PHỤ LỤC

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

1. Cài đặt và triển khai

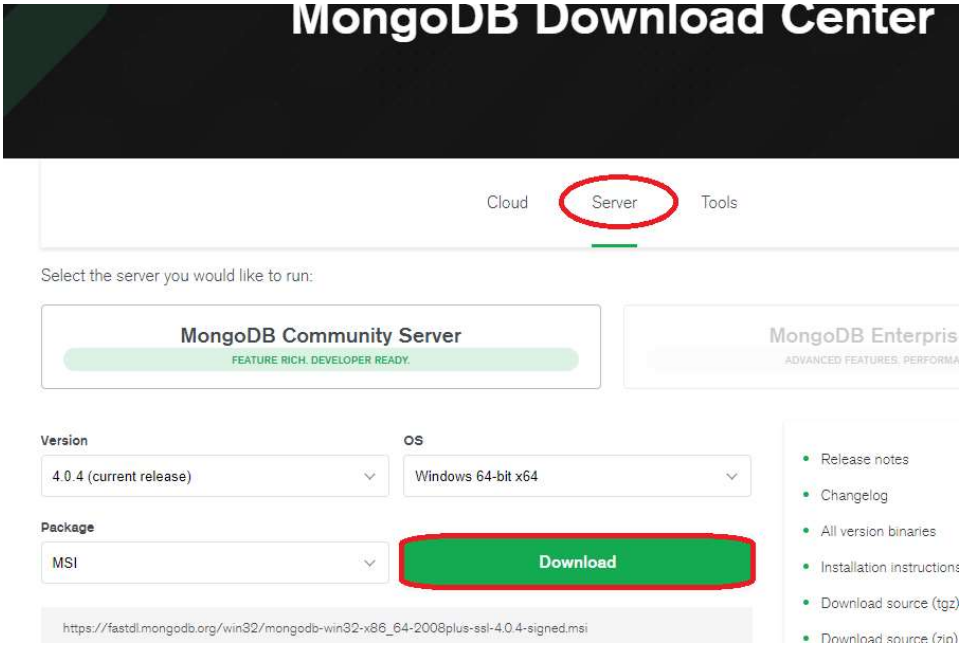
1.1. Cài đặt môi trường

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

1.2. Cài đặt hệ quản trị cơ sở dữ liệu

Sau khi 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 1. 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.

1.3. Cài đặt môi trường Angular và NodeJS

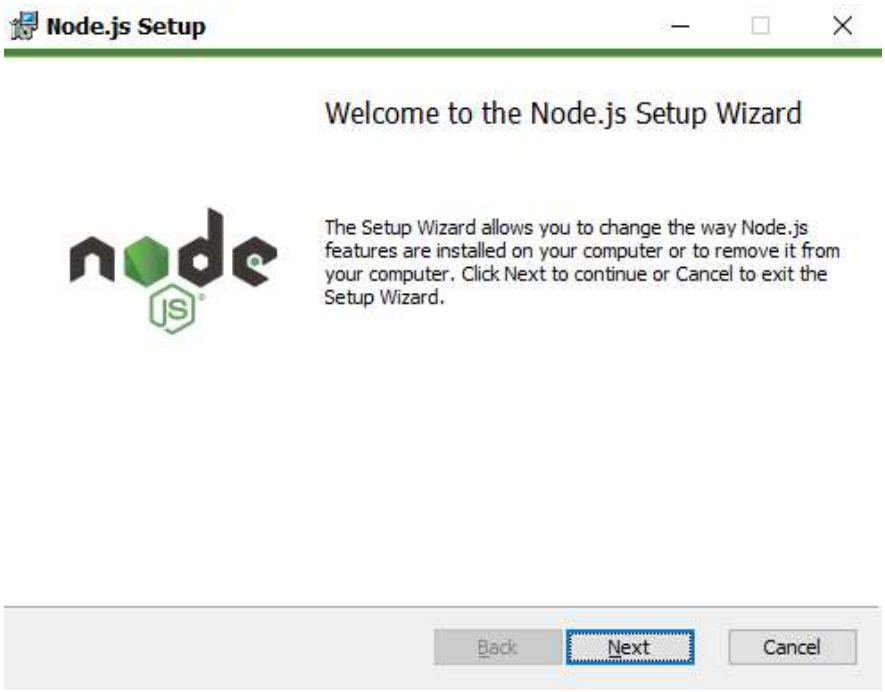
Để tiến hành sử dụng ứng dụng Angular, người dùng cần phải trải qua 2 bước cài đặt như sau:

1.3.1. 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 2. 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 3. Kiểm tra phiên bản NodeJS

1.3.2. Cài đặt Angular CLI

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

npm install –global @angular/cli

Lệnh trên sẽ cài đặt Angular CLI vào máy tính.

1.4. 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.

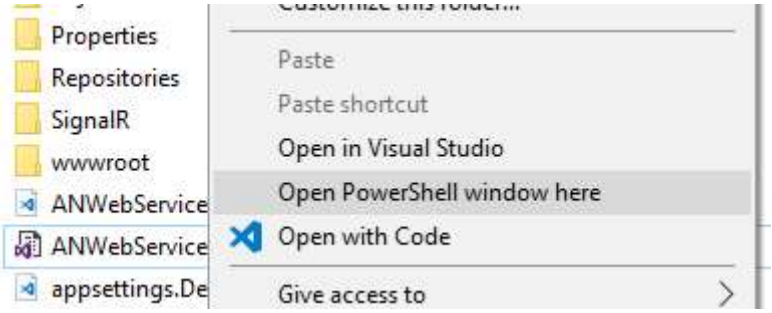
Người dùng có thể tải về Runtime mới nhất tại đây:
https://www.microsoft.com/net/download

1.5. 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*

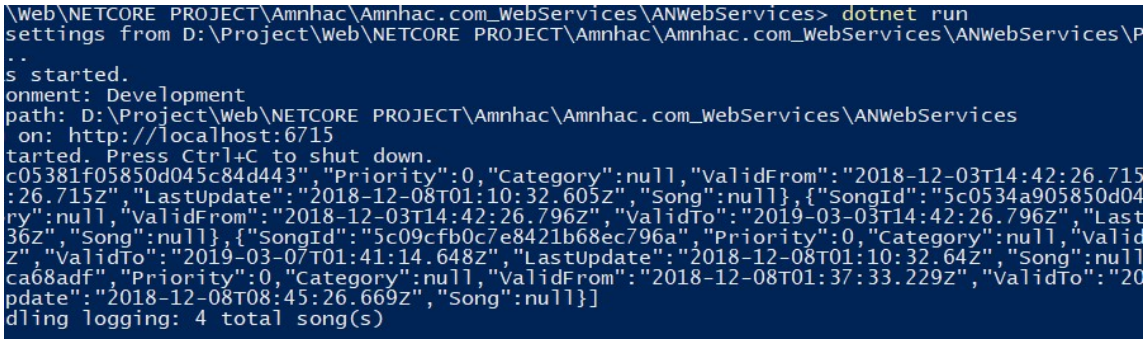
Hình 4. Thiết lập appsettings.json phía ASP.NET Core server

