mật, sẽ tắt chức năng thực hiện javascript nên ajax không thể chạy, hay trong một vài host, không hỗ trợ vào sâu cấu hình server nên hay bị lỗi "Access denied".

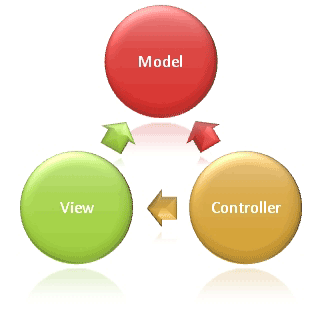
# Chương 3. THIẾT KẾ - CÀI ĐẶT

## 3.1 Kiến trúc hệ thống

### 3.1.1 Mô hình MVC

#### 3.1.1.1 Tổng quan

Mô hình MVC là một chuẩn mô hình và đóng vai trò quan trọng trong quá trình xây dựng – phát triển – vận hành và bảo trì một hệ thống hay một ứng dụng – phần mềm. Nó tạo ra một mô hình 3 lớp Model – View – Controller tách biệt và tương tác nhau, giúp các chuyên gia có thể dễ dàng dựa vào mô hình để trao đổi và xử lý những nghiệp vụ một cách nhanh chóng. Đây là một mô hình đã xuất hiện từ những năm 70 của thế kỷ 20 tại phòng thí nghiệm Xerox PARC ở Palo Alto, nó không phụ thuộc vào môi trường, nền tảng xây dựng hay ngôn ngữ phát triển. Chúng ta có thể áp dụng mô hình MVC vào các dự án trong môi trường Windows, Linux… và sử dụng bất kỳ ngôn ngữ nào như PHP, ASP, JSP…



**Mô hình MVC** được chia làm 3 lớp xử lý gồm **Model – View – Controller :**

**- Model** : là nơi chứa những nghiệp vụ tương tác với dữ liệu hoặc hệ quản trị cơ sở dữ liệu (mysql, mssql… ); nó sẽ bao gồm các class/function xử lý nhiều nghiệp vụ như kết nối database, truy vấn dữ liệu, thêm – xóa – sửa dữ liệu…

**- View** : là nới chứa những giao diện như một nút bấm, khung nhập, menu, hình ảnh… nó đảm nhiệm nhiệm vụ hiển thị dữ liệu và giúp người dùng tương tác với hệ thống.

**- Controller** : là nới tiếp nhận những yêu cầu xử lý được gửi từ người dùng, nó sẽ gồm những class/ function xử lý nhiều nghiệp vụ logic giúp lấy đúng dữ liệu thông tin cần thiết nhờ các nghiệp vụ lớp **Model** cung cấp và hiển thị dữ liệu đó ra cho người dùng nhờ lớp **View.**

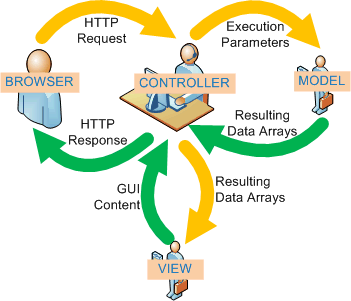
#### ****3.1.1.2 Sự tương tác giữa các lớp trong mô hình MVC****

**Controller – View** sẽ lấy những hình ảnh, nút bấm…hoặc hiển thị dữ liệu được trả ra từ Controller để người dùng có thể quan sát và thao tác. Trong sự tương tác này cũng có thể không có dữ liệu được lấy từ Model và khi đó nó chỉ chịu trách nhiệm hiển thị đơn thuần như hình ảnh, nút bấm…

**Controller – Model** là luồng xử lý khi controller tiếp nhận yêu cầu và các tham số đầu vào từ người dùng, controller sẽ sử dụng các lớp/hàm trong Model cần thiết để lấy ra những dữ liệu chính xác.

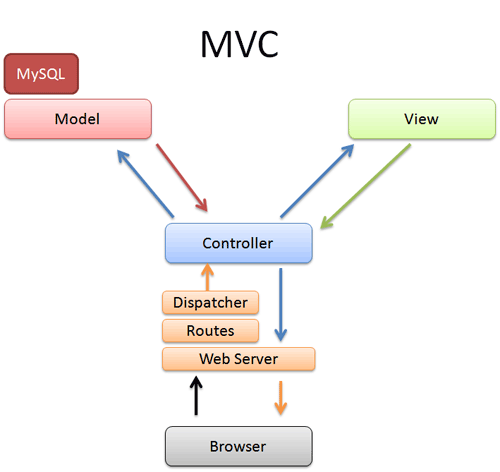
**View – Model** có thể tương tác với nhau mà không qua Controller, nó chỉ đảm nhận hiển thị dữ liệu chứ không phải qua bất kỳ xử lý nghiệp vụ logics nào. Nó giống như các vùng dữ liệu hiển thị tĩnh trên các website như block slidebar…

#### ****3.1.1.3 Quy trình hoạt động mô hình MVC trong dự án website****



Mô hình trên là thể hiện mô hình MVC trong các dự án website và sẽ hoạt động theo quy trình sau:

1. Người dùng sử dụng một BROWSER trình duyệt web bất kỳ (Firefox, Chrome, IE,…) để có thể gửi những yêu cầu (HTTP Request) có thể kèm theo những dữ liệu nhập tới những CONTROLLER xử lý tương ứng. Việc xác định Controllerr xử lý sẽ dựa vào một bộ Routing điều hướng.
2. Khi CONTROLLER nhận được yêu cầu gửi tới, nó sẽ chịu trách nhiệm kiểm tra yêu cầu đó có cần dữ liệu từ MODEL hay không? Nếu có, nó sẽ sử dụng các class/function cần thiết trong MODEL và nó sẽ trả ra kết quả( Resulting Arrays), khi đó CONTROLLER sẽ xử lý giá trị đó và trả ra VIEW để hiển thị.
3. Khi nhận được dữ liệu từ CONTROLLER, VIEW sẽ chịu trách nhiệm xây dựng các thành phẩn hiển thị như hình ảnh, thông tin dữ liệu… và trả về GUI Content để CONTROLLER  đưa ra kết quả lên màn hình BROWSER.
4. BROWSER sẽ nhận giá trị trả về( HTTP Response) và sẽ hiển thị với người dùng. Kết thúc một quy trình hoạt động.



#### ****3.1.1.4 Ưu điểm và nhược điểm của mô hình MVC****

#### Ưu điểm

Các dự án có thể áp dụng ngay mô hình MVC mà không phụ thuộc môi trường, nền tảng xây dựng hay ngôn ngữ lập trình phát triển.

Quy hoạch các class/ function vào các thành phần riêng biệt Controller – Model – View, khi đó sẽ dễ dàng xây dựng – phát triển – quản lý – vận hành và bảo trì một dự án, tạo sự rõ ràng, trong sáng trong quá trình phát triển dự án, kiểm soát được các luồng xử lý và tạo ra các thành phần xử lý nghiệp vụ chuyên biệt hóa.

Tạo thành mô hình chuẩn cho nhiều dự án, các chuyên gia sẽ tiếp cận – tìm hiểu những dự án đó một cách nhanh chóng và hiệu quả . Nếu bạn nắm rõ mô hình MVC của một dự án nào đó, thì khi tiếp cận với một dự án khác mà bạn chưa từng biết hoặc tiếp xúc, nhưng nó lại được xây dựng với mô hình MVC thì sẽ không khó khăn gì mà cực kỳ dễ dàng. Học một nhưng có thể hiểu và sử dụng được mười.

Giúp các chuyên gia lập trình, nhà quản lý, nhà đầu tư, PM… có thể hiểu được dự án hoạt động ra sao hoặc giúp các lập trình viên dễ dàng quản lý – phát triển dự án. Nó không phải ngôn ngữ, nhưng khi họ cùng nhìn vào nó thì sẽ tự hiểu nó là gì, khi đó họ có thể trao đổi các yêu cầu và bàn bạc công việc.

Đây là một mô hình chuẩn, nó tối ưu nhất hiện nay so với nhiều mô hình khác và được sử dụng trong nhiều dự án và nhiều lĩnh vực, đặc biệt trong công nghệ sản xuất ứng dụng – phần mềm. Các lập trình viên sử dụng mô hình chuẩn MVC để có thể dễ dàng phân phối và chuyển giao công nghệ.

Đây là mô hình đơn giản, xử lý những nghiệp vụ đơn giản, và dễ dàng triển khai với các dự án nhỏ.

#### Nhược điểm

Yêu cầu về chuyên môn khá cao, có kiến thức vững về các mô hình chuẩn, khó triển khai với những dự án yêu cầu phúc tạp hơn. Hiện nay đang có một khái niệm mô hình mới đó là HMVC đang dần thay thế cho MVC

### 3.1.2 Mô hình Client – Server

#### 3.1.2.1 Tổng quan

Client – Server Là một mô hình nổi tiếng trong mạng máy tính, được áp dụng rất rộng rãi và là mô hình của mọi trang web hiện có. Ý tưởng của mô hình này là máy con (đóng vài trò là máy khách) gửi một yêu cầu (request) để máy chủ (đóng vai trò người cung ứng dịch vụ), máy chủ sẽ xử lý và trả kết quả về cho máy khách.

Một mô hình ngược lại là mô hình master-slaver, trong đó máy chủ (đóng vai trò ông chủ) sẽ gửi dữ liệu đến máy con (đóng vai trò nô lệ) bất kể máy con có cần hay không.

#### 3.1.2.2 Web Client/Server

Thuật ngữ server được dùng cho những chương trình thi hành như một dịch vụ trên toàn mạng. Các chương trình server này chấp nhận tất cả các yêu cầu hợp lệ đến từ mọi nơi trên mạng, sau đó nó thi hành dịch vụ và trả kết quả về máy yêu cầu. Một chương trình được coi là client khi nó gửi các yêu cầu tới máy có chương trình server và chờ đợi câu trả lời từ server. Chương trình server và client nói chuyện với nhau bằng các thông điệp (messages) thông qua một cổng truyền thông liên tác IPC (Interprocess Communication).

Để một chương trình server và một chương trình client có thể giao tiếp được với nhau thì giữa chúng phải có một chuẩn để nói chuyện, chuẩn này được gọi là giao thức. Nếu một chương trình client nào đó muốn yêu cầu lấy thông tin từ server thì nó phải tuân theo giao thức mà server đó đưa ra. Bản thân chúng ta khi cần xây dựng một mô hình client/server cụ thể thì ta cũng có thể tự tạo ra một giao thức riêng nhưng thường chúng ta chỉ làm được điều này ở tầng ứng dụng của mạng. Với sự phát triển mạng như hiện này thì có rất nhiều giao thức chuẩn trên mạng ra đời nhằm đáp ứng nhu cầu phát triển này.

Các giao thức chuẩn (ở tầng mạng và vận chuyển) được sử dụng rộng rãi nhất hiện nay như: giao thức TCP/IP, giao thức SNA của IBM, OSI, ISDN, X.25 hoặc giao thức LAN-to-LAN NetBIOS. Một máy tính chứa chương trình server được coi là một máy chủ hay máy phục vụ (server) và máy chứa chương trình client được coi là máy tớ (client). Mô hình mạng trên đó có các máy chủ và máy tớ giao tiếp với nhau theo 1 hoặc nhiều dịch vụ được gọi là mô hình client/server.

