UAI - Laboratorium 5 - HTTP

Stanisław Kwiatkowski, Bartosz Ziemba, Kamil Stachowicz

1. Wprowadzenie

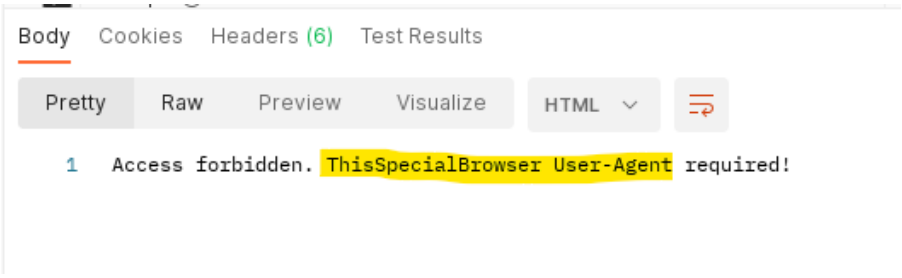
1.1. Cel laboratorium

Celem tego ćwiczenia jest zapoznanie się z usługą World Wide Web (WWW) i zrozumienie działania protokołów HTTP i TCP, które są wykorzystywane w komunikacji między klientami a serwerami. Będziemy również korzystać z narzędzia Wireshark, które umożliwi nam analizę ruchu sieciowego i zrozumienie, jak dane są przesyłane między urządzeniami sieciowymi i aplikacjami. Poprzez eksplorację podstaw HTML oraz naukę podstaw protokołu HTTP, będziemy mogli modyfikować kod źródłowy stron internetowych. Zapoznanie się z programem Wireshark pozwoli nam natomiast na monitorowanie i zapisywanie pakietów przesyłanych w sieci, co umożliwi nam lepsze zrozumienie procesu komunikacji w sieci.

2. Zadanie

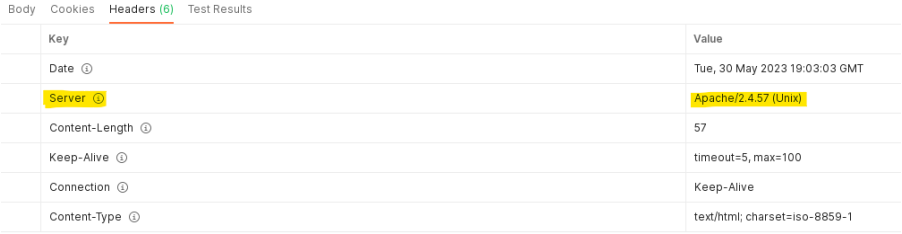
2.1. Próba pobrania pliku *index.html*

Podczas pierwszej próby pobrania pliku *index.html* używaliśmy zapytania typu GET wraz z domyślnym nagłówkiem. W wyniku wyłania naszego zapytania o plik *index.html* wyświetlił się nam błąd braku dostępu. Z wiadomości jesteśmy w stanie odczytać, że powinniśmy zmienić argument **User-Agent** do nagłówka naszego zapytania.



Rys. 1: Odpowiedź na zapytanie GET o *index.html*

Z analizy nagłówka odpowiedzi jesteśmy w stanie odczytać rodzaj i wersję oprogramowania serwera www i jest to **Apache/2.4.57**

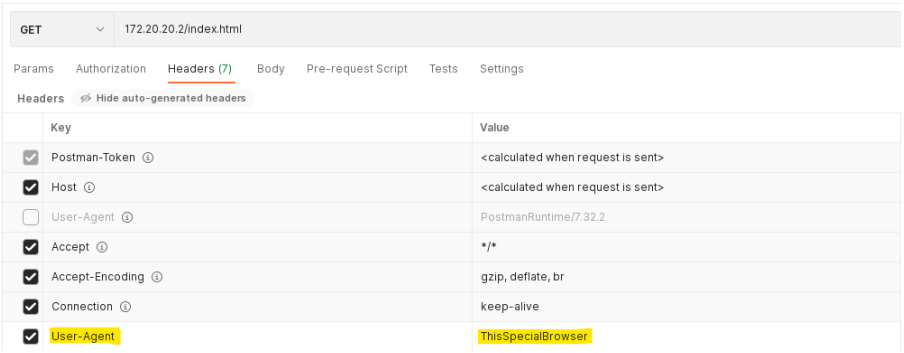


Rys. 2: Rodzaj i wersja serweru

2.2. Udane pobranie pliku *index.html*

2.2.1. Modyfikacja nagłówka

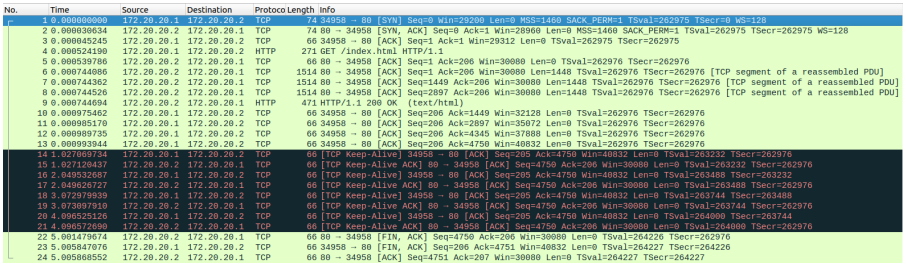
Wysłanie zapytania ze zmienionym atrybutem **User-Agent** w nagłówku dało nam efekt w postaci pobrania pliku *index.html*.



Rys. 3: Modyfikacje nagłówka

2.2.2. Wymiana wiadomości między klientem a serwerem w Wireshark

Wymiane wiadomości między klientem a serwerem przechwyconą przez Wireshark można zaobserwować na obrazku poniżej:

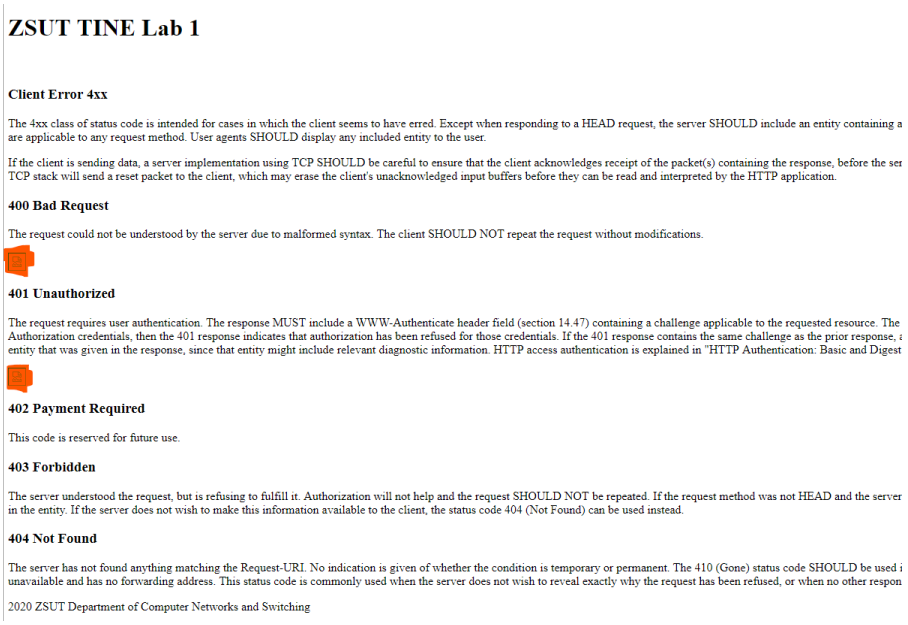


Rys. 4: Wymiana wiadomości.

2.2.3. Zawartość strony

Zawartość strony HTML wydaje się być częścią dokumentacji związanej z błędami 4xx klienta. Te kody błędów są zwracane, gdy klient wydaje się popełnić błąd, np. błędnie sformułowane żądanie.