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 5. Mở thao mục PowerShell

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

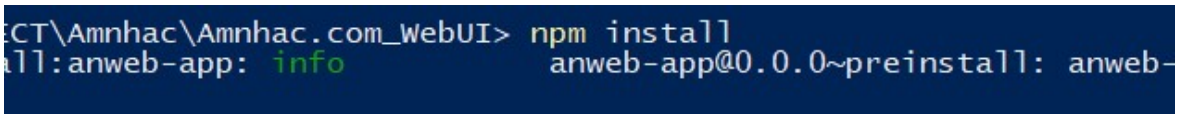


Hình 6. Chạy thử máy chủ

1.6. Triển khai máy chủ giao tiếp

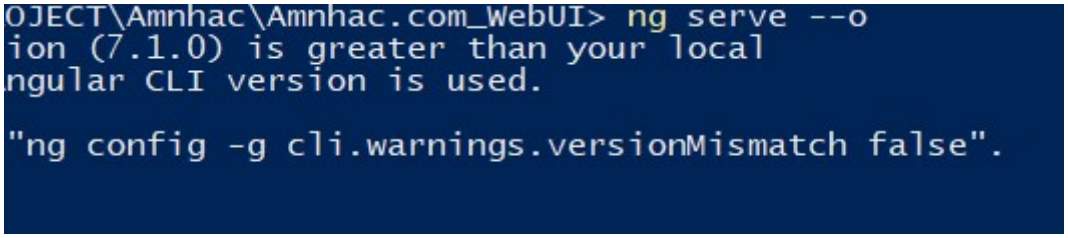
Là giao diện chính, để triển khai máy chủ giao tiếp, ta vào thư mục dự án chứa Web UI và mở Command Prompt/Windows Powershell/Terminal tại đây.

Gõ npm install để cài đặt các thành phần còn thiếu



Hình 7. Cài đặt các thành phần còn thiếu bằng lệnh npm install

Sau đó, gõ *ng serve --o*



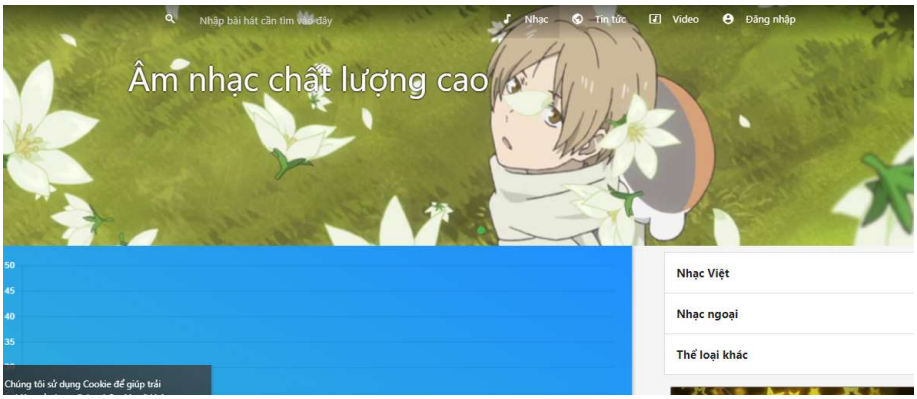
Hình 8. Chạy thử máy chủ development của Angular

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

1.7. 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 9. Giao diện trang chủ

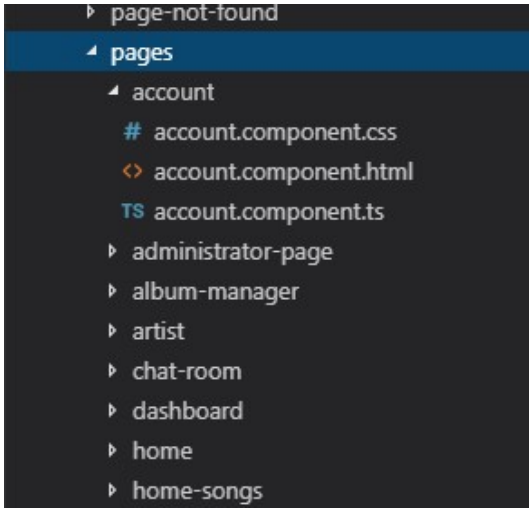
Sau đó điều hướng đến **/system/initializer**, nếu trang Web chưa được thiết lập đúng, lúc này hệ thống sẽ tự động thiết lập mọi thứ cho người dùng, tạo sẵn cơ sở dữ liệu mẫu và cấp cho họ một tài khoản quản trị viên.

