

Gestión de Historial Médico

Base de Datos NO SQL

Docente:

- Andrés Pastorini

Integrantes:

- Matías Llull
- Macarena Arambillety
- Luis Olivo

Introducción

Este proyecto consiste en la implementación de una solución backend para gestionar el historial médico de pacientes utilizando servicios REST y una base de datos NoSQL (en nuestro caso MongoDB). La API permite almacenar y consultar los datos de los pacientes, agregar registros médicos asociados, y realizar búsquedas filtradas por diferentes criterios.

La solución está diseñada para almacenar los datos en formato JSON y emplea MongoDB como sistema de gestión de base de datos NoSQL. El desarrollo se realizó en Python y se utilizó Visual Studio Code como entorno de desarrollo.

Tecnologías Utilizadas

- **Lenguaje de programación:** Python 3.13.0
- **Framework:** Flask (para la creación de la API REST)
- **Base de Datos:** MongoDB 8.0.1 (Base de Datos NoSQL)
- **IDE:** Visual Studio Code
- **Docker** (contenerización de la aplicación y la base de datos)
- **JMeter 5.6.3** (prueba de carga contra los servicios REST, y análisis de su comportamiento).
- **Postman:** Para pruebas manuales y automatizadas de la API

Requerimientos Funcionales

La solución ofrece los siguientes servicios REST:

1. Agregar Paciente

- **Método:** POST /pacientes
- **Descripción:** Permite agregar un nuevo paciente al sistema.
- **Cuerpo de la solicitud:**

Mensaje Éxito (201) Paciente agregado:

The screenshot displays a REST client interface with the following details:

- URL:** http://127.0.0.1:5000/pacientes
- Method:** POST
- Request Body (JSON):**

```
{  "ci": "1456780457",  "nombre": "Manuel",  "apellido": "Lima",  "fecha_nacimiento": "2005-02-07",  "sexo": "Masculino"}
```
- Response Status:** 201 CREATED
- Response Time:** 12 ms
- Response Size:** 327 B
- Response Body (JSON):**

```
{  {    "_id": "1456780457",    "apellido": "Lima",    "fecha_nacimiento": "2005-02-07",    "nombre": "Manuel",    "sexo": "Masculino"  },  {    "mensaje": "Paciente agregado correctamente"  }}
```

Mensaje Error (401) paciente ya existe:

[HTTP](#) <http://127.0.0.1:5000/pacientes> [Save](#) [</>](#)

POST [v](#)

<http://127.0.0.1:5000/pacientes>

Send [v](#)

[v](#)

Params Auth Headers (8) **Body** [Pre-req.](#) [Tests](#) [Settings](#) [Cookies](#)

raw [v](#) [JSON](#) [v](#) [Beautify](#)

```
1  {
2    "ci": "1456780457",
3    "nombre": "Manuel",
4    "apellido": "Lima",
5    "fecha_nacimiento": "2005-02-07",
6    "sexo": "Masculino"
7  }
8
```

Body [Cookies](#) [Headers \(5\)](#) [Test Results](#) [v](#) [401 UNAUTHORIZED](#) [6 ms](#) [211 B](#) [Save Response](#) [v](#)

[Pretty](#) [Raw](#) [Preview](#) [Visualize](#) [JSON](#) [v](#) [v](#)

```
1  {
2    "mensaje": "El paciente ya existe"
3  }
```

2. Agregar Registro Médico, asociado al paciente

- **Método:** POST `/pacientes/*CI*/registros`
- **Descripción:** Permite agregar un nuevo registro médico asociado a un paciente.
- **Cuerpo de la solicitud**

Mensaje Éxito (201) Registro de Médico asociado a Paciente agregado:

The screenshot displays a REST client interface with the following details:

- URL:** `http://127.0.0.1:5000/pacientes/2000200455/registros`
- Method:** POST
- Body (Request):**

```
{  "fecha": "2024-11-16",  "tipo": "Consulta",  "diagnostico": "Gripe",  "medico": "Dr. Borlas",  "institucion": "Hospital Central del Peru",  "descripcion": "El paciente presenta síntomas tos seca y fiebre.",  "medicacion": "Paracetamol"}
```
- Status:** 201 CREATED
- Time:** 10 ms
- Size:** 514 B
- Body (Response):**

```
{  "_id": "673b763ddb1d37c150249682",  "descripcion": "El paciente presenta síntomas tos seca y fiebre.",  "diagnostico": "Gripe",  "fecha": "2024-11-16",  "institucion": "Hospital Central del Peru",  "medicacion": "Paracetamol",  "medico": "Dr. Borlas",  "pacienteCI": "1456780457",  "tipo": "Consulta"},  {  "mensaje": "Registro médico agregado correctamente"}
```

Mensaje Error (402) el paciente no existe en el sistema:

HTTP

http://127.0.0.1:5000/pacientes/1234567890/registros

Save

</>

GET

http://127.0.0.1:5000/pacientes/1456780777/registros

Send

Params

Auth

Headers (8)

Body

Pre-req.

Tests

Settings

Cookies

Beautiful

raw

JSON

1

2

3

4

5

6

7

8

9

10

"fecha": "2024-11-16",

"tipo": "Consulta",

"diagnostico": "Gripe",

"medico": "Dr. Borlas",

"institucion": "Hospital Central de Bolivar",

"descripcion": "El paciente presenta síntomas tos seca y fiebre.",

"medicacion": "Paracetamol"

Body

Cookies

Headers (5)

Test Results

404 NOT FOUND

7 ms

242 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

"mensaje": "No existe un paciente con la cédula proporcionada."

3. Consultar Historial Médico

- **Método:** GET /pacientes/{ci}/registros
- **Descripción:** Permite consultar todos los registros médicos asociados a un paciente.
- **Respuesta Exitosa (200):** trae todos los registros del paciente

The screenshot shows a REST client interface with the following components:

- URL Bar:** `http://127.0.0.1:5000/pacientes/2000200457/registros`
- Method:** GET
- Send Button:** A blue button labeled "Send".
- Response Body:** A large empty text area for the response body.
- Response Status:** 200 OK, 5 ms, 451 B.
- Response Format:** JSON (selected).
- Response Content:** A JSON object representing a medical record.

```
1 {
2   "_id": "673b763ddb1d37c150249682",
3   "descripcion": "El paciente presenta síntomas tos seca y fiebre.",
4   "diagnostico": "Gripe",
5   "fecha": "2024-11-16",
6   "institucion": "Hospital Central del Peru",
7   "medicacion": "Paracetamol",
8   "medico": "Dr. Borlas",
9   "pacienteCI": "1456780457",
10  "tipo": "Consulta"
11 }
12
13 }
```

Error (402) no existe el paciente con la CÉDULA proporcionada:

GET http://127.0.0.1:5000/pacientes/2000200457/registros [CONFLICT] GET [CONFLICT] POST GET http://127.0.0.1:5000/pacientes/1456780458/registros POST http://127.0.0.1:5000/pacientes/1456780458/registros GET http://127.0.0.1:5000/pacientes/1456780458/registros GET http://127.0.0.1:5000/pacientes/1456780458/registros + ...

