EXAMEN ECONOMIC

Evaluación individual laboratorio. Septiembre 2022.

Visibilidad de restaurantes económicos y restaurantes no económicos

Se desea visualizar qué restaurantes son económicos en el listado de restaurantes del propietario y que aparezca una etiqueta € similar a la mostrada en la captura (en `brandSuccess`, abajo y a la derecha) cuando un restaurante sea económico, y dos símbolos de euro €€ cuando el restaurante no lo sea (en `brandPrimary`, abajo y a la derecha).

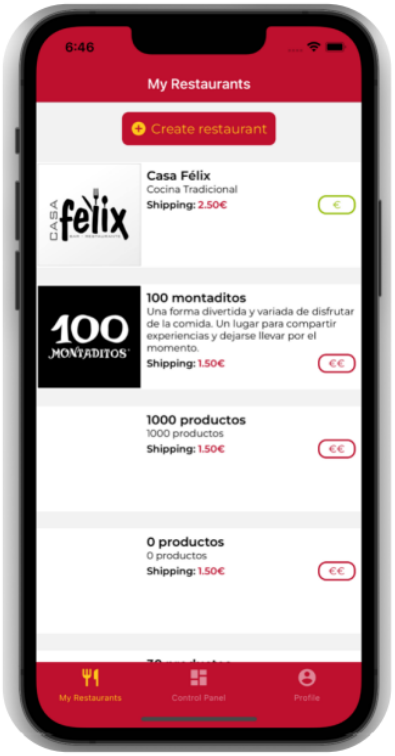
Para ello, cuando se da de alta un producto de un restaurante se deberá computar el precio medio de los productos del resto de restaurantes y compararlo con el precio medio de los productos del restaurante actual, incluyendo el producto recién creado.

Se considerará económico a aquellos restaurantes cuyo precio medio de productos sea inferior al precio medio del resto en el momento de la creación del producto.

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 el siguiente ejemplo donde se computa la media de los costes de envío de los pedidos que no se corresponden con el usuario `currentUserId`:



El planteamiento inicial , es añadir a restaurantes, la propiedad ‘economic’ de tipo boolean que es 0 o 1 , y modificar el créate del controller Product , para que cuando se cree un producto , calcule la media de productos en ese restaurante y los demás, si la media es menor que los demás se actualiza economic a true

BACKEND

Models , RESTAURANT.JS

```
}
Restaurant.init({
  name: DataTypes.STRING,
  description: DataTypes.TEXT,
  economic: {
    type: DataTypes.BOOLEAN,
    defaultValue: false

  },
  address: DataTypes.STRING,
  postalCode: DataTypes.STRING,
  url: DataTypes.STRING,
```

Añadimos la propiedad economic

Migration CREATE-RESTAURANT

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

    },
    address: {
      allowNull: false,
      type: Sequelize.STRING
    },
    postalCode: {
```

CONTROLLER

Restaurant-controller

```
exports.index = async function (req, res) {
  try {
    const restaurants = await Restaurant.findAll(
      {
        attributes: ['id', 'name', 'description', 'economic', 'address', 'postalCode', 'url', 'shippingCosts', 'averageServiceMinutes', 'email', 'phone', 'logo', 'heroIma
        include:
      {
        model: RestaurantCategory,
        as: 'restaurantCategory'
      },
        order: [[{ model: RestaurantCategory, as: 'restaurantCategory' }, 'name', 'ASC']]
      }
    )
    res.json(restaurants)
  } catch (err) {
    res.status(500).send(err)
  }
}

exports.indexOwner = async function (req, res) {
  try {
    const restaurants = await Restaurant.findAll(
      {
        attributes: ['id', 'name', 'description', 'economic', 'address', 'postalCode', 'url', 'shippingCosts', 'averageServiceMinutes', 'email', 'phone', 'logo', 'heroIma
        where: { userId: req.user.id }
      })
    res.json(restaurants)
  } catch (err) {
    res.status(500).send(err)
  }
```

Añadimos la propiedad economic , para poder llamarla en el fronted

Product-Controller

```
const updateRestaurantInexpensiveness = async function (restaurantId) {
  const queryResultOtherRestaurantsAvgPrice = await Product.findOne({
    where: {
      restaurantId: { [Sequelize.Op.ne]: restaurantId }
    },
    attributes: [
      [Sequelize.fn('AVG', Sequelize.col('price')), 'avgPrice']
    ]
  })
  const queryResultCurrentRestaurantAvgPrice = await Product.findOne({
    where: {
      restaurantId
    },
    attributes: [
      [Sequelize.fn('AVG', Sequelize.col('price')), 'avgPrice']
    ]
  })
  if (queryResultCurrentRestaurantAvgPrice !== null && queryResultOtherRestaurantsAvgPrice !== null) {
    const avgPriceOtherRestaurants = queryResultOtherRestaurantsAvgPrice.dataValues.avgPrice
    const avgPriceCurrentRestaurant = queryResultCurrentRestaurantAvgPrice.dataValues.avgPrice
    const economicc = avgPriceCurrentRestaurant < avgPriceOtherRestaurants
    Restaurant.update({ economic: economicc }, { where: { id: restaurantId } })
  }
}
```

Creamos esta constante (que tambien podriamos meterla dentro del create , pero asi se ve mas limpio)

Según el codigo que nos dan en el enunciado

queryResultOtherRestaurantAvgPrice → Calculamos la media de precio de los productos para otros restaurantes que no son el que se mete por parametro

queryResultRestaurantAvgPrice → Calculamos la media de precio de los productos para el restaurante en el que se esta creando el producto

Si los dos no son null:

Obtenemos el valor numerico de ambas medias

Si la media de precio del restaurante donde se esta creando es menor que la media de precio de otro restaurante , se actualiza el valor economic para el restaurante dnde se esta creando el producto y se pone a true

```
exports.create = async function (req, res) {
  let newProduct = Product.build(req.body)
  if (typeof req.file !== 'undefined') {
    newProduct.image = req.file.destination + '/' + req.file.filename
  }
  try {
    newProduct = await newProduct.save()
    updateRestaurantInexpensiveness(newProduct.restaurantId)

    res.json(newProduct)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Metemos la constante en el create

FRONTED

SCREENS

Restaurant-Screen

```
{item.economic && <TextRegular textStyle={[styles.economic, { color:
GlobalStyles.brandSuccess, borderColor: GlobalStyles.brandSuccess }]
}>€</TextRegular> }
        {!item.economic && <TextRegular textStyle={[styles.economic, {
color: GlobalStyles.brandPrimary, borderColor: GlobalStyles.brandPrimary
}] }>€€</TextRegular> }
```

Añadimos esto para ver los simbolitos de €

Donde pone styles.economic , creamos abajo :

```
const styles = StyleSheet.create({
  container: {
    flex: 1
  },
  economic: {
    marginLeft: 1220,
    marginTop: -30,
    borderWidth: 2,
    width: 45,
    borderRadius: 10

  },
  button: {
    borderRadius: 8,
```

SOLUCION