17 DE JULIO DE 2024

MANUAL DE USUARIO PARA DESARROLLADORES

CONCESIONARIOS DE AUTOMÓVILES

PROFESOR : DIEGO FRANCO

PRESENTADO POR : MIGUEL VARGAS

Introducción

Este manual está diseñado para desarrolladores que trabajarán con el sistema de información para un concesionario de automóviles. El sistema está construido utilizando Node.js y el framework Express para el backend, con una base de datos PostgreSQL. Proporciona una API RESTful para gestionar la venta de vehículos e insumos, que pueden provenir de diferentes almacenes.

Requisitos Previos

- Node.js y npm instalados en tu máquina.
- PostgreSQL instalado y configurado.
- Conocimientos básicos de Node.js, Express, y PostgreSQL.
- Postman o cualquier otro cliente HTTP para probar la API.
- Un servidor SMTP configurado para la funcionalidad de notificación por correo electrónico.

Instalación y Configuración

1. Clonar el Repositorio

   git clone <url-del-repositorio>

   cd <nombre-del-repositorio>

2. Instalar Dependencias

   npm install

3. Configurar la Base de Datos

Crea una base de datos PostgreSQL y actualiza el archivo de configuración con los detalles de tu base de datos.

Configura las credenciales de la base de datos en el archivo config.js.

```
const { Pool } = require('pg');

const pool = new Pool({
```

```
    user: 'postgres',
    host: 'localhost',
    database: 'ventas_sistemas',
    password: '1948',
    port: 5432, // puerto por defecto de PostgreSQL
});

module.exports = pool;
```

4. Ejecutar Migraciones

Asegúrate de tener todas las migraciones necesarias para configurar las tablas de la base de datos.

Crea un archivo .env en la raíz del proyecto y configura las variables de entorno necesarias:

Configuración de la base de datos (DB_HOST, DB_USER, DB_PASS, DB_NAME).

Configuración del servidor SMTP (SMTP_HOST, SMTP_PORT, SMTP_USER, SMTP_PASS).

5. Iniciar el Servidor

npm start

Estructura del Proyecto

El proyecto está organizado de la siguiente manera:

- routes/: Contiene todas las rutas para cada entidad.

- config/: Contiene la configuración de la base de datos.

- app.js: Punto de entrada de la aplicación.

- /middlewares: Incluye middleware personalizado para el manejo de solicitudes.

Descripción de Entidades y Rutas

Entidades Principales

1. Taller
2. Producto
3. Insumo
4. Vehículo
5. Vendedor
6. Concesionario

7. Cliente
8. Venta
9. Detalle_Venta
10. Producto_Taller

Rutas Disponibles

A continuación, se detallan las rutas disponibles para cada entidad.

Taller

- GET /talleres: Obtener todos los talleres.

- GET /talleres/:id: Obtener un taller por ID.

- POST /talleres: Crear un nuevo taller.

- PUT /talleres/:id: Actualizar un taller existente.

- PATCH /talleres/:id: Actualización parcial de un taller.

- DELETE /talleres/:id: Eliminar un taller.

Producto

- GET /productos: Obtener todos los productos.

- GET /productos/:id: Obtener un producto por ID.

- POST /productos: Crear un nuevo producto.

- PUT /productos/:id: Actualizar un producto existente.

- PATCH /productos/:id: Actualización parcial de un producto.

- DELETE /productos/:id: Eliminar un producto.

Insumo

- GET /insumos: Obtener todos los insumos.

- GET /insumos/:id: Obtener un insumo por ID.

- POST /insumos: Crear un nuevo insumo.

- PUT /insumos/:id: Actualizar un insumo existente.
- PATCH /insumos/:id: Actualización parcial de un insumo.
- DELETE /insumos/:id: Eliminar un insumo.

