Segundo Proyecto

Este proyecto está diseñado para gestionar productos, tanto perecederos como no perecederos, utilizando Node.js, Express y Sequelize con una base de datos MySQL. A continuación, se detallan los pasos para configurar y entender el proyecto, ideal para alguien que recién se incorpora al equipo.

**Requisitos Previos**

- Tener Node.js y npm instalados en tu sistema.
- Asegúrate de tener MySQL instalado y en funcionamiento.

# Etapa 1: Configuración Inicial

Iniciaremos creando el entorno de trabajo.

1. **Crear el proyecto**

   Utilizar los siguientes comandos para poder crear diseño de patrón

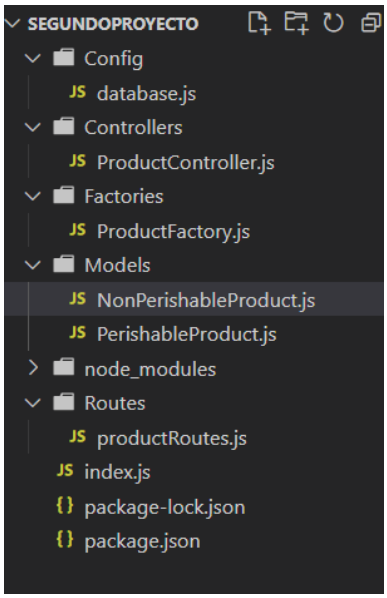
   mkdir SEGUNDOPROYECTO

   cd SEGUNDOPROYECTO

   npm init -y

   npm install express mysql2

2. **Estructura de archivos**



3. **Archivo database.js implementar el siguiente código**

```
// config/database.js
const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('products_db', 'root', '123456789', {
    host: 'localhost',
    dialect: 'mysql',
});

module.exports = sequelize;
```

4. **Archivo ProductController.js implementar el siguiente código**

```
// Controllers/ProductController.js
const PerishableProduct = require('../Models/PerishableProduct');
const NonPerishableProduct = require('../Models/NonPerishableProduct');
const ProductFactory = require('../Factories/ProductFactory');

// Crear productos
const createProduct = async (req, res) => {
    const { type, data } = req.body;
    try {
        const product = ProductFactory.createProduct(type, data);
        await product.save();
        res.status(201).json({ message: 'Product created successfully',
product });
    } catch (error) {
        res.status(400).json({ message: error.message });
    }
};

// Obtener productos
const getProducts = async (req, res) => {
    try {
        const perishableProducts = await PerishableProduct.findAll();
        const nonPerishableProducts = await NonPerishableProduct.findAll();
        res.status(200).json({
            perishableProducts,
            nonPerishableProducts
        });
    } catch (error) {
        res.status(500).json({ message: 'Error fetching products', error });
    }
};
```

```
module.exports = {
    createProduct,
    getProducts,
};
```

5. **Archivo ProductFactory.js implementar el siguiente código**

```
// Factories/ProductFactory.js
const PerishableProduct = require('../Models/PerishableProduct');
const NonPerishableProduct = require('../Models/NonPerishableProduct');

class ProductFactory {
    static createProduct(type, data) {
        switch (type) {
            case 'perishable':
                return new PerishableProduct(data);
            case 'nonPerishable':
                return new NonPerishableProduct(data);
            default:
                throw new Error('Invalid product type');
        }
    }
}

module.exports = ProductFactory;
```

6. **Archivo NonPerishableProduct.js implementar el siguiente código**

```
// Models/NonPerishableProduct.js
const { DataTypes } = require('sequelize');
const sequelize = require('../Config/database');

const NonPerishableProduct = sequelize.define('NonPerishableProduct', {
    name: {
        type: DataTypes.STRING,
        allowNull: false,
    },
    shelfLife: {
        type: DataTypes.INTEGER,
        allowNull: false,
    },
});

module.exports = NonPerishableProduct;
```

7. **Archivo PerishableProduct.js implementar el siguiente código**

```
// Models/PerishableProduct.js
const { DataTypes } = require('sequelize');
const sequelize = require('../Config/database');

const PerishableProduct = sequelize.define('PerishableProduct', {
    name: {
        type: DataTypes.STRING,
        allowNull: false,
    },
    expirationDate: {
        type: DataTypes.DATE,
        allowNull: false,
    },
});

module.exports = PerishableProduct;
```

8. **Archivo productRoutes.js implementar el siguiente código**

```
// Routes/productRoutes.js
const express = require('express');
const { createProduct, getProducts } =
require('../Controllers/ProductController');

const router = express.Router();

// Ruta para crear productos
router.post('/create', createProduct);

// Ruta para obtener productos
router.get('/obtener', getProducts);

module.exports = router;
```

9. **Archivo index.js implementar el siguiente código**

```
// index.js
const express = require('express');
const sequelize = require('./Config/database'); // Conexión a la base de
datos
const productRoutes = require('./Routes/productRoutes');

const app = express();
```

```
const port = 3000;

app.use(express.json());
app.use('/api/products', productRoutes);

// Sincroniza la base de datos y crea las tablas si no existen
sequelize.sync()
    .then(() => {
        console.log('Database synchronized');
        app.listen(port, () => {
            console.log(`Server running on http://localhost:${port}`);
        });
    })
    .catch(err => console.error('Unable to connect to the database:', err));
```

10. Crear la base de datos “products_db”
11. Ejecutar el proyecto “node index.js”
    (Este crea las tablas que se van a utilizar)

## Etapa 2: Pruebas de funcionamiento

http://localhost:3000/api/products/obtener (Para obtener los datos)

http://localhost:3000/api/products/create (Para crear los datos)

```
{
  "type": "perishable",
  "data": {
    "name": "Yogurt",
    "expirationDate": "2024-10-10"
  }
}
```

```
{
  "type": "nonPerishable",
```

```
 "data": {
  "name": "Book",
  "shelfLife": 12
 }
}
```