Web Services

HTTP, Request Headers, RESTful Web Services, Postman, Swagger

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#QA-BackEnd

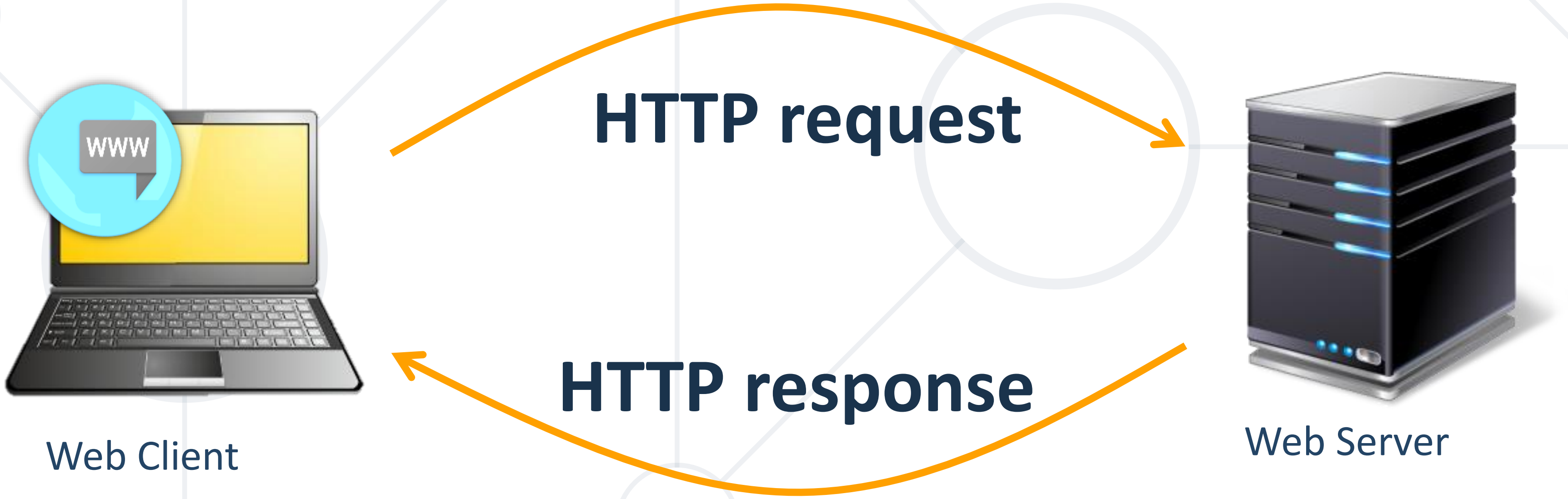
Table of Contents

HTTP Overview

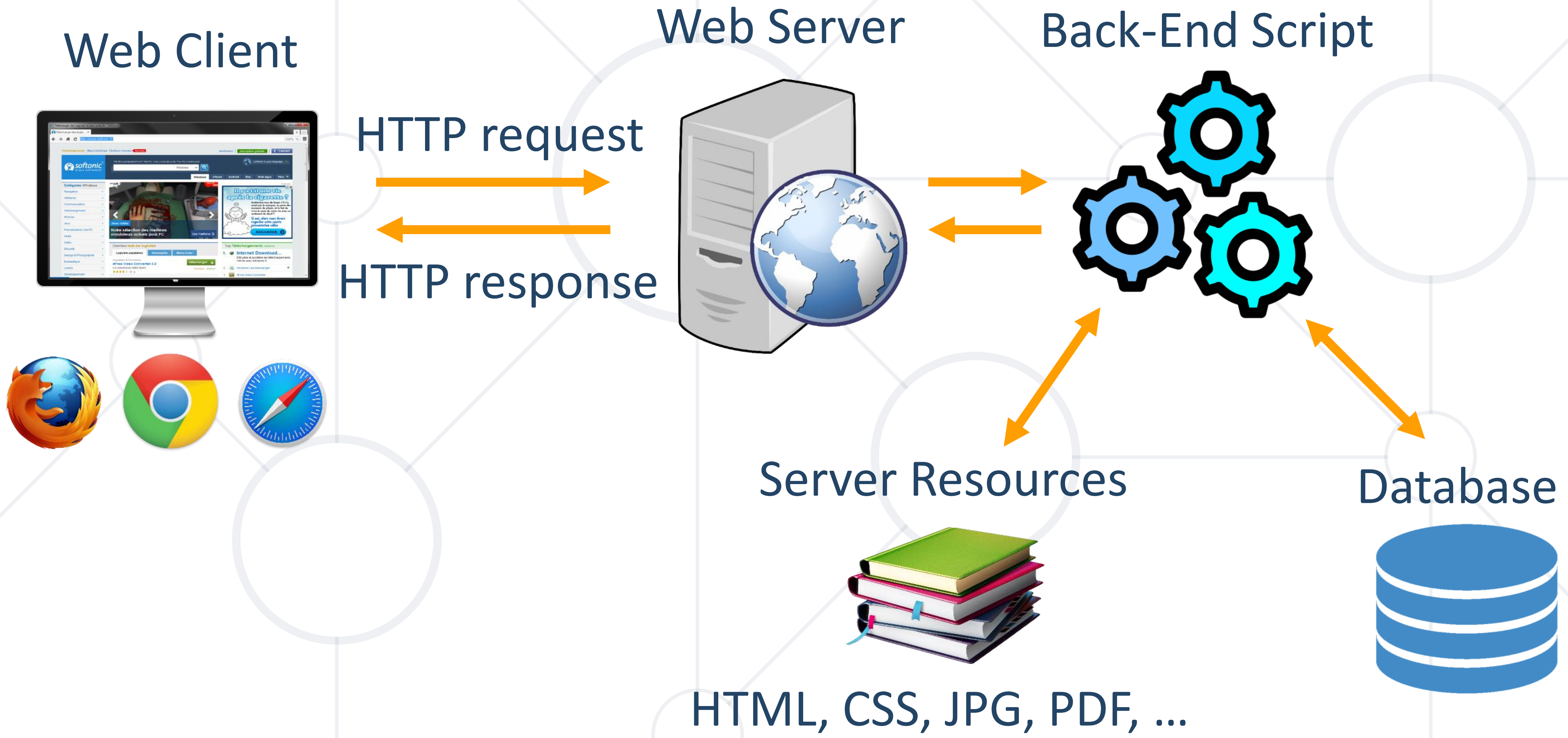
Hypertext Transfer Protocol

HTTP Basics

- HTTP (**H**yper **T**ext **T**ransfer **P**rotocol)
  - Text-based client-server protocol for the Internet
  - For transferring Web resources (HTML files, images, styles, etc.)
  - Request-response based



Web Server Work Model



HTTP Request Methods

- **HTTP request methods** specify the desired **action** to be performed on the requested resource (identified by URL)

Method	Description
GET	Retrieve a resource
POST	Create / store a resource
PUT	Update (replace) a resource
DELETE	Delete (remove) a resource
PATCH	Update resource partially (modify)
HEAD	Retrieve the resource's headers

CRUD == the four main functions of persistent storage

Other Methods
CONNECT
OPTIONS
TRACE

HTTP Response Status Codes

Status Code	Action	Description	
200	OK	Successfully retrieved resource	Success
201	Created	A new resource was created	
204	No Content	Request has nothing to return	
301 / 302	Moved	Moved to another location (redirect)	Redirect
400	Bad Request	Invalid request / syntax error	Error
401 / 403	Unauthorized	Authentication failed / access denied	
404	Not Found	Invalid resource requested	
409	Conflict	Conflict detected, e.g. duplicated email	
500 / 503	Server Error	Internal server error / service unavailable	

Content-Type and Disposition

- The **Content-Type** / **Content-Disposition** headers specify how the HTTP request / response body should be processed

JSON-encoded data

```
Content-Type: application/json
```

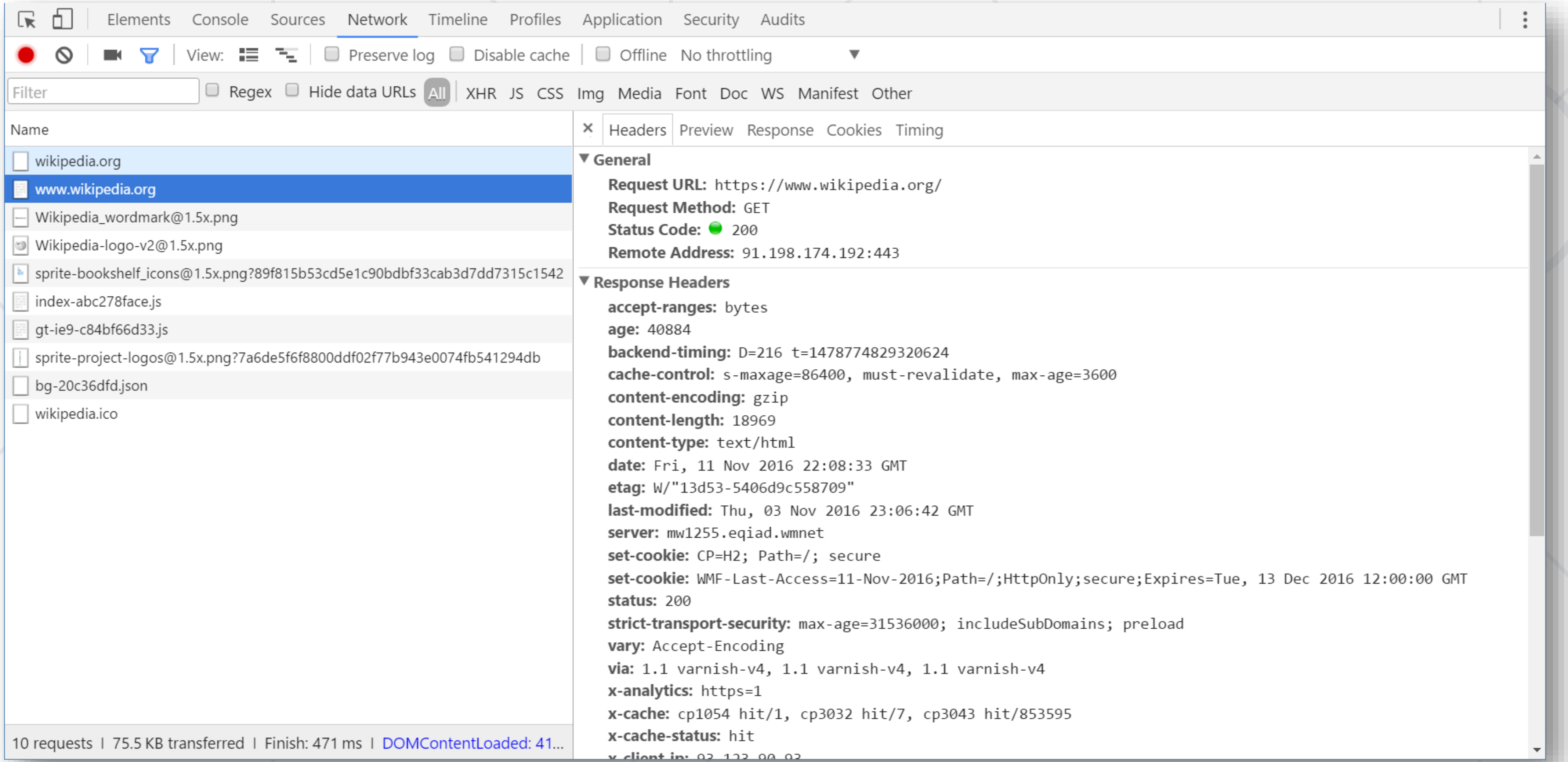
UTF-8 encoded HTML page. Will be shown in the browser

```
Content-Type: text/html; charset=utf-8
```

```
Content-Type: application/pdf
Content-Disposition: attachment;
filename="Financial-Report-April-2016.pdf"
```

This will download a PDF file named Financial-Report-April-2016.pdf

Browser Developer Tools



Elements Console Sources Network Timeline Profiles Application Security Audits
View: Preserve log Disable cache Offline No throttling
Filter Regex Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other
Headers Preview Response Cookies Timing

```
Name
wikipedia.org
www.wikipedia.org
Wikipedia_wordmark@1.5x.png
Wikipedia-logo-v2@1.5x.png
sprite-bookshelf_icons@1.5x.png?89f815b53cd5e1c90bdbf33cab3d7dd7315c1542
index-abc278face.js
gt-ie9-c84bf66d33.js
sprite-project-logos@1.5x.png?7a6de5f6f8800ddf02f77b943e0074fb541294db
bg-20c36dfd.json
wikipedia.ico

10 requests | 75.5 KB transferred | Finish: 471 ms | DOMContentLoaded: 41...
```

```
▼ General
  Request URL: https://www.wikipedia.org/
  Request Method: GET
  Status Code: ● 200
  Remote Address: 91.198.174.192:443
▼ Response Headers
  accept-ranges: bytes
  age: 40884
  backend-timing: D=216 t=1478774829320624
  cache-control: s-maxage=86400, must-revalidate, max-age=3600
  content-encoding: gzip
  content-length: 18969
  content-type: text/html
  date: Fri, 11 Nov 2016 22:08:33 GMT
  etag: W/"13d53-5406d9c558709"
  last-modified: Thu, 03 Nov 2016 23:06:42 GMT
  server: mw1255.eqiad.wmnet
  set-cookie: CP=H2; Path=/; secure
  set-cookie: WMF-Last-Access=11-Nov-2016;Path=/;HttpOnly;secure;Expires=Tue, 13 Dec 2016 12:00:00 GMT
  status: 200
  strict-transport-security: max-age=31536000; includeSubDomains; preload
  vary: Accept-Encoding
  via: 1.1 varnish-v4, 1.1 varnish-v4, 1.1 varnish-v4
  x-analytics: https=1
  x-cache: cp1054 hit/1, cp3032 hit/7, cp3043 hit/853595
  x-cache-status: hit
  x-client-ip: 93.123.90.93
```

Introduction to RESTful Services

Mapping CRUD Operations

What is an API?

Software University

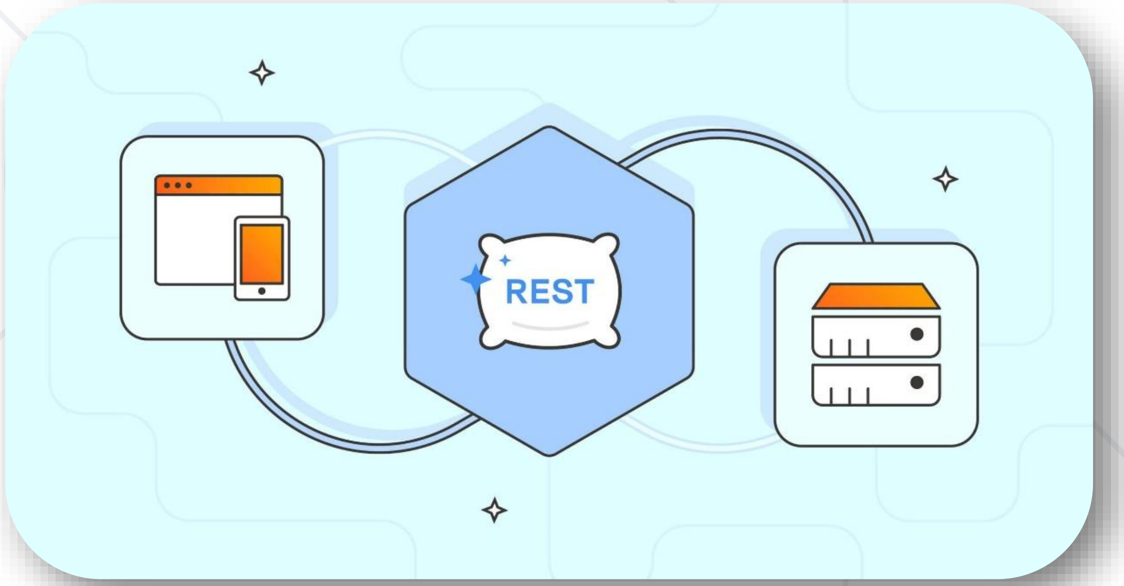
- Application Programming Interface
  - **APIs** – Mechanisms that enable **two software components** to **communicate with each other** using a set of definitions and **protocols**
- APIs in everyday life
  - Social media bots
  - Third-party login
  - E-commerce transactions
  - Weather apps

What is RESTful API?

- An **application programming interface** that follows the principles of **REpresentational State Transfer**
- **REST** - a **set of guidelines** for designing web services that are **scalable**, **uniform**, and **stateless**
- Allows clients to **access and manipulate resources** on a server using standard **HTTP methods** – **GET**, **POST**, **PUT**, and **DELETE**
- **Resource** - anything that **has a unique identifier**, such as a user, a product, or a post

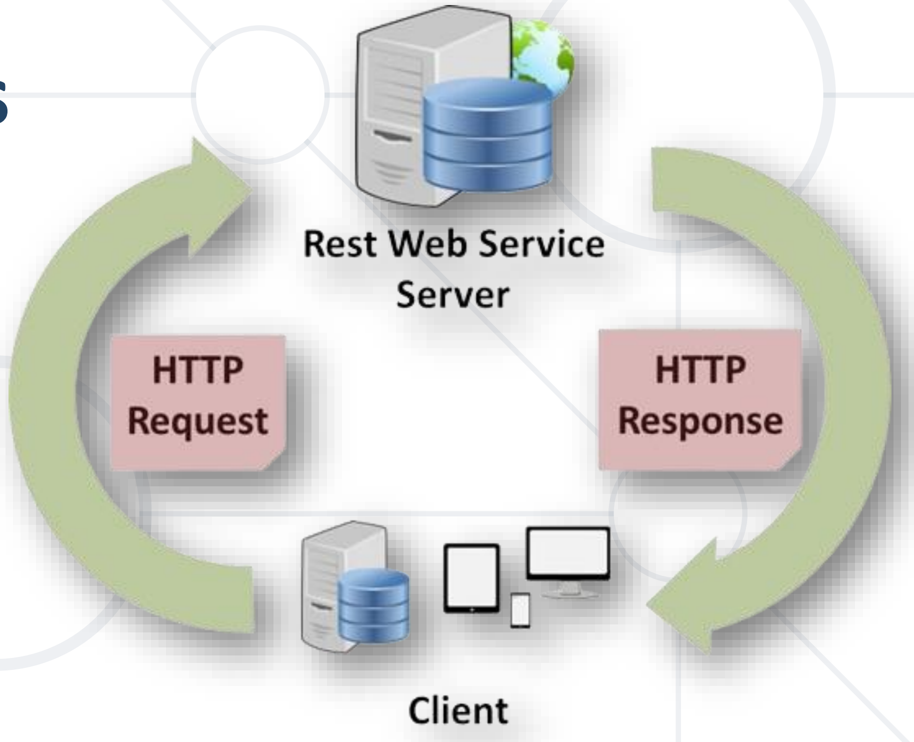
REST - Communication Standard

- REST - the most common communication standard between computers over Internet
- The **common API standard** used by most mobile and web **applications to talk to the servers is called REST**



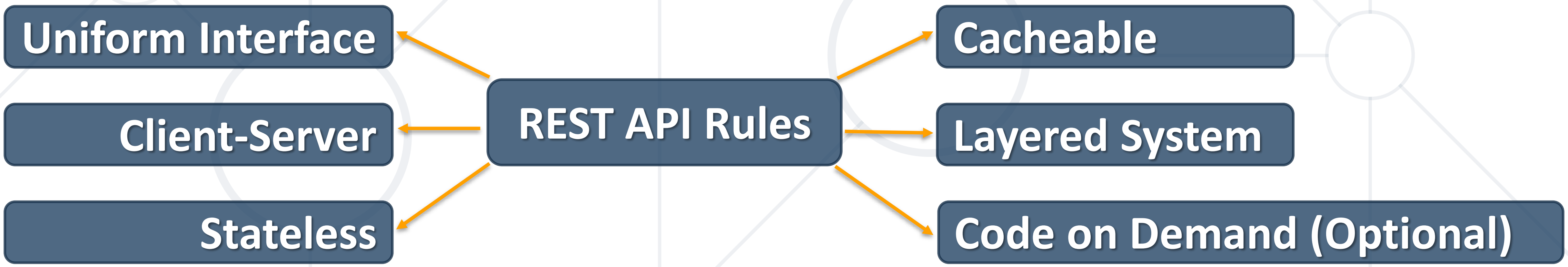
REST Architecture

- In **REST architecture**, a **REST Server** simply **provides access to resources** and **REST client accesses and modifies** the resources
- Each resource is identified by **URIs / global IDs**
- REST uses various **formats to represent a resource** like text, JSON, XML…



Set of Rules

- **REST** is not a specification
- It is a set of rules
  - Common standard for building web API since the early 2000s, introduced by [Dr. Roy Fielding]



Stateless

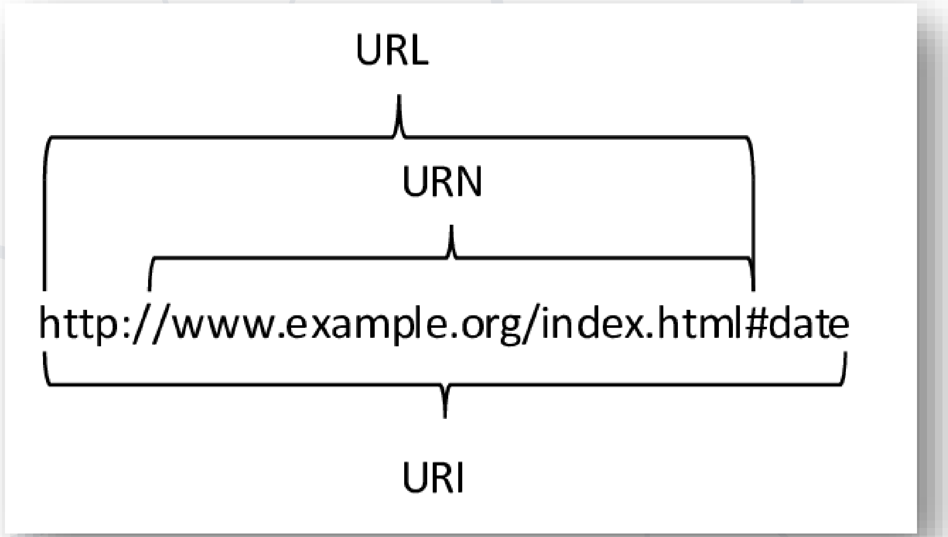
- In REST, the **client** and the **server** interact in a **stateless manner**
- Neither the client nor the server should assume the existence of the other's **state between requests**
- Each request from the client to the server must contain **all of the information the server needs** to understand the request and cannot take advantage of any stored context on the server
- **Session state** is therefore **kept entirely on the client**

Cacheable & Layered System

- **RESTful services** are designed to be **cacheable**
  - **Responses** must, implicitly or explicitly, define themselves as **cacheable, or not**
  - To prevent clients from **reusing old** or **inappropriate data** in response to further requests
- **Layered System**
  - A client cannot tell whether is connected **directly to the end server** or to an **intermediary** along the way
  - **Intermediate servers** improve **system scalability** by enabling **load-balancing** and by providing **shared caches**

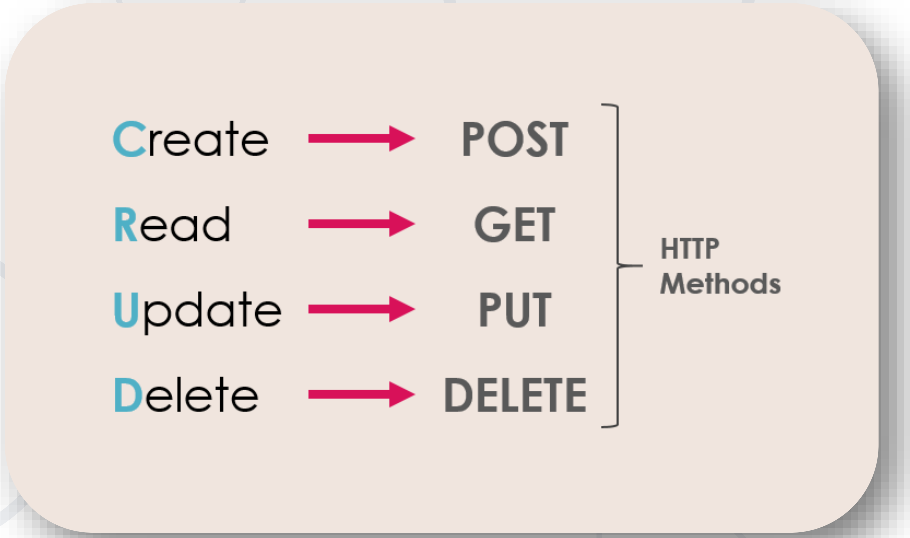
Uniform Interface

- **Uniform Interface** - The defined way a client interacts with the server **independent of the device or application**
- **Resource-Based** - The API needs to have a specific **URI** (uniform resource identifier) for each resource
- URIs are **identifiers of resources** that work across the Web



