EXAMEN 22-23 TARDE

Enunciado - Creación de categorías de restaurantes personalizadas

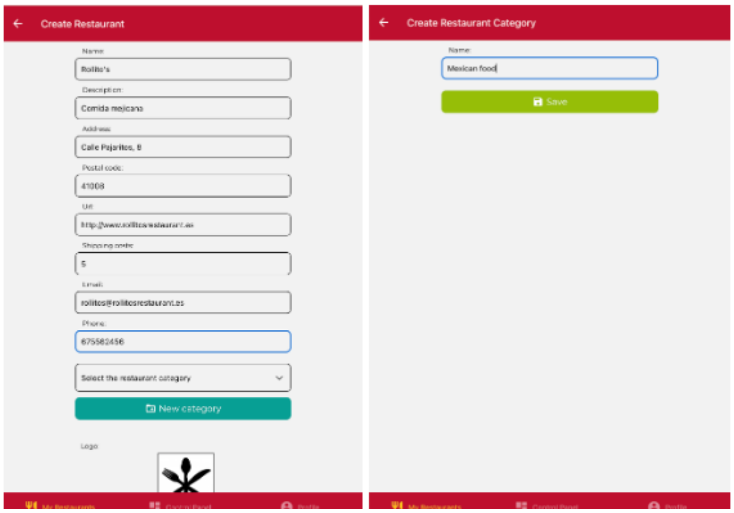
Realice las modificaciones que considere necesarias, tanto en backend como en frontend, para satisfacer los nuevos requisitos que a continuación se describen.

Se desea permitir a los dueños de restaurantes crear sus propias categorías de restaurantes. Para ello, en la pantalla de creación de restaurantes se incluirá un botón para acceder a una nueva pantalla que permitirá introducir un nuevo nombre de categoría de restaurante (ver capturas). Puede usar este icono para el botón:

```
<MaterialCommunityIcons name='folder-plus-outline' color={'white'} size={20} />
```

Al volver a la pantalla de creación de restaurantes tras introducir la nueva categoría, dicha categoría debe estar disponible en la lista desplegable de categorías de restaurantes.

No se debe permitir la creación de una categoría que ya existiera. En dicho caso, el Backend debe responder con un error que será visualizado en la pantalla de creación de categorías de restaurantes al pulsar el botón de submit. Además, el tamaño máximo para los nombres de las categorías de restaurante será de 50 caracteres. Esta restricción debe comprobarse tanto a nivel de formulario en el Frontend como a nivel de Backend.



Para la realización de este examen , tenemos que modificar el restaurantCategories Controller ya que hay que añadir la función de crear desde la app

BACKEND

Controller RESTAURANTCATEGORYCONTROLLER

```
exports.create = async function (req, res) {
  try {
    const newCategory = RestaurantCategory.build(req.body)
    const category = await newCategory.save()
    res.json(category)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Al crear esta función estamos permitiendo que a través de la aplicación se pueda crear un restaurant Category

Const newCategory = RestaurantCategory.build(req.body)→Build crea una instancia de modelo , req.body contiene los datos enviados en la solicitud HTTP para crear la nueva categoría

Const category = await newCategory.save() → Guarda la nueva categoria

Res.json(category) → Envia la respuesta HTTP con código de éxito

```
const models = require('../models')
const RestaurantCategory = models.RestaurantCategory
```

Importar

Validation RESTAURANT-CATEGORY-VALIDATION

Debemos crear un validation para la hora de crear una nueva categoría ,compruebe que no haya otra que se llame igual que ella

```
const { check } = require('express-validator')
const { RestaurantCategory } = require('../../models')

const checkCategoryExists = async (value, { req }) => {
  try {
    const restaurantCategory = await RestaurantCategory.findOne({
      where: { name: value }
    })
    if (restaurantCategory === null) {
      return Promise.resolve()
    } else {
      return Promise.reject(new Error('The category ' + value + ' already exists.'))
    }
  } catch (err) {
    return Promise.reject(new Error(err))
  }
}

module.exports = {
  create: [
    check('name').exists().isString().isLength({ min: 1, max: 50 }).trim().custom(checkCategoryExists)
  ],
  update: [
    check('name').exists().isString().isLength({ min: 1, max: 50 }).trim().custom(checkCategoryExists)

  ]
}
```

Error: En este caso al poner en el const models y solo poner ../models , me daba error , lo solucione poniendo ../../models

Error2: En findOne({.. → No he puesto el where y me ha crasheado

El checkCategoryExists comprueba que no haya otra categoría que tenga el mismo nombre

Creamos una constante que se llama categoría , que encuentra una categoría , con el name : value , es decir el mismo nombre que el valor que se mete en la petición

Si existe esa categoría con ese nombre , devuelve un error , sino no devuelve nada

Para añadirlo al créate y al update , añadimos al final , después del trim un

.custom(checkCategoryExists)

Tambien añadimos que el max de caracteres que se puedan poner sean de 50 , según lo dice el enunciado

Rutas RESTAURANT-CATEGORY-ROUTES

Añadimos a la ruta la función post , que lo que hace , es crear la categoría