HTTP http://127.0.0.1:5000/pacientes/2000200457/registros Save </>

GET http://127.0.0.1:5000/pacientes/1456780458/registros Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

raw JSON Beautify

1

Body Cookies Headers (5) Test Results 404 NOT FOUND 9 ms 242 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2  "mensaje": "No existe un paciente con la cédula proporcionada."
3
```


4. Obtener Registros por Criterio

- **Método:** `GET /registros`
- **Descripción:** Permite obtener registros médicos por un criterio de búsqueda como tipo, diagnóstico, médico o institución.
- **Parámetros de Búsqueda:**
 - `tipo`: Consulta, Examen, Internación
 - `diagnostico`
 - `medico`
 - `institucion`

Respuesta Exitosa (200): según criterio DIAGNOSTICO

GET http://127.0.0.1:5000/registros

HTTP http://127.0.0.1:5000/registros?diagnostico=Gripe Save </>

GET http://127.0.0.1:5000/registros?diagnostico=Gripe Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Body Cookies Headers (5) Test Results 200 OK 8 ms 729 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "_id": "673a1ad6d483f16a07e8397b",
4     "descripcion": "El paciente presenta síntomas tos constante.",
5     "diagnostico": "Gripe",
6     "fecha": "2024-11-17",
7     "institucion": "Hospital Central del Peru",
8     "medicacion": "Paracetamol",
9     "medico": "Dr. Borja",
10    "pacienteCI": "2000200455",
11    "tipo": "Consulta"
12  },
13  {
14    "_id": "673b763ddb1d37c150249682",
15    "descripcion": "El paciente presenta síntomas tos seca y fiebre.",
16    "diagnostico": "Gripe",
17    "fecha": "2024-11-16",
18    "institucion": "Hospital Central del Peru",
19    "medicacion": "Paracetamol",
20    "medico": "Dr. Borlas",
21    "pacienteCI": "1456780457",
22    "tipo": "Consulta"
23  }
24 ]
```

Respuesta Exitosa (200):según criterio TIPO y MEDICO

HTTP <http://127.0.0.1:5000/registros?diagnostico=Gripe> Save </>

GET ▼ <http://127.0.0.1:5000/registros?tipo=Consulta&&medico=Dr. Borlas> Send ▼ 📄

Params ● Auth Headers (6) Body Pre-req. Tests Settings Cookies

Body Cookies Headers (5) Test Results 🌐 200 OK 9 ms 451 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ 🔍

```
1 {
2   "_id": "673b763ddb1d37c150249682",
3   "descripcion": "El paciente presenta síntomas tos seca y fiebre.",
4   "diagnostico": "Gripe",
5   "fecha": "2024-11-16",
6   "institucion": "Hospital Central del Peru",
7   "medicacion": "Paracetamol",
8   "medico": "Dr. Borlas",
9   "pacienteCI": "1456780457",
10  "tipo": "Consulta"
11 }
12
13
```

Justificación de la Base de Datos NoSQL

Se ha optado por **MongoDB** como la base de datos NoSQL debido a las siguientes razones:

- **Escalabilidad:** MongoDB es adecuado para manejar grandes volúmenes de datos, lo cual es fundamental dado que el sistema puede crecer con el tiempo.
- **Flexibilidad en el esquema:** MongoDB no requiere una estructura fija, lo que es útil dado que los registros médicos pueden tener diferentes atributos y no todos los pacientes o registros siguen un formato exacto.
- **Alto rendimiento:** MongoDB proporciona un rendimiento adecuado para consultas rápidas y eficientes sobre grandes volúmenes de datos.

Modelo Lógico de la Base de Datos

La base de datos está estructurada en dos colecciones principales, las cuales son:

Colección de Pacientes

Cada documento en la colección de pacientes tiene los siguientes campos:

- **ci:** Clave primaria, identificación única del paciente (String)
- **nombre:** Nombre del paciente (String)
- **apellido:** Apellido del paciente (String)
- **fecha_nacimiento:** Fecha de nacimiento del paciente (Date)
- **sexo:** Género del paciente (String)

Colección de Registros Médicos

Cada documento en la colección de registros médicos tiene los siguientes campos:

- **fecha:** Fecha del registro médico (Date)
- **tipo:** Tipo de registro (Consulta, Examen, Internación) (String)
- **diagnostico:** Diagnóstico médico (String)
- **medico:** Nombre del médico (String)
- **institucion:** Nombre de la institución médica (String)
- **descripcion:** Descripción adicional (opcional) (String)
- **medicacion:** Medicación prescrita (opcional) (String)

Instalación y Configuración

Requisitos previos

- Visual Studio Code (recomendado como IDE)
- Python 3.13.0
- MongoDB 8.0.1 (local)
- Docker (elegida)
- JMeter 5.6.3

Pasos de Instalación:

1. Se generó el repositorio, luego nos conectamos remotamente.

```
git remote add origin git@github.com:PrinceChaos7/NOsql.git
```

2. Instalar las dependencias, entorno virtual para instalar las dependencias, necesarios en python:

```
python -m venv venv
source venv/bin/activate           # En Linux, ya que algunos lo utilizamos.
.\venv\Scripts\activate           # En Windows, otros optamos por este.
pip install -r requirements.txt
```

3. **Configurar MongoDB** usando Docker, levantar MongoDB con Docker Compose:

```
docker-compose up --build
```

4. **Ejecutar la aplicación** Una vez configurado todo, puedes ejecutar el servidor de desarrollo:

```
python app.py
```

Pruebas con Postman

Se han creado colecciones de Postman para facilitar las pruebas de la API.

Opcionales Realizados

1. **Dockerización:** Se ha incluido un archivo `docker-compose.yml` para contenerizar la aplicación y MongoDB.
2. **JMeter:** se realizó la prueba de carga y ver su comportamiento.
3. **Pruebas con Postman:** Se han generado casos de prueba para validar el correcto funcionamiento de los endpoints de la API.

