EXAMEN ORDENACIÓN DE PRODUCTOS POR PRECIO - JUNIO 2023

Enunciado - Ordenación de productos por precio
Realice las modificaciones que considere necesarias, tanto en backend como en frontend, para satisfacer los nuevos requisitos que a continuación se describen.

Se desea ofrecer a los propietarios que los productos de sus restaurantes aparezcan ordenados según el campo order de la entidad Producto o según el campo price del producto, y que puedan determinar cual será el orden predeterminado en cada restaurante, de manera que cuando se listen los productos aparezcan siempre según el orden que haya decidido.

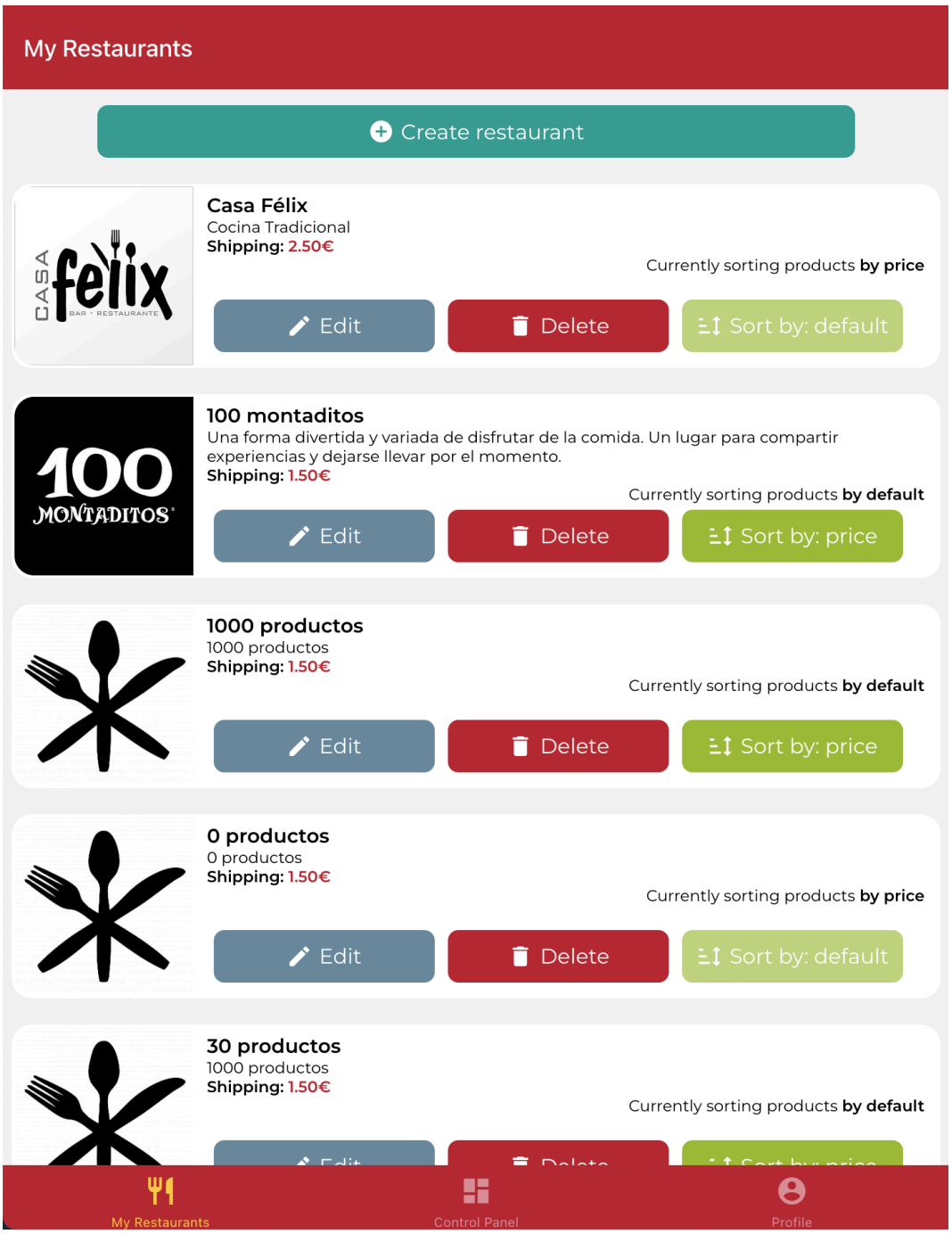
Recuerde que actualmente los productos se muestran en la pantalla de detalle del restaurante y el backend los devuelve siempre ordenados según el campo order. Por defecto, cada restaurante ordenará sus productos según el mencionado campo order.

