**TẠO WEB API VỚI ASP.NET CORE**

**KẾT NỐI ANGULAR 5**

1. **Cài đặt project với Angular 5**

-Trước khi cài Angular cần cài đặt môi trường cho thiết bị.

-Cần cài đặt Node.js, NPM, Typescript version mới nhất (có thể check version bằng lệnh command: node –v, npm –v, tsc -v)

**BƯỚC 1:**

-Cài đặt Angular bằng command sau: npm install -g @angular/cli

**BƯỚC 2: TẠO PROJECT**

-Mở thư mục vào VS Code chạy command sau trong Terminal: ng new FE (FE là tên project)

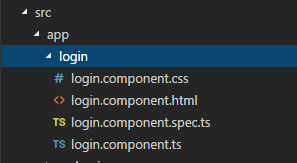
**BƯỚC 3: SERVE VÀO APP**

-Vào project vừa tạo cd FE và chạy ng serve –open

1. **Tạo component**

-Ngoài tạo tay có thể tạo component bằng lệnh command sau: ng g c login

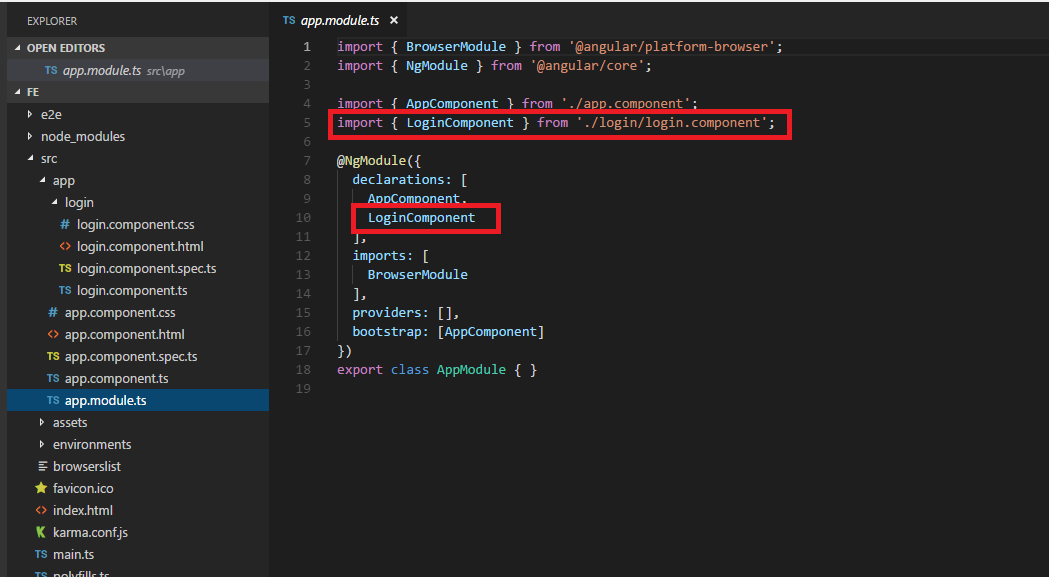
-Sau khi chạy lệnh, app sẽ tự động tạo sẵn thư mục cùng tên với các app như html, css, spect.ts, ts



-Tự link đến các trang trong login.component.ts và có selector là app-login (selector này để ta có thể thêm component này vào bất cứ trang html component nào khác bằng cách add: <app-login></app-login>)



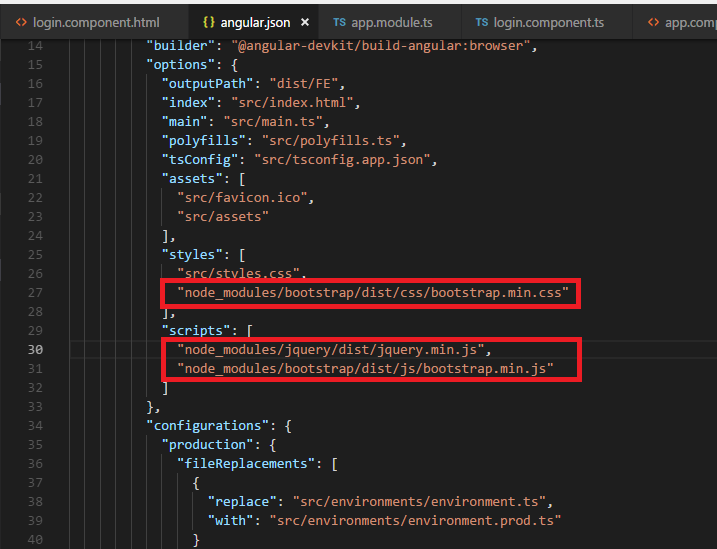
-Và import component vào app.module.ts



1. **Add bootstrap vào angular**

**-**Chạy lệnh sau trong Terminal: npm install bootstrap jquery --save

-Sau đó thêm vào styles và script trong angular.json đường dẫn đến bootstrap



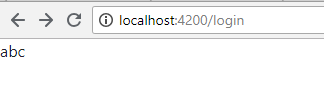
-Cuối cùng, chạy lại ng serve để kiểm tra

