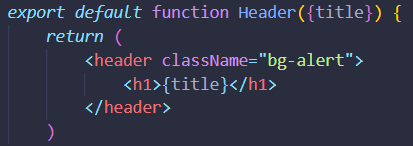
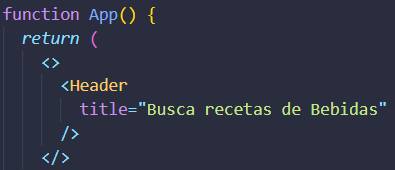
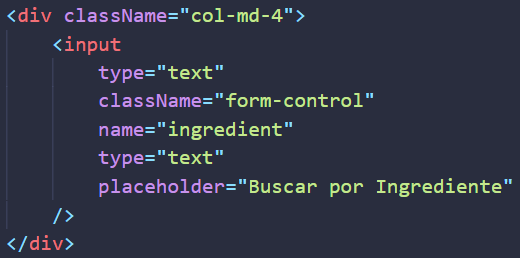
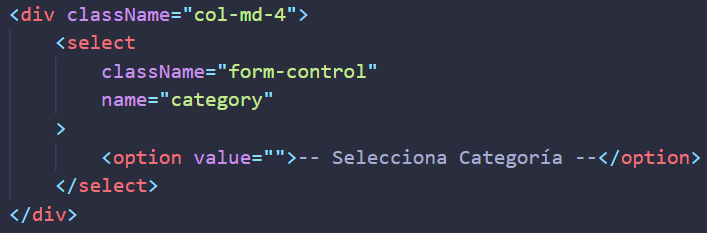
**Proyecto 11**

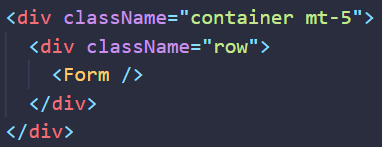
**Resultado final:**

**1**. Agregamos el **código** de **GitHub** en el <**head**>, en el **index.css** e **importamos** **Bootstrap**.

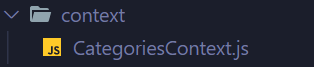
**2**. Instalamos axios y MATERIAL-UI: **npm i axios** **@material-ui/core**

**3**. **Creamos** el componente **Header.js** y lo imprimimos en App.js.

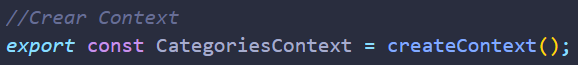
**4**. **Creamos** el componente **Form.js** que va a contar con un <input>, un <select> (las opciones se van a llenar con la API) y un <button>. Luego lo **importamos** dentro de **App.js.**

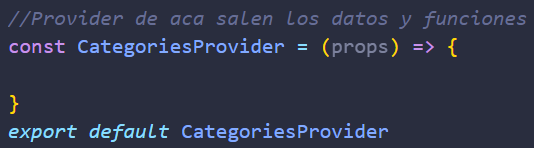


**5**. **Creando** **useContext Hook**: useContent nos **permite** **compartir información** **entre componentes** **sin** la necesidad de **estar** **pasándosela** por **props** de **unos a otros**. Vamos a crear uno de prueba:

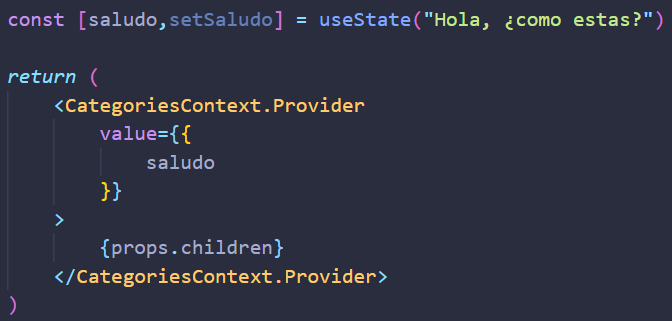
**a)** Dentro de src **creamos** la **carpeta** “**context**” y el archivo (**CategoriesContext.js**)

**b)** Allí debemos requerir **createContext** y crear el Context utilizando el método importado (lo exportamos de forma aislada)

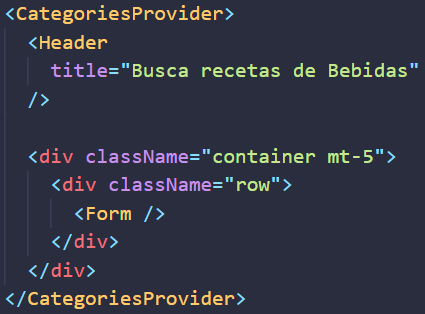


**c)** Luego **creamos** y **exportamos** el **Provider** que va a ser una **función** que recibe como **parámetro** **props**.

