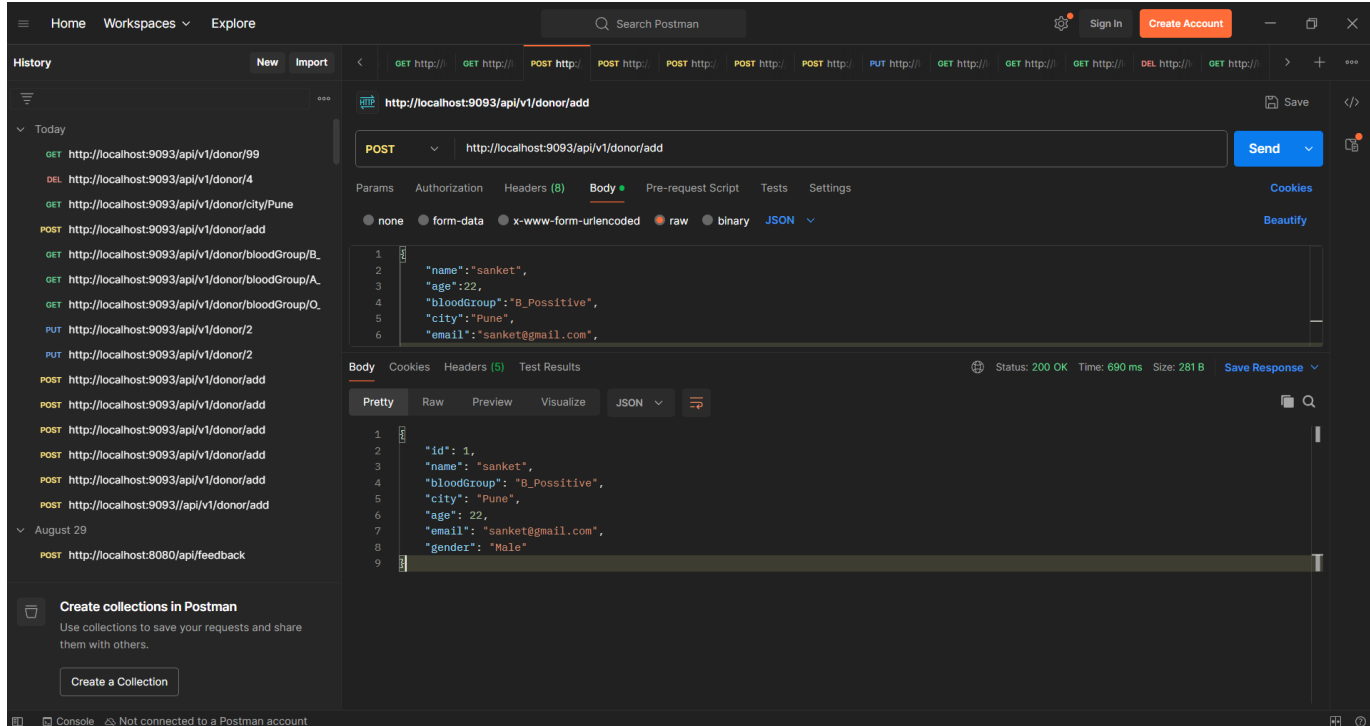


Project Name : Blood Donor Management Using Microservices.

Completed by: Sanket Chandrakant Bondre

//Adding Donor



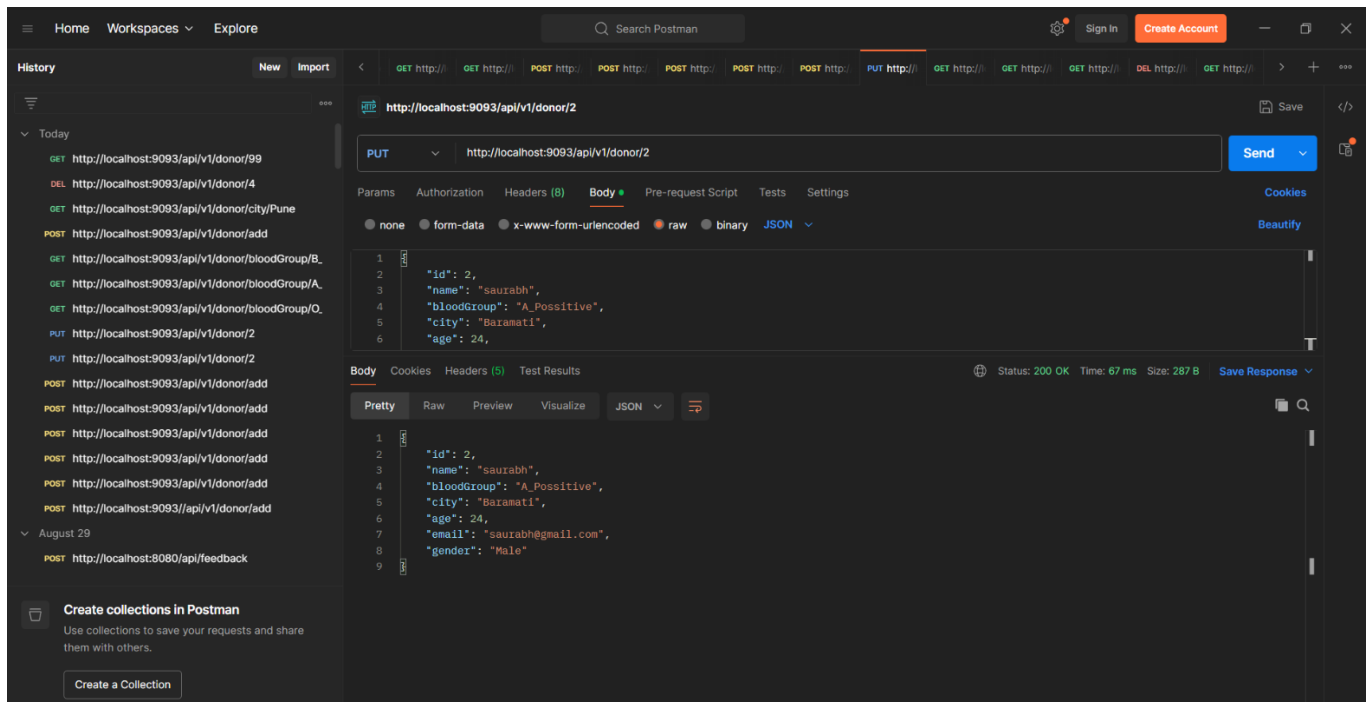
The screenshot shows the Postman interface for a POST request to `http://localhost:9093/api/v1/donor/add`. The request body is a JSON object:

```
{
  "name": "sanket",
  "age": 22,
  "bloodGroup": "B_Positive",
  "city": "Pune",
  "email": "sanket@gmail.com",
}
```

