

Architektura restowa

REST API

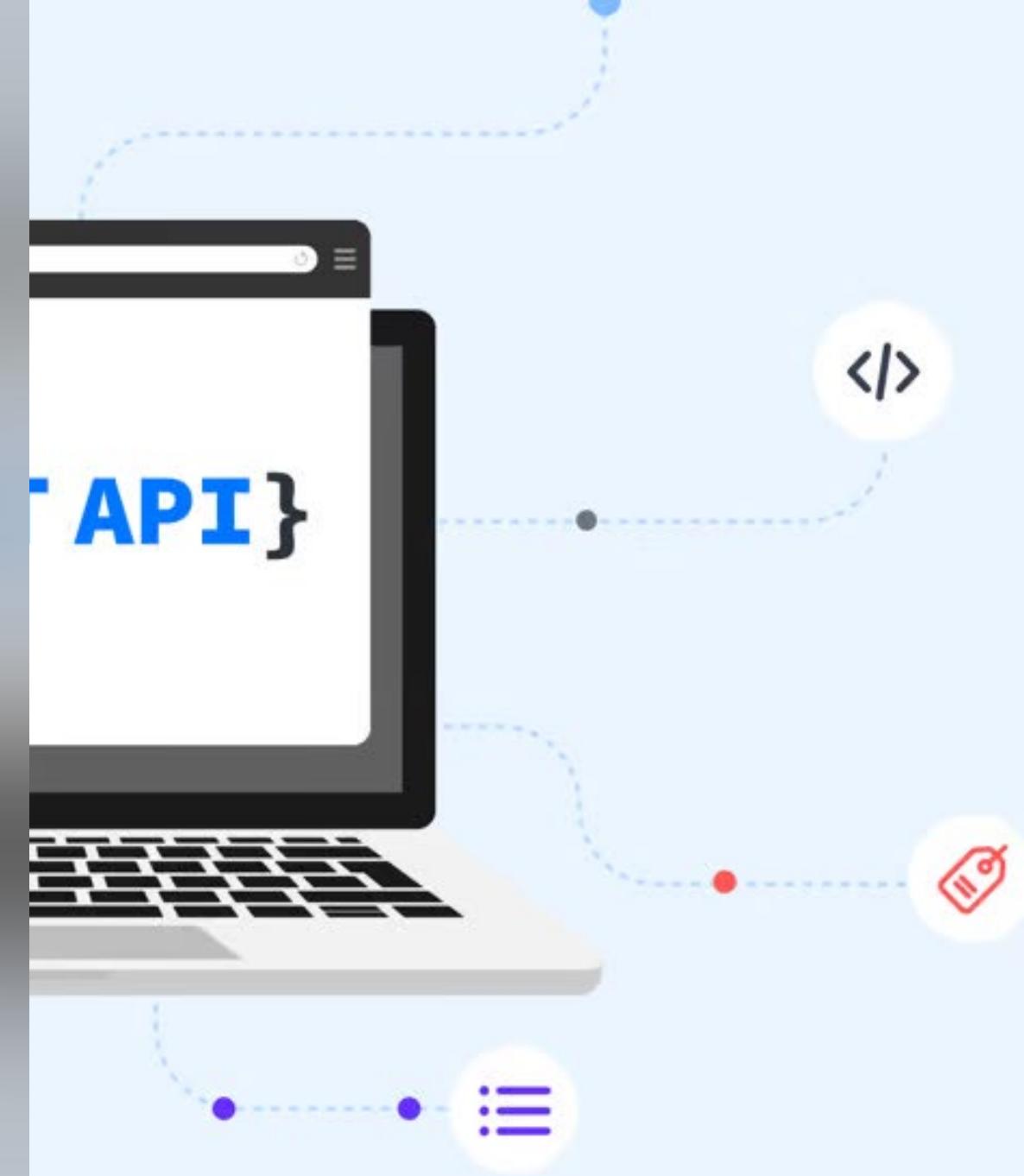
Dawid Mikoś

Edukacja techniczno-informatyczna

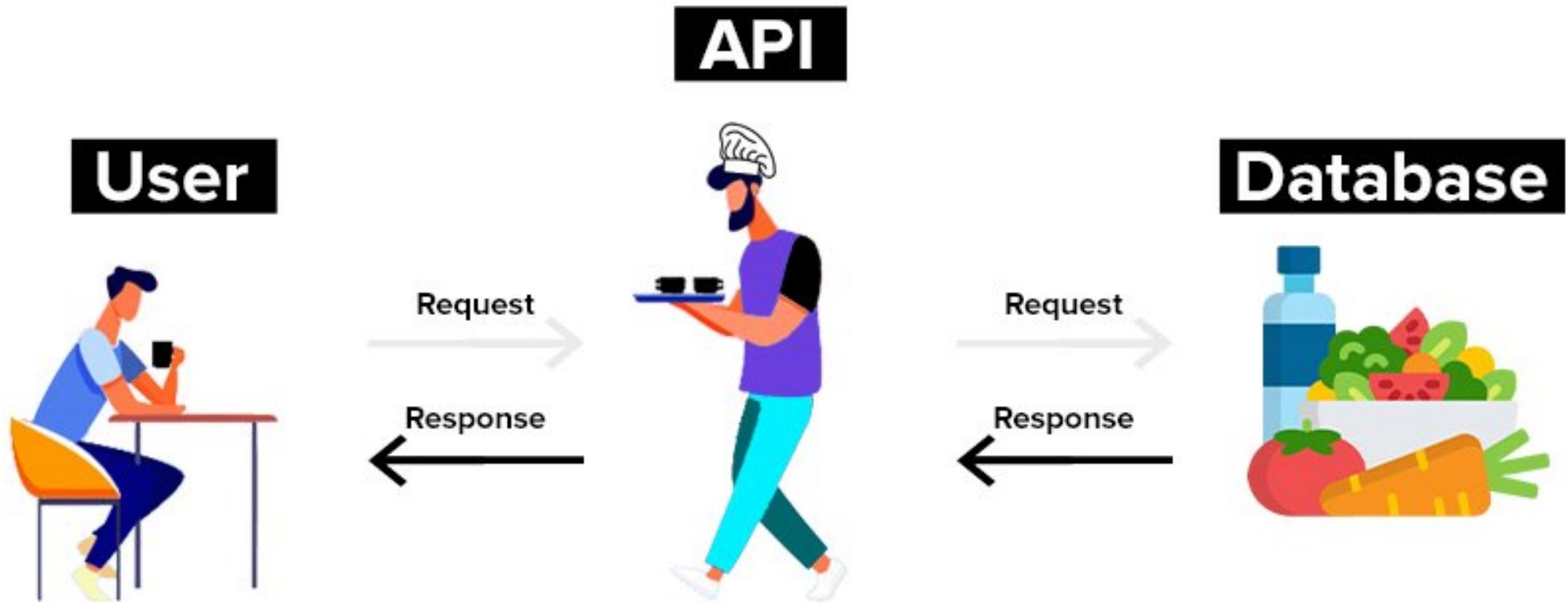
Akademia Górnictwo-Hutnicza im. Stanisława Staszica w Krakowie
AGH University of Krakow

REST, czyli **R**epresentational **S**tate **T**ransfer to styl architektury oprogramowania, opierający się o zbiór określonych reguł opisujących jak definiowane są zasoby, a także umożliwiających dostęp do nich. REST został zaprezentowany przez Roya Fieldinga w 2000 roku.

Należy podkreślić, że REST jest stylem architektury oprogramowania, a nie standardem



Czym jest REST API?



Czym jest
REST API?



Klient-serwer

1

klient musi skomunikować się z serwerem aby uzyskać dostęp do usługi



Jednolity interfejs

dla realizacji konkretnego działania
jest tylko jedna droga dojścia

2



3

Stateless

bez względu na to jakie wcześniej operacje zostały wykonane, to zawsze każda kolejna operacja nie będzie zmieniała rezultatu dla innej

Cacheable

wszystkie żądania, które trafiają do REST'a powinny być zapisane w buforze, jeżeli serwer nie jest w stanie obsłużyć na bieżąco wszystkich żądań.

4

System warstwowy

jasny podział na warstwy. Użytkownik niekoniecznie musi mieć dostęp bezpośrednio do aplikacji – aplikacja może mieć dodatkową warstwę np. uwierzytelniającą.

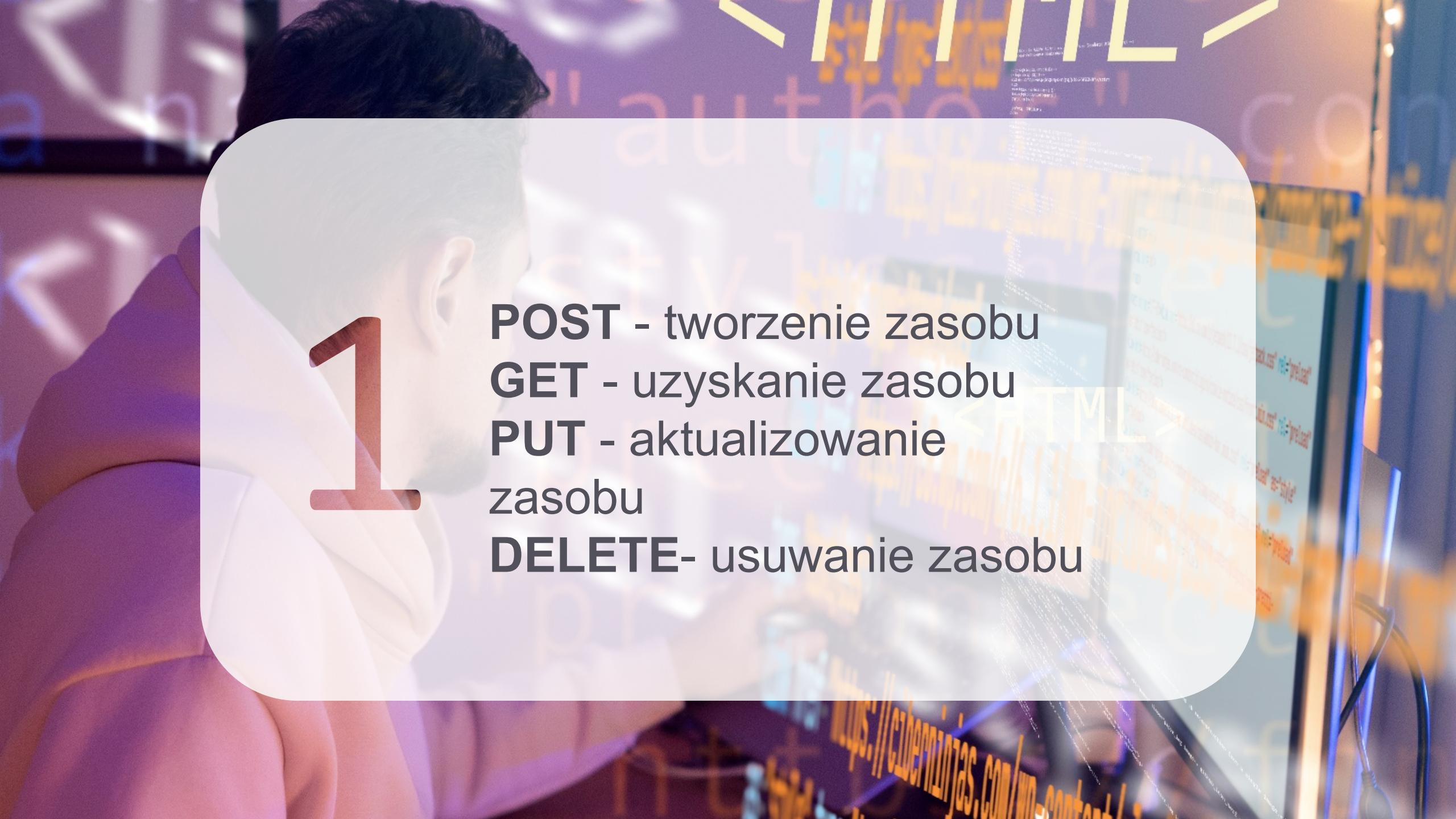
6

Kod na żądanie



Użytkownik po wykonaniu danego żądania, może otrzymać odpowiedź zwrotną w formie kodu, który następnie może wykorzystać w swojej aplikacji.

REST API
komunikuje się
za pomocą
żądań HTTP



1

POST - tworzenie zasobu

GET - uzyskanie zasobu

PUT - aktualizowanie
zasobu

DELETE- usuwanie zasobu

2

PATCH- aktualizuje jedynie
część zasobu

HEAD- GET ale bez body

TRACE- używane przy
testowaniu połączeń

OPTIONS- sprawdzenie
dostępnych metod

CONNECT- stworzenie tunelu
poprzez serwery proxy

```
1 from flask import Flask, jsonify, Response
2 from flask import render_template
3 from flask import request
4 import xml.etree.ElementTree as ET
5
6 app = Flask(__name__)
7
8 @app.route('/api/greeting', methods=['GET'])
9 def get_greeting():
10     message = {'message': 'Hello, welcome to the RESTful API!'}
11
12     # Formatowanie odpowiedzi w zależności od nagłówka Accept
13     if request.headers.get('Accept') == 'application/json':
14         return jsonify(message)
15     elif request.headers.get('Accept') == 'application/xml':
16         xml_data = dicttoxml(message)
17         return Response(xml_data, content_type='application/xml')
18     elif request.headers.get('Accept') == 'text/html':
19         return render_template('greeting.html', message=message)
20     elif request.headers.get('Accept') == 'text/plain':
21         return str(message)
22     elif request.headers.get('Accept') == 'application/python':
23         return repr(message)
24     else:
25         return jsonify(message)
26
27 def dicttoxml(d):
28     # Funkcja do konwersji słownika do formatu XML
29     root = ET.Element("root")
30     for key, value in d.items():
31         ET.SubElement(root, key).text = str(value)
32     return ET.tostring(root)
33
34 if __name__ == '__main__':
35     app.run(debug=True)
```

```
1 <!-- templates/greeting.html -->
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Greeting Page</title>
8 </head>
9 <body>
10    <h1>{{ message.message }}</h1>
11 </body>
12 </html>
```

GET <http://127.0.0.1:5000/api/greeting>

Params Authorization **Headers (7)** Body Pre-request Script Tests Settings

headers (6 hidden)

	Key	Value
<input checked="" type="checkbox"/>	Accept	text/plain
	Content-Type	



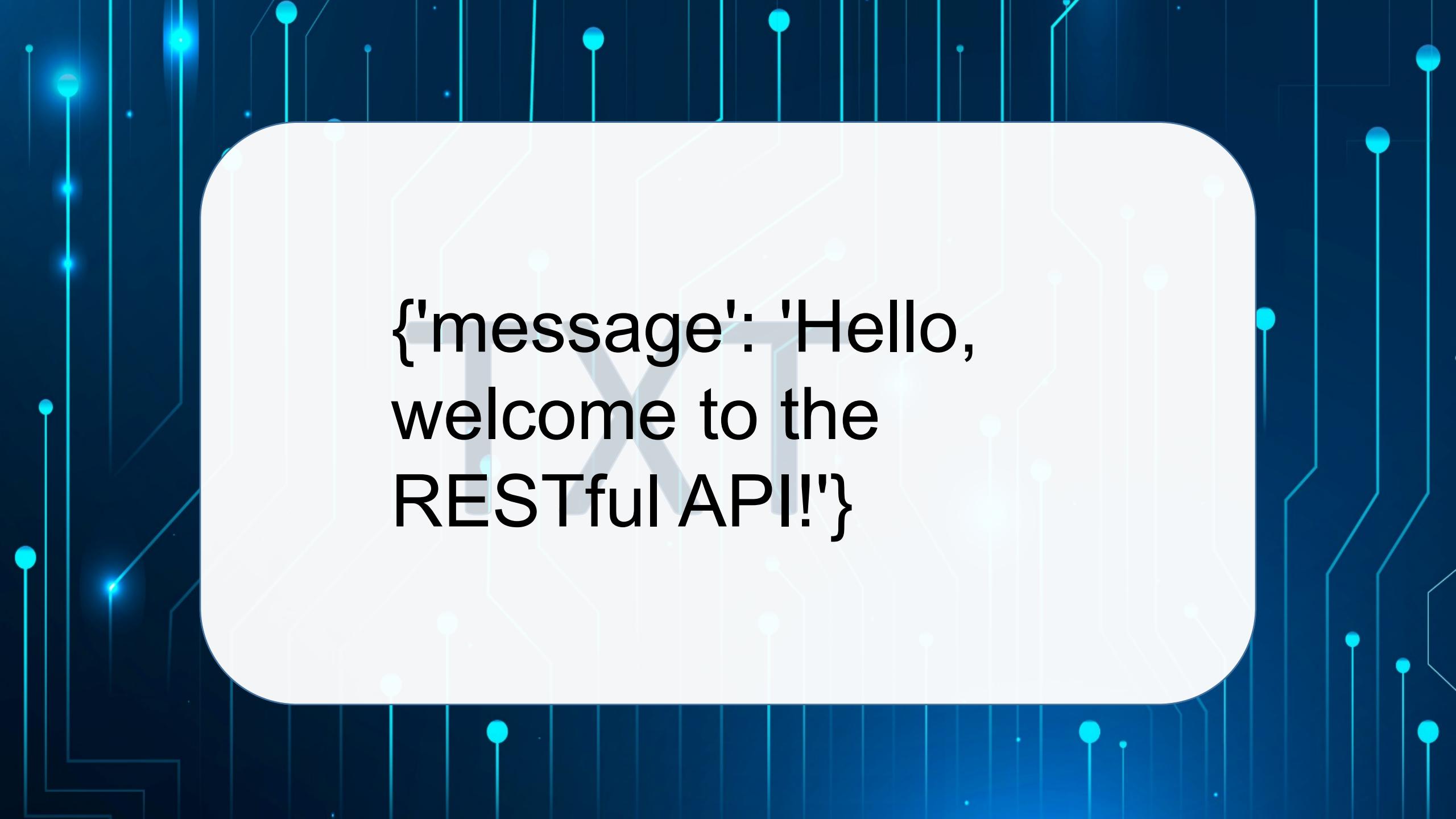
**Formaty danych
jakie możemy
otrzymywać z
REST API**

```
{  
  "message": "Hello, welcome to the RESTful API!"  
}
```

```
<root>
    <message>Hello, welcome
to the RESTful API!</message>
</root>
```

```
<!-- templates/greeting.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Greeting Page</title>
</head>
<body>
    <h1>Hello, welcome to the RESTful API!</h1>
</body>
</html>
```

```
{'message': 'Hello, welcome to the RESTful API!'}
```



```
{'message': 'Hello,  
welcome to the  
RESTful API!'}
```

Dziękuję za uwagę



JSON Web Token, JSON, XML/JSON

WOJCIECH PASIEKA



Wprowadzenie

- JSON, XML, JWT są formatami danych stosowanymi w programowaniu do reprezentowania informacji w formie tekstowej.
- Są niezależne od platformy, co oznacza, że mogą być używane w różnych systemach i językach programowania.
- Są szeroko stosowane w aplikacjach internetowych oraz w obszarach, gdzie istnieje potrzeba przesyłania, przechowywania i reprezentowania danych

Definicja i Struktura:

- Lekki i czytelny format wymiany danych.
- Oparty na dwóch głównych strukturach: obiektach i tablicach.
- Obiekty w JSON są ograniczone do par klucz-wartość
- Jego składnia jest prostsza i bardziej zwięzła niż XML, co sprawia, że jest łatwy do zrozumienia. Struktura obiektu JSON jest idealna do reprezentowania danych w sposób zorganizowany i zrozumiały dla programistów.

Zastosowanie:

- W aplikacjach internetowych odpowiedzialny za przesyłanie danych między serwerem a klientem.
- Współpraca z interfejsami API (Application Programming Interface).
- Konfiguracja i wymiana danych w formie tekstowej.

```
{  
  "imie": "John",  
  "nazwisko": "Doe",  
  "wiek": 30,  
  "miasto": "Nowy Jork",  
  "hobby": ["podróże", "sport", "czytanie"]  
}
```

JSON (JavaScript Object Notation)

XML (eXtensible Markup Language)

DEFINICJA I STRUKTURA:

- JĘZYK ZNACZNIKÓW UŻYWANY DO REPREZENTOWANIA STRUKTURALNYCH DANYCH W FORMIE TEKSTOWEJ.
- MOŻE REPREZENTOWAĆ BARDZIEJ ZŁOŻONE HIERARCHIE NIŻ JSON, NIE MA KONKRETYCH STRUKTUR DANYCH
- SKŁADNIA XML OPIERA SIĘ NA TAGACH, KTÓRE OTACZAJĄ DANE I NADAJĄ IM STRUKTURĘ HIERARCHICZNĄ:
`<NAZWISKO>DOE</NAZWISKO>`
`<OSOBA WIEK="30">JOHN</OSOBA>`
`<?XML VERSION="1.0" ENCODING="UTF-8"?> I <\XML>`.
- XML JEST BARDZIEJ ROZBUDOWANY NIŻ JSON, CO SPRAWIA, ŻE MOŻE BYĆ BARDZIEJ CZYTELNY DLA LUDZI W PRZYPADKU BARDZIEJ SKOMPLIKOWANYCH STRUKTUR DANYCH.

ZASTOSOWANIE:

- XML JEST SZEROKO STOSOWANY W RÓŻNYCH DZIEDZINACH, TAKICH JAK KONFIGURACJA USTAWIEŃ, WYMIANA DANYCH POMIĘDZY SYSTEMAMI, PRZECHOWYWANIE INFORMACJI O STRUKTURZE DOKUMENTÓW, ITP.
- PRZECHOWYWANIE DOKUMENTÓW Z HIERARCHICZNĄ STRUKTURĄ, TAKICH JAK DANE KSIĘGOWE.

JSON Web Token (JWT)

► Definicja i struktura:

- JWT, czyli JSON Web Token, to standard otwartego formatu do reprezentacji informacji między dwiema stronami w formie, którą można łatwo przesyłać między użytkownikami i zachować jej integralność
- Składa się z trzech głównych części: nagłówka (header), treść (payload) i sygnatury (signature):

➤ Zastosowanie:

- Autentykacji i autoryzacji użytkowników

Przykładowy token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG  
4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.Sf1KxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Nagłówek:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
```

Treść:

```
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ
```

Sygnatura:

```
Sf1KxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

➤ Przewagi:

- Mobilność: JWT może być przesyłany między różnymi systemami i usługami.
- Bezpieczeństwo: Sygnatura umożliwia zweryfikowanie, czy dane w tokenie nie zostały sfałszowane.
- Skalowalność

Różnice

- **JWT vs. JSON:**
 - **JWT:** Specyficzny rodzaj JSON używany do bezpiecznej transmisji informacji. Zawiera nagłówek, treść i sygnaturę.
 - **JSON:** Ogólna notacja danych oparta na obiektach i tablicach, stosowana w przesyłaniu danych między aplikacjami.
- **JWT vs. XML:**
 - **JWT:** Zazwyczaj bardziej kompaktowy, bez potrzeby przechowywania na serwerze, używany w kontekście autentykacji.
 - **XML:** Bardziej rozbudowany, często stosowany w konfiguracji ustawień, przechowywaniu danych w strukturach hierarchicznych.
- **JSON vs. XML:**
 - **JSON:** Lekki, bardziej zwięzły, czytelny, często używany w aplikacjach internetowych.
 - **XML:** Bardziej rozbudowany, umożliwia bardziej złożone struktury danych, stosowany w różnych dziedzinach, gdzie ważna jest hierarchiczna organizacja informacji.



Dziękuję za uwagę!