Thực tế thì mô hình **client/server** là sự mở rộng tự nhiên và tiện lợi cho việc truyền thông liên tiến trình trên các máy tính cá nhân. Mô hình này cho phép xây dựng các chương trình **client/serve**r một cách dễ dàng và sử dụng chúng để liên tác với nhau để đạt hiệu quả hơn.

Mô hình client/server như sau: **Client/Server** là mô hình tổng quát nhất, trên thực tế thì một server có thể được nối tới nhiều server khác nhằm làm việc hiệu quả và nhanh hơn. Khi nhận được 1 yêu cầu từ client, server này có thể gửi tiếp yêu cầu vừa nhận được cho server khác ví dụ như database server vì bản thân nó không thể xử lý yêu cầu này được. Máy server có thể thi hành các nhiệm vụ đơn giản hoặc phức tạp.

Ví dụ như một máy chủ trả lời thời gian hiện tại trong ngày, khi một máy client yêu cầu lấy thông tin về thời gian nó sẽ phải gửi một yêu cầu theo một tiêu chuẩn do server định ra, nếu yêu cầu được chấp nhận thì máy server sẽ trả về thông tin mà client yêu cầu. Có rất nhiều các dịch vụ server trên mạng nhưng nó đều hoạt động theo nguyên lý là nhận các yêu cầu từ client sau đó xử lý và trả kết quả cho client yêu cầu. Thông thường chương trình server và client được thi hành trên hai máy khác nhau. Cho dù lúc nào server cũng ở trạng thái sẵn sàng chờ nhận yêu cầu từ client nhưng trên thực tế một tiến trình liên tác qua lại (interaction) giữa client và server lại bắt đầu ở phía client, khi mà client gửi tín hiệu yêu cầu tới server. Các chương trình server thường đều thi hành ở mức ứng dụng (tầng ứng dụng của mạng).

Sự thuận lợi của phương pháp này là nó có thể làm việc trên bất cứ một mạng máy tính nào có hỗ trợ giao thức truyền thông chuẩn cụ thể ở đây là giao thức TCP/IP. Với các giao thức chuẩn này cũng giúp cho các nhà sản xuất có thể tích hợp nhiều sản phẩm khác nhau của họ lên mạng mà không gặp phải khó khăn gì. Với các chuẩn này thì các chương trình server cho một dịch vụ nào đấy có thể thi hành trên một hệ thống chia sẻ thời gian (timesharing system) với nhiều chương trình và dịch vụ khác hoặc nó có thể chạy trên chính một máy tính các nhân bình thường. Có thể có nhiều chương server cùng làm một dịch vụ, chúng có thể nằm trên nhiều máy tính hoặc một máy tính.

Với mô hình trên chúng ta nhận thấy rằng mô hình client/server chỉ mang đặc điểm của phần mềm không liên quan gì đến phần cứng mặc dù trên thực tế yêu cầu cho một máy server là cao hơn nhiều so với máy client. Lý do là bởi vì máy server phải quản lý rất nhiều các yêu cầu từ các clients khác nhau trên mạng. Ưu và nhược điểm chính Có thể nói rằng với mô hình client/server thì mọi thứ dường như đều nằm trên bàn của người sử dụng, nó có thể truy cập dữ liệu từ xa (bao gồm các công việc như gửi và nhận file, tìm kiếm thông tin, ...) với nhiều dịch vụ đa dạng mà mô hình cũ không thể làm được.

Mô hình **client/server** cung cấp một nền tảng lý tưởng cho phép tích hợp các kỹ thuật hiện đại như mô hình thiết kế hướng đối tượng, hệ chuyên gia, hệ thông tin địa lý (GIS) ... Một trong những vấn đề nảy sinh trong mô hình này đó là tính an toàn và bảo mật thông tin trên mạng. Do phải trao đổi dữ liệu giữa 2 máy ở 2 khu vực khác nhau cho nên dễ dàng xảy ra hiện tượng thông tin truyền trên mạng bị lộ.

#### 3.1.2.3 Các thành phần

#### 3.1.2.3.1 Client

Client Trong **mô hình client/server**, người ta còn định nghĩa cụ thể cho một máy client là một máy trạm mà chỉ được sử dụng bởi 1 người dùng với để muốn thể hiện tính độc lập cho nó. Máy client có thể sử dụng các hệ điều hành bình thường như Win9x, DOS, OS/2... Bản thân mỗi một client cũng đã được tích hợp nhiều chức năng trên hệ điều hành mà nó chạy, nhưng khi được nối vào một mạng LAN, WAN theo mô hình client/server thì nó còn có thể sử dụng thêm các chức năng do hệ điều hành mạng (NOS) cung cấp với nhiều dịch vụ khác nhau (cụ thể là các dịch vụ do các server trên mạng này cung cấp), ví dụ như nó có thể yêu cầu lấy dữ liệu từ một server hay gửi dữ liệu lên server đó…

Thực tế trong các ứng dụng của mô hình client/server, các chức năng hoạt động chính là sự kết hợp giữa client và server với sự chia sẻ tài nguyên, dữ liệu trên cả 2 máy Vai trò của client Trong mô hình client/server, client được coi như là người sử dụng các dịch vụ trên mạng do một hoặc nhiều máy chủ cung cấp và server được coi như là người cung cấp dịch vụ để trả lời các yêu cầu của các clients. Điều quan trọng là phải hiểu được vai trò hoạt động của nó trong một mô hình cụ thể, một máy client trong mô hình này lại có thể là server trong một mô hình khác.

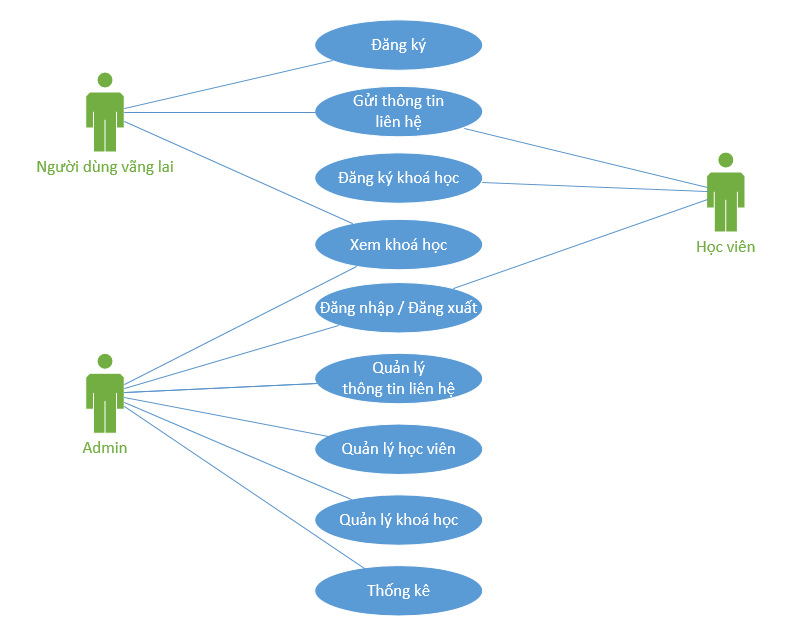
#### 3.1.2.3.2 Server

Server còn được định nghĩa như là một máy tính nhiều người sử dụng (multiuser computer). Vì một server phải quản lý nhiều yêu cầu từ các client trên mạng cho nên nó hoạt động sẽ tốt hơn nếu hệ điều hành của nó là đa nhiệm với các tính năng hoạt động độc lập song song với nhau như hệ điều hành UNIX, WINDOWS... Server cung cấp và điều khiển các tiến trình truy cập vào tài nguyên của hệ thống. Các ứng dụng chạy trên server phải được tách rời nhau để một lỗi của ứng dụng này không làm hỏng ứng dụng khác.

Tính đa nhiệm đảm bảo một tiến trình không sử dụng toàn bộ tài nguyên hệ thống. Vai trò của server. Như chúng ta đã bàn ở trên, server như là một nhà cung cấp dịch vụ cho các clients yêu cầu tới khi cần, các dịch vụ như cơ sở dữ liệu, in ấn, truyền file, hệ thống... Các ứng dụng server cung cấp các dịch vụ mang tính chức năng để hỗ trợ cho các hoạt động trên các máy clients có hiệu quả hơn. Sự hỗ trợ của các dịch vụ này có thể là toàn bộ hoặc chỉ một phần thông qua IPC.

Để đảm bảo tính an toàn trên mạng cho nên server này còn có vai trò như là một nhà quản lý toàn bộ quyền truy cập dữ liệu của các máy clients, nói cách khác đó là vai trò quản trị mạng. Có rất nhiều cách thức hiện nay nhằm quản trị có hiệu quả, một trong những cách đang được sử dụng đó là dùng tên Login và mật khẩu.

## 3.2 Sơ đồ Use-case



1. **Danh sách các actor**

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên Actor** | **Mô tả** |
| 1 | Admin | Chịu trách nhiệm quản lý hệ thống |
| 2 | Học viên | Là người dùng đã đăng ký tài khoản |
| 3 | Người dung vãng lai | Là người dùng vãng lai chưa đăng ký tài khoản nhưng vẫn có thể xem thông tin các khoá học và gửi feedback (thông tin liên hệ) cho hệ thống |

1. **Danh sách các usecase**

|  |  |
| --- | --- |
| **STT** | **Tên Use-case** |
| 1 | Đăng ký |
| 2 | Gửi thông tin liên hệ |
| 3 | Đăng ký khoá học |
| 4 | Xem khoá học |
| 5 | Đăng nhập / Đăng xuất |
| 6 | Quản lý thông tin liên hệ |
| 7 | Quản lý học viên |
| 8 | Quản lý khoá học |
| 9 | Thống kê |

1. **Đặc tả usecase**
2. **Đăng ký**

|  |
| --- |
| **Tóm tắt:** Đăng ký tài khoản vào hệ thống |
| **Dòng sự kiện chính:**   1. Người dùng bấm vào nút “Đăng ký” ở phía trên bên phải màn hình trang chủ 2. Hệ thống hiển thị form đăng ký 3. Người dùng điền đầy đủ các thông tin bắt buộc và bấm vào nút “Đăng ký” 4. Hệ thống kiểm tra thông tin đăng ký 5. Người dùng đăng ký và tự động đăng nhập vào hệ thống, sau đó hệ thống hiển thị trang chủ   Dòng sự kiện khác:   * Tên tài khoản đã tồn tại * Các trường bắt buộc chưa được nhập * Nhập lại mật khẩu không đúng |
| **Dòng sự kiện phụ:**   1. Tên tài khoản đã tồn tại: Hệ thống hiển thị hình ảnh để thông báo tên tài khoản không khả dụng. Người dùng nhập đến khi nào hợp lệ thì hiển thị hình ảnh thông báo hợp lệ. 2. Một trong các trường “Tên tài khoản”, “Mật khẩu”, “Nhập lại mật khẩu”, “Email” bị để trống: Hệ thống hiển thị thống báo “Vui lòng điền vào trường này” và yêu cầu nhập lại. 3. Nhập lại mật khẩu không đúng: Hệ thống thông báo nhập lại mật khẩu nhập lại khi 2 trường “Mật khẩu” và “Nhập lại mật khẩu” không giống nhau. |
| **Tiền điều kiện:**  Actor: Admin, Học viên  Điều kiện: Người dùng phải có tài khoản |
| **Hậu điều kiện:**  Actor: Admin, Học viên  Điều kiện: Người dùng phải có tài khoản |

