EXAMEN IISSI SIMULACRO

Enunciado

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 ofrecer a los propietarios la posibilidad de promocionar sus restaurantes. Cada propietario sólo podrá promocionar uno de sus restaurantes.

Un propietario podrá promocionar un restaurante de dos maneras distintas:

- En el formulario de creación 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 propietario, al pulsar el botón `Save` se mostrará un error y no se creará 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 propietarios como a los clientes. Además de presentarse al principio, los restaurantes promocionados deben destacarse visualmente, por lo que aparecerá una etiqueta de texto `¡En promoción!` con el color principal de la marca.

Ejercicio 1

Realice todos los cambios necesarios en el proyecto de backend para implementar el nuevo requisito.

Ejercicio 2

Realice todos los cambios necesarios en el proyecto de frontend para implementar el nuevo requisito.

BACKEND

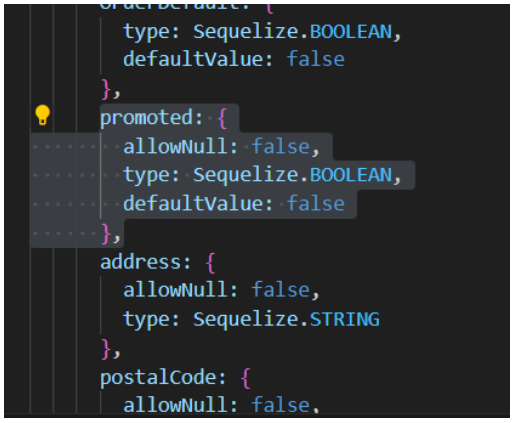
Models, RESTAURANT.JS

```
Restaurant.init({
  name: DataTypes.STRING,
  description: DataTypes.TEXT,
  OrderDefault: {
    type: DataTypes.BOOLEAN,
    defaultValue: false
  },
  promoted: {
    allowNull: false,
    type: DataTypes.BOOLEAN,
    defaultValue: false

  },
  address: DataTypes.STRING,
  postalCode: DataTypes.STRING,
  url: DataTypes.STRING,
  shippingCosts: DataTypes.DOUBLE,
  averageServiceMinutes: DataTypes.DOUBLE,
  email: DataTypes.STRING,
  phone: DataTypes.STRING,
  logo: DataTypes.STRING,
```

Añadimos la propiedad promoted , que no pueda ser false , de tipo Boolean

Migrations CREATE-RESTAURANT



Añadimos la propiedad promoted , con tipo Boolean y el valor por defecto false

Validations RESTAURANT-VALIDATIONS

Para realizar la primera parte del examen que esta subrayado en amarillo, hay que realizar una restricción

```
const noRestaurantPromoted = async (value, { req }) => {
  if (value) {
    try {
      const restaurant = await Restaurant.findOne({ where: { userId: req.user.id, promoted: true } })
      if (restaurant !== null) {
        return Promise.reject(new Error('There is already a promoted restaurant.'))
      } else { return Promise.resolve() }
    } catch (err) {
      return Promise.reject(new Error(err))
    }
  }
}
```

Esta funcion , si han metido un valor (en este caso si han pulsado el boton de promoted a la hora de crear un nuevo restaurante) , busca si hay algun restaurante ya en promocion , si lo hay , te sale un fallo que dice que ya hay otro restaurante promocionado

Si no , se ejecuta con normalidad.

```
check('name').exists().isString().isLength({ min: 1, max:
check('description').optional({ nullable: true, checkFals
check('promoted').custom(noRestaurantPromoted),
check('address').exists().isString().isLength({ min: 1, m
check('postalCode').exists().isString().isLength({ min: 1
```

Añadir en el exports.create , la restriccion

Error: Cuando estaba intentando crear el restaurante dándole a sabe, no me dejaba y me aparecía el error que estaba en el catch de validation. Se solución poniendo bien los imports

```
const { check } = require('express-validator')
const { checkFileIsImage, checkFileMaxSize } = require('./FileValidationHelper')
const maxFileSize = 2000000 // around 2Mb
const Models = require('../../models')
const Restaurant = Models.Restaurant
```

Error2: Aunque no pulsara el botón de promoted al crear un restaurante, es decir aunque no quisiera darle el valor 1 a la propiedad promote, me ejecutaba el validation como si le hubiera dado. Lo solucione , poniendo

If(value){ ←

…

}

Controller RESTAURANT-CONTROLLER

Debemos crear una nueva función , que a la hora de actualizar el valor promoted de un restaurante , si existe otro restaurante del mismo dueño que ya esta promoted , hay que cambiarlo a false , para poner modificar el primero a true

```
exports.changePromoted = async function (req, res) {
  const t = await models.sequelize.transaction()
  try {
    const promotedRestaurant = await Restaurant.findOne({ where: { userId: req.user.id, promoted: true } })

    if (promotedRestaurant) {
      await Restaurant.update(
        { promoted: false },
        { where: { id: promotedRestaurant.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)
  }
}
```

Error: Cuando en el fronted quería imprimir si un restaurante era promoted o no , con el item.promoted , no me dejaba porque no leía el valor promoted.Se soluciona añadiendo al index y al indexOwner el valor promoted .

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

Aquí vemos también , como ordenamos los rest por promoted

Ruta RESTAURANTS-ROUTE

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

El valor patch se refiere a actualizar solo una cosa y no el conjunto entero , es decir se centra en una sola cosa

FRONTED

EndPoint RESTAURANT-ENDPOINTS

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

export { getAll, getDetail, getRestaurantCategories, create, update, remove, ChangePromoted }
```

Importante: /${id}/changePromoted → Debe ser igual que el nombre del controller del backend

SCREEN---------

RESTAURANTS-SCREENS

Lo que esta subrayado en verde en el enunciado se hace aquí.

```
            <TextRegular textStyle={styles.text}>
              Delete
            </TextRegular>
          </View>
        </Pressable>
        <Pressable
          onPress={() => setRestaurantToBePromoted(item) }
          style={({ pressed }) => [
            {
              backgroundColor: pressed
                ? GlobalStyles.brandSuccess
                : GlobalStyles.brandSuccessDisabled
            },
            styles.actionButton
          ]}>
          <View style={[{ flex: 1, flexDirection: 'row', justifyContent: 'center' }]}>
          <MaterialCommunityIcons name='alert-octagram' color={'white'} size={20}/>            <TextRegular textStyle={styles.text}>
              Promoted
            </TextRegular>
          </View>
        </Pressable>
        <View style={{ position: 'absolute', top: -20, right: 10 }}>
          {item.promoted
            ? <TextRegular textStyle={{ color: 'red' }}>Promoted!!</TextRegular>
            : <TextRegular textStyle={{ color: 'red' }}>Not promoted !</TextRegular>
          }
      </View>
        </View>
      </ImageCard>
```

Lo que esta rodeado por lapiz azul es lo que se ve a la derecha arriba , Promoted!! O Not promoted , en rojo

```
<MaterialCommunityIcons name='alert-octagram' color={'white'} size={20}/>
```

Lo que esta subrayado , es el boton que hay que crear , por partes :

```
export default function RestaurantsScreen ({ navigation, route }) {
  const [restaurants, setRestaurants] = useState([])
  const [restaurantToBeDeleted, setRestaurantToBeDeleted] = useState(null)
  const [restaurantToBePromoted, setRestaurantToBePromoted] = useState(null)

  const { loggedInUser } = useContext(AuthorizationContext)

  useEffect(() => {
    if (loggedInUser) {
```

Creamos el restaurantToBePromoted y el setRestaurantToBePromoted

Todo esto porque necesita verificacion

Creamos la funcion ChangePromotedValue, que llama al changePromoted de los endPoints y actualiza los restaurantes por medio de fetchRestaurants

