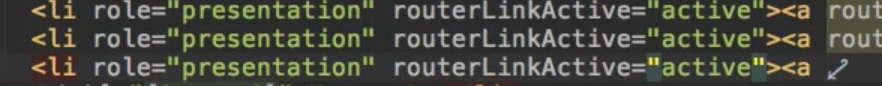
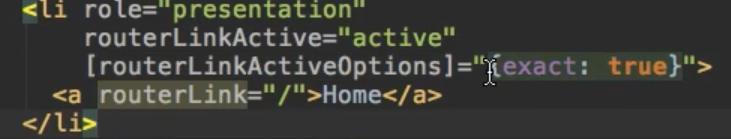
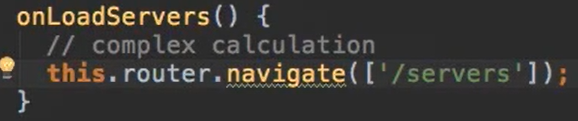
1. SECTION 11: ROUTING and NAVIGATION:
   1. Setting up and Loading Routes:Thiết lập các đường dẫn trên address Bar, đường dẫn URL.
      1. 
      2. 
      3. 
   2. Navigating with RouterLink: (Nhấp vào item thì tự động chuyển trang Route)
      1. 
   3. Understanding Navigation Bar: (Hiểu rõ hơn về đường dẫn tuyệt đối( relative), đường dẫn tương đối(absolute)).

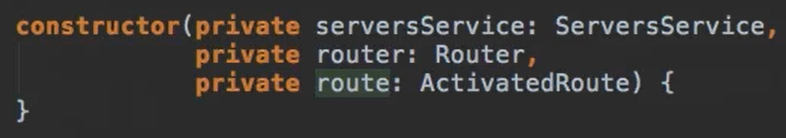
* GIẢ SỬ đang ở trang: abc.com/test
  + 1. Relative Path: 
  + So với trang root page: abc.com/server (tuyệt đối)
    1. Absolute Path
  + So với trang hiện tại: abc.com/test/servers
  1. Styling Active Router Links: Thiết lập CSS cho active item 1 cách tự động:
     1. 
  + Thêm property ‘routerLinkActive’ = “*className*” ở mỗi Item
    1. 
  + Thêm property ‘routerLinkActive’ cho 1 thằng Home Item thui.
  1. Navigating Programmatically : Điều hướng page bằng hàm .navigate():
     1. 

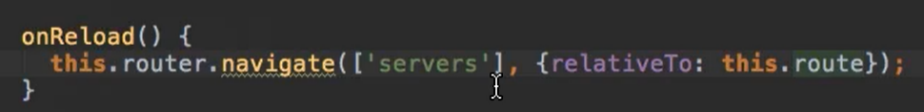




* + 1. Vì method() navigate() sẽ không biết vị trí hiện tại của mình ở đâu giống như routerLink vì thế mình cần phải thêm property ‘relativeTo: ’ cho method .navigate(), nhớ import thêm ActivatedRoute.







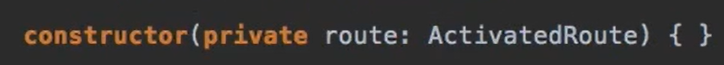
* 1. Passing Parameters to Routes:Gỉa sử đang ở path: abc.com/users
     1. 

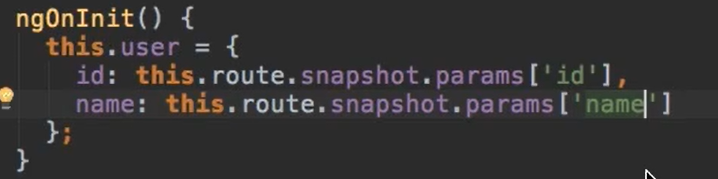
Tại route UserComponent ta có thể lấy được id từ url path khi vừa chuyển qua trang đó.

Đăng nhập: abc.com/users/’bất cứ gì’ đều chuyển trang đến UserComponent

* 1. Fetching Parameters Routes: Lấy :id đã truyền từ url Path đưa vào component đang tại vị.(mọi Parameters đều có thể access bằng ActivatedRoute)
     1. Import ActivatedRoute









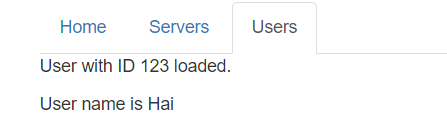
Cái ‘id’ phải giống với biến ‘/:id’ trên 1.6.1 mới được.

Như vậy thì khi vừa đăng nhâp vào trang, ta liền có thế lấy được :id và :name.

Sau đó gán cho biến User trong NgOnInit() là có thể khởi tạo User cho nó và giữ dữ liệu :id và :name.

Từ User{}, ta truyền ra ngoài Output.

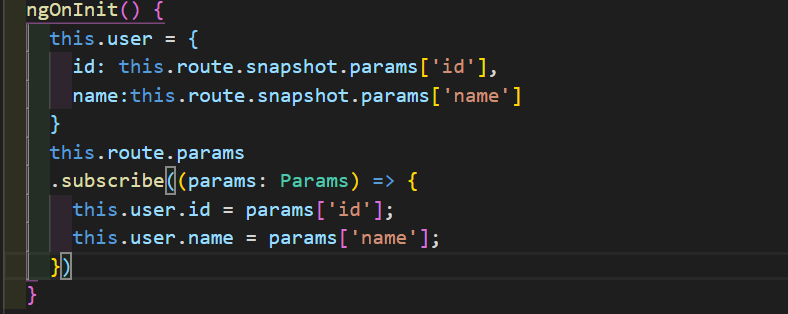
GIẢ SỬ đăng nhâp vào abc.com/users/123/Hai



* + 1. Fetching Route Parameters Reactively:Gỉa sử đang ở trang abc.com/users/1/Hai , trong đó có 1 routerLink = abc.com/users/69/Kim (routerLink chứ không phải refresh lại url path) thì sẽ bị lỗi. Vì nó đang ở Component A thì nó sẽ không cần phải khởi tạo lại Component A thì parameters thay đổi => sẽ không chạy lại hàm ngOnInit() như trên.

