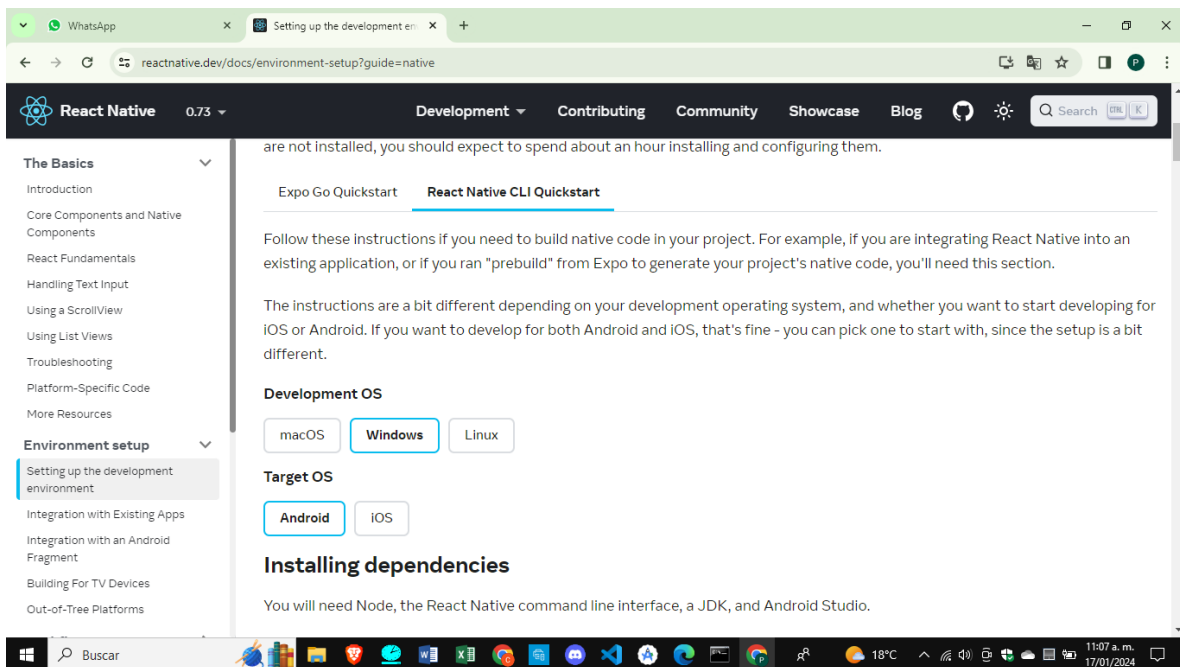
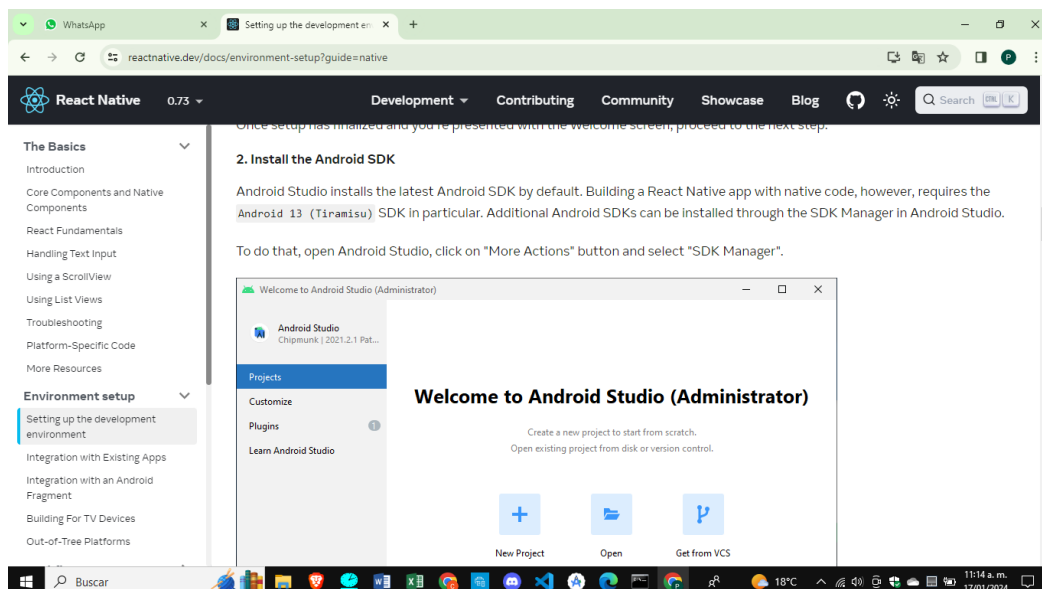


