REST API

(C)Hurence

Version v1.0, 18.02.2020: First draft

Table des matières

Introduction

L'api REST de l'historian est actuellement composé des endpoints ci-dessous :

- */api/historian/v0* : L'api principal de l'historian qui ne contient actuellement que des endpoints pour l'import et l'export de données.
- POST */api/historian/v0/import/csv*: Un endpoint pour importer des points depuis des fichiers csv.
- POST */api/historian/v0/import/json*: Un endpoint pour importer des points au format json envoyé dans le corps de la requête.
- POST */api/historian/v0/export/csv*: Un endpoint pour exporter des points au format csv.
- GET */api/grafana/simplejson*: L'api pour interagir avec Grafana en utilisant le plugin datasource "SimpleJson".
- POST */api/grafana/simplejson/query*: Permet d'obtenir les points de l'historian.
- POST */api/grafana/simplejson/search*: Permet d'obtenir les noms des métriques de l'historian.
- POST */api/grafana/simplejson/annotations*: Permet d'obtenir les annotations enregistrées dans l'historian.
- POST */api/grafana/simplejson/tag-keys*: Permet d'obtenir les clés disponibles pour utiliser l'endpoint '*/api/grafana/simplejson/tag-values*'.
- POST */api/grafana/simplejson/tag-values*: Permet d'obtenir les valeurs possibles pour la clé indiquer.
- GET */api/grafana/v0*: L'api pour interagir avec Grafana en utilisant le plugin datasource de hurence "Hurence-Historian".
- POST */api/grafana/v0/query*: Permet d'obtenir les points de l'historian.
- POST */api/grafana/v0/search*: Permet d'obtenir les noms des métriques de l'historian.
- POST */api/grafana/v0/annotations*: Permet d'obtenir les annotations enregistrées dans l'historian.
- POST */api/grafana/v0/tag-keys*: Permet d'obtenir les clés disponibles pour utiliser l'endpoint '*/api/grafana/v0/tag-values*'.
- POST */api/grafana/v0/tag-values*: Permet d'obtenir les valeurs possibles pour la clé indiquer.

Les endpoints 'tag-keys' et 'tag-values' ne sont utile que pour Grafana.

Description détaillée des endpoints

Ingestion de données

Ces endpoints servent à injecter des points dans l'historian :

POST /api/historian/v0/import/csv

Permet d'injecter des points dans l'historian à partir de fichiers csv. Le format de la requête attendue est une requête avec Content-Type: multipart/form-data;

Table 1. Paramètres de la requête

attribut	multi valué	description	requis	valeurs possible	valeur par défaut
my_csv_file	Oui	le chemin des fichier csv a importer. Attention chaque fichier est importé individuellement de manière indépendante. Autrement dit, joindre plusieurs fichiers ou faire une requête par fichier est équivalent. Cependant les autres paramètres de la requêtes sont commun à tous les fichiers joints. Les fichiers doivent obligatoirement contenir un header.	Oui		
mapping.name	Non	le nom de la colonne qui correspond à "name". C'est l'identifiant principal d'une métrique.	Non	doit être un nom de colonne des fichiers csv joints.	"metric"
mapping.value	Non	le nom de la colonne qui correspond à "value". C'est la valeur du point. Les valeurs de cette colonne doivent être des valeurs numériques.	Non	Doit être un nom de colonne des fichiers csv joints.	"value"
mapping.timestamp	Non	Le nom de la colonne qui correspond à "timestamp". C'est la date du point. Par défaut cette colonne doit contenir des un timestamp epoch en milliseconde (voir format_date)	Non	Doit être un nom de colonne des fichiers csv joints.	"timestamp"
mapping.quality	Non	Le nom de la colonne qui correspond à "quality". C'est la qualité du point (pour le moment la qualité n'est pas stocker dans l'historian)	Non	Doit être un nom de colonne des fichiers csv joints.	"quality"

