**Общая структура React-проекта**

src/

├── assets/ # Статические файлы (изображения, шрифты, иконки, стили)

├── components/ # Переиспользуемые UI-компоненты

├── pages/ # Страницы приложения

├── hooks/ # Пользовательские хуки

├── context/ # Контекст (Context API)

├── services/ # Работа с API и внешними сервисами

├── store/ # Состояние приложения (Redux/Zustand, и т. д.)

├── utils/ # Вспомогательные функции

├── types/ # Типы и интерфейсы (TypeScript)

├── App.tsx # Главный компонент приложения

├── index.tsx # Входная точка приложения

└── routes/ # Конфигурация маршрутов (React Router)

**1. 📂 assets/**

Папка для всех статических файлов, которые связаны с проектом:

Картинки, SVG-иконки, Шрифты, Глобальные стили (CSS, SCSS), Логотипы и прочее.

src/

└── assets/

    ├── images/

    │ ├── logo.png

    │ └── user-avatar.svg

    ├── fonts/

    │ └── Roboto-Regular.ttf

    └── styles/

        └── global.css

**2. 📂 components/**

Здесь находятся маленькие, многократно переиспользуемые UI-компоненты. Эти компоненты являются "глупыми" то есть только отображают данные, которые получают через props, без сложной логики.

Папка может быть разбита на подкатегории по функционалу.

src/

└── components/

    ├── Button/

    │ ├── Button.tsx

    │ ├── Button.module.css

    │ └── index.tsx

    ├── Input/

    │ ├── Input.tsx

    │ ├── Input.module.css

    │ └── index.tsx

    └── Modal/

        ├── Modal.tsx

        ├── Modal.module.css

        └── index.tsx

В каждом компоненте можно использовать отдельно стили, тесты и даже подгруппы хелперов через подход "один компонент — одна папка".

**3. 📂 pages/**

Папка для компонентов "страниц" приложения. Страницы обычно представляют собой крупные компоненты, которые собирают вместе другие компоненты и имеют привязку к маршрутам (например, через React Router).

src/

└── pages/

    ├── Home/

    │ ├── Home.tsx # Страница "Главная"

    │ ├── index.tsx

    ├── Login/

    │ ├── Login.tsx # Страница авторизации

    │ ├── Login.module.css

    └── Profile/

        ├── Profile.tsx # Страница профиля

        ├── Profile.module.css

        └── index.tsx

Примечания:

Папку pages/ удобно использовать совместно с конфигурацией маршрутов, если приложение имеет маршрутизацию.

В каждой странице могут использоваться компоненты из папки components/.

**4. 📂 hooks/**

Здесь хранятся пользовательские хуки, которые реализуют общую логику для компонента или страницы.

Например: Управление состоянием. Обработка событий.

Интерфейс к API.

src/

└── hooks/

    ├── useFetch.ts # Хук для работы с fetch

    ├── useAuth.ts # Хук авторизации пользователя

    └── useWindowSize.ts # Хук для получения размеров окна

**5. 📂 context/**

Если вы используете React Context API для управления состоянием, логично вынести логику в папку context/.

src/

└── context/

    ├── AuthContext.tsx # Контекст для авторизации/пользователя

    ├── ThemeContext.tsx # Контекст для темы/стиля

    └── index.ts

**6. 📂 services/**

Это папка для взаимодействия с внешними API или сервисами. Тут могут находиться функции для обращения к серверам, либо обертки над HTTP-запросами.

src/

└── services/

    ├── api/

    │ ├── userApi.ts # Запросы связанные с пользователями

    │ └── productApi.ts # Запросы связанные с продуктами

    ├── axiosInstance.ts # Настройка Axios

    └── config.ts # Конфигурации API

**7. 📂 store/ (опционально)**

Если вы используете стороннюю библиотеку для управления состоянием, например Redux, папка store/ — это подходящее место для логики хранилища (редьюсеров, действий, селекторов).

src/

└── store/

    ├── slices/ # Срезы состояния (например, Redux toolkit)

    │ ├── userSlice.ts

    │ ├── productSlice.ts

    └── store.ts # Настройка хранилища

**8. 📂 utils/**

Для вспомогательных функций, которые многократно используются в проекте, например:

Форматирование дат.

Утилиты для валидации данных.

Логика для преобразования объектов.

src/

└── utils/

    ├── formatDate.ts # Функция форматирования даты

    ├── debounce.ts # Функция debounce

    └── validateEmail.ts # Валидация email

**9. 📂 types/**

Если вы используете TypeScript, в этой папке можно хранить объявления типов и интерфейсов, чтобы переиспользовать их в коде.

Пример:

src/

└── types/

    ├── User.ts # Интерфейсы пользователей

    ├── Product.ts # Интерфейсы продуктов

    └── ApiResponse.ts # Общие интерфейсы ответа API

**10. Корневые файлы**

App.tsx: Главный компонент приложения, который рендерит маршруты или собирает общий каркас (например, общий макет, шапка, подвал).

index.tsx: Точка входа в приложение (рендерит `` в DOM через ReactDOM).

Что важно учитывать:

Не переусложняйте:\*\* В небольших проектах не делайте слишком много вложенных папок. Лучше упростить структуру.

Папки по смыслу:\*\* Организуйте файлы так, чтобы они логически группировались.

Гибкость:\*\* Не существует единой идеальной структуры. Вы можете адаптировать её под свои потребности или требования команды.