

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC
NGÀNH CÔNG NGHỆ THÔNG TIN**

**Đề tài
XÂY DỰNG ỨNG DỤNG NGHE NHẠC TRỰC TUYẾN
VỚI ANGULAR, MONGODB, ASP.NET CORE VÀ
SIGNALR**

SINH VIÊN THỰC HIỆN:

Nguyễn Trung Hiếu

MSSV: B1401044

Lớp: Tin học ứng dụng

Khóa: 40

Cần Thơ – 11/ 2018

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
BỘ MÔN TIN HỌC ỨNG DỤNG**

**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC
NGÀNH CÔNG NGHỆ THÔNG TIN**

**Đề tài
XÂY DỰNG ỨNG DỤNG NGHE NHẠC TRỰC TUYẾN
VỚI ANGULAR, MONGODB, ASP.NET CORE VÀ
SIGNALR**

**GIÁO VIÊN HƯỚNG DẪN:
ThS. GVC. Nguyễn Đức Khoa**

**SINH VIÊN THỰC HIỆN:
Nguyễn Trung Hiếu
MSSV: B1401044
Lớp: Tin học ứng dụng
Khóa: 40**

Cần Thơ – 11/ 2018

LỜI NÓI ĐẦU

Với dự án xây dựng trang web âm nhạc tiên tiến tại Việt Nam, bản thân em đã cố gắng rất nhiều để có thể tìm hiểu nhiều công nghệ mới nhất như ASP.NET Core, Angular. Bên cạnh tìm hiểu chúng, trong quá trình xây dựng, em đã luôn cập nhật những bản vá mới nhất có thể từ các công nghệ.

Những sự tỉ mỉ và cẩn thận trên không đâu khác chính là nhờ ngôi trường này, các thầy cô đã hướng dẫn em rất nhiều, em luôn chú ý lắng nghe những chia sẻ rất tuyệt vời từ thầy cô, vì thế, em rất tự tin vì mình đã có thể áp dụng những điều đó và tạo ra thành quả này, và cuối cùng, em đã có thể tốt nghiệp rồi.

Em rất cảm ơn thầy Nguyễn Đức Khoa đã hướng dẫn cho em hoàn tất luận văn này, cũng như những góp ý của thầy cho luận văn.

Em xin cảm ơn các thầy cô khoa Công Nghệ Thông Tin và Truyền Thông trường Đại Học Cần Thơ, và thầy cô bộ môn Tin Học Ứng Dụng nói riêng, đã cho em một môi trường học tập trong những năm vừa qua vô cùng tuyệt vời và thuận lợi để phát triển chính bản thân mình.

Cảm ơn ngôi trường Đại Học Cần Thơ đã mang lại cho em rất nhiều điều bổ ích.

Chúc quý thầy cô luôn vui vẻ và tràn đầy sức khỏe.

Nguyễn Trung Hiếu

[illegible]

Mục lục

ĐÁNH GIÁ CỦA GIÁO VIÊN HƯỚNG DẪN
DANH MỤC KÝ HIỆU, CHỮ VIẾT TẮT
DANH MỤC HÌNH ẢNH
DANH MỤC BẢNG
TÓM TẮT
ABSTRACTION.....
CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN	1
1.1. Tổng quan.....	1
1.1.1. Đặt vấn đề	1
1.1.2. Tình hình hiện nay.....	1
1.2. Ý tưởng.....	3
1.3. Mục tiêu đề tài	4
1.4. Công cụ.....	5
1.5. Hướng giải quyết.....	5
1.6. Bố cục	8
Phần 1: Cơ sở lý thuyết	9
CHƯƠNG 2: MONGODB	9
2.1. NoSQL.....	9
2.1.1. Giới thiệu.....	9
2.1.2. Ưu điểm của NoSQL	9
2.1.3. Cách thức lưu trữ	9
2.1.4. Tính năng của NoSQL	10
2.2. MongoDB	10
2.2.1. Tại sao nên dùng MongoDB?	10
2.2.2. Mục đích sử dụng MongoDB.....	10
2.2.3. Cài đặt	11
2.2.4. Định nghĩa	11
2.2.5. Truy vấn.....	12
2.2.6. MongoDB Compass	13
CHƯƠNG 3: ANGULAR.....	17
3.1. Giới thiệu về Angular.....	17
3.2. Mục đích sử dụng.....	17
3.3. Giới thiệu về NodeJS	18

3.4.	Cài đặt	19
3.4.1.	Cài đặt NodeJS.....	19
3.4.2.	Cài đặt Angular CLI	20
3.5.	Lập trình	20
3.5.1.	Lập trình trên môi trường mới.....	20
3.5.2.	Lập trình trên môi trường có sẵn	21
3.5.3.	Công cụ soạn thảo.....	22
3.5.4.	Các cú pháp dòng lệnh	23
3.5.5.	Các thành phần chính.....	23
3.5.6.	Các tính năng chính.....	30
CHƯƠNG 4: ASP.NET CORE VÀ MONGODB DRIVER		36
4.1.	Giới thiệu.....	36
4.2.	Vai trò.....	36
4.3.	Môi trường lập trình	36
4.3.1.	Cài đặt	37
4.3.2.	Tạo ứng dụng Web ASP.NET Core.....	37
4.4.	ASP.NET Core WebAPI cơ bản.....	39
4.4.1.	Cấu trúc	39
4.4.2.	Cú pháp	39
4.4.3.	Khởi chạy.....	42
4.5.	Sử dụng NuGet Package.....	42
4.6.	MongoDB Driver trong ASP.NET Core	43
4.6.1.	Model	43
4.6.2.	Database Context.....	44
4.6.3.	Repository	45
4.7.	Dependency Injection trong ASP.NET Core.....	46
4.7.1.	Khái niệm Inversion of Control.....	46
4.7.2.	Khái niệm Dependency Injection	46
4.7.3.	Sử dụng Dependency trong ASP.NET Core	46
CHƯƠNG 5: SIGNALR.....		49
5.1.	SignalR là gì?.....	49
5.1.1.	Giới thiệu.....	49
5.1.2.	SignalR trong ASP.NET Core	50
5.1.3.	Mục đích sử dụng SignalR	50

5.1.4.	Thuật ngữ Middleware.....	50
5.2.	Cài đặt	51
5.2.1.	DotNET Client	51
5.2.2.	Javascript Client.....	51
5.3.	Ứng dụng.....	52
5.3.1.	Thiết lập lắng nghe	52
5.3.2.	Thiết lập kết nối.....	53
5.3.3.	Giao tiếp.....	54
Phần 2:	Áp dụng.....	56
CHƯƠNG 6:	GIẢI QUYẾT VẤN ĐỀ	56
6.1.	Đặt tả vấn đề	56
6.2.	Thiết kế.....	57
6.2.1.	Mô hình Use-Case	57
6.2.2.	Xây dựng cơ sở dữ liệu.....	61
6.2.3.	Mô hình cơ sở dữ liệu quan hệ	66
6.2.4.	Mô hình lưu trữ MongoDB.....	67
6.2.5.	Sơ đồ phân rã chức năng	68
6.2.6.	Dòng dữ liệu.....	69
6.3.	Thiết kế cơ bản.....	71
6.3.1.	Mô hình hoạt động.....	71
6.3.2.	Thiết kế mô hình lớp	72
6.4.	Thiết kế quy trình	74
6.4.1.	Chức năng đăng ký	74
6.4.2.	Chức năng đăng nhập	75
6.4.3.	Chức năng tải nhạc.....	76
6.4.4.	Chức năng xóa nhạc	77
CHƯƠNG 7:	TỔNG KẾT	79
7.1.	Kết quả đạt được	79
7.2.	Các chức năng nổi bật.....	79
7.2.1.	Trang chủ.....	79
7.2.2.	Trang quản lý.....	79
7.2.3.	Phòng trò chuyện	82
7.3.	Đánh giá	84
7.3.1.	Những gì đạt được	84

7.3.2.	Những gì cần cải thiện.....	84
7.3.3.	Hướng phát triển.....	84
PHỤ LỤC		85
TÀI LIỆU THAM KHẢO.....		97
DẪN NGUỒN.....		99

DANH MỤC KÝ HIỆU, CHỮ VIẾT TẮT

Ký hiệu, chữ viết	Biểu thị	Ý nghĩa
HTTP	HyperText Transfer Protocol	Giao thức truyền tải siêu văn bản, là giao thức dùng ở các trang web dùng truyền tải html
Web	Website	Trang web
HTML	Hyper Text Markup Language	Ngôn ngữ đánh dấu siêu văn bản, loại ngôn ngữ định dạng giao diện cho các ứng dụng Web.
CSS	Cascading Style Sheets	Tập định kiểu theo tầng, miêu tả cách định kiểu trong HTML và XHTML
JS	Javascript	Ngôn ngữ lập trình kịch bản
AJAX	Asynchronous Javascript and XML	Là công nghệ chạy bất đồng bộ trong Javascript
API	Application Programming Interface	Giao diện lập trình ứng dụng, cung cấp các phép dịch vụ thực hiện yêu cầu.
DB	Database	Cơ sở dữ liệu
SQL	Structured Query Language	Ngôn ngữ truy vấn mang tính cấu trúc (cơ sở dữ liệu quan hệ)
NoSQL	Non Structured Query Language	Ngôn ngữ truy vấn phi cấu trúc (phi quan hệ)
MVC	Model – View – Controller	Mô hình thiết kế theo Mô hình – Khung nhìn – Bộ điều khiển.
JSON	JavaScript Object Notation	Ký pháp kiểu đối tượng trong Javascript, một kiểu dữ liệu theo dạng cặp thuộc tính – giá trị.
BSOJ	Binary JavaScript Object Notation	Là kiểu dữ liệu có cấu trúc Json nhưng được lưu ở dạng nhị phân (Binary)
App	Application	Ứng dụng
.NET	Dot Net	Nền tảng Runtime của Microsoft, bao gồm .NET Framework và .NET Core
IoC	Inversion of Control	Thuật ngữ chỉ sự thiết kế chương trình cho phép người dùng gọi ngược (callback) các sự kiện của một điều khiển.
DI	Dependency Injection	Thuật ngữ chỉ việc dung nạp các thành phần phụ thuộc thay vì khởi

		tạo sự phụ thuộc đó ngay trong một đối tượng.
JWT	JSON Web Token	Là một token dùng trong xác thực, nó là dạng mã hóa chứa sẵn các quyền và tính định danh của một người dùng.
DAO	Data Access Object	Lớp truy cập dữ liệu, dùng trong code-behind để truy vấn cơ sở dữ liệu

DANH MỤC HÌNH ẢNH

Chương 1:

Hình 1. 1. Tốc độ tải trang chưa được tối ưu (mp3.zing.vn).....	2
Hình 1. 2. Thị phần YouTube quá đồi lớn mạnh (nguồn: Youtube.com)	3
Hình 1. 3. Tính năng partial load của Youtube	3
Hình 1. 4. Biểu tượng Amnhac.com	4
Hình 1. 5. Thống kê số lượng request đáp ứng mỗi giây (nguồn: https://stackify.com/asp-net-core-features/)	5
Hình 1. 6. So sánh MongoDB và các cơ sở dữ liệu tốt nhất hiện nay (nguồn: ArangoDB)	6
Hình 1. 7. Mô hình sơ lược thiết kế ứng dụng tổng quát	7

Chương 2:

Hình 2. 1. So sánh NoSQL và SQL (nguồn: TechBlog)	9
Hình 2. 2. (nguồn: MongoDB)	10
Hình 2. 3. Tải về tại mongodb.com	11
Hình 2. 4. Giao diện dòng lệnh cho Mongo	12
Hình 2. 5. Giao diện kết nối cơ sở dữ liệu trong MongoDB Compass.....	14
Hình 2. 6. Giao diện sau khi kết nối máy chủ thành công.....	14
Hình 2. 7. Nhấp chọn một Database để xem hoặc thêm các Collection bên trong	15
Hình 2. 8. Bên trong một Collection sẽ được hỗ trợ Query, Thêm, sửa, xóa	15
Hình 2. 9. Khi chỉnh sửa có thể thay đổi dữ liệu, kiểu dữ liệu và xóa dữ liệu	16

Chương 3:

Hình 3. 1. Biểu tượng của Angular (nguồn: Angular.io)	17
Hình 3. 2. Tổng quan mô hình MVC của Angular (nguồn: Codeburst.io).....	18
Hình 3. 3. Biểu tượng NodeJS (nguồn: NodeJS.org).....	18
Hình 3. 4. Giao diện trình cài đặt Node.js trên Windows	19
Hình 3. 5. Kiểm tra phiên bản Node qua Command Prompt	20
Hình 3. 6. Quy trình tạo ứng dụng Angular đầu tiên.....	21
Hình 3. 7. Các tập tin và thư mục của ứng dụng Angular	21
Hình 3. 8. Biểu tượng Visual Studio Code (nguồn: visualstudio.microsoft.com).....	22
Hình 3. 9. Cài đặt Extension cho Visual Studio Code	22
Hình 3. 10. Mô hình hoạt động của Angular (nguồn: Ciphertrick.com).....	24
Hình 3. 11. Cú pháp định nghĩa Component.....	25
Hình 3. 12. Cách Angular nhét (manipulate) một Component ra giao diện	26
Hình 3. 13. Cú pháp định nghĩa Module	27
Hình 3. 14. Cú pháp định nghĩa Service	28
Hình 3. 15. Một Service được sử dụng bởi nhiều trang và thành phần.....	28
Hình 3. 16. Khai báo Service trong constructor để Angular inject vào Component	28
Hình 3. 17. Khai báo các route	29
Hình 3. 18. Tạo AppModule và nạp RouterModule kèm các khai báo route, sau đó xuất RouterModule với các thiết lập trên để các Module khác nạp vào sử dụng tiếp.....	29
Hình 3. 19. AppModule nạp AppModule vừa tạo	29

Hình 3. 20. Khai báo router-outlet tại thành phần cần điều hướng trang con, ở đây là AppComponent.....	30
Hình 3. 21. Tạo biến trong một Component (class)	30
Hình 3. 22. Cú pháp in biến trong class ra ngoài template (html)	30
Hình 3. 23. Kết quả hiển thị.....	31
Hình 3. 24. Dùng ngModel để ràng buộc đa chiều	31
Hình 3. 25. Khai báo module chức năng để hỗ trợ thuộc tính ngModel	31
Hình 3. 26. Hiển thị ở giao diện.....	32
Hình 3. 27. Hiển thị sau khi thay đổi hộp thoại.....	32
Hình 3. 28. Khai báo một hàm trong Component (class)	32
Hình 3. 29. Gán hàm gọi trên như một sự kiện ở phía template.....	33
Hình 3. 30. Hiển thị lúc này ở trình duyệt	33
Hình 3. 31. Nhấp nút đã được gán sự kiện trên và xem thay đổi	33
Hình 3. 32. Template (bên phải) và hiển thị ở phía trình duyệt (bên trái).....	33
Hình 3. 33. Template (bên phải) dùng ngFor và hiển thị ở phía trình duyệt (bên trái)	34
Hình 3. 34. Giá trị của biến array ở phía Component	34
Hình 3. 35. Vòng đời của một trang Angular	35

Chương 4:

Hình 4. 2. ASP.NET Core (nguồn: asp.net).....	36
Hình 4. 3. Visual Studio (nguồn: visualstudio.microsoft.com)	37
Hình 4. 4. Tạo ứng dụng ASP.NET Core qua Visual Studio 2017	38
Hình 4. 5. Chọn Template cho ứng dụng	38
Hình 4. 6. Cấu trúc cơ bản của ứng dụng	39
Hình 4. 7. Cú pháp trong Controller của ASP.NET Core	40
Hình 4. 8. Các dòng lệnh trong tệp Startup.cs	41
Hình 4. 9. Debug ứng dụng	42
Hình 4. 10. Tạo bản dựng cho ứng dụng qua giao diện Visual Studio 2017	42
Hình 4. 11. Mở Menu để quản lý gói mở rộng NuGet	43
Hình 4. 12. MongoDB Driver trên NuGet Packages	43
Hình 4. 13. Tạo một Model đơn giản nhằm lưu nó xuống Database sau này.....	44
Hình 4. 14. Khởi tạo class DataAccess kết nối đến Mongo Database	45
Hình 4. 15. Một Repository cơ bản chứa DataAccess.....	45
Hình 4. 16. Gọi Repository ra trong Controller để truy vấn.....	46
Hình 4. 17. Gọi AddSingleton để tạo một Singleton của Repository khi Runtime	47
Hình 4. 18. Khai báo Repository trong Constructor của Controller để ASP.NET Core inject	47
Hình 4. 19. Áp dụng với đa hình trong C#	48
Hình 4. 20. Khai báo một đối tượng Đa hình như Singleton.....	48

Chương 5:

Hình 5. 1. Công nghệ SignalR (nguồn: microsoft.com)	49
Hình 5. 2. Giao tiếp thời gian thực giữa Client và Server (nguồn: c-sharpcorner.com).....	49
Hình 5. 3. Một Request của người dùng có thể thông qua một Middleware hoặc nhiều Middleware tùy theo thiết lập logic bên trong	51
Hình 5. 4. SignalR cho ASP.NET Core trên NuGet packages.....	51

Hình 5. 5. Một SignalR Hub cơ bản	52
Hình 5. 6. Thêm SignalR như một phần của Service trong ASP.NET Core	52
Hình 5. 7. Tạo một Middleware cho SignalR	53
Hình 5. 8. Yêu cầu kết nối ở phía Client (nguồn: docs.microsoft.com)	53
Hình 5. 9. Thiết đặt lắng nghe sự kiện tên ReceiveMessage ở phía Client (nguồn: docs.microsoft.com)	54
Hình 5. 10. Câu lệnh gửi tin nhắn đến máy tất cả các máy con đang lắng nghe hàm ReceiveMessage	54
Hình 5. 11. Gọi lệnh trên Client có tên SendMessage ở phía máy Server và chờ kết quả trả về.....	54

Chương 6:

Hình 6. 1. Mô hình Use Case mức cơ bản	57
Hình 6. 2. Use Case mức trung ở phía Khách hàng	58
Hình 6. 3. Use case mức trung đối với quản trị viên	59
Hình 6. 4. Use case chi tiết Quản lý danh sách phát phía người dùng	60
Hình 6. 5. Use case chi tiết cho chức năng quản lý bài hát của Quản trị viên	60
Hình 6. 6. Mô hình dữ liệu vật lý	66
Hình 6. 7. Mô hình lưu trữ trong MongoDB	67
Hình 6. 8. Sơ đồ phân rã chức năng tổng quát	68
Hình 6. 9. Sơ đồ dòng dữ liệu tổng quan	69
Hình 6. 10. Sơ đồ dòng dữ liệu mức người dùng	69
Hình 6. 11. Sơ đồ dòng dữ liệu ở quản trị viên	70
Hình 6. 12. Dòng dữ liệu khi tải một bài hát lên (mức 2)	70
Hình 6. 13. Mô hình hoạt động của Amnhac.com	71
Hình 6. 14. Mô hình lớp cho phần Quản lý bài hát	72
Hình 6. 15. Mô hình lớp cho phần Tài khoản người dùng	73
Hình 6. 16. Tuần tự quy trình khi người dùng đăng ký trên ứng dụng	74
Hình 6. 17. Quy trình đăng nhập của người dùng	75
Hình 6. 18. Quy trình tải nhạc lên máy chủ	76
Hình 6. 19. Quy trình xóa một bài hát trong Amnhac.com (1)	77
Hình 6. 20. Quy trình xóa một bài hát trong Amnhac.com (2)	77

Chương 7:

Hình 7. 1. Giao diện trang chủ Amnhac.com	79
Hình 7. 2. Vào mục đăng nhập ở trên thanh tìm kiếm	79
Hình 7. 3. Dùng tài khoản đăng nhập hoặc đăng ký mới	80
Hình 7. 4. Từ thanh tìm kiếm, vào mục quản lý	80
Hình 7. 5. Giao diện quản lý bài hát	80
Hình 7. 6. Thử tải một bài hát và chọn bài hát vừa tải để duyệt	81
Hình 7. 7. Bật máy chơi nhạc ở biểu tượng góc dưới màn hình và vào danh sách phát	81
Hình 7. 8. Giao diện chơi nhạc khi phát một bài hát	81
Hình 7. 9. Về trang chủ và vào mục Trò chuyện	82
Hình 7. 10. Vào một phòng hoặc tạo phòng mới (khi là quản trị viên)	82
Hình 7. 11. Xem danh mục trò chuyện	83
Hình 7. 12. Thiết lập appsettings.json phía ASP.NET Core server	86

Hình 7. 13. Mở thao mục PowerShell	86
Hình 7. 14. Gõ lệnh dotnet build để tạo bản dựng	86
Hình 7. 15. Chép các tệp cần thiết vào thư mục chứa bản dựng.....	87
Hình 7. 16. Chạy thử máy chủ	87
Hình 7. 17. Cài đặt các thành phần còn thiếu cho Angular Web Application	87
Hình 7. 18. Tạo bản dựng cho việc triển khai.....	87
Hình 7. 19. Giao diện trang chủ sau khi triển khai	88

Phụ lục

Hình 1	89
Hình 2	89
Hình 3	90
Hình 4	90
Hình 5	91
Hình 6	91
Hình 7	91
Hình 8	92
Hình 9	92
Hình 10	93
Hình 11	93
Hình 12. Khởi tạo thiết lập cho Jwt Bearer Factory	93
Hình 13. Inject các option vào Factory	93
Hình 14. Tạo các Claim và nhờ Jwt mã hóa chúng	94
Hình 15. Token được mã hóa gửi về Client.....	94
Hình 16. Đặt thẻ chú thích Authorize để yêu cầu gửi kèm Token mỗi yêu cầu.....	94
Hình 17. Token được gửi kèm trong Header để xác thực.....	94
Hình 18	95
Hình 19	95
Hình 20	95
Hình 21	96

DANH MỤC BẢNG

Chương 2:

Bảng 2. 1. Định nghĩa thuật ngữ trong MongoDB.....	12
Bảng 2. 2. Các câu lệnh đơn giản trong MongoDB	13

Chương 3:

Bảng 3. 1. Cú pháp dòng lệnh hay sử dụng trong Angular CLI.....	23
---	----

Chương 6:

Bảng 6. 1. Thực thể Permission.....	61
Bảng 6. 2. Thực thể người dùng.....	61
Bảng 6. 3. Thực thể lưu trữ danh sách quốc gia	62
Bảng 6. 4. Thực thể lưu trữ nghệ sĩ.....	62
Bảng 6. 5. Thực thể lưu thể loại nhạc	63
Bảng 6. 6. Thực thể phân loại thể loại nhạc.....	63
Bảng 6. 7. Thực thể đường dẫn nhạc.....	63
Bảng 6. 8. Thực thể lưu trữ bài hát.....	64
Bảng 6. 9. Thực thể lưu trữ Album	65
Bảng 6. 10. Thực thể lưu lịch sử lượt nghe.....	65

TÓM TẮT

Với nhu cầu phát triển không ngừng về mọi mặt trong xã hội, đặc biệt trong thời đại 4.0, nhu cầu giải trí càng tăng cao. Do đó, với dự án xây dựng trang web âm nhạc, Amnhac.com, một trang web hướng về nhu cầu giải trí hằng ngày của mọi người: Âm nhạc.

Thị trường âm nhạc trong nước hiện rất ít xem trọng bản quyền âm nhạc, không chú trọng cầu nối giữa nghệ sĩ và công chúng, không coi trọng bảo mật thông tin người dùng, và cũng không chú ý đến sức mạnh thật sự của nền âm nhạc Việt Nam.

Hàn Quốc, Mỹ, Anh, Trung Quốc, Nhật Bản, đều cho mọi người thấy sức mạnh của âm nhạc, nó không chỉ là cầu nối giữa người với người, gỡ bỏ rào cản ngôn ngữ, mà còn là phản ánh sự phát triển con người, và còn là doanh thu khổng lồ từ chính thị trường này.

Amnhac.com sẽ đánh thức lại thị trường âm nhạc tại Việt Nam. Amnhac.com với hi vọng sẽ dần trở thành cầu nối âm nhạc trong nước và ngoài nước, là cái công tắc đánh thức tiềm năng của các tài năng âm nhạc, cũng là nơi mà nhịp điệu sẽ giúp con người thỏa mãn nhu cầu thư giãn và giải trí một cách lành mạnh. Với công nghệ mới nhất về Web như ASP.NET Core, Angular, MongoDB, hi vọng rằng Amnhac.com sau sẽ trở thành nền tảng âm nhạc lớn nhất tại Việt Nam trong tương lai.

ABSTRACTION

This country is growing bigger, especially the need of people about entertainment in this 4.0 era. How about the music? Many people love music, a lot of people need music, that's why this project is born, Amnhac.com, a musical website which will entertain everyone that need music in their life.

In Vietnam, currently there are many websites about music, music's news. However, they mostly do not concentrate in building a bridge between the artists and the listeners. Further, the Vietnamese underestimate the music copyright, did not be careful about their information and customers' information, and did not really focus on the development of the music market in Vietnam.

As you can see, Korea, American, Britain, China and Japan, and some other countries, they're all focus in the music market in their country. Because they realize the strength of music. It's not only is the bridge which connects people's feelings, but also removes the language barrier, be as a part of the society's development reflection, plus brings a huge income from this field.

Hopefully, Amnhac.com will wake up this country's music marketplace again and become a bridge to connect people in Vietnam and other country. Someday, Amnhac.com will become a lever which will wake up the music talent of all Vietnamese, and be the where everybody will take a visit to relax and enjoy the melody for life.

With newest technology for building a website: ASP.NET Core, Angular and MongoDB, Amnhac.com will continue grow up and hopefully someday it will become the biggest music platform in Vietnam.

CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN

1.1. Tổng quan

1.1.1. Đặt vấn đề

Phần lớn chúng ta đều cần đến âm nhạc, nó là điều không thể thiếu trong cuộc sống, cho từng cảm xúc của mỗi người dẫn nguồn: [1]. Cho nên, một vị khách hàng đã để ý đến thị phần hùng hậu này và muốn đầu tư vào nó.

Khách hàng yêu cầu thiết kế một trang web về Âm nhạc cho tên miền Amnhac.com của họ, họ muốn một trang web Âm nhạc có đầy đủ tính năng như nghe nhạc, tải nhạc lên, người dùng phân quyền nhiều cấp bậc từ miễn phí, trả phí đến cộng tác viên và quản lý trang web. Khi so sánh với các trang web âm nhạc lớn tại Việt Nam, nó phải có phần nào đó vượt trội hơn, đòi hỏi phải tối ưu và tính bảo mật cao.

- Phía người dùng: Nghe nhạc trực tuyến, phát theo danh sách phát, bình luận, đăng ký, đăng nhập và có thể thay đổi thông tin của mình, ngoài ra còn có thể tải nhạc lên. Họ có thể tìm kiếm các bài hát theo từ khóa.
- Phía người quản lý: Duyệt các bài hát đã tải lên, tải bài hát, quản lý các danh sách nghệ sĩ, thể loại, quốc gia, thay đổi quảng cáo.

Đề tài này không chỉ là để bán cho khách hàng, mà với mục tiêu phát triển thị trường âm nhạc Việt Nam ngày một tốt hơn.

1.1.2. Tình hình hiện nay

Hiện nay trên thị trường Việt Nam, có rất nhiều trang web về âm nhạc lớn và nổi tiếng, điển hình như mp3.zing.vn, nhaccuatui.com, nhac.vui.vn.

Âm nhạc là một đề tài mà khá nhiều nhà đầu tư đổ vào phát triển do lợi nhuận khổng lồ mà nó mang lại, riêng Spotify, doanh thu thu được từ quảng cáo của những người dùng nghe nhạc từ trang web của họ đạt 1,5 tỉ đô dẫn nguồn: [1][2]. Bên cạnh đó, âm nhạc là cầu nối cảm xúc giữa người với người, là nơi mang đến sự đồng điệu, do đó, khá nhiều trang web đầu tư mạnh vào thị trường này tại Việt Nam.

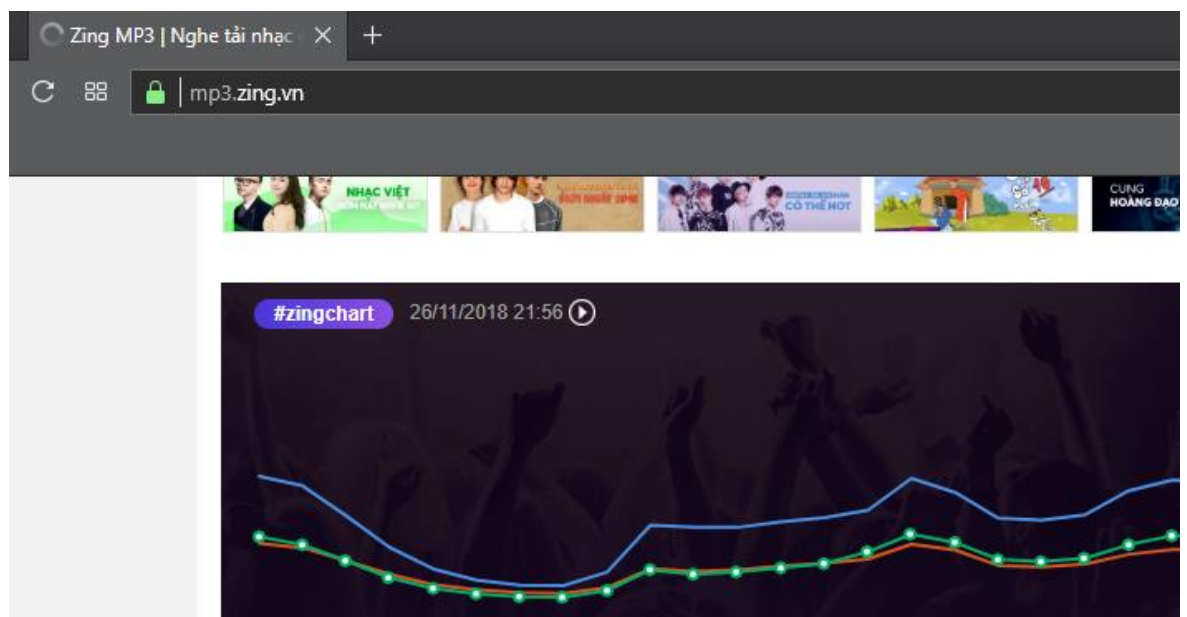
Nhìn chung các trang này đều có thị phần người nghe riêng và vô cùng đông đảo, các trang web nghe nhạc tại Việt Nam này có khá nhiều ưu điểm như:

- Nghe nhạc miễn phí, chất lượng

- Thống kê và cập nhật xếp hạng
- Luôn cập nhật nhanh nhất các bản nhạc Hot
- Kho nhạc đồ sộ

Tuy nhiên, điều khiến người dùng e ngại khi nghe nhạc tại các trang nghe nhạc tại Việt Nam là:

- Tải chậm
- Trang web cồng kềnh
- Mỗi lần nghe nhạc đều phải bật nhiều thẻ trên trình duyệt để chọn bài theo ý thích
- Mới đây, vào tháng 4/2018, VNG (hay được biết đến như đại diện của mp3.zing.vn) bị lộ hơn 160 triệu thông tin tài khoản Zing ID của người dùng do mức độ bảo mật kém. ^{dẫn nguồn: [3]}



Hình 1. 1. Tốc độ tải trang chưa được tối ưu (mp3.zing.vn)

Nếu so sánh với các trang lớn trên thế giới như YouTube, SoundCloud, chúng ta thật sự không thể nào bì được:

- Tính năng nghe nhạc mượt mà, partial load
- Bảo mật cao



Hình 1. 2. Thị phần YouTube quá dồi dào lớn mạnh (nguồn: Youtube.com)



All Distant Worlds Albums

Hình 1. 3. Tính năng partial load của Youtube

Song, trên nền tảng web, YouTube và SoundCloud cũng vẫn chưa đáp ứng được các nhu cầu như:

- Vừa nghe nhạc vừa chọn bài hát
- Tải ít hơn
- Máy chủ tại Việt Nam nhằm tải nhanh hơn.
- Tập trung vào từng thị trường âm nhạc.