{

Interfejsy programowania aplikacji

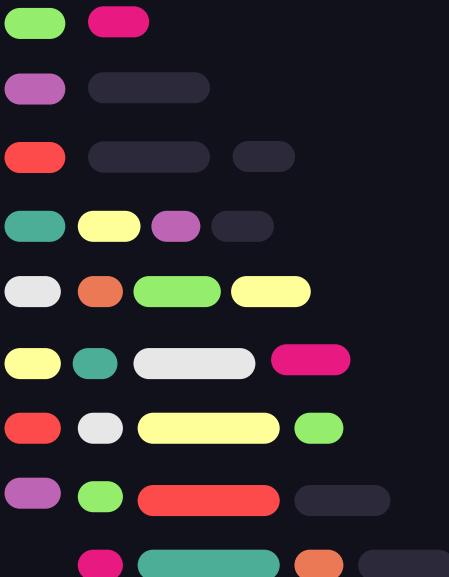
Mateusz Kulig, eti III rok

• • •

}

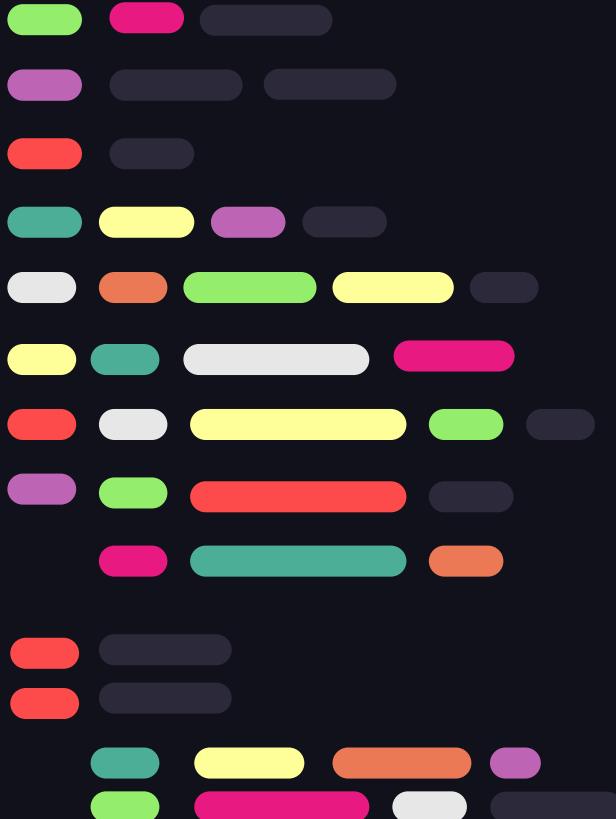
0 czym to?

01 Czym jest API?



02 Jak się porozumiewa?

03 Przykłady



API }

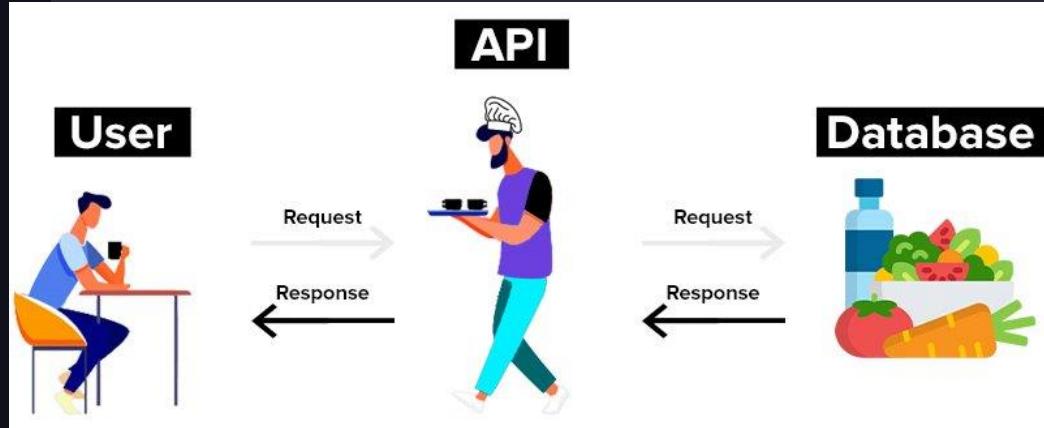
< Jest to zestaw reguł opisujący jak kod może komunikować się z danym zewnętrznym programem, który pozwala na użycie danej funkcjonalności >





Co to właściwie znaczy?

Interfejsy są tworzone przez twórców aplikacji i bibliotek aby móc korzystać z funkcji tych programów bez ich dogłębnej znajomości.



Rodzaje API

Interfejsy programowania aplikacji można podzielić na kilka rodzajów:

Natywne APIs

Standardowe biblioteki języków programowania, dostarczające takie funkcje jak `print()` do wyświetlania tekstu w konsoli.

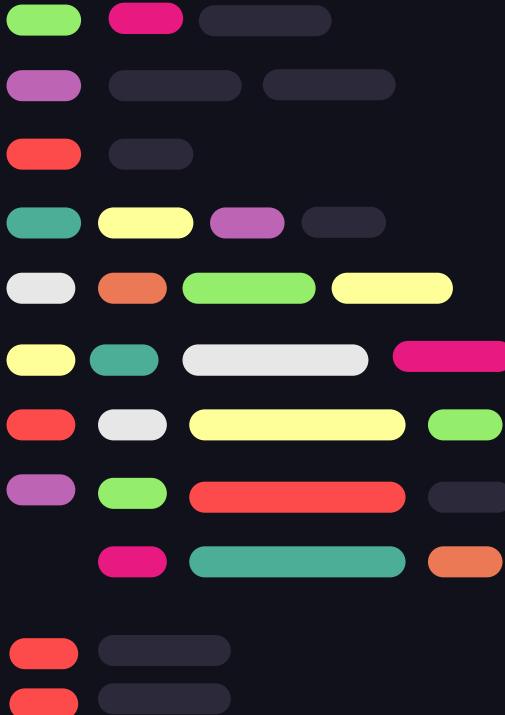
Browser APIs

Zbiór funkcji przeglądarki do modyfikowania stron, takich jak metody i pola obiektów `document` czy `navigator`.

Web APIs

Zewnętrzne API udostępnione publicznie w internecie, pozwalające na np. zdalne umieszczenie wpisu lub uzyskanie informacji w formatach JSON lub XML.

Jak się porozumiewają?



Interfejsy programowania aplikacji mogą się komunikować w dowolny, lecz opisany sposób. Najczęstsze rodzaje komunikacji:

- Poprzez wykonanie jakieś czynności i zwrócenie odpowiedniego kodu
- Poprzez zwrócenie danych np. w formacie JSON
- Poprzez zwrócenie obiektu na którym możliwe są dalsze działania

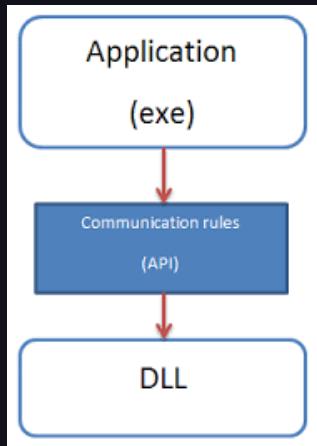
Rodzaj zwracanych danych zawsze powinien być opisany w dokumentacji.



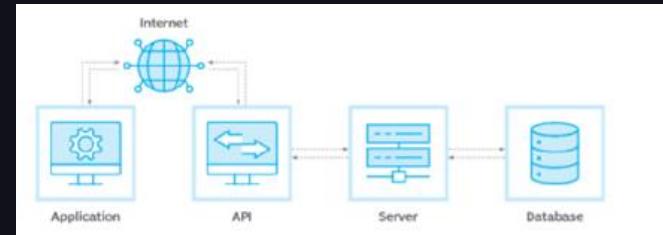
Właściwa komunikacja

{

Lokalnie



Przez Internet

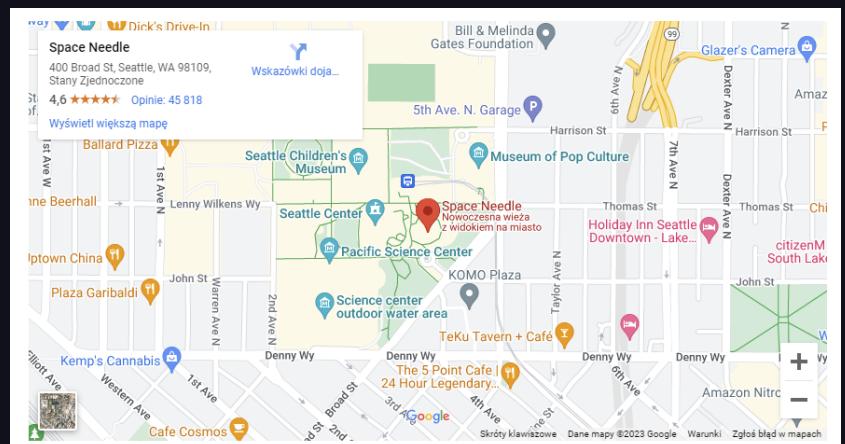


}



Przykład API Google Maps

```
<iframe  
  width="600"  
  height="450"  
  style="border:0"  
  loading="lazy"  
  allowfullscreen  
  referrerpolicy="no-referrer-when-downgrade"  
  src="https://www.google.com/maps/embed/v1/place?key=API_KEY  
    &q=Space+Needle,Seattle+WA">  
</iframe>
```



Przykład API marginem.pl



```
[{"id": 12147, "nick": "x-X matikokspl x-X", "world": "gordion", "lvl": 75, "prof": "p", "gender": "m",...},...]  
▼ 0: {"id": 12147, "nick": "x-X matikokspl x-X", "world": "gordion", "lvl": 75, "prof": "p", "gender": "m",...}  
  clan: 0  
  clan_rank: 0  
  gender: "m"  
  icon: "/pal/70/m_pall2.gif"  
  id: 12147  
  last: 1699054116  
  lvl: 75  
  nick: "x-X matikokspl x-X"  
  prof: "p"  
  world: "gordion"
```



Przykład API systemu Windows

```
#include <windows.h>

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
    PWSTR pCmdLine, int nCmdShow) {

    MSG msg;
    HWND hwnd;
    WNDCLASSW wc;

    wc.style      = CS_HREDRAW | CS_VREDRAW;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.lpszClassName = L"Window";
    wc.hInstance   = hInstance;
    wc.hbrBackground = GetSysColor(COLOR_3DFACE);
    wc.lpszMenuName = NULL;
    wc.lpfnWndProc = WndProc;
    wc.hCursor     = LoadCursor(NULL, IDC_ARROW);
    wc.hIcon       = LoadIcon(NULL, IDI_APPLICATION);

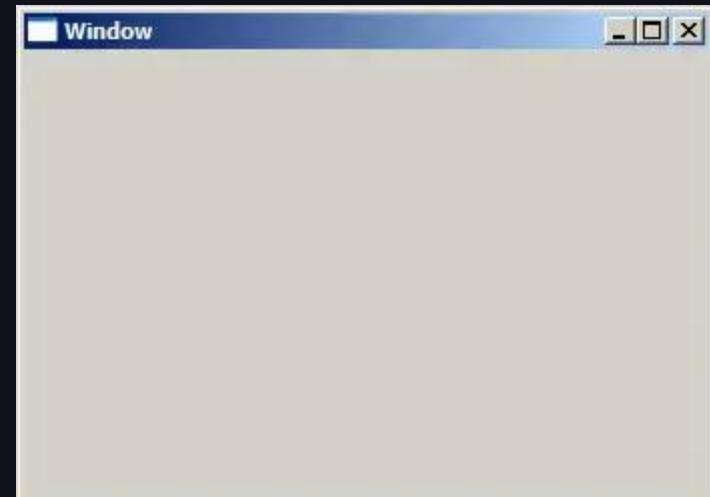
    RegisterClassW(&wc);
    hwnd = CreateWindow(wc.lpszClassName, L"Window",
        WS_OVERLAPPEDWINDOW | WS_VISIBLE,
        100, 100, 350, 250, NULL, NULL, hInstance, NULL);

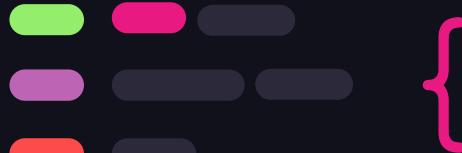
    ShowWindow(hwnd, nCmdShow);
    UpdateWindow(hwnd);

    while (GetMessage(&msg, NULL, 0, 0)) {

        DispatchMessage(&msg);
    }

    return (int) msg.wParam;
}
```



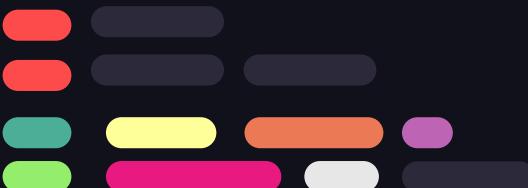


Dziękuję za uwagę!



źródła

- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction
- https://www.w3schools.com/js/js_api_intro.asp
- <https://support.apple.com/pl-pl/guide/shortcuts-mac/apd2e30c9d45/mac>
- <https://bykowski.pl/rest-api-efektywna-droga-do-zrozumienia/>
- <https://www.techtarget.com/searchapparchitecture/definition/API-economy>
- <https://developers.google.com/maps/documentation/embed/get-started?hl=pl>
- <https://zetcode.com/gui/winapi/window/>

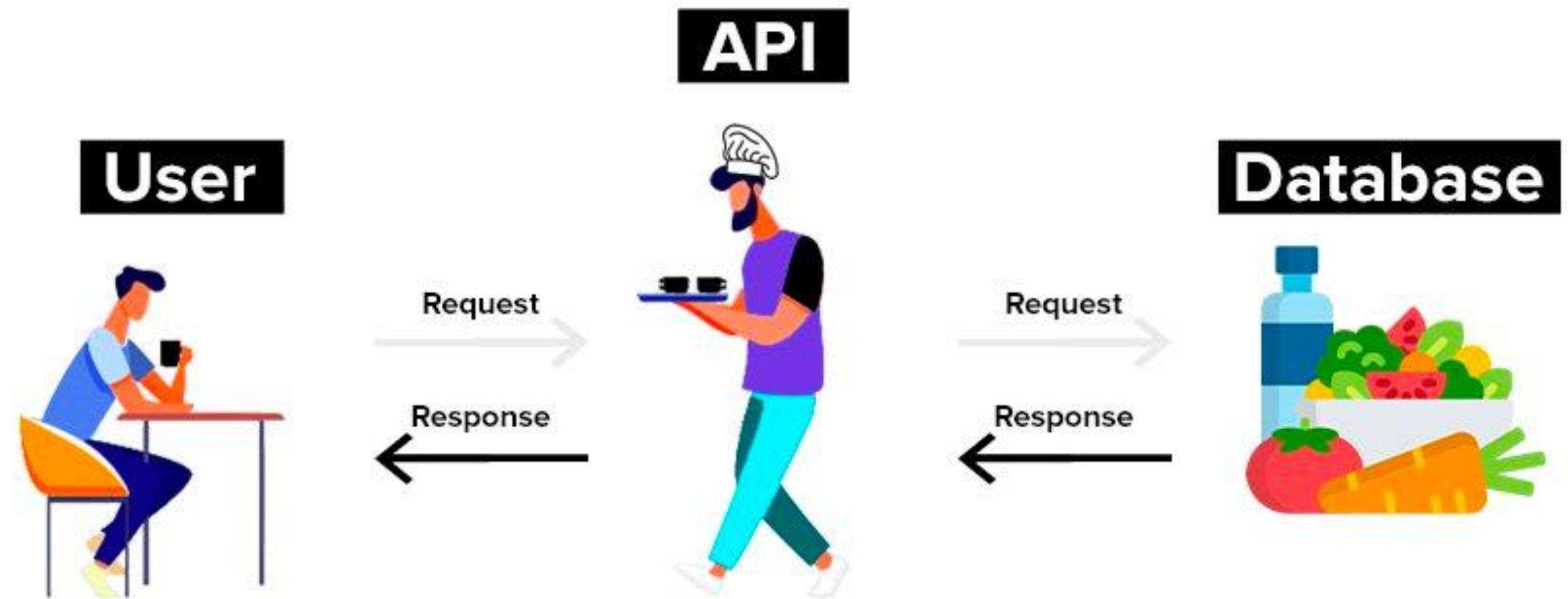


API

Application Programming Interface

Zestaw reguł definiujących komunikację między systemami komputerowymi oraz między systemem komputerowym a człowiekiem.

W przypadku REST API komunikacja następuje najczęściej między **aplikacją kliencką a aplikacją serwerową**. Aplikacją kliencką może być aplikacja webowa, mobilna lub inny, dowolny klient dający możliwość komunikacji z interfejsem.



REST

Representational State Transfer

Styl architektury oprogramowania, opierający się o zbiór określonych reguł opisujących jak definiowane są zasoby, a także umożliwiających dostęp do nich.

Opiera się o protokół HTTP(REST API komunikuje się za pomocą żądań HTTP).

Metody HTTP

HEAD,PATCH,
TRACE,OPTIONS,CONNECT

GET

pobiera informacje z serwera; nie wymaga ciała żądania

POST

tworzy nowy obiekt w bazie danych, wymaga ciała żądania z polami tego obiektu

PUT

całkowicie resetuje/aktualizuje określony obiekt w bazie danych

DELETE

usuwa obiekt z bazy danych

Kody odpowiedzi HTTP

Każda odpowiedź na zapytanie HTTP musi zawierać kod odpowiedzi.

Kody odpowiedzi HTTP są trzyznakowe i dzielą się na pięć kategorii:

- **100 - 199 (Informational)** - Informuję, że żądanie jest przetwarzane.
- **200 - 299 (Success)** - W tej grupie znajdują się kody oznaczające sukces zapytania.
- **300 - 399 (Redirection)** - Grupa kodów związana z przekierowaniami zapytań.
- **400 - 499 (Client Error)** - Kody oznaczające błędy, które są winą niewłaściwego zapytania wysłanego przez klienta.
- **500 - 599 (Server Error)** - Kody błędów, które są winą niewłaściwego działania serwera.

Odpowiedź pochodząca z REST API

Formaty danych jakie możemy otrzymywać z REST API to najczęściej:

- JSON – JavaScript Object Notation
- XML – Extensible Markup Language

Z czego najpopularniejszy jest JSON ze względu na lekkość.

Dla przykładu JSON:

```
{  
    "name": "Przemek",  
    "age": 18,  
    "website": "bykowski.pl"  
}
```

IMPLEMENTACJA REST API



METODY HTTP

GET

- Pobiera określony zasób według podanego identyfikatora.

POST

- Tworzy nowy zasób.
- Jest wykorzystywany do wszystkich innych operacji, które nie wpisują się w ramy innych metod.
- Może służyć do pobierania danych w przypadku kiedy musimy w ramach body dostarczyć dodatkowe parametry.

PUT

- Aktualizuje dany zasób na podstawie podanego identyfikatora.
- Może służyć do tworzenia nowego zasobu jeśli jego identyfikator jest znany.

DELETE

- Usuwa określony zasób według identyfikatora

PATCH

- Aktualizuje część wskazanego zasobu.

200 OK

- Najpopularniejszy kod.

Służy do informowania operacji zakończonej sukcesem.

201 CREATED

- Pomyślne utworzenie zasobu (przez POST lub PUT). Nagłówek Location powinien zawierać łącze do nowo utworzonego zasobu.
- Treść odpowiedzi jest opcjonalna.

204 NO CONTENT

Operacja zakończona sukcesem, jednak nic nie znajduje się w treści odpowiedzi. Często używane w operacjach DELETE i PUT.

400 BAD REQUEST

Ogólny błąd wykonania żądania po stronie klienta. Na przykład brakujące lub nieprawidłowe dane przesypane do serwera.



STATUSY ODPOWIEDZI

401 UNAUTHORIZED

Nieautoryzowany dostęp – żądanie zasobu wymaga uwierzytelnienia.

403 FORBIDDEN

Brak uprawnień użytkownika do danego zasobu ze względu na brak właściwych prawa użytkownika lub ograniczenia czasowe.

404 NOT FOUND

Nie znaleziono – serwer nie odnaleźał wskazanego zasobu.

409 CONFLICT

Próba wykonania operacji, która jest niedozwolona. Np. utworzenie zasobu, podczas gdy wymagana jest unikalność.

500 INTERNAL SERVER ERROR

Wewnętrzny błąd serwera – dane żądanie nie może zostać zrealizowane ze względu na błąd po stronie serwera.



REGUŁY OBSŁUGI METOD HTTP

Odpowiedź

Metoda	Idempotentność	Bezpieczność	Ciało żądania	Ciało odpowiedzi	Odpowiedź
GET	✓	✓	✗	✓	<ul style="list-style-type: none">- 200 (OK) - z zasobem/elementem zwracanym- 404 (Not Found) - jeśli zasoby nie odnaleziono
POST	✗	✗	✓	✓	<ul style="list-style-type: none">- 201 (Created) - jeśli udało się stworzyć pomyślnie- Header Location powinien zawierać link do zasobu- 409 (Conflict) - jeśli dany zasób już istnieje
PUT	✓	✗	✓	✓	<ul style="list-style-type: none">- 200 (OK) lub 204 (No Content)- 404 (Not Found) jeśli nie odnaleziono ID.- 409 (Conflict) - jeśli w ciele zostało przekazane inne ID niż w ścieżce
DELETE	✓	✗	✗	✗	<ul style="list-style-type: none">- 200 (OK).- 404 (Not Found) jeśli nie odnaleziono ID.
PATCH	✗	✗	✓	✓	<ul style="list-style-type: none">- 200 (OK) lub 204 (No Content).- 404 (Not Found) jeśli nie odnaleziono ID.- 409 (Conflict) - jeśli w ciele zostało przekazane inne ID niż w ścieżce

6 Zasad REST

1 Jednolity interfejs

Zapytanie od klienta musi zawierać komplet informacji koniecznych do wykonania zapytania/polecenia

Używaj rzeczowników zasobów

Używaj rzeczowników w liczbie mnogiej (np. /products, /users) do reprezentowania kolekcji zasobów i unikaj używania czasowników (np. /getProducts, /createUser).

Dbaj o prostotę i przewidywalność adresów URL

Projektuj intuicyjne i łatwo zrozumiałe adresy URL dla klientów, używając hierarchii zasobów do wyrażania relacji (np. /users/{id}/orders).

Ścieżka do zasobu powinna uwzględniać relacje między zasobami

customers/123/carts/456

Taki adres pokazuje hierarchię zasobów w systemie — *customer* jest rodzicem dla *cart*

Każde zapytanie od klienta musi zawierać komplet informacji, oraz że serwer nie przechowuje stanu o sesji użytkownika po swojej stronie

2 Stateless

Mechanizm pozwalający na znacznie szybsze zwrócenie odpowiedzi w przypadku powtarzających się zapytań

3 Cacheable

4 Endpointy

Adresy zasobów, powinny jednoznacznie wskazywać, do jakiego zasobu się odwołują

Ze struktury endpointu jednoznacznie powinno wynikać jaka akcja zostanie wykonana na serwerze

Punkt końcowy zawiera URI - Uniform Resource Identifier, który wskazuje jak i gdzie można znaleźć dany zasób.

Natomiast URL, czyli Uniform Resource Locator jest typem URI, który definiuje jak uzyskać dostęp do zasobu za pomocą Internetu.

Warto zapamiętać, że **każdy URL jest URI, ale nie każdy URI jest URL**.



5 System warstwowy

Żadna z warstw nie powinna bezpośrednio oddziaływać na inne warstwy.

Implementacja dodatkowych warstw i zewnętrznych API powinny być ukryte przed użytkownikiem API.

Możliwość udostępniania skryptów wykonywalnych użytkownikom.
Jest to opcjonalna reguła.

6 Kod na żądanie

RESTful API

Wszystkie wymienione regule są spełnione

RESTful

GET http://example.com/movies
GET http://example.com/movies/123
POST http://example.com/movies
PUT http://example.com/movies/123
DELETE http://example.com/movies/123

Nie RESTful

GET http://example.com/getMovieList
GET http://example.com/showMovie/123
POST http://example.com/addNewFilm
PUT http://example.com/updateMovie/123
DELETE http://example.com/removeFilm/123

Przykład

Chcemy dodać książkę

// Request:
POST example.com/books

// Body payload:
{
 "title": book1,
 "author": author1,
}

```
// if missing title -> http 400 .  
// if title not string -> http 400 .  
app.post('/api/items', (req, res) => {  
  if (req.body.title === undefined) {  
    res.status(400).send();  
  }  
  if (typeof req.body.title !== "string") {  
    res.status(400).send();  
  }  
  const new_id = new Date().valueOf();  
  items.push({ id: new_id, title: req.body.title })  
  res.status(201).send();  
})
```

Tworzymy nowy identyfikator (new_id)
na podstawie bieżącej daty i czasu.

Dodajemy nowy element do tablicy items.
Nowy element składa się z pola id
(utworzonego w poprzednim punkcie) i pola
title, którego wartość pochodzi z ciała
żądania (req.body.title).

