

EXAMEN VISIBILIDAD - JUNIO 2024

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 establecer un momento en el que los productos dejarán de ser visibles (`visibleUntil`).

Un propietario podrá establecer este momento al crear o actualizar un producto con dos escenarios posibles:

- Por defecto, este momento será nulo, por lo que se considera que siempre estará visible.
- Si un propietario establece este momento, el producto solo estará visible hasta la fecha (inclusive).

Además se deben cumplir las siguientes reglas de negocio:

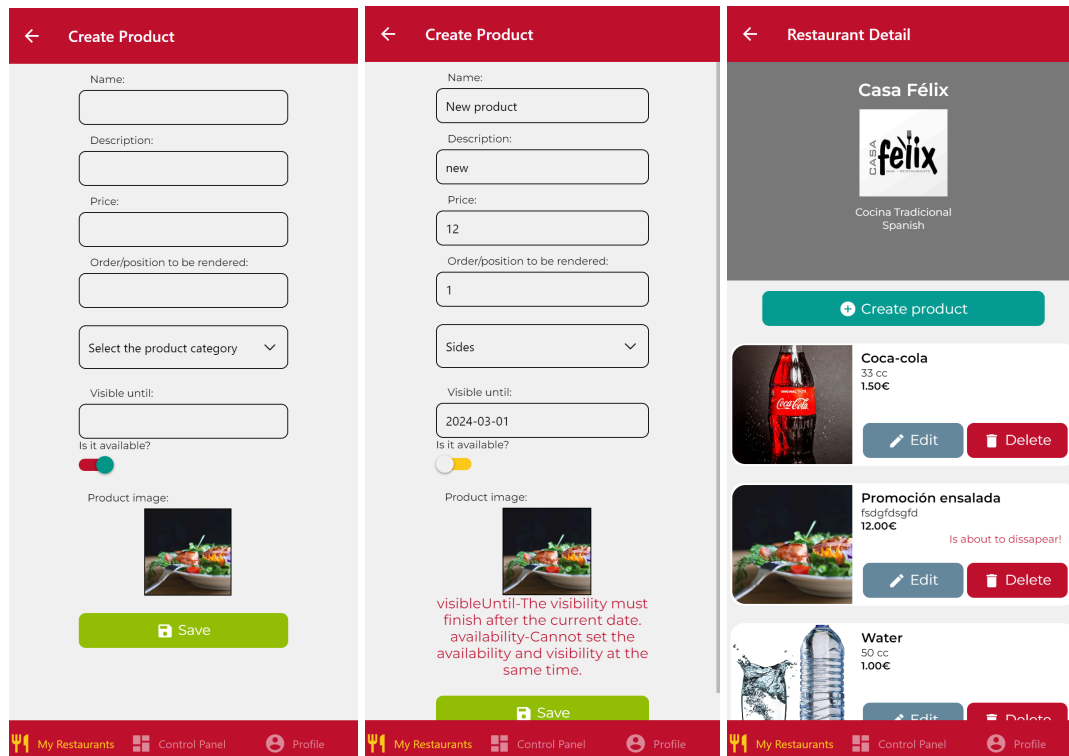
- Un propietario no podrá establecer un momento de fin anterior al momento actual.
- Un propietario no podrá establecer un producto como no disponible y, al mismo tiempo, un momento de fin.

Finalmente, los productos que estén a una semana o menos de desaparecer aparecerán en la interfaz marcados.

Ejercicio 1: Realice todos los cambios necesarios en el proyecto de backend para implementar el nuevo requisito asegurándose de que los test se ejecutan correctamente.

Ejercicio 2: Realice todos los cambios necesarios en el proyecto de frontend para implementar el nuevo requisito.

Puede renderizar el icono de fijado propuesto con:



BACKEND

Nada más empezar ejecutar el test:

```
FAIL tests/e2e/productsVisibility.test.js
  Get restaurant products
    ✗ Not visible products should not be listed (38 ms)
    ✓ Visible products should be listed (26 ms)
  Create product
    ✗ Should return 422 when visibleUntil is before today (21 ms)
    ✗ Should return 422 when setting availability to false and visibleUntil at the same time (15 ms)
    ✗ Should return 422 when visibleUntil is not a date (14 ms)
  Edit product
    ✗ Should return 422 when visibleUntil is before today (14 ms)
    ✗ Should return 422 when setting availability to false and visibleUntil at the same time (15 ms)
    ✗ Should return 422 when visibleUntil is not a date (17 ms)
```

PASO 1 - ¿QUÉ NOS ESTÁN PIDIENDO?

*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 establecer un **momento** en el que los productos dejarán de ser visibles (**visibleUntil**).*

- Nueva propiedad: visibleUntil
- De tipo date
- En el modelo de products

PASO 2 - AÑADIR LA NUEVA PROPIEDAD

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

```
visibleUntil: DataTypes.DATE
```

Y luego en database > migrations > createProduct:

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

PASO 3 - LEER BIEN LOS TEST PARA BUSCAR SOLUCIÓN

Primer test que da error:

× Not visible products should not be listed (41 ms)

Si nos vamos a su test:

```
it('Not visible products should not be listed', async () => {
  const notVisibleProduct = await
  getNewPaellaProductData(restaurant)
  const visibleDate = new Date()

  visibleDate.setDate(visibleDate.getDate() - 1)

  notVisibleProduct.visibleUntil = visibleDate

  let newProduct = Product.build(notVisibleProduct)
  newProduct = await newProduct.save()

  const responseRestaurant = await
  request(app).get(`/restaurants/${restaurant.id}`).send()
  expect(responseRestaurant.status).toBe(200)

  expect(responseRestaurant.body.products.every(product =>
  product.id !== newProduct.id)).toBe(true)
})
```

Vemos que el error está en la ruta:

```
.get(`/restaurants/${restaurant.id}`)
```

Si nos vamos a esa ruta:

```
app.route('/:restaurantId')
  .get(
    checkEntityExists(Restaurant, 'restaurantId'),
    RestaurantController.show)
```

Tendremos que cambiar algo de la función show del controlador, para que los productos que no sean visible no se muestren.

La función show del controlador de restaurant está así:

```
const show = async function (req, res) {
  // Only returns PUBLIC information of restaurants
  try {
    const restaurant = await
      Restaurant.findByPk(req.params.restaurantId, {
        attributes: { exclude: ['userId'] },
        include: [{
          model: Product,
          as: 'products',
          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)
  }
}
```

Tendremos que filtrar (where) para que muestre solo restaurantes con la propiedad visibleUntil:

ENUNCIADO:

- Por defecto, este momento será nulo, por lo que se considera que siempre estará visible.
- Si un propietario establece este momento, el producto solo estará visible hasta la fecha (inclusive).

Tendremos que mostrar los restaurantes en esos caso, que son:

- Productos que están siempre disponibles (donde visibleUntil es null).
- Productos cuyo tiempo de visibilidad todavía no ha expirado (donde visibleUntil es mayor que la fecha actual)

Luego puede tomar dos valores, sería un OR:

VER: <https://sequelize.org/docs/v6/core-concepts/model-querying-basics/>

visibleUntil = null OR visibleUntil > fecha actual

Quedando:

```
const show = async function (req, res) {
  // Only returns PUBLIC information of restaurants
  try {
    const restaurant = await
Restaurant.findByPk(req.params.restaurantId, {
  attributes: { exclude: ['userId'] },
  include: [{
    model: Product,
    as: 'products',
    where: {
      // visibleUntil = null OR visibleUntil > fecha actual
      visibleUntil: { [Sequelize.Op.or]: [
        { [Sequelize.Op.eq]: null },
        { [Sequelize.Op.gt]: new Date() }
      ]
    },
  },
  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)
  }
}
```

Si ejecutamos el test:

```
FAIL tests/e2e/productsVisibility.test.js
Get restaurant products
  ✓ Not visible products should not be listed (34 ms)
  ✓ Visible products should be listed (15 ms)
Create product
  ✗ Should return 422 when visibleUntil is before today (21 ms)
  ✗ Should return 422 when setting availability to false and visibleUntil at the same time (16 ms)
  ✗ Should return 422 when visibleUntil is not a date (41 ms)
Edit product
  ✗ Should return 422 when visibleUntil is before today (15 ms)
  ✗ Should return 422 when setting availability to false and visibleUntil at the same time (14 ms)
  ✗ Should return 422 when visibleUntil is not a date (22 ms)
```

Los siguientes errores estan en el create y edit product, al ser errores 422 sabemos que son de los Validations, aun así, si vemos los test, vemos que la ruta siempre es:

```
.post('/products')
```

En esa ruta:

```
app.route('/products')
  .post(
    isLoggedIn,
    hasRole('owner'),
    handleFilesUpload(['image'], process.env.PRODUCTS_FOLDER),
    ProductValidation.create,
    handleValidation,
    ProductMiddleware.checkProductRestaurantOwnership,
    ProductController.create
  )
```

Tendremos que validar los datos en `ProductValidation.create`, y al ser los test iguales para post y put, en ambas funciones tendremos que poner lo mismo.

Dice:

× **Should return 422 when visibleUntil is before today**

```
it('Should return 422 when visibleUntil is before today', async ()
=> {
  const invalidProduct = { ...productData }
  const visibleUntil = new Date()

  visibleUntil.setDate(visibleUntil.getDate() - 1)

  invalidProduct.visibleUntil = visibleUntil
  const response = await
request(app).post('/products').set('Authorization', `Bearer
${owner.token}`).send(invalidProduct)
  expect(response.status).toBe(422)
  const errorFields = response.body.errors.map(error =>
error.param)
  expect(['visibleUntil'].some(field =>
errorFields.includes(field))).toBe(true)
})
```

Luego tenemos que verificar la propiedad: `['visibleUntil']`

Tenemos que verificar que si la propiedad es anterior a hoy salte un error, en javascript para declarar una variable con la fecha actual vale con poner: `new Date()`, que sería equivalente al `LocalDate.now()` de java.

Además, si queremos verificar esto, la propiedad debe estar activada:

```
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() }
})
```

Dice:

× Should return 422 when visibleUntil is not a date (39 ms)

```
it('Should return 422 when visibleUntil is not a date', async () =>
{
    const invalidProduct = { ...productData }

    invalidProduct.visibleUntil = 'BADFORMAT'
    const response = await
request(app).post('/products').set('Authorization', `Bearer
${owner.token}`).send(invalidProduct)
    expect(response.status).toBe(422)
    const errorFields = response.body.errors.map(error =>
error.param)
    expect(['visibleUntil'].every(field =>
errorFields.includes(field))).toBe(true)
})
afterAll(async () => {
    await shutdownApp()
})
```

Luego tenemos que verificar la propiedad: `['visibleUntil']`

Tenemos que verificar que el parámetro sea una fecha, y si no salta un error:

```
check('visibleUntil').optional().isDate().toDate(),
```

Hacemos el toDate porque cuando metemos el parámetro lo ponemos como un String.

Dice:

× Should return 422 when setting availability to false and visibleUntil at the same time

```
it('Should return 422 when setting availability to false and
visibleUntil at the same time', async () => {
  const invalidProduct = { ...productData }

  const visibleUntil = new Date()

  visibleUntil.setDate(visibleUntil.getDate() + 1)

  invalidProduct.visibleUntil = visibleUntil

  invalidProduct.availability = false

  const response = await
request(app).post('/products').set('Authorization', `Bearer
${owner.token}`).send(invalidProduct)
  expect(response.status).toBe(422)
  const errorFields = response.body.errors.map(error =>
error.param)
  expect(['availability'].every(field =>
errorFields.includes(field))).toBe(true)
})
```

Luego tenemos que verificar la propiedad: `['availability']`

Tenemos que verificar que el valor de availability no este activado cuando el parámetro visibleUntil lo este:

```
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() }
})
```

Quedando al final en create y update, al final de la función:


```
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() }
})
})
```

Y si volvemos a ejecutar los test:

```
PASS tests/e2e/productsVisibility.test.js
Get restaurant products
  ✓ Not visible products should not be listed (31 ms)
  ✓ Visible products should be listed (15 ms)
Create product
  ✓ Should return 422 when visibleUntil is before today (15 ms)
  ✓ Should return 422 when setting availability to false and visibleUntil at the same time (11 ms)
  ✓ Should return 422 when visibleUntil is not a date (13 ms)
Edit product
  ✓ Should return 422 when visibleUntil is before today (11 ms)
  ✓ Should return 422 when setting availability to false and visibleUntil at the same time (13 ms)
  ✓ Should return 422 when visibleUntil is not a date (12 ms)
```

FRONTEND

Primero tenemos que añadir el hueco de Visible Until en la pantalla createProduct:

A screenshot of a form input field. The label 'Visible until:' is positioned above a rounded rectangular input box. The entire form element is set against a light gray background.

Añadimos la propiedad en el const:

```
const initialProductValues = { name: null, description: null, price: null, order: null, restaurantId: route.params.id, productCategoryId: null, availability: true, visibleUntil: null }
```

Y luego en el validationSchema:


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

Luego creamos el mismo hueco como estan los demás:

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

Y ya

Luego en EditProduct:


 **Edit Product**

Name:

Description:

Price:

Order/position to be rendered:


Starters 


Visible until:




Is it available?

