Documentación del Sistema de Gestión de compras e Inventario

Nombre de los integrantes:  
Tox Dzul Omax

Rosales Huey Francisco Jareth

Ameyalli Yaxelen Mejia Diaz

**1. Especificación de la API**

**Descripción General**

La API del Sistema de Gestión de Compras e Inventario permite a los usuarios administrar proveedores, productos y compras de manera eficiente. Implementada bajo la arquitectura RESTful, ofrece funcionalidades como:

* Generación de reportes sobre compras realizadas.
* Gestión de usuarios
* Manejo de inventario (productos y stock).
* Control de productos
* Administración de proveedores
* Registro de compras
* Generación de reportes

**Autenticación**

La API requiere autenticación mediante una Clave API (`claveApi`) para acceder a los recursos protegidos. La clave se genera en el momento del registro del usuario y se envía en las peticiones como un encabezado HTTP:

http

Endpoints y Métodos HTTP

Toda solicitud debe incluir un encabezado:

makefile

CopiarEditar

Authorization: Bearer TU\_CLAVE\_API

Además, para operaciones de escritura (POST, PUT, DELETE) es obligatorio enviar el encabezado:

pgsql

CopiarEditar

Content-Type: application/json

La API proporciona varios endpoints organizados por funcionalidad. A continuación, se detallan cada uno con los parámetros esperados y la estructura de respuesta.

**Usuarios**

Registro de usuario

- URL: POST /usuarios

- Descripción: Registra un nuevo usuario y devuelve una clave API.

- Parámetros:

json

{

"nombre": "Ejemplo Usuario",

"correo": "ejemplo@email.com",

"contrasena": "123456"

}

- Respuesta:

json

{

"estado": 1,

"mensaje": "Usuario registrado correctamente",

"claveApi": "ABCDEF123456"

}

**Autenticación**

- URL: POST /usuarios/autenticar`

- Descripción: Valida las credenciales del usuario.

**Parámetros**

**json**

{

"correo": "ejemplo@email.com",

"contrasena": "123456"

}

**- Respuesta:**

**json**

{

"estado": 1,

"mensaje": "Autenticación exitosa",

"usuario": {

"id": 1,

"nombre": "Ejemplo",

"correo": "ejemplo@email.com",

"claveApi": "ABCDEF123456"

}

}

**Productos  
Consulta de productos**

**- URL: GET /productos**

- Encabezado:

http

Authorization: {claveApi}

- Descripción: Obtiene todos los productos registrados por el usuario autenticado.

- Respuesta:

json

{

"estado": 1,

"mensaje": "Productos recuperados correctamente",

"datos": [

{

"idProducto": 1,

"nombre": "Laptop HP",

"precioCompra": 15000,

"precioVenta": 17500,

"stock": 5

},

{

"idProducto": 2,

"nombre": "Mouse Inalámbrico",

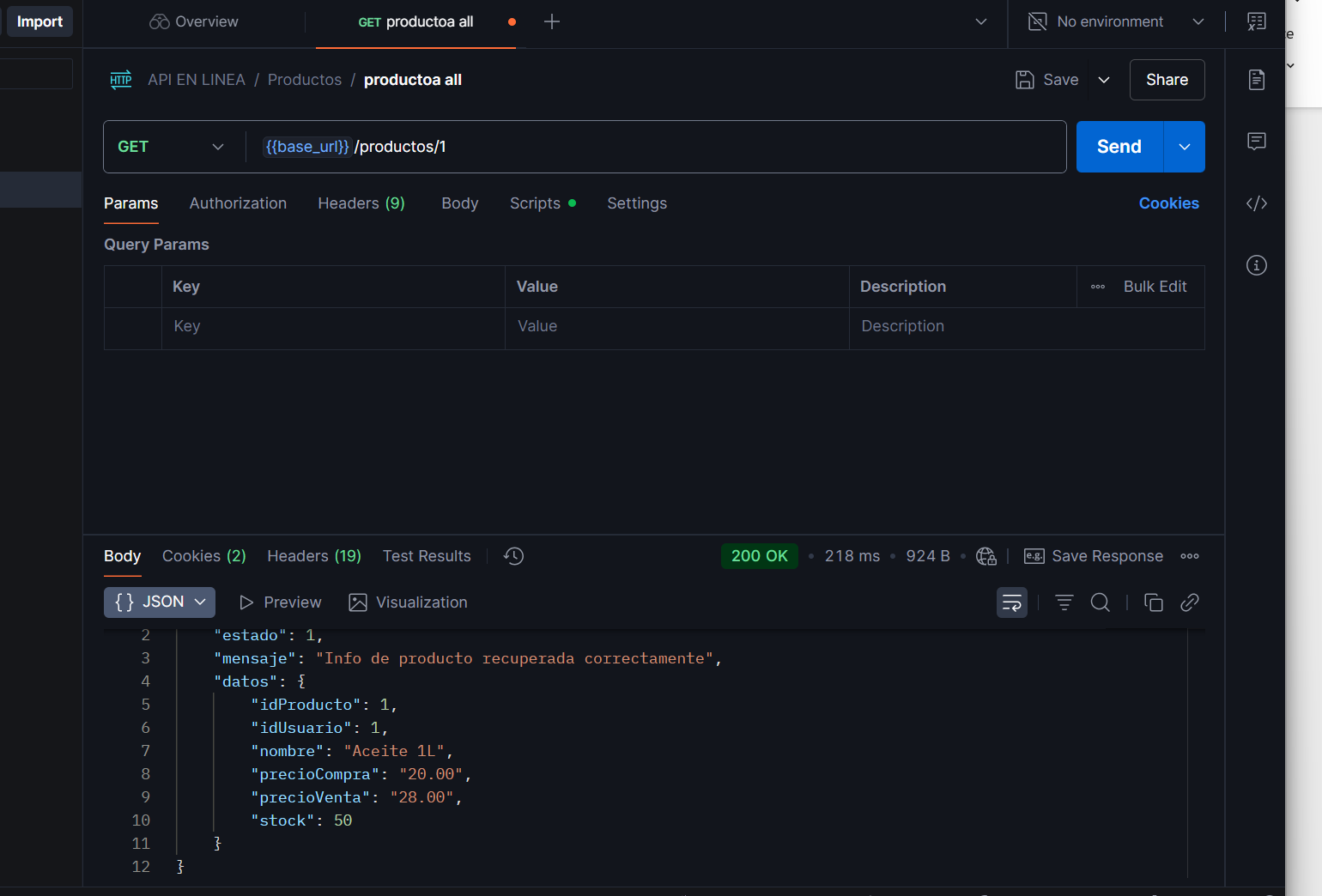
"precioCompra": 500,

"precioVenta": 750,

"stock": 20

}

]

}  


Filtrar productos por proveedor

- URL: GET /productos/proveedor/{idProveedor}

- Descripción: Obtiene productos suministrados por un proveedor específico.

- Respuesta:

json

{

"estado": 1,

"mensaje": "Productos encontrados correctamente",

"datos": [...]

}

**Compras**

**Registrar una compra**

- URL: POST /compras

- Parámetros:

json

{

"idProveedor": 2,

"productos": [

{ "idProducto": 1, "cantidad": 2, "precioUnitario": 10 }

]

}

**-Respuesta:**

**json**

{

"estado": 1,

"mensaje": "Compra registrada correctamente",

"idCompra": 5,

"total": 20

}

**Consultar compras**

- URL: GET /compras

- Encabezado:

http

Authorization: {claveApi}

- Descripción: Obtiene el historial de compras realizadas por el usuario.

**-Respuesta:**

**json**

{

"estado": 1,

"mensaje": "Compras recuperadas correctamente",

"datos":

}  
ejemplo de Postman:

**Proveedores**

**Consultar todos los proveedores**

-URL:GET /proveedores`

- Encabezado:

http

Authorization: {claveApi}

- Descripción: Obtiene la lista de proveedores registrados.