Przykład

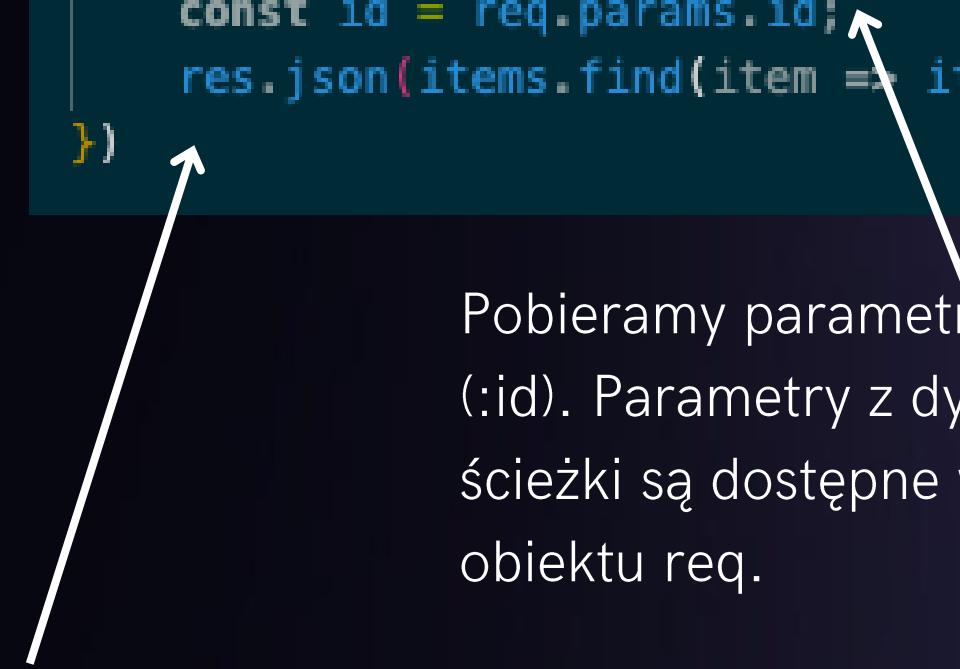
Chcemy pobrać książkę

// Request:
GET example.com/books/123

```
app.get('/api/items', (req, res) => {
  res.json(items)
})

// if item not found -> http 404
// if id not number -> http 400
app.get('/api/item/:id', (req, res) => {

  const id = req.params.id;
  res.json(items.find(item => item.id == id))
})
```



Pobieramy parametr id z częścią ścieżki (:id). Parametry z dynamiczną częścią ścieżki są dostępne w obiekcie params obiektu req.

Odpowiadamy znalezionym elementem o podanym identyfikatorze (id) z tablicy items. Jeśli żaden element nie zostanie znaleziony (null), odpowiedź będzie pusta.

Przykład

Chcemy zaktualizować książkę

// Request:

PUT example.com/books/123

// Body payload:

```
{  
  "title": book2,  
}
```

Szukamy indeks elementu o podanym identyfikatorze (id) w tablicy items.

```
// if item not found -> http 404  
// if missing title -> http 400 .  
// if title not string -> http 400  
app.put('/api/item/:id', (req, res) => {  
  
  if (typeof req.body.title != "string") {  
    res.status(400).send();  
  }  
  const id = req.params.id;  
  const index = items.indexOf(items.find(item => item.id == id));  
  items[index].title = req.body.title;  
  res.json(items[index]);  
})
```

Pobieramy identyfikator (id) z dynamicznej części ścieżki.

Aktualizujemy tytuł elementu o znalezionym indeksie na podstawie danych przesłanych w ciele żądania

Przykład

Chcemy usunąć książkę

// Request:
DELETE example.com/books/123

```
// if item not found -> http 404
// if id not number -> http 400
app.delete('/api/item/:id', (req, res) => {
  const id = req.params.id;
  const index = items.indexOf(items.find(item => item.id == id));
  items.splice(index, 1);
  res.status(204).send();
})
```

Pobiera identyfikator (id) z dynamicznej części ścieżki

Usuwa element o znalezionym indeksie z tablicy items. Metoda splice zmienia zawartość tablicy, usuwając lub zamieniając elementy.

Materiały

appmaster.io/pl/blog/szesc-zasad-odpoczynku-apis

appmaster.io/pl/blog/jak-tworzyc-punkty-koncowe-i-dlaczego-sa-one-potrzebne

appmaster.io/pl/blog/co-to-jest-rest-api-i-jak-rozni-sie-od-innych-typow

devszczepaniak.pl/wprowadzenie-do-rest-api/

devszczepaniak.pl/projektowanie-rest-api/

youtube.com/watch?v=P9b8-BrWdYs

youtube.com/watch?v=FV1Ugv1Temg&list=PL55RiY5tL51q4D-B63KBnygU6opNPFk_q&index=3

Richardson Maturity Model

Mateusz Szczepaniak
- Krupowski



Spis treści

01

Wprowadzenie

02

Podział na poziomy

03

System Pol-on

04

Zakończenie

05

Bibliografia

Co to jest system POL-on i model Richardsona?

Model Dojrzałości Richardsona (RMM) to model dojrzałości zaproponowany w 2008 roku przez Leonarda Richardsona, który klasyfikuje interfejsy API sieciowe na podstawie ich zgodności i zgodności z każdym z czterech poziomów modelu.

System POL-on to system teleinformatyczny obejmujący bazy danych związane ze szkolnictwem wyższym.

Poziomy według RMM

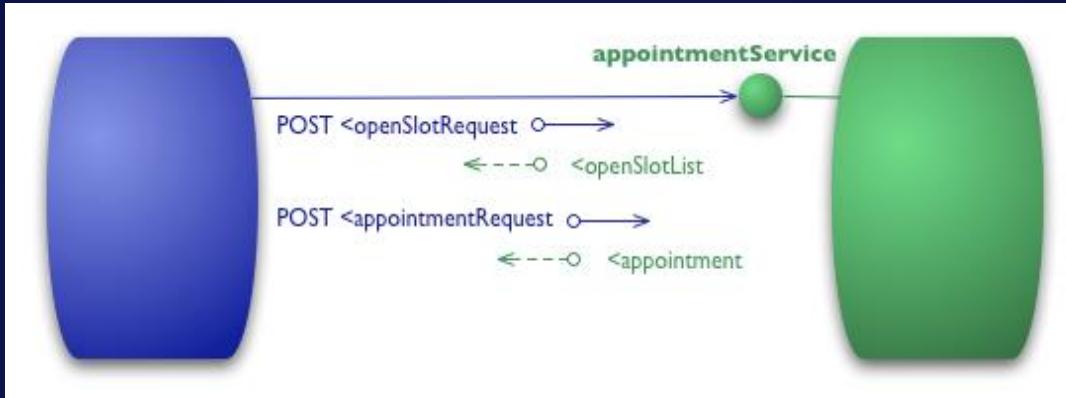
Level 3: Hypermedia Controls

Level 2: HTTP Verbs

Level 1: Resources

Level 0: The Swamp of POX

Poziom 0



POST /appointmentService HTTP/1.1

[various other headers]

```
<openSlotRequest date = "2010-01-04" doctor = "mjones"/>
```

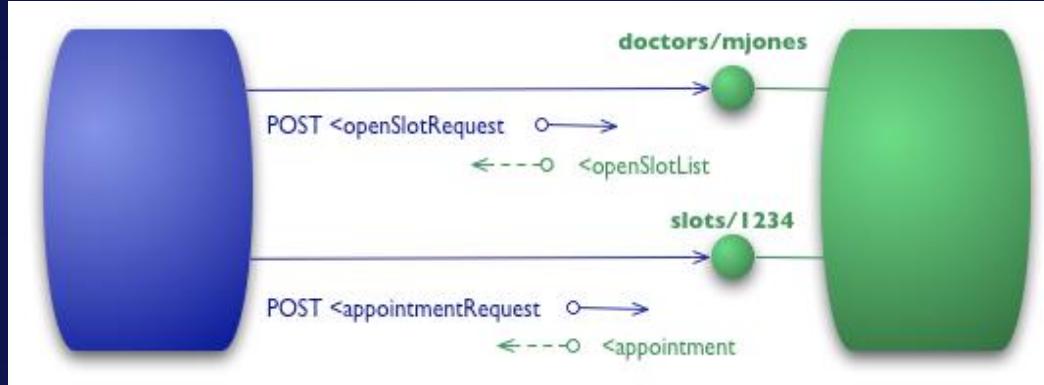
```
HTTP/1.1 200 OK
[various headers]

<openSlotList>
    <slot start = "1400" end = "1450">
        <doctor id = "mjones"/>
    </slot>
    <slot start = "1600" end = "1650">
        <doctor id = "mjones"/>
    </slot>
</openSlotList>
```

```
POST /appointmentService HTTP/1.1
[various other headers]

<appointmentRequest>
    <slot doctor = "mjones" start = "1400" end = "1450"/>
    <patient id = "jsmith"/>
</appointmentRequest>
```

Poziom 1



POST /doctors/mjones HTTP/1.1
[various other headers]

<openSlotRequest date = "2010-01-04"/>

HTTP/1.1 200 OK
[various headers]

<openSlotList>
 <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
 <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>
</openSlotList>

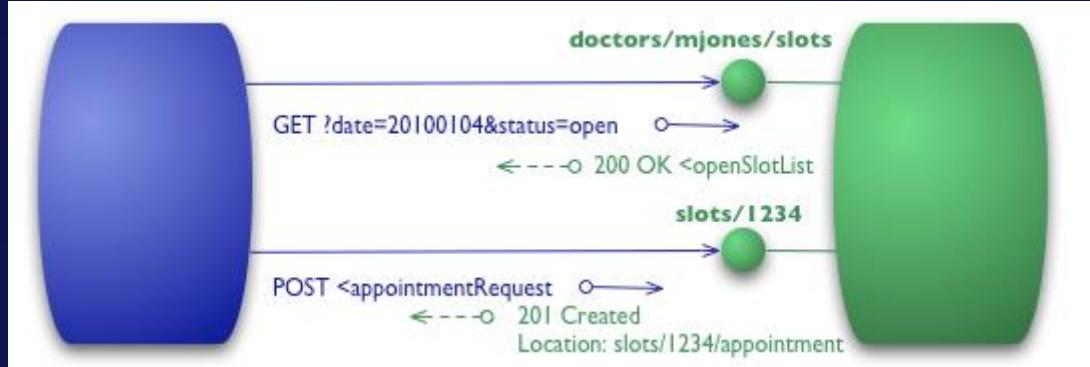
```
POST /slots/1234 HTTP/1.1  
[various other headers]
```

```
<appointmentRequest>  
  <patient id = "jsmith"/>  
</appointmentRequest>
```

```
HTTP/1.1 200 OK  
[various headers]
```

```
<appointment>  
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>  
  <patient id = "jsmith"/>  
</appointment>
```

Poziom 2



```
GET /doctors/mjones/slots?date=20100104&status=open HTTP/1.1
Host: royalhope.nhs.uk
```

```
HTTP/1.1 200 OK
[various headers]
```

```
<openSlotList>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>
</openSlotList>
```

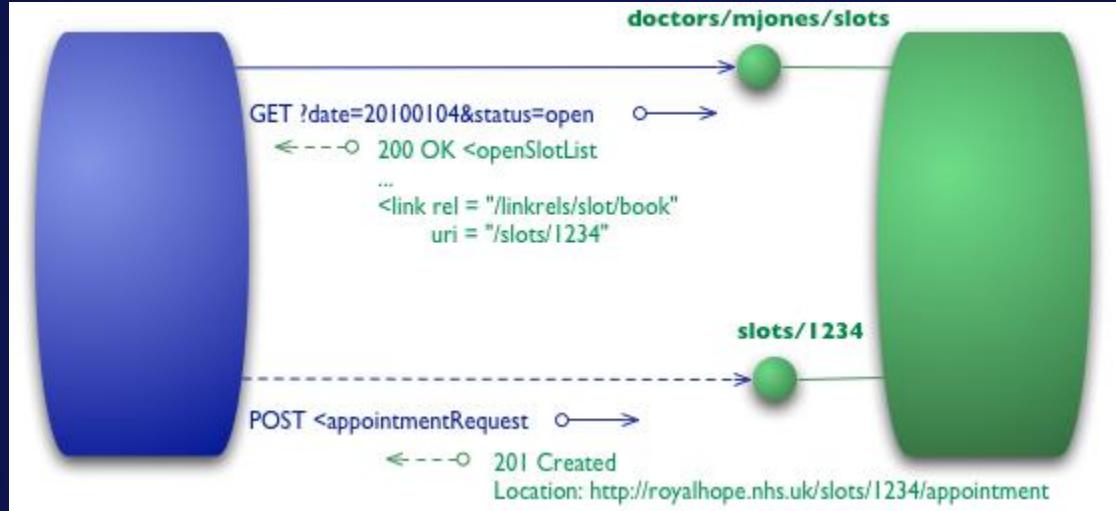
```
POST /slots/1234 HTTP/1.1  
[various other headers]  
  
<appointmentRequest>  
  <patient id = "jsmith"/>  
</appointmentRequest>
```

```
HTTP/1.1 201 Created  
Location: slots/1234/appointment  
[various headers]  
  
<appointment>  
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>  
  <patient id = "jsmith"/>  
</appointment>
```

```
HTTP/1.1 409 Conflict  
[various headers]  
  
<openSlotList>  
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>  
</openSlotList>
```

Grupa kodów	Znaczenie	Objaśnienie
1xx	Informacyjne	Komunikaty transferu na poziomie protokołu.
2xx	Powodzenie operacji	Wskazuje na powodzenie realizacji żądania klienta
3xx	Przekierowanie	Informuje, że do ukończenia żądania klient musi podjąć dodatkową akcję klienta
4xx	Błąd klienta	Wskazanie błędu leżącego po stronie klienta
5xx	Błąd serwera	Wskazanie błędu, za który odpowiada serwer.

Poziom 3



```
GET /doctors/mjones/slots?date=20100104&status=open HTTP/1.1
Host: royalhope.nhs.uk
```

```
HTTP/1.1 200 OK
[various headers]

<openSlotList>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450">
    <link rel = "/linkrels/slot/book"
          uri = "/slots/1234"/>
  </slot>
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650">
    <link rel = "/linkrels/slot/book"
          uri = "/slots/5678"/>
  </slot>
</openSlotList>
```

```
POST /slots/1234 HTTP/1.1
[various other headers]

<appointmentRequest>
  <patient id = "jsmith"/>
</appointmentRequest>
```

```
HTTP/1.1 201 Created
Location: http://royalhope.nhs.uk/slots/1234/appointment
[various headers]

<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
  <link rel = "/linkrels/appointment/cancel"
        uri = "/slots/1234/appointment"/>
  <link rel = "/linkrels/appointment/addTest"
        uri = "/slots/1234/appointment/tests"/>
  <link rel = "self"
        uri = "/slots/1234/appointment"/>
  <link rel = "/linkrels/appointment/changeTime"
        uri = "/doctors/mjones/slots?date=20100104&status=open"/>
  <link rel = "/linkrels/appointment/updateContactInfo"
        uri = "/patients/jsmith/contactInfo"/>
  <link rel = "/linkrels/help"
        uri = "/help/appointment"/>
</appointment>
```

Jak jest w systemie POL-on?

Założenia dotyczące operacji:

- POST - dodanie zasobu (utworzenie np. nowy student lub nowa zmiana biznesowa np. nowe nazwisko);
- PUT - aktualizacja, korekta;
- GET - pobranie zasobu;
- DELETE - usunięcie zasobu.

```
▼ 1:  
  uid:      "bsNFHs1eryegv3aRK0j5deg"  
  name:     "Warszawska Szkoła Wyższa z siedzibą w Otwocku"  
  status:   "OPERATING"  
  
▼ 2:  
  uid:      "b5IYHVFxXb6fSBPLS-jBg1Q"  
  ▼ name:    "Wyższa Szkoła Języków Obcych w Szczecinie w likwidacji z siedzibą w Szczecinie"  
  status:   "OPERATING"  
  
▼ 3:
```

HTTP Status 400 – Bad Request

Type Status Report

Description The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).

```
▼ results:  
  ▼ 0:  
    errorCode:  "2200"  
    message:    "Zły login lub hasło"
```

```
{  
  "ruleskis": [  
    {  
      "key": "POL_1018",  
      "content": "Pole \"PESEL\" ma niepoprawny format",  
      "invalidValue": "13sssdasa6"  
    }  
  ]  
}
```

Podsumowanie

Poziom 0	Poziom 1	Poziom 2	Poziom 3
			

Bibliografia

1. <https://devkr.pl/2018/04/10/restful-api-richardson-maturity-model/> [dostęp 13.11.2023]
2. <https://martinfowler.com/articles/richardsonMaturityModel.html> [dostęp 13.11.2023]
3. <https://polon.nauka.gov.pl/pomoc/wp-content/uploads/2020/03/POL-on-Specyfikacja-API-REST-w-systemie-POL-on-wersja-1.0.1.pdf> [dostęp 13.11.2023]
4. https://polon.nauka.gov.pl/pomoc/wp-content/uploads/2019/11/konsultacje_masowych_interfejsow_pol-on_iii_tura_studenci.pdf [dostęp 13.11.2023]
5. <https://polon.nauka.gov.pl/api-ws/api/api-docs?url=/api-ws/api/swagger.json#tag/common> [dostęp 13.11.2023]

HTTP HTTPS SSL

Michał Fila ETI WIMIP



•
•
•

Czym jest HTTP

Jest protokołem przesyłania danych między klientem a serwerem sieciowym.

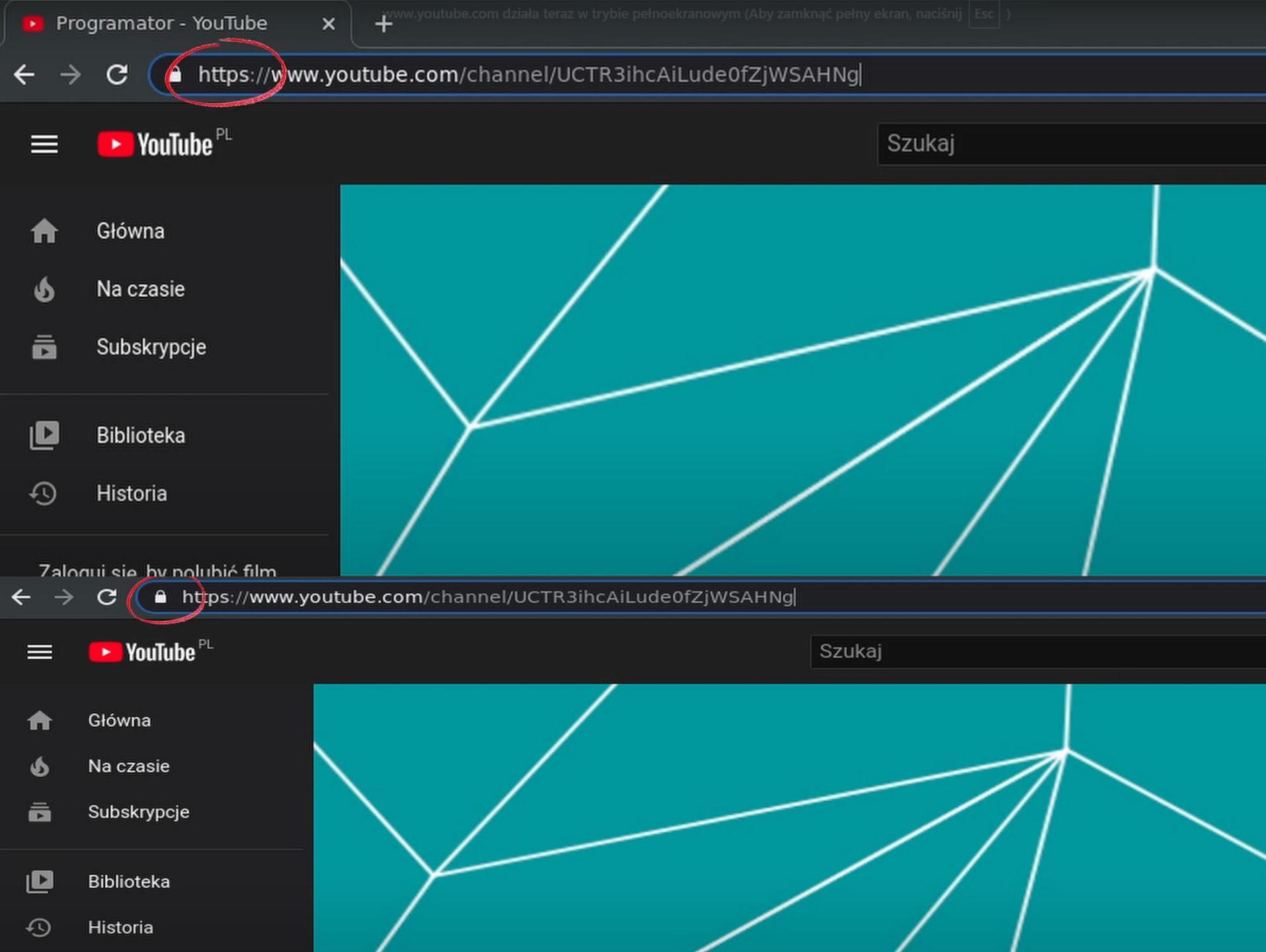
Jest protokołem tekstowym czyli wszystkie przesyłane informacje są tekstem

•
•
•

HTTPS

Obecnie HTTP jest wypierany ma rzecz
HTTPS

HTTPS działa tak samo tylko dane są
szyfrowane przed wysłaniem



-
-
-

Porównanie Protokołów

HTTP

domyślnie korzysta z portu 80.

Adres strony www rozpoczyna http://

Przesyłane informacje nie są szyfrowane - loginy, hasła itp. Są łatwe do odczytania.

HTTPS

Domyślnie korzysta z portu 443.

Strona internetowa poprzedzona jest przedrostkiem https://

Poufne dane przesyłane między klientem a serwerem są szyfrowane poprzez protokół SSL / TLS.

Dane są zabezpieczone przed przechwyceniem, odczytaniem i modyfikacją przez osoby trzecie.

-
-
-
-
-
-
-
-

Jak to działa ?

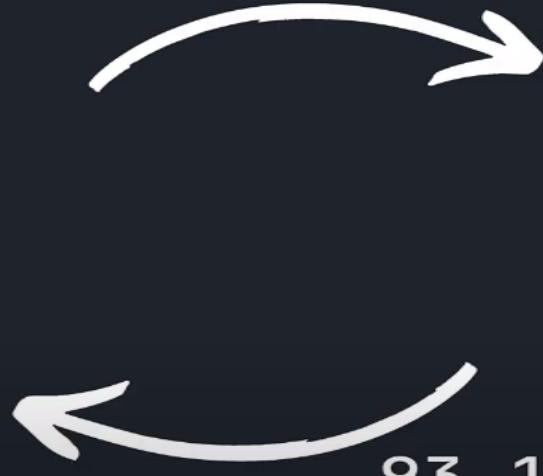
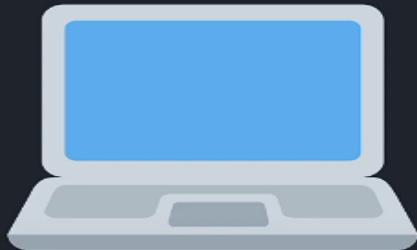




`example.com/page.html`

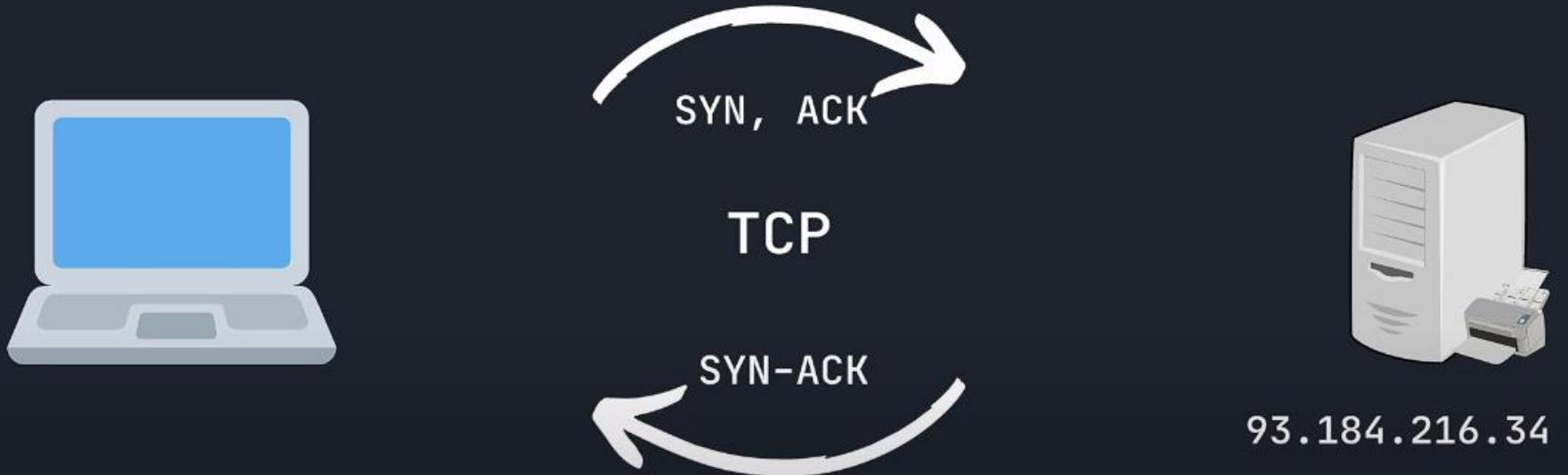
Kiedy wpiszemy w przeglądarkę adres jakiejś strony internetowej co zrobi przeglądarka ?

example.com ?



DNS

Po pierwsze ustalić adres IP serwera
obsługującego daną stronę za pomocą
systemu DNS który dla danej domeny zwróci
nam adres IP



93.184.216.34

Następnie przeglądarka otworzy połączenie
TCP z tym serwerem

(I TU SIĘ ZACZYNA ROLA HTTP)



```
GET /page.html HTTP/1.1  
Host: example.com  
Accept: text/html  
Cookie: user=12345;
```



Przeglądarka wysyła żądanie do serwera a serwer odpowiada na żądanie z treścią strony

```
HTTP/1.1 200 OK  
Content-Type: text/html  
Content-Length: 1453
```



```
<html>  
...  
</html>
```



Struktura zapytania

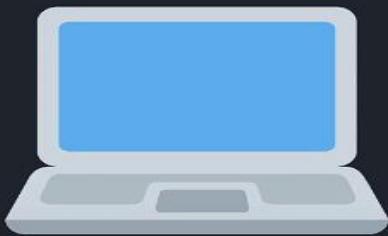




GET /page.html HTTP/1.1
Host: example.com
Accept: text/html
Cookie: user=12345;



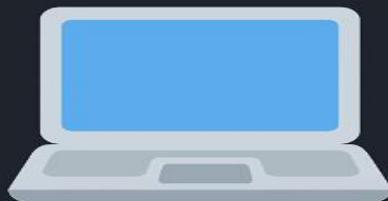
Zapytanie składa się z metody w naszym przypadku Get



```
GET /page.html HTTP/1.1  
Host: example.com  
Accept: text/html  
Cookie: user=12345;
```



Następnie podajemy ścieżkę do pliku który chcemy otrzymać



GET /page.html HTTP/1.1
Host: example.com
Accept: text/html
Cookie: user=12345;



Na końcu podajemy wersję protokołu



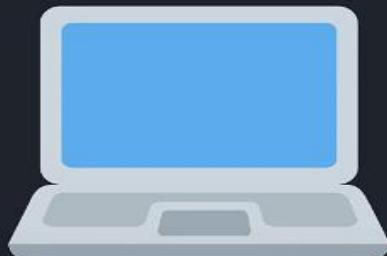
Po pierwszej linijce następują nagłówki zapytania



GET /page.html HTTP/1.1
Host: example.com
Accept: text/html
Cookie: user=12345;



Wskazuje domenę dla której chcemy pobrać dany plik stosujemy go gdyż serwer może hostować wiele stron na raz



```
GET /page.html HTTP/1.1  
Host: example.com  
Accept: text/html  
Cookie: user=12345;
```

Tutaj podajemy jaki format odpowiedzi
chcielibyśmy uzyskać tutaj HTML



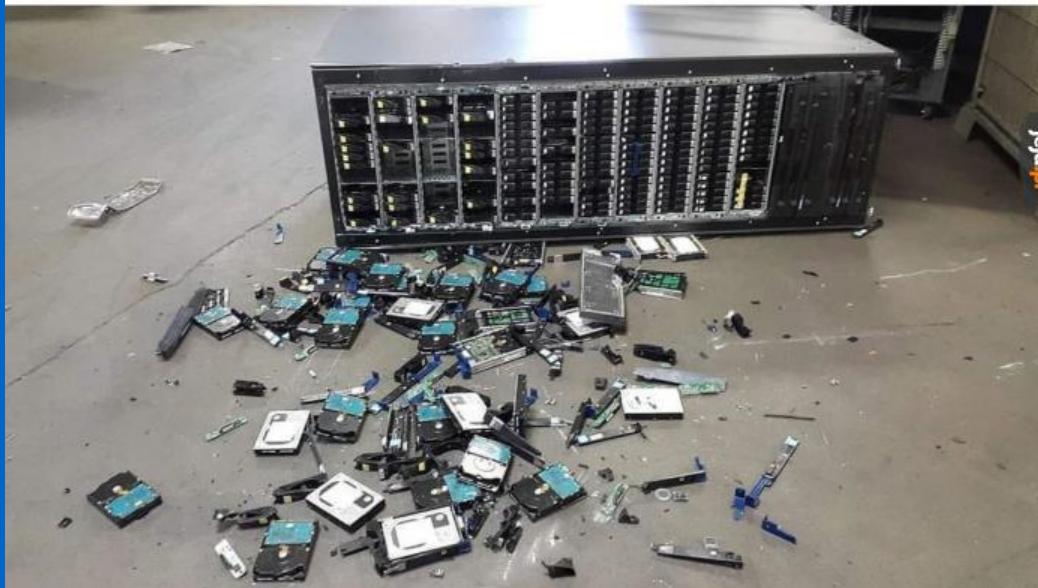
GET /page.html HTTP/1.1
Host: example.com
~~Accept: text/html~~
Cookie: user=12345;



Przeglądarka wysyła zawartość pliku cookies zapisanych dla danej strony

A co serwer na to – Struktura odpowiedzi

- Szefie, serwer padł!
- No to go zrestartujcie
- Emmm..., to nie będzie takie proste.



HTTP/1.1 200 OK

Content-Type: text/html

Content-Length: 1453



<html>

...

</html>



•
•
•

Co to jest SSL ?

To certyfikat który zapewnia szyfrowane połączenie pomiędzy klientem i serwerem. Dzięki niemu wysyłane dane są zabezpieczone



The screenshot shows a web browser window with a YouTube channel page loaded. At the top, the address bar displays a secure connection URL: <https://www.youtube.com/channel/UCTR3ihcAiLude0fZjWSAHNg>. The URL is circled in red. Below the address bar is the YouTube logo and a search bar labeled "Szukaj". On the left side of the screen is a sidebar with five items: "Główna", "Na czasie", "Subskrypcje", "Biblioteka", and "Historia", each with a corresponding icon. The main content area of the browser shows a network diagram consisting of several white lines forming a complex polygonal shape against a teal background.

Jeżeli koło adresu strony znajduje się zamknięta kłódka to znaczy że strona korzysta z certyfikatu i dane wysyłane pomiędzy przeglądarką a serwerem są zaszyfrowane

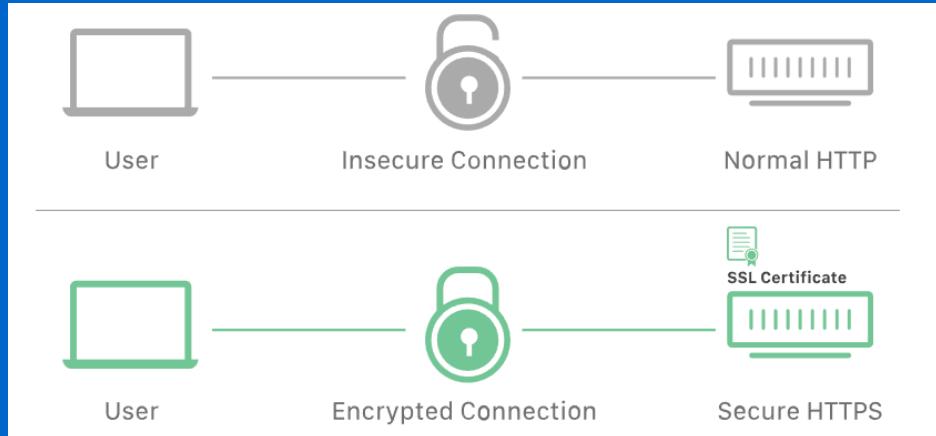
-
-
-

Jak działa SSL

Kiedy użytkownik próbuje połączyć się z witryną z certyfikatem SSL, następuje wymiana kluczy między przeglądarką a serwerem. Ta wymiana kluczy odbywa się z wykorzystaniem asymetrycznego szyfrowania. Klucz publiczny serwera jest wykorzystywany do zaszyfrowania danych, które są przesyłane do serwera.

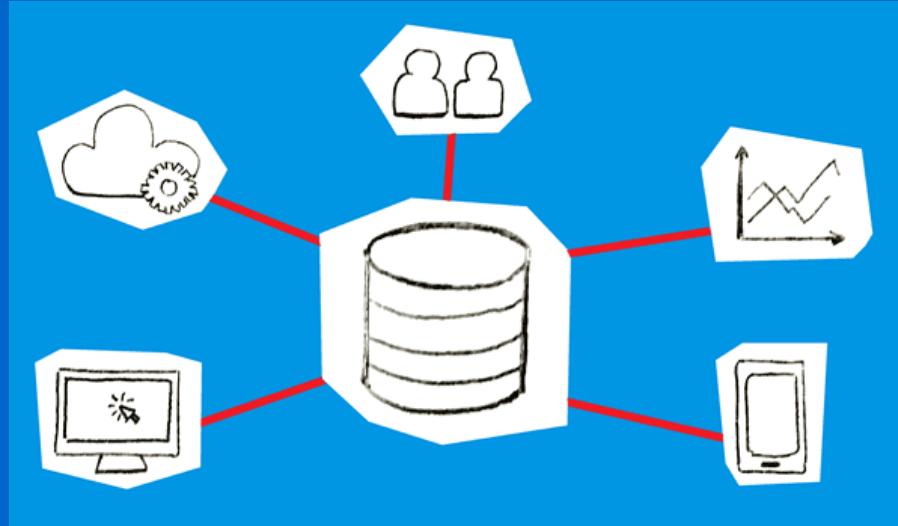


Jak działa SSL



Po nawiązaniu połączenia szyfrowanego dane są przesyłane między przeglądarką a serwerem w formie, którą tylko klucz prywatny serwera jest w stanie odszyfrować.

Integralność Danych



Certyfikat SSL gwarantuje integralność danych poprzez umożliwienie odbiorcy zweryfikowania, czy dane nie zostały zmienione ani naruszone podczas transmisji.

-
-
-

Źródła

Doświadczenie wyniesione z technikum

https://pl.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Metody_HTTP

<https://pasja-informatyki.pl/sieci-komputerowe/protokol-http/>

<https://forum.pasja-informatyki.pl/411959/certyfikat-ssl>

<https://www.elektroda.pl/rtvforum/topic2939212.html#14235933>

Żądanie http, metody protokołu http, nagłówki http, statusy http

Szymon Skubis ETI rok III



Krótką definicja HTTP (Hypertext Transfer Protocol)

HTTP, czyli Hypertext Transfer Protocol, to protokół komunikacyjny używany w World Wide Web (WWW) do przesyłania danych między klientem a serwerem. Jest to podstawowy protokół stosowany w architekturze klient-serwer, gdzie klient to aplikacja lub przeglądarka internetowa, a serwer to komputer przechowujący zasoby, takie jak strony internetowe, pliki czy inne dane.





Rola HTTP w komunikacji internetowej.

1. Przesyłanie Dokumentów Hipertekstowych:

Główną funkcją HTTP jest umożliwienie przesyłania dokumentów hipertekstowych, czyli stron internetowych, między serwerem a klientem. Te dokumenty mogą zawierać tekst, obrazy, linki i inne elementy, które tworzą interaktywne strony internetowe.

1. Model Żądanie-Odpowiedź:

HTTP działa na zasadzie modelu żądanie-odpowiedź, gdzie klient wysyła żądanie do serwera, a serwer odpowiada dostarczając odpowiednią treść. To umożliwia dynamiczne odświeżanie zawartości stron internetowych i interakcję użytkownika z serwerem.

1. Bezstanowość:

HTTP jest bezstanowy, co oznacza, że każde żądanie od klienta jest traktowane niezależnie, a serwer nie przechowuje informacji o poprzednich żądaniach klienta. To zwiększa skalowalność systemów internetowych.

1. Obsługa Różnych Mediów:

HTTP umożliwia przesyłanie różnego rodzaju danych, nie tylko tekstów HTML. Dzięki nagłówkom, które można dostosować, można przesyłać obrazy, pliki, multimedia, a także dane w formacie JSON, XML i inne.



Rola HTTP w komunikacji internetowej.

5. Podstawy Autentykacji i Bezpieczeństwo:

HTTP obsługuje podstawowe mechanizmy autentykacji, które pozwalają serwerom kontrolować dostęp do zasobów. W przypadku bezpiecznej wersji protokołu, HTTPS, zapewniane jest szyfrowanie danych, co zabezpiecza je przed nieautoryzowanym dostępem.

6. Obsługa Sesyjności:

Choć HTTP sam w sobie jest bezstanowy, często jest używany w połączeniu z mechanizmami sesji, takimi jak ciasteczka (cookies) czy nagłówki sesji. To pozwala na przechowywanie informacji o stanie klienta między różnymi żądaniami.

7. Obsługa Wersji:

HTTP umożliwia negocjację wersji protokołu między klientem a serwerem, co pozwala na ewolucję i aktualizację samego protokołu.



Historia protokołu HTTP

1. HTTP/0.9 (1991):

Pierwsza wersja wprowadzona przez Tima Berners-Lee.

Prosty protokół obsługujący tylko metodę GET i bazujący na tekście ASCII.

2. HTTP/1.0 (1996):

Dodano obsługę nagłówków, umożliwiając przesyłanie dodatkowych informacji.

Wprowadzenie nowych metod, takich jak POST.

3. HTTP/1.1 (1997):

Staje się najbardziej powszechnie stosowaną wersją.

Zawiera szereg usprawnień, takich jak persistent connections, gzip dla kompresji danych i obsługa byte-ranges.



Historia protokołu HTTP

4. HTTP/2 (2015):

Wprowadza innowacje, takie jak multiplexing, poprawiające wydajność.

Efektywniejsza obsługa kompresji i priorytetyzacji zasobów.

5. HTTP/3 (2018):

Eksperymentalna wersja z zastosowaniem protokołu QUIC.

Dodaje szyfrowaną warstwę transportową i wprowadza zmiany w architekturze, takie jak STREAMS.



Główne metody



Metoda GET

Znaczenie: Metoda GET służy do pobierania danych z serwera.

Zastosowanie: Wykorzystywana, gdy klient chce otrzymać dane zasobu. Przykłady to pobieranie stron internetowych, obrazów czy innych zasobów. Parametry są przekazywane w adresie URL.



Metoda POST

Znaczenie: Metoda POST służy do wysyłania danych do serwera w celu przetworzenia.

Zastosowanie: Wykorzystywana, gdy klient chce przesłać dane, takie jak formularze. Parametry są przesyłane w ciele żądania, co pozwala na przesyłanie większych ilości danych niż w przypadku GET.



Metoda PUT

Znaczenie: Metoda PUT służy do aktualizacji danych zasobu lub tworzenia nowego zasobu, jeśli nie istnieje.

Zastosowanie: Stosowana, gdy klient chce zmodyfikować istniejący zasób lub utworzyć nowy. Wartości przesyłane są w ciele żądania, a identyfikator zasobu jest umieszczany w adresie URL.



Metoda DELETE

Znaczenie: Metoda DELETE służy do usuwania zasobu na serwerze.

Zastosowanie: Wykorzystywana, gdy klient chce usunąć istniejący zasób. Identyfikator zasobu jest przekazywany w adresie URL, a żądanie skutkuje usunięciem tego zasobu.



Składnia i struktura URL (Uniform Resource Locator):

Składnia: **scheme://host:port/path?query#fragment**

Elementy:

scheme: Protokół komunikacyjny (np. http, https).

host: Adres serwera.

port: Numer portu (opcjonalny, domyślnie 80 dla HTTP, 443 dla HTTPS).

path: Ścieżka do zasobu na serwerze.

query: Parametry przekazywane w formie klucz-wartość.

fragment: Określa konkretną sekcję dokumentu.

Przykład:

`https://www.example.com:8080/path/to/resource?query=param#section`



Kluczowe nagłówki w żądaniach HTTP:



Host

Znaczenie: Określa nazwę hosta serwera, do którego skierowane jest żądanie.

Przykład: **Host: www.example.com**

Ważność: Konieczny dla serwera wirtualnego, gdy na jednym serwerze hostuje się wiele stron internetowych.



User-Agent

Znaczenie: Informuje serwer o rodzaju i wersji przeglądarki lub aplikacji klienta.

Przykład: **User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0)
Gecko/20100101 Firefox/70.0**

Ważność: Pomaga serwerowi dostosować odpowiedź do rodzaju i wersji klienta.



Accept

Znaczenie: Określa typy mediów (MIME types), które klient jest w stanie obsłużyć.

Przykład: **Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8**

Ważność: Pomaga serwerowi dostarczyć odpowiednią wersję zasobu zgodnie z preferencjami klienta.



Authorization

Znaczenie: Przesyła dane uwierzytelniające, zazwyczaj w formie tokena dostępowego.

Przykład: **Authorization: Bearer abc123xyz**

Ważność: Konieczny do autoryzacji, na przykład przy dostępie do zasobów chronionych hasłem.



Statusy HTTP



Czym jest status HTTP?

Status HTTP to numer identyfikujący stan odpowiedzi serwera na żądanie klienta w protokole HTTP (Hypertext Transfer Protocol). Jest to trzycyfrowy kod numeryczny, który jest częścią linii statusu w nagłówku odpowiedzi HTTP. Linia statusu zawiera wersję protokołu oraz status odpowiedzi, co pozwala klientowi zrozumieć, czy jego żądanie zostało pomyślnie zrealizowane, czy też wystąpił jakiś błąd.



Pięć kategorii statusu HTTP

1xx (Informacyjne): Otrzymano informację, kontynuacja żądania.

2xx (Sukces): Żądanie zostało pomyślnie zrealizowane.

3xx (Przekierowanie): Żądanie wymaga dodatkowych działań w celu ukończenia.

4xx (Błąd klienta): Żądanie zawierało błąd lub nie można go zrealizować.

5xx (Błąd serwera): Serwer napotkał błąd, uniemożliwiający zrealizowanie żądania.



Popularne kody statusu

200 OK: Sukces. Oznacza, że żądanie zostało pomyślnie zrealizowane.

201 Created: Sukces. Nowy zasób został utworzony.

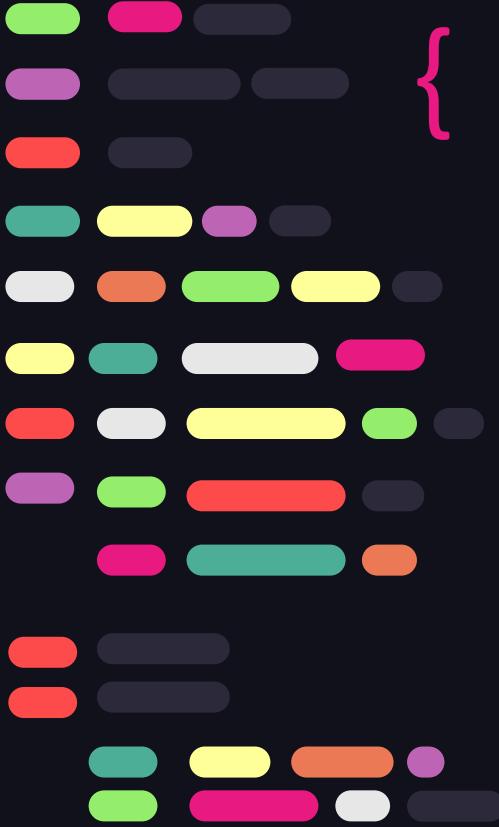
204 No Content: Sukces. Nie ma treści do przesłania w odpowiedzi.

400 Bad Request: Błąd klienta. Żądanie było niepoprawne.

401 Unauthorized: Błąd klienta. Wymagane uwierzytelnienie.

404 Not Found: Błąd klienta. Zasób nie został odnaleziony.

500 Internal Server Error: Błąd serwera. Ogólny błąd serwera.



{ Analiza strony AGH
pod kątem metody
żądania, statusu
odpowiedzi i
nagłówku żądania

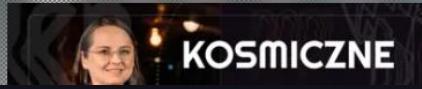
...

}



Hala sportowa AGH – obiekt marzeń dla sportowców i drużyn

AGH oficjalnie otworzyła wielofunkcyjną halę sportową. Nowy obiekt na mapie Krakowa służy m.in. studentom zrzeszonym w sekcjach sportowych AZS AGH, w tym pierwszoligowym zespole siatkarzy, koszykarzy i piłkarzy ręcznych.