1. **Tạo giao diện trang login**

**\*Data binding 1 chiều**

-Biến để trong cặp dấu {{ … }} được gọi là data binding 1 chiều truyền về giao diện.

VD: Ta khai báo **status: string = "abc";** trong login.component.ts phần export và gọi ở giao diện html **{{ status }}** kết quả sẽ như hình:



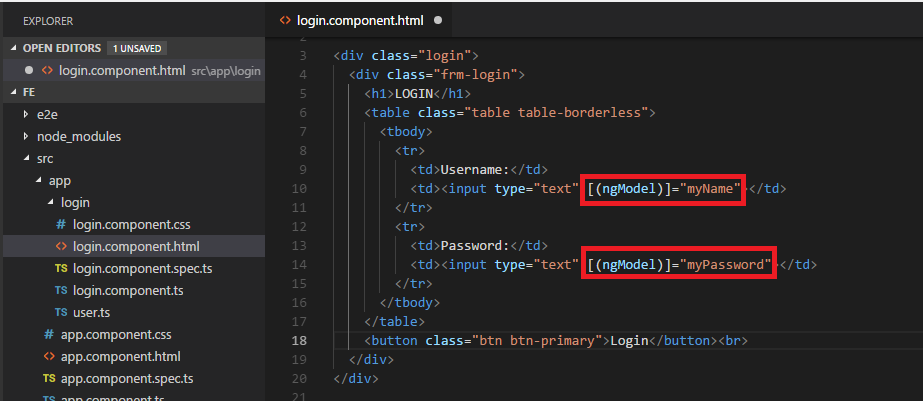
-Để gọi hàm ta sử dụng event binding (gọi là data binding 1 chiều gọi hàm): (event) = “abc()”

VD: Gọi hàm khi click vào nút:

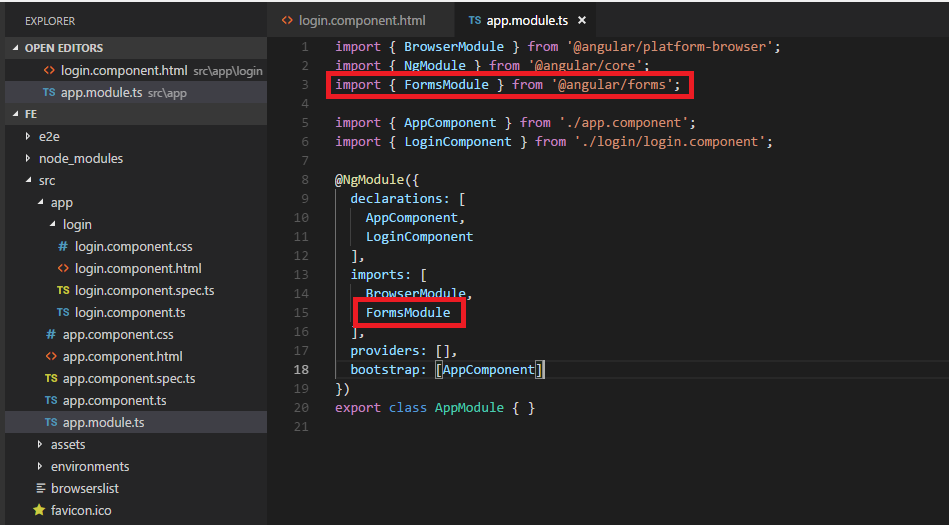


**\*Data binding 2 chiều**

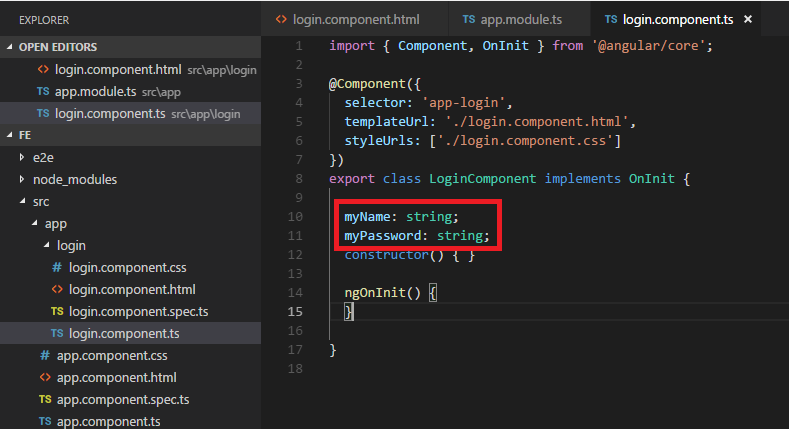
**-**-Tạo form đăng nhập trong app login.component.html sử dụng [(ngModel)] để tạo data binding hai chiều truyền dữ liệu và trả về. Tạo css trong login.component.css.

****

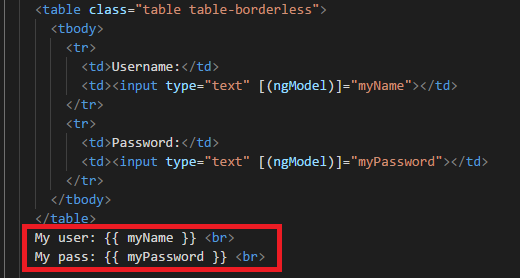
-Để sử dụng được [(ngModel)] phải import vào app.module.ts dòng sau:



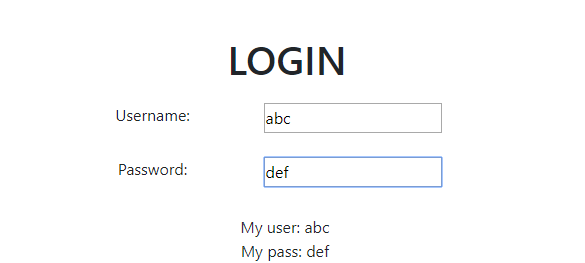
-Sau đó khai báo biến myName, myPassword để sử dụng [(ngModel)] trong login.component.ts



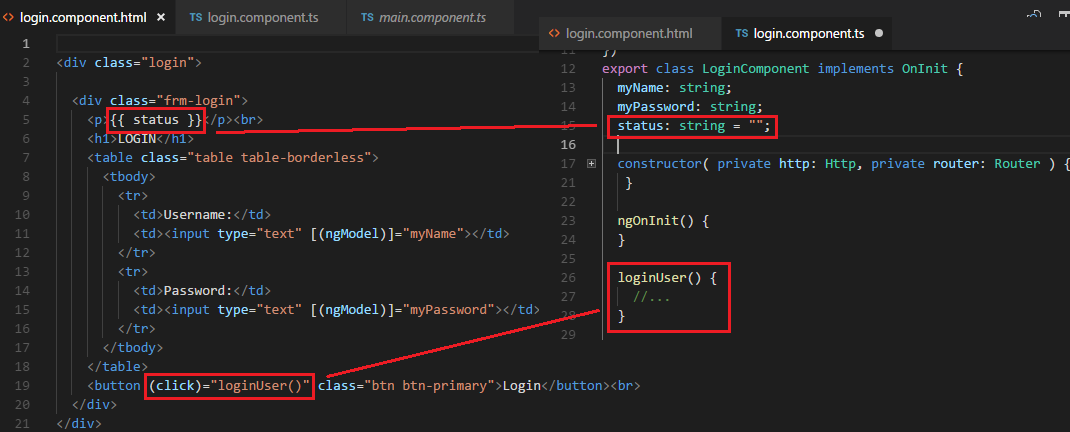
-Để test thử data binding hai chiều có thể thêm biến myName, myPassword trong {{ ... }}, ví dụ test như hình