Jak widać, strona jest niepełna. Jest to spowodowane faktem, iż pobraliśmy tylko jeden plik z rozszerzeniem *.html*, dlatego aby uzyskać dostęp do pełnej funkcjonalności strony potrzebujemy ręcznie pobrać brakujące elementy.



ZSUT TINE Lab 1

Client Error 4xx

The 4xx class of status code is intended for cases in which the client seems to have erred. Except when responding to a HEAD request, the server SHOULD include an entity containing a are applicable to any request method. User agents SHOULD display any included entity to the user.

If the client is sending data, a server implementation using TCP SHOULD be careful to ensure that the client acknowledges receipt of the packet(s) containing the response, before the ser TCP stack will send a reset packet to the client, which may erase the client's unacknowledged input buffers before they can be read and interpreted by the HTTP application.

400 Bad Request

The request could not be understood by the server due to malformed syntax. The client SHOULD NOT repeat the request without modifications.

401 Unauthorized

The request requires user authentication. The response MUST include a WWW-Authenticate header field (section 14.47) containing a challenge applicable to the requested resource. The Authorization credentials, then the 401 response indicates that authorization has been refused for those credentials. If the 401 response contains the same challenge as the prior response, a entity that was given in the response, since that entity might include relevant diagnostic information. HTTP access authentication is explained in "HTTP Authentication: Basic and Digest

402 Payment Required

This code is reserved for future use.

403 Forbidden

The server understood the request, but is refusing to fulfill it. Authorization will not help and the request SHOULD NOT be repeated. If the request method was not HEAD and the server in the entity. If the server does not wish to make this information available to the client, the status code 404 (Not Found) can be used instead.

404 Not Found

The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent. The 410 (Gone) status code SHOULD be used i unavailable and has no forwarding address. This status code is commonly used when the server does not wish to reveal exactly why the request has been refused, or when no other respon

2020 ZSUT Department of Computer Networks and Switching

Rys. 5: Brakujące elementy strony

2.3. Analiza wysłanych nagłówków w żądaniu GET

Opis nagłówków w żądaniu GET prezentuje się następująco:

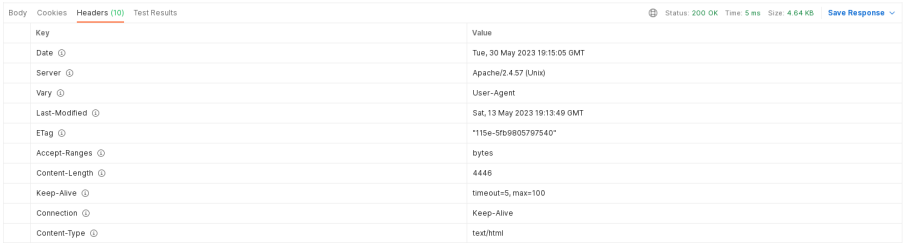
- **HTTP/1.1 200 OK**: Serwer zwrócił kod statusu 200, co oznacza, że żądanie zostało pomyślnie przetworzone.
- **Content-Length: 4446**: Długość zawartości (treści) odpowiedzi serwera wynosi 4446 bajtów.
- **Accept-Ranges: bytes**: Serwer akceptuje żądania zakresu bajtów, co pozwala na wznowienie przesyłania plików lub "strumieniowanie" plików w podzielonych na części żądaniach.
- **Connection: Keep-Alive**: Serwer akceptuje trwałe połączenie TCP.
- **Content-Type: text/html**: Typ zawartości zwracanej przez serwer to HTML.
- **Date: 22 GMT**: Data i czas odpowiedzi serwera.
- **ETag: "115e-5fb9805797540"**: ETag (entity tag) jest unikalnym identyfikatorem dla konkretnych wersji zasobu. Jest używany do optymalizacji wydajności sieci poprzez umożliwienie sprawdzania, czy zasób się zmienił.
- **Keep-Alive: timeout=5, max=100**: Parametry trwałego połączenia TCP określające maksymalny czas bezczynności połączenia (5 sekund) i maksymalną liczbę żądań, które mogą być obsłużone przez jedno połączenie (100).
- **Last-Modified: 49 GMT**: Data i czas ostatniej modyfikacji żądanego zasobu.
- **Server: Apache/2.4.57 (Unix)**: Serwer jest oparty na Apache 2.4.57 działającym na systemie Unix.
- **Vary: User-Agent**: Nagłówek "Vary" informuje, że odpowiedź serwera może się różnić w zależności od nagłówka "User-Agent" w żądaniu.