**GIẢI PHÁP:**

**Thêm** 1 method Subscribe() để kiểm tra sự thay đổi của parameters:

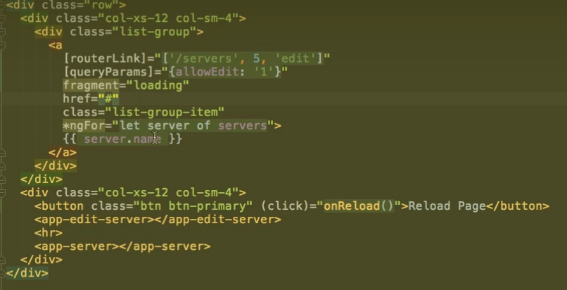


**PHÂN BIỆT TRƯỜNG HỢP SỬ DỤNG:**  dùng hàm Snapshot.params là chắc chắn có.

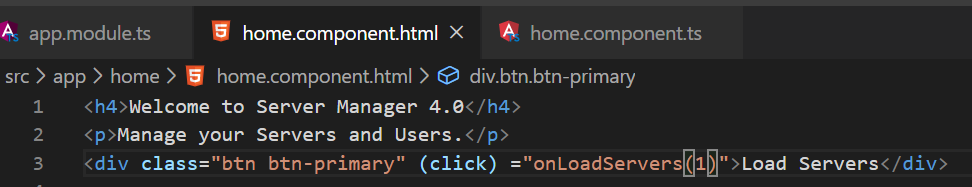
**\_ Không cần add, thêm , sử dụng thêm hàm subscribe()** nếu như biết chắc chắn rằng **Component**(không phải Trang) mà mình đang hoạt động , sẽ không reload lại chính **Component** nó từ trong nó .

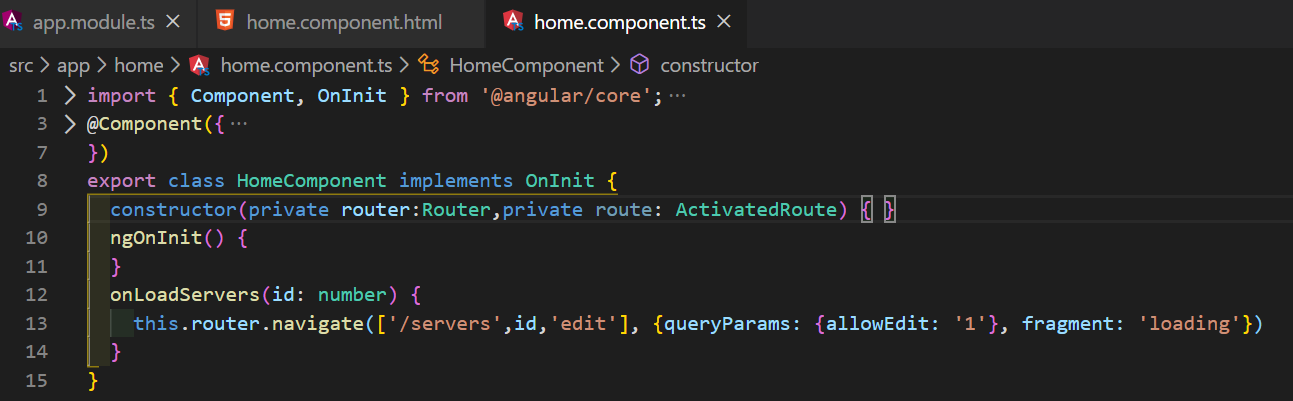
**\_ Ngược lại** thì add thêm vào.**Lưu ý**: Nếu có add thêm hàm Subcribe() thì phải có thêm hàm unsubscribe() ở lifecycle Hook ngOnDestroy()

* 1. Passing Query Parameters and Fragments: Truyền các ?mode=editing, #loading vào path Url 
     1. Cách thủ công (manually):

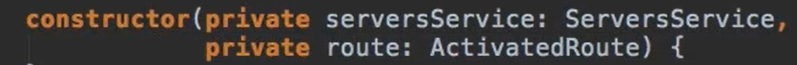


* + 1. Cách viết hàm (programmatically)

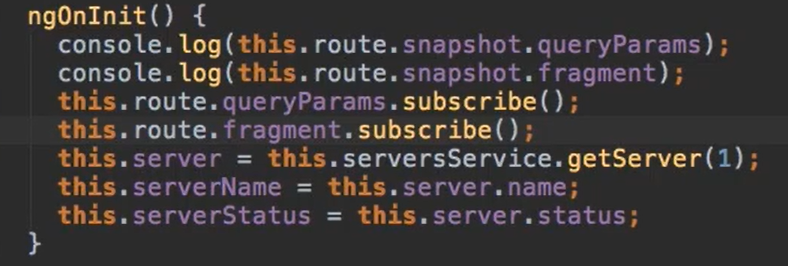




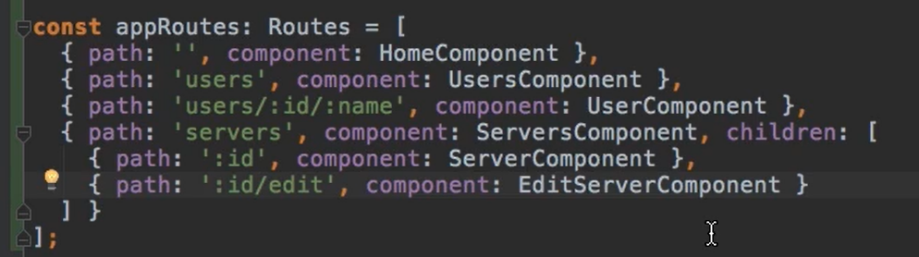
* 1. Retrieving Query Parameters and Fragments: Lấy (get) các ?mode=editing, #loading trong path Url, làm giống như 1.7 (fetching parameters nhưng có 1 ít thay đổi):
     1. Import ActivatedRoute:



* + 1. Dùng method Snapshot và subcribe() để lấy thông tin từ Url, nhưng thay vì dùng .params thì mình thay bằng .queryParams và .fragment.

