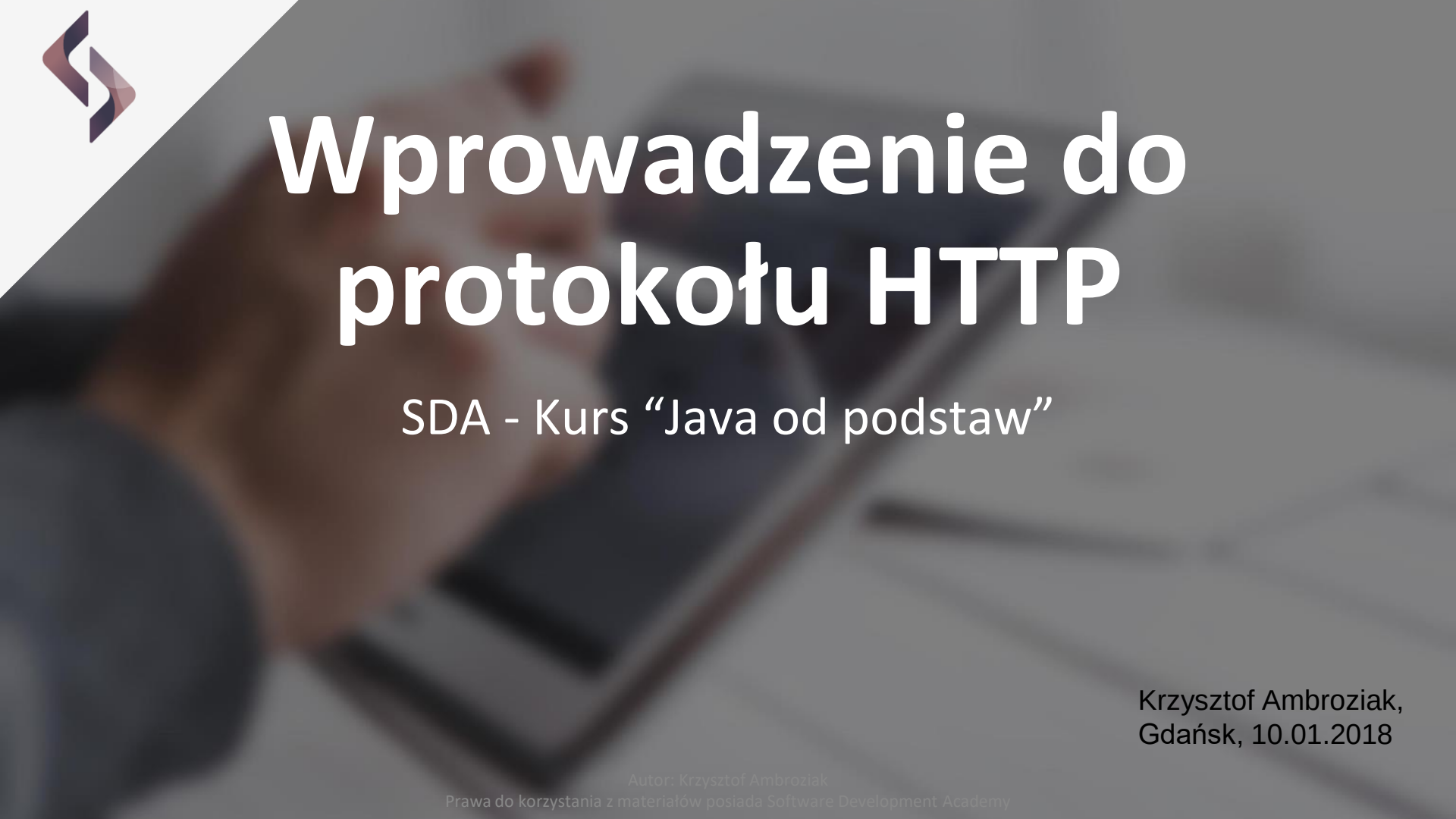


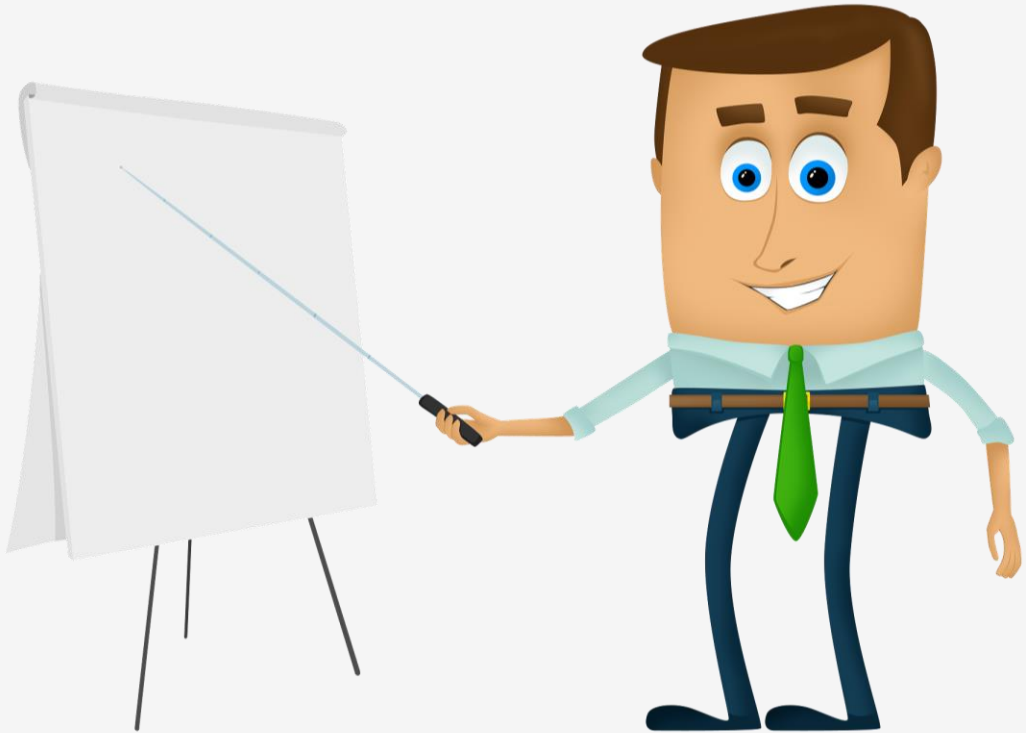
Wprowadzenie do protokołu HTTP

SDA - Kurs “Java od podstaw”

Krzysztof Ambroziak,
Gdańsk, 10.01.2018

Agenda

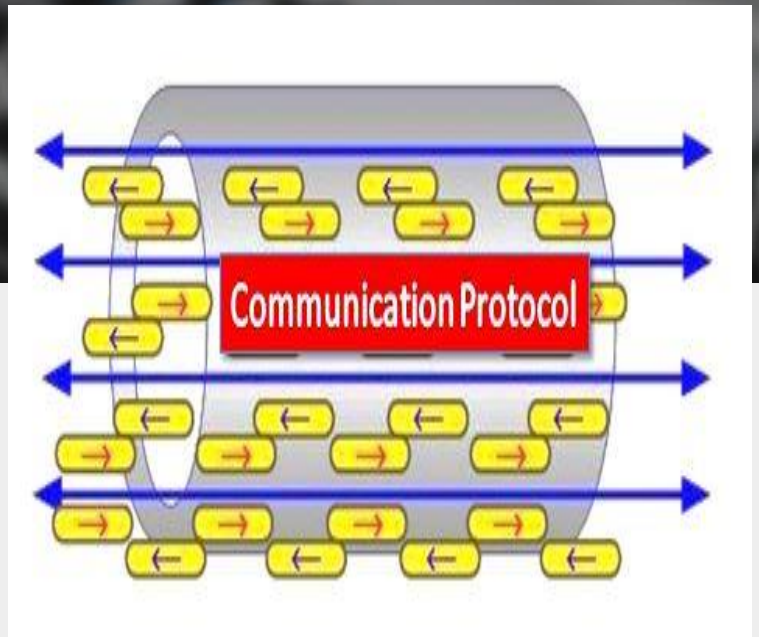
- Zarys tematu, trochę teorii
- Omówienie najważniejszych elementów HTTP
- HTTP z poziomu JAVA
- Narzędzia związane z HTTP
- Omówienie REST
- Implementacja przykładowego REST API



Co to właściwie jest protokół?

