EXAMEN MEJORAS ECOSISTEMA - OCTUBRE 2023

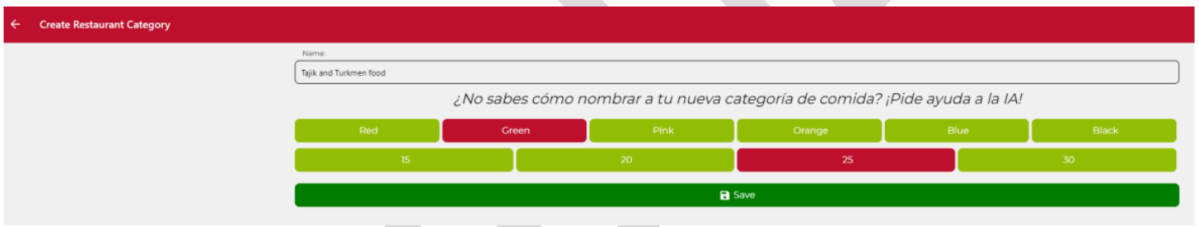
1. MEJORA UNO:
En la ventana “Create Restaurant Category”, debajo del campo para introducir el nombre de la nueva categoría a crear, aparecerá un mensaje de ayuda entre cinco posibles; esto es, deberás crear una variable que cuente con cinco mensajes de ayuda y que únicamente se muestre por pantalla uno de los cinco de manera aleatoria.
Use estos cinco segmentos como mensajes de ayuda:

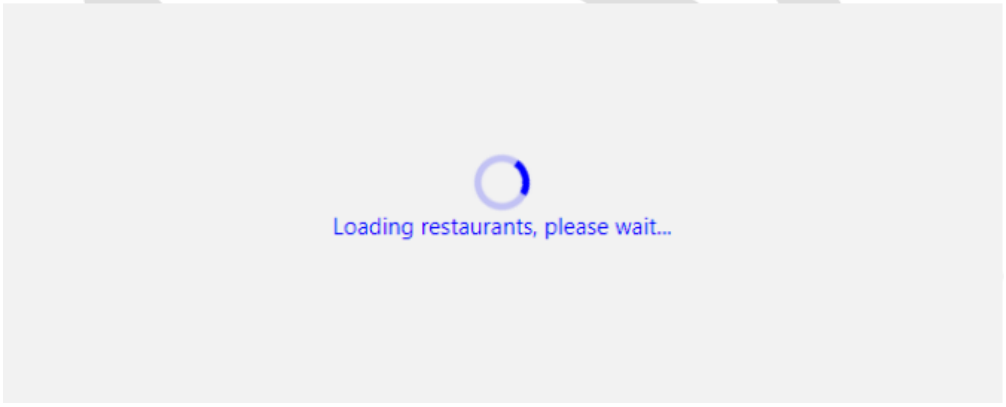
- **¡Crear una nueva categoría de comida implica un nuevo universo de sabores!**
- **¿No sabes cómo nombrar a tu nueva categoría de comida? ¡Pide ayuda a la IA!**
- **Sólo el creador sabe el secreto del sabor.**
- **¡Buenos días! Espero que te vaya todo bien en tu restaurante.**
- **Mi categoría favorita es la 'Spanish Food', ¿sabes por qué?: ¡porque España tiene la mejor dieta del mundo!**

Además, a modo de prueba, deberás colocar en dos secciones verticales contiguas dos tandas de botones de igual tamaño y estilo, tal que:

a) **Si pulsas un botón de color, el color del botón Save cambie al color indicado en el botón pulsado localizado en la sección de botones modificadores de color.**
b) **Si pulsas un botón numérico, el tamaño de fuente del mensaje de ayuda cambie al valor del tamaño de fuente indicado en el botón pulsado ubicado en la sección de botones modificadores de tamaño de fuente.**

Todo esto puede parecer algo confuso, por este motivo, se recomienda observar la primera fotografía adjunta.





BACKEND

El backend usaremos el de nuevas categorias.

