EXAMEN PINNED- JUNIO 2024

Después del lanzamiento inicial de DeliverUS, los inversores solicitaron una nueva función que permita a los propietarios fijar sus restaurantes. Cada propietario puede marcar tantos restaurantes como desee.

Un propietario puede fijar restaurantes de dos maneras diferentes:

- **En el formulario de creación de restaurante. De forma predeterminada, no se fijará, pero el propietario puede optar por fijarlo. Para hacer esto, Switch se debe proporcionar un que funcione con una propiedad llamada pinned. Si Switch está marcado, el restaurante debe crearse como fijado. El backend espera que la pinned propiedad sea booleana y opcional. Si la propiedad no está presente, se debe crear como no fijada.**
- **En la pantalla "Mis Restaurantes", a través de un icono que actuará como botón y se mostrará al lado de cada restaurante. Al hacer clic en él, el restaurante quedará fijado o desanclado. La aplicación debe pedir confirmación al propietario cuando se presiona el botón: use el componente ConfirmationModal proporcionado, similar al DeleteModal componente usado en clase. El sistema informará al usuario si el restaurante ha sido fijado o desanclado.**

Finalmente, los restaurantes fijados siempre aparecerán en la parte superior de las listas de restaurantes presentadas a su propietario y estarán ordenados por la fecha en que fueron fijados (los más antiguos primero), seguidos de los que no están fijados.

Tarea 1

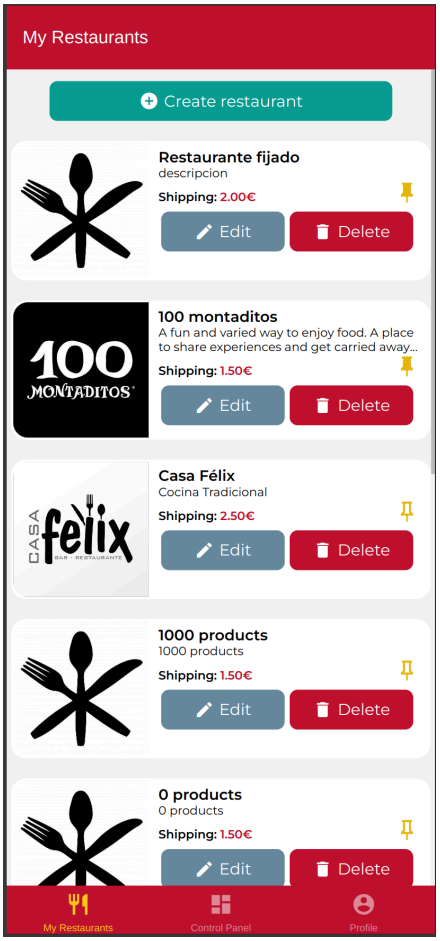
Realice todos los cambios necesarios en el proyecto backend para implementar el nuevo requisito. Las pruebas de backend esperan que la ruta sea: PATCH /restaurants/:restaurantId/togglePiny que los restaurantes tengan una nueva propiedad llamada pinnedAt