2.3.1. Nagłówki odpowiedzi otrzymane od serwera

Nagłówki odpowiedzi otrzymane od serwera zawiera ciekawe informacje o serwerze, jego wersji oraz połączeniu pomiędzy podmiotami.

- **Date** - Czas wysłania odpowiedzi.
- **Server** - Typ i model serwera.
- **Vary** - Informacja o różnych typach odpowiedzi w zależności od nagłówka "User-Agent".
- **ETag** - Unikalny identyfikator dla konkretnej wersji zasobu. Jest używany do optymalizacji wydajności sieci poprzez umożliwienie sprawdzania, czy zasób się zmienił.
- **Accept-Ranges** - Informuje klienta o możliwości pobierania zasobów w kawałkach, co pozwala klientom na pobieranie tylko określonych fragmentów plików.

- **Content-Length** - Informuje o ilości bajtów, które zostają przesłane w odpowiedzi na zapytanie. Jest to ważne, ponieważ użytkownik wie, jak duży plik jest do niego wysyłany.
- **Keep-Alive** - Parametry mechanizmu protokołu HTTP umożliwiającego utrzymanie jednego połączenia TCP między klientem a serwerem HTTP dla wielu żądań i odpowiedzi, co pozwala na znaczną optymalizację. MAX - Maksymalna liczba żądań na tym połączeniu, Timeout - maksymalny czas oczekiwania na kolejne żądanie.
- **Content-Type** - Typ przesłanej zawartości.

