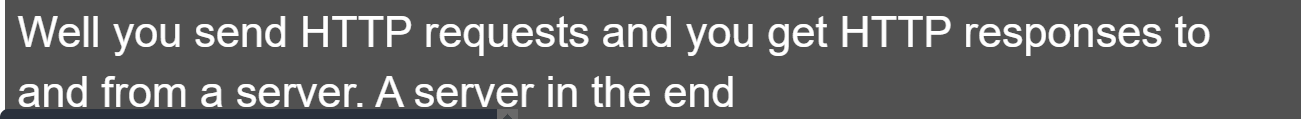
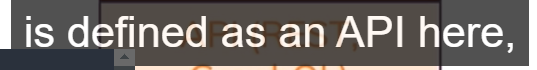
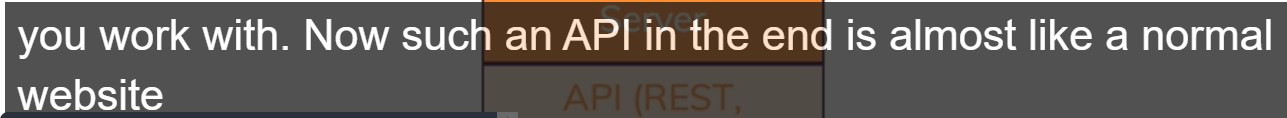
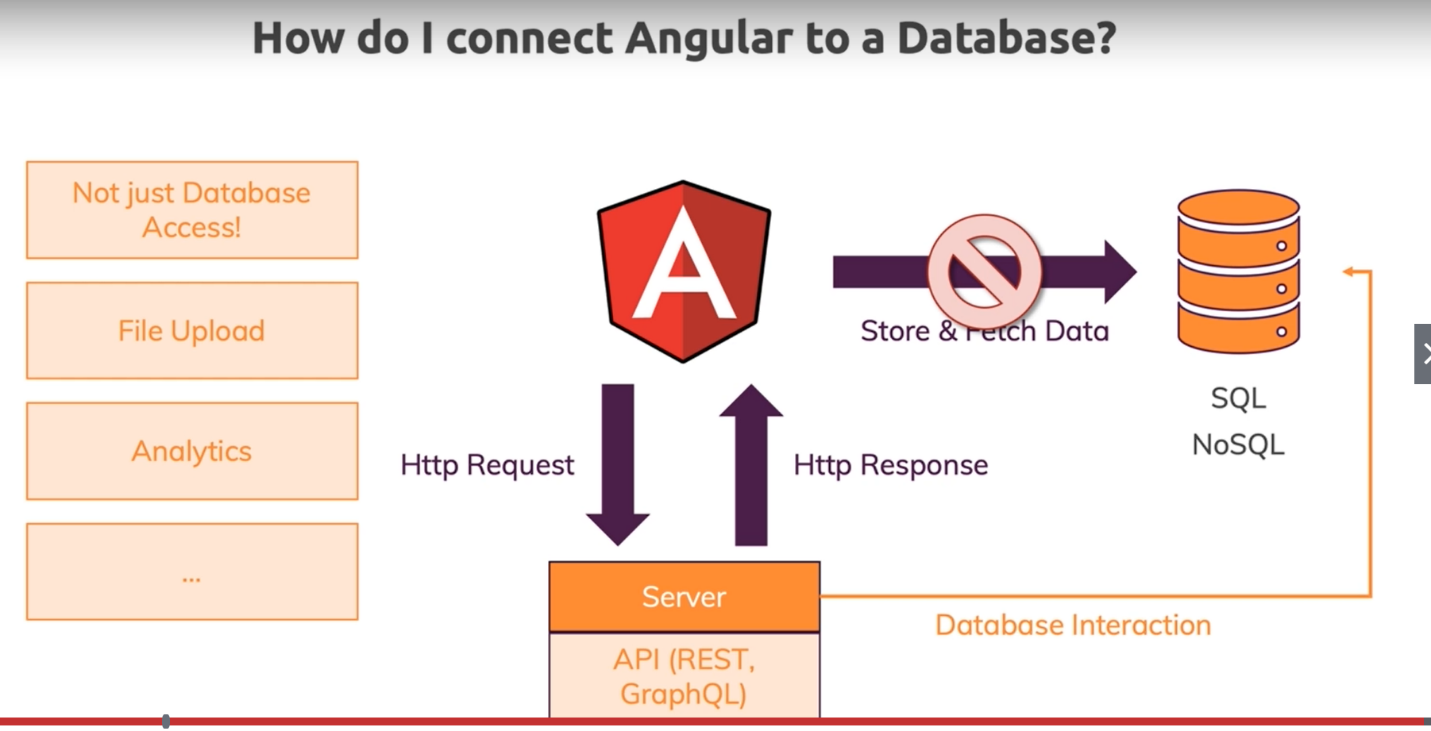
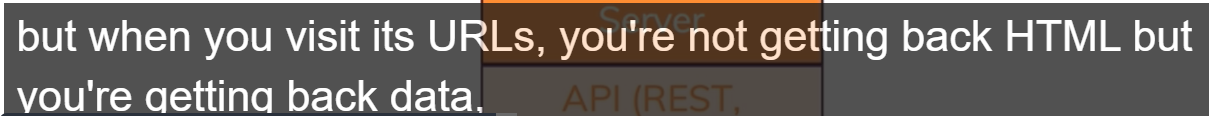
1. **SECTION 18:** Making Http Request: (learn how to: Angular communicates with a back-end, with a database, how to send Http Requests and how to transform data)
   1. **How does Angular interact with Backends**:Không thể dung Angular lưu trữ, lấy, thao tác trực tiếp với Database được vì vấn đề an toàn bảo mật thông tin.



