Dưới đây là hướng dẫn tổng hợp cách tạo dự án có Server (.NET 9.0.2) và Client (Angular 19.2.0) với tích hợp Tailwind, Bootstrap và custom fonts. Các câu lệnh được đánh dấu (đặt trong khối mã) để khi copy vào Word sẽ dễ nhìn hơn.

**1. Tạo dự án Server (.NET 9.0.2)**

**Lưu ý:** Mặc dù hiện tại .NET 9 có thể chưa ra mắt, bạn có thể thay đổi phiên bản theo nhu cầu. Giả sử bạn đã cài SDK phù hợp.

**1.1. Tạo Web API mới**

Mở terminal (hoặc Command Prompt) và chạy:

dotnet new webapi -n MyProject.Server --framework net9.0

**1.2. Chạy thử dự án**

Di chuyển vào thư mục dự án và chạy:

cd MyProject.Server

dotnet run

Nếu ứng dụng chạy thành công, bạn sẽ thấy thông báo lắng nghe trên một cổng (ví dụ: [http://localhost:5000](http://localhost:5000/)).

**2. Tạo dự án Client (Angular 19.2.0)**

**2.1. Cài đặt Angular CLI phiên bản 19.2.0**

Nếu bạn muốn sử dụng phiên bản cụ thể của Angular CLI, chạy:

npm install -g @angular/cli@19.2.0

**2.2. Tạo dự án Angular mới**

Chạy lệnh tạo dự án với routing và SCSS (hoặc CSS) tùy theo nhu cầu:

ng new MyProject.Client --routing --style=scss

Di chuyển vào thư mục dự án:

cd MyProject.Client

**3. Tích hợp Tailwind CSS, Bootstrap và Custom Fonts cho Client**

**3.1. Cài đặt Tailwind CSS**

Trong thư mục dự án Angular, chạy:

npm install tailwindcss@latest postcss@latest autoprefixer@latest

npx tailwindcss init

Sau đó, mở file **tailwind.config.js** và thiết lập cấu hình (ví dụ):

module.exports = {

content: [

"./src/\*\*/\*.{html,ts}",

],

theme: {

extend: {},

},

plugins: [],

}

Mở file **src/styles.scss** và thêm các chỉ thị của Tailwind:

@tailwind base;

@tailwind components;

@tailwind utilities;

**3.2. Cài đặt Bootstrap**

Chạy lệnh:

npm install bootstrap

Sau đó, mở file **angular.json** và thêm đường dẫn đến Bootstrap vào mảng **styles**:

"styles": [

"node\_modules/bootstrap/dist/css/bootstrap.min.css",

"src/styles.scss"

],

**3.3. Tích hợp Custom Fonts**

Giả sử bạn có file font (ví dụ: my-font.ttf) hoặc sử dụng Google Fonts.

* **Nếu dùng file font:**
  + Đặt file font vào thư mục **src/assets/fonts/**.
  + Trong **src/styles.scss** thêm đoạn sau:
  + @font-face {
  + font-family: 'MyCustomFont';
  + src: url('/assets/fonts/my-font.ttf') format('truetype');
  + font-weight: normal;
  + font-style: normal;
  + }
  + body {
  + font-family: 'MyCustomFont', sans-serif;
  + }
* **Nếu dùng Google Fonts:**
  + Thêm link trong **src/index.html** trong phần <head>:
  + <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500;700&display=swap" rel="stylesheet">
  + Và cập nhật **src/styles.scss**:
  + body {
  + font-family: 'Roboto', sans-serif;
  + }

**4. Cấu trúc thư mục tiêu chuẩn**

**4.1. Cấu trúc dự án Server (.NET)**

MyProject.Server/

├── Controllers/ // Các controller Web API

│ └── WeatherForecastController.cs

├── Properties/

├── appsettings.json // Cấu hình, bao gồm ConnectionStrings, Logging, …

├── Program.cs // Entry point của ứng dụng

├── MyProject.Server.csproj // File dự án

└── ...

