[Sequelize.Op.not]: null // Uso de Sequelize.Op.not para filtrar no nulos

TextRegular: sin formato

TextSemiBold: en negrita

CardBody-> textos de dentro tienen estilo cardText

**->Estilos:**

const styles = StyleSheet.create({

  container: {

    flex: 1,

    justifyContent: 'center',  //centra verticalmente

    alignItems: 'center',  //centra horizontalmente

  },

  restaurantList: {

    flex: 1,

    width: '100%',

  },

  cardBody: {

    alignContent: 'center',

    backgroundColor: 'rgba(255,105,105,5)',

    borderRadius: 20,  //bordes redondeados

    padding: 20, //espaciado interior

    width: 200,

    height: 250,

    marginBottom: 10,  //espacio inferior

    margin: 20   //espacio alrededor de los lados

  },

  price: {

    color: 'rgba(100, 20, 0,1)',

    fontSize: 30,  //Tamaño de la fuente

    fontWeight: 'bold',  //negrita

    marginTop: 8,  //esapcio superior

  },

  cardText: {

    width: 300,

    fontSize: 18,

    fontWeight: 'bold',

    marginBottom: 2,

  },

#### **->Boton:**

#### **Estilo style**

style={({ pressed }) => [

{

backgroundColor: pressed

? GlobalStyles.brandRedTap

: GlobalStyles.brandPrimary

},

styles.actionButton

]}

* Esta función retorna un arreglo de estilos.
* Si el botón está presionado (pressed), el color de fondo es GlobalStyles.brandRedTap.
* Si no está presionado, el color de fondo es GlobalStyles.brandPrimary.
* styles.actionButton es un estilo adicional que se aplica al botón.

**Icono dentro del botón**

<View style={[{ flex: 1, flexDirection: 'row', justifyContent: 'center' }]}>

<MaterialCommunityIcons name='minus-circle' color={'white'} size={20} />

</View>

* MaterialCommunityIcons muestra un ícono de "menos" (minus-circle).
* El ícono es blanco (color={'white'}) y tiene un tamaño de 20 (size={20}).
* El View tiene estilos que centran el ícono.

**->Mas estilos**

const styles = StyleSheet.create({

// El contenedor principal de la lista ocupará todo el espacio disponible.

container: {

flex: 1

},

// Estilo para las filas individuales de la lista.

// Añade padding interno y un margen inferior.

// También establece un color de fondo usando un estilo global.

row: {

padding: 15,

marginBottom: 5,

backgroundColor: GlobalStyles.brandSecondary

},

// Contenedor para el encabezado del restaurante.

// Establece una altura fija, padding interno y un color de fondo semitransparente.

// Usa una disposición de columna y centra los elementos.

restaurantHeaderContainer: {

height: 250,

padding: 20,

backgroundColor: 'rgba(0,0,0,0.5)',

flexDirection: 'column',

alignItems: 'center'

},

// Estilo para una imagen de fondo.

// La imagen ocupa todo el espacio del contenedor, se escala para cubrir y se centra el contenido.

imageBackground: {

flex: 1,

resizeMode: 'cover',

justifyContent: 'center'

},

// Estilo para una imagen dentro de la lista de productos.

// Establece dimensiones fijas y un margen.

image: {

height: 100,

width: 100,

margin: 10

},

// Estilo para la descripción del producto.

// Cambia el color del texto a blanco.

description: {

color: 'white'

},

// Estilo para los títulos de texto.

// Aumenta el tamaño de la fuente y cambia el color a blanco.

textTitle: {

fontSize: 20,

color: 'white'

},

// Estilo para el texto que se muestra cuando la lista está vacía.

// Centra el texto y añade padding interno.

emptyList: {

textAlign: 'center',

padding: 50

},

// Estilo para los botones.

// Establece un radio de borde, altura, margen superior, padding interno y ancho.

// Centra los botones en el contenedor.

button: {

borderRadius: 8,

height: 40,

marginTop: 12,

padding: 10,

alignSelf: 'center',

flexDirection: 'row',

width: '80%'

},

// Estilo para el texto dentro de los botones.