Pruebas con Docker

```
C:\Users\XXX>docker exec -it mongo_db /bin/bash
root@544b58d1d275:/# mongosh
Current Mongosh Log ID: 673a18985c73718f47c1c18b
Connecting to:
mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.3
Using MongoDB:      8.0.3
Using Mongosh:      2.3.3
```

```
test> show dbs
admin          40.00 KiB
config         92.00 KiB
historialMedicoDB 144.00 KiB
local          72.00 KiB
```

```
test> docker ps
```

```
C:\Users\XXX>docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
PORTS         NAMES
b38efd738799   historial_medico-flask-app          "python app.py"         48 minutes ago Up 47
minutes       0.0.0.0:5000->5000/tcp               historial_medico-flask-app
544b58d1d275   mongo:latest                        "docker-entrypoint.s..." 48 minutes ago Up 47
minutes       0.0.0.0:27017->27017/tcp             mongo_db
```

Comprobacion:

```
C:\Users\XXX>docker logs historial_medico-flask-app
```

```
* Serving Flask app 'app'  
* Debug mode: off
```

```
* Running on all addresses (0.0.0.0)  
* Running on http://127.0.0.1:5000  
* Running on http://172.19.0.3:5000
```

```
172.19.0.1 - - [17/Nov/2024 15:51:29] "POST /pacientes HTTP/1.1" 201 -  
172.19.0.1 - - [17/Nov/2024 15:51:39] "GET /registros?tipo=Consulta HTTP/1.1" 200 -  
172.19.0.1 - - [17/Nov/2024 15:51:42] "GET /pacientes/123456789/registros HTTP/1.1" 404 -  
172.19.0.1 - - [17/Nov/2024 15:51:51] "GET /pacientes/1234567890/registros HTTP/1.1" 404 -  
172.19.0.1 - - [17/Nov/2024 15:51:55] "GET /pacientes/1234567891/registros HTTP/1.1" 404 -  
172.19.0.1 - - [17/Nov/2024 15:52:13] "GET /pacientes/9898989832/registros HTTP/1.1" 200 -  
172.19.0.1 - - [17/Nov/2024 15:52:41] "GET /registros?tipo=Consulta HTTP/1.1" 200 -  
172.19.0.1 - - [17/Nov/2024 15:52:54] "GET /registros?tipo=Consulta HTTP/1.1" 200 -  
172.19.0.1 - - [17/Nov/2024 15:52:58] "GET /pacientes/1234567890/registros HTTP/1.1" 404 -  
172.19.0.1 - - [17/Nov/2024 15:53:10] "GET /pacientes/1234567890/registros HTTP/1.1" 404 -  
172.19.0.1 - - [17/Nov/2024 15:53:17] "GET /pacientes/1234567890/registros HTTP/1.1" 404 -  
172.19.0.1 - - [17/Nov/2024 15:54:04] "POST /pacientes/9898989832/registros HTTP/1.1" 201 -  
172.19.0.1 - - [17/Nov/2024 15:55:40] "GET /pacientes/1122334455/registros HTTP/1.1" 404 -  
172.19.0.1 - - [17/Nov/2024 15:55:52] "GET /pacientes/1122334455/registros HTTP/1.1" 404 -  
172.19.0.1 - - [17/Nov/2024 15:56:19] "POST /pacientes HTTP/1.1" 201 -  
172.19.0.1 - - [17/Nov/2024 15:56:35] "POST /pacientes/1122334455/registros HTTP/1.1" 201 -  
172.19.0.1 - - [17/Nov/2024 15:56:49] "GET /pacientes/1122334455/registros HTTP/1.1" 200 -  
172.19.0.1 - - [17/Nov/2024 15:57:30] "GET /registros?tipo=Consulta HTTP/1.1" 200 -  
172.19.0.1 - - [17/Nov/2024 15:57:39] "GET /registros?tipo=Consulta HTTP/1.1" 200 -  
172.19.0.1 - - [17/Nov/2024 15:58:10] "GET /registros?tipo=Examen HTTP/1.1" 200 -
```

172.19.0.1 - - [17/Nov/2024 15:58:36] "GET /registros?tipo=Examen HTTP/1.1" 200 -

172.19.0.1 - - [17/Nov/2024 16:01:01] "POST /pacientes HTTP/1.1" 201 -

172.19.0.1 - - [17/Nov/2024 16:01:11] "POST /pacientes HTTP/1.1" 201 -

172.19.0.1 - - [17/Nov/2024 16:01:21] "POST /pacientes/1234567890/registros HTTP/1.1" 201 -

172.19.0.1 - - [17/Nov/2024 16:01:26] "POST /registros?tipo=Consulta HTTP/1.1" 405 -

172.19.0.1 - - [17/Nov/2024 16:01:50] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:02:13] "GET /pacientes/1234567890/registros HTTP/1.1" 200 -

172.19.0.1 - - [17/Nov/2024 16:02:19] "GET /registros?tipo=Consulta HTTP/1.1" 200 -

172.19.0.1 - - [17/Nov/2024 16:02:39] "GET /registros?tipo=Consulta HTTP/1.1" 200 -

172.19.0.1 - - [17/Nov/2024 16:02:59] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:03:14] "POST /pacientes HTTP/1.1" 201 -

172.19.0.1 - - [17/Nov/2024 16:03:34] "POST /pacientes/9876543210/registros HTTP/1.1" 201 -

172.19.0.1 - - [17/Nov/2024 16:03:40] "POST /pacientes/1122334455/registros HTTP/1.1" 201 -

172.19.0.1 - - [17/Nov/2024 16:03:45] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:03:48] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:03:52] "POST /pacientes/1234567890/registros HTTP/1.1" 201 -

172.19.0.1 - - [17/Nov/2024 16:03:57] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:04:00] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:04:04] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:04:07] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:04:24] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:04:28] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:04:36] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:04:43] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:04:57] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:05:01] "POST /pacientes HTTP/1.1" 409 -

172.19.0.1 - - [17/Nov/2024 16:05:06] "POST /pacientes HTTP/1.1" 409 -
172.19.0.1 - - [17/Nov/2024 16:05:16] "POST /pacientes HTTP/1.1" 409 -
172.19.0.1 - - [17/Nov/2024 16:05:27] "POST /pacientes HTTP/1.1" 409 -
172.19.0.1 - - [17/Nov/2024 16:08:11] "POST /pacientes/1234567890/registros HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:08:25] "POST /pacientes/9876543210/registros HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:08:34] "POST /pacientes/1122334455/registros HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:08:43] "POST /pacientes/1234567890/registros HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:08:55] "POST /pacientes/9898989832/registros HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:09:00] "POST /pacientes/1122334455/registros HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:09:11] "POST /pacientes/1234567890/registros HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:09:21] "POST /pacientes/9876543210/registros HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:09:23] "POST /pacientes/1122334455/registros HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:09:27] "POST /pacientes/1234567890/registros HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:12:54] "GET /registros?tipo=Consulta HTTP/1.1" 200 -
172.19.0.1 - - [17/Nov/2024 16:14:47] "POST /pacientes/1234567890/registros HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:15:04] "GET /registros?tipo=Consulta HTTP/1.1" 200 -
172.19.0.1 - - [17/Nov/2024 16:15:48] "GET /registros?tipo=Examen HTTP/1.1" 200 -
172.19.0.1 - - [17/Nov/2024 16:17:48] "GET /registros?diagnostico=Gripe HTTP/1.1" 200 -
172.19.0.1 - - [17/Nov/2024 16:18:12] "GET /pacientes/9876543210/registros HTTP/1.1" 200 -
172.19.0.1 - - [17/Nov/2024 16:18:14] "GET /pacientes/1234565555/registros HTTP/1.1" 404 -
172.19.0.1 - - [17/Nov/2024 16:18:17] "GET /pacientes/1234567891/registros HTTP/1.1" 200 -
172.19.0.1 - - [17/Nov/2024 16:18:19] "GET /pacientes/1234567890/registros HTTP/1.1" 200 -
172.19.0.1 - - [17/Nov/2024 16:31:56] "POST /pacientes HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:33:26] "POST /pacientes/2000200455/registros HTTP/1.1" 201 -
172.19.0.1 - - [17/Nov/2024 16:33:38] "GET /registros?diagnostico=Gripe HTTP/1.1" 200 -
172.19.0.1 - - [17/Nov/2024 16:34:06] "GET /registros?diagnostico=Gripe HTTP/1.1" 200 -

