UNIVERSITY OF PADOVA
DEPARTMENT OF INFORMATION ENGINEERING

Biomedical Wearable Technologies
for Healthcare and Wellbeing

Application Programming Interface (API)

A.Y. 2023-2024

Giacomo Cappon

DEPARTMENT OF
INFORMATION
ENGINEERING
UNIVERSITY OF PADOVA

Application programming interface (API)

- Two separate applications need an **intermediary** to talk to each other and to allow one application to access the information or functionality of the other.

- **Application programming interface (API):** a connection or interface between computer programs, which **defines how** the systems can interact.
  - The kind of requests that can be made to a system
  - How to make them
  - The data formats that should be used
  - The conventions to follow

- In contrast to a user interface, which connects a computer to a person, the **API connects pieces of software**

Application programming interface (API)

- Why do we need API? To allow **interoperability**, i.e., to allow multiple systems to cooperate.

- An API defines how two software systems can interact while hiding the internal details of how each system works. The API exposes only those parts that are needed for the systems to interact and keeps them consistent even if the internal details of the systems later change → **Modular programming**

- The API is often made up of different tools or services that other applications can call.

- **API specification:** a document or standard describing how to build or use the API tools or services.

- An API may be **custom-built** for a particular pair of systems, or it may be a **shared standard** allowing interoperability among many systems.

Release policy

- Based on the release policy, API can be private, partner or public.

Private API

- Used only within an organization
- For applications that are built for company employees
- Common use cases: the integration of company apps or the development of new systems using existing resources

Partner API

- Openly promoted but available only for business partners
- The target for these API is an application of a business partner of the provider
- Common use cases: the integration between the applications of two organizations

Public API

- Available to any third-party developers
- For applications designed for end customers
- They can be open or commercial

Some API types

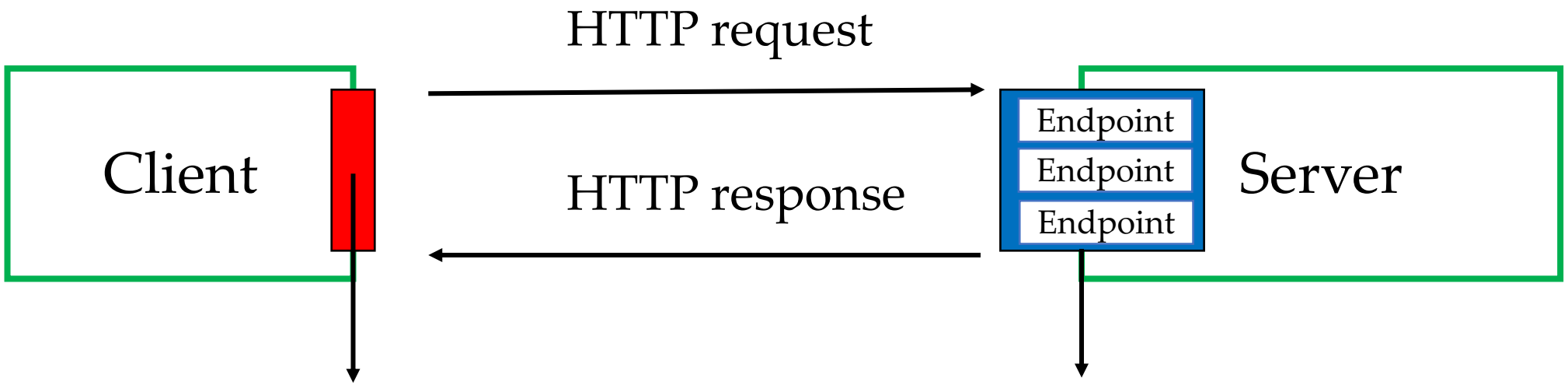
- **Library API**: the part of a software library that is accessible to programmers (the methods, subroutines or services that can be called by external programs and the specifications of how they can be called)
- **Language binding**: it allows a software written in a certain programming language to use a library written in another language.
  - Example: Python bindings are API that allows to call C libraries from Python
- **Operating systems API**: the interface between an application and the operating system.
  - Example: Windows API, which allows Windows programs to interact with the operating system
- **Database API**: They enable an application to communicate with a database management system (example: Drupal 7 Database API).
- **Web API**: API for web-based systems in a client-server architecture

Web API

- **Web API**: an interface for clients and servers that allows them to interact through the Web.
- **Server-side Web API**: the interface of a Web server towards the Web. It consists of one or more publicly exposed **endpoints**.
  - Endpoints: the URLs which specify where the resources that can be accessed by clients lie.
- **Client-side Web API**: a software interface that a client uses to send requests to a server-side Web API.
  - It allows a client web browser or web application to extend its functionality, by exploiting resources (data, programs, more in general services) provided by Web servers.

