Proyecto 2 - Plataforma de busquedas de películas

Bases de Datos 2

Integrantes:

- Fabián Bustos: 2020048344
- Ian Murillo: 2020148871
- Luany Masis: 2022155917
- David Blanco: 2020053806
- Harlen Quirós: 2022035693
- Mario Lara: 2020035341

La plataforma busca funcionar como una especie de buscador, donde el usuario tiene la capacidad de buscar películas por su nombre, nombres de actores y directores, reparto, y otra información relacionada. El usuario puede interactuar con la aplicación mediante un prototipo de Thunkable, el cual fue dirigido a uso en aparatos móviles.

Instalación

Los elementos del proyecto que necesitan instalación son Docker y Ngrok. Se asumirá que Docker ya se encuentra instalado.

Instalación Ngrok

1. Instalar ngrok via Chocolatey

   choco install ngrok

2. Agregar Token

   ngrok config add-authtoken

3. Correr en el puerto en el que se está ejecutando el proyecto

   ngrok http http://localhost:5000

Para llevar a cabo la instalación de los helm charts necesarios para la aplicación se requiere correr los siguientes comandos:

```
helm install application application
helm install monitoring-stack monitoring-stack
```

Neo4j

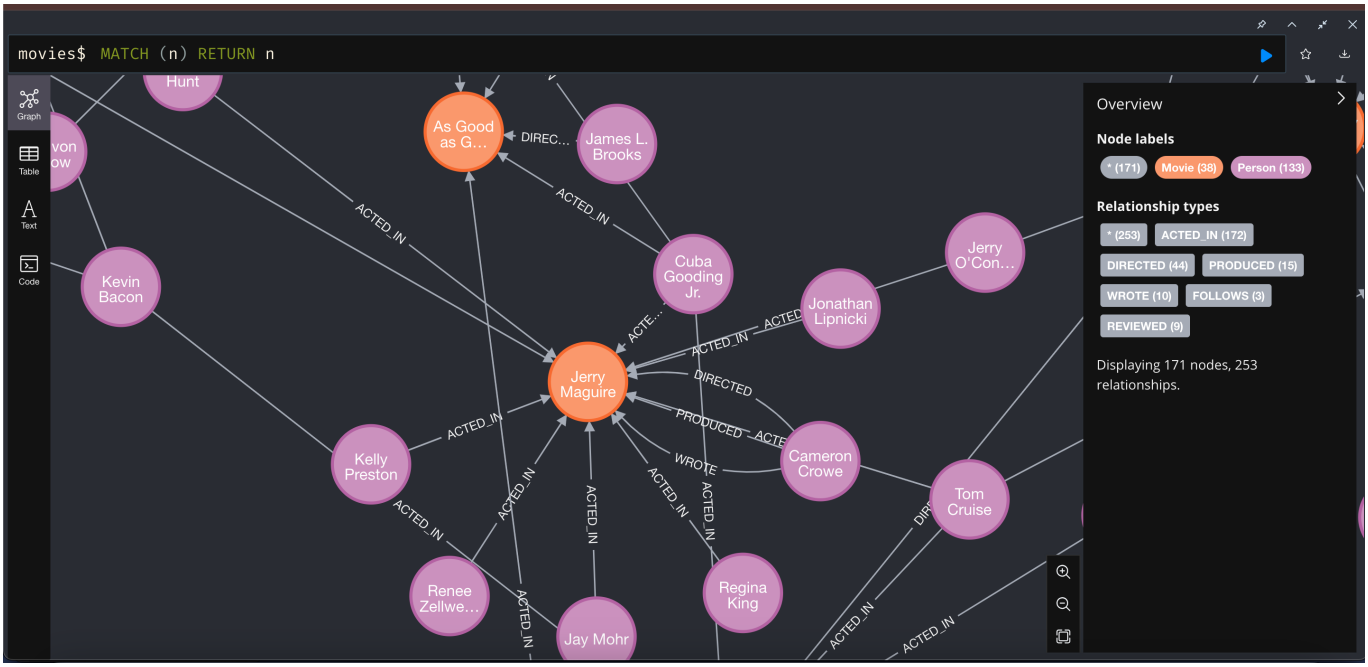
Una de las fuentes de la información de películas se obtiene de Neo4J. Neo4j es una sistema admninistrador de bases de datos de grafos. Esta almacena nodos, relaciones, y sus respectivos atributos. Esta propiedad la

hace sumamente eficiente en representar relaciones y patrones entre los datos que almacena.

Para esta aplicación en específico se utiliza la base de datos de ejemplo sobre películas. Contiene nodos que representan películas y personas, los cuales tienen relaciones entre si tales como "actuó en" o "dirigió", las cuales son esenciales para el funcionamiento de la aplicación.

Mediante una representación visual del modelo de grafos que se encuentra en la base de películas, se puede observar como este es sumamente útil para evaluar las relaciones colaborativas que se llevan a cabo en el ambiente de las películas.

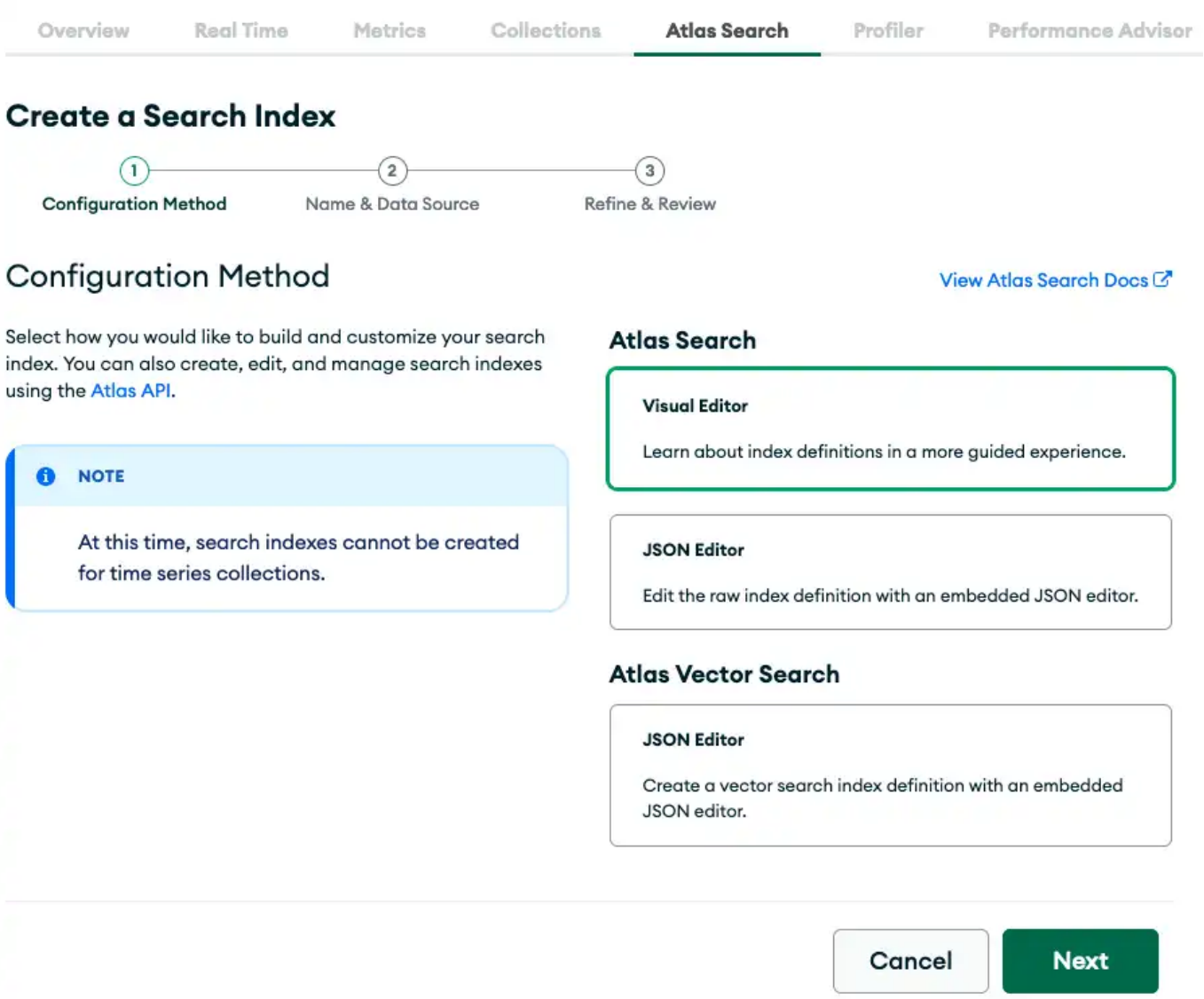
Observando con más detalle, se pueden observar los diferentes tipos de relaciones que hay entre cada nodo así como los atributos que tiene cada uno.



La información que se obtiene de esta base de datos conforma una parte del funcionamiento de la aplicación. Toda esta información se accede mediante un API construida en Flask, la cual será explicada más adelante.

Mongo Atlas

Con respecto a Mongo Atlas, se debe crear la base de datos. Para crear la base de datos, se puede realizar mediante internet. Una vez se tenga la base de datos creada, se procede a crear los índices. Se debe ir al apartado de Atlas Search, y se ingresa al botón 'Create Index' donde se mostrará la siguiente pantalla:



Se utilizará el JSON Editor, donde se configurará el mapping dependiendo del propósito del índice. Una vez realizados los índices, estos están listos para útilizar. Para efectos de este proyecto, se realizaron los índices de Cast, Directors y Plot.

Flask API

La aplicación utiliza un API de Python implementada utilizando Flask para comunicarse con las bases de datos necesarias. Ambas implementaciones utilizan librerías de Python para llevar a cabo las conexiones seguras con las bases de datos. Para neo4j se utiliza la librería del mismo nombre, para Mongo Atlas se utiliza la librería denominada "pymongo". Todo esto se encuentra en un contenedor y se puede ejecutar mediante Kubernetes. Este API contiene diversos endpoints con diferentes criterios de busqueda que son utilizados para Este será accedido mediante la herramienta llamada Ngrok.

Grafana

Grafana es una plataforma de análisis y monitoreo que se utilizará junto con Prometheus para supervisar el API y MongoDB Atlas.

Instalación y Configuración:

1. Instalar Grafana en Kubernetes: Instalar Grafana en Kubernetes.
2. Configurar Prometheus: Se utiliza y configura Prometheus para recolectar métricas del API y MongoDB Atlas.
3. Importar Dashboards: Utilizar dashboards para visualizar las métricas de el API, como el número de peticiones y el tiempo de respuesta.

Ngrok

Ngrok es una herramienta que permite realizar un port-forwarding de la computadora local y publicarlo en Internet. Ngrok juega un papel vital en este proyecto al facilitar la comunicación entre Thunkable y el API.

Uso: Ngrok funcionará como intermediario entre la aplicación móvil (Thunkable) y el API que se ejecuta localmente. Esto sirve para que la app pueda acceder al API local sin tener que desplegar el API en una nube.Ngrok proporciona una URL pública segura que redirige las solicitudes a la API local.

Instalación:

1. Instalar ngrok via Chocolatey

   choco install ngrok

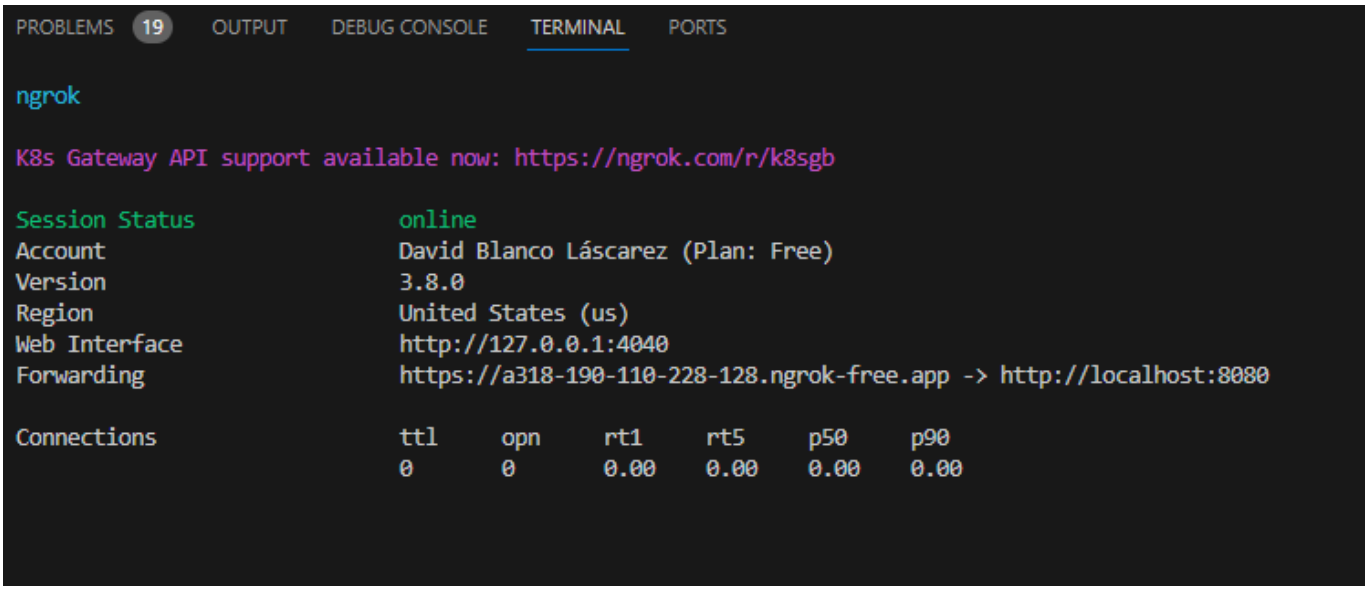
2. Agregar Token

   ngrok config add-authtoken

3. Correr en puerto

   ngrok http http://localhost:8080

Se obtiene lo siguiente:



Thunkable

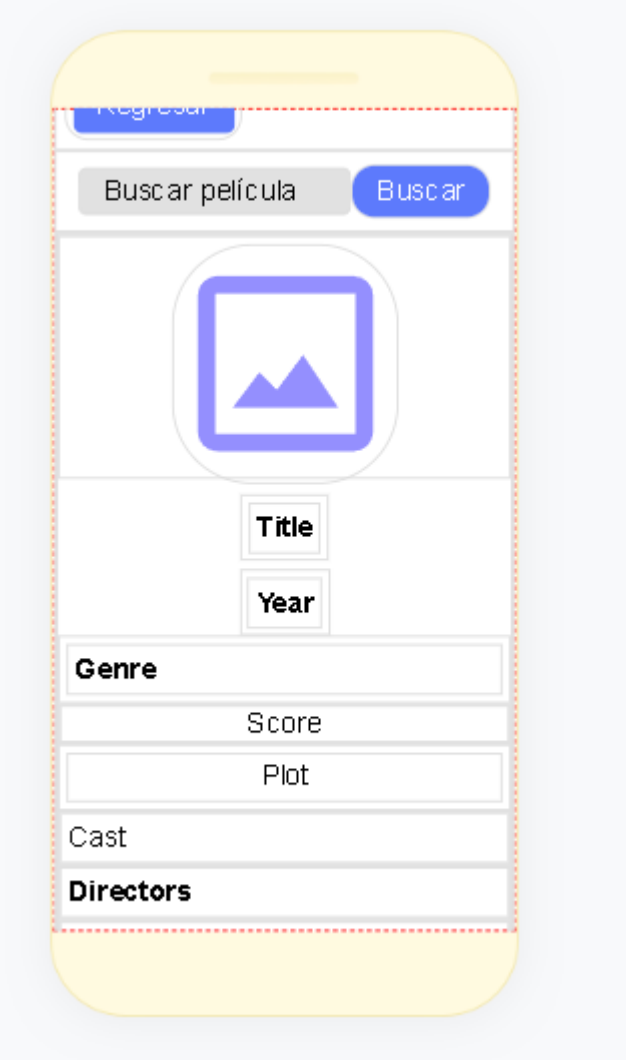
Thunkable es una plataforma de desarrollo de aplicaciones móviles que permite crear aplicaciones de manera visual sin necesidad de escribir código.

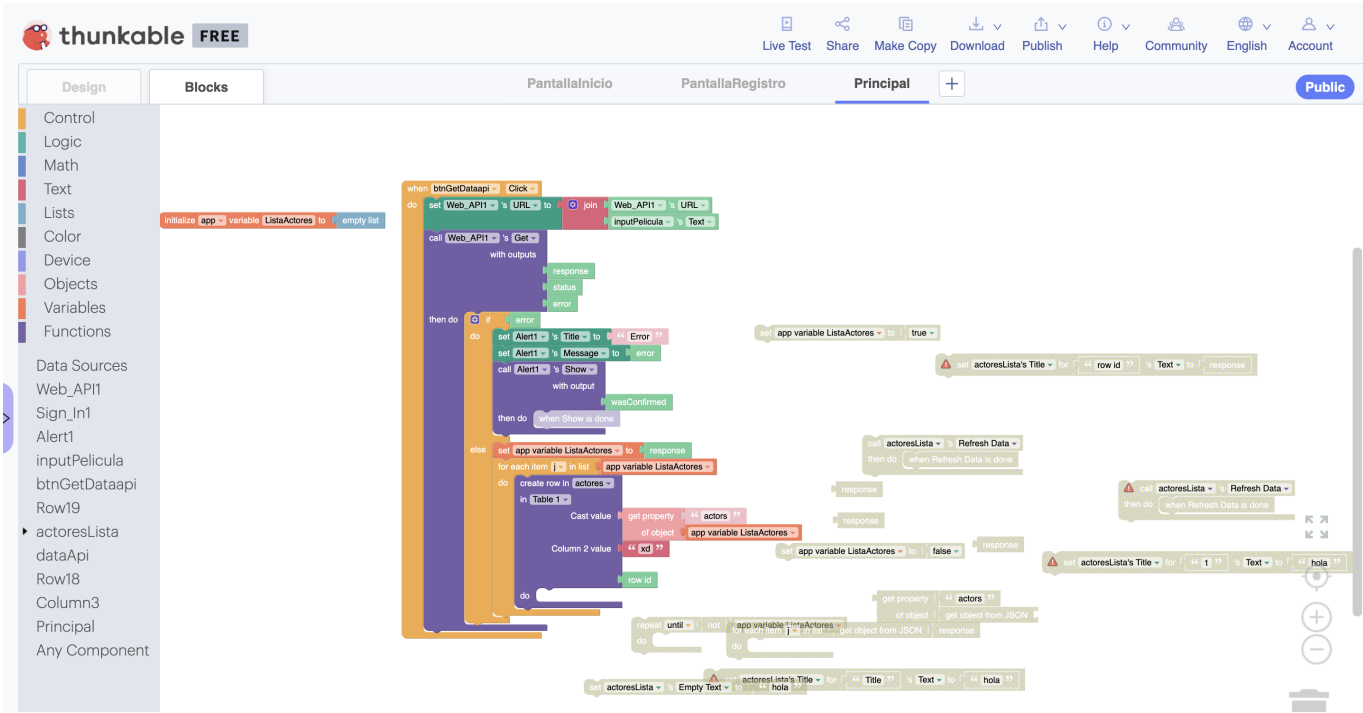
Link al proyecto:

https://x.thunkable.com/copy/2127285eed2109ec7f5ec3adcf2da0f6

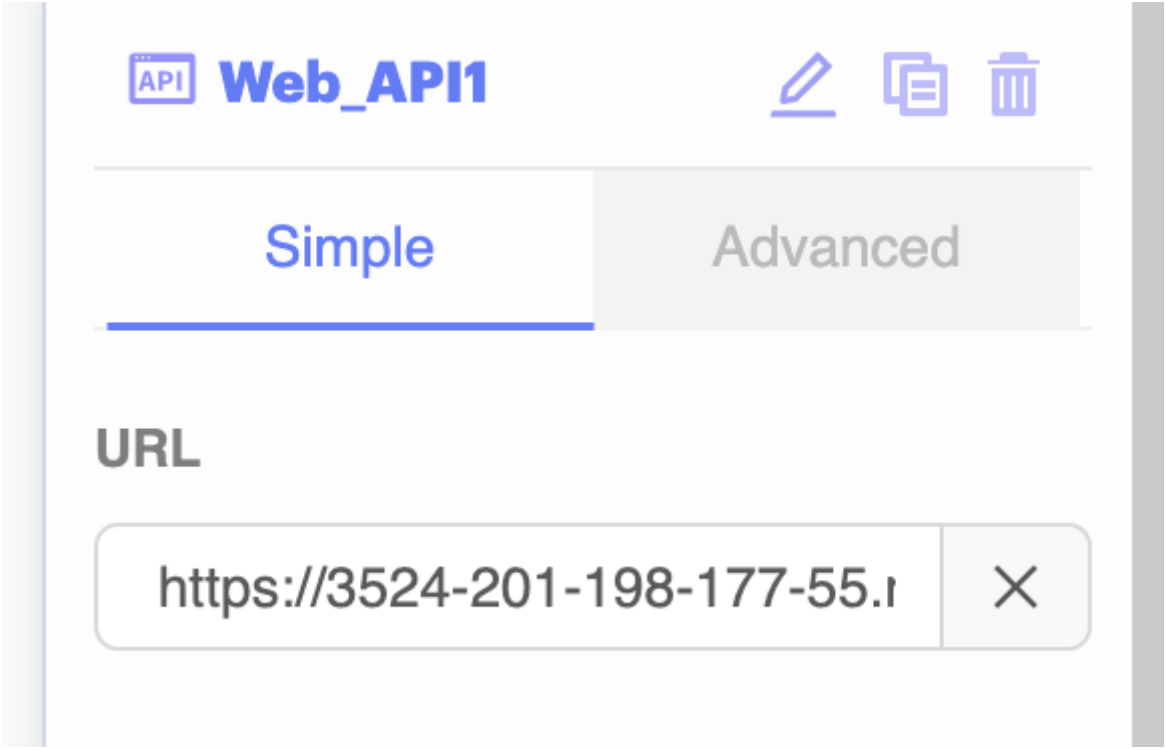
Implementación:

1. Crear Proyecto en Thunkable: Se crea un nuevo proyecto en Thunkable y se configuran las pantallas y componentes necesarios para la aplicación, se debe elaborar la parte de diseño de pantalla y la codificación se realiza con bloques.





2. Configurar Conexión al API: Usa las URLs proporcionadas por Ngrok para conectar la aplicación móvil al API.



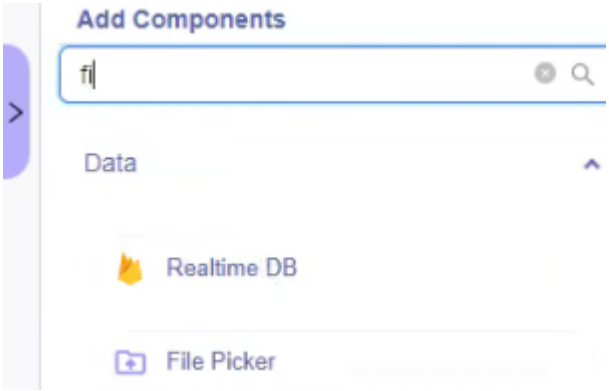
3. Autenticación y Autorización: Implementar la autenticación y autorización de usuarios utilizando Firebase dentro de Thunkable.

Firebase

La base de datos de Firebase se utiliza en nuestro programa con el fin de autenticar usuarios. En caso de que el usuario que intente ingresar a la aplicacion no sea valido, devuelve un error.

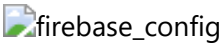
Para crear la base de datos de Firebase, simplemente en la pagina principal del perfil: Agregar app -> Web -> Agregar nombre -> Registrar.

Firebase es llamado desde Thunkable utilizando "Realtime DB".



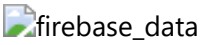
En el proyecto de Thunkable se agregan los "Firebase Settings" los cuales son necesarios para el funcionamiento de la base de datos.

El API key y el "Database URL" son encontrados dentro del perfil de Firebase, especificamente en: nombre_de_proyecto/Configuración de proyecto.



