Hacer Testing en node.js

Realizar testing mediante *mocha*, *chai* y *supertest*

1. En la carpeta identificada con el nombre **test**, crear un file con la extensión `.js`, con el nombre del ***recurso*** que voy a probar.
2. Relevar todos los requerimientos, de acuerdo a cada validación implementada en el recurso, por ejemplo:

   > ```
   > Books --> 1er describe
   >         Alta de libros --> 2do describe
   >         Modificación de libros --> 2do describe
   >         Eliminar libros --> 2do describe
   >         Obtener libros --> 2do describe
   > ```

3. Instalar desde la consola, todas las dependencias para hacer los tests, mediante el siguiente comando:

```
npm i mocha chai supertest
```

4. Estructurar el grupo de pruebas, por ejemplo:

```
describe('Books', funtion() {
                describe('Alta de libros', funtion() {
                        it('Requiere un usuario que exista en la DB', funtion() {

                        })

                        it('Requiere un autor que exista en la DB', funtion() {

                        })

                        it('Requiere una categoría que exista en la DB', funtion()
{

                        })

                        it('Requiere un lenguaje que exista en la DB', funtion() {

                        })
```

```
                    it('Requiere un año de edición que sea válido', funtion() {

                    })

                    it('Requiere un título válido para el libro', funtion() {

                    })

                    it('Requiere una sinopsis válida para el libro', funtion()
{

                    })

                    it.skip('Requiere un valor decimal válido para el precio de
alquiler del libro', funtion() {

                    })
                })
            })
```

5. En el recurso que estemos testeando, importamos las siguientes librería de **Chai**, **Supertest** y la **app**, haciendo destructuring de los mismos:

```
const { assert } = require('chai')
const request = require('supertest')
const app = require('../app')
```

6. Para ejecutar los tests, basta con ejecutar el comando `mocha` desde la consola. En el caso de necesitar ejecutar un único archivo de test particular, ejecutar el comando:

```
npm test --file test/books.js
```

7. En caso de que haga falta, definir las funciones **before** y **after**:

```
before(function() {

})

after(function() {

})
```

8. En caso de que haga falta, definir las funciones **beforeEach** y **afterEach**:

```
beforeEach(function() {

})
```

```
afterEach(function() {

})
```

9. Ejemplo de una estructura de prueba:

```
const { assert } = require('chai')
const request = require('supertest')
const app = require('../app')
const { Device, PhoneNumber } = require('../db/models')

let phoneId;
const AVAILABLE = 1;
const NOT_AVAILABE = 2;
let phoneNumbersFiltered;

const deleteAssociation = async (id) => {
    let deviceToDelete = await Device.findOne({
        where: {
            id
        }
    })
    phoneId = deviceToDelete.phoneNumberId;
    console.log('ID del phone number a modificar: ' + phoneId);
    return await deviceToDelete.destroy();
}


const setAvailablePhoneNumber = async (phoneNumberId) => {
    return await PhoneNumber.update(
        {
            availabilityId: AVAILABLE
        },
        {
            where: {
                id: phoneNumberId
            }
        }
    )
}


describe('Tests de Dispositivos', function() {

        after(async function() {
        for(let i = 0; i < phoneNumbersFiltered.length; i++) {
            await phoneNumbersFiltered[i].update(
                {
                    availabilityId: AVAILABLE
                }
            )
        }
    })
```

```
    it('Que al dar de alta un dispositivo bloqueado, falla el intento y responder
con un error', async function() {
        return request(app)
        .post('/associations')
        .send(
            {
                brand: "Xiaomi",
                model: "R45",
                imei: "010929/00/389023/2",
                locked: 1
            }
        )
        .expect(400)
        .then(async res => {
            assert.equal(res.body.message, 'DEVICE IS LOCKED')
        })
    })


    it('Que al dar de alta un dispositivo, si todo es correcto, responde con codigo
201', async function() {
        return request(app)
        .post('/associations')
        .send(
            {
                brand: "Apple",
                model: "R22",
                imei: "010948/00/389023/2",
                locked: 0
            }
        )
        .expect(201)
        .then(async (res) => {
            assert.isObject(res.body, 'El response del endpoint no es un objeto')
            await deleteAssociation(res.body.id);
            await setAvailablePhoneNumber(phoneId);
        })
    })


    it('Que si no hay lineas disponibles, falla y responde con error', async
function() {

        let phoneNumbers = await PhoneNumber.findAll();
        phoneNumbersFiltered = phoneNumbers.filter(phone => phone.availabilityId
=== AVAILABLE)
        for(let i = 0; i < phoneNumbersFiltered.length; i++) {
            await phoneNumbersFiltered[i].update(
                {
                    availabilityId: NOT_AVAILABE
                }
            )
        }

        return request(app)
```

```
        .post('/associations')
        .send(
            {
                brand: "Apple",
                model: "R22",
                imei: "010948/00/389023/2",
                locked: 0
            }
        )
        .expect(400)
        .then(async (res) => {
            assert.equal(res.body.message, 'NO PHONE NUMBERS AVAILABLE')
        })
    })
})
```