

Módulo em Python de integração com TSWS

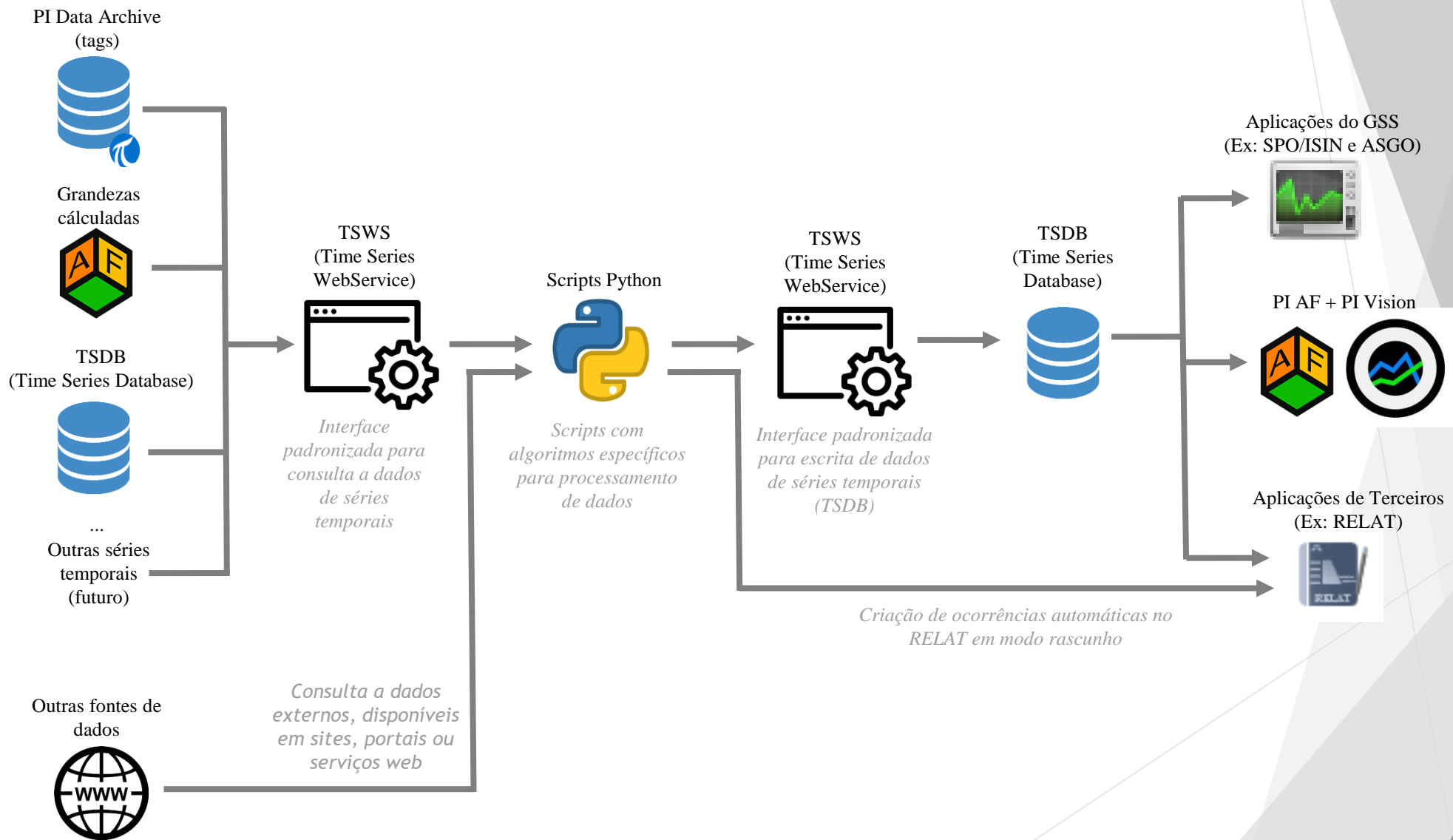
Rafael Conte Marodin

O que é o TSWS

- ▶ O TSWS é um serviço web, desenvolvido pelo GSS, que permite acesso padronizado a séries temporais
 - ▶ Leitura: tags do PI, atributos do PI-AF e tags do TSDB;
 - ▶ Escrita: tags do TSDB;
- ▶ As consultas são feitas através de requisições HTTPS
- ▶ A fim de abstrair e simplificar o uso do serviço TSWS desenvolveu-se este módulo Python.

