Vamos focar aqui na parte de funcionalidade do ReactJs, trabalharemos puramente com componentes funcionais, vamos entender como funciona os React hooks e outros componentes.

O **create-react-app** é **um atalho que permite a rapida criação de um app React que vai baixar tudo que é necessário para ter tudo que se precisa para começar um projeto React de maneira prática e rápida.** *npx create-react-app nome-do-arquivo*

O **readme** criado na pasta dessa forma contem diversos comando relacionados ao **npm**:

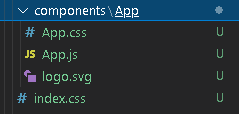
**npm start** roda o arquivo em modo de desenvolvedor no browser, o broswer padrão abre automaticamente quando se faz isso.

**npm test** roda o test runner em modo de interação visual.

**npm run build** torna o app para modo de produção em uma pasta **build**.

**npm run eject** retira uma build pronta do computador, **isso é irreversível**, mas uma outra build ainda pode ser feita.

A pasta **src** naturalmente é onde fica os arquivos de source do aplicativo, isso inclui testes de aplicativo e report. Nele tambem se tem um arquivo **index** de js e css (alem do html que fica na pasta **public**), eles são o coração do nosso projeto.

****Existem **boas maneiras** em nossa criação de um app React, a primeira é a criação de uma pasta **components** dentro de **src**, cada componente terá uma pasta, e **tudo que não for Index, ficará dentro de components**. Faz parte das boas maneiras o uso do setNomeDaVariavel ao fazer uma função que mude um valor de variável.

Aqui fizemos um pequeno script de teste **sem usar hooks**, para mostrar a diferença entre o uso e não uso de hooks, naturalmente depois fizemos utilizando hooks.

**Hooks** todos os states e efeitos que vamos utilizar são hooks, estes são **‘ganchos’, funções que se engancham nos seus componentes, fazendo com que sempre que seu componente sejam ativados a função em hook também vai ser acionada**, existem documentações sobre o uso de hook, são leituras recomendadas, também mencionando sobre o **contextAPI**, que vai ser útil no futuro. Em resumo, são funções que se prendem ao componente e são acionados quando o componente é utilizado.

O **hook de state** utiliza do **estado**, ele vai ter memória, ou seja ele consegue guardar um valor e atualiza-lo de acordo com os eventos.

O **hook de effect** é um “efeito colateral” do uso de uma funcionalidade, ou seja, é um gancho que ao atualizar um outro valor, ele se ativa e muda um outro componente. Os efeitos só vão ser executados **quando o valor proposto sofre uma ação** se por exemplo um valor “sofre” uma ação que não altera seu valor, **o efeito não ocorrerá pois não houve alteração do valor**.