

Biomedical Wearable Technologies
for Healthcare and Wellbeing

Authentication

A.Y. 2023-2024

Giacomo Cappon



Outline

- **Recap**

- The flow

- RESTful API in practice

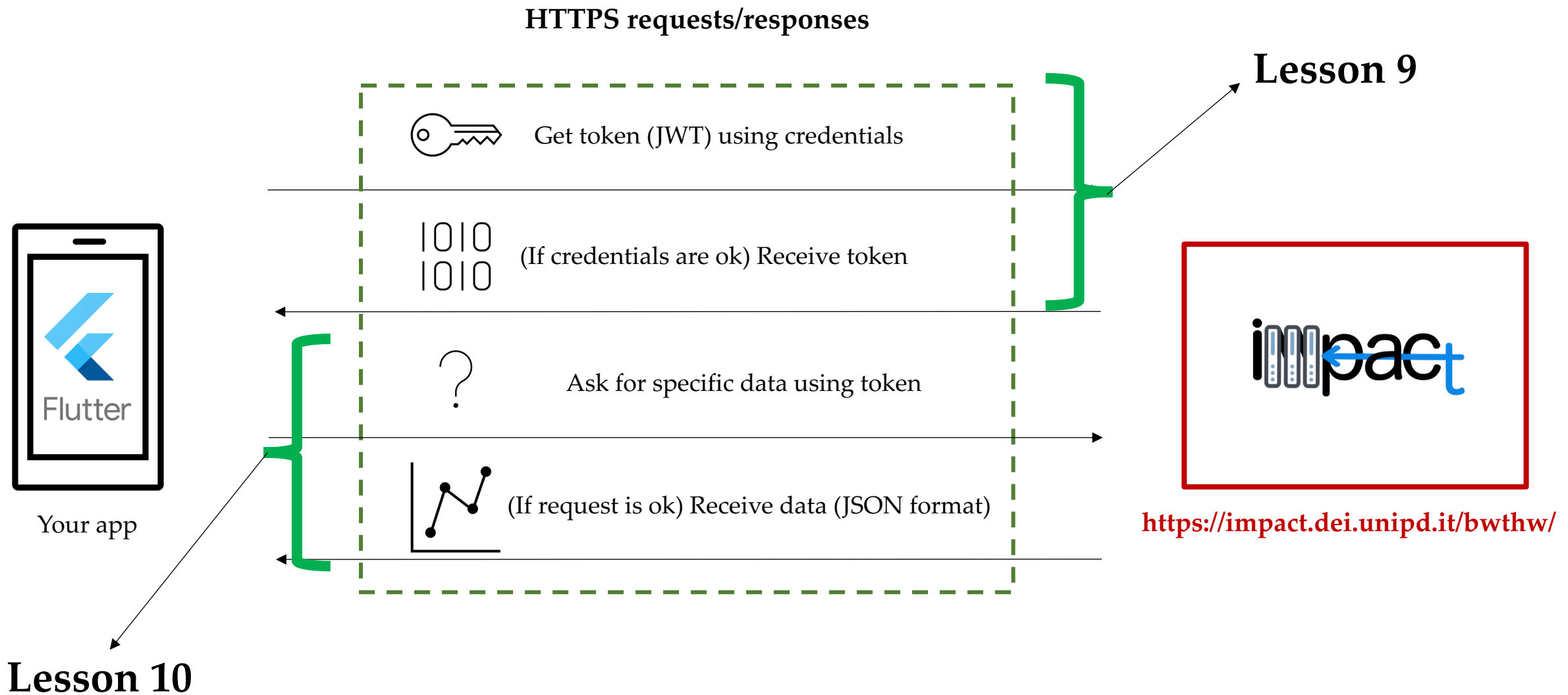
- Case study

