University of Padova
Department of Information Engineering

Biomedical Wearable Technologies for Healthcare and Wellbeing

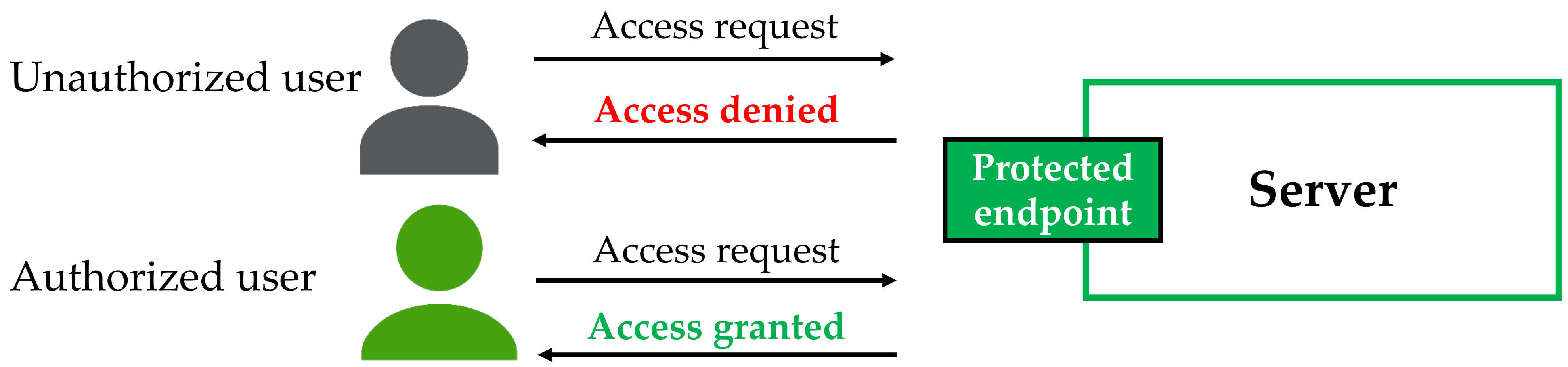
Authorization

A.Y. 2023-2024

Giacomo Cappon

Authorization

- **Authentication:** the process of proving that the user is who he/she declares to be.
- **Authorization:** the act of granting a user permission to do something.
  - Necessary when some resources/tasks are limited to users with special permission (**access control**).
  - It specifies what data a user is allowed to access and what the user can do with that data.



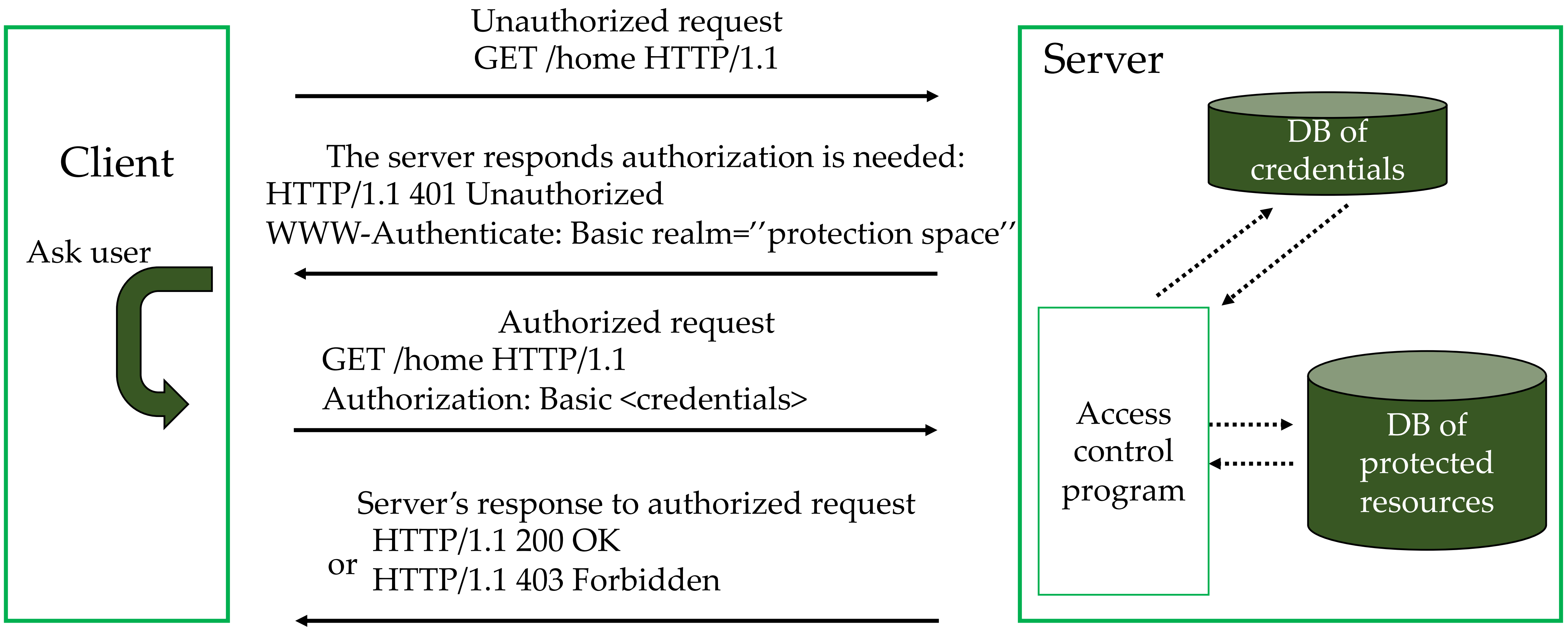
Basic access authentication

- Authorization using **account credentials**
  - Each user of the website/web application has an account with a **user ID** and a **password** (credentials).
  - The user logs into the website/web application providing its credentials.
  - The server verifies credentials and allows the user to access the resources it is authorized to access.
- HTTP request header to provide credentials:

  Authorization: Basic <credentials>

  where <credentials> includes user ID and password.
- Often authorization is performed after the server request it.

Basic access authentication



Unauthorized request
GET /home HTTP/1.1

The server responds authorization is needed:
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm=''protection space''

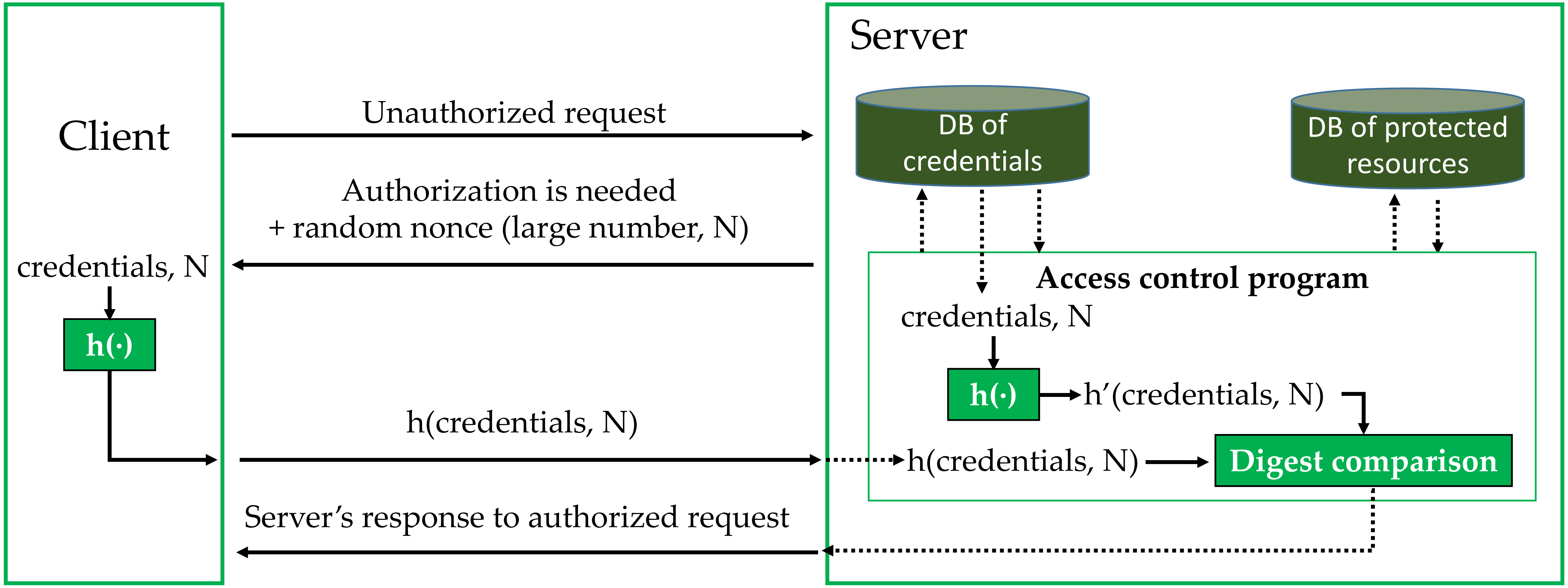
Authorized request
GET /home HTTP/1.1
Authorization: Basic <credentials>

Server's response to authorized request
HTTP/1.1 200 OK
or
HTTP/1.1 403 Forbidden

How can we protect the transmission of credentials?

Digest access authentication

- Its uses an **hash function** to protect credentials.



- Alternative solution: encrypted communication with the **HTTPS** protocol.

The problem of delegated authorization

- A third-party app wants to access some resources of another service provider, whose access is restricted by authorization.

  The authorization must be delegated to the third-party app.

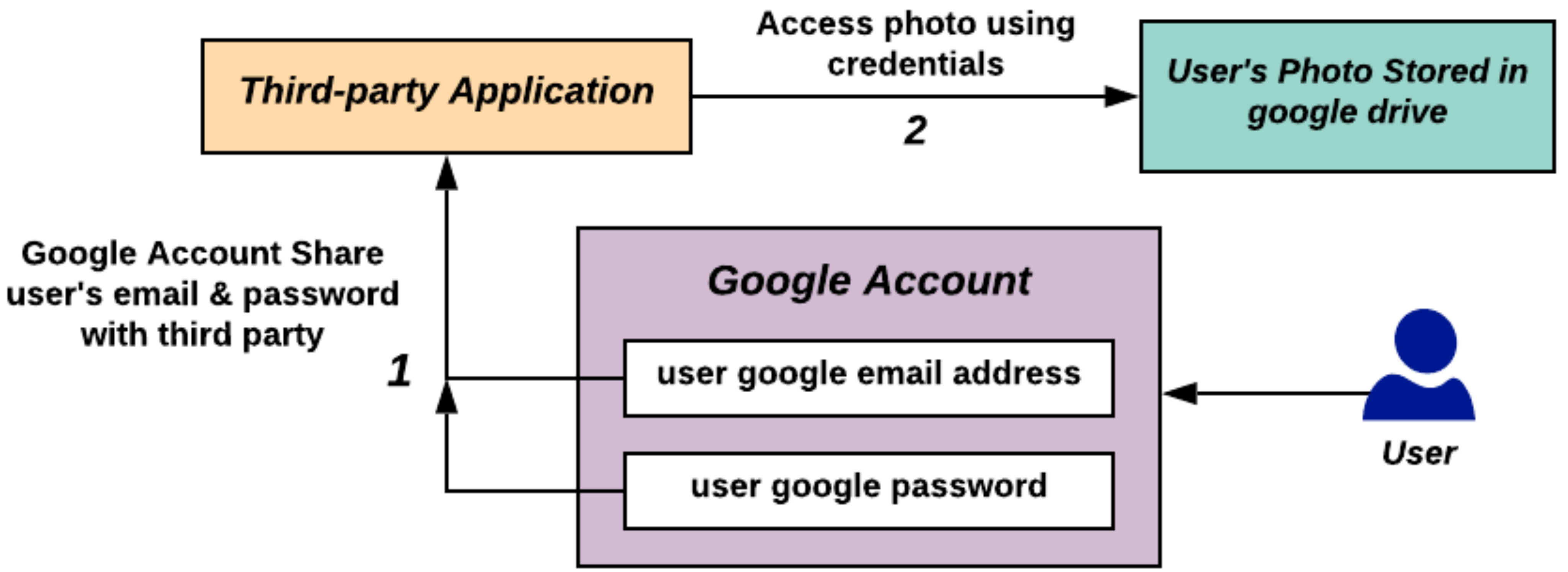
- Today delegated authorization is very common:
  - Your Strava app may synchronize your Fitbit data

    → Fitbit authorization is delegated to Strava
  - You may share your preferred song in Spotify on your Facebook profile

    → Facebook authorization is delegated to Spotify

- How can a third-party app perform delegated authorization?

The incorrect solution: sharing login credentials


Third-party Application
Access photo using credentials
2
User's Photo Stored in google drive
Google Account Share user's email & password with third party
1
Google Account
user google email address
user google password
User

A third party application in order to access user's photo stored in a google drive, google needs to share user's email address and password with the third party.

❌ *Nobody want this Right ?*

OAuth protocol

- **OAuth (Open Authorization):** a protocol for delegated authorization that does not require to share credentials with the third-party app.
- 2007: First version (OAuth 1.0)
- 2012: Second version (OAuth 2.0)
- Main co-author is Blaine Cook, one of the developers of Twitter.
- Today OAuth is commonly used by companies such as Twitter, Amazon, Google, Facebook and Microsoft to permit the users to share information about their accounts with third-party applications or websites.

Why do we study OAuth?

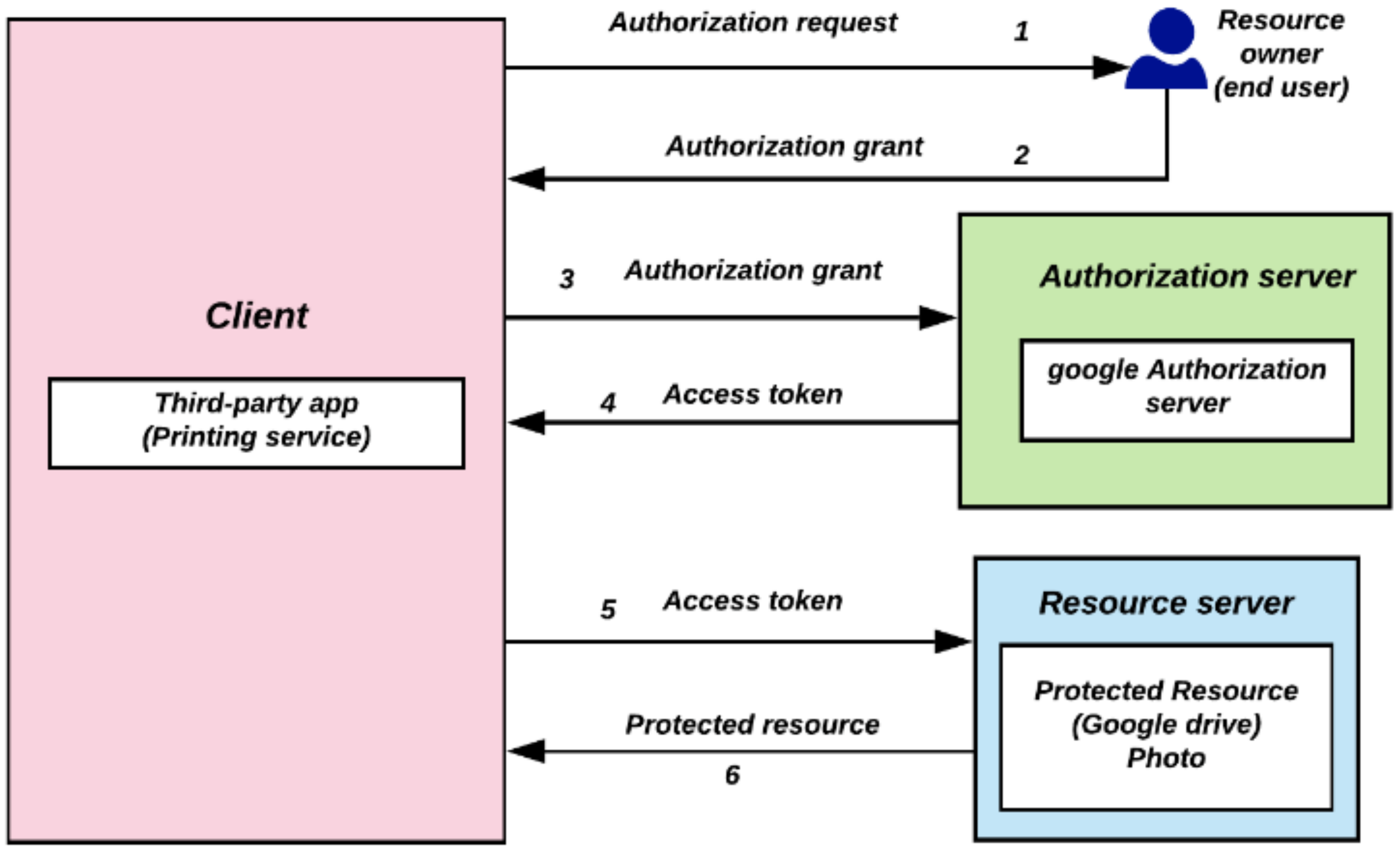
If you develop an app that integrates data from third-party servers that require authorization, you may need to use OAuth to request resources.

OAuth roles

- **Resource owner**: The *user* who authorizes a third-party *application* to access their data.
  - The application's access to the user's data is limited to the scope of the authorization granted (e.g., read or write access).
- **Client**: The third-party *application* that wants to access the *user*'s account.
  - The client must be authorized by the user.
  - The authorization must be validated by an API of the authorization server.
- **Resource server**: The server hosting the protected user data.
- **Authorization server**: It verifies the identity of the *user* and issues access tokens to the third-party *application*.

OAuth abstract model

- Example: a printing service app wants to access Google drive photos



Application registration

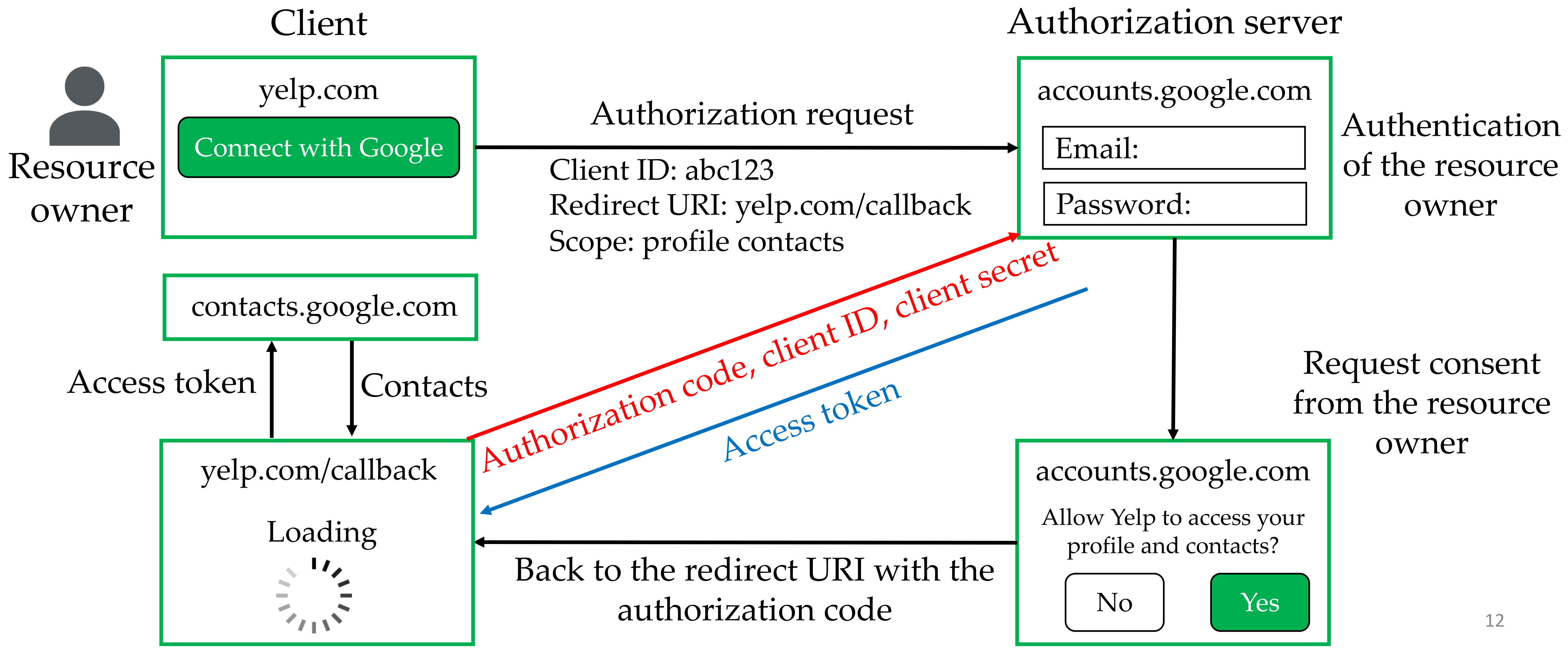
- The third-party app (e.g. the printing app) must be registered with the service provider (e.g. Google).
- The third-party app's developer must complete a **registration form** (usually in the developer page of the service's website) providing details about the app.

  Redirect URI (or **Callback URL**): where the service will redirect the user after they authorize (or deny) the third-party app → the part of the third-party app that handles authorization codes and access tokens.

- Once the third-party app is registered, the service issues client credentials:
  - **Client ID**: a public string used by the service to identify the client.
  - **Client secret**: a private password used to authenticate the identity of the client when the app requests to access a user's data

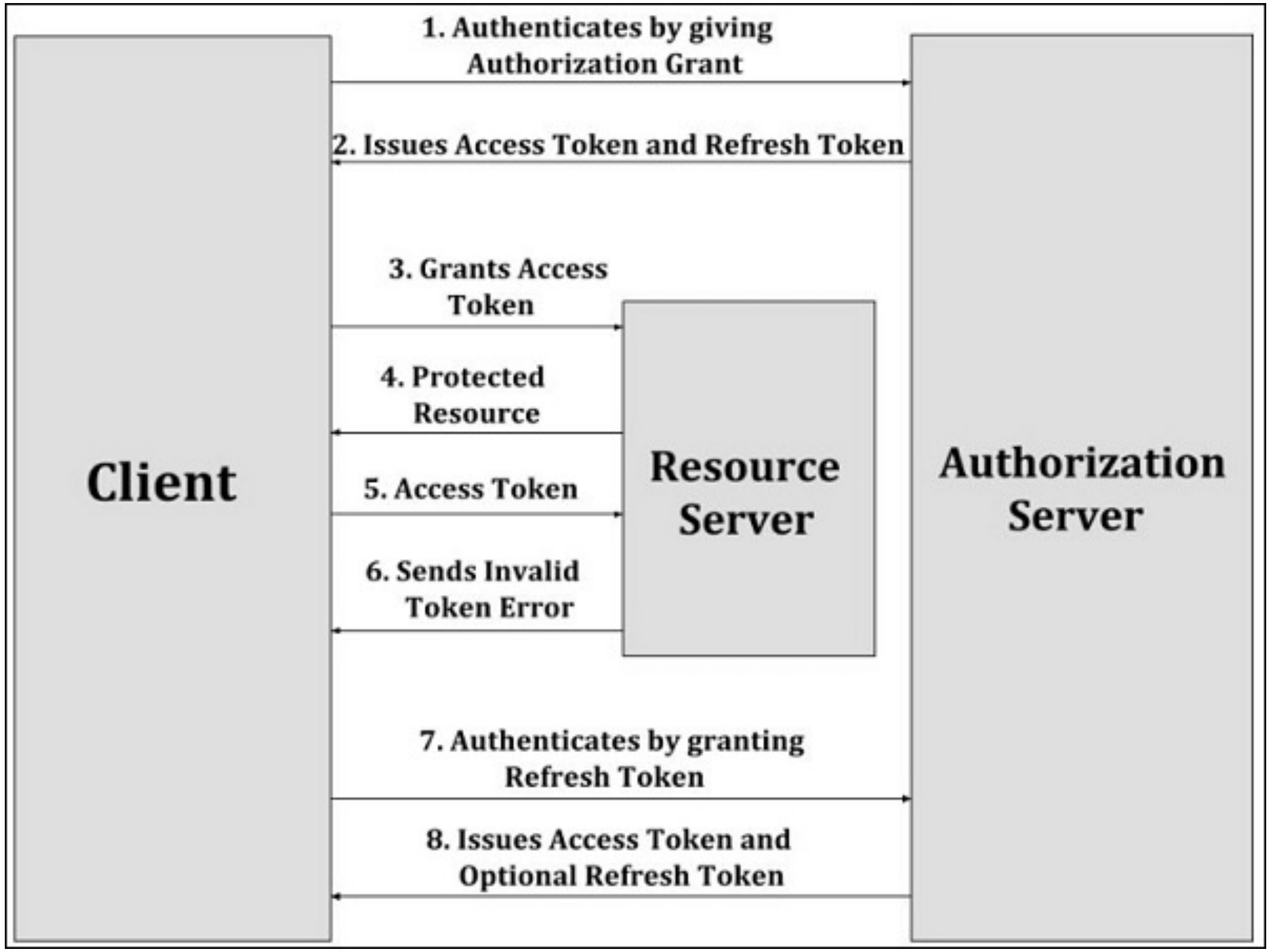
Example of delegated authentication with OAuth

Example: The app yelp.com wants to get access to a user's Google account profile contacts.



Refresh token

- Access token have an expiration time.
- A **refresh token** can be used by the client to request a new access token to the authorization server without the need to repeat the entire authorization process.


Client
Resource Server
Authorization Server
1. Authenticates by giving Authorization Grant
2. Issues Access Token and Refresh Token
3. Grants Access Token
4. Protected Resource
5. Access Token
6. Sends Invalid Token Error
7. Authenticates by granting Refresh Token
8. Issues Access Token and Optional Refresh Token

JSON Web token

- **JSON Web Token** (JWT): an Internet standard that defines a compact way for securely transmitting a token between parties as a JSON object. The token can be verified and trusted because it is digitally signed (e.g., using RSA with SHA-256).
- A JWT token consists of three parts separated by dots, a header, a payload, a signature, encoded in Base64Url (a type of encoding to represent binary data).

```
xxxxx.yyyyy.zzzzz
```

Header: the type of the token and the signing algorithm

{ "alg": “RS256", "typ": "JWT" }

JSON Web token

`xxxxx.yyyyy.zzzzz`

$\downarrow$

Payload: it contains claims, i.e., some statements.

```
{
"iss": "https://accounts.google.com",
"sub": "you@gmail.com",
"name": “John Smith"
"aud": "s6BhdRkqt3",
"exp": 1311281970,
"iat": 1311280970,
"auth_time": 1311280969
}
```

Standard payload claim fields

iss	Issuer	Identifies principal that issued the JWT.
sub	Subject	Identifies the subject of the JWT.
aud	Audience	Identifies the recipients that the JWT is intended for. If the principal processing the claim does not identify itself with a value in the aud claim when this claim is present, then the JWT must be rejected.
exp	Expiration Time	Identifies the expiration time on and after which the JWT must not be accepted for processing.
nbf	Not Before	Identifies the time on which the JWT will start to be accepted for processing.
iat	Issued at	Identifies the time at which the JWT was issued.

JSON Web token

xxxxx.yyyyy.zzzzz

Signature: it contains a signature of the header and payload

- The signature is calculated by these steps:
  - header and payload encoded using Base64url
  - concatenation of the encoded header and payload with a separating character
  - Encryption of the obtained string through the cryptographic algorithm specified in the header.

RS256(base64UrlEncode(header) + "." + base64UrlEncode(payload), signature_key)

Example of a JWT token

Header

eyJhbGciOiJSUzI1NiIsImtpZCI6IkRNa3Itd0JqRU1EYnhOY25xaVJISVhu
YUxubWI3UUpfWF9rWmJyaEtBMGMifQ
.

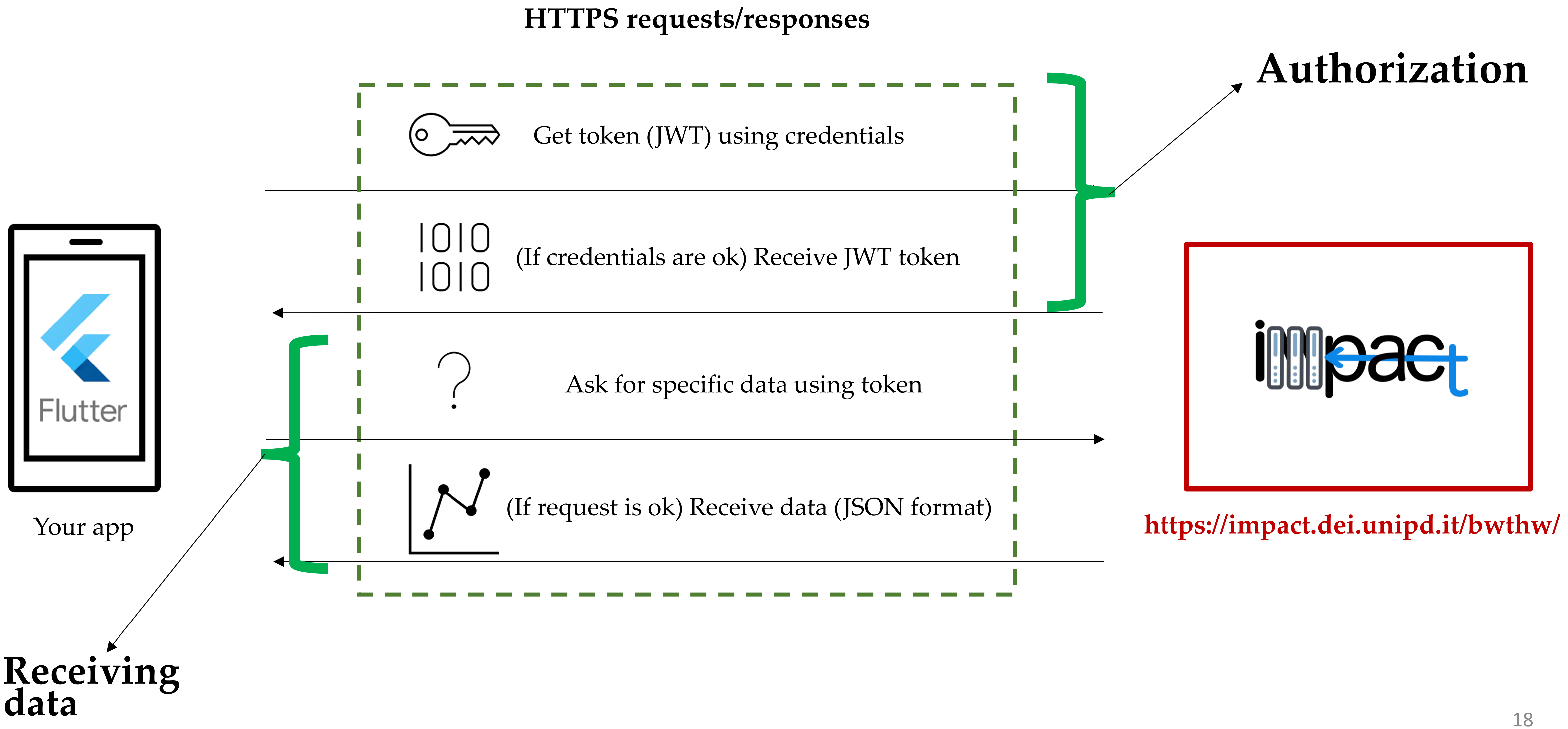
Payload

eyJzdWIiOiIwMHU5bzFuaWtqdk9CZzVabzBoNyIsInZlciI6MSwiaXNzIjoi
aHR0cHM6Ly9kZXYtMzQxNjA3Lm9rdGFwcmV2aWV3LmNvbS9vYXV0aDIvYXVz
OW84d3ZraG9ja3c5VEwwaDciLCJhdWQiOiJsWFNlbkx4eFBpOGtRVmpKRTVz
NCIsImlhdCI6MTUwOTA0OTg5OCwiZXhwIjoxNTA5MDUzNDk4LCJqdGkiOiJJ
RC5oa2RXSXNBSXZTbnBGYVFHTVRYUGNVSmhhMkgwS2c5Ykl3ZEVvVm1ZZHN3
IiwiYW1yIjpbImtiYSIsIm1mYSIsInB3ZCJdLCJpZHAiOiIwMG85bzFuaWpr
aWpLeGNpbjBoNyIsIm5vbmNlIjoidWpwMmFzeHlqN2UiLCJhdXRoX3RpbWUi
OjE1MDkwNDk3MTl9
.

Signature

dv4Ek8B4BDee1PcQT_4zm7kxDEY1sRIGbLoNtlodZcSzHzXU5GkKyl6sAVmdXOIPUlAIrJA
hNfQWQ-
_XZLBVPjETiZE8CgNg5uqNmeXMUnYnQmvN5oWlXUZ8Gcub-GAbJ8-
NQuyBmyec1j3gmGzX3wemke8NkuI6SX2L4Wj1PyvkknBtbjfiF9ud1-
ERKbobaFbnjDFOFTzvL6g34SpMmZWy6uc_Hs--n4IC-ex-
Ps3FcMwRggCW-
7o2FpH6rJTOGPZYrOx44n3ZwAu2dGm6axtPIsqU8b6sw7DaHpogD_hxsXgMIOzOBMbYsQ
EiczoGn71ZFz_1O7FiW4dH6g

How authorization works in the IMPACT backend


HTTPS requests/responses
Authorization
Get token (JWT) using credentials
(If credentials are ok) Receive JWT token
Ask for specific data using token
(If request is ok) Receive data (JSON format)
Flutter
Your app
https://impact.dei.unipd.it/bwthw/
Receiving data

The IMPACT user types and permissions

Patient
Generates and manages its data

Clinician
Enrolls patients, reviews data of patients in its clinical center, edits clinical study settings

Researcher
Accesses all data in read-only mode

Superuser/Study Administrator/Data Administrator
They have God-like powers

The researcher role

- Each group will have access to the data using the researcher role

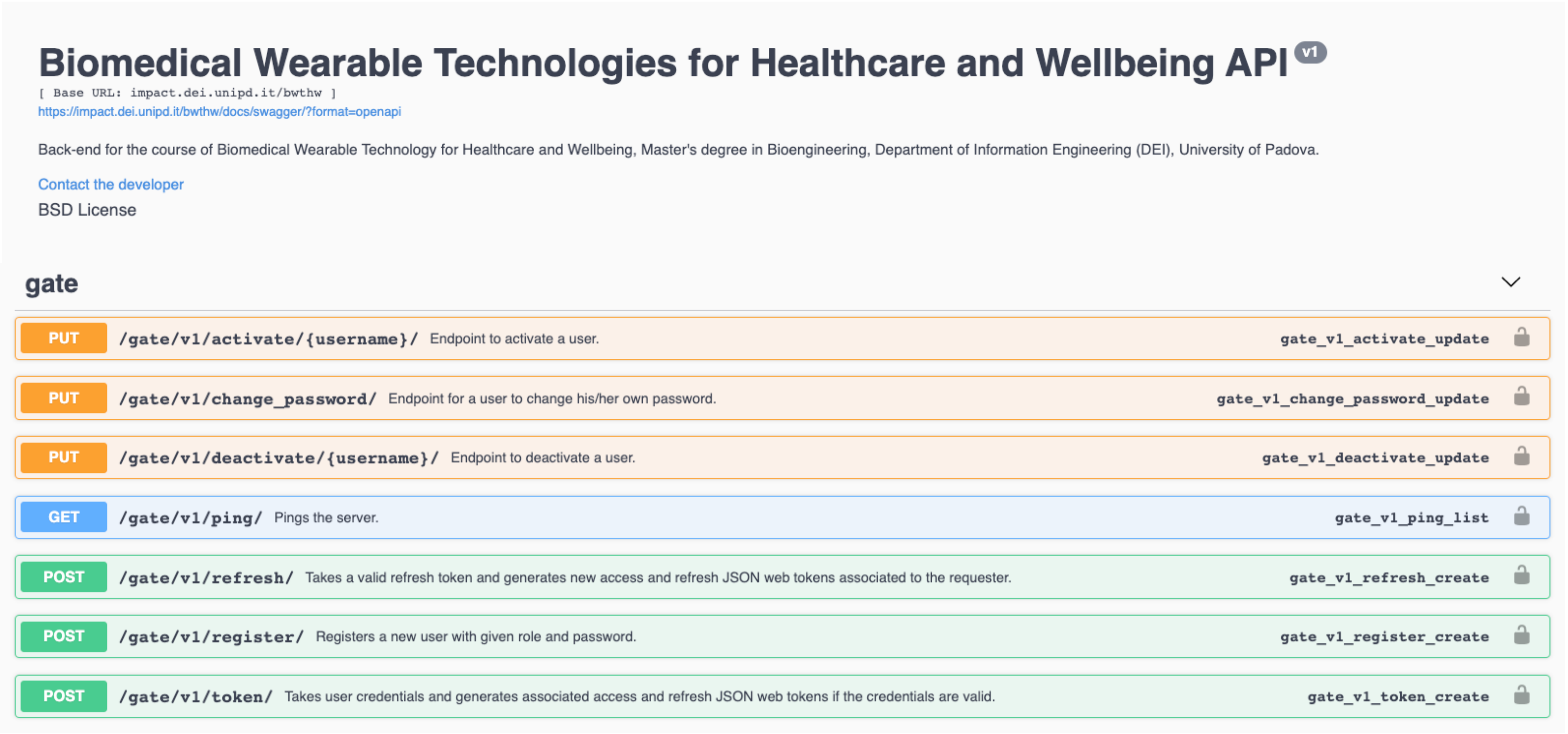
```
{
"username": ”<YOUR_USERNAME>",
"access_expiration_date": “<YOUR_ACCESS_EXPIRATION_DATE>”
}
```

- Each researcher in IMPACT is characterized by a username and an access expiration date.
- The access expiration date defines the date until a specific researcher is allowed to access data stored in the IMPACT database. Two examples:

```
{
"username": "nmRTl0v7W8",
"access_expiration_date": "2024-09-30"
}
```

```
{
"username": "nmRTl0v7W8",
"access_expiration_date": null
}
```

Get the authorization: The IMPACT gate



Biomedical Wearable Technologies for Healthcare and Wellbeing API v1

[Base URL: impact.dei.unipd.it/bwthw]
https://impact.dei.unipd.it/bwthw/docs/swagger/?format=openapi

Back-end for the course of Biomedical Wearable Technology for Healthcare and Wellbeing, Master's degree in Bioengineering, Department of Information Engineering (DEI), University of Padova.

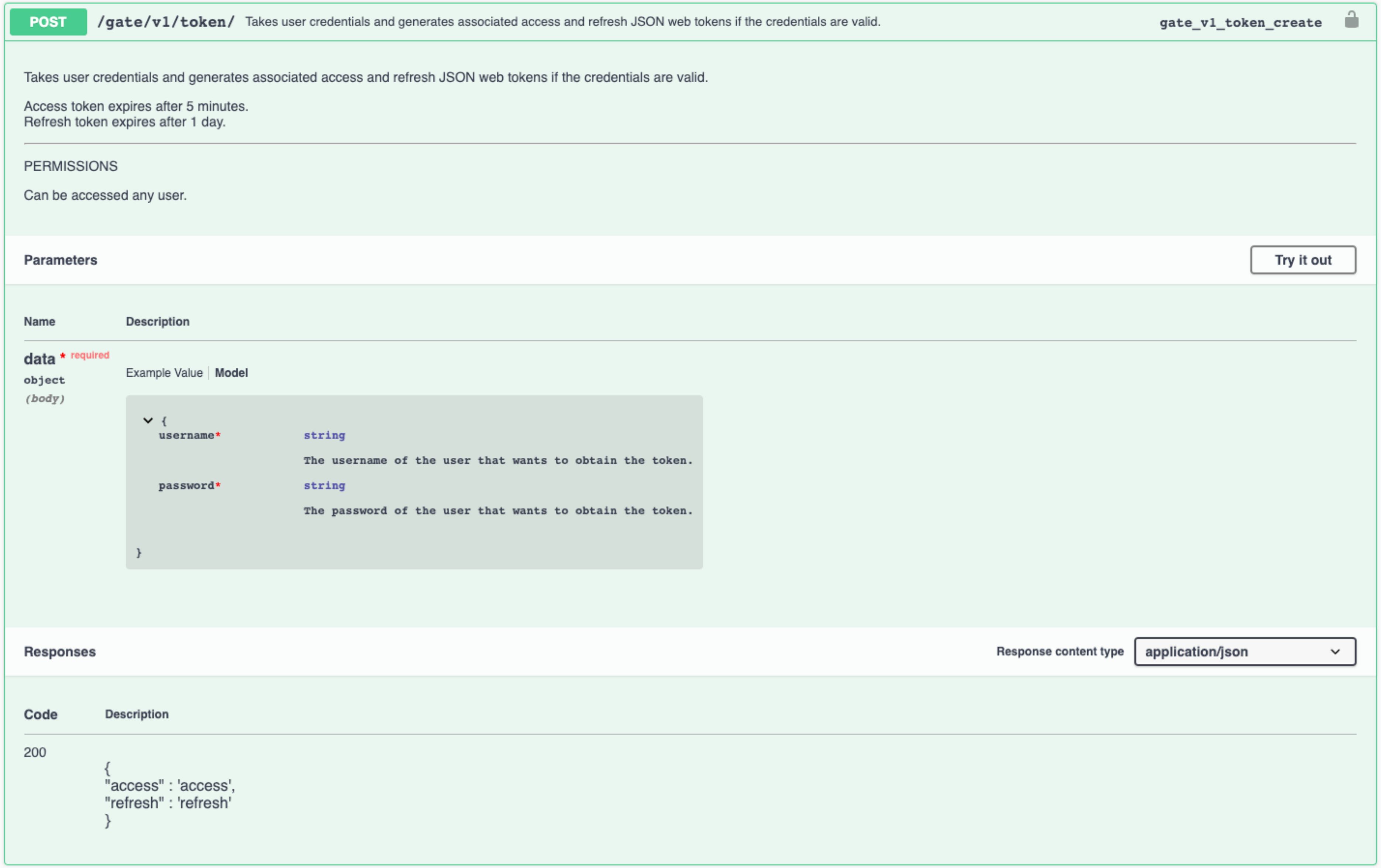
Contact the developer
BSD License

gate

PUT	/gate/v1/activate/{username}/	Endpoint to activate a user.	gate_v1_activate_update
PUT	/gate/v1/change_password/	Endpoint for a user to change his/her own password.	gate_v1_change_password_update
PUT	/gate/v1/deactivate/{username}/	Endpoint to deactivate a user.	gate_v1_deactivate_update
GET	/gate/v1/ping/	Pings the server.	gate_v1_ping_list
POST	/gate/v1/refresh/	Takes a valid refresh token and generates new access and refresh JSON web tokens associated to the requester.	gate_v1_refresh_create
POST	/gate/v1/register/	Registers a new user with given role and password.	gate_v1_register_create
POST	/gate/v1/token/	Takes user credentials and generates associated access and refresh JSON web tokens if the credentials are valid.	gate_v1_token_create

The IMPACT gate: token

- This endpoint allows to get the JWT token using your credentials
- It needs 2 parameter provided in the request body
- If successful (200) it will return a JSON containing the access and the refresh token pair


POST /gate/v1/token/ Takes user credentials and generates associated access and refresh JSON web tokens if the credentials are valid. gate_v1_token_create

Takes user credentials and generates associated access and refresh JSON web tokens if the credentials are valid.

Access token expires after 5 minutes.
Refresh token expires after 1 day.

PERMISSIONS

Can be accessed any user.

Parameters

Name	Description
data * required object (body)	Example Value | Model

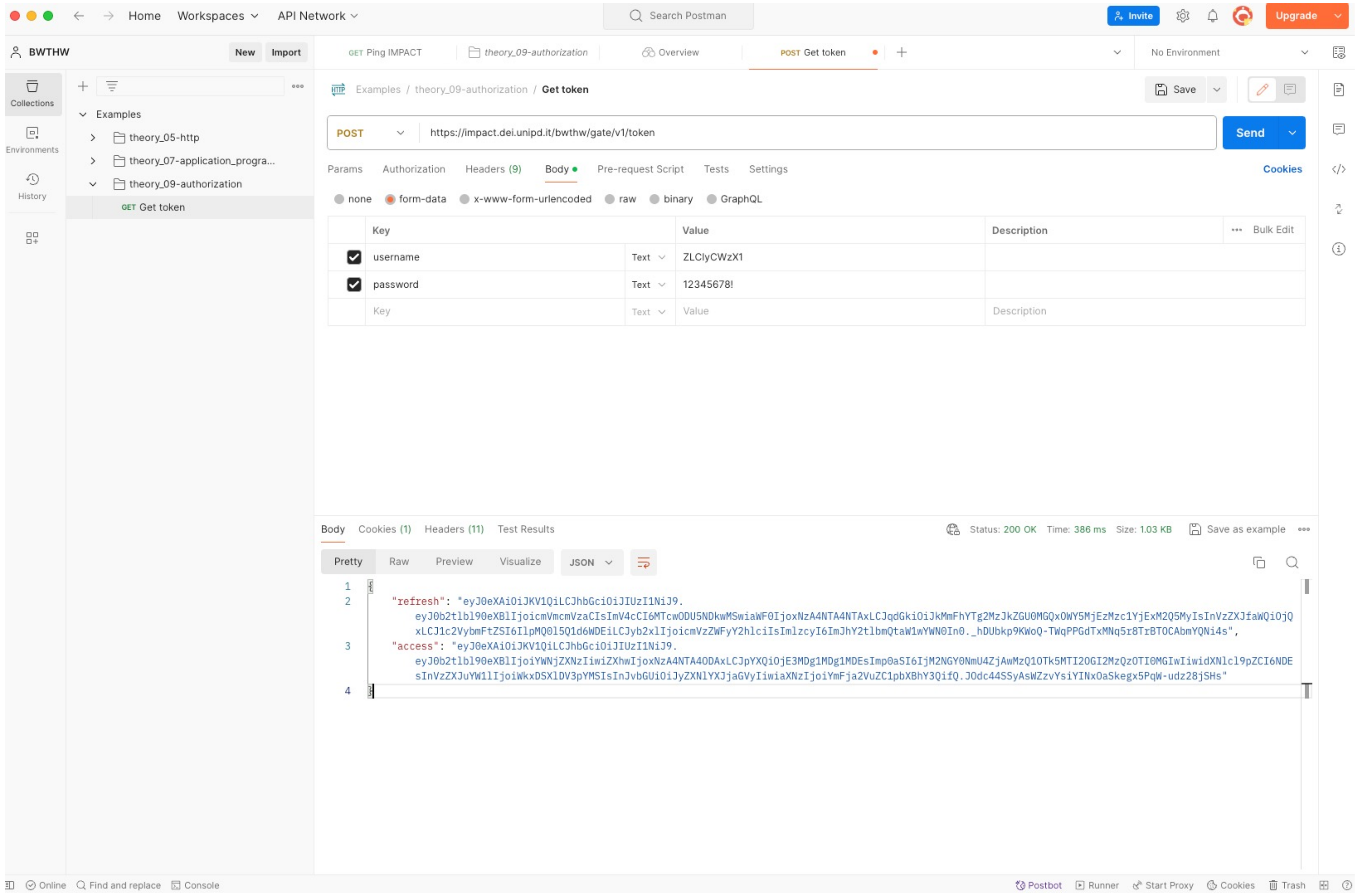
```
{
username*           string
                    The username of the user that wants to obtain the token.
password*           string
                    The password of the user that wants to obtain the token.
}
```

Responses — Response content type: application/json

Code	Description
200	{ "access" : 'access', "refresh" : 'refresh' }

Get the token using POSTMAN

```
{
"username": "ZLCIyCWzX1",
”access_expiration_date":
2024–06–30
}
```