FRONTEND

```
import React, { useState, useEffect } from 'react'
import { Pressable, ScrollView, StyleSheet, View } from 'react-native'
import { MaterialCommunityIcons } from '@expo/vector-icons'
import * as yup from 'yup'
import { createRestaurantCategories } from
'../../api/RestaurantEndpoints'
import InputItem from '../../components/InputItem'
import TextRegular from '../../components/TextRegular'
import * as GlobalStyles from '../../styles/GlobalStyles'
import { Formik } from 'formik'
import TextError from '../../components/TextError'

import { showMessage } from 'react-native-flash-message'

export default function CreateRestaurantCategoryScreen ({ navigation })
{
  const [backendErrors, setBackendErrors] = useState()
  // SOLUCIÓN
  const helpMessages = [
    '¡Crear una nueva categoría de comida implica un nuevo universo de
sabores!',
    '¿No sabes cómo nombrar a tu nueva categoría de comida? ¡Pide ayuda
a la IA!',
    'Sólo el creador sabe el secreto del sabor.',
    '¡Buenos días! Espero que te vaya todo bien en tu restaurante.',
    "Mi categoría favorita es la 'Spanish Food', ¿sabes por qué?:
¡porque España tiene la mejor dieta del mundo!"
  ]
  const [randomHelpMessage, setRandomHelpMessage] = useState()
  const [fontSize, setFontSize] = useState(16)
  const [backgroundColor, setBackgroundColor] =
useState(GlobalStyles.brandSuccess)
  const [selectedColor, setSelectedColor] =
useState(GlobalStyles.brandSuccess)

  const initialRestaurantCategoryValues = { name: null }
  const validationSchema = yup.object().shape({
```

```
    name: yup
      .string()
      .max(50, 'Name too long')
      .required('Name is required')

  })

  // SOLUCIÓN
  useEffect(() => {
    const randomIndex = Math.floor(Math.random() * helpMessages.length)
    setRandomHelpMessage(helpMessages[randomIndex])
  }, [])

  const createRestaurantCategory = async (values) => {
    setBackendErrors([])
    try {
      console.log(values)
      const createdRestaurantCategory = await
createRestaurantCategories(values)
      showMessage({
        message: `Restaurant Category named
${createdRestaurantCategory.name} succesfully created`,
        type: 'success',
        style: GlobalStyles.flashStyle,
        titleStyle: GlobalStyles.flashTextStyle
      })
      navigation.navigate('CreateRestaurantsScreen', { dirty: true })
    } catch (error) {
      console.log(error)
      setBackendErrors(error.errors)
    }
  }

  return (
    <Formik
      validationSchema={validationSchema}
      initialValues={initialRestaurantCategoryValues}
      onSubmit={createRestaurantCategory}>
      {({ handleSubmit }) => (
        <ScrollView>
          <View style={{ alignItems: 'center' }}>
            <View style={{ width: '60%' }}>
              <InputItem
```

```
                name='name'
                label='Name:'
              />

             {backendErrors &&
                backendErrors.map((error, index) => <TextError
key={index}>{error.param}----{error.msg}</TextError>)
              }

              {randomHelpMessage && (
                <TextRegular textStyle={[styles.helpMessage, { fontSize
}]}>
                  {randomHelpMessage}
                </TextRegular>
              )}

            <View style={styles.colorButtons}>
                <Pressable
                  onPress={() => {
                    setBackgroundColor('red')
                    setSelectedColor('red')
                  }}
                  style={({ pressed }) => [
                    styles.colorButton,
                    { backgroundColor: selectedColor === 'red' ? 'red'
: pressed ? '#333' : GlobalStyles.brandSuccess },
                    selectedColor === 'red' && styles.activeColorButton
                  ]}
                >
                  <TextRegular
textStyle={styles.text}>Red</TextRegular>
                </Pressable>
                <Pressable
                  onPress={() => {
                    setBackgroundColor('green')
                    setSelectedColor('green')
                  }}
                  style={({ pressed }) => [
                    styles.colorButton,
                    { backgroundColor: selectedColor === 'green' ?
'green' : pressed ? '#333' : GlobalStyles.brandSuccess },
                    selectedColor === 'green' &&
styles.activeColorButton
```