Engirav

Filter	Invert	Hide data URLs	Hide extension URLs	All	Fetch/XHR	JS	CSS	Img	Media	Font	Doc	WS	Wasm	Manifest	Other	Blocked response cookies	Blocked requests	3rd-party requests									
				20000 ms	40000 ms	60000 ms	80000 ms	100000 ms	120000 ms	140000 ms	160000 ms	180000 ms	200000 ms	220000 ms	240000 ms	260000 ms	280000 ms	300000 ms	320000 ms	340000 ms	360000 ms	380000 ms	400000 ms	420000 ms	440000 ms	460000 ms	480000 ms

Name	Status	Type	Initiator	Size	Time	Waterfall
www.agh.edu.pl	200	document	Other	14.3 kB	108 ms	
merged-1c08c701b09f6bc7a8644f282c941fbcd-1253582cdec057d910883174a183ceed.css.gzipped?1700...	200	stylesheet	(index):14		(memory cache)	0 ms
main.css	200	stylesheet	(index):33		(memory cache)	0 ms
Logo.svg	200	svg+xml	(index):53		(memory cache)	0 ms
csm_Otwarcie_Hali_Sportowej_KBochenek_budynek_48b3081dbf.webp	200	webp	(index):199		(memory cache)	0 ms
csm_dni_prof_hoborskiego_agh_e414f4436f.webp	200	webp	(index):216		(memory cache)	0 ms
csm_Barka_a7ff0a7bff.webp	200	webp	(index):258		(memory cache)	0 ms
csm_engi_rank_z_tiem_cf54c91ef1.webp	200	webp	(index):266		(memory cache)	0 ms
csm_budynek_wydzialu_fizyki_i_informatyki_stosowanej_agh_4f4f8e824d.webp	200	webp	(index):274		(memory cache)	0 ms
csm_superkomputery_dreamtime_8487230aa3.webp	200	webp	(index):282		(memory cache)	0 ms
csm_Cyber_Kampus_eac68692d8.webp	200	webp	(index):305		(memory cache)	0 ms
csm_naukowiec_mikroskop_elektronowy_4d7b8bf044.jpg	200	jpeg	(index):418		(memory cache)	0 ms
merged-1ac7bb48893ab12683cf8e7ae9c556e-38f73c9ae18281a2de3b29679a131d7e.js.gzipped?17001...	200	script	(index):567		(memory cache)	0 ms
agh_sg_bg.js?1700140550	200	script	(index):568		(memory cache)	0 ms
isolated-first.js	200	fetch	snippets.js:141	0 B	57 ms	
css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,5...;1,100;1,300;1,400;1,500;1,700;1,900&display=...	200	stylesheet	merged-1c08c70...-1253582....css.gzipped?170014112...		(memory cache)	0 ms
gajs	307	script / Redirect	(index):44	0 B	68 ms	
KFOICnqEu92Fr1MmSU5fChc4EsA.woff2	200	font	css2		(memory cache)	0 ms
KFOICnqEu92Fr1MmSU5fBc4.woff2	200	font	css2		(memory cache)	0 ms
KFOICnqEu92Fr1Mu7GxKoZy.woff2	200	font	css2		(memory cache)	0 ms
KFOICnqEu92Fr1Mu4mxK.woff2	200	font	css2		(memory cache)	0 ms
KFOICnqEu92Fr1MmEU9fChc4EsA.woff2	200	font	css2		(memory cache)	0 ms
KFOICnqEu92Fr1MmEU9fBc4.woff2	200	font	css2		(memory cache)	0 ms
KFOICnqEu92Fr1MmWUifChc4EsA.woff2	200	font	css2		(memory cache)	0 ms
KFOICnqEu92Fr1MmWUifBc4.woff2	200	font	css2		(memory cache)	0 ms
KFOICnqEu92Fr1MmYUtfChc4EsA.woff2	200	font	css2		(memory cache)	0 ms
KFOICnqEu92Fr1MmYUtfBc4.woff2	200	font	css2		(memory cache)	0 ms
icons.ttf?k2isih	200	font	merged-1c08c70...-1253582....css.gzipped		(memory cache)	0 ms
csm_Otwarcie_Hali_Sportowej_KBochenek_budynek_b8a6acb8f8.jpg	200	jpeg	merged-1c08c70...-1253582....css.gzipped		(memory cache)	0 ms
csm_dni_prof_hoborskiego_agh_f365558de1.jpg	200	jpeg	merged-1c08c70...-1253582....css.gzipped		(memory cache)	0 ms
google-analytics_gajs?secret=xzqf2j	200	script	gajs	4.8 kB	69 ms	
dekor_1_pasek.svg	200	svg+xml	main.css		(memory cache)	0 ms
dekor_1.svg	200	svg+xml	main.css		(memory cache)	0 ms
gradient_pod_rorator_2.png	200	png	merged-1c08c70...-1253582....css.gzipped		(memory cache)	0 ms

Filter		Invert	Hide data URLs	Hide extension URLs	All	Fetch/XHR	JS	CSS	Img	Media	Font	Doc	WS	Wasm	Manifest	Other	Blocked response cookies	Blocked requests	3rd-party requests						
20000 ms	40000 ms	60000 ms	80000 ms	100000 ms	120000 ms	140000 ms	160000 ms	180000 ms	200000 ms	220000 ms	240000 ms	260000 ms	280000 ms	300000 ms	320000 ms	340000 ms	360000 ms	380000 ms	400000 ms	420000 ms	440000 ms	460000 ms	480000 ms		
www.agh.edu.pl																									
merged-1c08c701b09f6bc7a8644f282c941fbdb-1253582cdec057d910883174a183ceed.css.gzp?1700...	200	document	Other															14.3 kB		108 ms					
main.css	200	stylesheet	(index):14															(memory cache)		0 ms					
Logo.svg	200	stylesheet	(index):33															(memory cache)		0 ms					
csm_Otwarcie_Hali_Sportowej_KBochenek_budynek_48b3081dbf.webp	200	vg+xml	(index):53															(memory cache)		0 ms					
csm_dni_prof_hoborskiego_agh_e414f4436f.webp	200	webp	(index):199															(memory cache)		0 ms					
csm_Barka_a7ff0a7bf.webp	200	webp	(index):216															(memory cache)		0 ms					
csm_engi_rank_z_tiem_cf54c91ef1.webp	200	webp	(index):258															(memory cache)		0 ms					
csm_budynek_wydzialu_fizyki_i_informatyki_stosowanej_agh_4f4f8e824d.webp	200	webp	(index):266															(memory cache)		0 ms					
csm_superkomputery_dreamtime_8487230aa3.webp	200	webp	(index):274															(memory cache)		0 ms					
csm_Cyber_Kampus_eac68692d8.webp	200	webp	(index):282															(memory cache)		0 ms					
csm_naukowiec_mikroskop_elektronowy_4d7b8bf044.jpg	200	webp	(index):305															(memory cache)		0 ms					
merged-1a7bb48893ab12683cf8e7ae9c556e-38f73c9ae18281a2de3b29679a131d7e.js.gzp?17001...	200	script	(index):567															(memory cache)		0 ms					
agh_sg_bg.js?1700140550	200	script	(index):568															(memory cache)		0 ms					
isolated-first.js	200	fetch	snippets.js:141															0 kB		57 ms					
css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,5...;1,100;1,300;1,400;1,500;1,700;1,900&display=...	200	stylesheet	merged-1c08c70...-1253582....css.gzp?170014112...															(memory cache)		0 ms					
gajs	307	script / Redirect	(index):44															0 kB		68 ms					
KFOICnqEu92Fr1MmSU5fChc4EsA.woff2	200	font	css2															(memory cache)		0 ms					
KFOICnqEu92Fr1MmSU5fbC4.woff2	200	font	css2															(memory cache)		0 ms					
KFOICnqEu92Fr1Mu7GxKOzY.woff2	200	font	css2															(memory cache)		0 ms					
KFOICnqEu92Fr1Mu4mxK.woff2	200	font	css2															(memory cache)		0 ms					
KFOICnqEu92Fr1MmEU9fChc4EsA.woff2	200	font	css2															(memory cache)		0 ms					
KFOICnqEu92Fr1MmEU9fB8c4.woff2	200	font	css2															(memory cache)		0 ms					
KFOICnqEu92Fr1MmWUifChc4EsA.woff2	200	font	css2															(memory cache)		0 ms					
KFOICnqEu92Fr1MmWUifB8c4.woff2	200	font	css2															(memory cache)		0 ms					
KFOICnqEu92Fr1MmYUtfChc4EsA.woff2	200	font	css2															(memory cache)		0 ms					
KFOICnqEu92Fr1MmYUtfB8c4.woff2	200	font	css2															(memory cache)		0 ms					
icons.ttf?k2isih	200	font	merged-1c08c70...-1253582....css.gzp?170014112...															(memory cache)		0 ms					
csm_Otwarcie_Hali_Sportowej_KBochenek_budynek_b8a6acb8f8.jpg	200	peg	merged-1c08c70...-1253582....css.gzp?170014112...														(memory cache)		0 ms						
csm_dni_prof_hoborskiego_agh_f365558de1.jpg	200	peg	merged-1c08c70...-1253582....css.gzp?170014112...														(memory cache)		0 ms						
google-analytics_gajs?secret=xzqf2j	200	script	gajs															4.8 kB		69 ms					
dekor_1_pasek.svg	200	vg+xml	main.css															(memory cache)		0 ms					
dekor_1.svg	200	vg+xml	main.css															(memory cache)		0 ms					
gradient_pod_rorator_2.png	200	png	merged-1c08c70...-1253582....css.gzp?170014112...															(memory cache)		0 ms					

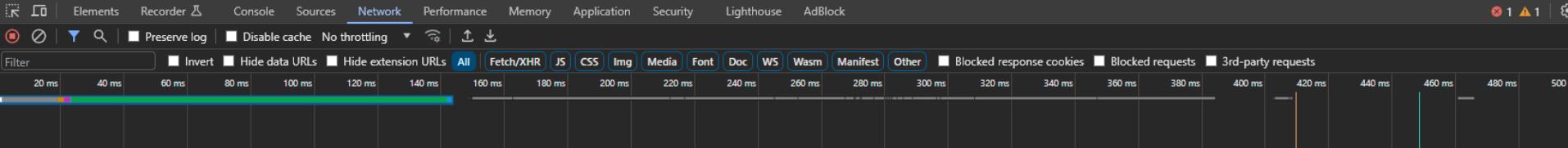
○ ⊖ | ⚡ 🔎 | ■ Preserve log | ■ Disable cache | No throttling ▾ ⚙

Filter	Invert	Hide data URLs	Hide extension URLs	All	Fetch/XHR	JS	CSS	Img	Media	Font	Doc	WS	Wasm	Manifest	Other	Blocked response cookies	Blocked requests	3rd-party requests		
Name															Status	Type	Initiator	Size	Time	Waterfall
www.agh.edu.pl															200	doc				
merged-1c08c701b09fbc7a8644f282c941fb-1253582cdec057d910883174a183ceed.css.gzip?1700..															200	sty				
main.css															200	sty				
Logo.svg															200	wg				
csm_Otwarcie_Hali_Sportowej_KBochenek_budynek_48b3081dbf.webp															200	wet				
csm_dni_prof_hoborskiego_agh_e414f4436f.webp															200	wet				
csm_Barka_a7ff0a7bff.webp															200	wet				
csm_engi_rank_z_tle_m_cf54c91ef1.webp															200	wet				
csm_budynek_wydziału_fizyki_i_informatyki_stosowanej_agh_4f4f8e824d.webp															200	wet				
csm_superkomputery_dreamtime_8487230aa3.webp															200	wet				
csm_Cyber_Kampus_eac68692d8.webp															200	wet				
csm_naukowiec_mikroskop_elektronowy_4d7b8bf044.jpg															200	peç				
merged-1ac7bb488938ab12683cf8e7ae9c556e-38f73c9ae18281a2de3b29679a131d7e.js.gzip?17001..															200	scrij				
agh_sq_bgjs?1700140550															200	scrij				
isolated-firstjs															200	etc				
css2?family=Robotoitalic,wght@0,100;0,300;0,400;0,5...;1,100;1,300;1,400;1,500;1,700;1,900&display=...															200	sty				
ga.js															307	scrij				
KFOICnqEu92Fr1MmSU5fChc4EsA.woff2															200	font				
KFOICnqEu92Fr1MmSU5fBc4.woff2															200	font				
KFOICnqEu92Fr1Mu7GxKOzY.woff2															200	font				
KFOICnqEu92Fr1Mu4mxK.woff2															200	font				
KFOICnqEu92Fr1MmEU9fChc4EsA.woff2															200	font				
KFOICnqEu92Fr1MmEU9fBc4.woff2															200	font				
KFOICnqEu92Fr1MmWUifChc4EsA.woff2															200	font				
KFOICnqEu92Fr1MmWUifBc4.woff2															200	font				
KFOICnqEu92Fr1MmYUtfChc4EsA.woff2															200	font				
KFOICnqEu92Fr1MmYUtfBc4.woff2															200	font				
icons.ttf?k2isih															200	ont				
csm_Otwarcie_Hali_Sportowej_KBochenek_budynek_b8a6acb8f8.jpg															200	peg	merged-1c08c70...-1253582...css.gzip	(memory cache)	0 ms	
csm_dni_prof_hoborskiego_agh_f365558de1.jpg															200	peg	merged-1c08c70...-1253582...css.gzip	(memory cache)	0 ms	
google-analytics_ga.js?secret=xqf2j															200	script	ga.js	4.8 kB	69 ms	
dekor_1_pasek.svg															200	vg+xml	main.css	(memory cache)	0 ms	
dekor_1.svg															200	vg+xml	main.css	(memory cache)	0 ms	
gradient_pod_rorator_2.png															200	ong	merged-1c08c70...-1253582...css.gzip	(memory cache)	0 ms	

Name	Status	Type	Initiator	Size	Time	Waterfall
www.agh.edu.pl	200	loc				
merged-1c08c701b09f6bc7a8644f282c941fbcd-1253582cdec057d910883174a183ceed.css.gzip?1700...	200	styl				
main.css	200	styl				
Logo.svg	200	vg				
csm_Otwarcie_Hali_Sportowej_KBochenek_budynek_48b3081dbf.webp	200	wet				
csm_dni_prof_hoborskiego_agh_e414f4436f.webp	200	wet				
csm_Barka_a7ff0a7bff.webp	200	wet				
csm_engi_rank_z_tlem_cf54c91ef1.webp	200	wet				
csm_budynek_wydziału_fizyki_i_informatyki_stosowanej_agh_4f4f8e824d.webp	200	wet				
csm_superkomputery_dreamtime_8487230aa3.webp	200	wet				
csm_Cyber_Kampus_eac68692d8.webp	200	wet				
csm_naukowiec_mikroskop_elektronowy_4d7b8bf044.jpg	200	peg				
merged-1a7cb488938ab12683cf8e7ae9c556e-38f73c9ae18281a2de3b29679a131d7e.js.gzip?17001...	200	scri				
agh_sg_bg.js?1700140550	200	scri				
isolated-first.js	200	etc				
css?family=Roboto:ital,wght@0,100;0,300;0,400;0,5...;1,100;1,300;1,400;1,500;1,700;1,900&display=...	200	styl				
ga.js	307	scri				
KFOICnqEu92Fr1MmSU5fChc4EsA.woff2	200	on				
KFOICnqEu92Fr1MmSU5fBc4.woff2	200	on				
KFOICnqEu92Fr1Mu7GxKOzY.woff2	200	on				
KFOICnqEu92Fr1Mu4mxK.woff2	200	on				
KFOICnqEu92Fr1MmEU9fChc4EsA.woff2	200	on				
KFOICnqEu92Fr1MmEU9fBc4.woff2	200	on				
KFOICnqEu92Fr1MmWUifChc4EsA.woff2	200	on				
KFOICnqEu92Fr1MmWUifBc4.woff2	200	on				
KFOICnqEu92Fr1MmYUtfChc4EsA.woff2	200	on				
KFOICnqEu92Fr1MmYUtfBc4.woff2	200	on				
icons.ttf?k2isih	200	ont	merged-1c08c70...-1253582....css.gzip	(memory cache)	0 ms	
csm_Otwarcie_Hali_Sportowej_KBochenek_budynek_b8a6acb8f8.jpg	200	peg	merged-1c08c70...-1253582....css.gzip	(memory cache)	0 ms	
csm_dni_prof_hoborskiego_agh_f365558de1.jpg	200	peg	merged-1c08c70...-1253582....css.gzip	(memory cache)	0 ms	
google-analytics_ga.js?secret=xzqf2j	200	script	ga.js	4.8 kB	69 ms	
dekor_1_pasek.svg	200	vg+xml	main.css	(memory cache)	0 ms	
dekor_1.svg	200	vg+xml	main.css	(memory cache)	0 ms	
gradient_pod_rorator_2.png	200	png	merged-1c08c70...-1253582....css.gzip	(memory cache)	0 ms	

Kody odpowiedzi HTTP

kod	opis słowny	znaczenie/zwracany zasób
200	OK	Zawartość żądanego dokumentu (najczęściej zwracany nagłówek odpowiedzi w komunikacji WWW Internetu)
307	Temporary Redirect	Tymczasowe przekierowanie – żądany zasób znajduje się chwilowo pod innym adresem URI, odpowiedź powinna zawierać zmieniony adres zasobu, na który klient zobowiązany jest się przenieść



Name	Headers	Preview	Response	Initiator	Timing	Cookies
<code>www.agh.edu.pl</code>						
<input type="checkbox"/> isolated-first.js						
<input checked="" type="checkbox"/> merged-1c08c701b09f6bc7...						
<input checked="" type="checkbox"/> main.css						
<input checked="" type="checkbox"/> csm_Otwarcie_Hali_Sportow...						
<input checked="" type="checkbox"/> csm_dni_prof_hoborskiego...						
<input checked="" type="checkbox"/> css?family=Robotoitalic,wg...						
<input checked="" type="checkbox"/> csm_Barka_8b00d656cf.webp						
<input checked="" type="checkbox"/> csm_engi_rank_z_titem_b7d3...						
<input checked="" type="checkbox"/> gajs						
<input checked="" type="checkbox"/> csm_budynek_wydziału_fizy...						
<input checked="" type="checkbox"/> merged-1ac7bb488938ab1...						
<input checked="" type="checkbox"/> agh_sg_bgjs?1700140550						
<input checked="" type="checkbox"/> Logo.svg						
<input checked="" type="checkbox"/> csm_superkomputery_drea...						
<input checked="" type="checkbox"/> csm_Cyber_Kampus_ff842c3...						
<input checked="" type="checkbox"/> csm_naukowiec_mikroskop_...						
<input checked="" type="checkbox"/> data:image/svg+xml,...						
<input checked="" type="checkbox"/> dekor_1_pasek.svg						
<input checked="" type="checkbox"/> data:image/svg+xml,...						
<input checked="" type="checkbox"/> KFOICnqEu92Fr1MmSU5FB...						
<input checked="" type="checkbox"/> KFOICnqEu92Fr1MmEU9fBB...						
<input checked="" type="checkbox"/> KFOICnqEu92Fr1Mu4mxK...						
<input checked="" type="checkbox"/> icons.ttf?k2iish						
<input checked="" type="checkbox"/> KFOICnqEu92Fr1MmWUlfB...						
<input checked="" type="checkbox"/> KFOICnqEu92Fr1MmYUtfBB...						
<input checked="" type="checkbox"/> KFOICnqEu92Fr1MmSU5Fch...						
<input checked="" type="checkbox"/> KFOICnqEu92Fr1Mu7GxKO...						
<input checked="" type="checkbox"/> KFOICnqEu92Fr1MmWUfC...						
<input checked="" type="checkbox"/> KFOICnqEu92Fr1MmEU9Ch...						
<input checked="" type="checkbox"/> KFOICnqEu92Fr1MmYUtfCh...						
<input checked="" type="checkbox"/> google-analytics_gajs?secre...						
<input checked="" type="checkbox"/> sddefault.jpg						
<input checked="" type="checkbox"/> sddefault.jpg						

Elements Recorder Console Sources Network Performance Memory Application Security Lighthouse AdBlock

Preserve log Disable cache No throttling

Filter Invert Hide data URLs Hide extension URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Blocked response cookies Blocked requests 3rd-party requests

20 ms 40 ms 60 ms 80 ms 100 ms 120 ms 140 ms 160 ms 180 ms 200 ms 220 ms 240 ms 260 ms 280 ms 300 ms 320 ms 340 ms 360 ms 380 ms 400 ms 420 ms 440 ms 460 ms 480 ms 500 ms

Name

www.agh.edu.pl

isolated-first.js

merged-1c08c701b09f6bc7...

main.css

csm_Otwarcie_Hali_Sportowej...

csm_dni_prof_hoborskiego_...

css?family=Robotoitalic,wg...

csm_Barka_8b00d656cf.webp

csm_engi_rank_z_titem_b7d3...

gajs

csm_budynek_wydziału_fizy...

merged-1ac7bb488938ab1...

agh_sg_bgjs?1700140550

Logo.svg

csm_superkomputery_drea...

csm_Cyber_Kampus_ff842c3...

csm_naukowiec_mikroskop_...

data:image/svg+xml,...

dékor_1_pasek.svg

data:image/svg+xml,...

KFOICnqEu92Fr1MmSU5FB...

KFOICnqEu92Fr1MmEU9fBB...

KFOICnqEu92Fr1Mu4mxK...

icons.ttf?k2iish

KFOICnqEu92Fr1MmWUlfB...

KFOICnqEu92Fr1MmYUtfBB...

KFOICnqEu92Fr1MmSU5Fc...

KFOICnqEu92Fr1Mu7GxKO...

KFOICnqEu92Fr1MmWUfC...

KFOICnqEu92Fr1MmEU9Ch...

KFOICnqEu92Fr1MmYUtfCh...

google-analytics_gajs?secre...

sddefault.jpg

sddefault.jpg

37 requests | 19.1 kB transferred

X Headers Preview Response Initiator Timing Cookies

General

Request URL: https://www.agh.edu.pl/

Request Method: GET

Status Code: 200 OK

Remote Address: 149.156.96.15:443

Referrer Policy: origin

Response Headers

Cache-Control: max-age=0

Connection: Upgrade, Keep-Alive

Content-Encoding: gzip

Content-Language: pl

Content-Length: 13710

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

Date: Tue, 21 Nov 2023 15:31:36 GMT

Expires: Tue, 21 Nov 2023 15:31:36 GMT

Keep-Alive: timeout=3, max=512

Server: Apache

Strict-Transport-Security: max-age=15552000

Upgrade: h2,h2c

Vary: Accept-Encoding

X-Content-Type-Options: nosniff

X-Typo3-Debug-Cache: Cached page generated 21-11-23 15:13. Expires 22-11-23 15:13

X-Typo3-Parsetime: 0ms

X-UA-Compatible: IE=edge

Request Headers

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7

Accept-Encoding: gzip, deflate, br

Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7

Cache-Control: max-age=0

Connection: keep-alive

Cookie: _ga=GA1.3.1346797581.1683382450; cookiesession1=678B28A7272578691D948A405BF58168

Host: www.agh.edu.pl

Referer: https://www.google.com/

Sec-Ch-Ua: "Chromium";v="118", "Opera GX";v="104", "Not=A?Brand";v="99"

Sec-Ch-Ua-Mobile: ?0

GET – pobranie zasobu wskazanego przez URI, może mieć postać warunkową, jeśli w nagłówku występują pola warunkowe takie jak „If-Modified-Since”

Elements Recorder Console Sources Network Performance Memory Application Security Lighthouse AdBlock

Preserve log Disable cache No throttling

Filter Invert Hide data URLs Hide extension URLs All Fetch/XHR JS CSS Img Media Font Doc

20 ms 40 ms 60 ms 80 ms 100 ms 120 ms 140 ms 160 ms 180 ms 200 ms 220 ms 240 ms

Name

- www.agh.edu.pl
- merged-1c08c701b09f6bc7...
- main.css
- csm_Otwarcie_Hali_Sportowej...
- csm_dni_prof_hoborskiego...
- css2?family=Robotoitalic,wg...
- csm_Barka_8b00d656cf.webp
- csm_engi_rank_z_titem_b7d3...
- gajs
- csm_budynek_wydziału_fizy...
- merged-1ac7bb488938ab1...
- agh_sg_bgjs?1700140550
- Logo.svg
- csm_superkomputery_drea...
- csm_Cyber_Kampus_ff842c3...
- csm_naukowiec_mikroskop_...
- data:image/svg+xml,...
- dekor_1_pasek.svg
- data:image/svg+xml,...
- KFOICnqEu92Fr1MmSU5FB...
- KFOICnqEu92Fr1MmEU9fBB...
- KFOICnqEu92Fr1Mu4mxK...
- icons.ttf?k2isih
- KFOICnqEu92Fr1MmWUlfB...
- KFOICnqEu92Fr1MmYUtfB...
- KFOICnqEu92Fr1MmSU5FCh...
- KFOICnqEu92Fr1Mu7GxKO...
- KFOICnqEu92Fr1MmWUfC...
- KFOICnqEu92Fr1MmEU9fCh...
- KFOICnqEu92Fr1MmYUtfCh...
- google-analytics_gajs?secre...
- sddefault.jpg
- sddefault.jpg

Headers Preview Response Initiator Timing Cookies

General

Request URL: https://www.agh.edu.pl/

Request Method: GET

Status Code: 200 OK

Remote Address: 149.156.96.15:443

Referrer Policy: origin

Response Headers

Cache-Control:	max-age=0
Connection:	Upgrade, Keep-Alive
Content-Encoding:	gzip
Content-Language:	pl
Content-Length:	13710
Content-Type:	text/html; charset=utf-8
Date:	Tue, 21 Nov 2023 15:31:36 GMT
Expires:	Tue, 21 Nov 2023 15:31:36 GMT
Keep-Alive:	timeout=3, max=512
Server:	Apache
Strict-Transport-Security:	max-age=15552000
Upgrade:	h2,h2c
Vary:	Accept-Encoding
X-Content-Type-Options:	nosniff
X-Type3-Debug-Cache:	Cached page generated 21-11-23 15:13. Expires 22-11-23 15:13
X-Type3-Parsetime:	0ms
X-UA-Compatible:	IE=edge

Request Headers

Accept:	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding:	gzip, deflate, br
Accept-Language:	pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Cache-Control:	max-age=0
Connection:	keep-alive
Cookie:	_ga=GA1.3.1346797581.1683382450; cookiesession1=678B28A7272578691D948A405BF58168
Host:	www.agh.edu.pl
Referer:	https://www.google.com/
Sec-Ch-Ua:	"Chromium";v="118", "Opera GX";v="104", "Not=A?Brand";v="99"
Sec-Ch-Ua-Mobile:	?0

• **Accept:** Określa rodzaje mediów (np. tekst, obraz, aplikacja), które klient jest gotów obsłużyć.

• **Accept-Encoding:** Informuje serwer o preferowanych algorytmach kompresji dla przesyłanej treści.

• **Accept-Language:** Określa preferowany język klienta dla treści, które mogą mieć warianty językowe.

• **Cache-Control:** Kontroluje zachowanie pamięci podręcznej, ustalając, czy dane powinny być cachowane i jak długo.

• **Connection:** Określa, czy połączenie powinno być utrzymane otwarte czy zamknięte po przesłaniu odpowiedzi.