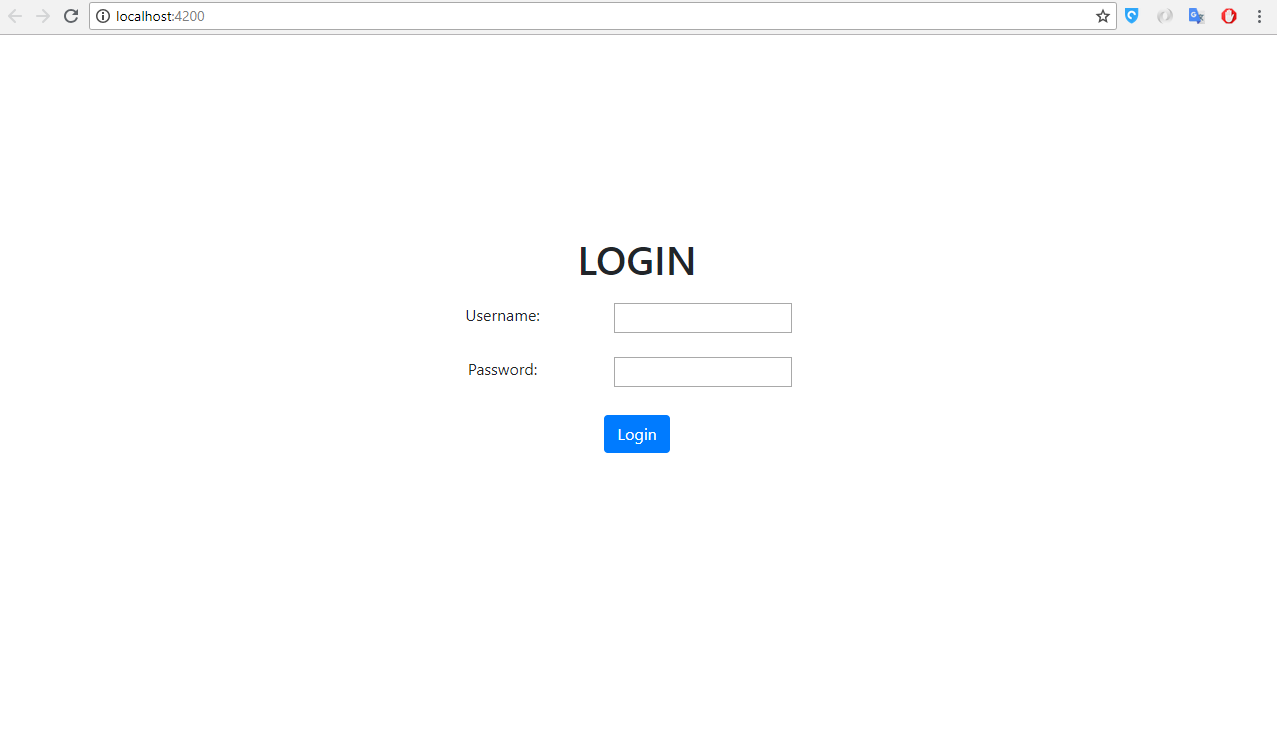
=> Kết quả nhập vào sẽ truyền xuống giá trị như hình:



-Ngoài ra, cần thêm biến status 1 chiều để thông báo đăng nhập, và gọi hàm loginUser() khi click vào Login



-Sau khi tạo xong giao diện login sẽ như sau



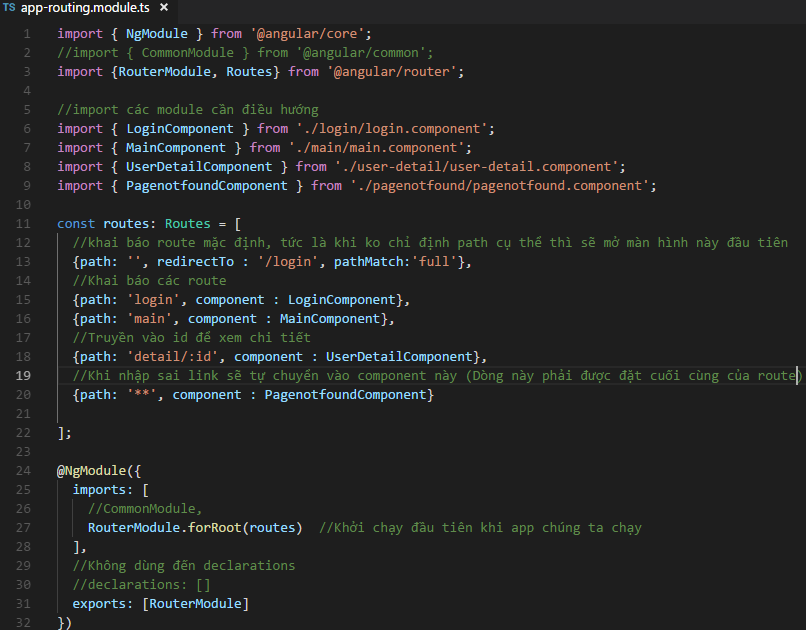
1. **Điều hướng giữa các trang bằng route, cách tạo màn hình mặc định**

