**Họ tên: Phan Văn Thọ**

**Lớp: LA-31-AM**

TÌM HIỂU VÀ VIẾT ỨNG DỤNG REACTJS QUẢN LÝ TIN TỨC ỨNG DỤNG KIẾN TRÚC REDUX VÀ CÁC HOOK VÀO ỨNG DỤNG

MỤC LỤC

[Phần 1. Tìm hiểu và mô tả các sử dụng Redux và các Hook trong reactjs: 2](#_Toc143356465)

[I. Tìm hiểu redux 2](#_Toc143356466)

[1. Redux là gì? 2](#_Toc143356467)

[2. Tại sao cần redux để quản lý trạng thái? 2](#_Toc143356468)

[II. Mô tả các sử dụng redux trong ứng dụng reactjs: 2](#_Toc143356469)

[III. Tìm hiểu các hook trong reactjs 4](#_Toc143356470)

[1. Hook là gì? 4](#_Toc143356471)

[2. Các hook trong reactjs: 4](#_Toc143356472)

[Phần 2. Xây dựng ứng dụng crud tin tức ứng dụng kiến trúc redux: 5](#_Toc143356473)

[I. Cài đặt reactjs và import 1 số thư viện cần thiết: 5](#_Toc143356474)

[II. Thực hiện chương trình: 7](#_Toc143356475)

[Phần 3. Kết quả: 12](#_Toc143356476)

[Hiển thị danh sách bài viết: 12](#_Toc143356477)

[Xóa bài viết 13](#_Toc143356478)

[Thêm bài viết 13](#_Toc143356479)

[Sửa bài viết 14](#_Toc143356480)

# Phần 1. Tìm hiểu và mô tả các sử dụng Redux và các Hook trong reactjs:

## Tìm hiểu redux

### Redux là gì?

* Redux là một thư viện quản lý trạng thái cho ứng dụng web được xây dựng bằng JavaScript. Nó thường được sử dụng trong các ứng dụng React để quản lý trạng thái của các thành phần và tạo ra một luồng dữ liệu đơn nhất và dễ dàng kiểm soát.

### Tại sao cần redux để quản lý trạng thái?

Redux cung cấp một cách mạnh mẽ để quản lý trạng thái trong ứng dụng, đặc biệt là trong các ứng dụng React hoặc các ứng dụng có giao diện người dùng phức tạp. Dưới đây là một số lý do tại sao Redux được sử dụng để quản lý trạng thái:

* + Redux giúp duy trì một nguồn dữ liệu duy nhất để lưu trữ trạng thái của toàn bộ ứng dụng. Điều này làm cho việc theo dõi và kiểm soát trạng thái trở nên dễ dàng và tránh việc trạng thái bị lẫn lộn hoặc không đồng nhất.
  + Trạng thái trong Redux không thể thay đổi trực tiếp. Thay vào đó, các thay đổi trạng thái được thực hiện thông qua các "actions" và "reducers". Điều này tạo ra một mô hình dự đoán và theo trình tự, giúp bạn dễ dàng theo dõi và kiểm tra những thay đổi nào ảnh hưởng đến trạng thái.
  + Redux cung cấp công cụ mạnh để gỡ lỗi và theo dõi lịch sử thay đổi trạng thái. Bạn có thể xem lại tất cả các hành động đã xảy ra và xem cách trạng thái thay đổi qua thời gian, điều này rất hữu ích khi phát triển và gỡ lỗi.
  + Redux tách biệt trạng thái khỏi các thành phần giao diện. Điều này giúp tạo ra một cấu trúc ứng dụng dễ quản lý hơn, giúp bạn tập trung vào việc xây dựng các thành phần mà không cần lo lắng về việc quản lý trạng thái.
  + Redux cho phép chia sẻ trạng thái dễ dàng giữa các thành phần khác nhau mà không cần truyền dữ liệu qua các lớp con hoặc thông qua các thành phần cha.
  + Redux cho phép sử dụng middleware để thực hiện các tác vụ như ghi nhật ký, xử lý bất đồng bộ, hay thay đổi dữ liệu trước khi nó đến reducers.

## Mô tả các sử dụng redux trong ứng dụng reactjs:

1. Cài đặt Redux:

Đầu tiên, cần cài đặt Redux và React-Redux (thư viện kết nối Redux với React) vào dự án của mình bằng cách sử dụng npm hoặc yarn.

1. State (Trạng thái) và Store (Kho dữ liệu):

Redux quản lý trạng thái của ứng dụng bằng cách lưu trữ nó trong một cấu trúc gọi là "store".

Trạng thái của ứng dụng được lưu trữ trong store và không thể được thay đổi trực tiếp.

1. Actions (Hành động):

Các "actions" là các sự kiện hoặc hành động muốn thực hiện để thay đổi trạng thái.

Mỗi action cần phải có một loại (type) để xác định loại hành động đang thực hiện.

1. Reducers (Bộ xử lý):

Các "reducers" là các hàm xác định cách trạng thái thay đổi dựa trên các action tương ứng.

Mỗi reducer nhận vào trạng thái hiện tại và action, sau đó trả về một trạng thái mới.

1. Store (Kho dữ liệu) và Kết nối với React:

Khi khởi tạo store, cần chuyển vào tất cả các reducers.

Sử dụng thư viện React-Redux, có thể "kết nối" các thành phần React với store để truy cập và cập nhật trạng thái.

1. Actions Creators (Trình tạo hành động):

Actions creators là các hàm được sử dụng để tạo ra các actions. Điều này giúp tránh việc phải viết lại cùng mã lệnh cho các actions tương tự.

1. Dispatch (Phát ra) Actions:

Khi muốn thay đổi trạng thái, sẽ "dispatch" một action đến store bằng cách gọi hàm dispatch(action).

1. Selectors (Bộ chọn):

Selectors là các hàm giúp truy cập trạng thái từ store và chọn dữ liệu cụ thể cần từ trạng thái đó.

1. Middleware (Phần mềm trung gian):

Middleware là một cơ chế có thể sử dụng để thực hiện các tác vụ trước khi action đến reducers, chẳng hạn như xử lý bất đồng bộ.