# Prueba técnica Desarrollador de aplicaciones móviles REACT NATIVE JR

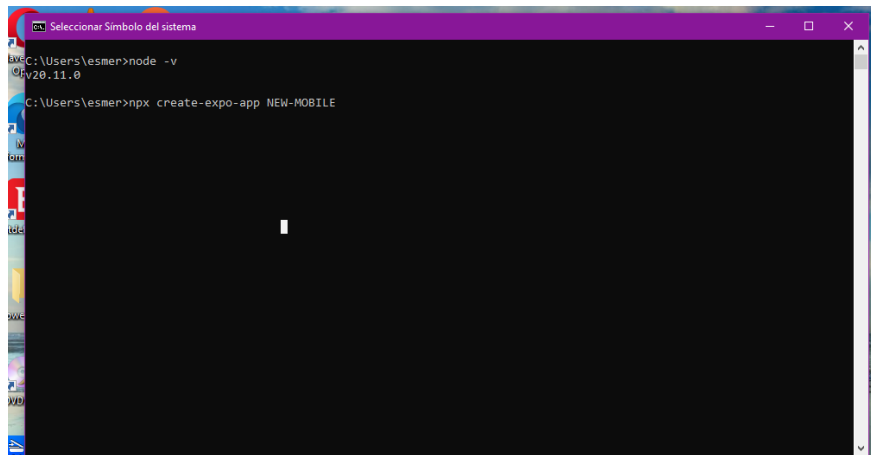
Lo primero que hice fue ver si tenía las herramientas necesarias o de lo contrario comenzar con la instalación de lo que se necesita para poder trabajar con React-Native. En la siguiente imagen esta la documentación de React-Native y todo lo que se necesita para su instalación.



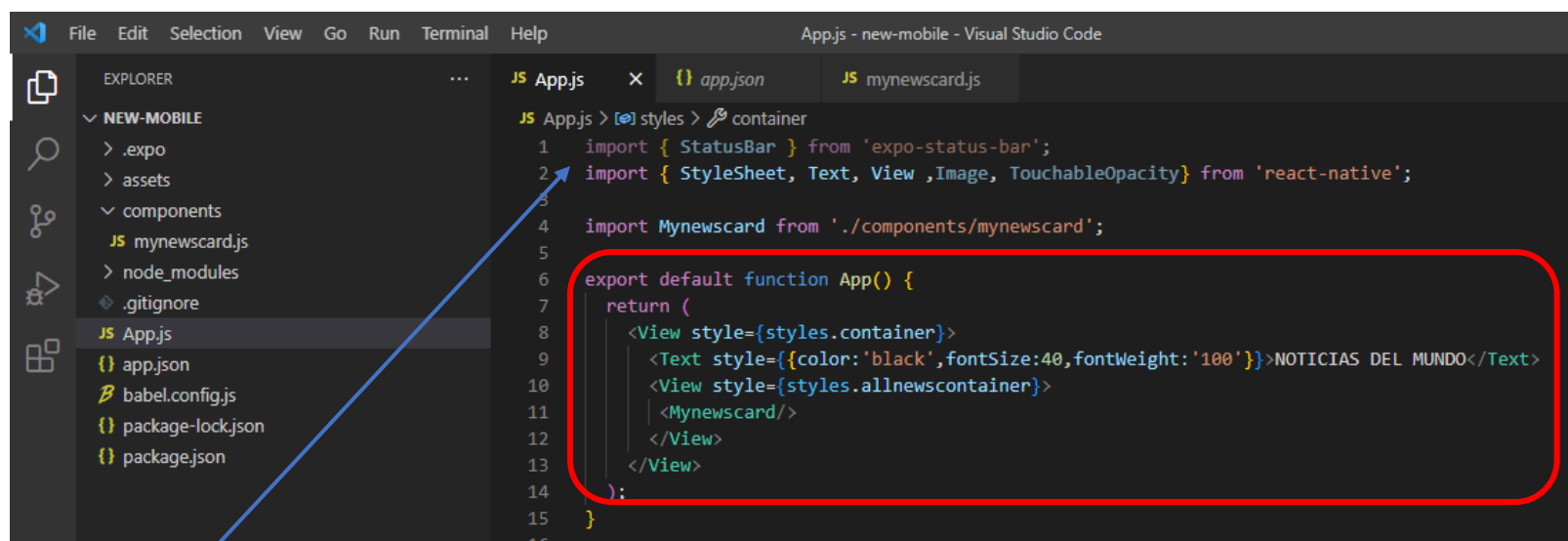
En mi caso no tenía instalado ninguna de las herramientas y empecé con la instalación de React y los componentes que requerían por ejemplo Android-Studio ya que también era una de las herramientas que requería.



Una vez teniendo todo lo necesario comencé con la creación del proyecto.

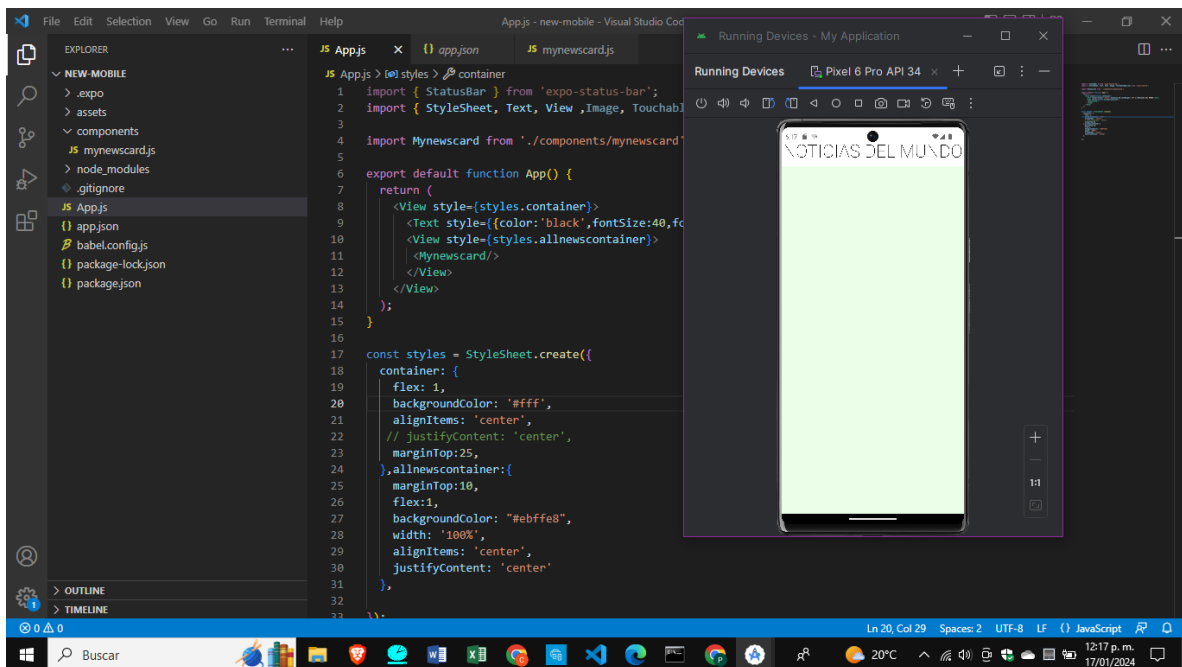


Una vez creado el proyecto lo abrí en un editor de texto en este caso fue VisualStudio-Code.



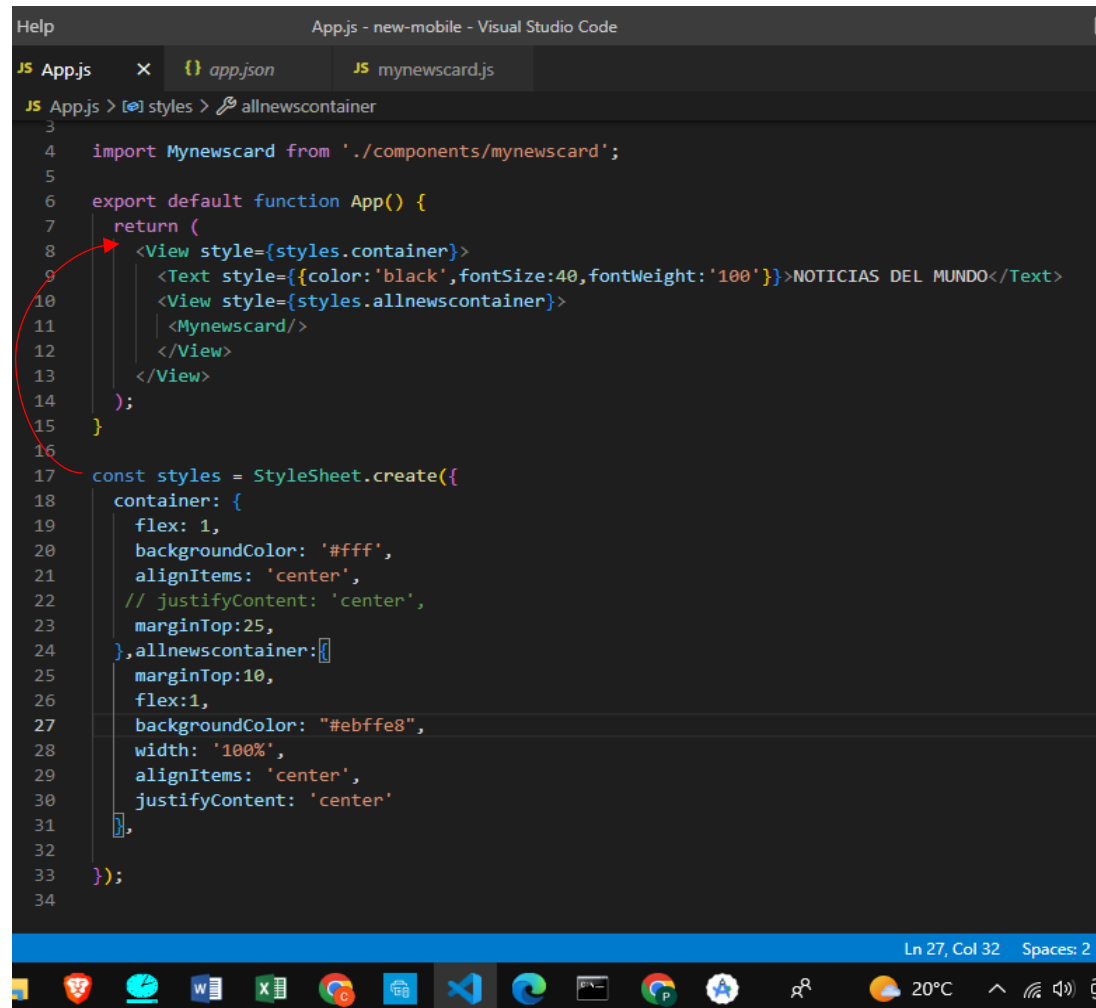
Ya dentro del proyecto empecé con la implementación de código en los import, mandamos todas las librerías o extensiones que vallamos a utilizar por ejemplo: Estilos, Textos, Vistas, Imágenes y acciones que se puedan realizar en la pantalla.

Dentro de lo que está marcado en rojo se realizó la primera vista de la pantalla y dentro de la vista se agregó un texto con ciertos caracteres y un texto que dice “Noticias del mundo” que será el Titulo de la primera vista y después con un emulador verificamos que se agregó correctamente.

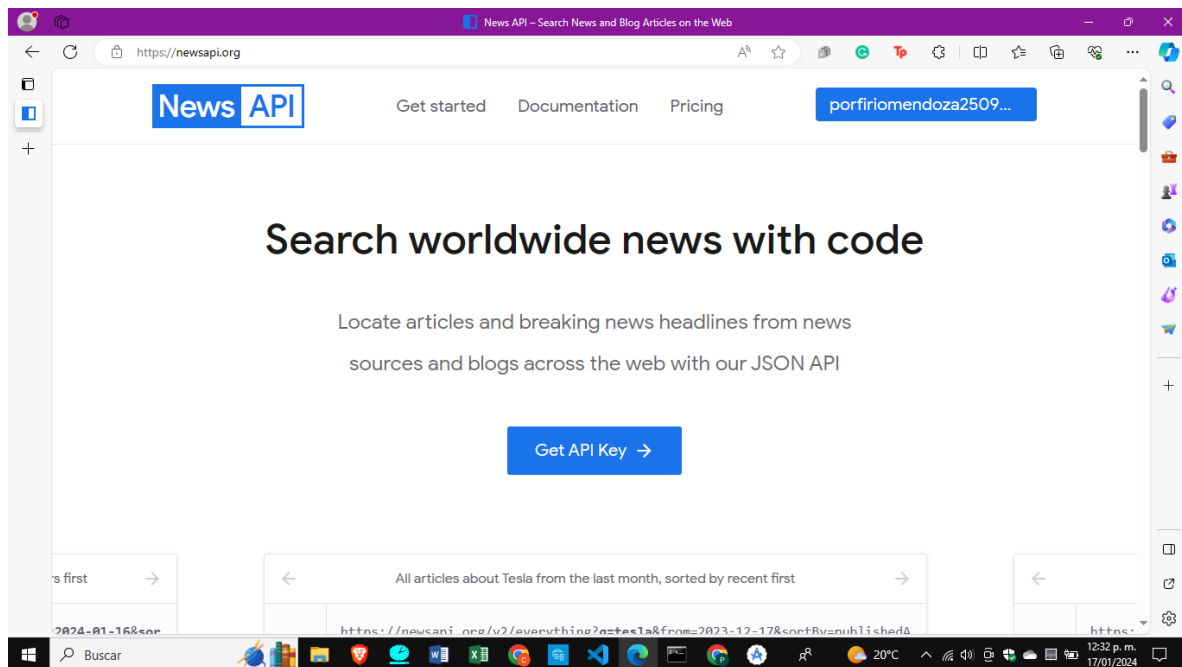


Como se puede ver en el emulador se muestra una pantalla con el título de Noticias del mundo.

Se agregaron estilos como el color, tamaño y posición del texto luego se mandó a llamar en la vista principal para que tomara todos los cambios que se le asignaron para la pantalla.



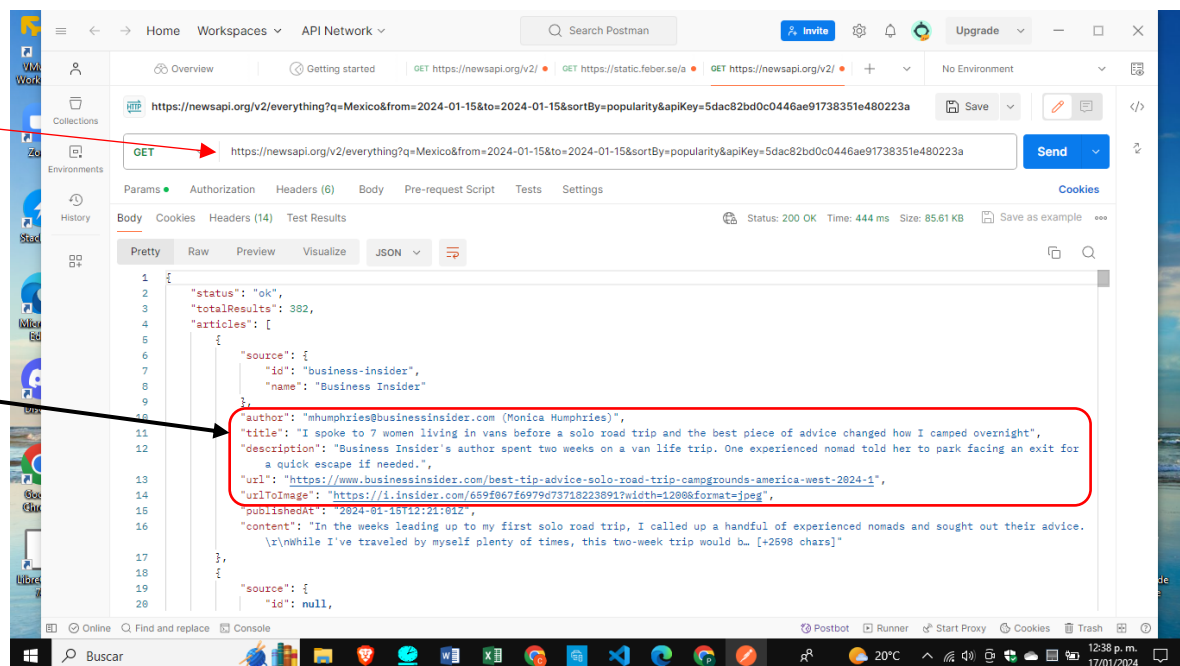
Una vez creada la vista se necesita obtener el api que nos solicitan en la siguiente página la obtenemos junto con la Key-api.



