

MANUAL DE USUARIO

ACCESO API REST

Versión 0.4

Introducción	4
Objetivos	4
Avisos preliminares	4
Visión general	4
Diagrama de secuencia	5
Generación de claves con openssl	5
Alta en el Gateway	6
Generación de JWT	6
Ejemplo en JavaScript	7
Ejemplo en Python	8
Pruebas de Servicios REST	8
SoapUI	8
A – REST TGT	8
B – REST - ST	9
C – Codificación a Base 64	10
D – REST – JWT2	11
E – API 1	13
POSTMAN	15
A – REST TGT	15
B – REST - ST	16
C – Codificación a Base 64	17
D – REST – JWT2	19
E – API 1	20

Introducción

Este documento describe el proceso por el cual un sistema cliente puede invocar servicios de GDE a través de su API Gateway.

El Gateway autentica al cliente por sus credenciales. El cliente debe presentar dentro del request HTTPS un JSon Web Token, en adelante JWT. Se asume que el lector conoce los conceptos básicos de un JWT, su estructura de datos: header, payload y signature. El Gateway analizará el JWT y verificará validez por: ISS, vigencia y firma digital.

Además hay servicios que requieren conocer quién es el usuario final que está operando en el sistema cliente. Ese usuario final debe tener un usuario GDE. El sistema cliente debe hacer una autenticación extra con las credenciales GDE del usuario.

Objetivos

Al finalizar la lectura de este documento el lector estará en condiciones de:

- Generar un JWT
- Obtener un JWT de GDE
- Consumir la API

Avisos preliminares

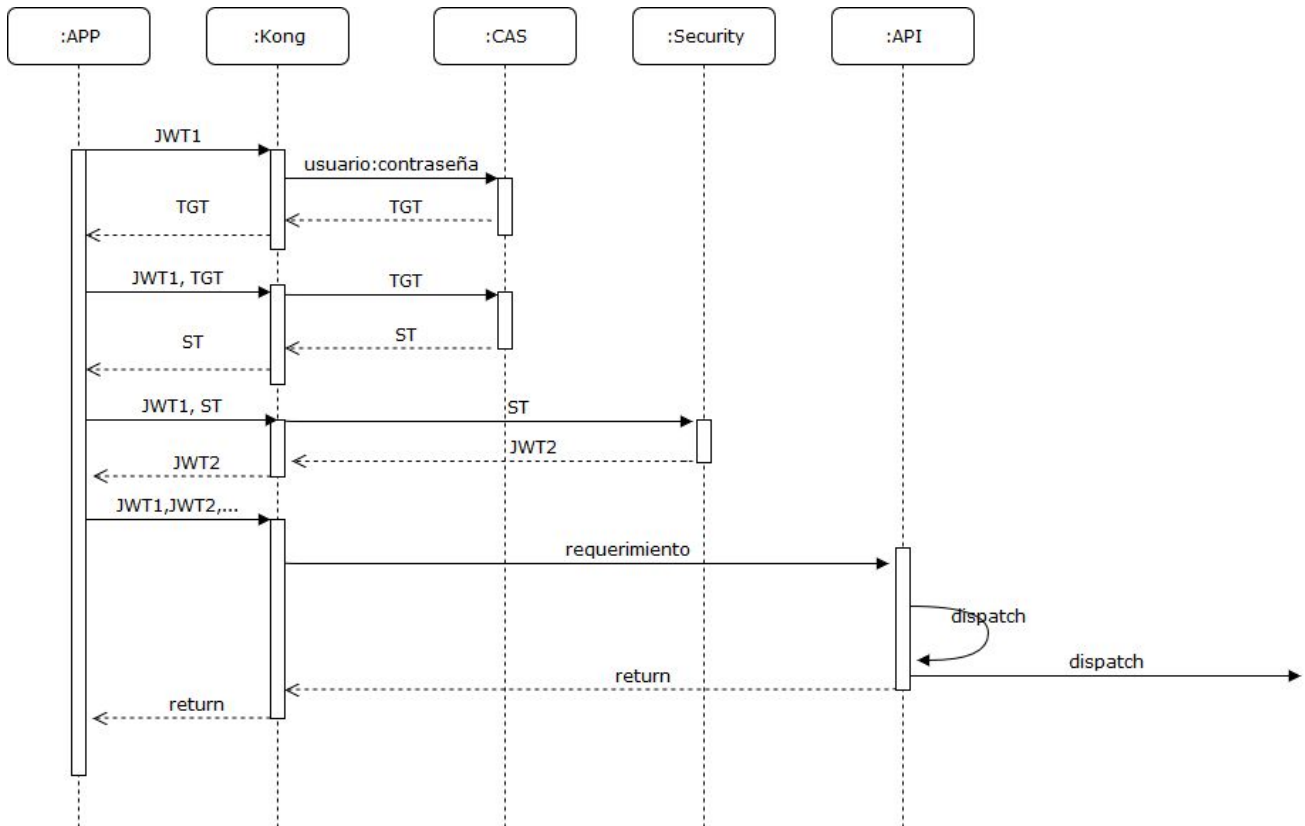
Se verá cómo acceder a los servicios REST en general, y autenticar un sistema cliente. El estudio de cada servicio REST está fuera del alcance de este Manual y debe consultar la documentación del mismo.

Un ejemplo de generación de JWT propuesta utiliza Python, no pretende limitar el uso de cualquier otro lenguaje de programación.

Visión general

El sistema cliente es el encargado de generar el JWT que acompañará la solicitud REST y presentársela al Gateway, quién se encargará de validar la firma y vigencia del JWT; si es válida, envía el requerimiento a la REST API.

Diagrama de secuencia



Generación de claves con openssl

NOTA: Si trabaja con certificado digital puede saltar esta sección.

```
$ openssl genrsa -out priv.pem 2048
```

```
$ openssl rsa -in priv.pem -pubout -out pub.pem
```

Salidas de ejemplo

priv.pem

```

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAtis/iHjG3Woq7SfIZfqKCIMacGvQmCXnLOryJjEl6/ZfUKGS
..... corte intencional .....
fbp5LwT1pU6Qthl2EJgbMqP8X/SdA896367v3N/WsdkBQ0Q0mN+LIw==
-----END RSA PRIVATE KEY-----
  
```

pub.pem

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtiS/iHjG3Woq7SfIZfqK
...
GC5wSfdTnnOy85Pvm9dr1HFInLC+9iEzfppHtSkoZzBf0sF5Lx0SnrOXr+VZreBC
nwIDAQAB
-----END PUBLIC KEY-----
```

Una vez obtenido el par de claves, enviar pub.pem en el ticket creado para infra. Conservar ambos archivos.

Alta en el Gateway

Cada consumidor de APIs será dado de alta en el Gateway. Los clientes con acceso al sistema de Incidentes creará un ticket a infraestructura en `incidencias.modernizacion.gob.ar`. Los clientes sin acceso al sistema de Incidencias enviará un email al contacto dentro del Ministerio de Modernización.

En ambos casos se enviará la clave pública, ver párrafo anterior, en texto claro como archivo .pem, .txt o pegado en el texto de la solicitud.

El solicitante debe enviar:

1. La clave pública del usuario (*ver Generación de claves con openssl*)

Recibirá:

1. Key/ISS: identificador-unico-del-cliente (ver Key)
2. URL de acceso a los servicios

Generación de JWT

El cliente debe generar un JWT con la siguiente estructura cada vez que requiera autenticar contra el Gateway:

Header:

```
{ "typ": "JWT", alg: "RS256" }
```

Payload:

```
{ iss : "identificador-unico-del-cliente"
```

nbf: timestamp de fecha de inicio de vigencia en formato unix timestamp

exp: timestamp de fecha de fin de vigencia en formato unix timestamp