Vehículo

- GET /vehiculos: Obtener todos los vehículos.
- GET /vehiculos/:id: Obtener un vehículo por ID.
- POST /vehiculos: Crear un nuevo vehículo.
- PUT /vehiculos/:id: Actualizar un vehículo existente.
- PATCH /vehiculos/:id: Actualización parcial de un vehículo.
- DELETE /vehiculos/:id: Eliminar un vehículo.

Vendedor

- GET /vendedores: Obtener todos los vendedores.
- GET /vendedores/:id: Obtener un vendedor por ID.
- POST /vendedores: Crear un nuevo vendedor.
- PUT /vendedores/:id: Actualizar un vendedor existente.
- PATCH /vendedores/:id: Actualización parcial de un vendedor.
- DELETE /vendedores/:id: Eliminar un vendedor.

Concesionario

- GET /concesionarios: Obtener todos los concesionarios.
- GET /concesionarios/:id: Obtener un concesionario por ID.
- POST /concesionarios: Crear un nuevo concesionario.
- PUT /concesionarios/:id: Actualizar un concesionario existente.
- PATCH /concesionarios/:id: Actualización parcial de un concesionario.
- DELETE /concesionarios/:id: Eliminar un concesionario.

Cliente

- GET /clientes: Obtener todos los clientes.

- GET /clientes/:id: Obtener un cliente por ID.

- POST /clientes: Crear un nuevo cliente.

- PUT /clientes/:id: Actualizar un cliente existente.

- PATCH /clientes/:id: Actualización parcial de un cliente.

- DELETE /clientes/:id: Eliminar un cliente.

Venta

- GET /ventas: Obtener todas las ventas.

Detalle de Venta

- GET /detallesventa: Obtener todos los detalles de ventas.

Producto Taller

- GET /productostaller: Obtener todos los productos en talleres.

- GET /productostaller/:id: Obtener un producto en taller por ID.

- POST /productostaller: Crear un nuevo producto en taller.

- PUT /productostaller/:id: Actualizar un producto en taller existente.

- PATCH /productostaller/:id: Actualización parcial de un producto en taller.

- DELETE /productostaller/:id: Eliminar un producto en taller.

Pruebas en Postman

Para cada entidad del sistema, se realizan pruebas de las APIs utilizando Postman. A continuación, se describen los pasos generales para realizar las pruebas:

1. Configurar la URL base: Configurar la URL base de la API en Postman (por ejemplo, `http://localhost:3001`).

2. Seleccionar el método HTTP: Seleccionar el método HTTP adecuado (GET, POST, PUT, PATCH, DELETE).

3. Configurar la ruta de la API: Configurar la ruta de la API correspondiente a la entidad y operación deseada (por ejemplo, `/talleres`, `/productos/:id`).

4. Enviar el cuerpo de la solicitud: Para operaciones POST, PUT y PATCH, enviar el cuerpo de la solicitud en formato JSON con los datos necesarios.

5. Enviar la solicitud: Hacer clic en "Send" para enviar la solicitud y revisar la respuesta del servidor.

Formato JSON

Las solicitudes y respuestas de la API se manejan en formato JSON. A continuación, se presenta un ejemplo de formato JSON para crear un nuevo producto:

Solicitud POST /productos


```
{
  "nombre": "Aceite de Motor",
  "descripcion": "Aceite sintético para motores de alta gama",
  "precio": 45.99,
  "stock": 100,
  "almacenId": 1
}
```


Respuesta :


```
{
  "id": 1,
  "nombre": "Aceite de Motor",
  "descripcion": "Aceite sintético para motores de alta gama",
  "precio": 45.99,
  "stock": 100,
```

"almacenId": 1,

"createdAt": "2024-07-15T12:00:00.000Z",

"updatedAt": "2024-07-15T12:00:00.000Z"

}

Así para cada una de las entidades:

- **Solicitud POST /talleres**
- **Solicitud POST /productos**
- **Solicitud POST /insumos**
- **Solicitud POST /vehículos**
- **Solicitud POST /vendedores**
- **Solicitud POST /concesionarios**
- **Solicitud POST /clientes**
- **Solicitud POST /ventas**
- **Solicitud POST /detalleventas**
- **Solicitud POST /productostaller**

Funcionalidad de Envío de Correos Electrónicos

Características

Envío automático de confirmaciones de compra a los clientes.