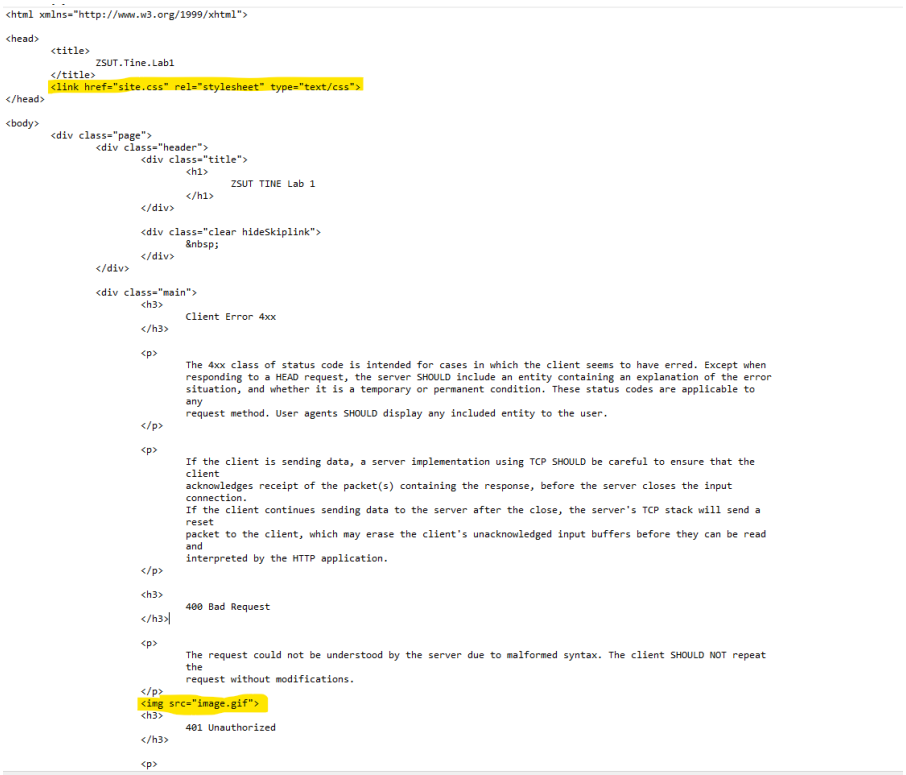


Rys. 6: Nagłowki odpowiedzi

2.4. Pobieranie brakujących elementów

Analizując kod źródłowy pobranego pliku *index.html* zauważyliśmy, że używa on zewnętrznych zasobów, których nie pobraliśmy wcześniejszym żądaniem http. Są to pliki o nazwach:

- *site.css*
- *image.gif*
- *image.png*
- *sound.mp3*



```
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
        <title>
                ZSUT.Tine.Lab1
        </title>
        <link href="site.css" rel="stylesheet" type="text/css">
</head>

<body>
        <div class="page">
                <div class="header">
                        <div class="title">
                                <h1>
                                        ZSUT TINE Lab 1
                                </h1>
                        </div>

                        <div class="clear hideSkiplink">
                                 
                        </div>
                </div>

                <div class="main">
                        <h3>
                                Client Error 4xx
                        </h3>

                        <p>
                                The 4xx class of status code is intended for cases in which the client seems to have erred. Except when
                                responding to a HEAD request, the server SHOULD include an entity containing an explanation of the error
                                situation, and whether it is a temporary or permanent condition. These status codes are applicable to
                                any
                                request method. User agents SHOULD display any included entity to the user.
                        </p>

                        <p>
                                If the client is sending data, a server implementation using TCP SHOULD be careful to ensure that the
                                client
                                acknowledges receipt of the packet(s) containing the response, before the server closes the input
                                connection.
                                If the client continues sending data to the server after the close, the server's TCP stack will send a
                                reset
                                packet to the client, which may erase the client's unacknowledged input buffers before they can be read
                                and
                                interpreted by the HTTP application.
                        </p>

                        <h3>
                                400 Bad Request
                        </h3>

                        <p>
                                The request could not be understood by the server due to malformed syntax. The client SHOULD NOT repeat
                                the
                                request without modifications.
                        </p>
                        <img src="image.gif">
                        <h3>
                                401 Unauthorized
                        </h3>

                        <p>
```

Rys. 7: Fragment kodu strony *index.html* z brakującymi multimediami

Zapisaliśmy je pod stosownymi nazwami, w folderze razem z plikiem *index.html*

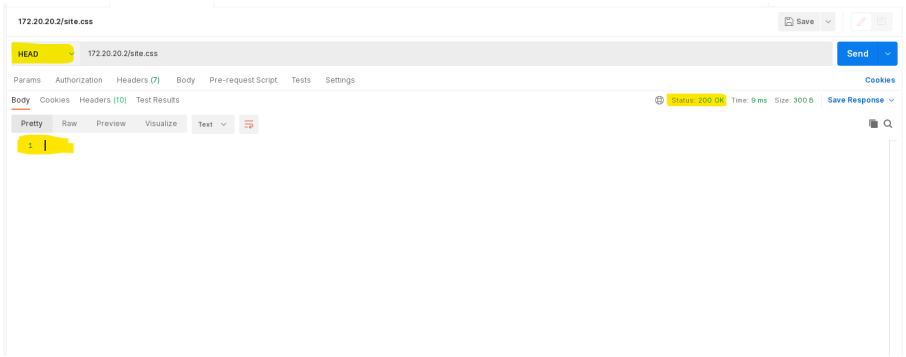
2.5. Nagłówek ETag

Przy pobieraniu pliku *site.css* zwróciliśmy uwagę na nagłówek **ETag**, którego użyjemy później.

