

Sprawozdanie Lista 5

Przedmiot	Technologie sieciowe
Prowadzący	Mgr inż. Dominik Bojko
Autor	Joanna Kulig
Indeks	261743
Grupa	Czw. 17:05 - 18:45
Kod grupy	K03-76d

1. Opis zadania.

Na podstawie [załączonego skryptu](#) należało:

- przeanalizować działanie serwera
- nawiązać połączenie za pomocą przeglądarki internetowej

Ponadto, zmieniając powyższy skrypt lub pisząc własny należało:

- zaimplementować zwracanie klientowi nagłówek jego żądania
- zaimplementować obsługę żądań klienta do prostego tekstowego serwisu WWW, który składałby się z kilku statycznych stron z wzajemnymi odwołaniami
- przeanalizować komunikaty wysyłane do i od serwera

1.1 Środowisko.

Zamiast korzystać z podanego skryptu, napisałam własną implementację, używając do tego języka Julia oraz prostego HTML.

2. Analiza skryptu.

```
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) {
    while (my $r = $c->get_request) {
        if ($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);
}
```

Powyższy skrypt, po włączeniu, uruchamia serwer daeomon na adresie lukim i porcie 4321 . Jednakże przy uruchomieniu musiałam zmienić adres serwera na localhost , aby serwer poprawnie działał.

Uruchomienie:

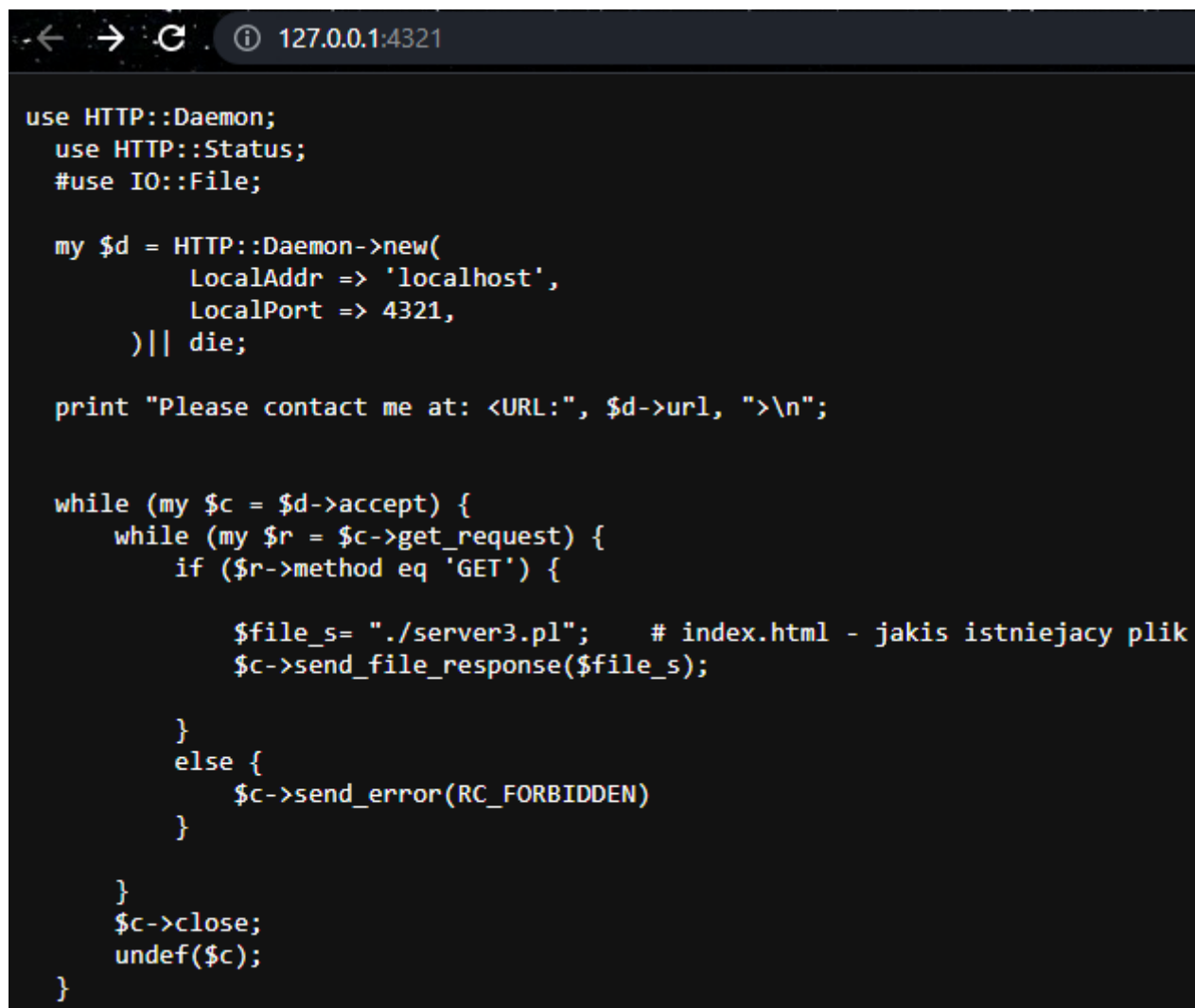
```
perl server3.pl
```

Wiadomość po uruchomieniu:

```
Please contact me at: <URL:http://127.0.0.1:4321/>
```

Pierwsza pętla while obsługuje działanie serwera, mianowicie jeżeli serwer działa, będzie on wykonywał żądania klienta. Serwer zostaje zamkniętym, gdy nie obsługuje już żądań klienta.

Druga pętla obsługuje żądania klienta. Jeżeli jest ono typu **GET**, zwraca statyczny plik index.html . W przeciwnym przypadku wysyłany jest error **403 Forbidden**.



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

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

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

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

            $file_s= "./server3.pl";    # index.html - jakis istniejacy plik
            $c->send_file_response($file_s);

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

Rysunek 1. Uruchomienie oraz zmiana programu [server3.pl](#), aby pokazywał swój kod źródłowy.

2. Implementacja serwera.

```
using HTTP, Sockets
```

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

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

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

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

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

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

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

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

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

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

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

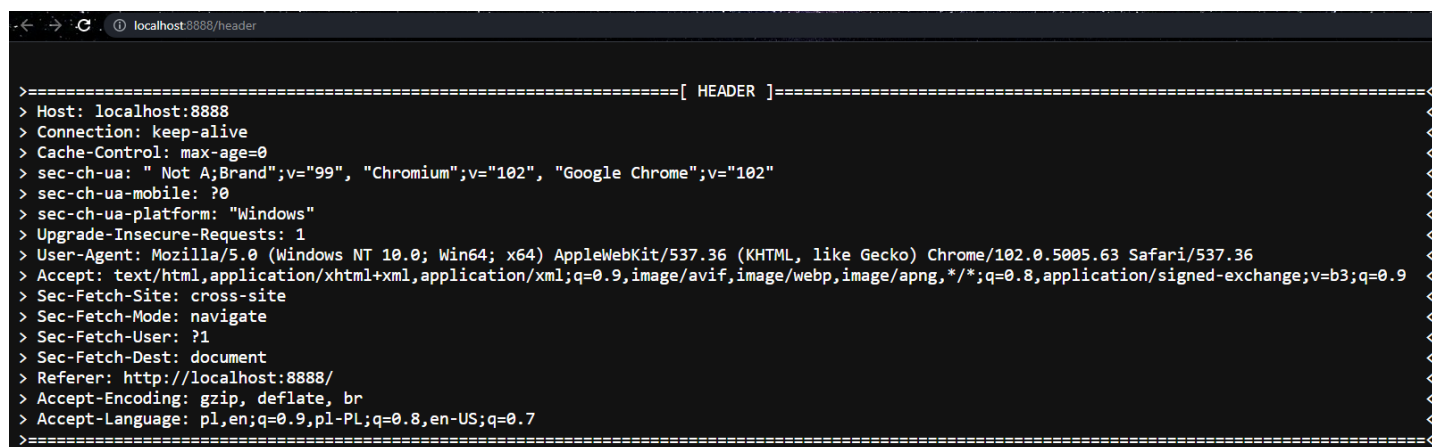
```
HTTP.serve(ROUTER, Sockets.localhost, 8888)
```

Na początku implementacji serwera tworzony jest `ROUTER`, który będzie obsługiwał i przekierowywał żądania od klienta. Każde z nich musi być zarejestrowane przez funkcję `HTTP.@register`, która zapisuje obsługę danego żądania.