Request-response Web API

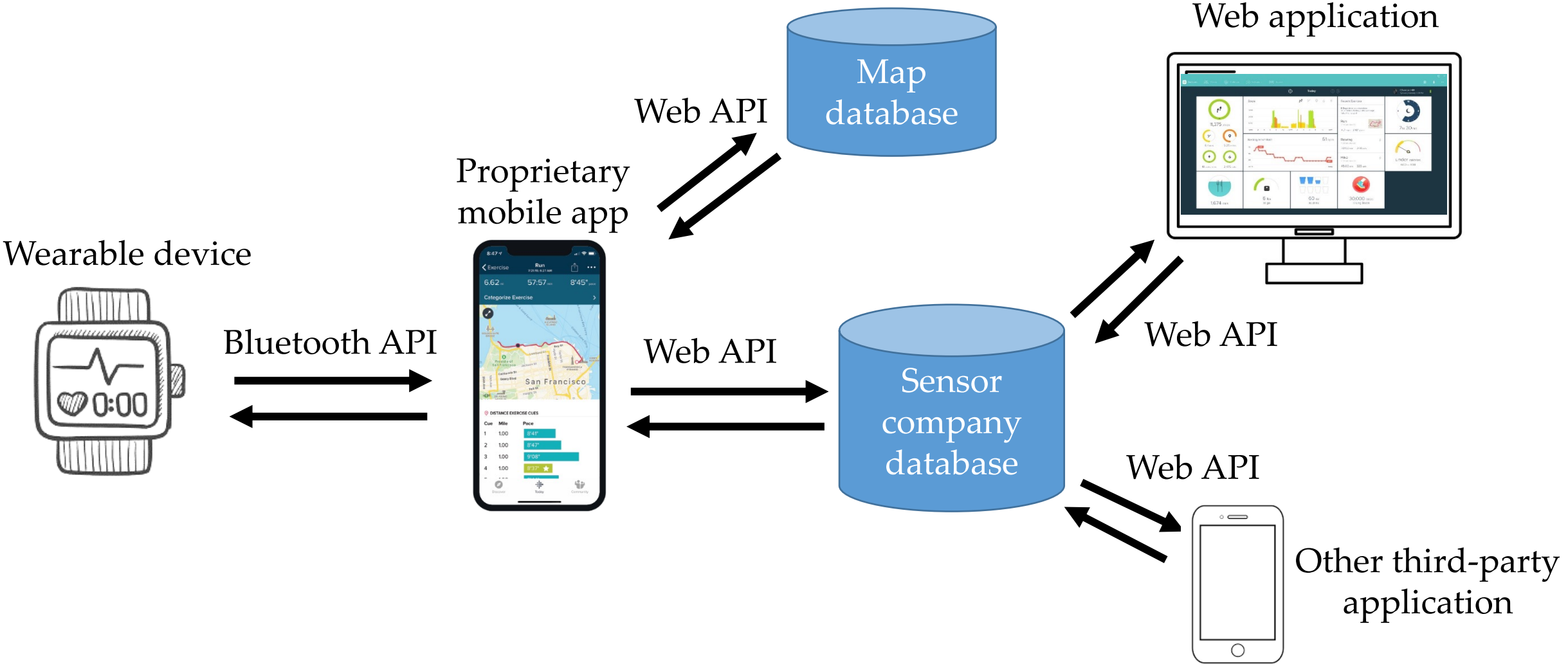
- Most of Web APIs are request-response APIs that rely on the HTTP protocol.



Client-side API:
The piece of software that sends requests to the server.

Server-side API:
The piece of software that receives the requests and the endpoints that provide the requested resources.

Examples of API in wearable sensors



Approaches to design request-response Web API

- Web APIs are realized using protocols and/or specifications to define the semantics and syntax of the messages exchanged by client and server.

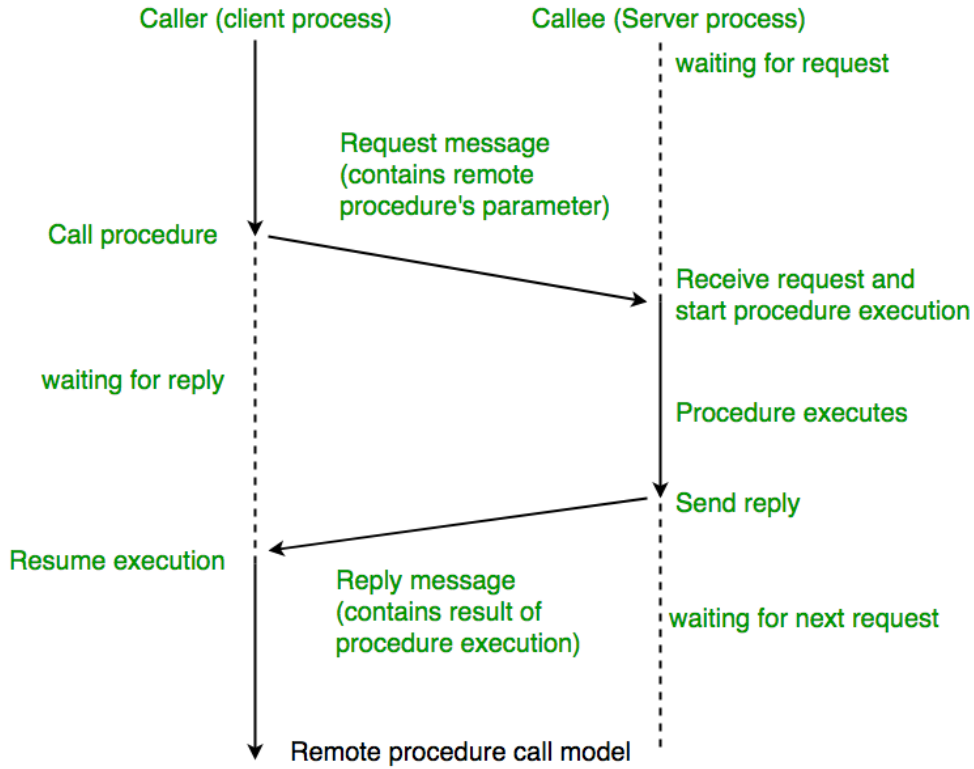
  → **API architectures**

- Most popular request-response API architectures:
  - **RPC**: Remote Procedure Call
  - **SOAP**: Simple Object Access Protocol
  - **REST** architectural style
    - In this case the API is called **RESTful API**.
  - **GraphQL**

Remote Procedure Call (RPC)

- The simpler and oldest web API approach, proposed by Bruce Jay Nelson in 1981.
- **Remote Procedure Call (RPC)** is a request-response protocol by which the client sends a request message to a known remote server to execute a specified procedure with supplied parameters.

- RPC extends the conventional local procedure calling so that the called procedure does not need to be in the same address space as the calling procedure. The two processes may be on the same system, or on different systems with a network connecting them.
- In RPC **the endpoints represent actions**



Remote Procedure Call (RPC)

- Different RPC protocols:
  - 1997: XML-RPC uses XML to encode its calls and HTTP to send requests and responses
  - 2005: JSON-RPC uses JSON to encode data and HTTP to send requests and responses
  - 2016: Google RPC (gRPC) uses Protocol Buffers to encode data and HTTP/2 to send requests/responses
- Pros:
  - **Straightforward and simple interaction:** all is done with GET and POST HTTP methods
  - **Easy to add functions**: a new function can be defined and associated to a new endpoint
  - **High performance**: having the functions in the URL allows to have lightweight payloads
- Cons:
  - **Tight coupling to the underlying system:** RPC messages are not standardized. Their format depends on the specific application → low reusability of the API
- Use cases: to perform extremely high-performance, low-overhead internal messaging within big companies (Google, Facebook, Twitch, etc.).

Example of RPC

- This example illustrates an online shop system that tries to handle a transaction with RPC.
- The function paymentService.prcessPayment is implemented in another system.
- User must know a lot of things to use the function: i.e., how to create a Card, function parameters, the structure of a Result, etc…

```
Card card = Card();
card.setCardNumber("1234 5678 1234 5678");
card.setExpiryDate("11/30");
card.setCVC("123");

Result result = await
paymentService.processPayment(card, 3.99,
Currency.EUR);

if(results.isSuccess()){
        fulfilOrder();
}
```

Simple Object Access Protocol (SOAP)

- Released by Microsoft in 1998
- Web API protocol with highly standardized messages written in XML
- It runs over HTTP or other application layer protocol

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

It defines the XML document as a SOAP message

Application-specific information specifying how the recipient should process the message (optional)

The message body

Error messages (optional)

Example of SOAP messages

Example of SOAP request:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-
encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

Example of SOAP response:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml;
charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-
envelope/"
soap:encodingStyle="http://www.w3.org/2003/0
5/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/s
tock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

SOAP: pros and cons

- Pros:
  - Web Services Security (**WS-Security**): an extention of SOAP that allows to use security mechanisms (e.g., signature of messages, encryption of messages, authentication)
  - SOAP allows to perform **stateful communications**, necessary for complex transactions in which multiple parties are involved (e.g., online banking and e-commerce applications)

- Cons:
  - Only XML
  - Heavy messages
  - Specialized knowledge of the protocol rules needed

