Vendor Management System Documentation

Introduction

This document provides an overview of the Vendor Management System developed using Django and Django REST Framework. The system facilitates vendor management, purchase order handling, and performance metric calculations.

Technical Stack

- Django 3.x
- Django REST Framework 3.x
- Python 3.x

Setup Instructions

1. Clone the repository from GitHub :
2. Install dependencies using pip:
3. Apply database migrations:
4. Run the development server:

API Endpoints

Vendor Endpoints

- **GET /api/vendors/**
- List all vendors or create a new vendor.
- **GET /api/vendors/{vendor_id}/**
- Retrieve, update, or delete a specific vendor.

Purchase Order Endpoints

- **GET /api/purchaseorders/**

- List all purchase orders or create a new purchase order.
- **GET /api/purchaseorders/{purchaseorder_id}/**
- Retrieve, update, or delete a specific purchase order.

Performance Metrics Endpoints

- **GET /api/vendors/{vendor_id}/performance/**
- Retrieve performance metrics for a specific vendor.
- **POST /api/purchase_orders/{po_id}/acknowledge/**
- Acknowledge a purchase order, updating acknowledgment date.

Data Validations

- Comprehensive data validations are implemented for models to ensure data integrity.
- Constraints and rules are enforced at the model level to maintain consistency.

Security

- API endpoints are secured with token-based authentication to control access and ensure user authentication.

Code Quality

- Codebase adheres to PEP 8 style guidelines for Python code to maintain readability and consistency.
- Django ORM is used for efficient database interactions, ensuring optimal performance with large datasets.