Protokół = Zbiór reguł

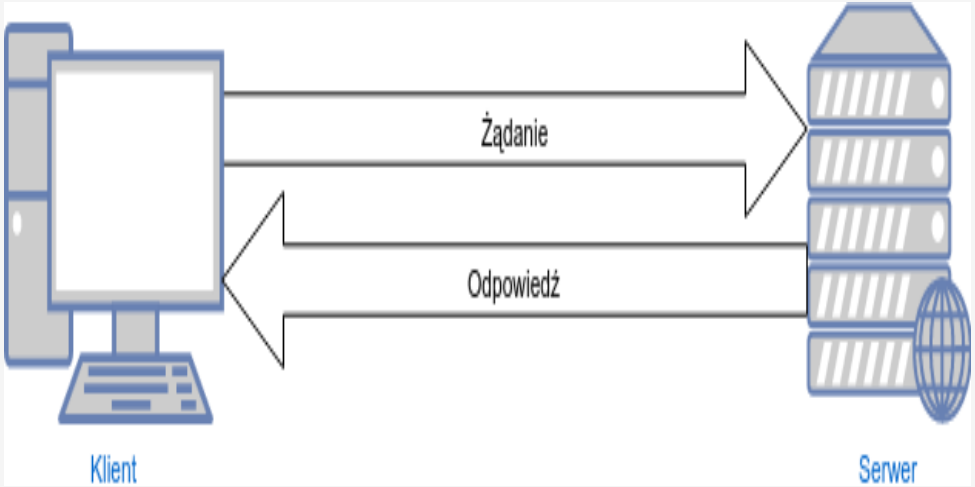
Protokół komunikacyjny to grupa wytycznych których stosowanie przez wszystkie strony komunikacji pozwala na skuteczne i efektywne przesyłanie danych



HTTP - Hypertext Transfer Protocol

Jest to protokół sieciowy służący do przesyłania dokumentów tekstowych, głównie stron internetowych (ale nie tylko) pomiędzy klientem a serwerem. Najpopularniejszym klientem HTTP jest przeglądarka internetowa ale może to być także narzędzie linii poleceń lub program napisany w JAVA

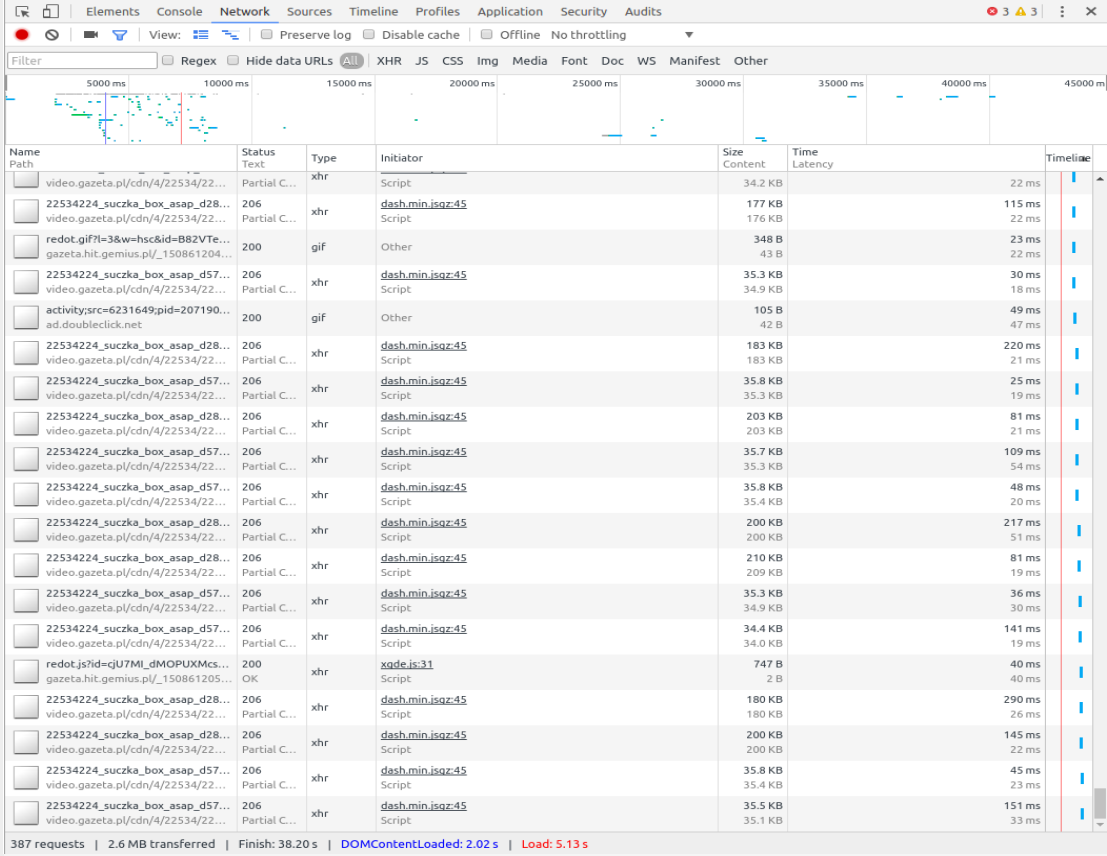
HTTP - czyli protokół klient-serwer



1. Klient wysyła wiadomość do serwera
2. Serwer odbiera i przetwarza wiadomość
3. Serwer odsyła odpowiedź
4. Klient odbiera I wyświetla odpowiedź

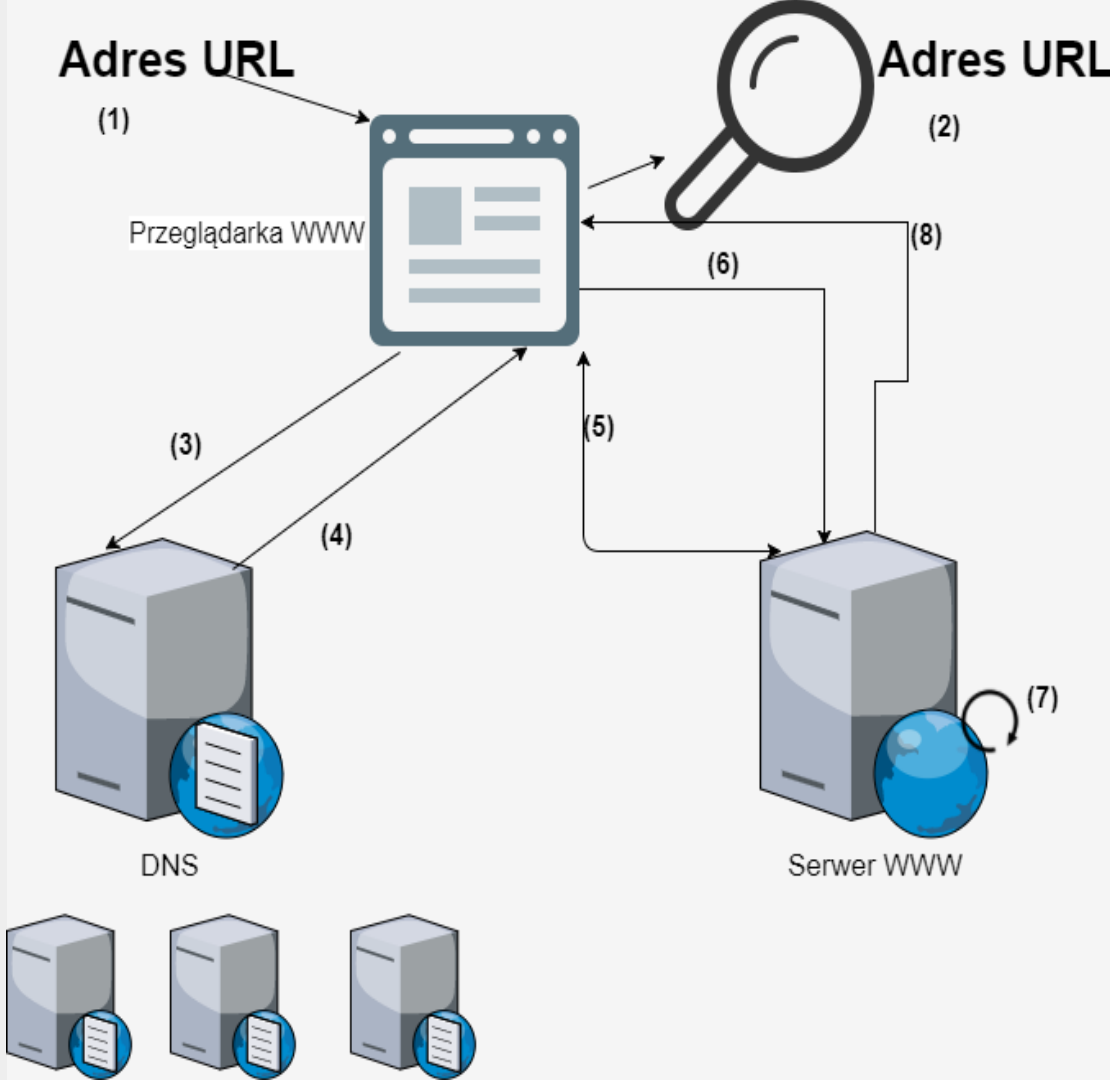
Konsola developera w CHROME

Konsola developera w zakładce sieciowej pokazuje wszystkie żądania jakie wysyła przeglądarka w celu załadowania strony



Co robi przeglądarka aby wyświetlić stronę

1. Wpisujemy adres
2. Przeglądarka parsuje go
3. Odpytuje serwer DNS o adres IP
4. Serwer DNS zwraca IP
5. Przeglądarka nawiązuje polaczenie TCP z serwerem WWW
6. Przeglądarka wysyła żądanie
7. Serwer WWW przetwarza żądanie
8. Serwer WWW odsyła odpowiedź
9. Przeglądarka wyświetla odpowiedź



Elementy HTTP

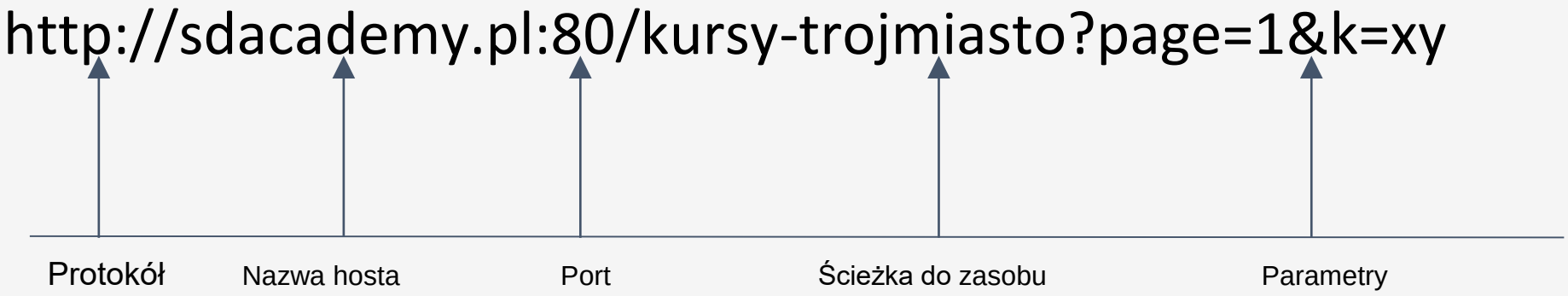
Żądanie

- Adres
- Metoda
- Nagłówki
- Opcjonalne ciało (treść)

Odpowiedź

- Nagłówki
- Ciało
- Status

Adres URL - jak to jest zbudowane



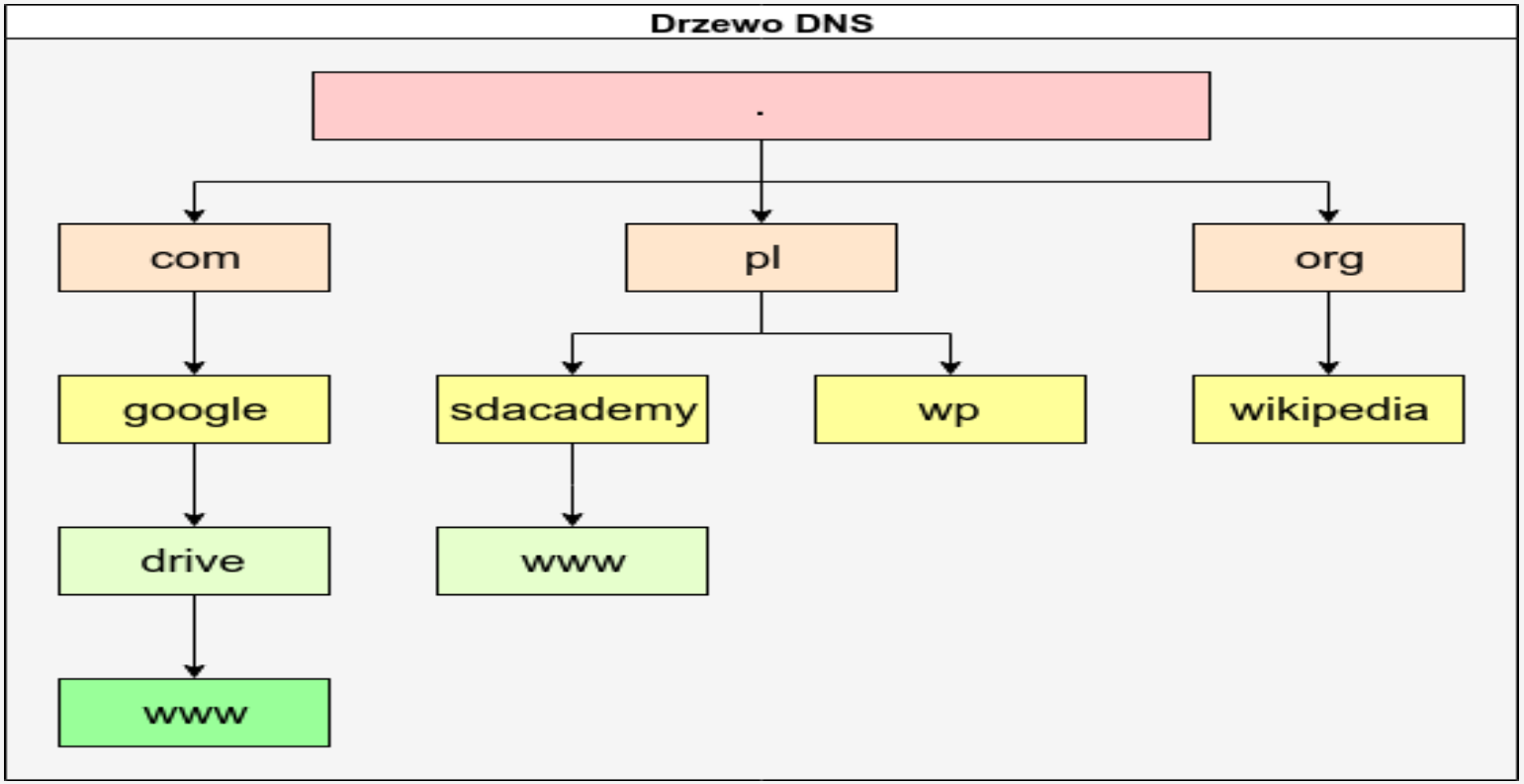
DNS - Domain Name System

- Komputery z reguły nie wykorzystują do komunikacji nazw tekstowych
- Każdy komputer działający w sieci ma przypisany numer IP
- Za tłumaczenie nazwa -> numer IP odpowiedzialny jest DNS
- Oparty jest na działaniu 13 serwerów (logicznych)
- DNS ma strukturę hierarchiczną - drzewiastą