Serwer uruchamiamy poprzez `HTTP.serve`, który zwróci serwer z routerem `ROUTER`, na adresie `localhost` z portem `8888`.

2.1 Zwracanie nagłówka.

Aby zwrócić nagłówek należało użyć funkcji `HTTP.Messages.headers()`. Po napisaniu funkcji do printowania, otrzymałam następujący nagłówek:



```
<=====[ HEADER ]=====>
> Host: localhost:8888
> Connection: keep-alive
> Cache-Control: max-age=0
> 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: cross-site
> Sec-Fetch-Mode: navigate
> Sec-Fetch-User: ?1
> Sec-Fetch-Dest: document
> Referer: http://localhost:8888/
> Accept-Encoding: gzip, deflate, br
> Accept-Language: pl,en;q=0.9,pl-PL;q=0.8,en-US;q=0.7
>=====
```

Rysunek 2. Nagłówek zwrócony przez serwer.

Po otwarciu analizatora sieciowego możemy sprawdzić, że zgadza się z tym wysłanym od serwera.

```
▼ 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,en;q=0.9,pl-PL;q=0.8,en-US;q=0.7
Cache-Control: max-age=0
Connection: keep-alive
Host: localhost:8888
Referer: http://localhost:8888/
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: cross-site
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
```

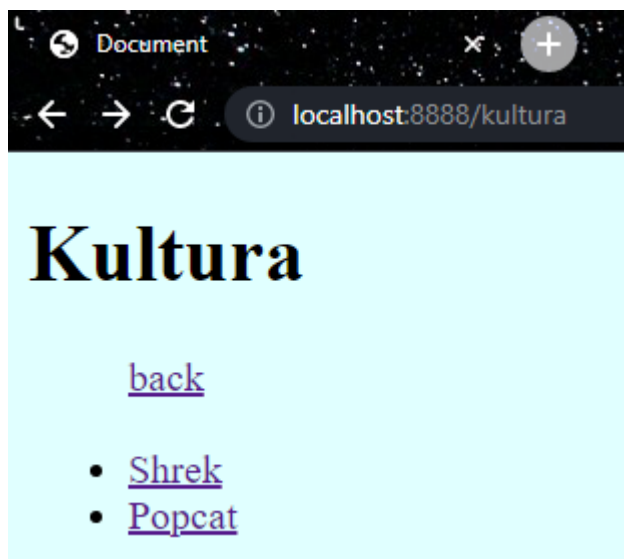
Rysunek 3. Nagłówek w analizatorze sieciowym.