Chạy câu lệnh sau trong Terminal: ng generate module app-routing --flat --module=app

**--flat** có thể có hoặc không, ý nghĩa nó là tạo file app-routing.module.ts ngay trong thư mục app, nếu không có nó sẽ tạo riêng thư mục và đường dẫn sẽ là app-routing/app-routing.module.ts

**--module=app** có nghĩa là tự động add module app-routing vào file app-module.ts (nên dùng cách này). nếu không có thì phải làm bằng tay rất mất công

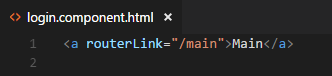
Sau đó, mở file app-routing.module.ts ra và chỉnh sửa 1 vài chỗ cần thiết như comment trong code:



-Như hình chúng ta hiện tại có 2 route chuyển trang là login và main, có thể thêm route nếu phát sinh thêm trang cần điều hướng. Path: ‘\*\*’ là khi user nhập sai link sẽ chuyển sang trang trên. Và path detail truyền vào id hướng dẫn ở cuối phần điều hướng.

-Để sử dụng chúng ta chỉ cần để <router-outlet></router-outlet> vào trang html chính.

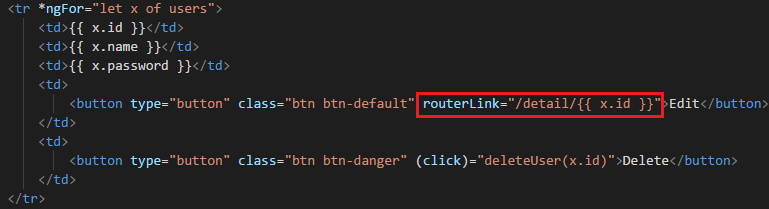
-Để điều hướng đến các trang ta sử dụng routerLink, ví dụ thẻ a như hình tại trang login.



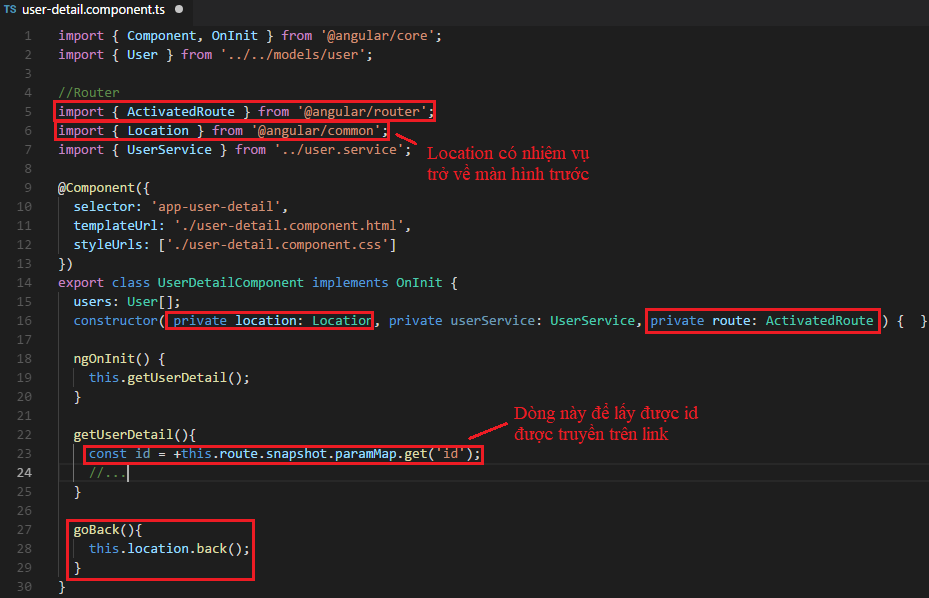
-Để sử dụng route chuyển trang trong các hàm ta cần import và gọi như hình:



-Để truyền vào id cho path detail trên, tại html chính ta chuyển sang trang chi tiết và truyền vào id tại mảng như hình:



-Sau đó tại trang chi tiết ta chỉnh sau:



-Ở đây, ta import thêm Location để trở về trang trước.

