

# Sprawozdanie **Lista nr 5**

| Przedmiot  | Technologie sieciowe   |
|------------|------------------------|
| Prowadzący | Mgr inż. Dominik Bojko |
| Autor      | Maciej Bazela          |
| Indeks     | 261743                 |
| Grupa      | Czw. 15:15-16:55       |
| Kod grupy  | K03-76c                |

Sprawozdanie znajduje się w repozytorium na moim [githubie](#).

## 1. Wymagania

Na podstawie załączonego skryptu należy:

- uruchomić go i zastanowić się jak działa serwer,
- połączyć się do niego przy użyciu przeglądarki internetowej,

Napisać własny serwer oraz:

- wysyłać do klienta nagłówki jego żądania
- obsługiwać żądania klienta do prostego tekstowego serwisu WWW (kilka statycznych stron z wzajemnymi odwołaniami) zapisanego w pewnym katalogu dysku lokalnego komputera na którym uruchomiony jest skrypt serwera,
- przechwycić przesyłane/przychodzące komunikaty za pomocą analizatora sieciowego (F12)

## 1.1 Środowisko

Do napisania serwera użyłem kombinacji języka programowania HTML oraz lekko gorszego Julia.

## 2. Analiza skryptu `server.pl`

```
use HTTP::Daemon;
use HTTP::Status;
#use IO::File;

my $d = HTTP::Daemon->new(
    LocalAddr => 'lukim',
    LocalPort => 4321,
)|| die;

print "Please contact me at: <URL:", $d->url, ">\n";

while (my $c = $d->accept) { # 1)
    while (my $r = $c->get_request) { # 2)
        if ($r->method eq 'GET') {

            $file_s= "./index.html";    # index.html - jakis istniejacy plik
            $c->send_file_response($file_s);

        }
        else {
            $c->send_error(RC_FORBIDDEN)
        }
    }
    $c->close;
    undef($c);
}
```

W załączonym do listy pliku znajduje się prosta implementacja serwera HTTP.

Przy uruchomieniu go włączany jest serwer daemon na adresie `lukim` i porcie `4321`.

W pętli 1) sprawdzamy, czy serwer może akceptować połączenia od klienta (dopóki może akceptować, wykonuj).

W pętli 2) Odbierane jest przychodzące żądanie od klienta z metodą **GET**. W odpowiedzi wysyłany jest statyczny plik **index.html**, który może być wyświetlony w przeglądarce.

Jeśli zapytanie nie jest **GET** odsyłany jest error **403 Forbidden**.

Kiedy klient nie wysyła już żądań, zamykamy jego połączenie i dereferujemy zmienną **\$c**.

Będąc całkowicie szczerzy, nie włączałem tego skryptu, bo nigdy nie korzystałem z Perla, ale całkiem łatwo się domysleć, o co tu chodzi.

### 3. Lepszy serwer w Julii

Napisałem podstawowy serwer z użyciem biblioteki HTTP w Julii:

using HTTP, Sockets

```
const ROUTER = HTTP.Router()
```

```
# Zmień skrypt (lub napisz własny serwer w dowolnym języku programowania) tak aby wysyłał do klienta nagłówek jego żądania.
```

```
println("Server running on port 8001 . . .")
```

```
function print_header(header)
```

```
    result = ""
```

```
    for pair in header
```

```
        result *= "${pair[1]}: ${pair[2]}\n"
```

```
    end
```

```
    return result
```

```
end
```

```
HTTP.@register(ROUTER, "GET", "/header", req->HTTP.Response(200, "\n$(print_header(HTTP.Messages.headers(req)))"))
```

```
# Zmień skrypt (lub napisz własny serwer w dowolnym języku programowania)
```

```
# tak aby obsługiwał żądania klienta do prostego tekstowego serwisu WWW
```

```
# (kilka statycznych ston z wzajemnymi odwołaniami) zapisanego w pewnym katalogu dysku lokalnego komputera na którym uruchomiony jest skrypt
```

```
HTTP.@register(ROUTER, "GET", "/", req->HTTP.Response(read("./index.html")))
```

```
HTTP.@register(ROUTER, "GET", "/papuez", req->HTTP.request("GET", "https://media.discordapp.net/attachments/868291982768894013/889955128252"))
```

```
HTTP.@register(ROUTER, "GET", "/fajnefotki", req->HTTP.Response(read("./fajne_fotki.html")))
```

```
HTTP.@register(ROUTER, "GET", "/kotek", req->HTTP.Response(read("./kotekasi.jpg")))
```

```
HTTP.@register(ROUTER, "GET", "/lorem", req->HTTP.Response(read("./lorem")))
```

```
HTTP.@register(ROUTER, "GET", "/bye", req->HTTP.Response(200, "Bye!"))
```

```
HTTP.@register(ROUTER, "GET", "/*", req->HTTP.Response(404, "Not found!"))
```

```
HTTP.serve(ROUTER, Sockets.localhost, 8001)
```