1.2. Ý tưởng

Sẽ như thế nào nếu tại Việt Nam, có một trang web nghe nhạc mà:

- Tải ít
- Nghe nhạc nhanh như YouTube
- Bảo mật và coi trọng thông tin người dùng
- Đặt máy chủ tại Việt Nam
- Vừa nghe nhạc vui vẻ, vừa lướt tin nhạc và vừa chọn thêm bài hát mà không phải qua quá nhiều thao tác

- Kho âm nhạc đồ sộ
- Cập nhật nhanh nhất các bản nhạc hot
- Là sân chơi âm nhạc cho tất cả mọi người

1.3. Mục tiêu đề tài

- Khắc phục và nhắc nhở các trang web âm nhạc tại Việt Nam nói riêng, các trang web tại Việt Nam nói chung coi trọng bảo mật và thông tin cá nhân của người dùng.
- Học hỏi và tiếp cận công nghệ mới nhất của thế giới.
- Xây dựng một nền tảng web Âm Nhạc lớn và tiên tiến nhất tại Việt Nam và phù hợp với thị trường nghe nhạc trong nước, với mục tiêu chính là phát triển thị trường âm nhạc tại Việt Nam ngày càng lớn mạnh hơn nữa.
 - o Nghe nhạc miễn phí, nhanh và trải nghiệm tuyệt vời như YouTube
 - o Cho phép người dùng chia sẻ, tải nhạc, kể cả các bản nhạc cover họ yêu thích.
 - o Xây dựng cộng đồng âm nhạc mới, đặc biệt dành cho những ai yêu âm nhạc, ca hát, hay sáng tác.
 - o Là nơi chia sẻ cảm xúc bằng âm nhạc, bằng sự đồng điệu của tất cả mọi người, âm nhạc là cách để kết nối mọi người với nhau.



Hình 1. 4. Biểu tượng Amnhac.com

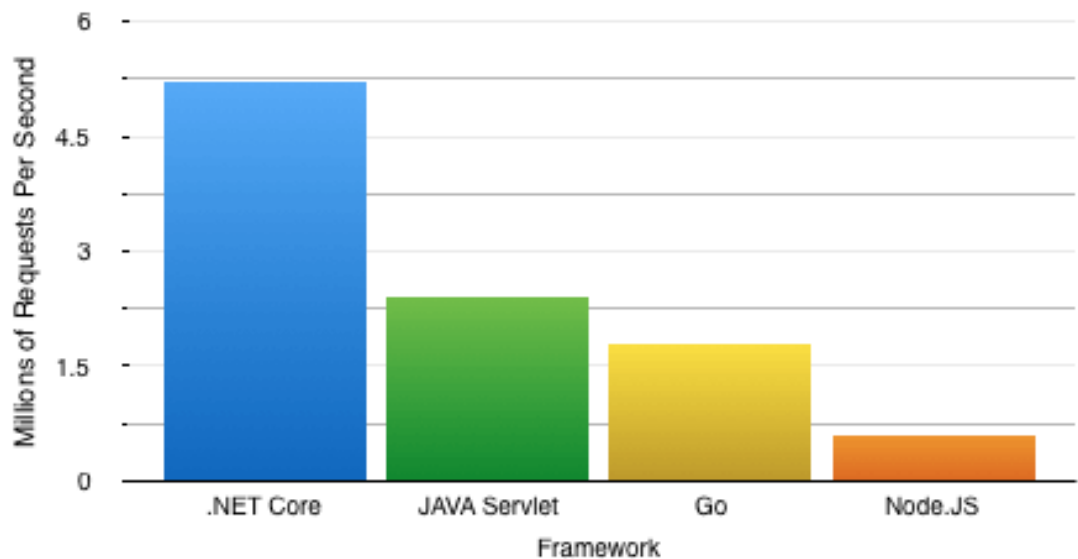
1.4. Công cụ

Chắc hẳn chúng ta sẽ phải cần một nền tảng Web để tiếp cận thị trường này, một nền tảng nhẹ và tiện dụng.

Song, để đáp ứng hầu hết nhu cầu, và có thể mở rộng lên đa nền tảng về sau, chúng ta sẽ đi theo hướng Dịch vụ Web, mà sử dụng Web View là bước tiên phong để phát triển thị trường.

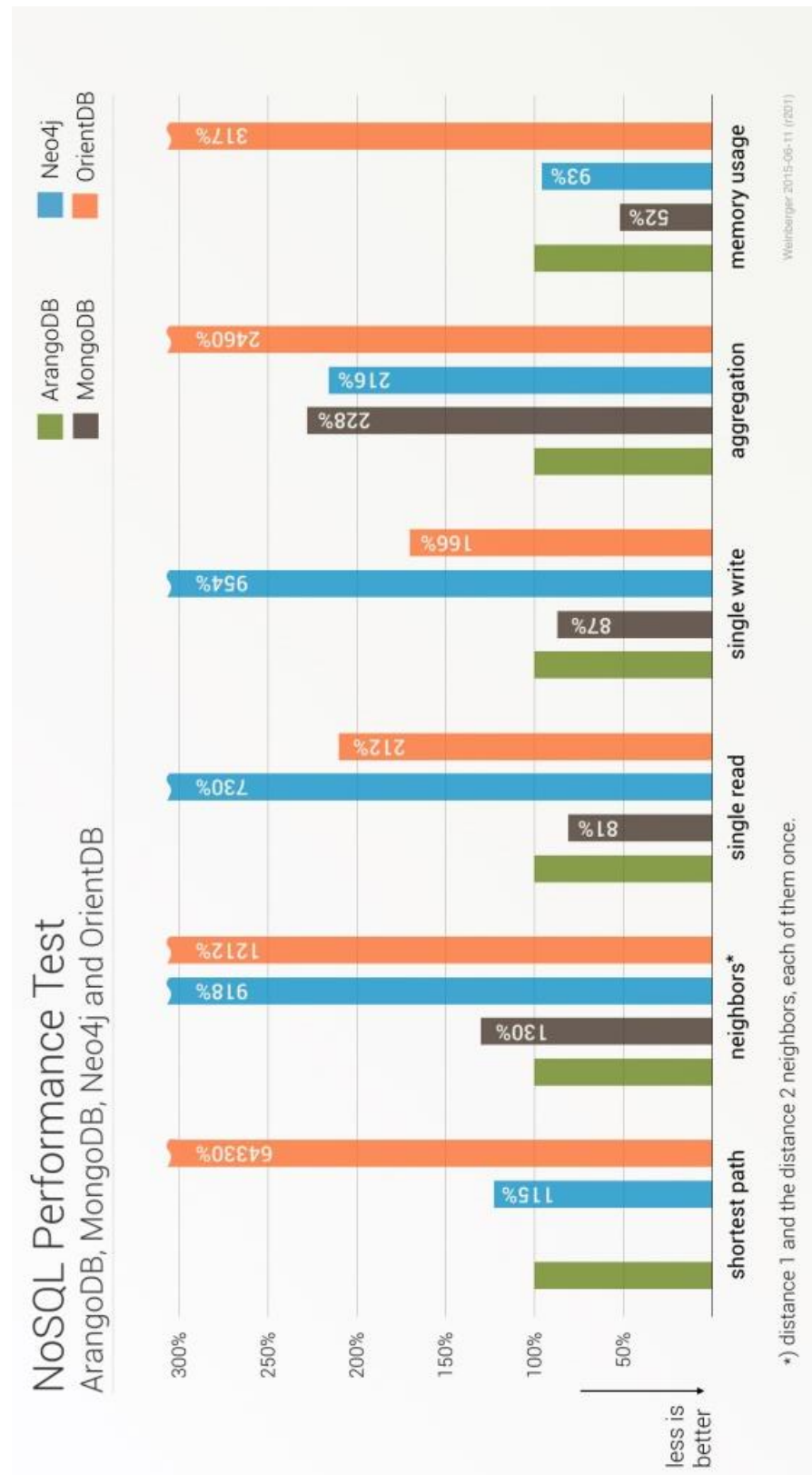
1.5. Hướng giải quyết

- Một dịch vụ web mở rộng cao, nhanh, và phản hồi tốt: Sử dụng ASP.NET Core 2.1, do nền tảng này dù mới nổi, nhưng tốc độ quá dồi vượt trội, lại đa nền tảng.



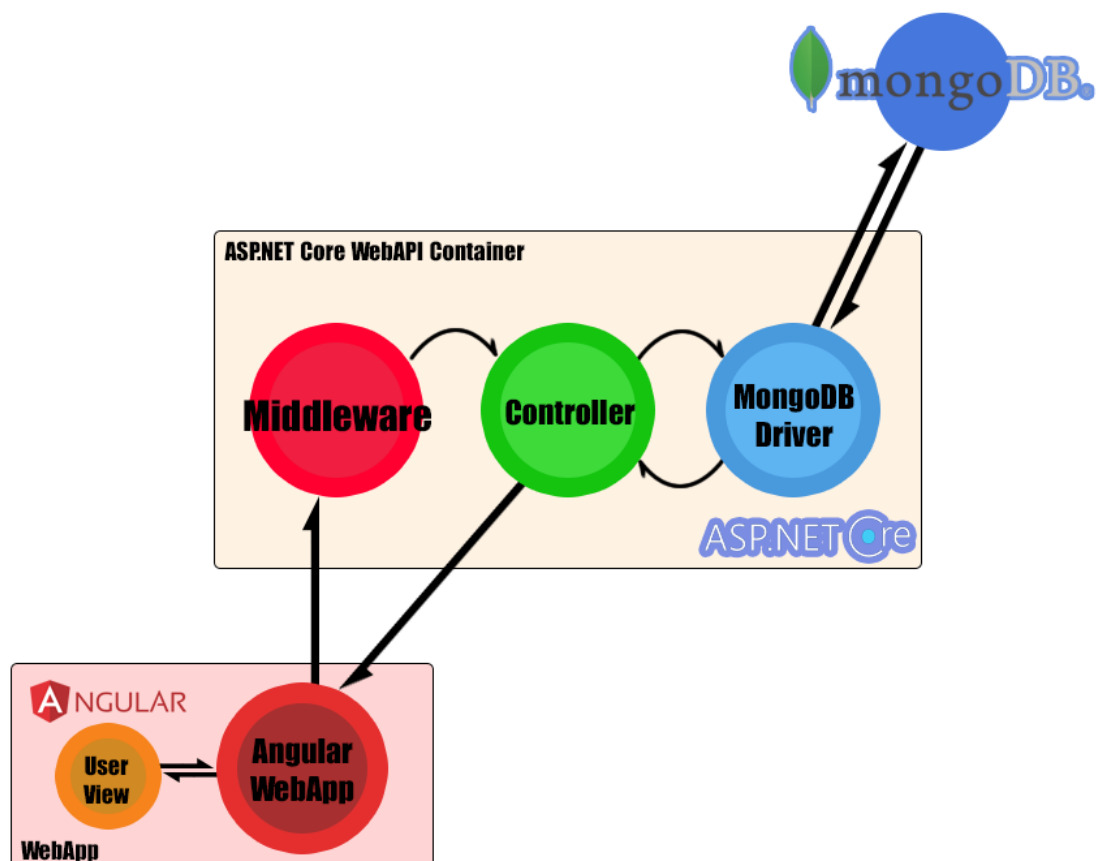
Hình 1. 5. Thống kê số lượng request đáp ứng mỗi giây (nguồn: <https://stackify.com/asp-net-core-features/>)

- Sử dụng MongoDB để giải quyết vấn đề về truy vấn cơ sở dữ liệu, do MongoDB là cơ sở dữ liệu dạng NoSQL, cho nên sẽ có thể cho hiệu suất tốt hơn so với các cơ sở dữ liệu khác. Mặc dù có rất nhiều cơ sở dữ liệu NoSQL khác tốt hơn, song MongoDB hỗ trợ nền tảng ASP.NET core chi tiết hơn và nhiều chức năng hơn nên ta chọn MongoDB.



Hình 1. 6. So sánh MongoDB và các cơ sở dữ liệu tốt nhất hiện nay (nguồn: ArangoDB)

- Sử dụng JWT Token Authentication cùng quy tắc độn Salt ngẫu nhiên giúp bảo mật thông tin người sử dụng.
- Dùng Partial Load trong ASP.NET Core qua FileStreamResult để tải nhạc từng phần, nhằm tối ưu chức năng phát nhạc.
- Sử dụng Angular để làm Web View đơn trang, như vậy người dùng không cần mở nhiều thẻ trên trình duyệt mà vẫn vừa lướt vừa nghe nhạc được, đồng thời, Angular sẽ giúp người dùng chỉ tải thành phần cần thiết trên trang, sẽ tiết kiệm thời gian tải.
- Để gửi cho người dùng các thông báo thời gian thực như thông tin, các bài hát hot và đòi hỏi bảo mật, ta có thể sử dụng SignalR để gửi thông báo thời gian thực và dùng SignalR Message Package để mã hóa thông tin trao đổi.
- Kết hợp giữa Angular và ASP.NET Core để tạo nên thành phần web và dịch vụ web, triển khai sẽ linh hoạt hơn.



Hình 1. 7. Mô hình sơ lược thiết kế ứng dụng tổng quát

1.6. Bố cục

Do khá nhiều công nghệ được áp dụng, cho nên để có thể dễ dàng hiểu và nghiên cứu về sau, luận văn này được chia thành 3 phần chính, bao gồm:

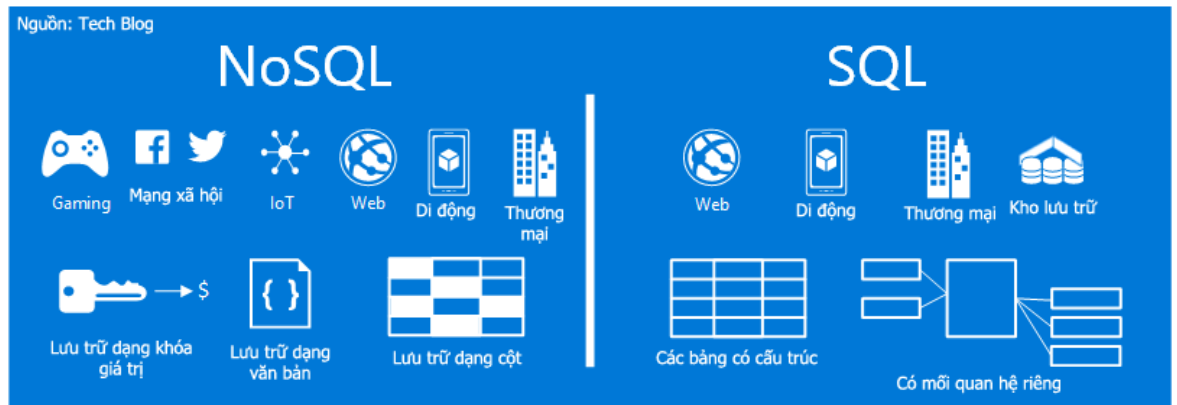
- **Phần cơ sở lý thuyết:** 4 Chương
 - o MongoDB: Giới thiệu về cơ sở dữ liệu NoSQL và MongoDB
 - o Angular: Giới thiệu về Angular
 - o ASP.NET Core: Giới thiệu về ASP.NET Core
 - o SignalR: Giới thiệu về SignalR trong ASP.NET Core
- **Phần áp dụng:** 1 Chương
 - o Giải quyết vấn đề
- **Phần tổng kết:** 1 Chương
 - o Tổng kết: Kết quả và đánh giá
- **Phụ lục và Tài liệu tham khảo:** Gồm tài liệu tham khảo và chi tiết các mẹo giải quyết vấn đề cho đề tài.

PHẦN 1: CƠ SỞ LÝ THUYẾT

CHƯƠNG 2: MONGODB

2.1. NoSQL

2.1.1. Giới thiệu



Hình 2. 1. So sánh NoSQL và SQL (nguồn: TechBlog)

NOSQL là loại cơ sở dữ liệu có ngôn ngữ truy vấn phi quan hệ, nó lưu dữ liệu theo nhiều cách, song, mỗi dòng dữ liệu có thể không có sự tương đồng nhau về cấu trúc hay chúng không có mối quan hệ với nhau hay mối quan hệ với các tập dữ liệu khác trong cùng một cơ sở dữ liệu. ⁽¹⁾

2.1.2. Ưu điểm của NoSQL

- **Linh hoạt:** Do nó không có cấu trúc nên có thể lưu trữ dữ liệu linh hoạt theo nhiều kiểu dữ liệu.
- **Tốc độ truy vấn nhanh:** Do không phải khiến hệ quản trị lo về cấu trúc và quan hệ, đồng thời lưu trữ đơn giản nên tốc độ truy vấn rất nhanh.
- **Tính mở rộng cao:** Mở rộng cơ sở dữ liệu bằng cách tăng dung lượng hay số lượng máy chủ lưu trữ thay vì cấu hình máy.

2.1.3. Cách thức lưu trữ

Mỗi hệ quản trị cơ sở dữ liệu dùng một cách thức lưu trữ khác nhau, bao gồm:

- Dạng cột

- Dạng tài liệu
- Dạng khóa-trị
- Dạng đồ thị
- Dạng đa mô hình

2.1.4. Tính năng của NoSQL

- **Tối ưu về dung lượng và tốc độ.**
- **ACID:** Một số hệ quản trị NoSQL giờ hỗ trợ tính ACID tương tự SQL thật sự.
- **Cụm dữ liệu con:** Một số hệ quản trị cho phép lưu một giá trị của một tập dữ liệu là tập các giá trị khác.

2.2. MongoDB

2.2.1. Tại sao nên dùng MongoDB?



Hình 2. 2. (nguồn: MongoDB)

Là hệ quản trị cơ sở dữ liệu NoSQL, miễn phí hoàn toàn. Nó lưu dữ liệu theo dạng tài liệu BSON (tương tự JSON). Nó kế thừa các đặc tính của một hệ quản trị cơ sở dữ liệu NoSQL.

MongoDB được đánh giá là dễ sử dụng, linh hoạt và tốc độ truy vấn cao, phù hợp trong nhiều trường hợp sử dụng.

2.2.2. Mục đích sử dụng MongoDB

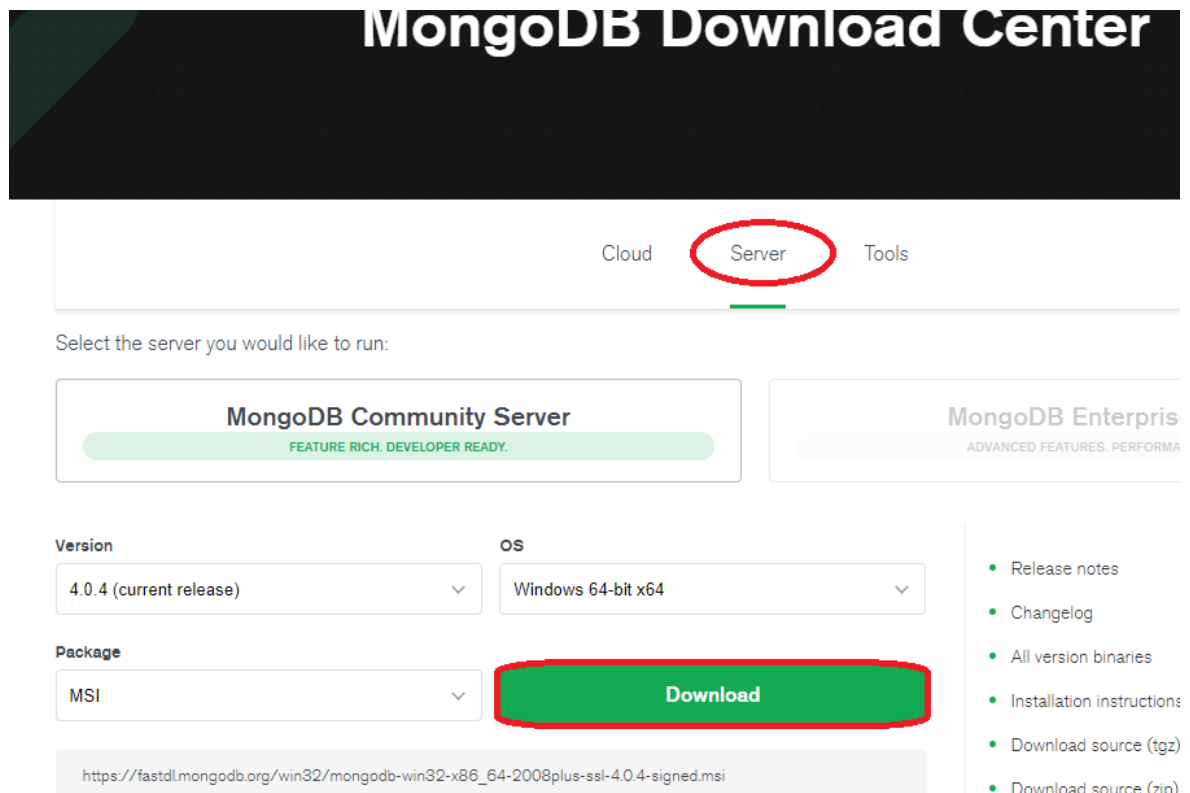
Trong dự án Amnhac, MongoDB là một cơ sở dữ liệu NoSQL, chắc hẳn nó sẽ đáp ứng được nhu cầu sử dụng cao và nhiều người dùng một lúc. Ngoài ra, do Amnhac sử dụng ASP.NET Core, một nền tảng web dựa trên C#, mà MongoDB lại có bộ Driver hỗ trợ rất tốt, tài liệu đầy đủ, cho nên, hiện tại MongoDB là ứng cử viên sáng giá nhất cho dự án này.

Trong dự án, MongoDB sẽ là cơ sở dữ liệu chính cho ứng dụng, hỗ trợ truy vấn nhanh và tìm kiếm Full-Text Lookup.

2.2.3. Cài đặt

Sau khi tải bộ cài từ trang chủ [MongoDb.com](https://mongodb.com), chạy bộ cài và theo hướng dẫn có sẵn của MongoDB là hoàn tất. Khá là đơn giản.

Ta sẽ tải về bản Community Edition



Hình 2. 3. Tải về tại mongodb.com

2.2.4. Định nghĩa

MongoDb có một vài định nghĩa về thuật ngữ khác biệt so với SQL khi lưu trữ, cụ thể như sau:

SQL	MongoDb	Ý nghĩa
Database	Database	Một cơ sở dữ liệu, dùng lưu trữ dữ liệu.
Table	Collection	Một tập dữ liệu, chứa nhiều dòng dữ liệu

Row	Document	Một dòng dữ liệu
Column	Field	Một cột dữ liệu, được định nghĩa kiểu dữ liệu cho từng dòng.
JOINS	\$lookup	Nối tập dữ liệu lúc tìm
GROUP_BY	Aggregation Pipeline	Gom nhóm dòng dữ liệu

Bảng 2. 1. Định nghĩa thuật ngữ trong MongoDB

2.2.5. Truy vấn

Phần lớn thao tác trên MongoDB là trên giao diện câu lệnh.

Từ giao diện câu lệnh (Command Prompt/Windows Power Shell trên Windows hoặc Terminal trên Linux) nhập “mongo” để vào giao diện nhập lệnh của MongoDB.

```
C:\Program Files\MongoDB\Server\3.6\bin>mongo
MongoDB shell version v3.6.1
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.1
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  http://docs.mongodb.org/
Questions? Try the support group
  http://groups.google.com/group/mongodb-user
```

Hình 2. 4. Giao diện dòng lệnh cho Mongo

MongoDb sử dụng cú pháp truy vấn riêng, theo hướng cú pháp của các truy vấn mảng trong Javascript.

Cú pháp	Ý nghĩa	Ví dụ
use <database>	Mở sử dụng một cơ sở dữ liệu	use user;
db.collection.find()	Truy vấn tìm kiếm dữ liệu trong <i>collection</i>	db.user.find();

<code>db.collection.insert()</code>	Thêm dữ liệu mới	<code>db.user.insert({ username:'DHCT', password:'CTU' });</code>
<code>db.collection.update()</code>	Cập nhật dòng dữ liệu	<code>db.users.update({ age: { \$gt: 25 } }, { \$set: { status: 'C' } }, { multi: true });</code>
<code>s.start_transaction()</code>	Bắt đầu một phiên giao dịch Transaction	
<code>s.commit_transaction()</code>	Hoàn tất phiên giao dịch Transaction	
<code>db.createUser({ user:<tài khoản>, pwd:<mật khẩu>, roles:[<danh sách quyền>] })</code>	Tạo người dùng với tài khoản, mật khẩu, kèm cấp quyền	<code>db.createUser({ user: "accountUser", pwd: "password", roles: ["readWrite", "dbAdmin"]})</code>

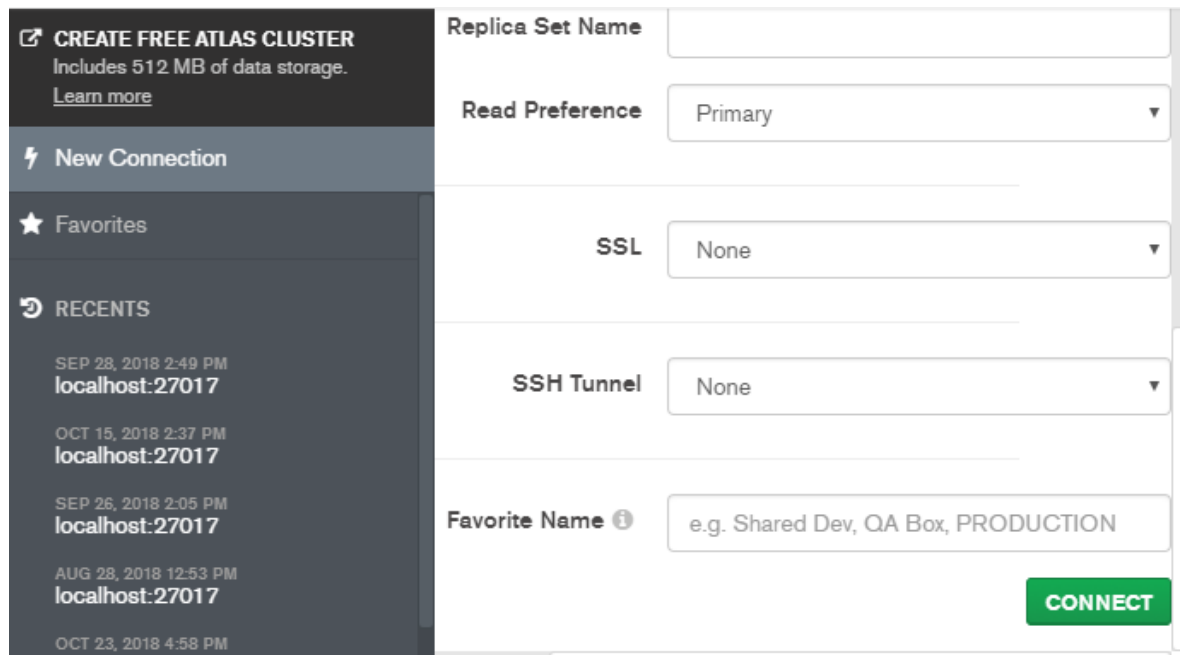
Bảng 2. 2. Các câu lệnh đơn giản trong MongoDB

2.2.6. MongoDB Compass

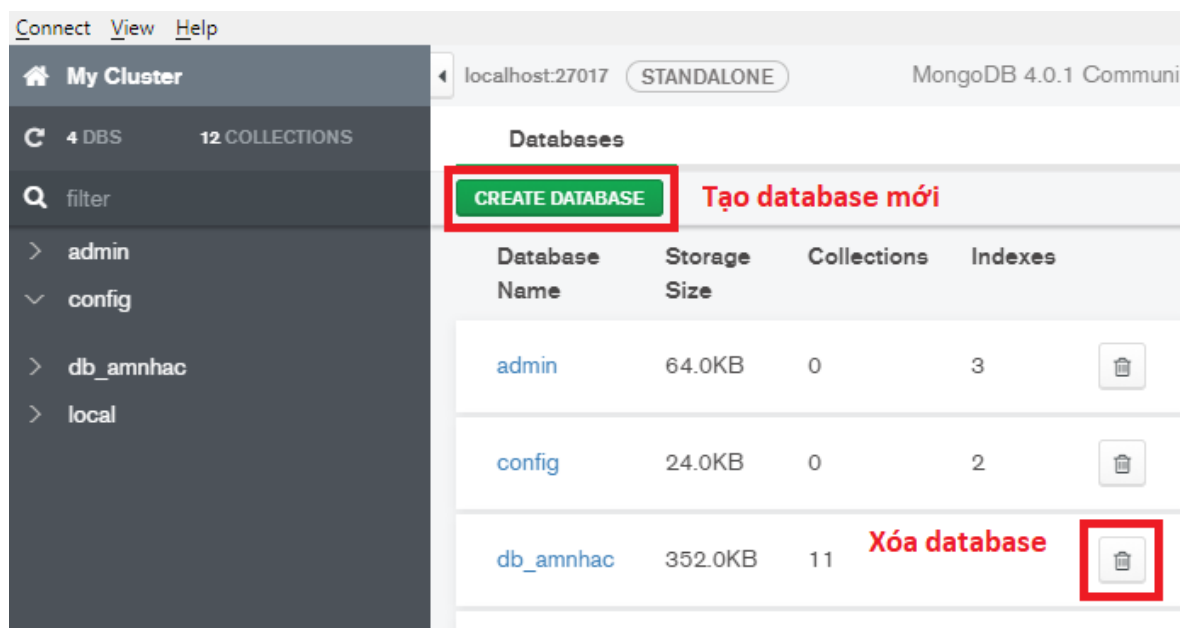
Đây là công cụ giúp quản trị cơ sở dữ liệu của MongoDB dưới giao diện người dùng thân thiện hơn.

MongoDb hiện cung cấp ba phiên bản của MongoDB Compass gồm:

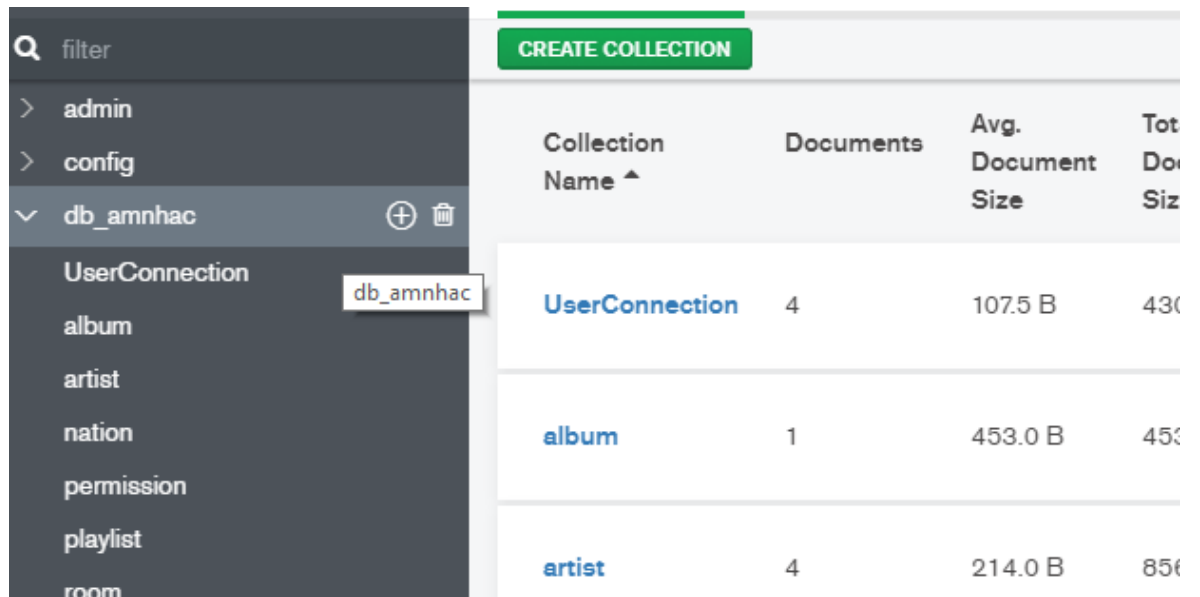
- **Phiên bản chỉ đọc (Readonly Edition)**
- **Phiên bản độc lập (Isolated Edition):** Chỉ dùng cho bản trả phí.
- **Phiên bản cộng đồng (Community Edition):** Bản miễn phí



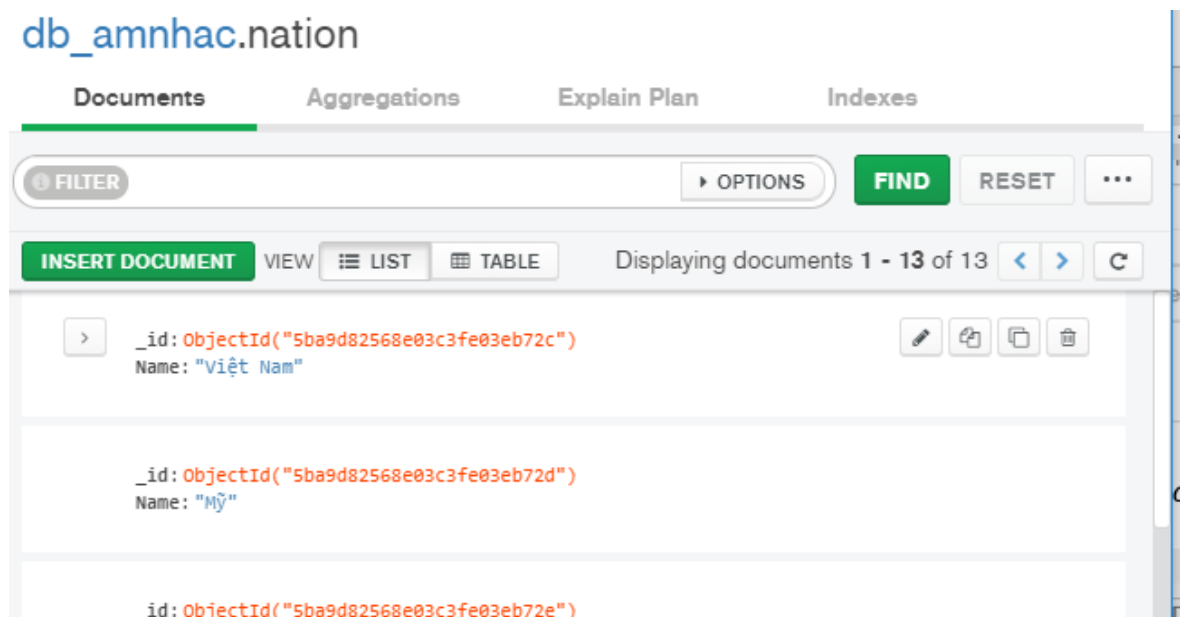
Hình 2. 5. Giao diện kết nối cơ sở dữ liệu trong MongoDB Compass



Hình 2. 6. Giao diện sau khi kết nối máy chủ thành công



Hình 2. 7. Nhấp chọn một Database để xem hoặc thêm các Collection bên trong



Hình 2. 8. Bên trong một Collection sẽ được hỗ trợ Query, Thêm, sửa, xóa



Hình 2. 9. Khi chỉnh sửa có thể thay đổi dữ liệu, kiểu dữ liệu và xóa dữ liệu

CHƯƠNG 3: ANGULAR

3.1. Giới thiệu về Angular



Hình 3. 1. Biểu tượng của Angular (nguồn: Angular.io)

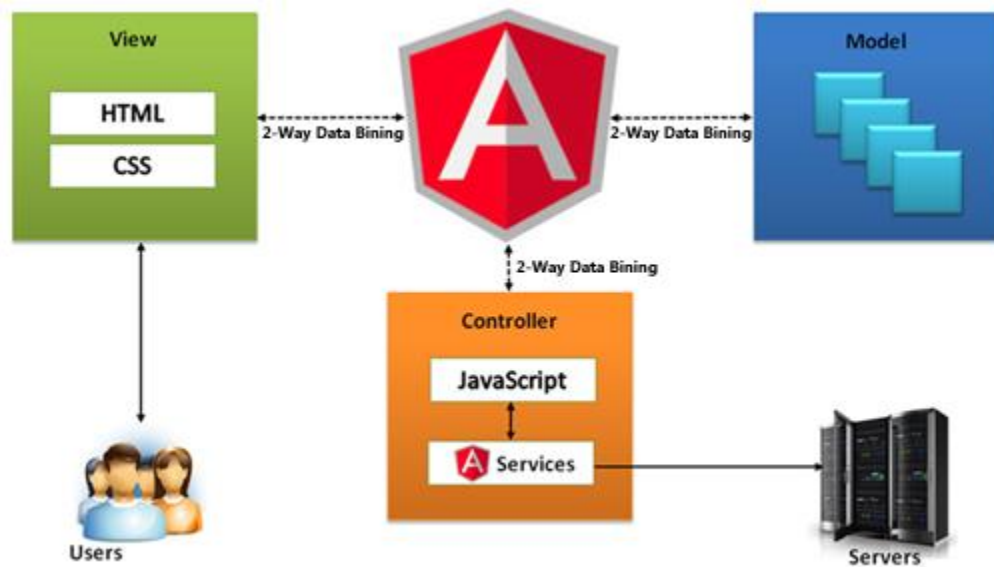
Là một Javascript Framework nổi danh, được đầu tư bởi Google, chức năng nổi bật nhất là dùng để thiết kế một trang web đơn trang (Single Page Component).

Với phiên bản Angular 2+, Angular được chuyển sang chạy dưới nền tảng của NodeJS, cùng với tập câu lệnh hỗ trợ người dùng thiết đặt và lập trình trang web tốt hơn.

3.2. Mục đích sử dụng

Với Amnhac.com, để thu hút người sử dụng, đồng thời khắc phục các điểm yếu của các trang web trong nước hiện tại, Angular sẽ là lựa chọn tốt nhất để xây dựng một trang web đơn trang, không phải tải quá nhiều thành phần ở các lần tải trang kế, không gián đoạn các hoạt động đang tiếp diễn.

Angular giúp Amnhac.com mang lại giao diện thân thiện cho người dùng, trong đó với lập trình viên, mã nguồn dễ đọc và dễ hiểu hơn, xử lý giao diện nhanh hơn với chức năng ràng buộc một chiều và ràng buộc hai chiều.



Hình 3. 2. Tổng quan mô hình MVC của Angular (nguồn: Codeburst.io)

Nhờ có Angular, người dùng giờ có thể vừa nghe nhạc vừa lướt tin tức âm nhạc chỉ qua một thẻ trên trình duyệt.

3.3. Giới thiệu về NodeJS



Hình 3. 3. Biểu tượng NodeJS (nguồn: NodeJS.org)

Hiện nổi danh dưới bóng Javascript Runtime, NodeJS tìm cho mình một chỗ đứng trong mảng ứng dụng mạng và ứng dụng Web.

Trong Angular, nó đóng vai trò cốt lõi để sửa lỗi và xây dựng một ứng dụng web, nó giúp đóng gói các thành phần cần thiết để cho ra thành phẩm là một trang web sử dụng Angular hoàn chỉnh.

3.4. Cài đặt

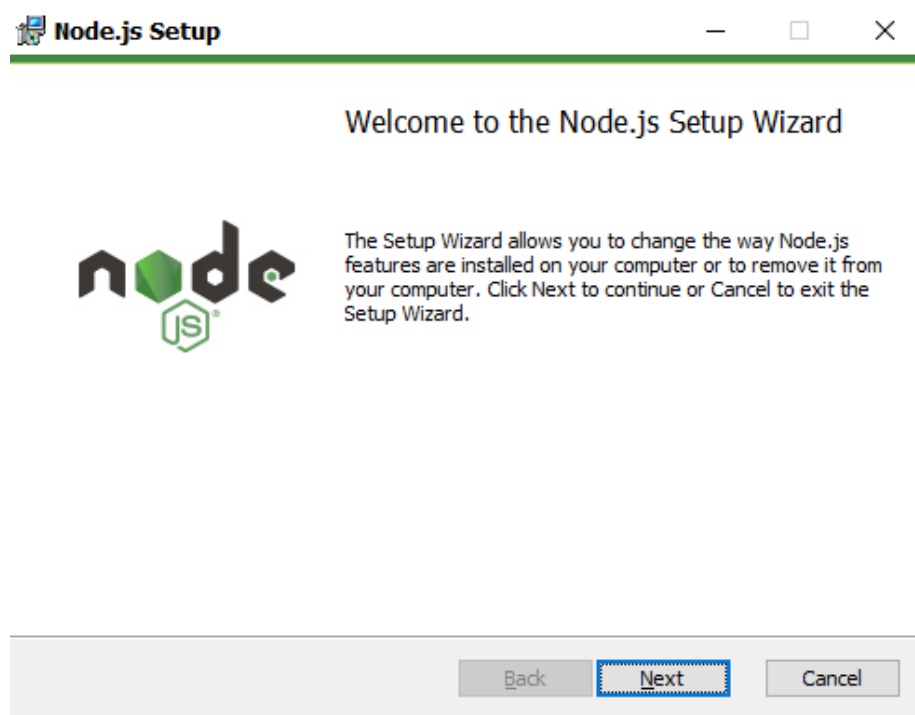
Để tiến hành sử dụng Angular, người dùng cần phải trải qua 2 bước cài đặt như sau:

3.4.1. Cài đặt NodeJS

Để cài đặt Angular, ta cần trước tiên cài đặt NodeJS.

Đối với hệ điều hành Windows, người dùng có thể tải về bản cài đặt ngay tại trang chủ NodeJS và cài đặt bình thường.

Tải bản cài đặt cho Windows tại nodejs.org



Hình 3. 4. Giao diện trình cài đặt Node.js trên Windows

Đối với hệ điều hành Linux, người dùng cần cài đặt Node Version Manage (NVM) trước.

Để cài đặt NVM, người dùng cần nhập lệnh sau trong Terminal:

```
curl -o-  
https://raw.githubusercontent.com/creationix/nvm/v0.33.11/install.sh | bash
```

Sau đó, ta tiến hành gõ trong terminal:

```
nvm install node
```

Sau đó, NVM sẽ cài đặt NodeJS vào máy.

Để kiểm tra NodeJS đã cài đặt chưa, người dùng có thể gõ lệnh sau để kiểm tra phiên bản

```
npm -version
```



```
C:\>npm -version
5.6.0
```

Hình 3. 5. Kiểm tra phiên bản Node qua Command Prompt

3.4.2. Cài đặt Angular CLI

Sau khi đã cài đặt NodeJS, để cài đặt Angular phiên bản mới nhất, người dùng có thể mở **Command Prompt** (đối với hệ điều hành Windows) hoặc **Terminal** (đối với hệ điều hành Linux), sau đó nhập lệnh sau:

```
npm install -global @angular/cli
```

Lệnh trên sẽ cài đặt Angular CLI vào máy tính.

3.5. Lập trình

3.5.1. Lập trình trên môi trường mới

Mở **giao diện dòng lệnh** tại thư mục cần tạo dự án.

Sau đó, gõ lệnh:

```
ng new myapp
```

Trong đó, myapp là tên của dự án người dùng muốn tạo.

```
PS D:\Project\Web\NETCORE PROJECT\Report\Test> ng new myapp
CREATE myapp/angular.json (3539 bytes)
CREATE myapp/package.json (1309 bytes)
CREATE myapp/README.md (1022 bytes)
CREATE myapp/tsconfig.json (384 bytes)
CREATE myapp/tslint.json (2805 bytes)
CREATE myapp/.editorconfig (245 bytes)
CREATE myapp/.gitignore (503 bytes)
CREATE myapp/src/environments/environment.prod.ts (51 bytes)
CREATE myapp/src/environments/environment.ts (631 bytes)
CREATE myapp/src/favicon.ico (5430 bytes)
CREATE myapp/src/index.html (292 bytes)
CREATE myapp/src/main.ts (370 bytes)
CREATE myapp/src/polyfills.ts (3194 bytes)
CREATE myapp/src/test.ts (642 bytes)
CREATE myapp/src/assets/.gitkeep (0 bytes)
CREATE myapp/src/styles.css (80 bytes)
CREATE myapp/src/browserslist (375 bytes)
CREATE myapp/src/karma.conf.js (964 bytes)
CREATE myapp/src/tsconfig.app.json (194 bytes)
CREATE myapp/src/tsconfig.spec.json (282 bytes)
CREATE myapp/src/tslint.json (314 bytes)
CREATE myapp/src/app/app.module.ts (314 bytes)
CREATE myapp/src/app/app.component.html (1141 bytes)
CREATE myapp/src/app/app.component.spec.ts (988 bytes)
CREATE myapp/src/app/app.component.ts (207 bytes)
CREATE myapp/src/app/app.component.css (0 bytes)
CREATE myapp/e2e/protractor.conf.js (752 bytes)
CREATE myapp/e2e/src/app.e2e-spec.ts (301 bytes)
CREATE myapp/e2e/src/app.po.ts (208 bytes)
CREATE myapp/e2e/tsconfig.e2e.json (213 bytes)
[.....] | fetchMetadata: 511ms range manifest for codelyzer@~4.2.1 fetched in 870ms
```

Hình 3. 6. Quy trình tạo ứng dụng Angular đầu tiên

Lúc này, kiểm tra thư mục sẽ thấy các tập tin dự án đã được tạo

Name	Date modified
e2e	9/15/2018 3:48 PM
node_modules	9/15/2018 3:49 PM
src	9/15/2018 3:48 PM
.editorconfig	9/15/2018 3:48 PM
.gitignore	9/15/2018 3:48 PM
angular.json	9/15/2018 3:48 PM
package.json	9/15/2018 3:48 PM
README.md	9/15/2018 3:48 PM
tsconfig.json	9/15/2018 3:48 PM
tslint.json	9/15/2018 3:48 PM

Hình 3. 7. Các tập tin và thư mục của ứng dụng Angular

3.5.2. Lập trình trên môi trường có sẵn

Thông thường, các dự án Angular sẽ có thư mục **node_modules**. Song nếu không có thư mục này, hãy mở giao diện dòng lệnh lên, chuyển đến thư mục dự án và gõ:

npm install

Lúc này NodeJS sẽ cài đặt môi trường cho người dùng.

3.5.3. Công cụ soạn thảo

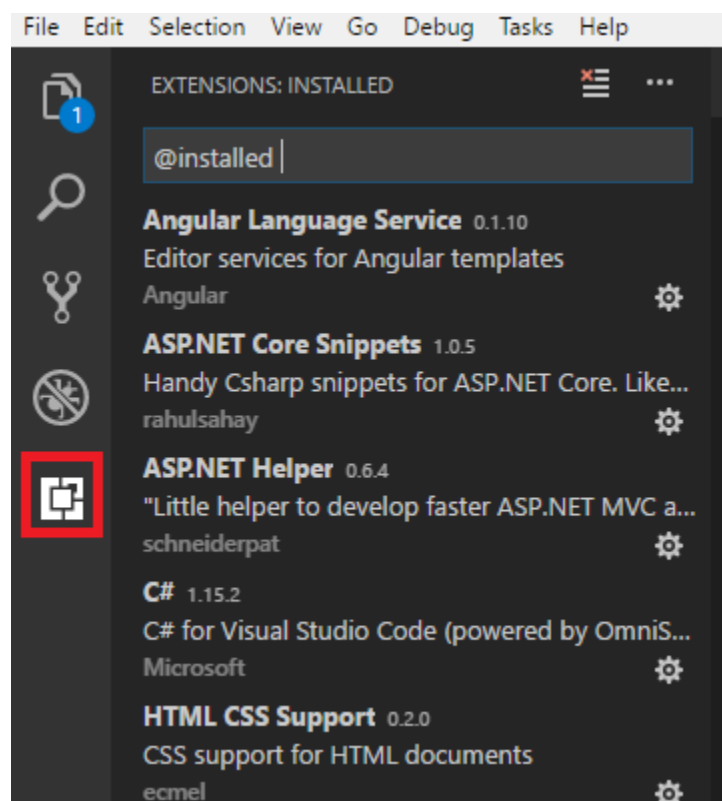
Bất kỳ chương trình hỗ trợ soạn thảo nào hỗ trợ HTML5 và CSS3 đều có thể dùng để lập trình ứng dụng Angular.

Khuyến nghị nên sử dụng Visual Studio Code mới nhất



Hình 3. 8. Biểu tượng Visual Studio Code (nguồn: visualstudio.microsoft.com)

Visual Studio code có thể cài đặt thêm các gói mở rộng trong giao diện, hỗ trợ và gợi ý người dùng viết HTML5, CSS3 và cả Angular TypeScript.



Hình 3. 9. Cài đặt Extension cho Visual Studio Code

3.5.4. Các cú pháp dòng lệnh

Thao tác khi xây dựng ứng dụng Web Angular đa phần sử dụng giao diện câu lệnh để tạo lập các trang, các thành phần, hay sửa lỗi hoặc đóng gói ứng dụng. Sau đây là các câu lệnh hay sử dụng:

Cú pháp	Ý nghĩa
ng new <project>	Tạo lập một dự án Angular mới
npm install	Cài đặt các thư viện phụ thuộc cho ứng dụng
ng serve	Chạy thử ứng dụng (Thông thường sẽ được mở tại localhost:4200)
ng build --prod	Đóng gói ứng dụng cho bản phát hành. Các gói được tối ưu và không hiển thị lỗi khi triển khai.
ng generate component <tên> (hoặc <i>ng g c <tên></i>)	Tạo component (Web Page/ thành phần Web Page) trong Angular.
ng generate service <tên> (hoặc <i>ng g s <tên></i>)	Tạo Service trong Angular
ng generate module <tên> (hoặc <i>ng g m <tên></i>)	Tạo Module trong Angular

Bảng 3. 1. Cú pháp dòng lệnh hay sử dụng trong Angular CLI

3.5.5. Các thành phần chính

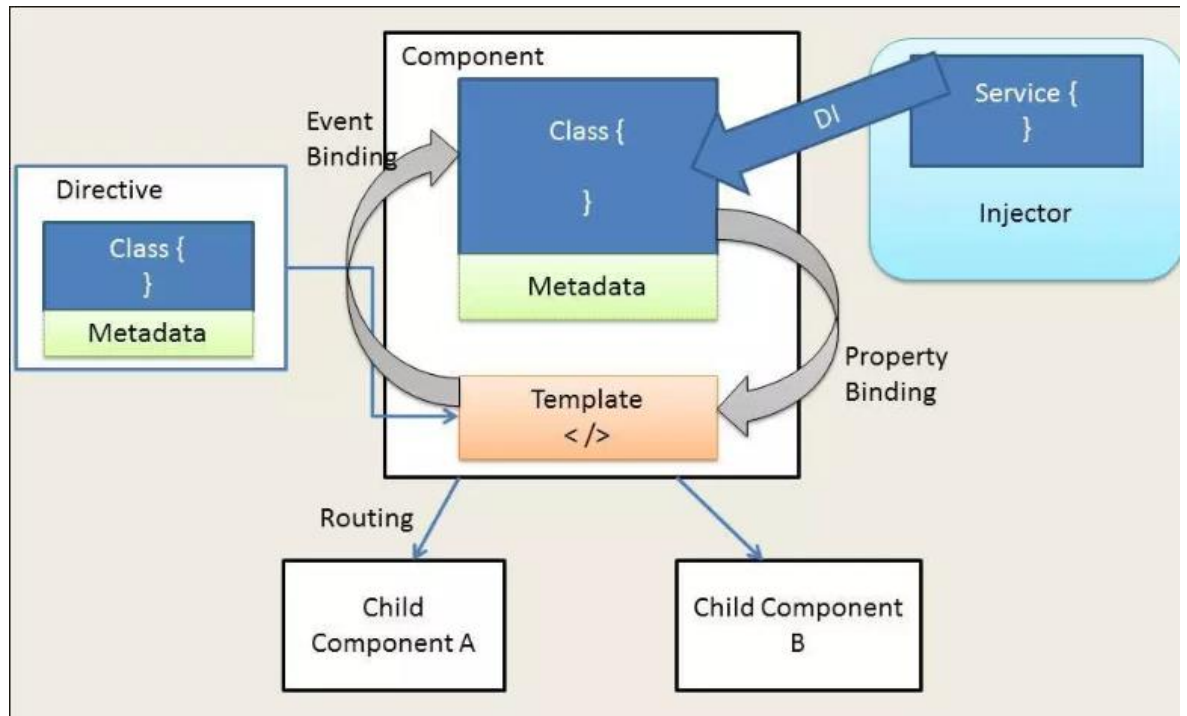
Để bắt đầu, vui lòng chạy trang web dưới chế độ sửa lỗi, mở thư mục chứa và gõ lệnh: *ng serve --o*

3.5.5.1. Cấu trúc chính

Khi khởi tạo dự án Angular, người dùng có thể thấy các thư mục sau:

- **node_modules:** Chứa thư viện cần thiết cho ứng dụng web Angular
- **src:** Thư mục chứa nội dung dự án

- **src/app:** Thư mục chứa các thành phần của dự án, gồm css, html và ts
- **src/assets:** Thư mục chứa tập tin tài nguyên như ảnh, css, js, nhạc, video.
- **src/environment:** Thư mục chứa tập tin thiết lập môi trường cho người lập trình.



Hình 3. 10. Mô hình hoạt động của Angular (nguồn: CIPHERTRICK.COM)

Một Ứng dụng Angular khi khởi chạy sẽ có rất nhiều thành phần được kết nối lẫn nhau, và chúng sẽ được ràng buộc các thuộc tính và sự kiện giữa code-behind (classes, function) và giao diện (template).

Ví dụ như khi một class thay đổi thuộc tính thì nó sẽ hiển thị thay đổi đó ra ngoài template, và khi template gõ giá trị gì cho một thuộc tính bên ngoài template thì nó sẽ kích hoạt sự kiện ở phía code-behind và thay đổi thuộc tính đó theo.

Để hiểu rõ hơn về cách hoạt động này, chúng ta cần hiểu rõ hơn các thành phần có trong Angular.

3.5.5.2. Component

Là các thành phần của Angular, nó có thể là một trang hoặc là một thành phần của trang.

Một thành phần của Angular bao gồm 3 tập tin chính:

- **Tập tin HTML:** Chứa các thẻ cho trang web.

- **Tệp định kiểu CSS:** Định kiểu cho tệp HTML trong thành phần.
- **Tệp sự kiện TS:** Khai báo các thư viện và các định nghĩa nhằm xác định các tệp liên quan là thành phần.

Tệp TS là tệp chính để có thể xác định chúng là một thành phần. Ví dụ, trong AppComponent.ts, thành phần App được định nghĩa như sau:

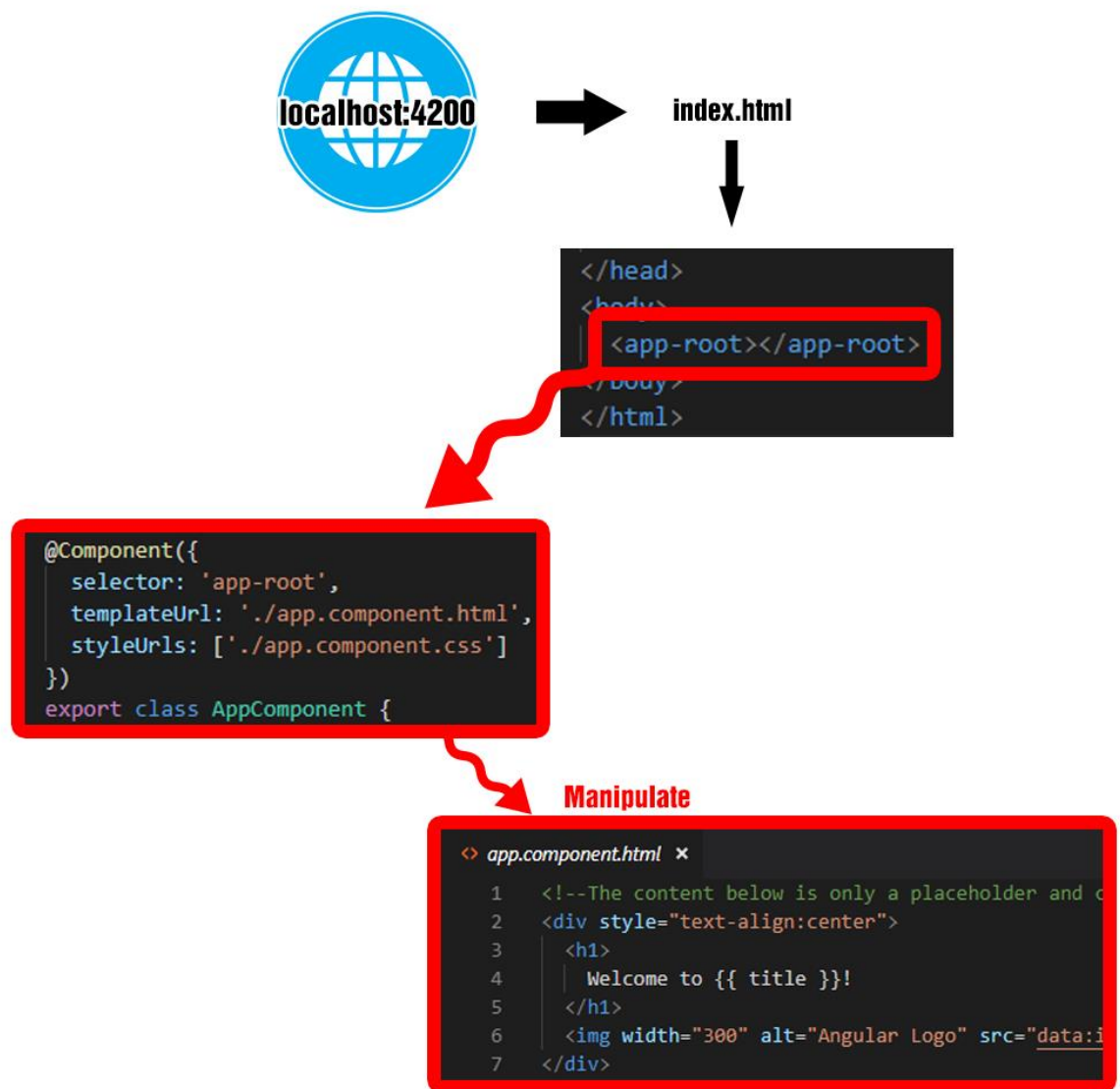
```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})  
export class AppComponent {
```

Hình 3. 11. Cú pháp định nghĩa Component

Thẻ Component sẽ khai báo cho class AppComponent các thuộc tính sau:

- **Selector:** Một khai báo như cho biết một thẻ HTML mới
- **templateUrl:** Đường dẫn tệp HTML để thay thế khi gặp Selector trên
- **styleUrls:** Đường dẫn tệp định kiểu CSS cho tệp HTML trên.

Mở tệp tin **index.html** ở thư mục **src** , người dùng sẽ thấy thẻ **<app-root>** được dùng đến. Lúc chạy, Angular sẽ nhét (manipulate) các định kiểu và định dạng thẻ HTML thật sự mà trình duyệt có thể hiểu



Hình 3. 12. Cách Angular nhét (manipulate) một Component ra giao diện

Người dùng ngoài ra còn có thể tạo thêm nhiều thành phần tương tự như AppComponent bằng cách gõ lệnh trong giao diện dòng lệnh:

ng g c <Tên thành phần>

Ví dụ: *ng g c MyAccount*

Do AppComponent là thành phần chính nên ta sẽ đặt các thành phần đó vào AppComponent.html.

3.5.5.3. Module

Là một tập tin TS được đặt thẻ Module nhằm khai báo và gắn kết các thành phần để có thể sử dụng được.

Angular có một Module lớn nhất là AppModule (app.module.ts), nơi đây sẽ khai báo các thư viện được sử dụng cho các thành phần và khai báo các trang mà người dùng tạo.

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Hình 3. 13. Cú pháp định nghĩa Module

Thẻ **@NgModule** định danh một module định danh các thành phần để nói cho Angular biết nên gộp và chạy trang web thế nào, nên thêm chức năng nào và bỏ chức năng nào không cần thiết.

- **Declarations:** Khai báo các trang, thành phần. Ví dụ như *Acomponent* vừa tạo nào đó, để có thể sử dụng thành phần này, ta phải khai báo trong đây.
- **Imports:** Khai báo các Module , dịch vụ chức năng, hay gọi nôm na là các thư viện chức năng như hoạt ảnh, routing,...
- **Providers:** Nơi khai báo các class được xem như là Service.
- **Bootstrap:** Khai báo thành phần gốc mà Angular xem nó như trang chính khi mở lên.

3.5.5.4. Service

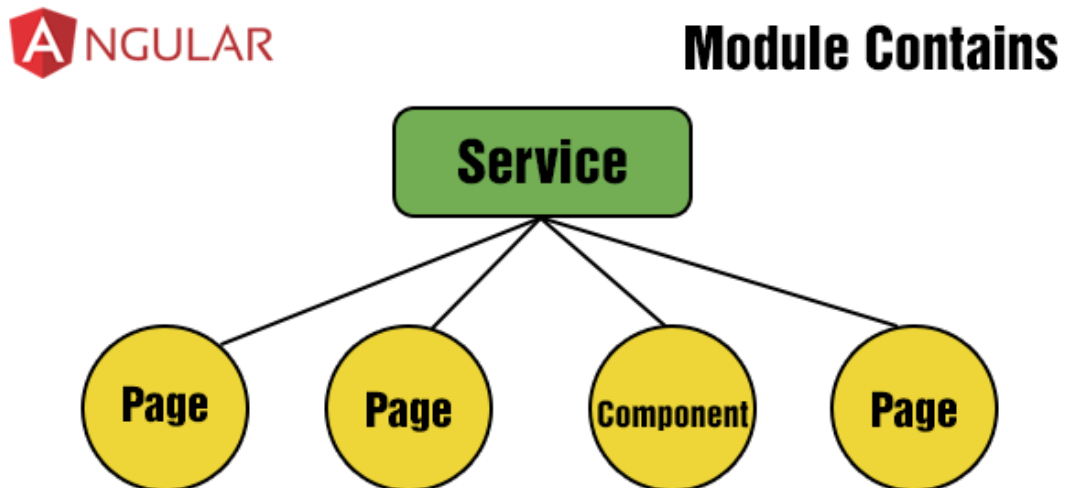
Khi đặt thẻ **@Injectable** trước một class, nó sẽ trở thành một Service.

Một class Service là một class mà Angular sẽ khởi tạo nó khi truy cập vào thành phần được đề cập trong providedIn hoặc được khai báo trong providers của một Module.

```
@Injectable({ providedIn: 'root' })
export class UserService implements OnInit {
```

Hình 3. 14. Cú pháp định nghĩa Service

Đối với providedIn là root, Service sẽ được khởi tạo ngay khi truy cập và có thể sử dụng ở toàn bộ các trang.



