**BÁO CÁO 6: TỐI ƯU TODOS VÀ TÌM HIỂU VỀ PHÂN TRANG TRONG ANT DESIGN**

1. **Tối ưu todos**
   1. **Vấn đề về component cha và con và thêm proptypes vào các component**

* Hiện tại các xử lý logic chính nằm ở trong component con, vì thế để tiện cho việc dò lỗi và update, nên chuyển tất cả các logic đó cho component cha và các component con chỉ nhận props từ component cha.
* Để đảm bảo cho việc mỗi props truyền vô component là chính xác, việc thêm proptypes vào các components là cần thiết. Ví dụ :

import PropTypes from "prop-types";

TodoFilter.propTypes = {

  numItem: PropTypes.number,

  clearCompleted: PropTypes.func,

  onClickAll: PropTypes.func,

  onClickActive: PropTypes.func,

  onClickCompleted: PropTypes.func,

};

* 1. **Vấn đề về tối ưu logic update và search**
* Mỗi lần update và search trong table đều phải get data lại để hiển thị trong màn hình. Điều này làm chương trình chạy rất chậm và không hiệu quả.
* Trong Mockapi, khi ta thực hiện một method bất kì nó sẽ trả về data của method đó, chính vì thế, việc lấy data từ response của Mockapi để hiển thị data sẽ giúp cải thiện tốc độ rất nhiều
  1. **Vấn đề về validate data**
* Trong Ant Design đã cũng cấp 1 số tính năng để validate data khi nhập vào bảng. Ví dụ trong một form thì props “rules” sẽ quy định data khi nhập vô sẽ thuộc kiểu nào.

rules={[

{

   required: true,

    pattern: new RegExp("([0-9]\\s\*)+"),

    message: "please input phone num",

  },

1. **Phân trang trong Ant Design**
   1. **Tại sao phải phân trang ?**

* Trong table của Ant Design đã tích hợp sẵn công cụ phân trang. Chỉ cần truyền dữ liệu cho table thì lập tức nó sẽ phân trang theo dữ liệu đó.
* Tuy nhiên với một dữ liệu cực lớn (bảng có 1 triệu dòng chẳng hạn) việc load toàn bộ data này sẽ rất mất thời gian khi tải trang. Chính vì thế nên phân thành những trang nhỏ để mỗi lần chuyển trang sẽ lấy đúng data mà trang đó có.
  1. **Cách phân trang khi lấy dữ liệu từ api**
* Phân trang trong Ant Design cần thuộc tính “pagination”, ngoài ra, cần thêm thuộc tính “onChange” để có thể xử lý logic khi chuyển trang.

<Table

        columns={columns}

        dataSource={data}

        pagination={tablePagination}

        loading={loading}

        onChange={handleTableChange}

      />

* Đây là hàm xử lý khi thay đổi một trang. Hàm fetch() có nhiệm vụ lấy dữ liệu theo trang được truyền vô

const handleTableChange = (page, pageSize) => {

    let currentPage = {

      current: (page.current - 1) \* page.pageSize,

      pageSize: pageSize,

    };

    console.log(page);

    fetch(currentPage);

  };

const tablePagination = {

    pageSize: paramsContact.limit,

    page: paramsContact.offset / paramsContact.limit + 1,

    total: paramsContact.total,

    pageSizeOptions: ["5", "10", "15", "20", "25"],

    showSizeChanger: true,

  };