```
}
```

IMPORTANTE: la vigencia DEBE ser suficiente para la transacción. Tiempos de exp cortos darán error por token expirado y tiempos de exp muy largos encierra riesgos innecesarios.

Ejemplo en JavaScript

La siguiente ilustración muestra un generador y verificador de código abierto adaptado a nuestras necesidades. Fue escrito en JavaScript y no precisa instalación, con el navegador disponible en el puesto de trabajo correrá localmente.

JWT generador y verificador

(Paso1) Completar campos del JWT

/gde/ejemplo	Identificador unico del cliente(iss)
5	Tiempo de expiracion en minutos(exp)

NOTA: El campo "nbf" no se tiene en cuenta para generar este JWT

(Paso2) Ingresar clave privada

RS256 (SHA256withRSA RSA2048bit:z4) with default private key ▼

Private key: -----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEaucGGJSclF6C0vCF/8/Zer+Eh7wvc6Wnr1F2Bb8xHkGxCsNzf
e65Zwr2bxg7HT7X23p5mQ48Nc/7uyIdWsU+axGXChZwGQ2/G9JZm2sx6IKnQxnh0

(Paso3) Click "Firmar!"

Firmar!

JWT generado: yXsYYE MMRHirUh_vnTRP82hn0sGxxVd4aXdK5uieEVLkcmBykBgTKGXabEuoTiM
zhrKMw9Ir0yyjVMCZ9m4EyED0NT0GRH7jEvUyAUXYDspBsEsGhVVjg_c0hBwNFnd
QUpg

(Paso4) Verificar estructura del JWT

Ingresar clave publica

Public key: -----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEaucGGJSclF6C0vCF/8/Zer+Eh7wvc6Wnr1F2Bb8xHkGxCsNzfe65Zwr2bxg7HT7X23p5mQ48Nc/7uyIdWsU+a

Descargar del siguiente link:

https://drive.google.com/file/d/1R0QTProoFC0t_ZHCuLiSmKIgw7VdqmxP/view?usp=sharing

Descomprimir el archivo.zip, y después en el directorio principal cargar el archivo index.html en un navegador.

Ejemplo en Python

```
import jwt
import time

nbf = int(time.time() - 60)
exp = int(time.time() + 60)

#Par de claves. El Consumer tiene asociada pub.pem
private_key = open ("priv.pem").read()
#Clave utilizada en iss. Identifica al Consumer
key = open ("key").read()
key = key[:-1]
encoded = jwt.encode({"iss": key, "nbf": nbf, "exp": exp}, private_key,
algorithm='RS256')
print("Authorization: Bearer " + encoded.decode('utf-8'), end='')
```

Pruebas de Servicios REST

En este ejemplo se invoca un servicio que requiere autenticar primero contra Kong, y luego contra GDE. Por lo tanto hay que enviar un JWT firmado, luego obtener un TGT y un ST de GDE y luego invocar al servicio.

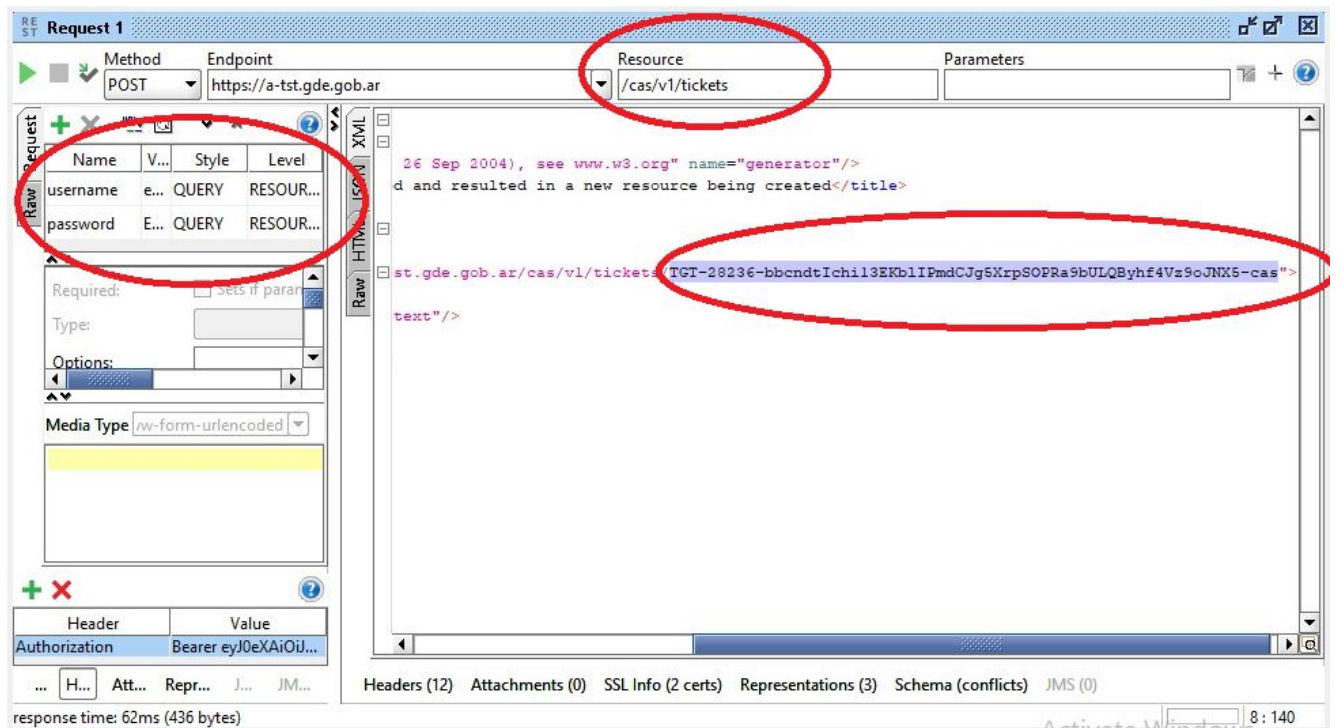
1. SoapUI

1.1. A – REST TGT

Servicio POST.

Se coloca usuario y clave (de GDE) y se marca que se agregue al query (URL) los datos (usuario y clave).

En el Header se coloca la Authorization (JWT1).



El resultado (en el request), se toma la URL donde se encuentra el ticket, y se copia el TGT para el paso siguiente.

action="**http://cas.tst.gde.gob.ar/cas/v1/tickets/TGT-7364-fLupn5Kpel5sTSEvGWtTPxHqHpma5bwBKd4MR0trMMyp90RXF-cas**">

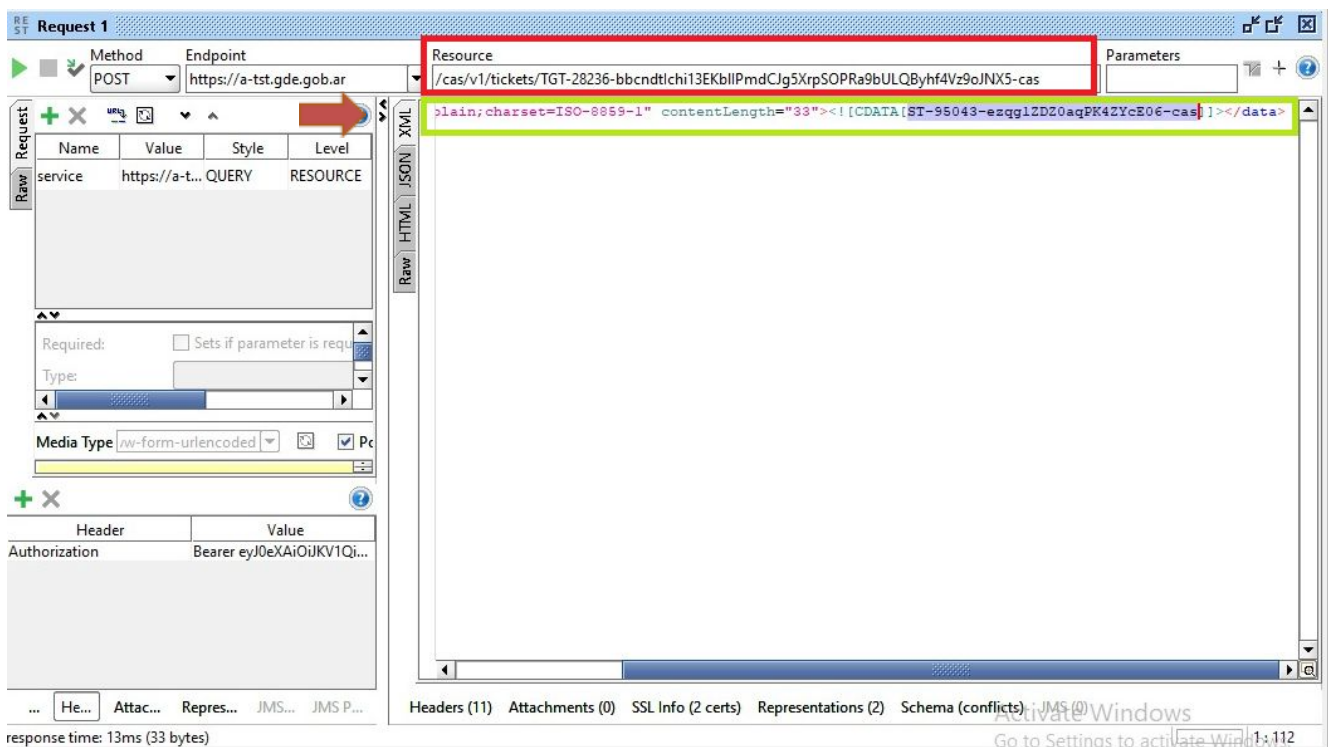
1.2. B – REST - ST

Servicio POST.

En el parámetro service se debe colocar:

https://a-tst.gde.gob.ar/rlm-web/j_spring_cas_security_check

En el Resource, luego de /cas/v1/tickets/, se concatena el resultado del punto anterior (a partir de “TGT” hasta “cas” inclusive)



En el XML resultante, dentro del campo “data”, se encuentra el Ticket (ST).

