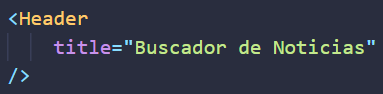
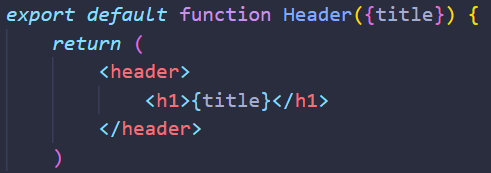
**Proyecto 8**

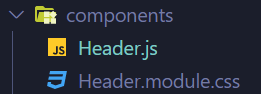
**Resultado Final:**

**1.** **Eliminamos** los **archivos** que **no usamos**, incluidos todos los CSS y **agregamos** en el **<head>** del index el **código** de **Github**.

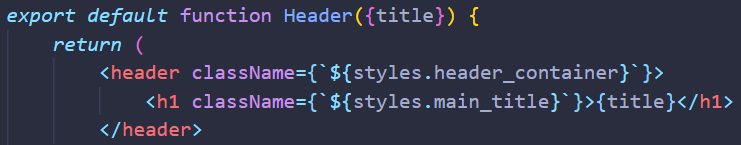
**2**. **Creamos** el componente **Header.js** y lo agregamos en **App.js.**



**3**. **Dando estilos con CSS Module**: para esto dentro de la carpeta components debemos **crear** un **archivo CSS** con el **nombre** del **componente** **seguido** de un “**.module**”. Por **ejemplo** para el Header.js el archivo es **Header.module.css**

**Importante**: es recomendable usar guiones bajos para separar las palabras cuando nombramos las clases.

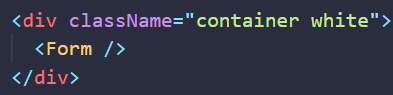
Una vez que creamos la **hoja de estilos**, la **importamos** **en el componente** y le **incluimos** las **clases** con **string literals**. Podemos incluir otras clases también como las de Bootstrap por ejemplo.



**4**. **Creamos** el componente **Form.js** y por el momento incluimos un título y un botón de submit (las clases se las damos con CSS modules). Luego lo imprimimos en **App.js**



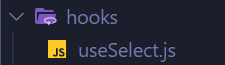


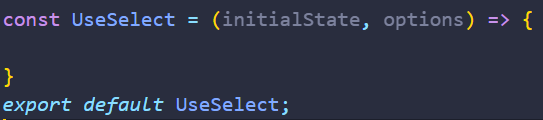


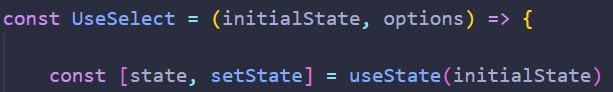
**5. Guardando la información de las categorías**: vamos a guardar en un array las categorías de noticias aceptadas por la API.

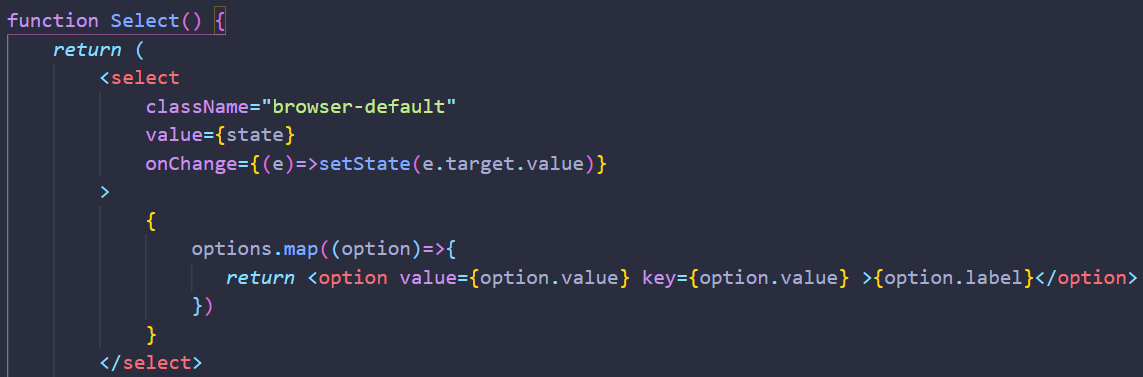
**6.** **Creando Hook para el select**: para crear un hook propio para el <select> debemos llevar a cabo los siguientes pasos:

**a)** Crear carpeta “hooks” y dentro el archivo **useSelect.js**



**b)** Todo **hook** se **inicia** con una **función** que **recibe** **determinados parámetros**. En este caso recibirá un **estado inicial** con el nombre de una **categoría** de noticias **y** un **array** con todas las **categorías disponibles**.

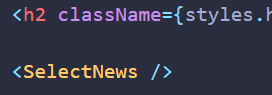
**c)** Hay que requerir useState y **crear** un **estado** que tome como **valor inicial** el primer **parámetro** de la **función**.

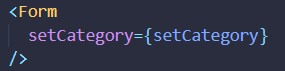
**d)** Ahora debemos **crear** el **código JSX** que va a **renderizar el hook**. Al <**select**> le damos como **value** el **estado inicial** (ya que al principio le vamos a dar la categoría de “general”). **Hacemos** que el **estado** se **actualice** **cuando** se **selecciona otra opción** (**Onchange**). **Finalmente** hacemos un **map** del **segundo parámetro** de la función que **contiene** todas las **categorías en un array**.

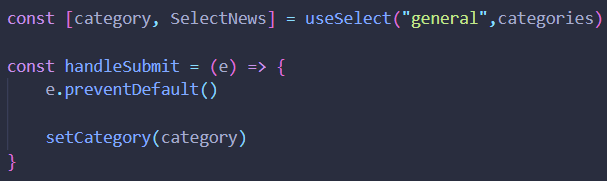
**e)** Por ultimo **retornamos** el **estado** y la función con el código JSX a renderizar.

**7**. **Usando el hook en el formulario**: **requerimos** el **hook** **y** la **data** de las **categorías**. Luego **hacemos** una **constante** que va a **tomar** los **datos retornados del hook** (el state y el código JSX del <select>) **les podemos poner cualquier nombre**. Esa constante la **igualamos** al **hook** (**useSelect**) y le **pasamos** los **parámetros** que **requiere** (la categoría inicial y la lista de categorías)

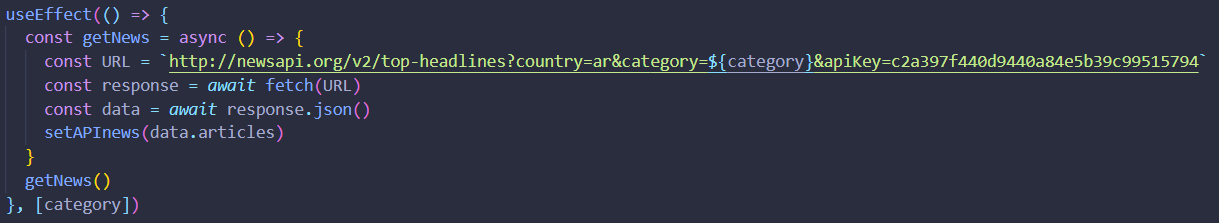


Finalmente **imprimimos** el **componente** que **desestructuramos** del **hook**.

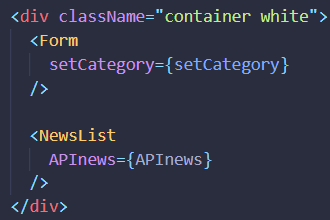
**8**. **Leer categoría que el usuario seleccione**: dentro de **App.js** **creamos** un **nuevo** **estado** que va a **almacenar** la **categoría** para **pedirle** a la **API** y se lo **pasamos** a **Form.js.** Luego en el **submit** del form le **pasamos** el **estado** del **hook** (que se inicia en general y cambia si el usuario selecciona otra opción)



**9**. **Pidiéndole las noticias a la API**: dentro de **App.js** creamos un **nuevo** **estado** para **guardar** las **noticias** que nos van a llegar. Hacemos un **useEffect** (con el estado de categorías como dependencia) y dentro del mismo una **función** **asincrónica** que va a **hacerle** el **pedido** a la **API** y luego **guardar** la **información** en el **estado**.



Si bien es cierto que cuando **carga** la **página** **no tenemos una categoría seleccionada** en el **estado** de **App.js**, si a la **API** **no se le pasa una categoría** **devuelve las noticias generales**. Con este useEffect cuando cargue la página se hará el pedido a las noticias generales de argentina.

**10**. **Mostrando las noticias**: creamos el componente **NewsList.js** que va a recibir como **props** el **array con las noticias de la API**, **hacemos** un **map** para que en cada iteración se **imprima** el **componente** **News.js** (que va a renderizar noticia por noticia)



Creamos el componente **News.js** que va a contener la estructura de la noticia. Allí **desestructuramos** las **propiedades** del **objeto** de **cada noticia**. Finalmente **mediante** una **clase** de **tarjeta** de **materialice** **incluimos** la **imagen**, la **fuente**, el **título**, la **descripción** y el **link** a la noticia.