2.2 Serwis WWW.

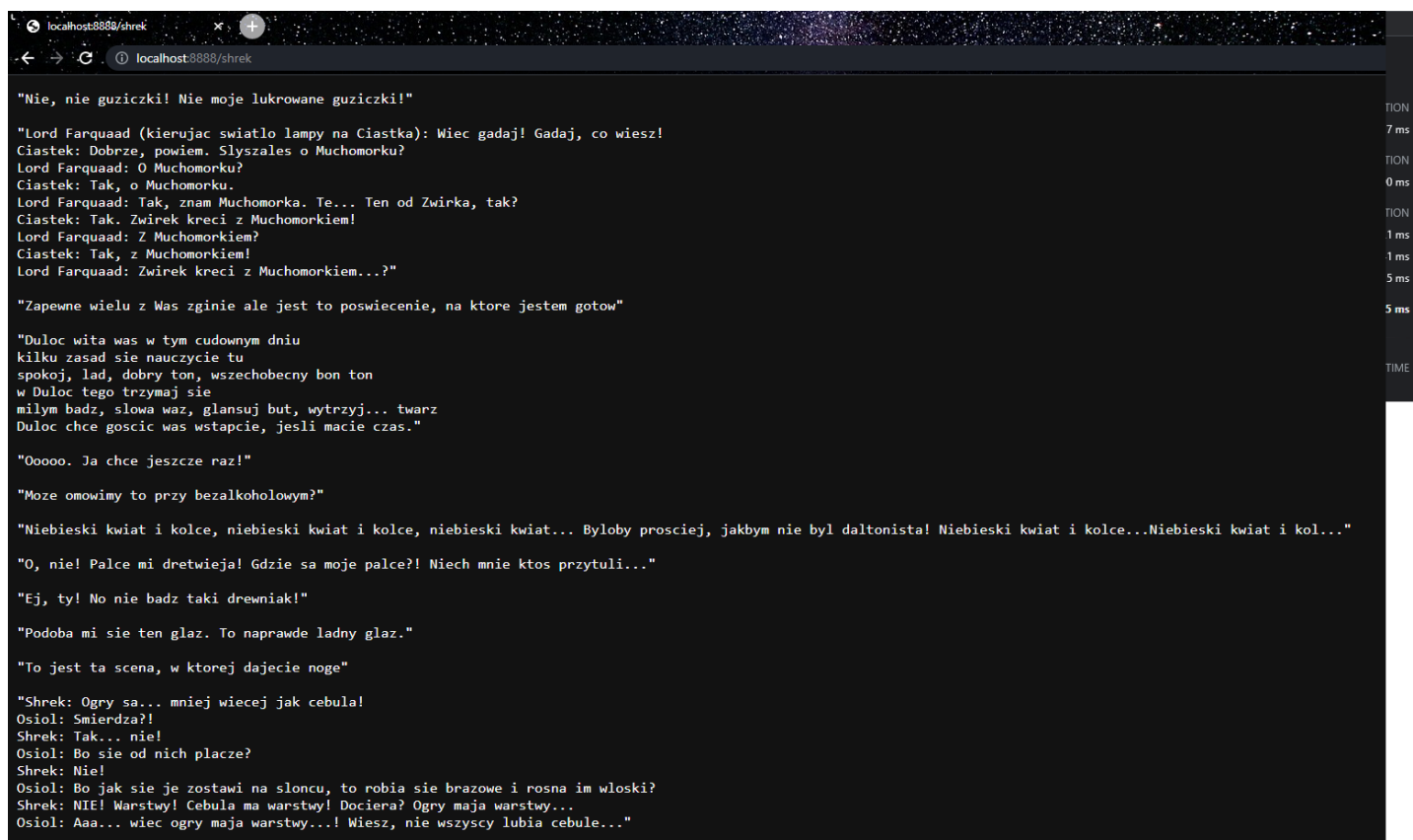
Zaprojektowałam prosty serwis w HTML, który posiada dwie podstrony, z których można się cofać do strony głównej lub przechodzić do danych plików zlokalizowanych na moim dysku. Jest również opcja, aby zobaczyć wyżej opisany header żądania.