POST https://impact.dei.unipd.it/bwthw/gate/v1/token

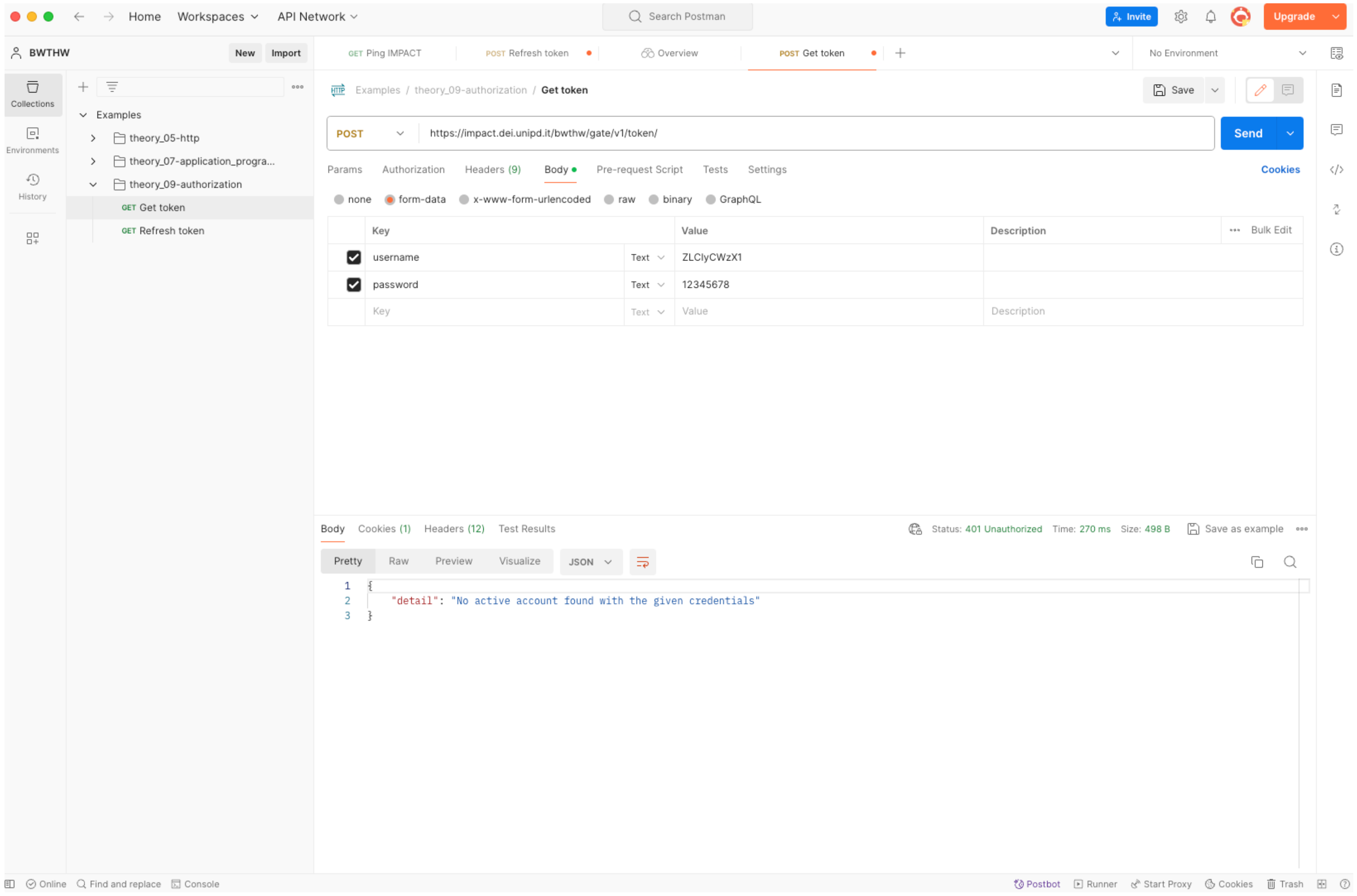
Body: form-data

	Key		Value	Description
✓	username	Text	ZLCIyCWzX1	
✓	password	Text	12345678!	
	Key	Text	Value	Description

Status: 200 OK Time: 386 ms Size: 1.03 KB

```
{
    "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
        eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTcwODU5NDkwMSwiaWF0IjoxNzA4NTA4NTAxLCJqdGkiOiJkMmFhYTg2MzJkZGU0MGQxOWY5MjEzMzc1YjExM2Q5MyIsInVzZXJfaWQiOjQ
        xLCJ1c2VybmFtZSI6IlpMQ0l5Q1d6WDEiLCJyb2xlIjoicmVzZWFyY2hlciIsImlzcyI6ImJhY2tlbmQtaW1wYWN0In0._hDUbkp9KWoQ-TWqPPGdTxMNq5r8TrBTOCAbmYQNi4s",
    "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
        eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNzA4NTA4ODAxLCJpYXQiOjE3MDg1MDg1MDEsImp0aSI6IjM2NGY0NmU4ZjAwMzQ1OTk5MTI2OGI2MzQzOTI0MGIwIiwidXNlcl9pZCI6NDE
        sInVzZXJuYW1lIjoiWkxDSXlDV3pYMSIsInJvbGUiOiJyZXNlYXJjaGVyIiwiaXNzIjoiYmFja2VuZC1pbXBhY3QifQ.JOdc44SSyAsWZzvYsiYINxOaSkegx5PqW-udz28jSHs"
}
```

(Wrongly) Get the token using POSTMAN


Examples / theory_09-authorization / Get token

POST https://impact.dei.unipd.it/bwthw/gate/v1/token/

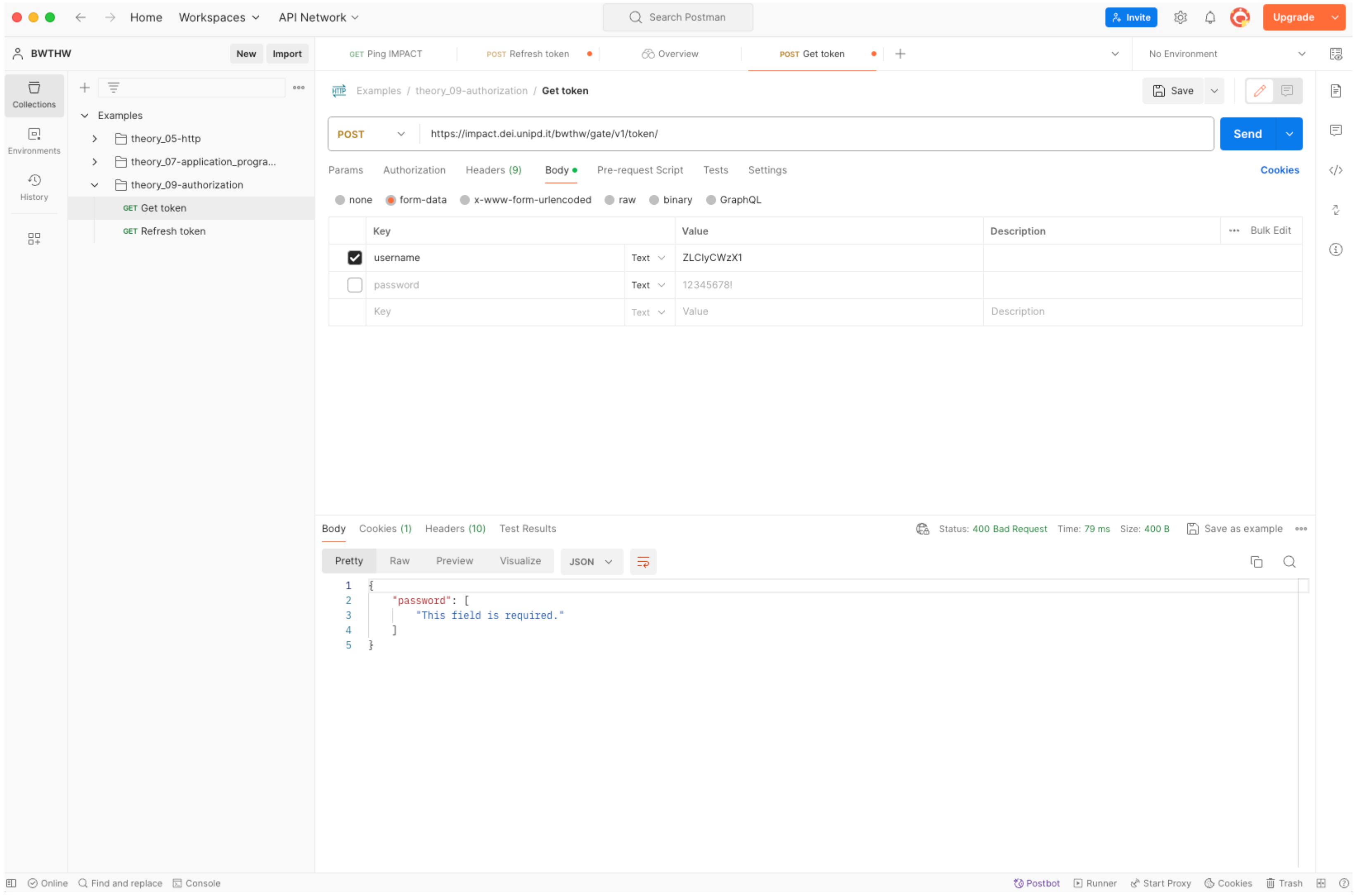
Body: none [✓ form-data] x-www-form-urlencoded raw binary GraphQL

	Key		Value	Description
✓	username	Text	ZLClyCWzX1	
✓	password	Text	12345678	

Status: 401 Unauthorized Time: 270 ms Size: 498 B

```
{
    "detail": "No active account found with the given credentials"
}
```

(Wrongly) Get the token using POSTMAN



Examples / theory_09-authorization / Get token

POST https://impact.dei.unipd.it/bwthw/gate/v1/token/

Body: form-data

	Key		Value	Description
✓	username	Text	ZLClyCWzX1	
	password	Text	12345678!	
	Key	Text	Value	Description

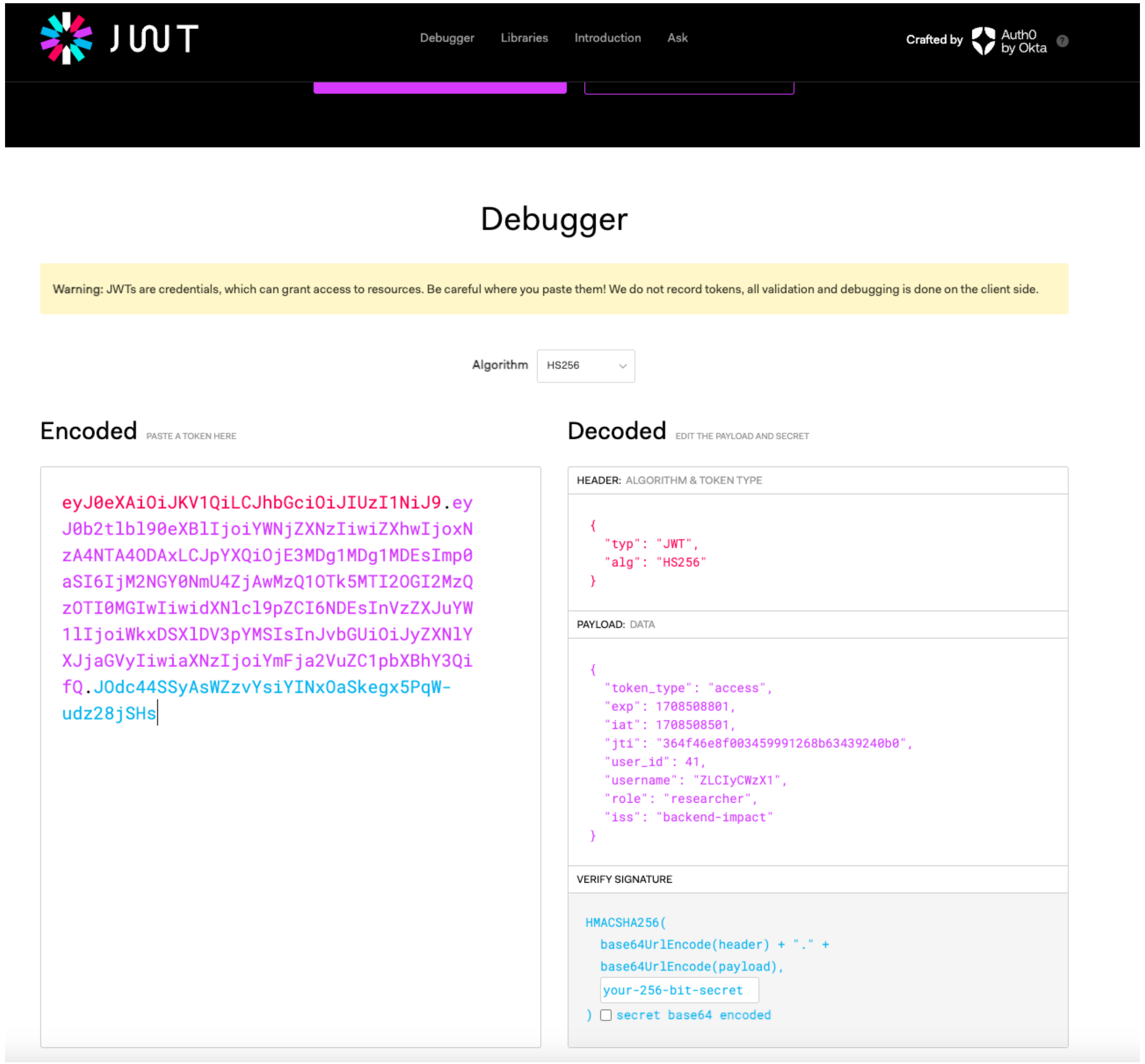
Body Cookies (1) Headers (10) Test Results — Status: 400 Bad Request Time: 79 ms Size: 400 B

```
{
    "password": [
        "This field is required."
    ]
}
```

Inspect the access token

- We can inspect the content of the JWT token using an handful online tool:

https://jwt.io



Debugger

Warning: JWTs are credentials, which can grant access to resources. Be careful where you paste them! We do not record tokens, all validation and debugging is done on the client side.

Algorithm HS256

Encoded PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ey
J0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxN
zA4NTA4ODAxLCJpYXQiOjE3MDg1MDg1MDEsImp0
aSI6IjM2NGY0NmU4ZjAwMzQ1OTk5MTI2OGI2MzQ
zOTI0MGIwIiwidXNlcl9pZCI6NDEsInVzZXJuYW
1lIjoiWkxDSXlDV3pYMSIsInJvbGUiOiJyZXNlY
XJjaGVyIiwiaXNzIjoiYmFja2VuZC1pbXBhY3Qi
fQ.JOdc44SSyAsWZzvYsiYINxOaSkegx5PqW-
udz28jSHs
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

