# Thông tin chung

* Họ và tên: Huỳnh Khang Vỹ
* Đề tài thực tập: Tiếp tục với UI lần trước tìm hiểu và ứng dụng i18n và redux
* Thời gian bắt đầu : 01/07/2025
* Thời gian kết thúc : 02/07/2025
* Đơn vị thực tập: FPT Software
* Người hướng dẫn: Lê Bá Quyền

**Frontend User Management System**

# Tổng quan

Đây là ứng dụng quản lý người dùng được phát triển với các công nghệ hiện đại như React, Redux Toolkit và Ant Design. Ứng dụng cung cấp giao diện thân thiện với người dùng để quản lý danh sách người dùng với các tính năng như tìm kiếm, lọc, sắp xếp và phân trang.

# Cấu trúc source code

- public/

* vite.svg
* locales/ – Tệp ngôn ngữ đa ngôn ngữ (i18n)
  + en/ – Tiếng Anh
  + fr/ – Tiếng Pháp
  + vi/ – Tiếng Việt

- src/

* api/ – Cấu hình API
  + axios.ts – Cấu hình Axios cho HTTP requests
* assets/ – Hình ảnh, fonts và tài nguyên khác
* components/ – Components tái sử dụng
  + ColumnsSelection.tsx – Lựa chọn cột hiển thị trong bảng
  + DeleteConfirm.tsx – Hộp xác nhận xóa
  + Loading.tsx – Hiển thị trạng thái đang tải
  + ResetFilter.tsx – Nút reset bộ lọc
  + SearchInput.tsx – Trường tìm kiếm
  + SwitchLanguage.tsx – Chuyển đổi ngôn ngữ
  + UserForm.tsx – Form thêm/sửa người dùng
  + UserTable.tsx – Bảng hiển thị danh sách người dùng
  + styles/ – Styled components
    - Container.style.tsx – Component container
    - CustomButton.style.tsx – Button tùy chỉnh
    - Header.style.tsx – Header tùy chỉnh
* features/ – Redux slices theo tính năng
  + language/ – Quản lý ngôn ngữ
    - languageSlice.ts
  + user/ – Quản lý người dùng
    - userApiSlice.ts – API queries (RTK Query)
    - userColumnsSlice.ts – State quản lý cột
    - userTableSlice.ts – State quản lý bảng
* hooks/ – Custom hooks
  + useGetApiParams.ts – Hook xử lý tham số API
  + useSetDefaultParams.ts – Hook thiết lập tham số mặc định
* providers/ – Providers React
  + AppWithLocale.tsx – Provider i18n
  + AppWithRouter.tsx – Provider React Router
* types/ – Định nghĩa kiểu TypeScript
  + global.d.ts – Các kiểu global
* utils/ – Tiện ích
  + i18n.ts – Cấu hình i18next
  + locate.ts – Xử lý ngôn ngữ và địa phương
  + userTableParams.ts – Xử lý tham số URL cho bảng
* App.css
* App.tsx
* index.css
* main.tsx
* store.ts – Cấu hình Redux store
* vite-env.d.ts

# Công nghệ và thư viện sử dụng

## Công nghệ chính

* **React**: Thư viện JavaScript để xây dựng giao diện người dùng
* **TypeScript**: Ngôn ngữ lập trình JavaScript với kiểu dữ liệu tĩnh
* **Vite**: Công cụ build nhanh cho các ứng dụng web hiện đại
* **Redux Toolkit**: Thư viện quản lý state toàn cục
* **React Router**: Thư viện định tuyến cho React
* **i18next**: Thư viện hỗ trợ đa ngôn ngữ

## Thư viện UI và component

* **Ant Design**: Hệ thống design và component cho React
* **date-fns**: Thư viện xử lý ngày tháng
* **use-debounce**: Hook để debounce các giá trị và sự kiện

## Thư viện xử lý API

* **RTK Query**: Công cụ quản lý dữ liệu API tích hợp với Redux Toolkit
* **Axios**: Thư viện HTTP client

# Mô tả UI và tính năng

## 1. Header

* **Chuyển đổi ngôn ngữ**: Hỗ trợ đa ngôn ngữ (Anh, Pháp, Việt)
* **Tiêu đề ứng dụng**: Hiển thị tiêu đề chính của ứng dụng
* **Nút thêm người dùng**: Cho phép thêm người dùng mới

## 2. Khu vực tìm kiếm và lọc

* **Ô tìm kiếm**: Tìm kiếm theo tên hoặc thông tin khác
* **Nút reset filter**: Xóa tất cả các bộ lọc đang áp dụng

## 3. Bảng người dùng

* **Cấu trúc bảng**:
  + Cột Tên: Hiển thị tên người dùng
  + Cột Thông tin: Hiển thị avatar, email và giới tính
  + Cột Thành phố: Hiển thị thành phố
  + Cột Ngày tạo: Hiển thị ngày tạo với định dạng phù hợp với ngôn ngữ
  + Cột Hành động: Chứa các nút để chỉnh sửa và xóa người dùng
* **Tính năng bảng**:
  + Sắp xếp: Cho phép sắp xếp theo tên, email, và ngày tạo
  + Lọc: Lọc theo giới tính (nam/nữ)
  + Phân trang: Phân trang với khả năng thay đổi số lượng hiển thị
  + Lựa chọn cột: Tùy chỉnh các cột hiển thị trong bảng
  + Lưu trạng thái: Lưu trạng thái bảng vào URL để chia sẻ và quay lại sau

## 4. Form người dùng

* **Thêm mới**: Form để thêm người dùng mới
* **Chỉnh sửa**: Form để cập nhật thông tin người dùng
* **Xác nhận xóa**: Hộp thoại xác nhận trước khi xóa người dùng

# Luồng dữ liệu

1. **Dữ liệu từ API**:
   * RTK Query gọi API để lấy dữ liệu người dùng
   * Tham số API được xây dựng từ state trong Redux
2. **Quản lý state**:
   * Sử dụng Redux để lưu trữ state toàn cục (ngôn ngữ, cài đặt bảng)
   * Các thay đổi state được dispatch thông qua actions
   * RTK Query quản lý cache và trạng thái tải dữ liệu
3. **Đồng bộ ba chiều**:
   * **Đồng bộ URL với Redux**: Tất cả các thay đổi state (sắp xếp, lọc, tìm kiếm) được đồng bộ vào URL và ngược lại
   * **Đồng bộ URL với Backend**: Các tham số từ URL được chuyển đổi thành tham số API để gửi đến backend
   * **Đồng bộ URL với Ant Design**: Các tham số URL được áp dụng vào các component Ant Design (Table, Filter, Sorter)
   * Đảm bảo trạng thái đồng nhất giữa URL, giao diện người dùng và dữ liệu từ backend
   * Khi tải lại trang, state được khôi phục từ URL và áp dụng cho cả Redux và Ant Design
   * Lưu lịch sử thao tác với trạng thái { replace: false }
4. **Đồng bộ ngôn ngữ**:
   * Ngôn ngữ đồng bộ giữa i18next và Ant Design
   * Locale của date-fns cũng được đồng bộ với ngôn ngữ hiện tại
   * Format ngày tháng theo định dạng phù hợp với ngôn ngữ đã chọn

