Laboratorio 4 – Inteligencia de Negocios – Grupo 15

Integrantes:

Andrés Felipe Pinzón

Santiago Forero

Juan José Rodríguez

Para el desarrollo de este documento se realizan 5 casos, distribuidos en 3 exitosos y 2 de falla, para los cuales se explica y se especifican la información de los resultados y de su posible falla. Son 5 casos para el primer punto y 5 casos para el segundo punto.

Modelamiento:

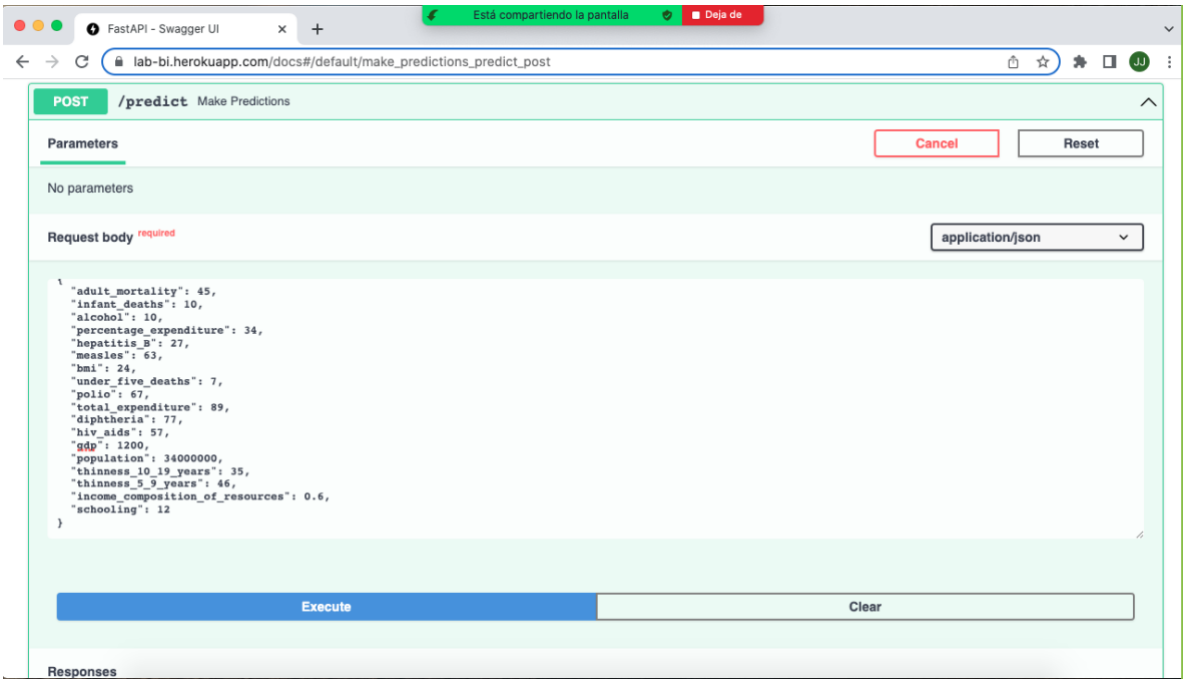
Los escenarios son los siguientes para ambas URLs:

1. Primer URL (POST /predict)

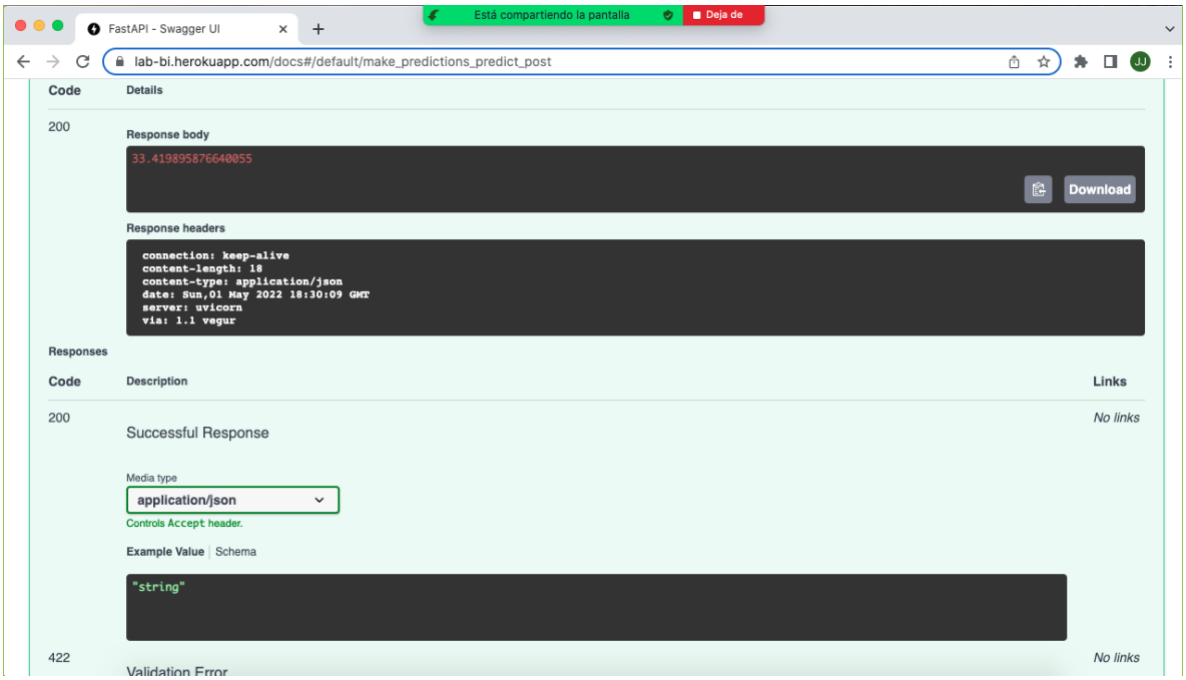
- Escenario 1:
  - Datos de entrada (formato JSON):

```
{
 "adult_mortality": 45,
 "infant_deaths": 10,
 "alcohol": 10,
 "percentage_expenditure": 34,
 "hepatitis_B": 27,
 "measles": 63,
 "bmi": 24,
 "under_five_deaths": 7,
 "polio": 67,
 "total_expenditure": 89,
 "diphtheria": 77,
 "hiv_aids": 57,
 "gdp": 1200,
 "population": 34000000,
 "thinness_10_19_years": 35,
 "thinness_5_9_years": 46,
 "income_composition_of_resources": 0.6,
 "schooling": 12
}
```

  - Resultado:
    33.419895876640055
  - Proceso realizado de Postman:
    1. Se colocan los valores de entrada y se ejecuta el programa:



2. Se dirige a la ventana de Responses, en Response body, del sistema y se verifica el resultado:

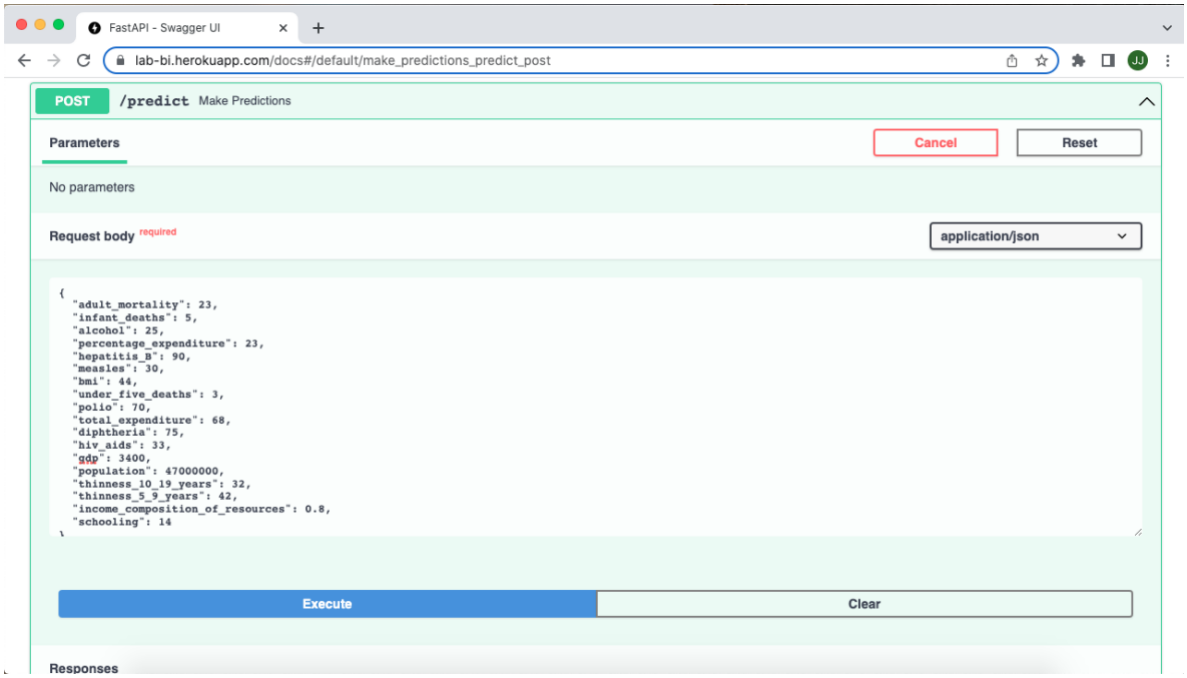


- Comentarios generales.
  El proceso tiene un tiempo de respuesta bastante rapido y no se toma mucho tiempo en el proceso de los datos. El resultado es el esperado, ya que se ponen valores de un indic de vida bastante corto. Por lo tanto, esta predicción que es baja demuestra un correcto entendimiento de los valores de entrada.
  No se encontraron ningún tipo de errores en este escenario.

