Prueba técnica Back-end
Teclogi

Realizada por: Laura Sofía Dueñas Bulla

Febrero 2022

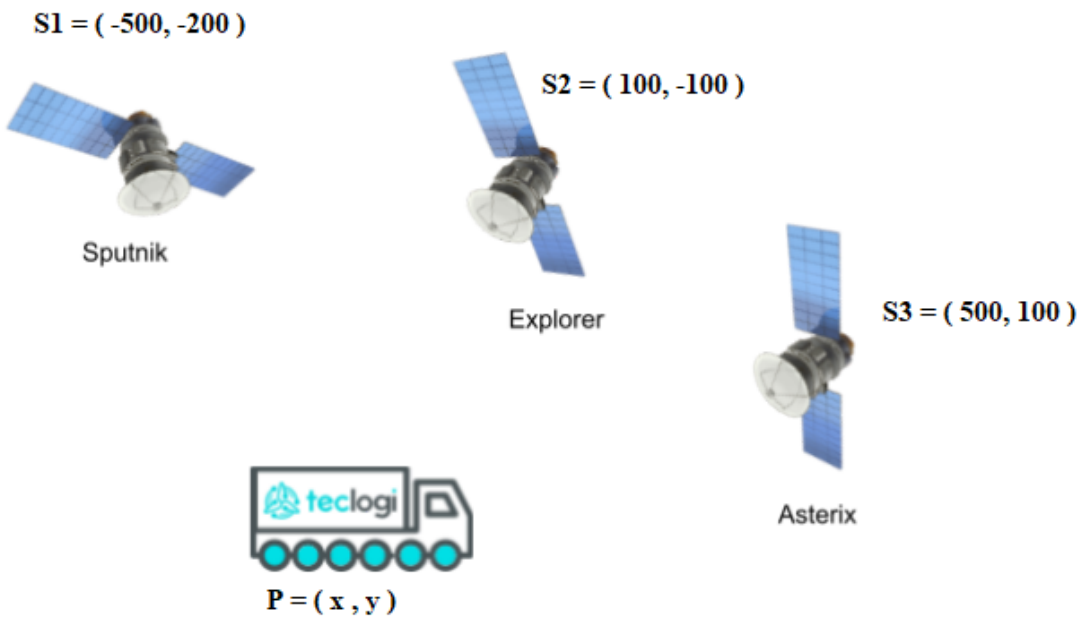
Tabla de contenido

1. Planteamiento del problema

Debido a que el área de Seguridad ha detectado un llamado de auxilio de un vehículo de carga a la deriva en una carretera; es necesario conocer su posición y estado actual (En peligro/ Seguro). Teniendo en cuenta:

- ➔ La ubicación actual de 3 satélites (Sputnik, Explorer, Asterix).
- ➔ Las distancias que existen entre la ubicación del vehículo y cada uno de los satélites.
- ➔ El mensaje codificado que ha sido enviado por el vehículo.

La posición de cada satélite se observa a continuación:



Y el mensaje codificado es enviado en una matriz como se observa a continuación:

A	T	G	C	G	A
C	A	G	T	G	C
T	T	A	T	T	T
A	G	A	C	G	G
G	C	G	T	C	A
T	C	A	C	T	G

¡Seguro!

A	T	G	C	G	A
C	A	G	T	G	C
T	T	A	T	G	T
A	G	A	A	G	G
C	C	C	C	T	A
T	C	A	C	T	G

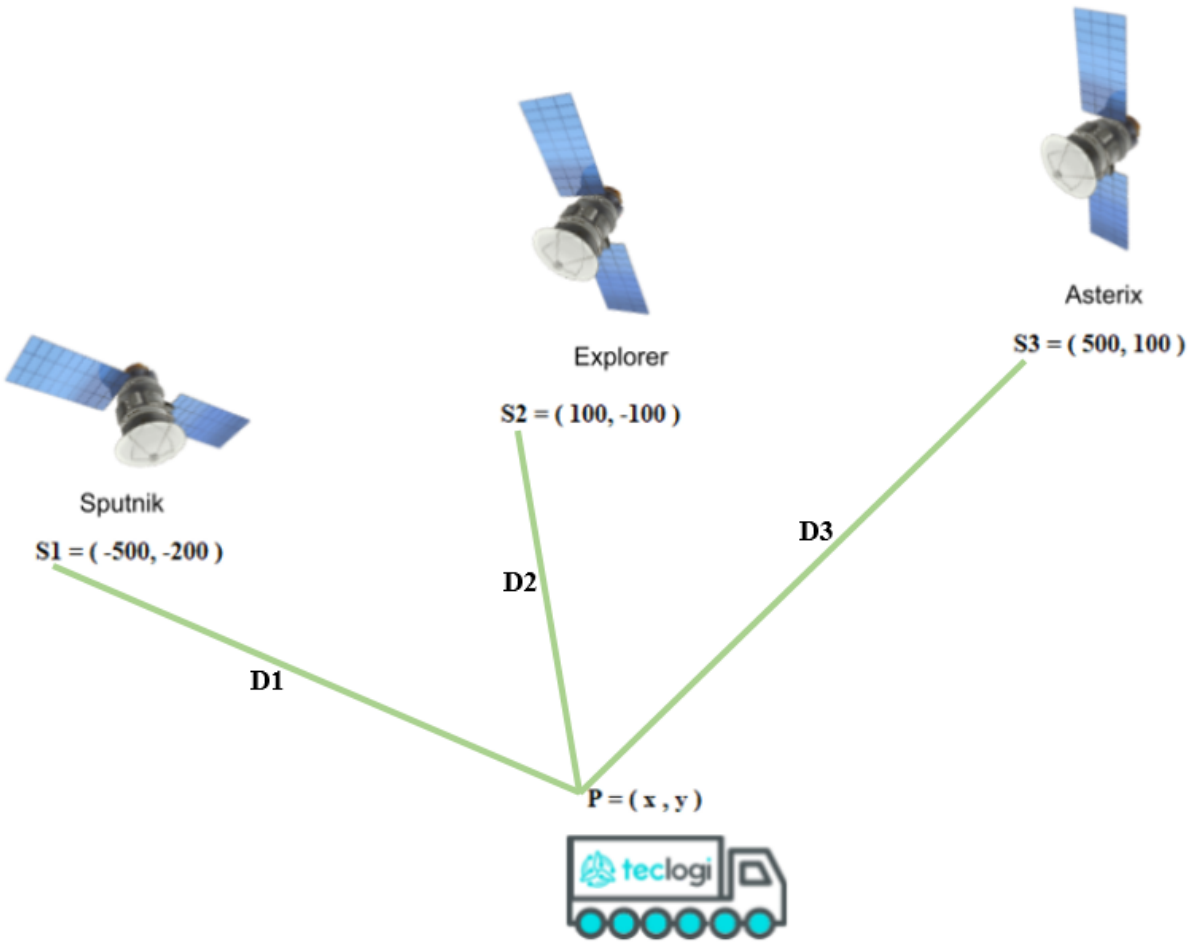
¡En peligro!

En donde si dentro de la matriz se encuentra más de una secuencia de 4 letras iguales de manera vertical, horizontal, o diagonal, el vehículo estará en peligro, de lo contrario estará seguro.