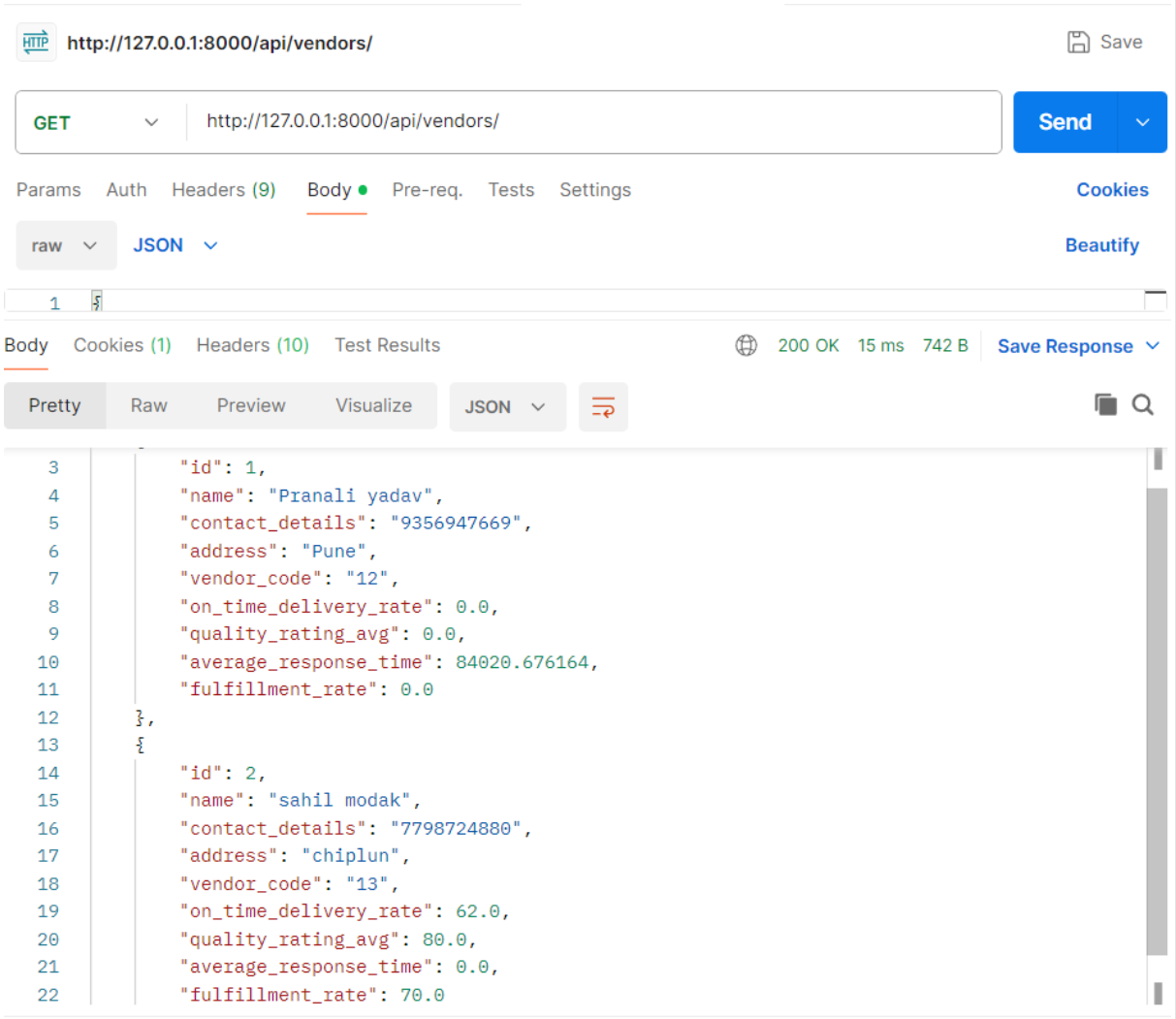
Testing

- A comprehensive test suite is provided to demonstrate the functionality and reliability of the endpoints.
- Unit tests and integration tests cover various scenarios to validate endpoint behavior.