Arquitetura de referência

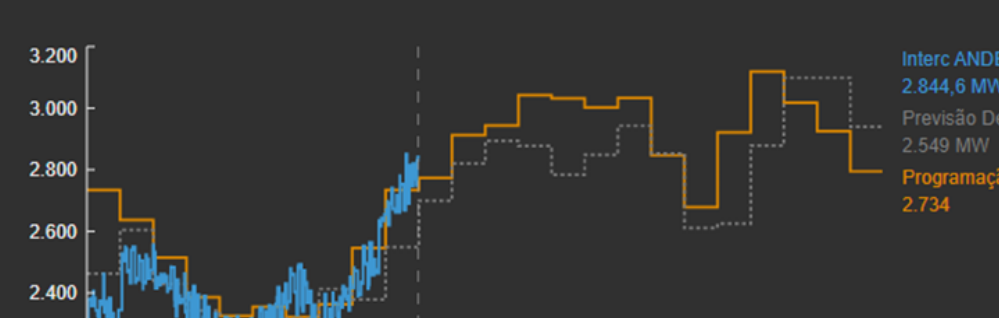
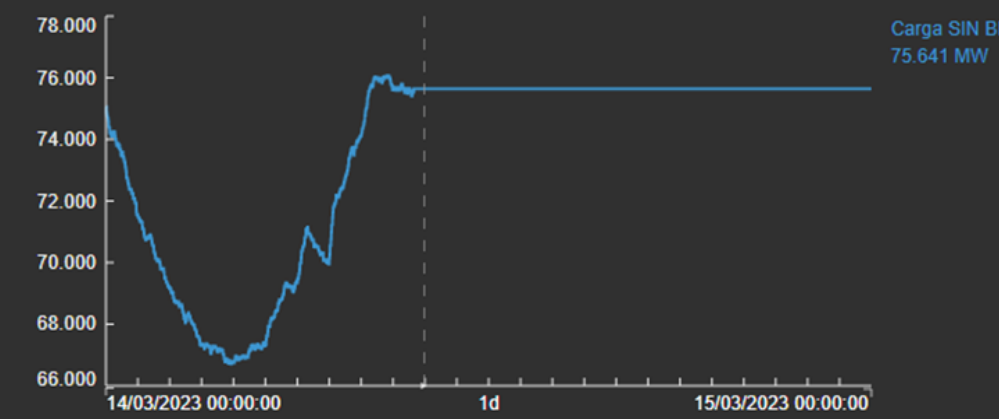
Scripts Python + TSWS