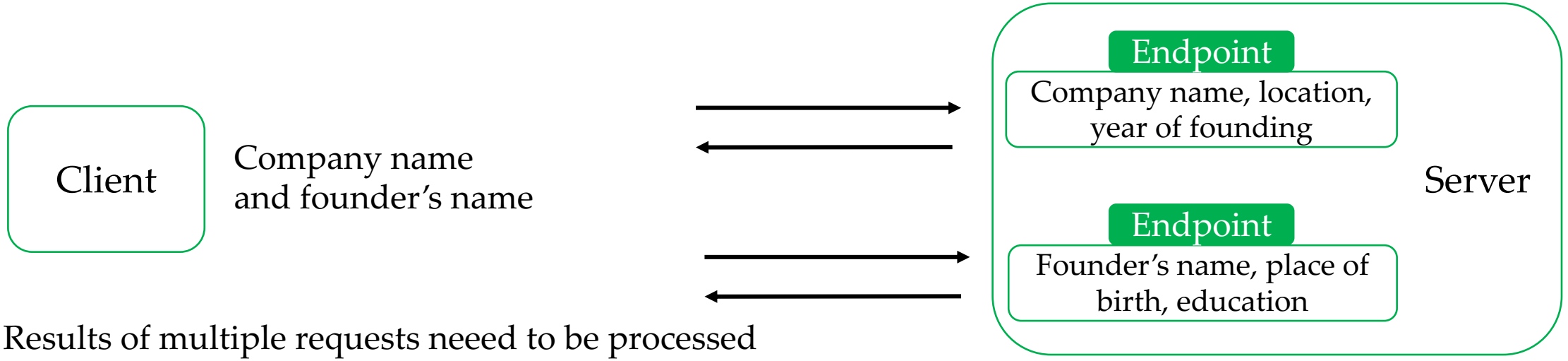
RESTful APIs

- With the expansion of modern Web, a more simple, flexible and scalable way of defining API was needed.
- **RESTful APIs**: APIs designed according to the REST principles, proposed in 2000 by Roy Fielding.
- REST is resource-based (instead of action-based) and it only uses the HTTP methods (GET, POST, PUT, DELETE) → no specialized knowledge required
- No strict protocol, only compliance with the 6 REST constraints
- No constraints on the message format → **JSON** preferred to XML

RESTful APIs: pros and cons

- Pros:
  - Easy, no specialized knowledge required
  - Multiple formats supported
  - Great flexibility and scalability
  - Cache-friendly

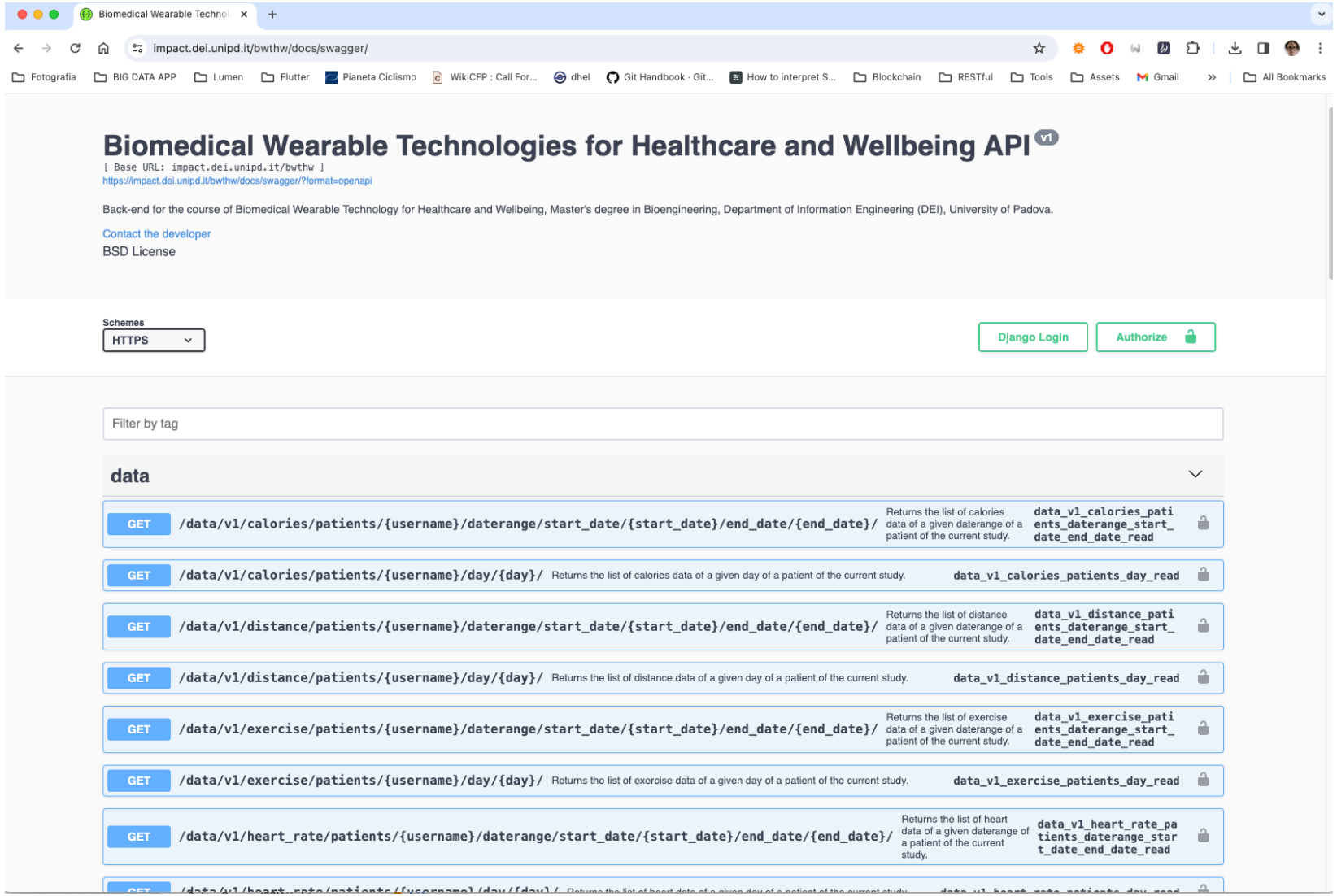
- Cons:
  - REST messages have large payloads to transfer metadata necessary to understand the state of the application and to enable complex actions.
  - Chatty interactions: it generally requires multiple requests to get the needed data. Often unnecessary data are transferred.