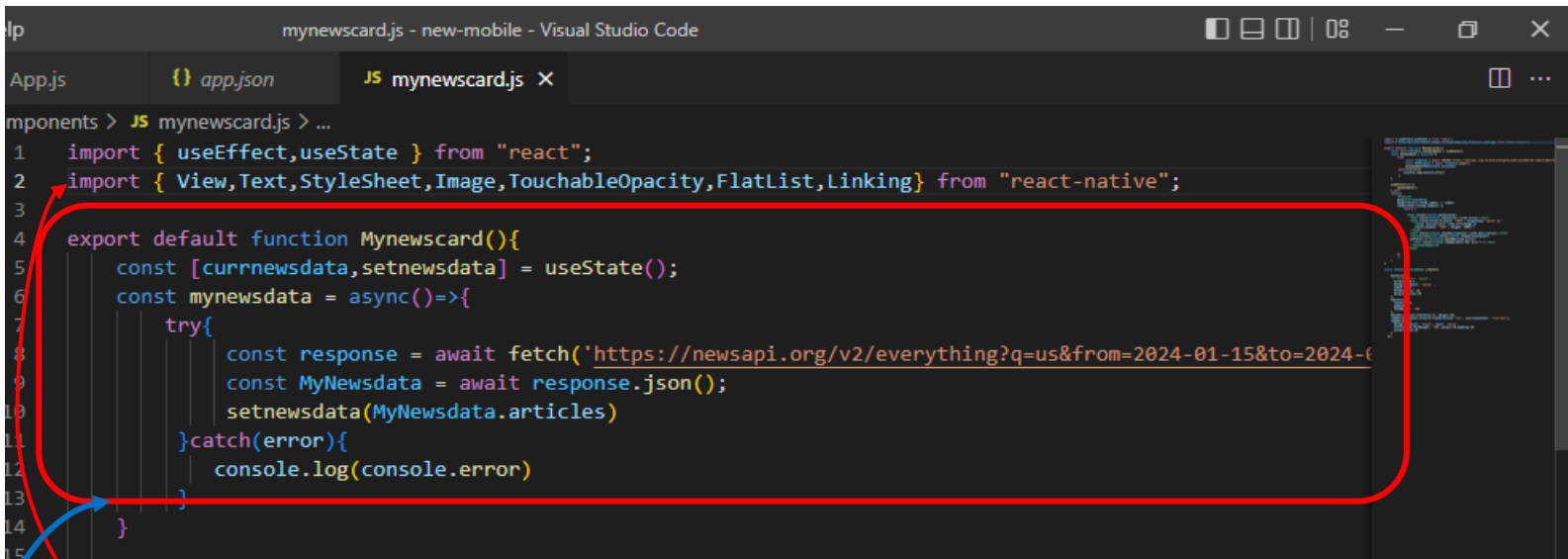
Ya que tenemos el api procedemos a probar la url para comprobar que tenemos datos dentro del json en mi caso utilicé postman para verificar que tipo de datos nos trae la url.

Insertamos la url para ver lo que nos trae y comprobar que si nos devuelve valores.

Nos aparece lo que son el titulo la descripción y una url para la imagen y efectivamente son los datos que nos solicitan y procedemos a decodificarlos



Dentro del proyecto creamos un nuevo archivo mynews.js con la extensión de .js al final porque es aquí donde obtendremos los valores del json para mostrarlos en la pantalla



```
1 import { useEffect,useState } from "react";
2 import { View,Text,StyleSheet,Image,TouchableOpacity,FlatList,Linking} from "react-native";
3
4 export default function Mynewscard(){
5   const [currnewsdata,setnewsdata] = useState();
6   const mynewsdata = async()=>{
7     try{
8       const response = await fetch('https://newsapi.org/v2/everything?q=us&from=2024-01-15&to=2024-01-15');
9       const MyNewsdata = await response.json();
10      setnewsdata(MyNewsdata.articles)
11    }catch(error){
12      console.log(console.error)
13    }
14  }
15 }
```

Nuevamente importamos las librerías que vamos a utilizar para llevar a cabo esta tarea y que nos lo proporciona react-native.