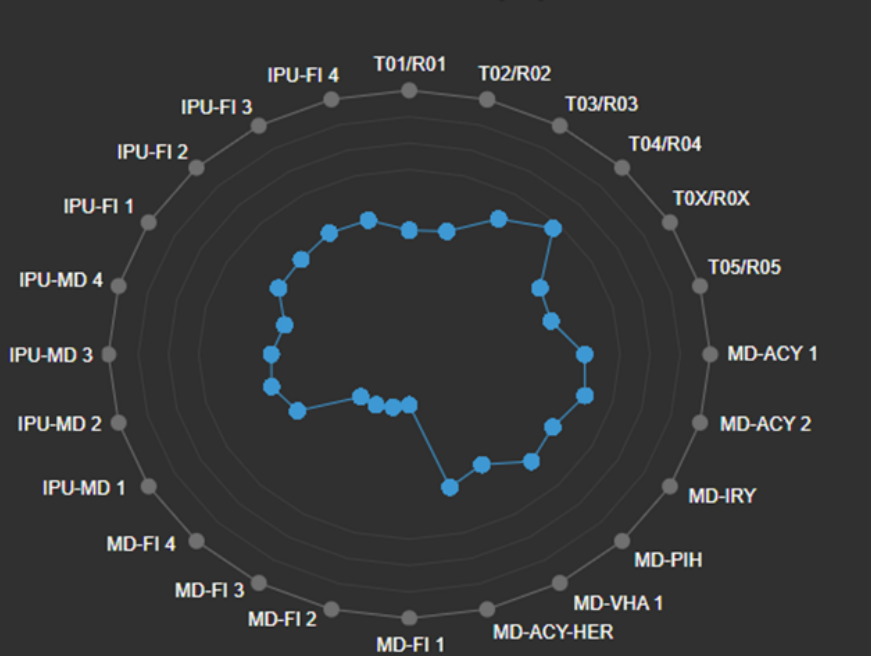
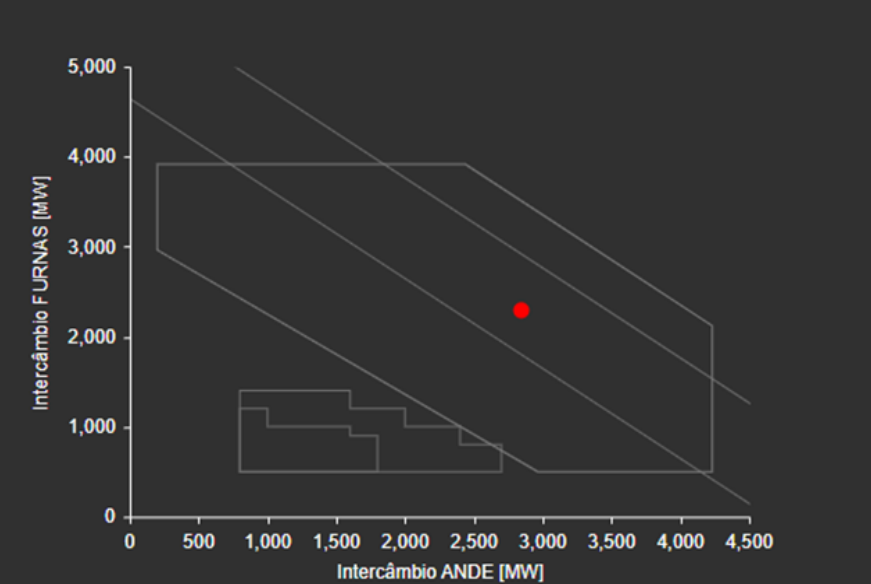
Exemplo de uso

Setor 50Hz	CAG 50Hz		
Geração 5.150 MW	Programa 4.472 MW	Limite Geração 6.354 MW	
Reserva 1.203 MW	Tempo rampa 10 MIN	Limite Transmissão ANDE 4.230,5	
		Limite Transmissão FURNAS 3.915 MW	

Estado Equipamentos		
LI 500kV 60Hz IPU-FI 1	LI 500kV MD-FI 1	LI 220kV MD-ACY 1
Ligada	Ligada	Ligada
LI 500kV 60Hz IPU-FI 2	LI 500kV MD-FI 2	LI 220kV MD-ACY 2
Ligada	Ligada	Ligada
LI 500kV 60Hz IPU-FI 3	LI 500kV MD-FI 3	LI 220kV MD-IRY
Ligada	Ligada	Ligada
LI 500kV 60Hz IPU-FI 4	LI 500kV MD-FI 4	LI 220kV MD-PIH
Ligada	Ligada	Ligada
	LI 500kV MD-VHA1	
	Ligada	



14/03/2023 08:58:51 1h 14/03/2023 09:58:51




Configurações iniciais

- ▶ As configurações do TSWS ficam armazenadas no arquivo config.json que deverá ser armazenado na pasta raiz do seu projeto

 config.json

 exemplo.py

 log_exemplo.py

 testing.py

- ▶ Os ambientes disponíveis do TSWS são 'dev' (desenvolvimento) e 'prd' (produção), descrevendo, respectivamente, aplicações que ainda estão em desenvolvimento e aplicações prontas para lançamento:
- ▶ Cada ambiente possui sua própria url:
 - ▶ Desenvolvimento: <https://chi764/TSWS/rest>
 - ▶ Produção: <https://op-gss/TSWS/rest>

Configurações iniciais: sistema de token

- ▶ Para autenticar requisições ao TSWS é utilizado um sistema de tokens para as consultas: quando o usuário fizer uma requisição ao sistema, seu token será fornecido de forma a confirmar sua identidade e garantir acesso as funções disponíveis
- ▶ Ou seja, o sistema de token funciona analogamente a um sistema de login e senha
- ▶ Portanto, deverá ser configurado no arquivo 'config.json' qual será o token utilizado para realizar as requisições.
- ▶ Os tokens não devem ser compartilhados entre os usuários, uma vez que, assim como o sistema de login e senha, identificam o utilizador e garantem acesso as ferramentas do sistema
- ▶ O usuário pode gerar um token para si utilizando o sistema de administração de tokens ATM.

Estrutura do arquivo 'config.json'

- ▶ Dentro de cada 'environment' do TSWS devem estar contidos pelo menos:
 - ▶ 'url': url base para a qual serão feitas as requisições
 - ▶ 'token': o token de autenticação que será usado para as requisições
- ▶ A estrutura do arquivo ficará, portanto:

```
"TSWS": {  
  "environments": {  
    "dev": {  
      "url": "https://chi764/TSWS/rest",  
      "token": ""  
    },  
    "prd": {  
      "url": "https://op-gss/TSWS/rest",  
      "token": ""  
    }  
  }  
}
```

Importação e inicialização

- ▶ Para importar o módulo:

```
from GSSLibs import log, TSWS
```

- ▶ Para inicializar o processo de log:

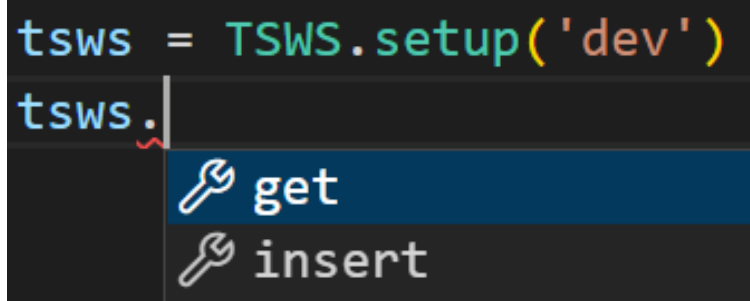
```
log.setup('GSSLib_test')
```

- ▶ Para inicializar as configurações do TSWS:

```
tsws = TSWS.setup('dev')
```


Modo de uso

- ▶ A interface criada para o TSWS é subdividida em duas propriedades: 'get' e 'insert'



```
tsws = TSWS.setup('dev')  
tsws.  
  get  
  insert
```

- ▶ Dentro da propriedade 'get' estão os métodos para aquisição de dados
- ▶ Dentro da propriedade 'insert' estão os métodos para inserção de dados
 - ▶ A inserção de dados não é diretamente possível para tags do ambiente PI
- ▶ Todos os métodos estão documentados e sua documentação pode ser acessada pelo atalho 'ctrl'+T tanto no VScode quando na IDE Spyder

Funções disponíveis:

```
tsws.get.tag_search(query)
```

(method) tag_search(query: str) -> Response

Procura por uma tag compatível com a expressão fornecida. O caracter '*' é utilizado como identificador universal, identificando quaisquer caracteres em qualquer quantidade em sua posição. Por exemplo, a string '*CAMMESA' corresponde a qualquer string começando com quaisquer caracteres e terminando com 'CAMMESA'. Ex: 'TSDB:CARGA CAMMESA'.

Args:

query (str): Expressão regular para realizar a busca

Returns:

requests.Response: Retorna a resposta do TSWs

► Exemplo de uso:

```
log.setup('GSSLib_test')
tsws = TSWs.setup('dev')
query = '*CAMMESA'
response = tsws.get.tag_search(query)
```

► Resposta:

```
{
  "tagResponse": [
    {
      "tagName": "TSDB:CARGA CAMMESA"
    }
  ]
}
```

Funções disponíveis:

```
tsws.get.time_series_snapshot(tagName)
```

```
(method) time_series_snapshot(tagName: str) -> Response
```

Retorna o último valor registrado para a tag

Args:

tagName (str): Nome da tag que se deseja

Returns:

requests.Response: Resposta retornada pelo TSWS

► Exemplo de uso:

```
log.setup('GSSLib_test')  
tsws = TSWS.setup('dev')  
query = 'TSDB:CARGA CAMMESA'  
response = tsws.get.time_series_snapshot(query)
```

► Resposta:

```
{  
  "timeSeriesResponse": {  
    "engUnit": "MW",  
    "numericValue": 247053.799474581,  
    "tagName": "TSDB:CARGA CAMMESA",  
    "timestamp": "2023-02-07T14:06:47",  
    "value": "247.053,80MW"  
  }  
}
```

Funções disponíveis:

```
tsws.get.time_series_events(query, startTimestamp, endTimestamp)
```

(method) time_series_events(query: str, startTimestamp: datetime, endTimestamp: datetime) -> Response

Busca nos eventos do SCADA e retorna por valores que sejam compatíveis com a expressão contida no parâmetro 'query' entre os tempos determinados. O caracter '*' identifica que naquela posição podem aparecer quaisquer caracteres em qualquer quantidade, sendo um identificador universal. Por exemplo, para procurar por todos os eventos que envolvam a Unidade Geradora 01 a string query seria '*U01*'. Naturalmente, se a string fosse 'U01', apenas seriam retornados resultados que correspondessem exatamente a string (provavelmente nenhum). Porém, a inserção do character '*' indica que independente dos caracteres que apareçam nessa posição o resultado ainda será válido

Args:

query (str): A expressão que se deseja nos eventos

startTimestamp (datetime): Tempo de início para a busca de eventos

endTimestamp (datetime): Tempo final para a busca de eventos

► Exemplo de uso:

```
log.setup('GSSLib_test')
tsws = TSWS.setup('dev')
query = '*U01*'
endTimestamp = datetime.now()
startTimestamp = endTimestamp - timedelta(minutes=30)
response = tsws.get.time_series_events(query, startTimestamp, endTimestamp)
```

► Resposta:

```
{
  "timeSeriesResponse": [
    {
      "timestamp": "2023-02-16T14:39:26",
      "value": "16-FEB-2023 14:39:26 U01 TURBINA BOMBA 3 DRENAJE NC CONECT CLD-01 C4"
    },
    {
      "timestamp": "2023-02-16T14:46:01",
      "value": "16-FEB-2023 14:46:01 U01 TURBINA BOMBA 3 DRENAJE NC DESCON CLD-01 C4 RTN"
    }
  ]
}
```

Funções disponíveis:

```
tsws.get.time_series_interpolate(tagName, timestamp)
```

(method) time_series_interpolate(tagName: str, timestamp: datetime) -> Response

Retorna o valor da tag no tempo especificado, podendo ser uma interpolação dos valores registrados ou a repetição do último valor registrado, dependendo da configuração da tag.

Args:

tagName (str): Nome da tag que se deseja

timestamp (datetime): Tempo no qual o valor é desejado

Returns:

requests.Response: Resposta retornada pelo TSWS

► Exemplo de uso:

```
log.setup('GSSLib_test')
tsws = TSWS.setup('dev')
tagName = 'TSDB:CARGA CAMMESA'
timestamp = datetime.now()
response = tsws.get.time_series_interpolate(tagName, timestamp)
```

► Resposta:

```
{
  "timeSeriesResponse": {
    "engUnit": "MW",
    "numericValue": 247053.799474581,
    "tagName": "TSDB:CARGA CAMMESA",
    "timestamp": "2023-02-17T10:56:52",
    "value": "247.053,80MW"
  }
}
```

Funções disponíveis:

```
tsws.get.time_series_range(tagName, startTimestamp, endTimestamp)
```

```
(method) time_series_range(tagName: str, startTimestamp: datetime, endTimestamp: datetime) -> Response
```

Busca pelos valores registrados da tag entre o tempo solicitado. Os valores devolvidos nos exatos instantes 'start' ou 'end' não são valores registrados no tempo e sim, dependendo da configuração da tag, uma interpolação dos valores registrados ou a repetição do último valor registrado. Já os valores devolvidos entre o 'start' e 'end' são valores que foram registrados no banco de dados

Args:

tag (str): Nome da tag que se deseja obter os valores

startTimestamp (datetime): limite inferior do intervalo de tempo no qual será realizada a busca

endTimestamp (datetime): limite superior do intervalo de tempo no qual será realizada a busca

► Exemplo de uso:

```
log.setup('GSSLib_test')
tsws = TSWs.setup('dev')
tagName = 'TSDB:CARGA CAMMESA'
endTimestamp = datetime.now()
startTimestamp = endTimestamp - timedelta(hours=1)
response = tsws.get.time_series_range(tagName, startTimestamp, endTimestamp)
```

► Resposta:

```
"timeSeriesResponse": [
  {
    "engUnit": "MW",
    "numericValue": 247053.799474581,
    "tagName": "TSDB:CARGA CAMMESA",
    "timestamp": "2023-02-17T10:19:15",
    "value": "247.053,80MW"
  },
  {
    "engUnit": "MW",
    "numericValue": 247053.799474581,
    "tagName": "TSDB:CARGA CAMMESA",
    "timestamp": "2023-02-17T11:19:15",
    "value": "247.053,80MW"
  }
]
```

