Setup inicial de una App en React JS

Antes de empezar

Verificar si node y npm estan instalados

```
node -v
npm -v
```

Si no está instalado, instalar nvm-windows y luego ejecutar con permisos de administrador:

```
nvm install lts
```

Verificar la instalación:

```
node -v
npm -v
```

Configurando el IDE

Instalar visual studio code. Opcionalmente se pueden instalar algunas extensiones como ESLint, Prettier y Javascript Code Snippets

Creando la aplicación

En el directorio donde se quiera crear la carpeta con el código:

```
npx create-react-app <name>
```

Ejecutar la aplicación

```
cd <name>
npm start
```

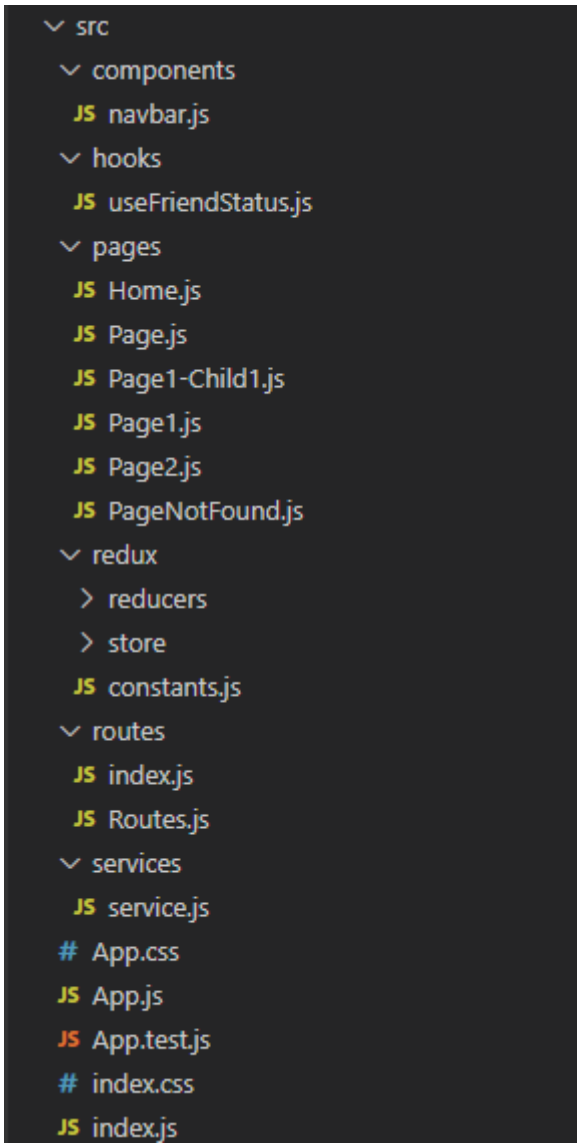
Para instalar nuevas dependencias:

```
npm i --save <package-name>
```

Por ejemplo, algunas de las dependencias que vamos a instalar:

```
npm i --save react-router-dom localforage match-sorter sort-by
npm i --save react-redux redux redux-thunk
npm i --save react-jss
```

Arquitectura de la aplicación



Separar los módulos del sistema en páginas. Si estas páginas requieren componentes y hooks propios, agruparlos dentro de una carpeta para la página.

Si se trata de componentes genéricos o usados en distintas partes de la aplicación, ubicarlos en la carpeta **components**. De forma similar, si se trata de hooks genéricos o usados en distintas partes de la aplicación, ubicarlos en la carpeta **hooks**.

Como regla general, si un solo archivo resuelve un único problema, no es necesario agruparlo dentro de una carpeta, pero si para resolver un módulo se necesita más de un archivo, se deben agrupar todos los archivos relacionados dentro de una carpeta. Ejemplo:

```
- components
   - Button.js
   - ListModal
      - ListModal.js
      - ListModalItem.js
- hooks
   - useSearch.js
...
...
- pages
   - Messages
      - components
         - MessageList
            - MessageList.js
            - MessageListItem.js
         - MessageBubble.js
         - MessageInput.js
      - hooks
         - useMessage
            - useMessage.js
            - useMessage.test.jsx
            - index.js
         - useReadStatus.js
      - Messages.js
   - PageNotFound.js
...
...
```

Buenas Prácticas

- Mantener los componentes lo más pequeños posible, y limitarlos a resolver una funcionalidad
- Mantener el CSS en el código .js, y limitado al scope del componente
- Comentar el código, pero sólo lo necesario

- Usar letras mayúsculas para los nombres de los componentes
- Todos los archivos relacionados a un componente deben estar en la misma carpeta
- Reescribir código e identificar componentes reutilizables a medida que crece el código

Más información

React Docs

Redux Use Guide

React Router Main Concepts

Otras Librerías útiles

Axios

MUI

React Hook Forms

Formik

React Virtualized