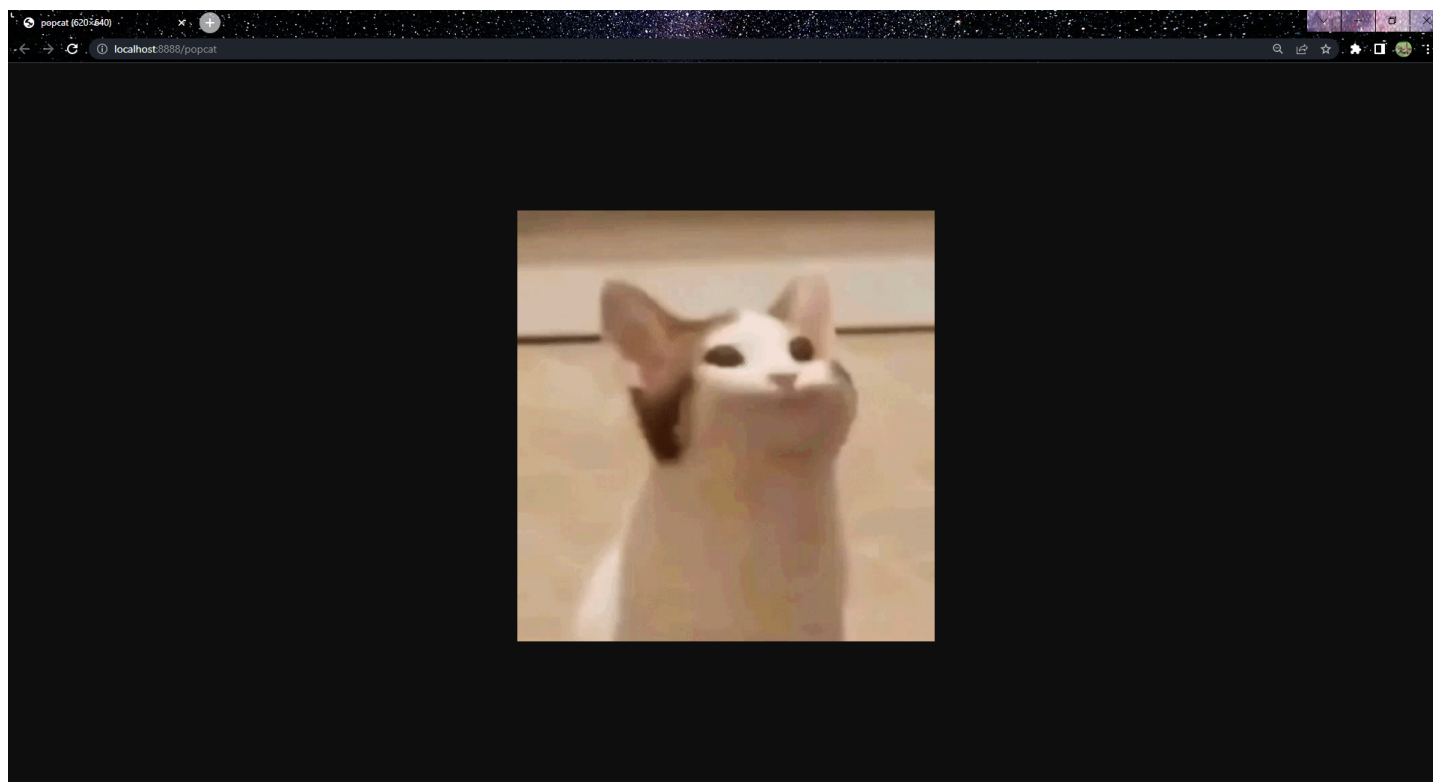
Rysunek 4. Uruchomienie powyższego serwera.



