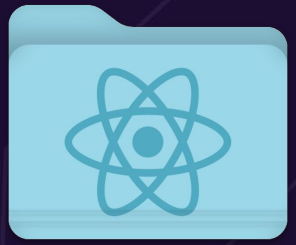
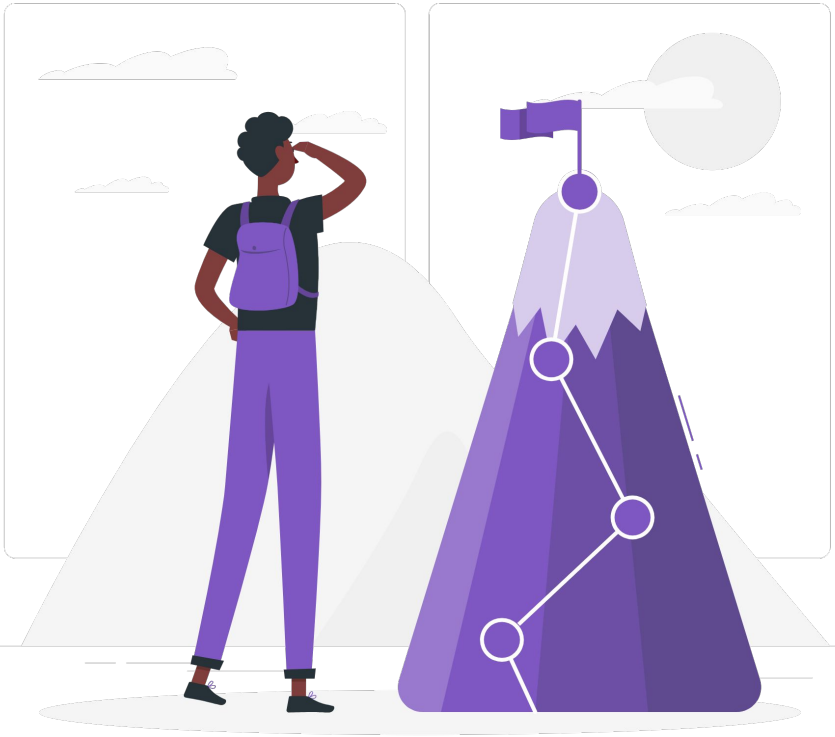


Estructura de Proyecto de React

DEV.F:
DESARROLLAMOS(PERSONAS);



Objetivos

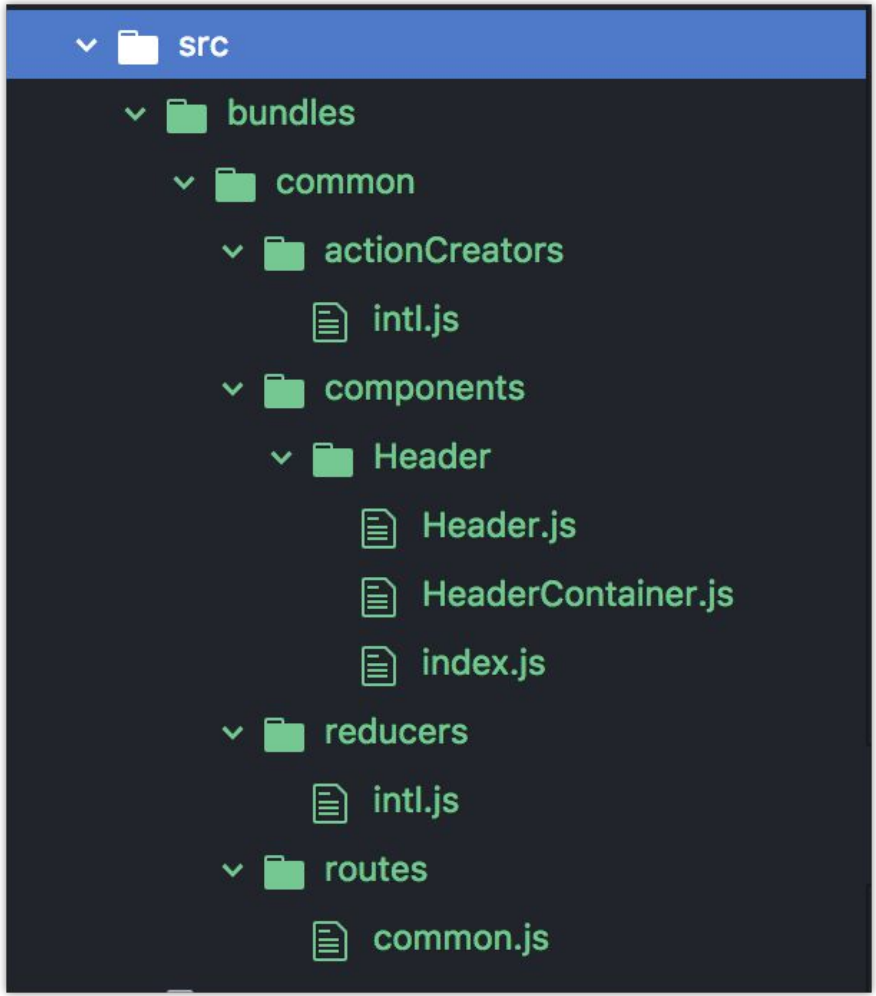
- Aprender las mejores prácticas para estructurar proyectos de React.