1. **Gửi thông tin liên hệ**

|  |
| --- |
| **Tóm tắt:** Gửi thông tin liên hệ, cần tư vấn, đặt câu hỏi |
| **Dòng sự kiện chính:**   1. Người dùng bấm vào nút “Liên hệ” ở phía trên bên trái màn hình trang chủ (trong thanh menu) 2. Hệ thống hiển thị form liên hệ 3. Người dùng điền đầy đủ các thông tin bắt buộc và bấm vào nút “Gửi” 4. Hệ thống kiểm tra thông tin đăng ký 5. Người dùng gửi thông tin vào hệ thống, hệ thống hiển thị trang chủ   Dòng sự kiện khác:   * Các trường bắt buộc chưa được nhập |
| **Dòng sự kiện phụ:**   1. Một trong các trường “Họ tên”, “Email”, “Chủ đề”, “Lời nhắn” bị để trống: Hệ thống hiển thị thống báo “Vui lòng điền vào trường này” và yêu cầu nhập lại. |
| **Tiền điều kiện:**  Actor: Học viên, Người dùng vãng lai  Điều kiện: Không có |
| **Hậu điều kiện:**  Actor: Học viên, Người dùng vãng lai  Điều kiện: Không có |

1. **Đăng ký khoá học**

|  |
| --- |
| **Tóm tắt:** Học viên đăng ký khoá học |
| **Dòng sự kiện chính:**   1. Người dùng đang nhập tài khoản học viên, chọn khoá học muốn đăng ký bấm vào nút “Đăng ký học” nếu chưa đăng ký khoá học đó 2. Khoá học sẽ được đăng kí, sau đó hệ thống hiển thị trang danh sách khoá học của người dùng |
| **Dòng sự kiện phụ:**  Không có |
| **Tiền điều kiện:**  Actor: Học viên  Điều kiện: Người dùng đăng nhập tài khoản học viên |
| **Hậu điều kiện:**  Actor: Học viên  Điều kiện: Người dùng đăng nhập tài khoản học viên |

1. **Xem khoá học**

|  |
| --- |
| **Tóm tắt:** Người dùng xem thông tin khoá học |
| **Dòng sự kiện chính:**   1. Người dùng chọn khoá học muốn xem thông tin 2. Hệ thống hiển thị trang thông tin khoá học đó |
| **Dòng sự kiện phụ:**  Không có |
| **Tiền điều kiện:**  Actor: Tất cả  Điều kiện: Không có |
| **Hậu điều kiện:**  Actor: Tất cả  Điều kiện: Không có |

Đăng nhập / Đăng xuất

1. **Đăng nhập**

|  |
| --- |
| **Tóm tắt:** Đăng nhập vào hệ thống |
| **Dòng sự kiện chính:**   1. Người dùng bấm vào nút “Đăng nhập” ở phía trên bên phải màn hình trang chủ 2. Hệ thống hiển thị form đăng nhập 3. Người dùng điền đầy đủ các thông tin bắt buộc và bấm vào nút “Đăng nhập” 4. Hệ thống kiểm tra thông tin đăng nhập 5. Người dùng đăng nhập vào hệ thống, sau đó hệ thống hiển thị trang chủ nếu là tài khoản học viên, hệ thống hiển thị trang quản trị nếu là tài khoản Admin   Dòng sự kiện khác:   * Thông tin được nhập không đúng   Nhập lại mật khẩu không đúng |
| **Dòng sự kiện phụ:**   1. Thông tin được nhập không đúng: người dùng để trống trường “Tên đăng nhập” hoặc “Mật khẩu” hoặc thông tin nhập không đúng. Hệ thống sẽ thông báo lỗi. |
| **Tiền điều kiện:**  Actor: Admin, Học viên  Điều kiện: Người dùng phải có tài khoản, chưa đăng nhập vào hệ thống |
| **Hậu điều kiện:**  Actor: Admin, Học viên  Điều kiện: Người dùng phải có tài khoản, đã đăng nhập vào hệ thống |

1. **Đăng xuất**

|  |
| --- |
| **Tóm tắt:** Đăng xuất khỏi hệ thống |
| **Dòng sự kiện chính:**   1. Người dùng bấm vào nút “Đăng xuất” ở phía trên bên phải màn hình trang chủ đối với tài khoản học viên, hoặc ở phía dưới bên trái màn hình quản trị đối với tài khoản Admin 2. Người dùng đăng xuất vào hệ thống, sau đó hệ thống hiển thị trang chủ |
| **Dòng sự kiện phụ:**  Không có |
| **Tiền điều kiện:**  Actor: Admin, Học viên  Điều kiện: Người dùng phải có tài khoản, đã đăng nhập vào hệ thống |
| **Hậu điều kiện:**  Actor: Admin, Học viên  Điều kiện: Người dùng phải có tài khoản, đăng xuất khỏi hệ thống |

1. **Quản lý thông tin liên hệ**

|  |
| --- |
| **Tóm tắt:** Quản lý thông tin liên hệ, xem danh sách và xoá thông tin liên hệ |
| **Dòng sự kiện chính:**   1. Người dùng đăng nhập vào hệ thống và truy cập trang quản lý liên hệ 2. Chọn thông tin liên hệ cần xoá và bấm nút “Xoá” 3. Hệ thống hiển thị lại danh sách thông tin liên hệ sau khi xoá |
| **Dòng sự kiện phụ:**  Không có |
| **Tiền điều kiện:**  Actor: Admin  Điều kiện: Người dùng phải có tài khoản Admin, đã đăng nhập vào hệ thống |
| **Hậu điều kiện:**  Actor: Admin  Điều kiện: Người dùng phải có tài khoản Admin, đã đăng nhập vào hệ thống |

1. **Quản lý học viên**

|  |
| --- |
| **Tóm tắt:** Quản lý thông tin học viên, xem danh sách và xoá học viên |
| **Dòng sự kiện chính:**   1. Người dùng đăng nhập vào hệ thống và truy cập trang quản lý học viên 2. Chọn học viên cần xoá và bấm “Xoá” 3. Hệ thống hiển thị lại danh sách học viên sau khi xoá |
| **Dòng sự kiện phụ:**  Không có |
| **Tiền điều kiện:**  Actor: Admin  Điều kiện: Người dùng phải có tài khoản Admin, đã đăng nhập vào hệ thống |
| **Hậu điều kiện:**  Actor: Admin  Điều kiện: Người dùng phải có tài khoản Admin, đã đăng nhập vào hệ thống |

1. **Quản lý khoá học**

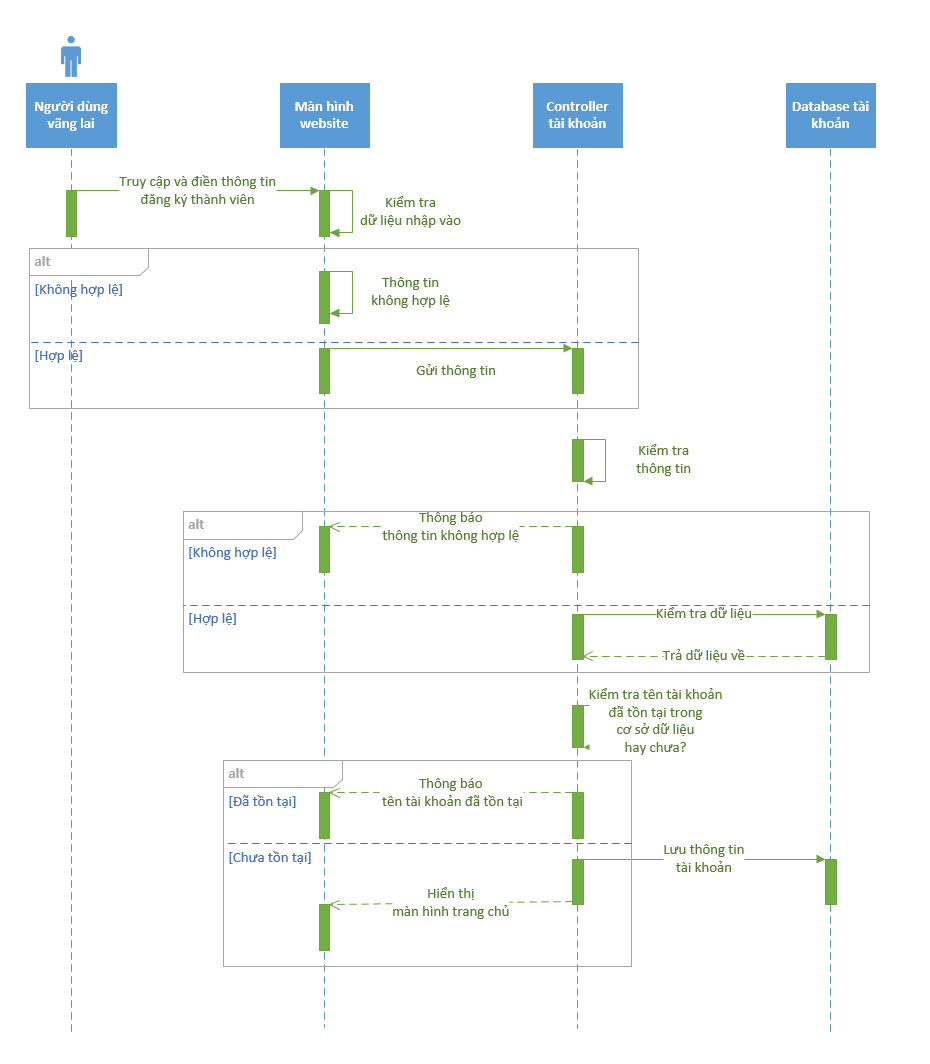
|  |
| --- |
| **Tóm tắt:** Quản lý thông tin khoá học, xem danh sách và thêm, xoá, sửa học viên |
| **Dòng sự kiện chính:**   1. Người dùng đăng nhập vào hệ thống và truy cập trang quản lý khoá học 2. Nhập từ khoá cần tìm (hoặc để trống), sau đó bấm nút “tìm khoá học” 3. Hệ thống hiển thị lại danh sách khoá học sau khi tìm kiếm   Dòng sự kiện khác:   * Thêm khoá học * Sửa khoá học * Xoá khoá học * Xem khoá học |
| **Dòng sự kiện phụ:**   1. Thêm khoá học: người dùng bấm vào nút “Thêm khoá học”, hệ thống chuyển sang trang thêm khoá học mới, người dùng điền các trường cần thiết và bấm vào nút “lưu khoá học”, sau đó hệ thống sẽ chuyển về trang quản lý khoá học. 2. Sửa khoá học: người dùng bấm vào nút “Sửa” vào khoá học cần sửa, hệ thống chuyển sang trang sửa thông tin khoá học, người dùng chỉnh sửa các trường cần thiết và bấm vào nút “lưu khoá học”, sau đó hệ thống sẽ chuyển về trang quản lý khoá học. 3. Xoá khoá học: người dùng bấm vào nút “Xoá” vào khoá học cần xoá, hệ thống sẽ nạp lại trang quản lý khoá học sau khi xoá. 4. Xem khoá học: người dùng bấm vào nút “Xoá” vào khoá học cần xem, hệ thống sẽ chuyển đến trang chi tiết khoá học. |
| **Tiền điều kiện:**  Actor: Admin  Điều kiện: Người dùng phải có tài khoản Admin, đã đăng nhập vào hệ thống |
| **Hậu điều kiện:**  Actor: Admin  Điều kiện: Người dùng phải có tài khoản Admin, đã đăng nhập vào hệ thống |

1. **Thống kê**

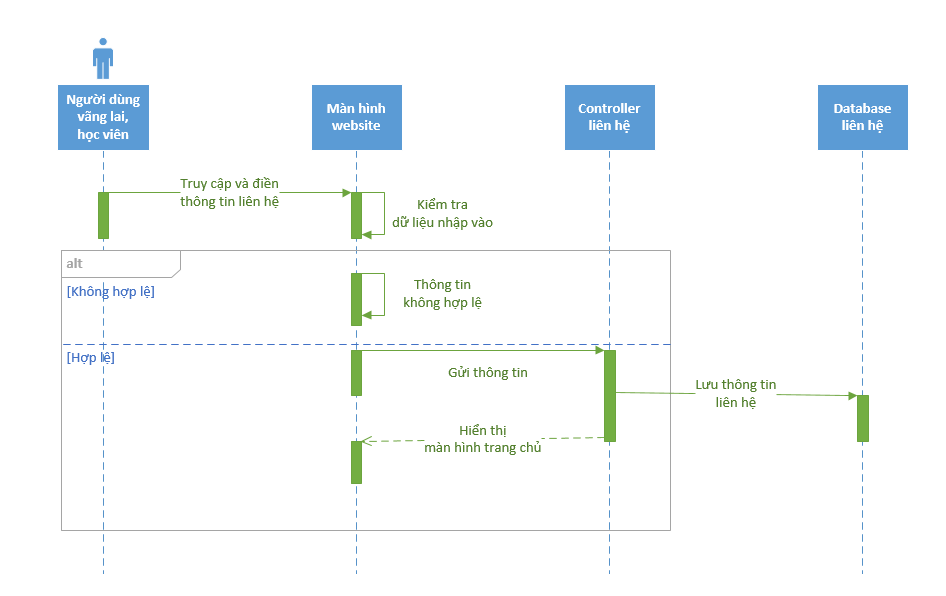
|  |
| --- |
| **Tóm tắt:** Thống kê số lượng khoá học, học viên, số học viên đã học, số phản hồi (thông tin liên hệ) |
| **Dòng sự kiện chính:**   1. Người dùng đăng nhập vào hệ thống và truy cập trang thống kê 2. Hệ thống hiển thị trang thống kê cho người dùng |
| **Dòng sự kiện phụ:**  Không có |
| **Tiền điều kiện:**  Actor: Admin  Điều kiện: Người dùng phải có tài khoản Admin, đã đăng nhập vào hệ thống |
| **Hậu điều kiện:**  Actor: Admin  Điều kiện: Người dùng phải có tài khoản Admin, đã đăng nhập vào hệ thống |