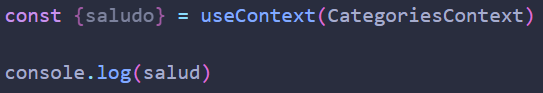
**d)** Dentro de la misma **creamos** el **estado** que **queremos compartir** y hacemos un **return** donde se van a **exportar los datos**. **Allí** **usamos** **como componente** el **contexto** que **creamos** en el **primer paso seguido de .Provider**, le pasamos como **propiedad** **value** que contendrá un **objeto** con los **datos** a **compartir**. Luego **como contenido** debemos poner **props.childen**.

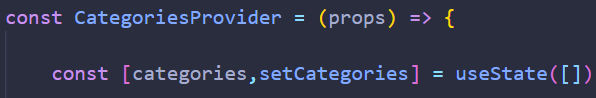


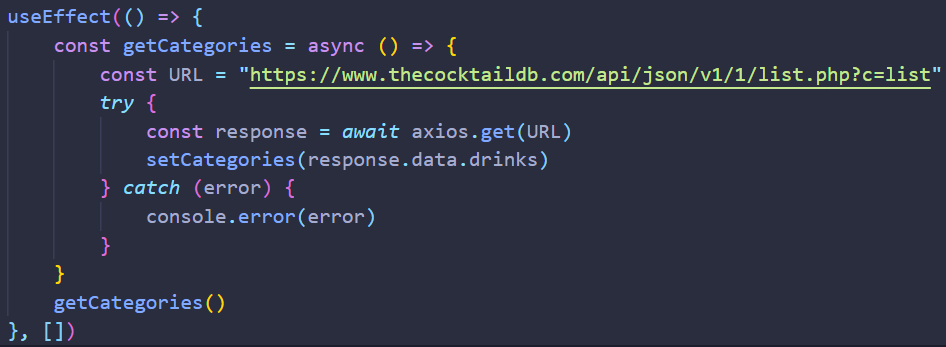
**6**. **Usando el useContext**: dentro de **App.js** debemos **requerir** el **Provider** y **utilizarlo** como **contenedor de los componentes** en los cuales queremos que esté disponible la información.

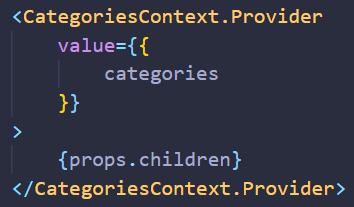


Vamos al **componente** **en el cual** queremos **usar** la **información** e **importamos** el hook **useContext** y el **contexto creado**. Luego **desestructuramos** la **información** que nos interesa y la **igualamos** a **useContext** pasándole como **parámetro** el **contexto creado**.

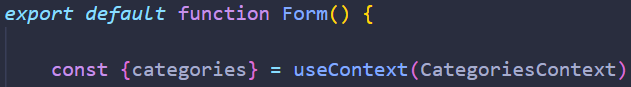


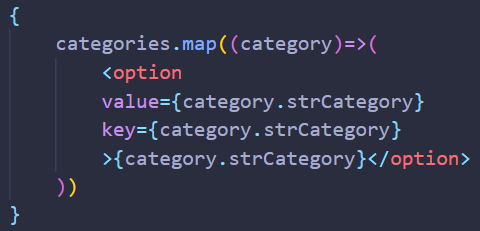
**7**. **Usando el useContext para obtener las categorías**: dentro de **Provider** creamos un **nuevo** **estado** para **almacenar** las **categorías** que nos llegan de la **API**. Luego hacemos un **useEffect**, **requerimos** la **información**, la **guardamos** en el **estado** y finalmente **exportamos** el **estado** en la propiedad **value**.

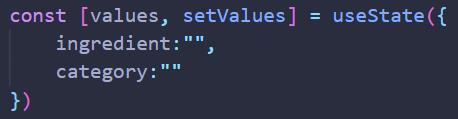


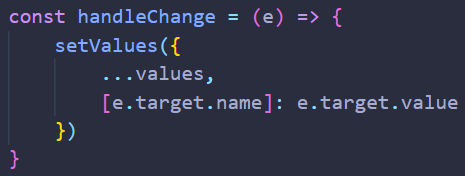


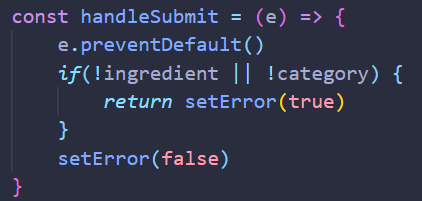
**8**. **Usando las categorías en el Form.js**: **importamos** **useContext** y el **contexto creado**. Luego **desestructuramos** las **categorías** y utilizamos **useContext** pasando como parámetro el contexto creado. Finalmente hacemos un **map** dentro de <**select**> imprimiendo las categorías.



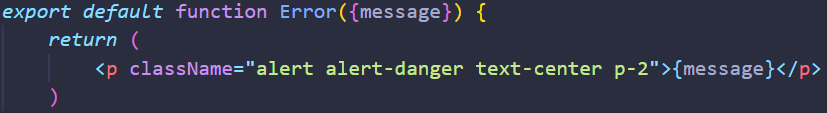


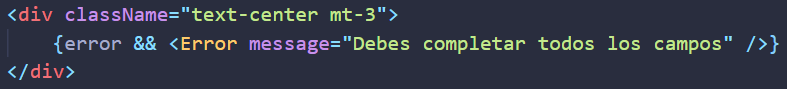
**9**. **Guardando los datos del formulario**: creamos un **estado** para **guardar** los **values** de los **inputs** y luego creamos una función que se ejecutara en el evento **Onchange** del <**input>** y el <**select**>.



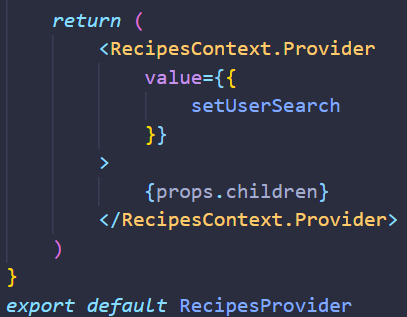
**10**. **Validando los datos y mostrando errores:** creamos un nuevo **estado** para los **errores** y bajo el evento **onSubmit** del <**form**> ejecutamos la función **handleSubmit**.

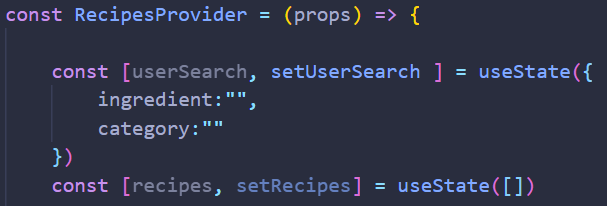
**Creamos** **Error.js** y lo **renderizamos** en caso de que su **valor** sea **true**.



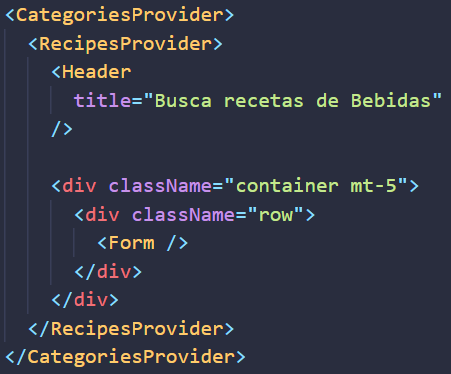


**11**. **Creando useContext para las recetas**: dentro de context **creamos** un nuevo archivo (**DrinkRecipesContext.js**) e **importamos** **createContext, useState y useEffect**. Creamos el **context** y el **Provider**. Dentro del mismo hacemos **2 estados**, **uno** para **guardar** los **datos** **validados** del **formulario** y el **otro** para **recibir** las **recetas** desde la **API**. Luego en el return usamos el contexto creado y **exportamos** el **value que guarda datos en el estado**.

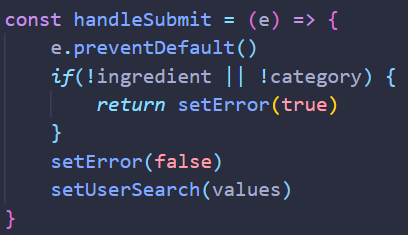


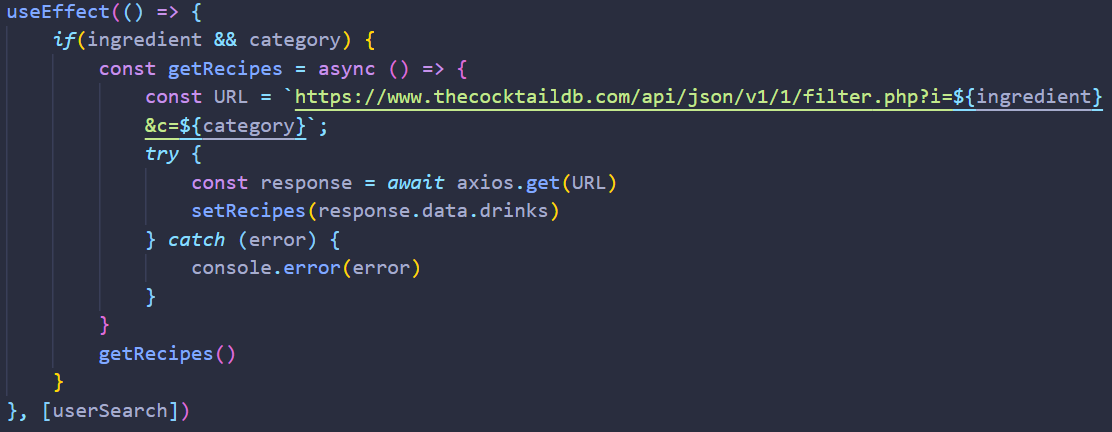


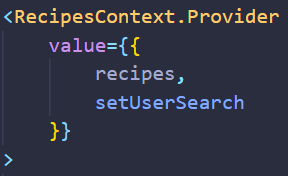
**12**. **Guardando datos validados**: **importamos** el **Provider** en **App.js** y lo incluimos después del de categorías.



Luego vamos a **Form.js**, **importamos** el **contexto creado**, **desestructuramos** el **valor** que n**os interesa** y lo igualamos a useContext pasándole como parámetro el contexto creado. Finalmente, **después de validar guardamos los datos**.

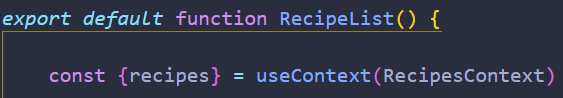


**13**. **Obteniendo las recetas de la API**: dentro de **DrinkRecipesContex.js** hacemos un **useEffect** que tome como **dependencia** los **datos validados**, es decir, cada vez que se ingresen nuevos datos se ejecutara el pedido. Allí preguntamos **si existen los datos** y de ser así hacemos una **función asincrónica** que va a **consultarle** a la **API** **pasándole** un **ingrediente** y una **categoría**, **esperando** recibir una **receta**. Luego de tenerla la **guardamos** en el **estado**.

**14**. **Usando las recetas en el componente**: dentro de **DrinkRecipesContex.js** **exportamos** el **array** de **recetas** y luego **creamos** un nuevo **componente** **RecipeList.js**.

En **RecipeList.js** importamos **useContext** y el **contexto creado** y **desestructuramos** las **recetas**. Luego hacemos un **map** y en **cada iteración** vamos a **imprimir** un **nuevo componente (Recipe.js)** que va a **desplegar a cada receta**.



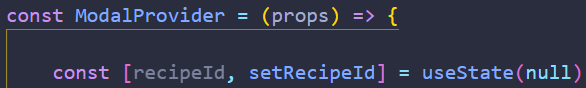


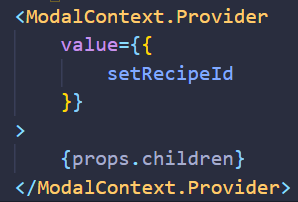


**15**. **Creando el componente de Receta**: creamos **Recipe.js desestructuramos** las 2 **propiedades del objeto** y **armamos** una **tarjeta** con clases de **Bootstrap**.

**16**. **Creando el context para la ventana modal**: dentro de **context** **creamos** el archivo **ModalContext.js**. Allí por el momento vamos a **crear** un **estado** para **identificar** el **id** de la **receta** que se está **seleccionando** (el estado lo iniciamos en null).