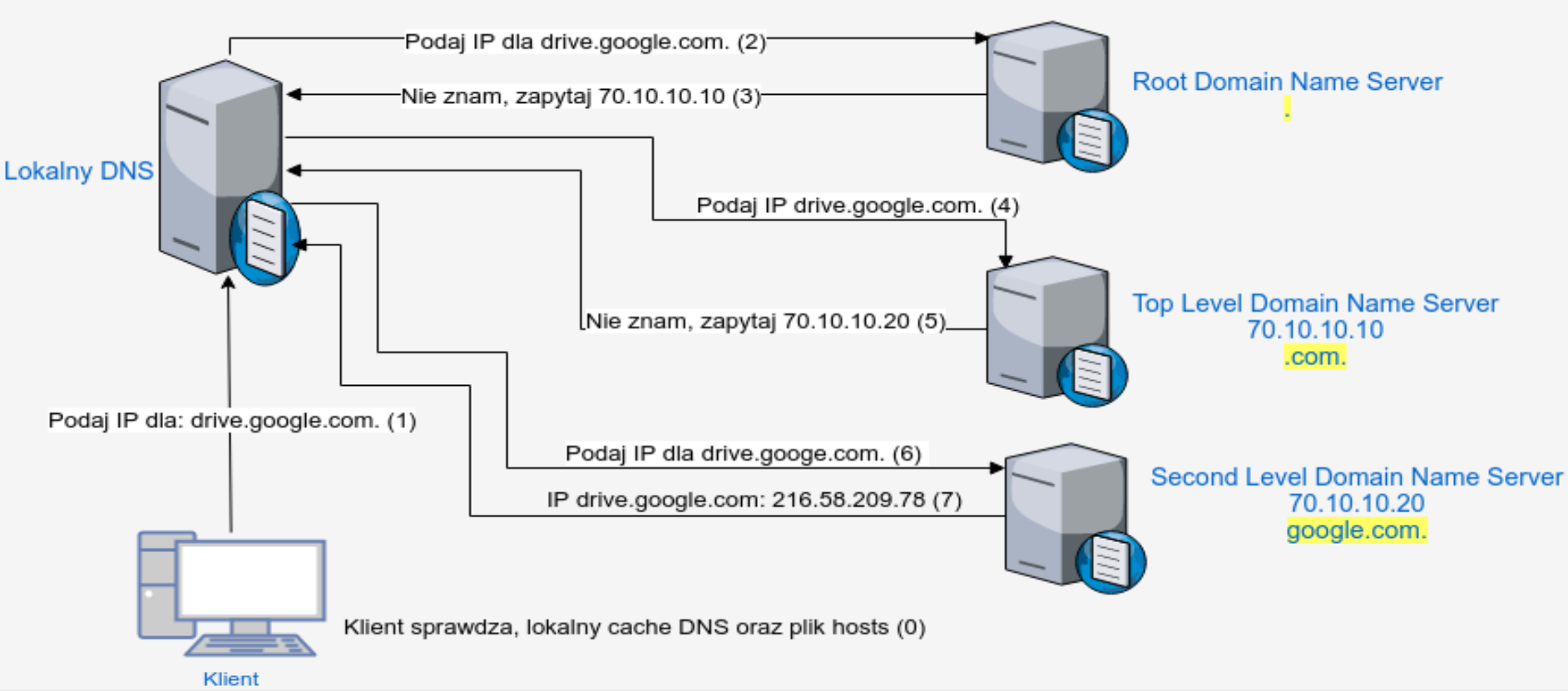
- Spójrzmy na adres:

http://sdacademy.pl.

DNS - Struktura drzewa



DNS - proces odpytania o adres



Schemat odpytania o adres drive.google.com

Adresacja IP

Adres IPv4 składa się z 4, 8-bitowych liczb, to znaczy że maksymalnym adresem jest: 255.255.255.255 (11111111. 11111111. 11111111. 11111111(2))

Istotnym sposobem podziału adresów IP jest rozdział na tzw adresy prywatne oraz adresy publiczne

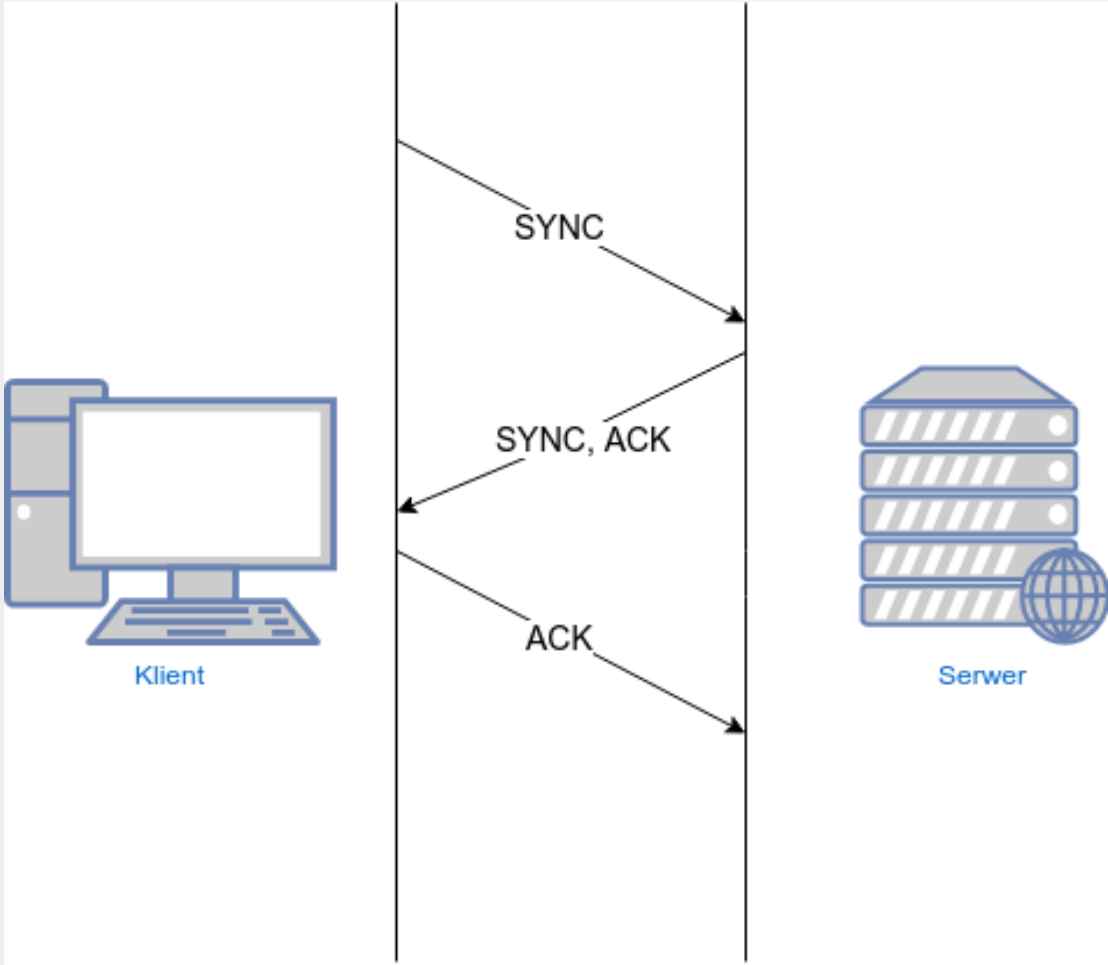
Adresy prywatne wykorzystywane są w sieciach lokalnych

Adresy publiczne w Internecie

Uruchom: ipconfig /all

Protokół TCP

- Protokół zapewniający komunikację pomiędzy procesami na zdalnych maszynach
- Posiada mechanizm zapewniające niezawodność transmisji
- Jest protokołem połączeniowym



Warstwowa natura sieci

- Warstwa dostępu do sieci – Kable urządzenia sieciowe np. Przełączniki
- Warstwa Internetu - protokół IP, routery
- Warstwa transportowa - protokół TPC
- Warstwa aplikacji - HTTP, DNS, itp..