## 3.3 Sơ đồ Sequence

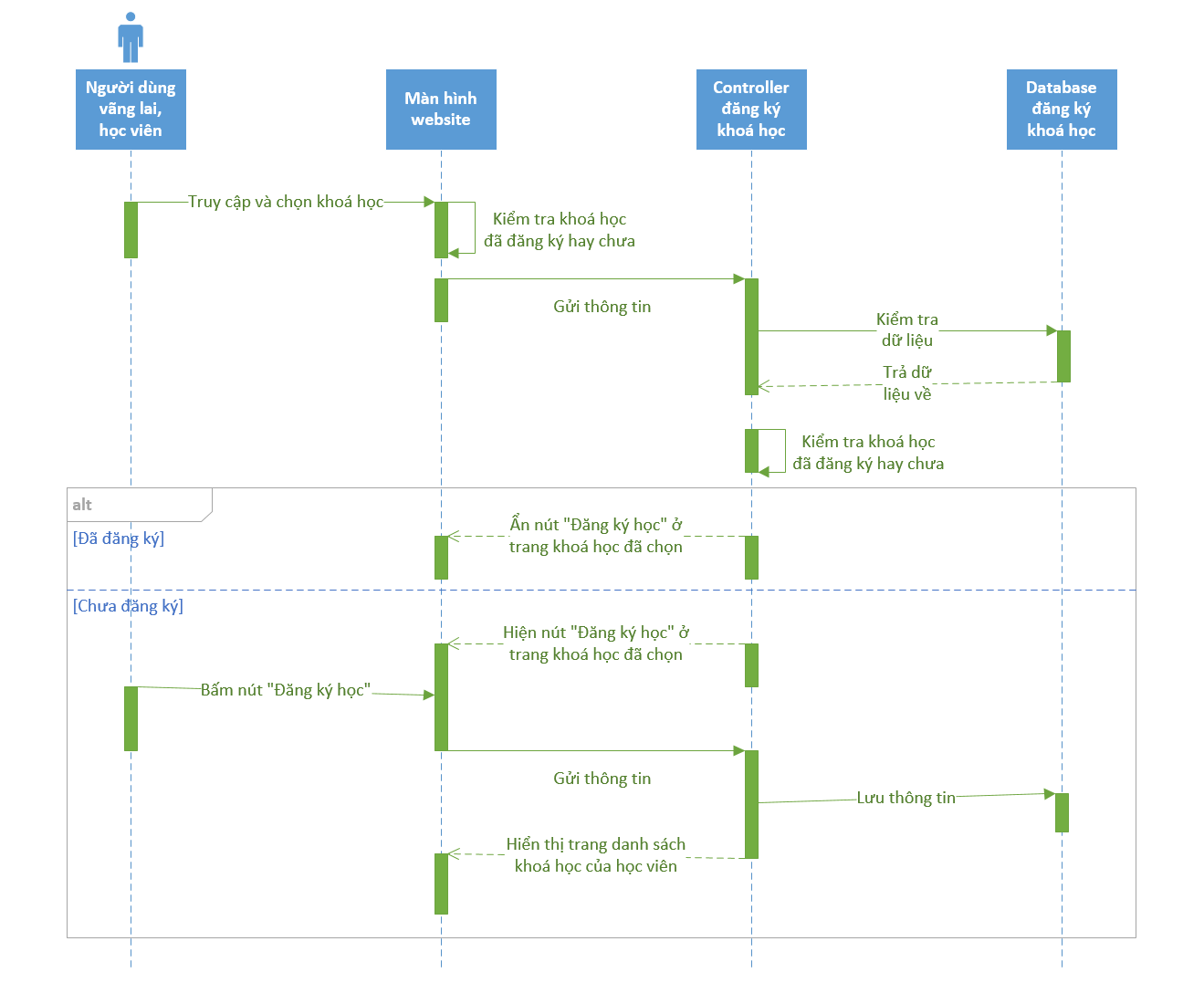
1. **Đăng ký**



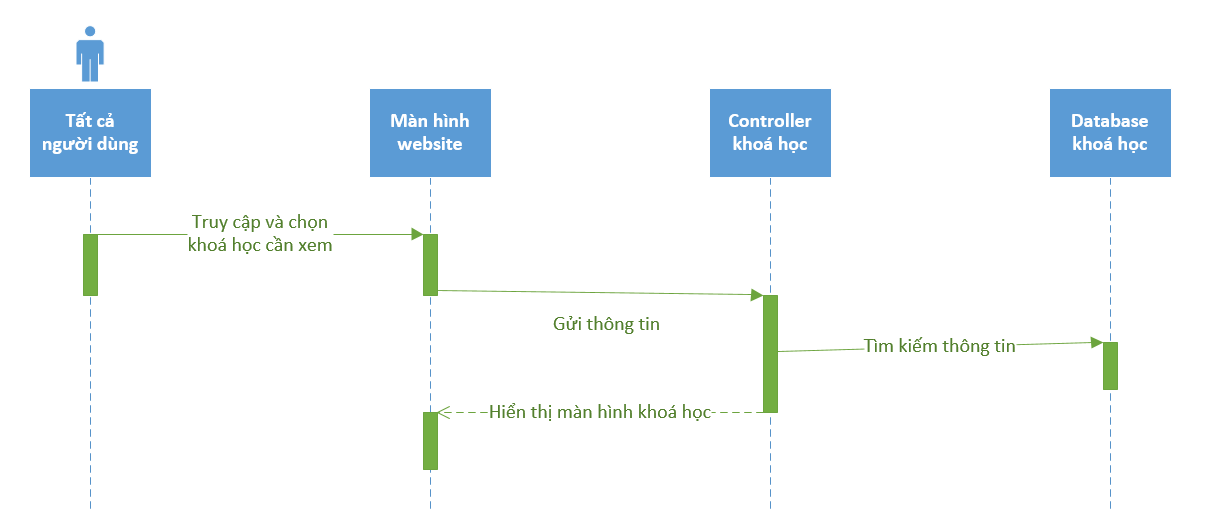
1. **Gửi thông tin liên hệ**



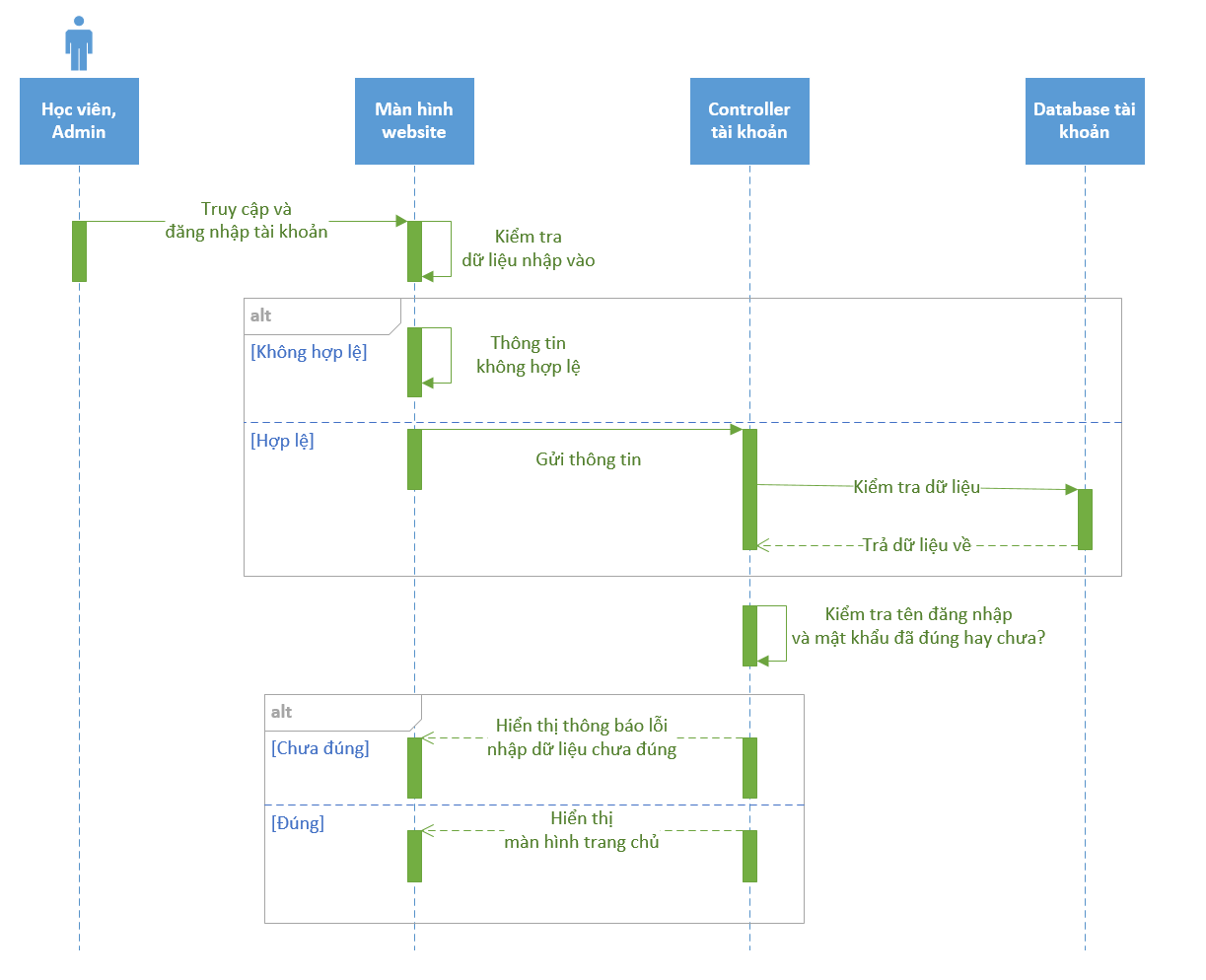
1. **Đăng ký khoá học**



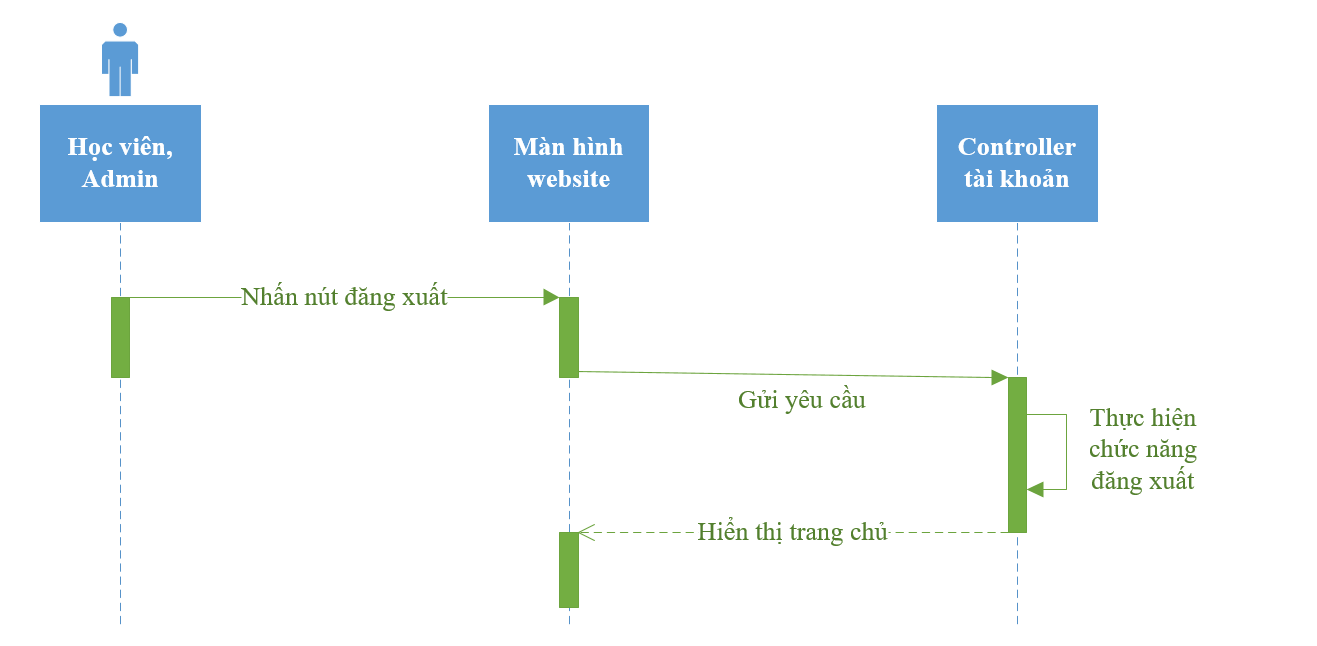
1. **Xem khoá học**



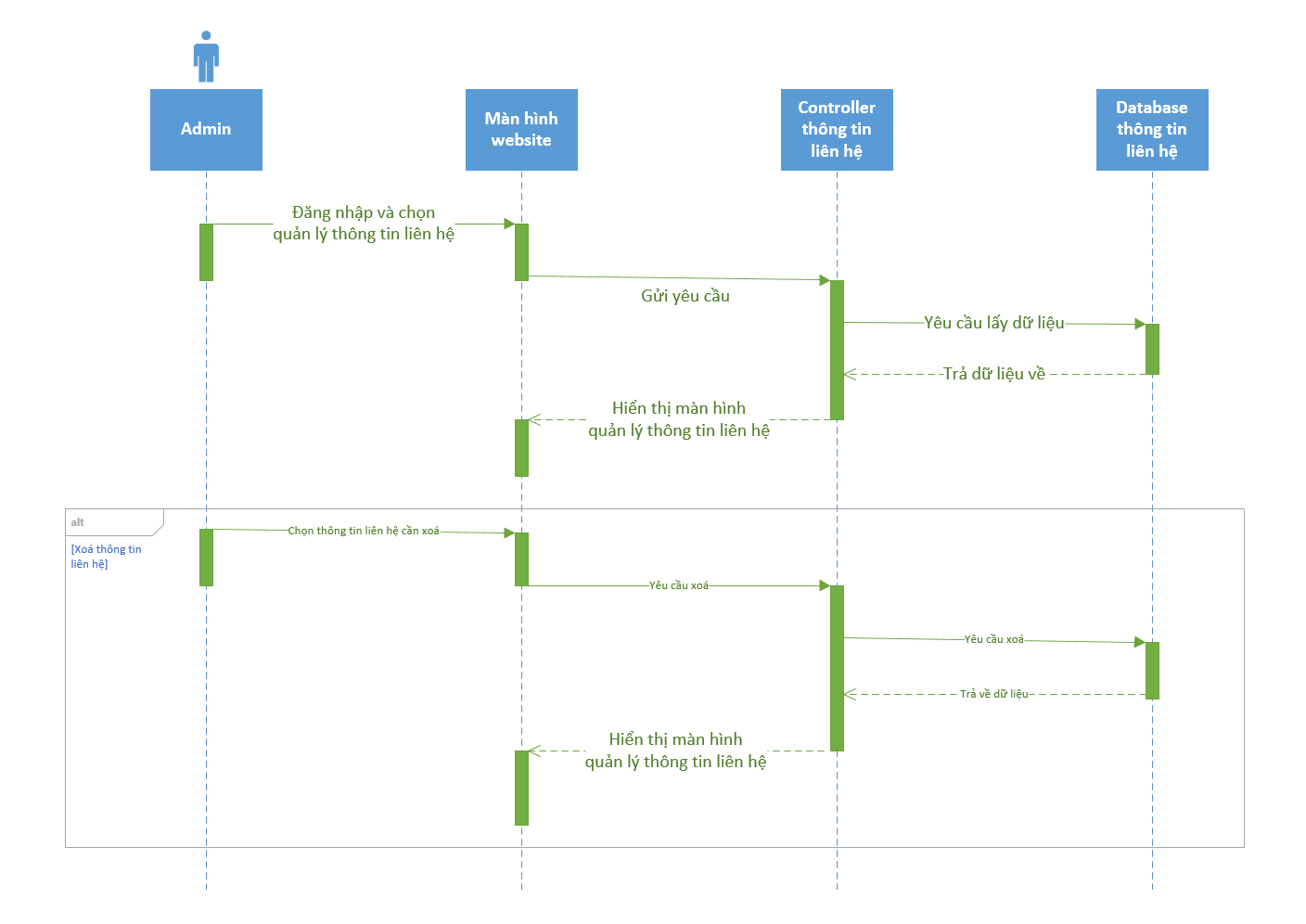