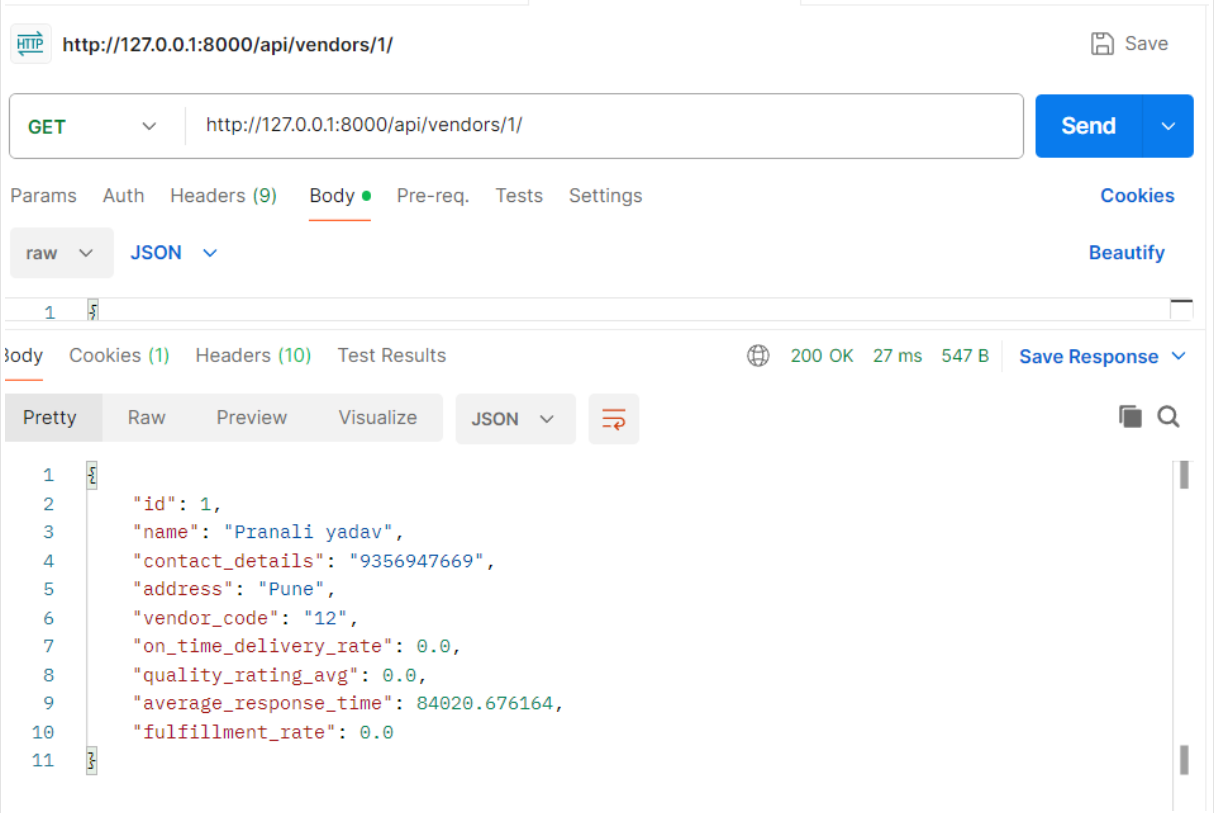
Additional Considerations

- **Efficient Calculation:** Logic for calculating metrics is optimized to handle large datasets without significant performance issues.
- **Data Integrity:** Checks are included to handle scenarios like missing data points or division by zero in calculations.
- **Real-time Updates:** Django signals are used to trigger metric updates in real-time when related PO data is modified.

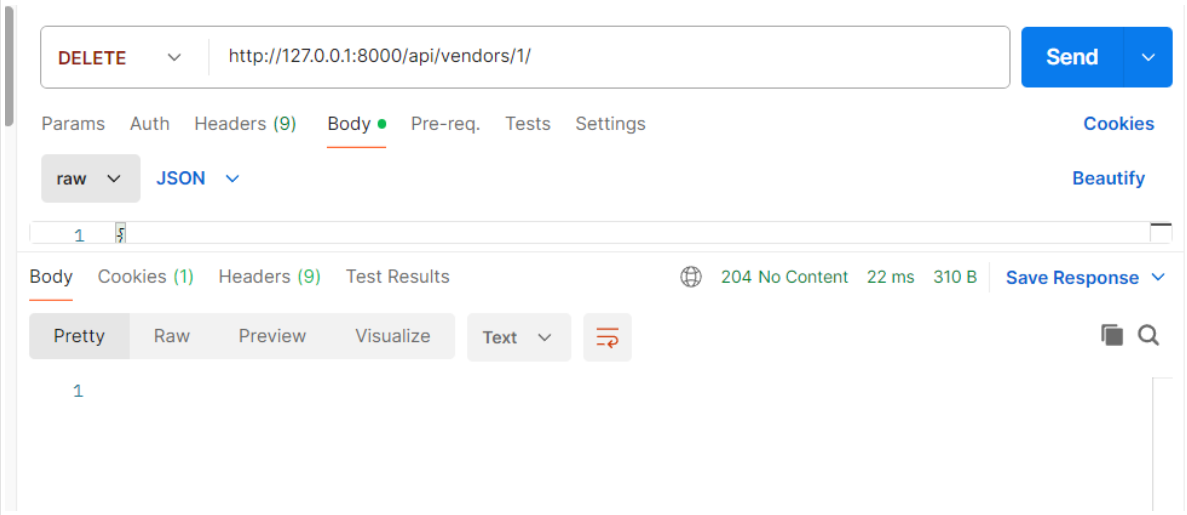
Screenshots of Apis:
Vendors :
GET :



GET element through id:

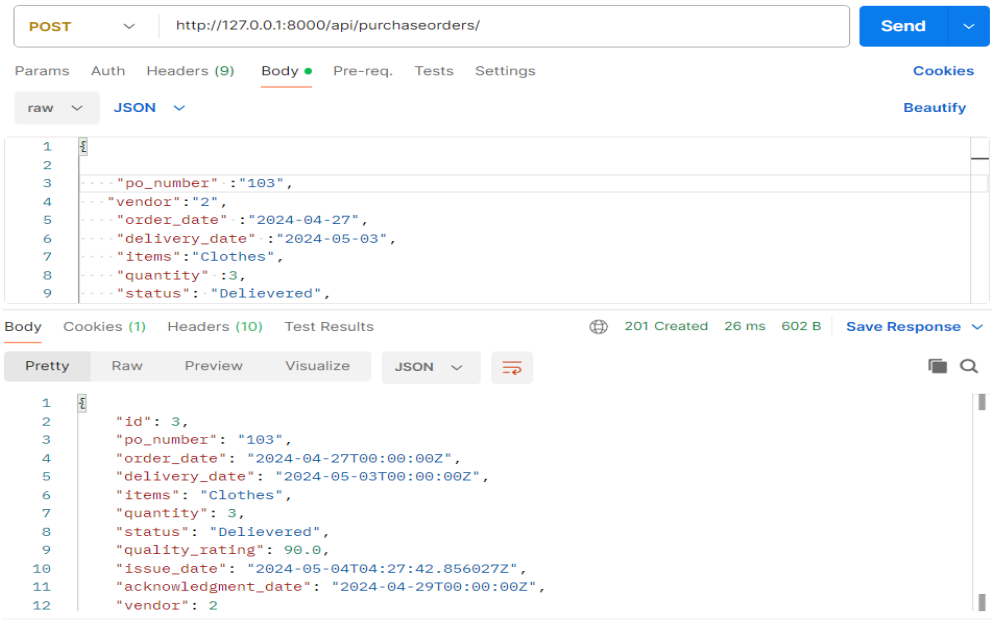


Delete :

