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 sẽ 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 cám 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 cám 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.

Cám 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

ĐÁNH GIÁ CỦA GIÁO VIÊN HƯỚNG DẪN

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

**Mục lục**

[ĐÁNH GIÁ CỦA GIÁO VIÊN HƯỚNG DẪN](#_Toc532210217)

[DANH MỤC KÝ HIỆU, CHỮ VIẾT TẮT](#_Toc532210218)

[DANH MỤC HÌNH ẢNH](#_Toc532210219)

[DANH MỤC BẢNG](#_Toc532210220)

[TÓM TẮT](#_Toc532210221)

[ABSTRACTION](#_Toc532210222)

[CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN 1](#_Toc532210223)

[1.1. Tổng quan 1](#_Toc532210224)

[1.1.1. Đặt vấn đề 1](#_Toc532210225)

[1.1.2. Tình hình hiện nay 1](#_Toc532210226)

[1.2. Ý tưởng 3](#_Toc532210227)

[1.3. Mục tiêu đề tài 4](#_Toc532210228)

[1.4. Công cụ 5](#_Toc532210229)

[1.5. Hướng giải quyết 5](#_Toc532210230)

[1.6. Bố cục 8](#_Toc532210231)

[Phần 1: Cơ sở lý thuyết 9](#_Toc532210232)

[CHƯƠNG 2: MONGODB 9](#_Toc532210233)

[2.1. NoSQL 9](#_Toc532210234)

[2.1.1. Giới thiệu 9](#_Toc532210235)

[2.1.2. Ưu điểm của NoSQL 9](#_Toc532210236)

[2.1.3. Cách thức lưu trữ 9](#_Toc532210237)

[2.1.4. Tính năng của NoSQL 10](#_Toc532210238)

[2.2. MongoDB 10](#_Toc532210239)

[2.2.1. Tại sao nên dùng MongoDB? 10](#_Toc532210240)

[2.2.2. Mục đích sử dụng MongoDB 10](#_Toc532210241)

[2.2.3. Định nghĩa 11](#_Toc532210242)

[2.2.4. Truy vấn 11](#_Toc532210243)

[2.2.5. MongoDb Compass 13](#_Toc532210244)

[CHƯƠNG 3: ANGULAR 16](#_Toc532210245)

[3.1. Giới thiệu về Angular 16](#_Toc532210246)

[3.2. Mục đích sử dụng 16](#_Toc532210247)

[3.3. Giới thiệu về NodeJS 17](#_Toc532210248)

[3.4. Lập trình 18](#_Toc532210249)

[3.4.1. Lập trình trên môi trường mới 18](#_Toc532210250)

[3.4.2. Lập trình trên môi trường có sẵn 19](#_Toc532210251)

[3.4.3. Công cụ soạn thảo 19](#_Toc532210252)

[3.4.4. Các cú pháp dòng lệnh 20](#_Toc532210253)

[3.4.5. Các thành phần chính 21](#_Toc532210254)

[3.4.6. Các tính năng chính 28](#_Toc532210255)

[CHƯƠNG 4: ASP.NET CORE VÀ MONGODB DRIVER 34](#_Toc532210256)

[4.1. Giới thiệu 34](#_Toc532210257)

[4.2. Vai trò 34](#_Toc532210258)

[4.3. Môi trường lập trình 35](#_Toc532210259)

[4.3.1. Trình soạn thảo 35](#_Toc532210260)

[4.3.2. Tạo ứng dụng Web ASP.NET Core 35](#_Toc532210261)

[4.4. ASP.NET Core WebAPI cơ bản 37](#_Toc532210262)

[4.4.1. Cấu trúc 37](#_Toc532210263)

[4.4.2. Cú pháp 37](#_Toc532210264)

[4.4.3. Khởi chạy 40](#_Toc532210265)

[4.5. Sử dụng NuGet Package 40](#_Toc532210266)

[4.6. MongoDB Driver trong ASP.NET Core 41](#_Toc532210267)

[4.6.1. Model 41](#_Toc532210268)

[4.6.2. Database Context 42](#_Toc532210269)

[4.6.3. Repository 43](#_Toc532210270)

[4.7. Dependency Injection trong ASP.NET Core 44](#_Toc532210271)

[4.7.1. Khái niệm Inversion of Control 44](#_Toc532210272)

[4.7.2. Khái niệm Dependency Injection 44](#_Toc532210273)

[4.7.3. Sử dụng Dependency trong ASP.NET Core 44](#_Toc532210274)

[CHƯƠNG 5: SIGNALR 47](#_Toc532210275)

[5.1.1. SignalR là gì? 47](#_Toc532210276)

[5.1.2. Giới thiệu 47](#_Toc532210277)

[5.1.3. SignalR trong ASP.NET Core 48](#_Toc532210278)

[5.1.4. Mục đích sử dụng SignalR 48](#_Toc532210279)

[5.1.5. Thuật ngữ Middleware 48](#_Toc532210280)

[5.2. Cài đặt 49](#_Toc532210281)

[5.2.1. DotNET Client 49](#_Toc532210282)

[5.2.2. Javascript Client 49](#_Toc532210283)

[5.3. Ứng dụng 50](#_Toc532210284)

[5.3.1. Thiết lập lắng nghe 50](#_Toc532210285)

[5.3.2. Thiết lập kết nối 51](#_Toc532210286)

[5.3.3. Giao tiếp 52](#_Toc532210287)

[Phần 2: Áp dụng 53](#_Toc532210288)

[CHƯƠNG 6: GIẢI QUYẾT VẤN ĐỀ 53](#_Toc532210289)

[6.1. Đặt tả vấn đề 53](#_Toc532210290)

[6.2. Thiết kế 54](#_Toc532210291)

[6.2.1. Mô hình Use-Case 54](#_Toc532210292)

[6.2.2. Xây dựng cơ sở dữ liệu 58](#_Toc532210293)

[6.2.3. Mô hình cơ sở dữ liệu quan hệ 63](#_Toc532210294)

[6.2.4. Mô hình lưu trữ MongoDB 64](#_Toc532210295)

[6.2.5. Sơ đồ phân rã chức năng 65](#_Toc532210296)

[6.2.6. Dòng dữ liệu 66](#_Toc532210297)

[6.3. Thiết kế cơ bản 68](#_Toc532210298)

[6.3.1. Mô hình hoạt động 68](#_Toc532210299)

[6.3.2. Thiết kế mô hình lớp 69](#_Toc532210300)

[6.4. Thiết kế quy trình 71](#_Toc532210301)

[6.4.1. Chức năng đăng ký 71](#_Toc532210302)

[6.4.2. Chức năng đăng nhập 72](#_Toc532210303)

[6.4.3. Chức năng tải nhạc 73](#_Toc532210304)

[6.4.4. Chức năng xóa nhạc 74](#_Toc532210305)

[CHƯƠNG 7: TỔNG KẾT 76](#_Toc532210306)

[7.1. Kết quả đạt được 76](#_Toc532210307)

[7.2. Các chức năng nổi bật 76](#_Toc532210308)

[7.2.1. Trang chủ 76](#_Toc532210309)

[7.2.2. Trang quản lý 76](#_Toc532210310)

[7.2.3. Phòng trò chuyện 79](#_Toc532210311)

[7.3. Đánh giá 81](#_Toc532210312)

[7.3.1. Kết quả đạt được 81](#_Toc532210313)

[7.3.2. Yếu điểm 81](#_Toc532210314)

[7.3.3. Hướng phát triển 81](#_Toc532210315)

[PHỤ LỤC 82](#_Toc532210316)

[TÀI LIỆU THAM KHẢO 95](#_Toc532210317)

[DẪN NGUỒN 97](#_Toc532210318)

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 | Tệp 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. |
| BSON | 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](#_Toc532209898)

[Hình 1. 2. Thị phần YouTube quá dỗi lớn mạnh (nguồn: Youtube.com) 3](#_Toc532209899)

[Hình 1. 3. Tính năng partial load của Youtube 3](#_Toc532209900)

[Hình 1. 4. Biểu tượng Amnhac.com 4](#_Toc532209901)

[Hình 1. 5. Thống kê số lượng request đáp ứng mỗi giây (nguồn: https://stackify.com/asp-net-core-features/) 5](#_Toc532209902)

[Hình 1. 6. So sánh MongoDB và các cơ sở dữ liệu tốt nhất hiện nay (nguồn: ArangoDB) 6](#_Toc532209903)

[Hình 1. 7. Mô hình sơ lược thiết kế ứng dụng tổng quát 7](#_Toc532209904)

Chương 2:

[Hình 2. 1. So sánh NoSQL và SQL (nguồn: TechBlog) 9](#_Toc532209890)

[Hình 2. 2. (nguồn: Mongodb.com) 10](#_Toc532209891)

[Hình 2. 3. Giao diện dòng lệnh cho Mongo 12](#_Toc532209892)

[Hình 2. 4. Giao diện kết nối cơ sở dữ liệu trong MongoDb Compass 13](#_Toc532209893)

[Hình 2. 5. Giao diện sau khi kết nối máy chủ thành công 14](#_Toc532209894)

[Hình 2. 6. Nhấp chọn một Database để xem hoặc thêm các Collection bên trong 14](#_Toc532209895)

[Hình 2. 7. Bên trong một Collection sẽ được hỗ trợ Query, Thêm, sửa, xóa 15](#_Toc532209896)

[Hình 2. 8. Khi chỉnh sửa có thể thay đổi dữ liệu, kiểu dữ liệu và xóa dữ liệu 15](#_Toc532209897)

Chương 3:

[Hình 3. 1. Biểu tượng của Angular (nguồn: Angular.io) 16](#_Toc532209857)

[Hình 3. 2. Tổng quan mô hình MVC của Angular (nguồn: Codeburst.io) 17](#_Toc532209858)

[Hình 3. 3. Biểu tượng NodeJS (nguồn: NodeJS.org) 17](#_Toc532209859)

[Hình 3. 4. Quy trình tạo ứng dụng Angular đầu tiên 18](#_Toc532209860)

[Hình 3. 5. Các tập tin và thư mục của ứng dụng Angular 19](#_Toc532209861)

[Hình 3. 6. Biểu tượng Visual Studio Code (nguồn: visualstudio.microsoft.com) 19](#_Toc532209862)

[Hình 3. 7. Cài đặt Extension cho Visual Studio Code 20](#_Toc532209863)

[Hình 3. 8. Mô hình hoạt động của Angular (nguồn: Ciphertrick.com) 22](#_Toc532209864)

[Hình 3. 9. Cú pháp định nghĩa Component 23](#_Toc532209865)

[Hình 3. 10. Cách Angular nhét (manipulate) một Component ra giao diện 24](#_Toc532209866)

[Hình 3. 11. Cú pháp định nghĩa Module 25](#_Toc532209867)

[Hình 3. 12. Cú pháp định nghĩa Service 26](#_Toc532209868)

[Hình 3. 13. Một Service được sử dụng bởi nhiều trang và thành phần 26](#_Toc532209869)

[Hình 3. 14. Khai báo Service trong constructor để Angular inject vào Component 26](#_Toc532209870)

[Hình 3. 15. Khai báo các route 27](#_Toc532209871)

[Hình 3. 16. 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 27](#_Toc532209872)

[Hình 3. 17. AppModule nạp AppRouterModule vừa tạo 27](#_Toc532209873)

[Hình 3. 18. Khai báo router-outlet tại thành phần cần điều hướng trang con, ở đây là AppComponent 28](#_Toc532209874)

[Hình 3. 19. Tạo biến trong một Component (class) 28](#_Toc532209875)

[Hình 3. 20. Cú pháp in biến trong class ra ngoài template (html) 28](#_Toc532209876)

[Hình 3. 21. Kết quả hiển thị 29](#_Toc532209877)

[Hình 3. 22. Dùng ngModel để ràng buộc đa chiều 29](#_Toc532209878)

[Hình 3. 23. Khai báo module chức năng để hỗ trợ thuộc tính ngModel 29](#_Toc532209879)

[Hình 3. 24. Hiển thị ở giao diện 30](#_Toc532209880)

[Hình 3. 25. Hiển thị sau khi thay đổi hộp thoại 30](#_Toc532209881)

[Hình 3. 26. Khai báo một hàm trong Component (class) 30](#_Toc532209882)

[Hình 3. 27. Gán hàm gọi trên như một sự kiện ở phía template 31](#_Toc532209883)

[Hình 3. 28. Hiển thị lúc này ở trình duyệt 31](#_Toc532209884)

[Hình 3. 29. Nhấp nút đã được gán sự kiện trên và xem thay đổi 31](#_Toc532209885)

[Hình 3. 30. Template (bên phải) và hiển thị ở phía trình duyệt (bên trái) 31](#_Toc532209886)

[Hình 3. 31. Template (bên phải) dùng ngFor và hiển thị ở phía trình duyệt (bên trái) 32](#_Toc532209887)

[Hình 3. 32. Giá trị của biến array ở phía Component 32](#_Toc532209888)

[Hình 3. 33. Vòng đời của một trang Angular 33](#_Toc532209889)

Chương 4:

[Hình 4. 1. ASP.NET Core (nguồn: asp.net) 34](#_Toc532209838)

[Hình 4. 2. Visual Studio (nguồn: visualstudio.microsoft.com) 35](#_Toc532209839)

[Hình 4. 3. Tạo ứng dụng ASP.NET Core qua Visual Studio 2017 35](#_Toc532209840)

[Hình 4. 4. Chọn Template cho ứng dụng 36](#_Toc532209841)

[Hình 4. 5. Cấu trúc cơ bản của ứng dụng 37](#_Toc532209842)

[Hình 4. 6. Cú pháp trong Controller của ASP.NET Core 38](#_Toc532209843)

[Hình 4. 7. Các dòng lệnh trong tệp Startup.cs 39](#_Toc532209844)

[Hình 4. 8. Debug ứng dụng 40](#_Toc532209845)

[Hình 4. 9. Tạo bản dựng cho ứng dụng qua giao diện Visual Studio 2017 40](#_Toc532209846)

[Hình 4. 10. Mở Menu để quản lý gói mở rộng NuGet 41](#_Toc532209847)

[Hình 4. 11. MongoDB Driver trên NuGet Packages 41](#_Toc532209848)

[Hình 4. 12. Tạo một Model đơn giản nhằm lưu nó xuống Database sau này 42](#_Toc532209849)

[Hình 4. 13. Khởi tạo class DataAccess kết nối đến Mongo Database 43](#_Toc532209850)

[Hình 4. 14. Một Repository cơ bản chứa DataAccess 43](#_Toc532209851)

[Hình 4. 15. Gọi Repository ra trong Controller để truy vấn 44](#_Toc532209852)

[Hình 4. 16. Gọi AddSingleton để tạo một Singleton của Repository khi Runtime 45](#_Toc532209853)

[Hình 4. 17. Khai báo Repository trong Constructor của Controller để ASP.NET Core inject 45](#_Toc532209854)

[Hình 4. 18. Áp dụng với đa hình trong C# 46](#_Toc532209855)