- **Respuesta:**

**son**

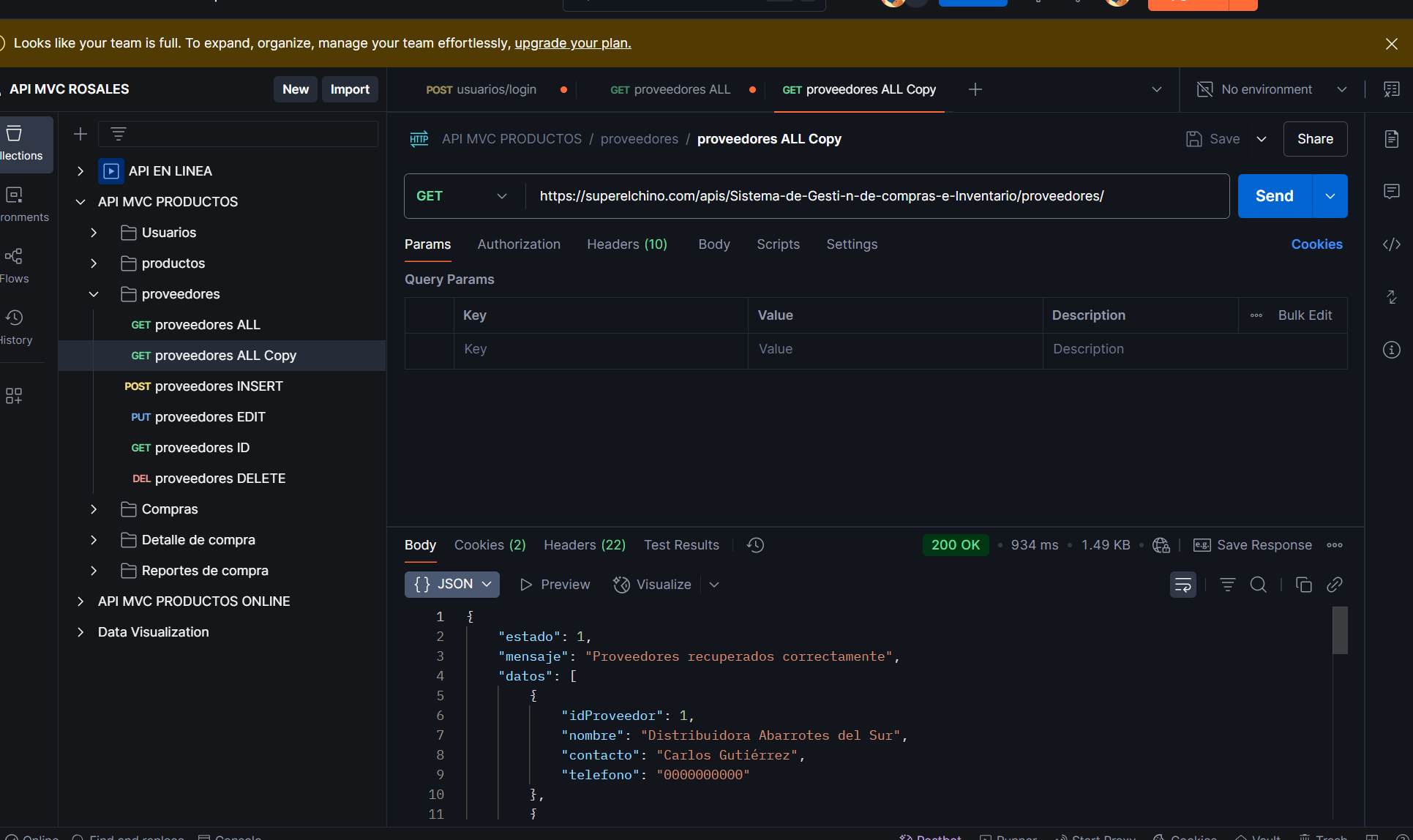
{

"estado": 1,

"mensaje": "Proveedores recuperados correctamente",

"datos":

}

Ejemplo postman:  


**5. Ejemplo de Consumo (Postman)**

* Método: GET
* URL: http://localhost/index.php?PATH\_INFO=productos
* Headers:
  + Authorization: Bearer 9e3b2ab4157ddc52bcf9dc2ee5dd57e6584675a91
  + Content-Type: application/json