Notificaciones por correo a los empleados y administradores sobre ventas realizadas y otras actividades relevantes.

Configuración

Instalar una biblioteca de envío de correos electrónicos.

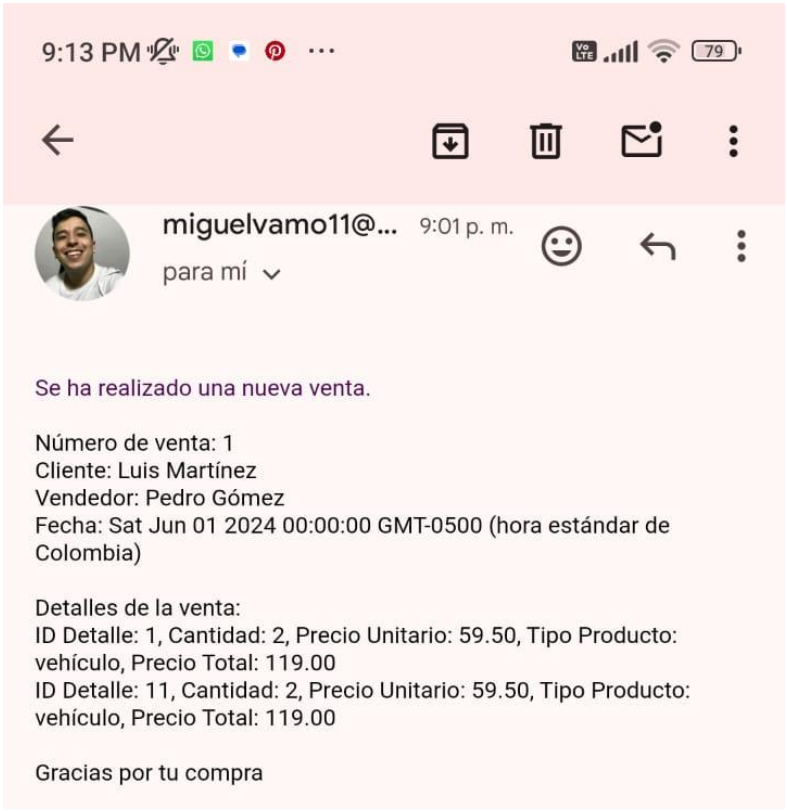
Configurar el servidor SMTP con las credenciales necesarias en el archivo de configuración del proyecto.

Implementar la lógica de envío de correos electrónicos en los puntos relevantes del sistema, como después de la creación de una venta.

```
const transporter = nodemailer.createTransport({
  service: 'gmail', // Puedes usar cualquier otro correo.
  auth: {
    user: 'miguelvamo11@gmail.com',
    pass: 'wwsg srmu ghfm pzxh'
  }
});
```

```
const sendMail = (to, subject, text) => {
  const mailOptions = {
    from: 'miguelvamo11@gmail.com',
    to,
    subject,
    text
  };

  transporter.sendMail(mailOptions, (error, info) => {
    if (error) {
      return console.log(error);
    }
    console.log('Email sent: ' + info.response);
  });
};
```



Buenas Prácticas

- Seguridad: Asegúrate de proteger las rutas que requieren autenticación y autorización.

- Validación de Datos: Implementa validaciones para los datos recibidos en las solicitudes.

- Manejo de Errores: Asegúrate de manejar los errores de manera adecuada y proporcionar respuestas significativas.

- Documentación: Mantén la documentación de la API actualizada para facilitar el trabajo de otros desarrolladores.

Conclusión

Este sistema de información para concesionarios de automóviles proporciona una base sólida para gestionar la venta de vehículos e insumos. Con las rutas detalladas y las prácticas recomendadas, los desarrolladores pueden extender y mantener el sistema de manera eficiente.