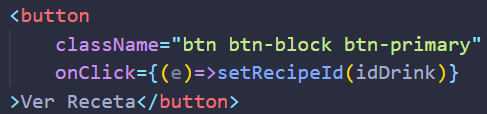




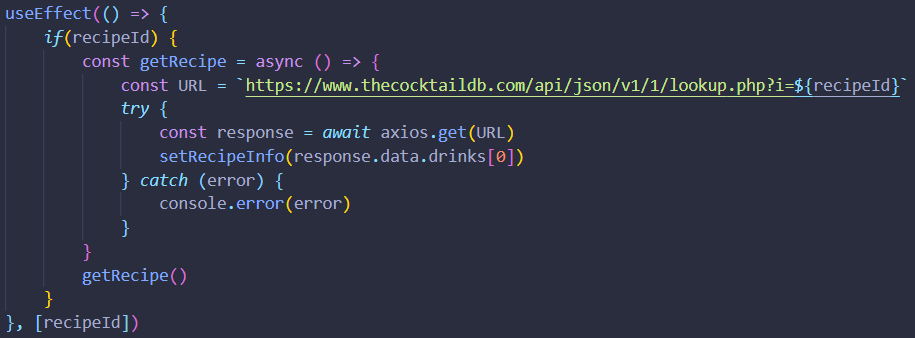


Importamos el **Provider** en **App.js** y luego **dentro** del componente **Recipe.js** el **contexto**. **Allí** en el <**Button**> bajo el evento **onClick** le **pasamos** el **id** de la **receta**.



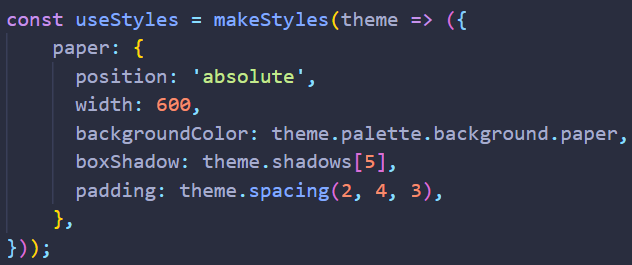


**17**. **Solicitando la receta**: dentro de **ModalContext.js** debemos **crear** un **nuevo estado** para **guardar** **receta** que nos va a traer la **API**. Luego hacemos un **useEffect** que tome como **dependencia** **recipeId**, de modo que si alguien selecciona una receta se hace el pedido a la API. **Si** **hay** un **recipeID** hacemos el **pedido asíncrono** y lo **guardamos en el estado**.

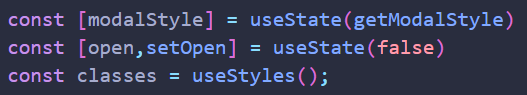


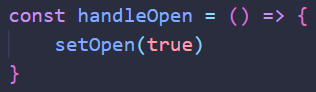
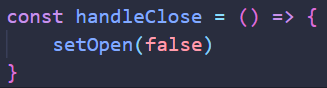
**18**. **Creando la ventana Modal**: debemos ingresar a [Material-UI](https://material-ui.com/components/modal/#modal) y **seleccionar** una ventana **modal**. Luego **copiamos** el **código** **dentro** del componente **Recipe.js**.

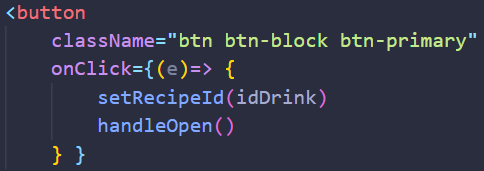
**a)** Importamos la librería y el modal.

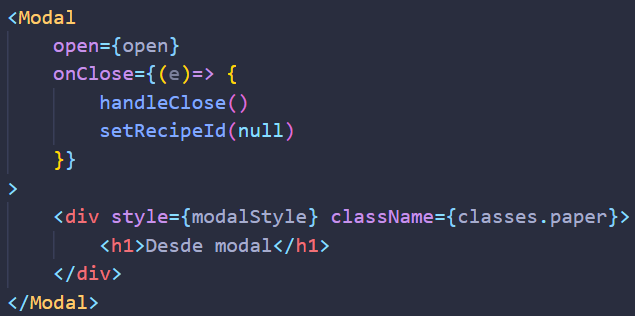
**b)** **Pegamos** los **estilos**, uno nos va a **servir** para la **propiedad style** y **otro** para el **className**.

