

Ý tưởng Dự án: Ứng dụng Quản lý Công Việc (To-Do List)
Ứng dụng này cho phép người dùng:

Thêm, sửa, xóa công việc.
Đánh dấu công việc đã hoàn thành.
Lưu trữ công việc vào một backend giả lập (API giả).
Lọc công việc theo trạng thái (tất cả, đã hoàn thành, chưa hoàn thành).
Sử dụng vite react,typescript
đã khởi tạo dự án
Cho tôi cấu trúc và hướng phát triển để thực hiện các kiến thức
react như state,useState,useEffect, react hook, ...

Dưới đây là cấu trúc và hướng phát triển cho dự án **Ứng dụng Quản lý Công Việc (To-Do List)** sử dụng Vite, React, và TypeScript. Mục tiêu là tận dụng các kiến thức React như `state`, `useState`, `useEffect`, và React Hooks.

1. Cấu trúc thư mục

```
css
src/
├── components/      # Các thành phần giao diện (React Components)
|   ├── TodoForm.tsx # Form để thêm/đổi công việc
|   ├── TodoItem.tsx # Một mục công việc
|   └── TodoList.tsx # Danh sách các công việc
└── hooks/           # Custom hooks (nếu cần)
    └── useTodos.ts  # Custom hook để quản lý logic của todo
├── services/         # API giả lập
|   └── fakeApi.ts   # Module mô phỏng API CRUD
├── types/            # Các kiểu dữ liệu (TypeScript types/interfaces)
|   └── Todo.ts       # Định nghĩa kiểu Todo
└── App.tsx          # Thành phần chính
└── main.tsx          # Điểm vào của ứng dụng
```

2. Các bước phát triển

2.1. Định nghĩa kiểu dữ liệu (`Todo.ts`)

```
typescript
export interface Todo {
  id: number;
  title: string;
  completed: boolean;
}
```

2.2. API giả lập (`fakeApi.ts`)

```
typescript
import { Todo } from '../types/Todo';

let todos: Todo[] = [];

export const fakeApi = {
  getTodos: async (): Promise<Todo[]> => {
    return Promise.resolve(todos);
  },
  addTodo: async (todo: Todo): Promise<Todo> => {
    todos.push(todo);
    return Promise.resolve(todo);
  },
  updateTodo: async (updatedTodo: Todo): Promise<Todo> => {
    todos = todos.map((todo) =>
      todo.id === updatedTodo.id ? updatedTodo : todo
    );
    return Promise.resolve(updatedTodo);
  },
  deleteTodo: async (id: number): Promise<void> => {
    todos = todos.filter((todo) => todo.id !== id);
    return Promise.resolve();
  },
};
```

2.3. Xây dựng giao diện

2.3.1. Form thêm/sửa công việc (`TodoForm.tsx`)

```
typescript
import React, { useState } from 'react';

interface TodoFormProps {
  onSubmit: (title: string) => void;
```

```
}

const TodoForm: React.FC<TodoFormProps> = ({ onSubmit }) => {
  const [title, setTitle] = useState('');

  const handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    if (title.trim()) {
      onSubmit(title);
      setTitle('');
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        placeholder="Thêm công việc..."
        value={title}
        onChange={(e) => setTitle(e.target.value)}
      />
      <button type="submit">Thêm</button>
    </form>
  );
};

export default TodoForm;
```

2.3.2. Mục công việc (TodoItem.tsx)

```
typescript
Copy code

import React from 'react';
import { Todo } from '../types/Todo';

interface TodoItemProps {
  todo: Todo;
  onToggle: (id: number) => void;
  onDelete: (id: number) => void;
}

const TodoItem: React.FC<TodoItemProps> = ({ todo, onToggle, onDelete }) => {
  return (
    <div>
      <input
        type="checkbox"
        checked={todo.completed}
        onChange={() => onToggle(todo.id)}
      />
      <span style={{ textDecoration: todo.completed ? 'line-through' : 'none' }}>
        {todo.title}
      </span>
      <button onClick={() => onDelete(todo.id)}>Xóa</button>
    </div>
  );
};

export default TodoItem;
```

