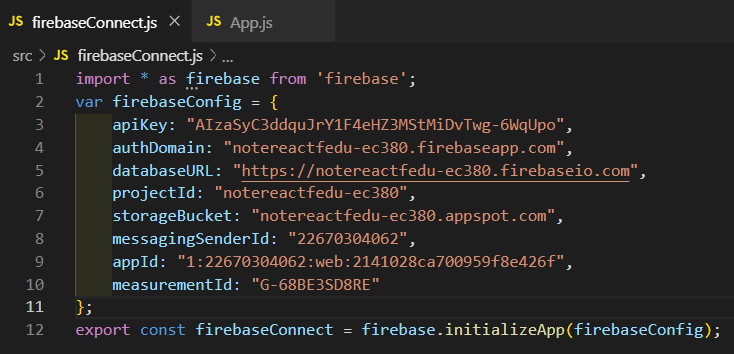
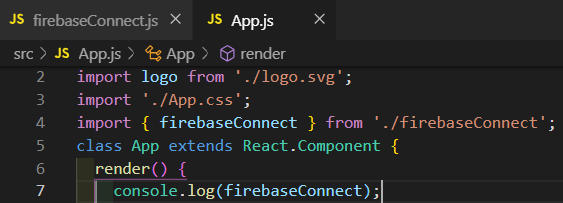
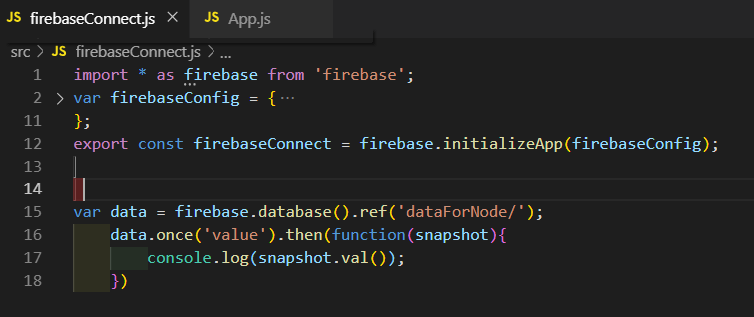
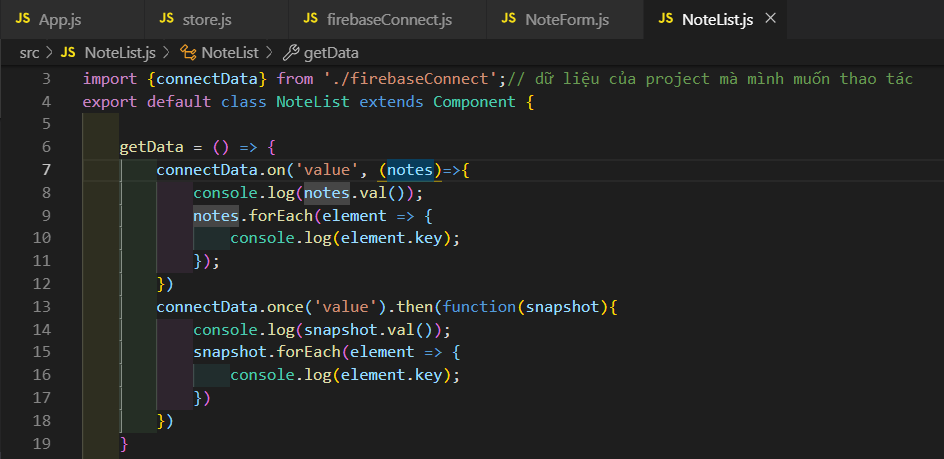
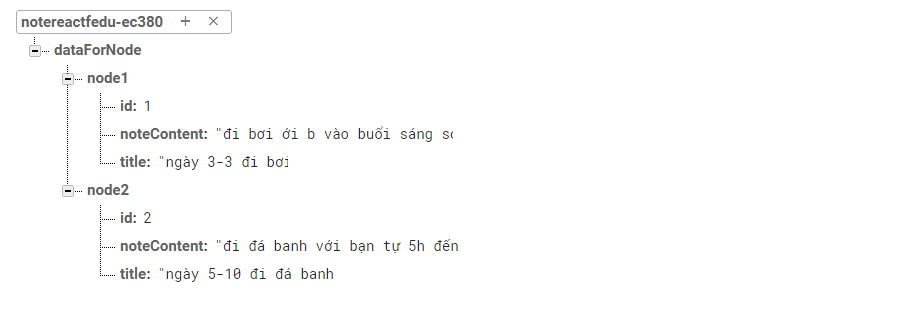
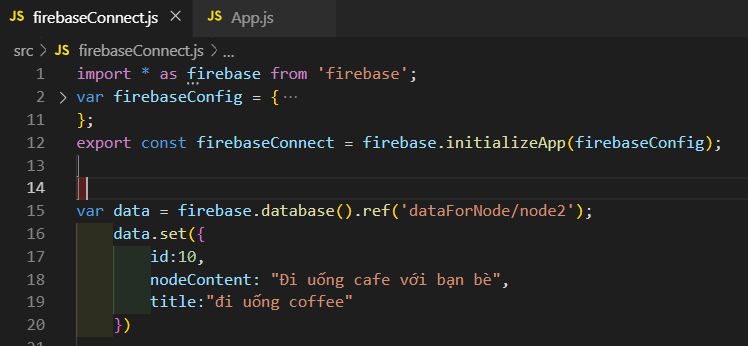
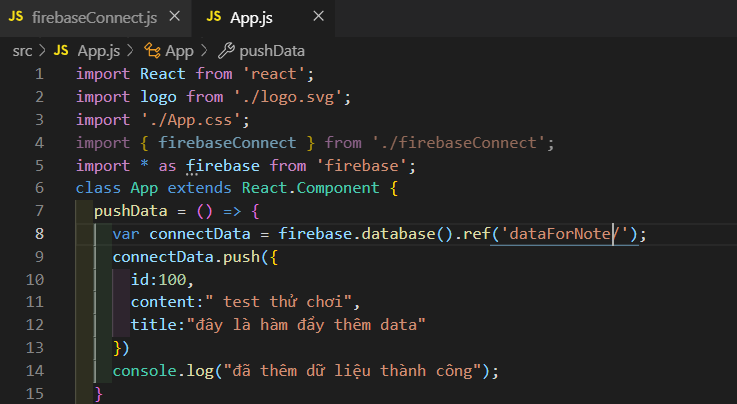
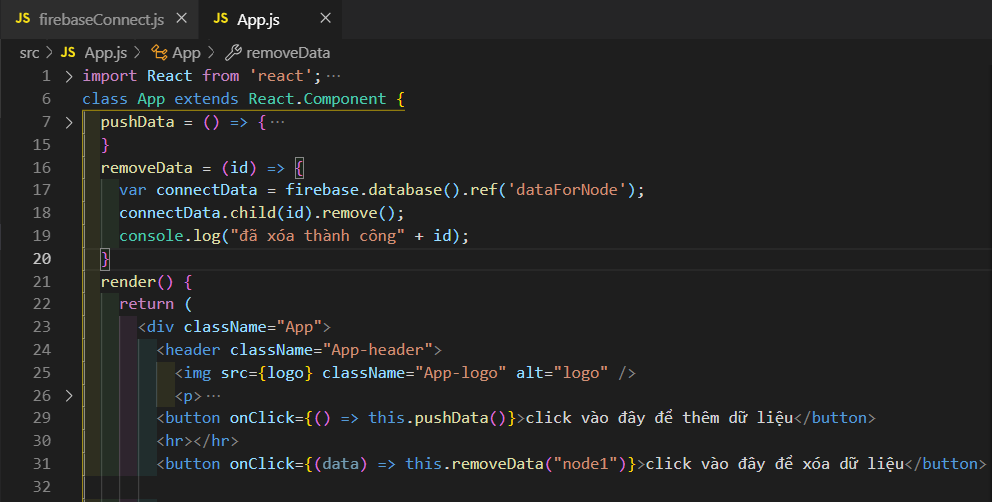
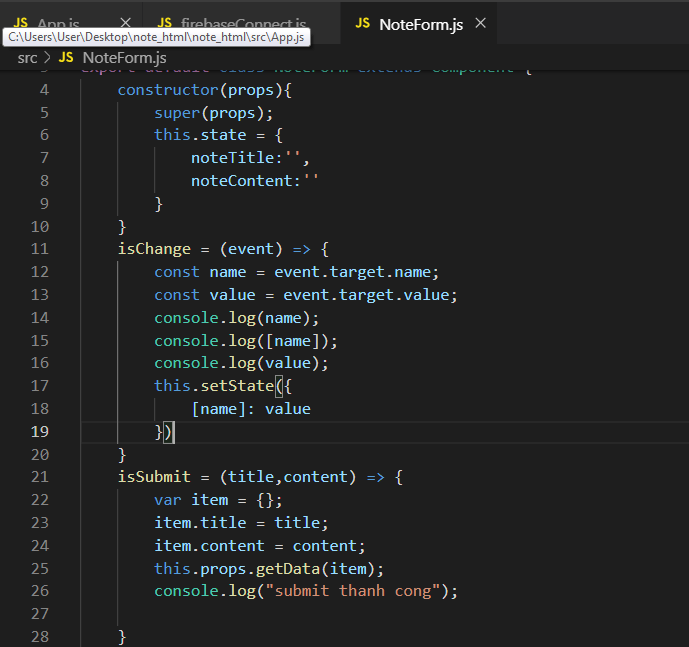
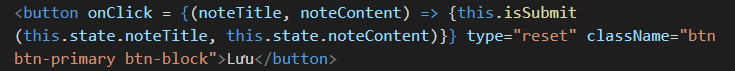
sCÁC CHỨC NĂNG:

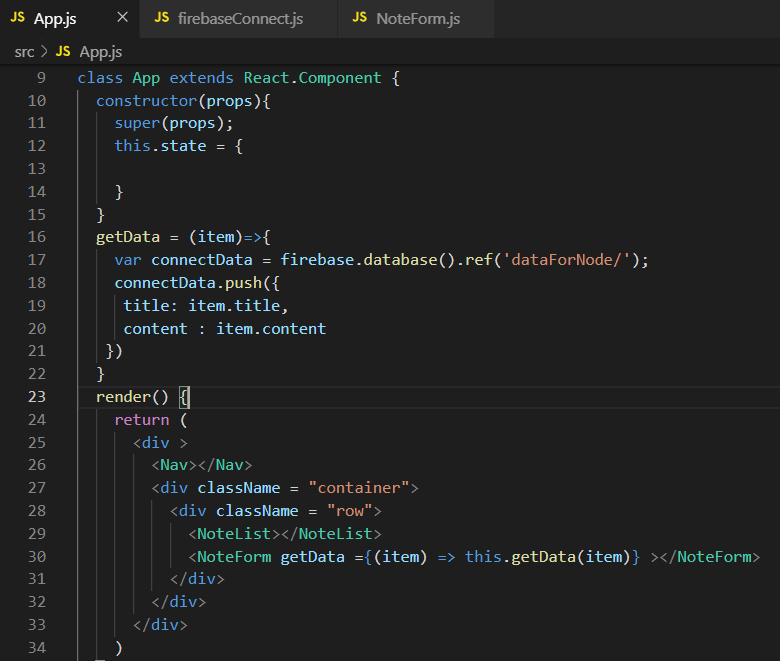
* Thêm, xóa, sửa,xem.

**Tạo giao diện úng dụng .**

* 1.Tạo giao diện html.
* 2.Convert sang Component.
* 3.Tạo Store và kết nối.
* 4.Viết logic xử lý từng chức năng.