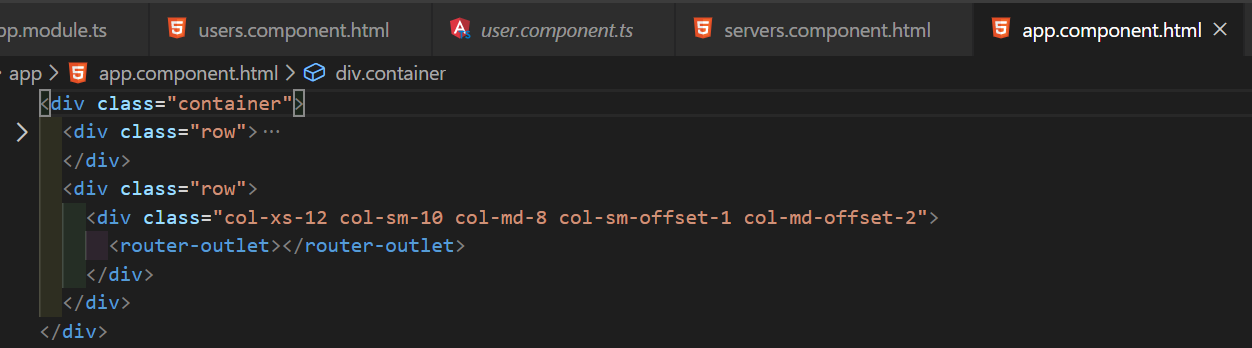


* 1. Setting up Child Route (Nested):

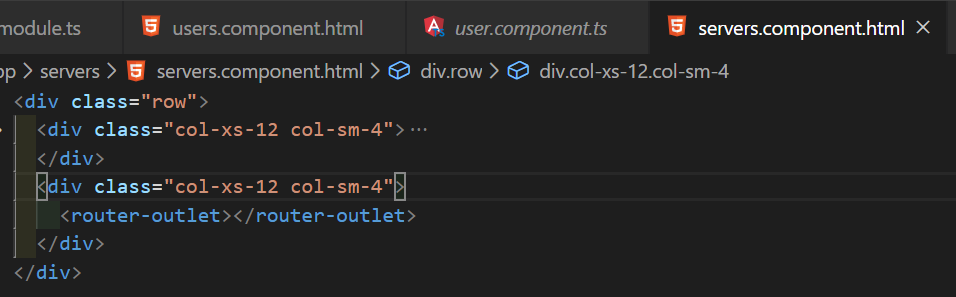


Lúc này sẽ bị lỗi vì: abc.com/server thì nó vào ServersComponent rồi, nếu truy cập vào abc.com/server/:id thì sẽ không có nơi nào hiển thị ServerComponent nữa.

Do đó phải có <router-outlet> con trong thằng <router-outlet> cha.