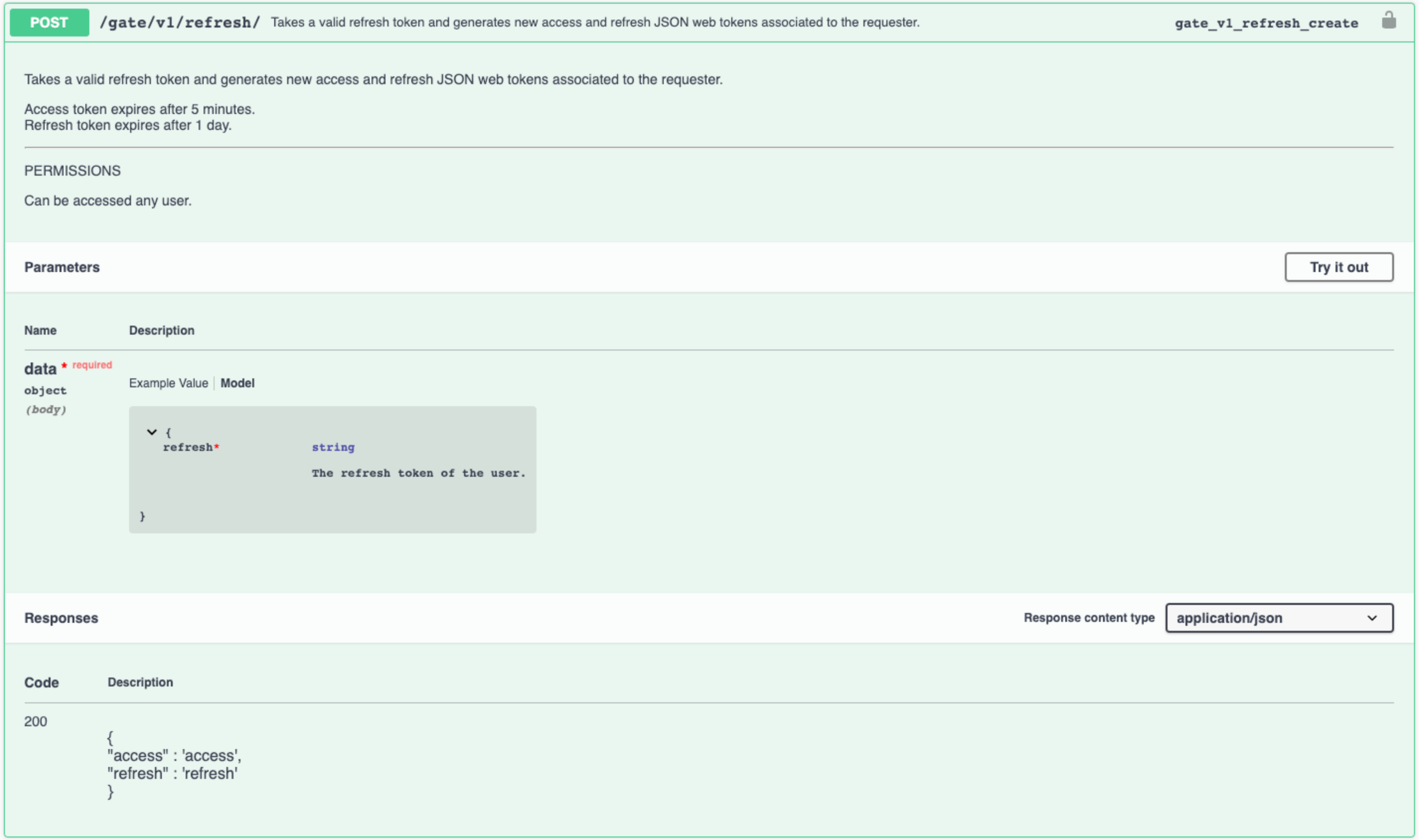
PAYLOAD: DATA

```
{
  "token_type": "access",
  "exp": 1708508801,
  "iat": 1708508501,
  "jti": "364f46e8f003459991268b63439240b0",
  "user_id": 41,
  "username": "ZLCIyCWzX1",
  "role": "researcher",
  "iss": "backend-impact"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

The IMPACT gate: refresh



POST /gate/v1/refresh/ Takes a valid refresh token and generates new access and refresh JSON web tokens associated to the requester. gate_v1_refresh_create

Takes a valid refresh token and generates new access and refresh JSON web tokens associated to the requester.

Access token expires after 5 minutes.
Refresh token expires after 1 day.

PERMISSIONS

Can be accessed any user.

Parameters

Name	Description
data * required object *(body)*	Example Value | **Model**

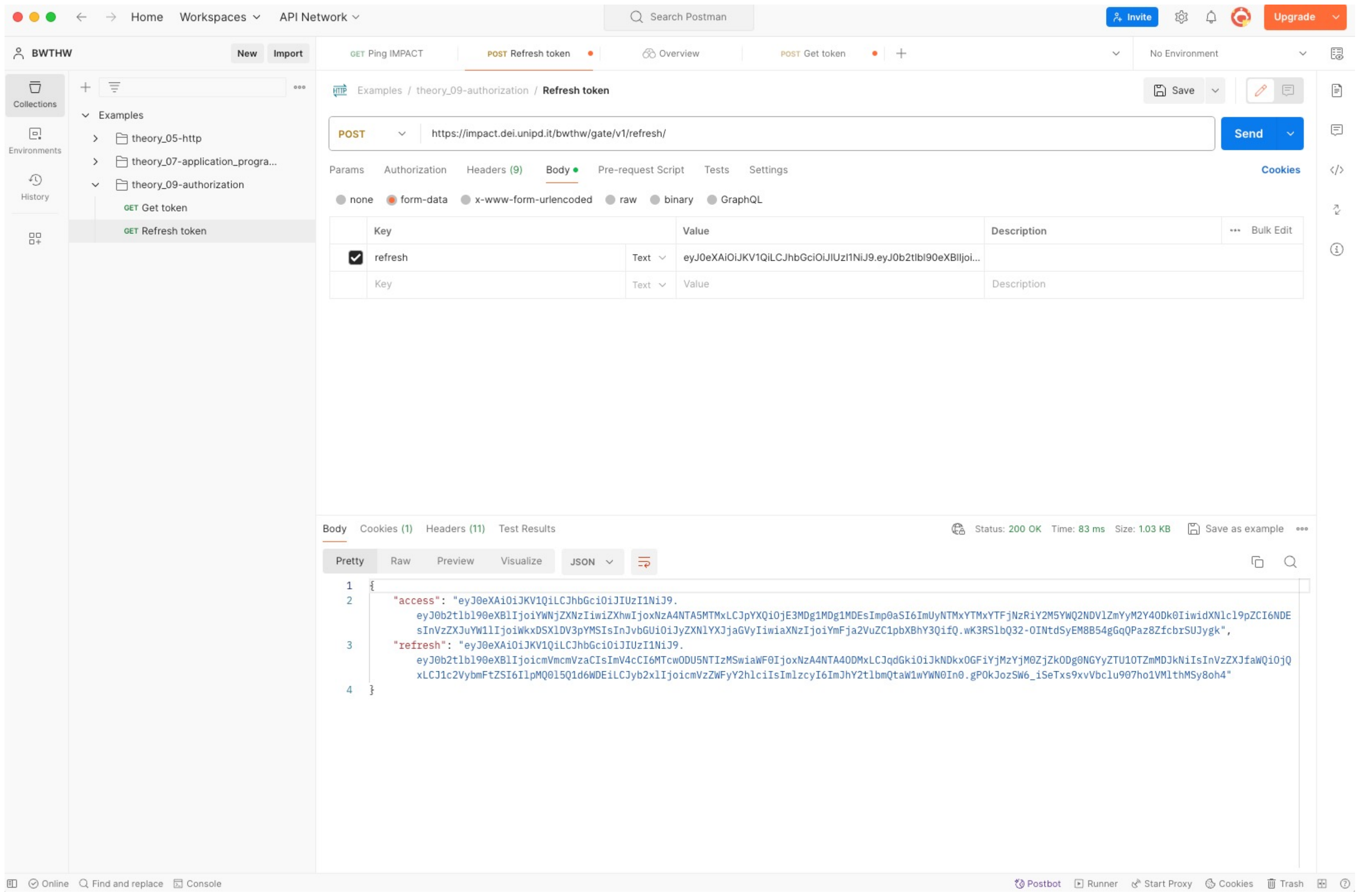
```
{
  refresh*            string
                      The refresh token of the user.
}
```

Responses

Response content type: application/json

Code	Description
200	{ "access" : 'access', "refresh" : 'refresh' }

Refresh the token using POSTMAN



Examples / theory_09-authorization / **Refresh token**

POST https://impact.dei.unipd.it/bwthw/gate/v1/refresh/

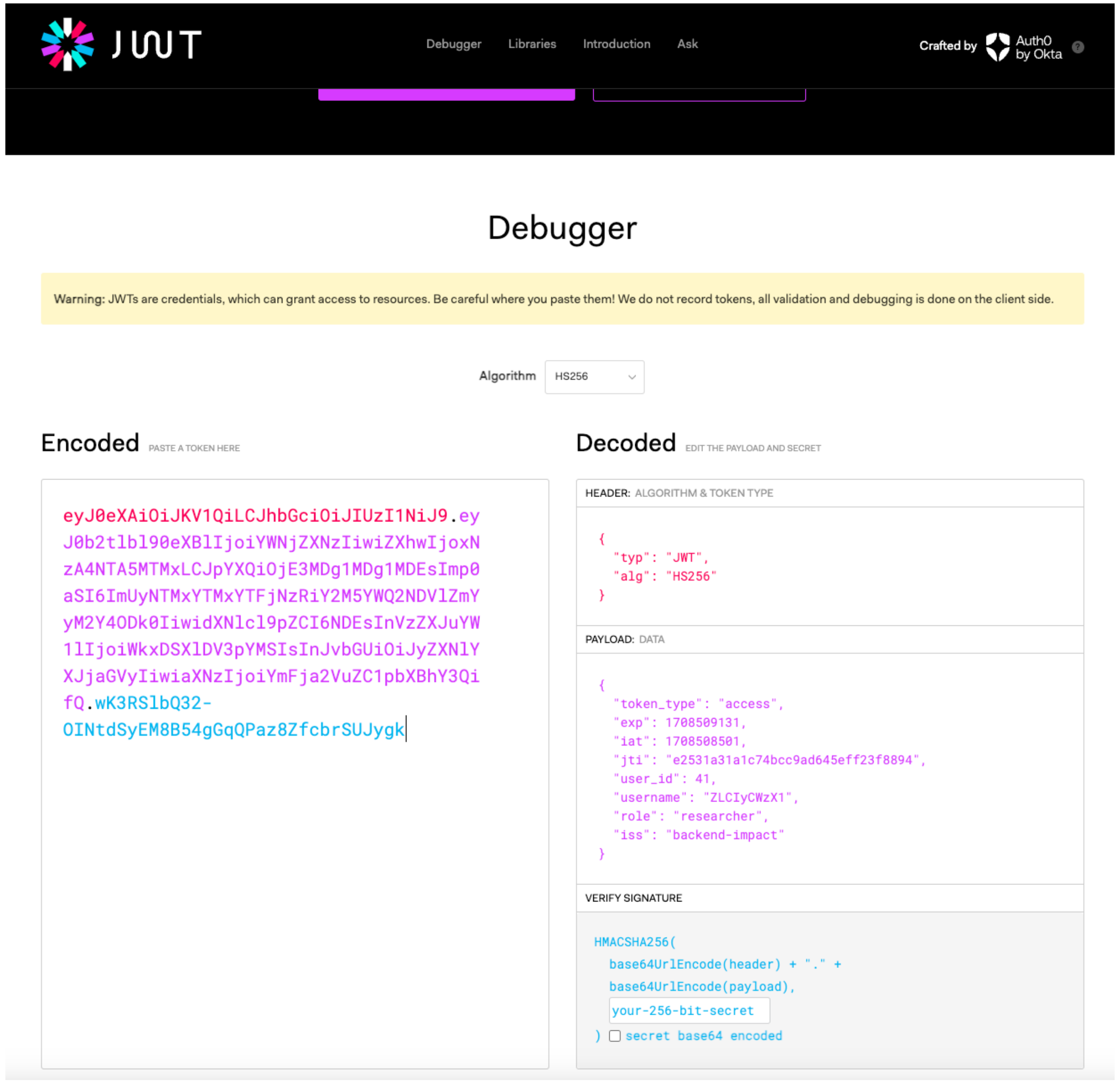
Body: form-data

Key	Value	Description
✓ refresh (Text)	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2tlbl90eXBlIjoi...	

Status: 200 OK Time: 83 ms Size: 1.03 KB

```
{
    "access": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
        eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNzA4NTA5MTMxLCJpYXQiOjE3MDg1MDg1MDEsImp0aSI6ImUyNTMxYTMxYTFjNzRiY2M5YWQ2NDVlZmYyM2Y4ODk0IiwidXNlcl9pZCI6NDE
        sInVzZXJuYW1lIjoiWkxDSXlDV3pYMSIsInJvbGUiOiJyZXNlYXJjaGVyIiwiaXNzIjoiYmFja2VuZC1pbXBhY3QifQ.wK3RSlbQ32-OINtdSyEM8B54gGqQPaz8ZfcbrSUJygk",
    "refresh": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
        eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTcwODU5NTIzMSwiaWF0IjoxNzA4NTA4ODMxLCJqdGkiOiJkNDkxOGFiYjMzYjM0ZjZkODg0NGYyZTU1OTZmMDJkNiIsInVzZXJfaWQiOjQ
        xLCJ1c2VybmFtZSI6IlpMQ0l5Q1d6WDEiLCJyb2xlIjoicmVzZWFyY2hlciIsImlzcyI6ImJhY2tlbmQtaW1wYWN0In0.gPOkJozSW6_iSeTxs9xvVbclu907ho1VMlthMSy8oh4"
}
```

Inspect the new access token



Debugger

Warning: JWTs are credentials, which can grant access to resources. Be careful where you paste them! We do not record tokens, all validation and debugging is done on the client side.

Algorithm HS256

Encoded PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ey
J0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxN
zA4NTA5MTMxLCJpYXQiOjE3MDg1MDg1MDEsImp0
aSI6ImUyNTMxYTMxYTFjNzRiY2M5YWQ2NDVlZmY
yM2Y4ODk0IiwidXNlcl9pZCI6NDEsInVzZXJuYW
1lIjoiWkxDSXlDV3pYMSIsInJvbGUiOiJyZXNlY
XJjaGVyIiwiaXNzIjoiYmFja2VuZC1pbXBhY3Qi
fQ.wK3RSlbQ32-
OINtdSyEM8B54gGqQPaz8ZfcbrSUJygk
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

PAYLOAD: DATA

```
{
  "token_type": "access",
  "exp": 1708509131,
  "iat": 1708508501,
  "jti": "e2531a31a1c74bcc9ad645eff23f8894",
  "user_id": 41,
  "username": "ZLCIyCWzX1",
  "role": "researcher",
  "iss": "backend-impact"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

References

- Tanenbaum, Wetherall – Computer Networks – Fifth Edition
  - Chapter 8 – Network security

- jwt.io: https://jwt.io