```
  }
}

const ChangePromotedValue = async (restaurant) => {
  try {
    await ChangePromoted(restaurant.id)
    await fetchRestaurants()
    setRestaurantToBePromoted(null)
  } catch (err) {
    setRestaurantToBePromoted(null)
    console.log(err)
  }
}
const removeRestaurant = async (restaurant) => {
  try {
    await remove(restaurant.id)
    await fetchRestaurants()
```

```
      onConfirm={() => removeRestaurant(restaurantToBeDeleted)}>
        <TextRegular>The products of this restaurant will be deleted as well</TextRegular>
        <TextRegular>If the restaurant has orders, it cannot be deleted.</TextRegular>
      </DeleteModal>

      <ConfirmationModal
        isVisible={restaurantToBePromoted !== null}
        onCancel={() => setRestaurantToBePromoted(null)}
        onConfirm={() => ChangePromotedValue(restaurantToBePromoted)}>
          <TextRegular>The products of this restaurant will be promoted as well</TextRegular>

      </ConfirmationModal>
    </>
  )
}

const styles = StyleSheet.create({
  container: {
    flex: 1
  },
  button: {
```

Por medio del CorfimationModal (clase ya creada) , realizamos esto para configurarla con el restaurantToBePromoted y el

setRestaurantToBePromoted, el texto que añadamos es lo que se leerá en la ventana emergente

CreateRestaurantScreen

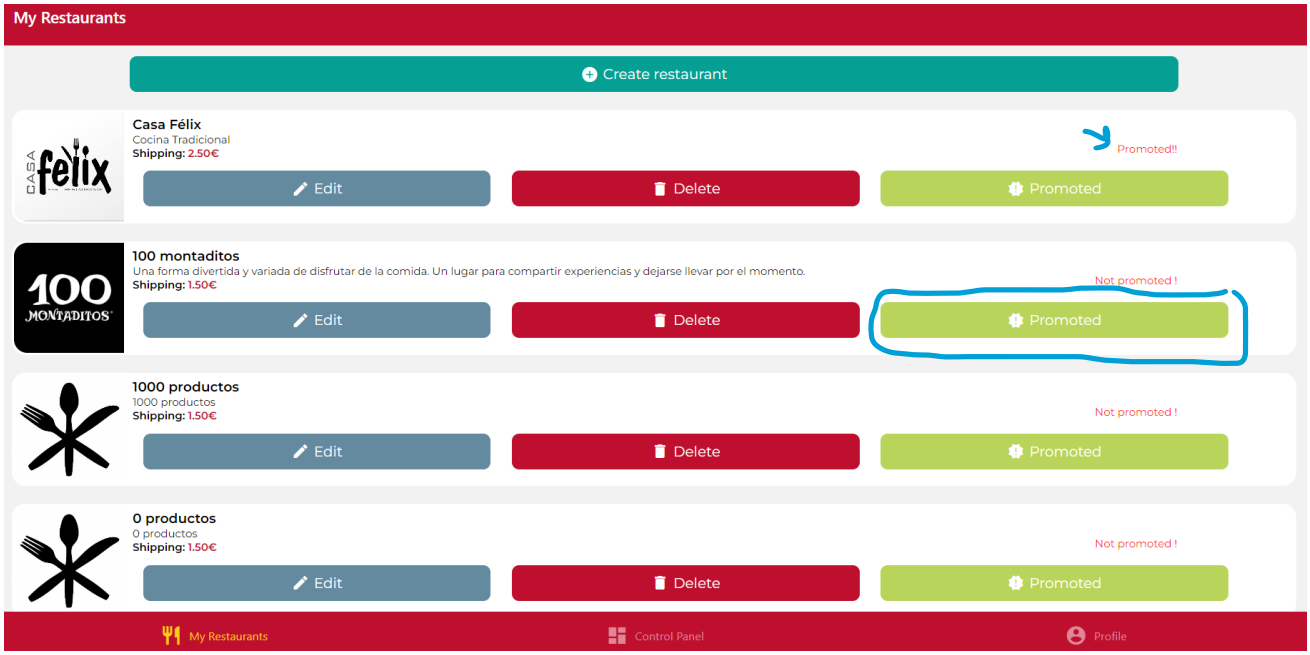
```