2. Solución del problema

2.1 Algoritmo 1

Este algoritmo permite obtener la posición actual del vehículo, para su desarrollo se realizó el siguiente análisis:



Como se puede observar la distancia entre cada satélite y el vehículo se puede hallar mediante la fórmula de distancia entre dos puntos:

$$d = \sqrt{(x - x_1)^2 + (y - y_1)^2}$$

Por ello al plantear las ecuaciones de distancia que hay entre cada satélite y el vehículo, teniendo en cuenta:

$P = (x, y)$

$S_1 = (-500, -200) = (x_1, y_1)$

$S_2 = (100, -100) = (x_2, y_2)$

$S_3 = (500, 100) = (x_3, y_3)$

Se obtiene:

$$d_1 = \sqrt{(x - x_1)^2 + (y - y_1)^2}$$

$$d_2 = \sqrt{(x - x_2)^2 + (y - y_2)^2}$$
$$d_3 = \sqrt{(x - x_3)^2 + (y - y_3)^2}$$

Ya que las distancias d_1, d_2, d_3 serán conocidas, puesto que se enviarán como parámetro al método getLocation, es posible despejar el valor de $x\ e\ y$ de las anteriores ecuaciones, sin embargo es pertinente elevar al cuadrado cada una de ellas, para eliminar la raíz:

$$d_1^{\ 2} = (x - x_1)^2 + (y - y_1)^2$$
$$d_2^{\ 2} = (x - x_2)^2 + (y - y_2)^2$$
$$d_3^{\ 2} = (x - x_3)^2 + (y - y_3)^2$$

Como se puede observar son ecuaciones de circunferencias, por ende se puede decir que las distancias d_1, d_2, d_3 corresponden a los radios r_1, r_2, r_3 de cada una de ellas, lo cual queda expresado como:

$$r_1^{\ 2} = (x - x_1)^2 + (y - y_1)^2 \quad (1)$$
$$r_2^{\ 2} = (x - x_2)^2 + (y - y_2)^2 \quad (2)$$
$$r_3^{\ 2} = (x - x_3)^2 + (y - y_3)^2 \quad (3)$$

Es decir que se necesita hallar el punto de corte de las 3 circunferencias, el cual corresponderá a la ubicación del vehículo. Ya que se encuentra un sistema de ecuaciones de segundo orden con 3 ecuaciones y 2 incógnitas, es posible obtener un sistema de primer orden al restar las ecuaciones (1) y (2) y las ecuaciones (2) y (3):

$(1) - (2)$

$$r_1^{\ 2} - r_2^{\ 2} = (x - x_1)^2 - (x - x_2)^2 + (y - y_1)^2 - (y - y_2)^2$$
$$r_1^{\ 2} - r_2^{\ 2} = (x - x_1 + x - x_2)(x - x_1 - (x - x_2)) + (y - y_1 + y - y_2)(y - y_1 - (y - y_2))$$
$$r_1^{\ 2} - r_2^{\ 2} = (2x - x_1 - x_2)(x_2 - x_1) + (2y - y_1 - y_2)(y_2 - y_1)$$
$$r_1^{\ 2} - r_2^{\ 2} = 2x(x_2 - x_1) + (-(x_1 + x_2)(-(x_1 - x_2)) + 2y(y_2 - y_1) + (-(y_1 + y_2)(-(y_1 - y_2)))$$
$$r_1^{\ 2} - r_2^{\ 2} = 2x(x_2 - x_1) + (x_1 + x_2)(x_1 - x_2) + 2y(y_2 - y_1) + (y_1 + y_2)(y_1 - y_2)$$
$$r_1^{\ 2} - r_2^{\ 2} = 2x(x_2 - x_1) + x_1^{\ 2} - x_2^{\ 2} + 2y(y_2 - y_1) + y_1^{\ 2} - y_2^{\ 2} \quad (4)$$

$(2) - (3)$

$$r_2^{\ 2} - r_3^{\ 2} = (x - x_2)^2 - (x - x_3)^2 + (y - y_2)^2 - (y - y_3)^2$$
$$r_2^{\ 2} - r_3^{\ 2} = (x - x_2 + x - x_3)(x - x_2 - (x - x_3)) + (y - y_2 + y - y_3)(y - y_2 - (y - y_3))$$
$$r_2^{\ 2} - r_3^{\ 2} = (2x - x_2 - x_3)(x_3 - x_2)) + (2y - y_2 - y_3)(y_3 - y_2))$$

$$r_2^2 - r_3^2 = 2x(x_3 - x_2) + (-(x_2 + x_3)(-(-x_3 + x_2)) + 2y(y_3 - y_2) + (-(y_2 + y_3)(-(-y_3 + y_2)))$$
$$r_2^2 - r_3^2 = 2x(x_3 - x_2) + (x_2 + x_3)(x_2 - x_3) + 2y(y_3 - y_2) + (y_2 + y_3)(y_2 - y_3)$$
$$r_2^2 - r_3^2 = 2x(x_3 - x_2) + x_2^2 - x_3^2 + 2y(y_3 - y_2) + y_2^2 - y_3^2 \quad (5)$$

Por ende las ecuaciones resultantes (4) y (5) son de primer orden, teniendo en cuenta que la Regla de Crammer permite obtener soluciones de sistemas de ecuaciones de primer orden 2x2 a través de determinantes, como se observa a continuación:

$$\begin{cases} ax + by = e \\ cx + dy = f \end{cases} \qquad \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} e \\ f \end{bmatrix}$$

$$x = \frac{\begin{vmatrix} e & b \\ f & d \end{vmatrix}}{\begin{vmatrix} a & b \\ c & d \end{vmatrix}} = \frac{ed - bf}{ad - bc}, \qquad y = \frac{\begin{vmatrix} a & e \\ c & f \end{vmatrix}}{\begin{vmatrix} a & b \\ c & d \end{vmatrix}} = \frac{af - ec}{ad - bc}$$

Por lo tanto, de acuerdo a las ecuaciones (4) y (5):

$$2x(x_2 - x_1) + 2y(y_2 - y_1) = r_1^2 - r_2^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2 \quad (4)$$
$$2x(x_3 - x_2) + 2y(y_3 - y_2) = r_2^2 - r_3^2 - x_2^2 + x_3^2 - y_2^2 + y_3^2 \quad (5)$$

Por tanto las variables a, b, c, d, e, f corresponden a:

$$a = 2(x_2 - x_1)$$
$$b = 2(y_2 - y_1)$$
$$c = 2(x_3 - x_2)$$
$$d = 2(y_3 - y_2)$$
$$e = r_1^2 - r_2^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2$$
$$f = r_2^2 - r_3^2 - x_2^2 + x_3^2 - y_2^2 + y_3^2$$

De esta forma la posición del vehículo P(x,y) corresponde a:

$$x = \frac{e(d) - b(f)}{a(d) - b(c)}$$
$$y = \frac{a(f) - e(c)}{a(d) - b(c)}$$

2.2 Algoritmo 2

Este algoritmo permite determinar si el vehículo se encuentra en peligro o seguro, de acuerdo a un mensaje que es enviado a través un array de strings, al almacenarlo en una matriz de caracteres es posible realizar recorridos de manera: horizontal, vertical y diagonal, con el fin

de localizar secuencias de 4 letras iguales, para el desarrollo de este algoritmo se realizó el siguiente análisis:

➔ Recorrido horizontal:

A	T	G	C	G	A
C	A	G	T	G	C
T	T	A	T	G	T
A	G	A	A	G	G
C	C	C	C	T	A
T	C	A	C	T	G

A	T	G	C	G	A
C	A	G	T	G	C
T	T	A	T	G	T
A	G	A	A	G	G
D	C	C	C	C	A
T	C	A	C	T	G

A	T	G	C	G	A
C	A	G	T	G	C
T	T	A	T	G	T
A	G	A	A	G	G
D	T	C	C	C	C
T	C	A	C	T	G

Como se puede observar ya que el tamaño de la matriz es de 6x6, la secuencia de 4 caracteres se puede encontrar de 3 maneras distintas en una fila. Por ello es necesario realizar la comparación de los caracteres en estas posiciones teniendo en cuenta los 3 casos.

➔ Recorrido vertical:

A	T	G	C	G	A
C	A	G	T	G	C
T	T	A	T	G	T
A	G	A	A	G	G
C	C	C	C	T	A
T	C	A	C	T	G

A	T	G	C	A	A
C	A	G	T	G	C
T	T	A	T	G	T
A	G	A	A	G	G
D	C	C	C	G	A
T	C	A	C	T	G

A	T	G	C	A	A
C	A	G	T	A	C
T	T	A	T	G	T
A	G	A	A	G	G
D	T	C	C	G	C
T	C	A	C	G	G

De manera similar al recorrido horizontal, las secuencias de caracteres se pueden encontrar de 3 maneras en una columna, por ello también es necesario tener en cuenta todos los casos para las comparaciones.