Procedemos a desenvolver el json para obtener los datos y poder utilizarlos, creamos una función y dentro utilizamos un try-catch para poder desenvolver el json de forma segura y que en caso de obtener errores podamos manejarlos y ver porque no se realizó la petición

Después creamos un FlatList donde vamos a empezar a mostrar los datos en pantalla en forma de lista dentro de los view tenemos lo que son el ítem.title, ítem.description y ítem.url estos datos son los que obtenemos del json y a los cuales podemos acceder una vez que sean obtenidos por el try-catch y pasados a código y dentro de cada view tenemos los styles para cada vista que fueron creados en otra parte del código a las cuales podemos acceder llamándolos dentro de cada vista y dependiendo de cuales sean los styles que se requieran.

```
16   useEffect(()=>{
17     mynewsdata();
18   },[]);
19   return(
20     <FlatList
21       data={currnewsdata}
22       keyExtractor={({Item,index}) => index}
23       renderItem={({item,index})=>{
24         return (
25           <View style={styles.mynewscard}>
26             <Text style={styles.newstitle}>{item.title}</Text>
27             <View style={{flex:0,width: '100%', alignItems:'center'}}>
28               <Image source={{uri: item.urlToImage,}}
29                 style={{width: '100%', height: 200}}/>
30             </View>
31             <Text style={styles.newsdescription}>{item.description}</Text>
32             <TouchableOpacity style={styles.readmorecontainer}
33               onPress={()=>{Linking.openURL(item.url)}}>
34               <Text style={styles.readmorebtn}>Ver más{">"}</Text>
35             </TouchableOpacity>
36           </View>
37         );
38       }
39     );
40   />
41 )
42 }
43 }
```