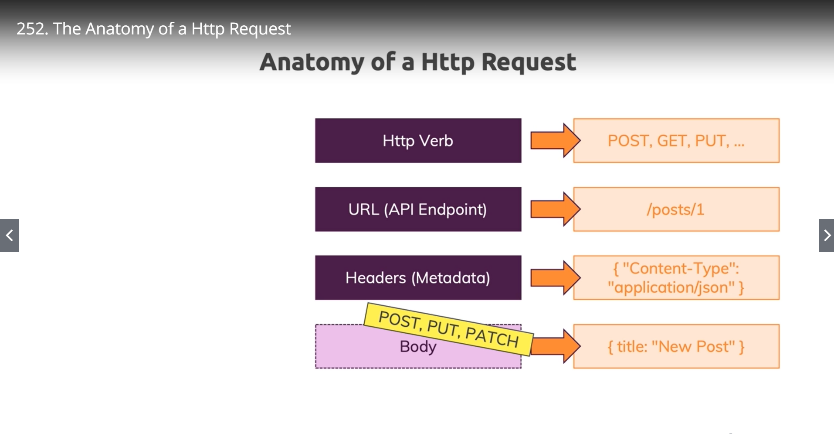




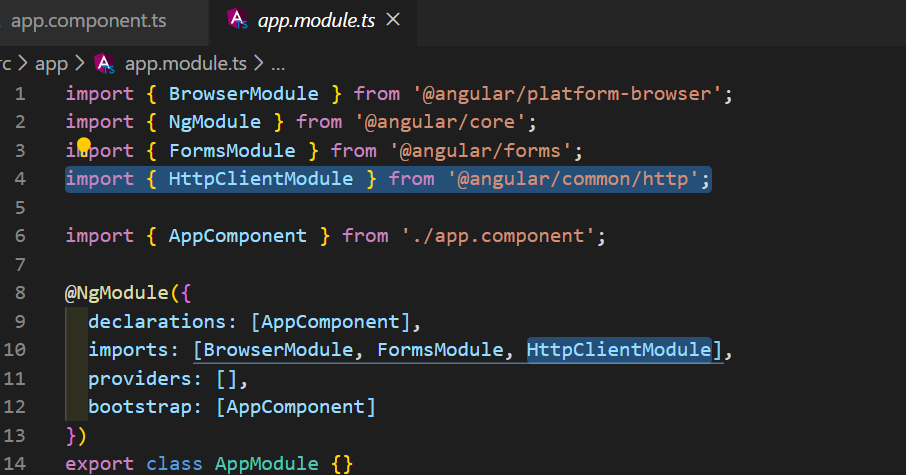


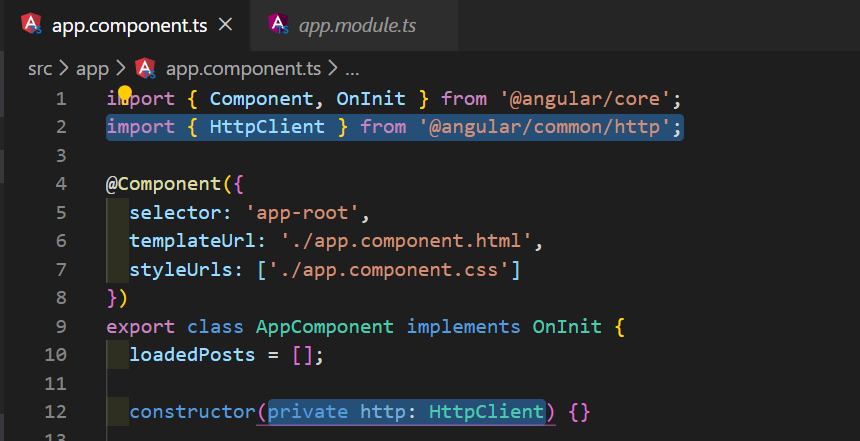
Các Data trả về hầu hết nằm ở dạng JSON

* 1. **The Anatomy of a Http Request:** hiểu rõ hơn, chi tiết hơn về Http Request là gì



* 1. **Backend (firebase ) Setup :** firebase is not just a database. It is a complete back-end service.
     1. Vào firebase.google.com để tạo project như khóa học Reactjs lần trước đã học
  2. **Sending a Post Request**
     1. Set up đầy đủ HttpClientModule:

****

****

* + 1. Send HTTP Requests: mình đã inject các HTTPModule rồi thì bây giờ có thể sử dụng các method() như post, delete ………

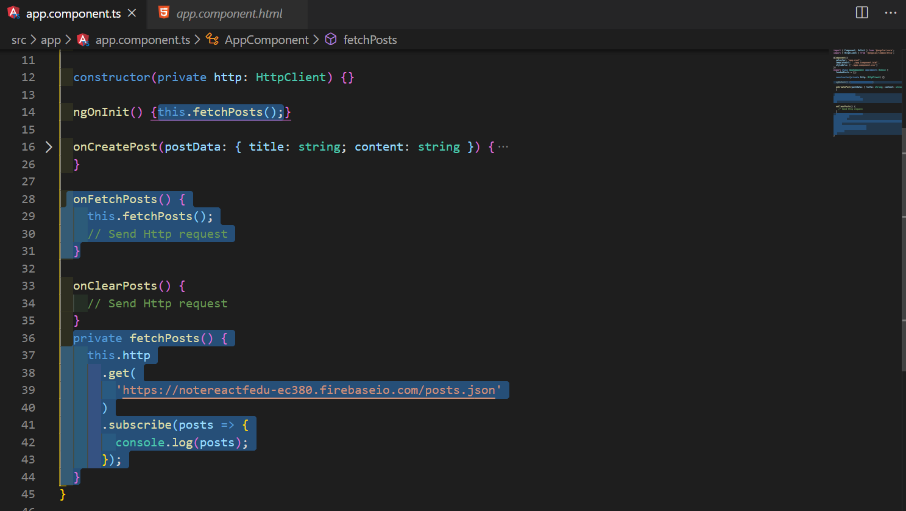


.post() gồm 2 argument:

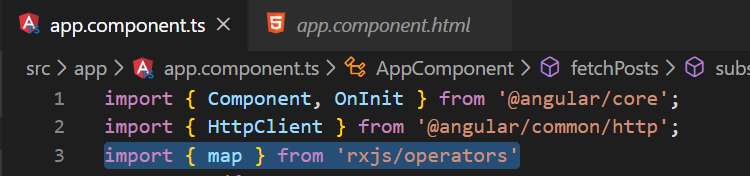
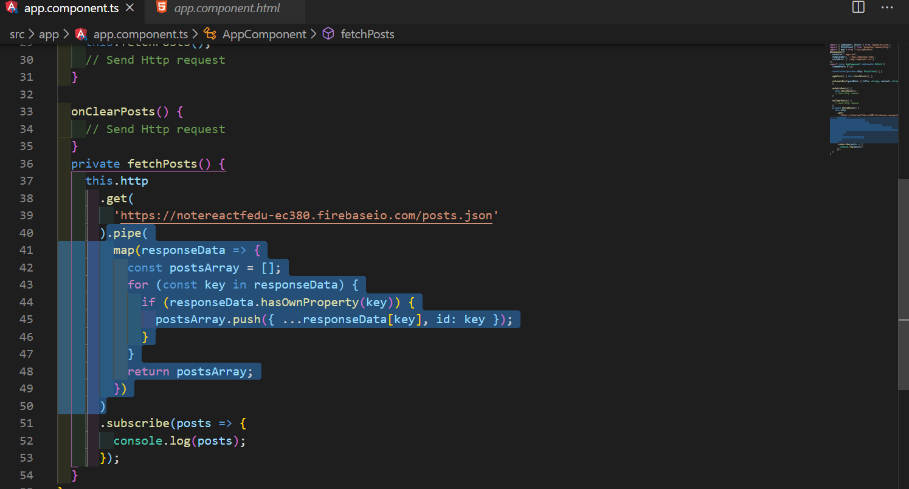
Argument đầu: đường dẫn firebase + các endpoints.json

Argument thứ 2: là Request body, nghĩa là dâta gửi về Database

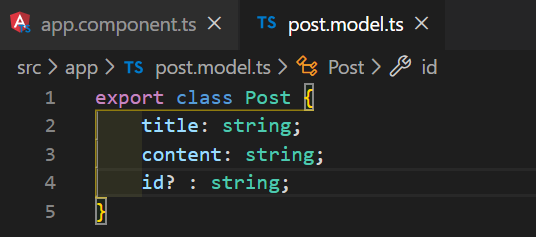
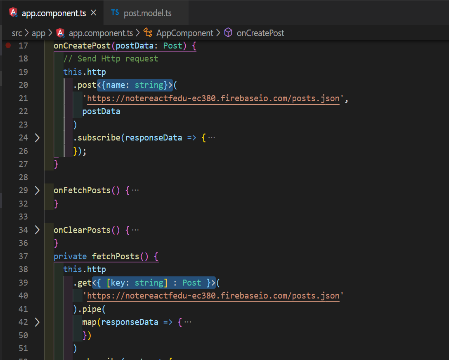
Vậy là đã có thể gửi Post request lên server và thêm data vào firebase.

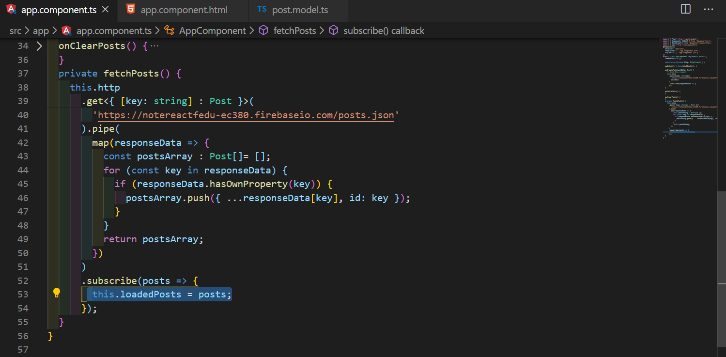
* 1. **GETting Data:**
     1. 