[Hình 4. 19. Khai báo một đối tượng Đa hình như Singleton 46](#_Toc532209856)

Chương 5:

[Hình 5. 1. Công nghệ SignalR (nguồn: microsoft.com) 47](#_Toc532209818)

[Hình 5. 2. Giao tiếp thời gian thực giữa Client và Server (nguồn: c-sharpcorner.com) 47](#_Toc532209819)

[Hình 5. 3. Một Request của người dùng có thể thông qua một Middleware hoặc nhiều Middleware tùy theo thiết lập logic bên trong 49](#_Toc532209820)

[Hình 5. 4. SignalR cho ASP.NET Core trên NuGet packages 49](#_Toc532209821)

[Hình 5. 5. Một SignalR Hub cơ bản 50](#_Toc532209822)

[Hình 5. 6. Thêm SignalR như một phần của Service trong ASP.NET Core 50](#_Toc532209823)

[Hình 5. 7. Tạo một Middleware cho SignalR 51](#_Toc532209824)

[Hình 5. 8. Yêu cầu kết nối ở phía Client (nguồn: docs.microsoft.com) 51](#_Toc532209825)

[Hình 5. 9. Thiết đặt lắng nghe sự kiện tên ReceiveMessage ở phía Client (nguồn: docs.microsoft.com) 52](#_Toc532209826)

[Hình 5. 10. Câu lệnh gửi tin nhắn đến máy tất cả các máy con đang lắng nghe hàm ReceiveMessage 52](#_Toc532209827)

[Hình 5. 11. Gọi lệnh trên Client có tên SendMessage ở phía máy Server và chờ kết quả trả về 52](#_Toc532209828)

Chương 6:

[Hình 6. 1. Mô hình Use Case mức cơ bản 54](#_Toc532209798)

[Hình 6. 2. Use Case mức trung ở phía Khách hàng 55](#_Toc532209799)

[Hình 6. 3. Use case mức trung đối với quản trị viên 56](#_Toc532209800)

[Hình 6. 4. Use case chi tiết Quản lý danh sách phát phía người dùng 57](#_Toc532209801)

[Hình 6. 5. Use case chi tiết cho chức năng quản lý bài hát của Quản trị viên 57](#_Toc532209802)

[Hình 6. 6. Mô hình dữ liệu vật lý 63](#_Toc532209803)

[Hình 6. 7. Mô hình lưu trữ trong MongoDB 64](#_Toc532209804)

[Hình 6. 8. Sơ đồ phân rã chức năng tổng quát 65](#_Toc532209805)

[Hình 6. 9. Sơ đồ dòng dữ liệu tổng quan 66](#_Toc532209806)

[Hình 6. 10. Sơ đồ dòng dữ liệu mức người dùng 66](#_Toc532209807)

[Hình 6. 11. Sơ đồ dòng dữ liệu ở quản trị viên 67](#_Toc532209808)

[Hình 6. 12. Dòng dữ liệu khi tải một bài hát lên (mức 2) 67](#_Toc532209809)

[Hình 6. 13. Mô hình hoạt động của Amnhac.com 68](#_Toc532209810)

[Hình 6. 14. Mô hình lớp cho phần Quản lý bài hát 69](#_Toc532209811)

[Hình 6. 15. Mô hình lớp cho phần Tài khoản người dùng 70](#_Toc532209812)

[Hình 6. 16. Tuần tự quy trình khi người dùng đăng ký trên ứng dụng 71](#_Toc532209813)

[Hình 6. 17. Quy trình đăng nhập của người dùng 72](#_Toc532209814)

[Hình 6. 18. Quy trình tải nhạc lên máy chủ 73](#_Toc532209815)

[Hình 6. 19. Quy trình xóa một bài hát trong Amnhac.com (1) 74](#_Toc532209816)

[Hình 6. 20. Quy trình xóa một bài hát trong Amnhac.com (2) 74](#_Toc532209817)

Chương 7:

[Hình 7. 1. Giao diện trang chủ Amnhac.com 76](#_Toc532209787)

[Hình 7. 2. Vào mục đăng nhập ở trên thanh tìm kiếm 76](#_Toc532209788)

[Hình 7. 3. Dùng tài khoản đăng nhập hoặc đăng ký mới 77](#_Toc532209789)

[Hình 7. 4. Từ thanh tìm kiếm, vào mục quản lý 77](#_Toc532209790)

[Hình 7. 5. Giao diện quản lý bài hát 77](#_Toc532209791)

[Hình 7. 6. Thử tải một bài hát và chọn bài hát vừa tải để duyệt 78](#_Toc532209792)

[Hình 7. 7. Bật máy chơi nhạc ở biểu tượng góc dưới màn hình và vào danh sách phát 78](#_Toc532209793)

[Hình 7. 8. Giao diện chơi nhạc khi phát một bài hát 78](#_Toc532209794)

[Hình 7. 9. Về trang chủ và vào mục Trò chuyện 79](#_Toc532209795)

[Hình 7. 10. Vào một phòng hoặc tạo phòng mới (khi là quản trị viên) 79](#_Toc532209796)

[Hình 7. 11. Xem danh mục trò chuyện 80](#_Toc532209797)

Phụ lục

[Hình 1. Tải về MongoDB tại mongodb.com 82](#_Toc532209746)

[Hình 2. Giao diện cài đặt NodeJS 83](#_Toc532209747)

[Hình 3. Kiểm tra phiên bản NodeJS 84](#_Toc532209748)

[Hình 4. Thiết lập appsettings.json phía ASP.NET Core server 85](#_Toc532209749)

[Hình 5. Mở thao mục PowerShell 85](#_Toc532209750)

[Hình 6. Chạy thử máy chủ 85](#_Toc532209751)

[Hình 7. Cài đặt các thành phần còn thiếu bằng lệnh npm install 86](#_Toc532209752)

[Hình 8. Chạy thử máy chủ development của Angular 86](#_Toc532209753)

[Hình 9. Giao diện trang chủ 86](#_Toc532209754)

[Hình 10 87](#_Toc532209755)

[Hình 11 88](#_Toc532209756)

[Hình 12 88](#_Toc532209757)

[Hình 13 89](#_Toc532209758)

[Hình 14 89](#_Toc532209759)

[Hình 15 90](#_Toc532209760)

[Hình 16 90](#_Toc532209761)

[Hình 17 90](#_Toc532209762)

[Hình 18 91](#_Toc532209763)

[Hình 19 91](#_Toc532209764)

[Hình 20 91](#_Toc532209765)

[Hình 21. Khởi tạo thiết lập cho Jwt Bearer Factory 92](#_Toc532209766)

[Hình 22. Inject các option vào Factory 92](#_Toc532209767)

[Hình 23. Tạo các Claim và nhờ Jwt mã hóa chúng 92](#_Toc532209768)

[Hình 24. Token được mã hóa gửi về Client 93](#_Toc532209769)

[Hình 25. Đặt thẻ chú thích Authorize để yêu cầu gửi kèm Token mỗi yêu cầu 93](#_Toc532209770)

[Hình 26. Token được gửi kèm trong Header để xác thực 93](#_Toc532209771)

[Hình 27 93](#_Toc532209772)

[Hình 28 94](#_Toc532209773)

[Hình 29 94](#_Toc532209774)

[Hình 30 94](#_Toc532209775)

DANH MỤC BẢNG

Chương 2:

[Bảng 2. 1. Định nghĩa thuật ngữ trong MongoDB 11](#_Toc532210030)

[Bảng 2. 2. Các câu lệnh đơn giản trong MongoDB 13](#_Toc532210031)

Chương 3:

[Bảng 3. 1. Cú pháp dòng lệnh hay sử dụng trong Angular CLI 21](#_Toc532210041)

Chương 6:

[Bảng 6. 1. Thực thể Permission 58](#_Toc532210057)

[Bảng 6. 2. Thực thể người dùng 58](#_Toc532210058)

[Bảng 6. 3. Thực thể lưu trữ danh sách quốc gia 59](#_Toc532210059)

[Bảng 6. 4. Thực thể lưu trữ nghệ sĩ 59](#_Toc532210060)

[Bảng 6. 5. Thực thể lưu thể loại nhạc 60](#_Toc532210061)

[Bảng 6. 6. Thực thể phân loại thể loại nhạc 60](#_Toc532210062)

[Bảng 6. 7. Thực thể đường dẫn nhạc 60](#_Toc532210063)

[Bảng 6. 8. Thực thể lưu trữ bài hát 61](#_Toc532210064)

[Bảng 6. 9. Thực thể lưu trữ Album 62](#_Toc532210065)

[Bảng 6. 10. Thực thể lưu lịch sử lượt nghe 62](#_Toc532210066)

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.

Với dự án Amnhac.com, em sẽ mang lại một tư duy rất khác về xây dựng một trang web Âm nhạc tại Việt Nam. Một trang web mà coi trọng bản quyền, thông tin khách hàng, một trang web tiện lợi, tối ưu.

Amnhac.com sẽ sử dụng JWT Token cùng các phương thức mã hóa SHA256 và AES để bảo mật tài khoản người dùng. Tuân theo các đạo luật Châu Âu EU GDPR về Cookie và quản lý thông tin người sử dụng, Amnhac.com sẽ lưu trữ dữ liệu trên máy tính của họ nhưng không làm lộ thông tin người dùng.

Bên cạnh đó, trải nghiệm của người dùng sẽ được cải thiện hơn nhờ công nghệ Angulars’s Single Page Website, giúp người dùng nghe nhạc mà không bị gián đoạn khi đọc tin tức trên cùng một trang.

Một dịch vụ web nền tảng ASP.NET Core 2.1 mới nhất, giúp cho dự án có thể mở rộng sang đa nền tảng, đa máy chủ, được đi cùng với MongoDB, một cơ sở dữ liệu NoSQL nhằm đáp ứng nhu cầu của hơn hàng triệu người dùng cùng một lúc mà vẫn đảm bảo tốc độ phản hồi.

Trải nghiệm giao tiếp thời gian thực cũng sẽ được tích hợp trong Amnhac.com, nơi các thông báo và tin nhắn giao tiếp giữa cộng đồng người yêu nhạc. Với SignalR dành cho ASP.NET Core, thiết lập một Hub Proxy cùng phương thức bảo mật tin tức MessagePack Protocol được đi kèm sẽ giúp cho trang web thời gian thực vừa nhanh, vừa an toàn cho người dùng.

Hi vọng rằng luận văn này sẽ giúp mọi người đến gần hơn với các công nghệ nóng nhất hiện nay, những công nghệ dành cho tương lai : Angular, ASP.NET Core, MongoDB và SignalR, cũng như một lời mở đầu cho dự án nghe nhạc trực tuyến lớn tại Việt Nam: Amnhac.com.

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, we should becareful about user information more than before, and especially the copyright of every song which uploaded to our site.

Amnhac.com then will be the wake up call for everyone in Vietnam. We do care about our user’s information security. We do care about copyright of song, and we will pay for it.

Amnhac.com will introduce you how we secure our customer’s information by using JWT Token with SHA256 hash algorithm and AES (Advance Encryption Standard). We will base on the EU’s General Data Protection Regulation (GDPR) for our data privacy and cookie usage.

Beside, the website will also deliver the best experience for users by using Angular’s Single Page Website technology. Your experience will be a smooth and enjoyable moment in melodies while reading news on our page.

With ASP.NET Core 2.1 as a Web Service, it could also make this project flexible, extensible, fast and cross-platform. With MongoDB Driver, a NoSQL Database along with ASP.NET Core, we promise it will bring you the best experience ever, this website is aimed for serving more than a million users at a time.

Further, we also intergrated SignalR as a Hub for real-time communicate. This will assurance that the communicate between our users and server are real-time, optimized, fast and secure. SignalR with MessagePack Protocol will ensure your data are being encrypted and compressed before it is sent.

Interest? Hopefully the thesis will deliver you a great knowledge of future’s technologies, the best technologies: Angular, ASP.NET Core, MongoDB and SignalR. As it was also an introduction about Amnhac.com, which will become the biggest website about Music Marketplace in Vietnam in the future.

CHƯƠNG 1:   
GIỚI THIỆU TỔNG QUAN

* 1. Tổng quan
     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 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. 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]



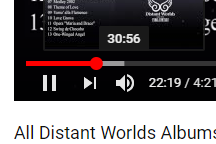
Hình 1. 1. Tốc độ tải trang chưa được tối ưu (mp3.zing.vn)

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 dỗi lớn mạnh (nguồn: Youtube.com)



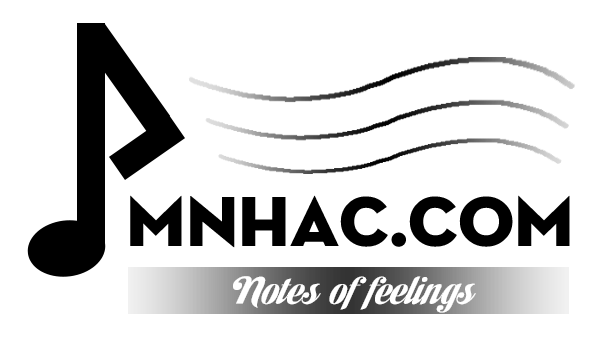
Hình 1. . Tính năng partial load của Youtube

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. 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 vi vu, 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. 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.
  + Nghe nhạc miễn phí, nhanh và trải nghiệm tuyệt vời như YouTube
  + Cho phép người dùng chia sẽ, tải nhạc, kể cả các bản nhạc cover họ yêu thích.
  + Xây dựng cộng đồng âm nhạc mới, đặc biệt dành cho những ai yêu âm nhạc, ca hát, hay sáng tác.
  + Là nơi chia sẽ 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.



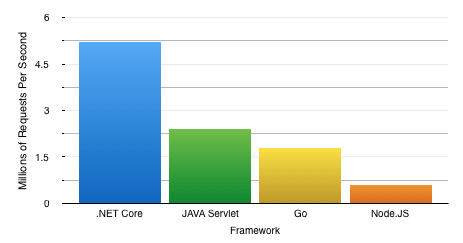
Hình 1. . Biểu tượng Amnhac.com

* 1. 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.