Generando la imagen Docker:

```
C:\Users\XXXe\Desktop\historial_medico>docker-compose up --build
```

```
[+] Building 1.3s (10/10) FINISHED
```

```
docker:desktop-linux
```

```
=> [flask-app internal] load build definition from Dockerfile
```

```
0.0s
```

```
=> => transferring dockerfile: 399B
```

```
0.0s
```

```
=> [flask-app internal] load metadata for docker.io/library/python:3.12-slim
```

```
0.6s
```

```
=> [flask-app internal] load .dockerignore
```

```
0.0s
```

```
=> => transferring context: 73B
```

```
0.0s
```

```
=> [flask-app 1/4] FROM docker.io/library/python:3.12-
```

```
slim@sha256:2a6386ad2db20e7f55073f69a98d6da2cf9f168e05e7487d2670baeb9b 0.1s
```

```
=> => resolve docker.io/library/python:3.12-
```

```
slim@sha256:2a6386ad2db20e7f55073f69a98d6da2cf9f168e05e7487d2670baeb9b7601c5
```

```
0.0s
```

```
=> [flask-app internal] load build context
```

```
0.2s
```

```
=> => transferring context: 149.39kB
```

```
0.1s
```

```
=> CACHED [flask-app 2/4] WORKDIR /app
```

```
0.0s
```

```
=> CACHED [flask-app 3/4] COPY . /app
```

```
0.0s
```

```
=> CACHED [flask-app 4/4] RUN pip install --no-cache-dir -r requirements.txt
```

```
0.0s
```

```
=> [flask-app] exporting to image
```

```
0.2s
```

```
=> => exporting layers
```

```
0.0s
```