The response is a JSON object:

```
{
  "id": 1,
  "name": "sanket",
  "bloodGroup": "B_Positive",
  "city": "Pune",
  "age": 22,
  "email": "sanket@gmail.com",
  "gender": "Male"
}
```

//Updating Donor



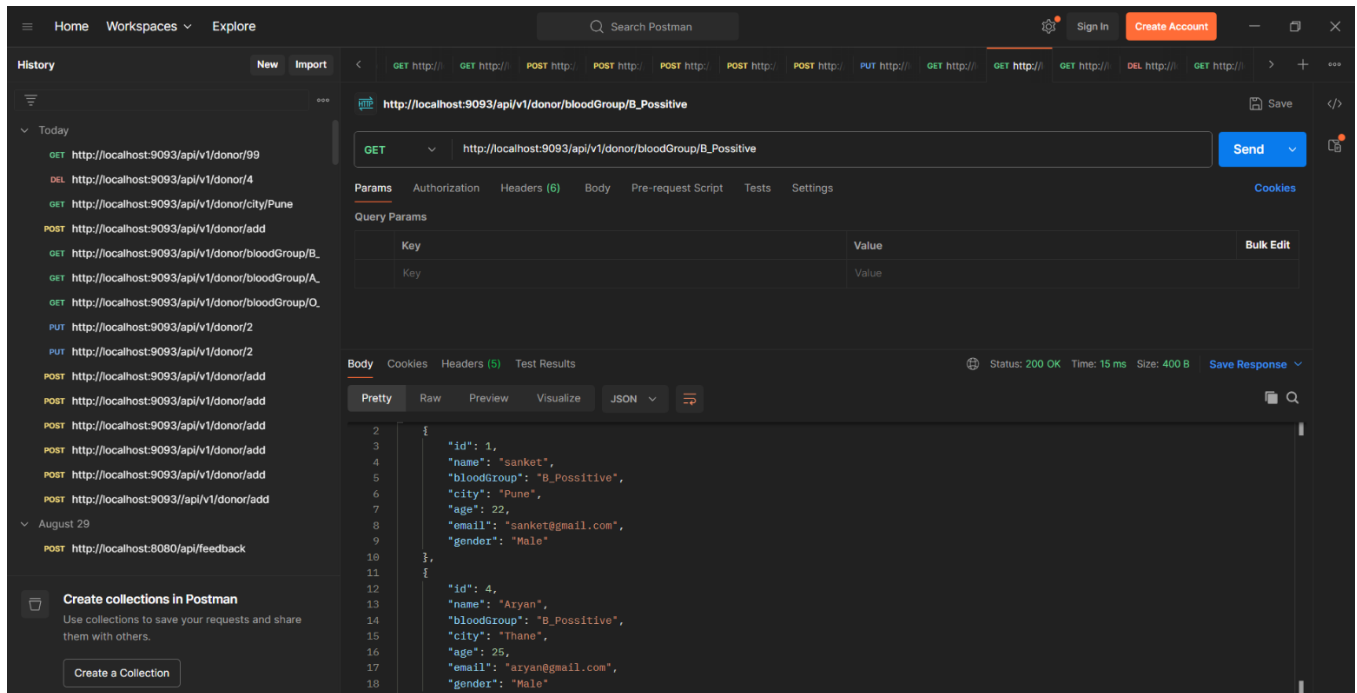
The screenshot shows the Postman interface for a PUT request to `http://localhost:9093/api/v1/donor/2`. The request body is a JSON object:

```
{
  "id": 2,
  "name": "saurabh",
  "bloodGroup": "A_Positive",
  "city": "Baramati",
  "age": 24,
}
```

The response is a JSON object:

```
{
  "id": 2,
  "name": "saurabh",
  "bloodGroup": "A_Positive",
  "city": "Baramati",
  "age": 24,
  "email": "saurabh@gmail.com",
  "gender": "Male"
}
```