**Cơ sở dữ liệu FIREBASE:**

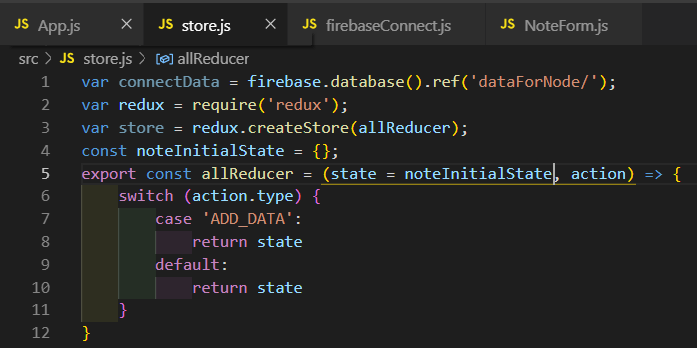
* WHY:
  + Tối đa 5G.
  + Realtime.
  + Cơ sở dữ liệu trực tuyến => Không cần cài đặt nhu SQL, MySQL.
  + Cũng không cần API
  + ReactJs giao tiếp trực tiếp với firebase
  + Chỉ dùng cho các web, app nhỏ, các web lớn có nhiều chức năng thì không làm được
* HOW: <https://firebase.google.com/>
  + Tạo project:
    - <https://console.firebase.google.com/u/0/?hl=en>
    - Realtime
    - Export JSON
  + Npm install firebase
  + Các lệnh cơ bản của firebase:
    - Thêm mới dữ liệu thông qua reactJs(trực tiếp)
    - Lấy dữ liệu
    - Sửa dữ liệu
    - Xóa dữ liệu
* ***KẾT NỐI REACTJS – FIREBASE***:
  + Tạo CSDL trên firebase.google.com và copy mã script.
  + Tạo app reactjs.
  + Cài đặt modult ‘npm install firebase –save’.
  + Tạo file ‘firebaseConnect.js’.
  + Trong file ‘firebaseConnect.js’, copy và past đoạn script trên firebase.google.com:
    - 
  + Ở ‘App.js’:
    - Import Firebase:
    - 
* ***CÁC CHỨC NĂNG CƠ BẢN:*** lấy, sửa, thêm, xóa….
  + **LẤY DỰ LIỆU** CỦA BẢNG DỮ LIỆU BẰNG ‘DataSnapShot’:
    - Dữ liệu của firebase được lưu trong phần tử gọi là dataSnapShot(ảnh chụp dữ liệu)
    - Do đó, đầu tiên phải khai báo đối tượng dataSnapShot trước:
      * 
      * HOẶC DÙNG HÀM on(): đây là kiểu callBack
      * 
      * Cả 2 cách trên điều được nhương On là callback, Once() là promise.
    - 
  + **SỬA DỮ LIỆU** TRONG FIREBASE: dùng hàm set
    - 
  + **THÊM DỮ LIỆU** TRONG FIREBASE:
    - Gồm 2 cách: hàm set() như trên, hoặc dùng hàm push().
    - Hàm Set() thì phải tự đặt id, nó ko hỗ trợ tự tạo id.
    - Hàm Push() hỗ trợ tự đặt id (id do firebase cung cấp) nên tiện dụng hơn:
      * 
  + **XÓA DỮ LIỆU** TRONG FIREBASE:
    - Dùng hàm .child(data).remove():
    - 
* **THỰC HÀNH ỨNG DỤNG GHI CHÚ:**
  + Chức năng **THÊM MỚI** vào cơ sở dữ liệu:
    - Gồm 3 bước:
      * 1.Kết nối CSDL(làm như trên).
      * 2.Click nút “lưu” thì lấy thông tin của dữ liệu vào:
        + 
        + 
        + 
      * 3.Lấy được dữ liệu thì update vào CSDL, gồm 2 cach:
        + Không sử dụng redux (reactjs thuần)



* + - * + Sử dụng redux:



1. Tạo Store (file ‘store.js’):Cài đặt redux và khởi tạo reducer

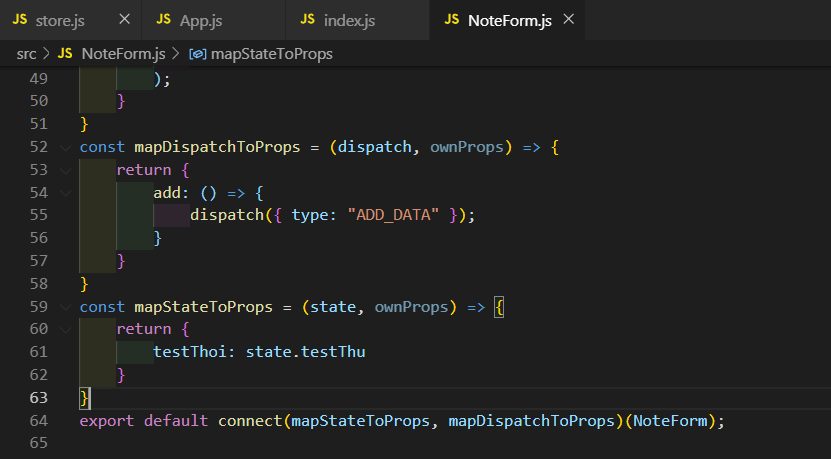


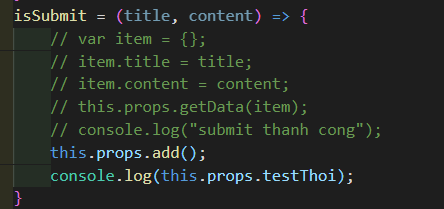
2.Kết nối Store với component <NoteForm>:

Gồm 3 bước cơ bản:

Thực hiện kết nối Store với component như mình đã học.

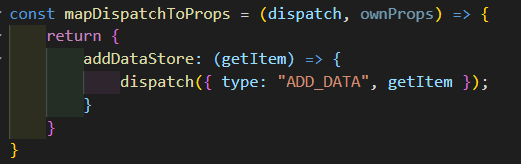


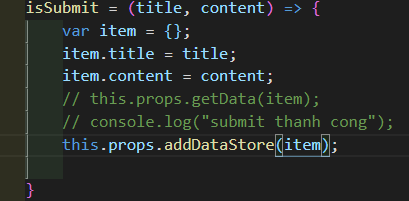


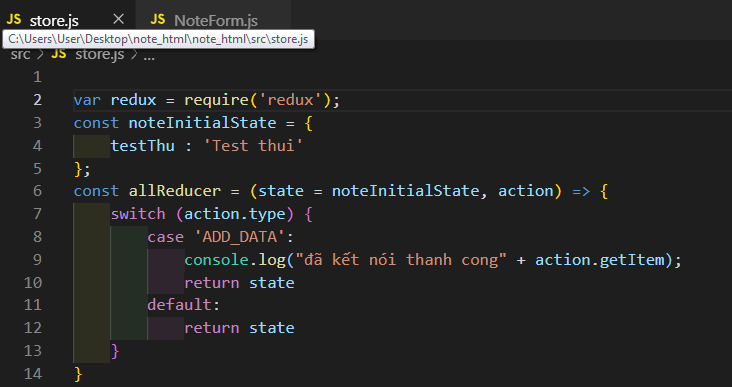


Truyền tham số vào Store và xử lý tham số ấy trong store:

Truyền tham số vào các Reducer(các chỉ thị, thực thi) ở trong store với cú pháp thực thi là: action.tenThamSo .