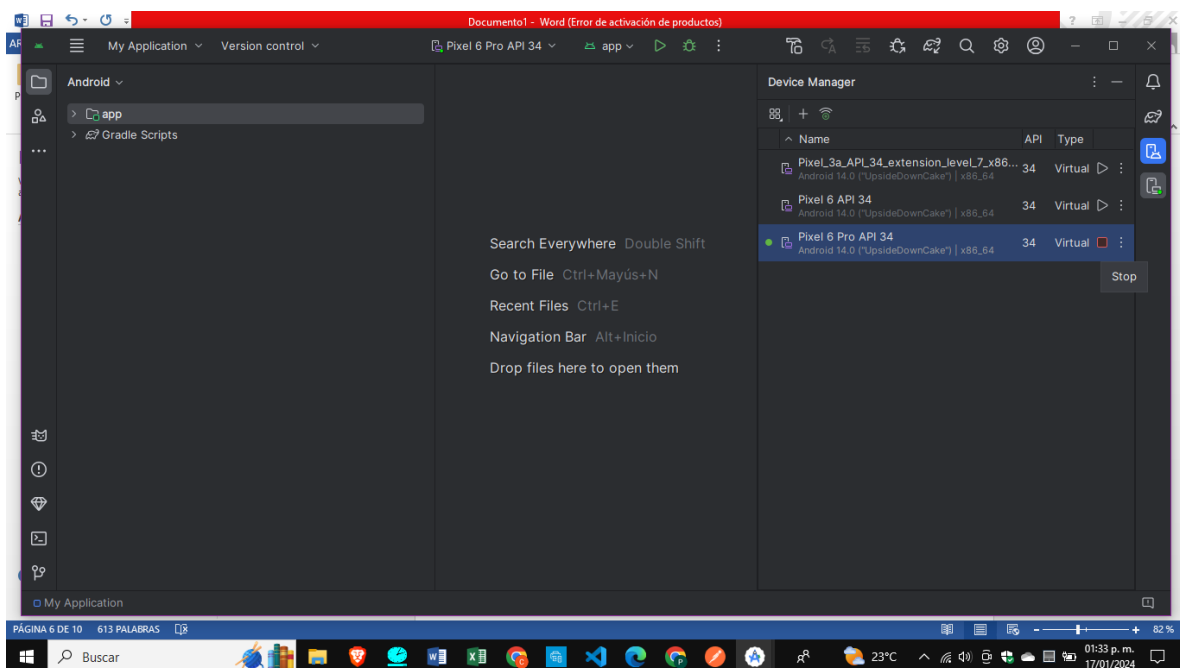
```

43
44 const styles = StyleSheet.create({
45
46   mynewscard:{
47     // borderColor: 'black',
48     borderWidth:1 ,
49     backgroundColor: 'white' ,
50     width: '95%',
51     borderRadius: 10,
52     marginVertical:10
53   },
54   newstitle:{
55     fontSize:25,
56     padding:5,
57     fontWeight: '700'
58   },
59   newsdescription:{fontSize:17, margin:10},
60   readmorecontainer:{flex:0,flexDirection:'row', justifyContent: 'flex-end'},
61   readmorebtn:{
62     backgroundColor: 'black', color: 'white',
63     fontSize:20,fontWeight: '700',margin:10,padding:10,
64     borderRadius:10
65   }
66 });

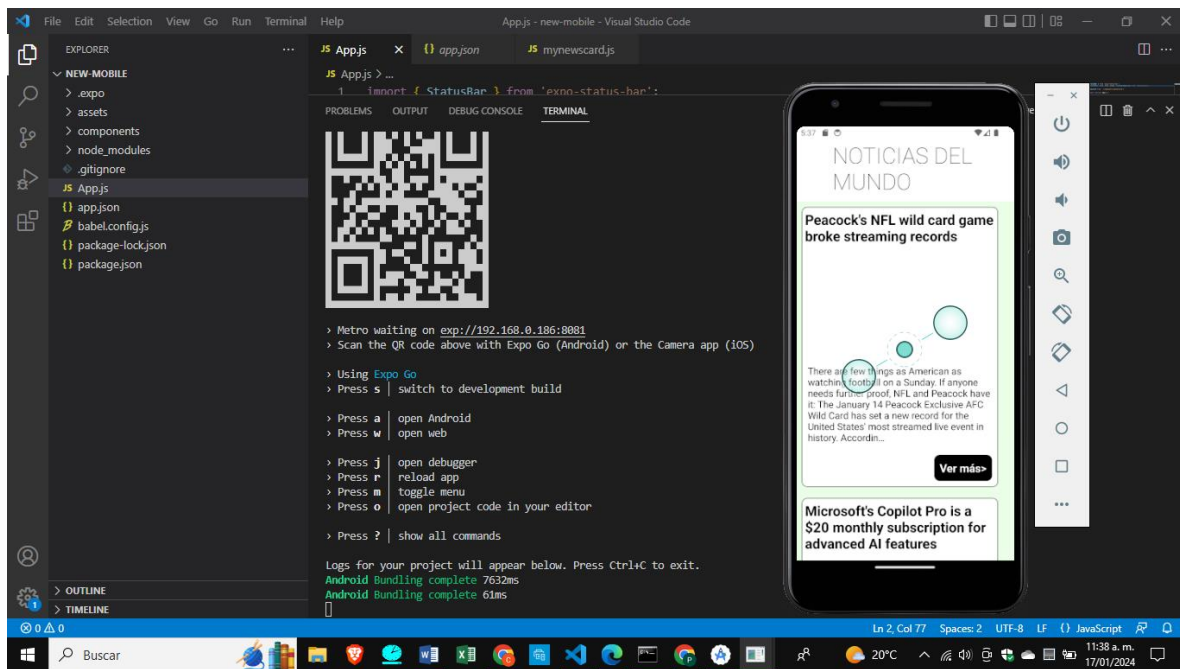
```

Estos fueron los estilos que se utilizaron para cada View y que fueron llamados dependiendo de las características y estilos que contenían para ser utilizados y adecuarse a la pantalla y mostrar una mejor presentación.

Una vez que tengamos todo procedemos a probar la aplicación, necesitaremos de un emulador para poder visualizar la aplicación en este caso nos apoyaremos de los emuladores que nos ofrece Android Studio, escogemos el emulador y procedemos a probar la app.



Probamos la aplicación en un teléfono Pixel\_3a\_API\_34\_extension\_level\_7\_x86 y nos muestra la pantalla con las noticias y el titulo de cada noticia y un botón que dice ver más a la cual nos llevara a la otra pantalla donde podemos ver más sobre la noticia



También se probó la aplicación entro otro emulador distinto esta vez fue en un teléfono Pixel 6 Pro API 34 donde igual nos muestra las noticias pero esta vez con una imagen