**c)** Dentro del **componente** **creamos dos estados**, **uno** que **toma** los **styles** y el **otro** que va a **determinar** **si** **hay que mostrar** o no el **Modal**. **Luego** creamos una **constante** que va a **ejecutar** los **estilos** del **className**.

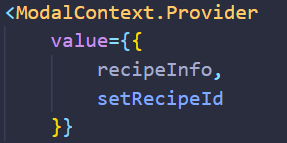
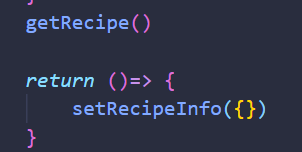
****

**d)** Debemos **crear** las **funciones** que se encargar de **determinar** **si** la **ventana** se debe **mostrar** **o no**.

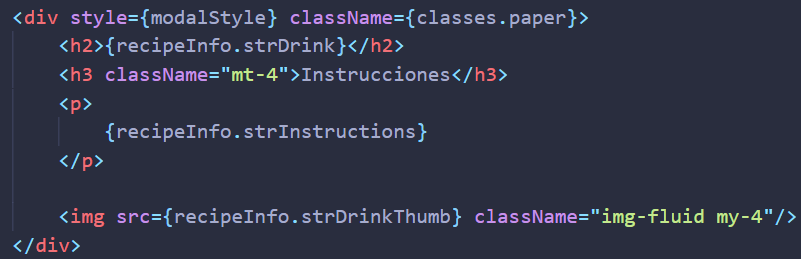
**e)** Luego al <**button**>, además de la función que determina el id, le **agregamos** la **función** que **abre la ventana**.

**f)** **Finalmente** **usamos** el **Modal** como un **componente** y le **damos dos propiedades**: **open** con el **valor del estado** (abierto o cerrado), y bajo el evento **OnClose** **ejecutamos** la **función** que **cierra la ventana** **y reiniciamos el ID de la receta**, ya que no tenemos ninguna seleccionada al cerrar

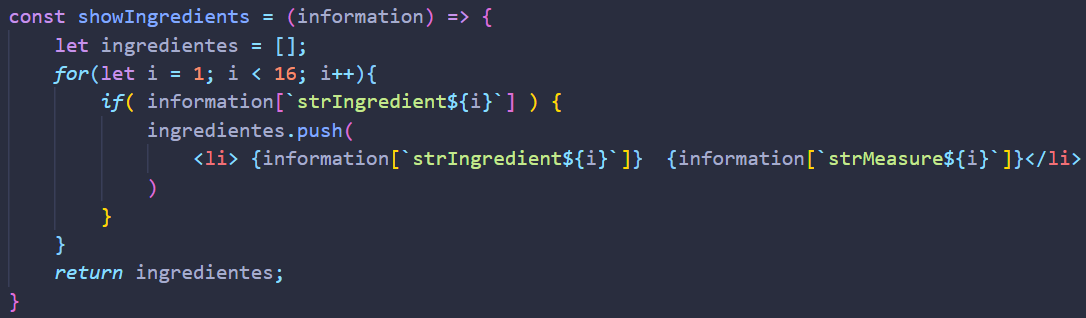
**19**. **Mostrando datos en la ventana modal**: debemos **compartir** la **receta** desde **ModalContext.js** **hacia** el componente **Recipe.js**. **Importante**: **luego de obtener la receta** debemos **reiniciar el estado que la almacena** para **evitar que se muestre la misma cuando se selecciona otra.**



Dentro de **Recipe.js** **tomamos** el **dato** de la **receta** y **utilizamos** sus **propiedades** dentro del **Modal**.



**20**. **Colocando los ingredientes y valores**: como en la API los ingredientes y valores no vienen en un array, sino que se enumeran del 1 al 16 mostrando valores nulos en caso de no tener 16 ingredientes, debemos hacer lo siguiente:

**Creamos** una **función** que **recorra** el **objeto 16 veces** y en **caso que** el haya un **ingrediente** con un valor **lo agregue al array**. **Si hay un ingrediente** siempre **va a haber una medida**, por lo cual **la agregamos también**. Finalmente retornamos el array.

Luego en el final del **Modal** mandamos a **llamar** la **función** para renderizar la lista de ingredientes y cantidades.