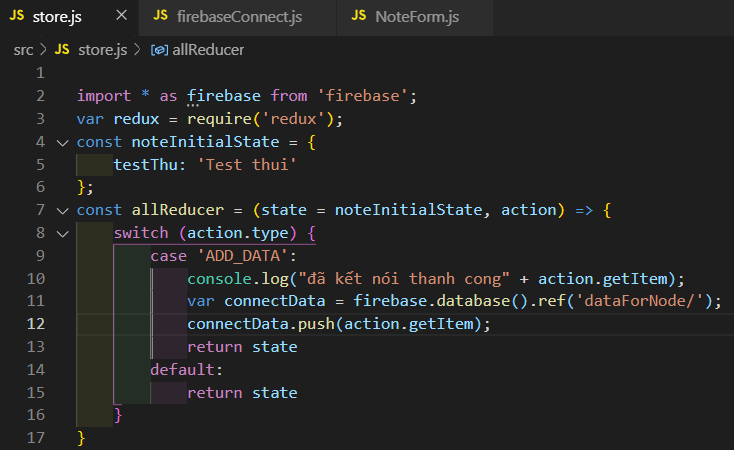


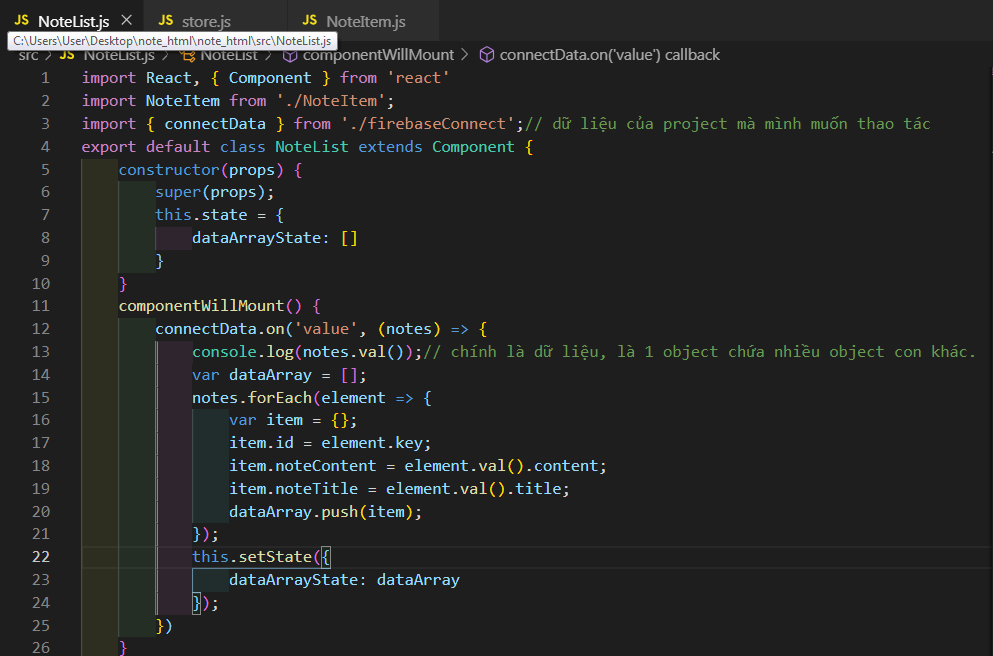


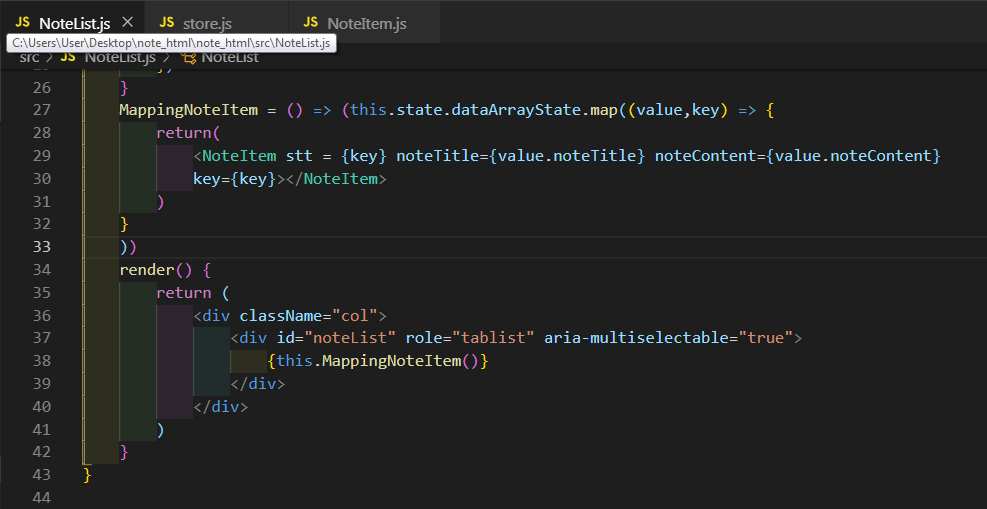


Hoàn chỉnh chức năng THÊM MỚI data:

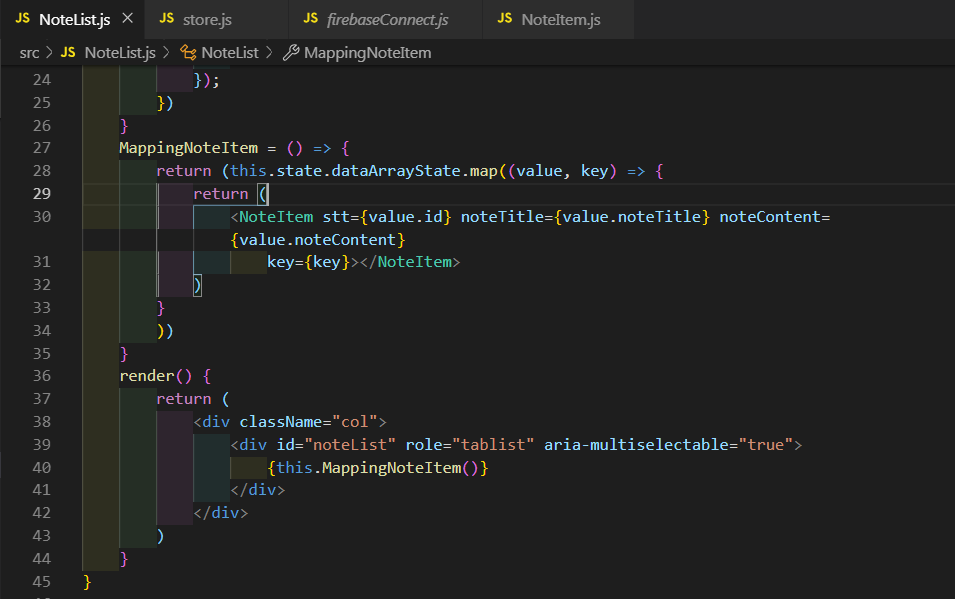
Sau khi store nhận được dữ liệu thì dùng hàm Push() đẩy item lên CSDL Firebase



* + Chức năng LẤY DỮ LIỆU từ CSDL và IN DỮ LIỆU ra ở <NoteList>:
    - Gồm 2 bước :
      * 1.Lấy dữ liệu từ FIREBASE:
        + 
        + ‘notes’ chính là dữ liệu data trên firebase.
        + notes.val() là 1 object chứa nhiều object con khác, vì thế, cần phải biến đổi (convert) object lớn ấy thành các object item con bằng cách trên.
        + Tạo 1 cái mảng ‘dataArray’ và đẩy các đối tượng trên vào mảng bằng hàm .push().
        + Cập nhật lại State bằng mảng vừa tạo.
      * 2.In dữ liệu ra bằng hàm Foreach() hoặc map():
        + Map():



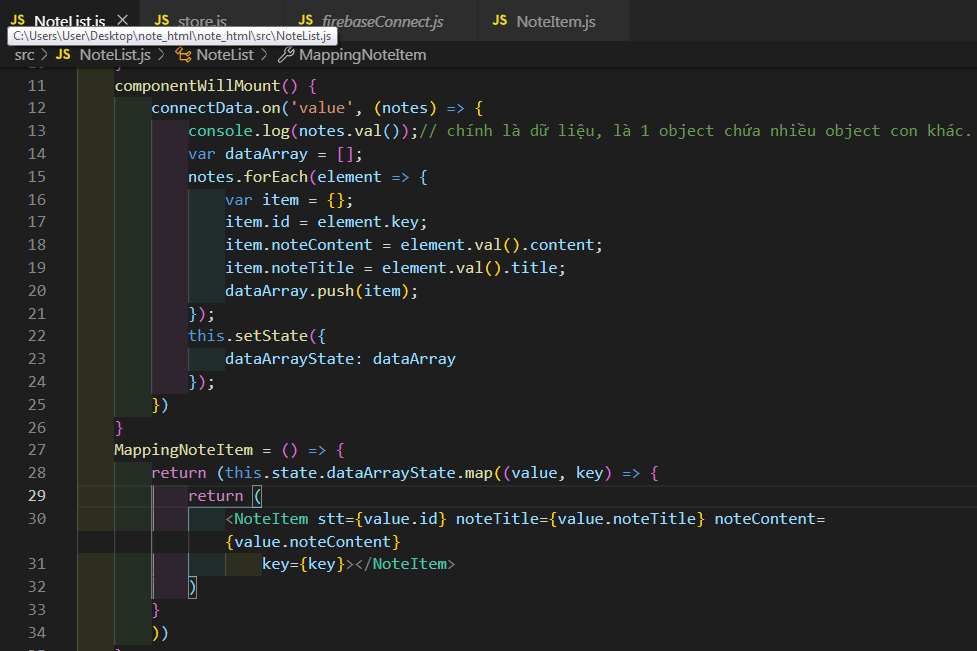
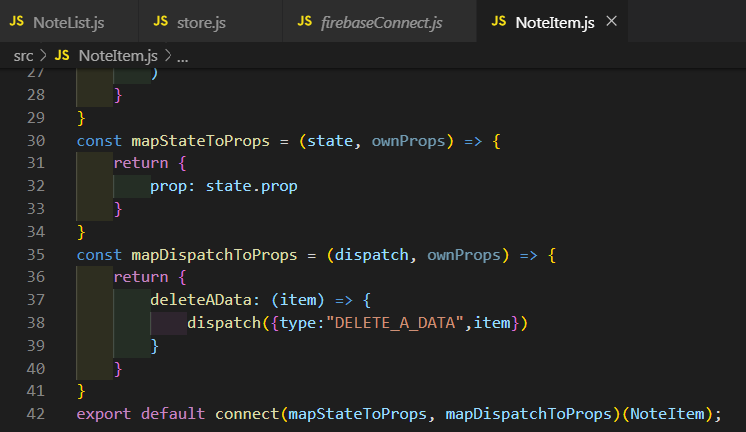
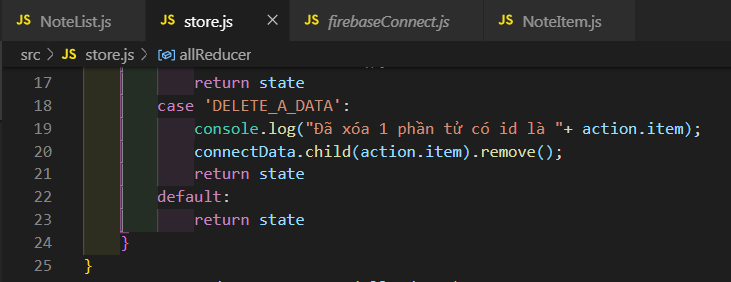
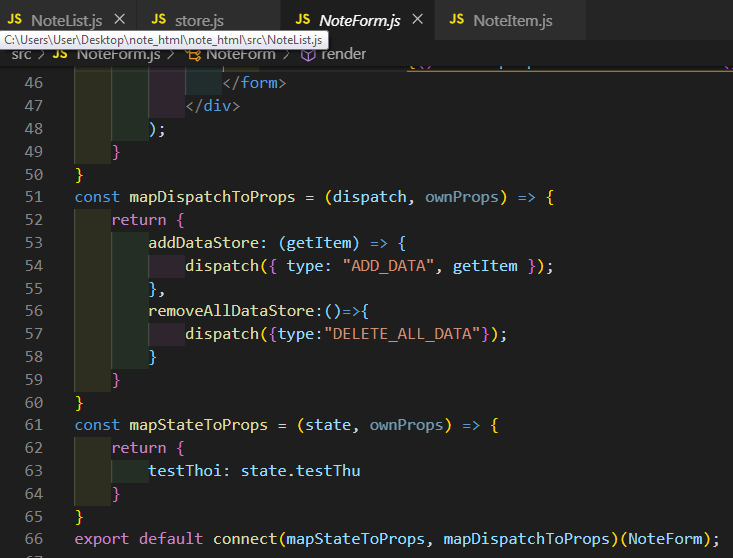
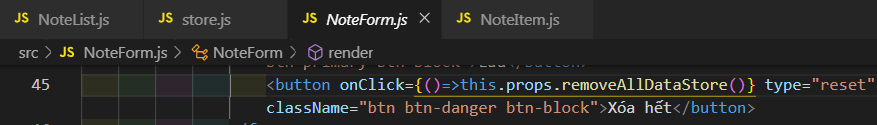
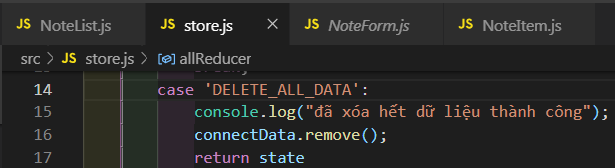
Hoặc:



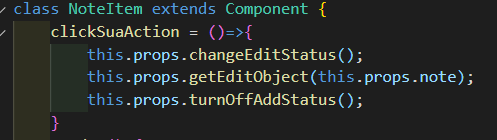
* + - * + ForEach(): Thực hiện không được vì:

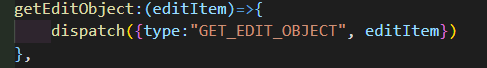
ForEach() không thể trả về (return) bất kỳ giá trị nào. Bất kỳ giá trị nào trả về đều bị loại bỏ (undefined).

<https://viblo.asia/p/so-sanh-foreach-va-map-trong-javascript-L4x5x1YBKBM>

* + Chức năng **XÓA DỮ LIỆU** ở CSDL: xóa 1 dữ liệu và xóa hết dữ liệu:
    - **XÓA 1 NOTE:**
      * ****
      * **Tại <NoteList>, ta lấy** được ‘id’ của từng Note bằng ‘element.key’.
      * **Truyền** ‘id’ của các NOTE xuống <ListItem> bằng props ‘stt’.(vậy là từng NoteItem đều nhận được id riêng của mình như -LtspbTZixU3AjHuNkPi; -Ltspba\_tdC5JvJLuhsT).
      * **Tại <NoteItem>, kết nối với Store bằng các bước**  đã học:
        + ****
      * **Tại** Store.js: Khai báo case “DELETE\_A\_DATA” và nhận vào 1 action.item (id của note đó)
      * Dùng hàm child(id).remove() để xóa Note cần xóa tại CSDL Firebase.
      * ****
    - **XÓA TẤT CẢ NOTE:**
      * ****
      * **Tại** <NoteForm>, kết nối component này với store như trên.
      * Khai báo hàm removeAllDataStore() để xóa tất cả dữ liệu
      * Tại <button> ‘xóa hết’ , onClick để xử lý sự kiện xóa hết dữ liệu:
        + ****
      * ****
      * **Tại** store.js, viết case “delete\_all\_data” để xóa hết tất cả dữ liệu trên csdl FireBase.
  + Chức năng **SỬA ghi chú**:
    - Gồm 6 bước logic cần thực hiện:
      * 1. Lấy dữ liệu từ phần tử cần sửa và hiện Form sửa khi Click vào nút “Sửa”
      * 2. Đẩy dữ liệu vừa nhận được lên store, , lưu tạm dữ liệu vừa nhận vào state ‘editObject’.
      * 3. Truyền state ‘editObject’ từ store về <NoteForm>
      * 4. <NoteForm> nhận dữ liệu note, điền các thông tin dữ liệu cần sửa vào form, vào các ô trống như tiêu đề, nội dung.
      * 5. Click nút lưu thì lấy được thông tin đã sửa và đẩy về store.
        + Xác định được là Click vào là edit hay là thêm mới.
        + Nếu mà không sửa gì thì cũng phải lấy được dữ liệu cần sửa.
        + Cần phải giải quyết lỗi: nếu không Sửa dữ liệu gì, click vào nút Lưu thì vẫn lưu các dữ liệu cũ, chứ không phải là xóa hết.
      * 6. Tại store, cập nhật vào cơ sở dữ liệu.
    - Các bước Kỹ Thuật:
      * Bước 1, Bước 2: Lấy dữ liệu từ phần tử cần sửa và hiện Form sửa khi Click vào nút “Sửa”,(cùng chung bước 2)
      * 2. Đẩy dữ liệu vừa nhận được lên store, , lưu tạm dữ liệu vừa nhận vào state ‘editObject’:
        + Tại <NoteItem>:

****

****

****

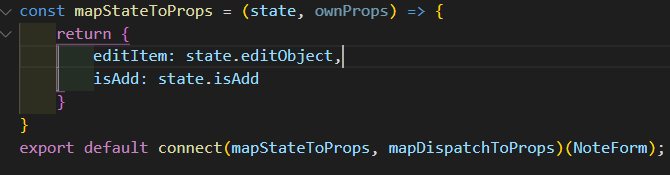
Gọi hàm “changeEditStatus” từ Store để hiển thị Form Sửa.

**‘This.props.note’** chính là thông tin của NoteItem đó, được truyền từ <NoteList>, có cả Id,Title, Content.

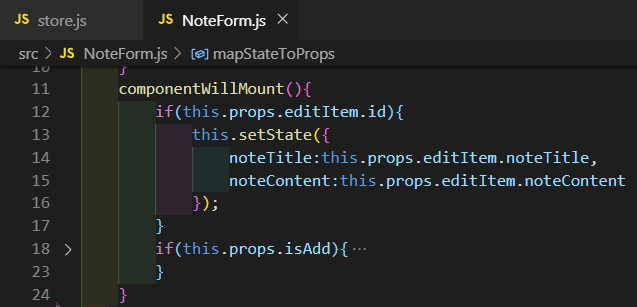
* + - * + Tại Store:

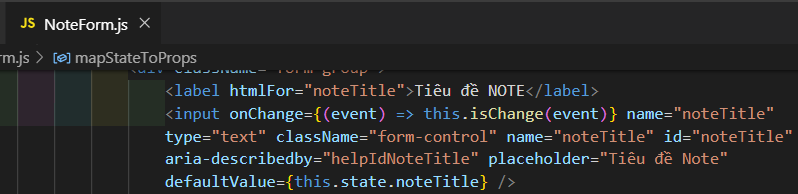
Vậy là Store nhận được đối tượng, dữ liệu của <NoteItem> cần sửa và lưu vào State ‘editObject’.

* + - * Bước 3, bước 4: Truyền state ‘editObject’ từ store về <NoteForm>.<NoteForm> nhận dữ liệu note, điền các thông tin dữ liệu cần sửa vào form, vào các ô trống như tiêu đề, nội dung:
        + Tại <NoteForm>:

****

Nhận được dữ liệu Note cần sửa (editItem).

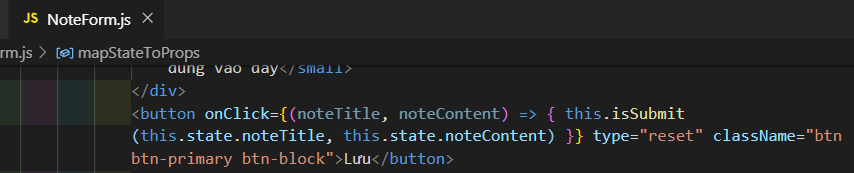
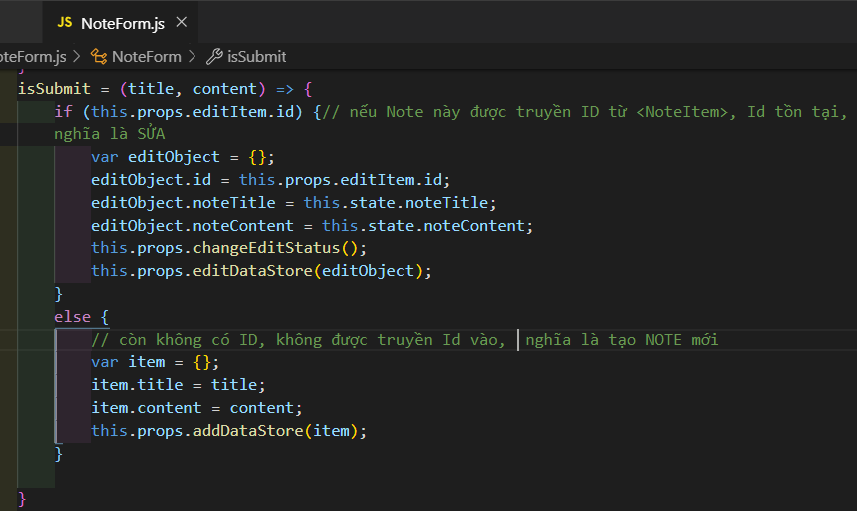
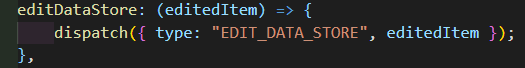


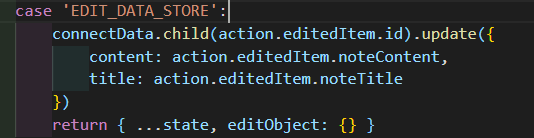


Kiểm tra xem coi có đối tượng nhận vào có Id không? Có nghĩa là đây là Sửa (edit).

Là sửa thì cập nhật dữ liệu ấy vào state để hiển thị dữ liệu ấy ra form thông qua attribute “defaultValue”.

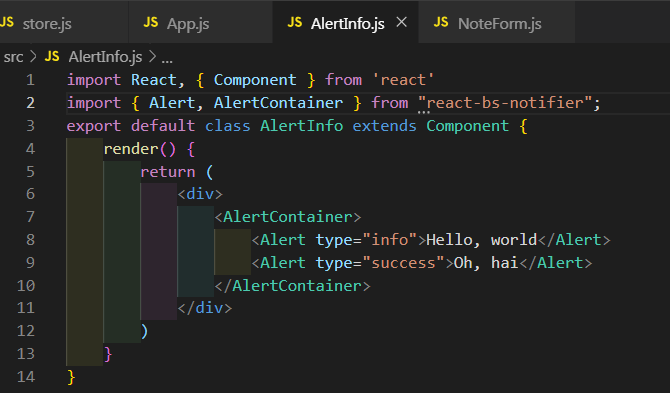
Nếu ko setState thì state.title = ‘ ’ => Form trống .

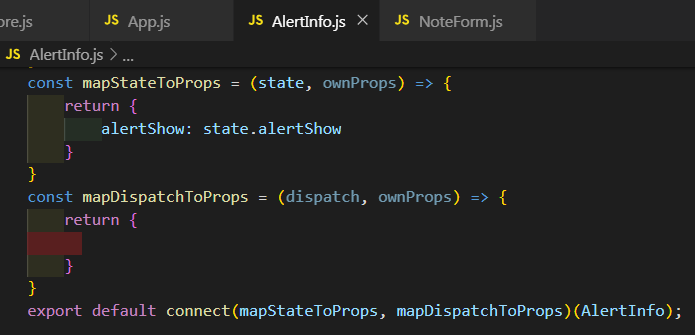
* + - * Bước 5, Bước 6: Click nút lưu thì lấy được thông tin đã sửa và đẩy về store:
        + ****
        + ****
        + ****
        + **Tại Store:**

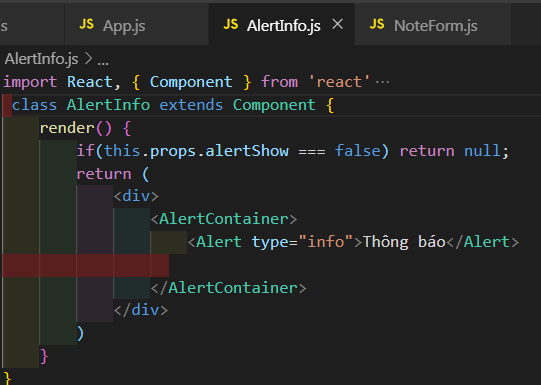
****

**‘editObject’ phải trả** về {} vì sau khi SỬA dữ liệu THÀNH CÔNG rồi thì không lưu thông tin Note ấy nữa, phải xóa thông tin Note ấy.

**Dùng** hàm update() để cập nhật thông tin sửa lên store.

* + - Còn các chức năng phụ thì tự xem project có làm hết.
  + Chức năng **HIỂN THỊ THÔNG BÁO** sau các **THAO TÁC** bằng plugin của ReactJs:plugin ‘react-bs-notifier’:
    - <https://github.com/chadly/react-bs-notifier>
    - <https://chadly.github.io/react-bs-notifier/>
    - **1. Cài đặt: npm install react-bs-notifier –save**
    - **2. Các bước** làm:
      * Bước 1: Định nghĩa Component đặt trong App.js <App>:
        + ****
      * Bước 2: Khi mà thao tác chức năng (thêm, sửa, xóa)=> Truyền thông tin, tiêu đề… vào thông báo để hiển thị ra.
        + 2.1: Định nghĩa biến alertShow để qui định lúc nào thì hiển thị, lúc nào không hiển thị.

****

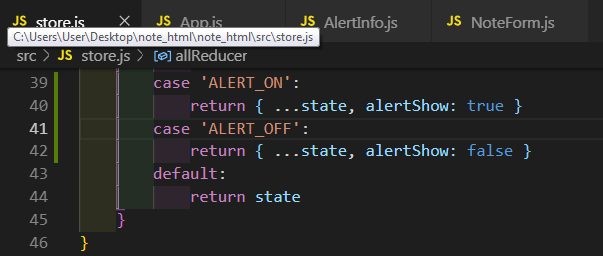
****

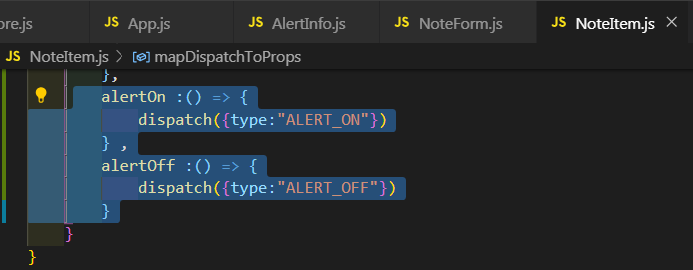
Định nghĩa 1 biến (alertShow) trong state để theo dõi tình trạng hiển thị của <AlertInfo> này(true= = hiển thị /false = biến mất ).

Kết nối <AlertInfo> với Store.<AlertInfo> nhận được state ‘alertShow’ từ store.

* + - * + 2.2: chức năng: Sửa xong thì thông báo, xóa xong thì thông báo

How:

****

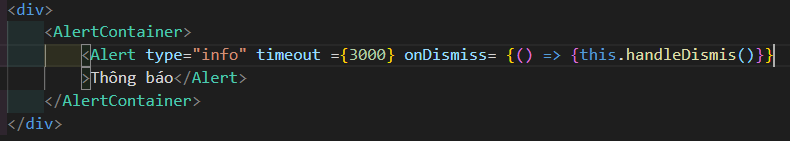
****

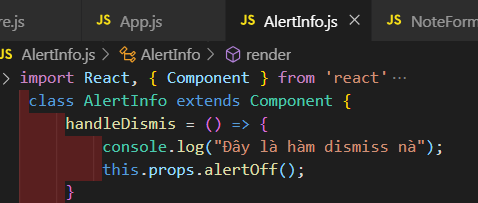
****

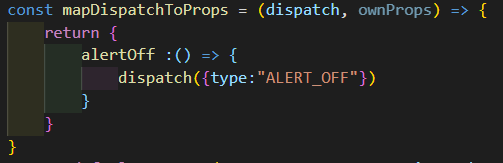
Thay đổi state ‘alertShow’ trong store => thông báo sẽ hiển thị hoặc không hiển thị ra.Phải có action tương ứng để thay đổi state.

Định nghĩa 2 action “ALERT\_ON” và “ALERT\_OFF”

**Chức năng:** Bảng Alert thông báo tự động tắt hoặc ấn nút X để tắt:

****

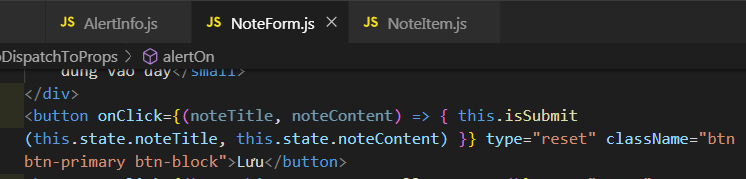
****

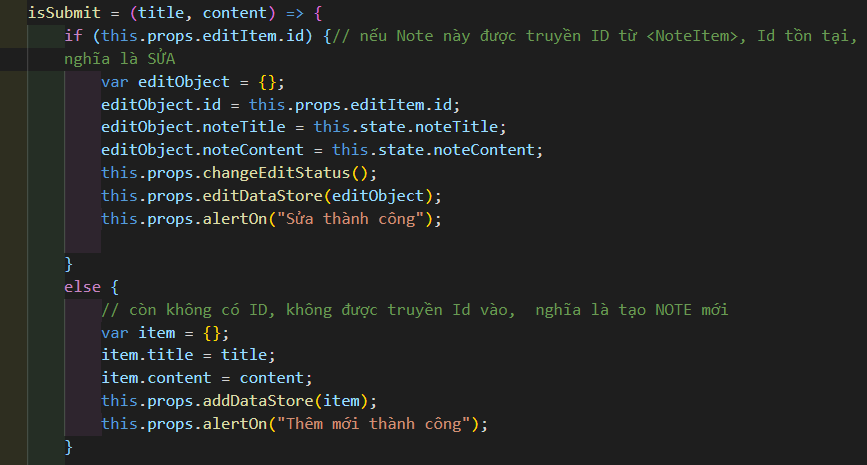
****

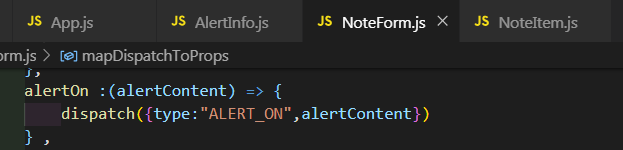
**Sau** 3s ( của hàm timeout()) thì sẽ chạy hàm nằm trong sử kiện onDismiss

Nghĩa là sau 3s sẽ chạy hàm handleDismis(). Mình có thể tự định nghĩa hàm handleDismis() tùy ứng để chỉnh hiệu ứng tắt thông báo.

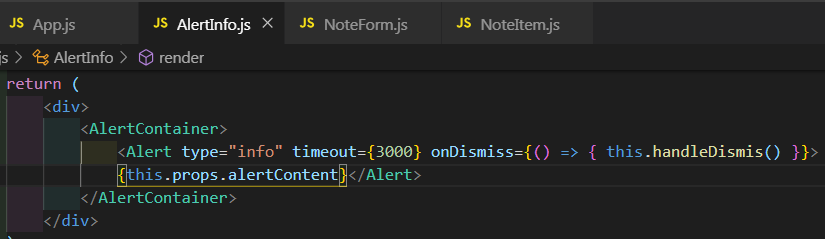
Chức năng: truyền tham số, nội dung vào Alert thông qua State:

****

****

****

****

****

Định nghĩa state ‘alertContent’ ở store để lưu tạm thời nội dung thông báo.

Khi gọi hàm AlertOn() thì truyền vào tham số alertContent để đưa tham số về store, rồi lưu vào state.

Sau có truyền state “alertContent” ấy từ store cho <AlertInfo>.

* + CÁC CHỨC NĂNG PHỤ KHÁC TỰ COI TRONG PROJECT.
  + HẾT.