Hình 3. 15. Một Service được sử dụng bởi nhiều trang và thành phần

Các biến lưu trữ, các sự kiện trong class Service này sẽ có thể dùng chung cho các trang và các thành phần.

Để có thể sử dụng Service mà có thể dùng chung các biến với các trang khác, người dùng cần khai báo nó trong constructor của một class thuộc thành phần.

```
export class LoginComponent implements OnInit, AfterContentInit {
  constructor(
    private userService: UserService,
  ) { }
```

Hình 3. 16. Khai báo Service trong constructor để Angular inject vào Component

Nếu không khai báo providedIn, người dùng cần khai báo class Service trong Providers của **NgModule** để sử dụng.

3.5.5.5. RouterModule

Thành phần giúp phân trang cho một trang web Angular. Thông thường, các trang của người dùng đều là một thành phần, phải sử dụng selector và đặt vào AppComponent để nó hoạt động.

Song, nếu người dùng có quá nhiều thành phần, hoặc các tập thành phần lớn cần tách thành một trang thì nên sử dụng *RouterModule*.

```
const routes=[
  { path: '', redirectTo:'home', pathMatch:'full' },
  { path:'artist', loadChildren: ()=>ArtistModule },
  { path:'login', component: LoginComponent },
```

Hình 3. 17. Khai báo các route

```
@NgModule({
  imports: [
    RouterModule.forRoot(routes)
  ],
  exports:[ RouterModule ],
  declarations: []
})
export class AppRouterModule { }
```

Hình 3. 18. Tạo AppRouterModule và nạp RouterModule kèm các khai báo route, sau đó xuất RouterModule với các thiết lập trên để các Module khác nạp vào sử dụng tiếp

```
],
imports: [
  BrowserModule,
  HttpClientModule,
  FormsModule,
  AppRouterModule
],
})
export class AppModule { }
```

Hình 3. 19. AppModule nạp AppRouterModule vừa tạo

```
<!-- Router content -->
<div class="router-content">
  <router-outlet></router-outlet>
</div>
```

Hình 3. 20. Khai báo router-outlet tại thành phần cần điều hướng trang con, ở đây là AppComponent

3.5.6. Các tính năng chính

3.5.6.1. Ràng buộc – Binding

Angular có thể ràng buộc các giá trị trong class tương ứng ra ngoài thành phần html của mình.

Ví dụ: Khai báo biến *title* trong AppComponent

```
export class AppComponent {
  title = 'Tour of Heroes';
}
```

Hình 3. 21. Tạo biến trong một Component (class)

Sau đó, mở AppComponent.html lên, chèn cú pháp sau:

```
/app/app.component.html

<h1>{{title}}</h1>
```

Hình 3. 22. Cú pháp in biến trong class ra ngoài template (html)

Kết quả khi mở trình duyệt:



Hình 3. 23. Kết quả hiển thị

Ngoài ra, Angular còn hỗ trợ ràng buộc hai chiều, tức là giá trị trong class có thể in ra ngoài html, và từ html có thể thay đổi giá trị trong code-behind.

Sử dụng `[(ngModel)]` và đặt vào một thẻ input của tệp HTML như sau:

```
<h1>{{title}}</h1>
<input type="text" [(ngModel)]="title">
```

Hình 3. 24. Dùng ngModel để ràng buộc đa chiều

Tuy nhiên, lúc này khi chạy, trình duyệt sẽ báo lỗi không thể hiểu giá trị thẻ `ngModel` là gì. Đó là do chức năng **ngModel** vẫn chưa được khai báo.

Lúc này, mở `AppModule.ts` và khai báo `FormModule`

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Hình 3. 25. Khai báo module chức năng để hỗ trợ thuộc tính ngModel

Xem thành quả đạt được:

Tour of Heroes

Tour of Heroes

Hình 3. 26. Hiện thị ở giao diện

Thay đổi hộp thoại bên dưới và xem sự thay đổi đó:

Xin chào!

Xin chào!

Hình 3. 27. Hiện thị sau khi thay đổi hộp thoại

3.5.6.2. Sự kiện

Angular có thể ràng buộc các sự kiện vào cho các thẻ HTML, theo một hướng khác, song tương đương như khi thao tác ở HTML và JavaScript cơ bản.

```
export class AppComponent {  
  title = 'Tour of Heroes';  
  
  setTitle(){  
    this.title="Clicked!";  
  }  
}
```

Hình 3. 28. Khai báo một hàm trong Component (class)

```
<h1>{{title}}</h1>
<input type="text" [(ngModel)]="title"><br>
<button type="button" (click)="setTitle()">Click to change</button>
```

Hình 3. 29. Gán hàm gọi trên như một sự kiện ở phía template

Tour of Heroes

Hình 3. 30. Hiện thị lúc này ở trình duyệt

Clicked!

Hình 3. 31. Nhấp nút đã được gán sự kiện trên và xem thay đổi

Angular hỗ trợ hầu hết các sự kiện đặc trưng của JavaScript như: mouseup, mousedown, hover, dblclick, keyup, keydown, keypress,...

3.5.6.3. Khai báo điều kiện

Với thuộc tính ***ngIf**, nếu thỏa mãn điều kiện thì thẻ HTML kèm toàn bộ con của thẻ sẽ hiện ra, ngược lại sẽ không được tạo ra:

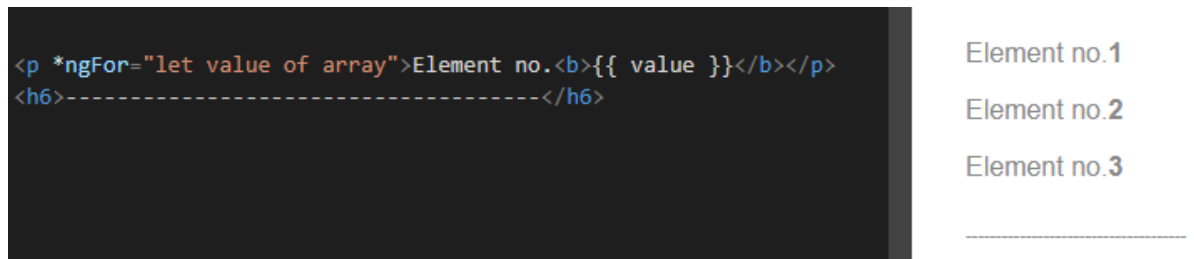
```
<h4 *ngIf="title != null">Not Null</h4>
<h4 *ngIf="title == null">Null</h4>
<h6>-----</h6>
```

Not Null

Hình 3. 32. Template (bên phải) và hiển thị ở phía trình duyệt (bên trái)

Lưu ý: Khi sử dụng thuộc tính ***ngIf** này mà điều kiện không thỏa, thẻ HTML sẽ **không được render** trong nội dung của HTML thay vì được ẩn đi.

Với thuộc tính ***ngFor**, thẻ HTML có thuộc tính sẽ được lặp lại đúng số phần tử của một mảng:



Hình 3. 33. Template (bên phải) dùng *ngFor* và hiển thị ở phía trình duyệt (bên trái)

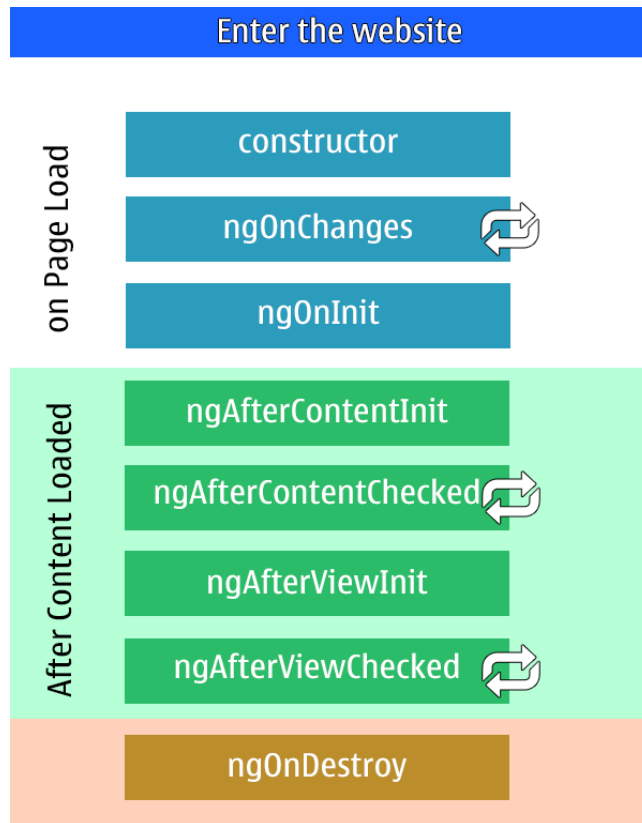
```
export class AppComponent {
  title = 'Tour of Heroes';
  array : number[]=[1,2,3];
}
```

Hình 3. 34. Giá trị của biến *array* ở phía Component

3.5.6.4. Vòng đời

Mỗi thành phần, trang của Angular đều có vòng đời, mỗi sự kiện của vòng đời sẽ được gọi trong **Component** khi lập trình viên bổ sung và ghi đè các sự kiện tương ứng đó:

- **Constructor:** Được gọi khi chạy vào script của Angular.
- **ngOnChanges:** Kích hoạt khi các biến được thay đổi giá trị.
- **ngOnInit:** Gọi khi bắt đầu ràng buộc các thuộc tính, thiết đặt các giá trị.
- **ngAfterContentInit:** Gọi khi nội dung bên ngoài đã được khởi tạo xong
- **ngAfterContentChecked:** Gọi mỗi khi nội dung bên ngoài bị thay đổi
- **ngAfterViewInit:** Được kích hoạt khi thẻ đã hoàn tất render lần đầu.
- **ngAfterViewChanges:** Chạy khi View render hoàn tất các thay đổi.
- **ngOnDestroy:** Khi thành phần này bị xóa khỏi trang hoặc rời khỏi trang.



Hình 3. 35. Vòng đời của một trang Angular

CHƯƠNG 4: ASP.NET CORE VÀ MONGODB DRIVER



Hình 4. 1. ASP.NET Core (nguồn: asp.net)

4.1. Giới thiệu

Ngày càng nổi lên nhờ sự thích nghi đa nền tảng, tốc độ được tối ưu, cộng đồng đông đúc và phát triển dưới ngôn ngữ hướng đối tượng C#, được hậu thuẫn Microsoft, ứng dụng ASP.NET Core ngày càng nổi lên với những điểm tuyệt vời sau:

- Miễn phí
- Ngôn ngữ dễ học, hướng đối tượng
- Đa nền tảng
- Tốc độ chạy vô cùng mượt mà và tối ưu.
- Tính mở rộng cao, tùy biến tốt.
- Cộng đồng đông đúc, thư viện hỗ trợ ngày càng phong phú.

4.2. Vai trò

- **Ứng dụng Web bình thường:** Một trang web html bình thường.
- **ASP.NET Core MVC:** Một ứng dụng hướng theo mô hình MVC dưới ngôn ngữ C# và Razor WebPage
- **Web API:** Một ứng dụng theo hướng dịch vụ web. Đây là hướng chủ yếu mà bài viết sẽ hướng dẫn.

Trong dự án này, Amnhac.com sẽ sử dụng .NET Core như một nền tảng back-end Web Service cho trang web của mình. Nó sẽ là một máy chủ nhằm lắng nghe các yêu cầu gọi đến, đặc biệt từ ứng dụng Web Angular, và trả về kết quả truy vấn từ MongoDB hay các tài nguyên như hình ảnh, âm thanh tương ứng.

4.3. Môi trường lập trình

4.3.1. Cài đặt

4.3.1.1. Runtime

Nền tảng ứng dụng .NET Core được chạy nhờ bộ Runtime được cung cấp cài trên máy tính. Microsoft hiện cung cấp bộ Runtime này miễn phí và có thể chạy trên mọi hệ điều hành vi tính.

Người dùng có thể tải về tại đây: <https://www.microsoft.com/net/download>

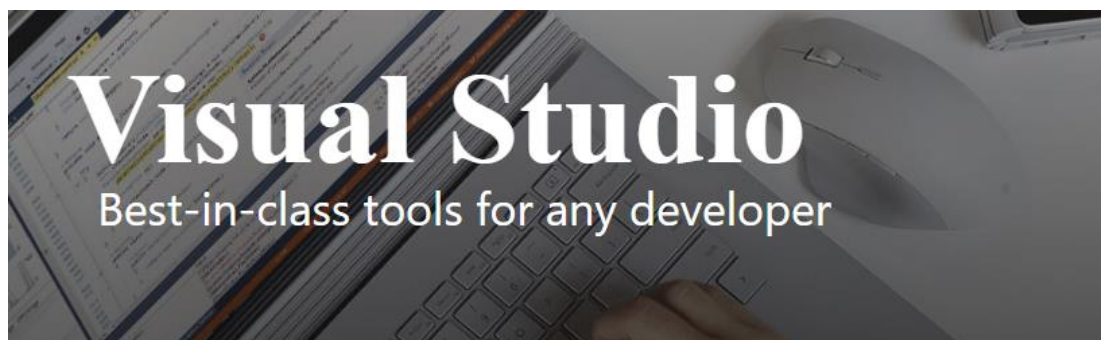
DotNet Core hiện gồm các phiên bản sau:

- .NET Core 1.0: Phiên bản đời đầu
- .NET Core 1.1
- .NET Core 2.0
- **.NET Core 2.1**: Phiên bản ra mắt vào 5/2018, tối ưu và thêm nhiều chức năng mới, hiện tại có tốc độ chạy vượt mặt các đối thủ khác.

4.3.1.2. Trình soạn thảo

Người dùng có thể sử dụng **Visual Studio Code** để lập trình ứng dụng Web ASP.NET Core hoặc **Visual Studio 2017** bản Community (miễn phí đối với người dùng cá nhân).

Bài viết này sẽ hướng dẫn người dùng với **Visual Studio 2017**

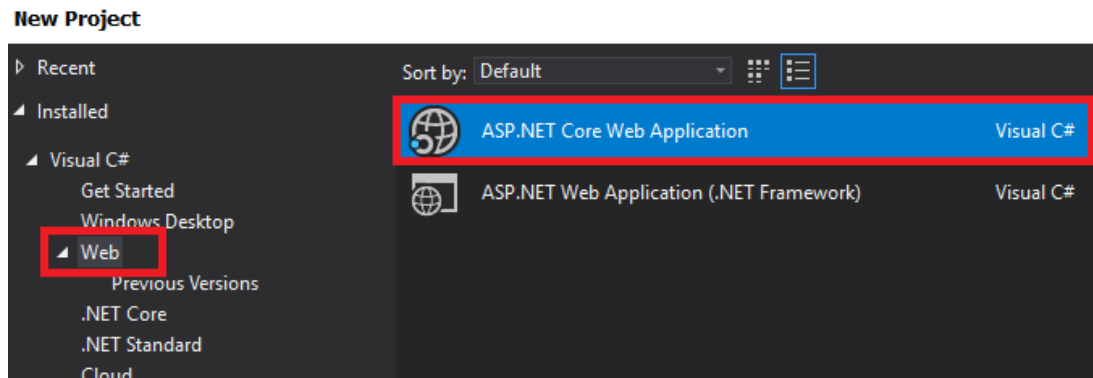


Hình 4. 2. Visual Studio (nguồn: visualstudio.microsoft.com)

4.3.2. Tạo ứng dụng Web ASP.NET Core

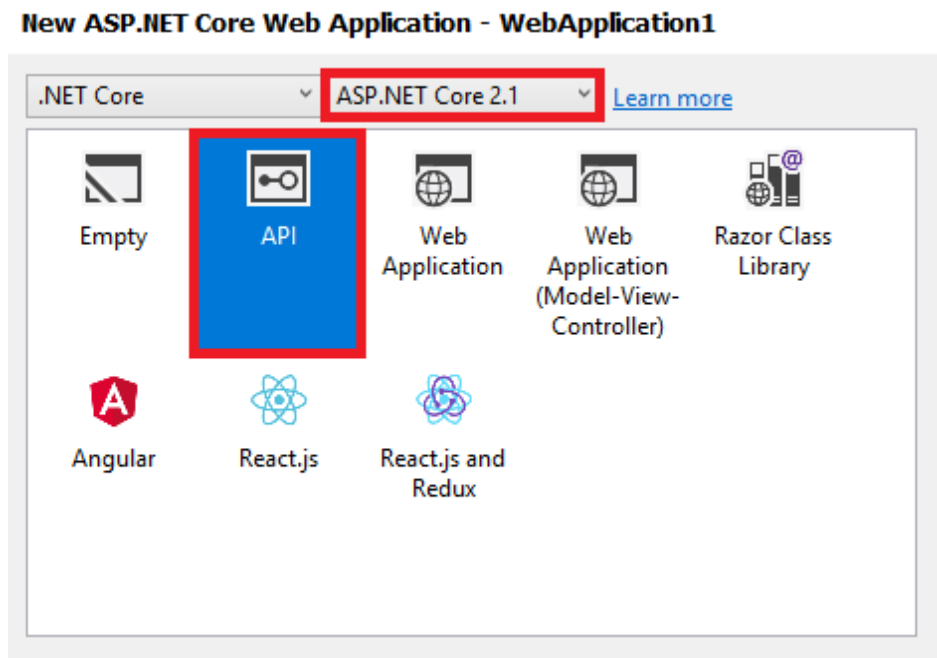
Bắt đầu từ **Visual Studio 2017**, chọn New Project để tiến hành tạo dự án mới.

Chọn Web – ASP.NET Core để tạo ứng dụng Web ASP.NET Core



Hình 4. 3. Tạo ứng dụng ASP.NET Core qua Visual Studio 2017

Chọn API để tạo ứng dụng Web ASP.NET Service (Dịch vụ Web)

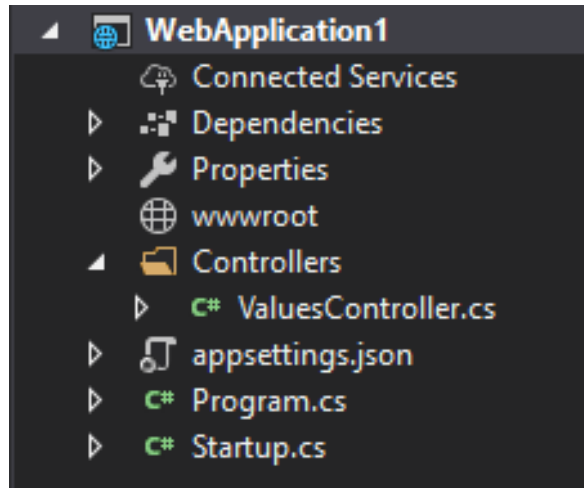


Hình 4. 4. Chọn Template cho ứng dụng

Hoàn tất bước tạo lập dự án ASP.NET Core

4.4. ASP.NET Core WebAPI cơ bản

4.4.1. Cấu trúc



Hình 4. 5. Cấu trúc cơ bản của ứng dụng

- **Program.cs:** Tập chạy chính, chứa các câu lệnh khởi chạy một ứng dụng DotNet Core.
- **Startup.cs:** Tập chứa các lệnh giúp thiết lập các tính năng của ứng dụng Web.
- **Appsettings.json:** Tập lưu trữ các chuỗi thiết lập.
- **Wwwroot:** Thư mục chứa các “static files”, các tệp tin trong wwwroot sẽ được phục vụ cho các yêu cầu lưu trữ hoặc đọc.
- **Controllers:** Thư mục chứa các tệp cs Controller, các tệp này chứa các lệnh cấu hình cho ứng dụng lắng nghe thông qua các đường dẫn.

4.4.2. Cú pháp

Về cơ bản thì ASP.NET Core sử dụng C# là ngôn ngữ lập trình chính, tương tự như Java, C# có class, function, tính kế thừa, bao gói,...

Với một ứng dụng Web ASP.NET, ta phải làm quen với khái niệm mới như **thể chú thích** nằm trước một class hoặc một function như dưới đây:


```
[Route("api/[controller]")]
[ApiController]
public class ValuesController : ControllerBase
{
    // GET api/values
    [HttpGet]
    public ActionResult<IEnumerable<string>> Get()
    {
        return new string[] { "value1", "value2" };
    }

    // GET api/values/5
    [HttpGet("{id}")]
    public ActionResult<string> Get(int id)
    {
        return "value";
    }
}
```

Hình 4. 6. Cú pháp trong Controller của ASP.NET Core

Trong các tệp Controller:

- Thẻ **[Route]**: Định dạng đường dẫn gốc nếu đặt cho một class khi nó lắng nghe yêu cầu.
- Thẻ **[ApiController]**: Cho biết một class thuộc Controller chuyên phục vụ cho việc lắng nghe và trả về các kết quả không phải là một View.
- Một lớp là Controller sẽ kế thừa ControllerBase để được xem là Controller.
- Thẻ **[HttpGet]**: Cho biết phương thức trong class sẽ được gọi trả về khi dùng phương thức HttpGet truy cập. Thẻ này có thể đặt đường dẫn thêm vào, song nếu không có khai báo, chỉ cần truy cập đường dẫn gốc với phương thức GET là được. Ví dụ trên là : `/api/values` (GET)
- Thẻ **[HttpPost]**: Tương tự như HttpGet, song phải dùng phương thức POST.
- Thẻ **[HttpGet("{id}")]**: Cú pháp {tên_biến} bên trong cho biết đường dẫn này nhận các giá trị phía sau khi yêu cầu như một tham số, tên_biến phải tương ứng với tên tham số đại diện. Ví dụ: `/api/values/5` sẽ nhận **5** như một tham số cho biến **id** trong câu lệnh.

```
// This method gets called by the runtime. Use this method to
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Default);
}

// This method gets called by the runtime. Use this method to
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseHsts();
    }

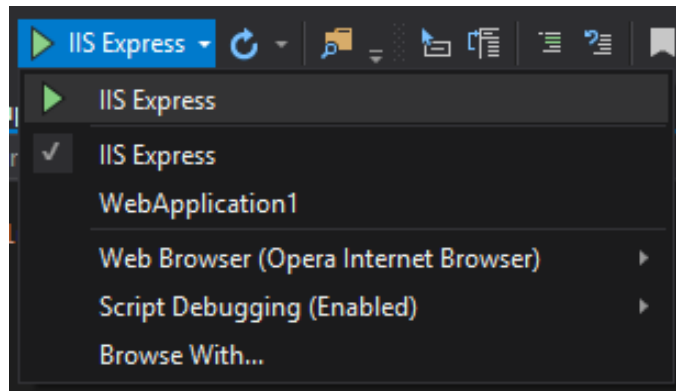
    app.UseHttpsRedirection();
    app.UseMvc();
}
```

Hình 4. 7. Các dòng lệnh trong tệp Startup.cs

Trong tệp thiết lập **Startup.cs**:

- **Services.AddMvc()**: Thêm dịch vụ Mvc() và thiết lập, khi dịch vụ Mvc được bật, ASP.NET Core sẽ có thể phục vụ các gói tin trong thư mục wwwroot.
- **App.UseMvc()**: Bật dịch vụ Mvc() đã được thiết lập phía trên.
- **App.UseDeveloperExceptionPage()**: Bật môi trường Debug.
- **App.UseHsts()**: Bật header bảo mật dùng trong IIS.
- **App.UseHttpsRedirection()**: Bật Https cho trang.

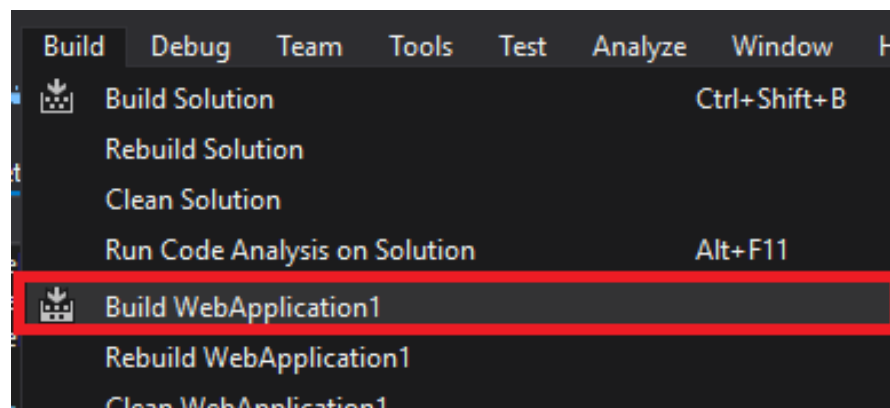
4.4.3. Khởi chạy



Hình 4. 8. Debug ứng dụng

Visual Studio cung cấp 2 tùy chọn để khởi chạy ứng dụng gồm sử dụng IIS Express hoặc sử dụng nhân Kestrel có sẵn trong Runtime của DotNet Core để chạy.

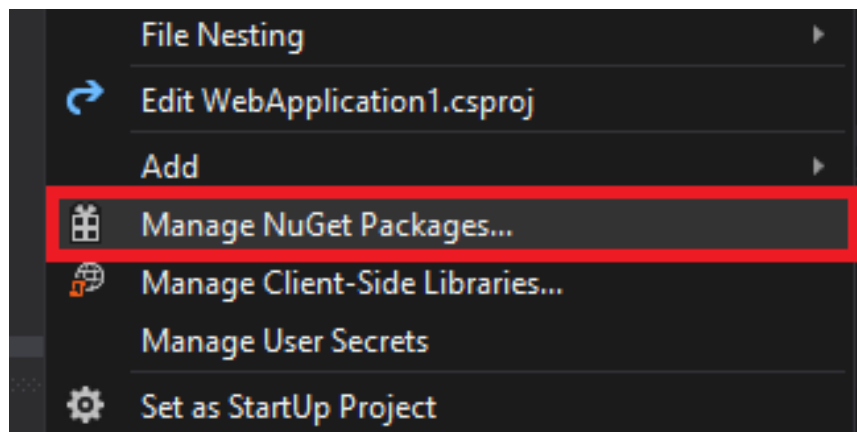
Ngoài ra, để Build một ứng dụng, người dùng có thể chọn Build ở trên menu của Visual Studio, sau đó vào thư mục bin để tìm các tệp dll liên quan được build.



Hình 4. 9. Tạo bản dựng cho ứng dụng qua giao diện Visual Studio 2017

4.5. Sử dụng NuGet Package

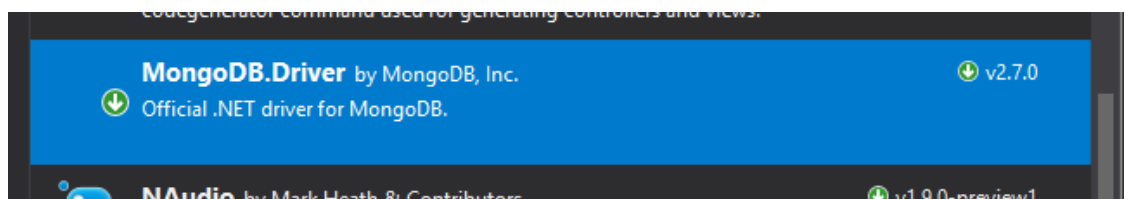
NuGet Package là các gói thư viện cho chương trình được lưu trên nuget, Visual Studio 2017 hỗ trợ việc quản lý các gói nuget này qua giao diện người dùng.



Hình 4. 10. Mở Menu để quản lý gói mở rộng NuGet

4.6. MongoDB Driver trong ASP.NET Core

MongoDB là gói thư viện có sẵn trên NuGet, người dùng có thể tải về từ NuGet và cài đặt cho chương trình của mình để có thể sử dụng MongoDB trong môi trường ngôn ngữ C#.



Hình 4. 11. MongoDB Driver trên NuGet Packages

Ở môi trường ngôn ngữ C#.NET, MongoDB truy vấn thông qua LINQ dưới dạng cú pháp Lambda.

4.6.1. Model

Để có thể vào bước đầu sử dụng MongoDB Driver trong ASP.NET Core, người dùng cần tạo các class Model, nó sẽ là cách mà người dùng sẽ định hình một collection thật sự trong MongoDB khi người dùng truy vấn và liên kết dữ liệu với C#.

```
public class Permission
{
    [BsonId]
    public ObjectId Id { get; set; }
    public string Name { get; set; }
    public string Desc { get; set; }
}
```

Hình 4. 12. Tạo một Model đơn giản nhằm lưu nó xuống Database sau này

Một Model dùng cho MongoDB Driver sẽ có các thuộc tính như bình thường, có các phương thức set, get. Bên cạnh đó, nó còn có các **thẻ ghi chú** dùng cho Model của MongoDB , như ta thấy ở trên:

- **[BsonId]**: Định danh một thuộc tính là khóa chính trong Collection
- **[BsonIgnore]**: Định danh một thuộc tính, thuộc tính này tồn tại khi chạy trong ứng dụng ASP.NET Core, song nó sẽ không được lưu vào Database.
- **[BsonRequired]**: Định danh một thuộc tính là bắt buộc có trong Collection.
- **[BsonIgnoreIfNull]**: Tương tự BsonIgnore, nhưng khi giá trị có dữ liệu, thuộc tính này sẽ được lưu vào Database.
- **[BsonElement]**: Định danh này kèm theo các tùy chọn đi kèm như tên của trường khi được lưu vào Database thật sự.
- **[BsonRepresentation]**: Định danh này đi kèm để tái định dạng lại thuộc tính đó khi được truy xuất từ Database lên Model.

4.6.2. Database Context

Context sẽ là nơi giao tiếp với MongoDB, class này sẽ chứa hàm gọi kết nối với MongoDB bằng các hàm tiện ích trong thư viện MongoDB Driver cung cấp.

```
public class DataAccess
{
    private IMongoDatabase _database;
    public DataAccess()
    {
        try
        {
            var client = new MongoClient("mongodb://localhost");
            if (client != null)
                _database = client.GetDatabase("ExampleDatabase");
        }
        catch (Exception) { }
    }

    public IMongoCollection<Permission> Permission {
        get { return _database.GetCollection<Permission>(nameof(Permission)); }
    }
}
```

Hình 4. 13. Khởi tạo class DataAccess kết nối đến Mongo Database

Ưu điểm của MongoDB Driver đó là nó sẽ tự động tạo Cơ sở dữ liệu nếu chúng không tồn tại, tự động tạo các Collection, dễ dàng thay đổi kiểu giá trị của một Collection.

4.6.3. Repository

Repository là các class chứa các câu lệnh truy vấn cơ sở dữ liệu

Repository sẽ dùng Database Context để thực hiện các câu lệnh này.

```
public class PermissionRepos
{
    private DataAccess context;
    public PermissionRepos()
    {
        context = new DataAccess();
    }
    public List<Permission> GetAll()
    {
        return context.Permission.Find(_ => true).ToList();
    }
}
```

Hình 4. 14. Một Repository cơ bản chứa DataAccess

Sau khi đã tạo một class Repository, khởi tạo và sử dụng trong các Controller như sau:

```
[HttpGet]
public ActionResult<IEnumerable<Permission>> Get()
{
    var permission = new PermissionRepos();
    return permission.GetAll();
}
```

Hình 4. 15. Gọi Repository ra trong Controller để truy vấn

4.7. Dependency Injection trong ASP.NET Core

4.7.1. Khái niệm Inversion of Control

Đây là một thuật ngữ chỉ về một nguyên lý thiết kế chương trình theo nguyên tắc dòng sự kiện bất định. Inversion of Control (IoC) sẽ cho phép người dùng để mở rộng chương trình của mình ra hơn bằng kiểu thiết kế các sự kiện hay các giá trị theo hướng chung chung, gọi ngược (callback) các hành động tùy chọn nào đó khi một sự kiện xảy ra.

4.7.2. Khái niệm Dependency Injection

Một khái niệm thuộc *Inversion of Control*, thay vì một class nào đó, khai báo một đối tượng khác ngay trong cấu trúc và tạo mới đối tượng thì với Dependency Injection (DI), class này chỉ cần khai báo một loại *đa hình (Interface hoặc abstraction)* của đối tượng, sau đó truyền tham số vào khi khởi tạo class.

4.7.3. Sử dụng Dependency trong ASP.NET Core

Sử dụng Repository như cách trên vẫn ra kết quả, song, chúng ta lại không thể quản lý được thời gian sống của chúng như mong muốn, và đồng thời Controller và Repository quá phụ thuộc lẫn nhau, sau này sẽ khó cho việc nâng cấp, thay đổi.

Rất may mắn, ASP.NET Core đã có sẵn các hàm để giúp chúng ta cải thiện điều này qua Dependency Injection.

Vào **Startup.cs** và gọi lệnh sau:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton(typeof(PermissionRepos));

    services.AddMvc().SetCompatibilityVersion(CompatibilityVers
}
```

Hình 4. 16. Gọi *AddSingleton* để tạo một Singleton của Repository khi Runtime

Sau đó sửa lại Controller của chúng ta như sau:

```
private readonly PermissionRepos permission;
public ValuesController(PermissionRepos permission)
{
    this.permission = permission;
}

// GET api/values
[HttpGet]
public ActionResult<IEnumerable<Permission>> Get()
{
    return permission.GetAll();
}
```

Hình 4. 17. Khai báo Repository trong Constructor của Controller để ASP.NET Core inject

Lúc này, mặc dù permission không có bất kỳ lệnh tạo mới nào, thay vào đó thêm một tham số trong constructor, và gán tham số đó vào.

Thật ra lúc này sau khi khởi chạy chương trình, service của ứng dụng sẽ giúp chúng ta khởi tạo biến này và truyền vào class thông qua constructor, điều này có phần tương tự như sử dụng một Service trong Angular vậy.

Tùy vào hình thức gọi hàm trong service mà thời gian sống của thông số truyền cho class khác nhau. Bao gồm *AddSingleton*, chúng ta có:

- **AddSingleton:** Hỗ trợ tạo đối tượng có thời gian sống dài, và chỉ tạo một đối tượng xuyên suốt thời gian chạy chương trình.

- **AddScoped**: Tạo đối tượng có thời gian sống trong một Request Session, mỗi Request Session của ứng dụng sẽ tạo một đối tượng mới.
- **AddTransient**: Tạo đối tượng mỗi lần được yêu cầu đến.

Đối với một đối tượng đa hình, được bổ sung bởi một Interface như sau:

```
public class PermissionRepos : IPermissionRepos
{
    private DataAccess context;
    public PermissionRepos()
    {
        context = new DataAccess();
    }

    public List<Permission> GetAll()
    {
        return context.Permission.Find(_ => true).ToList();
    }

    public bool Insert(Permission obj)
    {
    }
```

Hình 4. 18. Áp dụng với đa hình trong C#

Thì ta có thể gọi lệnh như sau:

```
// This method gets called by the runtime. Use this method to add
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IPermissionRepos, PermissionRepos>();
}
```

Hình 4. 19. Khai báo một đối tượng Đa hình như Singleton

Bằng cách này, chỉ cần thay đổi class tương ứng khi truyền class dịch vụ cho Interface là có thể thay đổi lập luận cho toàn bộ ứng dụng.

CHƯƠNG 5: SIGNALR

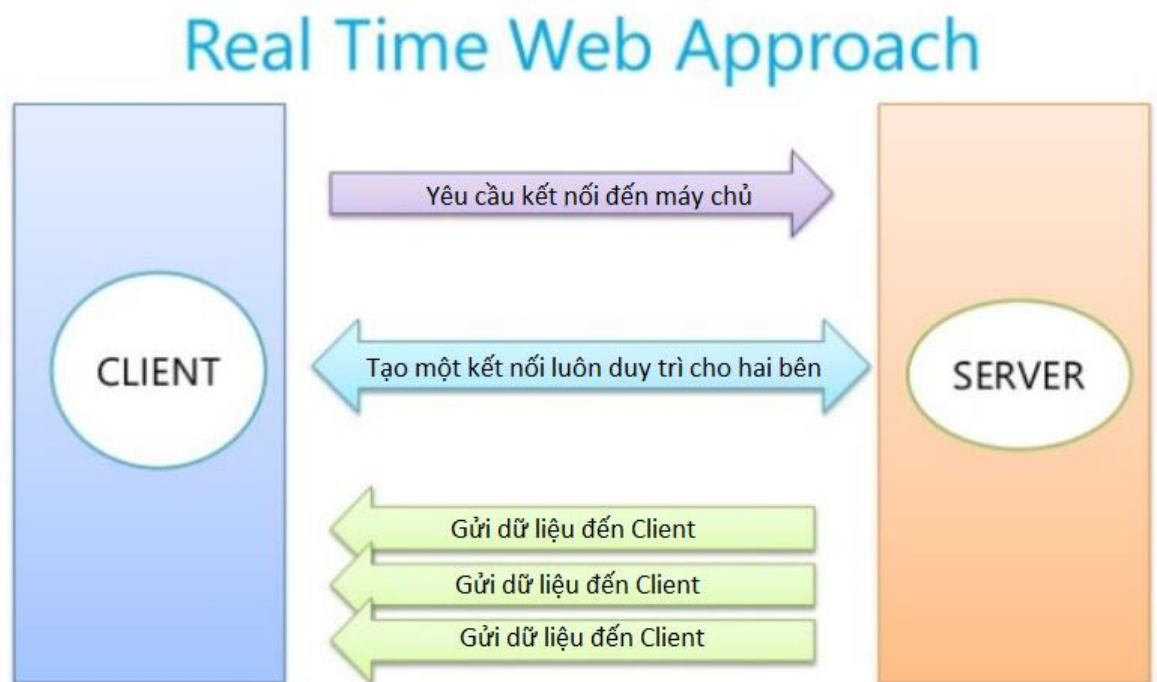
5.1. SignalR là gì?



Hình 5. 1. Công nghệ SignalR (nguồn: microsoft.com)

5.1.1. Giới thiệu

SignalR là bộ thư viện dành cho các ứng dụng web phát triển có tính năng thời gian thực. Nó cho phép giữ trò chuyện giữa máy chủ và máy con, giao tiếp với nhau theo cả hai chiều.



Hình 5. 2. Giao tiếp thời gian thực giữa Client và Server (nguồn: c-sharpcorner.com)

5.1.2. SignalR trong ASP.NET Core

SignalR hỗ trợ cho cả ASP.NET Core, có thể chạy trên các trình duyệt hiện hành và Internet Explorer từ phiên bản 11.

SignalR thường được sử dụng khi:

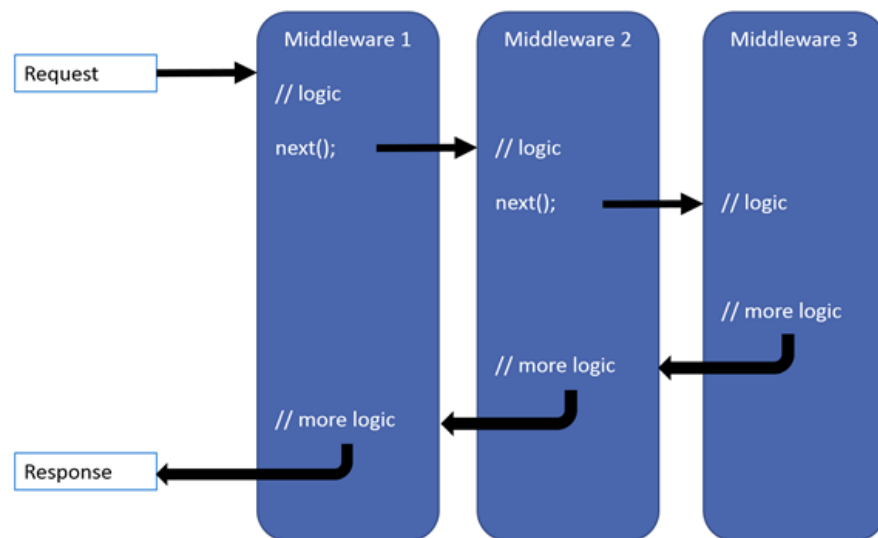
- Người dùng cần thay đổi theo thời gian thực cho ứng dụng như game, trắc địa, định vị, đo lường, quan trắc, ...
- Người dùng muốn một ứng dụng giao tiếp xã hội thời gian thực như trò chuyện qua mạng, Facebook, Twitter, các ứng dụng cần thông báo...
- Người dùng muốn ứng dụng có thể truyền tải đa phương tiện trực tuyến.
- Các ứng dụng theo dõi và giám sát, bao gồm cảnh báo hoặc thống kê số liệu.

5.1.3. Mục đích sử dụng SignalR

SignalR trong dự án này sẽ giúp Amnhac.com mang lại thay đổi thời gian thực cho người dùng, giúp trải nghiệm của người dùng xuyên suốt mà không bị gián đoạn khi có các thay đổi xảy ra. Ví dụ như khi quản trị viên cập nhật quảng cáo, toàn bộ người dùng sẽ thấy sự thay đổi đó ngay khi đang nghe nhạc mà không cần làm mới lại trang.

5.1.4. Thuật ngữ Middleware

Middleware trong ASP.NET Core có thể hiểu là một thành phần ở giữa, phục vụ công tác lắng nghe các yêu cầu đến và chuyển hướng đến hành động thích hợp. Ví dụ: Người dùng yêu cầu đến đường dẫn */signalR* thì sẽ chuyển hướng sang hướng các dịch vụ của signalR, còn các đường dẫn khác không phải thì sẽ phục vụ trang web bình thường.

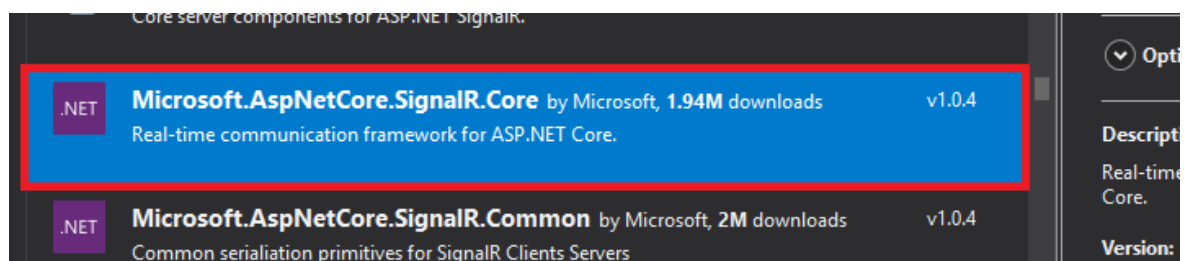


Hình 5. 3. Một Request của người dùng có thể thông qua một Middleware hoặc nhiều Middleware tùy theo thiết lập logic bên trong

5.2. Cài đặt

5.2.1. DotNET Client

Để cài đặt SignalR cho ứng dụng DotNET Core, người dùng có thể vào phần quản lý Gói tin NuGet để cài đặt.



Hình 5. 4. SignalR cho ASP.NET Core trên NuGet packages

5.2.2. Javascript Client

Thường sử dụng cho phía máy con, tức là trên phần mặt web giao tiếp với người dùng, SignalR trong Javascript sẽ thiết lập một kết nối đến phía máy chủ SignalR.

Nếu đang sử dụng NodeJS, người dùng chỉ cần gõ lệnh sau:

```
npm install @aspnet/signalr
```

5.3. Ứng dụng

5.3.1. Thiết lập lắng nghe

5.3.1.1. Tạo SignalR Hub

Tạo một class kế thừa Hub của SignalR như sau:

```
public class ChatHub: Hub
{
    public async Task SendMessage(string user, string message)
    {
        await Clients.All.SendAsync("ReceiveMessage", user, message);
    }
}
```

Hình 5. 5. Một SignalR Hub cơ bản

5.3.1.2. Thiết lập lắng nghe cho SignalR

Để máy chủ có thể lắng nghe được máy con trên SignalR, ta mở tệp Startup.cs để tiến hành thiết lập lắng nghe.

Thêm lệnh sau vào hàng vi thiết lập dịch vụ:

```
// This method gets called by the runtime. Use this method to
// configure the HTTP request pipeline.
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IPermissionRepos, PermissionRepos>();
    services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Default);
    services.AddSignalR();
}

// This method gets called by the runtime. Use this method to
```

Hình 5. 6. Thêm SignalR như một phần của Service trong ASP.NET Core

Sau đó sử dụng SignalR như một Middleware trong ứng dụng:

```

public void Configure(IApplicationBuilder app)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseHsts();
    }

    app.UseHttpsRedirection();

    app.UseSignalR(routes =>
    {
        routes.MapHub<ChatHub>("/chatHub");
    });

    app.UseMvc();
}

```

Hình 5. 7. Tạo một Middleware cho SignalR

- Lưu ý: Thứ tự đặt các câu lệnh `app.Use...` rất quan trọng, nếu sử dụng các *Middleware* như SignalR thì nên đặt trước `UseMvc()` để khi lắng nghe, ứng dụng sẽ ưu tiên xử lý *Middleware* của SignalR trước.

5.3.2. Thiết lập kết nối

Ở phía trang web bình thường, sau khi đã nhúng SignalR.js vào trang web của mình, trong phần Javascript của trang web, ta có thể gọi yêu cầu kết nối đến SignalR bằng câu lệnh sau:

```

var connection = new signalR.HubConnectionBuilder().withUrl("/chatHub").build();

connection.start().catch(function (err) {
    return console.error(err.toString());
});

```

Hình 5. 8. Yêu cầu kết nối ở phía Client (nguồn: docs.microsoft.com)

Kế tiếp, đặt một hàm lắng nghe sự kiện ở phía máy con:

```
connection.on("ReceiveMessage", function (user, message) {  
    var msg = message.replace(/&/g, "&amp;").replace(/</g,  
    var encodedMsg = user + " says " + msg;  
    var li = document.createElement("li");  
    li.textContent = encodedMsg;  
    document.getElementById("messagesList").appendChild(li  
});
```

Hình 5. 9. Thiết đặt lắng nghe sự kiện tên *ReceiveMessage* ở phía Client
(nguồn: docs.microsoft.com)

5.3.3. Giao tiếp

Sau khi đã hoàn tất các bước trên, từ phía máy chủ, ta có thể gọi ChatHub để gửi tin nhắn đến máy con:

```
private readonly PermissionRepos permission;  
private readonly IHubContext<ChatHub> hub;  
public ValuesController(PermissionRepos permission,  
    IHubContext<ChatHub> hub  
    )  
{  
    this.hub = hub;  
    this.permission = permission;  
  
    hub.Clients.All.SendAsync("ReceiveMessage", "Some message");  
}
```

Hình 5. 10. Câu lệnh gửi tin nhắn đến máy tất cả các máy con đang lắng nghe
hàm *ReceiveMessage*

Hay gọi một lệnh trong Hub ở phía Javascript của máy con:

```
connection.invoke("SendMessage", user, message).catch(function (err) {  
    return console.error(err.toString());  
});
```

Hình 5. 11. Gọi lệnh trên Client có tên *SendMessage* ở phía máy Server và chờ
kết quả trả về

Chi tiết hơn về ứng dụng trò chuyện thời gian thực áp dụng SignalR (real-time chat), mọi người có thể ghé qua phần phụ lục để biết thêm chi tiết.

PHẦN 2: ÁP DỤNG

CHƯƠNG 6: GIẢI QUYẾT VẤN ĐỀ

6.1. Đặt tả vấn đề

Một trang web cho lĩnh vực âm nhạc tại Việt Nam, Amnhac.com là một dự án không chỉ nhằm để kinh doanh, mà còn với mục tiêu xây dựng cầu nối giữa mọi người, là nơi mà các nhịp cầu cảm xúc bắt qua, và là nơi mọi lứa tuổi đều có thể thấu hiểu.

Website Amnhac.com sẽ cho phép người dùng đăng ký với tên tài khoản, mật khẩu, họ tên, số điện thoại, email và số căn cước. Người dùng đã đăng ký là người dùng thường, ngoài ra còn có quản trị viên và các cấp bậc khác do quản trị viên thiết đặt.

Nơi đây là nơi chia sẻ, mọi người sẽ có thể tải lên bài nhạc mình muốn chia sẻ cho mọi người bằng tệp tin nhạc, tên bài nhạc, nghệ sĩ trình bày, thể loại nhạc.

Mỗi bài nhạc được tải lên sẽ được duyệt bởi quản trị viên và sau khi một bài nhạc được duyệt, mọi người có thể tìm kiếm và nghe, mỗi lượt nghe sẽ được tính và được xếp hạng mỗi ngày.

Bên cạnh đó, người nghe nhạc còn có thể tạo cho mình một danh sách phát riêng, có thể hoặc không cần lưu vào cơ sở dữ liệu, song mỗi danh sách phát đều chứa danh sách các bài hát họ muốn nghe. Nếu họ muốn lưu một danh sách phát, người dùng có thể đặt tên cho danh sách phát này, cấp quyền riêng tư hoặc công khai cho danh sách phát riêng của họ.

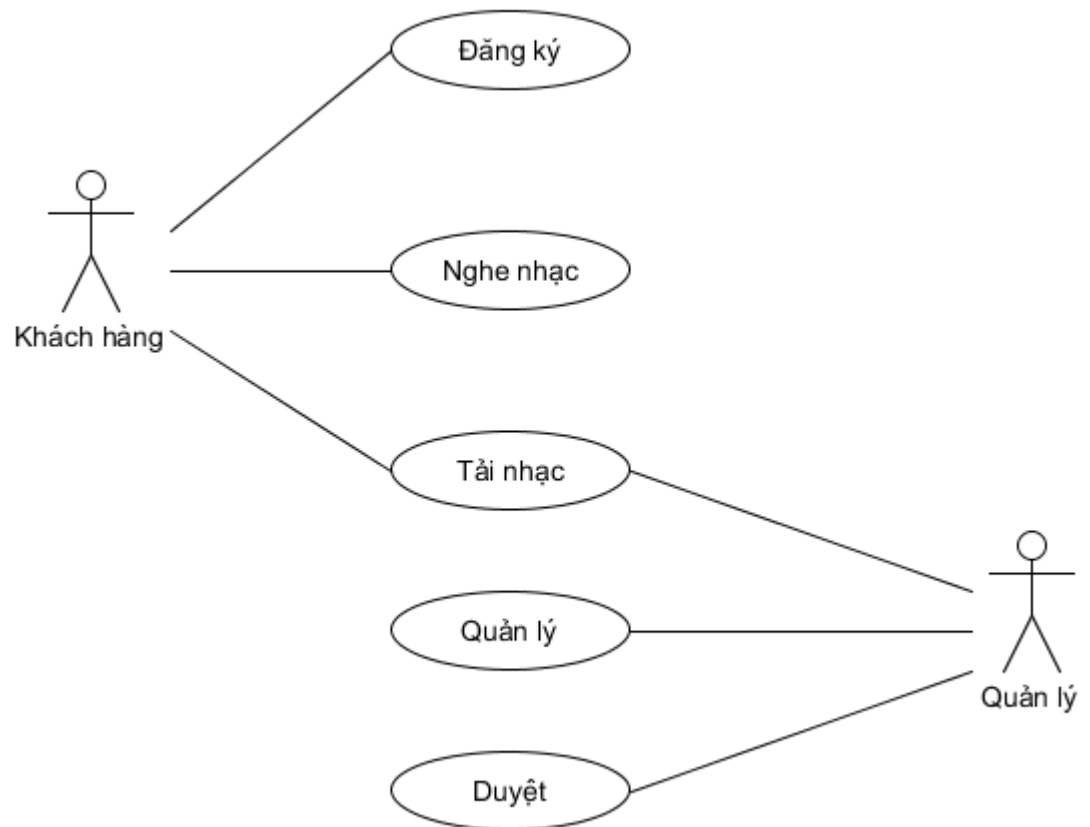
Quản trị viên ngoài việc quản lý và duyệt các bài nhạc được chia sẻ ra, họ có thể quản lý được quảng cáo của trang web, mỗi mục quảng cáo bao gồm hình ảnh, tiêu đề, mô tả đi kèm, thời gian quảng cáo chạy và hạn chót quảng cáo được treo. Khi cập nhật, thêm mới quảng cáo, sẽ thay đổi trên các trang chạy quảng cáo ngay lập tức.

Quản trị viên còn có thể tạo lập các Album bài nhạc, mỗi Album được tạo sẽ gồm các nghệ sĩ trình bày, danh sách các bài nhạc trong album, tên Album, mô tả của Album, bìa Album.

6.2. Thiết kế

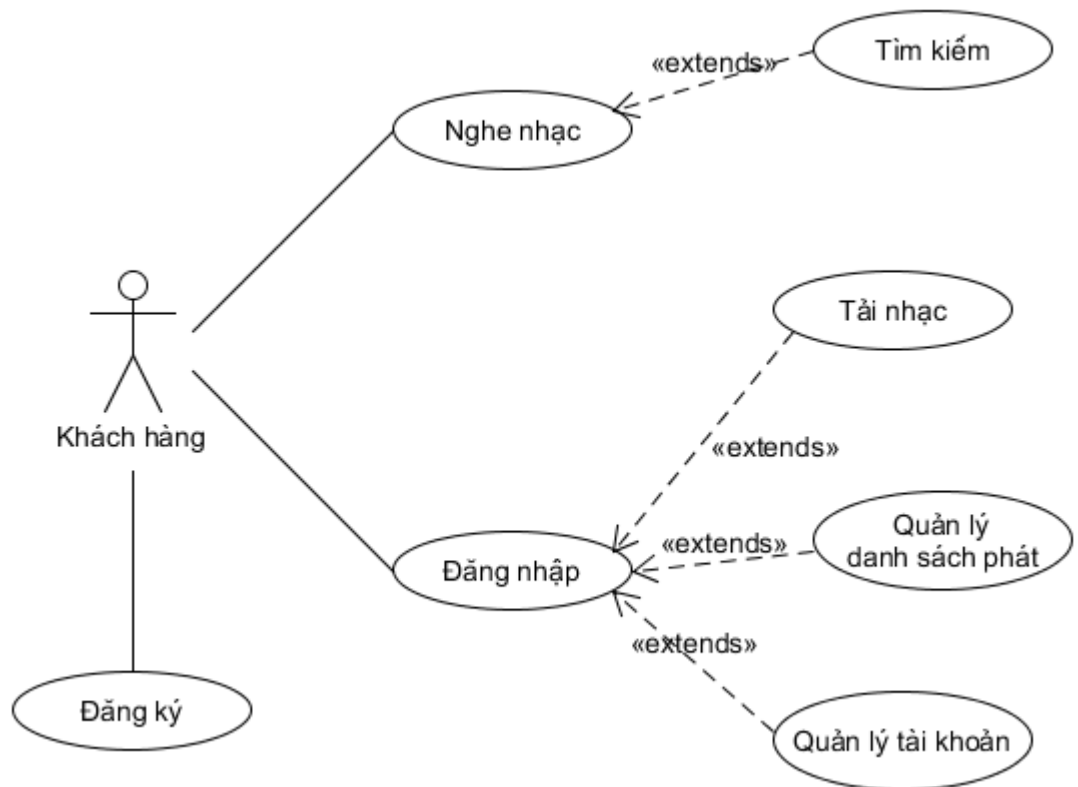
6.2.1. Mô hình Use-Case

Mô hình ở mức cơ bản (mức 0):



Hình 6. 1. Mô hình Use Case mức cơ bản

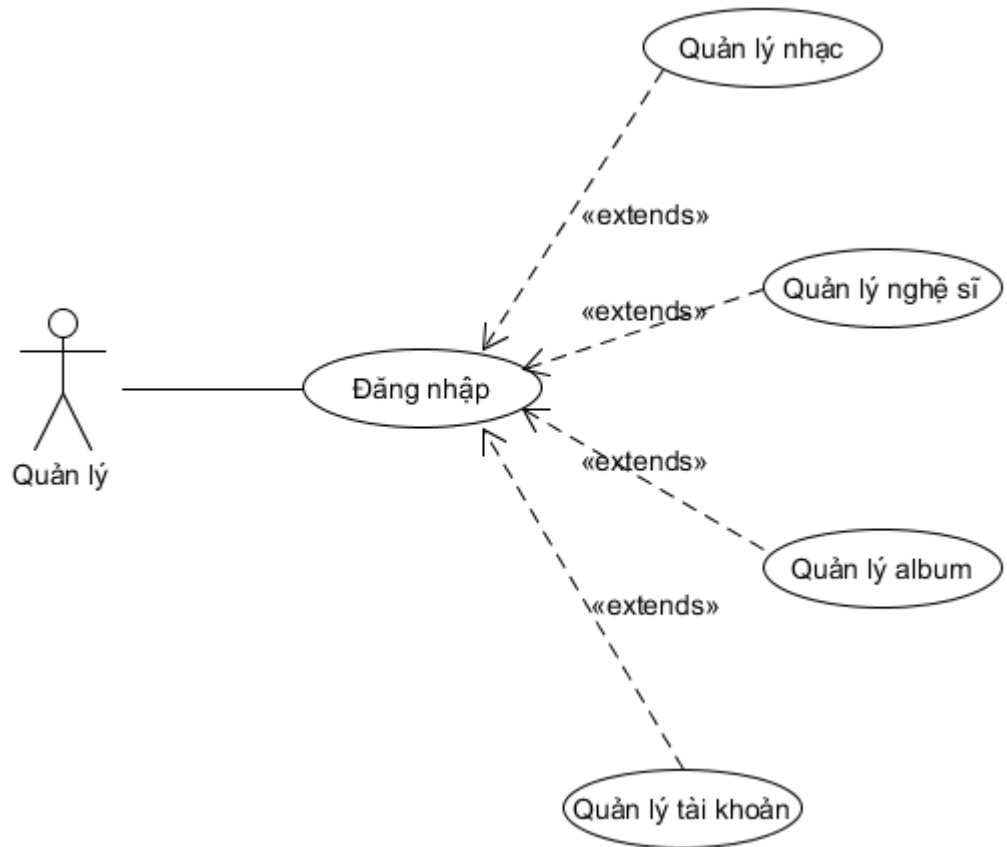
Mô hình mức trung (mức 1):



Hình 6. 2. Use Case mức trung ở phía Khách hàng

- Trường hợp người sử dụng thường:

- Khách hàng có thể vào đăng ký, đăng nhập, và nghe nhạc.
- Khách hàng đã xác thực có thể tải nhạc, quản lý danh sách phát của họ và thay đổi thông tin tài khoản.

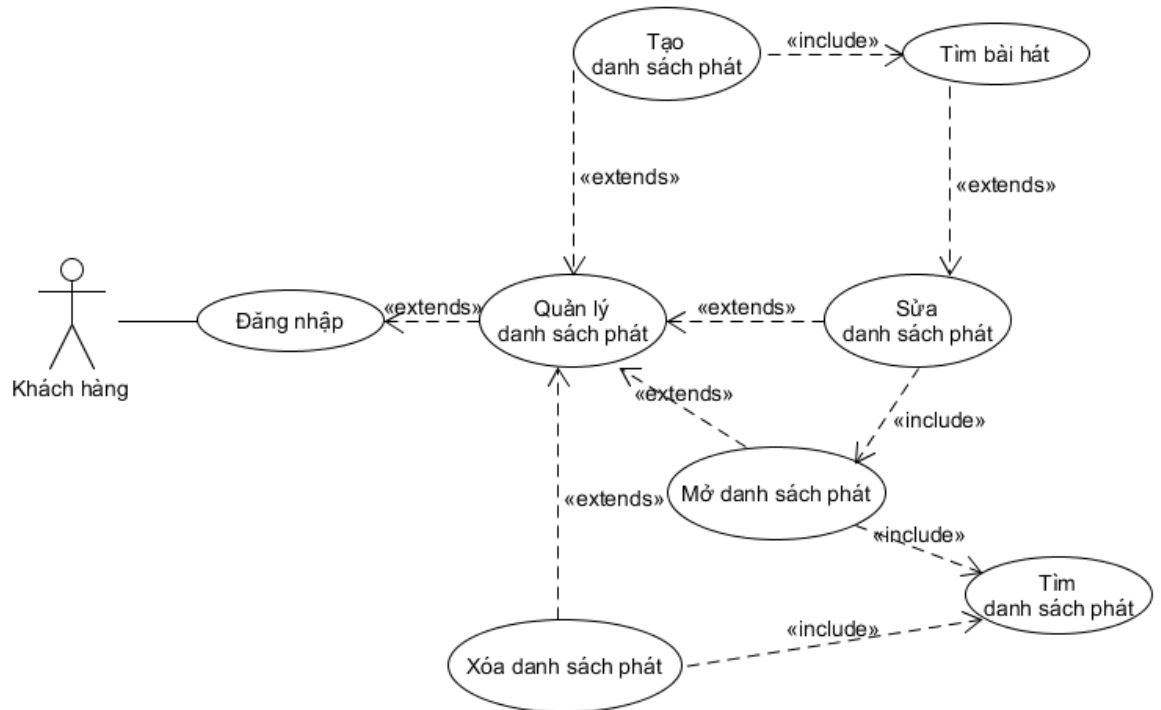


Hình 6. 3. Use case mức trung đối với quản trị viên

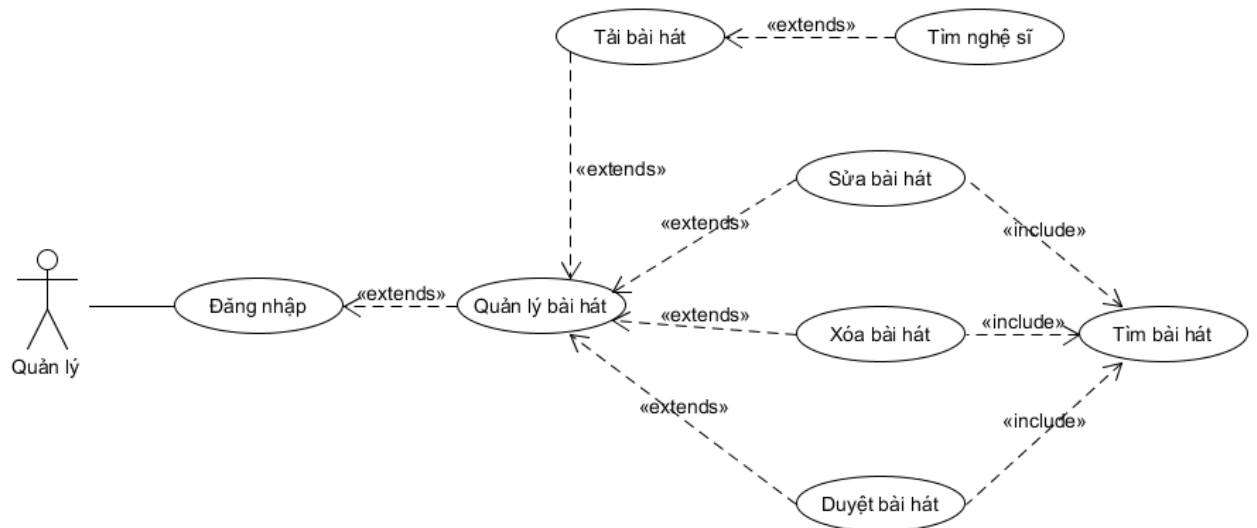
- **Trường hợp đối với quản trị viên:**

- Quản trị viên buộc phải đăng nhập để sử dụng
- Quản trị viên có thể quản lý các bài hát đã tải lên, duyệt chúng hoặc xóa chúng.
- Quản trị viên có thể quản lý các nghệ sĩ, ký danh các nghệ sĩ.
- Quản trị viên có thể tạo các Album
- Quản trị viên có thể quản lý các tài khoản, cấp quyền, sửa tên hiển thị của các tài khoản.

Mô hình mức chi tiết (mức 2):



Hình 6. 4. Use case chi tiết Quản lý danh sách phát phía người dùng



Hình 6. 5. Use case chi tiết cho chức năng quản lý bài hát của Quản trị viên

6.2.2. Xây dựng cơ sở dữ liệu

Permission		
Quyền hạn		
Tên cột	Kiểu dữ liệu	Ý nghĩa
Id	string	Khóa chính
Name	string	Tên quyền
Category	int	Cấp bậc quyền

Bảng 6. 1. Thực thể Permission

User		
Người dùng		
Tên cột	Kiểu dữ liệu	Ý nghĩa
Id	string	Khóa chính
Username	string	Tên tài khoản
Password	string	Mật khẩu
FullName	string	Họ tên
Email	string	Email
Phone	string	Số điện thoại
<u>Permission</u>	String (khóa ngoại)	Quyền hạn người dùng

Bảng 6. 2. Thực thể người dùng

Nation		
Quốc gia		
Tên cột	Kiểu dữ liệu	Ý nghĩa
Id	string	Khóa chính
Name	string	Tên quốc gia

Bảng 6. 3. Thực thể lưu trữ danh sách quốc gia

Artist		
Nghệ sĩ		
Tên cột	Kiểu dữ liệu	Ý nghĩa
Id	string	Khóa chính
Name	string	Tên nghệ sĩ
Desc	string	Mô tả về tiểu sử
DateOfBirth	DateTime	Sinh nhật
<u>Nation</u>	String (khóa ngoại)	Quốc gia, chứa khóa chính của quốc gia
Tag	string	Thẻ chứa từ khóa dùng để tìm kiếm Full-Text Lookup

Bảng 6. 4. Thực thể lưu trữ nghệ sĩ

SongType		
Thể loại nhạc		
Tên cột	Kiểu dữ liệu	Ý nghĩa
Id	string	Khóa chính
Name	string	Tên thể loại

Bảng 6. 5. Thực thể lưu thể loại nhạc

Group		
Phân loại thể loại nhạc		
Tên cột	Kiểu dữ liệu	Ý nghĩa
Id	string	Khóa chính
Name	string	Tên phân loại
<u>SongType</u>	SongType[] (khóa ngoại)	Thể loại nhạc

Bảng 6. 6. Thực thể phân loại thể loại nhạc

SongLink		
Lưu đường dẫn lưu trữ tài nguyên nhạc		
Tên cột	Kiểu dữ liệu	Ý nghĩa
Quality	string	Chất lượng tài nguyên
Link	string	Đường dẫn nhạc trên tài nguyên máy chủ

Bảng 6. 7. Thực thể đường dẫn nhạc

Song		
Bài hát		
Tên cột	Kiểu dữ liệu	Ý nghĩa
Id	string	Khóa chính
Name	string	Tên bài hát
<u>Artists</u>	String[] (khóa ngoại)	Danh sách nghệ sĩ biểu diễn, lưu bằng khóa chính thực thể Artist
<u>SongType</u>	String (khóa ngoại)	Thể loại của bài hát, lưu bằng khóa chính thực thể SongSubType
<u>Uploader</u>	string (khóa ngoại)	Người tải lên, lưu bằng khóa chính thực thể User
<u>SongLinks</u>	SongLink[] (khóa ngoại)	Đường dẫn tài nguyên bài hát
Tag	string	Thẻ lưu từ khóa dùng cho việc tìm kiếm Full-Text Lookup

Bảng 6. 8. Thực thể lưu trữ bài hát

Album		
Album		
Tên cột	Kiểu dữ liệu	Ý nghĩa
Id	string	Khóa chính
Name	string	Tên Album

Tag	string	Thẻ lưu từ khóa dùng cho việc tìm kiếm Full-Text Lookup
<u>Artists</u>	String[] (khóa ngoại)	Danh sách nghệ sĩ biểu diễn, lưu bằng khóa chính thực thể Artist
<u>SongList</u>	String[] (khóa ngoại)	Danh sách bài hát trong Album

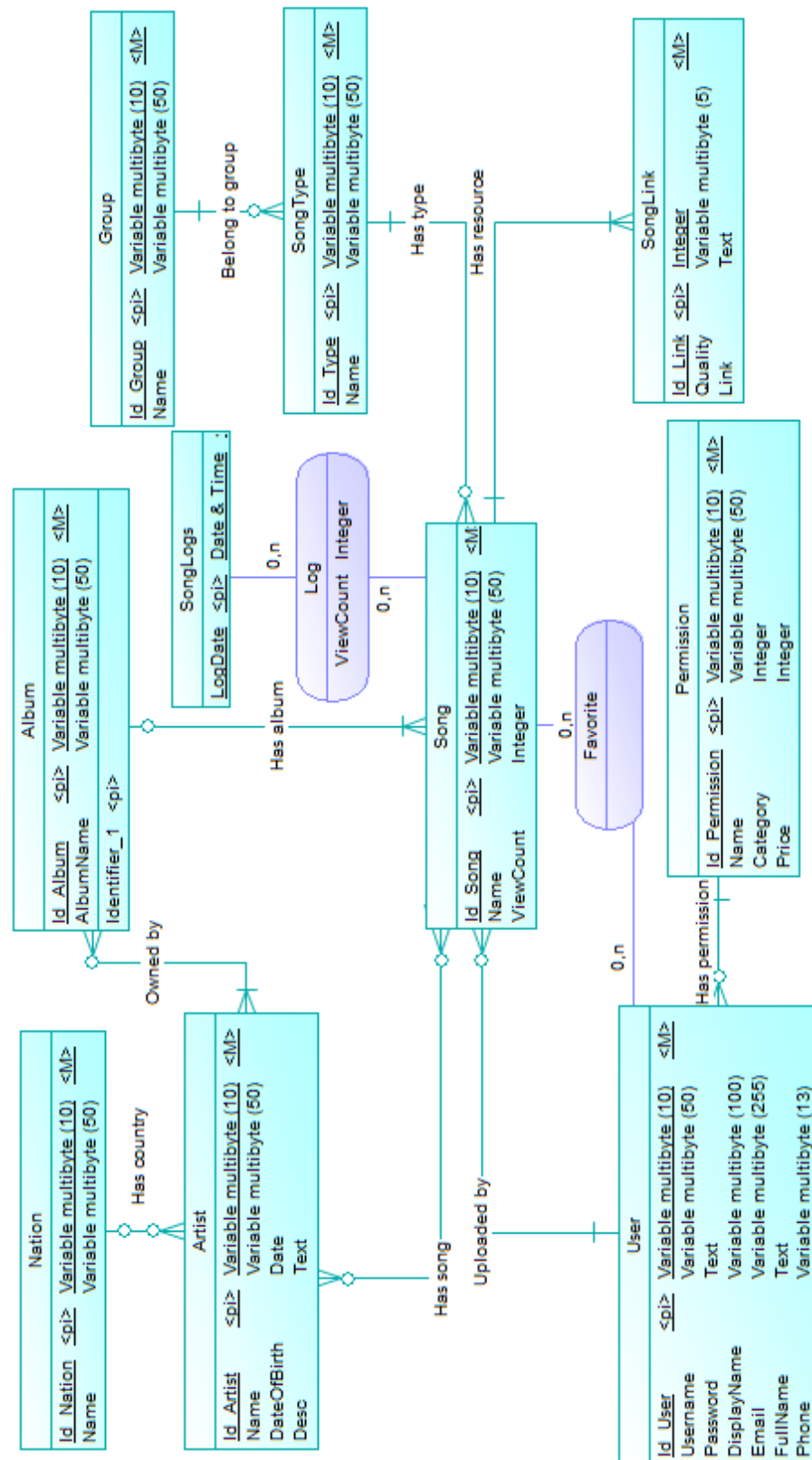
Bảng 6. 9. Thực thể lưu trữ Album

SongLogs		
Ghi lịch sử lượt nghe		
Tên cột	Kiểu dữ liệu	Ý nghĩa
CreatedOn	DateTime	Khóa chính, ngày ghi lịch sử
<u>SongId</u>	string	Khóa chính, bài hát được ghi
ViewCount	Long	Tổng lượt nghe tính đến ngày được ghi

Bảng 6. 10. Thực thể lưu lịch sử lượt nghe

Do MongoDB không có mối quan hệ thực thể, thiết kế này dựa theo quy tắc lưu trữ của MongoDB.

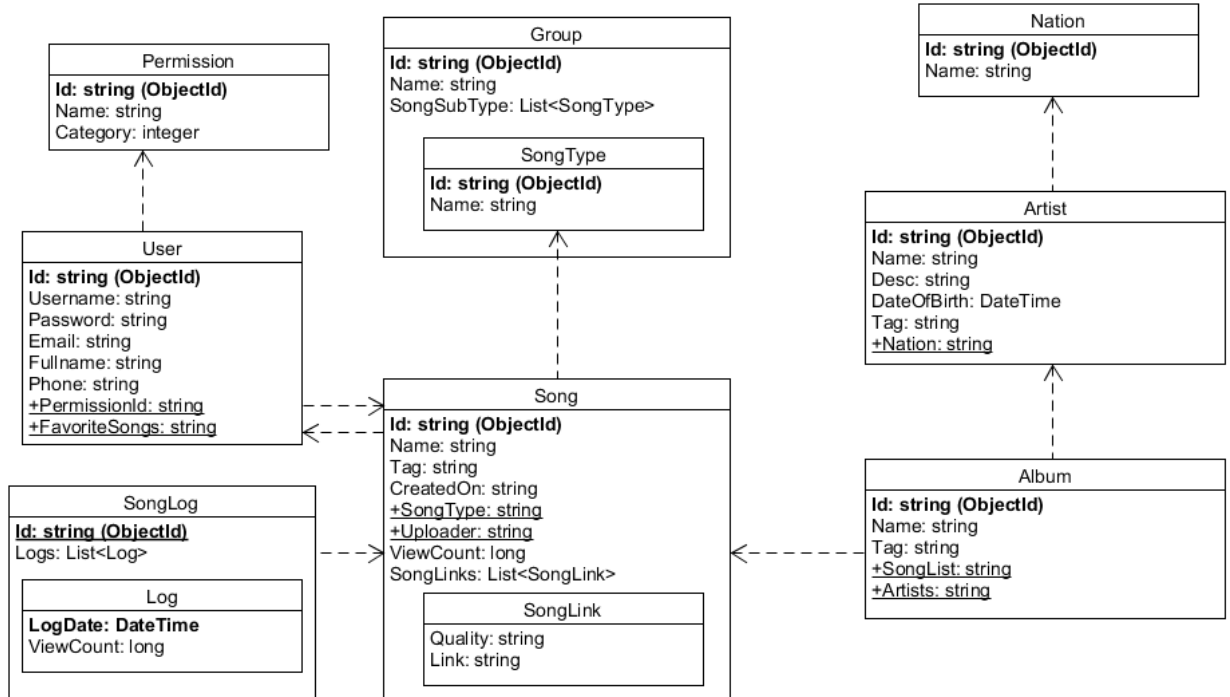
6.2.3. Mô hình cơ sở dữ liệu quan hệ



Hình 6. 6. Mô hình dữ liệu vật lý

6.2.4. Mô hình lưu trữ MongoDB

Ở mục phía trên, mô hình dữ liệu vật lý chỉ cho người dùng thấy mối quan hệ giữa các bảng dữ liệu thế nào, tuy nhiên, trong MongoDB, lưu trữ của các bảng được lưu như sau:



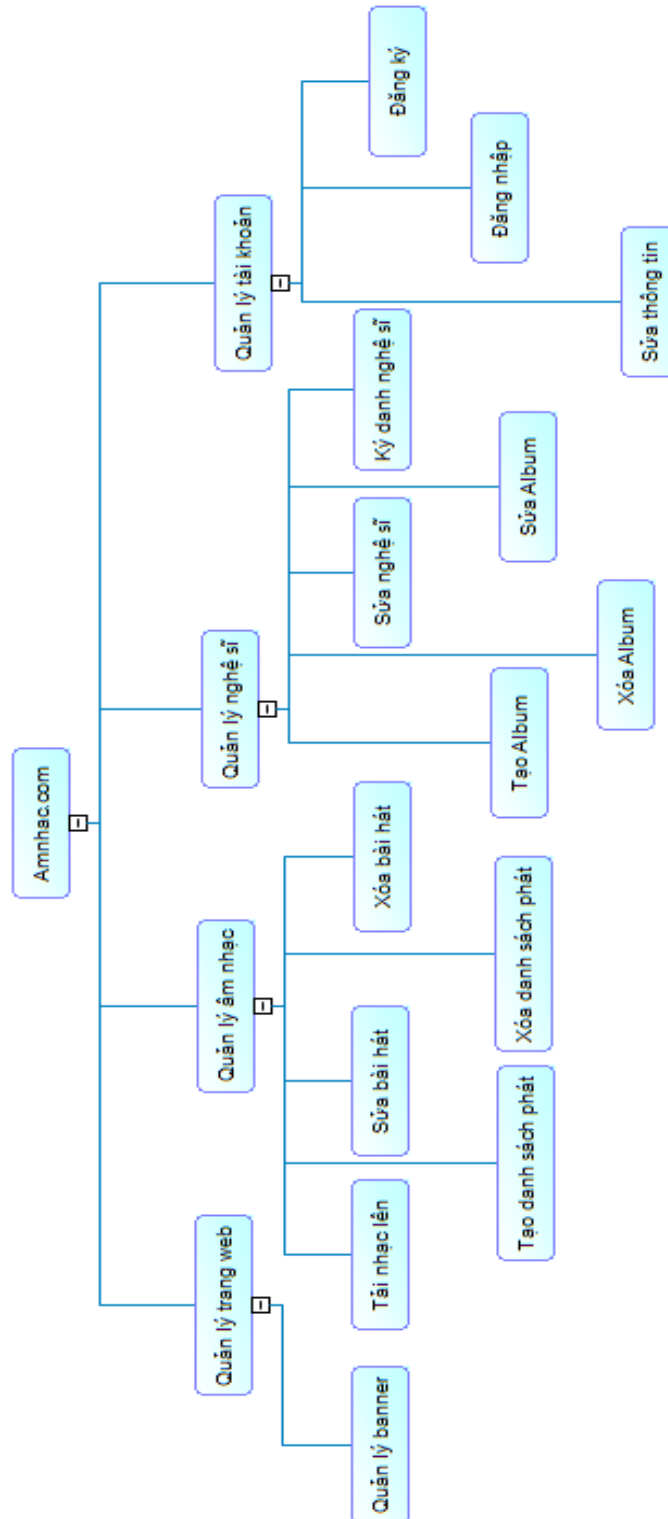
Hình 6. 7. Mô hình lưu trữ trong MongoDB

Các dấu mũi tên chỉ biểu thị chúng nó quan hệ ngầm hiểu, chứ cơ sở dữ liệu sẽ không giúp chúng ta kiểm tra tính khả dụng của các khóa ngoại.

Ngoài ra, trong một bảng của MongoDB (Collection), nó còn có thể chứa bảng con bên trong nó, ví dụ như bảng Song chứa các SongLink bên trong nó.

6.2.5. Sơ đồ phân rã chức năng

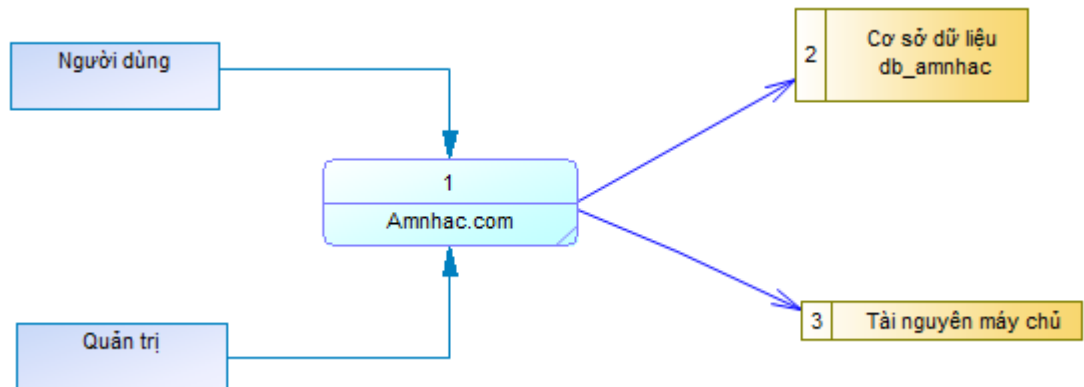
Amnhac.com sẽ có các chức năng sau nhìn về tổng quát:



Hình 6. 8. Sơ đồ phân rã chức năng tổng quát

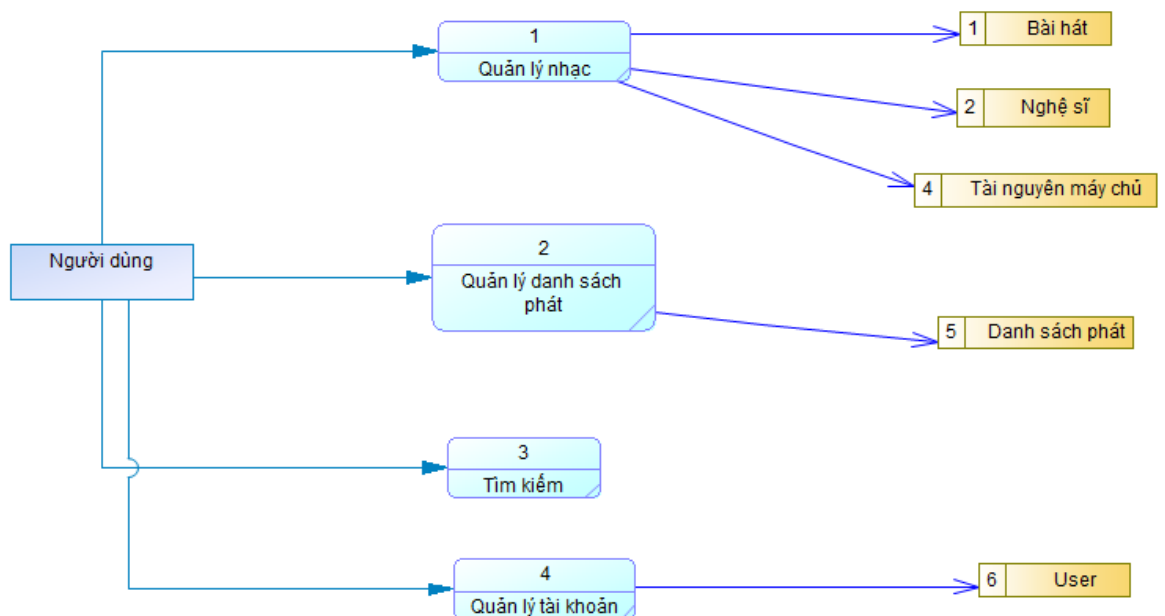
6.2.6. Dòng dữ liệu

Amnhac.com chia trang web của mình ra làm hai phần người dùng chính, và hai mục lưu trữ chính:



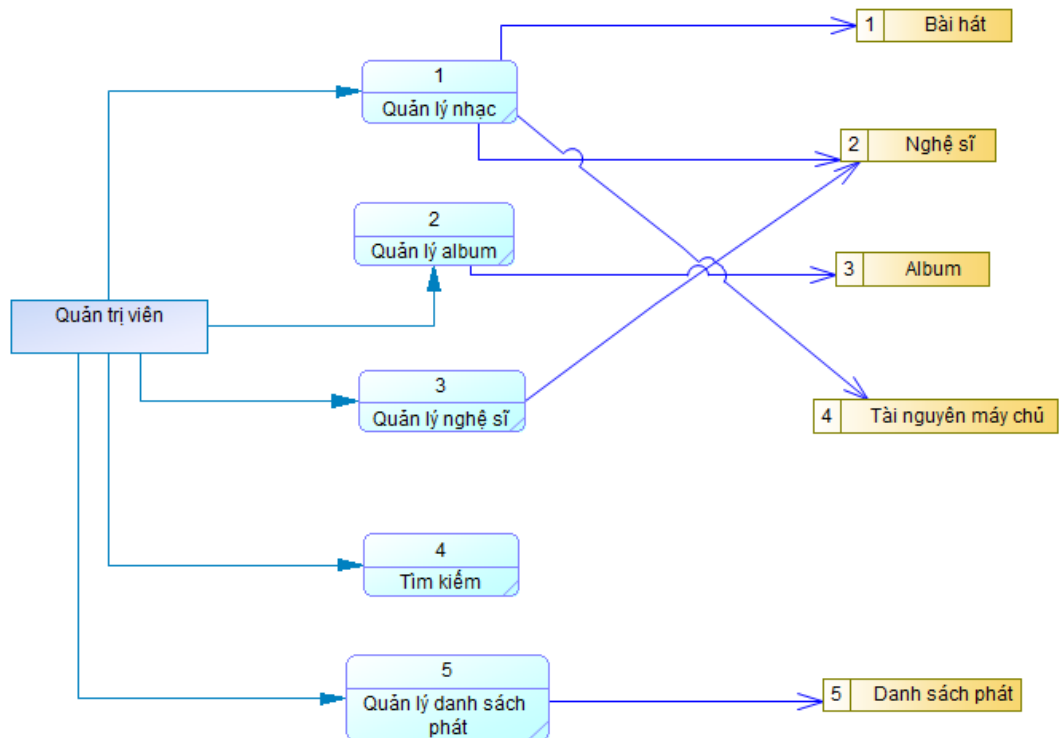
Hình 6. 9. Sơ đồ dòng dữ liệu tổng quan

Ở phân bậc người dùng, họ có thể thao tác theo các chức năng như sau (mức 1):



