➔ PROMOCION SIMULACRO

- **Backend**

-Restaurant(models)

```
promoted: {
  type: DataTypes.BOOLEAN,
  dafaultValue: false
},
```

-Restaurant (migrations)

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

-RestaurantValidation

```
const checkOnePromotedRestaurant = async (ownerId,promotedValue) =>{
  if(promotedValue){
    try{
      const promotedRestaurants = await Restaurant.findAll({where:{usedId:ownerId,promoted:true}})
      if(promotedRestaurants.length > 0){
        return Promise.reject(new Error("You can only promote one restaurant"))
      }
    }catch(err){
      return Promise.reject(new Error(err))
    }
  }
  return Promise.resolve('ok')
}
```

***Owner id se puede poner como req.user.id**

```
check('promoted').custom((value,{req})=> {
  return checkOnePromotedRestaurant(req.body.userId,value)
}).withMessage('You can only promoted one restaurant '),
```

-RestaurantController

```
/*const promote2 = async function(req,res){
  const t = await sequelizeSession.transaction()
  try{
    const existingPromotedRestaurant = await Restaurant.findOne({where:{userId: req.user.id,promoted:true}})
    if(existingPromotedRestaurant){
      await Restaurant.update(
        {promoted:false},
        {where:{id:existingPromotedRestaurant.id}},
        {transaction: t}
      )
    }
    await Restaurant.update(
      {promoted:true},
      {where: {id: req.params.restaurantId}},
      {transaction: t}
    )
    await t.commit();
    const updatedRestaurant = await Restaurant.findByPk(req.params.restaurantId)
    res.json(updatedRestaurant)
  }catch(err){
    await t.rollback();
    res.status(500).send(err)
}
}*/
```

```
const promote = async function(req,res){
  try{
    const restaurant = await Restaurant.findByPk(req.params.restaurantId)
    if(restaurant.promoted === true){
      restaurant.promoted = false
    }else{
      restaurant.promoted = true
    }
    await restaurant.save()
    res.json(restaurant)
  }catch(err){
    res.status(500).send(err)
  }
}
```

```
const index = async function (req, res) {
  try {
    const restaurants = await Restaurant.findAll(
      {
        attributes: { exclude: ['userId'] },
        include:
      {
        model: RestaurantCategory,
        as: 'restaurantCategory'
      },
        order: [['promoted','DESC'],[{ model: RestaurantCategory, as: 'restaurantCategory' }, 'name', 'ASC']]
      }
    )
    res.json(restaurants)
  } catch (err) {
    res.status(500).send(err)
  }
}


const indexOwner = async function (req, res) {
  try {
    const restaurants = await Restaurant.findAll(
      {
        attributes: { exclude: ['userId'] },
        where: { userId: req.user.id },
        order: [['promoted','DESC']],
        include: [{
          model: RestaurantCategory,
          as: 'restaurantCategory'
        }]
      })
    res.json(restaurants)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

(para que aparezcan arriba del listado los promocionados)

-RestaurantRoutes

```
app.route('/restaurants/:restaurantId/promote')
    .patch(
      isLoggedIn,
      hasRole('owner'),
      checkEntityExists(Restaurant, 'restaurantId'),
      RestaurantMiddleware.checkRestaurantOwnership,
      RestaurantController.promote
    )
```

- **Frontend**

-RestaurantEndPoints

***que coincida con la ruta**

```
function promote (id) {
  return patch(`restaurants/${id}/promote`)
}
```

***importar patch y exportar promote**

-ConfirmationModal

***Copiar y pegar DeleteModal y cambiar el mensaje y el nombre del botón**

-RestaurantScreen

***importar ConfirmationModal y promote del endpoint**

```
const [restaurantToBePromoted,setRestaurantToBePromoted] = useState(null)
```

```
{/* Solution */}
<Pressable
    onPress={() => { setRestaurantToBePromoted(item) }}
    style={({ pressed }) => [
      {
        backgroundColor: pressed
          ? GlobalStyles.brandSuccessTap
          : GlobalStyles.brandSuccess
      },
      styles.actionButton
    ]}>
  <View style={[{ flex: 1, flexDirection: 'row', justifyContent: 'center' }]}>
    <MaterialCommunityIcons name='octagram' color={'white'} size={20}/>
    <TextRegular textStyle={styles.text}>
      Promote
    </TextRegular>
  </View>
</Pressable>
```

```
const promoteRestaurant = async (restaurant) => {
  try {
    await promote(restaurant.id)
    await fetchRestaurants()
    setRestaurantToBePromoted(null)
    showMessage({
      message: `Restaurant ${restaurant.name} succesfully promoted`,
      type: 'success',
      style: GlobalStyles.flashStyle,
      titleStyle: GlobalStyles.flashTextStyle
    })
  } catch (error) {
    console.log(error)
    setRestaurantToBePromoted(null)
    showMessage({
      message: `Restaurant ${restaurant.name} could not be promoted.`,
      type: 'error',
      style: GlobalStyles.flashStyle,
      titleStyle: GlobalStyles.flashTextStyle
    })
  }
}
```

```
<ConfirmationModal
    isVisible={restaurantToBePromoted !== null}
    onCancel={() => setRestaurantToBePromoted(null)}
    onConfirm={() => promoteRestaurant(restaurantToBePromoted)}>
        <TextRegular>Other promoted restaurant, if any, will be unpromoted</TextRegular>
</ConfirmationModal>
</>
```

```
<View style = {{flexDirection:'row',justifyContent: 'space-between',alignItems:'flex-end'}}>
  <TextSemiBold>Shipping: <TextSemiBold textStyle={{ color: GlobalStyles.brandPrimary }}>{item.shippingCosts.toFixed(2)}€</T
  { item.promoted &&
      <TextRegular textStyle={[styles.badge,{color:GlobalStyles.brandPrimary, borderColor: GlobalStyles.brandSuccess}]}>
        ¡En promocion!
      </TextRegular>
  }
</View>
```

***El shipping se pone dentro del View para que salga en la misma línea que el pero al final**

```
badge:{
    textAlign: 'center',
    borderWidth: 2,
    paddingHorizontal:10,
    borderRadius:10
}
```

-borderRadius: redondo

-paddingHorizontal:lo pegado que esta al texto el circulo

-borderWidth : anchura del circulo

```
<TextRegular> it is promoted? </TextRegular>
<Switch
    trackColor= {{false: GlobalStyles.brandBlue,true:GlobalStyles.brandPrimary}}
    thumbColor= {values.promoted? GlobalStyles.brandSecondary: '#648a9f'}
    value= {values.promoted}
    style= {styles.switch}
    onValueChange={value => setFieldValue('promoted',value)}></Switch>
```

➔ ONLINE/OFFLINE

- **Backend**

-RestaurantController

```
const changeStatus = async function(req,res) {
  try{
    const restaurant = await Restaurant.findByPk(req.params.restaurantId)
    if(restaurant.status === 'online'){
      restaurant.status = 'offline'
    } else {
      return restaurant.status = 'online'
    }
    await restaurant.save()
    res.json(restaurant)
  }catch(err){
    res.status(500).send(err)
  }
}
```

-RestaurantMiddlewares

***porque hay que comprobar algo que no es un atributo de restaurant (por eso no se hace en Validation)**

```
const allOrdersAreDelivered = async (req ,res, next) => {
  try{
    const orders = await Order.findOne({where: {restaurantId: req.params.restaurantId, deliveredAt:null}})
    if(orders===null){
      return next()
    }
    return res.status(401).send('This restaurants has orders with deliveredAt null')
  }catch(err){
    return res.status(500).send(err.message)
  }
}
```

-RestaurantRoutes

```
app.route('/restaurants/:restaurantId/status')
    .patch(
      isLoggedIn,
      hasRole('owner'),
      checkEntityExists(Restaurant, 'restaurantId'),
      RestaurantMiddleware.checkRestaurantOwnership,
      RestaurantMiddleware.allOrdersAreDelivered,
      RestaurantController.changeStatus
    )
```

- **Frontend**

-RestaurantEndPoints

***importar patch y exportar status**

```
function status (id) {
  return patch(`restaurants/${id}/status`)
}
```

-RestaurantScreen

```
const changeStatus = async (restaurant) => {
  try {
    await status(restaurant.id)
    await fetchRestaurants()
    showMessage({
      message: `Status of restaurant ${restaurant.name} succesfully change`,
      type: 'success',
      style: GlobalStyles.flashStyle,
      titleStyle: GlobalStyles.flashTextStyle
    })
  } catch (error) {
    console.log(error)
    //setRestaurantToBeDeleted(null)
    showMessage({
      message: `Status of restaurant ${restaurant.name} could not be change.`,
      type: 'error',
      style: GlobalStyles.flashStyle,
      titleStyle: GlobalStyles.flashTextStyle
    })
  }
}
```