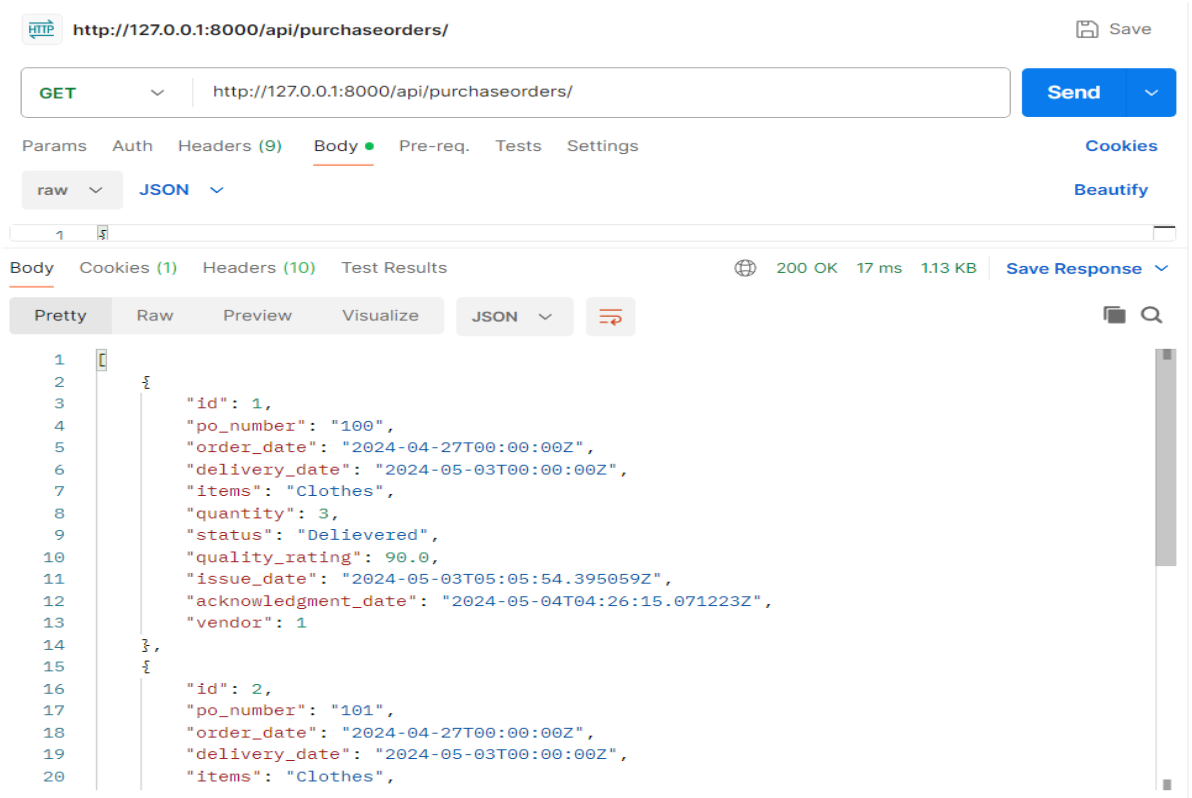


Purchase orders

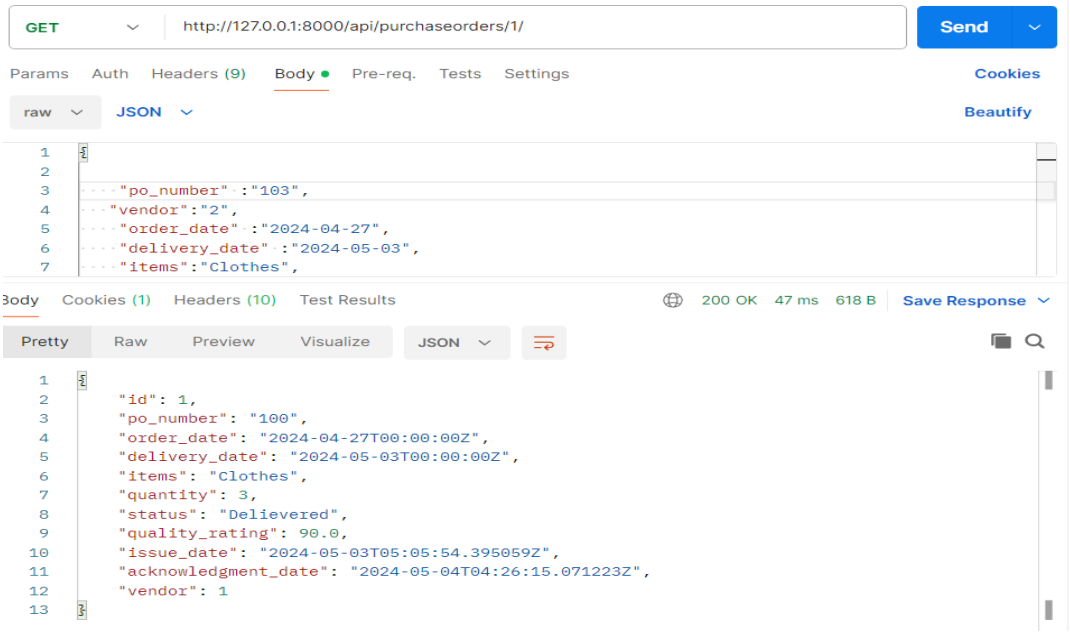
POST:



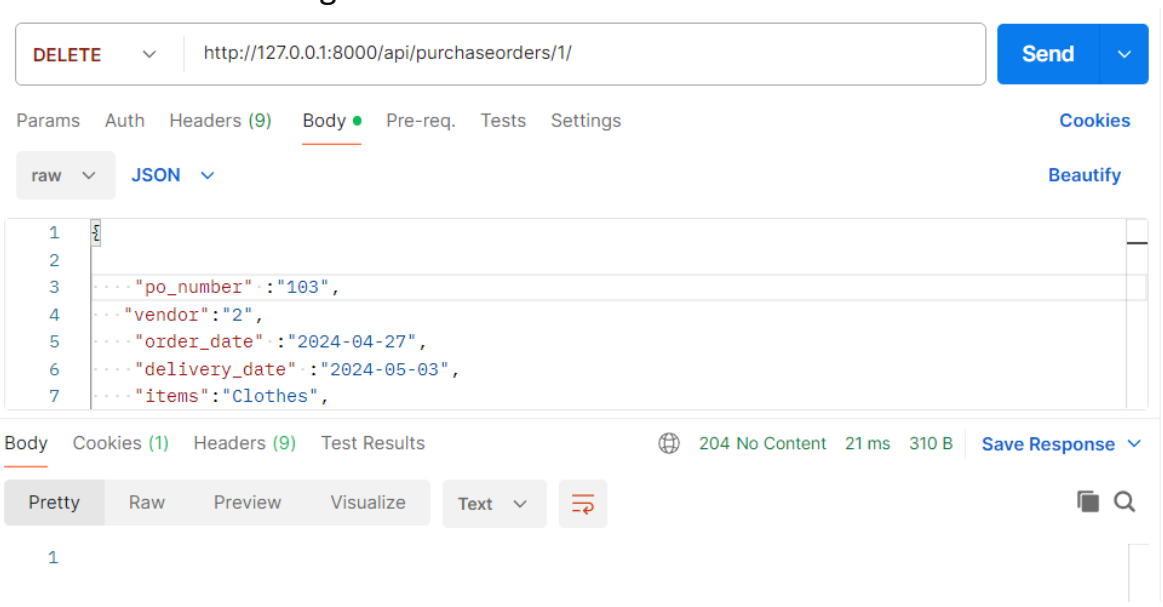
GET :