➔ Recorrido diagonal:

A	G	T	C	G	A
C	A	G	T	G	C
T	C	A	G	T	T
A	T	C	A	G	T
C	C	T	C	T	A
T	C	A	T	T	G

C	T	G	C	A	A
D	A	G	T	G	C
T	C	A	G	G	T
A	G	C	A	G	G
D	C	C	C	A	G
T	C	A	C	C	G

C	T	G	C	A	A
C	C	G	T	A	C
T	T	A	T	G	T
A	G	A	A	G	G
D	T	C	C	A	C
T	C	A	C	G	A

Para obtener estas diagonales basta con observar la posición inicial de las primeras 3 filas y a partir de cada una de ellas ir aumentando la posición de la columna, como se observa a continuación:

i=0 j=0	i=0 j=1	i=0 j=2			
A	G	T	C	G	A
C	A	G	T	G	C
T	C	A	G	T	T
A	T	C	A	G	T
C	C	T	C	T	A
T	C	A	T	T	G

	i=1 j=1	i=1 j=2				
	A	T	T	C	G	A
i=1 j=0	C	A	G	T	G	C
	T	C	A	G	T	T
	A	T	C	A	G	T
	C	C	T	C	T	G
	T	C	A	T	T	G

	i=2 j=1	i=2 j=2				
	C	T	T	C	G	A
	C	C	G	T	G	C
i=2 j=0	T	C	A	G	T	T
	A	T	C	A	G	T
	C	C	T	C	A	G
	T	C	A	T	C	A

Y para el recorrido de derecha a izquierda, ya que se trata de una matriz cuadrada, es posible hallar su matriz espejo, en donde se obtendrá la matriz observada anteriormente, de esta forma se le puede aplicar el mismo algoritmo.

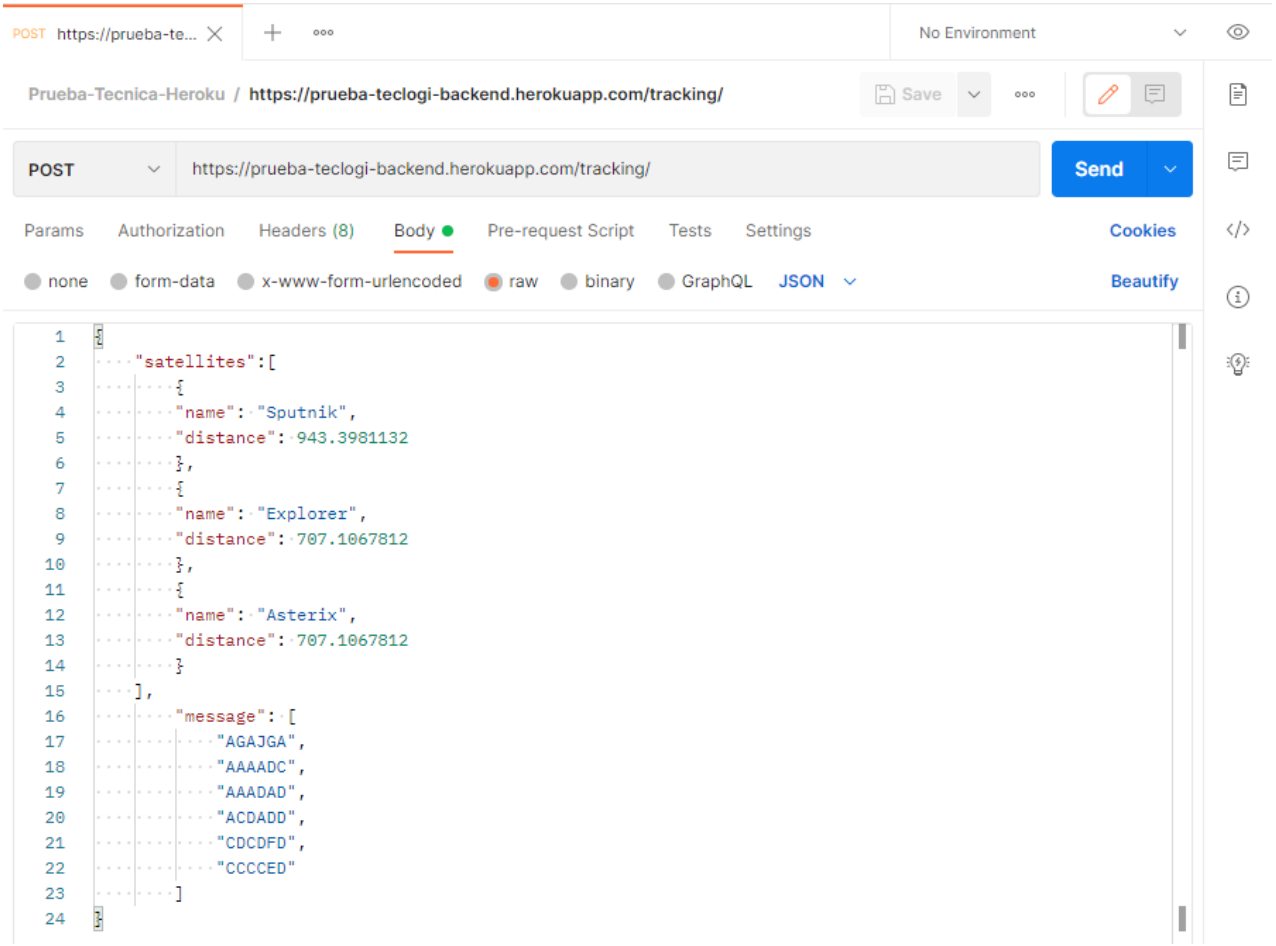
2.3 Deployment

El servicio /tracking/ construido en una API a través del framework Spring boot, que permite enviar la petición HTTP POST con el correspondiente payload, se encuentra disponible en el cloud Heroku a través de la siguiente URL:

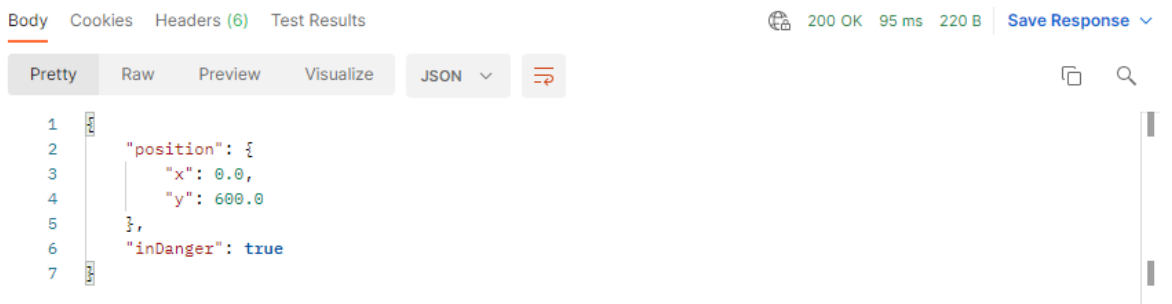
https://prueba-teclogi-backend.herokuapp.com/tracking/

3. Ejecución del programa

Para la ejecución del programa se realizará la correspondiente petición HTTP a través de Postman (programa que permite realizar peticiones a APIs), accediendo a la URL mencionada anteriormente, seleccionando el método POST, Body y raw, de esta forma aparecerá una sección para añadir la información del payload: las distancias de cada satélite, indicando su nombre y el mensaje codificado, como se observa a continuación:



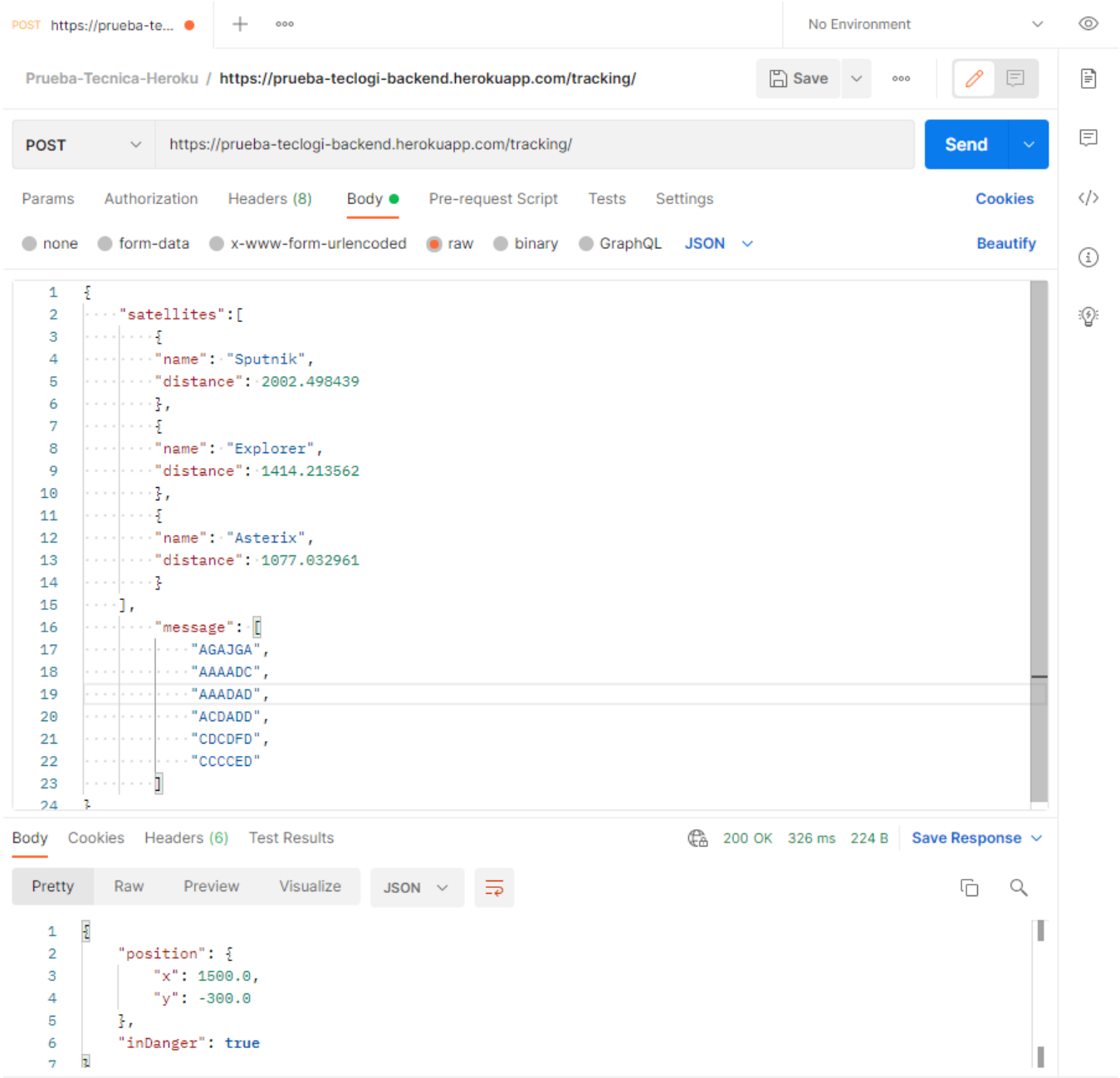
Al dar clic en el botón Send, se obtendrá la respuesta como se observa a continuación:



3.1 Casos de prueba

3.1.1 Parte 1

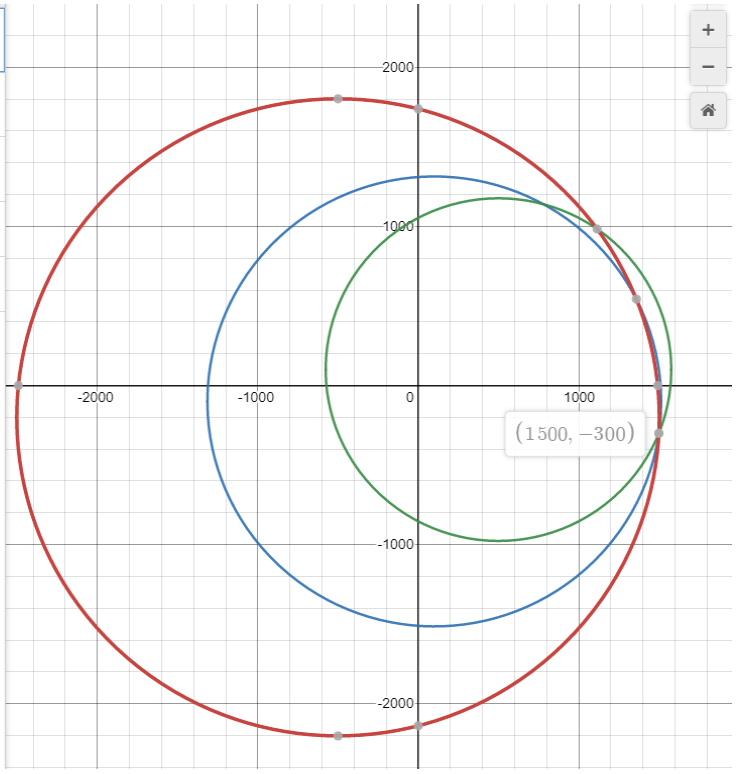
Caso #1:



$$2002.498439^2 = (x+500)^2 + (y+200)^2$$

$$1414.213562^2 = (x-100)^2 + (y+100)^2$$

$$1077.032961^2 = (x-500)^2 + (y-100)^2$$



POST https://prueba-te...

No Environment

Prueba-Tecnica-Heroku / https://prueba-teclogi-backend.herokuapp.com/tracking/

Save

POST https://prueba-teclogi-backend.herokuapp.com/tracking/

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

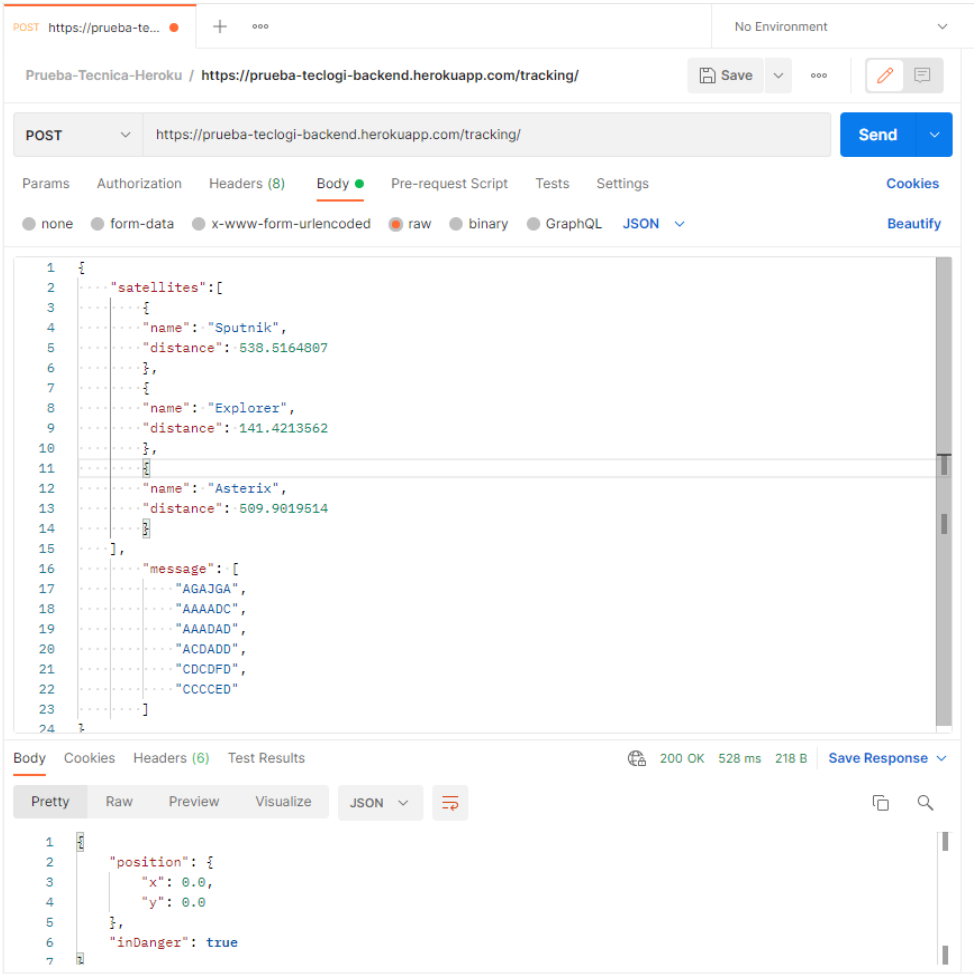
```
{
    "satellites":[
        {
        "name": "Asterix",
        "distance": 1077.032961
        },
        {
        "name": "Explorer",
        "distance": 1414.213562
        },
        {
        "name": "Sputnik",
        "distance": 2002.498439
        }
    ],
        "message": [
            "AGAJGA",
            "AAAADC",
            "AAADAD",
            "ACDADD",
            "CDCDFD",
            "CCCCED"
        ]
}
```