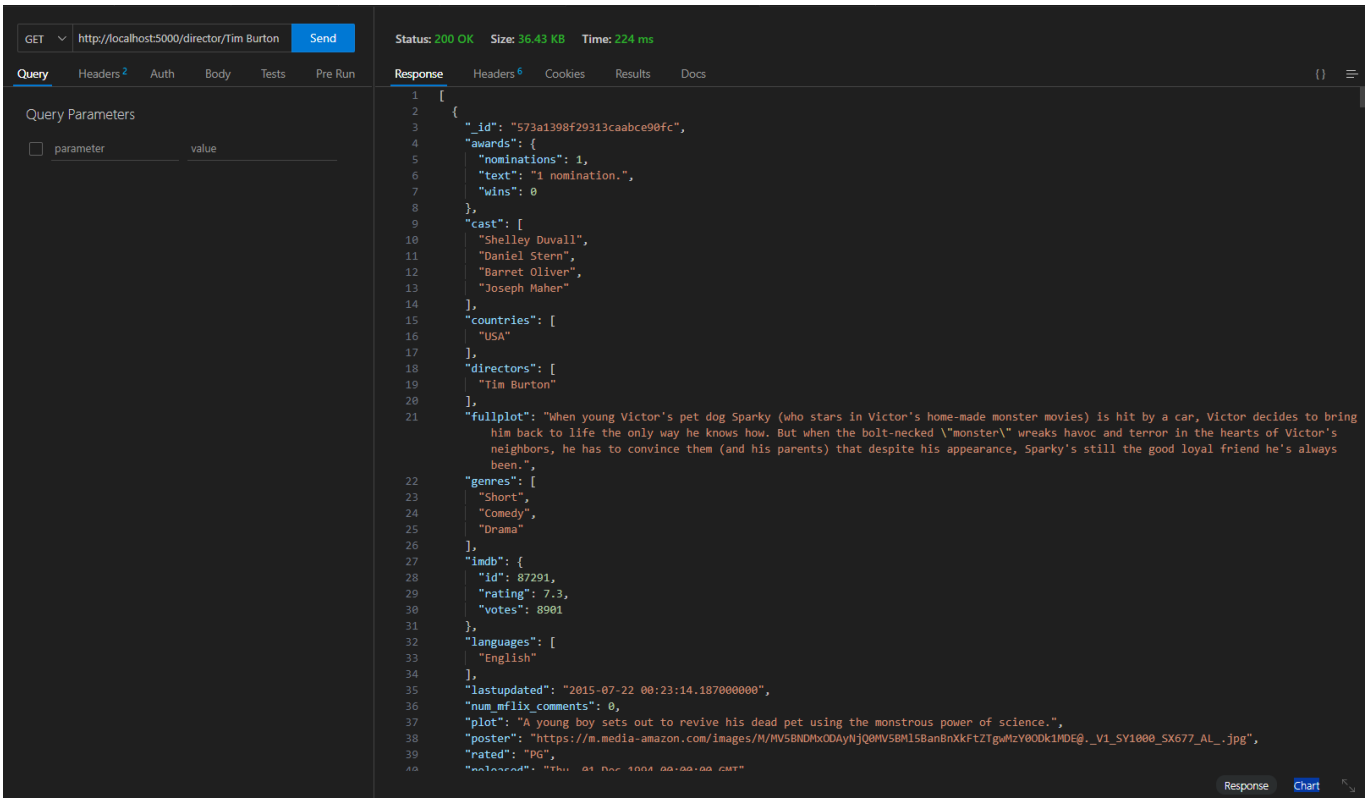
En la pestaña de autenticacion dentro de Firebase, se pueden observar los usuarios y métodos de acceso (en nuestro caso se utiliza correo electrónico y contraseña)

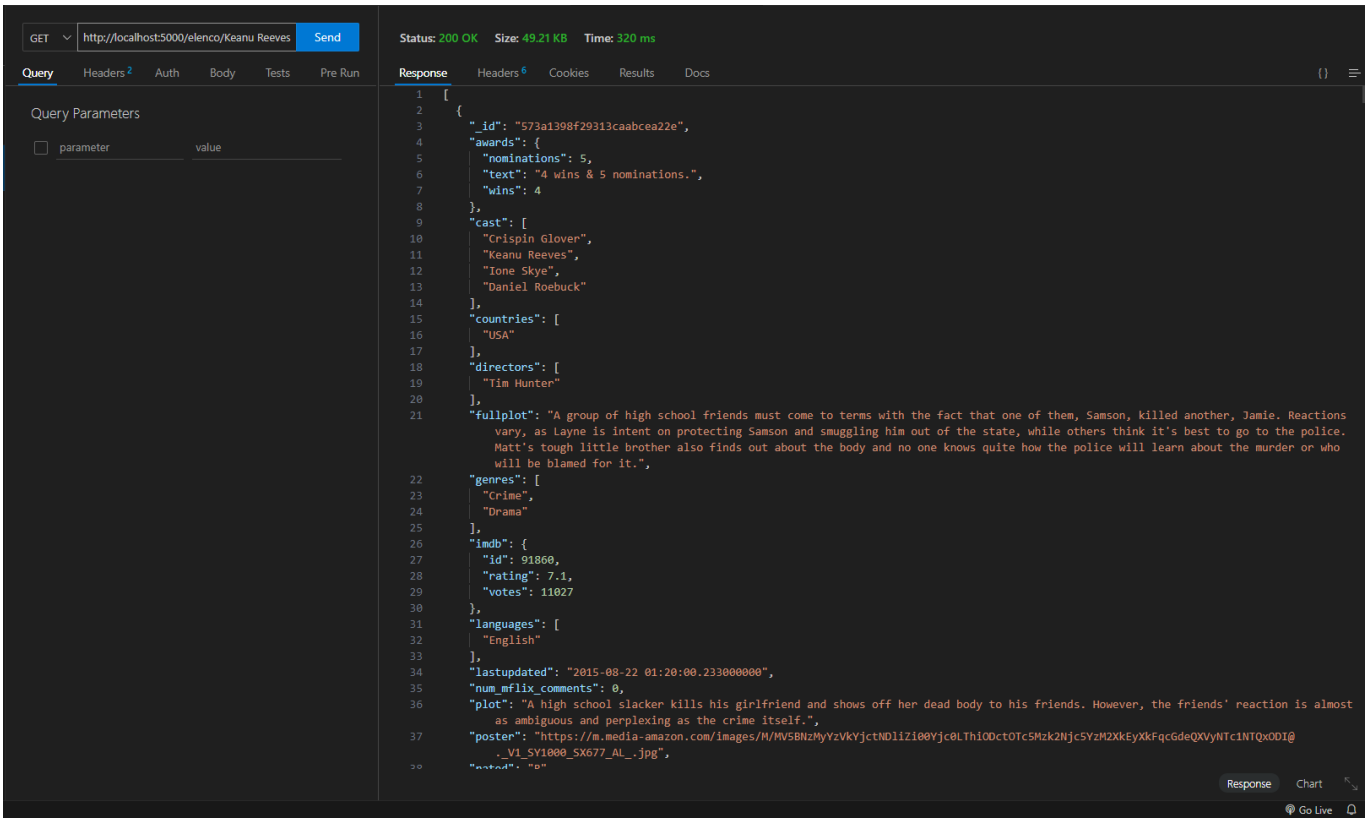
En la pestaña de "Realtime Database" se pueden observar los datos que hay dentro de la base de datos. Por ejemplo se muestran los datos de un correo de prueba:

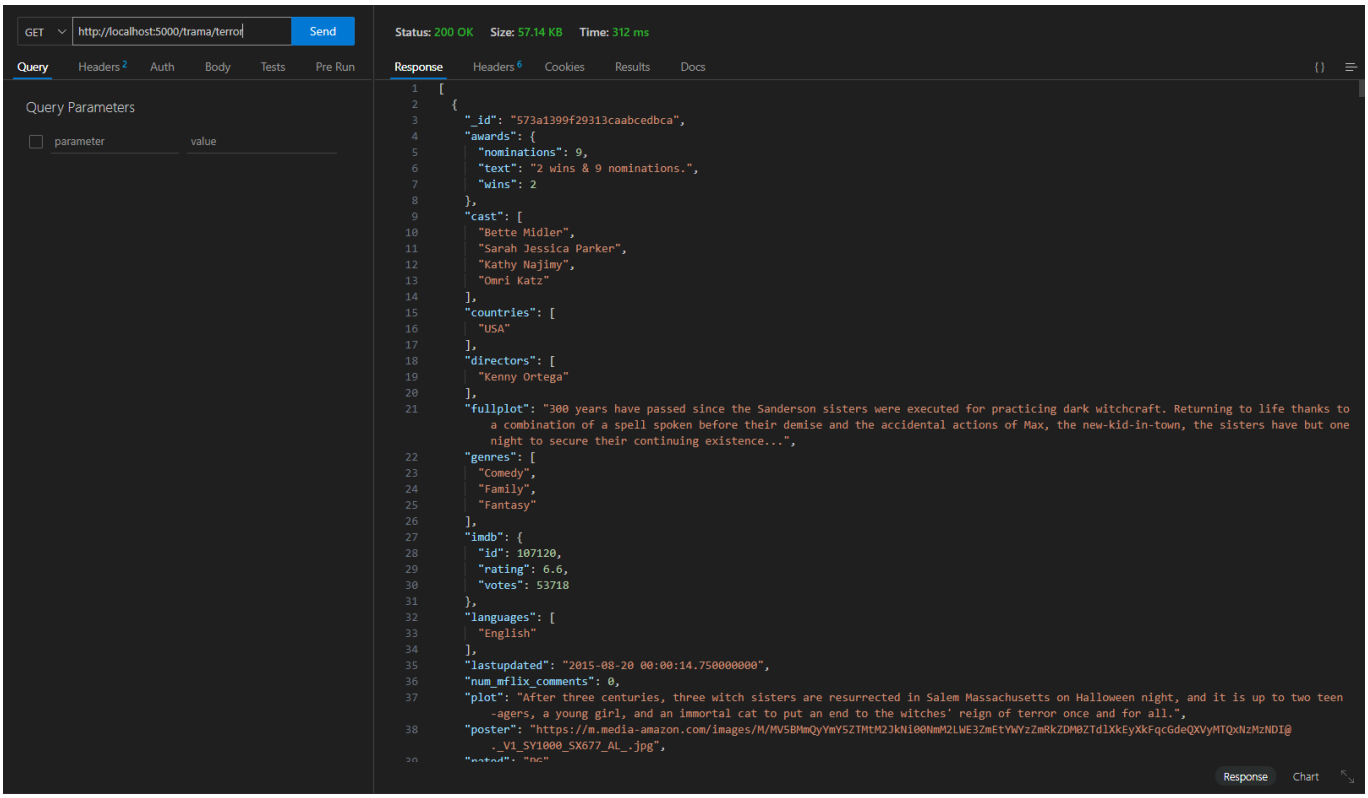


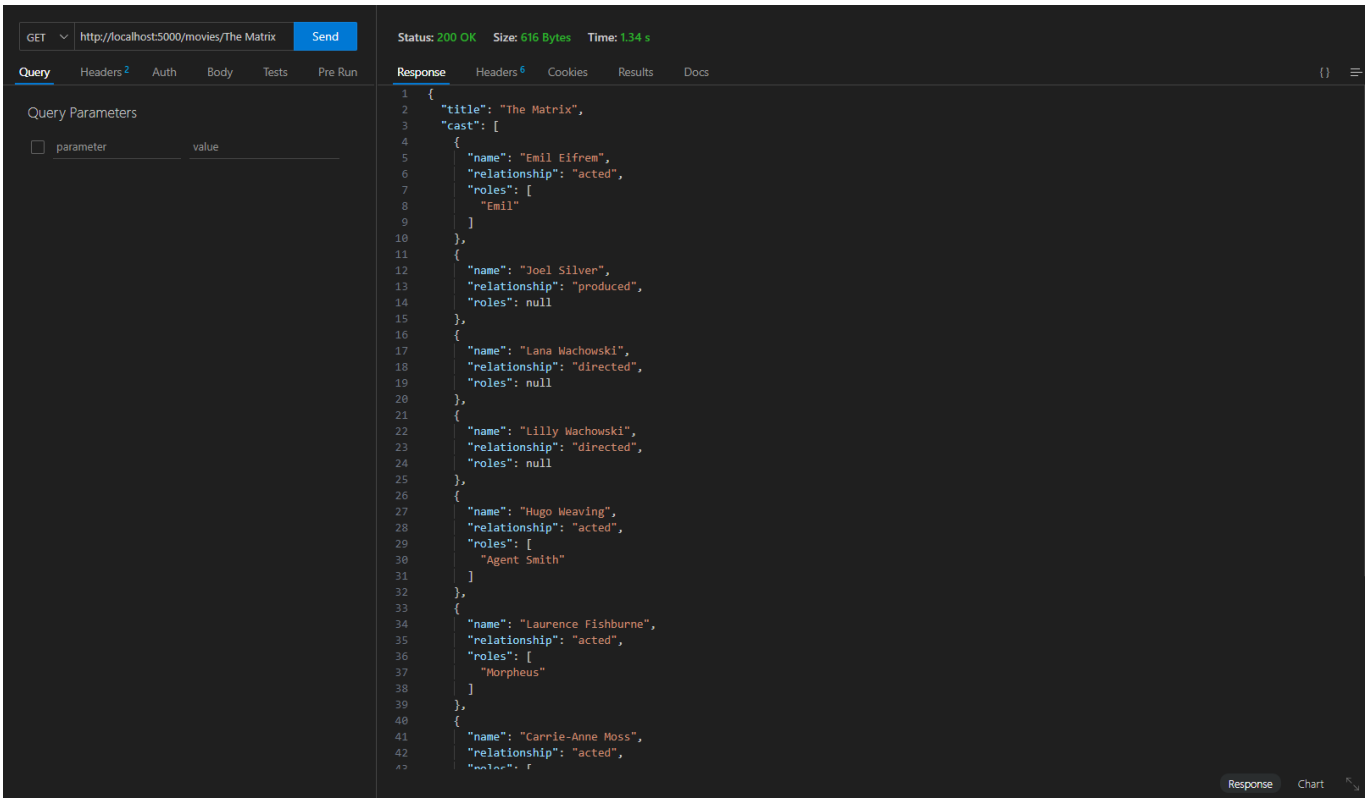
Pruebas Unitarias

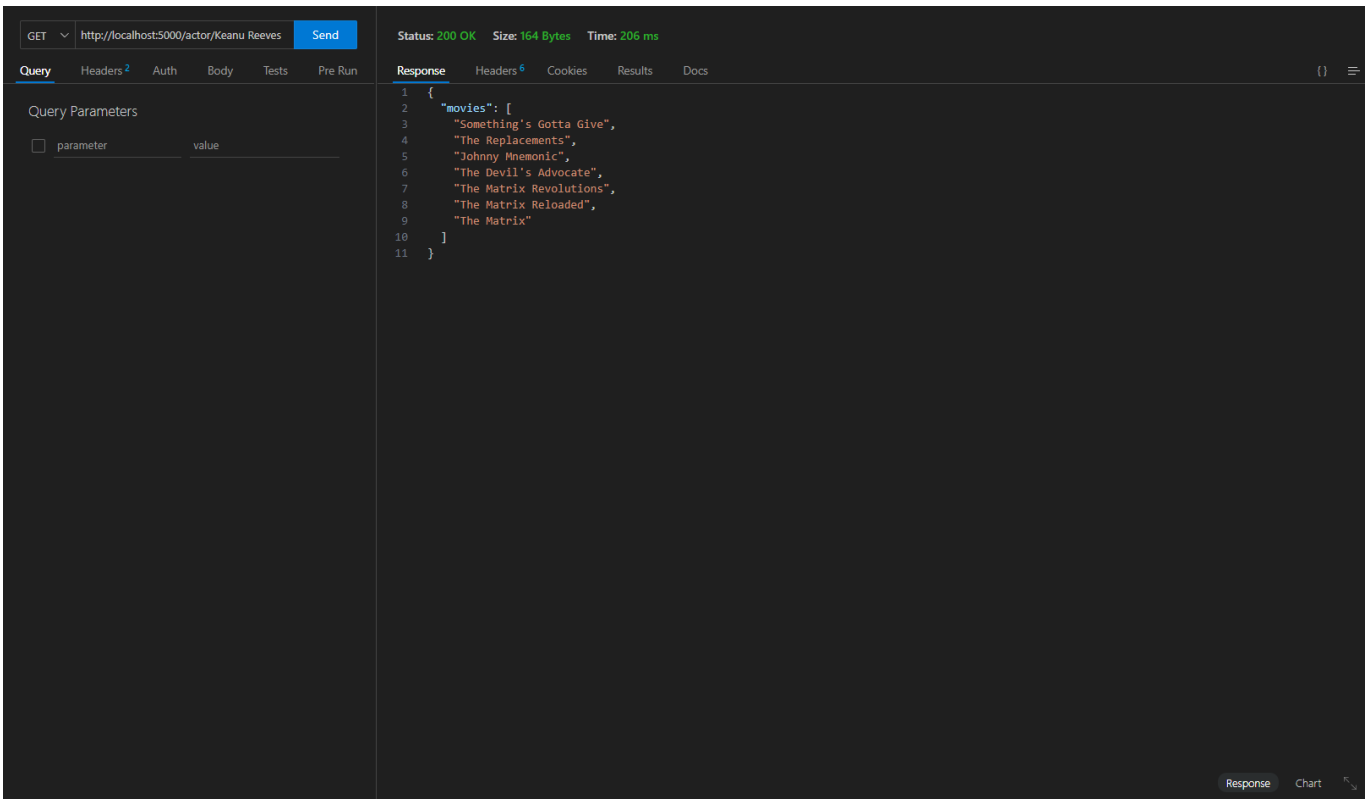
Para las pruebas unitarias se va a utilizar una extensión de visual studio code llamda Thunder Client, la misma hace request a un URL deseado, en este se puede editar tanto el body, como el header del request. Pruebas a cada endpoint de la api:

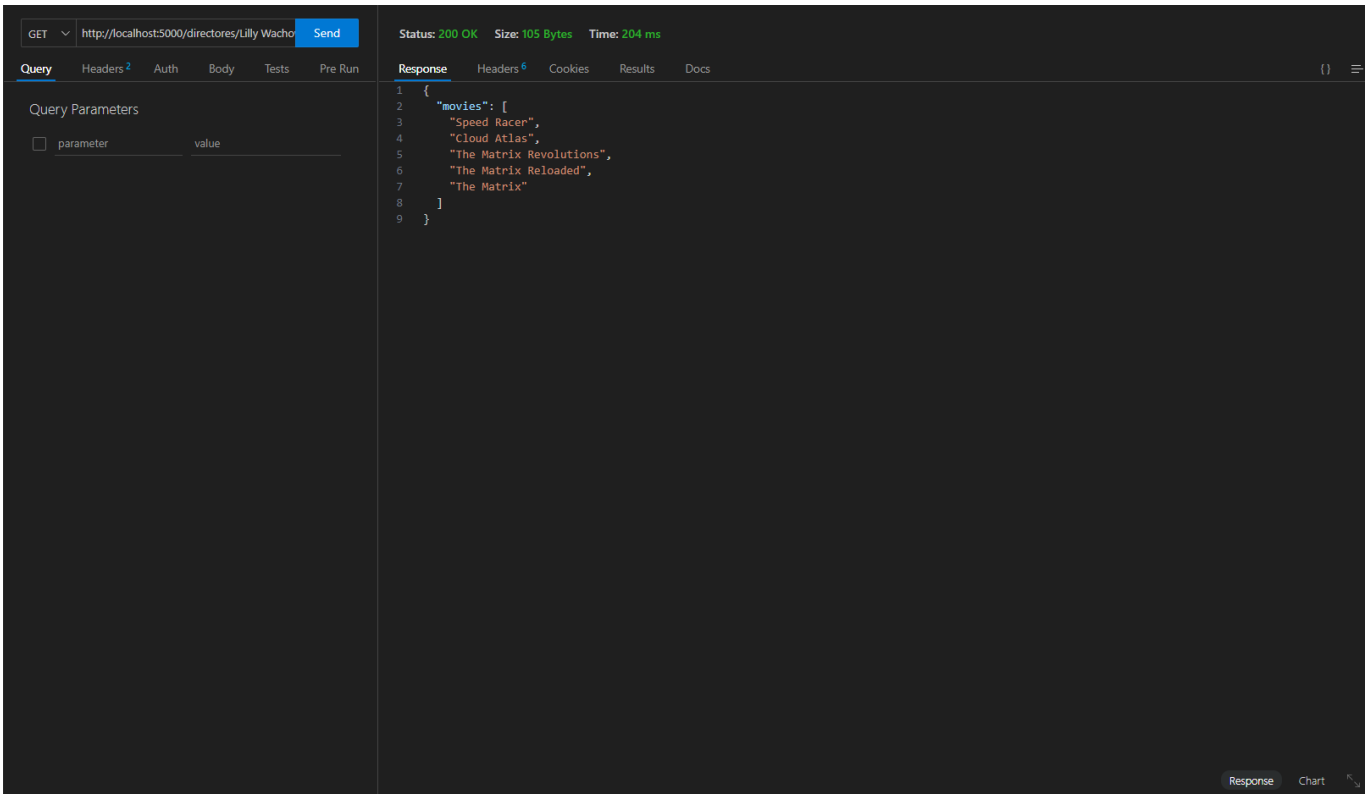


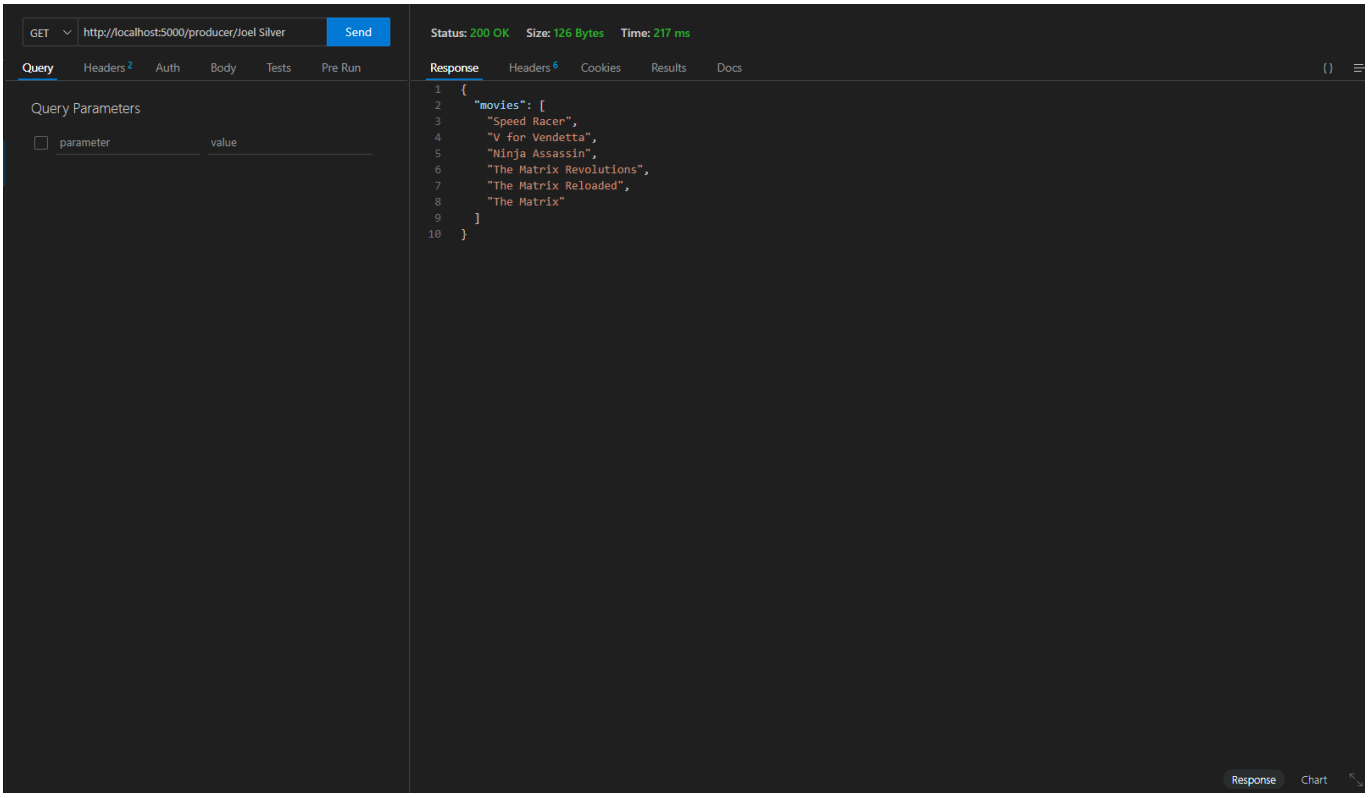

GET
http://localhost:5000/elenco/Keanu Reeves
Send
Query
Headers 2
Auth
Body
Tests
Pre Run
Query Parameters
parameter
value
Status: 200 OK
Size: 49.21 KB
Time: 320 ms
Response
Headers 6
Cookies
Results
Docs
{
"_id": "573a1398f29313caabcea22e",
"awards": {
"nominations": 5,
"text": "4 wins & 5 nominations.",
"wins": 4
},
"cast": [
"Crispin Glover",
"Keanu Reeves",
"Ione Skye",
"Daniel Roebuck"
],
"countries": [
"USA"
],
"directors": [
"Tim Hunter"
],
"fullplot": "A group of high school friends must come to terms with the fact that one of them, Samson, killed another, Jamie. Reactions vary, as Layne is intent on protecting Samson and smuggling him out of the state, while others think it's best to go to the police. Matt's tough little brother also finds out about the body and no one knows quite how the police will learn about the murder or who will be blamed for it.",
"genres": [
"Crime",
"Drama"
],
"imdb": {
"id": 91860,
"rating": 7.1,
"votes": 11027
},
"languages": [
"English"
],
"lastupdated": "2015-08-22 01:20:00.233000000",
"num_mflix_comments": 0,
"plot": "A high school slacker kills his girlfriend and shows off her dead body to his friends. However, the friends' reaction is almost as ambiguous and perplexing as the crime itself.",
"poster": "https://m.media-amazon.com/images/M/MV5BNzMyYzVkYjctNDliZi00Yjc0LThiODctOTc5Mzk2Njc5YzM2XkEyXkFqcGdeQXVyNTc1NTQxODI@._V1_SY1000_SX677_AL_.jpg",
Response
Chart
Go Live