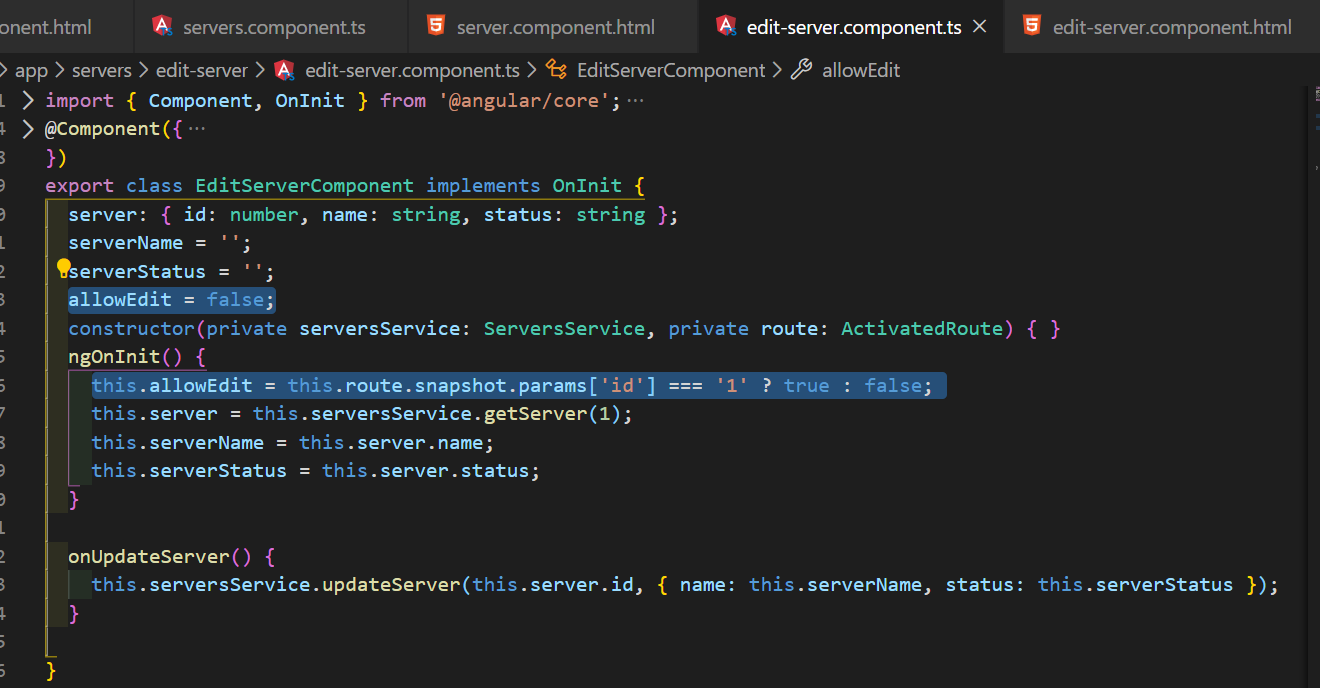


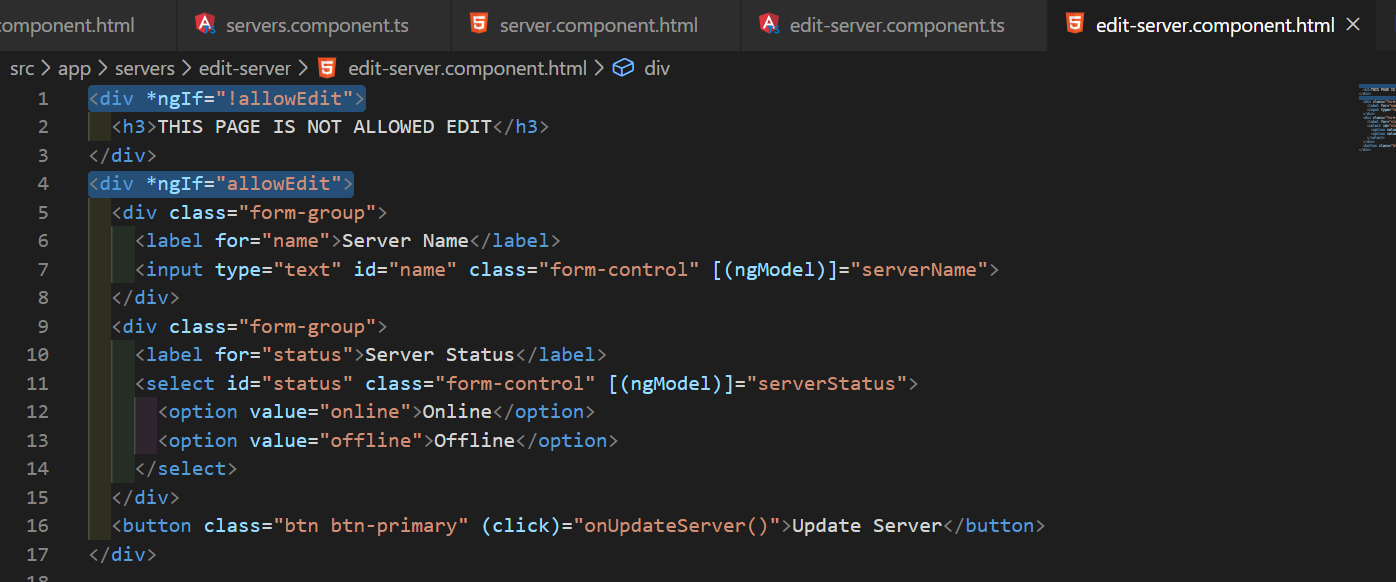
-Router-Outler Cha cho ServersComponent.



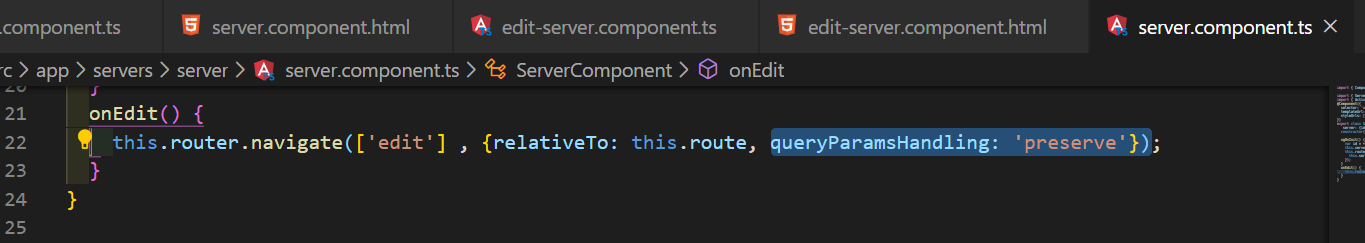
-Router-Outler Con hiển thị ServerComponent, hiển thị trong ServersComponent cha.

1.11 :Using Query Parameters (lấy (get) các Query Parameter (:id) để set các điều kiện hiển thị ra Output hay tính toán), trong bài này, dùng :id để xem xét coi có cho phép Edit Server (chỉnh sửa) hay không. (allowEdit).

Lấy :id rồi xét coi nó được phép Edit hay không.Sau đó tính toán xong, sẽ hiển thị ra Output EditServer.html

Nếu bằng true thì hiển thị, không thì thông báo ko hiển thị.

* 1. Configuring the Handling of Query Parameters. (how to: merge(gộp) hoặc preserve(bảo tồn) các Query Parameters từ thằng component Cha sang component Con)



* 1. **Redirecting and Wildcard Routes:**
     1. Khi ta nhập url path: abc.com/’sadsadsa không có trong routes Path’ thì sẽ không truy cập được Component để hiển thị ra <Router-Outlet> => Lỗi

**BIỆN PHÁP:**

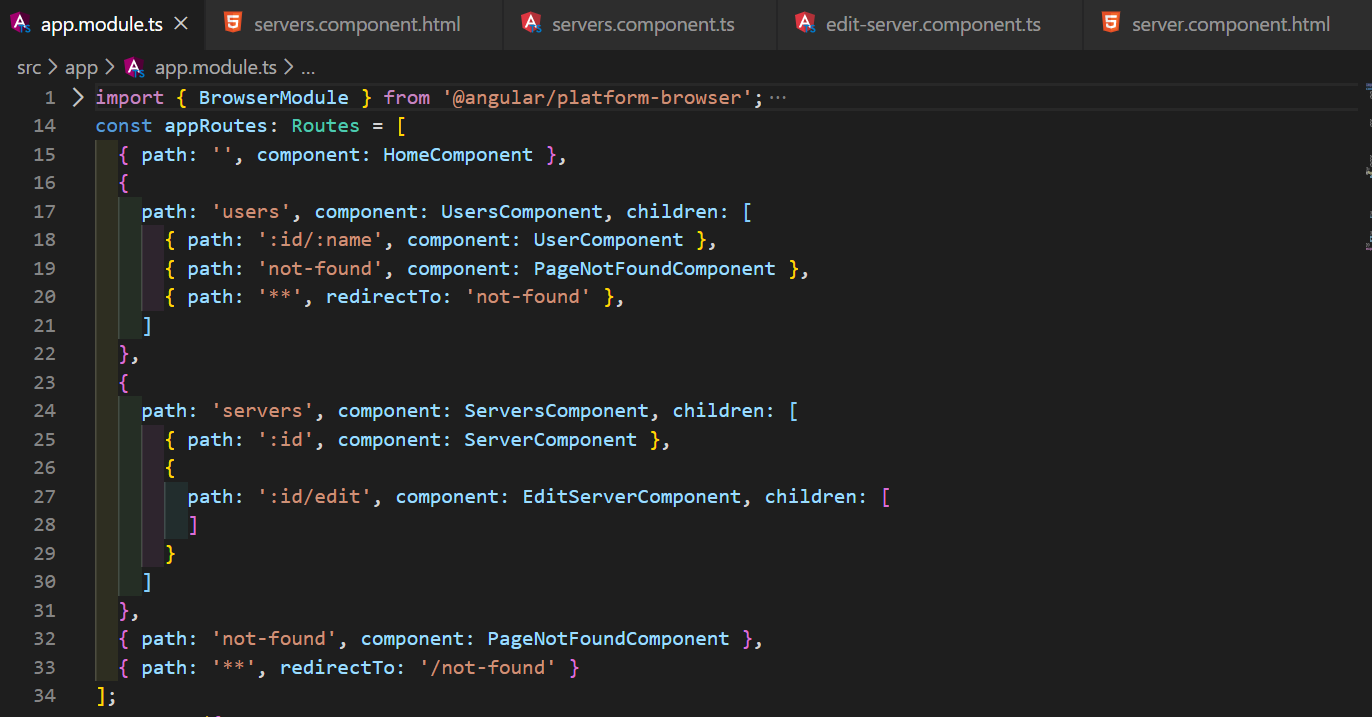
Cần giải quyết như lỗi error 404.