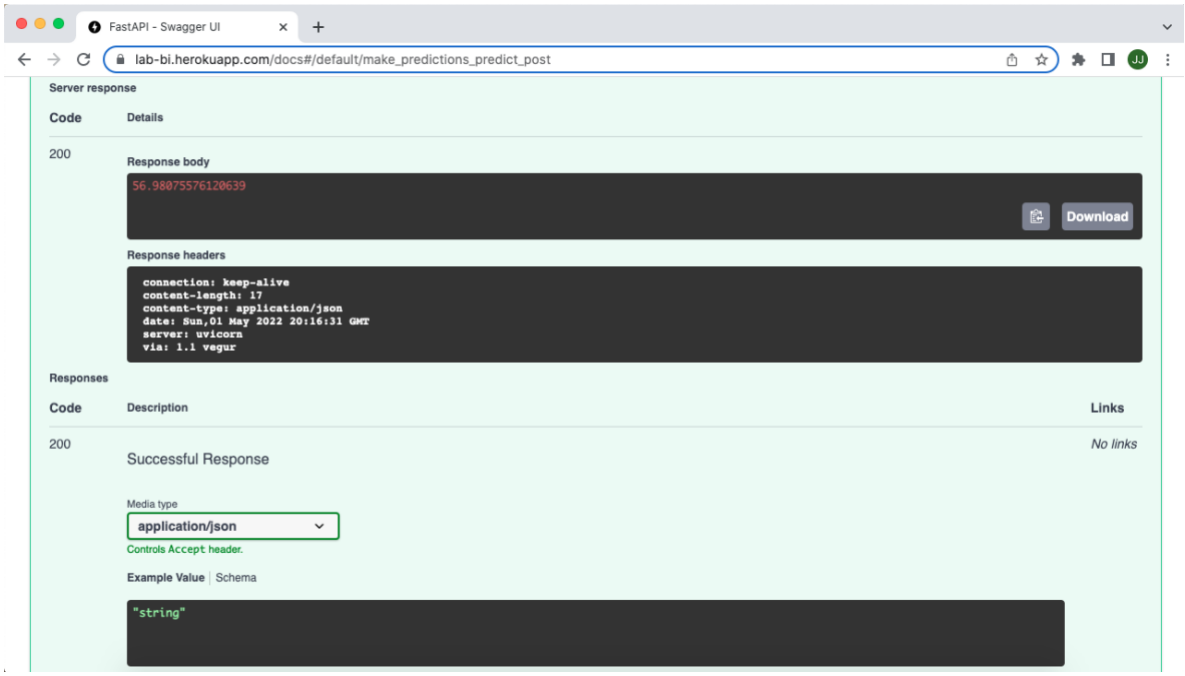
- Escenario 2:
  - Datos de entrada (formato JSON):

```
{
 "adult_mortality": 23,
 "infant_deaths": 5,
 "alcohol": 25,
 "percentage_expenditure": 23,
 "hepatitis_B": 90,
 "measles": 30,
 "bmi": 44,
 "under_five_deaths": 3,
 "polio": 70,
 "total_expenditure": 68,
 "diphtheria": 75,
 "hiv_aids": 33,
 "gdp": 3400,
 "population": 47000000,
 "thinness_10_19_years": 32,
 "thinness_5_9_years": 42,
 "income_composition_of_resources": 0.8,
 "schooling": 14
}
```

  - Resultado:
    56.98075576120639
  - Proceso realizado de Postman:
    1. Se colocan los valores de entrada y se ejecuta el programa:



2. Se dirige a la ventana de Responses, en Response body, del sistema y se verifica el resultado:

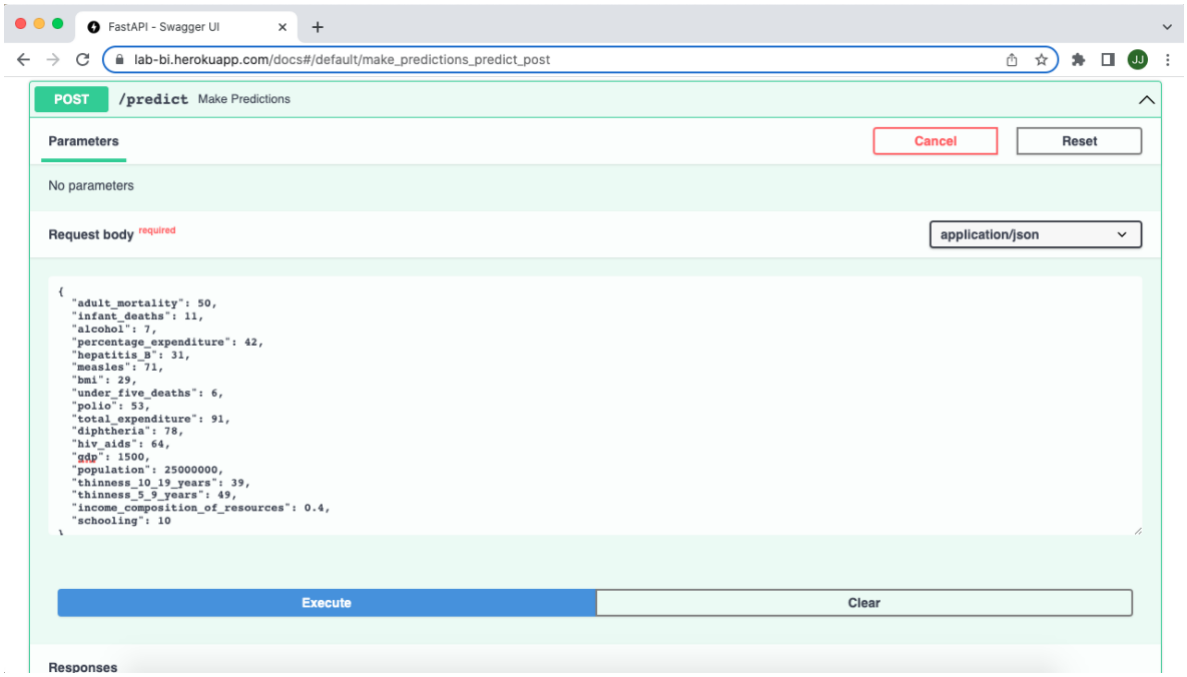


- Comentarios generales:
  En este caso, se optó por usar el caso de una población en buenas condiciones. Se tienen datos de un índice de vida mucho mayor y con menores niveles de enfermedades. Se obtiene una respuesta con un porcentaje más alto y se ve que se tiene una buena predicción por parte del modelo.
  No se encontraron errores en este punto.

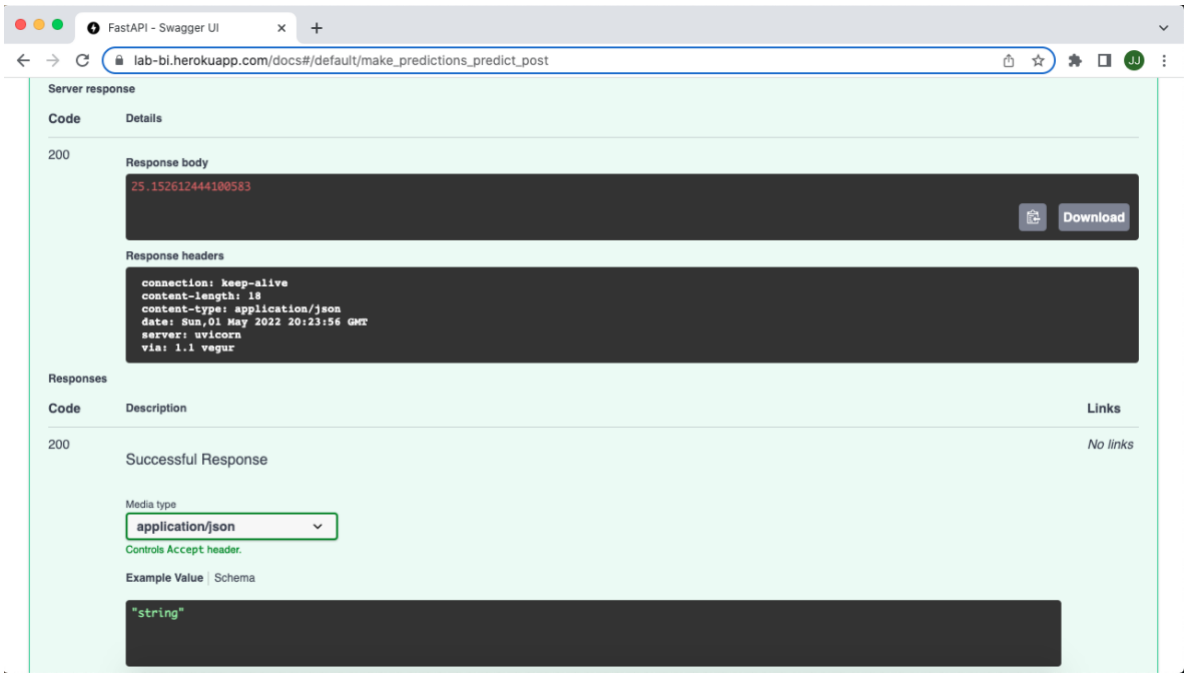
- Escenario 3:
  - Datos de entrada (formato JSON):
    {
    "adult_mortality": 50,
    "infant_deaths": 11,
    "alcohol": 7,
    "percentage_expenditure": 42,
    "hepatitis_B": 31,
    "measles": 71,
    "bmi": 29,
    "under_five_deaths": 6,
    "polio": 53,
    "total_expenditure": 91,
    "diphtheria": 78,
    "hiv_aids": 64,
    "gdp": 1500,
    "population": 25000000,
    "thinness_10_19_years": 39,
    "thinness_5_9_years": 49,

```
 "income_composition_of_resources": 0.4,
 "schooling": 10
}
```


- Resultado:
  25.152612444100583
- Proceso realizado de Postman:
  1. Se colocan los valores de entrada y se ejecuta el programa:



  2. Se dirige a la ventana de Responses, en Response body, del sistema y se verifica el resultado:



- Comentarios generales:

En este caso, se pone a prueba una población de bajos recursos y con un índice de vida bastante bajo. Se tienen mayores tasa de mortalidad y porcentaje de enfermedades, asi como bajos indices de recursos. En general, el programa logra predecir correctamente un porcentaje más bajo que en casos anteriores, lo que demuestra que los datos pueden afectar de manera efectiva el resultado que se obtiene.