2. Kỹ thuật lập trình

2.1. Ứng dụng web đơn trang

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

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



Hình 10

Sau đó tạo nội dung cho các trang.

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

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

Hình 11

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

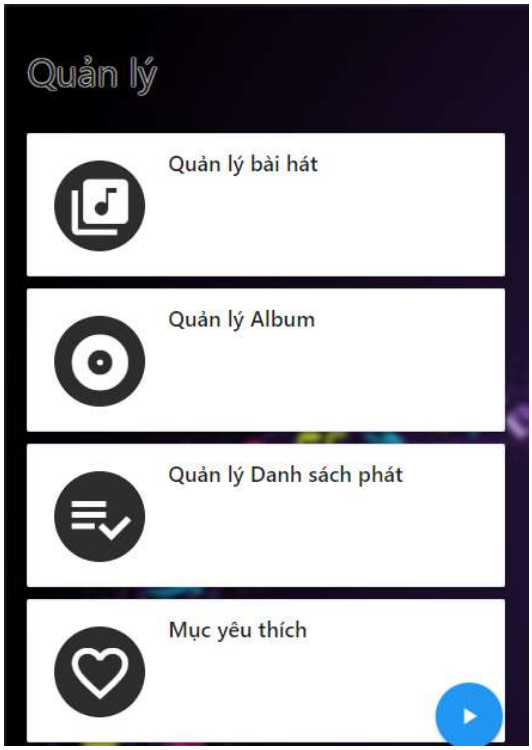
Đặt thuộc tính routerLink để dùng RouterModule điều hướng thay vì dùng href của HTML:

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

Hình 12

2.2. Dịch vụ web hướng đa nền tảng

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



Hình 13

2.3. Tối ưu bằng Caching

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

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

Hình 14

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

Hình 15

2.4. Phát một phần tập tin qua luồng

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

```
Song s = await _song.GetById((id));
if (s == null) return NotFound();
string basePath = _env.WebRootPath + "/_media/";
//================================================
FileStream fs;
try
{
    fs = new FileStream(basePath + s.Paths[0].Path,
        FileMode.Open, FileAccess.Read,
        FileShare.Read, 4069, FileOptions.Asynchronous);
}
catch (Exception e)
{
    return BadRequest(e.Message);
}
```

Hình 16

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

Hình 17

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

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

2.5. Bảo mật với Xác thực JWT

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

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

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

```
},
"Crypto": {
  "secret_key": "ZzX38kCHPYVl2YriHlUCeGqkY7qkEVFOBnUToZ471Gcb8tb9FusYBvubFWvCmMC",
  "sha_key": "jb4n7huyPibrEcRhjTDkgKtQgZRwUXZv.r92b|A#xS+H9^1kW0-WL"
},
```

Hình 18

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

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

Hình 19

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

Hình 20

```
// Add Token Validator
var tokenValidationParameters = new TokenValidationParameters
{
    ValidateIssuer = true,
    ValidIssuer = jwtAppSettingOptions[nameof(JwtOptions.Issuer)],

    ValidateAudience = true,
    ValidAudience = jwtAppSettingOptions[nameof(JwtOptions.Audience)

    ValidateIssuerSigningKey = true,
    IssuerSigningKey = _signingKey,

    RequireExpirationTime = true,
    ValidateLifetime = true
};
services.AddAuthentication(options => {
    options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(o =>
{
    o.TokenValidationParameters = tokenValidationParameters;
    o.IncludeErrorDetails = true;

});
```

Hình 21. Khởi tạo thiết lập cho Jwt Bearer Factory

```
private JwtOptions _jwtOptions;
public static string AuthorizationHeader = "Amnhac";

public JwtFactory(IOptions<JwtOptions> jwtOptions)
{
    _jwtOptions = jwtOptions.Value;
    ThrowIfInvalidOptions(_jwtOptions);
}
```