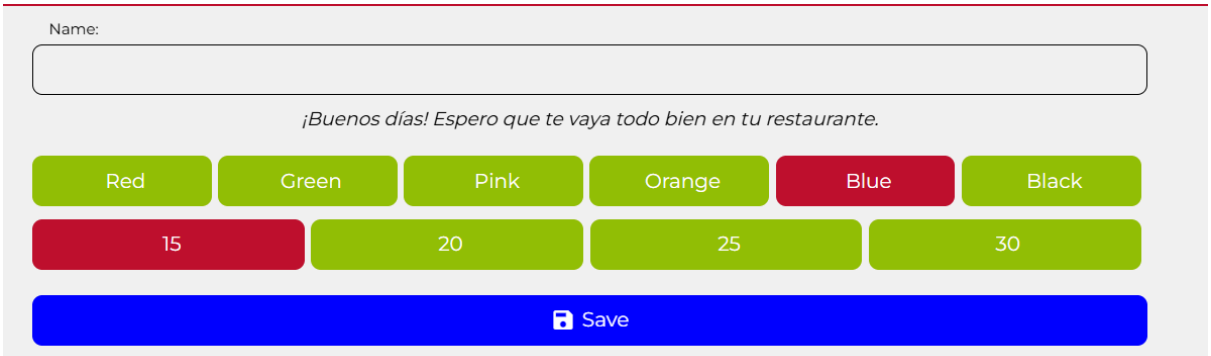
```
                ]}
              >
                <TextRegular
textStyle={styles.text}>Green</TextRegular>
              </Pressable>
              <Pressable
                onPress={() => {
                  setBackgroundColor('pink')
                  setSelectedColor('pink')
                }}
                style={({ pressed }) => [
                  styles.colorButton,
                  { backgroundColor: selectedColor === 'pink' ?
'pink' : pressed ? '#333' : GlobalStyles.brandSuccess },
                  selectedColor === 'pink' &&
styles.activeColorButton
                ]}
              >
                <TextRegular
textStyle={styles.text}>Pink</TextRegular>
              </Pressable>
              <Pressable
                onPress={() => {
                  setBackgroundColor('orange')
                  setSelectedColor('orange')
                }}
                style={({ pressed }) => [
                  styles.colorButton,
                  { backgroundColor: selectedColor === 'orange' ?
'orange' : pressed ? '#333' : GlobalStyles.brandSuccess },
                  selectedColor === 'orange' &&
styles.activeColorButton
                ]}
              >
                <TextRegular
textStyle={styles.text}>Orange</TextRegular>
              </Pressable>
              <Pressable
                onPress={() => {
                  setBackgroundColor('blue')
                  setSelectedColor('blue')
                }}
                style={({ pressed }) => [
```

```
                    styles.colorButton,
                    { backgroundColor: selectedColor === 'blue' ?
'blue' : pressed ? '#333' : GlobalStyles.brandSuccess },
                    selectedColor === 'blue' &&
styles.activeColorButton
                  ]}
                >
                  <TextRegular
textStyle={styles.text}>Blue</TextRegular>
                </Pressable>
                <Pressable
                  onPress={() => {
                    setBackgroundColor('black')
                    setSelectedColor('black')
                  }}
                  style={({ pressed }) => [
                    styles.colorButton,
                    { backgroundColor: selectedColor === 'black' ?
'black' : pressed ? '#333' : GlobalStyles.brandSuccess },
                    selectedColor === 'black' &&
styles.activeColorButton
                  ]}
                >
                  <TextRegular
textStyle={styles.text}>Black</TextRegular>
              </Pressable>

              </View>

              <View style={styles.fontSizeButtons}>
                <Pressable
                  onPress={() => setFontSize(15)}
                  style={({ pressed }) => [
                    styles.fontSizeButton,
                    { backgroundColor: pressed ?
GlobalStyles.brandSuccessTap : GlobalStyles.brandSuccess },
                    fontSize === 15 && styles.activeFontSizeButton
                  ]}
                >
                  <TextRegular textStyle={styles.text}>15</TextRegular>
                </Pressable>
                <Pressable
                  onPress={() => setFontSize(20)}
```

```
                style={({ pressed }) => [
                  styles.fontSizeButton,
                  { backgroundColor: pressed ?
GlobalStyles.brandSuccessTap : GlobalStyles.brandSuccess },
                  fontSize === 20 && styles.activeFontSizeButton
                ]}
              >
                <TextRegular textStyle={styles.text}>20</TextRegular>
              </Pressable>
              <Pressable
                onPress={() => setFontSize(25)}
                style={({ pressed }) => [
                  styles.fontSizeButton,
                  { backgroundColor: pressed ?
GlobalStyles.brandSuccessTap : GlobalStyles.brandSuccess },
                  fontSize === 25 && styles.activeFontSizeButton
                ]}
              >
                <TextRegular textStyle={styles.text}>25</TextRegular>
              </Pressable>
              <Pressable
                onPress={() => setFontSize(30)}
                style={({ pressed }) => [
                  styles.fontSizeButton,
                  { backgroundColor: pressed ?
GlobalStyles.brandSuccessTap : GlobalStyles.brandSuccess },
                  fontSize === 30 && styles.activeFontSizeButton
                ]}
              >
                <TextRegular textStyle={styles.text}>30</TextRegular>
              </Pressable>
            </View>
            <Pressable
              onPress={() => {
                handleSubmit()
                navigation.goBack()
              }}
              style={({ pressed }) => [
                {
                  backgroundColor: pressed ?
GlobalStyles.brandSuccessTap : backgroundColor
                },
                styles.button
```

```
                ]}
              >
                <View style={[{ flex: 1, flexDirection: 'row',
justifyContent: 'center' }]}>
                  <MaterialCommunityIcons name="content-save"
color={'white'} size={20} />
                  <TextRegular textStyle={styles.text}>
                    Save
                  </TextRegular>
                </View>
              </Pressable>
            </View>
          </View>
        </ScrollView>
      )}
    </Formik>
  )
}

const styles = StyleSheet.create({
  button: {
    borderRadius: 8,
    height: 40,
    padding: 10,
    width: '100%',
    marginTop: 20,
    marginBottom: 20
  },
  text: {
    fontSize: 16,
    color: 'white',
    textAlign: 'center',
    marginLeft: 5
  },
  helpMessage: {
    fontSize: 14,
    textAlign: 'center',
    marginTop: 10,
    marginBottom: 10,
    fontStyle: 'italic'
  },
  colorButtons: {
    flexDirection: 'row',
```

```
    justifyContent: 'space-between',
    marginTop: 10
  },
  colorButton: {
    flex: 1,
    borderRadius: 8,
    height: 40,
    padding: 10,
    marginRight: 5
  },
  fontSizeButtons: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    marginTop: 10
  },
  fontSizeButton: {
    flex: 1,
    borderRadius: 8,
    height: 40,
    padding: 10,
    marginRight: 5
  },
  activeFontSizeButton: {
    backgroundColor: GlobalStyles.brandPrimary
  },
  activeColorButton: {
    backgroundColor: GlobalStyles.brandPrimary
  }
})
```