**6. Ejemplo de Consumo desde PHP**

php

CopiarEditar

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_URL, "http://localhost/index.php?PATH\_INFO=productos");

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

curl\_setopt($ch, CURLOPT\_HTTPHEADER, [

"Authorization: Bearer TU\_CLAVE\_API",

"Content-Type: application/json"

]);

$response = curl\_exec($ch);

curl\_close($ch);

echo $response;

**7. Ejemplo de Consumo desde JavaScript**

javascript

CopiarEditar

fetch("http://localhost/index.php?PATH\_INFO=productos", {

method: "GET",

headers: {

"Authorization": "Bearer TU\_CLAVE\_API",

"Content-Type": "application/json"

}

})

.then(res => res.json())

.then(data => console.log(data));

**8. Reportes PDF**

* Generados desde el endpoint /reportes
* Utiliza la librería fpdf.php
* Se crean con base en las compras, productos o proveedores
* El PDF se descarga automáticamente o se visualiza en el navegador

**Reportes**

**Generar reportes de compras**

**- URL: GET /reportes/compras**

**-Encabezado:**

http

Authorization: {claveApi}

**- Descripción: Obtiene un resumen de compras realizadas.**

**-Respuesta:**

**json**

{

"estado": 1,

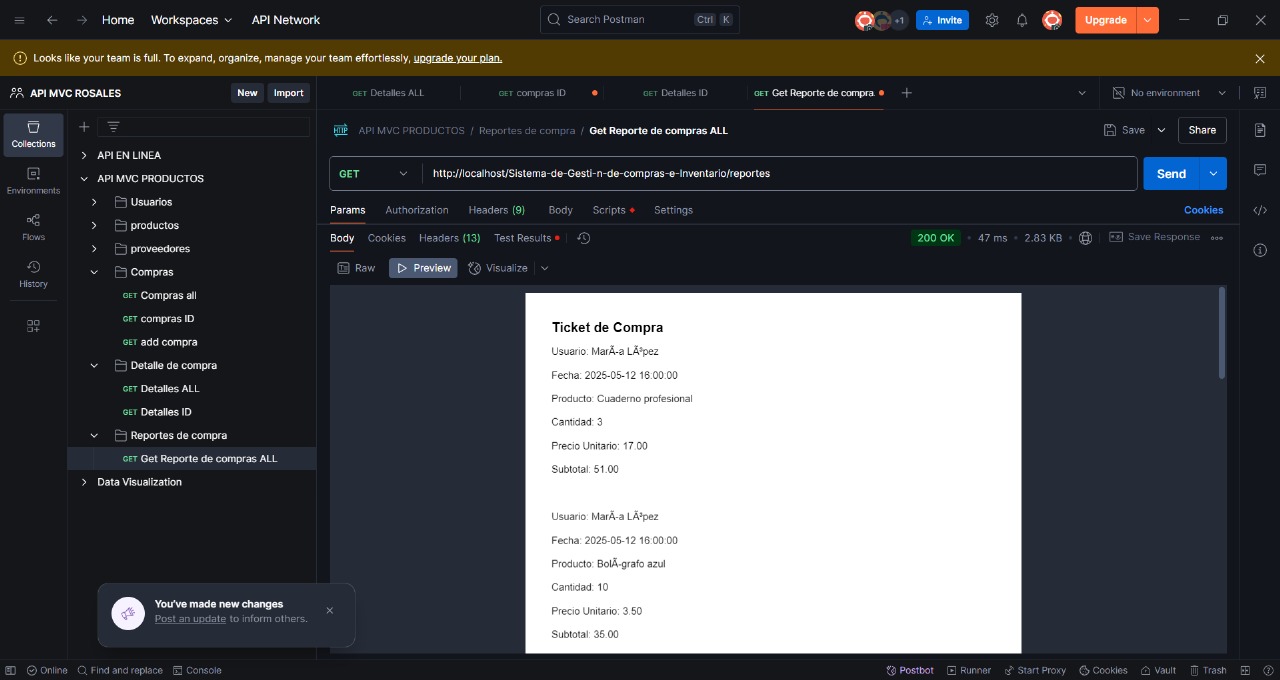
"mensaje": "Reporte generado",

"datos":