```
<data contentType="text/plain; charset=ISO-8859-1"
  contentLength="33"><![CDATA[ST-24072-9Ed96otT7Gkez425DCrj-cas]]></data>
```

Tomar el ticket (ST), copiándolo para el siguiente paso.

1.3. C – Codificación a Base 64

Se anexa el path del servicio de seguridad

(https://a-tst.gde.gob.ar/rlm-web/j_spring_cas_security_check), concatenándolo al ST del punto anterior, con una coma de por medio (sin espacios).

Por ejemplo:


https://a-tst.gde.gob.ar/rlm-web/j_spring_cas_security_check,ST-24072-9Ed96otT7Gkez425DCrj-cas

Este texto debe pasarse a “Base64”.

Hay varias páginas donde se puede realizar dicha transformación en forma gratuita y online.

Para el ejemplo actual, utilizamos:

<http://www.contadordecaracteres.info/codificador-base64-online-para-incrustar-imagenes-html-css>



Luego de ingresar el texto, se presiona el botón de “Codificar”.

Codifica TEXTO en base64

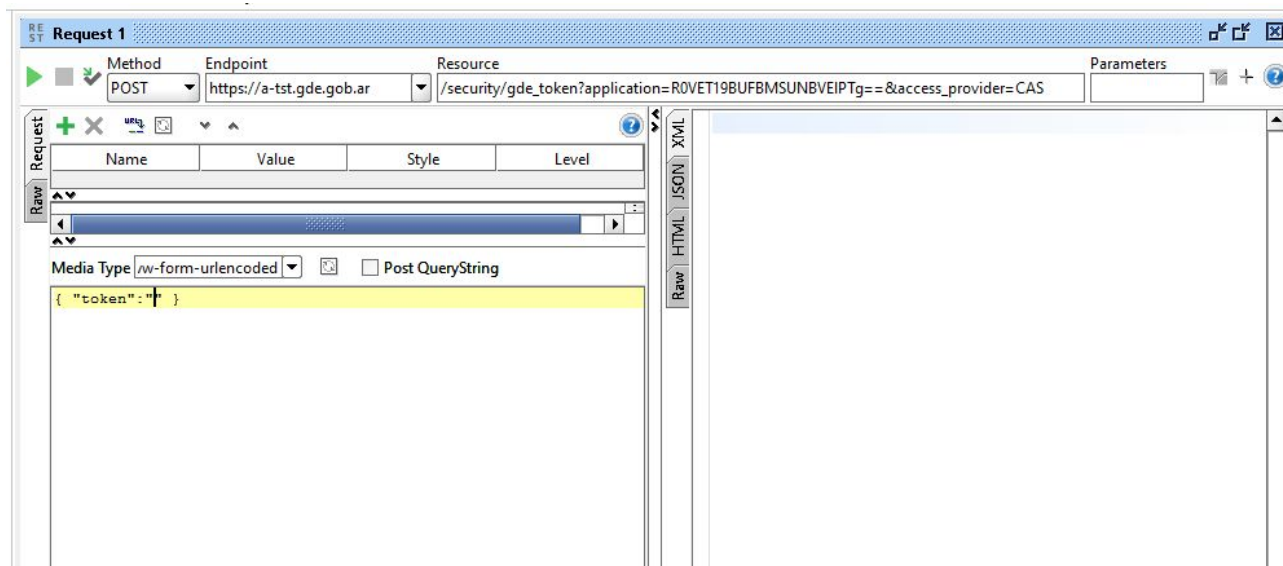


Copiar el texto codificado resultante:

aHR0cHM6Ly9hLXRzdC5nZGUuZ29iLmFyL3Jsbs13ZWlval9zcHJpbmdfY2FzX3NIY3VyaXR5X2NoZWNrLFNULTI0MDcyLTIFZDk2b3RUN0drZXo0MjVEQ3JqLWNhcw==

1.4. D – REST – JWT2

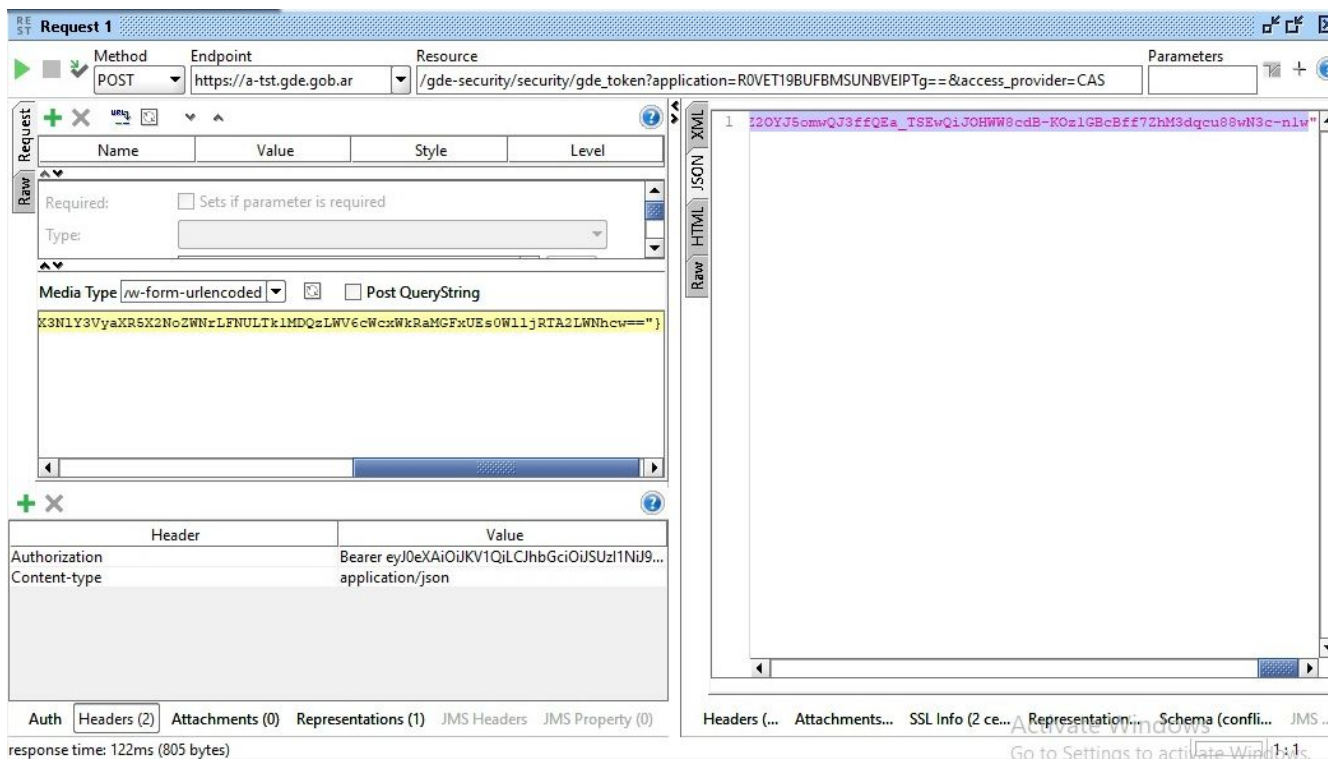
Servicio POST.



En el Body, se quita el texto asignado previamente (entre comillas). El valor que se debe colocar entre comillas es el texto resultante del punto anterior (la codificación en base 64).

Por ejemplo:

```
{
  "token": "aHR0cHM6Ly9hLXRzdC5nZGUuZ29iLmFyL3JsbnS13ZWlval9zcHJpbmdfY2FzX3NI
Y3VyaXR5X2NoZWNrLFNULTI0MTU4LUdoNTBNWWc1UVZNQmVMdktGbGI4LWNhew=="
}
```



"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJleHAiOjE1MzE0MTM3MTMsImdkZSIwYXIsb2FkIjp7InBsYXRmb3JrTGVS2ZWwiOiIxIiwic2VjTGV2ZWxEZXNjIjoiaR0VETyAoMSkgLSBDQVMgKDEpliwidXNlcil6eyJ1c2VybW FtZSI6IkRBTkItFEFMQkVSVE9HT05aQUxFWiJ9LCJnZGVSYXdUb2tlibi6eyJ0b2tlibi6ImFIUjBjSE02THk5aExYUnpkQzVuWkdVdVoyOWIMbUZ5TDNKc2JTMtNaV0l2YWw5emNISnBibWRmWTJGelgzTmxZM1Z5YVhSNVgyTm9aV05yTEZOVUxUSTBNVFU0TFVkb05UQk5XV2MxVWZaTlFtVk1ka3RHYkdsNEExXTmhjdz09IiwiY3RoZXJJbmZvcmlhdGlvbi6bnVsbH0sInByb3ZpZGVyIjoiaQ0FTIn19.jCrZvXj5pq1aP1ZIgggBHWZWY-psZk3TKqp8W9wKqAayY2i0gOvaxiJNZCmGCssKK59SGCl-4uvefL18rimQ1bygvnPawxzP6VVXWf8SkCW5Ihi-gKDDsqOZBalgdecO1IH_F2wZzQ0KcRH428VR974uSobPulqET0WZt791PqNdQhYSKqpL_3Kg9mjizfMmDYRdsZtf45dPGxqGAvuxT5rQw90sTKTWM5VgLWlJFb9XT_IKMgyOD_sm5WrA3Sc8sGTSaYtl4q8wR_L2vG8D51DhRuFUj9u6nyJmTJeoQoQX-cQmpFDca2G49DsytN9F3SWLr8qLgJRiLBYv0tyg"

Servicio GET

En el Header se pasan:

- Cada uno de ellos comienzan con “Bearer ” con un espacio entre “Bearer” y el texto del JWT correspondiente.

Request 1

Method: GET Endpoint: https://a-tst.gde.gob.ar Resource: /rlm-web/RLMRestServices/CamposExternos/listAllTipoRegistroPublico Parameters: ?codigo=RLTST0001

Raw Request

Name	Value	Style	Level
codi...	RLTS...	QUE...	

Required: ☐ Type:

Header	Value
Authorization	Bearer eyJ0eX...
Auth	Bearer eyJ0eX...

Esto se modificará

... A... R... J... J...

response time: 5255ms (2506 bytes)

Raw JSON

```
1 {
2   {
3     "id": 21821,
4     "codigo": "RLTST0085",
5     "descripcion": "Registro de Postulantes para Auditores Internos"
6   },
7   {
8     "id": 22220,
9     "codigo": "RLTST0086",
10    "descripcion": "Registro Único de Biodiversidad"
11  },
12  {
13    "id": 22420,
14    "codigo": "RLTST0087",
15    "descripcion": "Registro Nacional de Establecimientos Fitosanitarios"
16  },
17  {
18    "id": 22620,
19    "codigo": "RLTST0088",
20    "descripcion": "Registro Nacional de Establecimientos de Servicios de Salud"
```

Headers (12) Attachments (0) SSL Info (2 certs) Representations (3) Schema (conflicts) JMS (0)

Activata Windows 3:40

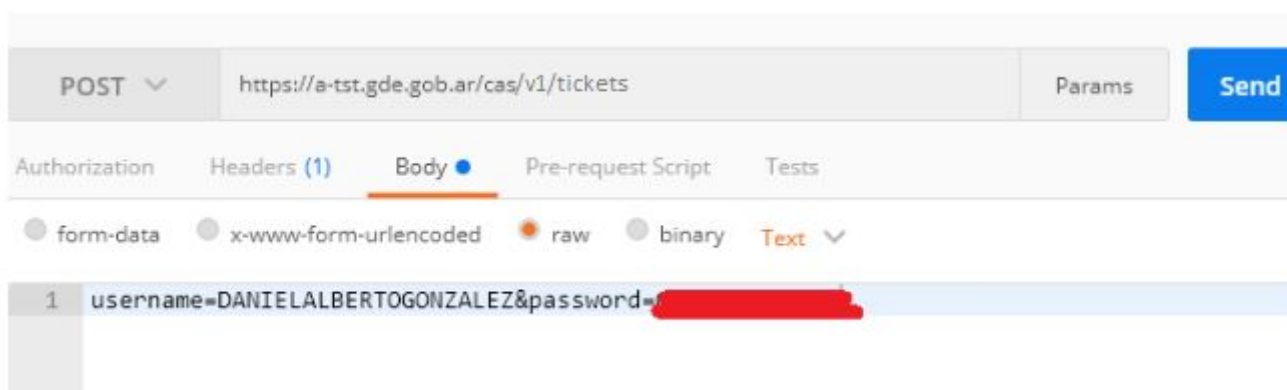
2. POSTMAN

2.1. A – REST TGT

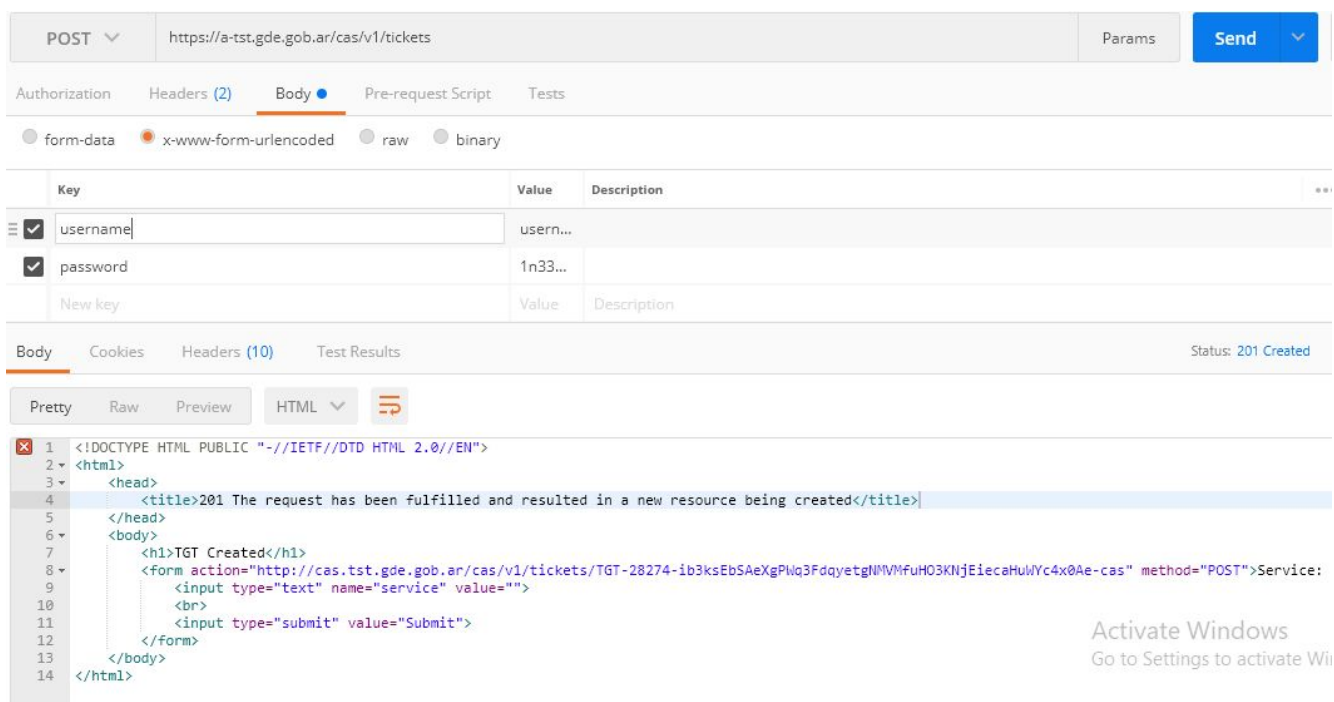
Servicio POST.

En el Body, colocar usuario y contraseña de GDE.

En el Header colocar como parámetro Authorization, y como valor la identificación correspondiente.



Al ejecutarse,, se recibe como respuesta la URL del ticket.



El resultado (en el request), se toma la URL donde se encuentra el ticket, y se copia el TGT para el paso siguiente.

<form

action="**http://cas.tst.gde.gob.ar/cas/v1/tickets/TGT-7407-MDOko9aNrT6y1R6jkjF9c4tbIIFO95moxN7P3hccAVYWEP0uzU-cas**" method="POST">Service:

Siendo el ticket **TGT-7407-MDOko9aNrT6y1R6jkjF9c4tbIIFO95moxN7P3hccAVYWEP0uzU-cas**.

2.2. B – REST - ST

Servicio POST.

Se toma el ticket (a partir de TGT) resultante del punto anterior, colocándolo luego de “https://a-tst.gde.gob.ar/cas/” en el cuadro de POST.

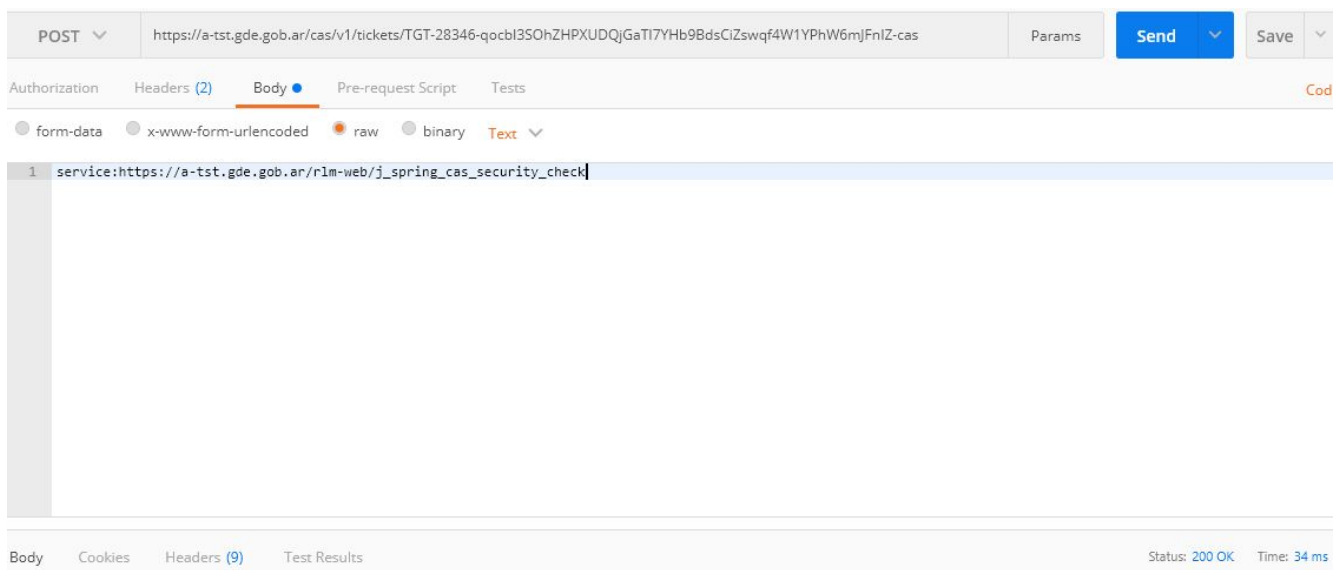
Ejemplo:

https://a-tst.gde.gob.ar/cas/v1/tickets/TGT-7407-MDOko9aNrT6y1R6jkjF9c4tbIIFO95moxN7P3hccAVYWEP0uzU-cas

En el Body, seleccionando raw, y colocar:

https://a-tst.gde.gob.ar/r1m-web/j_spring_cas_security_check

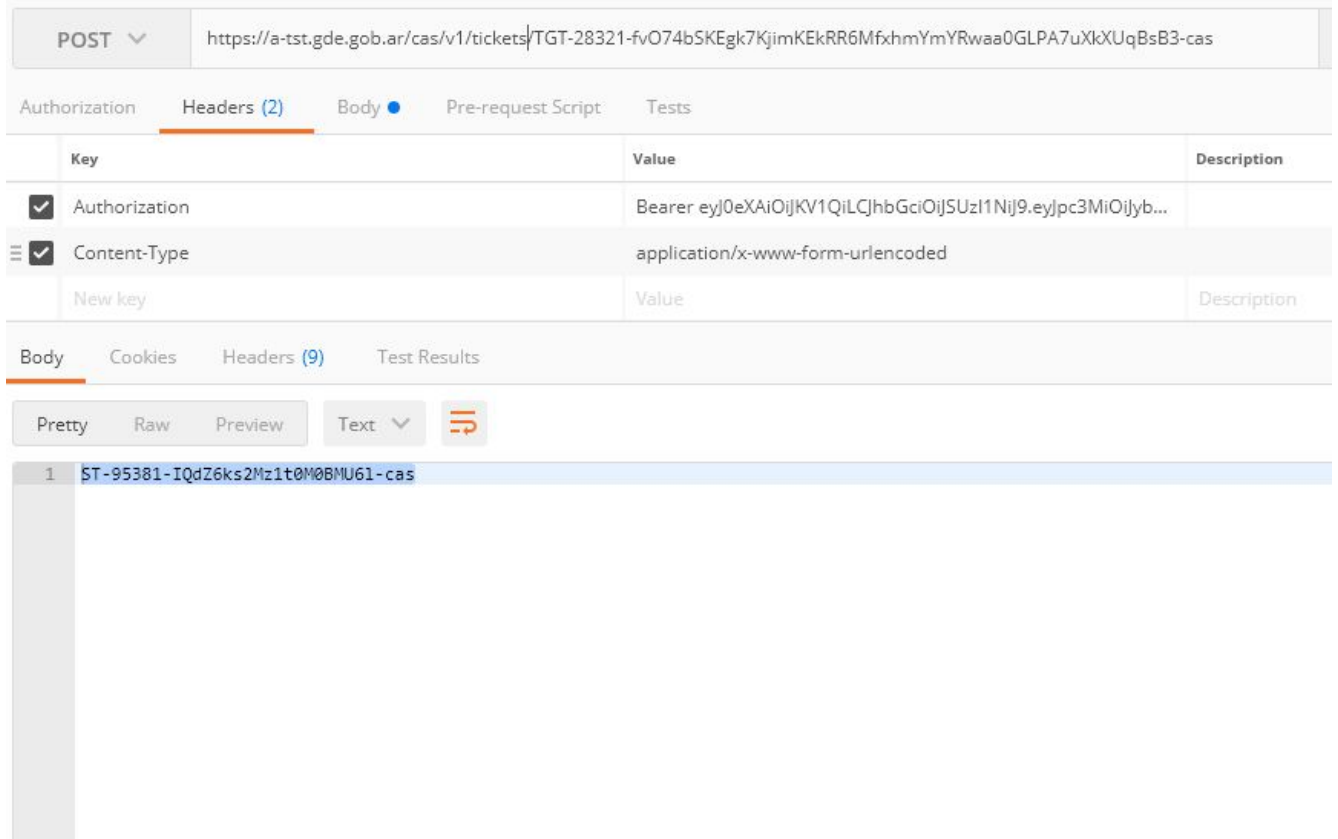
En el Header, se debe mantener el Authorization del punto anterior.



Al ejecutar, se recibe el Ticket.

En el caso del ejemplo, el resultado es:

ST-24283-woaWgFk1ILYaoeljwDup-cas



2.3. C – Codificación a Base 64

Se anexa el path del servicio de seguridad (`https://a-tst.gde.gob.ar/rlm-web/j_spring_cas_security_check`), concatenándolo al ST del punto anterior, con una coma de por medio (sin espacios).

Por ejemplo:

`https://a-tst.gde.gob.ar/rlm-web/j_spring_cas_security_check,ST-24283-woaWgFk1ILYaoeljwDup-cas`

Este texto debe pasarse a “Base64”.

Hay varias páginas donde se puede realizar dicha transformación en forma gratuita y online.

Para el ejemplo actual, utilizamos:

<http://www.contadordecaracteres.info/codificador-base64-online-para-incrustar-imagenes-html-css>

🔒 No seguro | www.contadordecaracteres.info/codificador-base64-online-para-incrustar-imagenes-html-css

📁 DCyAT 📁 Banco 📁 Deportes 📁 Vs 📁 Utils 📁 Clima 📁 Ver 📁 Laboral

Codifica TEXTO en base64

Escribir texto para codificar y pulsar "Codificar" para ver el resultado

https://a-tst.gde.gob.ar/r/m-web/j_ spring_cas_security_check.ST-24283-woaWgFk1ILYaoelJwDup-cas

Codificar **borrar**

Luego de ingresar el texto, se presiona el botón de “Codificar”.

🏠 🔒 www.contadordecaracteres.info/codificador-base64-online-para-incrustar-imagenes-html-css

es 📁 DCyAT 📁 Banco 📁 Deportes 📁 Vs 📁 Utils 📁 Clima 📁 Ver 📁 Laboral

Codifica TEXTO en base64

Escribir texto para codificar y pulsar "Codificar" para ver el resultado

aHR0cHM6Ly9hLXRzdC5nZGUuZ29iLmFyL3JsS13ZWlval9zcHJpbmdfY2FzX3NIY3VyaXR5X2NoZWNrLFNULTI0MjgzLXdvYVdnRmsxbExZYW9lbGp3RHVwLWNhcw==

Codificar **borrar**

El resultante es el que se debe copiar:

aHR0cHM6Ly9hLXRzdC5nZGUuZ29iLmFyL3JsbS13ZWlval9zcHJpbmdfY2FzX3NIY3VyaXR5X2NoZWNrLFNULTI0MjgzLXdvYVdnRmsxbExZlW9lbGp3RHVwLWNhew==

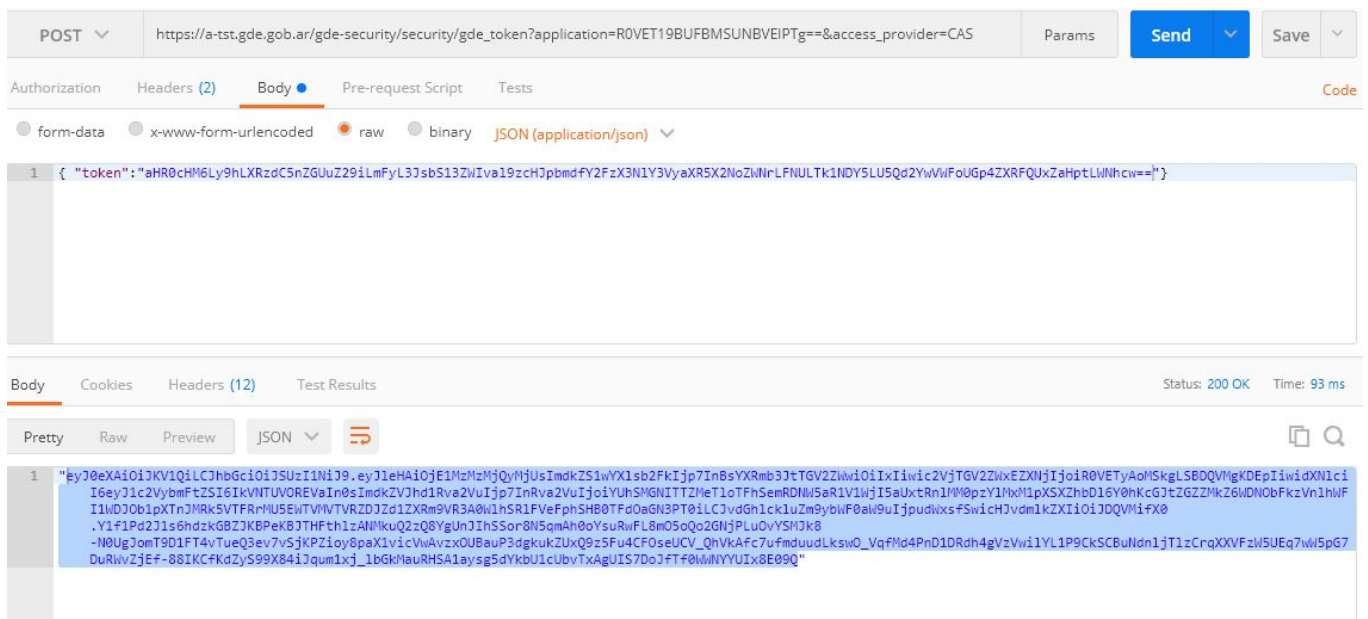
2.4. D – REST – JWT2

Servicio POST

En el Body, se quita el texto asignado previamente (entre comillas). El valor que se debe colocar entre comillas es el texto resultante del punto anterior (la codificación en base 64).

Por ejemplo:

```
{ "token": "aHR0cHM6Ly9hLXRzdC5nZGUuZ29iLmFyL3JsbS13ZWlval9zcHJpbmdfY2FzX3NIY3VyaXR5X2NoZWNrLFNULTI0MjgzLXdvYVdnRmsxbExZlW9lbGp3RHVwLWNhew==" }
```



Al ejecutarse, se genera un resultado como el siguiente:

"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJleHAiOiJlMzE0MTY2NjQsImdkZS1wYXlsb2FkIjp7InBsYXRmb3JtTGVS2ZWwiOiIxIiwic2VjTGVS2ZWxEZlW9lbGp3RHVwLWNhew=="

eZhPOW4XUPHTyHToQF7kuqk1NwOMTCxaRe9Oc7_qp6oAhZKAJbj-9bDUpYLbz1dYbsn
L7Wsu33_FfKdYhFK5twmS4ZKAP3OcEnLJfah5ggCa-DVS_-z4I3UsWhq8Syq_GrHaPPz9B5
usJmwr3A6vA0e_ktUWPFX_VA_FnvQPf0af1Q_tU8nX3dfWEsFavdVsu7Aiw2j75Ck_Bgtl4A
DqXS7n872q57qVWFoQpl1fGIf_emqFM0O_nle5g6trfWR2c0w"

2.5. E – API 1

Servicio GET

En el cuadro del GET, se mapea a la API, colocando los parámetros correspondientes.

En el Header (por ahora) se pasan el Authorization (el mismo que se pasó oportunamente), y el Auth (del punto anterior). Cada uno de ellos comienzan con “Bearer ” (con un espacio entre “Bearer” y el texto siguiente).

The screenshot shows a REST client interface with the following components:

- GET** dropdown and URL: `https://a-tst.gde.gob.ar/rim-web/RLMRestServices/CamposExternos/listAllTipoRegistroPublico?codigo=RLTST0001`
- Parameters Table:**

Key	Value	Description
<input checked="" type="checkbox"/> codigo	RLTST0001	
New key	Value	Description

- Headers Tab:**

Key	Value	Description
<input checked="" type="checkbox"/> Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJyb...	
<input checked="" type="checkbox"/> Auth	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJleHAiOiE1M...	
New key	Value	Description

- Body Tab:** Shows the JSON response in Pretty format.

```
[
  {
    "id": 21821,
    "codigo": "RLTST0085",
    "descripcion": "Registro de Postulantes para Auditores Internos"
  },
  {
    "id": 22220,
    "codigo": "RLTST0086",
    "descripcion": "Registro Único de Biodiversidad"
  },
  {
    "id": 22420,
    "codigo": "RLTST0087",
    "descripcion": "Registro de Postulantes para Auditores Externos"
  }
]
```

Authorization > es el JWT1

Auth >

Security. Es el JWT2.

Código >

es el parámetro de la API correspondiente.

CAS Simple

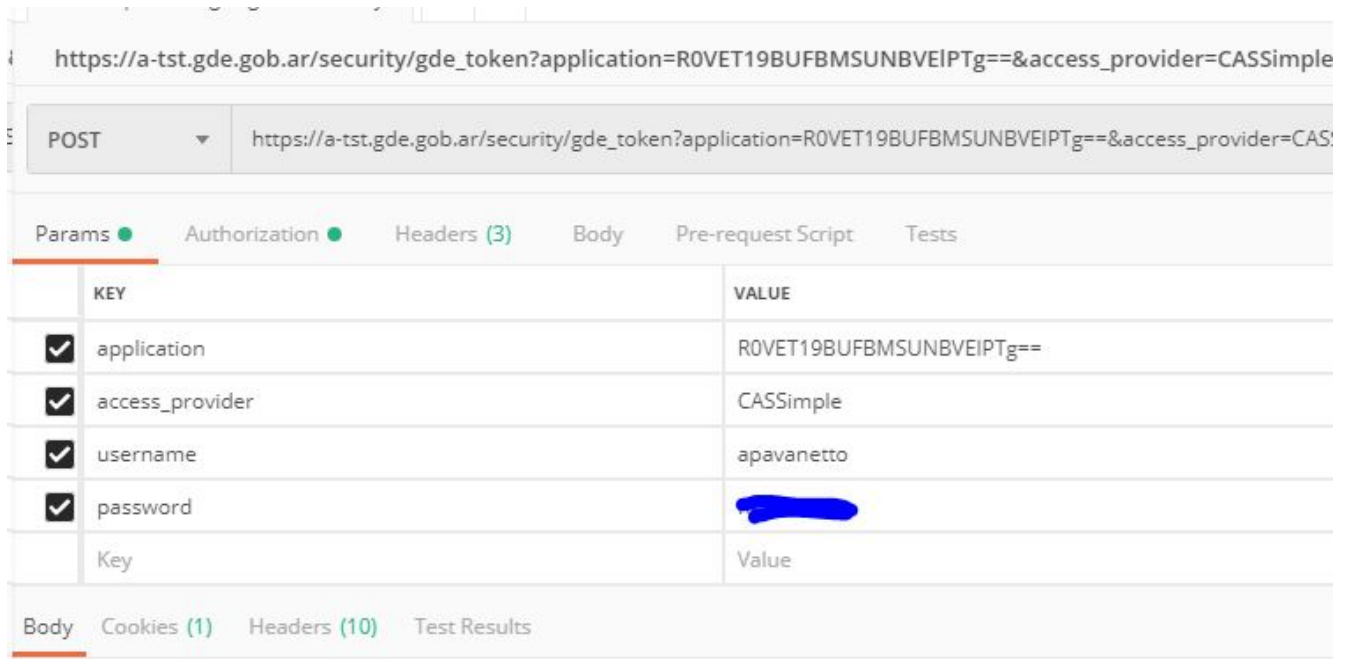
CAS Simple es una API de seguridad con compatibilidad hacia atrás. A diferencia de las anteriores, esta API entrega un JWT2 en un paso; eliminando codificación y manipulación de datos. Su “payload” incluye la CLAVE/ISS requerida por el gateway durante la validación de la petición. Una vez autorizado el ingreso, el requerimiento llega a la API a consumir sin transformaciones.

NOTA: El path deja de ser **/gde-security/security** y pasa a ser **/security**

El ejemplo utiliza POSTMAN NATIVE

1. Parámetros con gde_token

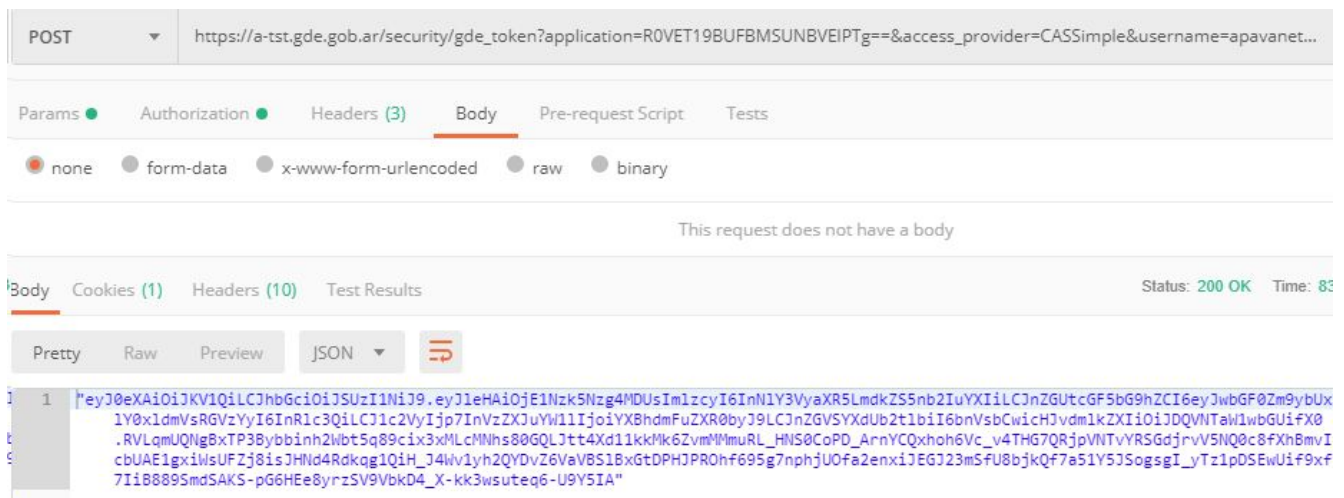
https://a-tst.gde.gob.ar/security/gde_token?application=R0VET19BUFBMSUNBVEIPTg==&access_provider=CASSimple&username=*****&password=*****



2. Authorization

Con el POST enviar el encabezado

Authorization: Bearer JWT1



5. Análisis de JWT2

Header

```

{
  "typ": "JWT",
  "alg": "RS256"
}

```

Payload

```

{
  "exp": 1579978805,
  "iss": "security.gde.gob.ar",
  "gde-payload": {
    "platformLevel": "1",
    "secLevelDesc": "test",
    "user": {
      "username": "*****"
    },
    "gdeRawToken": null,
    "provider": "CASSimple"
  }
}

```

6. Parámetros con auth

https://a-tst.gde.gob.ar/security/auth?application=R0VET19BUFBMSUNBVEIPTg==&access_provider=CASSimple&username=*****&password=*****

POST

https://a-tst.gde.gob.ar/security/auth?application=R0VET19BUBFMSUNBVEIPTg==&access_provider=CASSimple&username=

Params

Authorization

Headers (3)

Body

Pre-request Script

Tests

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> application	R0VET19BUBFMSUNBVEIPTg==	
<input checked="" type="checkbox"/> access_provider	CASSimple	
<input checked="" type="checkbox"/> username	apavanetto	
<input checked="" type="checkbox"/> password		
Key	Value	Description

Body

Cookies (1)

Headers (10)

Test Results

Pretty

Raw

Preview

JSON