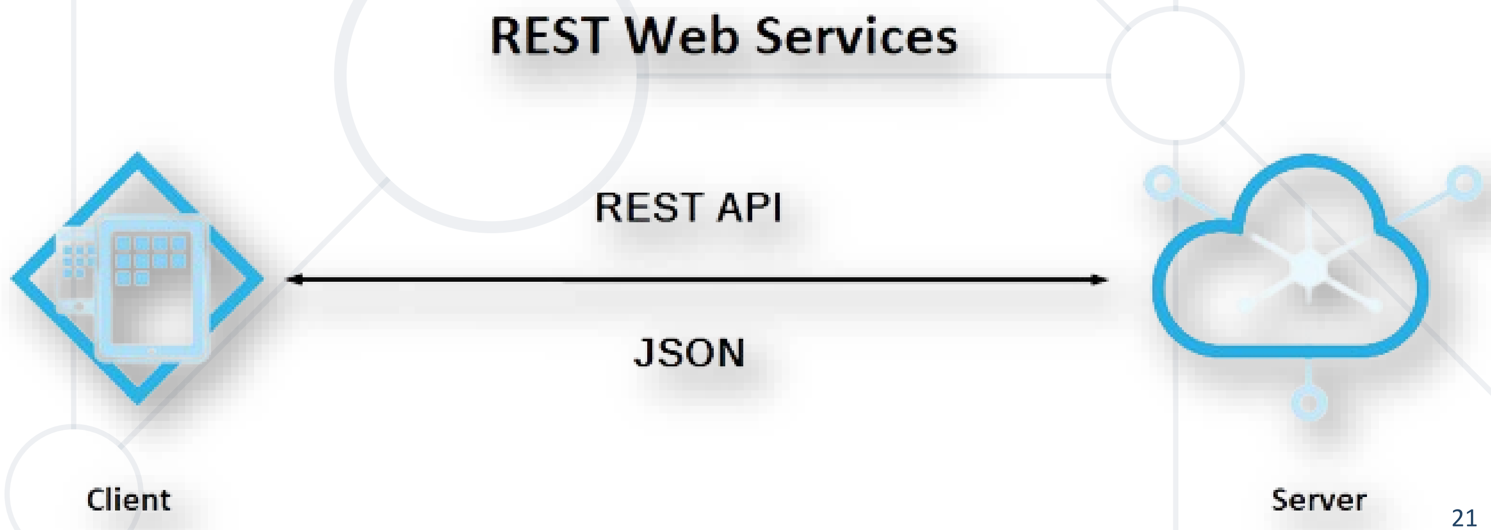
HTTP Methods –> CRUD Operations

- The following **HTTP methods** are commonly used in **REST-based architecture**:
  - **POST** – Used to **create** a new resource
  - **GET** – Provides a **read-only access** to a resource
  - **PUT** – Used to **update** an existing resource or create a new resource
  - **DELETE** – Used to **remove** a resource



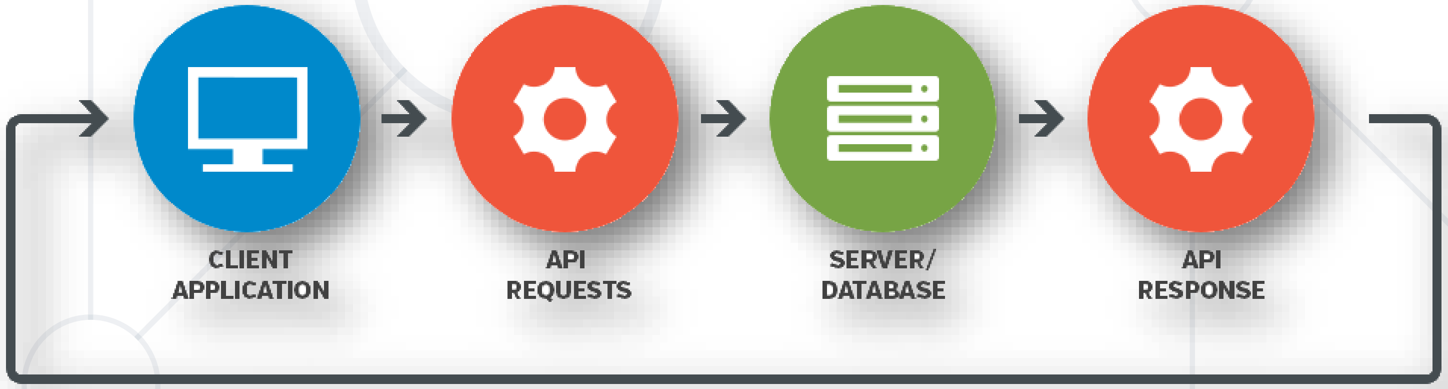
REST Web Services

- **REST Web Service**
  - A **lightweight**, **maintainable**, and **scalable** service
  - Built on the **REST architecture**
- **Expose API** from an application **to the calling client** in a
  - Secure
  - Uniform
  - Stateless manner



Web Services and APIs

- **Web services** expose **back-end APIs** over the **network**
  - May use different **protocols** and **data formats**: HTTP, REST, GraphQL, gRPC, SOAP, JSON-RPC, JSON, BSON, XML, YML, …
- **Web services** are hosted on a Web server (HTTP server)
  - Provide a set of functions, invokable from the Web (Web API)
- **RESTful APIs** is the most popular Web service standard



Authentication and Authorization

RESTful APIs

Authentication

- The process of **verifying the identity of a user or a system**
- This process **often** involves checking whether a **username and password provided are correct**
- **In RESTful services**, authentication can be achieved through various methods, including
  - Basic authentication (using a base64 encoded **username and password**)
  - **Tokens** such as JSON Web Tokens (JWTs)
  - **HMAC** (hash-based message **authentication codes**)

Authorization

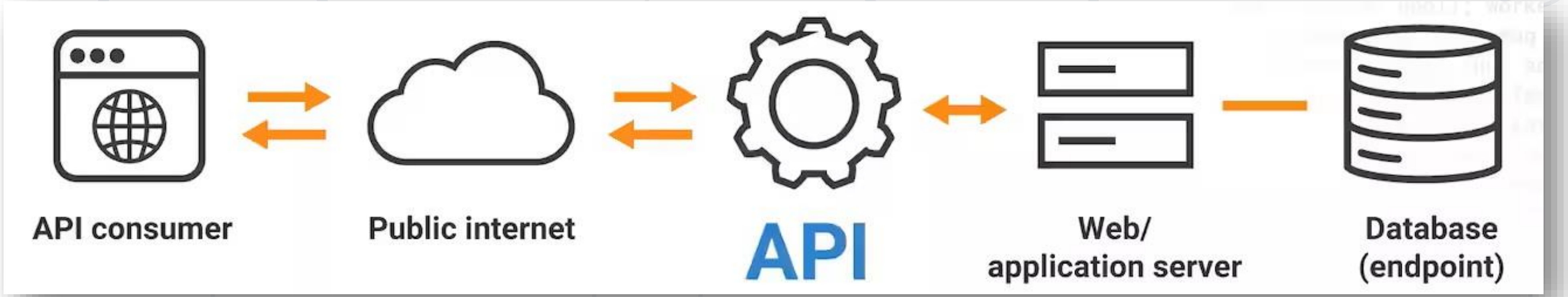
- **Authorization** - granting an **authenticated user permission** to access different resources or perform specific actions
- **Once a user is authenticated**, the system must check **what resources the user is allowed to access** or **what actions** they are **permitted to perform**
- **It's important to note that both authentication and authorization mechanisms must be protected with HTTPS to prevent man-in-the-middle attacks and to ensure that sensitive information is transmitted securely**

Authentication vs Authorization

- **Authentication**
  - Determines **whether users are** who they claim to be
  - Generally, transmits info through an **ID Token**
  - Usually done before authorization
- **Authorization**
  - Determines **what users can and cannot access**
  - Generally, transmits info through an **Access Token**
  - Usually done after successful authentication

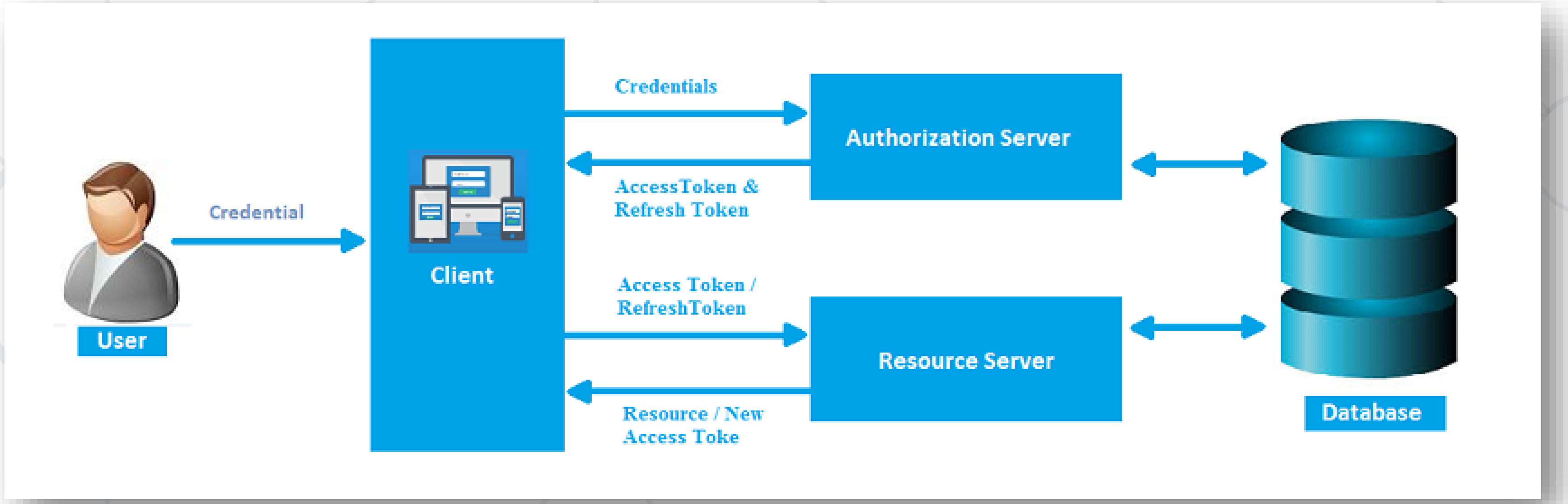
REST API Authentication

- **REST APIs** have become approach for modern **web** and **mobile application platforms**
- They **separate data** and **presentation layers**, allowing systems to scale in size and feature sophistication over time
- As **data moves across boundaries**, **security** becomes a **key concern** for REST APIs containing **sensitive information**



REST API Authentication

- One of the most straightforward ways to **secure these APIs** is to implement **authentication mechanisms** that control their exposure
- Mainly through **user credentials** and **encrypted access codes**


User
Credential
Client
Credentials
AccessToken & Refresh Token
Access Token / RefreshToken
Resource / New Access Toke
Authorization Server
Resource Server
Database

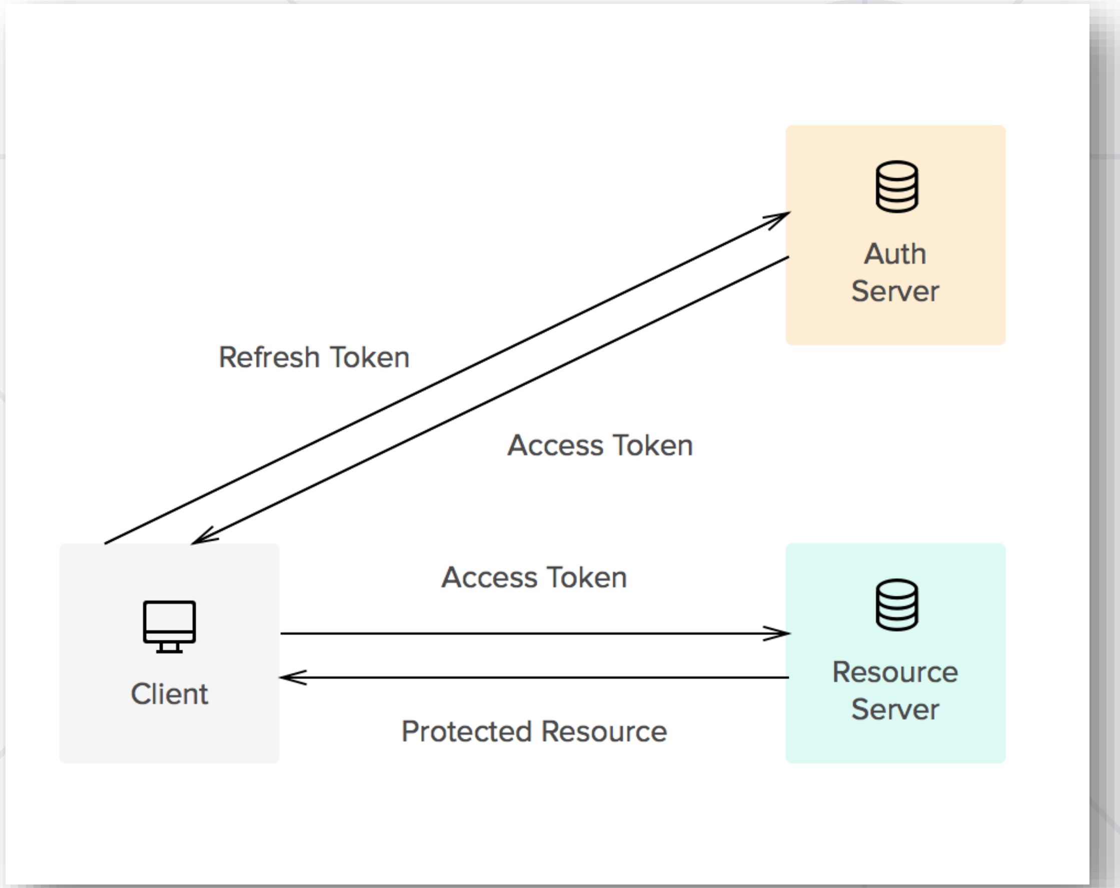
Token Authentication

Software University

- Token authentication is also known as **Bearer Authentication**
- To use it, you just specify Authorization:
  - **Bearer <token>**
- Token is a string that **represents** the **user's identity** and **permissions**
  - If you **have (bear) the token**, you can get the appropriate **access to the API**

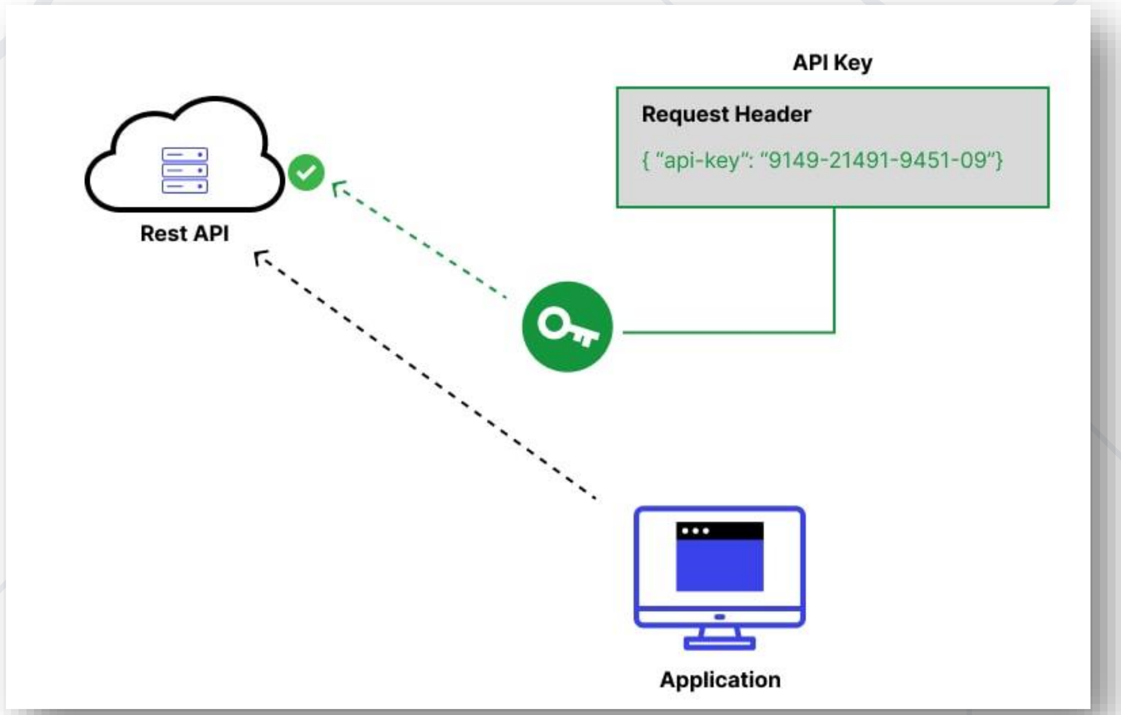
Bearer Authentication

- **Bearer Authentication** is a method of **sending a token with HTTP requests** to authenticate
- The token is a string, often in **JWT (JSON Web Token)** format, that the server can use to verify the request's authenticity and integrity



REST API Authorization

- After you prove the **user's identity**, you can check which data that user is **allowed to access**
- Authorization ensures that the **user is authorized** to **view** or **edit** a specific **set of data**



Postman

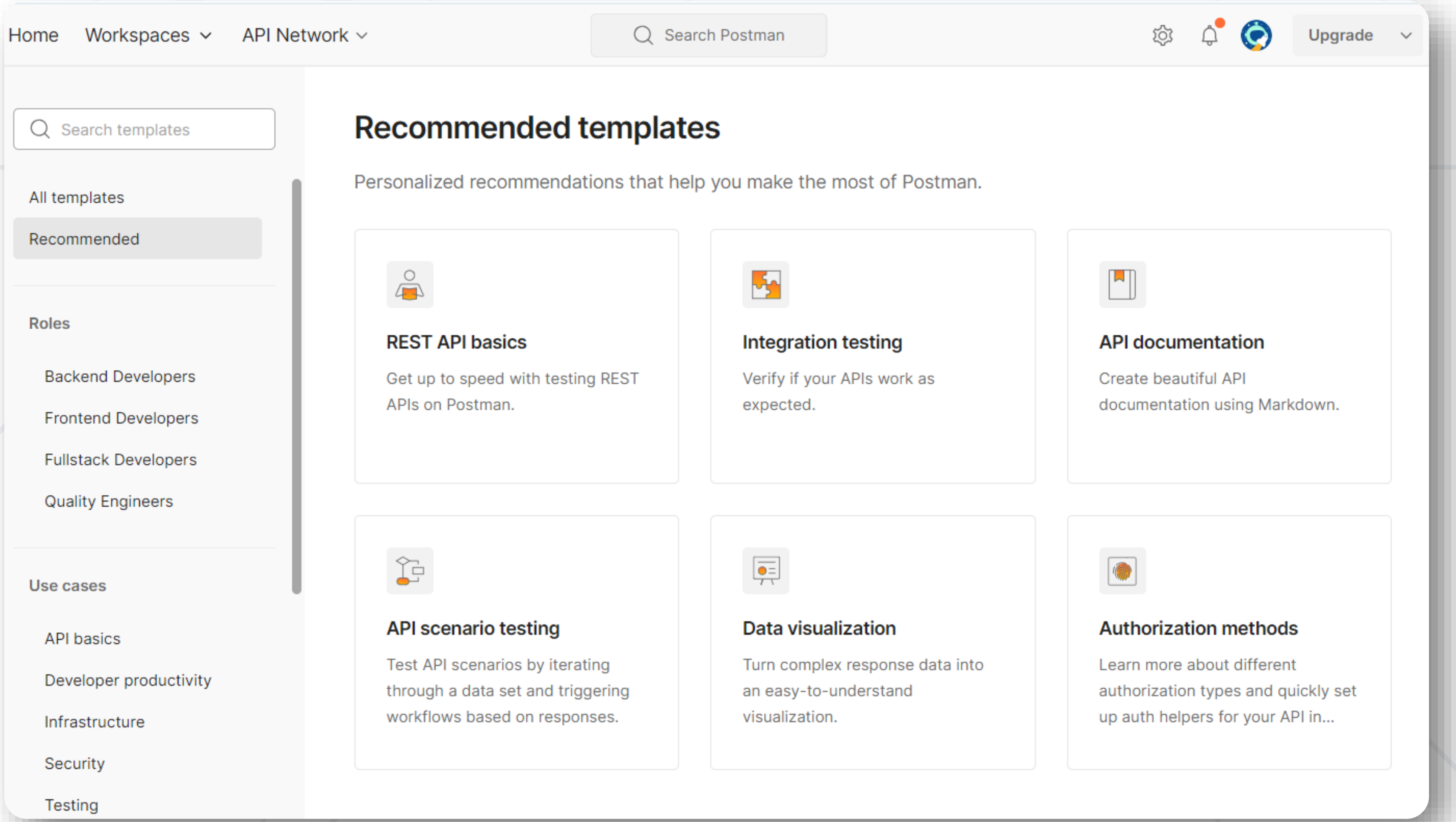
Testing Tool for RESTful APIs

Postman

Software University

Postman

- HTTP **client tool** for developers and QAs
- Compose and send **HTTP requests**



Recommended templates

Personalized recommendations that help you make the most of Postman.

REST API basics
Get up to speed with testing REST APIs on Postman.

Integration testing
Verify if your APIs work as expected.

API documentation
Create beautiful API documentation using Markdown.

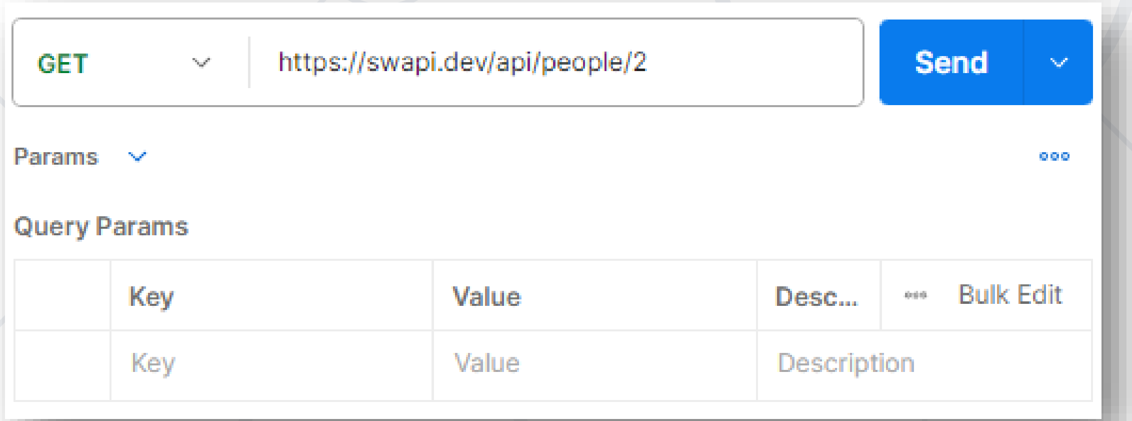
API scenario testing
Test API scenarios by iterating through a data set and triggering workflows based on responses.

Data visualization
Turn complex response data into an easy-to-understand visualization.

Authorization methods
Learn more about different authorization types and quickly set up auth helpers for your API in...