Rys. 8: Wartość **ETag** pliku *site.css*

2.6. Zapytanie wykorzystujące metodę HEAD

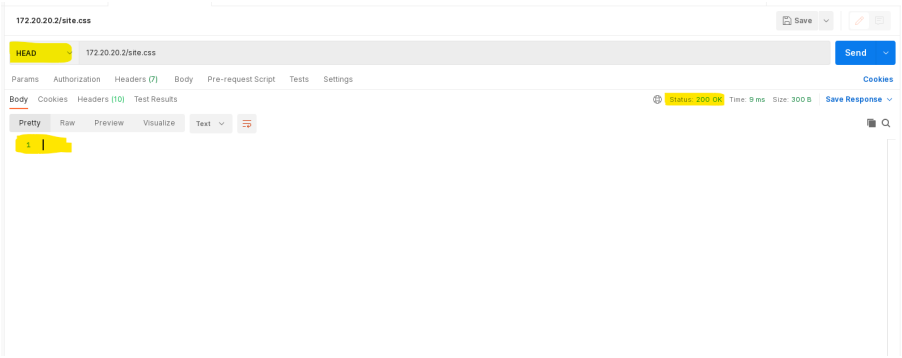
Następnie wysłaliśmy zapytanie typu HEAD odnośnie pliku *site.css*. Służy ono do pobierania informacji o zasobie zapisanych w nagłówkach (stąd nazwa - HEADers). W sekcji *headers* otrzymaliśmy więc identyczny rezultat jak na zrzucie nr 8. W sekcji *body* nie otrzymaliśmy jednak niczego, co jest zgodne z celem stosowania metody HEAD. Warto zauważyć, że status odpowiedzi to "200 OK".



Rys. 9: Wynik zapytania typu HEAD

2.7. Nagłówek *If-None-Match*

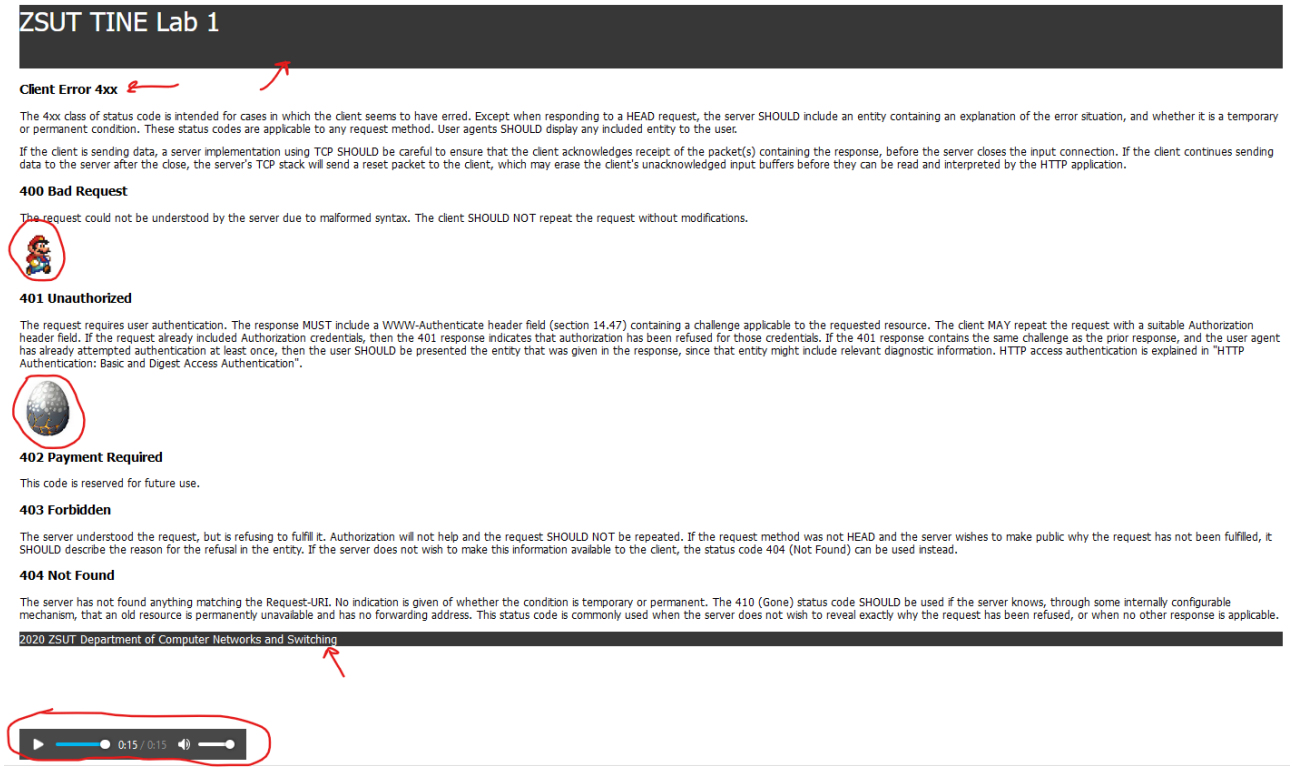
W kolejnym kroku ponownie wysłaliśmy zapytanie typu GET, tym razem dodając do niego nagłówek *If-None-Match* wraz z wartością *ETag* zdobytą w podrozdziale 2.5. Ten nagłówek służy do przekazywania informacji serwerowi oznaczającej, że klient posiada kopię zasobu identyfikowaną przez daną wartość *ETag*. Jeśli ta wartość pasuje do aktualnej wersji zasobu na serwerze, serwer zwróci odpowiedź z kodem stanu "304 Not Modified", informując klienta o tym, że zasób nie uległ zmianie i może użyć swojej lokalnej kopii. Jako, że plik *site.css* w istocie nie uległ zmianie na serwerze od czasu pobrania jego wartości *ETag*, otrzymaliśmy wiadomość zwrotną o statusie właśnie "304 Not Modified".