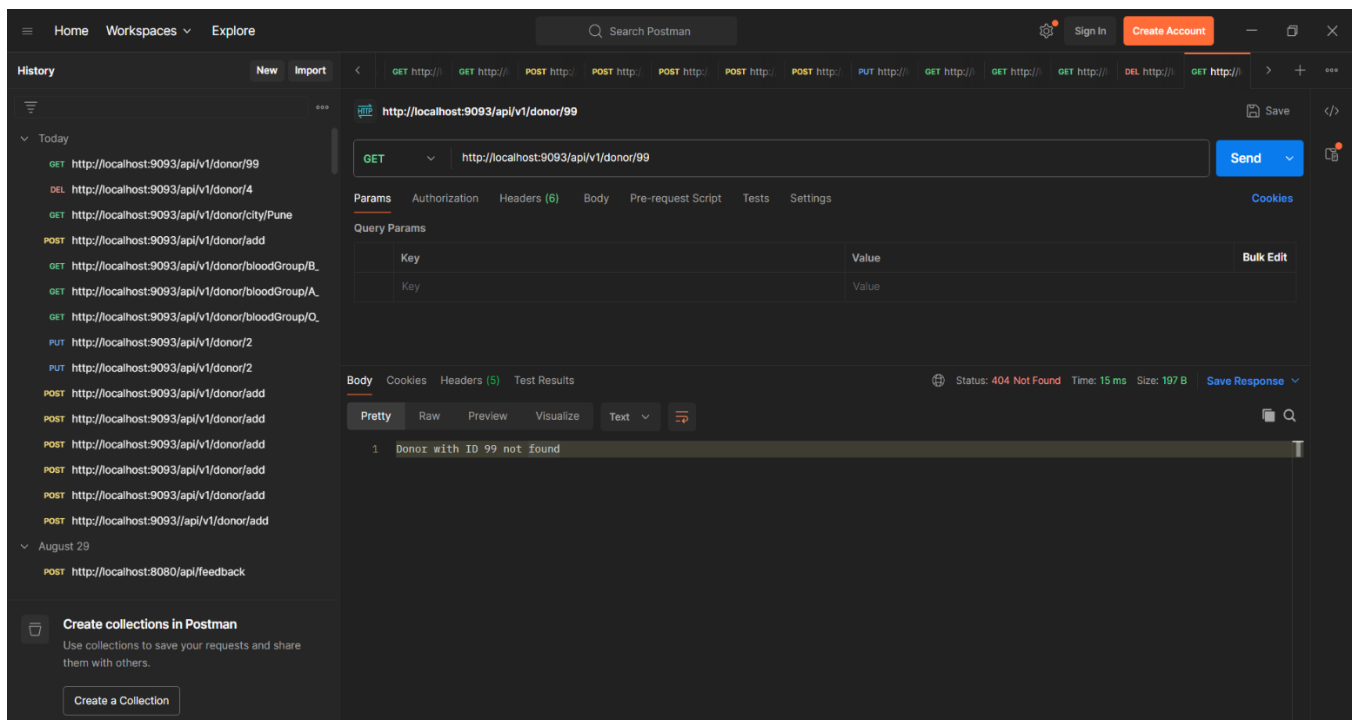
//Getting Donor by BloodGroup



The screenshot shows the Postman interface with a GET request to `http://localhost:9093/api/v1/donor/bloodGroup/B_Positive`. The response status is 200 OK, and the body is a JSON array of two donor objects.

```
2 {
3   "id": 1,
4   "name": "sanket",
5   "bloodGroup": "B_Positive",
6   "city": "Pune",
7   "age": 22,
8   "email": "sanket@gmail.com",
9   "gender": "Male"
10 },
11 {
12   "id": 4,
13   "name": "Aryan",
14   "bloodGroup": "B_Positive",
15   "city": "Mhane",
16   "age": 25,
17   "email": "aryan@gmail.com",
18   "gender": "Male"
19 }
```

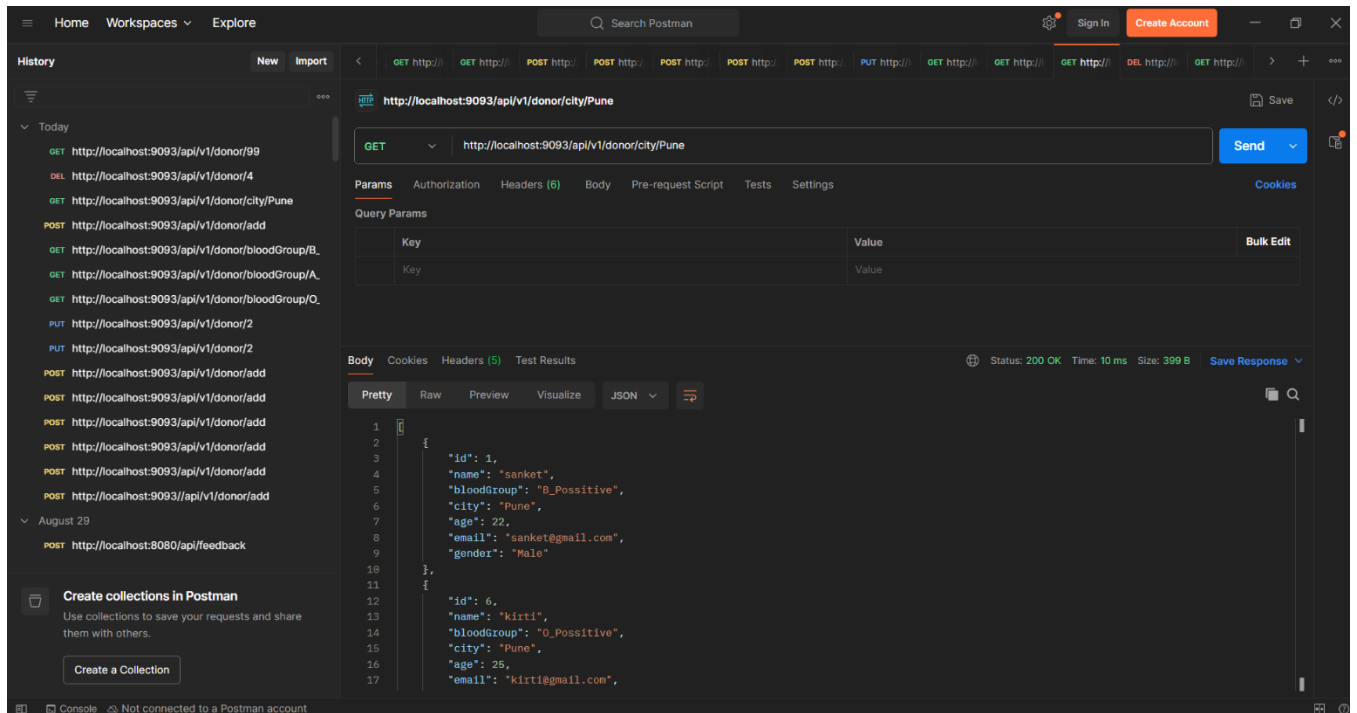
//Donor with Id Not Found Exception



The screenshot shows the Postman interface with a GET request to `http://localhost:9093/api/v1/donor/99`. The response status is 404 Not Found, and the body contains the message "Donor with ID 99 not found".

```
1 Donor with ID 99 not found
```