Na samym początku działania tworzony jest obiekt ROUTER, który odpowiada za przekierowanie żądań na dane adresy na odpowiednie pliki html/txt/jpg.

Na przykład, zapytania **GET** na *localhost:8001/fajnefotki* zwróci w odpowiedzi plik *fajne\_fotki.html*.

Każda taka ścieżka musi zostać zarejestrowana w naszym routerze poprzez użycie *HTTP.@register*.

Na samym końcu uruchamiamy serwer z routerem **ROUTER** na adresie **localhost** na porcie **8001**.

### 3.1 Zwracanie headera użytkownikowi

Aby podpatrzeć nagłówek żądania HTTP Julii trzeba wywołać funkcję **HTTP.Messages.headers()** na obiekcie **req** (request - żądanie).

Funkcja ta zwraca słownik *SubString* => *SubString*, więc napisałem prosty parser, który zapisuje "pole nagłówka: wartość" do Stringa i zwracam go klientowi jako odpowiedź na żądanie.

```
Host: localhost:8001
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="102", "Google Chrome";v="102"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
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=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:8001/
Accept-Encoding: gzip, deflate, br
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
```

*Rysunek 1. Przykład odesłania headera do klienta.*

#### ▼ 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=0.9
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
Host: localhost:8001
Referer: http://localhost:8001/
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="102", "Google Chrome";v="102"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
```

*Rysunek 2. Rzeczywisty nagłówek (zgadza się z tym co odsełałem).*

Opis wszystkich wartości nagłówka:

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=0.9  
// Opisuje jakie typy danych możemy przesłać do klienta

Accept-Encoding: gzip, deflate, br  
// Opisuje jakie typy kodowania możemy używać przy rozmowie z klientem

Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7  
// Opisuje preferencje językowe klienta

Cache-Control: max-age=0  
// Zawiera dyrektywy kontrolujące proces cachingu przeglądarki  
