UNIVERSIDAD DON BOSCO

Materia: Diseño de Software Multiplataforma

Grupo Teórico: 01L

Tema Para Evaluar: Desafío Práctico #2

Docente: Karens Medrano

Integrantes:

- **David Isaí Alfaro López AL201498**
- **Keila Jael Rivas Jiménez RJ202336**

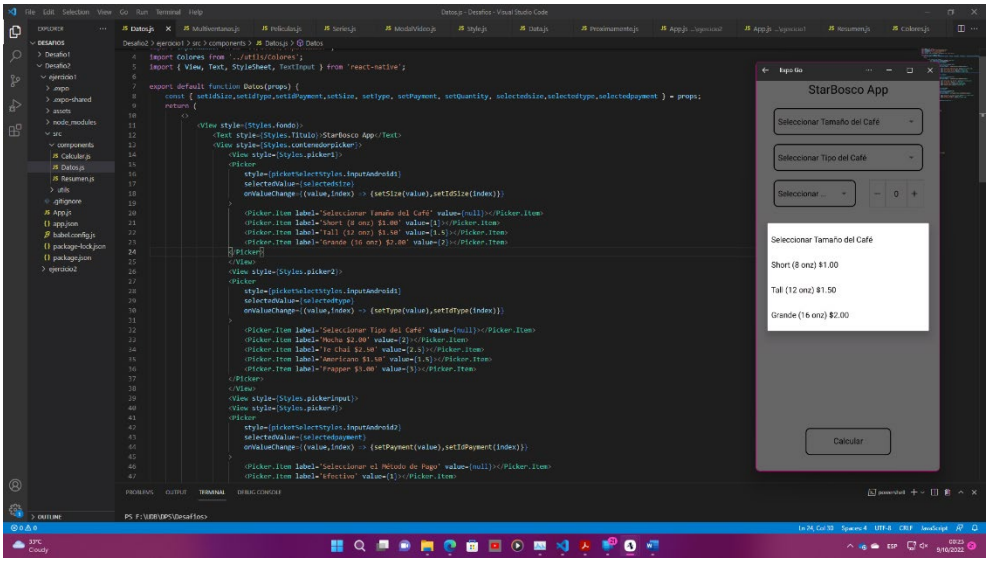
San Salvador, 09 de octubre de 2022

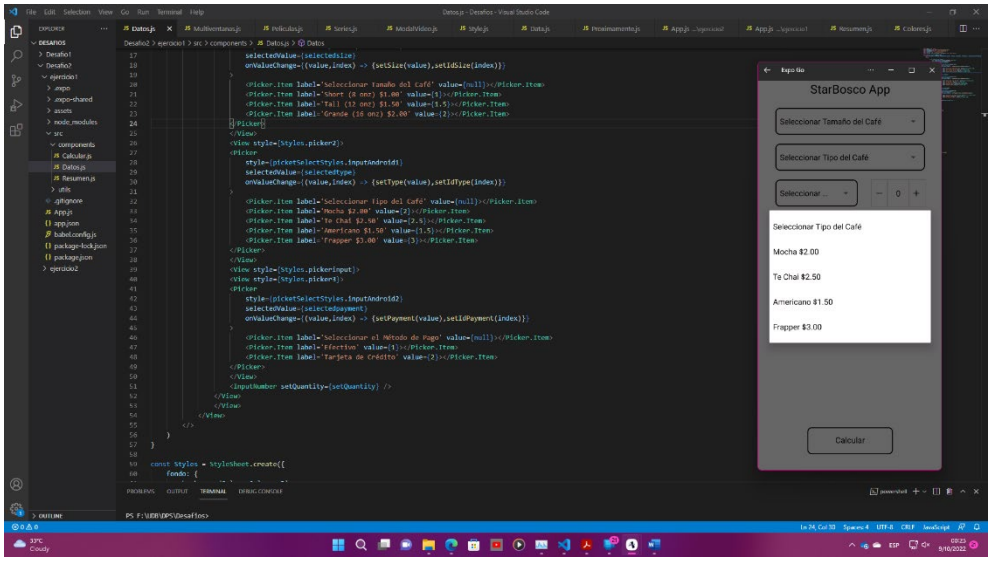
Proceso: Ejercicio #1

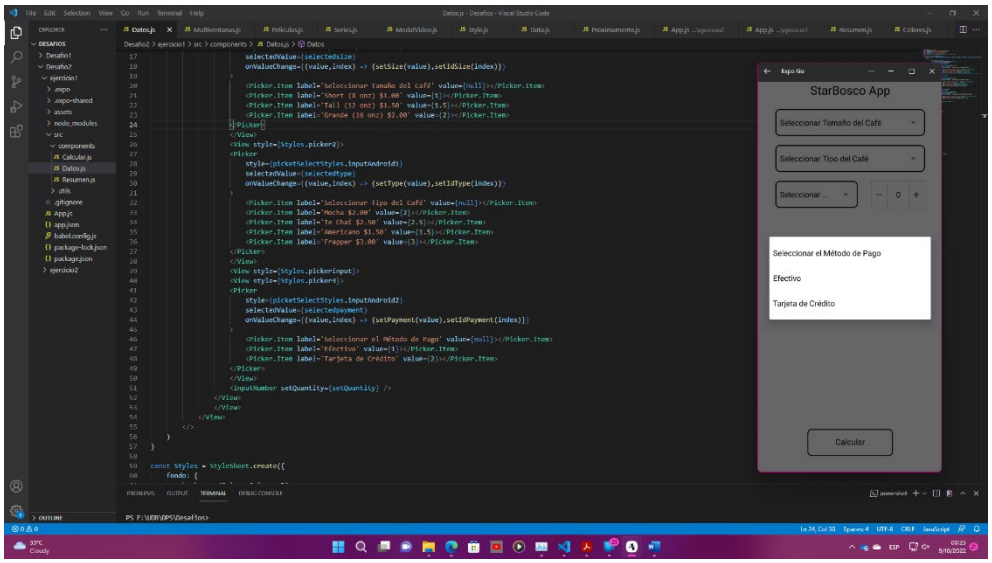
Paso #1: Se crean los componentes de la aplicación, los cuales se llaman: Calcular, Datos, Resumen. Se inicia con el componente de Datos el cual va a contener los pickers para seleccionar el tamaño del café, el tipo del mismo, y el método de pago, así como también, un Text Input de tipo número con el que se va a seleccionar la cantidad de cafés.