# Tính năng đặc biệt

1. **Đa ngôn ngữ và đồng bộ locale**:
   * Hỗ trợ chuyển đổi nhanh giữa nhiều ngôn ngữ (Anh, Pháp, Việt)
   * Đồng bộ locale giữa i18next, Ant Design và date-fns
   * Định dạng ngày tháng, số và các phần tử UI thay đổi theo ngôn ngữ
2. **Đồng bộ state ba chiều**:
   * Đồng bộ giữa URL - Redux State - Ant Design UI
   * Bất kỳ thay đổi nào ở một trong ba phần đều được phản ánh ở hai phần còn lại
   * Đảm bảo trải nghiệm người dùng nhất quán khi tải lại trang hoặc chia sẻ URL
3. **Responsive**:
   * Giao diện thích ứng với nhiều kích thước màn hình
   * Tối ưu hiển thị trên thiết bị di động và máy tính bảng
4. **Tối ưu hiệu suất**:
   * Sử dụng useMemo để tối ưu việc render lại
   * Debounce input tìm kiếm để giảm số lần gọi API
   * Quản lý trạng thái tải hợp lý (loading, fetching)
   * Sử dụng các hook tùy chỉnh để tách logic và tái sử dụng mã

# Phân tích thư viện: Ưu nhược điểm và so sánh

## Redux Toolkit so với các giải pháp quản lý state khác

#### Redux Toolkit

**Ưu điểm:**

* Giảm thiểu boilerplate code so với Redux truyền thống
* Tích hợp sẵn các middleware phổ biến (thunk, immer)
* RTK Query giúp đơn giản hóa việc gọi API và quản lý cache
* TypeScript-first, hỗ trợ tốt type safety
* Phù hợp với ứng dụng phức tạp có nhiều state toàn cục

**Nhược điểm:**

* Khá nặng cho các ứng dụng nhỏ
* Setup ban đầu tốn thời gian
* Có thể dư thừa cho các ứng dụng đơn giản

#### So sánh với các giải pháp khác:

|  |  |  |  |
| --- | --- | --- | --- |
| Thư viện | Ưu điểm | Nhược điểm | Phù hợp cho |
| **Redux Toolkit** | • Giảm boilerplate code  • Tích hợp middleware sẵn  • RTK Query mạnh mẽ  • TypeScript-first  • Dev-tools tốt | • Bundle size lớn  • Setup phức tạp | Ứng dụng phức tạp, nhiều state, yêu cầu cache thông minh |
| **Context API** | • Built-in React  • Nhẹ hơn  • Dễ học, setup đơn giản | • Hiệu suất kém với state phức tạp  • Không có dev-tool  • Không có cache API | Ứng dụng nhỏ, state đơn giản, ít cập nhật |
| **Zustand** | • API đơn giản  • Bundle size nhỏ  • Re-render tối ưu | • Không có RTK Query  • Ecosystem nhỏ hơn  • Ít tính năng phức tạp | Ứng dụng vừa, quan tâm đến bundle size |

## Ant Design so với các thư viện UI khác

#### Ant Design

**Ưu điểm:**

* Hệ thống component phong phú và đầy đủ
* Table component mạnh mẽ với nhiều tính năng sẵn có
* TypeScript support tốt
* Hỗ trợ i18n và theme tùy chỉnh
* Documentation chi tiết
* Cộng đồng lớn và active

**Nhược điểm:**

* Bundle size lớn (có thể tối ưu bằng import theo nhu cầu)
* Phong cách thiết kế đặc trưng, khó tùy biến hoàn toàn
* Performance có thể chậm ở một số component phức tạp
* Đôi khi khó tùy chỉnh CSS sâu