• **Cookie:** Przesyła dane ciasteczka, które wcześniej zostały ustawione przez serwer.

• **Host:** Określa nazwę hosta serwera, do którego kierowane jest żądanie.

• **Referer:** Informuje serwer o źródle (URL), z którego pochodzi żądanie.

• **Sec-Ch-Ua:** Zawiera informacje o User-Agent zabezpieczeń przeglądarki.

• **Sec-Ch-Ua-Mobile:** Informuje o preferencjach przeglądarki mobilnej w kontekście zabezpieczeń.

Elements Recorder Console Sources Network Performance Memory Application Security Lighthouse AdBlock

Preserve log Disable cache No throttling

Filter Invert Hide data URLs Hide extension URLs All Fetch/XHR JS CSS Img Media Font Doc WS

20 ms 40 ms 60 ms 80 ms 100 ms 120 ms 140 ms 160 ms 180 ms 200 ms 220 ms 240 ms

Name

- www.agh.edu.pl
- isolated-first.js
- merged-1c8c701b09f6bc7...
- main.css
- csm_Otwarcie_Hali_Sportow...
- csm_dni_prof_hoborskiego...
- css2?family=Robotoitalic,wg...
- csm_Barka_8b00d656cf.webp
- csm_engi_rank_z_tlem_b7d3...
- gajs
- csm_budynek_wydziału_fizy...
- merged-1ac7bb488938ab1...
- agh_sg_bg.js?1700140550
- Logo.svg
- csm_superkomputery_drea...
- csm_Cyber_Kampus_ff842c3...
- csm_naukowiec_mikroskop...
- data:image/svg+xml,...
dekor_1_pasek.svg
- data:image/svg+xml,...
- KFOICnqEu92Fr1MmSU5FB...
- KFOICnqEu92Fr1MmEU9BB...
- KFOICnqEu92Fr1Mu4mxK...
- icons.ttf?k2isih
- KFOICnqEu92Fr1MmWUIfb...
- KFOICnqEu92Fr1MmYUtfB...
- KFOICnqEu92Fr1MmSU5fCh...
- KFOICnqEu92Fr1Mu7GxKO...
- KFOICnqEu92Fr1MmWUIC...
- KFOICnqEu92Fr1MmEU9Ch...
- KFOICnqEu92Fr1MmYUtfCh...
- google-analytics_gajs?secre...
- sddefault.jpg
- sddefault.jpg

X Headers Preview Response Initiator Timing

General

Request URL: https://ssl.google-analytics.com/ga.js
Request Method: GET
Status Code: 307 Temporary Redirect
Referrer Policy: strict-origin-when-cross-origin

Response Headers

Location: chrome-extension://kccohkcppjkkjppopfnfnebibpida/web_accessible...

Non-Authoritative-Reason: WebRequest API

Request Headers

⚠ Provisional headers are shown. [Learn more](#)

Referer: https://www.agh.edu.pl/
Sec-Ch-Ua: "Chromium",v="118", "Opera GX",v="104", "Not=A?Brand",v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 OPR/104.0.0.0

• **Referer:** Informuje serwer o źródle (URL), z którego pochodzi żądanie.

• **Sec-Ch-Ua:** Zawiera informacje o User-Agent zabezpieczeń przeglądarki.

• **Sec-Ch-Ua-Mobile:** Informuje o preferencjach przeglądarki mobilnej w kontekście zabezpieczeń.

• **Sec-Ch-Ua-Platform:** Zawiera informacje o platformie (np. "Windows", "Linux", "Mac") zabezpieczeń przeglądarki.

• **User-Agent:** Nagłówek żądania, który dostarcza informacje o przeglądarce, systemie operacyjnym i innych charakterystykach klienta, pomagając serwerowi dostosować treść i funkcje odpowiedzi do konkretnego środowiska użytkownika.

37 requests | 19.1 kB transferred

•**Cookie:** Przesyła dane ciasteczka, które wcześniej zostały ustawione przez serwer.

•**Host:** Określa nazwę hosta serwera, do którego kierowane jest żądanie.

•**Referer:** Informuje serwer o źródle (URL), z którego pochodzi żądanie.

•**Sec-Ch-Ua:** Zawiera informacje o User-Agent zabezpieczeń przeglądarki.

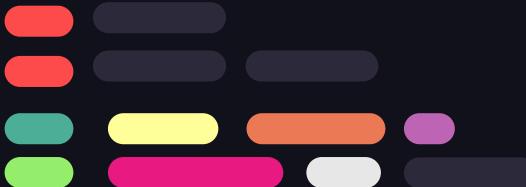
•**Sec-Ch-Ua-Mobile:** Informuje o preferencjach przeglądarki mobilnej w kontekście zabezpieczeń.



{

Koniec

Wykonał: Jakub Skorupa



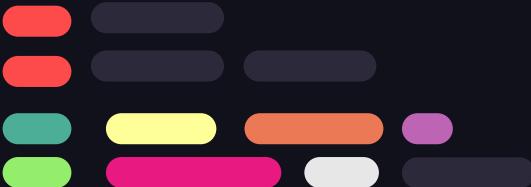
}

Źródła:

{

- <https://www.agh.edu.pl>
- <https://www.ibm.com/docs/pl/ibm-mq/9.0?topic=http-headers>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP>Status>

}



REST API i HTTP, a(g)RPC

Adam Kostuch ETI rok 3



>>>

Spis treści

.....

01 HTTP (API)

02 REST API

03 (g)RPC

04 Porównanie

01



HTTP

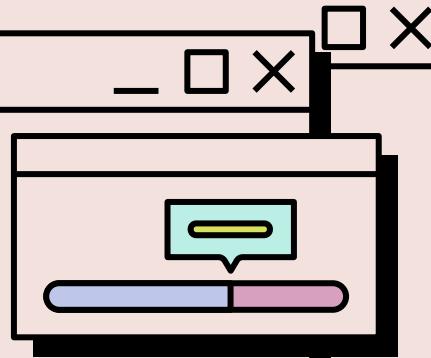


HTTP



>>>>

HTTP - definicja



HTTP (Hypertext Transfer Protocol) to protokół komunikacyjny, który służy do przesyłania danych między klientem a serwerem w World Wide Web. Jest to podstawowy protokół używany do żądania zasobów, takich jak strony internetowe, obrazy, pliki CSS, itp., z serwera do przeglądarki internetowej klienta i przesyłania odpowiedzi z klienta do serwera.



>>>>

HTTP - zalety



Prostota

HTTP jest prostym protokołem, co ułatwia jego zrozumienie i implementację

Uniwersalność

Powszechnie stosowany na całym świecie, co sprawia, że jest powszechnie obsługiwany przez różne platformy i urządzenia.

Skalowalność

Protokół HTTP jest skalowalny i może obsługiwać duże ilości klientów i serwerów.

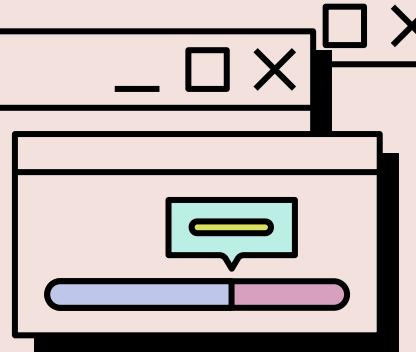
Rozszerzalność

HTTP może być rozszerzany poprzez dodawanie funkcji i nagłówków, co pozwala na różne zastosowania.



>>>>

HTTP API - definicja



HTTP API (Application Programming Interface) to interfejs programistyczny, który wykorzystuje protokół HTTP jako środek komunikacji między aplikacjami lub serwisami. API

HTTP definiuje konkretne punkty końcowe (URL), formaty żądań i odpowiedzi oraz dostępne operacje, które aplikacje mogą wykonywać w celu uzyskania dostępu do danych lub funkcji oferowanych przez serwer.

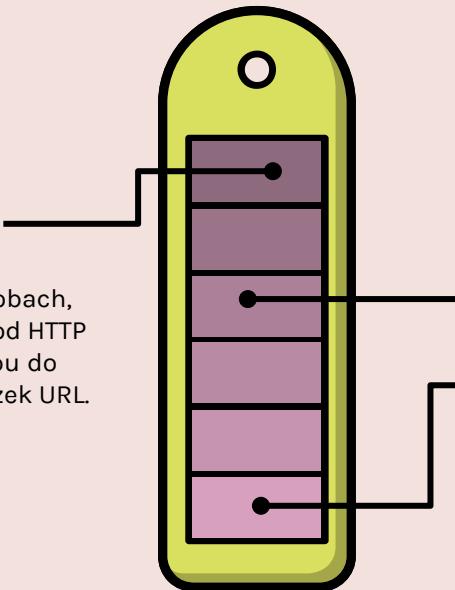


HTTP API - wykorzystanie ~~~~>>>

.....

REST API

Styl architektoniczny oparty na zasobach, który korzysta z standardowych metod HTTP (GET, POST, PUT, DELETE) do dostępu do zasobów za pomocą unikalnych ścieżek URL.



GraphQL

Język zapytań i środowisko wykonawcze, które umożliwia klientowi określenie dokładnie jakie dane chce pobrać, co pozwala na bardziej elastyczną komunikację z serwerem.



SOAP

to protokół komunikacyjny, który definiuje, jak klient i serwer wymieniają dane w formie strukturalnie zdefiniowanych wiadomości XML, niezależnie od platformy i języka programowania.



02

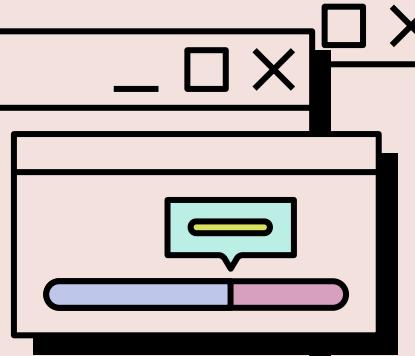
REST API

{REST}



>>>>

REST API - definicja

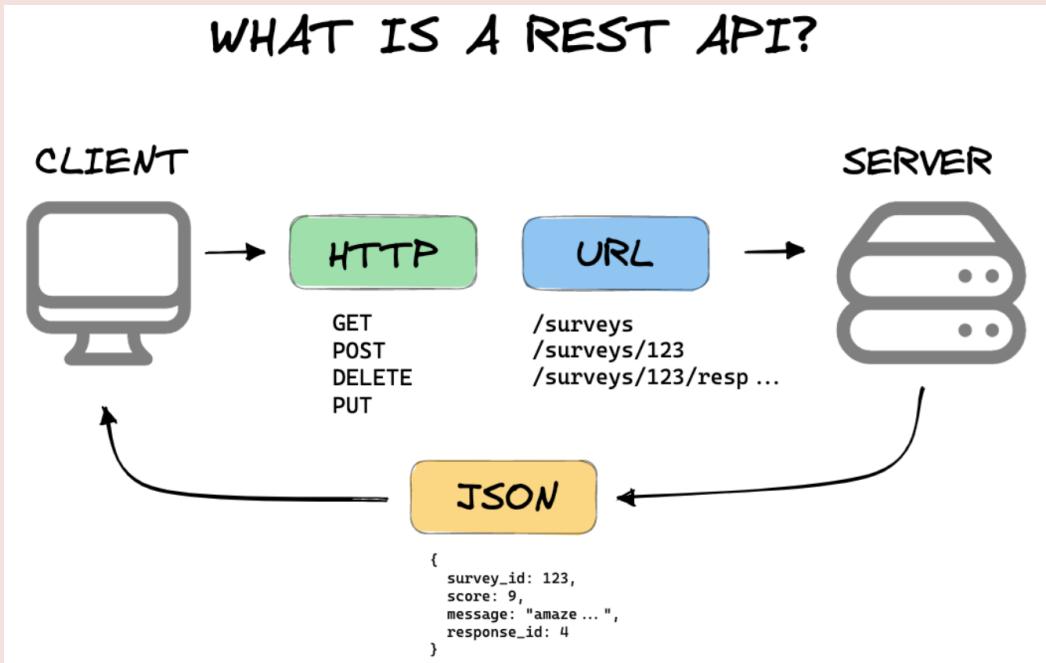


REST API (Representational State Transfer Application Programming Interface) to styl architektury oprogramowania, który definiuje zestaw zasad i konwencji, które umożliwiają komunikację między różnymi systemami komputerowymi w sposób zrozumiały dla ludzi i maszyn. REST jest jednym z popularnych sposobów projektowania API w świecie aplikacji internetowych.



.....

REST API - działanie



REST API - zalety

>>>

Zasoby

Na przykład, jeśli projektujesz REST API dla systemu zarządzania książkami, każda książka mogłaby być traktowana jako osobny zasób, a każda operacja na książce byłaby dostępna poprzez odpowiednie adresy URL.

Bezwładność

REST jest bezstanowy, co oznacza, że każde zapytanie od klienta do serwera musi zawierać wszystkie informacje potrzebne do zrozumienia żądania. Serwer nie przechowuje żadnego stanu o klientach między zapytaniami.

Klient-Serwer

REST promuje podział systemu na klienta i serwer. Klient jest odpowiedzialny za interakcję z użytkownikiem i prezentację danych, podczas gdy serwer jest odpowiedzialny za przechowywanie i zarządzanie danymi oraz przetwarzanie żądań klienta.

.....

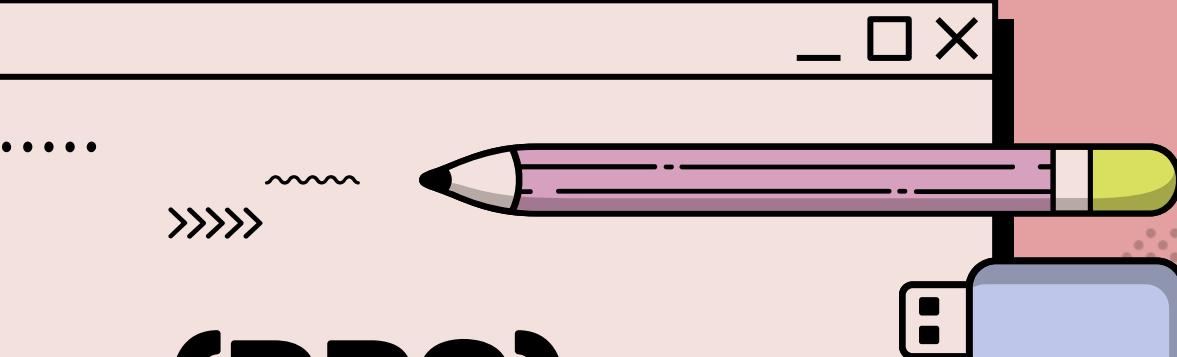
REST API - przykłady

~~~~~  
>>>>

|                                        |                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------------------------|--------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>API-Football</b>                    | +1 000 gier ligowych & spotkań.<br>Wyniki na żywo (15s), typy na żywo, wydarzenia, składy, trenerzy... | - GET <a href="https://api-football-v1.p.rapidapi.com/v3/predictions">https://api-football-v1.p.rapidapi.com/v3/predictions</a><br>- GET <a href="https://api-football-v1.p.rapidapi.com/v3/odds/bets">https://api-football-v1.p.rapidapi.com/v3/odds/bets</a><br>- GET <a href="https://api-football-v1.p.rapidapi.com/v3/teams">https://api-football-v1.p.rapidapi.com/v3/teams</a>                                                   |
| <b>Yahoo Finance</b>                   | API podaje wszystkie dane finansowe ze strony finance.yahoo.com.                                       | - GET <a href="https://apidojo-yahoo-finance-v1.com/market/get-earnings">https://apidojo-yahoo-finance-v1.com/market/get-earnings</a><br>- GET <a href="https://apidojo-yahoo-finance-v1.com/stock/get-summary">https://apidojo-yahoo-finance-v1.com/stock/get-summary</a><br>- POST <a href="https://apidojo-yahoo-finance-v1.p.rapidapi.com/news/v2/list">https://apidojo-yahoo-finance-v1.p.rapidapi.com/news/v2/list</a>            |
| <b>WordsAPI</b>                        | Informacji o angielskich słowach, ich definicjach, synonimach, a nawet rymach.                         | - GET <a href="https://wordsapi.v1.p.rapidapi.com/words/hatchback/typeOf">https://wordsapi.v1.p.rapidapi.com/words/hatchback/typeOf</a><br>- GET <a href="https://wordsapi.v1.p.rapidapi.com/words/%7Bword%7D/rhymes">https://wordsapi.v1.p.rapidapi.com/words/%7Bword%7D/rhymes</a><br>- GET <a href="https://wordsapi.v1.p.rapidapi.com/words/incredible/syllables">https://wordsapi.v1.p.rapidapi.com/words/incredible/syllables</a> |
| <b>Amazon Product/Reviews/Keywords</b> | Informacje na żywo z API Amazonu. Produkty, oceny, słowa kluczowe, łatwo dostępne.                     | - GET <a href="https://amazon-product-reviews-keywords.com/product/search">https://amazon-product-reviews-keywords.com/product/search</a><br>- GET <a href="https://amazon-product-reviews-keywords.com/categories">https://amazon-product-reviews-keywords.com/categories</a><br>- GET <a href="https://amazon-product-reviews-keywords.com/product/reviews">https://amazon-product-reviews-keywords.com/product/reviews</a>           |

03

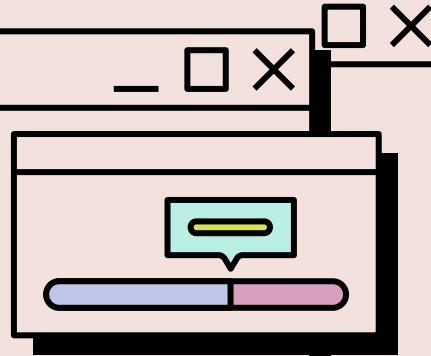
# g(RPC)





>>>>

## RPC - definicja



RPC (Remote Procedure Call) to mechanizm, który umożliwia zdalne wywoływanie procedur lub funkcji w innej przestrzeni adresowej, na innym komputerze lub w innej zdalnej lokalizacji, jako gdyby były wywoływane lokalnie.

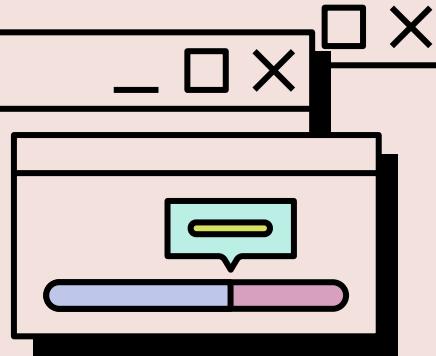
RPC pozwala na komunikację między aplikacjami lub komponentami w sposób, który jest abstrahowany od szczegółów komunikacji sieciowej.





>>>>

## gRPC - definicja



gRPC (gRPC Remote Procedure Call) to otwarty framework opracowany przez Google, który umożliwia zdalne wywoływanie procedur, czyli komunikację między aplikacjami na różnych systemach. Jest to technologia stworzona w oparciu o protokół HTTP/2, co pozwala na efektywną komunikację między klientem a serwerem.

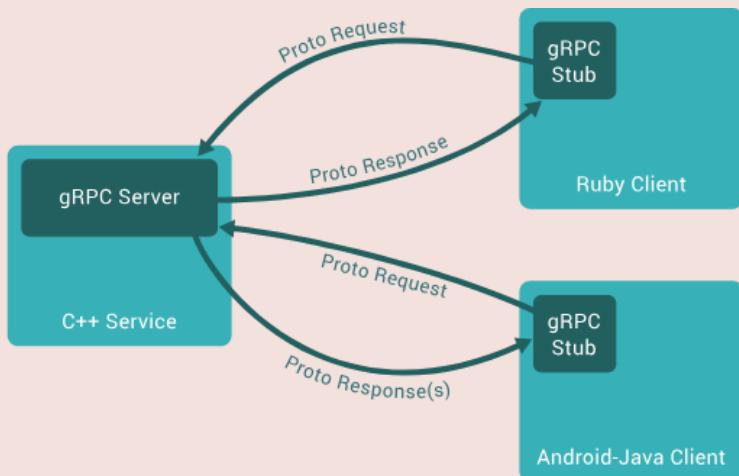


~~~~~  
.....

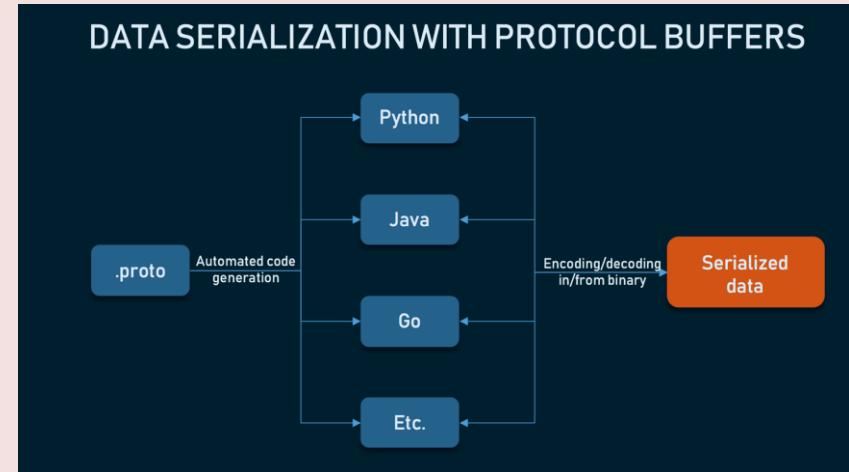


.....