No se encontraron errores en este punto.

- Escenario 4:
  - Datos de entrada (formato JSON):

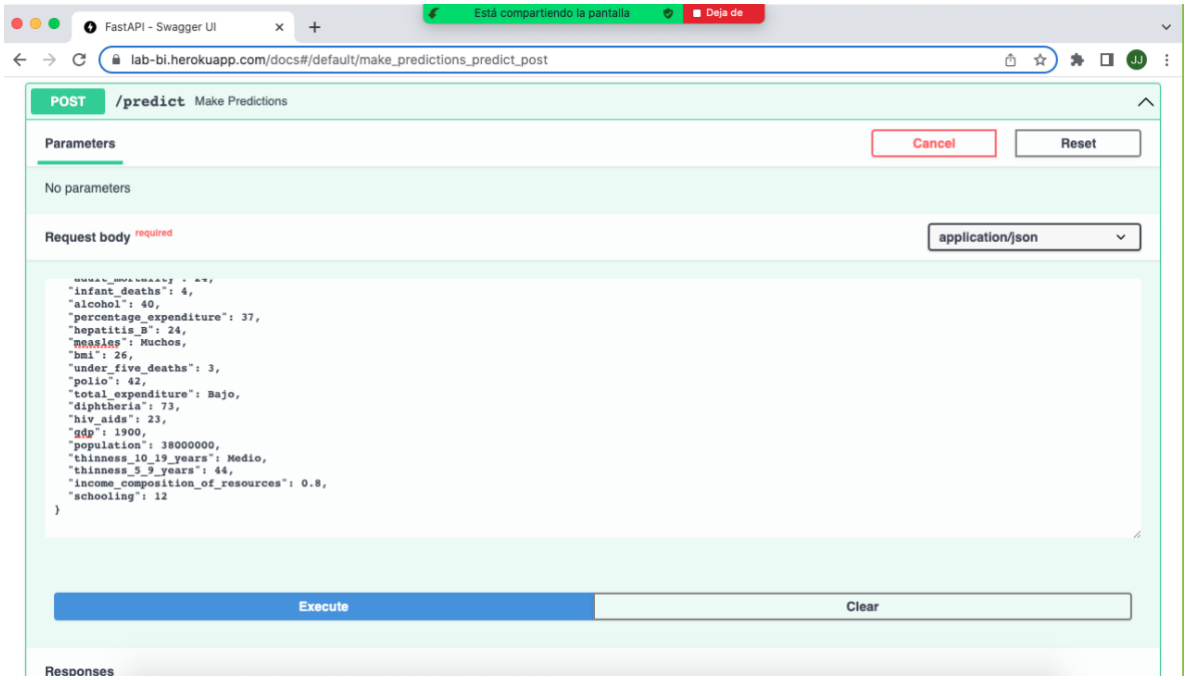
```
{
 "adult_mortality": 24,
 "infant_deaths": 4,
 "alcohol": 40,
 "percentage_expenditure": 37,
 "hepatitis_B": 24,
 "measles": Muchos,
 "bmi": 26,
 "under_five_deaths": 3,
 "polio": 42,
 "total_expenditure": Bajo,
 "diphtheria": 73,
 "hiv_aids": 23,
 "gdp": 1900,
 "population": 38000000,
 "thinness_10_19_years": Medio,
 "thinness_5_9_years": 44,
 "income_composition_of_resources": 0.8,
 "schooling": 12
}
```

  - Resultado:

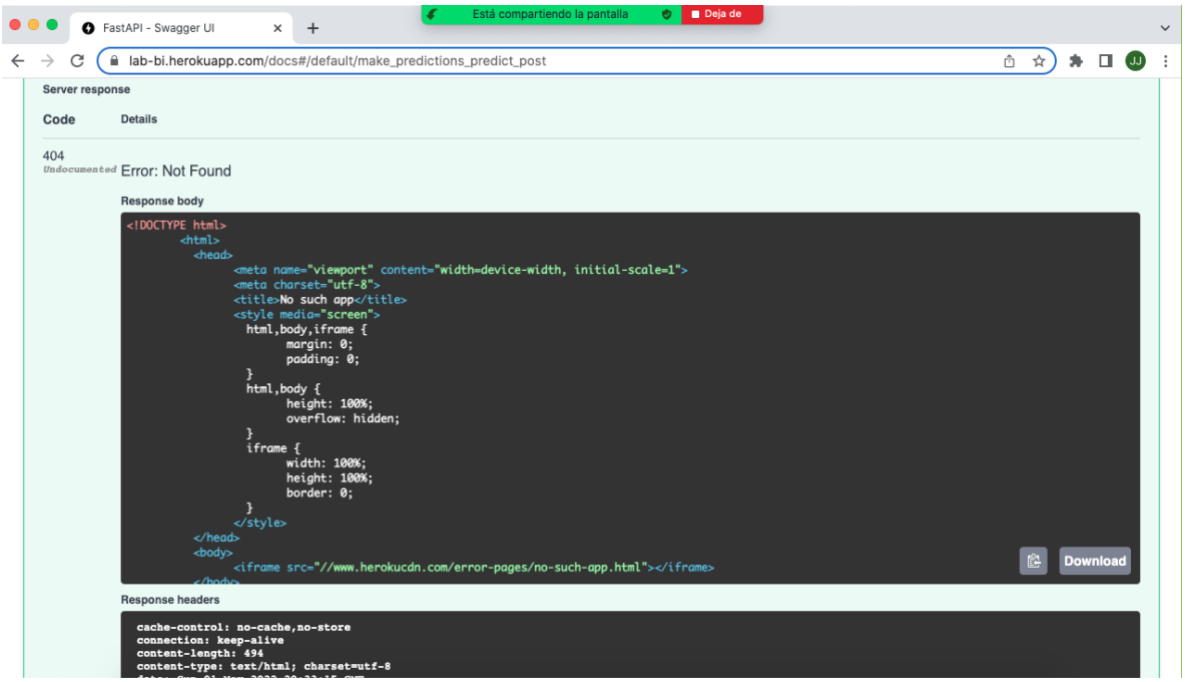
```
<!DOCTYPE html>
    <html>
     <head>
            <meta name="viewport" content="width=device-width, initial-scale=1">
            <meta charset="utf-8">
            <title>No such app</title>
            <style media="screen">
             html,body,iframe {
                    margin: 0;
                    padding: 0;
             }
             html,body {
                    height: 100%;
                    overflow: hidden;
```

```
            }
            iframe {
                width: 100%;
                height: 100%;
                border: 0;
            }
            </style>
        </head>
        <body>
            <iframe src="//www.herokucdn.com/error-pages/no-such-app.html"></iframe>
        </body>
    </html>
```

- Proceso realizado de Postman:
  1. Se colocan los valores de entrada y se ejecuta el programa:



  2. Se dirige a la ventana de Responses, en Response body, del sistema y se verifica el resultado:

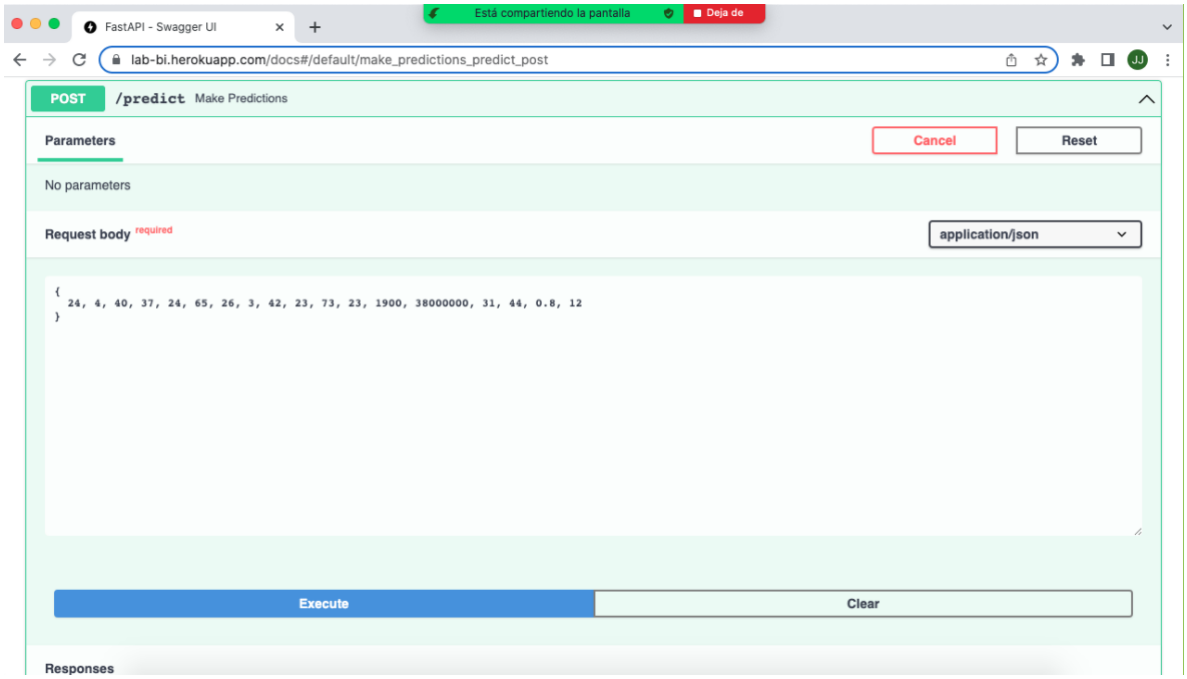


- Comentarios generales:
  Este es el primer caso que se prueba a error. Como se puede apreciar, en los datos de entrada se introducen valores en formato de String (letras y caracteres) y estos son validos como entrada para la ejecución del programa. Como se espera el programa no puede ejecutarse correctamente debido a los valores propuestos, y se muestra un error como respuesta. El programa identifica que el formato del dato no es un float y termina rápidamente la ejecución.