Rys. 10: Wynik zapytania typu GET z nagłówkiem *If-None-Match*

2.8. Finalny obraz strony

Po umieszczeniu wszystkego w jednym folderze i zmodyfikowaniu pliku *index.html* oraz odpaleniu go w przeglądarce naszym oczom ukazuje się finalny obraz strony z całą jego zawartością zarówno wizualną jak i dzwiękową.



Rys. 11: Finalny wygląd stony

3. Podsumowanie

Na podstawie wykonywanych zadań laboratorium, możemy sformułować następujące wnioski:

- Aplikacja Postman to potężne narzędzie do testowania API, które pozwoliło nam na bezpośrednie pobranie pliku index.html z serwera, a także na analizę odpowiedzi serwera, co jest kluczowe dla zrozumienia, jakie nagłówki są wymagane do prawidłowego pobrania obiektów z serwera.

- Przy pomocy narzędzi takich jak Wireshark, byliśmy w stanie analizować komunikację między klientem a serwerem na poziomie pakietów sieciowych, co pozwoliło nam lepiej zrozumieć procesy zachodzące podczas wymiany danych w protokołach TCP/IP.
- W trakcie laboratorium zrozumieliśmy także, jak ważne jest prawidłowe konstruowanie żądań HTTP, w tym użycie odpowiednich nagłówków, które wpływają na to, jak serwer przetwarza nasze żądanie i jakie elementy nam zwraca.
- Dzięki badaniu kodu HTML strony mogliśmy zidentyfikować brakujące elementy (takie jak obrazy, dźwięki, filmy, CSS) i pobrać je za pomocą dodatkowych żądań GET. To pokazuje, jak złożona może być struktura strony internetowej i jak wiele różnych zasobów może być zaangażowanych w jej utworzenie.
- Przez manipulację nagłówkiem Etag, zrozumieliśmy, jak serwery wykorzystują te nagłówki do optymalizacji przesyłania danych i unikania niepotrzebnego przesyłania identycznych danych.
- Końcowe zadanie pokazało, jak zasoby strony internetowej mogą być skonfigurowane do ładowania z lokalnych ścieżek, co pozwoliło na wyświetlanie strony offline. To podkreśla uniwersalność technologii HTML i jej zdolność do pracy zarówno z zasobami sieciowymi, jak i lokalnymi.