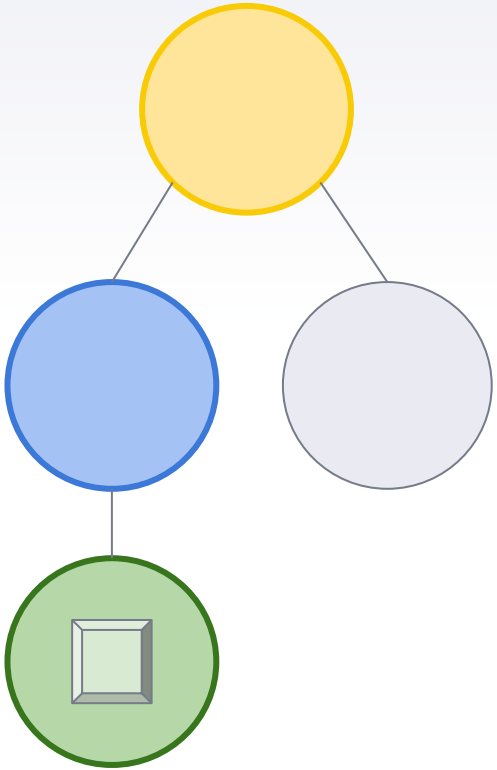
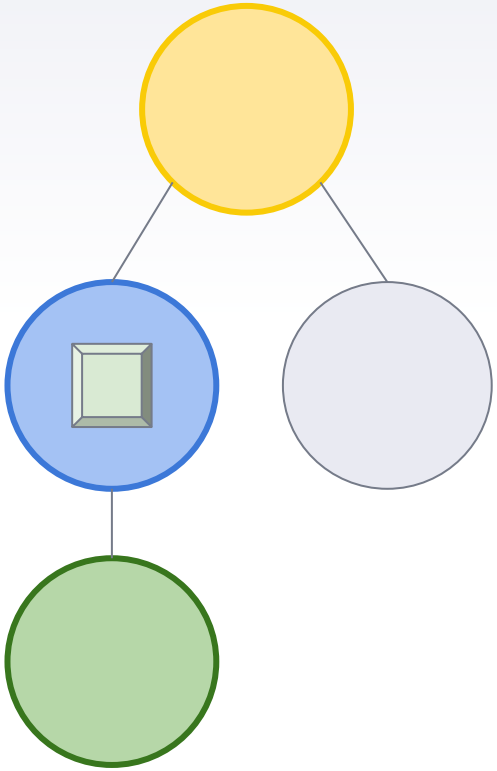
Estructura de la aplicación



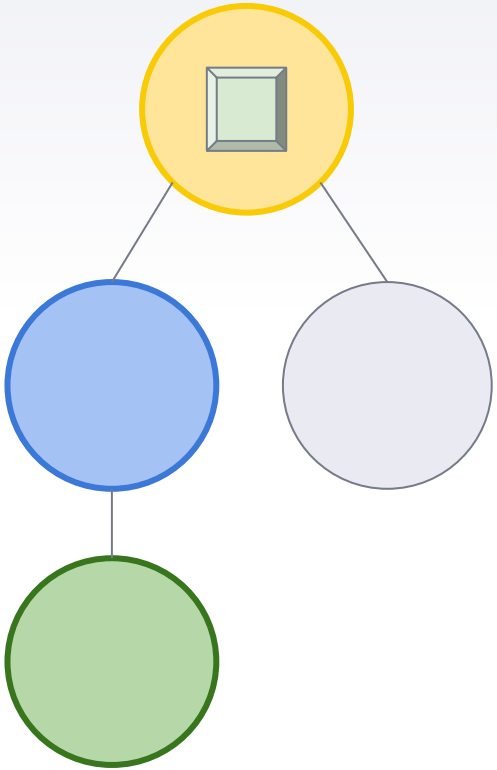
Estructura de la aplicación

Lugar donde se encuentra el estado

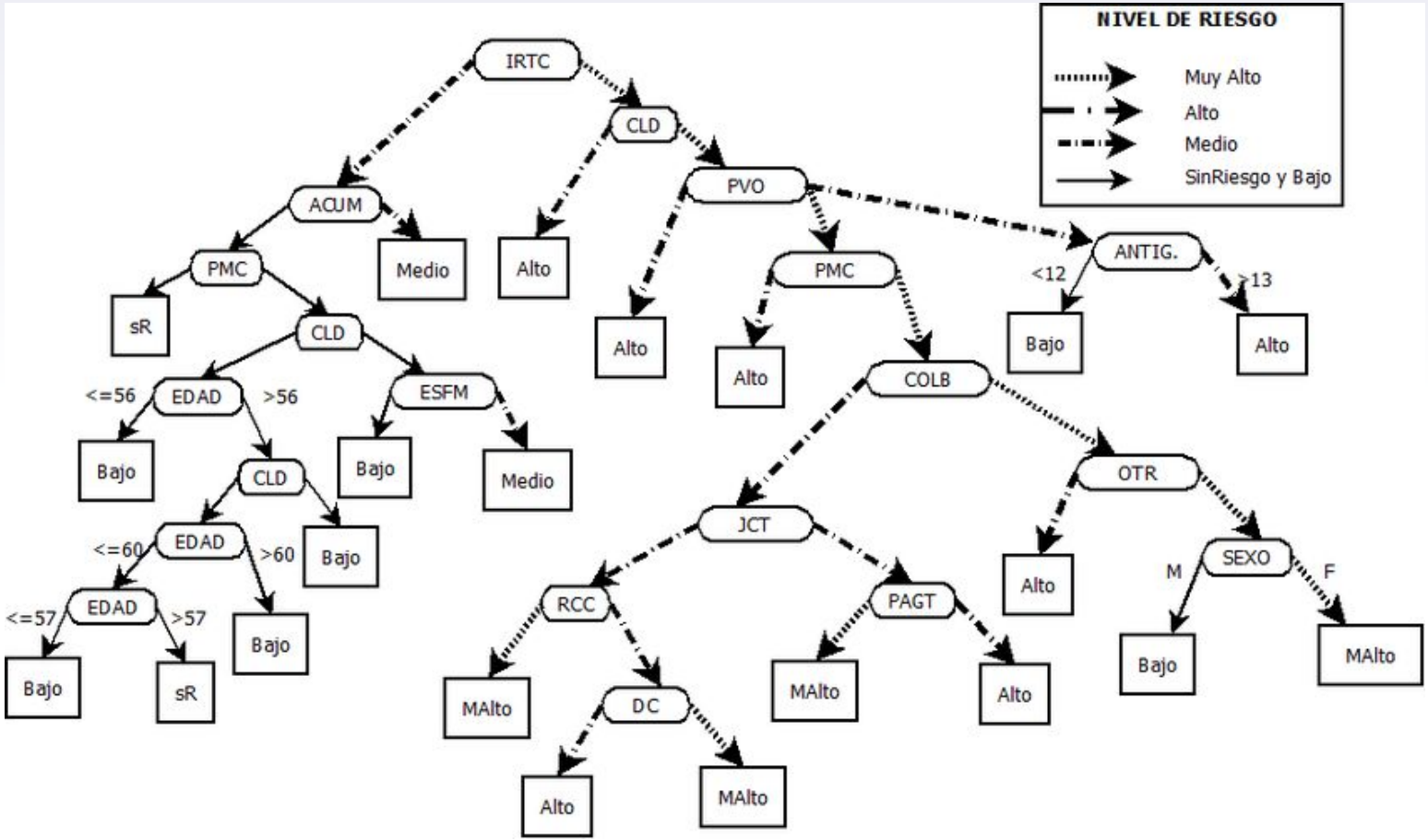


Estructura de la aplicación

Lugar donde se encuentra el estado

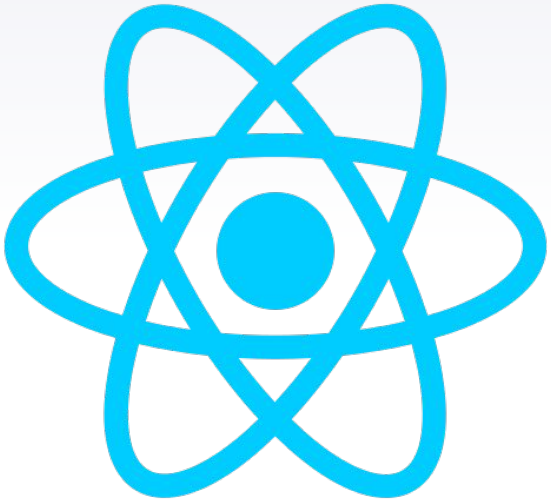


Estructura de la aplicación

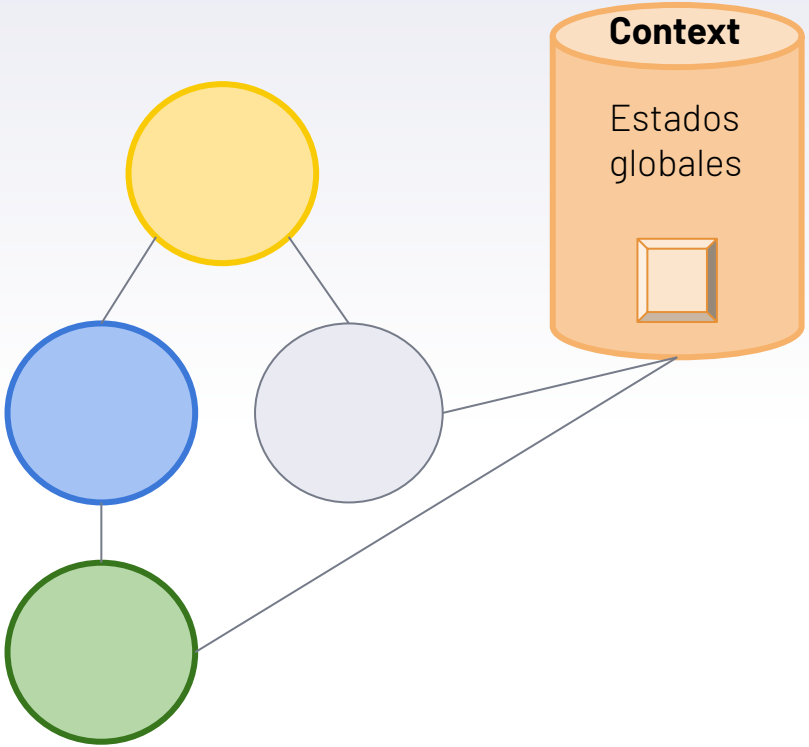


Estados globales

useContext



Estructura de la aplicación



Cómo se conforma el Context

```
1   import { createContext, useContext, useState } from "react";
```

Importamos createContext y useContext

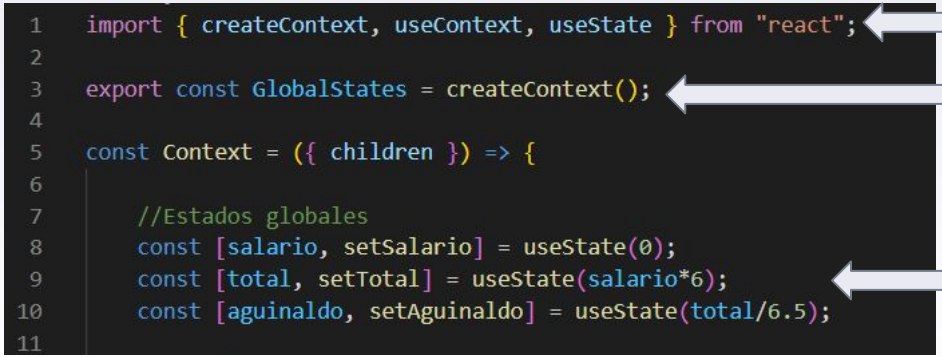
Cómo se conforma el Context

```
1  import { createContext, useContext, useState } from "react";
2
3  export const GlobalStates = createContext();
4
```

Importamos createContext y useContext

Creamos el contexto y lo guardamos en una constante

Cómo se conforma el Context



```
1  import { createContext, useContext, useState } from "react";
2
3  export const GlobalStates = createContext();
4
5  const Context = ({ children }) => {
6
7      //Estados globales
8      const [salario, setSalario] = useState(0);
9      const [total, setTotal] = useState(salario*6);
10     const [aguinaldo, setAguinaldo] = useState(total/6.5);
11
```

Importamos createContext y useContext

Creamos el contexto y lo guardamos en una constante

Declaramos los estados que vayamos a necesitar

Cómo se conforma el Context

```
1  import { createContext, useContext, useState } from "react";
2
3  export const GlobalStates = createContext();
4
5  const Context = ({ children }) => {
6
7      //Estados globales
8      const [salario, setSalario] = useState(0);
9      const [total, setTotal] = useState(salario*6);
10     const [aguinaldo, setAguinaldo] = useState(total/6.5);
11
12     return (
13         <GlobalStates.Provider value={{
14             salario, setSalario,
15             total, setTotal,
16             aguinaldo, setAguinaldo
17         }}>
18             {children}
19         </GlobalStates.Provider>
20     )
21 }
```

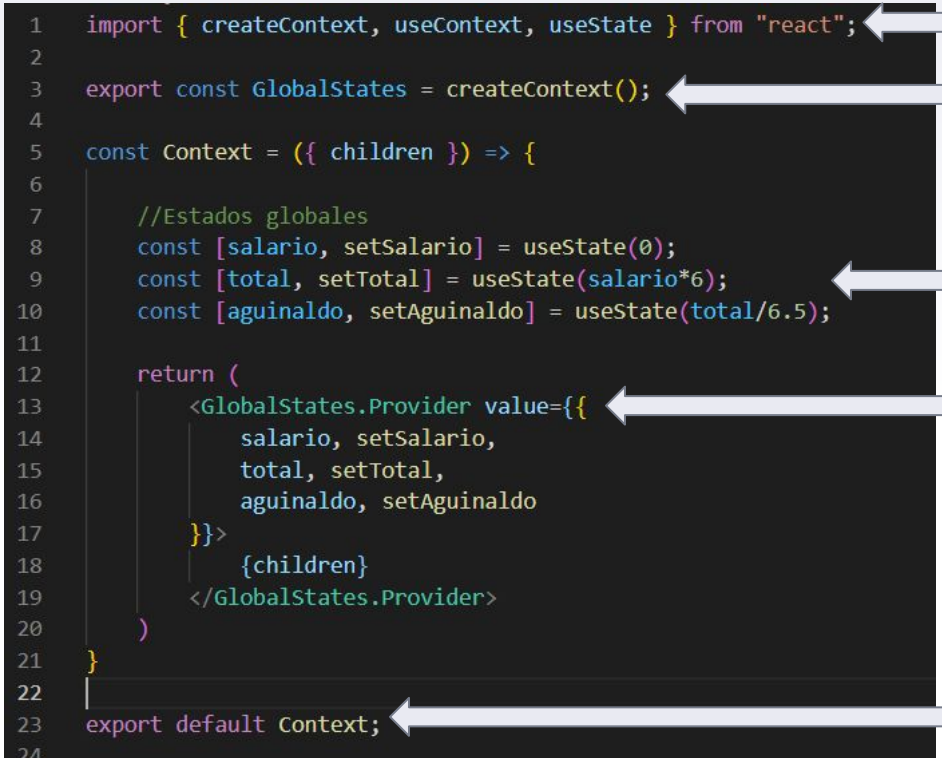
Importamos createContext y useContext

Creamos el contexto y lo guardamos en una constante

Declaramos los estados que vayamos a necesitar

Creamos el layout que va a contener toda la aplicación

Cómo se conforma el Context



```
1  import { createContext, useContext, useState } from "react";
2
3  export const GlobalStates = createContext();
4
5  const Context = ({ children }) => {
6
7      //Estados globales
8      const [salario, setSalario] = useState(0);
9      const [total, setTotal] = useState(salario*6);
10     const [aguinaldo, setAguinaldo] = useState(total/6.5);
11
12     return (
13         <GlobalStates.Provider value={{
14             salario, setSalario,
15             total, setTotal,
16             aguinaldo, setAguinaldo
17         }}>
18             {children}
19         </GlobalStates.Provider>
20     )
21 }
22
23 export default Context;
```

Importamos createContext y useContext

Creamos el contexto y lo guardamos en una constante

Declaramos los estados que vayamos a necesitar

Creamos el layout que va a contener toda la aplicación

Lo exportamos

Cómo se conforma el Context

```
1  import { createContext, useContext, useState } from "react";
2
3  export const GlobalStates = createContext();
4
5  const Context = ({ children }) => {
6
7      //Estados globales
8      const [salario, setSalario] = useState(0);
9      const [total, setTotal] = useState(salario*6);
10     const [aguinaldo, setAguinaldo] = useState(total/6.5);
11
12     return (
13         <GlobalStates.Provider value={{
14             salario, setSalario,
15             total, setTotal,
16             aguinaldo, setAguinaldo
17         }}>
18             {children}
19         </GlobalStates.Provider>
20     )
21 }
22
23 export default Context;
24
25 export const useGlobalStates = () => {
26     return useContext(GlobalStates);
27 }
28
```

Importamos createContext y useContext

Creamos el contexto y lo guardamos en una constante

Declaramos los estados que vayamos a necesitar

Creamos el layout que va a contener toda la aplicación

Lo exportamos

Función que servirá para ahorrarnos código

De qué manera conectamos el Context a nuestra aplicación

```
6  import Context from './Context';
7
8  const root = ReactDOM.createRoot(document.getElementById('root'));
9  root.render(
10   <React.StrictMode>
11     <Context>
12       <App />
13     </Context>
14   </React.StrictMode>
15 );
```

useContext

Una vez que el Context está integrado a la aplicación, podemos llamar los estados globales a cualquier componente que se encuentre dentro del árbol, de la siguiente forma:

```
import { useGlobalStates } from '../../Context'

const MiComponente = () => {
const { estadoGlobal } = useGlobalStates();
```

Luego, una vez que se llame a “estadoGlobal” lo podemos usar como si fuese un estado propio del componente.

useContext

Una vez que el Context está integrado a la aplicación, podemos llamar los estados globales a cualquier componente que se encuentre dentro del árbol, de la siguiente forma:

```
import { useContext } from 'react'
import { GlobalStates } from '../../Context'
const MiComponente = () => {
const { estadoGlobal } = useContext(GlobalStates);
```

Luego, una vez que se llame a “estadoGlobal” lo podemos usar como si fuese un estado propio del componente.

En conclusión

- useContext es un Hook de la librería de React que nos permite manipular los estados de la aplicación en cualquier componente.
- Es importante, tener en cuenta cómo se conforman las partes, como se integra a la App y finalmente cómo debemos llamar los estados dentro de los componentes que vamos a utilizar.
- Esto nos va a servir para poder reutilizar los datos entre componentes de manera más dinámica, sin necesidad de estar pasando las propiedades entre componentes padres y componentes hijos.