Inicio

- Creo las carpetas hooks, context y components.
- Creo AppClima.jsxen la carpeta components y la renderizo en App.jsx
- Lo hago para poder acceder al state desde este componente a todos los demás
- Creo un Formulario y lo incorporo a AppCLima. Lo importo y lo renderizo en un main

---

- En el formulario añado un contenedor con un form y un div con la clase campo
- Pongo un label, un select...queda así

```
import React from 'react'

const Formulario = () => {
  return (
    <div className="contenedor">
      <form>
        <div className="campo">
          <label htmlFor='ciudad'>Ciudad</label>
          <input
          type="text"
          id="ciudad"
          name="ciudad"/>
        </div>
        <div className="campo">
          <label htmlFor='Pais'>País</label>
          <select id="pais"
                  name="pais">
              <option value="">Seleccione el país</option>
              <option value="US">Estados Unidos</option>
              <option value="MX">México</option>
              <option value="ES">España</option>
              <option value="PE">Perú</option>
              <option value="CO">Colombia</option>
              <option value="AR">Argentina</option>
          </select>
        </div>
        <input type="submit"
                value="BUSCAR"/>

      </form>
    </div>
  )
}

export default Formulario
```

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

APIKEY como varianle de entrono

- En el directorio raíz, añado un archivo llamado .env
- Como uso vite, la sintaxis es

```
VITE_API_KEY= cf0a5ab8d26d66e27d91dc742ee8eae2
```

- Para imprimirlo en consola

```
console.log(import.meta.env.VITE_API_KEY)
```

Consultando el Clima en la API

- Instalo AXIOS y lo importo en el provider
- Hago un try y un catch y desestructuro ciudad y pais para hacerlos por separado
- Guardo el apiKey en una variable llamada appId
- Guardo en una url con un template string la consulta a la API e inyecto ciudad y pais
- http://api.openweathermap.org/geo/1.0/direct?q={city name},{state code},{country code}&limit={limit}&appid={API key}

Queda asi, lo pruebo con un console.log!

```
const consultarClima= async (datos) =>{
      try {
          const {ciudad, pais} = datos
          const appId= import.meta.env.VITE_API_KEY
          const url = `http://api.openweathermap.org/geo/1.0/direct?
q=${ciudad},${pais}&limit=1&appid=${appId}`

          console.log(url)

      } catch (error) {
          console.log(error)
      }
  }
```

- Si copio y pego la url en consola en la barra de navegación, sale el objeto JSON Lo que interesa es extraer latitud y longitud

```
try {
          const {ciudad, pais} = datos
          const appId= import.meta.env.VITE_API_KEY
          const url = `http://api.openweathermap.org/geo/1.0/direct?
q=${ciudad},${pais}&limit=1&appid=${appId}`
          const {data} =  await axios(url)

          console.log(data)
      }
```

- Devuelve un arreglo, yo le puedo especificar la posición 0 entre corchetes, donde esta lat y long
- Extraigo la latitud y la longitud

```
const {lat, long} = data[0]
```

- Ahora falta la url para consultar el clima
- La url es esta
- https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}
- Necesito extraer data pero ya la tengo declarada, renombro

```
 const {lat, long} = data
          const urlClima= `https://api.openweathermap.org/data/2.5/weather?
lat=${lat}&lon=${lon}&appid=${appId}`
          const {data: clima} = await axios(urlClima)
```

- Inicio el state con resultado, setResultado y un objeto vacío
- Cambio el console.log de clima por setResultado y paso resultado al provider
- Creo el componente Resultado.jsx y lo importo en AppClima
- Quiero que se muestre si ya hay resultado
- Importo el hook para el context
- Cuando vas a renderizar un objeto e inicia vacío siempre se mostrará.
- Es conveniente añadirle un ? y hacer la evaluación con .name o similar

```
<main className="dos-columnas">
        <Formulario />

      {resultado?.name &&  <Resultado/>}

    </main>
```

```
const Resultado = () => {
    const {resultado} = useClima()
    const {name} = resultado
  return (
    <div className="contenedor">
        <h2>El clima de {name} es: </h2>
    </div>
  )
}
```

El resto es mostrar la info que viene en resultado. En el main vienen las temperaturas