#### So sánh với các thư viện UI khác:

|  |  |  |  |
| --- | --- | --- | --- |
| Ant Design | • Component phong phú  • Table mạnh mẽ, đầy đủ tính năng  • TypeScript support tốt  • Cộng đồng lớn  • Documentation chi tiết | • Bundle size lớn  • Phong cách thiết kế khó tùy biến  • Performance có thể chậm | Ứng dụng doanh nghiệp, bảng dữ liệu phức tạp |
| **Material-UI** | • Material Design của Google  • Hệ thống Theme mạnh mẽ  • Hỗ trợ TypeScript tốt | • Bundle size lớn  • Table kém phong phú  • Cần nhiều code tùy biến | Ứng dụng theo phong cách Google, tùy biến theme |
| **Chakra UI** | • API đơn giản  • Accessibility tốt  • Bundle size nhỏ hơn | • Thiếu component phức tạp  • Table còn hạn chế  • Ecosystem nhỏ hơn | Ứng dụng đơn giản, cần tùy biến cao |
| **TailwindCSS + Headless UI** | • Tự do tùy chỉnh UI  • Bundle size nhỏ  • Kiểm soát tối đa | • Phải tự xây dựng tính năng phức tạp  • Học ban đầu khó  • Cần nhiều thời gian phát triển | Dự án cần UI độc đáo, tùy biến cao |

## i18next so với các giải pháp đa ngôn ngữ khác

#### i18next

**Ưu điểm:**

* Hỗ trợ đầy đủ các tính năng i18n phức tạp
* Ecosystem phong phú (react-i18next, i18next-http-backend…)
* Hỗ trợ plural, context, interpolation
* Dễ dàng tích hợp với backend
* TypeScript support tốt

**Nhược điểm:**

* Cấu hình ban đầu khá phức tạp
* Bundle size lớn hơn các giải pháp nhẹ
* Có thể quá mức cần thiết cho các ứng dụng đơn giản

#### So sánh với các giải pháp i18n khác:

|  |  |  |  |
| --- | --- | --- | --- |
| Thư viện | Ưu điểm | Nhược điểm | Phù hợp cho |
| **i18next** | • Tính năng i18n đầy đủ  • Ecosystem phong phú  • Hỗ trợ plural, context  • TypeScript support tốt | • Cấu hình phức tạp  • Bundle size lớn  • Có thể quá mức cho app đơn giản | Ứng dụng đa ngôn ngữ phức tạp, nhiều ngôn ngữ |
| **react-intl** | • Là một phần của FormatJS  • API đơn giản  • Hiệu suất tốt | • Ít tính năng cao cấp  • Ít plugin mở rộng  • Không linh hoạt bằng | Ứng dụng đa ngôn ngữ đơn giản đến trung bình |
| **LinguiJS** | • Trích xuất messages tự động  • Tích hợp công cụ dịch  • Hiệu suất cao | • Ecosystem nhỏ  • Ít phổ biến  • Hỗ trợ cộng đồng hạn chế | Ứng dụng cần trích xuất text tự động từ code |

## RTK Query so với các thư viện gọi API khác

#### RTK Query

**Ưu điểm:**

* Tích hợp sẵn với Redux Toolkit
* Quản lý cache và invalidation thông minh
* Tự động dedupe requests
* Tạo hooks tự động cho việc gọi API
* TypeScript support tốt
* Hỗ trợ optimistic updates

**Nhược điểm:**

* Yêu cầu sử dụng Redux Toolkit
* Cấu hình ban đầu phức tạp
* Bundle size lớn hơn các giải pháp chuyên biệt

#### So sánh với các thư viện gọi API khác:

|  |  |  |  |
| --- | --- | --- | --- |
| Thư viện | Ưu điểm | Nhược điểm | Phù hợp cho |
| **RTK Query** | • Tích hợp sẵn Redux Toolkit  • Cache và invalidation thông minh  • Auto-generated hooks  • Optimistic updates  • Dedupe requests | • Yêu cầu Redux Toolki  t• Cấu hình ban đầu phức tạp  • Bundle size lớn | Ứng dụng đã dùng Redux, cần quản lý API phức tạp |
| **React Query** | • Độc lập với state management  • API đơn giản  • Cache tương tự RTK Query  • Bundle size nhỏ hơn | • Không tích hợp với Redux  • Cần setup riêng  • Devtools ít mạnh mẽ hơn | Ứng dụng không dùng Redux, cần data fetching linh hoạt |
| **SWR** | • API cực kỳ đơn giản  • Tập trung vào fetch/cache  • Bundle size nhỏ nhất  • Dễ học | • Ít tính năng phức tạp  • Mutation ít mạnh mẽ  • Cache ít tùy biến | Ứng dụng nhỏ, cần data fetching đơn giản |
| **Apollo Client** | • Mạnh mẽ với GraphQL  • Cache phức tạp mạnh mẽ  • Ecosystem lớn | • Bundle size lớn  • Quá mức cho REST API | Ứng dụng dùng GraphQL, cần cache phức tạp |