- Exercise

- Homework

- Resources

Recap: The IMPACT backend

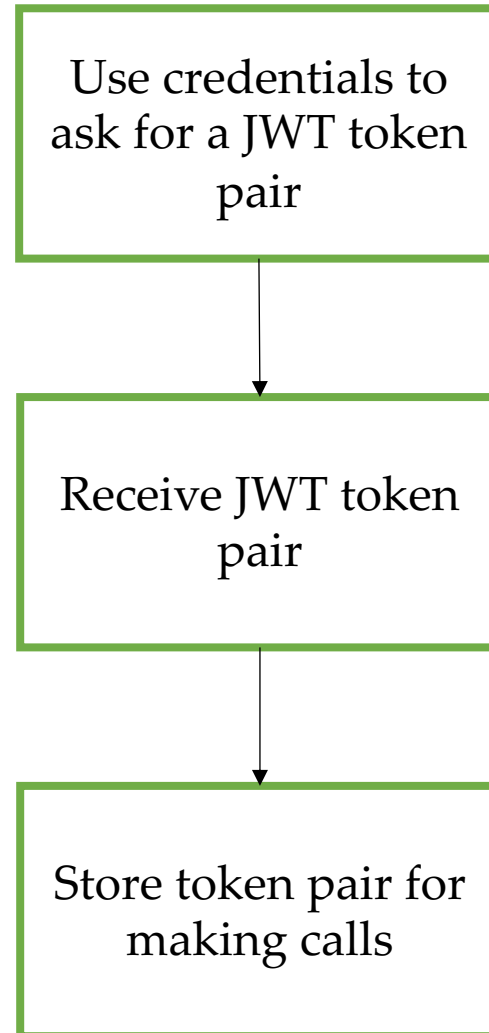


Outline

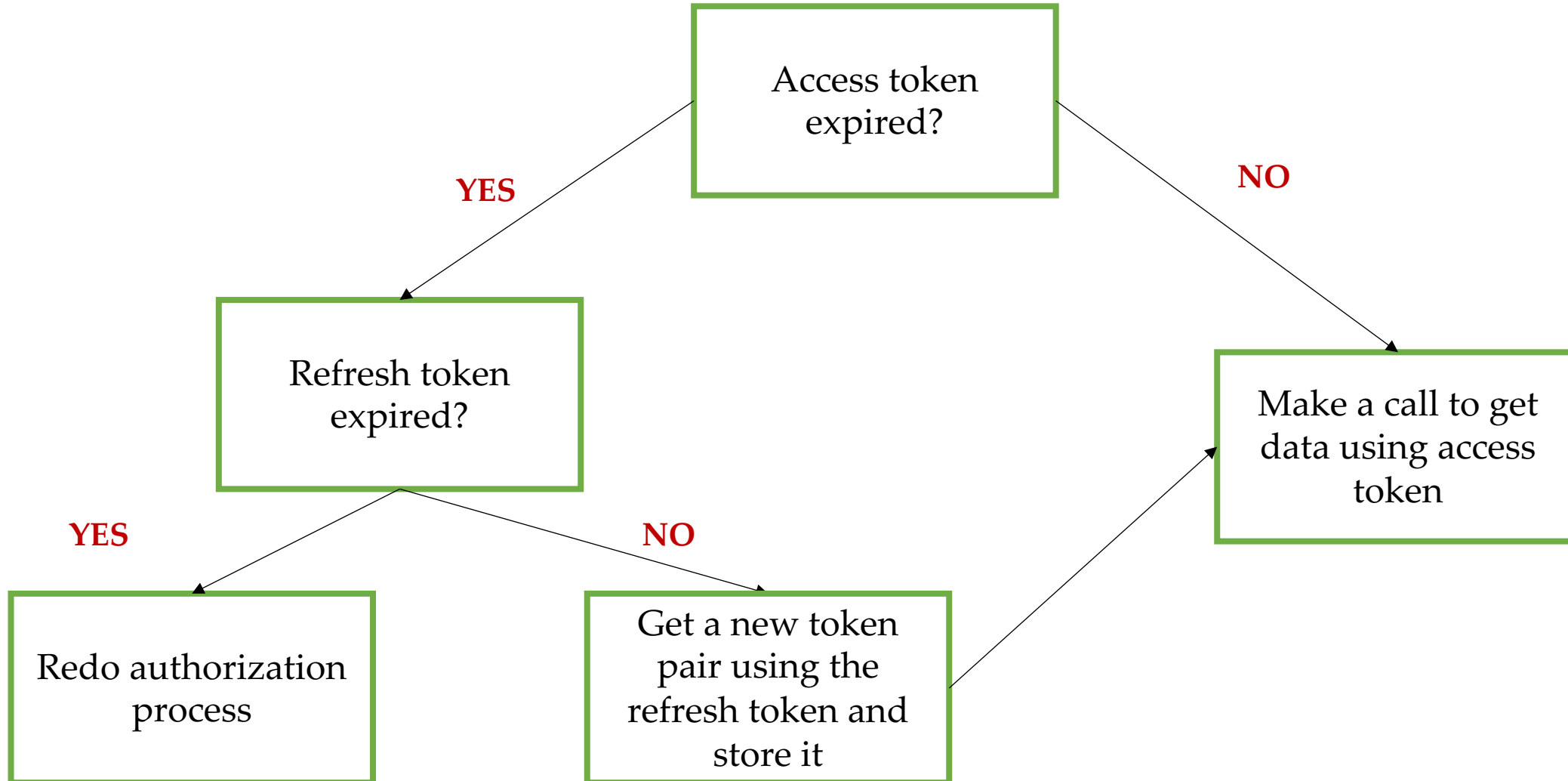
- Recap
- **The flow**
- RESTful API in practice
- Case study