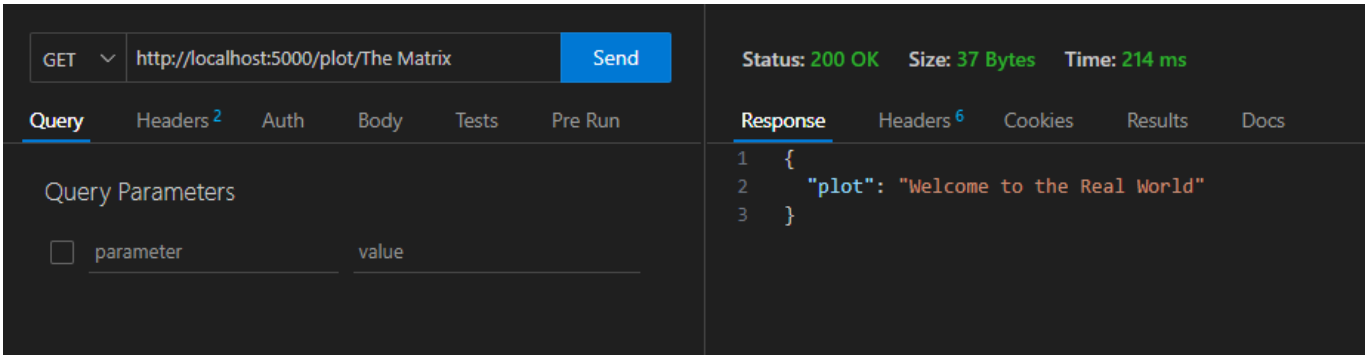

GET
http://localhost:5000/trama/terror
Send
Query
Headers 2
Auth
Body
Tests
Pre Run
Query Parameters
parameter
value
Status: 200 OK
Size: 57.14 KB
Time: 312 ms
Response
Headers 6
Cookies
Results
Docs
{
"_id": "573a1399f29313caabcedbca",
"awards": {
"nominations": 9,
"text": "2 wins & 9 nominations.",
"wins": 2
},
"cast": [
"Bette Midler",
"Sarah Jessica Parker",
"Kathy Najimy",
"Omri Katz"
],
"countries": [
"USA"
],
"directors": [
"Kenny Ortega"
],
"fullplot": "300 years have passed since the Sanderson sisters were executed for practicing dark witchcraft. Returning to life thanks to a combination of a spell spoken before their demise and the accidental actions of Max, the new-kid-in-town, the sisters have but one night to secure their continuing existence...",
"genres": [
"Comedy",
"Family",
"Fantasy"
],
"imdb": {
"id": 107120,
"rating": 6.6,
"votes": 53718
},
"languages": [
"English"
],
"lastupdated": "2015-08-20 00:00:14.750000000",
"num_mflix_comments": 0,
"plot": "After three centuries, three witch sisters are resurrected in Salem Massachusetts on Halloween night, and it is up to two teen-agers, a young girl, and an immortal cat to put an end to the witches' reign of terror once and for all.",
"poster": "https://m.media-amazon.com/images/M/MV5BMmQyYmY5ZTMtM2JkNi00NmM2LWE3ZmEtYWYzZmRkZDM0ZTdlXkEyXkFqcGdeQXVyMTQxNzMzNDI@._V1_SY1000_SX677_AL_.jpg",
Response
Chart

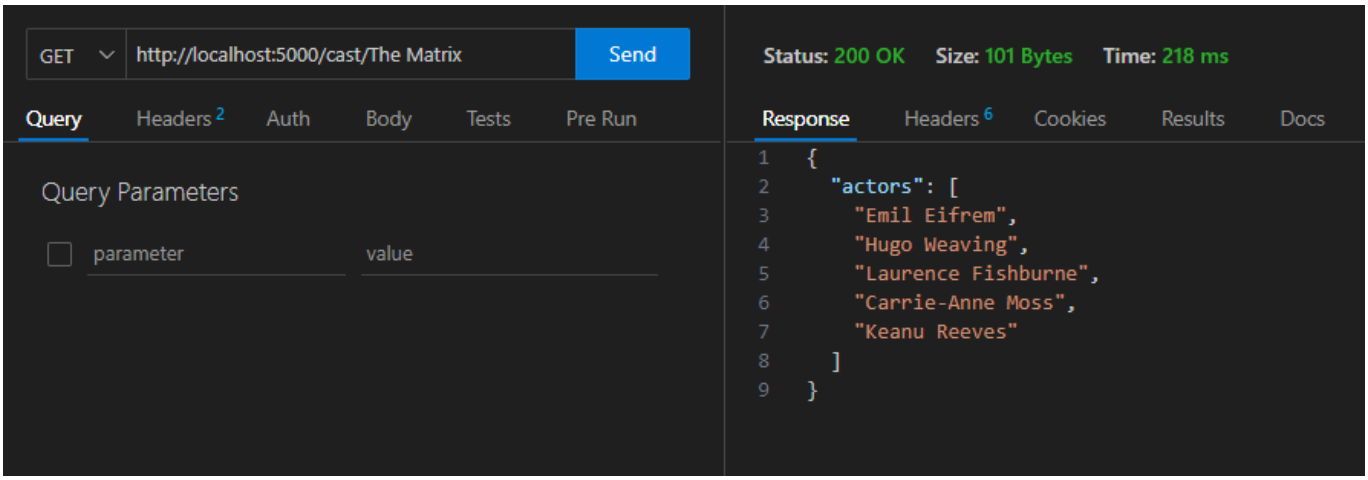

GET
http://localhost:5000/movies/The Matrix
Send
Query
Headers 2
Auth
Body
Tests
Pre Run
Query Parameters
parameter
value
Status: 200 OK
Size: 616 Bytes
Time: 1.34 s
Response
Headers 6
Cookies
Results
Docs
{
"title": "The Matrix",
"cast": [
{
"name": "Emil Eifrem",
"relationship": "acted",
"roles": [
"Emil"
]
},
{
"name": "Joel Silver",
"relationship": "produced",
"roles": null
},
{
"name": "Lana Wachowski",
"relationship": "directed",
"roles": null
},
{
"name": "Lilly Wachowski",
"relationship": "directed",
"roles": null
},
{
"name": "Hugo Weaving",
"relationship": "acted",
"roles": [
"Agent Smith"
]
},
{
"name": "Laurence Fishburne",
"relationship": "acted",
"roles": [
"Morpheus"
]
},
{
"name": "Carrie-Anne Moss",
"relationship": "acted",
Response
Chart

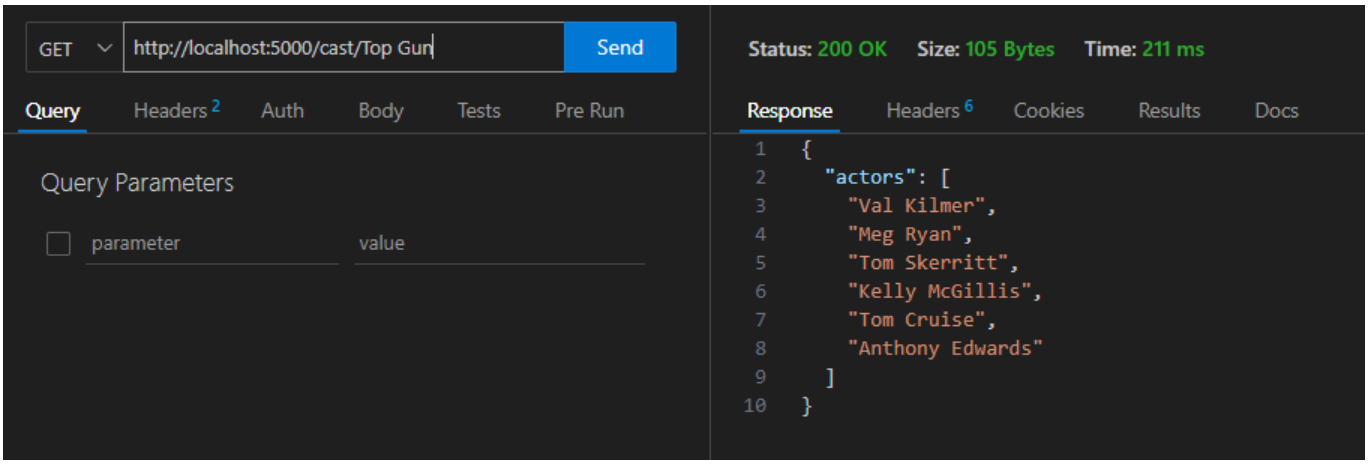

GET
http://localhost:5000/actor/Keanu Reeves
Send
Query
Headers 2
Auth
Body
Tests
Pre Run
Query Parameters
parameter
value
Status: 200 OK
Size: 164 Bytes
Time: 206 ms
Response
Headers 6
Cookies
Results
Docs
{
"movies": [
"Something's Gotta Give",
"The Replacements",
"Johnny Mnemonic",
"The Devil's Advocate",
"The Matrix Revolutions",
"The Matrix Reloaded",
"The Matrix"
]
}
Response
Chart


GET
http://localhost:5000/directores/Lilly Wacho
Send
Query
Headers 2
Auth
Body
Tests
Pre Run
Query Parameters
parameter
value
Status: 200 OK
Size: 105 Bytes
Time: 204 ms
Response
Headers 6
Cookies
Results
Docs
{
"movies": [
"Speed Racer",
"Cloud Atlas",
"The Matrix Revolutions",
"The Matrix Reloaded",
"The Matrix"
]
}
Response
Chart


GET
http://localhost:5000/producer/Joel Silver
Send
Query
Headers 2
Auth
Body
Tests
Pre Run
Query Parameters
parameter
value
Status: 200 OK
Size: 126 Bytes
Time: 217 ms
Response
Headers 6
Cookies
Results
Docs
{
"movies": [
"Speed Racer",
"V for Vendetta",
"Ninja Assassin",
"The Matrix Revolutions",
"The Matrix Reloaded",
"The Matrix"
]
}
Response
Chart