Funções disponíveis:

```
tsws.get.time_series_fixed_step_range(tagName, step, stepAmount,
```

```
(method) time_series_fixed_step_range(tagName: str, step: str, stepAmount: int,
startTimestamp: datetime, endTimestamp: datetime) -> Response
```

Retorna os valores referentes à tag entre os tempos solicitados com passo fixo, sendo que os valores retornados em cada ponto são um interpolação dos valores registrados ou a repetição do último valor registrado, dependendo da configuração da tag

Args:

tagName (str): Nome da tag que se deseja

step (str['SECOND' | 'MINUTE' | 'HOUR' | 'DAY' | 'MONTH' | 'YEAR']): Unidade referente ao passo

stepAmount (int): Valor numérico do passo

startTimestamp (datetime): Tempo inicial do intervalo

endTimeStamp (datetime): Tempo final do intervalo

► Exemplo de uso:

[illegible]

► **Resposta:**

```
"timeSeriesResponse": [
  {
    "engUnit": "MW",
    "numericValue": 247053.799474581,
    "tagName": "TSDB:CARGA CAMMESA",
    "timestamp": "2023-02-17T10:53:37",
    "value": "247.053,80MW"
  },
  {
    "engUnit": "MW",
    "numericValue": 247053.799474581,
    "tagName": "TSDB:CARGA CAMMESA",
    "timestamp": "2023-02-17T10:53:37",
    "value": "247.053,80MW"
  },
]
```

```
{
  "engUnit": "MW",
  "numericValue": 247053.799474581,
  "tagName": "TSDB:CARGA_CAMMESA",
  "timestamp": "2023-02-17T11:23:37",
  "value": "247.053,80MW"
}
```


Funções disponíveis:

```
tsws.get.time_series_aggregate(tagName, startTimestamp,
```

```
(method) time_series_aggregate(tagName: str, startTimestamp: datetime, endTimestamp: datetime, aggregationFunction: str) -> Response
```

Retorna o valor agregado para a tag entre os tempos solicitados

Args:

tagName (str): Nome da tag que se deseja

startTimestamp (datetime): tempo inicial para a agregação

endTimestamp (datetime): tempo final para a agregação

aggregationFunction (str['MAX' | 'MIN' | 'AVG' | 'INT' | 'COUNT']): Função que será utilizada para realizar a agregação

► Exemplo de uso:

```
log.setup('GSSLib_test')
tsws = TSWS.setup('dev')
tagName = 'TSDB:CARGA CAMMESA'
endTimestamp = datetime.now()
startTimestamp = endTimestamp - timedelta(hours=1)
aggregationFunction = 'AVG'
response = tsws.get.time_series_aggregate(tagName, startTimestamp,
                                          endTimestamp, aggregationFunction)
```

► Resposta:

```
{
  "timeSeriesResponse": {
    "aggregationFunction": "AVG",
    "endTimestamp": "2023-02-17T13:36:00.226459",
    "engUnit": "MW",
    "numericValue": 247053.799474581,
    "startTimestamp": "2023-02-17T12:36:00.226459",
    "tagName": "TSDB:CARGA CAMMESA",
    "value": "247.053,80MW"
  }
}
```


Funções disponíveis:

```
tsws.insert.time_series_add_value(tagName, timestamp, value)
```

(method) time_series_add_value(tagName: str, timestamp: datetime, value: Any) -> Response

Insere uma informação de tempo & valor para a tag especificada

Args:

tagName (str): Nome correspondente a tag

timestamp (datetime): Tempo associado ao valor a ser inserido

value (any): Valor que será inserido

► Exemplo de uso:

```
log.setup('GSSLib_test')
tsws = TWS.setup('dev')
tagName = 'TSDB:CARGA CAMMESA'
timestamp = datetime.now()
value = 20e3
response = tsws.insert.time_series_add_value(tagName, timestamp, value)
```

Funções disponíveis:

```
tsws.insert.time_series_add_value_list(tagName, timestamp, value)
```

(method) time_series_add_value_list(tagName: str, timestamp: List[datetime], value: List) -> Response