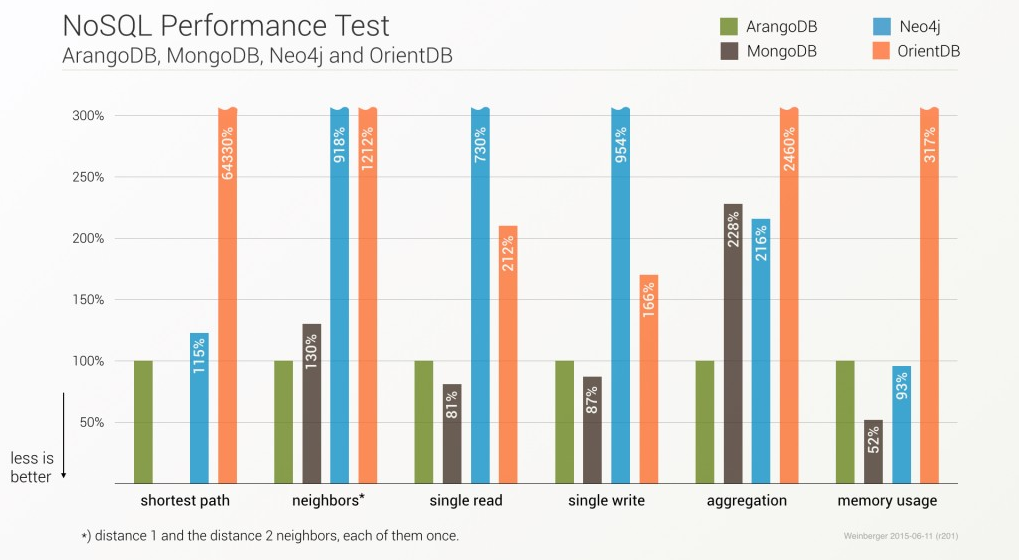
Song, để đáp ứng hầu hết nhu cầu, và có thể mở rộng lên đa nền tảng về sau, chúng ta sẽ đi theo hướng Dịch vụ Web, mà sử dụng Web View là bước tiên phong để phát triển thị trường.

* 1. 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 dỗi vượt trội, lại đa nền tảng.



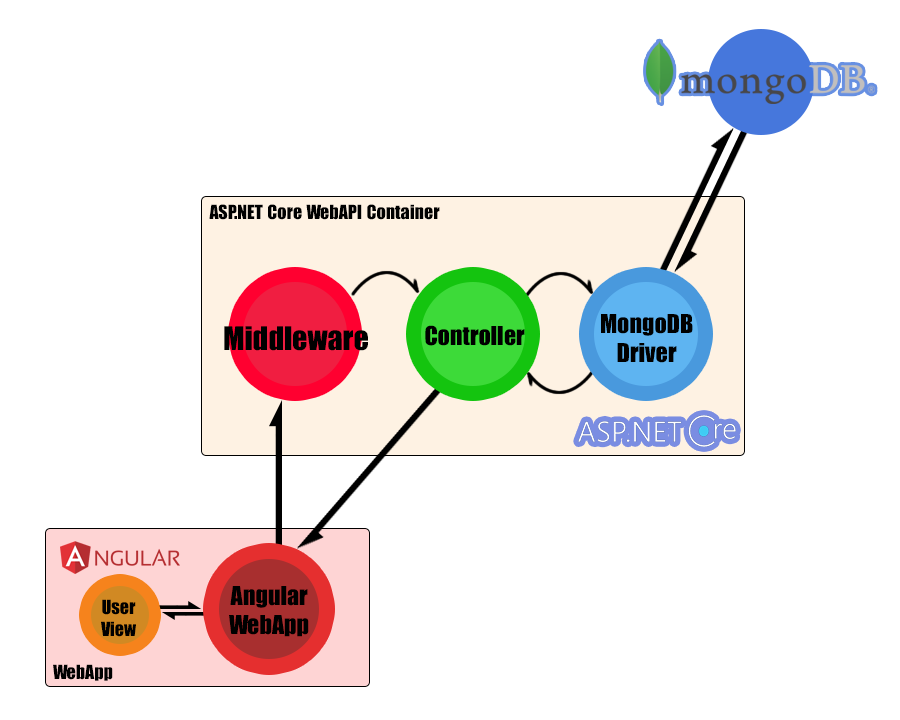
Hình 1. 5. Thống kê số lượng request đáp ứng mỗi giây (nguồn: https://stackify.com/asp-net-core-features/)

* 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. Bố cục

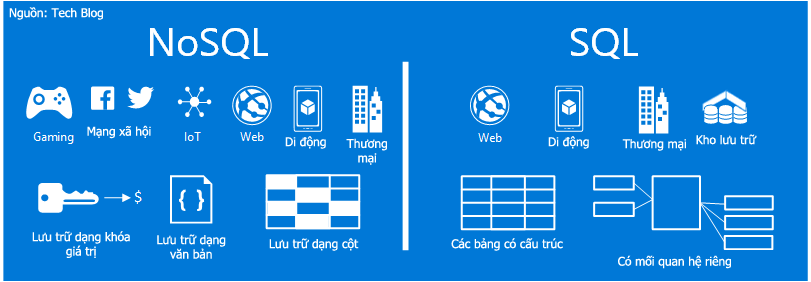
Do khá nhiều công nghệ được áp dụng, cho nên để có thể dễ dàng hiểu và nghiên cứu về sau, luận văn này được chia thành 3 phần chính, bao gồm:

* **Phần cơ sở lý thuyết:** 4 Chương
  + MongoDB: Giới thiệu về cơ sở dữ liệu NoSQL và MongoDB
  + Angular: Giới thiệu về Angular
  + ASP.NET Core: Giới thiệu về ASP.NET Core
  + SignalR: Giới thiệu về SignalR trong ASP.NET Core
* **Phần áp dụng:** 1 Chương
  + Giải quyết vấn đề
* **Phần tổng kết:** 1 Chương
  + 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**

* 1. NoSQL
     1. Giới thiệu



Hình 2. . 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)

* + 1. 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.
  + 1. 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
  + 1. 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.
  1. MongoDB
     1. Tại sao nên dùng MongoDB?



Hình 2. . (nguồn: Mongodb.com)

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.

* + 1. 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.

* + 1. **Định nghĩa**

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

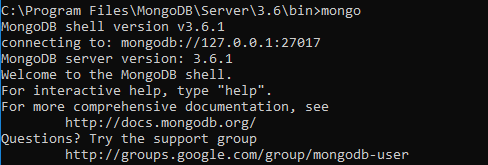
|  |  |  |
| --- | --- | --- |
| **SQL** | **MongoDb** | **Ý nghĩa** |
| Database | Database | Một cơ sở dữ liệu, dùng lưu trữ dữ liệu. |
| Table | Collection | Một tập dữ liệu, chứa nhiều dòng dữ liệu |
| Row | Document | Một dòng dữ liệu |
| Column | Field | Một cột dữ liệu, được định nghĩa kiểu dữ 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

* + 1. 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.



Hình 2. 3. 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 *collection* | db.user.find(); |
| db.collection.insert() | Thêm dữ liệu mới | db.user.insert({  username:’DHCT’,  password:’CTU’  }); |
| db.collection.update() | Cập nhật dòng dữ liệu | db.users.update({  age: { $gt: 25 } },  { $set: { status: 'C' } },  { multi: true }); |
| s.start\_transaction() | Bắt đầu một phiên giao dịch Transaction |  |
| s.commit\_transaction() | Hoàn tất phiên giao dịch Transaction |  |
| db.createUser({  user:<tài khoản>,  pwd:<mật khẩu>,  roles:[<danh sách quyền>]  }) | Tạo người dùng với tài khoản, mật khẩu, kèm cấp quyền | db.createUser({  user: "accountUser",  pwd: "password",  roles: [ "readWrite", "dbAdmin" ]}) |

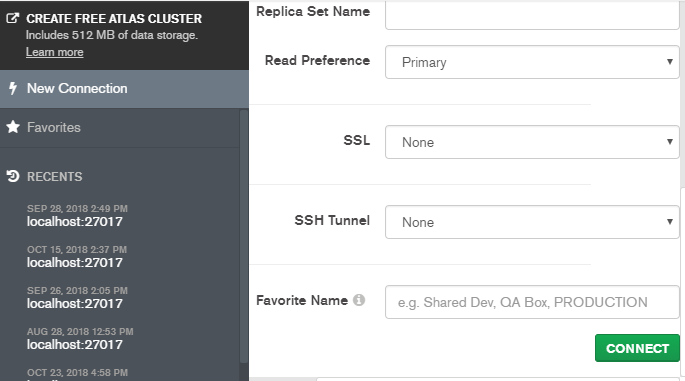
Bảng 2. 2. Các câu lệnh đơn giản trong MongoDB

* + 1. MongoDb Compass

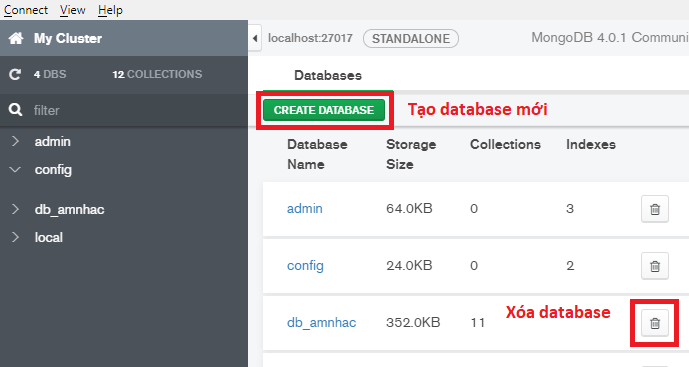
Đây là công cụ giúp quản trị cơ sở dữ liệu của MongoDb dưới giao diện người dùng thân thiện hơn.

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

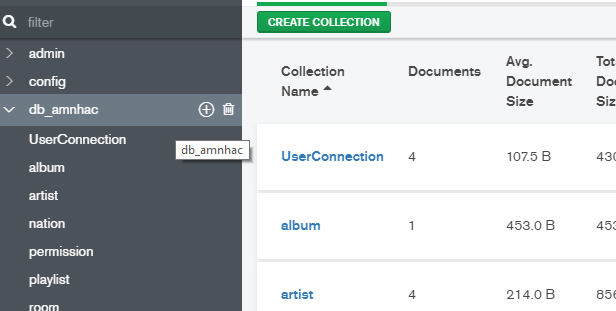
* **Phiên bản chỉ đọc (Readonly Edition)**
* **Phiên bản độc lập (Isolated Edition):** Chỉ dùng cho bản trả phí.
* **Phiên bản cộng đồng (Community Edition):** Bản miễn phí



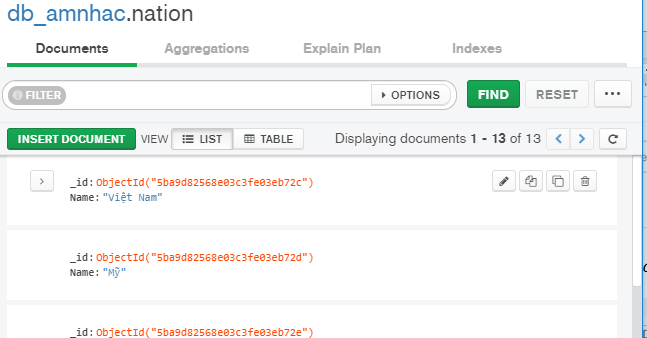
Hình 2. . Giao diện kết nối cơ sở dữ liệu trong MongoDb Compass



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



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



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



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

CHƯƠNG 3:   
ANGULAR

* 1. **Giới thiệu về Angular**



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

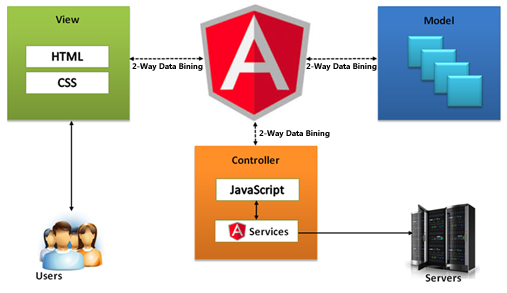
Là một Javascript Framework nổi danh, được đầu tư bởi Google, chức năng nổi bật nhất là dùng để thiết kế một trang web đơn trang (Single Page Component).

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.

* 1. **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. . 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.

* 1. **Giới thiệu về NodeJS**



Hình 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.

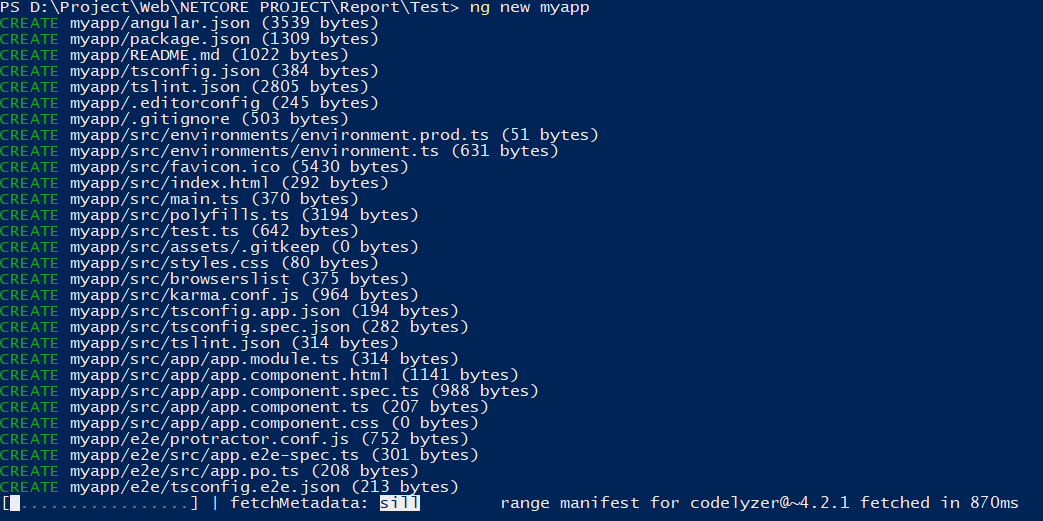
* 1. **Lập trình**
     1. **Lập trình trên môi trường mới**

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

Sau đó, gõ lệnh:

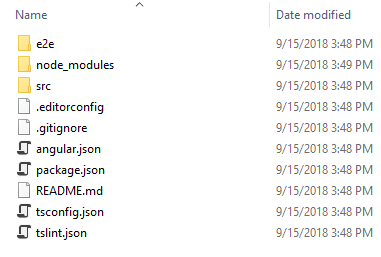
*ng new myapp*

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



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

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



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

* + 1. **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.

* + 1. **Công cụ soạn thảo**

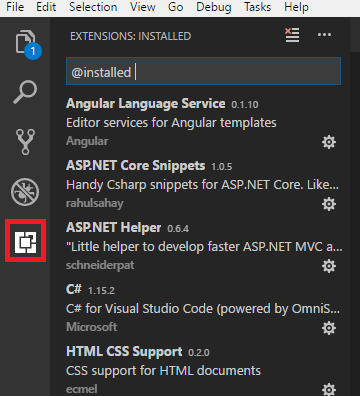
Bất kỳ chương trình hỗ trợ soạn thảo nào hỗ trợ HTML5 và CSS3 đều có thể dùng để lập trình ứng dụng Angular.