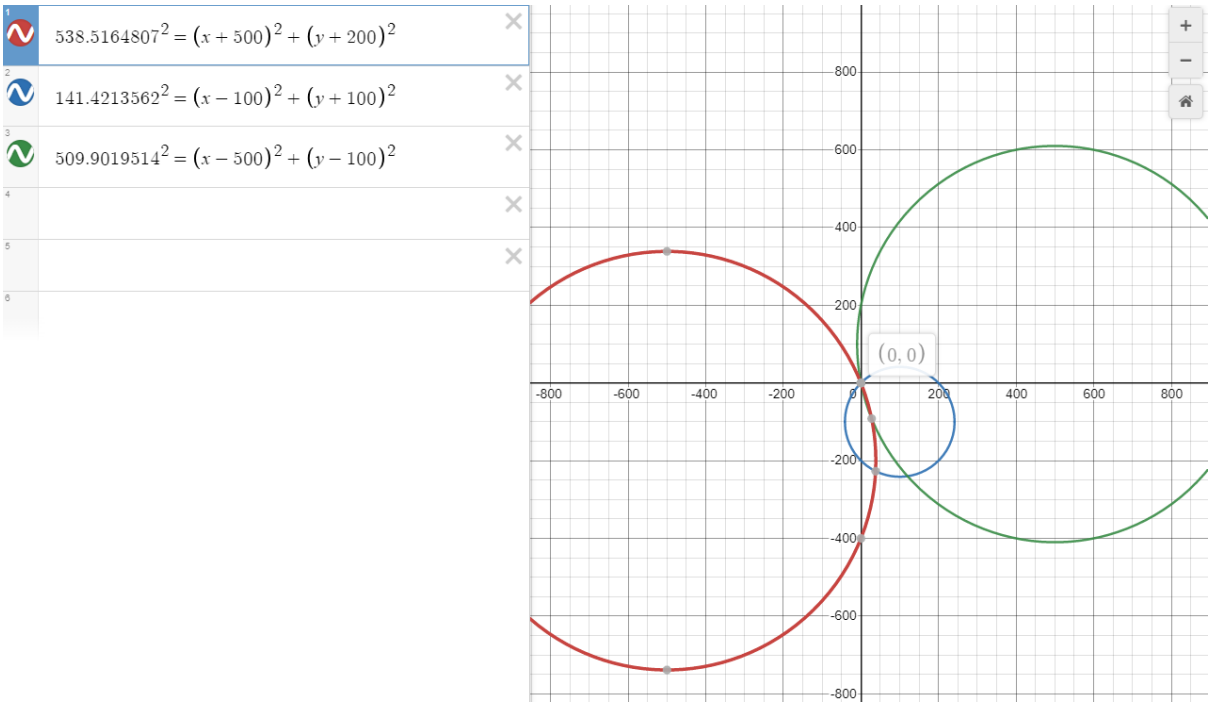
Body Cookies Headers (6) Test Results 200 OK 387 ms 224 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "position": {
        "x": 1500.0,
        "y": -300.0
    },
    "inDanger": true
}
```

Caso #2:


POST https://prueba-te...
No Environment
Prueba-Tecnica-Heroku / https://prueba-teclogi-backend.herokuapp.com/tracking/
Save
POST
https://prueba-teclogi-backend.herokuapp.com/tracking/
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"satellites":[
{
"name": "Sputnik",
"distance": 538.5164807
},
{
"name": "Explorer",
"distance": 141.4213562
},
{
"name": "Asterix",
"distance": 509.9019514
}
],
"message": [
"AGAJGA",
"AAAADC",
"AAADAD",
"ACDADD",
"CDCDFD",
"CCCCED"
]
}
Body
Cookies
Headers (6)
Test Results
200 OK
528 ms
218 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"position": {
"x": 0.0,
"y": 0.0
},
"inDanger": true
}


$538.5164807^2 = (x+500)^2 + (y+200)^2$
$141.4213562^2 = (x-100)^2 + (y+100)^2$
$509.9019514^2 = (x-500)^2 + (y-100)^2$
(0, 0)
800
600
400
200
-200
-400
-600
-800
-800
-600
-400
-200
200
400
600
800

Caso #3:

POST https://prueba-te...

No Environment

Prueba-Tecnica-Heroku / https://prueba-teclogi-backend.herokuapp.com/tracking/

Save

POST https://prueba-teclogi-backend.herokuapp.com/tracking/

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "satellites":[
        {
        "name": "Sputnik",
        "distance": 1298.371895
        },
        {
        "name": "Explorer",
        "distance": 1165.292547
        },
        {
        "name": "Asterix",
        "distance": 1131.872507
        }
    ],
        "message": [
            "AGAJGA",
            "AAAADC",
            "AAADAD",
            "ACDADD",
            "CDCDFD",
            "CCCCED"
        ]
}
```

Body Cookies Headers (6) Test Results 200 OK 103 ms 224 B Save Response

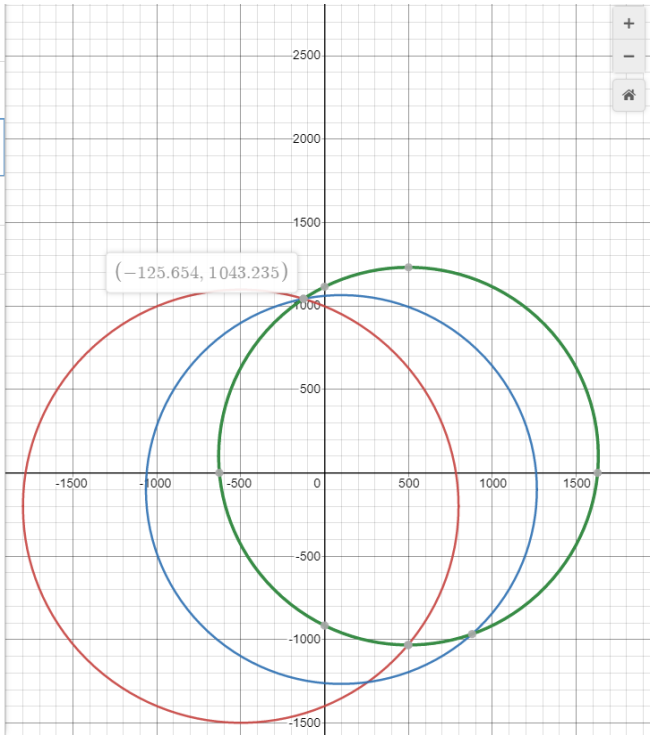
Pretty Raw Preview Visualize JSON

```
{
    "position": {
        "x": -125.7,
        "y": 1043.2
    },
    "inDanger": true
}
```