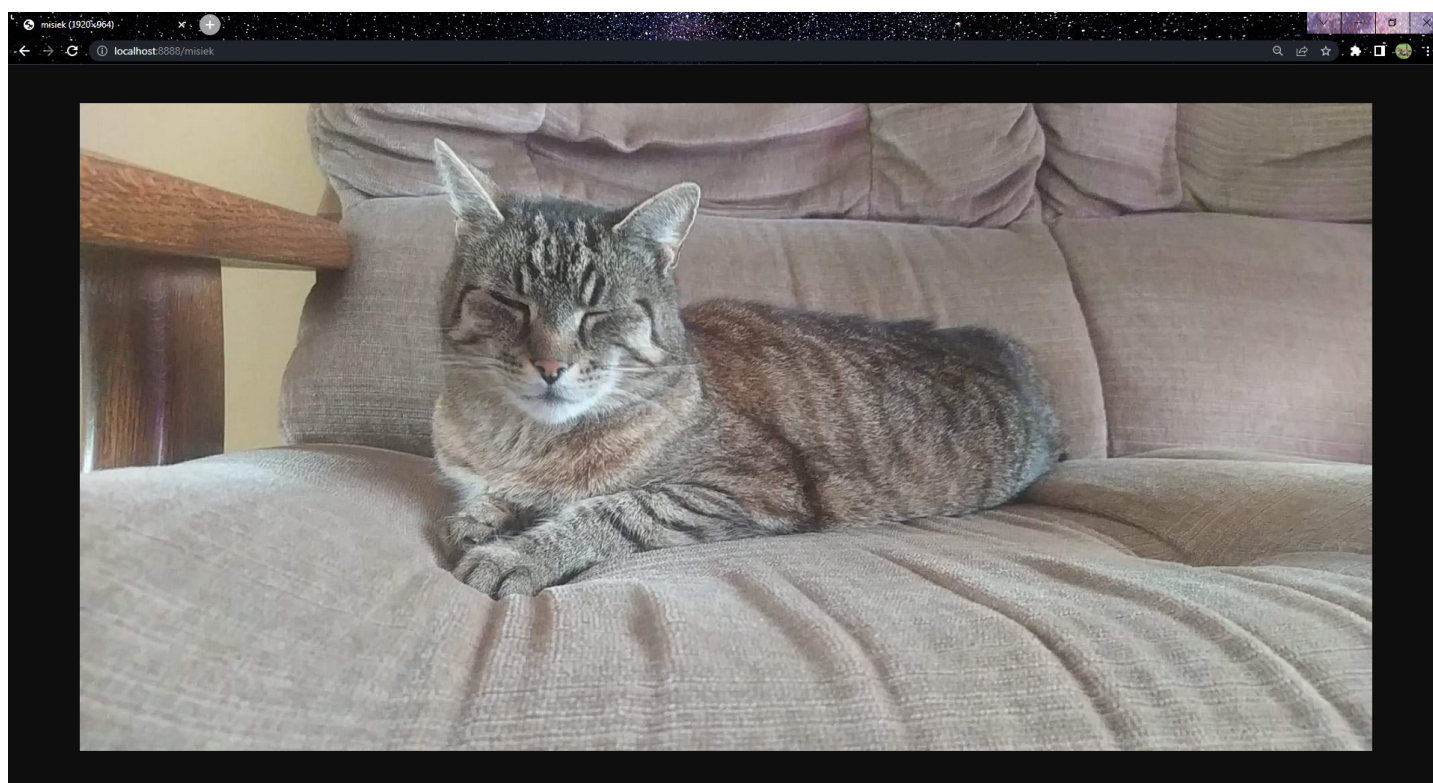
Rysunek 5. Podstrona /kultura z możliwością powrotu do strony głównej.



Rysunek 6. Plik tekstowy Shrek.