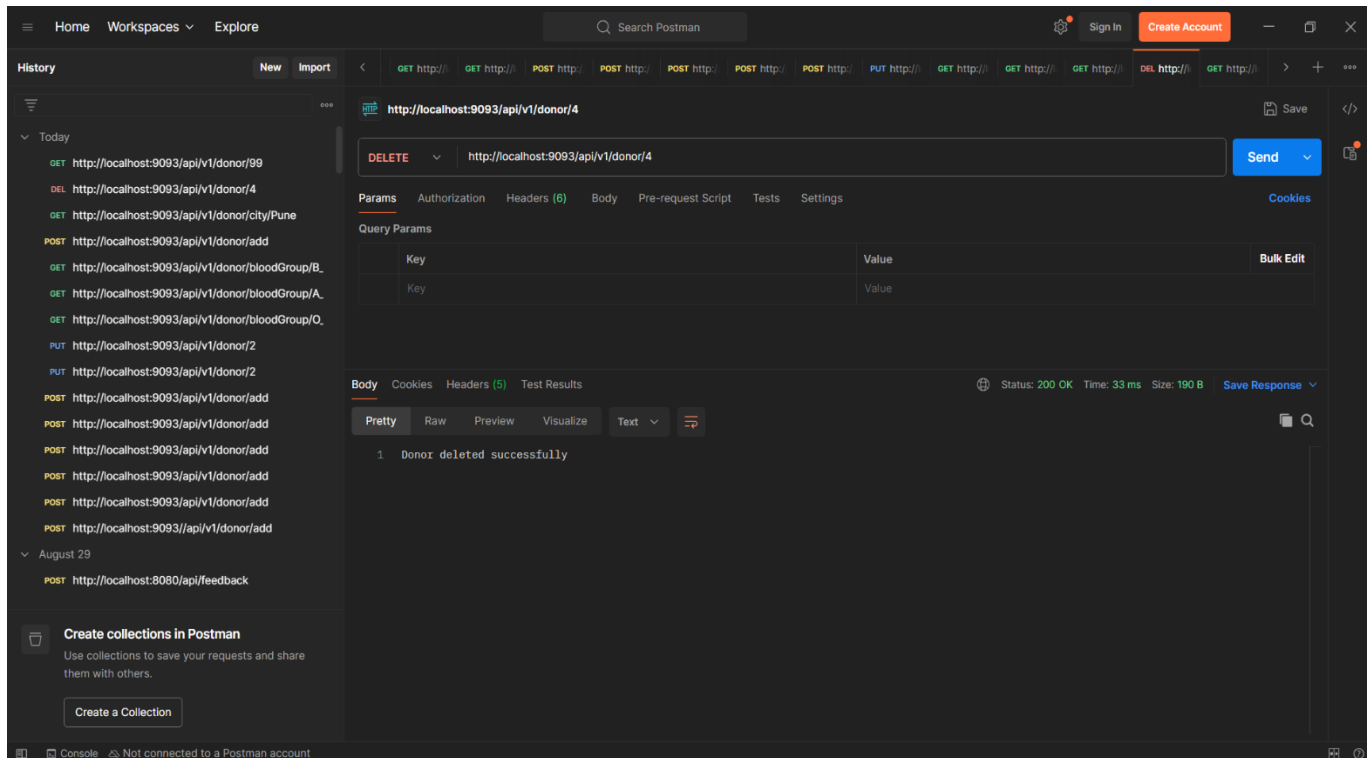
//Getting Donor by cityname



The screenshot shows the Postman interface with a GET request to `http://localhost:9093/api/v1/donor/city/Pune`. The response is a JSON array of two donor objects. The status is 200 OK, time is 10 ms, and size is 399 B.

```
1 {
2   {
3     "id": 1,
4     "name": "sanket",
5     "bloodGroup": "B_Positive",
6     "city": "Pune",
7     "age": 22,
8     "email": "sanket@gmail.com",
9     "gender": "Male"
10  },
11  {
12    "id": 6,
13    "name": "kirti",
14    "bloodGroup": "O_Positive",
15    "city": "Pune",
16    "age": 25,
17    "email": "kirti@gmail.com",
18  }
19 }
```

//delete Donor by Id



The screenshot shows the Postman interface with a DELETE request to `http://localhost:9093/api/v1/donor/4`. The response is a text message: "Donor deleted successfully". The status is 200 OK, time is 33 ms, and size is 190 B.

```
1 Donor deleted successfully
```

//Adding BloodService

The screenshot shows a Postman interface for a POST request to `http://localhost:9094/api/v1/blood/add`. The request body is a JSON object: `{ "bloodGroup": "B_Positive", "status": "Available", "quantity": 5 }`. The response is a 200 OK status with a JSON body: `{ "id": 1, "bloodGroup": "B_Positive", "quantity": 5, "status": "Available" }`. The status bar indicates a successful response with a 200 OK status, 347 ms time, and 233 B size.

```
POST http://localhost:9094/api/v1/blood/add
```

```
{
  "bloodGroup": "B_Positive",
  "status": "Available",
  "quantity": 5
}
```

```
{
  "id": 1,
  "bloodGroup": "B_Positive",
  "quantity": 5,
  "status": "Available"
}
```

Status: 200 OK Time: 347 ms Size: 233 B

//Getting BloodService

The screenshot shows a Postman interface for a GET request to `http://localhost:9094/api/v1/blood/all`. The response is a 200 OK status with a JSON array body containing three blood service records. The status bar indicates a successful response with a 200 OK status, 269 ms time, and 592 B size.

```
GET http://localhost:9094/api/v1/blood/all
```

```
[
  {
    "id": 1,
    "bloodGroup": "B_Positive",
    "quantity": 5,
    "status": "Available"
  },
  {
    "id": 2,
    "bloodGroup": "B_Positive",
    "quantity": 3,
    "status": "Available"
  },
  {
    "id": 3,
    "bloodGroup": "A_Positive",
    "quantity": 5,
    "status": "used"
  }
]
```

Status: 200 OK Time: 269 ms Size: 592 B

HTTP client interface showing a GET request to `http://localhost:9094/api/v1/blood/all`. The response is a JSON array of blood service records.