$1298.371895^2 = (x+500)^2 + (y+200)^2$

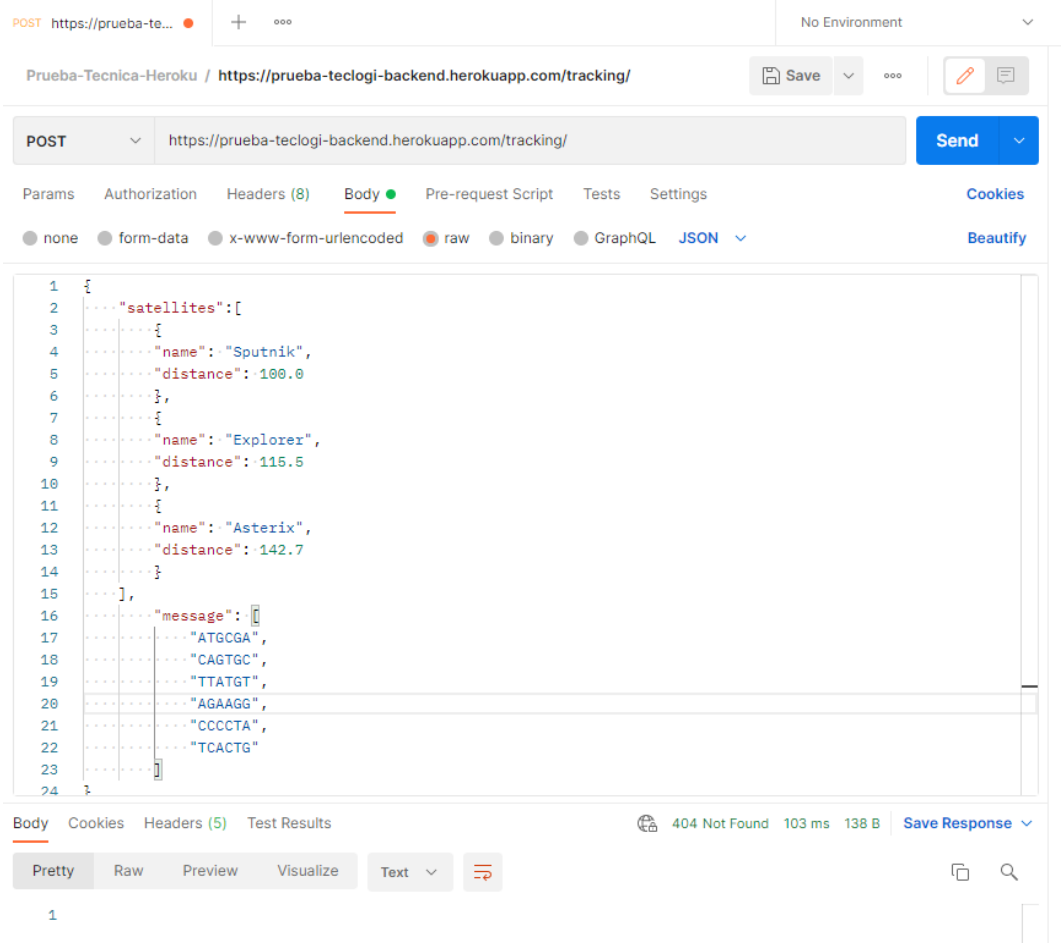
$1165.292547^2 = (x-100)^2 + (y+100)^2$

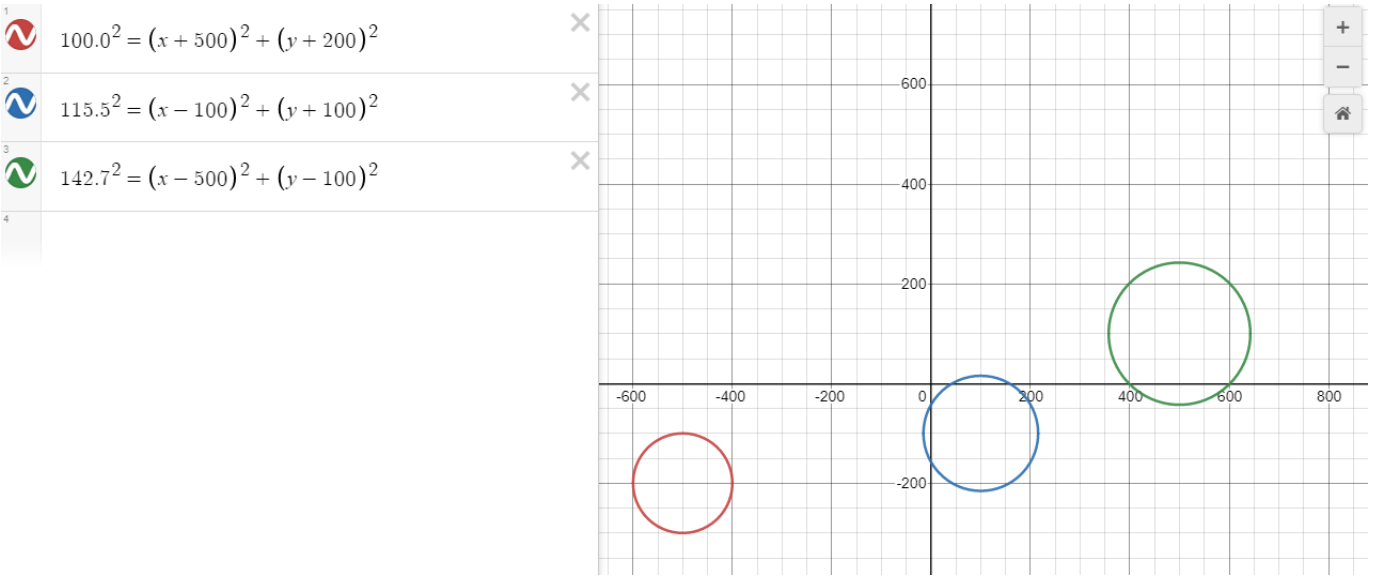
$1131.872507^2 = (x-500)^2 + (y-100)^2$



impulsado por desmos

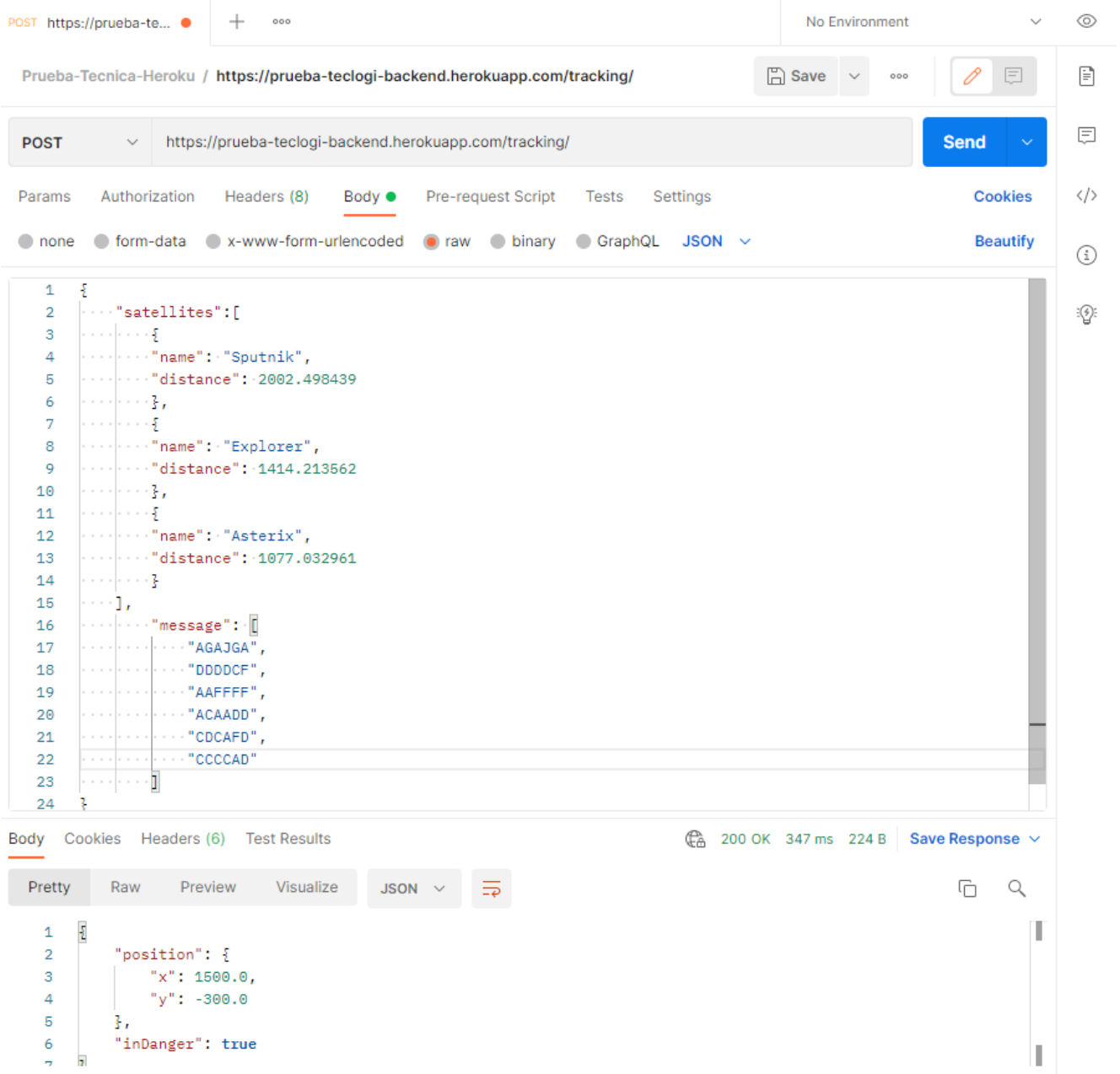
Caso #4:


POST https://prueba-te...
No Environment
Prueba-Tecnica-Heroku / https://prueba-teclogi-backend.herokuapp.com/tracking/
Save
POST
https://prueba-teclogi-backend.herokuapp.com/tracking/
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify
{
"satellites":[
{
"name": "Sputnik",
"distance": 100.0
},
{
"name": "Explorer",
"distance": 115.5
},
{
"name": "Asterix",
"distance": 142.7
}
],
"message": [
"ATGCGA",
"CAGTGC",
"TTATGT",
"AGAAGG",
"CCCCTA",
"TCACTG"
]
}
Body Cookies Headers (5) Test Results
404 Not Found 103 ms 138 B Save Response
Pretty Raw Preview Visualize Text
1

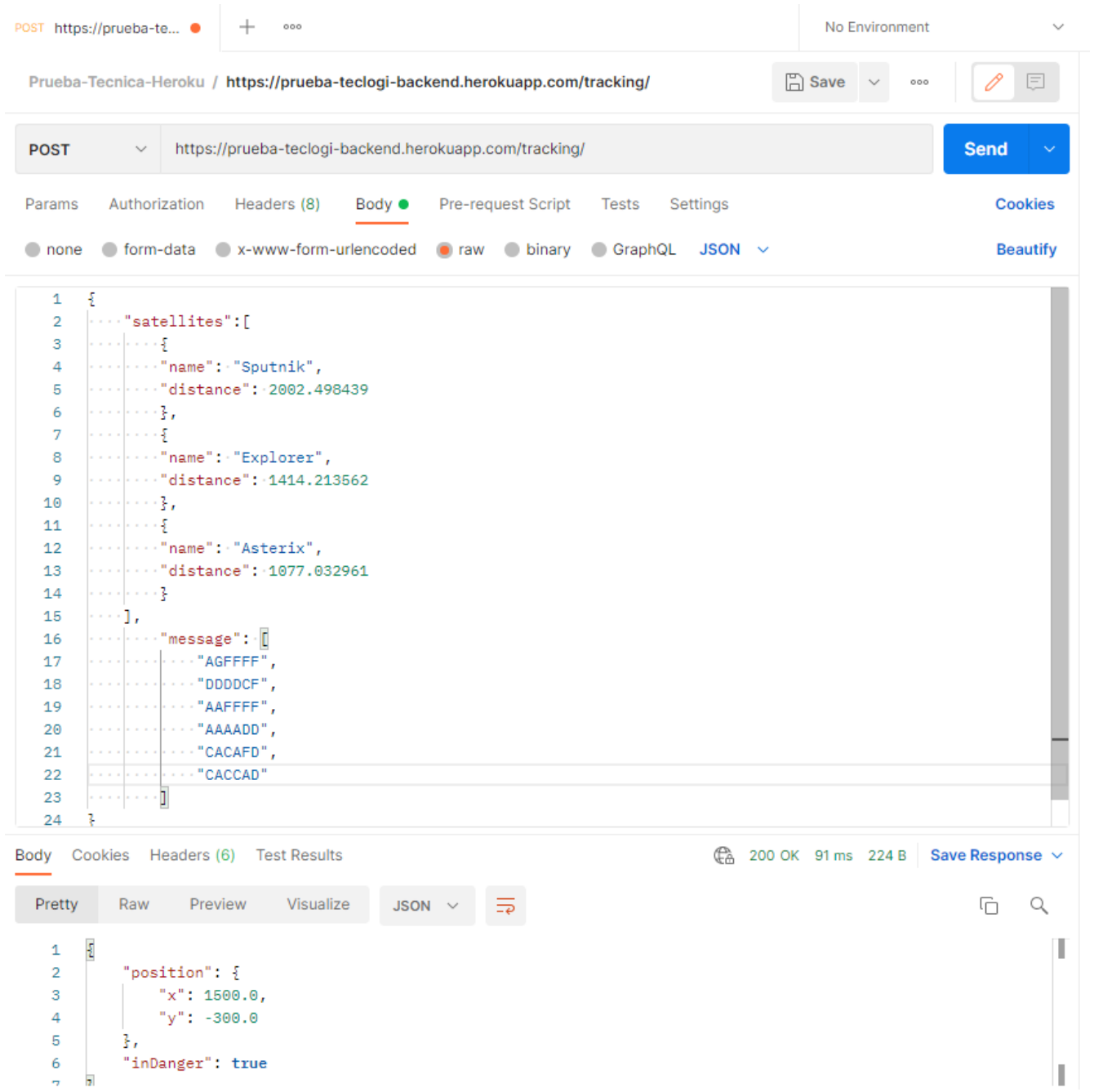

$100.0^2 = (x + 500)^2 + (y + 200)^2$
$115.5^2 = (x - 100)^2 + (y + 100)^2$
$142.7^2 = (x - 500)^2 + (y - 100)^2$
600
400
200
-200
-600
-400
-200
0
200
400
600
800

