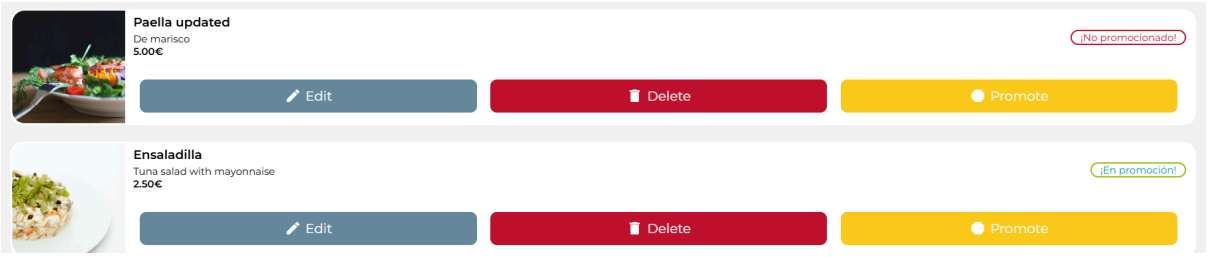
EXAMEN PRODUCTOS PROMOCIONADOS - JUNIO 2022

Se desea permitir a los dueños de restaurantes promocionar los productos de sus restaurantes.
Un dueño podrá promocionar un producto dentro de un restaurante de dos maneras distintas:

- **En el formulario de creación y/o edición de producto. Por defecto, se seleccionará la opción de “no promocionado”. Si el propietario indica que el producto debe estar promocionado, pero ya existían productos promocionados del mismo restaurante, al pulsar el botón Save se mostrará un error y no se creará o editará el producto. El mecanismo de promocionar o no producto, en este caso, será un deslizable.**
- **En la pantalla de "Detalles del restaurante", mediante un botón mostrado junto a cada producto, que permitirá mediante su pulsación promocionar el producto en cuestión. Si el propietario pulsa el botón para promocionar un nuevo producto y ya existían otros productos promocionados del mismo restaurante, se procederá a promocionar el producto indicado y se marcará como "no promocionado" el producto que lo fuese con anterioridad. La aplicación debe pedir confirmación al propietario cuando se pulse el botón; utilice para ello el componente suministrado ConfirmationModal, similar al componente DeleteModal utilizado en la asignatura.**

Además, los productos promocionados aparecerán siempre al principio de los listados de productos que se le presenten tanto a los dueños como a los clientes dentro de un cierto restaurante.
Además de presentarse al principio, los productos promocionados deben destacarse visualmente, por lo que aparecerá una etiqueta de texto Promoted! con brandSuccess; por su contra, aparecerá Not promoted con brandPrimary



BACKEND

Añadir la nueva propiedad:

Tenemos que añadir al modelo de Product la propiedad promoted, que indicará si el producto se promociona (true) o no (false):

Modelo de Product:

```
    promoted: DataTypes.BOOLEAN
```

CreateProduct:

```
promoted: Sequelize.BOOLEAN
```

Y añadimos la validación:

- ***Si el propietario indica que el producto debe estar promocionado, pero ya existían productos promocionados del mismo restaurante, al pulsar el botón Save se mostrará un error y no se creará o editará el producto.***

Solo puede haber un producto promocionado:

```
const checkOnlyOneProductPromoted = async (value, { req }) => {
  try{
    const ProductAlreadyPromoted = await Product.findOne(
      {where: {promoted: true, restaurantId: req.body.restaurantId}}
    )
    if(ProductAlreadyPromoted.length !== 0) {
      return Promise.reject(new Error('Only one product can be
promoted.'))
    }else {
      return Promise.resolve('OK')
    }
  }catch(err){
    return Promise.reject(new Error(err))
  }
}
```

Y añadimos el check al create y update:

```
check('promoted').custom(checkOnlyOneProductPromoted).withMessage('Only
one product can be promoted.')
```

Tenemos que añadir la función que promociona los productos:

```
const promote = async function (req, res) {
  const t = await sequelizeSession.transaction()
  try {
    // const product = await Product.findByPk(req.params.productId)
    const existingPromotedProduct = await Product.findOne({ where: {
restaurantId: req.body.restaurantId, promoted: true } })
    if (existingPromotedProduct) {
      await Product.update(
        { promoted: false },
        { where: { restaurantId: existingPromotedProduct.restaurantId }
},
        { transaction: t }
      )
    }
    await Product.update(
      { promoted: true },
      { where: { restaurantId: req.body.restaurantId } },
      { transaction: t }
    )
    await t.commit()
    const updatedProduct = await Product.findByPk(req.params.productId)
    res.json(updatedProduct)
  } catch (err) {
    await t.rollback()
    res.status(500).send(err)
  }
}
```

Sin transacciones:

```
const promote = async function (req, res) {
  try {
    const product = await Product.findByPk(req.params.productId)
    const productToBeDemoted = await Product.findOne({ where: {
restaurantId: product.restaurantId, promoted: true } })
    if (productToBeDemoted) {
      productToBeDemoted.promoted = false
      await productToBeDemoted.save()
    }
    product.promoted = true
    const promotedProduct = await product.save()
    res.json(promotedProduct)
  } catch (err) {
```

```
    res.status(500).send(err)
  }
}
```

Y además piden:
Además, los productos promocionados aparecerán siempre al principio de los listados de productos que se le presenten tanto a los dueños como a los clientes dentro de un cierto restaurante.