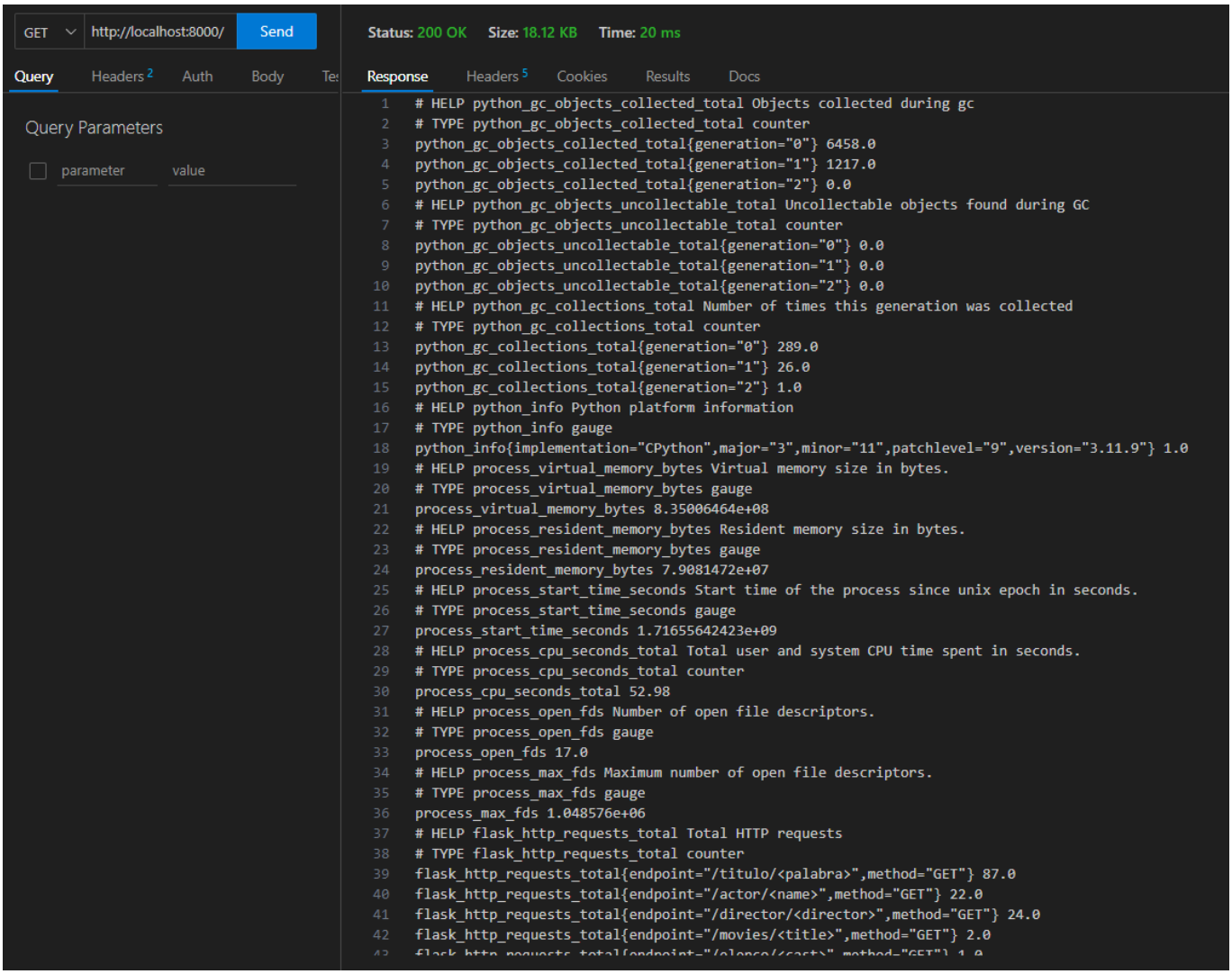

GET
http://localhost:5000/plot/The Matrix
Send
Query
Headers 2
Auth
Body
Tests
Pre Run
Query Parameters
parameter
value
Status: 200 OK
Size: 37 Bytes
Time: 214 ms
Response
Headers 6
Cookies
Results
Docs
{
"plot": "Welcome to the Real World"
}

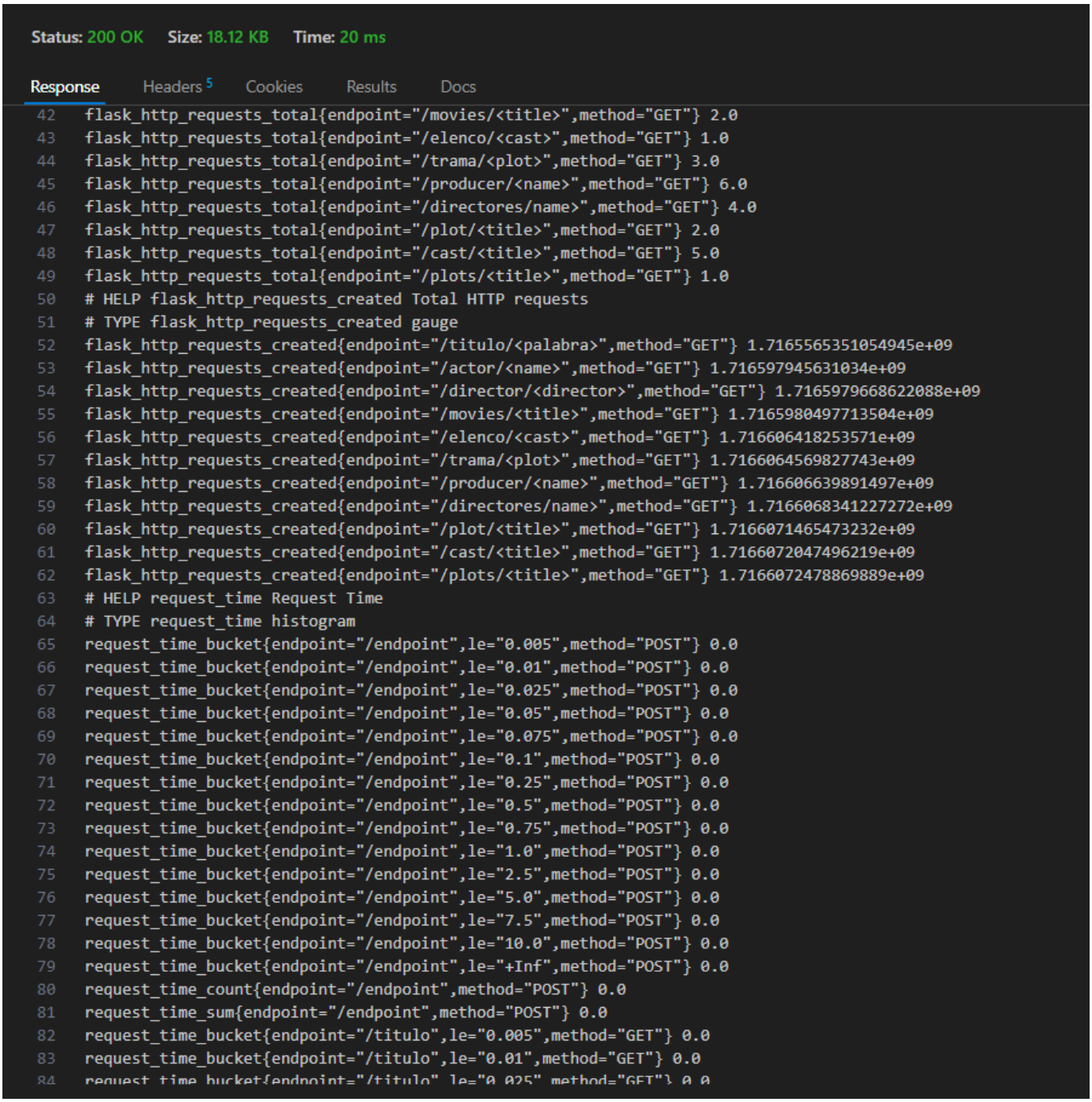




Métricas del API

Estas son las métricas resultantes de distintos request hechos a cada endpoint de la API, estos se pueden ver en el puerto 8000 de la computadora. En estas se ve reflejado la cantidad de request hechas al endpoint y el tiempo de duración de los request.

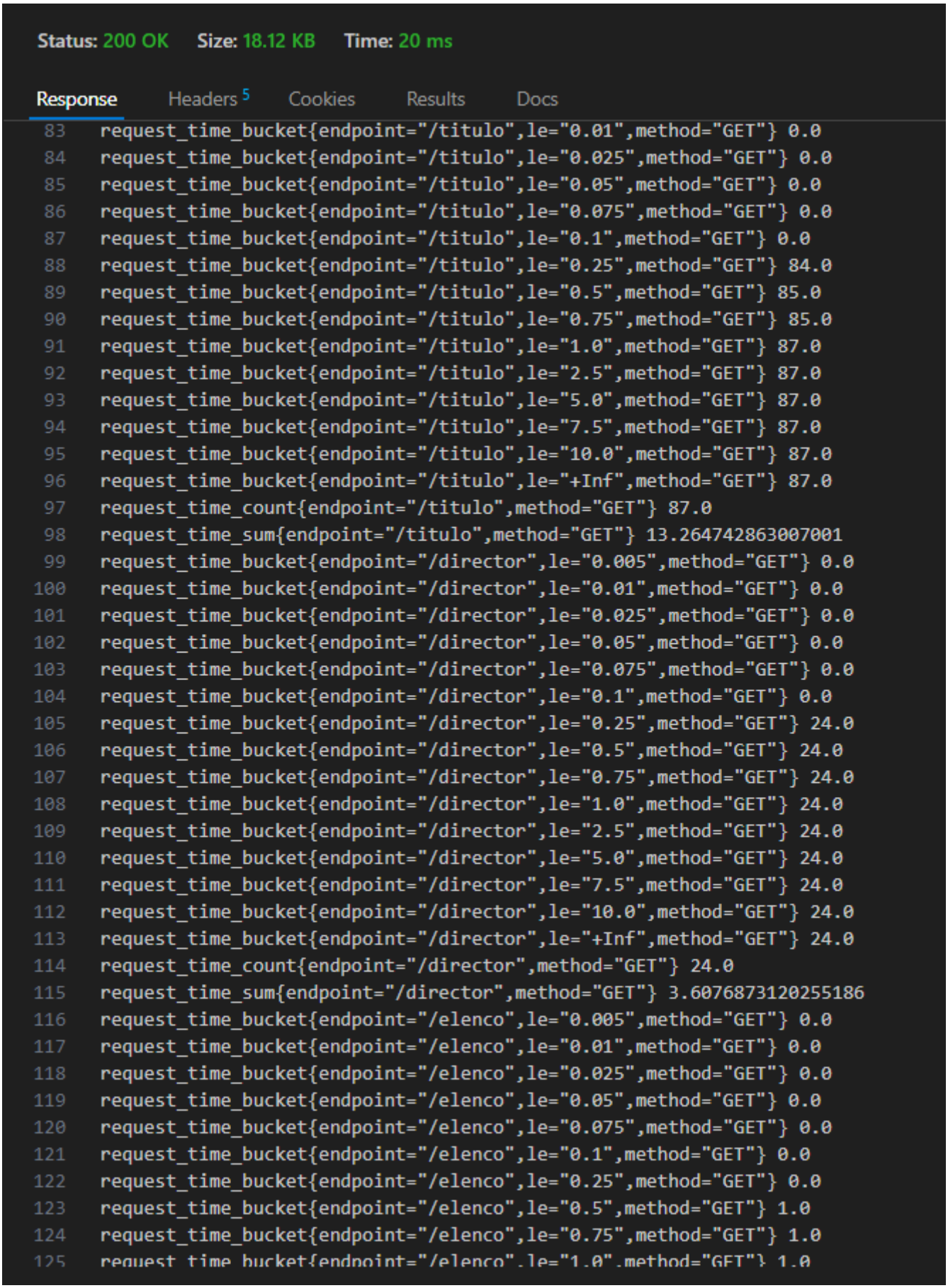

GET
http://localhost:8000/
Send
Status: 200 OK
Size: 18.12 KB
Time: 20 ms
Query
Headers 2
Auth
Body
Response
Headers 5
Cookies
Results
Docs
Query Parameters
parameter
value
HELP python_gc_objects_collected_total Objects collected during gc
TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 6458.0
python_gc_objects_collected_total{generation="1"} 1217.0
python_gc_objects_collected_total{generation="2"} 0.0
HELP python_gc_objects_uncollectable_total Uncollectable objects found during GC
TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
HELP python_gc_collections_total Number of times this generation was collected
TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 289.0
python_gc_collections_total{generation="1"} 26.0
python_gc_collections_total{generation="2"} 1.0
HELP python_info Python platform information
TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="11",patchlevel="9",version="3.11.9"} 1.0
HELP process_virtual_memory_bytes Virtual memory size in bytes.
TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 8.35006464e+08
HELP process_resident_memory_bytes Resident memory size in bytes.
TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 7.9081472e+07
HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
TYPE process_start_time_seconds gauge
process_start_time_seconds 1.71655642423e+09
HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 52.98
HELP process_open_fds Number of open file descriptors.
TYPE process_open_fds gauge
process_open_fds 17.0
HELP process_max_fds Maximum number of open file descriptors.
TYPE process_max_fds gauge
process_max_fds 1.048576e+06
HELP flask_http_requests_total Total HTTP requests
TYPE flask_http_requests_total counter
flask_http_requests_total{endpoint="/titulo/<palabra>",method="GET"} 87.0
flask_http_requests_total{endpoint="/actor/<name>",method="GET"} 22.0
flask_http_requests_total{endpoint="/director/<director>",method="GET"} 24.0
flask_http_requests_total{endpoint="/movies/<title>",method="GET"} 2.0



