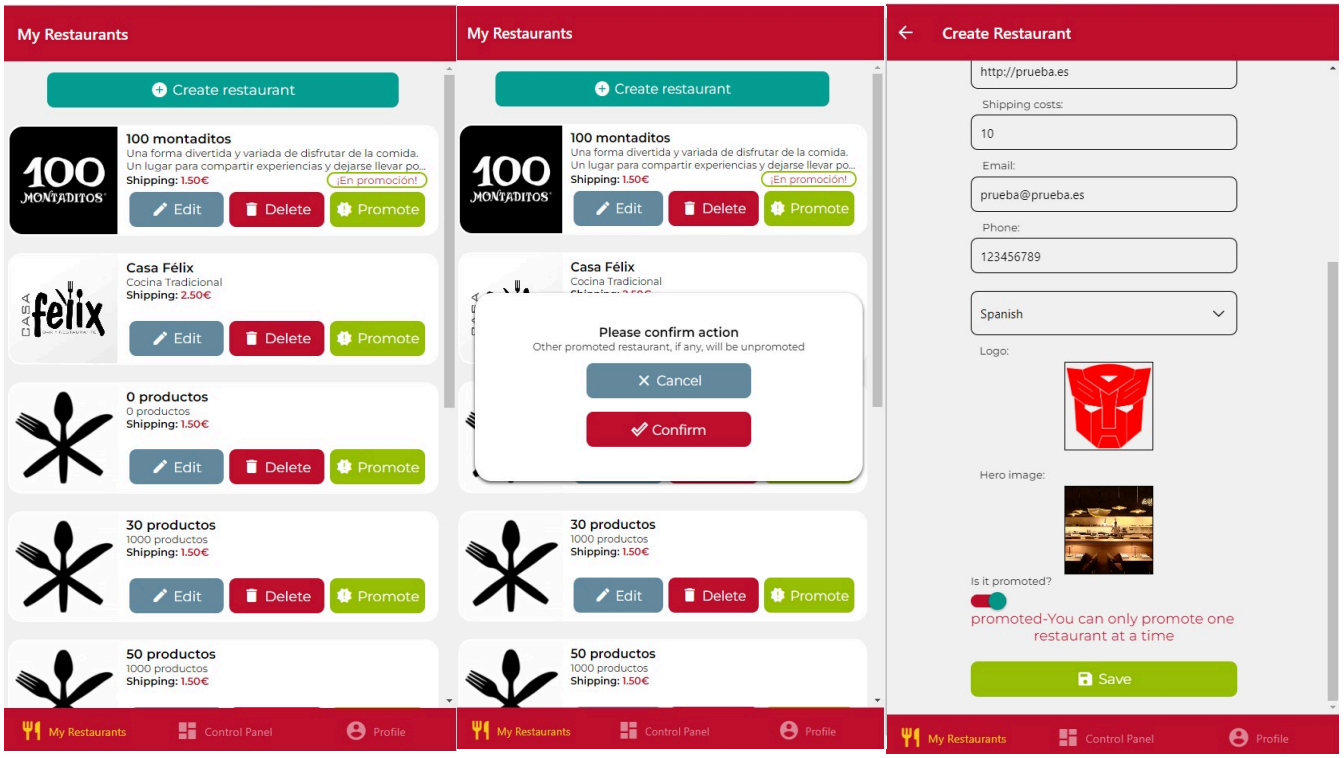
EXAMEN RESTAURANTES PROMOCIONADOS - JUNIO 2022

Se desea permitir a los dueños de restaurantes promocionar sus restaurantes. Un dueño podrá promocionar un restaurante de dos maneras distintas:

- **En el formulario de creación y/o edición de restaurante. Por defecto, se seleccionará la opción de "no promocionado". Si el propietario indica que el nuevo restaurante debe estar promocionado, pero ya existían restaurantes promocionados del mismo dueño, al pulsar el botón Save se mostrará un error y no se creará o editará el restaurante.**
- **En la pantalla de "Mis restaurantes", mediante un botón mostrado junto a cada restaurante, que permitirá mediante su pulsación promocionar el restaurante en cuestión. Si el propietario pulsa el botón para promocionar un nuevo restaurante y ya existían otros restaurantes promocionados del mismo dueño, se procederá a promocionar el restaurante indicado y se marcará como "no promocionado" el restaurante que lo fuese anteriormente.**

La aplicación debe pedir confirmación al propietario cuando se pulse el botón; utilice para ello el componente suministrado ConfirmationModal, similar al componente DeleteModal utilizado en clase.

Además, los restaurantes promocionados aparecerán siempre al principio de los listados de restaurantes que se le presentan tanto a los dueños como a los clientes. Además de presentarse al principio, los restaurantes promocionados deben destacarse visualmente, por lo que aparecerá una etiqueta de texto Promoted! con brandSuccess; por su contra, aparecerá Not promoted con brandPrimary.



BACKEND

Añadir la nueva propiedad:

Una propiedad booleana que indica que si el restaurante está promocionado (true) o no promocionado (false), pondremos como valor por defecto false.

Al modelo restaurant:

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

Y al create Restaurant:

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

Ahora veamos las validaciones:

- ***Si el propietario indica que el nuevo restaurante debe estar promocionado, pero ya existían restaurantes promocionados del mismo dueño, al pulsar el botón Save se mostrará un error y no se creará o editará el restaurante.***

Es decir, que solo haya una promocionado:

```
const checkOnlyOnePromoted = (ownerId, promotedValue) => {
  if (promotedValue) {
    try {
      const AllRestaurantsPromoted = Restaurant.findAll(
        { where: { userId: ownerId, promoted: true } })
      if (AllRestaurantsPromoted.length !== 0) {
        return Promise.reject(new Error('Only one restaurant can be
          promoted.'))
      } else {
        return Promise.resolve()
      }
    } catch (err) {
      return Promise.reject(new Error(err))
    }
  }
}
```

Y lo añadimos al create y update:

```
  check('promoted').custom((value, { req }) => {
    return checkOnlyOnePromoted(req.user.id, value)
  }).withMessage('Only one restaurant can be promoted.')
```

Ahora vamos a crear la función del controlador que nos permita promocionar un restaurante:

```
const promote = async function (req, res) {
  const t = await sequelizeSession.transaction()
  try {
    const existingPromotedRestaurant =
      //BUSCAMOS UNO YA PROMOCIONADO
      await Restaurant.findOne(
      { where: { userId: req.user.id, promoted: true } })
      //SI EXISTE
    if (existingPromotedRestaurant) {
      // LO DESPROMOCIONAMOS
      await Restaurant.update(
        { promoted: false },
        { where: { id: existingPromotedRestaurant.id } },
        { transaction: t }
      )
    }
      // PROMOCIONAMOS EL RESTAURANTE QUE QUEREMOS
    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)
  }
}
```

Y por último creamos a ruta:

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

Y como dice que:

Además, los restaurantes promocionados aparecerán siempre al principio de los listados de restaurantes que se le presentan tanto a los dueños como a los clientes.

Tenemos que añadir orden al index e indexOwner:

```
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)
  }
}
```

Y yaaa.

FRONTEND

Tenemos que ir paso a paso:

Empezamos con Restaurant Screen:

Primero añadimos la ruta que recibe la función de promocionar:

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

Y ahora el botón:

- Siempre que haya un botón y un modal, tenemos que hacer un useState:

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

Y añadimos el botón:

```
 <Pressable
            onPress={() => { setRestaurantToBePromoted(item) }}
            style={({ pressed }) => [
              {
                backgroundColor: pressed
                  ? GlobalStyles.brandSecondaryTap
                  : GlobalStyles.brandSecondary
              },
              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>
```

Una vez añadido, tenemos que crear la función que promociona llamando a la del endpoint:

```
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
      })
    }
  }
```

Esta función se llamará en el modal, tenemos que crear el ConfirmationModal, copiando el de Delete Modal:

```
<ConfirmationModal
      isVisible={restaurantToBePromoted !== null}
      onCancel={() => setRestaurantToBePromoted(null)}
      onConfirm={() => promoteRestaurant(restaurantToBePromoted)}>
        <TextRegular>This Restaurant will be promoted</TextRegular>
    </ConfirmationModal>
```

Y ya tendriamos el Modal, falta añadir el texto si se ha promocionado:

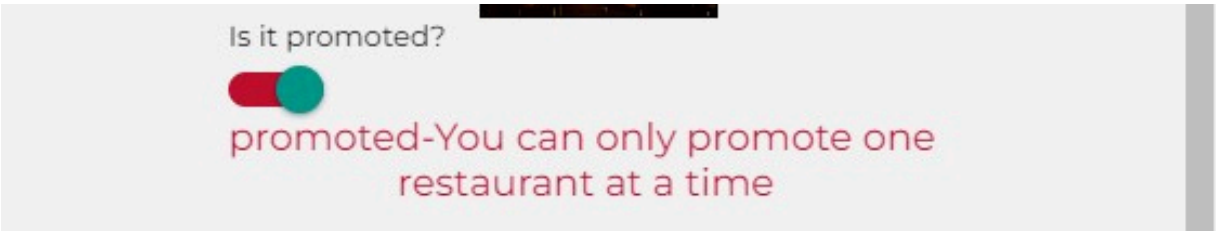
Esta en el mismo view que Shipping, luego añadimos un nuevo view:

```
<View style={[{ flexDirection: 'row', justifyContent: 'space-between',
alignItems: 'flex-end' }]}>
          <TextSemiBold>Shipping: <TextSemiBold textStyle={{ color:
GlobalStyles.brandPrimary
}}>{item.shippingCosts.toFixed(2)}€</TextSemiBold></TextSemiBold>
          {item.promoted &&
              <TextRegular textStyle={[styles.badge, { color:
GlobalStyles.brandPrimary, borderColor: GlobalStyles.brandSuccess }]}>
                ¡En promoción!
              </TextRegular>
          }
        </View>
```

Donde badge:

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

Y ya tendríamos todo, falta el Create Restaurant Screen:



Tenemos que añadir ese switch, que en EditProductScreen esta creado.

Pero antes añadimos la nueva propiedad al const:

```
  const initialRestaurantValues = { name: null, description: null,
address: null, postalCode: null, url: null, shippingCosts: null, email:
null, phone: null, restaurantCategoryId: null, promoted: false }
```

Y añadimos el switch:

```
<TextRegular>Is it promoted?</TextRegular>
              <Switch
                trackColor={{ false: GlobalStyles.brandSecondary, true:
GlobalStyles.brandPrimary }}
                thumbColor={values.promoted ?
GlobalStyles.brandSecondary : '#f4f3f4'}
                // onValueChange={toggleSwitch}
                value={values.promoted}
                style={styles.switch}
                onValueChange={value =>
                  setFieldValue('promoted', value)
                }
              />
              <ErrorMessage name={'promoted'} render={msg =>
<TextError>{msg}</TextError> }/>
```