1. Thực hành và Điều hướng dự án:

Xác định cách trạng thái sẽ được tổ chức và xử lý bằng Redux.

Xác định các actions và reducers tương ứng để thay đổi trạng thái.

Kết nối các thành phần React với store bằng React-Redux.

Sử dụng selectors để truy cập dữ liệu từ store.

## Tìm hiểu các hook trong reactjs

### Hook là gì?

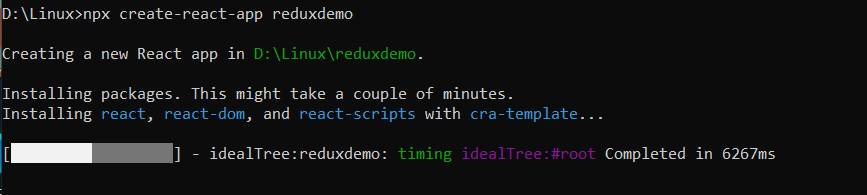
* Trong ReactJS, hook là một tính năng cho phép sử dụng trạng thái và các tính năng của React trước đây trong các thành phần hàm chức năng (functional components). Các hook giúp chương trình có thể sử dụng các khả năng như trạng thái, hiệu ứng và vòng đời trong các thành phần hàm một cách dễ dàng hơn mà không cần chuyển đổi thành các thành phần lớp.

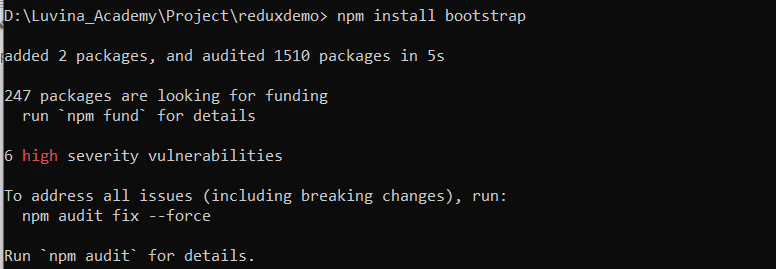
### Các hook trong reactjs:

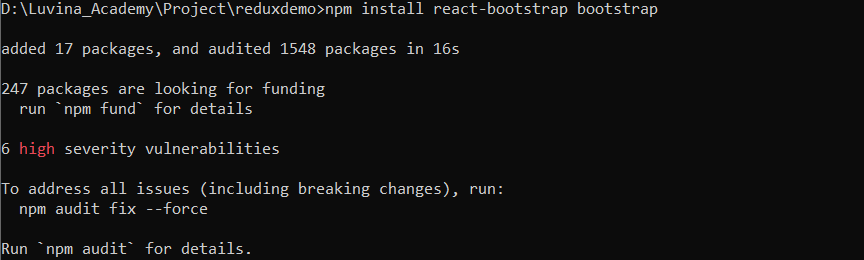
* useState: Cho phép bạn sử dụng và quản lý trạng thái trong một thành phần hàm.
* useEffect: Cho phép bạn thực hiện các hiệu ứng (side effects) sau mỗi lần render, chẳng hạn như gọi API, thao tác với DOM, v.v.
* useContext: Cho phép bạn truy cập vào React Context từ trong một thành phần hàm.
* useReducer: Tương tự như useState, nhưng thay vì quản lý trạng thái dưới dạng giá trị, bạn quản lý trạng thái thông qua reducer, tương tự như Redux.
* useMemo: Cho phép bạn tối ưu hóa tính toán bằng cách lưu giữ giá trị tính toán trước đó giữa các lần render.
* useCallback: Tương tự như useMemo, nhưng được sử dụng để lưu giữ các hàm callback trước đó giữa các lần render.
* useRef: Cho phép bạn tham chiếu đến các phần tử DOM hoặc lưu giữ các giá trị khác mà không gây ra việc render lại khi thay đổi.
* useCustomHook: Bạn có thể tạo ra các hook tùy chỉnh của riêng mình để chia sẻ logic và tính năng giữa các thành phần.

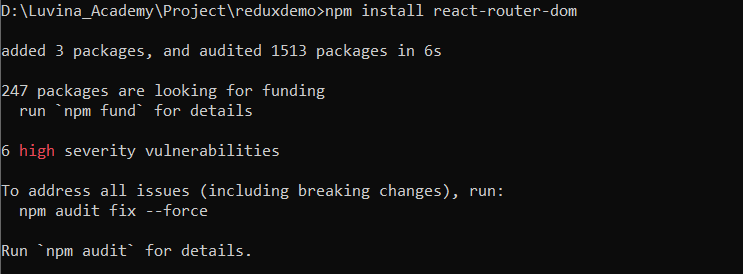
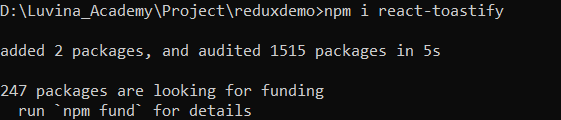
# Phần 2. Xây dựng ứng dụng crud tin tức ứng dụng kiến trúc redux:

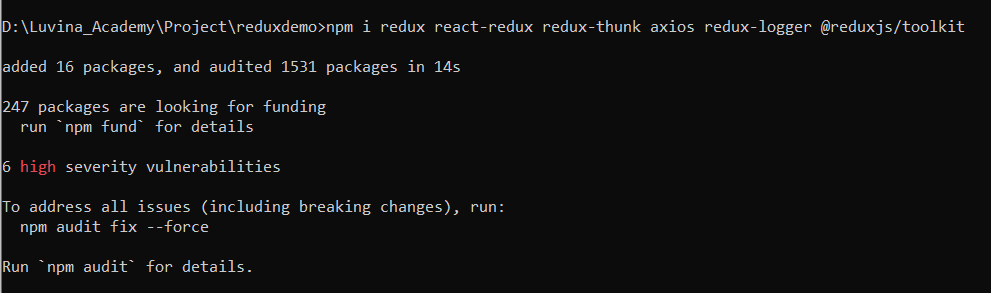
## Cài đặt reactjs và import 1 số thư viện cần thiết:

* Tạo dự án reactjs:
* Cài đặt bootstrap và react-bootstrap:

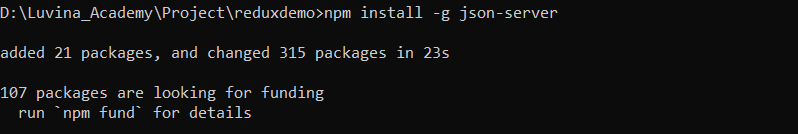




* Cài đặt react-router-dom
* Cài đặt toastify(hiển thị thông báo)
* Cài đặt redux và axios:

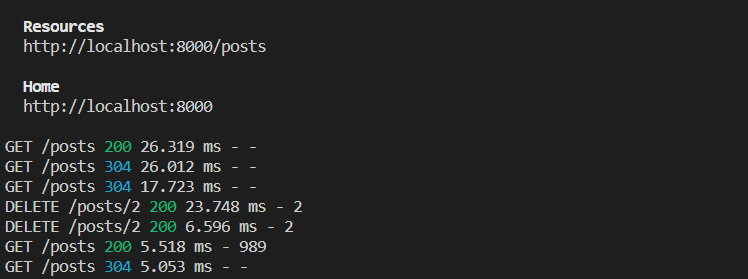


* Cài đặt json-server:

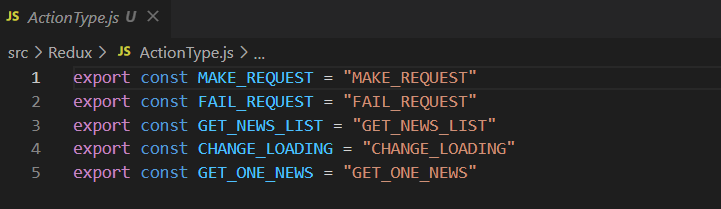


## Thực hiện chương trình:

* Bước 1: Chạy json-server bằng lệnh sau: json-server --watch src/Data/db.json --port 8000



* Bước 2: Tạo 1 folder Redux chứa các file sau để cài đặt kiến trúc redux cho ứng dụng:
  + File ActionType.js: chứa các hằng số đại diện cho các loại hành động (actions) mà ứng dụng có thể thực hiện



MAKE\_REQUEST: Đại diện cho việc thực hiện một yêu cầu hoặc hành động gửi đi (request) trong ứng dụng. Điều này có thể liên quan đến việc gọi API hoặc tương tác bất đồng bộ khác.

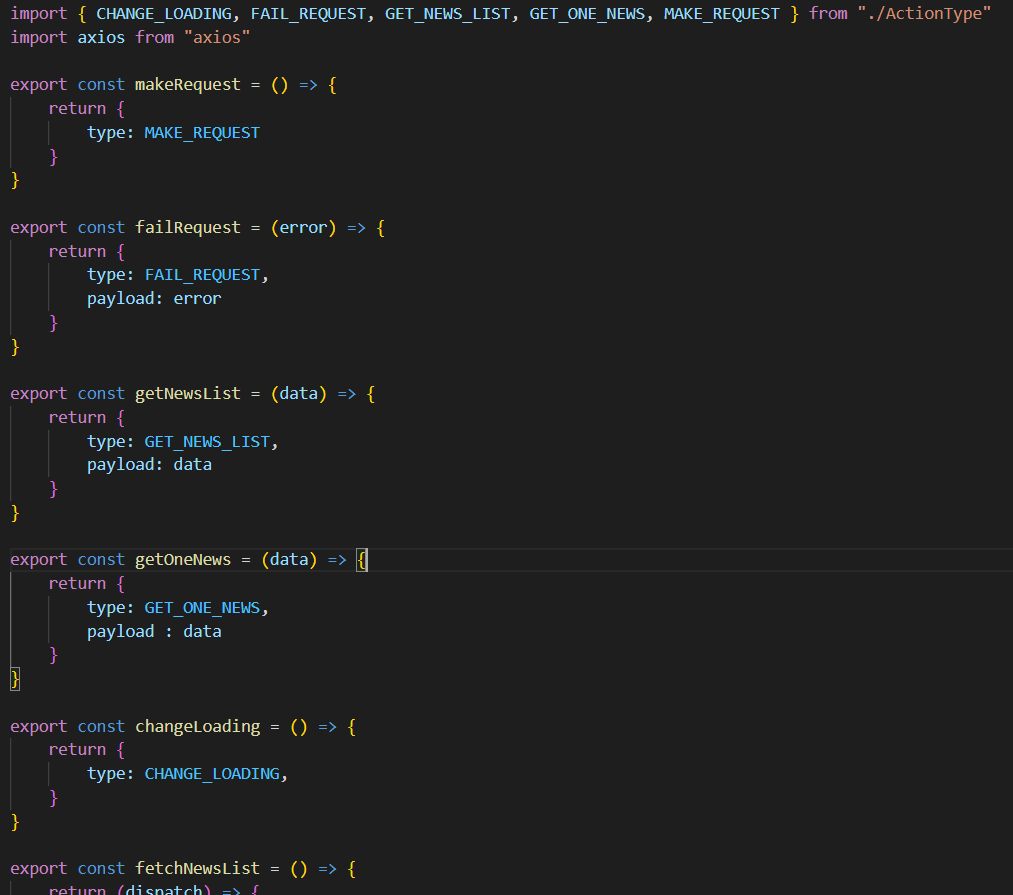
FAIL\_REQUEST: Đại diện cho trường hợp xảy ra lỗi hoặc thất bại trong quá trình thực hiện yêu cầu.