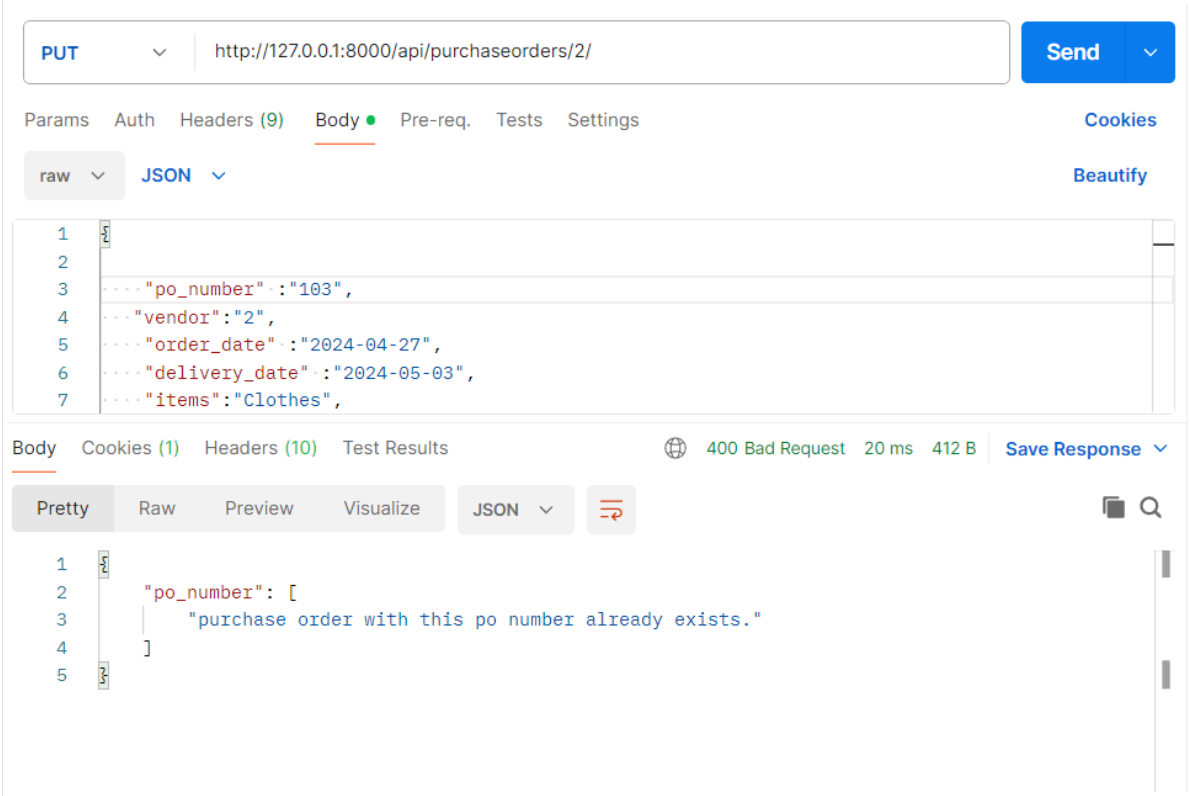
GET Elemets through id :



