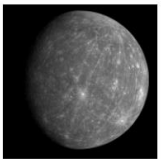
Análisis Informe Api

Esta es un api que está basada en planetas, donde está la información de si el planeta tiene agua, cuantos satélites posee, y el tiempo orbital alrededor del sol.



Mercurio

Satellites	Orbital periods	Has water?
0	0.24	Yes

Tierra

Satellites	Orbital periods	Has water?
1	1	Yes

El objetivo de este documento usando la herramienta apipecker desarrollada por Pablo Fernández, es a hacer unas series de request al endpoint de analítica y decir cuántos usuarios concurrentes soporta.

Endpoint: */api/v1/planets/analytics/{limit}*

***Param**: limit – Ejemplo /api/v1/planets/analytics/5*

El endpoint lo que calcula, el promedio de satélites y el tiempo orbital de todos los planetas registrados además de decir cuántos tienen agua y cuantos no.

El parámetro limite servirá para delimitar a cuantos recursos se le quiere hacer la operación, ya que si se tiene muchos recursos en la base de datos puede ser pesado, por ende, dejo al usuario elegir de cuanto quiere hacer el request.

Antes de empezar a mostrar los resultados dejare los recursos a los que el api tenía acceso:

- Procesador: AMD Ryzen 9 4900HS with Radeon Graphics 3.00 GHz
- RAM: 40 GB

Tabla de Parámetros

ID	Usuarios Concurrente	Interacciones	Retraso en ms	Recursos de Planetas
A1	1	100	1000	100k
A2	10	1	1000	100k
A3	100	1	1000	100k
A4	100	1	100	100

Tabla de Resultados

ID	Tiempo Mínimo en ms	Tiempo Máximo en ms	Tiempo promedio en ms	Desviación Estándar en ms
A1	409	1002	640	149
A2	6936	8164	7896	405
A3	62629	280279	210121	55545
A4	231.354	1085	555	253

Como se puede ver para el A1 con un usuario concurrente puede llegar hasta 1 segundo en tiempo máximo y un tiempo mínimo de 409, si este hiciera 100 interacciones.

A2: Con 10 Usuarios concurrente se puede ver que es un déficit y que seguirá creciendo exponencialmente mientras Mas usuarios existan. Ya que es un request muy pesado con 100k de recursos.

A3: A medida que se aumenten crecerá constantemente el tiempo de respuesta.

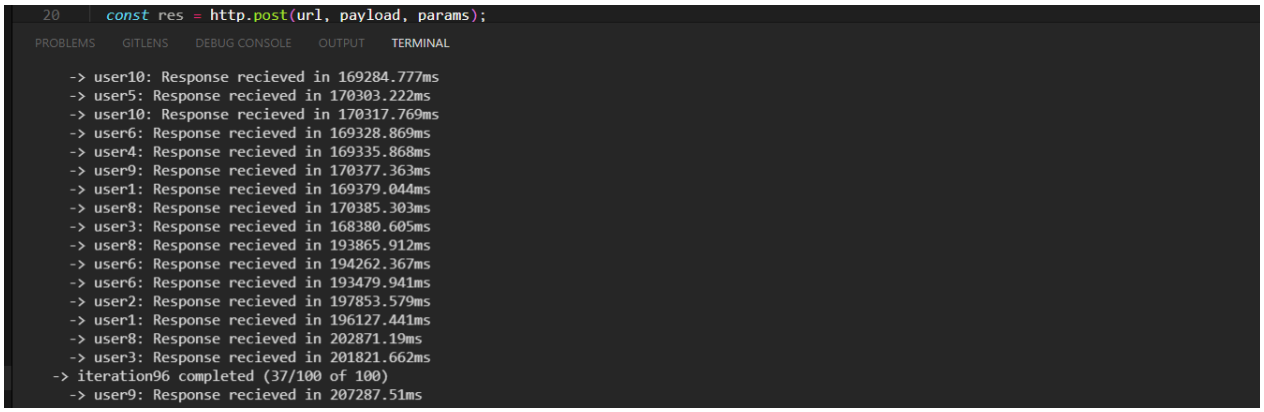
A4: Aquí hicimos una prueba en vez de usar 100k de recursos se utilizó 100 solamente, y como resultado dio menos de un segundo en latencia. El resultado vario tanto si solo limitas el request a una parte de los recursos

Referencias:

https://www.npmjs.com/package/apipecker

Evidencias:

Evidencia de A2



Evidencia de A4

```
    -> user96: Response recieved in 1071.569ms
    -> user97: Response recieved in 1072.815ms
    -> user100: Response recieved in 1073.727ms
    -> user98: Response recieved in 1077.297ms
    -> user95: Response recieved in 1081.692ms
    -> user94: Response recieved in 1084.94ms
    -> user99: Response recieved in 1085.537ms
  -> iteration1 completed (1/1 of 1)

Result:
{
  "count": 100,
  "min": 231.354,
  "max": 1085.537,
  "mean": 555.909,
  "std": 253.831
}
```