-Hàm getUserDetail() được viết bên phần service và gọi tại đây để sử dụng (hướng dẫn phần Sử dụng service bên dưới)

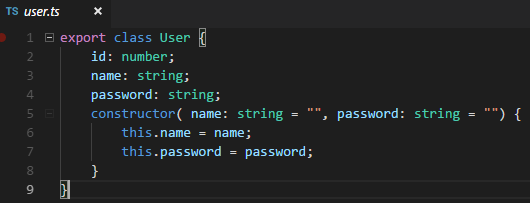
1. **Viết các hàm để kiểm tra đăng nhập**

-Ta có url của api lấy users: http://localhost:5000/api/user (Đọc thêm phần hướng dẫn tạo web API với ASP.NET CORE)

**-Ý tưởng quy trình:** Khi bấm Login sẽ gọi hàm và gán user và password vào 1 class và sử dụng phương thức POST để gửi về Server. Sau khi Server kiểm tra sẽ trả về json và ta biến json đó thành mảng và đếm, nếu như có giá trị (tức là > 0) thì sẽ thực hiện lệnh, ngược lại thông báo fail.

**-Thực hiện:**

-Đầu tiên, ta tạo class user.ts trong thư mục modules cùng cấp với app để sử dụng truyền giá trị user và pass vào



-Chúng ta kết nối đến api có 4 phương thức chính là GET, POST, PUT, DELETE. Để sử dụng được kết nối đến HTTP cần import Http, RequestOptions, Headers vào **login.component.ts**



\*Để sử dụng được Http trong component khác thì cần import HttpModule vào **app.module.ts**



Ngoài import Http, RequestOptions, Headers ta cần import vào User (lớp vừa tạo), map (để map kết quả trả về ra định dạng mong muốn), Router để chuyển trang (xem thêm hướng dẫn trên). Và tạo biến apiUrl là đường link API sử dụng private.



-Sau đó code trong hàm loginUser() với chú thích sau:



-Ngoài ra chúng ta cần tạo thêm 1 hàm để kiểm tra session lưu tại server, để khi load lại trang nếu đã login thì sẽ chuyển thẳng về trang main. Hàm sẽ được tạo bên service (Tìm hiểu bên dưới)

1. **Sử dụng Angular service tạo hàm dùng chung các component**

-Trước khi viết trang để lấy dữ liệu, ta tạo service để viết các hàm.

\*Mục đích phải viết các hàm trong service mà không để trực tiếp trong các file ts của component vì những hàm có thể sử dụng lại nhiều giữa các component chúng ta chỉ cần gọi ở service mà không cần viết lại y hệt và cũng dễ bảo trì sau này. (Ví dụ hàm getUsers() ở hàm login cũng cần sử dụng tại hàm main)

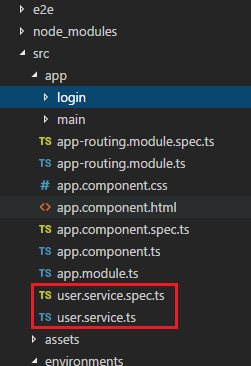
-Để tạo service ta chạy dòng sau trong Terminal:

ng generate service user

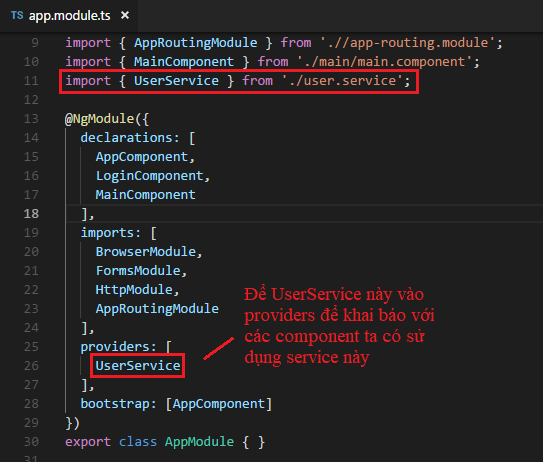
(user: tên service)

(Có thể thêm --module=app vào sau lệnh để tự động add vào module)

-Sau khi chạy sẽ tạo ra 2 file service như hình:



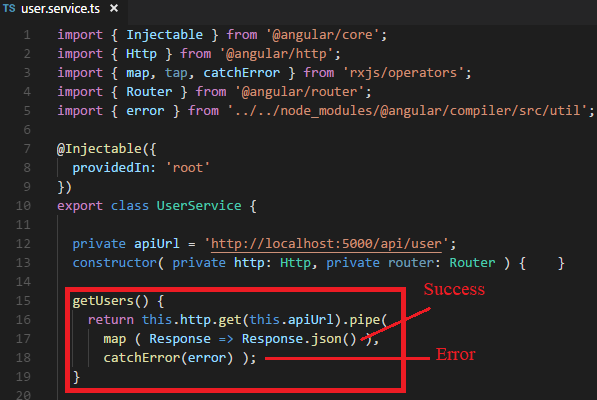
-Sau đó, add service vào module như hình:



1. **Lấy dữ liệu về (Get)**

**-Viết hàm trong service:**

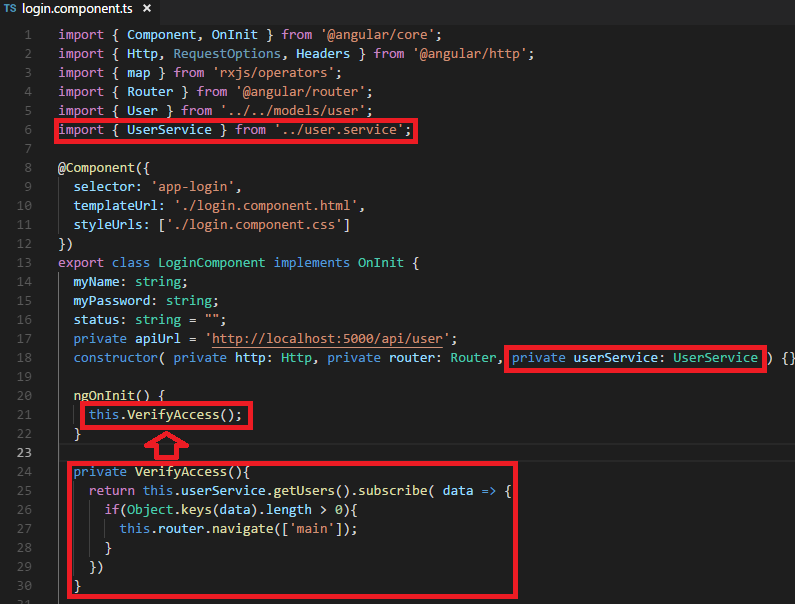
-Ta add hàm getUsers(), viết tương tự phần POST hướng dẫn phần Login trên, nhưng GET không cần tham số truyền đi và không cần Headers và RequestOption, chỉ dẫn đến đường link.



\*Các hàm để tại service không cần Subscribe dữ liệu trả về, mà ta sẽ Subscribe ngay tại component gọi hàm đó.

**-Sử dụng các hàm trong service:**

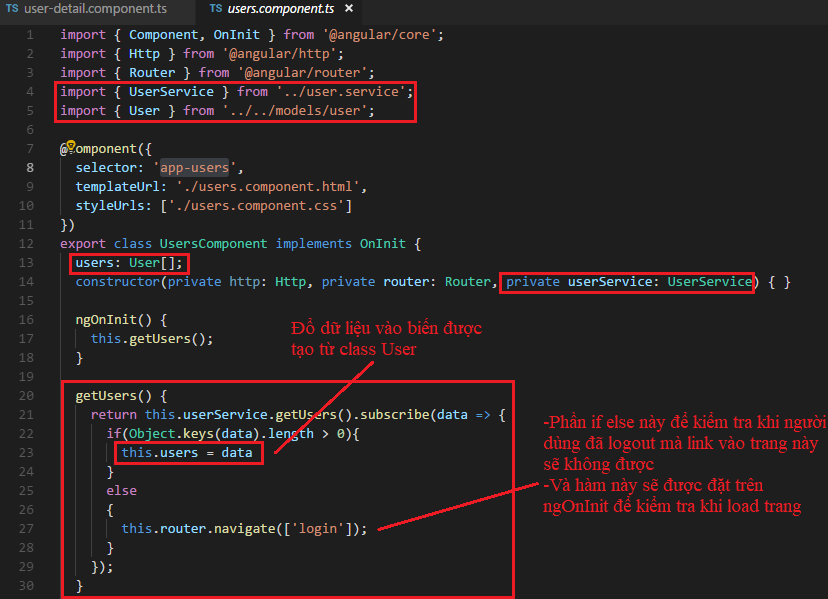
-Để sử dụng service tại component, ta gọi hàm get tại login.module.ts để kiểm tra đăng nhập như sau:



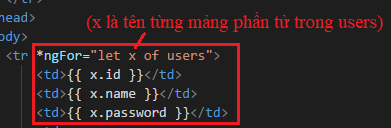
-Sau đó đặt hàm VerifyAccess() vào ngOnInit để khi load trang sẽ vào hàm này tránh đặt vào Constructor vì khi khởi tạo còn thiếu thuộc tính sẽ ảnh hưởng đến performance.