}

2. Ejemplo de Generación de Reportes

La API permite generar reportes detallados sobre compras realizadas por los usuarios. Se pueden mostrar en formato JSON o como tablas en un sistema web. **Ejemplo de reporte de compras:**



**Conclusión**

El sistema de gestión de compras e inventario desarrollado presenta una solución robusta, modular y extensible para la administración eficiente de procesos comerciales relacionados con el abastecimiento de productos, control de proveedores, monitoreo de inventario y generación de reportes. Esta solución ha sido construida sobre una arquitectura RESTful que promueve la interoperabilidad entre sistemas y facilita su integración con diversas plataformas frontend o clientes externos como aplicaciones móviles, SPA en JavaScript, o incluso sistemas legacy en PHP.

**1. Estructura técnica y lógica de negocio**

El backend del sistema está implementado en PHP, utilizando una arquitectura basada en controladores y modelos, y conectada a una base de datos relacional (MySQL/MariaDB). La lógica se encuentra separada adecuadamente en capas:

* **Controladores**: reciben las peticiones, validan datos y llaman a los modelos correspondientes.
* **Modelos**: interactúan directamente con la base de datos.
* **Vistas**: aunque en este sistema se manejan principalmente respuestas JSON, están preparadas para retornar datos en formatos estandarizados.

Esta separación de responsabilidades permite una mayor mantenibilidad, escalabilidad y facilidad para la implementación de nuevas funcionalidades.

**2. Seguridad y autenticación**

La API implementa un mecanismo de autenticación simple pero eficaz, mediante el uso de claves API (tokens) tipo Bearer. Este enfoque garantiza que solo usuarios autenticados puedan acceder a recursos protegidos como productos, compras o proveedores.

Cada operación sensible exige el envío del token correspondiente, lo cual es verificado antes de ejecutar cualquier acción en el servidor. Esto evita accesos no autorizados y permite asociar cada operación con un usuario específico, fortaleciendo el control de auditoría.

**3. Consumo de servicios (API REST)**

La API está diseñada para ser consumida por cualquier cliente HTTP capaz de enviar solicitudes con encabezados y cuerpos JSON. La utilización de rutas claras y verbos HTTP estándar permite:

* Obtener productos, proveedores y compras (GET)
* Crear nuevos registros (POST)
* Editar recursos (PUT)
* Eliminar entidades (DELETE)

Además, cuenta con filtros como:

* /productos/bajo-stock/{umbral}
* /productos/por-proveedor/{id}

Lo cual mejora la flexibilidad y adaptabilidad del sistema frente a diferentes necesidades de visualización o consulta.

**4. Reportes y exportación de datos**

Un aspecto clave del sistema es la generación de reportes en formato PDF mediante el uso de la librería FPDF. Este módulo permite exportar información de compras, inventario o proveedores en documentos que pueden ser impresos, enviados por correo o archivados como respaldo. Esta funcionalidad mejora el seguimiento y la transparencia de las operaciones realizadas en el sistema.

**5. Facilidad de integración**

Gracias al uso de CORS (Cross-Origin Resource Sharing), el sistema está preparado para ser consumido desde clientes externos (por ejemplo, un frontend hecho en Vue, React o Angular). Esto abre la posibilidad de escalar el sistema a arquitecturas modernas desacopladas, utilizando este backend como motor central de procesamiento de datos.

También se puede interactuar con la API desde sistemas existentes (por ejemplo, integraciones con ERP, CRM o tiendas virtuales) mediante PHP o JavaScript, lo que garantiza su flexibilidad y reutilización en distintos contextos.

**6. Base de datos normalizada**

La base de datos sistema\_compras está bien estructurada, cumpliendo con principios de normalización y con uso adecuado de claves foráneas. Esto garantiza la integridad referencial y facilita operaciones de JOIN, consultas eficientes y operaciones en cascada como eliminación controlada.

Tablas clave incluyen:

* usuarios
* productos
* proveedores
* compras
* detalle\_compra

.