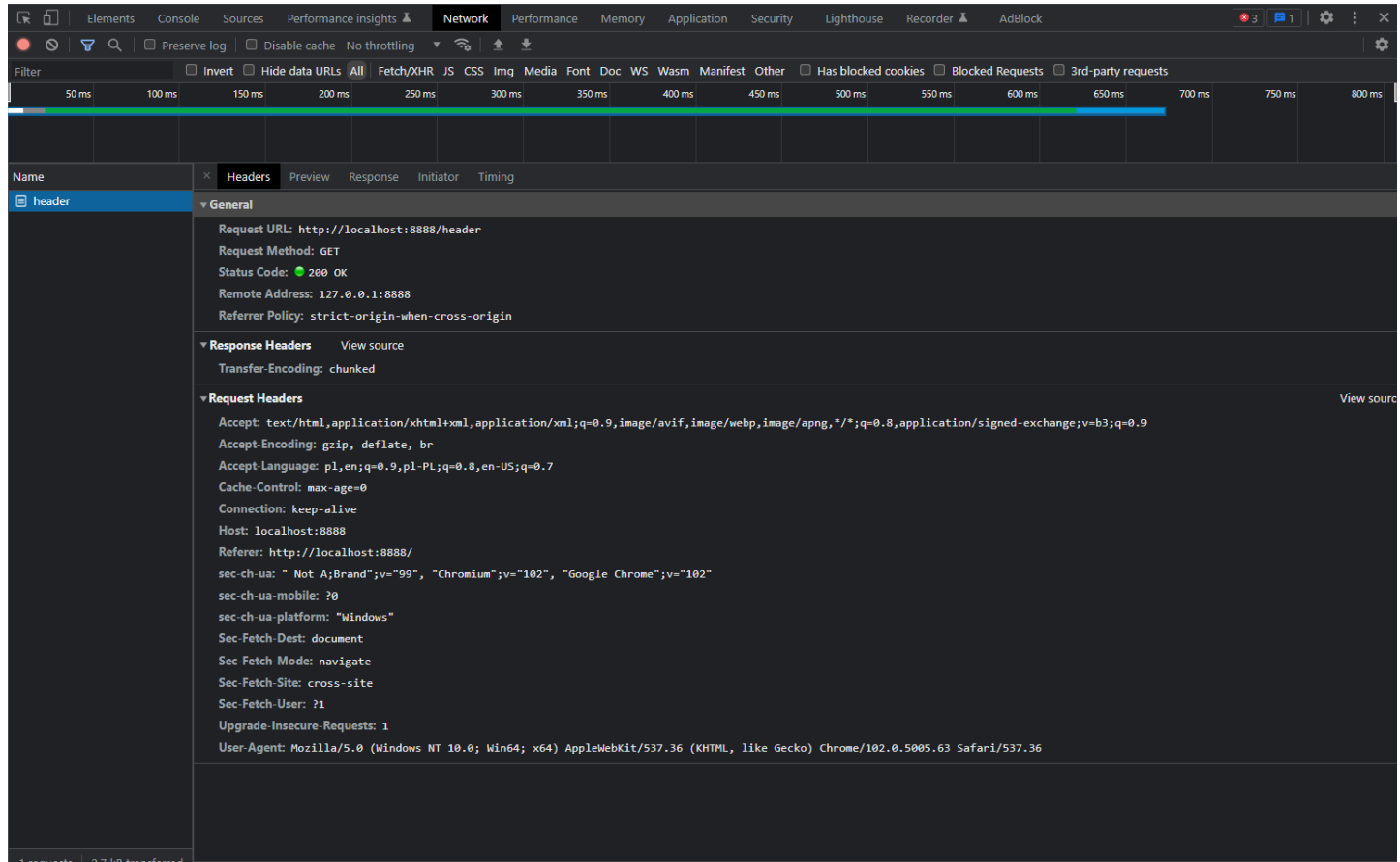


Rysunek 7. Gif Popcat.



Rysunek 8. Zdjęcie kota.

3. Analizator sieciowy.



Rysunek 9. Analizator sieciowy.

3.1 Analiza nagłówka.

Accept // informuje jakie typy danych możemy wysłać w odpowiedzi

Accept-Encoding // Opisuje jakie typy kodowania możemy używać przy realizowaniu żądań

Accept-Language // Informuje jaki język jest używany

Cache-Control // Opisuje dyrektywy mechanizmu cache przy żądaniach i odpowiedziach

Connection // Kontroluje czy połączenie zostaje otwarte po zakończeniu żądania, keep-alive zapo

Host // Pokazuje adres i port serwera odbierającego żądania

Referer // Adres stront, z której wysłano żądanie do serwera

sec-ch-ua // Posiada informacje o przeglądarce i wersjach jego komponentów

sec-ch-ua-mobile // Informacja o tym, czy użytkownik jest na urządzeniu mobilnym: 0? - nie jes

sec-ch-ua-platform // Pokazuje system wykorzystywany przez użytkownika.

Sec-Fetch-Dest // Pokazuje cel rządania

Sec-Fetch-Mode // Pokazuje tryb żądania

Sec-Fetch-Site // Pokazuje relację między źródłem żądania a celem żądania, same-site - żądanie

Sec-Fetch-User // Pokazuje czy żądanie przyszło od użytkownika: 0? nie, 1? tak

Upgrade-Insecure-Requests // Wysyła sygnał do serwera, który przekauje preferencje użytkownika

User-Agent // Pozwala zidentyfikować typ apikacji, system operacyjny oraz przeglądarkę i jej w

3.2 Analiza pozostałych danych.

Oprócz informacji o nagłówku żądania, są tu także inne informacje:

Request URL // Pokazuje na jaki adres przyszło żądanie

Request Method // Pokazuje metodę żądania

Status Code // Pokazuje kod statusu odpowiedzi

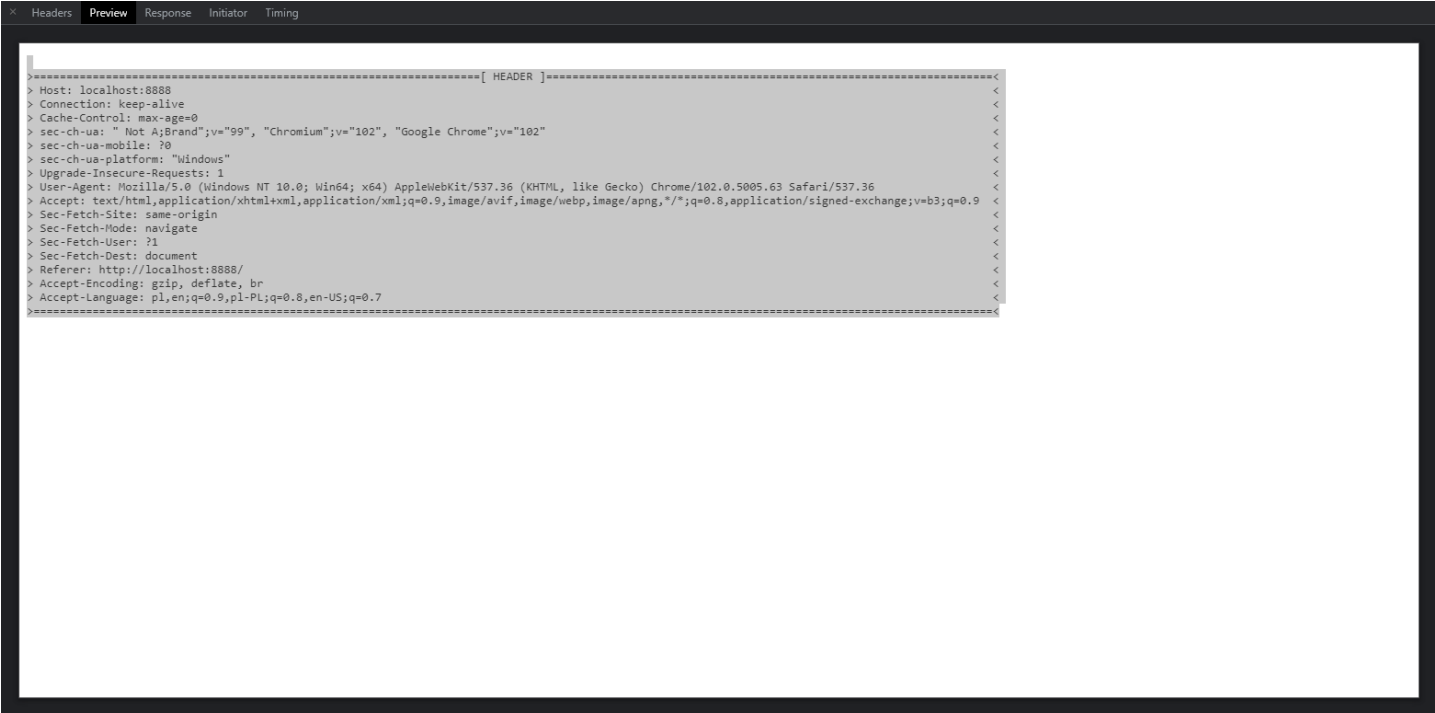
Remote Address // Pokazuje adres serwera

Referrer Policy // Zakres informacji, które powinny być zawarte w żądaniu

// Response Header

Transfer Encoding // Określa formę szyfrowania, która została użyta do przesłania zasobu do uży

W zakładce **Preview** można zobaczyć podgląd wysłanej użytkownikowi odpowiedzi, a w **Response** odpowiedź serwera.



Rysunek 10. Preview.

W zakładce **Time** można się przyrzeć czasom działania poszczególnych zadań.



Rysunek 11. Time.