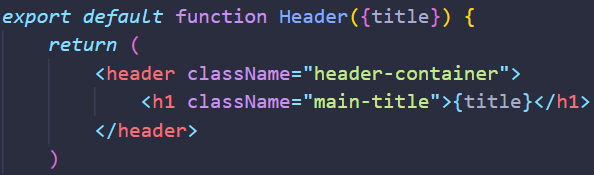
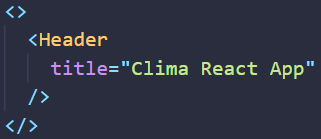
**Proyecto 6 Clima**

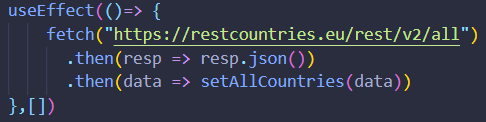
**1**. Eliminamos los archivos de testing, app.css y logo.

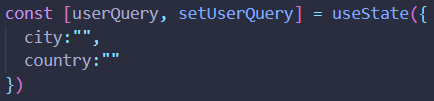
**2**. Dentro de index.html (public) pegamos el código que nos dan desde gitHub.

**3**. En el index.css pegamos el código css de gitHub.

**4**. Creamos el componente **Header.js** y lo renderizamos dentro de App.js.

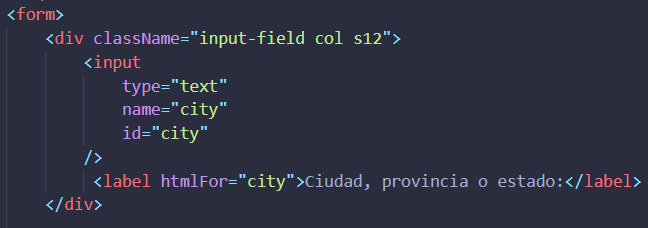
**5**. Dentro de **App.js** vamos a **crear** un **estado** para **almacenar** todos los **países** del mundo, luego dentro de un **useEffect** sin dependencias, hacemos el **llamado** a la **API**, **guardando** la **información** en el **estado**.



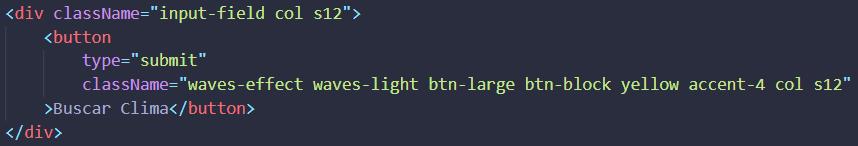
**6**. Creamos **dentro** de **App.js** un **estado** que va a **almacenar** la **ciudad** y el **país** que el **usuario** va a **elegir**.

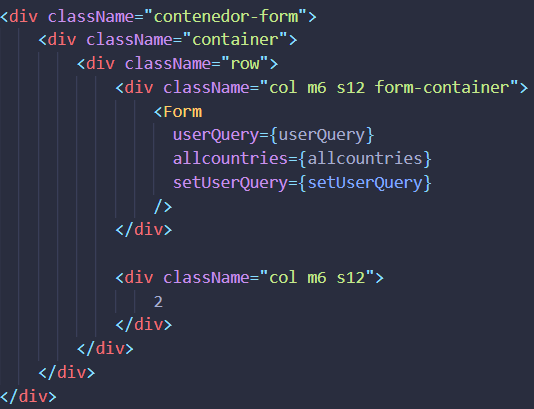
**7**. **Creamos** el componente **Form.js** donde el **usuario** va a **elegir** la **ciudad** y el **país** del cual quiere saber el clima. Este componente va a recibir: el estado que creamos en el paso anterior y la lista de países.

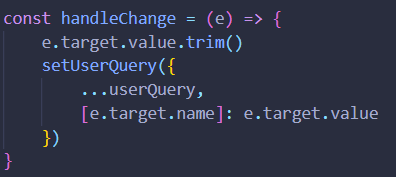
**a)** Primero incluimos un input.

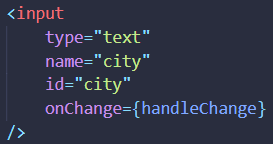
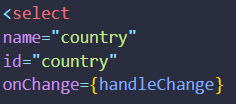


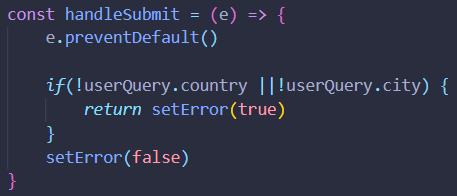
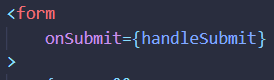
**b)** Luego preguntamos si el array que contiene a todos los países tiene información, y si la tiene que renderize un <select> con una opción inicial y luego toda la lista de países. Lo hacemos de esta forma por las limitaciones que nos da el framework de Css que estamos utilizando.

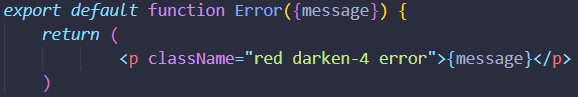
**c)** Incluimos el botón de submit y renderizamos el componente dentro de App.js (notese que creamos distintos <div> para organizar el contenido en dos columnas.



**8**. **Guardando los datos en el estado**: creamos la función handleChange y hacemos lo de siempre, luego se la pasamos tanto al input como al select.



**9**. **Validando la información**: **creamos** un **estado** para los **errores** y lo **iniciamos** en **false**, luego la función **handleSubmit** validando los errores. En este caso no estamos guardando los datos al final de la función porque eso ya se está haciendo en el onChange.

**10**. **Mostrando los errores**: creamos el componente Error.js que va a constar de un simple <p>. Luego dentro de Form.js preguntamos si el estado de error es true y en caso de serlo renderizamos el componente.



**11**. **Consultando la API de clima**: para poder llevar a cabo esto vamos a requerir de varios pasos:

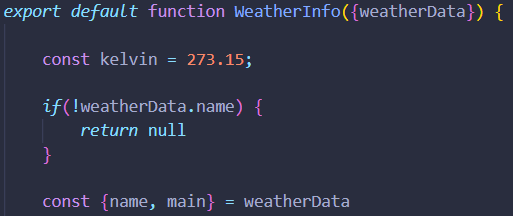
**a)** Saber cuándo la elección del usuario esta validada: creamos un nuevo estado y lo iniciamos en false, se lo pasamos a Form.js y luego de validar los campos lo pasamos a true.

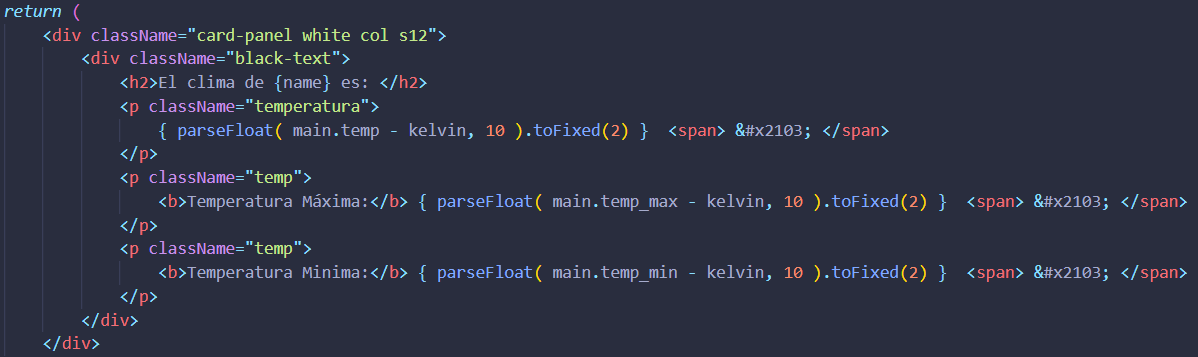


**b)** Creamos un estado para almacenar la información de la API: sabemos que nos va a llegar como un objeto literal

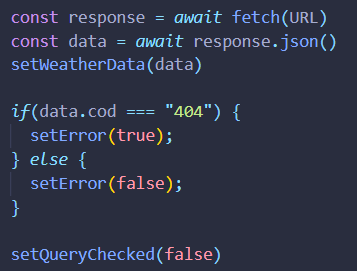
**c)** Hacemos un useEffect: **creamos** una **función** **asincrónica**, **preguntamos** si la **información** del **formulario** esta **chequeada**, en caso de que si continuamos. **Guardamos** la **API key** y la **URL** del sitio **en variables**. Le **pasamos** los **parámetros** a la **URL** con los **string literals**, **hacemos** el **pedido** a la **API** con el **fetch** Y **guardamos la información en el estado**. Finalmente pasamos el chequeo del formulario a false y le agregamos la dependencia al useEffect.



**12**. **Mostrando los resultados del clima**: creamos el componente **WeatherInfo.js** que va a **recibir** la **información** de la **API**. Dentro del mismo preguntamos si nos viene la información y en caso de que no retornamos null evitando que el componente se renderize. Luego extraemos name y main ya que allí estará la información que nos interesa.

Le pasamos la información al cuerpo del componente y usamos el método parseFloat para que nos queden únicamente dos números después de la coma. Hay una resta debido a que la información se pasa en grados Kelvin pero la queremos en grados centígrados.

**13**. **Mensaje de error si no existe la ciudad**: creamos un **estado** de **error** dentro de **App.js** y dentro del useEffect luego de obtener la información del clima preguntamos si viene con un código 404, en caso que si ponemos error en true y de lo contrario en false (importante hacer el if sobre la constante que recibe el pedido (data) y no sobre el estado)



**14**. **Mostrando el mensaje de error**: dentro de App.js hacemos un ternario verificando si error esta en true imprimimos el error, de lo contrario la información del clima.