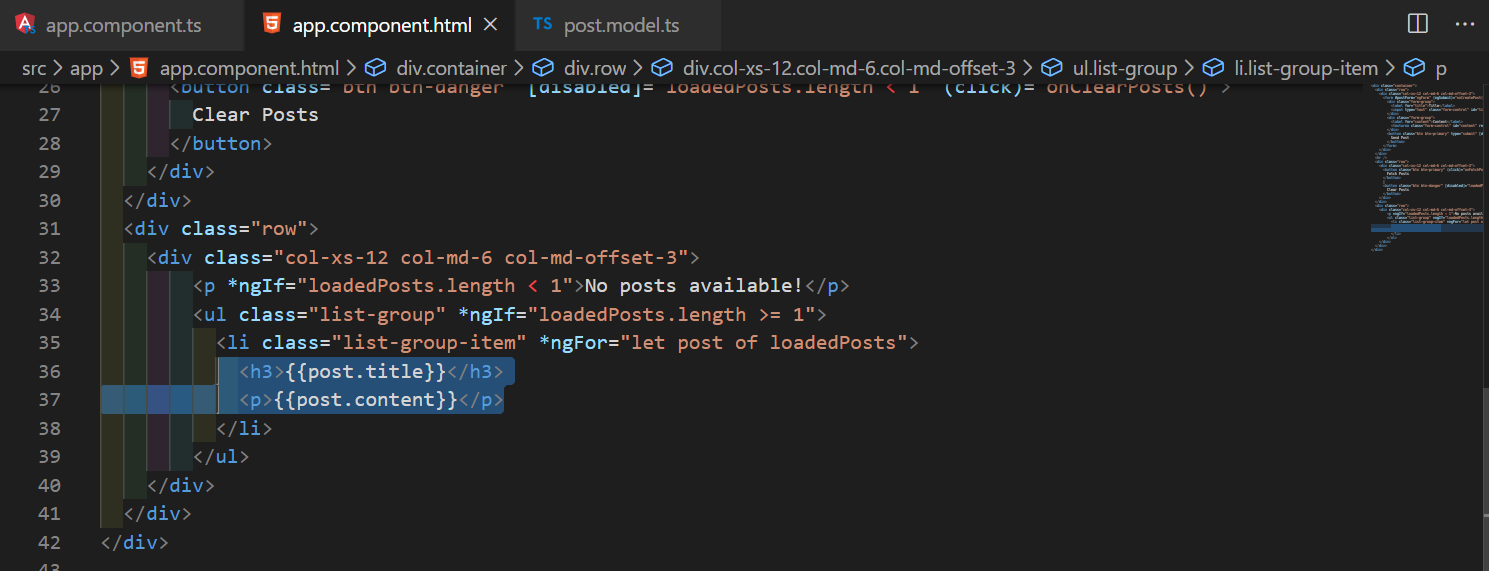
Làm giống như POST Request nhưng thay ‘.post’ thành ‘.get’, chỉ nhận vào 1 argument là đường link firebase, không cần truyền thêm data.

* 1. Using RxJs Operations to Transform Response Data: Nhận về được data nhưng kiểu dữ liệu ấy không phù hợp nên cần phải transform nó thành 1 mảng các Object:
     1. 
     2. Tìm hiểu thêm hoặc xem lại clip 256 để hiểu rõ hơn.
  2. **Using Types with the HttpClient**: Vấn đề là mình chưa thể quy định các kiểu dữ liệu cuả data khi gửi đi (post ) hoặc nhận về (get).

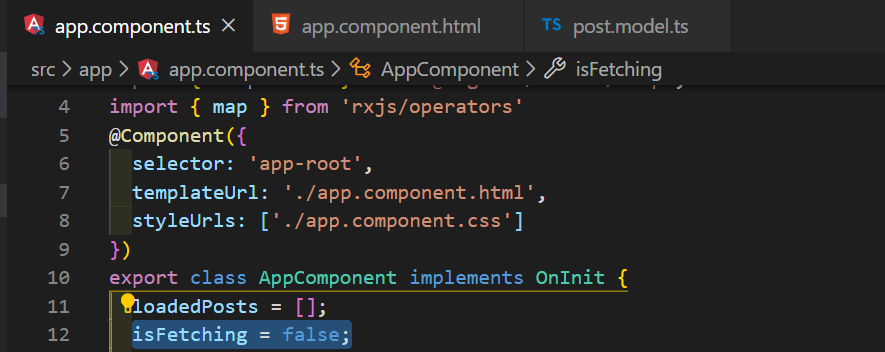
**Giai quyết**:

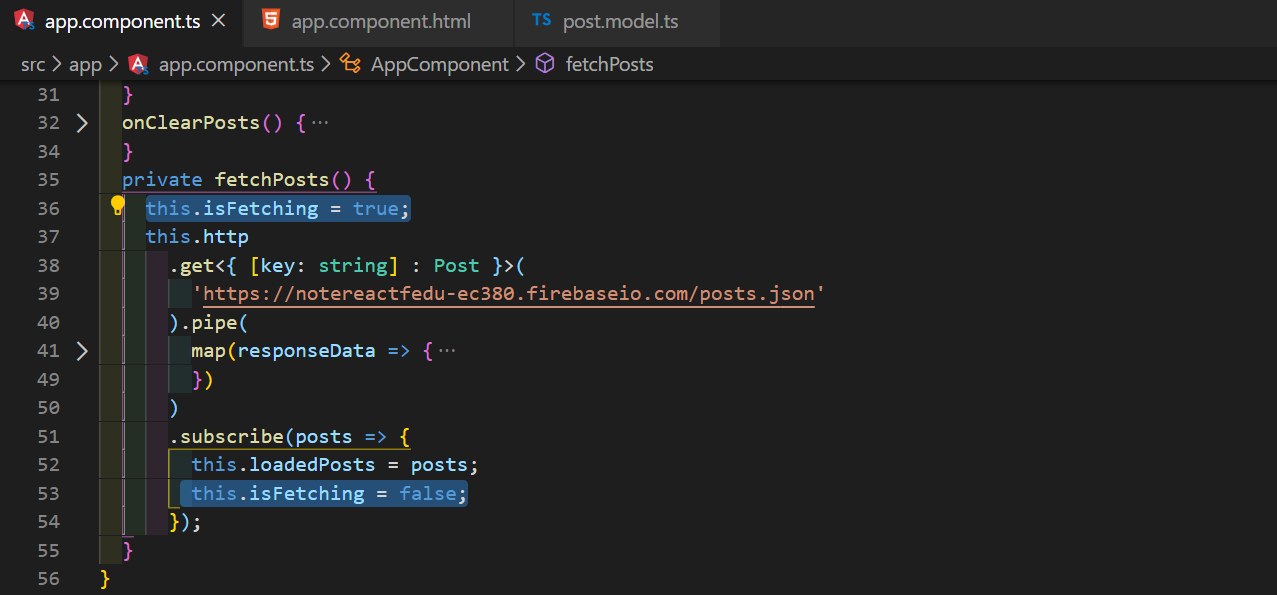
* + - Tạo 1 file model kiểu Post :
    - Chèn thêm kiểu dữ liệu trong <> sau hàm .get, .post: 
  1. **Outputing Data:** Xuất các post mới lấy về từ firebase ra màn hình.Làm tương tự giống như ta lấy được data từ Service thui. Khi ta lấy được các posts trong hàm .subscribe() => gán posts[] cho property loadedPosts[]