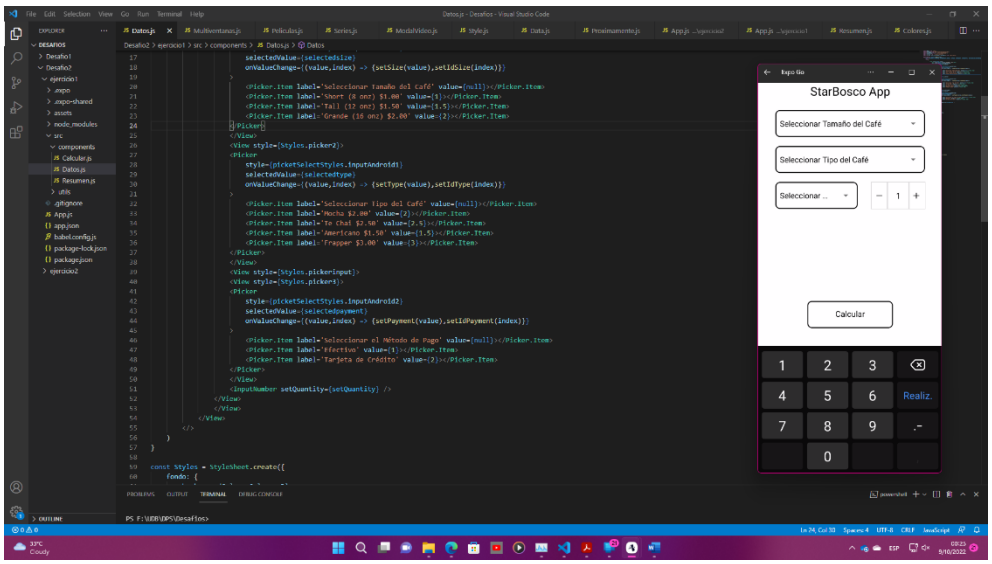
Además, se crean una constante de estilos llamada "style" para darle un borde redondeado de color negro a los pickers.

Nota: Se utiliza la librería "react-native-paper" para el TextInput de tipo numérico.



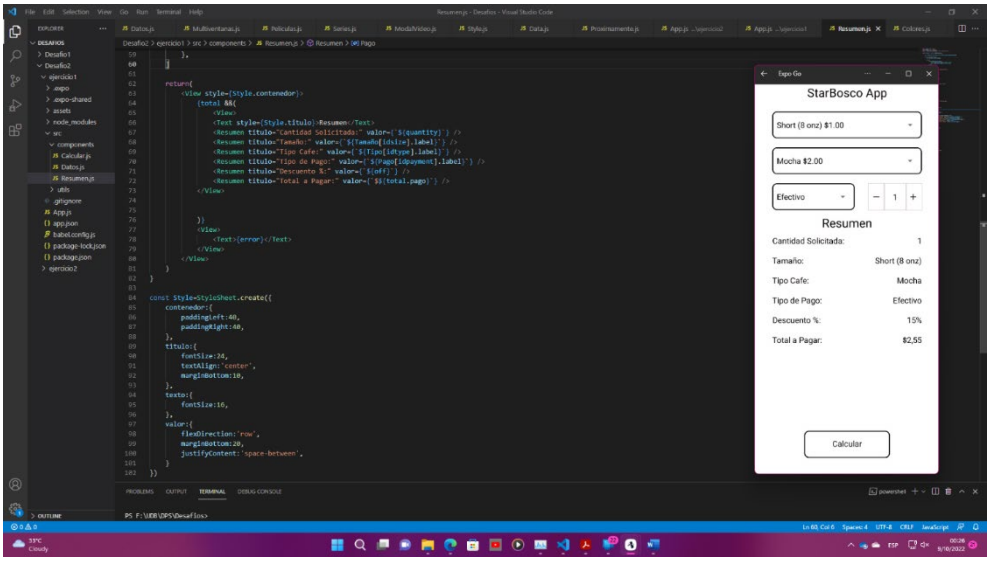


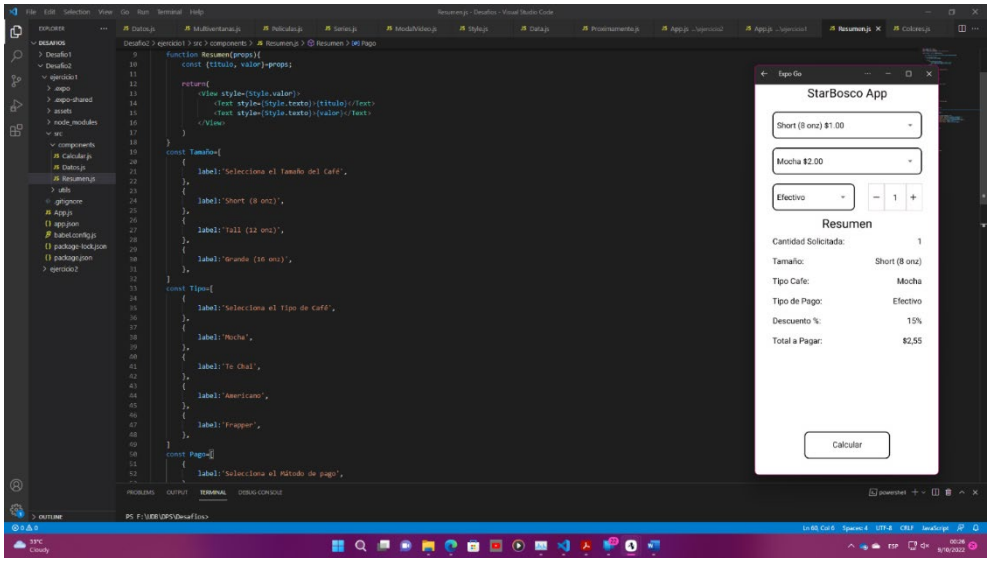

StarBosco App
Seleccionar Tamaño del Café
Seleccionar Tipo del Café
Seleccionar ...
0
Seleccionar el Método de Pago
Efectivo
Tarjeta de Crédito
Calcular