attribut	multi valué	description	requis	valeurs possible	valeur par défaut
mapping.tags	Oui	Les noms des colonnes à considérer comme un tag. Les tags sont des notions qui décrivent la métrique en plus de son nom (colonne "name"). Toutes les colonnes non renseigné dans le mapping seront ignorées lors de l'injection. TODO expliquer on prend le first	Non	Doit être un nom de colonne des fichiers csv joints.	
format_date	Non	Le format attendu pour les valeurs de la colonne "timestamp".	Non	Doit être soit une valeur parmi: [MILLISECONDS_EPOCH,SECONDS_EPOCH,MICROSECONDS_EPOCH,NANOSECONDS_EPOCH] NANOSECONDS_EPOCH : la valeur doit être le nombre de nanosecondes depuis le 1 janvier 1970 UTC. MICROSECONDS_EPOCH : la valeur doit être le nombre de microsecondes depuis le 1 janvier 1970 UTC. MILLISECONDS_EPOCH : la valeur doit être le nombre de millisecondes depuis le 1 janvier 1970 UTC. SECONDS_EPOCH : la valeur doit être le nombre de secondes depuis le 1 janvier 1970 UTC. Soit une autre valeur, dans ce cas cette valeur doit être un format de date valide, par exemple "yyyy-mm-dd".	"MILLISECONDS_EPOCH"
timezone_date	Non	le timezone du date dans le csv.	Non	doit être un timezone valide.	"UTC"

attribut	multi valué	description	requis	valeurs possible	valeur par défaut
group_by	Oui	Ce paramètre est très important ! Si il est mal utilisé il est possible de corrompre les données déjà existantes. Il faut lister ici tous les champs qui vont être utiliser pour construire les chunks. Par défaut on groupe les points en chunk uniquement en fonction de leur nom. Cependant il est possible de grouper également en fonction de la valeur de certains tags. Il faut savoir que cela va impacter la manière dont les données sont récupérer par la suite. Par exemple si on injecte des données en groupant par "name" et le tag "usine". Alors on pourra requêter les valeurs pour chaque usine en filtrant sur la bonne valeur du tag usine (voir l'api pour requêter les points). Seulement si par la suite quelqu'un injecte d'autres données sans grouper par le tag "usine" alors les données vont se retrouver mélanger.	Non	Les valeurs acceptés sont "name" (peu importe le mapping utilisez "name" et non le nom de la colonne dans le csv). Sinon les autres valeurs acceptées sont les noms de colonnes ajouter comme tag.	"name"

- S'il y a un problème avec la requête on reçoit une réponse 400 BAD_REQUEST.
- Si tout s'est bien passé on reçoit une réponse 201 CREATED :

Table 2. Description de la réponse

json path	type	description
tags	array	les tags renseignés dans la requête.
grouped_by	array	les champs qui sont utilisés pour le group by (renseignés dans la requête).
report	json object	un rapport sur les chunks qui ont été injectés.

Exemple de commande curl minimal :

```
curl -X POST \
http://localhost:8080/historian-server/ingestion/csv \
-F 'my_csv_file=@path/de/mon/csv/file.csv'
```

Exemple csv minimal :

```
metric,timestamp,value
metric_1, 1, 1.2
metric_1, 2, 2
metric_1, 3, 3
```

Exemple de commande curl :

```
curl -X POST \
http://localhost:8080/historian-server/ingestion/csv \
-F 'my_csv_file=@path/de/mon/csv/file.csv' \
-F 'my_csv_file2=@path/de/mon/csv/file2.csv' \
-F mapping.name=metric_name_2 \
-F mapping.value=value_2 \
-F mapping.timestamp=timestamp \
-F mapping.quality=quality \
-F mapping.tags=sensor \
-F mapping.tags=code_install \
-F timestamp_unit=MILLISECONDS_EPOCH \
-F group_by=name \
-F group_by=tags.sensor \
-F format_date=yyyy-D-m HH:mm:ss.SSS \
-F timezone_date=UTC \
```