Request: GET `http://localhost:9094/api/v1/blood/all`

Response Body (JSON):

```
[{"id": 4, "bloodGroup": "O_Positive", "quantity": 5, "status": "Not available"}, {"id": 5, "bloodGroup": "AB_Positive", "quantity": 4, "status": "Not available"}, {"id": 6, "bloodGroup": "O_Negative", "quantity": 1, "status": "Not available"}]
```

Status: 200 OK, Time: 269 ms, Size: 592 B

//Getting BloodService By Id

HTTP client interface showing a GET request to `http://localhost:9094/api/v1/blood/1`. The response is a single blood service record.

Request: GET `http://localhost:9094/api/v1/blood/1`

Response Body (JSON):

```
{"id": 1, "bloodGroup": "B_Positive", "quantity": 5, "status": "Available"}
```

Status: 200 OK, Time: 30 ms, Size: 233 B

//Updating BloodService

The screenshot shows a Postman interface for a PUT request to `http://localhost:9094/api/v1/blood/updateStatus?id=3&status=used`. The request is configured with the following parameters:

Key	Value
id	3
status	used

The response is displayed in the Body tab, showing a JSON object:

```
{  "id": 3,  "bloodGroup": "A_Positive",  "quantity": 5,  "status": "used"}
```

Response details: Status: 200 OK, Time: 32 ms, Size: 228 B.

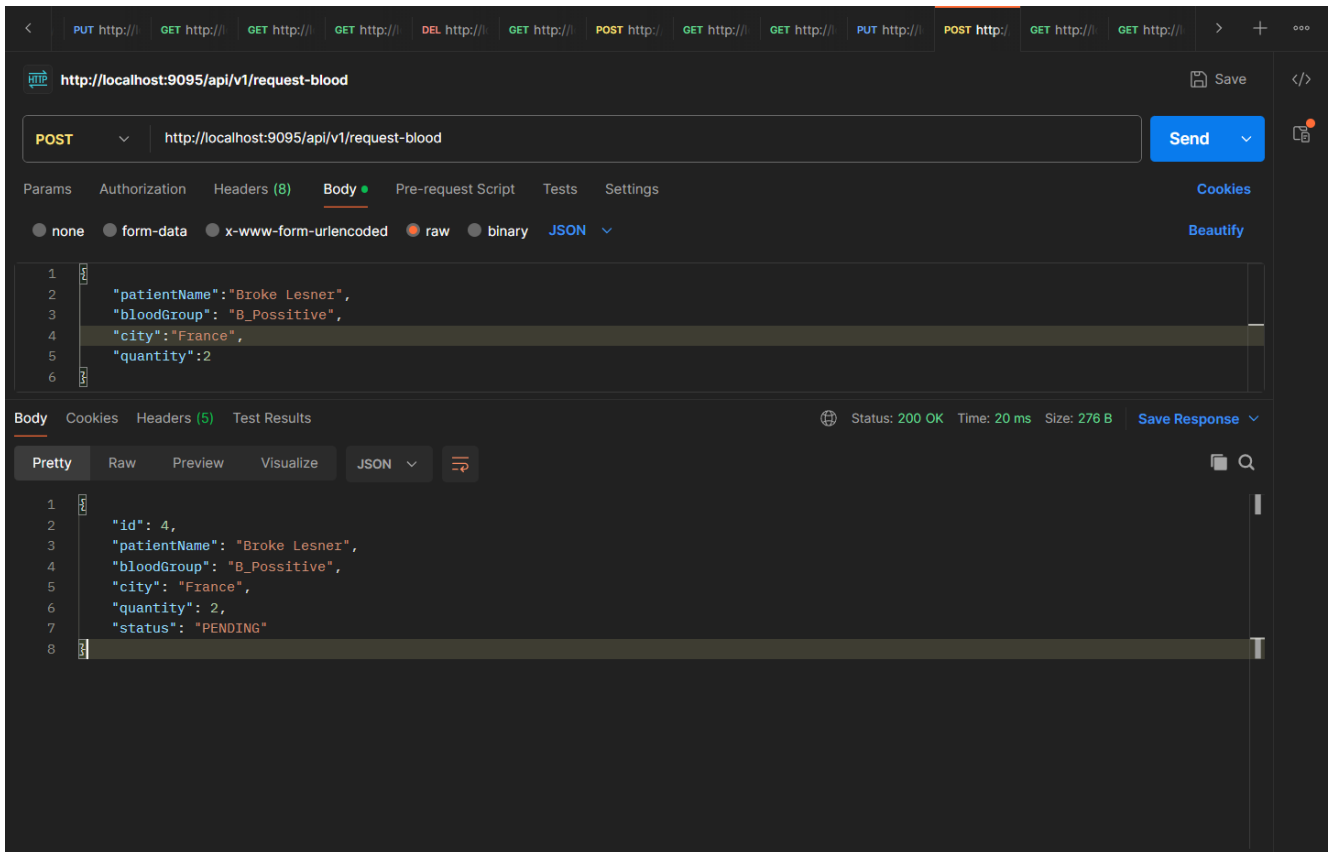
//Linking Blood with Donor and getting Donor by BloodGroup

The screenshot shows a Postman interface for a GET request to `http://localhost:9094/api/v1/blood/donors/B_Positive`. The response is displayed in the Body tab, showing a JSON object:

```
{  "id": 1,  "name": "sanket",  "bloodGroup": "B_Positive",  "city": "Pune",  "age": 22,  "email": "sanket@gmail.com",  "gender": "Male"}
```

Response details: Status: 200 OK, Time: 927 ms, Size: 283 B.