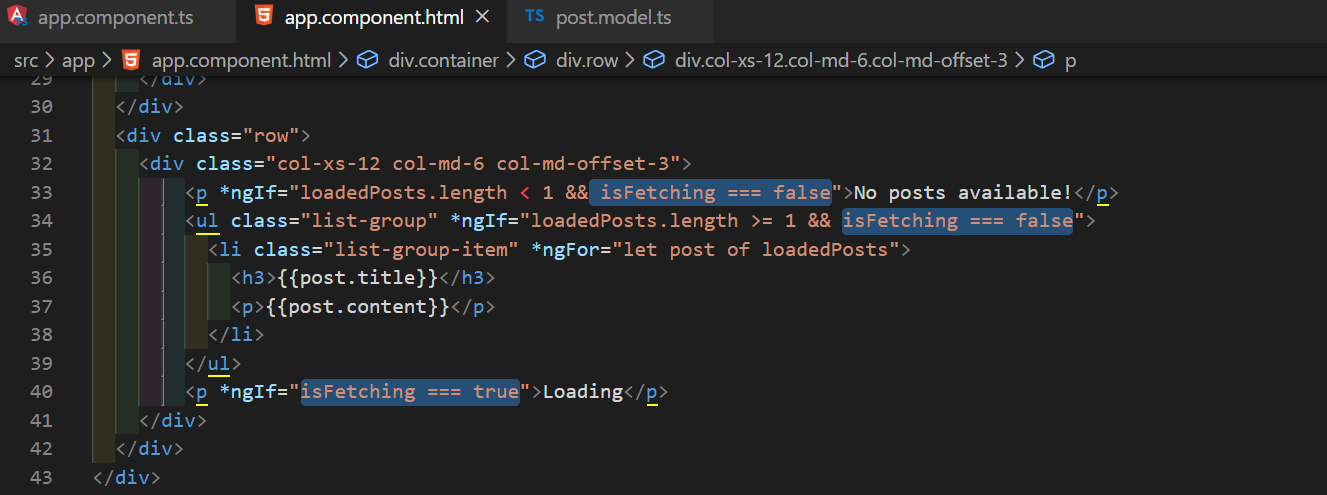


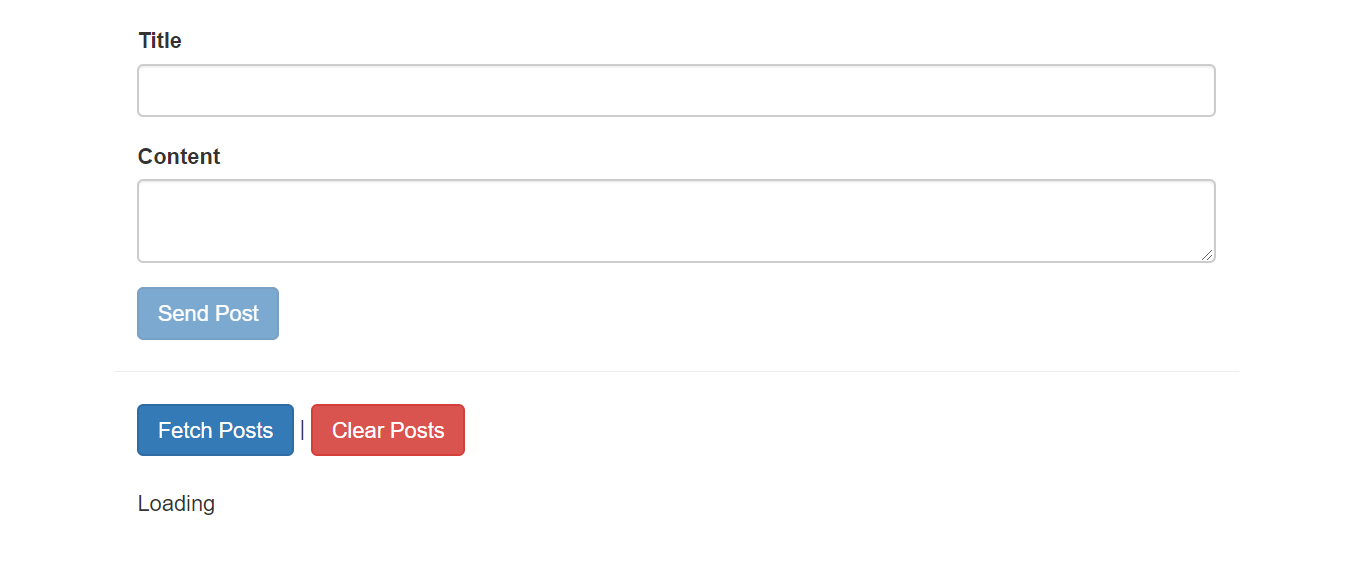


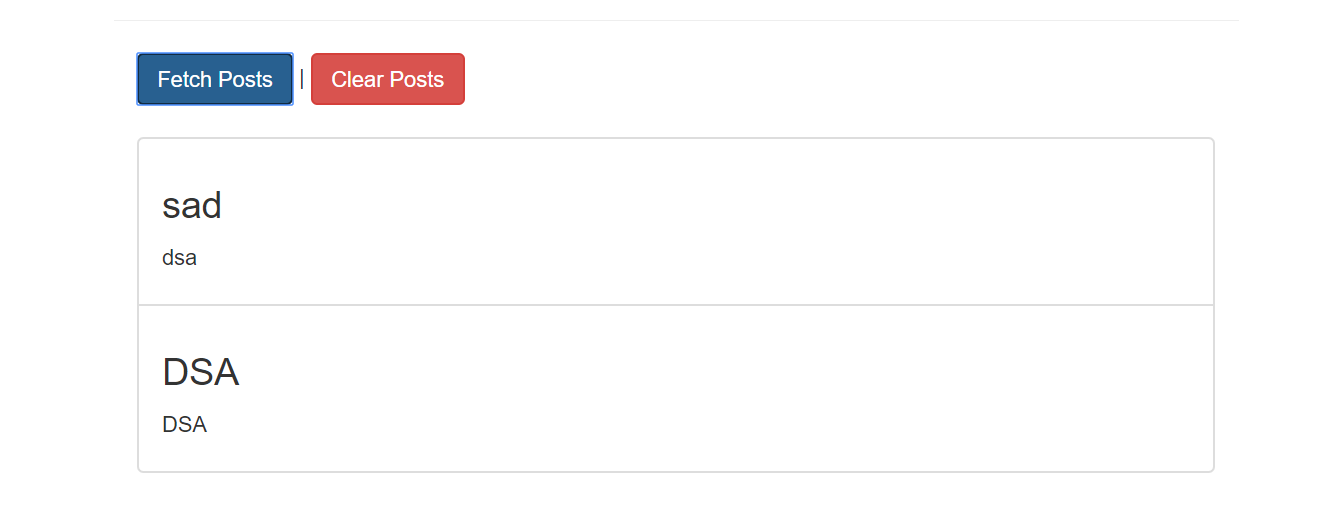
* 1. **Showing a Loading Indicator:** Thông báo là đang loading data hay loaded data rồi

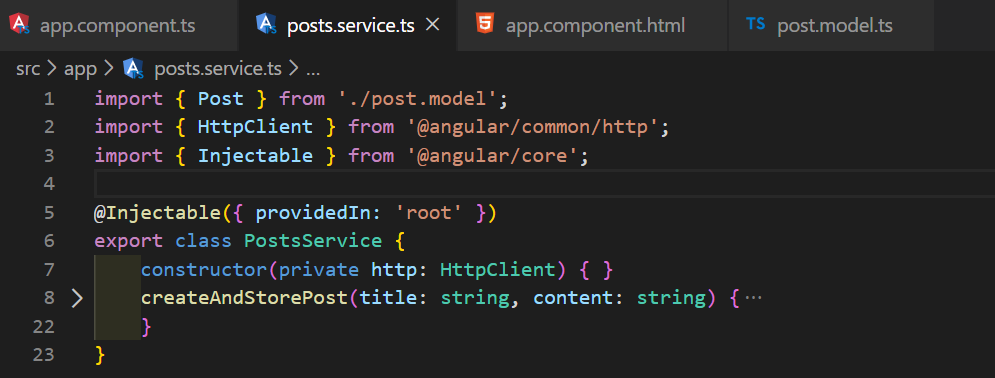
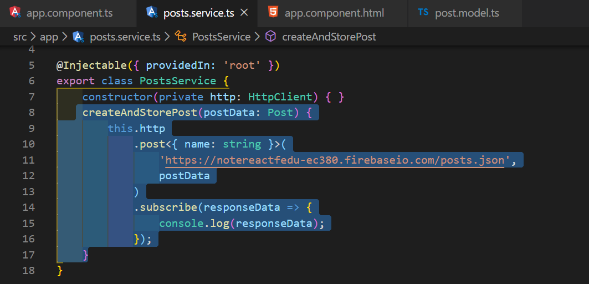
****