// Tutaj informacja max-age=0 oznacza, że po 0 sekundach przekazana w żądaniu informacja staje się przedawniona (remains "fresh" for 0 seconds)

Connection: keep-alive  
// Kontroluje sposób połączenia klient-serwer, keep-alive oznacza, że połączenie ma być trwałe i niezamykane

Host: localhost:8001  
// Host i port serwera, na który jest wysyłane żądanie

Referer: http://localhost:8001/  
// Zawiera cały/część adres/u strony, z której wysłano żądanie do serwera

sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="102", "Google Chrome";v="102"  
// "User agent hint" -> zawiera branding user-agenta (niżej wyjaśnione co to jest) i informacje o wersjach komponentów przeglądarki.

sec-ch-ua-mobile: ?0  
// ?0 - użytkownik nie jest na urządzeniu mobilnym  
// ?1 - użytkownik jest na urządzeniu mobilnym

sec-ch-ua-platform: "Windows"  
// Jaki system/platformę wykorzystuje użytkownik?

Sec-Fetch-Dest: document  
// Wskazuje cel żądania (o co prosimy? o dokument)

Sec-Fetch-Mode: navigate  
// Wskazuje tryb żądania

Sec-Fetch-Site: same-origin

// Wskazuje czy żądanie przyszło z tej samej domeny czy nie (same-origin - ta sama domena, cross-site - żądanie z zewnętrznej strony)

Sec-Fetch-User: ?1

// Żądanie zostało wykonane bezpośrednio przez użytkownika

Upgrade-Insecure-Requests: 1

// Wysyła informację do serwera, która oznacza, że klient preferuje szyfrowanie i uwierzytelnianie odpowiedzi

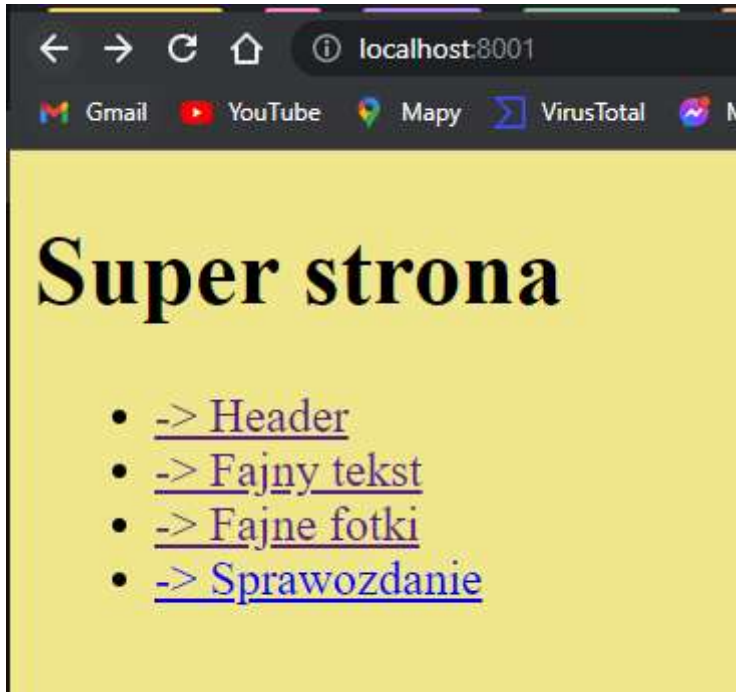
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36

(KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36

// Pozwala serwerom rozpoznać aplikację, system operacyjny, vendor oraz wersje z których korzysta użytkownik

## 3.2 Prosty serwis WWW

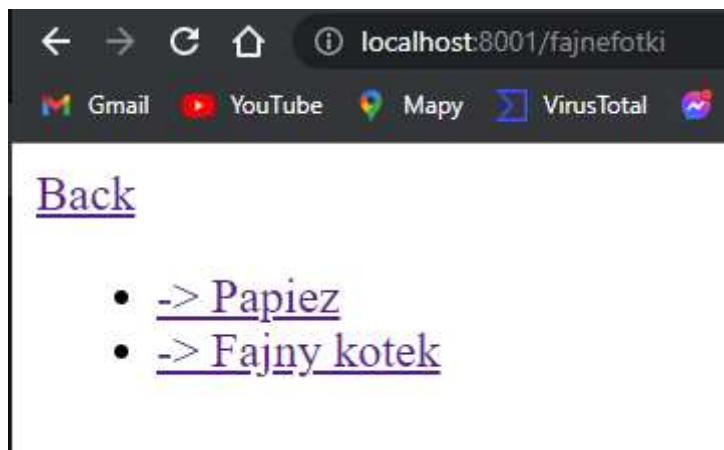
Zaprojektowałem prostą stronę w języku programowania HTML, która zawiera parę linków do innych plików na moim dysku:



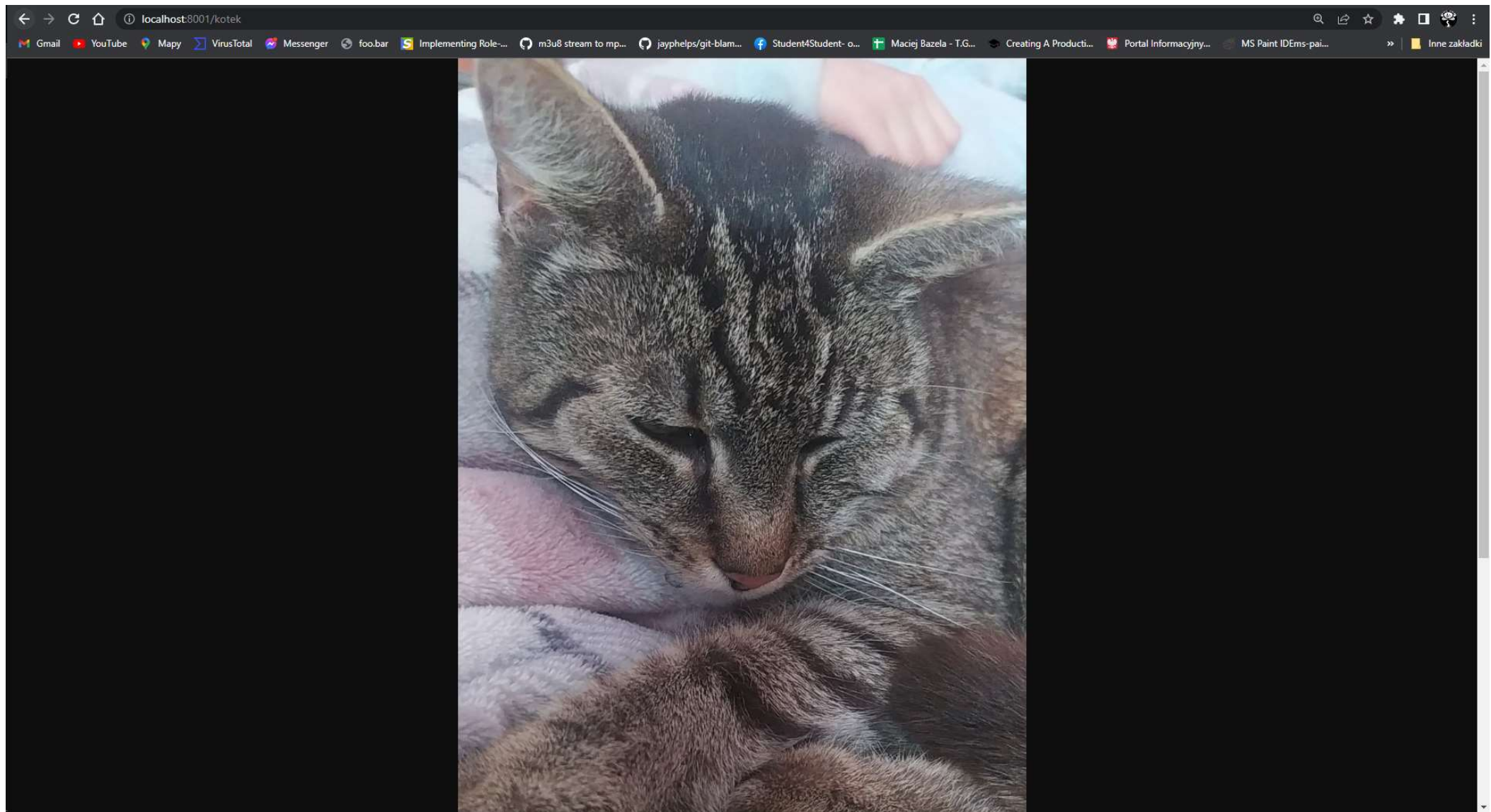


Rysunek 3. Super strona.

Zamieszczone linki prowadzą np. do zadania ze zwracaniem headera, do podstrony zawierającej linki do fajnych zdjęć (fotek) i jakiś przykładowy tekst (lorem ipsum), aby pokazać, że można odsyłać plain text:



Rysunek 4. Fajne fotki.



*Rysunek 5. Fajny kotek.*

