

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES
CENTRO NACIONAL DE MONITORAMENTO E ALERTAS DE DESASTRES NATURAIS

Plataforma de Entrega de Dados (PED)

MANUAL DO USUÁRIO

Versão 1.0 RC1

São José dos Campos – Novembro 2022

Plataforma de Entrega de Dados (PED)	1
Sumário	3
1. Formas de Acesso aos Serviços e Dados	3
1.1. Através dos Webservices	3
1.2. Através da Plataforma Web	5
2. Acesso aos Catálogos de Serviços	5
2.1. Catálogo de Serviços do SGAA	5
2.2. Catálogo de Serviços do SWS	9
2.2.1 Solicitação de Dados Históricos	15
2.2.2 Visualização do Status dos Agendamentos	19
3. Acesso via Código-Fonte	22
3.1. Java	22
3.2. Javascript	28
3.2.1 Recuperação do Token do Usuário	28
3.2.1.1 Recuperação do Token do Usuário usando AJAX	28
3.2.1.2 Recuperação do Token do Usuário usando Fech	29
3.2.1.3 Recuperação do Token do Usuário usando Axios	30
3.2.2 Exemplo de chamada a um Serviço Web	31
3.3. Python	33
3.3.1. Recuperação do Token do Usuário	33
3.3.2. Exemplo de chamada a um Serviço Web	34

Sumário

Este é o manual de uso para orientar o acesso aos dados disponibilizados pelos sistemas que compõem a **Plataforma de Entrega de Dados (PED)** do Cemaden.

Este documento é referente à versão 1.0 RC1 do PED, que será avaliada por um conjunto específico de usuários. Melhorias e novas funcionalidades serão constantemente agregadas aos sistemas.

1. Formas de Acesso aos Serviços e Dados

De forma a propiciar disponibilidade e escalabilidade, o PED foi projetado sob uma arquitetura robusta, composta por diversos sistemas, desde a recuperação, armazenamento e indexação dos dados até a entrega de dados personalizados para o usuário.

Principalmente por questões de limitação de infraestrutura, o PED contém uma camada de segurança gerida através do sistema do projeto chamado “*Sistema de Gerenciamento de Autenticação e Autorização (SGAA)*”.

Dessa forma, para realizar qualquer acesso aos dados e serviços do PED, o usuário deverá efetuar um cadastro no sistema, fornecendo um *e-mail* e uma *senha* válidos. É importante ressaltar que *o sistema não armazena nenhuma informação pessoal do usuário*, e todas as informações de acesso ao sistema são armazenadas de forma criptografada através de um método não-reversível. Assim, não existem considerações a serem feitas em relação a solicitação desses dados com a Lei Geral de Proteção de Dados Pessoais (LGPD).

Além disso, esse cadastro é efetuado exclusivamente pela Plataforma Web, que é outro sistema do PED, a ser disponibilizado em versões posteriores. Para esta versão do PED, os usuários (*e-mail* e *senha*) serão cadastrados diretamente na base de dados pelos responsáveis técnicos pelo projeto.

Abaixo estão dispostas as formas de acesso aos dados e serviços do PED.

1.1. Através dos Webservices

Uma das formas de solicitação dos dados disponibilizados pelo PED é diretamente através das chamadas dos webservices. Os webservices estão disponíveis no sistema do projeto chamado “*Sistema de WebServices (SWS)*”. Essa maneira de acessar os dados está disponível nesta versão.

Para acessar os dados diretamente através dos webservices, o usuário deverá possuir um **token** válido. Esse token é obtido através da chamada a um webservice do SGAA, informando como parâmetros o *e-mail* e *senha* do usuário. Dessa forma, após a obtenção do token, o usuário poderá utilizá-lo para a chamada dos demais webservices disponibilizados pelo PED.

Por questões de segurança, esse token é **individual** e **temporário**, e foi configurado para expirar depois de **4 horas** de sua criação. Após a expiração do token, para o usuário realizar novas requisições, este deverá obter um novo token da mesma forma mencionada anteriormente. Exemplos em código-fonte podem ser encontrados na seção 3 deste manual.

Na Figura 1 é possível visualizar o fluxo geral para realizar a chamada de um webservice disponibilizado no PED.