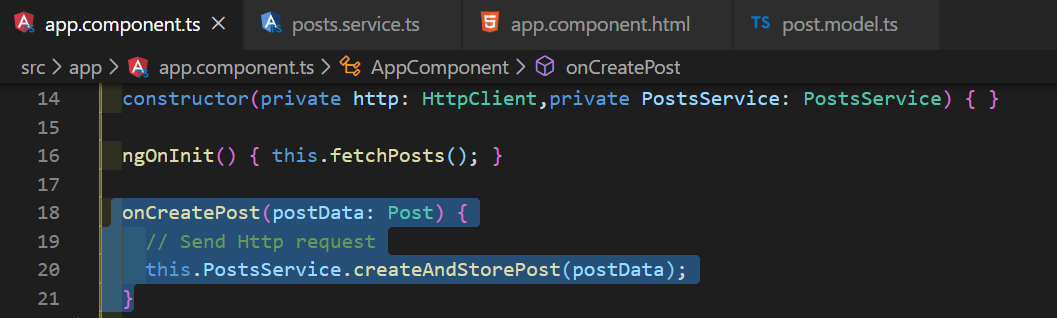
****

****

**…loading**

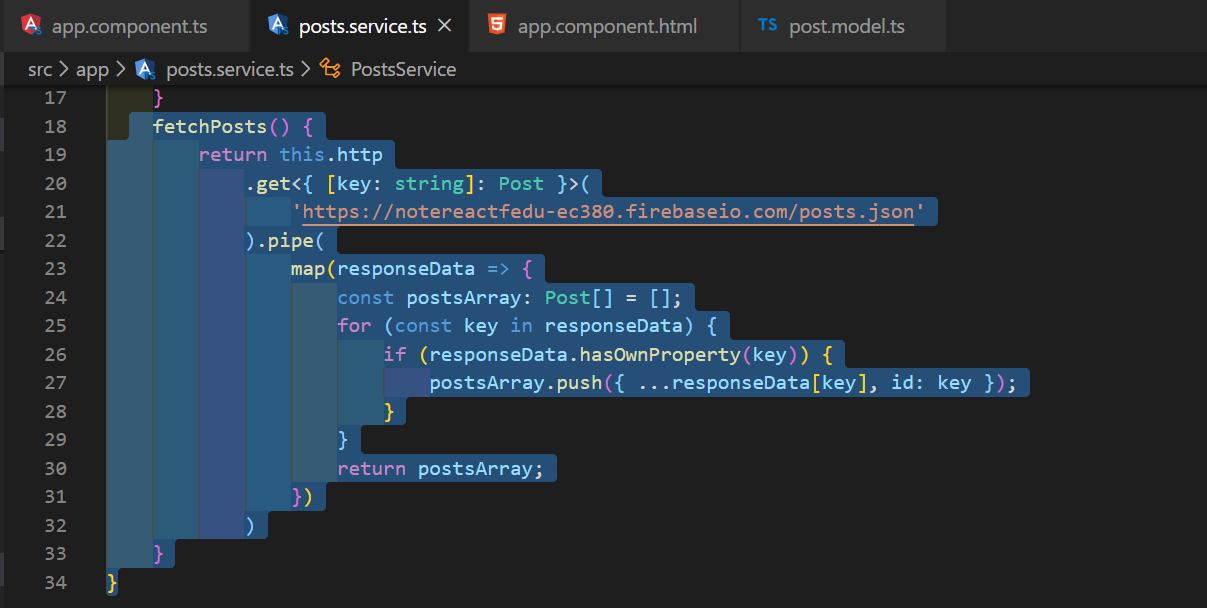
…**loaded**

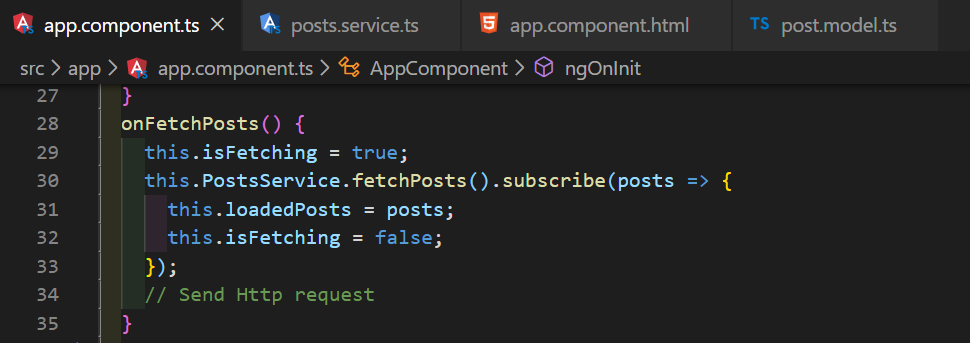
* 1. **Using a Service for Http Requests (Outsource into Services)**
     1. Set up posts.service.ts: 
     2. **Tạo hàm CreateAndStore để Post Requests lên firebase:** Định nghĩa trong PostService.

Gọi hàm trong App để post request khi xử lý sự kiện Click.

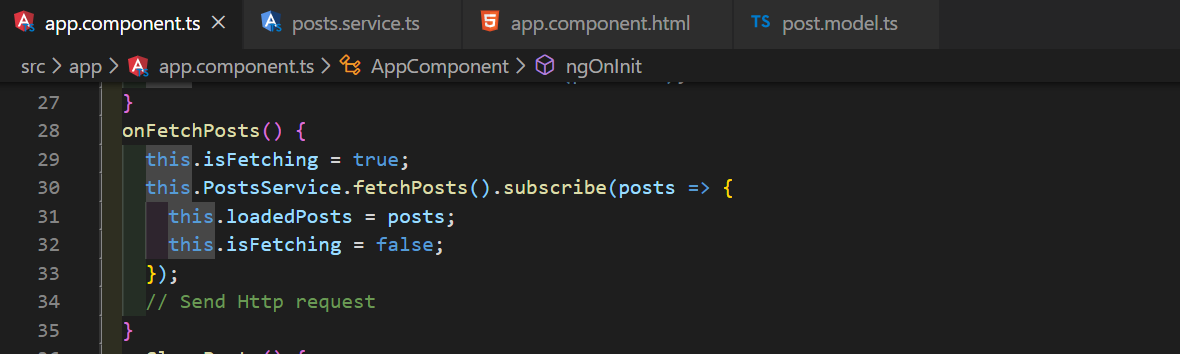
* + 1. **Tạo hàm onFetchPosts() để gửi Get Request lên firebase để lấy data về:**

**Cách Chuẩn:** Không dùng hàm .subscribe() trong service vì sẽ bị lỗi, xóa hàm .subscribe() => return về 1 Observation. Sau đó khi nào ta gọi hàm onFetchPosts() của service thì mới chèn thêm hàm .subcribe() để xử lý dữ liệu được trả về (response data).

