***React JS*** – это *JavaScript* библиотека для создания пользовательских интерфейсов. Она является декларативной, что позволяет безболезненно создавать интерактивные интерфейсы. Для работы рекомендуется использовать *Vite* – инструмент для настройки сред разработки (сборщик модулей для *frontend*).

Для создания шаблона *react*-проекта с помощью *Vite* нужно в целевой папке проекта набрать команды:

*npm create vite@latest*

*// задать имя проекта, выбрать react и javaScript*

*cd react*

*npm install*

*npm run dev*

Для сборки пакета *Dist* для развертывания на сервере, используется команда:

*npm run build*

Изначально под каждую задачу был отдельный файл – для *html*-разметки, для *css*-стиля и для *javascript*-логики. Файлы с расширением *JSX* позволяют совмещать их функционал.

Основными строительными блоками *react* являются **компоненты**. Компоненты – это элементы кода, которые мы можем переиспользовать в нашем приложении. Названия компонент начинаются с большой буквы. Рассмотрим пример компонента *UserProfile*:

*const UserProfile = ({ name, age,gender, …props }) => {*

*return (*

*<div>*

*<h1>{name}</h1>*

*<p>{age}</p>*

*<img src={`https://randomuser.me/api/portraits/${gender}/78.jpg`}/>*

*{props.children}*

*</div>*

*);*

*}*

*export default UserProfile;*

Для передачи в функцию неограниченного количества элементов используется ***…props***.

***State hooks*** позволяют компоненту запоминать его текущее состояние. Например, компонент формы может хранить входное значение, а компонент галереи хранить индекс изображения.

*function CounterButton() {*

*const [counter, setCounter] = useState(0); // запоминаем состояние счетчика, начиная с нуля*

*return (*

*<div>*

*<button onClick = { () => setCounter(prevCounter => prevCounter + 1) } >*

*Increment counter // увеличиваем счетчик при нажатии*

*<>/button*

*<h1>{ counter }</h1>*

*</div>*

*)*

*}*

***Effect hook*** позволяет компоненту подключаться и синхронизироваться с внешними элементами. Например, работа с сетью, DOM браузера, анимация, сторонние виджеты и другой внешний код.

*function LoadingDelay() {*

*const [isLoading, setIsLoading] = useState(false);*

*useEffect( () => {*

*setIsLoading(true) // устанавливаем страницу загрузки на 4 секунды*

*setTimeout(() => {*

*setIsLoading(false)*

*}, 4000)*

*return () => {*

*console.log("cleanup functions") // освобождаем ресурсы компонента*

*}*

*}, [])*

*if (isLoading) {*

*return "loading..." // возвращаем страницу загрузки*

*}*

*return “Finish loading!” // возвращаем успешную страницу*

*}*

***Context hook*** позволяет компоненту получать информацию от удаленных родителей не передавая её в качестве реквизитов. Например, верхнеуровневый компонент может передавать визуальную тему всем вложенным компонентам независимо от их глубины. Значит у нас появились глобальные состояния для реквизитов, логина, пароля и токена.

Найти уже готовые *UI*-компоненты можно на тематических сайтах. Например, на <https://mui.com>, <https://ant.design/> или <https://chakra-ui.com/>.

Для установки библиотеки ***Chakra-UI*** перейдем в папку с проектом и выполним команду:

*npm i @chakra-ui/react @emotion/react @emotion/styled framer-motion*

Затем импортируем и используем ***ChakraProvider*** в приложении согласно офф. документации.

Для быстрого создания формы рекомендуется использовать шаблоны <https://formik.org/>.

Для настройки маршрутизации установим библиотеку ***react-router***:

*npm install react-router-dom@6.8.0*

Для редиректа на другую страницу используется команда ***useNavigate()***.

Для хранения информации можно использовать хешмапу ***localStorage***. Она встроена в *react*.

Для расшифровки информации из токена установим библиотеку ***jwt-decode***:

*npm install jwt-decode*