☒

Product image:



 Save

 My Restaurants  Control Panel  Profile

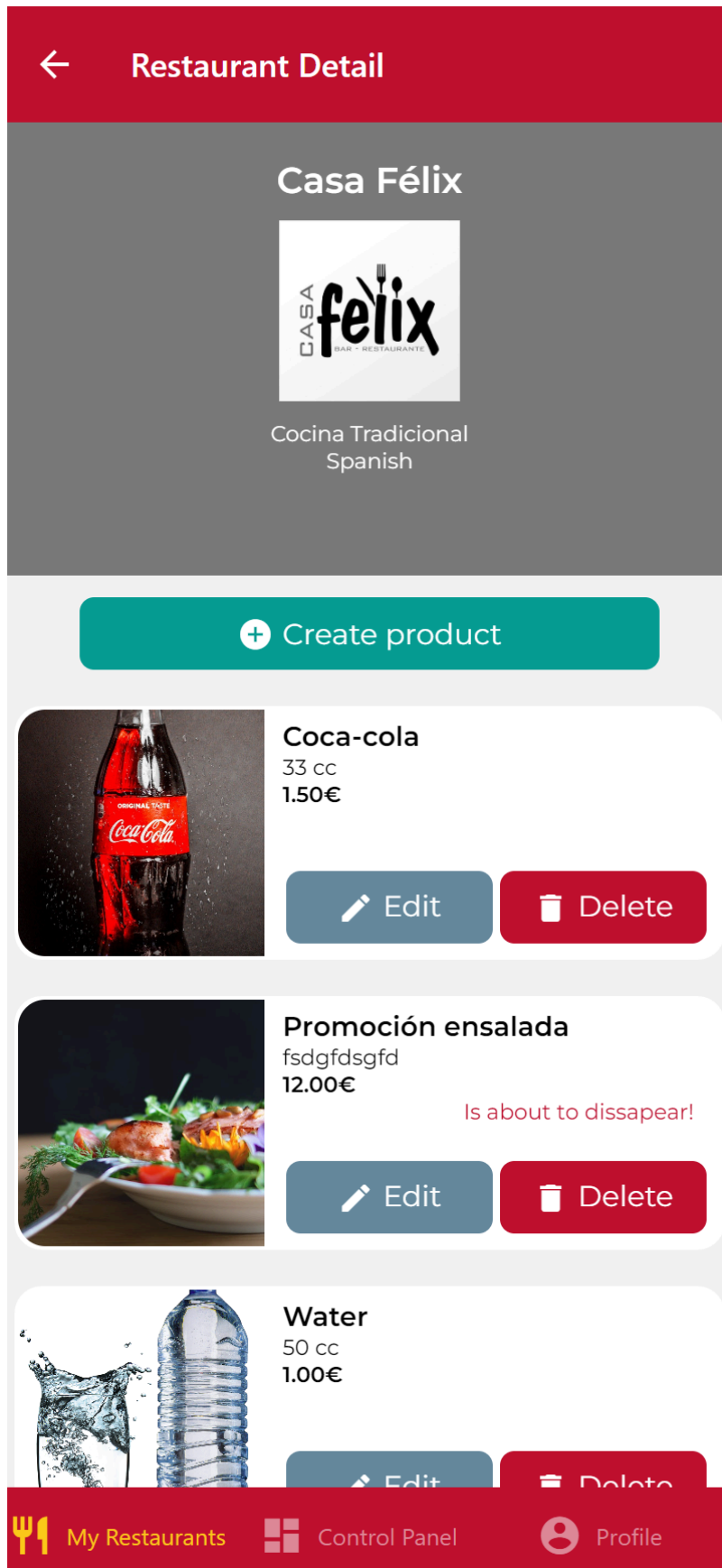
Tenemos que hacer lo mismo:

```
const [initialProductValues, setInitialProductValues] = useState({
  name: null, description: null, price: null, order: null,
  productCategoryId: null, availability: null, image: null, visibleUntil:
  null })
```

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

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

Y por ultimo en RestaurantDetailScreen:



Tenemos que añadir ese texto en función de que quede menos de una semana para que deje de estar visible:

Creamos una función que calcule los días que quedan y devuelva un booleano si quedan menos de 7:

```
const BooleanMenosDe7DiasParaDesaparecer = (date) => {  
  const hoy = new Date()  
  const fecha = new Date(date)  
  const diferencia = fecha.getTime() - hoy.getTime()  
  
  return Math.floor(diferencia / (1000 * 60 * 60 * 24)) <= 7  
}
```

Extra:

```
const daysLeft = (date) => {  
  const hoy = new Date()  
  const fecha = new Date(date)  
  const diferencia = fecha.getTime() - hoy.getTime()  
  
  return Math.floor(diferencia / (1000 * 60 * 60 * 24))  
}
```

Y luego añadimos en el renderProduct la nueva función:

```
{item.visibleUntil &&  
BooleanMenosDe7DiasParaDesaparecer(item.visibleUntil) &&  
  <TextRegular textStyle={styles.visible}>Is about to disappear  
in {daysLeft(item.visibleUntil)} days!</TextRegular>  
}
```