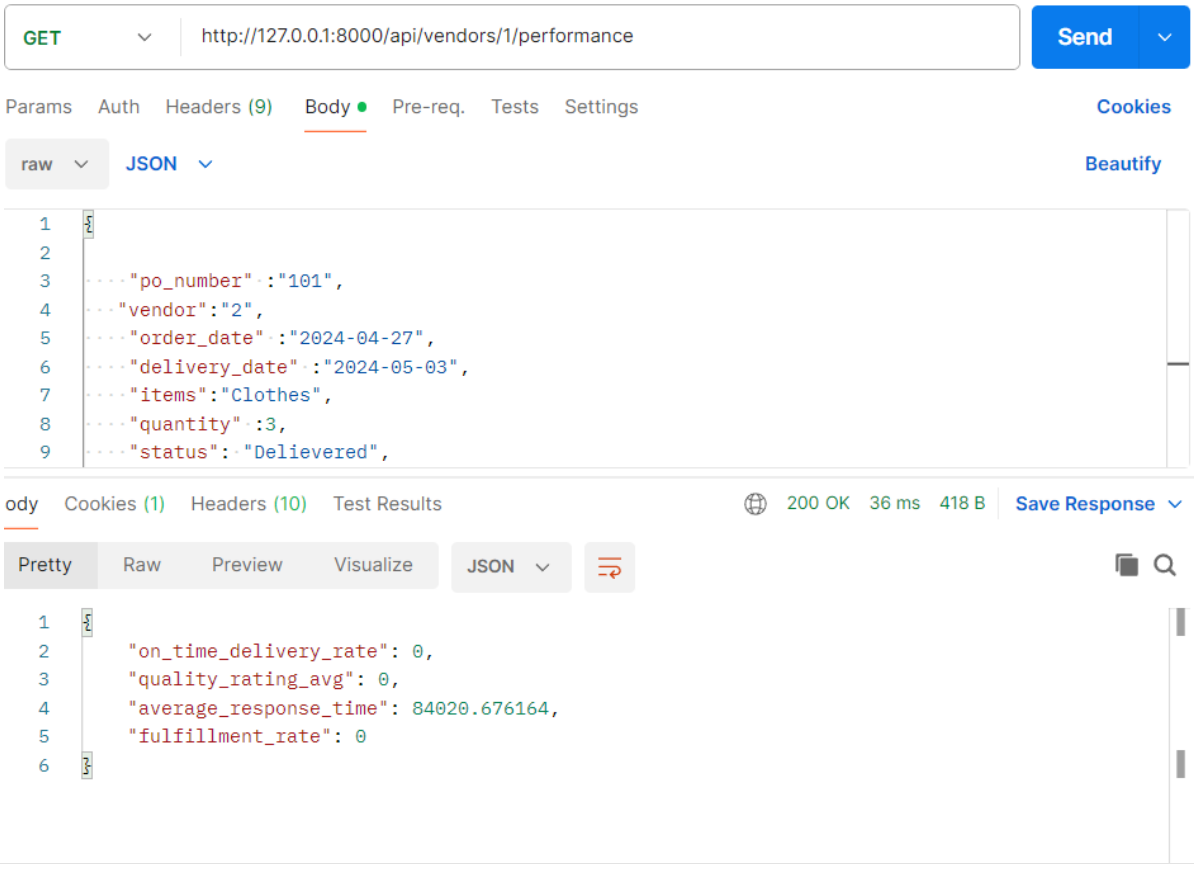
Delete element through id :