//Creating RequestBlood



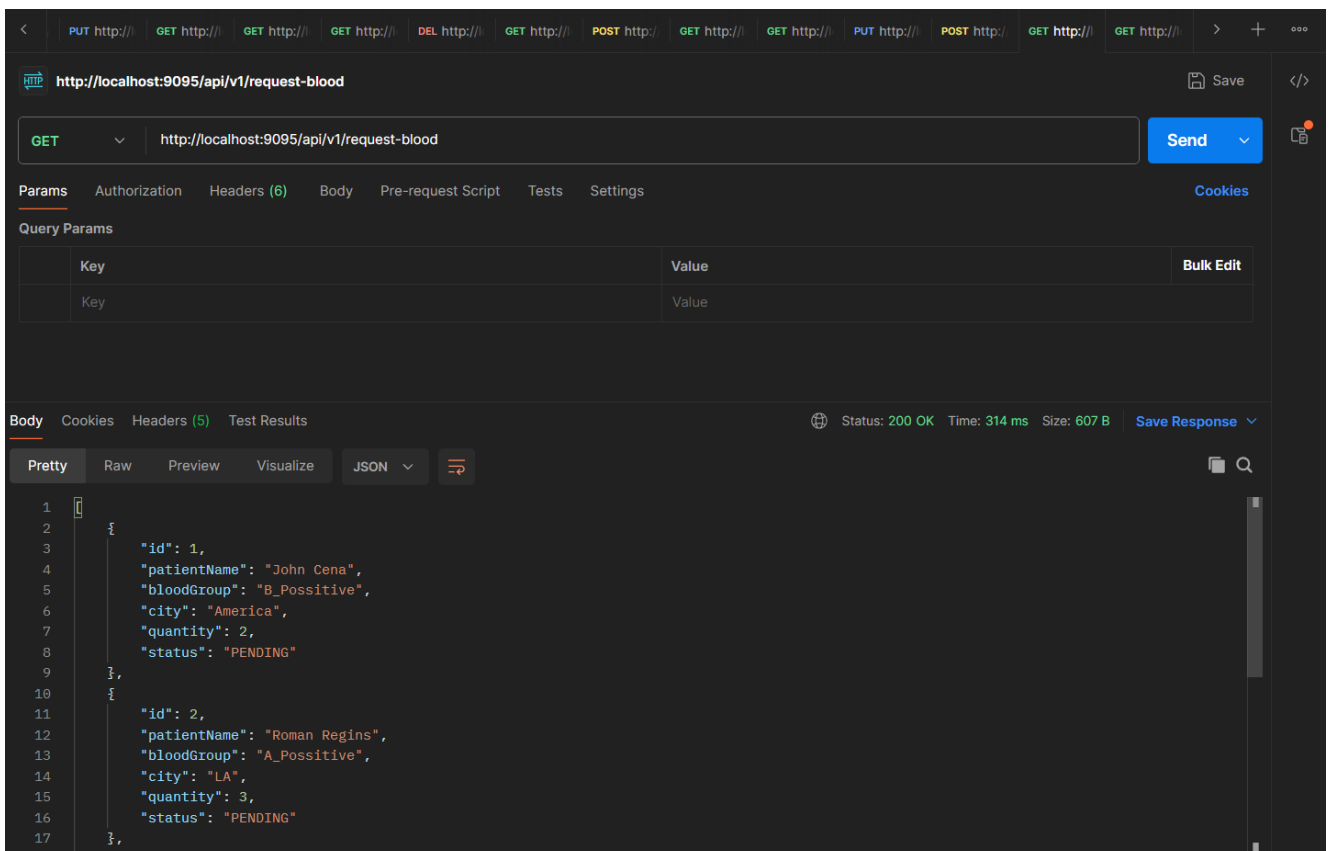
Postman interface showing a POST request to `http://localhost:9095/api/v1/request-blood`. The request body is a JSON object:

```
{  "patientName": "Broke Lesner",  "bloodGroup": "B_Positive",  "city": "France",  "quantity": 2}
```

The response is a JSON object:

```
{  "id": 4,  "patientName": "Broke Lesner",  "bloodGroup": "B_Positive",  "city": "France",  "quantity": 2,  "status": "PENDING"}
```

//Getting RequestBlood



Postman interface showing a GET request to `http://localhost:9095/api/v1/request-blood`. The response is a JSON array of two objects:

```
[  {    "id": 1,    "patientName": "John Cena",    "bloodGroup": "B_Positive",    "city": "America",    "quantity": 2,    "status": "PENDING"  },  {    "id": 2,    "patientName": "Roman Regins",    "bloodGroup": "A_Positive",    "city": "LA",    "quantity": 3,    "status": "PENDING"  }]
```

HTTP client interface showing a GET request to `http://localhost:9095/api/v1/request-blood`. The response status is 200 OK, Time: 314 ms, Size: 607 B.

Query Params

Key	Value
Key	Value

Body (JSON)

```
{
  "id": 3,
  "patientName": "Seth Rollins",
  "bloodGroup": "B_Positive",
  "city": "UK",
  "quantity": 2,
  "status": "PENDING"
},
{
  "id": 4,
  "patientName": "Broke Lesner",
  "bloodGroup": "B_Positive",
  "city": "France",
  "quantity": 2,
  "status": "PENDING"
}
```

//Getting Request Blood By Id

HTTP client interface showing a GET request to `http://localhost:9095/api/v1/request-blood/2`. The response status is 200 OK, Time: 43 ms, Size: 272 B.

