Prueba Desarrollador

Elaborado Por: Elis Francisco Biches González.

Nota: para las pruebas se recomienda usar **Postman**

Esta prueba debe hacerse con NodeJS, ORM Sequelize y base de datos Postgres o Mysql.

Realice un API Rest con entrada y salida de tipo JSON de un inventario con las siguientes especificaciones:

- Registro (Datos básicos) y login de usuarios. Los usuarios deben tener un rol Administrador o Cliente.
  -Para el registro se deberá mandar en formato JSON por petición POST la siguiente estructura:
  Endpoint: localhost:3000/usuarios/registro

```
{
	"nombre": "pablo",
	"email": "sfdsfsf@gmail.com",
	"password": "password",
	"rol" : "Administrador"
}
```

  En el rol se Puede ser Administrador o Cliente, en el ejemplo es Administrador.

  -para el login se deberá mandar en formato JSON por petición GET la siguiente estructura:
  Endpoint: localhost:3000/usuarios/login

```
{
	"email": "sfdsfsf@gmail.com",
	"password": "password"
}
```

  Si es exitosa le devolverá un toquen de usuario que deberá mandar por Headers con la key “user-token” de lo contrario el usuario no podrá acceder a consultar la información de productos y facturas, el token a método de ejemplo tiene una duración para expirar de 5 minutos, si se requiere mas tiempo modifique la línea 81 en el archivo /routes/usuario.

- Administrador:
  - CRUD de productos del inventario, cada producto debe tener las siguientes especificaciones:
    - Número de lote
    - Nombre
    - Precio
    - Cantidad disponible
    - Fecha de ingreso

- **Create:** Para el registro de productos deberá mandar en formato JSON y por petición POST la siguiente estructura:
  EndPoint: localhost:3000/productos

```
{
        "numeroLote" : 41,
        "nombre" : "Tennis Adidas",
        "precio" : 300000,
        "cantidadDisponible" : 40
}
```

- **Read:** Para consultar los productos deberá mandar una petición GET a la siguiente Endpoint: **localhost:3000/productos**

- **Update:** Para actualizar un producto deberá mandar en formato JSON y por petición PUT la siguiente estructura:
  Endpoint: **localhost:3000/productos/"numeroLote"**

```
{
        "nombre" : "Tennis Adidas",
        "precio" : 300000,
        "cantidadDisponible" : 40
}
```

  Nota: en el Endpoint deberá incluir el numero de lote del producto a actualizar.

- **Delete:** para eliminar un producto deberá mandar por petición DELETE al Endpoint: **localhost:3000/productos/"numeroLote"**
  Nota: en "numeroLote" debe digitar el numero del lote del producto a eliminar.

  - Visualización de compras realizadas por los clientes, debe incluir fecha de la compra, cliente, productos comprados, cantidad por producto y el precio total de la compra,

    Deberá hacer una petición GET al siguiente Endpoint:
    localhost:3000/factura/admin

- Cliente
  - Módulo de compras, donde se puedan agregar 1 o varios productos y la cantidad por cada producto.
    Deberá hacer una petición POST al siguiente Endpoint:
    localhost:3000/factura/compra

```
{
        "idCliente": 2,
        "productos": [
        {
                "idFactura": 2,
                "idProducto": 1,
                "cantidad": 2,
                "Subtotal": 42
        },
```

```
        {
                "idFactura": 2,
                "idProducto": 2,
                "cantidad": 4,
                "Subtotal": 70000
        }
        ]
}
```

Nota: puede agregar los productos que requiera siguiendo el formato JSON.

- Visualización de una factura donde salga la información completa de la compra. Deberá hacer una petición GET al siguiente Endpoint:
  localhost:3000/factura/all/"id_de_factura"
  Nota: id_de_factura corresponde al identificador asignado por la base de datos a esa factura.

- Historial de productos comprados.
  Deberá hacer una petición GET al siguiente Endpoint:
  localhost:3000/factura/buy/"idCliente"
  Nota: idCliente corresponde al id asignado por la base de datos de ese cliente.

Subir el código a github y adjunta un documento con las instrucciones del software.

Link github: https://github.com/DeveloperBilches/Prueba-Desarrollador-senior-1.6.git

Para tener en cuenta:

```
"dependencies": {
        "bcryptjs": "^2.4.3",
        "cors": "^2.8.5",
        "express": "^4.17.3",
        "express-validator": "^6.14.0",
        "jwt-simple": "^0.5.6",
        "moment": "^2.29.3",
        "mysql2": "^2.3.3",
        "sequelize": "^6.19.0"
    }
```

Base de datos en Mysql.

Nombre de la base de datos: desarrolladorseniorprueba

Para puntos extra, tomar en cuenta logs, captura de errores, validaciones de datos y documentar el código preferiblemente con apidoc.