**Ejemplo de implementacion con flask:** <https://gist.github.com/linar-jether/95ff412f9d19fdf5e51293eb0c09b850>

<https://oznetnerd.com/2018/04/17/writing-a-grafana-backend-using-the-simple-json-datasource-flask/>

**Endpoint: GET /**

**Request:** Vacío.

**Response:** 200 OK.

**Función:** Se usa para testear conexión con la API.

**Endpoint: POST /search**

**Request:** Puede tener parámetros o no por lo que hay que contemplar el caso en el que no tenga body (Si no lo contemplas salta excepción). En el caso de que tenga parámetros recibiremos algo como: { "target": "upper\_50" }

**Return:** Tiene que devolver una lista de categorías en JSON.

Ejemplo: ["speed", "pressure", "temperature", "latitude", "voltage"]

**Función:** Tiene que devolver las categorías de las métricas.

**Endpoint: POST /query**

**Request:**

Ejemplo:

{

"timezone": "browser",

"panelId": 2,

"range": {

"from": "2018-04-15T04:57:48.126Z",

"to": "2018-04-15T05:42:32.733Z",

"raw": {

"from": "2018-04-15T04:57:48.126Z",

"to": "2018-04-15T05:42:32.733Z"

}

},

"rangeRaw": {

"from": "2018-04-15T04:57:48.126Z",

"to": "2018-04-15T05:42:32.733Z"

},

"interval": "2s",

"intervalMs": 2000,

"targets": [

{

"target": "host1\_ts",

"refId": "A",

"type": "timeserie"

}

],

"maxDataPoints": 928,

"scopedVars": {

"\_\_interval": {

"text": "2s",

"value": "2s"

},

"\_\_interval\_ms": {

"text": 2000,

"value": 2000

}

}

}

**from and to:** UTC ISO 8601. Tendremos que convertirlo a Unix Timestamp para la response.

**interval y intervalMs:** Es el intervalo entre puntos que grafana espera recibir. Como ejemplo, si seleccionamos en Grafana los últimos 5 minutos del gráfico, se espera recibir datapoints con una distancia de 200 ms.

**target y type:** Target desde el que queremos recibir los datos, el tipo puede ser o *timeserie* para gráficos de tiempo o *table* para gráficos de tabla.

**maxDataPoints:** Techo que indica el número máximo de puntos que tenemos que enviar. (Si nos pasamos puede petar el browser del cliente).

**Response:** Puede contener cualquier tipo de dato que se pueda convertir a un Grafana Data Frame usando esta función:

<https://github.com/grafana/grafana/blob/1e024f22b8f767da01c9322f489d7b71aeec19c3/packages/grafana-data/src/dataframe/processDataFrame.ts#L284>

Ejemplo (El valor de la métrica tiene que se float, el valor del tiempo Unix Timestamp en milis):

[

{

"target":"pps in",

"datapoints":[

[622,1450754160000],

[365,1450754220000]

]

},

{

"target":"pps out",

"datapoints":[

[861,1450754160000],

[767,1450754220000]

]

},

{

"target":"errors out",

"datapoints":[

[861,1450754160000],

[767,1450754220000]

]

},

{

"target":"errors in",

"datapoints":[

[861,1450754160000],

[767,1450754220000]

]

}

]

La relación entre *target* en el request y response en 1:n. Puedes devolver múltiples *targets* en la response para un solo *target* en la request.

**Función:** Tiene que devolver datos del panel o anotaciones.