gRPC - działanie



~~~~~  
>>>



04



# Porównanie



GRPC

{REST}

.....

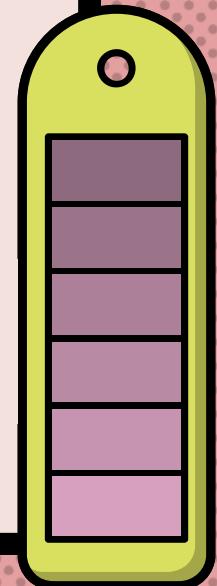
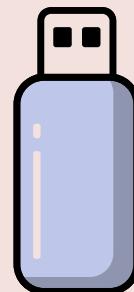
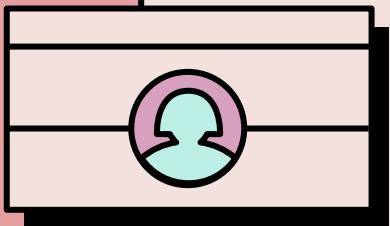
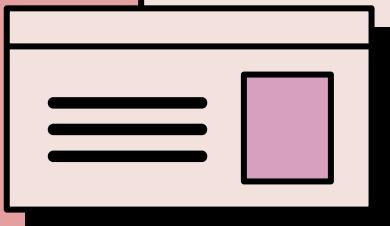
# Porównanie

&gt;&gt;&gt;&gt; ~~~~

| Zasób             | gRPC                              | REST                               |
|-------------------|-----------------------------------|------------------------------------|
| Protokół          | HTTP/2 (szybkie)                  | HTTP/1.1 (wolne)                   |
| Przesyłane dane   | Protobuf (bit)                    | JSON (tekst)                       |
| Generacja kodu    | Protoc (wbudowany)                | Swagger (zewnętrzne)               |
| Protokół ochronny | TLS/SSL                           | TLS/SSL                            |
| Transmisja        | Dwukierunkowa                     | Jednokierunkowa (klient -> serwer) |
| Dostępność        | Ograniczona (gRPC web - wymagane) | Wszystkie przeglądarki             |
| Prostota          | Cieższy w zrozumieniu             | Jedno z łatwiejszych rozwiązań     |

# KONIEC

Adam Kostuch ETI rok 3





# Bibliografia

1. <https://grpc.io/about/> (dostęp z dnia 07.11.2023)
2. <https://medium.com/swlh/grpc-fundamental-and-concept-93414d7956df>  
(dostęp z dnia 07.11.2023)
3. <https://cloudacademy.com/course/what-is-an-api-how-do-they-work-1932/the-difference-between-http-and-rest/> (dostęp z dnia 09.11.2023)
4. <https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-vs-rest.html> (dostęp z dnia 09.11.2023)
5. <https://www.redhat.com/en/topics/api/what-is-a-rest-api> (dostęp z dnia 10.11.2023)

# XMLHttpRequest VS fetch

Szymon Skubis ETI rok 3



# Czym jest AJAX?

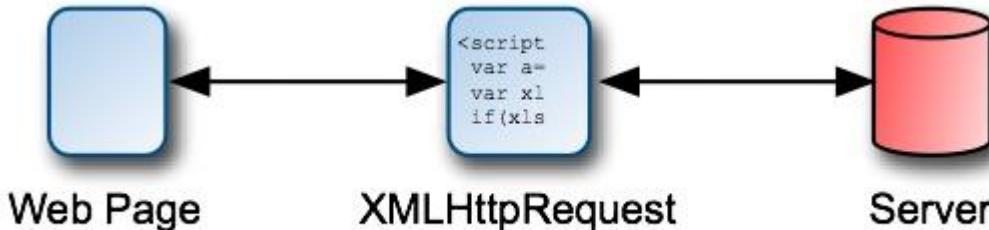
AJAX, czyli Asynchronous JavaScript and XML, to technika programowania umożliwiająca asynchroniczną komunikację między klientem (przeglądarką internetową) a serwerem. Głównym celem AJAX jest umożliwienie dynamicznego ładowania danych na stronie internetowej bez konieczności przeładowywania całej strony. Dzięki temu użytkownicy doświadczają płynniejszych i bardziej interaktywnych doświadczeń online.

Do wykonania żądań HTTP w technologii AJAX używane są dwie główne technologie: XMLHttpRequest i Fetch API.



# XMLHttpRequest

Jest to starsza technologia, wprowadzona pierwotnie przez Microsoft. Pozwala na wysyłanie żądań HTTP z poziomu przeglądarki oraz obsługuje odpowiedzi w różnych formatach, w tym w formie XML, JSON lub tekstu. Jednak API XMLHttpRequest ma skomplikowany interfejs i jest mniej intuicyjne w użyciu.



# XMLHttpRequest

## Zalety:

- Wsparcie dla starszych przeglądarek: XMLHttpRequest jest obsługiwane przez większość współczesnych przeglądarek, co pozwala na stosowanie tej technologii w projektach, które muszą być kompatybilne z starszymi przeglądarkami.
- Obsługa różnych typów danych: XMLHttpRequest może obsługiwać różne typy danych w odpowiedzi, takie jak tekst, XML, JSON.

## Wady:

- Skomplikowany interfejs API: Interfejs XMLHttpRequest jest dość złożony i wymaga wielu linii kodu do wykonania prostych operacji. Z tego powodu może być trudny w użyciu, szczególnie dla początkujących programistów.
- Obsługa różnych stanów readyState: Do obsługi asynchronicznych operacji, XMLHttpRequest korzysta ze zmiennych, takich jak readyState które reprezentują różne etapy przetwarzania żądania. Programiści muszą śledzić te stany i sprawdzać, czy żądanie jest gotowe do przetworzenia.
- Brak wsparcia dla Promises: XMLHttpRequest nie obsługuje Promises, co może sprawić, że kod staje się mniej czytelny i bardziej podatny na tzw. "callback hell" (syndrom zwany także "callback pyramid").



# Fetch API

Jest to nowsze API wprowadzone wraz z ECMAScript 6. Zapewnia bardziej nowoczesny i bardziej elastyczny interfejs do obsługi żądań HTTP. Jest bardziej rozbudowane i bardziej zgodne z nowoczesnymi standardami JavaScript. Fetch API obsługuje Promises, co ułatwia pracę z danymi asynchronicznymi.





# Fetch API

## Zalety Fetch API:

- **Prostszy interfejs:** Fetch API oferuje znacznie prostszy interfejs w porównaniu do XMLHttpRequest. Wykorzystuje Promises, co znacznie ułatwia zarządzanie asynchronicznym kodem i eliminuje konieczność obsługi różnych stanów `readyState`.
- **Obsługa Promises:** Fetch API bazuje na koncepcji Promises, co sprawia, że kod staje się bardziej czytelny i bardziej zorganizowany. Działa to efektywnie w środowisku opartym na asynchroniczności, co znacznie ułatwia pracę programistyczną.
- **Nowoczesna składnia:** Fetch API korzysta z bardziej nowoczesnej składni, co sprawia, że jest bardziej zgodne z nowymi standardami JavaScript. Jest bardziej zwięzłe i wygodne w użyciu, co przyczynia się do poprawy czytelności kodu.
- **Łatwa obsługa nagłówków i parametrów żądania:** Fetch API umożliwia łatwe dodawanie nagłówków i parametrów do żądania, co jest bardziej wygodne niż w przypadku XMLHttpRequest.



## **Przykłady sytuacji, w których jedna z technologii może być bardziej odpowiednia niż druga:**

### **1. Jeśli prostota i czytelność są kluczowe:**

- **Fetch API:** Bardziej nowoczesna składnia oparta na Promises sprawia, że kod jest bardziej zwięzły i czytelny.

### **2. Jeśli istnieje potrzeba obsługi różnych typów danych:**

- **XMLHttpRequest:** W przypadku, gdy konieczne jest bezpośrednie zarządzanie różnymi typami danych, XMLHttpRequest może być bardziej elastyczne.

### **3. Gdy obsługa błędów jest istotna:**

- **Fetch API:** Automatyczna obsługa błędów za pomocą Promises może uprościć kod i zwiększyć jego czytelność.

### **4. Dla starszych projektów i przeglądarek:**

- **XMLHttpRequest:** Jeśli projekt wymaga obsługi starszych przeglądarek, XMLHttpRequest może być bardziej praktyczne ze względu na swoją powszechną obsługę.

### **5. Podczas wykonywania synchronicznych żądań:**

- **XMLHttpRequest:** W przypadku, gdy konieczne jest wykonywanie synchronicznych żądań (choć jest to praktyka zalecana do unikania).

# Obsługa Danych w XMLHttpRequest i Fetch API:

## XMLHttpRequest:

Formaty Danych:

- XMLHttpRequest obsługuje różne formaty danych, w tym tekst (`responseText`), XML (`responseXML`) i inne.

Obsługa JSON:

- Obsługa JSON wymaga dodatkowego kroku, takiego jak parsowanie danych przy użyciu `JSON.parse()` w przypadku odpowiedzi w formie tekstu.

```
// Przykład obsługi JSON w XMLHttpRequest
var xhr = new XMLHttpRequest();
xhr.open('GET', 'https://example.com/data', true);
xhr.onreadystatechange = function() {
  if (xhr.readyState == 4 && xhr.status == 200) {
    var responseData = JSON.parse(xhr.responseText);
    console.log(responseData);
  }
};
xhr.send();
```

# Obsługa Danych w XMLHttpRequest i Fetch API:

## Fetch API:

- Formaty Danych:
  - Fetch API oferuje bardziej elastyczną obsługę różnych formatów danych, zwracając obiekt Response.
- Obsługa JSON:
  - Dla danych w formie JSON, można skorzystać z metody json(), która zwraca Promise, a następnie można bezpośrednio pracować z danymi JSON

```
// Przykład obsługi JSON w Fetch API
fetch('https://example.com/data')
  .then(response => {
    if (!response.ok) {
      throw new Error('Network response was not ok');
    }
    return response.json();
  })
  .then(data => {
    console.log(data);
  })
  .catch(error => {
    console.error('Fetch error:', error);
});
});
```



# Porównanie obsługi danych w tych technologiach

## 1. Przejrzystość i Prostota:

- **XMLHttpRequest:** Obsługa różnych formatów danych może być bardziej skomplikowana, szczególnie w przypadku JSON, gdzie wymaga dodatkowego kroku parsowania.
- **Fetch API:** Fetch API oferuje bardziej przejrzyste podejście do obsługi danych. Metoda `json()` ułatwia pracę z danymi JSON, eliminując konieczność ręcznego parsowania.

## 2. Zarządzanie Asynchronicznoscią:

- **XMLHttpRequest:** Wymaga obsługi zdarzenia `onreadystatechange`, co może sprawić, że kod stanie się bardziej rozbudowany, zwłaszcza przy wielu żądaniach.
- **Fetch API:** Działa na zasadzie Promises, co ułatwia zarządzanie asynchronicznym kodem, eliminując potrzebę obsługi wielu stanów `readyState`.

## 3. Wsparcie dla Różnych Formatów:

- **XMLHttpRequest:** Obsługuje różne formaty danych, ale wymaga precyzyjnej konfiguracji i ręcznej obsługi.
- **Fetch API:** Bardziej elastyczne, obsługuje różne formaty danych za pomocą różnych metod (np. `json()`, `text()`).

# Przykład użycia XMLHttpRequest do pobieranie danych z serwera:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>XMLHttpRequest Example</title>
</head>
<body>
    <button onclick="fetchData()">Pobierz Dane</button>
    <div id="result"></div>

    <script>
        function fetchData() {
            var xhr = new XMLHttpRequest();
            xhr.open('GET', 'https://jsonplaceholder.typicode.com/posts/1', true);
            xhr.onreadystatechange = function() {
                if (xhr.readyState == 4 && xhr.status == 200) {
                    var responseData = JSON.parse(xhr.responseText);
                    document.getElementById('result').innerHTML =
                        `<p>Title: ${responseData.title}</p>
                        <p>Body: ${responseData.body}</p>
                    `;
                }
            };
            xhr.send();
        }
    </script>
</body>
</html>
```

# Przykład użycia Fetch API do pobieranie danych z serwera:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fetch API Example</title>
</head>
<body>
    <button onclick="fetchData()">Pobierz Dane</button>
    <div id="result"></div>

    <script>
        function fetchData() {
            fetch('https://jsonplaceholder.typicode.com/posts/1')
                .then(response => {
                    if (!response.ok) {
                        throw new Error('Network response was not ok');
                    }
                    return response.json();
                })
                .then(data => {
                    document.getElementById('result').innerHTML = `
                        <p>Title: ${data.title}</p>
                        <p>Body: ${data.body}</p>
                    `;
                })
                .catch(error => {
                    console.error('Fetch error:', error);
                });
        }
    </script>
</body>
</html>
```

# **POSTMAN**

## Instalacja i testowanie API

Oliwia Putyra



# PLAN PREZENTACJI

**01**

CZYM JEST POSTMAN?

**02**

GŁÓWNE FUNKCJE POSTMANA

**03**

INSTALACJA

**04**

TESTOWANIE  
API



# 01

## Czym jest Postman?

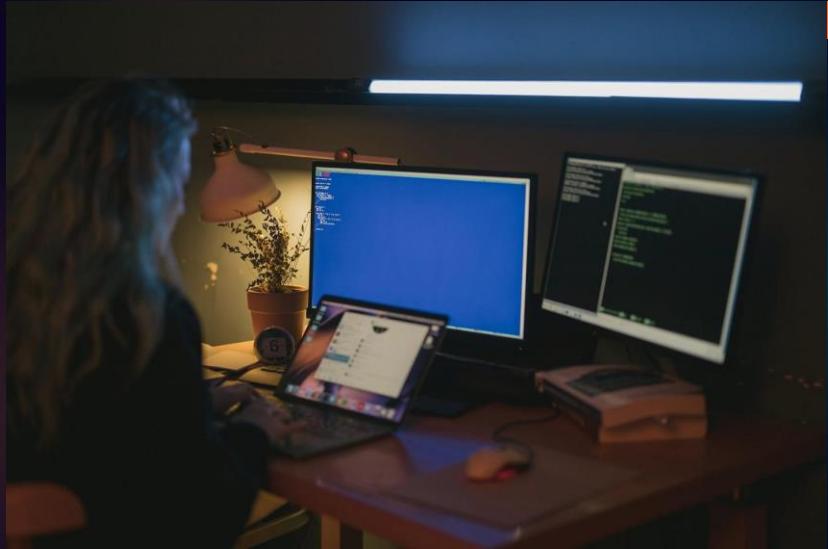
# NARZĘDZIE DO TESTOWANIA API



Postman to platforma, która umożliwia programistom tworzenie, udostępnianie, testowanie i debugowanie zapytań HTTP związanych z interakcjami API.

# **INTUICYJNY INTERFEJS**

**Click Send to get a response**



## 02

# Główne funkcje Postmana

# GŁÓWNE FUNKCJE POSTMANA



## WYSYŁANIE ZAPYTAŃ HTTP

GET, POST, PUT,  
DELETE itp.



## TESTOWANIE API

Testy jednostkowe i  
automatyczne



## ZARZĄDZANIE KOLEKCJAMI

Ułatwienie pracy

# GŁÓWNE FUNKCJE POSTMANA



## TWORZENIE DOKUMENTACJI

Udostępnianie  
zasobów w zespole



## MONITOROWANIE API

Śledzenie wydajności,  
rejestrowanie  
problemów



03

# Instalacja Postmana



# WSPARCIE I WYMAGANIA

## WINDOWS

### Do wersji Postmana v9.4

- 32-bit
- 64-bit

### Powyżej wersji v9.4

- 64-bit x86

## MACOS

- Od wersji macOS **10.11 (El Capitan)** i wyżej

## LINUX

- **Ubuntu** od wersji **14.04** i wyżej
- **Fedora 24**
- **Debian** od wersji **8** i wyżej

# WERSJA PRZEGŁĄDARKOWA

Aplikacja internetowa Postman obsługuje następujące przeglądarki:

- Chrome (78 i nowsze)
- Firefox (76 i nowsze)
- Edge (79 i nowsze)
- Safari (13.1.1 i nowsze)

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Notion's Public Workspace' selected, displaying various API collections like Notion API, Databases, Pages, etc. The main workspace shows a 'GET Retrieve a database' request. The URL is https://api.notion.com/v1/databases/{id}. The 'Params' tab is active, showing a 'Query params' table with one entry: 'id' with value '{DATABASE\_ID}'. Below it is a 'Path Variables' table with one entry: 'id' with value '{id}'. The 'Body' tab shows a JSON response with a single object: "Publisher": { "id": "83E924Pb", "name": "Publisher", "type": "select", "select": { "options": [ { "id": "c5ee409a-f307-4176-99ee-6e424fa89afa", "name": "NYT", "color": "default" } ] } }. The right side of the screen has sections for 'Documentation' (with a link to https://api.notion.com/v1/databases/{id}), 'Authorization' (using Bearer Token), 'Request Header' (Notion-Version: 22-02-22), and 'Path Variables' (id).

# STRONA INTERNETOWA POSTMAN

Product ▾ Pricing Enterprise Resources and Support ▾ Public API Network

Contact Sales Sign In Sign Up for Free

Learning Center ▾

Home / Get started / Install and configure

## Install and update Postman

To get the latest version of the Postman desktop app, visit the [Download Postman page](#) and select the option for your operating system. Postman is available as a native desktop app for macOS (Intel or Apple silicon), Windows (Intel 32-bit or 64-bit), and Linux (64-bit).

Postman is also available as a web app at [go.postman.co/home](https://go.postman.co/home). You can use the Postman web app to carry out many of your API development and testing tasks in your web browser. Keep in mind that some features aren't supported when using the [Postman web app](#), so use the Postman desktop app for the full Postman experience.

### Contents

- ♦ Install Postman on the desktop
  - ♦ Windows

[Edit This Doc](#)

**Additional resources**

Videos

Unboxing What's New in Postman v10 | Postman Intergalactic

Agent for the Postman Web Client | Postman Level Up

Blog posts

Announcing Postman v10: The API Platform for an API-First World

Introducing the Postman Agent: Send API Requests from Your Browser without Limits

# POBRANIE APLIKACJI

Product ▾ Pricing Enterprise Resources and Support ▾ Public API Network

Contact Sales Sign In Sign Up for Free

## Download Postman

Download the app to get started using the Postman API Platform today. Or, if you prefer a browser experience, you can try the web version of Postman.

### The Postman app

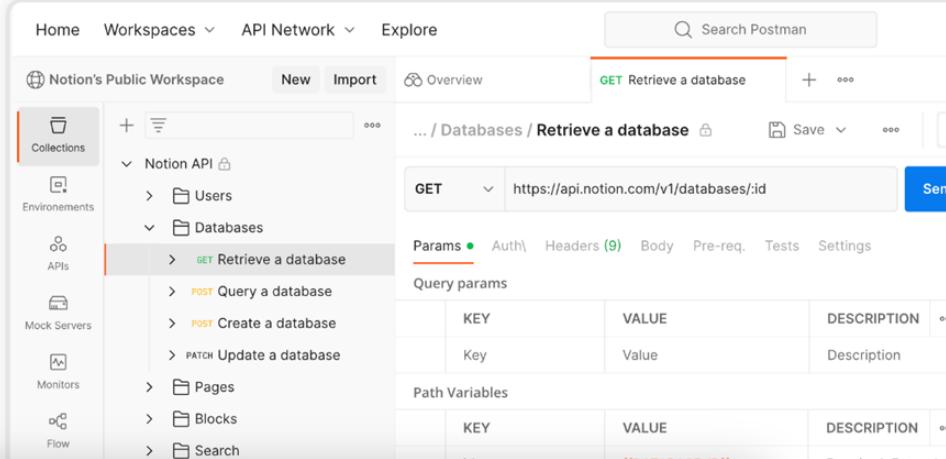
Download the app to get started with the Postman API Platform.

[Mac Intel Chip](#) [Mac Apple Chip](#)

By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

[Release Notes](#) · [Product Roadmap](#)

Not your OS? Download for Windows (x64) or Linux (x64, arm64)





04

## Testowanie API

# BIBLIOGRAFIA

- <https://www.droptica.pl/blog/co-jest-postman-do-czego-sluzy-i-jakie-sa-jego-funkcjonalnosci/>
- <https://www.postman.com/>

# DZIĘKUJĘ

**CREDITS:** This presentation template was created by [Slidesgo](#), and includes icons by [Flaticon](#), and infographics & images by [Freepik](#)

Please keep this slide for attribution

# Swagger

INSTALACJA, ZASADA DZIAŁANIA, PRZYKŁAD ZASTOSOWANIA

# Czym jest swagger?

Swagger to narzędzie, które umożliwia dokumentowanie, rozwijanie i testowanie interfejsów API (Interfejsów Programowania Aplikacji). Jest to narzędzie open-source, które pomaga zdefiniować strukturę, format i dostępność do różnych funkcji API.

Głównym celem Swaggera jest ułatwienie komunikacji pomiędzy zespołami programistycznymi, a także ułatwienie korzystania z API przez deweloperów, testerów i innych zainteresowanych.



# Instalacja Swaggera

W terminalu należy użyć komendy instalującą paczkę `swagger-jsdoc`

```
npm install swagger-jsdoc swagger-ui-express --save
```

Następnie stworzyć plik konfiguracyjny i zdefiniować w nim specyfikacje swojej aplikacji (np. swagger.js)

```
const swaggerJSDoc = require('swagger-jsdoc');

const options = {
  definition: {
    openapi: '3.0.0',
    info: {
      title: 'Nazwa Twojej Aplikacji',
      version: '1.0.0',
      description: 'Opis Twojej aplikacji',
    },
  },
  apis: ['./ściezka/do/plików/z/routingiem/*.js'], // Ścieżka do plików z routingiem
};

const swaggerSpec = swaggerJSDoc(options);

module.exports = swaggerSpec;
```

# Instalacja Swaggera

W pliku konfiguracji aplikacji Express należy dodać następujące linie kodu

```
const express = require('express');
const swaggerUi = require('swagger-ui-express');
const swaggerSpec = require('./ściezka/do/swagger.js');

const app = express();

