# Luồng chạy dự án react-crud-jsonserver

## Luồng chạy (flow) App.jsx

## 🔁 1. Khi ứng dụng được khởi chạy (mount)

```
js
Sao chépChỉnh sửa
useEffect(() => {
  loadUsers();
}, []);
```

- `useEffect` với dependency rỗng (`[]`) chỉ chạy **một lần sau khi component `App` được mount**.
- Gọi hàm `loadUsers`, hàm này:

  ```
  js
  Sao chépChỉnh sửa
  const loadUsers = () => {
    getUsers().then((res) => setUsers(res.data));
  };
  ```

  → Gọi API `getUsers()`
  → Khi có dữ liệu (response), gọi `setUsers(res.data)` để cập nhật danh sách người dùng (`users`) trong state.

---

## ✍️ 2. Khi người dùng thêm hoặc sửa (submit form)

```
js
Sao chépChỉnh sửa
const handleSubmit = (data) => {
  if (data.id) {
    updateUser(data.id, data).then(loadUsers);
    setSelectedUser(null);
  } else {
    addUser(data).then(loadUsers);
  }
};
```

### Trường hợp 1: Sửa (edit)

- Nếu `data.id` tồn tại → Gọi `updateUser(id, data)` để cập nhật.
- Sau khi cập nhật xong:
  - Gọi lại `loadUsers()` để refresh danh sách.

- Reset `selectedUser` về `null` để form quay về trạng thái tạo mới.

**Trường hợp 2: Thêm mới**

- Nếu không có `data.id` → Gọi `addUser(data)`.
- Sau khi thêm xong → `loadUsers()` để cập nhật lại danh sách.

---

# 🗑️ 3. Khi người dùng xóa (delete)

```
js
Sao chépChỉnh sửa
const handleDelete = (id) => {
  deleteUser(id).then(loadUsers);
};
```

- Gọi `deleteUser(id)` để xóa user theo `id`.
- Sau khi xóa xong → Gọi `loadUsers()` để cập nhật danh sách.

---

# ✏️ 4. Khi người dùng chọn chỉnh sửa một user

```
js
Sao chépChỉnh sửa
const handleEdit = (user) => setSelectedUser(user);
```

- Khi click nút "Edit" trong danh sách → `setSelectedUser(user)`
- Dữ liệu `selectedUser` này sẽ được truyền xuống component `UserForm`.

---

# 🧩 5. Hiển thị giao diện

```
js
Sao chépChỉnh sửa
<UserForm
  onSubmit={handleSubmit}
  selectedUser={selectedUser}
  clearSelection={() => setSelectedUser(null)}
/>

<UserList
  users={users}
  onEdit={handleEdit}
  onDelete={handleDelete}
```

```
/>
```

- `UserForm`: hiển thị form nhập liệu (thêm hoặc sửa), dựa vào `selectedUser`.
- `UserList`: danh sách user, có nút "Edit" và "Delete".

---

# ✅ Tổng kết dòng chảy chính:

```
plaintext
Sao chépChỉnh sửa
1. App mount → useEffect chạy → loadUsers → gọi API getUsers → setUsers

2. User click Edit → setSelectedUser → UserForm hiển thị dữ liệu → Submit:
    - Nếu có ID → gọi updateUser → loadUsers → clear selectedUser
    - Nếu không có ID → gọi addUser → loadUsers

3. User click Delete → gọi deleteUser → loadUsers
```

# Luồng chạy (flow) UserForm.jsx

## 📦 1. Khởi tạo component

```
js
Sao chépChỉnh sửa
const [formData, setFormData] = useState({ name: "", email: "" });
```

- Tạo state `formData` để lưu dữ liệu form (gồm `name` và `email`).
- Ban đầu là chuỗi rỗng: `{ name: "", email: "" }`.

---

## 🔁 2. Khi `selectedUser` thay đổi

```
js
Sao chépChỉnh sửa
useEffect(() => {
  if (selectedUser) setFormData(selectedUser);
}, [selectedUser]);
```

- Khi prop `selectedUser` thay đổi (ví dụ: user click nút Edit ở `UserList`) → `useEffect` chạy.
- Nếu `selectedUser` tồn tại, gán dữ liệu của user đó vào form để hiển thị cho người dùng sửa.

### ✅ Ví dụ:

Nếu `selectedUser = { name: "An", email: "an@example.com" }`
→ Form sẽ hiển thị sẵn dữ liệu của user "An".

