

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.

The image displays two side-by-side screenshots of a mobile application interface. The left screenshot shows the 'Create Restaurant' screen, which features a list of input fields: Name (filled with 'Rollito's'), Description (filled with 'Comida mexicana'), Address (filled with 'Calle Pajaritos, 8'), Postal code (filled with '41008'), URL (filled with 'http://www.rollitosrestaurant.es'), Shipping costs (filled with '5'), Email (filled with 'rollitos@rollitosrestaurant.es'), and Phone (filled with '675582456'). There is also a dropdown menu labeled 'Select the restaurant category' and a green button labeled 'New category'. The right screenshot shows the 'Create Restaurant Category' screen, which has a single input field for 'Name' containing the text 'Mexican food' and a green button labeled 'Save'.

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) → Envía 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 `create` 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(
    middleware.isLoggedIn,
    middleware.hasRole('owner'),
    RestaurantCategoryValidation.create,
    middleware.handleValidation,
    RestaurantCategoryController.create
  )
```

Importamos `RestaurantCategoryValidation` , que llama al `exports.create` , 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 características visuales del boton

```
...containerStyle={{ height: 40, margin: 10 }}
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'
  component={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)

```
1  import React, { useState } from 'react'
2  import { Pressable, ScrollView, StyleSheet, View } from 'react-native'
3  import { MaterialCommunityIcons } from '@expo/vector-icons'
4  import * as yup from 'yup'
5  import { createRestaurantsCategory } from '../../api/RestaurantEndpoints'
6  import InputItem from '../../components/InputItem'
7  import TextRegular from '../../components/TextRegular'
8  import * as GlobalStyles from '../../styles/GlobalStyles'
9  import { showMessage } from 'react-native-flash-message'
10 import { Formik } from 'formik'
11 import TextError from '../../components/TextError'
12
13 export default function CreateRestaurantCategoryScreen ({ navigation }) {
14   const [backendErrors, setBackendErrors] = useState()
15
16   const initialRestaurantValues = { name: null }
17   const validationSchema = yup.object().shape({
18     name: yup
19       .string()
20       .max(50, 'Name too long')
21       .required('Name is required')
22   })
23
24   const createRestaurantCategories = async (values) => {
25     setBackendErrors([])
26     try {
27       const createdRestaurant = await createResturantsCategory(values)
28       showMessage({
29         message: `Restaurant Category ${createdRestaurant.name} succesfully created`,
30         type: 'success',
31         style: GlobalStyles.flashStyle,
32         titleStyle: GlobalStyles.flashTextStyle
33       })
34       navigation.navigate('CreateRestaurantScreen', { dirty: true })
35     } catch (error) {
36       console.log(error)
37       setBackendErrors(error.errors)
38     }
39   }
```

```

}
return (
  <Formik
    validationSchema={validationSchema}
    initialValues={initialRestaurantValues}
    onSubmit={createRestaurantCategories}>
    {{{ handleSubmit, setFieldValue, values }} => (
      <ScrollView>
        <View style={{ alignItems: 'center' }}>
          <View style={{ width: '60%' }}>
            <FormItem
              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>
        </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 CREATERESTAURANTSSCREEN

```

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 NuevaCategoríaa succesfully created

Shipping costs:

Email:

Phone:

NuevaCategoría

▼

Create restaurantCategory

