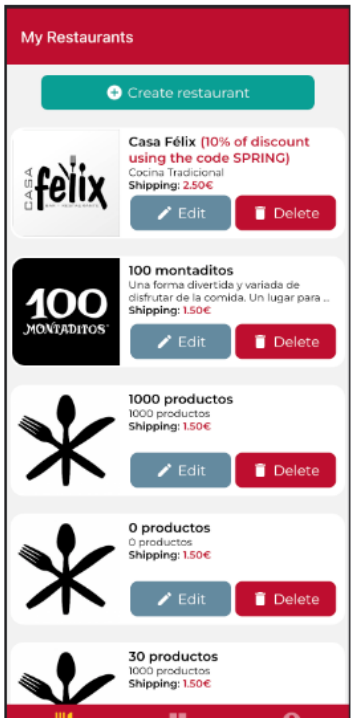
EXAMEN DESCUENTO

Extracto - Códigos de descuento para restaurantes

Realice las modificaciones que considere necesarias, tanto en el backend como en el frontend, para cumplir con los nuevos requisitos que se describen a continuación.

La compañía ha decidido ofrecer a los propietarios la posibilidad de asociar un código de descuento (p. ej.: VENTAS20) a sus restaurantes, para que el sistema muestre y aplique posteriormente el código de promoción de descuento especificado. Como ejemplo ilustrativo, el dueño de un restaurante podría aplicar el código PRIMAVERA asociado a un 10% de descuento a *Casa Félix* , y otro código de descuento, por ejemplo EXTREMO, asociado a un 30% de descuento a otro restaurante.

El sistema debe mostrar los restaurantes con el descuento registrado cuando tanto el código de descuento como el valor del descuento tienen un valor, como se muestra en la siguiente captura de pantalla:



Para el planteamiento de este problema , tenemos que crear 2 propiedades en la entidad restaurante , que sean ‘codigo descuento’ y ‘descuento’

A la hora de crear un restaurante debemos añadir el código que queremos aplicar y también el descuento asociado a ese código

BACKEND

Models RESTAURANT.JS

```
Restaurant.init({
  name: DataTypes.STRING,
  description: DataTypes.TEXT,
  //
  codigoDescuento: {
    type: DataTypes.STRING,
    defaultValue: ''
  },
  descuento: {
    type: DataTypes.INTEGER,
    defaultValue: 0
  },
  //
  address: DataTypes.STRING,
  postalCode: DataTypes.STRING,
```

Migrations CREATE-RESTAURANT

```
    type: Sequelize.TEXT
  },
  //
  codigoDescuento: {
    type: Sequelize.STRING,
    defaultValue: ''
  },
  descuento: {
    type: Sequelize.INTEGER,
    defaultValue: 0
  },
  //
  address: {
    allowNull: false,
```

Validation RESTAURANT-VALIDATION

```
// El código de descuento no se puede repetir para restaurantes propiedad del mismo propietario.
const checkDescuento = async (value, { req }) => {
  try {
    const descuento = await Restaurant.findOne({ where: { userId: req.user.id, codigoDescuento: req.body.codigoDescuento } })
    if (descuento !== null) {
      return Promise.reject(new Error('Ya hay un restaurante con este descuento'))
    } else { return Promise.resolve() }
  } catch (err) {
    return Promise.reject(new Error(err))
  }
}
```

Primero en la constate descuento , buscamos un restaurante con id : req.user.id es decir del mismo propietario y con código de descuento req.body.vodigoDescuento , es decir con el mismo código de despuesto que el restaurante actual

Si se encuentra un restaurante con esas características

Devuelve un error diciendo que ya existe un restaurante con ese descuento

```
module.exports = {
  create: [
    check('name').exists().isString().isLength({ min: 1, max: 255 }).trim(),
    check('description').optional({ nullable: true, checkFalsy: true }).isString().trim(),
    //
    check('codigoDescuento').optional({ nullable: true, checkFalsy: true }).isString().isLength({ min: 1, max: 10 }).trim().custom(checkDescuento),
    check('descuento').exists().isFloat({ min: 0, max: 99 }).toFloat(),
    //
    check('address').exists().isString().isLength({ min: 1, max: 255 }).trim(),
    check('postalCode').exists().isString().isLength({ min: 1, max: 255 }),
```

Controller RESTAURANT-CONTROLLER

```
exports.index = async function (req, res) {
  try {
    const restaurants = await Restaurant.findAll(
      {
        attributes: ['id', 'name', 'description', 'codigoDescuento', 'descuento', 'address', 'postalCode', 'url', 'shippingCosts', 'averageServiceMinutes', 'email'
        include:
      {
        model: RestaurantCategory,
        as: 'restaurantCategory'
      },
        order: [['codigoDescuesto', 'DESC'], [{ 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', 'codigoDescuento', 'descuento', 'address', 'postalCode', 'url', 'shippingCosts', 'averageServiceMinutes', 'email'
        where: { userId: req.user.id },
        order: [['descuento', 'DESC']]

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

Añadimos las propiedades para poder utilizarlas en el Fronted y añadimos el order , para que los restaurante sean ordenados, primero los que tienen descuento y después los que no

PRODUCT-CONTROLLER

```
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 {
    const PrecioO = newProduct.price
    const restaurant = await Restaurant.findOne({ where: { id: req.body.restaurantId } })
    const descuento = restaurant.descuento
    newProduct.price = PrecioO - PrecioO * (descuento / 100)
    newProduct = await newProduct.save()
    res.json(newProduct)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Esto me lo he inventado yo

Si cuando creamos un producto nuevo en un restaurante que tiene un codigo de descuento , que ese codigo de descuento se aplic al precio de ese producto

Por tanto metemos en una constante el precio inicial , y metemos en la constante restaurant el restaurante donde estamos creando el producto

Obtenemos el descuento del restaurante

Y mediante esa formula , calculamos el precio con el descuento

FRONTED

RESTAURANT-SCREEN

```
<TextRegular numberOfLines={2}>{item.description}</TextRegular>
{item.averageServiceMinutes !== null &&
  <TextSemiBold>Avg. service time: <TextSemiBold textStyle={{ color: GlobalStyles.brandPrimary }}>{item.averageServiceMinutes} min.</TextSemiBold></TextSemiBold>
}
<TextSemiBold>Shipping: <TextSemiBold textStyle={{ color: GlobalStyles.brandPrimary }}>{item.shippingCosts.toFixed(2)}€</TextSemiBold></TextSemiBold>

<View style={{ position: 'absolute', top: 0, right: 10 }}>
  {item.codigoDescuento
    ? <TextRegular textStyle={{ color: 'red' }}>
    {item.descuento}% of discount using the code {item.codigoDescuento}</TextRegular>
    : <TextRegular></TextRegular> }

</View>

<View style={styles.actionButtonsContainer}>
  <Pressable
    onPress={() => navigation.navigate('EditRestaurantScreen', { id: item.id })
    }
    style={({ pressed }) => [
```

Añadimos esto , para que a la derecha de cada restaurante , si tienen descuento aparezca el descuento y su código

CREATE-RESTAURANT

```
export default function CreateRestaurantScreen ({ navigation }) {
  const [open, setOpen] = useState(false)
  const [restaurantCategories, setRestaurantCategories] = useState([])
  const [backendErrors, setBackendErrors] = useState()

  const initialRestaurantValues = { name: null, description: null, codigoDescuento: null, descuento: null, address: null, postalCode: null, url: nu
  const validationSchema = yup.object().shape({
    name: yup
      .string()
      .max(255, 'Name too long')
      .required('Name is required'),
    codigoDescuento: yup
      .string()
      .max(10, 'Descuento too long'),
    descuento: yup
      .number()
      .positive()
      .integer()
      .min(1)
      .max(99),
    address: yup
      .string()
      .max(255, 'Address too long')
      .required('Address is required'),
    postalCode: yup
```

En la pantalla créate restaurant , metemos las propiedades para crearlas junto al restaurante

```
    label='Name:'
  />
  <InputItem
    name='description'
    label='Description:'
  />
  <InputItem
    name='codigoDescuento'
    label='Codigo de Descuento:'
  />
  <InputItem
    name='descuento'
    label='Descuento:'
  />
  <InputItem
    name='address'
    label='Address:'
  />
  <InputItem
    name='postalCode'
```

RESTAURANT-DETAIL-SCREEN

```
<TextSemiBold textStyle={styles.price}>{item.price.toFixed(2)}€</TextSemiBold>
{!item.availability &&
  <TextRegular textStyle={styles.availability }>Not available</TextRegular>
}
  <View style={{ position: 'absolute', top: 0, right: 10 }}>
  {restaurant.codigoDescuento
    ? <TextRegular textStyle={{ color: 'red' }}>Este producto tiene un {restaurant.descuento}% de descuento </TextRegular>
    : <TextRegular></TextRegular> }

  </View>
 <View style={styles.actionButtonsContainer}>
  <Pressable
    onPress={() => navigation.navigate('EditProductScreen', { id: item.id })
    }
    style={({ pressed }) => [
      {
        backgroundColor: pressed
          ? GlobalStyles.brandBlueTap
```

Esto me lo he inventado yo , a la derecha de cada producto en el que se le aplique el descuento del restaurante aparecerá el aviso de que ese producto esta rebajado

SOLUCION