StarBosco App
Seleccionar Tamaño del Café
Seleccionar Tipo del Café
Seleccionar ...
1
Calcular
1
2
3
4
5
6
Realiz.
7
8
9
.-
0

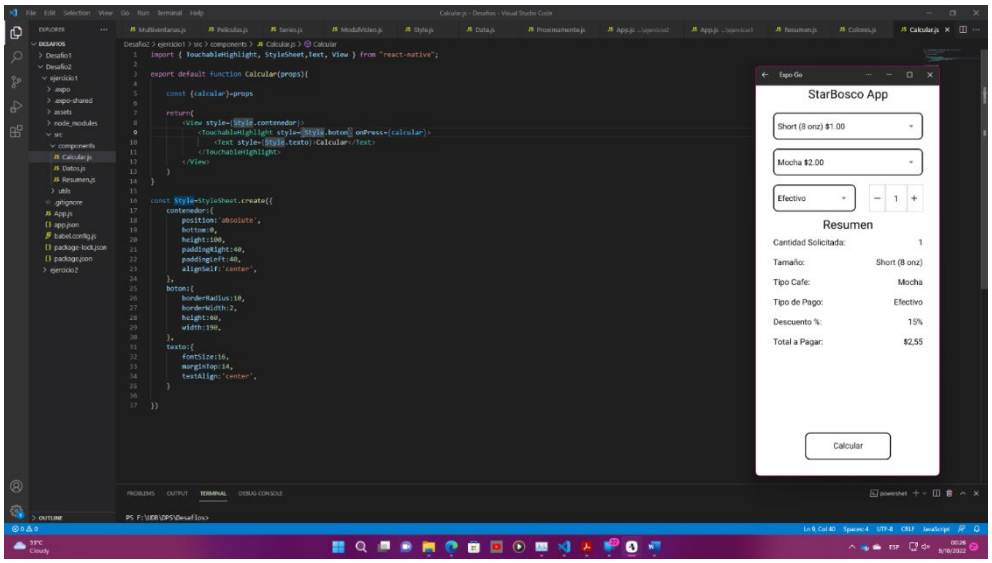
Paso #2: Se realiza el componente “Resumen.js” el cual va a dar el resultado detallado de lo que se va a comprar, es decir, el tamaño del café, el tipo del café, el método de pago, la cantidad solicitada y un descuento realizado según el método de pago escogido, y el total a pagar por el usuario.

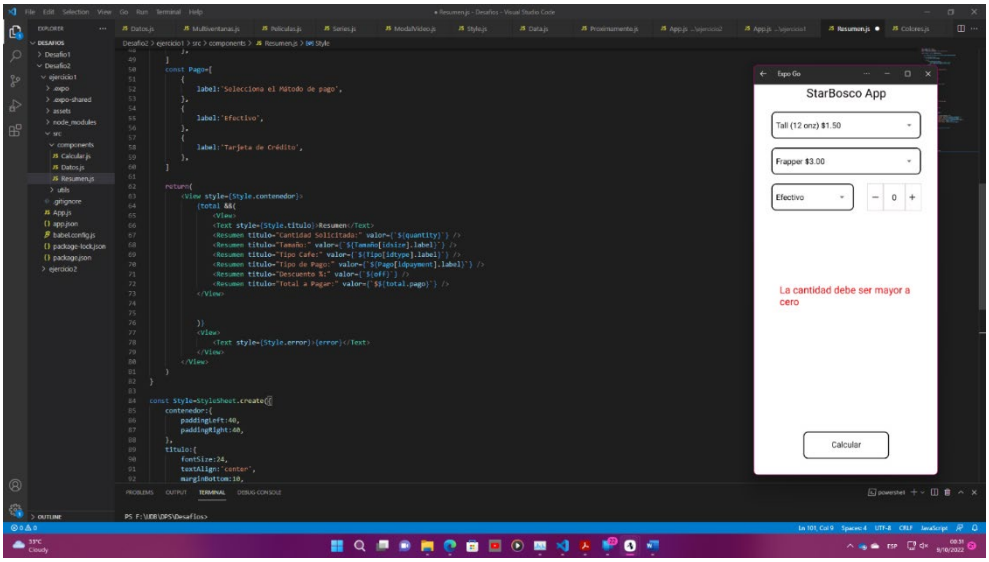
Nota: Se utilizaron unos Arrays que contienen los label de los pickers del componente “Datos.js” ya que el valor que salen de los pickers son numéricos y por ende son utilizados para los cálculos matemáticos. Se muestra el label adecuado gracias al índice obtenido de los pickers que se actualiza con el useState en la App.js





Paso #3: Se crea el componente “Calcular.js” el cual es un botón que realiza la operación mostrada en el componente “Resumen.js”. Sin embargo, se ha utilizado el hook “useEffect” para que dicho componente se actualiza si el usuario modifica el valor de los pickers o la cantidad sin necesidad de presionar dicho botón. No obstante, si hace falta algún dato, el useEffect no se va a realizar, y si se presiona el botón mandará un mensaje de error, según sea el dato que falta.