DEV.F::

DEV.F:

DISCUSIÓN

*¿Por qué debe preocuparnos **como organizar las carpetas/archivos** en nuestro proyecto de React?*



Problemática

Debido a la propia naturaleza de ReactJS **no existe una estructura de carpetas definida**, quedando a voluntad del desarrollador escoger la que más se adecua al proyecto que va a desarrollar.

Sin embargo al no haber una estructura estándar a menudo surgen distintas formas de plantear esta organización, surgiendo a menudo la pregunta de si lo estaremos haciendo “bien” o por qué hacerlo en primer lugar.

DEV.F::

DEV.F

A tener qué cuenta qué ***no existe una metodología perfecta para estructurar un proyecto de React****, ya que depende de cada uno y de la naturaleza del proyecto la estructura a realizar.*

Sin embargo ***existen convenciones que podríamos usar como base****.*

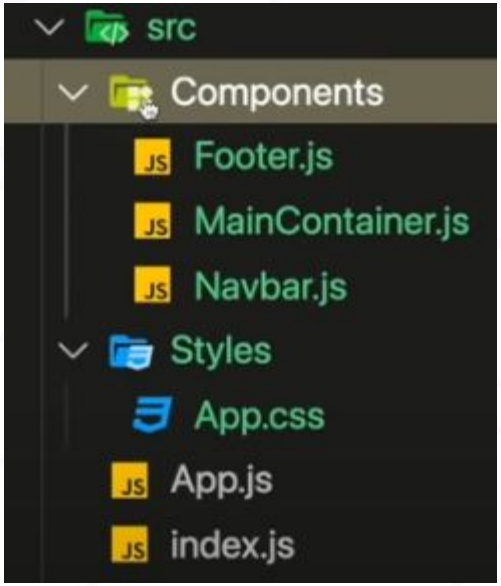
CONVENCIONES ÚTILES

De acuerdo al nivel de experiencia y tipo de proyecto

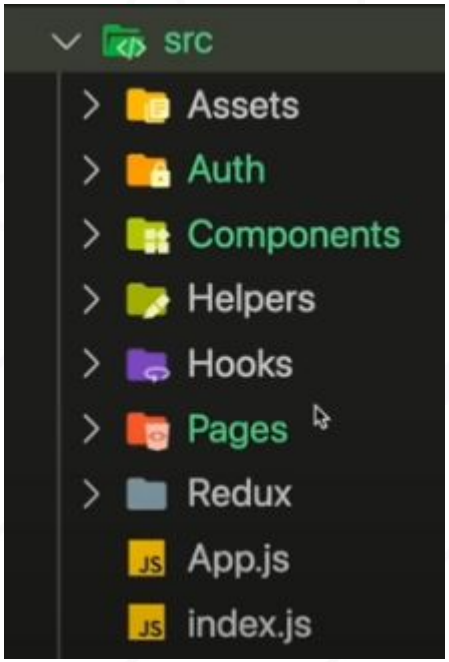
Conforme avanzamos en nuestro aprendizaje y creamos cada vez proyectos más complejos, la correcta organización del mismo nos facilitará mucho la tarea de seguir modificando el proyecto.