Query Params

Key	Value
Key	Value

Body (JSON)

```
{
  "id": 2,
  "patientName": "Roman Regins",
  "bloodGroup": "A_Positive",
  "city": "LA",
  "quantity": 3,
  "status": "PENDING"
}
```


//Linking RequestBlood with Blood and Getting Blood

The screenshot shows a Postman interface with a GET request to `http://localhost:9094/api/v1/blood/all`. The response is a JSON array of three blood items, each with an id, bloodGroup, quantity, and status.

```
1 {
2   "id": 1,
3   "bloodGroup": "B_Positive",
4   "quantity": 5,
5   "status": "Available"
6 },
7 {
8   "id": 2,
9   "bloodGroup": "B_Positive",
10  "quantity": 3,
11  "status": "Available"
12 },
13 {
14   "id": 3,
15   "bloodGroup": "A_Positive",
16   "quantity": 5,
17 }
```

//Creating Notifications

The screenshot shows a Postman interface with a POST request to `http://localhost:9096/notifications`. The request body is a JSON object with `requestId`, `recipient`, and `message`. The response is a JSON object with `id`, `recipient`, `message`, `sentAt`, and `requestId`.

```
1 {
2   "requestId": 4,
3   "recipient": "raj@gmail.com",
4   "message": "Initial message"
5 }
6
```

```
1 {
2   "id": 4,
3   "recipient": "raj@gmail.com",
4   "message": "Request ID 4 is currently PENDING",
5   "sentAt": "2025-09-03T23:30:59.1321699",
6   "requestId": 4
7 }
```

//Promethues:

The screenshot shows the Prometheus web interface at `localhost:9090/targets`. The page displays a list of targets under the 'Status > Target health' tab. Each target is represented by a card showing its name, endpoint, labels, last scrape time, and state.

Target Name	Endpoint	Labels	Last scrape	State
blood-request-service	<code>http://localhost:9095/actuator/prometheus</code>	<code>app="blood-request-service"</code> , <code>instance="localhost:9095"</code> , <code>job="blood-request-service"</code>	11.505s ago, 137ms	up
blood-service	<code>http://localhost:9094/actuator/prometheus</code>	<code>app="blood-service"</code> , <code>instance="localhost:9094"</code> , <code>job="blood-service"</code>	11.657s ago, 83ms	up
donor-service	<code>http://localhost:9093/actuator/prometheus</code>	<code>app="donor-service"</code> , <code>instance="localhost:9093"</code> , <code>job="donor-service"</code>	11.787s ago, 110ms	up
notification-service	<code>http://localhost:9096/actuator/prometheus</code>	<code>app="notification-service"</code> , <code>instance="localhost:9096"</code> , <code>job="notification-service"</code>	4.057s ago, 131ms	up

//Grafana:

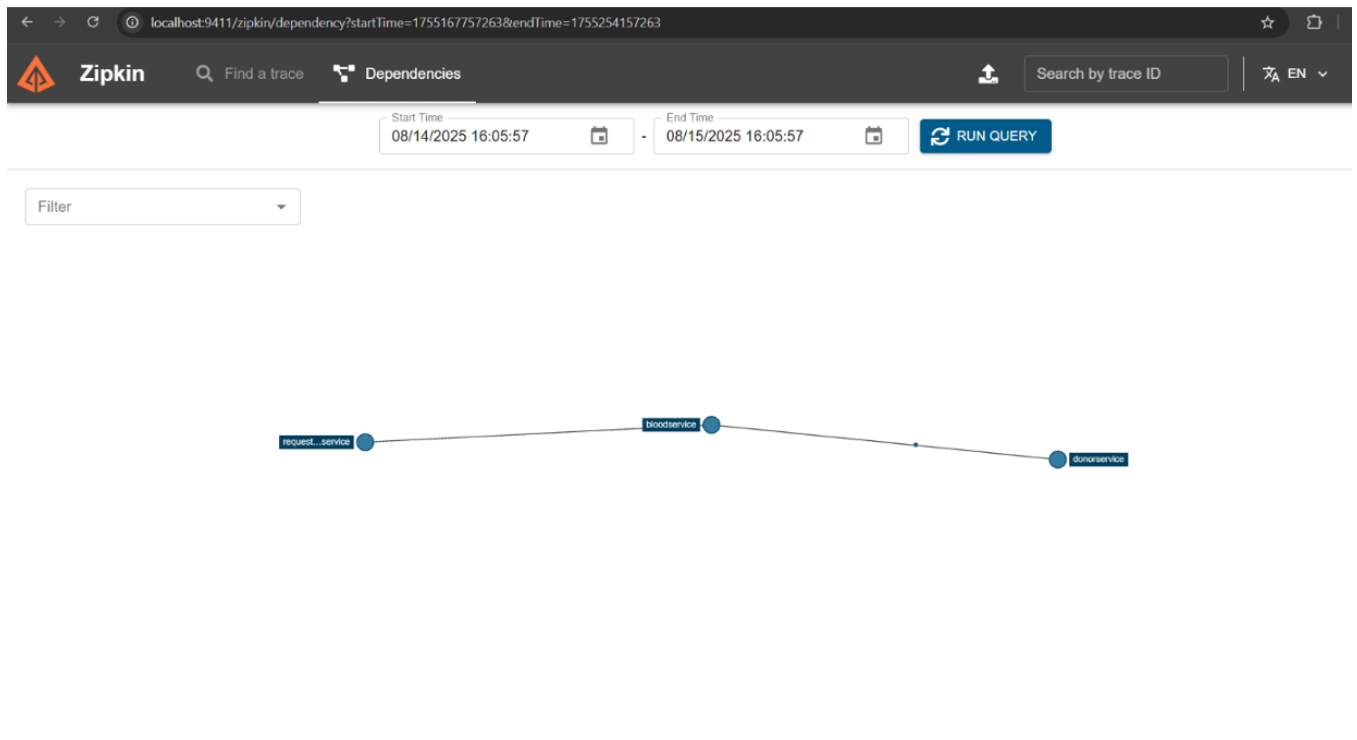
The screenshot shows the Grafana web interface at `localhost:3000/connections/datasources/edit/deuum2vz4035se/`. The page is titled 'prometheus - Data sources - C...' and shows the configuration for a Prometheus data source. The left sidebar contains navigation links for Home, Bookmarks, Starred, Dashboards, Explore, Drilldown, Alerting, Connections, Add new connection, Data sources, Administration, Provisioning, General, Plugins and data, Users and access, and Authentication.