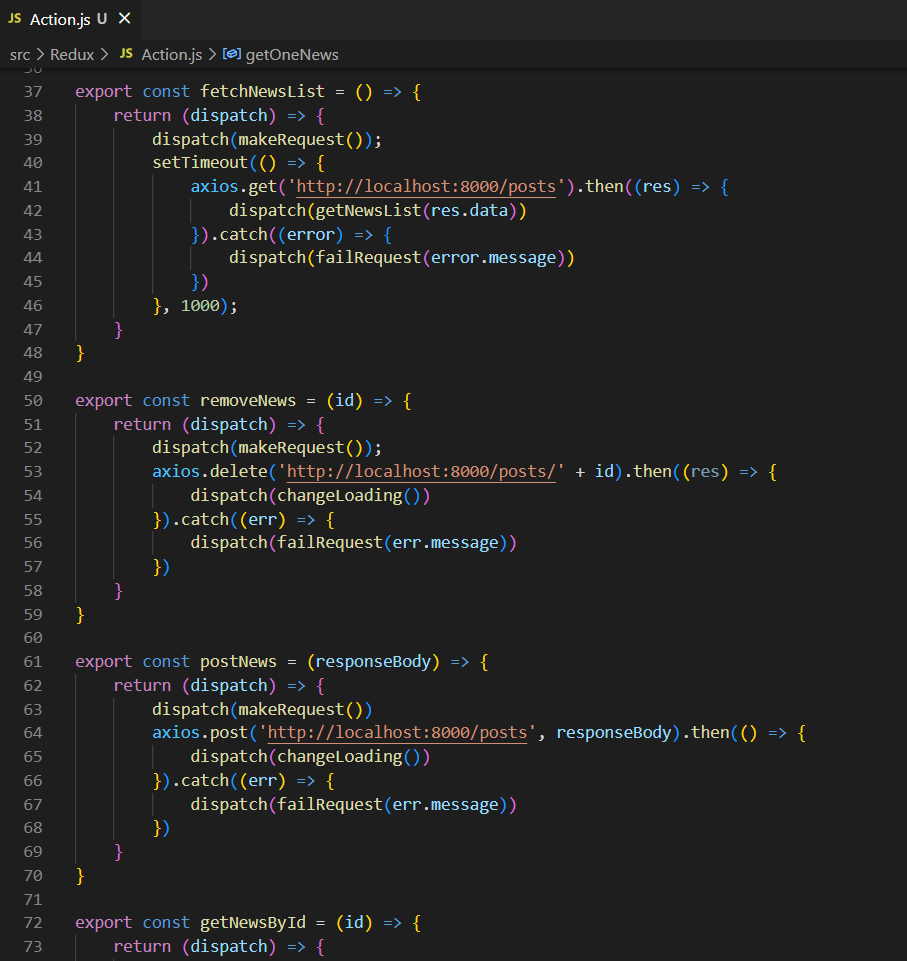
GET\_NEWS\_LIST: Đại diện cho việc lấy danh sách tin tức (news list) từ nguồn dữ liệu nào đó.

CHANGE\_LOADING: Đại diện cho việc thay đổi trạng thái của biểu thị tải dữ liệu, chẳng hạn khi đang đợi dữ liệu được tải.

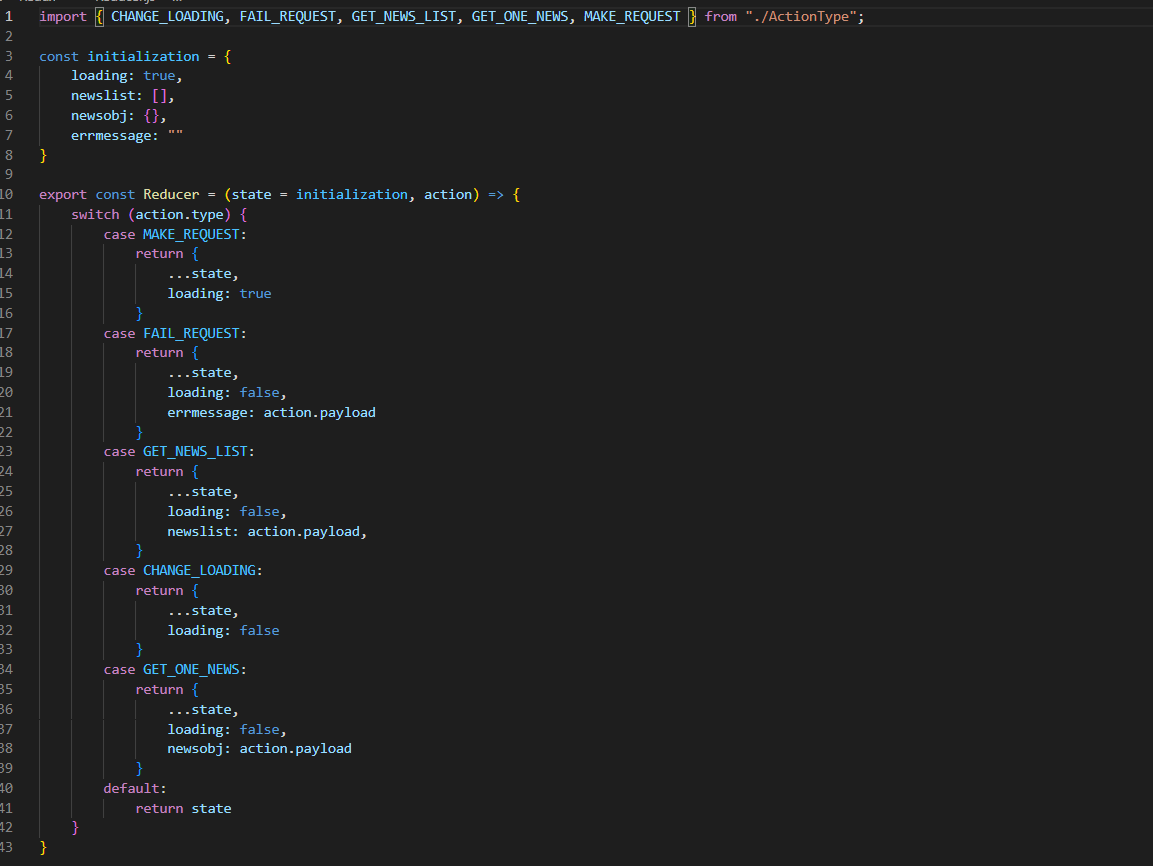
GET\_ONE\_NEWS: Đại diện cho việc lấy dữ liệu của một tin tức cụ thể.