```
localhost:8001/header

Host: localhost:8001
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="102", "Google Chrome";v="102"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
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=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:8001/
Accept-Encoding: gzip, deflate, br
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
```

Rysunek 6. Zadanie z headerem.

```
localhost:8001/lorem

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin sit amet ipsum quis mauris gravida congue ac ut dui. Phasellus et metus purus. Suspendisse potenti. Phasellus ultrices porta lacus, nec vulputate ex mollis ac. Suspendisse dapibus risus et mauris molestie efficitur. Praesent fermentum pulvinar lacus id viverra. Ut in volutpat libero, id auctor sem. Maecenas ut neque volutpat, tincidunt lorem vitae, venenatis tortor. Nulla ut sodales felis. Nam consectetur est eget nisi fermentum posuere. Curabitur vehicula sed libero a gravida.

In lobortis, justo egestas aliquet feugiat, leo urna bibendum augue, sed volutpat ex eros vitae enim. Fusce efficitur blandit congue. Pellentesque vehicula risus enim, id maximus dui suscipit ut. Etiam viverra posuere dui, vitae sodales risus laoreet sed. Vestibulum bibendum lectus eu augue luctus consequat. Maecenas euismod sed lorem ac egestas. Curabitur tellus sem, lacinia ac lectus ut, congue lobortis ipsum.

Vestibulum condimentum enim et libero pharetra, sit amet eleifend nisl consequat. Donec enim ipsum, venenatis quis nisl ut, gravida semper ex. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. In ac ex massa. Aenean nec nisl mattis, vulputate nulla ac, elementum nunc. Donec ante libero, placerat id augue vitae, maximus tincidunt risus. Nunc ut sapien sit amet augue varius feugiat.

Proin ut interdum nibh. Etiam eget velit non ex tempus eleifend in vel neque. Pellentesque maximus, mauris aliquam placerat gravida, metus ligula ultrices velit, quis placerat enim magna in nisl. In egestas sagittis massa, eu tincidunt turpis malesuada et. Donec molestie, velit congue sagittis pretium, arcu ante gravida libero, at egestas velit purus vel lorem. Mauris imperdiet tincidunt turpis sit amet vehicula. Aenean dignissim erat sed augue porta, vel scelerisque nulla commodo. Phasellus finibus ipsum tincidunt aliquam scelerisque. Maecenas tristique blandit velit, id venenatis orci facilisis vel. Vestibulum cursus a metus eget hendrerit. Aenean arcu dolor, vestibulum iaculis dui vitae, iaculis egestas nunc. In bibendum neque ac posuere rutrum. Nulla malesuada magna vitae ante varius, eget ultricies ipsum auctor.