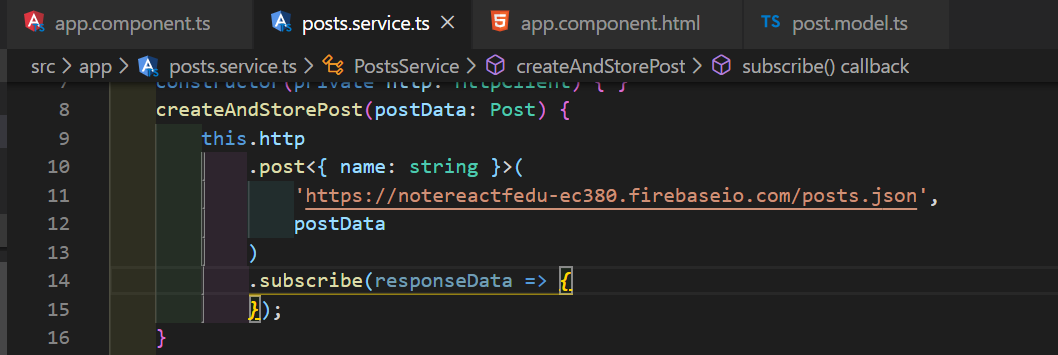
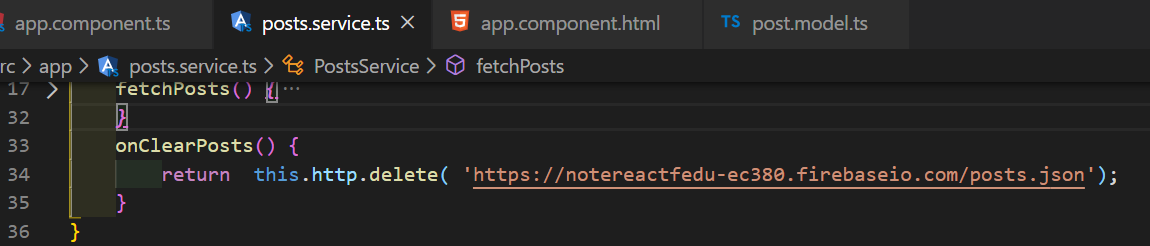
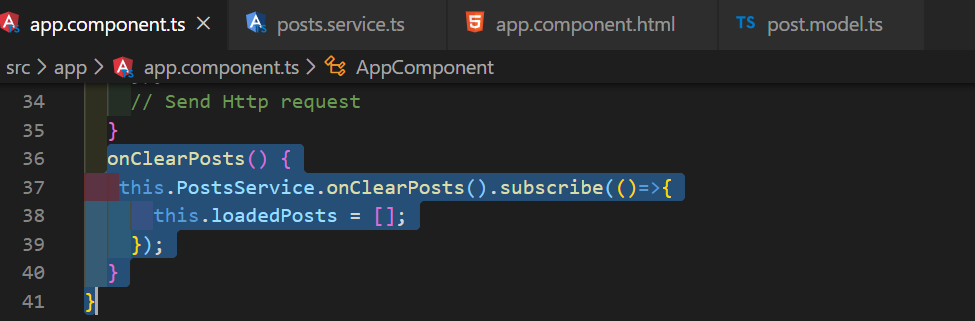
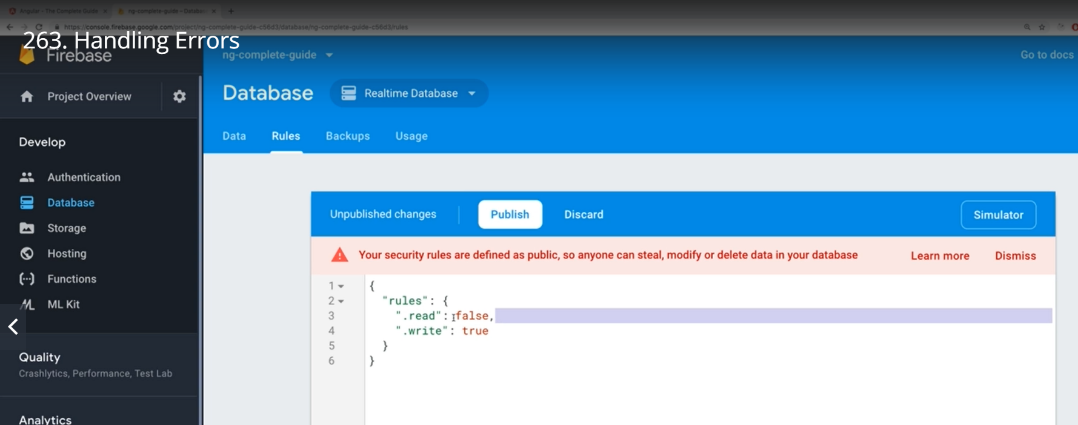
Chỉ trả về 1 observation, không có hàm .subscribe().

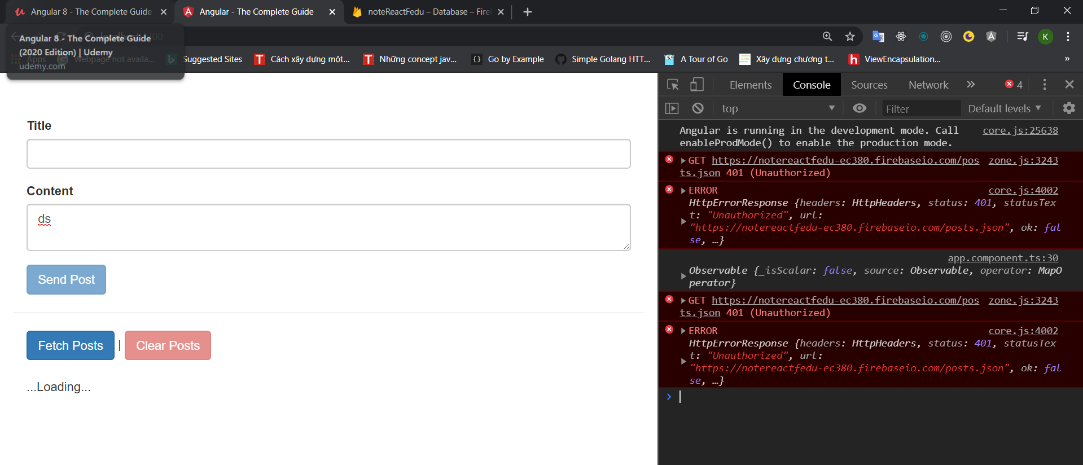
gọi method của postService, sau đó gọi thêm hàm Subcribe() để xử lý các data vừa lấy về

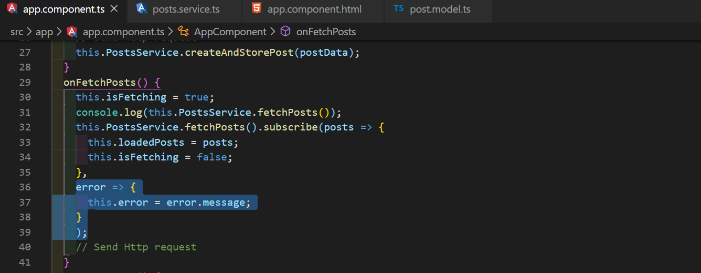
**Phân biệt**: Khi nào nên tách hàm .subscribe() ra khỏi service, mà chỉ return về 1 observation (Như hình 1) thui ?

* + - Khi nào cần theo dõi các dữ liệu, các status lấy về từ Response về để cập nhật vào các Property trong Component đó, trong Template đó, ảnh hưởng đến Output của Component đó
    - Ví dụ:
    - do phải gán, cập nhật property ‘This.loadedPost’ thông qua ‘Posts’ nên ta phải tách hàm .subscribe() ra khỏi service, theo dõi ở Component.

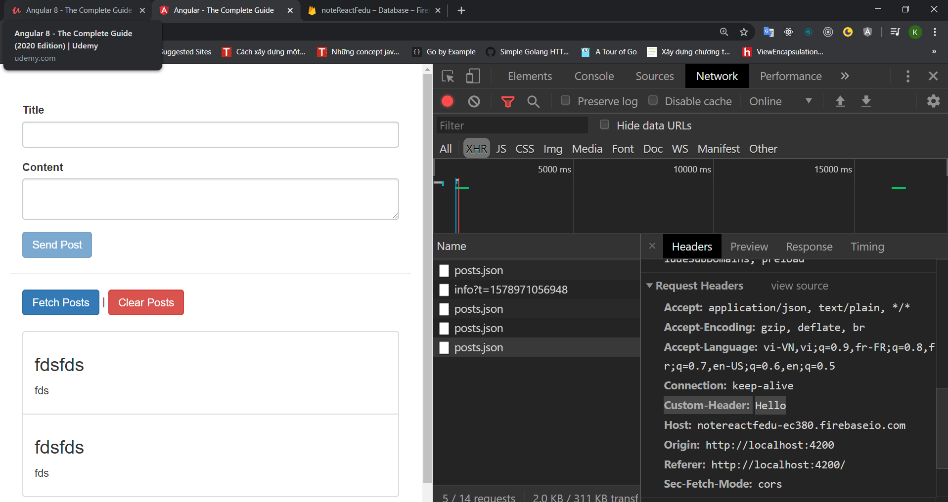
**Phân biệt**: Khi nào nên để hàm .subscribe() trong service (Như hình 2.10.2)?

