EXAMEN RESTAURANTES ECONÓMICOS - JUNIO 2022

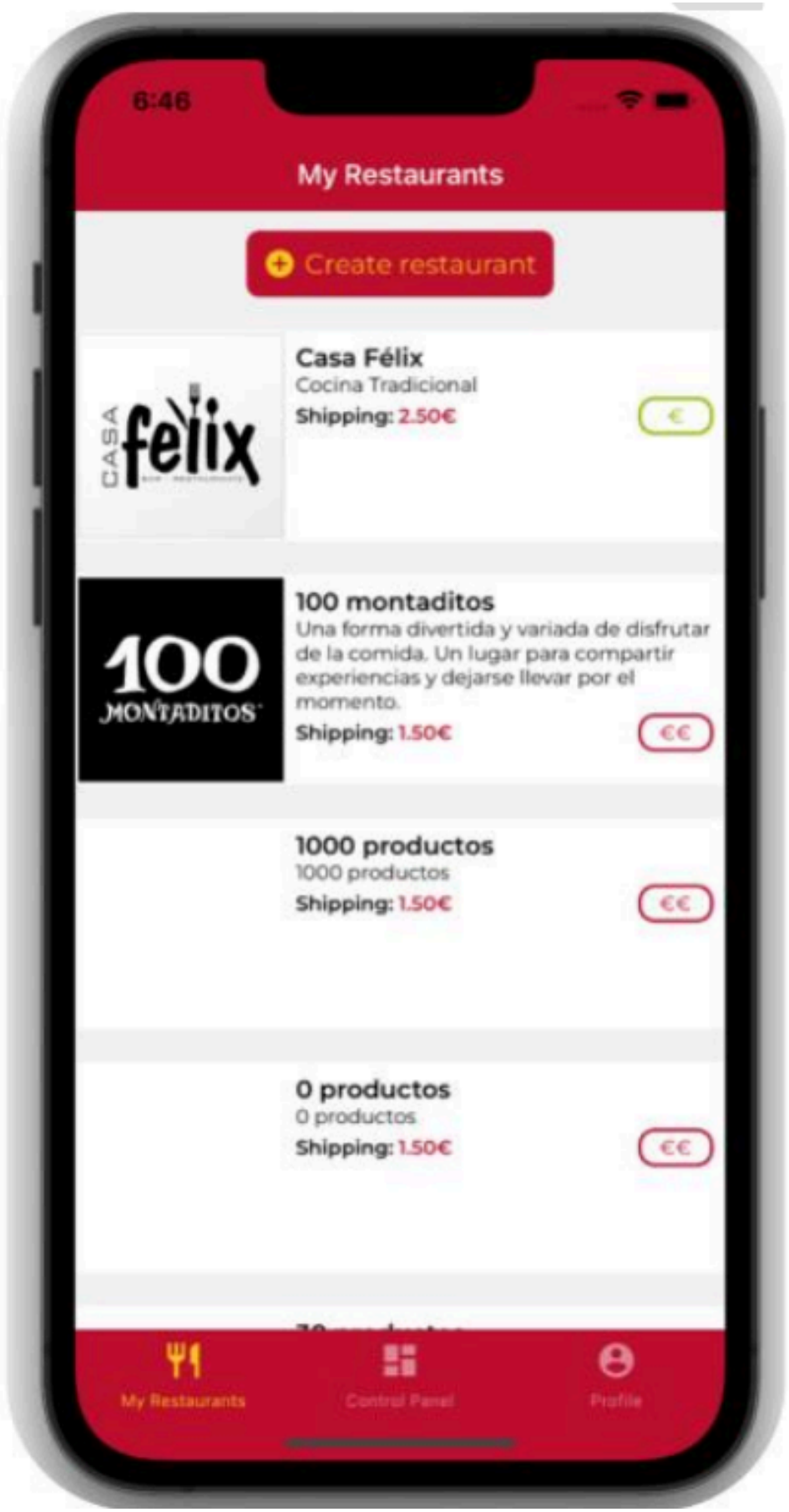
Se desea visualizar los restaurantes que son económicos. Un restaurante económico tendrá una etiqueta € y un restaurante no económico tendrá una etiqueta €€.

Haga uso de la coloración ‘brandSuccess’ y ‘brandPrimary’ respectivamente para las etiquetas anteriores. Un restaurante será económico si, una vez se dé de alta un producto del mismo, el precio medio de los productos del mismo sea menor que el precio medio de los productos del resto de restaurantes.

Nota1: Para hacer un filtrado de productos cuyo restaurante sea distinto del restaurante actual puede usar el operador not equal (Sequelize.Op.ne)

Nota2: Para computar el valor medio de una columna, deberá usar la función Sequelize.fn('AVG', Sequelize.col('columnName')).

Para mayor claridad, puede observar el uso de estos operadores en una de las fotografías adjuntas, donde se computa la media de los costes de envío de los pedidos que no se corresponden con el usuario currentUserId.


6:46
My Restaurants
Create restaurant
Casa Félix
Cocina Tradicional
Shipping: 2.50€
€
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€
€€
1000 productos
1000 productos
Shipping: 1.50€
€€
0 productos
0 productos
Shipping: 1.50€
€€
My Restaurants
Control Panel
Profile

BACKEND

Añadir las nueva propiedad:

Un restaurante económico tendrá una etiqueta € y un restaurante no económico tendrá una etiqueta €€.

Tendremos que añadir una nueva propiedad boolean que indica si es economico (true) o no lo es (false), la añadimos al modelo de Restaurant:

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

Y al create Restaurant:

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

Ahora añadimos la lógica para que la función cambie a económico, pues hemos puesto por defecto no económico (false).

Un restaurante será económico si, una vez se dé de alta un producto del mismo, el precio medio de los productos del mismo sea menor que el precio medio de los productos del resto de restaurantes.
Nota1: Para hacer un filtrado de productos cuyo restaurante sea distinto del restaurante actual puede usar el operador not equal (Sequelize.Op.ne)

En el Product Controller:

```
const updateEconomicRestaurants = async function (RestaurantId) {
  const AvgPriceOtherRestaurants = Product.findAll({
    where: { restaurantId: { [Sequelize.Op.ne]: RestaurantId }},
    attributes: [
      [Sequelize.fn('AVG', Sequelize.col('price')), 'avgPrice']
    ]
  })
  const AvgPriceMyRestaurant = Product.findOne({
    where: {restaurantId: RestaurantId},
    attributes: [
      [Sequelize.fn('AVG', Sequelize.col('price')), 'avgPrice']
    ]
  })
  const restaurant = await Restaurant.findByPk(RestaurantId)
  if (AvgPriceMyRestaurant !== null && AvgPriceOtherRestaurants !==
null) {
    const finalValue = AvgPriceMyRestaurant.avgPrice <
AvgPriceOtherRestaurants.avgPrice
    restaurant.economic = finalValue
  }
  await restaurant.save()
}
```

Y como dice en el enunciado crear, la añadimos al create:

```
const create = async function (req, res) {
  let newProduct = Product.build(req.body)
  try {
    newProduct = await newProduct.save()
    updateEconomicRestaurants(newProduct.restaurantId)
    res.json(newProduct)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

FRONTEND

Tenemos que añadir los símbolos en RestaurantScreen:

En restaurant Screen:

```
<View style={{ flexDirection: 'row', justifyContent: 'space-between',
alignItems: 'flex-end' }} >
          <TextSemiBold>Shipping: <TextSemiBold textStyle={{ color:
GlobalStyles.brandPrimary
}}>{item.shippingCosts.toFixed(2)}€</TextSemiBold></TextSemiBold>
          {item.economic &&
              <TextRegular textStyle={[styles.badge, { color:
GlobalStyles.brandSuccess, borderColor: GlobalStyles.brandSuccess }]
}>€</TextRegular>
          }
          {!item.economic &&
              <TextRegular textStyle={[styles.badge, { color:
GlobalStyles.brandPrimary, borderColor: GlobalStyles.brandPrimary }]
}>€€</TextRegular>
          }
        </View>
```

donde:

```
 badge: {
    textAlign: 'center',
    borderWidth: 2,
    width: 45,
    paddingVertical: 2,
    paddingHorizontal: 10,
    borderRadius: 10
```