```
{  "accessToken": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1eHAiOiJlNTEzNTZjZmZmImdkZS1hY2Nlc3Nub2t1b1I6eyJwbGF0Zm9ybUxldmVsIjoimSIzInN1Y0xldmVsRGVzYyI6InRlc3Qi6eyJ0eXB1IjoiriInVzZXJyYW11IjoieXBhdmdFuZXR0byIsInJlcGFydG1jYW9uIjoieRE5TQV1GRGNNTSIsInN1Y3RvciI6I1BWRClSibG1jYSBOYWVpbn25hbCIsImN1aXQiOiIyMDEyNTIyMDQ1MyIsInNob3J0RnVsY25hbnB1IjoibnRvbm1vIFBhdmdFuZXR0byIsImZpcnN0IldHRvIiwiaWF0Ij01JmZlZm91dHRvQGdKZS5nb2IuYX1iIj01fSwiaWRBdXRoIjoimThkYTA2MmYyZTFiYS00MjcyLWExZjctNmEyYTk4IjLCJwcm92aWR1ciI6InBU1NpbXBzSj9.1J8wEmPuUae5LHHJicA83J1wLcgnZ0ccoXKw7-s8yf58h71aETj1cs2u9YhPzKkf1GxTsw6p-crQ41TwMC02AhrYlvTHotByS4C5WGaT9XIhiyMFV_vUcQs7LLMgqS0caDrjpxty5t1x8gsx_e8GFJjJut-6M0p9gaZgm_pgL1cqfpGm-dXyvoebD4F1ANGTwKTFM63rCMHkNjDaxl21TqxMfnvR7wDG443S7A-92MzK9Tocv5R1ldHA7--gm4M19Yn71n22Pkt0sfe7aV13Pmaa"
```

POST https://a-tst.gde.gov.ar/security/auth?application=R0VET19BUFBMSUNBVEIPTg==&access_provider=CASSimple&username=apavanetto...

Params ● Authorization ● **Headers (3)** Body Pre-request Script Tests

KEY	VALUE	DESCRIPTION
Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiOiw...	
<input checked="" type="checkbox"/> Content-Type	application/json	
<input checked="" type="checkbox"/> Accept	application/json;charset=UTF-8	
Key	Value	Description

Body Cookies (1) Headers (10) Test Results Status: 200 OK Time: 12

Pretty Raw Preview JSON ▼

```
{
  "accessToken": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9
    .eyJleHAiOiJlNTEzNTkxZWZmMjY2MTI1IiwiaXNjaWkiOiJwYXBhdFZXR0byIsInRlcGFydGJ1bmFuIjoieRE5TQVGRGNNTSIsInN1Y3Rvcii6I1BWRClSImpcm1zZGIjY2Vl
    ibGljYS80YWNPbzI5bnBvImN1aXQioiIyMDQ1MyIsInNob3J0RnVsE5hbWUiOiJBbnRvbmlvIFBhdmdFuZR0byIsImZpcnN0bmFtZTEiOiJBbnRv
    ldHRvIiwiaWF0Ij01OTY1dHdvOGdkZS5nb2IuYXIifSwiaWRSdXRoIjoieMThkYTAA2MwytZTFiYS00MjcylLExZjctNmEyYTk4ZTRmMjA2In0sIm1z
    iLCJwcm92aWRlcii6IkNBUIPnpbXBSZSJ9.J8wEmPuUae5LHHjiCA83JJwLcgnZcccoKXw7-s8yf58h7laEtjlcs2u9YhpZKkf1Gxtsw6p3fCIWdySt8x9PsC
    -crQ4ITwMC02AhdryLVtHotByS4CSWCgaT9XIhiyMFV_vUcqs7LLMggSOcaDrjpxty5tlx8gsx_eBGfJjut-6M0p9gaZgm_pgLl1cqfpGmqD1QBKSUuzIyy
    -dxivoghD4FJANGIWIE63rCMHjKnidaxU2JTqxMfpvr7WDGA43S7A-92Mzk9Tocy5RildHAZ--gm4Mj9Yp71o22Pkt0sfzaYijPmaaABQqereCYmtV_OHe
  ",
  "refreshToken": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9
    .eyJnZGUtcvmcmVzaFRva2VuIjp7In8sYXRB3JtTGZ2LWwiOiIxIiwic2VjTGZ2LWxEZXNjIjoieGVzdCIsmmpzb25LZXkiOiJnZGUtcvmcmVzaFRva2Vl
    "
```

{

"accessToken":

"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJleHAiOiE1NTEzNjkwNzMsImdkZS1hY2Nlc3NUb2tlbiI6eyJwbGF0Zm9ybUxldmVsljoiMSIsInNlY0xldmVsRGVzYyI6InRlc3QiLCJqc29uS2V5IjoiZ2RlLWJjY2VzclRva2VuIiwidXNlciI6eyJ0eXBIIjoiRiIsInVzZXJuYXW1IjoiYXBhdmFuZXR0byIsInJlcGFydGljaW9uIjoiRE5TQVlGRGNNTSIsInNlY3Rvcil6IiBWRClSImp1cmIzZGljY2lvbiI6IkFkbWluaXN0cmFjaC0zbiBQw7pibGljYSBOYWNPb25hbCIImN1aXQiOiIyMDEyNTIyMDQ1MyIsInNob3J0RnVsbE5hbWUiOiJBbnRvbmlvIFBhdnFuZXR0byIsImZpcnN0bmFtZTEiOiJBbnRvbmlvIiwibGFzdG5hbWUxIjoiUGF2YW5ldHRvIiwidW1haWwiOiJhcGF2YW5ldHRvQGdkZS5nb2IuYXIIifSwiaWRBdXRoiMThkYTA2MwYtZTFiYS00MjcyLWExZjctNmEyYTk4ZTRmMjA2In0sImIzcyI6InNlY3VyaXR5LmdkZS5nb2IuYXIIiLCJwcm92aWRlciI6IkNBUE1NpbXBsZSJ9.J8wEmPuUae5LHHJlcA83JlwLcgnZ0ccoXKw7-s8yF5Bh7laETjles2u9YhPzKkflGxTsW6p3fCIWdYiST8x9PsOc5XSaM-crQ4ITwMC02AhdryLvtHotByS4C5WCGaT9XIhiyMfV_vUcQs7LLMgqSOcaDrjpxty5tlx8gsx_eBGFJjut-6M0p9gaZgm_pgL1cqfpGmqDIQBkSUuz1yy-dXivoghD4FJANGIwKIEM63rCMHjKniDaxU2JTqxMfpvR7WDGA43S7A-92MzK9Tocy5RildHAZ--gm4Mj9Yp7lo22Pkt0sfezaYiJPmaaABQqereCYmtV_OHe22sJdfs4Ig",

"refreshToken":

"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJnZGUtcmVmcmVzaFRva2VuIjpw7InBsYXRmb3JtTGZ2ZWwiOiIxiIiwic2VjTGV2ZWxEZXNjIjoiZGVzdCIzImpzb25LZXkiOiJnZGUtcmVmcmVzaFRva2VuIiwiaWRBdXRoiMThkYTA2MwYtZTFiYS00MjcyLWExZjctNmEyYTk4ZTRmMjA2In0sImIzcyI6InNlY3VyaXR5LmdkZS5nb2IuYXIIifSwiaWRBdXRoiMThkYTA2MwYtZTFiYS00MjcyLWExZjctNmEyYTk4ZTRmMjA2In0sImIzcyI6InNlY3VyaXR5LmdkZS5nb2IuYXIIiLCJwcm92aWRlciI6IkNBUE1NpbXBsZSJ9.J8wEmPuUae5LHHJlcA83JlwLcgnZ0ccoXKw7-s8yF5Bh7laETjles2u9YhPzKkflGxTsW6p3fCIWdYiST8x9PsOc5XSaM-crQ4ITwMC02AhdryLvtHotByS4C5WCGaT9XIhiyMfV_vUcQs7LLMgqSOcaDrjpxty5tlx8gsx_eBGFJjut-6M0p9gaZgm_pgL1cqfpGmqDIQBkSUuz1yy-dXivoghD4FJANGIwKIEM63rCMHjKniDaxU2JTqxMfpvR7WDGA43S7A-92MzK9Tocy5RildHAZ--gm4Mj9Yp7lo22Pkt0sfezaYiJPmaaABQqereCYmtV_OHe22sJdfs4Ig",

"apoderadosToken": null,

"aclToken": null,

"moreInfoToken": null

}