Hình 22. Inject các option vào Factory

```
var claims = new[]
{
    new Claim(JwtRegisteredClaimNames.Sub, userName),
    new Claim(JwtRegisteredClaimNames.Jti, await _jwtOptions.JtiGenerator()),
    new Claim(JwtRegisteredClaimNames.Iat,
       ToUnixEpochDate(_jwtOptions.IssuedAt).ToString(), ClaimValueTypes.Integer64),
    new Claim(ClaimTypes.Role, id.FindFirst("rol").Value),
    id.FindFirst("id")
};
// Create the JWT security token and encode it.
var jwt = new JwtSecurityToken(
    issuer: _jwtOptions.Issuer,
    audience: _jwtOptions.Audience,
    claims: claims,
    notBefore: _jwtOptions.NotBefore,
    expires: _jwtOptions.Expiration.AddMinutes(addMinute).AddHours(addHour),
    signingCredentials: _jwtOptions.SigningCredentials);

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

return encodedJwt;
```

Hình 23. Tạo các Claim và nhờ Jwt mã hóa chúng

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

Hình 24. Token được mã hóa gửi về Client

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

Hình 25. Đặt thẻ chú thích Authorize để yêu cầu gửi kèm Token mỗi yêu cầu

▼ **Request Headers** view source

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

Accept-Encoding: gzip, deflate, br

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

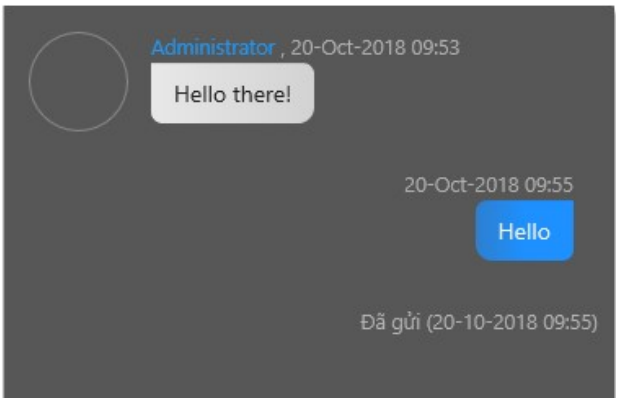
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJyb290
IiwianRpIjoiYjA3MDRiYWQtMDZlZC00YTA0LTg3YWQtMjUyMmQxMWFiN2YxIiwiaWF0IjoxN
QwMDAyODg1LCJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnR
dHkvY2xhaW1zL3JvbGUiOiIxMCIsImlkIjoiNWI5Y2VkMDM4YzlmYzgyNTY4OGU1ZjQyIiwib
JmIjoxNTQwMDAyODg0LCJleHAiOjE1NDAwMDY0ODQsImlzcyI6Iklzc3VlciIsImF1ZCI6IkF
ZGllbmNlIn0.q1NuHxM371T3cOY-bvJdEj1nVv57r8VG2eGmcRWd82U

Connection: keep-alive

Hình 26. Token được gửi kèm trong Header để xác thực

2.6. Giao tiếp thời gian thực an toàn

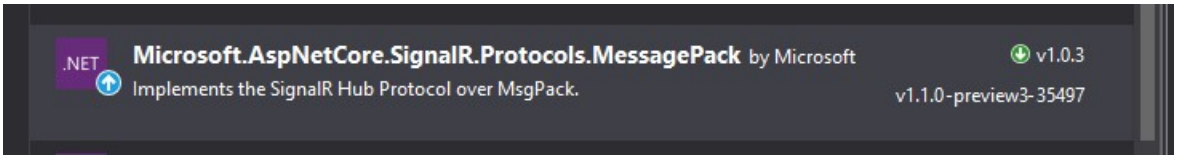
Với SignalR, quá dễ dàng để tạo lập một phòng trò chuyện nho nhỏ:



Hình 27

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 28

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

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

Hình 29

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 30

TÀI LIỆU THAM KHẢO

Tiếng Việt

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

Tiếng Anh

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

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

DẪN NGUỒN

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