Ahora lo de cargar:

```
// SOLUCIÓN
  const [isLoading, setIsLoading] = useState(true)

  /* useEffect(() => {
    if (loggedInUser) {
      fetchRestaurants()
    } else {
      setRestaurants(null)
    }
  }, [loggedInUser, route]) */

  useEffect(() => {
    console.log('Loading restaurants, please wait 5 seconds')
    setTimeout(() => {
      if (loggedInUser) {
        fetchRestaurants()
      } else {
        setRestaurants([])
      }
      setIsLoading(false)
      console.log('Restaurants loaded')
    }, 650)
  }, [loggedInUser, route])
```

Y:

```
return (
    <>
    {isLoading
      ? (
        <View style={styles.loadingContainer}>
          <ActivityIndicator size="large" color = {GlobalStyles.blue}
/>
          <TextRegular style = {{ color: GlobalStyles.blue, fontSize:
15 }}>Loading restaurants, please wait...</TextRegular>
        </View>
        )
      : (
        <>
    <FlatList
      style={styles.container}
      data={restaurants}
      renderItem={renderRestaurant}
```

```
      keyExtractor={item => item.id.toString()}
      ListHeaderComponent={renderHeader}
      ListEmptyComponent={renderEmptyRestaurantsList}
    />
    <DeleteModal
      isVisible={restaurantToBeDeleted !== null}
      onCancel={() => setRestaurantToBeDeleted(null)}
      onConfirm={() => removeRestaurant(restaurantToBeDeleted)}>
        <TextRegular>The products of this restaurant will be deleted as
well</TextRegular>
        <TextRegular>If the restaurant has orders, it cannot be
deleted.</TextRegular>
    </DeleteModal>
    </>
        )}
    </>
  )
}
```