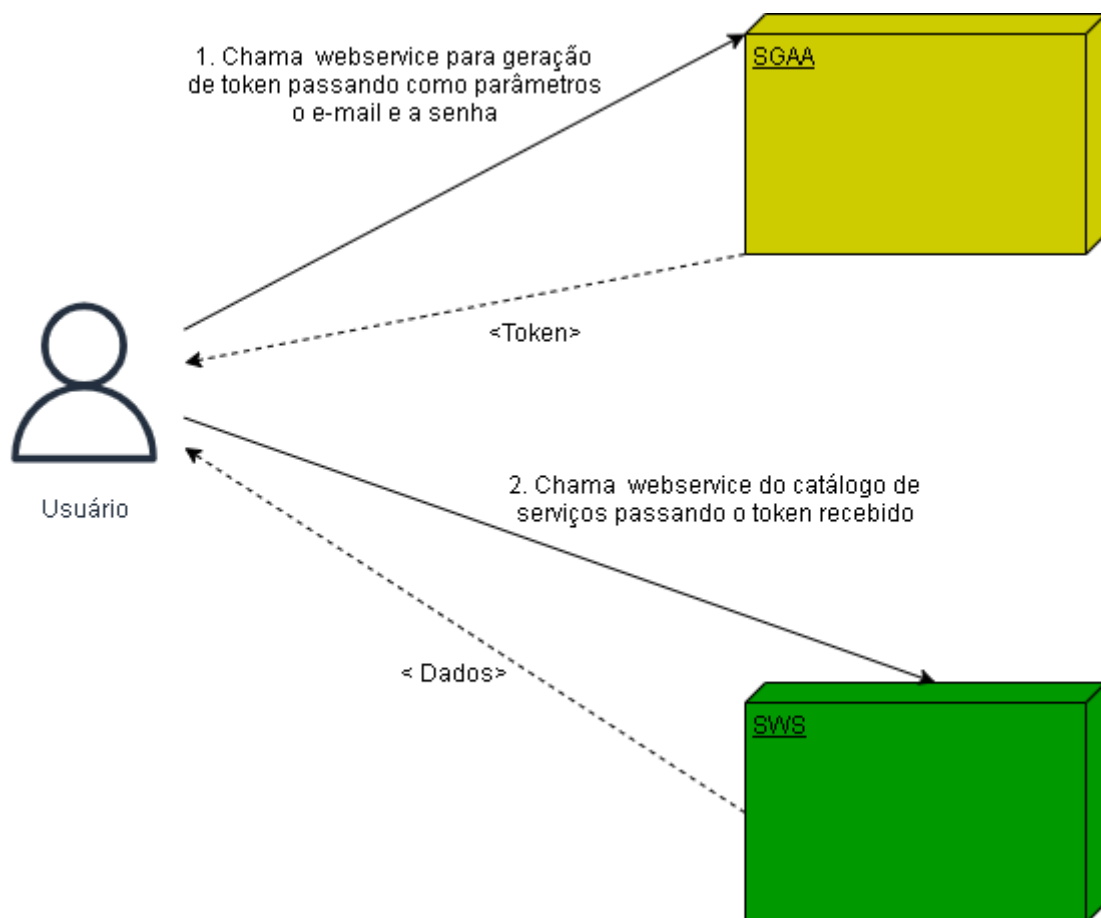


Figura 1 - Fluxo geral de acesso aos serviços disponibilizados pelo PED, com o acionamento dos sistemas responsáveis.

A interação do usuário com o webservice de solicitação de um novo token (item 1 da Figura 1) será melhor detalhada nas Figuras 3 e 4. Além disso, as Figuras 6 e 7 mostram um exemplo de chamada de webservice com o token recebido (item 2 da Figura 1).

Por questões de facilidade para o usuário, serão disponibilizados os *Catálogos de Serviços* do SGAA e SWS contendo a lista e a descrição de todos os serviços disponibilizados nessa versão. Através dos catálogos, o usuário poderá efetuar chamadas aos webservices e visualizar seus respectivos retornos. Os *Catálogos de Serviços* do SGAA e SWS serão apresentados na seção 2 deste manual.

1.2. Através da Plataforma Web

Um dos sistemas do PED consiste em uma *Plataforma Web*, que disponibilizará telas para o acesso do usuário aos dados e serviços disponibilizados pelo sistema. Essa plataforma **não está prevista nesta versão**, devendo ser disponibilizada em versões futuras.

Através dessa plataforma, realizado o cadastro, o usuário poderá efetuar o login no sistema, inserindo as suas credenciais de *e-mail* e *senha*. Após isso, o usuário poderá navegar pela plataforma e solicitar os dados através das telas disponibilizadas. Quando disponibilizada, a plataforma web se encarregará de realizar o controle do *token* de forma transparente para o usuário, em *background*, sem que o usuário precise realizar qualquer ação neste fluxo, além do login.

2. Acesso aos Catálogos de Serviços

O usuário poderá interagir com os webservices disponibilizados pelo PED através dos seus Catálogos de Serviços. Dois catálogos de serviço serão disponibilizados:

1. Catálogo de Serviços do **SGAA**.
2. Catálogo de Serviços do **SWS**.

2.1. Catálogo de Serviços do SGAA

Para que o usuário consiga executar os serviços de obtenção de dados, este precisará ter em mãos um token exclusivo e individual, que deverá ser obtido mediante a chamada a um serviço gerido pelo sistema SGAA.

Para se ter acesso aos serviços que são disponibilizados pelo sistema SGAA, deve-se acessar o seguinte endereço:

<http://sgaa.cemaden.gov.br>

A Figura 2 apresenta o Catálogo de Serviços do SGAA. Inicialmente, apenas o serviço de obtenção de tokens de acesso será disponibilizado. Demais serviços, em sua maioria de natureza administrativa, serão disponibilizados em versões posteriores.

Sistema de Gestão de Autenticação e Autorização (SGAA)

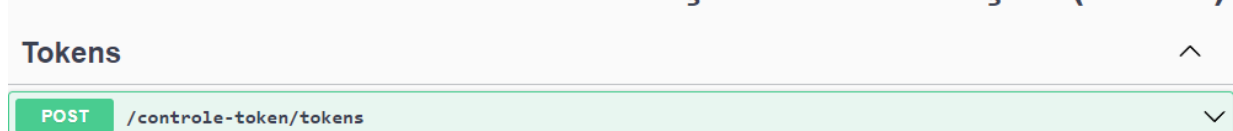


Figura 2 - Catálogo de Serviços do SGAA.

O webservice que está disponibilizado neste Catálogo de Serviços e que deverá ser utilizado pelo usuário para obtenção de um *token* de acesso é o seguinte:

Tokens	POST <u>/controle-token/tokens</u>
--------	---

POST /controle-token/tokens	
Web service para geração e entrega de um token para o usuário. Caso não haja nenhum token ativo para o usuário, este será criado. Caso contrário, ou seja, se já existir um token ativo para o usuário, este próprio token será apenas devolvido para o usuário.	
Parameters Try it out	
Name	Description
format string (query)	[Opcional] - Formato dos dados de saída. Valores possíveis: CSV, JSON, XML. Formato padrão: JSON. <div>format</div>
body object (body)	OBRIGATÓRIO - Objeto contendo as credenciais do usuário Example Value Model <div>{ "email": "...", "password": "..." }</div> Parameter content type <div>application/json</div>
Responses	
Code	Description
201	Response Resposta contendo as informações do token solicitado. Será retornado o tempo de expiração do token em milissegundos (campo "timeToExp") e uma string contendo o token (campo "token"). Example Value Model <div>{ "timeToExp": "604799", "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhdWQiOiIiLCJwYXNzd29yZCI6IjZkYS5tDeTZsaURoTWlOUV4Y0tQZ1FjMTRH ZFJR2RNNc1JBWFlibEh2MHc9IiwiaXNzIjoiyYnIuZ292LmNlbWFKZW4iLCJleHAiOjE2MzI4Mjc5Mjk5Imh0dCI6MTYzMjIyMzEyMzEyOjU0GVlYjItMDZlOC00NWQ1LWFjZTEtMMJhMGY4ZDgwZDQ4IiwidXN1cm5hbWUiOiJPRUttFOHU1d3M2MGpQTzZ4UGQ1RzdITk1BSVQxZGdFV3Y4L0 1jTTVMOTJvPSJ9.oDFycUF4oSp1e2UmrjFq5zAZb5ptKk8Fxli19dixQ" }</div>

6

Nesta tela, é possível clicar sobre o botão “*Try it out*”, onde serão desbloqueados os campos para o usuário informar os parâmetros para realização da chamada do webservice.

A Figura 4 apresenta o resultado do clique sobre o botão “*Try it out*” da tela do webservice */controle-tokens/tokens*, bem como o preenchimento dos parâmetros exigidos neste serviço.

Figura 4 - Exemplo de chamada do webservice */controle-tokens/tokens* do Catálogo de Serviços do SGAA.

Através dessa tela é possível solicitar um token passando como parâmetros o *e-mail* e *senha* do usuário, conforme mostrado na Figura 4. Para isso, basta o usuário preencher os campos solicitados e clicar sobre o botão “**Execute**”. Além disso, o Catálogo de Serviços também disponibiliza o *download* dos dados, através do clique sobre o botão “**Download**”, conforme apresentado na Figura 4.

Além disso, na Figura 4 também é possível verificar o retorno do webservice a esta chamada. Trata-se de um objeto **JSON** contendo os atributos ***timeToExp*** e ***token***. O atributo *token* será diretamente utilizado nas chamadas dos demais webservices, enquanto que o atributo *timeToExp* indica o tempo restante para expiração do token em segundos.

Conforme mencionado anteriormente, em posse desse token temporário, o usuário poderá realizar a chamada aos demais webservices disponíveis no Catálogo do SWS. Para tanto, basta o usuário informá-lo no parâmetro ***token*** presente no **header** da requisição. Exemplos de como fazer isso através de programação são apresentados na seção 3 deste documento.

Por exemplo, se a *Server response* for:

Tabela 2 - Resposta do SGAA.

Code	Details
200	Response body
	<pre>{ "timeToExp": "600000", "token": "aG9tZSBvZmZpY2U" }</pre>
	Response headers
	connection: keep-alive content-length: 82 content-type: application/json date: Sun, 1 Jan 2022 00:00:00 GMT

Neste caso, o ***token*** será ***aG9tZSBvZmZpY2U***, e, no caso do uso manual, deve-se copiá-lo sem as aspas.

2.2. Catálogo de Serviços do SWS

Para se ter acesso aos serviços que são disponibilizados pelo sistema SWS, deve-se acessar o seguinte endereço:

<http://sws.cemaden.gov.br>

Os webservices que serão disponibilizados, e que estão presentes neste Catálogo de Serviços, são os seguintes:

Tabela 3 - Serviços do SWS disponibilizados.

Acumulados	GET <u>/pcds-acum/acumulados-historicos</u>
	GET <u>/pcds-acum/acumulados-recentes</u>
Dados Ambientais	GET <u>/pcds/dados_pcd</u>
	GET <u>/pcds/dados_rede</u>
	GET <u>/pcds/pcds-dados-atualizado-pg</u>
	GET <u>/pcds/pcds-dados-pg</u>
	GET <u>/pcds/pcds-dados-recentes</u>
Dados Cadastrais da Rede Observacional	GET <u>/pcds-cadastro/cidades</u>
	GET <u>/pcds-cadastro/dados-cadastrais</u>
	GET <u>/pcds-cadastro/estacoes</u>
Paginação dos Dados	GET <u>/controle-paginacao/pagina</u>
Requisições Agendadas	GET <u>/controle-agendamento/agendamentos</u>
	GET <u>/controle-agendamento/pcds-dados-historicos</u>
Tipos de Estações	GET <u>/pcds-tipo-estacao/sensores</u>
Conversor de Dados	GET <u>/modelo/formatos</u>

A Figura 5 apresenta o Catálogo de Serviços do SWS que está disponível para o usuário.



Plataforma de Entrega de Dados (PED)

Acumulados ^

GET /pcds-acum/acumulados-historicos v

GET /pcds-acum/acumulados-recentes v

Conversor de Dados ^

GET /modelo/formatos v

Dados Ambientais ^

GET /pcds/dados_pcd v

GET /pcds/dados_rede v

GET /pcds/pcds-dados-atualizado-pg v

GET /pcds/pcds-dados-pg v

GET /pcds/pcds-dados-recentes v

Dados Cadastrais da Rede Observacional ^

GET /pcds-cadastro/cidades v

GET /pcds-cadastro/dados-cadastrais v

GET /pcds-cadastro/estacoes v

Paginação dos Dados ^

GET /controle-paginacao/pagina v

Requisições Agendadas ^

GET /controle-agendamento/agendamentos v

GET /controle-agendamento/pcds-dados-historicos v

Tipos de Estações ^

GET /pcds-tipo-estacao/sensores v

Figura 5 - Catálogo de Serviços do SWS.

Ao clicar no botão correspondente ao webservice desejado, esta seção será expandida e será exibido um formulário de consulta para o usuário. A Figura 6 apresenta a tela que será exibida quando o usuário clicar sobre o webservice */pcds/pcds-dados-recentes* do Catálogo de Serviços do SWS, enquanto que a Figura 7 exibe a chamada deste webservice após o usuário clicar sobre o botão *“Try it out”*, preencher os dados e, posteriormente, clicar sobre o botão *“Execute”*.

Deve-se lembrar que o **token** obtido seguindo as instruções da seção 2.1 deverá ser inserido como primeiro parâmetro em todos os serviços disponíveis no Catálogo do SWS.



GET /pcds/pcds-dados-recentes

Webservice para recuperação dos dados ambientais recentes das PCDs, referentes às últimas 3 horas, dado um conjunto de parâmetros.

Parameters Try it out

Name	Description
token string (header)	OBRIGATÓRIO - JWT Token, recuperado pelo webservice /controle-token/tokens. <div>token - OBRIGATÓRIO - JWT Token, recupe</div>
codestacao string (query)	[Opcional] - Código da estação no Cemaden. Ex.: 292740804A. <div>codestacao - [Opcional] - Código da estação</div>
codibge string (query)	[Opcional] - Código do município no IBGE. Ex.: 2927408. <div>codibge - [Opcional] - Código do município n</div>
formato string (query)	[Opcional] - Formato dos dados de saída. Valores possíveis: CSV, JSON, XML. Formato padrão: JSON. <div>formato - [Opcional] - Formato dos dados de</div>
rede string (query)	OBRIGATÓRIO - Identificador da rede observacional. Ex.: 11. <div>rede - OBRIGATÓRIO - Identificador da rede</div>
sensor string (query)	[Opcional] - Identificador do sensor da estação no Cemaden. Ex.: 10. <div>sensor - [Opcional] - Identificador do sensor (</div>
tipoestacao string (query)	[Opcional] - Identificador do tipo de estação no Cemaden. Ex.: 1. <div>tipoestacao - [Opcional] - Identificador do tip</div>
uf string (query)	OBRIGATÓRIO - Unidade Federativa (UF). Ex.: BA. <div>uf - OBRIGATÓRIO - Unidade Federativa (UI</div>

Responses

Code	Description
200	<p>Lista com os dados ambientais das últimas 3 horas das PCDs, correspondente aos parâmetros de busca informados, ordenada por datahora de forma crescente.</p> <p>Example Value Model</p> <pre>[{ "cidade": "CARANDAI", "codestacao": "311320601A", "datahora": "2021-11-30 14:10", "id_sensor": 240, "latitude": -20.953, "longitude": -43.805, "nome": "Centro2 MG", "offset": null, "qualificacao": "0000", "uf": "MG", "valor": 0 }, {</pre>

Figura 6 - Tela do Catálogo de Serviços do SWS referente ao webservice /pcds/pcds-dados-recentes.