Tạo ra 1 component ‘page-not-found’.

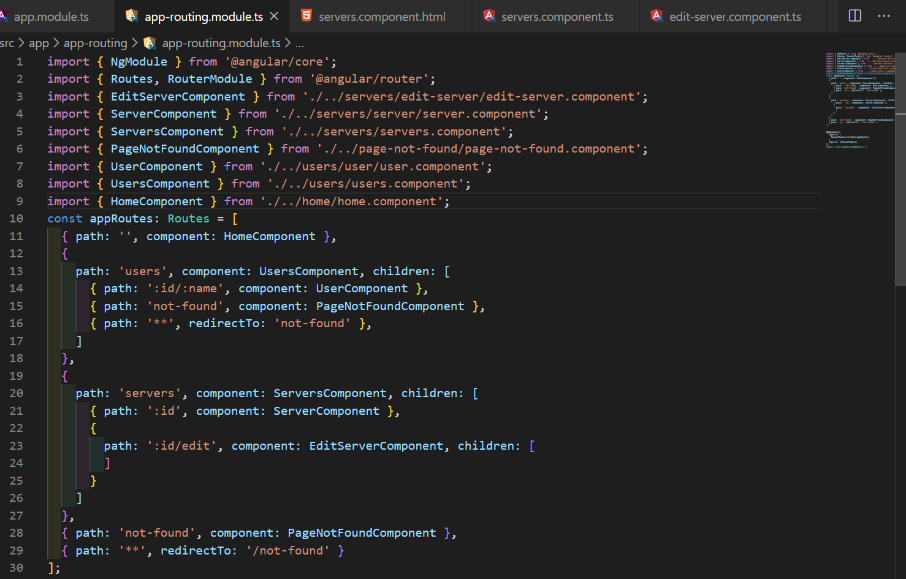
Tạo đường dẫn path “not-found”tới PageNotFoundComponent.

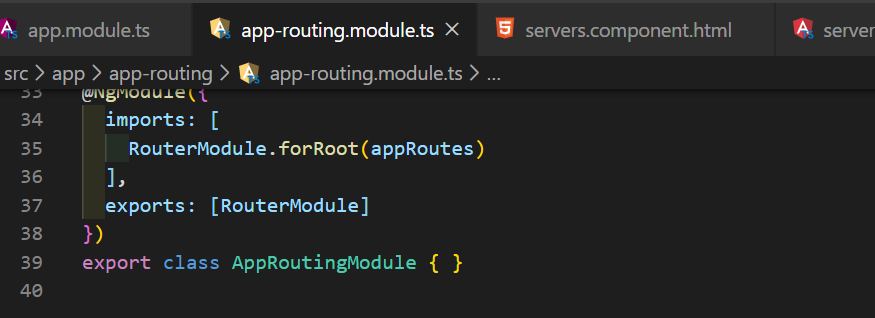
Tạo 1 đường dẫn khác có chức năng RedirectTo, đường dẫn này sẽ trò về đường dẫn “./not-found” khi người nhập nhâp sai với path: “\*\*”.

**Lưu ý**: 2 đường dẫn này phải đặt ở cuối cùng trong Routes, nếu không sẽ lỗi.



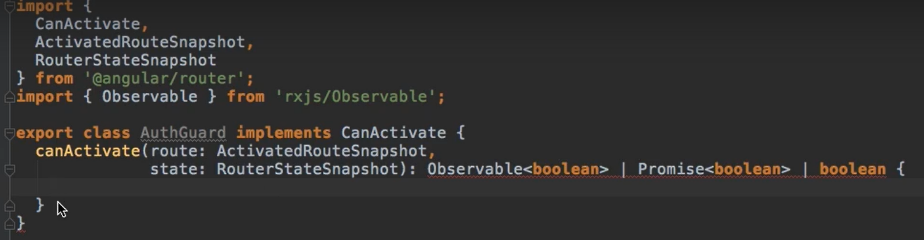
* 1. Outsourcing The Route Configuration. (Tách Routes ra thành 1 file AppRoutingModule riêng biệt, không viết trong file AppModule nữa):
     1. Ở App folder, ng g m app-routing
     2. Tạo các Routes như thường:

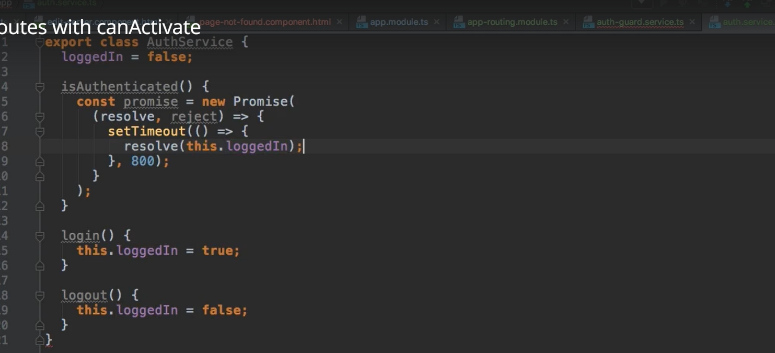
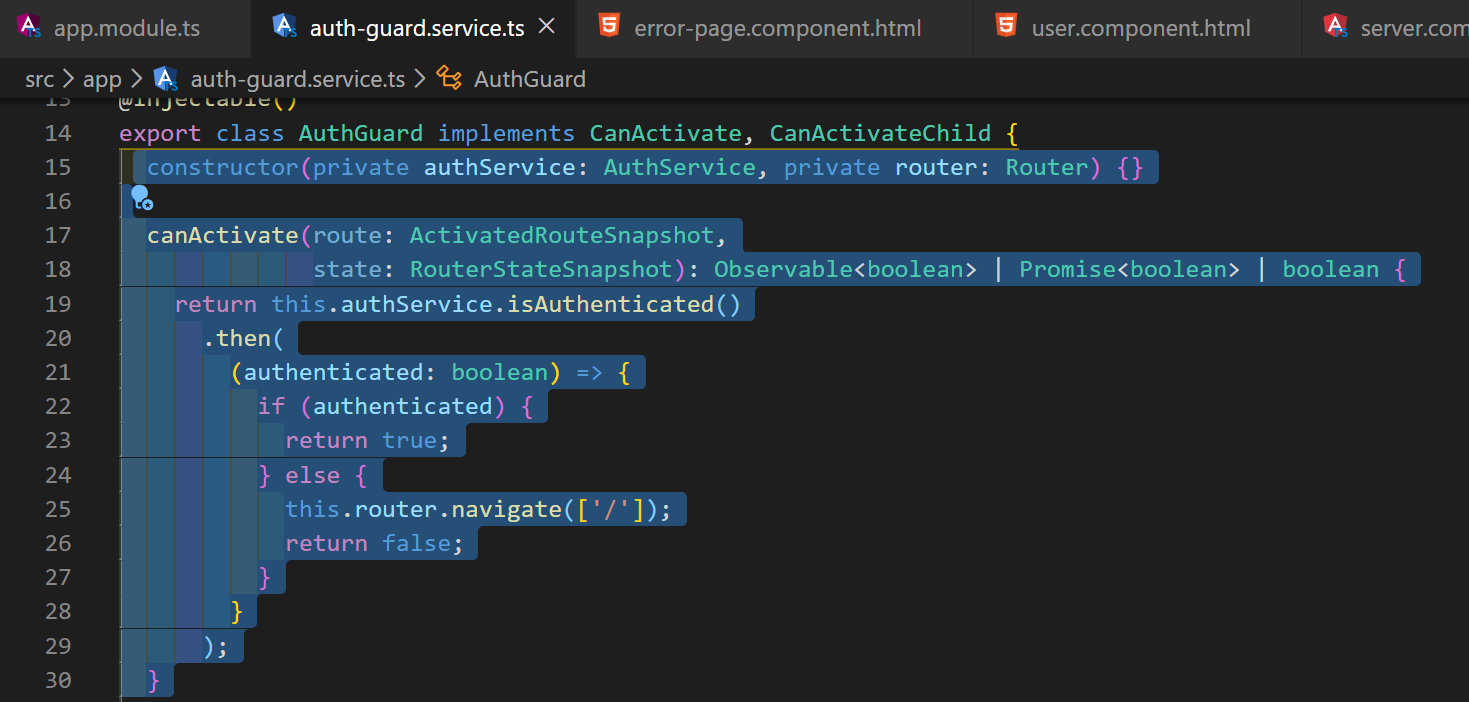
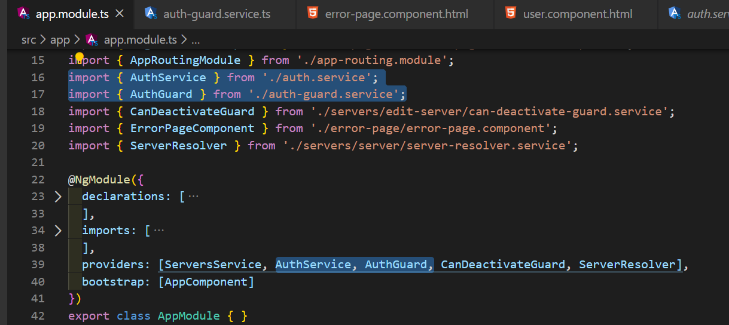


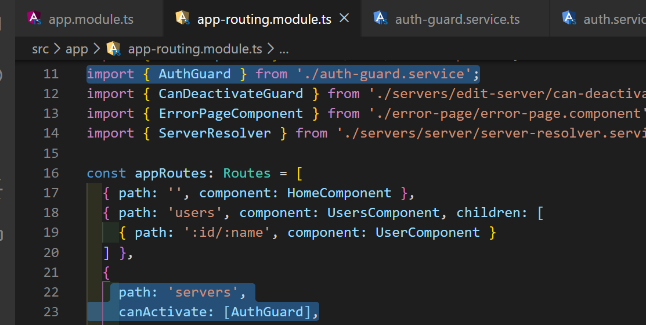


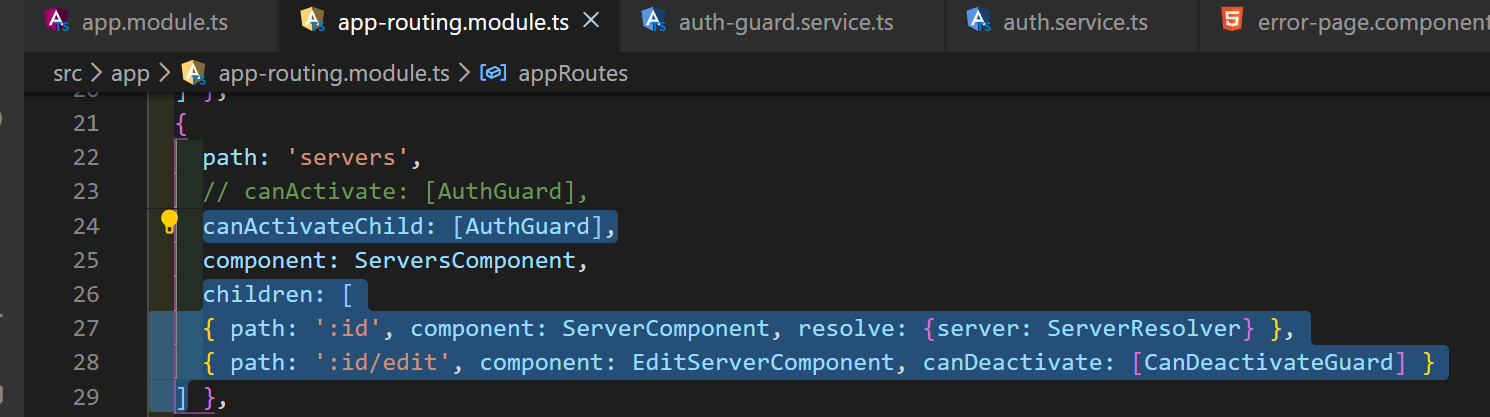
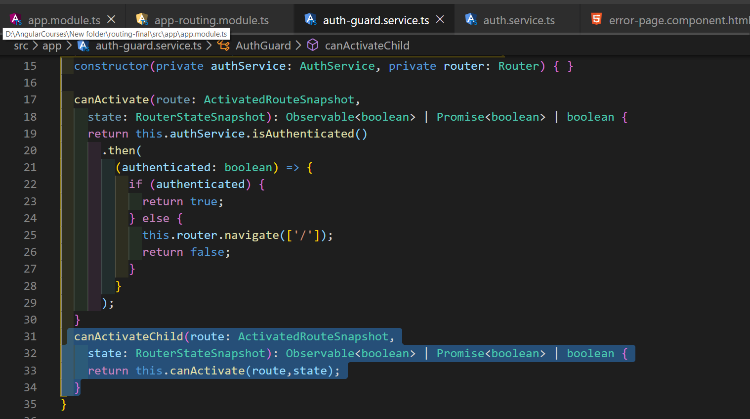