Postman – Send Your First Request

- Create a new "GET" request to the following link
  - https://swapi.dev/api/people/2



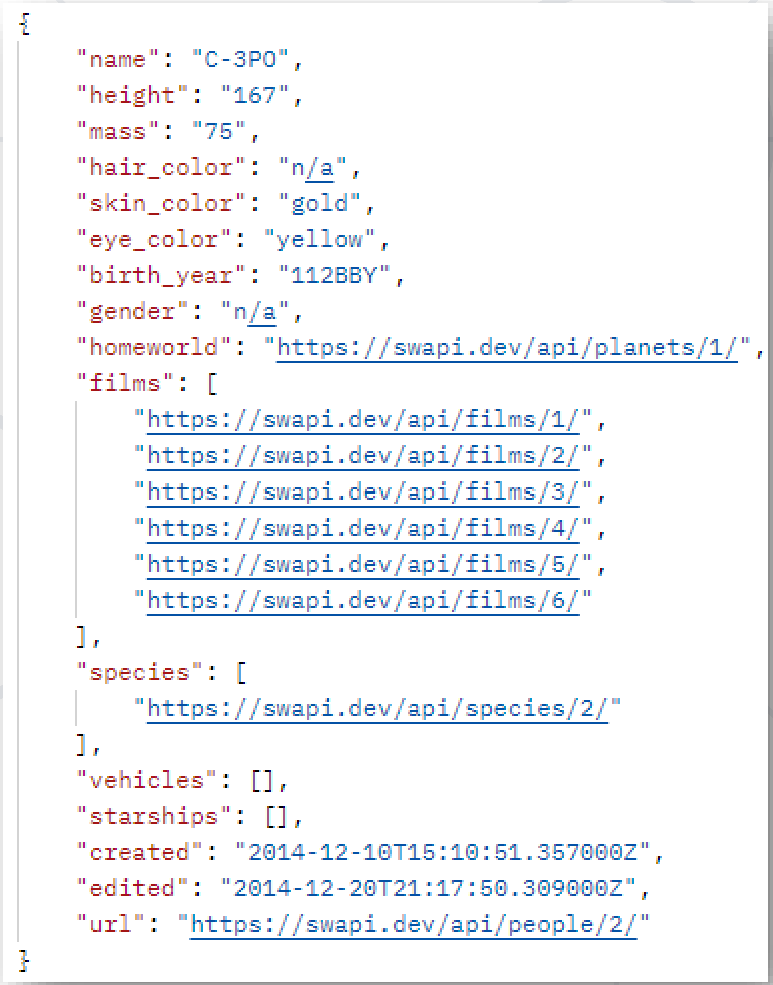
GET https://swapi.dev/api/people/2 Send

Params

Query Params

	Key	Value	Desc...	Bulk Edit
	Key	Value	Description	

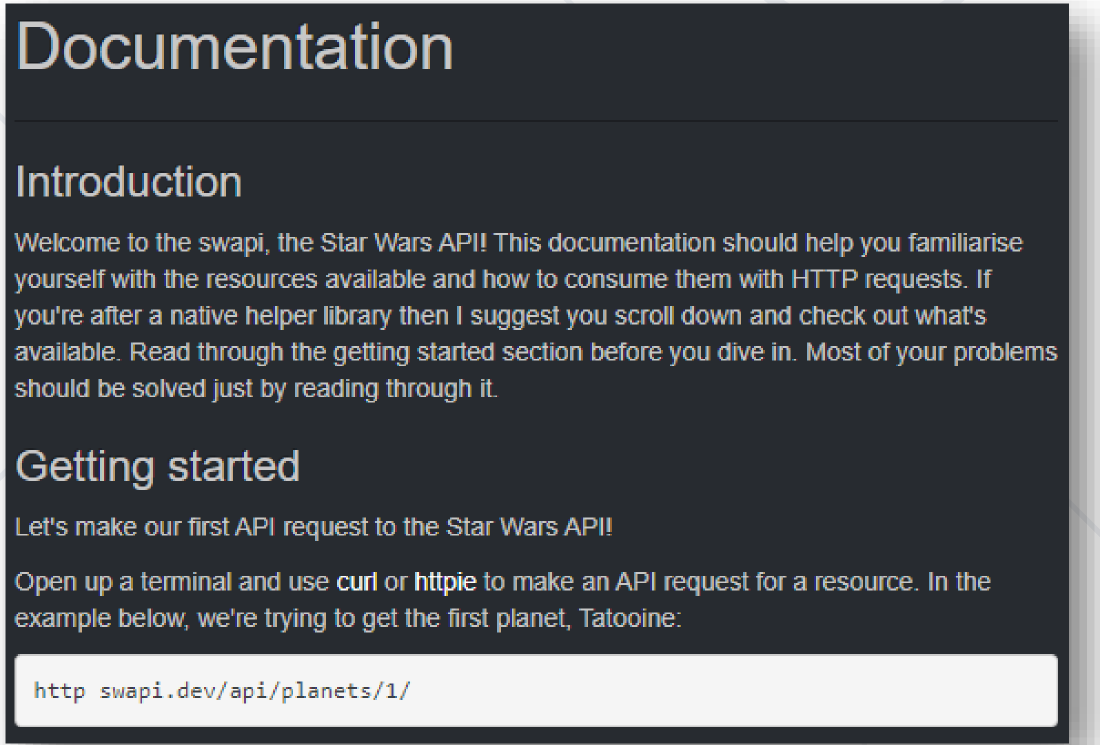
- You should receive detailed information about Star Wars person C-3PO



```
{
    "name": "C-3PO",
    "height": "167",
    "mass": "75",
    "hair_color": "n/a",
    "skin_color": "gold",
    "eye_color": "yellow",
    "birth_year": "112BBY",
    "gender": "n/a",
    "homeworld": "https://swapi.dev/api/planets/1/",
    "films": [
        "https://swapi.dev/api/films/1/",
        "https://swapi.dev/api/films/2/",
        "https://swapi.dev/api/films/3/",
        "https://swapi.dev/api/films/4/",
        "https://swapi.dev/api/films/5/",
        "https://swapi.dev/api/films/6/"
    ],
    "species": [
        "https://swapi.dev/api/species/2/"
    ],
    "vehicles": [],
    "starships": [],
    "created": "2014-12-10T15:10:51.357000Z",
    "edited": "2014-12-20T21:17:50.309000Z",
    "url": "https://swapi.dev/api/people/2/"
}
```

Postman – Practice Another Request

- Each API has **documentation**, where you can see how to use the API. You can find the documentation of this API here
  - https://swapi.dev/documentation
- Try a few more requests
  - Get request for planets
  - Get request for films



Documentation

Introduction

Welcome to the swapi, the Star Wars API! This documentation should help you familiarise yourself with the resources available and how to consume them with HTTP requests. If you're after a native helper library then I suggest you scroll down and check out what's available. Read through the getting started section before you dive in. Most of your problems should be solved just by reading through it.

Getting started

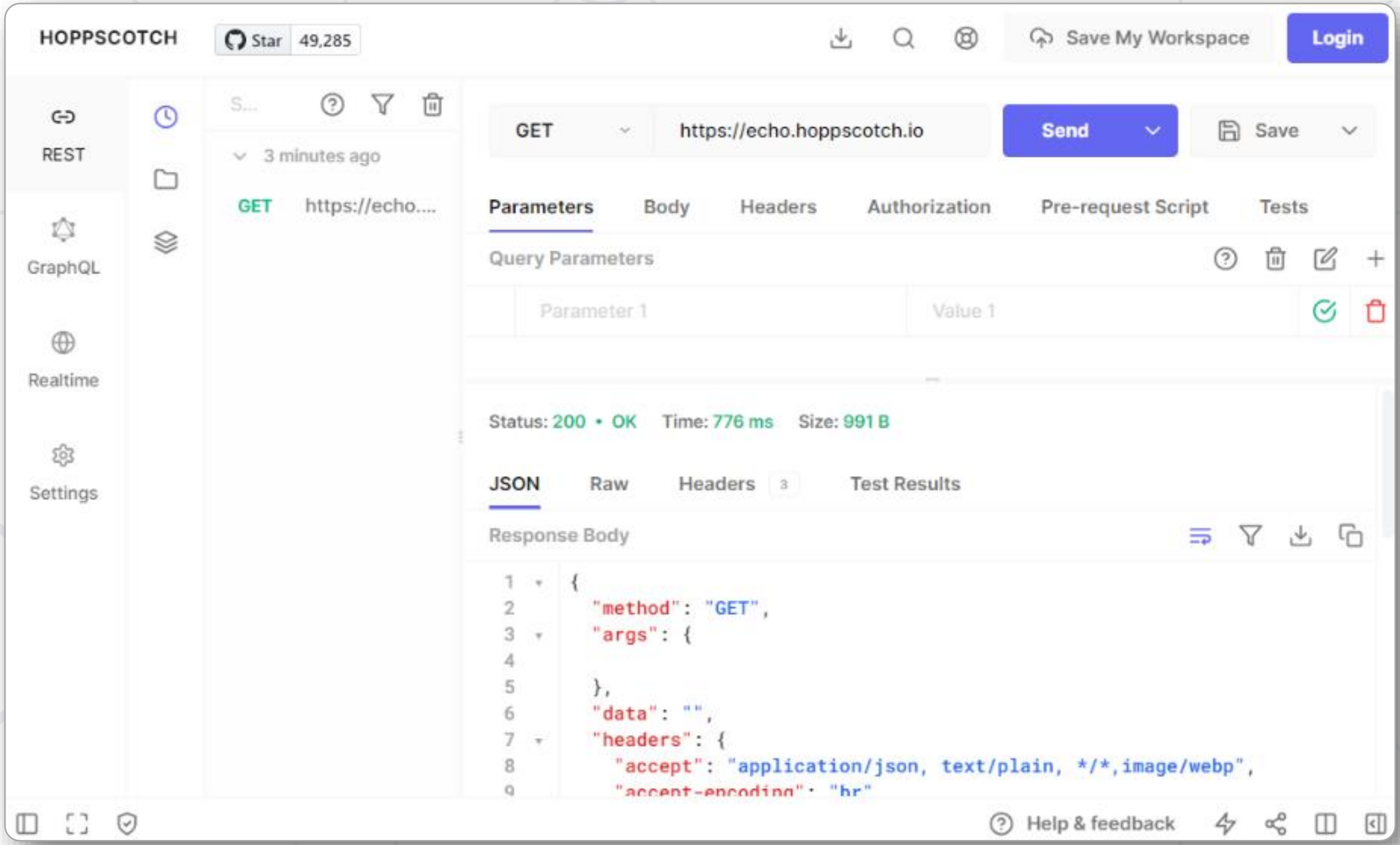
Let's make our first API request to the Star Wars API!