Figura 7 - Exemplo de chamada do webservice */pcds/pcds-dados-recentes* do Catálogo de Serviços do SWS.

Ao realizar a chamada ao webservice */pcds/pcds-dados-recentes* através do Catálogo de Serviços do SWS, o usuário poderá introduzir os parâmetros solicitados pelo webservice, bem como o *token* de segurança que deverá ter sido gerado previamente. Note que na Figura 7 foi informado um token no parâmetro “token” (*header*), gerado previamente conforme mostra a Figura 4.

Observação: O parâmetro **codibge** poderá ser obtido a partir da consulta ao website do IBGE (<https://www.ibge.gov.br/explica/codigos-dos-municipios.php>) ou utilizando o serviço [/pcds-cadastro/cidades](#), acessível a partir da seção *Dados Cadastrais da Rede Observacional* (ver figura 5) do Catálogo de Serviços do SWS.

2.2.1 Solicitação de Dados Históricos

Para solicitar os dados históricos disponibilizados pelo PED, o usuário deverá utilizar o serviço correspondente no Catálogo de Serviços do SWS.

Na Figura 8 abaixo está indicado o serviço que o usuário deverá acionar para requisitar os dados desejados.

Plataforma de Entrega de Dados (PED)

Acumulados

GET /pcds-acum/acumulados-historicos

GET /pcds-acum/acumulados-recentes

Conversor de Dados

GET /modelo/formatos

Dados Ambientais

GET /pcds/dados_pcd

GET /pcds/dados_rede

GET /pcds/pcds-dados-atualizado-pg

GET /pcds/pcds-dados-pg

GET /pcds/pcds-dados-recentes

Dados Cadastrais da Rede Observacional

GET /pcds-cadastro/cidades

GET /pcds-cadastro/dados-cadastrais

GET /pcds-cadastro/estacoes

Paginação dos Dados

GET /controle-paginacao/pagina

Requisições Agendadas

GET /controle-agendamento/agendamentos

GET /controle-agendamento/pcds-dados-historicos

Tipos de Estações

GET /pcds-tipo-estacao/sensores

Figura 8 - Indicação do serviço para solicitação de dados históricos no Catálogo de Serviços do SWS.

Após o usuário acionar esse serviço será aberto um formulário com vários campos disponíveis para pesquisa, que irão compor os parâmetros do webservice que será chamado pelo sistema, conforme é possível visualizar na Figura 9.

Nesta tela, é possível clicar sobre o botão “**Try it out**”, onde serão desbloqueados os campos para o usuário informar os parâmetros para realização da chamada do webservice. Neste formulário será necessário informar o token obtido pelo usuário, no campo correspondente e os campos que irão compor a pesquisa. É possível observar que alguns campos são de preenchimento obrigatório, enquanto outros são opcionais. Após o usuário clicar no botão “**Execute**” será exibido no campo “**Response body**” do formulário o resultado dessa consulta pelo sistema.

No caso desse serviço em particular, o que será exibido como resposta será um identificador numérico da consulta (campo **id** indicado pela seta vermelha na Figura 9), que poderá ser identificado posteriormente em outro webservice responsável pela busca dos agendamentos efetuados pelo usuário, que será explicado em maiores detalhes na próxima seção.

Figura 9 - Tela do Catálogo de Serviços do SWS referente ao webservice [/controle-agendamento/pcds-dados-historicos](#), com um exemplo de chamada ao serviço e indicação do campo identificador (“id”) da consulta.

Ainda referente à Figura 9, de acordo com a descrição deste webservice, o usuário poderá optar pelo seu arquivo ser entregue com os dados em um de quatro formatos possíveis (campo **arquivo** do formulário), sendo estes dispostos abaixo:

1. JSON;
2. CSV;
3. XML;
4. CUSTOMIZADO.


Os três primeiros formatos (JSON, CSV e XML) são padrões internacionais de entrega de dados. O quarto formato referenciado acima, nomeado como CUSTOMIZADO, refere-se ao padrão de dados definido pelo Cemaden. Esse formato de dados é semelhante ao CSV, porém dispõe de cabeçalho, de outro separador de colunas e de casas decimais, dentre outras diferenças, sendo esse o motivo de ter sido nomeado como CUSTOMIZADO.


2.2.2 Visualização do Status dos Agendamentos

O usuário poderá realizar uma ou múltiplas solicitações de dados históricos através do serviço [/controle-agendamento/pcds-dados-historicos](#) disponibilizado pelo Catálogo de Serviços do SWS. Cada uma dessas solicitações será agendada e posteriormente atendida pelo *Sistema de Gerenciamento de Requisições Agendadas (SGRA)*. O usuário deverá utilizar o serviço [/controle-agendamento/agendamentos](#) para verificar o status dos agendamentos que realizou, conforme indicado na Figura 10.


Plataforma de Entrega de Dados (PED)

Acumulados


GET /pcds-acum/acumulados-historicos 

GET /pcds-acum/acumulados-recentes 


Conversor de Dados

GET /modelo/formatos 


Dados Ambientais

GET /pcds/dados_pcd 

GET /pcds/dados_rede 


GET /pcds/pcds-dados-atualizado-pg 


GET /pcds/pcds-dados-pg 

GET /pcds/pcds-dados-recentes 


Dados Cadastrais da Rede Observacional

GET /pcds-cadastro/cidades 


GET /pcds-cadastro/dados-cadastrais 


GET /pcds-cadastro/estacoes 

Paginação dos Dados

GET /controle-paginacao/pagina 

Requisições Agendadas

GET /controle-agendamento/agendamentos 

GET /controle-agendamento/pcds-dados-historicos 

Tipos de Estações


GET /pcds-tipo-estacao/sensores 

Figura 10 - Indicação do serviço para visualização dos agendamentos de solicitação de dados históricos no Catálogo de Serviços do SWS.