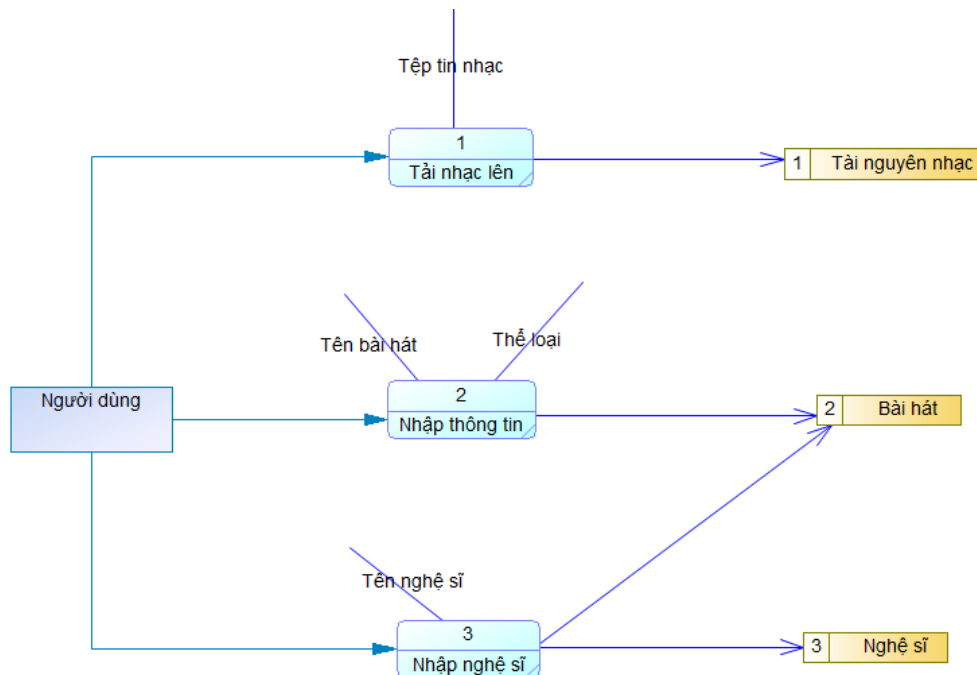
Hình 6. 10. Sơ đồ dòng dữ liệu mức người dùng

Ở phân bậc quản trị viên, họ có thể thực hiện các chức năng tiêu biểu sau (mức 1):



Hình 6.11. Sơ đồ dòng dữ liệu ở quản trị viên

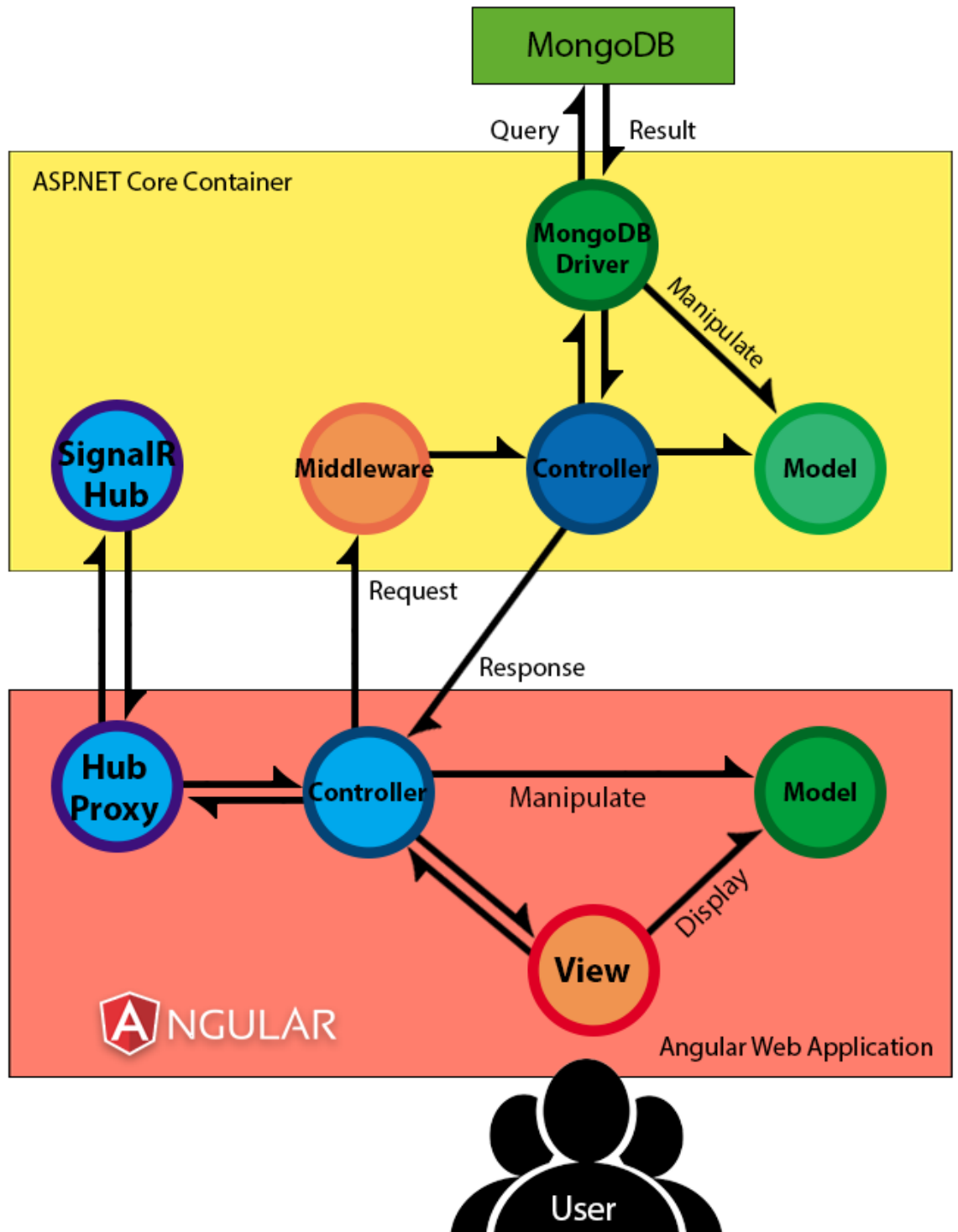
Sơ đồ chi tiết cho chức năng tải bài nhạc lên (tiêu biểu – mức chi tiết):



Hình 6.12. Dòng dữ liệu khi tải một bài hát lên (mức 2)

6.3. Thiết kế cơ bản

6.3.1. Mô hình hoạt động

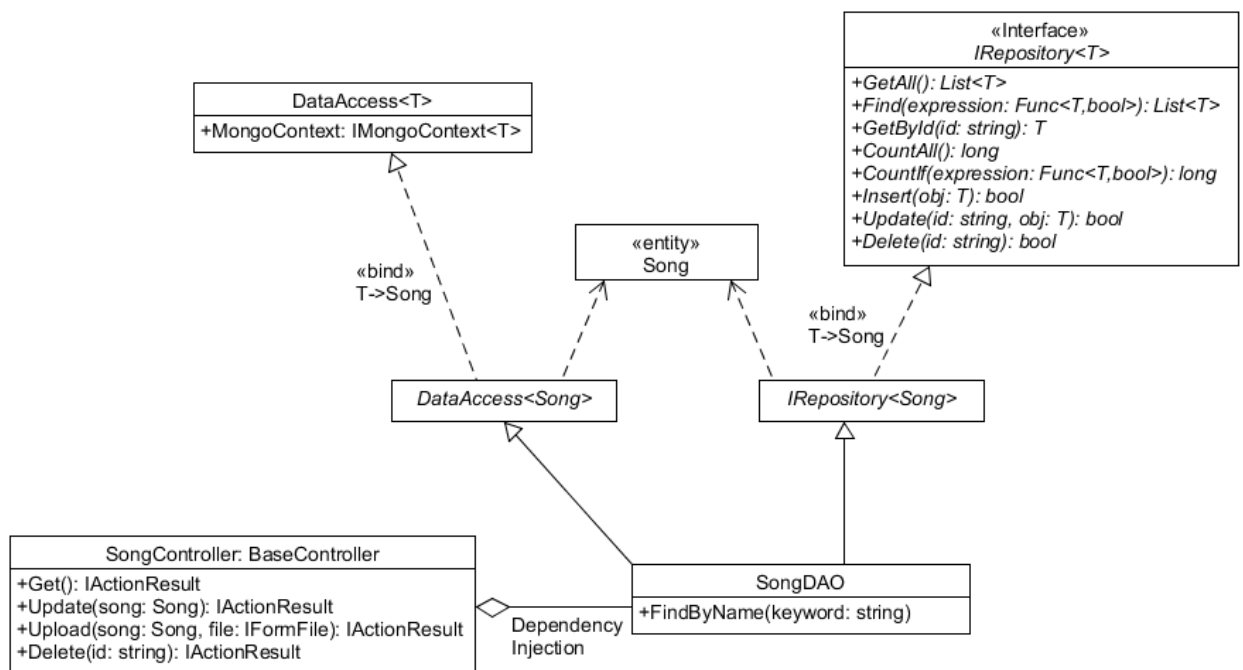


Hình 6. 13. Mô hình hoạt động của Amnhac.com

Chương trình sẽ được chia làm ba phần:

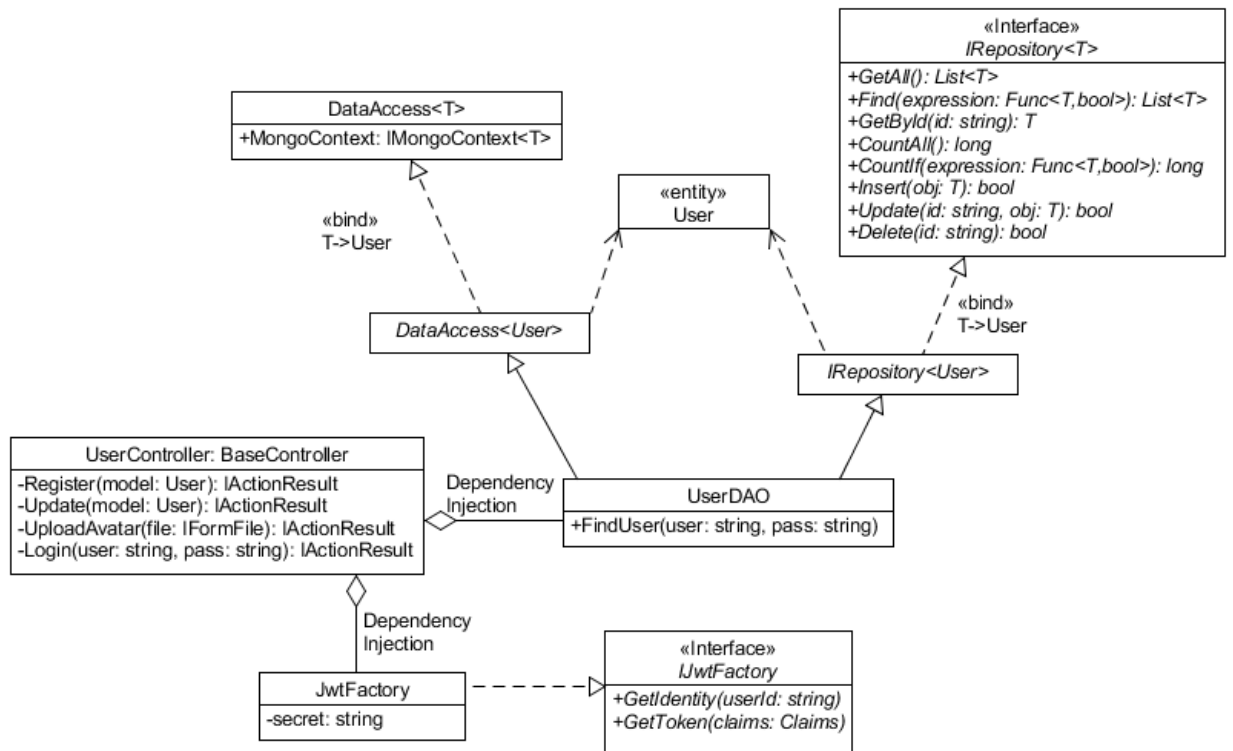
- **Giao diện người dùng:** Được viết bằng Angular 6, có nhiệm vụ giao tiếp với người dùng.
- **Máy chủ dịch vụ:** Dùng nền tảng ASP.NET Core để thiết lập dưới nền tảng dịch vụ Web, lắng nghe thông qua Middleware và trả kết quả.
- **Cơ sở dữ liệu MongoDB:** Lưu trữ cơ sở dữ liệu, thực hiện các thay đổi thông qua MongoDB.Driver

6.3.2. Thiết kế mô hình lớp



Hình 6. 14. Mô hình lớp cho phần Quản lý bài hát

Trong mô hình lớp dành cho Server, một Controller sẽ truy vấn đến MongoDB qua lớp `DataAccessObject`. Hình trên đại diện cho một Controller trong Server ASP.NET Core, các Controller khác sẽ có phần tương tự.

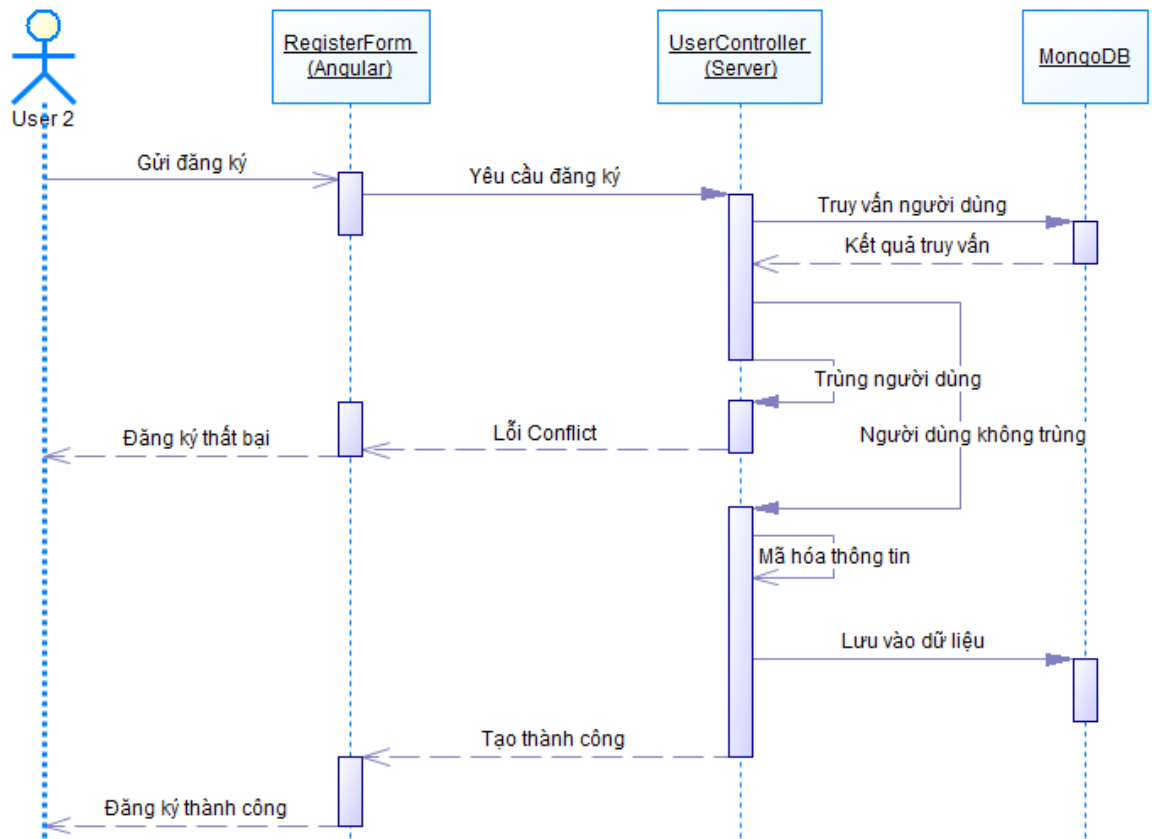


Hình 6. 15. Mô hình lớp cho phần Tài khoản người dùng

Đối với UserController ở Server, để tăng cường bảo mật, UserController sẽ nhờ JwtFactory để mã hóa thông tin người dùng thành token.

6.4. Thiết kế quy trình

6.4.1. Chức năng đăng ký

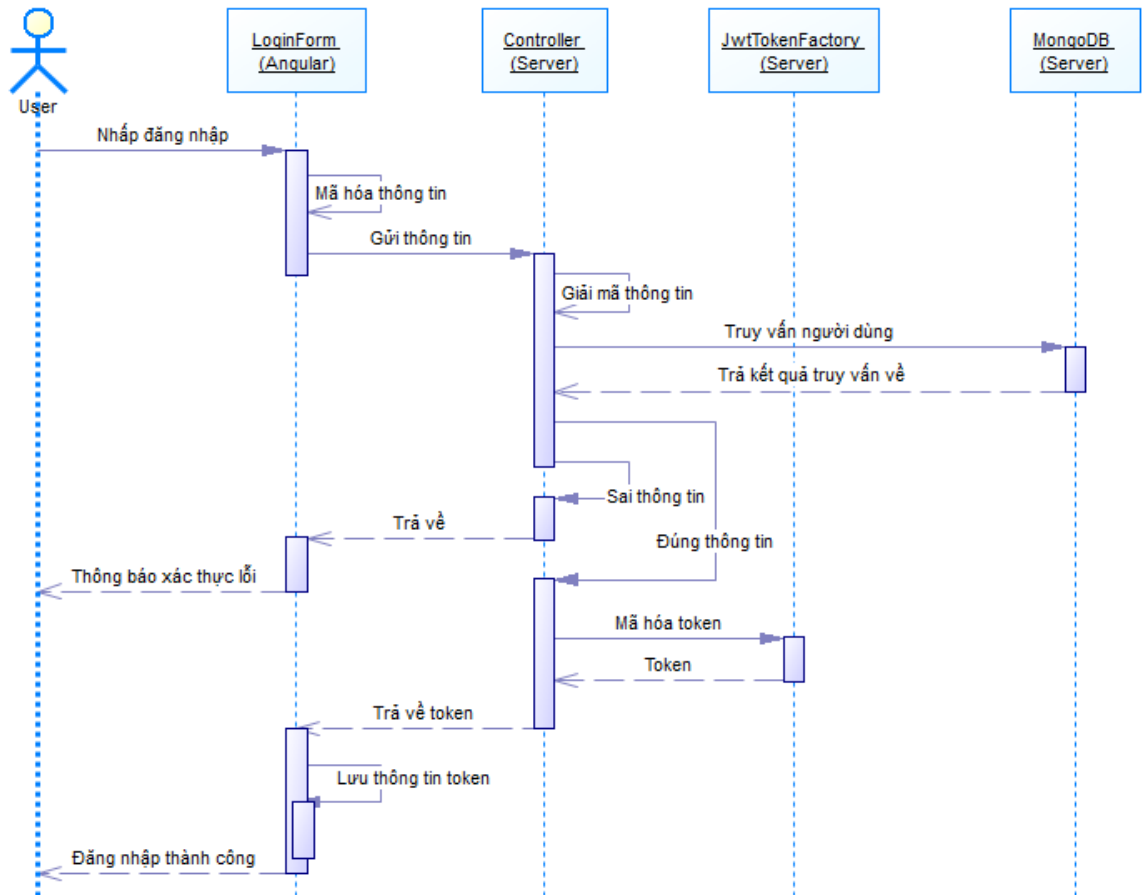


Hình 6. 16. Tuần tự quy trình khi người dùng đăng ký trên ứng dụng

Khi người dùng đăng ký từ trang web, yêu cầu sẽ được gửi lên phía Server để kiểm tra, mã hóa và lưu vào cơ sở dữ liệu.

- Nếu phát hiện trùng, server sẽ trả về lỗi Conflict cho View, View sẽ xử lý và hiển thị ra thông báo đăng ký lỗi cho người dùng.
- Nếu phát hiện không bị trùng, chương trình sẽ mã hóa thông tin và lưu vào dữ liệu, đồng thời thông báo đăng ký thành công ở phía View.

6.4.2. Chức năng đăng nhập



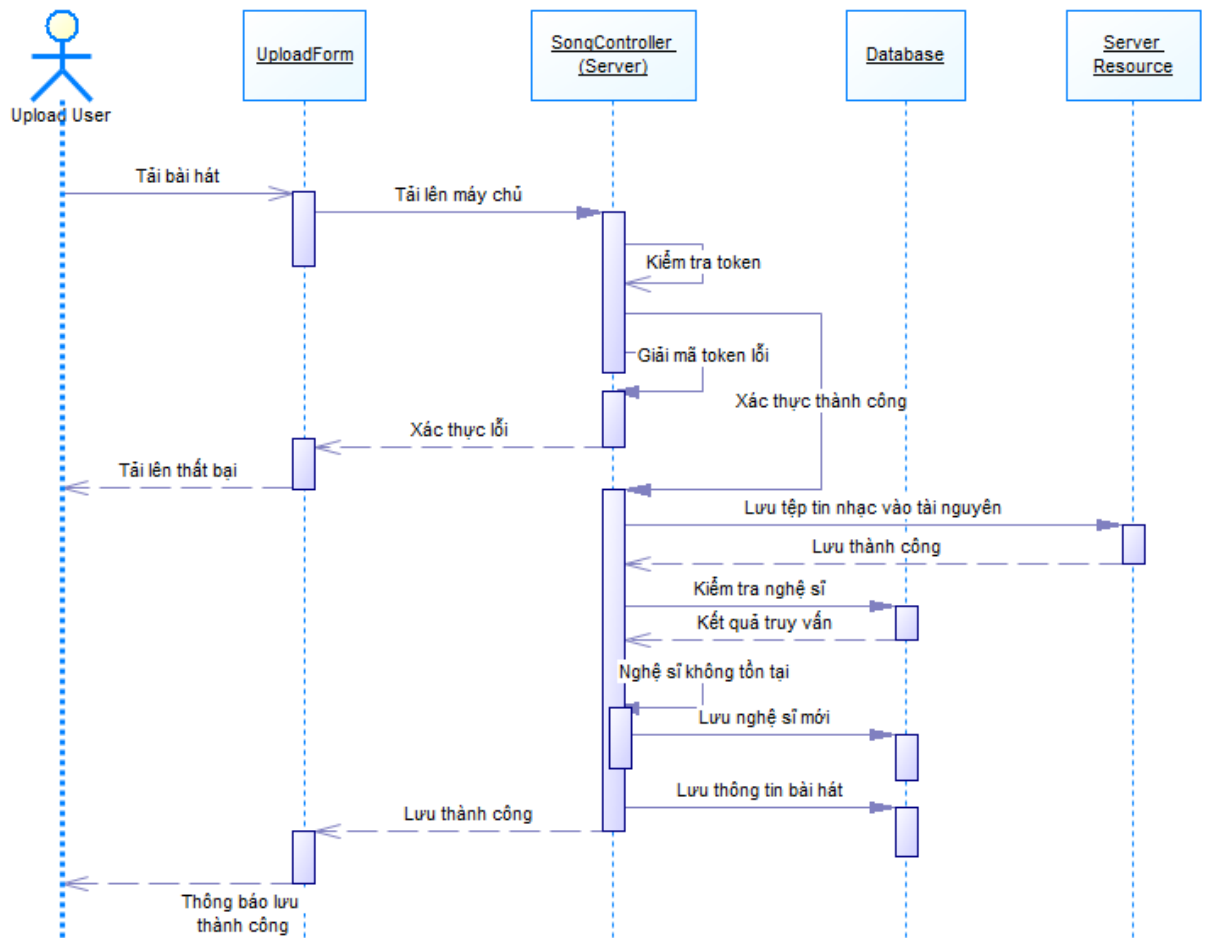
Hình 6. 17. Quy trình đăng nhập của người dùng

Khi người dùng đăng nhập, ở phía Client sẽ mã hóa thông tin cần gửi đi trước khi gửi, sau đó gửi lên Server.

Do được thiết lập cùng quy tắt mã hóa, server sẽ giải mã và truy vấn thông tin người dùng ở phía cơ sở dữ liệu.

- Nếu truy vấn sai, Server sẽ gửi lỗi xác thực về.
- Nếu truy vấn đúng, Server sẽ mã hóa các thông tin cần thiết qua JwtFactory và trả về Token cho Client.
- Client sau khi nhận được token sẽ lưu Token nhằm dùng cho lần vào kế, đồng thời thông báo người dùng đăng nhập thành công.

6.4.3. Chức năng tải nhạc

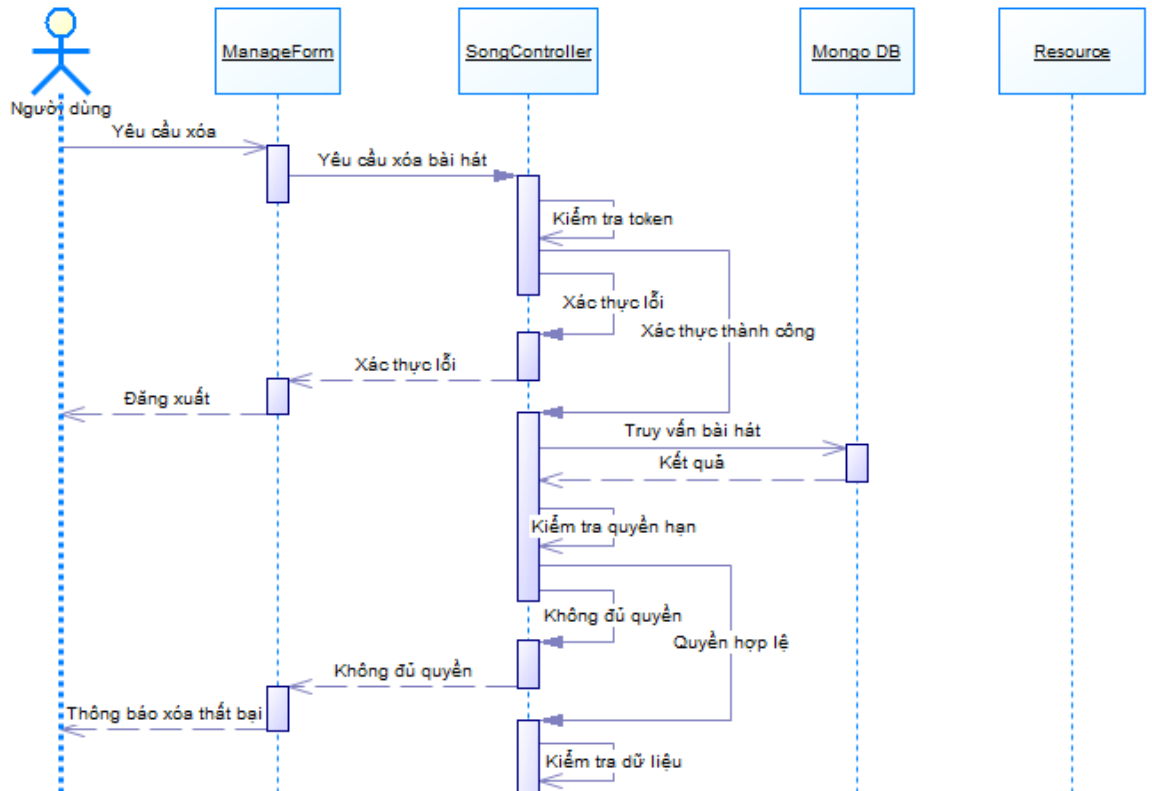


Hình 6. 18. Quy trình tải nhạc lên máy chủ

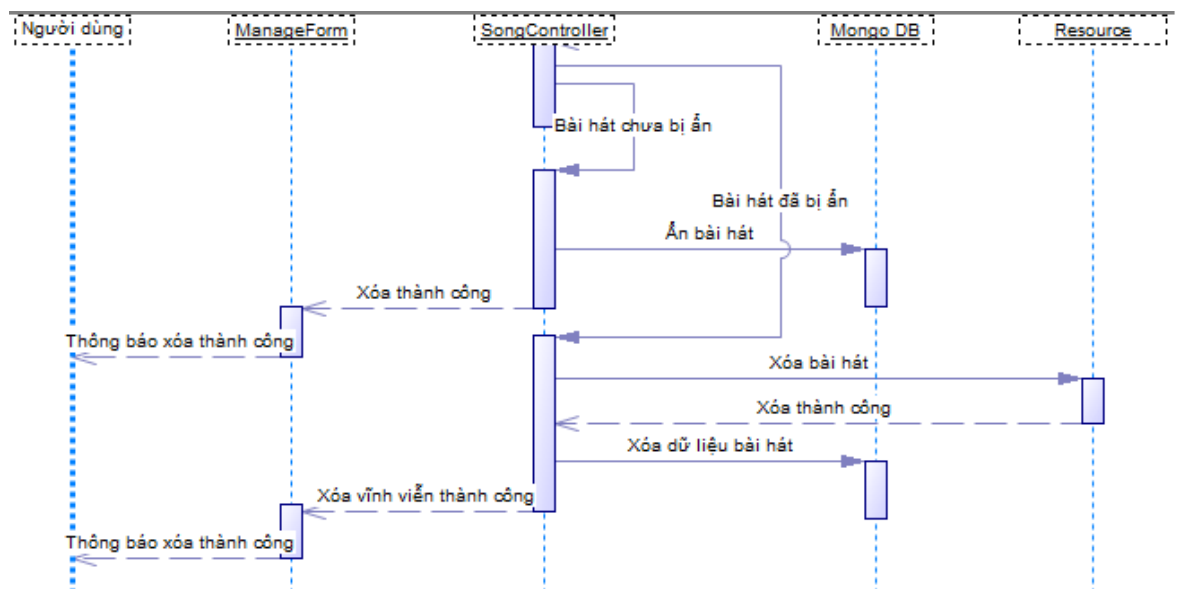
Người dùng lúc này tiến hành tải bài hát lên máy chủ, khi gửi Header sẽ kèm theo một Token.

- Máy chủ nhận token và tiến hành kiểm tra
- Nếu giải mã lỗi, tiến trình sẽ dừng và trả về lỗi xác thực.
- Nếu không có lỗi, tiến trình sẽ tiếp tục, lưu bài hát vào tài nguyên máy và kiểm tra nghệ sĩ đã nhập.
- Nếu nghệ sĩ chưa có thì sẽ được tạo mới và lưu vào cơ sở dữ liệu.
- Cuối cùng, lưu thông tin bài hát vào dữ liệu và trả về kết quả.

6.4.4. Chức năng xóa nhạc



Hình 6. 19. Quy trình xóa một bài hát trong Amnhac.com (1)



Hình 6. 20. Quy trình xóa một bài hát trong Amnhac.com (2)

Trong Amnhac.com, người dùng khi xóa một bài hát đòi hỏi phải xác thực token của họ, sau đó hệ thống sẽ xem người dùng vừa xác thực đó có đủ quyền hạn thao tác lên bài hát đó hay không như có phải người tải lên của chính bài hát đó hay là quản trị viên.

- Nếu là người quản trị viên hoặc người tải lên, bài hát sẽ được truy vấn từ cơ sở dữ liệu.
- Nếu người dùng là người tải lên, bài hát này sẽ bị thay đổi thuộc tính nhằm ẩn bài hát đi với người sử dụng, sau đó lưu thông tin vào cơ sở dữ liệu và thông báo xóa thành công.
- Nếu người dùng là quản trị viên, nếu bài hát muốn xóa chưa bị ẩn, bài hát sẽ thay đổi thành bị ẩn với người sử dụng rồi lưu thông tin vào cơ sở dữ liệu và thông báo xóa thành công. Nếu bài hát muốn xóa đã bị ẩn trước đó, bài hát sẽ bị xóa vĩnh viễn trong tài nguyên máy tính và cơ sở dữ liệu, sau đó trả về thông báo xóa thành công.

CHƯƠNG 7: TỔNG KẾT

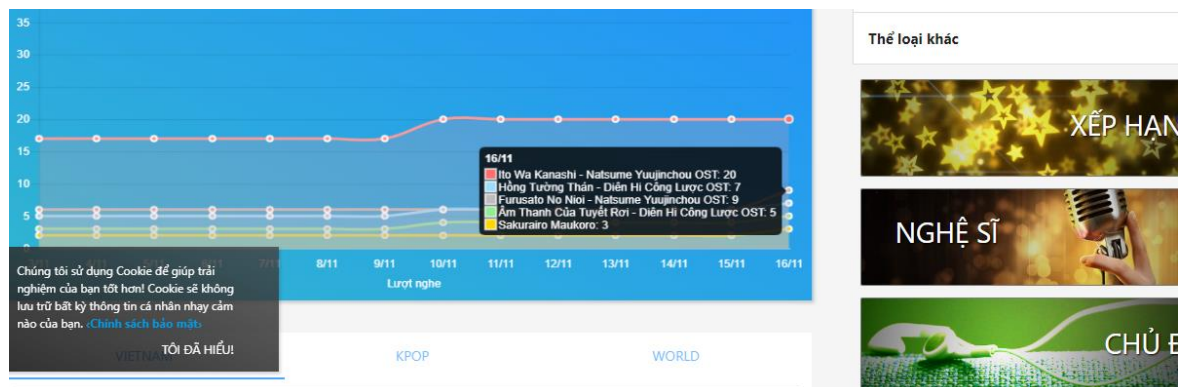
7.1. Kết quả đạt được

Amnhac.com đã có thể bắt đầu giai đoạn một, hoàn thiện các chức năng cơ bản với ý tưởng ban đầu. Amnhac.com đã trở thành một nơi để có thể chia sẻ được âm nhạc, giai điệu, và là nơi lắng nghe giai điệu của mọi người. Từ đây phát triển này, Amnhac.com có thể sẽ triển khai trên thị trường Việt Nam trong một ngày không xa.

7.2. Các chức năng nổi bật

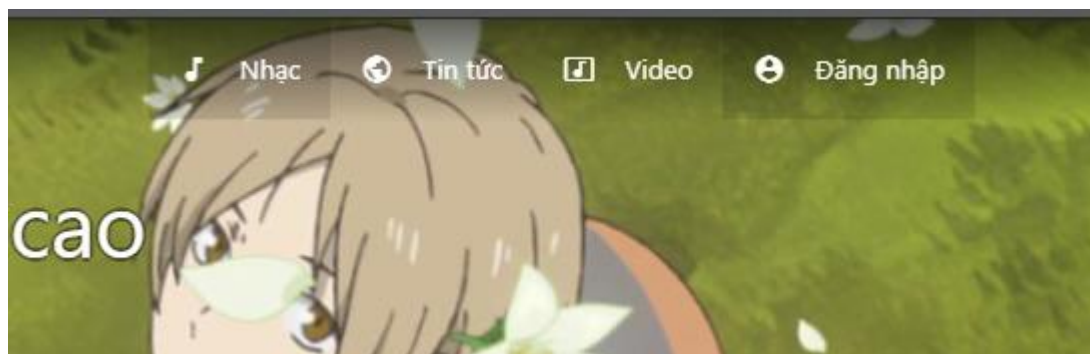
7.2.1. Trang chủ

Trang chủ có thể hiển thị thống kê, các thể loại

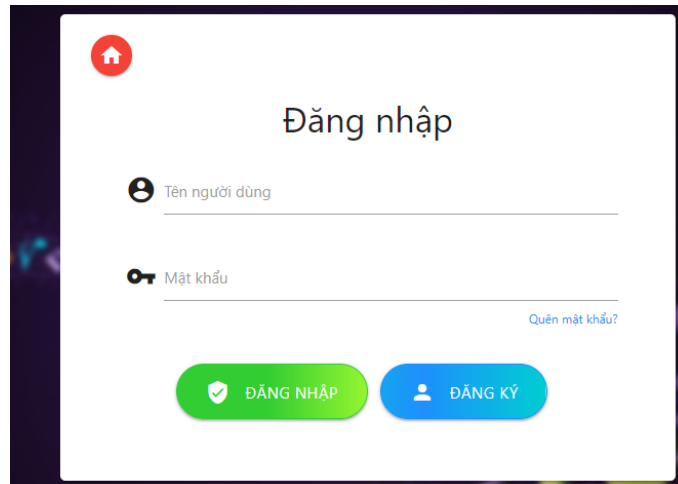


Hình 7. 1. Giao diện trang chủ Amnhac.com

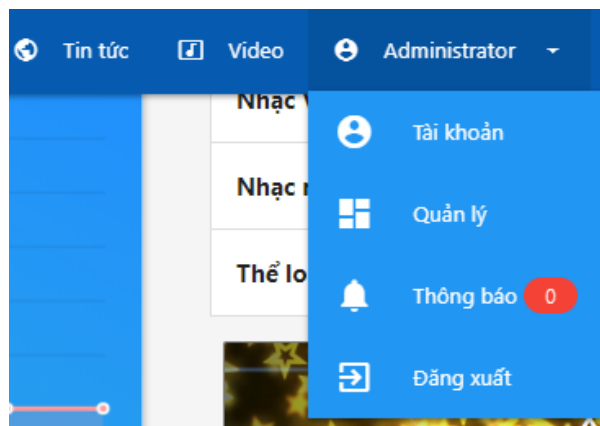
7.2.2. Trang quản lý



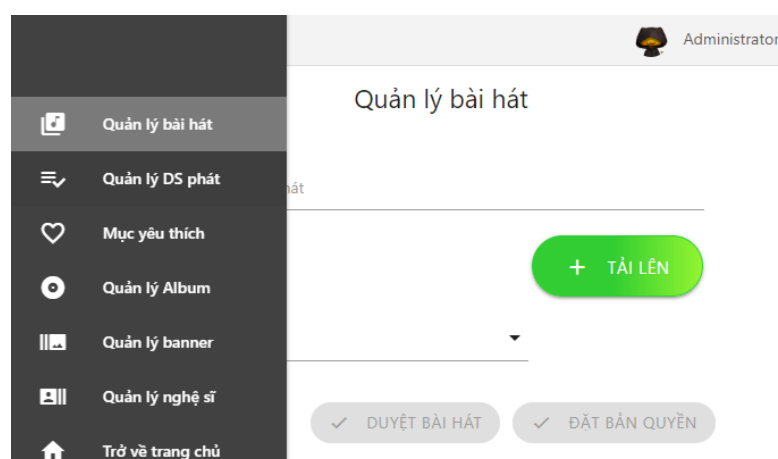
Hình 7. 2. Vào mục đăng nhập ở trên thanh tìm kiếm



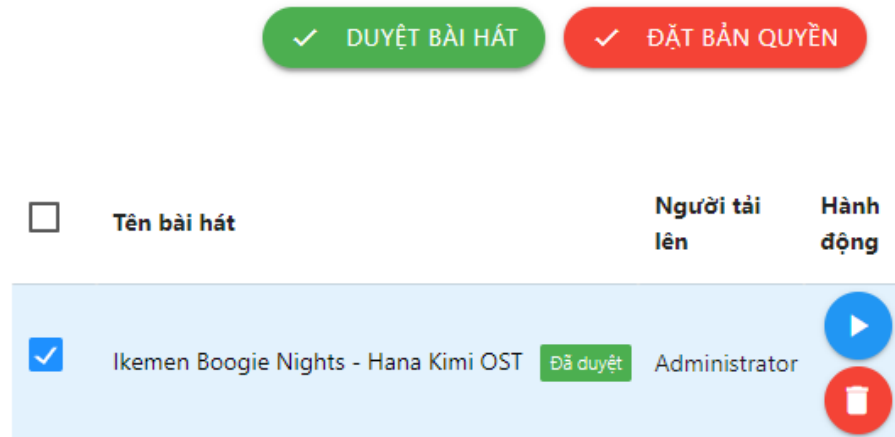
Hình 7. 3. Dùng tài khoản đăng nhập hoặc đăng ký mới



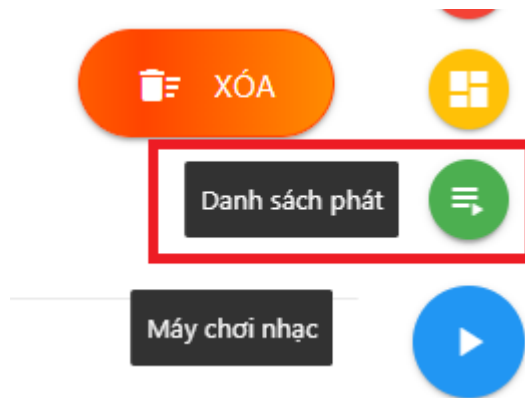
Hình 7. 4. Từ thanh tìm kiếm, vào mục quản lý



Hình 7. 5. Giao diện quản lý bài hát



Hình 7. 6. Thử tải một bài hát và chọn bài hát vừa tải để duyệt



Hình 7. 7. Bật máy chơi nhạc ở biểu tượng góc dưới màn hình và vào danh sách phát

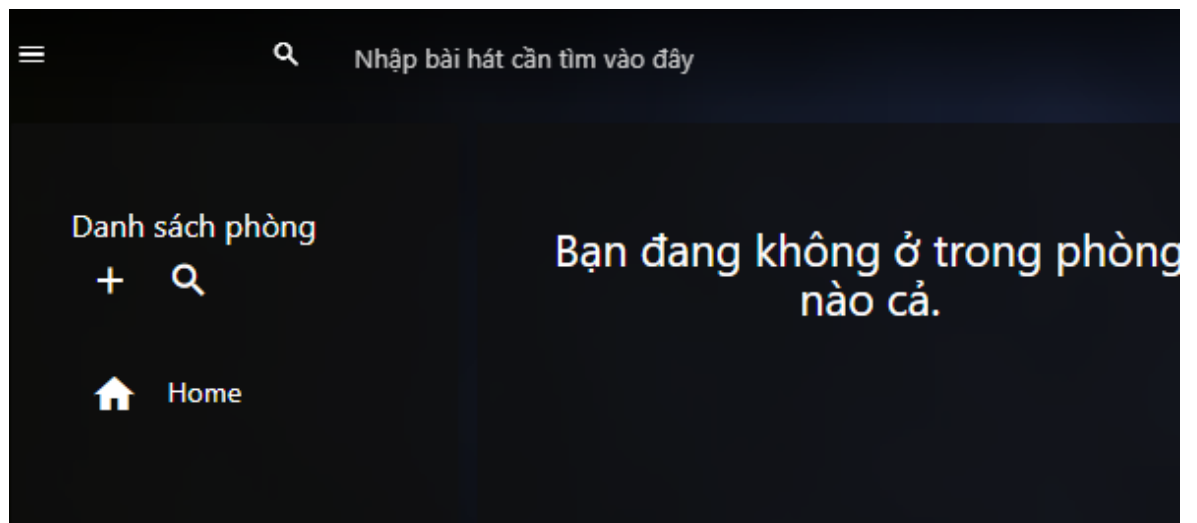


Hình 7. 8. Giao diện chơi nhạc khi phát một bài hát

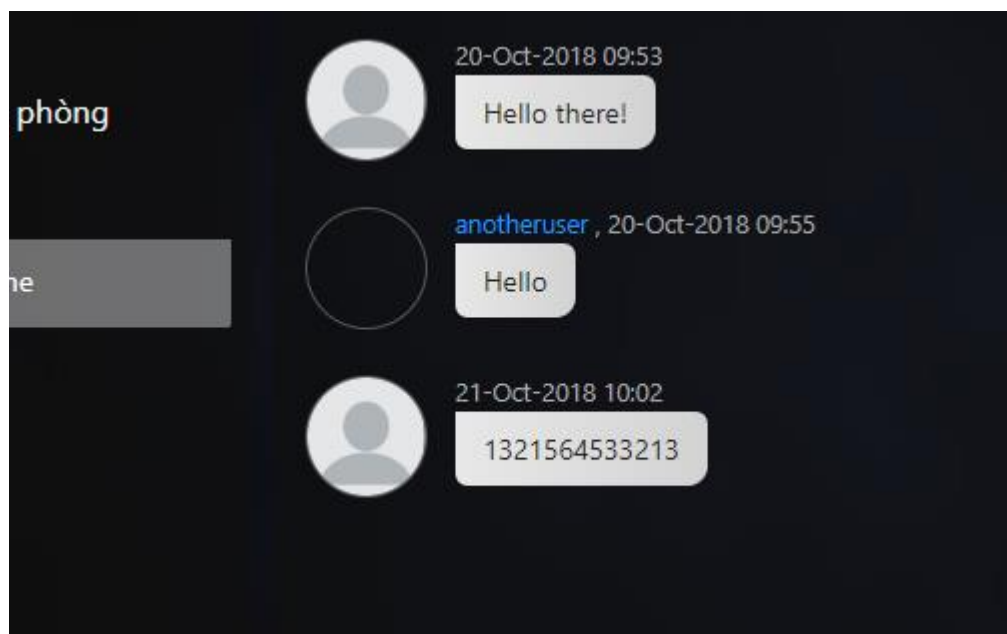
7.2.3. Phòng trò chuyện



Hình 7. 9. Về trang chủ và vào mục Trò chuyện



Hình 7. 10. Vào một phòng hoặc tạo phòng mới (khi là quản trị viên)



Hình 7. 11. Xem danh mục trò chuyện

7.3. Đánh giá

7.3.1. Những gì đạt được

- Ứng dụng đầy đủ chức năng, được áp dụng những công nghệ mới nhất và tốt nhất có thể.
- Xây dựng một trang ứng dụng web đủ phức tạp và quan tâm đến bảo mật của người dùng, đồng thời cũng tối ưu hóa sao cho trang web chạy tốt nhất.
- Giao diện bắt mắt, tự thiết kế dựa trên framework CSS Materialize.

7.3.2. Những gì cần cải thiện

- Quá tham vọng nên nhiều chức năng vẫn có thể chưa hoàn tất, thiếu.
- Có thể có lỗi trong quá trình vận hành
- Nên làm theo nhóm và chia nhỏ công việc.

7.3.3. Hướng phát triển

Amnhac.com là dự án sẽ được phát hành tại thị trường Việt Nam trong tương lai, với bước đệm đầu, Amnhac.com sẽ cố gắng hoàn thiện thêm:

- Hát với tôi, mọi người đều có thể hát trong phòng trò chuyện.
- Hoàn thiện hơn chức năng quản lý trang web như thêm, sửa thể loại, sửa quy tắc xếp hạng, thêm các hạng mục mới tùy biến mà không cần đến lập trình viên.
- Bất kỳ trang web nào cũng cần kinh phí để duy trì, Amnhac.com sẽ cung cấp thêm tính năng trả phí nhằm thu tiền bản quyền tải nhạc, là cầu nối giữa nghệ sĩ và công chúng.

PHỤ LỤC

Giới thiệu tính năng Amnhac.com và hướng dẫn thực hiện

1. Cài đặt và triển khai

1.1. Cài đặt môi trường

- Yêu cầu môi trường:
 - Hệ điều hành: Linux, Windows hoặc Mac
 - NodeJS 8.1.4 hoặc cao hơn
 - Angular CLI
 - Runtime .NET Core 2.1.4 hoặc cao hơn
 - MongoDB 4.0 hoặc cao hơn.

1.2. Cài đặt hệ quản trị cơ sở dữ liệu

Sau khi cài đặt MongoDB 4.0 vào máy tính, hãy tạo lập một người dùng cho MongoDB qua câu lệnh sau trong MongoDB Shell:

```
use admin  
  
db.createUser({  
  user: "admin",  
  pwd: "password",  
  roles: [ "readWrite", "dbAdmin" ]});
```

Trong đó, user là tên tài khoản, pwd là mật khẩu xác thực.

1.3. Triển khai máy chủ dịch vụ

Ngay trong thư mục dự án, vào thư mục chứa của máy chủ dịch vụ, tìm đến tập tin *appsettings.json*



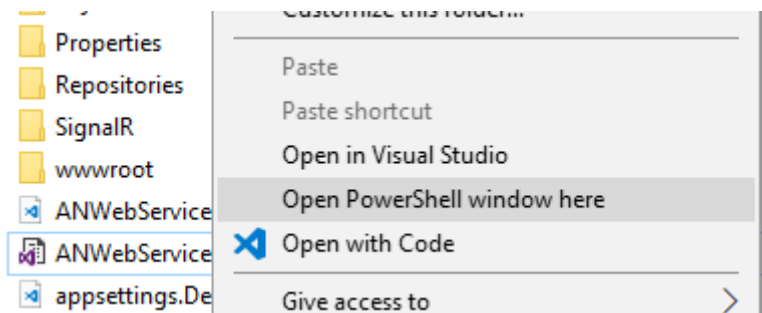
```

14 },
15
16 "MongoConnection": {
17   "ConnectionString": "mongodb://admin:password@localhost",
18   "Database": "db_amnhac"
19 },

```

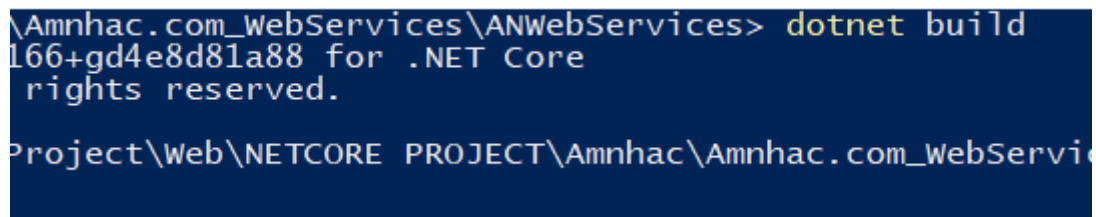
Hình 7. 12. Thiết lập appsettings.json phía ASP.NET Core server

Sau khi thay đổi, ta giữ Shift và phải chuột vào thư mục hiện hành để mở giao diện dòng lệnh Windows Powershell, hoặc có thể mở Terminal (Linux) hoặc Command Prompt và di chuyển đến thư mục dự án.



Hình 7. 13. Mở thao mục PowerShell

Sau đó gõ lệnh dotnet build để tiến hành tạo bản dựng



```

\Amnhac.com_WebServices\ANWebServices> dotnet build
166+gd4e8d81a88 for .NET Core
rights reserved.

Project\Web\NETCORE PROJECT\Amnhac\Amnhac.com_WebService

```

Hình 7. 14. Gõ lệnh dotnet build để tạo bản dựng

Sao chép thư mục wwwroot và tệp appsettings.json vào thư mục bin/Debug/netcoreapp2.1

CORE PROJECT > Amnhac > Amnhac.com_WebServices > ANWebServices > bin > Debug > netcoreapp2.1 >				
Name	Date modified	Type	Size	
wwwroot	11/11/2018 7:20 PM	File folder		
ANWebServices.deps.json	11/4/2018 6:12 PM	JSON Source File	265 KB	
ANWebServices.dll	11/11/2018 7:18 PM	Application extens	424 KB	

Hình 7. 15. Chép các tệp cần thiết vào thư mục chứa bản dựng

Sau đó chuyển địa chỉ dòng lệnh vào thư mục này và gõ *dotnet* *ANWebServices.dll* để tiến hành host một máy chủ.

```
ANWebServices\bin\Debug\netcoreapp2.1> dotnet ANWebServices.dll
LogHandler has started.
Hosting environment: Production
Content root path: D:\Project\Web\NETCORE PROJECT\Amnhac\Amnhac.com_WebServices\ANWebServices\bin\Debug\netcoreapp2.1
Now listening on: http://localhost:5000
Now listening on: https://localhost:5001
Application started. Press Ctrl+C to shut down.
{"ServerId": "5bd3d60c-0206423578712628", "Priority": 0, "ValidFrom": "2018-11-11T07:20:00Z", "ValidTo": "2018-11-11T07:20:00Z"}
```

Hình 7. 16. Chạy thử máy chủ

1.4. Triển khai máy chủ giao tiếp

Là giao diện chính, để triển khai máy chủ giao tiếp, ta vào thư mục dự án chứa Web UI và mở Command Prompt/Windows Powershell/Terminal tại đây.

Gõ *npm install* để cài đặt các thành phần còn thiếu

```
CT\Amnhac\Amnhac.com_webUI> npm install
all:anweb-app: info anweb-app@0.0.0~preinstall: anweb-
```

Hình 7. 17. Cài đặt các thành phần còn thiếu cho Angular Web Application

Sau đó, gõ *ng build --prod*

```
> ng build --prod
```

Hình 7. 18. Tạo bản dựng cho việc triển khai

Copy toàn bộ tệp tin trong thư mục *dist/* và dán chúng vào thư mục trên máy chủ Web, nơi cần phục vụ cho người dùng.

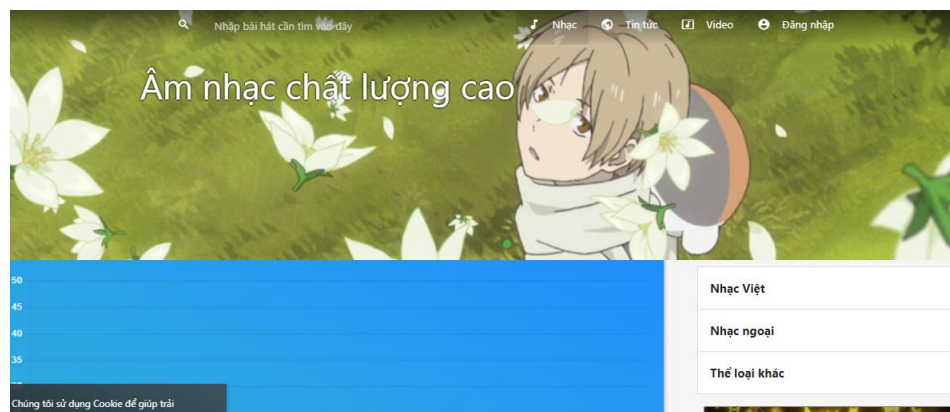
Tùy môi trường host và sẽ có thiết lập khác nhau, lúc này, cài đặt sao cho chuyển hướng các tệp không tìm thấy về trang *index.html*.

Nếu người dùng chỉ muốn chạy thử không cần tạo bản dựng, chỉ cần gõ *ng serve --o* là có thể xem được giao diện giao tiếp tại cổng 4200.

1.5. Triển khai ứng dụng

Lúc này, người quản trị đã có thể sử dụng Amnhac.com, tuy nhiên, họ vẫn chưa hoàn toàn có thể quản trị trang web do không được cấp tài khoản quản trị.

Để tiến hành khởi động trang, vào trang web:



Hình 7. 19. Giao diện trang chủ sau khi triển khai

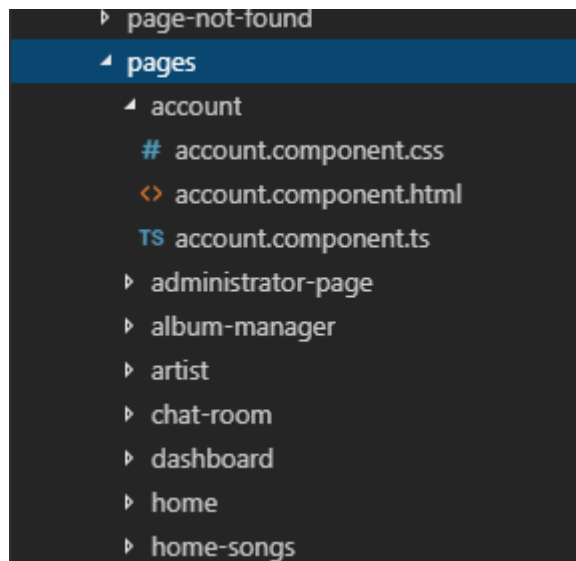
Sau đó điều hướng đến */system/initializer*, nếu trang Web chưa được thiết lập đúng, lúc này hệ thống sẽ tự động thiết lập mọi thứ cho người dùng, tạo sẵn cơ sở dữ liệu mẫu và cấp cho họ một tài khoản quản trị viên.

2. Lập trình chương trình

2.1. Ứng dụng web đơn trang

Nhờ Angular, các xử lý trên trang rất mượt mà, chỉ tải những thành phần cần thiết.

Tạo các trang qua câu lệnh *ng g c <tên trang>*



Hình 1

Sau đó tạo nội dung cho các trang.

Áp dụng Router trong Angular để làm một ứng dụng đơn trang như sau:

```
const routes:Routes=[
  { path:'', component: HomeComponent, children:[
    { path:'', redirectTo:'songs', pathMatch:'full'},
    { path: 'songs', loadChildren:()=>> HomeSongsModule }
  ] },
  { path:'management', loadChildren: ()=>> AdministratorPage
  { path:'artist', loadChildren: ()=>>ArtistModule },
  { path:'login', component: LoginComponent },
  { path:'register',loadChildren:()=>>MyRegisterModule},
  { path:'system',loadChildren:()=>>AdministratorBoardModule
  { path:'dashboard', component: DashboardComponent, canActivate:
  { path:'chat', loadChildren: ()=>> ChatRoomModule, canActivate:
  { path:'account', component: AccountComponent, canActivate:
  { path:'**', component: PageNotFoundComponent }
];
@NgModule({
  imports: [
    RouterModule.forRoot(routes)
  ],
  exports:[ RouterModule ],
  declarations: []
})
export class AppRouterModule { }
```

Hình 2

Tệp Router này là một NgModule, nó sẽ nhận RouterModule như là thành phần chính, nhận các thiết lập đường dẫn đã tạo trong **.forRoot**.

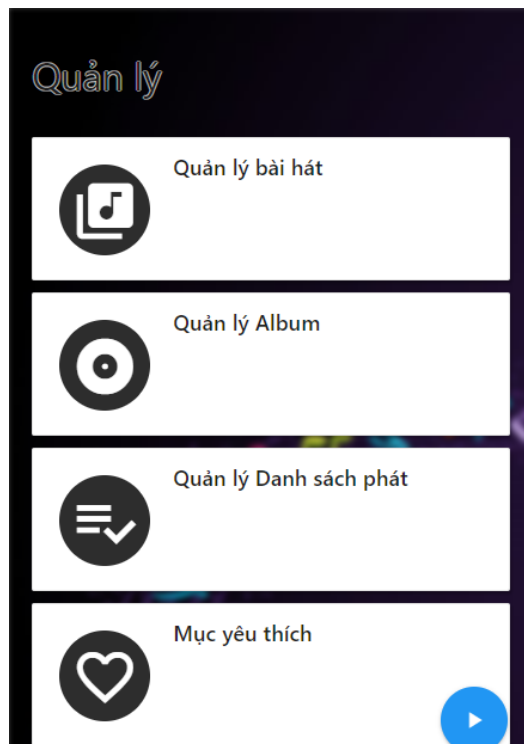
Đặt thuộc tính `routerLink` để dùng `RouterModule` điều hướng thay vì dùng `href` của HTML:

```
<li><a routerLink="/"><i class="material-icons left">home</i>{{ lang.ui.ho  
<li><a routerLink="/news"><i class="material-icons left">public</i>{{ lang  
<li><a routerLink="/video"><i class="material-icons left">music_video</i>{  
<li><a routerLink="/account"><i class="material-icons left">account_circle
```

Hình 3

2.2. Dịch vụ web hướng đa nền tảng

Dưới dạng dịch vụ Web, máy chủ web ASP.NET Core có thể lắng nghe không chỉ các yêu cầu từ Ajax Website mà còn các yêu cầu thuộc giao thức HTTP.



Hình 4

2.3. Tối ưu bằng Caching

ASP.NET Core cung cấp phản hồi caching, giúp lưu các thông tin trả về dạng cache ở phía người dùng, như vậy, ở phía người dùng sẽ yêu cầu đến cache trên máy của họ thay vì yêu cầu lại máy chủ.

```
// GET: api/<controller>
[HttpGet]
[AllowAnonymous]
[ResponseCache(Duration = 1500, Location = ResponseCacheLocation.Any)]
public async Task<ActionResult> Get([FromQuery] int page, [FromQuery] int size, [Fr
{
    long count = 0;
    string alphabet = null;
```

Hình 5

Khi người dùng yêu cầu một kết quả đã xử lý trước đó lần nữa:

```
Request URL: http://localhost:6715/ap
i/song
Request Method: GET
Status Code: 200 OK (from disk cac
he)
```

Hình 6

2.4. Phát một phần tập tin qua luồng

Với ASP.NET Core, dịch vụ hỗ trợ việc phát tập tin qua luồng, tức là khi yêu cầu một bài nhạc, máy dịch vụ chỉ trả về một luồng, người dùng yêu cầu đoạn nào, thì máy chủ sẽ chỉ xử lý từ đoạn đó đến một phần nhỏ của tập tin đó rồi gửi về cho người dùng. Như vậy, tốc độ phát nhạc sẽ được cải thiện đáng kể.

```
Song s = await _song.GetById((id));
if (s == null) return NotFound();
string basePath = _env.WebRootPath + "/_media/";
//=====
FileStream fs;
try
{
    fs = new FileStream(basePath + s.Paths[0].Path,
        FileMode.Open, FileAccess.Read,
        FileShare.Read, 4096, FileOptions.Asynchronous);
}
catch (Exception e)
{
    return BadRequest(e.Message);
}
long fileSize = fs.Length;
```

Hình 7

```
Response.ContentType = "audio/" + s.Paths[0].Extension;
Response.Headers.Add("Content-Accept", Response.ContentType);
Response.Headers.Add("Content-Length", desSize.ToString());
Response.Headers.Add("Content-Range", string.Format("bytes {0}-{1}/{2}", startbyte, endbyte, fSize));
Response.Headers.Add("Accept-Ranges", "bytes");
Response.Headers.Remove("Cache-Control");
//Data
fs.Seek(startbyte, SeekOrigin.Begin);
if (startbyte == 0 && s.Approved > 0)
{
    s.View++;
    song.Update((s.Id), s);
}
return new FileStreamResult(fs, Response.ContentType);
```

Hình 8

Trong phần trả về Header trên có các thuộc tính sau:

- **Content-Accept:** Trả về kiểu tập tin
- **Content-Length:** Độ dài của tập tin (đọc từ metadata)
- **Content-Range:** Trả về vị trí byte yêu cầu (startbyte), số byte đã đọc được (endbyte), và tổng số byte của tập tin (fSize)
- **Accept-Ranges:** Content được trả về theo dạng bytes.
- **Remove("Cache-Control"):** Do trả về luồng, ta yêu cầu máy người dùng không lưu cache.

2.5. Bảo mật với Xác thực JWT

Jwt là cơ chế xác thực bằng Token. Khi người dùng xác thực đúng, thư viện sẽ lưu các thông tin cần thiết.

Để mở xác thực Jwt trong ASP.NET Core, chúng ta phải trả qua khá nhiều bước phức tạp.

Mở appsetting.json trong phần WebServices, thêm các thiết lập về khóa bí mật cho JWT:

```
{
  "Crypto": {
    "secret_key": "ZzX38kCHPYV12YriHlUCeGqkY7qkEVF08nUToZ471Gcb8tb9FusYBvubFWvCmMC",
    "sha_key": "jb4n7huyPibrEcRhjTDkgKtQgZRwUXZv.r92b|A#xS+H9^1kW0-WL"
  },
  "Jwt": {
    "secret_key": "ZzX38kCHPYV12YriHlUCeGqkY7qkEVF08nUToZ471Gcb8tb9FusYBvubFWvCmMC",
    "sha_key": "jb4n7huyPibrEcRhjTDkgKtQgZRwUXZv.r92b|A#xS+H9^1kW0-WL"
  }
}
```

Hình 9

Mở tệp Startup.cs và thêm các câu lệnh sau vào hành vi thiết lập dịch vụ:

```
string SecretKey= Configuration.GetSection("Crypto:secret_key").Value;  
Startup.secret_key = Configuration.GetSection("Crypto:sha_key").Value; ;  
SymmetricSecurityKey _signingKey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(SecretKey));
```

Hình 10

```
services.Configure<JwtOptions>(options =>  
{  
    options.Issuer = jwtAppSettingOptions[nameof(JwtOptions.Issuer)];  
    options.Audience = jwtAppSettingOptions[nameof(JwtOptions.Audience)];  
    options.SigningCredentials = new SigningCredentials(_signingKey, SecurityAlgorithms.HmacSha256);  
});
```

Hình 11

```
// Add Token Validator  
var tokenValidationParameters = new TokenValidationParameters  
{  
    ValidateIssuer = true,  
    ValidIssuer = jwtAppSettingOptions[nameof(JwtOptions.Issuer)],  
  
    ValidateAudience = true,  
    ValidAudience = jwtAppSettingOptions[nameof(JwtOptions.Audience)],  
  
    ValidateIssuerSigningKey = true,  
    IssuerSigningKey = _signingKey,  
  
    RequireExpirationTime = true,  
    ValidateLifetime = true  
};  
services.AddAuthentication(options => {  
    options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;  
})  
    .AddJwtBearer(o =>  
    {  
        o.TokenValidationParameters = tokenValidationParameters;  
        o.IncludeErrorDetails = true;  
    });
```

Hình 12. Khởi tạo thiết lập cho Jwt Bearer Factory

```
private JwtOptions _jwtOptions;  
public static string AuthorizationHeader = "Amnhac";  
  
public JwtFactory(IOption<JwtOptions> jwtOptions)  
{  
    _jwtOptions = jwtOptions.Value;  
    ThrowIfInvalidOptions(_jwtOptions);  
}
```

Hình 13. Inject các option vào Factory

```
var claims = new[]
{
    new Claim(JwtRegisteredClaimNames.Sub, userName),
    new Claim(JwtRegisteredClaimNames.Jti, await _jwtOptions.JtiGenerator()),
    new Claim(JwtRegisteredClaimNames.Iat,
        ToUnixEpochDate(_jwtOptions.IssuedAt).ToString(), ClaimValueTypes.Integer64),
    new Claim(ClaimTypes.Role, id.FindFirst("rol").Value),
    id.FindFirst("id")
};

// Create the JWT security token and encode it.
var jwt = new JwtSecurityToken(
    issuer: _jwtOptions.Issuer,
    audience: _jwtOptions.Audience,
    claims: claims,
    notBefore: _jwtOptions.NotBefore,
    expires: _jwtOptions.Expiration.AddMinutes(addMinute).AddHours(addHour),
    signingCredentials: _jwtOptions.SigningCredentials);

var encodedJwt = new JwtSecurityTokenHandler().WriteToken(jwt);

return encodedJwt;
```

Hình 14. Tạo các Claim và nhờ Jwt mã hóa chúng

```
\\"access_token\\": \\"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJyb290Ii
```

Hình 15. Token được mã hóa gửi về Client

```
[HttpPost("save")]
[Authorize]
public async Task<IActionResult> Create(IFormCollection collection)
{
    Console.WriteLine(collection["model"].ToString());
    var model = JsonConvert.DeserializeObject<Playlist>(collection["m
```

Hình 16. Đặt thẻ chú thích Authorize để yêu cầu gửi kèm Token mỗi yêu cầu

▼ Request Headers view source

Accept: application/json, text/plain, */*

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9

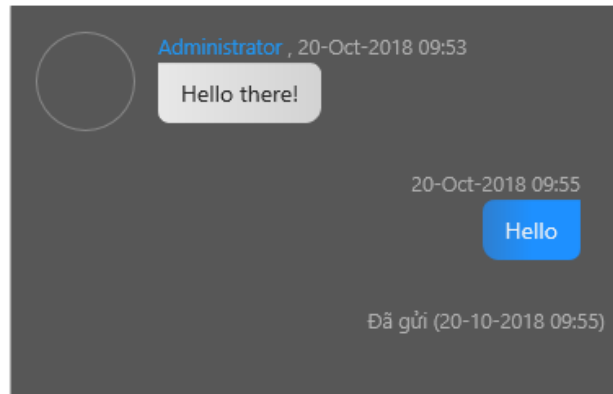
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJybnZ9O
IiwianRpIjoiaWY3MDRlYWQzMjZCOTYTA0LTg3YWQtMjUyYmMxZWVhbiF0IjoxNjEwODg3LjE3ODQ3MTk3MTB3c0YtYmVjdEkiLCJleHAiOiJlNDAwMDY0ODQsImZlc3VlciIsImF1ZCI6IkpXVCJ9.eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJybnZ9O

Connection: keep-alive

Hình 17. Token được gửi kèm trong Header để xác thực

2.6. Giao tiếp thời gian thực an toàn

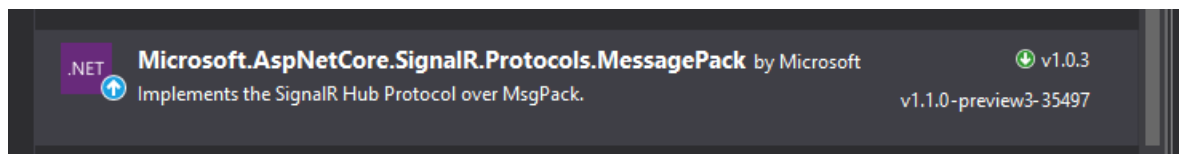
Với SignalR, quá dễ dàng để tạo lập một phòng trò chuyện nho nhỏ:



Hình 18

Song, để tăng cường tính bảo mật, SignalR hỗ trợ một giao thức Packaged Message, giúp mã hóa các giao tiếp giữa máy chủ và máy con.

Để sử dụng giao thức này, chúng ta cần phải bật lên như sau:



Hình 19

Sau đó, ta vào Startup.cs để sử dụng giao thức này cho SignalR trong ConfigureServices:

```
services.AddSignalR().AddMessagePackProtocol();
```

Hình 20

Ở phía front-end hay SignalR Client, ta cũng cần cài đặt giao thức này.

Cài đặt qua npm bằng câu lệnh:

```
npm install @aspnet/signalr-protocol-msgpack
```


Sau đó thêm dòng sau vào Builder của SignalR Core:

```
this.hub=new HubConnectionBuilder()  
.withUrl(this.url+"/signalr")  
.withHubProtocol(new MessagePackHubProtocol())  
.build();
```

Hình 21

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Bách Khoa Toàn Thư Mở Việt Nam, “NoSQL”:
<https://vi.wikipedia.org/wiki/NoSQL>
2. Phân tích thiết kế hệ thống thông tin – Thầy Nguyễn Đức Khoa – Trường Đại Học Cần Thơ
3. Giáo trình SQL – Đại học Khoa học Huế - Thầy Trần Nguyên Phong – 2004

Tiếng Anh

4. Werner Vogels - allthingsdistributed.com:
<https://www.allthingsdistributed.com/2012/01/amazon-dynamodb.html>
5. W3school.com
6. MongoDB.com: <https://mongodb.com>
7. Angular.io – Google: <https://angular.io/>
 - a. “Tutorial - Modules”: <https://angular.io/guide/architecture-modules>
 - b. “Tutorial – Services”: <https://angular.io/tutorial/toh-pt4>
 - c. “Tutorial – Routing”: <https://angular.io/tutorial/toh-pt5>
 - d. “Deployment”: <https://angular.io/guide/deployment>
8. Tanya Gray – Medium Blog, “Angular Route Transition Animation”:
<https://medium.com/@tanya/angular4-animated-route-transitions-b5b9667cd67c>
9. Bách khoa toàn thư mở:
 - a. “Inversion of Control”:
https://en.wikipedia.org/wiki/Inversion_of_control
 - b. “Dependency Injection”:
https://en.wikipedia.org/wiki/Dependency_injection
 - c. “Định nghĩa JSON Web Token”:
https://en.wikipedia.org/wiki/JSON_Web_Token

10. Petru Faurescu on Quality App Design – “Using MongoDB with .NET Core Web API”: <https://www.qappdesign.com/using-mongodb-with-net-core-webapi/>
11. MindingData on Stackoverflow, Answer for “How to enable CORS in ASP.NET Core App”: <https://stackoverflow.com/questions/44379560/how-to-enable-cors-in-asp-net-core-webapi>
12. Steve Smith, Scott Addie, Luke Latham – Microsoft Docs, “Dependency Injection in ASP.NET Core”: <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-2.1#service-lifetimes-and-registration-options>
13. Tom Dykstra, Evan Ogas, Luke Latham, Scott Addie, Benjamin N. Summerton, Rachel Appel & Nate Barbettini – Microsoft Docs, “Get Started with ASP.NET Core SignalR”: <https://docs.microsoft.com/en-us/aspnet/core/tutorials/signalr?view=aspnetcore-2.1&tabs=visual-studio>
14. Mark Macneil on FullStackMark, “JWT Authentication with ASP.NET Core and Angular 5”: <https://fullstackmark.com/post/13/jwt-authentication-with-aspnet-core-2-web-api-angular-5-net-core-identity-and-facebook-login>
15. Martin Dawson on Stackoverflow, Answer for “MVC audio controls playing song from bytes”: <https://stackoverflow.com/questions/31246314/mvc-audio-controls-playing-song-from-bytes>
16. Brennan Controy – Docs.Microsoft.Com – “Use MessagePack Protocol in SignalR for ASP.NET Core”: <https://docs.microsoft.com/en-us/aspnet/core/signalr/messagepackhubprotocol?view=aspnetcore-2.1>
17. Medium.com

DẪN NGUỒN

1. <https://vietnambiz.vn/kinh-doanh-nhac-so-nho-du-lieu-lon-48352.html>
2. <http://www.misa.com.vn/tin-tuc/chi-tiet/newsid/53932/Cac-doanh-nghiep-dang-su-dung-am-nhac-vao-Marketing-nhu-the-nao>
3. <https://tinhte.vn/threads/vng-phan-hoi-ve-viec-hon-160-trieu-thong-tin-tai-khoan-bi-lo.2789251/>
4. <https://codeburst.io/angularjs-4-101-6675076784aa>