1. **Đăng nhập**



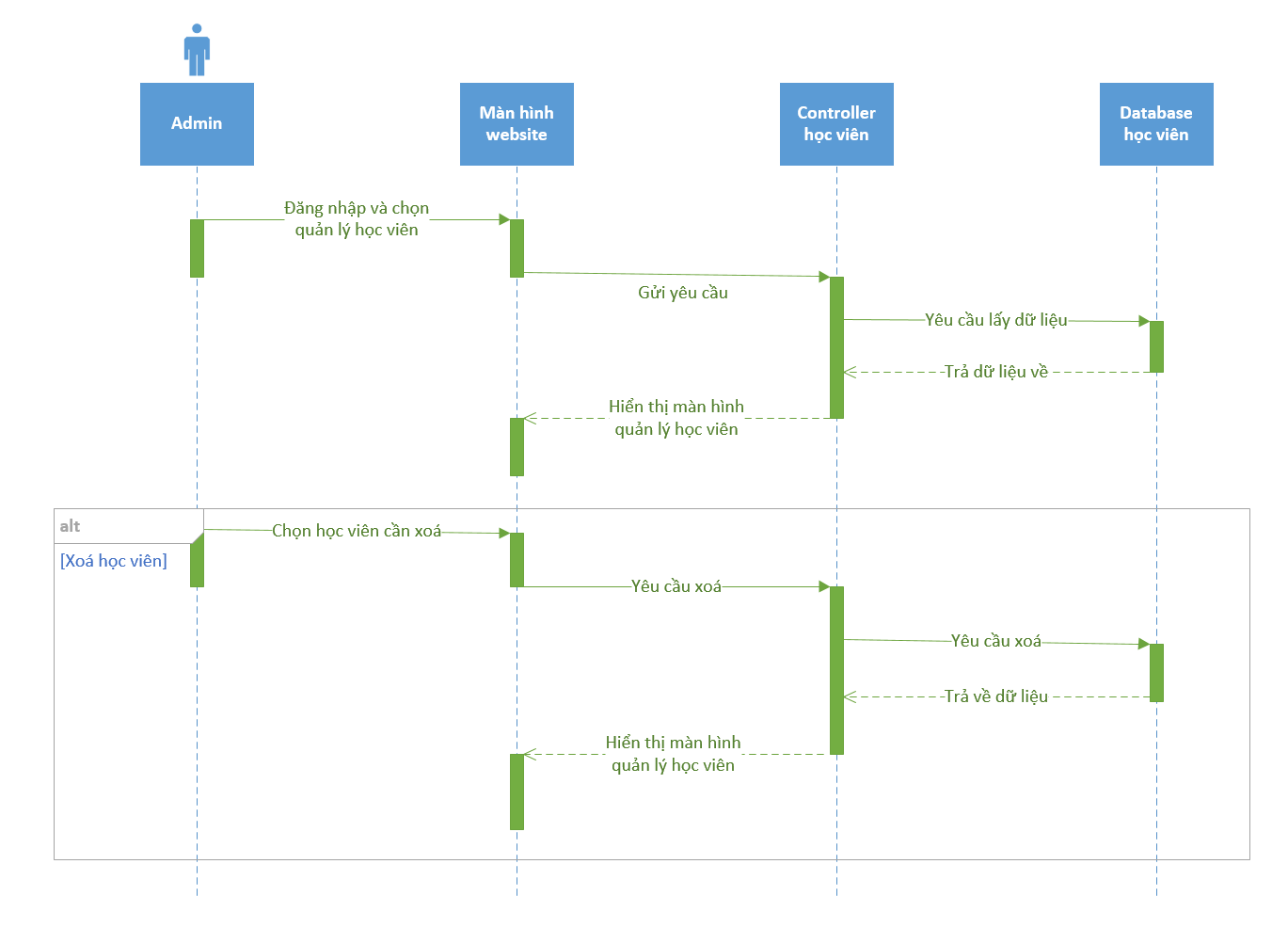
1. **Đăng xuất**



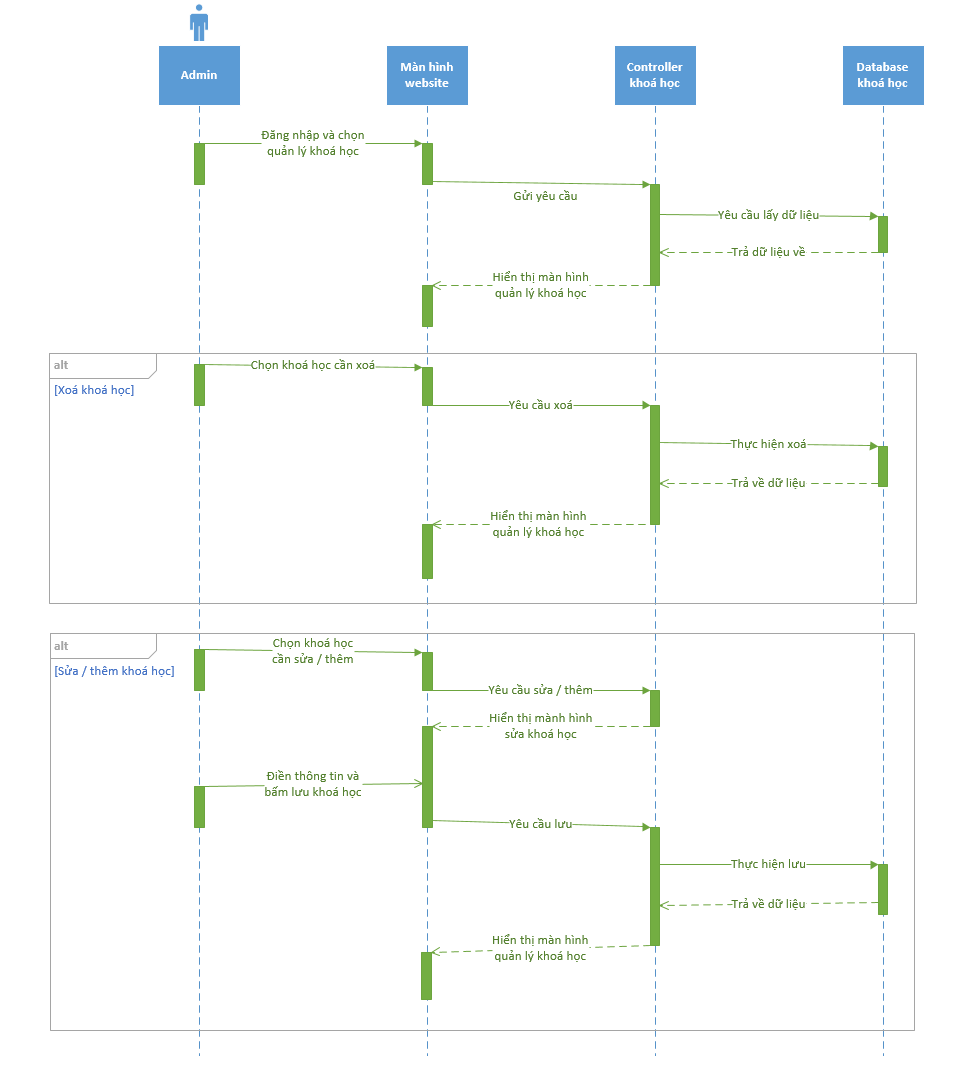
1. **Quản lý thông tin liên hệ**



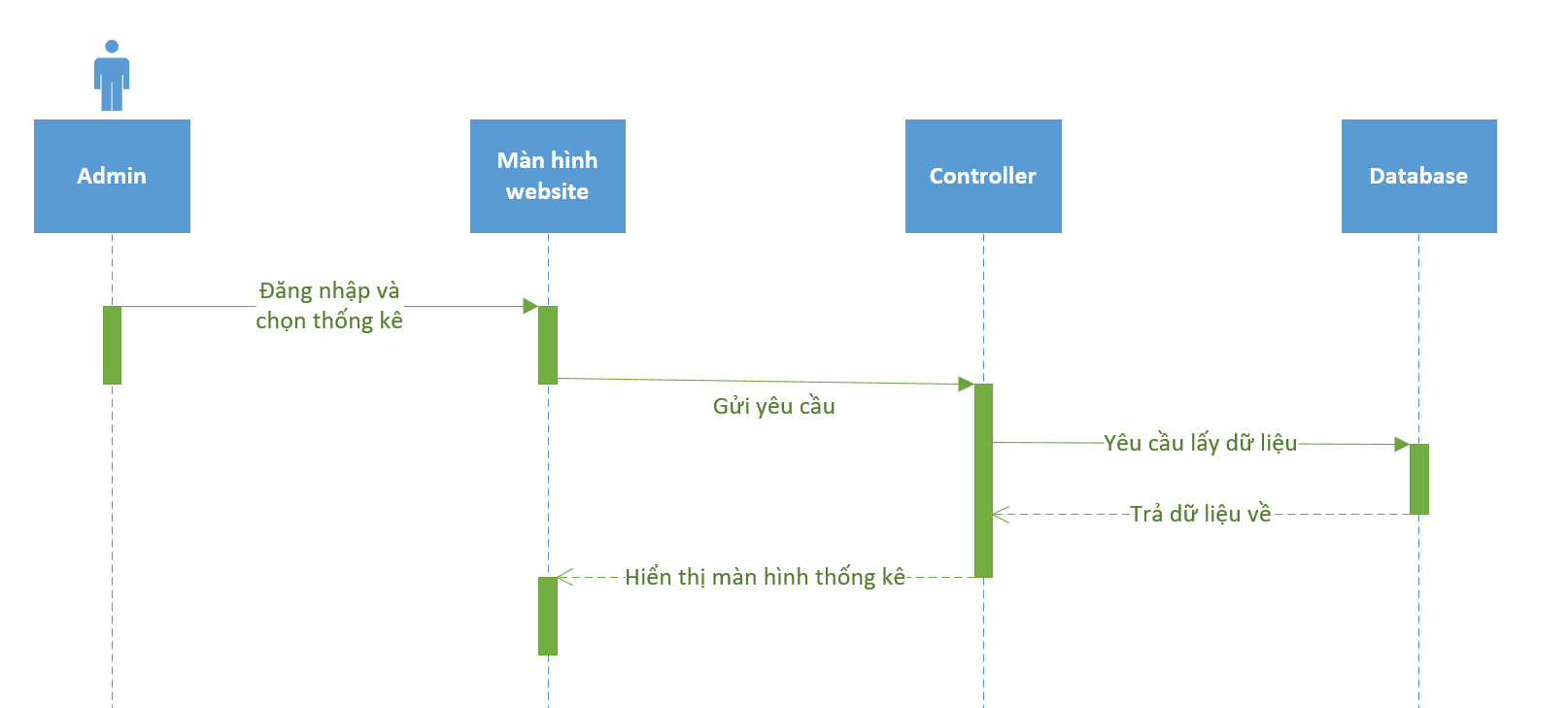
1. **Quản lý học viên**



1. **Quản lý khoá học**



1. **Thống kê**

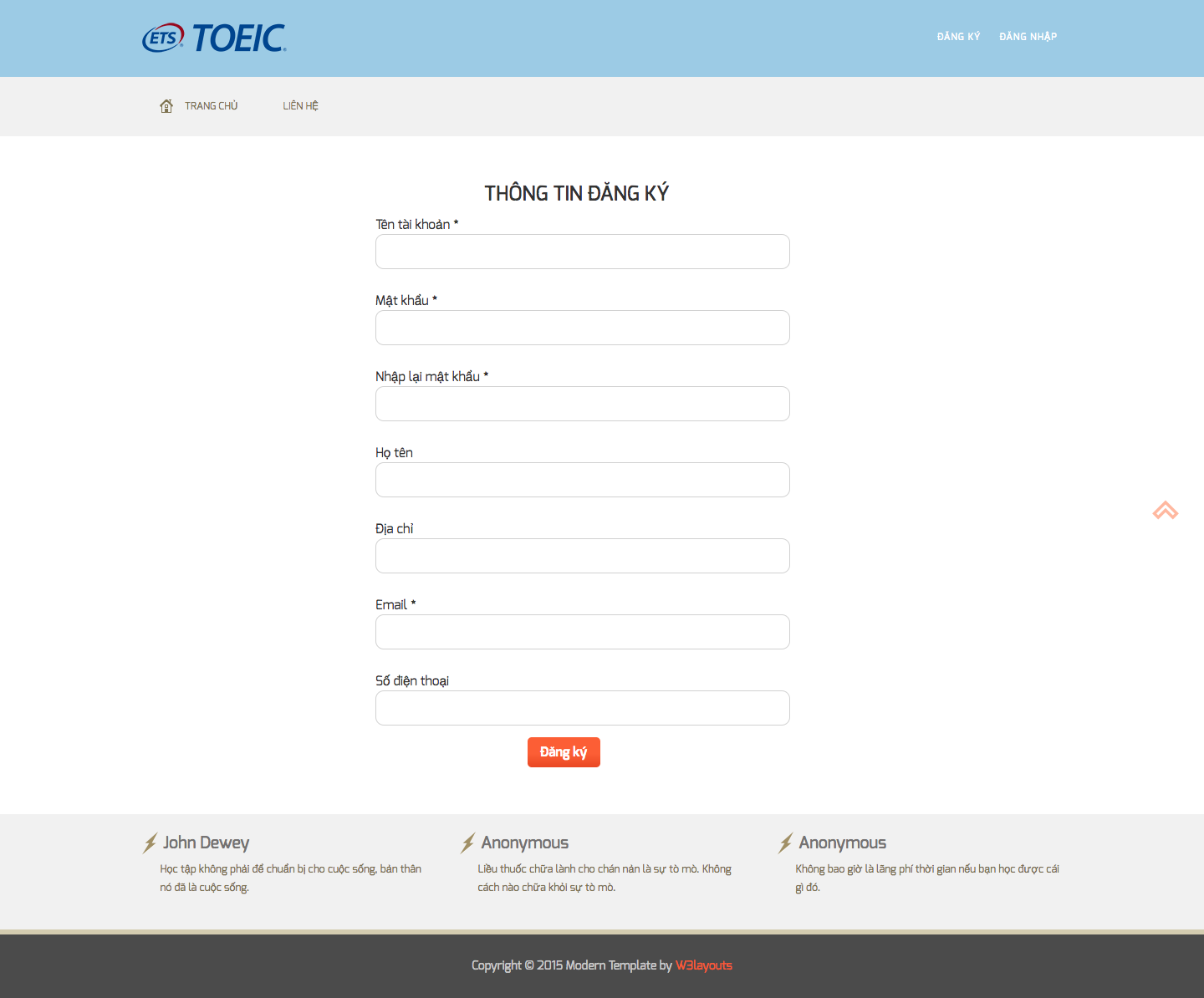