Khuyến nghị nên sử dụng Visual Studio Code mới nhất



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

* + 1. **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 *ng g c <tên>*) | Tạo component (Web Page/ thành phần Web Page) trong Angular. |
| ng generate service <tên>  (hoặc *ng g s <tên>*) | Tạo Service trong Angular |
| ng generate module <tên>  (hoặc *ng g m <tên>*) | Tạo Module trong Angular |

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

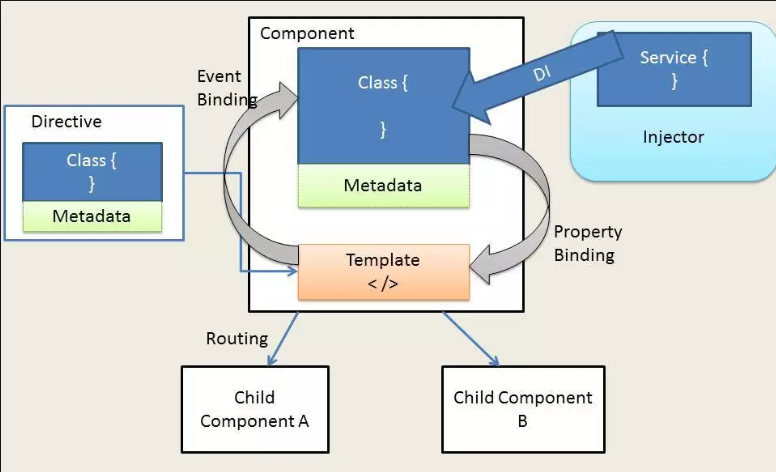
* + 1. **Các thành phần chính**

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

* + - 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. . 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.

* + - 1. **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:

* **Tệp 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:

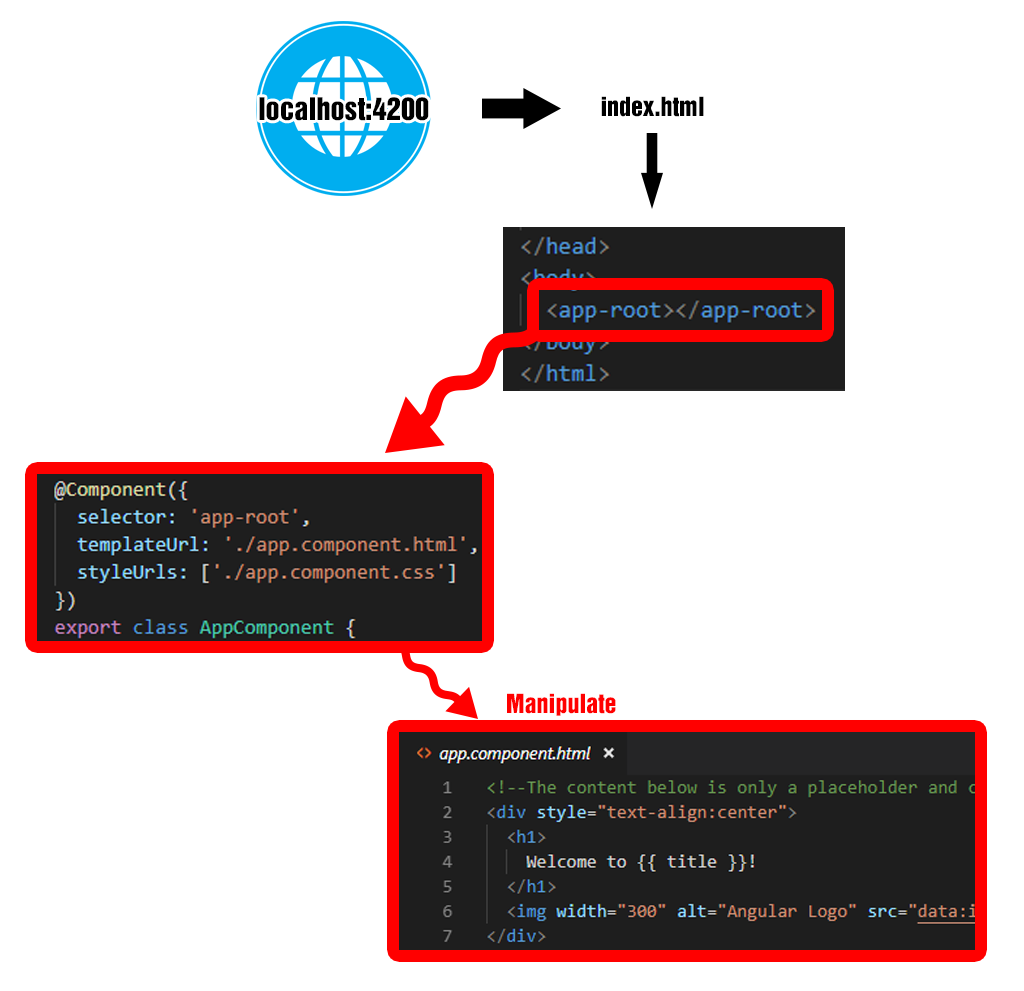


Hình 3. . Cú pháp định nghĩa Component

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. . 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.

* + - 1. **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.



Hình 3. . 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.
  + - 1. **Service**

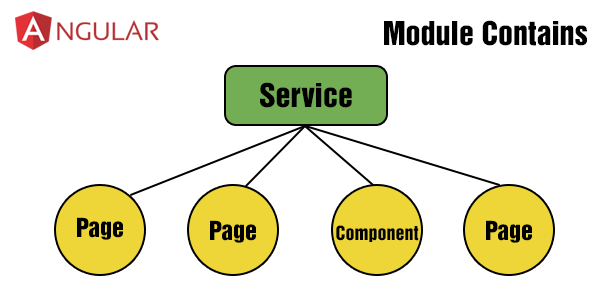
Khi đặt thẻ *@Injectable* trước một class, nó sẽ trở thành một Service.

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.



Hình 3. . Cú pháp định nghĩa Service

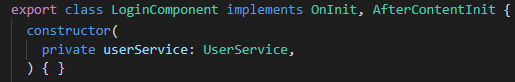
Đối với providedIn là root, Service sẽ được khởi tạo ngay khi truy cập và có thể sử dụng ở toàn bộ các trang.



Hình 3. . 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.



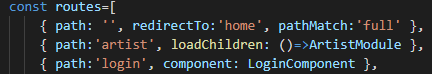
Hình 3. . Khai báo Service trong constructor để Angular inject vào Component

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

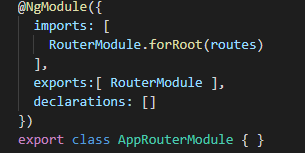
* + - 1. **RouterModule**

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.

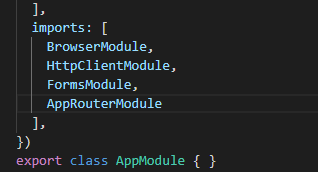
Song, nếu người dùng có quá nhiều thành phần, hoặc các tập thành phần lớn cần tách thành một trang thì nên sử dụng *RouterModule*.



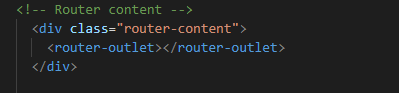
Hình 3. . Khai báo các route



Hình 3. . 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



Hình 3. . AppModule nạp AppRouterModule vừa tạo



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

* + 1. **Các tính năng chính**
       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



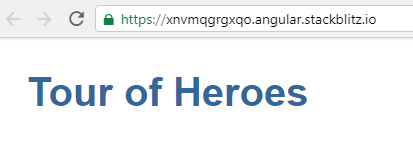
Hình 3. . Tạo biến trong một Component (class)

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



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

Kết quả khi mở trình duyệt:



Hình 3. . 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:



Hình 3. . 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



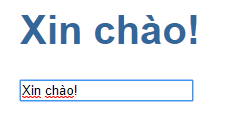
Hình 3. . Khai báo module chức năng để hỗ trợ thuộc tính ngModel

Xem thành quả đạt được:



Hình 3. . Hiển thị ở giao diện

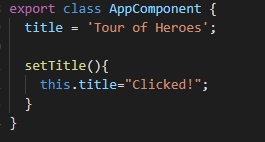
Thay đổi hộp thoại bên dưới và xem sự thay đổi đó:



Hình 3. . Hiển thị sau khi thay đổi hộp thoại

* + - 1. **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.



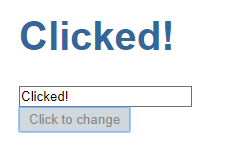
Hình 3. . Khai báo một hàm trong Component (class)



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



Hình 3. . Hiển thị lúc này ở trình duyệt



Hình 3. . 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,…

* + - 1. **Khai báo điều kiện**

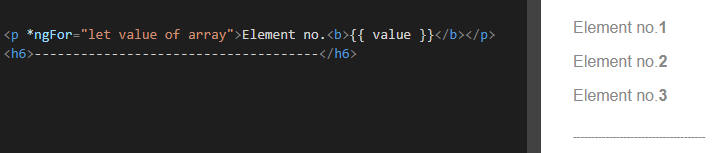
Với thuộc tính **\*ngIf** , nếu thỏa mãn điều kiện thì thẻ HTML kèm toàn bộ con của thẻ sẽ hiện ra, ngược lại sẽ không được tạo ra:



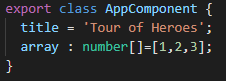
Hình 3. . Template (bên phải) và hiển thị ở phía trình duyệt (bên trái)

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 thử của một mảng:



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

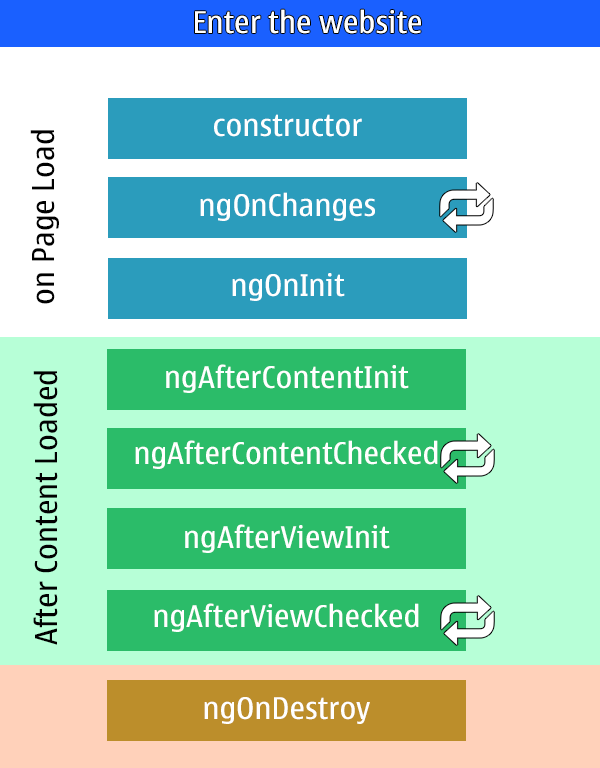


Hình 3. . Giá trị của biến array ở phía Component

* + - 1. **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. . Vòng đời của một trang Angular

CHƯƠNG 4:   
ASP.NET CORE VÀ MONGODB DRIVER



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

* 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:

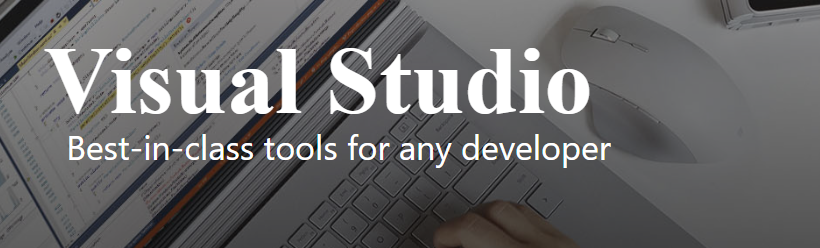
* Miễn phí
* Ngôn ngữ dễ học, hướng đối tượng
* Đa nền tảng
* Tốc độ chạy vô cùng mượt mà và tối ưu.
* Tính mở rộng cao, tùy biến tốt.
* Cộng đồng đông đúc, thư viện hỗ trợ ngày càng phong phú.
  1. Vai trò
* **Ứng dụng Web bình thường:** Một trang web html bình thường.
* **ASP.NET Core MVC:** Một ứng 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.

* 1. Môi trường lập trình
     1. 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**

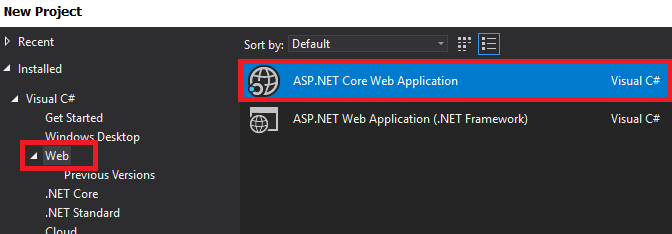


Hình 4. . Visual Studio (nguồn: visualstudio.microsoft.com)

* + 1. 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. . Tạo ứng dụng ASP.NET Core qua Visual Studio 2017

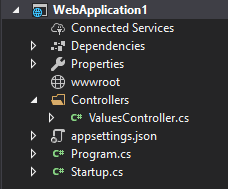
Chọn API để tạo ứng dụng Web ASP.NET Service (Dịch vụ Web)



Hình 4. . Chọn Template cho ứng dụng

Hoàn tất bước tạo lập dự án ASP.NET Core

* 1. ASP.NET Core WebAPI cơ bản
     1. Cấu trúc

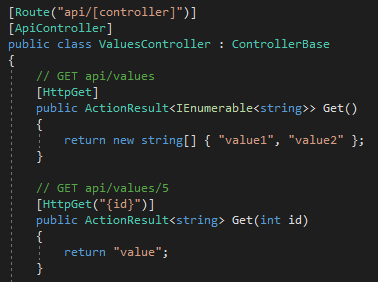


Hình 4. . Cấu trúc cơ bản của ứng dụng

* **Program.cs:** Tệp chạy chính, chứa các câu lệnh khởi chạy một ứng dụng DotNet Core.
* **Startup.cs:** Tệp chứa các lệnh giúp thiết lập các tính năng của ứng dụng Web.
* **Appsettings.json:** Tệp 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.
  + 1. 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,…

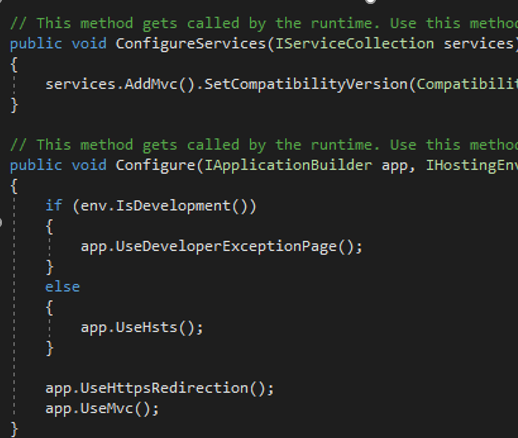
Với một ứng dụng Web ASP.NET, ta phải làm quen với khái niệm mới như **thẻ chú thích** nằm trước một class hoặc một function như dưới đây:



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

Trong các tệp Controller:

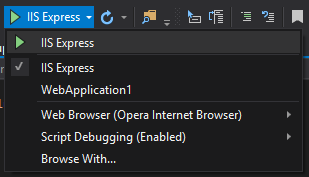
* Thẻ **[Route]**: Định dạng đường dẫn gốc nếu đặt cho một class khi nó lắng nghe yêu cầu.
* Thẻ **[ApiController]**: Cho biết một class thuộc Controller chuyên phục vụ cho việc lắng nghe và trả về các kết quả không phải là một View.
* Một lớp là Controller sẽ kế thừa ControllerBase để được xem là Controller.
* 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 dường 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 dường 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.