Metody HTTP

- **GET** - Pobierz zasób z serwera
- **POST** - Wyślij dane do serwera w celu przetworzenia
- **PUT** - Zachowaj zawartość żądania na serwerze. Uaktualnij wskazany zasób
- **DELETE** - Usuń wskazany zasób z serwera
- **PATCH** - Aktualizuj część danych zasobu
- **HEAD** - Pobiera jedynie nagłówki
- **OPTIONS** - Sprawdź obsługiwane metody
- **TRACE** - Wyślij żądanie “testowe”

Metody http - podsumowanie

Metoda	Ciało żądania	Ciało odpowiedzi	Bezpieczny	Idempotent
GET	Opcjonalnie	TAK	TAK	TAK
POST	TAK	TAK	NIE	NIE
PUT	TAK	TAK	NIE	TAK
DELETE	NIE	TAK	NIE	TAK
PATCH	TAK	TAK	NIE	NIE

Bezpieczny – metoda niezmieniająca stanu(reprezentacji) zasobu
Idempotent – niezależnie ile razy metoda zostanie użyta zasób ma taki stan jak po pierwszym wywołaniu

Nagłówki żądania i odpowiedzi

- Dodatkowe informacje przesyłane od klienta do serwera lub w przeciwnym kierunku
- Mają postać klucz:wartość
- Istnieją nagłówki o zdefiniowanym znaczeniu, które powinny skutkować określonym zachowaniem strony odbierającej (klienta lub serwera)
- Programiści mogą definiować własne nagłówki oraz nadawać im znaczenia

Przykładowe nagłówki żądania

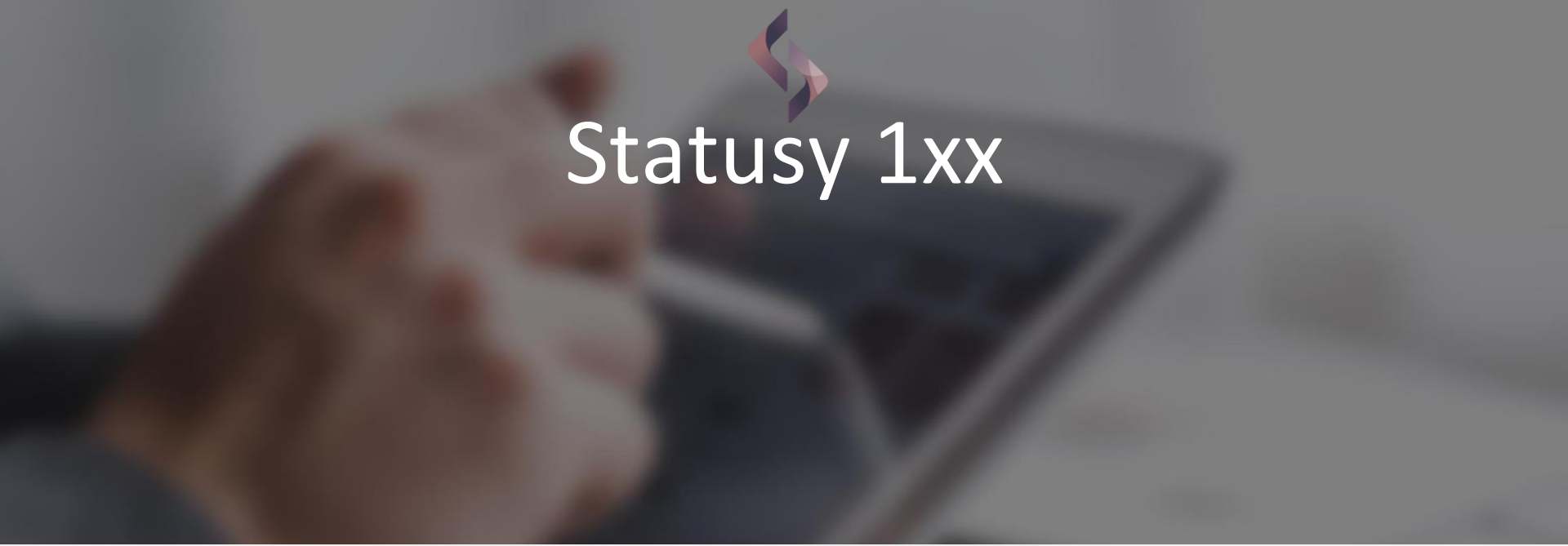
- **Accept** - określa akceptowany przez klienta format odpowiedzi. Przykład: Accept: application/json
- **Accept-Language** - akceptowany przez klienta język odpowiedzi. Przykład: Accept-Language: pl-PL
- **Host** - nazwa domenowa serwera.
- Przykład: Host:wp.pl
- **User-Agent** - identyfikator klienta. Przykład: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.101 Safari/537.36
- **Cookie** - ciasteczko związane z żądaniem
- **Referer** – pokazuje z jakiej lokalizacji przyszliśmy

Przykładowe nagłówki odpowiedzi

- **Server** - nazwa serwera który nam odpowiedział.
  Przykład: Server: Apache/2.4.10 (Unix) OpenSSL/1.0.1e-fips mod_bwlimited/1.4
- **Date** - czas wysłania odpowiedzi.
  Przykład Date: Sat, 21 Oct 2017 10:44:53 GMT
- **Content-type** - typ zwracanej odpowiedzi.
  Przykład: Content-type: text/html; charset=UTF-8
- **Set-cookie** - polecenie dla przeglądarki ustawienia wartości w ciasteczku.
- **Content-length** - wielkość odpowiedzi w bajtach
- **Location** - wskazanie adresu na który ma przekierować się przeglądarka
- Przykład: Location: http://sdacademy.pl/
- **Cache-Control, Last-Modified, Expires**

Statusy odpowiedzi

W ramach odpowiedzi serwer wysyła jej kod:
1xx - status informacyjne
2xx – Udało się! - Wszystko poszło ok
3xx - Grupa statusów przekierowań – serwer chce aby przeglądarka udała się pod inna lokalizacje
4xx – Błąd po stronie klienta – błąd w zapytaniu np. niepoprawne nagłówki
5xx – Coś poszło nie tak - grupa błędów po stronie serwera

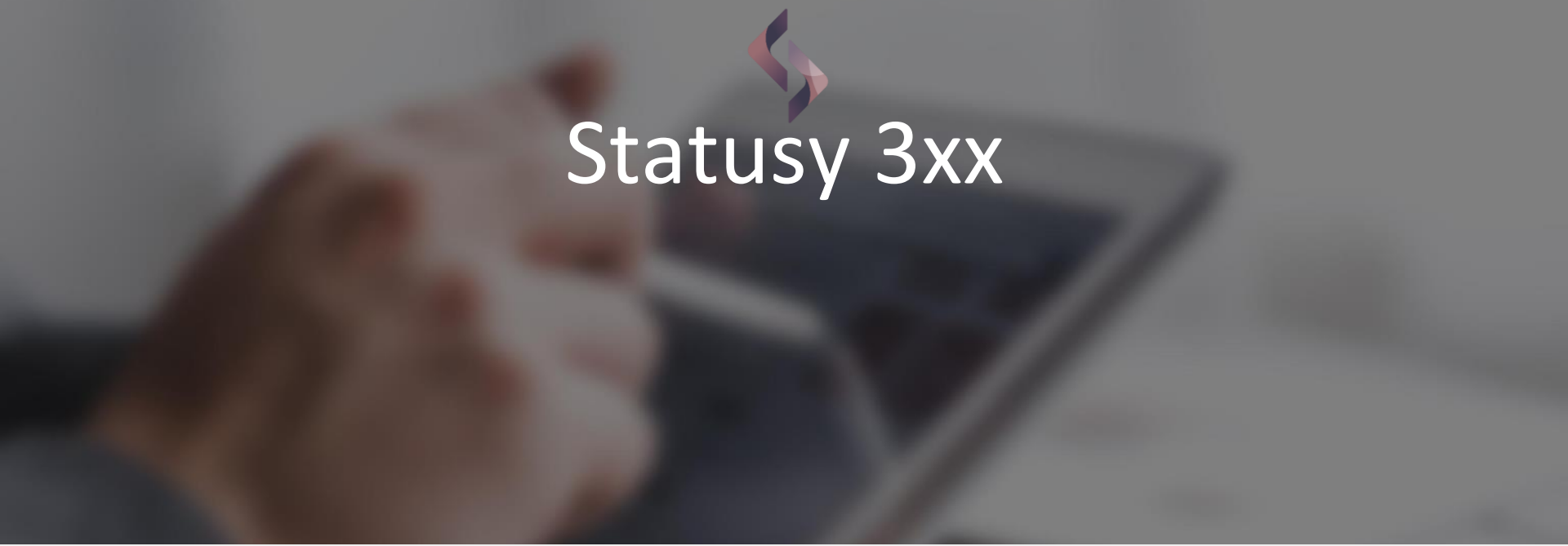


Statusy 1xx

- 100 Continue – „Prześlij resztę danych”
- 101 – Switching Protocols – przetworze to żądanie jeśli wyślesz je za pomocą protokołu z nagłówka Upgrade

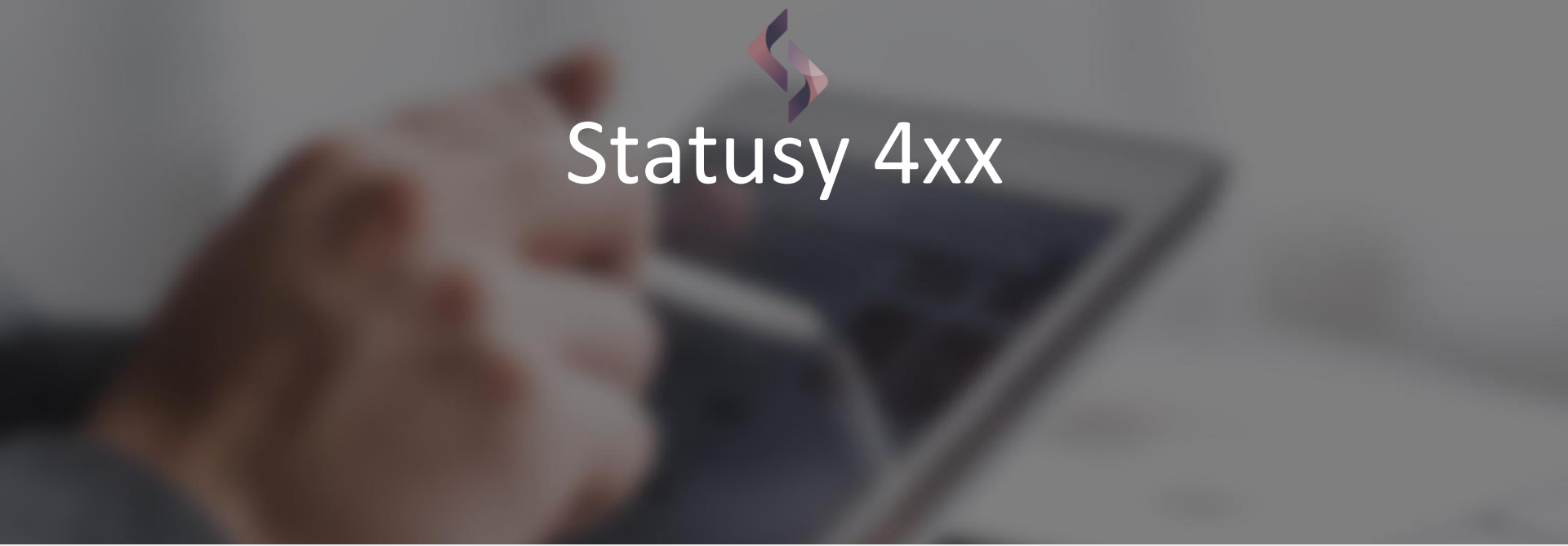
Statusy 2xx

- 200 OK - Żądanie wykonane prawidłowo
- 201 Created – Nowy zasób został utworzony
- 202 Accepted – Zrozumiałem i przyjąłem Twoje żądanie



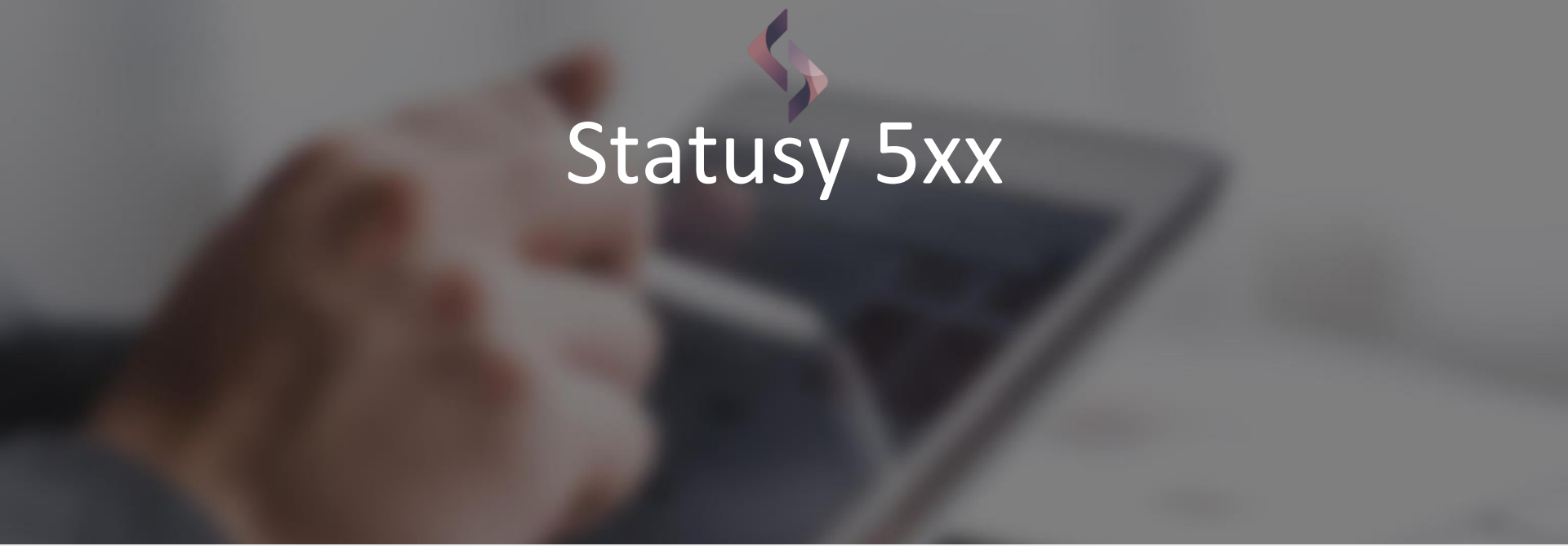
Statusy 3xx

- 301 Moved Permanently – Kiedyś był tutaj zasób ale już go nie ma i nie będzie
- 302 – Moved Temporarily – Zasób chwilowo nieobecny pod tą lokalizacją
- 303 See other – Zasób którego szukasz nie znajduje się tutaj ale sprawdź tam



Statusy 4xx

- 400 Bad Reqest – Żądanie nie spodobało się serwerowi prawdopodobnie czegoś w nim brakuje
- 404 – Not Found – Wskazany zasób nie istnieje
- 405 – Method Not Allowed – Zła metoda do wywołania tego adresu
- 415 – Unsupported Media Type – Typ zawartości żądania nie jest obsługiwany



Statusy 5xx

- 500 Internal Server Error – Coś poszło nie tak podczas obsługi żądania
- 503 Service Unavailable – Serwer nie był w stanie odpowiedzieć z powodu przeciążenia
- 504 Gateway Timeout – Przesyłanie danych pomiędzy serwerami trwało zbyt długo

Apache HTTP Client

Popularna biblioteka służąca do komunikacji z wykorzystaniem HTTP

```
<dependency>
   <groupId>org.apache.httpcomponents</groupId>
   <artifactId>httpclient</artifactId>
   <version>4.5.3</version>
</dependency>
```

- Pobierzmy jakąś stronę
- Wyślijmy przykładowe dane za pomocą POST'a
- Spróbujmy usunąć stronę SDA :)
- Potestujmy działanie innych metod http
- Przyjrzyjmy się formatowi JSON na stronie http://json.org/ i pobierzmy informacje o Polsce w formacie JSON z https://restcountries.eu/

GSON

Popularna biblioteka od Googla służąca do pracy z formatem JSON

```
<dependency>
  <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.2</version>
</dependency>
```

Wykorzystując HTTP Clienta i GSON'a pobierzmy informacje o Polsce i napiszmy zdanie:

„Polska ma …. ludności, graniczy z …… a po włosku to: .. ”

```
JsonParser jsonParser = new
JsonParser();
JsonElement parse =
jsonParser.parse(polandInJson);
JsonArray asJsonArray =
parse.getAsJsonArray();
String name =
asJsonArray.get(0).getAsJsonObject().get
("name").getAsString();
System.out.println( name );
```

GSON – budowanie JSON'a

Za pomocą GSON'a można również stworzyć text w formacie JSON, po to aby np. wysłać go potem na serwer.