Status: 200 OK Size: 18.12 KB Time: 20 ms

Response Headers 5 Cookies Results Docs

```
flask_http_requests_total{endpoint="/movies/<title>",method="GET"} 2.0
flask_http_requests_total{endpoint="/elenco/<cast>",method="GET"} 1.0
flask_http_requests_total{endpoint="/trama/<plot>",method="GET"} 3.0
flask_http_requests_total{endpoint="/producer/<name>",method="GET"} 6.0
flask_http_requests_total{endpoint="/directores/name>",method="GET"} 4.0
flask_http_requests_total{endpoint="/plot/<title>",method="GET"} 2.0
flask_http_requests_total{endpoint="/cast/<title>",method="GET"} 5.0
flask_http_requests_total{endpoint="/plots/<title>",method="GET"} 1.0
# HELP flask_http_requests_created Total HTTP requests
# TYPE flask_http_requests_created gauge
flask_http_requests_created{endpoint="/titulo/<palabra>",method="GET"} 1.7165565351054945e+09
flask_http_requests_created{endpoint="/actor/<name>",method="GET"} 1.716597945631034e+09
flask_http_requests_created{endpoint="/director/<director>",method="GET"} 1.7165979668622088e+09
flask_http_requests_created{endpoint="/movies/<title>",method="GET"} 1.7165980497713504e+09
flask_http_requests_created{endpoint="/elenco/<cast>",method="GET"} 1.716606418253571e+09
flask_http_requests_created{endpoint="/trama/<plot>",method="GET"} 1.7166064569827743e+09
flask_http_requests_created{endpoint="/producer/<name>",method="GET"} 1.716606639891497e+09
flask_http_requests_created{endpoint="/directores/name>",method="GET"} 1.7166068341227272e+09
flask_http_requests_created{endpoint="/plot/<title>",method="GET"} 1.7166071465473232e+09
flask_http_requests_created{endpoint="/cast/<title>",method="GET"} 1.7166072047496219e+09
flask_http_requests_created{endpoint="/plots/<title>",method="GET"} 1.7166072478869889e+09
# HELP request_time Request Time
# TYPE request_time histogram
request_time_bucket{endpoint="/endpoint",le="0.005",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="0.01",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="0.025",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="0.05",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="0.075",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="0.1",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="0.25",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="0.5",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="0.75",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="1.0",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="2.5",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="5.0",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="7.5",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="10.0",method="POST"} 0.0
request_time_bucket{endpoint="/endpoint",le="+Inf",method="POST"} 0.0
request_time_count{endpoint="/endpoint",method="POST"} 0.0
request_time_sum{endpoint="/endpoint",method="POST"} 0.0
request_time_bucket{endpoint="/titulo",le="0.005",method="GET"} 0.0
request_time_bucket{endpoint="/titulo",le="0.01",method="GET"} 0.0
request_time_bucket{endpoint="/titulo",le="0.025",method="GET"} 0.0
```


Status: 200 OK Size: 18.12 KB Time: 20 ms
Response Headers 5 Cookies Results Docs
request_time_bucket{endpoint="/titulo",le="0.01",method="GET"} 0.0
request_time_bucket{endpoint="/titulo",le="0.025",method="GET"} 0.0
request_time_bucket{endpoint="/titulo",le="0.05",method="GET"} 0.0
request_time_bucket{endpoint="/titulo",le="0.075",method="GET"} 0.0
request_time_bucket{endpoint="/titulo",le="0.1",method="GET"} 0.0
request_time_bucket{endpoint="/titulo",le="0.25",method="GET"} 84.0
request_time_bucket{endpoint="/titulo",le="0.5",method="GET"} 85.0
request_time_bucket{endpoint="/titulo",le="0.75",method="GET"} 85.0
request_time_bucket{endpoint="/titulo",le="1.0",method="GET"} 87.0
request_time_bucket{endpoint="/titulo",le="2.5",method="GET"} 87.0
request_time_bucket{endpoint="/titulo",le="5.0",method="GET"} 87.0
request_time_bucket{endpoint="/titulo",le="7.5",method="GET"} 87.0
request_time_bucket{endpoint="/titulo",le="10.0",method="GET"} 87.0
request_time_bucket{endpoint="/titulo",le="+Inf",method="GET"} 87.0
request_time_count{endpoint="/titulo",method="GET"} 87.0
request_time_sum{endpoint="/titulo",method="GET"} 13.264742863007001
request_time_bucket{endpoint="/director",le="0.005",method="GET"} 0.0
request_time_bucket{endpoint="/director",le="0.01",method="GET"} 0.0
request_time_bucket{endpoint="/director",le="0.025",method="GET"} 0.0
request_time_bucket{endpoint="/director",le="0.05",method="GET"} 0.0
request_time_bucket{endpoint="/director",le="0.075",method="GET"} 0.0
request_time_bucket{endpoint="/director",le="0.1",method="GET"} 0.0
request_time_bucket{endpoint="/director",le="0.25",method="GET"} 24.0
request_time_bucket{endpoint="/director",le="0.5",method="GET"} 24.0
request_time_bucket{endpoint="/director",le="0.75",method="GET"} 24.0
request_time_bucket{endpoint="/director",le="1.0",method="GET"} 24.0
request_time_bucket{endpoint="/director",le="2.5",method="GET"} 24.0
request_time_bucket{endpoint="/director",le="5.0",method="GET"} 24.0
request_time_bucket{endpoint="/director",le="7.5",method="GET"} 24.0
request_time_bucket{endpoint="/director",le="10.0",method="GET"} 24.0
request_time_bucket{endpoint="/director",le="+Inf",method="GET"} 24.0
request_time_count{endpoint="/director",method="GET"} 24.0
request_time_sum{endpoint="/director",method="GET"} 3.6076873120255186
request_time_bucket{endpoint="/elenco",le="0.005",method="GET"} 0.0
request_time_bucket{endpoint="/elenco",le="0.01",method="GET"} 0.0
request_time_bucket{endpoint="/elenco",le="0.025",method="GET"} 0.0
request_time_bucket{endpoint="/elenco",le="0.05",method="GET"} 0.0
request_time_bucket{endpoint="/elenco",le="0.075",method="GET"} 0.0
request_time_bucket{endpoint="/elenco",le="0.1",method="GET"} 0.0
request_time_bucket{endpoint="/elenco",le="0.25",method="GET"} 0.0
request_time_bucket{endpoint="/elenco",le="0.5",method="GET"} 1.0
request_time_bucket{endpoint="/elenco",le="0.75",method="GET"} 1.0

Status: 200 OK Size: 18.12 KB Time: 20 ms

Response Headers 5 Cookies Results Docs

```
request_time_bucket{endpoint="/elenco",le="0.75",method="GET"} 1.0
request_time_bucket{endpoint="/elenco",le="1.0",method="GET"} 1.0
request_time_bucket{endpoint="/elenco",le="2.5",method="GET"} 1.0
request_time_bucket{endpoint="/elenco",le="5.0",method="GET"} 1.0
request_time_bucket{endpoint="/elenco",le="7.5",method="GET"} 1.0
request_time_bucket{endpoint="/elenco",le="10.0",method="GET"} 1.0
request_time_bucket{endpoint="/elenco",le="+Inf",method="GET"} 1.0
request_time_count{endpoint="/elenco",method="GET"} 1.0
request_time_sum{endpoint="/elenco",method="GET"} 0.30762096899707103
request_time_bucket{endpoint="/trama",le="0.005",method="GET"} 0.0
request_time_bucket{endpoint="/trama",le="0.01",method="GET"} 0.0
request_time_bucket{endpoint="/trama",le="0.025",method="GET"} 0.0
request_time_bucket{endpoint="/trama",le="0.05",method="GET"} 0.0
request_time_bucket{endpoint="/trama",le="0.075",method="GET"} 0.0
request_time_bucket{endpoint="/trama",le="0.1",method="GET"} 0.0
request_time_bucket{endpoint="/trama",le="0.25",method="GET"} 2.0
request_time_bucket{endpoint="/trama",le="0.5",method="GET"} 3.0
request_time_bucket{endpoint="/trama",le="0.75",method="GET"} 3.0
request_time_bucket{endpoint="/trama",le="1.0",method="GET"} 3.0
request_time_bucket{endpoint="/trama",le="2.5",method="GET"} 3.0
request_time_bucket{endpoint="/trama",le="5.0",method="GET"} 3.0
request_time_bucket{endpoint="/trama",le="7.5",method="GET"} 3.0
request_time_bucket{endpoint="/trama",le="10.0",method="GET"} 3.0
request_time_bucket{endpoint="/trama",le="+Inf",method="GET"} 3.0
request_time_count{endpoint="/trama",method="GET"} 3.0
request_time_sum{endpoint="/trama",method="GET"} 0.5040287270021508
request_time_bucket{endpoint="/movies",le="0.005",method="GET"} 0.0
request_time_bucket{endpoint="/movies",le="0.01",method="GET"} 0.0
request_time_bucket{endpoint="/movies",le="0.025",method="GET"} 0.0
request_time_bucket{endpoint="/movies",le="0.05",method="GET"} 0.0
request_time_bucket{endpoint="/movies",le="0.075",method="GET"} 0.0
request_time_bucket{endpoint="/movies",le="0.1",method="GET"} 0.0
request_time_bucket{endpoint="/movies",le="0.25",method="GET"} 1.0
request_time_bucket{endpoint="/movies",le="0.5",method="GET"} 1.0
request_time_bucket{endpoint="/movies",le="0.75",method="GET"} 1.0
request_time_bucket{endpoint="/movies",le="1.0",method="GET"} 1.0
request_time_bucket{endpoint="/movies",le="2.5",method="GET"} 2.0
request_time_bucket{endpoint="/movies",le="5.0",method="GET"} 2.0
request_time_bucket{endpoint="/movies",le="7.5",method="GET"} 2.0
request_time_bucket{endpoint="/movies",le="10.0",method="GET"} 2.0
request_time_bucket{endpoint="/movies",le="+Inf",method="GET"} 2.0
request_time_count{endpoint="/movies",method="GET"} 2.0
request_time_sum{endpoint="/movies",method="GET"} 1.5618032499987748
```