Results of multiple requests neeed to be processed (combined and filtered) on client side.

The IMPACT BWTHW RESTful API

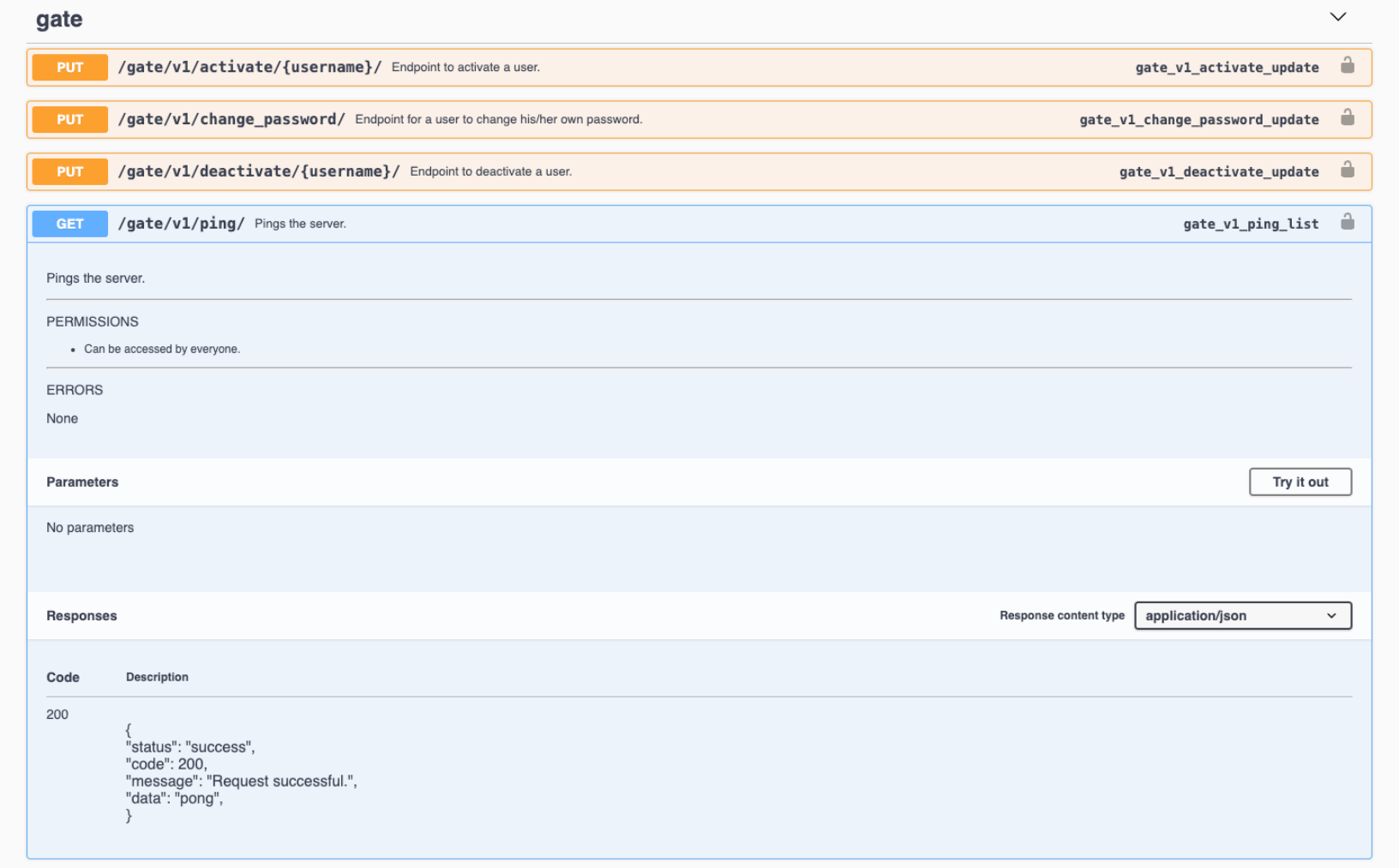
- This is the API you will use to build your project
- It allows to authenticate, ping (to know if the server is alive), and to get data
- Full URL: https://impact.dei.unipd.it/bwthw/