// Cambia el tamaño y color de la fuente y centra el texto verticalmente.

text: {

fontSize: 16,

color: 'white',

alignSelf: 'center',

marginLeft: 5

},

// Estilo para el texto que muestra la disponibilidad del producto.

// Centra el texto, cambia el tamaño de la fuente, lo hace cursiva y ajusta el color a rojo.

availability: {

textAlign: 'center',

fontSize: 20,

fontStyle: 'italic',

marginRight: 70,

color: 'red'

},

// Estilo para los botones de acción (aumentar/disminuir cantidad).

// Establece un radio de borde, altura, margen superior e interno, y ancho.

// Centra el contenido del botón horizontalmente.

actionButton: {

borderRadius: 8,

height: 40,

marginTop: 5,

padding: 10,

alignSelf: 'end',

flexDirection: 'row',

width: '4%',

margin: '1%'

},

// Contenedor para los botones de acción (aumentar/disminuir cantidad).

// Dispone los elementos en fila y los centra horizontalmente.

actionButtonsContainer: {

flexDirection: 'row',

position: 'absolute',

width: '90%',

alignSelf: 'center',

marginTop: 30,

justifyContent: 'flex-end'

},

// Estilo para el borde alrededor de la cantidad del producto.

// Establece un borde sólido, margen superior, radio de borde, altura, padding interno y ancho.

// Centra el contenido horizontal y verticalmente.

quantityBorder: {

border: 'solid',

marginTop: 5,

borderRadius: 8,

height: 40,

padding: 10,

width: '4%',

margin: '1%',

justifyContent: 'space-around',

alignSelf: 'center'

}

})

  function renderDateTime (item) {

    const newDate = new Date(item)

    const day = newDate.getDate()

    const month = newDate.getMonth() + 1

    const year = newDate.getFullYear()

    const hour = newDate.getHours()

    const minutes = newDate.getMinutes()

    return `${day}/${month}/${year} ${hour}:${minutes}`

  }

**->SWITCH**

  <TextRegular> it is promoted? </TextRegular>

              <Switch

                trackColor= {{false: // Color del riel cuando está apagado GlobalStyles.brandSecondary,true:GlobalStyles.brandPrimary}}

                thumbColor= {values.promoted? GlobalStyles.brandSecondary: 'f4f3f4'}

                value= {values.promoted}

                style= {styles.switch}

                onValueChange={value => setFieldValue('promoted',value)}></Switch>

**estilo:**

  switch:{

    marginTop:5

 Si values.promoted es verdadero, el color del pulgar será GlobalStyles.brandSecondary.

 Si values.promoted es falso, el color del pulgar será 'f4f3f4'.(el circulito)

 **onValueChange**: Es una función que se ejecuta cuando cambia el valor del interruptor. Actualiza el campo promoted con el nuevo valor usando setFieldValue.

* **Dentro de un view:**

 <View style = {{flexDirection:'row',justifyContent: 'space-between',alignItems:'flex-end'}}>

 flexDirection: 'row': Organiza los elementos hijos en una fila.

 justifyContent: 'space-between': Distribuye los elementos hijos con el máximo espacio posible entre ellos.

 alignItems: 'flex-end': Alinea los elementos hijos al final del eje transversal (parte inferior en este caso).

where: { // Solution (pro)

visibleUntil: {

[Sequelize.Op.or]: [

{ [Sequelize.Op.eq]: null },

{ [Sequelize.Op.gt]: currentDate }

]

}

}

### Descripción de la condición where

* **visibleUntil:**
  + Es el campo de la base de datos que se está evaluando en esta condición.

**[Sequelize.Op.or]:**

* + Sequelize.Op.or es un operador lógico "OR" en Sequelize. Indica que la condición será verdadera si alguna de las condiciones dentro del arreglo es verdadera.

**Condiciones dentro del or:**

* + { [Sequelize.Op.eq]: null }: Esta condición verifica si el valor de visibleUntil es null. Sequelize.Op.eq es el operador de igualdad.
  + { [Sequelize.Op.gt]: currentDate }: Esta condición verifica si el valor de visibleUntil es mayor que currentDate (la fecha actual). Sequelize.Op.gt es el operador "greater than" (mayor que).