**4.2. Cấu trúc dự án Client (Angular)**

MyProject.Client/

├── e2e/ // End-to-end tests

├── node\_modules/ // Các package npm

├── src/

│ ├── app/

│ │ ├── components/ // Các component dùng chung (Header, Footer, …)

│ │ ├── pages/ // Các trang của ứng dụng (Home, About, …)

│ │ ├── services/ // Giao tiếp với API, xử lý logic

│ │ ├── models/ // Các model (class hoặc interface)

│ │ └── app.module.ts // Module gốc

│ ├── assets/ // Hình ảnh, fonts, …

│ │ └── fonts/

│ ├── environments/ // Cấu hình môi trường (development, production)

│ ├── index.html // Entry point HTML

│ └── styles.scss // Styles chính, tích hợp Tailwind và Bootstrap

├── angular.json // Cấu hình Angular CLI

├── package.json // Quản lý package npm

└── README.md

**5. Tạo các thành phần (Component, Directive, Service, …) trên Angular**

Dưới đây là các lệnh Angular CLI mẫu (đã được đánh dấu trong khối mã) để tạo thành phần:

* **Tạo Component:**

ng g c components/header

* **Tạo Service:**

ng g s services/api

* **Tạo Directive:**

ng g d directives/custom-directive

* **Tạo Guard:**

ng g guard guards/auth

* **Tạo Interceptor:**

ng g interceptor interceptors/auth

* **Tạo Module:**

ng g module modules/admin --routing

Các lệnh trên sẽ tạo ra các file tương ứng và tự động cập nhật module nếu cần.

**6. Kết nối Server và Client**

Thông thường, các dự án như thế này có thể được tích hợp bằng cách:

* **CORS:** Cấu hình trong Server (.NET) để cho phép gọi API từ Client (Angular).
* **Proxy (trong Angular):** Tạo file **proxy.conf.json** để chuyển tiếp request từ client đến server trong quá trình phát triển.

Ví dụ, tạo file **proxy.conf.json** trong thư mục Client:

{

"/api": {

"target": "http://localhost:5000",

"secure": false,

"changeOrigin": true,

"logLevel": "debug"

}

}

Sau đó, trong **angular.json**, thêm cấu hình cho serve:

"architect": {

"serve": {

"options": {

"proxyConfig": "proxy.conf.json"

}

}

}

Chạy Angular:

ng serve

Khi client gọi các endpoint bắt đầu bằng /api, chúng sẽ được chuyển tiếp tới server.

**Cách để tạo 1 số thành phần trong angular**

**1. Tạo Component**

**Cú pháp chung**:

ng generate component <đường\_dẫn>/<tên\_component> [tùy\_chọn]

**Ví dụ**: Tạo component photo-management trong thư mục app/admin/photo-management:

ng g c admin/photo-management

Các tùy chọn thường dùng:

* --flat: Tạo file component ngay tại thư mục chỉ định (không tạo thêm thư mục con).
* --inline-style (hoặc -s): Tạo style ngay trong file .ts.
* --inline-template (hoặc -t): Tạo template ngay trong file .ts.
* --skip-tests: Bỏ qua file spec cho component.

**2. Tạo Directive**

**Cú pháp chung**:

ng generate directive <đường\_dẫn>/<tên\_directive> [tùy\_chọn]

**Ví dụ**: Tạo directive text-input trong thư mục app/\_directives/text-input:

ng g d \_directives/text-input

Hoặc nếu bạn muốn nó nằm trong cùng thư mục:

ng g d \_directives/text-input --flat

*(Khi dùng --flat, directive sẽ nằm chung thư mục, không tạo thêm thư mục con.)*

**3. Tạo Service**

**Cú pháp chung**:

ng generate service <đường\_dẫn>/<tên\_service> [tùy\_chọn]

**Ví dụ**: Tạo service members trong thư mục app/\_services/members:

ng g s \_services/members/members

* Lệnh trên sẽ sinh ra members.service.ts và members.service.spec.ts.
* Bạn cũng có thể thêm --skip-tests để bỏ qua file spec.