* + - Khi Component đó không quan tâm về các response and không quan tâm rằng request có được thực thi hay không (như hàm CreateAndStorePost()).
    - Nếu Component đó không quan tâm thì ko có lý do gì để subscribe() component đó cả.
    - Ví dụ:
    - 
  1. **Sending a DELETE request**
     1. Tạo 1 hàm OnClearPost() trong service: 
     2. Gọi hàm OnClearPosts() từ PostsService và đồng nhất các Posts với Firebase, reset lại this.loadedPosts: 
  2. **Handling Errors**: Nếu ở Server hoặc database bị lỗi gì đó, không thể nhận được response, ta phải có hướng giải quyết khi xảy ra trường hợp này:
     1. Gỉa sử Firebase không cho Đọc dữ liệu: Không cho đọc dữ liệu

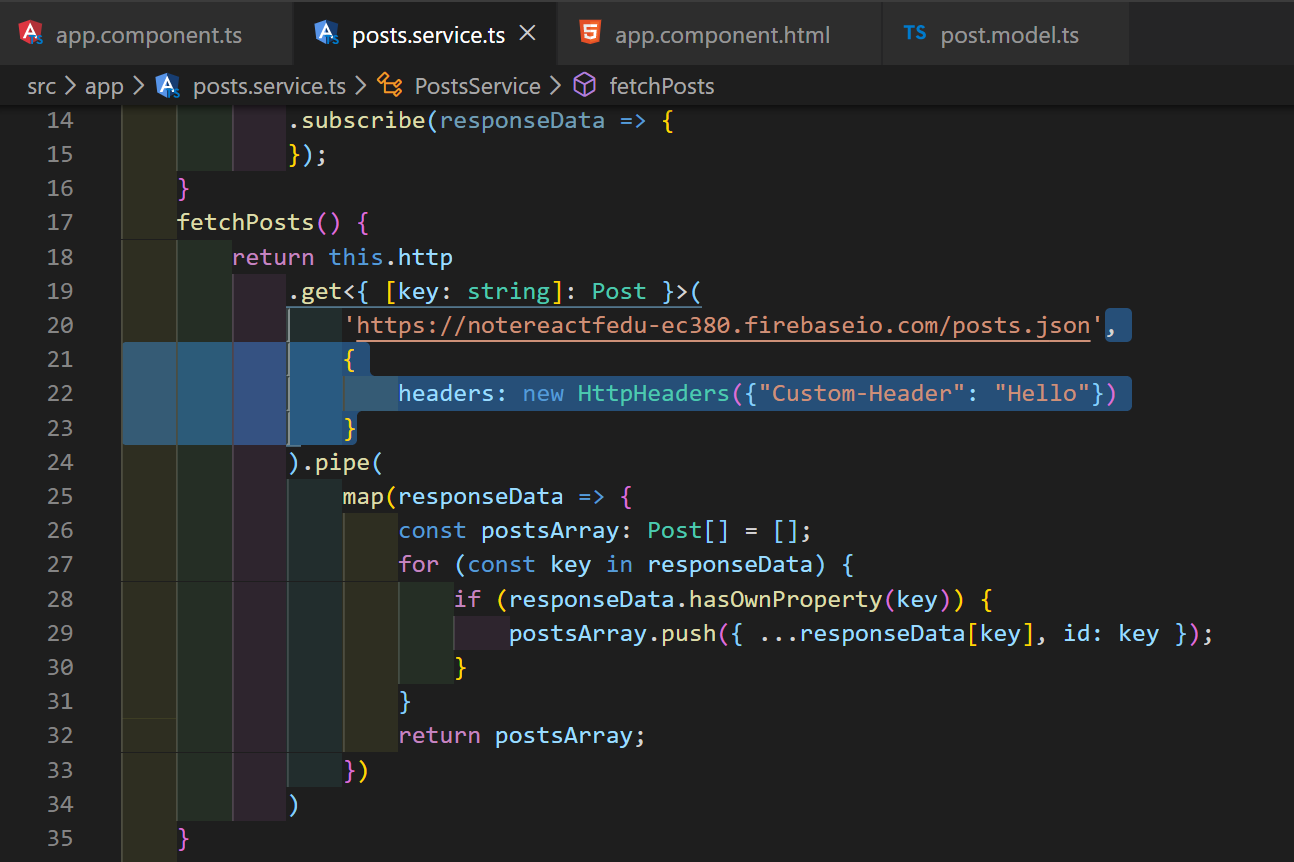
Bị lỗi khi ta Fetch dữ liệu, không đọc được dữ liệu.

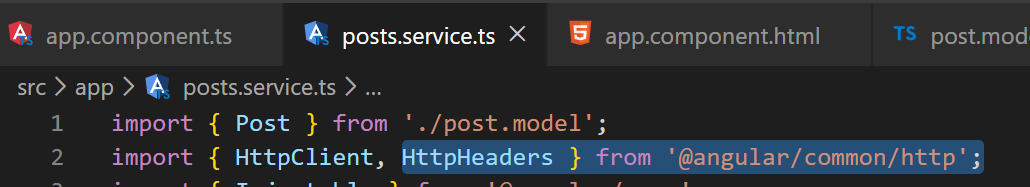
**Cách 1:**Sử dụng thêm 1 argument Error ở trong hàm .subcribe() để theo dõi mỗi khi có lỗi, ta cũng xử lý vấn đề trong hàm ấy: 

**Cách 2:** clip 264 coi lại khó hiểu

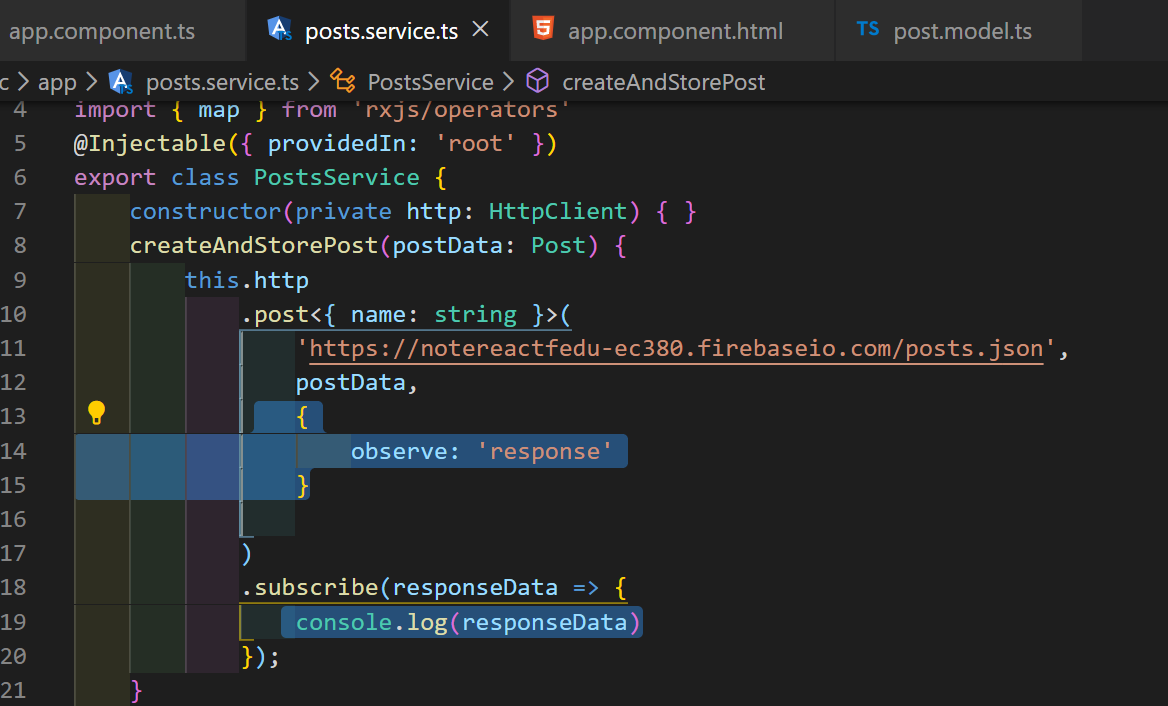
* 1. **Setting Headers:** Nghĩa là tạo ra 1 cặp Key-Value trong Headers: ****

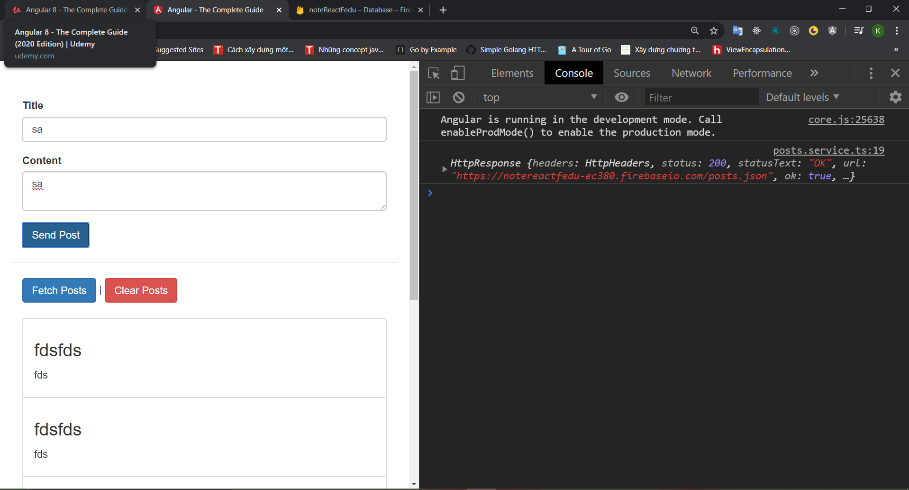
Cách làm:

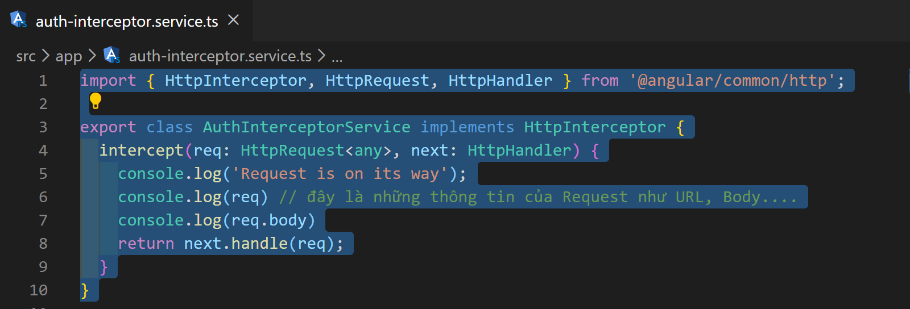
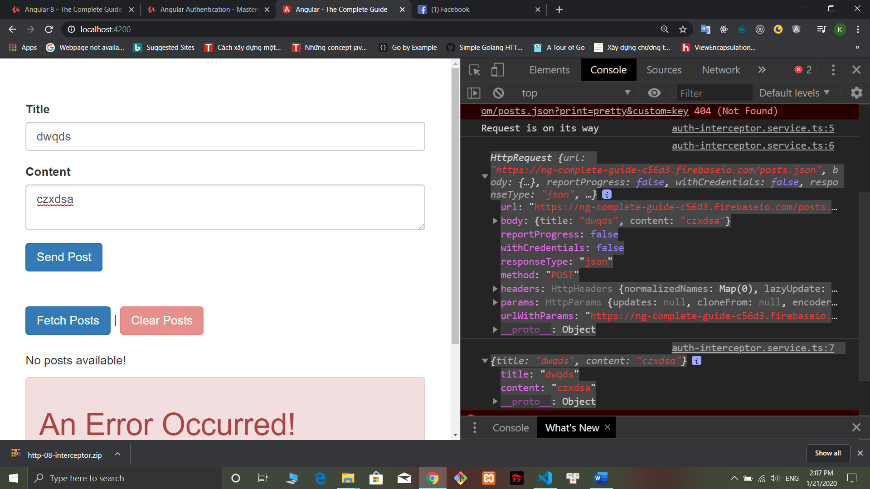
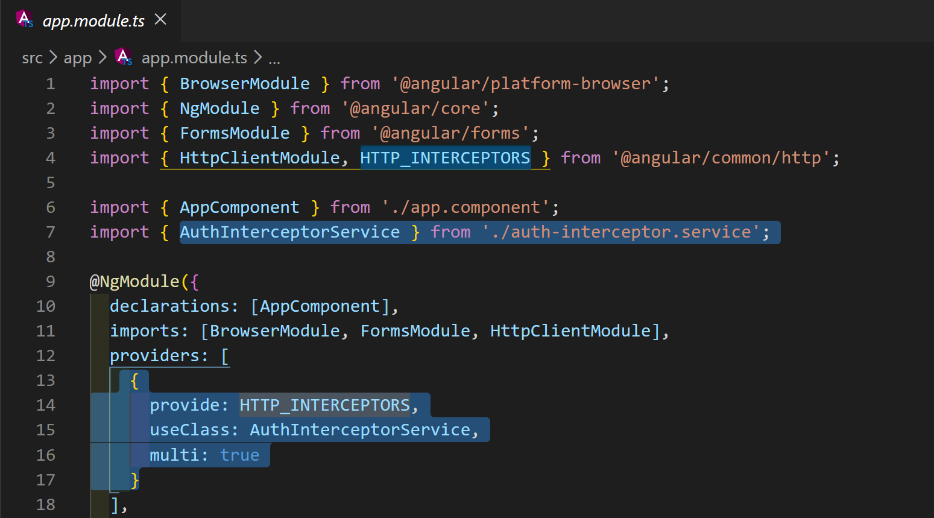
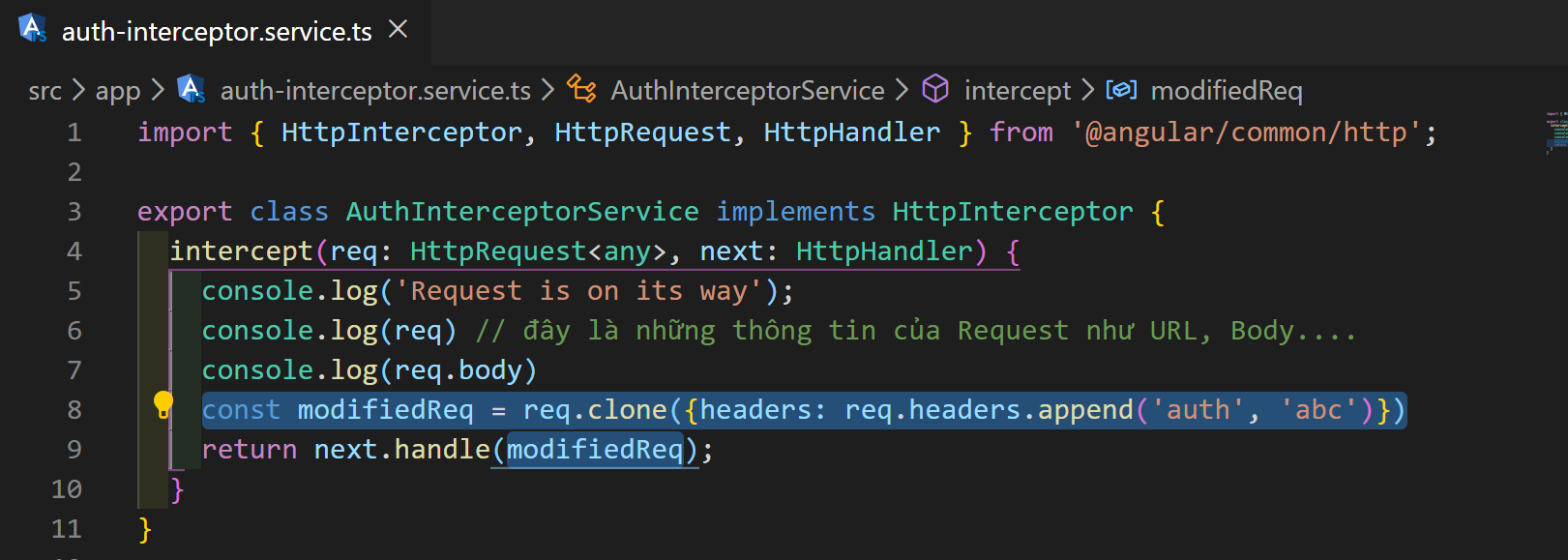
**,** thêm cái dòng highlight vào.

, Nhớ Import vào.

* 1. **Observing Different Types of Response:** lấy về toàn bộ Response Object nhờ Observe object
     1. **Á**

Line 13, 14, 15 là syntax, Muốn access vào response thì xuống hàm Subcribe().

Dòng Consosole của hàm Subcribe(), truy cập vào Response mà mình get back được.

* 1. **Introducing Interceptors:** 
     1. Mục đích: dùng để chạy 1 số code ngay trước khi gửi 1 Request nào đó về server. (giả sử lấy token để xác thực user trước khi gửi request về server để thực hiện 1 event nào đó)
     2.  -Tạo file authInterceptor.Service.ts, method Intercept() sẽ chạy trước khi Request được gửi về server(before the request leave the application)- Đây là các thông tin của req khi gửi Request về server, có body, có url, có method….
     3.  -Setting Interceptors để sử dụng. HOÀN TẤT.
  2. **Manipulating Request Objects:** Vận dụng, thay đổi, modify the Request Object trong Interceptor
     1.  - Gửi về Server 1 cái request đã bị thay đổi so với request ban đầu, modify lại property headers (thêm auth: abc)