Nulla facilisi. Nulla viverra, magna et consequat auctor, libero orci mattis ante, in faucibus nulla nibh in sem. Sed egestas accumsan hendrerit. Aenean ante ex, pharetra vitae tincidunt ac, mollis vitae nunc. Phasellus ac malesuada elit. Mauris euismod ac nulla vel tempus. Phasellus suscipit mauris ut felis fringilla tincidunt. Nam suscipit mauris lectus, eu fringilla neque posuere ac.
```

*Rysunek 7. Lorem.*

### **3.3 Analizator sieciowy**

Po naciśnięciu F12 uruchamia się następujące narzędzie:

ElementsConsoleSourcesNetworkPerformanceMemoryApplicationSecurityLighthouseRecorderPerformance insights

Filter

Preserve log

Disable cache

No throttling

Fetch/XHR

JS

CSS

Img

Media

Font

Doc

WS

Wasm

Manifest

Other

Has blocked cookies

Blocked Requests

3rd-party requests

Use large request rows

Group by frame

Show overview

Capture screenshots

10 ms

20 ms

30 ms

40 ms

50 ms

60 ms

70 ms

80 ms

90 ms

100 ms

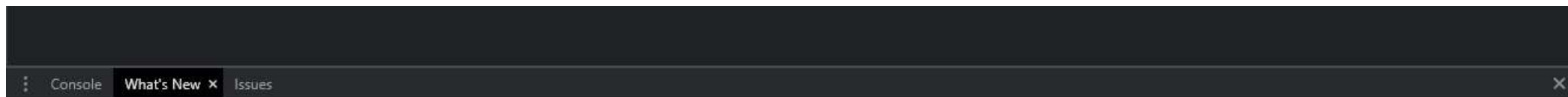
110 ms

Recording network activity...

Perform a request or hit **Ctrl + R** to record the reload.

[Learn more](#)





*Rysunek 8. Dev tools w Google Chrome.*

Służy ona jako narzędzie developerskie, a przydaje się szczególnie do wyłapywania wysyłanych/odbieranych żądań na danych stronach.

Aby poprawnie jej użyć, trzeba przełączyć się na karte *Network*, zaznaczyć przechwytywanie na *All* i odświeżyć stronę.

Kiedy to zrobimy, zobaczymy przepływ komunikatów:

localhost:8001/lorem

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights

Filter ☐ Invert ☐ Hide data URLs ☒ All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other ☐ Has blocked cookies ☐ Blocked Requests ☐ 3rd-party requests

☐ Use large request rows ☐ Group by frame ☒ Show overview ☐ Capture screenshots

10 ms 20 ms 30 ms 40 ms 50 ms 60 ms 70 ms 80 ms 90 ms 100 ms 110 ms

| Name  | Status | Type     | Initiator | Size   | Ti... | Waterfall |
|-------|--------|----------|-----------|--------|-------|-----------|
| lorem | 200    | document | Other     | 2.8 kB | 2 ... |           |

1 requests | 2.8 kB transferred | 2.7 kB resources | Finish: 2 ms | DOMContentLoaded: 22 ms | Load: 22 ms

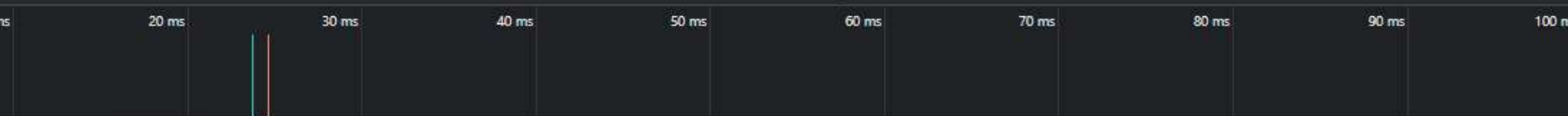
Console What's New Issues

Rysunek 9. Widok na requesty.

**Network**

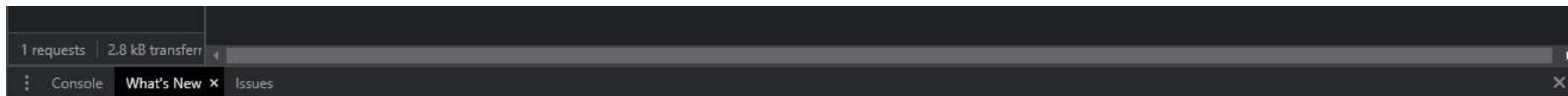
Filter: ☐ Invert ☐ Hide data URLs ☒ All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other ☐ Has blocked cookies ☐ Blocked Requests ☐ 3rd-party requests

☐ Use large request rows ☐ Group by frame  
☒ Show overview ☐ Capture screenshots



| Name  | Headers  | Preview   | Response  | Initiator | Timing |
|-------|--|---|---|-----------|--------|
| lorem | <b>General</b><br>Request URL: http://localhost:8001/lorem<br>Request Method: GET<br>Status Code: 200 OK<br>Remote Address: 127.0.0.1:8001<br>Referrer Policy: strict-origin-when-cross-origin | <b>Response Headers</b> View source<br>Transfer-Encoding: chunked | <b>Request Headers</b> View source<br>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=0.9<br>Accept-Encoding: gzip, deflate, br<br>Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7<br>Cache-Control: max-age=0<br>Connection: keep-alive<br>Host: localhost:8001<br>Referer: http://localhost:8001/<br>sec-ch-ua: "Not A;Brand";v="99", "Chromium";v="102", "Google Chrome";v="102"<br>sec-ch-ua-mobile: ?0<br>sec-ch-ua-platform: "Windows"<br>Sec-Fetch-Dest: document<br>Sec-Fetch-Mode: navigate<br>Sec-Fetch-Site: same-origin<br>Sec-Fetch-User: ?1<br>Upgrade-Insecure-Requests: 1<br>User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 |           |        |





*Rysunek 10. Szczegóły danego żądania.*

Możemy dokładnie odczytać każde pole naszego żądania i zwróconej odpowiedzi.

Request headers jest opisane wyżej, natomiast poza response headerem pojawia się szereg dodatkowych informacji:

```
Request URL: http://localhost:8001/lorem
// Na jaki URL przyszło żądanie

Request Method: GET
// Jaka jest metoda żądania?

Status Code: 200 OK
// Jaki jest status (kod statusu) odpowiedzi na żądanie
// 200 - wszystko jest OK

Remote Address: 127.0.0.1:8001
// Adres serwera (localhost:8001)

Referrer Policy: strict-origin-when-cross-origin
// Kontroluje ilość informacji przesyłanej w headerze Referer
```

W Response headerze znajduje się informacja co zostało zwrócone: chunked plik (czyli po prostu przekazaliśmy plik z serwera do klienta).

Normalnie byłoby tu dużo więcej informacji, jak na przykład informacje o cachingu, serwerze, customowe nagłówki etc.

Na innych stronach mojego "serwisu" żądania wyglądają praktycznie tak samo.

Dodatkowo, nad podanymi informacjami znajduje się oś czasu, która liczy czas w ms od załadowania strony. Można na niej podpatrzyć kiedy przesłano dane żądanie.

W zakładce previews można zobaczyć (jak można się domyślić) podgląd przesłanej informacji, response zawiera np. kod, w przypadku kiedy przesyłamy np. plik .html.