**4. Tạo Guard**

**Cú pháp chung**:

ng generate guard <đường\_dẫn>/<tên\_guard> [tùy\_chọn]

**Ví dụ**: Tạo guard auth trong thư mục app/\_guards:

ng g guard \_guards/auth

Sau đó Angular CLI sẽ hỏi bạn chọn loại guard (CanActivate, CanActivateChild, …). Chọn theo nhu cầu.

**5. Tạo Interceptor**

**Cú pháp chung**:

ng generate interceptor <đường\_dẫn>/<tên\_interceptor> [tùy\_chọn]

**Ví dụ**: Tạo interceptor auth trong thư mục app/\_interceptors:

ng g interceptor \_interceptors/auth

Interceptor thường dùng để thêm **header** hoặc xử lý **token** khi gọi HTTP.

**6. Tạo Resolver**

**Cú pháp chung**:

ng generate resolver <đường\_dẫn>/<tên\_resolver> [tùy\_chọn]

**Ví dụ**: Tạo resolver user trong thư mục app/\_resolvers:

ng g resolver \_resolvers/user

Resolver được sử dụng để **load data trước** khi vào route.

**7. Tạo Class Model**

Không có lệnh “model” chính thức, nhưng bạn có thể dùng **class**:

ng generate class <đường\_dẫn>/<tên\_class> [tùy\_chọn]

**Ví dụ**: Tạo class user.model.ts trong thư mục app/\_models:

ng g class \_models/user --type=model

Sẽ tạo file user.model.ts.

**8. Ví dụ cụ thể cho cấu trúc của bạn**

Giả sử bạn muốn tạo các thành phần tương ứng với cấu trúc:

1. **\_directives/date-picker**

ng g d \_directives/date-picker

1. **\_directives/text-input**

ng g d \_directives/text-input

1. **\_guards/auth** (giả sử tên guard là auth)

ng g guard \_guards/auth

1. **\_interceptors/auth**

ng g interceptor \_interceptors/auth

1. **\_services/user** (service tên user)

ng g s \_services/user/user

1. **admin/admin-panel** (component)

ng g c admin/admin-panel

1. **admin/photo-management** (component)

ng g c admin/photo-management

1. **admin/user-management** (component)

ng g c admin/user-management

1. **errors/not-found** (component)

ng g c errors/not-found

1. **errors/server-error** (component)

ng g c errors/server-error

1. **errors/test-errors** (component)

ng g c errors/test-errors

1. **home/home** (component)

ng g c home/home

1. **lists/lists** (component)

ng g c lists/lists

1. **members/members** (component)

ng g c members/members

1. **messages/messages** (component)

ng g c messages/messages

1. **nav/nav** (component)

ng g c nav/nav

1. **register/register** (component)

ng g c register/register

**Lưu ý & Mẹo nhỏ**

* Bạn có thể **thêm tùy chọn** --module để tự động import component/service/guard vào một module. Ví dụ:
* ng g c admin/photo-management --module app.module

Khi đó, CLI sẽ import PhotoManagementComponent vào AppModule cho bạn.

* Nếu bạn muốn **tạo module riêng** cho folder admin, bạn có thể làm:
* ng g module admin --routing

Rồi sau đó tạo component trong module admin:

ng g c admin/photo-management --module admin/admin.module

* Bạn có thể **bỏ qua file test** bằng --skip-tests nếu muốn:
* ng g c admin/photo-management --skip-tests

**Tóm tắt**

1. **Component**: ng g c <path>/<componentName>
2. **Directive**: ng g d <path>/<directiveName>
3. **Service**: ng g s <path>/<serviceName>
4. **Guard**: ng g guard <path>/<guardName>
5. **Interceptor**: ng g interceptor <path>/<interceptorName>
6. **Resolver**: ng g resolver <path>/<resolverName>
7. **Class (Model)**: ng g class <path>/<className> --type=model

Áp dụng tương tự cho các thư mục: \_directives, \_guards, \_services, admin, errors, home, lists, members, messages, nav, register... Bạn chỉ cần thay đổi **<path>** theo vị trí mong muốn trong app/.

Nếu có thắc mắc thêm, bạn cứ hỏi nhé! Chúc bạn code Angular vui vẻ. 🚀