---

## ⌨️ 3. Khi người dùng gõ trong form

```
js
Sao chépChỉnh sửa
const handleChange = (e) => {
  const { name, value } = e.target;
  setFormData((prev) => ({ ...prev, [name]: value }));
};
```

- Mỗi lần gõ vào `input`, hàm này sẽ:
  - Lấy `name` của input (có thể là `"name"` hoặc `"email"`).
  - Cập nhật giá trị tương ứng trong `formData`.

### Ví dụ:

- Gõ "Hà" vào ô tên → `formData` sẽ thành `{ name: "Hà", email: "" }`.

---

## 📩 4. Khi người dùng bấm nút Submit

```
js
Sao chépChỉnh sửa
const handleSubmit = (e) => {
  e.preventDefault();
  onSubmit(formData);
  setFormData({ name: "", email: "" });
};
```

- Ngăn reload trang (`e.preventDefault()`).
- Gọi hàm `onSubmit(formData)` từ prop — chính là hàm `handleSubmit` bên trong `App.js`.
  → Gửi dữ liệu form đi (có thể là thêm mới hoặc cập nhật).
- Reset form về rỗng sau khi gửi.

---

## ❌ 5. Khi người dùng bấm nút "Huỷ" (nếu đang sửa)

```
js
Sao chépChỉnh sửa
{selectedUser && <button onClick={clearSelection}>Huỷ</button>}
```

- Chỉ hiện nút "Huỷ" nếu đang trong trạng thái sửa (`selectedUser` có giá trị).
- Bấm nút → gọi `clearSelection()` (được truyền từ `App.js`)
  → Thường là `setSelectedUser(null)`, giúp form trở về trạng thái **thêm mới**.

---

## 🧩 6. Giao diện được hiển thị như sau:

```
jsx
Sao chépChỉnh sửa
<form onSubmit={handleSubmit}>
  <input ... />  // Nhập tên
  <input ... />  // Nhập email
  <button type="submit">
    {selectedUser ? "Cập nhật" : "Thêm mới"}
  </button>
  {selectedUser && <button onClick={clearSelection}>Huỷ</button>}
</form>
```

---

## ✅ Tổng kết luồng hoạt động:

```
plaintext
Sao chépChỉnh sửa
1. Form khởi tạo với state trống { name: "", email: "" }

2. Nếu có selectedUser → form tự động điền dữ liệu vào

3. Người dùng nhập dữ liệu → formData được cập nhật liên tục

4. Submit:
   - Gọi onSubmit(formData)
   - Reset form

5. Nếu đang sửa → hiển thị nút "Huỷ"
   - Bấm "Huỷ" → reset selectedUser → form quay về trạng thái thêm mới
```

# Luồng chạy (flow) UserList.jsx

## 🧩 Tổng quan component `UserList`

```
js
Sao chépChỉnh sửa
```

```
const UserList = ({ users, onEdit, onDelete }) => (
  <ul>
    {users.map((user) => (
      <li key={user.id}>
        {user.name} - {user.email}
        <button onClick={() => onEdit(user)}>Sửa</button>
        <button onClick={() => onDelete(user.id)}>Xoá</button>
      </li>
    ))}
  </ul>
);
```

- **`users`**: Mảng danh sách người dùng (được truyền từ `App.js`).
- **`onEdit`**: Hàm xử lý khi người dùng click "Sửa".
- **`onDelete`**: Hàm xử lý khi người dùng click "Xoá".

---

# 🔁 Luồng chạy chi tiết

## 1. Render danh sách người dùng

```
js
Sao chépChỉnh sửa
users.map((user) => (
  <li key={user.id}>...</li>
))
```

- Lặp qua mảng `users`.
- Với mỗi phần tử (`user`), render ra:
  - Tên
  - Email
  - Hai nút: **Sửa** và **Xoá**
- Mỗi phần tử có `key` duy nhất là `user.id` (giúp React render hiệu quả hơn).

---

## 2. Khi người dùng click nút "Sửa"

```
js
Sao chépChỉnh sửa
<button onClick={() => onEdit(user)}>Sửa</button>
```

- Gọi hàm `onEdit` và truyền toàn bộ đối tượng `user`.
- Tác động:
  - Trong `App.js`, `onEdit = handleEdit`, nó sẽ:

    ```
    js
    Sao chépChỉnh sửa
    ```

```
setSelectedUser(user);
```

- Component `UserForm` sẽ nhận được `selectedUser` mới → Form sẽ tự động điền thông tin để sửa.

---

### 3. Khi người dùng click nút "Xoá"

```
js
Sao chépChỉnh sửa
<button onClick={() => onDelete(user.id)}>Xoá</button>
```

- Gọi hàm `onDelete` và truyền `user.id`.
- Tác động:
  - Trong `App.js`, `onDelete = handleDelete`, nó sẽ:

    ```
    js
    Sao chépChỉnh sửa
    deleteUser(id).then(loadUsers);
    ```

  - Gọi API xoá người dùng → Sau khi xoá xong, gọi lại `loadUsers()` để cập nhật danh sách.

---

# ✅ Tóm tắt luồng hoạt động:

```
plaintext
Sao chépChỉnh sửa
1. users được truyền từ App → UserList → hiển thị thành danh sách <li>

2. Bấm nút "Sửa":
   - Gọi onEdit(user)
   - → App setSelectedUser(user)
   - → UserForm nhận selectedUser → Form chuyển sang trạng thái sửa

3. Bấm nút "Xoá":
   - Gọi onDelete(user.id)
   - → App gọi deleteUser API → loadUsers → danh sách được cập nhật
```

# Luồng chạy (flow) api.js

## 📦 1. Phân tích `api.js`

```
js
```

```
Sao chépChỉnh sửa
import axios from "axios";

const API_URL = "http://localhost:3001/users";
```

Bạn đang sử dụng `json-server` hoặc backend RESTful tại `localhost:3001/users`.

### Các hàm API:

| Hàm API | Mô tả | HTTP Method |
|---|---|---|
| `getUsers()` | Lấy danh sách tất cả người dùng | `GET /users` |
| `addUser(data)` | Thêm người dùng mới | `POST /users` |
| `updateUser(id, data)` | Cập nhật người dùng có ID | `PATCH /users/:id` |
| `deleteUser(id)` | Xoá người dùng theo ID | `DELETE /users/:id` |

---

# 🔁 📎 Tổng hợp luồng hoạt động toàn ứng dụng

## 📌 1. Ứng dụng khởi chạy (`App.js`)

- Gọi `useEffect()` → `loadUsers()` → `getUsers()`
- Server trả danh sách user → `setUsers(...)` → render `UserList`

---

## 📌 2. Hiển thị danh sách người dùng (`UserList.js`)

- Nhận prop `users`
- Lặp qua và hiển thị danh sách `<li>` gồm `name`, `email`, nút "Sửa" và "Xoá"
- Gắn sự kiện `onEdit(user)` và `onDelete(user.id)` từ `App.js`

---

## 📌 3. Sửa người dùng (Edit)

1. Bấm nút "Sửa" → gọi `onEdit(user)` → trong `App.js`, `setSelectedUser(user)`
2. `selectedUser` được truyền vào `UserForm`
3. `useEffect()` trong `UserForm` chạy → cập nhật `formData`
4. Người dùng sửa thông tin rồi bấm "Cập nhật"
5. Gọi `onSubmit(formData)` → trong `App.js`, nếu có `data.id` → gọi:

```
js
Sao chépChỉnh sửa
updateUser(data.id, data)
```

6. Sau khi API thành công → `loadUsers()` để làm mới danh sách

---

### 📌 4. Thêm người dùng (Add)

1. Form ban đầu rỗng
2. Người dùng nhập tên & email → `formData` được cập nhật
3. Bấm "Thêm mới" → gọi `onSubmit(formData)`
4. Trong `App.js`, nếu không có `data.id` → gọi:

```
js
Sao chépChỉnh sửa
addUser(data)
```

5. Sau khi thêm thành công → gọi lại `loadUsers()`

---

### 📌 5. Xoá người dùng (Delete)

1. Bấm nút "Xoá" → gọi `onDelete(user.id)`
2. Trong `App.js`, gọi:

```
js
Sao chépChỉnh sửa
deleteUser(id)
```

3. Sau khi xoá xong → gọi lại `loadUsers()`

---

## ✅ Tổng sơ đồ luồng dữ liệu:

```
plaintext
Sao chépChỉnh sửa
              App (App.js)
                  |
     ------------------------------
     |                            |
UserForm                      UserList
(Thêm/Sửa)                      (Hiển thị danh sách)

     ↑                            ↓
    setSelectedUser()      Click Sửa/Xoá gọi onEdit/onDelete
     |                            ↓
  selectedUser      ←---------    onDelete(id)
     ↓                            ↓
```

```
useEffect cập nhật form          deleteUser(id) → loadUsers()
      ↓
Gõ input → setFormData
      ↓
Submit → onSubmit()
      ↓
Nếu có id: updateUser(id, data)
Nếu không: addUser(data)
      ↓
→ loadUsers() → getUsers() → setUsers()
      ↓
→ cập nhật danh sách người dùng
```

---

# 🧪 Bạn có thể thử kiểm tra gì?

- ✅ Gọi API đúng chưa? (`localhost:3001/users`)
- ✅ Bạn đang dùng `json-server`? → cần chạy bằng:

  ```
  bash
  Sao chépChỉnh sửa
  json-server --watch db.json --port 3001
  ```

- ✅ Cấu trúc dữ liệu JSON trong `db.json` phải là:

  ```
  json
  Sao chépChỉnh sửa
  {
    "users": [
      { "id": 1, "name": "An", "email": "an@example.com" },
      ...
    ]
  }
  ```

# 💡 Mục tiêu ứng dụng:

Cho phép người dùng **thêm, sửa, xoá và xem danh sách người dùng**, dữ liệu lưu trên **JSON Server** (REST API tại `localhost:3001/users`).