## 3.4 Giao diện ứng dụng

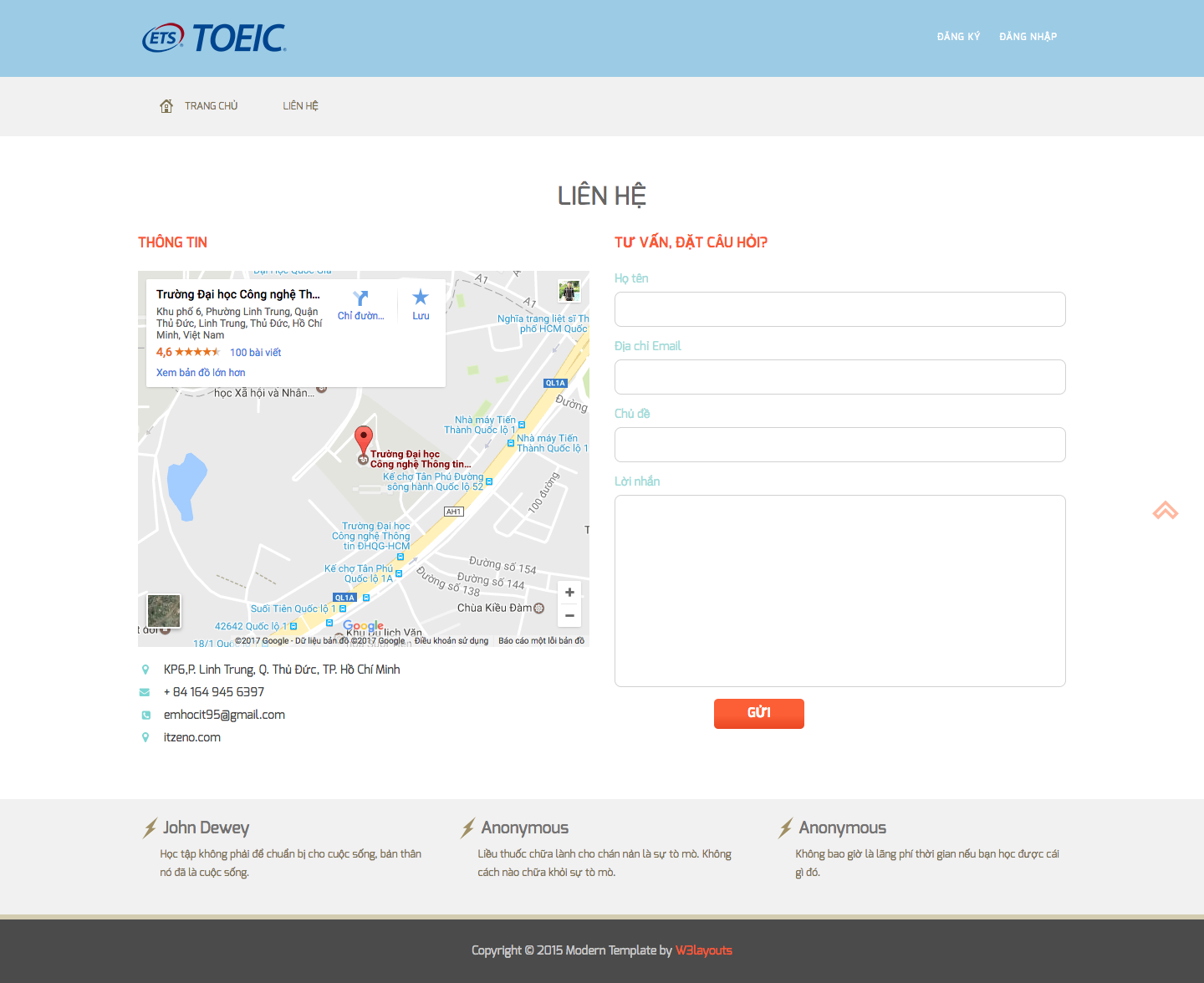
### 3.4.1 Trang chủ



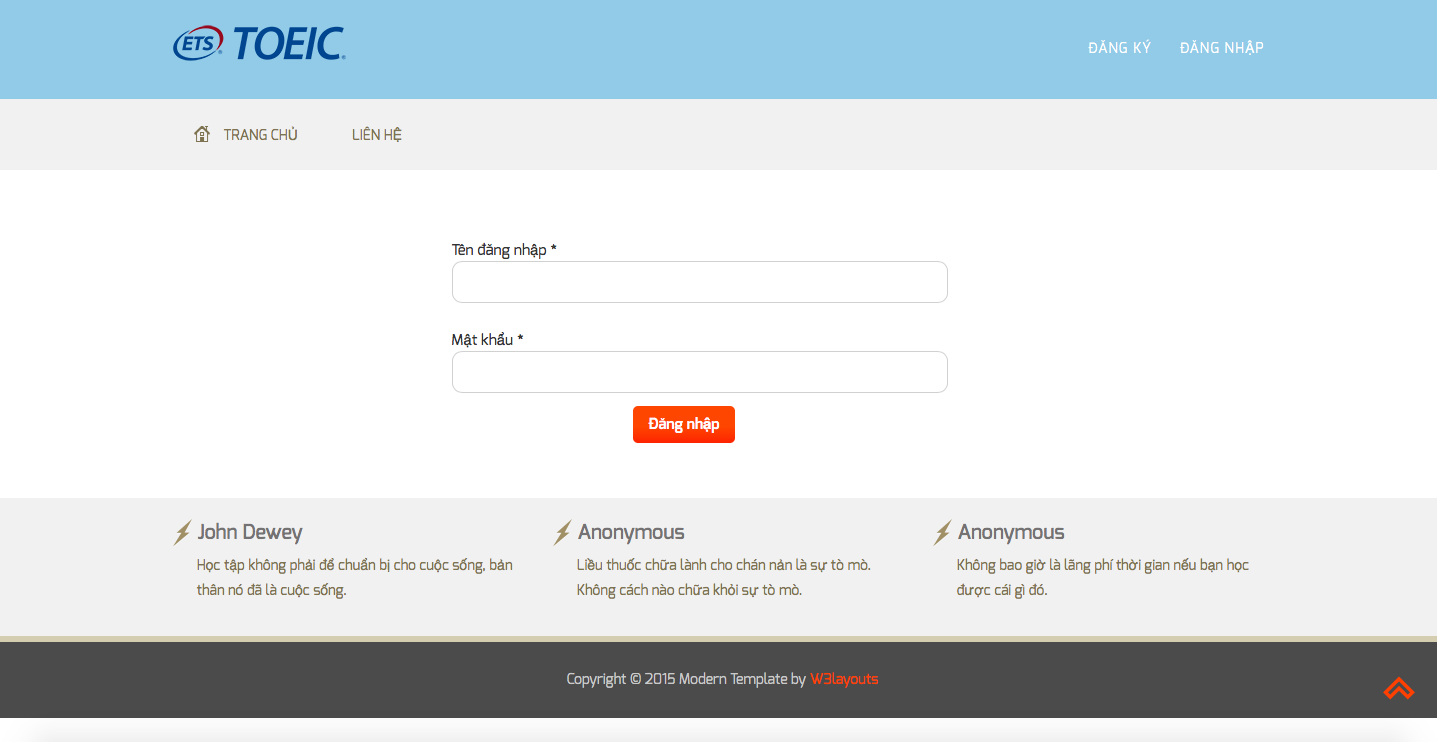
### 3.4.2 Đăng kí



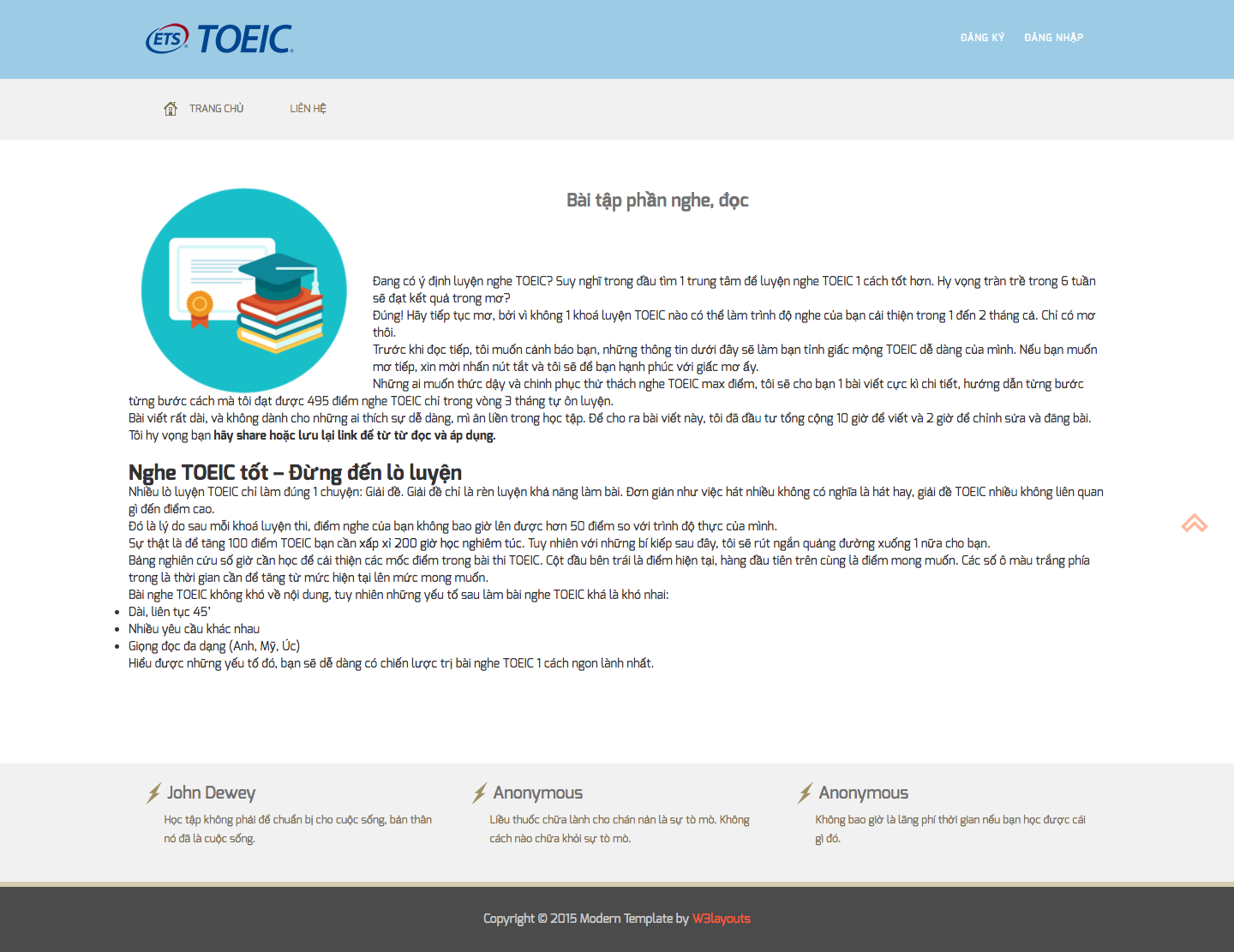
### 3.4.3 Liên hệ



### 3.4.4 Đăng nhập



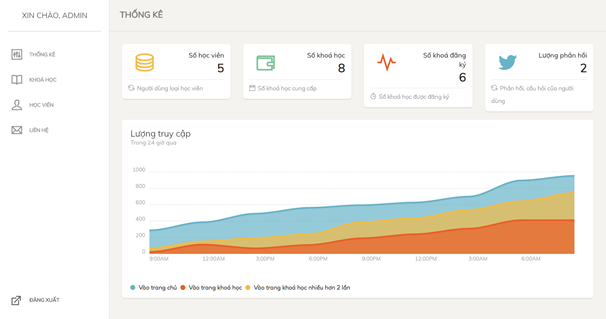
### 3.4.5 Xem chi tiết khoá học



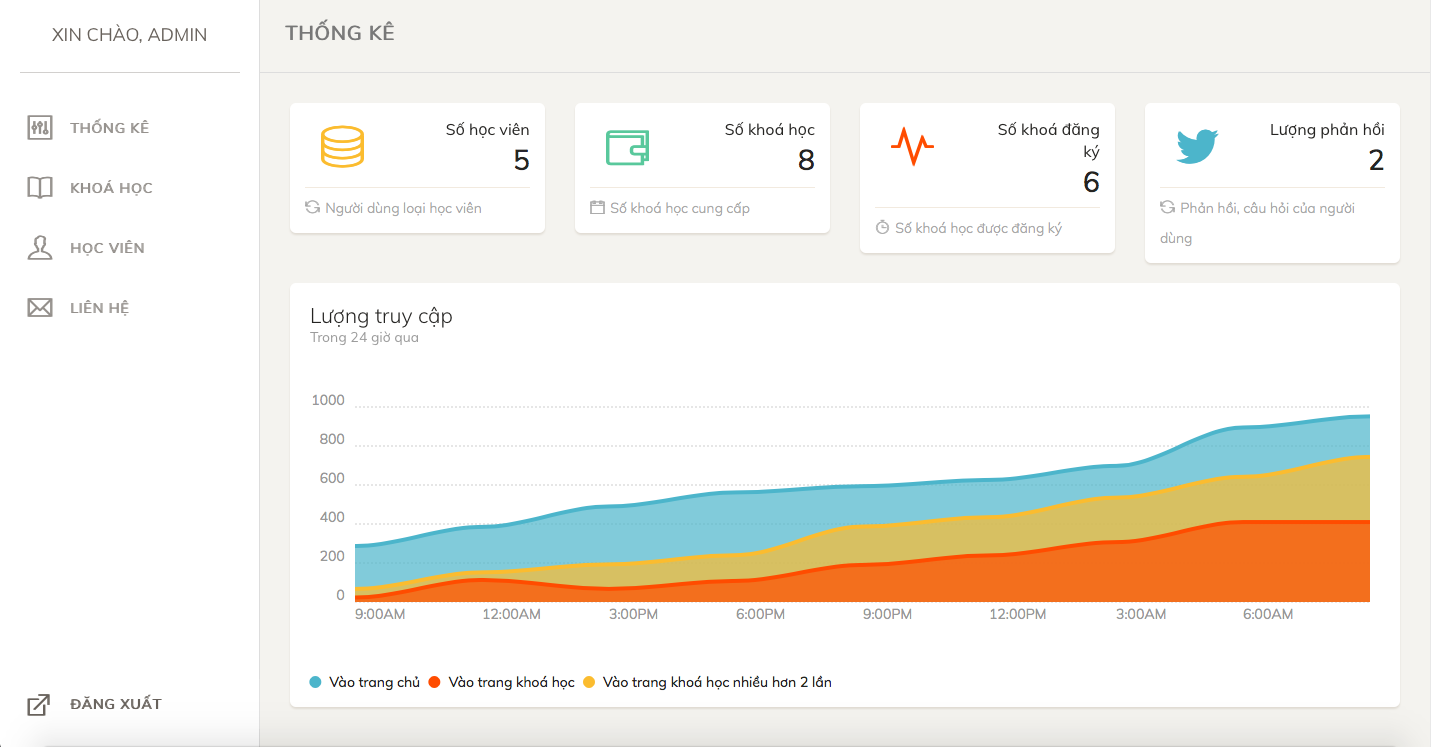