Stwórzmy obiekt typu Student a potem wyślijmy go na:

https://protected-cove-60658.herokuapp.com/students

POSTMAN

Wtyczka do przeglądarki CHROME

Pozwala w łatwy sposób budować żądania HTTP z wykorzystaniem metod nagłówków, ciał

Zapamiętuje wykonowane requesty dzięki czemu łatwo je ponawiać

Spróbujmy pobawić się jakimś publicznym API za pomocą POSTMANA np. https://restcountries.eu/

Narzędzia diagnostyczne

PING

Służy do sprawdzania połączenia do zdalnego hosta

Odpytuje serwery DNS – służy często do sprawdzania numeru IP

Mierzy czasy oczekiwania na odpowiedź

Wykonajmy:

Ping sdacademy.pl

Ping –c 3 wp.pl

Ping –w 1 192.168.100.100

TRACEROUTE (TRACERT)

Służy do sprawdzania trasy jaką przemierzają pakiety w drodze do celu

Może służyć do diagnozowania problemów pomiędzy konkretnymi przeskokami

W praktyce, ciekawe narzędzie

Np. traceroute gogle.com

Pierwsza linia żądania http. Narzędzie telnet

- Telnet – powstał w 1972 roku, jest narzędziem służącym do łączenia się do zdalnych maszyn, w praktyce często wykorzystywany do sprawdzania możliwości połączenia miedzy maszynami w sieci korporacyjnej (pakiet cygwina: inetutils)
- Wykonajmy żądanie http z linii poleceń!
- telnet sdacademy.pl 80

Klient HTTP z linii poleceń

cURL

Narzędzie do przesyłania danych z wykorzystaniem adresów URL, w praktyce wykorzystywane do interakcj z REST API, wysyłania POST'ow
Wywołajmy:

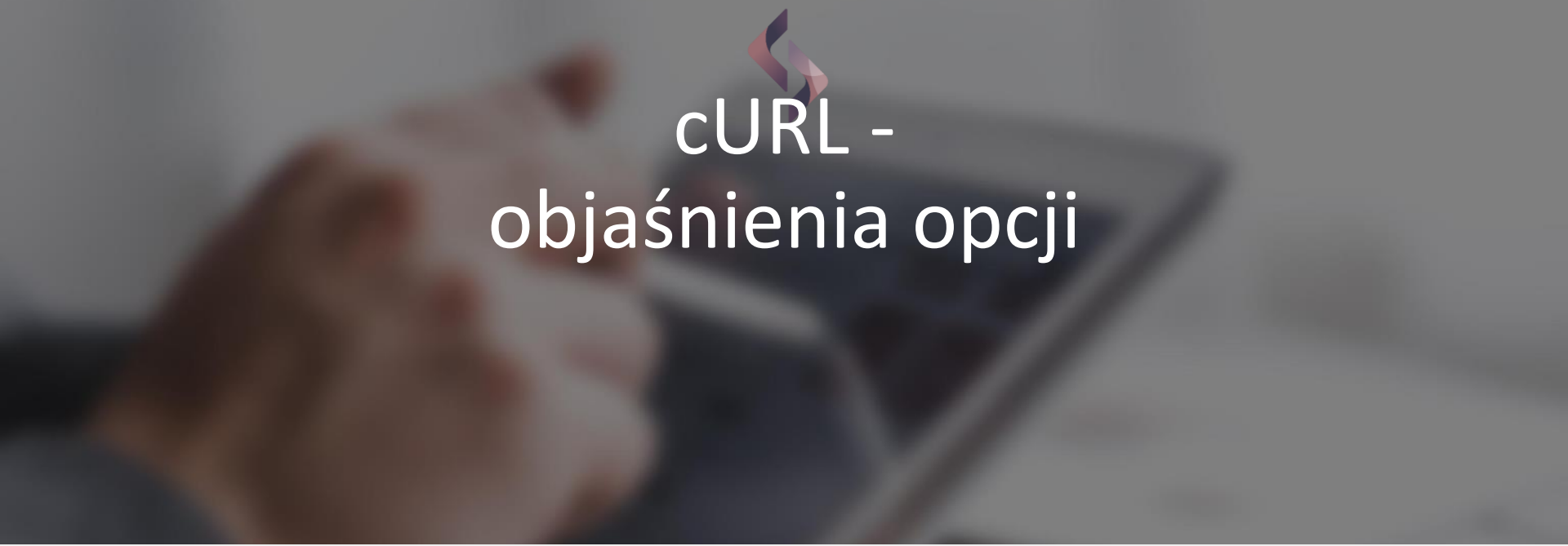
```
curl www.sdacademy.pl
curl -X GET www.sdacademy.pl
curl -X GET -i www.sdacademy.pl
curl -X GET -i -L www.sdacademy.pl -o
    output.txt
curl -X POST www.wp.pl
```

wGET

Narzędzie dużo prostsze od cURL

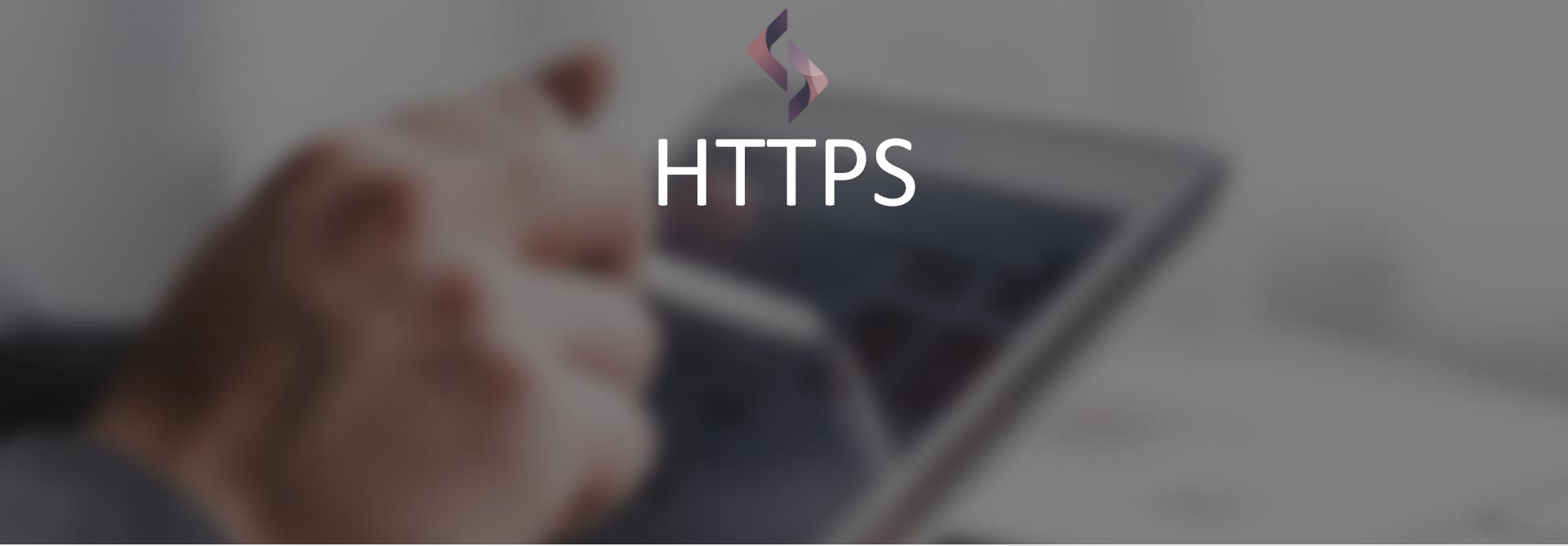
Umożliwia wykonywanie tylko GET'ow

W praktyce wykorzystywane do pobierania z poziomu konsoli np. paczek zip, instalek oprogramowania
wget -O "nazwa" adres - pobranie zasobu i zapisanie pod wskazaną nazwą
--input list_of_url.txt - pobranie zasobów ze wszystkich adresów umieszczonych w plikach
wget -r -H --convert-links -l2 --user-agent=Mozilla onet.pl



cURL - objaśnienia opcji

W poprzednim ćwiczeniu wykorzystaliśmy następujące opcje:
-X POST - metoda http
-i - wyświetla nagłówki odpowiedzi
-L - włącza podążanie za przekierowaniem