Exemple de fichier csv :

```
metric_name_2,timestamp,value_2,quality,sensor,code_install
metric_1, 1970-1-1 00:00:00.001, 1.2 ,1.4,sensor_1,code_1
metric_1, 1970-1-1 00:00:00.002, 2 ,1.4,sensor_1,code_1
metric_1, 1970-1-1 00:00:00.003, 3 ,1.4,sensor_2,code_1
metric_2, 1970-1-1 00:00:00.004, 4 ,1.5,sensor_2,code_1
```

avec la requête suivante :

```
curl -X POST \
http://localhost:8080/historian-server/ingestion/csv \
-F 'my_csv_file=@path/de/mon/csv/file.csv' \
-F 'my_csv_file2=@path/de/mon/csv/file.csv' \
-F mapping.name=metric_name_2 \
-F mapping.value=value_2 \
-F mapping.timestamp=timestamp \
-F mapping.quality=quality \
-F mapping.tags=sensor \
-F mapping.tags=code_install \
-F timestamp_unit=MILLISECONDS_EPOCH \
-F group_by=name \
-F group_by=tags.sensor \
-F format_date=yyyy-D-m HH:mm:ss.SSS \
-F timezone_date=UTC \
```

et le fichier "file.csv" :

```
metric_name_2,timestamp,value_2,quality,sensor,code_install
metric_1, 1970-1-1 00:00:00.001, 1.2 ,1.4,sensor_1,code_1
metric_1, 1970-1-1 00:00:00.002, 2 ,1.4,sensor_1,code_1
metric_1, 1970-1-1 00:00:00.003, 3 ,1.4,sensor_2,code_1
metric_2, 1970-1-1 00:00:00.004, 4 ,1.5,sensor_2,code_1
```

on recoit cette réponse :

```
{
  "tags": ["sensor", "code_install"],
  "grouped_by": ["name", "sensor"],
  "report" : [
    {
      "name": "metric_1",
      "sensor": "sensor_1",
      "number_of_points_injected": 4,
      "number_of_point_failed": 0,
      "number_of_chunk_created": 2
    },
    {
      "name": "metric_1",
      "sensor": "sensor_2",
      "number_of_points_injected": 2,
      "number_of_point_failed": 0,
      "number_of_chunk_created": 2
    },
    {
      "name": "metric_2",
      "sensor": "sensor_2",
      "number_of_points_injected": 2,
      "number_of_point_failed": 0,
      "number_of_chunk_created": 2
    }
  ]
}
```

(on remarque ici on'a utilisé le même fichier deux fois.)

POST /api/historian/v0/import/json

Permet d'injecter des points dans l'historian à partir d'un corp de requête au format json. Le format de la requête attendue est une requête avec Content-Type: application/json;

Exemple de requête :

```
[
    {
        "name": "temp",
        "points": [
            [100, 1.0],
            [200, 1.2]
        ]
    },
   {
        "name": "temp_2",
        "points": [
           [100, 1.7],
           [200, 1.9]
        ]
   }
]
```

Exemple de requête minimal :

```
[
    {
        "name": "temp",
        "points": [
            [100, 1.0]
        ]
    }
]
```

Le corp de la requête attendue est une liste json d'objet. Chaque objet doit être composé des attribut suivants :

Table 3. Propriété des objets json

json path	**type**	**description**	**requis**	**/name**	**String**
Le nom de la métrique pour laquelle on va injecter les points.	Oui	/points	Array d'array	Les points pour créer un chunk avec le nom ed métrique assigné à 'name'. Les points sont a renseigné sous la forme [[timestsamp<long>, value<double>],..,[timestsamp<long>, value<double>]].	Oui

Exemple de réponse

```
[
  {
    "target":"upper_75",
    "datapoints":[
      [622,1450754160000],
      [365,1450754220000]
    ]
  },
  {
    "target":"upper_90",
    "datapoints":[
      [861,1450754160000],
      [767,1450754220000]
    ]
  }
]
```

Export de données

POST /api/historian/v0/export/csv

Permet de rechercher des points et les exporter en csv pour les métriques désirées.

Ce endpoint utilise la même requête que l'endpoint query mais il donne la réponse en csv.

Exemple de réponse :

```
metric,value,date
temp_a,622.1,1477895624866
temp_a,-3.0,1477916224866
temp_a,365.0,1477917224866
temp_b,861.0,1477895624866
temp_b,767.0,1477917224866
```

Interactions avec Grafana

Les api ci-dessous sont utilisées pour interagir avec Grafana en utilisant certaines datasources (une notion de Grafana).

Api compatible avec le plugin "SimpleJson"

GET /api/grafana/simplejson

Cet endpoint renvoie une réponse 200 'OK' si l'historian fonctionne correctement.

POST /api/grafana/simplejson/query

Permet de rechercher des points pour les métriques désirées. Divers paramètres sont disponible.

Exemple de requête :

```
{
    "panelId": 1,
    "range": {
        "from": "2016-10-31T06:33:44.866Z",
        "to": "2016-10-31T12:33:44.866Z",
        "raw": {
            "from": "now-6h",
            "to": "now"
        }
    },
    "rangeRaw": {
        "from": "now-6h",
        "to": "now"
    },
    "interval": "30s",
    "intervalMs": 30000,
    "targets": [
        { "target": "upper_50", "refId": "A", "type": "timeserie" },
        { "target": "upper_75", "refId": "B", "type": "timeserie" }
    ],
    "adhocFilters": [{
        "key": "City",
        "operator": "=",
        "value": "Berlin"
    }],
    "format": "json",
    "maxDataPoints": 550
}
```

Exemple de requête minimal :

```
{
    "targets": [
      { "target": "upper_50" }
    ]
}
```

Table 4. Paramètres de la requête

json path	type	description	requis	valeurs possible	valeur par défaut
/targets	array	Permet de renseigner les métriques pour lesquelles on souhaite obtenir des points.	Oui		
/targets/target	String	Le nom d'une métrique pour laquelle on souhaite obtenir des points.	Oui (au moins une métrique doit être renseigné)		
/range/from	String	La date de début pour rechercher les points. Il ne sera retourner que les points avec une date supérieur ou égal à cette date.	Non	Doit réprésenter une date au format suivant (UTC) : **yyyy-MM-dd'T'HH:mm:ss.SSS**	Le 1 Janvier 1960 (UTC)
/range/to	String	La date de fin pour rechercher les points. Il ne sera retourner que les points avec une date inférieur ou égal à cette date.	Non	Doit réprésenter une date au format suivant (UTC) : **yyyy-MM-dd'T'HH:mm:ss.SSS**	La valeur par défaut est l'infini
/maxDataPoints	long	Le nombre maximum de point désirer pour chaque métrique (En effet le but est de tracer des points sur un graphe). Si nécessaire les points seront echantilloné avec un algorithm sa sampling par défaut.	Non	positif	1000
/adhocFilters	String	Utiliser par grafana	Non		

NOTE	Le reste des paramètres sont des paramètres qui sont envoyé par grafana mais qui ne sont pas exploité pour le moment.

Exemple de réponse

```
[
  {
    "target":"upper_75",
    "datapoints":[
      [622,1450754160000],
      [365,1450754220000]
    ]
  },
  {
    "target":"upper_90",
    "datapoints":[
      [861,1450754160000],
      [767,1450754220000]
    ]
  }
]
```

POST /api/grafana/simplejson/search

Permet de chercher les différentes métrics disponibles.

Exemple de requête :

```
{ "target": "upper_50" }
```

Table 5. Paramètres de la requête

json path	type	description	requis	valeurs possible	valeur par défaut
/target	String	Une partie du nom de la métric recherché.	Non		

NOTE Le corps de la requête est optionnel.

Exemple de réponse :

```
["upper_25","upper_50","upper_75","upper_90","upper_95"]
```

POST /api/grafana/simplejson/annotations

Permet de chercher les annotations.

Exemple de requête :

```
{
  "range": {
    "from": "2020-2-14T01:43:14.070Z",
    "to": "2020-2-14T06:50:14.070Z"
  },
  "limit" : 100,
  "tags": ["tag1", "tag2"],
  "matchAny": false,
  "type": "tags"
}
```

Exemple de réponse actuelle

```
[
  {
    "annotation": "annotation",
    "time": "time",
    "title": "title",
    "tags": "tags",
    "text": "text"
  }
]
```

Table 6. Paramètres de la requête

json path	type	description	requis	valeurs possible	valeur par défaut
/range/from	String	La date de début pour rechercher les annotations. Il ne sera retourner que les annotations avec une date supérieur ou égal à cette date.	Non	Doit réprésenter une date au format suivant (UTC) : **yyyy-MM-dd'T'HH:mm:ss.SSS**	Le 1 Janvier 1960 (UTC)
/range/to	String	La date de fin pour rechercher les annotations. Il ne sera retourner que les annotations avec une date inférieur ou égal à cette date.	Non	Doit réprésenter une date au format suivant (UTC) : **yyyy-MM-dd'T'HH:mm:ss.SSS**	La valeur par défaut est l'infini
/limit	Integer	Le nombre maximum d'annotation a renvoyer	Non	entier positif	100
/tags	Liste de string	Le nom des tags pour filtrer les annotations. Le comportement dépend de la valeur du champs */type*	Non		[] (empty array)

json path	type	description	requis	valeurs possible	valeur par défaut
/matchAny	Boolean	Si le champs *type* vaut *TAGS*. Si *true* les annotations doivent avoir au moins un des tags du champs *tags* sinon les annotations doivent contenir tous les tags.	Non		true
/type	String	Le type de requête. Si la valeur est "ALL", toutes les annotations dans l'interval de temps sera renvoyé. Si le type est "TAGS", les annotations devront en plus contenir des tags (soit tous soit au moins un selon la valeur de *matchAny*.	Non	une valeur parmis [ALL, TAGS]	ALL

POST /api/grafana/simplejson/tag-keys

Permet de savoir les clés utilisable dans l'endpoint *tag_values*.

Exemple de requête :

```
{}
```

Exemple de réponse :

```
[
    {"type":"string","text":"City"},
    {"type":"string","text":"Country"}
]
```

Les valeurs dans text sont les valeurs utilsable comme clé dans l'endpoint tag-values.

Table 7. Paramètres de la requête

json path	type	description	requis	valeurs possible	valeur par défaut
/range/from	String	La date de début pour rechercher les annotations. Il ne sera retourner que les annotations avec une date supérieur ou égal à cette date.	Non	Doit réprésenter une date au format suivant (UTC) : **yyyy-MM-dd'T'HH:mm:ss.SSS**	Le 1 Janvier 1960 (UTC)

json path	type	description	requis	valeurs possible	valeur par défaut
/range/to	String	La date de fin pour rechercher les annotations. Il ne sera retourner que les annotations avec une date inférieur ou égal à cette date.	Non	Doit réprésenter une date au format suivant (UTC) : **yyyy-MM-dd'T'HH:mm:ss.SSS**	La valeur par défaut est l'infini
/limit	Integer	Le nombre maximum d'annotation a renvoyer	Non	entier positif	100
/tags	Liste de string	Le nom des tags pour filtrer les annotations. Le comportement dépend de la valeur du champs */type*	Non		[] (empty array)
/matchAny	Boolean	Si le champs *type* vaut *TAGS*. Si *true* les annotations doivent avoir au moins un des tags du champs *tags* sinon les annotations doivent contenir tous les tags.	Non		true
/type	String	Le type de requête. Si la valeur est "ALL", toutes les annotations dans l'interval de temps sera renvoyé. Si le type est "TAGS", les annotations devront en plus contenir des tags (soit tous soit au moins un selon la valeur de *matchAny*.	Non	une valeur parmis [ALL, TAGS]	ALL

POST /api/grafana/simplejson/tag-values

Permet de chercher les différentes métrics disponibles.

Exemple de requête :

```
{"key": "City"}
```

Exemple de réponse :

```
[
    {"text": "Eins!"},
    {"text": "Zwei"},
    {"text": "Drei!"}
]
```

These are the values available for the corresponding keys.

Api compatible avec le plugin "Hurence-Historian"

GET /api/grafana/v0

Cet endpoint renvoie une réponse 200 'OK' si l'historian fonctionne correctement.

POST /api/grafana/v0/query

Permet de rechercher des points pour les métriques désirées. Divers paramètres sont disponible.

Exemple de requête :

```
{
  "from": "2016-10-31T06:33:44.866Z",
  "to": "2020-10-31T12:33:44.866Z",
  "names": ["metric_1"],
  "format": "json",
  "max_data_points": 30,
  "tags": {
      "sensor" : "sensor_1"
  },
  "sampling":{
     "algorithm": "MIN",
     "bucket_size" : 100
  }
}
```

Exemple de requête minimal :

```
[
    {
        "name": "metric_1",
        "datapoints": [
            [
                622,
                1477895624866
            ],
            [
                -3,
                1477916224866
            ],
            [
                365,
                1477917224866
            ]
        ]
    }
]
```

{ "tags": { "sensor" : "sensor_1" }, "sampling":{ "algorithm": "MIN", "bucket_size" : 100 } }

Table 8. Paramètres de la requête

json path	**type**	**description**	**requis**	**valeurs possible**	**valeur par défaut**
/names	array	Permet de renseigner les métriques pour lesquelles on souhaite obtenir des points.	Oui		
from	String	La date de début pour rechercher les points. Il ne sera retourner que les points avec une date supérieur ou égal à cette date.	Non	Doit réprésenter une date au format suivant (UTC) : **yyyy-MM-dd'T'HH:mm:ss.SSS**	Le 1 Janvier 1960 (UTC)
to	String	La date de fin pour rechercher les points. Il ne sera retourner que les points avec une date inférieur ou égal à cette date.	Non	Doit réprésenter une date au format suivant (UTC) : **yyyy-MM-dd'T'HH:mm:ss.SSS**	La valeur par défaut est l'infini
/max_data_points	long	Le nombre maximum de point désirer pour chaque métrique (En effet le but est de tracer des points sur un graphe). Si nécessaire les points seront echantilloné avec un algorithm sa sampling par défaut.	Non	positif	1000
/sampling	String	Un objet qui va contenir les information des paramètres à utiliser pour l'echantillonage des points si le nombre de points a retourner est supérieur au nombre maximum de point demandé.	Non		

json path	type	description	requis	valeurs possible	valeur par défaut
/sampling/algorithm	String	L'algorithme a utilisé pour l'échantillonage	Non	les valeurs possibles sont ["NONE", "AVERAGE", "FIRST", "MIN", "MAX"]	AVERAGE (la moyenne des points pour chaque bucket, le timestamp utilisé est celui du premier point du bucket)
/sampling/bucket_size	Int	La taille des buckets souhaité pour échantillonné les données. Attention cette taille peut être changé par le serveur si cela est incompatible avec le nombre maximum de point retourné souhaité.	Non		Calculer automatiquement par le serveur afin d'obtenir au plus le nombre maximum de points.
/tags	String	Permet d'obtenir uniquement les points des chunks qui contiennent les tags renseignés. Les tags doivent être indiqué sous la forme d'une map de clé valeur.	Non		Pas de tags.

Exemple de réponse

```
[
  {
    "name":"upper_75",
    "datapoints":[
      [622,1450754160000],
      [365,1450754220000]
    ]
  },
  {
    "name":"upper_90",
    "datapoints":[
      [861,1450754160000],
      [767,1450754220000]
    ]
  }
]
```

POST /api/grafana/v0/search

Permet de chercher les différentes métrics disponibles.

Exemple de requête :

```
{
  "name": "upper_50",
  "limit": 5
}
```

Table 9. Paramètres de la requête

json path	type	description	requis	valeurs possible	valeur par défaut
/name	String	Une partie du nom de la métrique recherché.	Non		
/limit	Int	Le nombre maximum de nom de métrique a retourné.	Non		

NOTE Le corps de la requête est optionnel.

Exemple de réponse :

```
["upper_25","upper_50","upper_75","upper_90","upper_95"]
```

POST /api/grafana/v0/annotations

Permet de chercher les annotations.

Exemple de requête :

```
{
  "from": "2020-2-14T01:43:14.070Z",
  "to": "2020-2-14T06:50:14.070Z",
  "limit" : 100,
  "tags": ["tag1", "tag2"],
  "matchAny": false,
  "type": "tags"
}
```

Exemple de réponse actuelle

```
{
  "annotations" : [
    {
      "time": 1581669794070,
      "text": "annotation 7",
      "tags": ["tag2","tag3"]
    },
    {
      "time": 1581666194070,
      "text": "annotation 6",
      "tags": ["tag3","tag5"]
    }
  ],
  "total_hit" : 2
}
```

Table 10. Paramètres de la requête

json path	type	description	requis	valeurs possible	valeur par défaut
/from	String	La date de début pour rechercher les annotations. Il ne sera retourner que les annotations avec une date supérieur ou égal à cette date.	Non	Doit réprésenter une date au format suivant (UTC) : **yyyy-MM-dd'T'HH:mm:ss.SSS**	Le 1 Janvier 1960 (UTC)

json path	type	description	requis	valeurs possible	valeur par défaut
/to	String	La date de fin pour rechercher les annotations. Il ne sera retourner que les annotations avec une date inférieur ou égal à cette date.	Non	Doit réprésenter une date au format suivant (UTC) : **yyyy-MM-dd'T'HH:mm:ss.SSS**	La valeur par défaut est l'infini
/limit	Integer	Le nombre maximum d'annotation a renvoyer	Non	entier positif	100
/tags	Liste de string	Le nom des tags pour filtrer les annotations. Le comportement dépend de la valeur du champs */type*	Non		[] (empty array)
/matchAny	Boolean	Si le champs *type* vaut *TAGS*. Si *true* les annotations doivent avoir au moins un des tags du champs *tags* sinon les annotations doivent contenir tous les tags.	Non		true
/type	String	Le type de requête. Si la valeur est "ALL", toutes les annotations dans l'interval de temps sera renvoyé. Si le type est "TAGS", les annotations devront en plus contenir des tags (soit tous soit au moins un selon la valeur de *matchAny*.	Non	une valeur parmis [ALL, TAGS]	ALL

POST /api/grafana/v0/tag-keys

Permet de savoir les clés utilisable dans l'endpoint *tag_values*.

Exemple de requête :

```
{}
```

Exemple de réponse :

```
[
    {
        "type": "string",
        "text": "Algo"
    },
    {
        "type": "int",
        "text": "Bucket size"
    }
]
```

Les valeurs dans text sont les valeurs utilsable comme clé dans l'endpoint tag-values.

POST /api/grafana/simplejson/tag-values

Permet de chercher les différentes métrics disponibles.

Exemple de requête :

```
{"key": "Algo"}
```

Exemple de réponse :

```
[
    {
        "text": "NONE"
    },
    {
        "text": "AVERAGE"
    },
    {
        "text": "FIRST"
    },
    {
        "text": "MIN"
    },
    {
        "text": "MAX"
    }
]
```

These are the values available for the corresponding keys.