2.3.3. Danh sách công việc (TodoList.tsx)

```
typescript
Copy code

import React from 'react';
import { Todo } from '../types/Todo';
import TodoItem from './TodoItem';

interface TodoListProps {
  todos: Todo[];
  onToggle: (id: number) => void;
  onDelete: (id: number) => void;
}

const TodoList: React.FC<TodoListProps> = ({ todos, onToggle, onDelete }) => {
  return (
    <div>
      {todos.map((todo) => (
        <TodoItem
          key={todo.id}
          todo={todo}
          onToggle={onToggle}
          onDelete={onDelete}
        />
      ))}
    </div>
  );
};

export default TodoList;
```

2.3.4. Bộ lọc công việc (FilterBar.tsx)

```
typescript
Copy code
```

```

import React from 'react';

interface FilterBarProps {
  filter: string;
  setFilter: (filter: string) => void;
}

const FilterBar: React.FC<FilterBarProps> = ({ filter, setFilter }) => {
  return (
    <div>
      <button onClick={() => setFilter('all')}>Tất cả</button>
      <button onClick={() => setFilter('completed')}>Đã hoàn thành</button>
      <button onClick={() => setFilter('active')}>Chưa hoàn thành</button>
    </div>
  );
};

export default FilterBar;

```

2.4. Tích hợp trong App.tsx

```

typescript
Copy code

import React, { useEffect, useState } from 'react';
import { fakeApi } from './services/fakeApi';
import { Todo } from './types/Todo';
import TodoForm from './components/TodoForm';
import TodoList from './components/TodoList';
import FilterBar from './components/FilterBar';

const App: React.FC = () => {
  const [todos, setTodos] = useState<Todo[]>([]);
  const [filter, setFilter] = useState('all');

  useEffect(() => {
    fakeApi.getTodos().then(setTodos);
  }, []);

  const handleAddTodo = (title: string) => {
    const newTodo: Todo = {
      id: Date.now(),
      title,
      completed: false,
    };
    fakeApi.addTodo(newTodo).then((todo) => setTodos([...todos, todo]));
  };

  const handleToggleTodo = (id: number) => {
    const updatedTodo = todos.find((todo) => todo.id === id);
    if (updatedTodo) {
      updatedTodo.completed = !updatedTodo.completed;
      fakeApi.updateTodo(updatedTodo).then(() =>
        setTodos(todos.map((todo) => (todo.id === id ? updatedTodo : todo)))
      );
    }
  };

  const handleDeleteTodo = (id: number) => {
    fakeApi.deleteTodo(id).then(() => setTodos(todos.filter((todo) => todo.id !== id)));
  };

  const filteredTodos = todos.filter((todo) => {
    if (filter === 'completed') return todo.completed;
    if (filter === 'active') return !todo.completed;
    return true;
  });

  return (
    <div>
      <h1>Quản lý công việc</h1>
      <TodoForm onSubmit={handleAddTodo} />
      <FilterBar filter={filter} setFilter={setFilter} />
      <TodoList
        todos={filteredTodos}
        onToggle={handleToggleTodo}
        onDelete={handleDeleteTodo}
      />
    </div>
  );
};

export default App;

```

3. Mục tiêu học tập

1. State Management:

- Sử dụng `useState` để quản lý danh sách công việc, trạng thái bộ lọc.

2. Side Effects:

- Dùng `useEffect` để tải công việc từ API gián tiếp.

3. Props Drilling:

- Truyền các hàm và trạng thái qua nhiều cấp (giữa các components).

4. Custom Hooks:

- Tách logic phức tạp (nếu cần) vào một custom hook (`useTodos`).

Bạn có thể mở rộng thêm tính năng như tìm kiếm, sắp xếp, hoặc kết nối với API thực.

⋮ ⌂ ⌃ ⌄ ⌅ ⌆

Message ChatGPT

✉ 📲 ⓘ



ChatGPT can make mistakes. Check important info.

?