```
app.route('/restaurantCategories')
  .get(RestaurantCategoryController.indexRestaurantCategory)
  .post(
    middlewares.isLoggedIn,
    middlewares.hasRole('owner'),
    RestaurantCategoryValidation.create,
    middlewares.handleValidation,
    RestaurantCategoryController.create
  )
```

Importamos RestaurantCategoryValidation , que llama al exports créate , que contiene el checker de que no puede haber 2 categorias de mismo nombre

FRONTED

EndPoint RESTAURANT-ENDPOINTS

```
function createResturantsCategory (data) {
  return post('restaurantCategories', data)
}

export { getAll, getDetail, getRestaurantCategories, create, update, remove, createResturantsCategory }
```

Añadimos el EndPoint

data → Es un objeto que contiene la informacion necesaria para crear la categoria

post('restaurantCategories',data) → Realiza una solicitud de post (subir, crear) , post se utiliza para enviar datos a un servidor para crear un nuevo recurso en una API

SCREENS

CREATERESTAURANTSCREEN

Debemos crear , una nueva pantalla a donde dirigirnos al pulsar el boton Create RestaurantCategory , por tanto en el boton debemos poner un :

navigation.navigate(‘Nombre de la pantalla a la que queremos dirigirnos’)

Lo demas son caracteristicas visuales del boton

```
  style={{ backgroundColor: GlobalStyles.brandBackground }}
  dropDownStyle={{ backgroundColor: '#fafafa' }}
/>
<ErrorMessage name={'restaurantCategoryId'} render={msg => <TextError>{msg}</TextError> }/>
<Pressable
  onPress={() => navigation.navigate('CreateRestaurantCategoryScreen')
  }
  style={({ pressed }) => [
    {
      backgroundColor: pressed
        ? GlobalStyles.brandGreenTap
        : GlobalStyles.brandGreen
    },
    styles.button
  ]}>
    <View style={[{ flex: 1, flexDirection: 'row', justifyContent: 'center' }]}>
    <MaterialCommunityIcons name='folder-plus-outline' color={'white'} size={20} />
    <TextRegular textStyle={styles.text}>
     Create restaurantCategory
    </TextRegular>
  </View>
</Pressable>
<Pressable onPress={() =>
  pickImage(
    async result => {
      await setFieldValue('logo', result)
    }
```

Mostrar este icono 

<MaterialCommunityIcons name='folder-plus-outline' color={'white'} size={20} />

STACK

A la hora de crear una nueva pantalla , es muy importante añadirla al Stack

```
        title: 'Create Restaurant'
      }} />
      <Stack.Screen
        name='CreateRestaurantCategoryScreen'
        component={CreateRestaurantCategoryScreen}
        options={{
          title: 'Create restaurant category'
        }} />
      <Stack.Screen
        name='CreateProductScreen'
```

```
import RestaurantDetailScreen from './RestaurantDetailScreen'
import RestaurantsScreen from './RestaurantsScreen'
import CreateRestaurantCategoryScreen from './CreateRestaurantCategoryScreen'

