EXAMEN DESCUENTOS RESTAURANTES Y PRODUCTOS- JULIO 2023

La empresa ha decidido ofrecer a los propietarios la posibilidad de asociar un descuento (entero entre 0 y 100, por ejemplo: 10) a sus restaurantes, para que posteriormente el sistema muestre y aplique el descuento especificado a sus productos. Los propietarios podrán establecer un porcentaje de descuento diferente para cada uno de sus restaurantes y que después se aplicarían a los productos de cada restaurante.

añadri propiedades a migrations y a models

Sin embargo, algunos propietarios han indicado que no todos los productos de sus catálogos se promocionan habitualmente. Por ello, la empresa ha decidido, además de permitir al restaurante indicar el porcentaje de descuento sobre el precio original, dar la posibilidad de indicar en cada producto si será promocionado o no.

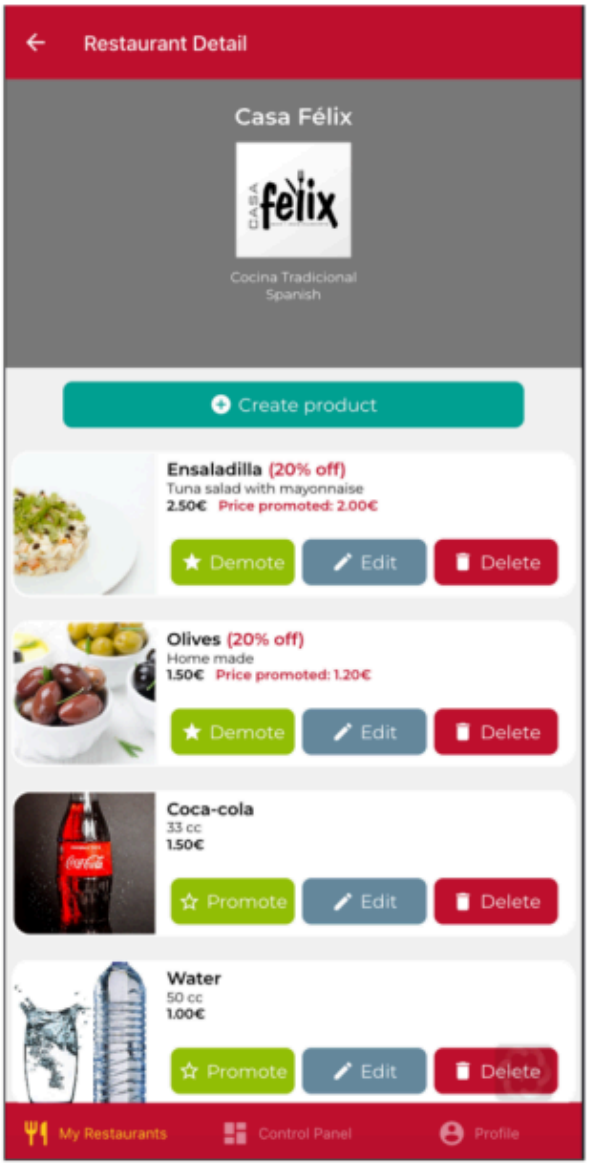
Todos los productos promocionados de un restaurante serán promocionados con el mismo porcentaje de descuento que el propietario indique en las propiedades de sus restaurantes. Tenga en cuenta que un producto puede estar promocionado pero, si el porcentaje de descuento indicado en el restaurante es 0%, no tendrá ningún efecto. Sólo los propietarios de un restaurante podrán activar y desactivar los productos promocionados así como indicar el descuento promocional.

Por defecto, los restaurantes no tendrán ningún porcentaje de descuento (es decir, el porcentaje de descuento será 0).
Recuerde validar en el backend que, al promocionar/despromocionar un producto, el restaurante asociado tenga un porcentaje de descuento superior a 0. En caso contrario, no podrá ser promocionado.

Por lo tanto, las vistas de edición y creación de los restaurantes deben incorporar un campo nuevo en el formulario (InputItem) para cambiar el porcentaje de descuento, permitiendo al propietario ajustar el valor de la propiedad. Recuerde validar en frontend (y backend) que el valor de ese nuevo campo es compatible con el tipo de dato y las restricciones mencionadas.

Para que el propietario pueda determinar a qué productos se les aplica el descuento del restaurante, modifique la vista de detalle del restaurante, donde se muestra el listado de sus productos, para que incluya un botón que permita al propietario activar o desactivar la promoción para ese producto. En caso de que el restaurante no tenga un descuento mayor que 0, no se mostrará este botón.
Las vistas de creación y edición de producto no incluirán la opción de aplicar o no aplicar el descuento.