* + File Action.js: định nghĩa một loạt các action creators, mỗi cái tương ứng với một loại hành động trong ứng dụng. Mỗi action creator trả về một action có cấu trúc gồm type (kiểu hành động) và payload (dữ liệu đi kèm).

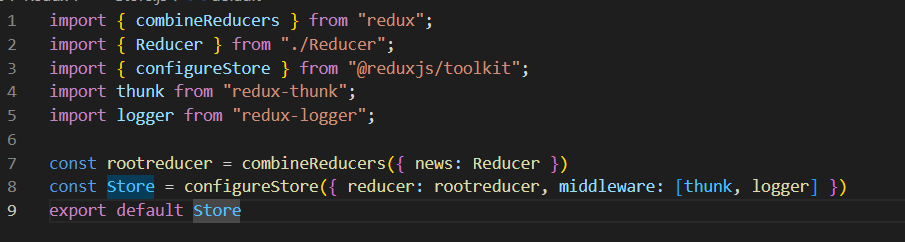




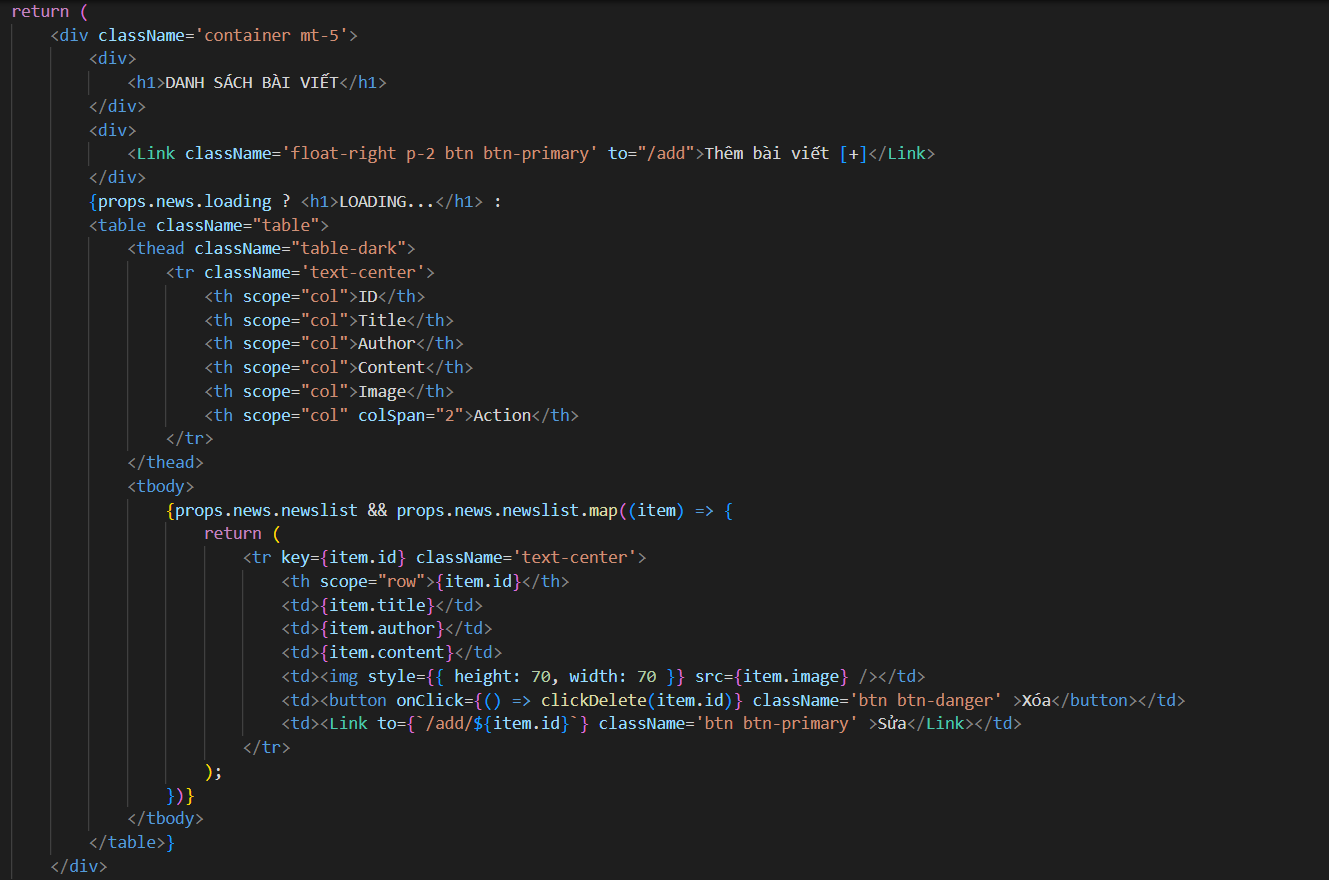
* + File Reducer.js: có nhiệm vụ xử lý các hành động (actions) và cập nhật trạng thái của ứng dụng

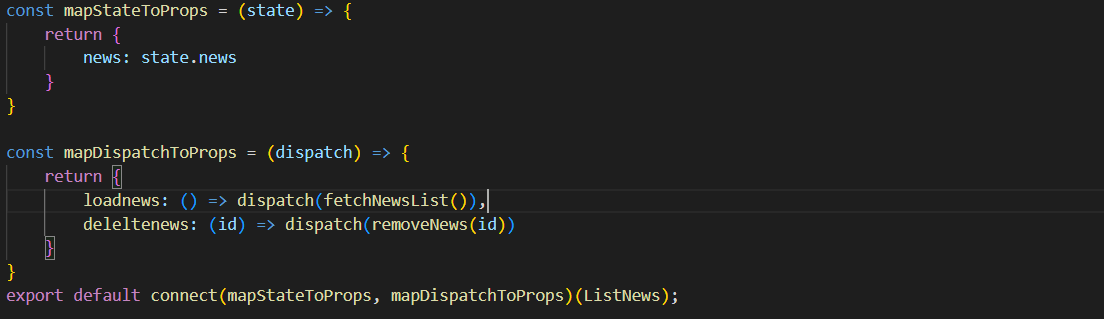


* + File Store.js: chứa cấu hình Redux store sử dụng Redux Toolkit và các middleware như Redux Thunk và Redux Logger, được import và sử dụng trong ứng dụng của để tạo Redux store và quản lý trạng thái của ứng dụng.