Open up a terminal and use curl or httpie to make an API request for a resource. In the example below, we're trying to get the first planet, Tatooine:

```
http swapi.dev/api/planets/1/
```

Hoppscotch

- Hoppscotch.io
- Postman alternative



Status: 200 • OK Time: 776 ms Size: 991 B

```
{
  "method": "GET",
  "args": {

  },
  "data": "",
  "headers": {
    "accept": "application/json, text/plain, */*,image/webp",
    "accept-encoding": "br"
```

Swagger

API Documentation and Testing Tool

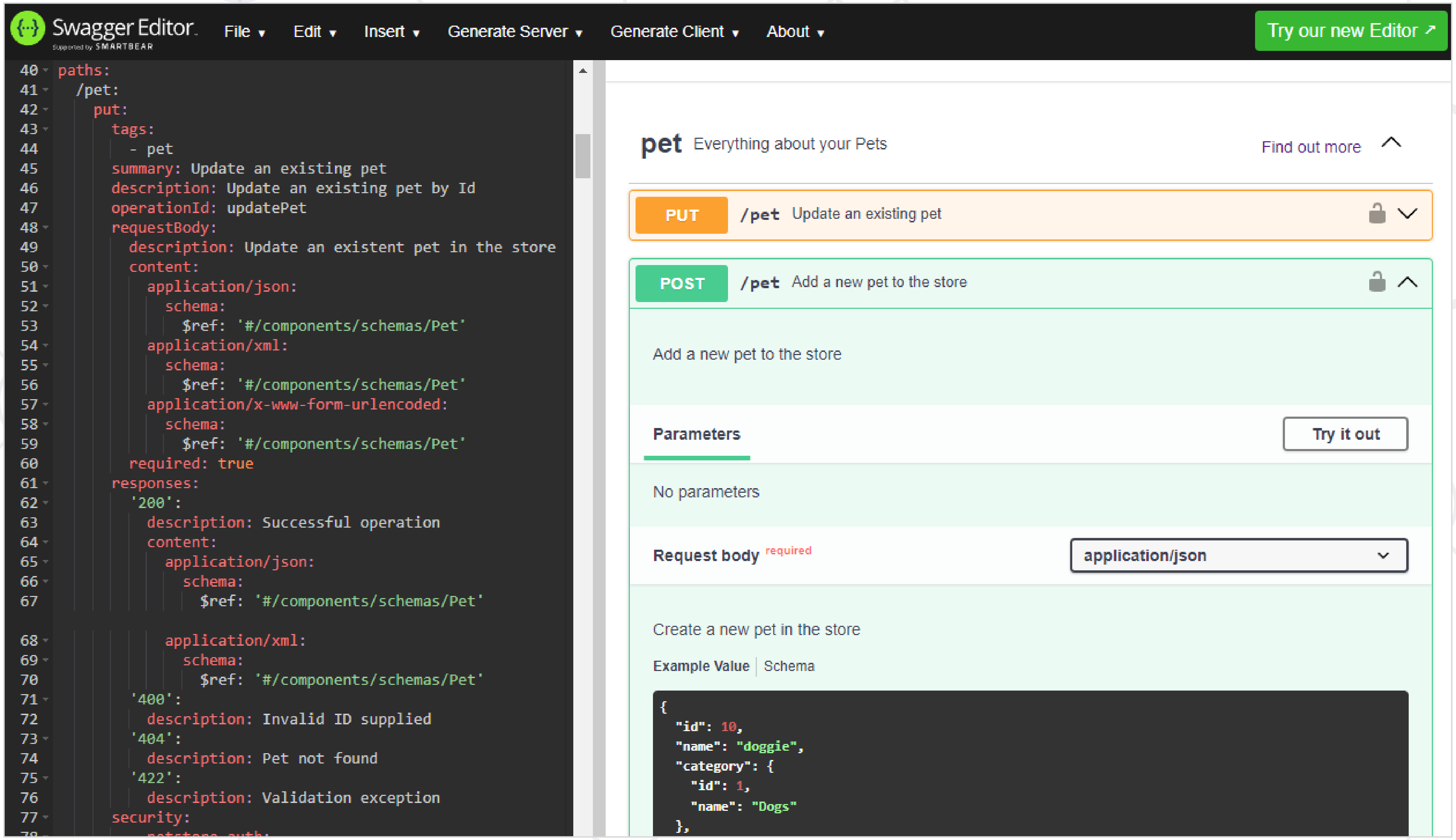
What is Swagger UI?

- Swagger is an **open-source framework** that helps developers **design**, **build**, **document**, and **consume** **RESTful web services**
- It simplifies **API development** by offering a **standardized approach to documenting APIs**
- Provides clear guidance for developers on **how to interact with an API**
- Reduces confusion and errors during integration

Benefits

- It creates **consistent documentation** across teams and projects
- Developers can try **API calls** directly in the browser with Swagger UI
- With tools like **Swagger Codegen**, you can automatically generate client SDKs and server stubs
- Validates **API requests** and **responses** against the API contract
- Teams can **work together** on API design and documentation

Swagger UI Example



```yaml
paths:
  /pet:
    put:
      tags:
        - pet
      summary: Update an existing pet
      description: Update an existing pet by Id
      operationId: updatePet
      requestBody:
        description: Update an existent pet in the store
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Pet'
          application/xml:
            schema:
              $ref: '#/components/schemas/Pet'
          application/x-www-form-urlencoded:
            schema:
              $ref: '#/components/schemas/Pet'
        required: true
      responses:
        '200':
          description: Successful operation
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Pet'

            application/xml:
              schema:
                $ref: '#/components/schemas/Pet'
        '400':
          description: Invalid ID supplied
        '404':
          description: Pet not found
        '422':
          description: Validation exception
      security:
```

pet Everything about your Pets — Find out more

PUT `/pet` Update an existing pet

POST `/pet` Add a new pet to the store

Add a new pet to the store

Parameters — Try it out

No parameters

Request body required — application/json

Create a new pet in the store

Example Value | Schema

```json
{
  "id": 10,
  "name": "doggie",
  "category": {
    "id": 1,
    "name": "Dogs"
  },
```

Summary

- Hypertext Transfer Protocol
- HTTP Response Status Codes
- RESTful Services – Introduction
- RESTful APIs
- Authentication and Authorization in REST API

Software University

Questions?

SoftUni

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

SoftUni Diamond Partners

Software University

SmartIT

createX

VIVACOM

INDEAVR
Serving the high achievers

Coca-Cola HBC Bulgaria

Postbank
Решения за твоето утре

AMBITIONED

SUPER HOSTING .BG

DRAFT KINGS
THE CROWN IS YOURS

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License