    backendErrors.map((error, index) => <TextError key={index}>{error.param}-{error.msg}</TextErr
  }
  <TextRegular style={styles.switch}>Promoted ?</TextRegular>
  <Switch
    trackColor={{ false: GlobalStyles.brandSecondary, true: GlobalStyles.brandPrimary }}
    thumbColor={values.promoted ? GlobalStyles.brandSecondary : '#f4f3f4'}
    value={values.promoted}
    style={styles.switch}
    onValueChange={value =>
      setFieldValue('promoted', value)
    }
  />
  <Pressable
    onPress={handleSubmit}
    style={({ pressed }) => [
```

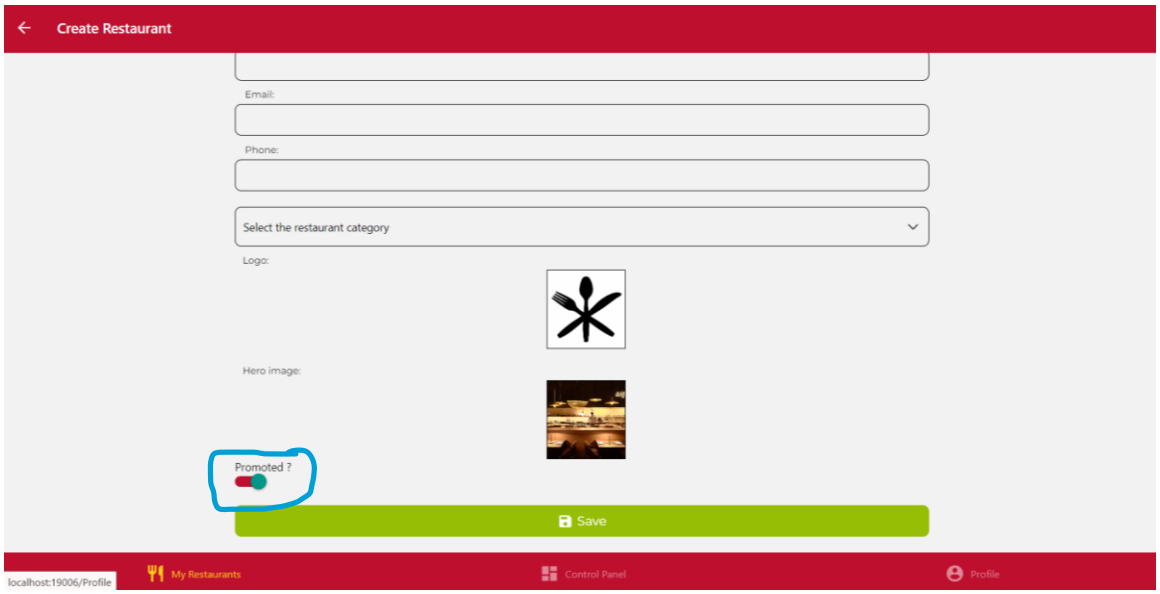
Añadimos el switch, que solo hace falta que se cambien las propiedades , es decir donde pone ‘promoted’ se cambia por la propiedad boolean que se quiera modificar

Copiar switch

```
<Switch
                  trackColor={{ false: GlobalStyles.brandSecondary, true:
GlobalStyles.brandPrimary }}
                  thumbColor={values.promoted ?
GlobalStyles.brandSecondary : '#f4f3f4'}
                  value={values.promoted}
                  style={styles.switch}
                  onValueChange={value =>
                    setFieldValue('promoted', value)
                  }
                />
```

SOLUCION




Create Restaurant
Email:
Phone:
Select the restaurant category
Logo:
Hero image:
Promoted ?
Save
localhost:19006/Profile
My Restaurants
Control Panel
Profile