=> => exporting manifest

sha256:8410a642f1c1a73ef4d878cf1f9c729b9f0b032646a8a3d9d48f7db6438005f6

0.0s

=> => exporting config

sha256:b4766b529a86866524a4d8429f52b0bd9f2465cb9c43f9563fcb47bd6e0441c2

0.0s

=> => exporting attestation manifest

sha256:36f80bf0519dc00969218a0e405b6b42e8bf044955b90b9e64aa2f048c7fa7b9

0.1s

=> => exporting manifest list

sha256:c4cbcd5036c9bc7d5f8bc79adcbbf48e7aa69525b1ec1b673a0c45d8bb6e394a

0.0s

=> => naming to docker.io/library/historial_medico-flask-app:latest

0.0s

=> => unpacking to docker.io/library/historial_medico-flask-app:latest

0.0s

=> [flask-app] resolving provenance for metadata file

0.0s

[+] Running 3/3

✓ Network historial_medico_historial_medico_network Created

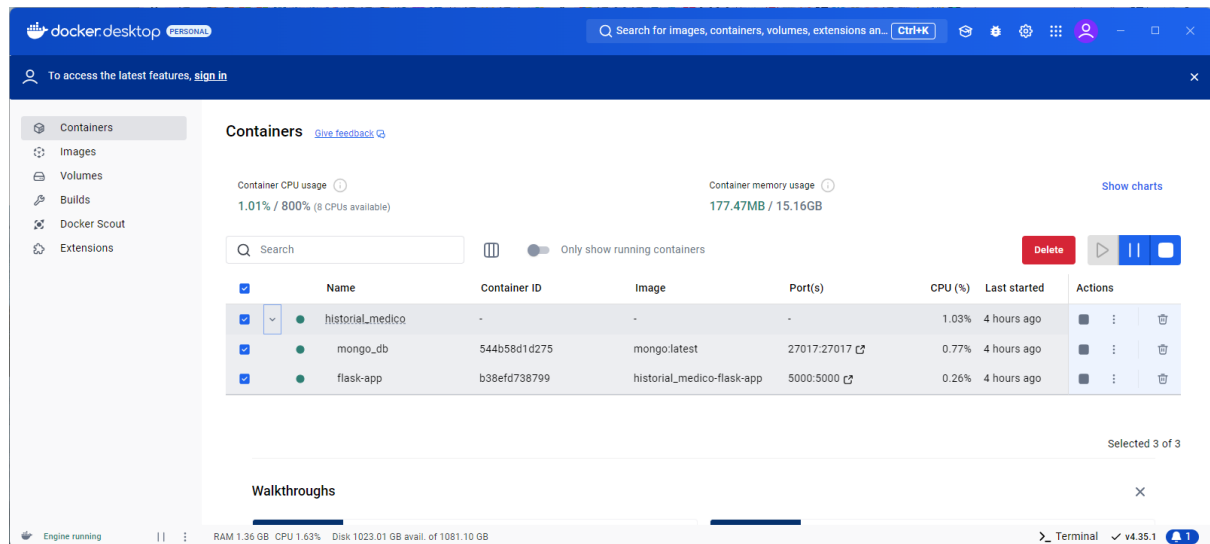
0.1s

✓ Container mongo_db Created

0.2s

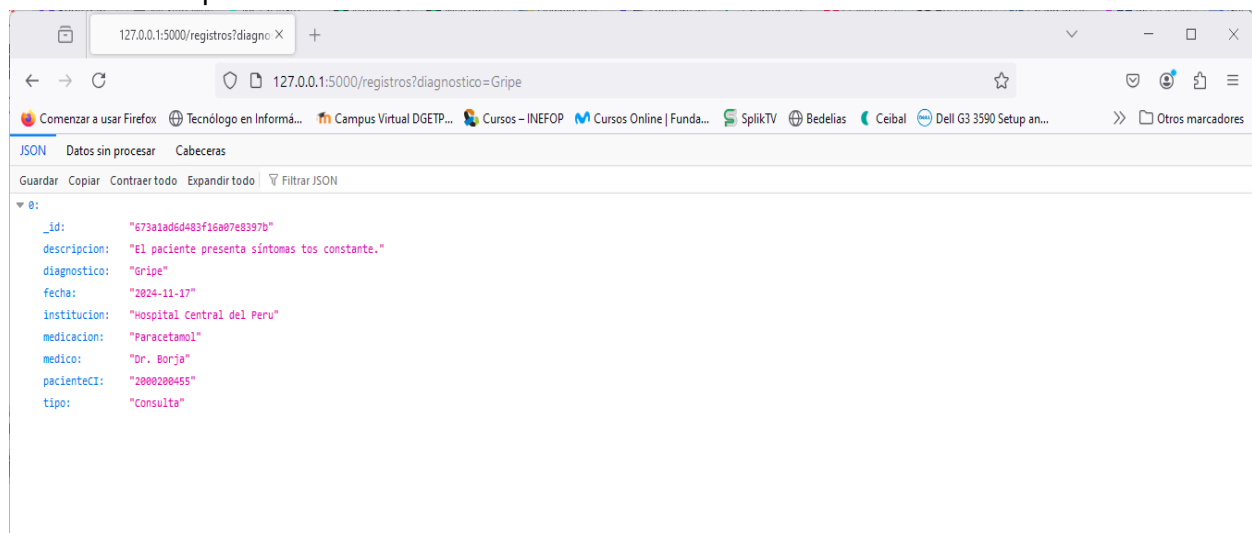
✓ Container historial_medico-flask-app Created

Imanen DOCKER:

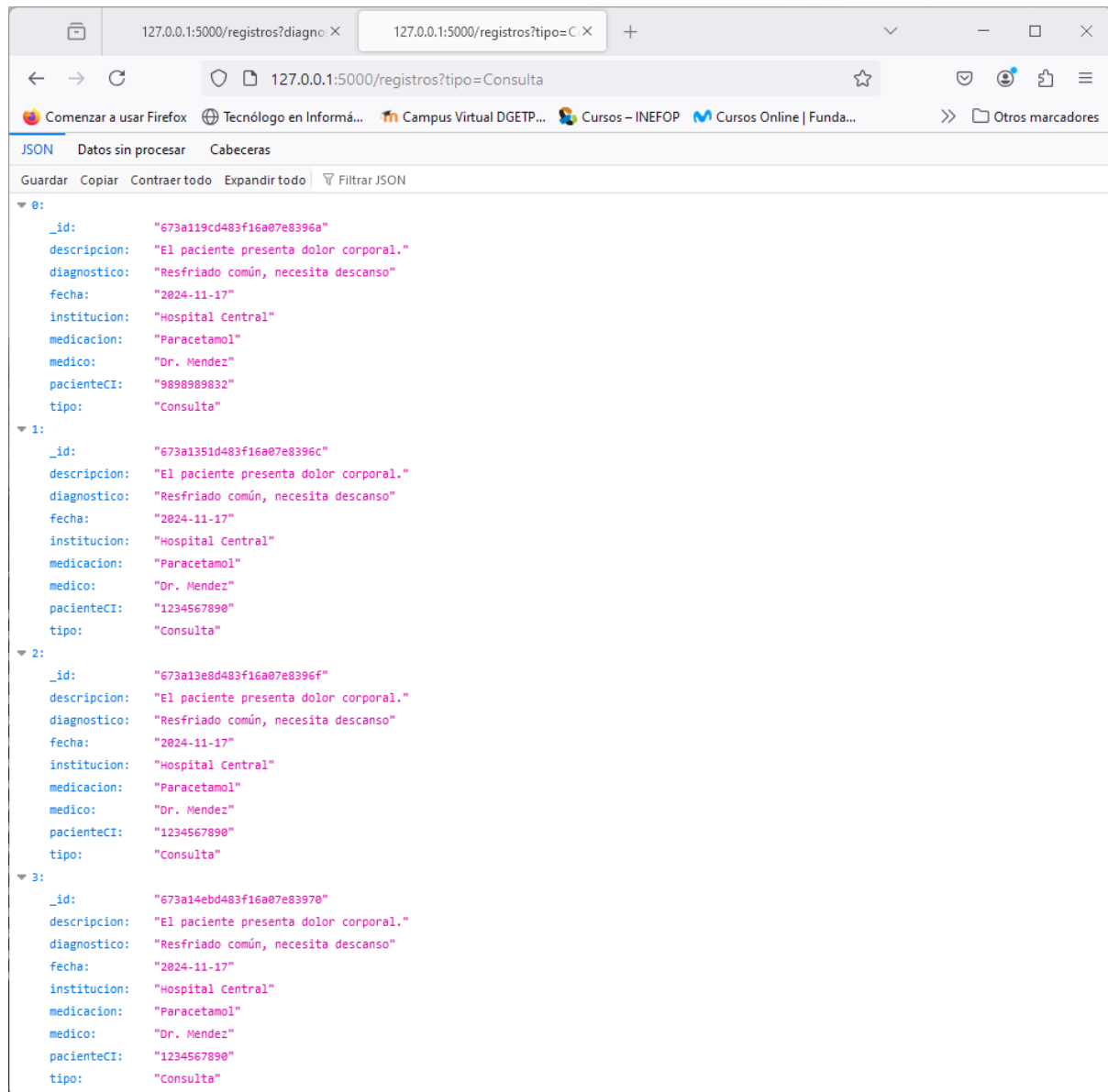


Luego de generado la imagen y ya Dockerizado, podemos verlo desde el navegador, a continuación, algunos ejemplos de ello:

Consulta GET por: DIAGNOSTICO:



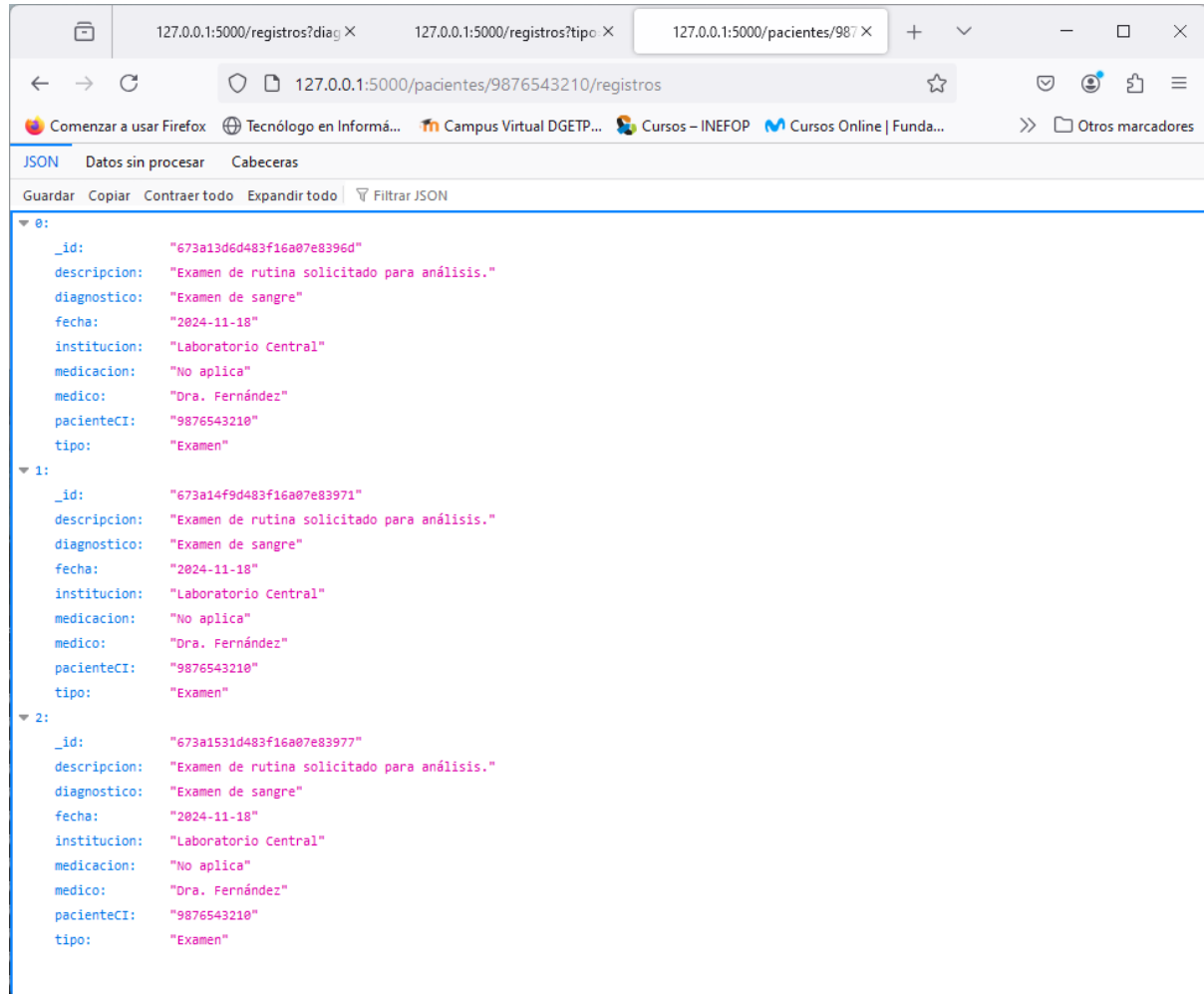
Consulta GET por: TIPO



The screenshot shows a web browser window with two tabs. The active tab is titled "127.0.0.1:5000/registros?tipo=C". The address bar shows the URL "127.0.0.1:5000/registros?tipo=Consulta". The browser's developer tools are open, displaying the JSON response of the GET request. The response is an array of 4 objects, each representing a medical record. The objects are indexed 0, 1, 2, and 3. Each object contains the following fields: _id, descripcion, diagnostico, fecha, institucion, medicacion, medico, pacienteCI, and tipo. The data is as follows:

Index	_id	descripcion	diagnostico	fecha	institucion	medicacion	medico	pacienteCI	tipo
0	"673a119cd483f16a07e8396a"	"El paciente presenta dolor corporal."	"Resfriado común, necesita descanso"	"2024-11-17"	"Hospital Central"	"Paracetamol"	"Dr. Mendez"	"9898989832"	"Consulta"
1	"673a1351d483f16a07e8396c"	"El paciente presenta dolor corporal."	"Resfriado común, necesita descanso"	"2024-11-17"	"Hospital Central"	"Paracetamol"	"Dr. Mendez"	"1234567890"	"Consulta"
2	"673a13e8d483f16a07e8396f"	"El paciente presenta dolor corporal."	"Resfriado común, necesita descanso"	"2024-11-17"	"Hospital Central"	"Paracetamol"	"Dr. Mendez"	"1234567890"	"Consulta"
3	"673a14ebd483f16a07e83970"	"El paciente presenta dolor corporal."	"Resfriado común, necesita descanso"	"2024-11-17"	"Hospital Central"	"Paracetamol"	"Dr. Mendez"	"1234567890"	"Consulta"

Consulta GET por: PACIENTE(cedula)