### 3.4.6 Xem khóa học đã đăng kí



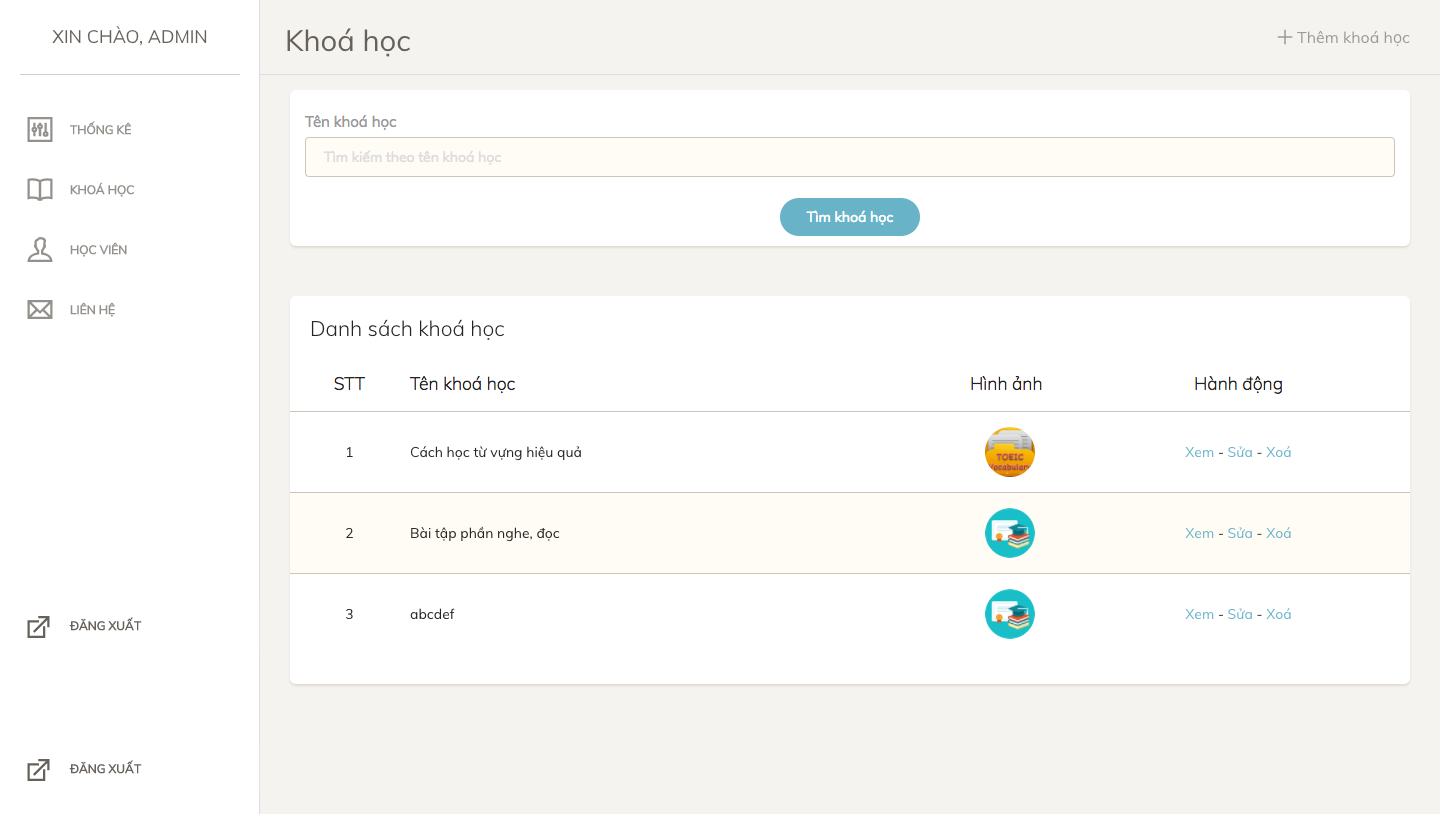
### 3.4.6 Trang Admin



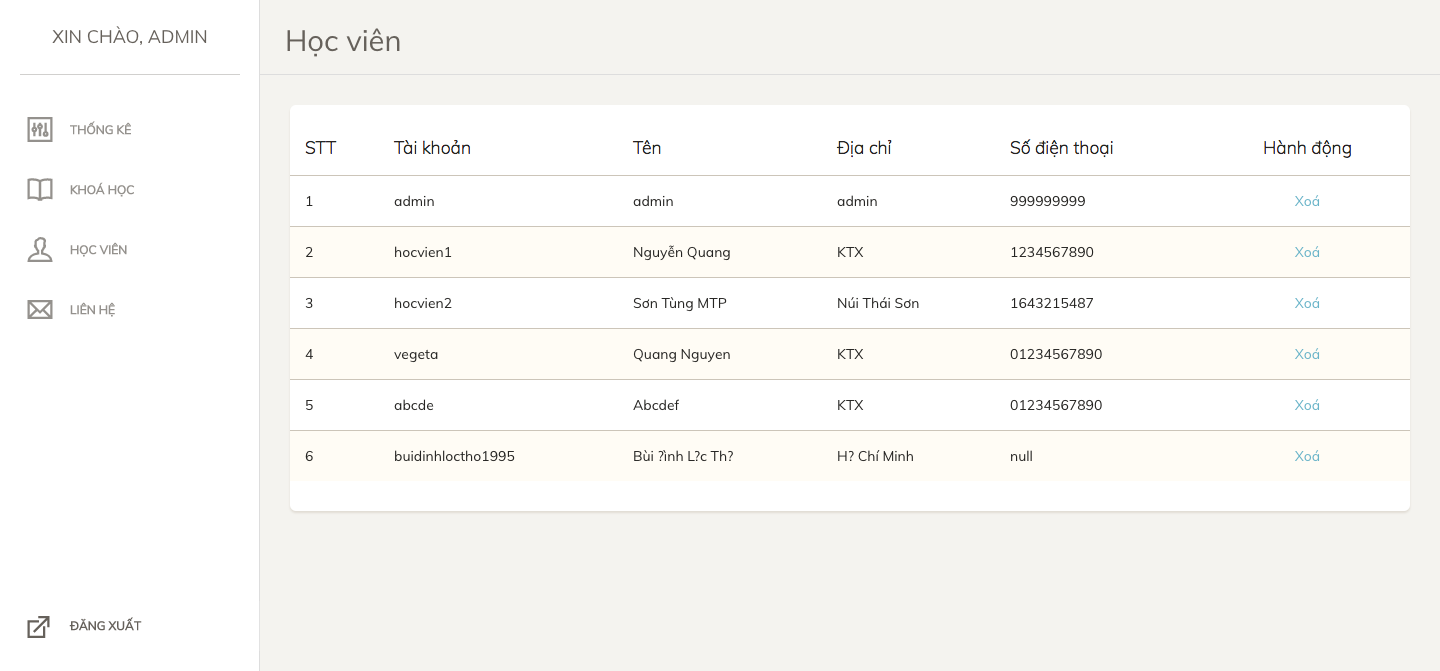
### 3.4.7 Trang Admin – Thống kê



### 3.4.8 Trang Admin – Khóa học



### 3.4.9 Trang Admin – Học viên



### 3.4.10 Trang Admin – Liên hệ