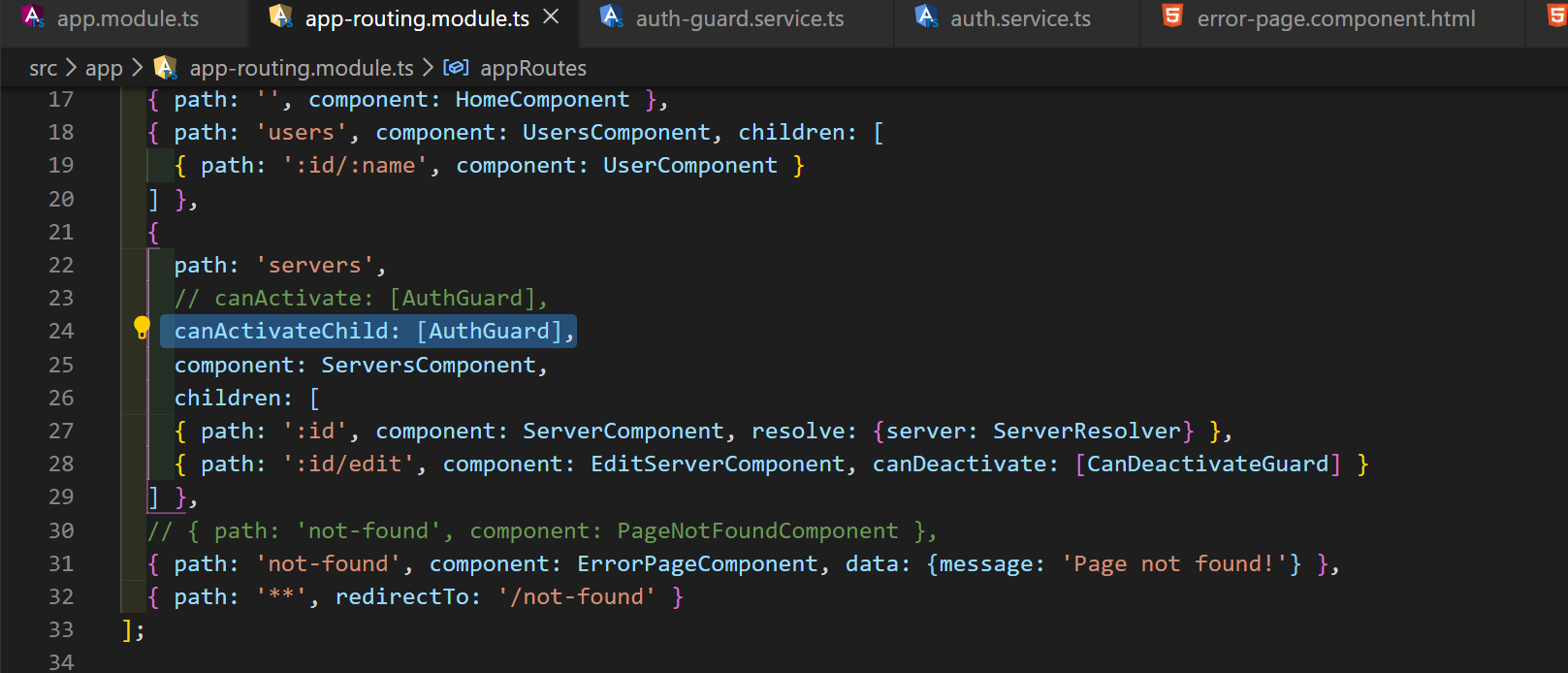
* 1. **Protecting Routes with CanActivate:** 
     1. Tạo auth-guard.service.ts:
        1. Implement canActivate, nhận vào 2 argument : activatedRouteSnapshot, RouterStateSnapshot, có thể return về **1 Observables,** biến Observables này theo dõi biến Boolean, **hoặc** 1 promise, **hoặc** 1 biến Boolean:

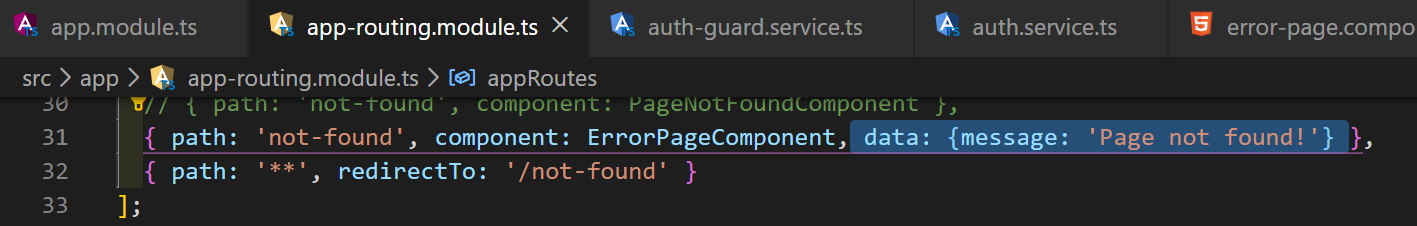
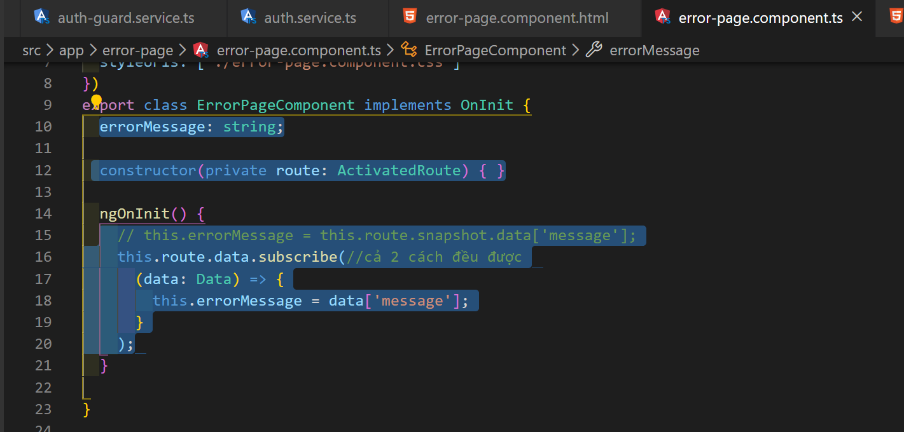
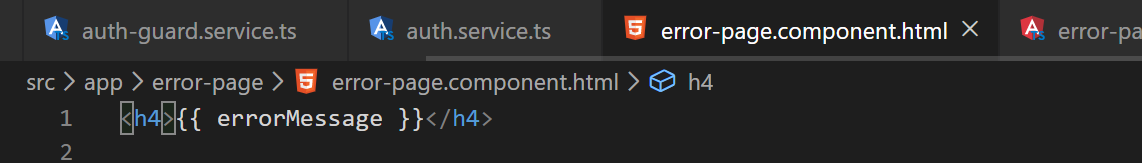
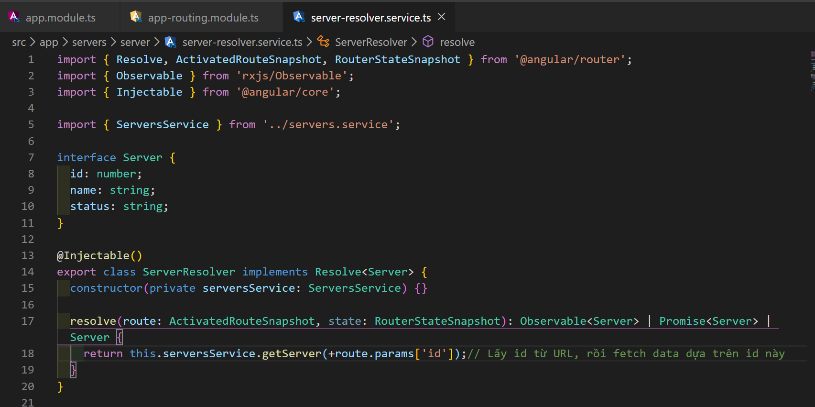
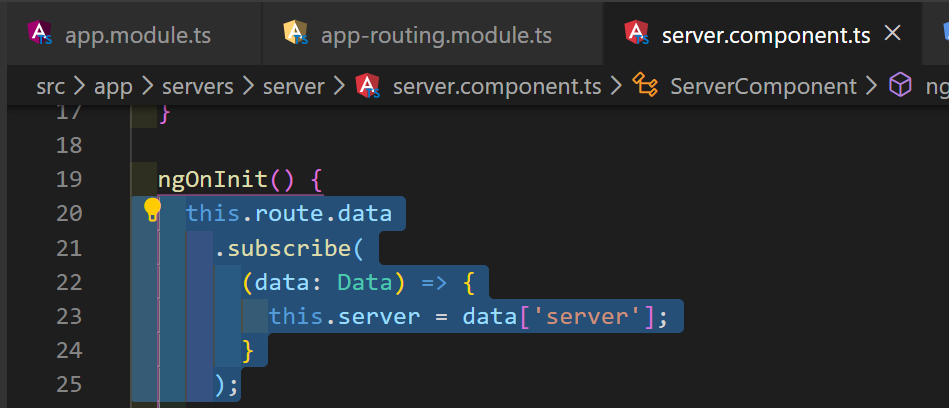


* + 1. Tạo auth.service.ts: file này dùng viết các chức năng như để kết nối với server, cho phép chúng ta login hay log-out hay kiểm tra trạng thái xác thức (authenticated state)
       1. 
    2. Inject AuthService vào AuthGuard và xác định authenticatedState để navigate đến các Route thích hợp:
       1.  - Inject AuthService vào AuthGuard
       2. -Hàm canActivate() return True thì guard cho truy cập vào Component đó, ngược lại, trả về false thì không được.
    3. Phải Import AuthService, AuthGuard vào ‘*App.module.ts’:*
    4. Để sử dụng guard đó, ta phải vào App-Routing.module để thiết lập xem route nào nên được bảo vệ bởi Guard :
       1. Gỉa sử ta muốn bảo vể component Server, chỉ khi nào Login=== true thì mới truy cập được navigate vào path : ‘server’:



* 1. **Protecting Child Routes with canActivateChild():** Dùng để bảo về các Route con như - cho truy cập vào route ‘servers’ nhưng không cho truy cập vào route ‘servers/:id’ và route ‘servers/:id/edit’.
     1. Định nghĩa hàm canActivateChild():  -Đơn giản là gọi lại method canActivate() trên.

 - thêm canActivateChild: [] vào route nào mình muốn bảo về child Routes của Route đó

* 1. **Passing Static Data to a Route:** truyền dữ liệu tĩnh vào 1 component thông qua Route:
     1. Gỉa sử muốn truyền 1 dữ liệu, ‘message’ vào ErrorPageComponet thông qua Route :
     2.  - có thể dùng cả 2 cách để lấy data.
     3.  -Hiển thị ra ngoài Output.
  2. **Resolving Dynamic Data with the Resolve Guard:** Truyền dữ liệu động thông qua Resolver:
     1. Resolver là 1 service giống như canActivate, cho phép ta chạy 1 số code do ta thiết lập trước khi 1 route được render.Resolver khác canActivate là nó không quyết định xem route này có được render hay không
     2. Resolver luôn render lại component, nhưng sẽ làm 1 số chức năng trước pre-loading như fetch data….
     3. Ví dụ: ta muốn lấy các thông tin Server từ Back-end để hiển thị trên mỗi server: (nghĩa là ta sẽ chạy 1 số code trước pre-loading để fetch data từ server). Để làm được điều đó, ta tạo file server-resolver.service.ts
     4. -Hàm resolve() sẽ được gọi mõi khi component được render lại.
     5. 
     6.  - [‘server’] dòng 23 phải trùng tên do mình đặt với {server: ServerResolver} dòng 27 hình trên.