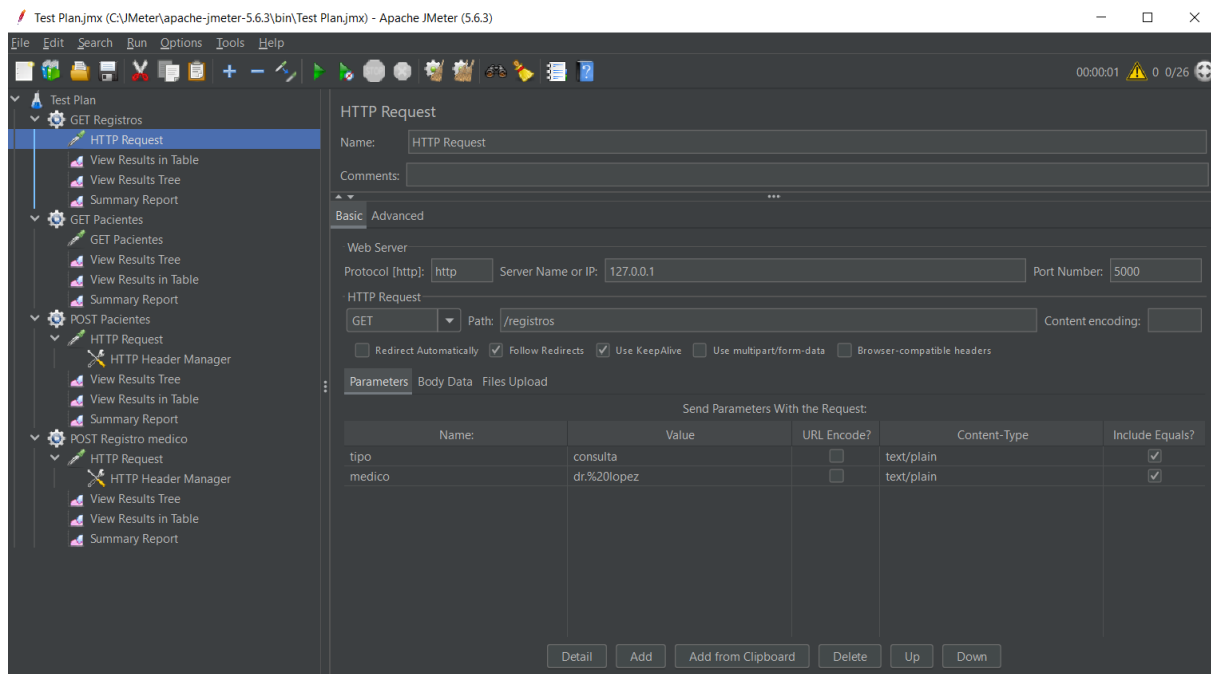
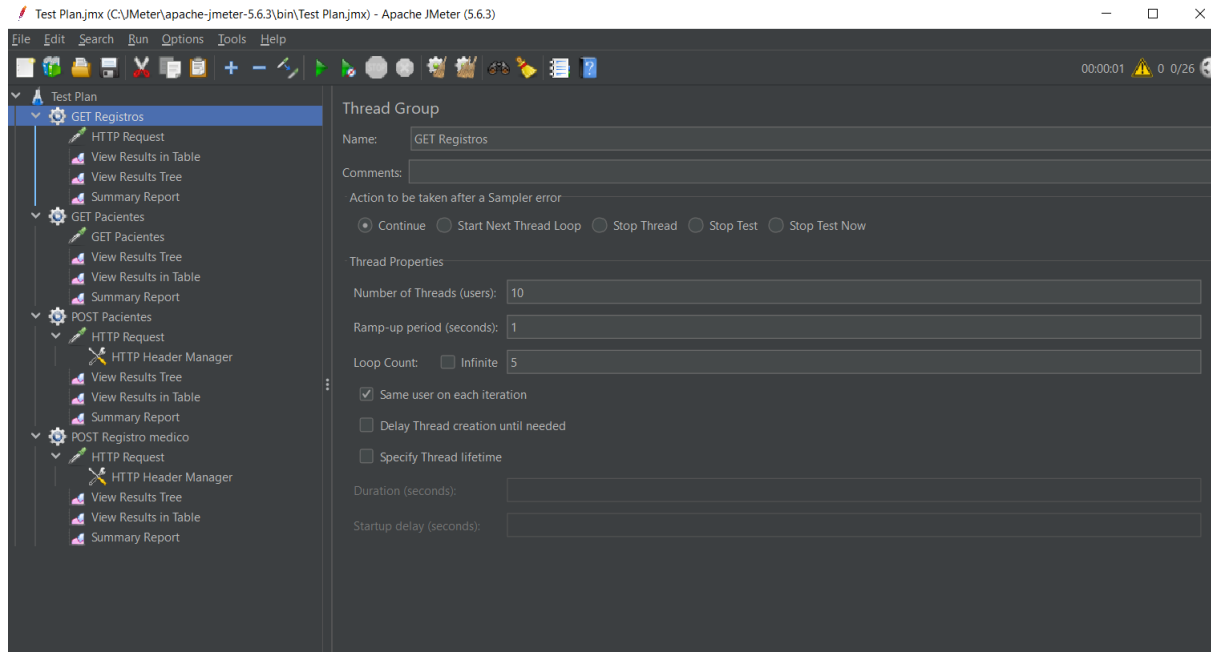
The screenshot shows a web browser window with a REST client interface. The address bar displays the URL `127.0.0.1:5000/pacientes/9876543210/registros`. The browser's developer tools are open, showing the JSON response of a GET request. The response is an array of three objects, each representing a medical exam record for a specific patient.

JSON Response:

```
{
  "0": {
    "_id": "673a13d6d483f16a07e8396d",
    "descripcion": "Examen de rutina solicitado para an\u00e1lisis.",
    "diagnostico": "Examen de sangre",
    "fecha": "2024-11-18",
    "institucion": "Laboratorio Central",
    "medicacion": "No aplica",
    "medico": "Dra. Fern\u00e1ndez",
    "pacienteCI": "9876543210",
    "tipo": "Examen"
  },
  "1": {
    "_id": "673a14f9d483f16a07e83971",
    "descripcion": "Examen de rutina solicitado para an\u00e1lisis.",
    "diagnostico": "Examen de sangre",
    "fecha": "2024-11-18",
    "institucion": "Laboratorio Central",
    "medicacion": "No aplica",
    "medico": "Dra. Fern\u00e1ndez",
    "pacienteCI": "9876543210",
    "tipo": "Examen"
  },
  "2": {
    "_id": "673a1531d483f16a07e83977",
    "descripcion": "Examen de rutina solicitado para an\u00e1lisis.",
    "diagnostico": "Examen de sangre",
    "fecha": "2024-11-18",
    "institucion": "Laboratorio Central",
    "medicacion": "No aplica",
    "medico": "Dra. Fern\u00e1ndez",
    "pacienteCI": "9876543210",
    "tipo": "Examen"
  }
}
```

JMeter:

GET:



Test Plan.jmx (C:\Meter\apache-jmeter-5.6.3\bin\Test Plan.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:01 0 0/26

Test Plan

- GET Registros
 - HTTP Request
 - View Results in Table**
 - View Results Tree
 - Summary Report
- GET Pacientes
 - GET Pacientes
 - View Results Tree
 - View Results in Table
 - Summary Report
- POST Pacientes
 - HTTP Request
 - HTTP Header Manager
 - View Results in Table
 - Summary Report
- POST Registro medico
 - HTTP Request
 - HTTP Header Manager
 - View Results Tree
 - View Results in Table
 - Summary Report

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes Configure

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Tim...
81	15:35:07.155	GET Registro...	HTTP Request	8	✓	167	158	8	2
82	15:35:07.164	GET Registro...	HTTP Request	7	✓	167	158	7	1
83	15:35:07.172	GET Registro...	HTTP Request	8	✓	167	158	8	1
84	15:35:07.181	GET Registro...	HTTP Request	9	✓	167	158	8	1
85	15:35:07.190	GET Registro...	HTTP Request	8	✓	167	158	7	2
86	15:35:07.256	GET Registro...	HTTP Request	7	✓	167	158	7	1
87	15:35:07.265	GET Registro...	HTTP Request	8	✓	167	158	7	1
88	15:35:07.273	GET Registro...	HTTP Request	9	✓	167	158	8	2
89	15:35:07.282	GET Registro...	HTTP Request	7	✓	167	158	7	2
90	15:35:07.290	GET Registro...	HTTP Request	6	✓	167	158	6	1
91	15:35:07.355	GET Registro...	HTTP Request	6	✓	167	158	6	1
92	15:35:07.362	GET Registro...	HTTP Request	8	✓	167	158	7	1
93	15:35:07.370	GET Registro...	HTTP Request	7	✓	167	158	7	1
94	15:35:07.377	GET Registro...	HTTP Request	7	✓	167	158	7	2
95	15:35:07.384	GET Registro...	HTTP Request	7	✓	167	158	7	2
96	15:35:07.454	GET Registro...	HTTP Request	7	✓	167	158	7	1
97	15:35:07.461	GET Registro...	HTTP Request	8	✓	167	158	8	2
98	15:35:07.470	GET Registro...	HTTP Request	6	✓	167	158	6	1
99	15:35:07.476	GET Registro...	HTTP Request	6	✓	167	158	6	1
100	15:35:07.492	GET Registro...	HTTP Request	8	✓	167	158	7	1