3.2.2 Parte 2

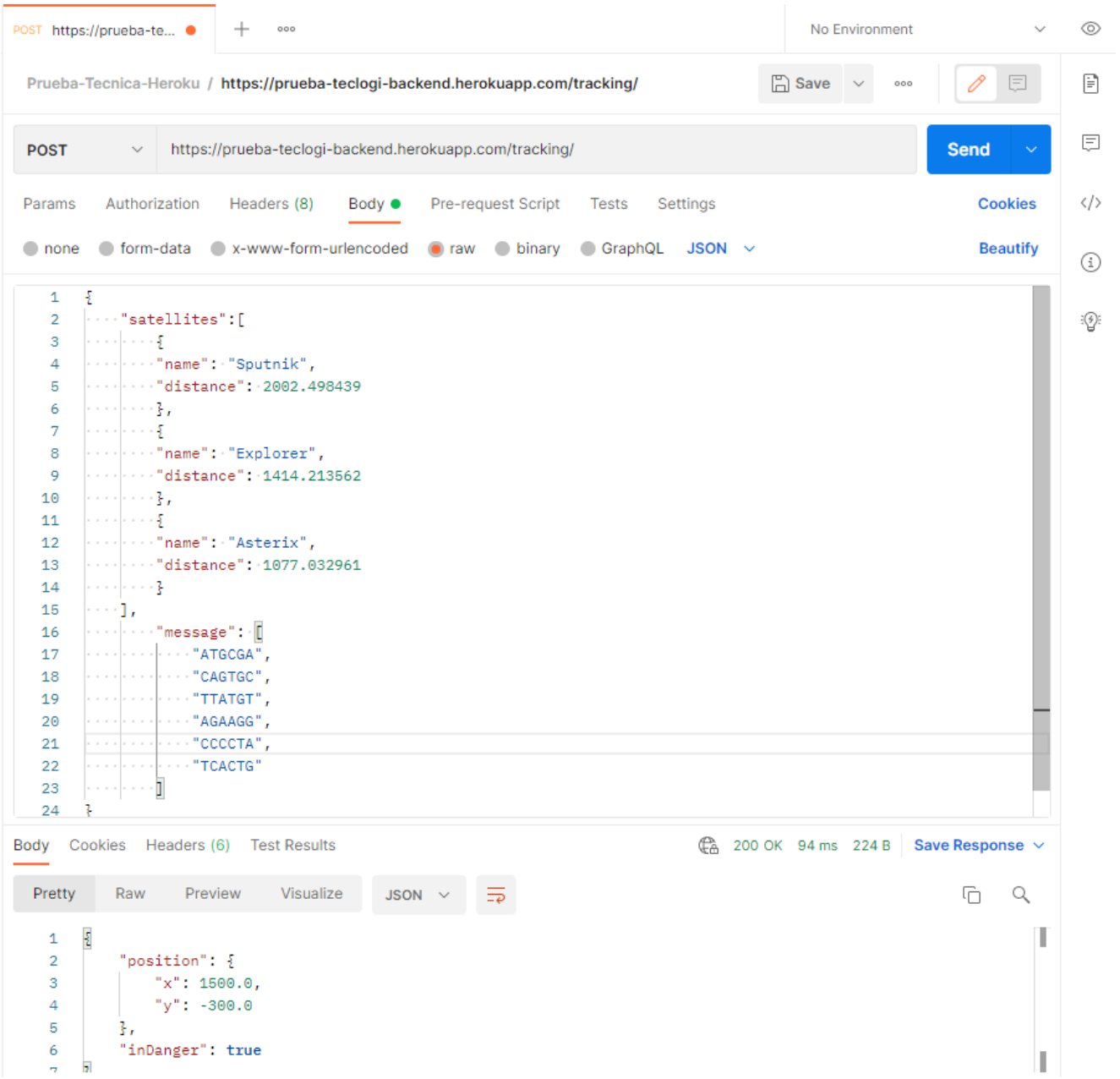
Caso #1



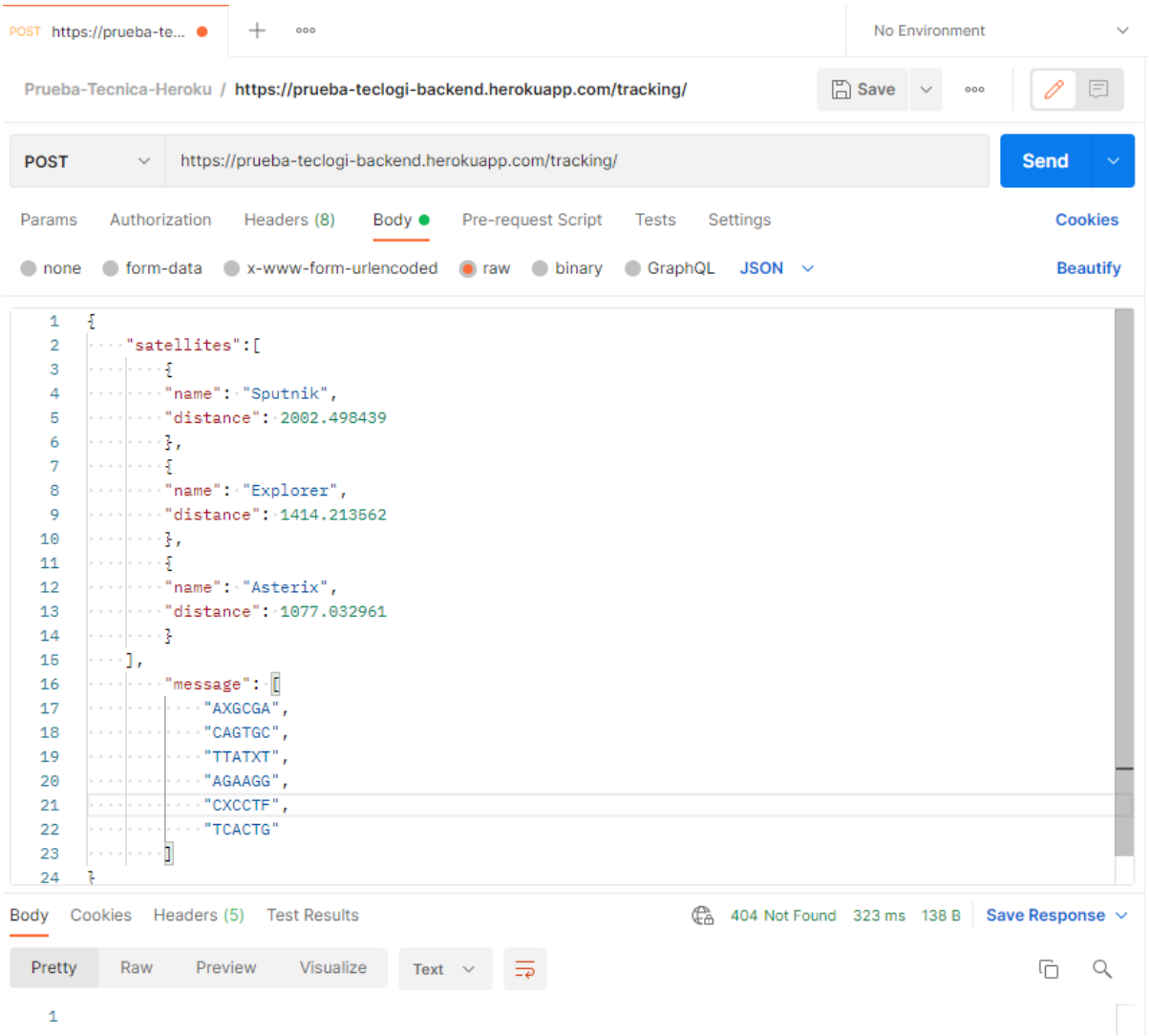
Caso #2


POST https://prueba-te...
No Environment
Prueba-Tecnica-Heroku / https://prueba-teclogi-backend.herokuapp.com/tracking/
Save
POST
https://prueba-teclogi-backend.herokuapp.com/tracking/
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"satellites":[
{
"name": "Sputnik",
"distance": 2002.498439
},
{
"name": "Explorer",
"distance": 1414.213562
},
{
"name": "Asterix",
"distance": 1077.032961
}
],
"message": [
"AGFFFF",
"DDDDCF",
"AAFFFF",
"AAAADD",
"CACAFD",
"CACCAD"
]
}
Body
Cookies
Headers (6)
Test Results
200 OK
91 ms
224 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"position": {
"x": 1500.0,
"y": -300.0
},
"inDanger": true
}

Caso #3


POST https://prueba-te...
No Environment
Prueba-Tecnica-Heroku / https://prueba-teclogi-backend.herokuapp.com/tracking/
Save
POST
https://prueba-teclogi-backend.herokuapp.com/tracking/
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"satellites":[
{
"name": "Sputnik",
"distance": 2002.498439
},
{
"name": "Explorer",
"distance": 1414.213562
},
{
"name": "Asterix",
"distance": 1077.032961
}
],
"message": [
"ATGCGA",
"CAGTGC",
"TTATGT",
"AGAAGG",
"CCCCTA",
"TCACTG"
]
}
Body
Cookies
Headers (6)
Test Results
200 OK
94 ms
224 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"position": {
"x": 1500.0,
"y": -300.0
},
"inDanger": true
}

Caso #4



4. Repositorio GitHub

El repositorio con el código fuente de la Api se encuentra disponible en la siguiente URL: https://github.com/lsofiadb/Prueba-tecnica-Backend