- Exercise
- Homework
- Resources

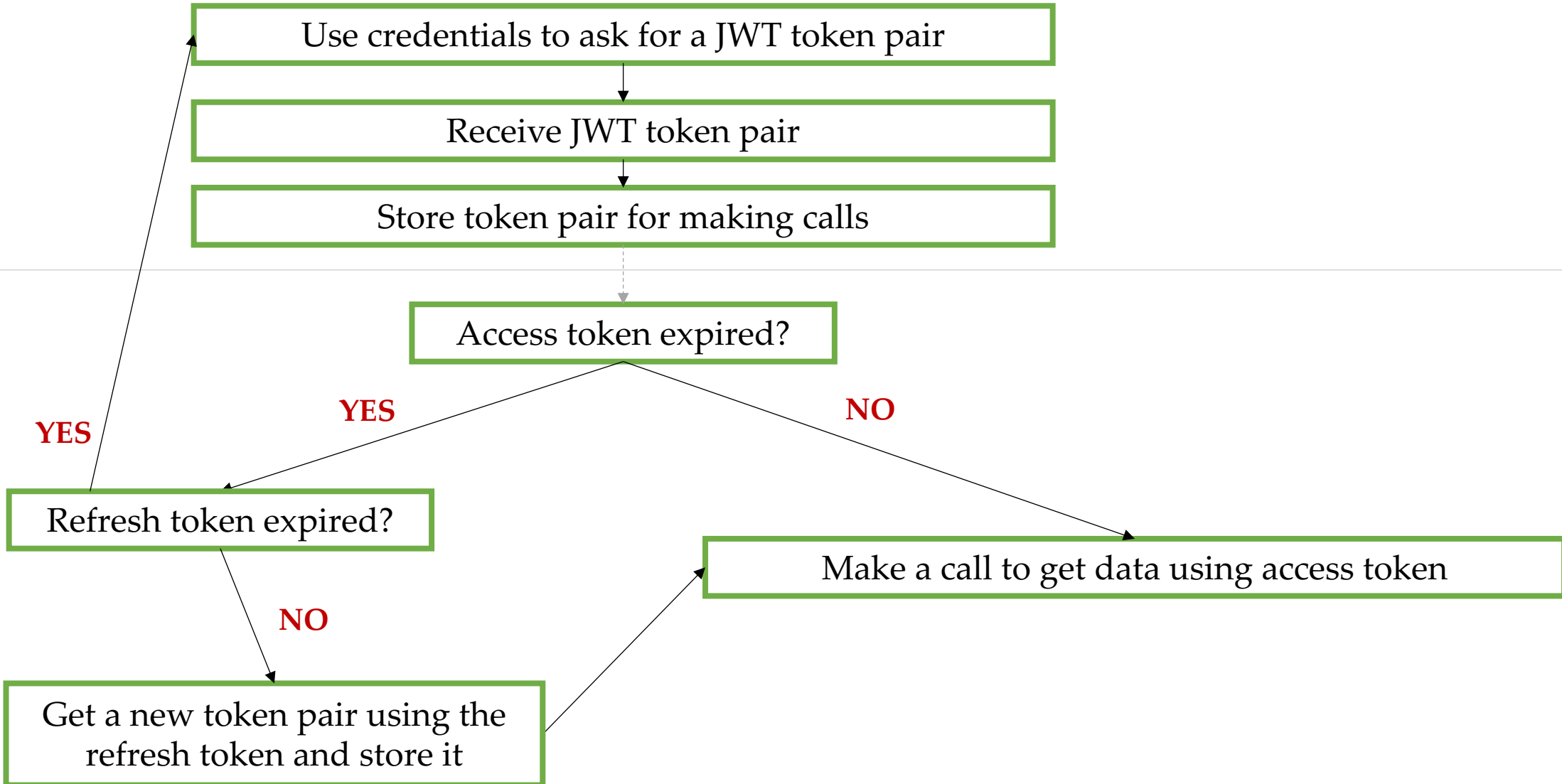
The (JWT) authorization flow in practice



“Obtaining data” flow in practice



The entire flow in practice



Outline

- Recap
- The flow
- **RESTful API in practice**
- Case study

- Exercise
- Homework
- Resources

RESTful API in practice

- From the practical point of view, RESTful API can be used via http/https following three steps:
 - Step 1: Send an http/https request to the RESTful API
 - Step 2: Await for the response
 - Step 3: Process the response (usually in a JSON format)
- The http/https request has the following structure:

<METHOD> <HTTP or HTTPS>://<DOMAIN>/<ENDPOINT>?<PARAMETERS>
+
<BODY> and <HEADERS>

- <METHOD> and <ENDPOINT> defines the so-called **route**, e.g.:
GET /heartrate/today/
DELETE /user/1

RESTful API in practice

- Beside its content a response contains an **HTTP status code**, i.e., a special number that tells to the frond-end if the request is successful or, otherwise, why it is not successful. Here's the most common:
 - 200: OK
 - 401: UNAUTHORIZED
 - 403: FORBIDDEN
 - 404: NOT FOUND
 - 500: INTERNAL SERVER ERROR
- Normally, the front-end developer has to manage these codes based on the API specifics

Outline

- Recap
- The flow
- RESTful API in practice
- **Case study**
- Exercise
- Homework
- Resources

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: ping

GET **/gate/v1/ping/** Pings the server. gate_v1_ping_list

Pings the server.

PERMISSIONS

- Can be accessed by everyone.

ERRORS

None


Parameters Try it out

No parameters

Responses Response content type **application/json**

Code	Description
200	<pre>{ "status": "success", "code": 200, "message": "Request successful.", "data": "pong", }</pre>

The IMPACT gate: token

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 Try it out

Name	Description
data <small>★ required</small>	Example Value Model
object (body)	<div><div>▼ {</div><div><div>username★</div><div>string</div><div>The username of the user that wants to obtain the token.</div></div><div><div>password★</div><div>string</div><div>The password of the user that wants to obtain the token.</div></div></div> <div>}</div>

Responses Response content type **application/json** ▼

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 Try it out

Name	Description
data <small>★ required</small> object <i>(body)</i>	Example Value Model <div><pre>▼ { refresh★ string The refresh token of the user. }</pre></div>

Responses Response content type **application/json** ▼

Code	Description
200	<pre>{ "access" : 'access', "refresh" : 'refresh' }</pre>

http package

- To be able to make calls we will use the http package. This provides a simple web client to be used to make http calls.

http 1.2.1 

Published 2 months ago •  dart.dev [Dart 3 compatible](#)

[SDK](#) [DART](#) [FLUTTER](#) [PLATFORM](#) [ANDROID](#) [IOS](#) [LINUX](#) [MACOS](#) [WEB](#) [WINDOWS](#)

 7.4K

[Readme](#) [Changelog](#) [Example](#) [Installing](#) [Versions](#) [Scores](#)

pub v1.2.1 publisher dart.dev

A composable, Future-based library for making HTTP requests.

This package contains a set of high-level functions and classes that make it easy to consume HTTP resources. It's multi-platform (mobile, desktop, and browser) and supports multiple implementations.

Using

The easiest way to use this library is via the top-level functions. They allow you to make individual HTTP requests with minimal hassle:

```
import 'package:http/http.dart' as http;

var url = Uri.https('example.com', 'whatsit/create');
var response = await http.post(url, body: {'name': 'doodle', 'color': 'blue'});
print('Response status: ${response.statusCode}');
print('Response body: ${response.body}');

print(await http.read(Uri.https('example.com', 'foobar.txt')));
```


Android-specific action

- To be able to access to internet functionalities in Android you are required to provide a specific permission:
- To do so, in the android>app>src>main folder open the AndroidManifest.xml file, and add the following after the <manifest ...> tag:

```
<manifest xmlns:android...>
```

```
...
```

```
<uses-permission android:name="android.permission.INTERNET" />
```

```
<application ...
```

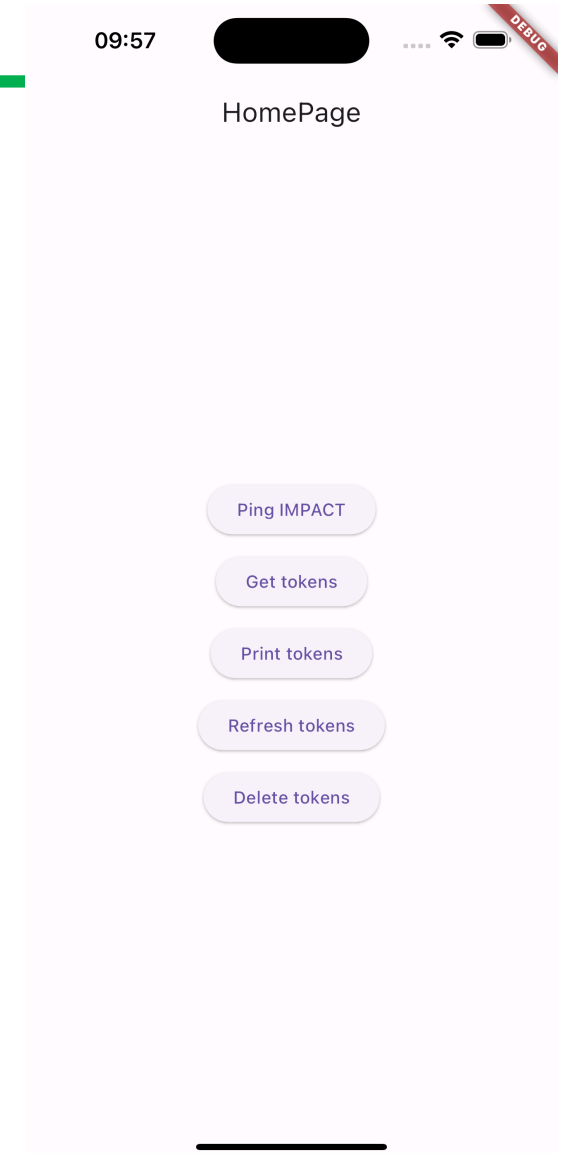
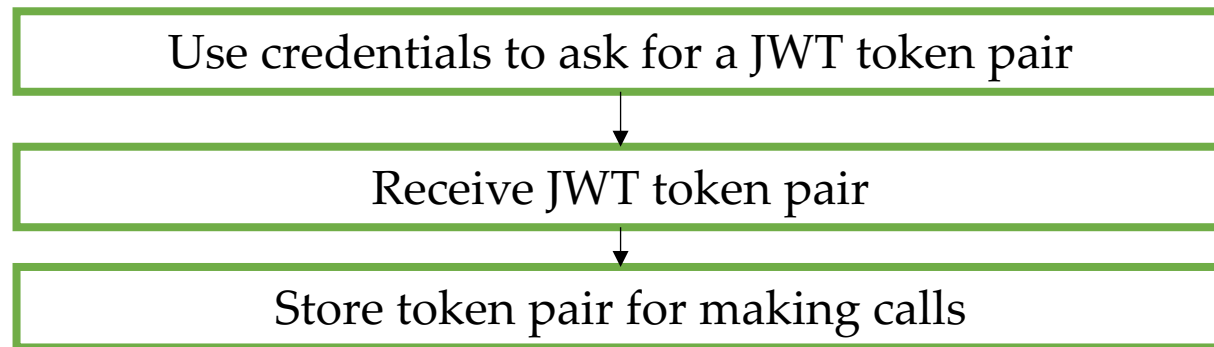
```
</manifest>
```

Only if you are using your
physical device

- No need to do this in iOS.

Case of study - Live

- Let's implement the operations required by the authorization flow



Outline

- Recap
- The flow
- RESTful API in practice
- Case study
- **Exercise**
- Homework
- Resources

Exercise

- Get familiar with the example
- Try to integrate the authorization flow in your project

Outline

- Recap
- The flow
- RESTful API in practice
- Case study
- Exercise
- **Homework**
- Resources

Homework

- Get familiar with the http package

Outline

- Recap
- The flow
- RESTful API in practice
- Case study

- Exercise
- Homework
- **Resources**

Resources

➤ IMPACT documentation

- <https://impact.dei.unipd.it/bwthw/docs/swagger/>

➤ HTTP Status Codes

- https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

➤ Fetch data from the internet. Cookbook by the Flutter community

- <https://docs.flutter.dev/cookbook/networking/fetch-data>

➤ Send data to the internet. Cookbook by the Flutter community

- <https://docs.flutter.dev/cookbook/networking/send-data>