app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerSpec));
```

W tym momencie można przeglądać dokumentacje Swagger pod adresem  
`<http://localhost:PORT/api-docs>`

# Działanie Swaggera

## Dokumentacja API

Swagger umożliwia generowanie czytelnej i interaktywnej dokumentacji dla interfejsu API. Ta dokumentacja zawiera szczegółowe informacje na temat dostępnych ścieżek, operacji, parametrów, typów danych, odpowiedzi HTTP, a także przykłady użycia.

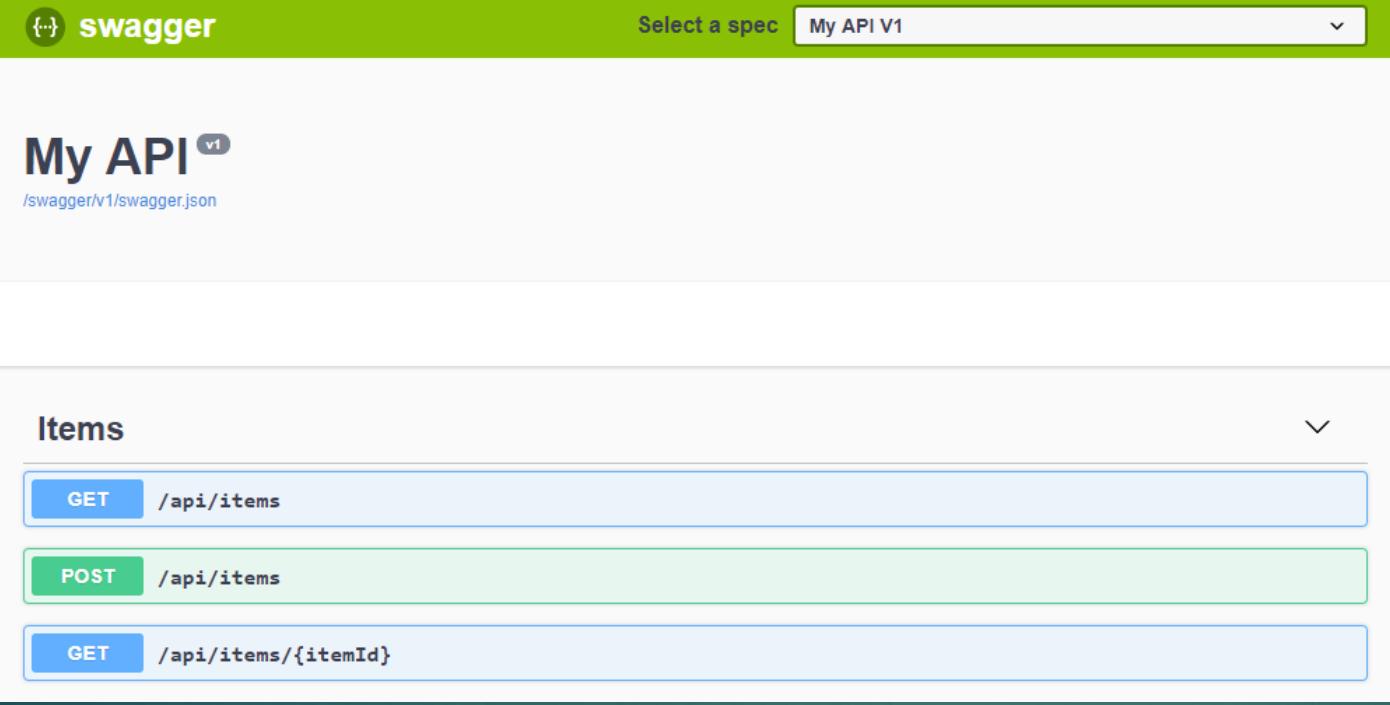
## Testowanie API

Swagger dostarcza interaktywne środowisko do testowania operacji API. Deweloperzy mogą eksperymentować z różnymi parametrami, wysyłać żądania HTTP i sprawdzać, jak API reaguje na różne sytuacje.

## Ułatwienie komunikacji w zespole

Dzięki dostępowi do jednolitej dokumentacji, zespoły programistyczne mogą łatwiej komunikować się i współpracować nad projektami. Swagger stanowi wspólny punkt odniesienia dla specyfikacji API.

# Dokumentacja



The screenshot shows the Swagger UI interface for a RESTful API. At the top, there's a navigation bar with the 'swagger' logo, a dropdown menu 'Select a spec' containing 'My API V1', and a red 'Logout' button. Below the navigation, the title 'My API v1' is shown with the URL '/swagger/v1/swagger.json'. A large, semi-transparent circular watermark with the text 'API DOCUMENTATION' is overlaid on the page. The main content area has a heading 'Items' with a dropdown arrow. Underneath, there are three API endpoints listed in cards:

- GET /api/items** (blue card)
- POST /api/items** (green card)
- GET /api/items/{itemId}** (blue card)

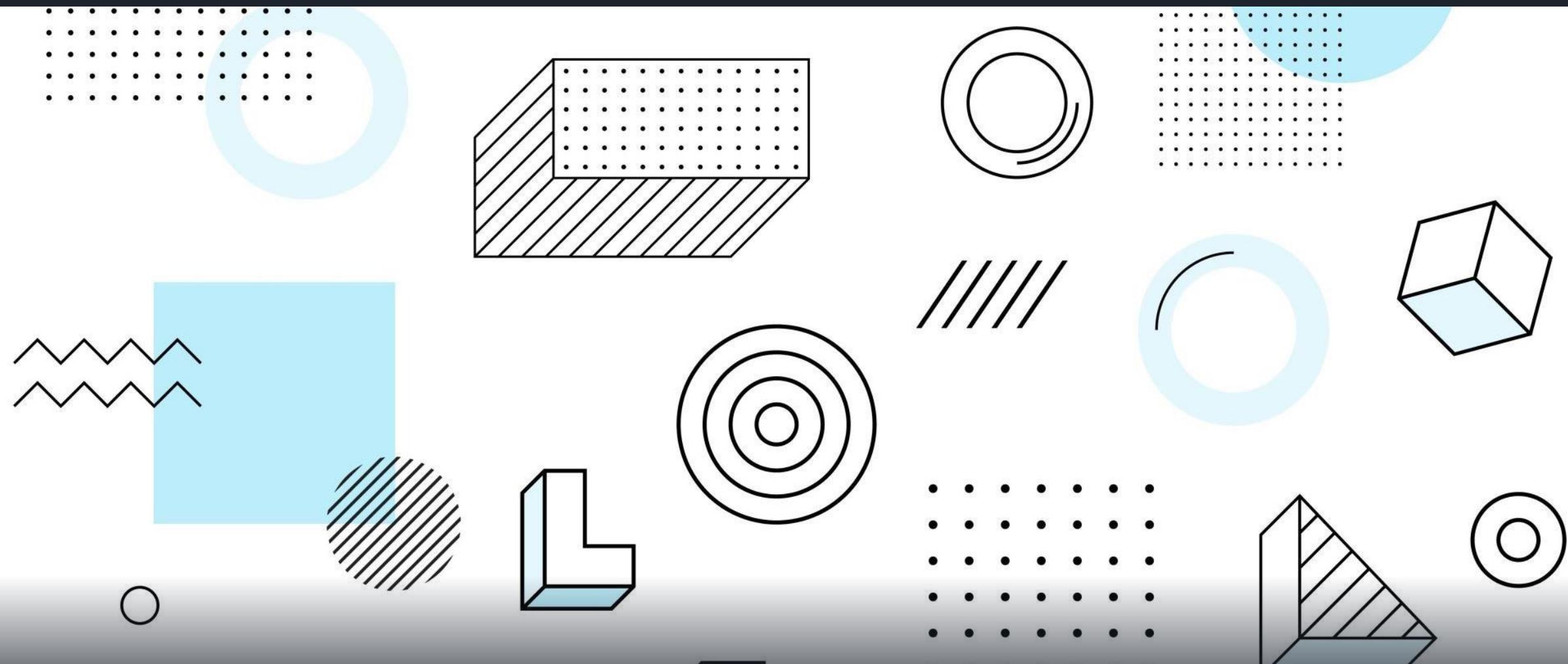
Na tej stronie mamy spis wszystkich naszych punktów wejścia (*endpoints*) wraz z akcjami umożliwiającymi wysłanie żądania (*request*) do naszej aplikacji.

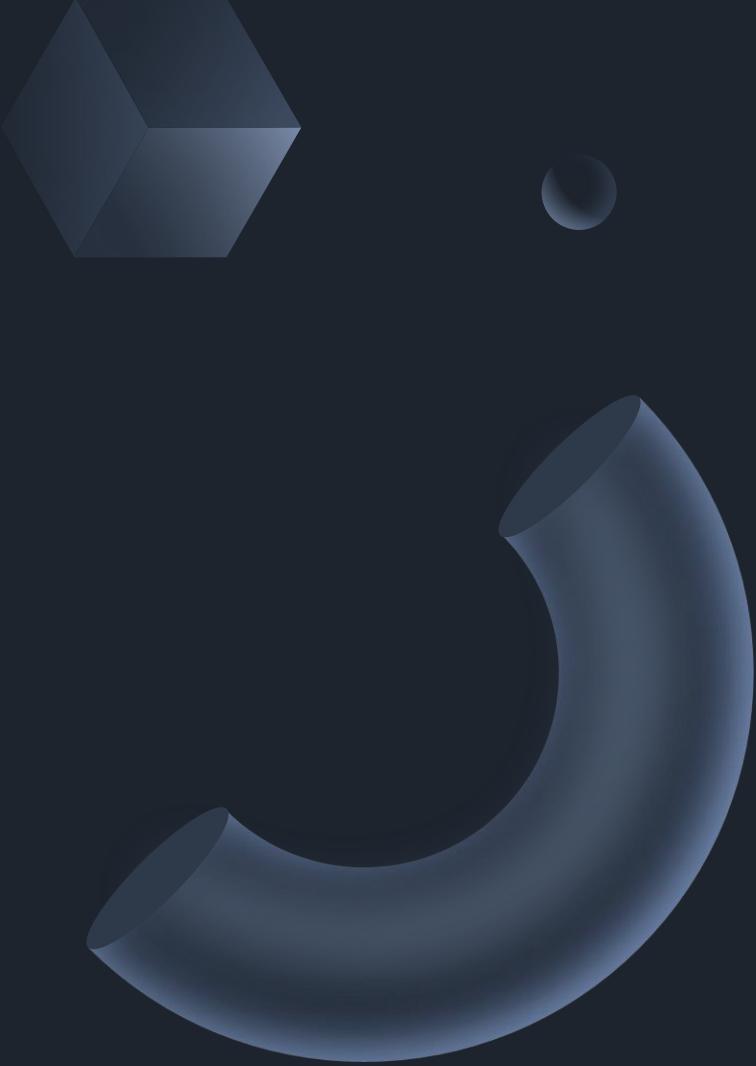
Swagger jest używany w celu usprawnienia procesu tworzenia, dokumentacji i zarządzania interfejsami API, co przyczynia się do zwiększenia efektywności pracy zespołów programistycznych i ułatwienia korzystania z API przez innych deweloperów.



# TCP, OSI, Handshake

# Inżynieria Internetu

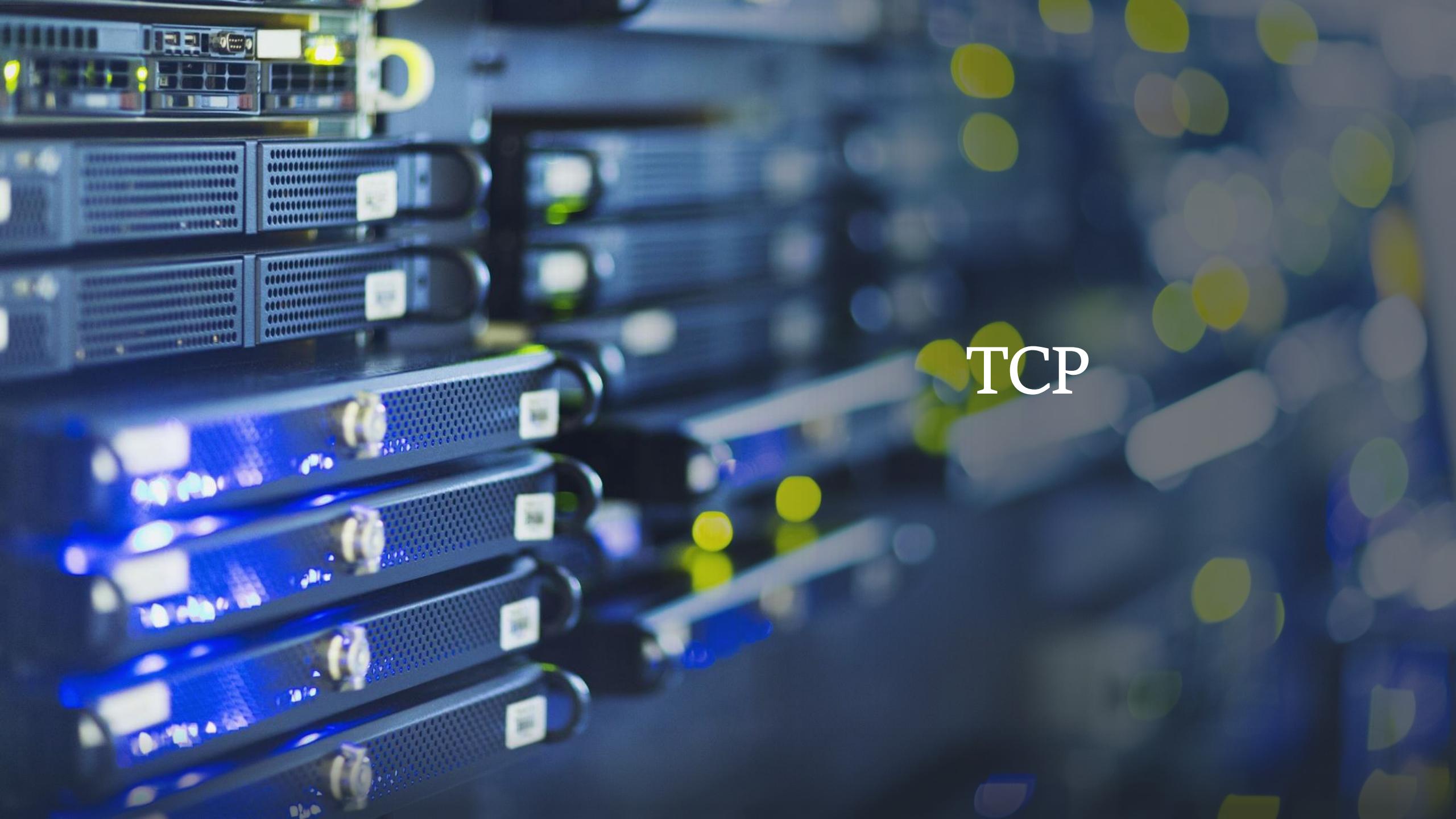


The background features abstract, semi-transparent geometric shapes in a dark blue shade. On the left, there is a large, curved, translucent shape resembling a thick pipe or a stylized letter 'G'. Above it, a smaller, darker, more solid hexagonal shape is visible. A single small, isolated dot is positioned above the hexagon.

# *Handshake*

# Definicja

- wymiana informacji między dwoma urządzeniami, która ma na celu ustalenie wspólnych parametrów transmisji danych, takich jak np. szybkość transmisji.
- Analogiczny proces wykorzystywany jest też w niektórych programowych protokołach komunikacyjnych (np. TCP) do rozpoczęcia sesji i ustalenia jej parametrów



TCP

# TCP- omówienie



- Połączeniowy, niezawodny, strumieniowy protokół komunikacyjny stosowany do przesyłania danych między procesami uruchomionymi na różnych maszynach, będący częścią szeroko wykorzystywanego obecnie stosu TCP/IP.
- Protokół sterowania transmisją operuje w warstwie transportowej modelu OSI.

# Charakterystyka działania

- TCP jest protokołem działającym w trybie klient-serwer. Serwer oczekuje na nawiązanie połączenia na określonym porcie. Klient inicjuje połączenie do serwera.
- W przeciwieństwie do UDP, TCP gwarantuje wyższym warstwom komunikacyjnym dostarczenie wszystkich pakietów w całości, z zachowaniem kolejności i bez duplikatów.
- Zapewnia to wiarygodne połączenie kosztem większego narzułu w postaci nagłówka i większej liczby przesyłanych pakietów.



# Nawiązanie połączenia

W protokole sterowania transmisją do nawiązania połączenia między dwoma hostami stosowana jest procedura nazwana three-way handshake.

Host A wysyła do hosta B segment SYN wraz z informacją o dolnej wartości numerów sekwencyjnych używanych do numerowania segmentów wysyłanych przez host A

Host B, po otrzymaniu segmentu SYN, przechodzi w stan SYN-RECEIVED i, jeżeli również chce nawiązać połączenie, wysyła hostowi A segment SYN z informacją o dolnej wartości numerów sekwencyjnych używanych do numerowania segmentów wysyłanych przez host B

Host A po odebraniu segmentów SYN i ACK od hosta B przechodzi w stan ESTABLISHED i wysyła do niego segment ACK potwierdzający odebranie segmentu SYN (numer sekwencji ustawiony na 301).

Host B odbiera segment ACK i przechodzi w stan ESTABLISHED.

Host A może teraz rozpoczęć przesyłanie danych.

# Transmisja danych

- W celu weryfikacji wysyłki i odbioru TCP używa sum kontrolnych i sekwencyjnych numerów pakietów. Odbiorca potwierdza otrzymanie pakietów o określonych numerach sekwencyjnych ustawiając flagę ACK. Brakujące pakiety są retransmitowane.

# Zakończenie połączenia

- Prawidłowe zakończenie połączenia może być zainicjowane przez dowolną stronę. Polega ono na wysłaniu pakietu z ustawioną flagą FIN (finished). Pakiet taki wymaga potwierdzenia flagą ACK. Najczęściej po otrzymaniu pakietu z flagą FIN, druga strona również kończy komunikację wysyłając pakiet z flagami FIN i ACK





# OSI

# Co to jest OSI?



standard zdefiniowany przez ISO oraz ITU-T opisujący strukturę komunikacji w sieci komputerowej.



Model ISO OSI RM jest traktowany jako model odniesienia (wzorzec) dla większości rodzin protokołów komunikacyjnych. Podstawowym założeniem modelu jest podział systemów sieciowych na 7 warstw (ang. layers) współpracujących ze sobą w ścisłe określony sposób

# Kapsułkowanie danych

- Model OSI opisuje drogę danych od aplikacji w systemie jednej stacji roboczej do aplikacji w systemie drugiej. Przed wysłaniem dane wraz z przekazywaniem do niższych warstw sieci zmieniają swój format, co nosi nazwę procesu kapsułkowania

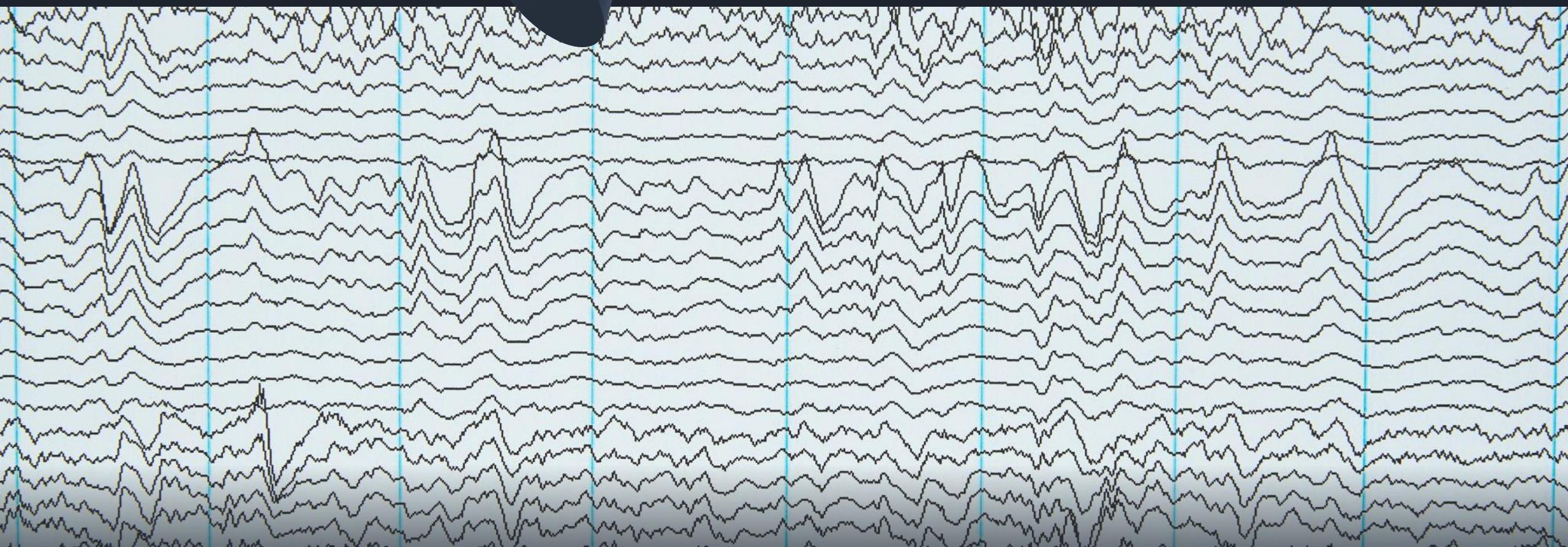
# Warstwa 7

- Warstwa aplikacji jest warstwą najwyższą, zajmuje się specyfikacją interfejsu, który wykorzystują aplikacje do przesyłania danych do sieci . W przypadku sieci komputerowych aplikacje są zwykle procesami uruchomionymi na odległych hostach. Interfejs udostępniający programistom usługi dostarczane przez warstwę aplikacji opiera się na obiektach nazywanych gniazdami



# Warstwa 6

- Podczas ruchu w dół zadaniem warstwy prezentacji jest przetworzenie danych od aplikacji do postaci kanonicznej zgodnej ze specyfikacją OSI-RM, dzięki czemu niższe warstwy zawsze otrzymują dane w tym samym formacie. Kiedy informacje płyną w góre, warstwa prezentacji tłumaczy format otrzymywanych danych na zgodny z wewnętrzną reprezentacją systemu docelowego



# Warstwa 5

- Warstwa sesji otrzymuje od różnych aplikacji dane, które muszą zostać odpowiednio zsynchronizowane. Synchronizacja występuje między warstwami sesji systemu nadawcy i odbiorcy. Warstwa sesji „wie”, która aplikacja łączy się z którą, dzięki czemu może zapewnić właściwy kierunek przepływu danych – nadzoruje połączenie.



# Warstwa 4

- Warstwa transportowa segmentuje dane oraz składa je w strumień. Warstwa ta zapewnia całościowe połączenie między stacjami: źródłową oraz docelową, które obejmuje całą drogę transmisji. Następuje tutaj podział danych na części, które są kolejno indeksowane i wysyłane do docelowej stacji. Na poziomie tej warstwy do transmisji danych wykorzystuje się dwa protokoły TCP

# Warstwa 3

- Warstwa sieciowa jako jedyna dysponuje wiedzą dotyczącą fizycznej topologii sieci. Rozpoznaje, jakie drogi łączą poszczególne komputery (trasowanie) i decyduje, ile informacji należy przesłać jednym z połączeń, a ile innym. Jeżeli danych do przesłania jest zbyt wiele, to warstwa sieciowa po prostu je ignoruje



# Warstwa 2

- Warstwa łącza danych jest czasami nazywana warstwą liniową lub kanałową. Ma ona nadzorować jakość przekazywanych informacji. Nadzór ten dotyczy wyłącznie warstwy niższej. Warstwa łącza danych ma możliwość zmiany parametrów pracy warstwy fizycznej, tak aby obniżyć liczbę pojawiających się podczas przekazu błędów. Zajmuje się pakowaniem danych w ramki i wysyłaniem do warstwy fizycznej. Rozpoznaje błędy związane z gubieniem pakietów oraz uszkodzeniem ramek i zajmuje się ich naprawą.

# Skład ramki



Ramka danych przeważnie składa się z:

ID odbiorcy – najczęściej adres MAC stacji docelowej lub bramy domyślnej,

ID nadawcy – najczęściej adres MAC stacji źródłowej,

Informacja sterująca – zawiera dane o typie ramki, trasowaniu, segmentacji itp.,

CRC (ang. Cyclic Redundancy Check) – kod kontrolny cyklicznej – odpowiada za korekcję błędów i weryfikację poprawności danych otrzymywanych przez stację docelową.

# Warstwa 1

Fundamentem, na którym zbudowany jest model referencyjny OSI, jest jego warstwa fizyczna. Określa ona wszystkie składniki sieci niezbędne do obsługi elektrycznego, optycznego, radiowego wysyłania i odbierania sygnałów. Warstwa fizyczna składa się z czterech obszarów funkcjonalnych:

mechanicznego,

elektrycznego,

funkcjonalnego,

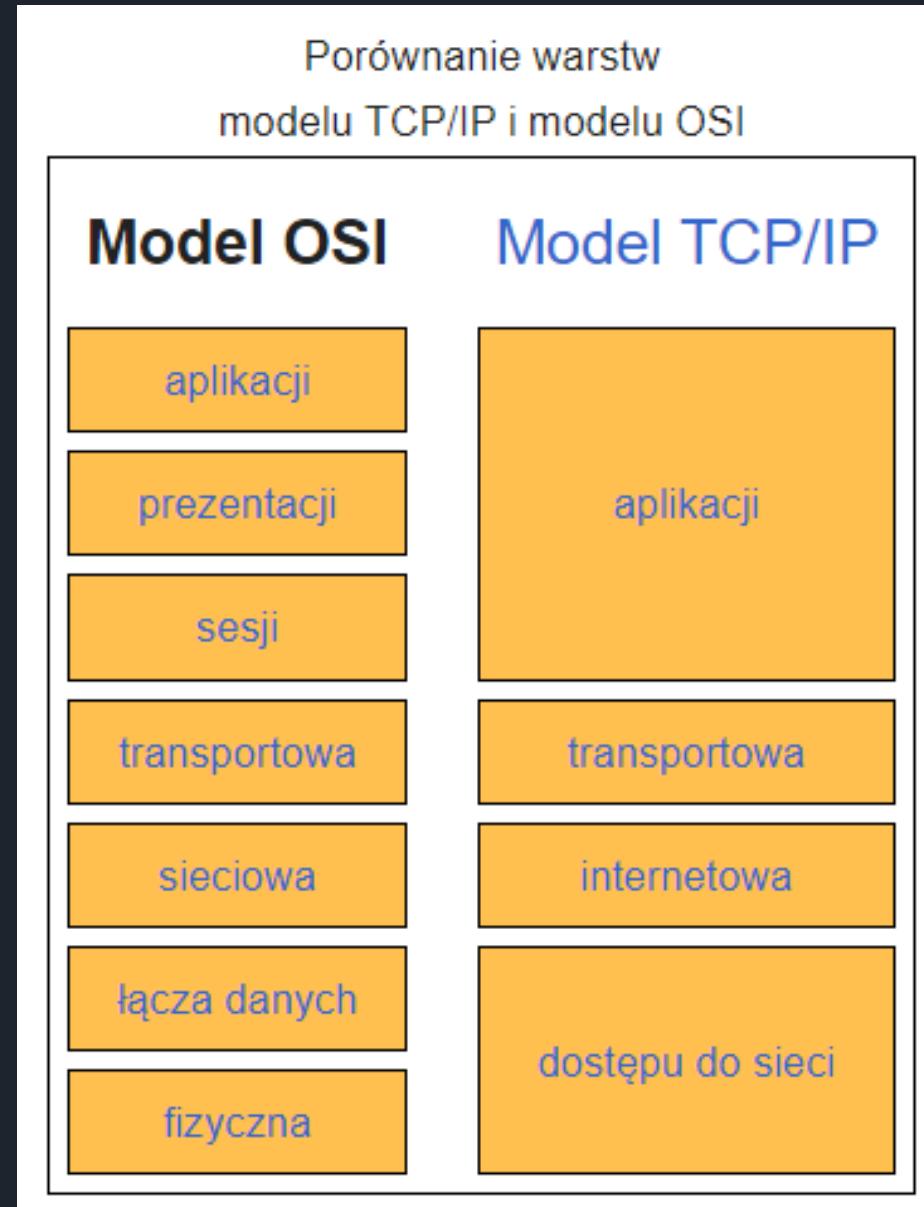
proceduralnego.

# Warstwa 1

- Warstwa fizyczna przesyła i odbiera sygnały zaadresowane dla wszystkich protokołów jej stosu oraz aplikacji, które je wykorzystują. Musi ona więc wykonywać kilka istotnych funkcji – w szczególności:
- Aby móc nadawać dane, musi ona:
- zamieniać dane znajdujące się w ramkach na strumienie binarne,
- wykonywać taką metodę dostępu do nośnika, jakiej żąda warstwa łącza danych,
- przesyłać ramki danych szeregowo (czyli bit po bicie) w postaci strumieni binarnych.

# OSI a TCP/IP

- Warstwa aplikacji zawiera w sobie warstwę aplikacji, prezentacji i większość z warstwy sesji modelu OSI,
- Warstwa transportu zawiera część warstwy sesji, jak i warstwy transportowej modelu OSI,
- Warstwa Internetu jest odpowiednikiem warstwy sieci modelu OSI,
- Warstwa dostępu do sieci zawiera w sobie warstwę łącza danych, jak również warstwę fizyczną modelu OSI.



# Dziękuje za uwagę

- Bartek Sacha ETI Rok 3 gr 2