The ping endpoint

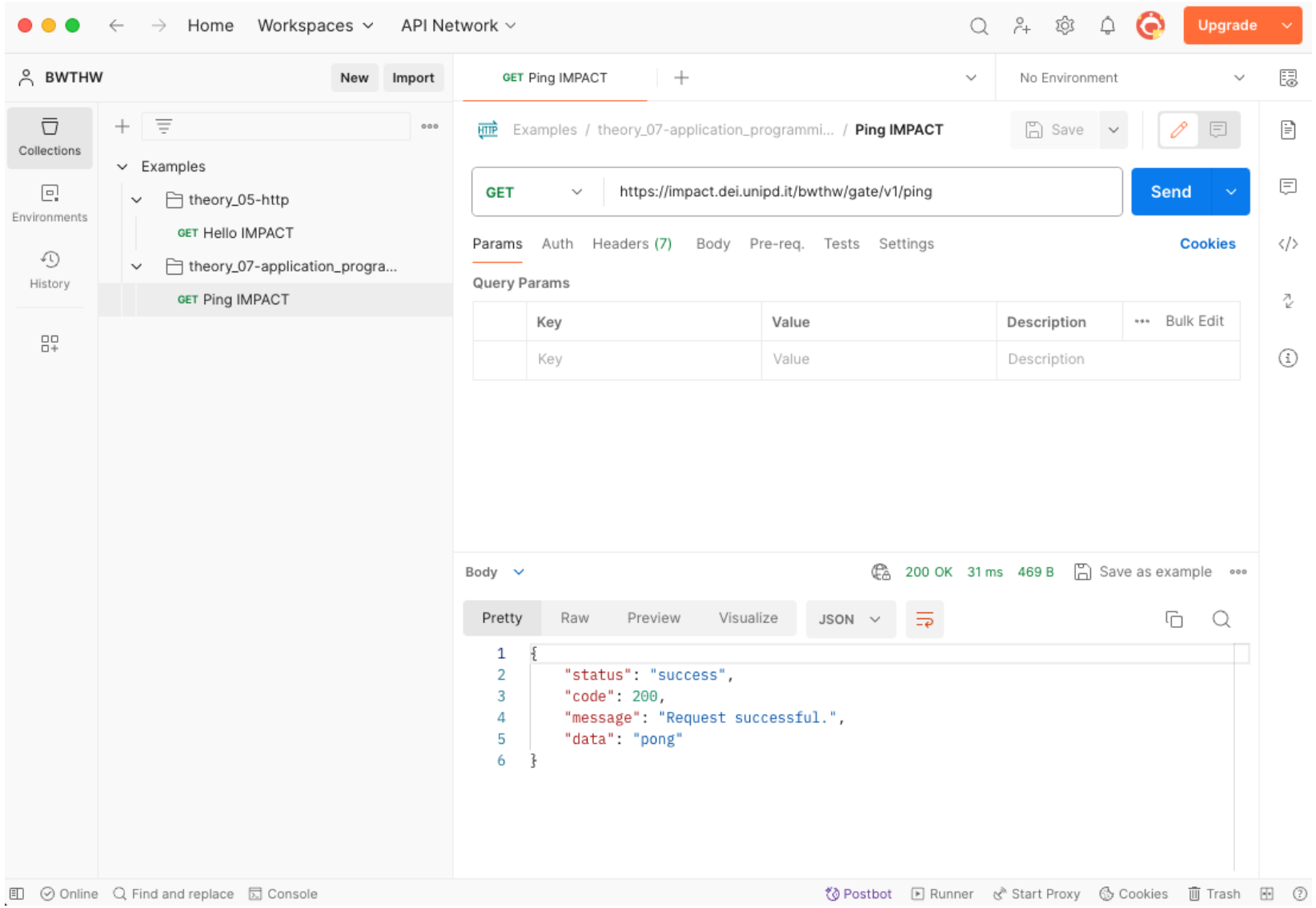
- Let's inspect the **ping** endpoint

- Key things we can learn from the docs:
  - Method: GET
  - Full URL: https://impact.dei.unipd.it/gate/v1/ping/
  - It does not require parameters
  - It does not require to be authenticated
  - If successful, the response will be a JSON



The ping endpoint

- POSTMAN can be used to make a request and test it



GraphQL

- Released by Facebook in 2015
- **GraphQL** is a syntax to make precise data requests (**queries**) to servers
- Servers provide the clients with a **schema** that specifies which resources are available, how they are structured and the types of queries that can be done.
- Based on the schema, the client can formulate specific queries and send them to the server which will return exactly the data requested by the client's query (no more, no less).

REST	**GraphQL**
▪ Multiple endpoints	▪ Single endpoint
▪ Servers decide how data are returned	▪ Clients decide how data are returned
▪ No schema	▪ Schema required

GraphQL: Example

NEW SLIDE

For example, a GraphQL service that tells you who the logged in user is (me) as well as that user's name might look like this:

```
type Query {
   me: User
}
```

```
type User {
  id: ID
  name: String
}
```

Along with functions for each field on each type:

```
function Query_me(request) {
  return request.auth.user
}
```

```
function User_name(user) {
  return user.getName()
}
```

Then, GraphQL queries like:

```
{
  me {
    name
  }
}
```

Could produce the following JSON result:

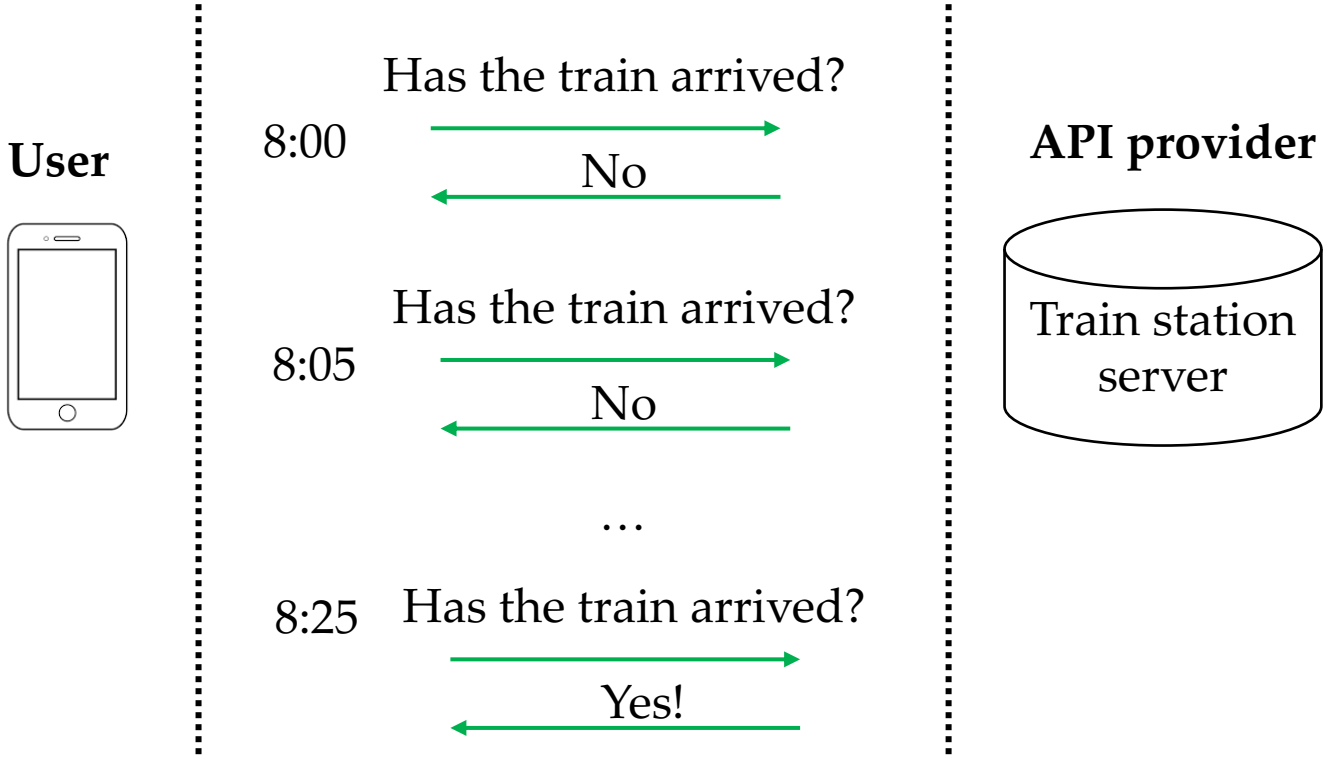
```
{
  "me": {
    "name": "Luke Skywalker"
  }
}
```

Summary of types of request-response Web APIs

	RPC	SOAP	RESTful	GraphQL
Organized in terms of	Local procedure calling	Enveloped message structure	Compliance with 6 architectural constraints	Schema of server's resources
Format	Multiple (JSON, XML, Protobuf, …)	XML only	Multiple (JSON, XML, HTML, plain text, …)	JSON
Statefull or stateless	Both	Both	Stateless	Stateless
Learning curve	Easy	Difficult	Easy	Medium
Community	Large	Small	Large	Growing
Use cases	Command and action-oriented APIs, internal high performance and massive micro-services	Payment gateways, identity management, financial applications	Public APIs, simple resource-driven app	Mobile APIs, complex systems

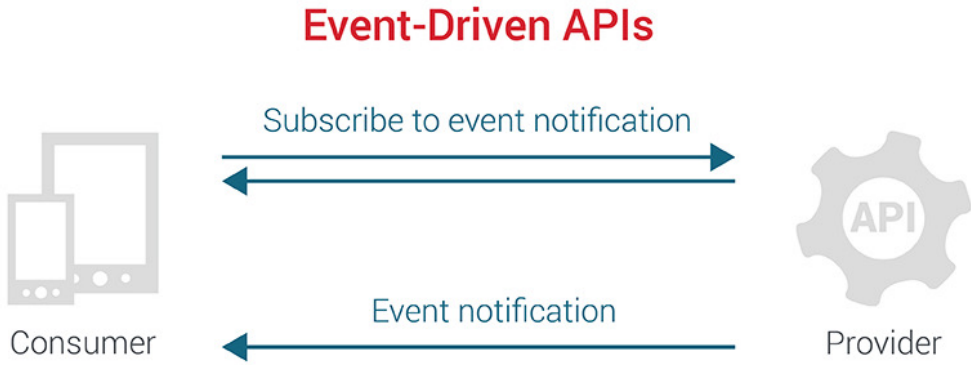
Request-response API architecture

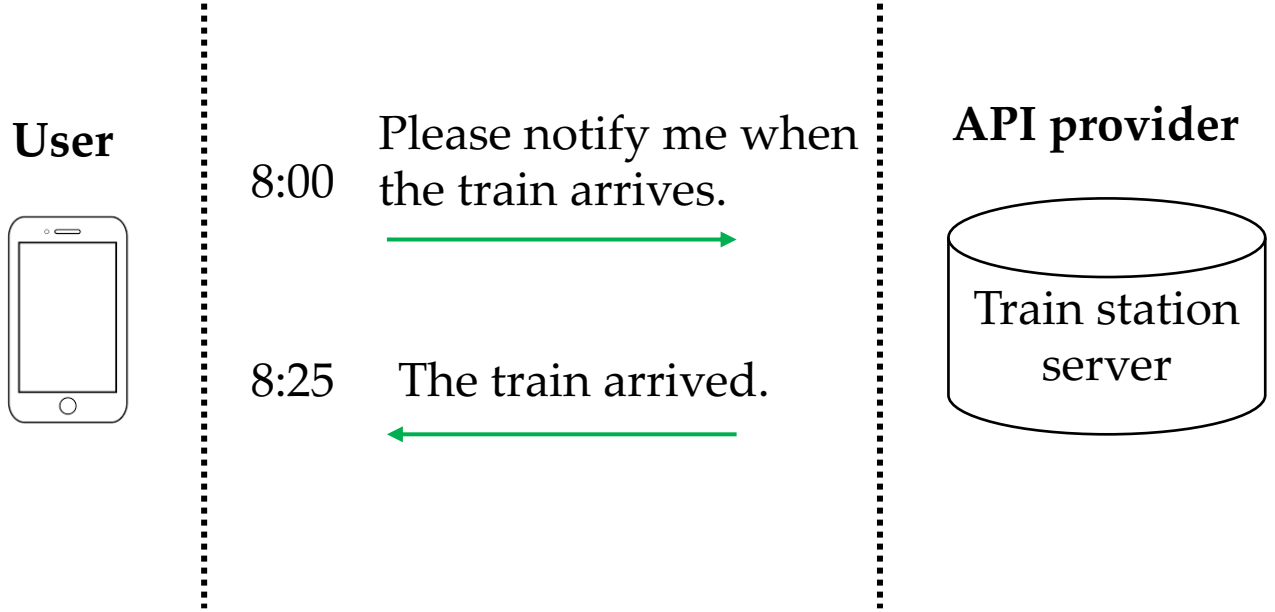
- RPC, SOAP, RESTful, GraphQL are all types of **request-response Web API**
- To get the latest information, the client always have to **poll** the API provider (server).



Event-driven API architecture

- In **event-driven APIs** the API provider notifies the client when something interesting happens. Notifications do not need a request from the client.
- An event-driven API must offer two capabilities to its clients
  - A mechanism to allow clients to subscribe to events of their interest.
  - Delivery of events to subscribed clients in an asynchronous manner.





WebSocket

- An example of protocol for event-driven APIs
- **WebSocket** allows for constant, bi-directional communication between the server and client, which means both parties can communicate and exchange data as and when needed.
- Suitable for realizing Web chats.