* Bước 3: Tạo component ListNews để hiển thị giao diện và sử dụng kiến trúc Redux:



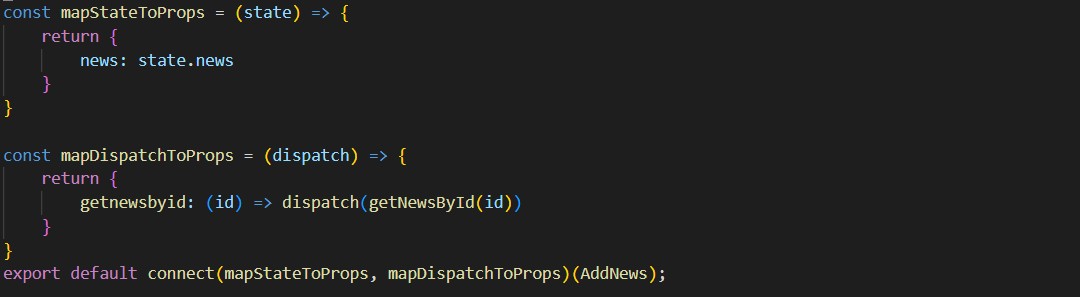


Đoạn mã trên thực hiện việc kết nối thành phần ListNews với Redux store bằng cách sử dụng connect từ thư viện react-redux

Phương thức mapStateToProps truyền dữ liệu từ trạng thái Redux (phần news) vào props của ListNews

Phương thức mapDispathToProps truyền các action creators vào props để gửi các hành động

* Bước 4: Tạo component AddNews để xử lý trường hợp thêm sửa bài viết sử dụng kiến trúc Redux:





Phương thức submitNews thực hiện các hành động liên quan đến việc thêm dữ liệu của tin tức lên server hoặc chỉnh sửa dữ liệu của một tin tức cụ thể thông qua Redux actions

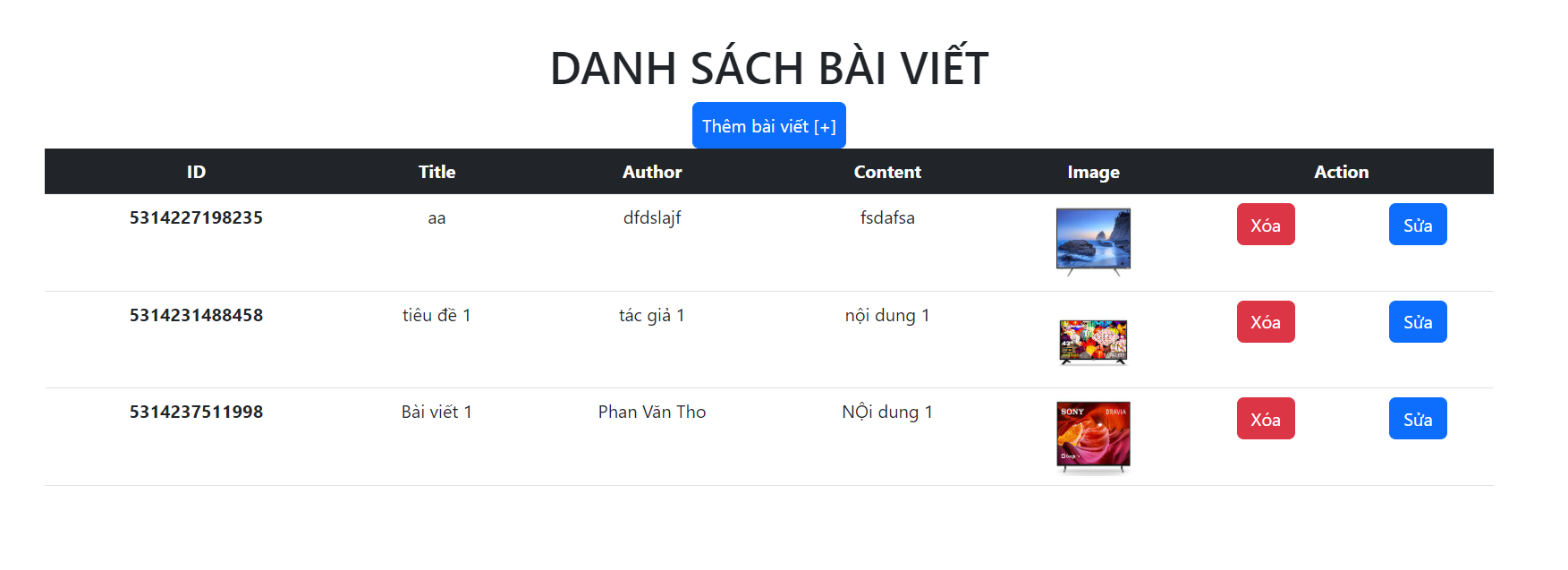
Phương thức dispathch() sẽ gửi các action cụ thể đến redux để xử lý



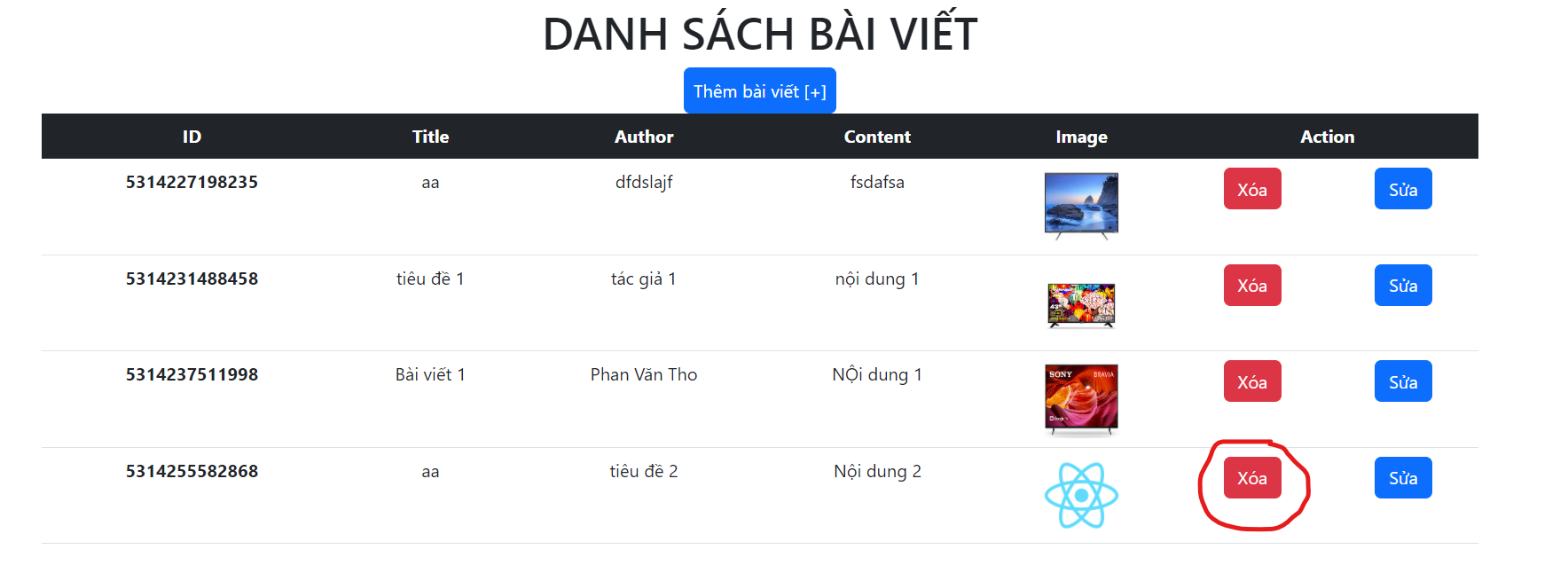
Đối với mode edit sẽ gọi phương thức getnewbyid(id) đến redux để hiển thị thông tin bài viết cần chỉ sửa lên form

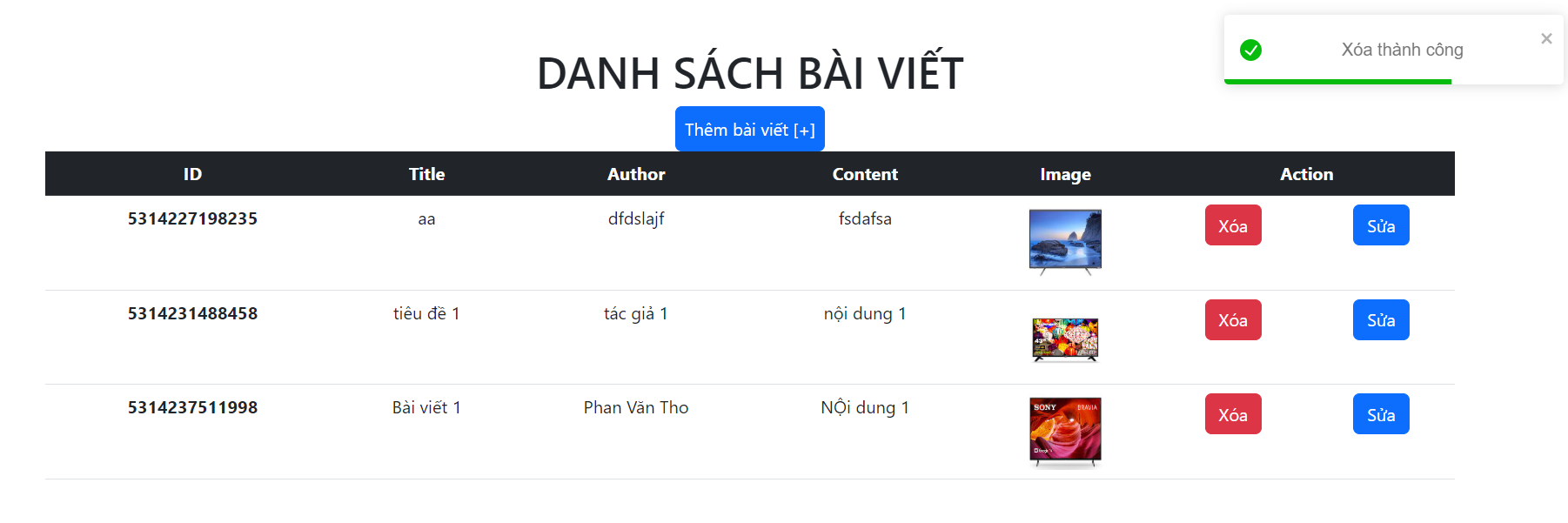
# Phần 3. Kết quả:

## Hiển thị danh sách bài viết:

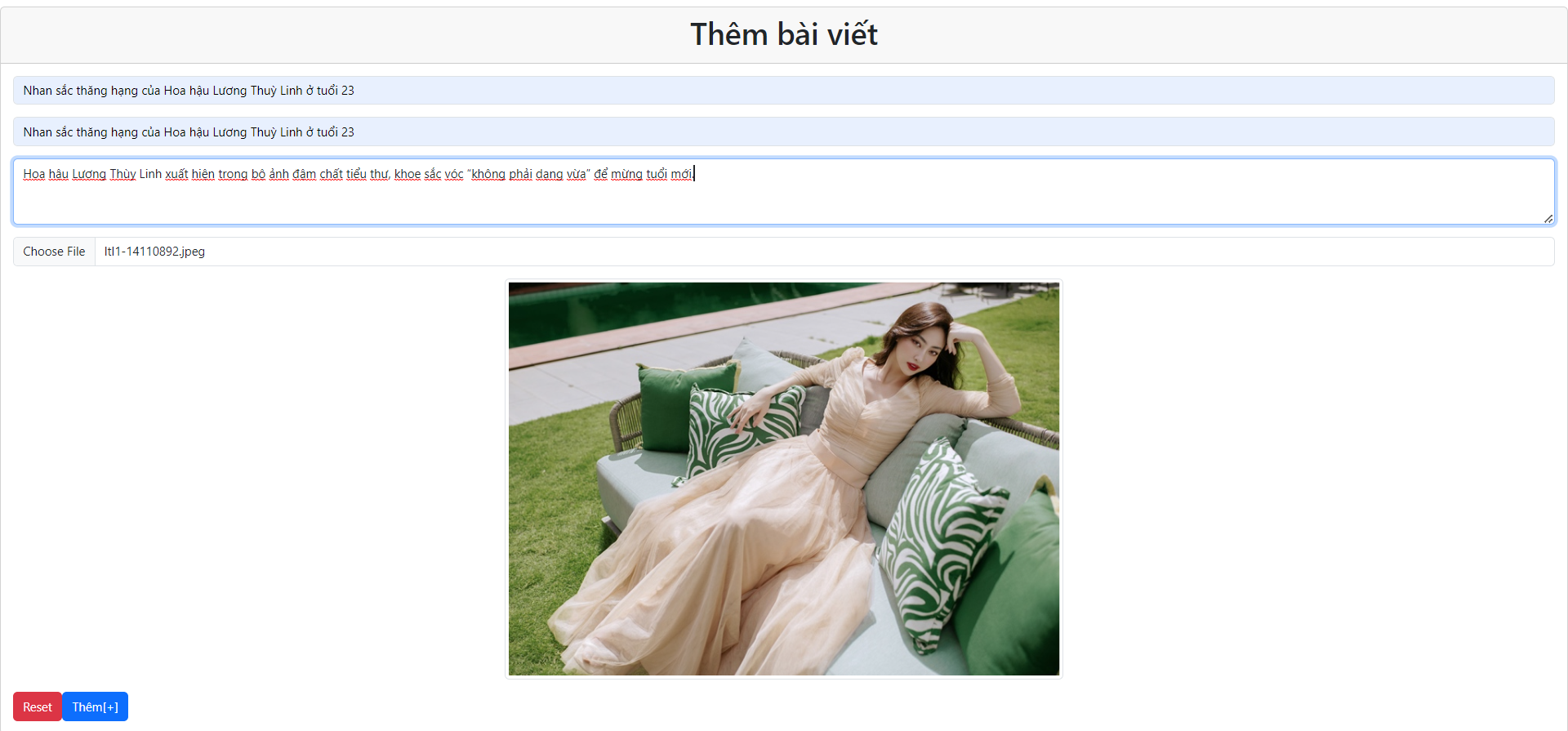


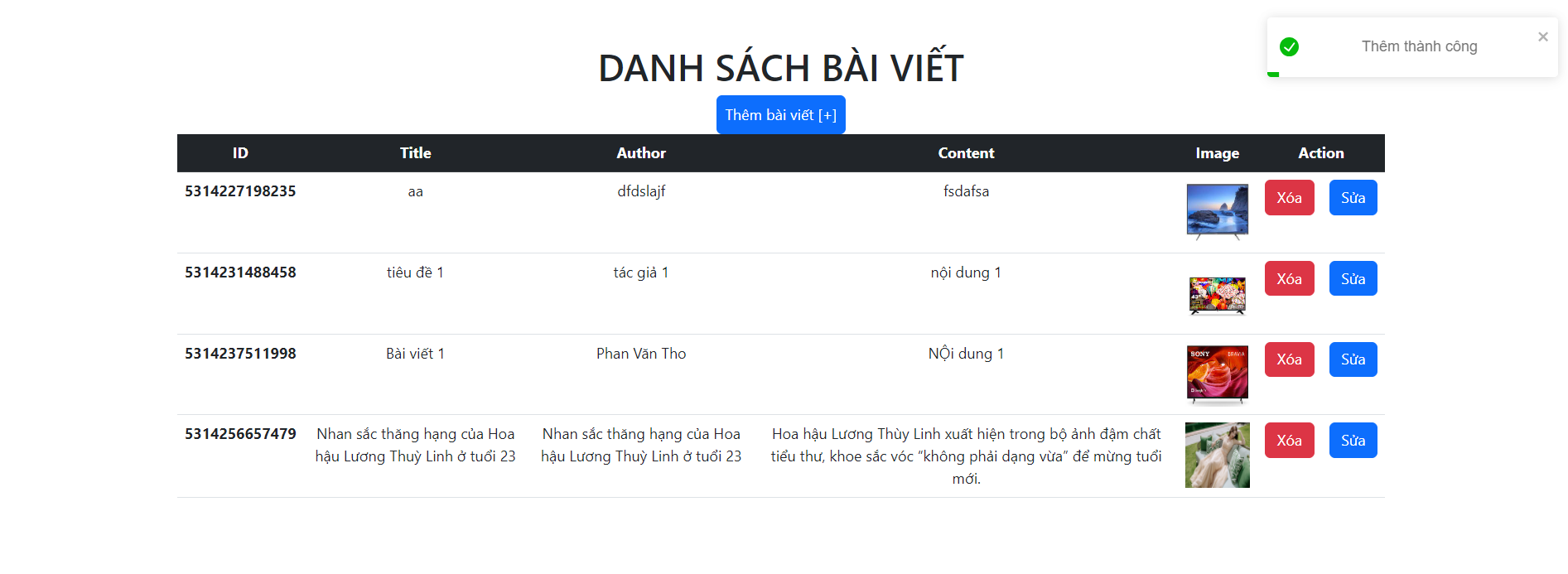
## Xóa bài viết





## Thêm bài viết





## Sửa bài viết

