

EXAMEN 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.

← Create Restaurant

Name:

Rollito's

Description:

Comida mejicana

Address:

Calle Pajaritos, 8

Postal code:

41008

Url:

http://www.rollitosrestaurant.es

Shipping costs:

5

Email:

rollitos@rollitosrestaurant.es

Phone:


675562456

Select the restaurant category

▼

New category

Logo:



My Restaurants

Control Panel

Profile

← Create Restaurant Category

Name:

Mexican food

Save

My Restaurants

Control Panel

Profile

← Create Restaurant

Name:

Rollito's

Description:

Comida mejicana

Address:

Calle Pajaritos, 8

Postal code:

41008

Url:

http://www.rollitosrestaurant.es

Shipping costs:

5

Email:

rollitos@rollitosrestaurant.es

Phone:

675562456


Select the restaurant category

^

Spanish

Fast food

Mexican food



My Restaurants

Control Panel

Profile

← Create Restaurant Category

Name:

Fast food

name-The category Fast food already exists.

Save

My Restaurants

Control Panel

Profile

BACKEND

¿ Sobre qué modelo vamos a trabajar ?

- Estamos trabajando sobre el modelo Restaurant Category, el cual ya tiene definida la propiedad name, luego no tendremos que añadir una nueva propiedad al modelo.
- El enunciado dice:
 - ***“No se debe permitir la creación de una categoría que ya existiera”***

Luego, tenemos que añadir que la propiedad name sea única, y así no se pueda repetir, luego añadimos en el modelo de Restaurant Category:

```
name: {  
  type: DataTypes.STRING,  
  allowNull: false,  
  unique: true,  
  len: [1, 50]  
},
```

Y en el migration:

```
name: {  
  type: DataTypes.STRING,  
  allowNull: false,  
  unique: true,  
  len: [1, 50]  
},
```

Una vez hecho esto, añadimos las validaciones del backend para crear una nueva categoría:

- (1) *“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.”*
- (2) ***“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 (1):

Tenemos que validar que no exista la nueva categoría, para ello, tendremos que crear el nuevo fichero llamado **RestaurantCategoryValidation** en la carpeta de **DeliverUS-Backend\src\controllers\validation**.

En este fichero, tenemos que crear una nueva función que verifique:
categoríaACrear existe ? se puede crear : no se puede crear

Luego, la nueva función que se va a crear es:

```
// SOLUCION
const checkCategoryExists = async (value, { req }) => {
  try {
    const category = RestaurantCategory.findOne({ where: { name: value } })
    if (!category) {
      return Promise.resolve('Category successfully created')
    } else {
      return Promise.reject(new Error('Restaurant category successfully created'))
    }
  } catch (err) {
    return Promise.reject(new Error(err))
  }
}
```

Para (2) añadimos la restricción del String, y añadimos la nueva función con el custom:

```
const create = [
  check('name').custom(checkCategoryExists),
  // El tamaño máximo para los nombres de las categorías de restaurante
  // será de 50 caracteres
  // SOLUCION
  check('name').exists().isString().isLength({ min: 1, max: 50 })
  .trim()
]
export { create }
```

Y por último crear la función de create en el controlador del modelo, hay más funciones de create en otros modelos que se pueden copiar y pegar y cambiar el .build al modelo que estamos creando una nueva entidad:

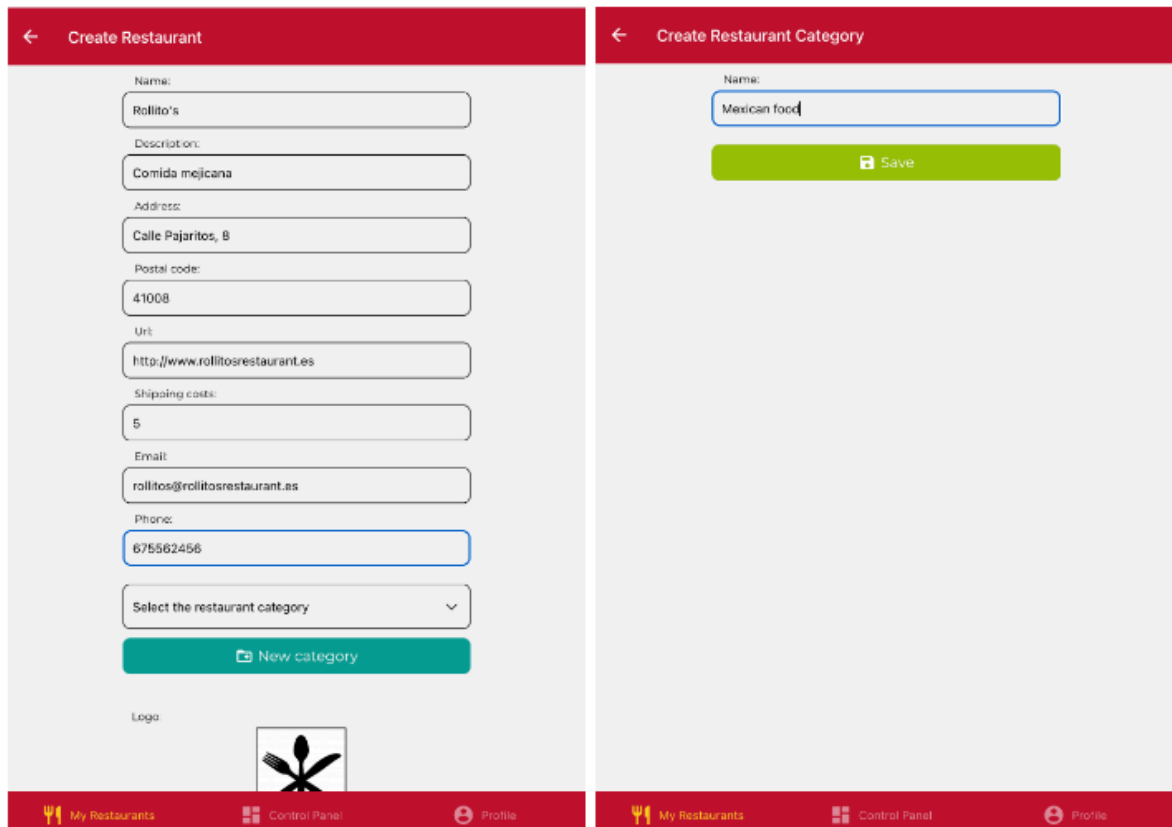
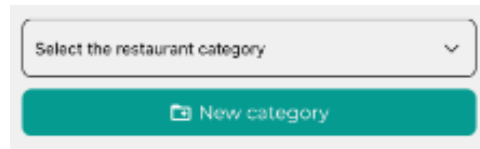
```
// SOLUCION
// Función para crear una nueva categoría
const create = async function (req, res) {
  const newCategory = RestaurantCategory.build(req.body)
  try {
    const restaurantCategories = await newCategory.save()
    res.json(restaurantCategories)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Y por último creamos la ruta:

```
const loadFileRoutes = function (app) {  
  app.route('/restaurantCategories')  
    .get(RestaurantCategoryController.index)  
    .post(  
      isLoggedIn,  
      hasRole('owner'),  
      RestaurantCategoryValidation.create,  
      handleValidation,  
      RestaurantCategoryController.create)  
}
```

FRONTEND

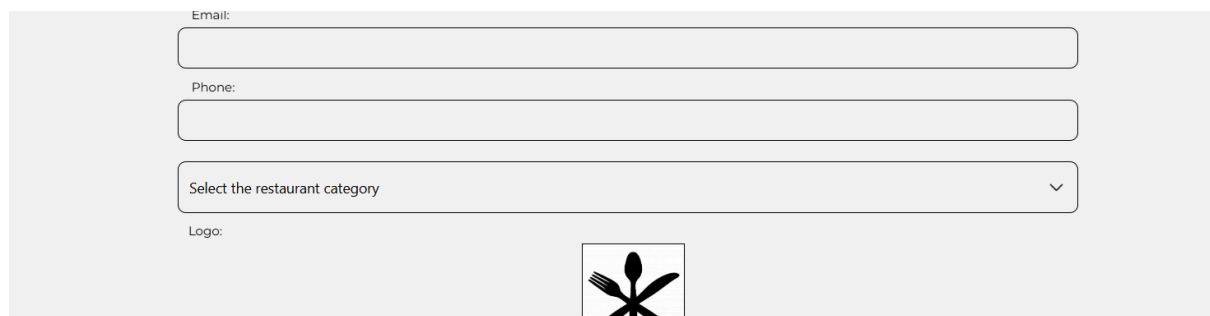
Tenemos que añadir el botón de crear una nueva categoría y que nos dirija a la pantalla de Create Restaurant Category:



Para añadir el botón nos dan que podemos usar:

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

Tenemos:



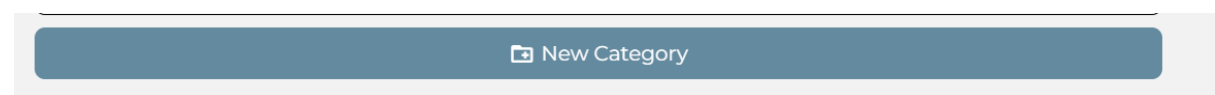
Para añadir este botón, primero tenemos que añadir el endpoint de Restaurant para que podamos crear las nuevas categorías:

```
// SOLUCION
function createRestaurantCategories (data) {
  return post('restaurantCategories', data)
}
```

Ahora en el createRestaurantScreen añadimos el botón que navega a CreateRestaurantCategory:

```
{/* SOLUCION */}
    <Pressable
      onPress={() =>
navigation.navigate('CreateRestaurantCategoryScreen')}
      style={({ pressed }) => [
        {
          backgroundColor: pressed
            ? GlobalStyles.brandBlueTap
            : GlobalStyles.brandBlue
        },
        styles.button
      ]}>
      <View style={[{ flex: 1, flexDirection: 'row',
justifyContent: 'center' }]}>
        <MaterialCommunityIcons name='folder-plus-outline'
color={'white'} size={20} />
        <TextRegular textStyle={styles.text}>
          New Category
        </TextRegular>
      </View>
    </Pressable>
  {/* SOLUCION */}
```

Tenemos esto:



La lógica para que se vaya a la otra pantalla es:

```
onPress={() => navigation.navigate('CreateRestaurantCategoryScreen')}
```

Ahora tenemos que crear la nueva pantalla `CreateRestaurantCategoryScreen` para que permita crear las nuevas categorías:

Copiar la clase de `createRestaurant` y cambiar a:

```
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 { 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()

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

  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
      })
    } catch (error) {
      setBackendErrors(error.message)
    }
  }

  return (
    <View style={GlobalStyles.container} >
      <TextRegular style={GlobalStyles.heading} >Create Restaurant Category</TextRegular>
      <Formik
        initialValues={initialRestaurantCategoryValues}
        validationSchema={validationSchema}
        onSubmit={createRestaurantCategory}
      >
        <InputItem
          label="Name"
          placeholder="Name"
          style={GlobalStyles.input}
        />
        <TextError
          style={GlobalStyles.errorText}
          message={backendErrors}
        />
        <Pressable
          style={GlobalStyles.button}
          onPress={handleSubmit}
        >
          Create Restaurant Category
        </Pressable>
      </Formik>
    </View>
  )
}
```



```

    ))
    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>)
            }

            <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>
)

```

```

        </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
  }
})

```

Y por último tenemos que añadir la nueva navegación en Restaurantes Stack:

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