The main content area shows the configuration for the 'prometheus' data source. It includes options for 'Incremental querying (beta)' and 'Disable recording rules (beta)', both of which are disabled. Under the 'Other' section, there are settings for 'Custom query parameters' (Example: `max_source_resolution=5m&tpin`), 'HTTP method' (POST), 'Series limit' (40000), and 'Use series endpoint' (disabled). There is also an 'Exemplars' section with a '+ Add' button.

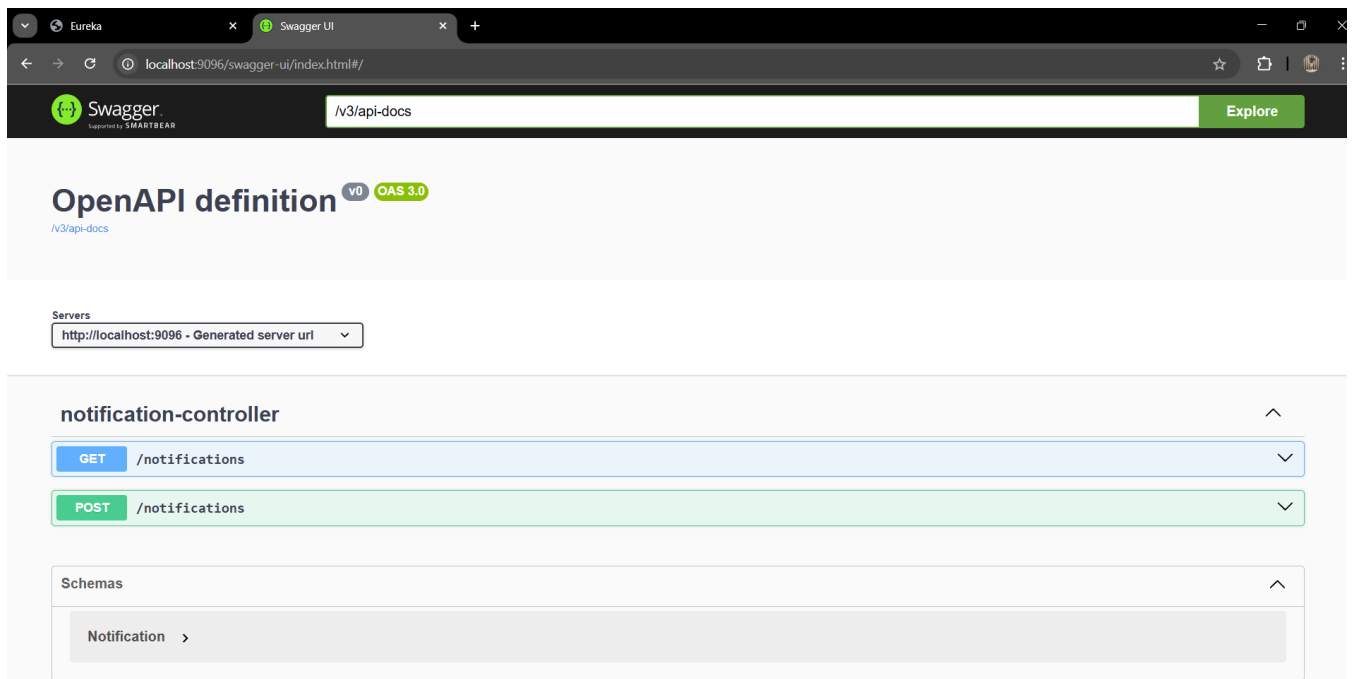
A success message at the bottom states: 'Successfully queried the Prometheus API. Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view.'

At the bottom of the configuration page, there are buttons for 'Delete' and 'Save & test'.

//Zipkin:



//Swagger:



//Swagger api/docs

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "OpenAPI definition",
    "version": "v0"
  },
  "servers": [
    {
      "url": "http://localhost:9096",
      "description": "Generated server url"
    }
  ],
  "paths": {
    "/notifications": {
      "get": {
        "tags": [
          "notification-controller"
        ],
        "operationId": "getAllNotifications",
        "responses": {
          "200": {
            "description": "OK",
            "content": {
              "*/*": {
                "schema": {
                  "type": "array",
                  "items": {
                    "$ref": "#/components/schemas/Notification"
                  }
                }
              }
            }
          }
        }
      },
      "post": {
        "tags": [
          "notification-controller"
        ],
        "operationId": "sendNotification",
        "requestBody": {
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Notification"
              }
            }
          }
        }
      }
    }
  }
}
```

```
    },
    "components": {
      "schemas": {
        "Notification": {
          "required": [
            "message",
            "recipient"
          ],
          "type": "object",
          "properties": {
            "id": {
              "type": "integer",
              "format": "int64"
            },
            "recipient": {
              "type": "string"
            },
            "message": {
              "type": "string"
            },
            "sentAt": {
              "type": "string",
              "format": "date-time"
            },
            "requestId": {
              "type": "integer",
              "format": "int64"
            }
          }
        }
      }
    }
  }
}
```

// Eureka Server:

Eureka

Search tabs

localhost:8761

☆ | 📄 | 📌 | ⌵

System Status

Environment	test	Current time	2025-09-03T21:43:45 +0530
Data center	default	Uptime	00:05
		Lease expiration enabled	false
		Renews threshold	11
		Renews (last min)	8

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - Sanket-Bondre:API-GateWay:8765
BLOODSERVICE	n/a (1)	(1)	UP (1) - Sanket-Bondre:BloodService:9094
CONFIGSERVER	n/a (1)	(1)	UP (1) - Sanket-Bondre:ConfigServer:9091
DONORSERVICE	n/a (1)	(1)	UP (1) - Sanket-Bondre:DonorService:9093
NOTIFICATIONSERVICE	n/a (1)	(1)	UP (1) - Sanket-Bondre:NotificationService:9096
REQUESTBLOODSERVICE	n/a (1)	(1)	UP (1) - Sanket-Bondre:RequestBloodService:9095