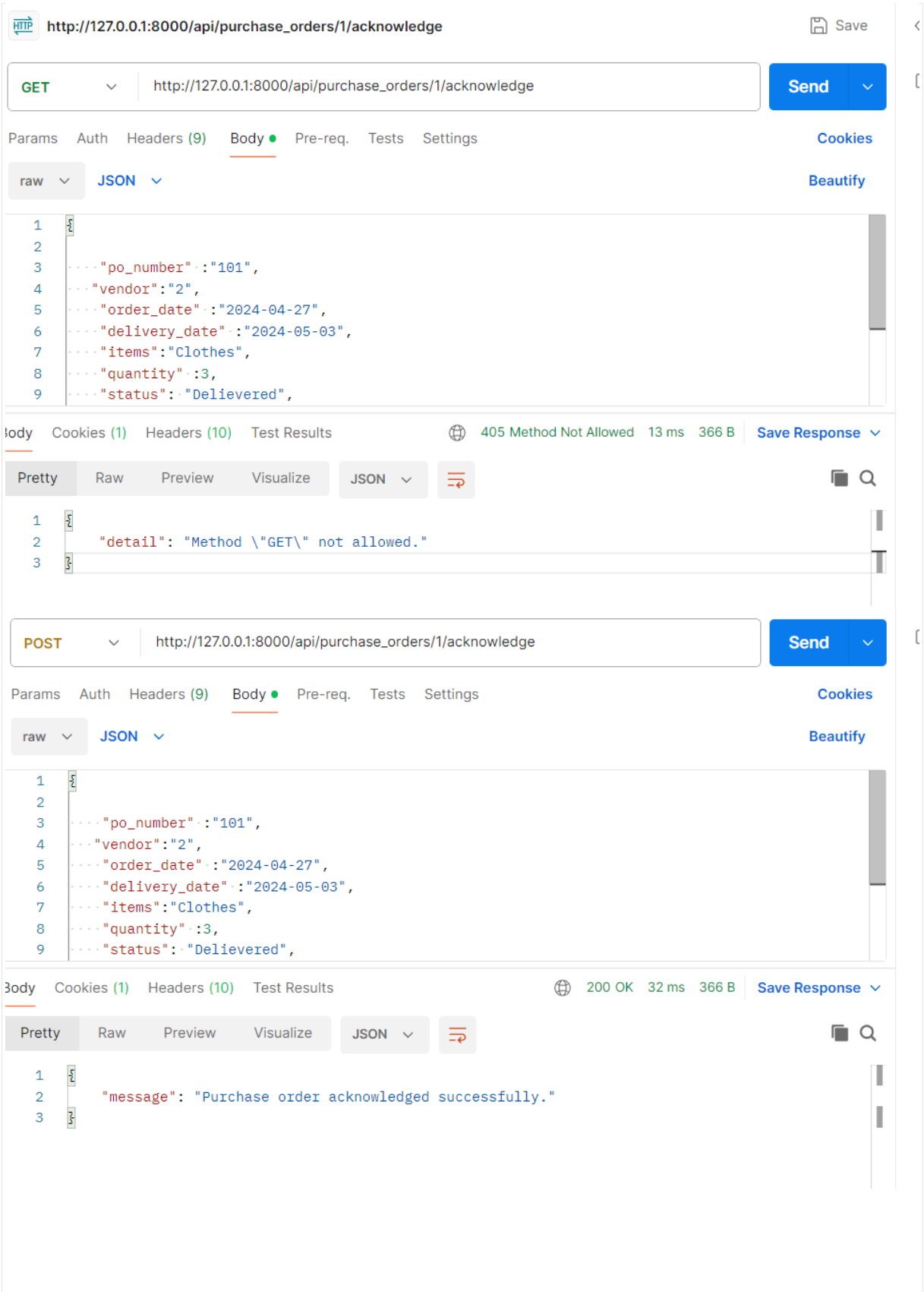


Update element through id :



Performance Metrices :




http://127.0.0.1:8000/api/purchase_orders/1/acknowledge
Save
GET
http://127.0.0.1:8000/api/purchase_orders/1/acknowledge
Send
Params Auth Headers (9) Body Pre-req. Tests Settings
Cookies
raw JSON
Beautify
{
"po_number" :"101",
"vendor":"2",
"order_date" :"2024-04-27",
"delivery_date" :"2024-05-03",
"items":"Clothes",
"quantity" :3,
"status": "Delievered",
Body Cookies (1) Headers (10) Test Results
405 Method Not Allowed 13 ms 366 B
Save Response
Pretty Raw Preview Visualize JSON
{
"detail": "Method \"GET\" not allowed."
}
POST
http://127.0.0.1:8000/api/purchase_orders/1/acknowledge
Send
Params Auth Headers (9) Body Pre-req. Tests Settings
Cookies
raw JSON
Beautify
{
"po_number" :"101",
"vendor":"2",
"order_date" :"2024-04-27",
"delivery_date" :"2024-05-03",
"items":"Clothes",
"quantity" :3,
"status": "Delievered",
Body Cookies (1) Headers (10) Test Results
200 OK 32 ms 366 B
Save Response
Pretty Raw Preview Visualize JSON
{
"message": "Purchase order acknowledged successfully."
}