Editamos el show de Restaurant Controller:

```
const show = async function (req, res) {
  // Only returns PUBLIC information of restaurants
  try {
    const 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: [[{ model: Product, as: 'products' }, 'promoted', DESC]]
    }
    )
    res.json(restaurant)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Y por último añadimos la nuevas ruta que recibirá el frontend:

```
app.route('/products/:productId/promote')
    .patch(
      isLoggedIn,
      hasRole('owner'),
      checkEntityExists(Product, 'productId'),
      ProductMiddleware.checkProductOwnership,
      ProductController.promote
    )
```

FRONTEND

Primero tenemos que añadir en el endpoint la ruta para que se promociones:

```
function promote (id) {
  return patch(`products/${id}/promote`)
}
```

Y ahora añadir el botón que promociona:

- *En la pantalla de "Detalles del restaurante", mediante un botón mostrado junto a cada producto, que permitirá mediante su pulsación promocionar el producto en cuestión. Si el propietario pulsa el botón para promocionar un nuevo producto y ya existían otros productos promocionados del mismo restaurante, se procederá a promocionar el producto indicado y se marcará como "no promocionado" el producto que lo fuese con anterioridad. La aplicación debe pedir confirmación al propietario cuando se pulse el botón; utilice para ello el componente suministrado ConfirmationModal, similar al componente DeleteModal utilizado en la asignatura.*

Como hay que crear un modal, hacemos el useState:

```
const [productToBePromoted, setProductToBePromoted] = useState(null)
```

Y añadimos el botón:

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

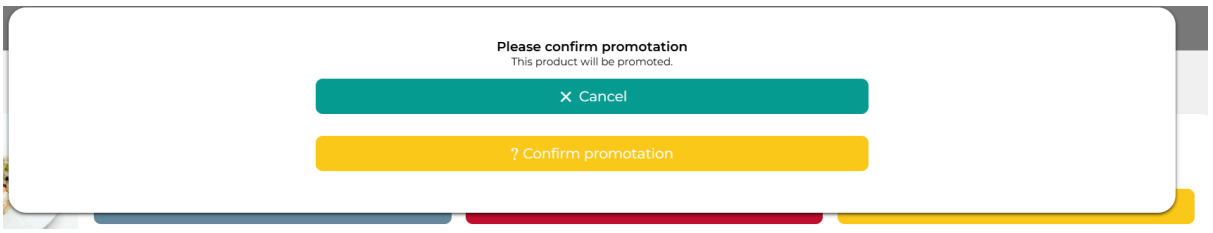
Ahora tenemos que crear la función que permite promocionar el producto:

```
const promoteProduct = async (product) => {
   try {
     await promote(product.id)
     await fetchRestaurantDetail()
     setProductToBePromoted(null)
     showMessage({
       message: `Product ${product.name} succesfully promoted`,
       type: 'success',
       style: GlobalStyles.flashStyle,
       titleStyle: GlobalStyles.flashTextStyle
     })
   } catch (error) {
     console.log(error)
     setProductToBePromoted(null)
     showMessage({
       message: `Product ${product.name} could not be promoted.`,
       type: 'error',
       style: GlobalStyles.flashStyle,
       titleStyle: GlobalStyles.flashTextStyle
     })
   }
 }
```

Y añadimos el modal:

```
<ConfirmationModal
       isVisible={productToBePromoted !== null}
       onCancel={() => setProductToBePromoted(null)}
       onConfirm={() => promoteProduct(productToBePromoted)}>
         <TextRegular>This product will be promoted.</TextRegular>
     </ConfirmationModal>
```

Y ya tenemos cuando pinchamos en promote:



Ahora tenemos que añadir el el texto de promocionado o no promocionado:

Como esta al lado de price, creamos una vista entre estos dos:

```
 <View style={[{ flexDirection: 'row', justifyContent: 'space-between',
alignItems: 'flex-end' }]}>
          <TextRegular
numberOfLines={2}>{item.description}</TextRegular>
          {item.promoted &&
            <TextRegular textStyle={[styles.badge, { color:
GlobalStyles.brandGreen, borderColor: GlobalStyles.brandSuccess }]}>
              ¡En promoción!
            </TextRegular>
          }
          {!item.promoted &&
            <TextRegular textStyle={[styles.badge, { color:
GlobalStyles.brandPrimary, borderColor: GlobalStyles.brandPrimary }]}>
              ¡No promocionado!
            </TextRegular>
          }
        </View>
```

Y ya tendriamos:

Y si la promocionamos:

Para el create product igual.

En el edit añadir la propiedad y el switch:

```
<TextRegular>Is it promoted?</TextRegular>
              <Switch
                trackColor={{ false: GlobalStyles.brandSecondary, true:
GlobalStyles.brandPrimary }}
                thumbColor={values.promoted ?
GlobalStyles.brandSecondary : '#f4f3f4'}
                // onValueChange={toggleSwitch}
                value={values.promoted}
                style={styles.switch}
                onValueChange={value =>
                  setFieldValue('promoted', value)
                }
              />
              <ErrorMessage name={'promoted'} render={msg =>
<TextError>{msg}</TextError> }/>
```

Y tendremos: