Crear Context

- Guardo el createContext en ClimaContext
- Creo la función ClimaProvider con el children
- En el return coloco el .Provider con el children y el value obligatorio con doble llave al ser un objeto dentro de js
- Exporto ambas, el Context por defecto

```
import {useState, createContext} from 'react'

const ClimaContext = createContext();

const ClimaProvider = ({children}) =>{
    return(
        <ClimaContext.Provider
        value={{

            }}>
            {children}
        </ClimaContext.Provider>
    )
}

export {ClimaProvider}

export default ClimaContext
```

- En App jsx importo ClimaProvider entre llaves y rodeo la App con ClimaProvider
- Coloco un header con un h1

```
function App() {


  return (
    <ClimaProvider>
      <header>
        <h1>Buscador de Clima</h1>
      </header>
      <AppClima />

    </ClimaProvider>
  )
}
export default App
```

- Creo el hook para no tener que importar y declarar el context

```
import {useContext} from 'react'
import ClimaContext from '../context/ClimaProvider'

const useClima = () =>{
    return useContext(ClimaContext)
}

export default useClima
```

- Los resultados de la búsqueda los voy a requerir en más de un componente por lo que declaro el state en el provider con un objeto y dentro ciudad y país con un string vacío
- Creo la función del onChange y lo paso por el value del provider. Lo mismo hago con el state
- Me traigo el state anterior con el spread para que no lo sobreescriba ni lo borre
- Computo el valor de la izquierda entre corchetes para poder escribirlo con el de la derecha que es el value que estoy introduciendo.

```
import {useState, createContext} from 'react'

const ClimaContext = createContext();

const ClimaProvider = ({children}) =>{

    const [busqueda, setBusqueda] = useState({
        ciudad: '',
        pais: ''
    })
  const datosBusqueda =(e)=>{
    setBusqueda({
        ...busqueda,
        [e.target.name]: e.target.value
    })
  }
    return(

    <ClimaContext.Provider
    value={{
        busqueda,
        datosBusqueda
    }}>
         {children}

    </ClimaContext.Provider>

   )
}

export {
    ClimaProvider
```

```
}
export default ClimaContext
```

- Extraigo los valores por desestructuración en el formulario

```
 const {busqueda, datosBusqueda} = useClima()
  const {ciudad, pais}= busqueda
```

- Añado {datosBusqueda} a los onChange y asocio los value a {ciudad} y {pais}
- Ahora que ya tengo el state colocado hay que hacer el onSubmit
- Para hacer la validación crearé un state local alerta y setAlerta, incia vacío
- Muestro el mensaje antes del form

```
 const handleSubmit = (e)=>{
    e.preventDefault()
    if(Object.values(busqueda).includes('')){
      setAlerta('Todos los campos son obligatorios')
      return
    }
  }
  return (
    <div className="contenedor">
      {alerta && <p>{alerta}</p>}
      <form onSubmit={handleSubmit}>
```

Declaro la función consultarClima en Provider, debajo de datosBusqueda Le pongo un console.log(datos) y la paso por el value del Provider

- La extraigo del hook y lo coloco fuera del return en el handleSubmit

```
  const handleSubmit = (e)=>{
    e.preventDefault()
    if(Object.values(busqueda).includes('')){
      setAlerta('Todos los campos son obligatorios')
      return
    }
    consultarClima(busqueda)
  }
```

Ahora puedo ver en consola las dos opciones seleccionadas y como vienen desde el provider