Insere várias informações de tempo & valor para a tag especificada

Args:

tagName (str): Nome correspondente a tag

timestamp (List[datetime]): lista dos tempos associados a cada valor

value (List[any]): Valores a serem inseridos

► Exemplo de uso:

```
log.setup('GSSLib_test')
tsws = TWS.setup('dev')
tagName = 'TSDB:CARGA CAMMESA'
now = datetime.now()
timestamp = [now, now - timedelta(minutes=30)]
value = [20e3, 21e3]
response = tsws.insert.time_series_add_value_list(tagName, timestamp, value)
```

Exemplo de aquisição de dados: nome da tag

- ▶ A função `get.tag(str)` retorna as tags que correspondem a string parâmetro

```
query = '*CAMMESA'  
response = tsws.get.tag_search(query)
```

- ▶ Os asteriscos presentes na string parâmetro de exemplo permitem a presença de quaisquer caracteres e em qualquer quantidade nas posições denotadas pelos asteriscos, dessa forma, qualquer nome de tag que contenha a palavra 'CAMMESA' no meio é considerada válida
- ▶ Após, checamos o status da resposta, que será 200 quando o procedimento for executado com sucesso, loggamos um erro e repetimos a requisição após um minuto se necessário.

```
while response.status_code != 200:  
    log.error(f'Waitin 1 min to search tag \'{query}\'' again')  
    time.sleep(60)  
    response = tsws.get.tag(query)
```

Exemplo de aquisição de dados: nome da TAG

► Resposta:

```
{
  "tagResponse": [
    {
      "tagName": "TSDB:CARGA CAMMESA"
    }
  ]
}
```

► Extraindo o campo "tagName":

```
tagName = response.json()['tagResponse'][0]['tagName']
```

Exemplo de aquisição de dados: Valores em um intervalo de tempo

- Definindo-se os valores do intervalo de tempo:

```
endTimeStamp = datetime.now()
startTimeStamp = endTimeStamp - timedelta(days=1)
```

- Requisição ao TSWS:

```
response = tsws.get.time_series_range(tagName, startTimestamp,
                                     endTimestamp)
```

Exemplo de aquisição de dados: Valores em um intervalo de tempo

► Resposta:

```
{
  "timeSeriesResponse": [
    {
      "engUnit": "MW",
      "numericValue": 203082.849757918,
      "tagName": "TSDB:CARGA CAMMESA",
      "timestamp": "2023-02-05T13:57:43",
      "value": "203.082,85MW"
    },
    {
      "engUnit": "MW",
      "numericValue": 203082.849757918,
      "tagName": "TSDB:CARGA CAMMESA",
      "timestamp": "2023-02-06T13:57:43",
      "value": "203.082,85MW"
    }
  ]
}
```

Exemplo de aquisição de dados: Valores em um intervalo de tempo

- ▶ Exemplo de iteração sobre os dados recebidos:

```
timeSeriesResponse = response.json()['timeSeriesResponse']
for data in timeSeriesResponse:
    time_value = datetime.fromisoformat(data['timestamp'])
    value = data['numericValue']
    print(f'O valor registrado na data {time_value} foi de {value}')
```

- ▶ Com esse código exibe-se no console:

```
O valor registrado na data 2023-02-05 13:57:43 foi de 203082.849757918
O valor registrado na data 2023-02-06 13:57:43 foi de 203082.849757918
```

Exemplo de inserção de dados: Inserindo um novo valor para determinada TAG

- Extraindo-se a média dos valores recebidos:

```
media = statistics.mean([data['numericValue'] for data in timeSeriesResponse])
```

- O valor inserido será uma variação aleatória em torno da média

```
novo_valor = media*(random.random()+0.5)
```

- Realizando a requisição para inserção:

```
response = tsws.insert.time_series_add_value(tagName, datetime.now(),
                                             value)
```

- Exibindo a resposta recebida:

```
print(response.text)
```

- Texto exibido no console:

Valores inseridos com sucesso: 1

Exemplo de aquisição de dados: último valor registrado

- ▶ Requisição para o último valor registrado:

```
response = tsws.get.time_series_snapshot(tagName)
```

- ▶ Resposta recebida:

```
{
  "timeSeriesResponse": {
    "engUnit": "MW",
    "numericValue": 208348.972588064,
    "tagName": "TSDB:CARGA CAMMESA",
    "timestamp": "2023-02-06T13:58:34",
    "value": "208.348,97MW"
  }
}
```