**-**Ngoài ra, ta có thể sử dụng lại hàm getUsers() ở service để lấy dữ liệu đổ vào trang user tương tự cách trên.

-Ở bên trang users tạo bên file ts như chú thích sau:



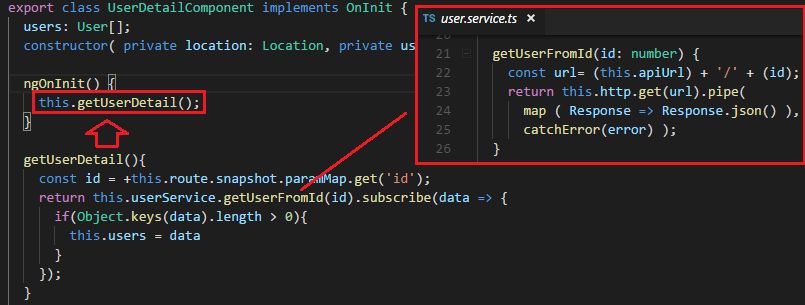
-Và ta sử dụng mảng users hiện lên trang html bằng \*ngFor như sau:



-Với of là lấy ra giá trị trong users, ngoài ra còn có in là lấy ra vị trí mảng trong users.

1. **Lấy dữ liệu về qua id (Get{id})**

**-**Như ở phần hướng dẫn điều hướng chuyển sang trang detail qua cách truyền id lên link trên ta đã có được id và hàm được gọi từ service

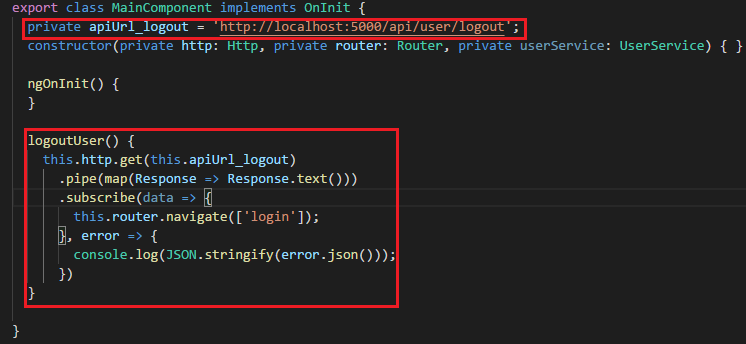


-Hàm sẽ được đặt trong ngOnInit() để khi chuyển

1. **Logout**

**-**Đây cũng là 1 dang của Get nhưng lấy đường link api khác.

-Hàm này có thể viết ngay tại hàm chứa chức năng Logout không cần để trong service vì hàm không cần thiết sử dụng nhiều (tùy vào nhu cầu mà thiết kế)

****

1. **Thêm (Post)**

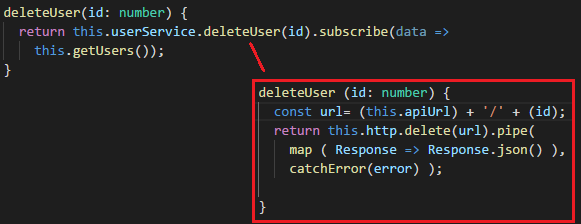
**-**Ta dùng [( ngModel )] để lấy thông tin thêm và tạo hàm được gọi từ service:



1. **Xóa phần tử qua id (Delete{id})**

-Ta dùng http.delete để sử dụng xóa tại service.

-Hàm được viết tại file ts như sau:



-Sau khi xóa sẽ sẽ gọi lại hàm getUsers() để load lại trang.

1. **Cập nhật phần tử qua id (Put{id})**

-Tương tự ta dùng http.put viết ở service và gọi như sau:



1. **Local Strorage và Session Storage**

**-Local Storage** được sử dụng để chứa dữ liệu không giới hạn thời gian. Những dữ liệu này sẽ không bị xóa khi người dùng đóng trình duyệt.

-Để set ta thực hiện cú pháp như ví dụ:



(user: tên đặt cho Local Storage, this.myName là giá trị cần gán)

-Để gọi:



-Để xóa:



**-Session Storage** hoạt động giống như Local Storage ngoại trừ việc nó chỉ lưu trữ dữ liệu trên một phiên làm việc. Dữ liệu sẽ bị xóa khi người dùng đóng cửa sổ trình duyệt.

-Cú pháp sử dụng như Local Storage.