El sistema debería mostrar los restaurantes con el descuento aplicado y la opción de promocionarlos o quitarles la promoción tal y como se muestra en la siguiente captura de pantalla:



Como ejemplo ilustrativo, el propietario de un restaurante podría aplicar un descuento del 10% al restaurante1, y otro descuento del 20% al restaurante2. Recuerda que:

1. **El descuento está en el intervalo [0,100].**
2. **Si un restaurante activa el descuento promocional, los productos pueden estar promocionados o no.**

Implemente los cambios necesarios en Backend y Frontend para incluir esta funcionalidad.

Añadir la nueva propiedad al modelo de restaurant:

```
discount: {
      type: DataTypes.INTEGER,
      defaultValue: 0
    }
```

Y al modelo de product:

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

Y al create de Restaurant:

```
discount: {
        type: Sequelize.INTEGER,
        defaultValue: 0
      },
```

Y el de Product:

```
applicable: {
        type: Sequelize.BOOLEAN,
        allowNull: false,
        defaultValue: false
      }
```

Añadimos la validación tanto al create y al update::

```
  // El descuento está en el intervalo [0,100].
  check('discount').isInt({ min: 0, max: 100 }).toInt()
```

Añadimos en el middleware de product:
Todos los productos promocionados de un restaurante serán promocionados con el mismo porcentaje de descuento que el propietario indique en las propiedades de sus restaurantes. Tenga en cuenta que un producto puede estar promocionado pero, si el porcentaje de descuento indicado en el restaurante es 0%, no tendrá ningún efecto.

```
const checkProductCanBePromoted = async (req, res, next) => {
  try {
    const product = await Product.findByPk(req.params.productId)
    const restaurant = await Restaurant.findByPk(product.restaurantId)
    if (restaurant.percentage > 0) {
      return next()
    } else {
      return res.status(409).send('This product cannnot be promoted')
    }
  } catch (err) {
    return res.status(500).send(err.message)
  }
}
```

Y por último la función que promociona el producto:

```
const promote = async function (req, res) {
  try {
    const t = sequelizeSession.transaction()
    const productToBeUpdated = await
Product.findByPk(req.params.productId)
    if (productToBeUpdated.applicable === true) {
      await Product.update({ applicable: false }, { where: { id:
productToBeUpdated.id } }, { transaction: t })
    } else {
      await Product.update({ applicable: true }, { where: { id:
productToBeUpdated.id } }, { transaction: t })
    }
    await t.commit()
    const updatedProduct = await Product.findByPk(req.params.productId)
    res.json(updatedProduct)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Añadimos el orden:

```
const indexRestaurant = async function (req, res) {
  try {
    const products = await Product.findAll({
      where: {
        restaurantId: req.params.restaurantId
      },
      include: [
        {
          model: ProductCategory,
          as: 'productCategory'
        }],
      // SOLUCION
      order: [['applicable', 'DESC']]
    })
    res.json(products)
  } catch (err) {
    res.status(500).send(err)
  }
}

const show = async function (req, res) {
```

```
  // Only returns PUBLIC information of products
  try {
    const product = await Product.findByPk(req.params.productId, {
      include: [
        {
          model: ProductCategory,
          as: 'productCategory'
        }],
      // SOLUCION
      order: [['applicable', 'DESC']]
    }
    )
    res.json(product)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Y la ruta:

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

Creamos la funcion que llama al endpoint:

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

Añadimos el botón:

```
{restaurant.discount > 0 &&
         <Pressable
            onPress={() => { setProductToBePromoted(item) }}
            style={({ pressed }) => [
              {
                backgroundColor: pressed
                  ? GlobalStyles.brandGreenTap
                  : GlobalStyles.brandGreen
              },
              styles.actionButton
            ]}>
          <View style={[{ flex: 1, flexDirection: 'row',
justifyContent: 'center' }]}>
            <MaterialCommunityIcons name={item.applicable ? 'star' :
'star-outline'} color={'white'} size={20}/>
            <TextRegular textStyle={styles.text}>
              {item.applicable ? 'Demote' : 'Promote'}
            </TextRegular>
          </View>
        </Pressable>}
```

Luego añadimos el titulo:

```
 const renderCardTitle = (item) => {
    return (<>
        <TextSemiBold style={{ fontSize: 15
}}>{item.name}</TextSemiBold>
         <TextSemiBold style={{ marginLeft: 5, fontSize: 15, color:
GlobalStyles.brandPrimary }}>{item.applicable ? restaurant.discount +
'% OFF!' : ''}</TextSemiBold>
      </>)
  }
```

Y:

```
<ImageCard
        imageUri={item.image ? { uri: process.env.API_BASE_URL + '/' +
item.image } : defaultProductImage}
        title={renderCardTitle(item)}
      >
        <TextR
```

Y queda añadir el precio:

```
 const finalP = (item) => {
    return item.price - ((restaurant.discount * item.price) / 100)
  }
```

```
 {item.applicable &&
        <TextSemiBold textStyle={styles.price}>Final
price:{finalP(item.price)}€</TextSemiBold>
        }
```