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

Trong tệp thiết lập **Startup.cs:**

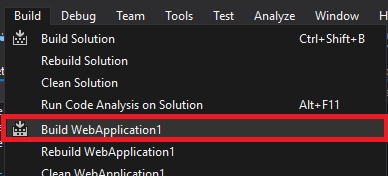
* **Services.AddMvc():** Thêm dịch vụ Mvc() và thiết lập, khi dịch vụ Mvc được bật, ASP.NET Core sẽ có thể phục vụ các gói tin trong thư mục wwwroot.
* **App.UseMvc():** Bật dịch vụ Mvc() đã được thiết lập phía trên.
* **App.UseDeveloperExceptionPage():** Bật môi trường Debug.
* **App.UseHsts():** Bật header bảo mật dùng trong IIS.
* **App.UseHttpsRedirection():** Bật Https cho trang.
  + 1. Khởi chạy



Hình 4. . 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.

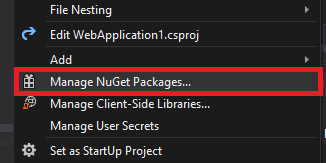
Ngoài ra, để Build một ứng dụng, người dùng có thể chọn Build ở trên menu của Visual Studio, sau đó vào thư mục bin để tìm các tệp dll liên quan được build.



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

* 1. Sử dụng NuGet Package

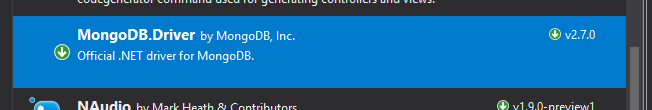
NuGet Package là các gói thư viện cho chương trình được lưu trên nuget, Visual Studio 2017 hỗ trợ việc quản lý các gói nuget này qua giao diện người dùng.



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

* 1. 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#.

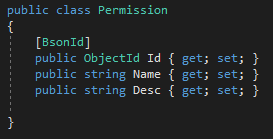


Hình 4. . MongoDB Driver trên NuGet Packages

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

* + 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#.

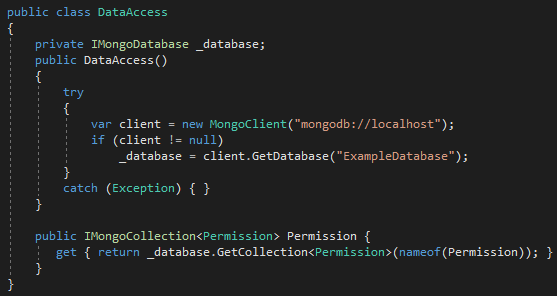


Hình 4. . Tạo một Model đơn giản nhằm lưu nó xuống Database sau này

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.
  + 1. 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.



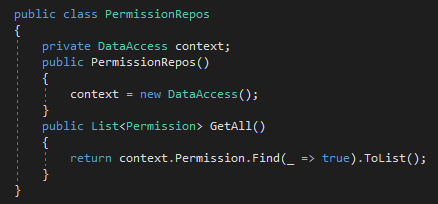
Hình 4. . Khởi tạo class DataAccess kết nối đến Mongo Database

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.

* + 1. Repository

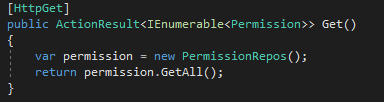
Repository là các class chứa các câu lệnh truy vấn cơ sở dữ liệu

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



Hình 4. . Một Repository cơ bản chứa DataAccess

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



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

* 1. Dependency Injection trong ASP.NET Core
     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 dễ 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.

* + 1. Khái niệm Dependency Injection

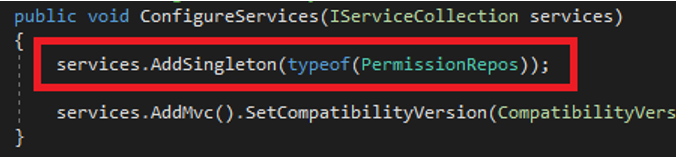
Một khái niệm thuộc *Inversion of Control*, thay vì một class nào đó, khai báo một đối tượng khác ngay trong cấu trúc và tạo mới đối tượng thì với Dependency Injection (DI), class này chỉ cần khai báo một loại *đa hình (Interface hoặc abstraction)* của đối tượng, sau đó truyền tham số vào khi khởi tạo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

* + 1. 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 chung ta cải thiện điều này qua Dependency Injection.

Vào **Startup.cs** và gọi lệnh sau:



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

Sau đó sửa lại Controller của chúng ta như sau:



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

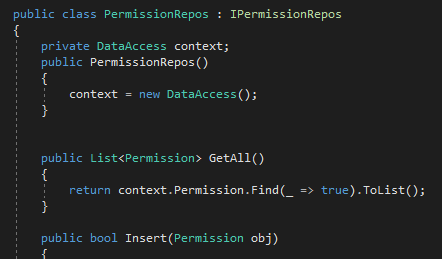
Lúc này, mặc dù permission không có bất kỳ lệnh tạo mới nào, thay vào đó thêm một tham số trong constructor, và gán tham số đó vào.

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:

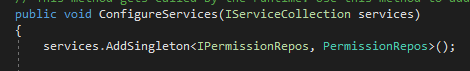
* **AddSingleton**: Hỗ trợ tạo đối tượng có thời gian sống dài, và chỉ tạo một đối tượng xuyên suốt thời gian chạy chương trình.
* **AddScoped**: Tạo đối tượng có thời gian sống trong một Request Session, mỗi Request Session của ứng dụng sẽ tạo một đối tượng mới.
* **AddTransient**: Tạo đối tượng mỗi lần được yêu cầu đến.

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



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

Thì ta có thể gọi lệnh như sau:



Hình 4. . 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

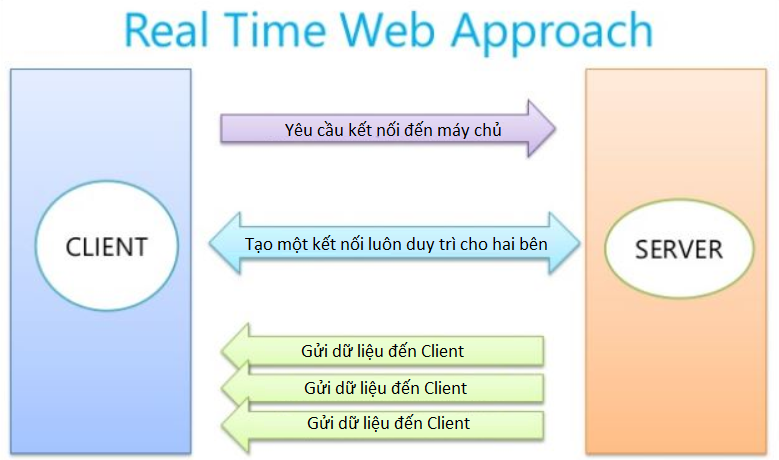
* + 1. SignalR là gì?



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

* + 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. . Giao tiếp thời gian thực giữa Client và Server (nguồn: c-sharpcorner.com)

* + 1. 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.

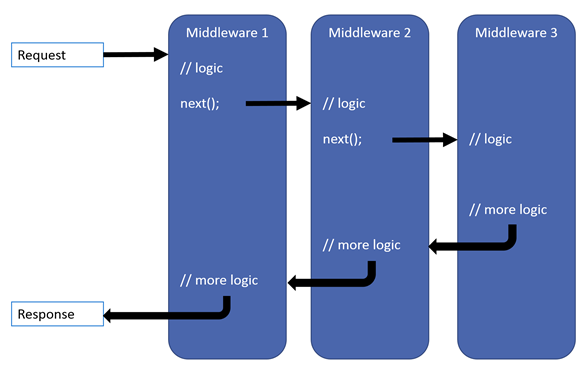
SignalR thường được sử dụng khi:

* 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.
  + 1. 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.

* + 1. Thuật ngữ Middleware

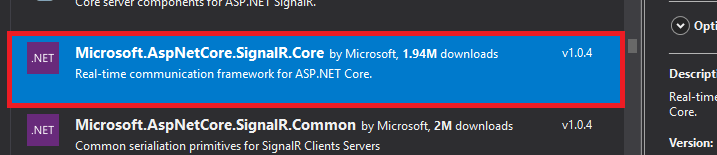
Middleware trong ASP.NET Core có thể hiểu là một thành phần ở giữa, phục vụ công tác lắng nghe các yêu cầu đến và chuyển hướng đến hành động thích hợp. Ví dụ: Người dùng yêu cầu đến đường dẫn */signalR* thì sẽ chuyển hướng sang hướng các dịch vụ của signalR, còn các đường dẫn khác không phải thì sẽ phục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



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

* 1. Cài đặt
     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. . SignalR cho ASP.NET Core trên NuGet packages

* + 1. Javascript Client

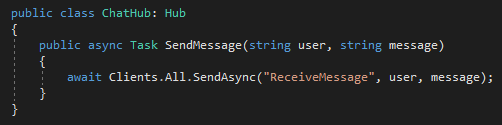
Thường sử dụng cho phía máy con, tức là trên phần mặt web giao tiếp với người dùng, SignalR trong Javascript sẽ thiết lập một kết nối đến phía máy chủ SignalR.

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

*npm install @aspnet/signalr*

* 1. Ứng dụng
     1. Thiết lập lắng nghe
        1. Tạo SignalR Hub

Tạo một class kế thừa Hub của SignalR như sau:

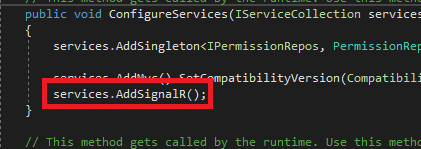


Hình 5. . Một SignalR Hub cơ bản

* + - 1. Thiết lập lắng nghe cho SignalR

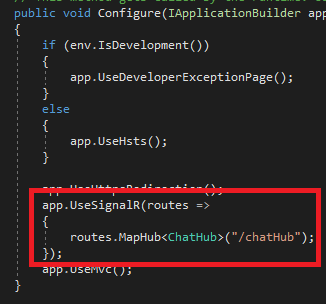
Để máy chủ có thể lắng nghe được máy con trên SignalR, ta mở tệp Startup.cs để tiến hành thiết lập lắng nghe.

Thêm lệnh sau vào hành vi thiết lập dịch vụ:



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

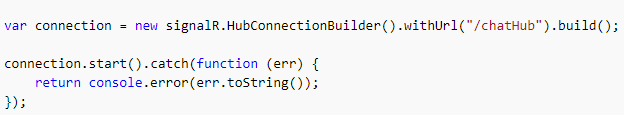
Sau đó sử dụng SignalR như một Middleware trong ứng dụng:



Hình 5. . 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.
  + 1. 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:



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

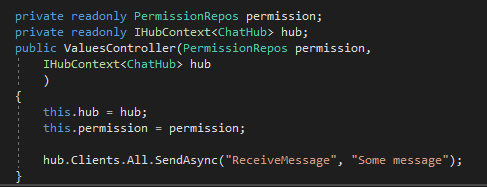
Kế tiếp, đặt một hàm lắng nghe sự kiện ở phía máy con:



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

* + 1. 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:



Hình 5. . 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:



Hình 5. . 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

* 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 sẽ, mọi người sẽ có thể tải lên bài nhạc mình muốn chia sẽ 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.

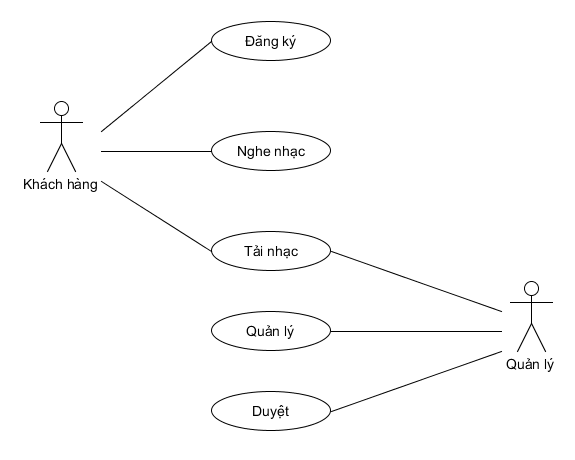
Bên cạnh đó, người nghe nhạc còn có thể tạo cho mình một danh sách phát riêng, có thể hoặc không cần lưu vào cơ sở dữ liệu, song mỗi danh sách phát đều chứa danh sách các bài hát họ muốn nghe. Nếu họ muốn lưu một danh sách phát, người dùng có thể đặt tên cho danh sách phát này, cấp quyền riêng tư hoặc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

Quản trị viên ngoài việc quản lý và duyệt các bài nhạc được chia sẽ 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.

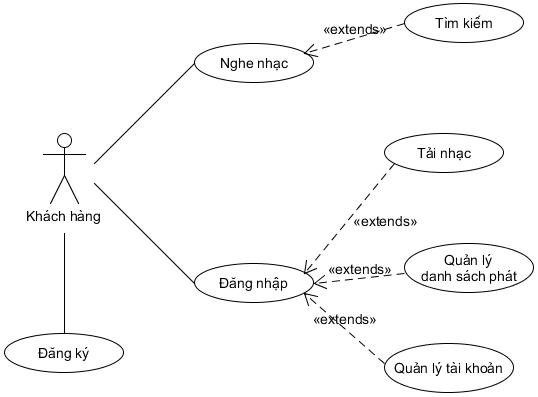
* 1. Thiết kế
     1. Mô hình Use-Case

**Mô hình ở mức cơ bản (mức 0):**



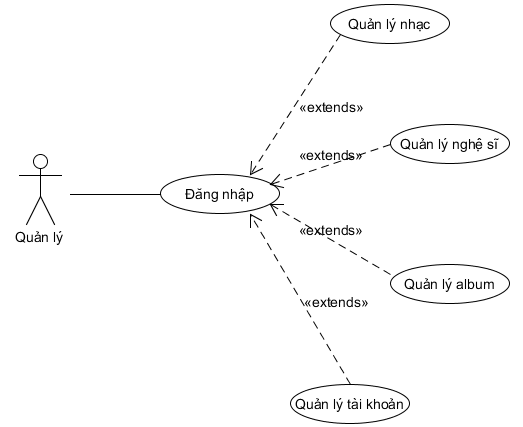
Hình 6. . Mô hình Use Case mức cơ bản

**Mô hình mức trung (mức 1):**

****

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

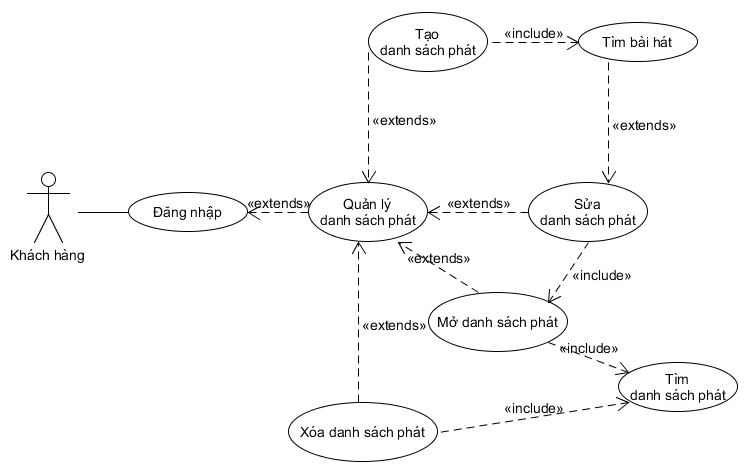
* **Trường hợp người sử dụng thường:** 
  + Khách hàng có thể vào đăng ký, đăng nhập, và nghe nhạc.
  + Khách hàng đã xác thực có thể tải nhạc, quản lý danh sách phát của họ và thay đổi thông tin tài khoản.



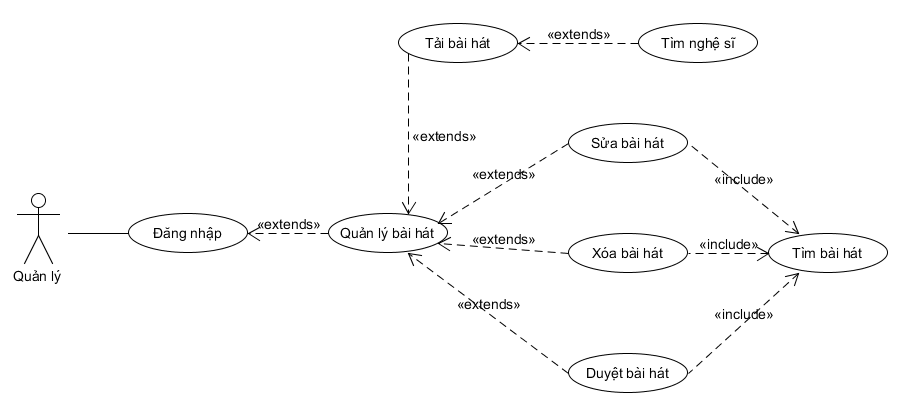
Hình 6. . Use case mức trung đối với quản trị viên

* **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. . Use case chi tiết Quản lý danh sách phát phía người dùng



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

* + 1. 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. . 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 |
| Permission | String (khóa ngoại) | Quyền hạn người dùng |

Bảng 6. . 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. . 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 |
| Nation | 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. . 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. . 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 |
| SongType | SongType[] (khóa ngoại) | Thể loại nhạc |

Bảng 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. . Thực thể đường dẫn nhạc

|  |  |  |
| --- | --- | --- |
| **Song** | | |
| **Bài hát** | | |
| **Tên cột** | **Kiểu dữ liệu** | **Ý nghĩa** |
| **Id** | string | Khóa chính |
| Name | string | Tên bài hát |
| Artists | String[] (khóa ngoại) | Danh sách nghệ sĩ biểu diễn, lưu bằng khóa chính thực thể Artist |
| SongType | String (khóa ngoại) | Thể loại của bài hát, lưu bằng khóa chính thực thể SongSubType |
| Uploader | string (khóa ngoại) | Người tải lên, lưu bằng khóa chính thực thể User |
| SongLinks | 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. . 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 |
| Artists | String[] (khóa ngoại) | Danh sách nghệ sĩ biểu diễn, lưu bằng khóa chính thực thể Artist |
| SongList | String[] (khóa ngoại) | Danh sách bài hát trong Album |

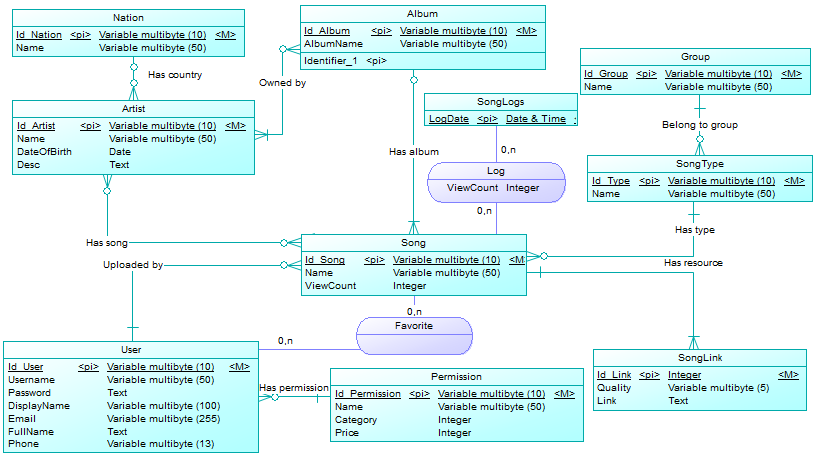
Bảng 6. . Thực thể lưu trữ Album

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

Bảng 6. . Thực thể lưu lịch sử lượt nghe

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

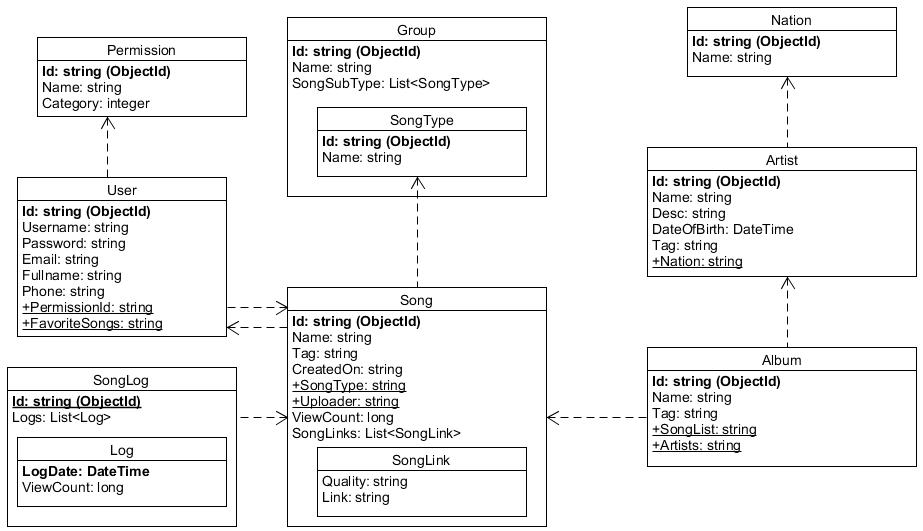
* + 1. Mô hình cơ sở dữ liệu quan hệ



Hình 6. . Mô hình dữ liệu vật lý

* + 1. 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:



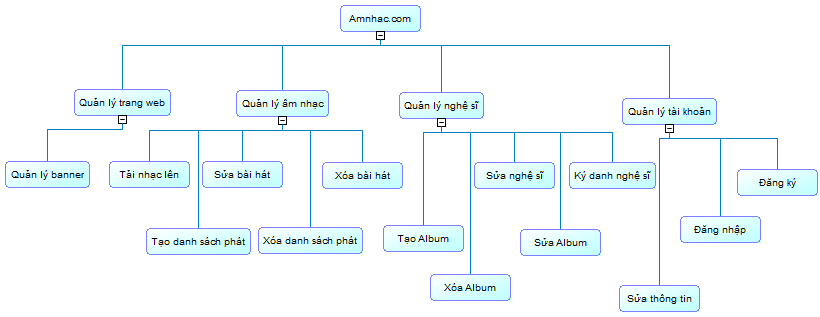
Hình 6. . Mô hình lưu trữ trong MongoDB

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.

* + 1. Sơ đồ phân rã chức năng

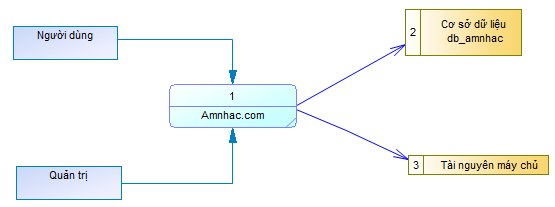
Amnhac.com sẽ có các chức năng sau nhìn về tổng quát:



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

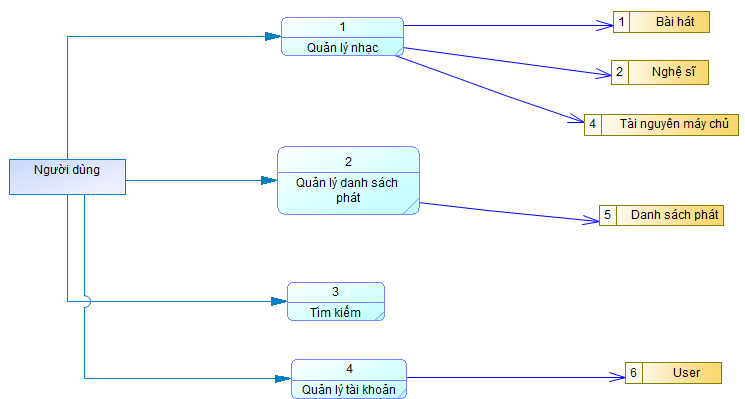
* + 1. Dòng dữ liệu

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



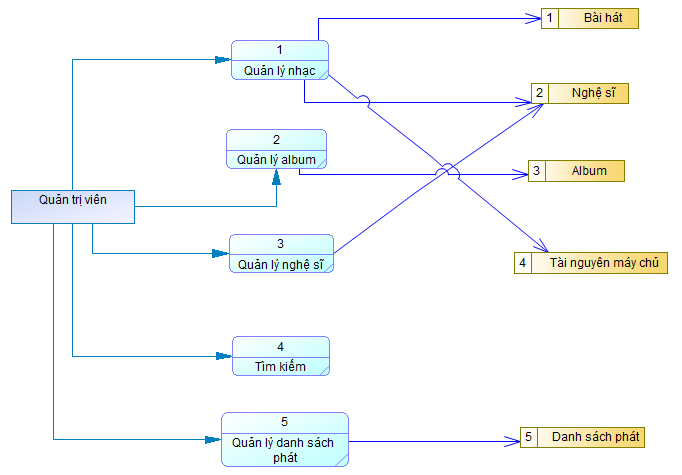
Hình 6. . Sơ đồ dòng dữ liệu tổng quan

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



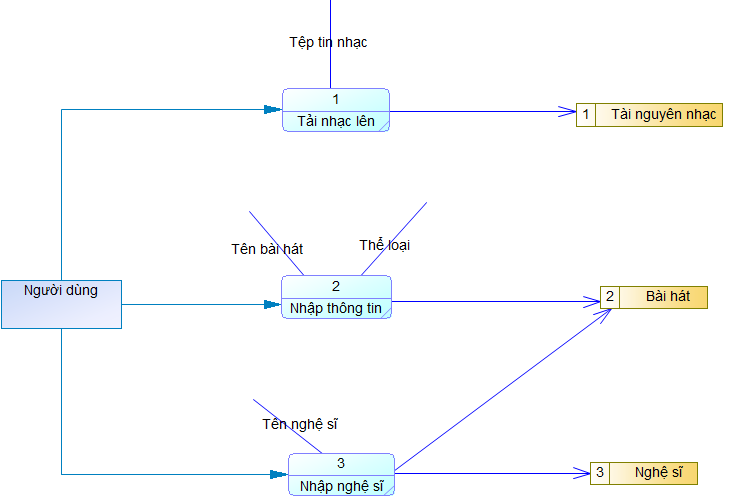
Hình 6. . Sơ đồ dòng dữ liệu mức người dùng

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

****

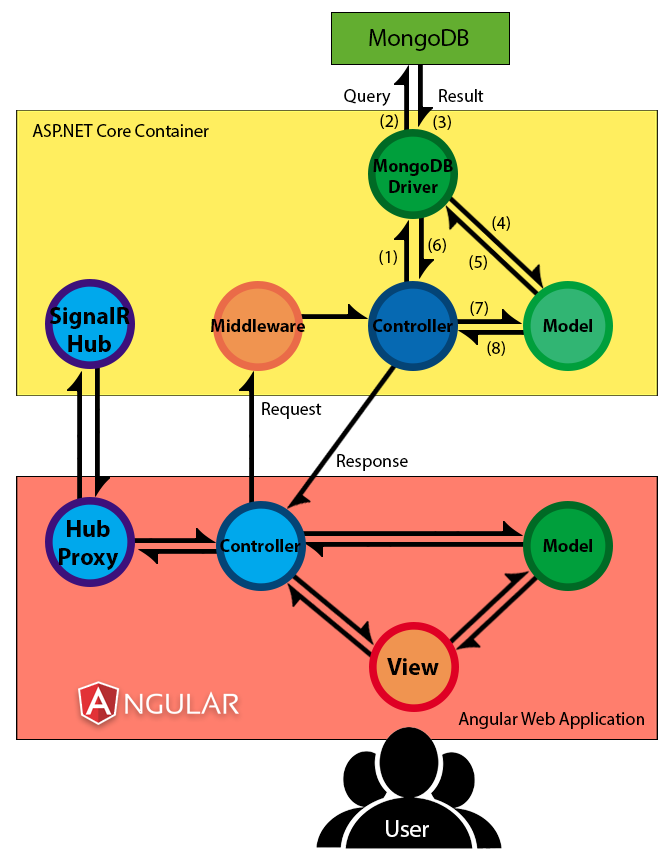
Hình 6. . Sơ đồ dòng dữ liệu ở quản trị viên

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



Hình 6. . Dòng dữ liệu khi tải một bài hát lên (mức 2)

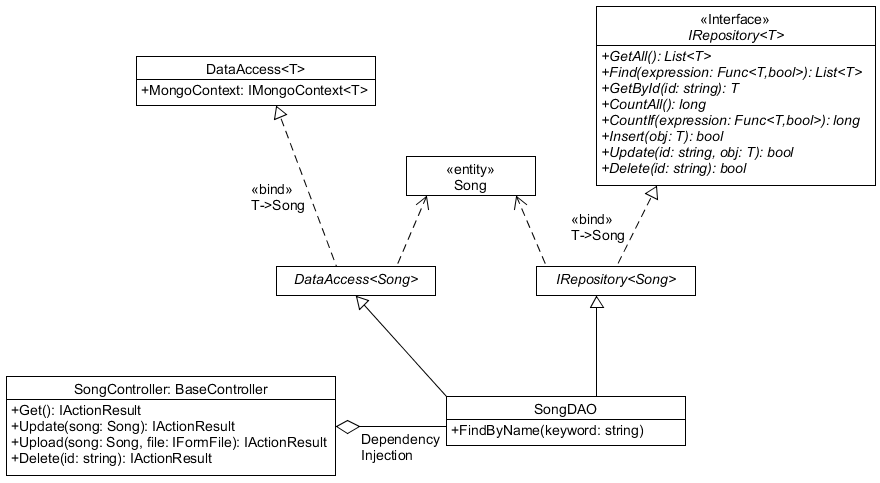
* 1. Thiết kế cơ bản
     1. Mô hình hoạt động



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

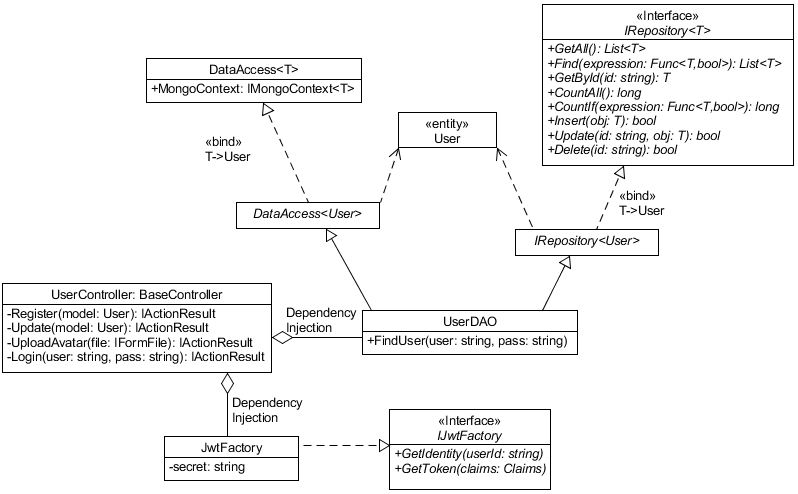
Chương trình sẽ được chia làm ba phần:

* **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
  + 1. Thiết kế mô hình lớp



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

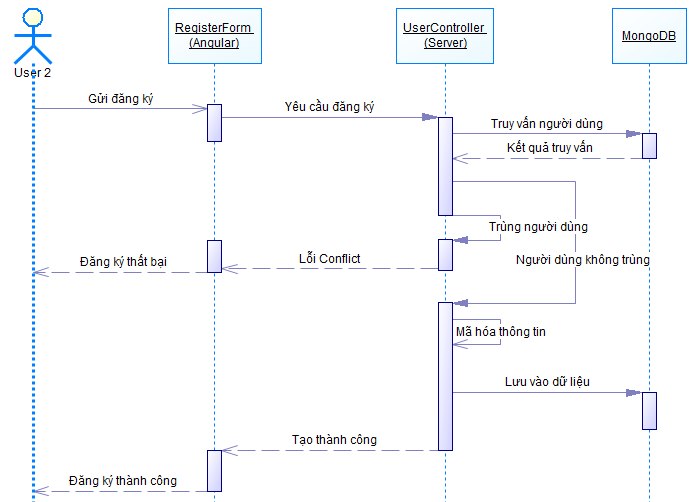
Trong mô hình lớp dành cho Server, một Controller sẽ truy vấn đến MongoDB qua lớp DataAccessObject. Hình trên đại diện cho một Controller trong Server ASP.NET Core, các Controller khác sẽ có phần tương tự.



Hình 6. . 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.

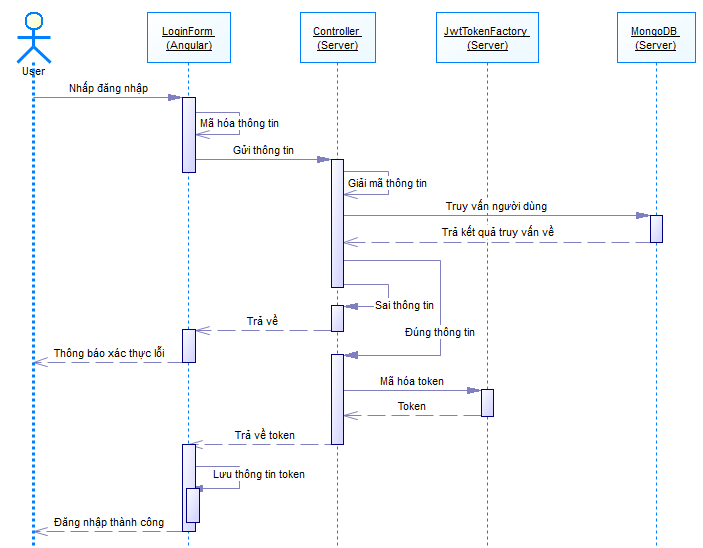
* 1. Thiết kế quy trình
     1. Chức năng đăng ký



Hình 6. . 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.
  + 1. Chức năng đăng nhập

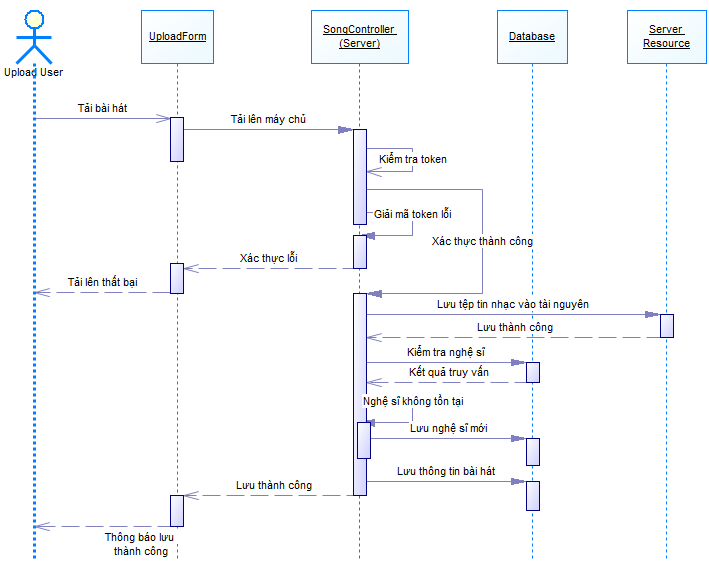


Hình 6. . Quy trình đăng nhập của người dùng

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.

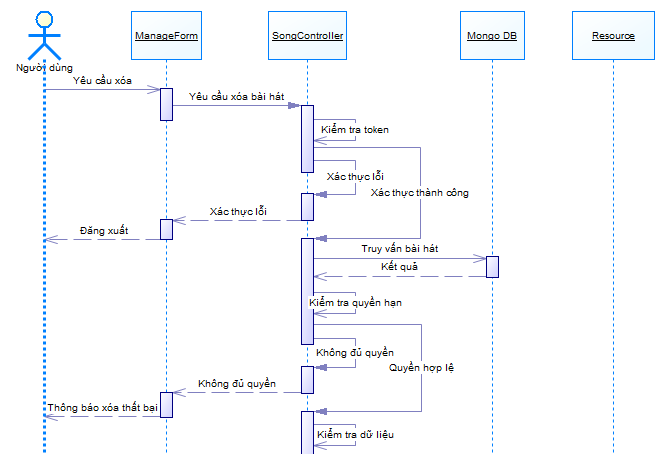
* Nếu truy vấn sai, Server sẽ gửi lỗi xác thực về.
* Nếu truy vấn đúng, Server sẽ mã hóa các thông tin cần thiết qua JwtFactory và trả về Token cho Client.
* Client sau khi nhận được token sẽ lưu Token nhằm dùng cho lần vào kế, đồng thời thông báo người dùng đăng nhập thành công.
  + 1. Chức năng tải nhạc



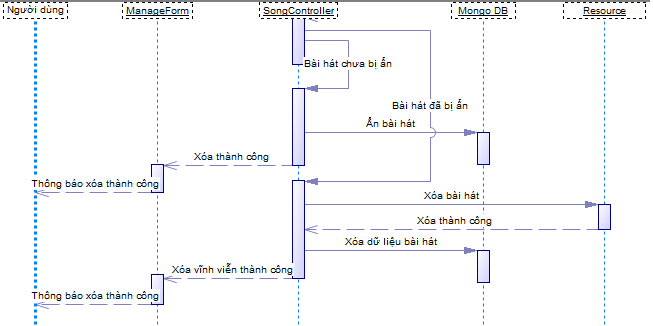
Hình 6. . Quy trình tải nhạc lên máy chủ

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 tiến 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.
  + 1. Chức năng xóa nhạc



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



Hình 6. . 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 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

* 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 sẽ được âm nhạc, giai điệu, và là nơi lắng nghe giai điệu của mọi người. Từ đà phát triển này, Amnhac.com có thể sẽ triển khai trên thị trường Việt Nam trong một ngày không xa.

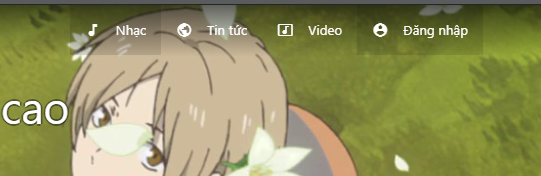
* 1. Các chức năng nổi bật
     1. Trang chủ

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

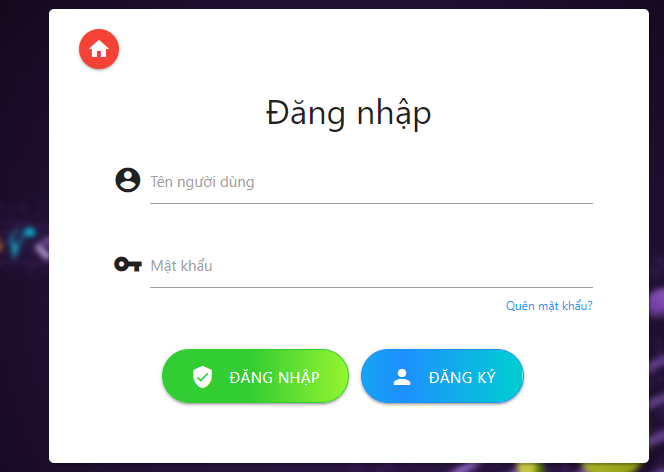


Hình 7. . Giao diện trang chủ Amnhac.com

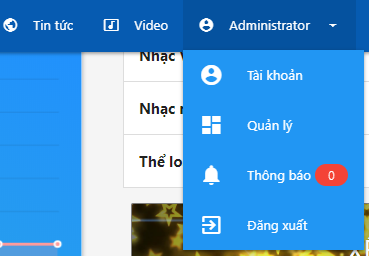
* + 1. Trang quản lý



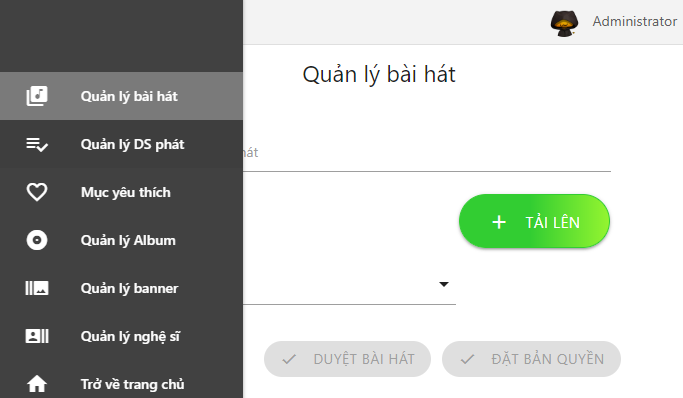
Hình 7. . Vào mục đăng nhập ở trên thanh tìm kiếm



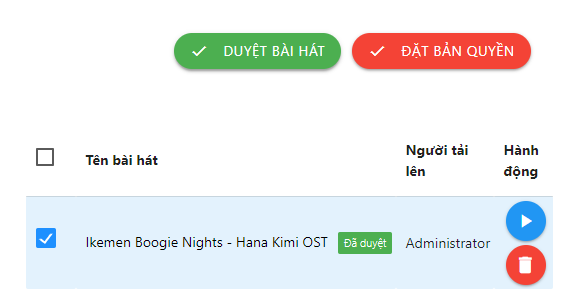
Hình 7. . Dùng tài khoản đăng nhập hoặc đăng ký mới



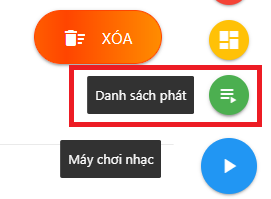
Hình 7. . Từ thanh tìm kiếm, vào mục quản lý



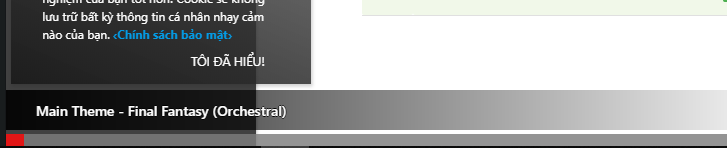
Hình 7. . Giao diện quản lý bài hát



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



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

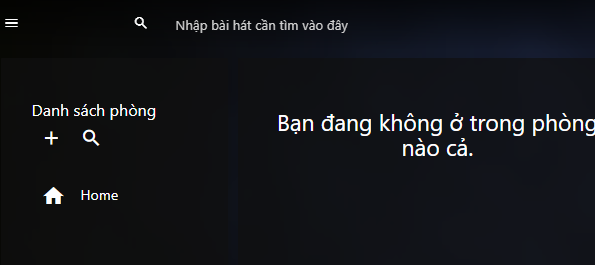


Hình 7. . Giao diện chơi nhạc khi phát một bài hát

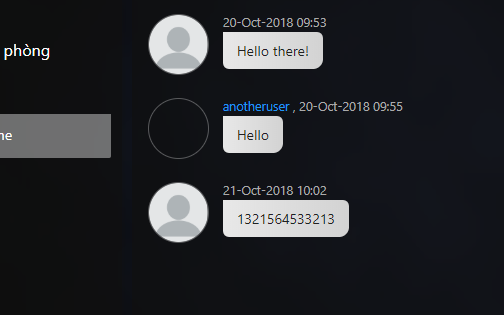
* + 1. Phòng trò chuyện



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



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



Hình 7. . Xem danh mục trò chuyện

* 1. Đánh giá
     1. Kết quả đạt được
* Ứng dụng 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.
  + 1. Yếu điểm
* 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.
  + 1. Hướng phát triển

Amnhac.com là dự án sẽ được phát hành tại thị trường Việt Nam trong tương lại, 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

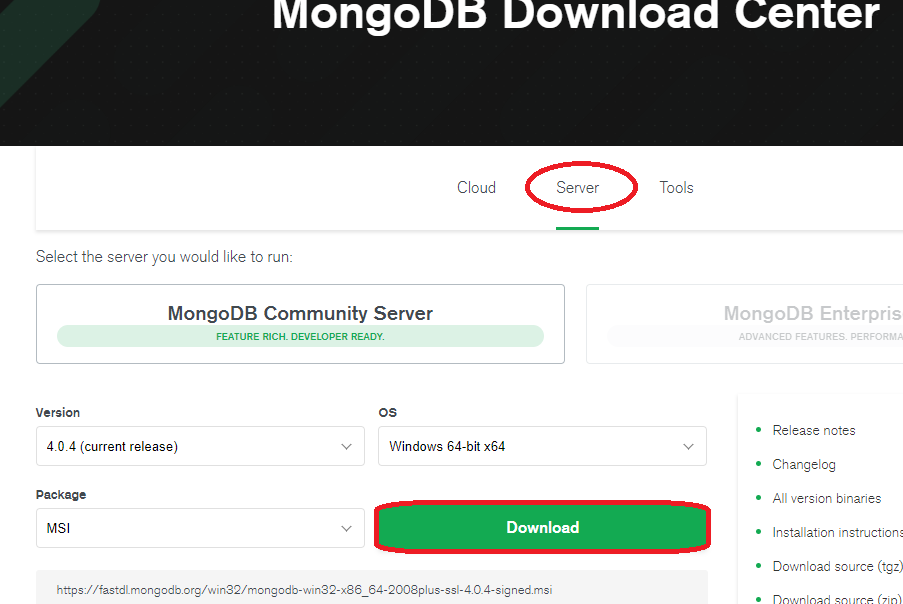
**Giới thiệu tính năng Amnhac.com và hướng dẫn thực hiện**

1. Cài đặt và triển khai
   1. 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. Cài đặt hệ quản trị cơ sở dữ liệu

Sau khi tải bộ cài từ trang chủ 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 . Tải về MongoDB tại mongodb.com

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. Cài đặt môi trường Angular và NodeJS

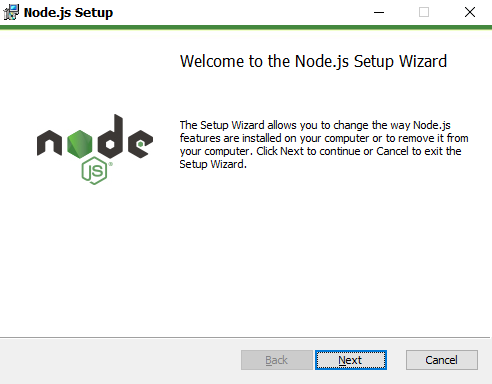
Để tiến hành sử dụng ứng dụng Angular, người dùng cần phải trải qua 2 bước cài đặt như sau:

* + 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 . Giao diện cài đặt NodeJS

**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*



Hình . Kiểm tra phiên bản NodeJS

* + 1. **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.