HTTPS

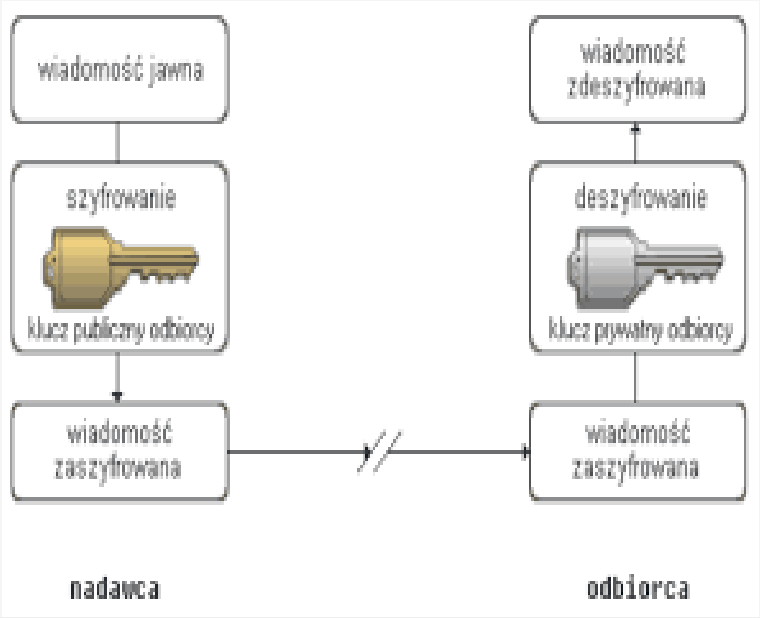
HTTP nie jest protokołem bezpiecznym - wysyła informacje czystym tekstem w sposób niezaszyfrowany

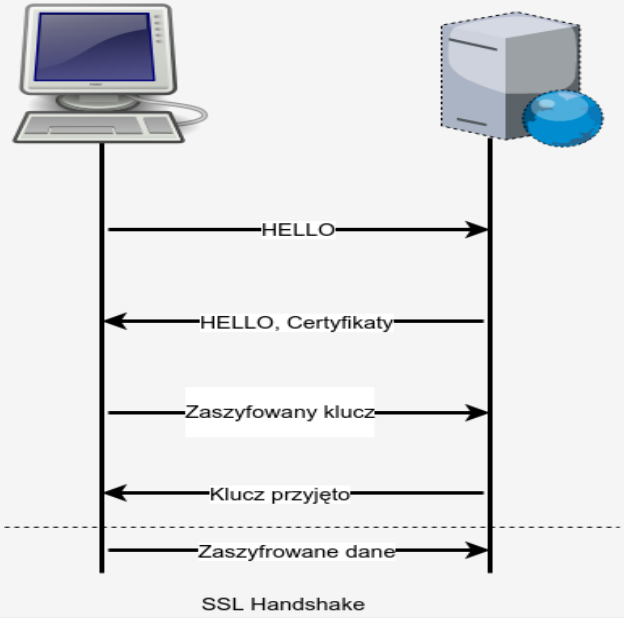
Jest to wystarczające do wyświetlania prostych stron lub nawet złożonych serwisów tak długo jak nie wyświetlają one lub nie wysyłamy do nich poufnych danych

Rozwiązaniem jest zastosowanie rozszeżenia HTTP**S** (port 443)

HTTPS

- HTTPS wykorzystuje mechanizmy kryptograficzne aby zapewnić poufność przesyłanych informacji, zarówno algorytmy symetryczne jak i asymetryczne także certyfikaty oraz PKI





Narzędzia - podsumowanie

Poznaliśmy jak do tej pory kilka narzędzi:

- ping - diagnostyka, dobry do pozyskiwania numerów IP
- telnet - diagnostyka, możliwa komunikacja ze zdalną maszyną
- cURL - potężne narzędzie do pracy z HTTP (i nie tylko)
- wGet - pobieranie zasobów z sieci
- whireshark - “okienkowe” podsłuchiwanie ruchu
- tcpdump - nasłuchiwanie ruchu z poziomu linii poleceń
- POSTMAN - klient http, REST API

REST - Representational State Transfer

Styl projektowania aplikacji, w której klient odpytuje konkretny zasób, a serwer odpowiada nie całą stroną, a tylko i wyłącznie reprezentacją zasobu (lub informacją o nim) sformatowaną w odpowiedni sposób np. JSON.

REST dzieli serwer na kontrolery zasobów, pod które trafiają odpowiednie zapytania. Każdy z kontrolerów ma zaimplementowany zestaw metod do zarządzania jego stanem. W zależności od otrzymanej autoryzacji (użytkownik może nie mieć uprawnień do odczytania lub edycji zasobu) wykonuje, bądź nie zadaną czynność.

Działanie	Instrukcja SQL	Metoda HTTP
Create	Insert	POST
Read	Select	GET
Update	Update	PUT
Delete	Delete	DELETE

Projekt przykładowego API

- Założenia:
  - Mamy następujące zasoby:
    - Student, Szkoła Programowania, Kurs, Wykład
- Zaprojektujmy REST API które umożliwi:
  - Pełnego CRUD’a dla Studentów i Szkoły
  - Dodawanie kursów do szkoły programowania
  - Listowanie kursów w danej szkole, z wyszukiwaniem po nazwie i roku startu
  - Dodawania wykładów do kursów
  - Listowanie wykładów w ramach kursu w danej szkole
  - Zapisywanie studentów na kurs w danej szkole

Projekt przykładowego API CD.

- CRUD dla Studenta
  - GET /students – wyświetla wszystkich studentów
  - GET /students/{id}
  - POST /students – dodaje studenta
  - DELTE /students/{id} – usuwa studenta o podanym ID
  - PUT /students/{id} – aktualizuje studenta o podanym ID
- CRUD dla Szkoły
  - GET /schools – wyświetla wszystkie szkoły
  - GET /schools /{id} – wyświetla szkołę o podanym ID
  - POST /schools – dodaje szkołę
  - DELTE /schools/{id} – usuwa studenta o podanym ID
  - PUT /schools/{id} – aktualizuje szkołę o podanym ID

Projekt przykładowego API CD.

- Dodawanie kursów do szkoły programowania
  - POST /schools/{schoolId}/courses
- Listowanie kursów w danej szkole
  - GET /schools/{schoolId}/courses?startYear=123&name=„Java”
- Dodawania wykładów do kursów
  - POST /schools/{schoolId}/courses/{courseId}/lectures
- Listowanie wykładów w ramach kursu w danej szkole
  - GET /schools/{schoolId}/courses/{courseId}/lectures
- Zapisywanie studentów na kurs w danej szkole
  - POST /schools/{schoolId}/courses/{courseId}/users

Hierarchia REST

Level 3
- HATEOAS
- Samodokumentujące się API

Level 2
- Wiele zasobów
- Wiele czasowników (metod HTTP)

Level 1
- Wiele zasobów
- Jeden czasownik (POST)

Level 0
- Jeden zasób
- Jeden czasownik (POST)

HATEOAS

- Najwyższy poziom rozwoju API REST
- Samodokumentujące się
- Nie trzeba definiować adresów endpointów „na sztywno”, klient sam może pobrać informacje z ciała odpowiedzi


```
{
   "Department":"Enforcement",
   "Id":"123",
   "Links":[
      {
         "Rel":"GetDetails",
         "Url":"/api/employees/56789"
      },
      {
         "Rel":"Fire",
         "Url":"/api/employees/56789"
      }
   ],
   "Name":"John Q Law"
}
```

Jak to się implementuje w praktyce

- W praktyce REST API w Javie implementuje się bardzo często z wykorzystaniem specyfikacji JAX-RS
- Najpopularniejszymi implementacjami są RestEasy i Jersey
- Specyfikacja określa zbiór adnotacji których stosowanie pozwala na czytanie danych z żądań http oraz generowanie odpowiedzi

- @Path – definiuje ścieżkę do zasobu np. @Path(„coures/{courseId}/leacture”)
- @PathParam – czyta fragment ścieżki będący zmienną np. @PathParam(„courseId”) Integer courseId
- @Produces(MediaType.APPLICATION_JSON) – określa content-type odpowiedzi jako „application/json”
- @Consumes(MediaType.TEXT_PLAIN) – określa akceptowalny content-type żądania jako text
- @POST, @GET… - określają obsługiwane metody http
- @QueryParam(„xyz”) – odczytuje ze ścieżki żądania wartość parametru xyz
- javax.ws.rs.core.Response – klasa odpowiedzi z endpointu