const Stack = createNativeStackNavigator()
```

CREATE-RESTAURANT-CATEGORY-SCREEN

He copiado , la pantalla createProduct , que se parecía a la que queríamos obtener , y he quitado cosas(como propiedades que no necesitábamos , ya que categoría solo necesita añadir el nombre) y modificado otras (como poner en todos las CreateRestaurantCategoryScreen)

CreateRestaurantCategoryScreen.js | RestaurantsStack.js | CreateRestaurantScreen.js

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

export default function CreateRestaurantCategoryScreen ({ navigation }) {
  const [backendErrors, setBackendErrors] = useState()

  const initialRestaurantValues = { name: null }
  const validationSchema = yup.object().shape({
    name: yup
      .string()
      .max(50, 'Name too long')
      .required('Name is required')
  })

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

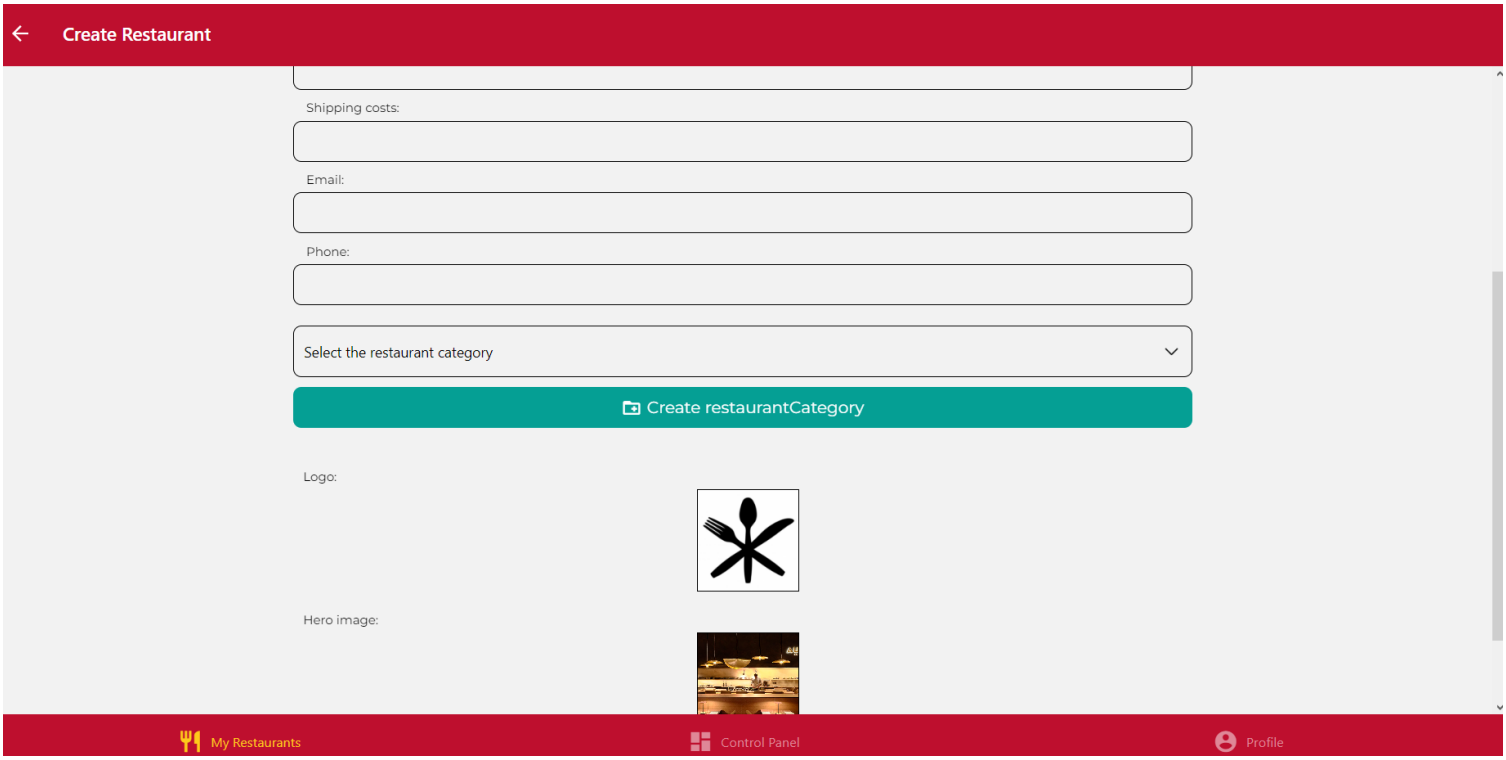
```
}
return (
  <Formik
    validationSchema={validationSchema}
    initialValues={initialRestaurantValues}
    onSubmit={createRestaurantCategories}>
    {({ handleSubmit, setFieldValue, values }) => (
      <ScrollView>
        <View style={{ alignItems: 'center' }}>
          <View style={{ width: '60%' }}>
            <InputItem
              name='name'
              label='Name:'
            />
            {backendErrors &&
              backendErrors.map((error, index) => <TextError key={index}>{error.param}-{error.msg}</TextError>)
            }