GET

/controle-agendamento/agendamentos

↗

WebService para recuperação das requisições agendadas do usuário, dado um conjunto de parâmetros.

Parameters

Cancel

Name	Description
token string (header)	OBRIGATÓRIO - JWT Token, recuperado pelo webservice /controle-token/tokens. <div>eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ</div>
formato string (query)	[Opcional] - Formato dos dados de saída. Valores possíveis: CSV, JSON, XML. Formato padrão: JSON. <div>JSON</div>
statusreq string (query)	[Opcional] - Status das requisições agendadas. Valores possíveis: PENDENTE, CONCLUIDA, REJEITADA, EXPIRADA. <div>PENDENTE</div>

Execute

Clear

Responses

Code	Details
200	<div>Response body</div> <div><pre>[{ "dtCreate": "2022-11-01 12:47:07", "dtLastUpdate": "2022-11-01 12:47:07", "id": 219, "link": null, "parameters": "arquivo=XML&datafim=202102010000&datainicio=202101010000&formato=CSV&rede=11&uf=AC", "service": { "id": 2, "method": "GET", "path": "/PED/rest/controle-agendamento/pcds-dados-historicos", "description": "WebService para agendamento de requisição por dados ambientais históricos das PCDs, dado um conjunto de parâmetros.", "system": { "id": 2, "name": "SMS", "description": "Sistema de WebServices" } }, "status": { "id": 1, "description": "PENDENTE" } }]</pre></div> <div><div>Download</div></div>

Response headers

```
access-control-allow-credentials: false
access-control-allow-headers: token,accept,content-type
access-control-allow-methods: GET
access-control-allow-origin: http://localhost:8080/
access-control-max-age: 1
connection: keep-alive
content-encoding: gzip
content-length: 364
content-type: application/json
date: Tue,01 Nov 2022 13:47:38 GMT
```

Figura 11 - Tela do Catálogo de Serviços do SWS referente ao webservice */controle-agendamento/agendamentos*, com um exemplo de chamada ao serviço.

Nesse serviço, o usuário deverá fornecer o status de interesse no campo **statusreq**, bem como o seu token no campo **token**. Após clicar no botão **“Execute”**, serão exibidos os agendamentos feitos pelo usuário no campo **“Response body”**. No exemplo da Figura 11, o usuário consultou os agendamentos com o status = “PENDENTE”. Caso o usuário queira visualizar todos os seus agendamentos, basta ele realizar a consulta sem preencher o campo de status.

Os dados solicitados pelo usuário serão disponibilizados através de um arquivo, cujo link está disponível no campo **link** de retorno do webservice, conforme é possível visualizar no recorte da Figura 12.

```
{
  "dtCreate": "2022-10-21 14:01:34",
  "dtLastUpdate": "2022-10-21 14:16:25",
  "id": 2,
  "link": "http://sgra.cemaden.gov.br/pedidos/PED_Cemaden_2e4c8b77b1af4699962852ba6f0b5fdf.zip",
  "parameters": "datafim=202102010000&datainicio=202101010000&rede=11&uf=sp",
  "service": {
    "id": 2,
    "method": "GET",
    "path": "/PED/rest/controle-agendamento/pcds-dados-historicos",
    "description": "Webservice para agendamento de requisição por dados ambientais históricos das PCDs, dado um conjunto de parâmetros.",
    "system": {
      "id": 2,
      "name": "SWS",
      "description": "Sistema de WebServices"
    }
  },
  "status": {
    "id": 2,
    "description": "CONCLUIDA"
  }
}
```

Figura 12 - Exemplo de retorno do serviço de consulta de agendamentos de dados históricos, cuja seta vermelha indica o link com o arquivo gerado para *download*.

No exemplo da Figura 12 também é possível visualizar os parâmetros que o usuário utilizou para solicitar os dados históricos, bem como a data e horário em que o pedido foi realizado. Importante notar que os arquivos solicitados ficarão disponíveis para download por um período de 30 dias.

3. Acesso via Código-Fonte

Esta seção apresenta exemplos para a criação de um **sistema automatizado** para consumir os webservices disponíveis no PED. Inicialmente, para esta versão, este manual contempla exemplos nas linguagens *Java* (seção 3.1), *JavaScript* (seção 3.2) e *Python* (seção 3.3).

3.1. Java

Nesta subseção serão apresentados exemplos de código-fonte em Java para que o usuário possa realizar, de forma automatizada, o fluxo de comunicação descrito na Figura 1.

Iniciaremos com a definição das classes que serão utilizadas no *request* e no *response* do webservice POST */controle-tokens/tokens*, que gera um *token* para o usuário, dado um *e-mail* e *senha*.

O Código 1 representa a classe que será utilizada para compor o *body* da requisição HTTP para solicitação de um novo token, enquanto o Código 2 representa a classe que será utilizada para recuperar o retorno do webservice em caso de status 200 - OK.

Código 1 - Classe utilizada no *request* do webservice POST */controle-tokens/tokens*.

```
1  package test;
2
3  public class User {
4
5      private String email;
6
7      private String password;
8
9      public User(){
10         // This constructor intentionally left blank.
11     }
12
13     public User(String email, String password) {
14         this.email = email;
15         this.password = password;
16     }
17
18     public String getEmail() {
19         return email;
20     }
21
22     public void setEmail(String email) {
23         this.email = email;
24     }
25
26     public String getPassword() {
27         return password;
28     }
29
30     public void setPassword(String password) {
31         this.password = password;
32     }
33 }
```

Código 2 - Classe utilizada no *response* do webservice POST */controle-tokens/tokens*.

```
1  package test;
2
3  public class Token {
4
```

```
5     private String token;
6
7     private String timeToExp;
8
9     public Token(){
10         // This constructor intentionally left blank.
11     }
12
13     public Token(String token, String timeToExp){
14         this.token = token;
15         this.timeToExp = timeToExp;
16     }
17
18     public String getToken() {
19         return token;
20     }
21
22     public void setToken(String token) {
23         this.token = token;
24     }
25
26     public String getTimeToExp() {
27         return timeToExp;
28     }
29
30     public void setTimeToExp(String timeToExp) {
31         this.timeToExp = timeToExp;
32     }
33 }
```

Dado isso, o Código 3 apresenta uma classe que realiza chamadas POST e GET genéricas, dado uma URL, um mapa com os parâmetros do *header*, e, no caso do POST, uma entidade a ser passada no *body* da requisição.

Código 3 - Classe utilizada para a realização de chamadas POST e GET.

```
1  package test;
2
3  import javax.ws.rs.client.Client;
4  import javax.ws.rs.client.ClientBuilder;
5  import javax.ws.rs.client.Entity;
6  import javax.ws.rs.client.WebTarget;
7  import javax.ws.rs.core.MediaType;
8  import javax.ws.rs.core.MultivaluedMap;
9  import javax.ws.rs.core.Response;
10
11  public class GenericRestClient<T> {
12
13      public Response doPost(String url, T entity,
14          MultivaluedMap<String, Object> headers){
15
16      }
```



```
16      Client client = ClientBuilder.newClient();
17      WebTarget target = client.target(url);
18      return target.
19          request(MediaType.APPLICATION_JSON).
20          headers(headers).
21          post(Entity.json(entity));
22  }
23
24  public Response doGet(String url,
25      MultivaluedMap<String, Object> headers){
26
27      Client client = ClientBuilder.newClient();
28      WebTarget target = client.target(url);
29      return target.
30          request(MediaType.APPLICATION_JSON).
31          headers(headers).
32          get();
33  }
34 }
```

Feito esse preâmbulo, com a definição das classes acessórias a serem utilizadas na chamada do webservice `/controle-tokens/tokens` do SGAA, o Código 4 apresenta uma classe para a recuperação dos dados desejados no SWS.

Código 4 - Classe utilizada para recuperação dos dados desejados.

```
1  package test;
2
3  import java.io.IOException;
4  import java.io.InputStream;
5  import java.util.LinkedHashMap;
6  import java.util.List;
7
8  import javax.ws.rs.core.MultivaluedHashMap;
9  import javax.ws.rs.core.MultivaluedMap;
10 import javax.ws.rs.core.Response;
11 import javax.ws.rs.core.Response.Status;
12
13 import com.fasterxml.jackson.core.exc.StreamReadException;
14 import com.fasterxml.jackson.databind.DatabindException;
15 import com.fasterxml.jackson.databind.ObjectMapper;
16
17 public class PEDConnection {
18
19     private static String token;
20
21     public static void test() throws StreamReadException,
22     DatabindException, IOException {
23
24         Response response =
25         getWSResponse("http://sws.cemaden.gov.br/PED/rest/pcds-acum/acumulado
```

```
s-recentes?codibge=3550308");
24     InputStream inputStream = (InputStream) response.getEntity();
25
26     ObjectMapper om = new ObjectMapper();
27     List<LinkedHashMap<?, ?>> jsonList =
om.readValue(inputStream, List.class);
28     inputStream.close();
29
30     for(LinkedHashMap<?, ?> json: jsonList)
31         System.out.println(json.toString());
32 }
33
34 private static <T> Response getWSResponse(String url){
35
36     GenericRestClient<T> genericRestClient = new
GenericRestClient<>();
37     //Prepara o header da requisição
38     MultivaluedMap<String, Object> headers = new
MultivaluedHashMap<>();
39     headers.putSingle("token", token);
40     //Efetua a requisição com o token corrente
41     Response response = genericRestClient.doGet(url, headers);
42
43     //Caso o token não esteja mais válido (esteja em branco ou
expirado) e o SGAA retorne status 400 (BAD_REQUEST) ou 401
(UNAUTHORIZED) indicando isso
44     if((response.getStatus() ==
Status.UNAUTHORIZED.getStatusCode()) ||
45         (response.getStatus() ==
Status.BAD_REQUEST.getStatusCode())) {
46
47         //Então solicita um novo token ao SGAA
48         token = getNewToken();
49         //Prepara o header da requisição, agora com o novo token
50         headers.putSingle("token", token);
51         //Efetua a requisição com o novo token
52         response = genericRestClient.doGet(url, headers);
53
54         //Insira aqui tratamentos adicionais
55     }
56
57     //Insira aqui tratamentos adicionais
58
59     return response;
60 }
61
62 private static String getNewToken() {
63
64     String url =
"http://sgaa.cemaden.gov.br/SGAA/rest/controle-token/tokens";
65     User user = new User ("EMAIL", "SENHA");
66
```

```
67         GenericRestClient<User> genericRestClient = new
GenericRestClient<>();
68         Response response = genericRestClient.doPost(url, user,
null);
69         InputStream inputStream = (InputStream) response.getEntity();
70
71         ObjectMapper om = new ObjectMapper();
72         Token tokenObj = null;
73         try {
74             tokenObj = om.readValue(inputStream, Token.class);
75             inputStream.close();
76         } catch (IOException e) {
77             //Insira aqui seu tratamento da exceção
78         }
79
80         //Insira aqui tratamentos adicionais
81
82         return tokenObj.getToken();
83     }
84 }
```

Primeiramente, é importante ressaltar que antes de executar o Código 4, deve-se substituir as chaves EMAIL e SENHA pelos seus valores reais.

Dessa forma, o método *getNewToken* do Código 4 (linha 62) apresenta um exemplo de como pode ser solicitado ao SGAA um novo token para o usuário, dado suas credenciais de e-mail e senha.

Com o token em mãos, o usuário poderá efetuar o acesso aos dados e serviços do SWS. O método *getWSResponse* do Código 4 (linha 34) realiza a chamada de um webservice GET qualquer passando o *token* recuperado como um parâmetro do *header*. Note que as linhas 38 e 39 preparam o *header* a ser utilizado na chamada de um webservice do Catálogo de Serviços do SWS. Assim, na linha 39, é possível observar a inserção do parâmetro *token* no objeto que representa o *header* da requisição. Esse token é o token corrente do usuário, recuperado anteriormente.

Devido ao caráter temporário do token, pode ser que o mesmo tenha expirado em um determinado momento, e, nesse caso, será retornado pelo webservice o status 401 ("Unauthorized"). Além disso, caso o token esteja em branco, o webservice irá retornar status 400 ("Bad Request"). Nesses casos, o usuário deverá solicitar um novo token ao sistema, conforme mostra a linha 48, através da chamada do método *getNewToken*.

A chamada do método *getWSResponse* poderá ser realizada conforme apresentado no método *test* do Código 4 (linha 21). A título de exemplo, vamos considerar a recuperação do *response* do webservice */pcds-acum/acumulados-recentes* (linha 23) e a deserialização do mesmo em caso de sucesso - status 200 (linha 27). Como o parâmetro *formato* não foi informado, os dados serão recuperados no formato padrão, que é em JSON. As linhas 30 e 31 mostram um exemplo de como os dados recuperados podem ser impressos.

Assim, com os códigos-fonte apresentados anteriormente, foi possível apresentar um exemplo de como realizar o fluxo de comunicação da Figura 1 na linguagem Java.

3.2. Javascript

Nessa seção iremos mostrar para o usuário como implementar o fluxo de comunicação descrito na Figura 1, de forma similar ao que foi mostrado na seção 3.1, porém, utilizando para isso código-fonte em Javascript.

É importante mencionar que os procedimentos para chamar os webservices disponibilizados pela plataforma, e que foram apresentadas neste manual, não deveriam ser executados em um navegador através de um código cliente (*frontend*), mas sim executadas em um servidor em *backend* devido à questões de segurança (**CORS** - *Cross-origin Resource Sharing*). O **CORS** é um mecanismo utilizado pelos navegadores para compartilhar recursos entre diferentes origens. Trata-se de uma especificação do W3C que faz uso de *headers* do HTTP para informar aos navegadores se determinado recurso pode ser ou não acessado. Isso acontece devido a um mecanismo de segurança presente nos navegadores chamado de *same-origin policy*, que é usado para limitar como um documento ou script de uma origem pode interagir com recursos de outra origem. Para maiores informações, consulte em: https://www.w3.org/wiki/CORS_Enabled.

3.2.1 Recuperação do Token do Usuário

O primeiro passo é efetuar uma chamada ao serviço de entrega de token que será utilizado nas chamadas dos serviços de dados. Nesta seção serão apresentadas três maneiras de se conectar ao serviço de entrega de tokens:

1. Utilizando a tecnologia *Ajax*;
2. Utilizando a API *Fetch*;
3. Utilizando a biblioteca do *Axios*.

3.2.1.1 Recuperação do Token do Usuário usando AJAX

No Código 5 podemos observar um trecho de código que realiza a chamada ao serviço de obtenção de token (*/SGAA/rest/controle-token/tokens*) utilizando a tecnologia AJAX. O usuário deverá realizar um cadastro prévio na plataforma. De posse de um *email* e *password* válidos, ele então deverá inserir estes parâmetros conforme indicado.

Código 5 - Método de recuperação do token do usuário utilizando a tecnologia AJAX.

```
1  const urlPrefix =  
2  'http://sgaa.cemaden.gov.br/SGAA/rest/controle-token/tokens';  
3  
4  let token = '';  
5  
6  let credencial = {  
7    email: 'EMAIL',  
8    password: 'SENHA'  
9  }
```

```
10
11 token = obterToken(credencial);
12 //Função utilizada para obtenção de token a partir das credenciais do
13 usuário
14 function obterToken(params) {
15     $.ajax({
16         url: urlPrefix,
17         type: 'post',
18         contentType: "application/json",
19         data: JSON.stringify(params),
20         success: function (data, status) {
21             console.log("Success!!");
22             console.log(data);
23             console.log(status);
24             token = data.token;
25         },
26         error: function (xhr, desc, err) { console.log(xhr);
27             console.log("Desc: " + desc + "\nErr:" + err);
28         }
29     });
30     return token;
31 }
32
33 console.log(token);
34 alert(token);
```

A função *obterToken* irá realizar a comunicação com o servidor no qual o webservice está implantado e, dessa forma, irá devolver ao usuário o token.

3.2.1.2 Recuperação do Token do Usuário usando *Fetch*

No Código 6 é apresentado um código com a mesma finalidade que o código exibido na subseção anterior, porém utilizando a API *Fetch*.

Código 6 - Método de recuperação do token do usuário utilizando a API *Fetch*.

```
1 const urlPrefix =
2   'http://sgaa.cemaden.gov.br/SGAA/rest/controle-token/tokens';
3
4 let token = '';
5
6 let credencial = {
7     email: 'EMAIL',
8     password: 'SENHA'
9 }
10
11 token = obterToken(credencial);
12 //Função utilizada para obtenção de token a partir das credenciais do
13 usuário
14 function obterToken(params) {
15     fetch(urlPrefix, {
16         method: 'POST', // or 'PUT'
```

```
17     headers: {
18         'Accept': 'application/json',
19         'Content-Type': 'application/json',
20     },
21     body: JSON.stringify(params),
22 }).then(response => response.json())
23 .then(data => {
24     console.log('Success:', data);
25     token = data.token;
26 })
27 .catch((error) => {
28     console.error('Error:', error);
29 });
30
31 return token;
32 }
33
34 console.log(token);
35 alert(token);
```

3.2.1.3 Recuperação do Token do Usuário usando *Axios*

Por fim, no Código 7 é exibido um trecho de código que faz uso da biblioteca *Axios* para obter o token para o usuário.

Código 7 - Método de recuperação do token do usuário utilizando a biblioteca *Axios*.

```
1  const urlPrefix =
2  'http://sgaa.cemaden.gov.br/SGAA/rest/controle-token/tokens';
3
4  let token = '';
5
6  credencial = {
7      email: 'EMAIL',
8      password: 'SENHA'
9  }
10
11  token = obterToken(credencial);
12  //Função utilizada para obtenção de token a partir das credenciais do
13  usuário
14  function obterToken(params) {
15      axios({
16          method: 'post',
17          url: urlPrefix,
18          data: params
19      })
20      .then(response => {
21          console.log(response);
22          token = response.data.token;
23      })
24      .catch(error => {
25          console.log(error);
```

```
26     })
27
28     return token;
29 }
    console.log(token);
    alert(token);
```

Para utilizar a biblioteca do *Axios* é necessário instalar o pacote correspondente. Isto poderá ser realizado através de uma das formas abaixo:

Utilizando *npm*:

```
$ npm install axios
```

Utilizando *bower*:

```
$ bower install axios
```

Utilizando *yarn*:

```
$ yarn add axios
```

Caso opte por utilizar o CDN, terá de incluí-lo na tag *head* no arquivo html.

Exemplo:

```
<head>
  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
</head>
```

ou importar o *axios* CDN utilizando a tag *script*. Por exemplo:

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js">
</script>
```

3.2.2 Exemplo de chamada a um Serviço Web

De posse do token, o usuário poderá realizar as chamadas aos serviços web que necessita. Abaixo, o Código 8 apresenta como exemplo a chamada a um serviço web que lista as cidades que possuem estações pertencentes à rede observacional do Cemaden, dado uma unidade de federação (UF):

Código 8 - Exemplo de chamada a um serviço que recupera as cidades pertencentes à rede observacional do Cemaden dado uma UF. Obs: Este exemplo utiliza a tecnologia *Ajax*.