## Kết luận về lựa chọn thư viện

Dự án này đã lựa chọn bộ công nghệ cân bằng giữa tính năng phong phú và hiệu suất. Redux Toolkit + RTK Query cung cấp giải pháp mạnh mẽ cho state management và API, trong khi Ant Design mang đến UI components đầy đủ tính năng cho table phức tạp. i18next đảm bảo hỗ trợ đa ngôn ngữ mạnh mẽ.

Những lựa chọn này phù hợp với yêu cầu của ứng dụng quản lý người dùng có nhiều tính năng phức tạp như filtering, sorting và i18n. Tuy bundle size có thể lớn hơn các giải pháp nhẹ, nhưng đổi lại là tính năng phong phú và thời gian phát triển nhanh hơn.

# Đề xuất cải tiến

Dưới đây là một số đề xuất để cải tiến ứng dụng trong tương lai:

## 1. Tối ưu hiệu suất

|  |  |  |
| --- | --- | --- |
| Đề xuất cải tiến | Mô tả | Lợi ích |
| **Code splitting** | Tách code thành các bundle nhỏ hơn, chỉ tải khi cần | • Giảm thời gian tải ban đầu  • Cải thiện First Contentful Paint |
| **Tree shaking Ant Design** | Import có chọn lọc từ Ant Design thay vì import toàn bộ | • Giảm bundle size  • Cải thiện thời gian tải |
| **Virtualized List** | Sử dụng react-window hoặc react-virtualized cho bảng dữ liệu lớn | • Hiệu suất tốt hơn với dữ liệu lớn  • Giảm memory usage |
| **Memo hóa components** | Sử dụng memo và useCallback cho các component con thường xuyên re-render | • Giảm số lượng re-render không cần thiết |

## 2. Cải thiện trải nghiệm người dùng

|  |  |  |
| --- | --- | --- |
| Đề xuất cải tiến | Mô tả | Lợi ích |
| **Skeleton loading** | Hiển thị skeleton trong khi tải dữ liệu thay vì spinner | • Cảm giác tải nhanh hơn• Giảm cảm giác gián đoạn |
| **Export dữ liệu** | Cho phép xuất dữ liệu ra Excel, CSV hoặc PDF | • Tính năng nghiệp vụ cần thiết• Tăng tính tiện dụng |
| **Chế độ dark/light** | Thêm tùy chọn chuyển đổi giao diện tối/sáng | • Cải thiện trải nghiệm trong các điều kiện ánh sáng khác nhau |

## 3. Cải tiến kiến trúc và mã nguồn

## 4. Tính năng mới

|  |  |  |
| --- | --- | --- |
| Tính năng | Mô tả | Lợi ích |
| **Authentication/Authorization** | Thêm hệ thống xác thực và phân quyền | • Bảo mật  • Hạn chế quyền truy cập |
| **Thống kê và Dashboard** | Thêm biểu đồ và dashboard hiển thị số liệu | • Phân tích dữ liệu  • Hỗ trợ ra quyết định |
| **Nhập dữ liệu hàng loạt** | Cho phép import người dùng từ file | • Tự động hóa  • Tiết kiệm thời gian |
| **Tìm kiếm nâng cao** | Tìm kiếm với nhiều điều kiện phức tạp | • Tìm kiếm chính xác hơn  • Lọc dữ liệu hiệu quả |