            <Pressable
              onPress={handleSubmit}
              style={({ pressed }) => [
                {
                  backgroundColor: pressed
                    ? GlobalStyles.brandSuccessTap
                    : GlobalStyles.brandSuccess
                },
                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
  },
  imagePicker: {
    height: 40,
    paddingLeft: 10,
    marginTop: 20,
    marginBottom: 80
  },
  image: {
    width: 100,
    height: 100,
    borderWidth: 1,
    alignSelf: 'center',
    marginTop: 5
  }
})
```

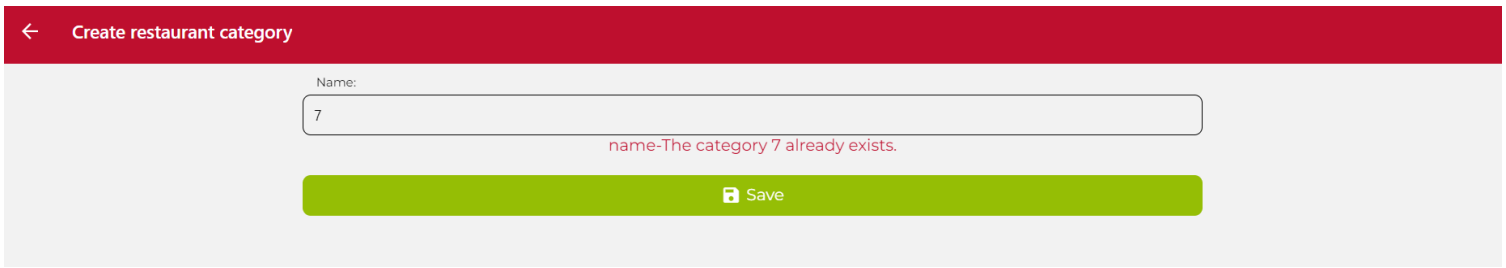
Error: Para que al crear una nueva categoria se almacene y se muestre directamente , sin necesidad de recargar la pagina, en CREATERESTAURANTSCREEN

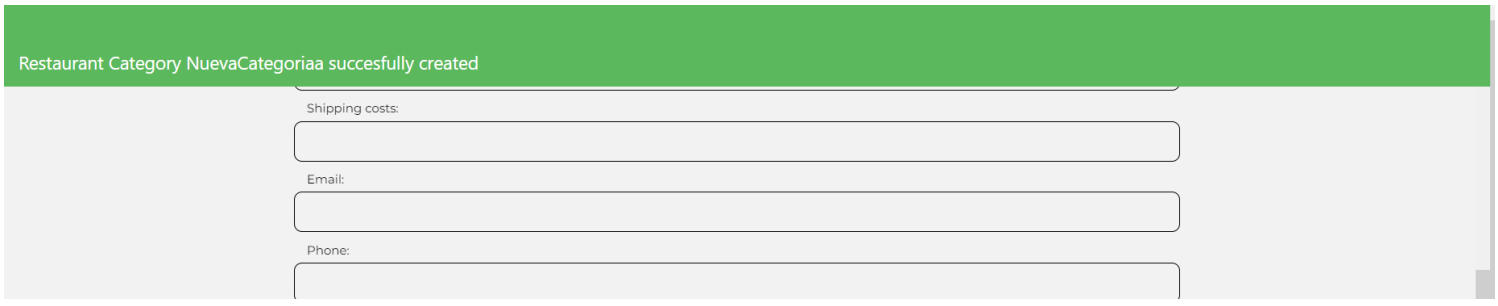
```
useEffect(() => {
  async function fetchRestaurantCategories () {
    try {
      const fetchedRestaurantCategories = await getRestaurantCategories()
      const fetchedRestaurantCategoriesReshaped = fetchedRestaurantCategories.map((e) => {
        return {
          label: e.name,
          value: e.id
        }
      })
      setRestaurantCategories(fetchedRestaurantCategoriesReshaped)
    } catch (error) {
      showMessage({
        message: `There was an error while retrieving restaurant categories. ${error} `,
        type: 'error',
        style: GlobalStyles.flashStyle,
        titleStyle: GlobalStyles.flashTextStyle
      })
    }
  }
  fetchRestaurantCategories()
}, [route])
```

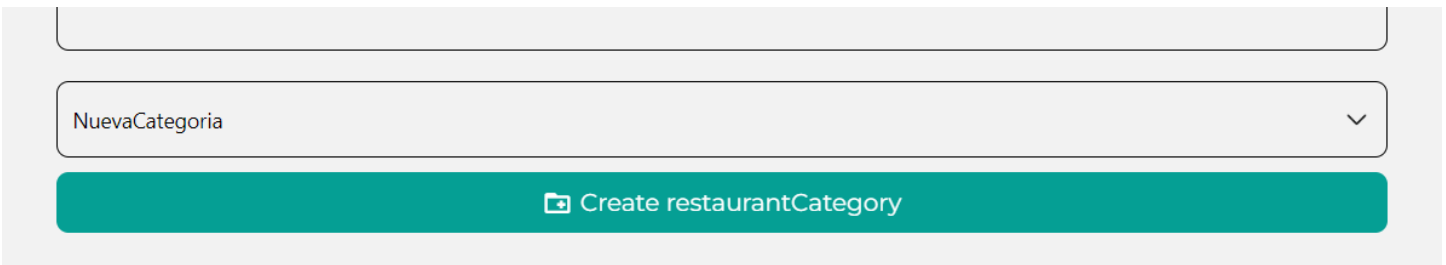
Añadir el route abajo , dentro de los corchetes

RESULTADO


Create Restaurant
Shipping costs:
Email:
Phone:
Select the restaurant category
Create restaurantCategory
Logo:
Hero image:
My Restaurants
Control Panel
Profile


Create restaurant category
Name:
7
name-The category 7 already exists.
Save


Restaurant Category NuevaCategoriaa succesfully created
Shipping costs:
Email:
Phone:


NuevaCategoria
Create restaurantCategory