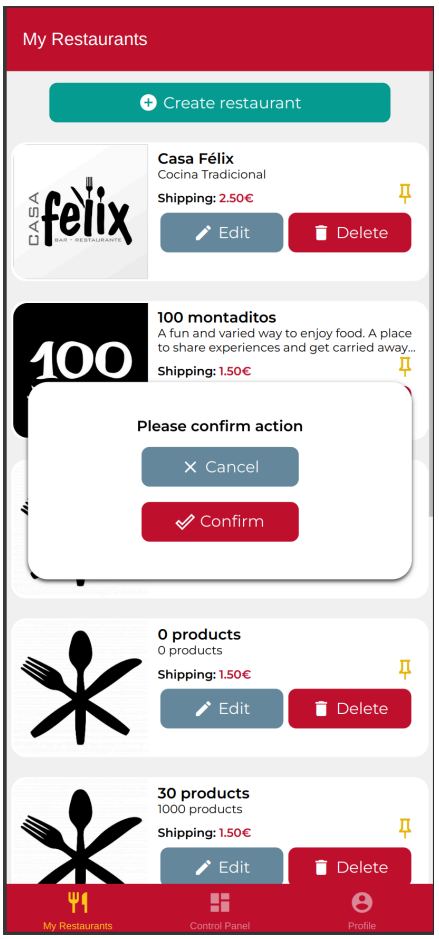
Tarea 2

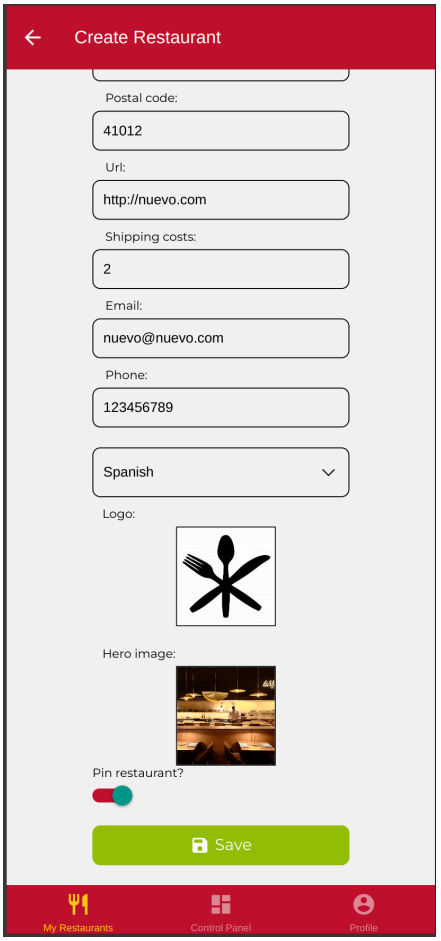
Realice todos los cambios necesarios en el proyecto frontend para implementar el nuevo requisito.

Puedes renderizar el icono de pin propuesto con

```
<MaterialCommunityIcons
   name={item.pinnedAt ? 'pin' : 'pin-outline'}
   color={GlobalStyles.brandSecondaryTap}
   size={24}
/>
```

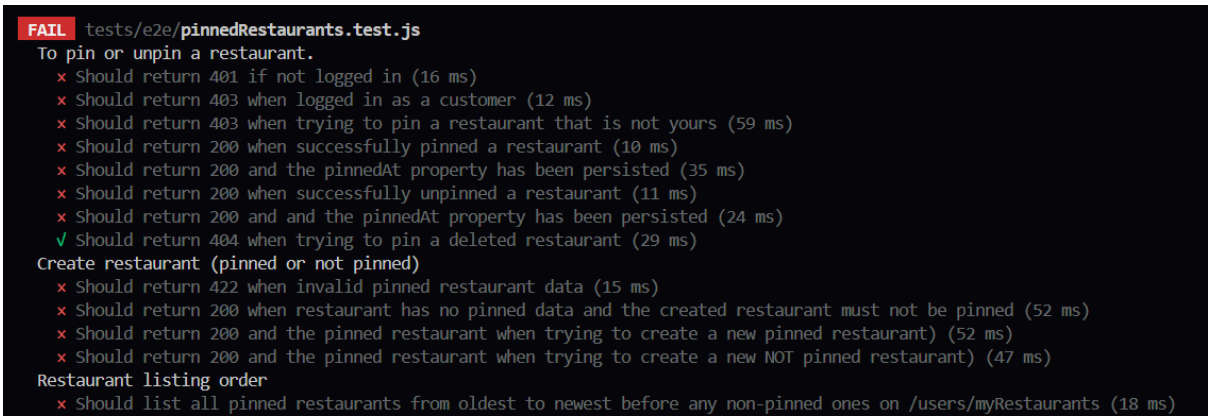

My Restaurants
Create restaurant
Restaurante fijado
descripcion
Shipping: 2.00€
Edit
Delete
100 montaditos
A fun and varied way to enjoy food. A place to share experiences and get carried away...
Shipping: 1.50€
Edit
Delete
Casa Félix
Cocina Tradicional
Shipping: 2.50€
Edit
Delete
1000 products
1000 products
Shipping: 1.50€
Edit
Delete
0 products
0 products
Shipping: 1.50€
Edit
Delete
My Restaurants
Control Panel
Profile


My Restaurants
Create restaurant
Casa Félix
Cocina Tradicional
Shipping: 2.50€
Edit
Delete
100 montaditos
A fun and varied way to enjoy food. A place to share experiences and get carried away...
Shipping: 1.50€
Please confirm action
Cancel
Confirm
0 products
0 products
Shipping: 1.50€
Edit
Delete
30 products
1000 products
Shipping: 1.50€
Edit
Delete
My Restaurants
Control Panel
Profile


Create Restaurant
Postal code:
41012
Url:
http://nuevo.com
Shipping costs:
2
Email:
nuevo@nuevo.com
Phone:
123456789
Spanish
Logo:
Hero image:
Pin restaurant?
Save
My Restaurants
Control Panel
Profile

BACKEND

Ejecutar el test:



PASO 1 - ¿QUÉ NOS ESTÁN PIDIENDO?