Test Plan.jmx (C:\Meter\apache-jmeter-5.6.3\bin\Test Plan.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:01 0 0/26

Test Plan

- GET Registros
 - HTTP Request
 - View Results in Table
 - View Results Tree**
 - Summary Report
- GET Pacientes
 - GET Pacientes
 - View Results Tree
 - View Results in Table
 - Summary Report
- POST Pacientes
 - HTTP Request
 - HTTP Header Manager
 - View Results Tree
 - View Results in Table
 - Summary Report
- POST Registro medico
 - HTTP Request
 - HTTP Header Manager
 - View Results Tree
 - View Results in Table
 - Summary Report

View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes Configure

Search: Case sensitive Regular exp. Search Reset

Text

Sampler result Request Response data

Thread Name:UsuariosNoSQL 1-8
Sample Start:2024-11-17 15:13:24 UYT
Load time:5
Connect Time:1
Latency:5
Size in bytes:167
Sent bytes:158
Headers size in bytes:164
Body size in bytes:3
Sample Count:1
Error Count:0
Data type ("text"|"bin"|"") :text
Response code:200
Response message:OK

HTTPSampleResult fields:
ContentType: application/json
DataEncoding: null

Test Plan.jmx (C:\Meter\apache-jmeter-5.6.3\bin\Test Plan.jmx) - Apache JMeter (5.6.3)

FileEditSearchRunOptionsToolsHelp

Test Plan

- GET Registros
 - HTTP Request
 - View Results in Table
 - View Results Tree
 - Summary Report
- GET Pacientes
 - View Results Tree
 - View Results in Table
 - Summary Report
- POST Pacientes
 - HTTP Request
 - HTTP Header Manager
 - View Results Tree
 - View Results in Table
 - Summary Report
- POST Registro medico
 - HTTP Request
 - HTTP Header Manager
 - View Results Tree
 - View Results in Table
 - Summary Report

00:00:01 0 0/26

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename

Browse...

Log/Display Only

Errors

Successes

Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	550	9	4	94	7.41	0.00%	12.8/min	0.03	0.03	167.0
TOTAL	550	9	4	94	7.41	0.00%	12.8/min	0.03	0.03	167.0

☐ Include group name in label?

☐ Save Table Data

☒ Save Table Header

POST:

The screenshot displays the Apache JMeter 5.6.3 interface. The top window shows a Thread Group configuration with the name "POST Pacientes". The bottom window shows the HTTP Header Manager configuration for the selected "HTTP Request" under the "POST Pacientes" thread group. The headers are defined as follows:

Name	Value
Content-Type	application/json

The bottom window also shows the "Basic" tab of the HTTP Request configuration, with the following details:

- Web Server: Protocol (http), Server Name or IP: 127.0.0.1, Port Number: 5000
- HTTP Request: Method (POST), Path: /pacientes, Content encoding: (empty)
- Parameters: Body Data, Files Upload
- Body Data:

```
1 {  
2   "id": "1234567890123",  
3   "nombre": "Narcis",  
4   "apellidos": "Garcia",  
5   "fecha_nacimiento": "1988-01-03",  
6   "sexo": "M",  
7 }
```

Test Plan.jmx (C:\Meter\apache-jmeter-5.6.3\bin\Test Plan.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:01 0 0/25

Test Plan

- GET Registros
 - HTTP Request
 - View Results in Table
 - View Results Tree
 - Summary Report
- GET Pacientes
 - GET Pacientes
 - View Results Tree
 - View Results in Table
 - Summary Report
- POST Pacientes
 - HTTP Request
 - HTTP Header Manager
 - View Results Tree
 - View Results in Table
 - Summary Report
- POST Registro medico
 - HTTP Request
 - HTTP Header Manager
 - View Results Tree
 - View Results in Table
 - Summary Report

View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes

Search: ☐ Case sensitive ☐ Regular exp.

Text

HTTP Request

Thread Name: POST Pacientes 3-1
Sample Start: 2024-11-17 16:26:31 UYT
Load time: 31
Connect Time: 4
Latency: 31
Size in bytes: 268
Sent bytes: 312
Headers size in bytes: 170
Body size in bytes: 98
Sample Count: 1
Error Count: 0
Data type: ["text"] ["bin"] .text
Response code: 201
Response message: CREATED

HTTPSampleResult fields:
ContentType: application/json
DataEncoding: null

☐ Scroll automatically?

Raw Parsed

Test Plan.jmx (C:\Meter\apache-jmeter-5.6.3\bin\Test Plan.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:01 0 0/26

Test Plan

- GET Registros
 - HTTP Request
 - View Results in Table
 - View Results Tree
 - Summary Report
- GET Pacientes
 - GET Pacientes
 - View Results Tree
 - View Results in Table
 - Summary Report
- POST Pacientes
 - HTTP Request
 - HTTP Header Manager
 - View Results Tree
 - View Results in Table
 - Summary Report
- POST Registro medico
 - HTTP Request
 - HTTP Header Manager
 - View Results Tree
 - View Results in Table
 - Summary Report

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes ☐ Configure

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	162631.932	POST Pacientes 3-1	HTTP Request	31		268	312		31

☐ Scroll automatically? ☐ Child samples? No of Samples: 1 Latest Sample: 31 Average: 31 Deviation: 0

Test Plan.jmx (C:\Meter\apache-jmeter-5.6.3\bin\Test Plan.jmx) - Apache JMeter (5.6.3)

File Edit Search Run Options Tools Help

00:00:01 0 0/26

Test Plan

- GET Registros
 - HTTP Request
 - View Results in Table
 - View Results Tree
 - Summary Report
- GET Pacientes
 - GET Pacientes
 - View Results Tree
 - View Results in Table
 - Summary Report
- POST Pacientes
 - HTTP Request
 - HTTP Header Manager
 - View Results Tree
 - View Results in Table
 - Summary Report
- POST Registro medico
 - HTTP Request
 - HTTP Header Manager
 - View Results Tree
 - View Results in Table
 - Summary Report

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes ☐ Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	1	31	31	31	0.00	0.00%	32.3/sec	8.44	9.83	268.0
TOTAL	1	31	31	31	0.00	0.00%	32.3/sec	8.44	9.83	268.0

☐ Include group name in label? ☐ Save Table Data ☒ Save Table Header

Conclusión

Este proyecto implementa una solución completa para la gestión de historial médico de pacientes utilizando una API REST en Python con Flask y MongoDB como base de datos NoSQL. La aplicación cumple con los requisitos funcionales y no funcionales del proyecto, y es fácilmente escalable y extensible para futuros requerimientos.