A continuación se presentan algunas convenciones de acuerdo al nivel de experiencia y tipo de proyecto.

Formas de Organizar un Proyecto



En proyectos pequeños o cuando comenzamos a aprender, mantenemos una estructura múy básica



En proyectos más complejos o cuando aprendemos más, comenzamos a estructurar mejor nuestro proyecto.

Carpeta: Components

Los componentes son los bloques de construcción de cualquier proyecto react. Esta carpeta consiste en una colección de componentes de interfaz de usuario como botones, modales, entradas, cargador, etc., que pueden ser utilizados a través de varios archivos en el proyecto.

En aplicaciones más complejas, cada componente suele tener una carpeta, e incluso puede tener subcarpetas si hace uso de subcomponentes específicos para dicho componente.

Carpeta: Styles

Existen diversas formas de colocar estilos un proyecto de React, desde incluirlas directamente en cada componente a través de soluciones ***CSS-in-JS*** como ***Styled-Components***, pero en ocasiones queremos utilizar un archivo de estilos globales con CSS.

Es en esta carpeta donde colocaremos los archivos de estilos globales de la aplicación, ya sea en formato CSS, SASS ó SCSS.

Carpeta: Assets

Esta carpeta contiene **todos los activos multimedia**, como imágenes, vídeos, archivos json, etc.

Carpeta: Layouts

Como su nombre lo indica, contiene diseños disponibles y que pueden ser reutilizados para todo el proyecto o partes del mismo: como el encabezado, el pie de página, barras laterales, etc.

Carpeta: Hooks

Esta carpeta contendrá todos los custom Hooks que usemos en el proyecto.

Recordemos que un custom hook es una función de JavaScript que hace uso de hooks nativos de react, pero no renderiza vistas, solo procesa información.

Un custom Hook debe poder ser reutilizable, por lo que podríamos llevarlos sin problemas a otros proyectos de React.

Carpeta: Pages

Los archivos de la carpeta pages indican las vistas de la aplicación de react, por lo que **cada archivo de esta carpeta deberá contener su ruta**.

Una página puede contener subcarpetas de ser necesario.

Cada página tiene su estado y suele utilizarse para llamar a una operación asíncrona (ejemplo: llamar a una API).

Suele estar formada por varios componentes agrupados.

Carpeta: Routes

Esta carpeta se colocan todas las rutas de la aplicación.

Consta de rutas de todo tipo:

- Públicas
- Privadas/Protegidas

Incluso podemos crear sub-rutas.

Por ejemplo, aquí es donde debemos implementar el ruteo con react router dom.

Carpeta: Services

Suele usarse para contener archivos de JavaScript con lógica de negocio relacionada a métodos de llamadas a la API externas.

De esta forma evitamos colocar esa lógica de negocio directamente en un componente y este siempre puede referenciar al servicio que necesite.

Carpeta: Utils

La carpeta ***Utils*** debe contener **funciones de JavaScript** que son usadas a largo de todo el proyecto para completar tareas abstractas de **forma genérica**.

Por ejemplo, funciones para dar formato a fechas, generar números aleatorios, dar formato a datos, regex, etc.

Estas funciones al ser tan genéricas, **podrían reutilizarse en otros proyectos.**

Carpeta: Helpers

Un **helper** es algo que nos ayuda a cumplir un propósito para un proyecto en específico, cuya lógica podemos reutilizar a lo largo del proyecto, pero qué no va ligado a un componente en específico.

Muchos desarrolladores tratan a la carpeta Helpers como sinónimo de **Utils**. Sin embargo existe una pequeña diferencia entre ambos:

Helpers son cosas específicas para el proyecto actual, mientras que **Utils** son funciones más genéricas que podrían reutilizarse en diversos proyectos.

Carpeta: Contexts

La carpeta de ***Context*** es una carpeta que sólo contiene el estado que tiene que ser compartido a través de estos componentes.

Cada página puede tener varios contextos anidados, y cada contexto sólo pasa los datos hacia abajo. Pero para evitar la complejidad, es mejor tener sólo un archivo de contexto.

Carpeta: Config

Contiene los archivos de configuración especiales que hagan uso de variables de entorno (env)