ENUNCIADO: Realice todos los cambios necesarios en el proyecto backend para implementar el nuevo requisito. Las pruebas de backend esperan que la ruta sea: PATCH /restaurants/:restaurantId/togglePiny que los restaurantes tengan una nueva propiedad llamada ***pinnedAt***

- Nueva propiedad: pinnedAt
- De tipo date
- En el modelo de Restaurant

PASO 2 - AÑADIR LA NUEVA PROPIEDAD

Vamos a poner en el modelo de Restaurant la nueva propiedad:

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

Y en la migrations de Restaurant:

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

PASO 3 - LEER BIEN LOS TEST PARA BUSCAR SOLUCIÓN

To pin or unpin a restaurant.
  √ Should return 401 if not logged in (11 ms)
  √ Should return 403 when logged in as a customer (13 ms)
  √ Should return 403 when trying to pin a restaurant that is not yours (55 ms)
  √ Should return 200 when successfully pinned a restaurant (17 ms)
  √ Should return 200 and the pinnedAt property has been persisted (31 ms)
  √ Should return 200 when successfully unpinned a restaurant (16 ms)
  √ Should return 200 and and the pinnedAt property has been persisted (22 ms)
  √ Should return 404 when trying to pin a deleted restaurant (27 ms)

Estas se arreglan en la propia ruta:
  √ Should return 401 if not logged in (11 ms)
  √ Should return 403 when logged in as a customer (13 ms)
  √ Should return 403 when trying to pin a restaurant that is not yours (55 ms)

```
app.route('/restaurants/:restaurantId/togglePinned')
    .patch(
      isLoggedIn, //Should return 401 if not logged in
      hasRole('owner'),//Should return 403 when logged in as a customer
      checkEntityExists(Restaurant, 'restaurantId'), // Should return
403 when trying to pin a restaurant that is not yours
      RestaurantMiddleware.checkRestaurantOwnership,
 )
```

Estas son la función que hace pinnear un restaurante:

√ Should return 200 when successfully pinned a restaurant (17 ms)
√ Should return 200 and the pinnedAt property has been persisted (31 ms)
√ Should return 200 when successfully unpinned a restaurant (16 ms)
√ Should return 200 and and the pinnedAt property has been persisted (22 ms)

```
const togglePinned = async function (req, res) {
 try {
  const restaurant = await Restaurant.findByPk(req.params.restaurantId)
    await Restaurant.update(
      { pinnedAt: restaurant.pinnedAt ? null : new Date() },
      { where: { id: restaurant.id } }
    )
  const updatedRestaurant = await
      Restaurant.findByPk(req.params.restaurantId)
    res.json(updatedRestaurant)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Con transacciones:

```
const pin = async function (req, res) {
  const t = await sequelizeSession.transaction()
  try {
    const restaurantPinned = await
Restaurant.findByPk(req.params.restaurantId)
    await Restaurant.update(
      { pinnedAt: restaurantPinned.pinnedAt ? null : new Date() },
      { where: { id: restaurantPinned.id } },
      { 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 añadiremos la nueva función a la ruta.
Restaurant listing order
√ Should list all pinned restaurants from oldest to newest before any non-pinned ones on /users/myRestaurants (16 ms)

```
app.route('/users/myRestaurants')
    .get(
      isLoggedIn,
      hasRole('owner'),
      RestaurantController.indexOwner)
```

Tenemos que editar el indexOwner para que muestre primero los pinneados ordenados por fecha de pinneacion y despues los no pinneados:

Hacemos una función que devuelve una array con los restaurantes que están pinneados y ordenados por fecha:

```
async function getPinnedRestaurants (req) {
  return await Restaurant.findAll({
    attributes: { exclude: ['userId'] },
    where: {
      userId: req.user.id, // Filter by the authenticated user's userId
      pinnedAt: {
        [Sequelize.Op.not]: null // Filter by pinnedAt not being null
      }
    },
    order: [['pinnedAt', 'ASC']], // Order the results by pinnedAt in
ascending order
    include: [       {
        model: RestaurantCategory,
        as: 'restaurantCategory'
      }
    ]
  })
}
```

