EXAMEN NUTRIENTES

Evaluación individual laboratorio. Junio 2022.

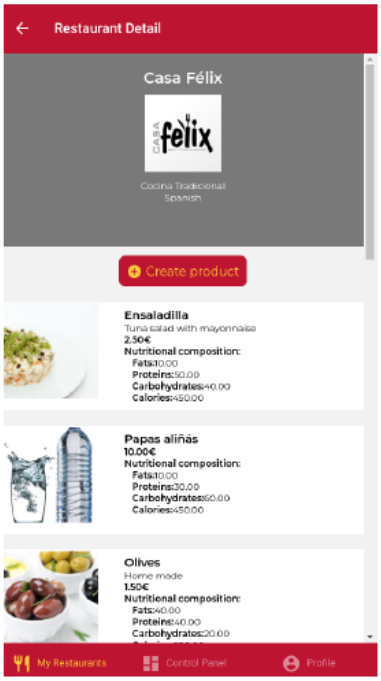
Nutrientes saludables

Una vez se ha puesto en marcha la primera versión de DeliverUS, los inversores han solicitado la inclusión de una nueva funcionalidad que consiste en informar al cliente de los gramos de macronutrientes que contienen cada uno de los productos. Tras la reunión con el cliente, se ha establecido los siguientes requisitos:

1.- Debido a la normativa reguladora de alimentación, se pide que se informe al cliente de la cantidad de carbohidratos, proteínas y grasas por cada 100 gramos de cada uno de los productos que están a la venta en DeliverUS.

2.- Dada la existencia de platos hipercalóricos que no están recomendados en una dieta saludable se pide que un plato no pueda contener más de 1000 calorías por 100g de producto. Para ello, se usará la siguiente formula aproximada de cálculo energético

Calorías producto = (grasas * 9) + (proteínas * 4) + (carbohidratos * 4)



Se pide que:

1.- Implemente los cambios que sean necesarios en el backend.

2.- Implemente validación de servidor y cliente con los requisitos de calorías máximas.

3.- Muestre los datos de calorías por 100g, carbohidratos, proteínas y grasas en cada uno de los productos que se muestran en la pantalla RestaurantDetailScreen.

- Nota: No es necesario modificar los seeders. Recuerde que los valores numéricos enviados por el frontend, llegan al backend como string cuando se usan TextInputs en el componente InputItem. No olvide realizar un casting a float de dichos valores para que el backend los trate sin problema

Para este examen tenemos que añadir los valores grasas, proteínas, carbohidratos y calorías , después para calcular las calorias según la formula anterior , hay que modificar el controller del créate

BACKEND

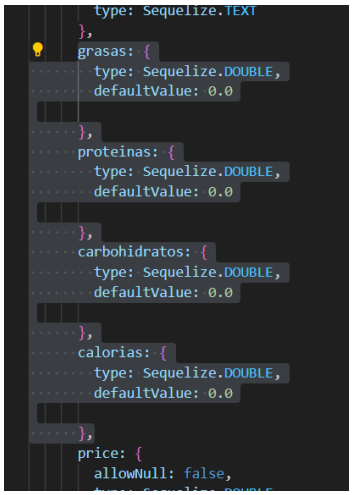
Models PRODUCT-MODELS

```
Product.init({
  name: DataTypes.STRING,
  description: DataTypes.STRING,
  grasas: {
    type: DataTypes.DOUBLE,
    defaultValue: 0.0

  },
  proteinas: {
    type: DataTypes.DOUBLE,
    defaultValue: 0.0
```

```
},
carbohidratos: {
  type: DataTypes.DOUBLE,
  defaultValue: 0.0

},
calorias: {
  type: DataTypes.DOUBLE,
  defaultValue: 0.0

},
price: DataTypes.DOUBLE,
image: DataTypes.STRING,
```

Añadimos todas las propiedades y ponemos el defaultValue a 0.0

Migrations CREATE-PRODUCT



Añadimos todas las propiedades y ponemos el defaultValue a 0.0

Controller 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 grasas = newProduct.grasas
    const proteinas = newProduct.proteinas
    const carbohidratos = newProduct.carbohidratos
    newProduct.calorias = 9 * grasas + 4 * proteinas + 4 * carbohidratos
    newProduct = await newProduct.save()
    res.json(newProduct)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Modificamos la funcion create , para que a la hora de crear un producto , las calorias se calculen según la formula y los valores de grasas, proteinas y carbohidratos

Resvisamos todas las funciones y si en alguna están escritos todos los atributos ,añadimos los nuevos

Validations PRODUCT-VALIDATION

Según el enunciado , no se pueden crear producto con mas de 1000 calorias por cada 100 gramos

```
const checkNoMore1000Calorias = async (value, { req }) => {
  try {
    const grasas = req.body.grasas
    const proteinas = req.body.proteinas
    const carbohidratos = req.body.carbohidratos
    const calorias = (9 * grasas) + (4 * proteinas) + (4 * carbohidratos)
    if (calorias < 1000 && grasas + proteinas + carbohidratos === 100) {
      return Promise.resolve()
    } else { return Promise.reject(new Error('Unhealthy product.')) }
  } catch (err) {
    return Promise.reject(new Error(err))
  }
}
```

Para los valores entrante (req.body) de grasas, proteinas y carbohidratos , si las calorias son menos que 1000 , y la suma de las propiedades son 100(es decir son 100 gramos) , se ejecuta

Error: Al quitar en el enunciado el value , aunque no se utilice , ya me daba error .Por tanto NO QUITAR EN VALUE

```
check('description').optional({ checkNull: true, checkFalsy: true }
check('grasas').exists().isFloat({ min: 0 }).toFloat(),
check('proteinas').exists().isFloat({ min: 0 }).toFloat(),
check('carbohidratos').exists().isFloat({ min: 0 }).toFloat(),
check('calorias').custom(checkNoMore1000Calorias),
check('price').exists().isFloat({ min: 0 }).toFloat(),
```

Añadir al créate , las propiedades , como solo nos interesa crear productos , no se añade al update

FRONTED

SCREENS- RESTAURANT-DETAIL-SCREEN

```
const renderProduct = ({ item }) => {
  return (
    <ImageCard
      imageUri={item.image ? { uri: process.env.API_BASE_URL + '/' + item.image } : defaultProductImage}
      title={item.name}
    >
      <TextRegular numberOfLines={2}>{item.description}</TextRegular>
      <TextSemiBold textStyle={styles.price}>{item.price.toFixed(2)}€</TextSemiBold>
      {!item.availability &&
        <TextRegular textStyle={styles.availability }>Not available</TextRegular>
      }
      <View style={{ }}>
       <TextSemiBold>Nutritional Composition:</TextSemiBold>
        <TextRegular> Grasas: {item.grasas}</TextRegular>
        <TextRegular> Proteinas: {item.proteinas}</TextRegular>
        <TextRegular> Carbohidratos: {item.carbohidratos}</TextRegular>
        <TextRegular> Calorias: {item.calorias}</TextRegular>

      </View>

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

```
    flexDirection: 'column',
    width: '33%'
  },
  actionButtonsContainer: {
    flexDirection: 'row',
    bottom: 5,
    position: 'absolute',
    width: '90%',
    marginLeft: 200

  }
})
```

Para que se puedan ver bien los valores nutricionales , movemos los botones hacia la izquierda , añadiendo :

CREATE-PRODUCT-SCREEN

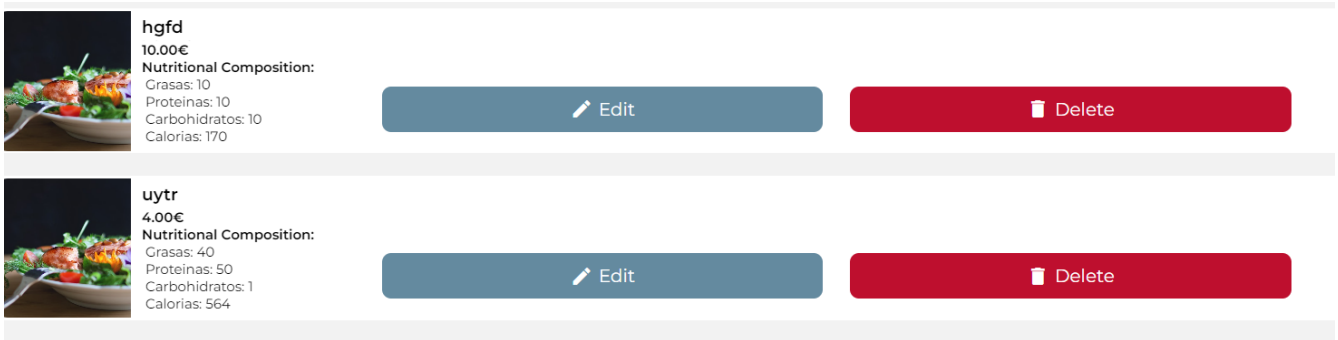
```
const [backendErrors, setBackendErrors] = useState()

const initialProductValues = { name: null, description: null, grasas: 0.0, proteinas: 0.0, carbohidratos: 0.0, price: null, order: n
const validationSchema = yup.object().shape({
  name: yup
    .string()
    .max(255, 'Name too long')
    .required('Name is required'),
  grasas: yup
    .number()
    .positive('Please provide a positive grasas value')
    .required('grasas is required'),
  proteinas: yup
    .number()
    .positive('Please provide a positive proteinas value')
    .required('proteinas is required'),
  carbohidratos: yup
    .number()
    .positive('Please provide a positive carbohidratos value')
    .required('carbohidratos is required'),
  price: yup
    .number()
    .positive('Please provide a positive price value')
```

Ponemos los valor que queremos añadir , y sus ‘restricciones’

```
      />
      <InputItem
        name='description'
        label='Description:'
      />
      <InputItem
        name='grasas'
        label='Grasas:'
      />
      <InputItem
        name='proteinas'
        label='Proteinas:'
      />
      <InputItem
        name='carbohidratos'
        label='Carbohidratos:'
      />
      <InputItem
        name='price'
        label='Price:'
      />
      <InputItem
```

Con esto creamos la celda , donde metemos los valores

SOLUCION



Description:

Grasas:

Proteinas:

Carbohidratos:

Price:

calorias-Unhealthy product.