```
{item.status!='closed' && item.status!= 'temporarily closed' &&<Pressable
    onPress={() => { changeStatus(item) }}
    style={({ pressed }) => [
      {
        backgroundColor: pressed
          ? GlobalStyles.brandGreenTap
          : GlobalStyles.brandGreen
      },
      styles.actionButton
    ]}>
    <View style={[{ flex: 1, flexDirection: 'row', justifyContent: 'center' }]}>
      <MaterialCommunityIcons name='check' color={'white'} size={20}/>
      <TextRegular textStyle={styles.text}>
        {item.status}
      </TextRegular>
    </View>
  </Pressable>}
```

➔ CREAR RESTAURANTCATEGORY

- **Backend**

-RestaurantCategoryValidation

***solo se comprueba que existe una vez porque es en el create**

```
import { RestaurantCategory } from '../../models/models.js'
import { check } from 'express-validator'

const checkNonExistingCategory = async (value, req) => {
  try {
    const category = await RestaurantCategory.findOne({ where: { name: value }
    })
    if (category) {
      return Promise.reject(new Error('This category already exists'))
    }
    return Promise.resolve()
  } catch (err) {
    return Promise.reject(new Error(err))
  }
}


const create = [
  check('name').isString().isLength({ min: 1, max: 50 }).trim(),
  check('name').custom((value, { req }) => {
  return checkNonExistingCategory(value, req)
  })
]

export { create }
```

-RestaurantCategoryController

```
const create = async function (req, res) {
  const newRestaurantCategory = RestaurantCategory.build(req.body)
  try {
    const restaurant = await newRestaurantCategory.save()
    res.json(restaurant)
  } catch (err) {
    console.log('error')
    res.status(500).send(err)
  }
}
```

-RestaurantCategoryRoute

```
app.route('/restaurantCategories')
    .post(
      isLoggedIn,
      hasRole('owner'),
      RestaurantCategoryValidation.create,
      handleValidation,
      RestaurantCategoryController.create)
```

- **Frontend**

-RestaurantEndPoint

```
function createCategory(data){
  return post(`restaurantCategories`,data)
}
```

-CreateRestaurant

```
<Pressable
  onPress={()=> navigation.navigate('CreateRestaurantCategoryScreen')}
  style={({ pressed }) => [
    {
      backgroundColor: pressed
        ? GlobalStyles.brandGreenTap
        : GlobalStyles.brandGreen
    },
    styles.button
  ]}>
<View style={[{ flex: 1, flexDirection: 'row', justifyContent: 'center' }]}>
<MaterialCommunityIcons name='folder-plus-outline' color={'white'} size={20} />
  <TextRegular textStyle={styles.text}>
    New category
  </TextRegular>
</View>
</Pressable>
```

-CreateRestaurantCategory

```jsx
import React, { useEffect, useState } from 'react'
import { Image, Platform, Pressable, ScrollView, StyleSheet, View} from 'react-native'
import { MaterialCommunityIcons } from '@expo/vector-icons'
import * as yup from 'yup'
import { createCategory } from '../../api/RestaurantEndpoints'
import InputItem from '../../components/InputItem'
import TextRegular from '../../components/TextRegular'
import * as GlobalStyles from '../../styles/GlobalStyles'
import { showMessage } from 'react-native-flash-message'
import TextError from '../../components/TextError'
import { ErrorMessage, Formik } from 'formik'

export default function CreateRestaurantScreen ({ navigation }) {
  const [backendErrors, setBackendErrors] = useState()

  const initialRestaurantValues = { name: null, createdAt:null,updatedAt:null}
  const validationSchema = yup.object().shape({
    name: yup
      .string()
      .max(50, 'Name too long')
      .required('Name is required')
  })

  const createRestaurantCategory = async (values) => {
    setBackendErrors([])
    try {
      const createdRestaurantCategory = await createCategory(values)
      showMessage({
        message: `Restaurant ${createdRestaurantCategory.name} succesfully created`,
        type: 'success',
        style: GlobalStyles.flashStyle,
        titleStyle: GlobalStyles.flashTextStyle
      })
      navigation.navigate('CreateRestaurantScreen', { dirty: true })
    } catch (error) {
      console.log(error)
      setBackendErrors(error.errors)
    }
  }
```

```
return (
  <Formik
    validationSchema={validationSchema}
    initialValues={initialRestaurantValues}
    onSubmit={createRestaurantCategory}>
    {({ handleSubmit, setFieldValue, values }) => (
      <ScrollView>
        <View style={{ alignItems: 'center' }}>
          <View style={{ width: '60%' }}>
            <InputItem
              name='name'
              label='Name:'
            />