Y otra que devuelva un array con los restaurantes no pinneados:

```
async function getNotPinnedRestaurants (req) {
  return await Restaurant.findAll({
    attributes: { exclude: ['userId'] },
    where: {
      userId: req.user.id,
      // The `pinnedAt` attribute of the restaurant is null
      pinnedAt: null // Filtrar por 'pinnedAt' nulo
    },
    include: [{
      model: RestaurantCategory,
      as: 'restaurantCategory'
    }]
  })
}
```

Y por último añadimos estas dos funciones al indexOwner:

Para concatenar dos arrays [...ARRAY1, …ARRAY2]

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

Create restaurant (pinned or not pinned)
  √ Should return 200 when restaurant has no pinned data and the created restaurant must not be pinned (54 ms)
  √ Should return 200 and the pinned restaurant when trying to create a new pinned restaurant) (52 ms)
  √ Should return 200 and the pinned restaurant when trying to create a new NOT pinned restaurant) (63 ms)

Tenemos que modificar la función create de Restaurant para los errores 20X:

```
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)
  }
}
```

Si tmb fuera update:

```
const update = async function (req, res) {
  try {
    //  req.body.pinnedAt = req.body.pinned ? new Date() : null
    await Restaurant.update(req.body, { where: { id:
req.params.restaurantId } })
    const updatedRestaurant = await
Restaurant.findByPk(req.params.restaurantId)
    res.json(updatedRestaurant)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Y para el error 422, tenemos que validar los datos de entrada en la función create:
√ Should return 422 when invalid pinned restaurant data (12 ms)

```
 it('Should return 422 when invalid pinned restaurant data', async ()
=> {
    const notValidRestaurant = { ...bodeguitaRestaurant }
    notValidRestaurant.pinned = 'invalidPinnedValue'
    const response = await
request(app).post('/restaurants').set('Authorization', `Bearer
${owner.token}`).send(notValidRestaurant)
    expect(response.status).toBe(422)
    const errorFields = response.body.errors.map(error => error.param)
    expect(['pinned'].every(field =>
errorFields.includes(field))).toBe(true)
  })
```

SOL:

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

Y acabamos con la ruta:

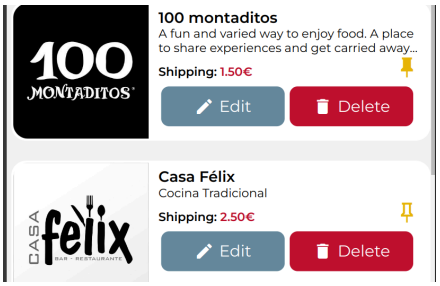
```
  app.route('/restaurants/:restaurantId/togglePinned')
    .patch(
      isLoggedIn,
      hasRole('owner'),
      checkEntityExists(Restaurant, 'restaurantId'),
      RestaurantMiddleware.checkRestaurantOwnership,
      RestaurantController.pin)
```

FRONTEND

Primero añadimos la nueva ruta:

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

Vamos a añadir el boton:



Cómo va a haber un modal, creamos el const:

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

Añadimos el boton:

```
 <View style={{ flexDirection: 'row', justifyContent: 'space-between',
alignItems: 'flex-end' }} >
        <TextSemiBold>Shipping: <TextSemiBold textStyle={{ color:
GlobalStyles.brandPrimary
}}>{item.shippingCosts.toFixed(2)}€</TextSemiBold></TextSemiBold>

        <Pressable
            onPress={() => { setRestaurantToBePinned(item) }}>
            <MaterialCommunityIcons
              name={item.pinnedAt ? 'pin' : 'pin-outline'}
              color={GlobalStyles.brandSecondaryTap}
              size={24}
            />
          </Pressable>
        </View>
```

La funcion que recibe la funcion del endpoint:

```
const pinnRestaurant = 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
      })
    }
  }
```

Y el modal que recibe esa función:

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

Tenemos que añadir el switch:

En CreateRestaurantScreen añadiremos el nuevo switch:

Primero el const:

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

Y en el validationEschema no tenemos que hacer nada porque no estamos introduciendo nada

En el return pondremos:

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