---

# 🔄 1. Khi ứng dụng khởi chạy (`App.js`)

## 🎯 Mục tiêu:

Tải danh sách người dùng từ server.

### 🔗 Chi tiết:

- `useEffect(() => loadUsers(), [])` được gọi ngay sau khi component `App` mount.
- `loadUsers()` gọi `getUsers()` trong `api.js`:

  ```
  js
  Sao chépChỉnh sửa
  axios.get("http://localhost:3001/users")
  ```

- Khi dữ liệu trả về (`res.data`), `setUsers(...)` cập nhật state.
- `users` được truyền xuống `UserList` để hiển thị.

---

## 📄 2. Hiển thị danh sách người dùng (`UserList.js`)

### 🎯 Mục tiêu:

Cho phép xem, sửa, xoá từng người dùng.

### 🔗 Chi tiết:

- Với mỗi phần tử trong `users`, hiển thị:

  ```
  css
  Sao chépChỉnh sửa
  Tên - Email [Sửa] [Xoá]
  ```

- Nút **Sửa** gọi `onEdit(user)` → hàm `handleEdit()` trong `App` → cập nhật `selectedUser`.
- Nút **Xoá** gọi `onDelete(id)` → `handleDelete()` trong `App` → gọi API `deleteUser(id)` và reload danh sách.

---

## ✍️ 3. Sửa người dùng (`UserForm.js` với selectedUser)

### 🎯 Mục tiêu:

Cho phép sửa người dùng đã chọn.

### 🔗 Chi tiết:

- Khi `selectedUser` thay đổi, `useEffect` trong `UserForm` chạy và set form:

```
js
Sao chépChỉnh sửa
setFormData(selectedUser);
```

- Người dùng chỉnh sửa thông tin rồi nhấn nút "Cập nhật".
- Gọi `onSubmit(formData)` → trong `App`:
  - Có `data.id` → gọi `updateUser(id, data)`
  - Sau đó gọi lại `loadUsers()` để cập nhật danh sách.

---

# ➕ 4. Thêm người dùng mới (`UserForm.js` không có selectedUser)

## 🎯 Mục tiêu:

Nhập form trống để thêm user mới.

## 🔧 Chi tiết:

- `formData` ban đầu là `{ name: "", email: "" }`.
- Khi người dùng nhập xong và bấm "Thêm mới":
  - Gọi `onSubmit(formData)` trong `App`.
  - Vì không có `id` → gọi `addUser(data)` trong `api.js` → POST `/users`
  - Sau khi thêm thành công, gọi `loadUsers()`.

---

# 🗑️ 5. Xoá người dùng

## 🎯 Mục tiêu:

Xoá người dùng khỏi danh sách và server.

## 🔧 Chi tiết:

- Trong `UserList`, người dùng click "Xoá" → gọi `onDelete(id)`
- Trong `App`, `handleDelete(id)` gọi:

```
js
Sao chépChỉnh sửa
deleteUser(id).then(loadUsers);
```

- Gọi `DELETE /users/:id` qua Axios → Khi thành công thì tải lại danh sách.

---

## 🧠 Tóm tắt tổng quan theo dòng sự kiện:

```
plaintext
Sao chépChỉnh sửa
1. App mount → useEffect → loadUsers → GET /users → hiển thị danh sách

2. User click "Thêm mới":
   - Điền form → Submit → addUser → POST /users → loadUsers

3. User click "Sửa":
   - App setSelectedUser → form auto-fill → Submit → updateUser → PATCH
/users/:id → loadUsers

4. User click "Xoá":
   - Gọi deleteUser → DELETE /users/:id → loadUsers
```

---

## ✅ Thành phần chính & vai trò

| Thành phần | Vai trò |
|---|---|
| `App.js` | Trung tâm điều phối – gọi API, quản lý state |
| `UserForm.js` | Nhận `selectedUser`, hiển thị form thêm/sửa |
| `UserList.js` | Hiển thị danh sách, gọi callback `onEdit` / `onDelete` |
| `api.js` | Đóng gói các hàm gọi REST API |
| `json-server` | Làm giả backend lưu dữ liệu người dùng |