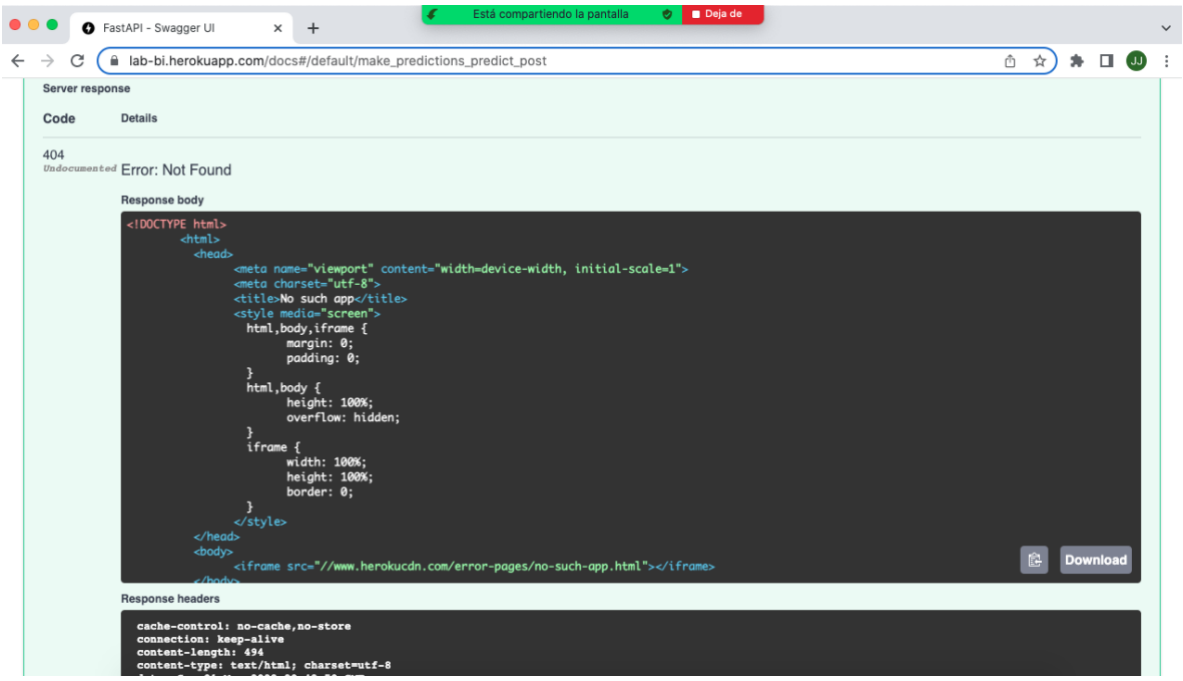
- Escenario 5:
  - Datos de entrada (formato JSON):
    {
     24, 4, 40, 37, 24, 65, 26, 3, 42, 23, 73, 23, 1900, 38000000, 31, 44, 0.8, 12
    }
  - Resultado:
    <!DOCTYPE html>
    <html>
    <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta charset="utf-8">
    <title>No such app</title>
    <style media="screen">
    html,body,iframe {
    margin: 0;
    padding: 0;
    }
    html,body {
    height: 100%;
    overflow: hidden;

```
            }
            iframe {
                width: 100%;
                height: 100%;
                border: 0;
            }
        </style>
    </head>
    <body>
        <iframe src="//www.herokucdn.com/error-pages/no-such-app.html"></iframe>
    </body>
</html>
```

- Proceso realizado de Postman:
  1. Se colocan los valores de entrada y se ejecuta el programa:



  2. Se dirige a la ventana de Responses, en Response body, del sistema y se verifica el resultado:



- Comentarios generales:
  En este caso, se muestra una segunda forma de error. Se edita la entrada del programa y este no es capaz de reconocerla correctamente. Se tiene los valores de entrada de forma númerica, pero no poseen los indicadores de que atributo representan. De esta forma, se puede apreciar que el programa no es capaz de asignar los valores respectivamente al orden de entrada o por separación de comas. Esto demuestra que es necesaria la implementación del formato JSON, para el correcto entendimiento de los datos por parte de la aplicación. Por lo tanto, el programa al no reconocer los valores de entrada, termina la ejecución.

2. Segundo URL (POST /predictionpunto2)

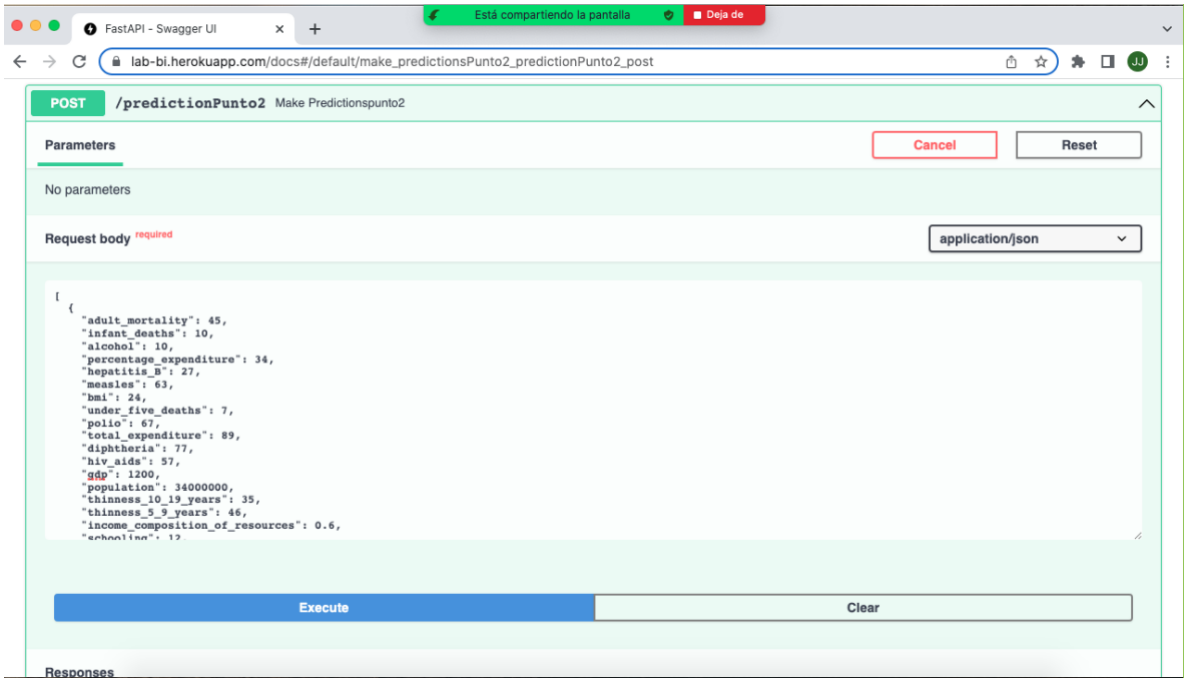
- Escenario 1:
  - Datos de entrada (formato JSON):

```
[
 {
  "adult_mortality": 45,
  "infant_deaths": 10,
  "alcohol": 10,
  "percentage_expenditure": 34,
  "hepatitis_B": 27,
  "measles": 63,
  "bmi": 24,
  "under_five_deaths": 7,
  "polio": 67,
  "total_expenditure": 89,
  "diphtheria": 77,
  "hiv_aids": 57,
```

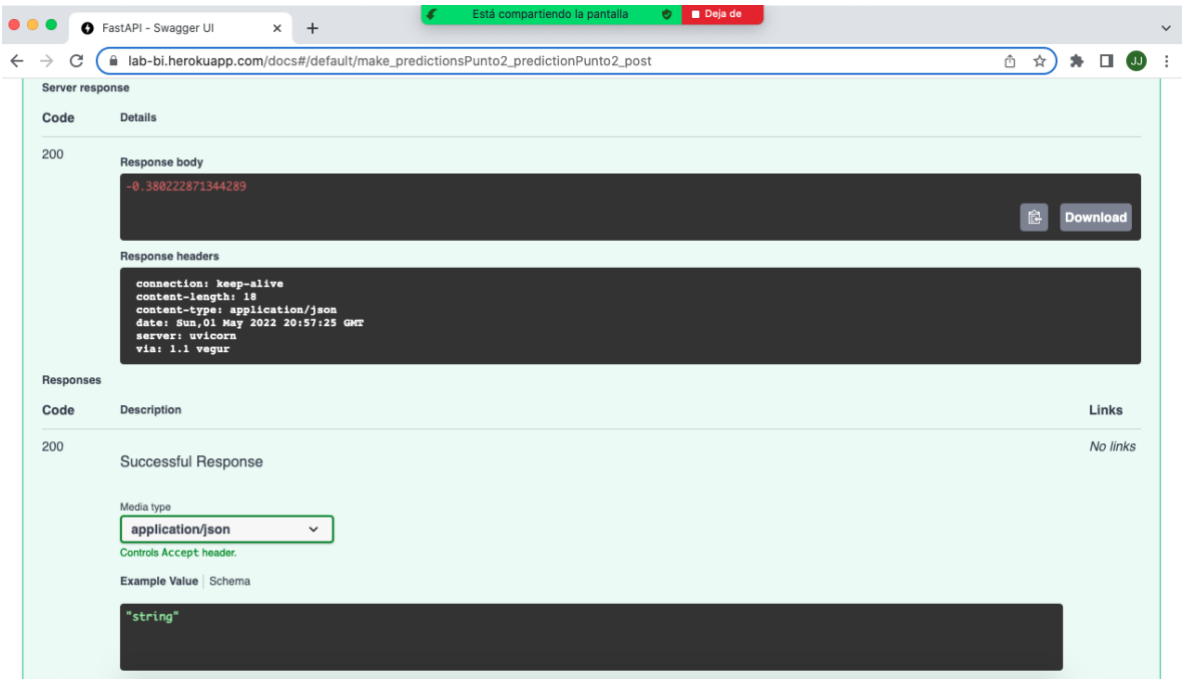
```
    "gdp": 1200,
    "population": 34000000,
    "thinness_10_19_years": 35,
    "thinness_5_9_years": 46,
    "income_composition_of_resources": 0.6,
    "schooling": 12,
    "Valor_Esperado": 40
  },
 {
    "adult_mortality": 42,
    "infant_deaths": 17,
    "alcohol": 9,
    "percentage_expenditure": 33,
    "hepatitis_B": 25,
    "measles": 60,
    "bmi": 26,
    "under_five_deaths": 8,
    "polio": 71,
    "total_expenditure": 84,
    "diphtheria": 71,
    "hiv_aids": 55,
    "gdp": 1400,
    "population": 32000000,
    "thinness_10_19_years": 32,
    "thinness_5_9_years": 44,
    "income_composition_of_resources": 0.7,
    "schooling": 11,
    "Valor_Esperado": 31
  },
 {
    "adult_mortality": 45,
    "infant_deaths": 10,
    "alcohol": 10,
    "percentage_expenditure": 34,
    "hepatitis_B": 27,
    "measles": 63,
    "bmi": 24,
    "under_five_deaths": 7,
    "polio": 67,
    "total_expenditure": 89,
    "diphtheria": 77,
    "hiv_aids": 57,
    "gdp": 1200,
    "population": 34000000,
```

```
  "thinness_10_19_years": 35,
  "thinness_5_9_years": 46,
  "income_composition_of_resources": 0.6,
  "schooling": 12,
  "Valor_Esperado": 30
 }
]
```


- Resultado:
  -0.380222871344289
- Proceso realizado de Postman:
  1. Se colocan los valores de entrada y se ejecuta el programa:



  2. Se dirige a la ventana de Responses, en Response body, del sistema y se verifica el resultado:



- Comentarios generales.
  En este caso, se ponen valores de entrada acordes al formato requerido y se espera un resultado en la ejecución. En este sentido, el programa responde con un valor negativo. Esto demuestra que el modelo tiene variables innecesarias o porbablemente los datos poseen un problema de colinealidad. Esto implica que no hay una buena relación entre los grupos de entrada marcados en la ejecución.

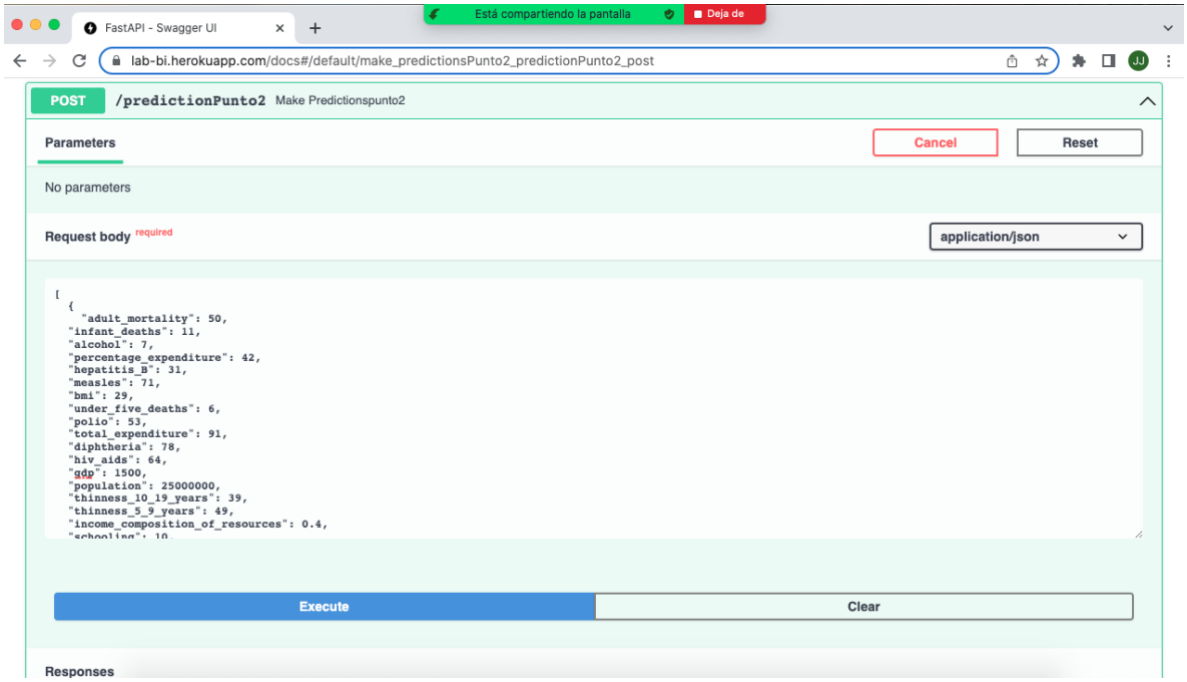
- Escenario 2:
  - Datos de entrada (formato JSON):

```
[
 {
  "adult_mortality": 50,
 "infant_deaths": 11,
 "alcohol": 7,
 "percentage_expenditure": 42,
 "hepatitis_B": 31,
 "measles": 71,
 "bmi": 29,
 "under_five_deaths": 6,
 "polio": 53,
 "total_expenditure": 91,
 "diphtheria": 78,
 "hiv_aids": 64,
 "gdp": 1500,
 "population": 25000000,
 "thinness_10_19_years": 39,
 "thinness_5_9_years": 49,
 "income_composition_of_resources": 0.4,
```

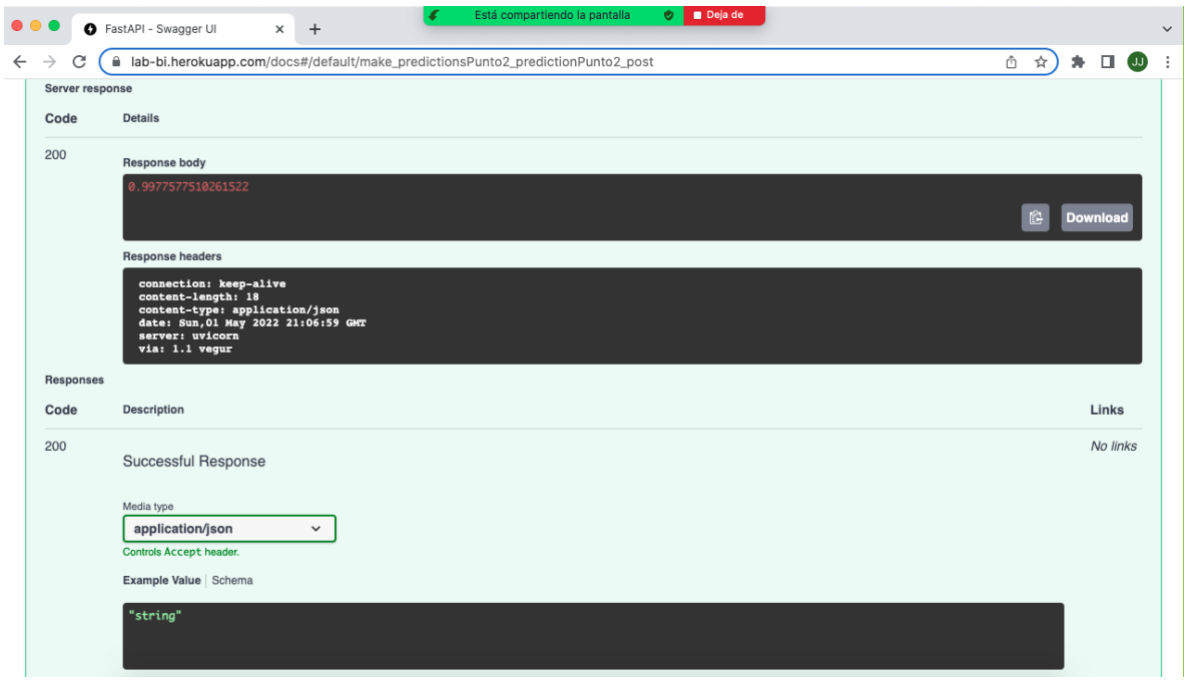
```
  "schooling": 10,
    "Valor_Esperado": 25
  },
{
    "adult_mortality": 23,
  "infant_deaths": 5,
  "alcohol": 25,
  "percentage_expenditure": 23,
  "hepatitis_B": 90,
  "measles": 30,
  "bmi": 44,
  "under_five_deaths": 3,
  "polio": 70,
  "total_expenditure": 68,
  "diphtheria": 75,
  "hiv_aids": 33,
  "gdp": 3400,
  "population": 47000000,
  "thinness_10_19_years": 32,
  "thinness_5_9_years": 42,
  "income_composition_of_resources": 0.8,
  "schooling": 14,
    "Valor_Esperado": 56
  },
{
    "adult_mortality": 45,
  "infant_deaths": 10,
  "alcohol": 10,
  "percentage_expenditure": 34,
  "hepatitis_B": 27,
  "measles": 63,
  "bmi": 24,
  "under_five_deaths": 7,
  "polio": 67,
  "total_expenditure": 89,
  "diphtheria": 77,
  "hiv_aids": 57,
  "gdp": 1200,
  "population": 34000000,
  "thinness_10_19_years": 35,
  "thinness_5_9_years": 46,
  "income_composition_of_resources": 0.6,
  "schooling": 12,
    "Valor_Esperado": 33
```

}
]

- Resultado:
  0.9977577510261522
- Proceso realizado de Postman:
  1. Se colocan los valores de entrada y se ejecuta el programa:



  2. Se dirige a la ventana de Responses, en Response body, del sistema y se verifica el resultado:



- Comentarios generales:

En este caso, se colocaron los tres modelos probados en el primer URL, con sus valores obtenidos. En este sentido, se espera que el programa entienda que se espera el valor correcto de estos modelos. Se evidencia un correcto funcionamiento del programa, ya que muestra un valor positivo con los datos de entrada. Esto demuestra un buen entedimiento del grupo de datos y de la predicción obtenida de los mismos. El valor positivo en el resultado, muestra que los modelos presentados tienen el valor esperado que se propuso.

- Escenario 3:
  - Datos de entrada (formato JSON):

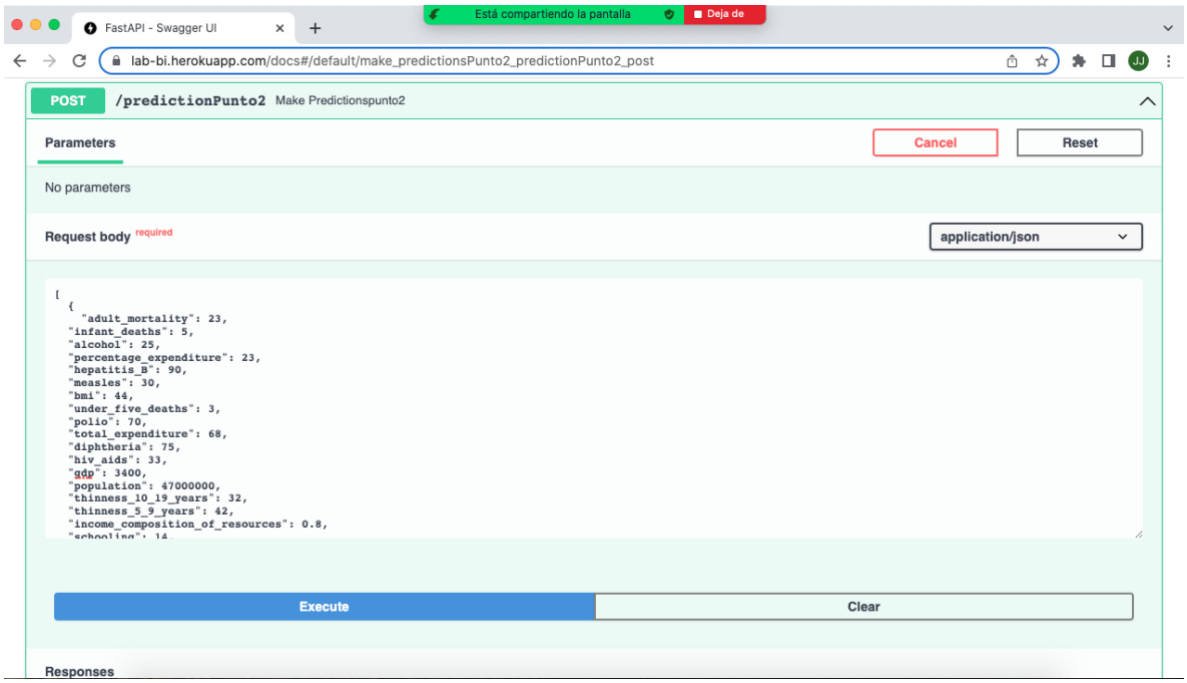

```
[
 {
  "adult_mortality": 23,
 "infant_deaths": 5,
 "alcohol": 25,
 "percentage_expenditure": 23,
 "hepatitis_B": 90,
 "measles": 30,
 "bmi": 44,
 "under_five_deaths": 3,
 "polio": 70,
 "total_expenditure": 68,
 "diphtheria": 75,
 "hiv_aids": 33,
 "gdp": 3400,
 "population": 47000000,
 "thinness_10_19_years": 32,
 "thinness_5_9_years": 42,
 "income_composition_of_resources": 0.8,
 "schooling": 14,
  "Valor_Esperado": 56
 },
{
  "adult_mortality": 45,
 "infant_deaths": 10,
 "alcohol": 10,
 "percentage_expenditure": 34,
 "hepatitis_B": 27,
 "measles": 63,
 "bmi": 24,
 "under_five_deaths": 7,
 "polio": 67,
 "total_expenditure": 89,
 "diphtheria": 77,
 "hiv_aids": 57,
```

```
 "gdp": 1200,
 "population": 34000000,
 "thinness_10_19_years": 35,
 "thinness_5_9_years": 46,
 "income_composition_of_resources": 0.6,
 "schooling": 12,
  "Valor_Esperado": 33
 }
]
```

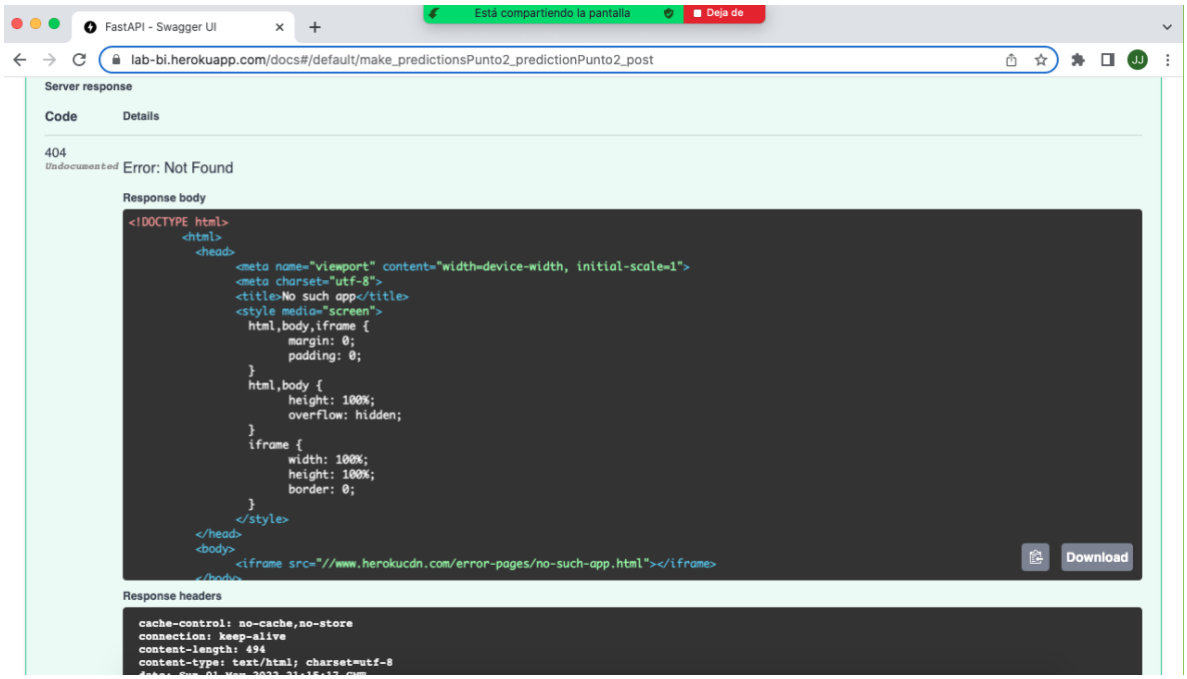

- Resultado:


```
<!DOCTYPE html>
    <html>
     <head>
            <meta name="viewport" content="width=device-width, initial-scale=1">
            <meta charset="utf-8">
            <title>No such app</title>
            <style media="screen">
             html,body,iframe {
                    margin: 0;
                    padding: 0;
             }
             html,body {
                    height: 100%;
                    overflow: hidden;
             }
             iframe {
                    width: 100%;
                    height: 100%;
                    border: 0;
             }
            </style>
     </head>
     <body>
            <iframe src="//www.herokucdn.com/error-pages/no-such-
app.html"></iframe>
     </body>
    </html>
```


- Proceso realizado de Postman:
  1. Se colocan los valores de entrada y se ejecuta el programa:



2. Se dirige a la ventana de Responses, en Response body, del sistema y se verifica el resultado:



- Comentarios generales:
  En este caso, se propone un caso de error en el cual no se presentan los casos suficientes para que el programa logre comparar los modelos. En este sentido, se espera que el programa no se ejecute correctamente. Por ende, se muestra un error en la ejecución, ya que no estan los modelos necesarios para la aplicación.

- Escenario 4:
  - Datos de entrada (formato JSON):
    [

```
 {
   "adult_mortality": 50,
  "infant_deaths": 11,
  "alcohol": 7,
  "percentage_expenditure": Medio,
  "hepatitis_B": 31,
  "measles": 71,
  "bmi": 29,
  "under_five_deaths": Baja,
  "polio": 53,
  "total_expenditure": Alto,
  "diphtheria": 78,
  "hiv_aids": 64,
  "gdp": 1500,
  "population": 25000000,
  "thinness_10_19_years": 39,
  "thinness_5_9_years": 49,
  "income_composition_of_resources": Poco,
  "schooling": 10,
    "Valor_Esperado": 25
 },
{
    "adult_mortality": 23,
  "infant_deaths": 5,
  "alcohol": 25,
  "percentage_expenditure": 23,
  "hepatitis_B": Alto,
  "measles": 30,
  "bmi": 44,
  "under_five_deaths": No,
  "polio": 70,
  "total_expenditure": 68,
  "diphtheria": 75,
  "hiv_aids": Medio,
  "gdp": 3400,
  "population": 47000000,
  "thinness_10_19_years": 32,
  "thinness_5_9_years": 42,
  "income_composition_of_resources": 0.8,
  "schooling": 14,
    "Valor_Esperado": 56
 },
{
    "adult_mortality": 45,
```

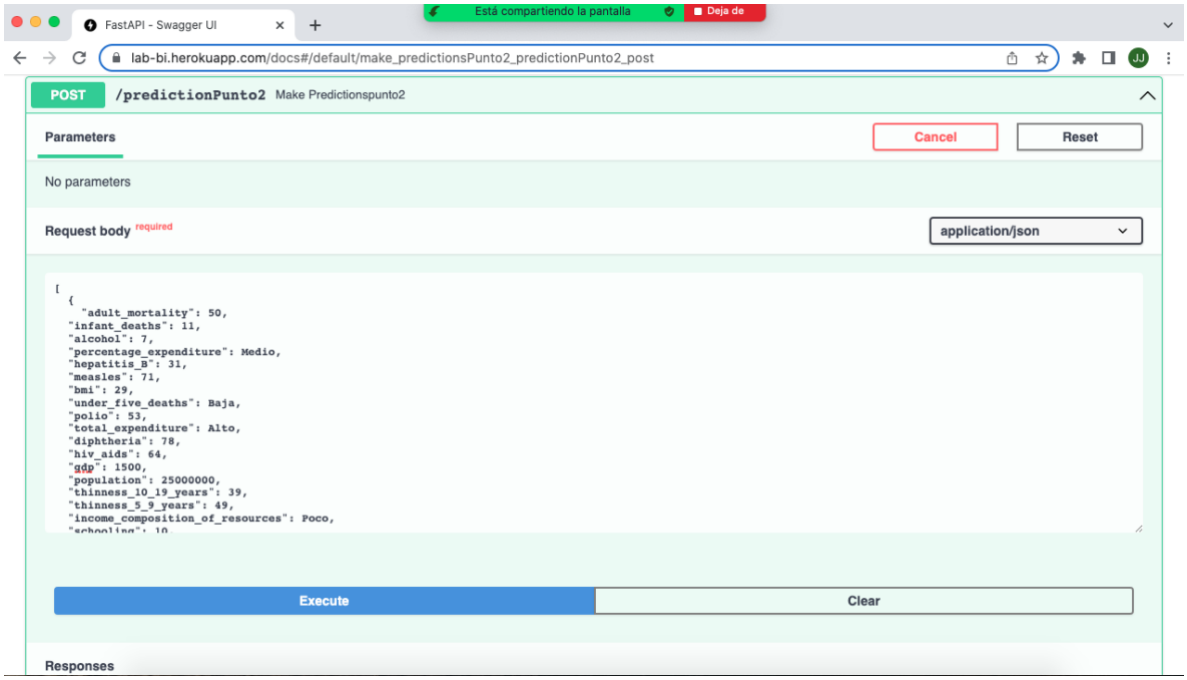
```
 "infant_deaths": 10,
 "alcohol": 10,
 "percentage_expenditure": 34,
 "hepatitis_B": 27,
 "measles": No,
 "bmi": 24,
 "under_five_deaths": 7,
 "polio": 67,
 "total_expenditure": Ochenta,
 "diphtheria": 77,
 "hiv_aids": 57,
 "gdp": 1200,
 "population": 34000000,
 "thinness_10_19_years": 35,
 "thinness_5_9_years": Alta,
 "income_composition_of_resources": 0.6,
 "schooling": 12,
  "Valor_Esperado": 0
 }
]
```


- Resultado:

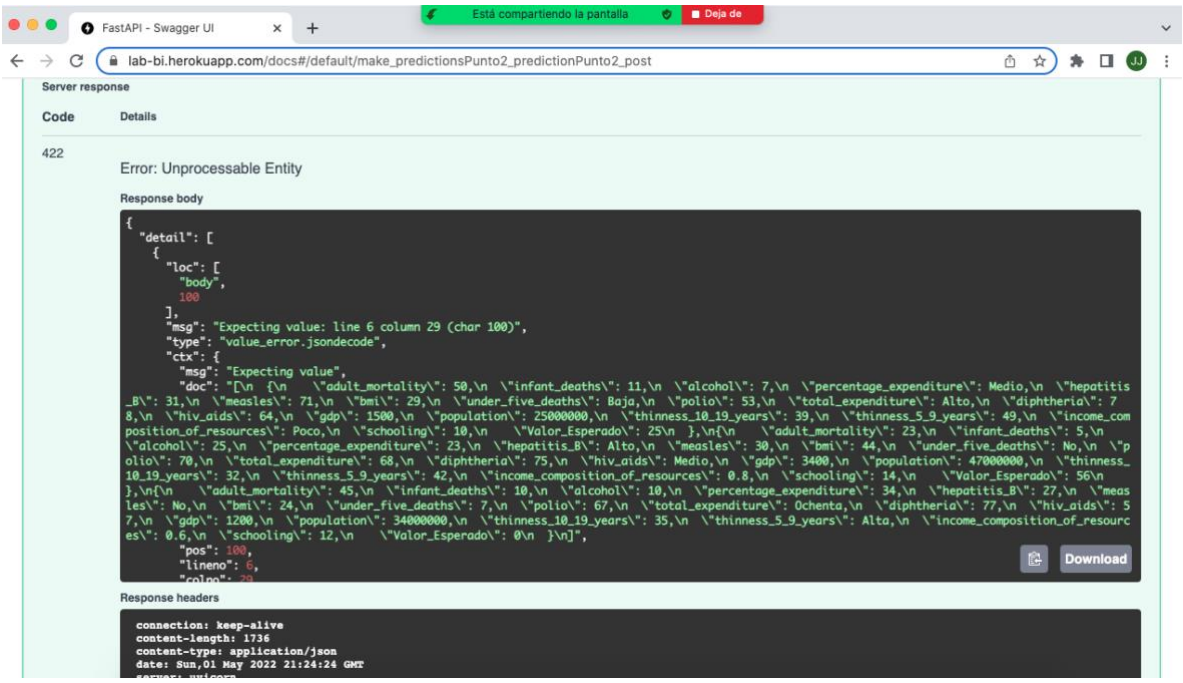

```
{
 "detail": [
  {
   "loc": [
    "body",
    100
   ],
   "msg": "Expecting value: line 6 column 29 (char 100)",
   "type": "value_error.jsondecode",
   "ctx": {
    "msg": "Expecting value",
    "doc": "[\n {\n  \"adult_mortality\": 50,\n  \"infant_deaths\": 11,\n  \"alcohol\":
7,\n  \"percentage_expenditure\": Medio,\n  \"hepatitis_B\": 31,\n  \"measles\":
71,\n  \"bmi\": 29,\n  \"under_five_deaths\": Baja,\n  \"polio\": 53,\n
\"total_expenditure\": Alto,\n  \"diphtheria\": 78,\n  \"hiv_aids\": 64,\n  \"gdp\":
1500,\n  \"population\": 25000000,\n  \"thinness_10_19_years\": 39,\n
\"thinness_5_9_years\": 49,\n  \"income_composition_of_resources\": Poco,\n
\"schooling\": 10,\n  \"Valor_Esperado\": 25\n },\n{\n  \"adult_mortality\": 23,\n
\"infant_deaths\": 5,\n  \"alcohol\": 25,\n  \"percentage_expenditure\": 23,\n
\"hepatitis_B\": Alto,\n  \"measles\": 30,\n  \"bmi\": 44,\n  \"under_five_deaths\":
No,\n  \"polio\": 70,\n  \"total_expenditure\": 68,\n  \"diphtheria\": 75,\n
\"hiv_aids\": Medio,\n  \"gdp\": 3400,\n  \"population\": 47000000,\n
\"thinness_10_19_years\": 32,\n  \"thinness_5_9_years\": 42,\n
```

\"income_composition_of_resources\": 0.8,\n \"schooling\": 14,\n \"Valor_Esperado\": 56\n },\n{\n \"adult_mortality\": 45,\n \"infant_deaths\": 10,\n \"alcohol\": 10,\n \"percentage_expenditure\": 34,\n \"hepatitis_B\": 27,\n \"measles\": No,\n \"bmi\": 24,\n \"under_five_deaths\": 7,\n \"polio\": 67,\n \"total_expenditure\": Ochenta,\n \"diphtheria\": 77,\n \"hiv_aids\": 57,\n \"gdp\": 1200,\n \"population\": 34000000,\n \"thinness_10_19_years\": 35,\n \"thinness_5_9_years\": Alta,\n \"income_composition_of_resources\": 0.6,\n \"schooling\": 12,\n \"Valor_Esperado\": 0\n }\n]",
"pos": 100,
"lineno": 6,
"colno": 29
}
}
]
}

- Proceso realizado de Postman:
  1. Se colocan los valores de entrada y se ejecuta el programa:



  2. Se dirige a la ventana de Responses, en Response body, del sistema y se verifica el resultado:

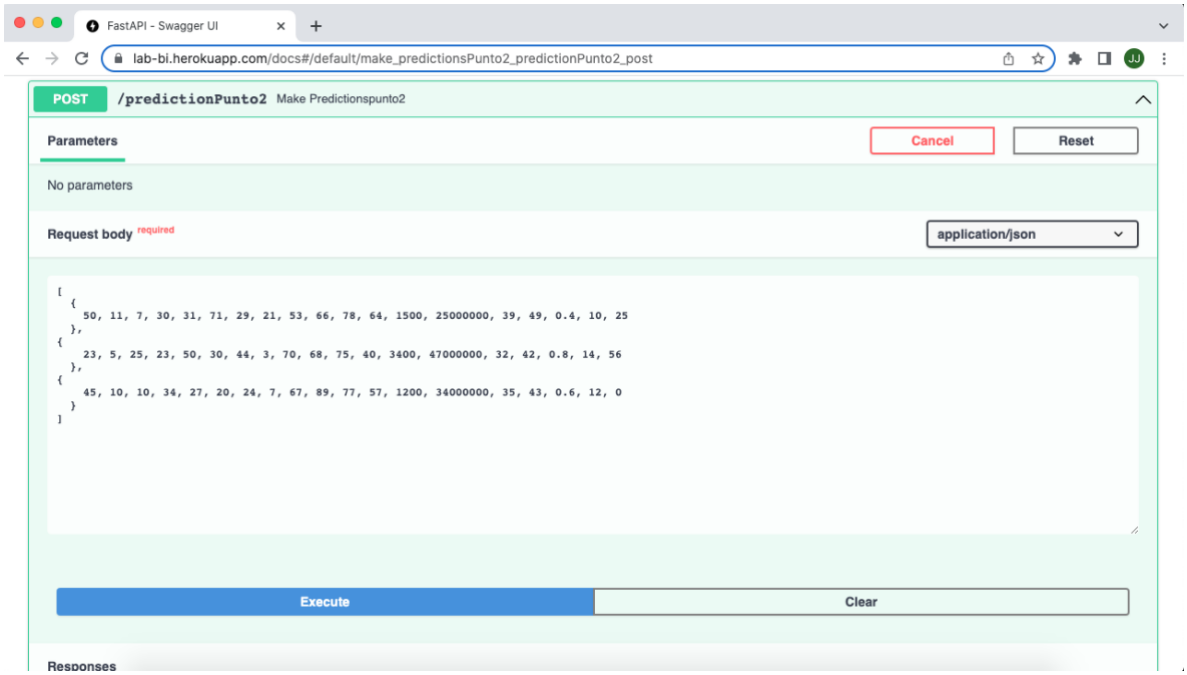


- Comentarios generales:
  En este caso, los modelos tenian problemas en sus datos de entrada, ya que poseen letras en algunos valores de los atributos. Lo que se espera es que el problema no logre ejecutar correctamente el programa. Como se esperaba, el progrma identifica que algunos de los valores de entrada de los modelos no poseen el tipo de dato correcto, y esto impide que se pueda hacer una analisis coherente y acertivo de los datos. De esta forma, se muestra un error con los valores erroneos al ejecutar el programa.

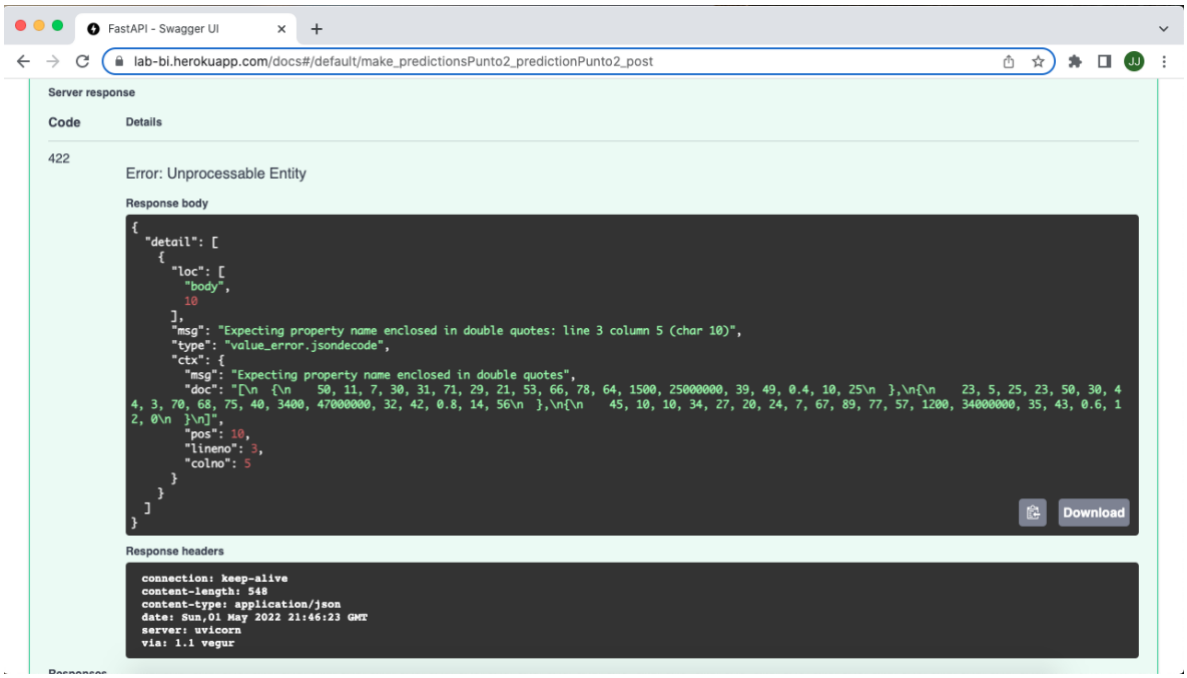
- Escenario 5:
  - Datos de entrada (formato JSON):
    [
     {
      50, 11, 7, 30, 31, 71, 29, 21, 53, 66, 78, 64, 1500, 25000000, 39, 49, 0.4, 10, 25
     },
    {
      23, 5, 25, 23, 50, 30, 44, 3, 70, 68, 75, 40, 3400, 47000000, 32, 42, 0.8, 14, 56
     },
    {
      45, 10, 10, 34, 27, 20, 24, 7, 67, 89, 77, 57, 1200, 34000000, 35, 43, 0.6, 12, 0
     }
    ]
  - Resultado:
    {
     "detail": [
      {
       "loc": [
        "body",

```
      10
     ],
     "msg": "Expecting property name enclosed in double quotes: line 3 column 5 (char
10)",
     "type": "value_error.jsondecode",
     "ctx": {
      "msg": "Expecting property name enclosed in double quotes",
      "doc": "[\n  {\n    50, 11, 7, 30, 31, 71, 29, 21, 53, 66, 78, 64, 1500, 25000000, 39,
49, 0.4, 10, 25\n  },\n{\n    23, 5, 25, 23, 50, 30, 44, 3, 70, 68, 75, 40, 3400, 47000000,
32, 42, 0.8, 14, 56\n  },\n{\n    45, 10, 10, 34, 27, 20, 24, 7, 67, 89, 77, 57, 1200,
34000000, 35, 43, 0.6, 12, 0\n  }\n]",
      "pos": 10,
      "lineno": 3,
      "colno": 5
     }
    }
  ]
}
```


- Proceso realizado de Postman:
  1. Se colocan los valores de entrada y se ejecuta el programa:



  2. Se dirige a la ventana de Responses, en Response body, del sistema y se verifica el resultado:



- Comentarios generales:
  En este caso, se presenta un patrón de error en donde solo se colocan numeros para los modelos de entrada. Lo que se espera es que el programa no se ejecute debido al formato de los datos de entrada. Por ende, lo que se tiene es que el programa no se ejecuta correctamente, ya que no tiene los indicadores de los valores de cada atributo y no sabe como asignarlos. En este sentido, no se tiene una buena comprensión de los datos en los modelos, ya que no cumplen el formato JSON.

Manejo de errores:

Para disminuir y manejar los errores generados se pueden tomar varias medidas:

1. Transformar los valores no numéricos que lleguen como parámetro de entrada para poder convertirlos a su respectivo valor numérico.

2. Aceptar más de un formato para que los usuarios puedan usar otros diferentes a JSON. En tal caso de que se use uno no compatible, comunicarle al usuario cuales son los formatos permitidos.

3. Cuando en el Endpoint numero 2 se ingresen menos de 2 elementos, comunicarle al usuario que es necesario mandar mas elementos.

Links de interes

Github => https://github.com/Pinzon98/Laboratorio-4-BI

API => https://lab-bi.herokuapp.com/docs