Status: 200 OK Size: 18.12 KB Time: 20 ms

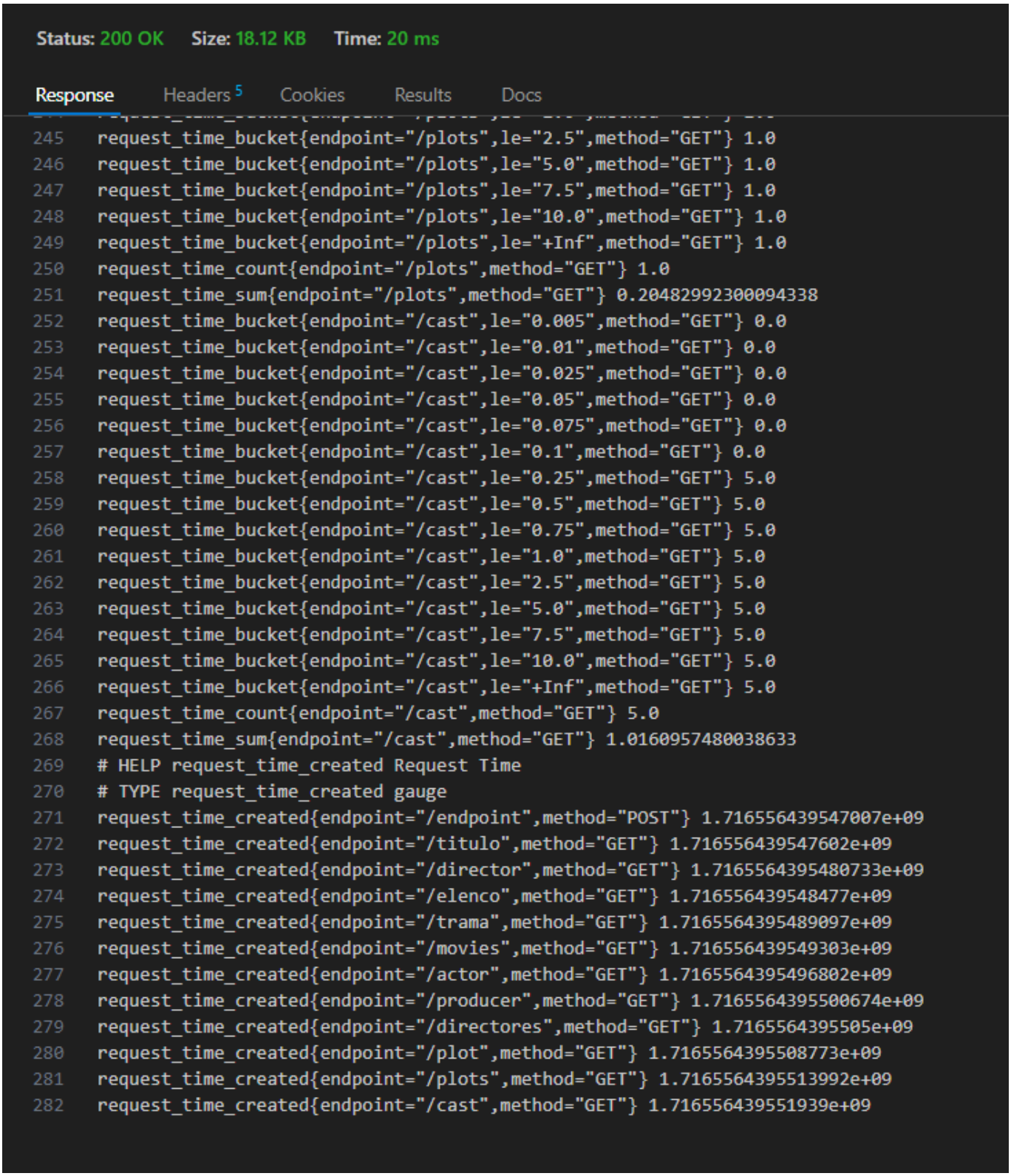
Response Headers 5 Cookies Results Docs

```
request_time_count{endpoint="/movies",method="GET"} 2.0
request_time_sum{endpoint="/movies",method="GET"} 1.5618032499987748
request_time_bucket{endpoint="/actor",le="0.005",method="GET"} 0.0
request_time_bucket{endpoint="/actor",le="0.01",method="GET"} 0.0
request_time_bucket{endpoint="/actor",le="0.025",method="GET"} 0.0
request_time_bucket{endpoint="/actor",le="0.05",method="GET"} 0.0
request_time_bucket{endpoint="/actor",le="0.075",method="GET"} 0.0
request_time_bucket{endpoint="/actor",le="0.1",method="GET"} 0.0
request_time_bucket{endpoint="/actor",le="0.25",method="GET"} 20.0
request_time_bucket{endpoint="/actor",le="0.5",method="GET"} 20.0
request_time_bucket{endpoint="/actor",le="0.75",method="GET"} 20.0
request_time_bucket{endpoint="/actor",le="1.0",method="GET"} 20.0
request_time_bucket{endpoint="/actor",le="2.5",method="GET"} 22.0
request_time_bucket{endpoint="/actor",le="5.0",method="GET"} 22.0
request_time_bucket{endpoint="/actor",le="7.5",method="GET"} 22.0
request_time_bucket{endpoint="/actor",le="10.0",method="GET"} 22.0
request_time_bucket{endpoint="/actor",le="+Inf",method="GET"} 22.0
request_time_count{endpoint="/actor",method="GET"} 22.0
request_time_sum{endpoint="/actor",method="GET"} 6.593826174008427
request_time_bucket{endpoint="/producer",le="0.005",method="GET"} 0.0
request_time_bucket{endpoint="/producer",le="0.01",method="GET"} 0.0
request_time_bucket{endpoint="/producer",le="0.025",method="GET"} 0.0
request_time_bucket{endpoint="/producer",le="0.05",method="GET"} 0.0
request_time_bucket{endpoint="/producer",le="0.075",method="GET"} 0.0
request_time_bucket{endpoint="/producer",le="0.1",method="GET"} 0.0
request_time_bucket{endpoint="/producer",le="0.25",method="GET"} 6.0
request_time_bucket{endpoint="/producer",le="0.5",method="GET"} 6.0
request_time_bucket{endpoint="/producer",le="0.75",method="GET"} 6.0
request_time_bucket{endpoint="/producer",le="1.0",method="GET"} 6.0
request_time_bucket{endpoint="/producer",le="2.5",method="GET"} 6.0
request_time_bucket{endpoint="/producer",le="5.0",method="GET"} 6.0
request_time_bucket{endpoint="/producer",le="7.5",method="GET"} 6.0
request_time_bucket{endpoint="/producer",le="10.0",method="GET"} 6.0
request_time_bucket{endpoint="/producer",le="+Inf",method="GET"} 6.0
request_time_count{endpoint="/producer",method="GET"} 6.0
request_time_sum{endpoint="/producer",method="GET"} 1.2459044089991949
request_time_bucket{endpoint="/directores",le="0.005",method="GET"} 0.0
request_time_bucket{endpoint="/directores",le="0.01",method="GET"} 0.0
request_time_bucket{endpoint="/directores",le="0.025",method="GET"} 0.0
request_time_bucket{endpoint="/directores",le="0.05",method="GET"} 0.0
request_time_bucket{endpoint="/directores",le="0.075",method="GET"} 0.0
request_time_bucket{endpoint="/directores",le="0.1",method="GET"} 0.0
request_time_bucket{endpoint="/directores",le="0.25",method="GET"} 4.0
```

Status: 200 OK Size: 18.12 KB Time: 20 ms

Response Headers 5 Cookies Results Docs

```
request_time_bucket{endpoint="/directores",le="0.1",method="GET"} 0.0
request_time_bucket{endpoint="/directores",le="0.25",method="GET"} 4.0
request_time_bucket{endpoint="/directores",le="0.5",method="GET"} 4.0
request_time_bucket{endpoint="/directores",le="0.75",method="GET"} 4.0
request_time_bucket{endpoint="/directores",le="1.0",method="GET"} 4.0
request_time_bucket{endpoint="/directores",le="2.5",method="GET"} 4.0
request_time_bucket{endpoint="/directores",le="5.0",method="GET"} 4.0
request_time_bucket{endpoint="/directores",le="7.5",method="GET"} 4.0
request_time_bucket{endpoint="/directores",le="10.0",method="GET"} 4.0
request_time_bucket{endpoint="/directores",le="+Inf",method="GET"} 4.0
request_time_count{endpoint="/directores",method="GET"} 4.0
request_time_sum{endpoint="/directores",method="GET"} 0.8471278970100684
request_time_bucket{endpoint="/plot",le="0.005",method="GET"} 0.0
request_time_bucket{endpoint="/plot",le="0.01",method="GET"} 0.0
request_time_bucket{endpoint="/plot",le="0.025",method="GET"} 0.0
request_time_bucket{endpoint="/plot",le="0.05",method="GET"} 0.0
request_time_bucket{endpoint="/plot",le="0.075",method="GET"} 0.0
request_time_bucket{endpoint="/plot",le="0.1",method="GET"} 0.0
request_time_bucket{endpoint="/plot",le="0.25",method="GET"} 2.0
request_time_bucket{endpoint="/plot",le="0.5",method="GET"} 2.0
request_time_bucket{endpoint="/plot",le="0.75",method="GET"} 2.0
request_time_bucket{endpoint="/plot",le="1.0",method="GET"} 2.0
request_time_bucket{endpoint="/plot",le="2.5",method="GET"} 2.0
request_time_bucket{endpoint="/plot",le="5.0",method="GET"} 2.0
request_time_bucket{endpoint="/plot",le="7.5",method="GET"} 2.0
request_time_bucket{endpoint="/plot",le="10.0",method="GET"} 2.0
request_time_bucket{endpoint="/plot",le="+Inf",method="GET"} 2.0
request_time_count{endpoint="/plot",method="GET"} 2.0
request_time_sum{endpoint="/plot",method="GET"} 0.40571065899712266
request_time_bucket{endpoint="/plots",le="0.005",method="GET"} 0.0
request_time_bucket{endpoint="/plots",le="0.01",method="GET"} 0.0
request_time_bucket{endpoint="/plots",le="0.025",method="GET"} 0.0
request_time_bucket{endpoint="/plots",le="0.05",method="GET"} 0.0
request_time_bucket{endpoint="/plots",le="0.075",method="GET"} 0.0
request_time_bucket{endpoint="/plots",le="0.1",method="GET"} 0.0
request_time_bucket{endpoint="/plots",le="0.25",method="GET"} 1.0
request_time_bucket{endpoint="/plots",le="0.5",method="GET"} 1.0
request_time_bucket{endpoint="/plots",le="0.75",method="GET"} 1.0
request_time_bucket{endpoint="/plots",le="1.0",method="GET"} 1.0
request_time_bucket{endpoint="/plots",le="2.5",method="GET"} 1.0
request_time_bucket{endpoint="/plots",le="5.0",method="GET"} 1.0
request_time_bucket{endpoint="/plots",le="7.5",method="GET"} 1.0
request_time_bucket{endpoint="/plots",le="10.0",method="GET"} 1.0
```



Recomendaciones

- Utilizar los comandos de Docker con sumo cuidado. Partes pequeñas del comando por correr pueden tener un gran impacto en qué contenedores o imágenes se crean. Algunos componentes del proyecto cuentan con imagenes oficiales de Docker, por lo que si no se es cuidadoso se puede llegar a crear o correr una imagen que no corresponde a la del proyecto si no a esta imagen oficial. Siempre utilizar los comandos de docker incluyendo su usuario propio para evitar confusiones.

- Utilizar índices de busqueda para trabajar con Mongo Atlas. Estos incrementan el rendimiento de las consultas.

- Estudiar bien la documentación de ambas bases de datos. Al ser de modelos que pueden ser relativamente poco conocidos, como el modelo de documentos y de grafos, puede conllevar cierta curva de aprendizaje.
- Familiarizarse con el lenguaje de query de Neo4J. Este implementa su propio lenguaje para realizar consultas, el cual está sumamente orientado a las relaciones por lo que se utiliza de forma muy distinta al SQL conocido. Estas funcione s se vuelven importantes para la implementación del API para Neo4J.
- Se recomienda probar los índices de Mongo Atlas, para asegurarse que el mapping sea el adecuado para la búsqueda.
- En caso de utilizar el método gratuito de Mongo Atlas, es iportante tener en cuenta que solo se pueden crear 3 índices. Es por esto que elegir la búsqueda que se hará sin índices antes de comenzar ahorra tiempo y trabajo.
- Monitorear el sistema utilizando Grafana y Prometheus, con el fin de obtener metricas para ver el rendimiento del sistema bajo un flujo de datos.
- Tomar en cuenta medidas de seguridad, para que los usuarios que utilicen la aplicación sean autenticados y verificados para un retorno de datos. Esto puede evitar problemas con las conexiones a bases de datos.
- Optimizar las consultas de Neo4j y Mongo Atlas mediante el uso de índices puede ser conveniente para el rendimiento de la aplicación.
- Es importante tener Python instalado, y sabes ubicar la ubicación del interprete para lograr instalar las librerias y dependencias mediante "PIP" sin problema alguno.
- El uso de Github es necesario para la implementacion de un proyecto con tantas partes. Su uso simplifica el trabajo en equipo para un resultado mas ordenado y fácil. Mediante la herramienta de GitHub Desktop se pueden realizar todo tipo de modificaciones al repositorio del proyecto.

Conclusiones

- El uso de bases de datos de grafos como Neo4j están muy bien diseñadas para la representación de datos con gran cantidad de relaciones. Esta forma de representarlos se aproxima de manera más acertada a la forma humana de visualizar relaciones entre datos que el modelo relacional de bases de datos.
- El uso de Ngrok es una manera relativamente fácil de permitir conexiones remotas a un API. Esto facilitó mucho la implementación y el testeo del API.
- Neo4j es una herramienta muy útil para manejar datos con muchas relaciones. Ekl hecho de que las consultas sean mediante grafos facilita la comprensión y análisis de relaciones complejas.
- MongoDB Atlas es muy flexible a la hora de manejar grandes volumenes de datos no estructurados. También es muy bueno a la hora de realizar busquedas y almacenar información.
- El uso de Thunkable junto a Firebase facilita la creación de aplicaciones funcionales sin necesidad de desarrollar código extenso y complicado.

- La importancia de monitorear datos para medir el rendimiento de la base de datos y el API permite observar que el sistema si funcione sin interrupciuones ni problemas. De igual manera nos ha ayudado con la resolución de problemas.
- La utilización de tecnologías tan nuevas es muy beneficioso para una aplicación, ya que es más escalable y más eficiente que el uso de tecnologías desactualizadas.
- El uso de archivos JSON para enviar datos entre la base de datos y la aplicacion Thunkable es fundamental para facilitar el manejo de datos.
- El uso de CORS al implementar FLASK-Cors fue muy importante en nuestra aplicación de búsqueda de películas para facilitar la interacción entre el frontend y el backend.
- El desarrollo de la aplicacion fue fundamental para el aprendizaje acerca de conectar multiples tecnologias a la vez. De la misma manera para controlarlas por medio de comandos de consola.