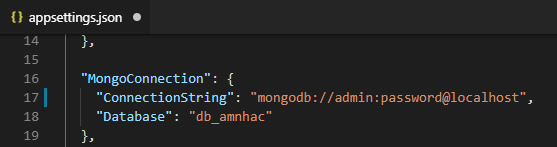
* 1. Cài đặt Runtime cho ASP.NET Core:

Để chạy được ứng dụng .NET Core, ta phải cài đặt bộ Runtime tương ứng. 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 Runtime mới nhất tại đây: <https://www.microsoft.com/net/download>

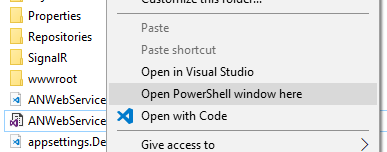
* 1. Triển khai máy chủ dịch vụ

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



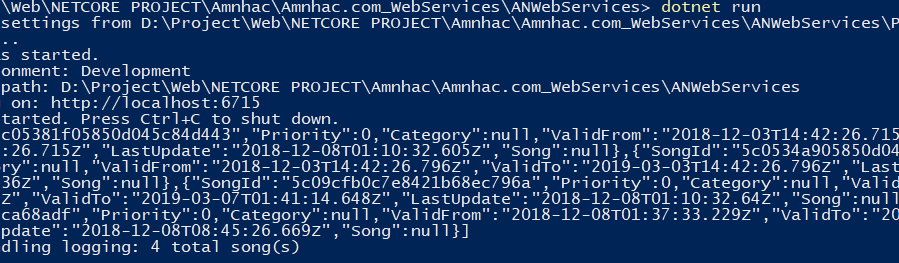
Hình . Thiết lập appsettings.json phía ASP.NET Core server

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 . Mở thao mục PowerShell

Sau đó gõ lệnh *dotnet run* để tiến hành host một máy chủ dịch vụ



Hình . Chạy thử máy chủ

* 1. Triển khai máy chủ giao tiếp

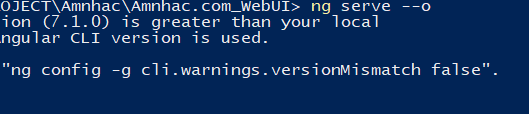
Là giao diện chính, để triển khai máy chủ giao tiếp, ta vào thư mục dự án chứa Web UI và mở Command Prompt/Windows Powershell/Terminal tại đây.

Gõ npm install để cài đặt các thành phần còn thiếu



Hình . Cài đặt các thành phần còn thiếu bằng lệnh npm install

Sau đó, gõ *ng serve --o*



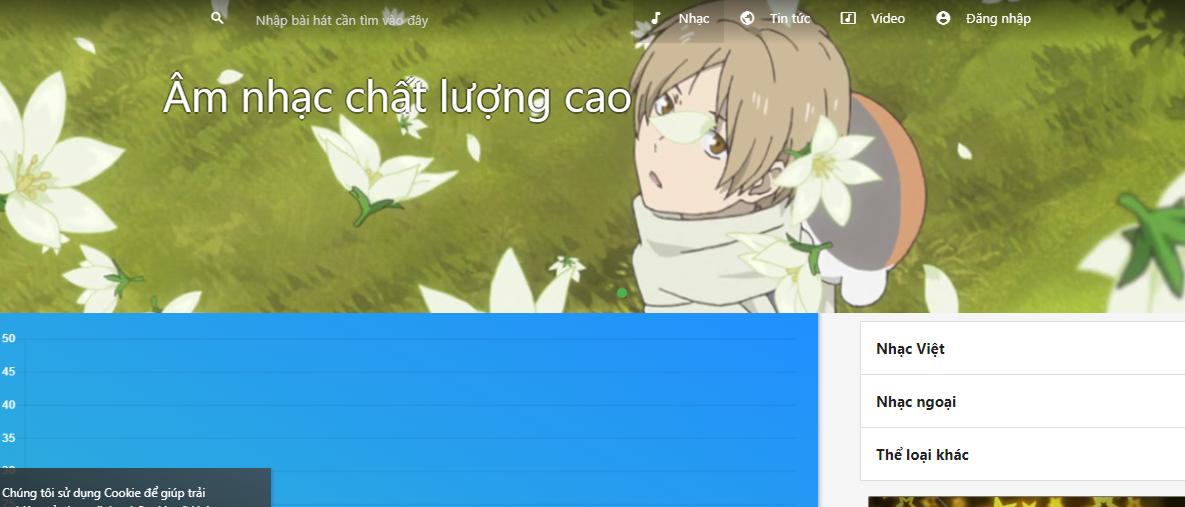
Hình . Chạy thử máy chủ development của Angular

Sau đó người dùng có thể vào *localhost:4200* để xem ứng dụng web.

* 1. Khởi tạo bộ dữ liệu mẫu

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:



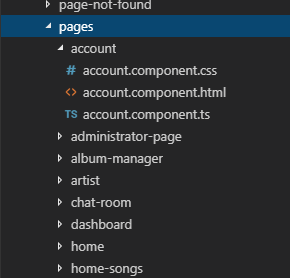
Hình . Giao diện trang chủ

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.

1. Kỹ thuật lập trình
   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

Sau đó tạo nội dung cho các trang.

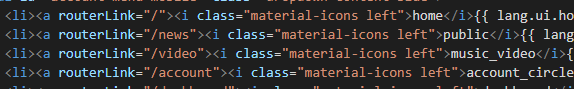
Áp dụng Router trong Angular để làm một ứng dụng đơn trang như sau:



Hình

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.**

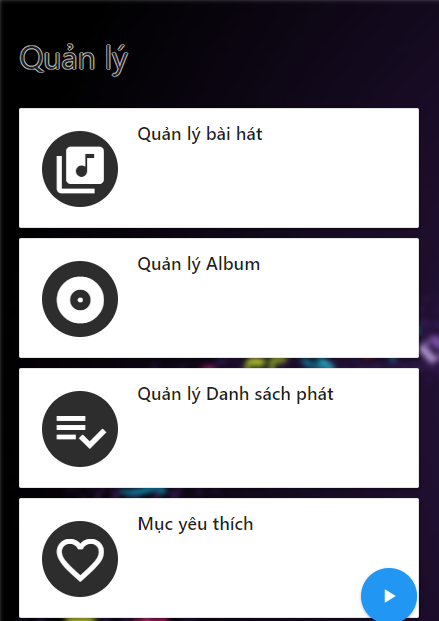
Đặt thuộc tính routerLink để dùng RouterModule điều hướng thay vì dùng href của HTML:



Hình

* 1. Dịch vụ web hướng đa nền tảng

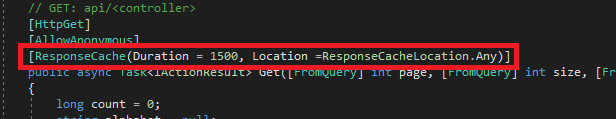
Dưới dạng dịch vụ Web, máy chủ web ASP.NET Core có thể lắng nghe không chỉ các yêu cầu từ Ajax Website mà còn các yêu cầu thuộc giao thức HTTP.



Hình

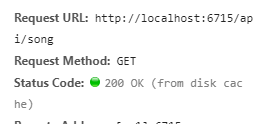
* 1. Tối ưu bằng Caching

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.



Hình

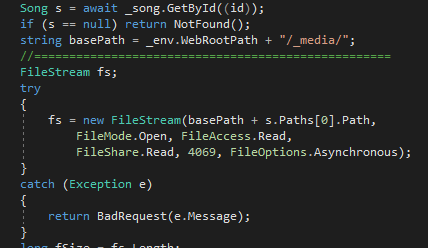
Khi người dùng yêu cầu một kết quả đã xử lý trước đó lần nữa:



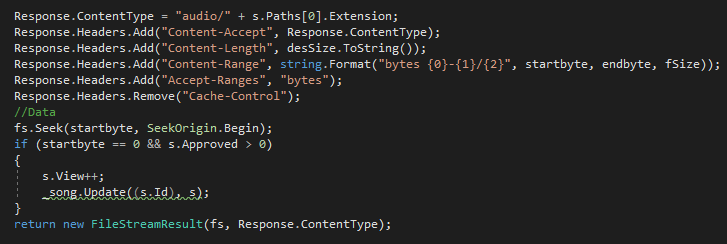
Hình

* 1. 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.



Hình



Hình

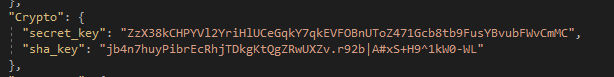
Trong phần trả về Header trên có các thuộc tính sau:

* **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.
  1. 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:

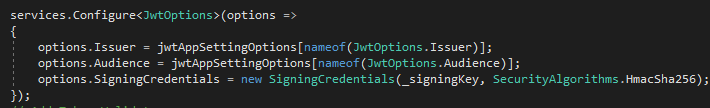


Hình

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



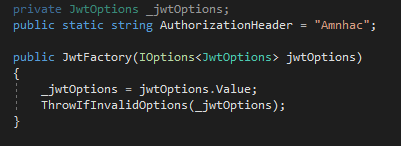
Hình



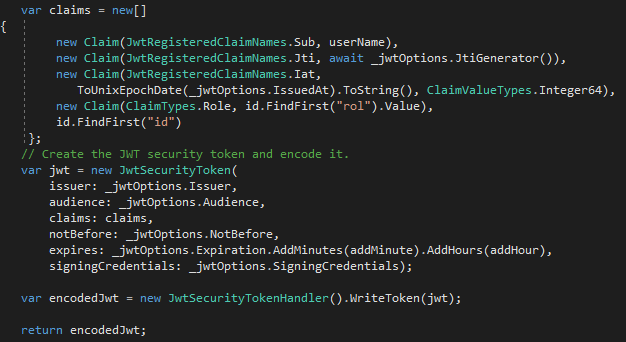
Hình



Hình . Khởi tạo thiết lập cho Jwt Bearer Factory



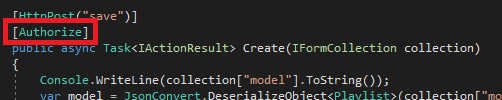
Hình . Inject các option vào Factory



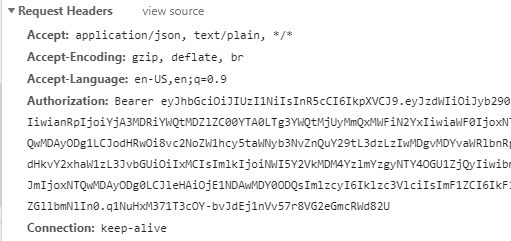
Hình . Tạo các Claim và nhờ Jwt mã hóa chúng



Hình . Token được mã hóa gửi về Client



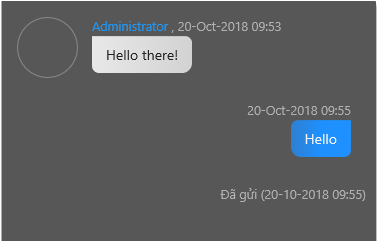
Hình . Đặt thẻ chú thích Authorize để yêu cầu gửi kèm Token mỗi yêu cầu



Hình . Token được gửi kèm trong Header để xác thực

* 1. Giao tiếp thời gian thực an toàn

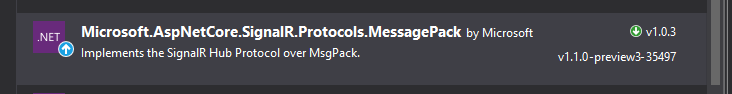
Với SignalR, quá dễ dàng để tạo lập một phòng trò chuyện nho nhỏ:



Hình

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

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



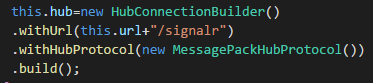
Hình

**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:



Hình

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

1. Werner Vogels - allthingsdistributed.com: <https://www.allthingsdistributed.com/2012/01/amazon-dynamodb.html>
2. W3school.com
3. MongoDB.com: <https://mongodb.com>
4. Angular.io – Google: <https://angular.io/>
   1. “Tutorial - Modules”: <https://angular.io/guide/architecture-modules>
   2. “Tutorial – Services”: <https://angular.io/tutorial/toh-pt4>
   3. “Tutorial – Routing”: <https://angular.io/tutorial/toh-pt5>
   4. “Deployment”: <https://angular.io/guide/deployment>
5. Tanya Gray – Medium Blog, “Angular Route Transition Animation”: <https://medium.com/@tanya/angular4-animated-route-transitions-b5b9667cd67c>
6. Bách khoa toàn thư mở:
   1. “Inversion of Control”: <https://en.wikipedia.org/wiki/Inversion_of_control>
   2. “Dependency Injection”: <https://en.wikipedia.org/wiki/Dependency_injection>
   3. “Định nghĩa JSON Web Token”: <https://en.wikipedia.org/wiki/JSON_Web_Token>
7. Petru Faurescu on Quality App Design – “Using MongoDB with .NET Core Web API”: <https://www.qappdesign.com/using-mongodb-with-net-core-webapi/>
8. 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>
9. 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>
10. 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>
11. 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>
12. 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>
13. 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>
14. 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>