```
1  const urlPrefix =
2  'http://sws.cemaden.gov.br/PED/rest/pcds-cadastro/cidades?';
3
4  var formato = 'Json';
5  var uf = 'AC';
6  var params = 'formato='+formato+'&'+uf;uf;
```

```
7  var url = urlPrefix + params;
8  var cidades = [];
9  var token = '<Token do usuário>';
10
11  /* Chamada para obtenção de cidades pertencentes à rede do Cemaden
12   * a partir das unidades da federação.
13   */
14  $.ajax({
15      url: url,
16      headers: ("token": token),
17      type: 'get',
18      contentType: "application/json",
19      success: function (response, status) {
20          console.log("Success!!");
21          console.log(response);
22          console.log(status);
23          cidades = response;
24      },
25      error: function (xhr, desc, err) {
26          console.log(xhr);
27          console.log("Desc: " + desc + "\nErr:" + err);
28      }
29  });
30
31  //Imprime o resultado
32  console.log(cidades);
```

Neste trecho de código é possível observar a montagem da chamada à API. Na linha 9, o usuário informa o seu *token* pessoal obtido no passo anterior, que é repassado à chamada *Ajax* na linha 14 do Código 8.

No Código 9, apresentamos um exemplo de chamada a este webservice utilizando a biblioteca *Axios*.

Código 9 - Exemplo de chamada a um serviço que recupera as cidades pertencentes à rede observacional do Cemaden, dado uma UF. Obs: Este exemplo utiliza a biblioteca *Axios* (vide <https://github.com/axios/axios#axios-api> ou https://axios-http.com/ptbr/docs/api_intro).

```
1  const urlPrefix =
2  'http://sws.cemaden.gov.br/PED/rest/pcds-cadastro/cidades?';
3
4  var formats:, = 'Json';
5  var uf = 'AC';
6  var params = 'formato='+formato+'&'+uf=uf;
7  var url = urlPrefix + params;
8  var cidades = [];
9  var token = '<Token do usuario>';
10
11
```



```
12  /* Chamada para obtenção de cidades pertencentes à rede do Cemaden
13  * a partir das unidades da federação.
14  */
15  axios({
16      method: 'get',
17      url: url,
18      contentType: "application/json",
19      headers: {"token": token
20  })
21  .then(response => {
22      console.log(response)
23      cidades = response.data;
24  })
25  .catch(error => {
26      console.log(error)
27  })
28  //Imprime o resultado
  console.log(cidades);
```

Em ambos os casos, o retorno da chamada assíncrona conterá um *array* de cidades no formato JSON devolvido pelo webservice.

3.3. Python

Nesta seção iremos apresentar ao usuário como implementar o fluxo de comunicação descrito na Figura 1 utilizando Python.

Deve-se instalar o pacote **requests**, que pode ser adquirido usando os seguintes comandos, dependendo de sua versão do Python.

Usando **PIP**:

```
pip install requests
```

Usando **Anaconda**:

```
conda install requests
```

3.3.1. Recuperação do Token do Usuário

O exemplo de código a seguir faz login no SGAA usando o e-mail e senha apontados pelo usuário (deve-se alterar esses termos em vermelho no código para os valores reais). Os endereços de cada servidor (disponíveis na Seção 2.1 e 2.2) também devem ser substituídos pelos valores reais.

Código 10 - Recuperação do token usando Python.

```
1  import requests
2  token_url =
3  'http://sgaa.cemaden.gov.br/SGAA/rest/controle-token/tokens'
```

```
4 login = {'email': 'EMAIL', 'password': 'SENHA'}
5 response = requests.post(token_url, json=login)
6 content = response.json()
  token = content['token']
```

Uma vez que a conexão tenha sucesso e o e-mail e senha estejam corretos, será recuperado o **token** pelo código acima. Caso contrário, o usuário pode verificar o conteúdo das variáveis **response** e **content** para verificar a origem do erro.

3.3.2. Exemplo de chamada a um Serviço Web

Em posse do **token**, ele poderá ser utilizado para acessar os dados do servidor. O código abaixo é um exemplo de como acessar os dados do servidor, no caso, o serviço de dados recentes para a rede do Cemaden (11) no estado do Acre (AC):

Código 11 - Requisição de dados usando Python.

```
7 sws_url
8 ='http://sws.cemaden.gov.br/PED/rest/pcds/pcds-dados-recentes'
9 params = dict(rede=11, uf='AC')
10 r = requests.get(sws_url, params=params, headers={'token': token})
  data = r.text
```

A variável **data** irá conter os dados em texto simples. Pode-se então salvar esta variável em algum arquivo, enviá-la para outro programa, ou usar outros pacotes do Python, como o **pandas**, para processá-los no mesmo fluxo de dados. Novamente, em caso de erro o usuário pode verificar o conteúdo das variáveis **r** e **data**.

4. Regras de Acesso para os serviços

Cada usuário cadastrado no sistema irá pertencer a um determinado perfil e, por sua vez, a cada perfil de acesso será atribuída uma regra de acesso.

A Tabela abaixo apresenta os campos indicativos do número máximo de acessos que um usuário poderá fazer ao sistema com determinado token, de acordo com a regra de acesso associada (campo **num_max_access**) e o intervalo temporal que servirá de base para as medições (campo **time_slot**).

Tabela 4: Tabela contendo os limites individuais para o número de acessos e intervalo temporal, de acordo com o perfil do usuário.

Identificador (Id)	Número máx de acessos (num_max_access)	Intervalo temporal em minutos (time_slot)
1	3600	1
2	12	1

3	180	1
---	-----	---

Dessa forma, de acordo com a tabela, o usuário que utilizar um token com o perfil de Administrador (Id=1) poderá realizar até 3600 acessos por minuto. Já o token com perfil de Público Interno (Id = 2) poderá realizar até 12 acessos por minuto, enquanto que para os parceiros (usuário externo do tipo parceiro), com Id = 3, poderá realizar no máximo 180 acessos por minuto e assim por diante. Importante destacar que o parâmetro *time_slot* elencado na tabela é relativo aos acessos do usuário no sistema (Ex: número máximo de acessos do usuário por unidade de tempo).

Caso o usuário ultrapasse esse limite, é realizado um bloqueio temporário no mesmo, por um período de 1 minuto, e então o mesmo é liberado para um novo acesso.

Estes limites foram adicionados com o intuito de limitar e controlar o acesso a utilização de recursos, sistemas e hosts apenas a pessoas autorizadas, preservando assim a infraestrutura de TIC do centro de ataques maliciosos.