            {backendErrors &&
              backendErrors.map((error, index) => <TextError key={index}>{error.param}-{error.msg}</TextError>)
            }

            <Pressable
              onPress={handleSubmit}
              style={({ pressed }) => [
                {
                  backgroundColor: pressed
                    ? GlobalStyles.brandSuccessTap
                    : GlobalStyles.brandSuccess
                },
                styles.button
              ]}>
              <View style={[{ flex: 1, flexDirection: 'row', justifyContent: 'center' }]}>
                <MaterialCommunityIcons name='content-save' color={'white'} size={20}/>
                <TextRegular textStyle={styles.text}>
                  Save
                </TextRegular>
              </View>
            </Pressable>
          </View>
        </View>
      </ScrollView>
    )}
  </Formik>
```

-RestaurantStack

***se modifica solo cuando creamos una nueva screen**

```
<Stack.Screen
  name='CreateRestaurantCategoryScreen'
  component={CreateRestaurantCategoryScreen}
  options={{
    title: 'Create Restaurant Category'
  }} />
```

➔ DESCUENTOS INPUT

- **Backend**

-Restaurant (Model)

```
discount: {
  allowNull: false,
  type: DataTypes.DOUBLE,
  defaultValue: 0.0
},
```

-Product (Model)

```
promote:{
  allowNull: false,
  type: DataTypes.BOOLEAN,
  defaultValue: false
}
```

-Restaurant (Migrations)

```
discount: {
  allowNull: false,
  type: Sequelize.DOUBLE,
  defaultValue: 0.0
},
```

-Product(Migrations)

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

-RestaurantValidation

***en create y update**

```
check('discount').optional().isFloat({min:0,max:100}).toFloat()
```

-ProductValidation

***en create y update**

```
check('promote').optional().isBoolean().toBoolean()
```

-ProductMiddleware

```
const checkRestaurantHaveDiscount = async(req,res,next) => {
  try{
    const product= await Product.findByPk(req.params.productId)
    const restaurant = await Restaurant.findByPk(product.restaurantId)
    if(restaurant.discount>0){
      return next()
    }else{
      return res.status(409).send('This restaurant does not have discount')
    }
  }catch(err){
    return res.status(500).send(err.message)
  }
}
```

-ProductController

```
const promoteProduct = async function (req,res) {
  try{
    const product = await Product.findOne({where: {id: req.params.productId}})
    if(product !== null){
      if(product.promote){
        product.promote=false
      }else{
        product.promote=true
      }
      await product.save()
      res.json(product)
    }else{
      res.status(404).send({message: 'Product not found'})
    }
  }catch(err){
    res.status(500).send(err)
  }
}
```

-ProductRoute

```
app.route('/products/:productId/promote')
.patch(
  isLoggedIn,
  hasRole('owner'),
  ProductController.promoteProduct,
  ProductMiddleware.checkRestaurantHaveDiscount
)
```

- **Frontend**

-ProductEndpoint

```
function promote(id){
  return patch(`/products/${id}/promote`)
}
```

-CreateRestaurantScreen

***InitialValues:**

```
lse,discount:null }
```

```
  discount: yup
    .number()
    .max(100)
    .min(0)
})
```

```
<InputItem
  name='discount'
  label='Discount:'
/>
```

-EditRestaurantScreen

***igual que en el otro**

-RestaurantDetailScreen

```
title={(
  <View style= {styles.productContainer}>
    <TextRegular textStyle={styles.productTitle}>{item.name}</TextRegular>
    {item.promote && <TextRegular textStyle= {styles.priceOff}numberOfLines={2}>({restaurant.discount} % off)</TextRegular>}
  </View>
)}

<TextRegular numberOfLines={2}>{item.description}</TextRegular>

View style = {styles.productContainer}>

  <TextSemiBold textStyle={styles.price}>{item.price.toFixed(2)}€ </TextSemiBold>
  {item.promote &&
    <TextRegular textStyle={styles.priceOff }> Price promoted: { item.price - (item.price * (restaurant.discount) / 100)}€</Te
  }
/View>
```

```
priceOff: {
  fontSize: 16,  //letra tamaño
  color: 'red',
  alignSelf: 'center',
  marginLeft: 5
},
productContainer: {
  flexDirection: 'row',
  marginBottom: 10
}
```

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

```
const promoteProduct = 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
    })
  }
}
```

➔ PINEAR

- **Backend**

-Restaurant(Model)

```
pinnedAt: {
  type: DataTypes.DATE,
  allowNull: true
},
```

-Restaurant(Migrations)

```
pinnedAt: {
  type: Sequelize.DATE,
  allowNull: true
},
```

-RestaurantValidation

```
check('pinned').optional().isBoolean().toBoolean()
```

-RestaurantController

```
async function _getPinnedRestaurants (req) {
  return await Restaurant.findAll({
    attributes: { exclude: ['userId'] },
    where: {
      userId: req.user.id,
      pinnedAt: {
        [Sequelize.Op.not]: null // Uso de Sequelize.Op.not para filtrar no nulos
      }
    },
    order: [['pinnedAt', 'ASC']], // Ordenados ascendente por 'pinnedAt'
    include: [{
      model: RestaurantCategory,
      as: 'restaurantCategory'
    }]
  })
}

async function _getNoPinnedRestaurants (req) {
  return await Restaurant.findAll({
    attributes: { exclude: ['userId'] },
    where: {
      userId: req.user.id,
      pinnedAt: null
    },
    order: [['pinnedAt', 'ASC']], // Ordenados ascendente por 'pinnedAt'
    include: [{
      model: RestaurantCategory,
      as: 'restaurantCategory'
    }]
  })
}
```

```
const indexOwner = async function (req, res) {
  try {
    const restaurants = [...(await _getPinnedRestaurants(req)),...(await _getNoPinnedRestaurants(req))]
    res.json(restaurants)
  } catch (err) {
    res.status(500).send(err)
  }
}


const create = async function (req, res) {
  const newRestaurant = Restaurant.build(req.body)
  newRestaurant.userId = req.user.id // usuario actualmente autenticado
  newRestaurant.pinnedAt = req.body.pinned ? new Date() : null
  try {
    const restaurant = await newRestaurant.save()
    res.json(restaurant)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

***pinned es el boolean de si esta seleccionado o no en el frontend, pinnedAt es la fecha que le pasamos como atributo al restaurante**

```
const togglePinned = async function (req, res) {
  try {
    const restaurant = await Restaurant.findByPk(req.params.restaurantId)
    if(!restaurant.pinnedAt){
      restaurant.pinnedAt = new Date()
    }else{
      restaurant.pinnedAt = null
    }
    await restaurant.save()
    res.json(restaurant)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

-RestaurantRoute

```
app.route('/restaurants/:restaurantId/togglePinned')
  .patch(
    isLoggedIn,
    hasRole('owner'),
    checkEntityExists(Restaurant, 'restaurantId'),
    RestaurantMiddleware.checkRestaurantOwnership,
    RestaurantController.togglePinned
  )
```

- **Frontend**

-RestaurantEndpoint

```
function togglePinned (id) {
  return patch(`restaurants/${id}/togglePinned`)
}
```

-ConfirmationModal

***copiar DeleteModal y cambiar el nombre del texto y del boton**

-RestaurantScreen

```
const [restaurantToBePinned,setRestaurantToBePinned] = useState(null)
```

```
const pinnedRestaurant = async (restaurant) => {
  try {
    await togglePinned(restaurant.id)
    await fetchRestaurants()
    setRestaurantToBePinned(null)
    showMessage({
      message: `Restaurant ${restaurant.name} succesfully pinned`,
      type: 'success',
      style: GlobalStyles.flashStyle,
      titleStyle: GlobalStyles.flashTextStyle
    })
  } catch (error) {
    console.log(error)
    setRestaurantToBePinned(null)
    showMessage({
      message: `Restaurant ${restaurant.name} could not be pinned.`,
      type: 'error',
      style: GlobalStyles.flashStyle,
      titleStyle: GlobalStyles.flashTextStyle
    })
  }
}
```

```
<ConfirmationModal
    isVisible={restaurantToBePinned !== null}
    onCancel={() => setRestaurantToBePinned(null)}
    onConfirm={() => pinnedRestaurant(restaurantToBePinned)}>
</ConfirmationModal>
```

```
<View style={{ flexDirection: 'row', justifyContent: 'space-between', alignItems: 'flex-end' }} >
  <Pressable onPress={() => { setRestaurantToBePinned(item) }}>
    <MaterialCommunityIcons
      name={item.pinnedAt ? 'pin' : 'pin-outline'}
      color={GlobalStyles.brandSecondaryTap}
      size={30}
    />
  </Pressable>
</View>

</View>
```

***el botón dentro del view porque tiene que estar al final, y no hay vista dentro porque solo hay un elemento sin texto**

-CreateRestaurantScreen

```
,pinned:false }
```

```
<Switch
  trackColor= {{false: GlobalStyles.brandBlue,true:GlobalStyles.brandPrimary}}
  thumbColor= {values.promoted? GlobalStyles.brandSecondary: '#648a9f'}
  value= {values.pinned}
  style= {styles.switch}
  onValueChange={value => setFieldValue('pinned',value)}></Switch>
```

➔ VISIBLE

- **Backend**

-Product(Model)

```
visibleUntil: DataTypes.DATE,
```

-Product(Migrations)

```
visibleUntil: {
  type: Sequelize.DATE
},
```

-ProductValidation

***en create y update**

```
check('visibleUntil').optional().isDate().toDate(),
check('visibleUntil').custom((value, { req }) => {
  const currentDate = new Date()
  if (value && value < currentDate) {
    return Promise.reject(new Error('The visibility must finish after the current date.'))
  } else { return Promise.resolve() }
}),
check('availability').custom((value, { req }) => {
  if (value === false && req.body.visibleUntil) {
    return Promise.reject(new Error('Cannot set the availability and visibility at the same time.'))
  } else { return Promise.resolve() }
})
```

-RestaurantController

```
const show = async function (req, res) {
  try {
    const currentDate = new Date();
    const restaurant = await Restaurant.findByPk(req.params.restaurantId, {
      attributes: { exclude: ['userId'] },
      include: [
        {
          model: Product,
          as: 'products',
          where: {
            visibleUntil: {
              [Sequelize.Op.or]: [
                { [Sequelize.Op.eq]: null },
                { [Sequelize.Op.gt]: currentDate }
              ]
            }
          },
          include: {
            model: ProductCategory,
            as: 'productCategory'
          }
        },
        {
          model: RestaurantCategory,
          as: 'restaurantCategory'
        }
      ],
      order: [[{ model: Product, as: 'products' }, 'order', 'ASC']]
    });

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

- **Frontend**

-CreateProductScreenç

***en initialValues**

```
visibleUntil:null}
```

```
visibleUntil: yup
  .date()
  .nullable()
)
```

```
<InputItem
  name='visibleUntil'
  label='Visible until:'
/>
```

-EditProductScreen

***igual todo que el anterior**

-RetaurantDetailScreen

```
/*
La división entre 1000 * 3600 * 24 se realiza para convertir la diferencia de tiempo entre dos fechas de milisegundos a días.
Aquí está el razonamiento detrás de esta operación:
Milisegundos a Segundos: La diferencia de tiempo entre dos fechas (timeDiff) se expresa inicialmente en milisegundos.
Para convertir esta cantidad a segundos, se divide por 1000. Esto se debe a que hay 1000 milisegundos en un segundo.
Segundos a Horas: Después de obtener el tiempo en segundos, se divide entre 3600 para convertirlo a horas.
Esto se debe a que hay 3600 segundos en una hora.
Horas a Días: Finalmente, para obtener el tiempo en días, se divide entre 24, ya que hay 24 horas en un día.*/

const isAboutToBeInvisible = (deadline) => {
  console.log(deadline)
  console.log(typeof (deadline))
  const currentDate = new Date()
  const deadlineDate = new Date(deadline)

  const timeDiff = deadlineDate.getTime() - currentDate.getTime()  //devuelve el tiempo en milisegundos

  const daysLeft = Math.ceil(timeDiff / (1000 * 3600 * 24))

  return daysLeft <= 7
}
```

```
{item.visibleUntil && isAboutToBeInvisible(item.visibleUntil) &&
  <TextRegular textStyle= {styles.visible}>It is about to disapear!</TextRegular>
}
```

```
visible: {
  textAlign: 'right',
  marginRight: 5,
  color: GlobalStyles.brandPrimary
},
```

➔ DESCUENTOS

- **Backend**

-Restaurant(Model)

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

-Product(Model)

```
basePrice: DataTypes.DOUBLE,
```

-Restaurant(Migrations)

```
percentage:{
  type: Sequelize.DOUBLE,
  defaultValue: 0.0
},
```

-Product(Migrations)

```
visibleUntil: {
  type: Sequelize.DATE
},
```

-Product(Seeder)

⁎Tenga en cuenta que este cambio DEBE persistir el precio del producto en la base de datos, aplicando el nuevo precio al listado de productos de dicho restaurante individualmente.

```
lad with mayonnaise', price: 2.5,basePrice:2.5,
price: 1.5,basePrice:1.5, image: process.env.PR
```

***añadir en todos los productos el basePrice igual que el price**

-RestaurantValidation

***en update**

```
check('percentage').exists().isFloat({ min: -5, max: 5 }).toFloat()
```

-RestaurantController

```
const update = async function (req, res) {
  try {
    const transaction = await sequelizeSession.transaction()
    await Restaurant.update(req.body, { where: { id: req.params.restaurantId } },transaction)

    const productsToBeUpdated = await Product.findAll({
      where: { restaurantId: req.params.restaurantId },
    });

    for(const p of productsToBeUpdated){
      const newPrice = p.basePrice + p.basePrice * (req.body.percentage / 100);
      await p.update({price:newPrice},transaction);
    }
    await transaction.commit();

    const updatedRestaurant = await Restaurant.findByPk(req.params.restaurantId)
    res.json(updatedRestaurant)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

-ProductController

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

```
const update = async function (req, res) {
  try {
    req.body.basePrice = req.body.price
    await Product.update(req.body, { where: { id: req.params.productId } })
    let updatedProduct = await Product.findByPk(req.params.productId)
    res.json(updatedProduct)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

- **Frontend**

-ConfirmationModal

***el de siempre**

-EditRestaurantScreen

***boolean que dira si cambia o no el porcentaje(si es distinto de 0)**

```
const [percentageShowDialog, setPercentageShowDialog] = useState(false)
```

***initialValues**

```
percentage:0}
```

```
percentage: yup
  .number()
  .max(5)
  .min(-5)
)
```

```
const updateRestaurant = async (values) => {
  setBackendErrors([])

  if(values.percentage !==0 && !percentageShowDialog){
    setPercentageShowDialog(true)
  }else{
    setPercentageShowDialog(false)
  }
  try {
    const updatedRestaurant = await update(restaurant.id, values)
    showMessage({
      message: `Restaurant ${updatedRestaurant.name} succesfully updated`,
      type: 'success',
      style: GlobalStyles.flashStyle,
      titleStyle: GlobalStyles.flashTextStyle
    })
    navigation.navigate('RestaurantsScreen', { dirty: true })
  } catch (error) {
    console.log(error)
    setBackendErrors(error.errors)
  }
}
```

***row es en fila**

***flexDirection: 1 para que estén uno encima de otros**

***justifyContent: para donde va a estar situado todo el view, con espacios entre elementos...**

***alignItems: alinear los elementos**

```
{/* Solution */}
<View style={{ flexDirection: 'row', justifyContent: 'center', alignItems: 'center', marginTop: 20, marginBottom: 10 }} >
  <Pressable onPress={() => {
    let newPercentage = values.percentage + 0.5
    if (newPercentage < 5)
      setFieldValue('percentage', newPercentage)
  }}>
    <MaterialCommunityIcons
      name={'arrow-up-circle'}
      color={GlobalStyles.brandSecondaryTap}
      size={40}
    />
  </Pressable>

  <TextSemiBold>Porcentaje actual: <TextSemiBold textStyle={{ color: GlobalStyles.brandPrimary }}>{values.percentage.toFixed(1)}%</Te

  <Pressable onPress={() => {
    let newPercentage = values.percentage - 0.5
    if (newPercentage > -5)
      setFieldValue('percentage', newPercentage)
  }}>
    <MaterialCommunityIcons
      name={'arrow-down-circle'}
      color={GlobalStyles.brandSecondaryTap}
      size={40}
    />
  </Pressable>
</View>
```

```
<ConfirmationModal2
  isVisible={percentageShowDialog}
  onCancel={() => setPercentageShowDialog(false)}
  onConfirm={() => updateRestaurant(values)}>
</ConfirmationModal2>
```

-CreateRestaurantScreen

***igual que el edit**

-RestaurantScreen

```
{item.percentage != 0 && <View style={{ flexDirection: 'row', justifyContent: 'space-between', alignItems: 'flex-end' }} >
  <TextSemiBold textStyle={{ color: item.percentage > 0 ? 'red' : 'green' }}>{item.percentage > 0 ? '¡Incremento de precios aplicados!' : '¡Descuentos aplicados!'}</Te
</View>
}
```

***alternativo sin view**

```
{item.percentage != 0 && (
  <TextSemiBold textStyle={{ color: item.percentage > 0 ? 'red' : 'green' }}>{item.percentage > 0 ? '¡Incremento de precios aplica
)}
```

➔ USOS DE SEQUALIZE.OP

- **Operadores de comparación básicos:**
  - `$eq`: Igual a
  - `$ne`: No igual a
  - `$gt`: Mayor que
  - `$gte`: Mayor o igual que
  - `$lt`: Menor que
  - `$lte`: Menor o igual que
- **Operadores de comparación avanzados:**
  - `$between`: Entre un rango específico (ejemplo: `{ $between: [6, 10] }`)
  - `$notBetween`: No entre un rango específico
  - `$in`: En un conjunto de valores (ejemplo: `{ $in: [1, 2] }`)
  - `$notIn`: No en un conjunto de valores
- **Operadores de texto y búsqueda:**
  - `$like`: Similar a (puede contener caracteres comodín `%`)
  - `$notLike`: No similar a
  - `$iLike`: Similar a, insensible a mayúsculas/minúsculas
  - `$notILike`: No similar a, insensible a mayúsculas/minúsculas
  - `$regexp`: Coincide con una expresión regular
  - `$notRegexp`: No coincide con una expresión regular
- **Operadores booleanos:**
  - `$or`: Uno de varios criterios
  - `$and`: Todos los criterios
  - `$not`: No cumple con un criterio dado

- **Operadores de conjunto:**
    - `$contains`: Contiene todos los elementos de un conjunto (solo para tipos de datos de conjunto)
    - `$contained`: Está contenido en un conjunto (solo para tipos de datos de conjunto)
    - `$overlap`: Se superpone con un conjunto (solo para tipos de datos de conjunto)
- **Operadores de conjunto y matriz:**
    - `$any`: Al menos uno de un conjunto o matriz de valores

➔ ENLACES

Repositorio: https://github.com/migalcval

-SOBRE SEQUALIZE:

Estructura: https://sequelize.org/docs/v6/other-topics/migrations/#migration-skeleton

DataTypes: https://sequelize.org/v5/manual/data-types.html

-SOBRE SEEDERS:

https://sequelize.org/docs/v6/other-topics/migrations/#creating-the-first-seed

-Propiedades del req (y cosas sobre el app.route creo): https://expressjs.com/en/4x/api.html#req

-Info codigos de estado: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

-Info para crear middlewares: https://expressjs.com/en/guide/writing-middleware.html

-Info para usar middlewares: https://expressjs.com/en/guide/using-middleware.html

-Métodos de validation: https://github.com/validatorjs/validator.js#validators

-Documentacion flatList: https://reactnative.dev/docs/flatlist

https://reactnative.dev/docs/flatlist#listemptycomponent

-Documentacion encabezado: https://reactnative.dev/docs/flatlist#listheadercomponent

-Documentacion componente 'tarjeta': https://m3.material.io/components/cards/

-Documentacion flexbox(los componentes de <text> y demas): https://reactnative.dev/docs/flexbox

-Documentacion sobre image picker de expo: https://docs.expo.dev/tutorial/image-picker/

-Documentacion sobre dropdownpicker: https://hossein-zare.github.io/react-native-dropdown-picker-website/docs/usage

-Documentacion sobre switch: https://reactnative.dev/docs/switch

-En este enlace estan los símbolos que puedes poner en los bótones: https://static.enapter.com/rn/icons/material-community.htm

➔ **TABLA DE ERRORES**

1XX Informational	
100	Continue
101	Switching Protocols
102	Processing

2XX Success	
200	OK
201	Created
202	Accepted
203	Non-authoritative Information
204	No Content
205	Reset Content
206	Partial Content
207	Multi-Status
208	Already Reported
226	IM Used

3XX Redirectional	
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
307	Temporary Redirect
308	Permanent Redirect

4XX Client Error	
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout

4XX Client Error Continued	
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Payload Too Large
414	Request-URI Too Long
415	Unsupported Media Type
416	Requested Range Not Satisfiable
417	Expectation Failed
418	I'm a teapot
421	Misdirected Request
422	Unprocessable Entity
423	Locked
424	Failed Dependency
426	Upgrade Required
428	Precondition Required
429	Too Many Requests
431	Request Header Fields Too Large
444	Connection Closed Without Response
451	Unavailable For Legal Reasons
499	Client Closed Request

5XX Server Error	
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported
506	Variant Also Negotiates
507	Insufficient Storage
508	Loop Detected
510	Not Extended
511	Network Authentication Required
599	Network Connect Timeout Error