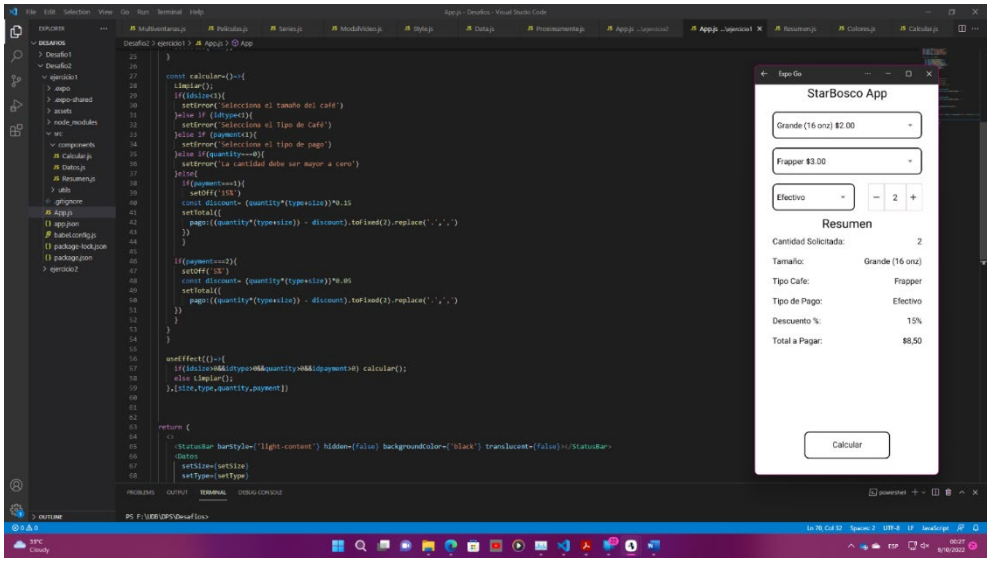


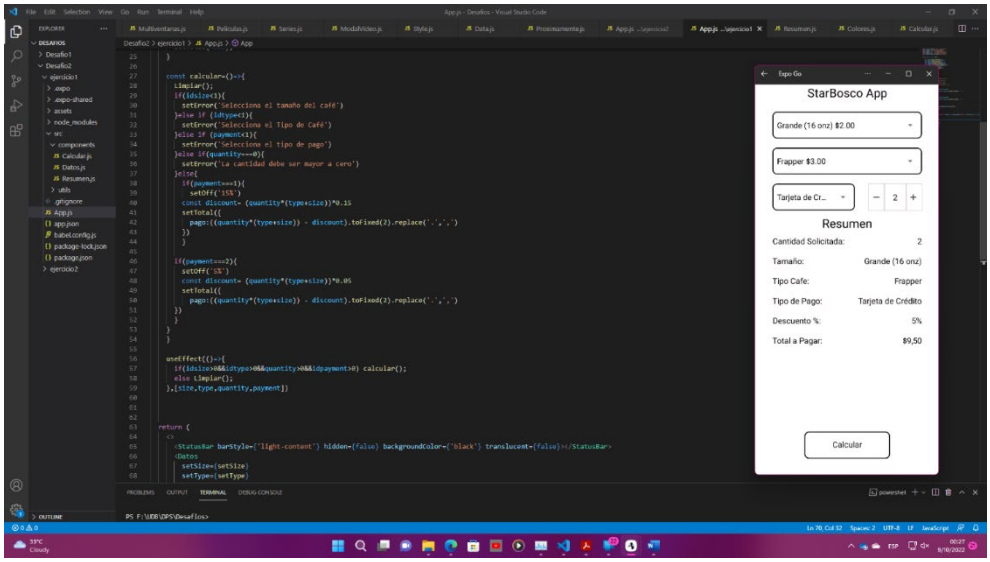


Paso #4: Por último, se modifica el App.js donde se realizan los cálculos matemáticos mediante la constante “calcular” haciendo uso del useState para obtener los datos que viene del componente “Datos.js” por medio del setState y mandarlo al componente “Resumen.js” una vez se haya realizado el cálculo.

Lo más destacable de “calcular” es que si el método seleccionado por el usuario es “Efectivo” se hará un descuento total del 15% mientras que si se usa el método “Tarjeta de crédito” solo será de 5%, además que se realizan algunas validaciones dentro de la misma.

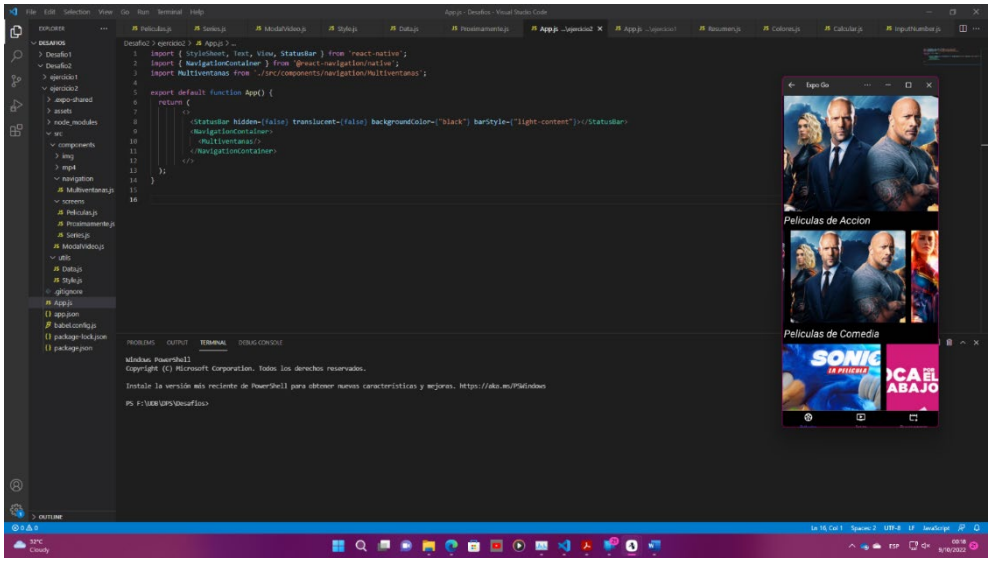
Y el useEffect utiliza la constante “calcular” para actualizar el componente “Resumen.js”



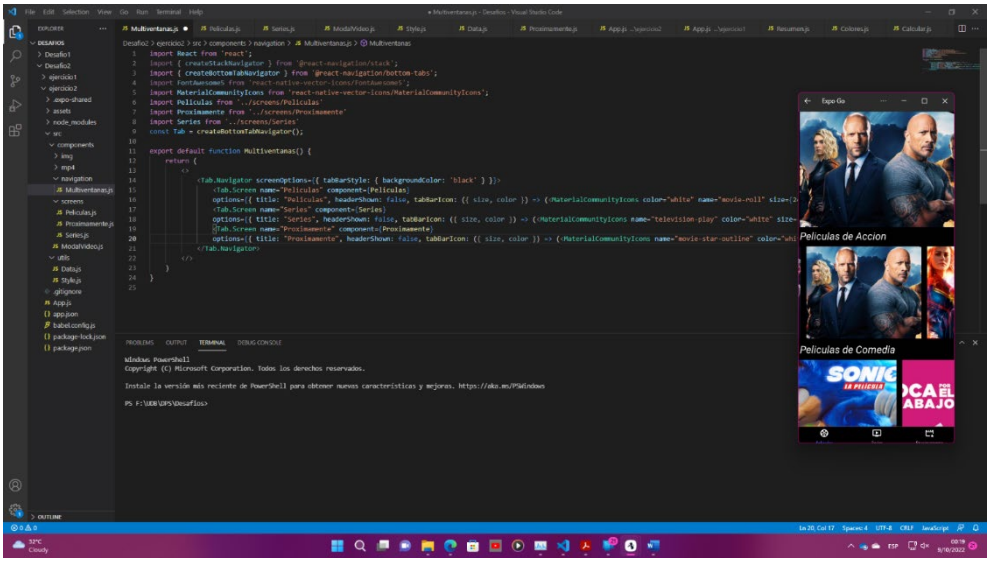


Proceso: Ejercicio #2:

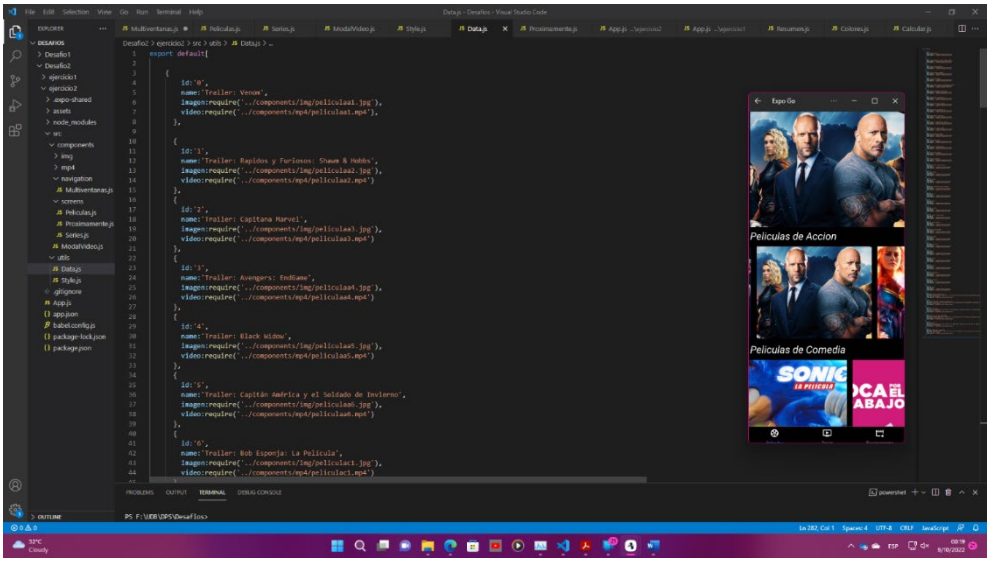
Paso #1: Primero se realiza la recolección de información, es decir, obtener todas las imágenes y videos a utilizar para la aplicación.
Luego en el App.js solamente se va a utilizar un NavigationContainer que se utiliza para manejar el Tab.Navigator que contiene las pantallas Películas, Series y Próximamente.

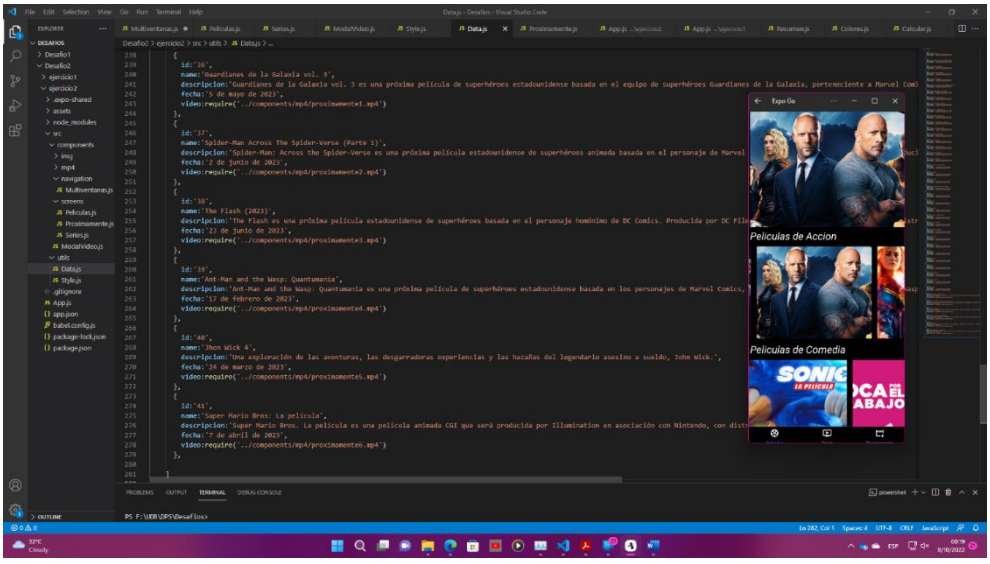


Paso #2: En el componente Multiventanas.js se colocan las pantallas de Películas, Series y Próximamente, mencionadas anteriormente, y las cuales tienen íconos de tipo MaterialCommunityIcons.

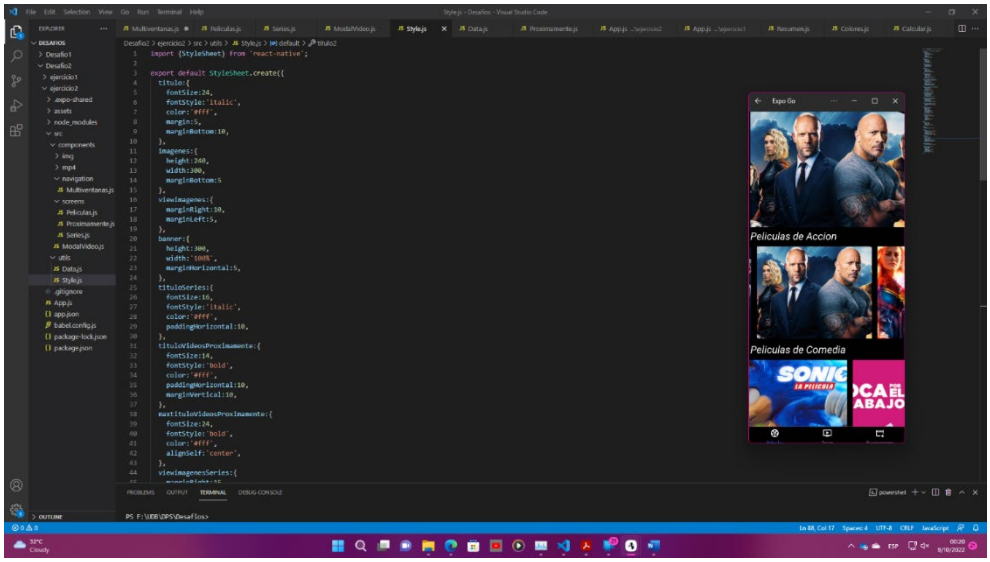


Paso #3: Se crea un Array llamado Data.js el cual se utilizará para leer todas las propiedades necesarias para las pantallas de Películas, Series y próximamente, obteniendo de este, el nombre, el número de temporadas de las series, las descripciones y fechas de estreno de la pantalla próximamente, así como la ubicación de las imágenes y videos correspondientes.

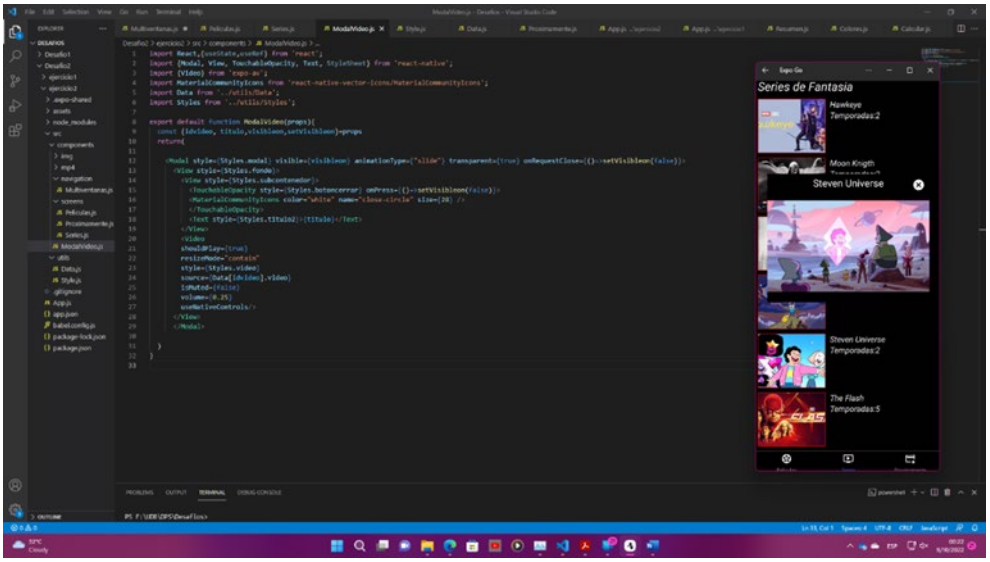




Paso #4: Se crea un archivo simulando un .css llamado Style.js el cual permite abarcar todos los estilos a utilizar en las diferentes pantallas de la aplicación.



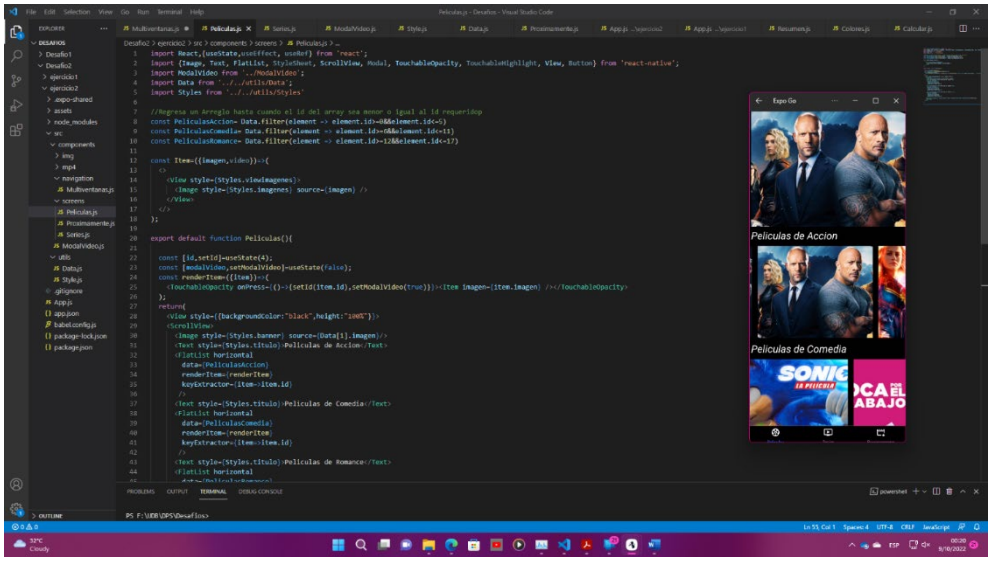
Paso #5: Luego se crea un componente llamado ModalVideo.js el cual sirve para qu cuando se presione la imagen de una película o serie se abra una ventana que tenga el reproductor de video que muestre el tráiler correspondiente, utilizando el id del Array Data.js, el video se reproduce de forma automática, y también se puede cerrar una vez se haya terminado de ver por el usuario.

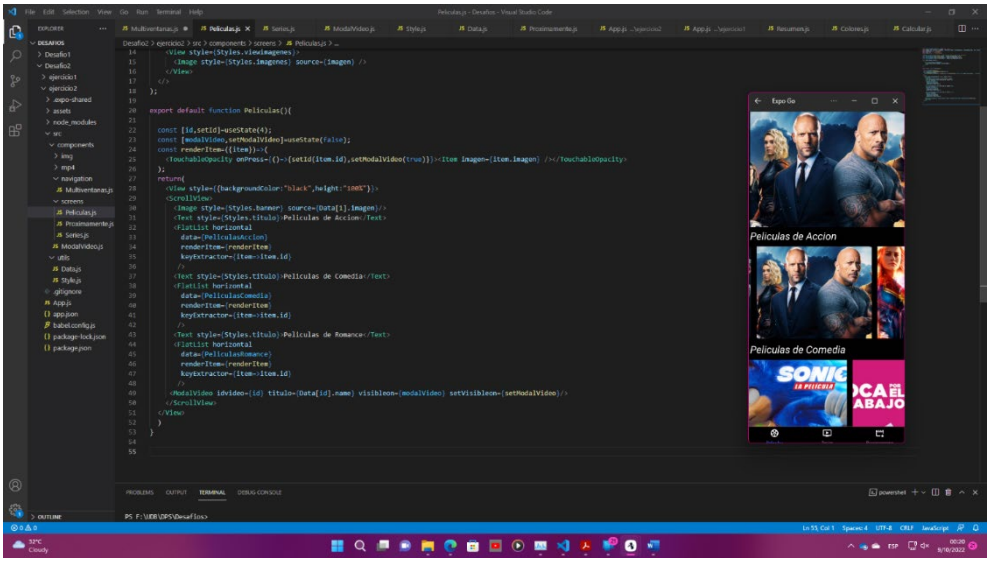


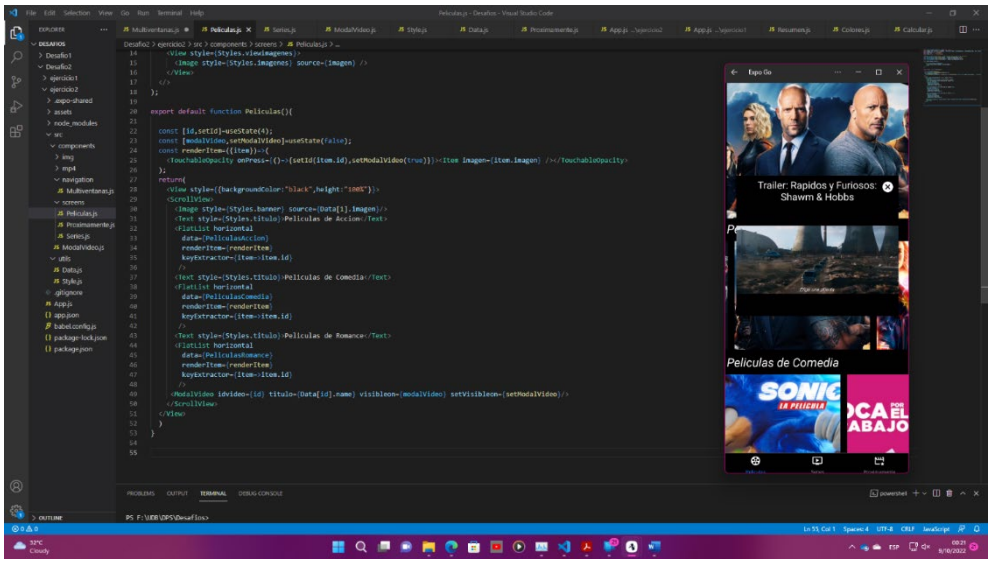
Paso #6: Se utiliza el componente Peliculas.js, donde cual primero se crean tres nuevos Arrays de datos que serán utilizados para ser leídos por los FlatList, que en este caso están en forma horizontal, donde el renderItem incluye la Imagen correspondiente.

Todas las imágenes están dentro de un botón (TouchableOpacity) para que cuando se dan click se abra (o se haga visible) un ModalVideo que reproduce el video de acuerdo con el id obtenido de la data de la FlatList.

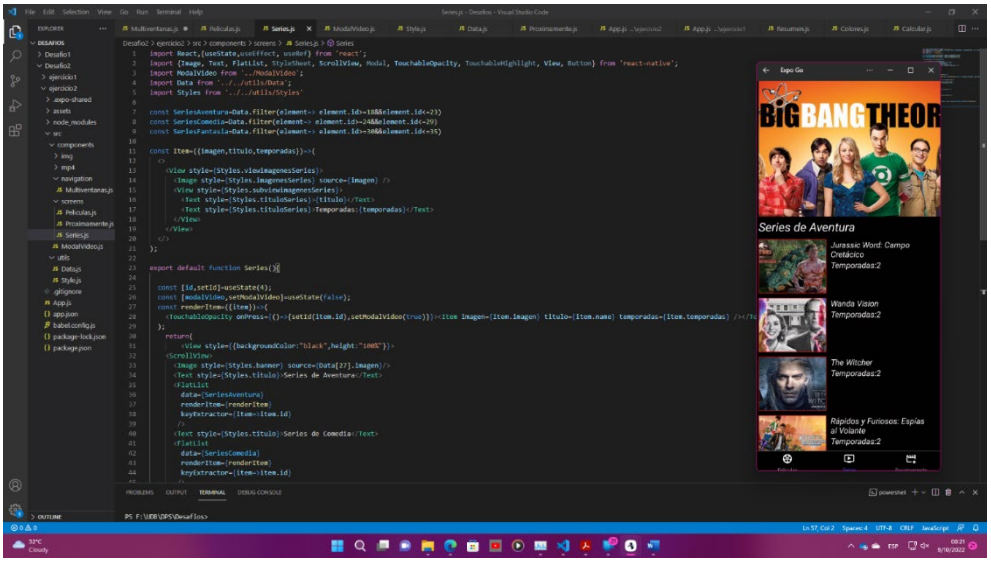
Adicionalmente se añade una imagen que funciona como banner de la pantalla.

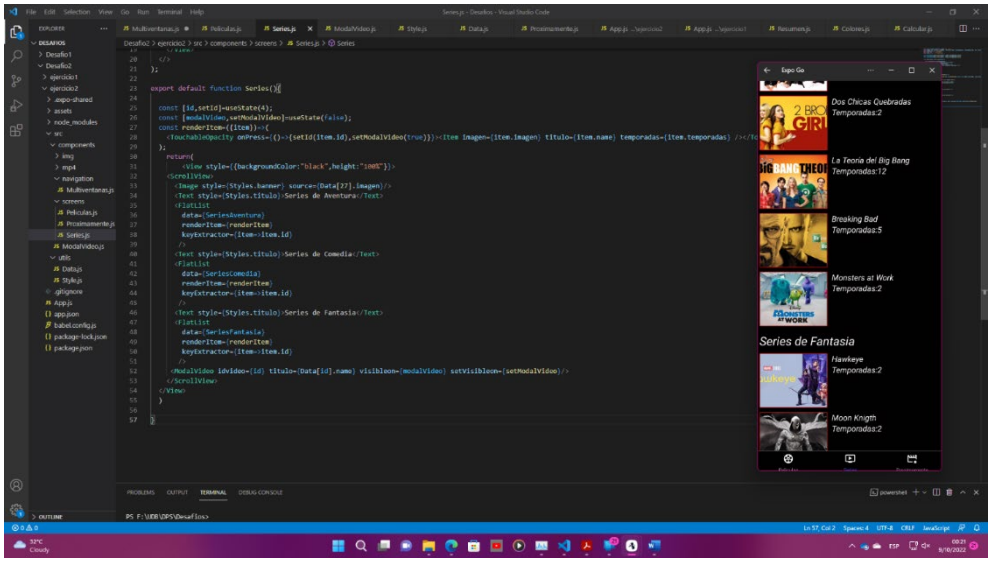


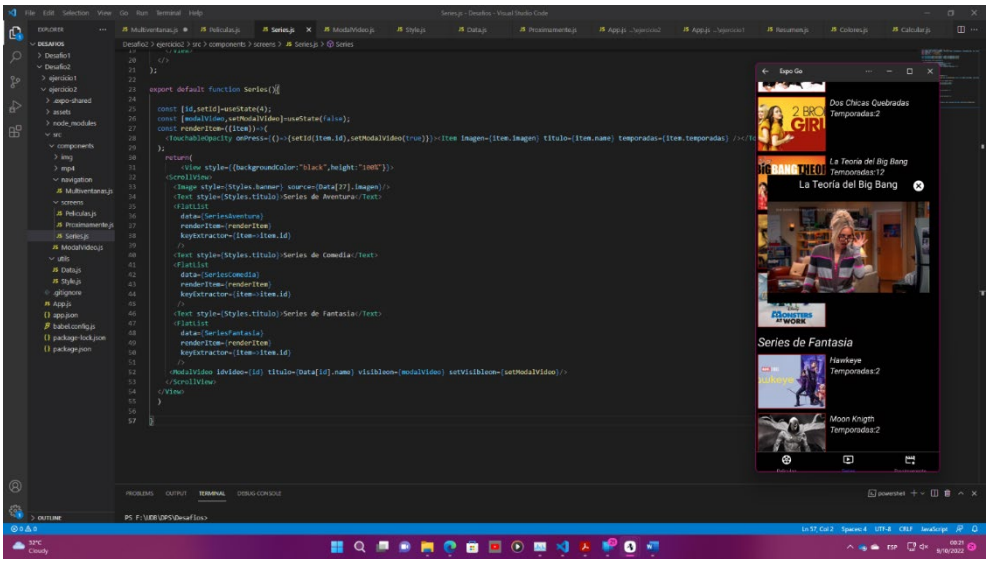




Paso #7: Se utiliza el componente Series.js el cual funciona de manera muy similar al componente de Películas.js solo modificando los Arrays que se obtienen ya que se necesitan las series en este caso. Los FlatList se muestran de forma vertical en este caso, y el renderItem de estos incluyen además de las imágenes, el nombre de la serie, así como el número de temporadas que poseen. Siempre incluyen un banner y el ModalVideo que se abre cuando se presiona la imagen de la serie.




Dos Chicas Quebradas
Temporadas:2
La Teoria del Big Bang
Temporadas:12
Breaking Bad
Temporadas:5
Monsters at Work
Temporadas:2
Series de Fantasia
Hawkeye
Temporadas:2
Moon Knigth
Temporadas:2


Dos Chicas Quebradas
Temporadas:2
La Teoria del Big Bang
Temporadas:12
La Teoria del Big Bang
Series de Fantasia
Hawkeye
Temporadas:2
Moon Knigth
Temporadas:2

Paso #8: Por ultimo se utiliza el componente Próximamente.js el cual funciona un poco diferente a las otras pantallas, ya que aquí solo se utiliza un Array y un FlatList, y el renderItem ya incluye el componente de Video, que siempre se obtiene por el id que proviene de la data de FlatList, además del nombre, la descripción y la fecha de estreno de la película.