Implemente los cambios necesarios en Backend y Frontend para incluir dicha funcionalidad. Se espera que el Frontend siga un diseño como el que se muestra en las siguientes capturas.

Nótese que no se pide modificación de las pantallas de creación o edición de restaurante.

NOTA: Puede usar el siguiente icono para el nuevo botón:

<MaterialCommunityIcons name='sort' color={'white'} size={20} />
y los siguientes colores de fondo: brandSuccess o brandSuccessDisabled


My Restaurants
Create restaurant
Casa Félix
Cocina Tradicional
Shipping: 2.50€
Currently sorting products by price
Edit
Delete
Sort by: default
100 montaditos
Una forma divertida y variada de disfrutar de la comida. Un lugar para compartir experiencias y dejarse llevar por el momento.
Shipping: 1.50€
Currently sorting products by default
Edit
Delete
Sort by: price
1000 productos
1000 productos
Shipping: 1.50€
Currently sorting products by default
Edit
Delete
Sort by: price
0 productos
0 productos
Shipping: 1.50€
Currently sorting products by price
Edit
Delete
Sort by: default
30 productos
1000 productos
Shipping: 1.50€
Currently sorting products by default
My Restaurants
Control Panel
Profile

BACKEND

Añadir nueva propiedad

Tenemos que añadir una nueva propiedad al modelo de restaurant, que será un booleano que tomará valor true si ordenamos por precio o false si ordenamos por el valor por defecto que es order.

Añadimos la propiedad al modelo de Restaurant:

```
orderByPrice:{
      type: DataTypes.BOOLEAN,
      defaultValue: false
    },
```

Ponemos que el valor por defecto sea false, osea, que esten ordenados por 'order'.

Y en el create Restaurant también la añadimos:

```
 orderByPrice: {
        type: Sequelize.BOOLEAN,
        defaultValue: false
      },
```

No tenemos que validar nada, pues no indican nada.

Tenemos que editar la función que muestra los restaurantes para que los ordene según el order o el price:

```
const show = async function (req, res) {
  // Only returns PUBLIC information of restaurants
  try {
    let restaurant = await Restaurant.findByPk(req.params.restaurantId)
      const orderingBy = restaurant.orderByPrice ?
        [[{ model: Product, as: 'products' }, 'price', 'ASC']] :
        [[{ model: Product, as: 'products' }, 'order', 'ASC']]

        restaurant = await Restaurant.findByPk(req.params.restaurantId,
{
      attributes: { exclude: ['userId'] },
      include: [{
        model: Product,
        as: 'products',
        include: { model: ProductCategory, as: 'productCategory' }
      },
      {
        model: RestaurantCategory,
        as: 'restaurantCategory'
      }],
      order: orderingBy
    }
    )
    res.json(restaurant)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Y por último tenemos que crear la función que ordena los productos para el frontend:

```
const orderingBy = async function (req, res) {
  try {
    let restaurant = await Restaurant.findByPk(req.params.restaurantId)
    if(!restaurant.orderByPrice){
      restaurant.orderByPrice = true
    }else{
      restaurant.orderByPrice = false
    }
    const orderRestaurant = await restaurant.save()
    res.json(orderRestaurant)
  }
  catch (err) {
    res.status(500).send(err)
  }
}
```

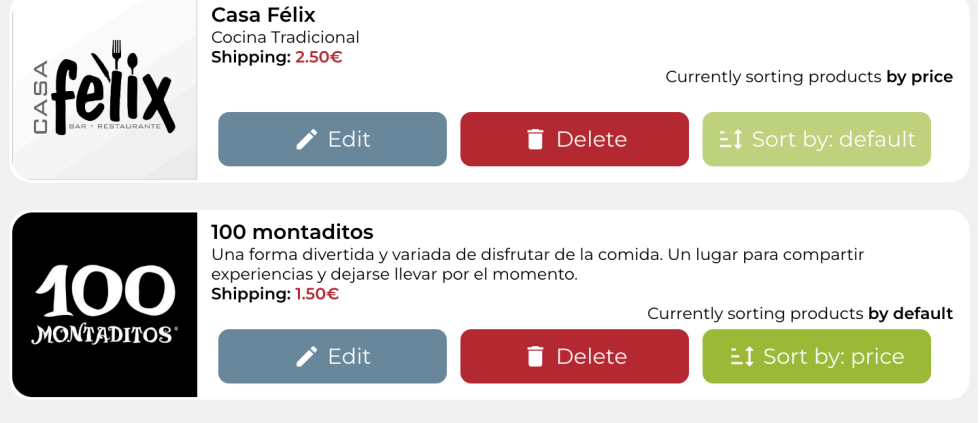
O mas facil:

```
const orderingBy = async function (req, res) {
  try {
    const restaurant = await Restaurant.findByPk(req.params.restaurantId)
    restaurant.sortByPrice = !restaurant.sortByPrice // Si el valor actual de "sortByPrice" es "Falso", pasará a "True"
    // y viceversa cada vez que se pulse sobre el botón, de manera que se ordenen los productos del restaurante de una manera u otra.
    await restaurant.save()
    res.json(restaurant)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Y por último añadimos la ruta, que será una actualización parcial (patch):

```
  app.route('/restaurants/:restaurantId/toggleProductsSorting')
      .patch(
        isLoggedIn,
        hasRole('owner'),
        checkEntityExists(Restaurant, 'restaurantId'),
        RestaurantMiddleware.checkRestaurantOwnership,
        RestaurantController.orderingBy)
```

FRONTEND



Tenemos que añadir ese botón, que funcionara según la función orderBy que hemos creado en el backend, primero añadimos el endpoint:

```
function orderBy (id) {
  return patch(`restaurants/${id}/toggleProductsSorting`)
}
```

Y ahora editamos el Restaurant Screen, primero añadimos la función que recibe la función del endpoint que hemos añadido:

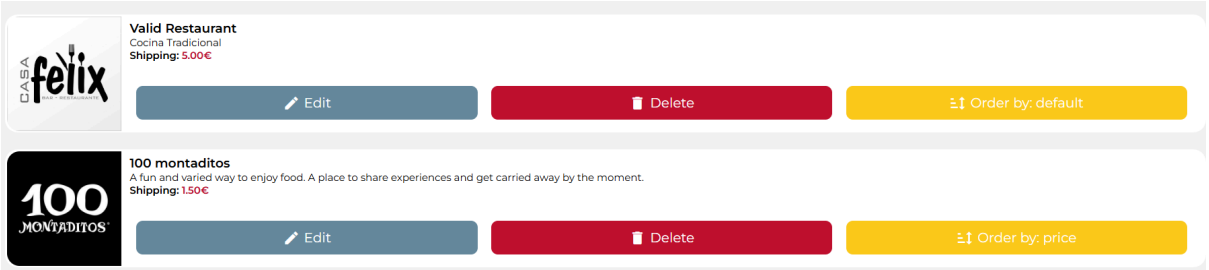
```
const orderRestaurant = async (restaurant) => {
    try {
      const modifiedRestaurant = await orderBy(restaurant.id)
      if (modifiedRestaurant) {
        await fetchRestaurants()
        showMessage({
          message: `Restaurant ${restaurant.name} succesfully ordered
by ${restaurant.orderByPrice ? 'price' : 'default'}`,
          type: 'success',
          style: GlobalStyles.flashStyle,
          titleStyle: GlobalStyles.flashTextStyle
        })
      }
    } catch (error) {
      console.log(error)
      showMessage({
        message: `Restaurant ${restaurant.name} could not be ordered.`,
        type: 'error',
        style: GlobalStyles.flashStyle,
        titleStyle: GlobalStyles.flashTextStyle
      })
    }
  }
```

Y añadimos el botón:

```
<Pressable
            onPress={() => { orderRestaurant(item) }}
            style={({ pressed }) => [
              {
                backgroundColor: pressed
                  ? GlobalStyles.brandSecondaryTap
                  : GlobalStyles.brandSecondary
              },
              styles.actionButton
            ]}>
          <View style={[{ flex: 1, flexDirection: 'row',
justifyContent: 'center' }]}>
            <MaterialCommunityIcons name='sort' color={'white'}
size={20}/>
            <TextRegular textStyle={styles.text}>
              Order by: {item.orderByPrice ? 'price' : 'default'}
            </TextRegular>
          </View>
        </Pressable>
```

Habrá que bajar el porcentaje de buttonConatiner a 63%

Y ya saldrá:



Falta añadir la frase:

```
        <TextRegular textStyle={{ textAlign: 'right' }}>Currently
sorting products<TextSemiBold> by {item.orderByPrice ? 'price' :
'default'}</TextSemiBold></TextRegular>
```

Y saldrá: